January 2015

Towards the Real-Time Application of Indirect Methods for Hypersonic Missions

Michael Sparapany
Purdue University

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Michael J. Sparapany

Entitled
Towards the Real-Time Application of Indirect Methods for Hypersonic Missions

For the degree of Master of Science in Aeronautics and Astronautics

Is approved by the final examining committee:

Michael J. Grant Chair	
James M. Longuski Co-chair	
William A. Crossley Co-chair	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Michael J. Grant

Approved by: Weinong Chen 12/3/2015
Head of the Departmental Graduate Program Date

TOWARDS THE REAL-TIME APPLICATION OF INDIRECT METHODS FOR HYPERSONIC MISSIONS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Michael J. Sparapany

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Aeronautics and Astronautics

December 2015

Purdue University

West Lafayette, Indiana

For Mom, Dad, Nicki, and Haley.

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 Haley Snell 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 teaching 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 were vital to the foundation of my research. 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 for his valuable insight, collaboration, and recommendations in regards to this research topic. Kshitij Mall, Harish Saranathan, Zhenbo Wang, Joseph Williams, and everyone else in my research group should also be accredited for their contributions and constructive discussions on my work.

Finally, I would like to acknowledge the Air Force Research Laboratory at Eglin AFB and Universities Space Research Association for providing financial backing through the Air Armament Scholars Program. 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, Jim Whitford, Rick Womack, and Patrick La Petina for their continued support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

C_D	Drag coefficient
C_L	Lift coefficient
c	Constant of motion
$\mathscr{D}_{\mathscr{H}}$	Hamiltonian vector field of $\mathscr{H}$
D	Drag, N
d	Exterior derivative
$\bar{g}$	Boundary conditions
$\mathscr{H}$	Hamiltonian
H	Scale height, m
h	Vehicle height, m
$\bar{\bar{I}}_N$	$N \times N$ Identity matrix
$\mathscr{J}$	Jacobian matrix
J	Cost functional
J'	Augmented cost functional
$\pounds$	Lie derivative
$\mathscr{L}$	Lagrangian
L	Lift, N
m	Vehicle mass, kg
$\bar{p}$	Scalar parameters
r	Radial position, m
$\bar{u}$	Control variables
v	Velocity, m/s
$\bar{X}$	Augmented state vector
$\bar{x}$	State vector
α	Angle-of-attack, rad

γ	Flight path angle, rad
δ	Variational operator
ϵ	Boundary condition residual
ζ	Transformation
$\bar{\bar{\Phi}}$	State Transition Matrix
$\bar{\Phi}$	Initial constraint vector
ϕ	Terminal cost and Latitude, rad
η	Neural network output
θ	Longitude, rad
μ	Gravitational parameter, $\mathrm{m}^3/\mathrm{s}^2$
ν	Lagrange multiplier of adjoined constraint
ξ	Bank angle, rad
$\bar{\Psi}$	Terminal constraint vector
ψ	Heading, rad
ω	Differential 2-form

ABBREVIATIONS

ANN	Artificial Neural Network
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
EOM	Equation of Motion
FEM	Finite Element Model
FPA	Flight-Path Angle
GPOPS	General Pseudospectral Optimization Software
GPU	Graphical Processing Unit
MSM	Multiple Shooting Method
NLP	Nonlinear Programming
ODE	Ordinary Differential Equation
POST	Program to Optimize Simulated Trajectories
RK4	Runge-Kutta 4th Order
SSM	Single Shooting Method
STM	State Transition Matrix

ABSTRACT

Sparapany, Michael J. M.S.A.A, Purdue University, December 2015. Towards the Real-Time Application of Indirect Methods for Hypersonic Missions . Major Professor: Michael J. Grant.

Conceptual hypersonic mission design has typically been performed in a computationally intensive, iterative manner using direct optimization methods. The introduction of modern computing has resulted in the widespread adoption of direct methods, and useful information associated with optimal solutions has been lost. Optimization through indirect methods leverages this information, yielding high quality trajectories while reducing the dimensionality of the overall problem.

The amount of content that can fit on a single chip is approaching physical limitations, resulting in state-of-the-art systems to use more chips. Due to this, present day computational systems are transitioning towards massively parallel frameworks thus creating a need for parallel algorithms to make effective use of available resources. The Multiple Shooting Method provides an effective means of constructing indirect solutions for hypersonic systems using parallel computational architectures. For systems with complex dynamics, it is expected that the chips will become fully saturated with computations, providing performance increases over the serial counterpart.

One restriction to performing optimization using indirect methods is the requirement of high quality initial guesses that must be sufficiently close to a solution for convergence. Sophisticated nonlinear prediction models are used to overcome this limitation. Dimension reductions are performed using Noether's First Theorem with a generalization to Hamiltonian systems. A surrogate model is used to test and validate the outputs of the nonlinear prediction model are high-quality, thus increasing confidence in the constructed initial guess.

The combination of parallel processing with generated high-quality initial guesses is shown to reduce the time to obtain a solution as well as increase the confidence that convergence to a solution will be obtained. Both these criteria must be known to perform real-time hypersonic optimization on-board a vehicle.

1. INTRODUCTION

Since the invention of the first transistor in 1947 [1], steady progress has been made in device miniaturization resulting in more content per chip [2]. This progress was accurately predicted using the well-known Moore's Law [3, 4], though like most extrapolation techniques based on a small sample size, Moore's Law does not model the fundamental limitations of silicon technology and physics [5]. Because of this, CPU clock rates are approaching a plateau, and it is not expected that clock rates can keep increasing at the same rate as they historically have been. Though some techniques exist to increase performance out of a single CPU core, such as Hyper-Threading [6, 7], current state-of-the-art computational architectures are moving towards massively parallel systems [8]. New algorithms are required to make effective use of these massively parallel computational architectures while retaining all of the benefits of the previously mentioned methods.

Direct methods are widely used techniques for solving optimal control problems. In general, this is performed by discretizing a trajectory into several finite segments parameterized by nodes containing state information. Various parameter optimization techniques are used to optimize the nodes of the trajectory while satisfying a set of constraints, including initial, terminal, and path constraints. Newton's Second Law of Motion, or the dynamic constraints of the vehicle, are adjoined as path constraints to ensure the trajectory is physically realizable. Solving the defined optimization problem requires an iterative process that will satisfy the Karush-Kuhn-Tucker conditions [15,16] within tolerance. Industry and government standard programs such as NASA's POST [9], DIDO [10], and GPOPS [11, 12] utilize sophisticated algorithms that incorporate collocation [13] and pseudo-spectral [14] methods.

Indirect methods are based on the Calculus of Variations [17] and Pontryagin's Minimum Principle [18,19]. Pontryagin's Minimum Principle states that for a system to be at a local optimum, the Hamiltonian [20] must lie on an extremum over the set of admissible controls. The Hamiltonian is formulated by adjoining the equations of motion to the cost functional with Lagrange multipliers (co-states). The Lagrange multipliers have their own equations of motion by definition of the Hamiltonian. Integral curves in the defined vector space represent optimal trajectories given that the necessary conditions of optimality are derived from Pontryagin's Minimum Principle and the Hamiltonian. This process converts a continuous optimization problem into a finite dimensional boundary valued root-solving problem while retaining all original numerical information.

While it is evident that indirect methods benefit by making use of known analytical quantities that are sometimes ignored with direct methods, other mathematical quantities are often ignored in optimal control problems. Noether's First Theorem, published in 1918, has been an extraordinary benefit for scientists and researchers due to its practicality and insight it provides into physical systems. Multiple versions exist for practical application in any scenario derived by many different researchers. The original published version was a generalization applicable to a multitude of situations. Emmy Noether considered groups of global symmetries as well as their infinitesimal generating functions whereas physicists of the time largely ignored the work of Sophus Lie and his contributions to continuous symmetry [21]. Emmy Noether also used concepts from Calculus of Variations in her publication long before the great contributions of Lev Pontryagin and his students to the field. Typically applicable to Lagrangian systems, formulations of Noether's First Theorem have been presented for Hamiltonian systems and optimal control problems as well [22]. This generalization presents a much more compact and comprehensive approach to determining symmetries and conserved quantities encompassing the results in a Lagrangian system as well [23].

Optimization methods, both direct and indirect, focused on an inherently serial process for formulating numerical solutions. Considering state-of-the-art computational frameworks are transitioning towards parallel systems, there is a need for more algorithms to be employed in parallel [24]. Trajectory optimization problems are a subset of optimization dealing with infinite dimensional problems, requiring functional optimization. In the case of very high speed vehicles, typically traveling as speeds over Mach 5, computer guidance algorithms control the vehicle. Practical applications include modeling the entry phase of orbital vehicles and designing hypersonic vehicles to fit a defined mission. Historically, hypersonic trajectory optimization has been performed using direct methods due to three main challenges associated with indirect methods.

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.

The difficulties associated with solving optimization problems in this manner have been overcome through modern symbolic manipulation tools and homotopy continuation methods [25,26]. This process has also been demonstrated with recent hypersonic applications [27].

Real-time and on-board applications of optimization have been studied previously. The authors of Ref. 28 and Ref. 29 present a hybrid analytical and numerical approach to solving the constrained trajectory of an ascent vehicle and also comment on it's feasibility for on-line usage. Guidance is typically solved in an open-loop manner for atmospheric flight and in a closed-loop manner when the vehicle is in a vacuum. When a vehicle is flying in a vacuum, there are either analytic or near analytic solutions that exist for the guidance problem. The authors of Ref. 28 cite costly launch delays as one motivation for investigating on-line optimization of the atmospheric phase. First, an analytical solution is generated for a simplified problem, specifically the vehicle in a vacuum. Because near-analytic solutions exist for exoatmospheric flight [30], an initial guess is rapidly generated for the full problem. Instead of solving the entire atmospheric guidance problem in a single step, the atmosphere model is incorporated through a homotopy method. This method allows the authors to reliably generate solutions to the full problem rapidly.

For real-time hypersonic trajectory optimization to be performed on-board, two criteria must be met by an algorithm.

1. Convergence to a flyable trajectory is guaranteed.

2. Time to converge to a solution is known.

In an effort to mitigate both of these issues, Artifical Neural Networks (ANNs) are employed to construct high-quality initial guesses of optimal trajectories. ANNs are a form of nonlinear curve prediction and pattern recognition inspired by Biological Neural Networks. With the ability to model and reproduce massive amounts of information accurately, ANNs are used in a wide variety of fields such as automotive, defense, entertainment, financial, insurance, manufacturing, robotics, telecommunications, and transportation industries [31]. One goal of this research is to lessen the gap between present day technology and the ability to perform optimization on-board a hypersonic vehicle.

Current state-of-the-art algorithms currently are unable to perform real-time hypersonic trajectory optimization on-board. This is primarily due to the fact that convergence and time to solution are not guaranteed for any one algorithm. The research presented aims to decrease time to solution by generating high-quality initial guesses from a ANN. Contributions are also made in increasing confidence that an ANN will return high-quality guesses through application of Noether's First Theorem and validation of a Taylor series surrogate model, making an effort to leverage as many known analytical quantities as possible.

2. OVERVIEW OF INDIRECT TRAJECTORY OPTIMIZATION

2.1 Calculus of Variations

Calculus of Variations is a field of mathematics that has found use in a wide variety of applications ranging from quantum and classical mechanics to aerospace engineering. The main focus of Variational Calculus is the extrema of functionals. A functional is a mapping from a function to real numbers. One of the earliest and most important problems, the brachistochrone problem, was first posed and solved by John Bernoulli in 1696. Given two points, A and B, in a vertical plane, a frictionless mass placed at point A, and a uniform gravity field, the question is: What is the path the mass should take from point A to point B such that time of travel is minimized? Isaac Newton, Jacob Bernoulli (John's brother), Gottfried Leibnitz, Ehrenfried Walther von Tschirnhaus, and Guillaume de l'Hôpital all submitted solutions for the problem. What appeared to be a simple mathematics problem spawned an entire field of mathematics with a rich history and very important applications.

A general statement for an optimal control problem minimizing path and terminal cost is shown in Equation 2.1. $f(\bar{x}, \bar{u}, t)$, $\bar{\Phi}$, and $\bar{\Psi}$ shown in Equations 2.2-2.4 represent the dynamic, initial, and terminal constraints respectively and J represents a continuous-time cost functional to be minimized. Equation 2.5 refers to the set of admissible controls.

$$\text{minimize } J = \phi(\bar{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathscr{L}(\bar{x}, \bar{u}, t)\text{dt} \tag{2.1}$$

subject to:

$$\dot{\bar{x}} - \bar{f}(\bar{x}, \bar{u}, t) = 0 \tag{2.2}$$

$$\bar{\Phi}(\bar{x}(t_0), t_0) = 0 \tag{2.3}$$

$$\bar{\Psi}(\bar{x}(t_f), t_f) = 0 \tag{2.4}$$

$$\bar{u} \in U \tag{2.5}$$

2.2 Necessary Conditions for Optimality

For a finite dimensional constrained optimization problem, the cost function can be written simply as an analytical function of the input parameters.

$$\text{minimize } J = \phi(\bar{x}, \bar{u}) \tag{2.6}$$

subject to:

$$f(\bar{x}, \bar{u}) = 0 \tag{2.7}$$

Constrained optimization problems formulated this way can be typically solved in a fairly straightforward manner utilizing sophisticated algorithms such as Nelder-Mead, Simulated Annealing, or Genetic Algorithms. Solving an optimization problem in this manner is typically referred to as using a "direct" method. First and second-order necessary conditions are implemented to verify a solution is at a local optimum and are shown in Equations 2.8a and 2.8b. For a solution to be at a local optimum, these conditions must be satisfied. A sufficient condition guarantees a solution is a local minimum and is shown in Equation 2.9.

