

RICE UNIVERSITY

**A Comparison of Transcription Techniques for the Optimal
Control of the International Space Station**

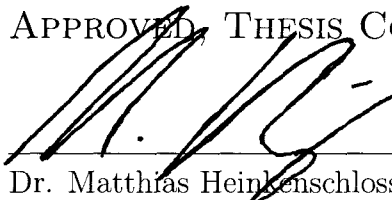
by

Jonah Reeger

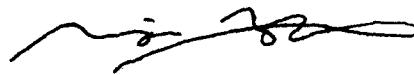
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Arts

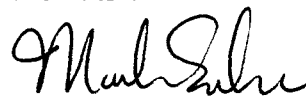
APPROVED, THESIS COMMITTEE:



Dr. Matthias Heinkenschloss, Chairman
Professor of Computational and Applied
Mathematics



Dr. Yin Zhang
Professor of Computational and Applied
Mathematics



Dr. Mark Embree
Associate Professor of Computational and
Applied Mathematics



Dr. Nazareth Bedrossian
Group Lead, Manned Space Systems,
Charles Stark Draper Laboratory, Inc.

HOUSTON, TEXAS

APRIL, 2009

UMI Number: 1485793

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1485793

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Abstract

A Comparison of Transcription Techniques for the Optimal Control of the International Space Station

by

Jonah Reeger

The numerical solution of Optimal Control Problems has received much attention over the past few decades. In particular, direct transcription methods have been studied because of their convergence properties for even relatively poor guesses at the solution. This thesis explores two of these techniques—Legendre-Gauss-Lobatto (LGL) Pseudospectral (PS) Collocation and Multiple Shooting (MS), and draws distinct comparisons between the two, allowing the reader to decide which would be better applied to a particular application. Specifically, comparisons will be made on accuracy, computation time, adjoint estimation, and storage requirements. It will be shown that the most distinct advantage for LGL PS collocation and MS methods will lie in computation time and storage requirements, respectively, using a nonlinear interior-point method described in this thesis.

Acknowledgements

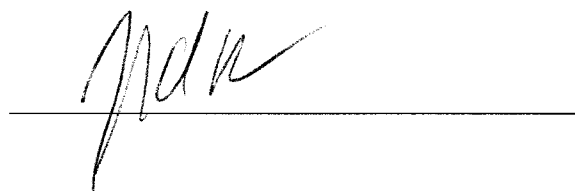
First, I would like to thank Professor Matthias Heinkenschloss, you have certainly assured my success during my time here at Rice University and provided more help in this research than I could have ever asked for. Next, I would like to thank Dr. Nazareth Bedrossian and Sagar Bhatt for your help, support, and motivation at the lab. Dr. Embree and Dr. Zhang, thank you for your advice and time. To the rest of the DLFs in Houston, it has been fun. Daria and the faculty, staff, and students in the CAAM department, thank you. Finally, I would like to thank my wife (name omitted due to Air Force policy) for her love and support, and for making this journey with me deep into the heart of Texas.

Acknowledgement

April 2009

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under NASA contract NNJ06HC37C.

Publication of this this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

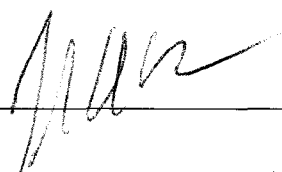
A handwritten signature in black ink, appearing to be 'J. D. R.', is written over a horizontal line.

(author's signature)

Assignment

Draper Laboratory Report Number T-1634.

In consideration for the research opportunity and permission to prepare my thesis by and at The Charles Stark Draper Laboratory, Inc., I hereby assign my copyright of the thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts.

A handwritten signature in dark ink, appearing to be 'J. W. R.', written over a horizontal line.

(author's signature)

9 April 2009

(date)

Contents

Abstract	iii
Acknowledgements	iv
Acknowledgement	v
Assignment	vi
List of Figures	xii
List of Tables	xv
1 Introduction	1
1.1 Existing Work	2
1.2 Organization of the Thesis	6
1.3 Notation	7
2 The Optimal Control Problem	9
2.1 General Form of the Optimal Control Problem	9

2.2	Optimality Conditions	11
2.3	Direct Solution of Optimal Control Problems	13
3	The Optimization Algorithm	16
3.1	Introduction	16
3.2	Interior Point Methods for Nonlinear Programs	17
3.3	Step Size Selection	21
3.3.1	Fraction to the Boundary Rule	21
3.3.2	An l_1 Merit Function and the Penalty Parameter	22
3.3.3	Step Acceptance and the Second-Order Correction	24
3.3.4	Dealing With Small Steps	26
3.4	Feasibility Restoration	26
3.5	Initialization of Primal and Dual Variables	30
4	Direct Transcription using Pseudospectral Collocation	32
4.1	Introduction	32
4.2	Problem Formulation	33
4.3	Legendre-Gauss-Lobatto (LGL) Pseudospectral Collocation	35
4.4	Well-posedness and Convergence of the LGL PS Collocation	37

4.5	Adjoint Estimation Properties for Pseudospectral Collocation	40
4.6	Singularity of the LGL PS KKT Matrix	44
4.7	Structure and Sparsity of the KKT Systems	47
5	Direct Transcription using Multiple Shooting	51
5.1	Introduction	51
5.2	Problem Formulation	52
5.3	Derivative Computations	56
5.3.1	First Order Derivatives	56
5.3.2	Second Order Derivatives	59
5.4	Control Parameterizations	61
5.4.1	Piecewise Constant Control Parameterization	62
5.4.2	Piecewise Linear Control Parameterization	62
5.4.3	Piecewise Cubic Control Parameterization	63
5.5	Well-posedness and Convergence of the MS Transcription	63
5.6	Structure and Sparsity of the KKT Systems	65
6	Numerical Examples	68
6.1	Introduction	68
6.2	Linear-Quadratic Example	70
6.3	Piecewise-Constant Control Example	74

	x
6.4 A Final Example	80
7 A Practical Application: The ISS ZPM	86
7.1 Introduction	86
7.2 International Space Station Dynamics	87
7.3 ZPM Optimal Control Problem	91
7.4 ISS 90-degree Maneuver	91
7.5 Numerical Comparisons	107
7.5.1 Computation Time	108
7.5.2 A Measure of Accuracy	110
8 Conclusions	115
Bibliography	117
A Matlab Code Organization	126
A.1 User Defined Settings and Problem Specific Functions	127
A.2 Driver Files	131
A.3 Nonlinear Interior Point Method Function	132
A.4 NLP Objective Functions	134
A.5 NLP Constraint Functions	135
A.6 Fixed-Step Runge-Kutta 4 Integrator	136
A.7 ODE Functions for the MS Methods	137
A.8 NLP Hessian Functions	138

A.9 LHS/RHS functions for the MS Methods	139
A.10 Quasi-Newton Update to the Hessian	140
A.11 Feasibility Restoration	140
A.12 NLP Feasibility Hessian Functions	142
A.13 Miscellaneous Functions for LGL PS Method	143

List of Figures

4.1	Sparsity pattern for LGL PS KKT matrix	48
4.2	Lower and upper triangular matrices for factorization of LGL PS KKT matrix	49
4.3	Comparison of different orderings of the PS KKT matrix	50
5.1	Sparsity pattern for MS KKT matrix	66
5.2	Lower and upper triangular matrices for factorization of MS KKT matrix	67
5.3	Comparison of storage requirements for LGL PS collocation and MS constant, linear, and cubic transcriptions	67
6.1	Optimal state for example (6.5)	71
6.2	State error for LGL PS collocation and MS (constant) transcription .	71
6.3	Optimal control for example (6.5)	72
6.4	Control error for LGL PS collocation and MS (constant) transcription	72
6.5	Optimal adjoint for example (6.5)	73
6.6	Adjoint error for LGL PS collocation and MS (constant) transcription	73

6.7	Optimal state for example (6.7)	76
6.8	State error for LGL PS collocation and MS (constant) transcription .	76
6.9	Optimal control for example (6.7)	77
6.10	Control error for LGL PS collocation and MS (constant) transcription	77
6.11	Optimal adjoint for example (6.7)	78
6.12	Adjoint error for LGL PS collocation and MS (constant) transcription	78
6.13	Optimal multiplier ($\mu_1(t)$) for example (6.7)	79
6.14	Multiplier ($\mu_1(t)$) error for LGL PS collocation and MS (constant) transcription	79
6.15	Optimal state ($x_2(t)$) for example (6.9)	82
6.16	State ($x_2(t)$) error for LGL PS collocation and MS (constant) tran- scription	82
6.17	Optimal control for example (6.9)	83
6.18	Control error for LGL PS collocation and MS (constant) transcription	83
6.19	Optimal adjoint ($\lambda_1(t)$) for example (6.9)	84
6.20	Adjoint ($\lambda_1(t)$) error for LGL PS collocation and MS (constant) tran- scription	84
6.21	Optimal multiplier ($\mu_1(t)$) for example (6.9)	85
6.22	Multiplier ($\mu_1(t)$) error for LGL PS collocation and MS (constant) transcription	85
7.1	ISS body reference frame	90

7.2	ISS LVLH reference frame	90
7.3	90-deg ZPM CMG momentum magnitude	94
7.4	90-deg ZPM CMG momentum (roll axis)	95
7.5	90-deg ZPM CMG momentum (pitch axis)	96
7.6	90-deg ZPM CMG momentum (yaw axis)	97
7.7	90-deg ZPM rotational rate (roll axis)	98
7.8	90-deg ZPM rotational rate (pitch axis)	99
7.9	90-deg ZPM rotational rate (yaw axis)	100
7.10	90-deg ZPM YPR Euler angles (roll axis)	101
7.11	90-deg ZPM YPR Euler angles (pitch axis)	102
7.12	90-deg ZPM YPR Euler angles (yaw axis)	103
7.13	90-deg ZPM CMG torque (roll axis)	104
7.14	90-deg ZPM CMG torque (pitch axis)	105
7.15	90-deg ZPM CMG torque (yaw axis)	106
7.16	CPU time required to evaluate objective and constraint functions . .	109
7.17	Rotational rate error in LGL PS and MS (cubic) methods	112
7.18	CMG momentum error in LGL PS and MS (cubic) methods	113
7.19	YPR Euler angle error in LGL PS and MS (cubic) methods	114
A.1	Directory structure for the code developed for this thesis	127

List of Tables

7.1	Initial conditions for 90-deg ZPM	92
7.2	Final conditions for 90-deg ZPM	92
7.3	Performance of the optimization algorithm on 90-deg ZPM	93

Chapter 1

Introduction

A relatively young topic in applied mathematics, particularly in the area of numerical optimization, is the direct numerical solution of Optimal Control Problems (OCP) governed by ordinary differential equations. This field of study is rich in application, and one of these applications appearing in the aerospace industry is explored in the numerical study of this thesis. The wealth of applications has led to many contributions in this field even over the past couple of decades, to include the development of efficient transcription techniques that convert a continuous OCP into a problem that can be handled by the various nonlinear programming techniques that are available. Of the many transcription techniques, two have been studied in the research of this thesis; the first because of its use in the particular aerospace application that was studied, and the second as an alternative to the first.

This research was prompted by the need to generate solutions to a particular

optimal control problem, dubbed the Zero Propellant Maneuver (ZPM), quickly, with limited memory, and accurately. The ZPM is a concept that substitutes reaction control systems, like thruster firing, with momentum storage devices, i.e. Control Moment Gyroscopes (CMGs), to control the attitude of a spacecraft without the use of expensive propellant [6, Section 1.1]. The attitude control performed by the ZPM is completed by exchanging the angular momentum created by rotating the CMGs and the momentum introduced by external disturbance torques, like aerodynamic torque and gravity gradient torque [6, Section 1.1]. This concept can be stated as an OCP, formulated in general in Chapter 2, which transitions a spacecraft from an initial orientation to a final orientation in a fixed amount of time without exceeding a constraint on the CMG momentum. The OCP is governed by a set of equations of motion that describe the change of attitude, rate, and CMG momentum in time, and it seeks to achieve the optimal value of some predetermined objective.

The ZPM was defined and successfully tested as a method of control for the International Space Station (ISS) (see, for example, [3]). However, the ZPM can be applied to any spacecraft equipped with CMGs and an appropriate controller.

1.1 Existing Work

The solution of optimal control problems has been the subject of a significant amount of study since the Seventeenth Century when Newton applied the calculus of variations to the Brachistochrone problem [9, Page 405]. The general numerical methods

that have been applied to generate optimal controls of dynamic systems were categorized and described by Betts in the 1990s [5]. Betts describes the two most general approaches to solving these problems as indirect—the application of calculus of variations to find the necessary conditions of an optimal control and the numerical solution of the resulting two point boundary value problem (TPBVP), and direct—application of a discretization technique to the states and controls or controls alone and the solution of the resulting nonlinear program (NLP). In particular, Betts introduces collocation and multiple shooting as methods for solving both the TPBVP of the indirect approach and the NLP of the direct approach [5, Section VI]. This thesis takes the direct approach to solve optimal control problems of a general form through two particular techniques. These are Legendre-Gauss-Lobatto (LGL) Pseudospectral (PS) collocation and multiple shooting (MS).

LGL PS collocation essentially approximates the states and controls of the problem using global basis functions that can be differentiated exactly to approximate the time derivative. However, these global basis functions introduce an explicit global dependence between the states and controls across the entire interval of interest. This is not necessarily a problem when adequate memory is available, but poses a problem when memory is limited since the optimization often requires storage of large, dense matrices. The advantageous properties of PS collocation are highlighted by many. Particularly, LGL PS collocation has been studied extensively by Fahroo and Ross in [14, 15, 16, 47] who highlighted the connection between the costates

of the indirect problem and the multipliers of the direct problem and developed a transcription package known as DIDO [15]. Gong, et al., [21] provide results on the convergence of the solution of the discretized OCP using LGL PS collocation to a solution of the original OCP.

Likewise, MS transcription parameterizes only the controls of the problem using local basis functions, splits the entire interval of interest into several subintervals, and solves a series of initial value problems while introducing some matching conditions between consecutive subintervals. The introduction of this local basis instead of a global basis leads to only an implicit dependence between the states and controls across the entire interval of interest. This translates into storage of large, sparse matrices that often require an order of magnitude less in storage space over those stored for PS collocation. Morrison, et. al, are credited with some of the earliest application of MS to OCPs as an alternative to Finite Differencing and single shooting [39]. Bock and Plitt [8] introduced a recursive quadratic programming technique, tailored to MS transcription, for solving optimal control problems. MS transcription has also been shown to share many of the desirable properties that LGL PS collocation exhibits. For instance, Grimm and Markl [24] showed the consistencies between the costates of the continuous problem and the multipliers of the discrete problem discretized by MS transcription.

This thesis will make some distinct comparisons between these two transcription techniques, that, until now, have almost always appeared only as alternatives to

one another without direct comparison. These comparisons will be made generally on the ability to approximate the adjoints of the continuous problem, the storage requirements to compute an optimal solution to the OCP using an Interior Point method, and the computation time to evaluate the constraint functions of the resulting nonlinear programs (NLPs). Further, the total computation time and a measure of accuracy will be compared for a particular example. This example is a ZPM taken from Bhatt [6].

The ZPM has been developed and modified over the past ten years by several individuals. First, McCants [37] showed the validity of solving an unconstrained optimal control problem governed by the attitude dynamics of the ZPM. Next, Pietz [42] employed DIDO [15] to solve a constrained optimal control problem that decreased the momentum state of the CMGs from a nonzero value to a zero value without exceeding a constraint on the norm of the momentum; this essentially was the first type of ZPM. Pietz further highlighted the consistencies between the costates of the continuous problem and the multipliers of the nonlinear program. Following Pietz, Bhatt [6] successfully developed maneuvers for a 90-degree reorientation of the ISS that were implemented and verified during an actual flight test in November 2006. Finally, in an effort to make available trajectories applicable to perturbed initial conditions and uncertainties in the dynamic model of the ISS, Roady [46] applied the idea of Neighboring Optimal Control (NOC), a one-step Newton-type correction to an existing trajectory, to the ZPM and drew a comparison to a Linear Model Predictive

Control (LMPC) algorithm.

1.2 Organization of the Thesis

Following this introductory chapter, this thesis proceeds in the following order.

Chapter 2 presents the general optimal control problem formulation of this thesis along with the optimality conditions for both the indirect approach and the direct approach to solving the problem.

Chapter 3 describes an interior-point method for nonlinear programs that combines aspects of the interior-point methods given by Waltz, et al., [56] and Wächter and Biegler [55]. This chapter first provides some of the theoretical development of interior point methods, and then explains step-size calculations and an attempt at increasing the robustness of the algorithm through a feasibility restoration.

Chapter 4 introduces Pseudospectral Collocation in general, and also describes the particular PS formulation that is used in the development of the results of this thesis; that is, Legendre-Gauss-Lobatto Pseudospectral Collocation. Then, the consistency of the direct approach is discussed relative to the indirect approach, so that the adjoint estimates for the LGL PS method can be developed. A singularity result that prohibits the use of LGL PS methods for certain problems is also given, and, finally, the structure of the linear system of equations that must be solved throughout the optimization procedure is given.

Chapter 5 parallels the development in Chapter 4, but for the Multiple Shooting

method. The general problem formulation is given, followed by the derivative computations that are required for this method. Three control parameterizations that have been tried in the development of the results of this thesis are also presented. Again, consistency of the direct problem relative to the indirect problem is given so that estimates of the adjoints can be cited. Finally, the structure of the linear system of equations is given.

Chapter 6 highlights some of the merits of the two transcription techniques on some examples that can be found in the literature. These examples illustrate the effects of smoothness on obtaining an accurate numerical solution.

Chapter 7 presents a numerical example that has practical implications. This chapter explores the 90-degree Zero-Propellant-Maneuver given by Bhatt [6], and highlights the performance of the optimization algorithm of Chapter 3 relative to the two transcription techniques given in Chapters 4 and 5 in a few areas. Some general conclusions are made both relative to this optimization algorithm and excluding the algorithm in terms of computation time and a specific measure of accuracy.

Finally, Chapter 8 presents some concluding remarks about the content of this thesis.

1.3 Notation

First, throughout this thesis \vec{e} will represent a vector of 1s of appropriate size, whereas the notation $\exp(\cdot)$ represents the exponential function with the argument as its ex-

ponent and e itself will be a function specified in the next chapter. Second, bold-face variables represent transcribed, or discretized, variables. Finally, due to discrepancies between the definitions in the works referenced by this thesis, three important notations denoting the partial derivatives of scalar and vector valued functions with respect to a vector of variables need to be defined. Consider $x \in \mathbb{R}^n$ and define the function $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$, then the gradient, $\nabla_x f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^n$, is the vector

$$\nabla_x f(x) = \left[\begin{array}{cccc} \frac{\partial}{\partial x_1} f(x) & \frac{\partial}{\partial x_2} f(x) & \cdots & \frac{\partial}{\partial x_n} f(x) \end{array} \right]^T,$$

where x_i , $i = 1, \dots, n$ is the i^{th} element of x , and the Hessian, $\nabla_{xx}^2 f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$, is the matrix

$$\nabla_{xx}^2 f(x) = \left[\begin{array}{cccc} \frac{\partial^2}{\partial x_1^2} f(x) & \frac{\partial^2}{\partial x_2 \partial x_1} f(x) & \cdots & \frac{\partial^2}{\partial x_n \partial x_1} f(x) \\ \frac{\partial^2}{\partial x_1 \partial x_2} f(x) & \frac{\partial^2}{\partial x_2^2} f(x) & \cdots & \frac{\partial^2}{\partial x_n \partial x_2} f(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_1 \partial x_n} f(x) & \frac{\partial^2}{\partial x_2 \partial x_n} f(x) & \cdots & \frac{\partial^2}{\partial x_n^2} f(x) \end{array} \right].$$

Now redefine $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$, then the Jacobian $D_x f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times n}$ is the matrix

$$D_x f(x) = \left[\begin{array}{cccc} \frac{\partial}{\partial x_1} f_1(x) & \frac{\partial}{\partial x_2} f_1(x) & \cdots & \frac{\partial}{\partial x_n} f_1(x) \\ \frac{\partial}{\partial x_1} f_2(x) & \frac{\partial}{\partial x_2} f_2(x) & \cdots & \frac{\partial}{\partial x_n} f_2(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} f_m(x) & \frac{\partial}{\partial x_2} f_m(x) & \cdots & \frac{\partial}{\partial x_n} f_m(x) \end{array} \right],$$

where $f_i(x)$, $i = 1, \dots, m$, is the i^{th} row of $f(x)$.

Chapter 2

The Optimal Control Problem

2.1 General Form of the Optimal Control Problem

This thesis considers optimal control problems with state and control variables

$$x : \mathbb{R} \mapsto \mathbb{R}^n \text{ and } u : \mathbb{R} \mapsto \mathbb{R}^m,$$

respectively. Given functions

$$l : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \mapsto \mathbb{R}, \quad (\text{integral objective})$$

$$e : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}, \quad (\text{final time objective})$$

$$f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \mapsto \mathbb{R}^n, \quad (\text{dynamics})$$

$$g : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^{n_g}, \quad (\text{state/control constraints})$$

$$a : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^{n_a}, \text{ and} \quad (\text{initial conditions})$$

$$b : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^{n_b}, \quad (\text{final conditions})$$

which are all assumed to be twice continuously differentiable with respect to x , u , and t , the general form of the optimal control problem under consideration is

$$\min \int_{t_0}^{t_f} l(x(t), u(t), t) dt + e(x(t_f), t_f) \quad (2.1a)$$

s.t.

$$\dot{x}(t) = f(x(t), u(t), t), \quad t \in [t_0, t_f] \quad (2.1b)$$

$$a(x(t_0), t_0) = 0 \quad (2.1c)$$

$$b(x(t_f), t_f) = 0 \quad (2.1d)$$

$$g(x(t), u(t), t) \leq 0, \quad t \in [t_0, t_f]. \quad (2.1e)$$

The OCP seeks states $x \in W^{1,\infty}([t_0, t_f], \mathbb{R}^n)$ and controls $u \in L^\infty([t_0, t_f], \mathbb{R}^m)$ that are local solutions of (2.1) [19, Section 2].

In the literature there are two main approaches to solving problems of various forms similar to (2.1). Some of the earliest approaches involved the development of optimality conditions by employing calculus of variations. This is often known as the indirect approach. Perhaps the most recognized theory resulting from this approach is that due to Pontryagin, et al. [44]. The optimality conditions lead a two point boundary value problem (TPBVP). This TPBVP is often too difficult, or impossible, to solve analytically. Instead, these TPBVP are usually solved using various numerical techniques. See for example the article by von Stryk and Bulirsch [52].

The other popular approach to solving problems similar to (2.1), often dubbed

the direct approach, consists of discretizing the dynamics and the objective function. Discretization of the optimal control problem (2.1) leads to a nonlinear program for the discretized states and controls. This thesis studies two direct approaches, a pseudospectral discretization (see Chapter 4) and a multiple shooting discretization (see Chapter 5) of (2.1).

2.2 Optimality Conditions

Solution of a problem of type (2.1) requires satisfaction of a set of optimality conditions that are often derived through various applications of Pontryagin's Minimum Principle.

First order necessary conditions for similar problems are presented in many resources. For instance, if the constraint on the states at time t_0 is fixed in the sense that

$$a(x(t_0), t_0) = x(t_0) - x_0$$

for some vector $x_0 \in \mathbb{R}^n$, then Hartl, Sethi, and Vickson [27] provide various formal and informal theorems outlining these necessary conditions. The necessary conditions presented here are those given by Göllman, Kern, and Maurer [19], which are adapted from a similar formulation to (2.1). Note also, for the purposes of this thesis all of the constraints $g(x(t), u(t), t)$ are considered to be explicitly dependent on $u(t)$ for the development of the adjoint estimates presented in chapters 3 and 4.

Let

$$\mathcal{G}^*(t) = \{j : g_j(x^*(t), u^*(t), t) = 0\}$$

denote the set of active inequality constraints at a locally optimal state and control pair (x^*, u^*) . The following regularity assumptions must be satisfied at (x^*, u^*)

$$\text{rank} \begin{bmatrix} D_x a(x^*(t_0), t) \\ D_x b(x^*(t_f), t) \end{bmatrix} = n_a + n_b \quad (2.2)$$

and

$$\text{rank} \left[\left(\frac{\partial}{\partial u} g_j(x^*(t), u^*(t), t) \right)_{j \in \mathcal{G}^*(t)} \right] = |\mathcal{G}^*(t)|. \quad (2.3)$$

Define the Hamiltonian

$$\begin{aligned} H(x(t), u(t), \tilde{\lambda}(t), \tilde{\mu}(t), t) &= l(x(t), u(t), t) + \tilde{\lambda}(t)^T f(x(t), u(t), t) \\ &\quad + \tilde{\mu}(t)^T g(x(t), u(t), t) \end{aligned}$$

and the control region

$$\Omega(x(t), t) = \{u \in \mathbb{R}^m : g(x(t), u, t) \leq 0\}.$$

Theorem 2.2.1 *Let $\{x^*, u^*\}$ be a locally optimal pair for problem (2.1). Assume that the regularity conditions (2.2) and (2.3) are satisfied. Then there exist a costate (adjoint) function $\tilde{\lambda}^* \in W^{1,\infty}([t_0, t_f], \mathbb{R}^n)$, a multiplier function $\tilde{\mu}^* \in L^\infty([a, b], \mathbb{R}^{n_g})$, and multipliers $\tilde{\nu}_0^* \in \mathbb{R}^{n_a}$ and $\tilde{\nu}_f^* \in \mathbb{R}^{n_b}$, such that the following conditions hold for all $t \in [t_0, t_f]$:*

- (i) *adjoint differential equation:*

$$\dot{\tilde{\lambda}}^*(t)^T = -\nabla_x H(x^*(t), u^*(t), \tilde{\lambda}^*(t), \tilde{\mu}^*(t), t) \quad (2.4a)$$

- (ii) transversality conditions:

$$\tilde{\lambda}^*(t_0)^T = -(\nu_0^*)^T D_x a(x^*(t_0), t_0) \quad (2.4b)$$

$$\tilde{\lambda}^*(t_f)^T = (\nu_f^*)^T D_x b(x^*(t_f), t_f) + \nabla_x e(x^*(t_f), t_f) \quad (2.4c)$$

- (iii) minimum condition for the Hamiltonian function

$$0 = \nabla_u H(x^*(t), u^*(t), \tilde{\lambda}^*(t), \tilde{\mu}^*(t), t) \quad (2.4d)$$

- (iv) multiplier condition and complementarity

$$\tilde{\mu}^*(t) \geq 0 \text{ and } \tilde{\mu}_i^*(t) g_i(x^*(t), u^*(t), t) = 0, \quad i = 1, \dots, n_g. \quad (2.4e)$$

One can even show that the Pontryagin minimum principle

$$H(x^*(t), u^*(t), \tilde{\lambda}^*(t), \tilde{\mu}^*(t), t) \leq H(x^*(t), u, \tilde{\lambda}^*(t), \tilde{\mu}^*(t), t), \quad \forall u \in \Omega(x^*(t), t)$$

is valid. These conditions describe the TPBVP that must be solved to find a local minimum of the objective functional with respect to the dynamic, path and boundary constraints.

2.3 Direct Solution of Optimal Control Problems

Given a transcription technique, two of which will be further explained in the coming chapters, the optimal control problem (2.1) can be transformed into a nonlinear

program (NLP) of the form

$$\min J(\mathbf{y}) \quad (2.5a)$$

s.t.

$$F(\mathbf{y}) = 0 \quad (2.5b)$$

$$G(\mathbf{y}) \leq 0, \quad (2.5c)$$

where $\mathbf{y} \in \mathbb{R}^{n_y}$, $J : \mathbb{R}^{n_y} \mapsto \mathbb{R}$, $F : \mathbb{R}^{n_y} \mapsto \mathbb{R}^{n_F}$, and $G : \mathbb{R}^{n_y} \mapsto \mathbb{R}^{n_G}$. This is the general form for the nonlinear programs that will appear throughout this thesis. The Lagrangian associated with (2.5) is given by

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J(\mathbf{y}) + \boldsymbol{\lambda}^T F(\mathbf{y}) + \boldsymbol{\mu}^T G(\mathbf{y}).$$

The first order necessary optimality conditions for (2.5) are given as follows (see Nocedal and Wright [40]).

Recall the Mangasarian Fromovitz Constraint Qualification (MFCQ) [36]. Let $\mathcal{A}(\mathbf{y}) = \{i | G_i(\mathbf{y}) = 0\}$ be the active set at \mathbf{y} . The MFCQ holds at \mathbf{y} if there exists $d \in \mathbb{R}^n$ such that

$$\nabla G_i(\mathbf{y})^T d < 0, \quad i \in \mathcal{A}(\mathbf{y}^*)$$

$$\nabla F_i(\mathbf{y})^T d = 0, \quad i = 1, \dots, n_F$$

and if $\nabla F_1(\mathbf{y}), \dots, \nabla F_{n_F}(\mathbf{y}^*)$ are linearly independent.

Theorem 2.3.1 *Suppose that \mathbf{y}^* is a local solution of (2.5) and that the functions J , F , and G are continuously differentiable. If the MFCQ holds at \mathbf{y}^* , then there are*

multipliers λ^ and μ^* , such that*

$$\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{y}^*, \lambda^*, \mu^*) = 0 \quad (2.6a)$$

$$F(\mathbf{y}^*) = 0 \quad (2.6b)$$

$$G(\mathbf{y}^*) \leq 0 \quad (2.6c)$$

$$\mu^* \geq 0 \quad (2.6d)$$

$$(\mu^*)^T G(\mathbf{y}^*) = 0 \quad (2.6e)$$

These necessary conditions are often referred to as the Karush-Kuhn-Tucker (KKT) conditions.

Numerical techniques for solving nonlinear programs are employed. This thesis will adopt an interior-point/barrier algorithm to solve the NLPs that arise from the discretization of optimal control problems.

Chapter 3

The Optimization Algorithm

3.1 Introduction

In order to compare, and truly understand, the two transcription techniques studied in this thesis, and to take advantage of some of the properties of the two techniques, an optimization algorithm was implemented in the MATLAB environment. This algorithm combines several aspects of the interior-point algorithms described by Waltz, et al., [56] and Wächter and Biegler [55]. The interior-point algorithms described in these papers are not the only methods available to solve nonlinear programs like (2.5). For instance, active-set methods or any sequential quadratic programming solver, like those described in Nocedal and Wright [40], could also be employed. Interior-point methods were chosen because of the favorable convergence properties, compared to active-set methods, they exhibited when experimenting with the optimization algo-

rithms available with MATLABs `fmincon` to solve ZPM OCPs. The development of this algorithm also stands as a step toward creating an optimization algorithm efficient and compact enough to be fieldable for ISS applications.

3.2 Interior Point Methods for Nonlinear Programs

According to Nocedal and Wright, [40, Chapter 19], to solve the nonlinear program (2.5) with inequality constraint $G(\mathbf{y}) \leq 0$, a vector of nonnegative slack variables $\mathbf{z} \in \mathbb{R}^{n_G}$ can be introduced and the problem can then be transformed to

$$\begin{aligned} \min_{\mathbf{y}, \mathbf{z}} \quad & J(\mathbf{y}) \\ \text{s.t.} \quad & F(\mathbf{y}) = 0 \\ & G(\mathbf{y}) + \mathbf{z} = 0 \\ & \mathbf{z} \geq 0. \end{aligned} \tag{3.1}$$

This transformation still requires the handling of the inequality $\mathbf{z} \geq 0$, which is often done by adding a barrier term to the objective $J(\mathbf{y})$. For instance, to handle the inequality, the term $-\eta \sum_{i=1}^{n_G} \ln(\mathbf{z}_i)$ in the following nonlinear program serves as a way of penalizing slack variables that become too small because of the natural

logarithm's asymptotic behavior near zero [40, Section 19.1]:

$$\begin{aligned} \min_{\mathbf{y}, \mathbf{z}} \quad & J(\mathbf{y}) - \eta \sum_{i=1}^{n_G} \ln(\mathbf{z}_i) \\ \text{s.t.} \quad & F(\mathbf{y}) = 0 \\ & G(\mathbf{y}) + \mathbf{z} = 0. \end{aligned} \tag{3.2}$$

The parameter η is known as the barrier parameter, and as $\eta \rightarrow 0$ the solution to (3.2) approaches the solution of (2.5).

