



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**WELL-CONDITIONED PSEUDOSPECTRAL OPTIMAL
CONTROL METHODS AND THEIR APPLICATIONS**

by

Nicholas R. Koeppen

June 2018

Thesis Advisor:
Co-Advisors:

I. Michael Ross
Ronald J. Proulx
Lucas C. Wilcox

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE June 2018	3. REPORT TYPE AND DATES COVERED Master's thesis		
4. TITLE AND SUBTITLE WELL-CONDITIONED PSEUDOSPECTRAL OPTIMAL CONTROL METHODS AND THEIR APPLICATIONS			5. FUNDING NUMBERS	
6. AUTHOR(S) Nicholas R. Koeppen				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Pseudospectral optimal control is an established discipline with flight-proven results. Aerospace applications have included the implementation of minimum-time and zero-propellant maneuvers on high-value space assets. Standard pseudospectral methods have been sufficient for these and other applications that do not require more than approximately 250 nodes. Currently, pseudospectral optimal control uses the Lagrange differential operator, $\mathbf{D}$, which is ill-conditioned such that the condition number grows as $O(n^2)$ for first-order systems. Thus, applications in need of higher temporal resolution—such as satellite maneuver and collection planning—have relied upon suboptimal heuristics, inefficient algorithms, or optimal control via domain decomposition. In this thesis, well-conditioned pseudospectral optimal control methods are established, which use the Birkhoff integral operator that exhibits condition number stability of $O(1)$. By forming a well-conditioned system, these methods expand the applicability of optimal control. For satellite maneuver planning, this means the ability to optimize long-duration, low-thrust orbital maneuvers. Satellite collection planning can also be solved with optimal control formulations based on nonsmooth calculus. These high-resolution applications require many more nodes than ill-conditioned methods allow. Even low-resolution optimal control problems can see improvements in computation time through stability.				
14. SUBJECT TERMS optimal control, well-conditioned, pseudospectral, satellite maneuvers			15. NUMBER OF PAGES 173	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**WELL-CONDITIONED PSEUDOSPECTRAL OPTIMAL CONTROL METHODS
AND THEIR APPLICATIONS**

Nicholas R. Koeppen
Captain, United States Marine Corps
BS, University of Wisconsin-Madison, 2011

Submitted in partial fulfillment of the
requirements for the degrees of

AsTE, ASTRONAUTICAL ENGINEER

and

**MASTER OF SCIENCE IN ENGINEERING SCIENCE
(ASTRONAUTICAL ENGINEERING)**

and

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

**NAVAL POSTGRADUATE SCHOOL
June 2018**

Approved by: I. Michael Ross
Advisor

Ronald J. Proulx
Co-Advisor

Lucas C. Wilcox
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

Craig W. Rasmussen
Chair, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Pseudospectral optimal control is an established discipline with flight-proven results. Aerospace applications have included the implementation of minimum-time and zero-propellant maneuvers on high-value space assets. Standard pseudospectral methods have been sufficient for these and other applications that do not require more than approximately 250 nodes. Currently, pseudospectral optimal control uses the Lagrange differential operator, $\mathbf{D}$, which is ill-conditioned such that the condition number grows as $O(n^2)$ for first-order systems. Thus, applications in need of higher temporal resolution—such as satellite maneuver and collection planning—have relied upon suboptimal heuristics, inefficient algorithms, or optimal control via domain decomposition. In this thesis, well-conditioned pseudospectral optimal control methods are established, which use the Birkhoff integral operator that exhibits condition number stability of $O(1)$. By forming a well-conditioned system, these methods expand the applicability of optimal control. For satellite maneuver planning, this means the ability to optimize long-duration, low-thrust orbital maneuvers. Satellite collection planning can also be solved with optimal control formulations based on nonsmooth calculus. These high-resolution applications require many more nodes than ill-conditioned methods allow. Even low-resolution optimal control problems can see improvements in computation time through stability.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	The Path to Well-Conditioned Optimal Control	1
1.1	Optimal Control Example	1
1.2	Pseudospectral Foundation	3
1.3	Ill-Conditioned Effects	3
1.4	Well-Conditioned Schemes	5
1.5	Thesis Road Map	5
2	Solving High-Resolution Systems	7
2.1	Problem: Resolving an Impulse	7
2.2	Current Approaches	14
2.3	High Node Applications	14
2.4	Summary	15
3	Optimal Control Theory and Computation	17
3.1	The Time Domain	17
3.2	Notation	18
3.3	An Optimal Control Problem	19
3.4	Discretization	21
3.5	Feasibility and Optimality	22
3.6	The Nonlinear Program (NLP)	25
3.7	Basic Satellite Problems	26
3.8	Basic Problem 1: Minimum-Time Orbit Transfer	28
3.9	Basic Problem 2: A Maximum-Energy Orbit	31
3.10	From Concept to Application	34
4	Standard Pseudospectral Optimal Control	35
4.1	Lagrange Interpolation	35
4.2	Orthonormal Bases	38
4.3	Exact Quadrature	42

4.4	Lagrange Collocation (LCOL)	44
4.5	LCOL Implementation	46
4.6	Basic Problem Solutions using LCOL	48
4.7	Motivation for a Change	58
4.8	A Perfect Preconditioner	60
5	The Perfect Preconditioner	61
5.1	The Preconditioner	61
5.2	An Arbitrary Grid	64
5.3	First-Order Linear Systems	65
5.4	First-Order Validation	71
5.5	Second-Order Linear Systems	81
5.6	Second-Order Validation	87
5.7	Adapting to Optimal Control.	91
6	Well-Conditioned Psuedospectral Optimal Control	93
6.1	The Preconditioner for Optimal Control	93
6.2	Reformulations of the Problem	99
6.3	Well-Conditioned Optimal Control Collocation	107
6.4	BCOL Implementation	109
6.5	Basic Problem Solutions using BCOL	110
6.6	High Resolution Problems.	122
6.7	Method Analysis	131
7	Summary and Future Work	133
Appendix A	Formulations of B	135
A.1	First-Order: $\mathbf{B}_1$	135
A.2	Second-Order: $\mathbf{B}_2$	136
Appendix B	Example Problems	139
B.1	BP1: Minimum-Time Orbit Transfer.	139

B.2	BP2: Maximizing Orbital Energy	140
B.3	Lower-Thrust, Minimum-Time Orbit Transfers	140
B.4	Minimum-Time, LEO to GEO Orbit Transfers	142
Appendix C Notation		145
List of References		147
Initial Distribution List		151

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Figure 1.1	State solution to the double integrator control system, P1, using only a few Chebyshev–Gauss–Lobatto nodes, an Gauss quadrature node set. Errors relative to the analytical solution (1.2) are shown in the lower plot.	2
Figure 1.2	The image of a unit circle under the nominal $\mathbf{A}$ and perturbed $\tilde{\mathbf{A}}$. An example mapping of a given input vector $\mathbf{x}$ is shown. Extent of errors are given as blue ellipses.	4
Figure 1.3	The pre-image of a given $\mathbf{b}$ and corresponding $\mathbf{x}$ for the nominal $\mathbf{A}\mathbf{x} = \mathbf{b}$ and perturbed $\tilde{\mathbf{A}}\mathbf{x} = \mathbf{b}$ linear systems. Extent of errors are given as orange ellipses.	5
Figure 2.1	Growth in the number of nodes needed to capture a Gaussian function with decreasing bump width. A capture is defined by at least 2 nodes on the bump portion of the function.	9
Figure 2.2	Capture of a representative Gaussian function using three numbers of CGL nodes. $\sigma = 10^{-5}$ and $w = 6 \times 10^{-5}$	10
Figure 2.3	The same Gaussian function capture seen in Figure 2.2, but scaled to highlight only the “bump” portion of the function.	10
Figure 2.4	Pointwise error of barycentric Lagrange interpolation of a Gaussian function with $\sigma = 10^{-3}$. Interpolants from 10^5 nodes shown. . .	11
Figure 2.5	Maximum pointwise error of a Gaussian function with $\sigma = 10^{-3}$ as the number of nodes increases. Even with a significant number of nodes, equally-spaced nodes struggle to compete with CGL. . . .	12
Figure 2.6	Growth in the number of nodes needed to interpolate a Gaussian function with decreasing bump width. The interpolation threshold is set as having a maximum pointwise error 10^{-6}	13
Figure 2.7	Growth in the number of nodes needed to interpolate a Gaussian function with decreasing bump width. The interpolation threshold is set as having a maximum pointwise error 10^{-8}	13

Figure 2.8	Example state solution for optimal control for low-thrust using Bellman's principle. Source: [12].	14
Figure 3.1	Optimal control problem formulation flow, which demonstrates the covector mapping principle. Source: [18].	24
Figure 3.2	The basic satellite model used for BP1 and BP2.	26
Figure 3.3	BP1 visualization in canonical units of DU and angle in degrees. Initial and final satellite positions are red circles. The initial and final circular orbits are dashed black circles. The solution trajectory is blue.	29
Figure 3.4	Time-optimal controls to BP1 as determined by a 176-node DIDO solution.	30
Figure 3.5	Time-optimality as shown by a near-constant Hamiltonian value of -1 as determined by a 176-node DIDO solution.	30
Figure 3.6	BP2 visualization in canonical units of DU and angle in degrees. Initial and final satellite positions are red circles. The initial circular orbits is a dashed black circle. The solution trajectory is blue. . .	32
Figure 3.7	Optimal controls to BP2 as determined by a 176-node DIDO solution.	33
Figure 3.8	Optimality as shown by a near-constant Hamiltonian value of 0 as determined by a 176-node DIDO solution.	33
Figure 4.1	The Runge phenomenon using 10-node interpolants. Equally-spaced nodes are contrasted to Chebyshev–Gauss–Lobatto (CGL) nodes, an Gauss-like quadrature node set.	37
Figure 4.2	The Runge phenomenon for a 50-node interpolation. Absolute function values are used to show scale of oscillation, i.e., $ f(\tau) $	38
Figure 4.3	Control ($u = \beta$) trajectory for BP1 using 128 nodes. Both CGL and CGR solutions produces similar results, which closely track the DIDO optimal control solution.	48
Figure 4.4	Radial position (r) trajectory for BP1. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with <i>ode45</i> and a 128-node DIDO solution.	49

Figure 4.5	Radial (r) and angular (θ) position errors for BP1. L-CGL and DIDO errors are relative to an <i>ode45</i> -propagated solutions using the controls given by respective solution.	49
Figure 4.6	Velocity trajectories (v_r and v_t) for BP1. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with <i>ode45</i> and a 128-node DIDO solution.	50
Figure 4.7	Velocity trajectories errors (v_r and v_t) for BP1. L-CGL and DIDO errors are relative to an <i>ode45</i> -propagated solutions using the controls given by respective solution.	50
Figure 4.8	Maximum pointwise error of LCOL solutions to BP1 using Knitro in comparison to DIDO solutions. Error is relative to propagated solution using interpolated control solution.	51
Figure 4.9	Number of nonlinear function evaluations required for LCOL solutions to BP1 using Knitro. L-CGR can be faster, but is less reliable.	52
Figure 4.10	BP1 solution times for LCOL using Knitro in comparison to DIDO solutions. CGR solutions show less stability due to lack of +1 node.	52
Figure 4.11	Control ($u = \beta$) inputs over fixed time span of 50 TU. DIDO control solution provided for reference.	53
Figure 4.12	Radial position (r) trajectory for BP2. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with <i>ode45</i> and a 128-node DIDO solution.	54
Figure 4.13	Radial (r) and angular (θ) position errors for L-CGL and DIDO solutions. Error is relative to <i>ode45</i> -propagated solutions using controls given by respective solution.	54
Figure 4.14	Velocity trajectories (v_r and v_t) for BP2. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with <i>ode45</i> and a 128-node DIDO solution.	55
Figure 4.15	Velocity trajectories errors (v_r and v_t) for L-CGL and DIDO solutions. Error is relative to <i>ode45</i> -propagated solutions using controls given by respective solution.	55
Figure 4.16	Maximum pointwise error of LCOL solutions to BP2 using Knitro in comparison to DIDO solutions. Error is relative to propagated solution using interpolated control solution.	56

Figure 4.17	Number of nonlinear function evaluations required for LCOL solutions to BP2 using Knitro.	57
Figure 4.18	BP2 solution times for LCOL using Knitro in comparison to DIDO solutions.	57
Figure 4.19	Condition number growth of $\mathbf{D}_{in}$, the non-singular portion of the differential operator. By construction, $\mathbf{D}$ is always a singular matrix.	59
Figure 5.1	Exact solution to example first-order linear IVP (5.42) in the computational domain with a Birkhoff Collocation (BCOL) solution superimposed.	72
Figure 5.2	First-order IVP maximum absolute pointwise error. BCOL errors achieved with iterative <i>bicgstab</i> solver and <i>linsolve</i> are virtually the same. LCOL required the use of <i>linsolve</i>	73
Figure 5.3	First-order IVP condition number behavior disparity between collocation methods. See Section 5.3.1 for definitions of A_{B-CGR} and A_{L-CGR} . Note that BCOL is $O(1)$ and LCOL is $O(n^2)$	74
Figure 5.4	First-order IVP iteration count for <i>bicgstab</i> . All LCOL iterations do not converge.	75
Figure 5.5	First-order IVP solution time for BCOL and LCOL using MATLAB's <i>linsolve</i> direct solver. BCOL achieved the same results, but faster with <i>bicgstab</i>	76
Figure 5.6	First-order IVP solution time. BCOL requires less time using iterative <i>bicgstab</i> than LCOL, which required MATLAB's <i>linsolve</i> direct solver.	77
Figure 5.7	First-order IVP solution in $[0,2]$ domain with $x_0 = x(0) = 0$	78
Figure 5.8	The highly oscillatory behavior of this example first-order IVP when applied to $[0,2]$ domain.	78
Figure 5.9	First-order IVP maximum absolute pointwise error in $[0,2]$ domain. BCOL uses iterative <i>bicgstab</i> and LCOL uses direct <i>linsolve</i> . Note that all <i>bicgstab</i> iterations for LCOL return trivial results: error > 1	79
Figure 5.10	First-order IVP iteration count for <i>bicgstab</i> in $[0,2]$ domain. All LCOL iterations produce trivial results.	80

Figure 5.11	First-order IVP solution time in $[0,2]$ domain. Increase in number of nodes widens the gap in performance from Figure 5.6).	80
Figure 5.12	Second-order BVP maximum absolute pointwise error. BCOL uses iterative <i>bicgstab</i> solver and LCOL uses direct <i>linsolve</i>	88
Figure 5.13	Second-order BVP maximum absolute pointwise error for <i>bicgstab</i> solutions.	89
Figure 5.14	Second-order BVP iteration count for <i>bicgstab</i> . All LCOL solutions past 600 nodes produce trivial results.	89
Figure 5.15	Second-order BVP condition number behavior disparity between collocation methods. See Section 5.5.1 for definitions of A_{B-CGL} and A_{L-CGL} . Note that BCOL is $O(1)$ and LCOL is $O(n^4)$	90
Figure 6.1	Control jitter at final nodes of B-CGL solution to BP1.	97
Figure 6.2	Detail of B-CGL jitter shown in Figure 6.1.	97
Figure 6.3	Lack of control jitter in a B-CGR solution to BP1.	98
Figure 6.4	Detail of final nodes in B-CGR solution shown in Figure 6.3. . .	98
Figure 6.5	The condition growth for a right-preconditioning scheme using a mix node sets: $\mathbf{D}_{in,CGL}$ and $\mathbf{B}_{in,CGR}$	103
Figure 6.6	Condition number growth of the imperfect left preconditioner $\mathbf{B}_{in}\mathbf{D}_{in} = \mathbf{I}_{n-1}$ using the CGL 2 (6.47) formulation. Trends as $O(n)$	106
Figure 6.7	Condition number growth of the imperfect right preconditioner $\mathbf{D}_{in}\mathbf{B}_{in} = \mathbf{I}_{n-1}$ using the CGL 2 (6.47) formulation. Trends as $O(n^4)$	106
Figure 6.8	128-node control solutions for minimum-time solution to BP1. Both well-conditioned methods—PL-CGR and B-CGR—are effectively coincident. L-CGL control solution from Section 3.8 provided for reference.	111
Figure 6.9	Radial position (r) trajectory for BP1 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with <i>ode45</i> . The LCOL CGL (L-CGL) solution from Section 3.8 using 128 nodes is shown for comparison. . . .	112

Figure 6.10	Radial (r) and angular (θ) position errors for B-CGR and L-CGL solutions to BP1. Error is relative to <i>ode45</i> -propagated solutions using controls given by respective solution.	112
Figure 6.11	Velocity trajectories (v_r and v_t) for BP1 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with <i>ode45</i> . The LCOL CGL (L-CGL) solution from Section 3.8 using 128 nodes is shown for comparison. . . .	113
Figure 6.12	Velocity trajectories errors (v_r and v_t) for B-CGR and L-CGL solutions to BP1. Error is relative to <i>ode45</i> -propagated solutions using controls given by respective solution.	113
Figure 6.13	Maximum pointwise error when using Knitro to solve BP1. Error is measured against <i>ode45</i> -propagated states when using solution controls.	114
Figure 6.14	Evaluations of the BP1 nonlinear function needed for Knitro solution.	115
Figure 6.15	Knitro solution time for BP1 using different collocation schemes.	116
Figure 6.16	Radial position (r) trajectory for BP2 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with <i>ode45</i> . The LCOL CGL (L-CGL) solution from Section 3.9 using 128 nodes is shown for comparison. . . .	117
Figure 6.17	Radial (r) and angular (θ) position errors for B-CGR and L-CGL solutions to BP2. Error is relative to <i>ode45</i> -propagated solutions using controls given by respective solution.	117
Figure 6.18	Velocity trajectories (v_r and v_t) for BP2 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with <i>ode45</i> . The LCOL CGL (L-CGL) solution from Section 3.9 using 128 nodes is shown for comparison. . . .	118
Figure 6.19	Velocity trajectories errors (v_r and v_t) for B-CGR and L-CGL solutions to BP2. Error is relative to <i>ode45</i> -propagated solutions using controls given by respective solution.	118
Figure 6.20	Control ($u = \beta$) inputs for BP2 over the fixed time span of 50 TU. L-CGL control solution from Section 3.9 provided for reference.	119

Figure 6.21	Maximum pointwise error when using Knitro to solve BP2. Error is measured against <i>ode45</i> -propagated states when using solution controls.	120
Figure 6.22	Evaluations of the BP2 nonlinear function needed for Knitro solution.	121
Figure 6.23	Knitro solution time for BP2 using different collocation schemes.	122
Figure 6.24	State trajectory for HR1: the intermediate-thrust ($A = 0.001$) adaption of BP1. Solution was found with 320 nodes via B-CGL. . .	123
Figure 6.25	Intermediate-thrust ($A = 0.001$) control solution for HR1 using 320 nodes and the B-CGL method.	124
Figure 6.26	State trajectory for HR2: the low-thrust ($A = 5 \times 10^{-4}$) adaption of BP1. Solution was found with 320 nodes via B-CGL.	125
Figure 6.27	Very low-thrust ($A = 5 \times 10^{-4}$) control solution to the a minimum-time problem similar to BP1 using 512 nodes and the B-CGL method.	126
Figure 6.28	State trajectory for HR3 using 352 nodes via B-CGL.	127
Figure 6.29	The control solution for HR3 ($A = 0.005$) using 352 nodes and the B-CGL method.	128
Figure 6.30	Position and corresponding error for 352-node B-CGL solution for HR3 ($A = 0.005$). Error relative to <i>ode45</i> -propagated solution using <i>pchip</i> to interpolate controls.	129
Figure 6.31	Velocity and corresponding error for 352-node B-CGL solution for HR3 ($A = 0.005$). Error relative to <i>ode45</i> -propagated solution using <i>pchip</i> to interpolate controls.	129
Figure 6.32	State trajectory for HR4 using 352 nodes via B-CGL.	130
Figure 6.33	Intermediate-thrust ($A = 0.001$) control solution for HR4 using 352 nodes and the B-CGL method.	131

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 5.1	Validation of exact same condition numbers as in [1] with unpublished additional significant figures.	75
Table 5.2	Second-order BVP condition number growth/stability	90
Table C.1	Summary of optimal control variables	145
Table C.2	Various measures of dimensionality	145

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:

The Path to Well-Conditioned Optimal Control

Dynamic optimization techniques offer the opportunity to streamline processes by facilitating more efficient use of resources over a defined time horizon. For engineering applications, this dynamic optimization is generally focused on finding optimal controls to a dynamic system in order to minimize a cost function. These optimal control problems require care in modeling the relevant states of the system and their dynamics, determining the appropriate constraints, and drafting the correct cost function in order to determine how optimality is defined. Once modeled, they are solved numerically using pseudospectral methods. The improvement in the stability of these methods is the focus of this thesis

A simple optimal control problem is shown in Section 1.1. Next, Sections 1.2 and 1.3 cover the standard pseudospectral methods used to solve these problems numerically. However, the standard method of Lagrange collocation is ill-conditioned and needs to be reformulated in order to remain stable with a large number of nodes [1]. This is covered briefly in Sections 1.3 and 1.4 and will be discussed in depth in Chapters 4 and 5.

1.1 Optimal Control Example

A simple example control system is a rest-to-rest maneuver using double integrator kinematics and defined as

$$(P1) \left\{ \begin{array}{l} \text{Minimize} \quad \frac{1}{2} \int_{t_0}^{t_f} \mathbf{u}^2 dt, \\ \text{Subject to:} \\ \quad \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ u \end{bmatrix}, \\ \quad \begin{bmatrix} x_0 & v_0 & t_0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \\ \quad \begin{bmatrix} x_f & v_f & t_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}. \end{array} \right. \quad (1.1)$$

From the formulation shown in (1.1), the optimal controls are defined by the minimum accelerations needed to execute a rest-to-rest maneuver and set as a running cost on control

effort. For a solution to be feasible, it must satisfy well-posed constraints. In this case, the object at rest ($v_0 = 0$) moves a unit distance ($[x_0, x_f] = [0, 1]$) and returns to a resting position ($v_f = 0$) after one unit of time ($[t_0, t_f] = [0, 1]$). The analytical solution to P1 is

$$\begin{bmatrix} x(t) \\ v(t) \\ u(t) \end{bmatrix} = \begin{bmatrix} -2t^3 + 3t^2 \\ -6t^2 + 6t \\ -12t + 6 \end{bmatrix}. \quad (1.2)$$

The result to this simple problem can be found numerically using a variety of methods. For more complicated problems, collocation schemes using pseudospectral operators are employed. As seen in Figure 1.1, these methods can achieve at or near machine precision solutions quickly and with only a few discrete points.

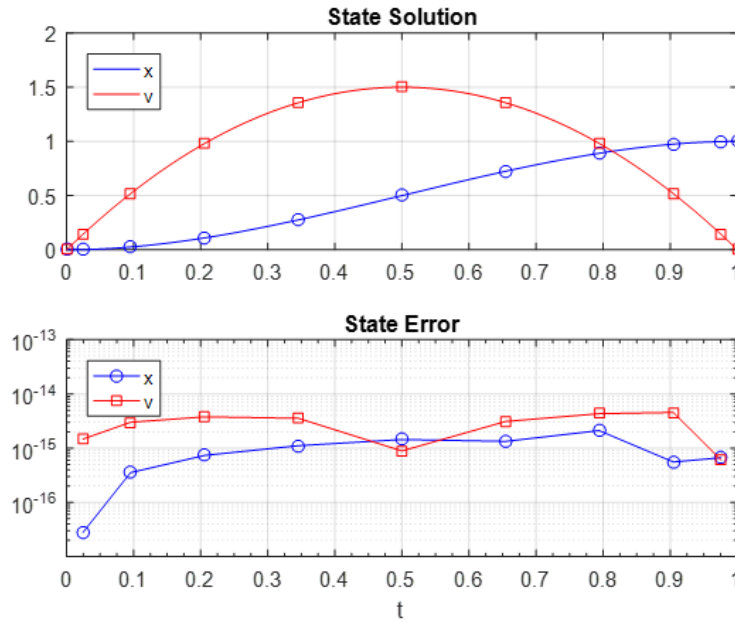


Figure 1.1. State solution to the double integrator control system, P1, using only a few Chebyshev–Gauss–Lobatto nodes, an Gauss quadrature node set. Errors relative to the analytical solution (1.2) are shown in the lower plot.

1.2 Pseudospectral Foundation

Given a well-posed optimal control problem, like (1.1), pseudospectral schemes can be employed to arrive at discrete solutions for optimal controls and the resultant system states. In fact, standard pseudospectral optimal control methods have demonstrated impressive results in space systems applications. It was used to achieve minimum-time slew maneuvers for the TRACE spacecraft [2] and reduce, or eliminate the propellant needed to perform maneuvers of the International Space Station [3]. These methods are the foundation of the commercial optimal control software, DIDO [4]. However, the standard pseudospectral method, Lagrange collocation, is a scheme based on an inherently ill-conditioned operator: the pseudospectral differential operator, $\mathbf{D}$. The basic idea of Lagrange collocation is to approximate the derivative of a function, $\phi(\tau)$, such that

$$\frac{d}{d\tau}(\phi(\tau)) \approx \mathbf{D}\phi(\tau), \quad (1.3)$$

where $\tau \in [-1, 1]$ is time as mapped to the computational domain and τ is a vector of discrete evaluation points called nodes.

1.3 Ill-Conditioned Effects

Generally, ill-conditioned behavior can be understood as an instability in which small changes in inputs result in large changes in outputs. In some cases, a desired precision in output is infeasible since the required input precision cannot be achieved numerically—being within machine precision. Therefore, ill-conditioned systems may not be solvable with a given machine precision. More often, ill-conditioned systems require more time to converge to a solution due to wide oscillations in outputs while the iterative method attempts to make small adjustments in input.

As an example, one can envision a two-dimensional system with an ill-conditioned operator, $\mathbf{A}$, which maps the input vector, $\mathbf{x}$, of a unit circle to ever-elongating ellipse as the condition number grows. As Figure 1.2 shows, this elongation further complicates accurate mapping if one considers minor perturbations to $\mathbf{A}$, as seen via $\tilde{\mathbf{A}}$. For a linear system of equations, $\mathbf{A}\mathbf{x} = \mathbf{b}$, the ill-conditioned operations can result in significant error and different solution spaces depending on minor perturbations to the operator $\mathbf{A}$. This is shown in Figure 1.3.

In particular for $\mathbf{D}$, the condition number of as a p th-order differential operator increases as $O(n^{2p})$ for n nodes [1], [5]. In the optimal control case, only a first-order operator is required and thus the condition number for Lagrange collocation is $O(n^2)$.

For more than two decades, mathematicians have been working to reduce the effects of this ill-conditioned operator [5]. Mitigating these effects enables Lagrange collocation to use more nodes to solve systems in need of higher resolution. In particular, Therefore, additional methods of domain decomposition and preconditioning have been used to reduce the effect of this condition number growth.

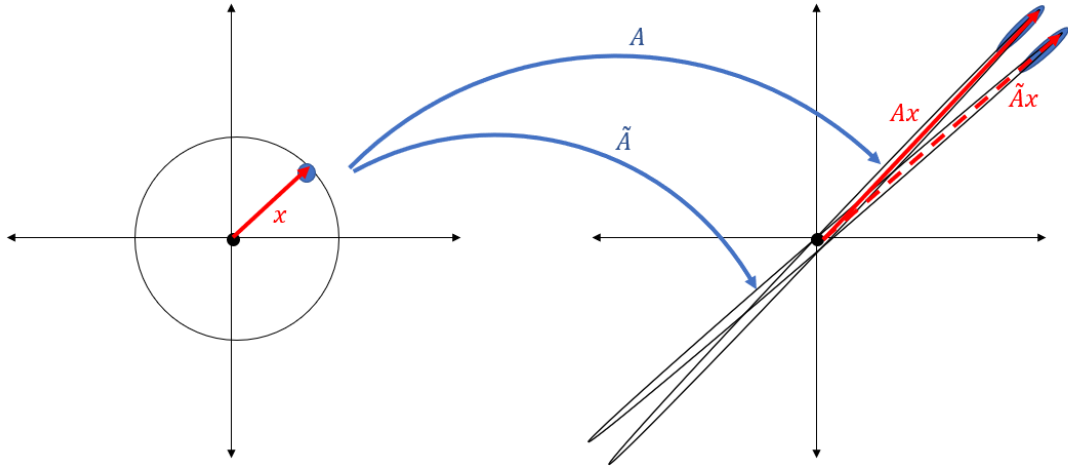


Figure 1.2. The image of a unit circle under the nominal $\mathbf{A}$ and perturbed $\tilde{\mathbf{A}}$. An example mapping of a given input vector x is shown. Extent of errors are given as blue ellipses.

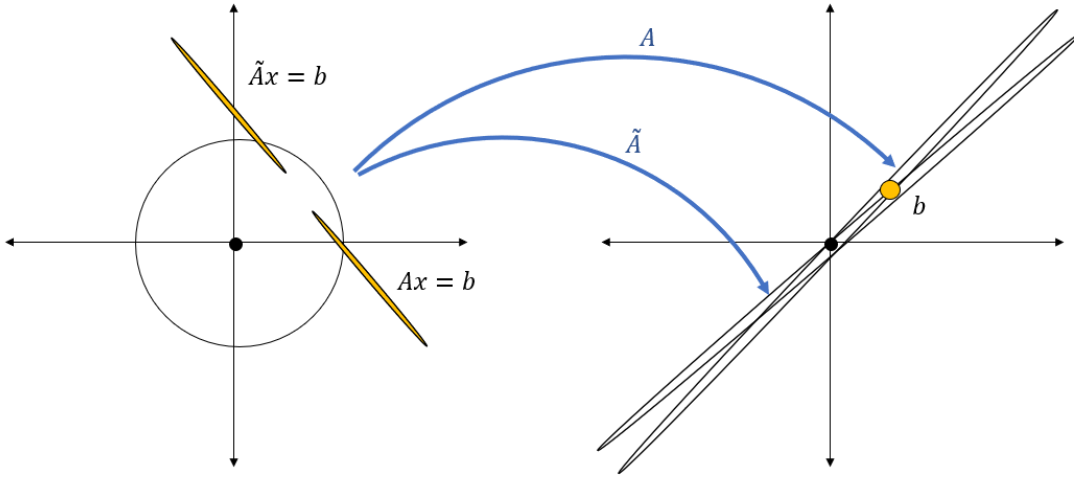


Figure 1.3. The pre-image of a given $\mathbf{b}$ and corresponding $\mathbf{x}$ for the nominal $\mathbf{A}\mathbf{x} = \mathbf{b}$ and perturbed $\tilde{\mathbf{A}}\mathbf{x} = \mathbf{b}$ linear systems. Extent of errors are given as orange ellipses.

1.4 Well-Conditioned Schemes

Recent reformulations of spectral and pseudospectral collocation [1], [6]–[8] have shown that by using integral operators the collocation scheme can remain well-conditioned regardless of the number of nodes used: $O(1)$. Unfortunately, most authors on this subject focus on linear problems instead of nonlinear applications like optimal control. This thesis seeks to take well-conditioned linear pseudospectral techniques and to apply them to optimal control in order to reap the benefits of stability in resolving the system’s dynamics at high node counts as well as reduce the number of iterations needed to arrive at feasible and optimal solutions.

1.5 Thesis Road Map

Chapter 2 begins the discussion into well-conditioned methods by highlighting the need for high resolution. Next, Chapter 3 discusses the basics of optimal control and introduces a few basic optimal control problems. With an understanding of how optimal control problems are formulated, Chapter 4 discusses the standard Lagrange collocation method as applied to optimal control. This chapter sets a baseline framework for the implementation

of the well-conditioned optimal control methods. Chapter 5 then dives into the details of the pivotal integral operator needed to create a new collocation scheme called Birkhoff collocation. This chapter contains both first- and second-order linear systems derivations and results. With all the pieces assembled, Chapter 6 applies integral operators to optimal control in order to form well-conditioned pseudospectral optimal control.

CHAPTER 2: Solving High-Resolution Systems

Current pseudospectral optimal control methods have proven effective for numerous real-world applications [2], [3], [9], but have been limited by the number of nodes in the problem discretization for reasons discussed in Chapter 1. This node limitation restricts naive implementations of pseudospectral optimal control to dynamics that do not involve impulsive or oscillatory characteristics that need to be captured in order to have a meaningful solution. For these high-resolution systems, the basic pseudospectral method requires modification via preconditioning or knotting [10].

This chapter provides more detail on the need for larger node pseudospectral constructions and current practices for overcoming this limitation. Section 2.1.2 demonstrates the high node requirement for impulsive systems. Next, Section 2.2 highlights the approaches being used currently to overcome this requirement for a large number of nodes. The chapter concludes with a brief overview of physical applications for these approaches in Section 2.3.

2.1 Problem: Resolving an Impulse

A Gaussian function will be used as a proxy of varying width impulses in order to highlight the need for increasing the number of nodes used in a numerical solution. Resolving the function will be shown in two steps. First, in Section 2.1.2, the “capture” of the $\pm 3\sigma$ region will be considered the placement of at least two nodes in this “bump” region. Next, and most applicable to optimal control, the interpolation of the Gaussian is shown in Section 2.1.3. In both sections, Chebyshev–Gauss–Lobatto (CGL) nodes are used to capture and interpolate these functions. More on these orthogonal polynomials and Gauss-like quadrature nodes can be found in Section 4.2.

2.1.1 The Gaussian Distribution

Generally, a Gaussian distribution is defined as

$$f(t) = a \exp\left(\frac{-(t - \mu)^2}{2\sigma^2}\right), \quad (2.1)$$

where μ is the mean, σ is the standard deviation, and a is the height of the peak. The independent variable t represents a point in the unit time domain used here,

$$t \in [0, 1]. \quad (2.2)$$

The improper integral of $f(t)$ is the Gauss integral. Via the Gauss integral, it can be determined that if

$$a = \frac{1}{\sigma\sqrt{2\pi}}, \quad (2.3)$$

then the improper integral is always equal to 1, i.e.,

$$\int_{-\infty}^{\infty} f(t)dt = 1. \quad (2.4)$$

This value for a results in $f(t)$ being the probability density function of a normal distribution. This can be particularly useful in probabilities and other integration applications. In this section, a single Gaussian function with a unity peak height ($a = 1$) is centered in the middle of a unit time domain, thus $\mu = 0.5$.

By changing σ , variously-sized impulses can be represented. A capture of an impulse is considered as placing at least two nodes on the “bump” portion of the Gaussian function. This “bump” is defined by the portion of the function within $\pm 3\sigma$ of the mean, i.e.,

$$f_{\text{bump}}(t) = f(t) \quad t \in [\mu - 3\sigma, \mu + 3\sigma]. \quad (2.5)$$

As is the case with a normal distribution, 99.7% of the area under $f(t)$ is contained within this $\pm 3\sigma$ “bump” domain. Therefore, this “bump” width, is defined as

$$w_{\text{bump}} = 6\sigma. \quad (2.6)$$

This characteristic width is used to demonstrate the time span of an impulse relative to the domain. In other words, the Gaussian serves as a proxy for the ability of an arbitrary discretization scheme to capture and interpolate impulses in a dynamic system.

2.1.2 Impulse “Capture”

It becomes quickly apparent that, for short-duration impulses, thousands of nodes can be required to simply “capture” the dynamics of system, as shown in Figure 2.1.

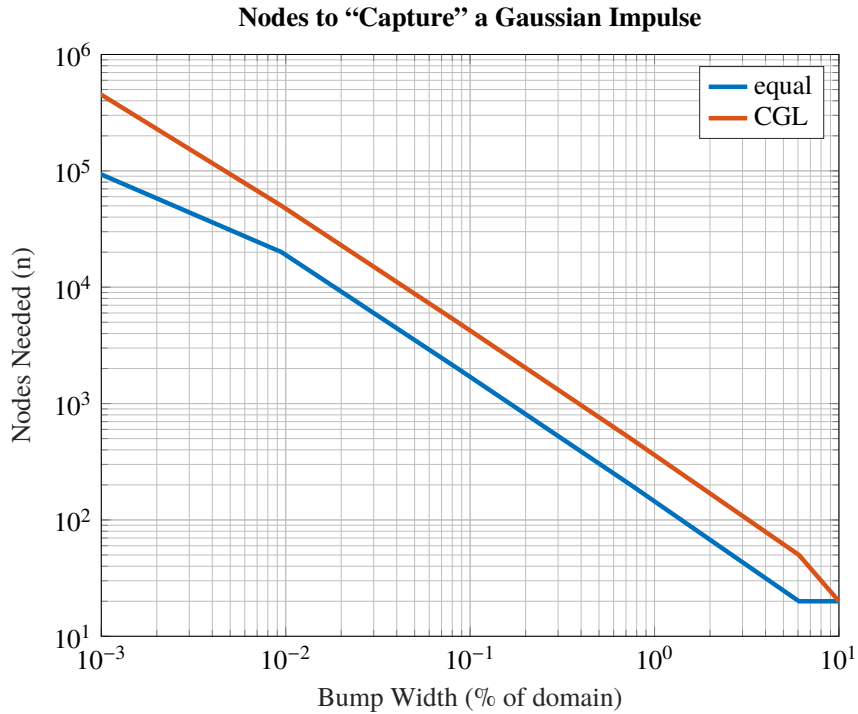


Figure 2.1. Growth in the number of nodes needed to capture a Gaussian function with decreasing bump width. A capture is defined by at least 2 nodes on the bump portion of the function.

Figure 2.1 shows that fewer equally-spaced nodes are required to “capture” the impulse than for the CGL case. This is primarily to the endpoint clustering seen with CGL and the fact that this test case has the impulse in the middle of the domain. See Figure 2.2 and Figure 2.3 for a single example impulse. Importantly, in each of these figures, the interpolating polynomial does include these few points on the bump, but instead provide a consistent zero value. Therefore, this metric of “capture” is not sufficient for dynamic

optimization methods. These methods will fail to resolve the impulse, even with a significant number of nodes, but it does make a simple, visual argument for the need for higher number of nodes.

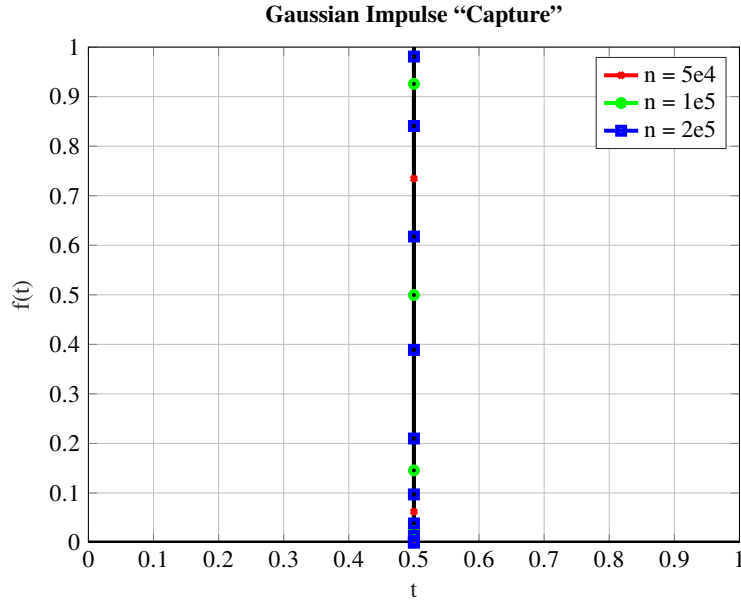


Figure 2.2. Capture of a representative Gaussian function using three numbers of CGL nodes. $\sigma = 10^{-5}$ and $w = 6 \times 10^{-5}$.

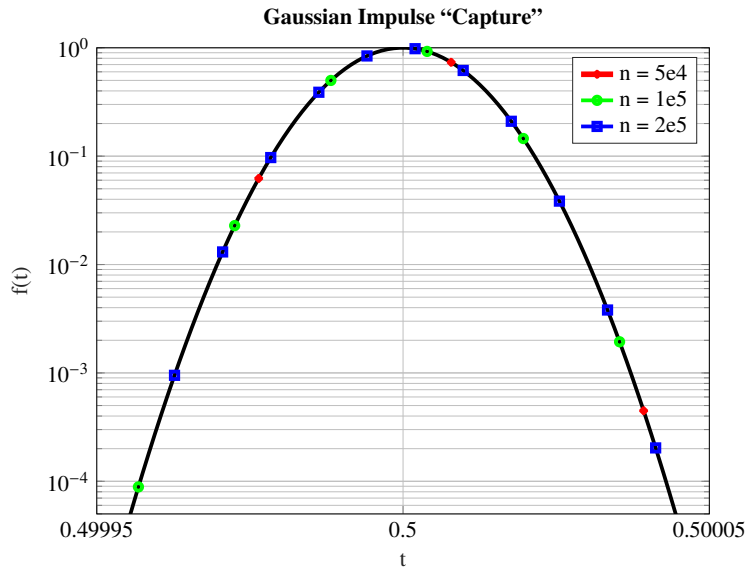


Figure 2.3. The same Gaussian function capture seen in Figure 2.2, but scaled to highlight only the “bump” portion of the function.

2.1.3 Impulse Interpolation

As discussed, simply placing nodes on impulse is just a simple illustration of whether an impulsive phenomenon may be detected. A more accurate method for testing the ability of a numeric method to resolve the relevant feature is through interpolation error. In this task, even more nodes are required to change the interpolating polynomial shape near the “bump” portion. For interpolation, a Gaussian with $\sigma = 10^{-3}$ is used.

Figure 2.4 shows how interpolation of this function with 10^5 nodes requires a significant amount of nodes to reach a maximum error near machine precision. Notably, equally-spaced nodes suffer near the impulse and at the endpoints of the domain. Gauss-like quadrature nodes like CGL perform much better with fewer nodes. For more details refer to Section 4.2.

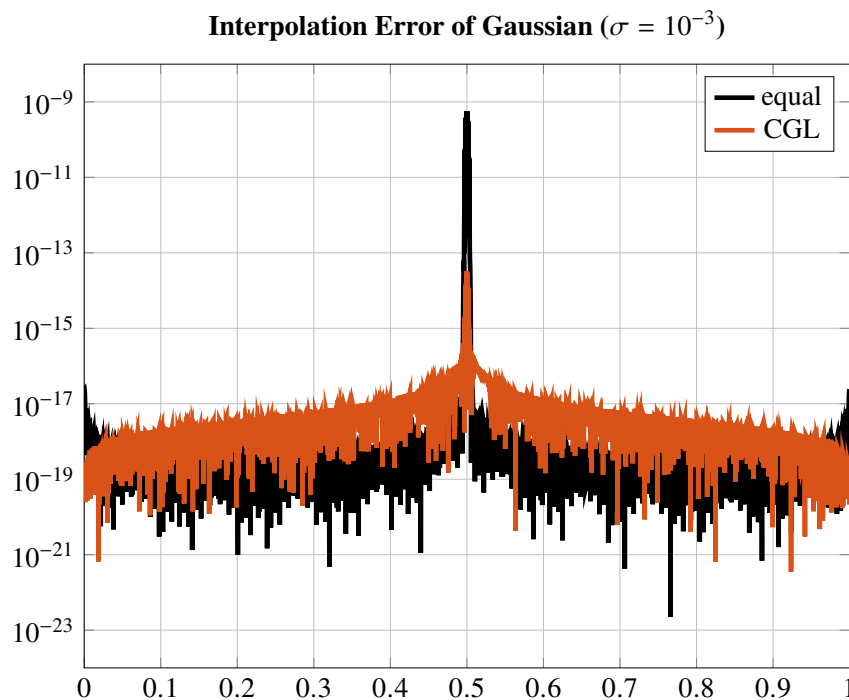


Figure 2.4. Pointwise error of barycentric Lagrange interpolation of a Gaussian function with $\sigma = 10^{-3}$. Interpolants from 10^5 nodes shown.

This difference in performance with the choice of nodes is even more apparent in Figure 2.5. As can be seen in that figure, errors persist until the bump is represented in the interpolating polynomial. At this critical point, Gauss-like quadrature nodes rapidly reduce pointwise error, but equally-spaced nodes only slowly improve.

