AEROSPACE MISSION DESIGN ON QUOTIENT MANIFOLDS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Michael J. Sparapany

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Michael J. Grant, Chair

School of Aeronautics and Astronautics

Dr. James M. Longuski

School of Aeronautics and Astronautics

Dr. William A. Crossley

School of Aeronautics and Astronautics

Dr. Michael A. Bolender

AFRL Wright-Patterson AFB

Approved by:

Dr. Gregory A. Blaisdell

Head of the School Graduate Program

ACKNOWLEDGMENTS

I would like to thank a number of individuals who have provided support in the course of my research. First, I want to recognize my parents, John and Ann, and my sister, Nichole, who have been constant sources of encouragement. All of my accomplishments can be attributed to the time and energy they invested in me throughout my life. I would also like to thank my wife Haley Snell-Sparapany whose positive outlook, enthusiasm, and competitive spirit has always inspired me to aim high and achieve my goals.

I owe much gratitude to my advisor Professor Michael Grant, whose meticulous instruction has been instrumental to this research. I am grateful for all of the tools and knowledge he has given me, and I know it will serve me well in the future. I also would like to extend my gratitude to my committee members Professor James Longuski and Professor William Crossley, whose classes and superb teaching styles were vital to the foundation of my research. Dr. Bolender deserves special recognition. This dissertation is the ultimate result of his thorough insight and probing questions. I would also like to show appreciation for the rest of the staff and faculty in the School of Aeronautics and Astronautics at Purdue for providing support for all aspiring Aeronautics and Astronautics students.

I would specifically like thank Thomas Antony and Sean Nolan for their valuable insight, collaboration, and recommendations in regards to this research topic. Kshitij Mall, Harish Saranathan, and everyone else in my research group should also be accredited for their contributions and constructive discussions on my work.

I couldn't forget my gymnastics friends Kyle Thackston, Clay Thomas, Tim Michaels, and Sarah Percival for providing the proof that athletics and academics are homeomorphic. Bundle squad!

Finally, I would like to acknowledge the Air Force Research Laboratory, Eglin, AFB and Leidos, Reston, VA for providing financial backing. Moreover, I want to thank my mentors Dr. Crystal Pasiliao and Dr. Daniel Reasor for their guidance. My research was also funded by a Teaching Assistantship through the Purdue School of Engineering Education and would especially like to thank Dr. Mary Pilotte, Lynn Hegewald, Jill Folkerts, Anne DeLion, Jim Whitford, Rick Womack, and the rest of the support team over their years, as well as Dr. Michele Strutz and all other instructors for their support.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

SYMBOLS xii

ABSTRACT xv

1 INTRODUCTION 1
1.1 State-of-the-Art Methods 3
1.1.1 Direct Methods 3
1.1.2 Indirect Methods 7
1.1.3 Hybrid Methods 9
1.2 Foundations of Modern Indirect Trajectory Optimization 11
1.2.1 Calculus of Variations 13
1.2.2 Dualization 19
1.2.3 Nonlinear Programming 24
1.3 End-to-End Mission Design 27
1.4 Overview 30

2 ALGEBRAS 33
2.1 Tensor Algebra 34
2.2 Grassmann Algebra 35
2.3 Lie Algebra 37
2.4 Co-algebras 38
2.5 Differential Graded Algebra 39
2.6 Summary 41

3 GEOMETRIC OPTIMAL CONTROL THEORY 42
3.1 Differential Geometry 42
3.1.1 Topological Objects 43
3.2 Symplectic Manifolds 49
3.3 Nonliner Control Systems and Optimal Control 55
3.4 Collocation 57
3.5 Shooting 59
3.6 Summary 60

4 ADJOINING OF OPTIMAL INFORMATION 61
4.1 Integrated Control Regularization Method 64

Page
4.1.1 Differential Optimal Control Law . . . 64
4.1.2 Symplectification of the Differential Optimal Control Law . . . 67
4.2 Results . . . 70
4.2.1 Goddard Rocket Problem . . . 70
4.3 Summary . . . 72

5 REDUCTION . . . 74
5.1 Topological Reduction . . . 75
5.1.1 Reduction . . . 75
5.1.2 Separation . . . 78
5.1.3 Reconstruction . . . 80
5.2 Algebraic Reduction . . . 81
5.3 Examples . . . 86
5.3.1 Currentless Zermelo's Problem . . . 87
5.3.2 Direct Brachistochrone . . . 90
5.3.3 Indirect Brachistochrone . . . 92
5.3.4 Free Parameters . . . 94
5.3.5 Clohessy-Wiltshire Equations . . . 96
5.3.6 Planar Atmospheric Flight . . . 98
5.4 Summary . . . 101

6 NUMERICAL SOLUTION TO REDUCED-DIMENSIONAL BOUNDARY-VALUE PROBLEMS . . . 104
6.1 Boundary-Value Problems on Reduced Manifolds . . . 105
6.1.1 Reduction of Constants-of-Motion . . . 105
6.1.2 Reduction of Symmetries . . . 106
6.1.3 Combined Reduction . . . 108
6.2 Numerical Algorithms . . . 109
6.2.1 Reduced Dimensional Collocation . . . 109
6.2.2 Reduced Dimensional Shooting . . . 110
6.3 Results . . . 113
6.3.1 Modified FORTRAN BVP Test Cases . . . 113
6.3.2 Brachistochrone Problem . . . 118
6.4 Summary . . . 122

7 COMPOSABLE CONSTRUCTION OF HAMILTONIAN BVPS . . . 124
7.1 Category Theory . . . 125
7.2 Optimal Control Functors . . . 130
7.2.1 Eps-Trig Regularization . . . 131
7.2.2 Universal Trigonomerization Method . . . 132
7.2.3 Relaxed Autonomously Switched Hybrid System . . . 134
7.2.4 Time-Shift . . . 136
7.3 Dualizing Methods . . . 137
7.3.1 Traditional Methods . . . 137

Page

7.3.2 Differential Geometric Methods . . . 138
7.4 Symplectomorphic Functors . . . 138
7.4.1 Symplectic ICRM . . . 140
7.5 Hybrid Methods . . . 141
7.5.1 Reduction . . . 141
7.6 Examples . . . 143
7.6.1 Brachistochrone . . . 143
7.6.2 Orbit Raising . . . 150
7.6.3 Moon Lander . . . 156
7.7 Extension . . . 161
7.8 Summary . . . 163

8 CONCLUSIONS . . . 164

9 FUTURE WORK . . . 167
9.1 Sequential Mishchenko-Fomenko Reduction . . . 167
9.2 Reduced Dimensional Co-vector Mapping Principle . . . 167
9.3 High Performance Numerical BVP Solvers . . . 170
9.4 0-D Boundary-Value Problems . . . 170
9.5 Lie-Poisson Reformulation . . . 171

REFERENCES . . . 172

PUBLICATIONS . . . 186

LIST OF TABLES

Table Page

1.1 Available Software for Direct Methods 6

1.2 Available Software for Indirect Methods 9

4.1 Parameters of the Goddard rocket. 72

6.1 Parameters for problems T2, T8, and T18 113

6.2 Results for the Shooting Method with $\eta = 1 \times 10^{-2}$ 117

6.3 Information on the Brachistochrone problem setup (30 collocation nodes) 120

6.4 Residual after each shooting update 122

7.1 Example composition table. 162

LIST OF FIGURES

Figure Page

1.1 State-of-the-art trajectory optimization categories. 4

1.2 Discretization of an OCP turns a mathematical problem into a computational one. The primary goal of computational Problem Σ^N is to seek confidence in the convergence of the approximation to Problem Σ. 4

1.3 Commutative diagram for the dualization and discretization of an OCP (reproduced from [11, 12, 88]). 10

1.4 A commutative diagram establishing $h = g \circ f$. 12

1.5 End-to-end mission design process. 27

1.6 End-to-end mission design process as constructed in this dissertation. . . . 32

2.1 Dual representation of the parallelogram defined by the wedge product. . . 37

2.2 Commutative diagram identifying $(W \otimes W) \otimes W \cong W \otimes (W \otimes W)$ and $K \otimes W \cong W \otimes K \cong W$. 39

2.3 The de Rham complex on $\mathfrak{g}^*$. 41

3.1 A 2-dimensional manifold. 43

3.2 The tangent bundle and some associated components. Note that since $\dim(M) = 1$, then $\dim(TM) = 2$ 46

3.3 Vector field on M 46

3.4 A vector field as the section of the tangent bundle 47

3.5 Co-vector field on M 48

3.6 A potential mapping between the tangent and co-tangent bundles of manifold M. 48

4.1 Results from the numerical solution of the BVP. 73

5.1 Marsden-Weinstein-Meyer Reduction. 77

5.2 Straightening out a circle 79

5.3 Cycloid curve as a solution to the Brachistochrone problem. 90

5.4 Brachistochrone state-space. 94

Figure Page

5.5 The Clohessy-Wiltshire equations govern the relative motion of a perturbed spacecraft in a rotating frame. 96

5.6 The planar configuration of a vehicle in atmospheric flight over a spherical Earth. 99

5.7 Trajectory of a maneuvering hypersonic vehicle. 101

5.8 Energy of a maneuvering hypersonic vehicle. 102

6.1 Solutions to R2, R8, and R18 with various difficulty parameters. 115

6.2 Times to solution of R2, R8, and R18 with $\eta = 1 \times 10^{-2}$. 116

6.3 Comparison of reduced and standard solutions to the Brachistochrone problem 120

6.4 Time to solution of the Brachistochrone problem for varying number of collocation nodes 121

7.1 A category with 4 objects and 3 mappings. 127

7.2 Category of OCPs. 127

7.3 Category of Hamiltonian BVPs. 128

7.4 A functor on $\mathscr{A}$. 129

7.5 OCFs preserve the CMP by modifying OCPs prior to dualization. 131

7.6 Symplectomorphic functors may preserve the CMP. 140

7.7 Commutative diagram for the dualization and reduction of an OCP possessing symmetries. 142

7.8 The configuration of the Brachistochrone problem is a principal G-bundle whose positions are fibered over velocity. 144

7.9 Solutions on the quotient manifold of the Brachistochrone problem. Every curve is a unique solution to the same OCP as the infinitesimal symmetry rotates through the configuration's fibers. 149

7.10 Quads of the Brachistochrone solution decomposed into ∂_x and ∂_y components. 150

7.11 Solutions in the low and high thrust orbit raising problems. 156

7.12 Symmetries in the low and high thrust orbit raising problems. 156

7.13 Symmetries in the low and high thrust orbit raising problems. 161

8.1 End-to-end mission design process as constructed in this dissertation. . . 164

Figure Page

9.1 Marsden-Weinstein-Meyer reduction and sequential Mishchenko-Fomenko reduction may result in diffeomorphic manifolds. 168

9.2 Commutative diagram for dualization and discretization of an OCP's dynamical system possessing constants-of-motion. 169

9.3 Commutative diagram for dualization and discretization of an OCP's dynamical system possessing symmetries. 169

SYMBOLS

$\cdot, (\cdot)$	Empty argument
$\bullet$	Empty grade
$\circ$	Composition
$\emptyset$	Null
$\oplus$	Direct sum
$\otimes$	Tensor product
$\wedge$	Exterior product
$\{\cdots\}$	Set
$\sharp$	Musical isomorphism "sharp"
$\flat$	Musical isomorphism "flat"
$\pounds$	Lie derivative
$\mathbb{d}$	Exterior derivative
$\varpi_\cdot$	Input bundle
∇	Gradient
$\rightarrow$	Morphism
$\mapsto$	Maps to
$\langle \cdot, \cdot \rangle$	Natural pairing
$[\cdot, \cdot]$	Concatenator
$[\cdot, \cdot]_{\mathrm{L}}$	Lie bracket
$[\cdot, \cdot]_{\mathrm{P}}$	Poisson bracket
$[\cdot, \cdot]_{\text{L-P}}$	Lie-Poisson bracket (future work only)
Δ	State-transition matrix
Λ, Problem Λ	Symplectic manifold, Category of Hamiltonian BVPs
$\boldsymbol{\Xi}$	Terminal constraints (Λ)
Σ, Problem Σ	Dynamics manifold, Category of optimal control problems

$\mathbf{\Phi}$	Initial constraints (Λ)
Υ	Trajectory (Λ)
β	Hessian matrix
γ	Trajectory (Σ)
δ	Variation
$\delta.$	Symmetry variation
ϵ	Tolerance
η	Terminal cost
θ	Liouville-Poincaré form
$\iota, \lrcorner$	Interior derivative
$\boldsymbol{\lambda}$	Lagrange or KKT multipliers
μ	Fixed point
$\boldsymbol{\nu}$	Non-dynamical Lagrange multipliers
$\boldsymbol{\xi}$	Terminal constraints (Σ)
$\pi.$	Co-tangent bundle
$\tau.$	Tangent bundle
$\boldsymbol{\phi}$	Initial constraints (Σ)
ω	Symplectic 2-form
$\mathscr{D}$	Dualization functor
$\mathscr{E}$	Euler-Lagrange operator
$\mathscr{F}$	Functor
$\mathscr{J}$	Jacobian matrix
$\mathfrak{g}$	Lie algebra
$\mathfrak{g}^*$	Lie co-algebra
$\mathfrak{h}$	Lie subalgebra
$\mathfrak{h}^*$	Lie co-subalgebra
A	Sensitivity matrix
B	Input manifold (Σ)
$C.$	Collocation coefficient

D	Derivation
E	Input manifold (Λ)
F	Flow
G	Lie group
H	Hamiltonian
J	Momentum map
K	Cost functional
L	Lagrangian
M	Manifold
N	Noether map
N^*	Noether co-map
O	Computational complexity
Q	State-space manifold
S	Symmetry
T	Tangent
T^*	Co-tangent
X	Hamiltonian vector field
$\boldsymbol{f}$	State equations-of-motion
$\boldsymbol{g}$	Constants-of-motion, constraints
$\boldsymbol{h}$	Quad equations-of-motion
i	Inclusion mapping
k	Number of terminal conditions
m	Number of initial conditions
n	Number of path-constraints
$\boldsymbol{p}$	Dynamical parameters
t	Time
$\boldsymbol{q}$	Quadratures “quads”
$\boldsymbol{u}$	Controls
$\boldsymbol{x}$	States

ABSTRACT

Sparapany, Michael J. PhD, Purdue University, May 2020. Aerospace Mission Design on Quotient Manifolds. Major Professor: Michael J. Grant.

Conceptual aerospace mission design has typically been performed in a computationally intensive and iterative manner. The introduction of modern computing has resulted in the widespread adoption of various numerical methods. As a result, useful information associated with the optimal solution is largely ignored. Optimization through indirect methods, while still computationally intense, leverages this information and also reveals a much deeper mathematical structure. This mathematical structure provides the gateway to reformulating the problem definition to one with certain desirable properties. In the presence of symmetries and constants-of-motion, the dynamical systems of indirect methods live in a reduced dimensional quotient manifold. Studies leveraging this reduced dimensional quotient manifold may benefit in performance by using fewer operations per iteration.

Many limitations prevent the use of these quotient manifolds in practical aerospace mission design. The five main issues include (1) rephrasing indirect methods in terms of differential geometry in an efficient manner, (2) Pontryagin's minimum principle generating a large number of valid dynamical systems, (3) implementing reduction in a global manner for highly non-linear systems, (4) numerical boundary-value problem solvers not supporting missions on quotient manifolds, and (5) scalability of the methods to real aerospace missions. This work addresses all five issues.

In previous studies, computer algebra systems have been proven to be an effective tool for automating complex indirect methods. However, when posed in the language

of differential geometry, the majority of support from digital software is lost. A version of indirect methods is recasted using differential geometry that effectively retains all information of so-called traditional methods. Exploitation of the anti-symmetric differential structure enables large-scale problems to be studied.

Root-solving the stationary Hamiltonian condition may generate several, potentially valid dynamical systems. Each system must be evaluated at every point along the trajectory using Pontryagin's minimum principle. This process prohibits later analytical derivations on the dynamical system. In the Integrated Control Regularization Method, the control law is posed as a state of the dynamical system with an equation-of-motion, thereby moving complicated root-solving to the boundary where it is solved once. Introduction of the control law as a new state is done using geometric adjoining methods where the original mathematical structure of the problem is preserved.

Reduction is traditionally studied in a topological context where there is a wealth of information. In terms of aerospace missions, there are very few applications in existence. These traditional studies rely on the quotient of entire global spaces. This is impossible to apply on non-integrable, non-linear dynamical systems. To get around this, a compact procedure is developed where the Lie algebra identifies dynamical sub-systems that may be effectively eliminated. This removes the reliance of integrability on the symmetry space.

In reduction, various dimensions desirable to a designer may be eliminated from the system defined on a reduced dimensional quotient manifold. Crucial to satisfying mission requirements, present day numerical solvers do not have the capability to perform the necessary reconstruction. A modified collocation and shooting algorithm with this functionality is given and a numerical example of a problem on a reduced

dimensional quotient manifold is explored.

Including reduction, nearly all advanced analytical techniques on dynamical systems introduce their own set of complexities. Typical aerospace designers do not want to deal with the many difficulties associated with each technique. By formalizing optimal control theory as a composable functorial process, these advanced strategies are compartmentalized with well-defined and predictable results. This enables the use and reuse of many different techniques in series, vastly improving on the automation of indirect methods.

1. INTRODUCTION

Aerospace missions are mathematical representions of real-world scenarios involving an aerospace vehicle. There is an enormous variety of vehicles including launch vehicles, manned near-Earth stations, interplanetary and deepspace craft, subsonic commercial airliners, Mars EDL, strategic bombers, supersonic air superiority fighters, and highly advanced hypersonic maneuvering vehicles. At the heart of every single one of these vehicles is the mission it was designed to fly. This mission includes some set of requirements guaranteeing success and a performance index that justifies its use over a competing vehicle.

The iterative design and development process of these vehicles is an expensive, man-hour intensive process. This process can range from a few hundred million dollars in venture capital funds for startups and small companies, to established programs costing a few billion US dollars [1,2]. These organizations strive to obtain high-quality representations of a vehicle's performance early on in the design process to prevent costly iteration and redesigns. To accomplish this, vehicle performance is estimated using a joint theoretical-computational field known as optimal control theory.

Optimal control theory is a broad mathematical field whose main concern is the minimization of functionals, a generalization of variational calculus. Historical application of optimal control dates back to Queen Elissar of Carthage who studied the isoperimetric problem [3]. In a more modern format, Newton, L'Hôpital, Leibniz, and the Bernoulli brothers studied the Brachistochrone problem [4]. The study of these problems resulted in the creation of groundbreaking analytic strategies. The introduction of digital computers and efficient nonlinear programming (NLP) strategies created a fork in optimal control theory.

***Definition* 1** (Optimal Control Problem)**.** An optimal control problem (OCP) is a functional minimization problem of the following structure

$$\text{Problem } \Sigma \begin{cases} \min\limits_{\boldsymbol{u}} K = \displaystyle\int_{t_0}^{t_f} L(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t + \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) \\ \text{Subject to: } \dfrac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{\phi}(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0)) = 0 \\ \boldsymbol{\xi}(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) = 0 \\ \boldsymbol{g}_{\text{lower}} \leq \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}) \leq \boldsymbol{g}_{\text{upper}} \end{cases} \tag{1.1}$$

End definition.

One category of numerical methods is the "direct" methods, which are widely known as the computational branch of optimal control theory. Direct methods seek to solve OCPs by discretizing the infinite-dimensional mathematical problem into an approximate finite-dimensional computational problem. These problems are relatively easy to set up on a digital computer and often times give very good results. The majority of algorithms in the direct methods branch do not leverage fundamental progress made in the theoretical branch of optimal control theory.

The other category of methods is the "indirect" methods, which is the theoretical branch of optimal control theory. Indirect methods seek to solve an OCP by solving a separate but equivalent boundary-value problem (BVP). The process of solving the BVP is a numerical root-solving problem as opposed to an optimization process. The majority of algorithms in the indirect methods branch ignore fundamental progress made in computational procedures as a result of the current digital age. This progress has largely not been numerically implemented due to the widespread use of direct methods for digital computer applications.

A third, but less related, category based on Hamilton-Jacobi-Bellman's (HJB) principle and dynamic programming exists and is mathematically distinct from both direct and indirect methods. HJB methods are not treated here because, although they have been used with some success [5], they are seldom used for general aerospace

missions due to their curse of dimensionality. It is worth mentioning that HJB's method offers the most mathematically powerful strategy proving global conditions for optimality [6–8], whereas both direct and indirect only provide local conditions [9,10].

Ultimately, direct and indirect methods are intertwined, and progress in either category is of benefit to both. There are several examples of this. The Co-vector Mapping Principles (CMPs) is one example [3,11,12]. The CMP allows the Karush-Kuhn-Tucker multipliers of the NLP problem in direct methods to be mapped to the co-states from indirect methods, and therefore provide a strategy for verifying optimality. General advancements in NLP, collocation, and pseudospectral routines driven by direct methods research are shared with indirect methods. One example of this is Kraft's `SLSQP` software. `SLSQP`'s original application was direct methods, although it is a common NLP solver of choice for indirect methods. The fact that the CMP exists teases the existence of so-called hybrid methods; methods that may leverage the benefits of both direct and indirect methods while minimizing their respective downsides. A figure showing each category and their driving numerical processes is in Fig. 1.1.

1.1 State-of-the-Art Methods

In this section, we will give a brief overview and history of the current state-of-the-art methods used in aerospace missions for solving optimal control problems.

1.1.1 Direct Methods

Direct methods operate by transcribing the OCP into an NLP. This effectively turns a mathematical problem into a computational problem. Save for direct shooting and other recursive integration methods, most techniques directly discretize the trajectory into a set of nodes. Every node represents a state approximation as a function of time. The NLP solver attempts to satisfy dynamics, i.e., Newton's laws, by interpolating between nodes and imposing an NLP constraint at the midpoint of

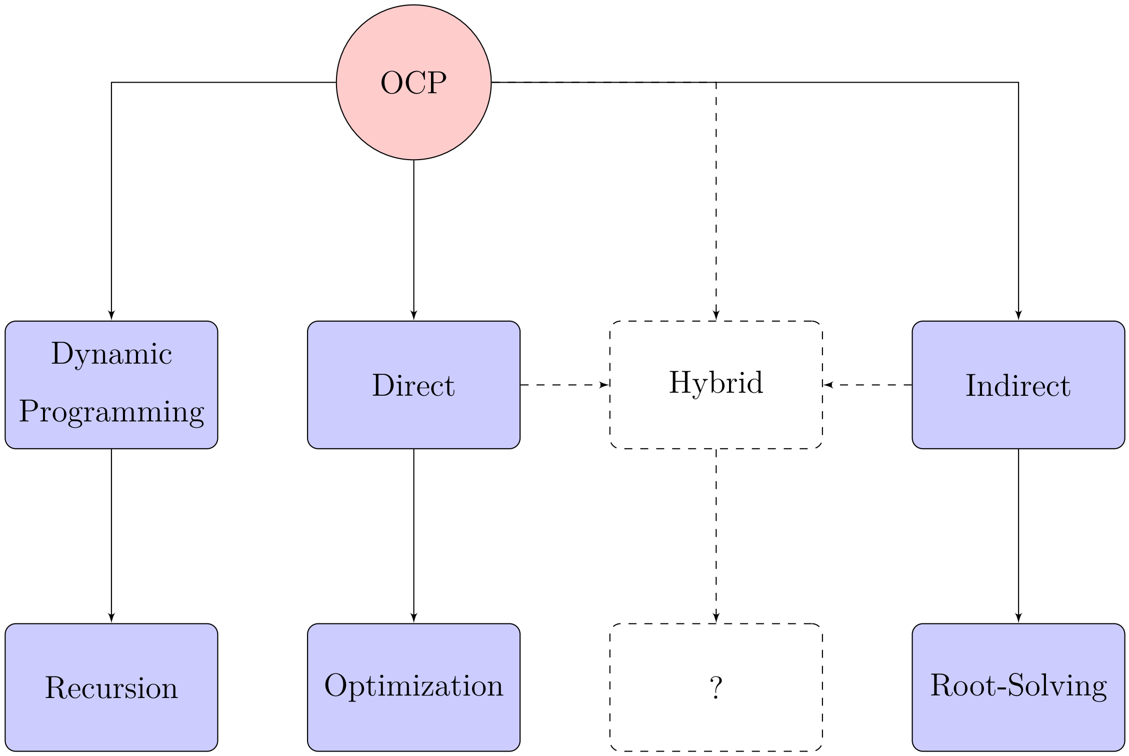


Figure 1.1. State-of-the-art trajectory optimization categories.

each pair of nodes. Mathematical OCPs are known as problems of type "Problem Σ" while their discretized brethren are known as problems of type "Problem Σ^N".

$$\text{Problem } \Sigma \underset{\text{discretization}}{\overset{\text{convergence}}{\leftrightarrows}} \text{Problem } \Sigma^N$$

Figure 1.2. Discretization of an OCP turns a mathematical problem into a computational one. The primary goal of computational Problem Σ^N is to seek confidence in the convergence of the approximation to Problem Σ.

Direct methods are widespread due to their generality and ease of use. Programs such as `POST` [13] are based on direct-shooting techniques, while `POST2`, `GESOP`, `DIRCOL`, `DYNOPT`, and `opty` [14] use various types of direct collocation techniques [15]. Other programs such as `GPOPS-II` [16, 17], `DIDO` [18], `PROPT`, `PSOPT`, and OpenGod-

dard use state-of-the-art pseudospectral methods [12]. The solution processes are typically driven by `IPOPT` [19], `NPSOL` [20], `SNOPT` [21], or another compiled nonlinear programming problem (NLP) solver. While there is an ongoing debate among users of direct methods on the benefits and disadvantages of each of these implementations [22], there is clearly no shortage of high-performance codes. In general, these programs are able to solve complex multiphase problems with multiple types of constraints, and in some specialized cases, contact forces [23].

Like all modern computational processes, there are drawbacks to using direct methods. Two of the most frequently cited downsides are listed below

1. Direct methods are less accurate than indirect methods.

2. Direct methods are difficult to parallelize.

While these drawbacks plague the majority of today's state-of-the-art solvers, these statements are broadly untrue. A brief word on these statements is listed below

1. While it is true that it is typically very easy to produce high quality trajectories with indirect methods, the broad statement that direct methods are less accurate is untrue. There exist highly sensitive problems where propagation of the indirect equations is inaccurate or prohibitively expensive. The inexactness of modern direct methods is precisely what allows them to have a high rate of convergence and often better accuracy over indirect methods. In fact, there exist problems with an infinite number of global optima where indirect methods do not yield a single solution. In these scenarios, direct methods can yield a solution [24]. In many other cases, indirect methods yield higher quality solutions.

2. Due to the optimization process at the core of direct methods, there are single-threaded operations that block computational resources, leaving other functions waiting for a response. While it is impossible for a general-purpose trajec-

tory optimization algorithm to reach full computational parallelization [1], there are certainly opportunities for "tricks". There is ongoing research in the parameter optimization field to create efficient parallel NLP solvers [25–28], research in computer science to create automated parallelization in digital compilers [29–33], and research in trajectory optimization itself through parallel direct shooting algorithms. Fundamentally, all trajectory optimization algorithms, regardless of category, must satisfy dynamics by solving sequential sub-problems. Both direct and indirect methods suffer from this drawback equivalently.

Overall, direct methods are used heavily with tremendous success. We list some of the successful solvers in Table 1.1.

Table 1.1. Available Software for Direct Methods

ACADO [34]	Astrogator (from STK)	BOCOP [35, 36]
CAMTOS [37]	Copernicus [38–42]	DIDO [18]
DIRCOL [43–45]	dynopt [46, 47]	FROST [48]
GESOP	GPOPS I and II [16, 17]	ICLOCS I and II [49]
MUSCOD I and II [50, 51]	NUDOCCCS [52, 53]	OpenGoddard [54]
OpenOCL [55]	OptimTraj [56, 57]	OPTRAGEN [58]
opty [14]	OTIS [59]	POST I and II [13]
PROPT [60]	PSOPT [61]	PyKEP [62]
Quickshot	RIOTS-95 [63]	SOCS [64]
SPARTAN [65]	TAOS [66]	

[1] An exception to this statement are general non-linear systems that satisfy the Liouville-Arnold theorem. However, one can argue that such a system is no longer "trajectory optimization" and require specialized routines for an efficient solution.

1.1.2 Indirect Methods

Prior to the rise of the modern digital computer, trajectory optimization was performed using so-called indirect methods. These methods are based on the calculus of variations [67] and Pontryagin's minimum principle (PMP) [68, 69]. They are advantageous by providing the designer additional information about the optimal solution. In fact relatively simple and low dimensional problems may be solved analytically "by hand", and larger problems are parallelizable, to an extent. Though heavily dependent on the problem, indirect methods typically scale better than their direct counterpart. This formulation rephrases the trajectory optimization problem as a BVP, which can be solved using BVP solvers such as `MATLAB`'s `BVP4C` function or `SciPy`'s `solve_bvp` method. Though the mathematics involved in setting up such a problem for such a solver is beyond that of a typical designer with no knowledge of optimal control theory, it has been previously shown that these complex mathematics can be automated through computer algebra systems (CAS) [70–74]. In fact, there appears to be an abundance of CASs with very rich histories [75–81]. These tools enable an indirect solver to utilize an interface similar to that of `GPOPS-II`, providing a familiar environment to an aerospace designer who may otherwise be unfamiliar with the mathematics involved.

Similar to direct methods, there are various downsides associated with indirect methods, some more valid than others, which are listed here

1. Specialized knowledge of calculus of variations as well as optimal control theory is required to pose a well defined problem.

2. Incorporating path constraints requires prior knowledge of the order of the constrained arc sequence.

3. The initial guess of a trajectory needs to be substantially close to the optimal solution. This issue is further obfuscated by the co-states whose values generally have no meaningful real-world interpretation.

4. Sensitivity associated with propagating the first-order necessary conditions [10].

Some comments on each of these frequently cited downsides are below

1. CASs make the first point easier by automating the construction of the necessary conditions for optimality. The earliest example of this is `OCCAL` [70]. We already mentioned some of the modern work using `MATLAB`'s symbolic engine and `SymPy`, however Enrico Bertolazzi, Francesco Biral, et. al. [71, 73] deserve special recognition for being pioneers using `Maple` as a CAS to automate indirect methods. This dissertation will further improve on this point in Chapters 2 and 7.

2. The phaseless construction of Enrico Bertolazzi et. al. [71,73] and Kshitij Mall et. al. [82–84] has completely eliminated the need for any a priori knowledge of path constraints.

3. A pseudo-hybrid method can overcome this point where initial guesses for costate variables are given by the Lagrange multipliers [85], or what is essentially the CMP, combined with a homotopy continuation procedure that tracks solutions to nearby dynamical systems. Since the point is partially overcome using a direct method, its (in)validity depends on the viewpoint of direct and indirect methods either being companion or competing strategies.

4. The Hamiltonian sensitivity issue exists in propagation, not batch integration. That is, this sensitivity issue does not exist to the same degree in Hermite-Simpson collocation. Often times Hermite-Simpson collocation is cited as not being a so-called "complete method". This only applies to direct methods. In indirect methods, Hermite-Simpson collocation is the current state-of-the-art algorithm.

Overall, indirect methods are significantly more complicated than direct methods. The list of available indirect solvers is in Table 1.2. This list is much shorter than

the list of direct methods. Additionally, `beluga` and `PINS` are the only two software packages based on indirect methods that can be considered general-purpose. `BNDSCO`, `CAMTOS`, and `PyKEP` use indirect methods, but not to the extent where they can be considered for general aerospace missions.

Table 1.2. Available Software for Indirect Methods

beluga	BNDSCO [86]	CAMTOS [37]
PINS [73]	PyKEP [62]	

1.1.3 Hybrid Methods

At the end of his 1998 publication, Betts closes out with the following remark [87]

> "...one may expect many of the best features of seemingly disparate techniques to merge, forming still more powerful methods."
>
> (J. T. Betts)

This comment is in reference to the then state-of-the-art direct and indirect methods. Less than 5 years after this publication, Ross and Fahroo would go on to publish one of the most significant advancements in trajectory optimization in recent history: the Co-vector Mapping Principle (CMP) [11, 12, 88]. The CMP is a specialization of the Riesz-Fréchet theorem [89–91], establishing a mathematical link between direct and indirect methods. From this link, so-called complete methods may be developed. Modern complete methods include Legendre pseudospectral methods and Hager's Runge-Kutta (RK) methods [92]. More important than the development of the present state-of-the-art computational tools, the CMP provides the mathematical bridge between theoretical analysis and computational tools. Ross and Fahroo's popular commutative diagram is reproduced here in Fig. 1.3.

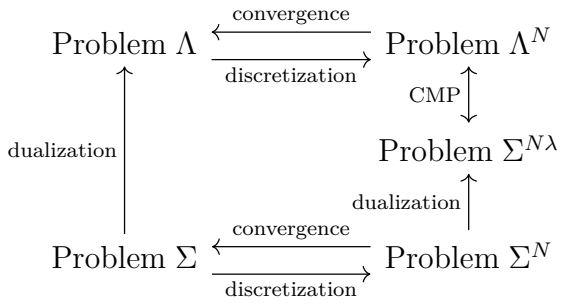


Figure 1.3. Commutative diagram for the dualization and discretization of an OCP (reproduced from [11, 12, 88]).

Throughout this dissertation, we will attempt to transform our problem definitions in a manner that preserves the CMP. Returning to Fig. 1.1, the existence of so-called hybrid methods is teased. This is made possible through the CMP. Although some specialized hybrid methods have existed for quite some time [37, 93], to my best knowledge, there is no hybrid direct and indirect method that can be considered general-purpose. However, herein lies one of the most significant contributions of the CMP: an advancement in either direct methods or indirect methods is of benefit to both categories of trajectory optimization. Instead of being viewed as competing strategies, direct and indirect methods should be viewed as companion strategies. This dissertation heavily focuses on indirect methods, which make a significant contribution to `beluga`, as an effort to bring these methods up to a level where general-purpose hybrid methods may be obtainable.

1.2 Foundations of Modern Indirect Trajectory Optimization

In this section, we will cover the majority of tools in modern trajectory optimization using indirect methods. We will broadly touch on topics in the calculus of variations and non-linear programming, as well as introduce concepts that will be formally defined in later chapters. One such concept already introduced in Fig. 1.3 is that of mappings in a commutative diagram.

***Definition* 2** (Mapping)**.** A mapping, which is denoted by an arrow $\rightarrow$, is a transformation between two objects referred to as the domain and co-domain. This mapping takes objects in the domain and transforms them in a well-known manner to the co-domain. *End definition.*

***Remark* 1.** The mapping, $\rightarrow$, is distinct from its implementation, $\mapsto$. For instance, consider a "squared" function mapping real numbers to their squared value. This complete mapping can be written as follows

$$\begin{aligned} \text{squared} : \mathbb{R} &\rightarrow \mathbb{R} \\ x &\mapsto x^2 \end{aligned} \tag{1.2}$$

End remark.

***Definition* 3** (After)**.** The composition operator, $\circ$, is the combination of two mappings to create a single chain mapping. Let, f and g be mappings where the co-domain of f is the domain of g

$$\begin{aligned} f &: A \rightarrow B \\ g &: B \rightarrow C \end{aligned} \tag{1.3}$$

The composition of mappings f and g is as follows

$$g(f) = g \circ f : A \rightarrow C \tag{1.4}$$

This has the pronunciation "g after f". *End definition.*

***Definition* 4** (Commutative Diagram)**.** A commutative diagram is a collection of objects and mappings where all paths taken by the same initial point that end in the same terminal point are equivalent. Given $f : A \rightarrow B$, $g : B \rightarrow C$, and $h : A \rightarrow C$, then $h = g \circ f$. This setup is shown in Fig. 1.4. *End definition.*

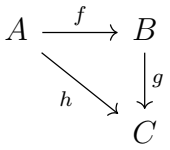


Figure 1.4. A commutative diagram establishing $h = g \circ f$.

Note that many definitions are introduced in the following section. These definitions are true to the modern usage of these tools in aerospace mission design, however, we will redefine many of these terms and quantities in later chapters.

1.2.1 Calculus of Variations

To understand indirect methods in trajectory optimization we must first introduce the calculus of variations, its role in physics, as well as some of its algebraic machinery. We will explore the calculus of variations in detail since it is the foundation of this dissertation. First introduced by Joseph-Louis Lagrange in 1788, the cornerstone of the calculus of variations is the variation, δ [94]. This operator represents an off-shell infinitesimal change in a system's state sometimes called a virtual displacement. An off-shell variation is a variation that may not satisfy the classical equations-of-motion. Rather, the displacement may perturb a point off of the energy shell thus violating conservation of energy [95]. The variation is distinct from the differential, d, in that a differential represents an on-shell infinitesimal change in a system's state. An on-shell variation is a variation that satisfies the classical equations-of-motion. To see how a variation differs from the differential, consider the following function

$$\begin{aligned} F : \mathbb{R} \times \mathbb{R}^n &\to \mathbb{R} \\ (t, \boldsymbol{q}) &\mapsto F(t, \boldsymbol{q}) \end{aligned} \tag{1.5}$$

The function, F is an $(n+1)$-dimensional slice of $(n+2)$-dimensional space. Assuming t is an independent variable, taking the differential of F gives us the following well-known result

$$\mathrm{d}F = \sum_{i=1}^{n} \frac{\partial F}{\partial q_i} \mathrm{d}F + \frac{\partial F}{\partial t} \mathrm{d}t \tag{1.6}$$

Alternatively, the first variation of F is

$$\delta F = \sum_{i=1}^{n} \frac{\partial F}{\partial q_i} \delta q_i \tag{1.7}$$

By definition, variations take place in spacial dimensions only, so $\delta t = 0$. In finite-dimensional parameter optimization, there is a significant focus on necessary conditions for optimality. The calculus of variations has a nearly identical condition that F is necessarily stationary about some fixed point $\boldsymbol{q}_0$. Mathematically, this becomes

$$\delta F \big|_{\boldsymbol{q}=\boldsymbol{q}_0} = 0, \ \forall \ \delta \boldsymbol{q} \tag{1.8}$$

Substituting this condition back into the variation of F, we find

$$0 = \sum_{i=1}^{n} \left.\frac{\partial F}{\partial q_i}\right|_{\boldsymbol{q}=\boldsymbol{q}_0} \delta q_i \tag{1.9}$$

Assuming all variations are independent and all constraints are non-holonomic, then we have

$$\left.\frac{\partial F}{\partial q_i}\right|_{\boldsymbol{q}=\boldsymbol{q}_0} = 0 \tag{1.10}$$

This is our primary result from the first variation of F. Often in optimization, a Hessian is constructed. The Hessian serves a multitude of purposes, the most common being as a classifier of critical points for a function. In the calculus of variations, a similar quantity exists as the second variation of F

$$\delta^2 F = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{\partial^2 F}{\partial q_i \partial q_j} \right) \delta q_i \delta q_j \tag{1.11}$$

Evaluating $\delta^2 F$ at the fixed point, $\boldsymbol{q}_0$, generates a symmetric matrix. From this matrix, can conclude that if

- $\delta^2 F|_{\boldsymbol{q}=\boldsymbol{q}_0}$ is positive definite, then the fixed point, $\boldsymbol{q}_0$, is a local minimum
- $\delta^2 F|_{\boldsymbol{q}=\boldsymbol{q}_0}$ is negative definite, then the reference point, $\boldsymbol{q}_0$ is a local maximum
- $\delta^2 F|_{\boldsymbol{q}=\boldsymbol{q}_0}$ is indefinite, then the reference point, $\boldsymbol{q}_0$, can be a saddle point, local minimum, or local maximum

In trajectory optimization using indirect methods, the first variation provides the first-order necessary conditions for optimality, whereas the second variation generates the generalized Legendre-Clebsch condition [96]. All the complexities of aerospace missions cannot be phrased within a single function F. Often there are several constraints that arise as smooth functions. Assume we have equations of constraint, $\boldsymbol{\phi}$, defined as follows

$$\begin{aligned} \boldsymbol{\phi} : \mathbb{R} \times \mathbb{R}^n &\to \mathbb{R}^m \\ (t, \boldsymbol{q}) &\mapsto (\phi_1(t, \boldsymbol{q}), \phi_2(t, \boldsymbol{q}), \cdots, \phi_m(t, \boldsymbol{q})) \end{aligned} \tag{1.12}$$

Taking the first variation of $\boldsymbol{\phi}$, we have

$$\delta\phi_j = \sum_{i=1}^{n} \frac{\partial \phi_j}{\partial q_i} \delta q_i \tag{1.13}$$

Since $\delta\boldsymbol{\phi}$ and δF exist as two independent quantities expressing behavior of a single system, there are obvious challenges in ensuring both quantities are simultaneously satisfied. To simultaneously satisfy these equations, we will use the Lagrange multiplier method to adjoin each constraint. This results in a single quantity expressing the necessary condition for a critical point

$$\sum_{i=1}^{n} \left[\frac{\partial F}{\partial q_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial \phi_j}{\partial q_i} \right|_{\boldsymbol{q}=\boldsymbol{q}_0} \delta q_i = 0 \tag{1.14}$$

To solve this condition for critical points, we must choose a value for $\boldsymbol{\lambda}$ such that

$$\left[\frac{\partial F}{\partial q_i} + \sum_{j}^{m} \lambda_j \frac{\partial \phi_j}{\partial q_i} \right|_{\boldsymbol{q}=\boldsymbol{q}_0} = 0, \quad \forall\, i = 1, \cdots, n \tag{1.15}$$

Eq. (1.13) and Eq. (1.15) provide us with $m + n$ equations to solve. We also have $m + n$ unknowns due to the introduction of m Lagrange multipliers along with n states. Viewing $\boldsymbol{\lambda}$ as variables at the same level as $\boldsymbol{q}$, F^* is defined as

$$\begin{aligned} F^* : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m &\to \mathbb{R} \\ (t, \boldsymbol{q}, \boldsymbol{\lambda}) &\mapsto \left(F(t, \boldsymbol{q}) + \sum_{j=1}^{m} \lambda_j \phi_j(t, \boldsymbol{q}) \right) \end{aligned} \tag{1.16}$$

Then, our first variation becomes

$$\begin{aligned} \delta F^* &= \delta(F + \boldsymbol{\lambda}\boldsymbol{\phi}) \\ &= \frac{\partial F}{\partial \boldsymbol{q}} \delta \boldsymbol{q} + \frac{\partial F}{\partial \boldsymbol{\lambda}} \delta \boldsymbol{\lambda} + \boldsymbol{\lambda} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}} \delta \boldsymbol{q} + \boldsymbol{\phi} \delta \boldsymbol{\lambda} \end{aligned} \tag{1.17}$$

Collecting terms with multipliers $\delta\boldsymbol{q}$ and $\delta\boldsymbol{\lambda}$ we find

$$\delta F^* = \left(\frac{\partial F}{\partial \boldsymbol{q}} + \boldsymbol{\lambda} \frac{\partial \phi}{\partial \boldsymbol{q}} \right) \delta \boldsymbol{q} + \left(\frac{\partial F}{\partial \boldsymbol{\lambda}} + \phi \right) \delta \boldsymbol{\lambda} \tag{1.18}$$

Then, evaluating at the fixed point $(\boldsymbol{q}, \boldsymbol{\lambda}) = (\boldsymbol{q}_0, \boldsymbol{\lambda}_0)$, we have n equations of the form

$$0 = \left[\frac{\partial F}{\partial \boldsymbol{q}} + \boldsymbol{\lambda} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}} \right|_{(\boldsymbol{q}, \boldsymbol{\lambda}) = (\boldsymbol{q}_0, \boldsymbol{\lambda}_0)} \tag{1.19}$$

Considering $\partial F/\partial \boldsymbol{\lambda} = 0$, there are also m equations of the form

$$0 = \boldsymbol{\phi}|_{(\boldsymbol{q},\boldsymbol{\lambda})=(\boldsymbol{q}_0,\boldsymbol{\lambda}_0)} \tag{1.20}$$

This method of augmenting functions using the Lagrange multiplier method gives us a compact procedure for mathematically linking functions in a manner that would otherwise have to be performed using a computationally intensive numerical procedure. In general, not all equations of interest are in algebraic form. Commonly, conditions exist as integrals. Recall our introduction of the variation, δ. A stationary integral is an integral whose value is unchanging under its first variation. We begin with the problem of finding a function, $y_0(x)$, such that the integral, K, has a stationary value

$$K = \int_{t_0}^{t_f} f(t, y(t), \dot{y}(t))\mathrm{d}t \tag{1.21}$$

where t is the independent variable, y is the dependent variable, and $\dot{y} = \mathrm{d}y/\mathrm{d}t$. We have fixed endpoints, t_0 and t_f, $f(y, \dot{y}, t)$ is a known function, and $y_0(t)$ is an unknown function. Using the concept of a variation

$$y(t) = y_0(t) + \delta y(t) \tag{1.22}$$

That is, y_0 represents the stationary solution and y the varied solution. We can rewrite this equation as

$$y(t, \alpha) = y_0(t) + \alpha\eta(t) \tag{1.23}$$

with α as a small parameter and η an arbitrary smooth function. If we assume $\delta y = 0$ at the endpoints, then we find

$$\eta(t_0) = \eta(t_f) = 0 \tag{1.24}$$

Then for any given $\eta(t)$ the integral K is a function of α only. Thus a necessary condition for K to be stationary is

$$\delta K = \alpha \frac{\mathrm{d}K}{\mathrm{d}\alpha}\bigg|_{\alpha=0} = 0 \tag{1.25}$$

where the derivative of the integral is

$$\frac{\mathrm{d}K}{\mathrm{d}\alpha} = \int_{t_0}^{t_f} \left(\frac{\partial f}{\partial y}\frac{\partial y}{\partial \alpha} + \frac{\partial f}{\partial \dot{y}}\frac{\partial \dot{y}}{\partial \alpha} \right) \mathrm{d}t \tag{1.26}$$

from the chain rule. Then, from Eq. 1.23, we have

$$\frac{\partial y}{\partial \alpha} = \eta(t) \tag{1.27}$$

Also from Eq. 1.23, we can find

$$\begin{aligned} \dot{y} &= \dot{y}_0(t) + \alpha\dot{\eta}(t) \\ \frac{\partial \dot{y}}{\partial \alpha} &= \dot{\eta}(t) \end{aligned} \tag{1.28}$$

Using these equations in Eq. 1.26 and restricting $\alpha = 0$, we have

$$\left.\frac{\mathrm{d}K}{\mathrm{d}\alpha}\right|_{\alpha=0} = \int_{t_0}^{t_f} \left[\frac{\partial f}{\partial y}\eta(t) + \left.\frac{\partial f}{\partial \dot{y}}\dot{\eta}(t)\right|_{\alpha=0}\right] \mathrm{d}t = 0 \tag{1.29}$$

To get this into a format that is more convenient, we integrate the second term by parts

$$\begin{aligned} u &= \frac{\partial f}{\partial \dot{y}}, \quad v = \eta(t) \\ \mathrm{d}u &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial f}{\partial \dot{y}}\right), \quad \mathrm{d}v = \dot{\eta}(t)\mathrm{d}t \end{aligned} \tag{1.30}$$

$$\begin{aligned} \int u\mathrm{d}v &= uv - \int v\mathrm{d}u \\ \int \frac{\partial f}{\partial \dot{y}}\dot{\eta}(t)\mathrm{d}t &= \left.\frac{\partial f}{\partial \dot{y}}\eta(t)\right|_{t_0}^{t_f} - \int \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial f}{\partial \dot{y}}\right)\eta(t)\mathrm{d}t \end{aligned} \tag{1.31}$$

Combining the result here with our previous integral terms, and also noting that the endpoints are fixed gives us

$$\left.\frac{\mathrm{d}K}{\mathrm{d}\alpha}\right|_{\alpha=0} = \int_{t_0}^{t_f} \left[\frac{\partial f}{\partial y}\eta(t) - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial f}{\partial \dot{y}}\right)\eta(t)\right] \mathrm{d}t = 0 \tag{1.32}$$

Collecting non-constant coefficients of $\eta(t)$, we find the Euler-Lagrange equation of f

$$\frac{\partial f}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial f}{\partial \dot{y}}\right) = 0 \tag{1.33}$$

As a more general case, consider the multidimensional curve

$$\begin{aligned} \boldsymbol{y} &: \mathbb{R} \to \mathbb{R}^n \\ t &\mapsto \boldsymbol{y}(t) \end{aligned} \tag{1.34}$$

The general set of Euler-Lagrange equations of f are then

$$\frac{\partial f}{\partial \boldsymbol{y}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial f}{\partial \dot{\boldsymbol{y}}} = 0 \tag{1.35}$$

The Euler-Lagrange equations of f provide the necessary and sufficient conditions for a stationary functional. In general, optimal trajectories $\boldsymbol{y}$ are minimizers of Eq. (1.21), and the Euler-Lagrange equations are a practical tool in finding such trajectories. In certain cases, there may exist trajectories that minimize the functional K, but these may occur in a different parameterization. As a concrete example, let K be the functional

$$K = \int_{t_0}^{t_f} f(t, \boldsymbol{y}(t), \dot{\boldsymbol{y}}(t))\mathrm{d}t \tag{1.36}$$

Then, define the following on-shell variation

$$\delta_\alpha \boldsymbol{y} = \alpha\eta(t, \boldsymbol{y}, \dot{\boldsymbol{y}}) = \boldsymbol{y}' - \boldsymbol{y} \tag{1.37}$$

where $\boldsymbol{y}'$ is a varied trajectory. Note that since this variation is on-shell, we are restricting our focus on the set of trajectories that are also minimizers of this functional. In general, we call on-shell variations the symmetry of a system.