Derivation of the first-order necessary conditions for the problem (3.2) for a fixed barrier parameter leads to the system of equations

$$\begin{aligned} \nabla_{\mathbf{y}} \mathcal{L}(\mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= 0 \\ -\eta Z^{-1} \vec{e} + \boldsymbol{\mu} &= 0 \\ F(\mathbf{y}) &= 0 \\ G(\mathbf{y}) + \mathbf{z} &= 0 \end{aligned} \tag{3.3}$$

and Lagrangian

$$\mathcal{L}(\mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J(\mathbf{y}) - \eta \sum_{i=1}^{n_G} \ln(\mathbf{z}_i^k) + F(\mathbf{y})^T \boldsymbol{\lambda} + [G(\mathbf{y}) + \mathbf{z}]^T \boldsymbol{\mu}.$$

An iterative Newton type approach can be taken to satisfy (3.3) by linearizing these constraints around the current iterate. Throughout this chapter, define the current iteration of the algorithm to be k , the primal variables at the current iteration to be \mathbf{y}^k and \mathbf{z}^k , and the dual variables at the current iteration to be $\boldsymbol{\lambda}^k$ and $\boldsymbol{\mu}^k$. Also, let the superscript k on a function denote the function evaluated at the current iterate.

The resulting KKT system for the interior point problem is

$$\begin{bmatrix} \nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}^k & 0 & (D_{\mathbf{y}} F^k)^T & (D_{\mathbf{y}} G^k)^T \\ 0 & M^k (Z^k)^{-1} & 0 & I \\ D_{\mathbf{y}} F^k & 0 & 0 & 0 \\ D_{\mathbf{y}} G^k & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta \mathbf{y}^k \\ \delta \mathbf{z}^k \\ \delta \boldsymbol{\lambda}^k \\ \delta \boldsymbol{\mu}^k \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{y}} \mathcal{L}^k \\ -\eta^k (Z^k)^{-1} \vec{e} + \boldsymbol{\mu}^k \\ F^k \\ G^k + \mathbf{z}^k \end{bmatrix}$$

where $Z^k = \text{diag}(\mathbf{z}^k)$ and $M^k = \text{diag}(\boldsymbol{\mu}^k)$. In the way of Nocedal and Wright [40, Section 19.3] this system of equations can be reduced using the fourth and second equations to eliminate $\delta \mathbf{z}^k$ and $\delta \boldsymbol{\mu}^k$, respectively. Notice that

$$\delta \mathbf{z}^k = - (D_{\mathbf{y}} G^k \delta \mathbf{y}^k + G^k + \mathbf{z}^k)$$

and

$$\delta \boldsymbol{\mu}^k = - (M^k (Z^k)^{-1} \delta \mathbf{z}^k - \eta^k (Z^k)^{-1} \vec{e} + \boldsymbol{\mu}^k).$$

Substituting these results into the first equation, the reduced KKT system has the matrix

$$\begin{bmatrix} \nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}^k + (D_{\mathbf{y}} G^k)^T M^k (Z^k)^{-1} D_{\mathbf{y}} G^k & (D_{\mathbf{y}} F^k)^T \\ D_{\mathbf{y}} F^k & 0 \end{bmatrix} \quad (3.4)$$

on the left hand side—also known as the KKT matrix, and right hand side

$$- \begin{bmatrix} \nabla_{\mathbf{y}} \mathcal{L}^k + (D_{\mathbf{y}} G^k)^T (Z^k)^{-1} [M^k G^k + \eta^k \vec{e}] \\ F^k \end{bmatrix}. \quad (3.5)$$

Once this system is solved for $\delta \mathbf{y}$ and $\delta \boldsymbol{\lambda}$ and the values of $\delta \mathbf{z}$ and $\delta \boldsymbol{\mu}$ have been

computed, the next iterate can be defined by

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k \quad (3.6a)$$

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \alpha_p^k \delta \mathbf{z}^k \quad (3.6b)$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \alpha_d^k \delta \boldsymbol{\lambda}^k \quad (3.6c)$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \alpha_d^k \delta \boldsymbol{\mu}^k \quad (3.6d)$$

for some step sizes α_p^k and α_d^k .

For a fixed barrier parameter, the iteration to solve (3.2) is continued until a set of termination criteria are met. Several different criteria have been tried, including those from Wächter and Biegler [55, Section 2.1], but, for consistency with some of the parameter choices, the algorithm terminates based on the criteria of Waltz, et al., [56, Section 3.5]. Define $(G^k)^+$ to be $(G_i^k)^+ = \max\{0, G_i^k\}$ for each row $i = 1, \dots, n_G$ of G^k . Then the termination criteria for the barrier problem are

$$\|\nabla_{\mathbf{y}} J^k + (D_{\mathbf{y}} F^k)^T \boldsymbol{\lambda}^k + (D_{\mathbf{y}} G^k)^T \boldsymbol{\mu}^k\|_{\infty} \leq \max\{1, \|\nabla_{\mathbf{y}} J^k\|_{\infty}\} \epsilon_{\eta}^{\text{opt}}, \quad (3.7a)$$

$$\|Z^k \boldsymbol{\mu}^k - \eta^k \bar{\mathbf{e}}\|_{\infty} \leq \max\{1, \|\nabla_{\mathbf{y}} J^k\|_{\infty}\} \epsilon_{\eta}^{\text{opt}}, \text{ and} \quad (3.7b)$$

$$\|[(F^k)^T \quad (G^k + \mathbf{z}^k)^T]^T\|_{\infty} \leq \max\{1, \|[(F^0)^T \quad ((G^0)^+)^T]^T\|_{\infty}\} \epsilon_{\eta}^{\text{feas}}, \quad (3.7c)$$

where $\epsilon_{\eta}^{\text{opt}} = \max\{\theta \eta^k, \epsilon^{\text{opt}} - \eta^k\}$ and $\epsilon_{\eta}^{\text{feas}} = \max\{\theta \eta^k, \epsilon^{\text{feas}}\}$ for parameter θ and predefined tolerances ϵ^{opt} and ϵ^{feas} (currently, $\theta = 1$ and $\epsilon^{\text{opt}} = \epsilon^{\text{feas}} = 10^{-6}$).

Once the termination criteria (3.7) are met, the barrier parameter is decreased by the following procedure taken from Waltz, et al., [56, Section 3.5]. First, η^0 is

initialized to be 0.1. Then, assuming that the current barrier problem was solved at iteration k , the barrier parameter is decreased by $\eta^{k+1} = \max\{\eta_{\min}, \eta^k/100\}$ if the barrier problem was solved in fewer than three iterations and $\eta^{k+1} = \max\{\eta_{\min}, \eta^k/5\}$ otherwise. The minimum barrier parameter $\eta_{\min} = \min\{\epsilon^{\text{opt}}, \epsilon^{\text{feas}}\}/100$ is chosen so that the barrier parameter does not decrease too much relative to the predefined tolerances ϵ^{opt} and ϵ^{feas} [56, Section 3.5].

Further, once a barrier problem is solved to meet (3.7), the termination criteria of the optimization algorithm are checked. These termination criteria are

$$\|\nabla_{\mathbf{y}} J^k + (D_{\mathbf{y}} F^k)^T \boldsymbol{\lambda}^k + (D_{\mathbf{y}} G^k)^T \boldsymbol{\mu}^k\|_{\infty} \leq \max\{1, \|\nabla_{\mathbf{y}} J^k\|_{\infty}\} \epsilon^{\text{opt}}, \quad (3.8a)$$

$$\|Z^k \boldsymbol{\mu}^k\|_{\infty} \leq \max\{1, \|\nabla_{\mathbf{y}} J^k\|_{\infty}\} \epsilon^{\text{opt}}, \text{ and} \quad (3.8b)$$

$$\|[(F^k)^T \quad ((G^k)^+)^T]^T\|_{\infty} \leq \max\{1, \|[(F^0)^T \quad ((G^0)^+)^T]^T\|_{\infty}\} \epsilon^{\text{feas}}, \quad (3.8c)$$

[56, Section 3.5].

3.3 Step Size Selection

3.3.1 Fraction to the Boundary Rule

To guarantee the nonnegativity of \mathbf{z}^{k+1} and $\boldsymbol{\mu}^{k+1}$ the step sizes α_p^k and α_d^k are chosen such that the length of the step does not allow the iterate to violate these constraints. This is done by enforcing a fraction to the boundary rule ([40, Section 19.2], [55,

Section 2.4], [56, Section 2]), defined by

$$\alpha_p^k = \max\{\alpha \in (0, 1] : \mathbf{z}^k + \alpha \delta \mathbf{z}^k \geq (1 - \tau^k) \mathbf{z}^k\}$$

$$\alpha_d^k = \max\{\alpha \in (0, 1] : \boldsymbol{\mu}^k + \alpha \delta \boldsymbol{\mu}^k \geq (1 - \tau^k) \boldsymbol{\mu}^k\},$$

where $\tau^k \in (0, 1]$ is typically a parameter close to 1. In the implementation of this thesis, $\tau^k = \max\{0.99, 1 - \eta^k\}$ so that $\tau^k \rightarrow 1$ as $\eta^k \rightarrow 0$ allowing \mathbf{z}^k and $\boldsymbol{\mu}^k$ to move closer to the boundary. This heuristic is implemented by Wächter and Biegler in their filter interior point code [55].

3.3.2 An l_1 Merit Function and the Penalty Parameter

After the fraction to the boundary rule is enforced, the step size is bisected using a backtracking line search starting from α_p^k and α_d^k . The backtracking line search can be completed in many ways. For instance, Wächter and Biegler [55] propose a filter line-search that only takes steps that are acceptable to some pre-defined filter.

This algorithm considers the merit function proposed by Waltz, et al., [56]. This merit function is defined by

$$\phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k) = J^k - \eta^k \sum_{i=1}^N \mathbf{z}_i^k + \nu^k \|F^k\|_1 + \nu^k \|G^k\|_1$$

with the strictly positive penalty parameter ν^k . This penalty parameter is chosen to coincide with the parameter in Waltz, et al. [56, Section 3.1]. Without explanation

the choice of ν^k is the following. Begin by letting

$$C(\mathbf{y}^k, \mathbf{z}^k) = \begin{bmatrix} F^k \\ G^k + \mathbf{z}^k \end{bmatrix}$$

and compute the trial penalty parameter ν_{trial} by

$$\nu_{trial} = \frac{\nabla_{\mathbf{y}} J^k \delta \mathbf{y}^k - \eta^k (Z^k)^{-1} \delta \mathbf{z}^k + \frac{\sigma}{2} [(\delta \mathbf{y}^k)^T \nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}^k \delta \mathbf{y}^k + (\delta \mathbf{z}^k)^T (Z^k)^{-1} M^k \delta \mathbf{z}^k]}{(1 - \rho) \|C^k\|}.$$

The parameter σ is chosen so that if the curvature of the Lagrangian with respect to \mathbf{y} and \mathbf{z} is not negative, then it is not considered in the computation of the penalty parameter, and ρ is chosen so that the reduction in the merit function is sufficient.

The computation of σ is as described below:

$$\sigma = \begin{cases} 1, & \text{if } (\delta \mathbf{y}^k)^T \nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}^k \delta \mathbf{y}^k + (\delta \mathbf{z}^k)^T (Z^k)^{-1} M^k \delta \mathbf{z}^k; \\ 0, & \text{otherwise.} \end{cases}$$

Finally, choose ν^{k+1} by

$$\nu^{k+1} = \begin{cases} \nu^k, & \text{if } \nu^k \geq \nu_{trial}; \\ \nu_{trial} + 1, & \text{otherwise.} \end{cases}$$

(Initially, $\nu^0 = 1$ in this algorithm.) The barrier function ϕ_{ν^k} has a directional derivative defined by

$$\mathcal{D}(\phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k); \delta \mathbf{y}^k, \delta \mathbf{z}^k) = D_{\mathbf{y}} J^k \delta \mathbf{y}^k - \eta^k (Z^k)^{-1} \delta \mathbf{z}^k - \nu^k \|F^k\|_1 - \nu^k \|G^k\|_1$$

in the direction of $\delta \mathbf{y}^k$ and $\delta \mathbf{z}^k$; this is a result that can be found in, for instance, Nocedal and Wright [40, Section 18.3]. Their results suggest that under certain conditions this direction will be a descent direction, and the choice of ν^k here attempts to satisfy those conditions.

3.3.3 Step Acceptance and the Second-Order Correction

With the merit function fully defined, the backtracking line search is completed by bisecting α_p^k and α_d^k until the Armijo condition, defined by

$$\phi_{\nu^k}(\mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k, \mathbf{z}^k + \alpha_d^k \delta \mathbf{z}^k) \leq \phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k) + c \alpha_p^k \mathcal{D}(\phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k); \delta \mathbf{y}^k, \delta \mathbf{z}^k),$$

is satisfied for some constant c ($c = 10^{-4}$ in this implementation). This Armijo condition suggests a sufficient decrease in the merit function before a step is acceptable (see, for example, Nocedal and Wright [40, Section 3.1]).

Line search algorithms that employ merit functions like $\phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k)$ suffer from a phenomenon where certain steps computed by solving the KKT system lead to larger values of the merit function and constraints, but would otherwise lead to quadratic convergence of the algorithm if accepted by the merit function [40, Section 15.5]. This rejection of steps is known as the Maratos effect. To overcome this obvious shortcoming of line search methods, a second-order correction can be computed and combined with the original step. At the first iteration of the line search procedure, the algorithm described here employs a second order correction when

$$J(\mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k) - \eta^k \sum_{i=1}^{n_G} \ln(\mathbf{z}_i^k + \alpha_p^k \delta \mathbf{z}_i^k) \leq J^k - \eta^k \sum_{i=1}^{n_G} \ln(\mathbf{z}_i^k)$$

and

$$\phi_{\nu^k}(\mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k, \mathbf{z}^k + \alpha_p^k \delta \mathbf{z}^k) > \phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k) + c \alpha_p^k \mathcal{D}(\phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k); \delta \mathbf{y}^k, \delta \mathbf{z}^k),$$

an idea proposed and implemented by Waltz, et al., [56, Section 3.2]. The second-order correction step, $\delta \mathbf{y}_{\text{soc}}^k$ and $\delta \mathbf{z}_{\text{soc}}^k$, is computed by modifying the vector of the

right hand side (3.5) to be

$$\begin{bmatrix} \nabla_{\mathbf{y}} J^k + (D_{\mathbf{y}} F^k)^T (\boldsymbol{\lambda}^k + \alpha_d^k \delta \boldsymbol{\lambda}^k) + \alpha_p^k \nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}^k \delta \mathbf{y}^k + (D_{\mathbf{y}} G^k)^T (Z^k)^{-1} (M^k G^k + \eta^k \bar{e}) \\ F(\mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k) \end{bmatrix}$$

and solving the system with the original KKT matrix (3.4) on the left hand side.

With $\delta \mathbf{y}_{\text{soc}}^k$ and $\delta \boldsymbol{\lambda}_{\text{soc}}^k$, $\delta \mathbf{z}_{\text{soc}}^k$ and $\delta \boldsymbol{\mu}_{\text{soc}}^k$ can then be computed by

$$\begin{aligned} \delta \mathbf{z}_{\text{soc}}^k &= -(D_{\mathbf{y}} G^k \delta \mathbf{y}_{\text{soc}}^k + G(\mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k) + (\mathbf{z}^k + \alpha_p^k \delta \mathbf{z}^k)) \text{ and} \\ \delta \boldsymbol{\mu}_{\text{soc}}^k &= \alpha_d^k \delta \boldsymbol{\mu}^k + ((Z^k)^{-1} M^k) (D_{\mathbf{y}} G^k \delta \mathbf{y}_{\text{soc}}^k + G(\mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k)) \\ &\quad + \eta^k (Z^k)^{-1} - (\boldsymbol{\mu}^k + \alpha_d^k \delta \boldsymbol{\mu}^k). \end{aligned}$$

Again, the fraction to the boundary rule is applied by

$$\begin{aligned} \alpha_{p,\text{soc}}^k &= \max\{\alpha \in (0, 1] : \mathbf{z}^k + \alpha(\alpha_p^k \delta \mathbf{z}^k + \delta \mathbf{z}_{\text{soc}}^k) \geq (1 - \tau^k) \mathbf{z}^k\} \\ \alpha_{d,\text{soc}}^k &= \max\{\alpha \in (0, 1] : \boldsymbol{\mu}^k + \alpha(\alpha_d^k \delta \boldsymbol{\mu}^k + \delta \boldsymbol{\mu}_{\text{soc}}^k) \geq (1 - \tau^k) \boldsymbol{\mu}^k\}, \end{aligned}$$

and the second order correction is accepted if

$$\begin{aligned} &\phi_{\nu^k}(\mathbf{y}^k + \alpha_{p,\text{soc}}^k (\alpha_p^k \delta \mathbf{y}^k + \delta \mathbf{y}_{\text{soc}}^k), \mathbf{z}^k + \alpha_{p,\text{soc}}^k (\alpha_p^k \delta \mathbf{z}^k + \delta \mathbf{z}_{\text{soc}}^k)) \\ &\leq \phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k) + c \alpha_p^k \mathcal{D}(\phi_{\nu^k}(\mathbf{y}^k, \mathbf{z}^k); \delta \mathbf{y}^k, \delta \mathbf{z}^k), \end{aligned}$$

and the line search is discontinued so that

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha_{p,\text{soc}}^k (\alpha_p^k \delta \mathbf{y}^k + \delta \mathbf{y}_{\text{soc}}^k), \quad (3.9a)$$

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \alpha_{p,\text{soc}}^k (\alpha_p^k \delta \mathbf{z}^k + \delta \mathbf{z}_{\text{soc}}^k), \quad (3.9b)$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \alpha_{d,\text{soc}}^k (\alpha_d^k \delta \boldsymbol{\lambda}^k + \delta \boldsymbol{\lambda}_{\text{soc}}^k), \text{ and} \quad (3.9c)$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \alpha_{d,\text{soc}}^k (\alpha_d^k \delta \boldsymbol{\mu}^k + \delta \boldsymbol{\mu}_{\text{soc}}^k). \quad (3.9d)$$

Otherwise, the second order correction is rejected and α_p^k and α_d^k are bisected with the regular line search procedure, where the iterates are updated by (3.6).

3.3.4 Dealing With Small Steps

The backtracking line search sometimes continues until the length of the step is too small to make any meaningful progress toward the solution. That is, when

$$\max\{\alpha_p^k \delta \mathbf{y}^k\} \leq \epsilon^y$$

($\epsilon^y = 10^{-12}$ in this implementation). In such a case, the optimization tries to find a \mathbf{y}^k that satisfies the constraints better. As a safeguard, the current implementation enters the feasibility restoration when

$$\alpha_p^k \leq \sqrt{\epsilon^y} \text{ and } \left\| \begin{bmatrix} (F^k)^T & ((G^k)^+)^T \end{bmatrix}^T \right\|_\infty \geq \sqrt{\epsilon^{\text{feas}}}.$$

Otherwise, if

$$\max\{\alpha_p^k \delta \mathbf{y}^k\} \leq \epsilon^y \text{ and } \left\| \begin{bmatrix} (F^k)^T & ((G^k)^+)^T \end{bmatrix}^T \right\|_\infty < \sqrt{\epsilon^{\text{feas}}}$$

the optimization terminates and a new guess for \mathbf{y}^k should be given to restart the optimization.

3.4 Feasibility Restoration

The feasibility restoration algorithm seeks a point \mathbf{y} such that the constraint error defined by

$$\max\{\|F(\mathbf{y})\|_\infty, \|(G(\mathbf{y}))^+\|_\infty\}$$

is decreased by some proportion (this proportion is half in this algorithm). Nonlinear programs of the form

$$\min_{\mathbf{y}} \quad \|F(\mathbf{y})\|_1 + \|(G(\mathbf{y}))^+\|_1 \quad (3.10)$$

have been considered, but the discontinuity in the derivatives of such an objective could create obvious problems. Instead, a smooth reformulation of the problem, suggested by Nocedal and Wright [40, Section 17.2] and Wächter and Biegler [55, Section 3.3], is employed. Consider the problem

$$\begin{aligned} \min_{\mathbf{y}, \mathbf{p}, \mathbf{n}, \mathbf{t}} \quad & \frac{\zeta}{2} \|M_R(\mathbf{y} - \mathbf{y}_R)\|_2^2 + \rho \sum_{i=1}^{n_F} (\mathbf{p}_i + \mathbf{n}_i) + \rho \sum_{i=1}^{n_G} (\mathbf{t}_i) \\ \text{s.t.} \quad & F(\mathbf{y}) - \mathbf{p} + \mathbf{n} = 0 \\ & G(\mathbf{y}) \leq \mathbf{t} \\ & \mathbf{p}, \mathbf{n}, \mathbf{t} \geq 0. \end{aligned} \quad (3.11)$$

Clearly $\mathbf{p} \in \mathbb{R}^{n_F}$, $\mathbf{n} \in \mathbb{R}^{n_F}$, and $\mathbf{t} \in \mathbb{R}^{n_G}$. This reformulation has the same goal as (3.10), with the added goal of finding feasible points close to some reference value $\mathbf{y}_R \in \mathbb{R}^{n_{\mathbf{y}}}$, which is currently considered to be the iterate of the regular interior point algorithm when the feasibility restoration is called. $M_R \in \mathbb{R}^{n_{\mathbf{y}} \times n_{\mathbf{y}}}$ is a scaling matrix defined as in [55, Section 3.3] by Wächter and Biegler to be

$$M_R = \text{diag}(\min\{1, 1/|\mathbf{y}_R^1|\}, \dots, \min\{1, 1/|\mathbf{y}_R^{n_{\mathbf{y}}}| \}). \quad (3.12)$$

The parameters ρ and ζ are chosen to weight the components of the objective appropriately for a desired outcome of the feasibility restoration. These parameters

are exactly those given by Wächter and Biegler [55, Section 3.3], where $\rho = 1000$ and $\zeta = \sqrt{\eta^0}$; here η^0 is the current barrier parameter of the original optimization algorithm.

The problem (3.11) is solved in the same manner as (2.5) using an interior-point approach with the same type of line-search. If the Lagrangian at the current iterate is defined as

$$\begin{aligned} \mathcal{L}(\mathbf{y}, \mathbf{p}, \mathbf{n}, \mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = & \frac{\zeta}{2} \|M_R(\mathbf{y} - \mathbf{y}_R)\|_2^2 + \rho \sum_{i=1}^{n_F} (\mathbf{p}_i + \mathbf{n}_i) + \rho \sum_{i=1}^{n_G} (\mathbf{t}_i) \\ & - \eta \sum_{i=1}^{n_F} (\ln(\mathbf{p}_i) + \ln(\mathbf{n}_i)) - \eta \sum_{i=1}^{n_G} (\ln(\mathbf{t}_i) + \ln(\mathbf{z}_i)) \\ & + \boldsymbol{\lambda}^T (F(\mathbf{y}) - \mathbf{p} + \mathbf{n}) + \boldsymbol{\mu}^T (G(\mathbf{y}) - \mathbf{t} + \mathbf{z}), \end{aligned} \quad (3.13)$$

the KKT conditions in this case are

$$\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{y}, \mathbf{p}, \mathbf{n}, \mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = 0, \quad (3.14a)$$

$$P\boldsymbol{\Lambda}^- - \eta\vec{e} = 0, \quad (3.14b)$$

$$N\boldsymbol{\Lambda}^+ - \eta\vec{e} = 0, \quad (3.14c)$$

$$TM^- - \eta\vec{e} = 0, \quad (3.14d)$$

$$ZM - \eta\vec{e} = 0, \quad (3.14e)$$

$$F(\mathbf{y}) - \mathbf{p} + \mathbf{n} = 0, \text{ and} \quad (3.14f)$$

$$G(\mathbf{y}) - \mathbf{t} + \mathbf{z} = 0, \quad (3.14g)$$

where $P = \text{diag}\{\mathbf{p}\}$, $N = \text{diag}\{\mathbf{n}\}$, $T = \text{diag}\{\mathbf{t}\}$, $Z = \text{diag}\{\mathbf{z}\}$, $\boldsymbol{\Lambda}^- = \text{diag}\{(\rho\vec{e} - \boldsymbol{\lambda})\}$, $\boldsymbol{\Lambda}^+ = \text{diag}\{(\rho\vec{e} + \boldsymbol{\lambda})\}$, $M^- = \text{diag}\{(\rho\vec{e} - \boldsymbol{\mu})\}$, and $M = \text{diag}\{\boldsymbol{\mu}\}$. Linearizing these

equations around the current iterate and, again, making some eliminations, the KKT system for the feasibility restoration has the KKT matrix

$$\begin{bmatrix} \nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}^k + (D_{\mathbf{y}} G^k)^T W^k D_{\mathbf{y}} G^k & (D_{\mathbf{y}} F^k)^T \\ D_{\mathbf{y}} F^k & -P^k((\Lambda^-)^k)^{-1} - N^k((\Lambda^+)^k)^{-1} \end{bmatrix},$$

the right hand side

$$-\begin{bmatrix} \nabla_{\mathbf{y}} \mathcal{L}^k + (D_{\mathbf{y}} G^k)^T W^k (\eta^k (M^k)^{-1} \vec{e} - \eta^k ((M^-)^k)^{-1} \vec{e} + G) \\ F^k - \mathbf{p}^k + \mathbf{n}^k + (\eta^k \vec{e} - (\Lambda^+)^k N^k)((\Lambda^+)^k)^{-1} - (\eta^k \vec{e} - (\Lambda^-)^k P^k)((\Lambda^-)^k)^{-1} \end{bmatrix},$$

and is solving for $\delta \mathbf{y}^k$ and $\delta \boldsymbol{\lambda}^k$, where $W^k = (M^-)^k M^k (M^k T^k + (M^-)^k Z^k)^{-1}$ is a diagonal matrix. The KKT matrix of the feasibility restoration has essentially the same structure as that of the original nonlinear program, allowing the use of the same factorization techniques for both matrices in many cases. When the KKT system is solved for $\delta \mathbf{y}^k$ and $\delta \boldsymbol{\lambda}^k$, $\delta \mathbf{p}^k$, $\delta \mathbf{n}^k$, $\delta \mathbf{t}^k$, $\delta \mathbf{z}^k$, and $\delta \boldsymbol{\mu}^k$ can be computed by the following relationships:

$$\delta \mathbf{p}^k = (\Lambda^-)^{-1} (\eta^k \vec{e} - P^k (\rho \vec{e} - \boldsymbol{\lambda}^k - \delta \boldsymbol{\lambda}^k))$$

$$\delta \mathbf{n}^k = (\Lambda^+)^{-1} (\eta^k \vec{e} - N^k (\rho \vec{e} + \boldsymbol{\lambda}^k + \delta \boldsymbol{\lambda}^k))$$

$$\delta \boldsymbol{\mu}^k = (M^k T^k + (M^-)^k Z^k)^{-1} (\eta^k \rho \vec{e} - 2\eta^k \boldsymbol{\mu}^k + (\rho M^k - (M^k)^2)(G^k + D_{\mathbf{y}} G^k \delta \mathbf{y}^k))$$

$$\delta \mathbf{t}^k = (M^-)^{-1} (\eta^k \vec{e} - T^k (\rho \vec{e} - \boldsymbol{\mu}^k - \delta \boldsymbol{\mu}^k))$$

$$\delta \mathbf{z}^k = M^{-1} (\eta^k \vec{e} - Z^k (\boldsymbol{\mu}^k + \delta \boldsymbol{\mu}^k)).$$

The step length $\alpha_{\mathbf{p}}^k$ is computed initially so that it satisfies

$$\alpha_{\mathbf{p}}^k = \max\{\alpha \in (0, 1] : \mathbf{p}^k + \alpha \delta \mathbf{p}^k \geq (1 - \tau) \mathbf{p}^k, \mathbf{n}^k + \alpha \delta \mathbf{n}^k \geq (1 - \tau) \mathbf{n}^k,$$

$$\mathbf{t}^k + \alpha \delta \mathbf{t}^k \geq (1 - \tau) \mathbf{y}^k, \mathbf{z}^k + \alpha \delta \mathbf{z}^k \geq (1 - \tau) \mathbf{z}^k\}$$

and α_d^k is computed as before. The backtracking line search continues in the same manner as the original NLP, with the merit function and penalty parameter defined analogously to the previous section, only excluding the second order correction. The iterates are then updated by

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha_p^k \delta \mathbf{y}^k$$

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \alpha_p^k \delta \mathbf{p}^k$$

$$\mathbf{n}^{k+1} = \mathbf{n}^k + \alpha_p^k \delta \mathbf{n}^k$$

$$\mathbf{t}^{k+1} = \mathbf{t}^k + \alpha_p^k \delta \mathbf{t}^k$$

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \alpha_p^k \delta \mathbf{z}^k$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \alpha_d^k \delta \boldsymbol{\lambda}^k$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \alpha_d^k \delta \boldsymbol{\mu}^k.$$

Once the error in the constraints has been reduced sufficiently, the feasibility restoration returns the current iterate \mathbf{y}^k to the original optimization.

3.5 Initialization of Primal and Dual Variables

At the beginning of the optimization algorithm, a guess at the primal variables \mathbf{y}^0 must be given. The slack variables are initialized as $\mathbf{z}^0 = \max\{10^{-5}, G(\mathbf{y}^0)\}$ to avoid singularities in the matrix Z^{-1} . Also, $\boldsymbol{\lambda}^0 = 0$ and $\boldsymbol{\mu}^0 = \vec{e}$.

In the feasibility restoration, $\boldsymbol{\lambda}^0 = 0$, $\boldsymbol{\mu}^0 = \vec{e}$ and \mathbf{y}^0 is chosen to be the current iterate of the original algorithm. The variables \mathbf{p}^0 , \mathbf{n}^0 , \mathbf{t}^0 , and \mathbf{z}^0 are chosen to satisfy

(3.14b–g) at \mathbf{y}^0 , and if the feasibility restoration computes a stepsize that is too small at step k , a feasibility restoration is attempted by satisfying (3.14b–g) at \mathbf{y}^k .

When the feasibility restoration started at \mathbf{y}^k is completed and the value \mathbf{y}^{k+1} is returned to the original optimization algorithm, $\mathbf{z}^{k+1} = \max\{10^{-5}, -G(\mathbf{y}^{k+1})\}$, $\boldsymbol{\lambda}^{k+1} = 0$, and $\boldsymbol{\mu}^{k+1} = \bar{\mathbf{e}}$. If a quasi-Newton update is used for $\nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}^{k+1}$, then it is reinitialized along with $\nu^{k+1} = 1$ and $\eta^{k+1} = 0.1$ as if the original algorithm were being restarted.

Chapter 4

Direct Transcription using Pseudospectral Collocation

4.1 Introduction

Collocation methods for the solution of the optimal control problem (2.1) approximate the states and control by (piecewise) polynomials and require that the dynamics and constraints in (2.1) with states and controls replaced by their (piecewise) polynomial approximations hold at specified points, the so-called collocation points, in the interval $[t_0, t_f]$.

Many collocation methods exist for optimal control problems. Reddien [45] used collocation at Gauss points. Reddien established best possible convergence rates for the Gauss PS method applied to unconstrained optimal control problems. Gauss PS

method also applied to nonlinear OCPs by Benson [4]. Stryk and Bulirsch [52], von Stryk [54], and Kameswaran and Biegler [30] use collocation at Radau points.

This thesis considers Legendre-Gauss-Lobatto (LGL) Pseudospectral (PS) methods. Here states and controls are approximated by polynomials of degree N and the collocation points are the roots of the derivatives of the N^{th} order Legendre Polynomial and the two boundary points. LGL PS collocation is described in [14, 15, 21, 31, 48], and was chosen because of its exclusive use in solving ZPM OCPs of Chapter 7 through an implementation in the software DIDO [15]. The Master's thesis of Bhatt [6] and the paper by Bedrossian and Bhatt [3] applied an LGL PS method to the ZPM. The computed control was successfully implemented and flight tested. See the Master's thesis of Bhatt [6].