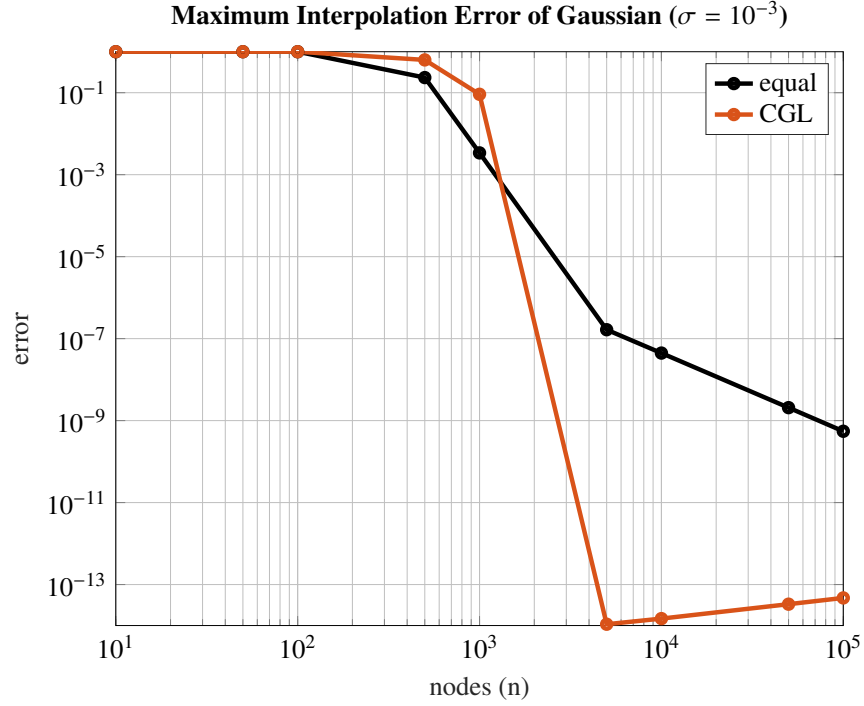


Figure 2.5. Maximum pointwise error of a Gaussian function with $\sigma = 10^{-3}$ as the number of nodes increases. Even with a significant number of nodes, equally-spaced nodes struggle to compete with CGL.

For $\sigma = 10^{-3}$, the critical number of nodes is approximately 10^3 . At this point in Figure 2.5, the interpolation error using CGL nodes decreases rapidly. Equally-spaced nodes do not exhibit this rapid decrease and are orders of magnitude higher in interpolation error. More will be discussed on the advantages of Gauss-like quadrature nodes in Chapter 4.

More importantly to the immediate goal of impulse resolution, as the bump width decreases, the number of nodes needed to meet a maximum allowable interpolation error increases significantly. This can be seen in Figure 2.6 with a threshold of 10^{-6} and Figure 2.7 with a threshold of 10^{-8} . Note that in Figure 2.7 equally-spaced nodes also show instability in interpolating the wider Gaussian functions.

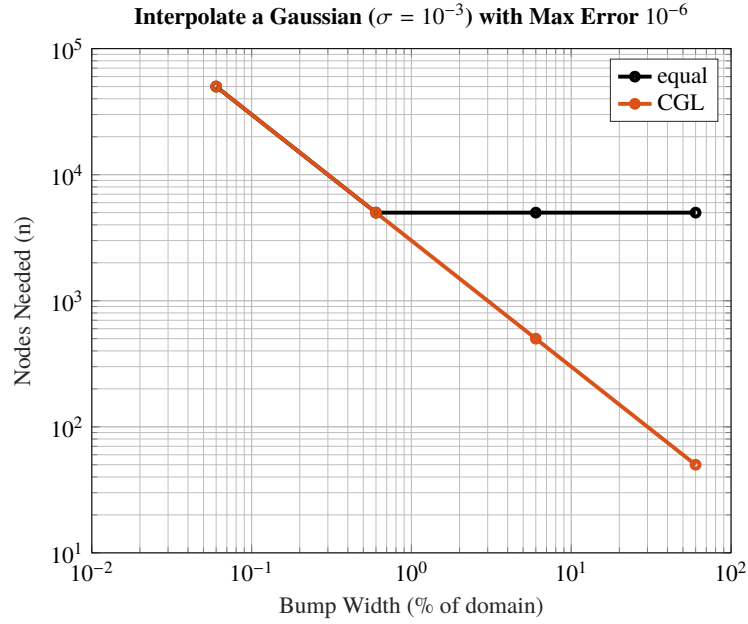


Figure 2.6. Growth in the number of nodes needed to interpolate a Gaussian function with decreasing bump width. The interpolation threshold is set as having a maximum pointwise error 10^{-6} .

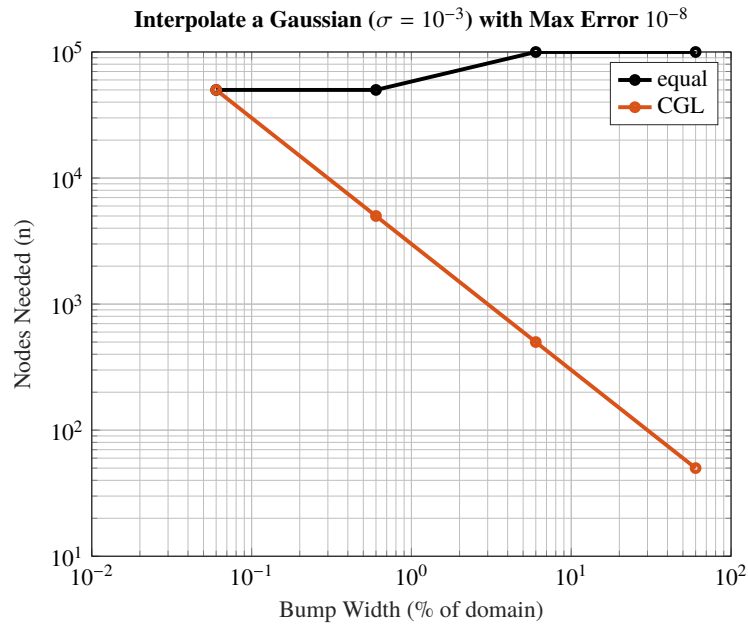


Figure 2.7. Growth in the number of nodes needed to interpolate a Gaussian function with decreasing bump width. The interpolation threshold is set as having a maximum pointwise error 10^{-8} .

2.2 Current Approaches

This need for high resolution is limited at the outset by ill-conditioned operators as discussed in Chapter 1 and as will be shown in Section 4.7. These ill-conditioned operators can be mitigated via domain decomposition. However, this often means introduces inelegant knotting at the domain boundaries [10]. Other integration-based techniques seek to modify the ill-conditioned effects through preconditioning, but due to formulation with weak boundary conditions the collocation scheme still has condition number growth of $O(\sqrt{n})$ for Dirichlet boundary conditions [5]. Other approaches to dynamic optimization problems, like the traveling salesman and orienteering problems, take significant effort to transform the entire dynamic problem into a graph problem [11].

2.3 High Node Applications

The aerospace applications for high resolution optimal control continue to grow. High specific impulse engines can extend the life of a spacecraft and are now ubiquitous as a means of satellite propulsion. However, this typically results in lower thrust levels and short duration pulses, which are required over a longer total burn time. The challenge of low-thrust optimal control mirrors that of the example impulse discussed in this chapter and can be seen visually in Figure 2.8.

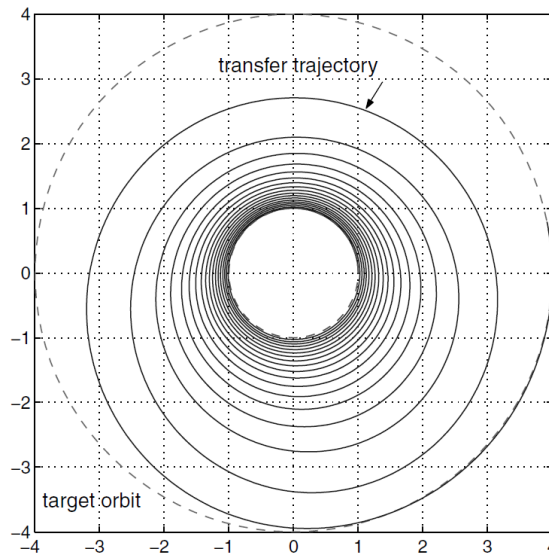


Figure 2.8. Example state solution for optimal control for low-thrust using Bellman's principle. Source: [12].

Furthermore, operationally-significant satellite collection planning problems can now be solved with the proper resolution. This collection problem is an instance of the orienteering problem, which is recognized as a non-deterministic, polynomial-time hard (NP-Hard) problem. Given a new optimal control formulation [13], the orienteering problem can be solved entirely via the constrained optimization framework of optimal control. However, the key component in this formulation is the resolution of approximations of an indicator function. By correctly resolving the placement and integration of the indicator functions, the binary behaviors of whether a target was visited can be determined.

2.4 Summary

As shown via the capture and interpolation of a Gaussian, a large number of nodes are necessary to capture impulsive behaviors in a dynamic system. The same applies for oscillatory problems. Results on these types of systems are shown throughout this thesis. Well-conditioned methods allow these systems to be solved with significantly more nodes than currently employed in optimal control. See Chapter 6 for these results. Chapters 3–5 explain the foundational knowledge of optimal control, pseudospectral methods, Lagrange collocation, and preconditioning that make these results possible.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:

Optimal Control Theory and Computation

Even though Chapter 2 begins to demonstrate the need for well-conditioned collocation methods, one must first understand optimal control and the standard ill-conditioned methods used previously. This chapter lays the foundation for how optimal control problems are formulated, designed, and analyzed.

In Section 3.1 and Section 3.2, notation and concepts are defined, with a particular emphasis on the time domain. The bulk of the content in these sections are reproduced in Appendix C for reference. Section 3.3 proceeds with the details of optimal control problem formulations. Section 3.5 briefly covers the foundations of optimality via Pontryagin's principle and the mapping of optimal control boundary value problems. Finally, Section 3.7 provides two basic satellite control problems that are used for the results shown in Chapter 4 and Chapter 6.

3.1 The Time Domain

Time is the primary independent variable that drives all optimal control formulations since the problems are, by their nature, dynamic optimization. This physical time domain is defined as $t \in [t_0, t_f]$, where t_0 and t_f are the initial and final times, respectively. However, many spectral methods are defined in the computational domain from $[-1, 1]$. When the computational domain is used, $\tau \in [-1, 1]$ is used in place of the physical time variable, t . Therefore, pseudospectral optimal control requires an affine transformation from the computational domain on which nodes are typically defined to the physical time domain.

3.1.1 Domain Transformation

Optimal control formulations require a mapping between the computational domain and the physical domain, i.e., $[t_0, t_f]$. The function for mapping from the computational to physical domain, $\Gamma(\tau)$, is defined such that

$$t = \Gamma(\tau) = \left(\frac{t_f - t_0}{2} \right) \tau + \frac{t_f + t_0}{2}. \quad (3.1)$$

The reverse mapping (i.e., physical to computational) is defined as

$$\tau = \Gamma^{-1}(t) = \left(\frac{2}{t_f - t_0} \right) t + \frac{t_f + t_0}{t_f - t_0}. \quad (3.2)$$

By default, any matrix operator in this thesis is defined in the computational domain. However, when defined in the physical domain, a bar over the variable is used, e.g., $\bar{\mathbf{D}}$ is the differentiation operator in the physical domain.

3.2 Notation

At this point, a brief aside is needed to clarify the notation of this thesis in order to avoid confusion among different sources. The interdisciplinary nature of this research means merging mathematical notation and conventions that are often in conflict between linear and optimal control applications. This thesis uses optimal control conventions as espoused in [14], but attempts to provide continuity to other sources where possible. Therefore, many equations and formulations in source materials are reproduced in this thesis for notational continuity and ease of reading. For those involved in optimal control can most likely skip ahead and reference this section, or Appendix C, as needed.

3.2.1 Domain Discretization

Pseudospectral optimal control problems specifically require discretization of the time domain in order to find numerical solutions. Thus, the time domain is split into N divisions, as is common for interpolation references and in [1], [15], [16]. The number of nodes, $n = N + 1$, is more a common metric used in optimal control applications [17]. This allows for the nodes, τ_i , to be indexed from 0 to N , i.e., $\boldsymbol{\tau} = (\tau_0, \tau_1, \dots, \tau_N)^T$. At times it may be convenient to restrict this domain to an “inner” portion such that $\boldsymbol{\tau}_{\text{in}} \subset \boldsymbol{\tau}$. When done so, a vector of indices, $\mathbf{i}_{\text{in}}$, will be defined where $\boldsymbol{\tau}_{\text{in}} = (\tau_i) \forall i \in \mathbf{i}_{\text{in}}$. Since time is the primary independent variable of the system, the discretization of the time domain propagates through the entire formulation of the problem.

3.2.2 Dimensionality

Scalars are represented as italic variables without emphasis, e.g., w_j for a specific quadrature weight. Vectors are presented as bold, lowercase, and italicized, e.g., $\mathbf{t}$ for a physical time

vector. Matrices will be bold, uppercase, and roman, e.g., $\mathbf{D}$ for a differentiation matrix.

In line with this styling, single functions are without emphasis, e.g., $x_1(t)$ represents the first state x_1 evaluated at time t . A vector of functions is bold, e.g., $\mathbf{x}(t)$ represents all states being evaluated at time t . In each case, the relevant independent variable(s) will be listed if specifically evaluated via that parameter. Functions that are not specifically evaluated are represented with function space notation, e.g., $x(\cdot)$. See [17] for more details.

3.2.3 Variables

The system's states, $\mathbf{x}$, are explicitly subject to dynamics, $\dot{\mathbf{x}}$, and other constraints, some of which can have a time dimension: path constraints, $\mathbf{h}$. Optimal control, as the name suggests, seeks a solution trajectory for all control variables, $\mathbf{u}(t)$, at any given time, t , which optimizes a given objective function. This variable notation can be particularly confusing due to the general use of x as an independent variable and the use of $u(x)$ as the dependent variable in major source documents for this thesis [1]. Therefore, equations and properties are reproduced to avoid confusion.

3.2.4 Bounds

Bounds on variables will be shown as between a lower limit and an upper limit, e.g., x_1^L and x_1^U would be the lower and upper bounds of the first variable, respectively. Numerical bounds will be represented by superscripts, e.g., $x_{1,f} = x_1^F$, such that x_1^F is known or can be determined through evaluation. By contrast, $x_{1,f}$ is simply shorthand notation for $x_1(t_f)$.

3.3 An Optimal Control Problem

Optimal control problems can be framed as a nonlinear constrained minimization problem defined in the physical time domain: $t \in [t_0, t_f]$. First, the nonlinear system is bisected into two sets of functions: states, $\mathbf{x}$, and controls, $\mathbf{u}$. The number of state and controls variables are n_x and n_u , respectively. Therefore,

$$\mathbf{x}(t) \in \mathbb{R}^{n_x} \quad \text{and} \quad \mathbf{u}(t) \in \mathbb{R}^{n_u}. \quad (3.3)$$

By formulation a control variable is unconstrained by its dynamics. If dynamical constraint on a control is required, the control is “elevated” to a state and its derivative becomes a control. Next, the states are constrained by their dynamics,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad (3.4)$$

endpoint conditions,

$$\mathbf{e}^L \leq \mathbf{e}(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) \leq \mathbf{e}^U, \quad (3.5)$$

where $\mathbf{x}_0 = \mathbf{x}(t_0)$ and $\mathbf{x}_f = \mathbf{x}(t_f)$, and path constraints,

$$\mathbf{h}^L \leq \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{h}^U. \quad (3.6)$$

These endpoint and path constraints are component-wise inequalities relative to vectors of lower and upper bounds, denoted by superscripts of L and U , respectively. Thus, in general, these constraints are viewed as bounded function values, but can be set to a fixed value via equality of the lower and upper bounds, e.g., $e_1^L = e_1 = e_1^U$. Finally, the minimization is accomplished via an objective function, J , which is also referred to as the cost function. The cost function, J , can also be segregated into two parts: the endpoint cost, E , and the running cost, F . These costs are nonlinear functions of the states and controls of the dynamical system such that

$$J(\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_0, t_f) = E(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) + \int_{t_0}^{t_f} F(\mathbf{x}(t), \mathbf{u}(t), t) dt. \quad (3.7)$$

Altogether, the generic optimal control problem, P, is defined concisely as follows:

$$(P) \left\{ \begin{array}{ll} \text{Minimize} & J(\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_0, t_f) \\ & = E(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) + \int_{t_0}^{t_f} F(\mathbf{x}(t), \mathbf{u}(t), t) dt, \\ \text{Subject to:} & \\ & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \\ & \mathbf{e}^L \leq \mathbf{e}(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) \leq \mathbf{e}^U, \\ & \mathbf{h}^L \leq \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{h}^U. \end{array} \right. \quad (3.8)$$

3.4 Discretization

Optimal control problems require discretization of the time domain in order to find numerical solutions. The immediate result of this discretization is the formation of a time vector, $\mathbf{t} = (t_0, t_1, \dots, t_N)^T$, with a dimensionality of $n \times 1$. At each timestep, t_i , all state variables must possess a value, which corresponds to a coefficient in a Lagrange polynomial approximation of the states of the continuous problem P. Therefore, a state matrix $\mathbf{X}$, exists such that each row represents all states at a given time, i.e., each element of $\mathbf{X}$, $x_{i,j} = x_j(t_i)$ where $0 \leq i \leq N$ and $1 \leq j \leq n_x$. The same is true for controls. Thus, $\mathbf{X}$ is an $n \times n_x$ matrix and $\mathbf{U}$ is an $n \times n_u$ matrix where

$$\mathbf{X} = \begin{bmatrix} x_{0,1} & \dots & x_{0,n_x} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \dots & x_{N,n_x} \end{bmatrix} \quad \text{and} \quad \mathbf{U} = \begin{bmatrix} u_{0,1} & \dots & u_{0,n_u} \\ \vdots & \ddots & \vdots \\ u_{N,1} & \dots & u_{N,n_u} \end{bmatrix}. \quad (3.9)$$

Therefore,

$$\mathbf{X} \in \mathbb{R}^{n \times n_x} \quad \text{and} \quad \mathbf{U} \in \mathbb{R}^{n \times n_u}. \quad (3.10)$$

Altogether, this discretization results in a constrained optimization problem referred to as P^n . This problem can be solved via a nonlinear optimization program. In the standard form shown in (3.8), the problem statement becomes

$$(P^n) \left\{ \begin{array}{ll} \text{Minimize} & J(\mathbf{X}, \mathbf{U}, t_0, t_f) \\ & = E(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) + \mathbf{I}^T \bar{\mathbf{W}} \mathbf{F} \\ \text{Subject to:} & \\ & \bar{\mathbf{D}} \mathbf{X} = \mathbf{f}(\mathbf{X}, \mathbf{U}, t), \\ & \mathbf{e}^L \leq \mathbf{e}(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) \leq \mathbf{e}^U, \\ & \mathbf{H}^L \leq \mathbf{h}(\mathbf{X}, \mathbf{U}, t) \leq \mathbf{H}^U, \end{array} \right. \quad (3.11)$$

where the new vectors, $\mathbf{I}^T$ and $\mathbf{F}$, and new matrices, $\bar{\mathbf{D}}$ and $\bar{\mathbf{W}}$ are as follows. Now, the scalar function F become a vector $\mathbf{F}$ of these functions evaluated at each timestep. For

instance,

$$\mathbf{F} = \begin{bmatrix} F(x_0, u_0, t_0) \\ F(x_1, u_1, t_1) \\ \vdots \\ F(x_f, u_f, t_f) \end{bmatrix}. \quad (3.12)$$

Furthermore, the running cost is transformed into a discretization of the integral, $\mathbf{I}^T \bar{\mathbf{W}} \mathbf{F}$, where

$$\mathbf{I}^T = (1, 1, \dots, 1) \quad (3.13)$$

and the quadrature weights in the physical domain are represented in matrix form as

$$\bar{\mathbf{W}} = \text{diag}(\bar{w}_0, \bar{w}_1, \dots, \bar{w}_N). \quad (3.14)$$

See Section 4.3 for more on quadrature. The dynamics function now returns a matrix such that

$$\mathbf{f}(\mathbf{X}, \mathbf{U}, t) = \begin{bmatrix} f_0(\mathbf{X}, \mathbf{U}, t) & f_1(\mathbf{X}, \mathbf{U}, t) & \cdots & f_{n_x}(\mathbf{X}, \mathbf{U}, t) \end{bmatrix}, \quad (3.15)$$

where f_i is the vector of dynamics for the i th component of $\mathbf{f}$ at all timesteps. Lastly, and most importantly, the states are differentiated via $\bar{\mathbf{D}}\mathbf{X}$ where $\bar{\mathbf{D}}$ is a differentiation matrix for the Lagrange approximation of $\mathbf{x}$. $\mathbf{P}^n$ is a discrete representation of the continuous optimal control problem $\mathbf{P}$. Therefore, integrals, like the running cost, require some form of quadrature or creation of a new state to track running cost.

3.5 Feasibility and Optimality

The proof of optimality and convergence in optimal control is beyond the scope of this thesis. This section, which is derived from [17], covers enough ground for the reader to understand the context of this thesis. The essential takeaway is that solution optimality is based on Pontryagin's principle. In this framework, a well-posed boundary value problem (BVP) is constructed through dualization. The primary result of dualization is that from each state there is a costate defined such that

$$\lambda(t) \in \mathbb{R}^{n_x} \quad (3.16)$$

is the costate vector of similar dimension to $\mathbf{x}$. Each costate accounts for the impact of the corresponding state on the overall cost. Therefore, the optimal control formulation becomes a system of $2n_x$ coupled differential equations defined by the dynamics of the states and costates, i.e.,

$$\begin{bmatrix} \dot{\lambda} \\ \dot{\mathbf{x}} \end{bmatrix}. \quad (3.17)$$

Also, there is an implicit need to define the temporal bounds, t_0 and t_f , of the optimal control problem and, thus, these equations. Therefore, a **well-posed** optimal control problem requires $2n_x + 2$ point conditions. For a time-fixed problem, in which t_0 and t_f are known, only $2n_x$ point conditions are required. However, in a time-free problem the temporal flexibility must be constrained elsewhere in the problem. The unspecified boundary conditions may be imposed via the costates.

Once a problem has been determined to be well-posed, the feasibility of a solution follows in that any trajectory using the system dynamics can achieve the conditions. The optimality of a feasible solution then relies on the Pontryagin's principle. The application of this principle essentially transforms the entire constrained minimization problem of optimal control into a problem of minimizing the Hamiltonian function, defined as

$$H(\lambda, \mathbf{x}, \mathbf{u}, t) := F(\mathbf{x}, \mathbf{u}, t) + \lambda^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t). \quad (3.18)$$

If path constraints, $\mathbf{h}$, are imposed, then the function to minimize is actually the Lagrangian of the Hamiltonian defined as

$$\bar{H}(\lambda, \mathbf{x}, \mathbf{u}, t) := H(\lambda, \mathbf{x}, \mathbf{u}, t) + \boldsymbol{\mu}^T \mathbf{h}(\mathbf{x}, \mathbf{u}, t), \quad (3.19)$$

where $\boldsymbol{\mu}$ is the path covector such that

$$\boldsymbol{\mu}(t) \in \mathbb{R}^{n_h}, \quad (3.20)$$

and n_h is the number of path constraints. For a much better and more detailed discussion on the construction of the Hamiltonian see [17]. For our purposes it is sufficient to now proceed to the fact, without an accompanying proof, that the optimal Hamiltonian, $\mathcal{H}$, is

minimized with respect to control such that

$$\frac{\partial \bar{H}}{\partial u} = 0. \quad (3.21)$$

This is known as the stationarity condition. By the Hamiltonian Evolution Equation,

$$\frac{\partial \mathcal{H}}{\partial t} = \frac{d\mathcal{H}}{dt}, \quad (3.22)$$

it can be shown that, for a Hamiltonian that is not a function of time,

$$\frac{\partial \mathcal{H}}{\partial t} = 0, \quad (3.23)$$

and thus $\mathcal{H}$ is a constant value. For minimum-time problems, where $J = t_f$,

$$\mathcal{H} = -1. \quad (3.24)$$

These constant values for $\mathcal{H}$ can be seen in the solutions of Sections 3.8 and 3.9.

Finally, it is important to note that the most effective implementation of pseudospectral optimal control takes advantage of the details of Pontryagin's principle and dualization. This is the basis of the covector mapping principle shown in Figure 3.1.

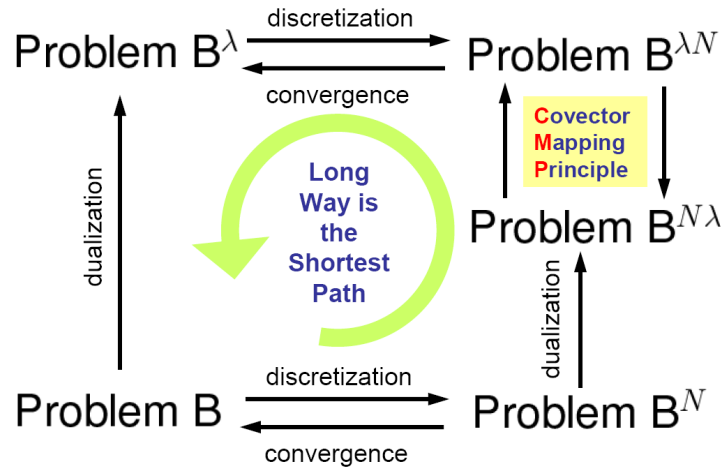


Figure 3.1. Optimal control problem formulation flow, which demonstrates the covector mapping principle. Source: [18].

These principles are implemented in Elissar Global's DIDO optimal control software package. DIDO is used to calculate the Hamiltonian for the basic example problems used in this thesis and described in Sections 3.7–3.9.

3.6 The Nonlinear Program (NLP)

By discretizing the time domain, as shown in Section 3.4, the standard optimal control problem (3.11) can be solved via formulation as a general nonlinear programming problem (NLP). Once the state dynamics function, f , is evaluated at discrete points, t , a differential approximation is required as will be discussed in Chapter 4. Skipping this for the time being, a standard nonlinear solver can then be employed. The typical nonlinear constrained minimization format is specified as

$$\begin{aligned}
 \text{(NLP)} \quad & \left\{ \begin{array}{ll} \text{Minimize} & J(\tilde{\mathbf{X}}) \\ \text{Subject to:} & \\ \text{linear constraints:} & \\ & \tilde{\mathbf{X}}^L \leq \tilde{\mathbf{X}} \leq \tilde{\mathbf{X}}^U, \\ & \mathbf{A}_{eq} \tilde{\mathbf{X}} = \mathbf{B}_{eq} \\ & \mathbf{A} \tilde{\mathbf{X}} \leq \mathbf{B}, \\ \text{and nonlinear constraints:} & \\ & \mathbf{c}_{eq}(\tilde{\mathbf{X}}) = \mathbf{0} \\ & \mathbf{c}(\tilde{\mathbf{X}}) \leq \mathbf{0}, \end{array} \right. \quad (3.25)
 \end{aligned}$$

where $\tilde{\mathbf{X}}$ is the decision variable matrix provided to the nonlinear solver. In this framework, the nonlinear solver stores each discrete point in the state and control trajectories as a decision variable. Likewise, for time-free problems, the initial and final times are given as bounded variables. For this thesis, all optimal control constraints were implemented via the variable bounds

$$\tilde{\mathbf{X}}^L \leq \tilde{\mathbf{X}} \leq \tilde{\mathbf{X}}^U \quad (3.26)$$

and the nonlinear constraints of $\mathbf{c}_{eq}(\tilde{\mathbf{X}})$ and $\mathbf{c}(\tilde{\mathbf{X}})$. No other linear constraints were used in the given results.

3.7 Basic Satellite Problems

Basic example problems were used to test the feasibility and performance of various collocation methods. These problems utilized two-dimensional satellite dynamics in polar coordinates. This section will outline the models, dynamics, and desired states.

3.7.1 Two-dimensional, Low-thrust Satellite Model

The state space ($\mathbf{x}$) is defined by position (r, θ), radial velocity (v_r), and tangential velocity (v_t). In this model, the only control (u) is the direction (β) of an always-on, low-thrust engine that produces a constant acceleration (A),

$$u = \beta. \quad (3.27)$$

See Figure 3.2 for a visual. This assumes that the change in mass from the expelled propellant is negligible relative to the overall spacecraft, which is a reasonable, first-order approximation over short time intervals and/or for engines with high specific impulse.

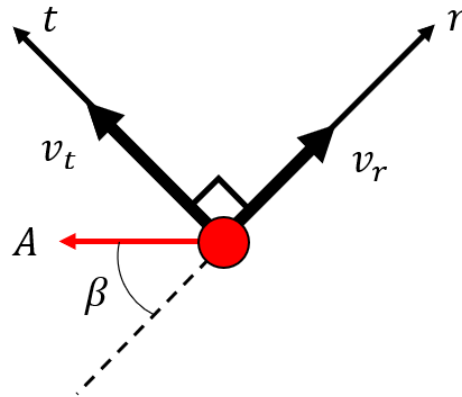


Figure 3.2. The basic satellite model used for BP1 and BP2.

Therefore, the dynamics (f_{BP}) for these basic example problems are

$$\dot{\mathbf{x}} = f_{BP} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) = \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{v}_r \\ \dot{v}_t \end{bmatrix} = \begin{bmatrix} v_r \\ \frac{v_t}{r} \\ \frac{v_t^2}{r} - \frac{\mu}{r^2} + A \sin \beta \\ -\frac{v_r v_t}{r} + A \cos \beta \end{bmatrix}. \quad (3.28)$$

3.7.2 Canonical Units

The relative scales involved in satellite dynamics can often result in ill-conditioned systems of equations before any pseudospectral methods are introduced. In order to scale these problems, canonical units are often used. These units normalize distance and velocity units, DU and VU, to initial orbital values, r_0 and v_0 , respectively. The canonical time unit, TU, is defined by the period of the initial orbit (T_0). In summary, the unit scaling is

$$\text{DU} = r_0 \quad (3.29)$$

$$\text{VU} = v_0 \quad (3.30)$$

$$\text{TU} = \frac{\text{DU}}{\text{VU}}. \quad (3.31)$$

When converted to canonical units, certain satellite dynamics are simplified. The gravitational parameter (μ) becomes unity when expressed in canonical units, i.e.,

$$\bar{\mu} = \mu \left(\frac{\text{TU}^2}{\text{DU}^2} \right) = 1. \quad (3.32)$$

Therefore, when converted to canonical units, the dynamic system (3.28) becomes

$$\dot{\bar{\mathbf{x}}} = \bar{f}_{BP} = f(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t), t) = \begin{bmatrix} \dot{\bar{r}} \\ \dot{\bar{\theta}} \\ \dot{\bar{v}}_r \\ \dot{\bar{v}}_t \end{bmatrix} = \begin{bmatrix} \bar{v}_r \\ \bar{v}_t \\ \frac{\bar{v}_t^2}{\bar{r}} - \frac{1}{\bar{r}^2} + A \sin \beta \\ -\frac{\bar{v}_r \bar{v}_t}{\bar{r}} + A \cos \beta \end{bmatrix}. \quad (3.33)$$

Likewise, an arbitrary time (T) is scaled by TU where

$$\bar{T} = \frac{T}{\text{TU}}. \quad (3.34)$$

Under this transformation, the angular units used in the thrust vector angle, β , and the satellite angular position, θ , are unchanged.

3.7.3 Test Parameters

For the results shown here, the constant acceleration, A , provided by the low-thrust engine is $0.01 \text{ TU}^2/\text{DU}$. This represents a 250mN thrust for a 25kg satellite. For both basic problems,

the initial orbit is circular and at a roughly 660 km altitude. This means a radius of 7025 km, given the volumetric mean radius of Earth of 6371 km. This defines all initial conditions, since velocity will be entirely in the tangential direction throughout the orbit (i.e., $v_{r,0} = 0$) and the angular position can be set arbitrarily (e.g., $\theta_0 = 0$) since no Earth-centered reference is required. In summary,

$$\begin{aligned}\mathbf{x}_0 &= \begin{bmatrix} r_0 & \theta_0 & v_{r,0} & v_{t,0} \end{bmatrix}^T \\ &= \begin{bmatrix} 7025 & 0 & 0 & \sqrt{\frac{3986e5}{7025}} \end{bmatrix}^T\end{aligned}\quad (3.35)$$

Conveniently, these initial conditions become

$$\bar{\mathbf{x}}_0 = \begin{bmatrix} \bar{r}_0 & \bar{\theta}_0 & \bar{v}_{r,0} & \bar{v}_{t,0} \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \quad (3.36)$$

in canonical units. The final conditions will be set differently for each problem.

3.8 Basic Problem 1: Minimum-Time Orbit Transfer

Basic Problem 1 (BP1) is defined by seeking a minimum-time trajectory that doubles the orbital radius, but maintains a final circular orbit for the satellite. The minimum-time is captured in the objective function

$$J(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}) = \bar{t}_f \quad (3.37)$$

and the circular constraint is enforced in the collective values of the final states such that

$$\begin{aligned}\bar{\mathbf{x}}_f &= \begin{bmatrix} \bar{r}_f & \bar{\theta}_f & \bar{v}_{r,f} & \bar{v}_{t,f} \end{bmatrix}^T \\ &= \begin{bmatrix} 2 & \theta^F & 0 & \sqrt{\frac{1}{2}} \end{bmatrix}^T.\end{aligned}$$

For a circular orbit, $v_r = 0$ and $v_t = \sqrt{\frac{\mu}{r}}$. In canonical units, this reduces to $\bar{v}_t = \sqrt{\frac{1}{r}}$. Also, note that the final angular position is not specified: $\theta_f = \theta^F$. Likewise, the final time cannot be set since this problem formulation seeks to minimize orbital transfer time: BP1

is a time-free problem. Altogether, the formulation is

$$(BP1) \left\{ \begin{array}{l} \text{Minimize} \quad J(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}) = \bar{t}_f \\ \text{Subject to:} \\ \quad \dot{\bar{\mathbf{x}}} = \bar{\mathbf{f}}_{BP}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t), t) \\ \quad \bar{\mathbf{x}}_0 = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \\ \quad \bar{\mathbf{x}}_f = \begin{bmatrix} 2 & \theta^F & 0 & \sqrt{\frac{1}{2}} \end{bmatrix}^T. \end{array} \right. \quad (3.38)$$

See Figure 3.3 for a visualization of the satellite trajectory of the solution. The satellite clearly recircularizes the orbit at twice the orbital radius, thus satisfying the final boundary conditions.

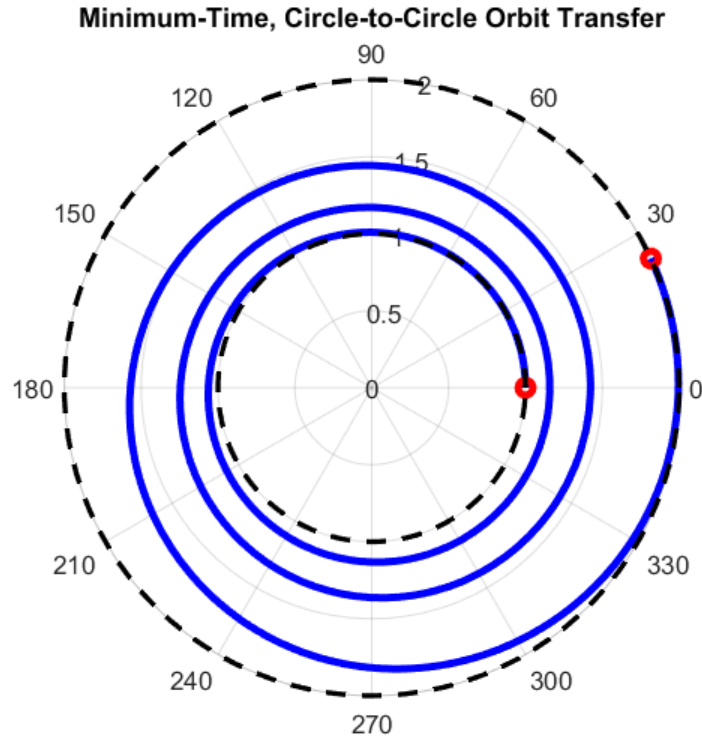


Figure 3.3. BP1 visualization in canonical units of DU and angle in degrees. Initial and final satellite positions are red circles. The initial and final circular orbits are dashed black circles. The solution trajectory is blue.

The optimal controls are displayed in Figure 3.4, which shows a 176-node control solution for BP1 achieved via DIDO, with dots are discrete nodes values and lines created via MATLAB's Piecewise Cubic Hermite Interpolating Polynomial, *pchip*, interpolation. Control oscillations correspond to the changing direction of the satellite's velocity vector as it moves around its continuously changing transfer orbit.

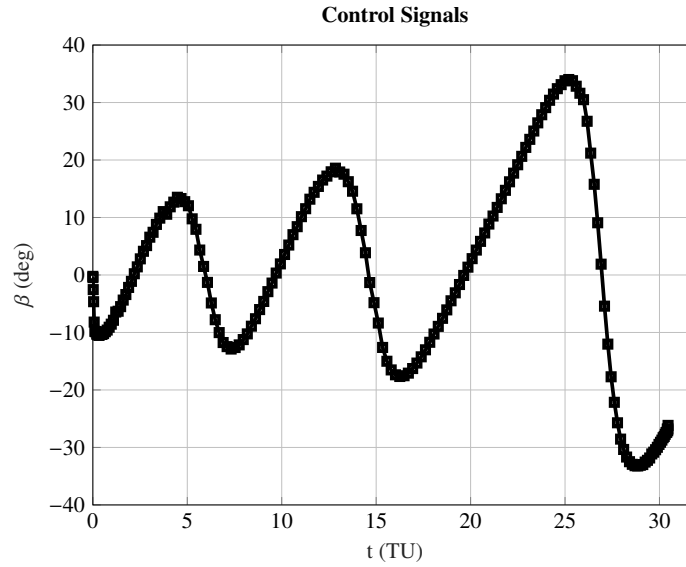


Figure 3.4. Time-optimal controls to BP1 as determined by a 176-node DIDO solution.

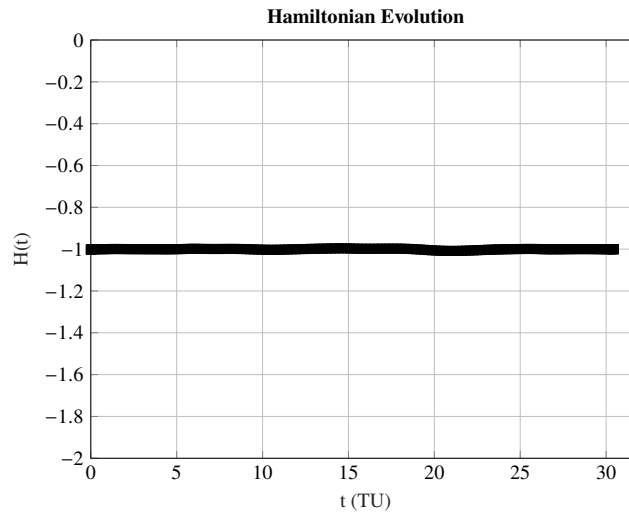


Figure 3.5. Time-optimality as shown by a near-constant Hamiltonian value of -1 as determined by a 176-node DIDO solution.

These controls are a suitable approximation of the time-optimal solution to BP1 as determined by analyzing the Hamiltonian evolution. As discussed in Section 3.5, this time-free solution has a constant trend line for the Hamiltonian at $H(t) = -1$ with only slight numeric deviation. See Figure 3.5.

3.9 Basic Problem 2: A Maximum-Energy Orbit

Basic Problem 2 (BP2) seeks to maximize orbital energy in order to achieve optimal low-thrust output over a fixed time span. The final time is defined as 50 TU, i.e., 50 orbital periods of the initial orbit. Specific orbital energy (ϵ), defined by both the kinetic energy (ϵ_k) and potential energy (ϵ_p), is

$$\begin{aligned}\epsilon &= \epsilon_k + \epsilon_p \\ &= \frac{v^2}{2} - \frac{\mu}{r} \\ &= \frac{v_r^2 + v_t^2}{2} - \frac{\mu}{r}.\end{aligned}\tag{3.39}$$

This orbital energy is maximized by minimizing a negation in the objective function. The problem formulation, in canonical units, is

$$\text{(BP2)} \left\{ \begin{array}{l} \text{Minimize} \quad J(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}) = \frac{1}{\bar{r}_f} - \frac{\bar{v}_{r,f}^2 + \bar{v}_{t,f}^2}{2} \\ \text{Subject to:} \\ \quad \dot{\bar{\mathbf{x}}} = \bar{\mathbf{f}}_{BP}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t), t) \\ \quad \bar{\mathbf{x}}_0 = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \\ \quad \bar{t}_f = 50 \end{array} \right. \tag{3.40}$$

In contrast to BP1, BP2 is a time-fixed problem. Note that the final position is not specified, but the objective function does act to orthogonalize the velocity vectors via the root-sum-square of v_r and v_t . No other explicit constraints are needed to achieve feasible results. Figure 3.6 shows the solution position trajectory for BP2. Note from Figure 3.6 the final orbit is not constrained to being circular as was the case with the BP1 via endpoint conditions.

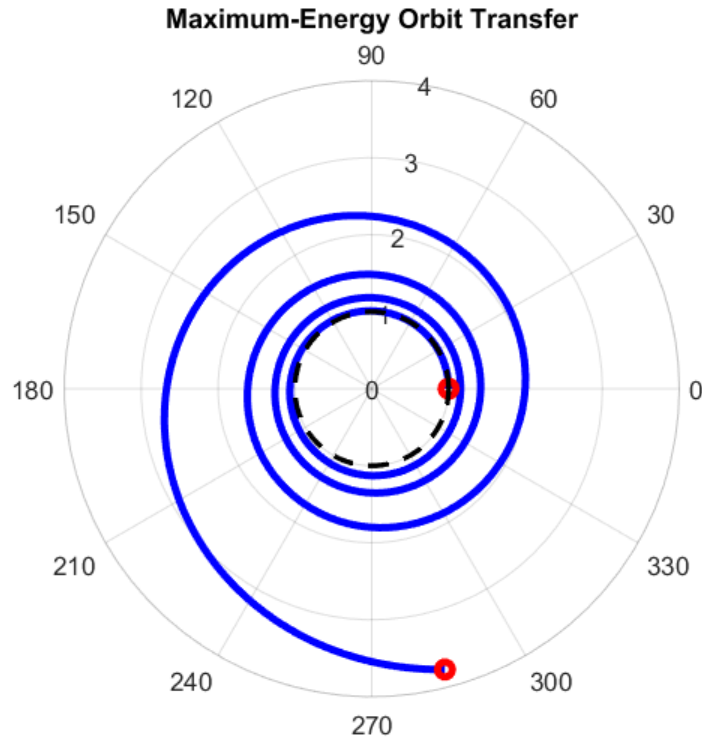


Figure 3.6. BP2 visualization in canonical units of DU and angle in degrees. Initial and final satellite positions are red circles. The initial circular orbits is a dashed black circle. The solution trajectory is blue.

The optimal controls for BP2 are displayed in Figure 3.7, which shows a 176-node control solution achieved via DIDO. These controls have the same expanding oscillations as BP1 due to directional optimality based on an expanding orbital position. For this control solution, the final orbit is not circular so there is no requirement to restrain outward radial velocity. Therefore, the thrust angle is not as symmetric about 0 as it was in BP1.