***Definition* 5** (Symmetry (1))**.** Let δ_α be a variation acting on functional K. Then δ_α is a symmetry if the variation is on-shell such that the varied solutions $\boldsymbol{y}'$ are also solutions to the Euler-Lagrange equations generated by δK. *End definition.*

Evaluating this variation of K, we find:

$$\delta_\alpha K = \delta_\alpha \int_{t_0}^{t_f} f(t, \boldsymbol{y}, \dot{\boldsymbol{y}})\mathrm{d}t \tag{1.38}$$

Recalling our previous derivation of the Euler-Lagrange equations using integration by parts, we have

$$\delta_\alpha K = \int_{t_0}^{t_f} \left[\frac{\partial f}{\partial \boldsymbol{y}}\delta_\alpha \boldsymbol{y} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial f}{\partial \dot{\boldsymbol{y}}}\delta_\alpha \boldsymbol{y}\right]\mathrm{d}t + \left.\frac{\partial f}{\partial \dot{\boldsymbol{y}}}\delta_\alpha \boldsymbol{y}\right|_{t_0}^{t_f} \tag{1.39}$$

Note that, even though $\delta\boldsymbol{y}(t_0) = \delta\boldsymbol{y}(t_f) = 0$, since $\eta = \eta(\boldsymbol{y})$, then $\delta_\alpha \boldsymbol{y}(t_0) \neq 0$ and $\delta_\alpha \boldsymbol{y}(t_f) \neq 0$. For the terms under the integral

$$\frac{\partial f}{\partial \boldsymbol{y}}\delta_\alpha \boldsymbol{y} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial f}{\partial \dot{\boldsymbol{y}}}\delta_\alpha \boldsymbol{y} = 0 \tag{1.40}$$

Rather, the varied path must still satisfy the Euler-Lagrange equations. Substituting this expression back into the $\delta_\alpha K$, we find

$$\delta_\alpha K = \frac{\partial f}{\partial \dot{\boldsymbol{y}}} \delta_\alpha \boldsymbol{y} \Big|_{t_0}^{t_f} \tag{1.41}$$

Since $\delta_\alpha \boldsymbol{y}$ is on-shell, $\delta_\alpha K = 0$ for all solutions. Integrating this quantity, we have

$$J \delta_\alpha \boldsymbol{y} = \frac{\partial f}{\partial \dot{\boldsymbol{y}}} \eta \delta_\alpha \boldsymbol{y} \tag{1.42}$$

Therefore J is a constant-of-motion of the system. This result is also referred to as a Noether current. In fact, for every on-shell symmetry of the system, Noether's Theorem guarantees the existence of a constant-of-motion [97].

***Definition* 6** (Noether's Theorem)**.** Let K be the functional

$$K(\boldsymbol{y}(t)) = \int_{t_0}^{t_f} f(t, \boldsymbol{y}(t), \dot{\boldsymbol{y}}(t)) \mathrm{d}t \tag{1.43}$$

If K is invariant under the on-shell variation $\delta_\alpha \boldsymbol{y} = \alpha \eta(\boldsymbol{y}, \dot{\boldsymbol{y}}, t)$, then the following quantity is conserved

$$J = \frac{\partial f}{\partial \dot{\boldsymbol{y}}} \eta \tag{1.44}$$

End definition.

While this version of Noether's Theorem is relatively powerful, it isn't very useful for general-purpose aerospace missions. This is primarily due to the fact that it is challenging to define an on-shell variation that leaves the functional invariant. Additionally, given a constant-of-motion, J, in the variational context there is no convenient manner of determining its associated on-shell variation. Later in Chapter 3, we will give a reversible version of Noether's Theorem that eliminates difficulties associated with defining the on-shell variations.

1.2.2 Dualization

A key operation in trajectory optimization by indirect methods is the construction of a dual problem to the OCP. The process of constructing such a problem is referred to as dualization.

***Definition* 7** (Dualization)**.** In trajectory optimization by indirect methods, dualization is the application of the Euler-Lagrange equations after adjoining constraints

$$\mathscr{D} \equiv \mathscr{E} \circ \mathscr{F}_{\lambda} \tag{1.45}$$

This converts an OCP into a dual problem, which is a Hamiltonian BVP

$$\text{Problem } \Lambda = \mathscr{D}(\text{Problem } \Sigma) \tag{1.46}$$

End definition.

***Remark* 2.** Let $\mathscr{D}(\gamma)$ be a trajectory in Problem $\Lambda = \mathscr{D}$(Problem Σ). If $\mathscr{D}(\gamma)$ is a solution to Problem Λ, then γ is a minimizer of the functional in Problem Σ. Ergo solutions to Problem Λ *indirectly* solve Problem Σ. *End remark.*

To give a concrete example on how dualization is traditionally performed, let Problem Σ be a non path-constrained OCP of the following form

$$\begin{aligned} \min_{\boldsymbol{u}} K &= \int_{t_0}^{t_f} L(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t + \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) \\ \text{Subject to: } \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} &= \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{\phi}(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0)) &= 0 \\ \boldsymbol{\xi}(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) &= 0 \end{aligned} \tag{1.47}$$

In the first step of dualization, we adjoin the constraints with the augmented Lagrange multiplier method introduced in Section 1.2.1. The boundary conditions are adjoined with non-dynamical Lagrange multipliers $(\boldsymbol{\nu}_0, \boldsymbol{\nu}_f)$, and the dynamics are adjoined with dynamical Lagrange multipliers $\boldsymbol{\lambda}$.

***Definition* 8** (Dynamical Parameter)**.** Let $\dot{\boldsymbol{x}} = \boldsymbol{f}$ be a dynamical system and let $\boldsymbol{\phi}$ and $\boldsymbol{\xi}$ be equations on the boundaries of solutions to the dynamical system. A dynamical parameter, $\boldsymbol{p}$, is a constant that affects the time-evolution of the dynamical system such that

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}), \quad \boldsymbol{p} \notin \boldsymbol{x} \\ \boldsymbol{\phi}(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}) &= 0 \\ \boldsymbol{\xi}(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), \boldsymbol{p}) &= 0 \end{aligned} \tag{1.48}$$

End definition.

***Definition* 9** (Non-Dynamical Parameter)**.** Let $\boldsymbol{p}$ be a parameter of a dynamical system. This parameter is non-dynamical if it does not affect the time-evolution of states

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}), \quad \boldsymbol{p} \notin \boldsymbol{f} \text{ and } \boldsymbol{p} \notin \boldsymbol{x} \tag{1.49}$$

End definition.

***Remark* 3.** Though non-dynamical parameters do not impact the time-evolution of a dynamical system, they may appear in functions at the boundaries

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}) &= 0 \\ \boldsymbol{\xi}(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), \boldsymbol{p}) &= 0 \end{aligned} \tag{1.50}$$

End remark.

Lagrange multipliers $(\boldsymbol{\nu}_0, \boldsymbol{\nu}_f)$ are non-dynamical parameters while Lagrange multipliers $\boldsymbol{\lambda}$ are dynamical states. The $\boldsymbol{\lambda}$'s are often called co-states. First writing the dynamical equations as $\boldsymbol{f} - \dot{\boldsymbol{x}} = 0$, we then augment the functional K as previously shown in Eq. (1.16)

$$K^* \equiv \int_{t_0}^{t_f} \left[L + \boldsymbol{\lambda} \left(\boldsymbol{f} - \dot{\boldsymbol{x}} \right) \right] \mathrm{d}t + \boldsymbol{\nu}_0 \boldsymbol{\phi} + (\eta + \boldsymbol{\nu}_f \boldsymbol{\xi}) \tag{1.51}$$

Next, we introduce the control Hamiltonian, $H \equiv L + \boldsymbol{\lambda} \boldsymbol{f}$. Then

$$K^* = \int_{t_0}^{t_f} \left(H - \boldsymbol{\lambda} \dot{\boldsymbol{x}} \right) \mathrm{d}t + \boldsymbol{\nu}_0 \boldsymbol{\phi} + (\eta + \boldsymbol{\nu}_f \boldsymbol{\xi}) \tag{1.52}$$

Eq. (1.52) is an unconstrained version of the original Problem Σ, called Problem Σ^*. We use the symbol "Σ" to signify that it is still an OCP, but place the star on it to emphasize its augmented nature. The second step in the dualization procedure is an application of the Euler-Lagrange equations

$$\mathscr{E} \equiv \frac{\partial}{\partial \boldsymbol{y}} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial}{\partial \dot{\boldsymbol{y}}} \tag{1.53}$$

Applying the Euler-Lagrange equations to Eq. (1.52) gives

$$\mathscr{E}(K^*) = 0 \tag{1.54}$$

For the dynamic portion of the functional, we have

$$\begin{aligned} \mathscr{E}\left(H - \boldsymbol{\lambda}\dot{\boldsymbol{x}}\right) &= \left(\frac{\partial}{\partial \boldsymbol{y}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial \dot{\boldsymbol{y}}}\right)\left(H - \boldsymbol{\lambda}\dot{\boldsymbol{x}}\right) \\ &= \frac{\partial\left(H - \boldsymbol{\lambda}\dot{\boldsymbol{x}}\right)}{\partial \boldsymbol{y}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\left(H - \boldsymbol{\lambda}\dot{\boldsymbol{x}}\right)}{\partial \dot{\boldsymbol{y}}} \end{aligned} \tag{1.55}$$

In the case of our OCP, $\boldsymbol{y} = \{\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}\}$. Acknowledging $\partial\boldsymbol{\lambda}/\partial\boldsymbol{z} = 0$ for $\boldsymbol{z} = \{\boldsymbol{y}\}/\{\boldsymbol{\lambda}\}$, we expand

$$\mathscr{E}\left(H - \boldsymbol{\lambda}\dot{\boldsymbol{x}}\right) = \frac{\partial H}{\partial \boldsymbol{x}} + \frac{\partial H}{\partial \boldsymbol{\lambda}} + \frac{\partial H}{\partial \boldsymbol{u}} + \frac{\partial\left(-\boldsymbol{\lambda}\dot{\boldsymbol{x}}\right)}{\partial \boldsymbol{\lambda}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial(-\boldsymbol{\lambda}\dot{\boldsymbol{x}})}{\partial \dot{\boldsymbol{x}}} \tag{1.56}$$

Simplifying this expression down gives

$$\mathscr{E}\left(H - \boldsymbol{\lambda}\dot{\boldsymbol{x}}\right) = \frac{\partial H}{\partial \boldsymbol{x}} + \frac{\partial H}{\partial \boldsymbol{\lambda}} + \frac{\partial H}{\partial \boldsymbol{u}} - \dot{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}} \tag{1.57}$$

From this, we choose $\boldsymbol{\lambda}$ such that the following holds

$$\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}},\ \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}},\ 0 = \frac{\partial H}{\partial \boldsymbol{u}} \tag{1.58}$$

Next, we look at the terms at the boundaries. In optimal control theory, emphasis is placed on the structure of boundary conditions. Specifically, state terms are separated into either being "fixed" or "free". Although this strategy works well for certain problems, the required structure of the boundary conditions is greatly restricted and doesn't lend itself well for general-purpose processes. Instead, the current state-of-the-art strategy is to adjoin each of boundary conditions to the initial and terminal cost functions with non-dynamical Lagrange multipliers. These terms then appear in the BVP's boundary conditions as follows

$$\begin{aligned} \boldsymbol{\Phi} &= \boldsymbol{\nu}_0\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} + \boldsymbol{\lambda} \\ \boldsymbol{\Xi} &= \frac{\partial \eta}{\partial \boldsymbol{x}} + \boldsymbol{\nu}_f\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}} - \boldsymbol{\lambda} \end{aligned} \tag{1.59}$$

where $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_f$ are newly introduced Lagrange multipliers. There is no special treatment for whether a state is "free" or "fixed". Finally, we arrive at the dynamic first-order necessary conditions for optimality

$$\text{Problem } \Lambda \begin{cases} \dot{\boldsymbol{x}} = \dfrac{\partial H}{\partial \boldsymbol{\lambda}},\ \dot{\boldsymbol{\lambda}} = -\dfrac{\partial H}{\partial \boldsymbol{x}},\ \mathbf{0} = \dfrac{\partial H}{\partial \boldsymbol{u}} \\ \boldsymbol{\Phi}(\boldsymbol{x}(t_0), \boldsymbol{\lambda}(t_0), \boldsymbol{u}(t_0)) = 0 \\ \boldsymbol{\Xi}(\boldsymbol{x}(t_f), \boldsymbol{\lambda}(t_f), \boldsymbol{u}(t_f)) = 0 \end{cases} \tag{1.60}$$

Eq. (1.60) is said to be of type Problem Λ. Note that the dynamical dimension of Problem Λ is twice that of Problem Σ. Also, Problem Λ is in differential-algebraic (DAE) form since $\partial H/\partial \boldsymbol{u} = 0$ must be solved simultaneously with $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{\lambda}}$. Recall that Noether's theorem from the calculus of variations sections in Section 1.2.1 enabled us to generate a constant-of-motion for every symmetry of a functional. This is due to the symplectic structure of the Hamiltonian system tying together constants-of-motion with their associated symmetries. In Chapter 5, we will show how this symplectic structure may be exploited to eliminate both the constant-of-motion and its associated symmetry. Before that, we must cover one more important result specifically from Hamiltonian mechanics.

Consider a non-control Hamiltonian system with phase-space, $(\boldsymbol{x}, \boldsymbol{\lambda}) \in \Lambda$. Let ρ be a density function and let $\boldsymbol{X}$ be the concatenation of phase-space coordinates, $[\boldsymbol{x}, \boldsymbol{\lambda}]$. Consider the continuity equation [98]

$$0 = \frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \dot{\boldsymbol{X}} \right) \tag{1.61}$$

Expanding the gradient term yields

$$\nabla \cdot \left(\rho \dot{\boldsymbol{X}} \right) = \frac{\partial \left(\rho \dot{\boldsymbol{X}} \right)}{\partial \boldsymbol{X}} \quad = \dot{\boldsymbol{x}} \frac{\partial \rho}{\partial \boldsymbol{x}} + \dot{\boldsymbol{\lambda}} \frac{\partial \rho}{\partial \boldsymbol{\lambda}} + \rho \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}} + \rho \frac{\partial \dot{\boldsymbol{\lambda}}}{\partial \boldsymbol{\lambda}} \tag{1.62}$$

Using the following relationships

$$\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}}, \quad \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} \tag{1.63}$$

We can expand the derivative terms, $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{\lambda}}$, on the right hand side of Eq. (1.62) to find

$$\rho \frac{\partial^2 H}{\partial \boldsymbol{x} \partial \boldsymbol{\lambda}} - \rho \frac{\partial^2 H}{\partial \boldsymbol{\lambda} \partial \boldsymbol{x}} = 0 \tag{1.64}$$

Substituting this back into Eq. (1.61), we find

$$0 = \frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial \boldsymbol{X}} \frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}t} \tag{1.65}$$

This expression is equivalent to the time-rate of change of ρ, $\mathrm{d}\rho/\mathrm{d}t$.

***Definition* 10** (Liouville's Theorem)**.** Let $\dot{\boldsymbol{X}} = \boldsymbol{f}$ be a Hamiltonian system generated by a Hamiltonian function, H. Then, the density of states nearby a given state is constant along every trajectory. *End definition.*

While Liouville's theorem is a fundamental result that applies to optimal control theory and has seen extensive theoretical usage [99, 100], there does not appear to be many applications of Liouville's theorem on the computational front of optimal control theory. Recall Noether's theorem as introduced in Section 1.2.1 was in a format that was difficult to use, especially in aerospace design. In a similar manner, Liouville's theorem as presented in this section is very difficult to use in general-purpose applications. Like Noether's theorem, in Chapter 3 we will provide a different version of Liouville's theorem that is easier to use.

1.2.3 Nonlinear Programming

Nonlinear programming is the general class of methods of solving parameter optimization problems subject to a set of nonlinear constraints. These methods are the language that ultimately enables a mathematical problem to be translated into a computational problem that is solved by a computer. This computational problem is referred to as a Non-Linear Programming (NLP) problem. NLP strategies are a large focus within direct methods. However, they are often neglected in the literature when indirect methods are involved. After all the analytical manipulations are performed within an indirect method, it is highly unlikely that the resulting problem will be analytically solvable. For general-purpose applications, we must resort to NLP strategies to generate approximate solutions to this problem.

***Definition* 11** (Non-Linear Programming Problem)**.** An NLP problem is a computational problem of the following form:

$$\begin{aligned} &\min K(\boldsymbol{x}) \\ \text{Subject to: } &g_i(\boldsymbol{x}) \leq 0 \; \forall \; i \in \{0, \cdots, m\} \\ &h_j(\boldsymbol{x}) = 0 \; \forall \; j \in \{0, \cdots, n\} \end{aligned} \tag{1.66}$$

End definition.

Aside from analytically solvable problems, all trajectory design problems, regardless if direct or indirect, are posed as an NLP and solved numerically. This is typically not a challenge due to the present-day widespread use of NLP solvers. Though somewhat anecdotal, NLP solvers that aim to directly satisfy the Karush-Kuhn-Tucker (KKT) conditions are the most well-behaved in trajectory design problems. The KKT conditions are a set of first-order necessary conditions for optimality of the NLP [101, 102]. To define the KKT conditions, we must first augment the cost function with KKT multipliers to create the augmented Lagrangian function

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = K(\boldsymbol{x}) + \sum_{i=0}^{m} \lambda_i g_i(\boldsymbol{x}) + \sum_{i=0}^{n} \lambda_{i+m} h_i(\boldsymbol{x}) \tag{1.67}$$

If $\boldsymbol{x}^*$ is an optimal solution to the NLP, the KKT conditions are then

1. Stationary: $0 = \nabla K(\boldsymbol{x}^*) + \sum_{i=0}^{m} \lambda_i \nabla g_i(\boldsymbol{x}^*) + \sum_{i=0}^{n} \lambda_{i+m} \nabla h_i(\boldsymbol{x}^*)$
2. Inequality Primal Feasibility: $g_i(\boldsymbol{x}^*) \leq 0, \; \forall \; i \in \{0, \cdots, m\}$
3. Equality Primal Feasibility: $h_i(\boldsymbol{x}^*) = 0, \; \forall \; i \in \{0, \cdots, n\}$
4. Dual Feasibility: $\lambda_i \geq 0, \; \forall \; i \in \{0, \cdots, m\}$
5. Complementary Slackness: $\lambda_i g_i(\boldsymbol{x}^*) = 0, \; \forall \; i \in \{0, \cdots, m\}$

Conditions 1-3 are directly obtained from the NLP problem definition in Eq. 1.66 and the augmented Lagrangian in Eq. 1.67. Conditions 4 and 5 come from the following dual problem

$$\begin{gathered}\max_{\boldsymbol{\lambda}} \left(\boldsymbol{\lambda} \mapsto \min_{\boldsymbol{x}} \; L(\boldsymbol{x}, \boldsymbol{\lambda})\right) \\ \text{Subject to: } \lambda_i \geq 0 \; \forall \; i \in \{0, \cdots, m\}\end{gathered} \tag{1.68}$$

Sequential Quadratic Programming (SQP) is a direct constrained method that utilizes a tiered iterative procedure to solve NLP problems. The upper tier uses KKT multipliers for incorporating the constraints. From this, a Quadratic Programming (QP) sub-problem with a quadratic objective function that approximates the Lagrangian function and linear constraints is then created. The lower tier is responsible for solving this QP sub-problem using convex optimization. Solutions to the QP sub-problem directly satisfy KKT conditions and provide the upper tier search direction information. The upper tier then uses this information to move the parameters to a new point and ultimately creates another approximation for the augmented Lagrangian and QP sub-problem. The upper tier continues this process until KKT conditions are satisfied.

To start the SQP process, an initial guess $\boldsymbol{x}_0$ is required. This guess is not required to be feasible. Then the QP sub-problem is defined as follows

$$\begin{gathered}\min Q(s) = K(\boldsymbol{x}_0) + \nabla K^T(\boldsymbol{x}_0)s + \frac{1}{2}s^T\beta s \\ \text{Subject to: } \nabla g_i^T(\boldsymbol{x}_0)s + \delta_i g_i(\boldsymbol{x}_0) \leq 0 \; \forall \; i \in \{0, \cdots, m\} \\ \nabla h_i^T(\boldsymbol{x}_0)s + \bar{\delta} h_i(\boldsymbol{x}_0) = 0 \; \forall \; i \in \{0, \cdots, n\}\end{gathered} \tag{1.69}$$

In the QP sub-problem, we introduced β as the estimate of the Hessian of the augmented Lagrangian function. The parameters δ_i are constant multipliers that prevent cutting off feasible space due to the linearization of the constraints. For the first upper tier iteration, β is the identity matrix. Choosing $\bar{\delta}$ is subjective and must be between 0 and 1, but is usually close to 0.92. The δ_i are then determined by

$$\begin{cases}\delta_i = 1 & \text{if} \quad g_i(\boldsymbol{x}_0) < 0 \\ \delta_i = \bar{\delta} & \text{if} \quad g_i(\boldsymbol{x}_0) \geq 0\end{cases} \tag{1.70}$$

In the special case where g_i or h_i are linear, the QP sub-problem can directly handle these constraints, and choosing $\bar{\delta} = 1$ and $\delta_i = 1$ yields better performance, but is not necessary. Solving this QP sub-problem yields values for s, as well as estimates for the KKT multipliers, $\boldsymbol{\lambda}$.

1.3 End-to-End Mission Design

An aerospace system designer is likely not concerned with the intricacies of optimal control theory. They simply want to pose a problem and get an answer. Beyond the modeling of their specific system, a designer does not want to be bothered with discretization schemes or Jacobian matrices, yet it is unfortunate that such an individual will most likely need to learn some amount of optimal control theory in order to be successful. When such a designer poses a hypothetical mission, this mission undergoes various transformations before it reaches a computational architecture and is solved numerically. An example process is shown in Fig. 1.5.

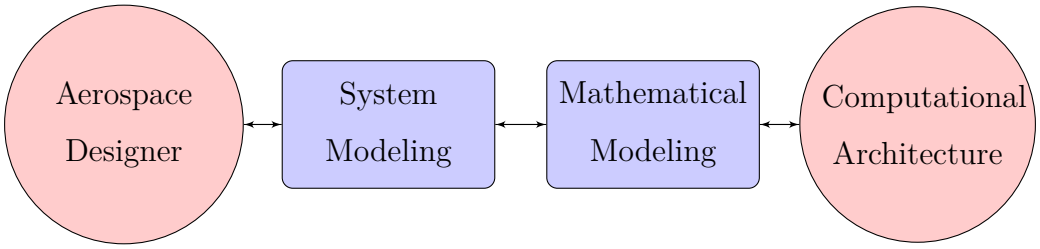


Figure 1.5. End-to-end mission design process.

Regardless of what approach for computing a trajectory a designer chooses, both direct and indirect methods are computationally intensive and iterative. In direct collocation and pseudospectral methods, increasing a mesh's accuracy, or number of nodes, is directly proportional to the size of the resulting nonlinear programming problem. In addition to mesh size, increasing the number of constraints also further increases the size of the NLP problem. On the other hand, shooting solvers common to indirect methods do not suffer the same performance impact with an increase in

mesh accuracy. Instead, a state-transition matrix captures sensitivity information thereby eliminating the need for a mesh. This state-transition matrix is proportional in size to the square of the number of constraints, therefore, it does have a performance impact similar to that of collocation and pseudospectral methods due to an increase in the number of constraints. Herein lies one major challenge of all mission design and trajectory optimization algorithms. As computational complexity increases, there is a disproportionate increase in time to solution. One common technique to mitigate this performance impact involves deploying algorithms on highly parallel architectures.

There is much present day research in mission design and trajectory optimization in both indirect [103] and direct methods [28] that aims to make efficient use of high-performance supercomputers [104], graphics processing units [105], field-programmable gate arrays, and other highly parallel systems. These algorithms have varying degrees of success; however, they experience a point of diminishing returns. For a low number of computational workers, there is a reasonable return on performance. As the number of computational workers increases, one cannot expect a continued proportional return on performance. This is due to the underlying nonlinear structure that exists in both direct and indirect methods. This is not an issue for some scenarios where the dynamics may be simplified to linear systems [106], or can be convexified [107, 108]. In general most nonlinear systems do not have these luxuries. The ideal number of parallel workers is not well defined and changes based on problem definition. This computational limitation is not unique to trajectory design and is well-known to computer scientists as Amdahl's law [109].

Since computational efficiency is ultimately limited by the nonlinear structure of the problem, the primary goal of this dissertation is to change the nonlinear structure to one with certain desirable properties. Specifically, complete parallelization is impossible since nonlinear systems are subject to causality. The entire dynamical space cannot be reasonably sampled and predicted for every possible perturbation, so computational workers are forced to consolidate their results and create an update in a single-threaded, single-operation manner. By eliminating causality, perturba-

tions that have no impact on the dynamical evolution of the system may be solved independently and efficiently in a parallel manner. Segregating a system's dynamics in this manner introduces so-called quotient or reduced manifolds where a system's true causal dynamics live. This effectively offloads complex calculations to sophisticated mathematics while saving computational resources for parts of the problem where it is actually required. In this process, preserving the original design problem and its mathematical structure through careful algebraic manipulations is incredibly important. To do this, we use results from differential geometry and theoretical physics.

Much of the modern research in theoretical physics aims to include gravity into quantum theory. In the majority of proposed solutions, the quantization of gravity introduces several issues from gauge theory. To overcome some of these issues, redundant quantities are introduced into the system which must be removed in the end. Handling these quotient constructions is typically done using the Becchi-Rouet-Stora-Tyutin formalism [110–113], which is a semi-rigorous approach, while the Batalin-Vilkovisky formalism [114–116] further generalizes this process. This work has been extended to Hamiltonian gauge theories through the Batalin-Fradkin-Vilkovisky formalism [117] in the form of homotopical Poisson reduction. In mission design and trajectory optimization, similar redundant quantities exist and, in fact, indirect methods with redundant quantities are structurally isomorphic to various systems that appear in theoretical physics. Though there isn't a single generally agreed upon theory of incorporating gravity into quantum theory, these tools have tremendous success in theoretical physics and have natural extensions to mission design and trajectory optimization through the field of geometric control theory.

Geometric control theory focuses on the structure and topological characteristics of dynamical systems using known results from differential geometry and theoretical physics. This field models dynamical systems and their control inputs as flows along manifolds as opposed to functions on a fixed chart, or coordinate system, as is done in both direct and indirect methods. The manifolds that arise from indi-

rect methods have a natural symplectic structure that can be exploited with the Marsden-Weinstein-Meyer reduction theorem [118, 119] and Poisson reduction theorems [120–122]. Since fixed charts are no longer used, this new setting may present itself as unfamiliar to a designer. Similar to how indirect methods can be automated, we aim to automate all differential geometric operations abstracting away the mathematical complexities from the designer. Currently, to the author's knowledge, there are no software tools available that fully automate the construction of quotient manifolds required for design. The contribution of this dissertation is end-to-end in the following sense: we will bring geometric control theory and Marsden-Weinstein-Reduction to a level where it can be applied to general-purpose aerospace mission design. As such, there is no single aerospace mission that has been selected for study, but rather we will create a system that has potential application to every aerospace mission.

1.4 Overview

The overall contribution of this dissertation is best described with an example. Consider a mathematical system of equations. The type of equations are not important. They may be algebraic linear, algebraic non-linear, or even ordinary differential equations. In this system, there are 8 unknown variables and 2 known quantities. If we manipulate the system by using its known quantities to solve for the unknowns, what is the fewest number of unknowns we can have in the resulting new system? Common sense tells us 6, yet this dissertation says 4. Using the standard engineering interpretation of mathematics, this is an impossible result. On the other hand, this has been a well-known result to mathematicians since the 1970's. The trick to this example is that there is vital information about the structure of the system that is being withheld. Let's now say our system is symmetric. By taking the transpose of each known quantity we can generate another known quantity. Now we have a clear picture that $8 - 2 * 2 = 4$. It is through the exploitation of structure in this

example that we can eliminate double the number of unknowns than we typically consider. For aerospace missions, symplectic mechanics exposes this information and the Marsden-Weinstein-Meyer reduction procedure is the recipe to reduce the system from 8 unknowns to 4 unknowns [118, 119].

The singular goal of this dissertation is create a general-purpose process for applying the Marsden-Weinstein-Meyer reduction procedure to aerospace missions [118, 119]. This novel approach to tackling general-purpose aerospace mission design will result in computational problems that are lower dimensional than the current state-of-the-art indirect methods provide. These computational problems ultimately require the numerical solution of fewer equations. This is a multi-step process that we break down into five unique sub-contributions in addition to some algebra.

Consider the presentation of the above example. It likely seemed unfair that vital information was provided *after* the main question had already been posed. This is precisely the limitation vector calculus and differential equations have in modern trajectory optimization. To break free from this limitation, we will cover some algebra in Chapter 2. Then, this will allow us to cover the five significant contributions of this dissertation:

- In Chapter 3, we visit indirect methods of mission design and trajectory optimization from a geometric control perspective. Several important definitions are reviewed before covering collocation and shooting methods on the differential geometric system.

- In Chapter 4, we explore how to rephrase complex design problems to be compatible with definitions given in Chapters 2 and 3. Derivatives are used in lieu of the typical algebraic operations from standard indirect methods thereby bypassing the prohibitive Pontryagin's minimum principle.

- In Chapter 5, we show how tools from differential geometry may be used to simplify design problems in a manner that is not possible with standard indirect methods.

- In Chapter 6, a modified collocation and shooting algorithm that operates on reduced manifolds is given and a sample problem with results is shown.
- In Chapter 7, we present a new method of constructing OCPs as well as associated Hamiltonian BVPs. This is a scalable strategy for constructing complicated path-constrained problems with compatibility with other strategies effectively future-proofing indirect methods.

Each of these contributions plays a unique role in the design of aerospace missions. By the end of this dissertation, we will have constructed a general-purpose process that takes an aerospace system designer's hypothetical mission and converts it into an NLP problem which may be solved using readily available NLP solvers. This process is shown in Fig. 1.6.

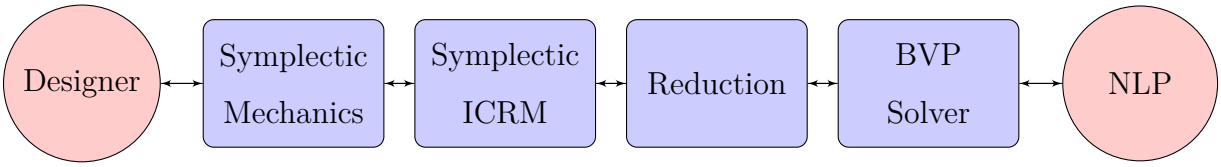


Figure 1.6. End-to-end mission design process as constructed in this dissertation.

2. ALGEBRAS

Various types of algebras are used throughout this dissertation that are unconventional in the typical engineering applications dominated by Gibbs (vector) algebra, but are becoming increasingly popular [123]. The purpose of this section is to give a quick introduction to some of their overarching rules, functionality, and purpose. Two algebras used heavily throughout are Grassmann (exterior) and differential graded algebras. Their purpose is to handle the modeling and implementation of algorithms, structures of manifolds and their charts. This allows the use of results from standard calculus and linear algebra in a more generalized manner. Lie algebras, and their associated groups, are primarily used for identifying and classifying symmetries and constants-of-motion into a group structure which will provide several topological results and serve as a guide for practical implementation. Rewriting many of our processes in these tensor-based algebras will give us two benefits.

1. Tensors are very efficient computational objects. Although the numerical aspects will be nearly identical, by using tensor-based algebras as our foundation we will create structurally consistent processes in Chapter 7.

2. Mathematical processes like reduction are inherently tensor-based due to their exploitation of differential structures on manifolds. Since with standard calculus, differential structure is not represented, application of reduction is nearly impossible without making simplifying assumptions. By using tensor-based algebras, the process we introduce in Chapter 5 will become enabled for general-purpose applications.

2.1 Tensor Algebra

Tensors have seen heavy use in engineering applications. Some examples include material stress [124], fluid mechanics [125], elasticity [126], electrodynamics [127], and countless other applications. Originally developed by Ricci and Levi-Civita in 1900 [128], tensors and their algebras are most famous for their applications in both special and general relativity. In this dissertations, tensors make up the foundation for many other algebras.

***Definition* 12** (Tensor Algebra)**.** Let W be a vector space over a field, K. Then, an n-graded tensor is defined as the following direct product

$$\bigotimes^{n} W = W \otimes W \oplus \cdots \otimes W \tag{2.1}$$

The tensor algebra is the set of all tensors and their operations as $n \to \infty$

$$\bigoplus_{i=0}^{\infty} \bigotimes^{i} W = K \oplus W \oplus W \otimes W \oplus \cdots \tag{2.2}$$

The multiplication for this algebra is defined as the follows

$$\otimes : \bigotimes^{n} W \times \bigotimes^{m} W \to \bigotimes^{n+m} W \tag{2.3}$$

End definition.

***Remark* 4.** The grade of a tensor can exceed the dimension of the underlying vector space. As such, the grade $n \to \infty$ is usually assumed. Consider $W = \mathbb{R}^2$ and the 3 tensors, $a\boldsymbol{e}_1$, $b\boldsymbol{e}_2$, and $c\boldsymbol{e}_2$ with $\boldsymbol{e}_1, \boldsymbol{e}_2 \in W$. Then the following grade-3 objects exist

$$abc\ \boldsymbol{e}_1 \otimes \boldsymbol{e}_2 \otimes \boldsymbol{e}_2 \neq abc\ \boldsymbol{e}_2 \otimes \boldsymbol{e}_1 \otimes \boldsymbol{e}_2 \neq abc\ \boldsymbol{e}_2 \otimes \boldsymbol{e}_2 \otimes \boldsymbol{e}_1 \tag{2.4}$$

End remark.

2.2 Grassmann Algebra

In addition to the now well-known linear algebra, the Grassmann (exterior) algebra was formalized by Hermann Grassmann in 1844 [129]. His original work was underappreciated, only being rediscovered by mathematicians between the 1860's and 1870's. The lack of reception was likely a contributing factor to his falling out from mathematics later in life [130]. However, his contributions were enormous, catching the attention of Hamilton himself [131]:

> "Grassmann is a great and most German genius; his view of space is at least as new and comprehensive as mine of time."
>
> (W. R. Hamilton)

Grassmann's role in mechanics were apparent from very early, having created a vector form for mechanics around 1832 [130]. Two key features of his algebra are its anti-symmetric structure and graded structure. In Section 1.1.2, we saw that the traditional form of Hamilton's equations appear as the following set of anti-symmetric equations

$$\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}}, \quad \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} \tag{2.5}$$

The anti-symmetric structure of the Grassmann algebra ensures that all constructions in this dissertation preserve the anti-symmetric structure of Hamilton's equations.

***Definition* 13** (Grassmann Algebra)**.** Given a vector space W, the n-graded Grassmann algebra is defined as the following quotient vector space

$$\bigwedge^{n} W = \bigotimes^{n} W / V_n \tag{2.6}$$

where V_n is a subspace of n-tensors and V_n contains all tensors that are constructed by tensor products in itself

$$x \otimes x \otimes \cdots \otimes x \in V_n, \quad x \in W \tag{2.7}$$

Additionally, V_n contains all tensors generated by transposes

$$x \otimes y \otimes \cdots \otimes z + y \otimes x \otimes \cdots \otimes z \in V_n, \quad x, y, z \in W \tag{2.8}$$

End definition.

This is a graded algebra whose elements are compatible with the operations of addition, associative multiplication, and scalar multiplication. While the addition and scalar multiplication are the usual operations, the associative multiplication operation in the Grassmann algebra is an anti-symmetric product referred to as the exterior, or wedge product.

***Definition* 14** (Wedge Product)**.** The wedge product, denoted $\wedge$, is a bilinear tensor product in the Grassmann algebra on a vector space, W, defined as follows

$$\begin{gathered} \wedge : \bigwedge^{n} W \times \bigwedge^{m} W \to \bigwedge^{n+m} W \\ \boldsymbol{e}_1 \wedge \boldsymbol{e}_2 = (-1)^{nm}\, \boldsymbol{e}_2 \wedge \boldsymbol{e}_1 \end{gathered} \tag{2.9}$$

where n and m are the grade of the input tensors

$$\begin{aligned} n &= \mathrm{Deg}(\boldsymbol{e}_1) \\ m &= \mathrm{Deg}(\boldsymbol{e}_2) \end{aligned} \tag{2.10}$$

End definition.

To give a concrete example, consider the space $W = \mathbb{R}^3$. Given two vectors of degree 1, we find $\mathrm{Deg}(\boldsymbol{e}_1 \wedge \boldsymbol{e}_2) = 2$ with $\boldsymbol{e}_1 \wedge \boldsymbol{e}_2$ representing a parallelogram with unique edges $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. This parallelogram is normal to the popular cross product of vectors, $\boldsymbol{e}_1 \times \boldsymbol{e}_2$. The Hodge star, $\star$, is a linear map sending an element of the Grassmann algebra to its dual representation. Using the Hodge star in $\mathbb{R}^3$, we can effectively decompose the cross product into the Hodge star and wedge product as shown in Eq. (2.11). This setup can also be seen visually in Fig. 2.1.

$$\star\,(\boldsymbol{e}_1 \wedge \boldsymbol{e}_2) = \boldsymbol{e}_1 \times \boldsymbol{e}_2 = \boldsymbol{e}_3, \quad W = \mathbb{R}^3 \tag{2.11}$$

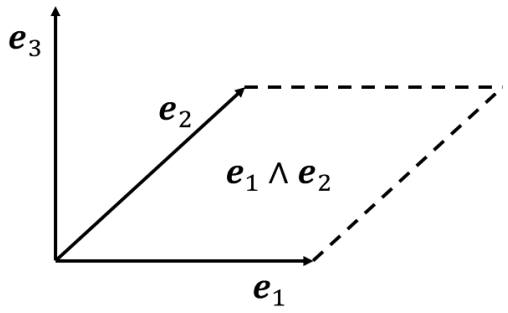


Figure 2.1. Dual representation of the parallelogram defined by the wedge product.

***Remark* 5.** Since Grassmann algebra is a quotient of a tensor algebra, it itself is a tensor algebra. As expected, the grade of a tensor can exceed the dimension of the underlying vector space. However, the grade $n \to \infty$ is usually never assumed. Consider $W = \mathbb{R}^2$ and the 3 tensors, $a\ \boldsymbol{e}_1$, $b\ \boldsymbol{e}_2$, and $c\ \boldsymbol{e}_2$ with $\boldsymbol{e}_1, \boldsymbol{e}_2 \in W$. Then the following grade-3 objects exist

$$abc\ \boldsymbol{e}_1 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_2 = abc\ \boldsymbol{e}_2 \wedge \boldsymbol{e}_1 \wedge \boldsymbol{e}_2 = abc\ \boldsymbol{e}_2 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_1 = 0\ \boldsymbol{e}_1 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_2 \tag{2.12}$$

These objects are dataless and therefore exploring tensors of grade $n > \dim(W)$ is typically not a useful exercise. *End remark.*

In general, the Grassman algebra is used to introduce geometric structure into optimal control systems by governing the behavior of vectors. The anti-symmetric nature of this algebra ensures the structure of Hamilton's equations are always preserved.

2.3 Lie Algebra

Lie algebras are used extensively in particle physics, quantum mechanics, and control systems with tremendous success. First introduced by Sophus Lie in the late 1870's, then subsequently by Wilhelm Killing, Lie algebras deal with infinitesimal transformations and their corresponding group structure [132].

***Definition* 15** (Lie algebra)**.** A Lie algebra is a vector space, $\mathfrak{g}$, together with an alternating bilinear bracket operation known as the Lie bracket

$$[\cdot,\cdot]_{\mathrm{L}} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g} \tag{2.13}$$

For some elements of the Lie algebra, f, g, and h, the following conditions are satisfied

1. $[f, f]_{\mathrm{L}} = 0$
2. $[f + g, h]_{\mathrm{L}} = [f, h]_{\mathrm{L}} + [g, h]_{\mathrm{L}}$
3. $[f, [g, h]_{\mathrm{L}}]_{\mathrm{L}} + [g, [h, f]_{\mathrm{L}}]_{\mathrm{L}} + [h, [f, g]_{\mathrm{L}}]_{\mathrm{L}} = 0$

The last condition is known as Jacobi's identity. *End definition.*

One extremely common Lie algebra is the vector space $\mathbb{R}^3$ together with the cross product operation as the Lie bracket. Another frequently used Lie algebra is the space of smooth functions together with the Poisson bracket, $[\cdot,\cdot]_{\mathrm{P}} : C^\infty \times C^\infty \to C^\infty$. This is commonly referred to as the Poisson algebra. In Chapter 5, we will asses the functional dependence of constants-of-motion under the Poisson bracket. This allows us to impose a very important restriction on the solution process that can be easily evaluated in an automated fashion.

For every Lie algebra, there is a corresponding Lie group. Lie groups provide important information about the topological characteristics of the underlying manifolds in optimal control systems and are leveraged to guarantee the existence of manifolds with desirable properties. Typically, Marsden-Weinstein-Meyer reduction takes quotients by Lie groups; however, in Chapter 5 we will introduce a strategy that is applied at the level of the algebra.

2.4 Co-algebras

In general, algebras provide the majority of the machinery required for calculations. However, there are a few instances of the dual representation of algebras, so-called co-algebras, that are crucial.

***Definition* 16** (Co-Algebra)**.** A co-algebra is the dual of an algebra such that the algebra's structure maps are reversed. That is, the following general algebra multiplication rule

$$[\cdot,\cdot] : W \otimes W \to W \tag{2.14}$$

and unit

$$e : K \to W \tag{2.15}$$

are reversed to form the following maps

$$\begin{aligned} \delta &: W \to W \otimes W \\ e^* &: W \to K \end{aligned} \tag{2.16}$$

This setup is shown in the following diagram *End definition.*

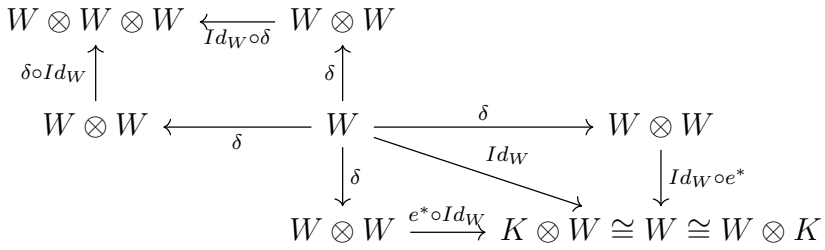


Figure 2.2. Commutative diagram identifying $(W \otimes W) \otimes W \cong W \otimes (W \otimes W)$ and $K \otimes W \cong W \otimes K \cong W$.

In Section 2.5, we will combine the concepts of differentials and geometry in a single algebra. This algebra leads to a natural definition of an important co-algebra.

2.5 Differential Graded Algebra

In engineering, geometry and structure is largely ignored whereas gradient information is heavily used for analysis. To ensure compatibility between geometric structure and gradient information, we use the concept of differential (graded) algebra first introduced by Joseph Ritt in 1950 [133]. Differential graded algebra is an

associative algebra with the operations of addition, multiplication, and scalar multiplication. As the name suggests, this algebra has both a graded structure and a differential structure, building on the tensor algebra introduced in Section 2.1. Additionally, it is equipped with a derivation, D, that obeys the Leibniz (product) rule

$$D(xy) = D(x)y + xD(y) \tag{2.17}$$

Note that the derivation, D, is abstract and specific implementations will be defined as necessary. We now introduce graded structure to the Leibniz rule to arrive at differential graded algebra.

***Definition* 17** (Derivation)**.** A derivation is a graded operation on an algebra that obeys the following graded Leibniz rule

$$D(xy) = D(x)y + (\epsilon)^{\mathrm{Deg}(x)\mathrm{Deg}(D)}x(Dy) \tag{2.18}$$

For $\epsilon = 1$, D is a derivation and for $\epsilon = -1$, D is an anti-derivation. *End definition.*

The most important graded anti-derivation is the exterior derivative.

***Definition* 18** (Exterior Derivative)**.** The exterior derivative is an arbitrary dimensional counterpart to the common differential of a function. It is defined in terms of the following invariant formula

$$\begin{aligned} \mathbb{d}\omega(X_0, X_1, \cdots, X_n) = &\sum_i (-1)^i X_i(\omega(X_0, \cdots, \hat{X}_i, \cdots, X_n)) \\ &+ \sum_{i<j} (-1)^{i+j} \omega([X_i, X_j]_{\mathrm{L}}, X_0, \cdots, \hat{X}_i, \cdots, \hat{X}_j, \cdots, X_n) \end{aligned} \tag{2.19}$$

where $[\cdot, \cdot]_{\mathrm{L}}$ is the Lie bracket of vector fields and the hat, $\hat{X}_i$, is the removal of that particular element. *End definition.*

***Remark* 6.** The exterior derivative is a grade 1 anti-derivation that is the inverse transpose of the Lie bracket

$$\mathbb{d} : \mathfrak{g}^* \to \mathfrak{g}^* \wedge \mathfrak{g}^* \tag{2.20}$$

More generally, the exterior derivative's signature is

$$\mathbb{d} : \bigwedge^{\bullet} \mathfrak{g}^* \to \bigwedge^{\bullet+1} \mathfrak{g}^* \tag{2.21}$$

Since $\mathbb{d}^2 = 0$, this forms a co-chain complex known as the de Rham complex

$$\cdots \xrightarrow{\mathbb{d}} \bigwedge^0 \mathfrak{g}^* \xrightarrow{\mathbb{d}} \bigwedge^1 \mathfrak{g}^* \xrightarrow{\mathbb{d}} \bigwedge^2 \mathfrak{g}^* \xrightarrow{\mathbb{d}} \bigwedge^3 \mathfrak{g}^* \xrightarrow{\mathbb{d}} \cdots$$

Figure 2.3. The de Rham complex on $\mathfrak{g}^*$.

End remark.

The exterior derivative together with $\mathfrak{g}^*$ are a differential graded algebra that also forms a Lie co-algebra. These tools, together with the Lie algebra on $\mathfrak{g}^*$, which is the Poisson algebra, will later be used to build a tensor form of Noether's theorem that can easily be automated.

2.6 Summary

In this chapter, we introduced the algebras that will be commonly used throughout this dissertation. We first introduced the idea of tensors and tensor algebra. Building off of tensors, we introduced the Grassmann algebra which uses a geometric construction that will become crucial in later chapters. Next, we gave an overview of Lie algebra and differential graded algebra which are important algebras that govern the manipulation of infinitesimal symmetries and constants-of-motion. These two algebras are related through the concept of a co-algebra and in Chapter 3, we will construct a method of transforming quantities between these algebras. After some reworking of current-state-of-the-art methods to a symplectic formulation in Chapter 4, we will then see the greater extent of the algebras' power in Chapter 5 where reduction is introduced. Reduction cannot be performed with standard calculus in general-purpose applications.

3. GEOMETRIC OPTIMAL CONTROL THEORY

In this chapter, optimal control theory will be rephrased from the typical Hamiltonian style of indirect methods to the language of differential geometry and, more specifically, symplectic manifolds. While the results of this section do not immediately offer any new information about optimal control systems, aside from a convenient tensorial version of Noether's theorem, the coordinate-free view of differential geometry will allow one to modify optimal control systems in a unique way. By rephrasing the system in this manner, the geometric structure of the system is exposed and will be exploited in Chapter 5 to lower a system's dimension to fewer equations than is traditionally possible.

The dualization of Problem Σ to Problem Λ yields a Hamiltonian BVP whose dynamics live in a symplectic manifold. In Section 3.3, we will see how this dualization is carried out. However, the Hamiltonian BVP generated contains control quantities and other terms adding complexity. In Section 3.2, we will look at symplectic manifolds in a traditional context where these additional complexities do not exist.

3.1 Differential Geometry

Differential geometry is an extremely broad and rich field of mathematics. Calculus, in the modern sense, was first introduced by Isaac Newton and Gottfried Leibniz in the mid 1600's, whereas geometry dates back to the ancient Egyptians around 2900 BC with axiomatic geometry typically attributed to Euclid of Alexandria around 300 BC. Differential geometry is a field of mathematics that merges differential calculus and geometry to study problems with elements of both fields. The majority of this content is adapted from Frederic Schuller's lectures on the geometrical anatomy of theoretical physics [134, 135], Vladimir Arnold's famous textbook on mathematical

methods of classical mechanics [136], and Stephanie Singer's digestible book on the role of symmetry in mechanics [137].

3.1.1 Topological Objects

Differential geometry is akin to the `Python` programming language in that they are generally regarded as gluing languages. At the local level, differential geometry uses various algebras, some of which were introduced in Chapter 2, to carry out operations explicitly. Similarly, the `FORTRAN` and `C` programming languages are commonly used to carry out computationally intensive operations that would be significantly slower in pure `Python`. However, a developer typically writes high-level `Python` code without ever being exposed to the underlying `C` and `FORTRAN` doing the heavy lifting for them. In this same spirit, a differential geometer works at a high-level with tools from an area known as topology. Using topology enables a differential geometer to explore mathematical constructs without being held back by the low-level implementation. The central tool in topology that enables this type of study is the manifold.

***Definition* 19** (Manifold)**.** A manifold, M, is a topological space where there exists an open Euclidean neighborhood, $\mathbb{R}^\bullet$, about every point.

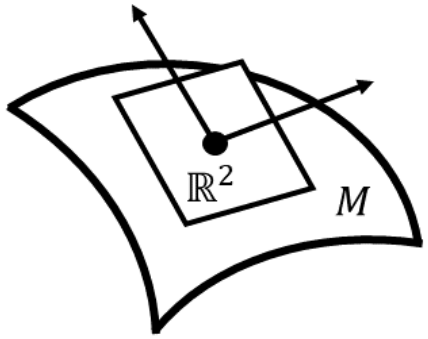


Figure 3.1. A 2-dimensional manifold.

End definition.

***Definition* 20** (Chart)**.** Let M be a manifold. Then, a chart is a mapping from an open subset $V \in M$ to an open subset of a Euclidean space, $\mathbb{R}^\bullet$. *End definition.*

***Definition* 21** (Smooth Manifold)**.** Smooth manifolds are manifolds where all mappings are infinitely differentiable. *End definition.*

These tools enable gradient information to be encoded as vector fields on manifolds and optimal motion as flows. We note that not all optimal motion is smooth as many practical design problems contain control laws with bang-bang and singular arcs. These scenarios are ignored for now and will be treated in Chapter 7. To describe how optimal control systems are modeled using differential geometry, we first introduce M as a smooth manifold and define a curve.

***Definition* 22** (Curve)**.** A curve is a function of a single variable on a real line mapping to M.

$$\gamma(t) : I \subset \mathbb{R} \to M \tag{3.1}$$

End definition.