This chapter will review some of the properties of the LGL PS collocation method.

4.2 Problem Formulation

The description of the Pseudospectral Collocation method appearing in this thesis will proceed in the same way as Pietz [42, Chapter 3]. Pseudospectral Collocation methods approximate the states and controls by polynomials defined by a given *global* basis function $\psi_i(t)$. That is, $x(t) \approx x^N(t) = \sum_{i=0}^N \mathbf{s}_i \psi_i(t)$ and $u(t) \approx u^N(t) = \sum_{i=0}^N \mathbf{q}_i \psi_i(t)$. Often, the chosen basis is the Lagrange basis

$$\psi_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^N \frac{(t - t_j)}{(t_i - t_j)}.$$

This basis has the useful property

$$\psi_i(t_j) = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{if } i \neq j, \end{cases}$$

so that $x^N(t_i) = \mathbf{s}_i$ and $u^N(t_i) = \mathbf{q}_i$. Further, the time derivative of the states can be approximated by $\dot{x}(t) \approx \dot{x}^N(t) = \sum_{i=0}^N \mathbf{s}_i \dot{\psi}_i(t)$. Integral objective functions in the optimal control problem are approximated by

$$\int_{t_0}^{t_f} l(x(t), u(t), t) dt \approx \int_{t_0}^{t_f} \sum_{i=0}^N \psi_i(t) l(\mathbf{s}_i, \mathbf{q}_i, t_i) dt = \sum_{i=0}^N w_i l(\mathbf{s}_i, \mathbf{q}_i, t_i),$$

where

$$w_i = \int_{t_0}^{t_f} \psi_i(t) dt, \quad i = 0, \dots, N.$$

Once the nodes t_i are chosen the optimal control problem (2.1) can be transcribed, resulting in a nonlinear program in the optimization variables $\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_N \in \mathbb{R}^n$ and $\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_N \in \mathbb{R}^m$. The PS transcription of the optimal control problem (2.1) is given by

$$\begin{aligned} \min \quad & \sum_{i=0}^N [w_i l(\mathbf{s}_i, \mathbf{q}_i, t_i)] + e(\mathbf{s}_N, t_N) \\ \text{s.t.} \quad & \\ & \sum_{j=0}^N \mathbf{D}_{ij} \mathbf{s}_j = f(\mathbf{s}_i, \mathbf{q}_i, t_i), \quad i = 0, \dots, N \\ & a(\mathbf{s}_0, t_0) = 0 \\ & b(\mathbf{s}_N, t_N) = 0 \\ & g(\mathbf{s}_i, \mathbf{q}_i, t_i) \leq 0, \quad i = 0, \dots, N, \end{aligned}$$

where $\mathbf{D}_{ij} = \dot{\psi}_j(t_i)$.

4.3 Legendre-Gauss-Lobatto (LGL)

Pseudospectral Collocation

This thesis will consider a particular form of pseudospectral collocation where the collocation points t_i , $i = 0, \dots, N$ are defined as the zeros of the derivative of the N^{th} -order Legendre polynomial

$$P_N(t) = \frac{1}{2^N N!} \frac{d^N}{dt^N} (t^2 - 1)^N$$

with the extra conditions that $t_0 = -1$ and $t_N = 1$ [42, Section 3.1]. For more information on Legendre polynomials see, e.g., Hesthaven, Gottlieb, and Gottlieb [28]. LGL PS collocation is only one of the many available PS discretizations and it is currently implemented in the software DIDO [15], which has been used exclusively in the solution of ZPM OCPs.

It is important to note that the roots of the derivatives of the Legendre polynomial of order N lie in the interval $(-1, 1)$, so that, along with the nodes t_0 and t_N , the collocation nodes lie generically in the interval $[-1, 1]$. To apply LGL PS collocation on an interval $[t_0, t_f]$, a change of variable $\tau : [t_0, t_f] \rightarrow [-1, 1]$ by $\tau = \frac{2}{(t_f - t_0)}(t - t_f) + 1$ must be applied to scale the time interval appropriately. This leads to a scaling of the differential equations due to

$$\frac{dx}{d\tau} = \frac{dx}{dt} \frac{dt}{d\tau} = \frac{2}{t_f - t_0} \frac{dx}{dt}$$

and a scaling of the integral objective function.

The differentiation matrix for Legendre-Gauss-Lobatto collocation is defined by

$$\mathbf{D}_{ij} = \begin{cases} \frac{P_N(t_i)}{P_N(t_j)} \frac{1}{t_i - t_j}, & i \neq j; \\ \frac{-N(N+1)}{4}, & i = j = 0; \\ \frac{N(N+1)}{4}, & i = j = N; \\ 0, & \text{otherwise,} \end{cases}$$

while the quadrature weights are

$$w_i = \int_{-1}^1 \psi_i(t) dt = \frac{2}{N(N+1)} \frac{1}{[P_N(t_i)]^2}$$

[14, Section III]. Applying Legendre-Gauss-Lobatto Collocation to the optimal control problem, the problem (2.1) is transformed into a nonlinear program in the variables

$$\mathbf{y} = \begin{bmatrix} \mathbf{s}_0^T & \mathbf{q}_0^T & \cdots & \mathbf{s}_N^T & \mathbf{q}_N^T \end{bmatrix}^T \in \mathbb{R}^{n_y} \quad (4.1)$$

that can be written in the form of (2.5). The functions of (2.5) are defined as,

$$J(\mathbf{y}) := \sum_{i=0}^N \left[\frac{t_f - t_0}{2} w_i l(\mathbf{s}_i, \mathbf{q}_i, t_i) \right] + e(\mathbf{s}_N, t_N) \quad (4.2a)$$

$$F(\mathbf{y}) := \begin{bmatrix} \frac{t_f - t_0}{2} f(\mathbf{s}_0, \mathbf{q}_0, t_0) - \sum_{j=0}^N \mathbf{D}_{0j} \mathbf{s}_j \\ \vdots \\ \frac{t_f - t_0}{2} f(\mathbf{s}_N, \mathbf{q}_N, t_N) - \sum_{j=0}^N \mathbf{D}_{Nj} \mathbf{s}_j \\ a(\mathbf{s}_0, t_0) \\ b(\mathbf{s}_N, t_N) \end{bmatrix}, \text{ and} \quad (4.2b)$$

$$G(\mathbf{y}) := \begin{bmatrix} g(\mathbf{s}_0, \mathbf{q}_0, t_0) \\ \vdots \\ g(\mathbf{s}_N, \mathbf{q}_N, t_N) \end{bmatrix}, \quad (4.2c)$$

where $n_y = (N+1)(n+m)$, $n_F = (N+1)n + n_a + n_b$ and $n_G = (N+1)n_g$. Further, (4.2) has equality constraint multipliers

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_0^T & \cdots & \boldsymbol{\lambda}_N^T & \boldsymbol{\nu}_0^T & \boldsymbol{\nu}_N^T \end{bmatrix}^T \quad (4.3)$$

and inequality constraint multipliers

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_0^T & \cdots & \boldsymbol{\mu}_N^T \end{bmatrix}^T, \quad (4.4)$$

where $\boldsymbol{\lambda}_i \in \mathbb{R}^n$ and $\boldsymbol{\mu}_i \in \mathbb{R}^{n_g}$ for $i = 0, \dots, N$, $\boldsymbol{\nu}_0 \in \mathbb{R}^{n_a}$, and $\boldsymbol{\nu}_N \in \mathbb{R}^{n_b}$, corresponding to the rows of (4.2b) and (4.2c).

It should be noted that in LGL PS collocation the state is approximated by a polynomial of degree N , but that the dynamics (2.1b) alone are enforced at the $N+1$ collocation points. Additional constraints on the states are enforced by the boundary conditions (2.1c), (2.1d). For a given control u^N this leads to an over-determined system for the discrete states.

4.4 Well-posedness and Convergence of the LGL PS Collocation

After the optimal control problem (2.1) has been discretized using LGL PS collocation several important questions arise:

- 1) Does the discretized optimal control problem have a solution (for sufficiently large N), provided that the original optimal control problem (2.1) has a solution

- 2) If $x^N(t) = \sum_{i=0}^N \mathbf{s}_i \psi_i(t)$ and $u^N(t) = \sum_{i=0}^N \mathbf{q}_i \psi_i(t)$ solve the discretized optimal control problem, does the sequence $\{x^N, u^N\}$ converge to a solution of the original optimal control problem (2.1)?

Answers to these questions are given by Gong, et al., [21]. The answer to the first problem is negative. A simple example given by Gong, et al., [21, Section 3] illustrates the absence of a feasible solution for even some linear systems. The lack of feasibility is related to the fact that in LGL PS collocation the state is approximated by a polynomial of degree N , but that the dynamics (2.1b) alone are enforced at the $N + 1$ collocation points. Since additional constraints on the states are enforced by the boundary conditions (2.1c), (2.1d) this leads to an over-determined system for the discrete states for a given control u^N .

To remedy this situation, Gong, et al., [21] propose to relax the problem. Suppose that a solution (x_*, u_*) of the original optimal control problem (2.1) exists with $x_* \in W^{n_d, \infty}([t_0, t_f], \mathbb{R}^n)$, $n_d \geq 2$. The relaxation to the discretized optimal control problem

is given by

$$\min \sum_{i=0}^N \left[\frac{t_f - t_0}{2} w_i l(\mathbf{s}_i, \mathbf{q}_i, t_i) \right] + e(\mathbf{s}_N, t_N) \quad (4.5a)$$

s.t.

$$\left\| \begin{bmatrix} \frac{t_f - t_0}{2} f(\mathbf{s}_0, \mathbf{q}_0, t_0) - \sum_{j=0}^N \mathbf{D}_{0j} \mathbf{s}_j \\ \vdots \\ \frac{t_f - t_0}{2} f(\mathbf{s}_N, \mathbf{q}_N, t_N) - \sum_{j=0}^N \mathbf{D}_{Nj} \mathbf{s}_j \\ a(\mathbf{s}_0, t_0) \\ b(\mathbf{s}_N, t_N) \end{bmatrix} \right\|_{\infty} \leq (N-1)^{\frac{3}{2}-n_d} \quad (4.5b)$$

$$\begin{bmatrix} g(\mathbf{s}_0, \mathbf{q}_0, t_0) \\ \vdots \\ g(\mathbf{s}_N, \mathbf{q}_N, t_N) \end{bmatrix} \leq (N-1)^{\frac{3}{2}-n_d} \vec{c}. \quad (4.5c)$$

Using polynomial approximation properties of functions in $W^{n_d, \infty}([t_0, t_f], \mathbb{R}^n)$, Gong, et al., [21] show that the relaxed problem (4.5) has a solution for sufficiently large N . The relaxation is used theoretically, but it does not seem to be used in actual numerical computations.

Let

$$\begin{aligned} x_*^N(t) &= \sum_{i=0}^N \mathbf{s}_i^{N*} \psi_i(t) \\ u_*^N(t) &= \sum_{i=0}^N \mathbf{q}_i^{N*} \psi_i(t) \end{aligned}$$

denote the solution of the relaxed (4.5) discretized optimal control problem (for sufficiently large N).

The following convergence result is proven by Gong, et al., [21, Thm. 2].

Theorem 4.4.1 *Let $x_*^N(t) = \sum_{i=0}^N \mathbf{s}_i^{N*} \psi_i(t)$, $u_*^N(t) = \sum_{i=0}^N \mathbf{q}_i^{N*} \psi_i(t)$ be a global minimizer of the relaxed problem (4.5). If the sequence $\{(\mathbf{s}_0^{N*}, \dot{x}_*^N, u_*^N)\}$ has a uniform accumulation point $(\mathbf{s}_0^\infty, \dot{x}^\infty, u^\infty)$ such that $t \mapsto \dot{x}^\infty(t)$, $t \mapsto u^\infty(t)$ are continuous on $[t_0, t_f]$ then u^∞ is an optimal control of (2.1) and $x^\infty(t) = \mathbf{s}_0^\infty + \int_{t_0}^t \dot{x}^\infty(\tau) d\tau$ is the corresponding state.*

The previous theorem applies to global minimizers of the relaxed problem (4.5). It does not make any statement about convergence of the adjoint variables, which will be examined next, and it provides no information about convergence rates.

4.5 Adjoint Estimation Properties for Pseudospectral Collocation

Let $\mathbf{s}_0, \dots, \mathbf{s}_N, \mathbf{q}_0, \dots, \mathbf{q}_N$ be a solution of the discretized optimal control problem (assuming it exists) and let $\boldsymbol{\lambda}_0, \dots, \boldsymbol{\lambda}_N, \boldsymbol{\mu}_0, \dots, \boldsymbol{\mu}_N, \boldsymbol{\nu}_0, \boldsymbol{\nu}_N$ be the corresponding Lagrange multipliers. Furthermore set

$$\begin{aligned} x^N(t) &= \sum_{i=0}^N \mathbf{s}_i \psi_i(t) & u^N(t) &= \sum_{i=0}^N \mathbf{q}_i \psi_i(t) \\ \boldsymbol{\lambda}^N(t) &= \sum_{i=0}^N \boldsymbol{\lambda}_i \psi_i(t) & \boldsymbol{\mu}^N(t) &= \sum_{i=0}^N \boldsymbol{\mu}_i \psi_i(t). \end{aligned}$$

How are the discrete adjoint variables related to the adjoint variables that arise in the first order optimality conditions stated in Theorem 2.2.1?

This question is examined by Fahroo, Ross, et al., in [14], [47], [21]. Suppose the first order optimality conditions (2.4) are discretized using LGL PS collocation. That is, replace $x, u, \tilde{\lambda}, \tilde{\mu}$ in (2.4) by

$$\begin{aligned} x^N(t) &= \sum_{i=0}^N \mathbf{s}_i^N \psi_i(t) & u^N(t) &= \sum_{i=0}^N \mathbf{q}_i^N \psi_i(t) \\ \lambda^N(t) &= \sum_{i=0}^N \tilde{\lambda}_i^N \psi_i(t) & \mu^N(t) &= \sum_{i=0}^N \tilde{\mu}_i^N \psi_i(t) \end{aligned}$$

and enforce (2.4a,d,e) at the collocation points. This leads to

$$\begin{aligned} 0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{s}} l(\mathbf{s}_i, \mathbf{q}_i, t_i) + \frac{t_f - t_0}{2} \tilde{\lambda}_i^T D_{\mathbf{s}} f(\mathbf{s}_i, \mathbf{q}_i, t_i) \\ &\quad + \sum_{j=0}^N \mathbf{D}_{ij} \tilde{\lambda}_j^T + \tilde{\mu}_i^T D_{\mathbf{s}} g(\mathbf{s}_i, \mathbf{q}_i, t_i), \quad i = 0, \dots, N, \end{aligned} \quad (4.6a)$$

$$\tilde{\lambda}_0^T = -\nu_0^T D_x a(x_0, t_0), \quad (4.6b)$$

$$\tilde{\lambda}_N^T = \nu_f^T D_x b(\mathbf{s}_N, t_N) + \nabla_x e(\mathbf{s}_N, t_N), \quad (4.6c)$$

$$\begin{aligned} 0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{q}} l(\mathbf{s}_i, \mathbf{q}_i, t_i) + \frac{t_f - t_0}{2} \tilde{\lambda}_i^T D_{\mathbf{q}} f(\mathbf{s}_i, \mathbf{q}_i, t_i) \\ &\quad + \tilde{\mu}_i^T D_{\mathbf{q}} g(\mathbf{s}_i, \mathbf{q}_i, t_i), \quad i = 0, \dots, N \end{aligned} \quad (4.6d)$$

and

$$\tilde{\mu}_i \geq 0, \quad \tilde{\mu}_i^T g(\mathbf{s}_i, \mathbf{q}_i, t_i), \quad i = 0, \dots, N. \quad (4.6e)$$

Now consider the discretization (4.2). The Lagrangian corresponding to the dis-

cretized optimal control problem (4.2) is given by

$$\begin{aligned}\mathcal{L}(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \sum_{i=0}^N \left[w_i \frac{t_f - t_0}{2} l(\mathbf{s}_i, \mathbf{q}_i, t_i) \right] + e(\mathbf{s}_N, t_N) \\ &\quad + \sum_{i=0}^N \left[\boldsymbol{\lambda}_i^T \left(\frac{t_f - t_0}{2} f^i(\mathbf{s}_i, \mathbf{q}_i, t_i) - \sum_{j=0}^N \mathbf{D}_{ij} \mathbf{s}_j \right) + \boldsymbol{\mu}_i^T g^i(\mathbf{s}_i, \mathbf{q}_i, t_i) \right] \\ &\quad + \boldsymbol{\nu}_0^T a(\mathbf{s}_0, t_0) + \boldsymbol{\nu}_N^T b(\mathbf{s}_N, t_N),\end{aligned}$$

where $\boldsymbol{\lambda}_i \in \mathbb{R}^n$ for $i = 0, \dots, N$, $\boldsymbol{\mu}_i \in \mathbb{R}^{n_g}$ for $i = 0, \dots, N$, and $\boldsymbol{\nu}_0 \in \mathbb{R}^{n_a}$ and $\boldsymbol{\nu}_N \in \mathbb{R}^{n_b}$ are the multipliers.

Recalling the KKT conditions, the control optimality conditions for $i = 0, \dots, N$ are

$$\nabla_{\mathbf{q}_i} \mathcal{L}(y) = w_i \frac{t_f - t_0}{2} \nabla_{\mathbf{q}_i} l(\mathbf{s}_i, \mathbf{q}_i, t_i) + \boldsymbol{\lambda}_i^T \frac{t_f - t_0}{2} D_{\mathbf{q}_i} f(\mathbf{s}_i, \mathbf{q}_i, t_i) + \boldsymbol{\mu}_i^T D_{\mathbf{q}_i} g(\mathbf{s}_i, \mathbf{q}_i, t_i) = 0.$$

Dividing by w_i gives

$$0 = \frac{t_f - t_0}{2} \nabla_{\mathbf{q}_i} l(\mathbf{s}_i, \mathbf{q}_i, t_i) + \frac{1}{w_i} \frac{t_f - t_0}{2} \boldsymbol{\lambda}_i^T D_{\mathbf{q}_i} f(\mathbf{s}_i, \mathbf{q}_i, t_i) + \frac{1}{w_i} \boldsymbol{\mu}_i^T D_{\mathbf{q}_i} g(\mathbf{s}_i, \mathbf{q}_i, t_i).$$

Similarly applying these first order necessary conditions, the state optimality conditions result in three cases. First, for $i = 1, \dots, N - 1$,

$$\begin{aligned}0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{s}_i} l(\mathbf{s}_i, \mathbf{q}_i, t_i) + \frac{1}{w_i} \frac{t_f - t_0}{2} \boldsymbol{\lambda}_i^T D_{\mathbf{s}_i} f(\mathbf{s}_i, \mathbf{q}_i, t_i) \\ &\quad - \frac{1}{w_i} \sum_{j=0}^N \mathbf{D}_{ji} \boldsymbol{\lambda}_j^T + \frac{1}{w_i} \boldsymbol{\mu}_i^T D_{\mathbf{s}_i} g(\mathbf{s}_i, \mathbf{q}_i, t_i).\end{aligned}\tag{4.7a}$$

For $i = 0$,

$$\begin{aligned}0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{s}_0} l(\mathbf{s}_0, \mathbf{q}_0, t_0) + \frac{1}{w_0} \frac{t_f - t_0}{2} \boldsymbol{\lambda}_0^T D_{\mathbf{s}_0} f(\mathbf{s}_0, \mathbf{q}_0, t_0) - \frac{1}{w_0} \sum_{j=0}^N \mathbf{D}_{j0} \boldsymbol{\lambda}_j^T \\ &\quad + \frac{1}{w_0} \boldsymbol{\mu}_0^T D_{\mathbf{s}_0} g(\mathbf{s}_0, \mathbf{q}_0, t_0) + \frac{1}{w_0} \boldsymbol{\nu}_0^T D_{\mathbf{s}_0} a(\mathbf{s}_0, t_0);\end{aligned}\tag{4.7b}$$

and, for $i = N$,

$$\begin{aligned}
 0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{s}_N} l(\mathbf{s}_N, \mathbf{q}_N, t_N) + \frac{1}{w_N} \frac{t_f - t_0}{2} \boldsymbol{\lambda}_N^T D_{\mathbf{s}_N} f(\mathbf{s}_N, \mathbf{q}_N, t_N) \\
 &\quad - \frac{1}{w_N} \sum_{j=0}^N \mathbf{D}_{jN} \boldsymbol{\lambda}_j^T + \frac{1}{w_N} \boldsymbol{\mu}_N^T D_{\mathbf{s}_N} g(\mathbf{s}_N, \mathbf{q}_N, t_N) + \frac{1}{w_N} \nabla_{\mathbf{s}_N} e(\mathbf{s}_N, t_N) \\
 &\quad + \frac{1}{w_N} \nu_N^T D_{\mathbf{s}_N} b(\mathbf{s}_N, t_N).
 \end{aligned} \tag{4.7c}$$

By the definition of the elements of the differentiation matrix for LGL PS collocation the following, shown by Fahroo and Ross [14, Section IV], are true: $w_i \mathbf{D}_{ij} = -w_j \mathbf{D}_{ji}$, when $i \neq j$, $\mathbf{D}_{ii} = 0$ for $i = 2, \dots, N-1$, $2\mathbf{D}_{NN} = \frac{1}{w_N}$, and $2\mathbf{D}_{00} = \frac{-1}{w_0}$. Using these relations, (4.7a-c) can be rewritten as follows:

$$\begin{aligned}
 0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{s}_i} l(\mathbf{s}_i, \mathbf{q}_i, t_i) + \frac{1}{w_i} \frac{t_f - t_0}{2} \boldsymbol{\lambda}_i^T D_{\mathbf{s}_i} f(\mathbf{s}_i, \mathbf{q}_i, t_i) \\
 &\quad + \sum_{j=0}^N \mathbf{D}_{ij} \frac{1}{w_j} \boldsymbol{\lambda}_j^T + \frac{1}{w_i} \boldsymbol{\mu}_i^T D_{\mathbf{s}_i} g(\mathbf{s}_i, \mathbf{q}_i, t_i)
 \end{aligned} \tag{4.8a}$$

for $i = 1, \dots, N-1$,

$$\begin{aligned}
 0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{s}_0} l(\mathbf{s}_0, \mathbf{q}_0, t_0) + \frac{1}{w_0} \frac{t_f - t_0}{2} \boldsymbol{\lambda}_0^T D_{\mathbf{s}_0} f(\mathbf{s}_0, \mathbf{q}_0, t_0) + \sum_{j=0}^N \mathbf{D}_{0j} \frac{1}{w_j} \boldsymbol{\lambda}_j^T \\
 &\quad + \frac{1}{w_0} \boldsymbol{\mu}_0^T D_{\mathbf{s}_0} g(\mathbf{s}_0, \mathbf{q}_0, t_0) + \frac{1}{w_0} \left(\frac{1}{w_0} \boldsymbol{\lambda}_0^T + \nu_0^T D_{\mathbf{s}_0} a(\mathbf{s}_0, t_0) \right)
 \end{aligned} \tag{4.8b}$$

for $i = 0$, and

$$\begin{aligned}
 0 &= \frac{t_f - t_0}{2} \nabla_{\mathbf{s}_N} l(\mathbf{s}_N, \mathbf{q}_N, t_N) + \frac{1}{w_N} \frac{t_f - t_0}{2} \boldsymbol{\lambda}_N^T D_{\mathbf{s}_N} f(\mathbf{s}_N, \mathbf{q}_N, t_N) \\
 &\quad - \frac{1}{w_N} \sum_{j=0}^N \mathbf{D}_{jN} \boldsymbol{\lambda}_j^T + \frac{1}{w_N} \boldsymbol{\mu}_N^T D_{\mathbf{s}_N} g(\mathbf{s}_N, \mathbf{q}_N, t_N) \\
 &\quad + \frac{1}{w_N} \left(-\frac{1}{w_N} \boldsymbol{\lambda}_N^T + \nabla_{\mathbf{s}_N} e(\mathbf{s}_N, t_N) + \nu_N^T D_{\mathbf{s}_N} b(\mathbf{s}_N, t_N) \right)
 \end{aligned} \tag{4.8c}$$

for $i = N$. Comparison of the KKT conditions (4.8) to (4.6) shows that the relationships $\boldsymbol{\lambda}_i = w_i \tilde{\boldsymbol{\lambda}}_i$ and $\boldsymbol{\mu}_i = \frac{t_f - t_0}{2} w_i \tilde{\boldsymbol{\mu}}_i$ hold provided that the so-called closure conditions

$$\begin{aligned} \frac{1}{w_0} \boldsymbol{\lambda}_0^T + \boldsymbol{\nu}_0^T D_{\mathbf{s}_0} a(\mathbf{s}_0, t_0) &= 0 \\ \frac{1}{w_N} \boldsymbol{\lambda}_N^T - \nabla_{\mathbf{s}_N} e(\mathbf{s}_N, t_N) - \boldsymbol{\nu}_N^T D_{\mathbf{s}_N} b(\mathbf{s}_N, t_N) &= 0 \end{aligned}$$

hold. For more details see Ross and Fahroo [47] and Gong, et al., [21]. It should be noted that the closure conditions are not automatically satisfied. See the example in Section 5 of Gong, et al., [21].

4.6 Singularity of the LGL PS KKT Matrix

One often overlooked shortcoming of Pseudospectral Collocation is the potential singularity of the KKT matrix for a class of optimal control problems. Consider now

optimal control problems of the form

$$\begin{aligned} \min_{x(t), \hat{x}(t), u(t)} \quad & \int_{t_0}^{t_f} [l(x(t), u(t), t) + L(x(t), u(t), t)^T \hat{x}(t)] dt \\ & + e(x(t_f), t_f) + E(x(t_f), t_f)^T \hat{x}(t_f) \end{aligned} \quad (4.9a)$$

s.t.

$$\dot{x}(t) = f(x(t), u(t), t), \quad t \in [t_0, t_f] \quad (4.9b)$$

$$\dot{\hat{x}}(t) = \hat{f}(x(t), u(t), t), \quad t \in [t_0, t_f] \quad (4.9c)$$

$$a(x(t_0), t_0) = 0 \quad (4.9d)$$

$$b(x(t_f), t_f) = 0 \quad (4.9e)$$

$$g(x(t), u(t), t) \leq 0, \quad t \in [t_0, t_f] \quad (4.9f)$$

with $\hat{x} \in \mathbb{R}^{\hat{n}}$, $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \rightarrow \mathbb{R}^{\hat{n}}$, and $E : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \rightarrow \mathbb{R}^{\hat{n}}$. This class of problems defines a particular form of (2.1), where some of the state variables, \hat{x} , do not appear in the right hand sides of (4.9b,c), the constraints (4.9d–f), and only appear linearly in the objective (4.9a). Problems with the structure (4.9) arise for example when the optimal control problem (2.1) in Bolza form is converted into an optimal control problem in Mayer form.

When transcribed by LGL PS collocation, the Lagrangian is

$$\begin{aligned} \mathcal{L}(\cdot) = & \sum_{i=0}^N \left(w_i \frac{t_f - t_0}{2} l(\mathbf{s}_i, \mathbf{q}_i, t_i) + L(\mathbf{s}_i, \mathbf{q}_i, t_i)^T \hat{\mathbf{s}}_i \right) + e(\mathbf{s}_N, t_N) + E(\mathbf{s}_N, t_N)^T \hat{\mathbf{s}}_N \\ & + \boldsymbol{\nu}_0^T a(\mathbf{s}_0, t_0) + \sum_{i=0}^N \left(\boldsymbol{\lambda}_i^T \begin{bmatrix} \sum_{j=0}^N \mathbf{D}_{ij} \mathbf{s}_j - \frac{t_f - t_0}{2} f^i(\mathbf{s}_i, \mathbf{q}_i, t_i) \\ \sum_{j=0}^N \mathbf{D}_{ij} \hat{\mathbf{s}}_j - \frac{t_f - t_0}{2} \hat{f}^i(\mathbf{s}_i, \mathbf{q}_i, t_i) \end{bmatrix} + \boldsymbol{\mu}_i^T g^i(\mathbf{s}_i, \mathbf{q}_i, t_i) \right) \\ & + \boldsymbol{\nu}_N^T b(\mathbf{s}_N, t_N). \end{aligned}$$

Let

$$F = \begin{bmatrix} \frac{t_f - t_0}{2} f^0(\mathbf{s}_0, \mathbf{q}_0, t_0) - \sum_{j=0}^N \mathbf{D}_{0j} \mathbf{s}_j \\ \frac{t_f - t_0}{2} \hat{f}^0(\mathbf{s}_0, \mathbf{q}_0, t_0) - \sum_{j=0}^N \mathbf{D}_{0j} \hat{\mathbf{s}}_j \\ \vdots \\ \frac{t_f - t_0}{2} f^N(\mathbf{s}_N, \mathbf{q}_N, t_N) - \sum_{j=0}^N \mathbf{D}_{Nj} \mathbf{s}_j \\ \frac{t_f - t_0}{2} \hat{f}^N(\mathbf{s}_N, \mathbf{q}_N, t_N) - \sum_{j=0}^N \mathbf{D}_{Nj} \hat{\mathbf{s}}_j \end{bmatrix},$$

then the KKT matrix for this particular class of problems has the form

$$\begin{bmatrix} \nabla_{\mathbf{s}\mathbf{s}}^2 \mathcal{L} & 0 & \nabla_{\mathbf{q}\mathbf{s}}^2 \mathcal{L} & D_{\mathbf{s}} F^T & D_{\mathbf{s}} a^T & D_{\mathbf{s}} b^T \\ 0 & 0 & 0 & -\mathbf{D}^T & 0 & 0 \\ \nabla_{\mathbf{s}\mathbf{q}}^2 \mathcal{L} & 0 & \nabla_{\mathbf{q}\mathbf{q}}^2 \mathcal{L} & D_{\mathbf{q}} F^T & 0 & 0 \\ D_{\mathbf{s}} F & -\mathbf{D} & D_{\mathbf{q}} F & 0 & 0 & 0 \\ D_{\mathbf{s}} a & 0 & 0 & 0 & 0 & 0 \\ D_{\mathbf{s}} b & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

At first, it is not so clear why this matrix could be singular; however, B. Fornberg [17, Section 4.4] states that the matrix \mathbf{D} for LGL PS collocation, and for all nonperiodic PS approximations, has only zero eigenvalues. This is a result of differentiating

the polynomials that approximate the states and controls a finite number of times [17, Section 4.4] and is substantiated by Gong, et al. [22, Section III]. This poses a problem when considering this class of problems, since the KKT matrix is always singular, particularly when transforming an OCP in Bolza form to a problem in Mayer form.

4.7 Structure and Sparsity of the KKT Systems

In order to solve the nonlinear program resulting from the transcription of (2.1) by LGL Pseudospectral collocation a variety of methods can be employed. Often these methods require the factorization of a matrix known as the KKT matrix. Essentially this matrix is the linearization of the KKT conditions with respect to the optimization variables, \mathbf{y} , and the multipliers, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. Visually, in its simplest form the KKT matrix looks like

$$\begin{bmatrix} \nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}(\mathbf{y}) + (D_{\mathbf{y}}G^k)^T M^k (Z^k)^{-1} D_{\mathbf{y}}G^k & D_{\mathbf{y}}F(\mathbf{y})^T \\ D_{\mathbf{y}}F(\mathbf{y}) & 0 \end{bmatrix},$$

when the variables are considered in the order given by $\begin{bmatrix} \mathbf{y}^T & \boldsymbol{\lambda}^T \end{bmatrix}^T$, with \mathbf{y} defined in (4.1) and $\boldsymbol{\lambda}$ defined in (4.3). However, if the variables are considered in the order

$$\begin{bmatrix} \boldsymbol{\nu}_0^T & \mathbf{s}_0^T & \mathbf{q}_0^T & \boldsymbol{\lambda}_0^T & \mathbf{s}_1^T & \mathbf{q}_1^T & \boldsymbol{\lambda}_1^T & \cdots & \mathbf{s}_N^T & \mathbf{q}_N^T & \boldsymbol{\lambda}_N^T & \boldsymbol{\nu}_N^T \end{bmatrix}^T \quad (4.11)$$

this matrix has the structure shown in figure 4.1 for LGL Pseudospectral collocation.

