Hybrid Solution of Stochastic Optimal Control Problems Using Gauss Pseudospectral Method and Generalized Polynomial Chaos Algorithms

DISSERTATION

Gerald C. Cottrill, Lieutenant Colonel, USAF

AFIT/DS/ENY/12-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/ENY/12-11

HYBRID SOLUTION OF STOCHASTIC OPTIMAL CONTROL PROBLEMS USING GAUSS PSEUDOSPECTRAL METHOD AND GENERALIZED POLYNOMIAL CHAOS ALGORITHMS

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Gerald C. Cottrill, BS, MS

Lieutenant Colonel, USAF

March 2012

AFIT/DS/ENY/12-11

HYBRID SOLUTION OF STOCHASTIC OPTIMAL CONTROL PROBLEMS USING GAUSS PSEUDOSPECTRAL METHOD AND GENERALIZED POLYNOMIAL CHAOS ALGORITHMS

Gerald C. Cottrill, BS, MS
Lieutenant Colonel, USAF

Approved:

Frederick G. Harmon — 9 Dec 2011

Lt. Col. Frederick G. Harmon, PhD
Dissertation Advisor — Date

Richard Cobb — 9 Dec 2011

Dr. Richard G. Cobb
Committee Member — Date

William P. Baker — 9 Dec 2011

Dr. William P. Baker
Committee Member — Date

Accepted:

M U Thomas — 27 Dec 2011

M. U. THOMAS
Dean, Graduate School of Engineering
and Management — Date

Abstract

A hybrid numerical algorithm combining the Gauss Pseudospectral Method (GPM) with a Generalized Polynomial Chaos (gPC) method to solve nonlinear stochastic optimal control problems with constraint uncertainties is presented. The GPM and gPC have been shown to be spectrally accurate numerical methods for solving deterministic optimal control problems and stochastic differential equations, respectively. The gPC uses collocation nodes to sample the random space, which are then inserted into the differential equations and solved by applying standard differential equation methods. The resulting set of deterministic solutions is used to characterize the distribution of the solution by constructing a polynomial representation of the output as a function of uncertain parameters. Optimal control problems are especially challenging to solve since they often include path constraints, bounded controls, boundary conditions, and require solutions that minimize a cost functional. Adding random parameters can make these problems even more challenging. The hybrid algorithm presented in this dissertation is the first time the GPM and gPC algorithms have been combined to solve optimal control problems with random parameters. Using the GPM in the gPC construct provides minimum cost deterministic solutions used in stochastic computations that meet path, control, and boundary constraints, thus extending current gPC methods to be applicable to stochastic optimal control problems. The hybrid GPM-gPC algorithm was applied to two concept demonstration problems: a nonlinear optimal control problem with multiplicative uncertain elements and a trajectory optimization problem simulating an aircraft flying through a threat field where exact locations of the threats are unknown. The results show that the expected value, variance, and covariance statistics of the polynomial output function approximations of

the state, control, cost, and terminal time variables agree with Monte-Carlo simulation results while requiring on the order of $\left(\frac{1}{40}\right)^{th}$ to $\left(\frac{1}{100}\right)^{th}$ the number of collocation points and computation time. It was shown that the hybrid algorithm demonstrated an ability to effectively characterize how the solutions to optimization problems vary with uncertainty, and has the potential with continued development and availability of more powerful computer workstations, to be a powerful tool applicable to more complex control problems of interest to the Department of Defense.

Acknowledgements

The people who make the most sacrifices while assigned to AFIT are the families of the students attending school. I have spent many nights and weekends away from my family doing school work, leaving my wife at home alone to take care of two small children and my little boys missing their daddy. It is to my wife and sons that I am most thankful for their patience, support, and understanding while I've been pursuing my Ph.D. Being able to provide a great future for my family is the reason that I began this program and the motivation that kept me going on days when I wanted to quit and return to a *normal* assignment. Reid, Craig, and David: I love all of you very much and can't wait to spend more time with you now that I've finished.

I wish to express my appreciation to Lt Col Fred Harmon for taking me on as his first Ph.D. student and for providing guidance, direction, and mentorship. I began this program having graduated from my M.S. program more than 10 years prior and not having worked in a technical assignment in about nine years. To say that I was rusty is an understatement, and taking on a topic in stochastic optimal control that neither of us was familiar with or had experience in was quite risky. I think it has been a rewarding learning experience for both of us.

I wish to thank Dr. Richard Cobb and Dr. William Baker for being on my committee and for providing instruction in pseudospectral methods, constructive recommendations for my research work, and assistance in finding ways to present the results in ways that are meaningful to potential users. I especially thank Dr. Cobb for acting as my advisor while Lt Col Harmon was deployed. His thoroughness in reviewing a conference paper and the first draft of this dissertation helped me to produce documents that I am truly proud of.

Lastly, I want to express my gratitude to Dr. Dongbin Xiu and Dr. Akil Narayan at Purdue University. As leaders in uncertainty quantification research, their opinions

that this research combining GPM and gPC methods to solve stochastic optimal control problems adds valuable contributions to the community provided confirmation that this research was a worthy dissertation topic. I thank Dr. Xiu and Dr. Narayan for taking time to meet with me to help me understand and apply the gPC method in this research. I could not have successfully completed my research without their help.

Gerald C. Cottrill

Table of Contents

Page

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
LQR	Linear Quadratic Regulator	13
MIMO	Multi-Input Multi-Output	13
PMP	Pontryagin's Minimum Principle	14
LQE	Linear Quadratic Estimator	14
LQG	Linear Quadratic Gaussian	14
EL	Euler-Lagrange	16
PMP	Pontryagin's Minimum Principle	21
BVP	Boundary Value Problem	22
HBVP	Hamiltonian Boundary Value Problem	23
ODEs	Ordinary Differential Equations	25
NLP	Nonlinear Programming	25
KKT	Karush-Kuhn-Tucker	25
IVP	Initial Value Problem	26
MPBVP	Multi-Point Boundary Value Problems	27
FE	Forward Euler	27
BE	Backward Euler	27
LF	Leap Frog	27
TM	Trapezoidal Method	27
R-K	Runge-Kutta	29
A-B	Adams-Bashforth	29
A-M	Adams-Moulton	30
FEM	Finite Element Method	34
LG	Legendre-Gauss	36
LGR	Legendre-Gauss-Radau	36

Abbreviation		Page
LGL	Legendre-Gauss-Lobatto	36
CG	Chebyshev-Gauss	36
CGR	Chebyshev-Gauss-Radau	36
CGL	Chebyshev-Gauss-Lobatto	36
PDE	Partial Differential Equation	51
LPM	Legendre Pseudospectral Method	52
CPM	Chebyshev Pseudospectral Method	52
RPM	Radau Pseudospectral Method	52
MSE	mean squared error	64
MCS	Monte-Carlo Simulation	66
PDF	probability distribution function	66
RV	random variable	69
FEM	Finite Element Methods	71
DAE	Differential Algebraic Equation	71
PD	Percent Difference	91
NM	Nautical Miles	98
kts	Knots	98
POK	Probability of Kill	101
CDF	Cumulative Distribution Function	113

List of Symbols

Symbol		Page
ϵ	Error tolerance	31
τ_i	Collocation Point	34
Q_K	Polynomial of degree K	34
I_i	i^{th} subinterval of time domain	34
R	Defect or residual function	34
Z	Matrix of defects or residuals	35
ϕ_k	Orthogonal Polynomial Basis Function	35
L_i	Lagrange Polynomial	36
P	Legendre Polynomial	36
T	Chebyshev Polynomial	36
w_k	Gaussian quadrature weight at the k^th node	36
$\mathbf{z}$	Vector of generic design variables in NLP problem	39
$f(\mathbf{z})$	Objective functional in NLP problem	39
$\mathbf{h}(\mathbf{z})$	Vector of equality constraints in NLP problem	39
$\mathbf{g}(\mathbf{z})$	Vector of inequality constraints in NLP problem	39
l	Lagrangian functional used in NLP formulation	40
$\mathbf{L}(\cdot)$	Hessian matrix of the Lagrangian functional	40
$\mathbf{F}(\cdot)$	Hessian matrix of the objective functional	40
$\mathbf{H}(\cdot)$	Hessian matrix of the equality constraints	40
$\mathbf{G}(\cdot)$	Hessian matrix of the inequality constraints	40
α_k	NLP algorithm step size at the k^{th} iteration	41
$\mathbf{d}_k$	NLP algorithm search direction	41
C_i	Cardinal Function	45
$\delta_{i,j}$	Kronecker delta function	46

Symbol		Page
$\mathbf{z}$	Outputs or observables in gPC collocation formulation	72
Ω	Sample space of all possible events	72
$\mathcal{A}$	Sigma-algebra of the random space	72
$\mathcal{P}$	Probability measure	72
Γ	Finite domain of probability space	73
W^{i,d_i}	One-dimensional orthogonal polynomial space	73
d_i	Highest degree of one-dimensional polynomial basis set	73
$\{\phi_m^c(p_i)\}$	One-dimensional polynomial basis set	73
$W_N^{P_c}$	N-variate orthogonal polynomial space	73
P_c	Total degree of N-variate orthogonal polynomial space	73
$\mathbb{P}_N^{P_c}$	Projection operator from discrete probability space to polynomial approximation space	74
$\hat{\mathbf{z}}_m$	gPC expansion coefficients	74
q_i	Number of gPC collocation points in the i^{th} dimension	75
Q	Number of gPC collocation points in tensor grid	75
$\mathbf{p}_j$	j $^{\text{th}}$ collocation node of random vector p	76
α_j	Collocation weight associated with the j $^{\text{th}}$ node	76
$\mathbb{E}$	Expected value	83
var	Variance	83
cov	Covariance	83
A	Uncertain parameter used in concept demonstration problems	87
B	Uncertain parameter used in concept demonstration problems	87
μ	Mean value of a Gaussian PDF	88

Symbol		Page
σ	Standard deviation of a Gaussian PDF	88
S_{alg}	Hybrid algorithm statistical quantity being compared	91
S_{MCS}	MCS statistical quantity being compared	91
x_1	Position with respect to x_1 axis, NM	99
x_2	Position with respect to x_2 axis, NM	99
θ	Angle between x_1 axis and velocity vector, Rad	99
V	Speed, kts	99
R_{min}	Minimum turn radius, ft	100
μ_{i,x_1}	Mean or Expected Value of POK function in x_1 axis	101
μ_{i,x_2}	Mean or Expected Value of POK function in x_2 axis	101
σ_{i,x_1}	Standard deviation of POK function in x_1 axis	101
σ_{i,x_2}	Standard deviation of POK function in x_2 axis	101
$\mathcal{W}$	Mayer term weighting factor in TO problem cost functional	102
$\mathcal{N}(\mu, \sigma)$	Gaussian (normal) probability distribution with mean, μ, and standard deviation, σ	104
$\mathcal{U}(a, b)$	Uniform probability distribution on closed interval a,b	104
H	Hermite polynomial	105
L	Legendre polynomial	105
$\mathbf{x}_{ub}$	Upper bound trajectory	113
$\mathbf{x}_{lb}$	Lower bound trajectory	113

HYBRID SOLUTION OF STOCHASTIC OPTIMAL CONTROL PROBLEMS USING GAUSS PSEUDOSPECTRAL METHOD AND GENERALIZED POLYNOMIAL CHAOS ALGORITHMS

I. Introduction

1.1 Introduction

Finding solutions to optimization problems is essential to planning and conducting military operations. Virtually every application of United States Air Force (USAF) air- and space-based weapons systems, ranging from Remotely Piloted Aircraft (RPA), manned fighter, bomber, and transport aircraft, to satellite systems is planned to maximize desired effects while minimizing costs and risks. Some real-world examples of Trajectory Optimization (TO) and Optimal Control (OC) problems are:

- Find the optimal orbit and body orientation that allows an RPA to maximize the time its camera has *eyes on target*.
- Find the flight path through a threat-rich environment that minimizes a bomber's exposure to threats while conducting a bombing run.
- Find the orbit and thrust (throttle) inputs that maximize loiter time of an air refueling or Intelligence, Surveillance, and Reconnaissance (ISR) platform.
- Find the optimal thrust profile that minimizes fuel expenditure during satellite orbital transfer.
- Find the best path through and urban environment that takes a vehicle from one point to another in minimum time while avoiding obstacles.
- Find the route that allows an attack aircraft to strike the maximum number of

targets in minimum distance traveled (military version of the travelling salesman problem).

Real-world TO and OC problems like these are typically solved using deterministic models, which are often simplified representations of a system's true physics that neglect several potential sources of uncertainty. Higher-fidelity models may be cost-prohibitive to develop, not feasible due to unknown physics, or difficult to use in engineering work. Additionally, virtually all sensors used to measure system states, for example speed, altitude, latitude, longitude, etc., are corrupted by noise thus inducing measurement errors that can degrade the validity of TO or OC solutions. The environment can also introduce randomness through phenomena such as wind gusts and turbulence. Therefore, all real-world problems have uncertainties that can make it difficult to find effective solutions and cause deterministic methods to fail to meet objectives.

Nonlinear optimization problems are typically too complex to use analytical solution techniques necessitating use of numerical methods. When stochastic elements are included, numerical methods become essentially the only way to solve these problems. Recently, stochastic nonlinear problems have become an area of research emphasis in the Air Force Research Laboratory (AFRL) both in terms of basic research and applied research to address Micro Air Vehicle (MAV) control challenges. In a briefing given at the 2008 International Symposium on Unmanned Aerial Vehicles, AFRL described its MAV research program [2] highlighting several "fundamental scientific challenges," including "unsteady aerodynamics at low Reynolds number[s]" and listed several engineering and design challenges including:

- Determine what aerodynamic models should be developed/used for inner-loop flight control
- Gust tolerance for MAV operations near walls and other obstacles

- Stabilization of low inertial flight vehicles [small, lightweight] in high disturbance, gusty environments
- Nonlinear responses

AFRL's briefing indicates that MAVs are nonlinear systems subject to random (stochastic) elements such as gusts and model uncertainties that are excellent candidates for future application of the stochastic optimal control method described in this dissertation. A 2009 Air Force Office of Scientific Research (AFOSR) briefing describes its dynamics and control research portfolio where the recurring theme is to investigate numerical methods to solve nonlinear stochastic optimal control problems [64]. Combined, these presentations emphasize that numerical tools are necessary to solve stochastic nonlinear optimization problems to support MAV control development and solve other real-world problems.

Two pseudospectral numerical methods, the Gauss Pseudospectral Method (GPM) and Generalized Polynomial Chaos (gPC), may both be useful in addressing stochastic optimization problems like those of interest to AFOSR and AFRL. The GPM is a powerful deterministic numerical method for solving both linear and nonlinear TO and OC problems. It is a direct solution method that uses pseudospectral discretization and Gaussian quadrature to recast the continuous-time problem into a nonlinear programming (NLP) problem that can be solved using existing NLP solvers. The gPC, sometimes referred to as a *deterministic sampling method*, is an equally powerful method for solving stochastic differential equations (SDE) where collocation points in the random domain are used as inputs to stochastic differential equations resulting in a set of deterministic problems that can be solved using existing differential equation solvers. The set of sampled deterministic solutions are then used in the gPC expansion to quantify the effects of the uncertain parameters on the solution.

1.2 Research Objective and Technical Contribution

The objective of this research is to develop a new method of analyzing the effects of uncertain parameters on optimal control and trajectory optimization problem solutions and apply it to two representative optimization problems to demonstrate its ability to quantify the effects of uncertainty on the solution. The approach used in this research is the first time the GPM and the collocation form of the gPC have been combined to form a hybrid algorithm. Using the GPM in the hybrid scheme provides spectrally accurate minimum cost solutions to the sampled deterministic problems that meet specified constraints. This combination of deterministic and stochastic pseudospectral methods provides a new numerical algorithm for addressing nonlinear optimization problems where model, measurement, and environmental disturbance uncertainties may be included in the formulation, and their effects assessed by constructing distribution functions showing dependence of the solutions on uncertain parameters. Thus, the main technical contribution of this research is to extend current gPC methods to be applicable to stochastic optimal control and trajectory optimization problems.

1.3 Concept Demonstration Problem Descriptions

The hybrid numerical algorithm combining the GPM and gPC methods presented in this document is applied to two types of optimization problems to demonstrate its potential to quantify the effects of uncertainty on TO and OC problem solutions.

The first is a general OC problem taken from a textbook and modified by adding Gaussian random parameters effecting the state variables. The problem has nonlinear dynamics, quadratic Lagrange performance index, fixed final state and time, and no path or control constraints. The objective of this problem is to investigate the feasibility of using the hybrid algorithm to solve optimization problems by applying

it to a very challenging nonlinear problem where uncertainties have significant effects on the solution.

The second problem, and really the main focus of this research, is based on a more operationally representative mission planning scenario that is of interest to United States Strategic Command (USSTRATCOM). The problem was formulated in response to USSTRATCOM's interest in using solutions calculated by the method presented in this work to initialize their mission planning software, with hopes of generating mission plans more efficiently. The sample problem is a trajectory optimization problem where the objectives are to find the path through a two-dimensional space that minimizes the probability a vehicle will be *killed* by lethal threats whose locations are uncertain, and then to quantify the effects those uncertainties have on the solution by estimating the statistical properties. A generic two-dimensional *Dubins* model consisting of three states and one bounded control is used in formulating the sample problem. The Dubins model was chosen in lieu of a specific aircraft model to focus on algorithm development while avoiding additional complexities of developing and implementing a dynamics model for a particular vehicle. Bi-variate Gaussian probability density functions are used to model lethality distributions of the threats in the space, i.e. *threat rings*, and construct the nonlinear cost functional to be minimized in seeking solutions to the trajectory optimization problem. Random variables (RV) are incorporated into the cost functional to represent uncertainties in the exact center locations of the threat rings, making the cost functional and associated state and control solutions themselves RVs whose distribution properties are estimated by the gPC expansion coefficients.