Necessary Conditions:

$$\frac{\partial J}{\partial \bar{u}} = 0 \tag{2.8a}$$

$$\frac{\partial^2 J}{\partial \bar{u}^2} \geq 0 \tag{2.8b}$$

Sufficient Condition:

$$\frac{\partial^2 J}{\partial \bar{u}^2} > 0 \tag{2.9}$$

The Augmented Multiplier Method introduces more unknown parameters into the system thereby eliminating dependent changes of the cost function. Using this method, also referred to as an "indirect" method, results in an augmented cost function.

$$J' = \phi(\bar{x},\bar{u}) + \lambda f(\bar{x},\bar{u}) \tag{2.10}$$

With new first-order necessary conditions:

$$\frac{\partial J'}{\partial \bar{x}} = \frac{\partial \phi}{\partial \bar{x}} + \lambda \frac{\partial f}{\partial \bar{x}} = 0 \tag{2.11a}$$

$$\frac{\partial J'}{\partial \bar{u}} = \frac{\partial \phi}{\partial \bar{u}} + \lambda \frac{\partial f}{\partial \bar{u}} = 0 \tag{2.11b}$$

$$\frac{\partial J'}{\partial \lambda} = f = 0 \tag{2.11c}$$

This finite dimensional example illustrates the solution process of indirect methods. The optimization of a trajectory is considered to be an infinite dimensional optimization problem where the control variables $\bar{u}$ have a time-history. To incorporate the constraints into Equation 2.1, the Augmented Lagrange Multiplier method is used. The cost functional is augmented and each constraint is adjoined with its own Lagrange multiplier. The dynamic constraints must be satisfied for all time and are included inside the integral. The corresponding Lagrange multipliers are now no longer constants.

$$J' = \phi(\bar{x}(t_f),t_f) + \int_{t_0}^{t_f} \left(\mathscr{L}(\bar{x},\bar{u},t) + \bar{\lambda}^T(\bar{x},\bar{u},t)(\bar{f}(\bar{x},\bar{u},t) - \dot{\bar{x}}(\bar{x},\bar{u},t))\right) \mathrm{dt} \\ + \bar{\nu}_0^T \bar{\Phi}(\bar{x}(t_0),t_0) + \bar{\nu}_f^T \bar{\Psi}(\bar{x}(t_f),t_f) \tag{2.12}$$

To minimize a cost functional J, a necessary condition is for the first-order variation of J to vanish.

$$\delta J' = \delta\phi + \delta \int_{t_0}^{t_f} \left(\mathscr{H} - \bar{\lambda}^T \dot{\bar{x}}\right) \mathrm{dt} + \delta(\bar{\nu}_0^T \bar{\Phi}) + \delta(\bar{\nu}_f^T \bar{\Psi}) \tag{2.13}$$

Using the Leibniz' Rule, the first-order variation is determined inside of the integral [20]. $\mathscr{H}$ is the Hamiltonian of the system defined in Equation 2.14.

$$\mathscr{H} := \mathscr{L} + \bar{\lambda}^T \bar{f} \tag{2.14}$$

$$\delta J' = \delta\phi + \int_{t_0}^{t_f} \left(\frac{\partial \mathscr{H}}{\partial \bar{x}}^T \delta\bar{x} + \frac{\partial \mathscr{H}}{\partial \bar{\lambda}}^T \delta\bar{\lambda} + \frac{\partial \mathscr{H}}{\partial \bar{u}}^T \delta\bar{u} - \dot{\bar{x}}^T \delta\bar{\lambda} - \bar{\lambda}^T \delta\dot{\bar{x}} \right) \mathrm{dt} + \delta(\bar{\nu}_0^T \bar{\Phi}) + \delta(\bar{\nu}_f^T \bar{\Psi}) \quad (2.15)$$

Considering t_0 to be a fixed initial time and t_f to be free, selecting $\bar{\lambda}$ and $\bar{\nu}_i$ such that all $\delta\bar{x}$ vanish results in the formulation of a Two-Point Boundary Valued Problem (TPBVP) listed in Equations 2.16-2.21.

$$\frac{\partial \mathscr{H}}{\partial \bar{u}} = 0 \quad (2.16)$$

$$\dot{\bar{x}}^T = \{\bar{x}, \mathscr{H}\} = \frac{\partial \mathscr{H}}{\partial \bar{\lambda}} \quad (2.17)$$

$$\dot{\bar{\lambda}}^T = \{\bar{\lambda}, \mathscr{H}\} = -\frac{\partial \mathscr{H}}{\partial \bar{x}} \quad (2.18)$$

$$\bar{\lambda}^T(t_0) = -\bar{\nu}_0^T \frac{\partial \bar{\Phi}}{\partial \bar{x}(t_0)} \quad (2.19)$$

$$\bar{\lambda}^T(t_f) = \frac{\partial \phi}{\partial \bar{x}(t_f)} + \bar{\nu}_f^T \frac{\partial \bar{\Psi}}{\partial \bar{x}(t_f)} \quad (2.20)$$

$$\left(\mathscr{H} + \frac{\partial \phi}{\partial t} + \bar{\nu}_f^T \frac{\partial \bar{\Psi}}{\partial t} \right) \Bigg|_{t_f} = 0 \quad (2.21)$$

Where $\{f, g\}$ denotes the Poisson bracket of the functions f and g, defined in Equation 2.22.

$$\{f, g\} = \frac{\partial f}{\partial \bar{x}}^T \frac{\partial g}{\partial \bar{\lambda}} - \frac{\partial f}{\partial \bar{\lambda}}^T \frac{\partial g}{\partial \bar{x}} \quad (2.22)$$

Such that

$$\frac{df(\bar{x}, \bar{\lambda}, t)}{dt} = \frac{\partial f(\bar{x}, \bar{\lambda}, t)}{\partial \bar{x}}^T \frac{\partial \bar{x}}{\partial t} + \frac{\partial f(\bar{x}, \bar{\lambda}, t)}{\partial \bar{\lambda}}^T \frac{\partial \bar{\lambda}}{\partial t} + \frac{\partial f(\bar{x}, \bar{\lambda}, t)}{\partial t} \quad (2.23a)$$

$$\frac{df(\bar{x}, \bar{\lambda}, t)}{dt} = \frac{\partial f(\bar{x}, \bar{\lambda}, t)}{\partial \bar{x}}^T \frac{\partial \mathscr{H}}{\partial \bar{\lambda}} - \frac{\partial f(\bar{x}, \bar{\lambda}, t)}{\partial \bar{\lambda}}^T \frac{\partial \mathscr{H}}{\partial \bar{x}} + \frac{\partial f(\bar{x}, \bar{\lambda}, t)}{\partial t} \quad (2.23b)$$

$$\frac{df(\bar{x}, \bar{\lambda}, t)}{dt} = \{f(\bar{x}, \bar{\lambda}, t), \mathscr{H}\} + \frac{\partial f(\bar{x}, \bar{\lambda}, t)}{\partial t} \quad (2.23c)$$

2.3 Single Shooting Method

One result of using indirect methods is the infinite-dimensional optimization problem defined in Equations 2.1-2.5 is rewritten as a finite-dimensional TPBVP shown in

Equations 2.16-2.21. The main benefit from going through process is that the original optimization problem is now posed as a root-solving problem. Solutions that satisfy these equations have guaranteed optimality. One popular algorithm for root-solving these equations is the Single Shooting Method (SSM) [32]. Figure 2.1 shows how a trajectory is propagated using state estimates in the SSM.

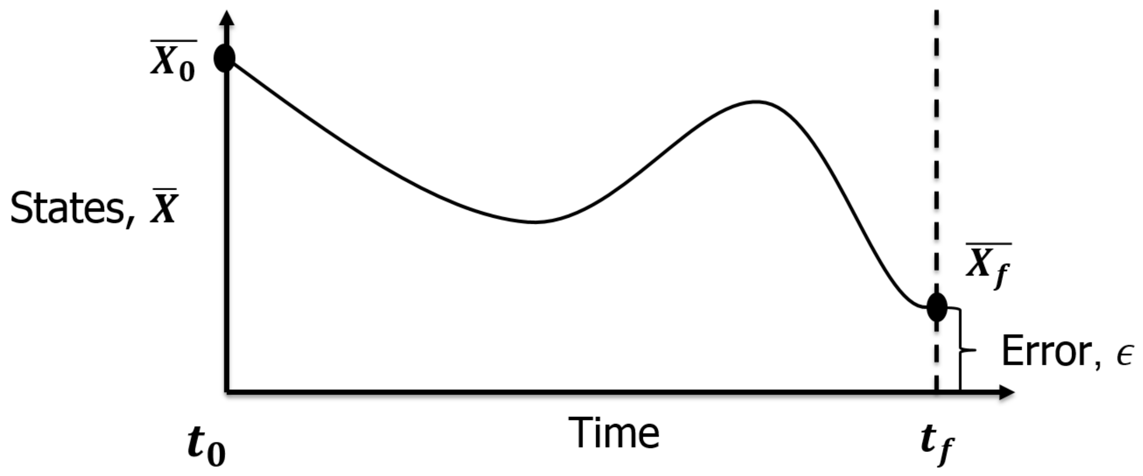


Figure 2.1. A single trajectory with error in terminal state

Consider the augmented state vector ($\bar{X}$) defined in Equation 2.24a. If the initial condition is $\bar{X}_0$ and the final time is t_f, then a complete trajectory may be reconstructed. The terminal state estimate ($\bar{X}_f$) is computed through forward propagation using a Runge-Kutta 4th Order (RK4) or similar integration scheme.

$$\bar{X} = \begin{bmatrix} \bar{x} \\ \\ \bar{\lambda} \end{bmatrix} \tag{2.24a}$$

$$\bar{X}_0 = \bar{X}(t_0) \tag{2.24b}$$

$$\bar{X}_f = \bar{X}_0 + \int_{t_0}^{t_f} \dot{\bar{X}} dt \tag{2.24c}$$

Boundary conditions ($\bar{g}$) are evaluated using estimates for $\bar{X}_0$ and $\bar{X}_f$ as well as estimates for any additional parameters ($\bar{p}$) defined by the trajectory problem. This obtains a residual vector, ϵ. The residual vector is shown in Equation 2.25a and Equation 2.25b shows how a correction vector ($\Delta\bar{X}_0$ and $\Delta\bar{p}$) applied to the initial state and parameters will drive the residual error to zero.

$$\bar{g}(\bar{X}_0, \bar{X}_f, \bar{p}) = \epsilon \tag{2.25a}$$

$$\bar{g}(\bar{X}_0 + \Delta\bar{X}_0, \bar{X}_f, \bar{p} + \Delta\bar{p}) = 0 \tag{2.25b}$$

Due to the nonlinear equations of motion (EOMs), an analytic solution for the correction vector typically can not be found. Instead, the correction vector is computed using a first-order approximation and Jacobian matrices of sensitivities. $\mathscr{J}_M$ and $\mathscr{J}_N$ are Jacobian matrices of the boundary conditions at the initial and terminal states respectively while $\mathscr{J}_p$ is the Jacobian matrix with respect to the parameters.

$$\mathscr{J}_M = \frac{\partial\bar{g}}{(\partial\bar{X}(\tau_0))}, \mathscr{J}_N = \frac{\partial\bar{g}}{(\partial\bar{X}(\tau_f))}, \mathscr{J}_p = \frac{\partial\bar{g}}{\partial\bar{p}} \tag{2.26}$$

Since the SSM only updates the initial conditions of the trajectory, relationships between the error in the terminal boundary conditions and initial correction vector are characterized through the use of a state transition matrix (STM), $\bar{\bar{\Phi}}$. The STM is a matrix that relates state information at some time t_1 to another point in time t_2 through products. Using the STM allows adjustments in the initial state vector to reduce residual error in the terminal state of the system. Equations 2.27a-2.27b define the STM and it is important to note that it has EOMs. The STM and states must be propagated forward in time together.

$$\dot{\bar{\bar{\Phi}}} = \frac{\partial\dot{\bar{X}}}{\partial\bar{X}} \cdot \bar{\bar{\Phi}} \tag{2.27a}$$

$$\bar{\bar{\Phi}}(t_0) = \bar{\bar{I}}_N \tag{2.27b}$$

With the STM and each individual Jacobian matrix, the completed Jacobian matrix for the entire system can be defined and is shown 2.28.

$$\mathscr{J} = [\mathscr{J}_M + \mathscr{J}_N\bar{\bar{\Phi}}, \mathscr{J}_p] \tag{2.28}$$

The correction vector can then be determined by solving the linear system in Equation 2.29.

$$\mathscr{J}\Delta\bar{X}_0 = -\epsilon \tag{2.29}$$

2.4 Multiple Shooting Method

Since a single trajectory is entirely continuous with one set of initial conditions, the entire solution process is serial. The Multiple Shooting Method (MSM) attempts to parallelize the process by allowing the integration to take place in parallel. By introducing discontinuities into the trajectory with corresponding boundary conditions, several independent smaller trajectories are formed each with their own set of initial conditions. Figure 2.2 illustrates how a single trajectory may be separated into smaller arcs, where τ represents the localized time of each independent sub-arc.