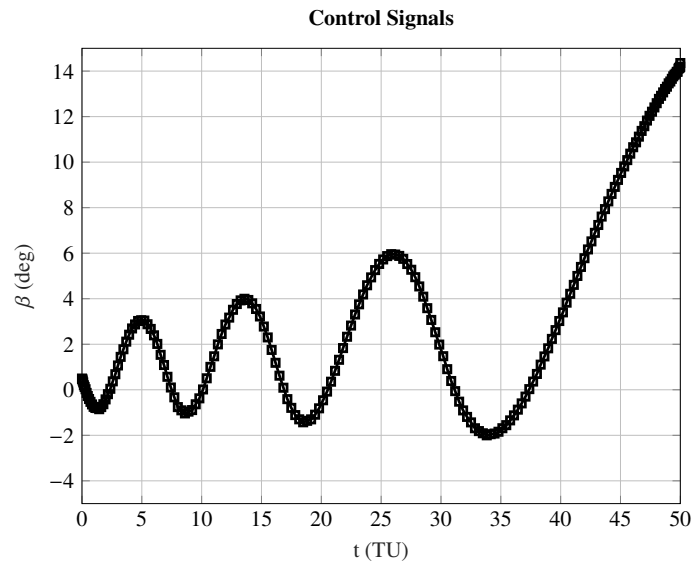


Figure 3.7. Optimal controls to BP2 as determined by a 176-node DIDO solution.

Control optimality for BP2 is shown by a constant Hamiltonian such that $H(t) = 0$, as shown in Figure 3.8.

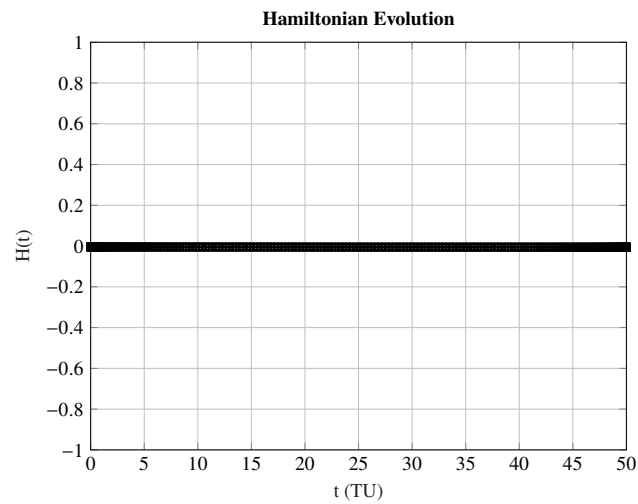


Figure 3.8. Optimality as shown by a near-constant Hamiltonian value of 0 as determined by a 176-node DIDO solution.

3.10 From Concept to Application

In this chapter, we have covered the basics of optimal control in continuous and discrete formulations. As seen in Sections 3.4 and 3.6, the discrete approximations of optimal control problems can be solved numerically if one can approximate the derivatives the states at each timestep. This was shown via $\bar{\mathbf{D}}$ in P^n . This is the pseudospectral differential operator in the physical domain. The construction and application of this operator via a method called Lagrange collocation is the subject of Chapter 4. LCOL solutions to BP1 and BP2 are then compared to the feasible and optimal solution trajectories shown in Section 3.8 and Section 3.9.

CHAPTER 4:

Standard Pseudospectral Optimal Control

The basics of optimal control were covered in Chapter 3. With optimal control understood, this chapter demonstrates how these conceptual models are implemented in practice to achieve numerical solutions. The standard approach uses pseudospectral methods to perform Lagrange collocation (LCOL).

Sections 4.1–4.3 cover the mathematical foundations for pseudospectral optimal control methods. Section 4.1 explains Lagrange interpolation basics and the ramifications of implementation using nodes of orthogonal bases. These orthogonal bases are covered in detail in Section 4.2 and their application to exact quadrature is discussed in Section 4.3. Next, the standard LCOL implementation of pseudospectral optimal control is covered in Section 4.4. LCOL results in Sections 4.6.1–4.6.3, are shown via the satellite control problems defined in Section 3.7.

Ultimately, these LCOL results provide a baseline solution for an implementation framework that is used in the well-conditioned collocation methods discussed in Chapter 6.

4.1 Lagrange Interpolation

The foundation of pseudospectral differentiation and integration is Lagrange interpolation. Lagrange interpolating polynomials, $L(\tau)$, uniquely exist and form a polynomial basis such that at every discrete and distinct node, τ_i , the polynomial assumes an exact specified value, f_i , i.e.,

$$L(\tau_i) = \sum_{j=0}^k f_j l_j(\tau_i) = f_i, \quad (4.1)$$

where the Lagrange basis polynomials are defined as

$$l_i(t) = \frac{\prod_{k=0}^n (\tau - \tau_k)}{\prod_{k=0}^n (\tau_i - \tau_k)} \quad k \neq i. \quad (4.2)$$

Importantly, these polynomials exhibit the behavior that

$$l_i(\tau_j) = \delta_{ij}, \quad (4.3)$$

where δ_{ij} is the Kronecker delta:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \quad (4.4)$$

The result is that for any polynomial of degree at most N , $p_N(\tau)$, the Lagrange polynomials for Gauss-like nodes $\{l_j(\tau)\}_{j=0}^N$ form a well-conditioned basis [19].

4.1.1 The Barycentric Form

A stable and computationally-efficient means of Lagrange interpolation is via the barycentric form, which is defined as

$$L(\tau) = \frac{\sum_{j=0}^N \frac{W_j}{\tau - \tau_j} f(\tau_j)}{\sum_{j=0}^N \frac{W_j}{\tau - \tau_j}}, \quad (4.5)$$

where τ_j are the nodes, f is a continuous function in the domain, and W_j are the barycentric weights for the particular node τ_j in the node set $\tau = (\tau_j)_{j=0}^N$. See Section 4.3 for more on quadrature. In the barycentric form, only $\mathcal{O}(n)$ operations are required to update and evaluate an interpolated value for a given τ [20].

4.1.2 Spectral Accuracy

When Gauss-like quadrature nodes are used in this interpolation scheme, the maximum pointwise error reduces exponentially with an increase in the number of nodes such that

$$\|f(\tau) - p_N(\tau)\|_\infty = \mathcal{O}(q^{-N}) \quad (4.6)$$

for some $q > 1$. This is called spectral accuracy [19], [20]. Both spectral and pseudospectral methods achieve this type of accuracy.

4.1.3 Runge Phenomenon

Along with spectral accuracy, Gauss-like quadrature nodes are suited to avoid interpolation error growth due to oscillations. By using a simple function,

$$f(\tau) = \frac{1}{1 + 25\tau^2}, \quad (4.7)$$

Carl Runge demonstrated that polynomial interpolation accuracy does not always improve by simply increasing the number of nodes. See Figures 4.1 and 4.2 for a demonstration on how endpoint oscillation in the interpolating polynomial can result in runaway interpolation error.

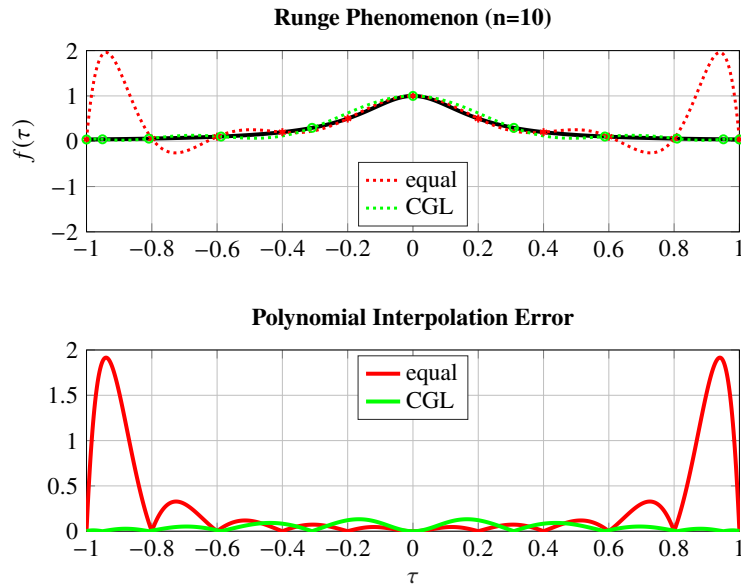


Figure 4.1. The Runge phenomenon using 10-node interpolants. Equally-spaced nodes are contrasted to Chebyshev–Gauss–Lobatto (CGL) nodes, an Gauss-like quadrature node set.

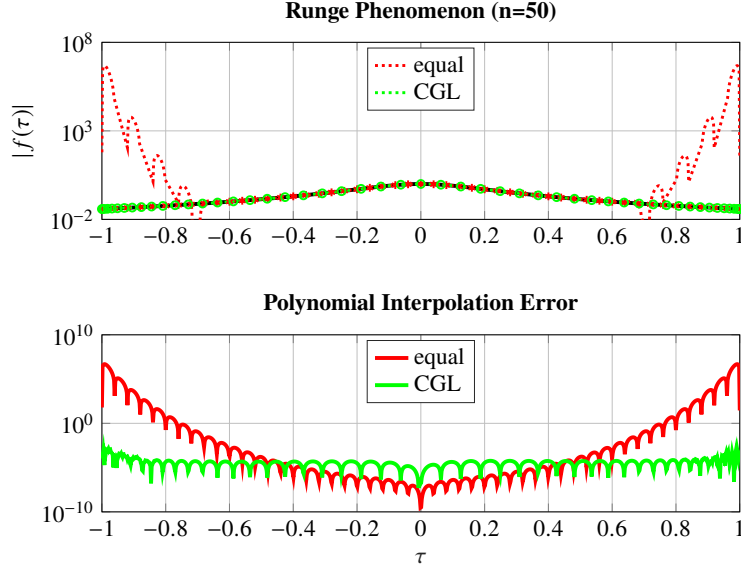


Figure 4.2. The Runge phenomenon for a 50-node interpolation. Absolute function values are used to show scale of oscillation, i.e., $|f(\tau)|$.

Equally-spaced nodes are particularly susceptible to this behavior due to the exponentially large variation in barycentric weights, as discussed in Section 5 of [20]. Note that by simply changing the node set from equidistant to a Gauss-like quadrature—Chebyshev–Gauss–Lobatto (CGL) in this case—the maximum error no longer diverges as n increases.

Due to these properties, Gauss-like quadrature nodes are employed to provide the structure to the differentiation matrix by translating a function into an approximation of its derivative. For further details on the use of Lagrange interpolation in spectral methods, see [20].

4.2 Orthonormal Bases

An orthogonal basis is also desired in order to achieve a linear combination of polynomials to represent the control system trajectories. Once normalized, these orthogonal polynomials are referred to as orthonormal and achieve the preservation of the linear and metric properties of the target space. In particular, these results are rooted in the fact that the inner product of these orthonormal polynomials is the Kronecker delta (4.4). In this thesis, two Jacobi polynomials are described: Chebyshev and Legendre. Further, in this thesis, these polynomials are also evaluated using two node sets: Gauss–Radau (CGR and LGR) and

Gauss–Lobatto (CGL and LGL). Gauss–Radau fix the first node to the the lower bound of the computational domain, i.e., $\tau_0 = -1$. Gauss-Lobatto fix the first and last nodes to the bounds of the domain, i.e., $\tau_0 = -1, \tau_N = 1$. Many references are available for these polynomials and node sets (see, for example, [16], [21]–[23]), but properties and constructions are provided here for concise reference.

4.2.1 Jacobi Polynomials

Jacobi polynomials are the classic orthogonal polynomial, which are solutions to the Jacobi differential equation, and defined as:

$$P_k^{(\alpha, \beta)}(\tau) = \frac{1}{2^k} \sum_{m=0}^k \binom{k+\alpha}{m} \binom{k+\beta}{k-m} (\tau-1)^{k-m} (\tau+1)^m, \quad (4.8)$$

where $k = 0, 1, \dots$ and α, β are real-valued parameters of the Jacobi differential equation [21]. They are orthogonal such that

$$\int_{-1}^1 \frac{P_k^{(\alpha, \beta)}(\tau) P_l^{(\alpha, \beta)}(\tau) \omega(\tau)}{\gamma_k} d\tau = \delta_{kl}, \quad (4.9)$$

where γ_k is a normalization constant and the weighting function

$$\omega(\tau) = (1-\tau)^\alpha (1+\tau)^\beta, \quad (4.10)$$

in the computational domain, and in the physical domain

$$\bar{\omega}(t) = (t_f - t)^\alpha (t - t_0)^\beta. \quad (4.11)$$

These weighting functions can be found in Section 18.3 of [21]. The two Jacobi polynomials are used in this work, Legendre and Chebyshev, have $\alpha = \beta$. The notation then changes from $P_k^{(\alpha, \beta)}$ to $P_k^{(\alpha, \alpha)}$. Legendre polynomials have $\alpha = \beta = 0$ and are denoted simply as $P_k(\tau)$. Chebyshev polynomials of the first kind have $\alpha = \beta = -1/2$ and are denoted as

$$T_k(\tau) = \frac{P_k^{(\alpha, \alpha)}}{\binom{k+\alpha}{k}}. \quad (4.12)$$

4.2.2 Special Values

Before proceeding to the specific polynomials, there are specific values to note. Refer to [16], [21]. First, extremal values are defined such that

$$P_k^{(\alpha, \alpha)}(\pm 1) = (\pm 1)^k \binom{k + \alpha}{k}, \quad k = 0, 1, \dots, N. \quad (4.13)$$

For both Chebyshev and Legendre polynomial, these extremal values further reduce to

$$P_k(\pm 1) = T_k(\pm 1) = (\pm 1)^k, \quad k = 0, 1, \dots, N. \quad (4.14)$$

These special values are shown in Equation 2.5 of [1]. Second, integrals of the Jacobi polynomials will be represented using an integral operator. The first-order integral of the Jacobi polynomial is defined such that

$$\partial_\tau^{-1} P_k^{(\alpha, \alpha)}(\tau) = \int_{-1}^{\tau} P_k^{(\alpha, \alpha)}(y) dy. \quad (4.15)$$

Recursive formulas of these first-order integrals are shown in (4.23) and (4.28). By extension, the second integral is

$$\partial_\tau^{-2} P_k^{(\alpha, \alpha)}(\tau) = \partial_\tau^{-1} (\partial_\tau^{-1} P_k^{(\alpha, \alpha)}(\tau)). \quad (4.16)$$

See Section 3.2 of [1] for the second-order formulas. Finally, note that, by definition, these integrals evaluate to zero at the lower bound of the domain, $\tau = -1$, for all k , i.e.,

$$\partial_\tau^{-1} P_k^{(\alpha, \alpha)}(-1) = 0, \quad (4.17)$$

$$\partial_\tau^{-2} P_k^{(\alpha, \alpha)}(-1) = 0, \quad (4.18)$$

where $k = 0, 1, \dots, N$.

4.2.3 Legendre Polynomials

Legendre polynomials, where $\alpha = \beta = 0$ in (4.8), are found via

$$P_k(\tau) = \frac{1}{2^k} \sum_{m=0}^{\lceil k \rceil/2} (-1)^m \binom{k}{m} \binom{2k-2m}{k} x^{k-2m}, \quad (4.19)$$

with further simplifications as shown in [22]. These polynomials exhibit orthonormal behavior such that

$$\int_{-1}^1 \frac{P_k(\tau)P_l(\tau)\omega(\tau)}{\gamma_k} d\tau = \delta_{kl}, \quad (4.20)$$

where the weighting function (4.10) reduces to

$$\omega(\tau) = 1, \quad (4.21)$$

and γ_k are the normalization constants. For Legendre,

$$\gamma_k = \frac{2}{2k+1} \quad (4.22)$$

where $k = 0, 1, \dots, N$. From Equation 3.13 of [1], the first integral of Legendre polynomials is defined by

$$\begin{aligned} \partial_\tau^{-1} P_0(\tau) &= 1 + \tau, \\ \partial_\tau^{-1} P_k(\tau) &= \frac{1}{2k+1} (P_{k+1}(\tau) - P_{k-1}(\tau)), \quad k \geq 1. \end{aligned} \quad (4.23)$$

4.2.4 Chebyshev Polynomials of the First Kind

These polynomials are defined as the set of solutions to the Chebyshev Differential Equation. For brevity, the Chebyshev polynomials of the first kind are merely called Chebyshev polynomials for the remainder of this thesis. These polynomials are constructed via a simple, explicit formula [21]:

$$T_k(\tau) = \cos(k \arccos(\tau)). \quad (4.24)$$

Chebyshev polynomials also exhibit orthonormal behavior such that

$$\int_{-1}^1 \frac{T_k(\tau)T_l(\tau)\omega(\tau)}{\gamma_k} d\tau = \delta_{kl}, \quad (4.25)$$

where the weighting function (4.10) reduces to

$$\omega(\tau) = \frac{1}{\sqrt{1 - \tau^2}}, \quad (4.26)$$

and γ_k are the normalization constants. For Chebyshev,

$$\begin{aligned} \gamma_0 &= \pi, \\ \gamma_k &= \frac{\pi}{2}, \end{aligned} \quad (4.27)$$

where $k = 0, 1, \dots, N$. From Equation 3.15 of [1], the first integral of Chebyshev polynomials is defined by

$$\begin{aligned} \partial_\tau^{-1} T_0(\tau) &= 1 + \tau, \\ \partial_\tau^{-1} T_1(\tau) &= \frac{\tau^2 - 1}{2}, \\ \partial_\tau^{-1} T_k(\tau) &= \frac{T_{k+1}(\tau)}{2(k+1)} - \frac{T_{k-1}(\tau)}{2(k-1)} - \frac{(-1)^k}{k^2 - 1}, \quad k \geq 2. \end{aligned} \quad (4.28)$$

4.2.5 Polynomial Comparison

Note that Chebyshev polynomials have a computationally-efficient, explicit formula. These polynomials work best at minimizing maximum pointwise error (the L^∞ norm). On the other hand, Legendre polynomials, are designed to minimize the L^2 norm. Legendre polynomials do have a closed-form, but are generally evaluated via a three-term recurrence.

4.3 Exact Quadrature

Ultimately, in order to utilize orthonormal properties, we seek to exactly represent the integral of a polynomial function, $\phi(\tau)$, as a linear combination of discrete function evaluations, $\phi(\tau_j)$, such that

$$\int_{-1}^1 \phi(\tau) \omega(\tau) d\tau = \sum_{j=0}^N \phi(\tau_j) w_j, \quad (4.29)$$

where τ_j are the nodes and w_j are the quadrature weights for a given node set.

4.3.1 Nodes

The Chebyshev nodes at Lobatto (CGL) and Radau (CGR) points are

$$\begin{array}{cc} \text{CGL} & \text{CGR} \\ \tau_j = -\cos\left(j\frac{\pi}{N}\right) & \tau_j = -\cos\left(j\frac{2\pi}{2N+1}\right), \end{array} \quad (4.30)$$

where $j = 0, 1, \dots, N$. The nodes for Legendre can not be found explicitly, but are determined through iterative or approximate schemes. Legendre polynomials ($\phi(\tau) = P_k(\tau)$) have nodes and quadrature weights that are found according to zeros of various functions of Legendre polynomials. For a reference, see Theorem 3.29 on Page 96 of [15]. Specifically, Equations 3.179 and 3.180 in [15] describe these weights for LGR and LGL, respectively.

4.3.2 Weights

The quadrature weights for Chebyshev polynomials ($\phi(\tau) = T_k(\tau)$) are:

$$\begin{array}{cc} \text{CGL} & \text{CGR} \\ w_0 = w_N = \frac{\pi}{2N} & w_0 = \frac{\pi}{2N+1} \\ w_j = \frac{\pi}{N} & w_j = \frac{2\pi}{2N+2}, \end{array} \quad (4.31)$$

where $j = 1, \dots, N-1$ for CGL and $j = 1, \dots, N$ for CGR. For a reference, see Theorem 3.30 on Page 108 of [15].

Note the difference between the Greek *omega*, ω , used for the weighting function and the Latin w for the quadrature weights. Since the weighting function is only used in the continuous case, these will rarely coexist to create confusion. Likewise, with a focus on numerical methods, this work will primarily refer to the quadrature weights.

4.3.3 Degree of Exactness

In general quadrature, exactness can be achieved for N -degree polynomials ($\mathbb{P}_N$) by simply choosing the n coefficients of the n nodes. However, in Gauss quadrature (4.29) there are $2n$ parameters: n nodes (τ_j) and n quadrature weights (w_j). Thus, without node restrictions

Gauss quadrature can achieve degree of exactness of $2n - 1$ ($\phi \in \mathbb{P}_{2N+1}$). This is reduced further if nodes are fixed to locations in the computational domain.

For Gauss–Radau (CGR and LGR), fixing $\tau_0 = -1$ reduces the degree of exactness by 1:

$$\phi \in \mathbb{P}_{2N}.$$

For Gauss–Lobatto (CGL and LGL), the degree is reduced by 2 due to both bounds being fixed: $\tau_0 = -1$ and $\tau_N = 1$. Therefore,

$$\phi \in \mathbb{P}_{2N-1}.$$

4.4 Lagrange Collocation (LCOL)

Lagrange collocation (LCOL) makes use of Lagrange interpolation using Gauss-like quadrature nodes in order to match the derivative of a polynomial.

4.4.1 Pseudospectral Differentiation Matrix

The means of derivative matching is accomplished via a Pseudospectral differentiation matrix (PSDM), $\mathbf{D}$. For optimal control, the PSDM uses Lagrange interpolation at n distinct Gauss-like nodes in the physical time domain. See Sections 4.1. The PSDM for optimal control, $\bar{\mathbf{D}}$, is used as the approximation actual system dynamics, i.e.,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad (4.32)$$

in (3.8) becomes

$$\bar{\mathbf{D}}\mathbf{X} = \mathbf{f}(\mathbf{X}, \mathbf{U}, t). \quad (4.33)$$

Therefore, root-finding algorithms can be utilized to optimize the controls, subject to the NLP constraints and system dynamics, such that

$$\bar{\mathbf{D}}\mathbf{X} - \mathbf{f}(\mathbf{X}, \mathbf{U}, t) \leq \varepsilon \quad (4.34)$$

where ε is a specified constraint error tolerance.

Given an n -dimensional time vector, the differentiation matrix is a square $n \times n$ matrix with indices i and j such that $0 \leq i, j \leq N$. As an example, the CGL differentiation matrix is:

$$d_{ij} = \begin{cases} \frac{\left(\frac{c_i}{c_j}\right) (-1)^{i+j}}{\tau_i - \tau_j} & i \neq j \\ -\frac{\tau_i}{2(1 - \tau_i^2)} & i = j \\ -\frac{2N^2 + 1}{6} & i = j = 0 \\ \frac{2N^2 + 1}{6} & i = j = N \end{cases} \quad (4.35)$$

where d_{ij} is an element of $\mathbf{D}$ at row i and column j , and c_i are constants resulting from differentiation of the underlying Lagrange polynomial and defined as

$$c_i = \begin{cases} 1 & 1 \leq i \leq N - 1 \\ 2 & i = 0, N \end{cases} \quad (4.36)$$

This corresponds to Equation 3.228 on Page 110 of [15]. References abound on differentiation matrices, which can be applied to different node sets as well as constructed in nodal and modal bases. Among others, see [5], [15], [16], [20].

4.4.2 Domain Transformation

Using the domain maps in Section 3.1, the computational domain can be easily transformed to a general, physical domain by use the chain rule for differentiation:

$$\left. \frac{d\phi}{dt} \right|_{t=t_0}^{t=t_f} = \left. \frac{d\phi}{d\tau} \right|_{\tau=\Gamma^{-1}(t_0)}^{\tau=\Gamma^{-1}(t_f)} \left. \frac{d\tau}{dt} \right|_{t=t_0}^{t=t_f} \quad (4.37)$$

where it can be seen from (3.2) that

$$\frac{d\tau}{dt} = \dot{\Gamma}^{-1}(t) = \frac{2}{t_f - t_0} \quad (4.38)$$

is a scalar in the differentiation. In line with (4.38), the computation-domain PSDM, $\mathbf{D}$, can be transformed into a physical-domain PSDM, $\bar{\mathbf{D}}$ using the same scalar. i.e.

$$\bar{\mathbf{D}} = \left(\frac{2}{t_f - t_0} \right) \mathbf{D}. \quad (4.39)$$

4.5 LCOL Implementation

For the constrained minimization NLP, shown in (3.25), the decision variable matrix given to the nonlinear solver is $\tilde{\mathbf{X}}$. For Lagrange collocation this matrix takes the form:

$$\begin{aligned} \tilde{\mathbf{X}} = \tilde{\mathbf{X}}_{LCOL} &= \begin{bmatrix} x_{0,1} & \dots & x_{0,n_x} & u_{0,1} & \dots & u_{0,n_u} & t_0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & t_f \\ x_{N,1} & \dots & x_{N,n_x} & u_{N,1} & \dots & u_{N,n_u} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{X} & \mathbf{U} & \mathbf{t} \end{bmatrix}. \end{aligned} \quad (4.40)$$

Therefore, the nonlinear solver attempts to make changes to each discrete point in the state and control trajectories in order to satisfy the given constraints. One approach to implementing these constraints via (3.25) will be discussed here.

4.5.1 Constraints and Search Space

A known point condition with a known value, c , becomes a fixed constraint by setting both the upper and lower bounds of the corresponding element in $\tilde{\mathbf{X}}$ to c . Thus, if $\tilde{x}_{ij}$ is an element of $\tilde{\mathbf{X}}$ at row i and column j , then

$$\tilde{x}_{ij}^L = \tilde{x}_{ij}^U = c$$

will enforce this value at $\tilde{x}_{ij}$. The initial conditions of BP1 and BP2 (3.36) as well as t_0 and t_f in BP2 are examples of these known values. Only t_0 is constrained in BP1. These fixed constraints combine with other boundary, path, and endpoint constraints imposed by the nonlinear functions or objective function.

Discrete points with unknown values, a given search space for numerical reasons. This search space specifies the lower and upper bounds on the controls and states as to allow for

a feasible solution. However, controls cannot be constrained without impacting the underlying continuous problem. For example, the controlled angle of thrust, β , in BP1 and BP2 was given a reasonable search space of $[-\frac{\pi}{2}, \frac{\pi}{2}]$ via

$$\tilde{x}_{ij}^L = -\frac{\pi}{2}, \quad \tilde{x}_{ij}^U = \frac{\pi}{2}$$

for all i, j where $\tilde{x}_{ij} \in \mathbf{U}$. Therefore, controls are bounded, but must not be constrained by this search space. This type of search space must necessarily be applied to unconstrained time variables in time-free problems like BP1 where $t_f^L \neq t_f^U$. Finally, any derivatives are left completely unbounded, i.e., the search space is $(-\infty, +\infty)$.

4.5.2 The Nonlinear Solver

For this thesis, Artelys' Knitro optimization software package was used via the a MATLAB interface: *knitromatlab* [24]. No problem-specific gradients, Hessians, or Jacobian sparsity patterns were provided to increase performance. By and large, default settings were used in which Knitro used an interior-point solver to converge on a solution to the constrained optimization problem. Step, function, and constraint tolerances were set to 10^{-6} . The maximum number of iterations was set to 10^4 .

4.5.3 Feasibility Analysis

Once a discrete control solution was determined using the nonlinear solver, feasibility was determined by use of polynomial interpolation error. Since the problems have no analytical solution, an approximation of the solution was found via a low-error tolerance Runge–Kutta method. MATLAB's *ode45* was used for this purpose. Relative and absolute tolerances for *ode45* were set to 10^{-13} , slightly above machine epsilon. Internal to *ode45*-propagated state solution the control was interpolated via the use of MATLAB's Piecewise Cubic Hermite Interpolating Polynomial, *pchip*, function. Altogether, this feasibility framework provided sufficient error tolerance control and stability for the problem sets explored. Therefore, error metrics for the collocation schemes could be made relative to this propagated solution.

4.6 Basic Problem Solutions using LCOL

4.6.1 BP1 Results

In this section, LCOL is used to solve Basic Problem 1 (BP1) of Section 3.8. Figure 4.3 shows 128-node LCOL solutions with CGL (L-CGL) and CGR (L-CGR) nodes, which track the optimal control trajectory from Figure 3.4.

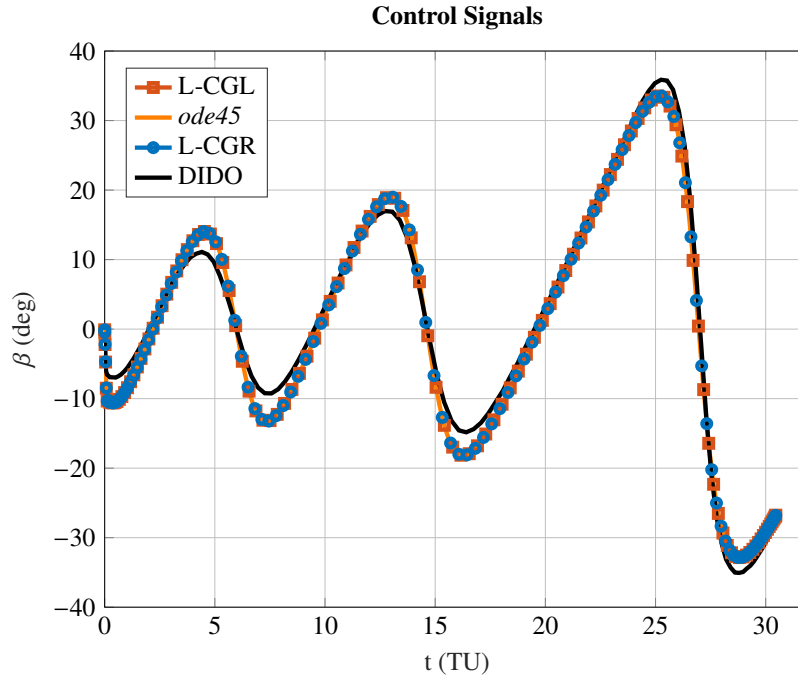


Figure 4.3. Control ($u = \beta$) trajectory for BP1 using 128 nodes. Both CGL and CGR solutions produces similar results, which closely track the DIDO optimal control solution.

The Knitro-based collocation methods (L-CGL and L-CGR) track well with the solution given by DIDO and shown as optimal in Section 3.8. The interpolated control trajectory used in *ode45* is shown for reference. The collocated controls resulted in similar kinematic solutions. For L-CGL and DIDO, solutions to position and velocity are shown in Figure 4.4 and Figure 4.6, respectively. Both methods produce reasonable error metrics for a low-thrust problem that requires three revolutions of the Earth and over 30 times the initial orbital period to achieve the final orbit. See Figure 4.5 and Figure 4.7 for error in position and velocity, respectively.

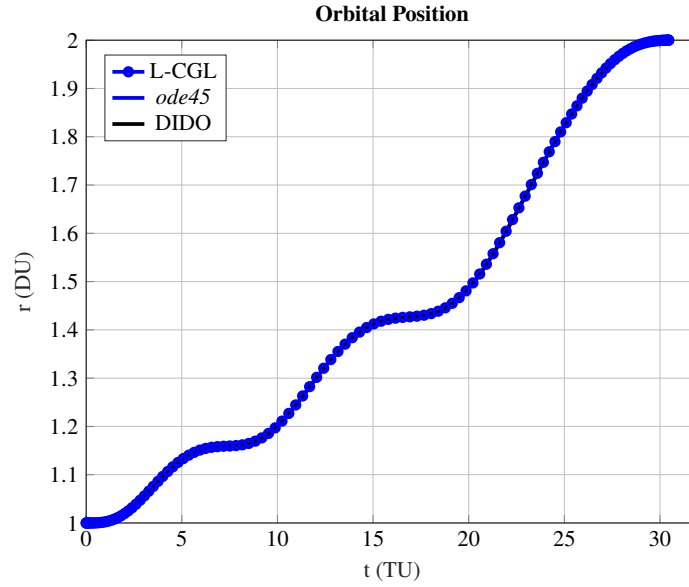


Figure 4.4. Radial position (r) trajectory for BP1. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with *ode45* and a 128-node DIDO solution.

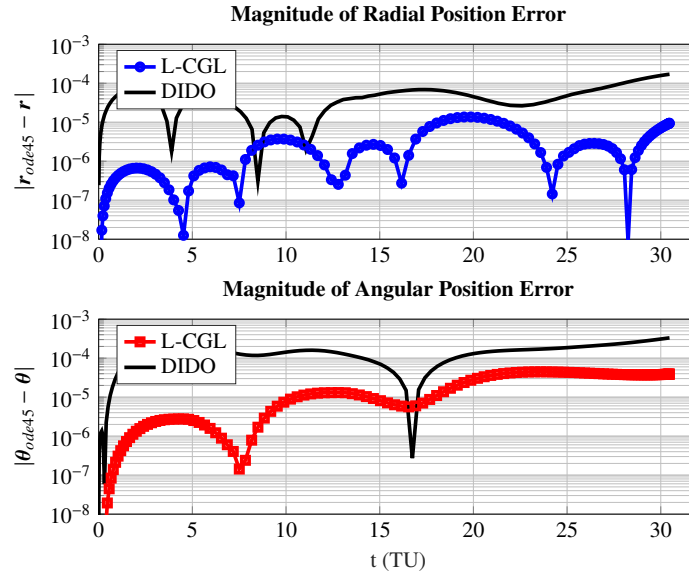


Figure 4.5. Radial (r) and angular (θ) position errors for BP1. L-CGL and DIDO errors are relative to an *ode45*-propagated solutions using the controls given by respective solution.

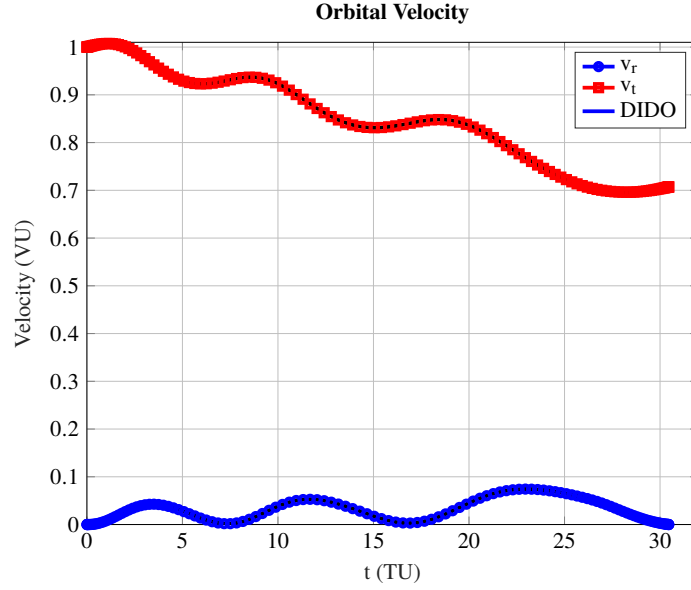


Figure 4.6. Velocity trajectories (v_r and v_t) for BP1. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with *ode45* and a 128-node DIDO solution.

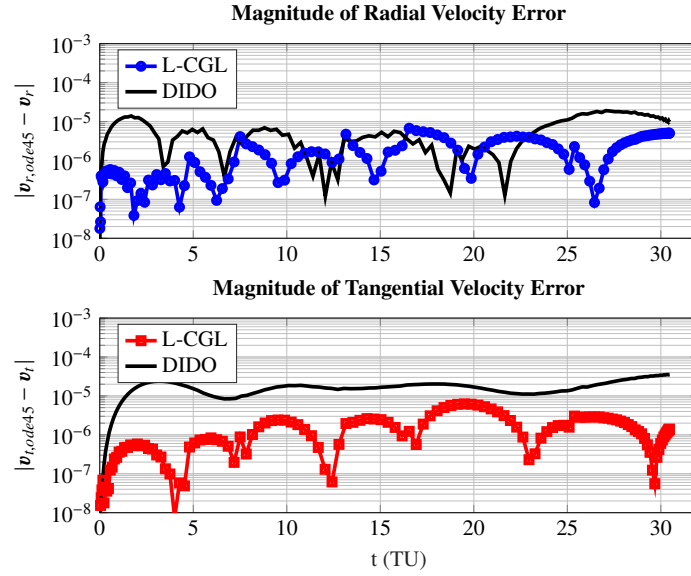


Figure 4.7. Velocity trajectories errors (v_r and v_t) for BP1. L-CGL and DIDO errors are relative to an *ode45*-propagated solutions using the controls given by respective solution.

Note on Figure 4.4 and Figure 4.6, the DIDO solution is covered by the L-CGL solution.

4.6.2 BP1 Performance

The Knitro-based LCOL methods, in general, produced better error metrics and arrived at these solutions faster than DIDO for BP1. This can be expected because there is no commonality in stopping tolerances or conditions. However, the important takeaway is that the baseline Knitro-based LCOL method obtains similar solutions to an industry standard. See Figure 4.8 for maximum pointwise error relative to *ode45*-propagated solutions given the control solution.

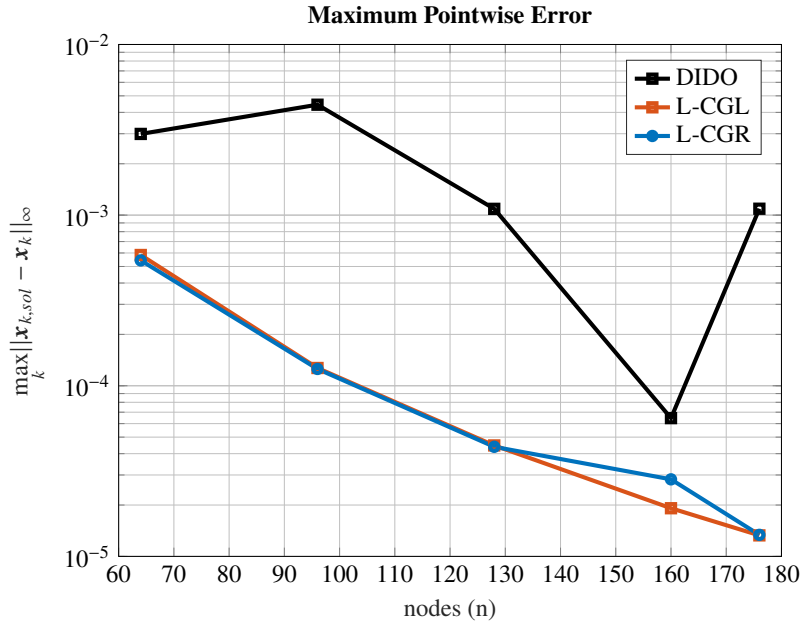


Figure 4.8. Maximum pointwise error of LCOL solutions to BP1 using Knitro in comparison to DIDO solutions. Error is relative to propagated solution using interpolated control solution.

Figure 4.8 shows that the choice of node set—CGL or CGR—has little impact on the solution error. However, as seen in Figure 4.9, L-CGR has an anomaly in the number of function evaluations needed for the 96-node solution. In general, L-CGR achieves faster results, but is not as stable in terms of tracking a trend line. This can be attributed to L-CGR being unable to meet a specified tolerance due to evaluation at $t_N \neq t_f$. This can be overcome by interpolating and imposing final constraints on t_f instead of t_N .

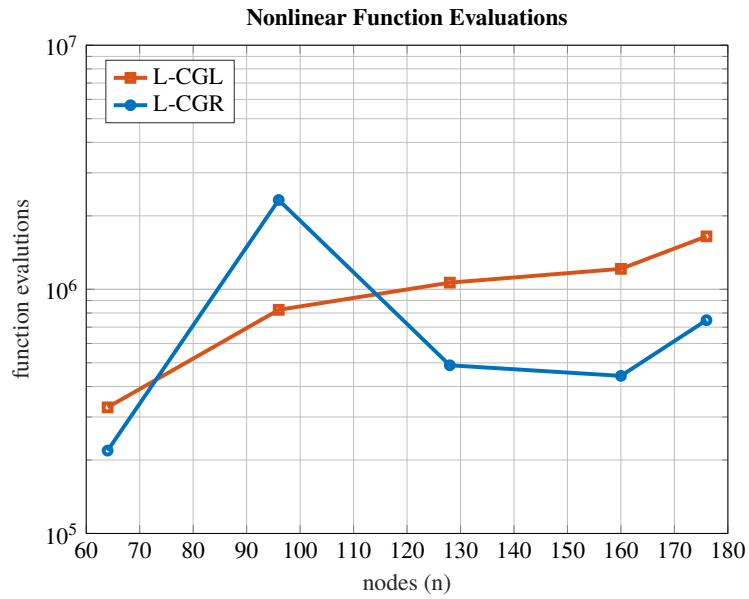


Figure 4.9. Number of nonlinear function evaluations required for LCOL solutions to BP1 using Knitro. L-CGR can be faster, but is less reliable.

Comparing Figure 4.9 to Figure 4.10 demonstrates how solution time was affected by the relative number of function evaluation required to achieve solutions to tolerance.

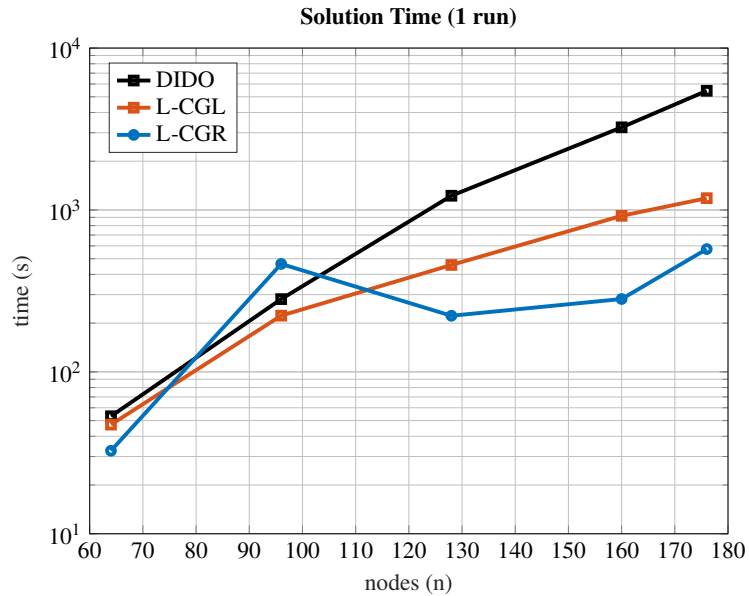


Figure 4.10. BP1 solution times for LCOL using Knitro in comparison to DIDO solutions. CGR solutions show less stability due to lack of +1 node.

These results can vary in subtle ways with implementation and solution times are dependent on the environment, but they indicate that the pseudospectral framework created for this thesis matches industry standard solutions in optimal control.

4.6.3 BP2 Results

Next, we look at LCOL solutions to Basic Problem 2 (BP2) of Section 3.9. Again, the resultant controls match the optimal control trajectory, as seen in Figure 4.11. For BP2, only L-CGL trajectories are shown, but L-CGR results are similar as indicated by Section 4.6.4. For each method, L-CGL and L-CGR, the controls and *pchip*-interpolated values used in the *ode45* solution track so well and appear as one in Figure 4.11.

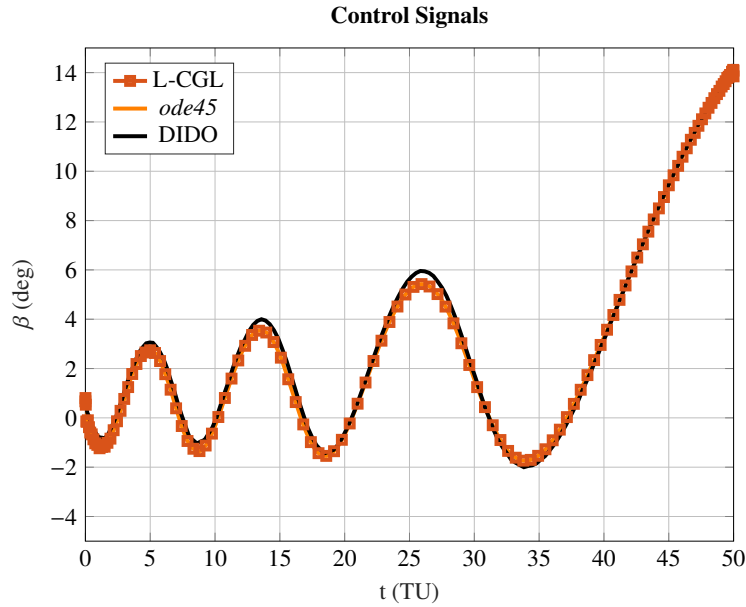


Figure 4.11. Control ($u = \beta$) inputs over fixed time span of 50 TU. DIDO control solution provided for reference.