In both concept demonstration problems, applying the hybrid algorithm results in quantification of the effects of the random parameters on the solution by generating a polynomial distribution function of the random inputs whose coefficients are used to

calculate expected value, variance, and covariance properties. An additional benefit is that if uncertain variables become certain, the distribution function can be evaluated at those points to provide specific optimal solutions.

1.4 Document Organization

This document is organized to present work related to solving optimal control problems, discuss application of the algorithm to two concept demonstration problems, and to recommend future research activities. Chapter II presents work found in the literature related to solving optimal control problems and stochastic differential equations. It begins with a brief history of the development of optimal control theory, defines an optimal control problem and describes the analytical solution approach based on calculus of variations theory, and discusses numerical methods available to solve optimal control problems, including a detailed description of the Gauss Pseudospectral Method. Readers who are familiar with optimal control theory may wish to skip directly to the Gauss Pseudospectral Method discussion in section 2.3.3.2. The chapter ends with a summary of methods used to solve stochastic differential equations and a detailed description of the Generalized Polynomial Chaos method. Chapter III gives a brief presentation of the hybrid algorithm combining the GPM and gPC method as it is applied to the demonstration problems in later chapters. The algorithm is presented as a step-by-step process and a code map of the MATLAB®implementation is included. Formulations and results of applying the hybrid algorithm to the concept demonstration problems are presented in Chapters IV and V, respectively. The document concludes with recommendations for further research in Chapter VI.

II. Related Concepts

The objective of OC theory is "to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criterion" [54]. The performance criterion is defined by an integral performance index, also referred to as a cost functional, that is to be minimized and the physical constraints are the system or process dynamics to be controlled. The idea is to cause the system to follow some ideal, or optimal, state trajectory by solving for the input, or control, that causes the system to follow the trajectory that minimizes the performance index. The performance index can be chosen to solve a variety of problems such as minimum time, minimum fuel consumption, minimum energy expended or control effort used, minimum error between desired and achieved end state (terminal control), minimum error between desired and achieved path (trajectory following), and so on. This chapter focuses on the body of work related to solving OC problems. The chapter begins with a brief historical summary of the development of OC theory. It continues by defining an OC problem and the classical analytical method of solving the problem stemming from the historical development. The chapter continues with a summary of numerical techniques used to solve OC problems, culminating with a description of a powerful numerical technique, the Pseudospectral Method (PSM) for approximating the solution. Next, related work is presented on solving SDEs along with current research applications. The chapter concludes by linking the theory of OC and the methods of solving SDEs that appear applicable to solving stochastic optimal control problems, which will set the stage for the work to follow in subsequent chapters.

2.1 A Brief History of Optimal Control Theory

Queen Dido of Carthage is generally thought of as to have solved one of the first optimization problems. Legend has it that she was promised the amount of land that could be enclosed using a bull's hide. Her solution was to cut the hide into thin strings and tie them together to form a circle [54]. To the ancient Greeks, the circle had to be the best solution because in Aristotle's thinking it represented the "perfect figure" [85]. She solved this problem before the mathematical development of the Calculus of Variations (CV), which proves that the optimal closed curve that maximizes the area is in fact a circle, and forms the mathematical basis for solving OC problems.

The first OC problem was posed by Johann Bernoulli (1667-1748) in 1696 when he issued a challenge to his contemporaries to solve the brachystochrone (from two Greek words meaning "shortest" and "time") problem [85]. Bernoulli's brachystochrone problem was really a re-statement of the problem posed by Galileo Galilei (1564-1642) in 1638 whose thoughts on the solution later proved to be incorrect [16, 85]. The goal of the brachystochrone problem was to "find the shape of a wire such that a bead sliding along it traverses the distance between the two end points in minimum time" [16]. Bernoulli noted in his challenge that the shortest distance between the end points would be a straight line, but the minimum time to travel from one end of the wire to the other would not be achieved by following the straight line path. The five mathematicians who solved the problem were Johann Bernoulli himself, his brother Jakob (1654-1705), Wilhelm Gottfried Leibniz (1646-1716), Marquis de l'Hôpital (1661-1704) and Isaac Newton (1642-1727). It's noteworthy to point out that Jakob Bernoulli's solution resembled future concepts of CV, Hamilton-Jacobi theory, and Dynamic Programming (DP) [21, 85]. This was among the earliest of OC problems since it dealt with finding the control input, in this case the shape of the

curved wire, that controlled the behavior of a dynamical system [85].

Willems notes that things were relatively quiet after the solution of Bernoulli's brachystochrone problem until about the 1960's [85]. The quiet period was probably due to the fact that a rigorous mathematical structure, known today as CV, needed to evolve in order to solve OC problems, which was developed in the 18^{th} and 19^{th} centuries [32].

Leibniz and Newton are said to have co-invented CV. Leibniz was interested in finding extrema of functions and published a paper titled *A new method for the determination of maxima and minima...* in 1684 [55]. Newton solved one of the first problems using CV when he correctly posed and solved a problem to find the shape of a projectile moving through air that resulted in minimum drag in 1685, and published in 1694 [16, 32, 55]. At the time, there was a need for the theory of Calculus to be extended to address more general problems involving finding paths, curves, or surfaces that result in stationary values (maxima or minima) of functionals [32, 55]. The work of Leibniz, Newton, and the solutions to Bernoulli's brachystochrone problem in 1696 can be thought of as the beginnings of CV.

Many early problems were shown to be of similar form, generally defined by the functional I and boundary conditions given as [79]:

$$\begin{aligned} I &= \int_a^b L(q(t), \dot{q}(t), t)dt \\ q(a) &= q_a \ , q(b) = q_b \\ a &\leq t \leq b \end{aligned} \tag{2.1}$$

Addressing the problem in (2.1) meant finding a function $q(t)$ that resulted in a stationary I, indicating either a maximum or minimum, while satisfying the boundary conditions [55]. Leonard Euler (1707-1783) was the first to formally study this

problem in the late 1720's and 1730's and published a book in 1744, referred to as the "birth year of the theory" of CV, entitled *A method for discovering curved lines that enjoy a maximum or a minimum property, or the solution of the isoperimetric problem taken in the widest sense* [32, 55]. An excellent detailed discussion of Euler's method can be found in [47]. Euler's main contribution to the development of CV theory was derivation of what is known as the *Euler equation*. Euler found that a necessary condition for $q(t)$ satisfying (2.1) to be a minimum curve had to also satisfy:

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}}\right] = 0 \tag{2.2}$$

Joseph Louis Lagrange (1736-1813) extended Euler's work by inventing his "method of variations" and his multiplier rule, published in 1760 in a paper entitled *Essay on a new method for determining maxima and minima for formulas of indefinite integrals*. Using these methods, he studied the first variation, denoted as δI, of the functional in (2.1). He concluded that the extremal solution $q(t)$ was obtained when the first variation vanished, analogous to solving for $f'(x) = 0$ in the case of finding maxima or minima of a function using traditional calculus. He showed that when the first variation vanished, Euler's equation was left, more convincingly showing that Euler's equation was a necessary condition for finding the function $q(t)$ that leads to an extremal value of I. Subsequently, Euler gave the field the name *Calculus of Variations* in honor of Lagrange's variational method and equation (2.2) became known as the *Euler-Lagrange equation*. The significance of Lagrange's work was to generalize Euler's geometric approach to applied problems by focusing on "algorithmic aspects of analysis" based on differential equations [32, 55].

In calculus, the second derivative test provides a necessary condition for determining if extremal points are maxima or minima. Likewise, Euler's equation and examining the first variation of the functional only provides the necessary condition

for a maximum or minimum curve. The second variation, $\delta^2 I$, provides the analogous test for determining maximum and minimum curves in CV. Adrien-Marie Legendre (1752-1833) was the first to look at the second variation of the functional and determined, without complete proof, that $L_{\dot{q}\dot{q}} \geq 0$ along a minimizing curve and $L_{\dot{q}\dot{q}} \leq 0$ (subscripts denote partial derivatives) along a maximizing one, as published in 1786. Karl Gustav Jacob Jacobi (1804-1851) later provided rigorous proof that $L_{\dot{q}\dot{q}} > 0$ along with the *Jacobi condition* provide sufficiency for showing $q(t)$ is a minimum curve [16, 32, 55].

Sir William Rowan Hamilton (1805-1865) was the next to add to the theory of CV. In papers he wrote in 1834 and 1835 he "showed that under certain conditions, problems in mechanics involving many variables and constraints can be reduced to an examination of the partial derivatives of a single function" [16, 32]. Hamilton's function was:

$$\mathcal{H} =< p, \dot{q} > -L(q, \dot{q}, t) \tag{2.3}$$

Where $p = L_{\dot{q}}(q, \dot{q}, t)$ and the $< p, \dot{q} >$ term denotes an inner product. Hamilton found that looking at $\mathcal{H}_p = \dot{q}$ and $-\mathcal{H}_q = \dot{p}$ was equivalent to (2.2) under the assumptions that $\dot{q}$ is treated not as an independent variable but as a function of q, p, and t and that the p equation can be solved for $\dot{q}$ [79]. Sussmann and Willems noted that Hamilton should have written his function in another way. Letting $u(t) = \dot{q}(t)$, defining the Hamiltonian, $\mathcal{H}$, as:

$$\mathcal{H}(q, u, p, t) =< p, u > -L(q, u, t) \tag{2.4}$$

and $p(t)$ as:

$$p(t) = L_u(q(t), \dot{q}(t), t) \tag{2.5}$$

leads to:

$$\begin{aligned} \mathcal{H}_p &= \dot{q} \\ -\mathcal{H}_q &= \dot{p} \\ \mathcal{H}_u &= 0 \end{aligned} \tag{2.6}$$

This is a more general result and is directly equivalent to (2.2). This result is referred to as the "control Hamiltonian" and Sussmann and Willems state, "the control version of the Hamiltonian equations is equivalent to the Euler-Lagrange system under completely general conditions...". Equation (2.4) will be the basis for finding the analytical OC solution presented in the next section.

Hamilton's proof was lacking details that Jacobi added in 1838. Jacobi was doing similar work at about the same time as Hamilton that showed that "the partial derivatives of the performance index with respect to each parameter of a family of extremals (which today we call 'states') obeyed a certain differential equation" [16]. Jacobi cleaned up Hamilton's results and simplified them [16, 32]. We now know this theory as Hamilton-Jacobi theory, which became the basis of DP developed by Bellman in the 1950's.

In the late 19^{th} and the first half of the 20^{th} centuries, mathematicians like Karl Theodor Wilhelm Weierstrass (1815-1897), Oskar Bolza (1857-1942), and Gilbert Bliss (1876-1951) added mathematical rigor to the theory of CV. Weierstrass discovered a condition which became the predecessor to maximum principles later stated by Bellman and Pontryagin and was the first to develop a complete sufficiency theorem for a minimum. Bolza and Bliss built upon Weierstrass' work at the University of Chicago giving CV its current rigorous mathematical structure, culminating in publication of the book *Lectures on the Calculus of Variations* in 1946 [16, 32].

The evolved CV theory was the enabling structure for OC theory to progress beginning in the 1950's. OC theory is a generalization of CV and allows nonlinear and non-smooth functionals to be optimized [32]. Bryson gives an outstanding summary of the roots of OC theory from 1950 to 1985 in [16]. His paper discusses the roots of OC theory in contexts of classical control, random processes, linear and nonlinear programming, and DP. The rest of the historical discussion of OC theory in this section is a summary of Bryson's main points.

Until the years surrounding World War II (WWII), classical linear control methods, such as Proportional (P), Proportional + Integral (PI), Proportional + Integral + Derivative (PID), Lead, Lag, and Lead-Lag involved a degree of ad-hoc design of control gains. Control gains were selected and adjusted based on observed performance until some acceptable result was achieved, leaving a considerable degree of subjectiveness to control design. During the WWII period, several methods were developed to systematically choose control gains including Evans' Root Locus (RL) method, the Bode plot method, and the Nyquist plot method. These methods, and others, allowed designers to select gains based on some desired stability and performance criteria. In the 1960's, Kalman introduced the Linear Quadratic Regulator (LQR) method using an integral performance index that placed quadratic penalties on output errors and amount of control used [51, 52]. He used CV to show that the OC inputs can be found using linear feedback of the state variables and was able to apply this method to time-varying and Multi-Input Multi-Output (MIMO) systems. Kalman further showed that the optimal state feedback gain matrix could be found by solving a backward Riccati equation [16]. His contribution was to replace ad-hoc gain selection with a method to solve for optimal feedback control gains.

Some would say that OC theory was truly born in the 1950's and 1960's with the publication of *The Mathematical Theory of Optimal Processes* [69] in 1961 by

Soviet mathematician Lev Semenovich Pontryagin (1908-1988) and a group of his students [79]. Pontryagin's well developed theory forms the basis for modern optimal control, is applicable to linear and nonlinear problems, and is usually the benchmark for comparing results produced by numerical methods developed later. His theory extended CV and Weierstrass' necessary condition to address inequality constraints and stated that a minimizing path must satisfy the Euler-Lagrange equations (2.2) and that the OC solution maximizes the Hamiltonian (2.3) in their bounded region at each point along the curve [16]. Pontryagin used a different sign convention on the Hamiltonian, thus the term *maximum principle*, but the principle is applied to find minimizing solutions of the Hamiltonian and is thus referred to as Pontryagin's Minimum Principle (PMP).

The field of OC is closely tied to the study of optimal signal filtering. Models representing a system's dynamics are typically assumed to be *noise-free*. That is to say that an assumption is made that the state of the system represented by a set of differential equations is perfectly known. In practice, models are a simplification of the true physical nature of the system and measurements of system states are noise-corrupted. Therefore, some method must be implemented to extract the best possible estimate of the state with the presence of model and measurement uncertainties. Pioneers in the study of optimal filtering were Norbert Wiener (1894-1964) in the 1940's [84] and Ruldolf Kalman and Richard Bucy in the 1950's and 1960's [51, 52]. These researchers essentially developed optimal techniques to minimize errors between the actual measurements and the estimated system states obtained from the dynamical models. Kalman showed that optimal filter gains are obtained by solving a forward Riccati equation and became known as the Linear Quadratic Estimator (LQE) [16]. The LQE combined with the LQR method became known as the Linear Quadratic Gaussian (LQG) compensator. The LQG compensator, an OC scheme

that is applicable to linear time-varying systems with uncorrelated Gaussian random inputs, feeds back optimal state estimates that are then used to solve the OC problem which minimizes the quadratic performance index [16].

Development of LQR, LQE, LQG, and Wiener filtering methods resulted in significant advancements in the OC field. However, these methods are primarily focused on linear systems. The power of the CV applied to OC problems is that it can be used to solve nonlinear problems. The difficulty, however, is that applying Euler-Lagrange, Hamilton-Jacobi, and PMP theories may result in differential equations where an analytical solution may not be discernible. The advent of the digital computer in the mid-1950's enabled development of numerical methods to solve these problems.

At this point the historical summary of OC theory will be left, not because there are not significant contributions, but rather because surveying significant accomplishments is not the main focus of this chapter, and in fact could be the topic of study in itself taking significant time to study, analyze, and summarize. Suffice it to say that efforts from the 1950's to present focus on numerical techniques to solve problems that were previously intractable, analytically or numerically, with pencil and paper analysis, and on showing that the numerical techniques produce results that are complimentary, if not equivalent, to CV-based solutions. Bellman's principle of optimality and a sequential decision making method, known as Dynamic Programming, was developed in the 1950's and is still widely used [16]. Following dynamic programming, nonlinear programming, based on gradient search methods, was developed in the 1960's [16], and spectral methods were developed in the 1970's and more prolifically applied to OC problems in the 1980's and 1990's [7].

This has been a brief survey of the history of theory related to solving OC problems. It is by no means exhaustive, and was never intended to be. The intent of this brief survey was to give a historical context for methods that will be discussed in this

document for solving OC problems. In the next section, the general OC problem will be stated and the CV-based formulation of the solution will be presented.

2.2 Optimal Control Problem Defined

In this section the general continuous-time OC problem is stated and the general procedure for solving the problem is summarized. Many textbooks have been written on the subject such as [54] and [17] and contain derivation of the Euler-Lagrange (EL) equations using CV principles. The concepts in this section are summarized from [54], [75], [7], and [50]. The notation used is a combination of that used in [54], [72], [7], and [50], attempting to match the most commonly used notation in current literature. References [54] and [48] also present equivalent development of discrete-time Euler-Lagrange equations, which will not be summarized here.

2.2.1 Formulation.

The objective of an OC problem is to find an *admissible* control function $\mathbf{u}^*(t)$ that transfers a system from some initial state $\mathbf{x}_0$ at the initial time t_0 to some final state $\mathbf{x}_f$ at the final time t_f following an *admissible* state trajectory $\mathbf{x}^*(t)$ that minimizes a performance index, J, of the form:

$$J(\mathbf{x}(t), \mathbf{u}(t), t) = \Phi(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt \tag{2.7}$$

The term *admissible* is used to indicate that the optimal solutions $\mathbf{u}^*(t)$ and $\mathbf{x}^*(t)$ lie within acceptable upper and lower limits. The "*" superscript notation is commonly used in many texts to distinguish the solution that minimizes the functional J. The first term in (2.7), $\Phi(\cdot)$, is known as the Mayer term and the integral term, $\int_{t_0}^{t_f} g(\cdot)dt$, is known as the Lagrange term. The cost functional is said to be in Bolza form when both terms are present. The state vector, $\mathbf{x}(\cdot)$, is in $\mathbb{R}^n$, the control vector, $\mathbf{u}(\cdot)$, is in

$\mathbb{R}^m$. In vector form, $\mathbf{x}(t)$ and $\mathbf{u}(t)$ can be written as:

$$\mathbf{x}(t) \triangleq \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \quad \mathbf{u}(t) \triangleq \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}$$

The performance index (2.7) minimization is subject to the dynamical constraints of the system to be controlled, generally represented as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.8}$$

The boundary conditions, $\boldsymbol{\phi}(\cdot)$, in general form can be stated as:

$$\boldsymbol{\phi}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = \mathbf{0} \tag{2.9}$$

Some of the boundary conditions that may be specified are summarized below.