It is important to note that only half of this matrix must be stored at any time since

it is symmetric. Also, this ordering of variables was chosen to correspond to that of the MS KKT matrix shown in the next chapter. Other orderings could be used, but after considering a few of the permutations given in MATLAB, none of these have a significant advantage over this predefined ordering. This comparison is shown in figure 4.3.

When factored into lower- and upper- triangular matrices by Gaussian elimination with pivoting, these matrices are dense for LGL PS collocation, and require a significant amount of storage space even compared to the storage of the KKT matrix itself. An example of this can be seen in figure 4.2, where the decomposed matrices correspond to the matrix in figure 4.1.

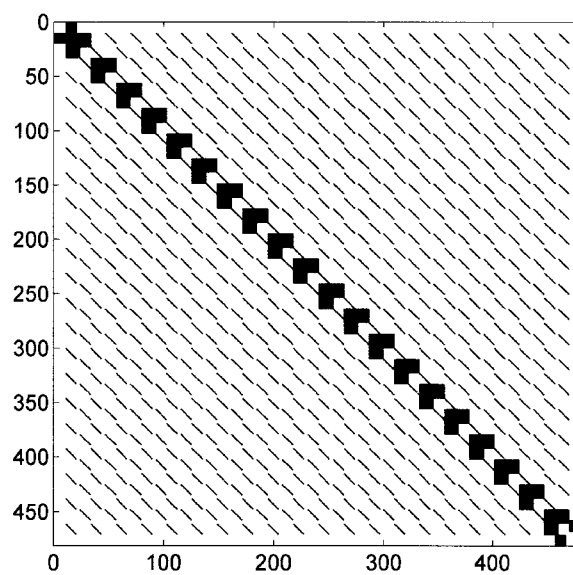


Figure 4.1: Sparsity pattern for LGL PS KKT matrix with $N = 20$, $n = n_a = n_b = 10$, and $m = 3$

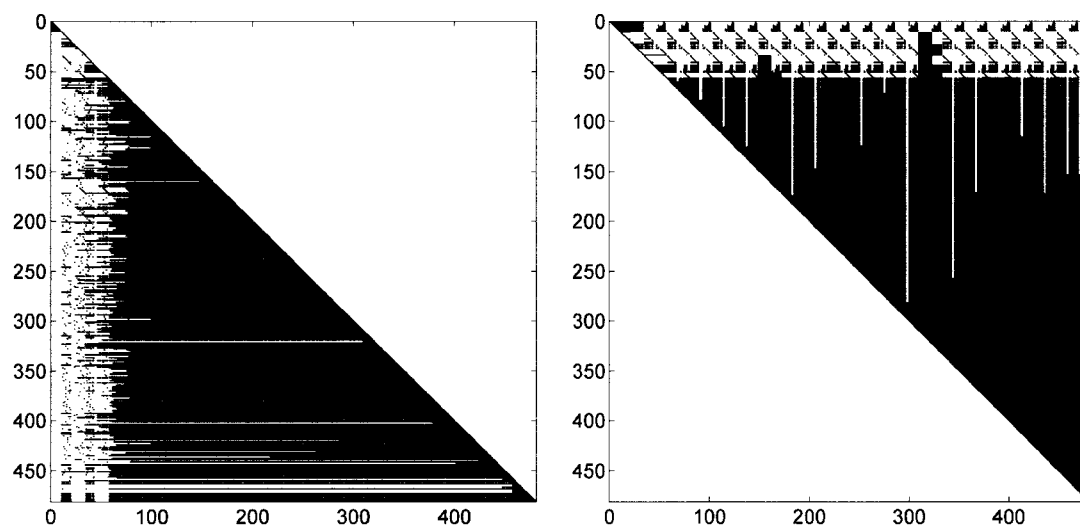


Figure 4.2: Lower triangular matrix (left) and upper triangular matrix (right) for factorization of LGL PS KKT matrix

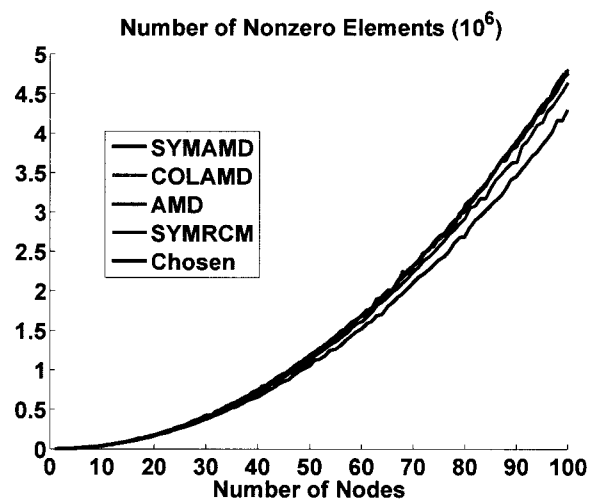


Figure 4.3: Comparison of different orderings of the PS KKT matrix. Chosen refers to the ordering given in (4.11), SYMAMD refers to MATLAB's symmetric approximate minimum degree permutation, COLAMD refers to MATLAB's column approximate minimum degree permutation, AMD refers to MATLAB's approximate minimum degree permutation, and SYMRCM refers to MATLAB's reverse Cuthill-McKee permutation.

Chapter 5

Direct Transcription using Multiple Shooting

5.1 Introduction

Shooting methods have originally been used for the solution of two-point boundary value problems (TPBVPs). Single shooting methods solve the initial value problem (IVP) corresponding to the TPBVP and adjust the initial data in such a way that the IVP solution satisfies the TPBVP. Due to instabilities of the IVP solution over longer time intervals, the TPBVP is broken up into a system of coupled TPBVP over smaller time subintervals. This leads to the Multiple Shooting (MS) method. MS methods for the solution of TPBVPs are described in [1, 2, 13, 51].

MS methods are also intensively used for transcribing optimal control problems

into nonlinear programs. See, e.g., [8, 32, 34, 33, 52]). One area that has used MS methods extensively is dynamic process optimization (see, e.g., [7, 32, 34].

5.2 Problem Formulation

Consider again the problem (2.1). This thesis considers a development of multiple shooting similar to that of Bock and Plitt [8].

First the controls u are approximated by piecewise polynomials. See Section 5.4. This approximation parameterizes the controls u by $\mathbf{q} \in \mathbb{R}^k$. After replacing the controls u by the parameterization \mathbf{q} , (2.1) can be written as follows (to simplify the notation, $l(x(t), u(\mathbf{q}; t), t)$ is written as $l(x(t), \mathbf{q}, t)$):

$$\min \int_{t_0}^{t_f} l(x(t), \mathbf{q}, t) dt + e(x(t_f), t_f) \quad (5.1a)$$

s.t.

$$\dot{x}(t) = f(x(t), \mathbf{q}, t), \quad t \in [t_0, t_f] \quad (5.1b)$$

$$a(x(t_0), t_0) = 0 \quad (5.1c)$$

$$b(x(t_f), t_f) = 0 \quad (5.1d)$$

$$g(x(t), \mathbf{q}, t) \leq 0, \quad t \in [t_0, t_f]. \quad (5.1e)$$

The interval $[t_0, t_f]$ is subdivided into subintervals

$$t_0 < t_1 < \dots < t_{N-1} < t_N = t_f,$$

and at each point t_i , $i = 0, \dots, N$, auxiliary states $\mathbf{s}_i \in \mathbb{R}^n$ are introduced. On each

subinterval, then, the initial value problem

$$\dot{x}(t) = f(x(t), \mathbf{q}, t), \quad t \in [t_i, t_{i+1}] \quad (5.2a)$$

$$x(t_i) = \mathbf{s}_i \quad (5.2b)$$

must be considered. Assuming that (5.2) has a unique solution $x_i(t; \mathbf{s}_i, \mathbf{q})$, the constraints (5.1b-d) have the solution

$$x(t) = x_i(t; \mathbf{s}_i, \mathbf{q}), \quad t \in [t_i, t_{i+1}], \quad i = 0, \dots, N-1$$

if and only if the auxiliary states $\mathbf{s}_i \in \mathbb{R}^n$, $i = 0, \dots, N$ satisfy

$$x_i(t_{i+1}; \mathbf{s}_i, \mathbf{q}) = \mathbf{s}_{i+1}, \quad i = 0, \dots, N-1, \quad (5.3a)$$

$$a(\mathbf{s}_0, t_0) = 0, \quad (5.3b)$$

$$b(\mathbf{s}_N, t_f) = 0. \quad (5.3c)$$

The integral term of the objective can be separated into

$$\sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} l(x(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q}, t) dt.$$

Each integral $\int_{t_i}^{t_{i+1}} l(x(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q}, t) dt$ can be computed by solving the auxiliary scalar IVP

$$\dot{L}(t) = l(x(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q}, t), \quad t \in [t_i, t_{i+1}] \quad (5.4a)$$

$$L(t_i) = 0. \quad (5.4b)$$

If L_i solves (5.4), then

$$\int_{t_i}^{t_{i+1}} l(x(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q}, t) dt = L_i(t_{i+1}).$$

Since the right hand side in (5.4) depends on the solution $x(t; \mathbf{s}_i, \mathbf{q})$ of (5.2), the IVPs (5.2), (5.4) are solved as one system of IVPs.

If (5.3) is satisfied, then the final time term in the objective reads

$$e(x(t_f), t_f) = e(\mathbf{s}_N, t_N).$$

Finally, the path constraints (5.1e) are enforced at the subinterval endpoints only.

That is, (5.1e) is replaced by

$$g(x(t_i), \mathbf{q}, t_i) \leq 0, \quad i = 0, \dots, N. \quad (5.5)$$

Thus, the optimal control problem (5.1), with (5.1e) replaced by (5.5), can be equivalently written as

$$\min \sum_{i=0}^{N-1} L_i(x(t_{i+1}; \mathbf{s}_i, \mathbf{q}), \mathbf{q}, t) + e(\mathbf{s}_N, t_N) \quad (5.6a)$$

s.t.

$$\mathbf{s}_{i+1} - x_i(t_{i+1}; \mathbf{s}_i, \mathbf{q}) = 0, \quad i = 0, \dots, N-1, \quad (5.6b)$$

$$a(\mathbf{s}_0, t_0) = 0 \quad (5.6c)$$

$$b(\mathbf{s}_N, t_N) = 0 \quad (5.6d)$$

$$g(\mathbf{s}_i, \mathbf{q}, t_i) \leq 0, \quad i = 0, \dots, N. \quad (5.6e)$$

The problem (5.6) is a nonlinear programming problem in the variables

$$\mathbf{y} = \begin{bmatrix} \mathbf{s}_0^T & \dots & \mathbf{s}_N^T & \mathbf{q}^T \end{bmatrix}^T \in \mathbb{R}^{n_y}$$

that can, again, be written in the form of (2.5). The functions of (2.5) are now defined by

$$J(\mathbf{y}) := \sum_{i=0}^{N-1} L_i(x(t_{i+1}; \mathbf{s}_i, \mathbf{q}), \mathbf{q}, t) + e(\mathbf{s}_N, t_N) \quad (5.7a)$$

$$F(\mathbf{y}) := \begin{bmatrix} \mathbf{s}_1 - x_0(t_1; \mathbf{s}_0, \mathbf{q}) \\ \mathbf{s}_2 - x_1(t_2; \mathbf{s}_1, \mathbf{q}) \\ \vdots \\ \mathbf{s}_N - x_{N-1}(t_N; \mathbf{s}_{N-1}, \mathbf{q}) \\ a(\mathbf{s}_0, t_0) \\ b(\mathbf{s}_N, t_N) \end{bmatrix}, \text{ and} \quad (5.7b)$$

$$G(\mathbf{y}) := \begin{bmatrix} g(\mathbf{s}_0, \mathbf{q}, t_0) \\ \vdots \\ g(\mathbf{s}_N, \mathbf{q}, t_N) \end{bmatrix}, \quad (5.7c)$$

where $n_y = (N+1)n + k$, $n_F = Nn + n_a + n_b$ and $n_G = (N+1)n_g$. Further, (5.7)

has equality constraint multipliers

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_0^T & \cdots & \boldsymbol{\lambda}_{N-1}^T & \boldsymbol{\nu}_0^T & \boldsymbol{\nu}_N^T \end{bmatrix}^T \quad (5.8)$$

and inequality constraint multipliers

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_0^T & \cdots & \boldsymbol{\mu}_N^T \end{bmatrix}^T, \quad (5.9)$$

where $\boldsymbol{\lambda}_i \in \mathbb{R}^n$, $i = 0, \dots, N-1$, and $\boldsymbol{\mu}_i \in \mathbb{R}^{n_g}$, $i = 0, \dots, N$, $\boldsymbol{\nu}_0 \in \mathbb{R}^{n_a}$, and $\boldsymbol{\nu}_N \in \mathbb{R}^{n_b}$, corresponding to the rows of (5.7b) and (5.7c). However, while (5.6) is a

finite dimensional NLP, the evaluation of the constraints (5.6b) require the solution of a system of ODEs (5.2) and (5.4).

5.3 Derivative Computations

5.3.1 First Order Derivatives

The Jacobians of the constraints (5.6b) and gradient of the sum in the objective (5.6a) require the computation of derivatives of the solution of initial value problems with respect to initial values and parameters.

The following results can be found in [26, Sec. I.14] (Theorems 14.1 and 14.3)

Theorem 5.3.1 *Let $x_i(\cdot; \mathbf{s}_i, \mathbf{q})$ be the solution of*

$$\dot{x}(t) = f(x(t), \mathbf{q}, t), \quad t \in [t_i, t_{i+1}] \quad (5.10a)$$

$$x(t_i) = \mathbf{s}_i \quad (5.10b)$$

and let $\mathcal{O} \subset \mathbb{R}^n \times \mathbb{R}^k \times \mathbb{R}$ be an open set such that $(x_i(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q}, t) \in \mathcal{O}$ for all $t \in [t_i, t_{i+1}]$.

(a) If the partial derivatives $D_x f$ and $D_{\mathbf{q}} f$ exist and are continuous in \mathcal{O} , then the partial derivative

$$Q_i(\cdot) = D_{\mathbf{q}} x_i(\cdot; \mathbf{s}_i, \mathbf{q})$$

exists, is continuous, and satisfies the differential equations

$$\dot{Q}(t) = D_x f(t, x_i(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q})Q(t) + D_{\mathbf{q}} f(t, x_i(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q}), \quad t \in [t_i, t_{i+1}] \quad (5.11a)$$

$$Q(t_i) = 0. \quad (5.11b)$$

(b) If the partial derivative $D_x f$ exists and is continuous in \mathcal{O} , then the solution $x_i(\cdot; \mathbf{s}_i, \mathbf{q})$ is differentiable with respect to \mathbf{s}_i and the derivative

$$S_i(\cdot) = D_{\mathbf{s}} x_i(\cdot; \mathbf{s}_i, \mathbf{q})$$

satisfies the differential equations

$$\dot{S}(t) = D_x f(t, x_i(t; \mathbf{s}_i, \mathbf{q}), \mathbf{q})S(t), \quad t \in [t_i, t_{i+1}] \quad (5.12a)$$

$$S(t_i) = I. \quad (5.12b)$$

If an ODE solver with adaptive time stepping is used to solve (5.10), then (5.10), (5.11), and (5.12) have to be solved simultaneously, i.e.,

$$\dot{x}(t) = f(x(t), \mathbf{q}, t) \quad (5.13a)$$

$$\dot{L}(t) = l(x(t), \mathbf{q}, t) \quad (5.13b)$$

$$\dot{Q}(t) = D_x f(x(t), \mathbf{q}, t)Q(t) + D_{\mathbf{q}} f(x(t), \mathbf{q}, t) \quad (5.13c)$$

$$\dot{S}(t) = D_x f(x(t), \mathbf{q}, t)S(t) \quad (5.13d)$$

$$\dot{L}_{\mathbf{s}}(t) = D_x l(x(t), \mathbf{q}, t)S(t) \quad (5.13e)$$

$$\dot{L}_{\mathbf{q}}(t) = D_x l(x(t), \mathbf{q}, t)Q(t) + D_{\mathbf{q}} l(x(t), \mathbf{q}, t) \quad (5.13f)$$

$$x(t_i) = \mathbf{s}_i, \quad (5.13g)$$

$$z(t_i) = 0, \quad (5.13h)$$

$$Q(t_i) = 0, \quad (5.13i)$$

$$S(t_i) = I, \quad (5.13j)$$

$$L_s(t_i) = 0, \quad (5.13k)$$

$$L_q(t_i) = 0. \quad (5.13l)$$

must be solved on each subinterval $t \in [t_i, t_{i+1}]$, $i = 0, \dots, N-1$, which is a system of $2n + nk + k + n^2 + 1$ ODEs in the unknowns $t \mapsto x(t) \in \mathbb{R}^n$, $t \mapsto L(t) \in \mathbb{R}$, $t \mapsto Q(t) \in \mathbb{R}^{n \times k}$, $t \mapsto S(t) \in \mathbb{R}^{n \times n}$, $t \mapsto L_s(t) \in \mathbb{R}^{1 \times n}$, and $t \mapsto L_q(t) \in \mathbb{R}^{1 \times k}$. See, e.g., [10, 35].

Under the assumptions of Theorem 5.3.1 the derivatives of the functions J, F, G defined in (5.7) can be computed. For example, the Jacobian of the equality constraints are given by

$$DF(\mathbf{y}) = \begin{bmatrix} -S_0(t_1) & I & 0 & \cdots & 0 & -Q_0(t_1) \\ 0 & -S_1(t_2) & I & \cdots & 0 & -Q_1(t_2) \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & -S_{N-1}(t_N) & I & -Q_{N-1}(t_N) \\ D_{\mathbf{s}_0}a & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & D_{\mathbf{s}_N}b & 0 \end{bmatrix}. \quad (5.14)$$

5.3.2 Second Order Derivatives

The second order derivatives of the constraints

$$x(t_{i+1}; \mathbf{s}_i, \mathbf{q}) - \mathbf{s}_{i+1} = 0, \quad i = 0, \dots, N-1,$$

which are needed for the computation of $\nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}$, can be computed exactly, in a manner similar to the computation of the first order derivatives. However, this significantly increases the number of differential equations that must be numerically integrated on each MS subinterval. Instead, quasi-Newton updates are used to approximate the Hessian.

Further, if it is assumed that the control is parameterized on a single subinterval by \mathbf{q}_i , $i = 0, \dots, N$ so that it does not appear in the definition of the parameterization on any other subinterval, then considering the Lagrangian of the problem (5.6), defined by

$$\begin{aligned} \mathcal{L} = & \sum_{i=0}^{N-1} L_i(x(t_{i+1}; \mathbf{s}_i, \mathbf{q}_i), \mathbf{q}_i, t) + e(\mathbf{s}_N, t_N) + \sum_{i=0}^{N-1} \lambda_i^T (\mathbf{s}_{i+1} - x_i(t_{i+1}; \mathbf{s}_i, \mathbf{q}_i)) \\ & + \sum_{i=0}^N \mu_i^T g(\mathbf{s}_i, \mathbf{q}_i, t_i) + \nu_0^T a(\mathbf{s}_0, t_0) + \nu_N^T b(\mathbf{s}_N, t_N), \end{aligned} \quad (5.15)$$

it is apparent that the second derivatives of all of the individual functions that compose (5.15) only depend on the intermediate initial values and control parameters of a single subinterval. This is a favorable property since the Hessian of (5.15) has a separable block structure; that is, the blocks of the Hessian corresponding to intermediate initial values and control parameters of different subintervals are identically zero.

The separable block structure then motivates the use of a block quasi-Newton update procedure, where the Hessian of (5.15) is updated on each subinterval individually. It is not clear whether this block update procedure has the same convergence properties as a regular quasi-Newton update. However, in practice fast, superlinear convergence has been observed when close to a solution.

The quasi-Newton update used in the analysis of this thesis is a damped BFGS update given by Nocedal and Wright [40, Section 18.3]. Let B^k be a symmetric and positive definite matrix, an approximation to a separable block of the Hessian of (5.15), and let the gradient of (5.15) with respect to the intermediate initial values and control parameters of the subinterval under consideration at the previous iterate and the current iterate be

$$\nabla_{\mathbf{y}_i} \mathcal{L}_i^k \text{ and } \nabla_{\mathbf{y}_i} \mathcal{L}_i^{k+1},$$

respectively, with $\mathbf{y}_i^k = \begin{bmatrix} (\mathbf{s}_i^k)^T & (\mathbf{q}_i^k)^T \end{bmatrix}^T$. Further, define

$$w_i^k = \mathbf{y}_i^{k+1} - \mathbf{y}_i^k \text{ and } v_i^k = (\nabla_{\mathbf{y}_i} \mathcal{L}_i^{k+1}) - (\nabla_{\mathbf{y}_i} \mathcal{L}_i^k).$$

Then the Hessian update can be defined by

$$B^{k+1} = B^k - \frac{B^k w^k (w^k)^T B^k}{(w^k)^T B^k w^k} + \frac{r^k (r^k)^T}{(v^k)^T r^k},$$

with $r^k = \theta^k v^k + (1 - \theta^k) B^k w^k$. θ^k is defined so that B^{k+1} is always positive definite by

$$\theta^k = \begin{cases} 1, & \text{if } (w^k)^T v^k \geq 0.2 (w^k)^T B^k w^k; \\ (0.8 (w^k)^T B^k w^k) / ((w^k)^T B^k w^k - (w^k)^T v^k), & \text{if } (w^k)^T v^k < 0.2 (w^k)^T B^k w^k, \end{cases}$$

[40, Section 18.3]. This is another favorable property that meets the assumptions of the merit function in Section 3.2.

Control parameterizations that elicit this structure are not difficult to surmise, and three of these are explored in the next section.

5.4 Control Parameterizations

The parameterization of u can be done in many ways. For instance, the techniques used in LGL PS collocation to interpolate the control could also be used in MS methods. However, such a parameterization would create an unnecessary explicit dependence between the parameters defining the control. Instead, the parameterization can be completed piecewise on each of the subintervals. These piecewise parameterizations do not themselves need to guarantee the continuity of the control or its derivatives at the junction of two subintervals. To guarantee these properties, the control parameterization on each subinterval could be defined analytically so that the control and derivatives at the junctions are continuous. However, a scheme like this would destroy the ability to approximate $\nabla_{\mathbf{y}\mathbf{y}}^2 \mathcal{L}$ with block quasi-Newton updates. To overcome this destruction, extra constraints can be imposed on the control parameterization. With this in mind, the three control parameterizations that have been used in this analysis will be detailed.

5.4.1 Piecewise Constant Control Parameterization

First, let $\mathbf{q} = \begin{bmatrix} \mathbf{a}_0^T & \dots & \mathbf{a}_N^T \end{bmatrix}^T$, $\mathbf{a}_i \in \mathbb{R}^m$ for $i = 0, \dots, N$, and define

$$u(t) = \mathbf{a}_i, \quad t \in [t_i, t_{i+1}), \quad \text{and } i = 0, \dots, N,$$

and $u(t_N) = \mathbf{a}_N$, such that $k = (N+1)m$. Clearly, defining nearly any extra equality constraints at the junctions would severely restrict the ability of this constant parameterization to effectively approximate the control. However, inequality constraints could be imposed to guarantee that the difference between controls on adjacent intervals is not too large, but these have not been studied in this thesis. The only extra constraint imposed on this constant parameterization is

$$\mathbf{a}_N - \mathbf{a}_{N-1} = 0. \tag{5.16}$$

Without this constraint, the KKT matrix tends to become singular if there is no constraint governing the behavior of the control at t_N . This is because the rows and columns of the Hessian of the Lagrangian corresponding to q_N will be zero, and the columns of the constraint Jacobian corresponding to these parameters will be zero.

5.4.2 Piecewise Linear Control Parameterization

Second, let $\mathbf{q} = \begin{bmatrix} \mathbf{a}_0^T & \mathbf{b}_0^T & \dots & \mathbf{a}_N^T & \mathbf{b}_N^T \end{bmatrix}^T$, $\mathbf{a}_i \in \mathbb{R}^m$ and $\mathbf{b}_i \in \mathbb{R}^m$ for $i = 0, \dots, N$, and let the control parameterization be

$$u(t) = \mathbf{a}_i t + \mathbf{b}_i, \quad t \in [t_0, t_f], \quad i = 0, \dots, N,$$

so that now $k = 2(N + 1)m$. To impose the continuity of this linear control parameterization, the constraints

$$(\mathbf{a}_{i+1} - \mathbf{a}_i)t_{i+1} + (\mathbf{b}_{i+1} - \mathbf{b}_i) = 0, \quad i = 0, \dots, N - 1 \quad (5.17)$$

are defined.

5.4.3 Piecewise Cubic Control Parameterization

Finally, let $\mathbf{q} = \begin{bmatrix} \mathbf{a}_0^T & \mathbf{b}_0^T & \mathbf{c}_0^T & \mathbf{d}_0^T & \dots & \mathbf{a}_N^T & \mathbf{b}_N^T & \mathbf{c}_N^T & \mathbf{d}_N^T \end{bmatrix}^T$, $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}_i \in \mathbb{R}^m$, for $i = 0, \dots, N$, and define

$$u(t) = \mathbf{a}_i t^3 + \mathbf{b}_i t^2 + \mathbf{c}_i t + \mathbf{d}_i, \quad t \in [t_0, t_f], \quad i = 0, \dots, N,$$

where $k = 4(N + 1)m$. In this case, the constraints imposed in the analysis contained in this thesis are on the continuity of the control and its derivative. That is,

$$(\mathbf{a}_{i+1} - \mathbf{a}_i)t_{i+1}^3 + (\mathbf{b}_{i+1} - \mathbf{b}_i)t_{i+1}^2 + (\mathbf{c}_{i+1} - \mathbf{c}_i)t_{i+1} + (\mathbf{d}_{i+1} - \mathbf{d}_i) = 0, \text{ and} \quad (5.18a)$$

$$\text{and } 3(\mathbf{a}_{i+1} - \mathbf{a}_i)t_{i+1}^2 + 2(\mathbf{b}_{i+1} - \mathbf{b}_i)t_{i+1} + (\mathbf{c}_{i+1} - \mathbf{c}_i) = 0, \quad (5.18b)$$

for $i = 0, \dots, N - 1$.

5.5 Well-posedness and Convergence of the MS Transcription

The MS transcription is performed in several stages. In the first step a control parameterization is applied to approximate the original optimal control problem (2.1)

by (5.1). For piecewise constant controls Goh and Teo [18] provide convergence results for the convergence of the parameterized control to a solution of (2.1) as the parameterization is refined. See also [53] and [43].

If (5.2) are integrated exactly, then a solution (x, \mathbf{q}) of the optimal control problem (5.1) with (5.1e) replaced by (5.5) is equivalent to a solution $(\mathbf{s}_0, \dots, \mathbf{s}_N, \mathbf{q})$ of the NLP (5.6) in the following way. If (x, \mathbf{q}) solves (5.1) with (5.1e) replaced by (5.5), then $(\mathbf{s}_0 = x(t_0), \dots, \mathbf{s}_N = x(t_N), \mathbf{q})$ solves (5.6). On the other hand, if $(\mathbf{s}_0, \dots, \mathbf{s}_N, \mathbf{q})$ solves (5.6), then (x, \mathbf{q}) solves (5.1) with (5.1e) replaced by (5.5), where

$$x(t) = x_i(t; \mathbf{s}_i, \mathbf{q}), \quad t \in [t_i, t_{i+1}], \quad i = 0, \dots, N-1$$

and x_i solves (5.2). Moreover, the Lagrange multipliers $\boldsymbol{\lambda}_0, \dots, \boldsymbol{\lambda}_{N-1}$ corresponding to the equality constraints in (5.6) are related to the adjoint variables λ corresponding to the dynamics (5.1b) in the optimal control problem (5.1) with (5.1e) replaced by (5.5) by the identity $\boldsymbol{\lambda}_i = \lambda(t_i)$. See Grimm and Markl [24]. Further, Grimm and Markl [24, Section 5.2] suggest that the Lagrange multipliers $\boldsymbol{\mu}$ belonging to the function G of (5.7) are proportional to the path constraint multipliers for the continuous problem (2.1), that is

$$\frac{\boldsymbol{\mu}_i}{(t_{i+1} - t_i)} \rightarrow \tilde{\boldsymbol{\mu}}_i,$$

when the control is parameterized by a piecewise constant function like that of Section 5.4.1. This result is originally due to the proof of Grimm [23].

5.6 Structure and Sparsity of the KKT Systems

The same methods that can be used to solve the NLP formed by transcribing (2.1) with LGL PS collocation can also be used to solve the NLP resulting from an MS transcription. This again leads to the construction and factorization of the KKT matrix. The structure for the KKT matrix resulting from an MS transcription (cubic control parameterization) with the ordering of variables

$$\begin{bmatrix} \boldsymbol{\nu}_0^T & \mathbf{s}_0^T & \mathbf{q}_0^T & \boldsymbol{\lambda}_0^T & \mathbf{s}_1^T & \mathbf{q}_1^T & \boldsymbol{\lambda}_1^T & \cdots & \mathbf{s}_{N-1}^T & \mathbf{q}_{N-1}^T & \boldsymbol{\lambda}_{N-1}^T & \mathbf{s}_N^T & \mathbf{q}_N^T & \boldsymbol{\nu}_N^T \end{bmatrix}^T \quad (5.19)$$

is shown in figure 5.1. Each of the $\boldsymbol{\lambda}_i$, $i = 1, \dots, N-1$, are the multipliers corresponding to the constraints

$$\mathbf{s}_{i+1} - x_i(t_{i+1}; \mathbf{s}_i, \mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}_i) = 0,$$

$$(\mathbf{a}_{i+1} - \mathbf{a}_i)t_{i+1}^3 + (\mathbf{b}_{i+1} - \mathbf{b}_i)t_{i+1}^2 + (\mathbf{c}_{i+1} - \mathbf{c}_i)t_{i+1} + (\mathbf{d}_{i+1} - \mathbf{d}_i) = 0, \text{ and}$$

$$3(\mathbf{a}_{i+1} - \mathbf{a}_i)t_{i+1}^2 + 2(\mathbf{b}_{i+1} - \mathbf{b}_i)t_{i+1} + (\mathbf{c}_{i+1} - \mathbf{c}_i) = 0,$$

so that $\boldsymbol{\lambda}_i \in \mathbb{R}^{n+2m}$. Again, only half of this matrix needs to be stored since it is symmetric.

The KKT systems that arise in MS discretizations of optimal control problems can be solved by structured direct methods that are tailored to the structure introduced by MS. See, e.g., [49, 50, 32, 34]. In this thesis a sparse LU decomposition is applied.

When factored into lower- and upper- triangular matrices by Gaussian elimination with pivoting, these matrices are sparse for MS transcription, and require a similar amount of storage space to that of the KKT matrix itself. These triangular matrices

are shown in figure 5.2. A comparison of the required storage for MS transcription with constant, linear, and cubic control parameterizations as well as LGL PS collocation for various N is given in figure 5.3. Clearly, for any of the given control parameterizations, MS transcription has a distinct advantage over PS collocation when storing the KKT matrices, particularly as N increases.