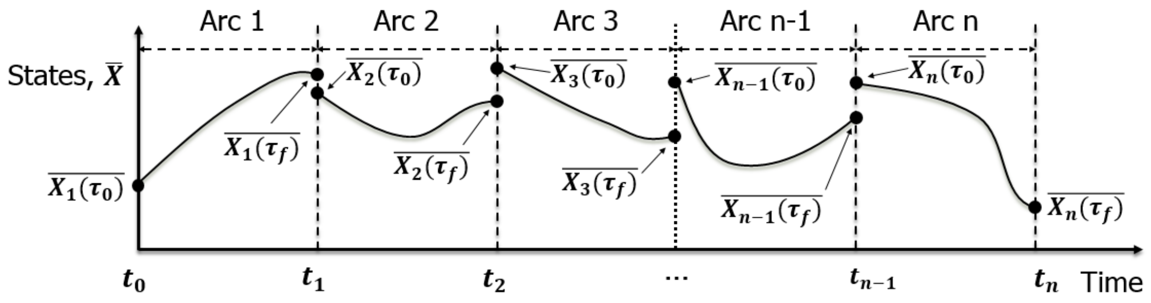


Figure 2.2. Multiple trajectories forming a Multi-Point Boundary Valued Problem

With multiple independent arcs, integration can now take place independently from one another. One issue that arises, however, is how the correction vector must be determined. With additional boundary conditions defined in Equation 2.30 enforcing continuity, new Jacobian matrices must be derived.

$$g = 0 = \bar{X}_i(\tau_0) - \bar{X}_{i-1}(\tau_f)\ i = 2, 3, \cdots, n \tag{2.30}$$

$$\mathscr{J}_{M_i} = \frac{\partial g}{(\partial \bar{X}_i(\tau_0))}, \mathscr{J}_{N_i} = \frac{\partial g}{(\partial \bar{X}_i(\tau_f))}, \mathscr{J}_P = \frac{\partial g}{\partial \bar{p}} \tag{2.31}$$

The new Jacobian matrix for the system is entirely coupled and shown in Equation 2.32.

$$\mathscr{J} = [\mathscr{J}_{M_1} + \mathscr{J}_{N_1}\bar{\bar{\Phi}}_1, \mathscr{J}_{M_2} + \mathscr{J}_{N_2}\bar{\bar{\Phi}}_2, \cdots, \mathscr{J}_{M_n} + \mathscr{J}_{N_n}\bar{\bar{\Phi}}_n, \mathscr{J}_P] \tag{2.32}$$

This allows the entire trajectory to be integrated in parallel, though the correction vector for the system must be determined after integration of each trajectory has been completed. Therefore the overall process will be limited by the slowest core. Even though this causes a slight performance reduction by a required matrix inversion at the end of integration, it enforces additional boundary conditions and provides each individual arc with corrections to its initial conditions so the resulting solution will be continuous.

One issue that commonly arises when solving the linear system in Equation 2.29 is that $\det(\mathscr{J}) = 0$. This produces an error commonly referred to as a "singular Jacobian" and can be caused by a number of issues. These issues include, but are not limited to a trajectory optimization problem with no solution, a poorly scaled problem, and linear dependencies in the Jacobian matrix. Because first order approximations were made, there is no guarantee the Jacobian matrix will be invertible for a nonlinear problem. This presents a problem for real-time applications since convergence must be guaranteed. Instead of directly solving Equation 2.29 by inverting the Jacobian, an optimization problem is posed. Equation 2.33 defines the optimization problem such that the distance between $-\epsilon$ and $\mathscr{J}\Delta\bar{X}_0$ is minimized.

$$\text{minimize } J = \left\|\mathscr{J}\Delta\bar{X}_0 + \epsilon\right\| \tag{2.33}$$

Using the Projection Theorem guarantees this problem has a solution. If $\mathscr{J}$ is one to one, then $(\mathscr{J}^T\mathscr{J})$ is invertible and the solution to the optimization problem is shown in Equation 2.34.

$$\Delta\bar{X}_0 = -(\mathscr{J}^T\mathscr{J})^{-1}\mathscr{J}^T\epsilon \tag{2.34}$$

3. CONSTANTS OF OPTIMAL MOTION

3.1 Constants of Optimal Motion Overview

An expression is considered to be an integral of motion if it is invariant along a trajectory and not a function of the independent variable.

$$I(\bar{X}, \dot{\bar{X}}) = const \tag{3.1}$$

Similarly, an expression is considered to be a constant of motion if it is invariant along a trajectory and also a function of the independent variable.

$$c(\bar{X}, \dot{\bar{X}}, t) = const \tag{3.2}$$

Therefore every integral of motion is also considered a constant of motion, but not every constant of motion is an integral of motion. Knowing constants of a system can be an especially powerful tool as they typically can be used to reduce the dimensionality of a system. Some of the most well-known constants of motion include linear and angular momentum, which are used on a regular basis in Physics and Engineering. These constants of motion allow Engineers and Scientists to analyze systems and make generalizations that otherwise would be numerically difficult or impossible.

3.2 Noether's First Theorem

In 1918, German mathematician Emmy Noether published *Invariante Variationsprobleme* in which she proved what is now an important fundamental tool in mathematics, physics, and optimal control theory [33]. Albert Einstein commented on Emmy Noether's contributions as a mathematician in a letter to the editor of The New York Times [21, 34]:

In the judgment of the most competent living mathematicians, Fräulein Noether was the most significant creative mathematical genius thus far produced since the higher education of women began.

Noether's First Theorem states that for every symmetry of the action functional of a physical system, there exists a corresponding conservation law. Applying Noether's First Theorem to a system can provide a set of first integrals for that system. The Lagrangian formulation of a mechanical system is shown in Equation 3.3 where T is a system's kinetic energy and U is a system's potential energy.

$$\mathscr{L}(\bar{x}, \dot{\bar{x}}, t) = T - U \tag{3.3}$$

The Euler-Lagrange equation defines the system's EOMs and is shown in Equation 3.4.

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{\bar{x}}} = \frac{\partial \mathscr{L}}{\partial \bar{x}} \tag{3.4}$$

Theorem 3.2.1 (Noether's First Theorem [35]) *If the functional*

$$J(\bar{x}) = \int_{t_0}^{t_f} \mathscr{L}(\bar{x}, \dot{\bar{x}}, t)\, dt \tag{3.5}$$

is invariant under the transformations

$$\bar{x} \to \bar{x} + \epsilon\bar{\zeta}_x(\bar{x}, \dot{\bar{x}}, t), \qquad \dot{\bar{x}} \to \dot{\bar{x}} + \epsilon\dot{\bar{\zeta}}_x(\bar{x}, \dot{\bar{x}}, t), \qquad t \to t \tag{3.6}$$

for arbitrary t_0 and t_f and constant infinitesimal ϵ, then there exists a corresponding constant quantity

$$\frac{\partial \mathscr{L}}{\partial \dot{\bar{x}}}^T \bar{\zeta}_x \tag{3.7}$$

Proof For the action functional

$$J(\bar{x}) = \int_{t_0}^{t_f} \mathscr{L}(\bar{x}, \dot{\bar{x}}, t)\, dt \tag{3.8}$$

to be invariant under the transformations listed in Equation 3.6, the change in the Lagrangian upon the transformation must vanish

$$\Delta\mathscr{L} = \mathscr{L}(\bar{x} + \epsilon\bar{\zeta}_x(\bar{x}, \dot{\bar{x}}, t), \dot{\bar{x}} + \epsilon\dot{\bar{\zeta}}_x(\bar{x}, \dot{\bar{x}}, t), t) - \mathscr{L}(\bar{x}, \dot{\bar{x}}, t) = 0 \tag{3.9}$$

For a first order Taylor series expansion on ϵ

$$\frac{\partial \mathscr{L}}{\partial \bar{x}}^T \epsilon \bar{\zeta}_x + \frac{\partial \mathscr{L}}{\partial \dot{\bar{x}}}^T \epsilon \dot{\bar{\zeta}} = 0 \tag{3.10}$$

Which is rewritten using the Euler-Lagrange equations in Equation 3.4

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{\bar{x}}}^T \epsilon \bar{\zeta}_x + \frac{\partial \mathscr{L}}{\partial \dot{\bar{x}}}^T \epsilon \dot{\bar{\zeta}} = 0 \tag{3.11}$$

Grouping similar terms

$$2\epsilon \frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\bar{x}}}^T \bar{\zeta}_x\right) = 0 \tag{3.12}$$

Implying that

$$\frac{\partial \mathscr{L}}{\partial \dot{\bar{x}}}^T \bar{\zeta}_x \tag{3.13}$$

is a constant. ■

3.3 Poisson Bracket Generalization

Equation 2.22 defines the Poisson bracket of a function with the Hamiltonian of a system. If a Hamiltonian is not an explicit function of time, where $\frac{\partial \mathscr{H}}{\partial t} = 0$, then $\{\mathscr{H}, \mathscr{H}\} = 0$ due to the antisymmetry of the Poisson bracket. Noether's First Theorem can be generalized further to Poisson brackets and represented in a more compact and convenient format. Consider the Hamiltonian vector field defined in Equation 3.14 and differential 2-form defined in Equation 3.15.

$$\mathscr{D}_{\mathscr{H}} := \frac{d}{dt} = \frac{\partial \mathscr{H}}{\partial \bar{\lambda}}^T \frac{\partial}{\partial \bar{x}} - \frac{\partial \mathscr{H}}{\partial \bar{x}}^T \frac{\partial}{\partial \bar{\lambda}} = \{\cdot, \mathscr{H}\} \tag{3.14}$$

$$\omega = \sum_{i=1}^{n} dx_i \wedge d\lambda_i \tag{3.15}$$

Where $\wedge$ is the wedge operator or exterior product. The Lie derivative ($\pounds$) of a function f can now be expressed in terms of the Poisson Bracket.

$$\pounds_{\mathscr{D}_g}(f) = df(\mathscr{D}_g) = \mathscr{D}_g(f) = \{f, g\} \tag{3.16}$$

Theorem 3.3.1 (Noether's First Theorem for a Hamiltonian System [36]) *For a continuous symmetry $\mathscr{D}$ of a Hamiltonian system such that $\pounds_{\mathscr{D}}\omega = 0$ and $\pounds_{\mathscr{D}}\mathscr{H} = 0$, there locally exists a constant of motion c such that $\mathscr{D} = \mathscr{D}_c$. Conversely, if c is a constant of motion, then $\mathscr{D}_c$ is a symmetry.*

Simply put, a constant of motion for a system will result in a vanishing quantity $\{c, \mathscr{H}\}$ if and only if the quantity $\{\mathscr{H}, c\}$ vanishes as well. Additionally, through Poisson's Theorem, given two constants c_1 and c_2 for a Hamiltonian $\mathscr{H}$, a third constant may be discovered by evaluating $\{c_1, c_2\}$. This compact formulation is better suited for this class of problems while encompassing more constants than the Lagrangian formulation.

The constants of motion cannot be directly applied to the trajectory optimization problem as it stands. Simply evaluating their first derivative recovers already known differential equations defined by $\{\bar{\lambda}, \mathscr{H}\}$, and they are unique to the trajectories they characterize. For example, the algorithms described in Sections 2.3-2.4 root-solve for the values of the co-state initial conditions, and likewise the constants of motion. At the beginning of the solution process, the initial co-states are unknown so, therefore, the constants are unknown as well. The exception to this is dependent on the constraints placed on the boundary conditions. Through the necessary conditions for optimality, some of these constants can be determined before hand. These constants of motion will later be used in Chapter 4 to make analysis of a Neural Network simpler and more accurate.

4. NEURAL NETWORK INITIAL GUESS

ANNs are a sophisticated form of nonlinear curve prediction and will be used to aid reconstructing entire trajectories. For obvious reasons, using an ANN to directly reconstruct an infinite dimensional trajectory is infeasible. Such a network would require an infinite number of neurons. One key result of solving trajectory optimization problems using indirect methods is that the infinite dimensional optimization problem is posed as a finite dimensional root-solving problem. The entire trajectory becomes characterized by a finite set of parameters and training an ANN to predict this set is possible.

4.1 Overview of Neural Networks

At their lowest level, ANNs are composed of artificial neurons. Figure 4.1 represents a single abstract neuron and its structure.