Assume M is embedded in $\mathbb{R}^m$ and there exists a point $x \in M$. Then, given some function, $f : M \to \mathbb{R}$, we can evaluate the derivative of f along $\gamma(t)$ at point x_0 by computing

$$\frac{\mathbb{d}}{\mathbb{d}t}\bigg|_{x_0} f = \lim_{h \to 0} \frac{f(\gamma(t_{x_0} + h)) - f(\gamma(t_{x_0}))}{h} \tag{3.2}$$

By choosing a local chart $\{x_1, \cdots, x_n\}$, we can rewrite our derivative as the following

$$\frac{\mathbb{d}}{\mathbb{d}t} f = \frac{\mathbb{d}x_1}{\mathbb{d}t}\frac{\partial}{\partial x_1} f + \cdots + \frac{\mathbb{d}x_n}{\mathbb{d}t}\frac{\partial}{\partial x_n} f \tag{3.3}$$

Then, eliminating the arbitrary function f, we have

$$\frac{\mathbb{d}}{\mathbb{d}t} = \frac{\mathbb{d}x_1}{\mathbb{d}t}\frac{\partial}{\partial x_1} + \cdots + \frac{\mathbb{d}x_n}{\mathbb{d}t}\frac{\partial}{\partial x_n} \tag{3.4}$$

Thus, $\mathbb{d}/\mathbb{d}t$ is a linear combination of partial derivatives, $\partial/\partial x_i$. Given a vector, $\boldsymbol{p}$, at point x_0, we write it in terms of the previous chart to find

$$\boldsymbol{p} = p_1 \frac{\partial}{\partial x_1} + \cdots + p_n \frac{\partial}{\partial x_n} \tag{3.5}$$

In Eq. (3.5) we used the partial derivatives, ∂_{x_i}, as basis vectors. Additionally, $\boldsymbol{p}$ may be used as a function in that $\boldsymbol{p}(f)$ evaluates the gradient of f along $\boldsymbol{p}$. In this case, $\boldsymbol{p}$ is a rank 1 contravariant tensor and degree 0 derivation. Consider the set of all possible curves, $\gamma_i(t)$, through the point, x_0. The collection of all such tangent vectors, $\boldsymbol{p}_i$, defines the tangent space of M at x_0, $T_{x_0}M$. While tangent spaces are effective in describing local phenomena, manifolds by nature are global structures with infinitely many tangent spaces. In an effort to describe more global characteristics, we now define the tangent bundle of M, denoted TM.

***Definition* 23** (Tangent Bundle)**.** Let M be a manifold. Then, the tangent bundle of M, TM, is defined as the disjoint union of all tangent spaces of M, that is

$$TM \equiv \bigsqcup_{x \in M} T_x M \tag{3.6}$$

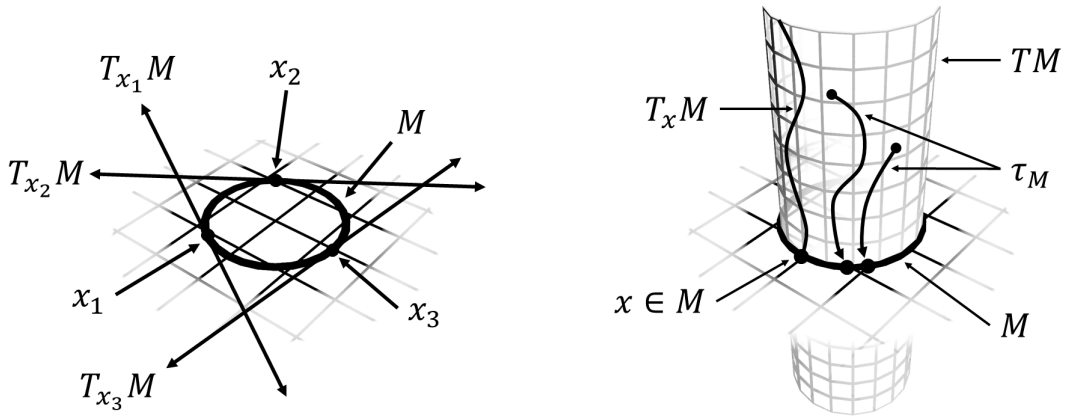
The tangent bundle is equipped with a natural projection $\tau_M : TM \to M$ and local vector basis $\{\partial_{x_1}, \cdots, \partial_{x_n}\}$. *End definition.*

The tangent bundle is a disjoint union in the sense that if some point, $p \in TM$, is chosen, the fiber over M associated with p may be located. The fiber is the topological space $\tau_M^{-1}(x)$ where $x \in M$. That is, it is possible to identify some tangent space $T_xM = \tau_M^{-1}(x)$ that contains a point $p \in TM$ with base point $x \in M$. This setup is illustrated in Fig. 3.2. Using the tangent bundle, we can define a vector field.

***Definition* 24** (Vector Field)**.** Let M be a manifold and $\tau_M : TM \to M$ be its tangent bundle. Then a vector field, $X : M \to TM$, is a section of the tangent bundle such that Fig. 3.3 commutes.

End definition.

Vector fields provide valuable information mapping from a manifold to its tangent bundle. In the case of dynamical systems, vector fields are the equations-of-motion that govern motion of the system. Some illustrations of this setup are in Fig. 3.4.



(a) A manifold with 3 tangent spaces (b) Tangent spaces collected into a bundle

Figure 3.2. The tangent bundle and some associated components. Note that since $\dim(M) = 1$, then $\dim(TM) = 2$

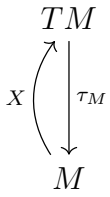
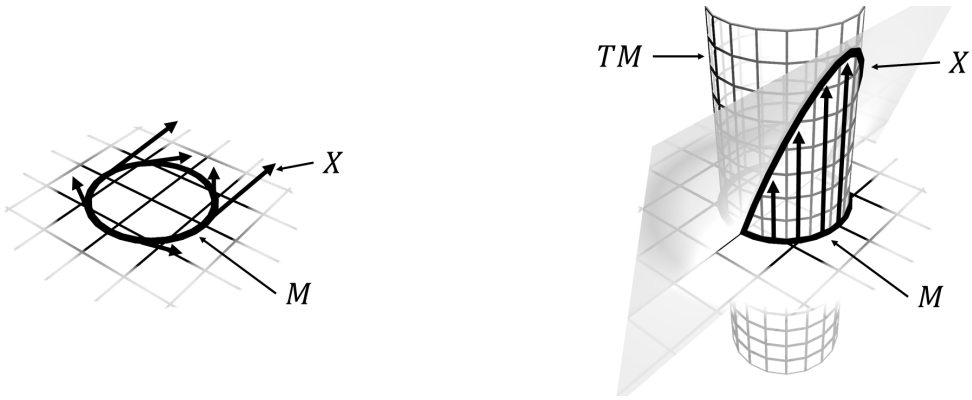


Figure 3.3. Vector field on M

Vector fields are typically denoted $X \in \mathfrak{X}(M)$. However, we will identify vector fields as $X \in \mathfrak{g}$. This is because the space of vector fields together with the Lie bracket form a Lie algebra. This bracket operation, sometimes called the Jacobi-Lie bracket, is defined in Eq. (3.7).

$$\begin{aligned} [\cdot,\cdot]_{\mathrm{L}} : \mathfrak{g} \times \mathfrak{g} &\to \mathfrak{g} \\ (X_1, X_2) &\mapsto X_1(X_2) - X_2(X_1) \end{aligned} \tag{3.7}$$

From this definition, we begin to see how the algebraic tools defined in Chapter 2 are constructed as a result of the topological definitions. Recall we also defined a Lie co-algebra in Chapter 2. One would expect there to exist a co-algebra to the Lie algebra of vector fields. This requires the dual representation of a manifold's tangent



(a) A vector field modeled with tangent vectors on M

(b) Vector field modeled with basis vectors on TM

Figure 3.4. A vector field as the section of the tangent bundle

spaces which are called co-tangent spaces. A co-tangent space is defined as the set of all linear functionals on T_xM denoted T_x^*M. Therefore, by choosing an element $\theta \in T_x^*M$, there is a map $\theta : T_xM \to \mathbb{R}$. In a similar fashion to how the tangent bundle was defined, we can now define the co-tangent bundle and co-vector fields.

***Definition* 25** (Co-Tangent Bundle)**.** Let M be a manifold and $\tau_M : TM \to M$ be its tangent bundle. Then, the co-tangent bundle of M, T^*M, is defined as the disjoint union of all co-tangent spaces on M, that is

$$T^*M \equiv \bigsqcup_{x \in M} T_x^*M \tag{3.8}$$

The co-tangent bundle is equipped with a natural projection $\pi_M : T^*M \to M$ and local co-vector basis $\{\mathrm{d}x_1, \cdots, \mathrm{d}x_n\}$. *End definition.*

***Definition* 26** (Co-Vector Field)**.** Let M be a manifold and $\pi_M : T^*M \to M$ be its co-tangent bundle. Then a co-vector field, $\theta : M \to T^*M$, is a section of the co-tangent bundle such that Fig. 3.5 commutes.

End definition.

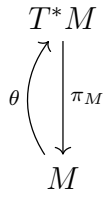


Figure 3.5. Co-vector field on M

Co-vector fields are typically denoted $\theta \in \Omega(M)$. However, we will identify co-vector fields as $\theta \in \mathfrak{g}^*$. This is because the space of co-vector fields together with the exterior derivative form a Lie co-algebra.

$$\mathbb{d} : \mathfrak{g}^* \to \mathfrak{g}^* \wedge \mathfrak{g}^* \tag{3.9}$$

Recall from Chapter 2, the exterior derivative defined a co-chain complex of differential forms known as the de Rham complex. In this framework, co-vector fields of grade k on M are typically denoted $\Omega^k(M)$, while we will opt to retain $\bigwedge^k \mathfrak{g}^*$. Having defined both a Lie algebra and Lie co-algebra from an arbitrary manifold, M, one might hypothesize that there exists a map between the two. Collecting what we know about tangent and co-tangent bundles, we can represent this as the commutative diagram in Fig. 3.6.

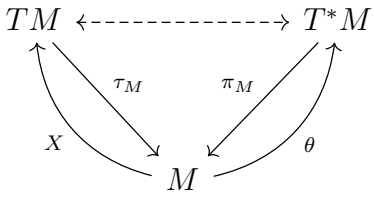


Figure 3.6. A potential mapping between the tangent and co-tangent bundles of manifold M.

Curiously, the tangent and co-tangent bundles are isomorphic, however co-tangent bundles carry canonical structures whereas tangent bundles do not. This is similar to how Lagrangian and Hamiltonian formulations of mechanics describe similar motion

at the same dimension, but are structured differently. For example, take M to be an n-dimensional smooth manifold with local coordinates $\boldsymbol{x}$. Let $(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda})$ be the induced coordinates on the co-tangent bundle, T^*M, and let $\pi_M : T^*M \to M$ be the natural projection defined as $(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}) \mapsto (\tilde{\boldsymbol{x}})$. This induces a $C^\infty(M)$-linear map on 1-forms denoted π_M^* known as the pullback by π_M, that is

$$\begin{aligned} \pi_M^* : \Omega^1(M) &\to \Omega^1(T^*M) \\ (\theta) &\mapsto (\pi_M^*(\theta)) \end{aligned} \tag{3.10}$$

Consider $\pi_M^* \, \mathbb{d}\boldsymbol{x} = \mathbb{d}\tilde{\boldsymbol{x}} = \mathbb{d}\boldsymbol{x}$ by choice of $\tilde{\boldsymbol{x}}$. Define a one-form on T^*M as $\pi^*_{M,(x,\lambda)}\boldsymbol{\lambda}$ mapping each point, $(x, \lambda) \in T^*M$, to the co-vector $(\lambda, 0) \in T^*_{(x,\lambda)}T^*M$, an element of the co-tangent space at (x, λ) on the co-tangent bundle. So using the formula for π_M^*, we have

$$\begin{aligned} \pi^*_{M,(x,\lambda)}\boldsymbol{\lambda} &= \boldsymbol{\lambda} \circ \pi_M \, \mathbb{d}\boldsymbol{x} \\ &= \boldsymbol{\lambda} \, \mathbb{d}\boldsymbol{x} + 0 \, \mathbb{d}\boldsymbol{\lambda} \\ &= \boldsymbol{\lambda} \, \mathbb{d}\boldsymbol{x} \end{aligned} \tag{3.11}$$

The quantity $\boldsymbol{\lambda} \, \mathbb{d}\boldsymbol{x}$ is sometimes referred to as the Liouville-Poincaré one-form [138]. This definition leads to a natural construction of symplectic manifolds which we will cover in Section 3.2.

3.2 Symplectic Manifolds

In this section, we will give a brief introduction to symplectic manifolds and their application to Hamiltonian dynamical systems. In addition to the Liouville-Poincaré one-form, there exist various tensor quantities on symplectic manifolds. We will show how we can use these quantities to create applied tensor versions of both Noether's and Liouville's theorems. To begin, we recall the Liouville-Poincaré one-form

$$\theta = \boldsymbol{\lambda} \, \mathbb{d}\boldsymbol{x} \tag{3.12}$$

Since $\theta \in \bigwedge^1 \mathfrak{g}^* \cong \mathfrak{g}^*$, then $\mathbb{d}\theta \in \bigwedge^2 \mathfrak{g}^*$ is well-defined and referred to as the standard symplectic form.

***Definition* 27** (Standard Symplectic Form)**.** The standard symplectic form, ω, is a closed and non-degenerate differential two-form defined as

$$\omega \equiv -\,\mathbb{d}\theta = \mathbb{d}\boldsymbol{x} \wedge \mathbb{d}\boldsymbol{\lambda} \tag{3.13}$$

End definition.

Consider a smooth manifold M. This manifold's co-tangent bundle $\Lambda = T^*M$ together with the standard symplectic form yield a symplectic manifold.

***Definition* 28** (Symplectic Manifold)**.** A symplectic manifold is the pair (Λ, ω), where Λ is an even dimensional smooth manifold equipped with a closed and non-degenerate differential two-form, ω. *End definition.*

Two restrictions on some form ω for it to be a symplectic form are that it is closed and non-degenerate. The closed condition provides the constraint that ω is unchanging under flows on Λ, or $\mathbb{d}\omega = 0$. This is usually easy to satisfy using anti-symmetric constructions with the wedge product. The non-degeneracy condition guarantees that for a function $H \in \Lambda$ there exists a unique vector field corresponding to every $\mathbb{d}H$. Together, these conditions can be written in a nice formula using the Lie derivative, but require the interior derivative.

***Definition* 29** (Interior Derivative)**.** The interior derivative is the degree -1 anti-derivation $\iota : \mathfrak{g} \times \bigwedge^\bullet \mathfrak{g}^* \to \bigwedge^{\bullet-1} \mathfrak{g}^*$, defined as

$$\begin{aligned} \iota : \mathfrak{g} \times \bigwedge^{\bullet} \mathfrak{g}^* &\to \bigwedge^{\bullet-1} \mathfrak{g}^* \\ (X, \omega) &\mapsto \omega(X, \cdot, \cdots, \cdot) \end{aligned} \tag{3.14}$$

End definition.

Using the interior derivative and Cartan's magic formula, we write the Lie derivative as

$$\pounds_{X_H}\omega = \mathbb{d}\left(\iota_{X_H}\omega\right) + \iota_{X_H}\,\mathbb{d}\omega \tag{3.15}$$

Since ω is closed this simplifies to

$$\pounds_{X_H}\omega = \mathbb{d}\left(\iota_{X_H}\omega\right) \tag{3.16}$$

Noting this form is exact, integration yields the symplectic equation, or the geometric form of Hamilton's equations where H is a Hamiltonian function.

***Definition* 30** (Symplectic Equation)**.** Given a symplectic manifold, (Λ, ω), and a Hamiltonian function, $H : \Lambda \to \mathbb{R}$, the standard symplectic equation is as follows

$$\iota_{X_H}\omega = \mathbb{d}H \tag{3.17}$$

The flow lines of the flow generated by X_H are trajectories of phase space.

End definition.

Eq. (3.17) is relatively easy to solve, especially with ω in its standard form. This equation completely encodes the classical form of Hamilton's equations without explicitly referencing coordinates. By using the symplectic equation, we can ensure that the symplectic structure is preserved with any arbitrary transformation. This will be done in Chapter 4, thus generating a slightly more difficult version of the symplectic equation that uses a non-standard symplectic form.

In Eq. (3.17), the differential structure ω is directly used to define X_H as opposed to the traditional methods in Section 1.1.2 which rely on variational methods. Since the differential structure is in an explicit tensorial form instead of an implicit variational form, one might expect there to be a tensorial analog to the variational Noether's theorem. To introduce this concept, we first need to introduce the tensorial versions of symmetries and constants-of-motion.

***Definition* 31** (Symmetry (2))**.** Given a symplectic manifold, (Λ, ω), a vector field $S : \Lambda \to T\Lambda \cong \mathfrak{g}$, and a Hamiltonian function, $H : \Lambda \to \mathbb{R}$, then S is a symmetry if it leaves both H and ω invariant under its flow. *End definition.*

***Definition* 32** (Constant-of-Motion)**.** Given a symplectic manifold, (Λ, ω) and a Hamiltonian function, $H : \Lambda \to \mathbb{R}$, a constant-of-motion is a quantity $J : \Lambda \to \mathbb{R} \cong \mathfrak{g}^*$ that is invariant under the flow generated by X_H. *End definition.*

The co-domain of a symmetry is the Lie algebra of vector fields and the co-domain of constants-of-motion are a Lie co-algebra. Given a symmetry, S, or a constant-of-motion, J, we can generate its corresponding constant-of-motion or symmetry using the differential structure. In the case where the symmetry is provided, since $\int_\Lambda : \bigwedge^\bullet \Lambda \to \bigwedge^{\bullet-1} \Lambda$, and $\omega : \mathfrak{g} \cong T\Lambda \to T^*\Lambda \cong \bigwedge^1 \mathfrak{g}^*$, we can define the Noether-like map $N = \int_\Lambda \circ \omega$, that is

$$\begin{aligned} N : \mathfrak{g} &\to \mathfrak{g}^* \\ S &\mapsto \int_\Lambda \iota_S \omega \end{aligned} \tag{3.18}$$

From this map, we have

$$N(S) = J \tag{3.19}$$

***Remark* 7.** In general, the Hamiltonian function governing the motion of a dynamical system is a constant-of-motion provided that it is not an explicit function of the independent variable. Therefore, the Noether-like map, N, is an integrated form of the symplectic equation in the sense that $N(X_H) = H$. *End remark.*

In general, it is easier to identify symmetries of a system as opposed to constants-of-motion due to their non-integrated form. However, there are plenty of cases where the constants-of-motion are easily identifiable. In these cases, having a Noether-like map N^* that sends constants-of-motion to their corresponding symmetries is desirable. To do this, we apply the same logic in reverse. Instead of searching for a map $\mathfrak{g} \to \mathfrak{g}^*$, we want to find the map $\mathfrak{g}^* \to \mathfrak{g}$. This can easily be done using the induced Poisson bracket from ω.

***Definition* 33** (Induced Poisson Bracket [139])**.** Given a symplectic manifold, (Λ, ω), the Poisson bracket induced by ω is the following quantity

$$[\cdot, \cdot]_{\mathrm{P}} : \mathfrak{g}^* \times \mathfrak{g}^* \to \mathfrak{g}^* \tag{3.20}$$

Such that $[F, G]_{\mathrm{P}} = \omega(X_F, X_G)$. *End definition.*

Using the induced Poisson bracket of ω, we can define the Noether-like map

$$\begin{aligned} N^* : \mathfrak{g}^* &\to \mathfrak{g} \\ J &\mapsto [\cdot, J]_{\mathrm{P}} \end{aligned} \tag{3.21}$$

***Remark* 8.** In general, the flow of a dynamical system is a symmetry of the phase space provided that the Hamiltonian function generating said flow is a constant-of-motion. Therefore, the Noether-like map, N^*, is a form of the symplectic equation in the sense that $N^*(H) = X_H$. *End remark.*

From these 2 maps, N and N^*, we have an applied tensorial form of Noether's theorem. This structure is desirable since it is significantly easier to automate in digital software than the variational form. Additionally, both N and N^* are essentially rearranged versions of the symplectic equation. Therefore, local application of the symplectic equation preserves Noether's theorem.

While Noether's theorem and the symplectic equation yield invaluable information about the dynamical structure of a system, these results typically can only be applied locally in practical scenarios. That is, in a typical aerospace mission, it is easy to generate infinitesimal symmetries from constants-of-motion, but generating the symmetries non-infinitesimal form may be prohibitively expensive or impossible. To overcome this, we will transform the system to a new coordinate system where a full symmetry is identified by a single state. In these cases, we can use Liouville's theorem to assist.

***Definition* 34** (Liouville's Theorem)**.** Given a symplectic manifold, (Λ, ω), and a Hamiltonian function, $H : \Lambda \to \mathbb{R}$, then Liouville's theorem necessarily states that the volume of an infinitesimal area of phase space is invariant under F_t, which is the flow generated by X_H

$$\int_{\Lambda} F_t^* \omega = \int_{\Lambda} \omega \tag{3.22}$$

End definition.

Since coordinates of phase-space are well-defined by the dualization process, this result may not appear to be immediately useful. Where Liouville's theorem shines is in defining so-called canonical transformations, or transformations that preserve the differential structure globally. For instance, consider two symplectic manifolds, (Λ_1, ω_1) and (Λ_2, ω_2) with local coordinates $(\boldsymbol{x}, \boldsymbol{\lambda}) \in \Lambda_1$ and $(\boldsymbol{q}, \boldsymbol{p}) \in \Lambda_2$, then a slight generalization of Liouville's theorem tells us the following

$$\int_{\Lambda_1} \omega_1 = \int_{\Lambda_2} \omega_2 \tag{3.23}$$

In other words, transformations *between* the manifolds Λ_1 and Λ_2 must preserve the infinitesimal volume of phase-space. If these manifolds are the dynamical manifolds of a Hamiltonian BVP, then this ensures solutions to Problem Λ_2 indirectly solve Problem Λ_1. Expanding the standard symplectic form for both of these manifolds, we have

$$\int_{\Lambda_1} \mathbb{d}\boldsymbol{x} \wedge \mathbb{d}\boldsymbol{\lambda} = \int_{\Lambda_2} \mathbb{d}\boldsymbol{q} \wedge \mathbb{d}\boldsymbol{p} \tag{3.24}$$

Integrating both sides, we find the homogeneous equation of canonical transformations

$$-\boldsymbol{\lambda}\, \mathbb{d}\boldsymbol{x} = -\boldsymbol{p}\, \mathbb{d}\boldsymbol{q} \tag{3.25}$$

Rearranging these terms to solve for $\mathbb{d}\boldsymbol{q}$, we have

$$\mathbb{d}\boldsymbol{q} = \frac{\boldsymbol{\lambda}}{\boldsymbol{p}}\, \mathbb{d}\boldsymbol{x} \tag{3.26}$$

Integrating both sides, we have the following definition of $\boldsymbol{q}$

$$\boldsymbol{q} \equiv \int_{\Lambda_1} \frac{\boldsymbol{\lambda}}{\boldsymbol{p}}\, \mathbb{d}\boldsymbol{x} \tag{3.27}$$

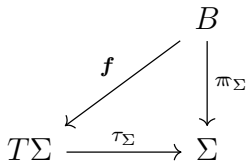
Therefore, by preservation Liouville's theorem, Eq. (3.27) defines $\boldsymbol{q}$ as a function of mixed coordinate functions $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{p})$. At this time, $\boldsymbol{p}$'s definition is not apparent; however, we will use Eq. (3.27) in Chapter 5 to assist in defining a new set of coordinates in reduction.

3.3 Nonliner Control Systems and Optimal Control

Up until this point, we have only studied dynamical systems from a geometric point of view. While interesting in their own regard, the general class of aerospace missions fall under the generalization known as optimal control theory. The introduction of control variables presents a new coordinate function; however, this is not a coordinate on a dynamical system's manifold, nor is it on the tangent or co-tangent bundles. In fact, the control variables yield a new topological object called the input bundle. Take Σ as a manifold, then the input bundle is defined as follows

$$\begin{aligned} \varpi_\Sigma : B &\to \Sigma \\ (\boldsymbol{x}, \boldsymbol{u}) &\mapsto \boldsymbol{x} \end{aligned} \tag{3.28}$$

A traditional nonlinear control system is the tuple $(B, \Sigma, \varpi_\Sigma, \boldsymbol{f})$. The user-supplied vector field is defined as $\boldsymbol{f} : B \to T\Sigma$. Considering the natural projection of $T\Sigma$ on Σ, $\tau_\Sigma : T\Sigma \to \Sigma$, then $\varpi_\Sigma = \tau_\Sigma \circ \boldsymbol{f}$. This setup is summarized in the following commutative diagram



***Definition* 35** (Trajectory)**.** A trajectory is a curve, $\gamma(t) : I \subset \mathbb{R} \to B$, that is also an integral curve of $\boldsymbol{f}$. The requirement that γ is an integral curve of $\boldsymbol{f}$ provides the following constraint

$$\frac{\mathrm{d}}{\mathrm{d}t}(\varpi_\Sigma \circ \gamma) = \boldsymbol{f} \circ \gamma \tag{3.29}$$

End definition.

Given a nonlinear control system, Σ, and a path cost, $L : B \to \mathbb{R}$, define the optimal control problem as

$$\text{Problem } \Sigma \begin{cases} \min\limits_{\boldsymbol{u}} K = \int_{t_0}^{t_f} L \circ \gamma \, \mathbb{d}t \\ \text{Subject to: } \dfrac{\mathbb{d}}{\mathbb{d}t}(\pi_\Sigma \circ \gamma) = \boldsymbol{f} \circ \gamma \\ \boldsymbol{\phi} \circ \gamma(t_0) = 0 \\ \boldsymbol{\xi} \circ \gamma(t_f) = 0 \\ \boldsymbol{g}_{\text{lower}} \leq \boldsymbol{g} \circ \gamma \leq \boldsymbol{g}_{\text{upper}} \end{cases} \tag{3.30}$$

Indirect methods of trajectory optimization refer to introducing a Hamiltonian structure to the optimization problem formulation and searching for solutions to the infinite-dimensional problem in a finite-dimensional phase space, or co-tangent bundle, $T^*\Sigma = \Lambda$ with $\pi_\Sigma : T^*\Sigma \to \Sigma$. Setting $\Lambda \equiv T^*\Sigma$ is the first step in defining Problem Λ.

***Remark* 9.** The notation "Problem M" refers to the fact that the main source of computational complexity lies in manifold M. This provides a convenient and topologically consistent means of integrating various transformations of our problem, such as dualization Problem $T^*\Sigma = \mathscr{D}(\text{Problem } \Sigma)$, into Ross and Fahroo's popular commutative diagram. *End remark.*

To implement this analytically, we introduce Lagrange multipliers as fiber coordinates, $\boldsymbol{\lambda} \in \pi_\Sigma^{-1}$ and augmenting the cost functional in Eq. (3.30) by adjoining dynamic constraints, $\boldsymbol{f}$. Considering these newly introduced coordinates, the dynamics in the indirect optimal control problem are written as follows

$$\iota_{X_H}\omega = \mathbb{d}H \tag{3.31}$$

For optimal control problems, $X_H : T \to T\Lambda$ is of the form

$$X_H = \frac{\mathbb{d}\boldsymbol{x}}{\mathbb{d}t}^T \frac{\partial}{\partial \boldsymbol{x}} + \frac{\mathbb{d}\boldsymbol{\lambda}}{\mathbb{d}t}^T \frac{\partial}{\partial \boldsymbol{\lambda}} \tag{3.32}$$

Note that the input-space manifold B contains the controls $\boldsymbol{u}$ which are not considered coordinates on M; therefore, they are accounted for in the differential of $H \in B/\Sigma \times \Lambda \cong E$

$$\mathbb{d}H = \frac{\partial H}{\partial \boldsymbol{x}}\,\mathbb{d}\boldsymbol{x} + \frac{\partial H}{\partial \boldsymbol{\lambda}}\,\mathbb{d}\boldsymbol{\lambda} + \frac{\partial H}{\partial \boldsymbol{u}}\,\mathbb{d}\boldsymbol{u} \tag{3.33}$$

From the above equations, we can see that a natural choice for $\mathbb{d}\boldsymbol{x}/\mathbb{d}t$ and $\mathbb{d}\boldsymbol{\lambda}/\mathbb{d}t$ are Hamilton's equations

$$\begin{aligned} \frac{\mathbb{d}\boldsymbol{x}}{\mathbb{d}t} &= \frac{\partial H}{\partial \boldsymbol{\lambda}} \\ \frac{\mathbb{d}\boldsymbol{\lambda}}{\mathbb{d}t} &= -\frac{\partial H}{\partial \boldsymbol{x}} \end{aligned} \tag{3.34}$$

Since optimal control problems naturally have a presymplectic structure, applying Eqs. 3.34 to Eqs. 3.31 leaves

$$0 = \frac{\partial H}{\partial \boldsymbol{u}}^T\,\mathbb{d}\boldsymbol{u} \tag{3.35}$$

To satisfy this equality, choose $\boldsymbol{u}^* : \Lambda \to B/\Sigma$ such that

$$\frac{\partial H}{\partial \boldsymbol{u}}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}^*(\boldsymbol{x}, \boldsymbol{\lambda})) = 0 \tag{3.36}$$

Simple problems return relatively simple expressions for $\partial H/\partial \boldsymbol{u}$, but complicated problems return increasingly complex algebraic expressions and may not admit a symplectic structure with the current set of relations. For these complicated cases, $\boldsymbol{u}^*$ will not have a single explicit solution and a different technique will be presented in Ch. 4.

3.4 Collocation

Collocation is a numerical method of solving optimal control problems. It is based on the assumption that optimal trajectories may be approximated by piece-wise third-order polynomials. To define a third-order piece-wise polynomial, four numerical pieces of information are required. At each endpoint, or node, of a polynomial, there

is state position information and state gradient information. Taking $\boldsymbol{x}_i$ to be states at node i, the coefficients for each polynomial are then

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_i \\ \dot{\boldsymbol{x}}_i \\ \boldsymbol{x}_{i+1} \\ \dot{\boldsymbol{x}}_{i+1} \end{bmatrix} \tag{3.37}$$

To ensure the polynomial is a high quality approximation of the underlying dynamical system, the predicted gradient at the midpoint of each polynomial is compared to the actual gradient from the equations-of-motion. The error between these two values vanishes when the curve is a high quality prediction, and therefore this condition is included as a constraint into an NLP-solver. The predicted values in between each node are

$$\begin{aligned} \tilde{\boldsymbol{x}}_{i+1/2} &= \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) + \frac{t_f(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_{i+1})}{8(N_{total} - 1)} \\ \dot{\tilde{\boldsymbol{x}}}_{i+1/2} &= -\frac{3}{2}\frac{(N_{total} - 1)}{t_f(\boldsymbol{x}_i - \boldsymbol{x}_{i+1})} - \frac{1}{4}(\dot{\boldsymbol{x}}_i + \dot{\boldsymbol{x}}_{i+1}) \end{aligned} \tag{3.38}$$

Using these predicted values, the following BVP represents a discretized computational problem that approximates the original BVP

$$\begin{aligned} \dot{\boldsymbol{x}}_{i+1/2} - \dot{\tilde{\boldsymbol{x}}}_{i+1/2} &= 0 \\ \boldsymbol{\Phi}(\boldsymbol{x}_0, t_0, \boldsymbol{p}) &= 0 \\ \boldsymbol{\Xi}(\boldsymbol{x}_f, t_f, \boldsymbol{p}) &= 0 \end{aligned} \tag{3.39}$$

where $\boldsymbol{\Phi}$ and $\boldsymbol{\Xi}$ are the usual boundary conditions from optimal control. The additional parameters, $\boldsymbol{p}$, are constants-of-motion and other non-dynamical quantities that may be adjusted by the solver. Since this algorithm solves the indirect problem, there is no explicit cost function to minimize because optimality is guaranteed once all boundary conditions and dynamic equations-of-motion are satisfied. Sequential quadratic programming (SQP) is used to drive the optimization process, where the constraints are adjoined to the cost with Lagrange multipliers. In the `Python` package `SciPy`, the `minimize` method is used from the `optimize` module with the `SLSQP`

option. This algorithm serves as the basis for the numerical methods used in solving indirect problems and will be modified in Section 5.1.3.

3.5 Shooting

Another set of common algorithms for solving boundary-value problems are shooting methods. Instead of using meshes like in collocation and pseudospectral methods, shooting methods capture sensitivity information in the State-Transition Matrix (STM), Δ. The STM has initial and terminal states Δ_0 and Δ_f. Given a system with n equations-of-motion, $\boldsymbol{f}$, and m parameters, $\boldsymbol{p}$, we construct the STM by first defining the sensitivity matrix of the equations-of-motion

$$A(t) = \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} & \frac{\partial f_1}{\partial p_1} & \frac{\partial f_1}{\partial p_2} & \cdots & \frac{\partial f_1}{\partial p_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} & \frac{\partial f_2}{\partial p_1} & \frac{\partial f_2}{\partial p_2} & \cdots & \frac{\partial f_2}{\partial p_m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} & \frac{\partial f_n}{\partial p_1} & \frac{\partial f_n}{\partial p_2} & \cdots & \frac{\partial f_n}{\partial p_m} \end{bmatrix} \tag{3.40}$$

Then, the state-transition matrix is defined as the following set of first-order differential equations

$$\Delta_0 = \begin{bmatrix} Id_M & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{3.41}$$

$$\dot{\Delta} = A(t)\Delta$$

Note that the state-transition matrix has dimension $n \times (n+m)$ and is not square when there are parameters. Since parameters are constants, they do not have equations-of-motion and their sensitivities with respect to changes in states and all parameters

are assumed zero. Using the state-transition matrix, sensitivities in the boundary conditions are given by the Jacobian matrix

$$\begin{aligned} M &= \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f, \boldsymbol{p})}{\partial \boldsymbol{x}_0} \\ P &= \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f, \boldsymbol{p})}{\partial \boldsymbol{p}} \\ \mathscr{J} &= [M, P] \end{aligned} \tag{3.42}$$

where $\mathscr{J}$ is the concatenation of M and P. Also, note that $\boldsymbol{x}_f$ is directly affected by perturbations in $\boldsymbol{x}_0$ and must be accounted for in M. Since the error in the boundary conditions are given as

$$\Phi(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f, \boldsymbol{p}) = \epsilon \tag{3.43}$$

Then the shooting algorithm is driven by Newton's Method with the update rule

$$\mathscr{J}\, \boldsymbol{x}_{update} = -\epsilon \tag{3.44}$$

3.6 Summary

In this chapter, we showed how the traditional indirect methods may be recasted in the language of differential geometry. Specifically, we showed that the symplectic equation correctly encoded the same information from traditional indirect methods. Next, we provided two numerical processes for solving the resulting BVPs. Although the procedure presented in this chapter yields a Hamiltonian BVP with identical properties to the one from Section 1.2.2, the symplectic equation will be used further in Chapters 4 and 5 and the numerical methods will be improved upon in Chapter 6. Specifically, in Chapter 5 we eliminate more equations-of-motion than previously possible and then design numerical solvers in Chapter 6 to solve the resulting problems.

Even though our system is now phrased in terms of symplectic mechanics, there is one major practical limitation to all indirect methods. That is the handling of control variables. In the next chapter, we will improve on a current state-of-the-art method for handling control variables by making it symplectic.

4. ADJOINING OF OPTIMAL INFORMATION

All indirect methods generate a Hamiltonian BVP. One of the primary challenges in creating a general-purpose process based on indirect methods is the large amount of algebraic manipulation involved in generating this Hamiltonian BVP. Specifically, Pontryagin's Minimum Principle (PMP) necessarily states that optimal motion is a minimizer of the control Hamiltonian for all time. This arises as the following mathematical condition [68, 69]

$$H(t, \boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{u}^*) \leq H(t, \boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{u}) \tag{4.1}$$

Directly solving Eq. (4.1) is challenging. For general purpose applications, it is possible to use a numerical optimization process such as Nelder-Mead simplex or sequential quadratic programming to find a $\boldsymbol{u}^*$ that satisfies Eq. (4.1). While using a numerical process that can handle discontinuities is overall very generalized, it is both numerically sensitive and computationally expensive. For large problems, the computational cost can become prohibitive. Instead, most modern indirect processes skip a direct solution of Eq. (4.1) and opt to solve the following condition in its place

$$\frac{\mathbb{d}H}{\mathbb{d}\boldsymbol{u}} = 0 \tag{4.2}$$

Rather, $\boldsymbol{u}^* = \boldsymbol{u}^*(t, \boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ is chosen such that Eq. (4.2) is satisfied without directly solving Eq. (4.1). The primary benefit of minimizing the Hamiltonian in this manner is that an analytic function for $\boldsymbol{u}^*$ is used as opposed to a numerical process. In general, analytic functions have a much faster evaluation time and also do not have the same numerical sensitivity issues that an optimization-based process would have. Finding $\boldsymbol{u}^*$ in this manner effectively eliminates $\boldsymbol{u}$ from the Hamiltonian BVP.

It has been shown numerous times that using symbolic manipulation software to solve Eq. (4.2) is a promising approach for implementing indirect methods for

general-purpose usage [70–74]. On the other hand, to the author's knowledge, there exist only two modern optimal control solvers based on indirect methods that can be considered for general-purpose applications [73, 140].

The form of Eq. (4.2) can be incredibly challenging to solve even considering the power of modern CASs. As the complexity and number of dimensions grows large, there is a disproportionate increase in the difficulty of solving Eq. (4.2). For instance, consider the thrust-limited Goddard rocket

$$\begin{aligned} \min_T J &= \int_0^{t_f} \epsilon_{\text{trig}} \left(\sec\left(\frac{\pi}{2} \frac{2T - T_{\max} - T_{\min}}{T_{\max} - T_{\min}} \right) - 1 \right) \mathrm{d}t - h(t_f) \\ \dot{h} &= v \\ \dot{v} &= \frac{T}{m} - \frac{D}{m} - g \\ \dot{m} &= -\frac{T}{c} \end{aligned} \tag{4.3}$$

with drag and gravity functions

$$\begin{aligned} D &= d_c v^2 \exp\left[-h_c \frac{h - h_0}{h_0} \right] \\ g &= g_0 \left(\frac{h_0}{h} \right)^2 \end{aligned} \tag{4.4}$$

Note that the control path-constraint has been trigonomerized [84]. Using Eq. (4.2) and `SymPy` [76] to generate a control law, it can be shown there are four control branches. Although the differences between each of the four branches is minor, each branch must be evaluated at every point along a trajectory. Then, Eq. (4.1) must still be used to find which control branch is optimal. In the Goddard rocket and similar "academic" problems, this is usually not an issue. As the complexity of problems grow, at best so does the complexity of the equations in each control branch as well as the number of control branches. For analysis, this effectively generates a new set of differential equations for every control branch. In the case of the Goddard rocket, because there are four control branches, there are four dynamical systems that must be solved simultaneously. At worst, Eq. (4.2) generates analytically unsolvable transcendental functions. Both scenarios are prohibitive in the usage of indirect methods for general-purpose applications.

Antony, et. al. tackled the issue of multiple control branches in their Integrated Control Regularization Method (ICRM) [74, 141]. The primary philosophy of Thomas's ICRM is simple: modern digital computers are better-equipped to solve ordinary-differential equations (ODEs) as opposed to algebraic equations. Therefore, the algebraic equations are recast as ODEs. This process has seen success in challenging path-constrained problems with various improvements by others [84,142–144]. Ideally, ICRM should operate as a functor on Hamiltonian BVPs

$$\mathscr{F}_{\text{ICRM}} : \text{Problem } \Lambda \to \text{Problem } \Lambda \tag{4.5}$$

That is, in a general-purpose process, the ICRM strategy should generate a new Hamiltonian BVP thereby enabling subsequent usage of other functors on Hamiltonian BVPs. In its current iteration, the ICRM strategy does produce a well-formed Two-Point Boundary-Value Problem (TPBVP); however, it is not Hamiltonian. Again considering the Goddard rocket, the BVP produced by ICRM is 7-dimensional whereas a Hamiltonian BVP is necessarily even-dimensional. Since mathematical structure is lost, any processes that follow the application of ICRM are not guaranteed to work as expected. This is the primary issue in constructing a general-purpose process where several different strategies may be used in series with one another.

The contribution of this section is to improve upon the ICRM method to preserve the Hamiltonian structure of BVPs. This will be done by encoding the optimal information directly into the differential structure of the dynamical system. In Chapters (2.5) and (4.3) of Ref. [137], the author describes how the differential structure of a system may be altered to encode information into the solution using an example of magnetism. Similarly, in Chapter (15) of Ref. [145], the author generalizes the notion of a Hamiltonian system composed of 3 objects (a manifold, a differential structure, and a Hamiltonian function) to that of a geometric mechanical system composed of two objects (an extended manifold and an extended differential structure). This eliminates the need for a Hamiltonian function to have a well-defined mechanical system.

These methods are adapted here for optimal control systems by introducing the control variables to the system as additional states with associated co-states. In doing so, this method preserves the geometric structure of the system while simultaneously eliminating the need to generate an augmented Hamiltonian to incorporate the newly formed system of equations. By preserving the geometric structure of the system, this enables the usage of powerful analytic tools that require preservation of mathematical structure including geometric integrators [146–148] and various reduction techniques [118, 149–151].

4.1 Integrated Control Regularization Method

4.1.1 Differential Optimal Control Law

In this section, we will introduce trajectory optimization by indirect methods in its traditional form [10]. From this process, the resulting system is a mixed set of Differential-Algebraic Equations (DAEs). Then, we will recast the algebraic equations as ODEs thereby translating the DAEs to a set of ODEs. The set of fixed initial time optimal control problems with unbounded controls are considered where the performance index to be minimized is of the Bolza form [9]

$$\begin{aligned} \min J &= \eta(t_f, \boldsymbol{x}(t_f), \boldsymbol{p}) + \int_0^{t_f} L(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}) \, \mathbb{d}t \\ \text{Subject to: } \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}) \\ \boldsymbol{\Phi} &= \boldsymbol{x}(t_0) - \boldsymbol{x}_0 \\ \boldsymbol{\Psi} &= \boldsymbol{x}(t_f) - \boldsymbol{x}_f \end{aligned} \tag{4.6}$$

We then adjoin time into the original OCP as a state

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix} \qquad \boldsymbol{f}' = \begin{bmatrix} t_f \boldsymbol{f} \\ t_f \end{bmatrix} \qquad \boldsymbol{p}' = \begin{bmatrix} \boldsymbol{p} \\ t_f \end{bmatrix} \tag{4.7}$$

This results in a new OCP with fixed endpoints on the independent variable $\tau \equiv t/t_f$

$$\begin{aligned} \min J &= \eta'(\boldsymbol{x}'(1), \boldsymbol{p}') + \int_0^1 L(\boldsymbol{x}'(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}')\, \mathrm{d}\tau \\ \text{Subject to: } \dot{\boldsymbol{x}} &= \boldsymbol{f}'(\boldsymbol{x}'(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}') \\ \boldsymbol{\Phi}' &= \boldsymbol{x}'(0) - \boldsymbol{x}_0' \\ \boldsymbol{\Psi}' &= \boldsymbol{x}'(1) - \boldsymbol{x}_1' \end{aligned} \tag{4.8}$$

The control Hamiltonian, $H = H(\boldsymbol{x}', \boldsymbol{\lambda}, \boldsymbol{u})$, is defined as

$$H \equiv L + \boldsymbol{\lambda}^T \boldsymbol{f}' \tag{4.9}$$

Because the control Hamiltonian is autonomous, the first-order necessary conditions for optimality are given as a two-point boundary valued problem with equations-of-motion

$$\dot{\boldsymbol{x}}' = \frac{\partial H}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}'(\boldsymbol{x}'(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}') \tag{4.10}$$

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}'} \tag{4.11}$$

$$0 = \frac{\partial H}{\partial \boldsymbol{u}} \tag{4.12}$$

Introducing $\boldsymbol{\xi}$ and $\boldsymbol{\nu}$ as Lagrange multipliers for the initial and terminal boundary constraint functions respectively, the boundary conditions for the two-point boundary valued problem are found as

$$\boldsymbol{x}'(a) = \boldsymbol{x}_a' \tag{4.13}$$

$$\boldsymbol{\lambda}(a) = \frac{\partial G}{\partial \boldsymbol{x}'(a)} \tag{4.14}$$

$$H(\tau_f) + \frac{\partial G}{\partial \tau_f} = 0 \tag{4.15}$$

$$G \equiv \eta'(\boldsymbol{x}'(1), \boldsymbol{p}') + \boldsymbol{\nu}^T \boldsymbol{\Psi}'(\boldsymbol{x}'(1)) - \boldsymbol{\xi}^T \boldsymbol{\Phi}'(\boldsymbol{x}'(0)) \tag{4.16}$$

where $a \in \{0, 1\}$. Note that there are three equations associated with the dynamic first-order necessary conditions, Eqs. (4.10 - 4.12), whereas the standard form of Hamilton's equations has two, only Eqs. (4.10 and 4.11). Simple problems return relatively simple expressions for $\partial H/\partial \boldsymbol{u}$, but complicated problems return increasingly

complex algebraic expressions. Often, there are several solutions to Eq. (4.12) resulting in multiple valid sets of ODEs. Typically, this can be considered a non-issue since PMP is used to select the optimal control branch. However, for general-purpose applications the amount of numerical subroutines needs to be kept to a minimum while simultaneously retaining the structure of the original problem formulation. To do this, we make the assumption that $\boldsymbol{u}^*$ is smooth, and the control law is reformulated as an ODE.

$$u_i^* = \int_0^t \dot{u}_i^* \, \mathrm{d}t + u_i^*(0) \tag{4.17}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial H}{\partial u_i^*} = 0 = \frac{\partial^2 H}{\partial \boldsymbol{x}' \partial u_i^*}^T \dot{\boldsymbol{x}}' + \frac{\partial^2 H}{\partial \boldsymbol{\lambda} \partial u_i^*}^T \dot{\boldsymbol{\lambda}} + \frac{\partial^2 H}{\partial \boldsymbol{u}^* \partial u_i^*}^T \dot{\boldsymbol{u}}^* \tag{4.18}$$

Rearranging these terms, we find

$$-\frac{\partial^2 H}{\partial \boldsymbol{x}' \partial u_i^*}^T \dot{\boldsymbol{x}}' - \frac{\partial^2 H}{\partial \boldsymbol{\lambda} \partial u_i^*}^T \dot{\boldsymbol{\lambda}} = \frac{\partial^2 H}{\partial \boldsymbol{u}^* \partial u_i^*}^T \dot{\boldsymbol{u}}^* \tag{4.19}$$

Where $i = 1, \cdots, m$ is the number of control input variables. In Eqs. (4.17) and (4.19), there are two new unknowns for every control variable. The first unknown, $\dot{u}_i^*$, is found by simultaneously solving Eq. (4.19) for all i. Since the coefficients of $\dot{\boldsymbol{u}}^*$ are known, finding $\dot{\boldsymbol{u}}^*$ is accomplished by solving a system of the form $A\dot{\boldsymbol{u}}^* = B$, that is

$$\begin{bmatrix} \frac{\partial^2 H}{\partial u_1^* \partial u_1^*} & \frac{\partial^2 H}{\partial u_1^* \partial u_2^*} & \cdots & \frac{\partial^2 H}{\partial u_1^* \partial u_m^*} \\ \frac{\partial^2 H}{\partial u_2^* \partial u_1^*} & \frac{\partial^2 H}{\partial u_2^* \partial u_2^*} & \cdots & \frac{\partial^2 H}{\partial u_2^* \partial u_m^*} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 H}{\partial u_m^* \partial u_1^*} & \frac{\partial^2 H}{\partial u_m^* \partial u_2^*} & \cdots & \frac{\partial^2 H}{\partial u_m^* \partial u_m^*} \end{bmatrix} \begin{bmatrix} \dot{u}_1^* \\ \dot{u}_2^* \\ \vdots \\ \dot{u}_m^* \end{bmatrix} = \begin{bmatrix} -\frac{\partial^2 H}{\partial \boldsymbol{x}' \partial u_1^*}^T \dot{\boldsymbol{x}}' - \frac{\partial^2 H}{\partial \boldsymbol{\lambda} \partial u_1^*}^T \dot{\boldsymbol{\lambda}} \\ -\frac{\partial^2 H}{\partial \boldsymbol{x}' \partial u_2^*}^T \dot{\boldsymbol{x}}' - \frac{\partial^2 H}{\partial \boldsymbol{\lambda} \partial u_2^*}^T \dot{\boldsymbol{\lambda}} \\ \vdots \\ -\frac{\partial^2 H}{\partial \boldsymbol{x}' \partial u_m^*}^T \dot{\boldsymbol{x}}' - \frac{\partial^2 H}{\partial \boldsymbol{\lambda} \partial u_m^*}^T \dot{\boldsymbol{\lambda}} \end{bmatrix} \tag{4.20}$$

As long as matrix A is invertible, the solution for $\dot{\boldsymbol{u}}^*$ is given by $A^{-1}B$. For the other unknown, $\boldsymbol{u}^*(0)$, there is no known general analytical formula; however, it only needs to be determined at one location on a trajectory. Therefore, the root-solving for $\boldsymbol{u}^*(0)$ is assumed to be handled by the same algorithm that solves for the boundary conditions of $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ by implementing the following boundary condition

$$0 = \left.\frac{\partial H}{\partial \boldsymbol{u}}\right|_{\tau=0} \tag{4.21}$$

In this formulation, $\boldsymbol{u}$ adds m additional states to the system with their own associated rates. The system is now described by $2n+m$ states each with equations-of-motion. Although this system will accurately model optimal motion, it is not Hamiltonian since m may take on odd integer values. We added the control variables to the system as states, and in the next section, we will show how this process may be modified to add a co-state variable for every control in a manner that preserves the Hamiltonian structure.