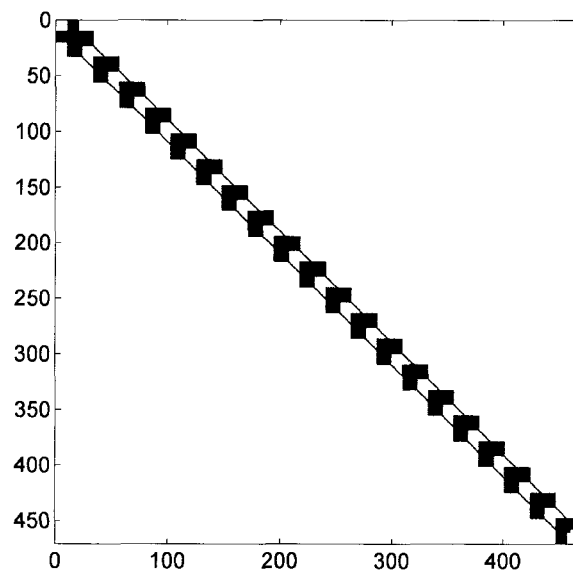


Figure 5.1: Sparsity pattern for MS KKT matrix with $N = 20$, $n = n_a = n_b = 10$, and $m = 3$

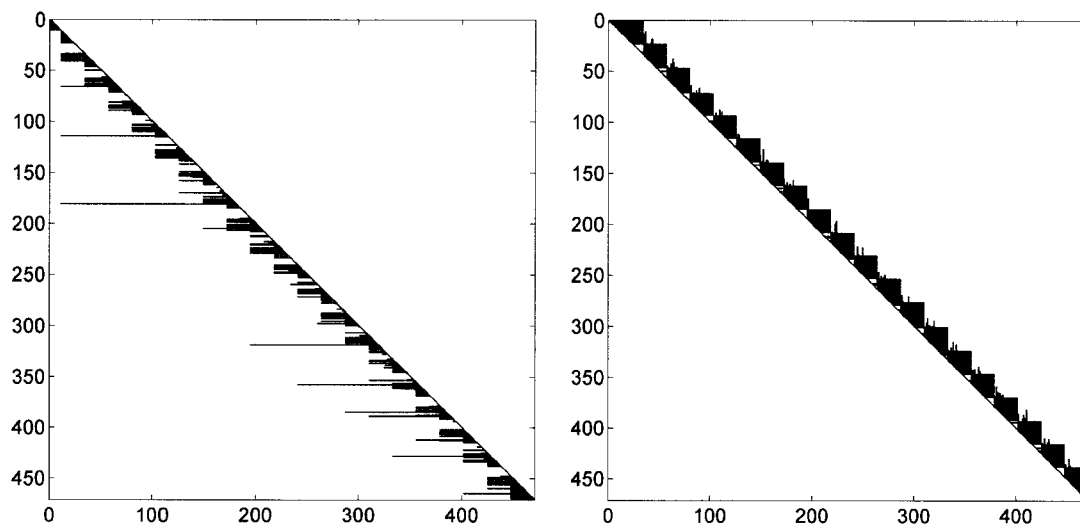


Figure 5.2: Lower triangular matrix (left) and upper triangular matrix (right) for factorization of MS KKT matrix

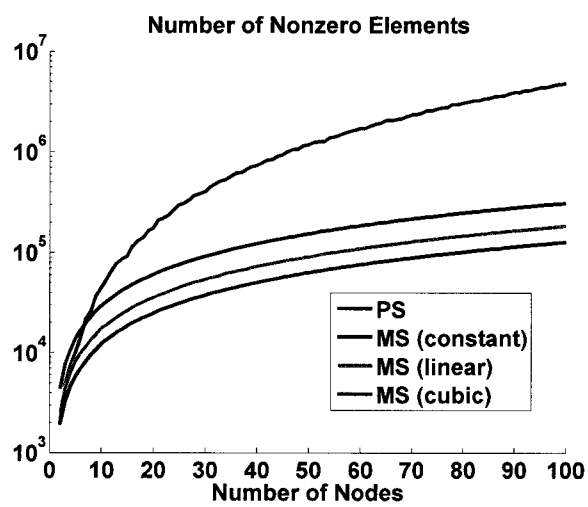


Figure 5.3: Comparison of storage requirements for LGL PS collocation and MS constant, linear, and cubic transcriptions with $N = 20$, $n = n_a = n_b = 10$, and $m = 3$

Chapter 6

Numerical Examples

6.1 Introduction

To illustrate the consistency of the numerical solution of the direct problem with that of the indirect problem, the following simple examples, chosen because of the availability of analytical solutions for the states, control, adjoints, and multipliers, were considered. Let \mathbf{s}_i , \mathbf{q}_i , $\boldsymbol{\lambda}_i$, and $\boldsymbol{\mu}_i$ represent the optimal numerical solution of a given example at node t_i found by the LGL PS method or the MS method with a constant control parameterization, satisfying the KKT conditions of Theorem 2.3.1. Also, let $x^*(t_i)$, $u^*(t_i)$, $\lambda^*(t_i)$, and $\mu^*(t_i)$ be the optimal analytical solution to a given example, which satisfies the necessary conditions of Theorem 2.2.1, evaluated at the transcription points t_i . Then define the state, control, adjoint, and multiplier errors

to be

$$x^{\text{error}}(t_i) = |x^*(t_i) - \mathbf{s}_i|, \quad (6.1)$$

$$u^{\text{error}}(t_i) = |u^*(t_i) - \mathbf{q}_i|, \quad (6.2)$$

$$\lambda^{\text{error}}(t_i) = |\lambda^*(t_i) - \boldsymbol{\lambda}_i|, \text{ and} \quad (6.3)$$

$$\mu^{\text{error}}(t_i) = |\mu^*(t_i) - \boldsymbol{\mu}_i|, \quad (6.4)$$

respectively.

The numerical solutions discussed here are found by interfacing the objective and constraint functions defined in Appendix A with MATLABs `fmincon` on a PC running Microsoft Windows XP Professional, Version 2002, Service Pack 3, with an AMD Athlon 64 X2 Dual Core Processor 5600+ at 2.91 GHz and 2.0 GB of RAM, and under the MATLAB 7.6.0 (R2008a) environment. `fmincon` was chosen for these comparisons so that they are more easily repeatable, without the need for the optimization algorithm described in this thesis. The default options of `fmincon` were altered to allow the definition of the objective gradient and constraint Jacobians, to specify the algorithm as interior-point, and to set stricter tolerances on the exit criteria of the algorithm. These options are as follows:

- 'GradObj' set to 'on'
- 'GradConstr' set to 'on'
- 'Algorithm' set to 'interior-point'

- 'TolX' set to 1×10^{-16}
- 'TolFun' set to 5×10^{-8}

Also, these examples serve as a way of highlighting the ability of the MS transcription to approximate the states, controls, adjoints, and multipliers of an OCP as well as LGL PS collocation, despite the spectral convergence of the PS method highlighted by Gong, et al. [20]. This becomes much more apparent when there are discontinuities in any of these variables or their derivatives.

6.2 Linear-Quadratic Example

This simple example is adapted from Hager [25] and is given by

$$\min_{x(t), u(t)} \int_0^1 x(t)^2 + \frac{1}{2} u(t)^2 dt \quad (6.5a)$$

$$\text{s.t. } \dot{x}(t) = \frac{1}{2}x(t) + u(t), \quad t \in [0, 1] \quad (6.5b)$$

$$x(0) = 1 \quad (6.5c)$$

The analytical solution is defined by

$$x^*(t) = \frac{2\exp(3t) + \exp(3)}{\exp(3t/2)(2 + \exp(3))}, \quad (6.6a)$$

$$u^*(t) = \frac{2(\exp(3t) - \exp(3))}{\exp(3t/2)(2 + \exp(3))}, \text{ and} \quad (6.6b)$$

$$\lambda^*(t) = -\frac{2(\exp(3t) - \exp(3))}{\exp(3t/2)(2 + \exp(3))} \quad (6.6c)$$

These solutions are shown in figures 6.1, 6.3, and 6.5. The smoothness of the analytical solution of this example leads to significantly better approximations of the states,

controls and adjoints by LGL PS collocation. This is due to the optimal solution being infinitely many times differentiable. When the solution of an optimal control problem is not smooth this is not the case and can be witnessed in the following examples.

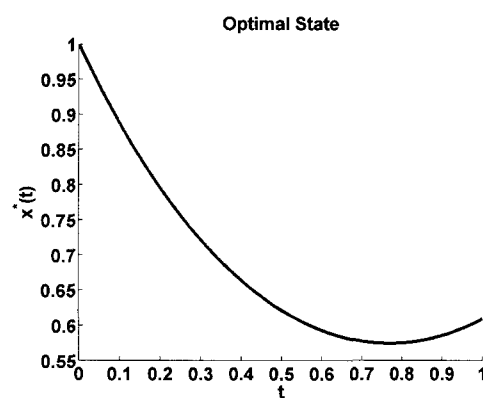


Figure 6.1: Optimal state for example (6.5).

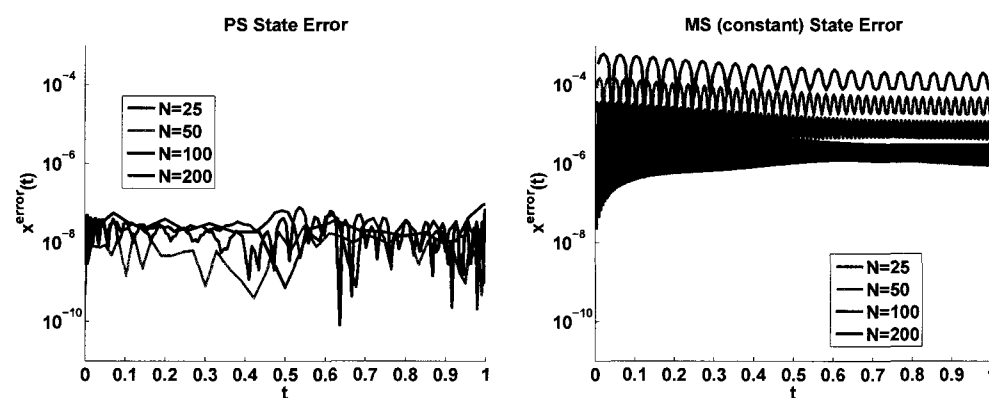


Figure 6.2: State error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.1) for the example (6.5).

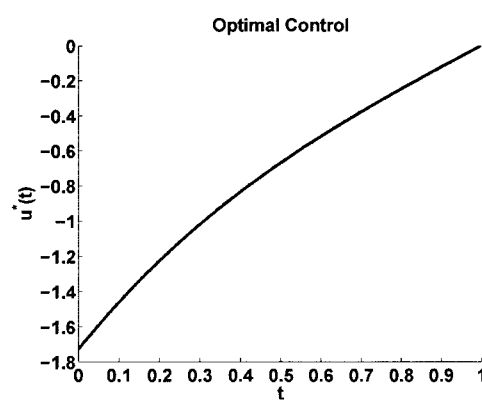


Figure 6.3: Optimal control for example (6.5).

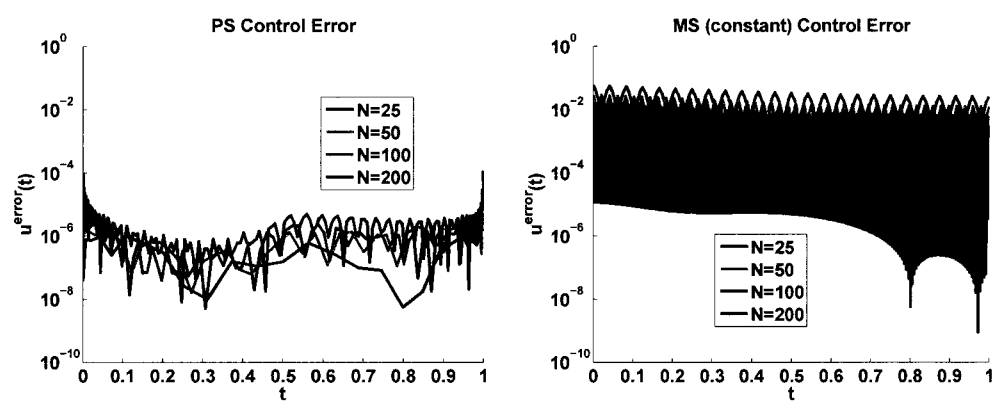


Figure 6.4: Control error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.2) for the example (6.5).

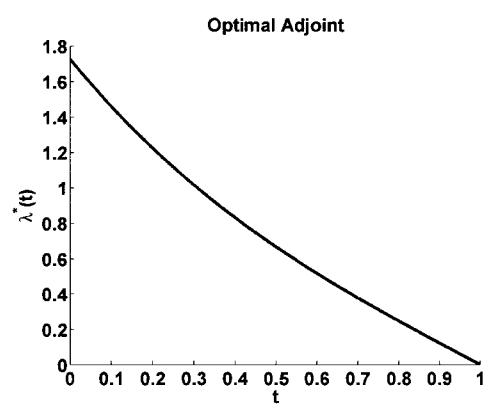


Figure 6.5: Optimal adjoint for example (6.5).

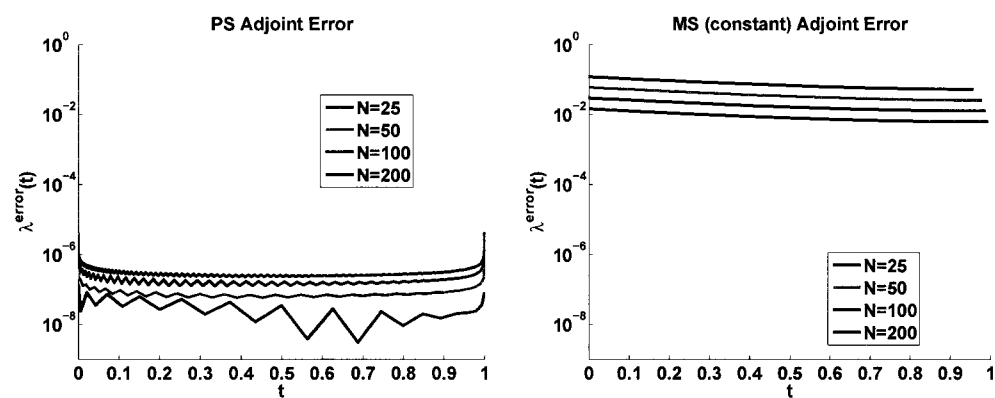


Figure 6.6: Adjoint error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.3) for the example (6.5).

6.3 Piecewise-Constant Control Example

$$\min_{x(t), u(t)} \int_0^3 x(t) dt \quad (6.7a)$$

$$\text{s.t. } \dot{x}(t) = u(t), \quad t \in [0, 3] \quad (6.7b)$$

$$x(0) = 1 \quad (6.7c)$$

$$-1 \leq u(t) \leq 1, \quad t \in [0, 3] \quad (6.7d)$$

$$x(t) \geq 0, \quad t \in [0, 3] \quad (6.7e)$$

$$x(3) = 1 \quad (6.7f)$$

This example is adapted from Hartl, et al. [27], and has an analytical solution defined by

$$x^*(t) = \begin{cases} 1 - t, & t \in [0, 1); \\ 0, & t \in [1, 2]; \\ t - 2, & t \in (2, 3]; \end{cases} \quad (6.8a)$$

$$u^*(t) = \begin{cases} -1, & t \in [0, 1); \\ 0, & t \in [1, 2]; \\ 1, & t \in (2, 3]; \end{cases} \quad (6.8b)$$

$$\lambda^*(t) = \begin{cases} 1 - t, & t \in [0, 1); \\ 0, & t \in [1, 2]; \\ 2 - t, & t \in (2, 3]; \text{ and} \end{cases} \quad (6.8c)$$

$$\mu_1^*(t) = \begin{cases} 1-t, \\ 0, \\ 0, \end{cases} \quad \mu_2^*(t) = \begin{cases} 0, \\ 0, \\ t-2, \end{cases} \quad \mu_3^*(t) = \begin{cases} 0, & t \in [0, 1); \\ 1, & t \in [1, 2]; \\ 0, & t \in (2, 3], \end{cases} \quad (6.8d)$$

where $\mu_1^*(t)$, $\mu_2^*(t)$, and $\mu_3^*(t)$ are the multipliers corresponding to the inequalities $u(t) \geq -1$, $u(t) \leq 1$, and $x(t) \geq 0$, respectively. These solutions are shown along with the errors of the numerical solutions in figures 6.7–6.14. Only one of the multipliers is shown; however, the others elicit similar behavior in the errors of the numerical solutions. The discontinuities of the control and the derivatives of the states, adjoints, and multipliers degrade the ability of LGL PS collocation to approximate the optimal state, control, adjoint, and multipliers, particularly at the times $t = 1$ and $t = 2$ where the discontinuities exist. These discontinuities also have an effect on the MS transcription, and introduce a significant amount of error at these points. A potential way to achieve similar accuracy to the example of Section 6.2 would be to discretize the problem on three separate intervals, so that on each interval the solution is smooth. However, in cases where the analytical solution is unknown, and the ability to anticipate the location of discontinuities in the solution is not available, such a scheme would be ill-conceived.

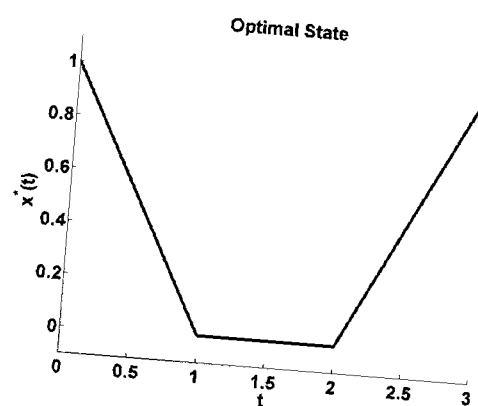


Figure 6.7: Optimal state for example (6.7).

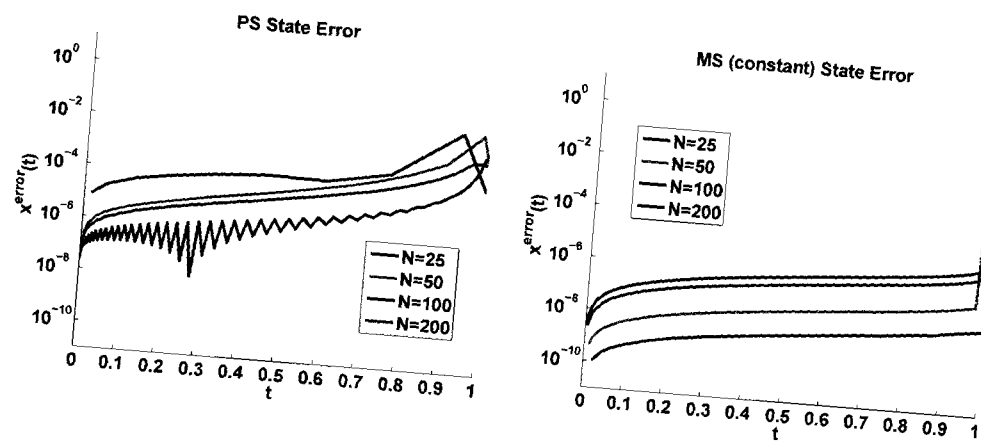


Figure 6.8: State error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.1) for the example (6.7).

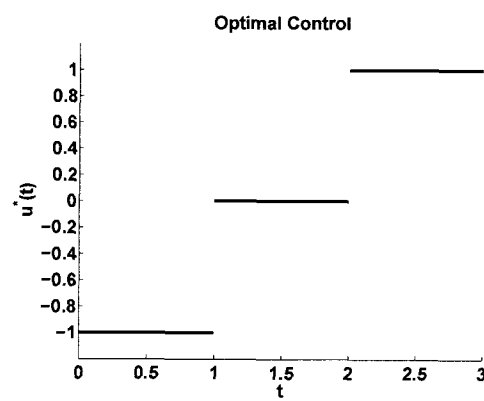


Figure 6.9: Optimal control for example (6.7).

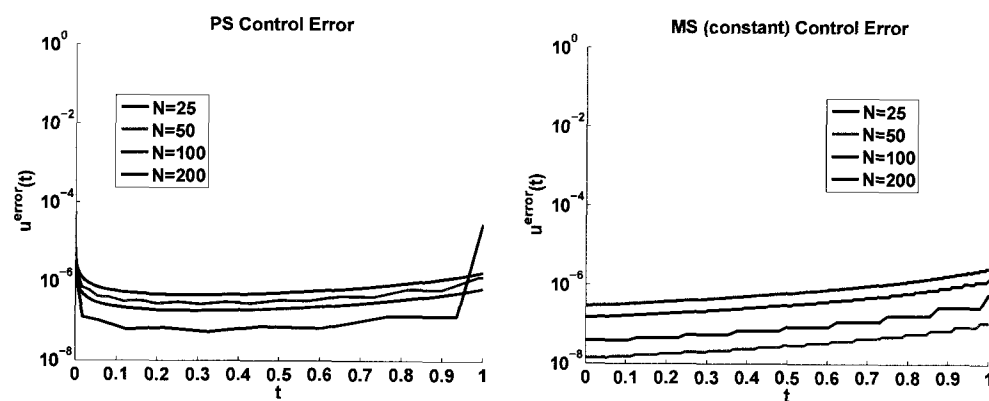


Figure 6.10: Control error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.2) for the example (6.7).

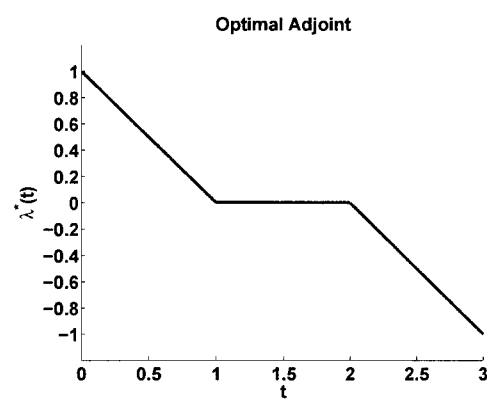


Figure 6.11: Optimal adjoint for example (6.7).

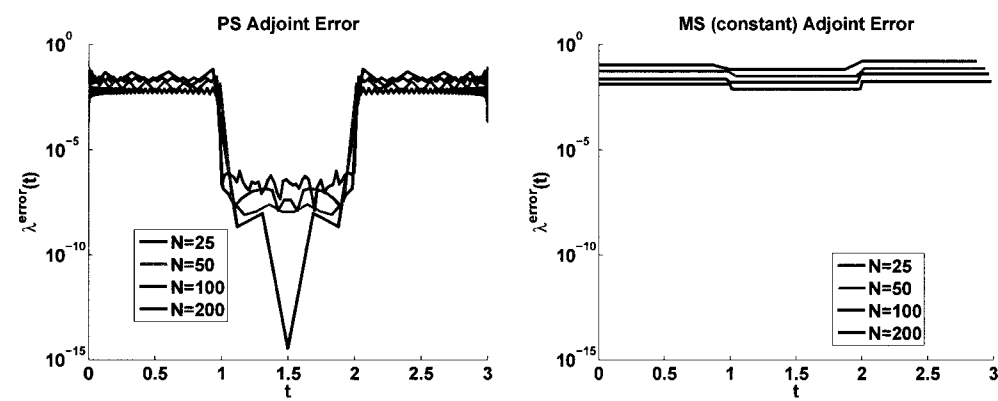


Figure 6.12: Adjoint error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.3) for the example (6.7).

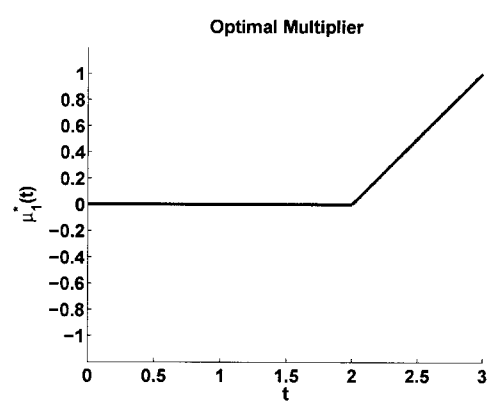


Figure 6.13: Optimal multiplier ($\mu_1(t)$) for example (6.7).

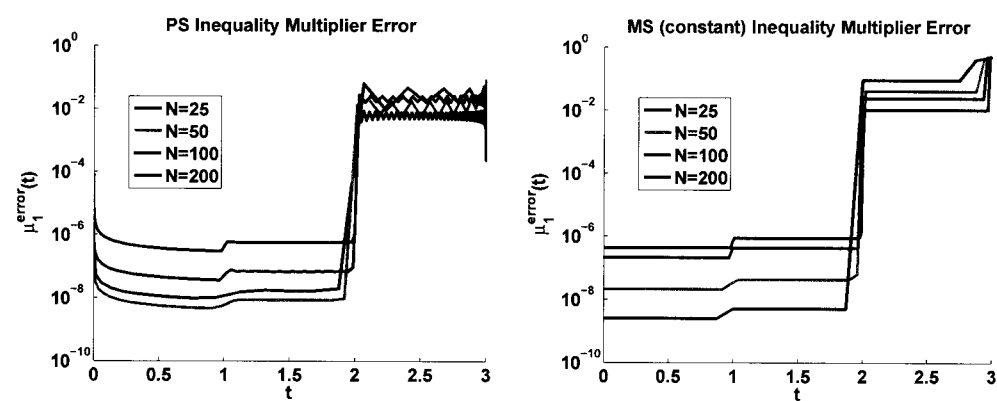


Figure 6.14: Multiplier ($\mu_1(t)$) error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.4) for the example (6.7).

6.4 A Final Example

$$\min_{x_1(t), x_2(t), u(t)} \int_0^3 2x_1(t) dt \quad (6.9a)$$

s.t.

$$\dot{x}_1(t) = x_2(t), \quad t \in [0, 3] \quad (6.9b)$$

$$\dot{x}_2(t) = u(t), \quad t \in [0, 3] \quad (6.9c)$$

$$x_1(0) = 2 \quad (6.9d)$$

$$x_2(0) = 0 \quad (6.9e)$$

$$-2 \leq u(t) \leq 2, \quad t \in [0, 3] \quad (6.9f)$$

$$x_1(t) \geq -1, \quad t \in [0, 3] \quad (6.9g)$$

Define $\tau = \sqrt{6}$ and $\sigma = \sqrt{6}/2$. This example is adapted from Hartl, et al. [27], and has an analytical solution defined by

$$x_1^*(t) = \begin{cases} 2 - t^2, \\ 2 + t^2 + 2\sigma^2 - 4\sigma t, \\ -1, \end{cases} \quad x_2^*(t) = \begin{cases} -2t, & t \in [0, \sigma); \\ 2(t - 2\sigma), & t \in [\sigma, \tau); \\ 0, & t \in [\tau, 3]; \end{cases} \quad (6.10a)$$

$$u^*(t) = \begin{cases} -2, & t \in [0, \sigma); \\ 2, & t \in [\sigma, \tau); \\ 0, & t \in [\tau, 3]; \end{cases} \quad (6.10b)$$

$$\lambda_1^*(t) = \begin{cases} 3\sigma - 2t, \\ 3\sigma - 2t, \\ 0, \end{cases} \quad \lambda_1^*(t) = \begin{cases} 2\sigma^2 - 3\sigma t + t^2, & t \in [0, \sigma); \\ 2\sigma^2 - 3\sigma t + t^2, & t \in [\sigma, \tau); \\ 0, & t \in [\tau, 3]; \end{cases} \quad \text{and} \quad (6.10c)$$

$$\mu_1^*(t) = \begin{cases} 2\sigma^2 - 3\sigma t + t^2, \\ 0, \\ 0, \end{cases} \quad \mu_2^*(t) = \begin{cases} 0, \\ 2\sigma^2 - 3\sigma t + t^2, \\ 0, \end{cases} \quad \mu_3^*(t) = \begin{cases} 0, & t \in [0, \sigma); \\ 0, & t \in [\sigma, \tau); \\ 2, & t \in [\tau, 3]; \end{cases} \quad (6.10d)$$

where $\lambda_1^*(t)$ and $\lambda_2^*(t)$ are the adjoints corresponding to the differential equations for $x_1(t)$ and $x_2(t)$, and $\mu_1^*(t)$, $\mu_2^*(t)$, and $\mu_3^*(t)$ are the multipliers corresponding to the inequalities $u(t) \leq 2$, $u(t) \geq -2$, and $x(t) \geq -1$, respectively. These solutions are shown along with the errors of the numerical solution in figures 6.15–6.22. Again, because of the points of discontinuity, $t = \sigma$ and $t = \tau$, in either the states, controls, adjoints, multipliers, or their derivatives the approximation of the optimal states, controls, adjoints, and multipliers degrades significantly for the LGL PS approximation, and likewise for the MS transcription. LGL PS collocation has a particular problem with approximating the adjoints at the boundaries, a defect highlighted by Benson [4, Section 3.3.2]. This is especially apparent in figure 6.20.

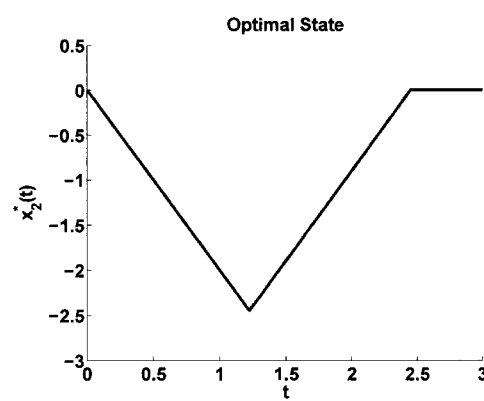


Figure 6.15: Optimal state ($x_2(t)$) for example (6.9).

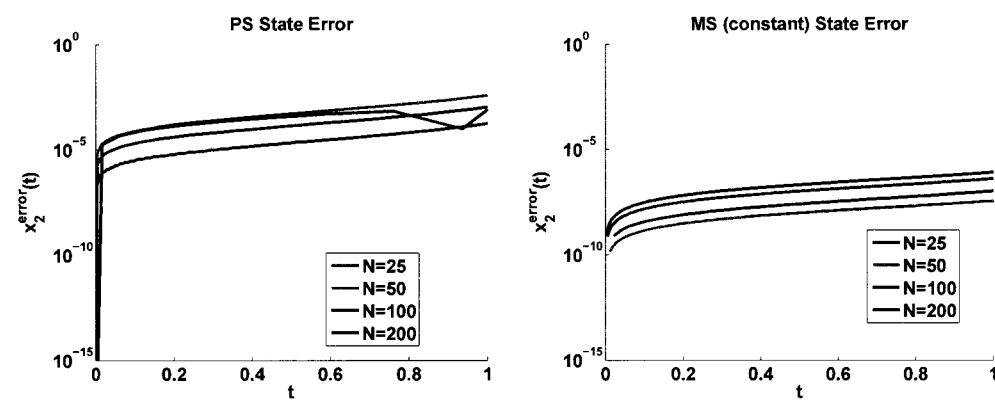


Figure 6.16: State ($x_2(t)$) error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.1) for the example (6.9).

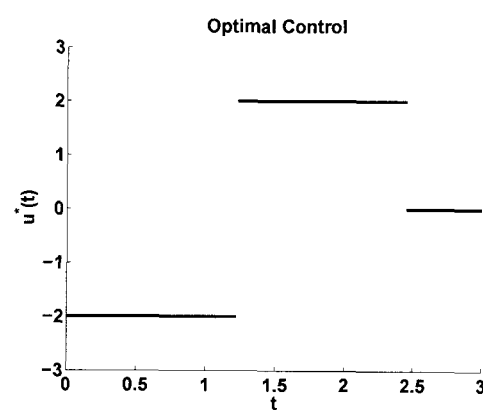


Figure 6.17: Optimal control for example (6.9).

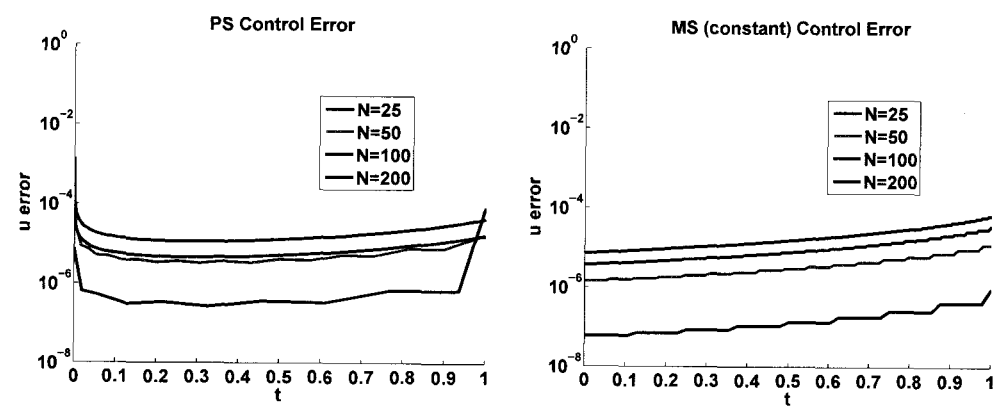


Figure 6.18: Control error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.2) for the example (6.9).