- **Fixed Final Time** - The terminal time may be specified with $\mathbf{x}(t_f)$ fixed, $\mathbf{x}(t_f)$ free, or $\mathbf{x}(t_f)$ on a surface $\mathbf{m}(\mathbf{x}(t)) = 0$.
- **Free Final Time** - The terminal time may not be specified with $\mathbf{x}(t_f)$ fixed, $\mathbf{x}(t_f)$ free, $\mathbf{x}(t_f)$ on a moving point $\theta(t)$, $\mathbf{x}(t_f)$ on a surface $\mathbf{m}(\mathbf{x}(t)) = 0$, or $\mathbf{x}(t_f)$ on a moving surface $\mathbf{m}(\mathbf{x}(t), t) = 0$.

Constraints may also be placed on the path the admissible solution can take, $\mathbf{C}(\cdot)$, including boundaries on admissible $\mathbf{x}(t)$ trajectories, generally represented by the inequality constraints:

$$\mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{0} \tag{2.10}$$

If the set of admissible controls is limited to some bounded region U, written as, $\mathbf{u}(t) \in U$, then the inequality constraints placed on the control can be written as:

$$u_{i,min} \leq u_i(t) \leq u_{i,max} \tag{2.11}$$

where $i = 1, ..., m$. The functional mappings of $\Phi(\cdot)$, $g(\cdot)$, $f(\cdot)$, $\phi(\cdot)$, and $\mathbf{C}(\cdot)$ in equations (2.7) - (2.10) are given by:

$$\begin{array}{lcl} \Phi(\mathbf{x}(t_f), t_f) & : & \mathbb{R}^n \times \mathbb{R} \to \mathbb{R} \\ g(\mathbf{x}(t), \mathbf{u}(t), t) & : & \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R} \\ f(\mathbf{x}(t), \mathbf{u}(t), t) & : & \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n \\ \boldsymbol{\phi}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) & : & \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^q \\ \mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t) & : & \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^c \end{array} \tag{2.12}$$

A brief development of the solution method based on CV techniques is presented next. Equations to apply boundary conditions will be presented for three of the cases listed above; t_f specified (fixed) with $\mathbf{x}(t_f)$ unspecified (free), t_f free with $\mathbf{x}(t_f)$ fixed, and both t_f and $\mathbf{x}(t_f)$ fixed. Equations needed to apply other types of boundary conditions can be found in [54].

2.2.2 Classical Analytical Solution.

With the general OC problem defined in the previous section, a general solution procedure based on CV can be obtained. The development that follows is a combination of information found in [7, 50, 54, 72]. The necessary conditions for a minimizing solution are obtained by looking at the first variation of the functional in (2.7) augmented by the differential equality constraints (system dynamics), inequality path constraints ($\mathbf{C}(\cdot)$), and boundary conditions ($\phi(\cdot)$), denoted as J_a. Lagrange multipliers $\boldsymbol{\nu} \in \mathbb{R}^q$, $\boldsymbol{\lambda}(t) \in \mathbb{R}^n$, referred to as the costate vector, and $\boldsymbol{\mu}(t) \in \mathbb{R}^c$ are

introduced to augment the cost functional with the boundary conditions, dynamics, and path constraints respectively. The augmented cost functional is given as:

$$\begin{aligned} J_a &= \Phi(\mathbf{x}(t_f), t_f) - \boldsymbol{\nu}^T \boldsymbol{\phi}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) \\ &\quad + \int_{t_0}^{t_f} \{g(\mathbf{x}(t), \mathbf{u}(t), t) - \boldsymbol{\lambda}^T(t) \left[\dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)\right] \\ &\qquad - \boldsymbol{\mu}^T \mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t)\} dt \end{aligned} \tag{2.13}$$

Taking the first variation of J_a, represented as δJ_a, following procedures similar to those found in [7, 50, 54] gives:

$$\begin{aligned} \delta J_a = \quad & [\Phi_{\mathbf{x}}(t_f) - \boldsymbol{\nu}^T \boldsymbol{\phi}_{\mathbf{x}}(t_f)]\delta \mathbf{x}_f - [\boldsymbol{\nu}^T \boldsymbol{\phi}_x(t_0)]\delta \mathbf{x}_0 - \delta \boldsymbol{\nu}^T \phi + [\Phi_t(t_f) \\ & - \boldsymbol{\nu}^T \phi_t(t_f) + g(t_f) - \boldsymbol{\lambda}^T(t_f)(\dot{\mathbf{x}}(t_f) - \mathbf{f}(t_f)) - \boldsymbol{\mu}^T(t_f)\mathbf{C}(t_f)]\delta t_f \\ & + [-\boldsymbol{\nu}^T \boldsymbol{\phi}_t(t_0) - g(t_0) + \boldsymbol{\lambda}^T(t_0)(\dot{\mathbf{x}}(t_0) - \mathbf{f}(t_0) + \boldsymbol{\mu}^T(t_0)\mathbf{C}(t_0)]\delta t_0 \\ & + \int_{t_0}^{t_f} \{(g_x + \boldsymbol{\lambda}^T \mathbf{f}_{\mathbf{x}} - \boldsymbol{\mu}^T \mathbf{C}_{\mathbf{x}})\delta \mathbf{x} + (g_u + \boldsymbol{\lambda}^T \mathbf{f}_u - \boldsymbol{\mu}^T \mathbf{C}_{\mathbf{u}})\delta \mathbf{u} \\ & - \delta \boldsymbol{\lambda}^T(\dot{\mathbf{x}} - \mathbf{f}) - \delta \boldsymbol{\mu}^T \mathbf{C} - \boldsymbol{\lambda}^T \delta \dot{\mathbf{x}}\} dt \end{aligned} \tag{2.14}$$

Notice that the arguments of the functions in (2.14) were left out for readability. It's clear from (2.13) what the functional dependencies are. Next, define the Hamiltonian functional:

$$\begin{aligned} \mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), \boldsymbol{\mu}(t), t) &= g(\mathbf{x}(t), \mathbf{u}(t), t) + \boldsymbol{\lambda}^T(t)\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ &\quad - \boldsymbol{\mu}^T(t)\mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t) \end{aligned} \tag{2.15}$$

Substituting (2.15) along with the partial derivatives $\mathcal{H}_{\mathbf{x}}$, $\mathcal{H}_{\boldsymbol{\lambda}}$, and $\mathcal{H}_{\mathbf{u}}$ into (2.14), using the following integration by parts formula on the $\delta\dot{\mathbf{x}}$ term:

$$\int_{t_0}^{t_f} -\boldsymbol{\lambda}^T \delta\dot{\mathbf{x}}\ dt = -\boldsymbol{\lambda}^T(t_f)\delta\mathbf{x}(t_f) + \boldsymbol{\lambda}^T(t_0)\delta\mathbf{x}(t_0) + \int_{t_0}^{t_f} \dot{\boldsymbol{\lambda}}^T \delta\mathbf{x}\ dt \tag{2.16}$$

incorporating the following identities for $\delta\mathbf{x}_f$ and $\delta\mathbf{x}_f$:

$$\begin{aligned} \delta\mathbf{x}_f &= \delta\mathbf{x}(t_f) + \dot{\mathbf{x}}(t_f)\delta t_f \\ \delta\mathbf{x}_0 &= \delta\mathbf{x}(t_0) + \dot{\mathbf{x}}(t_0)\delta t_0 \end{aligned} \tag{2.17}$$

and simplifying yields:

$$\begin{aligned} \delta J_a \quad &= [\Phi_{\mathbf{x}}(t_f) - \boldsymbol{\nu}^T\boldsymbol{\phi}_{\mathbf{x}}(t_f) - \boldsymbol{\lambda}^T(t_f)]\delta\mathbf{x}_f + [-\boldsymbol{\nu}^T\boldsymbol{\phi}_x(t_0) + \boldsymbol{\lambda}^T(t_0)]\delta\mathbf{x}_0 \\ &-\delta\boldsymbol{\nu}^T\phi + \{\mathcal{H}(t_f) - \Phi_t(t_f) - \boldsymbol{\nu}^T\phi_t(t_f)\}\delta t_f + \{-\mathcal{H}(t_0) - \boldsymbol{\nu}^T\boldsymbol{\phi}_t(t_0)\} \\ &\delta t_0 + \textstyle\int_{t_0}^{t_f}\{[\mathcal{H}_{\mathbf{x}} + \dot{\boldsymbol{\lambda}}^T]\delta\mathbf{x} + \mathcal{H}_{\mathbf{u}}\delta\mathbf{u} + (\mathbf{f} - \dot{\mathbf{x}})\delta\boldsymbol{\lambda}^T - \mathbf{C}\delta\boldsymbol{\mu}^T\}dt \end{aligned} \tag{2.18}$$

The EL equations needed to solve for the OC and the associated optimal state trajectory come from the integrand of (2.18). The variational terms inside the integrand are arbitrary, thus the necessary conditions for a minimum are found by finding the coefficients that cause the variations $\delta\mathbf{x}$, $\delta\mathbf{u}$, and $\delta\boldsymbol{\lambda}$ to vanish. Therefore, the necessary conditions that must be satisfied for a minimizing solution, regardless of the boundary conditions, are:

- **State equations**: The $\delta\boldsymbol{\lambda}$ term becomes zero if $(\mathbf{f} - \dot{\mathbf{x}}) = \mathbf{0}$. Therefore, the state equations are found by taking the partial derivative $\mathcal{H}_{\boldsymbol{\lambda}}$ of the Hamiltonian. This may seem a little redundant since the state equations are given as differential equality constraints in (2.8), but confirms that the Hamiltonian has been constructed correctly and that the minimization depends on the system

dynamics.

- **Costate equations**: The $\delta\mathbf{x}$ term becomes zero if $(\mathcal{H}_{\mathbf{x}} + \dot{\boldsymbol{\lambda}}^T) = \mathbf{0}$. This condition defines a set of differential equations for the costates by taking the negative of the partial derivative $\mathcal{H}_{\mathbf{x}}$ of the Hamiltonian.
- **Control equations**: The $\delta\mathbf{u}$ term becomes zero if the partial derivative $\mathcal{H}_{\mathbf{u}}$ is zero. This generally produces an OC solution that is a function of the costates and possibly the states. This condition is true for an unbounded control. Pontryagin's Minimum Principle (PMP) will be discussed below to account for bounded controls.

The variational terms in (2.18) that are not inside the integral operator define how boundary conditions are applied. Generally speaking, if a boundary condition is specified, either initial, final, or both, then the variation related to that quantity is zero and the term drops out of equation (2.18). For example, if the initial time and initial state are given, then δt_0 and $\delta\mathbf{x}_0$ are $\mathbf{0}$. If a quantity is unspecified, or free, then the variation on that quantity is arbitrary and the minimizing solution must drive the coefficient on that term to zero. For example, if the final state is free, then $\delta\mathbf{x}_f$ is arbitrary, thus $[\Phi_{\mathbf{x}}(t_f) - \boldsymbol{\nu}^T\boldsymbol{\phi}_{\mathbf{x}}(t_f) - \boldsymbol{\lambda}^T(t_f)]$ must be $\mathbf{0}$. This reasoning allows for the definition of three common boundary conditions: *fixed final time and free final state*, *fixed final state and free final time*, and *fixed final state and fixed final time* as follows.

- **Fixed final time with free final state**: In this case, the coefficient on the $\delta\mathbf{x}_f$ term must be zero. Therefore:

$$\boldsymbol{\lambda}^T(t_f) = \Phi_x(t_f) - \boldsymbol{\nu}^T\boldsymbol{\phi}_x(t_f) \tag{2.19}$$

- **Fixed final state with free final time**: In this case, the coefficient on the

δt_f term must be zero, leading to:

$$\begin{aligned} \Phi_t(t_f) - \boldsymbol{\nu}^T \boldsymbol{\phi}_t(t_f) + g(t_f) - \boldsymbol{\mu}^T(t_f)\mathbf{C}(t_f) + \boldsymbol{\lambda}^T(t_f)\mathbf{f}(t_f) &= \mathbf{0} \quad (2.20) \\ \Phi_t(t_f) - \boldsymbol{\nu}^T \boldsymbol{\phi}_t(t_f) + \mathcal{H}(t_f) &= \mathbf{0} \end{aligned}$$

- **Fixed final state with fixed final time**: This is the simplest case where $\mathbf{x}(t_0) = \mathbf{x}_0$ and $\mathbf{x}(t_f) = \mathbf{x}_f$ provides the necessary information to determine the solution by treating the system of equations as a Boundary Value Problem (BVP).

Other boundary conditions may be specified as previously mentioned. A table of the relevant boundary condition equations is found in [54].

Looking at the first variation of J_a has revealed a procedure for finding necessary conditions for a minimum solution. The variational method effectively translates to finding the $\mathbf{u}^*(t)$ and the associated $\mathbf{x}^*(t)$ that minimizes the cost functional of (2.7) to finding the $\mathbf{u}^*(t)$ and the associated $\mathbf{x}^*(t)$ that minimizes the Hamiltonian in (2.15). The process can be stated as:

1. Build the **Hamiltonian** using (2.15)

2. Write the **state equations** as:

$$\dot{\mathbf{x}}^* = \frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t) \quad (2.21)$$

3. Define the **costate equations** as:

$$\dot{\boldsymbol{\lambda}}^* = -\frac{\partial \mathcal{H}}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t) \quad (2.22)$$

4. Express the **control equations** as:

$$\mathbf{0} = \frac{\partial \mathcal{H}}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t) \tag{2.23}$$

5. Simultaneously solve the system of Euler-Lagrange equations (2.21), (2.22), and (2.23) and apply the appropriate boundary condition equations from [54]. The boundary condition equations determine the terminal time and state, resulting in what is commonly referred to as the Hamiltonian Boundary Value Problem (HBVP). Note that (2.23) defines $\mathbf{u}^*(t)$ as a function $\boldsymbol{\lambda}^*(t)$, $\boldsymbol{\mu}^*(t)$, and $\mathbf{x}^*(t)$ which may be substituted into (2.21) and (2.22) to remove explicit functionality of $\mathbf{u}^*(t)$ if $\mathbf{u}^*(t)$ can be isolated.

The $\mathbf{u}^*(t)$ determined by (2.23) is a necessary condition for an optimum solution in the case where there are no bounds on the control or where the resultant $\mathbf{u}^*(t)$ never breaks the upper or lower limits on $\mathbf{u}(t)$. The necessary and sufficient conditions for $\mathcal{H}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t)$ to be a *global minimum* are that equations (2.21), (2.22), and (2.23) be satisfied (*necessary*) and that the $m \times m$ matrix $\mathcal{H}_{\mathbf{uu}}$ be positive definite (*sufficient*) [75]. In order to pick an admissible $\mathbf{u}^*(t)$, it's necessary to consider how to account for the bounded control in (2.11) using PMP. The necessary condition provided by PMP states that for all admissible controls $u(t) \in U$:

$$\mathcal{H}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t) \leq \mathcal{H}(\mathbf{x}^*(t), \mathbf{u}(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t) \tag{2.24}$$

This equation says that if there is an admissible OC signal $\mathbf{u}^*(t)$ then there exists an optimal costate vector $\boldsymbol{\lambda}^*(t)$ and vector of Lagrange multipliers $\boldsymbol{\mu}^*(t)$ multiplying the path constraints that satisfies (2.22) and (2.10) at every point in the time interval

$t \in [t_0, t_f]$ such that (2.24) is true [75]. That is to say that:

$$\min_{\mathbf{u} \in U} \mathcal{H}(\mathbf{x}^*(t), \mathbf{u}(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t) = \mathcal{H}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), \boldsymbol{\mu}^*(t), t) \tag{2.25}$$

The OC is the one out of the entire set of admissible controls that causes $\mathcal{H}$ to be its global minimum. In practice, applying the PMP involves solving for $\mathbf{x}^*(t)$, $\boldsymbol{\lambda}^*(t)$, and $\mathbf{u}^*(t)$ using equations (2.21) - (2.23), initially assuming unbounded controls, and then comparing the resultant $\mathbf{u}^*$ with the admissible region to see if there are areas where $\mathbf{u}^*$ is out of bounds. If so, then the control signal is re-evaluated using (2.25) until the minimum $\mathcal{H}$ is found that satisfies the given boundary conditions [54].

An alternate method of taking into account boundaries on the state trajectory is presented in [54], which may eliminate (2.10) from the problem (if $\mathbf{C}(\cdot)$ only consists of bounds on the states) without adding significant complexity. Kirk presents a procedure for transforming the c inequality constraints into a single equality constraint. The net effect is that there are two additional equations, one for $\dot{x}_{n+1}$ and one for $\dot{\lambda}_{n+1}$. Therefore, there are $n + 1$ equations in (2.21) and in (2.22) to solve. The procedure stays virtually the same as indicated above, but eliminates the Lagrange multipliers $\boldsymbol{\mu}$, and does not affect applying the PMP for determining the admissible control vector.

In general, analytically solving the system of differential equations (2.21) and (2.22) with substitutions made using (2.23) is only possible for very simple problems. These equations are typically coupled nonlinear systems where standard integral tables may not offer closed-form solutions and linear methods such as Laplace transforms are not helpful. Therefore, numerical methods are necessary to find the solution to most meaningful OC problems.

2.2.3 Formulation and Classical Solution Summary.

In this section, the OC problem was defined and the Calculus of Variations and Pontryagin's Minimum Principle were used to derive the Hamiltonian Boundary Value Problem, which defines the first-order necessary conditions for optimality. Solving the resulting set of differential equations indirectly solves the original OC problem by finding the solution that minimizes the Hamiltonian, which turns out to also be the minimizing solution to the original cost functional. Solving the differential equations of the HBVP analytically can be difficult, if not impossible, requiring numerical techniques to solve the problem. A survey of numerical techniques for solving the OC problem are described in the next section.