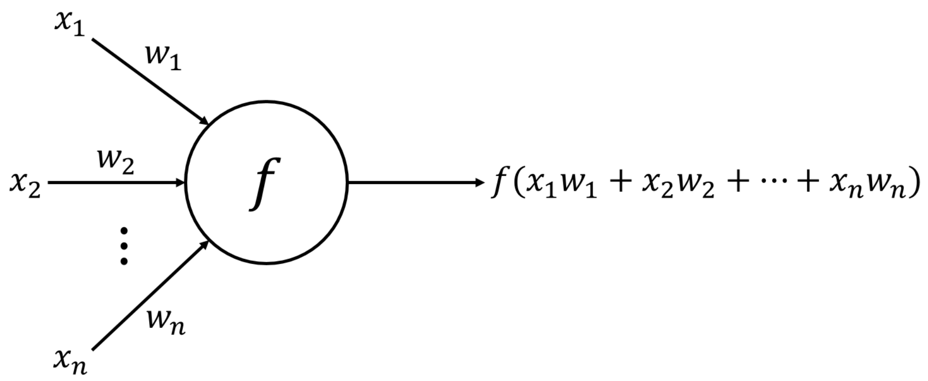


Figure 4.1. An abstract neuron (reproduced from Ref. 31)

x_1 through x_n represent an abstract number of inputs into a neuron while w_1 through w_n represent the weight of each input. The input information is then aggregated at the neuron and passed through an activation function. The activation function is what determines the output of a neuron given the inputs and can be selected to suit the application. Equations 4.1-4.4 and Figure 4.2 present examples of a few basic activation functions along with some various parameters.

$$\text{Step Function} = f(s) = \begin{cases} 0 & \text{if } s < 0 \\ 1 & \text{if } s > 0 \end{cases} \tag{4.1}$$

$$\text{Linear Function} = f(s) = \begin{cases} -1 & \text{if } s < \frac{-1}{\beta} \\ 1 & \text{if } s > \frac{1}{\beta} \\ \beta s & \text{otherwise} \end{cases} \tag{4.2}$$

$$\text{Log-Sigmoid Function} = f(s) = \frac{1}{1 + \exp(-\beta s)} \tag{4.3}$$

$$\text{Tan-Sigmoid Function} = f(s) = \tanh(\beta s) \tag{4.4}$$

The neurons are then arranged into several layers to form a feedforward network of neurons. A user can directly interface with the input and output layers of a Neural Network, while the hidden layers remain unseen. Figure 4.3 represents a feedforward Neural Network with m output variables corresponding to n input variables. With known activation functions, each output η_i can be analytically represented as a function of the input variables x_i.

4.2 Construction of Co-state Predicting Neural Networks

In machine-learning, training a Neural Network is typically organized into 2 distinct tasks: supervised and unsupervised learning. In unsupervised learning, a Neural Network is provided with a set of inputs and no outputs. The Neural Network will then attempt to draw inferences about the input data set without any response data set. In supervised learning, both an input and output data set are provided during the training process. The network will then attempt to infer underlying functions and

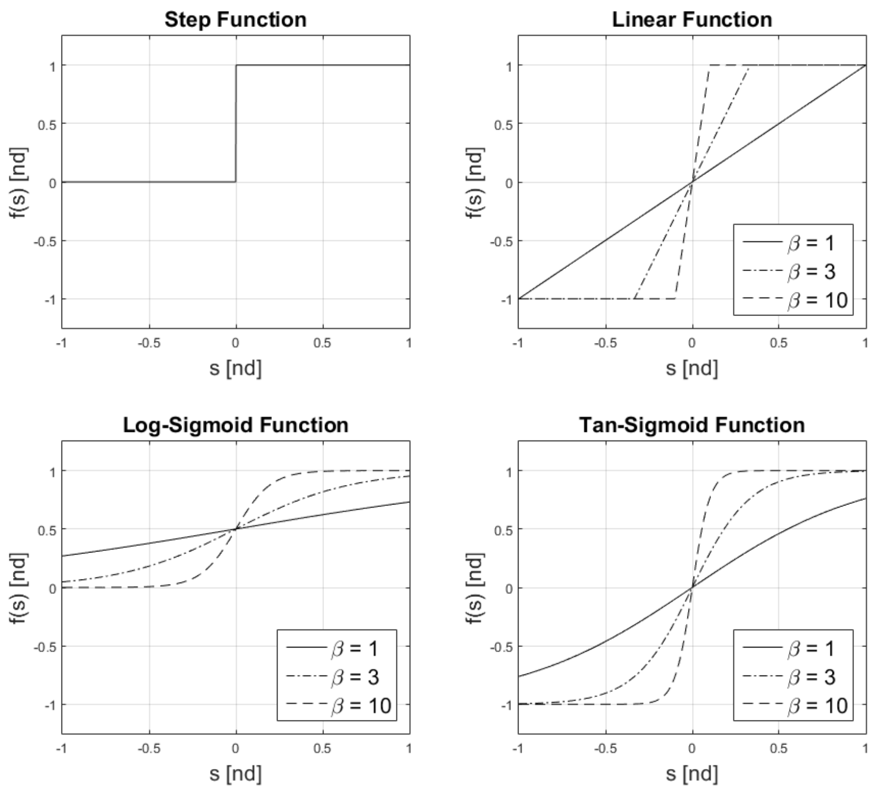


Figure 4.2. Various common activation functions

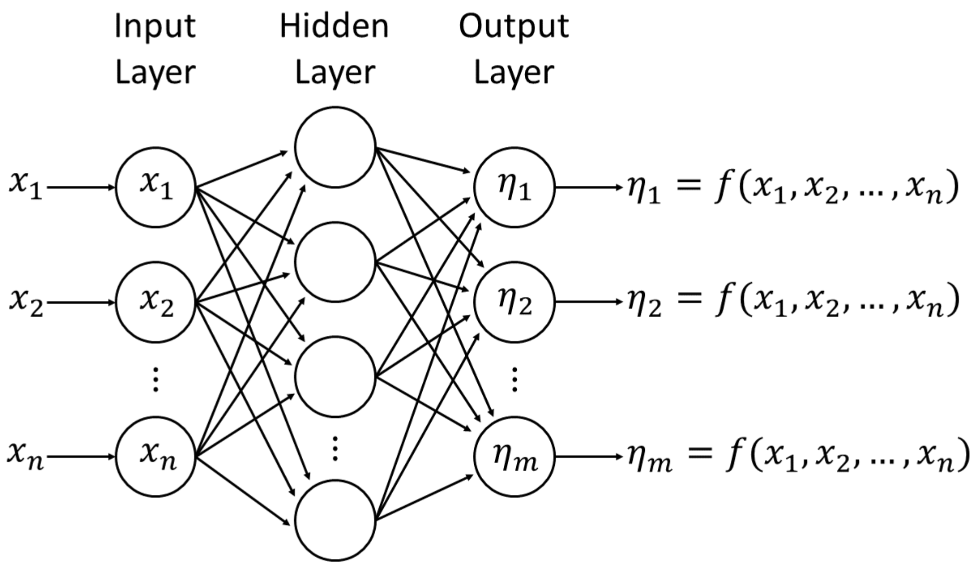

Input Layer
Hidden Layer
Output Layer
x_1
x_2
x_n
η_1
η_2
η_m
$\eta_1 = f(x_1, x_2, \ldots, x_n)$
$\eta_2 = f(x_1, x_2, \ldots, x_n)$
$\eta_m = f(x_1, x_2, \ldots, x_n)$

Figure 4.3. Neural Network with one hidden layer

relationships between the given inputs and outputs. To accomplish this, typically a backwards propagation of errors [37] in combination with a parameter optimization method such as Levenberg-Marquardt [38] or gradient descent. In the case of optimal control problems, the underlying functions the Neural Network predicts are relationships between the state and co-state variables in a region of optimal solutions. A Neural Network is used for difficult optimal control problems where there is no known closed-form expression for the state and co-state variables.

In the case of optimal control problems, unsupervised learning is most relevant. A set of data is created by numerically solving the optimal control problem for a range of boundary constraints. Using the data that were generated, the Neural Network is trained to infer relationships about the unknown co-state initial and terminal conditions. A Neural Network being used in this manner to reconstruct trajectories is shown in Figure 4.4.

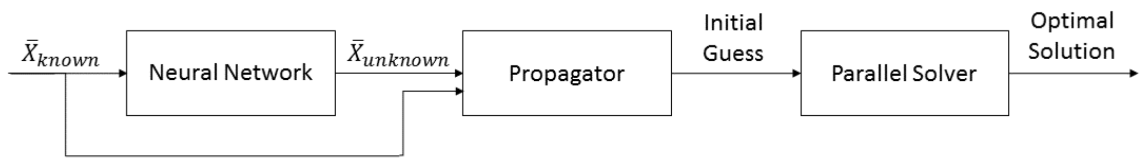


Figure 4.4. Neural Network solution strategy

Additionally, using constants of motion derived from Theorem 3.2.1 and Theorem 3.3.1, the dimension of the Neural Network can be reduced. This reduction serves to reduce the time required to train a Neural Network and increases accuracy with known analytic expressions. The previously mentioned solution strategy has been modified to include the constants of motions and is shown in Figure 4.5

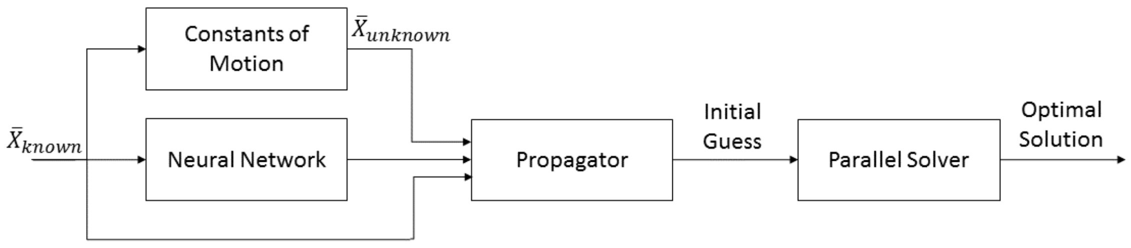


Figure 4.5. Reduced Neural Network solution strategy

4.3 Neural Network Validation

One criterion that must be satisfied for an algorithm to be considered for real-time on-board hypersonic trajectory optimization is that the convergence to an optimal solution must be guaranteed. For a solver using indirect methods to converge to a solution, the initial guess supplied must be accurate. Reconstructing an initial guess from a Neural Network is relatively simple for well-behaved problems since forward integration does this in a quick manner. Validating that the Neural Network satisfies the necessary conditions of optimality across the infinite possible input parameters that may be used during on-board applications is not as straightforward. Brute-force propagating trajectories from the Neural Network and validating they converge to optimal solutions is not a test that is rigorous enough. Theoretically, validating a Neural Network in this manner for 1,000 trajectories provides no guarantee that trajectory number 1,001 will converge as well. As stated earlier, for real-time applications convergence to an optimal solution must be guaranteed for the infinite number of possible inputs. This approach will need to leverage the analytical activation functions inside of the Neural Network.

The Neural Network is expressed as a combination of known functions and weights as described in Section 4.1, the outputs can be also be expressed as known symbolic functions of the inputs. An abstract expression output from a constructed Neural

Network is shown in Equation 4.5. The known and unknown variables depend on the optimal control problem definition.

$$\bar{X}_{unknown} = \bar{\eta}(\bar{X}_{known}) \tag{4.5}$$

Considering that for standard hypersonic systems, the Hamiltonian is autonomous such that $\{\mathscr{H}, \mathscr{H}\} = 0$, $\mathscr{H} = c_0$. The terminal boundary condition of the Hamiltonian is known, and therefore the Hamiltonian initial value is also known. Using the relationships provided by the Neural Network output in Equation 4.5, analytic approximations for the unknown states and co-states are substituted. For a Neural Network that accurately predicts the Hamiltonian, the residual is expected to be close to 0 as show in Equation 4.6.

$$\mathscr{H}(\bar{X}_{known}, \bar{\eta}) - c_0 \approx 0 \tag{4.6}$$

Determining the maximum residual of the initial Hamiltonian condition analytically can provide confidence that the Neural Network satisfies the necessary conditions for optimality. Theoretically, the location of extrema in the Hamiltonian residual can be located by evaluating where its first derivative vanishes. Root-solving Equation 4.7 analytically and evaluating these points in the residual equation will guarantee the Neural Network accurately predicts the Hamiltonian within an interval for a specific region of optimal solutions, excluding the boundaries.

$$\frac{\partial \mathscr{H}}{\partial \bar{X}_{known}} = 0 \tag{4.7}$$

While this method provides a robust process of determining the maximum Hamiltonian residual, it is fairly impractical for more complex problems where the use of a Neural Network is warranted in the first place. Due to the highly nonlinear expressions that the Neural Network returns, it is typically infeasible to root-solve Equation 4.7. In this case, an alternate method of checking the Hamiltonian residual must be used.