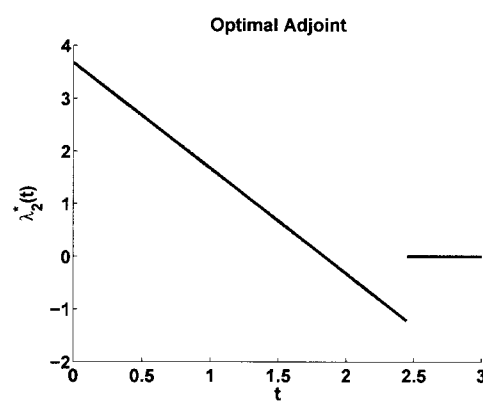


Figure 6.19: Optimal adjoint ($\lambda_1(t)$) for example (6.9).

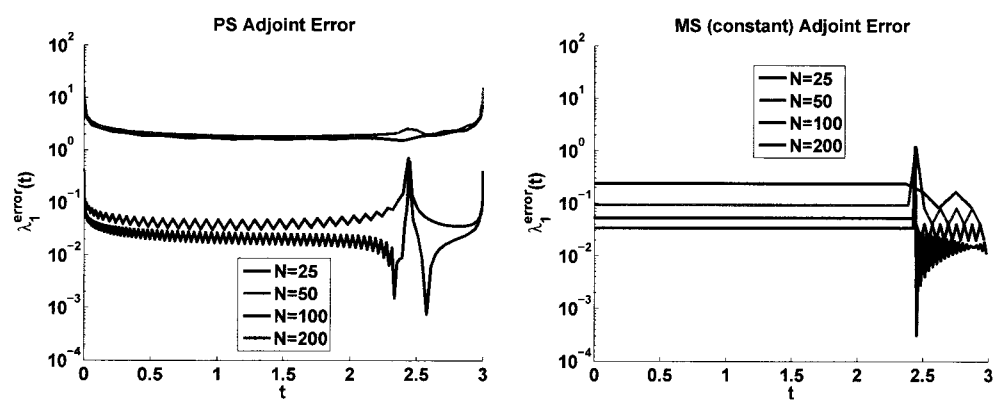


Figure 6.20: Adjoint ($\lambda_1(t)$) error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.3) for the example (6.9).

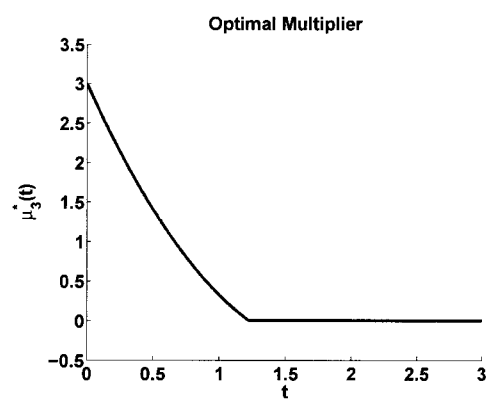


Figure 6.21: Optimal multiplier ($\mu_1(t)$) for example (6.9).

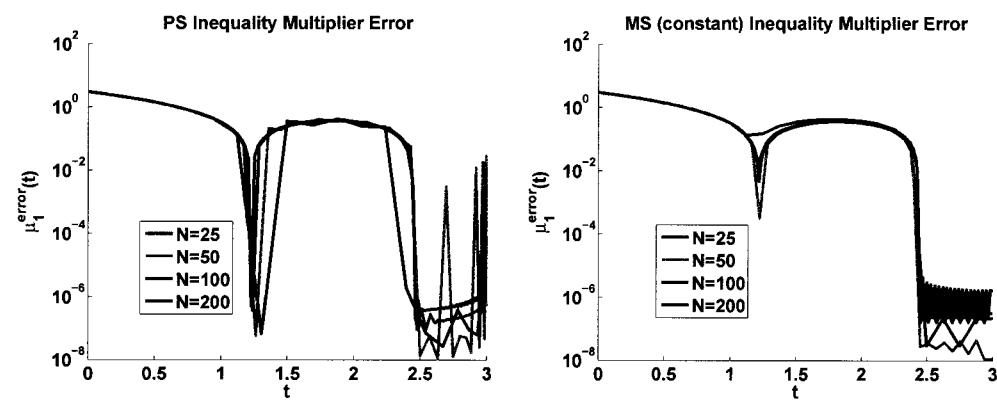


Figure 6.22: Multiplier ($\mu_1(t)$) error for LGL PS collocation (left) and MS (constant) transcription (right). Error computed by (6.4) for the example (6.9).

Chapter 7

A Practical Application: The ISS

ZPM

7.1 Introduction

The difficulty of finding an OCP that is complicated to solve numerically, but has a numerical solution, paired with the desire to understand the capability of LGL PS collocation and MS methods on the ISS ZPMs, has prompted the testing of the algorithm described in this thesis on a particular ZPM. The particular ZPM under consideration is that of the first ZPM flight test demonstration, since it has been proven to work in practice [6, Chapter 5].

7.2 International Space Station Dynamics

The International Space Station (ISS) dynamics, assuming circular low Earth orbit, can be described by a set of differential equations using the rotational rate $\omega : \mathbb{R} \rightarrow \mathbb{R}^3$ of the body reference frame with respect to an inertial frame, the total three-axis momentum of the ISS Control Moment Gyroscopes (CMGs) $h : \mathbb{R} \rightarrow \mathbb{R}^3$, and the CMG control torque $\tau_c : \mathbb{R} \rightarrow \mathbb{R}^3$ [6, Chapter 2]. The states $\omega(t)$, $h(t)$, and $\tau_c(t)$ are expressed in the body reference frame, which is fixed with respect to the ISS with the positive x-axis directed toward the nose and the positive y-axis along the main truss pointing starboard. Also, the dynamics are specified with respect to the orientation of the body reference frame with respect to the Local Vertical Local Horizontal (LVLH) frame, where the positive x-axis points in the direction of the velocity vector, the positive z-axis towards the Earth, and the y-axis perpendicular to the orbit plane. These reference frames are depicted in figures 7.1 and 7.2. The optimizations performed in this thesis will use the orientation described by a quaternion, $q : \mathbb{R} \rightarrow \mathbb{R}^4$; however, this orientation can be described in other ways as well. For instance, the orientation could be described using Euler angles or any sequence of rotations about the axes of the body reference frame [57, Section 5.5.2].

Quaternions, also known as Euler parameters, can be described by an angle, θ :

$\mathbb{R} \mapsto \mathbb{R}$, and a unit vector, $\hat{e}(t) : \mathbb{R} \mapsto \mathbb{R}^3$, pointing along the rotation axis:

$$q(t) = \begin{bmatrix} q_0(t) \\ q_1(t) \\ q_2(t) \\ q_3(t) \end{bmatrix} = \begin{bmatrix} \cos(\frac{\theta(t)}{2}) \\ \hat{e}_i(t) \sin(\frac{\theta(t)}{2}) \\ \hat{e}_j(t) \sin(\frac{\theta(t)}{2}) \\ \hat{e}_k(t) \sin(\frac{\theta(t)}{2}) \end{bmatrix}$$

[57, Section 5.4.1]. Further, quaternions are constrained to be unit norm, that is, $\|q(t)\|_2 = 1$. With these definitions, identities that are often described by other rotations sequences can be written relative to quaternions. For instance, the direction cosine matrix, $C : \mathbb{R}^4 \mapsto \mathbb{R}^{3 \times 3}$, can be written as

$$C(q(t)) = \begin{bmatrix} 1 - 2(q_2(t)^2 + q_3(t)^2) & 2(q_1(t)q_2(t) + q_0(t)q_3(t)) & 2(q_1(t)q_3(t) - q_0(t)q_2(t)) \\ 2(q_1(t)q_2(t) - q_0(t)q_3(t)) & 1 - 2(q_1(t)^2 + q_3(t)^2) & 2(q_2(t)q_3(t) + q_0(t)q_1(t)) \\ 2(q_1(t)q_3(t) + q_0(t)q_2(t)) & 2(q_2(t)q_3(t) - q_0(t)q_1(t)) & 1 - 2(q_1(t)^2 + q_2(t)^2) \end{bmatrix},$$

where the convention C_i will be used to denote the i^{th} column of C [57, Section 5.4.1].

Define parameters $J \in \mathbb{R}^{3 \times 3}$ and $\omega_{orb} \in \mathbb{R}$ to represent the moment of inertia of the spacecraft in the body reference frame and the orbital rate of the space station around the Earth, respectively. These two parameters can vary with time, but for simplicity are not described by their respective differential equations or other functional relationships.

Bhatt cites two of the prevailing disturbance torques acting on the ISS as the gravity gradient torque and aerodynamic torque [6, Sections 2.4.1 and 2.4.2]; however,

only the first of these is treated in this numerical example. The gravity gradient torque, written with respect to $q(t)$, is defined as [29]

$$\tau_{gg}(q(t)) = 3\omega_{orb}^2 C_3(q(t)) \times (JC_3(q(t))).$$

The representation of the states relative to the body reference frame and the dynamics relative to the LVLH reference frame requires a description of the LVLH rate at the current orientation defined by $q(t)$. This is given by [6, Section 2.2.1]

$$\omega_0(q(t)) = -\omega_{orb} C_2(q(t)).$$

With these definitions in mind, the rotational dynamics of a spacecraft in a circular orbit are given by [57, 6, 29]

$$\dot{\omega}(t) = J^{-1}[-\omega(t) \times (J\omega(t) + h(t)) - \tau_c(t) + \tau_{gg}(q(t))],$$

$$\dot{h}(t) = \tau_c(t), \text{ and}$$

$$\dot{q}(t) = \frac{1}{2}T(q(t)) \begin{bmatrix} 0 & (\omega(t)^T - \omega_0(q(t))^T) \end{bmatrix}^T,$$

where $T(q(t))$ is the quaternion operator

$$T(q(t)) = \begin{bmatrix} q_0(t) & -q_1(t) & -q_2(t) & -q_3(t) \\ q_1(t) & q_0(t) & -q_3(t) & q_2(t) \\ q_2(t) & q_3(t) & q_0(t) & -q_1(t) \\ q_3(t) & -q_2(t) & q_1(t) & q_0(t) \end{bmatrix}.$$



Figure 7.1: ISS body reference frame (adapted from [41]).

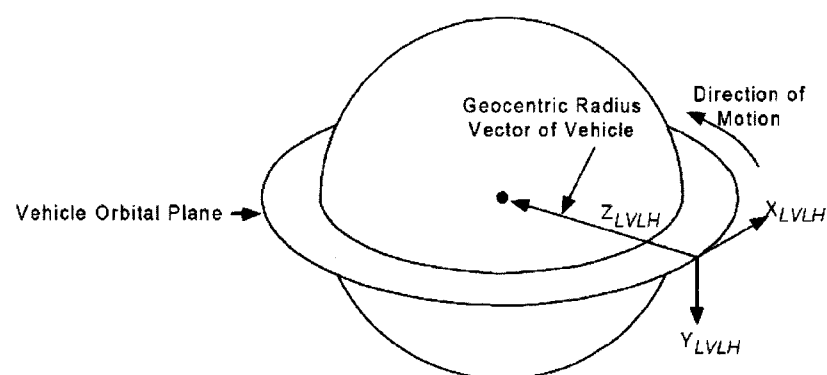


Figure 7.2: ISS LVLH reference frame [6, Section 2.1] (adapted from [38]).

7.3 ZPM Optimal Control Problem

The ZPM OCP transitions the states $\omega(t)$, $h(t)$, and $q(t)$ from predetermined initial values to a set final values without exceeding the path boundary conditions on $h(t)$ at any time $t \in [t_0, t_f]$. This OCP is now stated in a particular formulation:

$$\begin{aligned}
 & \min_{\omega(t), h(t), q(t), \tau_c(t)} \int_0^{t_f} \tau_c(t)^T \tau_c(t) dt \\
 & \text{s.t.} \\
 & \begin{bmatrix} \dot{\omega}(t) \\ \dot{h}(t) \\ \dot{q}(t) \end{bmatrix} = \begin{bmatrix} J^{-1}[-\omega(t) \times (J\omega(t) + h(t)) - \tau_c(t) + \tau_{gg}(q(t))] \\ \tau_c(t) \\ \frac{1}{2}T(q(t)) \begin{bmatrix} 0 & (\omega^T(t) - \omega_0^T(q(t))) \end{bmatrix}^T \end{bmatrix} \\
 & h(t)^T h(t) \leq h_{\max}^2 \\
 & \begin{bmatrix} \omega(t_0) \\ h(t_0) \\ q(t_0) \end{bmatrix} = \begin{bmatrix} \bar{\omega}_0 \\ \bar{h}_0 \\ \bar{q}_0 \end{bmatrix} \\
 & \begin{bmatrix} \omega(t_f) \\ h(t_f) \\ q(t_f) \end{bmatrix} = \begin{bmatrix} \bar{\omega}_f \\ \bar{h}_f \\ \bar{q}_f \end{bmatrix}
 \end{aligned}$$

7.4 ISS 90-degree Maneuver

The prototypical example for the International Space Station ZPM Maneuvers is adapted from the Master's Thesis of S. Bhatt [6, Section 4.5]. The problem was to rotate the Space Station 90-degrees from the orientation in Table 7.1 to the orientation

in Table 7.2 in 7,200 seconds. The parameters chosen for this maneuver were

$$J = \begin{bmatrix} 18,836,544 & 3,666,370 & 2,965,301 \\ 3,666,370 & 27,984,088 & -1,129,004 \\ 2,965,301 & -1,129,004 & 39,442,649 \end{bmatrix} \text{ slug-ft}^2$$

and $\omega_{orb} = 0.0657$ deg/sec.

State	Roll	Pitch	Yaw	Units
$e(t_0)$	2	-9.34	13	deg
$h(t_0)$	1,000	-500	-4,200	ft-lb-sec

Table 7.1: Initial conditions for 90-deg ZPM

State	Roll	Pitch	Yaw	Units
$e(t_f)$	-2.19	-7.88	-89.85	deg
$h(t_f)$	-9	-3,557	-135	ft-lb-sec

Table 7.2: Final conditions for 90-deg ZPM

The optimal state and control trajectories computed from the PS method with 30 collocation points and the MS method with 30 equally spaced nodes and 30 integration intervals on each subinterval are shown in figures 7.3–7.15. The optimization algorithm described in Chapter 3 was applied until the solution met the tolerances $\epsilon^{\text{opt}} = 5 \times 10^{-2}$ and $\epsilon^{\text{feas}} = 5 \times 10^{-6}$, or $\epsilon^y = 5 \times 10^{-12}$. The performance of the optimization algorithm on this example is given in the table 7.3. Notice that the

solutions produced by the LGL PS and MS methods, with each of the control parameterizations, are similar and produce objective values that differ at most by 19 percent from the best value found, which is the value for MS (linear) in table 7.3. These differences are likely due to the differences in the transcription techniques and the choice of $N = 30$. The results of Sections 4.4 and 5.5 suggest that these solutions would approach one another, and approach the true optimal solution of the continuous OCP if $N \rightarrow \infty$.

Transcription Technique	Function Evaluations	Iterations	CPU Time [sec]	Objective Value
LGL PS	145	29	427	70.9
MS (constant)	114	31	1813	74.1
MS (linear)	117	34	1916	62.3
MS (cubic)	84	24	1405	66.4

Table 7.3: Performance of the optimization algorithm described in Chapter 3 on 90-deg ZPM

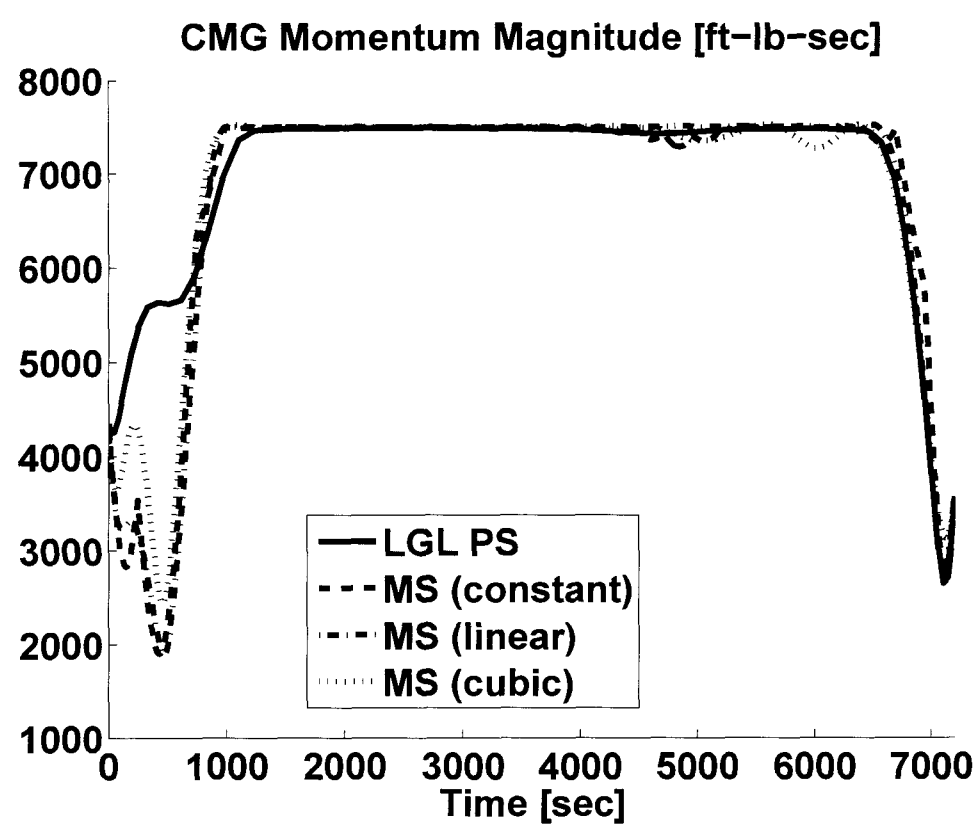


Figure 7.3: 90-deg ZPM CMG momentum magnitude

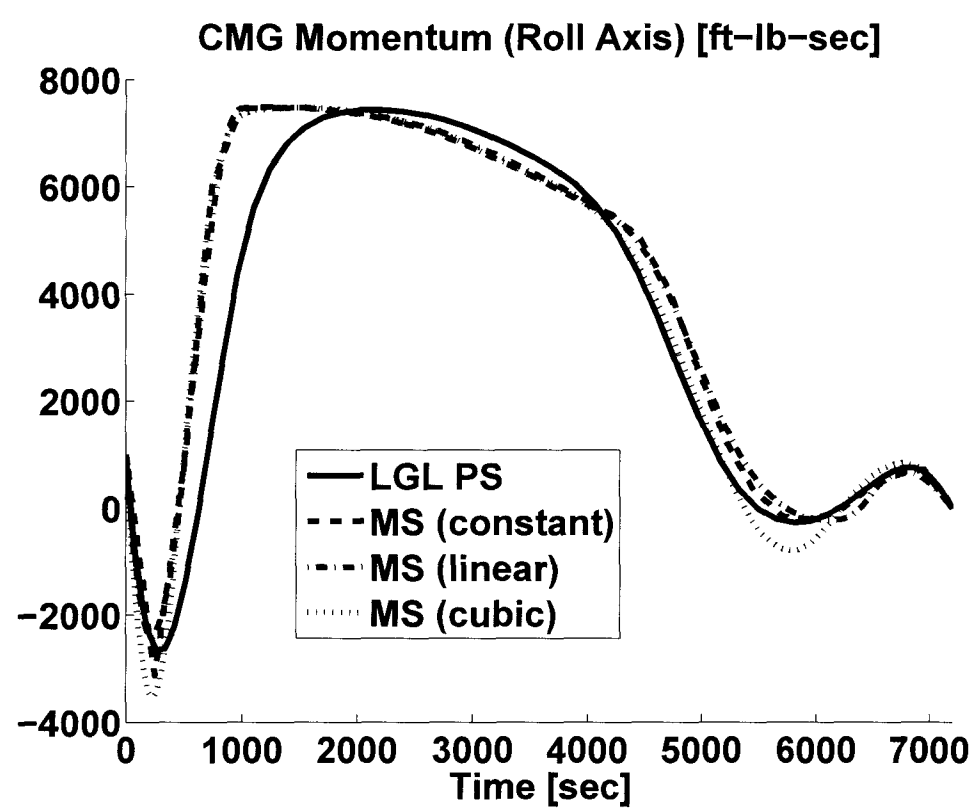


Figure 7.4: 90-deg ZPM CMG momentum (roll axis)

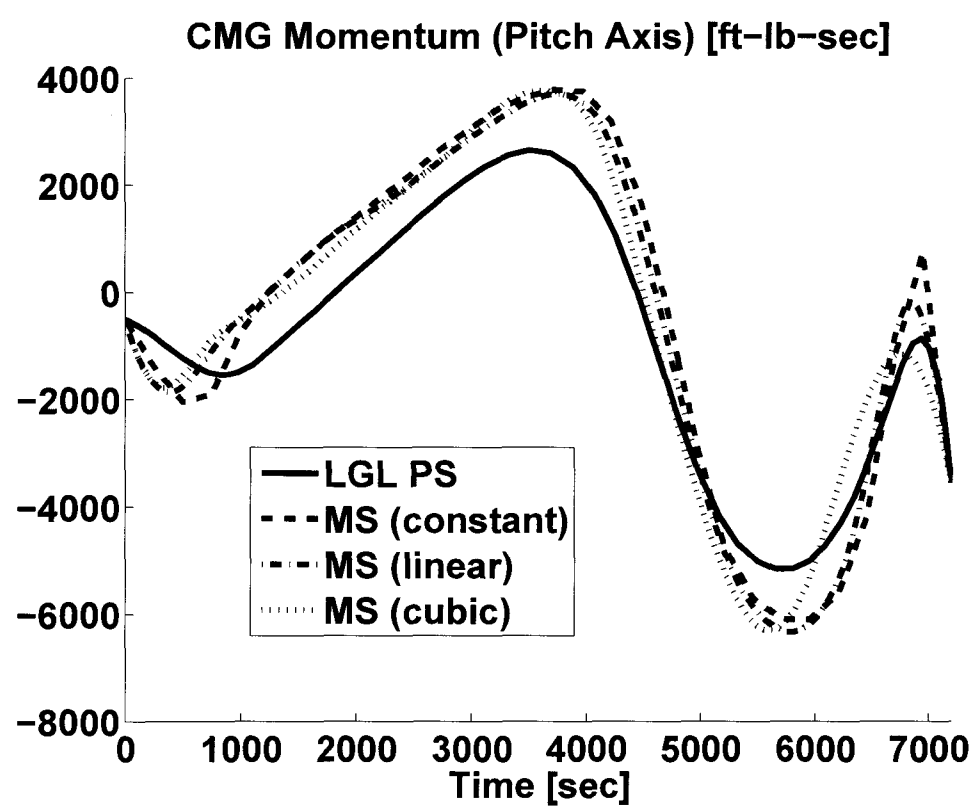


Figure 7.5: 90-deg ZPM CMG momentum (pitch axis)

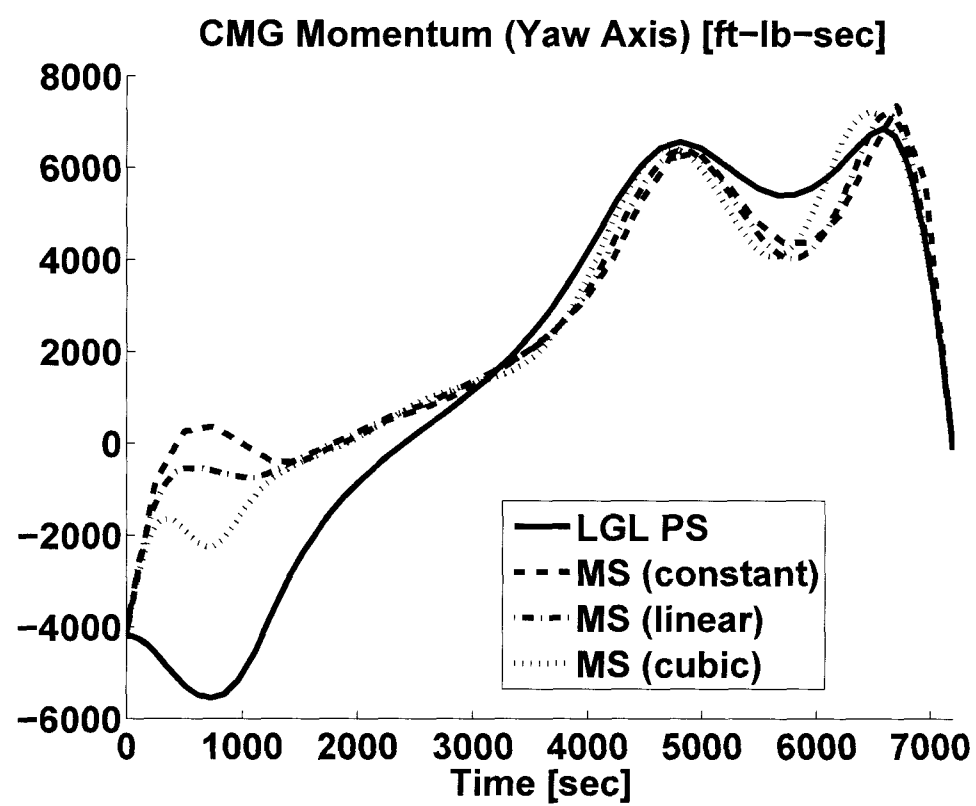


Figure 7.6: 90-deg ZPM CMG momentum (yaw axis)

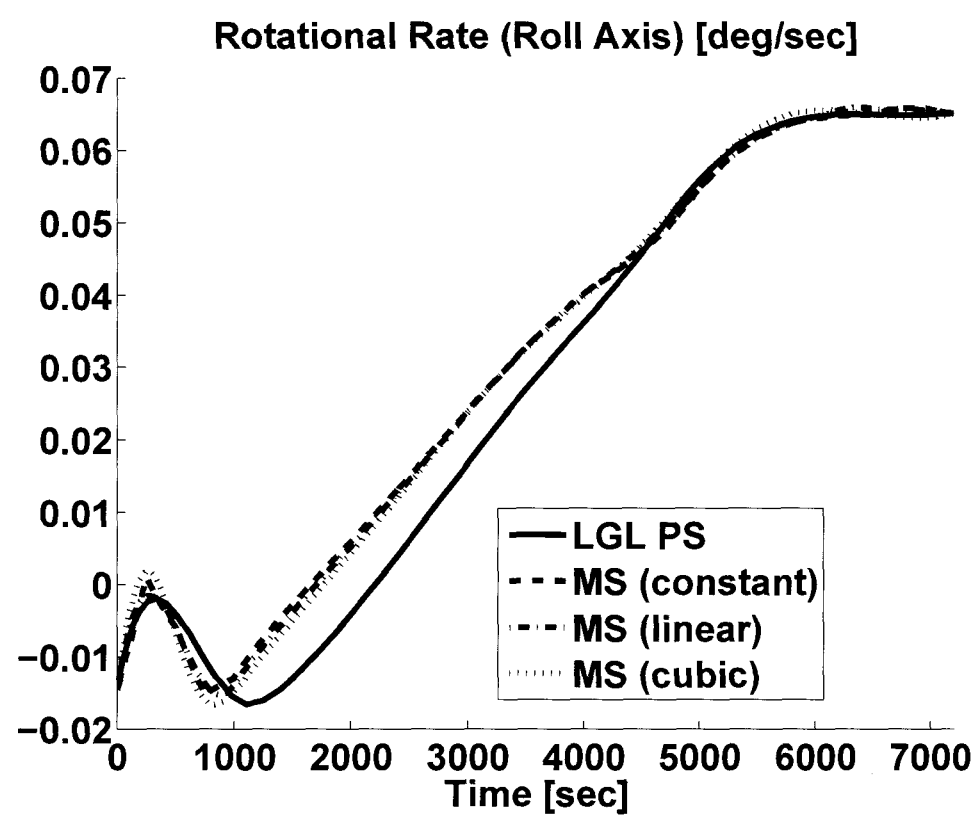


Figure 7.7: 90-deg ZPM rotational rate (roll axis)

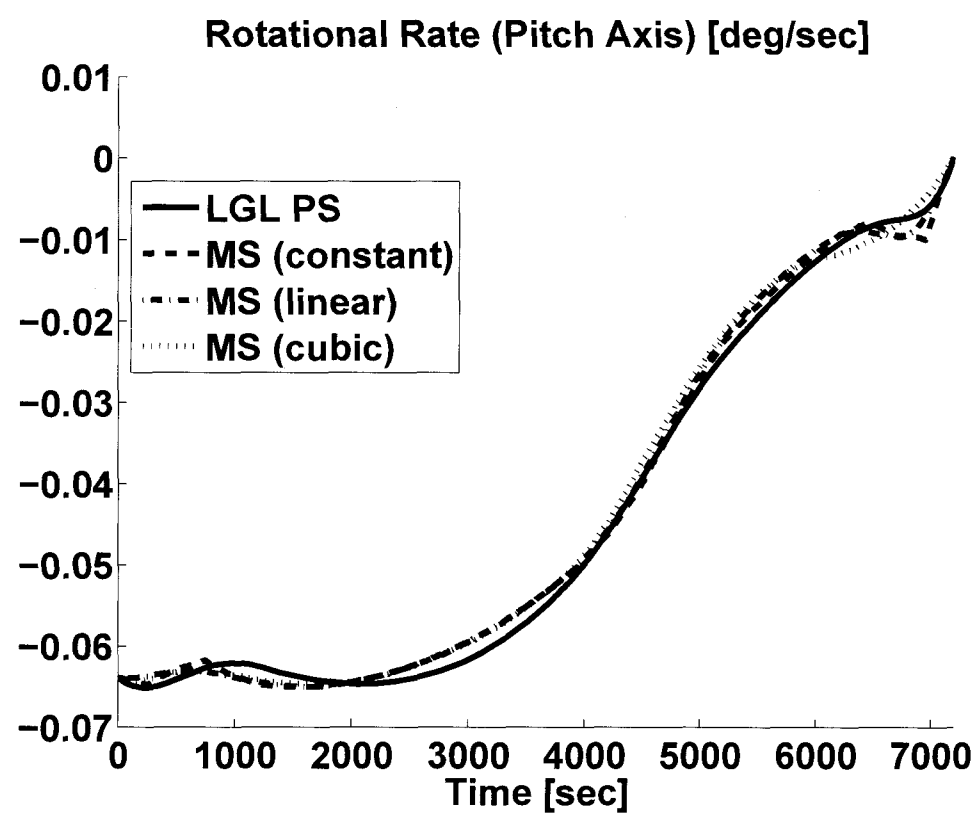


Figure 7.8: 90-deg ZPM rotational rate (pitch axis)