For this time-fixed problem, the Knitro-based collocation method (L-CGL) and DIDO solutions are smoother and track each other better. This means more similarity in error metrics relative to the *ode45*-propagated solution. For BP2, the collocated controls from L-CGL and DIDO resulted nearly identical profiles for position and velocity, as shown in Figure 4.12 and Figure 4.14. This similarity is all the more apparent in the error metrics

shown in Figure 4.13 and Figure 4.15. For each method, even 80 nodes resolves the necessary dynamics to determine optimal controls despite the longer timespan of 50 TU.

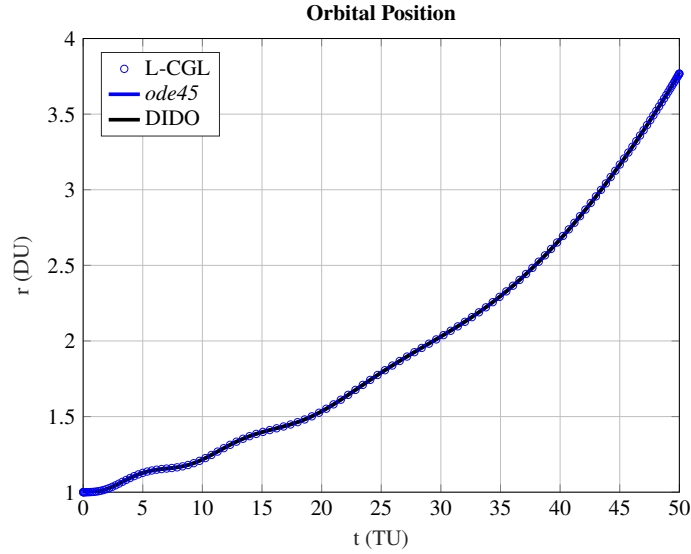


Figure 4.12. Radial position (r) trajectory for BP2. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with *ode45* and a 128-node DIDO solution.

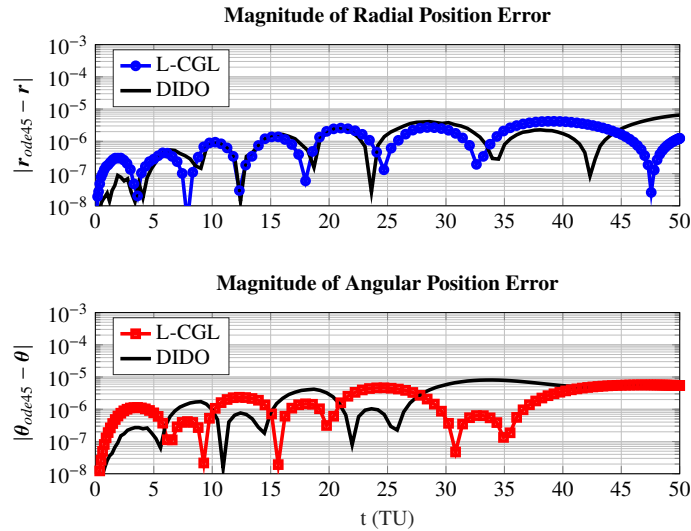


Figure 4.13. Radial (r) and angular (θ) position errors for L-CGL and DIDO solutions. Error is relative to *ode45*-propagated solutions using controls given by respective solution.

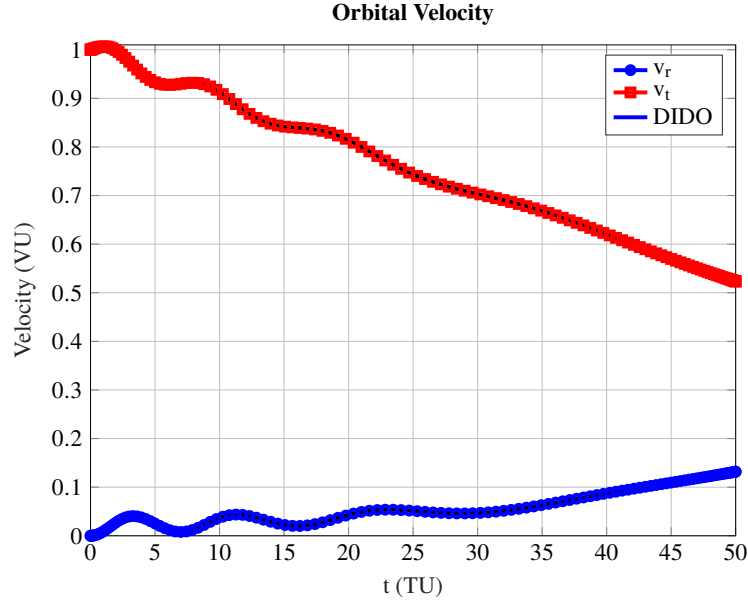


Figure 4.14. Velocity trajectories (v_r and v_t) for BP2. A 128-node L-CGL solution is compared to the position resulting from propagating solution controls with *ode45* and a 128-node DIDO solution.

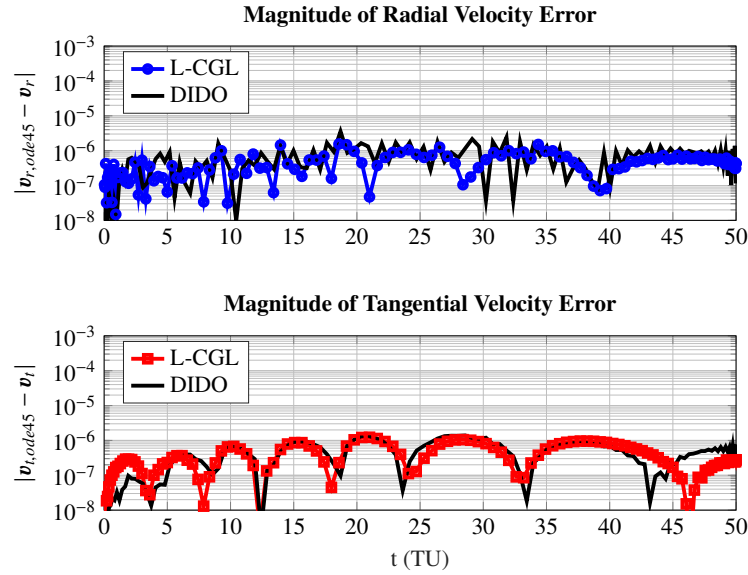


Figure 4.15. Velocity trajectories errors (v_r and v_t) for L-CGL and DIDO solutions. Error is relative to *ode45*-propagated solutions using controls given by respective solution.

4.6.4 BP2 Performance

For BP2, the Knitro-based LCOL methods again produced better error metrics, but took longer to compute their solutions than DIDO. All the variability in performance is a function of imperfect knowledge and implementation of the black-box nonlinear solver. Nevertheless, BP2 results trend appropriately enough to establish confidence in the pseudospectral method. Figure 4.16 shows maximum pointwise error relative to the propagated solutions. Interestingly, Figure 4.17 shows how the nature of BP2 as a time-fixed problem induces more similarity between CGL and CGR in evaluating this problem. Finally, as seen in Figure 4.18, this problem does not outperform the DIDO solution in terms of time. BP2 proved a more difficult problem to solve given the nominal initial guess and tolerance parameters provided for the Knitro-based solutions.

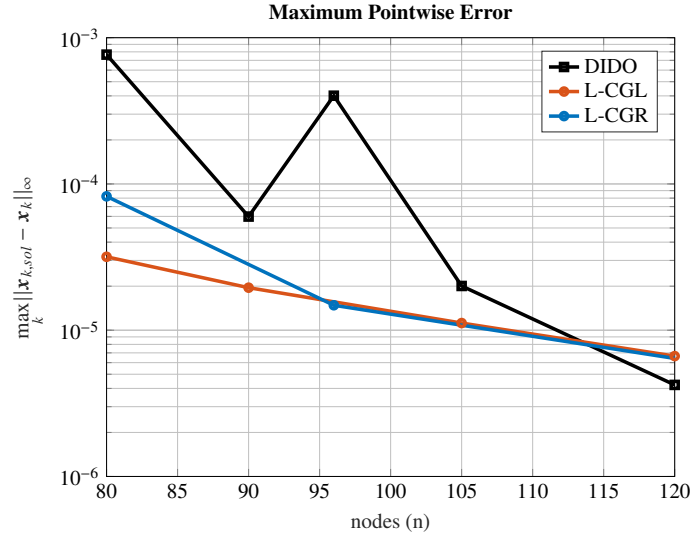


Figure 4.16. Maximum pointwise error of LCOL solutions to BP2 using Knitro in comparison to DIDO solutions. Error is relative to propagated solution using interpolated control solution.

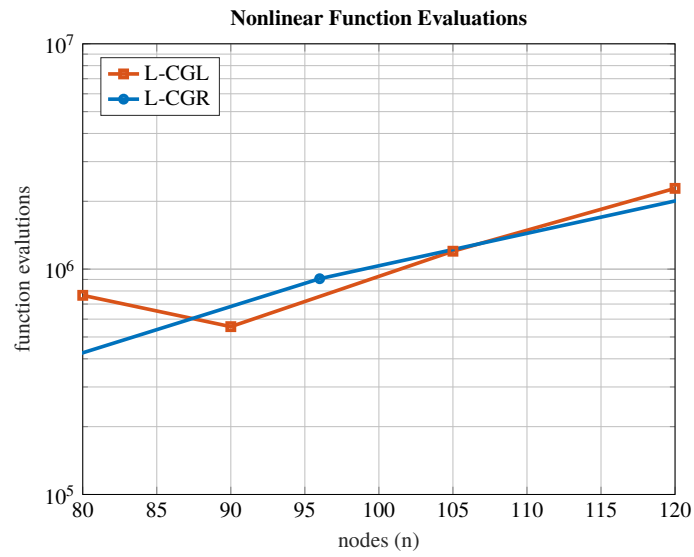


Figure 4.17. Number of nonlinear function evaluations required for LCOL solutions to BP2 using Knitro.

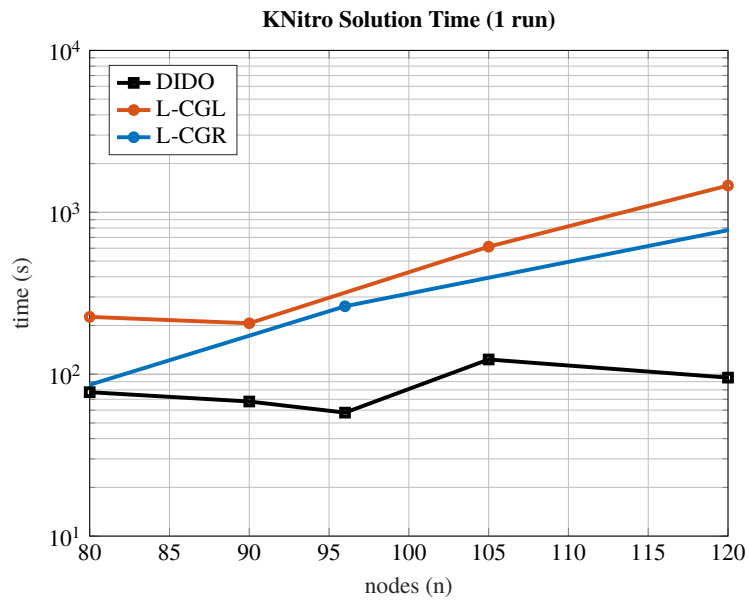


Figure 4.18. BP2 solution times for LCOL using Knitro in comparison to DIDO solutions.

4.7 Motivation for a Change

Lagrange collocation (LCOL) is sufficient for numerous problems that do not require high-dimensionality in terms of the number of nodes. However, LCOL becomes ill-conditioned as the number of nodes increases. This results in poor performance of the nonlinear solver through increased computation time and/or the inability to converge on a solution.

4.7.1 Condition Number Growth

The ill-conditioned nature of $\mathbf{D}$ quickly presents itself in most optimal control problems. However, the condition number of $\mathbf{D}$ in its entirety cannot be measured because it must, by definition, be a singular matrix. Since the derivative of any constant must equal zero, information is lost in differentiation. Thus, the reverse process of integration must require application of this constant of integration. In matrix terms, $\mathbf{D}$ cannot simply be inverted.

Therefore, in order to demonstrate condition number growth, the condition number is computed for a subset of $\mathbf{D}$. Since $\mathbf{D}$ is used in optimal control such that initial and final boundary conditions can be set, the “inner” square portion of $\mathbf{D}$, $\mathbf{D}_{\text{in}}$, will be used such that the top and bottom rows and the left and right columns are removed. Therefore, as discussed in Section 3.2.1, the “inner” indices are

$$\mathbf{i}_{\text{in}} = (1 : N - 1) \quad (4.41)$$

and $\mathbf{D}_{\text{in}}$ consists of all elements of $\mathbf{D}$ where

$$d_{ij} \quad 1 \leq i, j \leq N - 1. \quad (4.42)$$

By using $\mathbf{D}_{\text{in}}$, the condition number growth of this differentiation matrix becomes clear in Figure 4.19. Moreover, Figure 4.19 confirms that the condition number of $\mathbf{D}_{\text{in}}$ is $O(n^2)$ regardless of the Jacobi node set used: CGL, CGR, LGL, or LGR.

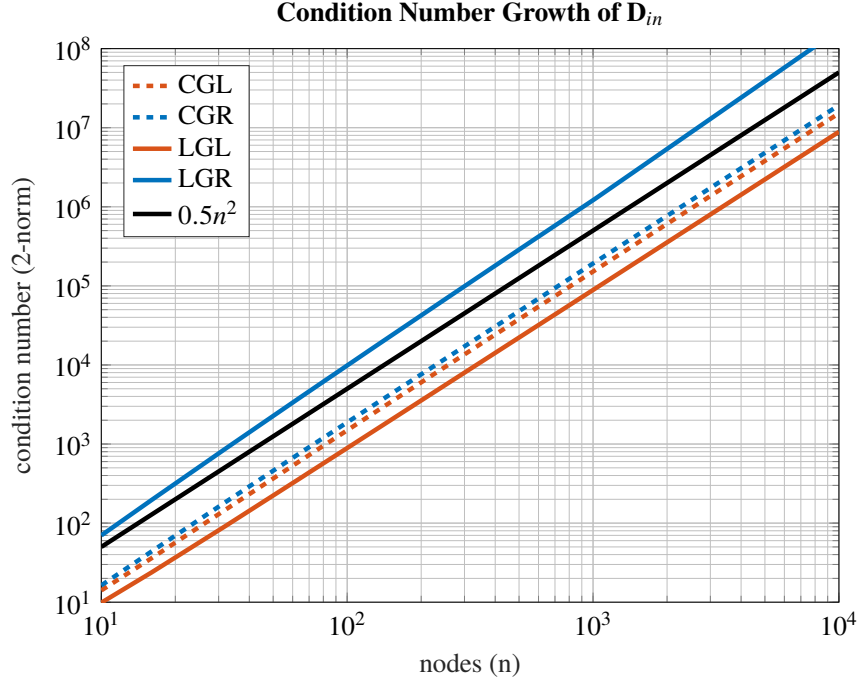


Figure 4.19. Condition number growth of $\mathbf{D}_{in}$, the non-singular portion of the differential operator. By construction, $\mathbf{D}$ is always a singular matrix.

4.7.2 Seeking a Preconditioner

Due to this rapid condition number growth, pseudospectral optimal control seeks a preconditioner matrix, $\mathbf{M}$, in order to create a well-conditioned and stable dynamic system to solve. An ideal preconditioner would cause the system to remain well-conditioned regardless of the number of nodes increased and would completely eliminate the ill-conditioned $\mathbf{D}$. If used as a left-preconditioner, the optimal control LCOL dynamics are transformed such that:

$$\mathbf{MDX} = \mathbf{M}f(\mathbf{X}, \mathbf{U}, t). \quad (4.43)$$

This scheme is referred to as Preconditioned-Lagrange collocation (P-LCOL). If $\mathbf{M}$ is used as a right preconditioner, the optimal control LCOL scheme is broken into two parts:

$$\begin{aligned} \mathbf{X} &= \mathbf{MV} \\ \mathbf{DMV} &= f(\mathbf{X}, \mathbf{U}, t). \end{aligned} \quad (4.44)$$

This scheme is referred to as Birkhoff Collocation (BCOL). This naming convention is derived from [1] and will be seen in Chapter 5.

4.8 A Perfect Preconditioner

Ideally, this preconditioner, $\mathbf{M}$, when left- or right-multiplied with the ill-conditioned $\mathbf{D}$ results in the identity matrix, $\mathbf{I}$, in order to completely remove $\mathbf{D}$ from the collocation scheme. Likewise, since $\mathbf{M}$ remains in both the P-LCOL and BCOL schemes, the resultant linear system needs to be well-conditioned in order to have a stable condition number as the number of nodes increases. As will be seen in Chapter 5, this does not require that $\mathbf{M}$ itself is well-conditioned. For the right preconditioner, that means we seek:

$$1 = \text{cond}(\mathbf{I}) \leq \text{cond}(\mathbf{DM}) < \text{cond}(\mathbf{D}_{\text{in}}). \quad (4.45)$$

Therefore, if

$$\mathbf{DM} = \mathbf{I}, \quad (4.46)$$

the two-part dynamical system (4.44) reduces to:

$$\begin{aligned} \mathbf{X} &= \mathbf{MV} \\ \mathbf{V} &= \mathbf{f}(\mathbf{X}, \mathbf{U}, t). \end{aligned} \quad (4.47)$$

Due to the singularity of $\mathbf{D}$, it will be seen in Chapter 5 that subtle adjustments to (4.46) and (4.47) are required to achieve this perfect preconditioning of $\mathbf{D}$ such that (4.46) applies. Both of these properties are met with the pseudospectral integration matrix (PSIM) discussed in [1]. Chapter 5 details the derivation of the PSIM, extend the available applicability These results will be begin with example single-state, linear systems. After validation in the linear case, Chapter 6 cover the use of this PSIM as the preconditioner, $\mathbf{M}$, used in nonlinear optimal control formulations.

CHAPTER 5:

The Perfect Preconditioner

As seen in the last chapter, Lagrange collocation (LCOL) is powerful, but suffers from the effects of the poorly-conditioned pseudospectral differentiation matrix, $\mathbf{D}$, as the number of nodes increases. The condition number of a p -order differential operator increases as $\mathcal{O}(n^{2p})$ [1], [5]. Due to this behavior, pseudospectral optimal control has been impractical as the problem dimensionality increases.

In order to overcome the limitations of this ill-conditioned operator, this thesis derives, and expands upon, a method of preconditioning outlined in [1], which creates a well-conditioned, linear collocation scheme. Although the work in [1] is restricted to linear systems, their pseudospectral integration matrix, $\mathbf{B}$, does exhibit the perfect preconditioner properties outlined in Section 4.7 and will be utilized for pseudospectral optimal control in Chapter 6.

In that work, the authors demonstrate well-conditioned collocation schemes for linear systems that exhibit condition number stability irregardless of the number of nodes, i.e., $\mathcal{O}(1)$. In order to extend this work to nonlinear systems of equations and, thus, optimal control, it was necessary to understand, verify, and explain the mathematical foundations of this revolutionary approach to pseudospectral collocation.

Due to the nature of this validation, most of the equations and principles investigated in this chapter originate from [1]. Where those authors take the liberty of brevity in order to publish their work, this chapter expands on the foundations, explains the process, and extends their formulation to suit optimal control. Since notation differs in sometimes confusing ways from [1], many equations are reproduced here with the notation discussed in Section 3.2 and compiled in Appendix C.

5.1 The Preconditioner

The perfect preconditioner, $\mathbf{B}$, is formed in order to achieve a well-conditioned preconditioner to the pseudospectral differentiation matrix, $\mathbf{D}$, discussed in Chapter 3. $\mathbf{B}$ can be

defined for different orders of systems of differential equations. Therefore, a subscript p is used to denote the system order and a superscript (k) is used similar to the LCOL derivative notation, i.e., $\mathbf{D}^{(k)} = \mathbf{D}^k$. Therefore, the notation $\mathbf{B}_p^{(k)}$ will be used to avoid ambiguity.

$$\mathbf{B}_p^{(k)} = \mathbf{D}^{(k)} \mathbf{B}_p^{(0)}, \quad (5.1)$$

where $1 \leq k \leq p$. Only the first-order and second-order are shown in this thesis. Refer to Section 4.2 of [1] for higher-order considerations.

The relation shown in (5.1) requires one to determine the initial condition, $\mathbf{B}_p^{(0)}$. More importantly, $\mathbf{B}$ forms a perfect preconditioner such that:

$$\mathbf{B}_p^{(0)} \mathbf{D}^{(p)} = \mathbf{I}_n, \quad (5.2)$$

$$\tilde{\mathbf{D}}^{(p)} \mathbf{B}_p^{(0)} = \mathbf{I}_n, \quad (5.3)$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix and $\tilde{\mathbf{D}}^{(p)}$ is obtained by replacing the rows of $\mathbf{D}^{(p)}$ with Cartesian unit row vectors according to boundary conditions and order, p , of the system of differential equations. Cartesian unit vectors are represented by $\mathbf{e}_{i+1}$ where i refers to the row in $\mathbf{D}^{(p)}$ being replaced. The elements of $\mathbf{e}_{i+1}$, $\mathbf{e}_m$ exhibit the Kronecker delta (4.4) such that

$$e_m = \delta_{im}, \quad 0 \leq i \leq N. \quad (5.4)$$

Thus, for a first-order IVP, the first row of $\tilde{\mathbf{D}}^{(1)}$, is the Cartesian unit vector $\mathbf{e}_1$, i.e.,

$$\tilde{\mathbf{D}}_{0j}^{(1)} = \mathbf{e}_1 = (1, 0, \dots, 0). \quad (5.5)$$

A second-order BVP requires that $\tilde{\mathbf{D}}^{(2)}$ has Cartesian unit vectors in the first and last rows such that

$$\begin{aligned} \tilde{\mathbf{D}}_{0j}^{(2)} &= \mathbf{e}_1 = (1, 0, \dots, 0), \\ \tilde{\mathbf{D}}_{Nj}^{(2)} &= \mathbf{e}_n = (0, \dots, 0, 1). \end{aligned} \quad (5.6)$$

Note that $\tilde{\mathbf{D}}^{(k)}$ is invertible by (5.3). This construction of $\tilde{\mathbf{D}}^{(k)}$ is the basis of the classic approach to solving spectral methods. In a linear system, the row replacements of $\mathbf{D}$ result in implementation of discrete boundary conditions. See the introduction of [25].

5.1.1 Boundary Conditions

As will be seen, $\mathbf{B}_p^{(k)}$ includes explicit boundary conditions. Therefore, the domain is often restricted to the “inner” portion of the domain. This portion of the domain excludes the boundary nodes on which constraints are placed. For a first-order IVP,

$$\mathbf{i}_{\text{in}} = (1 : N), \quad \boldsymbol{\tau}_{\text{in}} = \{\tau_i\}_{i=1}^N, \quad (5.7)$$

and a second-order BVP,

$$\mathbf{i}_{\text{in}} = (1 : N - 1), \quad \boldsymbol{\tau}_{\text{in}} = \{\tau_i\}_{i=1}^{N-1}, \quad (5.8)$$

When only concerned with the “inner” portion of the domain—(5.7) or (5.8)—the perfect preconditioner exhibit the property that

$$\mathbf{D}_{\text{in}}^{(p)} \mathbf{B}_{p,\text{in}}^{(0)} = \mathbf{I}_{n-p}. \quad (5.9)$$

This maps to information in Section 3.1 of [1] with the addition of p in order to distinguish between the order of the system of differential equations.

5.1.2 Domain Transformation

As was the case with the differentiation matrix, a scalar is introduced when transforming the PSIM between the computational and physical domains. Using the domain maps in Section 3.1, the computational-domain PSIM can be transformed to a physical-domain PSIM via the standard change of variables formula for integration:

$$\int_{t_0}^{t_f} \phi(t) dt = \int_{\Gamma(\tau_0)}^{\Gamma(\tau_f)} \phi(t) dt = \int_{\tau_0}^{\tau_f} \phi(\Gamma(\tau)) \dot{\Gamma}(\tau) d\tau, \quad (5.10)$$

where it can be seen from $\Gamma(\tau)$ (see 3.1) that

$$\dot{\Gamma}(\tau) = \frac{t_f - t_0}{2} \quad (5.11)$$

is a constant in the integration and, thus, can be moved outside of the integral. Therefore, the computation-domain PSIM, $\mathbf{B}$, can be transformed into a physical-domain PSIM, $\bar{\mathbf{B}}$,

using the same scalar, i.e.,

$$\overline{\mathbf{B}} = \left(\frac{t_f - t_0}{2} \right) \mathbf{B}. \quad (5.12)$$

Further, it becomes obvious that the integration scalar must be the inverse of the differentiation scalar in order to achieve perfect preconditioning of the physical-domain pseudospectral differentiation matrix, $\overline{\mathbf{D}}$, i.e.,

$$\mathbf{I} = \overline{\mathbf{B}}\overline{\mathbf{D}} = \left(\frac{t_f - t_0}{2} \right) \mathbf{B} \left(\frac{2}{t_f - t_0} \right) \mathbf{D} = \mathbf{B}\mathbf{D}. \quad (5.13)$$

With the properties, applicability, and ramifications of the well-conditioned PSIM understood, the next step is to explore the derivation of the PSIM in terms of the orthogonal polynomials, discussed in Section 4.2, that define its structure.

5.1.3 New Collocation Schemes

By using this perfect preconditioner, standard linear and nonlinear LCOL schemes can be modified to be well-conditioned and therefore stable at higher node counts. In Section 3.3 of [1], the authors describe two different linear schemes: Preconditioned-LCOL (P-LCOL) and Birkhoff collocation (BCOL). P-LCOL uses the PSIM as a left preconditioner and BCOL uses the PSIM as a right preconditioner. For the optimal control research done here, BCOL proved most beneficial. Therefore, presented results reflect this preference.

5.2 An Arbitrary Grid

It is crucial to note at this point, that the PSIM applies to an **arbitrary grid** such that

$$\boldsymbol{\tau} \in \mathbb{R}^n. \quad (5.14)$$

Therefore, node sets discussed in this thesis are simply choices made based on convergence of polynomial interpolation when used to evaluate orthogonal polynomials and/or convenience of construction. In this thesis, the first- and second-order preconditioners are derived for both Chebyshev and Legendre polynomials using either Gauss–Radau (CGR and LGR) or Gauss–Lobatto (CGL and LGL) nodes. The second-order preconditioner, as shown, applies only to the Gauss–Lobatto nodes with Dirichlet boundary conditions.

No other formulations were investigated for the second-order case since optimal control problem formulations are first-order state space BVPs. These formulations best support the investigation of Dirichlet boundary conditions with the explicit definition of the boundaries of the domain ($\tau = \pm 1$). Furthermore, the second-order operator is provided as the best connection to concepts and results discussed in [1].

5.3 First-Order Linear Systems

For optimal control, the first-order PSIM is the most applicable derivation of the concepts in [1]. The state-space formulation with an NLP means that the differential algebraic equations can be represented by a first-order system and boundary conditions can be imposed via the NLP. As a reference to the reader, this section maps explores concepts from Section 4.1 of [1].

The following applies for all Gauss–Lobatto and Gauss–Radau formulations of concern. However, the Jacobi polynomial, $P_k^{(\alpha,\beta)}$, quadrature weights, w_j , normalization constants, γ_k , and the nodes, τ_j , will differ.

5.3.1 Collocation Schemes

In [1], the authors present the following collocation schemes for second-order, linear systems. Those second-order schemes will be covered in Section 5.5.1. However, a first-order, state-space system of differential equations is used for pseudospectral optimal control. Therefore, this chapter begins with first-order systems.

The generic single-state, linear, first-order IVP is:

$$\dot{x}(\tau) + s(\tau)x(\tau) = f(\tau), \quad (5.15)$$

where the initial state is known: $x(-1) = x^0$. The only nodal specification for this linear IVP is the presence of a node at $\tau = -1$. Therefore, both Gauss–Radau and Gauss–Lobatto points are candidates. With a final value problem (FVP), only the final state is known: $x(1) = x^F$.

5.3.2 Lagrange Collocation (LCOL)

For a first-order linear IVP problem, the LCOL scheme can then be represented in matrix form as:

$$(\mathbf{D}_{\text{in}} + \mathbf{\Lambda}_s) \mathbf{x}_{\text{in}} = \mathbf{f}_{\text{in}} - \mathbf{x}_B, \quad (5.16)$$

where the function $s(\tau)$ is now a diagonal matrix such that:

$$\mathbf{\Lambda}_s = \text{diag}(s(\tau_1), \dots, s(\tau_N)). \quad (5.17)$$

This diagonalization of the function $s(\tau)$ is due to the linear nature of the differential equation. In this format, the boundary conditions are moved to the right side of the equation via $\mathbf{x}_B$, which is the vector defined by:

$$\mathbf{x}_B = (x^0 d_{i0})_{i=1}^N, \quad (5.18)$$

where d_{ij} is the entry of $\mathbf{D}$ at row i and column j . For complete clarity, $\mathbf{D} = \mathbf{D}^{(1)}$. Likewise, the “inner” notation discussed in Section 3.2.1, means here that $\mathbf{f}_{\text{in}} = (f(\tau_i))_{i=1}^N$ and $\mathbf{x}_{\text{in}} = (x_N(\tau_i))_{i=1}^N$. Therefore, from (5.16) the matrix being inverted to solve first-order, linear LCOL is

$$\mathbf{A}_{\text{LCOL}} = \mathbf{D}_{\text{in}} + \mathbf{\Lambda}_s \quad (5.19)$$

Using this LCOL basis, the P-LCOL and BCOL schemes are as follows. In these scheme, no ill-conditioned differentiation operator remains in $\mathbf{A}$.

5.3.3 Preconditioned-Lagrange Collocation (P-LCOL)

The left preconditioner scheme uses the identity shown in (5.2) to create the linear system:

$$(\mathbf{I}_N + \mathbf{B}_{1,\text{in}} \mathbf{\Lambda}_s) \mathbf{x}_{\text{in}} = \mathbf{B}_{1,\text{in}} (\mathbf{f}_{\text{in}} - \mathbf{x}_B). \quad (5.20)$$

For P-LCOL, the $\mathbf{A}$ matrix,

$$\mathbf{A}_{\text{PLCOL}} = \mathbf{I}_N + \mathbf{B}_{1,\text{in}} \mathbf{\Lambda}_s, \quad (5.21)$$

becomes well-conditioned regardless of the number of nodes: $\mathcal{O}(1)$.

5.3.4 Birkhoff Collocation (BCOL)

The right preconditioner scheme uses identity (5.9) to create the linear system:

$$(\mathbf{I}_N + \mathbf{\Lambda}_s \mathbf{B}_{1,\text{in}}) \mathbf{v} = \mathbf{f}_{\text{in}} - \mathbf{v}_B, \quad (5.22)$$

where $\mathbf{v}$ is a vector of the state derivatives ($\mathbf{v} = (\dot{x}(\tau_i))_{i=1}^N$) and $\mathbf{v}_B$ is an analog of $\mathbf{x}_B$ such that

$$\mathbf{v}_B = (x^0 s(\tau_i) b_{i0})_{i=1}^N. \quad (5.23)$$

For BCOL,

$$\mathbf{A}_{BCOL} = \mathbf{I}_N + \mathbf{B}_{1,\text{in}} \mathbf{\Lambda}_s, \quad (5.24)$$

also becomes well-conditioned regardless of the number of nodes. However, unlike in P-LCOL, one must recover the states after convergence to a solution for $\mathbf{v}$ via:

$$\mathbf{x}_{\text{in}} = \mathbf{B}_{1,\text{in}} \mathbf{v} + x^0 \mathbf{b}_0, \quad (5.25)$$

where $\mathbf{b}_0 = B_0(\tau_i)_{i=1}^N$.

5.3.5 Desired Properties for $\mathbf{B}_1$

For the rest of this section, $\mathbf{B}_1$ —as defined in Section 5.1—will be shown as $\mathbf{B}$ in order to reduce clutter and confusion. As derived in [1], the first-order PSIM solves an IVP. The properties here follow from that pursuit of $p(\tau) \in \mathbb{P}_N$ such that

$$p(\tau) = x(-1)B_0(\tau) + \sum_{j=1}^N \dot{x}(\tau_j)B_j(\tau), \quad (5.26)$$

where the Birkhoff interpolation basis polynomials $\{B_j\}_{j=0}^N \subseteq \mathbb{P}_N$ satisfy

$$B_0(-1) = 1, \quad \dot{B}_0(\tau_i) = 0, \quad 1 \leq i \leq N, \quad (5.27)$$

$$B_j(-1) = 0, \quad \dot{B}_j(\tau_i) = \delta_{ij}, \quad 1 \leq i, j \leq N, \quad (5.28)$$

such that $p(-1) = x(-1)$, $\dot{p}(\tau_j) = \dot{x}(\tau_j)$. These properties correspond to Section 4.1 in [1]. It is important to note that this choice of constructing an IVP is arbitrary. As will be seen in Chapter 6, a FVP formulation is possible as well.

5.3.6 Birkhoff Polynomials

In order to match the desired characteristics of $\mathbf{B}_1$, a polynomial expansion using an orthonormal basis is used to define the derivative of the Birkhoff interpolation polynomials:

$$\dot{B}_j(\tau) = \sum_{k=0}^{N-1} \alpha_{kj} \frac{P_k^{(\alpha,\alpha)}(\tau)}{\gamma_k}, \quad 1 \leq j \leq N, \quad (5.29)$$

where α_{kj} are the expansion coefficients and γ_k are the normalization constants from Section 4.2: specifically (4.22) and (4.27). Note that $k = 0 \dots N-1$ since one degree of freedom is used to fix the initial value and $p = 1$ (i.e., $\mathbf{B} = \mathbf{B}_1$). The Birkhoff polynomials are then found by integrating (5.29), which results in

$$B_j(\tau) = \sum_{k=0}^{N-1} \alpha_{kj} \frac{\partial^{-1} P_k^{(\alpha,\alpha)}(\tau)}{\gamma_k} + c, \quad (5.30)$$

where the integral, $\partial^{-1} P_k^{(\alpha,\alpha)}(\tau)$, is as defined in Section 4.2.1. The constant of integration, c , is found using the desired property that $B_j(-1) = 0$ and (4.17). Thus, $c = 0$ and the Birkhoff polynomials for first-order integration are

$$B_j(\tau) = \sum_{k=0}^{N-1} \alpha_{kj} \frac{\partial^{-1} P_k^{(\alpha,\alpha)}(\tau)}{\gamma_k}, \quad 1 \leq j \leq N. \quad (5.31)$$

Note that the definition for these Birkhoff basis polynomials is independent of the node set used for evaluation. Gauss, Gauss–Radau, and Gauss–Lobatto nodes are all valid and, as will be seen, result in exact quadrature.

5.3.7 Expansion Coefficients

To solve for the expansion coefficients, α_{kj} , one starts with the expansion of the derivatives of the Birkhoff polynomials, (5.29). By using the orthonormal properties of the basis, (4.9), many terms can be eliminated in the summation of (5.29). To achieve this orthogonality, both sides of (5.29) will be multiplied by the weighted polynomial basis functions (i.e., $P_l^{(\alpha,\alpha)}(\tau)\omega(\tau)$) and integrating over the computational domain (i.e., $\int_{-1}^1 d\tau$) in order to take the inner product. For simplicity, this will be shown with Chebyshev polynomials, but

applies to Legendre as well. The reduction is as follows:

$$\begin{aligned}
\int_{-1}^1 \dot{B}_j(\tau) T_l(\tau) \omega(\tau) d\tau &= \int_{-1}^1 \left(\sum_{k=0}^{N-1} \alpha_{kj} \frac{T_k(\tau)}{\gamma_k} \right) T_l(\tau) \omega(\tau) d\tau \\
&= \sum_{k=0}^{N-1} \left(\alpha_{kj} \left(\int_{-1}^1 \frac{T_k(\tau) T_l(\tau) \omega(\tau)}{\gamma_k} d\tau \right) \right) \\
&= \sum_{k=0}^{N-1} \alpha_{kj} \delta_{kl}.
\end{aligned} \tag{5.32}$$

Therefore, for $l = 0, \dots, N$,

$$\sum_{k=0}^{N-1} \alpha_{kj} \delta_{kl} = \int_{-1}^1 \dot{B}_j(\tau) T_l(\tau) \omega(\tau) d\tau. \tag{5.33}$$

By the Kronecker delta, only the terms where $l = k$ in (5.33) remain. Thus, (5.33) reduces to a continuous solution for the expansion coefficients such that

$$\alpha_{kj} = \int_{-1}^1 \dot{B}_j(\tau) T_k(\tau) \omega(\tau) d\tau, \tag{5.34}$$

where $1 \leq j \leq N$ and $0 \leq k \leq N-1$. The continuous integral in (5.34) can then be represented exactly via quadrature, as shown in (4.29) where $\phi(\tau) = \dot{B}_j(\tau) T_k(\tau)$. Since $\dot{B}_j(\tau) \in \mathbb{P}_{N-1}$ and $T_k(\tau)$ is at most $\mathbb{P}_{N-1}$, the function being integrated $\phi_k(\tau) \in \mathbb{P}_{2N-2}$. Therefore, we achieve exact quadrature for Gauss, Gauss–Radau, and Gauss–Lobatto nodes. See Section 4.3. Note two additional things for the quadrature:

1. $\dot{B}_j(-1)$ is unspecified and needs to be determined and
2. $\dot{B}_j(\tau_i) = \delta_{ij} \forall j \neq 0$ resulting in only the $T_k(\tau_j) w_j$ term $\forall j \neq 0$.

With these observations, the quadrature of (5.34) becomes a discrete expression of expansion coefficients where

$$\begin{aligned}
\alpha_{kj} &= \sum_{i=0}^N \dot{B}_j(\tau_i) T_k(\tau_i) w_i \\
&= \dot{B}_j(-1) T_k(-1) w_0 + T_k(\tau_j) w_j \\
&= [(-1)^k \dot{B}_j(-1)] w_0 + T_k(\tau_j) w_j.
\end{aligned} \tag{5.35}$$

To solve for the unknown $\dot{B}_j(-1)$, we use the fact that

$$0 = \int_{-1}^1 \dot{B}_j(\tau) T_N(\tau) \omega(\tau) d\tau. \tag{5.36}$$

This is true because the remaining, unused polynomial, $T_N(\tau)$, is orthogonal to all polynomials used in the expansion of $\dot{B}_j(\tau)$, i.e., $T_0(\tau) \dots T_{N-1}(\tau)$, so

$$\begin{aligned}
0 &= [(-1)^N \dot{B}_j(-1)] w_0 + T_N(\tau_j) w_j, \\
\dot{B}_j(-1) &= -(-1)^N \left(\frac{w_j}{w_0} \right) T_N(\tau_j).
\end{aligned} \tag{5.37}$$

This can be seen as evaluating (5.35) at $k = N$. By inserting this solution to $\dot{B}_j(-1)$ into (5.35) one arrives at the expression for the first-order expansion coefficients:

$$\alpha_{kj} = w_j (T_k(\tau_j) - (-1)^{N+k} T_N(\tau_j)), \tag{5.38}$$

where $1 \leq j \leq N$ and $0 \leq k \leq N - 1$.

5.3.8 Domain Transformation

The derivations shown in Section 5.3.6 and 5.3.7 are applicable to the computational domain. With slight adjustment to include the integration scalar from (5.11), the expansion coefficients equation, (5.34), for the physical domain becomes:

$$\bar{\alpha}_{kj} = \left(\frac{t_f - t_0}{2} \right) \int_{-1}^1 \dot{B}_j(\tau) T_k(\tau) \omega(\tau) d\tau. \tag{5.39}$$

This constant term in each expansion coefficient results in a common constant term in (5.31) so that the physical-domain version of the Birkhoff polynomials becomes:

$$\bar{B}_j(\tau) = \left(\frac{t_f - t_0}{2} \right) \sum_{k=0}^{N-1} \alpha_{kj} \frac{\partial^{-1} P_k^{(\alpha, \alpha)}(\tau)}{\gamma_k}, \quad 1 \leq j \leq N. \quad (5.40)$$

This validates the properties discussed in (5.12).

5.4 First-Order Validation

Given these two methods of constructing the PSIM preconditioner, the next step was to validate this understanding via implementation. In [1], the authors use the following example IVP:

$$\dot{x}(\tau) - \sin(\tau)x(\tau) = 20 \sin(500\tau^2). \quad (5.41)$$

This differential equation has the following solution:

$$x_{sol}(\tau) = 20 \exp(-\cos(\tau)) \int_{-1}^{\tau} \exp(\cos(y)) \sin(500y^2) dy. \quad (5.42)$$

For LCOL, a CGR differentiation matrix was implemented via Equation 3.229 on Page 100 of [15]. Along with having a common author with [1], this text provides a quality reference for many variations of spectral methods.

5.4.1 Results Validation

The authors of [1] claim to use MATLAB's *bicgstab* for comparison of these two methods. This thesis attempted to only use *bicgstab* to validate their results. However, there are some important implementation details to note. (1) Since solution tolerance was left unspecified in [1], an acceptable value above machine epsilon (*eps*) is used for *bicgstab*: 10^{-15} . (2) A maximum iteration count of 1000 is allowed. (3) Most importantly, *bicgstab* does not converge for most LCOL solutions. This requires the use of *linsolve*. Results in line with [1] are achieved via this implementation.

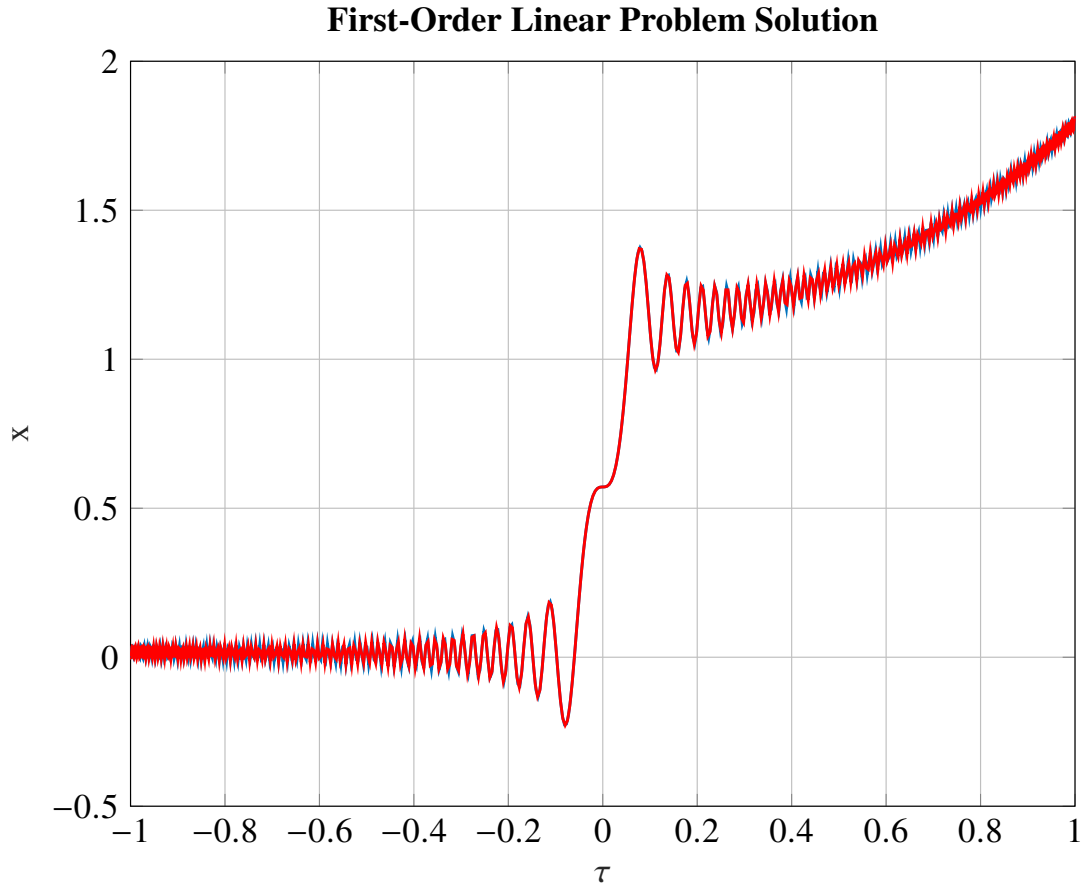


Figure 5.1. Exact solution to example first-order linear IVP (5.42) in the computational domain with a Birkhoff Collocation (BCOL) solution superimposed.