Using an activation function with well known analytic derivatives, such as the log-sigmoid function shown in Equation 4.3, a Taylor series expansion of the Hamiltonian residual may be used. Using an activation function that is C^{∞} guarantees the existence of higher order derivatives, thus allowing more accurate approximations by leveraging more known analytical values from the original function. Determining the order of the Taylor series expansion is limited by present-day computational power. For example, the residual plot in Figure 4.6 has a minimum of 12 extrema within the plotted boundaries, yet the highest order Taylor series expansion that could be generated in a reasonable amount of time was a 5^{th} order expansion. One Taylor series clearly is unable to represent the full plotted region considering the number of visible extrema. Due to this, several Taylor series expansions must be taken across the length of the region.

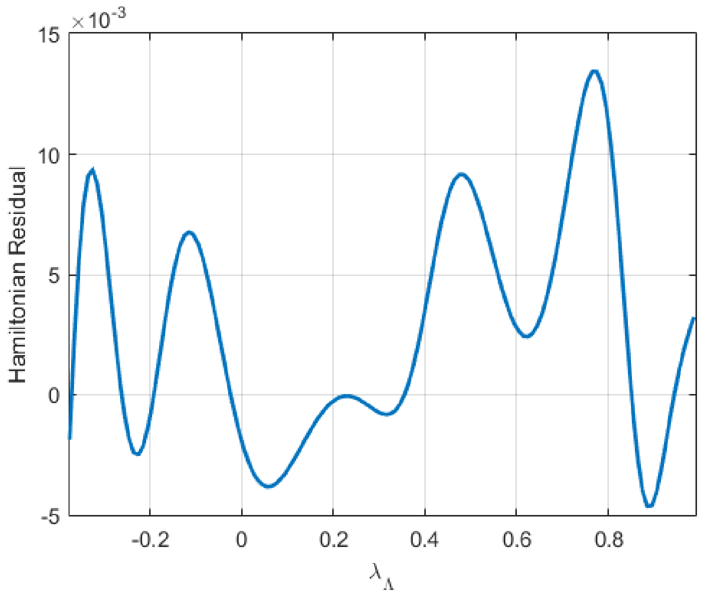
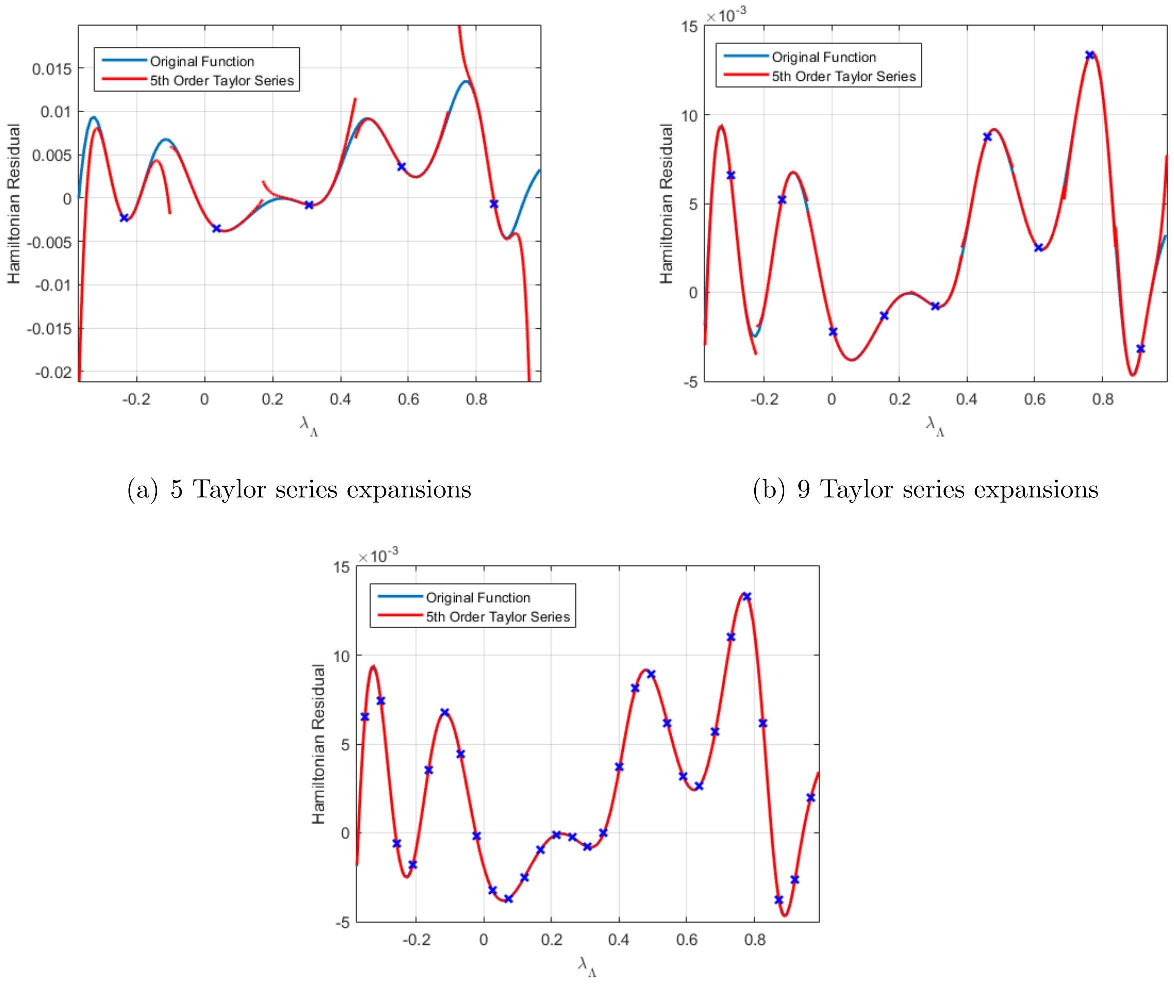


Figure 4.6. Example Hamiltonian residual [39]

Taylor series approximations use the values of a function's derivatives at a single point on a surface. Since it is infeasible to model all of a surface's derivatives, some error due to truncation exists. As approximations are made further away from this "center-point", the error tends to become worse and the Hamiltonian residual from this approximation will tend to appear worse as well. As shown in Figure 4.7, multiple Taylor series expansion may be used in finite intervals spanning the full region of interest to help mitigate this error. Similar to determining the order of the Taylor series approximations, determining the number of subdivisions that should be used is also limited by available computational power and time constraints.



(a) 5 Taylor series expansions

(b) 9 Taylor series expansions

(c) 29 Taylor series expansions

Figure 4.7. Taylor series approximations of the Hamiltonian residual for different numbers of intervals [39]

Extrema of the constructed Taylor series surrogate model of the Hamiltonian residual are found by evaluating where the first derivative vanishes. For 5th-order Taylor series expansions, this can be done analytically. Higher order expansions require a numerical method to root solve the first-derivative and therefore were avoided in favor of time and accuracy. Increasing the number of Taylor series expansions used to model the surface proved to be more beneficial than increasing the order past five. Using more local approximations creates a more accurate global approximation.

4.4 Hamiltonian Residual Boundary

Following the process outlined in Section 4.3 only guarantees extrema within a region of optimal solutions excluding the boundaries are found. This presents an issue with functions that contain an extremum along the boundary. Figure 4.8 shows how a function with a global extremum along the boundary may have a non-vanishing first derivative. Evaluating $\frac{df}{ds} = 0$ within $[s_1, s_2]$ captures two local extrema, whereas it can be seen visually that the global extremum is located at s_2 which is not captured since $\frac{df}{ds}(s_2) \neq 0$.

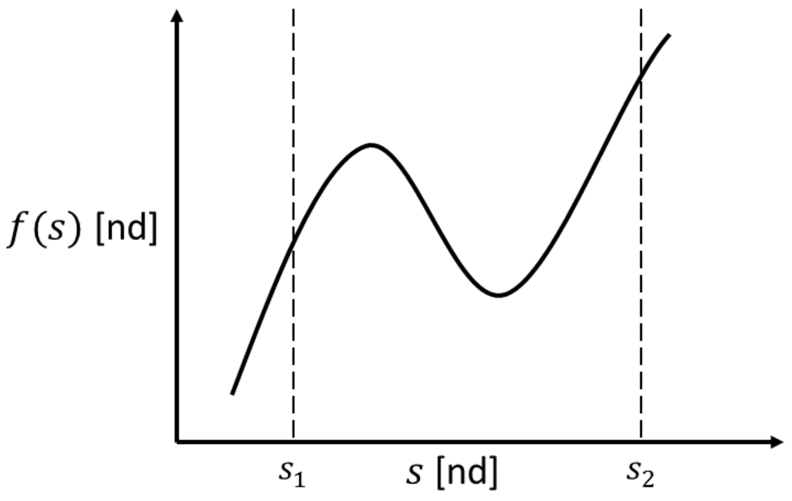


Figure 4.8. Example function with global extremum located at s_2

For a 1-dimensional case, one can simply include the boundary points as local extrema. This case corresponds to a region of optimal solutions with only 1 swept parameter. A region of optimal solutions with several swept parameters is more beneficial in a real application because of the greater number of cases it may predict.

For a function with more than one dimension, the boundaries are also continuous functions. Automatically including these boundaries as extrema is impossible because there are an infinite number of points associated with each one. For example, a 2-dimensional case may be predicted with a plane, mapping two inputs to one output. The boundaries of this plane are 1-dimensional lines. To account for this, the process outlined in Chapter 4.3 is repeated for additional residual functions generated by evaluating the Hamiltonian along its boundaries. The number of unique functions and unique points that are evaluated in this manner depends on the dimension and boundaries of Equation 4.6. A list containing total unique features for a footprint swept in a hypercube is shown in Table 4.1.

Table 4.1. Number of unique features on a hypercube

Dimension	Unique Functions	Unique Points
1-Cube (Line)	1	2
2-Cube (Square)	5	4
3-Cube (Cube)	19	8
4-Cube (Tesseract)	65	16
5-Cube	211	32
6-Cube	665	64

4.5 Maximum Residual Determination

Determining the number of Taylor series expansions required per unique function is not well defined. The main trade-off for adding additional expansions is an added computational cost. Recording the maximum predicted residual over a range of subdivisions is shown in Figure 4.9. As the number of Taylor series expansions used increases, the maximum residual predicted will converge and likewise it is expected that the individual Taylor series expansions have converged to the Hamiltonian surface.

After the residuals have converged to a single value, the highest residual corresponding to the greatest number of expansions is recorded. This value is associated with the initial Hamiltonian residual and given that it is sufficiently small, it is assumed that the error associated with the Neural Network output will be small as well.

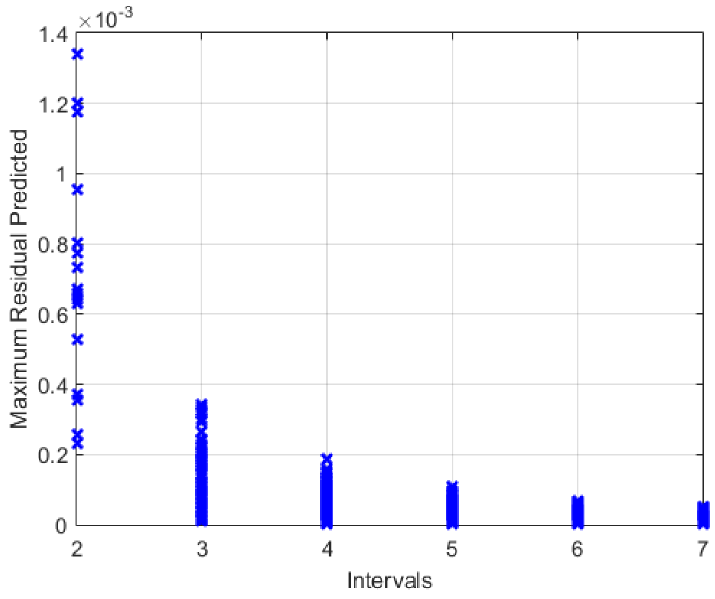


Figure 4.9. Maximum residual of predicted Hamiltonian by the Taylor series expansions [39]

5. PARALLEL IMPLEMENTATION

5.1 Parallel Processing Overview

Programming an application for a parallel architecture is fundamentally different than its serial counterpart. In an interpreted language, instructions that are written by a designer are directly executed by an interpreter. The interpreter reads code line-by-line and performs different actions on a CPU one at a time. This makes many standard interpreted languages extremely convenient to write simple programs in but creates challenges for designers who wish to take full advantage of systems with multiple cores.

The Multiple Shooting Method described in Chapter 2.4 is a good candidate algorithm to run on several cores unlike the SSM. The independence of one sub-trajectory from another allows computational resources to be handled independently whereby each sub-trajectory is assigned to an individual core on a computational node. Each sub-trajectory is solved independently with correction vectors determined based on a first-order method.