4.1.2 Symplectification of the Differential Optimal Control Law

In this section, we will introduce trajectory optimization by indirect methods in its differential-geometric form. From this process, the resulting system is a mixed set of DAEs. Then we recast the algebraic equations as ODEs, thereby translating the differential-algebraic system to a set of ODEs. These ODEs will then be finally adjoined back into the differential-geometric system. We begin by rewriting Eqs. (4.10) to (4.12) as follows

$$\iota_{X_H}\omega = \mathbb{d}H \tag{4.22}$$

Eq. (4.22) is called the symplectic equation and it serves as the geometric representation of Hamiltonian systems. For optimal control problems, X_H is of the form

$$X_H = \frac{\mathrm{d}\boldsymbol{x}'^T}{\mathrm{d}\tau}\frac{\partial}{\partial \boldsymbol{x}'} + \frac{\mathrm{d}\boldsymbol{\lambda}^T}{\mathrm{d}\tau}\frac{\partial}{\partial \boldsymbol{\lambda}} \tag{4.23}$$

From this, it is relatively straight forward to show that Eq. (4.22) recovers the dynamic first-order necessary conditions for optimality in coordinates

$$\iota_{X_H}\omega = \mathbb{d}H \tag{4.24}$$

Expanding this equation, we have

$$\left(\frac{\mathrm{d}\boldsymbol{x}'^T}{\mathrm{d}\tau}\frac{\partial}{\partial \boldsymbol{x}'} + \frac{\mathrm{d}\boldsymbol{\lambda}^T}{\mathrm{d}t}\frac{\partial}{\partial \boldsymbol{\lambda}}\right) \lrcorner\, (\mathbb{d}\boldsymbol{x}' \wedge \mathbb{d}\boldsymbol{\lambda}) = \frac{\partial H}{\partial \boldsymbol{x}'}^T \mathbb{d}\boldsymbol{x}' + \frac{\partial H}{\partial \boldsymbol{\lambda}}^T \mathbb{d}\boldsymbol{\lambda} + \frac{\partial H}{\partial \boldsymbol{u}}^T \mathbb{d}\boldsymbol{u} \tag{4.25}$$

This equation is rearranged into the following form

$$\left(\dot{\boldsymbol{x}}' - \frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T \mathbb{d}\boldsymbol{\lambda} - \left(\dot{\boldsymbol{\lambda}} + \frac{\partial H}{\partial \boldsymbol{x}'}\right)^T \mathbb{d}\boldsymbol{x}' + \left(\frac{\partial H}{\partial \boldsymbol{u}}\right)^T \mathbb{d}\boldsymbol{u} = 0 \tag{4.26}$$

In a hypothetical sense, root-solving Eq. (4.12) allows one to reformulate the phase space entirely in terms of $\boldsymbol{x}$ and $\boldsymbol{\lambda}$. As mentioned before this is not practical in every scenario. If $\partial H/\partial \boldsymbol{u}$ is substantially complex, explicit solutions of the form $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\lambda})$ may not exist. This is where ICRM is applied to the system. One major issue with using ICRM in its current formulation is that the natural symplectic structure is lost in the process. The symplectic structure is necessarily even-dimensional because each state must have an associated co-state. To introduce co-states for each newly formed state, one may consider modifying ICRM by augmenting the Hamiltonian using the well-known Lagrange multiplier method. This does not properly account for the new states and co-states. The following is an incorrect formulation

$$H' = H + \boldsymbol{\lambda}_u^T \dot{\boldsymbol{u}}^* \tag{4.27}$$

Evaluating $\iota_{X_H}\omega = \mathbb{d}H'$ yields incorrect results. Fundamentally, this changes the original functional optimization, and thus, the resulting necessary conditions for optimality are altered as well. Since the canonical symplectic structure is a 2-form, the optimal control laws must be 2-forms as well. Forming the control law using the differential-algebraic structure provides the following constraint 1-form

$$\begin{aligned} \frac{\mathbb{d}\boldsymbol{u}^*}{\mathbb{d}t} &= \dot{\boldsymbol{u}}^* \\ \mathbb{d}\boldsymbol{u}^* &= \dot{\boldsymbol{u}}^* \, \mathbb{d}t \\ \mathbb{d}\boldsymbol{u}^* - \dot{\boldsymbol{u}}^* \, \mathbb{d}t &= 0 \end{aligned} \tag{4.28}$$

where $\dot{\boldsymbol{u}}^*$ is a known solution to Eq. (4.20). To preserve the symplectic structure of the system, a corresponding co-state $\boldsymbol{\lambda}_u$ must be introduced. Currently, no meaningful interpretation of this co-state is expected, so the assumption that its value and time-rate of change are both 0 is made. Using the canonical definition of ω, the 2-form control law is written as follows

$$(\mathbb{d}\boldsymbol{u}^* - \dot{\boldsymbol{u}}^* \, \mathbb{d}t) \wedge (\mathbb{d}\boldsymbol{\lambda}_u - 0 \, \mathbb{d}t) = 0 \tag{4.29}$$

The original canonical symplectic structure of the system is augmented by adding the associated 2-form for each control, that is [137, 145]

$$\omega' = \omega + \sum_{i}^{m} (\mathbb{d}u_i^* - \dot{u}_i^* \, \mathbb{d}t) \wedge (\mathbb{d}\lambda_{u_i} - 0 \, \mathbb{d}t) \tag{4.30}$$

Represented in a matrix format, ω' has the following structure

$$\omega' = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & -\dot{\boldsymbol{u}}^* \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \dot{\boldsymbol{u}}^* & -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{4.31}$$

To account for the fact that $\boldsymbol{u}^*$ and $\boldsymbol{\lambda}_u$ are variables with equations of motion, the Hamiltonian vector field must be augmented as well

$$X'_H = X_H + \sum_{i}^{m} \frac{\mathbb{d}u_i^*}{\mathbb{d}\tau} \frac{\partial}{\partial u_i^*} + \frac{\mathbb{d}\lambda_{u_i}}{\mathbb{d}\tau} \frac{\partial}{\partial \lambda_{u_i}} \tag{4.32}$$

The full dynamical system may be now represented using the augmented Hamiltonian vector field, augmented symplectic form, and the original control Hamiltonian

$$\iota_{X'_H} \omega' = \mathbb{d}H \tag{4.33}$$

Evaluating Eq. (4.33) can be slightly more challenging than evaluating Eq. (4.22) due to the increased complexity of ω'. One key requirement of symplectic systems is anti-symmetry. This anti-symmetry can be easily seen in Eq. (4.31). By using a

Lower Upper (LU) decomposition, we exploit the anti-symmetric structure of ω' to solve Eq. (4.33) [152, 153]

$$\omega' = \left(\omega'_P\right)^{\mathrm{T}} \omega'_L \omega'_U =$$

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\dot{\boldsymbol{u}}^* & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & -\dot{\boldsymbol{u}}^* \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{4.34}$$

For problems of very high dimension, it is possible to also exploit the sparsity of ω' in the decomposition

$$\omega'_P \omega' \omega'_Q = \omega'_L \omega'_U \tag{4.35}$$

While exploiting sparsity is certainly an option, using the standard LU decomposition appears to rapidly solve all systems tested with the differential optimal control law. Solving the symplectic equation yields all the expected results for $\dot{\boldsymbol{x}}$, $\dot{\boldsymbol{\lambda}}$, and $\dot{\boldsymbol{u}}$, while $\dot{\boldsymbol{\lambda}}_u$ has the following interesting result

$$\frac{\mathrm{d}\lambda_{u_i}}{\mathrm{d}\tau} = \frac{\partial H}{\partial u_i} \tag{4.36}$$

At optimal conditions, Eq. (4.36) vanishes. Thus, the co-states for each control variable represent the corresponding stationary conditions from the original optimal control system.

4.2 Results

4.2.1 Goddard Rocket Problem

As mentioned prior, certain problems have dynamics that are so complex that Computer Algebra Systems are unable to solve for the optimal control law $\boldsymbol{u}^*$. The thrust-limited Goddard rocket problem is one such example of many problems where

this starts to become an issue. Root-solving $\mathrm{d}H/\mathrm{d}T$ results in four separate control branches. Instead, by using ICRM the control law is reformulated as a differential equation to utilize the information in the problem. The equation-of-motion for the control variable is

$$\frac{\mathrm{d}T}{\mathrm{d}\tau} = \frac{(A - B - \lambda_v T)\,(cm^2)^{-1}}{C + D} t_f \tag{4.37}$$

where:

$$\begin{aligned}
A &= \lambda_v \left(T - d_c v^2 \exp\left(-h_c \frac{h - h_0}{h_0} \right) \right) \\
B &= cm \left(2m^{-1} d_c \lambda_v v \exp\left(-h_c \frac{h - h_0}{h_0} \right) - \lambda_h \right) \\
C &= \left[(T_{max} - T_{min})^2 \cos^3(p) \right]^{-1} \left[2\pi^2 \epsilon_{\mathrm{trig}} \sin^2(p) \right] \\
D &= \left(\pi^2 \epsilon_{\mathrm{trig}} \right)^{-1} \left[(T_{max} - T_{min})^2 \cos(p) \right] \\
p &= \frac{\pi}{2} \frac{2T - T_{\max} - T_{\min}}{T_{\max} - T_{\min}}
\end{aligned} \tag{4.38}$$

Although Eqs. (4.37) and (4.38) together appear quite large, they were derived with ease using `SymPy` 1.4 [76]. Additionally, there is a single equation thereby reducing the total number of dynamical systems from four in PMP down to one in ICRM. In order for the system to be symplectic, it must have an even number of dimensions. Evaluating the symplectic equation to find the equation-of-motion for the co-state of thrust, we have

$$\frac{\mathrm{d}\lambda_T}{\mathrm{d}\tau} = -\left(\frac{\pi \epsilon_{\mathrm{trig}} \sin(p)}{(T_{max} - T_{min}) \cos^2(p)} + \frac{\lambda_v}{m} - \frac{\lambda_m}{c} \right) t_f \tag{4.39}$$

Note that Eq. (4.39) is equivalent to Eq. (4.36)

$$\frac{\mathrm{d}\lambda_T}{\mathrm{d}\tau} = \frac{\partial H}{\partial T} \tag{4.40}$$

We then solved this problem numerically using the numerical values shown in Table 4.1 for both the symplectic ICRM and another strategy using PMP. Some results can be seen in Fig. 4.1. The control histories in both PMP and ICRM are split into three phases: bang-singular-bang. These control histories are in very good overall agreement. Additionally, ICRM generates a co-state for the control variable which is also shown. We expected the co-state should be close to 0 at optimal conditions due

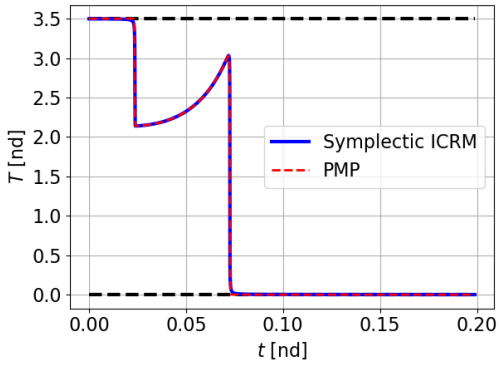
to Eq. (4.36) and we see there is agreement with the numerical results. The error is on the order of 1×10^{-6} which is in line with the tolerance of the numerical BVP solvers used.

Table 4.1. Parameters of the Goddard rocket.

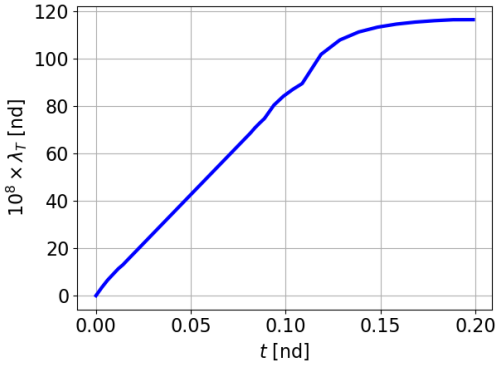
Symbol	Value [nd]
h_0	1.0
v_0	0.0
m_0	1.0
v_f	0
m_f	0.6
g_0	1.0
h_c	500
c	0.5
d_c	310.0
$T_{\min}$	0
$T_{\max}$	3.5

4.3 Summary

In this chapter, we further improved upon Thomas's ICRM strategy by applying it to the symplectic equation. In our application of ICRM to the symplectic equation, we shifted focus from directly modifying the state equations to encoding data in the symplectic form in a structure preserving manner. This ultimately enables usage of ICRM in problems that also will use dimension reduction strategies. We applied this strategy to the classic Goddard Rocket OCP and compared it to the traditional method of using PMP. The numerical results were in agreement thus validation this strategy.



(a) Comparison in the control time-histories of the symplectic ICRM and PMP strategies.



(b) Numerical values for the co-state of thrust are close to 0. With PMP, thrust does not have a co-state.

Figure 4.1. Results from the numerical solution of the BVP.

5. REDUCTION

In typical dynamical systems, it is often noted that each constant-of-motion may be used to eliminate a single equation-of-motion. While this is true in the Newtonian sense, for a dynamical system derived in the Hamiltonian formalism, every constant-of-motion may eliminate up to two equations-of-motion due to the underlying canonical structure [118, 119]. This reduction, known as symplectic reduction, guarantees that a symplectic manifold reduced by constants-of-motion and symmetries gives rise to another symplectic manifold of lower dimension.

Marsden-Weinstein-Meyer reduction, or symplectic reduction, has a very recent yet very rich history. Since its introduction, reduction has had very broad application and generalizations to Hamiltonian systems [154], Lagrangian systems [155, 156], contact manifolds [157], presymplectic manifolds [158], and even non-holonomic systems [159–163]. Application of reduction has also extended to optimal control systems [150, 151, 163–169]. Despite the fact that reduction has been successfully applied to many optimal control systems, there appear to be very few, if any, numerical results. Since aerospace missions are almost never analytically solvable, we will introduce reduction in this chapter in a manner than lends itself well to be solved numerically.

A common theme among every reduction strategy is that the structure of Hamilton's equations are intact. This means analytic and numerical tools applicable to symplectic or Hamiltonian systems, such as Poisson brackets, canonical transformations, symplectic integrators, and even further reductions, are still viable tools on the reduced manifold. Optimal control systems were previously rephrased as Hamiltonian systems on symplectic manifolds, and we therefore propose a new class of tools for indirect methods leveraging this reduced dimensional manifold. Our format in this chapter will directly enable the creation of a standardized set of numerical algorithms

which will be introduced later in Chapter 6. In addition to this, since reduction preserves structure, we will also see in Chapter 7 how the extra tools mentioned may be brought back into our process for usage.

This chapter is broken up into two halves: topological reduction and algebraic reduction. In the first half, we will explore reduction from a topological perspective. Since reduction is topological, this means that it only guarantees the existence of certain desirable spaces. We will not go through the strategies in their traditional form, but rather we will attempt to describe the impacts on the manifolds involved as they are relevant to an aerospace system designer. Despite this, an aerospace designer is likely to be left unsatisfied. These reduction strategies tease the existence of desirable spaces but do not provide an explicit recipe for generating the algorithms that can be applied to aerospace missions. The second half of this chapter is devoted to extending reduction at the algebraic level thereby enabling application to general aerospace missions. The reduction at the level of the algebra will then yield systems that will be solved numerically with special solvers in Chapter 6.

5.1 Topological Reduction

5.1.1 Reduction

The Marsden-Weinstein-Meyer reduction theorem formalizes the reduction process for symplectic manifolds. The full theorem and proof is available in Refs. [118, 119]; however, we will highlight some aspects of the theorem. The reduction process is a two-step procedure beginning with identifying symmetries, constants-of-motion, and their corresponding group structure. We first collect the constants-of-motion into a momentum map.

***Definition* 36** (Momentum Map)**.** Given a symplectic manifold, (Λ, ω), and a Lie group of symmetries, G, acting symplectically on this manifold. The momentum map is a map to the first de Rham co-homology, $J : M \to \mathfrak{g}^*$, such that

$$\mathbb{d} \langle J, \xi \rangle = \iota_{X_\xi} \omega \tag{5.1}$$

End definition.

After collecting the constants-of-motion into a momentum map, we evaluate it at the value $\mu \in \mathfrak{g}^*$. Next, we restrict motion to the set of all reachable states $J^{-1}(\mu)$. This ensures optimal motion on (M, ω) is also on level sets of J. Because motion is dependent on values of μ, this implies the value of μ must be known a priori. This is not an issue for optimal control problems because an initial guess is required by the majority of algorithms, so a sub-optimal value of μ is known.

Despite its unusual appearance and apparent complexity, this first reduction is fairly common. One such example is the example of a time-optimal launch of a Titan II in Ref. [9]. The authors note that since $\partial H / \partial x_1 = \dot{\lambda}_1 = 0$, then λ_1 is no longer needed as a state in the TPBVP effectively reducing the problem from 8 to 7 equations-of-motion. This is due to the fact that $\partial / \partial x_1$, or the vector corresponding to the downrange distance of the vehicle, is an infinitesimal symmetry such that

$$\begin{aligned} \int \frac{\partial}{\partial x_1} \lrcorner \, \omega &= \int \mathbb{d} \lambda_1 \\ &= \lambda_1 \end{aligned} \tag{5.2}$$

In this case, the equations-of-motion are odd dimensional and no longer Hamiltonian. The second step in the symplectic reduction procedure is to rewrite the problem on the quotient space of $J^{-1}(r)$ by an isotropy sub-group, G_μ. In the case of a group of symmetries forming a solvable Lie algebra, $G = G_\mu$, the reduced symplectic manifold is $M_\mu = J^{-1}(\mu)/G$ with dimension $\dim M - 2 \dim G$. The reduced symplectic form is then defined as $i_\mu^* \omega = \pi_\mu^* \omega_\mu$ with $i_\mu : J^{-1}(\mu) \to M$ and $\pi_\mu : J^{-1}(\mu) \to M_\mu$. The inclusion map, i_μ, is evaluated relatively easily knowing values of μ. However, the canonical projection, π_μ, is a principal G-bundle and it is typically not a trivial task to evaluate.

***Definition* 37.** A principal G-bundle, π_μ, is a fiber bundle whose fibers carry the structure of a Lie group.

$$\pi_\mu^{-1}(\nu) \cong G_\mu \tag{5.3}$$

End definition.

The symplectic reduction procedure is now complete, where the reduced symplectic manifold is $(J^{-1}(\mu)/G_\mu, \omega_\mu)$. Referring back to the example of a time-optimal launch of the Titan II, this step in the procedure implies x_1 is no longer needed as a state since $\{\partial/\partial x_1\} = G$ and $[\partial/\partial x_1, \partial/\partial x_1]_{\mathrm{L}} = 0$.

On the other hand, if a group of symmetries forms a non-solvable Lie algebra, $G \neq G_\mu$ and $\dim G_\mu < \dim G$. There are two options to handle this: one can either identify the isotropy group, G_μ, that leaves μ invariant, or one could identify a subalgebra, $\mathfrak{p} = \{p \in \mathfrak{g} | \mathrm{Ad}_p^* \mu = 0\}$ with Lie group P and define $M_\mu = J^{-1}(\mu)/P$. The former follows straight from the Marsden-Weinstein-Meyer theorem, whereas the latter is from the Mishchenko-Fomenko theorem [170]. In the algebraic portion of this chapter, we will use the smallest possible subalgebra; therefore, the Mishchenko-Fomenko theorem is most applicable.

Unfortunately, this reduction procedure only guarantees the existence of such a reduced manifold. It does not provide an explicit recipe for creating algorithms on this space. To overcome this, and other difficulties described above, we will attempt to separate the equations of motion in the next section using the fact that the reduced manifold is the base space of a principal G-bundle as a guide. Overall, the reduction process is described succinctly in the commutative diagram in Fig. 5.1.

$$\begin{array}{ccc} \Lambda & \xrightarrow{\quad H \quad} & \mathbb{R} \\ {\scriptstyle i_\mu}\uparrow & & \uparrow{\scriptstyle H_\mu} \\ J^{-1}(\mu) & \xrightarrow{\ \pi_\mu\ } & J^{-1}(\mu)/G_\mu \end{array}$$

Figure 5.1. Marsden-Weinstein-Meyer Reduction.

5.1.2 Separation

To create algorithms and perform design on $J^{-1}(\mu)/G_\mu$, the equations-of-motion must be described by a chart that decouples dynamics on the fibers from those on the base space of π_μ. Consider the Carathéodory-Jacobi-Lie theorem as a thought experiment. This theorem, at a high level, states that given involutive smooth functions, $\{f_1, \cdots, f_m\}$, additional functions, $\{f_{m+1}, \cdots, f_n, g_1, \cdots, g_n\}$, exist such that a local chart may be chosen

$$\omega = \sum_i \mathbb{d} f_i \wedge \mathbb{d} g_i \tag{5.4}$$

Given a Hamiltonian system, since $\mathbb{d}H \neq 0$, a coordinate chart can be chosen where the Hamiltonian is a coordinate. Since $\mathbb{d}_t H = 0$, then $H = \int \mathbb{d}H = \int \partial_{\lambda_H} \lrcorner\, \omega$ so that ∂_{λ_H} is a symmetry. This new chart decouples λ_H from the dynamics on the base space of π_μ. Applying this same procedure to arbitrary constants of motion, c_i, we see that since $\mathbb{d}c_i$ are rank 1 co-vector fields perpendicular to optimal motion, their associated symmetries are "straightened out" such that $\mathbb{d}\lambda_{c_i}$ is tangential to optimal motion.

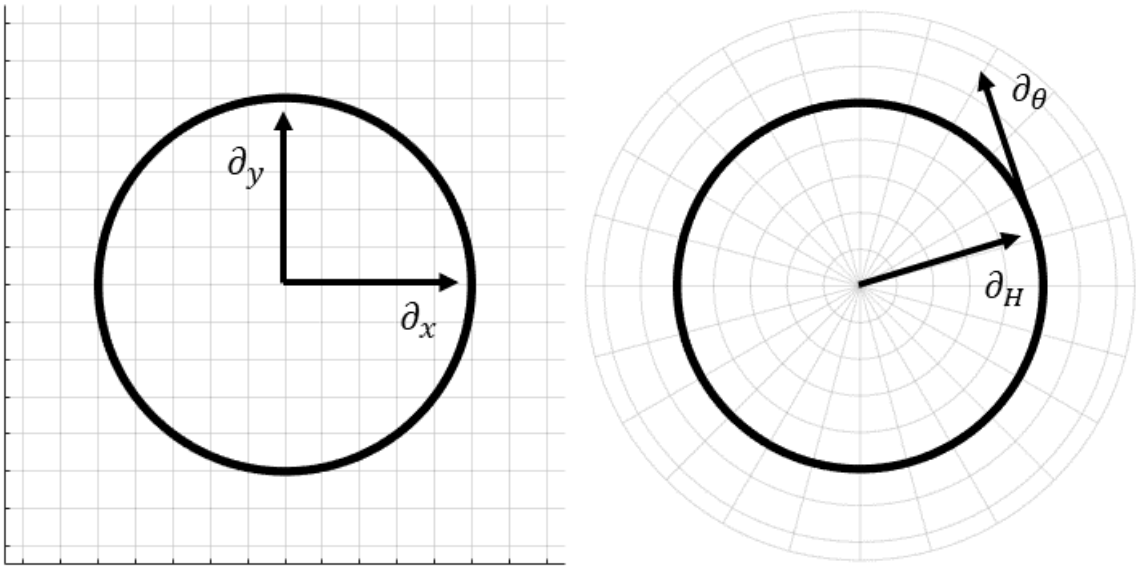
For example, consider a circle with Hamiltonian $H = x^2 + y^2$. Clearly, H represents the square of the radius of the circle and is invariant under the group of rotations. However, in this chart, equations-of-motion become

$$\dot{x} = 2y, \quad \dot{y} = -2x \tag{5.5}$$

By choosing a new chart where H is a coordinate and θ is its adjoint state, the straightened out equations-of-motion become

$$\dot{H} = 0, \quad \dot{\theta} = -1 \tag{5.6}$$

Notice how both H and θ no longer appear in the equations-of-motion. We began with two ODEs and then used a single constant-of-motion to arrive at Eq. (5.6). These new ODEs are solvable.



(a) Chart with $\{x, y\}$

(b) Chart with $\{H, \theta\}$

Figure 5.2. Straightening out a circle

5.1.3 Reconstruction

Clearly, eliminating up to two equations-of-motion for every constant-of-motion will have a significant beneficial impact on computational performance. However, in practical design scenarios, one is interested in guiding a vehicle from an initial state to a terminal state. This could be a hypersonic vehicle with fixed initial conditions flying to a terminal downrange or crossrange, a satellite performing an orbit raising manuever finishing at a fixed true-anomaly, or a robotic arm performing a required task. For hypersonic reentry, the spherical symmetry of the earth allows one to eliminate downrange or crossrange from the equations-of-motion. A spacecraft under the gravitational influence of a central body in a spherically symmetric gravity field no longer uses true-anomaly as a coordinate on the reduced space. A robotic arm is subject to a set of holonomic constraints from its joints that may be explicitly eliminated from the equations-of-motion. In each of these three scenarios, there is valuable information to a designer that has been removed from the equations-of-motion on the reduced manifold. Simply put, these coordinates no longer exist. How do we leverage the computational and stability benefits of numerical algorithms on $J^{-1}(\mu)/G_\mu$ when fundamentally, the design problem is on M? This section outlines the process of reconstructing the full dynamics on M from information on $J^{-1}(\mu)/G_\mu$.

Determining motion along $J^{-1}(\mu)$ is relatively straightforward. In fact, it is already known that μ must be provided a priori for a numerical algorithm to operate on the reduced dimensional space. However, motion along $\pi_\mu^{-1}(\nu)$ does not affect motion on $J^{-1}(\mu)/G_\mu$. Since the system on $\pi_\mu^{-1}(\nu)$ has been decoupled from the base space dynamics, we note that coordinates chosen on the fibers are by definition symmetries of the system. Therefore only an initial or terminal state is required to be known, one of which is provided to a numerical algorithm in the initial guess structure. The full set of dynamics then may be completely reconstructed by quadrature. This allows the equations-of-motion to be evaluated on the reduced dimensional space while simultaneously ensuring the boundary conditions of the original problem are satisfied.

In other words, the subsystem on the base space is the set of nonlinear equations-of-motion that affect the dynamics of the original system. They must be propagated or otherwise enforced numerically and are referred to as the states of the system. The subsystem on the vertical space, $\pi_\mu^{-1}(\nu)$, is not a part of the dynamics and therefore does not need to be propagated or otherwise enforced numerically. We refer to the vertical subsystem as the quads of the system. To solve for the quads of the system, trapezoidal, Simpson's, and other vectorized integration by quadrature techniques work well. To determine the quads of the system, the states must be known. Since the states are a function of time, integration of the quads are with respect to time as well.

We gave a high level description of MWM reduction, focusing primarily on its topological implications. In the next section, we will cover MWM reduction a second time; however, we will focus more on the algebraic implementations of reduction.

5.2 Algebraic Reduction

Given a set of symmetries or constants-of-motion, the dynamical dimension of Problem Λ may be reduced by up to 2 for every symmetry and constant-of-motion pair. This is the primary result from the Marsden-Weinstein-Meyer (MWM) reduction theorem [171, 172]. The MWM theorem is a topological result guaranteeing the existence of a reduced dimensional Λ, but does not provide an explicit recipe for performing the reduction. This section aims to perform the reduction under the special condition that the constants-of-motion and their associated symmetries are in involution with one another. A constant-of-motion is any function that is constant along extremal curves of the OCP

$$\mathfrak{g}^* \ni \boldsymbol{g}(t, \boldsymbol{x}^*, \boldsymbol{u}^*) = 0 \; \forall \; t \in [0, t_f] \tag{5.7}$$

where $\boldsymbol{x}^*$ and $\boldsymbol{u}^*$ are minimizers of Problem Σ. In Problem Σ, since $\boldsymbol{u}^*$ is not known a priori, it is difficult to make use of $\boldsymbol{g}$. On the other hand, after dualization we have

$$\mathscr{D}(\boldsymbol{g}(t, \boldsymbol{x}^*, \boldsymbol{u}^*)) = \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{\lambda}) = 0 \tag{5.8}$$

Since explicit $\boldsymbol{u}$ terms are eliminated in Problem Λ by dualization, it is significantly easier to search for $\boldsymbol{g} \in \Lambda$ instead of $\boldsymbol{g} \in \Sigma$. This is because all trajectories in Λ satisfy local dynamic necessary conditions. In the reduction of Λ, two pieces of information are required: the constant-of-motion and its associated symmetry. With information about the constant-of-motion, its infinitesimal symmetry field can be generated relatively easily as follows. Let $[\cdot,\cdot]_{\mathrm{P}} \equiv \omega^\sharp$ be the Poisson bracket induced by ω [139]

$$\begin{aligned} \omega^\sharp = [\cdot,\cdot]_{\mathrm{P}} &= (\mathbb{d}\boldsymbol{x} \wedge \mathbb{d}\boldsymbol{\lambda})^\sharp \\ &= \partial_{\boldsymbol{x}} \wedge \partial_{\boldsymbol{\lambda}} \end{aligned} \tag{5.9}$$

Then, given some $\boldsymbol{g}$, recall the infinitesimal symmetry fields can be generated using the Noether-like map from Chapter 3

$$X_{\boldsymbol{g}} = [\cdot, \boldsymbol{g}]_{\mathrm{P}} \tag{5.10}$$

The infinitesimal symmetry field is a Lie algebra element, $\mathfrak{g}$, and the full symmetry group can be generated through the exponential map [173]

$$\begin{aligned} \exp : \mathfrak{g} &\to G \\ (X_{\boldsymbol{g}}) &\mapsto (\exp(tX_{\boldsymbol{g}})) \end{aligned} \tag{5.11}$$

MWM reduction requires that one remove the entire symmetry group from Λ. We will not directly evaluate the exponential map, but rather use the Lie algebra elements to identify the states that may be removed. The MWM reduction procedure is a two-step process beginning with the elimination of constants-of-motion. First, we identify all independent $\boldsymbol{g}$. Independent $\boldsymbol{g}$ are said to be in involution with one another. Any dependent $\boldsymbol{g}$ need to be handled differently and are out of the scope of this dissertation. There is a brief word on those cases in Section 9.1. Independent $\boldsymbol{g}$ satisfy one the following equivalent conditions

$$\begin{aligned} 0 &= [g_i, g_j]_{\mathrm{P}} \\ &= \omega(X_{g_i}, X_{g_j}) \end{aligned} \tag{5.12}$$

That is, for all i and j the $\boldsymbol{g}$ commute under the Poisson bracket induced by ω. Note that for problems not explicitly dependent on time, $\mathbb{d}_t H = 0$ and therefore $H \in \boldsymbol{g}$. In

cases where $[g_i, g_j]_{\mathrm{P}} \neq 0$, the Mishchenko-Fomenko procedure [170] is a formal process for choosing the largest commutative subset satisfying

$$\mathfrak{h}^* = \{g \in \mathfrak{g}^* | [g_i, g_j]_{\mathrm{P}} = 0\} \subseteq \mathfrak{g}^* \tag{5.13}$$

On the other hand, if only a single constant-of-motion is used, then this step can be ignored and $\mathfrak{h}^* \cong \mathfrak{g}^*$. The commutative subset, $\mathfrak{h}^*$, forms a momentum map for the system Λ [174]

$$\begin{gathered} \boldsymbol{J} : \Lambda \to \mathfrak{h}^* \\ (\boldsymbol{x}, \boldsymbol{\lambda}, t) \mapsto (g_1, g_2, \cdots, g_n) \end{gathered} \tag{5.14}$$

With all the ingredients required for the first reduction, we perform it as follows. Invert the moment map at a fixed-point to identify the set of all reachable states at a given level of $\boldsymbol{g}$

$$\boldsymbol{J}^{-1}(\boldsymbol{x}_0, \boldsymbol{\lambda}_0, 0) : \mathfrak{h}^* \to \Lambda \tag{5.15}$$

The fixed point was chosen to be the initial conditions for Problem Λ due to the fact an initial guess is commonly provided, although any point in Λ is sufficient provided there are no singularities. Solving for Eq. (5.15) requires root-solving $\boldsymbol{g}$ and can therefore be a challenge; however, this will result in $\boldsymbol{g}$ being coordinates of the system. Because $\boldsymbol{g}$ are constants-of-motion, $\dot{\boldsymbol{g}} = 0$. This ultimately means that $\boldsymbol{g}$ are not dynamical states, but rather dynamical parameters. The dimension reduction in this stage is equal to $\dim(\boldsymbol{g})$, and the first reduced space is identified as $i : J^{-1}(\mu) \to \Lambda$ where μ is the fixed initial point $(t_0, \boldsymbol{x}(t_0), \boldsymbol{\lambda}(t_0))$. Problems that have eliminated constants-of-motion by moving these quantities to the dynamical parameters are solvable by modern-day BVP solvers and will generate solutions for Problem Σ. However, these problems are no longer of type Λ. This is because $\boldsymbol{g}$'s associated symmetries have not been eliminated. Similar to how ICRM introduced an arbitrary co-state variable, other operations on Λ must preserve both Noether's and Liouville's theorems [97]. MWM reduction accounts for this in the second step of its procedure. In the second

step, the MWM procedure requires the reduced problem to be the quotient of the inverted moment map with the Lie group of symmetries

$$\begin{aligned} \Lambda^* &= J^{-1}(\boldsymbol{x}_0, \boldsymbol{\lambda}_0, 0)/G \\ &= \Lambda//G \end{aligned} \tag{5.16}$$

This second reduction identifies the fully reduced manifold, $\tau : J^{-1}(\mu) \to J^{-1}(\mu)/G = \Lambda//G$. However, evaluating the quotient by the full group of symmetries can be a challenge. To perform the actual reduction, we use the infinitesimal symmetry fields, $X_{\boldsymbol{g}}$, instead of their exponentiation. This is possible with the Cartan-Lie (Lie III) theorem [175], which enables the indirect study of Lie groups by study of their Lie algebras. We use the Lie algebras $X_{\boldsymbol{g}}$ to identify removable states as follows

$$\dot{\boldsymbol{q}} = X_H^\flat([\cdot, \boldsymbol{g}]_{\mathrm{P}}) \tag{5.17}$$

where $\flat = \sharp^{-1}$ such that $\flat \circ \sharp = Id_\Lambda$. The above equation works by identifying the infinitesimal symmetry fields with the Noether-like map $N^*(\boldsymbol{g}) = [\cdot, \boldsymbol{g}]_{\mathrm{P}}$, then extracting the dynamics on the symmetry from the Hamiltonian vector field. The resulting function for $\dot{\boldsymbol{q}}$ is a 0-form, or scalar function. This equation bears a striking similarity to the Lie-Poisson bracket, a common tool in some Poisson reduction strategies [176]. Both Eq. 5.17 and the Lie-Poisson bracket explicitly leverage a Poisson structure as well as a Lie structure. A further discussion on the Lie-Poisson equation is in Section 9.5. Identified by Eq. 5.17, these removable states, or quads, are typically not analytically integrable since they are of the form $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}(t, \boldsymbol{x}, \boldsymbol{\lambda})$. In general, $\boldsymbol{x}(t)$ and $\boldsymbol{\lambda}(t)$ are have no known closed-form solution. This is a direct result of performing the reduction at the level of the algebra instead of the group. Because of this, the quads must still be integrated along with the remaining dynamics from reduction of Λ. The reduced dynamics are defined by

$$\begin{aligned} H_\mu \circ \tau &= H \circ i \\ \tau^* \omega_\mu &= i^* \omega \end{aligned} \tag{5.18}$$

To define the full coordinate transformation, recall Eq. (3.27) from Section 3.2. We use that transformation together with $\boldsymbol{g}$ to define

$$\boldsymbol{q} \equiv \int_{\Lambda} \frac{\boldsymbol{\lambda}}{\boldsymbol{g}} \, \mathbb{d}\boldsymbol{x} \tag{5.19}$$

We can then define the trajectory Υ_μ on the reduced manifold as well as all other quantities in terms of $\boldsymbol{g}$ and $\boldsymbol{q}$. The resulting problem is of the form

$$\text{Problem } \Lambda//G \begin{cases} \mathbb{d}H_\mu = \iota_{X_{H_\mu}} \omega_\mu \\ \quad \dot{\boldsymbol{q}} = X_H^\flat([\cdot, \boldsymbol{g}]_{\mathrm{P}}) \\ \quad 0 = \boldsymbol{\Phi} \circ i \circ \tau^{-1} \circ \Upsilon_\mu(t_0) \\ \quad 0 = \boldsymbol{\Xi} \circ i \circ \tau^{-1} \circ \Upsilon_\mu(t_f) \end{cases} \tag{5.20}$$

Note that the symplectic equation is on the reduced manifold $\Lambda//G$ whereas $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ are not. Instead, the $\boldsymbol{q}$ live in the fibers of τ.

The boundary condition functions, $\boldsymbol{\Phi}$ and $\boldsymbol{\Xi}$, have not been altered in any way. Instead, quantities from the reduced trajectory, Υ_μ, are mapped to the full-dimensional manifold for evaluation of the boundary conditions. Evaluating τ^{-1} uses results from $\dot{\boldsymbol{q}}$ and therefore must take place after solving for the reduced dynamics. Rather, the group action on τ's fibers is given by $\dot{\boldsymbol{q}}$. Alternatively, reduced boundary condition functions, $\boldsymbol{\Phi}_\mu$ and $\boldsymbol{\Xi}_\mu$, may be identified in a similar manner to how the reduced Hamiltonian H_μ was defined

$$\begin{aligned} \boldsymbol{\Phi}_\mu \circ \tau &= \boldsymbol{\Phi} \circ i \\ \boldsymbol{\Xi}_\mu \circ \tau &= \boldsymbol{\Xi} \circ i \end{aligned} \tag{5.21}$$

The reduced boundary conditions do not share the benefits of reduction so using reduced or unreduced formulations yields similar results. The overall purpose of the quads are to reconstruct information that has been eliminated from $\Lambda//G$ as by using data from $\Lambda//G$. This ensures that the reduced problem indirectly solves the original Problem Σ. Despite some unreduced information carrying over, we still call the new

formulation Problem $\Lambda//G$ since these quantities are non-dynamical. In coordinates, this becomes

$$\text{Problem } \Lambda//G \begin{cases} \dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}(t, \boldsymbol{x}, \boldsymbol{\lambda}) \\ \dot{\boldsymbol{\lambda}} = \dot{\boldsymbol{\lambda}}(t, \boldsymbol{x}, \boldsymbol{\lambda}) \\ \dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}(t, \boldsymbol{x}, \boldsymbol{\lambda}) \\ 0 = \boldsymbol{\Phi}_\mu \circ \Upsilon_\mu(t_0) \\ 0 = \boldsymbol{\Xi}_\mu \circ \Upsilon_\mu(t_f) \end{cases} \tag{5.22}$$

Since $\dot{\boldsymbol{q}}$ is not a function of $\boldsymbol{q}$, the quads are non-dynamical. This means that it can be solved independently from the dynamical states $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, and can also be solved rapidly in a parallel manner using numerical quadrature, hence the naming quads. Most modern day BVP solvers can solve problems of the form in Eq. 5.22, however they must stack $\dot{\boldsymbol{q}}$ together with $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{\lambda}}$ to be handled in a similar, non-parallel manner. This is a limitation of modern day BVP solvers. As modern computational architectures become more massively parallel, modern day BVP solvers can likely be modified to accommodate Problem $\Lambda//G$. Some such BVP solvers are given in Ref. [177] and Chapter 6. In the next section, reduction is carried out explicitly on several examples.

5.3 Examples

In this chapter, we explored the application of Marsden-Weinstein-Meyer reduction on optimal control systems. One significant difference between the resulting Hamiltonian BVP from reduction and the traditional Hamiltonian BVPs is the existence of "quads", or non-dynamical states. We use the term "quads" to imply how these states can be efficiently solved. That is by using numerical quadrature. In direct methods, nearly all orthogonal collocation methods use a high performance quadrature scheme to approximate the cost function [178]. This operation may be repeated for the quads, thereby potentially parallelizing the computation of the quads which was previously was computed in a serial manner [179–185]. It is important to note

that this parallelization is a true parallelization due to the elimination of causality from the system, as opposed to a parallelization "trick" as it is with multiple shooting methods. Currently, there appear to be a few direct solvers that can handle problems of this structure. On the other hand, there are no solvers on the indirect side that can solve these types of problems. Before we explore how BVP solvers can be built to solve these problems in Chapter 6, we will next show how this reduction can be carried out explicitly in a few example problems.

5.3.1 Currentless Zermelo's Problem

One remarkable feature of reduction is that given enough involutory constants-of-motion, a finite-dimensional Hamiltonian system may be reduced to an equivalent 0-dimensional system. A few comments on 0-dimensional systems are in Section 9.4. Let Σ be the optimal control problem defined by

$$\begin{aligned} \min_{\theta} K &= \int_0^{t_f} 1 \, \mathrm{d}t \\ \text{Subject to: } \dot{x} &= V\cos(\theta) \\ \dot{y} &= V\sin(\theta) \\ x(t_0) &= y(t_0) = 0 \\ x(t_f) &= x_f \\ y(t_f) &= y_f \end{aligned} \tag{5.23}$$

Dualization yields the following Hamiltonian system on Λ

$$\begin{aligned} H &= \lambda_x V\cos(\theta) + \lambda_y V\sin(\theta) + 1 \\ \dot{x} &= V\cos(\theta) \\ \dot{y} &= V\sin(\theta) \\ \dot{\lambda}_x &= 0 \\ \dot{\lambda}_y &= 0 \end{aligned} \tag{5.24}$$

Since the time rates-of-change of λ_x and λ_y are 0, their integrated forms become

$$\begin{aligned} \lambda_x &= g_1 \\ \lambda_y &= g_2 \end{aligned} \tag{5.25}$$

Where g_1 and g_2 are constants-of-motion for the system. The momentum map is then

$$\begin{aligned} J : \Lambda &\to \mathfrak{g}^* \\ (x, y, \lambda_x, \lambda_y) &\mapsto (g_1, g_2) \end{aligned} \tag{5.26}$$

The constants-of-motion g_1 and g_2 are in involution due to their commutativity under the Poisson bracket

$$\begin{aligned} [g_1, g_2]_{\mathrm{P}} &= \left(\partial_x \wedge \partial_{\lambda_x} + \partial_y \wedge \partial_{\lambda_y}\right)(g_1, g_2) \\ &= \big(\partial_x \partial_{\lambda_x} + \partial_y \partial_{\lambda_y} \\ &\quad -\partial_{\lambda_x} \partial_x - \partial_{\lambda_y} \partial_y\big)(\lambda_x, \lambda_y) \\ &= 0 + 0 + 0 + 0 \\ &= 0 \\ [g_1, g_2]_{\mathrm{P}} &= [g_2, g_1]_{\mathrm{P}} \end{aligned} \tag{5.27}$$

To perform the first reduction, $i : J^{-1}(\mu) \to \Lambda$, choose g_1 and g_2 to be new free-parameters in the dynamical system. Then, restrict motion of the original system by pulling back the dynamics along i

$$\begin{aligned} H_r &= g_1 V \cos(\theta) + g_2 V \sin(\theta) + 1 \\ \dot{x} &= V \cos(\theta) \\ \dot{y} &= V \sin(\theta) \end{aligned} \tag{5.28}$$

Note that λ_x and λ_y are eliminated in the process. The Poisson tensor is also reduced to the following

$$\begin{aligned} i^*(\partial_x \wedge \partial_{\lambda_x} + \partial_y \wedge \partial_{\lambda_y}) &= \partial_x \wedge \partial_{g_1} + \partial_y \wedge \partial_{g_2} \\ &= \partial_x \wedge 0 + \partial_y \wedge 0 \\ &= 0 \end{aligned} \tag{5.29}$$

The structure of the differential equations is no longer canonical and therefore motion on $J^{-1}(\mu)$ is not Hamiltonian. Instead, the differential equations in Eq. 5.28 will yield the correct motion to describe the original system and is Hamiltonian when reconstructed on Λ. To perform the second reduction, $\tau : J^{-1}(\mu) \to J^{-1}(\mu)/G = \Lambda//G$, identify the symmetries using the moment map and $[\cdot,\cdot]_{\mathrm{P}} \in \Lambda$

$$\begin{aligned}
[\cdot, g_1]_{\mathrm{P}} &= (\partial_x \wedge \partial_{\lambda_x} + \partial_y \wedge \partial_{\lambda_y})(\cdot, g_1) \\
&= \partial_x \partial_{\lambda_x}(\cdot, \lambda_x) - \partial_{\lambda_x}\partial_x(\cdot, \lambda_x) \\
&\quad + \partial_y \partial_{\lambda_y}(\cdot, \lambda_x) - \partial_{\lambda_y}\partial_y(\cdot, \lambda_x) \\
&= \partial_x \partial_{\lambda_x}(\cdot, \lambda_x) \\
&= \partial_x(\cdot)
\end{aligned} \tag{5.30}$$

Similarly, the symmetry generated by g_2 is ∂_y. Since $\partial_x = \hat{x}$ and $\partial_y = \hat{y}$, we identify that $\dot{x}$ and $\dot{y}$ may be eliminated from the dynamics of the system. To do this, define new quads as $q_1 = x$ and $q_2 = y$ with initial conditions $q_{10} = x_0$ and $q_{20} = y_0$. Then, restrict the dynamical motion of the original system by pulling back along τ

$$\begin{aligned}
H_\mu &= g_1 V \cos(\theta) + g_2 V \sin(\theta) + 1 \\
\dot{q}_1 &= V \cos(\theta) \\
\dot{q}_2 &= V \sin(\theta)
\end{aligned} \tag{5.31}$$

The Poisson tensor is already 0 so pulling back along τ also results in 0. At first glance, the systems in Eq. 5.28 and Eq. 5.31 are strikingly similar. Mathematically speaking, these two sets of equations are identical; however, the former set of dynamics are 2-dimensional and not Hamiltonian whereas the latter are 0-dimensional and Hamiltonian. This is because the quads of the system, q_1 and q_2, are not states in the dynamical sense because perturbations of the quads will not affect the time-history of the overall system. Instead, q_1 and q_2 are solved rapidly using well-known quadrature integration schemes. The results are mathematically equivalent dynamical systems; however, the 0-dimensional system has certain properties that require fewer computations to solve than the 2-dimensional system.

5.3.2 Direct Brachistochrone

Despite the fact that the primary focus of this work is indirect methods, the structure presented here is also well suited for direct methods. Since an indirect system is generated by an OCP through the dualization functor, there is a direct analog. In fact, reduction can take place on the direct side in an almost identical manner. However, without any additional structure, the dimension reduction limit is equal to the number of total symmetries and constants-of-motion combined. The Brachistochrone problem illustrates this well.

The Brachistochrone problem is a classic optimal control problem that is analytically solvable. Solutions to the Brachistochrone problem are paths that minimize the time a frictionless bead spends on a wire as it slide from two fixed points. Solutions to the Brachistochrone problem are well known to be cycloid curves. The initial point, terminal point, and a cycloid curve are shown in Fig. 5.3.

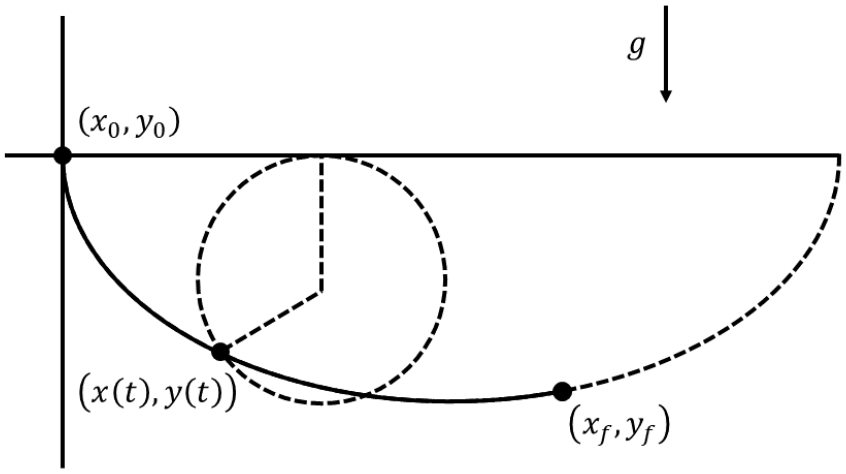


Figure 5.3. Cycloid curve as a solution to the Brachistochrone problem.

In this problem, g represents the gravitational force acting on the bead. This problem is posed as an OCP as

$$\begin{aligned}
\min_{\theta} K &= \int_0^{t_f} 1 \, \mathbb{d}t \\
\text{Subject to: } \dot{x} &= v\cos(\theta) \\
\dot{y} &= v\sin(\theta) \\
\dot{v} &= g\sin(\theta) \\
x(t_0) &= y(t_0) = v(t_0) = 0 \\
x(t_f) &= x_f \\
y(t_f) &= y_f
\end{aligned} \tag{5.32}$$

There are two infinitesimal symmetries

$$\{\partial_x, \partial_y\} \in \mathfrak{g} \tag{5.33}$$

Since this problem has not undergone dualization, there is no Poisson bracket and constants-of-motion cannot be generated by the symmetries. Since $\dim(\Sigma) = 3$ and $\dim(G) = \dim(\mathfrak{g}) = 2$, then $\dim(\Sigma/G) = 1$. Using $q_x \equiv x$ and $q_y \equiv y$, Problem Σ/G becomes

$$\begin{aligned}
\min_{\theta} K &= \int_0^{t_f} 1 \, \mathbb{d}t \\
\text{Subject to: } &\left.\begin{aligned} \dot{q}_x &= v\cos(\theta) \\ \dot{q}_y &= v\sin(\theta) \end{aligned}\right\} \text{quads} \\
&\left.\begin{aligned} \dot{v} &= g\sin(\theta) \end{aligned}\right\} \text{dynamical states} \\
q_x(t_0) &= q_y(t_0) = v(t_0) = 0 \\
q_x(t_f) &= q_x(t_0) + \int_0^{t_f} v\cos(\theta) \, \mathbb{d}t \\
q_y(t_f) &= q_y(t_0) + \int_0^{t_f} v\sin(\theta) \, \mathbb{d}t
\end{aligned} \tag{5.34}$$

Problem Σ/G is 1-dimensional in the sense that the velocity term is the only dynamical dimension. This is desirable from a computational standpoint since a solver

will only need to discretize velocity, while the x and y position variables are not discretized in the same manner. Many modern digital direct solvers can handle problems posed in the form above. In practice, quads are handled as integral constraints where the integration strategy is the same one used in evaluating the cost functional. For instance, nearly all orthogonal collocation methods use a high performance quadrature scheme to approximate the cost function [178]. This operation may be repeated for the quads, thereby potentially parallelizing a computation that previously was computed in a serial manner [179–185].

5.3.3 Indirect Brachistochrone

Consider the same OCP from before in the direct Brachistochrone problem. Dualization yields the following Hamiltonian system

$$\begin{aligned} H &= \lambda_x v\cos(\theta) + \lambda_y v\sin(\theta) + \lambda_v g\sin(\theta) + 1 \\ \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{v} &= g\sin(\theta) \\ \dot{\lambda_x} &= 0 \\ \dot{\lambda_y} &= 0 \\ \dot{\lambda_v} &= -\lambda_x\cos(\theta) - \lambda_y\sin(\theta) \end{aligned} \tag{5.35}$$

Clearly λ_x and λ_y are constants-of-motion due to their time rate-of-change being equal to 0. In a similar manner to Zermelo's problem, $\dot{x}$ and $\dot{y}$ are identified to be quads of the systems and the reduced problem becomes

$$\begin{aligned}
\text{dynamical states} &\begin{cases} \dot{v} = g\sin(\theta) \\ \dot{\lambda_v} = -\lambda_x\cos(\theta) - \lambda_y\sin(\theta) \end{cases} \\
\text{non-dynamical states ``quads''} &\begin{cases} \dot{q}_x = v\cos(\theta) \\ \dot{q}_v = v\sin(\theta) \end{cases} \\
\text{dynamical free parameters} &\begin{cases} \lambda_x \\ \lambda_y \end{cases} \\
\text{non-dynamical free parameters} &\begin{cases} q_{x_0} \\ q_{y_0} \end{cases}
\end{aligned} \tag{5.36}$$

When this problem is solved numerically, values for the quads are not reconstructed for the entire trajectory. Instead, only the points at the boundaries are required since the boundary conditions are a function of the quads. Though we have not yet explained in detail how to solve this type of problem, a plot of the state-space of the original OCP can be seen in Fig. 5.4. In this plot, it can clearly be seen that the 6-dimensional "traditional" BVP solves for the entire trajectory over x and y, whereas the 2-dimensional "reduced" BVP is only solved for the endpoints, thus eliminating some computations. In Chapter 6 and Ref. [177], we give more information on the specifics of the numerical solution.