2.3 Numerical Methods for Optimal Control

There are two main categories of numerical approaches for solving the OC problem defined by (2.7) - (2.11): *indirect methods* and *direct methods*. Indirect methods are based on numerically solving the system of Ordinary Differential Equations (ODEs), known as the Hamiltonian Boundary Value Problem, derived by the CV approach described in section 2.2.2 by equations (2.21) - (2.25). Direct methods, on the other hand, translate the continuous-time OC problem into a nonlinear optimization problem, also called a Nonlinear Programming (NLP) problem, where gradient search methods are applied to determine the optimal state trajectories and control sequences by satisfying Karush-Kuhn-Tucker (KKT) optimality conditions [71]. This section will briefly describe typical methods used in both categories and then discuss PSMs used for solving OC problems.

2.3.1 Indirect Methods.

Indirect methods are based on solving the HBVP given in equations (2.21) - (2.23) subject to the given boundary conditions. The benefit of using the CV-based indirect approach is that it yields highly accurate results while providing assurances that the first-order optimality conditions are satisfied [50]. However, difficulties arise in that explicit derivations of the costate (2.22) and control (2.23) equations are required, which can be very difficult depending on the OC problem being considered, and prior knowledge of the activeness of inequality constraints (2.10) is necessary [78]. Numerical techniques applied to the indirect equations generally require good initial guesses of the costates, which is often difficult since the costates generally do not have direct physical interpretations [7, 50]. There are two main approaches to solving the problem: shooting and collocation methods. Shooting methods are iterative techniques that enable solution of the BVP using Initial Value Problem (IVP) techniques while collocation methods allow simultaneous solution of unknown parameters [71]. Consider the general problem of finding the solution $x(t)$, where $t \in [t_0, t_f]$, to an IVP of the form:

$$\begin{aligned} \frac{d}{dt}x(t) &= f(x(t), t) \\ x(t_0) &= x_0 \end{aligned} \tag{2.26}$$

Equation (2.26) is presented here as a single ODE, but can in general represent a system of ODEs, which will be the case when discussing shooting methods. The solution to the IVP involves breaking $[t_0, t_f]$ into n subintervals, where the i^{th} subinterval is $[t_i, t_{i+1}]$ with length $h_i = t_{i+1} - t_i$. In many applications, the lengths of the n subintervals are equal and the size is denoted as h, but it is not necessary that the subinterval lengths be equal. The objective is to find the solution at each i^{th} node as

given in [71], which can be written as:

$$x_{i+1} = x_i + \int_{t_i}^{t_{i+1}} f(x(t), t)dt \tag{2.27}$$

The techniques in this section are designed to estimate the integral in (2.27). The subsections below outline time-marching algorithms to solve the IVP with extension to shooting methods to solve a BVP and collocation techniques to solve the BVP.

2.3.1.1 Time-Marching and Shooting Methods.

A time-marching technique solves the IVP by using the solution of x_i ($x(t)$) and possibly more previous points to determine the subsequent x_{i+1} ($x(t+h)$) solution. There are many ODE IVP solution techniques, but again, OC problems are BVPs or Multi-Point Boundary Value Problems (MPBVP). These ODE solution techniques still apply when implemented in a shooting algorithm that enables terminal boundary conditions to be satisfied. Some of the most common solution methods for IVPs are summarized here followed by a brief description of shooting methods.

There are several first-order approximations available to solve the IVP. The Forward Euler (FE), Backward Euler (BE), Leap Frog (LF), and Trapezoidal Method (TM) given in [12, 67] are examples where difference formulas are used to discretize the time derivative on the left-hand-side of (2.26) resulting in time-marching rules given in Table 2.1. The *Order of Error* column indicates the magnitude of error in

Table 2.1. Simple numerical schemes for time-marching [12, 67]

Method	Equation	Order of Error
FE	$x_{i+1} = x_i + hf(x_i, t_i)$	$\mathcal{O}(h^2)$
BE	$x_{i+1} = x_i + hf(x_{i+1}, t_{i+1})$	$\mathcal{O}(h^2)$
LF	$x_{i+1} = x_{i-1} + 2hf(x_i, t_i)$	$\mathcal{O}(h^3)$
TM	$x_{i+1} = x_i + \frac{h}{2}(f(x_{i+1}, t_{i+1}) + f(x_i, t_i))$	$\mathcal{O}(h^3)$

the estimate of x_{i+1} based on the truncated Taylor-series. For example, if h is 0.1, then for the FE and BE the estimate errors will be on the order of 10^{-2} while the LF and TM will have errors on the order of 10^{-3}. Also note that the BE and TM methods are *implicit* methods because the functions on the right-hand-sides of the respective schemes are evaluated at the point being solved for on the left-hand-side. An explicit scheme is necessary to estimate the solution x_{i+1} which is then used in the right-hand-side of an implicit scheme. The implicit calculation of x_{i+1} then refines the solution generated by the explicit equation [9,67]. These methods are reasonably simple to implement, but can be computationally expensive since the step size must be very small to accurately approximate derivatives using piecewise linear functions. These methods also have stability limitations that can affect the solution if the step size is not properly chosen [67, 71]. An improved second-order Euler method (error $\mathcal{O}(h^3)$) is presented in [12] as:

$$x_{i+1} = x_i + \frac{h}{2} f(x_i, t_i) + \frac{h}{2} f(x_i + h f(x_i, t_i), t_{i+1}) \tag{2.28}$$

While this implicit scheme may provide more accurate estimates, there are generally more effective time-marching algorithms that are preferred. The *Taylor-series method* can be used to generate a scheme of potentially high accuracy [53]. A fourth-order approximation (error $\mathcal{O}(h^5)$), assuming uniform time step, is achieved by writing out the Taylor-series for x_{i+1} as:

$$x_{i+1} = x_i + h x'(t_i) + \frac{h^2}{2!} x''(t_i) + \frac{h^3}{3!} x'''(t_i) + \frac{h^4}{4!} x^{(4)}(t_i) + \mathcal{O}(h^5) \tag{2.29}$$

To use this equation it is necessary that the partial derivatives with respect to time of $f(x,t)$ exist up to the highest order derivative in the Taylor-series expansion chosen, fourth derivative in this case. The difficulty with this method is that analyt-

ical determination of the derivatives may become very difficult depending on the composition of $f(x(t), t)$. Fortunately, symbolic software manipulation packages like Mathematica™and MATLAB®can help with the differentiation.

A more efficient method is the commonly used Runge-Kutta (R-K) scheme. The classical fourth-order (error $\mathcal{O}(h^5)$) R-K scheme is discussed in [9, 12, 19, 53, 65] and written as:

$$\begin{aligned} K_1 &= f(x_i, t_i) \\ K_2 &= f\left(x_i + \frac{h}{2}K_1, t_i + \frac{h}{2}\right) \\ K_3 &= f\left(x_i + \frac{h}{2}K_2, t_i + \frac{h}{2}\right) \\ K_4 &= f(x_i + hK_3, t_{i+1}) \\ x_{i+1} &= x_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \end{aligned} \tag{2.30}$$

Several modifications have been made to the classical R-K scheme to introduce an adaptive step size. Two notable schemes, *Runge-Kutta-Fehlberg* [53] and *Runge-Kutta-Dormand-Prince* algorithms, evaluate fourth- and fifth-order R-K estimates and use the difference to determine the estimate error, which is then used to determine the step size for the next iteration [19, 24]. The Dormand-Prince method is used in the MATLAB®*ode45* ODE solver.

The methods used to solve the IVP thus far, Euler methods, LF, TM, Taylor-series, and R-K, are *single-step* methods, meaning that only data from the previous mesh point is needed to approximate x_{i+1} [12]. The R-K methods are known as *multiple-stage* methods since they evaluate $f(x, t)$ at multiple places in $[t_i, t_{i+1}]$ to calculate x_{i+1} as a weighted sum [71]. *Multiple-step* algorithms are designed to be more efficient by using function evaluations from more than one previous time point. Two of the most common methods are the explicit Adams-Bashforth (A-B) and im-

plicit Adams-Moulton (A-M) schemes [9, 71]. Fifth-order A-B and A-M schemes are given respectively as:

$$\begin{aligned} x_{i+1} &= x_i + \frac{h}{720}[1901f(x_i, t_i) - 2774f(x_{i-1}, t_{i-1}) + \\ &\quad 2616f(x_{i-2}, t_{i-2}) - 1274f(x_{i-3}, t_{i-3}) + 251f(x_{i-4}, t_{i-4})] \end{aligned} \tag{2.31}$$

$$\begin{aligned} x_{i+1} &= x_i + \frac{h}{720}[251f(x_{i+1}, t_{i+1}) + 646f(x_i, t_i) + \\ &\quad -264f(x_{i-1}, t_{i-1}) + 106f(x_{i-2}, t_{i-2}) - 19f(x_{i-3}, t_{i-3})] \end{aligned} \tag{2.32}$$

The A-B and A-M schemes are often used together as a *predictor-corrector* scheme where the A-B (2.31) is used to predict the value of x_{i+1} which is then corrected by the A-M (2.32), improving the quality of the solution approximation [12, 53]. Since the A-B and A-M schemes require either three or four previous function evaluations, any of the single-step methods, R-K for example, can be used to start the process until sufficient data is available for the A-B/A-M predictor-corrector to take over [12].

Now that a sampling of the most common time-marching methods for solving IVPs has been presented, shooting methods can be described that allow the IVP solution schemes to be used to solve the BVP [49, 71]. Consider the following ODE to be solved on the interval $t \in [t_0, t_f]$ with given boundary conditions x_0 and x_f:

$$\begin{aligned} \frac{d^2}{dt^2}x(t) &= f(t, x(t), \frac{d}{dt}x(t)) \\ x(t_0) &= x_0 \\ x(t_f) &= x_f \end{aligned} \tag{2.33}$$

This ODE can be written as a first-order system of ODEs by changing variables.

Letting $x_1 = x$ and $\frac{d}{dt}x_1 = x_2$ leads to the following system of first-order ODEs that can be solved by any of the previously described time-marching techniques.

$$\begin{aligned} \frac{d}{dt}x_1(t) &= x_2(t) \\ \frac{d}{dt}x_2(t) &= f(t, x_1(t), x_2(t)) \end{aligned} \tag{2.34}$$

The *simple shooting* method converts the BVP in (2.33) to an IVP in (2.34) by setting the initial conditions as:

$$\begin{aligned} x_1(t_0) &= x(t_0) = x_0 \\ x_2(t_0) &= \frac{d}{dt}x(t_0) = u \end{aligned} \tag{2.35}$$

An error function is necessary to refine guesses of u. The error function, $E(u)$, can be defined as:

$$E(u) = x(t_f; u) - x_f \tag{2.36}$$

Equation (2.36) basically describes the shooting method as a root-finding problem where the objective is to find u such that $E(u)$ is driven to zero [71]. Thus, the procedure is as follows:

1. Set initial guess of u in (2.35)
2. Apply time-marching scheme of choice to integrate (2.34) from t_0 to t_f
3. Calculate $E(u)$ using (2.36): The solution has been found if the error is within a desired tolerance (ϵ), $|E(u)| \leq \epsilon$, else
4. Update guess of u using root-finding scheme. Newton's method, modified Newton's method, and secant method are some of the most common root-finding algorithms.

5. Repeat process until $|E(u)| \leq \epsilon$ condition is achieved.

The simple shooting method is attractive to use because of its simplicity. However, there are numerical shortcomings of this method, such as slow convergence [49] when the guess of u is far from the root of (2.36), assuming that a globally convergent Newton method is being used that is guaranteed to find the root [53], or non-convergence when applied to the OC problem due to "ill conditioning of the Hamiltonian dynamics" [71]. A *multiple-shooting method* can be used to alleviate these numerical issues.

The multiple shooting method is based on the problem formulation given in (2.34) - (2.36). The difference is that the time interval $[t_0, t_f]$ is broken into $M+1$ subintervals, where M is the number of interior mesh points, and shooting is applied over each $[t_i, t_{i+1}]_j$, $j = 1, ..., M+1$, subinterval. The values of x_1 and x_2 are unknowns in each subinterval to be determined through time-marching and root-finding. Continuity constraints are placed on adjoining j and $j+1$ subinterval boundaries by:

$$\begin{aligned} x_{1,i+1}^{j} - x_{1,i}^{j+1} &= 0 \\ x_{2,i+1}^{j} - x_{2,i}^{j+1} &= 0 \end{aligned} \tag{2.37}$$

Therefore, the root-finding problem is to find the u_{i+1} slopes that satisfy the continuity conditions and the given boundary conditions [49, 53, 71].

The advantage of the multiple shooting method is that it is very flexible in the types of constraints considered in OC problems, such as bounded states, controls, and path constraints [78]. Very accurate results can be achieved while alleviating the aforementioned numerical issues, but it is cumbersome to implement, considerable analytical work may be required in deriving the costate equations, a good initial guess of the solution is necessary, and may introduce discontinuities in a continuous

problem that can not be resolved [49, 78]. Holsapple, Venkataraman, and Doman developed a modified simple shooting method in [49] that combines the "favorable aspects" of both the simple and multiple shooting methods resulting in a "superior, faster method for solving TPBVPs." This research shows that improving shooting methods is still an active research topic in the OC community.

The shooting methods described in this section can be applied to the indirect CV-based OC problem formulated by (2.21) - (2.23), according to [71]. Guesses are made for the unknown initial conditions, time marching applied to solve the system of ODEs, and error evaluation performed to check satisfaction of boundary conditions at the end of the time interval. Refinements to guesses of the unknown conditions are made through application of root-finding techniques and iterations made until the solution is found that satisfies terminal constraints. The procedure is essentially the same for both simple and multiple shooting methods with the exception that multiple-shooting applies the algorithm over a specified number of subintervals introducing more variables to be solved.

2.3.1.2 Indirect Collocation.

Collocation methods have become quite popular in the OC community over the past couple of decades. They are very powerful techniques for solving the HBVP given in (2.21) - (2.23). These methods are also very powerful in solving the OC problem using *direct methods*, discussed in section 2.3.2. Collocation techniques form the basis for the PSM, discussed in section 2.3.3, that will be used in this research. Rao presents a brief description of collocation methods in [71] with the general idea being that the unknown solution is approximated using a polynomial, or set of polynomials. Unknown coefficients in the polynomials are simultaneously solved for using root-finding techniques such that estimates of the states and derivatives of the states

match the analytic function at a set of discrete points.

Again, consider the general BVP of the form:

$$\begin{aligned} \frac{d}{dt}x(t) &= f(x(t),t) \\ x(t_0) &= x_0 \\ x(t_f) &= x_f \end{aligned} \tag{2.38}$$

This ODE can be a single function or a system of functions, which is generally consistent with the EL state and costate equations. Also suppose that N discrete time points are chosen in $[t_0, t_f]$, called collocation points and denoted as τ_i for $i = 1, ...N$, resulting in $N+1$ subintervals. The points can be chosen in various ways, and again, the distance between two mesh points τ_i and τ_{i+1} for $i = 0, ..., N-1$ need not be uniform. In fact, the mesh points used in the PSM are not uniformly spaced. Additionally, depending on the method used to choose the collocation points, t_0 and t_f may or may not be collocation points. For now, assume collocation points have been specified. Options for choosing points will be discussed later in this section.

One collocation method is to use a piecewise polynomial, Q_K, of degree K, to approximate the solution $x(t)$ in each subinterval $[t_i, t_{i+1}]$ as described in [71], which is essentially a Finite Element Method (FEM) [14]. The form of the solution is:

$$x(t) \approx Q_K(t) = \sum_{k=0}^{K} a_k (t - t_i)^k, \quad t \in [t_i, t_{i+1}] = I_i \tag{2.39}$$

The coefficients $(a_0, ..., a_K)$ are unknowns to be solved for such that the approximation $Q(t)$ matches $x(t)$ at the collocation points, i.e. $Q(\tau_i) = x(\tau_i)$, in the interval I_i. Furthermore, the derivative, $\dot{Q}(t)$, found by differentiating (2.39), is set to match $f(x(t), t)$ at the collocation points, known as the *collocation condition*. In order to find the unknown coefficients, a defect function, R, also referred to in the literature

as the remainder, residual, or error function, is defined as:

$$R_j = \dot{Q}(\tau_j) - f(x(\tau_j), \tau_j), \quad \text{for } j = 1, ..., N \tag{2.40}$$

The defects, which are functions of the unknown coefficients, can be put into a matrix, Z,

$$\mathbf{Z} = \begin{bmatrix} R_1 \\ \vdots \\ R_N \end{bmatrix} \tag{2.41}$$

and the coefficients can be found simultaneously by applying a root-finding scheme to solve $Z = \mathbf{0}$.

Another, more effective, collocation technique involves using global orthogonal polynomial basis functions over the entire time interval. Instead of $Q_K(t)$ defined by (2.39), a polynomial approximation to the unknown solution $x(t)$, as given in [33], can be written as:

$$x(t) \approx Q_K(t) = \sum_{k=0}^{N} a_k \phi_k(t) \tag{2.42}$$

where ϕ_k is an orthogonal polynomial basis function. In the context of the OC solution using the indirect CV-based formulation, the state (2.21), costate (2.22), and control (2.23) (if not able to be isolated and substituted into the state and costate equations) equations can be represented in this manner. The collocation points are chosen to be the roots of an orthogonal polynomial of a given order [71]. There are a variety of orthogonal polynomial basis functions that can be used such as Legendre, Chebyshev, and Hermite polynomials [29, 68, 71]. Lagrange interpolating polynomials can also be used instead of the orthogonal polynomials [14]. The most common collocation methods used to solve OC problems are based on Lagrange interpolating polynomial

bases, L_i. Letting ϕ_k in (2.42) be a Lagrange interpolating polynomial gives:

$$\phi_k(\tau) \triangleq L_k(\tau) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j} \qquad (k = 0, ..., N) \tag{2.43}$$

The advantage of using Lagrange polynomials in (2.42) is that they have the *isolation property*, meaning:

$$L_k(\tau_j) = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{if } k \neq j \end{cases} \tag{2.44}$$

which leads to coefficients, a_k, in (2.42) being the values of the state at the collocation points, i.e. $a_k = x(\tau_k)$ [71].