5.2 Single-node Multi-core Implementation

Implementation of this algorithm was done using the open-source Python programming language [40]. Various Python-specific issues were addressed to implement the algorithm effectively in parallel. Typically, programs run, carry out computations, and return a result in serial. Additional work is required to make effective use of parallel architectures. For example, Python's interpreter implements a Global Interpreter Lock [41,42]. The Global Interpreter Lock will synchronize the execution of threads so that only one thread will run at a time. A Python application utilizing a single interpreter with multiple threads will only make effective use of a single core. Figure 5.1 illustrates how a properly multi-threaded application will run in serial due to the interpreter.

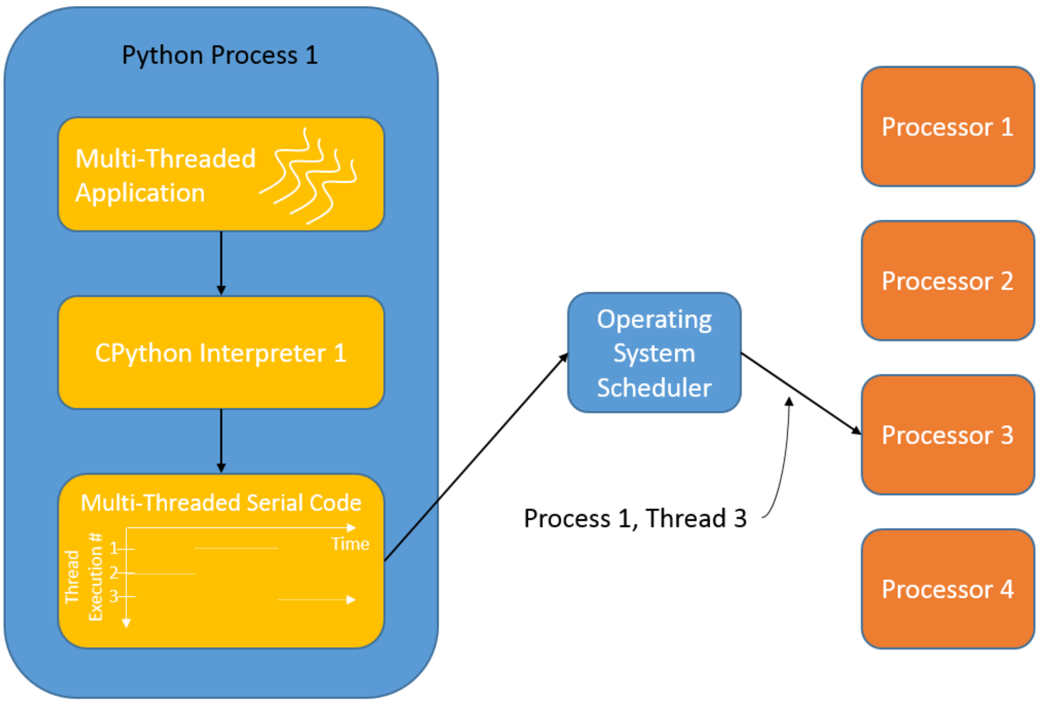


Figure 5.1. CPython interpreter bottlenecking a multi-threaded application

The Global Interpreter Lock is essential for applications that are not thread-safe. To run applications in parallel, more processes are created. Since each Python process contains its own interpreter with its own Global Interpreter Lock, the processes are separated from one another and can run in parallel. By creating additional processes in this manner, arcs can each be allocated to their own individual process and integrated independently in parallel. Figure 5.2 shows how creating new processes allows a program to take advantage of parallel architectures. It is important to note that this method works for any languages where the interpreter has a mutual exclusion lock.

Implementing code in this manner allows an entire computational node to be utilized. For problems with substantially complex derivatives, a boost in performance

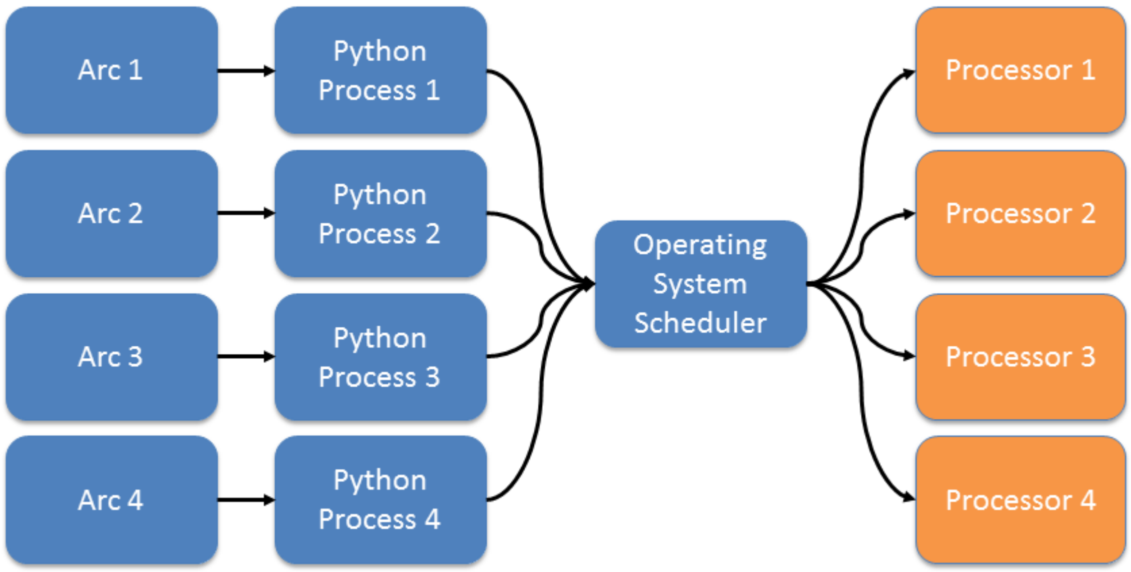


Figure 5.2. Spawning multiple processes utilizes multiple processors

is expected due to distributing heavy computations over multiple cores. For relatively simple problems with derivatives that are quick to evaluate, we expect there to be a decrease in performance. This decrease in performance would be caused by the overhead of spawning new processes. Since spawning new processes also requires copying memory and other I/O bound operations, creating new processes for problems that typically evaluate quickly with the SSM will not be beneficial.

Scaling up and scaling down the problem becomes an issue, since for realistic applications the optimal number of arcs the trajectory should be subdivided into will not equal the number of processors available. Solving a TPBVP problem using 4 arcs on a system with 8 cores has no need to spawn 8 processes. This would result in additional overhead. Likewise, a system that is several years old may not have many cores available, but a highly sensitive problem may require dozens or even hundreds of arcs to converge. A solution to this issue is to use an asynchronous scheduler in the program. Figure 5.3 illustrates how several arcs are distributed across the available processors. Implementing the algorithm in this manner ensures that the problem can effectively execute on state-of-the-art hardware as well as hardware that may be several years behind state-of-the-art.

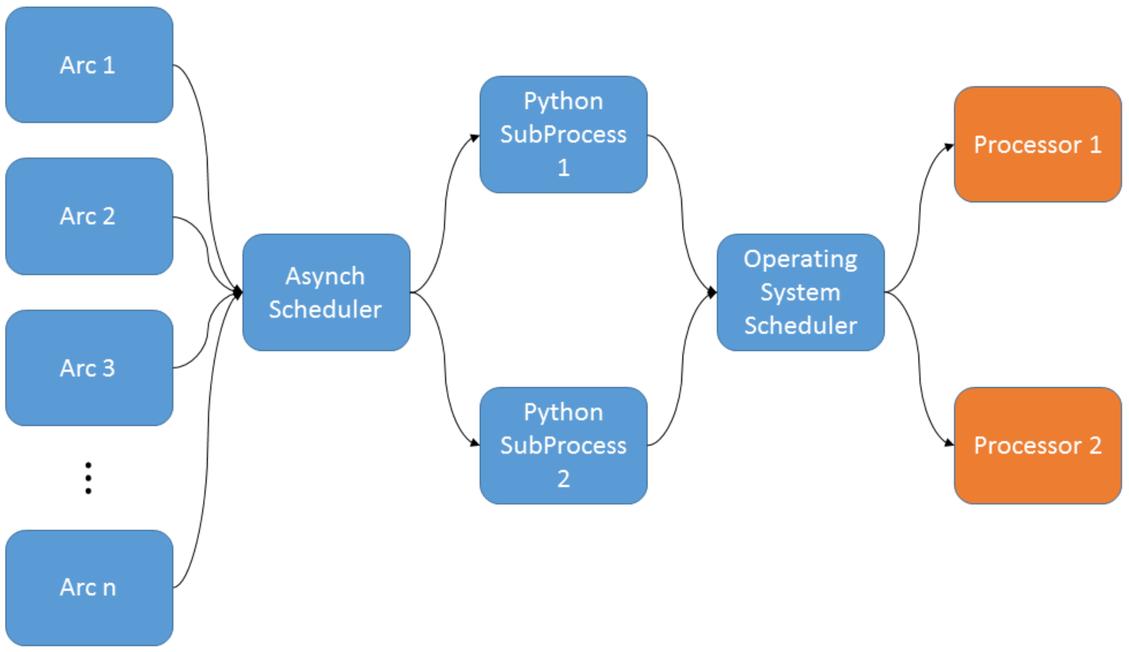


Figure 5.3. Asynchronously scheduling arc integration effectively uses the available resources

To test this, the MSM was implemented as detailed above and run on a Cray XC30 (U.S. Department of Defense High Performance Computing System "Lightning") in collaboration with the Air Force Research Laboratory at Eglin Air Force Base. An optimal control problem with a substantially expensive derivative was simulated using a fixed-step RK4 integration scheme with 5000 time subdivisions. The solution for the optimal control problem was obtained using a combination of continuation and the multiple shooting method. Timed trials were performed on a single-node of a Cray XC30 and results are shown in Figure 5.4. The shown times represent a theoretical maximum in terms of benefit of using more processors and confirms that the algorithm makes effective use of available processors.

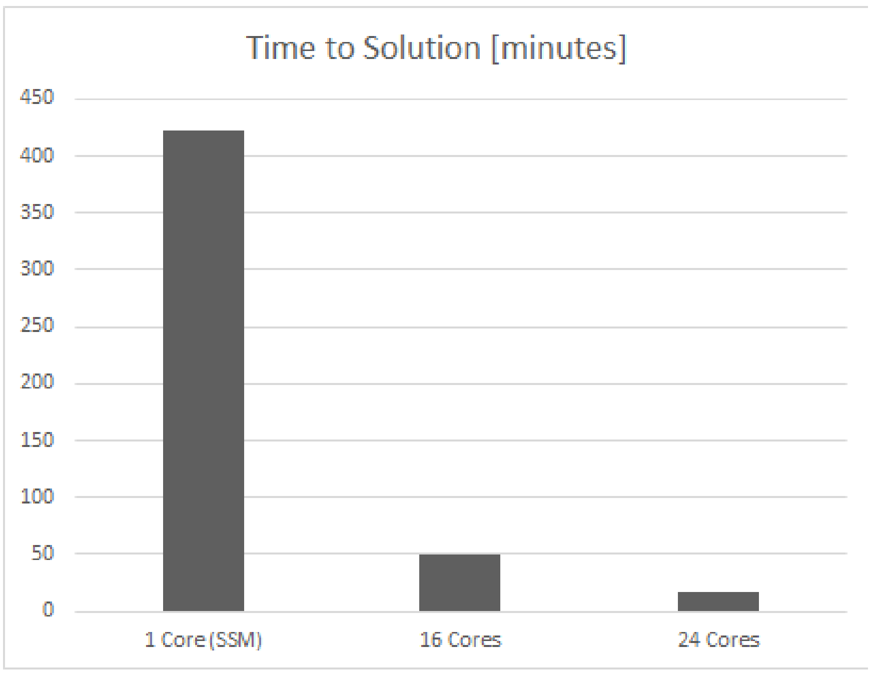


Figure 5.4. Timed trials of a computationally expensive optimal control problem

6. APPLICATION TO HYPERSONIC UN-POWERED FLIGHT

A vehicle in atmospheric flight traveling over five times the speed of sound is generally considered to be traveling at hypersonic speeds. The flight is considered unpowered as well due to the absence of thrust. Assuming the vehicle is traveling at hypersonic speeds allows the use of Newtonian theory for modeling aerodynamic forces, providing great simplifications. A non-rotating spherical Earth model with exponential density is also assumed, along with low-order curve-fit polynomials approximating C_L and C_D. A vehicle-centered polar coordinate system with 3 degree-of-freedom dynamics [43] is used to derive the equations of motion listed in Equations 6.1-6.6.

$$\dot{r} = v\sin(\gamma) \tag{6.1}$$

$$\dot{\theta} = \frac{v\cos(\gamma)\cos(\psi)}{r\cos(\phi)} \tag{6.2}$$

$$\dot{\phi} = \frac{v\cos(\gamma)\sin(\psi)}{r} \tag{6.3}$$

$$\dot{v} = -\frac{D}{m} - \frac{\mu\sin(\gamma)}{r^2} \tag{6.4}$$

$$\dot{\gamma} = \frac{L\cos(\xi)}{mv} - \frac{\mu\cos(\gamma)}{vr^2} + \frac{v\cos(\gamma)}{r} \tag{6.5}$$

$$\dot{\psi} = \frac{L\sin(\xi)}{mv\cos(\gamma)} - \frac{v\cos(\gamma)\cos(\psi)\tan(\phi)}{r} \tag{6.6}$$