One important point where problems Σ/G and $\Lambda//G$ differ is that Problem $\Lambda//G$ may be reduced an additional step using the Hamiltonian function as a constant-of-motion whereas the same reduction cannot be performed on Σ/G because the control law and Hamiltonian are unknown to it. The final reduced problems are 1-dimensional for the direct system and 0-dimensional for the indirect system. This agrees with the fact that the Brachistochrone problem is analytically solvable "by hand".

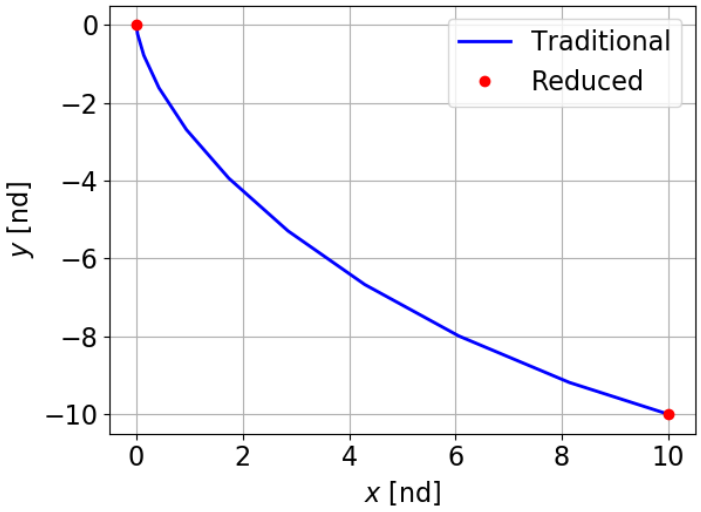


Figure 5.4. Brachistochrone state-space.

5.3.4 Free Parameters

In this example[1], we turn our attention to problems containing so-called free parameters, $\boldsymbol{p}$. Free parameters differ from states in that $\mathrm{d}\boldsymbol{p}/\mathrm{d}t = 0$. The physical implication of free parameters is that they are design parameters that are constant throughout a trajectory. For instance, some rocket motors generate a constant thrust while in use, but may be modeled as a tune-able free parameter where the OCP then aids in the selection of a rocket motor. The simplest method to incorporate free parameters into an indirect solver is to add them as states with time derivatives of zero. Despite being simple for the user, this method requires performing unnecessary computation of states and sensitivity matrices. This is particularly true for multiple shooting methods in which these calculations are computed for each arc [186, 187].

Non-dynamical parameters, which do not appear in the problem's dynamic equations or path cost, do not require an additional adjoined value. These parameters may appear in the terminal cost and the boundary conditions, or they may be internal

[1]This problem was a joint effort with Sean M. Nolan. I am grateful for his derivation of the co-parameters and their associated time rates-of-change, which I then proceeded to reduce.

to the solver. Dynamical parameters instead require the addition of corresponding "co-parameters" or $\boldsymbol{\lambda_p}$ mirroring co-states. The necessity of these co-parameters and how to treat them may be derived as the other necessary conditions from the augmented cost functional or more easily by treating parameters as states. The resulting state vector $\boldsymbol{x}'$, co-state vector $\boldsymbol{\lambda}'$, and dynamics $\boldsymbol{f}'$ are given in Eq. (5.37). The Hamiltonian remains the same as before as shown in Eq. (5.38)

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{p} \end{bmatrix} \qquad \boldsymbol{\lambda}' = \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda_p} \end{bmatrix} \qquad \boldsymbol{f}' = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \\ \boldsymbol{0} \end{bmatrix} \tag{5.37}$$

$$H = L + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{f} + \boldsymbol{\lambda_p}^{\mathrm{T}} \cdot \boldsymbol{0} = L + \boldsymbol{\lambda}'^{\mathrm{T}} \boldsymbol{f} \tag{5.38}$$

By applying the Euler-Lagrange equations, the necessary conditions for the co-parameters are given by

$$\begin{aligned} \boldsymbol{0} &= \left[\boldsymbol{\nu}_{0,\boldsymbol{p}}^{\mathrm{T}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{p}} + \boldsymbol{\lambda}_{\boldsymbol{p}}^{\mathrm{T}} \right]_{t=t_0} \\ \boldsymbol{0} &= \left[\frac{\partial}{\partial \boldsymbol{p}} \eta + \boldsymbol{\nu}_{f,\boldsymbol{p}}^{\mathrm{T}} \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}} - \boldsymbol{\lambda}_{\boldsymbol{p}}^{\mathrm{T}} \right]_{t=t_f} \\ \dot{\boldsymbol{\lambda}}_{\boldsymbol{p}} &= -\frac{\partial H}{\partial \boldsymbol{p}} \end{aligned} \tag{5.39}$$

Notably, this verifies that non-dynamical parameters do not require co-parameters because parameters are always constant and free. From Eq. (5.39), a non-dynamical parameter p_k only requires the condition that

$$0 = \frac{\partial}{\partial p_k} \eta + \nu_{0,p_k} \frac{\partial \boldsymbol{\xi}}{\partial p_k} + \nu_{0,p_k} \frac{\partial \boldsymbol{\phi}}{\partial p_k} \tag{5.40}$$

By definition, additional dynamical parameters that are treated as states are constants-of-motion since $\dot{\boldsymbol{p}} = 0$. However, their associated co-parameters are not constants-of-motion because $\dot{\boldsymbol{\lambda}}_{\boldsymbol{p}} \neq 0$. The additional states with rates equal to zero are identified as constants-of-motion that may be reduced. Using these constants-of-motion, we identify the associated set of infinitesimal symmetries

$$\begin{aligned} \mathfrak{g} &= \{ [\cdot, p_i]_{\mathrm{P}} | p_i \in \boldsymbol{p} \} \\ &= \{ \partial_{\lambda_{p_i}} | p_i \in \boldsymbol{p} \} \end{aligned} \tag{5.41}$$

The Lie algebra is spanned by the vectors corresponding to the co-parameters. Therefore, every co-parameter is a quad and the reduced system is written as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \frac{\partial H}{\partial \boldsymbol{x}} \\ \dot{\boldsymbol{\lambda}} &= -\frac{\partial H}{\partial \boldsymbol{x}} \\ \boldsymbol{\lambda}_{\boldsymbol{p}}(t_f) &= \int_{t_0}^{t_f} -\frac{\partial H}{\partial \boldsymbol{p}} \, \mathbb{d}t + \boldsymbol{\lambda}_{\boldsymbol{p}}(t_0) \end{aligned} \tag{5.42}$$

First, $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{\lambda}}$ must be solved simultaneously by numerical propagation, then the result can be used to solve for $\boldsymbol{\lambda}_{\boldsymbol{p}}(t_f)$ by numerical quadrature.

5.3.5 Clohessy-Wiltshire Equations

In spacecraft operations, the Clohessy-Wiltshire equations, or Hill's equations, are an important set of second-order differential equations that describe the relative orbital motion between two nearby spacecraft. The origin of the system is a spacecraft in a circular orbit and states (x, y, z) represent another spacecraft in a perturbed orbit about the nominal circular orbit. This setup is shown in Fig. 5.5.

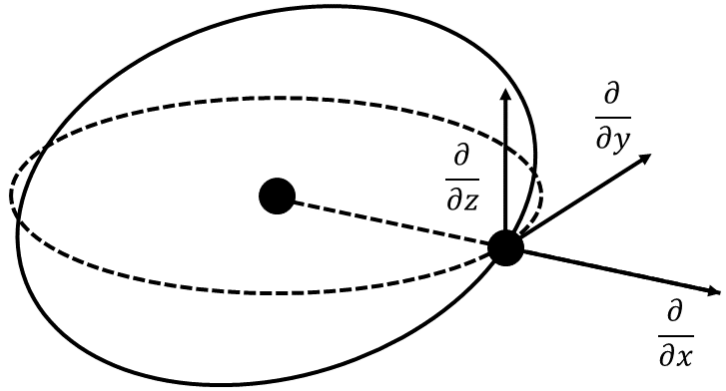


Figure 5.5. The Clohessy-Wiltshire equations govern the relative motion of a perturbed spacecraft in a rotating frame.

The equations-of-motion are

$$\begin{aligned}
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{z} &= v_z \\
\dot{v}_x &= 3n^2x + 2nv_y + F_x \\
\dot{v}_y &= -2nv_x + F_y \\
\dot{v}_z &= -2nz^2 + F_z
\end{aligned} \tag{5.43}$$

where $(F_x, F_y, F_z) = \boldsymbol{u}$ and represent a thruster's decomposed force. Assume the cost is of the form

$$\frac{\partial L}{\partial y} = 0 \tag{5.44}$$

Then y is a symmetry of the system. Because of this, λ_y is a constant-of-motion. This can easily be verified by dualizing the system

$$\begin{aligned}
H = \lambda_x v_x + \lambda_y v_y + \lambda_z v_z \\
+\lambda_{v_x}\left(3n^2x + 2nv_y + F_x\right) + \lambda_{v_y}\left(-2nv_x + F_y\right) \\
+\lambda_{v_z}\left(-2nz^2 + F_z\right) + L(\boldsymbol{x}, \boldsymbol{u}, t)
\end{aligned} \tag{5.45}$$

$$\begin{aligned}
\dot{x} &= v_x & \dot{\lambda}_x &= -3n^2\lambda_{v_x} - \frac{\partial L}{\partial x} \\
\dot{y} &= v_y & \dot{\lambda}_y &= 0 \\
\dot{z} &= v_z & \dot{\lambda}_z &= 4n\lambda_{v_z} z - \frac{\partial L}{\partial z} \\
\dot{v}_x &= 3n^2x + 2nv_y + F_x & \dot{\lambda}_{v_x} &= -\lambda_x + 2n\lambda_{v_y} - \frac{\partial L}{\partial v_x} \\
\dot{v}_y &= -2nv_x + F_y & \dot{\lambda}_{v_y} &= -\lambda_y - 2n\lambda_{v_x} - \frac{\partial L}{\partial v_y} \\
\dot{v}_z &= -2nz^2 + F_z & \dot{\lambda}_{v_z} &= -\lambda_z - \frac{\partial L}{\partial v_z}
\end{aligned} \tag{5.46}$$

Since $\dim(\Sigma) = 6$ and $\dim(G) = 1$, then we expect the reduced Hamiltonian system to have a dimension of $2\dim(\Sigma) - 2(1) = 10$. Introduce $q_y \equiv y$ and $g_y \equiv \lambda_y$, then Problem $\Sigma//G$ then becomes

$$\begin{aligned} H &= \lambda_x v_x + g_y v_y + \lambda_z v_z \\ &+\lambda_{v_x}\left(3n^2 x + 2n v_y + F_x\right) + \lambda_{v_y}\left(-2n v_x + F_y\right) \\ &+\lambda_{v_z}\left(-2n z^2 + F_z\right) + L(\boldsymbol{x}, \boldsymbol{u}, t) \end{aligned} \tag{5.47}$$

with equations-of-motion

$$\begin{aligned} \dot{x} &= v_x & \dot{\lambda}_x &= -3n^2\lambda_{v_x} - \frac{\partial L}{\partial x} \\ \dot{z} &= v_z & \dot{\lambda}_z &= 4n\lambda_{v_z} z - \frac{\partial L}{\partial z} \\ \dot{v}_x &= 3n^2 x + 2n v_y + F_x & \dot{\lambda}_{v_x} &= -\lambda_x + 2n\lambda_{v_y} - \frac{\partial L}{\partial v_x} \\ \dot{v}_y &= -2n v_x + F_y & \dot{\lambda}_{v_y} &= -g_y - 2n\lambda_{v_x} - \frac{\partial L}{\partial v_y} \\ \dot{v}_z &= -2n z^2 + F_z & \dot{\lambda}_{v_z} &= -\lambda_z - \frac{\partial L}{\partial v_z} \end{aligned} \tag{5.48}$$

and the following quad and integration rule

$$\begin{aligned} \dot{q}_y &= v_y \\ q_{yf} &= \int_{t_0}^{t_f} v_y \, \mathbb{d}t + q_{y0} \end{aligned} \tag{5.49}$$

In Eq. (5.48), we have to solve 10 differential equations which is a reduction of 2 from the unreduced Hamiltonian BVP. Then, to reconstruct the trajectory back to the full space, we solve the single differential equation in Eq. (5.49). Even though Eqs. (5.48) and (5.49) together make 11 differential equations, the reconstruction operation is completely parallelizable and a suitable high performance algorithm will scale as if there are only 10 differential equations.

5.3.6 Planar Atmospheric Flight

Consider a vehicle in atmospheric flight. Such a vehicle is subject to the lift and drag aerodynamic forces, as well as gravity. The states of the vehicle are shown in Fig. 5.6.

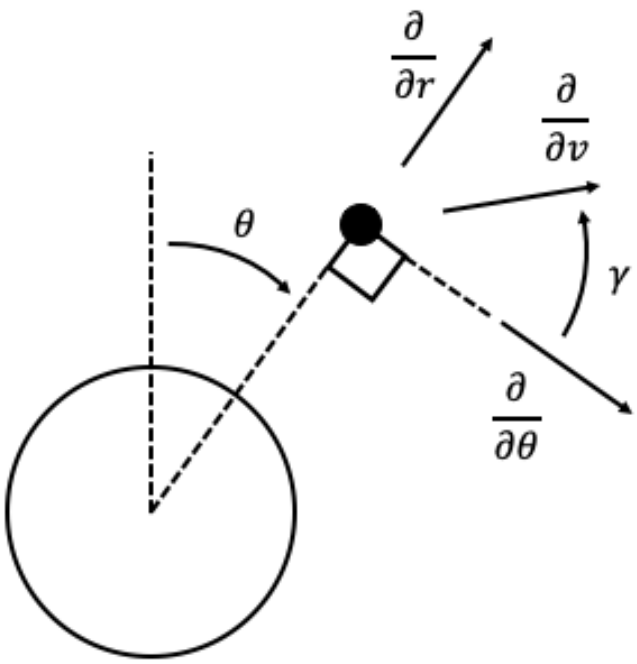


Figure 5.6. The planar configuration of a vehicle in atmospheric flight over a spherical Earth.

Problem Σ is stated as follows (note that γ in this context is the flight-path angle of the vehicle) [188]

$$\begin{aligned}
\min_{\alpha} K &= \int_0^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\, \mathrm{d}t + \eta(\boldsymbol{x}(t_f), t_f)) \\
\dot{h} &= v \sin(\gamma) \\
\dot{\theta} &= \frac{v \cos(\gamma)}{h + R_e} \\
\dot{v} &= -\frac{1}{2}\frac{\rho v^2 C_d A_{ref}}{m} - \frac{\mu \sin(\gamma)}{r^2} \\
\dot{\gamma} &= \frac{1}{2}\frac{\rho v^2 C_l A_{ref}}{mv} + \frac{v \cos(\gamma)}{r} - \frac{\mu \cos(\gamma)}{v r^2} \\
0 &= \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0) \\
0 &= \boldsymbol{\xi}(\boldsymbol{x}(t_f), t_f)
\end{aligned} \tag{5.50}$$

Then, assume the cost is of the form

$$\frac{\partial L}{\partial \theta} = 0 \tag{5.51}$$

Some problems that fit this criteria (up to coordinate transformation) are the supersonic minimum time to climb problem [189], ascent trajectories for launch vehicles [9, 190], atmospheric re-entry [191], and aerobraking [192]. In these examples, downrange is a symmetry of the system, $\{\partial_\theta\} \in \mathfrak{g}$. Additionally, there are two more symmetries associated with the system; however, these symmetries are associated with the crossrange and the heading of the vehicle and do not apply to the planar problem [193]. Because the resulting constants-of-motion are not involutive, they are beyond the scope of this procedure. Using only downrange, dualization and subsequent reduction yields the 6-dimensional dynamical system

$$\begin{aligned} H = {} & \lambda_h(v\sin(\gamma)) + g_\theta\left(\frac{v\cos(\gamma)}{h+R_e}\right) \\ & +\lambda_v\left(-\frac{1}{2}\frac{\rho v^2 C_d A_{ref}}{m} - \frac{\mu\sin(\gamma)}{r^2}\right) \\ & +\lambda_\gamma\left(\frac{1}{2}\frac{\rho v^2 C_l A_{ref}}{mv} + \frac{v\cos(\gamma)}{r} - \frac{\mu\cos(\gamma)}{vr^2}\right) \\ & +L(\boldsymbol{x},\boldsymbol{u},t) \end{aligned} \tag{5.52}$$

$$\begin{aligned} \dot{h} &= \frac{\partial H}{\partial \lambda_h} & \dot{\lambda_h} &= -\frac{\partial H}{\partial h} \\ \dot{v} &= \frac{\partial H}{\partial \lambda_v} & \dot{\lambda_v} &= -\frac{\partial H}{\partial v} \\ \dot{\gamma} &= \frac{\partial H}{\partial \lambda_\gamma} & \dot{\lambda_\gamma} &= -\frac{\partial H}{\partial \gamma} \end{aligned} \tag{5.53}$$

In addition to the dynamics above, there is the additional free parameter, g_θ, and its associated quad

$$\dot{\theta} = \frac{\partial H}{\partial g_\theta} \tag{5.54}$$

Although a study on performance was not done, we numerically solved a single unconstrained mission for a manuevering hypersonic vehicle. The trajectories generated as a result are shown in Figs. 5.7 and 5.8. In Fig. 5.7, we see a fully reconstructed trajectory from the 8-dimensional "traditional" problem, and two points at the boundaries for the 6-dimensional "reduced" problem. This may be slightly misleading since the altitude is fully reconstructed for both problems. It is not shown Fig. 5.7 because

the downrange consists of just two points at the boundaries. Instead, we confirm that the vehicle's states at the boundaries satisfy the mission requirements for both problems. In Fig. 5.8, we see that both altitude and velocity are fully reconstructed for both problems. The energy of both vehicles over the course of their trajectories are in agreement with one-another. Overall these problems indirectly solve the same OCP, yet the reduced formulation uses fewer numerical computations in doing so.

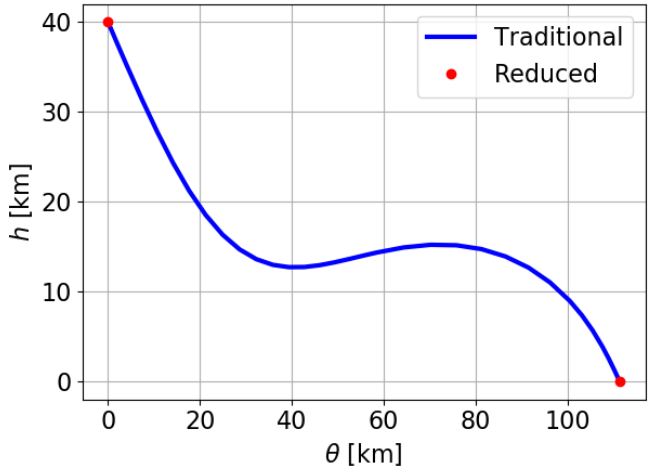


Figure 5.7. Trajectory of a maneuvering hypersonic vehicle.

The formulation shown in this example is valid for any missions that do not explicitly contain downrange in the cost function. Commonly used path-constraints in the hypersonic atmospheric flight problems are bounded angle-of-attack, heat rate, and G-loading. All of these path-constraints do not explicitly use downrange and therefore also have equivalent reduced formulations.

5.4 Summary

In this chapter, we introduced the central theme to this dissertation. We showed how using Marsden-Weinstein-Meyer reduction enables an aerospace system designer to eliminate more unknown quantities from their system than is allowed by traditional

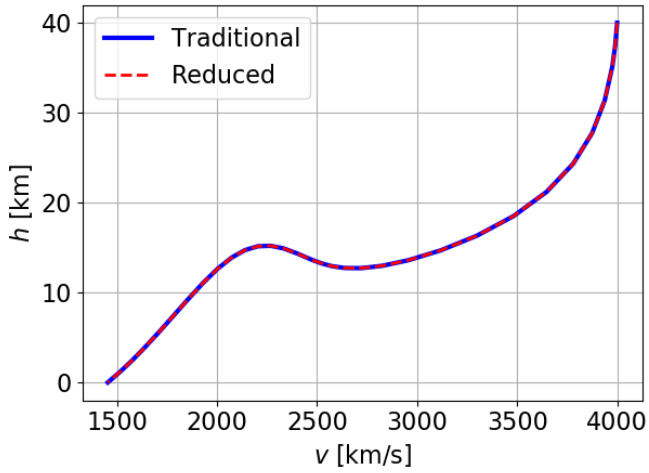


Figure 5.8. Energy of a maneuvering hypersonic vehicle.

indirect methods. In doing so, we explored some of the topological results of this theorem and discovered that for practical applications, an aerospace mission must be modeled on a principal G-bundle whose base space is a symplectic manifold and fibers are isomorphic to Lie groups. This is the first major result of this chapter and has a significant impact on how we will design BVP solvers in Chapter 6.

In other words, the states in the fibers of the G-bundle were identified to be non-dynamical "quads". This means their numerical solution is decoupled from the recursive integration required for dynamical equations-of-motion, or rather the states on the base space of the G-bundle. This was done by eliminating causality in the system. This directly enables integration of the quads using batch integration in a completely parallel manner in lieu of single-threaded recursive integration strategies.

Next, we explored how we will explicitly carry out reduction in an automated manner. We created a strategy that used both Noether's and Liouville's theorems that applies to the general set of problems with involutory functions. In carrying out reduction, there were special cases where we generated 0-dimensional BVPs. This means that this process can almost solve equations analytically, save for the terms at the boundaries. This is the second major contribution of this chapter and has signif-

icant impact on further application of these strategies to general-purpose aerospace mission design.

6. NUMERICAL SOLUTION TO REDUCED-DIMENSIONAL BOUNDARY-VALUE PROBLEMS

In this chapter, we will create novel BVP solvers that can solve BVPs as defined in Chapter 5. To give an overview, one key result from trajectory optimization using indirect methods is that the infinite-dimensional optimization problem is transcribed into a finite-dimensional BVP on a manifold. Various reduction techniques exist that may reduce this BVP to an equivalent, but lower dimensional problem. One such reduction technique is MWM reduction; however, our BVP solvers are not restricted to just using this result. In fact, our BVP solvers will be posed in a manner that support several reduction techniques such as MWM reduction [118, 119], Poisson reduction [120], Lagrangian (Routhian) reduction [194], and more.

There is a relatively modern and rich application of geometric structure to the numerical integration of Initial Value-Problems (IVPs) [195–199]. A common result from these studies generally suggest that numerical integrators leveraging dynamical structure yield higher quality results using fewer calculations over numerical integrators that do not leverage this structure.

While IVPs with geometric structure have proliferated and seen great success, there has been limited application to BVPs [200, 201]. In fact, there appear to be no BVP solvers that can solve BVPs defined on principal G-bundles. Since the BVPs generated from the process in Chapter 5 result in a mathematical problem defined on a principal G-bundle, the contribution of this chapter is to design algorithms that can solve these problems. To do this, we will first analyze the resulting BVP from the reduction of constants-of-motion. Next, we will look at the result from the reduction of symmetries. By looking at these two types of reduction independently, we will

then be able to define a general algorithm that can handle both types of reduction simultaneously. The result is a class of algorithms that can handle reduced BVPs from MWM reduction and other types of reduction that result in a similar structure.

6.1 Boundary-Value Problems on Reduced Manifolds

6.1.1 Reduction of Constants-of-Motion

A standard BVP is described as a set of first-order differential equations on a manifold, M, along with boundary conditions as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}) \\ \boldsymbol{\Phi}(t_0, \boldsymbol{x}_0, t_f, \boldsymbol{x}_f, \boldsymbol{p}) &= 0 \end{aligned} \tag{6.1}$$

Assume $\boldsymbol{f} \in M$ with no additional structure. Then the existence of n constants-of-motion allows one to reduce M by n dimensions as follows. Let $\boldsymbol{\mu}$ be the map [202]

$$\begin{aligned} \boldsymbol{J} &: M \to N \\ (t, \boldsymbol{x}, \boldsymbol{p}) &\mapsto \boldsymbol{\mu}\,(t, \boldsymbol{x}, \boldsymbol{p}) \end{aligned} \tag{6.2}$$

If $\boldsymbol{\mu}$ is a regular value of $\boldsymbol{J}$, then define the reduced dimensional manifold by the restriction $\boldsymbol{J}^{-1}(\boldsymbol{\mu})$. In this case, the system's true dynamical motion lives in the fibers of $\boldsymbol{\mu}$. The new parameters for the problem is the set of original parameters concatenated together with the regular value of $\boldsymbol{J}$

$$\boldsymbol{p}^* = [\boldsymbol{p}, \boldsymbol{\mu}] \tag{6.3}$$

We include the regular value as a parameter in the root-solving process since the fiber containing optimal motion is not known. This enables a root-solver to select the optimal value for the constant-of-motion, and therefore an optimal fiber. The new dynamical system is defined on the reduced dimensional manifold with reduced coordinates, $\boldsymbol{x}^* \in \boldsymbol{J}^{-1}(\boldsymbol{\mu})$ as

$$\dot{\boldsymbol{x}}^* = \boldsymbol{f}^*\,(t, \boldsymbol{x}^*, \boldsymbol{p}^*) \in \boldsymbol{J}^{-1}\,(\boldsymbol{\mu}) \tag{6.4}$$

Altogether, the reduced BVP is stated as

$$\begin{aligned} \dot{\boldsymbol{x}}^* &= \boldsymbol{f}^*\left(t, \boldsymbol{x}^*, \boldsymbol{p}^*\right) \\ \boldsymbol{\Phi}(t_0, \boldsymbol{x}_0^*, t_f, \boldsymbol{x}_f^*, \boldsymbol{p}^*) &= 0 \end{aligned} \tag{6.5}$$

Note that while the dynamical system has changed since $\dim \boldsymbol{f}^* < \dim \boldsymbol{f}$, the boundary conditions have not. Rather the original boundary conditions are used with a new parameterization. This format is compatible with most modern day boundary-value problem solvers because the only additional capability required by such a solver is the ability to adjust the parameters, $\boldsymbol{p}^*$. Since constants-of-motion by definition do not change over a trajectory; therefore, $\boldsymbol{\mu}(t_0)$ and $\boldsymbol{\mu}(t_f)$ are contained in $\boldsymbol{p}^*$ and no additional computations are necessary.

6.1.2 Reduction of Symmetries

Let the standard BVP be defined the same as in Eq. (6.1)

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}) \\ \boldsymbol{\Phi}(t_0, \boldsymbol{x}_0, t_f, \boldsymbol{x}_f, \boldsymbol{p}) &= 0 \end{aligned} \tag{6.6}$$

Assume $\boldsymbol{f} \in M$ with no additional structure. Then the existence of n symmetries allows one to reduce M by n dimensions as follows. Let G be the Lie group corresponding to the symmetries with action

$$\begin{aligned} S : G \times M &\to M \\ (g, \boldsymbol{x}) &\mapsto g\boldsymbol{x} \end{aligned} \tag{6.7}$$

If S_g is the action on M for every $g \in G$, then for an $\boldsymbol{x} \in M$ the orbit of the group action is [203]

$$O_{\boldsymbol{x}} = \{S_g(\boldsymbol{x}) | g \in G\} \tag{6.8}$$

Using all orbits on M, the reduced dimensional manifold is then defined as

$$M/G \equiv \{O_{\boldsymbol{x}} | \boldsymbol{x} \in M\} \tag{6.9}$$

Next, we introduce coordinates on the reduced dimensional manifold as $\boldsymbol{x}^* \in M/G$. Since the Lie group is also a manifold, introduce coordinates on it as $\boldsymbol{q} \in G$. The new dynamical system is defined on the reduced dimensional manifold with reduced coordinates as

$$\dot{\boldsymbol{x}}^* = \boldsymbol{f}^*(t, \boldsymbol{x}^*, \boldsymbol{p}) \in M/G \tag{6.10}$$

The new dynamical system is not a function of the coordinates in the symmetry space by definition, $\boldsymbol{f}^* \neq \boldsymbol{f}^*(\boldsymbol{q})$. Dynamical variables that live in the symmetry space are not constants-of-motion, $\boldsymbol{q}_0 \neq \boldsymbol{q}_f$; therefore, the terminal values must be reconstructed. Define $\boldsymbol{h}$ to be the equations-of-motion that live in the symmetry space such that

$$\dot{\boldsymbol{q}} = \boldsymbol{h}(t, \boldsymbol{x}^*, \boldsymbol{p}) \tag{6.11}$$

Since $\boldsymbol{h} \neq \boldsymbol{h}(\boldsymbol{q})$, the $\dot{\boldsymbol{q}}$ do not need to be numerically integrated with a recursive scheme such as RK4. However, there may not be a closed-form solution for $\boldsymbol{x}^*$, so there may not be an algebraic closed-form solution for $\boldsymbol{q}$ as a result since $\boldsymbol{h} = \boldsymbol{h}(\boldsymbol{x}^*)$. To evaluate $\boldsymbol{q}_f$ without recursive integration, we rewrite $\boldsymbol{q}_f$ in an integral form

$$\boldsymbol{q}_f = \int_{t_0}^{t_f} \boldsymbol{h}(t, \boldsymbol{x}^*, \boldsymbol{p}) \, \mathrm{d}t + \boldsymbol{q}_0 \tag{6.12}$$

Knowing a defined time-history $\boldsymbol{x}^* = \boldsymbol{x}^*(t)$, the integral in Eq. (6.12) is only a function of time. The integral is evaluated using a batch integration scheme such as Simpson's rule. This equation appears similar to a recursive integration; however, everything under the integral is known a priori so this implies there is an inherent parallelism to this operation that is otherwise unavailable. Altogether the reduced boundary-value problem is stated as

$$\begin{aligned} \dot{\boldsymbol{x}^*} &= \boldsymbol{f}^*(t, \boldsymbol{x}^*, \boldsymbol{p}) \\ \dot{\boldsymbol{q}} &= \boldsymbol{h}(t, \boldsymbol{x}^*, \boldsymbol{p}) \\ \boldsymbol{\Phi}(t_0, \boldsymbol{x}_0^*, \boldsymbol{q}_0, t_f, \boldsymbol{x}_f^*, \boldsymbol{q}_f, \boldsymbol{p}) &= 0 \end{aligned} \tag{6.13}$$

Note that, similar to the reduction of constants-of-motion case, the dynamical system has changed but the boundary conditions have not. The boundary conditions simply

have a new parameterization. The form of Eq. (6.13) is not standard because modern boundary-value problem solvers do not treat $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{x}}$ as separate quantities.

6.1.3 Combined Reduction

In the presence of either symmetries, constants-of-motion, or additional structure on M, many reduction theorems allow one to reduce a dynamical system by more states than there are known symmetries or constants-of-motion. The concept is analogous to Noether's theorem where, when there are additional structures on M, knowledge of the symmetries yields constants-of-motion and vice versa [97]. For example, in MWM reduction, the existence of n constants-of-motion allows one to reduce a system by more than n-dimensions, but no more than $2n$-dimensions. In general, reduction processes involve eliminating either constants-of-motion whose time rates of change are 0, or eliminating symmetries whose time rates of change are given by a Lie group action. A general dynamical system containing both information on eliminated constants-of-motion and symmetries may be written as

$$\begin{aligned} \dot{\boldsymbol{x}}^* &= \boldsymbol{f}^*(t, \boldsymbol{x}^*, \boldsymbol{p}^*) \\ \dot{\boldsymbol{q}} &= \boldsymbol{h}(t, \boldsymbol{x}^*, \boldsymbol{p}^*) \\ \boldsymbol{\Phi}(t_0, \boldsymbol{x}_0^*, \boldsymbol{q}_0, t_f, \boldsymbol{x}_f^*, \boldsymbol{q}_f, \boldsymbol{p}^*) &= 0 \end{aligned} \tag{6.14}$$

To solve Eq. (6.14) in a numerical boundary-value problem solver, $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{x}}^*$ may be concatenated and then propagated alongside each other. This forgoes the benefits from the reduction of symmetries. In Ref. 151, Ohsawa claims that problems which have been symmetry reduced are desirable from a computational standpoint due to their lower dimension. The current status quo of boundary-value problem solvers ignores information contained in $\boldsymbol{h}$, thereby preventing practical implementation. In the next section, we present new numerical algorithms that explicitly take $\boldsymbol{h}$ into account, thus enabling analysis on reduced dimensional manifolds.

6.2 Numerical Algorithms

We present two numerical algorithms for solving boundary-value problems on reduced dimensional manifolds. We assume dynamical systems are already reduced and are of the form in Eq. (6.14); however, the "star" notation is dropped.

6.2.1 Reduced Dimensional Collocation

Collocation is a numerical method of solving optimal control problems. One type of collocation is called Hermite-Simpson collocation [56]. It is based on the assumption that optimal trajectories may be approximated by piece-wise third-order polynomials. To define a third-order piece-wise polynomial, four parameters are required. At each node of a polynomial, there is state position information along with state gradient information. Taking $\boldsymbol{x}_i$ to be states at node i, the coefficients for each polynomial are then

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_i \\ \dot{\boldsymbol{x}}_i \\ \boldsymbol{x}_{i+1} \\ \dot{\boldsymbol{x}}_{i+1} \end{bmatrix} \tag{6.15}$$

To ensure the polynomial is an accurate approximation of the underlying dynamical system, the predicted gradient at the midpoint of each polynomial is compared to the actual gradient from the equations-of-motion. The predicted values between each node are

$$\begin{aligned} \tilde{\boldsymbol{x}}_{i+1/2} &= \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) + \frac{t_f(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_{i+1})}{8(N_{\text{total}} - 1)} \\ \dot{\tilde{\boldsymbol{x}}}_{i+1/2} &= -\frac{3}{2}\frac{(N_{\text{total}} - 1)}{t_f(\boldsymbol{x}_i - \boldsymbol{x}_{i+1})} - \frac{1}{4}(\dot{\boldsymbol{x}}_i + \dot{\boldsymbol{x}}_{i+1}) \end{aligned} \tag{6.16}$$

where $\tilde{\boldsymbol{x}}$ and $\dot{\tilde{\boldsymbol{x}}}$ refer to the predicted states and N_{total} is the total number of collocation nodes. The error between the predicted and actual values for the rate of change vanish when the curve is an accurate prediction, and therefore this condition is included as

a constraint into the NLP solver. Using these predicted values of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, the optimization problem becomes the following BVP

$$\begin{aligned} 0 &= \boldsymbol{f}(t_{i+1/2}, \tilde{\boldsymbol{x}}_{i+1/2}, \boldsymbol{p}) - \dot{\tilde{\boldsymbol{x}}}_{i+1/2} \\ 0 &= \boldsymbol{\Phi}(t_0, \boldsymbol{x}_0, \boldsymbol{q}_0, t_f, \boldsymbol{x}_f, \boldsymbol{q}_f, \boldsymbol{p}) \end{aligned} \tag{6.17}$$

The additional parameters $\boldsymbol{p}$ are constants and other non-dynamical quantities that may be adjusted by the solver. Since this algorithm solves a BVP, there is no explicit cost function to minimize. One issue with using modern day collocation solvers is that the $\boldsymbol{q}$ are ignored. This can be remedied by including the $\boldsymbol{q}_0$ as parameters in an NLP-solver; however, the $\boldsymbol{q}_f$ are dependent on the $\boldsymbol{q}_0$ and therefore are not parameters. Instead, after the solver has generated a third-order polynomial approximation, but prior to its update step, we reconstruct $\boldsymbol{q}_f$ by integrating over the trajectory

$$\boldsymbol{q}_f = \int_{t_0}^{t_f} \boldsymbol{h}(t, \boldsymbol{x}(t), \boldsymbol{p}) \, \mathrm{d}t + \boldsymbol{q}_0 \tag{6.18}$$

Since third-order polynomials are used to model the dynamics, $\boldsymbol{x}(t)$, we expect the integral in Eq. (6.18) to be accurate when evaluated by Simpson's rule.

Integration of Eq. (6.18) by numerical quadrature can be done in a highly efficient vectorized manner. Increasing the number of nodes will yield a more accurate integral without a significant impact on performance. Note that since $\partial_{\boldsymbol{p}} \boldsymbol{h} \neq 0$, sensitivities of $\boldsymbol{q}$ with respect to $\boldsymbol{p}$ must be computed. Because $\boldsymbol{x}$, $\boldsymbol{q}_0$, and $\boldsymbol{p}$ are all general parameters in an NLP-solver, these sensitivities are captured during the gradient estimation and update step. Shooting methods do not have this luxury, which we will explore in the next section.

6.2.2 Reduced Dimensional Shooting

Another common algorithm for solving boundary-value problems are shooting methods [204]. Instead of using meshes like in collocation and pseudospectral methods, shooting methods capture sensitivity information by using a state-transition matrix, Δ, with initial and terminal states Δ_0 and Δ_f. One important fact to keep

in mind is that variations in $\boldsymbol{q}$ do not affect $\boldsymbol{f}$ and $\boldsymbol{h}$, but variations in $(\boldsymbol{x}, \boldsymbol{p})$ affect $\boldsymbol{f}$ and $\boldsymbol{h}$. We construct the state-transition matrix by first defining the sensitivity matrix of the equations-of-motion

$$A(t) = \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} & \frac{\partial f_1}{\partial p_1} & \frac{\partial f_1}{\partial p_2} & \cdots & \frac{\partial f_1}{\partial p_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} & \frac{\partial f_2}{\partial p_1} & \frac{\partial f_2}{\partial p_2} & \cdots & \frac{\partial f_2}{\partial p_m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} & \frac{\partial f_n}{\partial p_1} & \frac{\partial f_n}{\partial p_2} & \cdots & \frac{\partial f_n}{\partial p_m} \end{bmatrix} \tag{6.19}$$

Then, the state-transition matrix is defined as the following set of first-order differential equations

$$\Delta_0 = \begin{bmatrix} Id & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{6.20}$$

$$\dot{\Delta} = A(t)\Delta$$

Or more explicitly

$$\dot{\Delta} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} & \frac{\partial f_1}{\partial p_1} & \frac{\partial f_1}{\partial p_2} & \cdots & \frac{\partial f_1}{\partial p_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} & \frac{\partial f_2}{\partial p_1} & \frac{\partial f_2}{\partial p_2} & \cdots & \frac{\partial f_2}{\partial p_m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} & \frac{\partial f_n}{\partial p_1} & \frac{\partial f_n}{\partial p_2} & \cdots & \frac{\partial f_n}{\partial p_m} \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \Delta \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \tag{6.21}$$

where the bottom $\dim(\boldsymbol{p})$ rows are removed prior to assignment in $\dot{\Delta}$. These rows need not be propagated since $\partial \boldsymbol{p}/\partial \boldsymbol{x} = 0$ by definition. Sensitivities in the boundary conditions are given by the Jacobian matrix

$$\begin{aligned} \mathscr{J} &= [M_0, P, Q_0] \\ M_0 &= \frac{\mathrm{d}\boldsymbol{\Phi}(t_0, \boldsymbol{x}_0, \boldsymbol{q}_0, t_f, \boldsymbol{x}_f, \boldsymbol{q}_f, \boldsymbol{p})}{\mathrm{d}\boldsymbol{x}_0} \\ P &= \frac{\mathrm{d}\boldsymbol{\Phi}(t_0, \boldsymbol{x}_0, \boldsymbol{q}_0, t_f, \boldsymbol{x}_f, \boldsymbol{q}_f, \boldsymbol{p})}{\mathrm{d}\boldsymbol{p}} \\ Q_0 &= \frac{\mathrm{d}\boldsymbol{\Phi}(t_0, \boldsymbol{x}_0, \boldsymbol{q}_0, t_f, \boldsymbol{x}_f, \boldsymbol{q}_f, \boldsymbol{p})}{\mathrm{d}\boldsymbol{q}_0} \end{aligned} \tag{6.22}$$

Note that we use the total derivative as opposed to partial derivative. This was done to capture the changes in $\boldsymbol{x}_f$ and $\boldsymbol{q}_f$ due to the perturbations in $\boldsymbol{x}_0$ and $\boldsymbol{q}_0$. The error vector in the boundary conditions is given as

$$\Phi(t_0, \boldsymbol{x}_0, \boldsymbol{q}_0, t_f, \boldsymbol{x}_f, \boldsymbol{q}_f, \boldsymbol{p}) = \epsilon \tag{6.23}$$

then the shooting algorithm uses Newton's Method to solve the linear system

$$\mathscr{J}\, \boldsymbol{x}_{update} = -\epsilon \tag{6.24}$$

In collocation, $\boldsymbol{q}_0$, $\boldsymbol{x}$, and $\boldsymbol{p}$ are included as parameters in the NLP-solver; therefore, changes in $\boldsymbol{q}_f$ due to perturbations in any of these parameters are captured by gradient information in the NLP-solver. In shooting algorithms, points along the trajectory between $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ are not explicitly parameters. Since adjustments to $\boldsymbol{x}_0$ change which fiber the full trajectory lives in, $\boldsymbol{q}_f$ is affected by any changes made in the dynamics. Using Eq. (6.18) is sufficient in determining $\boldsymbol{q}_f$ over an unperturbed trajectory; however, when integrating over a perturbed trajectory we use the full time-dependent state-transition matrix to capture this information

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{q}_f}{\mathrm{d}\boldsymbol{x}_0} &= \int_{t_0}^{t_f} \boldsymbol{h}(t, \Delta_t\left[\mathrm{d}\boldsymbol{x}_0, \boldsymbol{0}\right] + \boldsymbol{x}(t), \boldsymbol{p}) - \boldsymbol{h}(t, \boldsymbol{x}(t), \boldsymbol{p})\, \mathrm{d}t \\ \frac{\mathrm{d}\boldsymbol{q}_f}{\mathrm{d}\boldsymbol{p}} &= \int_{t_0}^{t_f} \boldsymbol{h}(t, \Delta_t\left[\boldsymbol{0}, \mathrm{d}\boldsymbol{p}\right] + \boldsymbol{x}(t), \boldsymbol{p} + \mathrm{d}\boldsymbol{p}) - \boldsymbol{h}(t, \boldsymbol{x}(t), \boldsymbol{p})\, \mathrm{d}t \end{aligned} \tag{6.25}$$

Note that, similar to Eq. (6.18), the integrals in Eq. (6.25) are over a known trajectory and may be evaluated using Simpson's rule or another batch integration scheme.

Although our implementation in this dissertation did not use parallelization, the batch integration here is totally parallelizable. This is in contrast to all other BVP parallelization techniques, such as multiple shooting, which use parallelization "tricks" and do not completely scale to the available computational resources. There is more info on the improvements that will benefit performance in Section 9.3.

6.3 Results

6.3.1 Modified FORTRAN BVP Test Cases

In Ref. 205, Mazzia, et. al. provide several boundary-value problems as a test set. Three of these test cases, referred to as "T2", "T8", and "T18", contain a symmetry that may be eliminated. The general form of the three problems is

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= s\frac{x_2}{\eta} \\ x_1(0) &= x_{10} \\ x_1(1) &= x_{1f} \end{aligned} \tag{6.26}$$

This problem has adjustable parameters s, x_{10}, and x_{1f}, in addition to a difficulty parameter η. Altering these parameters produce different families of solutions. The values used for the parameters are listed in Table 6.1.

Table 6.1. Parameters for problems T2, T8, and T18

	T2	T8	T18
s	1	-1	-1
x_{10}	1	1	1
x_{1f}	0	2	$\exp(-1/\eta)$

Since x_1 does not appear in the dynamics, it does not need to be propagated with x_2. The reduced system is in Eq. (6.27).

$$\begin{aligned} \dot{q}_1 &= x_2 \\ \dot{x}_2 &= s\frac{x_2}{\eta} \\ q_1(0) &= x_{10} \\ q_1(1) &= x_{1f} \end{aligned} \tag{6.27}$$

While Eq. (6.26) and Eq. (6.27) appear identical aside from notation, the key feature of Eq. (6.27) is that it is lower dimensional. Instead of a numerical BVP solver having to solve two differential equations, only the differential equation for $\dot{x}_2$ needs to be solved. Then, approximating $q_1(1)$ can be done in a single operation with Simpson's rule. The reduced problem ultimately has fewer equations to satisfy. This directly translates to a BVP solver using fewer operations to satisfy these equations. Using the same parameters from the "T" problems, we refer to the reduced problems as R2, R8, and R18 respectively. Solving problems R2, R8, and R18 indirectly solve problems T2, T8, and T18. Solutions to the "R" problems are shown in Fig. 6.1.

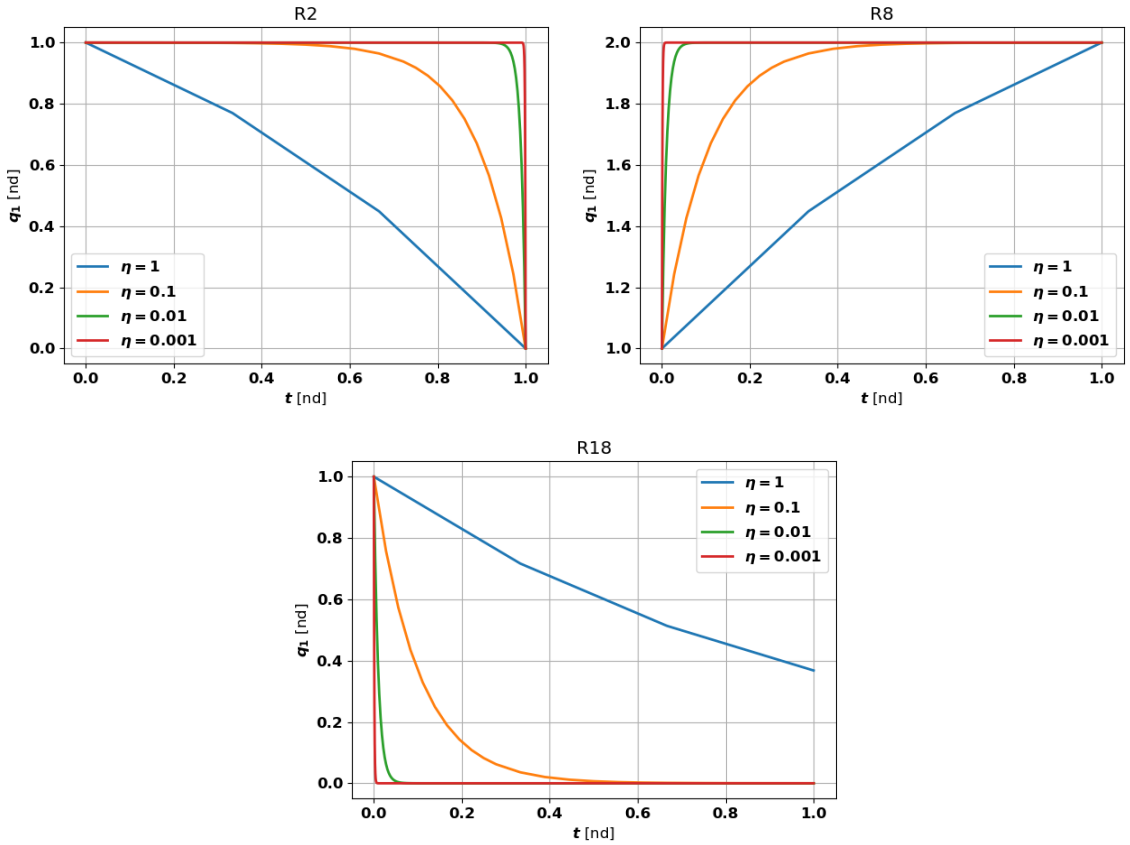


Figure 6.1. Solutions to R2, R8, and R18 with various difficulty parameters.

Comparing the "T" problems to the "R" problems in terms of performance, we get the following times to solution across varying numbers of nodes:

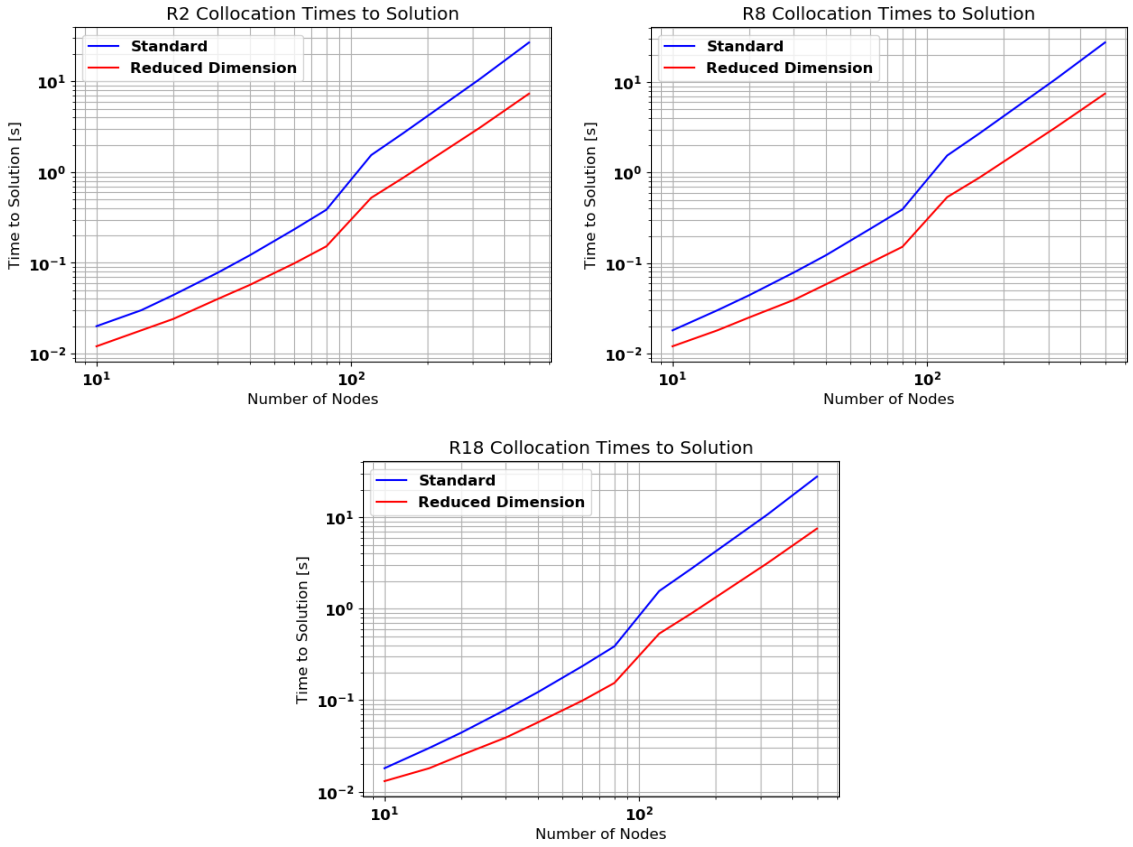


Figure 6.2. Times to solution of R2, R8, and R18 with $\eta = 1 \times 10^{-2}$.