Figure 5.1 demonstrates a BCOL solution tracking the exact solution (5.42) to nearly machine precision. Note the highly oscillatory portions of the function near the ends of the domain.

Maximum pointwise error followed the same curve so as to show a slight deviation in results between the methods at higher nodes (beyond 600 in this case). This is shown in Figure 5.2, which mirrors Figure 4 of [1].

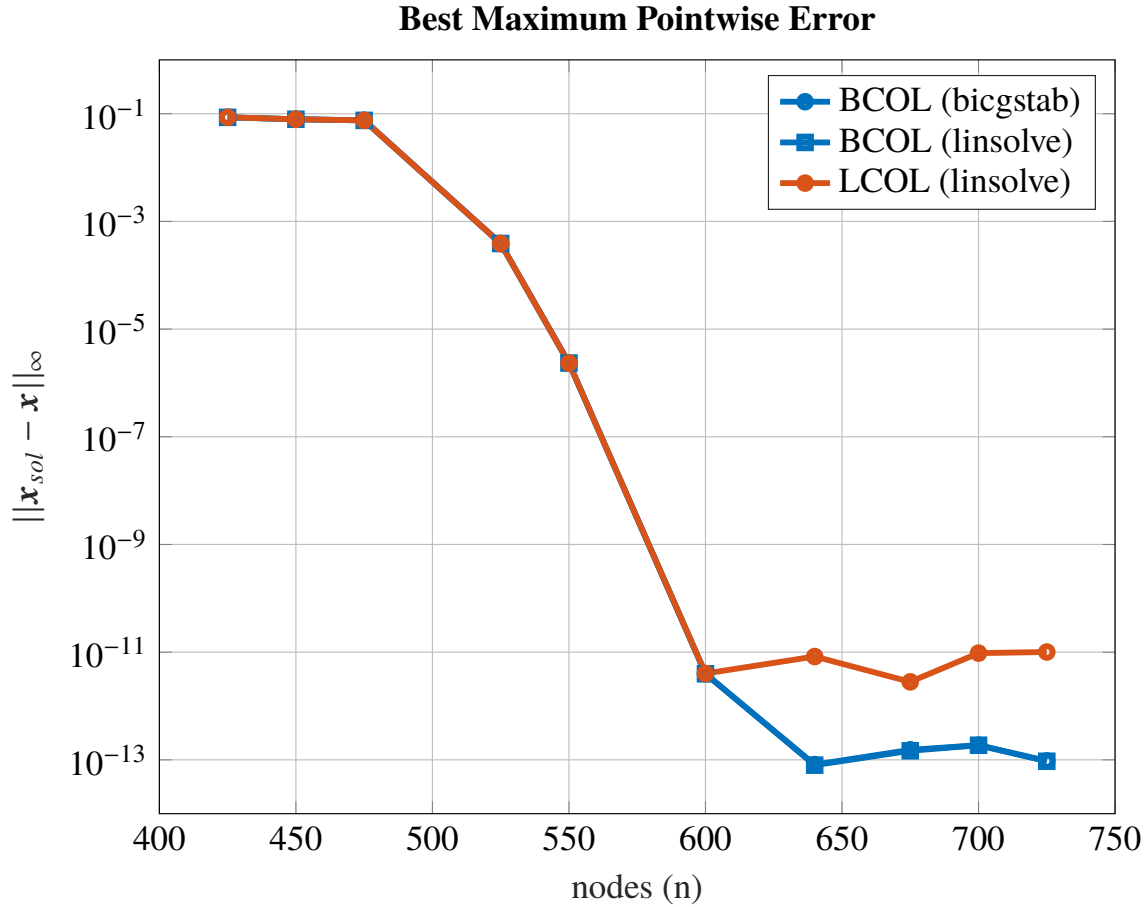


Figure 5.2. First-order IVP maximum absolute pointwise error. BCOL errors achieved with iterative *bicgstab* solver and *linsolve* are virtually the same. LCOL required the use of *linsolve*.

It is important to note that due to condition number growth, LCOL causes problems for *bicgstab* in achieving a solution that converges to tolerance. Even when tolerance is lowered to 10^{-10} , no solution is achieved. *bicgstab* returns an error flag indicating that “one of the scalar quantities calculated during *bicgstab* became too small or too large to continue computing”. Therefore, LCOL results shown in Figure 5.2 require the use of MATLAB’s linear solver, *linsolve*.

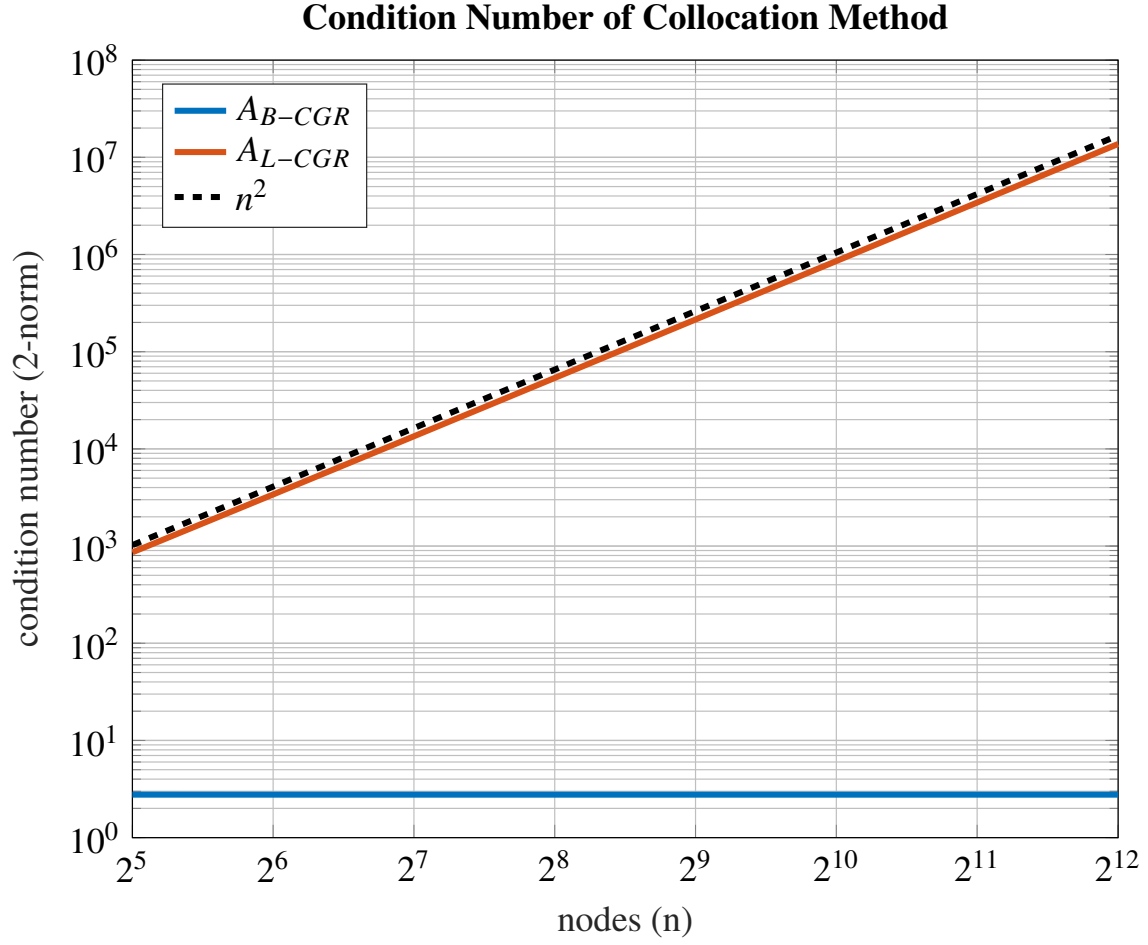


Figure 5.3. First-order IVP condition number behavior disparity between collocation methods. See Section 5.3.1 for definitions of A_{B-CGR} and A_{L-CGR} . Note that BCOL is $O(1)$ and LCOL is $O(n^2)$.

The condition numbers matched exactly to results published in [1]. See Section 5.3.1 for the relevant matrices. These can be seen in Figure 5.3 and in Table 5.1. These results are identical to those in Table 3 of [1]. Furthermore, the baseline of n^2 added to Figure 5.3 to demonstrate that the firstorder LCOL condition number matches expectation: $O(n^2)$.

Table 5.1. Validation of exact same condition numbers as in [1] with unpublished additional significant figures.

N	BCOL	LCOL
128	2.7684	1.3495e4
256	2.7684	5.3773e4
512	2.7684	2.1467e5
1024	2.7684	8.5786e5

BCOL takes minimal *bicgstab* iterations to converge on a solution, as seen in Figure 5.4. LCOL, on the other hand, exhibits a high iteration count before *bicgstab* returns the previously discussed ill-conditioned “scalar quantity” error.

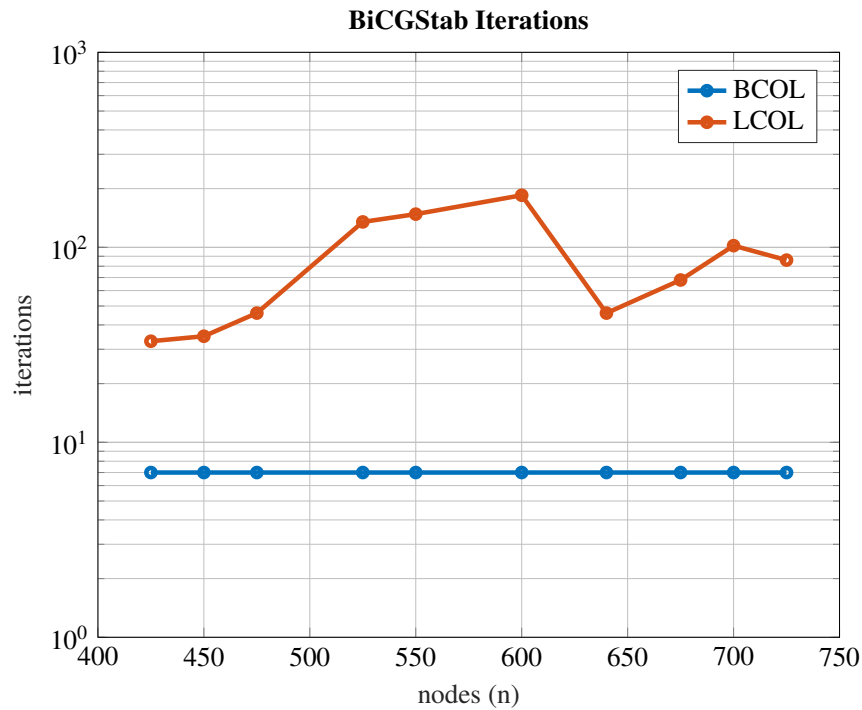


Figure 5.4. First-order IVP iteration count for *bicgstab*. All LCOL iterations do not converge.

Iteration comparison, like in Figure 5.4, would be the preferably metric for method performance. Unfortunately, LCOL cannot be solved via *bicgstab*. Since *linsolve* is a direct solver, iteration comparison is impossible. Thus, average solution time (over 100 runs) is

used to better visualization the impact on performance. The solution time advantage of BCOL is only slightly better than LCOL when using *linsolve*, as seen in Figure 5.5.

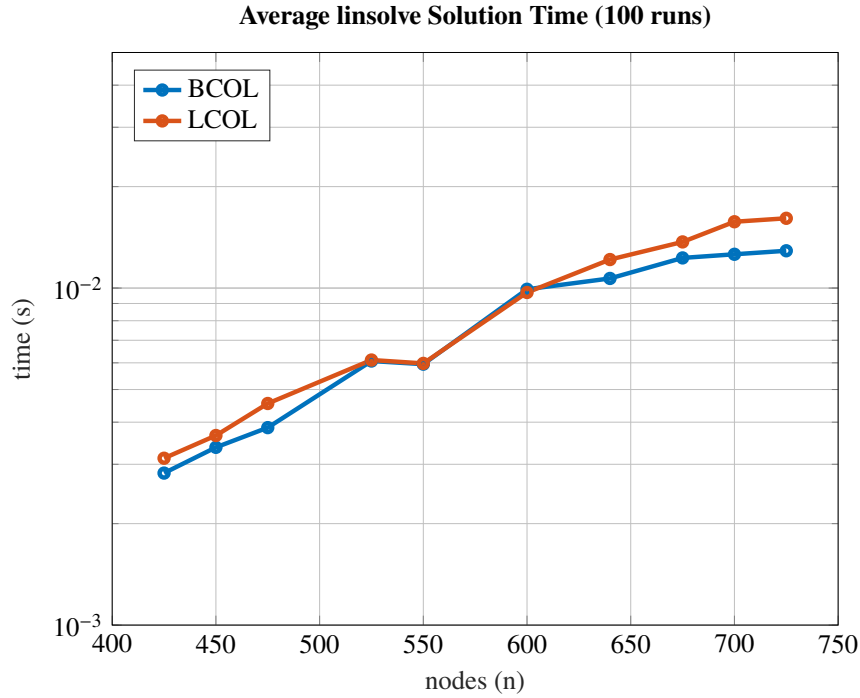


Figure 5.5. First-order IVP solution time for BCOL and LCOL using MATLAB's *linsolve* direct solver. BCOL achieved the same results, but faster with *bicgstab*.

However, BCOL achieves the same errors when using *bicgstab* due to the better conditioning. This results in the solution time advantage shown in Figure 5.6, which has BCOL solution times using *bicgstab* and LCOL using *linsolve*. This metric is very implementation dependent, but it is indicative of the performance gap between LCOL and BCOL in solving this rather simple linear system. As will be seen, this gap grows with the increase in the number of nodes used and with the increase in system complexity.

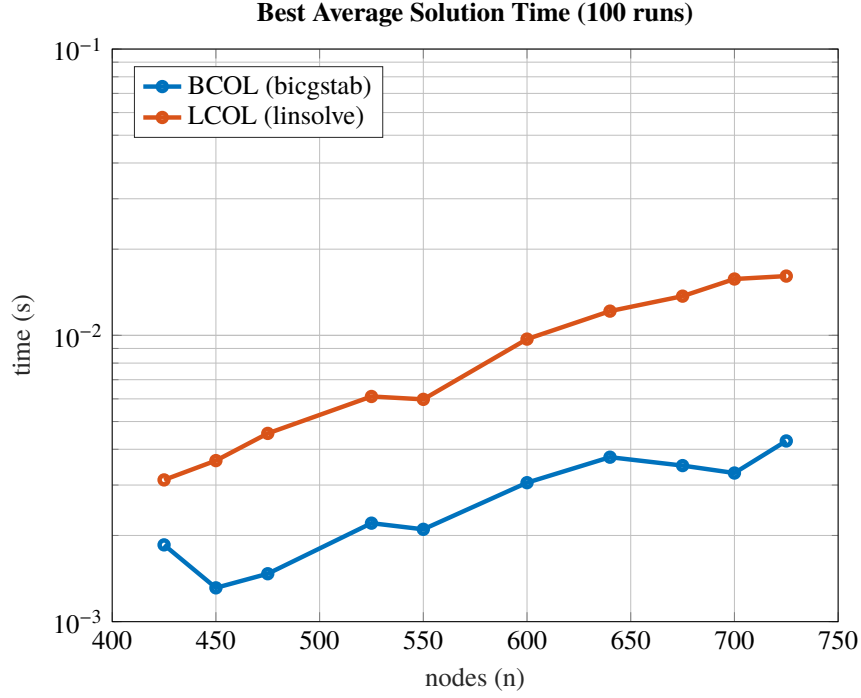


Figure 5.6. First-order IVP solution time. BCOL requires less time using iterative *bicgstab* than LCOL, which required MATLAB's *linsolve* direct solver.

5.4.2 Domain Transformation

Unlike in [1], this example problem was then extended to validate domain transformations and, in particular, the fact that only a constant scalar is required: i.e., (5.11) in Section 5.1.2. In the new domain, $[0, 2]$, many more nodes were required to capture the increased oscillatory behavior of the function. This allows for a better comparison of LCOL and BCOL at the order of magnitude in the number of nodes that is desired for high resolution problems. See Figure 5.7 and Figure 5.8 for a visual of this new domain.

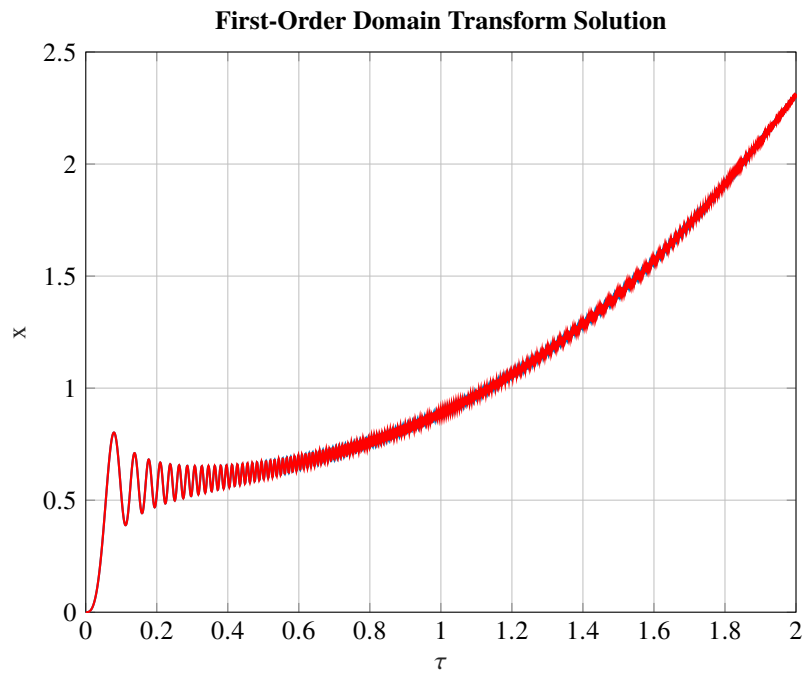


Figure 5.7. First-order IVP solution in $[0,2]$ domain with $x_0 = x(0) = 0$.

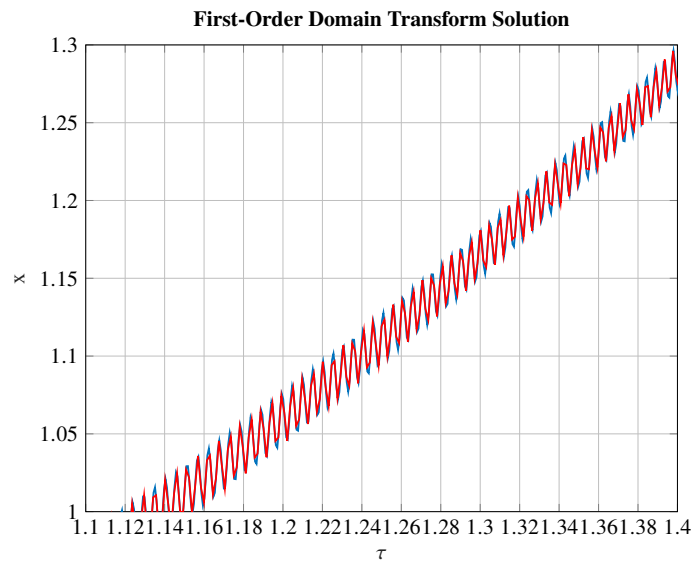


Figure 5.8. The highly oscillatory behavior of this example first-order IVP when applied to $[0,2]$ domain.

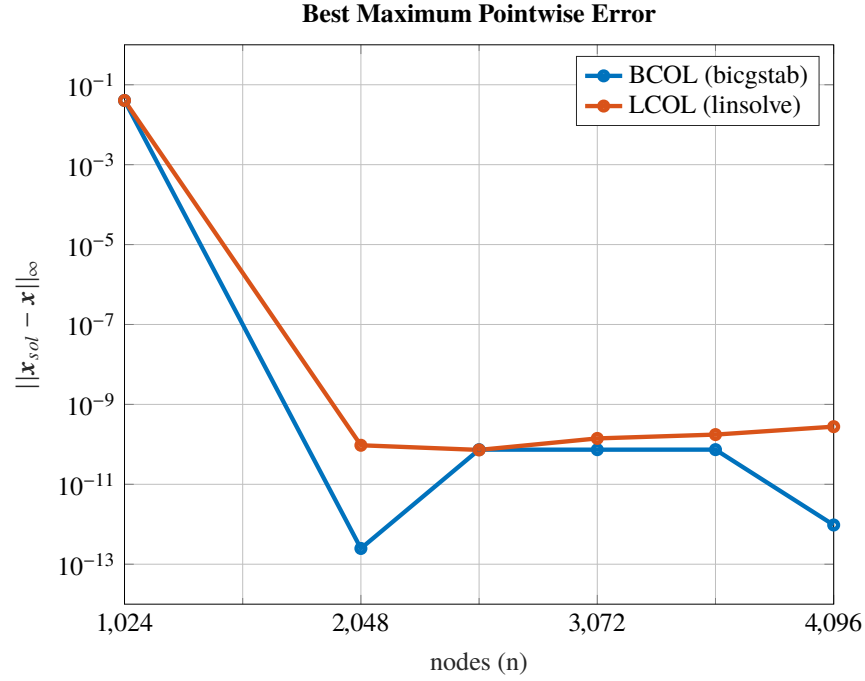


Figure 5.9. First-order IVP maximum absolute pointwise error in $[0,2]$ domain. BCOL uses iterative *bicgstab* and LCOL uses direct *linsolve*. Note that all *bicgstab* iterations for LCOL return trivial results: error > 1 .

In terms of error, the methods diverged similar to the computational domain case, shown in Figure 5.2. In Figure 5.9, it is clear that errors persisted with both methods until enough nodes were present to capture the sinusoidal wave. In this case, that point increased from 600 to roughly 2000. Some numbers of nodes see little to no difference in error between the methods, but BCOL always performs at or better than LCOL. This lack of differentiation in error between methods occurred in the computational domain case (Section 5.4.1 and Figure 5.2) that the authors of [1] avoided these numbers of nodes, which had a less distinguishable gap between methods.

As seen in Figure 5.10, BCOL continued to perform with stability in terms of successful *bicgstab* iterations. Again, all LCOL iterations failed to converge. This is due to the continued increase in the condition number for LCOL as can be observed in Figure 5.3.

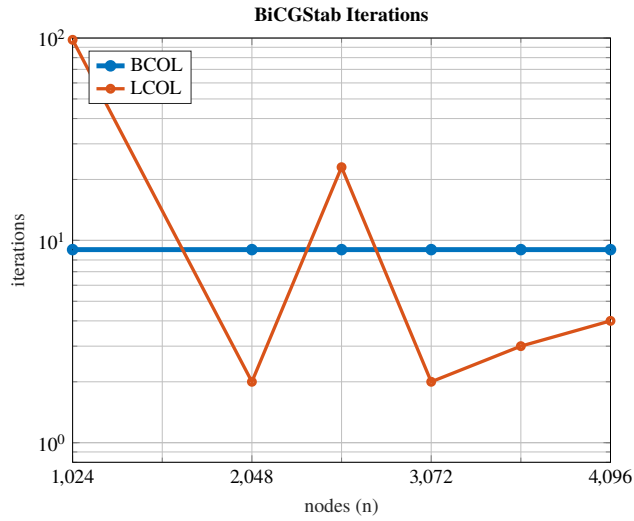


Figure 5.10. First-order IVP iteration count for *bicgstab* in $[0,2]$ domain. All LCOL iterations produce trivial results.

Where method performance really begins to shine through is in average solution time, again over 100 runs. Notice that in Figure 5.11, this difference begins to approach an order of magnitude, which can have meaningful impacts even on a linear application to signal processing, for example.

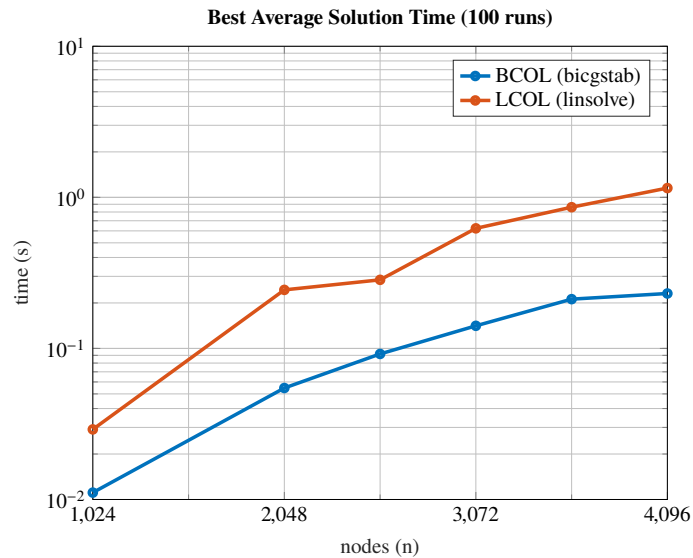


Figure 5.11. First-order IVP solution time in $[0,2]$ domain. Increase in number of nodes widens the gap in performance from Figure 5.6).

5.4.3 Different Nodes

As discussed previously, these results for BCOL at CGR nodes (B-CGR) was simply a choice. Similar results were achieved with BCOL at CGL nodes (B-CGL) as well as at LGR and LGL nodes. In Chapter 6, these first-order matrices are used to solve the basic satellite optimal control problems. Other formulations of BCOL are also shown in their application to optimal control specifically.

5.5 Second-Order Linear Systems

In linear formulations, the second-order problem allows for investigation of a single-state boundary value problem (BVP). Optimal control requires a BVP, but uses a system of first-order nonlinear dynamics instead. This BVP basis lends itself to Gauss–Lobatto nodes sets, which have values strictly at the bounds of the domain (i.e., $\tau_0 = -1$ and $\tau_f = 1$). This BVP construction is conspicuously missing from the linear first-order seen in Section 5.3.

Unlike the first-order preconditioner, the following second-order derivation was only applied to Gauss–Lobatto nodes. Furthermore, Dirichlet boundary conditions are built into this formulation since it is a BVP. As a reference to the reader, this section maps concepts discussed in Section 3.2 of [1], which covers a much more diverse array of boundary conditions.

5.5.1 Collocation Schemes

In [1], the authors focus their work on second-order, linear systems. Therefore, the second-order schemes shown in this section are adapted for this research in order to validate their work and explain their truncated proof in greater detail. For more general applicability to linear systems, the authors then take significant effort to adjust this work to higher- and lower-order problems as well as different boundary conditions.

The first-order preconditioner and collocation schemes shown in Section 5.3 are more applicable to optimal control, but an important understanding of the application to BVPs can be seen for the second-order case.

The generic single-state, second-order, linear BVP used is:

$$\ddot{x}(\tau) + r(\tau)\dot{x}(\tau) + s(\tau)x(\tau) = f(\tau), \quad (5.43)$$

where the initial and final states, $x(\pm 1) = x_{\pm}$, are known. This is an adaption of Equation 3.23 of [1]. It is important to note that Gauss–Lobatto points are inherent to this formulation since there are two degrees of freedom required to capture the boundary values (i.e., $x_{\pm}$).

5.5.2 Lagrange Collocation (LCOL)

For a second-order linear BVP, the LCOL scheme as:

$$(\mathbf{D}_{\text{in}}^{(2)} + \mathbf{\Lambda}_r \mathbf{D}_{\text{in}}^{(1)} + \mathbf{\Lambda}_s) \mathbf{x}_{\text{in}} = \mathbf{f}_{\text{in}} - \mathbf{x}_B. \quad (5.44)$$

Following the “inner” notation (in), $\mathbf{f}_{\text{in}} = \{f(\tau_i)\}_{i=1}^{N-1}$ and $\mathbf{x}_{\text{in}} = \{x_N(\tau_i)\}_{i=1}^{N-1}$. The boundary conditions have been moved to the right side of the equation via $\mathbf{x}_B$, which is the vector defined by:

$$\mathbf{x}_B = \{x_-(d_{i0}^{(2)} + r(\tau_i)d_{i0}^{(1)}) + x_+(d_{iN}^{(2)} + r(\tau_i)d_{iN}^{(1)})\}_{i=1}^{N-1}, \quad (5.45)$$

where $d_{ij}^{(k)}$ is the entry of $D^{(k)}$ at row i and column j . Finally, the functions $r(\tau)$ and $s(\tau)$ are now diagonal matrices such that:

$$\mathbf{\Lambda}_r = \text{diag}(r(\tau_1), \dots, r(\tau_{N-1})), \quad (5.46)$$

$$\mathbf{\Lambda}_s = \text{diag}(s(\tau_1), \dots, s(\tau_{N-1})). \quad (5.47)$$

Therefore, the relevant condition number for this method comes from

$$\mathbf{A}_{\text{LCOL}} = \mathbf{D}_{\text{in}}^{(2)} + \mathbf{\Lambda}_r \mathbf{D}_{\text{in}}^{(1)} + \mathbf{\Lambda}_s. \quad (5.48)$$

Using this LCOL basis, the P-LCOL and BCOL schemes are as follows.

5.5.3 Preconditioned-Lagrange Collocation (P-LCOL)

The left preconditioner scheme uses the identity shown in (5.2) to create the linear system:

$$(\mathbf{I}_{N-1} + \mathbf{B}_{2,\text{in}} \mathbf{\Lambda}_r \mathbf{D}_{\text{in}}^{(1)} + \mathbf{B}_{2,\text{in}} \mathbf{\Lambda}_s) \mathbf{x}_{\text{in}} = \mathbf{B}_{2,\text{in}} (\mathbf{f}_{\text{in}} - \mathbf{x}_B). \quad (5.49)$$

For the second-order case,

$$\mathbf{I}_{N-1} + \mathbf{B}_{2,\text{in}}\mathbf{\Lambda}_r\mathbf{D}_{\text{in}}^{(1)} + \mathbf{B}_{2,\text{in}}\mathbf{\Lambda}_s. \quad (5.50)$$

As can be seen, the higher-order differentiation matrix disappears, but the lower order $\mathbf{D}_{\text{in}}^{(1)}$ remains. Likewise, computation time has increased since more matrix-vector operations are required.

5.5.4 Birkhoff Collocation (BCOL)

The right preconditioner scheme uses identity (5.9) to create the linear system:

$$(\mathbf{I}_{N-1} + \mathbf{\Lambda}_r\mathbf{B}_{2,\text{in}}^{(1)} + \mathbf{\Lambda}_s\mathbf{B}_{2,\text{in}})\mathbf{v} = \mathbf{f}_{\text{in}} - \mathbf{v}_B, \quad (5.51)$$

where $\mathbf{v}$ is a vector of the second derivatives of the states, $\mathbf{v} = (\ddot{x}(\tau_i))_{i=1}^{N-1}$, and $\mathbf{v}_B$ is an analog of $\mathbf{x}_B$ such that

$$\mathbf{v}_B = \{x_-(b_{i0}^{(1)}r(\tau_i) + b_{i0}s(\tau_i) + x_+(b_{iN}^{(1)}r(\tau_i) + b_{iN}s(\tau_i)))\}_{i=1}^{N-1}. \quad (5.52)$$

Now the relevant BCOL condition number relates to

$$\mathbf{I}_{N-1} + \mathbf{\Lambda}_r\mathbf{B}_{2,\text{in}}^{(1)} + \mathbf{\Lambda}_s\mathbf{B}_{2,\text{in}}. \quad (5.53)$$

In this scheme, no ill-conditioned differentiation operators remain and the system remains computationally the same. However, one must recover the states after linear system convergence via:

$$\mathbf{x}_{\text{in}} = \mathbf{B}_{2,\text{in}}\mathbf{v} + x_- \mathbf{b}_0 + x_+ \mathbf{b}_N, \quad (5.54)$$

where $\mathbf{b}_j = \{B_j(x(\tau_i))\}_{i=1}^{N-1}$ for $j = 0, N$.

5.5.5 Desired Properties of $\mathbf{B}_2$

For the rest of the section, the second-order preconditioner, $\mathbf{B}_2$, will be shown as $\mathbf{B}$ to reduce notational clutter. As before, a polynomial $p(\tau) \in \mathbb{P}_N$ can be uniquely expressed by

$$p(\tau) = x(-1)B_0(\tau) + \sum_{j=1}^{N-1} \ddot{x}(\tau_j)B_j(\tau) + x(1)B_N(\tau) \quad (5.55)$$

where the Birkhoff interpolation basis polynomials $\{B_j\}_{j=0}^N \subseteq \mathbb{P}_N$ satisfy

$$B_0(-1) = 1, \quad B_0(1) = 0, \quad \ddot{B}_0(\tau_i) = 0, \quad 1 \leq i \leq N-1; \quad (5.56)$$

$$B_j(-1) = 0, \quad B_j(1) = 0, \quad \ddot{B}_j(\tau_i) = \delta_{ij}, \quad 1 \leq i, j \leq N-1; \quad (5.57)$$

$$B_N(-1) = 0, \quad B_N(1) = 1, \quad \ddot{B}_N(\tau_i) = 0, \quad 1 \leq i \leq N-1. \quad (5.58)$$

such that $p(\pm 1) = x(\pm 1)$, $\ddot{p}(\tau_j) = \ddot{x}(\tau_j)$. These properties correspond to Section 3.1 in [1].

5.5.6 Birkhoff Polynomials

As with the first-order case, we use an orthonormal polynomial expansion. This time the expansion defines the second derivative of the Birkhoff interpolation polynomials such that:

$$\ddot{B}_j(\tau) = \sum_{k=0}^{N-2} \beta_{kj} \frac{P_k^{(\alpha, \alpha)}(\tau)}{\gamma_k}, \quad (5.59)$$

where β_{kj} are the second-order expansion coefficients and γ_k is the normalization constants. By integrating (5.59) twice, one arrives at the solution for creating the second-order Birkhoff polynomials:

$$B_j(\tau) = \sum_{k=0}^{N-2} \beta_{kj} \frac{\partial^{-2} P_k^{(\alpha, \alpha)}(\tau)}{\gamma_k} + (1 + \tau)c_1 + c_2, \quad (5.60)$$

where $\partial_\tau^{-2} P_k^{(\alpha, \alpha)}(\tau)$ is defined according to (4.16).

Since $B_j(-1) = 0$ is desired and using (4.18), one can determine that $c_2 = 0$. This matches the first-order case. However, the same is not true for c_1 . Since $B_j(1) = 0$ is also dictated,

and noting that $\partial_\tau^{-2} P_k^{(\alpha, \alpha)}(1)$ does not necessarily vanish, it is generally true that

$$2c_1 = \sum_{k=0}^{N-2} \beta_{kj} \frac{\partial_\tau^{-2} P_k^{(\alpha, \alpha)}(1)}{\gamma_k}. \quad (5.61)$$

Therefore, the general, second-order Birkhoff polynomials are:

$$B_j(\tau) = \sum_{k=0}^{N-2} \frac{\beta_{kj}}{\gamma_k} \left(\partial_\tau^{-2} P_k^{(\alpha, \alpha)}(\tau) + \frac{1+\tau}{2} \partial_\tau^{-2} P_k^{(\alpha, \alpha)}(1) \right). \quad (5.62)$$

This matches the Chebyshev case shown in [1], save for a change in polynomial notation from $P_k^{(\alpha, \alpha)}$ to T_k . For Legendre, $\partial_\tau^{-m} P_k^{(\alpha, \alpha)}(1) = 0$ for $k \geq m$. Therefore, (5.62) reduces to:

$$B_j(\tau) = (\beta_{1j} - \beta_{0j}) \frac{\tau+1}{2} + \sum_{k=0}^{N-2} \left(\beta_{kj} \frac{\partial_\tau^{-2} P_k(\tau)}{\gamma_k} \right). \quad (5.63)$$

5.5.7 Expansion Coefficients

For the second-order expansion coefficients, the same approach is used as in the first-order case: the expansion (5.59) is defined in order to take advantage of the Kronecker delta behavior of the orthogonal polynomials. Likewise, notation and derivation will only be done for the Chebyshev polynomials, but applies to Legendre as well. See Appendix B in [1] for the abbreviated proof for Legendre polynomials upon which this proof is based. The second-order reduction is as follows:

$$\begin{aligned} \int_{-1}^1 \ddot{B}_j(\tau) T_l(\tau) \omega(\tau) d\tau &= \int_{-1}^1 \left(\sum_{k=0}^{N-2} \beta_{kj} \frac{T_k(\tau)}{\gamma_k} \right) T_l(\tau) \omega(\tau) d\tau \\ &= \sum_{k=0}^{N-2} \left(\beta_{kj} \left(\int_{-1}^1 \frac{T_k(\tau) T_l(\tau) \omega(\tau)}{\gamma_k} d\tau \right) \right) \\ &= \sum_{k=0}^{N-2} \beta_{kj} \delta_{kl}. \end{aligned} \quad (5.64)$$

Therefore, for $l = 0, \dots, N$,

$$\sum_{k=0}^{N-1} \alpha_{kj} \delta_{kl} = \int_{-1}^1 \dot{B}_j(\tau) T_l(\tau) \omega(\tau) d\tau. \quad (5.65)$$

By the Kronecker delta, only the terms where $l = k$ in (5.65) remain and the expansion reduces to:

$$\beta_{kj} = \int_{-1}^1 \ddot{B}_j(\tau) T_k(\tau) \omega(\tau) d\tau \quad (5.66)$$

The integral in (5.66) can be represented exactly via the orthogonal polynomial basis with $\phi(\tau) = \ddot{B}_j(\tau) T_k(\tau)$ as shown in (4.29). After quadrature has been applied,

$$\beta_{kj} = \ddot{B}_j(-1) T_k(-1) w_0 + \ddot{B}_j(1) T_k(1) w_N + T_k(\tau_j) w_j \quad (5.67)$$

$$= [(-1)^k \ddot{B}_j(-1) + \ddot{B}_j(1)] w_0 + T_k(\tau_j) w_j. \quad (5.68)$$

One arrives at (5.67) due to the fact that the inside of the quadrature ($x \neq \pm 1$) has the desired property that $\ddot{B}_j(\tau_i) = \delta_{ij}$, which results in only the $T_k(\tau_j) w_j$ term being non-zero. Furthermore, (5.68) results from the fact that $T_k(\pm 1) = (\pm 1)^k$ and $w_0 = w_N$ for Gauss–Lobatto.

Next, solutions for $\ddot{B}_j(\pm 1)$ are found by evaluating (5.68) at $k = N - 1, N$ where β_{kj} must equal zero due to orthogonality of $T_N(\tau)$ and $T_{N-1}(\tau)$ relative to all other Chebyshev polynomials already used for integration (i.e., $T_k(\tau)$ for $k = 0, 1, \dots, N - 2$). Together, these terms evaluate to the following system of equations:

$$0 = [(-1)^{N-1} \ddot{B}_j(-1) + \ddot{B}_j(1)] w_0 + T_{N-1}(\tau_j) w_j, \quad (5.69)$$

$$0 = [(-1)^N \ddot{B}_j(-1) + \ddot{B}_j(1)] w_0 + T_N(\tau_j) w_j.$$

The solution of the linear system (5.69),

$$\ddot{B}_j(\pm 1) = -(\pm 1)^n \left(\frac{w_j}{2w_0} \right) (T_N(\tau_j) \pm T_{N-1}(\tau_j)), \quad (5.70)$$

is then substituted into (5.68) and one arrives at an expression for the second-order expansion

coefficients:

$$\beta_{kj} = w_j \left(T_k(\tau_j) - \frac{1 - (-1)^{N+k} T_{N-1}(\tau_j)}{2} - \frac{1 + (-1)^{N+k} T_N(\tau_j)}{2} \right). \quad (5.71)$$

This completes the definition of the second-order Birkhoff polynomial basis used to form the second-order preconditioner, $\mathbf{B}_2$. Similar domain transformations follow from Section 5.3.8 such that

$$\bar{\mathbf{B}}_2 = \left(\frac{t_f - t_0}{2} \right)^2 \mathbf{B}_2. \quad (5.72)$$

5.6 Second-Order Validation

An example second-order linear boundary value problem (BVP),

$$\ddot{x}(\tau) - (1 + \sin \tau) \dot{x}(\tau) + e^\tau x(\tau) = 0, \quad \tau \in [-1, 1]; \quad x(\pm 1) = 1, \quad (5.73)$$

adapted from Equation 3.31 of [1], was used to compare more detailed results published than the first-order case. This validation was done in order to note slight deviations in results that are, most likely, attributable to implementation changes and adjustments to the *bicgstab* function in MATLAB. Therefore, while the second-order case is not strictly useful for optimal control applications, this analysis is provided for more detailed matching to results in [1].

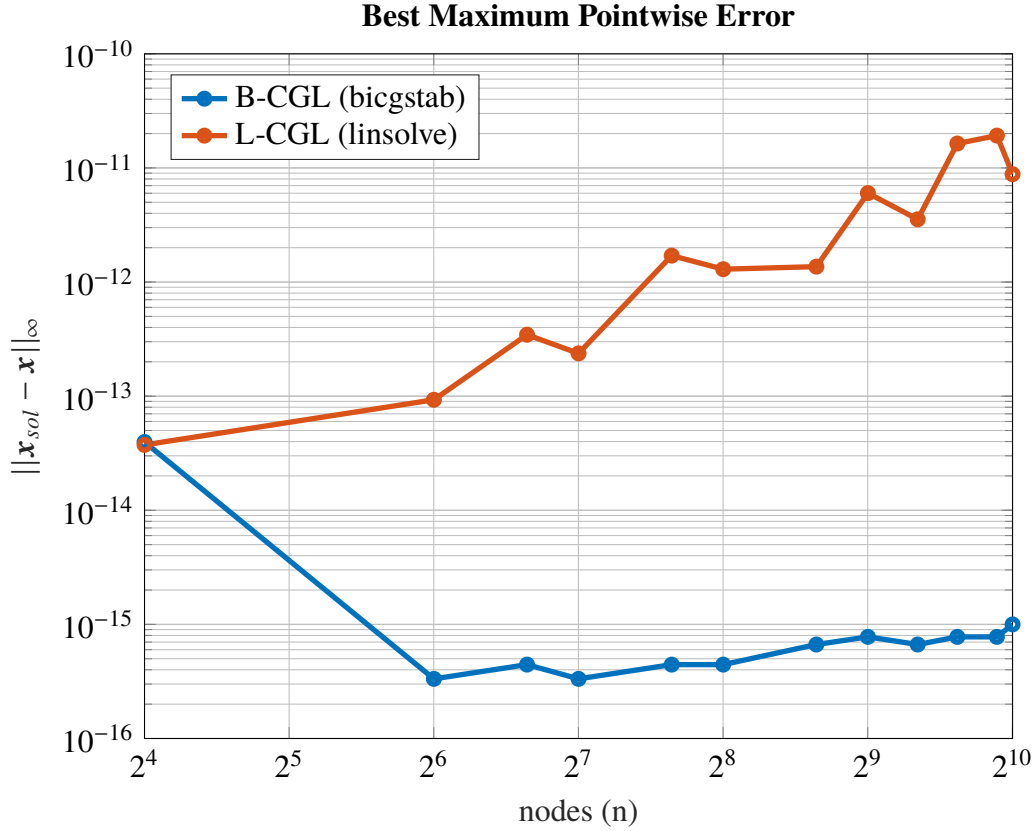


Figure 5.12. Second-order BVP maximum absolute pointwise error. BCOL uses iterative *bicgstab* solver and LCOL uses direct *linsolve*.