6.1 Maximum Energy Case

The selected maximum energy control problem refers to a vehicle maximizing impact velocity, or terminal kinetic energy and represents a hypothetical un-powered guided weapon. Realistic path constraints, such as those relating to heat rate, G-loading, and country overflight constraints were ignored. The cost functional is defined in Equation 6.7 with no path cost ($\mathscr{L} = 0$). The maximum energy case is selected to demonstrate the benefits of using a Neural Network to generate initial guesses.

$$J = -v_f^2 \tag{6.7}$$

6.1.1 Necessary Conditions

Derivation of the necessary conditions for optimality was performed by evaluating Equations 2.16-2.21 in Mathematica [44]. Symbolically root-solving Equation 2.16 results in multiple analytic control laws. Selection of the which control law is used at each time step is made using Pontryagin's Minimum Principle [18, 19] shown in Equation 6.8. For each point in time along the trajectory the Hamiltonian is evaluated with each resulting control law. Pontryagin's Minimum Principle states that the Hamiltonian must take on a minimum value over the set of admissible controls. Small perturbations in an optimal control history will result in an equivalent or increasing Hamiltonian.

$$\mathscr{H}(\bar{x}^*, \bar{\lambda}^*, \bar{u}^*, t) \leq \mathscr{H}(\bar{x}^*, \bar{\lambda}^*, \bar{u}, t) \tag{6.8}$$

6.1.2 Constants of Motion

Constants of motion for un-powered hypersonic flight over a spherical planet are derived using Theorem 3.2.1 and listed here for completeness [45]. Additionally, these constants of motion can be computed using Theorem 3.3.1, satisfying all associated properties of the Poisson bracket. The specific usage of the constants of motion is heavily dependent on the boundary conditions set for the optimal control problem and is addressed in Section 6.1.3.

$$c_1 = \lambda_\phi \cos(\theta) + \lambda_\theta \tan(\phi) \sin(\theta) - \lambda_\psi \frac{\sin(\theta)}{\cos(\phi)} \tag{6.9}$$

$$c_2 = \lambda_\phi \sin(\theta) - \lambda_\theta \tan(\phi) \cos(\theta) + \lambda_\psi \frac{\cos(\theta)}{\cos(\phi)} \tag{6.10}$$

$$c_3 = \lambda_\theta \tag{6.11}$$

6.1.3 Footprint and Neural Network Generation

Table 6.1 shows the set of initial and terminal conditions used to define the optimal control problem and Table 6.2 expands on the swept parameters used to define the optimal footprint. Samples were taken in the swept region with a $0.125^o \times 0.125^o$ resolution and some trajectories are shown in Figure 6.1. A reduction in output dimension was performed using the constants of motion derived earlier. Both latitude and longitude are known quantities at the initial and terminal states due to the constraints imposed upon the boundary conditions. Through the necessary conditions for optimality, a numeric quantity can be found for $\lambda_{\psi 0}$. Since ψ_0 is free, $\lambda_{\psi 0} = 0$. Because there are more unknowns than there are equations, some unknown states that appear in the constants of motion will still need to be calculated by the Neural Network. Predictions were given by the Neural Network for $\lambda_{\theta 0}$ and $\lambda_{\phi 0}$, then numeric values for $\lambda_{\phi f}$, $\lambda_{\theta f}$, and $\lambda_{\psi f}$ were found through the constants of motion.

After training, the resulting data file's size was 280 megabytes on disk. Neural Network training was performed with Levenberg-Marquardt backpropagation in 15 minutes. The output Neural Network file's size was 1 megabyte on disk and evaluates in an average time of 10.87 milliseconds.

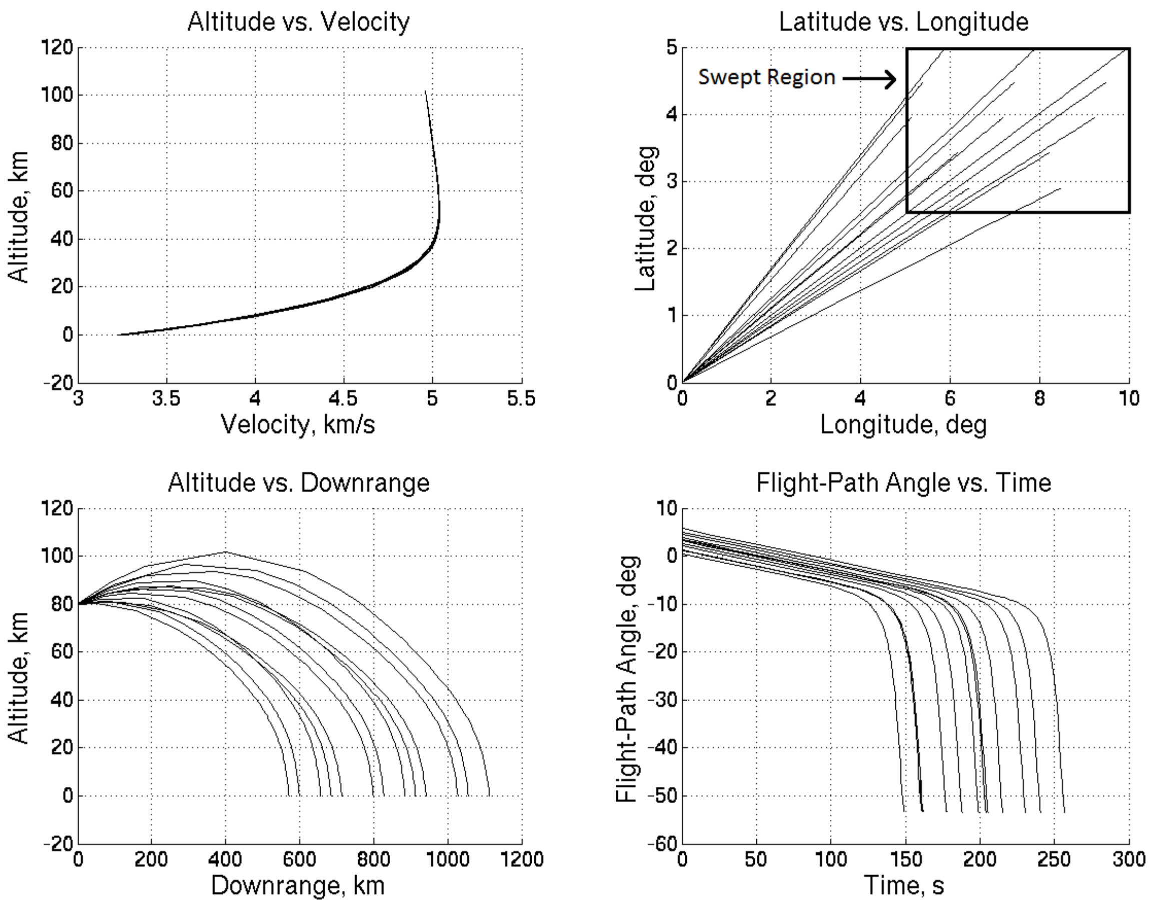


Figure 6.1. Various solutions sampled in swept region

Table 6.1. Vehicle initial "post-boost" and terminal conditions

State	Initial Value	Terminal Value
Height, h	80 km	0 km
Longitude, θ	0 deg	5-10 deg
Latitude, ϕ	0 deg	2.5-5 deg
Velocity, v	5 km/s	free
FPA, γ	free	free
Heading Angle, ψ	free	free

Table 6.2. Maximum energy footprint area

State	Width	Samples
Terminal Longitude, θ_f	5 degrees	40
Terminal Latitude, ϕ_f	2.5 degrees	20

6.1.4 Propagation and Solver Input

Using the outputs from the Neural Network, solutions are propagated forward in time from the vehicle's initial state with an RK4 integration scheme. These generated initial guess structures are then sent into either a SSM or MSM solver utilizing a range of computational cores. Timed trials were performed on a parallel system with an Intel Core i7-4700MQ CPU, and a summary of the timed trials is shown in Table 6.3.

It can be seen in Table 6.3 that there are benefits to using the MSM over the SSM. This parallelization specifically has two distinct advantages. The first benefit is the reduced time to solution through simultaneously integrating several arcs of the trajectory. The second benefit is the reduced number of iterations the solver must take to converge to an optimal solution. This is due to the fact that each individual arc has its own correction vector and therefore more state corrections per

Table 6.3. Timed trials with various algorithms

Algorithm	Number of Arcs	Average time to Solution, seconds	Average Number of Iterations
SSM	1	10.34	15.2
MSM	2	7.17	7.4
MSM	3	6.19	6.9
MSM	4	4.91	6.4
MSM	5	4.95	6.4
MSM	6	5.68	6.2
MSM	7	5.55	6.3
MSM	8	5.26	6.1

iteration are made. Table 6.3 was generated using an Intel Core i7-4700MQ CPU. The Intel Core i7-4700MQ CPU is a CPU with four physical cores, eight as seen by the operating system including the Hyper-Threading virtual cores. This means that while the processor may be able to simultaneously run 8 threads, it is not guaranteed. Increasing the number of processes used beyond the number of physical cores available will not guarantee a performance boost.

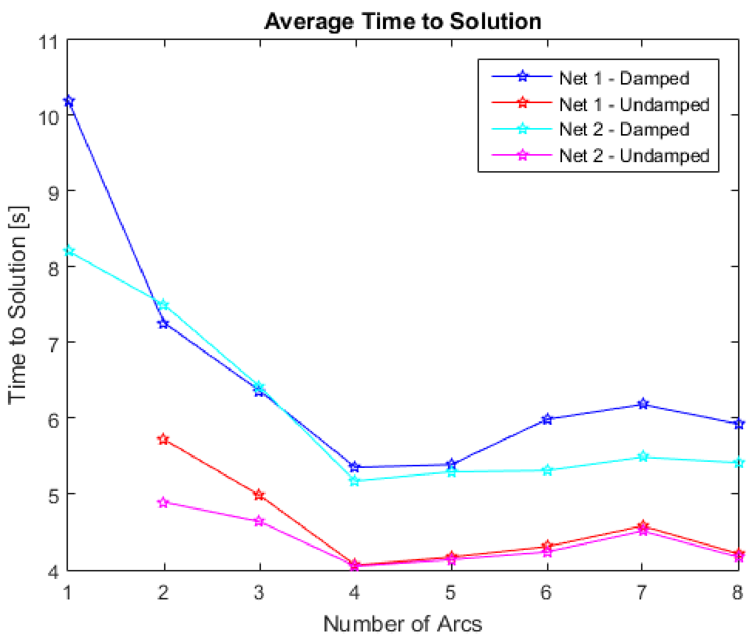
From the same problem definition, two Neural Networks were generated using two different datasets. Both datasets covered the same swept regions as shown in Figure 6.1 and cover the same family of solutions, though one dataset was generated to a higher tolerance than the other. Additionally, damping was removed from the solver in an effort to decrease the time and number of iterations to reach a solution, totaling four unique scenarios. Details regarding the Neural Networks generated are shown in Table 6.4. The training time of the Neural Network using the higher quality dataset was significantly longer since more iterations were required to sufficiently capture trends in the data. This was done in an attempt to maximize the quality of the network. The results of timed trials for a various number of arcs is shown in Figure 6.2.

Table 6.4. Generated Neural Network information

Network Number	Dataset Tolerance	Training Time	Estimated Max Residual
Net 1	1×10^{-4}	23 minutes	1.41×10^{-3}
Net 2	1×10^{-6}	3.7 hours	5.22×10^{-5}

Two trends are apparent from Figure 6.2. The first trend is that removing the damping from the solver significantly reduces time and number of iterations to approach a solution. While this is beneficial for some cases, timed trials for the SSM we not able to be performed for the undamped solver since the convergence rate was very low. Removing the damping came at the cost of decreasing robustness. The second trend that can be seen is that the benefits of adding additional arcs plateaus around four cores. Whether or not this is directly caused by the Hyper-Threading technology is unknown. A detailed investigation into the effects of Hyper-Threading technology was not performed. There is additional overhead with transferring data between Python processes as the number of processes grows. What is likely is that the detrimental effects of adding additional processors surpassed the benefits and the performance increase plateaued.

Even though the predicted residual for the second Neural Network is within acceptable limits, the solver still required some iterations to converge. This is likely due to the fact that only the Hamiltonian at the initial condition is checked across the infinite span of inputs to the Neural Network. The Neural Network was not validated at any other locations other than the initial state due to the fact that there is no closed-form solution for the trajectory. Errors in the initial state can potentially grow as the trajectory evolves.