Unsurprisingly, the impact of reduction is similar across R2, R8, and R18. The onset of computational complexity has been "delayed" by reduction. By offloading the computation of numerical solutions to differential equations to quadrature rules, a designer may use more nodes to obtain a higher quality solution at a lower cost. On the other hand, the following times to solve R2, R8, and R18 using a reduced dimensional shooting method are shown in Table 6.2. In this table, we also listed the number of differential equations for each system. Overall, the original system needed to integrate 6 differential equations recursively. The reduced system needed to integrate 4 differential equations, but 2 of these can be done in completely parallel manner with batch integration. Therefore, the reduced system will scale as if it is only integrating 2 differential equations.

Table 6.2. Results for the Shooting Method with $\eta = 1 \times 10^{-2}$

	Reduced Systems	Standard Systems
EOMs	1	2
STM Size	1	4
Quads	1	0
Quads STM Size	1	0
Times to Solution [T/R2, T/R8, T/R18] [s]	[1.76, 0.81, 0.81]	[1.19, 0.62, 0.61]

Interestingly, the solution times for the reduced dimensional "R" problems were slower than the "T" problems. This could be due to a number of different reasons. It is possible that this problem is not well-suited for shooting methods to begin with. Since reduction does not guarantee that the geometry of the reduced manifold is simpler, it is possible that numerical challenges existing in the standard problem are made worse by reduction. Another likely cause of this slightly worse time to solution is the digital implementation. In an effort to make a fair comparison between the reduced and standard formulations, the same numerical software was used for both. Also, the batch integration was performed in a single-threaded manner, thus eliminating any benefits of parallelization. This numerical software is not considered high-performance and the additional overhead caused by handling q_1 as a special term may have outweighed the benefits. A high performance solver would eliminate these issues. To the author's knowledge, there is no such high-performance solver in existence and is a point of future work Section 9.3.

6.3.2 Brachistochrone Problem

Recall the Brachistochrone problem from Chapter 5. We now set up the equations-of-motion and separate them based on their dynamical structure

$$
\begin{aligned}
&\boldsymbol{h}\begin{cases}\dot{x} = v\cos(\gamma)\\ \dot{y} = v\sin(\gamma)\end{cases}\\
[\boldsymbol{p},\boldsymbol{\lambda}] = \boldsymbol{p}^* \ni \boldsymbol{\lambda}&\begin{cases}\dot{\lambda}_x = 0\\ \dot{\lambda}_y = 0\end{cases}\\
&\boldsymbol{f}\begin{cases}\dot{v} = -g\sin(\gamma)\\ \dot{\lambda}_v = -\lambda_x\cos(\gamma) - \lambda_y\sin(\gamma)\end{cases}
\end{aligned}
\tag{6.28}
$$

Clearly, λ_x and λ_y are constants-of-motion since their time rates of change are 0. Since x does not appear explicitly in the equations-of-motion, then ∂_x is a symmetry, and similarly ∂_y is a symmetry. The only states that must be enforced through collocation or propagation are (v, λ_v), while (λ_x, λ_y) are guaranteed to be satisfied and (x, y) are determined by quadrature. The final time, t_f, is also included as a parameter to nondimensionalize the system with respect to time.

Table 6.3 shows a table of the number of equations-of-motion that must be solved. There were a total of 48 differential equations in the original system that must be solved in a single-threaded manner. In the reduced system, there were a total of 24 differential equations that must be solved. Without further exploitation of the fact that quads exist in the fibers of a G-bundle, this is already a 50% reduction in the number of differential equations that must be solved. If we further exploit this fact by integrating the quads using a completely parallel batch integration scheme, the reduced problem will scale as if it only has 12 differential equations total. This means that a theoretical upper bound on performance gain for this problem is an analytical reduction of 75%.

Solution processes on the reduced and standard spaces were run for different numbers of collocation nodes. In addition to the number of equations, Table 6.3 shows a comparison between shooting and collocation with 30 nodes. The reduced

system is significantly faster than the standard system by virtue of having fewer shooting equations-of-motion and fewer collocation parameters. Solutions and times to solution across several different numbers of collocation nodes are shown in Fig. 6.3 and Fig. 6.4. As the number of collocation nodes increases, there are diminishing returns on computational performance. Similar to the previous T/R2, T/R8, and T/R18 examples, we see the familiar impact where the reduced system delays the onset of the point of diminishing returns. This again enables a designer to use more nodes in an effort to capture sensitivities within a trajectory. For instance, at 320 nodes the standard boundary-value problem took 912.6 seconds to converge while reducing the dimension cut this time down to 315.5 seconds. Depending upon the application, the standard system may be too costly to run at the fine resolution of 320 nodes while the reduced system may not be.

Table 6.4 lists the residual after each shooting update. Note that prior to the first update, in iteration 0, the residuals are very different. The initial guesses differ slightly since the standard system requires a guess for the terminal states, while in the reduced system, the states that are transcribed to $\boldsymbol{f}$ do not need their final positions given as an initial guess.

Table 6.3. Information on the Brachistochrone problem setup (30 collocation nodes)

	Reduced System	Standard System
EOMs	2	6
STM Size	10	42
Quads	2	0
Quads STM Size	10	0
Collocation Parameters	65	181
Collocation Time to Solution	2.728 s	4.829 s
Shooting Time to Solution	1.114 s	1.633 s

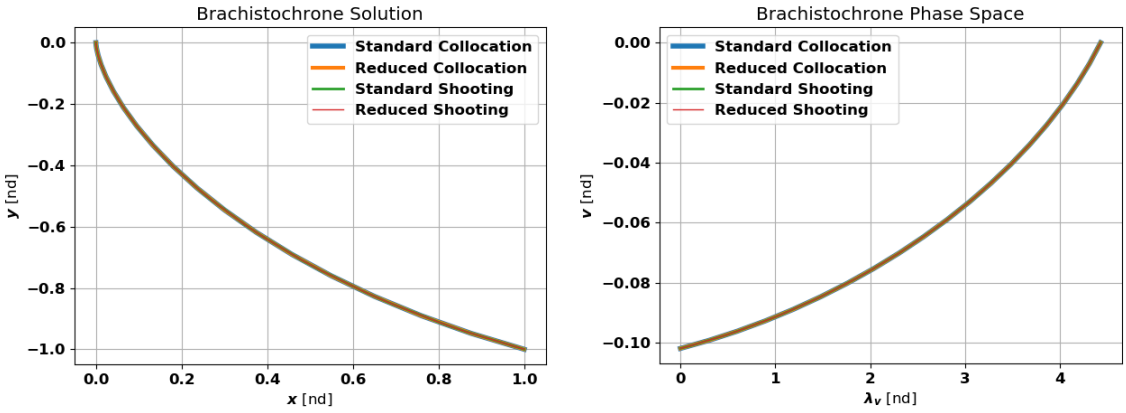


Figure 6.3. Comparison of reduced and standard solutions to the Brachistochrone problem

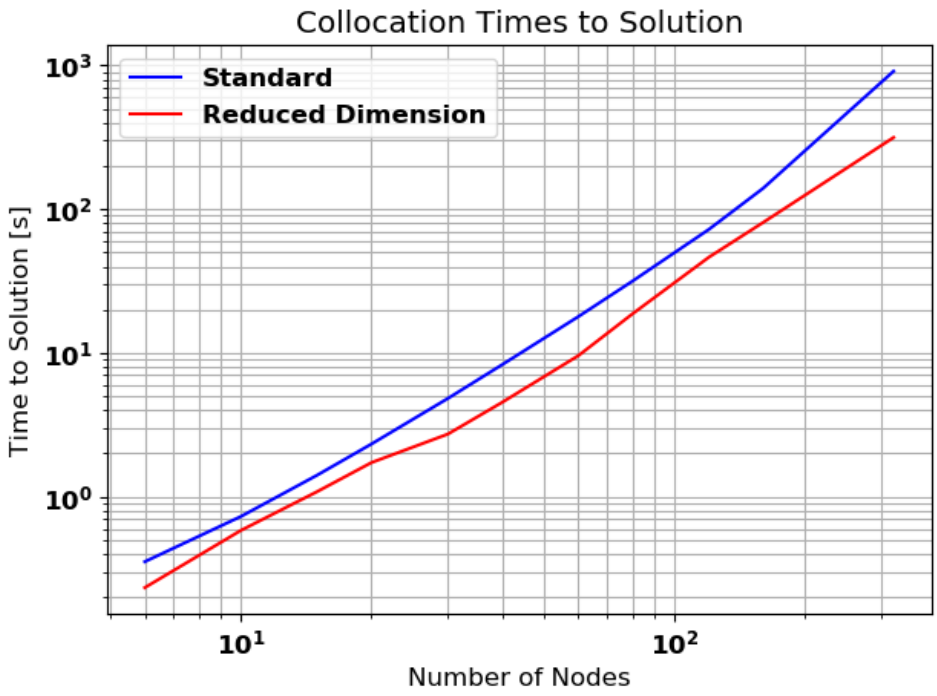


Figure 6.4. Time to solution of the Brachistochrone problem for varying number of collocation nodes

Table 6.4. Residual after each shooting update

Iteration #	Reduced System	Standard System
0	1.44656507159	0.85857864376
1	0.62306825414	0.62312567771
2	0.26542468279	0.26542701768
3	0.10337793521	0.10337150044
4	0.03889321712	0.03888822881
5	0.01444720778	0.01444450136
6	0.00534142035	0.00534014315
7	0.00197142153	0.00197086232
8	0.00072715373	0.00072692027
9	0.00026814625	0.00026805203
10	0.00009887360	0.00009883656
11	0.00003645658	0.00003644233
12	0.00001344210	0.00001343672
13	0.00000495629	0.00000495430
14	0.00000000002	0.00000000002

6.4 Summary

In this chapter, we explored the impact of reduction from the perspective of numerical methods. We analyzed two types of reduction: constant-of-motion reduction and symmetry reduction. Although modern BVP solvers can handle constant-of-motion reduced problems, we noted that modern BVP solvers are not equipped to handle symmetry reduced problems. We then created a general method of writing problem definitions for BVPs with an arbitrary number of constant-of-motion reductions and symmetry reductions. BVP solvers must use this general interface to handle the unique structure of BVPs. In addition to designing a universal interface, we con-

structed two numerical methods for solving reduced BVPs. These two solvers are a novel approach to solving BVPs and ultimately solve fewer equations using fewer operations.

Next, we used the solvers on two reduced BVPs and saw mixed results in terms of performance; however, the results confirmed the mathematical equivalence of solving reduced and unreduced problems. In the first example we achieved an analytical reduction of 50% fewer equations reducing the system from 8 to 4, plus 2 equations-of-motion on a G-bundle's fibers. In the second example we achieved an analytical reduction of 75% fewer equations, reducing the system from 48 to 12, plus 12 equations-of-motion on a G-bundle's fibers. Even though reduction did not always have a positive impact on performance, the solvers presented in this chapter are the only two solvers available that can solve reduced problems. Ultimately, this work has laid the foundation for designing newer and more efficient BVP solvers leveraging this novel structure. High performance solvers that are specifically designed to handle reduced dimensional problems will most likely improve performance significantly. This is a point of future work in Section 9.3.

7. COMPOSABLE CONSTRUCTION OF HAMILTONIAN BVPS

Indirect methods are traditionally cited as being too complicated for use in practical scenarios. Although some of this complexity is eased by the used of CASs, modern digital software for indirect methods becomes complicated as a result. An aerospace designer using such software will surely appreciate the ease of use enabled by CASs. However, a computer scientist maintaining such software still needs to deal with the complexities of indirect methods. To describe the computational implementation of mathematical processes, flowcharts are often used in combination with object-orientation [206]. While these are useful for NLP solvers and crucial for many-systems analysis, optimal control theory primarily focuses on a single system. Flowcharts and object-orientation are unnecessarily powerful for the methods described in this dissertation. In fact, they are so powerful that the additional flexibility causes more issues than they solve. The overall goal of this chapter is to delve deeper into an alternative to object-orientation and flowcharts. Their replacement is functional programming and commutative diagrams [207]. By restricting the tools we have available for implementation, we will be forced to place an emphasis on mathematical consistency. One major benefit of this is that other strategies outside of this dissertation can be pieced together in a predictable manner.

Very rarely are practical problems unconstrained single-phase OCPs resulting in TPBVPs; however, in Chapters 1 and 3 we ignored path-constraints and problems with multiple phases. This is because there already are several strategies for handling problems with these components. Over the past several years, numerous techniques have been developed by the students in the Rapid-Design-of-Systems-Laboratory (RDSL). Some of these methods are the Universal Trigonomerization

Method (UTM) [84], Relaxed Autonomously Switched Hybrid System (RASHS) [208], and the Integrated Control Regularization Method (ICRM) [74, 141]. Each of these techniques are an integral component in creating complex path-constrained multi-stage missions. In effort to tie all this work together in a single coherent indirect framework, we introduce the concept of a composable workflow. In this framework, all operations are modeled as functors on the categories of OCPs and Hamiltonian BVPs. By following a strict set of rules, these functors are guaranteed to return either OCPs or Hamiltonian BVPs. One surprising result is that, despite their near sole use in indirect methods, many of these functors can be applied equivalently to direct or indirect methods. This chapter aims to accomplish the following:

1. Recall and preserve techniques that have been previously developed.
2. Assemble various strategies that can be mixed and matched in a manner that preserves structure, and therefore aims to preserve the CMP.
3. Provide guidelines for extension with future techniques that have not yet been developed.

Within the scope of this dissertation, this chapter enables reduction for path-constrained multi-phase problems. This brings reduction up to a level suitable for general-purpose applications.

7.1 Category Theory

In order to formalize many of the state-of-the-art processes developed, we must first introduce some basic category theory. Category theory was first introduced in its modern form by Samuel Eilenberg and Saunders Mac Lane [209]. One of the significant roles that category theory has played in mathematics is in its ability to generalize widely used processes and apply them in a repeatable manner. One of the central concepts in category theory is the category; a collection of objects

and mappings. We define a category based on Eilenberg and Mac Lane's original definition.

***Definition* 38** (Category)**.** A category, $\mathscr{A}$, is a set of objects, A, and mappings, α. Assume that mappings α_1 and α_2 exist. These mappings may uniquely define a new product mapping $\alpha_2 \circ \alpha_1$. Under this assumption, the following 5 axioms of a category must hold

Axiom 1. Assume α_3 exists, then the mapping $\alpha_3 \circ (\alpha_2 \circ \alpha_1)$ exists if and only if $(\alpha_3 \circ \alpha_2) \circ \alpha_1$ exists such that $\alpha_3 \circ (\alpha_2 \circ \alpha_1) = (\alpha_3 \circ \alpha_2) \circ \alpha_1$; an associative property.

Axiom 2. Axiom 1 holds when the products $\alpha_3 \circ \alpha_2$ and $\alpha_2 \circ \alpha_1$ exist.

Axiom 3. For every mapping α_i there exists an identity Id_{i1} such that $\alpha_i \circ Id_{i1} = \alpha_i$ and there exists another identity Id_{i2} such that $Id_{i2} \circ \alpha_i = \alpha_i$.

Axiom 4. The map Id_A for each object A is an identity.

Axiom 5. For every Id_i, there is an object A such that $Id_A = Id_i$.

End definition.

Given that the objects and mappings of some category, $\mathscr{A}$, satisfy these axioms, we can represent this category visually. An example category is shown as a commutative diagram in Fig. 7.1. Future categories that are presented in this section will have identities omitted.

OCPs are defined as a large collection of objects $(B, \Sigma, T\Sigma, \mathbb{R}^\bullet)$ and mappings $(\pi, \tau_\Sigma, \boldsymbol{f}, \boldsymbol{g}, \boldsymbol{\phi}, \boldsymbol{\xi}, L, \eta)$. The physical meaning of $(B, \Sigma, T\Sigma)$ is that they describe the manifold an OCP lives in, where $\mathbb{R}^\bullet$ are the spaces of observable quantities pulled from these manifolds. The physical meaning of $(\boldsymbol{f}, \boldsymbol{g})$ is the laws of motion that a dynamical system must obey, $(\boldsymbol{\phi}, \boldsymbol{\xi})$ are mission requirements, and $(\pi_\Sigma, \tau_\Sigma)$ connect the manifolds as bundles. These objects together with their mappings form the category of OCPs, "Problem Σ", and is shown as a commutative diagram in Fig. 7.2.

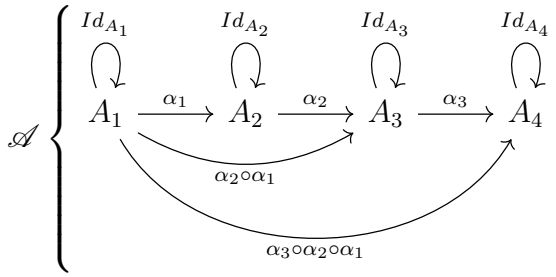


Figure 7.1. A category with 4 objects and 3 mappings.

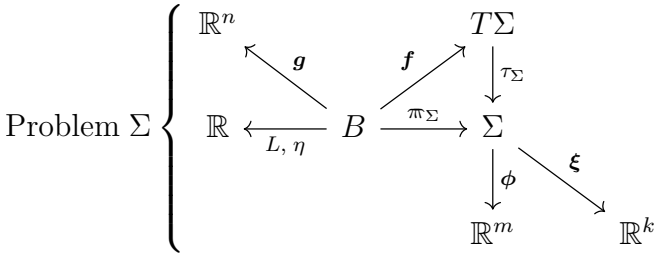


Figure 7.2. Category of OCPs.

The structure of an OCP may seem relatively simple in that it is primarily a dynamical system and a set of observables. Using category theory to represent such an object likely seems "overkill". The category of Hamiltonian BVPs is more complicated. After dualizing with $T^*\Sigma = \Lambda$, there are more manifolds, $(E, \Lambda, T\Lambda, T^*\Lambda)$, fewer observables $(H, \boldsymbol{\Phi}, \boldsymbol{\Xi})$, and more mappings $(\boldsymbol{f}, \omega, \tau_\Lambda, \pi_\Lambda, \varpi_\Lambda, \boldsymbol{u}^*)$. This setup is shown in Fig. 7.3.

While interesting in their own regard, categorizing OCPs and Hamiltonian BVPs does not yield too much additional information usable for aerospace missions. From category theory, functors are the essential tool used to map one category to another. Given two categories, $\{A_i, \alpha_i\} \in \mathscr{A}$ and $\{B_i, \beta_i\} \in \mathscr{B}$, a functor may be defined as follows

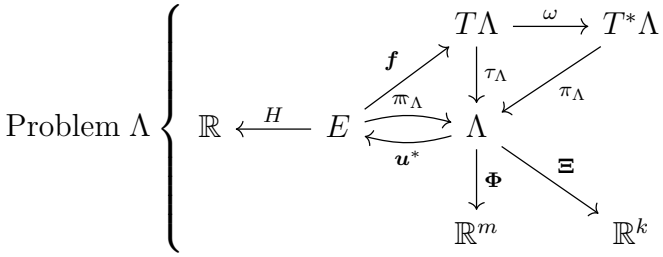


Figure 7.3. Category of Hamiltonian BVPs.

***Definition* 39** (Functor)**.** A functor, $\mathscr{F}$, is a mapping between two categories, $\mathscr{A}$ and $\mathscr{B}$. Every functor consists of the following set of data object-functions and mapping-functions

1. For every object, $A \in \mathscr{A}$, then $\mathscr{F}(A) \in \mathscr{B}$.
2. For every mapping, $\alpha \in \mathscr{A}$, then $\mathscr{F}(\alpha) \in \mathscr{B}$.

The following axioms also must hold

Axiom 1. Unitality: $\mathscr{F}(Id_A) = Id_{\mathscr{F}(A)}$.

Axiom 2. Composability: For every composable set of mappings, $\alpha_2 \circ \alpha_1$, then

$$\mathscr{F}(\alpha_2 \circ \alpha_1) = \mathscr{F}(\alpha_2) \circ \mathscr{F}(\alpha_1) \tag{7.1}$$

End definition.

Returning to the example category in Fig. 7.1, we can define a functor which takes objects and mappings in $\mathscr{A}$ and sends them to new objects and mappings also in $\mathscr{A}$.

As we saw in Chapter 4, modification of a Hamiltonian BVP must result in a new Hamiltonian BVP. By modeling our processes as functors between the categories of Problem Σ's and Problem Λ's, we can ensure that we will properly form OCPs and BVPs while simultaneously reducing the complexity of constructing such problems.

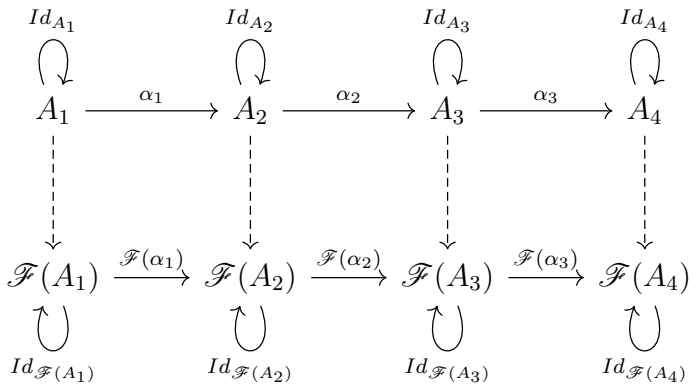


Figure 7.4. A functor on $\mathscr{A}$.

It is important to note that while the functors presented in this section provide a relatively accurate and convenient means for modeling complex OCPs in a homomorphic manner, the resulting OCPs are not necessarily equivalent to the original OCPs. A common theme in the majority of these functors is that they add a small amount of error in order to make the problem easier to solve numerically.

For example, a functor that transforms a path-constrained OCP into an unconstrained problem does not perfectly model the corners at which a path-constraint is entered and exited. Often times tolerance parameters, ϵ, are added such that as $\epsilon \to 0$ the unconstrained OCP converges to the constrained OCP. The mathematical rigor in this section does not guarantee the numerical equivalence of transforming OCPs, but only serves to guarantee structure. Considering our example, this makes an impact in that both the unconstrained OCP and the original path-constrained OCP are both OCPs of type Problem Σ. Regardless of what sequence of functors are applied, the resulting systems can always be dualized and the CMP is always preserved up to the resulting OCP from applied functors.

7.2 Optimal Control Functors

Optimal Control Functors (OCFs) are functors that transform one OCP into another OCP. In general, their functor signature is as follows

$$\mathscr{F} : \text{Problem } \Sigma \to \text{Problem } \Sigma \tag{7.2}$$

This can be done by defining each of the maps listed in Eq. (7.3).

$$\begin{aligned}
\mathscr{F} &: \text{Problem } \Sigma \to \text{Problem } \Sigma \\
\mathscr{F}(\boldsymbol{f}) &: \mathscr{F}(B) \to \mathscr{F}(T\Sigma) \\
\mathscr{F}(\boldsymbol{g}) &: \mathscr{F}(B) \to \mathscr{F}(\mathbb{R}^n) \\
\mathscr{F}(L) &: \mathscr{F}(B) \to \mathscr{F}(\mathbb{R}) \\
\mathscr{F}(\eta) &: \mathscr{F}(B) \to \mathscr{F}(\mathbb{R}) \\
\mathscr{F}(\boldsymbol{\xi}) &: \mathscr{F}(\Sigma) \to \mathscr{F}(\mathbb{R}^k) \\
\mathscr{F}(\boldsymbol{\phi}) &: \mathscr{F}(\Sigma) \to \mathscr{F}(\mathbb{R}^m) \\
\mathscr{F}(\tau_\Sigma) &: \mathscr{F}(T\Sigma) \to \mathscr{F}(\Sigma) \\
\mathscr{F}(\pi_\Sigma) &: \mathscr{F}(B) \to \mathscr{F}(\Sigma) \\
\text{Such that:} \quad \mathscr{F}(\tau_\Sigma \circ \boldsymbol{f}) &= \mathscr{F}(\pi_\Sigma)
\end{aligned} \tag{7.3}$$

Although there are many maps to define, this process ensures that when an OCF is used to construct an OCP, the results are predictable and repeatable. OCFs are always applied prior to dualization. Because of this order of operations, the CMP is preserved regardless of the process. All OCFs fit nicely into Ross and Fahroo's commutative diagram as shown in Fig. 7.5.

Next, we will introduce some OCFs that have been created by various students of RDSL. It is important to note that the functors themselves are not a contribution of this dissertation since they have been developed by other students. Rather, the composable implementation is the main contribution of this work.

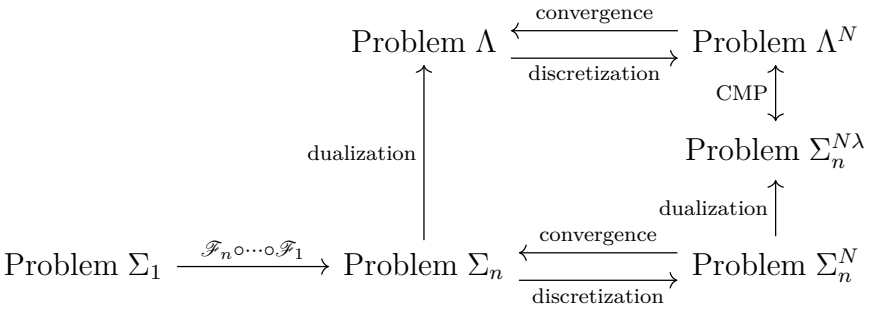


Figure 7.5. OCFs preserve the CMP by modifying OCPs prior to dualization.

7.2.1 Eps-Trig Regularization

Mall's Eps-Trig is a method of handling pure control-constraints in a smooth manner [82,83]. This method is very similar to Hull's method of replacing a bounded control with an unbounded one [210].

$$\mathscr{F}_{\text{Eps-Trig}} : \text{Problem } \Sigma \to \text{Problem } \Sigma \tag{7.4}$$

Beginning with a path-constrained OCP of the following form

$$\begin{aligned} \min_{\boldsymbol{u}} K &= \int_{t_0}^{t_f} L(t, \boldsymbol{x}, \boldsymbol{u}) \, \mathrm{d}t + \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) \\ \text{Subject to: } \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{\phi}(t_0, \boldsymbol{x}(t_0)) &= 0 \\ \boldsymbol{\xi}(t_f, \boldsymbol{x}(t_f)) &= 0 \\ \boldsymbol{u}_{\text{lower}} \leq \boldsymbol{u} &\leq \boldsymbol{u}_{\text{upper}} \end{aligned} \tag{7.5}$$

Eps-Trig transforms this OCP into another OCP in a two-step process. The first step is to substitute the control with a trigonomerized control

$$\boldsymbol{u} \equiv \frac{\boldsymbol{u}_{\text{upper}} - \boldsymbol{u}_{\text{lower}}}{2} \sin(\boldsymbol{u}_{\text{trig}}) + \frac{\boldsymbol{u}_{\text{upper}} + \boldsymbol{u}_{\text{lower}}}{2} \tag{7.6}$$

This results in a new input bundle, B^*. Using B^* as a domain, a new path-constraint function, $\boldsymbol{g}^*$, is defined with the control-constraints eliminated. The second step is to add an error control term to the cost function

$$L^* \equiv L(t, \boldsymbol{x}, \boldsymbol{u}_{\text{trig}}) - \boldsymbol{\epsilon}^T \cos(\boldsymbol{u}_{\text{trig}}) \tag{7.7}$$

where the dimension of $\boldsymbol{\epsilon}$ is equal to the dimension of $\boldsymbol{g}$'s co-domain. Therefore we have defined the following map

$$\begin{aligned}
\mathscr{F}_{\text{Eps-Trig}} &: \text{Problem } \Sigma \to \text{Problem } \Sigma \\
\mathscr{F}_{\text{Eps-Trig}}(B) &= B^* \\
\boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}) &\mapsto \boldsymbol{g}^*(t, \boldsymbol{x}, \boldsymbol{u}_{\text{trig}}) \\
L(t, \boldsymbol{x}, \boldsymbol{u}) &\mapsto L^*(t, \boldsymbol{x}, \boldsymbol{u}_{\text{trig}}, \boldsymbol{\epsilon}) \\
\eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) &\mapsto \eta(\boldsymbol{x}(t_f), \boldsymbol{u}_{\text{trig}}(t_f)) \\
\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) &\mapsto \boldsymbol{f}^*(t, \boldsymbol{x}, \boldsymbol{u}_{\text{trig}})
\end{aligned} \tag{7.8}$$

The unlisted maps are identities. Note that, despite the chosen values for $\boldsymbol{\epsilon}$ and the accuracy of the overall method, Eps-Trig may be freely applied to any optimal control problem provided that it fits within the restrictions of Eq. (7.5). If every path-constraint is a control path-constraint, then the path-constraint functions are mapped in the following manner

$$\mathscr{F}_{\text{Eps-Trig}} : (B \to \mathbb{R}^n) \to (B^* \to \emptyset) \tag{7.9}$$

Rather, the output of the Eps-Trig OCF is an unconstrained OCP. This results in a single-phase (up to discretization) system from a direct method, or TPBVP from an indirect method.

7.2.2 Universal Trigonomerization Method

With his Eps-Trig method, Mall was able to impose pure control path-constraints on OCPs, potentially simplifying what would have been a MPBVP down to a TPBVP for simple problems and eliminating some path-constraints for more complicated problems. Two drawbacks of Eps-Trig are as follows

1. General path-constraints containing state terms often times result in unsolvable transcendental control laws.

2. Eps-Trig involves replacing terms inside of the equations-of-motion. A user may enter parameters in a manner such that substitution does not properly occur.

As an improvement to his "trigonomerization" suite of tools, Mall introduced the Universal Trigonomerization Method (UTM) [84]. In UTM, path-constraints involving mixed state and control terms can be handled. This approach resolves the first issue from Eps-Trig. Additionally, UTM can be implemented entirely by adding additional terms to the path cost, which resolves the second issue from Eps-Trig. UTM has the following signature

$$\mathscr{F}_{\text{UTM}} : \Sigma \to \Sigma \tag{7.10}$$

To use UTM, we first begin with an OCP of the following form

$$\begin{aligned} \min_{\boldsymbol{u}} K &= \int_{t_0}^{t_f} L(t, \boldsymbol{x}, \boldsymbol{u}) \, \mathbb{d}t + \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f)) \\ \text{Subject to: } \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{\phi}(t_0, \boldsymbol{x}(t_0)) &= 0 \\ \boldsymbol{\xi}(t_f, \boldsymbol{x}(t_f)) &= 0 \\ \boldsymbol{g}_{\text{lower}} \leq \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) &\leq \boldsymbol{g}_{\text{upper}} \end{aligned} \tag{7.11}$$

In this OCP, there are path-constraint functions, $\boldsymbol{g}$, which are functions of mixed state and control terms, $(\boldsymbol{x}, \boldsymbol{u})$

$$\boldsymbol{g} : B \to \mathbb{R}^n \tag{7.12}$$

To account for these path-constraint functions, we first create the following penalty term

$$L_{\text{UTM}} = \boldsymbol{\epsilon}^T \left(\frac{1}{\cos\left(\frac{\pi}{2} \frac{2\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) - \boldsymbol{g}_{\text{upper}} - \boldsymbol{g}_{\text{lower}}}{\boldsymbol{g}_{\text{upper}} - \boldsymbol{g}_{\text{lower}}} \right)} - 1 \right) \tag{7.13}$$

where the dimension of $\boldsymbol{\epsilon}$ is equal to the dimension of $\boldsymbol{g}$'s co-domain. In this system, L_{UTM} is acting similar to an interior penalty method from parameter optimization. L_{UTM} is differentiable and infinite in value when the original constraint is active. Augmenting the Lagrangian, we have

$$L^* \equiv L + L_{\text{UTM}} \tag{7.14}$$

Then, we have the following OCF with no explicit path-constraints

$$\begin{aligned} \mathscr{F}_{\text{UTM}} : \text{Problem } \Sigma &\to \text{Problem } \Sigma \\ \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}) &\mapsto \boldsymbol{g}^*(t, \boldsymbol{x}, \boldsymbol{u}) \\ L(t, \boldsymbol{x}, \boldsymbol{u}) &\mapsto L^*(t, \boldsymbol{x}, \boldsymbol{u}, \epsilon) \end{aligned} \tag{7.15}$$

where $\boldsymbol{g}^*$ are the path-constraints that have not been treated with UTM. The unlisted mappings are identities. In the case where UTM is applied to every path-constraint, we have

$$\mathscr{F}_{\text{UTM}} : (B \to \mathbb{R}^n) \to (B^* \to \emptyset) \tag{7.16}$$

where $B^* = \mathscr{F}_{\text{UTM}}(B) \cong B$. That is, the UTM OCF converts a constrained OCP into an unconstrained OCP with diffeomorphic input bundles.

7.2.3 Relaxed Autonomously Switched Hybrid System

Saranathan's Relaxed Autonomously Switched Hybrid System (RASHS) is a method of incorporating discontinuous functions in an OCP by approximating them with smooth exponential functions [208, 211]. These discontinuous functions can be used to model discrete events that alter the dynamics of a system. For instance, the staging event of an ascent vehicle permanently modifies its mass and thrust profile. RASHS may be used to accurately model this staging event while retaining a simplified TP-BVP, instead of forming a more complicated MPBVP. Given an OCP of the following form

$$\begin{aligned} \min_{\boldsymbol{u}} K &= \int_{t_0}^{t_f} L(t, \boldsymbol{x}, \boldsymbol{u}, h(t)) \, \mathrm{d}t + \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), h(t_f)) \\ \text{Subject to: } \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, h(t)) \\ \boldsymbol{\phi}(t_0, \boldsymbol{x}(t_0), h(t_0)) &= 0 \\ \boldsymbol{\xi}(t_f, \boldsymbol{x}(t_f), h(t_f)) &= 0 \\ \boldsymbol{g}_{\text{lower}} \leq \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}, h(t)) &\leq \boldsymbol{g}_{\text{upper}} \end{aligned} \tag{7.17}$$

Assume $h(t)$ is a function of the form

$$h(t) = \begin{cases} a \text{ if } t < 0.5 \\ b \text{ otherwise} \end{cases} \tag{7.18}$$

Since $h(t)$ is discontinuous, traditional optimal control theory would have us generate a 3-point BVP with the middle point at $t = 0.5$ where the discontinuity occurs. Instead, Harish's RASHS method replaces the function $h(t)$ with the following smooth approximation

$$h^*(t, \epsilon) = \frac{a}{\exp\left(\frac{1}{\epsilon}(t - 0.5)\right) + 1} + \frac{b}{\exp\left(\frac{1}{\epsilon}(0.5 - t)\right) + 1} \tag{7.19}$$

As the arbitrary tolerance ϵ tends towards 0, $h^*(t)$ converges to $h(t)$. Harish's RASHS OCF is of the following form

$$\begin{aligned} \mathscr{F}_{\text{RASHS}} : \text{Problem } \Sigma &\to \text{Problem } \Sigma \\ \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, h(t)) &\mapsto \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, h^*(t, \epsilon)) \\ \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}, h(t)) &\mapsto \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}, h^*(t, \epsilon)) \\ L(t, \boldsymbol{x}, \boldsymbol{u}, h(t)) &\mapsto L(t, \boldsymbol{x}, \boldsymbol{u}, h^*(t, \epsilon)) \\ \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), h(t_f)) &\mapsto \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), h^*(t_f, \epsilon)) \\ \boldsymbol{\phi}(t_0, \boldsymbol{x}(t_0), h(t_0)) &\mapsto \boldsymbol{\phi}(t_0, \boldsymbol{x}(t_0), h^*(t_0, \epsilon)) \\ \boldsymbol{\xi}(t_f, \boldsymbol{x}(t_f), h(t_f)) &\mapsto \boldsymbol{\xi}(t_f, \boldsymbol{x}(t_f), h^*(t_f, \epsilon)) \end{aligned} \tag{7.20}$$

Applying RASHS to a constrained OCP yields a new constrained OCP.

7.2.4 Time-Shift

Especially in time-dependent OCPs, it is often easier to work with a dynamical system whose independent variable is a state. Take for instance the time-dependent OCP

$$\begin{aligned} \min_{\boldsymbol{u}} K &= \int_{t_0}^{t_f} L(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \, \mathrm{d}t + \eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), \boldsymbol{p}) \\ \text{Subject to: } \dot{\boldsymbol{x}} &= \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \\ \boldsymbol{\phi}(t_0, \boldsymbol{x}(t_0), \boldsymbol{p}) &= 0 \\ \boldsymbol{\xi}(t_f, \boldsymbol{x}(t_f), \boldsymbol{p}) &= 0 \\ \boldsymbol{g}_{\text{lower}} \leq \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) &\leq \boldsymbol{g}_{\text{upper}} \end{aligned} \tag{7.21}$$

Many modern BVP solvers cannot solve systems where $L = L(t)$ or $\boldsymbol{f} = \boldsymbol{f}(t)$. A known trick to get around this limitation is to make time a state with its own equation-of-motion. To do this, we set $\tau \equiv t$ and extend the manifold with the time coordinate

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix} \qquad \boldsymbol{f}' = \begin{bmatrix} \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \\ 1 \end{bmatrix} \tag{7.22}$$

Then the OCP becomes

$$\begin{aligned} \min_{\boldsymbol{u}} K &= \int_{\tau_0}^{\tau_f} L(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{p}) \, \mathrm{d}t + \eta(\boldsymbol{x}'(\tau_f), \boldsymbol{u}(\tau_f), \boldsymbol{p}) \\ \text{Subject to: } \dot{\boldsymbol{x}}' &= \boldsymbol{f}'(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{p}) \\ \boldsymbol{\phi}(\boldsymbol{x}'(\tau_0), \boldsymbol{p}) &= 0 \\ \boldsymbol{\xi}(\boldsymbol{x}'(\tau_f), \boldsymbol{p}) &= 0 \\ \boldsymbol{g}_{\text{lower}} \leq \boldsymbol{g}(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{p}) &\leq \boldsymbol{g}_{\text{upper}} \end{aligned} \tag{7.23}$$

Therefore, the time-shift functor is as follows

$$\begin{aligned}
\mathscr{F}_{\text{Time-Shift}} : \text{Problem } \Sigma &\to \text{Problem } \Sigma \\
\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) &\mapsto \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{p}) \\
\boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) &\mapsto \boldsymbol{g}(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{p}) \\
L(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) &\mapsto L(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{p}) \\
\eta(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), \boldsymbol{p}) &\mapsto \eta(\boldsymbol{x}'(\tau_f), \boldsymbol{u}(t_f), \boldsymbol{p}) \\
\boldsymbol{\phi}(t_0, \boldsymbol{x}(t_0), \boldsymbol{p}) &\mapsto \boldsymbol{\phi}(\boldsymbol{x}'(\tau_0), \boldsymbol{p}) \\
\boldsymbol{\xi}(t_f, \boldsymbol{x}(t_f), \boldsymbol{p}) &\mapsto \boldsymbol{\xi}(\boldsymbol{x}'(\tau_f), \boldsymbol{p})
\end{aligned} \tag{7.24}$$

This strategy was used in Chapter 4 prior to dualization. Though a similar time shift may be done on the Hamiltonian BVP, we give a reason for placing this functor in the OCF category in Section 7.7.

7.3 Dualizing Methods

In this section, we explore the two different tools available for dualizing an OCP. Functors in this section have the following signature

$$\mathscr{D} : \text{Problem } \Sigma \to \text{Problem } \Lambda \tag{7.25}$$

7.3.1 Traditional Methods

In Section 1.1.2, we provided a detailed introduction to indirect methods in their so-called "traditional" form. The Bryson-Ho functor $\mathscr{D}_{\text{BH}}$ turns an OCP into a well-defined Hamiltonian BVP [10]. The BVP generated by $\mathscr{D}_{\text{BH}}$ is symplectic; however, the operations that follow are severely restricted. This is because the map $\omega \circ \boldsymbol{f}$ is entirely stored in $\boldsymbol{f}$. There is no explicit ω generated by $\mathscr{D}_{\text{BH}}$. This is the primary shortcoming of the Bryson-Ho functor and was the motivating factor behind the work in Chapter 3.

7.3.2 Differential Geometric Methods

In Chapter 3, we redefined indirect methods in terms of differential geometric tools. The process of dualizing an OCP using differential geometric tools is represented by $\mathscr{D}_{\mathrm{DG}}$. Although there was no immediate payoff, we later saw in Chapters 4 and 5 how the additional exposed data from $\mathscr{D}_{\mathrm{DG}}$ could be leveraged.

7.4 Symplectomorphic Functors

In Section 7.2 we explored processes that can be, for the most part, freely applied to OCPs. This freedom is due to the fact that dualization occurs after all OCFs have been applied. After dualization occurs, our problem definition becomes significantly more restrictive. In a Hamiltonian BVP, there is no cost function. The cost information has been embedded in the Hamiltonian function as well as the co-state equations-of-motion. As a result, modifications of the Hamiltonian function effectively act as modifications of the cost function from the original OCP thereby changing the problem definition. Similarly, modifications of the co-state equations-of-motion also act as modifications of the cost function from the OCP. In this section, we explore functors that transform one Hamiltonian BVP into another. We call these Symplectomorphic Functors (SFs). First, we define a symplectomorphism

***Definition* 40** (Symplectomorphism)**.** Assume two symplectic manifolds, (Λ_A, ω_A) and (Λ_B, ω_B), and that a diffeomorphism between these two manifolds, $\mathscr{F} : \Lambda_A \to \lambda_B$, exists. A diffeomorphism is a symplectomorphism if it preserves the symplectic structure across the mapping such that:

$$\mathscr{F}^* \omega_B = \omega_A \tag{7.26}$$

End definition.

Typically, symplectomorphisms are studied in context of canonical transformations, Hamiltonian flows, and other Liouville theorem-preserving integrations. Our

application is slightly broader in that the mapping need not be an integration of phase-space.

***Definition* 41.** (Symplectomorphic Functor) A Symplectomorphic Functor (SF), $\mathscr{F}$, is a functor on the category of Hamiltonian BVPs that preserves the symplectic structure of the underlying dynamical system.

$$\begin{aligned} \text{Problem } \Lambda_2 &= \mathscr{F}(\text{Problem } \Lambda_1) \\ \text{Such that: } \omega_2 &= \mathscr{F}(\omega_1) \end{aligned} \tag{7.27}$$

End definition.

An SF may be created by defining the following maps

$$\begin{aligned} \mathscr{F} &: \text{Problem } \Lambda \to \text{Problem } \Lambda \\ \mathscr{F}(\boldsymbol{f}) &: \mathscr{F}(E) \to \mathscr{F}(T\Lambda) \\ \mathscr{F}(\omega) &: \mathscr{F}(T\Lambda) \to \mathscr{F}(T^*\Lambda) \\ \mathscr{F}(\boldsymbol{u}^*) &: \mathscr{F}(\Lambda) \to \mathscr{F}(E) \\ \mathscr{F}(H) &: \mathscr{F}(E) \to \mathscr{F}(\mathbb{R}) \\ \mathscr{F}(\boldsymbol{\Xi}) &: \mathscr{F}(\Lambda) \to \mathscr{F}(\mathbb{R}^k) \\ \mathscr{F}(\boldsymbol{\Phi}) &: \mathscr{F}(\Lambda) \to \mathscr{F}(\mathbb{R}^m) \\ \mathscr{F}(\tau_\Lambda) &: \mathscr{F}(T\Lambda) \to \mathscr{F}(\Lambda) \\ \mathscr{F}(\pi_\Lambda) &: \mathscr{F}(T^*\Lambda) \to \mathscr{F}(\Lambda) \\ \mathscr{F}(\varpi_\Lambda) &: \mathscr{F}(E) \to \mathscr{F}(\Lambda) \\ \text{Such that: } \quad \mathscr{F}(\tau_\Lambda \circ \boldsymbol{f}) &= \mathscr{F}(\varpi_\Lambda) \\ \mathscr{F}(\pi_\Lambda \circ \omega) &= \mathscr{F}(\tau_\Lambda) \end{aligned} \tag{7.28}$$

While daunting, ensuring every transformation of a Hamiltonian BVP is also a symplectomorphism guarantees that structure is carried throughout all mathematical processes. While not guaranteed to exist in a general sense, by preserving structure, we are not explicitly destroying prospects of a CMP. Where SFs fit in to the indirect

optimization process is shown in Fig. 7.6. Note that, unlike OCFs, SFs occur after dualization. Therefore, for an SF to preserve the CMP, one would expect there to be a corresponding OCF. We call SFs that have corresponding OCFs hybrid functors and these are treated in Section 7.5.

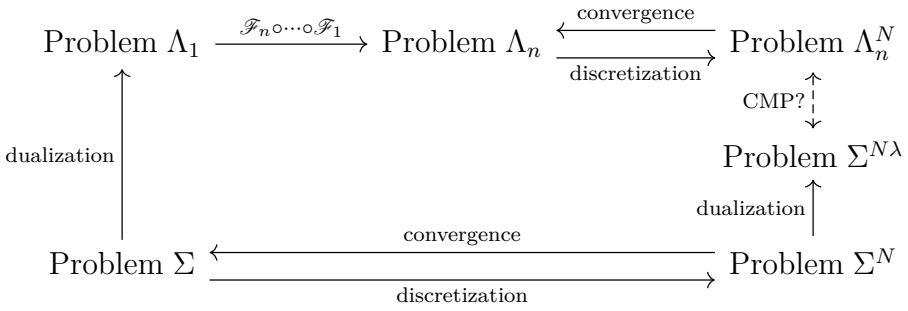


Figure 7.6. Symplectomorphic functors may preserve the CMP.

7.4.1 Symplectic ICRM

As initially introduced by Antony et al. [74, 141] as an application of index reduction [212, 213], ICRM does not fit the criteria for an OCF, nor does it fit the criteria for an SF. An extension was made by Sparapany et al. to symplectic systems [214] which was presented in detail in Chapter 4. The signature of ICRM is as follows

$$\mathscr{F}_{\text{ICRM}} : \text{Problem } \Lambda \to \text{Problem } \Lambda \tag{7.29}$$

There are two significant steps in the symplectic ICRM process: extension of the base manifold and augmentation of the symplectic form. These operations fit into the functor in Eq. (7.30).

$$\begin{aligned}
\mathscr{F}_{\text{ICRM}} : \text{Problem } \Lambda &\to \text{Problem } \Lambda \\
\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) &\mapsto \boldsymbol{f}(t, \boldsymbol{x}') \\
\mathscr{F}_{\text{ICRM}}(\Lambda) &= \Lambda' \\
\mathscr{F}_{\text{ICRM}}(T\Lambda) &= T\Lambda' \\
\mathscr{F}_{\text{ICRM}}(T^*\Lambda) &= T^*\Lambda' \\
\mathscr{F}_{\text{ICRM}}(E) &= E' \\
\omega &\mapsto \omega'
\end{aligned} \tag{7.30}$$

7.5 Hybrid Methods

In this section, we explore a special set of methods that can be applied to both an OCP or the resulting Hamiltonian BVP after dualization. Functors in this section have the following signature

$$\mathscr{F} : \begin{cases} \Sigma & \to \Sigma \\ \Lambda & \to \Lambda \end{cases} \tag{7.31}$$

These functors are OCFs in the sense that they will turn an OCP into another OCP, while they are also SFs in that their transformations are symplectomorphic.

7.5.1 Reduction

In Chapter 5, we explored reduction on symplectic manifolds in detail. Then, in Chapter 6, we explored how to construct BVP solvers that have the ability to solve problems defined on the reduced dimensional spaces. Though not explored explicitly in Chapter 5, the BVP solvers from Chapter 6 were specifically contructed to handle problems that were not "fully reduced". Rather, reduction is focused on finding a manifold $\Lambda//G$; however, the BVP solvers were designed to handle problems with

structure Λ/G or $\boldsymbol{J}^{-1}$. Since $\Lambda//G \cong \boldsymbol{J}^{-1}/G$, the BVP solvers can handle more general cases than were presented. This alludes to the idea that a symmetry may be eliminated from a system without removing its associated constant-of-motion. Similarly, a constant-of-motion may be eliminated without removal of its associated symmetry. While Noether's theorem guarantees the correspondence of symmetries and constants-of-motion, this only applies on Problem Λ. In Problem Σ, a differential structure is not guaranteed to exist. Therefore, reduction may be performed prior to dualization of Problem Σ by reducing only the symmetries and constants-of-motion that exist. Once dualized, the constants-of-motion that have been eliminated do not yield symmetries and the symmetries that have been eliminated do not yield constants-of-motion. This setup is shown in Fig. 7.7.

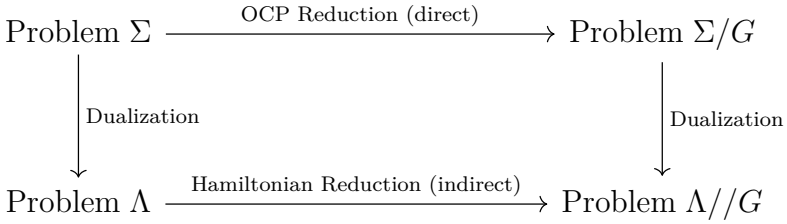


Figure 7.7. Commutative diagram for the dualization and reduction of an OCP possessing symmetries.

Ergo, a reduction functor designed to handle symmetries and constants-of-motion independently will commute with the dualization of an OCP provided it is dualized using the differential geometric formulation presented in Chapter 3.

$$\mathscr{D}_{\mathrm{DG}} \circ \mathscr{F}_{\mathrm{Reduction}} = \mathscr{F}_{\mathrm{Reduction}} \circ \mathscr{D}_{\mathrm{DG}} \tag{7.32}$$

This is a result we already saw in Chapter 5. We solved two examples: the direct and indirect Brachistochrone problems. Although we previously treated these as two independent problems, they are mathematically intertwined by Eq. (7.32). In other words, if a Co-vector Mapping Theorem (CMT) is an application specific

incarnation of the CMP, then Eq. (7.32) guarantees there exists a CMT between those two problems.