Maximum pointwise error, shown in Figure 5.12, are of a similar order, but do not exactly match data shown in Table 1 in [1]. This may be attributable to an apparent change in MATLAB's *bicgstab* that now include half-iterations. While these provided detailed error metrics were not provided for their first-order case, slight deviations from published results can be assumed to apply there as well. Also, like the first-order case, *linsolve* was required to achieve these results for LCOL, but not BCOL. However, as seen in Figure 5.13, *bicgstab* did achieve meaningful, if sometimes poor results for LCOL up until about 600 nodes. At this point, *bicgstab* solutions were trivial.

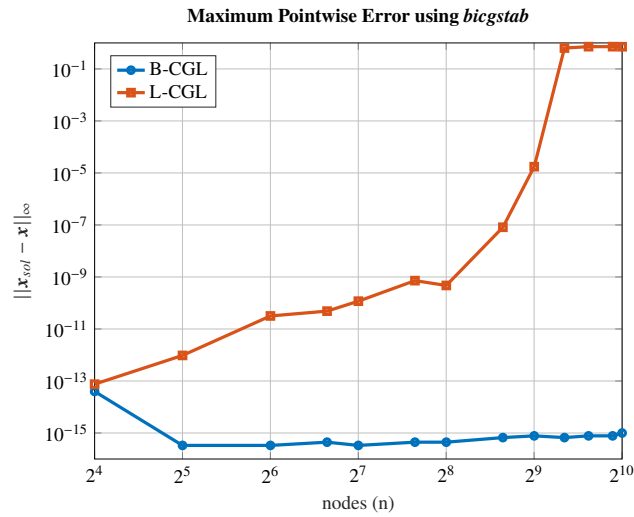


Figure 5.13. Second-order BVP maximum absolute pointwise error for *bicgstab* solutions.

As seen in Figure 5.14, BCOL takes minimal *bicgstab* iterations to converge on a solution. LCOL on the other hand exhibits a high iteration count before encountering the ill-conditioned “scalar quantity” error, discussed previously, or it simply reached the maximum iteration stopping criteria.

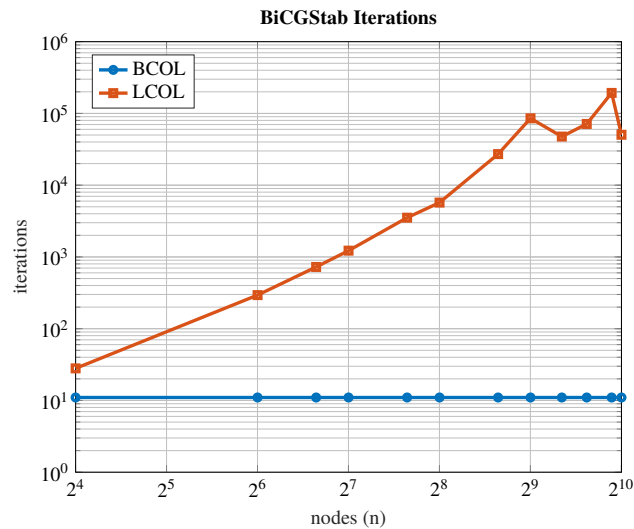


Figure 5.14. Second-order BVP iteration count for *bicgstab*. All LCOL solutions past 600 nodes produce trivial results.

Despite slight differences in error, the condition numbers—as defined by (5.48) and (5.53)—matched exactly to results published in Table 1 of [1]. This exact match in condition numbers indicate that the linear systems being solved are, most likely, consistent and that implementation differences are the cause of deviations in exact error values. The validated condition numbers are displayed numerically in Table 5.2 and shown visually in Figure 5.15.

Table 5.2. Second-order BVP condition number growth/stability

N	BCOL	LCOL
64	6.430	7.228e5
128	6.498	1.155e7
256	6.532	1.848e8
512	6.549	2.957e9
1024	6.557	4.731e10

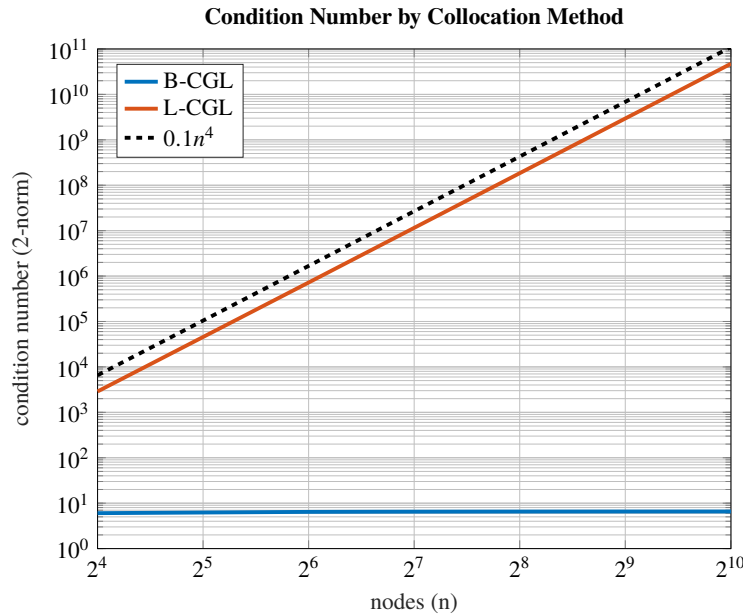


Figure 5.15. Second-order BVP condition number behavior disparity between collocation methods. See Section 5.5.1 for definitions of A_{B-CGL} and A_{L-CGL} . Note that BCOL is $O(1)$ and LCOL is $O(n^4)$.

5.7 Adapting to Optimal Control

As seen in this Chapter, neither the first-order IVP or second-order BVP linear formulations of the **B** shown in [1] fully address the needs of a preconditioner in optimal control. Optimal control formulations have a first-order system of nonlinear equations that are driven by controls outside the framework of state dynamics. Furthermore, optimal control ultimately expresses a BVP implementation of endpoint conditions. However, the **B** formulations in this chapter clearly demonstrated promise in linear applications, which—once understood—can be applied broadly to optimal control formulations with various node sets. This application to optimal control is the subject of Chapter 6.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:

Well-Conditioned Psuedospectral Optimal Control

In optimal control, the requirements differ from the linear collocation schemes of Chapter 5. A first-order state-space representation of dynamics is used, but boundary conditions are not restricted to a single state. Some states may have only initial or final conditions and other states may have both. Therefore, we seek a preconditioner that supports a first-order nonlinear system of differential equations that ultimately expresses a BVP. Only a first-order operator is desired as in Section 5.3, but with concerns beyond that of an IVP. These two goals present a conflict in construction for the Birkhoff preconditioner.

In Section 6.1, we will begin with consideration for applying Birkhoff preconditioners to optimal control. Section 6.2 demonstrates the inability to modify the $\mathbf{B}$ to strictly apply a BVP and explores two alternative reformulations for proceeding with applying Birkhoff preconditioners to optimal control. The implementation details for BCOL in optimal control are shown in Sections 6.3 and 6.4. Finally, Sections 6.5 and 6.6 cover the results of successful application of BCOL to optimal control problems.

6.1 The Preconditioner for Optimal Control

At first glance, the application of a Birkhoff preconditioner to optimal control seems a straightforward extension of the linear case. However, unlike in the linear case, there is no requirement that linear boundary conditions even exist. Furthermore, there is no invertible matrix $\mathbf{A}$ that solves a nonlinear problem. The relevant condition numbers are based on that of the changing Jacobians used by the nonlinear solver. This disconnect from the framework of Chapter 5 is seen in Section 6.1.1.

6.1.1 Basic Nonlinear Collocation

The generic nonlinear, first-order nonlinear problem is:

$$\dot{x}(\tau, x) = f(\tau, x), \tag{6.1}$$

where either an initial or final states are known, $x(-1) = x^0$ or $x(1) = x^F$, respectively. For this general problem, the LCOL scheme cannot be represented in matrix form as was done in Section 5.44. Also, optimal control boundary conditions are rarely exclusively defined by linear initial or final states. Therefore, the direct application of linear collocation schemes to optimal control is not straightforward. One must consider all the unique aspects of optimal control.

First, it is important to recognize that optimal control is not simply a nonlinear system of differential equations. Optimal control requires constrained optimization via nonlinear constraints and an objective. Further, the general case is a system of nonlinear differential algebraic equations. Lastly, and most importantly, optimal control is about finding a control trajectory. These controls do not obey the defined system dynamics, but are the important driving factor. An immediate, but subtle, result of these controls comes in the need for an additional control constraint.

6.1.2 Another Constraint for Controls

In the linear, first-order IVP the initial state, x_0 , are coupled to the dynamics via $x_0 = v_0$ and $\mathbf{x} = \mathbf{B}\mathbf{v}$. Therefore, by integrating the dynamics and reintroducing of the constant, all information is retained. However, in optimal control, the system dynamics are also driven by controls. Therefore, In order to couple the initial control with the state dynamics, one must relate the polynomial approximation of the states,

$$p(\tau) = x(-1)B_0(\tau) + \sum_{j=1}^N \dot{x}(\tau_j)B_j(\tau), \quad (6.2)$$

to the state dynamics,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau), \quad (6.3)$$

where x will be used to represent only state k such that

$$\dot{x} = \dot{x}_k = f_k(\mathbf{x}, \mathbf{u}, \tau), \quad (6.4)$$

By differentiating (6.2), one gets

$$\dot{p}(\tau) = x(-1)\dot{B}_0(\tau) + \sum_{j=1}^N \dot{x}(\tau_j)\dot{B}_j(\tau). \quad (6.5)$$

Since we seek a polynomial approximation of the derivatives that satisfies

$$\dot{p}(\tau) = \dot{x} = \mathbf{f}_k(\mathbf{x}, \mathbf{u}, \tau), \quad (6.6)$$

then it must hold that

$$\mathbf{f}_k(\mathbf{x}, \mathbf{u}, \tau) = x(-1)\dot{B}_0(\tau) + \sum_{j=1}^N \dot{x}(\tau_j)\dot{B}_j(\tau). \quad (6.7)$$

At the initial point, $\tau = -1$,

$$\mathbf{f}_k(\mathbf{x}_0, \mathbf{u}_0, \tau_0) = x(-1)\dot{B}_0(-1) + \sum_{j=1}^N \dot{x}(\tau_j)\dot{B}_j(-1). \quad (6.8)$$

From (5.28) it can be seen that B_0 is a polynomial with coefficients, $\mathbf{b}_0$, that are the solution to a linear system such that

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ & \mathbf{D}_{1j} & & & \\ & \mathbf{D}_{2j} & & & \\ & \vdots & & & \\ & \mathbf{D}_{Nj} & & & \end{bmatrix} \begin{bmatrix} b_{00}(\tau_0) \\ b_{01}(\tau_1) \\ \vdots \\ \vdots \\ b_{0N}(\tau_N) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \quad (6.9)$$

where the left-hand side operator is simply $\tilde{\mathbf{D}}^{(1)}$, which is known to be invertible by (5.3). Since the solution that $B_0(\tau) = 1$ exists, then it is also unique. Thus, $\dot{B}_0(-1) = 0$ and, by definition, $\dot{x}(\tau_j) = v_j$,

$$\mathbf{f}_k(\mathbf{x}_0, \mathbf{u}_0, \tau_0) = \sum_{j=1}^N v_j \dot{B}_j(-1). \quad (6.10)$$

Therefore, in order to properly constrain u_0 in the IVP formulation the initial dynamics, one must enforce

$$f_k(\mathbf{x}_0, \mathbf{u}_0, \tau_0) = \mathbf{v}_{in}^T \psi \quad (6.11)$$

where

$$\psi = (\dot{B}_1(-1), \dot{B}_2(-1), \dots, \dot{B}_N(-1))^T. \quad (6.12)$$

Further, this constraint can be shown as equivalent to implementing an LCOL-type constraint such that

$$f_k(\mathbf{x}_0, \mathbf{u}_0, \tau_0) = \sum_{j=1}^N D_j(-1) x_j. \quad (6.13)$$

Given this equivalence, it must hold that the domain transformation constant of $\mathbf{D}$ is used so that

$$f_k(\mathbf{x}_0, \mathbf{u}_0, t_0) = \left(\frac{2}{t_f - t_0} \right) \mathbf{v}_{in}^T \psi. \quad (6.14)$$

Since the same polynomial $p(\tau)$ is used to represent all states in $\mathbf{x}$, this extends to the entire collocation where

$$\mathbf{f}(\mathbf{x}_0, \mathbf{u}_0, t_0) = \left(\frac{2}{t_f - t_0} \right) \mathbf{V}_{in}^T \psi, \quad (6.15)$$

and $\mathbf{V}_{in}^T$ consists of only the derivative portions of $\mathbf{v}$ (i.e., $\mathbf{v}_{in}^T$) of each state. See Section 6.4 for more on BCOL implementation for optimal control. These principles also apply to other formulations like that of a Final Value Problem (FVP).

6.1.3 Neglecting the Control Constraint

Without the additional constraint discussed in Section 6.1.2, control solutions exhibit strange behaviors due to this unconstrained degree of freedom for the nonlinear solver. To visualize this behavior, control solutions to BP2 are used in this section.

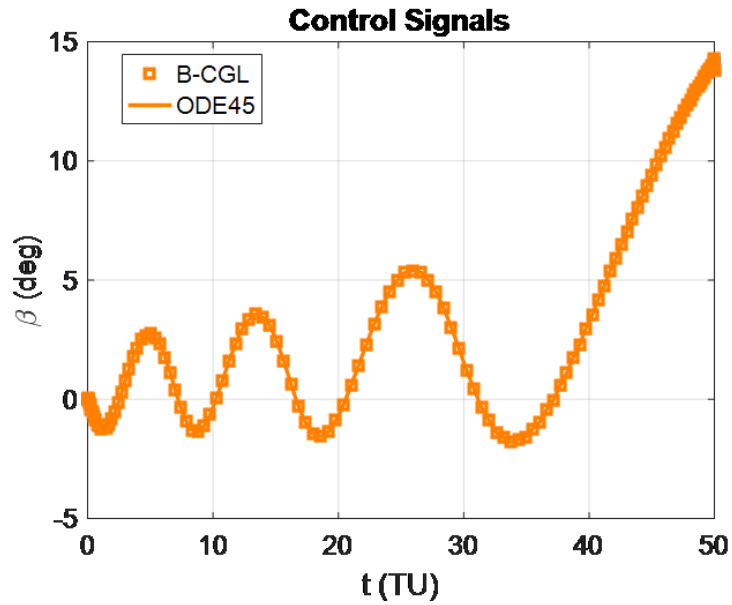


Figure 6.1. Control jitter at final nodes of B-CGL solution to BP1.

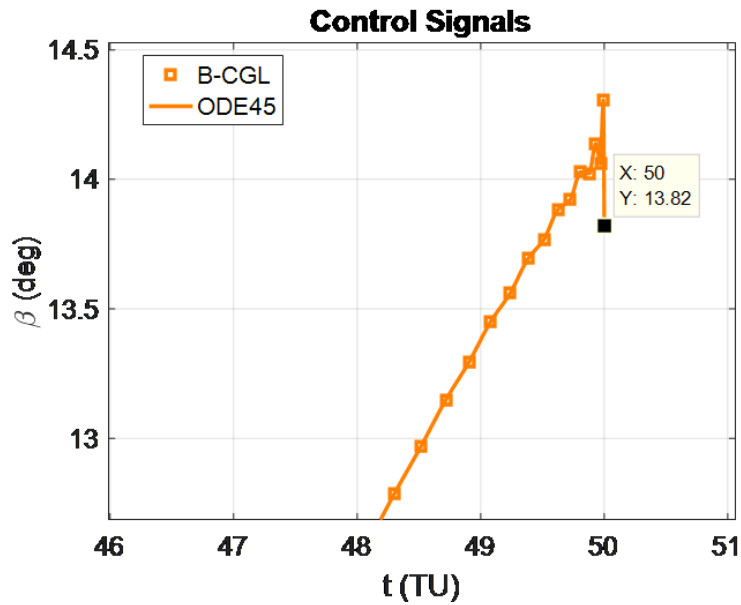


Figure 6.2. Detail of B-CGL jitter shown in Figure 6.1.

For B-CGL, the exactness of the final node being directly at the domain bound means that the solver can adjust controls slightly about the true solution without having a significant impact on feasibility. The B-CGL control jitter is shown Figure 6.1 and highlighted further

in Figure 6.2. However, the slight inaccuracy of B-CGR implementations actually proves more stable when this control constraint is missing. The nonlinear solver reaches stopping criteria without resorting to jittering controls about the true solution to achieve higher tolerance. See Figures 6.3 and 6.4.

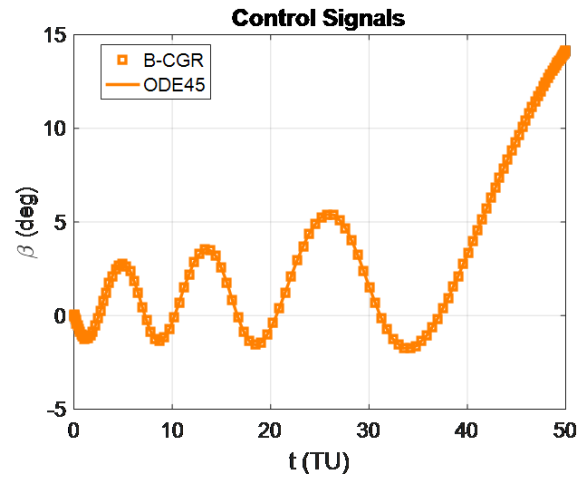


Figure 6.3. Lack of control jitter in a B-CGR solution to BP1.

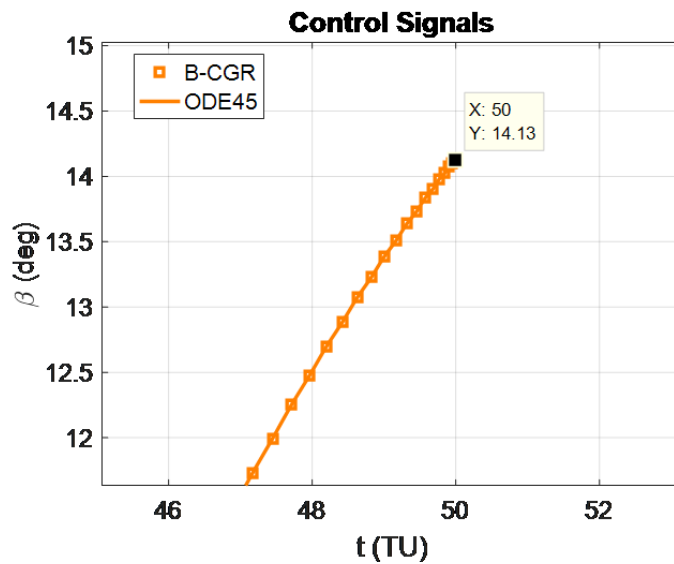


Figure 6.4. Detail of final nodes in B-CGR solution shown in Figure 6.3.

6.2 Reformulations of the Problem

The linear preconditioners used in Chapter 5 are based on systems that are dissimilar to the nonlinear problems, in general, and optimal control, in particular. For that reason, it is important to consider different formulations of $\mathbf{B}_1$ than the first-order IVP in Section 5.3.

6.2.1 The Over-Constrained Boundary Value Problem Formulation

Optimal control is a first-order system of nonlinear differential algebraic equations in state-space form. Therefore, it is not a single, unique polynomial that we seek to express. Instead, it is actually a set of polynomials, $\{p_l\}_{l=1}^{n_x} \in \mathbb{P}_N$, with different boundary conditions per polynomial as dictated by the initial and final conditions of the states of the optimal control problem.

However, for our immediate purposes, each of these polynomials will be assumed to have Dirichlet boundary conditions to cover the case where both initial and final states are known. This is the case for three of the four states in BP1. There is no nodal requirement, but Gauss–Lobatto points are a reasonable candidate due to nodes at the boundary conditions locations as was the case with the linear, second-order systems in Section 5.5. Therefore, for a BVP formulation we desire to have a polynomial $p(\tau) \in \mathbb{P}_N$ defined such that

$$p(\tau) = x(-1)B_0(\tau) + \sum_{j=1}^{N-1} \dot{x}(\tau_j)B_j(\tau) + x(1)B_N(\tau), \quad (6.16)$$

where the Birkhoff interpolation basis polynomials $\{B_j\}_{j=0}^N \subseteq \mathbb{P}_N$ satisfy

$$B_0(-1) = 1, \quad B_0(1) = 0, \quad \dot{B}_0(\tau_i) = 0, \quad 1 \leq i \leq N-1; \quad (6.17)$$

$$B_j(-1) = 0, \quad B_j(1) = 0, \quad \dot{B}_j(\tau_i) = \delta_{ij}, \quad 1 \leq i, j \leq N-1; \quad (6.18)$$

$$B_N(-1) = 0, \quad B_N(1) = 1, \quad \dot{B}_N(\tau_i) = 0, \quad 1 \leq i \leq N-1. \quad (6.19)$$

These properties are similar to those specified for second-order, linear collocation in Section 5.5.1 except for the derivatives of the Birkhoff polynomials are first derivatives. We begin by using a polynomial expansion to define this first derivative of the Birkhoff inter-

polation polynomials as was done in the linear case. Therefore,

$$\dot{B}_j(\tau) = \sum_{k=0}^{N-2} \alpha_{kj} \frac{P_k^{(\alpha,\alpha)}(\tau)}{\gamma_k}, \quad 1 \leq j \leq N-1, \quad (6.20)$$

where α_{kj} are the expansion coefficients and γ_k are the normalization constants. Now $k = 0 \dots N-2$ since two degrees of freedom are used to fix the initial and final values, $x(\pm 1)$. Via integration,

$$B_j(\tau) = \sum_{k=0}^{N-2} \alpha_{kj} \frac{\partial^{-1} P_k^{(\alpha,\alpha)}(\tau)}{\gamma_k} + c. \quad (6.21)$$

However, we already have an overconstrained problem. The constant of integration, c , cannot be found to satisfy the two conditions $B_j(\pm 1) = 0$. The condition that $B_j(-1) = 0$ and (4.17) combine to dictate that $c = 0$, as was the case for the first-order, linear preconditioner in Section 5.3.5. We also desire to have $B_j(1) = 0$. However, $\partial_\tau^{-1} P_k^{(\alpha,\alpha)}(1)$ does not completely vanish. Specifically, for Chebyshev $\partial_\tau^{-1} T_k(1) \neq 0$ for most k and for Legendre $\partial_\tau^{-1} P_0(1) = 2$. See Remark 3.4 in [1]. Therefore, to have $B_j(1) = 0$ we must have $c \neq 0$. This constant of integration cannot be set to meet both conditions.

For the single-state, first-order case devoid of controls, this is indeed an overconstrained problem. However, the BVP formed in optimal control is done so via Pontryagin's Principle as discussed in Section 3.5. In this formulation, the BVP becomes well-posed by accounting for costate dynamics associated with costs to optimality as well as the variability of the final time for time-free problems. Therefore, the set of polynomials we need to express is actually $\{p_l\}_{l=1}^{2n_x+1}$ and their coupling is complex and problem-dependent.

6.2.2 Final Value Problem Formulation

We now investigate the final value problem (FVP) formulation. The important takeaway from this construct is that the IVP formulation in Chapter 5 is not a requirement for a first-order operator. For this case, a polynomial $p(\tau) \in \mathbb{P}_N$ can be uniquely expressed by

$$p(\tau) = x(1)B_N(\tau) + \sum_{j=0}^{N-1} \dot{x}(\tau_j)B_j(\tau), \quad (6.22)$$

where the Birkhoff interpolation basis polynomials $\{B_j\}_{j=0}^{N-1} \subseteq \mathbb{P}_N$ satisfy

$$B_j(1) = 0, \quad \dot{B}_j(\tau_i) = \delta_{ij}, \quad 0 \leq i, j \leq N-1, \quad (6.23)$$

$$B_N(1) = 1, \quad \dot{B}_N(\tau_i) = 0, \quad 0 \leq i \leq N-1, \quad (6.24)$$

such that $\dot{p}(\tau_j) = \dot{x}(\tau_j)$, $p(1) = x(1)$. These properties mirror those of the IVP in Chapter 5. The Birkhoff polynomials in (5.30) differ from the IVP by the constant of integration, c . In the FVP case, $B_j(1) = 0$ and because, in general, $\partial^{-1}P_k^{(\alpha,\alpha)}(1) \neq 0$, the constant $c \neq 0$. Therefore,

$$B_j(\tau) = \sum_{k=0}^{N-1} \frac{\alpha_{kj}}{\gamma_k} \left(\partial^{-1}P_k^{(\alpha,\alpha)}(\tau) + \partial^{-1}P_k^{(\alpha,\alpha)}(1) \right), \quad 0 \leq j \leq N-1. \quad (6.25)$$

Along with the constant of integration, the expansion coefficients, α_{kj} , must change as well in order to define the Birkhoff polynomials for the FVP case. The quadrature remains the same in that

$$\alpha_{kj} = \int_{-1}^1 \dot{B}_j(\tau) T_k(\tau) \omega(\tau) d\tau = \sum_{i=0}^{N-1} \dot{B}_j(\tau_i) T_k(\tau_i) w_i \quad (6.26)$$

however, in the FVP case $\dot{B}_j(1)$ is unspecified. Thus,

$$\begin{aligned} \alpha_{kj} &= T_k(\tau_j) w_j + \dot{B}_j(1) T_k(1) w_N \\ &= T_k(\tau_j) w_j + (1)^k \dot{B}_j(1) w_N. \end{aligned} \quad (6.27)$$

Note that (6.27) restricts quadrature to a node being specifically at +1. This requirement means that Gauss–Radau nodes must first be negated and then flipped, along with corresponding quadrature weights, in order to align the fixed point to +1 instead of −1. No change to Gauss–Lobatto is required. By evaluating (6.27) at $k = N$,

$$\dot{B}_j(1) = -\left(\frac{w_j}{w_N} \right) T_N(\tau_j). \quad (6.28)$$

By (6.27) and (6.28), the first-order FVP expansion coefficients are

$$\alpha_{kj} = w_j (T_k(\tau_j) - T_N(\tau_j)), \quad 0 \leq j, k \leq N-1. \quad (6.29)$$

Together, (6.25) and (6.29) determine the Birkhoff polynomials for the FVP formulation. In order to constrain the final control, one differentiates (6.22) to get

$$\dot{p}(\tau) = \sum_{j=0}^{N-1} (\dot{x}(\tau_j) \dot{B}_j(\tau)) + x(1) \dot{B}_N(\tau). \quad (6.30)$$

As in the IVP case, it must hold that for state k represented as x ,

$$f_k(\mathbf{x}, \mathbf{u}, \tau) = \sum_{j=0}^{N-1} (\dot{x}(\tau_j) \dot{B}_j(\tau)) + x(1) \dot{B}_N(\tau). \quad (6.31)$$

When evaluated at the final point, $\tau = 1$,

$$f_k(\mathbf{x}_f, \mathbf{u}_f, \tau_f) = \sum_{j=0}^{N-1} (\dot{x}(\tau_j) \dot{B}_j(1)) + x(1) \dot{B}_N(1). \quad (6.32)$$

Similar to the IVP case, by (6.24) $\dot{B}_N(1) = 0$. Therefore,

$$f_k(\mathbf{x}_f, \mathbf{u}_f, \tau_f) = \sum_{j=0}^{N-1} (v_j \dot{B}_j(1)) . \quad (6.33)$$

In the FVP formulation, u_f is constrained via

$$f(\mathbf{x}_f, \mathbf{u}_f, \tau_f) = \mathbf{v}_{in}^T \psi \quad (6.34)$$

where

$$\psi = (\dot{B}_0(1), \dot{B}_1(1), \dots, \dot{B}_{N-1}(1))^T. \quad (6.35)$$

Further, it must hold that in the physical domain

$$f_k(\mathbf{x}_f, \mathbf{u}_f, t_f) = \left(\frac{2}{t_f - t_0} \right) \mathbf{v}_{in}^T \psi. \quad (6.36)$$

As was seen in the IVP case in Section 6.1.2, when all states in $\mathbf{x}$ are considered this becomes a vector of constraints where

$$\mathbf{f}(\mathbf{x}_0, \mathbf{u}_0, t_0) = \left(\frac{2}{t_f - t_0} \right) \mathbf{V}_{in}^T \boldsymbol{\psi}. \quad (6.37)$$

For the two first-order formulations, the “inner” portion of the domain is redefined such that

$$\begin{aligned} \mathbf{i}_{in,IVP} &= (1, 2, \dots, N), \\ \mathbf{i}_{in,FVP} &= (0, 1, \dots, N - 1). \end{aligned} \quad (6.38)$$

6.2.3 Imperfect Preconditioning

Section 6.2.1 represents an attempt at a perfect preconditioner (i.e., $\mathbf{D}_{in}\mathbf{B}_{in} = \mathbf{I}_{n-1}$) to the pseudospectral optimal control. However, it is not possible to construct a general preconditioner in this framework as shown by an overconstrained problem. Therefore, compromises to achieving perfect preconditioning are investigated. This can be seen as a reasonable course of action if the condition number of the collocation scheme grows slower than linear. As an example, the mixing of CGL and CGR nodes in a $\mathbf{D}_{in,CGL}\mathbf{B}_{in,CGR}$ right preconditioning scheme results in such behavior. See Figure 6.5.

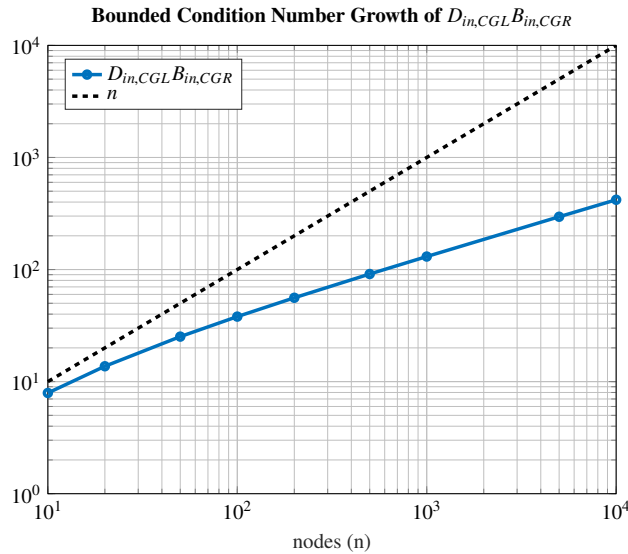


Figure 6.5. The condition growth for a right-preconditioning scheme using a mix node sets: $\mathbf{D}_{in,CGL}$ and $\mathbf{B}_{in,CGR}$.

The behavior of the condition number shown in Figure 6.5 makes sense since

$$\mathbf{D}_{in,CGL}\mathbf{B}_{in,CGL} = \mathbf{I}_{n-1}, \quad (6.39)$$

and due to the fact that CGL and CGR approach approximately the same points as node count increases. Furthermore, this imperfect matrix is not singular and, thus, can be used to create a collocation solution at CGL even though $\mathbf{B}$ is defined at CGR nodes.

6.2.4 Relaxing the Last Constraint

An attempt at this imperfect preconditioning can be made by relaxing one constraint from Section 6.2.1, which makes the problem overconstrained. In this section, the properties outlined in Section 6.2.1 are amended to

$$B_0(-1) = 1, \quad B_0(1) = 0, \quad \dot{B}_0(\tau_i) = 0, \quad 1 \leq i \leq N-1; \quad (6.40)$$

$$B_j(-1) = 0, \quad B_j(1) = 0, \quad \dot{B}_j(\tau_i) = \delta_{ij}, \quad 1 \leq i, j \leq N-1. \quad (6.41)$$

This sacrifices the one degree in the quadrature, but allows for partial specification of a final boundary condition via the expansion coefficients. Unlike the second-order, linear case, the NLP framework of optimal control allows for the imposition of the final conditions without the direct application via $B_N(1) = 1$. Only $\mathbf{B}_{in}$ is used. Now, however, the last row and column $i, j = N$ of $\mathbf{B}$ are also unused.

With one less degree in quadrature, the upper limits of j and k are reduced by 1. This is the only change to (5.31), which results in the Birkhoff polynomials being

$$B_j(\tau) = \sum_{k=0}^{N-2} \alpha_{kj} \frac{\partial^{-1} P_k^{(\alpha, \alpha)}(\tau)}{\gamma_k}, \quad 1 \leq j \leq N-1. \quad (6.42)$$

The expansion coefficients now have two unknown variables: $\dot{B}_j(-1)$ and $\dot{B}_j(1)$, as was the case in the second-order BVP. This means that equation for linear, first-order coefficients (5.35) becomes:

$$\alpha_{kj} = \dot{B}_j(-1)T_k(-1)w_0 + \dot{B}_j(1)T_k(1)w_N + T_k(\tau_j)w_j, \quad (6.43)$$

$$= [(-1)^k \dot{B}_j(-1) + \dot{B}_j(1)] w_0 + T_k(\tau_j)w_j. \quad (6.44)$$

Using the same logic as before—of the orthogonality of the remaining polynomials—a system of equations is found:

$$\begin{aligned} 0 &= [(-1)^{N-1} \dot{B}_j(-1) + \dot{B}_j(1)] w_0 + T_{N-1}(\tau_j) w_j \\ 0 &= [(-1)^N \dot{B}_j(-1) + \dot{B}_j(1)] w_0 + T_N(\tau_j) w_j. \end{aligned} \quad (6.45)$$

The solution to this system of equations is:

$$\dot{B}_j(\pm 1) = -(\pm 1)^N \left(\frac{w_j}{2w_0} \right) (T_N(\tau_j) \pm T_{N-1}(\tau_j)). \quad (6.46)$$

Inserting those solutions, (6.46), into (6.44) one arrives at the new expression for the expansion coefficients:

$$\alpha_{kj} = w_j \left(T_k(\tau_j) - \frac{1 - (-1)^{N+k} T_{N-1}(\tau_j)}{2} - \frac{1 + (-1)^{N+k} T_N(\tau_j)}{2} \right), \quad (6.47)$$

where $1 \leq j \leq N-1$ and $0 \leq k \leq N-2$.

Since CGL nodes are a clear choice for this method, this formulation is labeled as CGL 2 in Figure 6.6 and Figure 6.7. When comparing these two figures, it appears that P-LCOL, in which $\mathbf{B}_{\text{in}} \mathbf{D}_{\text{in}}$ is utilized, looks much more promising. All other methods shown produce a constant condition number of 1 as expected from the perfect preconditioners. Therefore, CGR, CGL, and LGR methods are coincident in Figure 6.6 and 6.7.

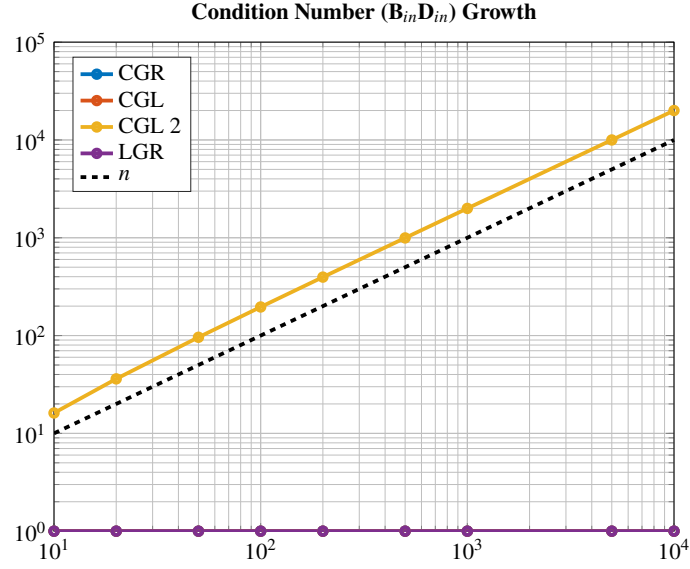


Figure 6.6. Condition number growth of the imperfect left preconditioner $\mathbf{B}_{in}\mathbf{D}_{in} = \mathbf{I}_{n-1}$ using the CGL 2 (6.47) formulation. Trends as $O(n)$.

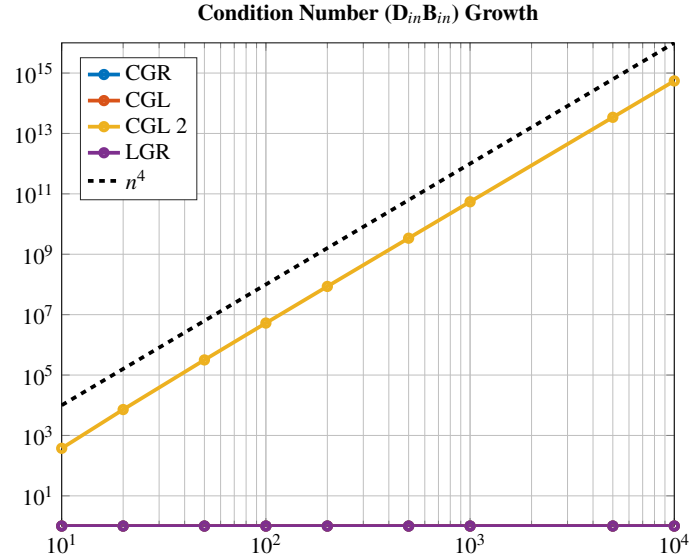


Figure 6.7. Condition number growth of the imperfect right preconditioner $\mathbf{D}_{in}\mathbf{B}_{in} = \mathbf{I}_{n-1}$ using the CGL 2 (6.47) formulation. Trends as $O(n^4)$.

Neither of these bounds are particularly desirable and, therefore, the work in this thesis does not focus on applying a BVP formulation. In principle, this type of formulation could

improve performance of the nonlinear solver by better coupling dynamic constraints and boundary conditions.

6.3 Well-Conditioned Optimal Control Collocation

Despite the current limitations coupling dynamics and boundary conditions explicitly in $\mathbf{B}$, constrained nonlinear optimization software packages like Knitro can still solve these problems. The initial and final states, which are used viewed as constants in $\mathbf{X} = \mathbf{B}\mathbf{V}$, can—in the NLP framework—be varied with all other free variables.

The pseudospectral preconditioner seeks to transform the LCOL scheme for optimal control:

$$\bar{\mathbf{D}}\mathbf{X} = f(\mathbf{X}, \mathbf{U}, t). \quad (6.48)$$

Using $\mathbf{B}$ as a left preconditioner, the P-LCOL scheme for optimal control becomes

$$\mathbf{X} = \bar{\mathbf{B}}f(\mathbf{X}, \mathbf{U}, t). \quad (6.49)$$

P-LCOL removes the ill-conditioned operator, but makes the Jacobian for the NLP dense where the original dynamics are generally not. Therefore, the right preconditioner in BCOL presents the most compelling choice for application to optimal control. This scheme begins with the introduction of a state derivatives matrix, $\mathbf{V}$, such that:

$$\mathbf{X} = \bar{\mathbf{B}}\mathbf{V}. \quad (6.50)$$

Here $\mathbf{V}$ is advantageous notation that can be remembered as velocity for states, which are often defined by kinematics. By combining (6.48) and (6.50), one arrives at an optimal control derivative matching solution for BCOL optimal control:

$$\bar{\mathbf{D}}_{\text{in}}\bar{\mathbf{B}}_{\text{in}}\mathbf{V}_{\text{in}} = \mathbf{V}_{\text{in}} = f_{\text{in}}(\mathbf{X}, \mathbf{U}, t). \quad (6.51)$$

The entire optimal control BCOL scheme is found by combining (6.51) state recovery from dynamics (6.50) and the additional control constraints shown in (6.15) for the IVP or (6.37)

for the FVP formulation. Therefore, optimal control BCOL scheme is

$$\begin{aligned}\mathbf{X} &= \bar{\mathbf{B}}\mathbf{V}, \\ \mathbf{V}_{\text{in}} &= f_{\text{in}}(\mathbf{X}, \mathbf{U}, t), \\ f(\mathbf{x}_y, \mathbf{u}_y, t_y) &= \left(\frac{2}{t_f - t_0} \right) \mathbf{V}_{\text{in}}^T \psi,\end{aligned}\tag{6.52}$$

where $y = 0$ for an IVP, $y = f$ for an FVP, $\psi = (\dot{B}_j(\tau_y))$ for $j \in \mathbf{i}_{\text{in}}$ as defined by (6.38).

6.3.1 The BCOL Optimal Control NLP

Using the right-preconditioning scheme, the discretized optimal control problem, P^n (3.11) can be transformed into a BCOL NLP. This is similar to the LCOL scheme (3.11). However, the nonlinear problem (NLP) now has two-part dynamics and a well-conditioned collocation foundation

$$\begin{aligned}(\text{NLP}_{\text{BCOL}}) \left\{ \begin{array}{l} \text{Minimize} \quad J(\mathbf{X}, \mathbf{U}, t_0, t_f) \\ \quad \quad \quad = E(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) + \mathbf{I}^T \bar{\mathbf{W}} \mathbf{F} \\ \text{Subject to:} \\ \quad \quad \quad \mathbf{X} = \mathbf{B}\mathbf{V} \\ \quad \quad \quad \mathbf{V}_{\text{in}} = f_{\text{in}}(\mathbf{X}, \mathbf{U}, t) \\ \quad \quad \quad f(\mathbf{x}_y, \mathbf{u}_y, t_y) = \left(\frac{2}{t_f - t_0} \right) \mathbf{V}_{\text{in}} \psi_y \\ \quad \quad \quad \mathbf{e}^L \leq \mathbf{e}(\mathbf{x}_0, \mathbf{x}_f, t_0, t_f) \leq \mathbf{e}^U, \\ \quad \quad \quad \mathbf{H}^L \leq \mathbf{h}(\mathbf{X}, \mathbf{U}, t) \leq \mathbf{H}^U. \end{array} \right. \tag{6.53}\end{aligned}$$

where y and “in” are defined as shown in (6.52) and the running cost is transformed into a discretization of the integral, $\mathbf{I}^T \bar{\mathbf{W}} \mathbf{F}$, where

$$\mathbf{I}^T = (1, 1, \dots, 1) \tag{6.54}$$

and the quadrature weights in the physical domain are represented in matrix form as

$$\bar{\mathbf{W}} = \text{diag}(\bar{w}_0, \bar{w}_1, \dots, \bar{w}_N). \tag{6.55}$$

By defining the nonlinear problem (NLP) in terms of the state derivatives, the scheme is notable devoid of the ill-conditioned differentiation operator, $\mathbf{D}$. This has two very important ramifications that must be noted. First, the state variables, $\mathbf{X}$, can be defined as *linear* constraints—as seen in (6.50)—if the problem has a fixed time domain. More on this later. Second, the NLP now appears well-conditioned due to the linear foundations of Chapter 5 and the results seen in Sections 6.5 and 6.6. can remain sparse if the state variables remain as constrained decision variables in the NLP.

6.4 BCOL Implementation

For all collocation methods, the solution is determined from a trivial initial guess. LCOL states were set to unity ($x_{ij} = 1 \forall i, j$) and BCOL state derivatives were set to zero ($v_{ij} = 0 \forall 1 \leq i, j \leq N - 1$).

LCOL used Gauss–Lobatto nodes and the well-conditioned methods used Gauss–Radau nodes. Gauss–Radau solutions shown here did not impose the endpoint condition exactly at t_f due to the last node not being at the domain bound, i.e., $\tau_f \neq 1$. Extending the endpoint constraints to the exact domain bound, via Lagrange interpolation, demonstrates negligible change to results shown here.

6.4.1 Constraints and Boundary Values

The handling of boundary values is another vital difference in the linear approach and the optimal control application. In the literature, state-space systems of equations are not often covered. Thus, the formulations take on higher-order variations with many different methods of imposing boundary conditions. For the optimal control case, these boundary conditions are imposed as constraints in the NLP and do not require explicit boundary value considerations to be baked in to the PSIM.