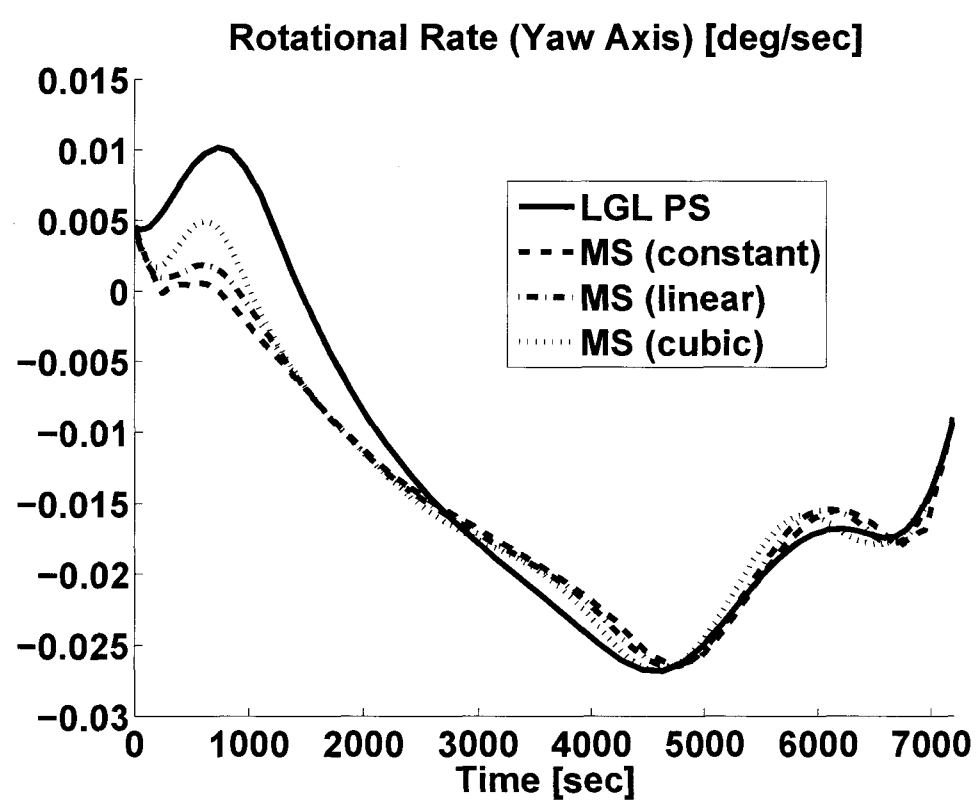


Figure 7.9: 90-deg ZPM rotational rate (yaw axis)

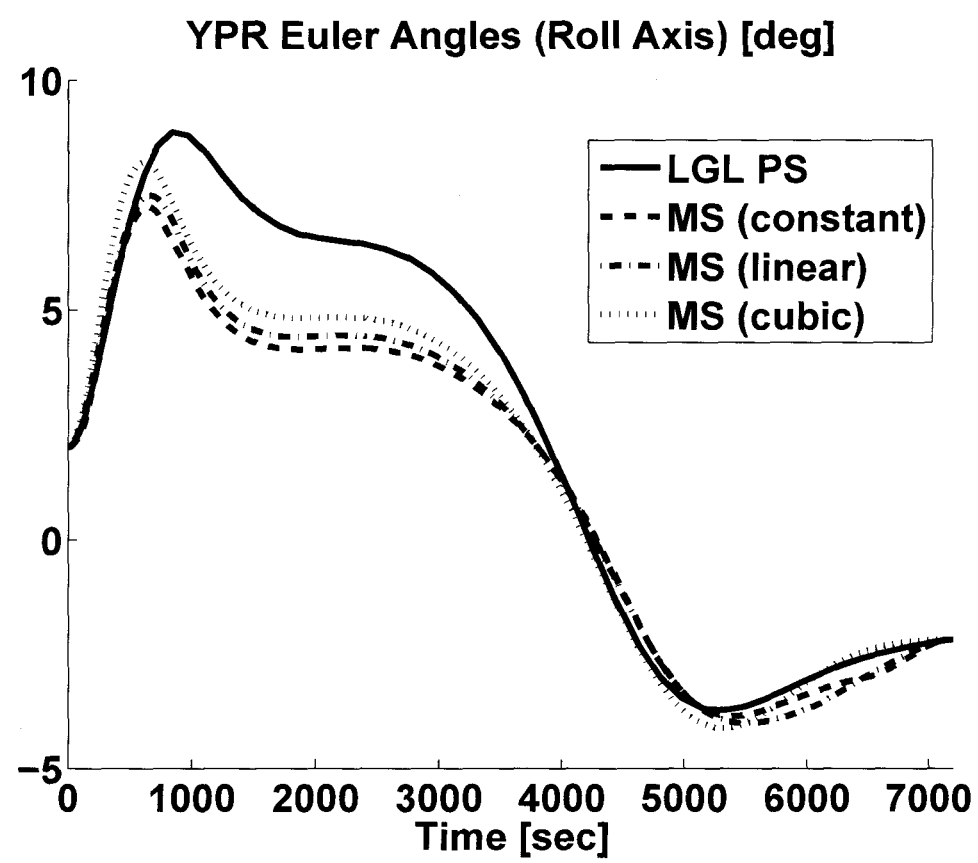


Figure 7.10: 90-deg ZPM YPR Euler angles (roll axis)

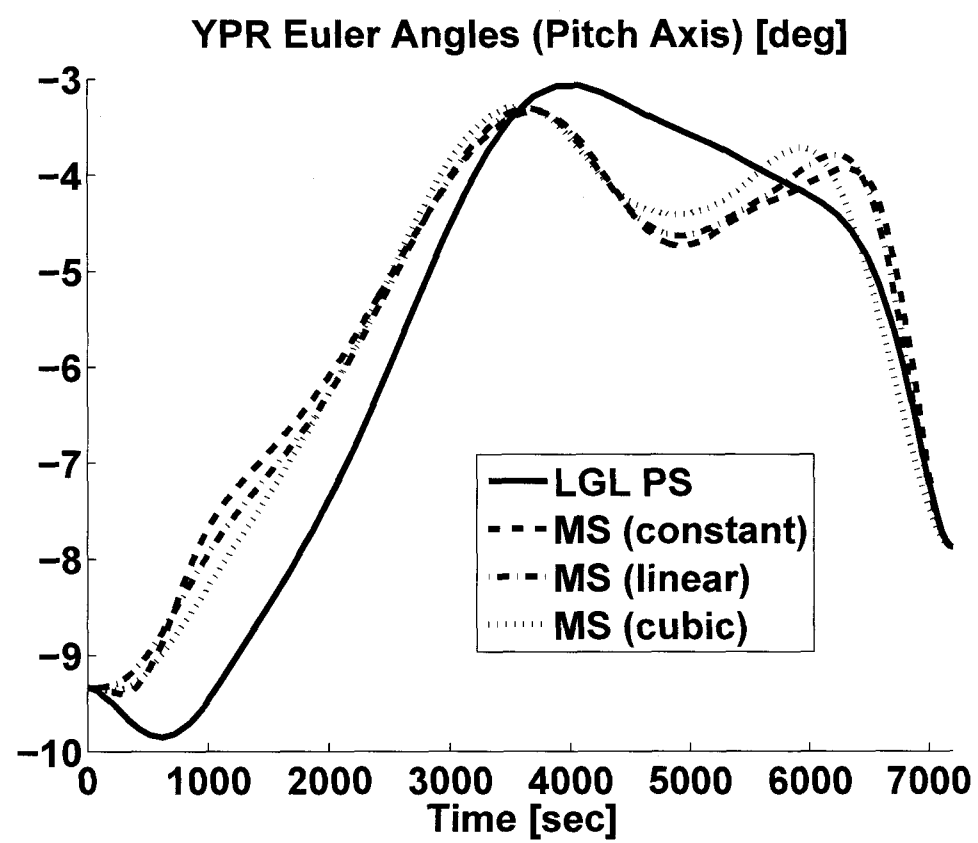


Figure 7.11: 90-deg ZPM YPR Euler angles (pitch axis)

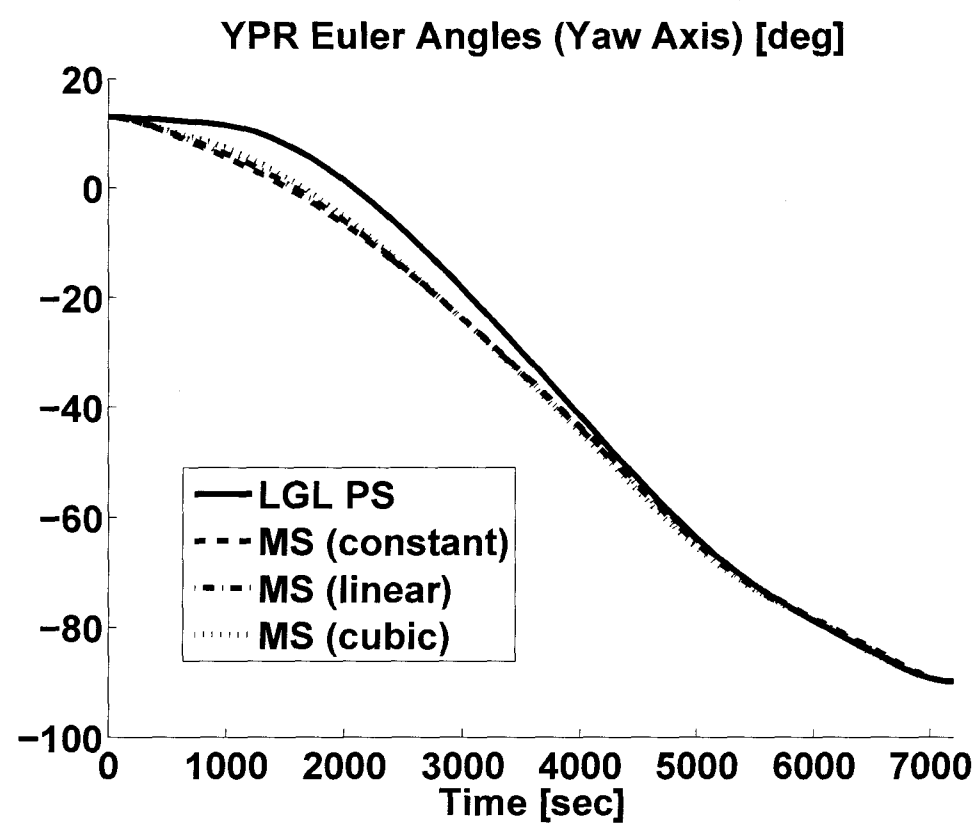


Figure 7.12: 90-deg ZPM YPR Euler angles (yaw axis)

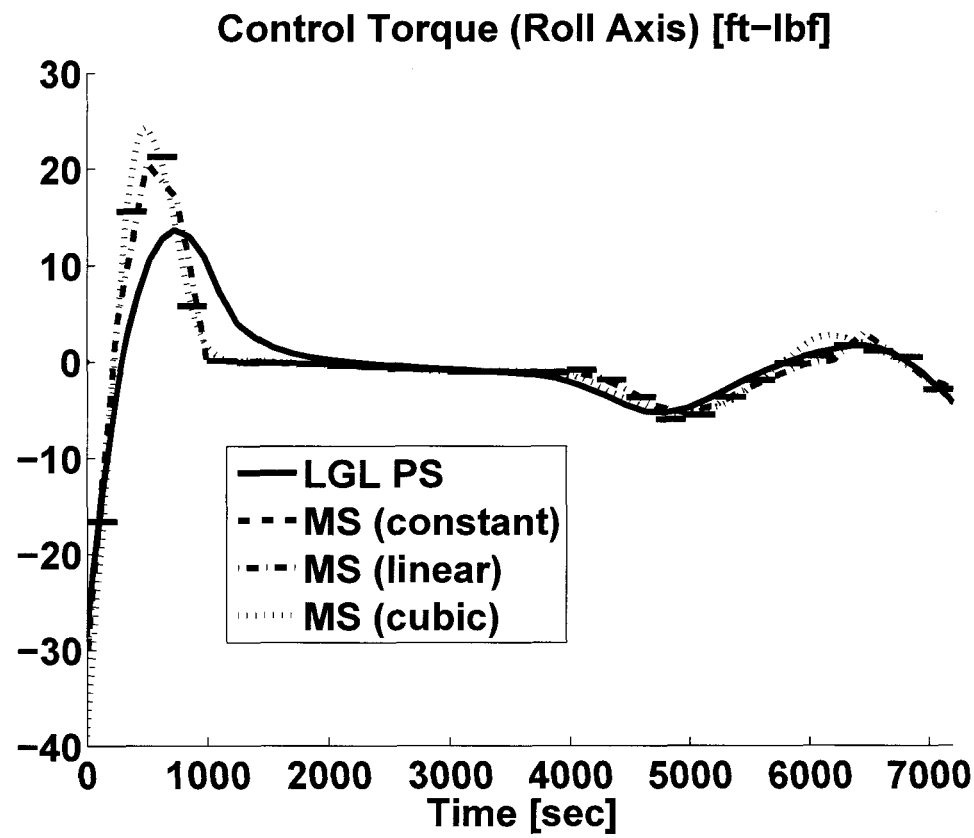


Figure 7.13: 90-deg ZPM CMG torque (roll axis)

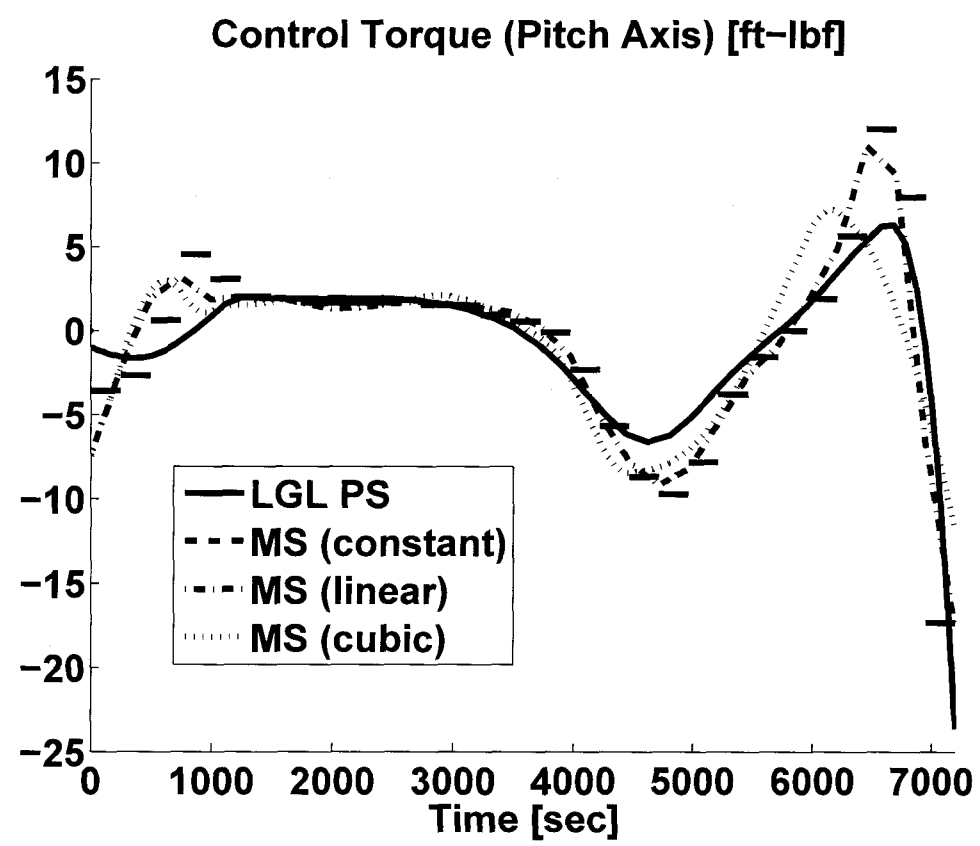


Figure 7.14: 90-deg ZPM CMG torque (pitch axis)

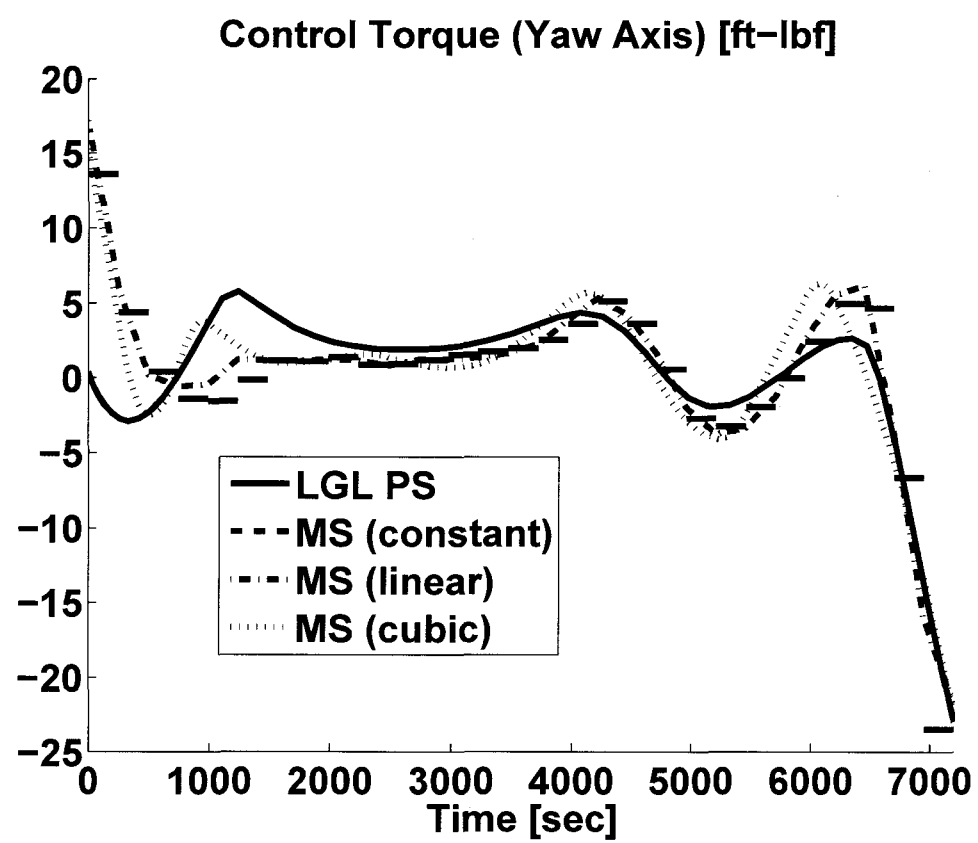


Figure 7.15: 90-deg ZPM CMG torque (yaw axis)

7.5 Numerical Comparisons

Defining fair, one-to-one comparisons between LGL PS collocation and MS transcription is not such an easy task. The optimization algorithm interfacing the objective and constraint functions defined by the two methods can have a great impact on the performance of one method or the other. For this reason, the comparison of the computation time for both methods will be on the time required to evaluate the objective and constraint functions. Essentially, the objective and constraint functions take the state and control trajectories at the current iterate and evaluate the functions and derivatives of the equations (4.2) of the LGL PS method or the equations (5.7), along with (5.16), (5.17), and (5.18), of the MS method with constant, linear, and cubic control parameterizations, respectively.

The accuracy of these methods is also a meaningful comparison, so a measure of accuracy that is fair to both of these methods was defined. In essence, this measure of accuracy compares the optimal state trajectory to a state trajectory resulting from a simulation of the dynamics subject to the optimal control. Note that optimal here refers to the optimal numerical solution computed by the algorithm of Chapter 3.

The comparisons of this section were completed using the ZPM example of the previous sections. The comparisons in this section were made on a PC running Microsoft Windows XP Professional, Version 2002, Service Pack 3, with an AMD Athlon 64 X2 Dual Core Processor 5600+ at 2.91 GHz and 2.0 GB of RAM, and under the MATLAB 7.6.0 (R2008a) environment.

7.5.1 Computation Time

A comparison of the computation time was made by evaluating the objective and constraint functions for $N = 3, \dots, 100$; evaluation began at $N = 3$ so that there was at least one node between t_0 and t_f . This was chosen as a fair comparison since the optimization algorithm spends a significant amount of time evaluating these functions. The objective and constraint functions are sometimes evaluated several times in one iteration of the optimization algorithm, and, in the optimization algorithm described in Chapter 3, these function were evaluated up to 23 times in one iteration. Figure 7.16 illustrates the time required to evaluate the constraint functions for the LGL PS and MS transcriptions.

Figure 7.16 shows that the NLP function evaluations for the LGL PS transcription are significantly faster compared to NLP function evaluations for the MS transcription. This is due to the fact that in the MS transcription method the system of ordinary differential equations defined by (5.13) must be evaluated over each of the $N - 1$ subintervals in a particular transcription. This point is fortified by comparing just the MS methods for different values of n_{int} . The computation time increases significantly as the number of integration intervals is increased, reaching nearly 2 minutes when $N = 100$ and $n_{\text{int}} = 70$. This would result in approximately 46 minutes for a single iteration of the optimization algorithm if the functions were evaluated the maximum number of times. However, this comparison needs be viewed as preliminary. Many aspects need to be included for a final computing time comparison.

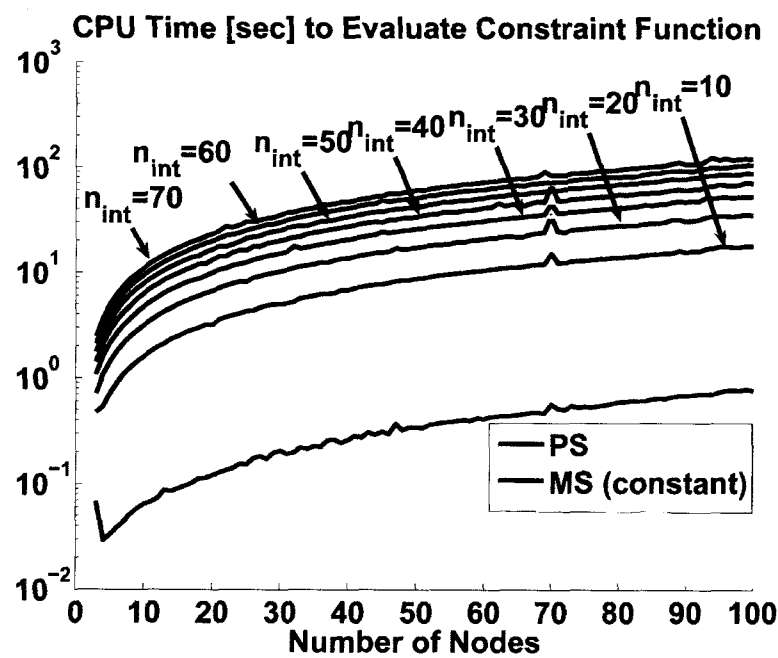


Figure 7.16: CPU time required to evaluate objective and constraint functions. The label n_{int} refers to the number of integration intervals between nodes of the MS method, using a simple, fixed-step Runge-Kutta 4 integrator.

For a better comparison of computing time requirements for single NLP function evaluations in the LGL PS transcription method and the MS transcription method several factors need to be included. The LGL PS transcription function evaluations are better vectorized than those for the MS transcription method, which is an advantage in the Matlab implementation. More importantly, the comparison was made for fixed number of points and Runge-Kutta time steps inside MS intervals. The use of an adaptive ODE solver inside the MS transcription method should be studied. For both transcription methods it would be better to study computational cost re-

quirements for desired accuracy. Ultimately, this requires adaptive discretizations for both methods. Additionally it requires an evaluation of the overall optimization. For example, considering the numerical example of Section 7.4, the number of function evaluations is significantly less for each of the control parameterizations of the MS method when compared to those required to solve the problem transcribed by the LGL PS method (see, table 7.3).

7.5.2 A Measure of Accuracy

The CPU time required to evaluate the objective and constraint functions of the MS method seems like a profoundly limiting factor in the use of the method compared to PS methods. However, if a measure of accuracy for the two methods is defined as the difference between the states of the optimal solution computed numerically and the actual states obtained by integrating the dynamics subject to the control, then it can be shown that MS methods can perform increasingly well, even with only a few transcription nodes, just by increasing the accuracy of the numerical integration technique on each subinterval, while PS methods require an increase in the number of collocation points to achieve greater accuracy.

Let $(x^*(t), u^*(t))$ be the optimal state and control trajectories obtained from the optimization algorithm described in Chapter 3 and $x(t; u^*(t))$ represent the state trajectory resulting from the integration of the dynamics subject to $u^*(t)$ in an accurate numerical integrator. Then the desired measure of accuracy is defined as the error

between $x^*(t)$ and $x(t; u^*(t))$:

$$\|x^*(t) - x(t; u^*(t))\|_2. \quad (7.2)$$

For the optimization performed in the comparison that follows, the tolerances on the optimization algorithm were chosen to be $\epsilon^{\text{opt}} = 5 \times 10^{-2}$, $\epsilon^{\text{feas}} = 5 \times 10^{-6}$, and $\epsilon^y = 5 \times 10^{-12}$. The stricter tolerance on feasibility is enforced since the satisfaction of the dynamic of the PS methods and continuity constraints of the MS method should be accurate at the transcription nodes if this measure of accuracy is to make sense. The accurate numerical integrator was chosen to be MATLAB's ode45, with "RelTol" set at 5×10^{-6} and "AbsTol" set at 5×10^{-6} to coincide with the tolerance ϵ^{feas} of the optimization. These comparisons are shown in figures 7.17, 7.18, and 7.19. For fixed N , the MS method does, indeed, perform better as the number of integration intervals, n_{int} , increases, while the only control over the accuracy available for the LGL PS method is to increase the number of nodes. This must be taken into consideration when the memory available to perform the optimization is limited since the required storage space increases with N , and it does so significantly for LGL PS collocation.

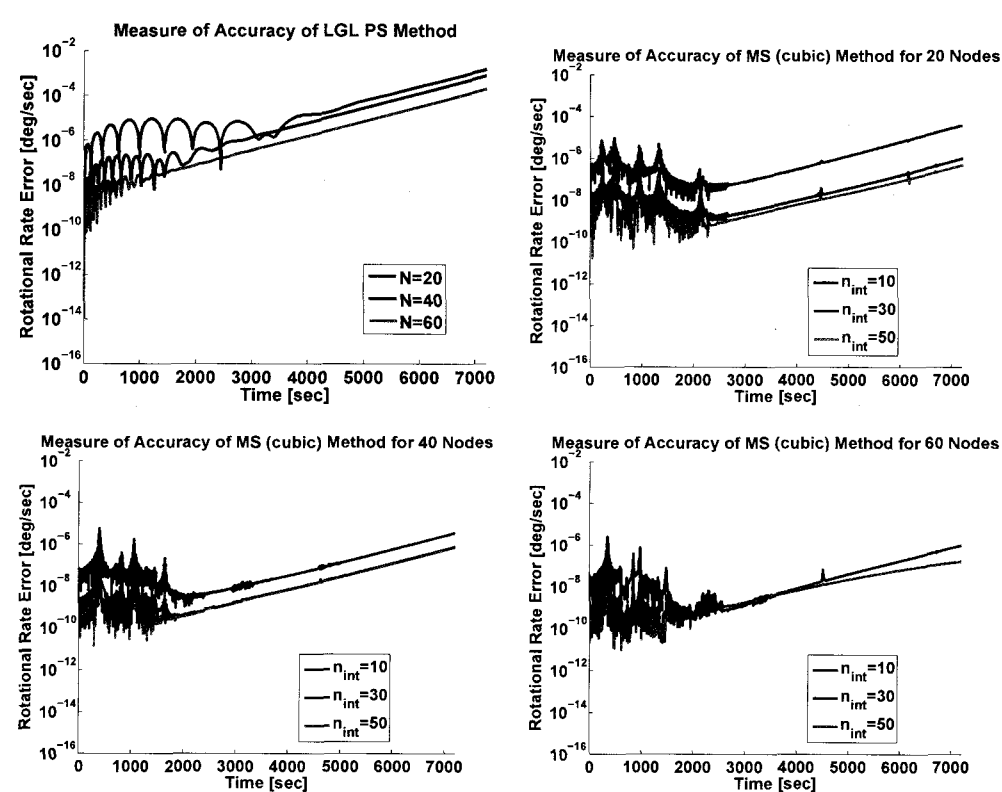


Figure 7.17: rotational rate Error in LGL PS and MS (cubic) methods. rotational rate error is computed by (7.2). The label n_{int} refers to the number of integration intervals between nodes of the MS method, using a simple, fixed-step Runge-Kutta 4 integrator.

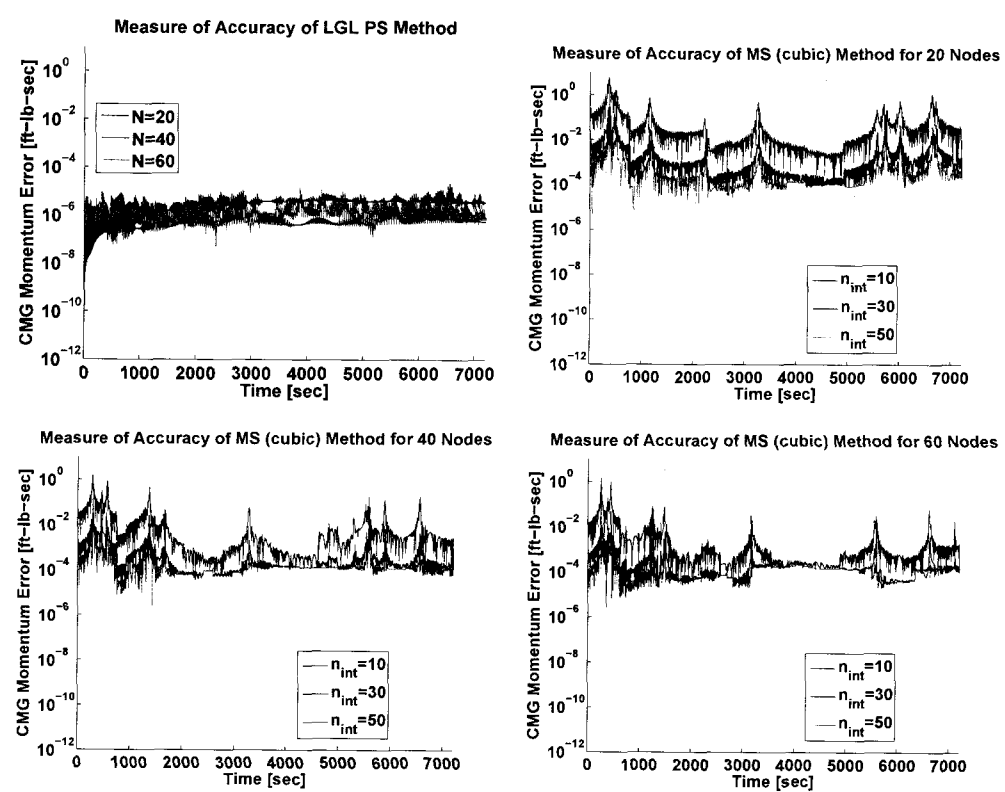


Figure 7.18: CMG momentum error in LGL PS and MS (cubic) methods. CMG momentum error is computed by (7.2). The label n_{int} refers to the number of integration intervals between nodes of the MS method, using a simple, fixed-step Runge-Kutta 4 integrator.

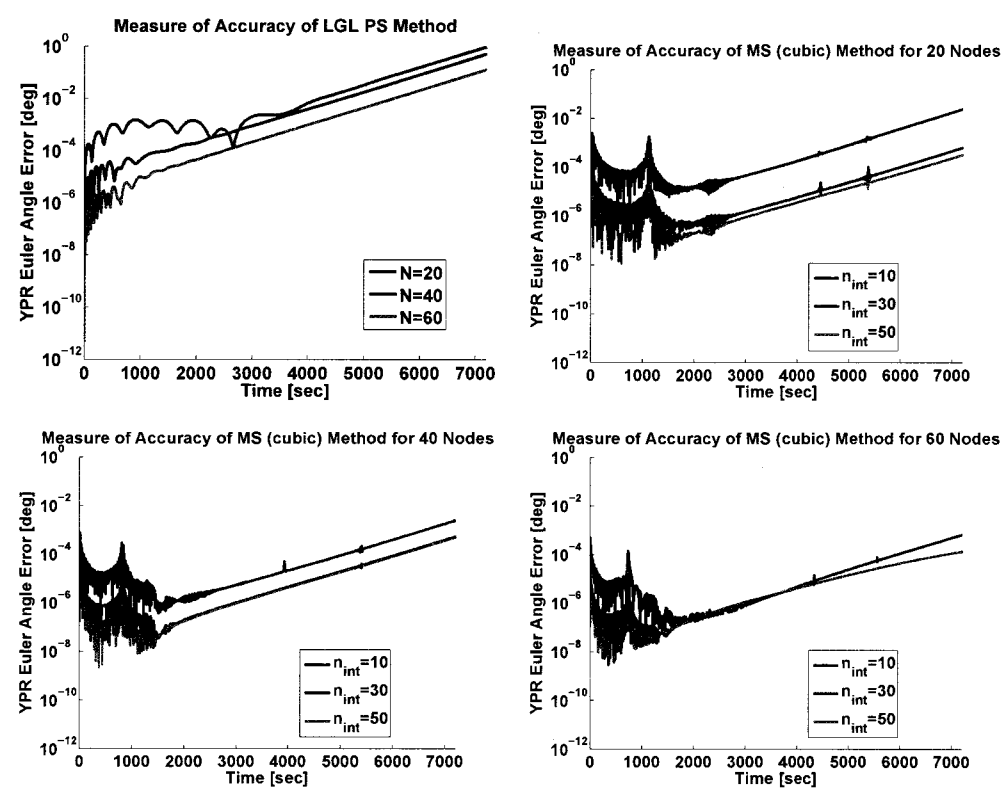


Figure 7.19: YPR Euler angle in LGL PS and MS (cubic) methods. YPR Euler angle error is computed by (7.2). The label n_{int} refers to the number of integration intervals between nodes of the MS method, using a simple, fixed-step Runge-Kutta 4 integrator.