7.6 Examples

We will now give a few examples of how functors may be used in the construction of numerically solvable Hamiltonian BVPs. These examples will be carried out step-by-step in detail. Since the functorial process requires full duplication of all objects, the results of every step are either a well-defined OCP or a well-defined Hamiltonian BVP. This means any example can be cut short and solved numerically using either the solvers from Chapter 6 or any standard direct solver in the case of an OCP. These examples will take up a large amount of space; however, we will also rewrite them using the new representation presented in this chapter. The result from this new representation is a mathematically consistent one-liner describing an OCPs entire transformation process into a Hamiltonian BVP.

7.6.1 Brachistochrone

In this section, we revisit the Brachistochrone problem from Section 6.3.2. In our first encounter, recall that we posed the problem in a novel manner that allowed us to solve it faster than the current state-of-the-art indirect method. To do this, we also required the designer to pass additional information into the BVP solver about the geometric structure of the problem. Recall there are two symmetries in this problem, ∂_x and ∂_y. Instead of identifying ∂_x and ∂_y as being two independent symmetries, we identify a single symmetry

$$S = \cos\theta \frac{\partial}{\partial x} + \sin\theta \frac{\partial}{\partial y} \tag{7.33}$$

In Eq. (7.33), $\theta \in [0, 2\pi]$ is an arbitrary parameter that is varied. By defining our symmetry in this manner, it forces our algorithm to choose an adequate basis for the reduced dimensional manifold. When $\theta = n\pi$ for all integer valued n, then $q \equiv \pm x$.

However, when $\theta = \pi(n + 1/2)$ for all integer valued n, then $q \equiv \pm y$. Therefore the algorithm cannot choose to annihilate x or y on the quotient manifold for every case, but must instead adapt to the structure of the symmetry provided. In geometric terms, we can consider the configuration of our problem as a fiber bundle whose fibers are the symmetries of the system. In reduction, we are choosing a vector that rotates through these fibers. A visual aid of this situation is shown in Fig. 7.8.

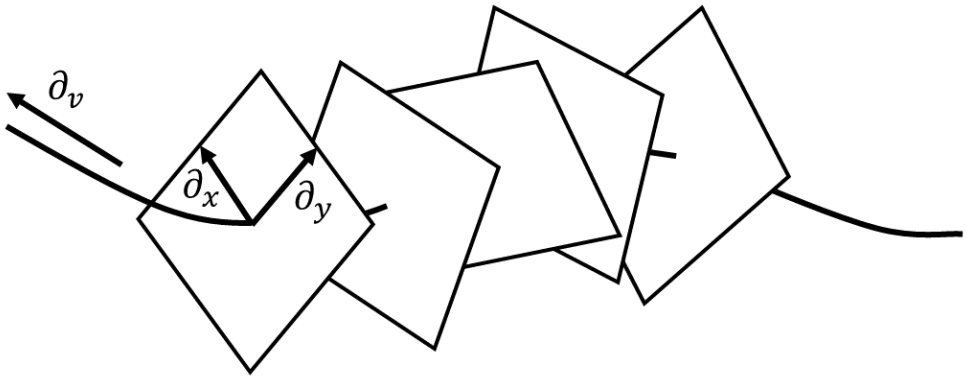


Figure 7.8. The configuration of the Brachistochrone problem is a principal G-bundle whose positions are fibered over velocity.

In constructing the Hamiltonian BVP, we use a sequential process with functors. We will go through each step manually starting with the definition of the OCP

$$\begin{aligned} \min_{\gamma} K &= \int_0^{t_f} 1 \, \mathrm{d}t \\ \text{Subject to: } \frac{\mathrm{d}x}{\mathrm{d}t} &= v\cos(\gamma) \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= v\sin(\gamma) \\ \frac{\mathrm{d}v}{\mathrm{d}t} &= g\sin(\gamma) \\ S &= \cos(\theta)\frac{\partial}{\partial x} + \sin(\gamma)\frac{\partial}{\partial y} \\ 0 &= x(t_0) = y(t_0) = v(t_0) \\ x(t_f) &= 10, y(t_f) = -10 \end{aligned} \tag{7.34}$$

First, we adjoin time as a state to the OCP

$$\begin{aligned}
\min_{\gamma} K &= \int_0^{\tau_f} 1 \, \mathrm{d}\tau \\
\text{Subject to: } \frac{\mathrm{d}x}{\mathrm{d}\tau} &= v\cos(\gamma) \\
\frac{\mathrm{d}y}{\mathrm{d}\tau} &= v\sin(\gamma) \\
\frac{\mathrm{d}v}{\mathrm{d}\tau} &= g\sin(\gamma) \\
\frac{\mathrm{d}t}{\mathrm{d}\tau} &= 1 \\
S &= \cos(\theta)\frac{\partial}{\partial x} + \sin(\gamma)\frac{\partial}{\partial y} \\
0 &= x(\tau_0) = y(\tau_0) = v(\tau_0) = t(\tau_0) \\
x(\tau_f) &= 10, y(\tau_f) = -10
\end{aligned} \tag{7.35}$$

Next, we dualize the problem using the differential geometric methods described in Chapter 3.

$$\begin{aligned}
H &= g\lambda_v \sin(\gamma) + \lambda_t + \lambda_x v\cos(\gamma) + \lambda_y v\sin(\gamma) + 1 \\
\omega &= \mathrm{d}x \wedge \mathrm{d}\lambda_x + \mathrm{d}y \wedge \mathrm{d}\lambda_y + \mathrm{d}v \wedge \mathrm{d}\lambda_v + \mathrm{d}t \wedge \mathrm{d}\lambda_t \\
S_1 &= \cos(\theta)\frac{\partial}{\partial x} + \sin(\theta)\frac{\partial}{\partial y} \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= x(\tau_0) = y(\tau_0) = v(\tau_0) = t(\tau_0) \\
0 &= \nu_1 + \lambda_x(\tau_0) = \nu_2 + \lambda_y(\tau_0) = \nu_3 + \lambda_v(\tau_0) = \nu_4 + \lambda_t(\tau_0) \\
x(\tau_f) &= 10, y(\tau_f) = -10 \\
0 &= \lambda_x(\tau_f) - \nu_5 = \lambda_y(\tau_f) - \nu_6 = \lambda_v(\tau_f) = \lambda_t(\tau_f) \\
0 &= g\lambda_v(\tau_f)\sin(\gamma) + \lambda_t(\tau_f) + \lambda_x(\tau_f)v(\tau_f)\cos(\gamma) + \lambda_y(\tau_f)v(\tau_f)\sin(\gamma) + 1
\end{aligned} \tag{7.36}$$

A part of this dualization is the usage of PMP, so we are not using the symplectic ICRM strategy presented in Chapter 4. Since the Hamiltonian is not an explicit function of time, we added a second infinitesimal symmetry, $S_2 \equiv \partial/\partial t$. After du-

alization, we scale the Hamiltonian BVP by multiplying every state by a parameter representing final time

$$
\begin{aligned}
H &= g\lambda_v \sin(\gamma) + \lambda_t + \lambda_x v\cos(\gamma) + \lambda_y v \sin(\gamma) + 1 \\
\omega &= \tau_f^{-1}\,\mathbb{d}x \wedge \mathbb{d}\lambda_x + \tau_f^{-1}\,\mathbb{d}y \wedge \mathbb{d}\lambda_y + \tau_f^{-1}\,\mathbb{d}v \wedge \mathbb{d}\lambda_v + \tau_f^{-1}\,\mathbb{d}t \wedge \mathbb{d}\lambda_t \\
S_1 &= \cos(\theta)\frac{\partial}{\partial x} + \sin(\gamma)\frac{\partial}{\partial y} \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= x(0) = y(0) = v(0) = t(0) \\
0 &= \nu_1 + \lambda_x(0) = \nu_2 + \lambda_y(0) = \nu_3 + \lambda_v(0) = \nu_4 + \lambda_t(0) \\
x(1) &= 10, y(1) = -10 \\
0 &= \lambda_x(1) - \nu_5 = \lambda_y(1) - \nu_6 = \lambda_v(1) = \lambda_t(1) \\
0 &= g\lambda_v(1)\sin(\gamma) + \lambda_t(1) + \lambda_x(1)v(1)\cos(\gamma) + \lambda_y(1)v(1)\sin(\gamma) + 1
\end{aligned}
\tag{7.37}
$$

Finally, we finish off the construction with two applications of Marsden-Weinstein-Meyer reduction as described in Chapter 5. The first application of MWM reduction

eliminates the symmetry defined in Eq. (7.33) and its associated constant-of-motion as defined by the Noether-like map introduced in Chapter 3

$$
\begin{aligned}
H &= g\lambda_v \sin(\gamma) + \lambda_t + v\left(\frac{c_1}{\cos(\theta)} - \lambda_y \tan(\theta)\right)\cos(\gamma) + 1 \\
\omega &= \tau_f^{-1}\,\mathbb{d}y \wedge \mathbb{d}\lambda_y + \tau_f^{-1}\,\mathbb{d}v \wedge \mathbb{d}\lambda_v + \tau_f^{-1}\,\mathbb{d}t \wedge \mathbb{d}\lambda_t \\
\dot{q}_1 &= \tau_f\left(-\lambda_y \sin(\gamma) - \left(\frac{c_1}{\cos(\theta)} - \lambda_y \tan(\theta)\right)\cos(\gamma)\right) \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= \frac{q_1(0)}{\cos(\theta)} - y(0)\tan(\theta) = y(0) = v(0) = t(0) \\
0 &= \frac{c_1}{\cos(\theta)} + \nu_1 - \lambda_y(0)\tan(\theta) = \nu_2 + \lambda_y = \nu_3 + \lambda_v = \nu_4 + \lambda_t \\
10 &= \frac{q_1(1)}{\cos(\theta)} - y(1)\tan(\theta), y(1) = -10 \\
0 &= \frac{c_1}{\cos(\theta)} - \nu_5 - \lambda_y(1)\tan(\theta) = \lambda_y(1) - \nu_6 = \lambda_v(1) = \lambda_t(1) \\
0 &= g\lambda_v(1)\sin(\gamma) + \lambda_t(1) + \lambda_y(1)v(1)\sin(\gamma) \\
&\quad + v(1)\left(\frac{c_1}{\cos(\theta)} - \lambda_y(1)\tan(\theta)\right)\cos(\gamma) + 1\sin(\gamma) + 1
\end{aligned}
\tag{7.38}
$$

where we introduced q_1 as the quad and $c_1 \equiv \lambda_x \cos(\theta) + \lambda_y \sin(\theta)$. In this first reduction, we introduced a singularity when $\theta = \pi(n + 1/2)$. In instances where this occurs, we eliminate the (y, λ_y) pair instead of the (x, λ_x) pair. Since this problem

does not contain any explicit dependencies on time, time is a symmetry. The second application of MWM reduction eliminates time and its adjoint state

$$
\begin{aligned}
H &= g\lambda_v \sin(\gamma) + c_2 + v\left(\frac{c_1}{\cos(\theta)} - \lambda_y \tan(\theta)\right)\cos(\gamma) + 1 \\
\omega &= \tau_f^{-1}\, \mathbb{d}y \wedge \mathbb{d}\lambda_y + \tau_f^{-1}\, \mathbb{d}v \wedge \mathbb{d}\lambda_v \\
\dot{q}_1 &= \tau_f\left(-\lambda_y \sin(\gamma) - \left(\frac{c_1}{\cos(\theta)} - \lambda_y \tan(\theta)\right)\cos(\gamma)\right) \\
\dot{q}_2 &= \tau_f \\
0 &= \frac{q_1(0)}{\cos(\theta)} - y(0)\tan(\theta) = y(0) = v(0) = q_2(0) \\
0 &= \frac{c_1}{\cos(\theta)} + \nu_1 - \lambda_y(0)\tan(\theta) = \nu_2 + \lambda_y = \nu_3 + \lambda_v = \nu_4 + c_2 \\
10 &= \frac{q_1(1)}{\cos(\theta)} - y(1)\tan(\theta), y(1) = -10 \\
0 &= \frac{c_1}{\cos(\theta)} - \nu_5 - \lambda_y(1)\tan(\theta) = \lambda_y(1) - \nu_6 = \lambda_v(1) = c_2 \\
0 &= g\lambda_v(1)\sin(\gamma) + c_2 + \lambda_y(1)v(1)\sin(\gamma) \\
&\quad + v(1)\left(\frac{c_1}{\cos(\theta)} - \lambda_y(1)\tan(\theta)\right)\cos(\gamma) + 1\sin(\gamma) + 1
\end{aligned}
\tag{7.39}
$$

The Hamiltonian BVP in Eq. (7.39) is our final BVP. The Brachistochrone problem is one of the simplest OCPs yet the full procedure was lengthy and error-prone if done by hand. Instead, we can leverage the functorial language introduced in this chapter. The following equation is equivalent to the entire procedure

$$
\text{Problem } \Lambda = (\mathscr{F}_{\text{MWM}} \circ \mathscr{F}_{\text{MWM}} \circ \mathscr{F}_{\text{scale time}} \circ \mathscr{D}_{\text{DG}} \circ \mathscr{F}_{\text{time shift}})(\text{Problem } \Sigma) \tag{7.40}
$$

While seemingly complicated, this entire process was automated using `beluga` v0.3.3 [140]. To numerically solve the resulting Problem Λ, we use the BVP solvers introduced in Chapter 6. We solved 64 different problems, each problem using a different value of θ between 0 and 2π. In Fig. 7.9, we plot the numerical values for q on the quotient problem as θ is rotated. The curves that are more red in color are closely aligned with ∂_x, while the curves that are more blue in color are closely aligned with ∂_y. Curves turn a purple color as the symmetry passes from ∂_x to ∂_y and back again.

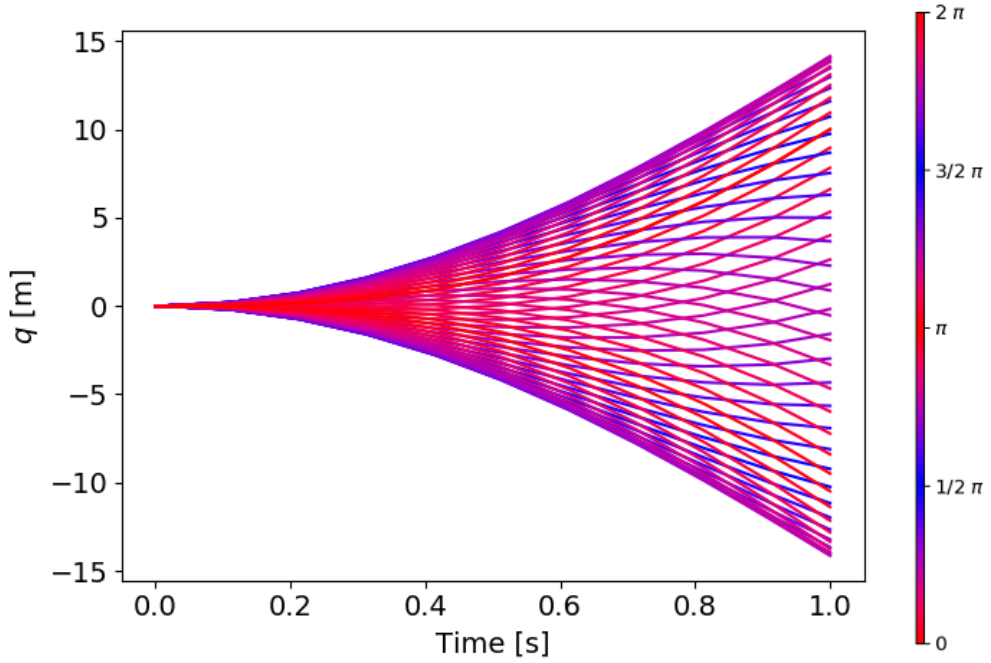


Figure 7.9. Solutions on the quotient manifold of the Brachistochrone problem. Every curve is a unique solution to the same OCP as the infinitesimal symmetry rotates through the configuration's fibers.

For this OCP, we chose the initial point $(x_0, y_0) = (0, 0)$ and terminal point $(x_f, y_f) = (10, -10)$. However, we clearly see from Fig. 7.9 that the maximum values of q are close to ± 15. To investigate this further, we decompose q into two separate plots. The first plot is for values of $-\pi/4 < \theta < \pi/4$ and $3\pi/4 < \theta < 5\pi/4$ representing infinitesimal symmetries more aligned with ∂_x. The rest are in the second plot representing infinitesimal symmetries more aligned with ∂_y. These two plots are in Fig. 7.10. From these two plots, we can see that when the curves are the most red or blue, they are at $\theta = n\pi/2$ (for integer n) and terminate at values of ± 10. This is consistent with the original terminal condition. However, the purple curves associated with $\pm\pi 1/4$ and $\pm 3\pi/4$ have terminal values of 0 and near 15. Along the $-\pi/4$ and $3\pi/4$ axis, the terminal point of the quad is 0 since the terminal point $(10, -10)$ is located along that axis. Along the $\pi/4$ and $-3\pi/4$ axis, the terminal

point of the quad is $\sqrt{10^2} \approx 14.14$. This makes sense since the terminal point is located at $(10, -10)$ and when the symmetry axis is perpendicular to the origin and the terminal point, the quad's terminal value is $\sqrt{10^2 + 10^2}$.

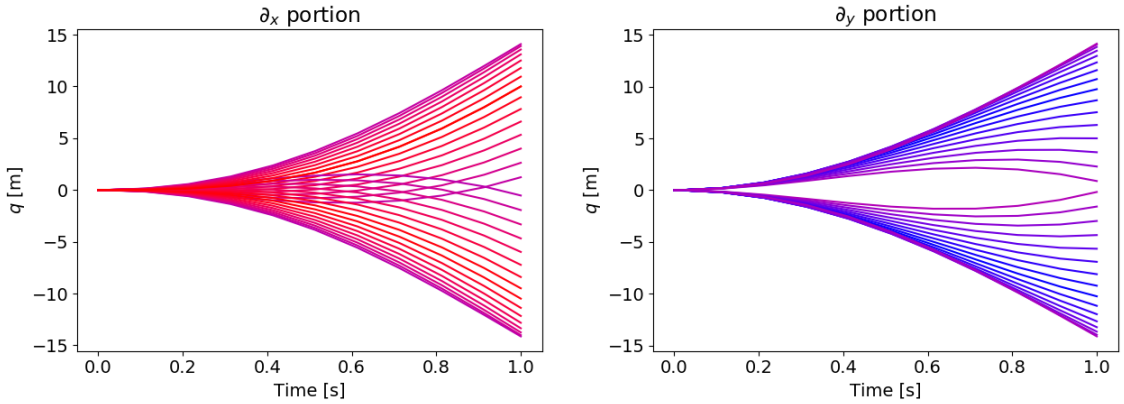


Figure 7.10. Quads of the Brachistochrone solution decomposed into ∂_x and ∂_y components.

7.6.2 Orbit Raising

In this example, we explore a spacecraft in a circular orbit about a central body. The spacecraft's mission is to raise its orbit to as high of a circular orbit as possible

in a fixed time-frame. The equations-of-motion that describe the spacecraft's motion along with its cost functional are as follows

$$
\begin{aligned}
\min_{\alpha} K &= -r^2(t_f) \\
\text{Subject to: } \frac{\mathrm{d}r}{\mathrm{d}t} &= v_r \\
\frac{\mathrm{d}\theta}{\mathrm{d}t} &= \frac{v_\theta}{r} \\
\frac{\mathrm{d}v_r}{\mathrm{d}t} &= \frac{v_\theta^2}{r} - \frac{\mu}{r^2} + \frac{T\cos\alpha}{m} \\
\frac{\mathrm{d}v_\theta}{\mathrm{d}t} &= -\frac{v_r v_\theta}{r} + \frac{T\sin\alpha}{m} \\
\frac{\mathrm{d}m}{\mathrm{d}t} &= m_{\text{rate}} \\
S &= \frac{\partial}{\partial\theta} \\
0 &= r(t_0) - r_0 = \theta(t_0) - \theta_0 = v_r(t_0) - v_{r_0} = v_\theta(t_0) - v_{\theta_0} \\
0 &= m(t_0) - m_0 = v_r(t_f) - v_{r_f} = v_\theta(t_f) - \sqrt{\frac{\mu}{r(t_f)}}
\end{aligned}
\tag{7.41}
$$

Solutions to this problem generally fall under two categories: low thrust and high thrust cases. Low thrust cases appear quite frequently in literature due to the interest in solar sails, ion drives, and other extremely high specific impulse propulsion. High thrust cases are typically the result low specific impulse propulsion. Numerically, the low thrust techniques are generally regarded to be significantly more difficult to solve due to sensitivity of propagating the differential equations. We were able to create a

Hamiltonian BVP out of this problem using a sequential process with functors. Our first functor applied was the time shift functor. This appends time as a state.

$$
\begin{aligned}
\min_{\alpha} K &= -r^2(\tau_f) \\
\text{Subject to: } \frac{\mathrm{d}r}{\mathrm{d}\tau} &= v_r \\
\frac{\mathrm{d}\theta}{\mathrm{d}\tau} &= \frac{v_\theta}{r} \\
\frac{\mathrm{d}v_r}{\mathrm{d}\tau} &= \frac{v_\theta^2}{r} - \frac{\mu}{r^2} + \frac{T\cos\alpha}{m} \\
\frac{\mathrm{d}v_\theta}{\mathrm{d}\tau} &= -\frac{v_r v_\theta}{r} + \frac{T\sin\alpha}{m} \\
\frac{\mathrm{d}m}{\mathrm{d}\tau} &= m_{\text{rate}} \\
\frac{\mathrm{d}t}{\mathrm{d}\tau} &= 1 \\
S &= \frac{\partial}{\partial\theta} \\
0 &= r(\tau_0) - r_0 = \theta(\tau_0) - \theta_0 = v_r(\tau_0) - v_{\tau_0} = v_\theta(\tau_0) - v_{\theta_0} \\
0 &= m(\tau_0) - m_0 = t(\tau_0) = v_r(\tau_f) - v_{r_f} = v_\theta(\tau_f) - \sqrt{\frac{\mu}{r(\tau_f)}}
\end{aligned}
\tag{7.42}
$$

With time appended as a state, we can now dualize the problem. To do this, we create the symplectic system as described in Chapter 3; however, we will use PMP

instead of Symplectic ICRM. This is because the problem can be solved relatively easily using PMP and therefore Symplectic ICRM is not needed.

$$
\begin{aligned}
H &= \lambda_m m_{\text{rate}} + \lambda_r v_r + \lambda_t + \lambda_\theta \frac{v_\theta}{r} + \lambda_{v_r}\left(\frac{T\cos(\alpha)}{m} - \frac{\mu}{r^2} + \frac{v_\theta^2}{r}\right) + \lambda_{v_\theta}\left(\frac{T\sin(\alpha)}{m} - \frac{v_r v_\theta}{r}\right) \\
\omega &= \mathbb{d}r \wedge \mathbb{d}\lambda_r + \mathbb{d}\theta \wedge \mathbb{d}\lambda_\theta + \mathbb{d}v_r \wedge \mathbb{d}\lambda_{v_r} + \mathbb{d}v_\theta \wedge \mathbb{d}\lambda_{v_\theta} + \mathbb{d}t \wedge \mathbb{d}\lambda_t \\
S_1 &= \frac{\partial}{\partial\theta} \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= r(\tau_0) - r_0 = \theta(\tau_0) - \theta_0 = v_r(\tau_0) - v_{\tau_0} = v_\theta(\tau_0) - v_{\theta_0} \\
0 &= \nu_1 + \lambda_r(\tau_0) = \nu_2 + \lambda_\theta(\tau_0) = \nu_3 + \lambda_{v_r}(\tau_0) = \nu_4 + \lambda_{v_\theta}(\tau_0) = \nu_5 + \lambda_m(\tau_0) = \nu_6 + \lambda_t(0) \\
0 &= m(\tau_0) - m_0 = t(\tau_0) = v_r(\tau_f) - v_{r_f} = v_\theta(\tau_f) - \sqrt{\frac{\mu}{r(\tau_f)}} \\
0 &= \lambda_r(\tau_f) + 2r(\tau_f) - \nu_7 \frac{1}{2r(\tau_f)}\sqrt{\frac{\mu}{r(\tau_f)}} = \lambda_\theta(\tau_f) = \lambda_{v_r}(\tau_f) - \nu_8 \\
0 &= \lambda_{v_\theta}(\tau_f) - \nu_9 = \lambda_m(\tau_f) = \lambda_t(\tau_f) - \nu_{10} = H(\tau_f)
\end{aligned}
\tag{7.43}
$$

In the dualization process, the infinitesimal symmetry S_2 was identified since both the cost and the states are not explicit functions of time. Next, we scale the current

Hamiltonian BVP with the final time of the problem. This introduces the final time τ_f as a dynamical parameter.

$$
\begin{aligned}
H &= \lambda_m m_{\text{rate}} + \lambda_r v_r + \lambda_t + \lambda_\theta \frac{v_\theta}{r} + \lambda_{v_r}\left(\frac{T\cos(\alpha)}{m} - \frac{\mu}{r^2} + \frac{v_\theta^2}{r}\right) + \lambda_{v_\theta}\left(\frac{T\sin(\alpha)}{m} - \frac{v_r v_\theta}{r}\right) \\
\omega &= \tau_f^{-1}\,\mathrm{d}r \wedge \mathrm{d}\lambda_r + \tau_f^{-1}\,\mathrm{d}\theta \wedge \mathrm{d}\lambda_\theta + \tau_f^{-1}\,\mathrm{d}v_r \wedge \mathrm{d}\lambda_{v_r} + \tau_f^{-1}\,\mathrm{d}v_\theta \wedge \mathrm{d}\lambda_{v_\theta} + \tau_f^{-1}\,\mathrm{d}t \wedge \mathrm{d}\lambda_t \\
S_1 &= \frac{\partial}{\partial\theta} \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= r(0) - r_0 = \theta(0) - \theta_0 = v_r(0) - v_0 = v_\theta(0) - v_0 \\
0 &= \nu_1 + \lambda_r(0) = \nu_2 + \lambda_\theta(0) = \nu_3 + \lambda_{v_r}(0) = \nu_4 + \lambda_{v_\theta}(0) = \nu_5 + \lambda_m(0) = \nu_6 + \lambda_t(0) \\
0 &= m(0) - m_0 = t(0) = v_r(f) - v_{r_f} = v_\theta(f) - \sqrt{\frac{\mu}{r(f)}} \\
0 &= \lambda_r(1) + 2r(1) - \nu_7 \frac{1}{2r(1)}\sqrt{\frac{\mu}{r(1)}} = \lambda_\theta(1) = \lambda_{v_r}(1) - \nu_8 \\
0 &= \lambda_{v_\theta}(1) - \nu_9 = \lambda_m(1) = \lambda_t(1) - \nu_{10} = H(1)
\end{aligned}
\tag{7.44}
$$

Next, we will perform two sequential applications of the reduction process as described in Chapter 5. The first reduction is of the pair (θ, λ_θ).

$$
\begin{aligned}
H &= \lambda_m m_{\text{rate}} + \lambda_r v_r + \lambda_t + c_1 \frac{v_\theta}{r} + \lambda_{v_r}\left(\frac{T\cos(\alpha)}{m} - \frac{\mu}{r^2} + \frac{v_\theta^2}{r}\right) + \lambda_{v_\theta}\left(\frac{T\sin(\alpha)}{m} - \frac{v_r v_\theta}{r}\right) \\
\omega &= \tau_f^{-1}\,\mathrm{d}r \wedge \mathrm{d}\lambda_r + \tau_f^{-1}\,\mathrm{d}v_r \wedge \mathrm{d}\lambda_{v_r} + \tau_f^{-1}\,\mathrm{d}v_\theta \wedge \mathrm{d}\lambda_{v_\theta} + \tau_f^{-1}\,\mathrm{d}t \wedge \mathrm{d}\lambda_t \\
\dot{q}_1 &= \tau_f * \frac{v_t heta}{r} \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= r(0) - r_0 = q_1(0) - \theta_0 = v_r(0) - v_0 = v_\theta(0) - v_0 \\
0 &= \nu_1 + \lambda_r(0) = \nu_2 + \lambda_\theta(0) = \nu_3 + \lambda_{v_r}(0) = \nu_4 + \lambda_{v_\theta}(0) = \nu_5 + \lambda_m(0) = \nu_6 + \lambda_t(0) \\
0 &= m(0) - m_0 = t(0) = v_r(f) - v_{r_f} = v_\theta(f) - \sqrt{\frac{\mu}{r(f)}} \\
0 &= \lambda_r(1) + 2r(1) - \nu_7 \frac{1}{2r(1)}\sqrt{\frac{\mu}{r(1)}} = c_1 = \lambda_{v_r}(1) - \nu_8 \\
0 &= \lambda_{v_\theta}(1) - \nu_9 = \lambda_m(1) = \lambda_t(1) - \nu_{10} = H(1)
\end{aligned}
\tag{7.45}
$$

By removing the pair (θ, λ_θ), we have lowered the dimension of our system from 10 to 8. Then, we will do the reduction process again using S_2 which will lower the dimension from 8 to 6.

$$
\begin{aligned}
H &= \lambda_m m_{\text{rate}} + \lambda_r v_r + c_2 + c_1 \frac{v_\theta}{r} + \lambda_{v_r}\left(\frac{T\cos(\alpha)}{m} - \frac{\mu}{r^2} + \frac{v_\theta^2}{r}\right) + \lambda_{v_\theta}\left(\frac{T\sin(\alpha)}{m} - \frac{v_r v_\theta}{r}\right) \\
\omega &= \tau_f^{-1}\,\mathrm{d}r \wedge \mathrm{d}\lambda_r + \tau_f^{-1}\,\mathrm{d}v_r \wedge \mathrm{d}\lambda_{v_r} + \tau_f^{-1}\,\mathrm{d}v_\theta \wedge \mathrm{d}\lambda_{v_\theta} \\
\dot{q}_1 &= \tau_f * \frac{v_t heta}{r} \\
\dot{q}_2 &= \tau_f \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= r(0) - r_0 = q_1(0) - \theta_0 = v_r(0) - v_0 = v_\theta(0) - v_0 \\
0 &= \nu_1 + \lambda_r(0) = \nu_2 + \lambda_\theta(0) = \nu_3 + \lambda_{v_r}(0) = \nu_4 + \lambda_{v_\theta}(0) = \nu_5 + \lambda_m(0) = \nu_6 + c_2 \\
0 &= m(0) - m_0 = q_2(0) = v_r(f) - v_{r_f} = v_\theta(f) - \sqrt{\frac{\mu}{r(f)}} \\
0 &= \lambda_r(1) + 2r(1) - \nu_7 \frac{1}{2r(1)}\sqrt{\frac{\mu}{r(1)}} = c_1 = \lambda_{v_r}(1) - \nu_8 \\
0 &= \lambda_{v_\theta}(1) - \nu_9 = \lambda_m(1) = c_2 - \nu_{10} = H(1)
\end{aligned}
\tag{7.46}
$$

This is our final Hamiltonian BVP that we will solve. Overall, this process was lengthy and error-prone when done by hand. Instead, we can succinctly describe this process using the language introduced in this chapter as follows

$$
\text{Problem } \Lambda = (\mathscr{F}_{\text{MWM}} \circ \mathscr{F}_{\text{MWM}} \circ \mathscr{F}_{\text{scale time}} \circ \mathscr{D}_{\text{DG}} \circ \mathscr{F}_{\text{time shift}})(\text{Problem } \Sigma) \tag{7.47}
$$

Using the sequence in Eq. (7.47), we were able to automatically generate the Hamiltonian BVP from the original OCP using `beluga` v0.3.3. To solve the resulting Hamiltonian BVP numerically, we use both shooting and collocation algorithms. The trajectories are shown in Fig. 7.11.

By eliminating θ and λ_θ in the reduction process, solving for θ is moved to the reconstruction process in the BVP solver. Solutions for $q \equiv \theta$ are shown in Fig. 7.12, and are displayed on top of known solutions for θ. From these figures, we see that there

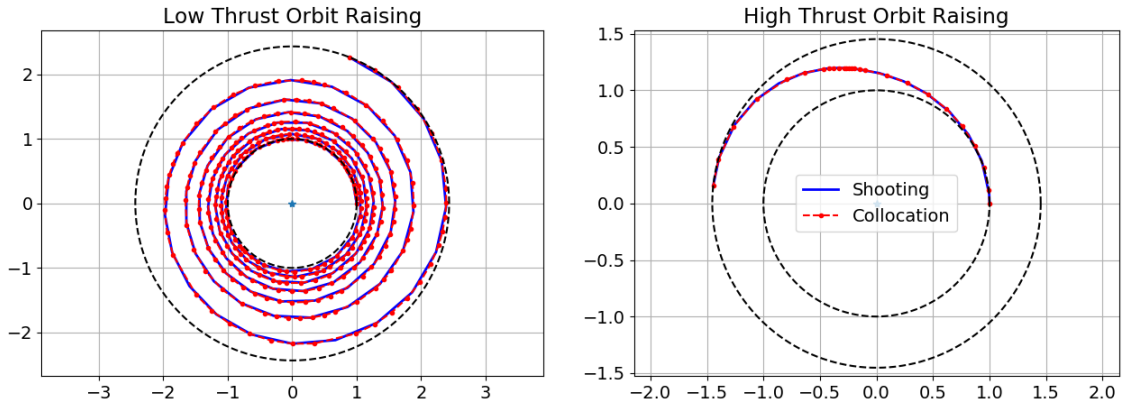


Figure 7.11. Solutions in the low and high thrust orbit raising problems.

is agreement across the known, reduced shooting, and also the reduced collocation solutions.

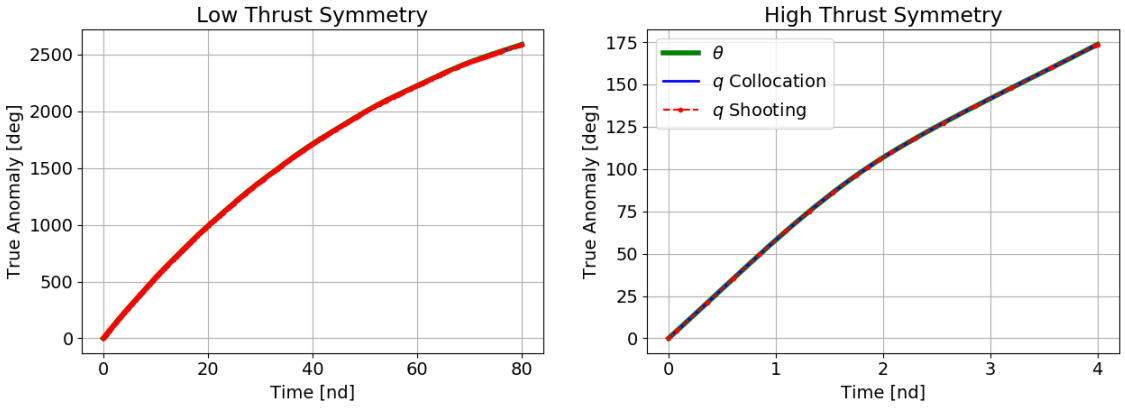


Figure 7.12. Symmetries in the low and high thrust orbit raising problems.

7.6.3 Moon Lander

The moon lander is a classic OCP that solves the problem of an aerospace vehicle in its terminal descent. The engines must be activated at a precise time to safely land the vehicle on a surface. If the engines are activated too late, the vehicle will strike the

surface with a non-zero velocity and if the engines are activate too early, the vehicle will stop moving above the surface. This problem has a control path-constraint which results in a bang-bang control. Mall's UTM strategy has previously been shown to be very effective for solving path-constrained problems in a phaseless manner. In this problem, we show that using a composable workflow like introduced in this section enables reduction and UTM to work together. The moon lander OCP is defined as follows

$$\begin{aligned}
\min_u K &= \int_0^{t_f} u \, \mathrm{d}t \\
\frac{\mathrm{d}x}{\mathrm{d}t} &= v \\
\frac{\mathrm{d}v}{\mathrm{d}t} &= u - g \\
S &= \frac{\partial}{\partial x} \\
0 &\leq u \leq 4 \\
0 &= h(t_0) - h_0 = v(t_0) - v_0 = h(t_f) = v(t_f)
\end{aligned} \tag{7.48}$$

Next, we will append time as a state.

$$\begin{aligned}
\min_u K &= \int_0^{\tau_f} u \, \mathrm{d}\tau \\
\frac{\mathrm{d}x}{\mathrm{d}\tau} &= v \\
\frac{\mathrm{d}v}{\mathrm{d}\tau} &= u - g \\
\frac{\mathrm{d}t}{\mathrm{d}\tau} &= 1 \\
S &= \frac{\partial}{\partial x} \\
0 &\leq u \leq 4 \\
0 &= h(\tau_0) - h_0 = v(\tau_0) - v_0 = t(\tau_0) = h(\tau_f) = v(\tau_f)
\end{aligned} \tag{7.49}$$

Then, we will deal with the path-constraint using Mall's UTM. This augments the cost functional with a trigonometric term.

$$\begin{aligned}
\min_u K &= \int_0^{\tau_f} \epsilon \left(\sec\left(\pi \frac{u-2}{4} \right) - 1 \right) + u \, \mathbb{d}\tau \\
\frac{\mathrm{d}x}{\mathbb{d}\tau} &= v \\
\frac{\mathrm{d}v}{\mathbb{d}\tau} &= u - g \\
\frac{\mathrm{d}t}{\mathbb{d}\tau} &= 1 \\
S &= \frac{\partial}{\partial x} \\
0 &= h(\tau_0) - h_0 = v(\tau_0) - v_0 = t(\tau_0) = h(\tau_f) = v(\tau_f)
\end{aligned} \tag{7.50}$$

Now that we have dealt with the path-constraint, we can dualize the problem using the procedure from Chapter 3. While still possible, root-solving the control law in PMP is very challenging for problems that use UTM. To ease the some of the burden on the algebra involved, we will instead use the Symplectic ICRM strategy described in Chapter 4.

$$\begin{aligned}
H &= \epsilon \left(\sec\left(\pi \frac{u-2}{4} \right) - 1 \right) + u + \lambda_x v + \lambda_t + \lambda_v (u - g) \\
\omega &= \mathbb{d}x \wedge \mathrm{d}\lambda_x + \mathbb{d}v \wedge \mathrm{d}\lambda_v + \mathrm{d}t \wedge \mathrm{d}\lambda_t + (\mathbb{d}u - \dot{u}^* \, \mathbb{d}t) \wedge \mathrm{d}\lambda_u \\
S_1 &= \frac{\partial}{\partial x} \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= h(\tau_0) - h_0 = v(\tau_0) - v_0 = t(\tau_0) = \nu_1 + \lambda_x(\tau_0) = \nu_2 + \lambda_v(\tau_0) = \nu_3 + \lambda_t(\tau_0) \\
0 &= \lambda_u(\tau_0) = h(\tau_f) = v(\tau_f) = \lambda_x(\tau_f) - \nu_4 = \lambda_v(\tau_f) - \nu_5 = \lambda_t(\tau_f) = H(\tau_f)
\end{aligned} \tag{7.51}$$

Where we define $\dot{u}$ to be

$$\dot{u} = \frac{\lambda_x}{2\pi^2 \epsilon \sin\left(\pi \frac{u-2}{4}\right)^2 \left(16 \cos\left(\pi \frac{u-2}{4} \right)^3 \right)^{-1} + \pi^2 \epsilon \left(16 \cos\left(\pi \frac{u-2}{4} \right) \right)^{-1}} \tag{7.52}$$

To eliminate the free final time of this Hamiltonian BVP, we scale the entire problem by the final time. This adds τ_f as a dynamical parameter.

$$\begin{aligned}
H &= \epsilon\left(\sec\left(\pi\frac{u-2}{4}\right)-1\right)+u+\lambda_x v+\lambda_t+\lambda_v(u-g)\\
\omega &= \tau_f^{-1}\,\mathbb{d}x\wedge\mathbb{d}\lambda_x+\tau_f^{-1}\,\mathbb{d}v\wedge\mathbb{d}\lambda_v+\tau_f^{-1}\,\mathbb{d}t\wedge\mathbb{d}\lambda_t+\tau_f^{-1}\left(\mathbb{d}u-\dot{u}^*\,\mathbb{d}t\right)\wedge\mathbb{d}\lambda_u\\
S_1 &= \frac{\partial}{\partial x}\\
S_2 &= \frac{\partial}{\partial t}\\
0 &= h(0)-h_0=v(0)-v_0=t(0)=\nu_1+\lambda_x(0)=\nu_2+\lambda_v(0)=\nu_3+\lambda_t(0)\\
0 &= \lambda_u(0)=h(1)=v(1)=\lambda_x(1)-\nu_4=\lambda_v(1)-\nu_5=\lambda_t(1)=H(1)
\end{aligned}\tag{7.53}$$

Finally, we will use our reduction procedure on both S_1 and S_2. Starting with S_1, we eliminate the (x,λ_x) pair.

$$\begin{aligned}
H &= \epsilon\left(\sec\left(\pi\frac{u-2}{4}\right)-1\right)+u+c_1 v+\lambda_t+\lambda_v(u-g)\\
\omega &= \tau_f^{-1}\,\mathbb{d}v\wedge\mathbb{d}\lambda_v+\tau_f^{-1}\,\mathbb{d}t\wedge\mathbb{d}\lambda_t+\tau_f^{-1}\left(\mathbb{d}u-\dot{u}^*\,\mathbb{d}t\right)\wedge\mathbb{d}\lambda_u\\
\dot{q}_1 &= \tau_f v\\
S_2 &= \frac{\partial}{\partial t}\\
0 &= q_1(0)-h_0=v(0)-v_0=t(0)=\nu_1+c_1=\nu_2+\lambda_v(0)=\nu_3+\lambda_t(0)\\
0 &= \lambda_u(0)=q_1(1)=v(1)=c_1-\nu_4=\lambda_v(1)-\nu_5=\lambda_t(1)=H(1)
\end{aligned}\tag{7.54}$$

In applying this procedure, we reduced our 8-dimensional system down to a 6-dimensional. Next, we applying the reduction procedure again taking our system from 6 dimensions

down to 4 dimensions. Our final BVP in a suitable form for the numerical solvers in Chapter 6 is as follows

$$\begin{aligned}
\dot{v} &= \tau_f(u-g) \\
\dot{u} &= \frac{\tau_f c_1}{2\pi^2\epsilon \sin\left(\pi\frac{u-2}{4}\right)^2 \left(16\cos\left(\pi\frac{u-2}{4}\right)^3\right)^{-1} + \pi^2\epsilon\left(16\cos\left(\pi\frac{u-2}{4}\right)\right)^{-1}} \\
\dot{\lambda_v} &= -\tau_f c_1 \\
\dot{\lambda_u} &= -\tau_f\left(\pi\epsilon\frac{\sin(\pi\frac{u-2}{4})}{4\cos(\pi\frac{u-2}{4})^2} + \lambda_v + 1\right) \\
\dot{q}_1 &= \tau_f v \\
\dot{q}_2 &= \tau_f \\
S_2 &= \frac{\partial}{\partial t} \\
0 &= q_1(0) - h_0 = v(0) - v_0 = q_2(0) = \nu_1 + c_1 = \nu_2 + \lambda_v(0) = \nu_3 + c_2 \\
0 &= \lambda_u(0) = q_1(1) = v(1) = c_1 - \nu_4 = \lambda_v(1) - \nu_5 = c_2 = H(1)
\end{aligned} \tag{7.55}$$

Although the system of differential equations in Eq. (7.55) appears to be 6-dimensional, it is seen as 4-dimensional by the BVP solver since both q_1 and q_2 are quads. In comparison to the both the Brachistochrone and orbit raising examples, the moon lander problem is significantly more tedious. This is due to the fact that we used UTM to handle the path-constraints and subsequently used Symplectic ICRM for the control law. This is incredibly challenging to perform by-hand. Instead, since every step can be boiled down to a functor transforming the problem definition in a known manner, we can use software to automate this entire process. To solve this OCP, we use `beluga` v0.3.3 with the following signature

$$\text{Problem } \Lambda = (\mathscr{F}_{\text{MWM}} \circ \mathscr{F}_{\text{MWM}} \circ \mathscr{F}_{\text{scale time}} \circ \mathscr{D}_{\text{DG}} \circ \mathscr{F}_{\text{UTM}} \circ \mathscr{F}_{\text{time shift}})(\text{Problem } \Sigma) \tag{7.56}$$

The resulting Hamiltonian BVP was then solved with collocation. A solution is shown in Fig. 7.13 with q overlayed on the known solution for x.

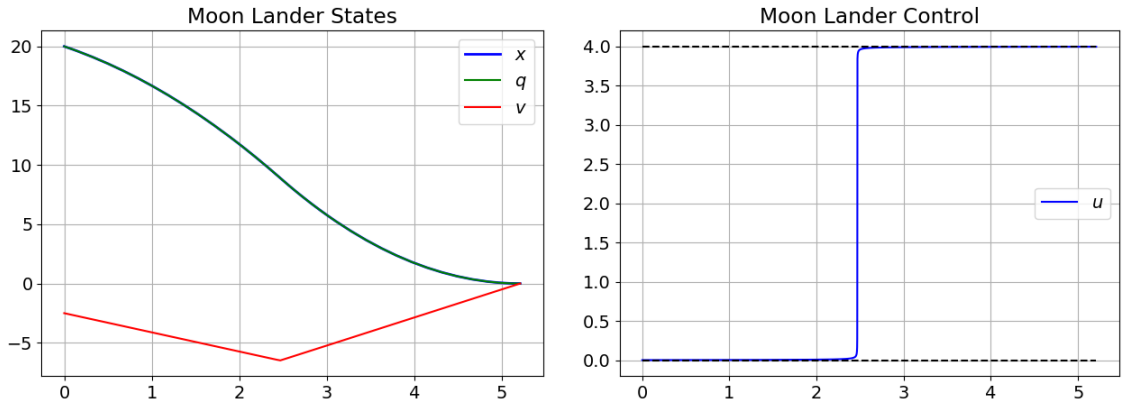


Figure 7.13. Symmetries in the low and high thrust orbit raising problems.

7.7 Extension

By compartmentalizing strategies into functors as described in this section, commutation relations between the strategies may be identified. For example, for a single path-constraint, application of UTM after Eps-Trig yields a different result than if Eps-Trig is applied after UTM. This is because with only a single path-constraint, the output of the first functor is an unconstrained OCP. The second functor has no path-constraints to adjoin, so in this example, our commutation relations are shown in Eq. (7.57).

$$\mathscr{F}_{\text{UTM}} \circ \mathscr{F}_{\text{Eps-Trig}} \neq \mathscr{F}_{\text{Eps-Trig}} \circ \mathscr{F}_{\text{UTM}} \tag{7.57}$$

In another example, say there exists an OCP with n path-constraints where $n \geq 2$. In this case, as long as Eps-Trig and UTM are each handling a different path-constraint, they commute. This relationship is shown in Eq. (7.58).

$$\mathscr{F}^{i}_{\text{UTM}} \circ \mathscr{F}^{j}_{\text{Eps-Trig}} = \mathscr{F}^{j}_{\text{Eps-Trig}} \circ \mathscr{F}^{i}_{\text{UTM}} \; \forall \; i \neq j \tag{7.58}$$

Assuming there is a restriction in place such that a designer is not allowed to apply both UTM and Eps-Trig to the same path-constraint, then Eq. (7.58) suggests that either process can be first. A developer constructing such a system would be free to choose where they place the UTM and Eps-Trig strategies without much concern

for adverse side-effects. On the other hand, take dualization and UTM for instance. Since UTM augments the path cost of an OCP, but a Hamiltonian BVP no longer has an explicit path cost function, these two functors do not commute. This is shown in Eq. (7.59).

$$\mathscr{F}_{\text{UTM}} \circ \mathscr{D} \neq \mathscr{D} \circ \mathscr{F}_{\text{UTM}} \tag{7.59}$$

In this case, it is likely obvious that UTM must be applied prior to dualization. Since UTM's signature is $(\Sigma \to \Sigma)$, it clearly can't take the Λ input given by dualization. This provides a mathematical rule for preventing UTM from appearing after dualization. As we saw before in the examples, strictly adhering to these rules enables one to design a system that can mix and match several functors. This same concept can be used with new functors that have not yet been developed. For instance, someone creates a strategy, $\mathscr{F}_{\text{new}}$. By filling out the commutation relations as shown in Table 7.1, some clues can be given as to where functor $\mathscr{F}_{\text{new}}$ should be used. In this example, the strategy commutes with all OCFs and MWM. This researcher likely developed a method for modifying data in the OCP, potentially "UTM at the boundaries". Even though this functor commutes with MWM, since MWM commutes with dualization, $\mathscr{F}_{\text{new}}$ would likely be placed before dualization.

Table 7.1. Example composition table.

Functor	Commutes with $\mathscr{F}_{\text{new}}$
$\mathscr{F}_{\text{UTM}}$	Y
$\mathscr{F}_{\text{Eps-Trig}}$	Y
$\mathscr{F}_{\text{RASHS}}$	Y
$\mathscr{F}_{\text{time shift}}$	Y
$\mathscr{D}$	N
$\mathscr{F}_{\text{ICRM}}$	N
$\mathscr{F}_{\text{scale time}}$	Y
$\mathscr{F}_{MWM}$	Y

One case where this strategy was used was in the time shift functor. We chose to place this strategy prior to dualization, and more specifically ICRM. This is because the time shift and symplectic ICRM strategies do not commute. In Chapter 4, the symplectic form was augmented with the following two-form

$$(\mathbb{d}\boldsymbol{u}^* - \dot{\boldsymbol{u}}^* \,\mathbb{d}t) \wedge (\mathbb{d}\boldsymbol{\lambda}_{\boldsymbol{u}} - 0\,\mathbb{d}t) \tag{7.60}$$

In this scenario, $\mathbb{d}t$ was used as an element of the co-vector basis, however such a quantity would not exist if t was an independent variable as opposed to a state. This implies time shifting must occur prior to ICRM.

7.8 Summary

In this chapter, we introduced the concept of a composable process for constructing Hamiltonian BVPs. This type of construction lends itself well to functional programming. By taking known strategies, such as UTM, RASHS, and reduction, and categorizing them into functors on either OCPs or Hamiltonian BVPs, we created a process that enables usage of these strategies together. This abstraction enabled us to further improve on our reduction process by extending it to multi-phase path-constrained problems. The ultimate benefit of this chapter is a repeatable process of indirect methods that bypasses the laborious and error-prone manipulations that an aerospace system designer either carries out by-hand or using an object-oriented process. This new representation compresses processes that can take several pages down to a single line equation.