The most common collocation point sets are Legendre-Gauss (LG), Legendre-Gauss-Radau (LGR), and Legendre-Gauss-Lobatto (LGL) points [28,33,37,71]. Chebyshev-Gauss (CG), Chebyshev-Gauss-Radau (CGR), and Chebyshev-Gauss-Lobatto (CGL) points can also be used [33]. These point sets are based on Gaussian quadrature rules, are the roots of a particular polynomial, and are defined on the open, semi-open, or closed interval of -1 to 1. Therefore, the time domain $[t_0, t_f]$ given in the problem must be linearly transformed into the -1 to 1 domain, and transformed back once a solution has been found. Fornberg gives a table in [33] with the details of these node sets, which is summarized in Table 2.2.

The table shows that the LG, LGR, and LGL nodal sets are the roots of Legendre polynomials (P) of the degree specified by the subscript and the CG, CGR, and CGL are the roots of Chebyshev polynomials (T) defined by the recursion relationships given. The last column of the table gives formulas for calculating the weights associated with each point set to be used in Gaussian quadrature (numerical integration). These weights, w_k, will be used in the direct collocation method in the PSM. Numerous MATLAB®routines have been written and are available for download

Table 2.2. Collocation points and Gaussian quadrature weights [33]

Point Set	Domain	Nodes (x_k) ($k = 0, ..., N$)	Weights (w_k) ($k = 0, ..., N$)
LG	$(-1, 1)$	Roots of P_{N+1}	$w_k = \frac{2}{(1-x_k^2)[P'_{N+1}(x_k)]^2}$
LGR	$[-1, 1)$ or $(-1, 1]$	Roots of $P_N + P_{N+1}$	$w_0 = \frac{2}{(N+1)^2}$ $w_k = \frac{1}{(N+1)^2}\frac{1-x_k}{[P_N(x_k)]^2}$
LGL	$[-1, 1]$	-1, Roots of P'_N, 1	$w_k = \frac{2}{N(N+1)}\frac{2}{[P_N(x_k)]^2}$
CG	$(-1, 1)$	$\cos\left[\frac{(2k+1)\pi}{2N+2}\right]$	$w_k = \frac{\pi}{N+1}$
CGR	$[-1, 1)$ or $(-1, 1]$	$\cos\left[\frac{2\pi k}{2N+1}\right]$	$w_0 = \frac{\pi}{2N+1}$ $w_k = \frac{2\pi}{2N+2}$
CGL	$[-1, 1]$	$\cos\left[\frac{\pi k}{N}\right]$	$w_0 = w_N = \frac{pi}{2N}$ $w_k = \frac{\pi}{N}$

from the MathWorks™ website to automatically calculate the collocation points and quadrature weights given the number of desired points.

Solving the BVP in (2.38) using orthogonal collocation is the same as previously stated. The coefficients $(a_0, ..., a_K)$ are unknowns, the polynomial approximation $Q(t)$ and its derivative are set to match $x(t)$ and $f(x(t), t)$ respectively at the collocation points. The defect matrix defined by (2.40) and (2.41) is constructed to simultaneously solve for the unknown coefficients using root-finding to solve $\mathbf{Z} = \mathbf{0}$.

Solving the HBVP given in (2.21) - (2.23) is a natural extension of the process described here. Each of the states, costates, and controls making up the state vector $\mathbf{x}(t)$, costate vector $\boldsymbol{\lambda}(t)$, and control vector $\mathbf{u}(t)$ are approximated using a series of Lagrange polynomials as given in (2.42) and (2.43), the coefficients of which are the

unknowns to be found. A set of collocation nodes is chosen from Table 2.2, the defect matrix is constructed, and nonlinear root-finding method is applied as stated above. Fahroo and Ross demonstrated this method by using Legendre polynomial approximations for $\mathbf{x}$, $\mathbf{u}$, and $\boldsymbol{\lambda}$, collocated at LGL points to solve a sample problem in [27], concluding that the global orthogonal collocation method is an effective alternative to multiple-shooting techniques to solve the HBVP.

2.3.2 Direct Methods.

Indirect methods transform the continuous-time OC problem, stated by equations (2.7) - (2.11) introduced in section 2.2.1, into the Hamiltonian system characterized by (2.15). Solving the ODEs, analytically if possible, but most often numerically, associated with minimizing $\mathcal{H}$ in (2.21) - (2.23), applying the appropriate boundary conditions, and applying the PMP (2.25) indirectly minimizes the original cost functional (2.7). Direct methods, on the other hand, apply an appropriate discretization to the OC problem directly transforming it from an infinite-dimensional continuous-time problem to a finite-dimensional discrete-time problem which then can be solved using nonlinear programming by iteratively searching for the state and control solutions that meet optimality conditions while satisfying constraints and evaluating the cost function to verify that it is a minimum [71]. Direct methods are attractive since it is not necessary to explicitly derive the first-order optimality conditions or to determine if the inequality constraints are active. The convergence of direct methods is less sensitive to the initial guess and do not require a guess for the costates [7]. However, direct methods are less accurate than indirect methods and there may actually be several minimizing solutions making it possible for the result to be incorrect [78]. Since direct methods do not provide costate information, there can be some uncertainty as to whether the NLP optimal solution is truly the optimal solution [50]. This

subsection briefly summarizes direct methods commonly used to solve OC problems.

2.3.2.1 Nonlinear Programming.

The NLP problem is generally stated, as presented in [61] and [71], as:

$$\begin{aligned} &\textit{minimize} && f(\mathbf{z}) \\ &\textit{subject to} && \mathbf{h}(\mathbf{z}) = \mathbf{0} \\ & && \mathbf{g}(\mathbf{z}) \leq \mathbf{0} \end{aligned} \tag{2.45}$$

where $\mathbf{z} \in \mathbb{R}^n$ is the vector of design variables, $f(\mathbf{z})$ is the objective functional, and $\mathbf{h}(\mathbf{z})$ and $\mathbf{g}(\mathbf{z})$ are the constraints. If there are m equality constraints and p inequality constraints, then $\mathbf{h}(\mathbf{z}) \in \mathbb{R}^m$ and $\mathbf{g}(\mathbf{z}) \in \mathbb{R}^p$. The first-order necessary conditions for an optimal solution, $\mathbf{z}^*$, are given by the following KKT conditions:

$$\begin{aligned} \nabla f(\mathbf{z}^*) + \boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{z}^*) + \boldsymbol{\mu}^T \nabla \mathbf{g}(\mathbf{z}^*) &= \mathbf{0} \\ h_i(\mathbf{z}^*) &= 0 \quad (i = 1, ..., m) \\ g_i(\mathbf{z}^*) &\leq 0 \quad (i = 1, ..., p) \\ \mu_i &\geq 0 \quad (i = 1, ..., p) \\ \mu_i g_i(\mathbf{z}^*) &= 0 \quad (i = 1, ..., p) \end{aligned} \tag{2.46}$$

where ∇ indicates the gradient of the argument function and $\boldsymbol{\lambda} \in \mathbb{R}^m$ and $\boldsymbol{\mu} \in \mathbb{R}^p$ are Lagrange multipliers associated with the equality and inequality constraints respectively. The second-order sufficient conditions for optimality are obtained from the Hessian (matrix of second-order partial derivatives) of the Lagrangian functional.

The Lagrangian, l, is given by:

$$l(\mathbf{z}) = f(\mathbf{z}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{z}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{z}) \tag{2.47}$$

The Hessian of the Lagrangian, evaluated at the candidate optimal solution, is then:

$$\mathbf{L}(\mathbf{z}^*) = \mathbf{F}(\mathbf{z}^*) + \boldsymbol{\lambda}^T \mathbf{H}(\mathbf{z}^*) + \boldsymbol{\mu}^T \mathbf{G}(\mathbf{z}^*) \tag{2.48}$$

where $\mathbf{L}(\cdot)$, $\mathbf{F}(\cdot)$, $\mathbf{H}(\cdot)$, and $\mathbf{G}(\cdot)$ denote the Hessian matrices of $l(\cdot)$, $f(\cdot)$, $\mathbf{h}(\cdot)$, and $\mathbf{g}(\cdot)$ respectively, following the notation in [61]. The second-order necessary condition is that in addition to the conditions in (2.46) being satisfied, the Hessian of the Lagrangian function (2.48) be positive-definite at the candidate $\mathbf{z}^*$.

An NLP method is an iterative search algorithm. It searches through the space bounded by the constraints in (2.45) to find candidate minimum points, $\mathbf{z}^*$, that satisfy the constraints with the lowest cost. The optimality conditions in equations (2.46) and (2.48) are checked at the candidate points to determine whether each is a minimum, maximum, or saddle point. The key elements in the search algorithm are determining the direction in the space and the size of the step that should be taken in that direction to find successive candidate points [71]. Care must be taken in selecting and implementing the appropriate search algorithm in order to find a minimum solution and to have some kind of guarantee that the algorithm is globally convergent. That is to say that the point that is found when the algorithm terminates is the global minimum or at least an acceptable local minimum in a small vicinity of the global solution [61]. Rao discusses the most common gradient search algorithms in [71] along with some heuristic ones that use some added randomness to direct the search. Luenberger and Ye present several algorithms in [61] that are effective in solving nonlinear constrained optimization problems. They also present proofs that

specify conditions for guaranteeing local and global minimum solutions.

Gradient methods are effective search algorithms using the gradient vector and Hessian matrix to select a direction of descent such that the objective function $f(\mathbf{x}_{k+1})$ sufficiently decreases [61]. One of the most common classes of descent methods uses the following to select subsequent points in the iterative process:

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \alpha_k \mathbf{d}_k \tag{2.49}$$

The step size, α_k, is calculated to minimize f and the search direction, $\mathbf{d}_k$, is given by:

$$\mathbf{d}_k = [\mathbf{F}(\mathbf{z}_k)]^{-1} \nabla f(\mathbf{z}_k)^T \tag{2.50}$$

which is a modified Newton method [61]. The steepest descent method is obtained if the inverse of the Hessian matrix, $[\mathbf{F}(\mathbf{z}_k)]^{-1}$, is replaced with the identity matrix [61]. This process depends on the second partial-derivatives of f existing and inversion of the resulting Hessian matrix at each iteration. Further modifications to the Newton method exist to approximate the inverse of the Hessian to avoid costly computation of the inverse at each iteration [61].

Shooting, multiple shooting, and collocation methods described in section 2.3.1 can be applied to solve the direct OC problem using NLP techniques. The concepts discussed for indirect methods are similar in direct methods, with some modifications. Rao presents algorithms for shooting and multiple shooting in [71] and Betts discusses several shooting and time-marching methods in the context of NLP formulation in [9]. Becerra's paper [6] discusses translating a continuous-time OC problem into an NLP problem and solves an example problem of controlling the movement of a pendulum-like system described by ODEs derived using Newton's Second Law. The main idea of all these schemes, as stated in these references, is that they provide

some way of choosing N discrete points the time domain, denoted as τ_i $(i = 1, ..., N)$, approximating the cost function by replacing the integral with a finite summation, and discretizing the dynamics equations. Conceptually, the discretization process leads to:

$$\begin{aligned} J(\mathbf{x}(t), \mathbf{u}(t), t) &\implies J(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i) \\ \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) &\implies \mathbf{x}(\tau_{i+1}) = \mathbf{f}(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i) \end{aligned}$$

After the discretization scheme has been applied, the NLP definition (2.45) looks like:

$$\begin{aligned} &\textit{minimize} && J(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i) \\ &\textit{subject to} && \mathbf{x}(\tau_{i+1}) - \mathbf{f}(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i) = \mathbf{0} \\ & && \mathbf{u}_{min} - \mathbf{u}_i \leq \mathbf{0},\ \mathbf{u}_i - \mathbf{u}_{max} \leq \mathbf{0} \\ & && \mathbf{x}_{min} - \mathbf{x}_i \leq \mathbf{0},\ \mathbf{x}_i - \mathbf{x}_{max} \leq \mathbf{0} \\ & && \mathbf{x}(t_0) = \mathbf{x}_0 \\ & && \mathbf{x}(t_f) = \mathbf{x}_f \\ &\textit{for} && i = 1, ..., M \end{aligned} \tag{2.51}$$

Other constraints, like path constraints, can be added to (2.51) as dictated by the problem. In contrast to the CV-based indirect methods, bounds on the states can be directly included in the NLP formulation as inequality constraints.

2.3.2.2 Direct Shooting and Collocation.

To summarize the shooting and collocation methods using the NLP formulation, it can be stated that these methods are simply ways of representing the integral cost function as a discrete sum and using discrete expressions for the solution of

the dynamic constraints. Note that the step size between two successive points is $h_i = t_{i+1} - t_i$, where $i = 1, ..., N$. The index on the step size can be dropped if the step sizes are equal. A quadrature rule for evaluating the cost functional (2.7) is necessary and should be selected to be consistent with the method used to solve the differential equation (2.8). For example, if an R-K scheme is used to solve the differential dynamics equations, then an R-K scheme of the same order should be used to evaluate the cost functional. An effective way to accomplish this is to augment the state vector, recalling that the size of the state vector is $n \times 1$, with:

$$\dot{x}_{n+1} = g(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.52}$$

resulting in the new set of differential equations:

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dots \\ \dot{x}_{n+1}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ \dots \\ g(\mathbf{x}(t), \mathbf{u}(t), t) \end{bmatrix} \tag{2.53}$$

The cost function can then be evaluated after the time-marching scheme has been applied using:

$$J = h(\mathbf{x}(t_f), t) + x_{n+1}(t_f) \tag{2.54}$$

In both the shooting and multiple shooting methods, the control is approximated using a parameterization of the form:

$$\mathbf{u}(t) \approx \sum_{i=1}^{N} \mathbf{a}_i \psi_i(t) \tag{2.55}$$

The coefficients, $\mathbf{a}_i$, are determined using NLP and $\psi_i(t)$ are basis functions of choice. The shooting method presented in [71] and [9] can be summarized as an NLP problem

with the following steps:

1. Make initial guess of unknown coefficients, $\mathbf{a}_i$.

2. Integrate state equation (2.8) over the interval $[t_0, t_f]$ using the parameterized control by applying a time-marching method discussed in section 2.3.1.1.

3. Evaluate the error between the solution and the specified boundary conditions.

4. Evaluate the cost functional using numerical integration.

5. If the cost is not at a minimum value and/or the boundary conditions have not been met, then update the values of $\mathbf{a}_i$ using a gradient method and repeat steps 2 through 5.

Multiple shooting uses the same procedure but applies the shooting method over each of the $N+1$ subintervals in $[t_0, t_f]$ and ensures that continuity is enforced at the interfaces between subintervals. In essence, the shooting method treats the OC problem over the interval as a single problem, whereas multiple shooting treats the OC problem as $N+1$ subproblems connected together at the subinterval interfaces.

Direct collocation is again similar to collocation methods described for indirect methods. The difference between indirect and direct collocation is that the differential equations for the costates do not need to be derived or evaluated and a quadrature scheme is used to approximate the integral in the cost functional (2.7), similar to using the procedure given by equations (2.52) - (2.54). Suitable discrete approximations of the states are made using some type of polynomial basis functions, which can be piecewise or global, similar to equations (2.39) and (2.42). Additionally, the direct method approximates the control functions in the same way that the states are approximated. These equations are then cast in the NLP formulation in (2.45) and solved for the states and controls at each τ_i that satisfy the KKT conditions, are

within the boundaries specified by inequality constraints on the states and controls, and result in the minimum cost value.

At this point, it's more valuable to focus on direct global orthogonal collocation since it is a key piece of the PSM described in the next section. Only the key equations are summarized here in order to avoid excessive repetition in the next section where further discussion will be essential to describing the PSM. These equations are presented in a general sense noting that there will be variations depending on the sets of collocation points and orthogonal polynomial bases chosen. The first step is to select the set of N collocation points, τ_i $(i = 1, ..., N)$, and calculate the associated quadrature weights, w_i, using possible basis sets in Table 2.2, recalling that the time domain $t \in [t_0, t_f]$ needs to be linearly mapped to $\tau \in [-1, 1]$. Next, a Gaussian quadrature rule is used to approximate the integral in the cost functional. Re-writing (2.7) to replace the integral with Gaussian quadrature gives:

$$J(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) = \Phi(\mathbf{x}(1), t_f) + \frac{t_f - t_0}{2} \sum_{i=1}^{N} g(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i) \tag{2.56}$$

The state and control functions can be approximated using a class of functions known as cardinal functions[1], C_i, given by:

$$\begin{aligned} \mathbf{x}(\tau) &\approx \sum_{i=1}^{N} \mathbf{x}(\tau_i) C_i(\tau) \\ \mathbf{u}(\tau) &\approx \sum_{i=1}^{N} \mathbf{u}(\tau_i) C_i(\tau) \end{aligned} \tag{2.57}$$

[1]The term *cardinal functions* was first used by Sir Edmund Whittaker in work published in 1915. It refers to orthogonal polynomial or trigonometric basis functions used to approximate functions using polynomial interpolation. These functions have the property that the i^{th} cardinal function is equal to one at the i^{th} collocation node and zero elsewhere. Cardinal functions are also known as the *Lagrange basis* and the *Fundamental polynomial of Lagrangian interpolation* [13, 14]

The cardinal functions can be constructed using Chebyshev, Legendre, Laguerre, or Hermite orthogonal polynomial basis functions [14], and have the general form:

$$C_i(\tau) = \frac{\phi_{N+1}(\tau)}{\dot{\phi}_{N+1}(\tau_i)(\tau - \tau_i)} \tag{2.58}$$

with the property:

$$C_i(\tau_j) = \delta_{i,j} \tag{2.59}$$

where $\delta_{i,j}$ is the Kronecker delta function. Lagrange interpolating polynomial bases are commonly used as the cardinal functions when applying direct collocation methods to solve OC problems, which have the form:

$$L_i(\tau) = \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2.60}$$

Using (2.57) with Lagrange interpolating polynomial bases (2.60), the dynamics in (2.8) can be approximated as:

$$\dot{\mathbf{x}}(\tau) \approx \sum_{i=1}^{N} \mathbf{x}(\tau_i)\dot{\mathbf{L}}_i(\tau) \tag{2.61}$$

Again, the right-hand-sides of (2.8) and (2.61) are set to match at the collocation points, resulting in the following equality constraint for the NLP:

$$\sum_{i=1}^{N} \mathbf{x}(\tau_i)\dot{\mathbf{L}}_i(\tau) - \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) = \mathbf{0} \tag{2.62}$$

These are the key elements necessary to transcribe the OC problem into an NLP problem using global orthogonal collocation. The NLP can be constructed to solve for the states and controls that satisfy first- and second-order optimality conditions, are within the boundaries specified by the inequality constraints, and result in the

minimum cost functional value.