Chapter 8

Conclusions

This thesis drew some distinct comparisons between two popular transcription techniques for optimal control problems governed by ordinary differential equations. Prior to this research, these two techniques existed separately from one another, and, often, in the detailed description of one technique the other was only mentioned as an alternative without explanation. The comparisons drawn in this thesis point to the limitations of these two methods, and suggest that the application certainly defines which should be used in particular situations. For instance, in situations where memory is limited, but accurate solutions are desired, MS methods would be the better choice. On the other hand, some situations require that solutions be found quickly, with little regard to the use of memory, and in this case, using the current implementation of these methods in MATLAB, the LGL PS method would be preferable.

Both methods suffer, in some aspect, in approximating the true optimal solution.

The absence of an existence result for LGL PS collocation is alarming, particularly when the continuous OCP has a solution but the LGL PS transcription does not. This can be overcome by the relaxation given in Section 4.4. MS methods suffer from their ability to approximate the solution to the continuous problem when the transcription is defined at only a small number of nodes. This can certainly be overcome by increasing the number of nodes. However, in the framework of the ZPM, these difficulties are not prohibitive since feasibility is far more important than optimality.

Much more can be done to strengthen these comparisons. For instance, these methods could be applied to many more OCPs to confirm the results in computation time and accuracy. Further, the optimization algorithm is hardly robust at this point, and could be tuned to handle a greater variety of nonlinear programs. One way to potentially increase this robustness is to introduce a trust region step like that described by Waltz, et al., [11, 12, 56]. Tuning this algorithm and reapplying these comparisons would also lead to even more meaningful comparisons. Ultimately, the goal is to transfer the optimization package, along with the transcription techniques into an environment other than MATLAB, which could be compiled on any system.

Bibliography

- [1] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. Classics in Applied Mathematics, Vol. 13. SIAM, Philadelphia, 1995.
- [2] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, 1998.
- [3] N. Bedrossian and S. Bhatt. Space station zero-propellant maneuver guidance trajectories compared to eigenaxis. *American Control Conference, 2008*, pages 4833–4838, June 2008.
- [4] D. Benson. *A Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February 2005.
- [5] J. T. Betts. A survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21:193–207, 1998.

- [6] S. A. Bhatt. Optimal reorientation of spacecraft using only control moment gyroscopes. Master's thesis, Department of Computational and Applied Mathematics, MS 134, Rice University, Houston, Texas, 77005-1892, 2007. Available as CAAM Technical Report TR07-08 http://www.caam.rice.edu/tech_reports/2007/TR07-08.pdf.
- [7] T. Binder, L. Blank, H. G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kroneder, W. Marquardt, J. P. Schlöder, and O. v. Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Groetschel, S. O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 295–340. Springer Verlag, Berlin, Heidelberg, New York, 2001.
- [8] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.
- [9] C. B. Boyer and U. C. Merzbach. *A History of Mathematics*. Wiley, New York, second edition, 1991.
- [10] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. Classics in Applied Mathematics, Vol. 14. SIAM, Philadelphia, 1995.

- [11] R. H. Byrd, J. C. Gilbert, and J. Nocedal. A trust-region method based on interior point techniques for nonlinear programming. *Mathematical Programming A*, 89:149–185, 2000.
- [12] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM J. Optimization*, 9:877–900, 1999.
- [13] P. Deuffhard and F. Bornemann. *Scientific computing with ordinary differential equations*, volume 42 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 2002.
- [14] F. Fahroo and I. M. Ross. Costate estimation by a Legendre pseudospectral method. *Journal of Guidance, Control and Dynamics*, 24:270–271, 2001.
- [15] F. Fahroo and I. M. Ross. User’s manual for dido 2002: A MATLAB application package for dynamic optimization. Technical Report NPS-AA-02-002, Naval Postgraduate School, Monterey, CA, June 2002.
- [16] F. Fahroo and M. Ross. A perspective on methods for trajectory optimization. AIAA/AAS Astrodynamics Conference, August 2002.
- [17] B. Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, Cambridge, New York, 1996.
- [18] C.J. Goh and K.L. Teo. Control parameterization: A unified approach to optimal control problems with general constraints. *Automatica*, 24:3–18, 1988.

- [19] L. Göllmann, D. Kern, and H. Maurer. Optimal control problems with delays in state and control variables subject to mixed control-state constraints. *Optimal Control Appl. Methods*, 2008. early view.
- [20] Q. Gong, W. Kang, N. S. Bedrossian, F. Fahroo, P. Sekhavat, and K. Bollino. Pseudospectral optimal control for military and industrial applications. In *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, Dec 2007.
- [21] Q. Gong, I. M. Ross, W. Kang, and F. Fahroo. Connections between the covector mapping theorem and convergence of pseudospectral methods for optimal control. *Comput. Optim. Appl.*, 41(3):307–335, 2008.
- [22] Qi Gong, Wei Kang, and I.M. Ross. A pseudospectral method for the optimal control of constrained feedback linearizable systems. *IEEE Transactions on Automatic Control*, 51(7):1115–1129, Jul. 2006.
- [23] W. Grimm. Convergence relations between optimal control and optimal parametric control. *Journal of Optimization Theory and Applications*, 92(2):263–283, 1997.
- [24] W. Grimm and A. Markl. Adjoint estimation from a direct multiple shooting method. *J. Optim. Theory Appl.*, 92:263–283, 1997.
- [25] W. W. Hager. Runge-Kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87:247–282, 2000.

- [26] E. Hairer, S. O. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer Series in Computational Mathematics, Vol. 8. Springer Verlag, Berlin, Heidelberg, New York, second edition, 1993.
- [27] R. Hartl, S. Sethi, and R. Vickson. A survey of the maximum principles for optimal control problems with state constraints. *SIAM Review*, 37(2):181–218, 1995.
- [28] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge Monographs on Applied and Computational Mathematics (No. 21). Cambridge University Press, Cambridge, 2007.
- [29] P. C. Hughes. *Spacecraft Attitude Dynamics*. John Wiley and Sons, New York, 1986.
- [30] S. Kameswaran and L. T. Biegler. Convergence rates for direct transcription of optimal control problems using collocation at Radau points. *Comput. Optim. Appl.*, 41(1):81–126, 2008.
- [31] W. Kang, Q. Gong, and I. M. Ross. Pseudospectral optimal control and its convergence theorems. In A. Astolfi and L. Marconi, editors, *Analysis and Design of Nonlinear Control Systems*, pages 109–124. Springer Verlag, Berlin, Heidelberg, New York, 2008.

- [32] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large scale dynamic process optimization. part I: Theoretical aspects. *Comput. Chem. Engng.*, 27:157–166, 2003.
- [33] D. B. Leineweber, H. G. Bock, J. P. Schlöder, J. V. Gallitzendörfer, A. Schäfer, and P. Jansohn. A boundary value problem approach to the optimization of chemical processes described by DAE models. Technical Report 97–14, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Universität Heidelberg, 1997.
- [34] D. B. Leineweber, H. Schäfer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large scale dynamic process optimization. part II: Software aspects and applications. *Comput. Chem. Engng.*, 27:167–174, 2003.
- [35] T. Maly and L. R. Petzold. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Applied Numerical Mathematics*, 20:57–79, 1996.
- [36] O. L. Mangasarian and S. Fromovitz. The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. *J. Math. Anal. Appl.*, 17:37–47, 1967.
- [37] E. McCants. Optimal open-loop cmg maneuvers. Master’s thesis, Department of Mechanical Engineering, Rice University, Houston, Texas, 2001.

- [38] Mission Operations Directorate Space Flight Training Division. International space station familiarization. Technical Report TD-9702A, NASA, July 1998.
- [39] D. D. Morrison, J. D. Riley, and J. F. Zancanaro. Multiple shooting method for two-point boundary value problems. *Commun. ACM*, 5(12):613–614, 1962.
- [40] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Verlag, Berlin, Heidelberg, New York, second edition, 2006.
- [41] Petty, J. I., ed. Iss assembly mission 12a. Internet Source, November 2007. Accessed 7 April 2009.
http://www.nasa.gov/mission_pages/station/structure/iss_assembly_12a.html.
- [42] J. A. Pietz. Pseudospectral collocation methods for the direct transcription of optimal control problems. Master’s thesis, Department of Computational and Applied Mathematics, MS 134, Rice University, Houston, Texas, 77005-1892, 2003. Available as CAAM Technical Report TR03-10
http://www.caam.rice.edu/tech_reports/2003/TR03-10.pdf.
- [43] E. Polak. *Optimization: Algorithms and Consistent Approximations*. Applied Mathematical Sciences, Vol. 124. Springer Verlag, Berlin, Heidelberg, New York, 1997.
- [44] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, and E.F. Mishchenko. *The Mathematical Theory of Optimal Processes*. John Wiley & Sons, New York, 1962.

- [45] G. W. Reddien. Collocation at Gauus points as a discretization in optimal control. *SIAM J. Control and Optimization*, 17(2), March 1979.
- [46] R. Roady. Real-time guidance for spacecraft reorientation using control moment gyroscopes. Master's thesis, Mechanical Engineering and Material Science, Rice University, Houston, Texas, 77005, Houston, Texas, 2008.
- [47] I. M. Ross and F. Fahroo. Legendre pseudospectral approximations of optimal control problems. In W. Kang, M. Xiao, and C. Borges, editors, *New Trends in Nonlinear Dynamics and Control, and their Applications*, Lecture Notes in Control and Information Sciences, Vol.295, pages 327–342, Heidelberg, 2003. Springer-Verlag.
- [48] I. M. Ross and F. Fahroo. Pseudospectral knotting methods for solving optimal control problems. *Journal of Guidance, Control and Dynamics*, 27:397–405, 2004.
- [49] M. C. Steinbach. A structured interior point SQP method for nonlinear optimal control problems. In R. Bulirsch and D. Kraft, editors, *Computational optimal control (Munich, 1992)*, volume 115 of *Internat. Ser. Numer. Math.*, pages 213–222. Birkhäuser, Basel, 1994.
- [50] M. C. Steinbach. Tree-sparse convex programs. *Math. Methods Oper. Res.*, 56(3):347–376, 2002.
- [51] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer Verlag, New York, Berlin, Heidelberg, London, Paris, second edition, 1993.

- [52] O. Von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37:357–373, 1992.
- [53] K.L. Teo, C.J. Goh, and K.H. Wang. A unified computational approach to the optimal control problems. *Pitman Monographs and Surveys in Pure and Applied Mathematics*, 55, 1991. limited.
- [54] O. von Stryk. Numerical solution of optimal control problems by direct collocation. *Optimal Control Theory and Numerical Methods*, 111:129–143, 1993.
- [55] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1, Ser. A):25–57, 2006.
- [56] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Math. Program.*, 107(3, Ser. A):391–408, 2006.
- [57] B. Wie. *Space Vehicle Dynamics and Control*. AIAA Education Series, Ohio, 1998.

Appendix A

Matlab Code Organization

The code developed for this thesis research has been written and structured for use only in MATLAB; this structure is shown in Figure A.1. One main directory titled *matlab* holds four main subdirectories *optimalgs*, *MSbolza*, *MSmayer*, and *PS*. The directories *MSbolza*, *MSmayer* and *PS* contain the files that handle the transcription of the continuous optimal control problem to the NLPs of (4.2) and (5.7). *MSbolza* and *MSmayer* hold essentially the same MATLAB m-files, but for the Bolza and Mayer formulations of the optimal control problem (2.1), respectively.

The following sections will describe the inputs to each file in these directories in the order they are called to solve an optimal control problem.

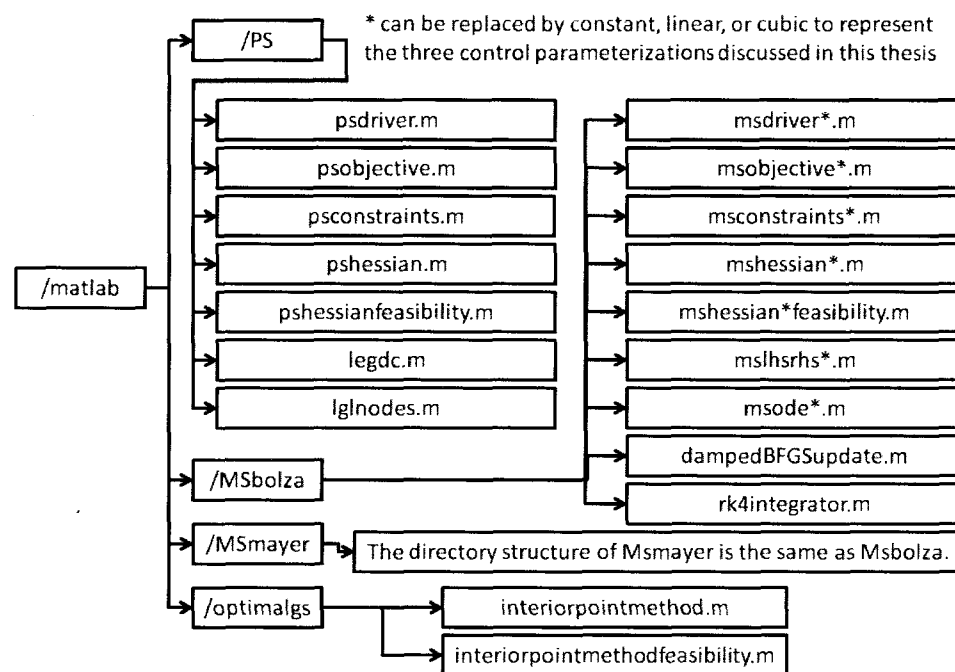


Figure A.1: Directory structure for the code developed for this thesis

A.1 User Defined Settings and Problem Specific Functions

Use of this code on a particular optimal control problem of the form (2.1) requires the definition a global structure, problem, defined by

- `problem.finalcostfunction` - handle to MATLAB function that computes and returns $l(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_x l(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_u l(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_{xx}^2 l(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_{xu}^2 l(\mathbf{s}_i, \mathbf{q}_i, t_i)$, and $\nabla_{uu}^2 l(\mathbf{s}_i, \mathbf{q}_i, t_i)$.
- `problem.costfunction` - handle to MATLAB function that computes and re-

turns $e(\mathbf{s}_N, t_N)$, $\nabla_x e(\mathbf{s}_N, t_N)$, and $\nabla_{xx}^2 e(\mathbf{s}_N, t_N)$.

- **problem.initeventfunctions** - handle to MATLAB function that computes and returns $a(\mathbf{s}_0, \mathbf{q}_0, t_0)$, $\nabla_x a(\mathbf{s}_0, \mathbf{q}_0, t_0)$, $\nabla_u a(\mathbf{s}_0, \mathbf{q}_0, t_0)$, $\nabla_{xx}^2 a(\mathbf{s}_0, \mathbf{q}_0, t_0)$, $\nabla_{xu}^2 a(\mathbf{s}_0, \mathbf{q}_0, t_0)$, and $\nabla_{uu}^2 a(\mathbf{s}_0, \mathbf{q}_0, t_0)$.
- **problem.finaleventfunctions** - handle to MATLAB function that computes and returns $b(\mathbf{s}_N, \mathbf{q}_N, t_N)$, $\nabla_x b(\mathbf{s}_N, \mathbf{q}_N, t_N)$, $\nabla_u b(\mathbf{s}_N, \mathbf{q}_N, t_N)$, $\nabla_{xx}^2 b(\mathbf{s}_N, \mathbf{q}_N, t_N)$, $\nabla_{xu}^2 b(\mathbf{s}_N, \mathbf{q}_N, t_N)$, and $\nabla_{uu}^2 b(\mathbf{s}_N, \mathbf{q}_N, t_N)$.
- **problem.odefunctions** - handle to MATLAB function that computes and returns $f(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_x f(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_u f(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_{xx}^2 f(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_{xu}^2 f(\mathbf{s}_i, \mathbf{q}_i, t_i)$, and $\nabla_{uu}^2 f(\mathbf{s}_i, \mathbf{q}_i, t_i)$.
- **problem.pathfunctions** - handle to MATLAB function that computes and returns $g(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_x g(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_u g(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_{xx}^2 g(\mathbf{s}_i, \mathbf{q}_i, t_i)$, $\nabla_{xu}^2 g(\mathbf{s}_i, \mathbf{q}_i, t_i)$, and $\nabla_{uu}^2 g(\mathbf{s}_i, \mathbf{q}_i, t_i)$.
- **problem.n** - number of states in the optimal control problem
- **problem.m** - number of controls in the optimal control problem
- **problem.N** - number of transcription nodes
- **problem.ng** - number of path constraint equations
- **problem.na** - number of initial time constraints

- `problem.nl` - flag indicating that there is an integral component to the cost (=1 if true, =0 if false).
- `problem.nb` - number of final time constraints
- `problem.tinitial` - starting time for the optimal control problem
- `problem.tfinal` - final time for the optimal control problem
- `problem.nint` - desired number of integration intervals on each of the (problem.N-1) subintervals of the MS method
- `problem.plotfunction` - handle to MATLAB function that plots state and control trajectories at all outer loop iterations of Direct Multiple Shooting. This function takes inputs $x = \{s_0, \dots, s_N\}$, $u = \{q_0, \dots, q_N\}$, and $t = \{t_0, \dots, t_N\}$.
If this functionality is not being used, the user must set `problem.plotfunction=''`.

In all cases the vector functions f , g , a , and b are column vectors, and for a given function $h(x) \in \mathbb{R}^m$, where $x \in \mathbb{R}^n$, the Jacobian matrix $h_x \in \mathbb{R}^{m \times n}$ is given by

$$h_x = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \dots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \dots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \frac{\partial h_m}{\partial x_2} & \dots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}.$$

Further, certain parameters can be defined for use in the optimization algorithm. The structure options contains the following:

- `options.iprint` - if equal to 1, relevant information at each major iteration of the interior point algorithm is printed to the command window; default = 0.
- `options.plotflag` - if equal to 1, the function pointed to by `problem.plotfunction` is called at each iteration of the algorithm; default = 0
- `options.maxit` - maximum number of major iterations; default = 1000.
- `options.opttol` - first order optimality stopping tolerance ϵ^{opt} ; default = 1.e-6
- `options.feastol` - tolerance on satisfaction of the constraints ϵ^{feas} ; default = 1.e-6
- `options.xtol` - stepsize stopping tolerance ϵ^{v} ; default = 1.e-7.

The following two options are predefined relative to the MS and PS transcriptions in this code, but can be modified when solving other nonlinear programs with the optimization algorithm:

- `options.hessian` - option to set the Hessian to either a BFGS (set by the string 'BFGS') quasi-Newton update or a user-supplied Hessian (set by the string 'user-supplied'); default = 'BFGS'
- `options.hessianfunction` - handle to function that computes the Hessian of the Lagrangian (defined in Section 3.2) with respect to the state and control variables.

An initial guess at the state and control trajectory must also be given to the optimization algorithm. This is done by defining the structure guess that follows:

- `guess.x` - guess at the state trajectory $x(t)$. This must be a `problem.n` by `problem.N` array where the trajectory of state i occupies row i of the guess. Must also be specified for at least t_0 and t_f .
- `guess.u` - guess at the control trajectory $u(t)$. This must be a `problem.m` by `problem.N` array where the trajectory of control i occupies row i of the guess. Must also be specified for at least t_0 and t_f .
- `guess.t` - node locations (time) of the elements of `guess.x` and `guess.u`.

A.2 Driver Files

- `[primal,dual]=msdriverconstant(guess,options)`
- `[primal,dual]=msdriverlinear(guess,options)`
- `[primal,dual]=msdrivercubic(guess,options)`
- `[primal,dual]=psdrivercubic(guess,options)`

Inputs:

- `guess` - the structure guess described in the previous section
- `options` - the structure options described in the previous section

Outputs:

- `primal` - structure containing `primal.states`, `primal.controls`, and `primal.nodes`.

These are arrays containing the optimal state and control trajectories at the locations defined by `primal.nodes`. The structure also contains `primal.cputime`—the runtime (seconds) of the multiple shooting algorithm, `primal.fcount`—the number of calls to the objective and constraint functions in the optimization algorithm, `primal.iterations`—the number of iterations required to complete the optimization algorithm, and `primal.objval`—the value of the objective function at the completion of the algorithm.

- `dual` - structure containing the corresponding dual variables of the optimization algorithm.

These driver files format the supplied guess so that it is useful to the interior point code, call the interior point code, and format the output of the interior point code into the primal and dual structures.

A.3 Nonlinear Interior Point Method Function

```
[x,lambda,muu,objval,out]=interiorpointmethod(...
    nlobjectivefunction,nlpconstraintfunction,x,options)
```

Inputs:

- `nlpobjectivefunction` - string containing the handle to a function that takes input `x` and outputs the scalar function `J` evaluated at `x` and the gradient with respect to `x` of `J` evaluated at `x`.
- `nlpconstraintfunction` - string containing the handle to a function that takes input `x` and outputs the vector functions `F` and `G` evaluated at `x` and the Jacobians with respect to `x` of `F` and `G` evaluated at `x`.
- `x` - initial guess or starting vector for the optimization algorithm
- `options` - structure defined as before

Outputs:

- `x` - optimal value for the vector `x`
- `lambda` - equality constraint multipliers at optimal `x`
- `muu` - inequality constraint multipliers at optimal `x`
- `objval` - `J` evaluated at optimal `x`
- `out` - structure containing the number of iterations taken to complete the optimization (`out.iterations`) and the number of calls to the objective and constraint functions (`out.fcount`).

`interiorpointmethod.m` solves the nonlinear program defined by (2.5).

A.4 NLP Objective Functions

- `[J,DJ]=msobjectiveconstant(x)`
- `[J,DJ]=msobjectivelinear(x)`
- `[J,DJ]=msobjectivecubic(x)`
- `[J,DJ]=psobjective(x)`

Inputs:

- `x` - input trajectories for states and controls of the optimal control problem. `x` is a vector in the form

$$\begin{bmatrix} \mathbf{s}_0 & \mathbf{q}_0 & \mathbf{s}_1 & \mathbf{q}_1 & \cdots & \mathbf{s}_N & \mathbf{q}_N \end{bmatrix}^T$$

where $N = \text{problem.N}$ and \mathbf{s}_i and \mathbf{q}_i for $i = 0, \dots, N$ are the state and control parameters defined in the context of this thesis.

Outputs:

- `J` - `J` evaluated at `x` from the function `problem.costfunction`
- `DJ` - Jacobian of `J` with respect to `x` evaluated at `x` from the function `problem.costfunction`

These objective functions convert the objective of the optimal control problem (2.1) to the objectives of the nonlinear programs (5.7) and (4.2) using the transcription techniques described in this thesis.

A.5 NLP Constraint Functions

- `[G,F,DG,DF]=msconstraintsconstant(x)`
- `[G,F,DG,DF]=msconstraintslinear(x)`
- `[G,F,DG,DF]=msconstraintscubic(x)`
- `[G,F,DG,DF]=psconstraints(x)`

Inputs:

- `x` - input trajectories for states and controls of the optimal control problem. `x` is a vector in the form

$$\begin{bmatrix} \mathbf{s}_0 & \mathbf{q}_0 & \mathbf{s}_1 & \mathbf{q}_1 & \cdots & \mathbf{s}_N & \mathbf{q}_N \end{bmatrix}^T$$

where $N = \text{problem.N}$ and \mathbf{s}_i and \mathbf{q}_i for $i = 0, \dots, N$ are the state and control parameters defined in the context of this thesis.

Outputs:

- `F` - `F` evaluated at `x` from the function `problem.odefunctions`
- `G` - `G` evaluated at `x` from the function `problem.pathfunctions`
- `DF` - Jacobian of `F` with respect to `x` evaluated at `x` from the function `problem.odefunctions`
- `DG` - Jacobian of `G` with respect to `x` evaluated at `x` from the function `problem.pathfunctions`

These constraint functions convert the constraints of the optimal control problem (2.1) to the constraints of the nonlinear programs (5.7) and (4.2) using the transcription techniques described in this thesis.

A.6 Fixed-Step Runge-Kutta 4 Integrator

```
[t,x]=rk4integrator(func,tspan,x0,N,varargin)
```

Inputs:

- **func** - handle to the function containing the ODE's to be numerically integrated
- **tspan** - 1 by 2 vector containing the lower and upper bounds for the integration interval
- **x0** - column vector initial conditions for the ODE's
- **N** - number of integration intervals
- **varargin** - arguments to pass to the function defined by the handle func

Outputs:

- **t** - node locations where the ODE's were integrated
- **x** - solution to the differential equations resulting from the numerical integration

`rk4integrator.m` numerically integrates a set of ODE's using Runge-Kutta-4 numerical integration.

A.7 ODE Functions for the MS Methods

- `[odestatedots]=msodeconstant(time,odestates,u,ti,problem)`
- `[odestatedots]=msodelinear(time,odestates,qi,qidot,ti,problem)`
- `[odestatedots]=msodecubic(time,odestates,apram,bparam,...
cparam,dparam,problem)`

Inputs:

- `time` - time at the current iteration of the numerical integration
- `states` - x , S , and Q at the current iteration of the numerical integration
- `u`, `qi`, `qidot`, `aparam`, `bparam`, `cparam`, `dparam` - control parameters at the beginning of the interval `tspan` defined in the function `rk4integrator.m` for the respective control parameterization
- `ti` - lower bound of the interval `tspan` defined in the function `rk4integrator.m`
- `problem` - structure defined in the first section of this appendix

Outputs:

- `odestatedots` - \dot{x} , \dot{S} , \dot{Q} computed from the ODE's defined in this thesis given the value of states.

These ODE functions compute \dot{x} , \dot{S} , \dot{Q} for the ODEs defining the continuity conditions and objective of the MS methods of this thesis.

A.8 NLP Hessian Functions

- `[del2L]=mshessianconstant(x,lambda,muu)`
- `[del2L]=mshessianlinear(x,lambda,muu)`
- `[del2L]=mshessiancubic(x,lambda,muu)`
- `[del2L]=pshessian(x,lambda,muu)`

Inputs:

- `x` - trajectories for states and controls of the optimal control problem. `x` is a vector in the form

$$\begin{bmatrix} s_0 & q_0 & s_1 & q_1 & \cdots & s_N & q_N \end{bmatrix}^T$$

where $N = \text{problem.N}$ and s_i and q_i for $i = 0, \dots, N$ are the state and control parameters defined in the context of this thesis.

- `lambda` - equality constraint multipliers at the current iteration of the optimization algorithm
- `muu` - inequality constraint multipliers at the current iteration of the optimization algorithm

Outputs:

- `del2L` - Hessian of the Lagrangian evaluated at the current iteration of the optimization algorithm

These Hessian functions compute the Hessians of the Lagrangians, with respect to the state and control variables, of the nonlinear programs (5.7) and (4.2) relative the transcription techniques described in this thesis.

A.9 LHS/RHS functions for the MS Methods

- `[rhs, lhs]=mslhsrhsconstant(lhs, rhs)`
- `[rhs, lhs]=mslhsrhslinear(lhs, rhs)`
- `[rhs, lhs]=mslhsrhscubic(lhs, rhs)`

Inputs:

- `rhs` - right hand side of the KKT system evaluated at the current iteration
- `lhs` - left hand side of the KKT system evaluated at the current iteration

Outputs:

- `rhs` - reordered right hand side of the KKT system evaluated at the current iteration
- `lhs` - reordered left hand side of the KKT system evaluated at the current iteration

These functions take the right- and left-hand sides of the KKT system and reorder them to take advantage of the structure resulting from the MS transcriptions.

A.10 Quasi-Newton Update to the Hessian

```
[Bkplus1]=dampedBFGSupdate(Bk,xkplus1,xk,gradkplus1,gradk)
```

Inputs:

- \mathbf{B}_k - approximation of the Hessian at the current iteration
- \mathbf{x}_{k+1} - value of \mathbf{x} at the next iteration
- \mathbf{x}_k - value of \mathbf{x} at the current iteration
- \mathbf{grad}_{k+1} - gradient evaluated at \mathbf{x}_{k+1}
- \mathbf{grad}_k - gradient evaluated at \mathbf{x}_k

Outputs:

- Bkplus1 - approximation of the Hessian at the next iteration

`dampedBFGSupdate.m` computes the damped BFGS approximation to the Hessian of a scalar function given the gradient of this function evaluated at the current and next iteration as well as the variables at the current and next iteration and a previous approximation to the Hessian.

A.11 Feasibility Restoration

```
[x,out]=interiorpointmethod(nlpconstraintfunction,x,muu,...
                             barrier,stoptest,options)
```

Inputs:

- `nlpconstraintfunction` - string containing the handle to a function that takes input `x` and outputs the vector functions `F` and `G` evaluated at `x` and the Jacobians with respect to `x` of `F` and `G` evaluated at `x`.
- `x` - initial guess or starting vector for the optimization algorithm
- `lambda` - current equality constraint multipliers of the regular interior point method
- `muu` - current inequality constraint multipliers of the regular interior point method
- `barrier` - current barrier parameter of the regular interior point method
- `stoptest` - constraint violation of the regular interior point method
- `options` - structure defined as before

Outputs:

- `x` - value for the vector `x` that satisfies the stopping criteria of the feasibility restoration, or `stoptest`
- `out` - structure containing the number of iterations taken to complete the optimization (`out.iterations`) and the number of calls to the objective and constraint functions (`out.fcount`).

`interiorpointmethodfeasibility.m` solves the nonlinear program defined by (3.11).

A.12 NLP Feasibility Hessian Functions

- `[del2Lfeas]=mshessianconstantfeasibility(x,lambda,muu,Dref,zeta)`
- `[del2Lfeas]=mshessianlinearfeasibility(x,lambda,muu,Dref,zeta)`
- `[del2Lfeas]=mshessiancubicfeasibility(x,lambda,muu,Dref,zeta)`
- `[del2Lfeas]=pshessianfeasibility(x,lambda,muu,Dref,zeta)`

Inputs:

- `x` - trajectories for states and controls of the optimal control problem. `x` is a vector in the form

$$\begin{bmatrix} s_0 & q_0 & s_1 & q_1 & \cdots & s_N & q_N \end{bmatrix}^T$$

where $N = \text{problem.N}$ and s_i and q_i for $i = 0, \dots, N$ are the state and control parameters defined in the context of this thesis.

- `lambda` - equality constraint multipliers at the current iteration of the optimization algorithm
- `muu` - inequality constraint multipliers at the current iteration of the optimization algorithm

- **Dref** - the matrix defined by (3.12)
- **zeta** - the weighting parameter of the objective defined in Section 3.4.

Outputs:

- **del2Lfeas**-Hessian of the Lagrangian of the feasibility nonlinear program evaluated at the current iteration of the optimization algorithm

These Hessian functions compute the Hessians of the Lagrangians, with respect to the state and control variables, of the nonlinear program (3.11) relative the transcription techniques described in this thesis.

A.13 Miscellaneous Functions for LGL PS Method

- **[t,w]=lglnodes(N,tf)**
- **[D]=legdc(N)**

Inputs:

- **N** - number of collocation points
- **tf** - t_f relative to the optimal control problem (2.1)

Outputs:

- **t** - node locations for the LGL PS Method.

- \mathbf{w} - weights for quadrature with respect to the LGL PS Method.
- \mathbf{D} - the differentiation matrix \mathbf{D} of the LGL PS Method.

These functions compute the node locations, quadrature weights, and differentiation matrix for the LGL PS method.