As will be seen, [1] does explore a first-order PSIM, $\mathbf{B}_1$, which handles the initial value problem (IVP). Optimal control requires a first-order PSIM, but must be able to handle the BVP. For this reason, only the “inner” portion of the PSIM, $\mathbf{B}_{1,\text{in}}$, is used. Future research could be done on an equivalent ultraspherical formulation [6], which takes boundary conditions into account after the core PSIM creation.

From the NLP perspective, the decision variables are comprised in a matrix, $\tilde{\mathbf{X}}$, such that for BCOL:

$$\begin{aligned}\tilde{\mathbf{X}}_{BCOL} &= \begin{bmatrix} v_{1,1} & \dots & v_{1,n_x} & u_{1,1} & \dots & u_{1,n_u} & t_0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & t_f \\ v_{n,1} & \dots & v_{n,n_x} & u_{n,1} & \dots & u_{n,n_u} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{V} & \mathbf{U} & \mathbf{t} \end{bmatrix}.\end{aligned}\tag{6.56}$$

The takeaway from (6.56) is that the nonlinear solver concerns itself with the entire problem space begin constrained. The exact placement of the variables is unimportant.

6.4.2 Linear Constraint Implementation

In order to avoid problems with domain transformation making the linear constraints, the coupling to t_0 and t_f can be offloaded by further splitting the NLP. The linear constraint remains:

$$\bar{\mathbf{X}} = \bar{\mathbf{B}}\mathbf{V},\tag{6.57}$$

$$\mathbf{X} = \left(\frac{t_f - t_0}{2}\right)\bar{\mathbf{X}}.\tag{6.58}$$

Therefore, (6.58) is executed inside the nonlinear constraint function. In order to introduce this linear constraint to the NLP, the decision variable matrix must be expanded to:

$$\begin{aligned}\tilde{\mathbf{X}}_{BCOL} &= \begin{bmatrix} x_{1,1} & \dots & x_{1,n_x} & u_{1,1} & t_0 & v_{1,1} & \dots & v_{1,n_x} \\ \vdots & \ddots & \vdots & t_f & \vdots & \ddots & \vdots & \\ x_{n,n_x} & \dots & x_{n,n_x} & u_{n,1} & & v_{n,1} & \dots & v_{n,n_x} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{X} & \mathbf{U} & \mathbf{t} & \mathbf{V} \end{bmatrix}.\end{aligned}\tag{6.59}$$

6.5 Basic Problem Solutions using BCOL

In this section, BCOL is applied to the basic satellite problems seen in Sections 3.8 and 3.9: BP1 and BP2, respectively. The BCOL results serve to validate the applicability of well-conditioned methods to nonlinear optimal control and provide performance metrics relative to LCOL. BP1 results and performance are shown in Sections 6.5.1 and 6.5.2, respectively.

BP2 results and performance are shown in Sections 6.5.3 and 6.5.4, respectively. For these results, example solutions were shown for CGR nodes. This is simply a choice and, as seen in the performance metrics, using CGL or CGR nodes achieve similar results.

6.5.1 Minimum-Time (BP1) Results

The BCOL solutions to BP1 were nearly identical to those determined by LCOL. Figure 6.8 shows the L-CGL solution coincident with both the nodal solutions from BCOL and P-LCOL. The same is true for *pchip*-interpolated controls used in the *ode45* solutions.

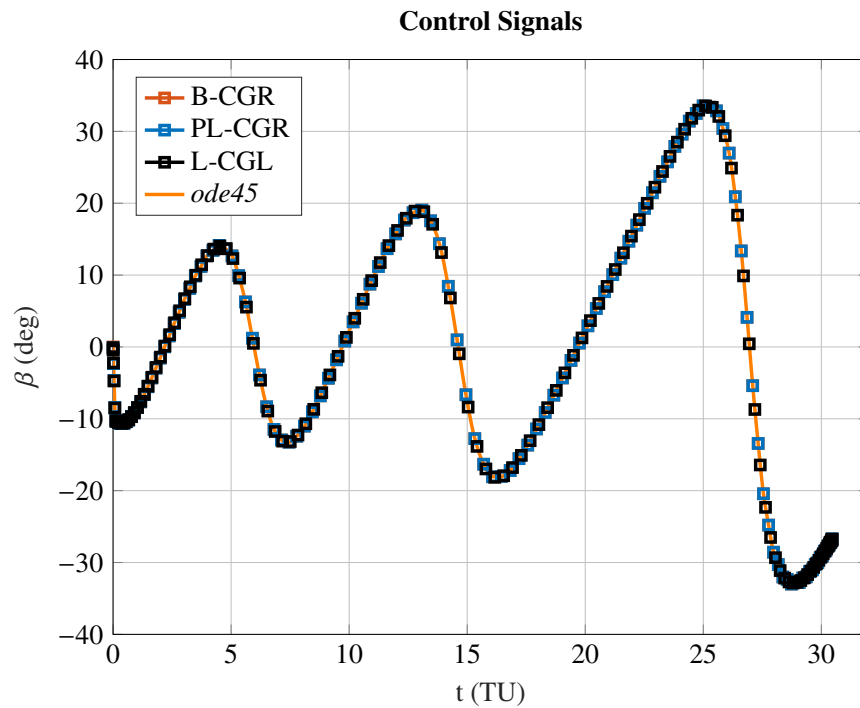


Figure 6.8. 128-node control solutions for minimum-time solution to BP1. Both well-conditioned methods—PL-CGR and B-CGR—are effectively coincident. L-CGL control solution from Section 3.8 provided for reference.

Further, these controls trajectories are found along with valid state trajectories that are shown in Figures 6.9–6.12. The B-CGR solution is indistinguishable from the L-CGL solution when viewed together, as in done in Figures 6.9 and 6.11. Due the change in node set and method, pointwise error is slightly different, but surprisingly consistent. See Figures 6.10 and 6.12.

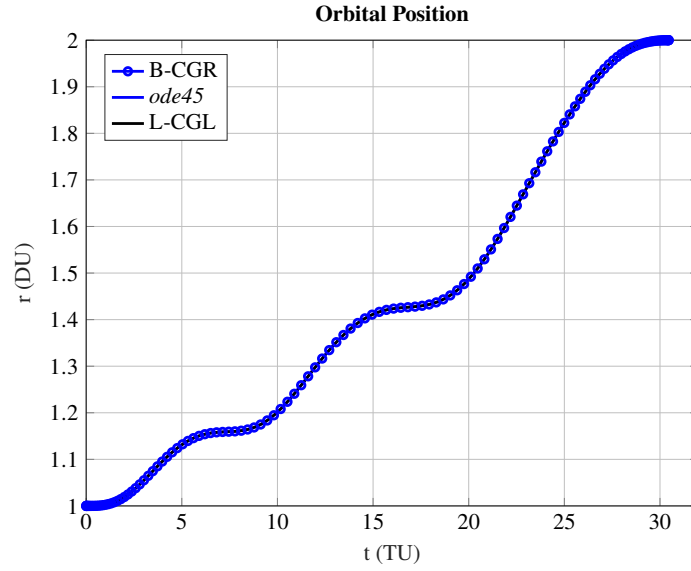


Figure 6.9. Radial position (r) trajectory for BP1 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with *ode45*. The LCOL CGL (L-CGL) solution from Section 3.8 using 128 nodes is shown for comparison.

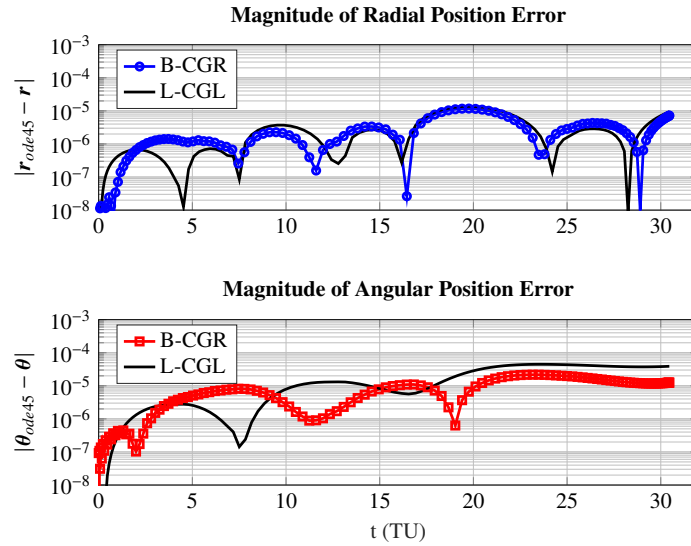


Figure 6.10. Radial (r) and angular (θ) position errors for B-CGR and L-CGL solutions to BP1. Error is relative to *ode45*-propagated solutions using controls given by respective solution.

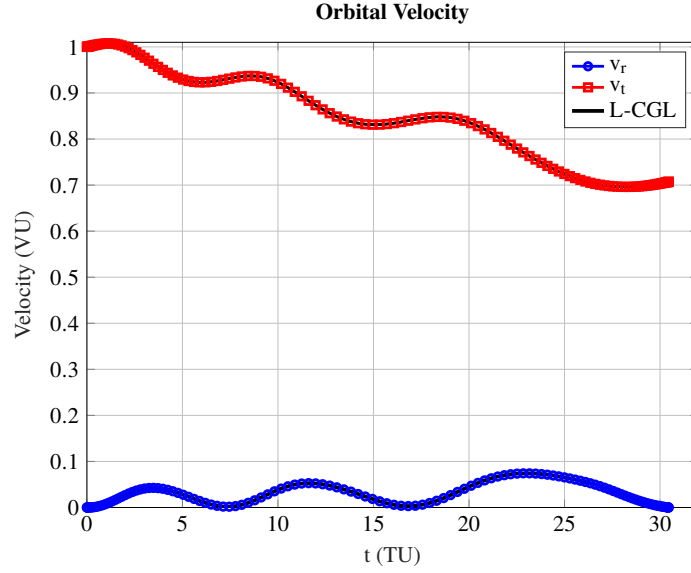


Figure 6.11. Velocity trajectories (v_r and v_t) for BP1 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with *ode45*. The LCOL CGL (L-CGL) solution from Section 3.8 using 128 nodes is shown for comparison.

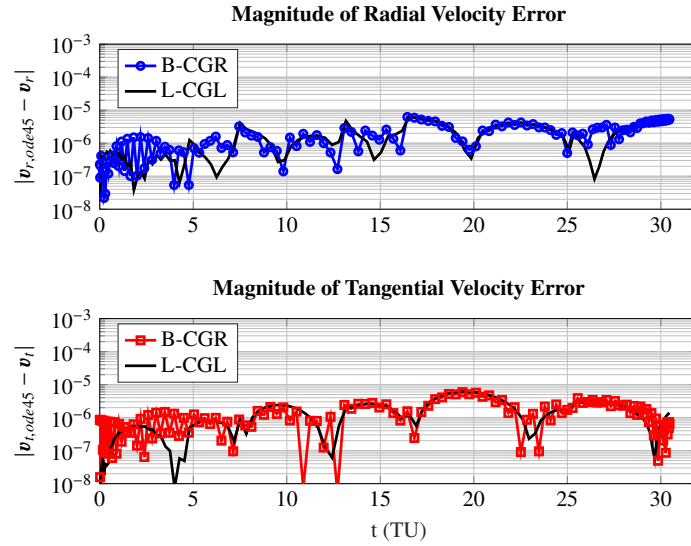


Figure 6.12. Velocity trajectories errors (v_r and v_t) for B-CGR and L-CGL solutions to BP1. Error is relative to *ode45*-propagated solutions using controls given by respective solution.

6.5.2 BP1 Performance

The solution of BP1, solved via LCOL in Section 3.8, is shown in comparison to the preconditioned optimal control methods. BCOL on CGR nodes achieves very similar error metrics to that of the LCOL. Both methods return a responsible order of accuracy given the number of nodes used. However, P-LCOL is about an order of magnitude worse for maximum pointwise error. See Figure 6.13.

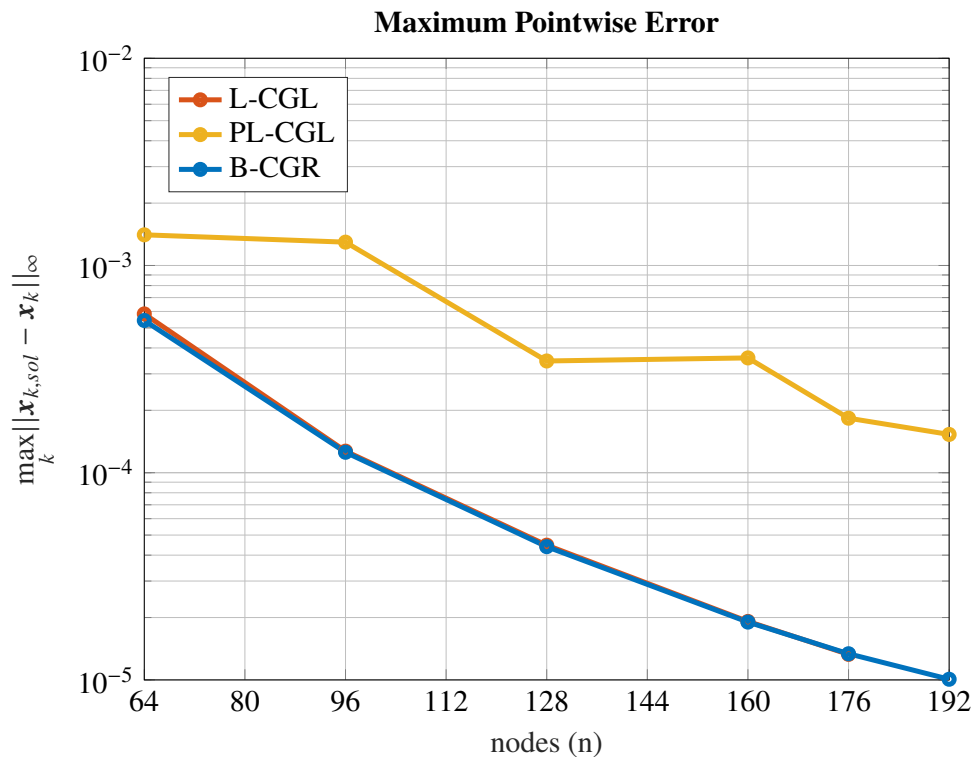


Figure 6.13. Maximum pointwise error when using Knitro to solve BP1. Error is measured against *ode45*-propagated states when using solution controls.

This poor pointwise error for P-LCOL does not come with any advantage in the number of function evaluations needed or time to solution. However, here again BCOL performs exceedingly well. See Figures 6.14 and 6.15.

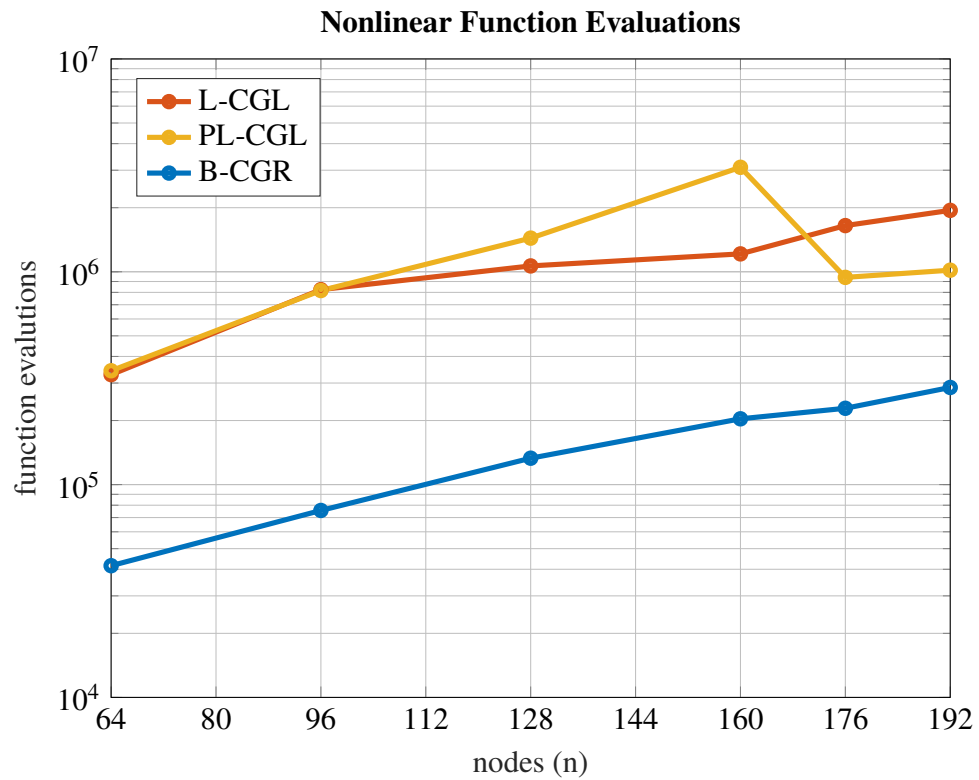


Figure 6.14. Evaluations of the BP1 nonlinear function needed for Knitro solution.

This nearly order of magnitude advantage that BCOL has over LCOL in the number of function evaluations needed appears to indicate that the well-conditioned behavior continues from the clear linear scheme to implementation in optimal control.

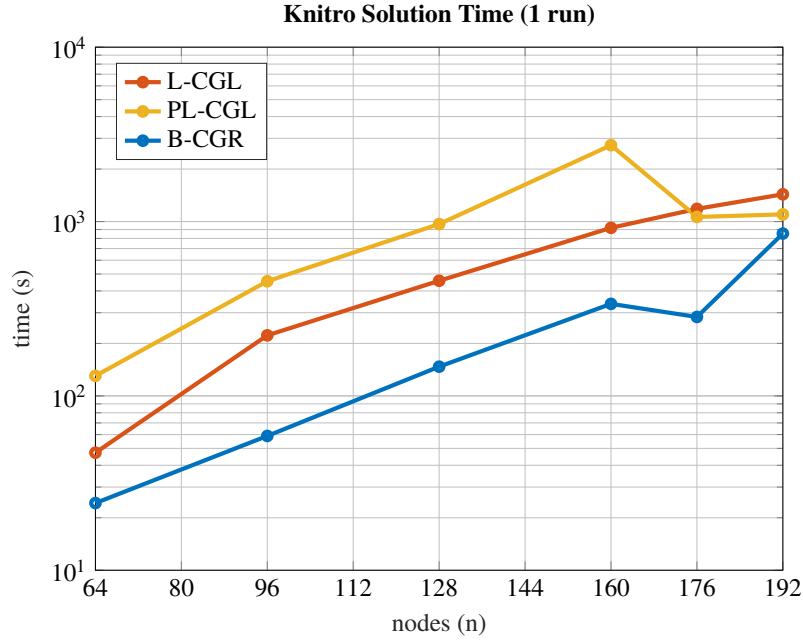


Figure 6.15. Knitro solution time for BP1 using different collocation schemes.

Note that in Figure 6.15, this order of magnitude difference in function evaluations results in a 192-node solution that takes an hour for LCOL and 13 minutes for BCOL.

6.5.3 Max-Energy (BP2) Results

BCOL successfully solves BP2 as well. The B-CGR solutions for the state variables are indistinguishable from those achieved in L-CGL. See Figures 6.16–6.19. In all the solutions presented, 128 nodes were used to achieve errors on the order of 10^{-6} . As is the case with these orbital problems, angular position errors appear to accrue and are thus where the highest pointwise errors occur. This can be seen in Figure 6.17. Both Figures 6.17 and 6.19 demonstrate that despite the change in method (LCOL to BCOL) and node set (CGL to CGR), the two collocation schemes produce very similar pointwise errors.

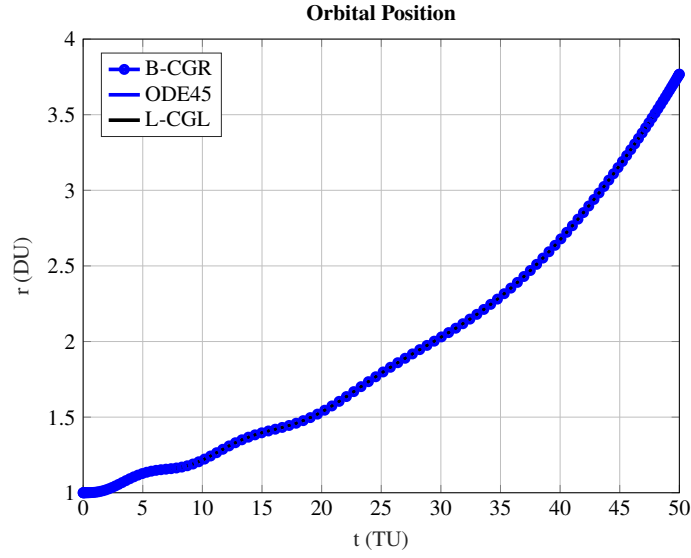


Figure 6.16. Radial position (r) trajectory for BP2 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with *ode45*. The LCOL CGL (L-CGL) solution from Section 3.9 using 128 nodes is shown for comparison.

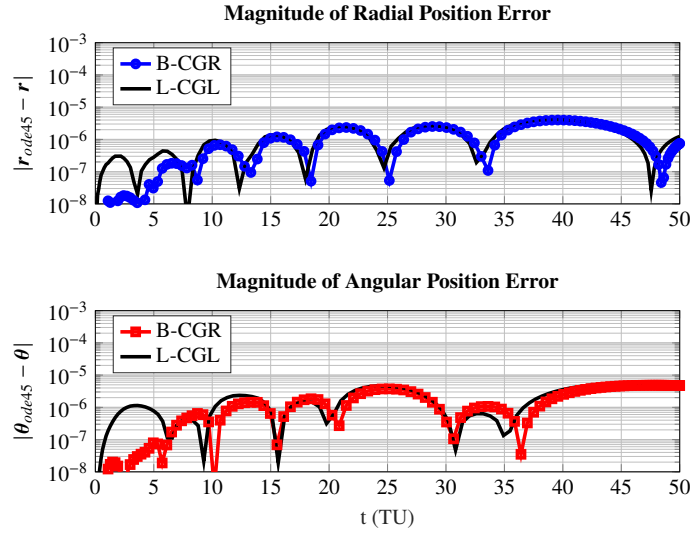


Figure 6.17. Radial (r) and angular (θ) position errors for B-CGR and L-CGL solutions to BP2. Error is relative to *ode45*-propagated solutions using controls given by respective solution.

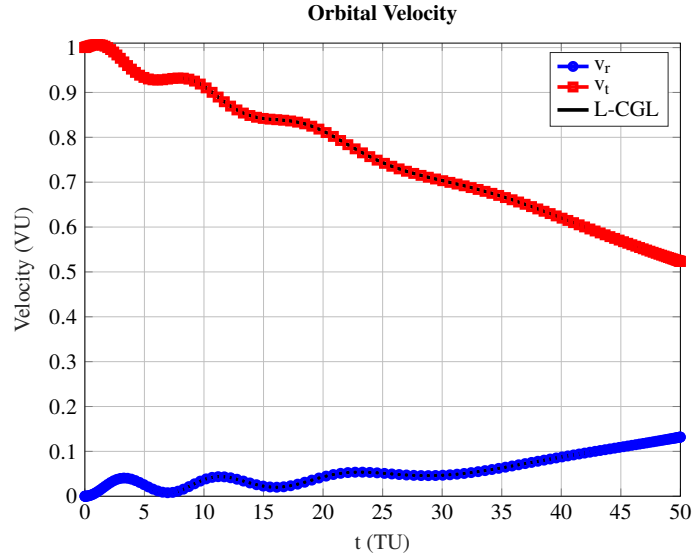


Figure 6.18. Velocity trajectories (v_r and v_t) for BP2 comparing a 128-node BCOL CGR (B-CGR) solution to the position resulting from propagating solution controls with *ode45*. The LCOL CGL (L-CGL) solution from Section 3.9 using 128 nodes is shown for comparison.

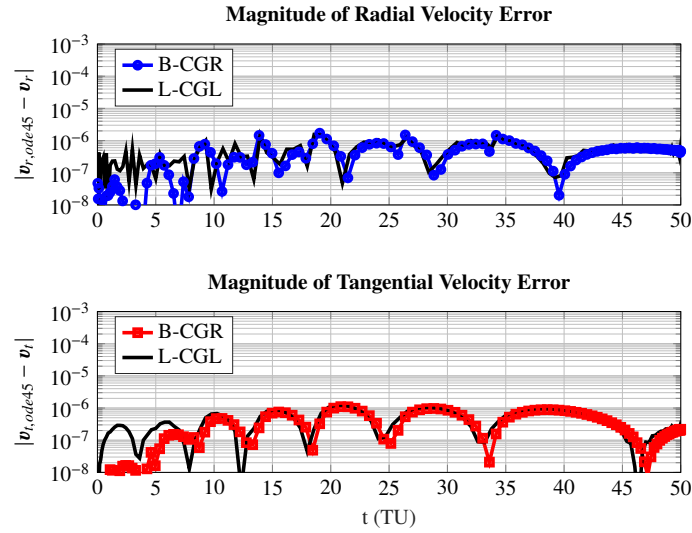


Figure 6.19. Velocity trajectories errors (v_r and v_t) for B-CGR and L-CGL solutions to BP2. Error is relative to *ode45*-propagated solutions using controls given by respective solution.

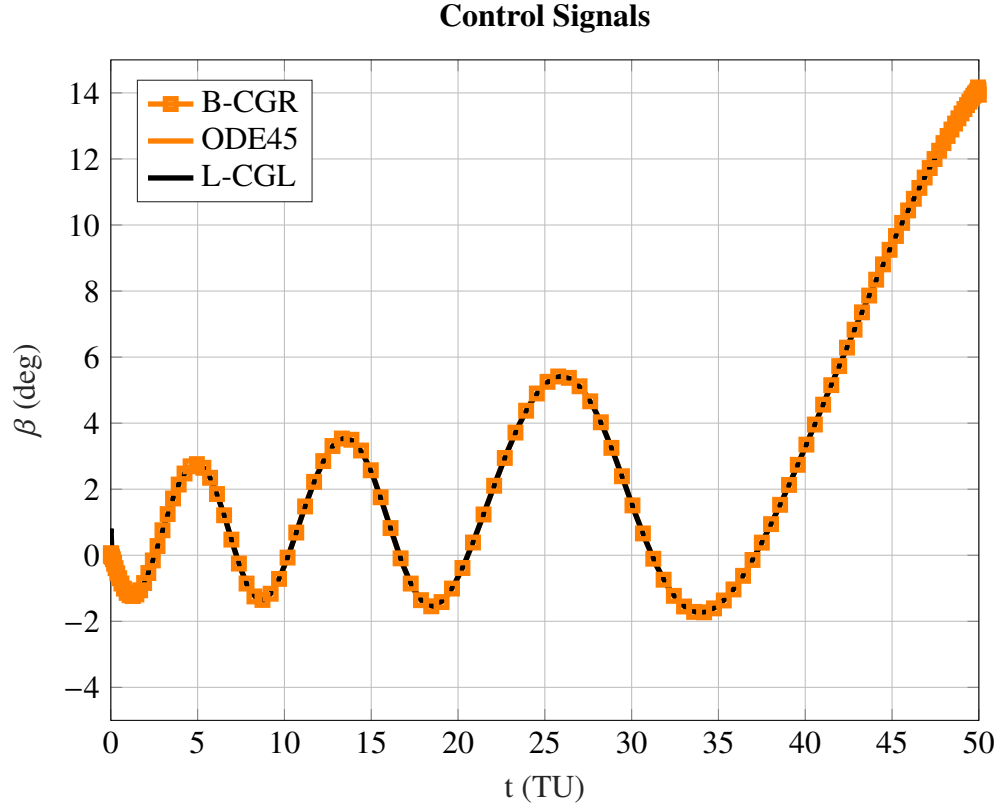


Figure 6.20. Control ($u = \beta$) inputs for BP2 over the fixed time span of 50 TU. L-CGL control solution from Section 3.9 provided for reference.

All methods—LCOL, P-LCOL, and BCOL—produce the same energy-optimal trajectory. Figure 6.20 shows the control solutions for B-CGR and L-CGL produce the same optimal solution, as determined in Section 3.9.

6.5.4 BP2 Performance

The LCOL solution of BP2 from Section 3.9 is shown in comparison to the preconditioned optimal control methods: BCOL and P-LCOL. As seen in Figure 6.21, P-LCOL had poor and unreliable error metrics. However, both LCOL and BCOL had similar and reasonable error trends. This P-LCOL error behavior and the potential for implementing some constraints linearly, as discussed in Section 6.4.2, makes BCOL the preferred well-conditioned pseudospectral method for optimal control presented here.

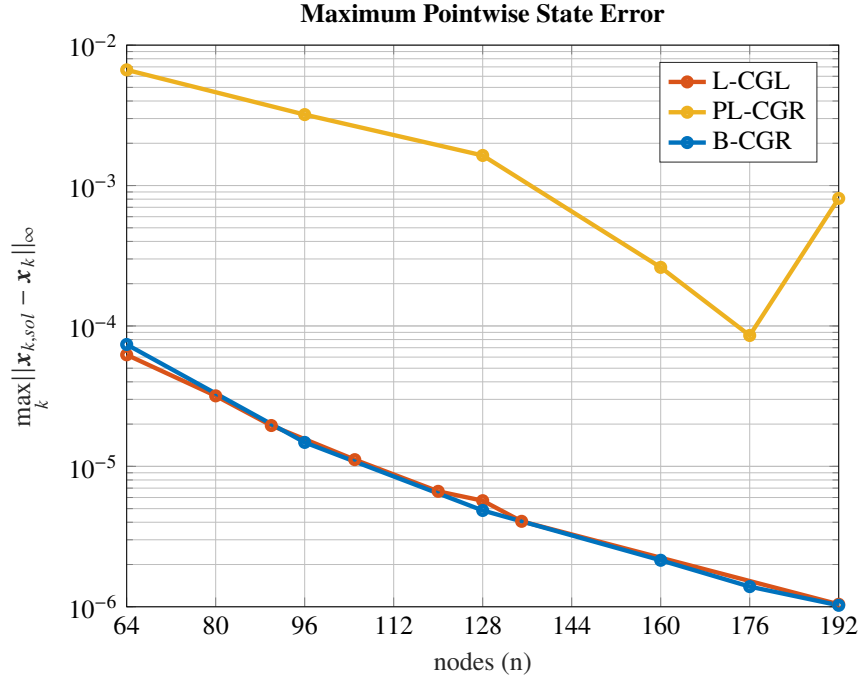


Figure 6.21. Maximum pointwise error when using Knitro to solve BP2. Error is measured against *ode45*-propagated states when using solution controls.

The well-conditioned BCOL method achieved similar maximum pointwise error to LCOL, but the nonlinear function evaluations required to compute this solution with Knitro were reduced by nearly an order of magnitude for a problem with less than 192 nodes. See Figure 6.22. This performance gap is highly encouraging for even low-node applications. Notably, P-LCOL solutions did not achieve a local optimal exit flag from Knitro, but instead stopped because the relative change in solution estimate was less than step tolerance for 3 consecutive iterations. Thus, by considering Figure 6.21 and Figure 6.22, one realizes that P-LCOL solutions required a significant number of function evaluations. If step tolerance had been increased, pointwise error may have improved, but only at the expense of more function evaluations.

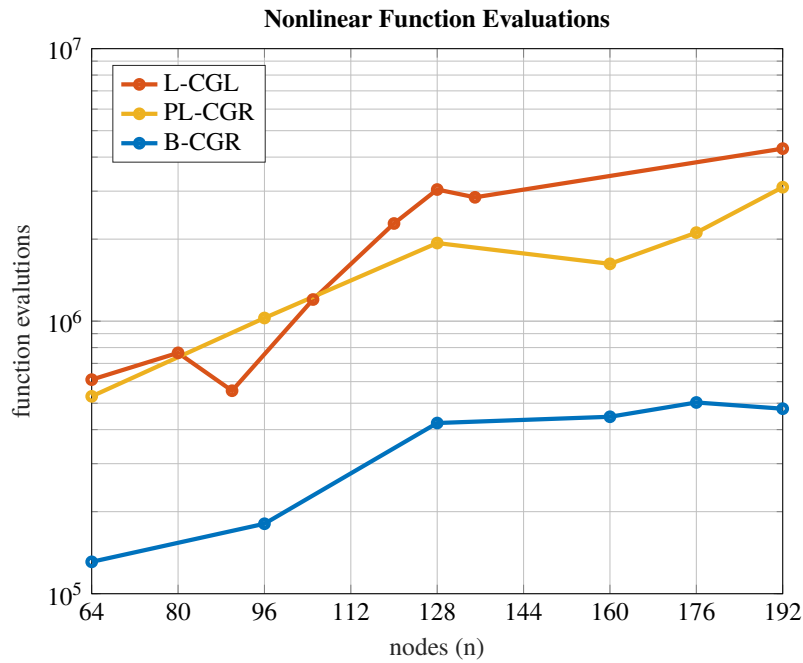


Figure 6.22. Evaluations of the BP2 nonlinear function needed for Knitro solution.

The repeatable performance metrics in Figure 6.22 can also be seen in computational time cost. Although this time is implementation dependent, it is instructive to see relative trends as shown in Figure 6.23.

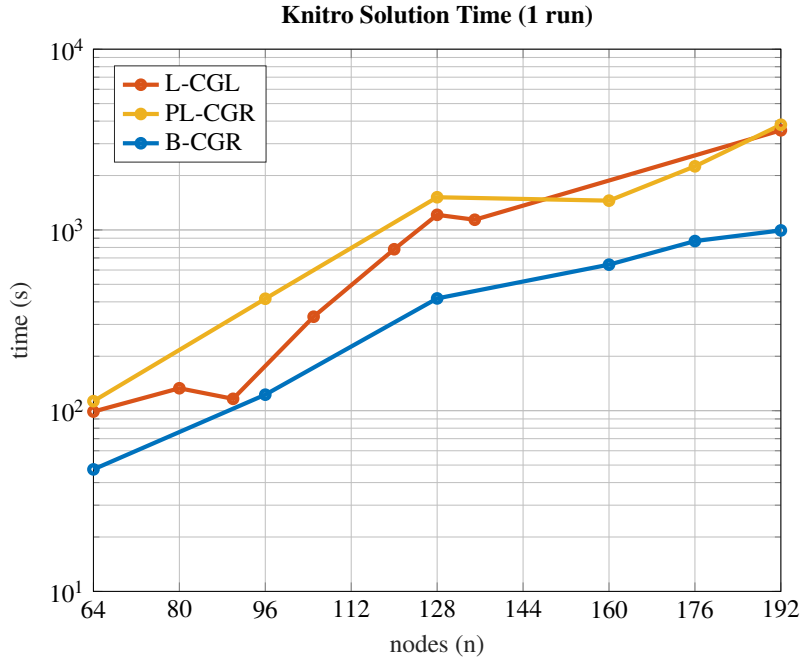


Figure 6.23. Knitro solution time for BP2 using different collocation schemes.

Note that in Figure 6.23, this order of magnitude difference in function evaluations results in a 192-node solution that takes an hour for LCOL and roughly 15 minutes for BCOL.

6.6 High Resolution Problems

In order to extend the basic satellite problems—BP1 and BP2—to applications beyond the reach of LCOL methods, high resolution variations to BP1 can be demonstrated. In this section, we will explore lowering the thrust and increasing the orbital radius. Both of these adjustments to BP1 result in optimal control problems that require many more nodes to capture the oscillatory nature of the thrust vector as the satellite makes small adjustments to its orbit. By choosing BP1, these differences in resolution can also be seen through the increasing minimum time of the optimal control solution. In a change from BP1 and BP2 results and simply as a choice for exhibition, these problems—HR1 to HR4—are shown using CGL nodes.

6.6.1 Lower Thrust: HR1 and HR2

By lowering the thrust, the satellite can no longer achieve a doubling of the original orbit over the same time frame. A solution necessarily requires more time to affect the same accelerations on the satellite to achieve progressively larger orbital radii. First, HR1 takes the thrust down an entire order of magnitude. Thus, HR1 makes the single adjustment to BP1 by having a constant acceleration of $A = 0.001$. This value represents a 100mN thruster on a 100kg satellite. As seen in Figure 6.24, the solution requires a significant number of orbits to achieve the same parameters final state as BP1.

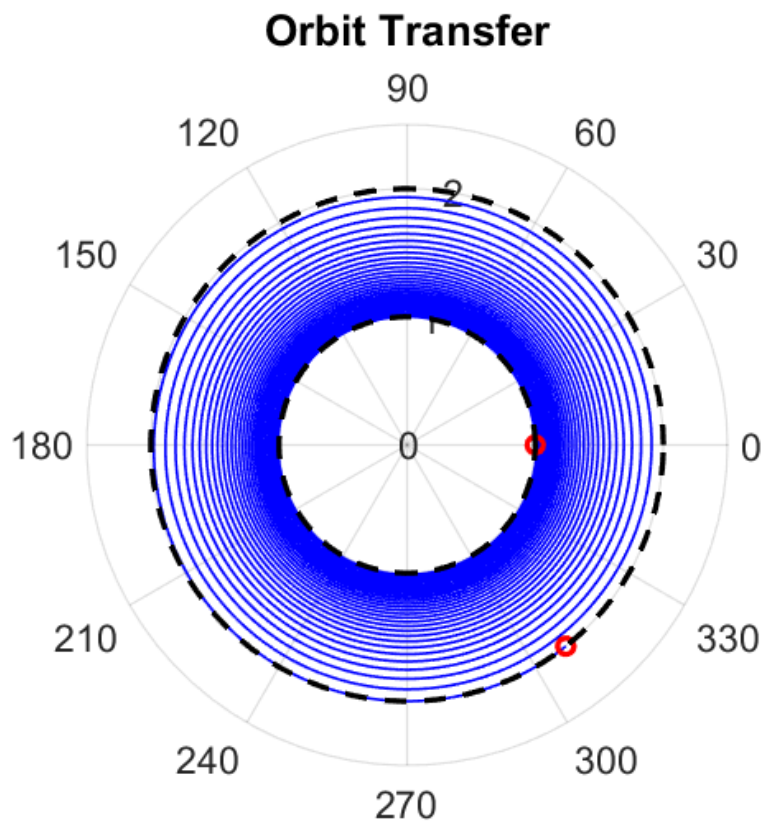


Figure 6.24. State trajectory for HR1: the intermediate-thrust ($A = 0.001$) adaption of BP1. Solution was found with 320 nodes via B-CGL.

The final solution used 320 nodes to resolve almost 30 revolutions of the Earth. Due to an increasing orbital radius, this meant the overall duration was 293 TU. For an original orbital altitude of 660 km this equates to 478 hours or almost 20 days. Figure 6.25 demonstrates

how dense the resulting control oscillations start to become in order to resolve the dynamics.

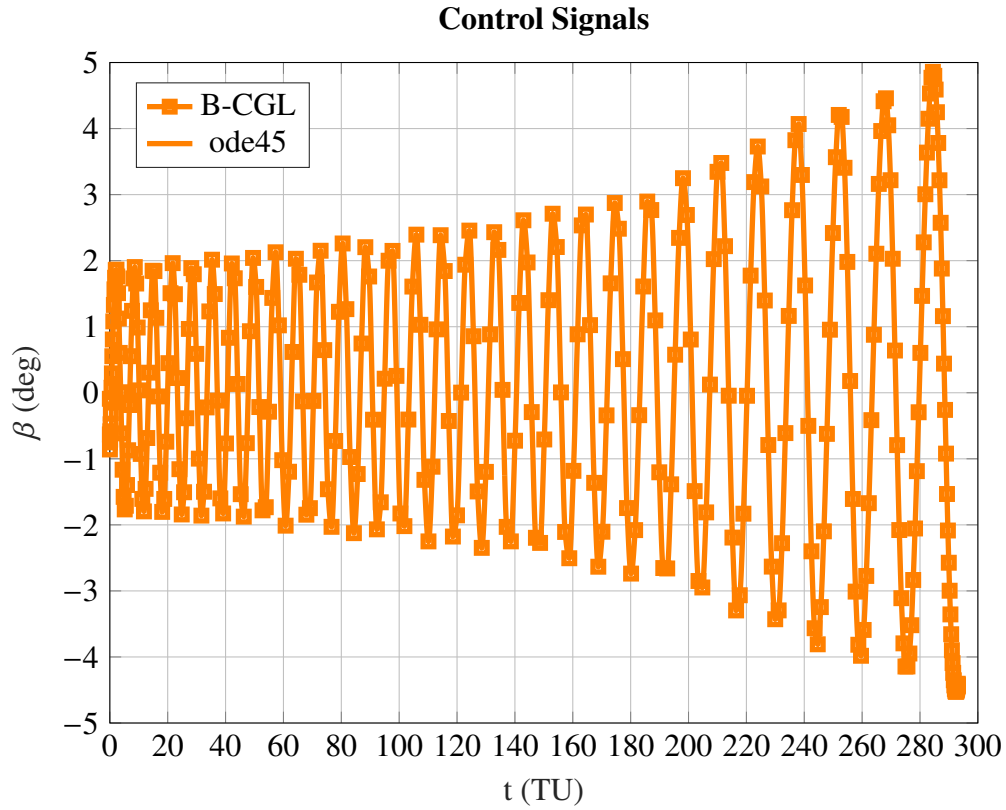


Figure 6.25. Intermediate-thrust ($A = 0.001$) control solution for HR1 using 320 nodes and the B-CGL method.

The extent of these optimal controls start to converge toward nearly tangential thrust vectors that oscillate direction according to orbital position. Now the maximum β is only 5° as opposed to nearly 35° in the case of BP1. However, the overall controls continue to oscillate with increasing amplitude over the course of the orbital transfer, as expected from an expanding orbital radius.

In order to highlight this trend further, HR2 demonstrates another halving of the thrust level. In HR2, $A = 5 \times 10^{-4}$, which represents a 100mN thruster on a 200kg satellite. This begins to approach the feasible range of high specific impulse thrusters on a small, but not insignificant satellite capable of generating requisite power levels. For HR2, a 512-node solution is shown both to exhibit the capability of BCOL with increasing nodes. Even 320

nodes do not resolve these dynamics enough for Knitro to converge on a solution.

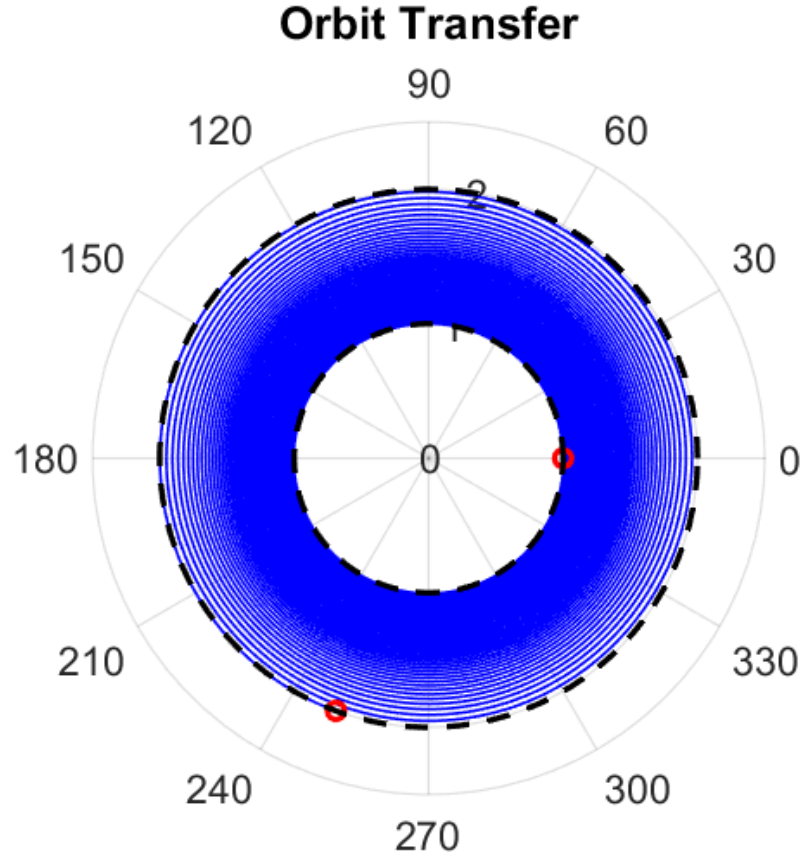


Figure 6.26. State trajectory for HR2: the low-thrust ($A = 5 \times 10^{-4}$) adaption of BP1. Solution was found with 320 nodes via B-CGL.