2.3.3 Pseudospectral Methods.

The discussion of numerical techniques for solving OC problems has been purposefully presented in a way that builds up to the PSM presented in this subsection. The PSM combines many of the previously discussed numerical tools into a very powerful numerical method, and has been generally presented as *global orthogonal collocation* in sections 2.3.1.2 and 2.3.2.2, but with details purposely left out until this section. Indirect methods use global orthogonal collocation as a discrete approximation of the HBVP first-order optimality conditions and numerically solve for $\mathbf{x}(t)$, $\mathbf{u}(t)$, and $\boldsymbol{\lambda}(t)$ that minimize $\mathcal{H}(\cdot)$, thus indirectly minimizing $J(\cdot)$, using root finding schemes. Direct methods use global orthogonal collocation to approximate $\mathbf{x}(t)$ and $\mathbf{u}(t)$, and Gaussian quadrature to approximate the integral cost functional, $J(\mathbf{x}(t), \mathbf{u}(t), t)$. The discretized continuous-time OC problem can then be solved using NLP techniques. This section begins with a brief description of spectral methods, discusses terminology to distinguish between *spectral* and *pseudospectral* methods, followed by a formulation of the direct Gauss Pseudospectral Method and software that is used in this research.

2.3.3.1 General Discussion of Spectral Methods.

References [14] and [33] provides excellent development and application of spectral methods to solve differential equations. Spectral methods are fundamentally a family of methods used to solve differential equations of the form:

$$Lu(x) = f(x) \tag{2.63}$$

where L is a general linear operator that can represent either differential or integral operators. To solve the equation, the unknown solution is represented as a series of

$N+1$ basis functions $\phi_n(x)$ as:

$$u(x) \approx u_N(x) = \sum_{n=0}^{N} a_n \phi_n(x) \tag{2.64}$$

and substituted into (2.63) to give:

$$Lu_N(x) \approx f(x) \tag{2.65}$$

The goal then is to choose the expansion coefficients, a_n, that minimizes the residual function:

$$Lu_N(x) - f(x) = R(x; a_0, a_1, ..., a_n) \tag{2.66}$$

The beauty of spectral methods is that L can be expressed as either a differentiation or integration matrix translating the differential equation to a matrix equation where matrix solvers can then be used to determine the expansion coefficients. With nonlinear problems, equation (2.65) generates a system of nonlinear algebraic equations that necessitates iterative solution techniques such as Newton's method or a variant thereof.

The difference between *spectral* methods lies in how the residual function is minimized. The term *spectral methods* is a broad term to include two main categories of methods: *interpolating* and *non-interpolating*.

Interpolating methods are known as *pseudospectral* methods. These methods associate a set of grid points with a selected basis function, as described by Table 2.2, and find the expansion coefficients such that $R(x; a_0, a_1, ..., a_n) = 0$. In other words, the function and the polynomial representation of the function exactly match at the grid points. Thus, $u_N(x)$ converges to $u(x)$ as N increases and, because the points are chosen according to the aforementioned table, the convergence rate is exponen-

tial, also called spectral convergence. Furthermore, the error of the pseudospectral approximation is $\approx \mathcal{O}[\frac{1}{N}]^N$ [14]. Other names used for the pseudospectral method are *orthogonal collocation*, which has been used in this document, and *method of selected points*. *Pseudospectral* is also used to imply that global basis functions are being used and that the function values are unknowns, as opposed to interpolation where the function values are known and finding the expansion coefficients can be easily solved for in order to approximate the function with an interpolating polynomial.

The keys to pseudospectral methods lie in selecting appropriate basis functions and collocation points. Boyd presents theorems and proofs showing convergence of several polynomial bases and that orthogonal polynomial basis functions with collocation points chosen to be Gaussian quadrature points, i.e. the roots of the N^{th}, $(N+1)^{th}$, or combination of the two as given in Table 2.2, provides the best possible result for both integral and derivative forms of (2.63). After presenting the proofs, Boyd states:

> ...since quadrature formulas are obtained by analytical integration of the interpolating polynomial, it follows that the best choice of quadrature points is also the best choice of interpolation points and vice-versa.

and that as a rule of thumb:

> The grid points should be the abscissas of a Gaussian quadrature associated with the basis set.

This is the reason why collocation methods presented earlier are chosen to solve OC problems.

Solving the pseudospectral problem involves either inverting the L matrix and performing the matrix multiplication $L^{-1}f$ or using a Newton solver in the nonlinear case. The attractive feature of this method is that only evaluations of f at the

collocation points are necessary to solve the problem. In contrast, as will be shown below, the Galerkin spectral method requires integration of the function.

Non-interpolating methods are often used as reference to *Galerkin* methods. The literature can, however, be confusing on terminology. Some references use the term *spectral method* as a synonym for *Galerkin method*, while others, like [14], use the convention adopted in this document where the term *spectral methods* is an umbrella that covers both pseudospectral and Galerkin methods as well as some others not presented here. The non-interpolating, i.e. Galerkin method, does not use a collocation grid, but determines the expansion coefficients by integration resulting from inner products with test functions. The general formulation and solution of a differential equation using the Galerkin method is briefly shown here. Only basic details are given here for comparison with the pseudospectral method and as motivation for using pseudospectral rather than Galerkin methods. Refer to [14] for more details on the Galerkin method.

Assume that L in equation (2.63) and (2.65) is a linear operator, either differential or integral. The Galerkin method solves the differential equation by introducing a *test function* or *weight function* ϕ_i and using it to take the inner products of both sides of (2.63) orthogonally projecting the approximation error onto the space spanned by the polynomial bases resulting in minimum error. Denoting the new linear operator H and function g after the inner product gives the following matrix equation:

$$\mathbf{Ha} = \mathbf{g} \tag{2.67}$$

where the elements of $\mathbf{H}$ and $\mathbf{g}$ are given by the inner products:

$$\begin{aligned} H_{ij} &= (\phi_i, L\phi_j) \qquad \text{for } i,j = 1, ..., N+1 \\ g_i &= (\phi_i, f) \end{aligned} \tag{2.68}$$

The inner products in (2.68) require integration operations to fill each element in $\mathbf{H}$ and $\mathbf{f}$, which makes coding the Galerkin method more complicated than the pseudospectral method. Note that even though the Galerkin formulation does not directly use collocation points, they do become necessary to apply Gaussian quadrature, as described in section 2.3.2.1, to numerically approximate the inner product integrals. Therefore, the Galerkin and pseudospectral methods are equivalent if the Galerkin quadrature points are the same as the pseudospectral collocation points and the accuracy of the pseudospectral is comparable to that of the Galerkin method. Boyd states that a Galerkin method using N expansion terms is equal in accuracy to the pseudospectral method using $N + 1$ or $N + 2$ terms.

Now all the pieces are in place to discuss the Gauss Pseudospectral Method applied to solve OC problems.

2.3.3.2 Gauss Pseudospectral Method for Optimal Control.

When it comes to solving differential equations, either Partial Differential Equation (PDE)s or ODEs, there are many PSMs available. The differences between methods are determined by the orthogonal polynomial basis functions chosen for spectral representation of the solution and the collocation points used. Reference [14] primarily discusses Fourier and Chebychev methods, [33] presents other methods, and [60] presents a Chebyshev method for solving BVPs. For example, Legendre polynomial basis functions can be used with Legendre-Gauss collocation points, Chebyshev polynomials with Chebyshev points, and so on. Lagrange interpolating polynomial basis functions can always be used in place of the other basis sets [14]. The HBVP can be solved using any of these combinations of basis functions and collocation points, such as in [27] where Lagrange polynomial bases combined with Chebyshev collocation points were used. However, since the first-order optimality conditions of the OC

problem given by (2.21) - (2.23), the associated boundary conditions, and switching conditions of the inequality constraints (2.10) may be extremely difficult to derive, it's attractive to use the direct PSM to solve OC problems, however the additional requirement of minimizing the cost functional makes direct OC solution methods more difficult than simply solving a set of differential equations [50].

The most common PSMs used to directly solve nonlinear OC problems have the same general procedure that parameterizes the states, $\mathbf{x}(t) \in \mathbb{R}^n$, and controls, $\mathbf{u}(t) \in \mathbb{R}^m$, using Lagrange interpolating polynomials, discretizes the constraints (2.8) - (2.10) and the general Bolza cost functional (2.7) using a quadrature rule, converting the differential dynamic constraints to algebraic constraints, and applies an NLP solver to find the solution to the resulting parameter optimization problem [7, 8, 28, 29, 50]. Some of the most common direct PSMs are the Legendre Pseudospectral Method (LPM) [26, 28], Chebyshev Pseudospectral Method (CPM) [29], GPM, and Radau Pseudospectral Method (RPM) [7, 36, 50]. Several software packages, a sampling of which are listed in [50], have been developed to directly solve the continuous-time OC problem by discretizing and transcribing it into an NLP problem using this general procedure. Two common packages, both written for use in MATLAB®and using SNOPT [42] as the NLP solver, are DIDO [1], which implements the Legendre pseudospectral method developed in [28], and GPOPS [72], which implements the GPM developed in [8], [50], and [7].

Selecting which PSM to use to solve the direct formulation of the OC problem can be confusing. It's known that solving the HBVP provides guarantees on meeting the optimality conditions [8, 29]. Significant research efforts have focused on determining which method to use for specific problem sets, for example finite horizon, infinite horizon, etc. [30, 35]. Regardless of the method chosen, it is advantageous to select a direct PSM that has been shown to have some equivalence to the numerical solution of

the continuous CV-based HBVP. One such method is the GPM, which Benson derived in [7]. Benson showed how the continuous-time OC problem can be transcribed to a NLP problem using Lagrange interpolating polynomial bases of degree N to parameterize the states and controls at $N+1$ LG collocation points. Benson also proposed and provided proof of the following theorem:

> The Karush-Kuhn-Tucker (KKT) conditions of the NLP are exactly equivalent to the discretized form of the continuous first-order necessary conditions of the Bolza problem. Furthermore, a costate estimate can be found from the KKT multipliers, $\tilde{\mathbf{\Lambda}}_k$ [costates of the pseudospectral representation of the OC problem], and Lagrange multipliers, $\boldsymbol{\nu}$, that satisfy the pseudospectral approximation to the costate dynamics.

Benson was able to conclude that solving the OC problem using direct transcription to an NLP problem is "exactly equivalent to solving the discretized form of the continuous first-order necessary conditions" given by the HBVP. Reference [8] summarizes the GPM development, pseudospectral representation of the CV-derived HBVP, and proves equivalence of the KKT conditions of the NLP and the first-order continuous necessary conditions of the HBVP.

Huntington extended Benson's work in [50] by adding inequality path constraints to the GPM formulation and again showed equivalence between the KKT multipliers and the first-order optimality conditions of the HBVP. He also investigated the same equivalence of the LPM, based on LGL collocation points, and the RPM, based on LGR collocation points, and showed mapping relationships between the KKT multipliers and the costates of the HBVP, noting that the GPM had the most direct mapping relationships, particularly at the endpoints. Fahroo and Ross have also studied the equivalence of the LPM KKT optimality conditions and the optimality conditions of the HBVP, noting that the KKT multipliers "...satisfy a discrete analog of the costate dynamics" at the interior points giving a check that the solution meets

optimality conditions. Similar to Benson, Huntington concluded that, although all three methods provide a mapping between the NLP KKTs and the HBVP costates, "...solving the NLP derived from Gauss pseudospectral transcription is equivalent to applying Gauss pseudospectral discretization to the continuous-time first-order optimality conditions." The main take-away from Huntington's work is that while researchers are currently investigating convergence of the direct transcription to the HBVP, equivalence of the KKT multipliers with first-order optimality conditions gives confidence that an optimal solution has been found.

The key equations transforming the continuous OC problem to a parameter optimization problem used by GPOPS and solved with SNOPT as derived in [7, 50] and summarized in [8, 72] are paraphrased below since GPOPS is a key tool used extensively in the research presented in this dissertation. Some repetition of previous sections will be evident, but is necessary for continuity in describing how GPOPS discretizes the continuous OC problem and transcribes it to the NLP problem for SNOPT to solve. In discussing the GPOPS implementation, there will be some notation deviation from the general discussion presented in previous sections. Again, this is necessary for the sake of continuity in discussing the specific implementation.

The continuous Bolza OC problem can be stated as [8, 72]: Determine the state, $\mathbf{x}(\tau) \in \mathbb{R}^n$, and control, $\mathbf{u}(\tau) \in \mathbb{R}^m$, and the terminal time, t_f if unknown, to minimize the cost functional:

$$J = \Phi(\mathbf{x}(-1), t_0, \mathbf{x}(1), t_f) + \frac{t_f - t_0}{2} \int_{-1}^{1} g(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau; t_0, t_f) d\tau \tag{2.69}$$

subject to the differential (dynamic), boundary, and path constraints, respectively:

$$\frac{d\mathbf{x}}{d\tau} = \frac{t_f - t_0}{2} \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau; t_0, t_f) \tag{2.70}$$

$$\boldsymbol{\phi}(\mathbf{x}(-1), t_0, \mathbf{x}(1), t_f) = \mathbf{0} \quad \in \mathbb{R}^q \tag{2.71}$$

$$\mathbf{C}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau; t_0, t_f) \leq \mathbf{0} \quad \in \mathbb{R}^c \tag{2.72}$$

Note that equations (2.69) - (2.72) have been written in the time variable τ to be in the proper domain for the GPM. Linear transformations between $t \in [t_0, t_f]$ and $\tau \in [-1, 1]$ are accomplished by:

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{2.73}$$

The GPM process is to discretize the continuous equations (2.69) - (2.72) and transcribe them into an NLP problem. The process begins by choosing N LG collocation points $\in (-1, 1)$, noting that the endpoints are not included in the set. The states, $\mathbf{x}(\tau)$, and controls, $\mathbf{u}(\tau)$, are approximated using N or $N+1$ Lagrange interpolating polynomials of the form:

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=0}^{N} \mathbf{X}(\tau_i) L_i^x(\tau) \tag{2.74}$$

$$\mathbf{u}(\tau) \approx \mathbf{U}(\tau) = \sum_{i=1}^{N} \mathbf{U}(\tau_i) L_i^u(\tau) \tag{2.75}$$

where lower case symbols, $\mathbf{x}(\tau)$ and $\mathbf{u}(\tau)$, denote the continuous state and control vectors while the capitalized symbols, $\mathbf{X}(\tau)$ and $\mathbf{U}(\tau)$, represent the corresponding polynomial approximations. Additionally, superscripts x and u are used to distinguish between Lagrange polynomials representing the state and control approximations, respectively. The Lagrange polynomials have the form:

$$L_i^x(\tau) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j} \quad (i = 0, ..., N) \tag{2.76}$$

$$L_i^u(\tau) = \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j} \quad (i = 1, ..., N) \tag{2.77}$$

and satisfy the isolation property:

$$L_i^x(\tau_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (i = 0, ..., N)$$

$$L_i^u(\tau_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (i = 1, ..., N)$$

The differential dynamic constraints (2.70) are approximated by differentiating (2.74):

$$\dot{\mathbf{x}}(\tau) \approx \dot{\mathbf{X}}(\tau) = \sum_{i=0}^{N} x(\tau_i) \dot{L}_i(\tau) \tag{2.78}$$

The derivatives of the Lagrange polynomials evaluated at the LG collocation points can be written in a differentiation matrix, $\mathbf{D}$, which has dimension $N \times N + 1$, and whose elements are expressed as:

$$D_{ki} = \dot{L}_i(\tau_k) = \sum_{l=0}^{N} \frac{\prod_{\substack{j=0 \\ j \neq i,l}}^{N} (\tau_k - \tau_j)}{\prod_{\substack{j=0 \\ j \neq i}}^{N} (\tau_i - \tau_j)} \tag{2.79}$$

where the counting indices are $k = 1, ..., N$ and $i = 0, ..., N$. The dynamic constraints (2.70) are discretized and transformed into algebraic constraints suitable for the NLP solver using (2.74), (2.75), (2.78), and (2.79) as follows:

$$\sum_{i=0}^{N} D_{ki} \mathbf{X}_i - \frac{t_f - t_0}{2} \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) = \mathbf{0} \quad (k = 1, ..., N) \tag{2.80}$$

For clarity, shorthand notation definitions used in the GPM development are sum-

marized in the following list:

- $\mathbf{X}_k \equiv \mathbf{X}(\tau_k) \in \mathbb{R}^n$ for $k = 1, ..., N$.
- $\mathbf{U}_k \equiv \mathbf{U}(\tau_k) \in \mathbb{R}^m$ for $k = 1, ..., N$.
- $\mathbf{X}_0 \equiv \mathbf{X}(-1)$, where $\tau = -1$ is not a collocation point.
- $\mathbf{X}_f \equiv \mathbf{X}(1)$, where $\tau = 1$ is not a collocation point.
- $g_k \equiv g(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$
- $\mathbf{f}_k \equiv \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$
- $C_{jk} \equiv C_j(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$
- $\tilde{\mathcal{H}}_k \equiv \tilde{\mathcal{H}}(\mathbf{X}_k, \tilde{\mathbf{\Lambda}}_k, \tilde{\mathbf{\Lambda}}_F, \tilde{\boldsymbol{\mu}}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$, where the Hamiltonian, $\tilde{\mathcal{H}}_k$, and Lagrange multipliers, $\tilde{\mathbf{\Lambda}}_k$, $\tilde{\mathbf{\Lambda}}_F$, and $\tilde{\boldsymbol{\mu}}_k$ are defined later.