Due to the more restrictive nature of this representation, systems produced in this manner must preserve mathematical structure. The result is that any strategy in this representation fits in Ross and Fahroo's famous commutative diagram. Thus, there is a deep connection between the computational implementation of a strategy and its mathematical implications. Because of this, we were then able to give a method for implementing new strategies that give predictable results and simultaneously fit in Ross and Fahroo's construct.

8. CONCLUSIONS

In this dissertation we explored the Marsden-Weinstein-Meyer reduction theorem, its application to aerospace mission design, and many practical considerations when numerical designing methods around this theorem. This novel approach to tackling general-purpose aerospace mission design results in computational problems that are lower dimensional than the current state-of-the-art indirect methods. These computational problems ultimately require the numerical solution of fewer equations. To recap the end-to-end process created, recall Fig. 1.6, which is reproduced here in Fig. 8.1.

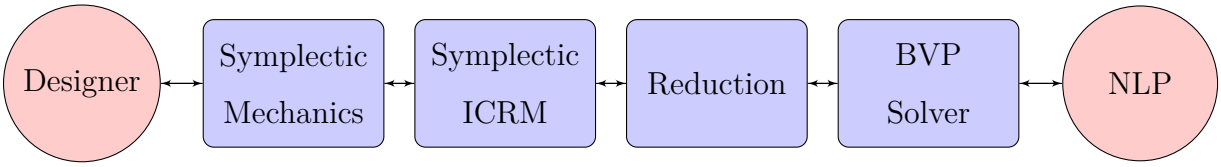


Figure 8.1. End-to-end mission design process as constructed in this dissertation.

In Chapter 3, we reformulated indirect methods in terms of differential geometry. Specifically symplectic mechanics was chosen due to its exposed differential structure as well as its easy-to-use tensorial language. The resulting Hamiltonian boundary-value problems from this chapter were identical to the traditional indirect methods. This reformulation is consistent with traditional approaches while also enabling more advanced techniques, such as reduction, that cannot be used with traditional approaches.

Next, in Chapter 4, we improved upon Thomas's Integrated Control Regularization Method (ICRM) thus creating a symplectic ICRM strategy. The issue of transcendental control laws exists equivalently in the traditional and symplectic systems. Thomas's ICRM has been used successfully several times on the traditional

system for solving these control laws where Pontryagin's Minimum Principle (PMP) fails. The contribution of Chapter 4 was a reformulation of ICRM in symplectic mechanics so that it may be applied to solve the symplectic system we have created. This new system enables reduction in cases where PMP fails to solve for control laws.

Then, in Chapter 5, we created a practical method for carrying out Marsden-Weinstein-Meyer reduction on a general set of problems. Even though reduction has been used on practical problems before, it has never been used as strategy for general-purpose applications. We explored some of the practical limitations as well as carried reduction on several examples. The resulting number of equations-of-motion for each of these problems was lower than is typically possible with standard calculus. This reduction in dimension can lead to a faster numerical solution of large problems and may enable solving problems of a large size that was not possible with traditional approaches.

The resulting mathematical problem from reduction had a strikingly different structure than is typically seen with BVPs. As a result, we created a new set of BVP solvers to handle this structure in Chapter 6. Despite that these solvers had mixed results in terms of performance, the reduced systems we examined had between 50% and 75% fewer equations to solve. The main contribution of these new BVP solvers is that they are a blueprint for creating new algorithms for solving reduced problems. Presently, the two solvers in Chapter 6 are the only solvers in existence that are capable of solving problems resulting from reduction.

This entire process carries an aerospace system designer's hypothetical aerospace mission to an NLP problem that may be solved using nearly any NLP solver. Rather, we created a practical process for applying Marsden-Weinstein-Meyer reduction to general-purpose aerospace mission design. Our final topic in Chapter 7 then touched on the issue of problems with path-constraints, discontinuous functions, and other difficulties that typically force a designer to create challenging multi-point boundary-value problems by hand. We created a composable process of constructing problems

using indirect methods that simplified constructions for a designer, eliminated error-prone calculations, and enabled usage of reduction with other strategies.

9. FUTURE WORK

9.1 Sequential Mishchenko-Fomenko Reduction

In the reduction section of this dissertation, we were limited to the reduction of Hamiltonian BVPs containing constants-of-motion that were in involution with one another. In the case where constants-of-motion under the Poisson bracket do not vanish, our strategy does not work. The introduction of a composable process for constructing Hamiltonian BVPs out of OCPs enables an incredibly complicated processes to be implemented in an automated fashion with ease. By repeatedly applying the reduction strategy presented in this dissertation in the composable manner as described in Chapter 7, it is possible that we may reduce each quantity in series. Instead of trying to apply a full Marsden-Weinstein-Meyer process using isotropy subgroups, it may be possible to apply the Mishchenko-Fomenko process sequentially. The sequential Mishenko-Fomenko process as described here is shown in Fig. 9.1.

9.2 Reduced Dimensional Co-vector Mapping Principle

Although the resulting BVP from dualization was heavily modified, special care was taken to preserve its structure. We expect the CMP to be preserved, thus creating a Reduced Dimensional CMP (RCMP). Since parameter reduction and symmetry reduction can both be applied to the direct system, the famous commutative diagram of Ross and Fahroo splits in to two diagrams: one for an OCP possessing symmetries and one for an OCP possessing constants-of-motion. These diagrams are shown in Figs. 9.2 and 9.3 respectively

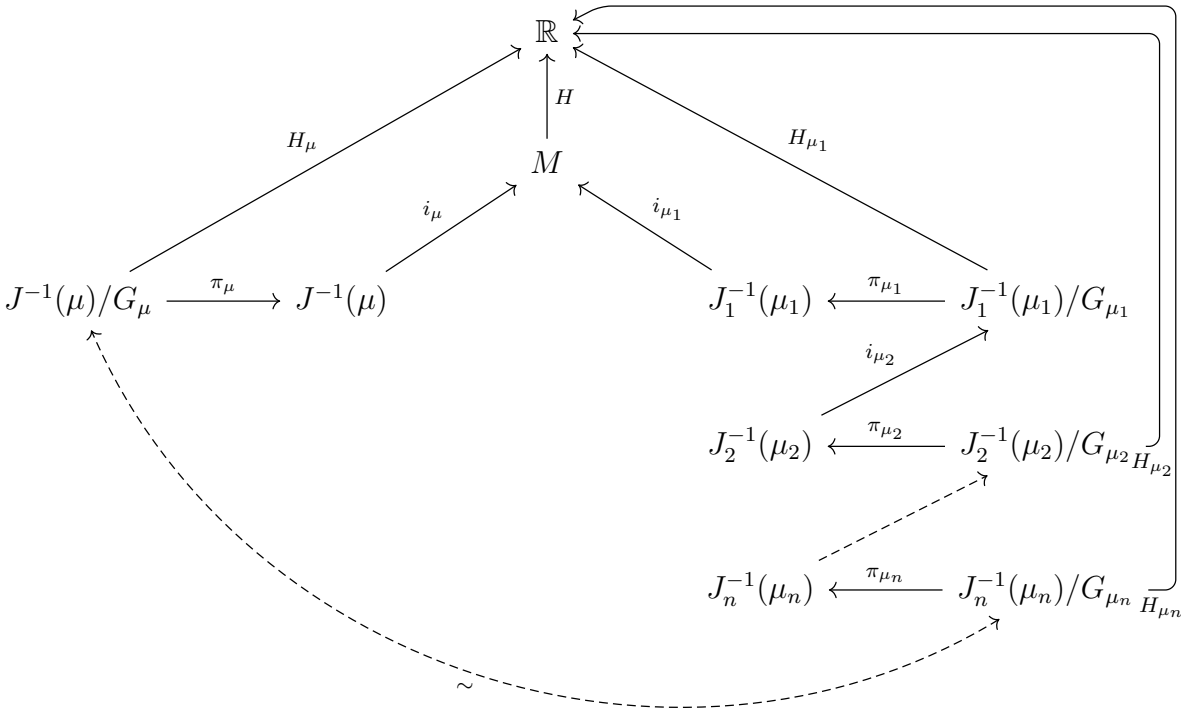


Figure 9.1. Marsden-Weinstein-Meyer reduction and sequential Mishchenko-Fomenko reduction may result in diffeomorphic manifolds.

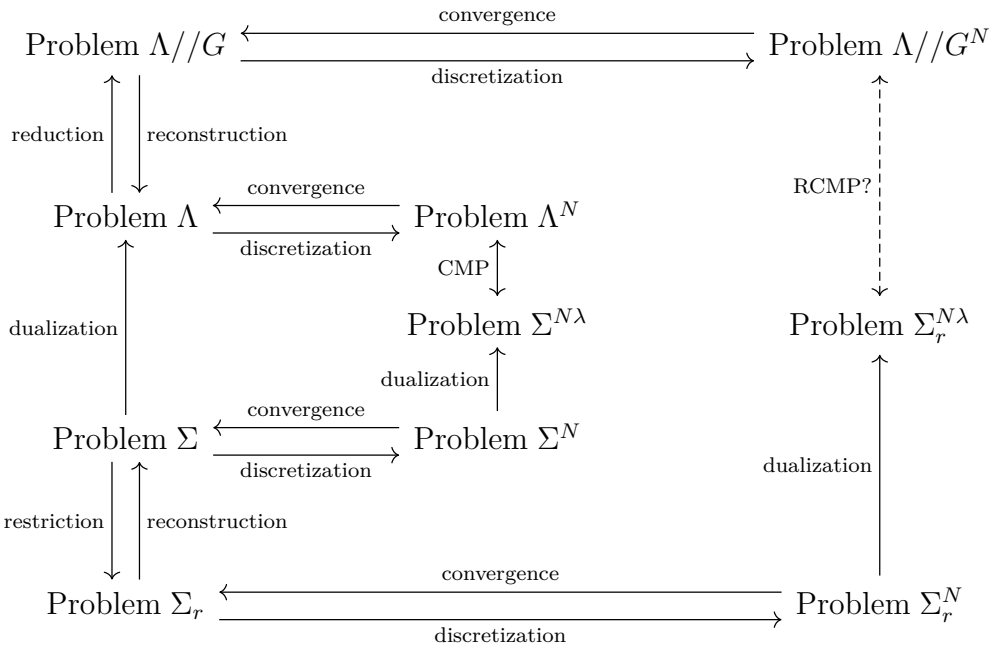


Figure 9.2. Commutative diagram for dualization and discretization of an OCP's dynamical system possessing constants-of-motion.

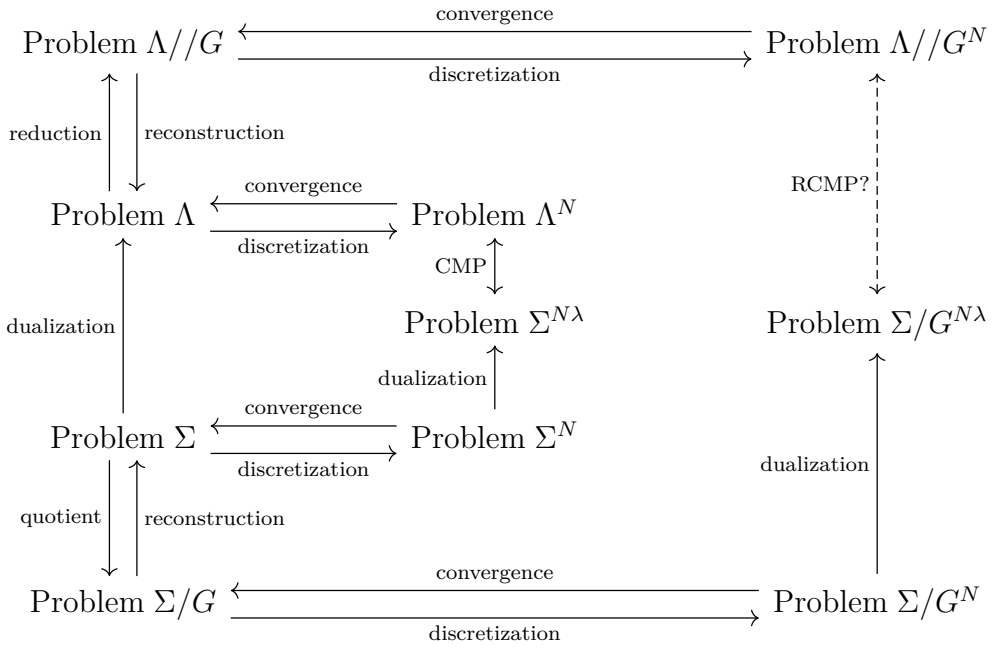


Figure 9.3. Commutative diagram for dualization and discretization of an OCP's dynamical system possessing symmetries.

9.3 High Performance Numerical BVP Solvers

Nonlinear programming techniques are typically driven by solving dense linear systems in the form of QP subproblems. At their worst, solving these dense linear systems has computational complexity $O(n^3)$. Increasing a problem size by 10 times will increase the computational cost by 1000 times. The numerical algorithms in this work were driven by such dense methods. It is possible to improve the performance of the numerical algorithms in this work by implementing sparse methods. There are a number of computational tools available for solving sparse linear systems [215]. One software that is particularly effective is `SuperLU` which has seen some use in sparse BVP solvers [216]. After an adequate sparse method has been developed, the next step is to implement parallelized batch integration techniques as described in Chapters 5 and 6.

9.4 0-D Boundary-Value Problems

In Chapter 5, we came across an example that started as a two-dimensional OCP, but was reduced to a zero-dimensional Hamiltonian BVP through reduction. In the literature, there appears to be no exploration of zero-dimensional BVPs. This is likely for two reasons:

1. Since the majority of reduction is applied to problems from a theoretical standpoint, numerical examples and NLPs are seldom generated. It is possible the only zero-dimensional BVPs in existence are in this dissertation.

2. A zero-dimensional BVP is not a BVP.

One would think that a zero-dimensional BVP is analytically solvable. From a dynamical standpoint this seems to be the case. The Liouville-Arnold theorem further strengthens this point. However, from an aerospace mission standpoint, there exist quantities at the boundaries that must still be solved. I hypothesize that, in general, zero-dimensional BVPs collapse down to a more traditional type of NLP that does

not need collocation, shooting, pseudospectral, or other BVP algorithms to be solved. Therefore, zero-dimensional BVPs should not be viewed as BVPs. A point of future work is to explore its theoretical implications, as well as practical implementations of a program to solve such a problem.

9.5 Lie-Poisson Reformulation

In Chapter 3, we reformulated modern trajectory optimization algorithms in terms of symplectic geometry, although some instances used a Poisson formulation that was more convenient. In Chapter 5, we created the following formula for extracting dynamics from a Hamiltonian vector field:

$$\dot{\boldsymbol{q}} = X_H^\flat([\cdot, \boldsymbol{g}]_{\mathrm{P}}) \tag{9.1}$$

We noted this equation bears a striking resemblance to the Lie-Poisson bracket:

$$[F, G]_{\text{L-P}} = \left\langle \xi, \left[\frac{\delta F}{\delta \xi}, \frac{\delta G}{\delta \xi} \right]_{\mathrm{L}} \right\rangle \tag{9.2}$$

Both of these equations appear to be leveraging the Poisson and Lie structures. A point of further improvement to this dissertation would be in exploring the Lie-Poisson bracket, its usage in indirect methods of trajectory optimization, and practical implementation in modern software packages.

REFERENCES

[1] Jeremiah Gertler. Air force f-22 fighter program. Library of Congress Washington DC Congressional Research Service, 2013.

[2] Department of Defense. Selected aquisition report: F-22 increment 3.2b modernization (f-22 inc 3.2b mod). Defense Acquisition Management Information Retrieval, DAMIR, 2019.

[3] I Michael Ross. A historical introduction to the covector mapping principle. In *Proceedings of Astrodynamics Specialists Conference*. Naval Postgraduate School (US), 2005.

[4] Ivor Grattan-Guinness and Henk JM Bos. *From the calculus to set theory, 1630-1910: An introductory history*. Princeton University Press, 2000.

[5] Céline Parzani and Stéphane Puechmorel. On a hamilton-jacobi-bellman approach for coordinated optimal aircraft trajectories planning. *Optimal Control Applications and Methods*, 39(2):933–948, 2018.

[6] Richard Bellman. Dynamic programming and a new formalism in the calculus of variations. *Proceedings of the National Academy of Sciences of the United States of America*, 40(4):231, 1954.

[7] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[8] Richard Ernest Bellman and Stuart E Dreyfus. An application of dynamic programming to the determination of optimal satellite trajectories. 1959.

[9] James M Longuski, José J Guzmán, and John E Prussing. *Optimal control with aerospace applications*. Springer, 2014.

[10] Arthur E. Jr Bryson and Ju-Chi Ho. Applied optimal control. 1975.

[11] I Michael Ross and Fariba Fahroo. A pseudospectral transformation of the convectors of optimal control systems. *IFAC Proceedings Volumes*, 34(13):543–548, 2001.

[12] I. Michael Ross and Fariba Fahroo. Legendre pseudospectral approximations of optimal control problems. In *New Trends in Nonlinear Dynamics and Control and their Applications*, pages 327–342. Springer, 2003.

[13] G. L. Brauer, D. E. Cornick, A. R. Habeger, F. M. Petersen, and R. Stevenson. Program to optimize simulated trajectories (POST). Volume 1: Formulation manual. *Martin Marietta Corp. Report*, 1, 1975.

[14] Jason K. Moore and Antonie van den Bogert. opty: Software for trajectory optimization and parameter identification using direct collocation. *The Journal of Open Source Software*, 3(21):300, January 2018.

[15] Charles R. Hargraves and Stephen W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[16] Ryan D. Gauntt. Aircraft course optimization tool using GPOPS matlab code. Technical report, DTIC Document, 2012.

[17] Anil V. Rao, David A. Benson, Christopher Darby, Michael A. Patterson, Camila Francolin, Ilyssa Sanders, and Geoffrey T. Huntington. Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):22, 2010.

[18] I. Michael Ross. A beginner's guide to dido. *Ellisar, LLC, Monterey, CA*, 2007, 1998.

[19] Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[20] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for sol/npsol: A fortran package for nonlinear programming. *Report SOL 83-12*, 1983.

[21] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, April 2002.

[22] Doug Nelson. Qualitative and quantitative assessment of otis and post. Technical report, Atlanta, Georgia: Georgia Institute of Technology, 2001.

[23] Michael Posa and Russ Tedrake. Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic foundations of robotics X*, pages 527–542. Springer, 2013.

[24] I. Michael Ross. A historical introduction to the covector mapping principle. In *Advances in the Astronautical Sciences: Astrodynamics 2005, Vol. 122, Paper AAS*, pages 05–332, 2005.

[25] CH Still. The parallel bfgs method for unconstrained minimization. Technical report, Lawrence Livermore National Lab., CA (United States), 1991.

[26] X Chen. A parallel bfgs-sqp method for stochastic linear programs. *Computational Techniques and Applications, World Scientific*, pages 67–74, 1995.

[27] Ernesto E Prudencio, Richard Byrd, and Xiao-Chuan Cai. Parallel full space sqp lagrange–newton–krylov–schwarz algorithms for pde-constrained optimization problems. *SIAM Journal on Scientific Computing*, 27(4):1305–1328, 2006.

[28] Yun Fei, Guodong Rong, Bin Wang, and Wenping Wang. Parallel l-bfgs-b algorithm on gpu. *Computers and Graphics*, 40:1 – 9, 2014.

[29] François Irigoin, Pierre Jouvelot, and Rémi Triolet. Semantical interprocedural parallelization: An overview of the pips project. In *ICS*, volume 91, pages 244–251, 1991.

[30] Keith D Cooper, Mary W Hall, Robert T Hood, Ken Kennedy, Kathryn S McKinley, John M Mellor-Crummey, Linda Torczon, and Scott K Warren. The parascope parallel programming environment. *Proceedings of the IEEE*, 81(2):244–263, 1993.

[31] William Blume, Rudolf Eigenmann, Keith Faigin, John Grout, Jay Hoeflinger, David Padua, Paul Petersen, William Pottenger, Lawrence Rauchwerger, Peng Tu, et al. Effective automatic parallelization with polaris. In *International Journal of Parallel Programming*. Citeseer, 1995.

[32] Mary W Hall, Saman P Amarasinghe, Brian R Murphy, Shih-Wei Liao, and Monica S Lam. Detecting coarse-grain parallelism using an interprocedural parallelizing compiler. In *Supercomputing'95: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, pages 49–49. IEEE, 1995.

[33] Mary W Hall, Jennifer M Anderson, Saman P. Amarasinghe, Brian R Murphy, Shih-Wei Liao, Edouard Bugnion, and Monica S Lam. Maximizing multiprocessor performance with the suif compiler. *Computer*, 29(12):84–89, 1996.

[34] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[35] Inria Saclay Team Commands. Bocop: an open source toolbox for optimal control. `http://bocop.org`, 2017.

[36] Frédéric Bonnans, Pierre Martinon, and Vincent Grélard. Bocop-a collection of examples, 2012.

[37] Peter Friedrich Gath. Camtos-a software suite combining direct and indirect trajectory optimization methods. 2002.

[38] Cesar Ocampo and Juan Senent. The design and development of copernicus: A comprehensive trajectory design and optimization system. In *Proceedings of the International Astronautical Congress, IAC-06-C1*, volume 4, 2006.

[39] Jacob Williams, Juan S Senent, Cesar Ocampo, Ravi Mathur, and Elizabeth C Davis. Overview and software architecture of the copernicus trajectory design and optimization system. 2010.

[40] Cesar Ocampo, Juan S Senent, and Jacob Williams. Theoretical foundation of copernicus: a unified system for trajectory design and optimization. 2010.

[41] Jacob Williams, Juan S Senent, Cesar Ocampo, and David E Lee. Recent improvements to the copernicus trajectory design and optimization system. 2012.

[42] Jacob Williams, Robert D Falck, and Izaak B Beekman. Application of modern fortran to spacecraft trajectory design and optimization. In *2018 Space Flight Mechanics Meeting*, page 1451, 2018.

[43] Oskar Von Stryk. Ein direktes verfahren zur bahnoptimierung von luft-und raumfahrzeugen unter berücksichtigung von beschränkungen. *Zeitschrift für angewandte Mathematik und Mechanik*, 71(6):T705–T706, 1991.

[44] Oskar Von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control*, pages 129–143. Springer, 1993.

[45] Oskar Von Stryk. User's guide for dircol: A direct collocation method for the numerical solution of optimal control problems. *Lehrstuhl für Höhere Mathematik und Numerische Mathematik, Technische Universität, München*, 2, 1999.

[46] M Cizniar, M Fikar, and MA Latifi. Matlab dynamic optimisation code dynopt. *User's Guide. Bratislava, Slovak Republic*, 2006.

[47] M Cizniar, D Salhi, M Fikar, and MA Latifi. A matlab package for orthogonal collocations on finite elements in dynamic optimisation. In *Proceedings of the 15th international conference process control*, volume 5, page 058f, 2005.

[48] Ayonga Hereid and Aaron D Ames. Frost: Fast robot optimization and simulation toolkit. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726. IEEE, 2017.

[49] Paola Falugi, Eric Kerrigan, and Eugene Van Wyk. Imperial college london optimal control software user guide (iclocs). *Department of Electrical and Electronic Engineering, Imperial College London, London, England, UK*, 2010.

[50] Daniel B. Leineweber. The theory of muscod in a nutshell. Master's thesis, University of Heidelberg, 1995.

[51] Peter Kühl, Joachim Ferreau, Jan Albersmeyer, Christian Kirches, Leonard Wirsching, Sebastian Sager, Andreas Potschka, Gerrit Schulz, Moritz Diehl, Daniel B Leineweber, et al. Muscod-ii users manual. *University of Heidelberg*, 2007.

[52] Christof Büskens. Direkte optimierungsmethoden zur numerischen berechnung optimaler steuerungen. Master's thesis, Institut für Numerische Mathematik, 1993.

[53] Christof Büskens. Optimierungsmethoden und sensitivitatsanalyse fur optimale steuerprozesse mit steuer-und zustands-beschrankungen. *Westfalische Wilhelms-Universitat Munster*, 1998.

[54] Takahiro Inagawa and Tanaka Susumu. Opengoddard - trajectory optimization for python. `https://github.com/istellartech/OpenGoddard`, 2019.

[55] Jonas C. Koenemann. Open optimal control library. `https://github.com/OpenOCL/OpenOCL`, 2019.

[56] Matthew Kelly. An introduction to trajectory optimization: how to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[57] Matthew Kelly. Optimtraj - trajectory optimization for matlab. `https://github.com/MatthewPeterKelly/OptimTraj`, 2019.

[58] Raktim Bhattacharya. Optragen: A matlab toolbox for optimal trajectory generation. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 6832–6836. IEEE, 2006.

[59] John Riehl, Stephen Paris, and Waldy Sjauw. Comparison of implicit integration methods for solving aerospace trajectory optimization problems. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 6033, 2006.

[60] Per E Rutquist and Marcus M Edvall. Propt-matlab optimal control software. *Tomlab Optimization Inc*, 260(1):12, 2010.

[61] Victor M Becerra. Solving complex optimal control problems at no cost with psopt. In *2010 IEEE International Symposium on Computer-Aided Control System Design*, pages 1391–1396. IEEE, 2010.

[62] Dario Izzo and Francesco Biscani. Pykep. `https://github.com/esa/pykep`, 2019.

[63] Adam Lowell Schwartz. *Theory and implementation of numerical methods based on Runge-Kutta integration for solving optimal control problems*. PhD thesis, University of California, Berkeley, 1996.

[64] John T Betts and William P Huffman. Sparse optimal control software socs. *Mathematics and Engineering Analysis Technical Document MEA-LR-085, Boeing Information and Support Services, The Boeing Company, PO Box*, 3707:98124–2207, 1997.

[65] Marco Sagliano, Stephan Theil, Vincenzo D'Onofrio, and Michiel Bergsma. Spartan: A novel pseudospectral algorithm for entry, descent, and landing analysis. In *4th EuroGNC Conference*, April 2017.

[66] David E Salguero. Trajectory analysis and optimization system (taos) users manual. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1995.

[67] Kevin W. Cassel. *Variational Methods with Applications in Science and Engineering*. Cambridge University Press, 2013.

[68] Vladimir G. Boltyanskii, Revaz V. Gamkrelidze, and Lev S. Pontryagin. Towards a theory of optimal processes. *Proceedings of the USSR Academy of Sciences*, 110(1):7–10, 1956.

[69] Lev S. Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[70] Rainer Schöpf and Peter Deuflhard. *OCCAL A Mixed Symbolic-Numeric Optimal Control CALculator*, pages 269–278. Birkhäuser Basel, Basel, 1994.

[71] Enrico Bertolazzi, Francesco Biral, and Mauro Da Lio. Symbolic-numeric efficient solution of optimal control problems for multibody systems. *Journal of computational and applied mathematics*, 185(2):404–421, 2006.

[72] Michael J Grant and Robert D Braun. Rapid indirect trajectory optimization for conceptual design of hypersonic missions. *Journal of Spacecraft and Rockets*, 52(1):177–182, 2014.

[73] Francesco Biral, Enrico Bertolazzi, and Paolo Bosetti. Notes on numerical methods for solving optimal control problems. *IEEJ Journal of Industry Applications*, 5(2):154–166, 2016.

[74] Thomas Antony. *Large Scale Constrained Trajectory Optimization Using Indirect Methods*. PhD thesis, Purdue University, 2018.

[75] Maple. *version 2016*. Waterloo Maple Inc., http://www.maplesoft.com, 2016.

[76] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[77] Kasper Peeters. Cadabra2: computer algebra for field theory revisited. *Journal of open source software.*, 3(32):1118, 2018.

[78] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[79] Inc. Wolfram Research. Mathematica., 2019.

[80] Maxima. Maxima, a computer algebra system. version 5.34.1, 2014.

[81] W. A. Stein et al. *Sage Mathematics Software (Version 8.8)*. The Sage Development Team, 2019. `http://www.sagemath.org`.

[82] Kshitij Mall and Michael J Grant. Trigonomerization of optimal control problems with bounded controls. In *AIAA Atmospheric Flight Mechanics Conference*, page 3244, 2016.

[83] Kshitij Mall and Michael J Grant. Epsilon-trig regularization method for bang-bang optimal control problems. *Journal of Optimization Theory and Applications*, 174(2):500–517, 2017.

[84] Kshitij Mall. *Advancing Optimal Control Theory Using Trigonometry For Solving Complex Aerospace Problems*. PhD thesis, Purdue University, 2019.

[85] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, Dec 1992.

[86] Hans Joachim Oberle and Werner Grimm. *BNDSCO: a program for the numerical solution of optimal control problems*. Inst. für Angewandte Math. der Univ. Hamburg Germany, 2001.

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

[88] I Michael Ross and Fariba Fahroo. A perspective on methods for trajectory optimization. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 4727, 2002.

[89] Frigyes Riesz. Sur une espèce de géométrie analytique des systèmes de fonctions sommables. 144:1409–1411, 1907.

[90] Frigyes Riesz. Sur les opérations fonctionnelles linéaires. *CR Acad. Sci. Paris*, 149:974–977, 1909.

[91] Maurice Fréchet. Sur les ensembles de fonctions et les opérations linéaires. *CR Acad. Sci. Paris*, 144:1414–1416, 1907.

[92] William W Hager. Runge-kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2):247–282, 2000.

[93] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of operations research*, 37(1):357–373, 1992.

[94] Joseph Louis Lagrange. *Mécanique analytique*, volume 1. 1788.

[95] Samuel S. M. Wong. *Introductory nuclear physics*, volume 129. Wiley Online Library, 1990.

[96] HM Robbins. A generalized legendre-clebsch condition for the singular cases of optimal control. *IBM Journal of Research and Development*, 11(4):361–372, 1967.

[97] Emmy Noether. Invariant variation problems. *Transport Theory and Statistical Physics*, 1(3):186–207, 1971.

[98] Richard Chace Tolman. *The principles of statistical mechanics*. Courier Corporation, 1979.

[99] Roger W Brockett. Optimal control of the liouville equation. *AMS IP Studies in Advanced Mathematics*, 39:23, 2007.

[100] Jan Bartsch, Alfio Borzì, Francesco Fanelli, and Souvik Roy. A theoretical investigation of brockett's ensemble optimal control problems. *Calculus of Variations and Partial Differential Equations*, 58(5):162, 2019.

[101] William Karush. Minima of functions of several variables with inequalities as side constraints. *M. Sc. Dissertation. Department of Mathematics, University of Chicago*, 1939.

[102] Harold W Kuhn and Albert W Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press*, pages 481–492, 1951.

[103] B. C. Fabien. Parallel collocation solution of constrained optimal control problems. In *2015 European Control Conference (ECC)*, pages 1147–1152, 2015.

[104] Michael J. Sparapany. Towards the real-time application of indirect methods for hypersonic missions. Master's thesis, Purdue University, 2015.

[105] Thomas Antony and Michael J. Grant. Rapid indirect trajectory optimization on highly parallel computing architectures. *Journal of Spacecraft and Rockets*, 54(5):1081–1091, Jun 2017.

[106] T.J. Dekker, W. Hoffmann, and K. Potma. Parallel algorithms for solving large linear systems. *Journal of Computational and Applied Mathematics*, 50(1):221 – 232, 1994.

[107] Zhenbo Wang and Michael J. Grant. Optimization of minimum-time low-thrust transfers using convex programming. *Journal of Spacecraft and Rockets*, pages 1–13, Dec 2017.

[108] Zhenbo Wang and Michael J. Grant. Constrained trajectory optimization for planetary entry via sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 40(10):2603–2615, Jul 2017.

[109] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[110] C. Becchi, A. Rouet, and R. Stora. Renormalization of the abelian higgs-kibble model. *Comm. Math. Phys.*, 42(2):127–162, 1975.

[111] C Becchi, A Rouet, and R Stora. Renormalization of gauge theories. *Annals of Physics*, 98(2):287 – 321, 1976.

[112] I. V. Tyutin. Gauge invariance in field theory and statistical physics in operator formalism, 2008.

[113] Glenn Barnich, Friedemann Brandt, and Marc Henneaux. Local brst cohomology in gauge theories. *Physics Reports*, 338(5):439 – 569, 2000.

[114] I.A. Batalin and G.A. Vilkovisky. Gauge algebra and quantization. *Physics Letters B*, 102(1):27 – 31, 1981.

[115] I. A. Batalin and G. A. Vilkovisky. Quantization of gauge theories with linearly dependent generators. *Phys. Rev. D*, 28:2567–2582, Nov 1983.

[116] I. A. Batalin and G. A. Vilkovisky. Existence theorem for gauge algebra. *Journal of Mathematical Physics*, 26(1):172–184, 1985.

[117] I.A. Batalin and E.S. Fradkin. A generalized canonical formalism and quantization of reducible gauge theories. *Physics Letters B*, 122(2):157 – 164, 1983.

[118] Jerrold Marsden and Alan Weinstein. Reduction of symplectic manifolds with symmetry. *Reports on mathematical physics*, 5(1):121–130, 1974.

[119] Kenneth R Meyer. Symmetries and integrals in mechanics. *Dynamical systems*, pages 259–273, 1973.

[120] Hernán Cendra, Jerrold E Marsden, Sergey Pekarsky, and Tudor S Ratiu. Variational principles for lie-poisson and hamilton-poincaré equations. *Mosc. Math. J.*, 3(CAG-ARTICLE-2003-004):833–867, 2003.

[121] Perinkulam S Krishnaprasad. Optimal control and poisson reduction. Technical report, University of Maryland College Park Institute For Systems Research, 1993.

[122] Jim Stasheff et al. Homological reduction of constrained poisson algebras. *Journal of Differential Geometry*, 45(1):221–240, 1997.

[123] Joan Lasenby, Anthony N. Lasenby, and Chris J. L. Doran. A unified mathematical language for physics and engineering in the 21st century. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 358(1765):21–39, 2000.

[124] Gérard A Maugin. A historical perspective of generalized continuum mechanics. In *Mechanics of Generalized Continua*, pages 3–19. Springer, 2011.

[125] Rutherford Aris. *Vectors, tensors and the basic equations of fluid mechanics*. Courier Corporation, 2012.

[126] Pei Chi Chou and Nicholas J Pagano. *Elasticity: tensor, dyadic, and engineering approaches*. Courier Corporation, 1992.

[127] John David Jackson. Classical electrodynamics, 1999.

[128] Gregorio Ricci-Curbastro and Tullio Levi-Civita. Méthodes de calcul différentiel absolu et leurs applications. *Mathematische Annalen*, 54(1-2):125–201, 1900.

[129] Hermann Günther Graßmann. *Die Wissenschaft der extensiven Grösse, oder die Ausdehnungslehre, eine neue mathematische Disciplin, etc.* Otto Wigand, 1844.

[130] Viktor Vasil_evich Prasolov. *Problems and theorems in linear algebra*, volume 134. American Mathematical Society, 1994.

[131] Michael J Crowe. *A history of vector analysis: The evolution of the idea of a vectorial system*. Courier Corporation, 1994.

[132] Thomas Hawkins. *Emergence of the theory of Lie groups: An essay in the history of mathematics 1869–1926*. Springer Science & Business Media, 2012.

[133] Joseph Fels Ritt. *Differential algebra*, volume 33. American Mathematical Soc., 1950.

[134] Frederic P. Schuller. *Lecture notes from the Central Lecture Course*. The WE-Heraeus International Winter School on Gravity and Light, https://gravity-and-light.herokuapp.com/, February 2015.

[135] Frederic P. Schuller. *Lectures on the Geometric Anatomy of Theoretical Physics*. Institute for Quantum Gravity, Friedrich-Alexander Universität Erlangen-Nürnberg, September 2015.

[136] Vladimir Igorevich Arnol'd. *Mathematical methods of classical mechanics*, volume 60. Springer Science & Business Media, 2013.

[137] Stephanie Frank Singer. Symmetry in mechanics: A gentle, modern introduction. *Meccanica*, 36(3):323–324, 2001.

[138] Paulette Libermann. On liouville forms. *Banach Center Publications*, 51(1):151–164, 2000.

[139] A Polishchuk. Algebraic geometry of poisson brackets. *Journal of Mathematical Sciences*, 84(5):1413–1444, 1997.

[140] Thomas Antony, Michael J. Grant, Michael J. Sparapany, Sean M. Nolan, Justin Mansell, and Kshitij. Mall. beluga. `https://github.com/Rapid-Design-of-Systems-Laboratory/beluga`, 2019.

[141] Thomas Antony and Michael J Grant. Path constraint regularization in optimal control problems using saturation functions. In *2018 AIAA Atmospheric Flight Mechanics Conference*, page 0018, 2018.

[142] Justin R Mansell and Michael J Grant. Adaptive continuation strategy for indirect hypersonic trajectory optimization. *Journal of Spacecraft and Rockets*, 55(4):818–828, 2018.

[143] Sean M Nolan. Navigation based path planning by optimal control theory. Master's thesis, Purdue University, 2018.

[144] Thomas Antony and Michael J Grant. Quasilinear chebyshev-picard iteration method for indirect trajectory optimization. In *AIAA Scitech 2019 Forum*, page 0260, 2019.

[145] Gerard Meurant. *Differential manifolds and theoretical physics*, volume 116. Academic Press, 1985.

[146] Elena Celledoni, Håkon Marthinsen, and Brynjulf Owren. An introduction to lie group integrators–basics, new developments and applications. *Journal of Computational Physics*, 257:1040–1061, 2014.

[147] Monique Chyba, Ernst Hairer, and Gilles Vilmart. The role of symplectic integrators in optimal control. *Optimal control applications and methods*, 30(4):367–382, 2009.

[148] Melvin Leok. An overview of lie group variational integrators and their applications to optimal control. In *International Conference on Scientific Computation and Differential Equations*, page 1, 2007.

[149] Perinkulam S Krishnaprasad. Optimal control and poisson reduction. Technical report, DTIC Document, 1993.

[150] Eduardo Martínez. Reduction in optimal control theory. *Reports on Mathematical Physics*, 53(1):79–90, 2004.

[151] Tomoki Ohsawa. Symmetry reduction of optimal control systems and principal connections. *SIAM Journal on Control and Optimization*, 51(1):96–120, 2013.

[152] James R Bunch. A note on the stable decompostion of skew-symmetric matrices. *Mathematics of Computation*, 38(158):475–479, 1982.

[153] Froilán M Dopico and Charles R Johnson. Parametrization of the matrix symplectic group and applications. *SIAM Journal on Matrix Analysis and Applications*, 31(2):650–673, 2009.

[154] Judith M Arms, Richard H Cushman, and Mark J Gotay. A universal reduction procedure for hamiltonian group actions. In *The geometry of Hamiltonian systems*, pages 33–51. Springer, 1991.

[155] Frans Cantrijn, José F Carinena, Mike Crampin, and LA Ibort. Reduction of degenerate lagrangian systems. *Journal of Geometry and Physics*, 3(3):353–400, 1986.

[156] Paulo R Rodrigues. Reduction of degenerate non-autonomous lagrangians. In *Mathematical Aspects of Classical Field Theory: Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference Held July 20-26, 1991, with Support from the National Science Foundation*, volume 132, page 275. American Mathematical Soc., 1992.

[157] Claude Albert. Le théoreme de réduction de marsden-weinstein en géométrie cosymplectique et de contact. *Journal of Geometry and Physics*, 6(4):627–649, 1989.

[158] A Echeverría-Enríquez, Miguel C Muñoz-Lecanda, and N Román-Roy. Reduction of presymplectic manifolds with symmetry. *Reviews in Mathematical Physics*, 11(10):1209–1247, 1999.

[159] Larry Bates et al. Nonholonomic reduction. *Reports on Mathematical Physics*, 32(1):99–115, 1993.

[160] Anthony M Bloch, PS Krishnaprasad, Jerrold E Marsden, and Richard M Murray. Nonholonomic mechanical systems with symmetry. *Archive for Rational Mechanics and Analysis*, 136(1):21–99, 1996.

[161] Frans Cantrijn, Manuel De León, Juan Carlos Marrero, and David Martín De Diego. Reduction of nonholonomic mechanical systems with symmetries. *Reports on mathematical Physics*, 42(1-2):25–45, 1998.

[162] Charles-Michel Marle. Reduction of constrained mechanical systems and stability of relative equilibria. *Communications in Mathematical Physics*, 174(2):295–318, 1995.

[163] PS Krishnaprasad, R Yang, and WP Dayawansa. Control problems on principal bundles and nonholonomic mechanics. In *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1133–1138. IEEE, 1991.

[164] Anthony M Bloch and Peter E Crouch. Optimal control, optimization, and analytical mechanics. In *Mathematical control theory*, pages 268–321. Springer, 1999.

[165] A Echeverría-Enríquez, J Marín-Solano, MC Munoz-Lecanda, and N Román-Roy. Geometric reduction in optimal control theory with symmetries. *arXiv preprint math-ph/0206036*, 2002.

[166] Sonia Martínez, Jorge Cortés, and Manuel De León. Symmetries in vakonomic dynamics: applications to optimal control. *Journal of Geometry and Physics*, 38(3-4):343–365, 2001.

[167] H Sussmann. Symmetries and integrals of motion in optimal control. *Banach Center Publications*, 32(1):379–393, 1995.

[168] AJ Van der Schaft. Symmetries in optimal control. *SIAM Journal on Control and Optimization*, 25(2):245–259, 1987.

[169] Michal Jozwikowski. Optimal control theory on almost-lie algebroids. *arXiv preprint arXiv:1111.1549*, 2011.

[170] Alexandr Sergeevich Mishchenko and Anatoly Timofeevich Fomenko. Generalized liouville method of integration of hamiltonian systems. *Functional analysis and its applications*, 12(2):113–121, 1978.

[171] Jerrold Marsden and Alan Weinstein. Reduction of symplectic manifolds with symmetry. *Reports on mathematical physics*, 5(1):121–130, 1974.

[172] Kenneth R Meyer. Symmetries and integrals in mechanics. *Dynamical systems*, pages 259–273, 1973.

[173] Elena Celledoni and Arieh Iserles. Approximating the exponential from a lie algebra to a lie group. *Mathematics of Computation*, 69(232):1457–1480, 2000.

[174] Juan-Pablo Ortega and Tudor S Ratiu. *Momentum maps and Hamiltonian reduction*, volume 222. Springer Science & Business Media, 2013.

[175] Elie Cartan. La théorie des groupes finis et continus et l'analysis situs. 1952.

[176] Jerrold E Marsden and Tudor Ratiu. Reduction of poisson manifolds. *Letters in Mathematical Physics*, 11(2):161–169, 1986.

[177] Michael J Sparapany and Michael Grant. Numerical algorithms for solving boundary-value problems on reduced dimensional manifolds. In *AIAA Aviation 2019 Forum*, page 3666, 2019.

[178] Gene H Golub and John H Welsch. Calculation of gauss quadrature rules. *Mathematics of computation*, 23(106):221–230, 1969.

[179] John R Rice. Parallel algorithms for adaptive quadrature-convergence. In *Information processing 74 : proceedings of IFIP Congress 74*, 1974.

[180] John R Rice. Parallel algorithms for adaptive quadrature ii metalgorithm correctness. *Acta Informatica*, 5(4):273–285, 1975.

[181] John R Rice. Parallel algorithms for adaptive quadrature. iii. program correctness. *ACM Transactions on Mathematical Software (TOMS)*, 2(1):1–30, 1976.

[182] James M Lemme and John R Rice. Speedup in parallel algorithms for adaptive quadrature. *Journal of the ACM (JACM)*, 26(1):65–71, 1979.

[183] Alan Genz. The numerical evaluation of multiple integrals on parallel computers. In *Numerical integration*, pages 219–229. Springer, 1987.

[184] Jarle Berntsen and Terje O Espelid. A parallel global adaptive quadrature algorithm for hypercubes. *Parallel Computing*, 8(1-3):313–323, 1988.

[185] David H Bailey and Jonathan M Borwein. Highly parallel, high-precision numerical integration. 2005.

[186] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[187] David D Morrison, James D Riley, and John F Zancanaro. Multiple shooting method for two-point boundary value problems. *Communications of the ACM*, 5(12):613–614, 1962.

[188] Nguyen X Vinh, Adolf Busemann, and Robert D Culp. Hypersonic and planetary entry flight mechanics. *NASA STI/Recon Technical Report A*, 81, 1980.

[189] Arthur E Bryson Jr, Mukund N Desai, and William C Hoffman. Energy-state approximation in performance optimization of supersonic aircraft. *Journal of Aircraft*, 6(6):481–488, 1969.

[190] David Benson. *A Gauss pseudospectral transcription for optimal control*. PhD thesis, Massachusetts Institute of Technology, 2005.

[191] Audrey Hermant. Optimal control of the atmospheric reentry of a space shuttle by an homotopy method. *Optimal Control Applications and Methods*, 32(6):627–646, 2011.

[192] Steven W Evans and Greg A Dukeman. Examination of a practical aerobraking guidance algorithm. *Journal of Guidance, Control, and Dynamics*, 18(3):471–477, 1995.

[193] H Gardner Moyer. Integrals for optimal flight over a spherical earth. *AIAA Journal*, 11(10):1441–1443, 1973.

[194] Hernán Cendra, Jerrold E Marsden, and Tudor S Raţiu. *Lagrangian reduction by stages*. Number 152. American Mathematical Soc., 2001.

[195] Jerrold E Marsden, George W Patrick, and Steve Shkoller. Multisymplectic geometry, variational integrators, and nonlinear pdes. *Communications in Mathematical Physics*, 199(2):351–395, 1998.

[196] Zhanhua Ma and Clarence W Rowley. Lie-poisson integrators: A hamiltonian, variational approach. *International journal for numerical methods in engineering*, 82(13):1609–1644, 2010.

[197] Robert I McLachlan, Klas Modin, and Olivier Verdier. Collective lie–poisson integrators on 3. *IMA Journal of Numerical Analysis*, 35(2):546–560, 2014.

[198] Adrián J Lew and Pablo Mata. A brief introduction to variational integrators. In *Structure-preserving Integrators in Nonlinear Structural Dynamics and Flexible Multibody Dynamics*, pages 201–291. Springer, 2016.

[199] Klas Modin and Milo Viviani. Lie-poisson methods for isospectral flows. *arXiv preprint arXiv:1808.02827*, 2018.

[200] Haijun Peng, Qiang Gao, Zhigang Wu, and Wanxie Zhong. Symplectic approaches for solving two-point boundary-value problems. *Journal of Guidance, Control, and Dynamics*, 35(2):653–659, 2012.

[201] Robert I McLachlan and Christian Offen. Symplectic integration of boundary value problems. *Numerical Algorithms*, 81(4):1219–1233, 2019.

[202] Rui Loja Fernandes, Juan-Pablo Ortega, and Tudor S Ratiu. The momentum map in poisson geometry. *American journal of mathematics*, 131(5):1261–1310, 2009.

[203] Paul S. Mostert. On a compact lie group acting on a manifold. 65(3):447–455, 1957.

[204] Mike R Osborne. On shooting methods for boundary value problems. *Journal of mathematical analysis and applications*, 27(2):417–433, 1969.

[205] Francesca Mazzia and Jeff R Cash. A fortran test set for boundary value problem solvers. In *AIP Conference Proceedings*, volume 1648, page 020009. AIP Publishing, 2015.

[206] Marilyn Bohl and Maria Rynn. *Tools For Structured and Object-Oriented Design*. Prentice Hall Press, Upper Saddle River, NJ, USA, 7th edition, 2007.

[207] François-Nicola Demers and Jacques Malenfant. Reflection in logic, functional and object-oriented programming: a short comparative study. In *Proceedings of the IJCAI*, volume 95, pages 29–38, 1995.

[208] Harish Saranathan and Michael J Grant. The relaxed autonomously switched hybrid system (rashs) approach to indirect multi-phase trajectory optimization for aerospace vehicles. In *2018 AIAA Atmospheric Flight Mechanics Conference*, page 0016, 2018.

[209] Samuel Eilenberg and Saunders MacLane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(2):231–294, 1945.

[210] David G Hull. *Optimal control theory for applications*. Springer Science & Business Media, 2013.

[211] Harish Saranathan. *Algorithmic Advances to Increase the Fidelity of Conceptual Hypersonic Mission Design*. PhD thesis, Purdue University, 2018.

[212] Stephen L Campbell and B Leimkuhler. Differentiation of constraints in differential-algebraic equations∗. *Journal of Structural Mechanics*, 19(1):19–39, 1991.

[213] Kathryn Eleda Brenan, Stephen L Campbell, and Linda Ruth Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14. Siam, 1996.

[214] Michael J. Sparapany and Michael J. Grant. *The Geometric Adjoining of Optimal Information in Indirect Trajectory Optimization*. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2018.

[215] X Li. Direct solvers for sparse matrices. In *http://crd.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf*. Citeseer, 2006.

[216] James W Demmel, Stanley C Eisenstat, John R Gilbert, Xiaoye S Li, and Joseph WH Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, 1999.

PUBLICATIONS

Upcoming Publications

1. Michael J. Sparapany, Sean M. Nolan, and Michael J. Grant. Reduction of Hamiltonian Systems with Involutory Functions. In *2020 IEEE Aerospace Conference*, March 2020
2. Michael J. Sparapany and Michael J. Grant. Trajectory Reduced-Order Modeling Using Indirect Methods. In *2020 IEEE Aerospace Conference*, March 2020
3. Journal submission to PeerJ Computer Science about `beluga`
4. Journal submission to AIAA Technical Notes about symplectic Integrated Control Regularization Method

Relevant Publications

1. Michael J. Sparapany and Michael Grant. Numerical Algorithms for Solving Boundary-Value Problems of Reduced Dimensional Manifolds. In *AIAA Aviation 2019 Forum*, June 2019.
2. Michael J. Sparapany and Michael J. Grant. The Geometric Adjoining of Optimal Information in Indirect Trajectory Optimization. In *2018 AIAA Guidance, Navigation, and Control Conference*, January 2018

Additional Publications

1. Michael Sparapany, Thomas Antony, Harish Saranathan, Lorenz Klug, Ben Libben, Eiji Shibata, Joseph Williams, Michael J. Grant, and Sarag J. Saikia. Enabling Mars Exploration Using inflatable Purdue Aerodynamic Decelerator with Deployable Entry Systems (iPADDLES) Technology. In *13th International Planetary Probe Workshop*, June 2016

2. Michael A. Bolender, Michael J. Grant, and Michael J. Sparapany. A Homotopy and Parallelization Approach for Improving the Solution Time of Hypersonic Footprints. In *20th AIAA International Space Planes and Hypersonic Systems and Technologies Conference*, July 2015