Figure 6.26 shows a solution with nearly 60 revolutions of the Earth and an overall duration was 586 TU. With the benchmark 660 km altitude start, this equates to 955 hours or almost 40 days. Figure 6.27 demonstrates how dense the resulting control oscillations start to become in order to resolve the dynamics. Maximum pointwise error relative to the *ode45*-propagated solution in the radial position is 1.43×10^{-4} and are on the order of 5×10^{-5} for velocities.

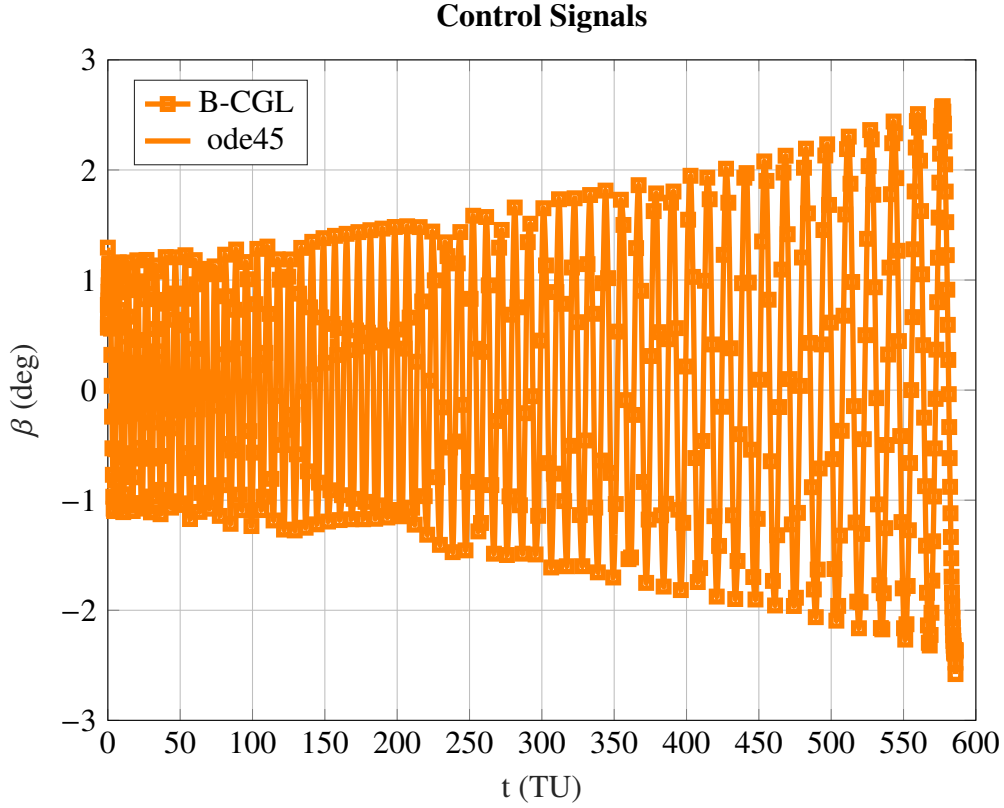


Figure 6.27. Very low-thrust ($A = 5 \times 10^{-4}$) control solution to the a minimum-time problem similar to BP1 using 512 nodes and the B-CGL method.

Note that due the endpoint clustering exhibited by the Gauss–Lobatto nodes, the first and last 50 TU segments of controls show a smooth extremal trend line. When these solutions begin to have fewer nodes near the center, there are clear notches in this trend line that can produce higher errors.

6.6.2 In-Plane Transfer to GEO: HR3

Higher resolution can also be motivated by extending the duration by transferring the low-Earth orbit satellite to geosynchronous orbit. This is problem HR3. For HR3, two changes are made to BP1. The thrust is lower so that $A = 0.005$ and the final state is set to

$$\bar{\mathbf{x}}_f = \begin{bmatrix} 6 & \theta^F & 0 & \sqrt{\frac{1}{6}} \end{bmatrix}^T. \quad (6.60)$$

This final state represents a 6-fold increase in the orbital radius of the original orbit. As defined in Section 3.7.3, the original orbit had an approximate altitude of 660 km and radius 7030 km. By definition, this radius defined the canonical distance unit and represents 1 DU. At 6 DU, or 42,157 km, the final conditions are therefore an in-plane geosynchronous orbit. Given an initial inclination of 0° , these in-plane maneuvers would achieve geostationary orbit. The solution trajectory is shown in Figure 6.28.

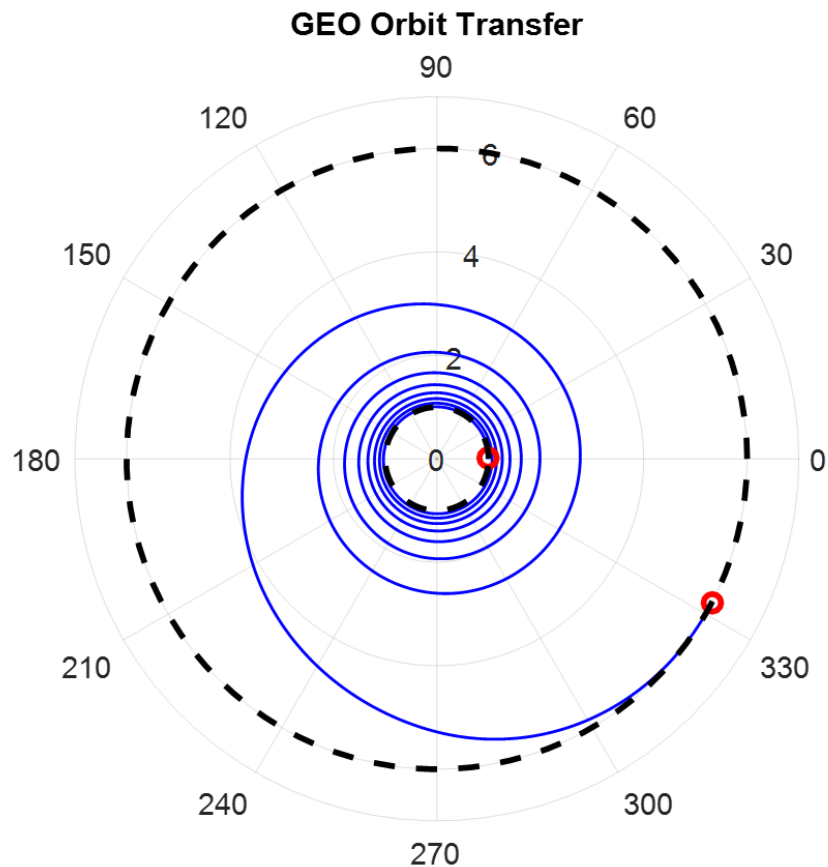


Figure 6.28. State trajectory for HR3 using 352 nodes via B-CGL.

This solution uses 352 nodes—more than necessary, but only takes an hour to converge to tolerance. Maneuvers occur over 129 TU that results in 8 revolutions of the Earth. The excess in nodes, allows the control trajectory to be resolved into smooth curves. See Figure 6.29.

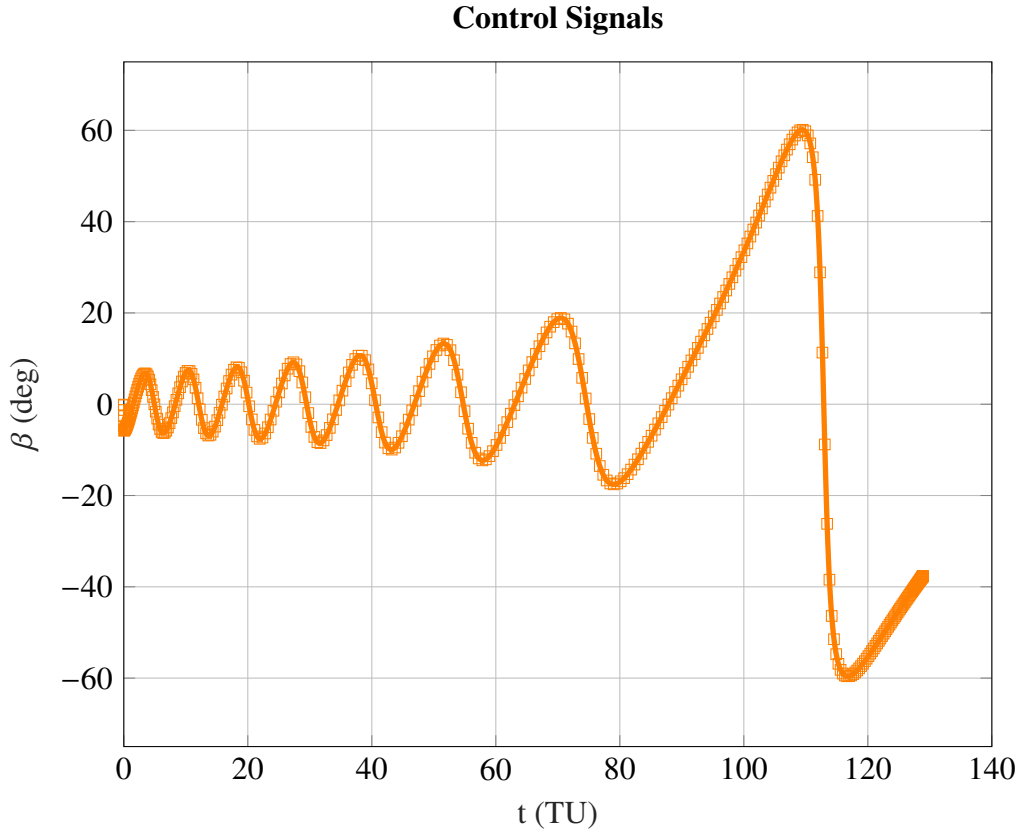


Figure 6.29. The control solution for HR3 ($A = 0.005$) using 352 nodes and the B-CGL method.

The resulting state trajectories—shown in Figures 6.30-6.31—produce average pointwise errors on the order of 10^{-6} . Note that the usual maximum pointwise error occurs in the angular position, which tends to accrue error throughout the orbital transfer. As can be seen, improvements to pointwise error can be achieved by taking a lower-node solution and seeding a higher-node run. This can also be done for slight variations in the problem to achieve faster convergence, as was done to achieve the results for HR4.

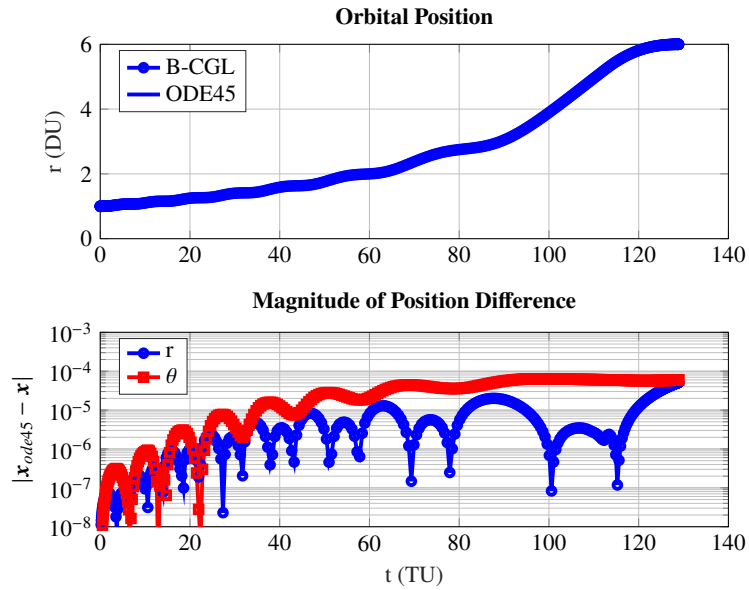


Figure 6.30. Position and corresponding error for 352-node B-CGL solution for HR3 ($A = 0.005$). Error relative to *ode45*-propagated solution using *pchip* to interpolate controls.

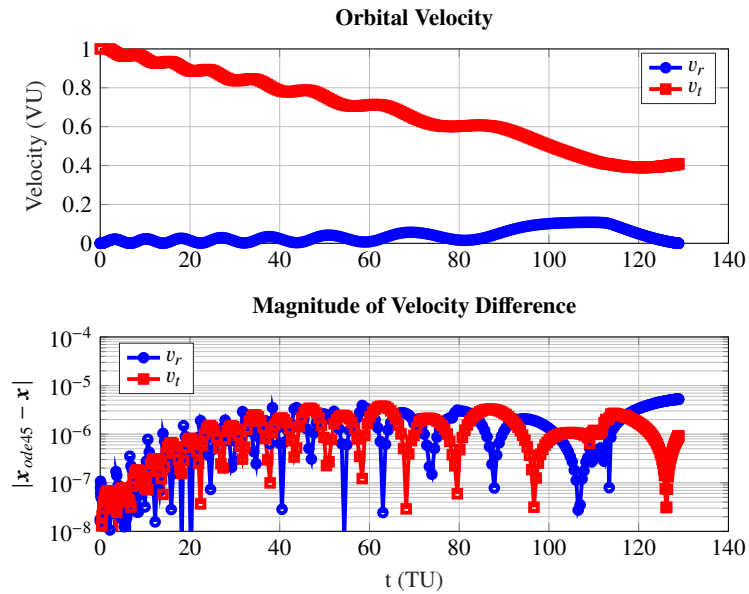


Figure 6.31. Velocity and corresponding error for 352-node B-CGL solution for HR3 ($A = 0.005$). Error relative to *ode45*-propagated solution using *pchip* to interpolate controls.

Finally, HR4 introduces a lower thrust of $A = 0.001$ to the transfer to GEO problem specified by HR3. In this case, the same 352 node CGL discretization is much more necessary to resolve the oscillations. A visual of the transfer is shown in Figure 6.32.

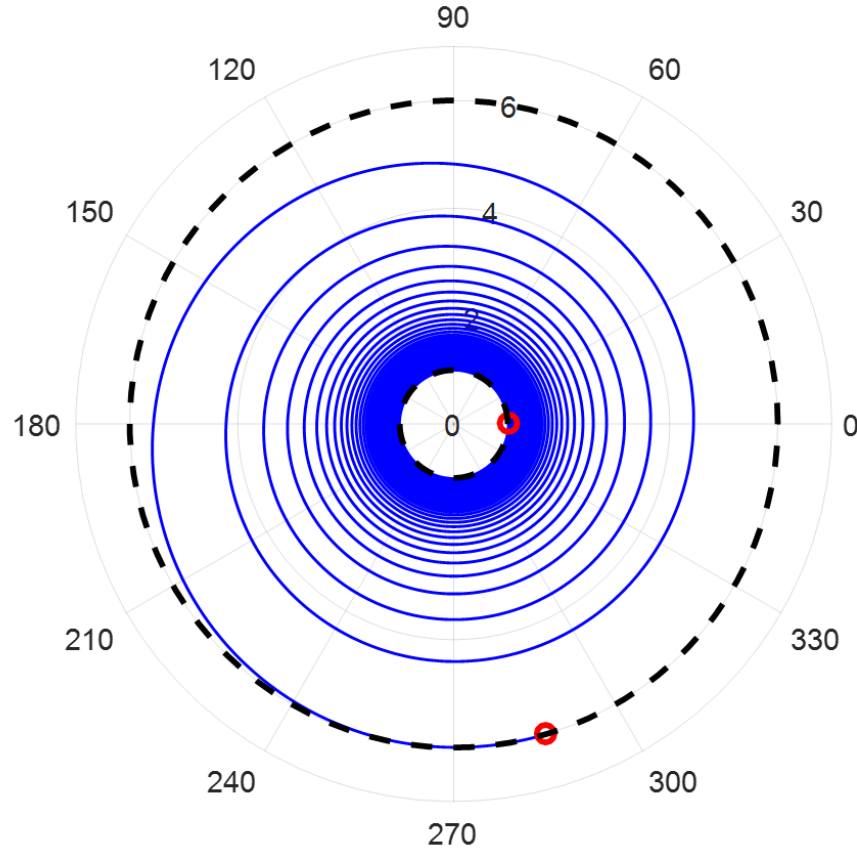


Figure 6.32. State trajectory for HR4 using 352 nodes via B-CGL.

This solution uses 352 nodes and was seeded with the HR3 solution in order to converge quicker. The entire orbit transfer results in roughly 39 revolutions of the Earth. These revolutions require a significant number of nodes to properly interpolate all of the control oscillations, seen in Figure 6.33, and determine the optimal means of expanding and circularizing the orbit.

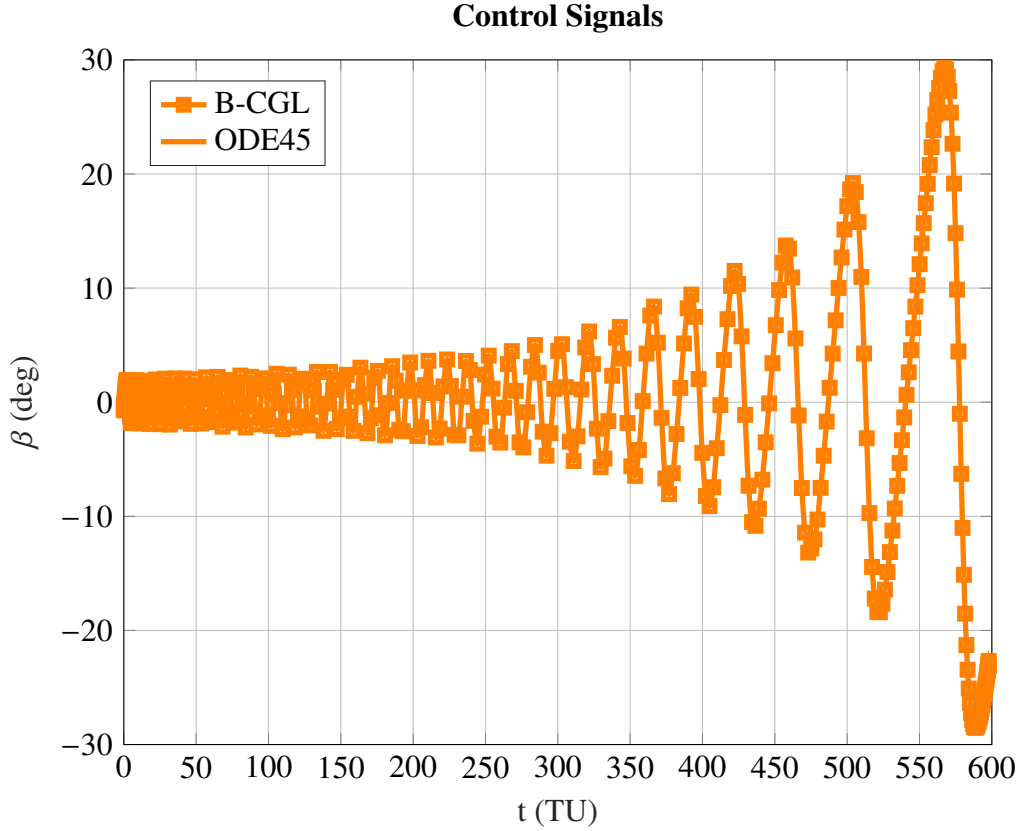


Figure 6.33. Intermediate-thrust ($A = 0.001$) control solution for HR4 using 352 nodes and the B-CGL method.

6.6.3 Performance

These high-resolution problems required significant computation time to achieve solutions due to the lack of more detailed implementation and the continued reliance on a trivial guess. Sparsity patterns and analytical Jacobians would improve these results, but would also obscure a means of comparative performance to BP1. To this end, the 512-node HR2 solution took 10 hours to compute. In comparison, the 320-node HR1 solution took only 2 hours and the 128-node BP1 solution took 3 minutes.

6.7 Method Analysis

Arguments could be made for achieving better results: specifying sparsity matrices, providing Hessian information, or a litany of other implementation tweaks. However, these results

attempt to isolate the effect of only changing the collocation method on the performance of the nonlinear solver. The clear takeaway is that BCOL performs at or better than LCOL in terms of error and that nonlinear solvers can take advantage of the well-conditioned PSIM to achieve nearly an order of magnitude faster results. This clear advantage is achieved even with a fully nonlinear implementation. Being able to implement $\mathbf{X} = \mathbf{BV}$ as a linear constraint, this gap may indeed grow.

CHAPTER 7:

Summary and Future Work

This thesis provides a proof of concept for well-conditioned optimal control collocation methods based on Birkhoff interpolation bases. The adjustments to linear formulations described within and the means of implementation are based on the nature of optimal control as a constrained optimization problem for a nonlinear dynamic system of differential algebraic equations. This application differs from the bulk of literature on well-conditioned pseudospectral methods, which focus more on linear applications. Some insights are:

- The well-conditioned nature of Birkhoff collocation (BCOL) appear to remain when applied to the nonlinear constrained optimization framework: Figures 6.14 and 6.22.
- The dynamic system is not only a function of state, but also controls. Controls, $\mathbf{u}$, present an additional constraint requirement: (6.15) and (6.37).
- Time-free problems require consideration if one pursue splitting BCOL into nonlinear and linear constraints within the NLP: Section 6.4.2.

Although, simple engineering applications are demonstrated, the mathematical foundations shown here can be extended to more complex optimal control problems without expectation of concern. With improvements to implementation, well-conditioned optimal control can be applied to thousands of nodes. In this thesis, no Jacobian sparsity patterns were provided to the nonlinear solver and most parameters for Knitro were left at or near default values. Likewise, the initial guess for the solver was given as a continuation of the initial state.

These basic approaches to solving the problem helped to create baseline performance metrics that would be less apparent with a better initial guess. Regardless, a 512-node solution to a low-thrust, circle-to-circle minimum time orbit transfer problem. Despite the simplistic dynamics, these low-thrust problems are the first steps in filling a need for true optimal control in an industry still run on heuristics. Ultimately, if these methods are combined with non-smooth calculus methods [13], solutions to the satellite collection planning problem can be solved over several orbits. All of these avenues of investigation can be seen as future work that builds off this proof-of-concept foundation for well-conditioned optimal control.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: Formulations of **B**

A.1 First-Order: **B**₁

A.1.1 Initial Value Problem

The initial value construct determines that $B_0(\tau) = 1$. All other Birkhoff polynomials are defined by

$$\begin{aligned} B_j(-1) &= 0 \\ B_j(\tau) &= \sum_{k=0}^{N-1} \alpha_{kj} \frac{\partial^{-1} P_k^{(\alpha, \alpha)}(\tau)}{\gamma_k} \\ \alpha_{kj} &= w_j (P_k^{(\alpha, \alpha)}(\tau_j) - (-1)^{N+k} P_N^{(\alpha, \alpha)}(\tau_j)) \end{aligned}$$

where $1 \leq j \leq N$ and $P_k^{(\alpha, \alpha)}$ applies to both P_k or T_k . For the IVP, $\mathbf{v}$ is defined such that

$$\mathbf{v} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}_1 \\ \vdots \\ \dot{x}_N \end{bmatrix}.$$

A.1.2 Final Value Problem

The final value construct determines that $B_N(\tau) = 1$. All other Birkhoff polynomials are defined by

$$\begin{aligned} B_j(1) &= 0 \\ B_j(\tau) &= \sum_{k=0}^{N-1} \frac{\alpha_{kj}}{\gamma_k} \left(\partial^{-1} P_k^{(\alpha, \alpha)}(\tau) + \partial^{-1} P_k^{(\alpha, \alpha)}(1) \right) \\ \alpha_{kj} &= w_j (P_k^{(\alpha, \alpha)}(\tau_j) - P_N^{(\alpha, \alpha)}(\tau_j)). \end{aligned}$$

where $0 \leq j \leq N-1$ and $P_k^{(\alpha, \alpha)}$ applies to both P_k or T_k . For the FVP, $\mathbf{v}$ is defined such that

$$\mathbf{v} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} \dot{x}_0 \\ \dot{x}_1 \\ \vdots \\ x_N \end{bmatrix}.$$

A.1.3 Imperfect Partial Boundary Value Problem

Similar to the IVP, the initial value construct determines that $B_0(\tau) = 1$. However, now the last node is no longer used such that $B_j(1) = 0 \quad \forall j$.

$$\begin{aligned} B_j(-1) &= 0 \\ B_j(\tau) &= \sum_{k=0}^{N-2} \alpha_{kj} \frac{\partial^{-1} P_k^{(\alpha, \alpha)}(\tau)}{\gamma_k} \\ \alpha_{kj} &= w_j \left(T_k(\tau_j) - \frac{1 - (-1)^{N+k} T_{N-1}(\tau_j)}{2} - \frac{1 + (-1)^{N+k} T_N(\tau_j)}{2} \right) \end{aligned}$$

where $0 \leq j \leq N-2$ and $P_k^{(\alpha, \alpha)}$ applies to both P_k or T_k .

A.2 Second-Order: B_2

The expansion coefficients are:

$$\begin{aligned} B_j(\tau) &= \sum_{k=0}^{N-2} \frac{\beta_{kj}}{\gamma_k} \left(\partial_\tau^{-2} P_k^{(\alpha, \alpha)}(\tau) + \frac{1 + \tau}{2} \partial_\tau^{-2} P_k^{(\alpha, \alpha)}(1) \right) \\ \beta_{kj} &= w_j \left(P_k^{(\alpha, \alpha)}(\tau_j) - \frac{1 - (-1)^{N+k}}{2} P_{N-1}^{(\alpha, \alpha)}(\tau_j) - \frac{1 + (-1)^{N+k}}{2} P_N^{(\alpha, \alpha)}(\tau_j) \right) \end{aligned}$$

where $1 \leq j \leq N-1$ and $P_k^{(\alpha, \alpha)}$ applies to both P_k or T_k . Slight simplifications of the constant of integration result in different expressions for the polynomials:

Chebyshev (CGL)

$$B_j(\tau) = \sum_{k=0}^{N-2} \left[\frac{\beta_{kj}}{\gamma_k} \partial^{-2} T_k(\tau) + \partial^{-2} T_k(1) \left(\frac{\tau+1}{2} \right) \right]$$

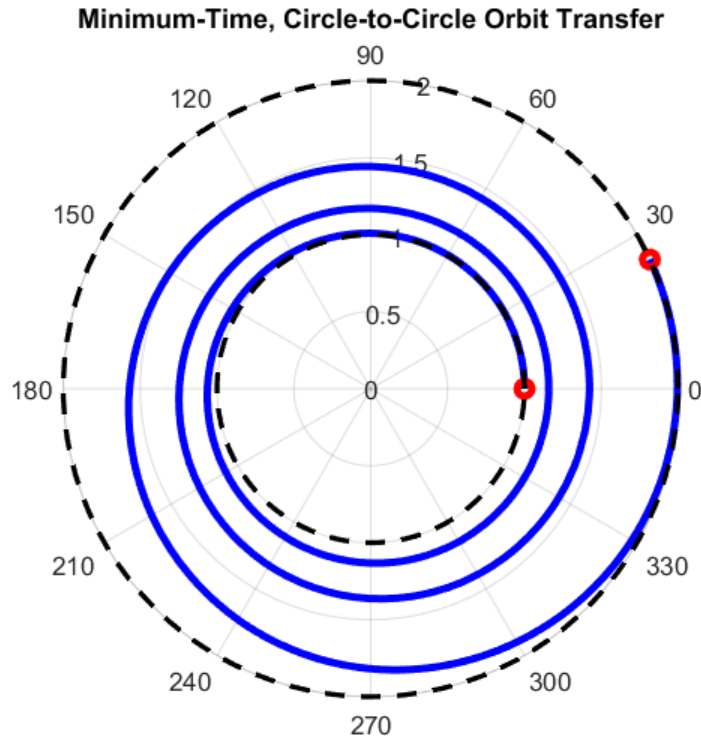
Legendre (LGL)

$$B_j(\tau) = (\beta_{1j} - \beta_{0j}) \frac{\tau+1}{2} + \sum_{k=0}^{N-2} \left(\beta_{kj} \frac{\partial^{-2} P_k(\tau)}{\gamma_k} \right)$$

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: Example Problems

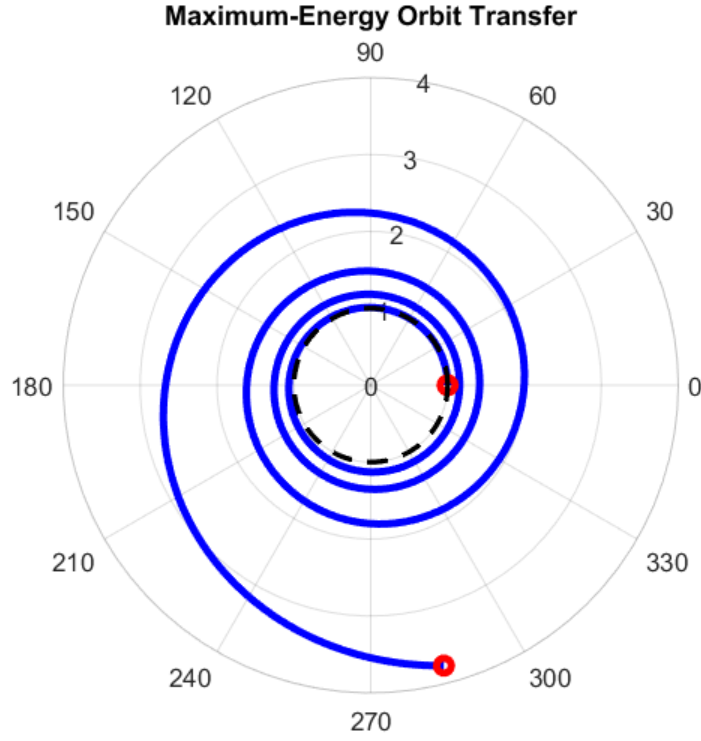
B.1 BP1: Minimum-Time Orbit Transfer



$$\text{(BP1)} \left\{ \begin{array}{ll} \text{Minimize} & J(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}) = \bar{t}_f \\ \text{Subject to:} & \\ & \dot{\bar{\mathbf{x}}} = \bar{\mathbf{f}}_{BP}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t), t) \\ & \bar{\mathbf{x}}_0 = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \\ & \bar{\mathbf{x}}_f = \begin{bmatrix} 2 & \theta^F & 0 & \sqrt{\frac{1}{2}} \end{bmatrix}^T. \end{array} \right. \quad \text{(B.1)}$$

For further details see Section 3.8.

B.2 BP2: Maximizing Orbital Energy



$$\text{(BP2)} \left\{ \begin{array}{l} \text{Minimize} \quad J(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}) = \frac{1}{\bar{r}_f} - \frac{\bar{v}_{r,f}^2 + \bar{v}_{t,f}^2}{2} \\ \text{Subject to:} \\ \quad \dot{\bar{\mathbf{x}}} = \bar{\mathbf{f}}_{BP}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t), t) \\ \quad \bar{\mathbf{x}}_0 = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \\ \quad \bar{t}_f = 50 \end{array} \right. \quad (\text{B.2})$$

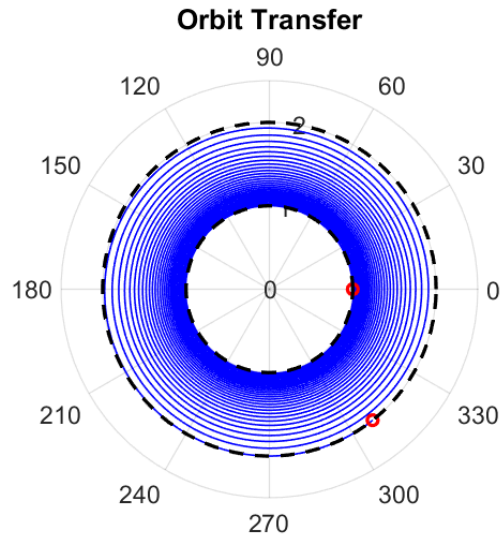
For further details see Section 3.9.

B.3 Lower-Thrust, Minimum-Time Orbit Transfers

In both HR1 and HR2, the BP1 formulation applies except for a given change to the constant acceleration, A , in $\bar{\mathbf{f}}_{BP}$.

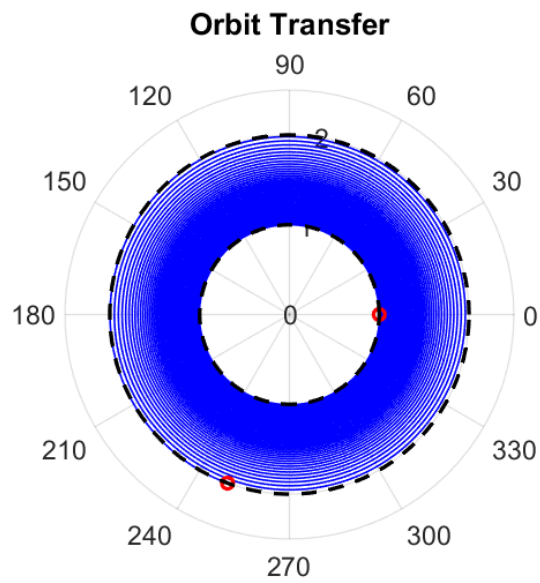
B.3.1 HR1: Intermediate Thrust

Intermediate thrust is $\frac{1}{10}$ the original in BP1 and BP2: $A = 10^{-3}$.



B.3.2 HR2: Low Thrust

Low thrust is $\frac{1}{20}$ the original: $A = 5 \times 10^{-4}$.



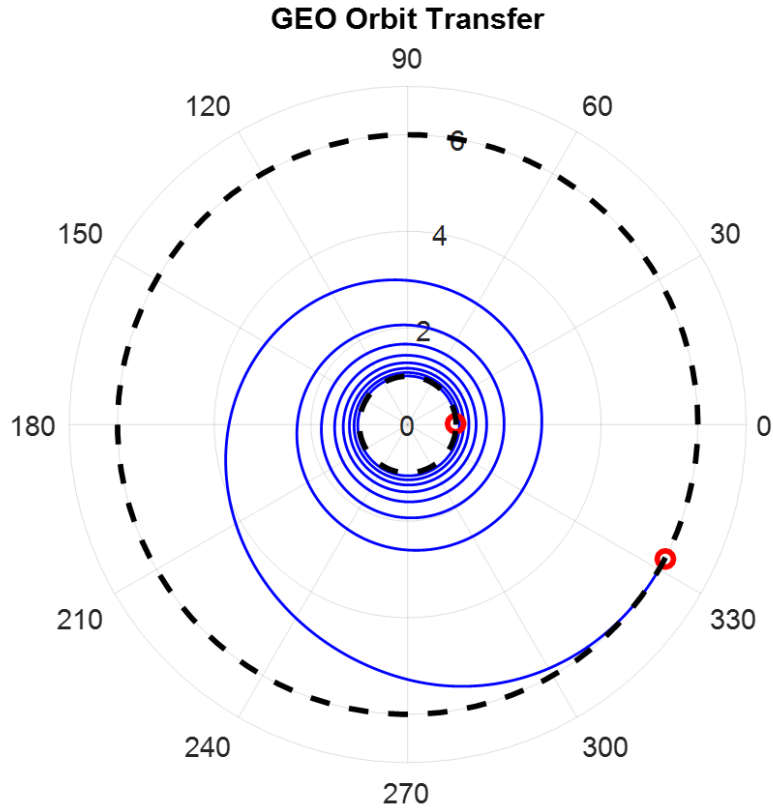
B.4 Minimum-Time, LEO to GEO Orbit Transfers

In both HR2 and HR3, the BP1 formulation applies except two changes: the thrust, A , in $\bar{f}_{BP}$ and the final state

$$\bar{\mathbf{x}}_f = \begin{bmatrix} 6 & \theta^F & 0 & \sqrt{\frac{1}{6}} \end{bmatrix}^T. \quad (\text{B.3})$$

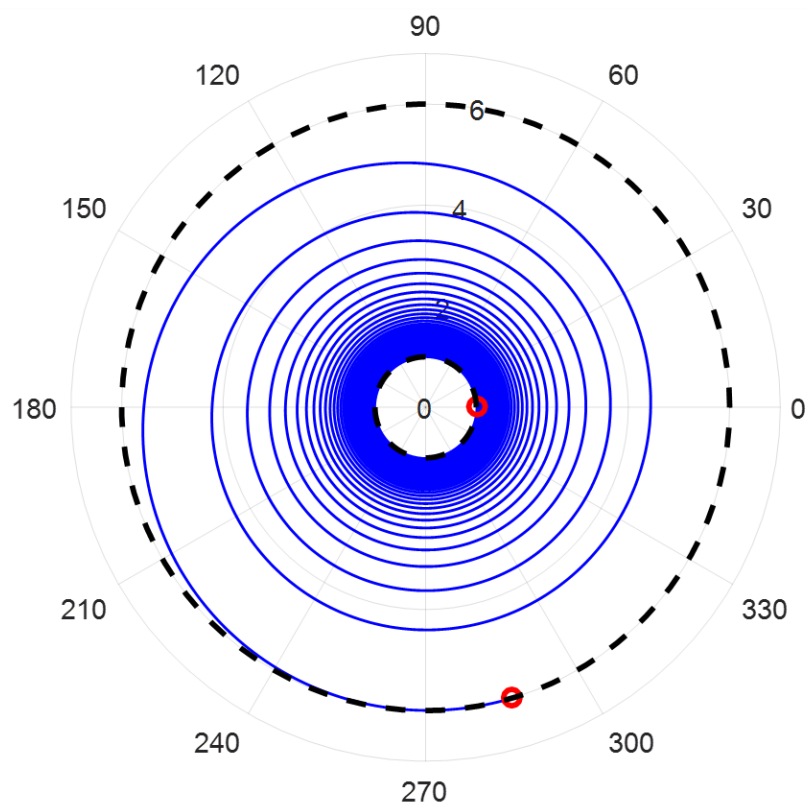
B.4.1 HR3: High Thrust

High thrust is still $\frac{1}{2}$ the original in BP1 and BP2: $A = 0.005$.



B.4.2 HR4: Low Thrust

Low thrust is $\frac{1}{10}$ the original: $A = 0.001$.



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C:

Notation

Table C.1. Summary of optimal control variables

Symbol	Description
t	Physical time
τ	Computational time
$\mathbf{x}$	State variables
$\mathbf{u}$	Control variables
$\mathbf{e}$	Endpoint conditions
$\mathbf{h}$	Path constraints
J	Cost function
E	Endpoint cost
F	Running cost
$\mathcal{H}$	Hamiltonian
λ	Costate variables
μ	Path covectors

Table C.2. Various measures of dimensionality

Symbol	Description	Formula
i	Matrix row index	
j	Matrix column index	
N	Number of discretization divisions	$0 \leq i \leq N$
n	Number of nodes	$n = N + 1$
p	Problem order	$\mathbf{B}_p$
(k)	Operator differential	$\mathbf{D}^{(k)} = \mathbf{D}^k$

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

- [1] L.-L. Wang, M. D. Samson, and X. Zhao, “A well-conditioned collocation method using a pseudospectral integration matrix,” *SIAM Journal on Scientific Computing*, vol. 36, no. 3, pp. A907–A929, 2014.
- [2] M. Karpenko, S. Bhatt, N. Bedrossian, and I. M. Ross, “Flight results for a revolutionary approach to maximizing spacecraft imaging capability,” in *International Symposium on Space Flight Dynamics*, 2012.
- [3] N. S. Bedrossian, S. Bhatt, W. Kang, and I. M. Ross, “Zero-propellant maneuver guidance,” *IEEE Control Systems*, vol. 29, no. 5, 2009.
- [4] “DIDO optimal control software,” [Online]. Available: https://www.mathworks.com/products/connections/product_detail/dido.html. Accessed Jun. 1, 2018.
- [5] J. S. Hesthaven, “Integration preconditioning of pseudospectral operators. I. Basic linear operators,” *SIAM Journal on Numerical Analysis*, vol. 35, no. 4, pp. 1571–1593, 1998.
- [6] E. M. Elbarbary, “Integration preconditioning matrix for ultraspherical pseudospectral operators,” *SIAM Journal on Scientific Computing*, vol. 28, no. 3, pp. 1186–1201, 2006.
- [7] S. Olver and A. Townsend, “A fast and well-conditioned spectral method,” *SIAM Review*, vol. 55, no. 3, pp. 462–489, 2013.
- [8] K. Du, “Preconditioning rectangular spectral collocation,” *Math arXiv preprint*, Oct. 2015, [Online]. Available: <http://arxiv.org/abs/1510.00195>. Accessed Jun. 1, 2018.
- [9] I. Ross, R. Proulx, J. Greenslade, and M. Karpenko, “Dynamic optimization for satellite image collection,” in *AAS/AIAA Space Flight Mechanics Conference*, no. 16-260, 2016, pp. 14–18.
- [10] I. M. Ross and F. Fahroo, “Pseudospectral knotting methods for solving nonsmooth optimal control problems,” *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 3, pp. 397–405, 2004.
- [11] J. Albiach, J. M. Sanchis, and D. Soler, “An asymmetric TSP with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation,” *European Journal of Operational Research*, vol. 189, no. 3, pp. 789–802, 2008.

- [12] I. M. Ross, Q. Gong, and P. Sekhavat, “Low-thrust, high-accuracy trajectory optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 921–933, 2007.
- [13] I. M. Ross, M. Karpenko, and R. J. Proulx, “A nonsmooth calculus for solving some graph-theoretic control problems,” *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 462–467, 2016.
- [14] I. M. Ross and M. Karpenko, “A review of pseudospectral optimal control: From theory to flight,” *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.
- [15] J. Shen, T. Tang, and L.-L. Wang, “Orthogonal Polynomials and Related Approximation Results,” in *Spectral Methods* (Springer Series in Computational Mathematics). Springer, Berlin, Heidelberg, 2011, pp. 47–140.
- [16] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb, *Spectral Methods for Time-Dependent Problems* (Cambridge Monographs on Applied and Computational Mathematics). Cambridge, U.K.: Cambridge University Press, Jan. 2007.
- [17] I. M. Ross, *A Primer on Pontryagin’s Principle in Optimal Control : Second edition*. Carmel, CA, U.S.: Collegiate Publishers, 2015.
- [18] I. M. Ross and F. Fahroo, “A Perspective on Methods for Trajectory Optimization,” in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit* (Guidance, Navigation, and Control and Co-located Conferences). American Institute of Aeronautics and Astronautics, Aug. 2002.
- [19] U. Ascher and C. Greif, *A First Course on Numerical Methods*. Vancouver, BC, Canada: SIAM Computational Science and Engineering, 2011.
- [20] J.-P. Berrut and L. N. Trefethen, “Barycentric lagrange interpolation,” *SIAM Review*, vol. 46, no. 3, pp. 501–517, 2004.
- [21] “National Institute of Standards and Technology (NIST) Digital Library of Mathematical Functions (DLMF) Chapter 18: Orthogonal Polynomials,” [Online]. Available: <http://dlmf.nist.gov/18>. Accessed Jun. 1, 2018.
- [22] P. J. Davis, *Interpolation and Approximation* (Dover Books on Mathematics). Mineola, NY, U.S.: Dover Publications, 1975.
- [23] G. Szeg, *Orthogonal Polynomials* (American Mathematical Society: Colloquium publication). Providence, RI, U.S.: American Mathematical Society, Dec. 1939, no. v. 23.

- [24] “Artelys Knitro MATLAB reference,” 2018, [Online]. Available: https://www.artelys.com/tools/knitro_doc/3_referenceManual/knitromatlabReference.html. Accessed Jun. 1, 2018.
- [25] T. A. Driscoll and N. Hale, “Rectangular spectral collocation,” *IMA Journal of Numerical Analysis*, vol. 36, no. 1, pp. 108–132, 2015.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California