Since $\mathbf{X}(\tau)$ and $\mathbf{U}(\tau)$ are not collocated at the endpoints, it's necessary to approximate the terminal state, $\mathbf{X}_f$, using Gaussian quadrature:

$$\mathbf{X}_f = \mathbf{X}_0 + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \tag{2.81}$$

The NLP cost function will be a discretized form of (2.69), which is again generated using Gaussian quadrature to approximate the integral term. The discretized cost functional is:

$$J = \mathbf{\Phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k g(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \tag{2.82}$$

The last necessary items for the NLP transcription are discretized forms of the boundary constraints (2.71) and path constraints (2.72), stated as:

$$\boldsymbol{\phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) = \mathbf{0} \tag{2.83}$$

and:

$$\mathbf{C}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \leq \mathbf{0} \quad (k = 1, ..., N) \tag{2.84}$$

With the discretization given above, the NLP problem can be stated as: find $\mathbf{X}_k$ and $\mathbf{U}_k$ that minimizes (2.82) subject to (2.80), (2.81), (2.83), (2.84), and $\mathbf{X}(\tau_0) = \mathbf{X}_0$. GPOPS calls the SNOPT NLP solver to find the solution, which is an approximate solution to the continuous Bolza OC problem.

The NLP solver searches for the solution that satisfies the KKT conditions. The KKT conditions, similar to those introduced in section 2.3.2.1, are listed below, again paraphrasing [8]. They are found using the augmented cost functional, also known as the Lagrangian, which is analogous to (2.13), constructed by adjoining the discrete constraints with the discretized cost functional though introduction of Lagrange multipliers $\tilde{\boldsymbol{\nu}}$, $\tilde{\boldsymbol{\mu}}$, and $\tilde{\boldsymbol{\Lambda}}$. The augmented cost function is given by:

$$\begin{aligned} J_a = \quad & \boldsymbol{\Phi} + \tfrac{t_f - t_0}{2} \textstyle\sum_{k=1}^{N} w_k g - \tilde{\boldsymbol{\nu}}^T \boldsymbol{\phi} - \sum_{k=1}^{N} \tilde{\boldsymbol{\mu}}_k^T C \\ & - \textstyle\sum_{k=1}^{N} \tilde{\boldsymbol{\Lambda}}_k^T \left[\sum_{i=0}^{N} D_{ki} \mathbf{X}_i - \tfrac{t_f - t_0}{2} \mathbf{f} \right] \\ & - \tilde{\boldsymbol{\Lambda}}_F^T \left[\mathbf{X}_f - \mathbf{X}_0 - \tfrac{t_f - t_0}{2} \textstyle\sum_{k=1}^{N} w_k \mathbf{f} \right] \end{aligned} \tag{2.85}$$

The arguments of the functions have been dropped for readability. Analogous to the development of the first-order necessary conditions of the continuous OC problem developed in section 2.2.2, the KKT conditions are found by setting to zero the partial derivatives of J_a with respect to $\mathbf{X}_0$, $\mathbf{X}_k$, $\mathbf{X}_f$, $\mathbf{U}_k$, $\tilde{\boldsymbol{\Lambda}}_k$, $\tilde{\boldsymbol{\Lambda}}_f$, $\tilde{\boldsymbol{\mu}}_k$, $\tilde{\boldsymbol{\nu}}$, t_0, and t_f. Defining the discretized Hamiltonian, $\tilde{\mathcal{H}}_k$, as:

$$\tilde{\mathcal{H}}_k \equiv g_k + \left(\frac{\tilde{\boldsymbol{\Lambda}}_k^T}{w_k} + \tilde{\boldsymbol{\Lambda}}_F^T \right) \mathbf{f}_k - \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \mathbf{C}_k \tag{2.86}$$

and substituting into the equations for the partial derivatives, yields the following set

of KKT conditions that must be satisfied for an optimal solution:

$$\sum_{i=0}^{N} \mathbf{X}_i D_{ki} - \frac{t_f - t_0}{2} \mathbf{f}_k = \mathbf{0} \tag{2.87}$$

$$\begin{aligned} - \quad & \sum_{i=1}^{N} \left(\frac{\tilde{\mathbf{\Lambda}}_i^T}{w_i} + \frac{\tilde{\mathbf{\Lambda}}_F^T}{w_k} \right) \frac{w_i}{w_k} D_{ik} + \tilde{\mathbf{\Lambda}}_F^T \sum_{i=1}^{N} \frac{w_i}{w_k} D_{ik} \\ & - \frac{t_f - t_0}{2} \left[- \frac{\partial g_k}{\partial \mathbf{X}_k} - \left(\frac{\tilde{\mathbf{\Lambda}}_k^T}{w_k} + \tilde{\mathbf{\Lambda}}_F^T \right) \frac{\partial \mathbf{f}_k}{\partial \mathbf{X}_k} + \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \frac{\partial \mathbf{C}_k}{\partial \mathbf{X}_k} \right] = \mathbf{0} \end{aligned} \tag{2.88}$$

$$\frac{\partial g_k}{\partial \mathbf{U}_k} + \left(\frac{\tilde{\mathbf{\Lambda}}_k^T}{w_k} + \tilde{\mathbf{\Lambda}}_F^T \right) \frac{\partial \mathbf{f}_k}{\partial \mathbf{U}_k} - \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \frac{\partial \mathbf{C}_k}{\partial \mathbf{U}_k} = \mathbf{0} \tag{2.89}$$

$$\boldsymbol{\phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) = \mathbf{0} \tag{2.90}$$

$$\tilde{\mathbf{\Lambda}}_0^T + \frac{\partial \mathbf{\Phi}}{\partial \mathbf{X}_0} - \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}_0} = \mathbf{0} \tag{2.91}$$

$$\tilde{\mathbf{\Lambda}}_F^T - \frac{\partial \mathbf{\Phi}}{\partial \mathbf{X}_f} + \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}_f} = \mathbf{0} \tag{2.92}$$

$$- \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \frac{\partial \tilde{\mathcal{H}}_k}{\partial t_0} + \frac{1}{2} \sum_{k=1}^{N} w_k \tilde{\mathcal{H}}_k - \frac{\partial \mathbf{\Phi}}{\partial t_0} + \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial t_0} = \mathbf{0} \tag{2.93}$$

$$\frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \frac{\partial \tilde{\mathcal{H}}_k}{\partial t_f} + \frac{1}{2} \sum_{k=1}^{N} w_k \tilde{\mathcal{H}}_k + \frac{\partial \mathbf{\Phi}}{\partial t_f} - \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial t_f} = \mathbf{0} \tag{2.94}$$

$$\mathbf{C}_k \leq \mathbf{0} \tag{2.95}$$

$$\tilde{\mu}_{jk} : \begin{cases} = 0 & \text{when } C_{jk} < 0 \\ \leq 0 & \text{when } C_{jk} = 0 \end{cases} \tag{2.96}$$

$$\mathbf{X}_f = \mathbf{X}_0 + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \tag{2.97}$$

$$\tilde{\mathbf{\Lambda}}_F - \tilde{\mathbf{\Lambda}}_0 - \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \left[-\frac{\partial g_k}{\partial \mathbf{X}_k} - \left(\frac{\tilde{\mathbf{\Lambda}}_k^T}{w_k} + \tilde{\mathbf{\Lambda}}_F^T \right) \frac{\partial \mathbf{f}_k}{\partial \mathbf{X}_k} + \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \frac{\partial \mathbf{C}_k}{\partial \mathbf{X}_k} \right] = \mathbf{0} \tag{2.98}$$

The pseudospectral discretization and NLP transcription formulation given by equations (2.74) - (2.84) form the basis for GPOPS software. The NLP problem is then solved by calling SNOPT to find the solution that minimizes the cost functional (2.82) and satisfies the KKT conditions (2.87) - (2.98). Reference [8] continues by formulating the pseudospectral numerical solution to the HBVP to show equivalence between the KKT multipliers and costates of the numerical HBVP approximation, showing that the GPM generates a solution that well approximates the numerical solution to the HBVP. Therefore, the GPM as coded in GPOPS is a powerful numerical tool for solving many types of OC problems.

2.3.4 Numerical Methods for Optimal Control Summary.

Topics related to numerically solving both indirect and direct formulations of the OC problem have been described in this section. Numerical methods applied to the indirect formulation solve the OC problem that has been converted to the Hamiltonian Boundary Value Problem while methods applied to the direct formulation solve the OC problem directly by transcribing the original problem into a parameter optimization problem and applying nonlinear programming techniques. This section has been presented purposely to discuss basic numerical techniques that when combined result in very powerful pseudospectral methods, particularly the Gauss Pseudospectral Method. References have also been provided that show equivalence between the direct pseudospectral and indirect pseudospectral solutions, adding confidence that direct pseudospectral transcription methods generate solutions that meet first-order optimality conditions derived via the Calculus of Variations. Lastly, a brief summary of the key equations in the development of the Gauss Pseudospectral Method and the

Karush-Kuhn-Tucker optimality conditions were presented in order to put together the key concepts of collocation and nonlinear programming applied to solving OC problems. This formulation is the basis for the *Gauss Pseudospectral Optimization Software* which performs pseudospectral discretization of the continuous OC problem, formulates the nonlinear programming problem, and calls the nonlinear programming solver to find the solution that meets optimality conditions. This is a powerful software tool that will be combined with stochastic methods, which will be discussed in the next section, to solve OC problems for systems with uncertainties.

2.4 Numerical Methods for Stochastic Differential Equations

The goal of the research presented in this dissertation is to numerically solve stochastic OC problems, or equivalently trajectory optimization problems, with uncertainties on the states, constraints, or both. The material presented thus far applies to deterministic problems. That is to say that the state of the system is perfectly modeled by differential dynamic equations or measurements of the system are certain. In reality, there are a multitude of uncertainties to cope with in trajectory optimization problems such as noisy measurements, uncertainties in system models, uncertainties in obstacle locations, random disturbances such as wind gusts, and so on. The presence of these uncertainties makes the problem more challenging, but does not make the methods described for solving OC problems invalid. Indeed, the aforementioned methods can be combined with additional methods to address randomness. In this section, brief descriptions of concepts related to the solution of stochastic problems are presented. The discussion will lead to *Generalized Polynomial Chaos* methods that when combined with deterministic methods provides a powerful tool for solving stochastic OC problems.

2.4.1 Survey of Stochastic Methods.

Solving nonlinear OC problems is very difficult and the literature is especially sparse on the topic of nonlinear stochastic OC [22]. Many nonlinear OC techniques addressing problems where systems are subject to uncertainties (noise), i.e. stochastic OC problems, attempt to linearize the systems and make simplifying assumptions about the noise. Several well-developed tools become available by considering linear systems with additive white (Gaussian) noise. Linear filtering techniques enable separation of noise from the signal of interest, for example measurement of system states or rejection of noise in the plant model, by minimizing the mean-square estimation error. Wiener filters (frequency domain) [15], and Kalman filters [51, 52] (state-space domain) provide such estimates. $\mathcal{H}_\infty$ estimation provides another method for estimating system states by minimizing the infinity-norm of the gain between disturbance inputs and estimation error [20]. The resulting state estimates can then be used as inputs to LQR controllers. $\mathcal{H}_2$ control is similar to the LQR, but whereas the LQR is not a stochastic control algorithm, the $\mathcal{H}_2$ controller rejects noise and minimizes the closed-loop system 2-norm. Linear filtering and estimation techniques work well for a wide range of engineering problems, but filtering applied to nonlinear systems is much more difficult since state estimation and control can not be separated from each other, noise rejection, state estimation, and control objectives compete with each other, and a given input probability distribution, Gaussian for example, generally does not map to the same distribution in the output [31, 46]. That said, there are effective nonlinear filtering techniques available, such as particle filters, that work quite well but may be difficult to implement. A survey of nonlinear filtering techniques is presented in [18]. For these reasons, other approaches are needed to solve the general nonlinear stochastic OC problem.

Representative nonlinear stochastic control methods based on DP, which are fun-

damentally sequential decision-making algorithms, are presented in [56], [22] and [73]. Reference [56] discretizes the continuous state dynamics using Markov Chains, which are random process where the state of the system at time step $k+1$ only depends on the state of the system at step k. The key element in the Markov structure is definition of a transition model, which is stated as: given the system state $x_k = s$ at time step k, the probability that the system will reach state $x_{k+1} = s'$ when control u_k is applied is $P(s, u, s')$ [74]. Written in another way, the transition probability is:

$$P_{s,s'}(u_k) = Prob\{x_{k+1} = s' | x_k = s, u_k\} \tag{2.99}$$

Kushner states that minimizing a DP cost functional using the Markov chain representation of the stochastic processes results in optimal solutions that are good approximations to the original problem. Rogers presents a slightly different DP approach, again using Markov chains, where "pathwise" deterministic optimization is conducted on many randomly sampled trajectories to find an approximate solution. He notes that this is a novel approach in solving stochastic OC problems, but numerical implementation needs further study [73]. The difficulties of applying these DP-based methods are that it is not clear how to define the transition probabilities based on the randomness in the OC problem and algorithms for actually implementing these methods are not clearly stated.

Reference [22] uses a "backward search algorithm" that is based on Bellman's principle of optimality to generate global control solutions to stochastic optimal control problems with fixed-state terminal conditions and state and/or control constraints that "can handle strongly nonlinear and non-smooth systems, can include various state and control constraints and leads to global solutions." Crespo and Sun's method can be summarized as:

- Formulate the SDE using random inputs modeled as independent Wiener pro-

cesses, which are also known as *Brownian motion*, and is a method of modeling uncertainty as a differential equation by passing white Gaussian noise through an integrator [15, 63].

- Specify the cost as the expectation of a Bolza-type functional.
- Apply Bellman's principle of optimality.
- Implement a backward search algorithm beginning at the terminal boundary conditions and searching backward through the space and time domains to find the optimal solution.

Through three examples, Crespo concluded that the "method leads to controls with excellent performance" and indicated that improvements and further studies are needed to solve higher-dimensional problems and "rigorous study of convergence and stability of the method has yet to be done" [22].

Girardeau compared two sample-based stochastic OC schemes in [43]: a scenario tree method and particle method. He considered a continuous-time finite horizon problem of controlling a dynamical system perturbed by exogenous noise, assumed to be uniformly distributed, using a mean squared error (MSE) cost function. Both methods involve discretization of the state space, time domain, constraints, and cost function and essentially become large sets of deterministic problems. In the scenario tree method, he defined a *scenario* as a set of sampled noise values at each time step in the discrete time domain. The method basically solves the discrete problem using a DP approach that uses knowledge of all previous state, control, and noise values to sequentially determine the state feedback control function at the current state and sampled noise value that minimizes the expected cost at future points in time. The control solution is then used to propagate the state to the next time point through the discretized dynamics. A numerical example indicates that the method is computationally expensive since numerical integration of the MSE cost function

is required at each time step and that as the time horizon increases, the number of scenarios needed to achieve an accurate solution exponentially increases. The particle method proposed is a mix of stochastic and dynamic programming where the necessary optimality conditions are formulated using variational techniques, i.e. a costate vector is generated, and then solved using sampling. The solution is found using gradient search methods where the sample grid adaptively discretizes portions of the state space that are most visited by the resulting deterministic optimal solutions. Girardeau showed that this method does not have the same exponential growth of scenarios as in the scenario tree method, but becomes unsolvable as the dimension of the state space increases. While these methods seem to be able to solve a nonlinear stochastic OC problem, the formulations and examples do not include fixed final states, thus it's not evident how they could be applied or adapted to solve a broad range of OC problems.

It was noted in section 2.3.3.2 that solving deterministic OC problems using the GPM generates solutions that well approximate the numerical solution to the HBVP. If OC problems are thought of as of a system of ODEs, as in the indirect HBVP formulation, then tools for solving SDEs may be available which can be combined with the GPM to solve stochastic OC problems. As will be shown later, solving SDEs will ultimately depend on repeated application of deterministic ODE solvers and, to foreshadow, there may be spectral, or more specifically pseudospectral, representations of stochastic solutions that are accurate, computationally efficient, and easy to use. A survey of solution methods for SDEs, specifically *Monte-Carlo Simulation*, *Perturbation Methods*, *Moment Equations*, *Operator Methods*, and *Generalized Polynomial Chaos*, are summarized in [87] and [88] and presented with greater detail in [39]. The discussion below is a paraphrased synthesis of the information in these reference with complementary references inserted as appropriate.

The most common technique for solving SDEs is the *brute force* Monte-Carlo Simulation (MCS) method. The MCS method generates random samples of a stochastic variable taken from an assumed probability distribution function (PDF). Each random sample is subsequently inserted into the SDE, making it a deterministic problem that can be solved using deterministic differential equation solution methods, such as those discussed in section 2.3, or existing ODE and PDE numerical solvers in MATLAB®. A set, or ensemble, of solutions is collected from which the statistical information, such as expected value (mean), variance, and covariance, is calculated. Convergence is checked at some arbitrary interval, after every 500 or 1000 samples for example, and is said to have been achieved when the change in expected value between successive intervals is below a desired tolerance. The solution to the problem is the combination of expected value, the most likely solution to the stochastic problem, and second moment variance and covariance information, which characterizes the distribution of the solution by showing how much the solution varies from the mean as a function of uncertain parameters and correlations between solution variables, respectively. The MCS method is relatively easy to implement using existing deterministic differential equation solvers, will eventually converge to a solution regardless of the PDF of the process, and is not limited by the amount of uncertainty in the variable. However, it's well known that the mean converges slowly, on the order of $\frac{1}{\sqrt{k}}$, where k is the number of realizations (samples) of the random variable, which implies high computational burden, especially in problems with multiple random variables. Even though the MCS is computationally burdensome, it is often capable of quantifying effects of uncertainty in problems were other methods seem to fail and is a valuable method in verifying results obtained using other methods [39].

Perturbation Methods, *Moment Equations*, and *Operator Methods* offer alternatives to solving SDEs, but will not be used in this research and are only mentioned

here for the sake of completeness. *Perturbation Methods* expand the field of the random variables about their mean values using truncated Taylor series expansions. These methods are limited to SDEs where the uncertainties are generally less than 10 percent and become extremely complicated for systems where there are more than two random variables. *Moment Equation* methods attempt to determine the moments of the random solution directly from the stated SDE problem. These methods are difficult to use since moments beyond the second moment (variance and covariance) are required but most likely unknown. *Operator Methods* attempt to express the inverse of the stochastic operator of the SDE as a Neumann series. Unfortunately, this method is also limited to problems with small uncertainties and are not very useful. Babuska and Chatzipantelidis present formulation, analysis, and numerical examples of perturbation and operator methods applied to solve elliptic PDEs in [3].