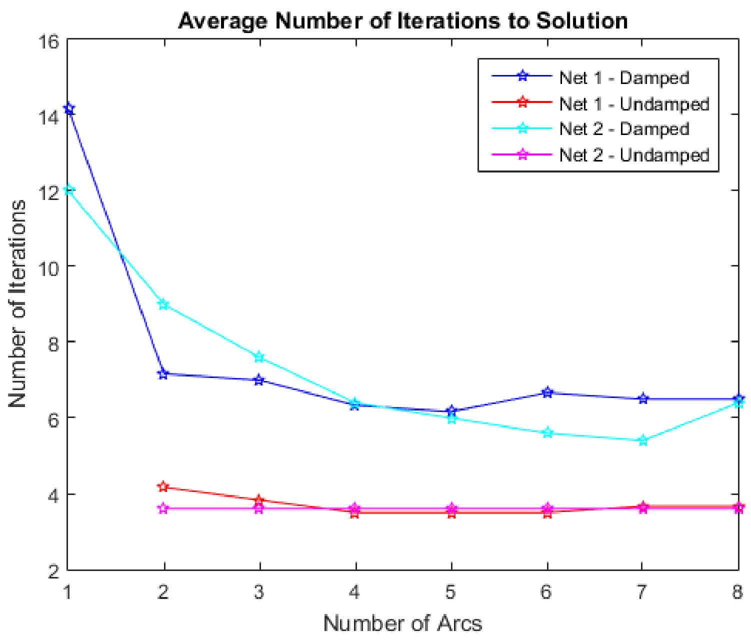


Figure 6.2. Summary of timed trials

7. SUMMARY

This study presents a parallelized method to support time-critical missions. An emphasis was placed on using known analytical quantities to reduce time to solution and error while reducing the overall size on disk of a set of solutions.

A Multiple Shooting Algorithm is developed and deployed onto a parallel system. Significant performance boosts were shown for problems with substantially difficult or complex derivatives. Utilizing more processors in parallel allows the solver to make more initial vector corrections per iteration thus minimizing the amount of wasted computational resources. A reduction in both time and number of iterations to a solution were shown for the Multiple Shooting Method.

A region of optimal solutions is generated and used to train an Artificial Neural Network. Using the Neural Network to predict the unknown parameters of the system provides accurate initial guesses for optimal solutions through forward integration. To further improve the accuracy of the Neural Network, Noether's Theorem is used to generate constants of motion associated with the optimal control system. Leveraging these known quantities allows one to bypass sections of the Neural Network where approximations would normally have taken place without the reduction. A method is developed for testing the accuracy of Neural Network by analyzing the Hamiltonian across the infinite number of inputs. This increase in confidence that a Neural Network will return high quality solutions has potential real-time applications. Characterizing a family of optimal solutions in this manner also reduces the size on disk eliminating long search times associated with large data files. Two example Neural Networks were generated and timed trials were performed with various algorithms utilizing both serial and parallel processes.

This research has applications to real-time hypersonic optimization as well as time-critical applications. Using a combination of parallel processing with generated high-quality initial guesses, this method is shown to reduce the time to solution as well as increase confidence that convergence on an optimal solution will be obtained.

8. FUTURE WORK

8.1 Hamiltonian Dimension Reduction

The integrals of optimal motion shown in Chapter 3 are conserved quantities as a result of physical symmetries that exist in the system. A system with $2n$ dimensions and 3 symmetries can potentially reduce to a system with $2n-6$ dimensions [46], thus resulting in a less complex problem to solve. Reductions have been performed in other systems where dynamics are decoupled into substantially simpler subsystems [47,48]. It is expected that application of a similar procedure to hypersonic design problems will result in a significantly simpler problem with more benefits than that of the reduced Neural Network defined in Chapter 4.

8.2 Symplectic Integration

Indirect methods excel because of its use of analytic information in the problem statement. For long-term integrations and trajectories where the Hamiltonian approaches a critical point, it has been show that structure-preserving symplectic integrators are superior to non-symplectic ones [49]. Integration of hypersonic systems is typically done using RK4 or a similar algorithm. For sensitive problems, integration cannot be performed in a simple manner using these algorithms. Implementation of a symplectic integrator may provide a means for reducing the sensitivities associated with these problems.

8.3 High Fidelity Vehicle Analysis

For hypersonic vehicle concept design, typically simplifying assumptions are made that decrease fidelity and accuracy in favor of time constraints. Previous examples include simultaneous aerodynamic and trajectory optimization using relatively simplified geometries [50, 51]. The authors of Ref 50 and Ref 51 demonstrate how a blunted cone, blended wedge, and blunted biconic were used to estimate the aerodynamics of the vehicle and also provide a generalized method for augmenting the problem. This simplified systems standpoint has proven to be a useful tool for designers in the early stages of development. Certain multidisciplinary analysis have made use of reduced-order models to improve the quality and speed of design. Such examples include the simulation of a flexible vehicle [52], modeling of aerothermalelasticity [53, 54], fluid-thermal-structural interaction [55], rapid loads prediction [56], and nonlinear reduced-order thermal models [57]. Such reduced-order models take a fraction of the time to evaluate in comparison to their full simulation counterparts. Due to the iterative nature of solving NLP problems, performing full CFD, FEM, or thermal analysis in-the-loop with trajectory optimization is infeasible even with current state-of-the-art computational architectures. Reduced-order models can be used in place of the full simulations where necessary for conceptual design. Including the reduced-order models can be done by adjoining them to the problem with their own Lagrange multipliers by the process outlined in Chapter 2.

REFERENCES

REFERENCES

[1] William F. Brinkman, Douglas E. Haggan, and William W. Troutman. A history of the invention of the transistor and where it will lead us. *Solid-State Circuits, IEEE Journal of*, 32(12):1858–1865, Dec 1997.

[2] Gordon E Moore et al. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[3] Robert R. Schaller. Moore's law: past, present and future. *Spectrum, IEEE*, 34(6):52–59, Jun 1997.

[4] Robert W. Keyes. Moore's law today. *Circuits and Systems Magazine, IEEE*, 8(2):53–54, Second 2008.

[5] Robert W. Keyes. Fundamental limits of silicon technology. *Proceedings of the IEEE*, 89(3):227–239, Mar 2001.

[6] Steve Melvin, Mario Nemirovsky, Enric Musoll, Jeff Huynh, Rodolfo Milito, Hector Urdaneta, Koroush Saraf, et al. A massively multithreaded packet processor. *Proc. of NP2, Held in conjunction with HPCA-9, Anaheim, CA, USA*, 2003.

[7] Tau Leng, Rizwan Ali, Jenwei Hsieh, Victor Mashayekhi, and Reza Rooholamini. An empirical study of hyper-threading in high performance computing clusters. *Linux HPC Revolution*, 2002.

[8] Krste Asanovic, Ras Bodik, Bryan C. Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William L. Plishker, John Shalf, Samuel W. Williams, et al. The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[9] 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.

[10] I. Michael Ross. A beginners guide to dido. *Ellisar, LLC, Monterey, CA*, 2007, 1998.

[11] Ryan D. Gauntt. Aircraft course optimization tool using GPOPS matlab code. Technical report, DTIC Document, 2012.

[12] 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.

[13] 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.

[14] 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.

[15] William Karush. *Minima of functions of several variables with inequalities as side constraints*. PhD thesis, Masters thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[16] Harold W Kuhn. Nonlinear programming: a historical view. In *Traces and Emergence of Nonlinear Programming*, pages 393–414. Springer, 2014.

[17] Kevin W. Cassel. *Variational Methods with Applications in Science and Engineering*. Cambridge University Press, 2013.

[18] 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.

[19] Lev S. Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[20] James M. Longuski, José J. Guzmán, and John E. Prussing. *Optimal Control with Aerospace Applications*. Springer, 2014.

[21] Yvette Kosmann-Schwarzbach. *The Noether theorems*. Springer, 2011.

[22] Delfim F.M. Torres. On the noether theorem for optimal control. *European Journal of Control*, 8(1):56–63, 2002.

[23] John R. Cary and Robert G. Littlejohn. Noncanonical hamiltonian mechanics and its application to magnetic field line flow. *Annals of Physics*, 151(1):1–34, 1983.

[24] Scott Kirkpatrick. Rough times ahead. *Science*, 299(5607):668–669, 2003.

[25] Francesco Decarolis, Ricardo Mayer, and Martin Santamaria. Homotopy continuation methods, 2002.

[26] Eugene L. Allgower and Kurt Georg. *Introduction to numerical continuation methods*, volume 45. SIAM, 2003.

[27] 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, Nov 2014.

[28] Anthony J. Calise, Nahum Melamed, and Seungjae Lee. Design and evaluation of a three-dimensional optimal ascent guidance algorithm. *Journal of Guidance, Control, and Dynamics*, 21(6):867–875, Nov 1998.

[29] Peter F. Gath and Anthony J. Calise. Optimization of launch vehicle ascent trajectories with path constraints and coast arcs. *Journal of Guidance, Control, and Dynamics*, 24(2):296–304, 2001.

[30] J-P Marec. Optimal space trajectories. *NASA STI/Recon Technical Report A*, 80:48848, 1979.

[31] Raúl Rojas. *Neural networks: a systematic introduction.* Springer Science & Business Media, 2013.

[32] Josef Stoer and Roland Bulirsch. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

[33] Emmy Noether. Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, mathematisch-physikalische Klasse*, 1918:235–257, 1918.

[34] Albert Einstein. Professor einstein writes in appreciation of a fellow-mathematician. *New York Times*, 1935.

[35] Floris Takens. Symmetries, conservation laws and variational principles. In Jacob Palis and Manfredo do Carmo, editors, *Geometry and Topology*, volume 597 of *Lecture Notes in Mathematics*, pages 581–604. Springer Berlin Heidelberg, 1977.

[36] Jeremy Butterfield. On symmetry and conserved quantities in classical mechanics. In *Physical Theory and its Interpretation*, pages 43–100. Springer, 2006.

[37] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

[38] Amir A. Suratgar, Mohammad B. Tavakoli, and Abbas Hoseinabadi. Modified levenberg-marquardt method for neural networks training. *World Acad Sci Eng Technol*, 6:46–48, 2005.

[39] Michael A Bolender, Michael J Grant, and Michael 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*, page 3564, 2015.

[40] Guido Van Rossum and Fred L. Drake. *The python language reference manual.* Network Theory Ltd., 2011.

[41] John Markus Bjørndalen, Brian Vinter, and Otto J Anshus. Pycsp-communicating sequential processes for python. In *CPA*, pages 229–248, 2007.

[42] David Beazley. Understanding the python gil. In *PyCON Python Conference. Atlanta, Georgia*, 2010.

[43] Nguyen X. Vinh, Adolf Busemann, and Robert D. Culp. Hypersonic and planetary entry flight mechanics. *NASA STI/Recon Technical Report A*, 81:16245, 1980.

[44] Wolfram Research, Inc. Mathematica, 2012.

[45] H.G. Moyer. Integrals for optimal flight over a spherical earth. *AIAA Journal*, 11(10):1441–1443, 1973.

[46] Reduction. In *Lectures on Symplectic Geometry*, volume 1764 of *Lecture Notes in Mathematics*, pages 173–179. Springer Berlin Heidelberg, 2001.

[47] Tomoki Ohsawa. Symmetry reduction of optimal control systems and principal connections. *SIAM Journal on Control and Optimization*, 51(1):96–120, 2013.

[48] Perinkulam S Krishnaprasad. Optimal control and poisson reduction. Technical report, DTIC Document, 1993.

[49] 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.

[50] Michael J Grant, Ian G Clark, and Robert D Braun. Rapid simultaneous hypersonic aerodynamic and trajectory optimization using variational methods. In *AIAA Atmospheric Flight Mechanics Conference*, page 6640, 2011.

[51] Michael J Grant. *Rapid simultaneous hypersonic aerodynamic and trajectory optimization for conceptual design.* PhD thesis, Georgia Institute of Technology, 2012.

[52] Ryan C. Kitson. Aeroelastic modeling and simulation of very flexible vehicles. In *3rd Annual Meeting of the AFRL Mathematical Modeling and Optimization Institute. Shalimar, Florida.* UF REEF, Jul 2015.

[53] Ryan Klock and Carlos E. Cesnik. chapter Aerothermoelastic Simulation of Air-Breathing Hypersonic Vehicles. AIAA SciTech. American Institute of Aeronautics and Astronautics, Jan 2014.

[54] Ryan Klock and Carlos E. Cesnik. chapter Aerothermoelastic Reduced-Order Model of a Hypersonic Vehicle. AIAA Aviation. American Institute of Aeronautics and Astronautics, Jun 2015.

[55] Emily R. Dreyer. Fluid-thermal-structural interaction analyses in extreme environments. In *2rd Annual Meeting of the AFRL Mathematical Modeling and Optimization Institute. Shalimar, Florida.* UF REEF, 2014.

[56] Emily R. Dreyer. Rapid loads prediction for hypersonic vehicles using cfd surrogates. In *3rd Annual Meeting of the AFRL Mathematical Modeling and Optimization Institute. Shalimar, Florida.* UF REEF, 2015.

[57] Ryan Klock. Nonlinear thermal reduced-order models for a hypersonic vehicle. In *3rd Annual Meeting of the AFRL Mathematical Modeling and Optimization Institute. Shalimar, Florida.* UF REEF, 2015.