Another powerful technique is the Generalized Polynomial Chaos (gPC) method. The gPC is a spectral approximation of a stochastic process used to solve SDEs, analogous to the deterministic spectral methods described in section 2.3.3, where solutions to SDEs are expressed as expansions of the random parameters using orthogonal polynomials. As with deterministic PSMs, there are both Galerkin and collocation implementations of the gPC. Research, which will be specifically cited later, has shown that the best choice of the orthogonal basis polynomials depends on the type of PDF describing the random inputs and that spectral convergence is achieved by selecting the appropriate basis set. While both gPC implementations are spectrally accurate and demonstrate "quick convergence," the added benefit of the collocation method, similar to Monte-Carlo Simulation, is that it is also fairly simple to implement since it only requires repeated deterministic solutions at a set of collocation points [87]. The gPC will be a central concept in this research and will be presented more thoroughly in the next subsection.

2.4.2 Generalized Polynomial Chaos.

In 1938 Norbert Wiener introduced, for what may be the first time, a "Homogeneous Chaos" which used Hermite polynomials to approximate Gaussian processes [83]. Xiu summarizes Wiener's work in [91] and cites references to theorems that have shown that the Hermite Chaos converges for any second-order random process, which are processes with finite second moments, and in fact, converges spectrally for Gaussian processes due to the fact that the Hermite weighting function is the same as that of the Gaussian PDF. Thus, Wiener's Homogeneous Chaos, known also as Wiener's Chaos or the Hermite Chaos, enables approximation of second-order random processes, which are representative of many physical systems, using orthogonal polynomials.

Building on cited work of Askey, who extended Wiener's Homogeneous chaos by linking other orthogonal polynomial bases and statistical distributions by matching weighting functions, Xiu presented a method for solving SDEs based on Galerkin projections that takes advantage of these correspondences [91]. Through numerical experimentation on an ODE where an exact solution is available, he demonstrated spectral convergence of the errors when these correspondences were followed. Xiu found that convergence was still obtained, but substantially slower, when the polynomial bases were not chosen to match the probability distribution. This generalization of Wiener's Hermite Chaos is known as the gPC and a table with the polynomial correlations to probability distributions is given in the table below.

There are two forms of gPC: Galerkin and collocation. The Galerkin form is described in [86], [87], and [91] and the procedure summarized below is a paraphrase from these references.

Consider the generic stochastic differential equation, assuming that boundary con-

Table 2.3. Correlation between probability density functions and orthogonal polynomial bases [88]

Distribution Type	Random Variable Distribution	Polynomial Basis	Domain
Continuous	Gaussian	Hermite	$[-\infty, \infty]$
	Gamma	Laguerre	$[0, \infty]$
	Uniform	Legendre	$[a, b]$
Discrete	Poisson	Charlier	$0, 1, 2, ...$
	Binomial	Krawtchouk	$0, 1, ..., N$
	Negative Binomial	Meixner	$0, 1, 2, ...$
	Hypergeometric	Hahn	$0, 1, ..., N$

ditions are specified:

$$\mathcal{L}(\mathbf{x}, t, \theta; u) = f(\mathbf{x}, t; \theta) \tag{2.100}$$

where $u = u(\mathbf{x}, t; \theta)$ is the solution and $f(\mathbf{x}, t; \theta)$ is the source term. The operator $\mathcal{L}$ is a differentiation operator which can generally involve time and space derivatives resulting in a generally nonlinear differential equation that can either be an ODE or PDE depending on the construction of the operator. The random parameter, θ, represents uncertainty in the system which can be present in the boundary conditions, initial conditions, physical properties of the system, etc. The presence of uncertainty causes the solution, u, to be a random process, which can be spectrally expanded as a polynomial chaos as:

$$u(\mathbf{x}, t; \theta) = \sum_{i=0}^{P_c} u_i(\mathbf{x}, t) \Phi_i^c(\boldsymbol{\zeta}(\theta))) \tag{2.101}$$

where $\boldsymbol{\zeta}(\theta)$ is a random variable (RV) that is a function of the random input θ. Notice that the sum has been truncated at the P_c^{th} term and the "c" subscript is used to indicate "chaos" to differentiate this symbol from a previously used one. The set of basis functions, $\{\Phi_i^c\}$, are orthogonal polynomials chosen based on the

assumed distribution of the random variable $\zeta(\cdot)$ as described in Table 2.3. Again, the superscript "c" is used to differentiate the "chaos" expansion basis functions from a similar formerly used symbol. The number of expansion terms, $(P_c + 1)$, is determined by the dimension, n, of the random variable $\boldsymbol{\zeta}$ and the highest order, p, of the polynomials $\{\Phi_i^c\}$ as:

$$(P_c + 1) = \frac{(n+p)!}{n!p!} \tag{2.102}$$

Substituting (2.101) into (2.100) yields:

$$\mathcal{L}\left(\mathbf{x}, t, \theta; \sum_{i=0}^{P_c} u_i \Phi_i^c\right) = f(\mathbf{x}, t; \theta) \tag{2.103}$$

To minimize the approximation error, a Galerkin projection is used to project (2.103) onto each orthogonal polynomial basis element, $\{\Phi_i^c\}$, using the following inner product definition:

$$\langle \mathcal{L}\left(\mathbf{x}, t, \theta; \sum_{i=0}^{P_c} u_i \Phi_i^c\right), \Phi_k^c \rangle = \langle f, \Phi_k^c \rangle \quad (k = 0, 1, ..., P_c) \tag{2.104}$$

References [86], [87], and [91] show that defining the inner product this way ensures minimum approximation error. The result of the projection is a set of $(P_c + 1)$ coupled equations for each $u_i(\mathbf{x}, t)$, where $i = 0, 1, \ldots, P_c$, that are deterministic since the uncertainty has been transferred into the polynomial bases. However, the inner products in (2.104) generate $(P_c + 1)$ integral equations which are solved numerically using quadrature rules. Therefore, the time domain and the space of $\mathbf{x}$ need to be discretized so that deterministic methods to solve the integral equations can be used. It should be obvious that solving SDEs using the Galerkin form of the gPC becomes increasingly difficult as the number of states increases. Further development of Galerkin gPC methods used to solve elliptic-type stochastic PDEs, most using

Finite Element Methods (FEM) for discretization of the physical space, can be found in [5, 23, 34, 40, 90]. A solution method for hyperbolic stochastic PDEs using the Galerkin gPC method, with application to a wave equation example, is presented in [45].

The Galerkin formulation of the gPC has been discussed and now the collocation form will be presented. As has been stated, the gPC is a spectral approximation of stochastic process, which is analogous to the discussion of spectral methods in section 2.3.3. Detailed development of the collocation type gPC for solving stochastic PDEs and ODEs can be found in [4, 25, 89] and rigorous convergence analysis is performed in [4, 66]. Pulch adapts the collocation gPC to solve BVPs in [70], noting that other numerical methods like shooting, finite difference, and spectral expansion methods described in section 2.3 have analogous applications to stochastic ODEs. The focus here will be placed on the approach formulated to solve ODEs and Differential Algebraic Equation (DAE)s, which are the most commonly seen types of equations in OC problems and the types addressed in this research effort. More details are presented than in the Galerkin case since the collocation formulation will be used in this research. The development of the key equations of the collocation gPC solution process are paraphrased from [86]. The author's notation has been preserved as much as possible since it appears to be fairly standard in the literature. The step-by-step procedure combining the GPM and collocation form of the gPC used in this research will be concisely presented in Chapter III and application of the algorithm to two types of optimization problems, an OC problem with state uncertainties and a TO problem with uncertainties effecting the cost functional, will be presented in Chapters IV and V, respectively.

The collocation form gPC algorithm development will consider a general system

of DAEs in the following form:

$$\begin{cases} F(t, \mathbf{y}, \mathbf{y}', ..., \mathbf{y}^{(l)}, \mathbf{p}) & = 0 \\ g(t_0, \mathbf{y}(t_0), ..., \mathbf{y}^{(l)}(t_0), \mathbf{p}) & = 0 \end{cases} \quad t \in (t_0, T] \tag{2.105}$$

where the key variables are:

- **State variables**: $\mathbf{y} = (y_1, ..., y_J) \in \mathbb{R}^J$
- **Stochastic input parameters (random variables)**: $\mathbf{p} = (p_1, ..., p_N) \in \mathbb{R}^N$
- **Outputs** or **observables**: $\mathbf{z} = G(\mathbf{y}) = (z_1, ..., z_K) \in \mathbb{R}^K$. Generally speaking, the outputs or observables can be the states or some function of the states that defines quantities of interest. When applied specifically to optimization problems, the observables can be the states, costates, controls, cost, terminal time (free-final-time problem), Hamiltonian, and so on.

Additionally, its necessary to define the probabilistic structure. The vector $\mathbf{p}$ is defined as an N-variate random vector whose elements are assumed to be independent of each other. If the random elements in $\mathbf{p}$ are not independent, then a transformation, such as the Karhunen-Loève decomposition or Rosenblatt transformation, should first be applied resulting in a set of uncorrelated RVs that are assumed to be independent [11, 39, 87, 89, 92]. Furthermore, assume that the probability space is defined by the tuple $(\Omega, \mathcal{A}, \mathcal{P})$, where the space of all possible basic outcomes is Ω, is equipped with σ-algebra $\mathcal{A}$, which can be thought of as the space of possible events that can be derived from Ω, and probability measure $\mathcal{P}$ [40]. Now define key probabilistic variables in the gPC collocation development:

- **PDF**: ρ_i is the PDF of the random variable $p_i(\omega)$, for $i = 1, ..., N$ and $\omega \in \Omega$.

Because the elements of $\mathbf{p}$ are independent, the joint PDF is given by:

$$\rho(\mathbf{p}) = \prod_{i=1}^{N} \rho_i(p_i) \tag{2.106}$$

- **Finite domain of ρ_i**: $\Gamma_i \equiv p_i(\Omega)$ transforms the infinite domain Ω to the finite domain Γ which are intervals in $\mathbb{R}$, for $i = 1, ..., N$. The total finite domain is defined as:

$$\Gamma \equiv \prod_{i=1}^{N} \Gamma_i \subset \mathbb{R}^N \tag{2.107}$$

One goal of the gPC expansion is to approximate the RVs in $\mathbf{p}$ using orthogonal polynomials. The one-dimensional orthogonal polynomial space, W^{i,d_i}, of highest degree, d_i, is defined by:

$$W^{i,d_i} \equiv \{v : \Gamma_i \to \mathbb{R} : v \in span\{\phi_m^c(p_i)\}_{m=0}^{d_i}\} \quad (i = 1, ..., N) \tag{2.108}$$

The one-dimensional polynomial basis set, $\{\phi_m^c(p_i)\}$, which has $d_i + 1$ basis elements and chosen according to Table 2.3, satisfies the following orthogonality conditions:

$$\int_{\Gamma_i} \rho_i(p_i)\phi_m^c(p_i)\phi_n^c(p_i)dp_i = \left[\int_{\Gamma_i} \rho_i \phi_m^{c^2} dp_i\right] \delta_{mn} = h_m^2 \delta_{m,n} \tag{2.109}$$

where the polynomial basis elements are made orthonormal with respect to the PDF by applying appropriate scaling factors such that $h_m^2 = 1$ for all m [87].

The N-variate space of orthogonal polynomials, $W_N^{P_c}$, of total degree of at most P_c, is constructed by taking the tensor product, i.e. all possible combinations, of the N one-dimensional orthogonal polynomial bases, written as:

$$W_N^{P_c} \equiv \bigotimes_{|\mathbf{d}| \leq P_c} W^{i,d_i} \tag{2.110}$$

where the relationship between d_i and P_c is given by:

$$|\mathbf{d}| = \sum_{i=1}^{N} d_i \leq P_c \tag{2.111}$$

and satisfy the orthogonality conditions:

$$\int_{\Gamma} \Phi_m^c(\mathbf{p})\Phi_n^c(\mathbf{p})\rho(\mathbf{p})d\mathbf{p} \equiv \mathbb{E}\left[\Phi_m^c(\mathbf{p})\Phi_n^c(\mathbf{p})\right] = \delta_{mn} \tag{2.112}$$

The effect of (2.111) is to select a subset of basis elements from the tensor product space defined by (2.110) in order to reduce computational burden, however the entire set of basis elements resulting from the tensor product can be used [87], in which case M is equal to the total number of basis elements resulting from the tensor product of the one-dimensional basis sets.

Now the approximation of the output, $\mathbf{z}$, as a function of the random parameters, $\mathbf{p}$, can be written as:

$$\mathbb{P}_N^{P_c}\mathbf{z} \equiv \mathbf{z}_N^{P_c}(\mathbf{p}) = \sum_{m=1}^{M} \hat{\mathbf{z}}_m \Phi_m^c(\mathbf{p}) \tag{2.113}$$

where:

$$M = \binom{N + P_c}{N} \tag{2.114}$$

or is the total number of tensor product elements if the entire set of basis elements are being used. The projection operator in (2.113) , $\mathbb{P}_N^{P_c}$, projects the discrete probability space Γ onto the polynomial space $W_N^{P_c}$. The coefficients, $\hat{\mathbf{z}}_m$, are determined by:

$$\hat{\mathbf{z}}_m = \mathbb{E}\left[\mathbf{z}(\mathbf{p})\Phi_m^c(\mathbf{p})\right] = \int_{\Gamma} \mathbf{z}(\mathbf{p})\Phi_m^c(\mathbf{p})\rho(\mathbf{p})d\mathbf{p} \quad (m = 1, ..., M) \tag{2.115}$$

The key element in the gPC collocation method is to approximate the integral in (2.115) using a Gaussian quadrature rule. To apply quadrature, a set of colloca-

tion points must be chosen and associated quadrature weights calculated. For each random dimension, Γ_i, $i = 1, \ldots, N$, q_i collocation nodes and weights are chosen to correspond with the polynomial representation described in Table 2.3 and determined by information given in Table 2.2. The N-dimensional grid of Q collocation nodes and weights is obtained by taking the tensor product of the one-dimensional node and weight sets. Alternatively, a sparse grid can be generated to reduce the total number of tensor product nodes while providing accurate integral approximation using the Smolyak algorithm, first published in [77] and included, along with other sparse grid schemes, in [38]. In general, a tensor product grid works well for low-dimensional problems, but suffers the *curse of dimensionality* as N gets larger ($N > 5$) making sparse girds a better choice [4, 87]. The following figure shows representative two-dimensional ($N = 2$) collocation grids generated using both schemes. The tensor product grid was generated using 20 LG points in each dimension resulting in a total of $Q = 400$ collocation points and quadrature weights. The sparse grid was generated using a MATLAB®routine [10, 11] that generated nodes and weights associated with a fifth-order quadrature rule, resulting in $Q = 290$ quadrature points and weights. The

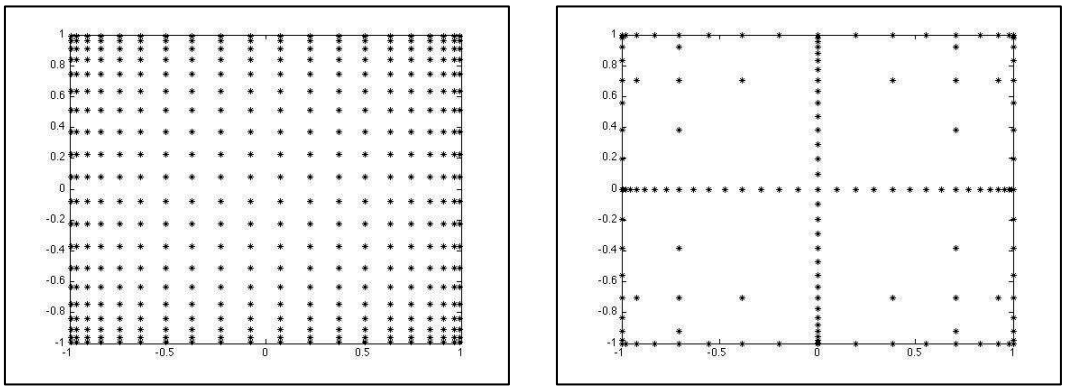


Figure 2.1. Collocation grids used in gPC collocation expansion. Left: Two-dimensional ($N = 2$) tensor product collocation grid with 20 LG points ($q_i = 20$) in each dimension. Right: Sparse collocation grid in two-dimensions based on 5^{th}-order quadrature rule.

collocation approximation to (2.115) based on the quadrature rule chosen is given by:

$$\hat{\mathbf{z}}_m = \sum_{j=1}^{Q} \mathbf{z}(\mathbf{p}_j)\Phi_m^c(\mathbf{p}_j)\alpha_j; \quad (m = 1, ..., M) \tag{2.116}$$

where $\mathbf{p}_j$ denotes the j^{th} collocation node of the random vector $\mathbf{p}_j = (p_{1,j}, ..., p_{N,j})$ and α_j denotes the associated quadrature weight, thus defining the set of collocation nodes and weights, $\{\mathbf{p}_j, \alpha_j\}_{j=1}^{Q}$. Note that $z(\mathbf{p}_j)$ is the deterministic solution using the j^{th} sample of the random vector. This gPC formulation is the basis of a general procedure which will be presented in Chapter III.

It can be seen that one advantage of the gPC collocation method, elegantly stated in [62], which also presents the gPC collocation algorithm development, is:

> ...while the standard Galerkin approach to Polynomial Chaos requires multi-dimensional summations over the stochastic basis functions, the stochastic collocation method enables to collapse those summations to a one-dimensional summation only.

In other words, a one-dimensional summation replaces multi-dimensional summations used in the Galerkin method to numerically approximate the integrals resulting from the inner products. Mathelin demonstrates this through formulation of a solution to a one-dimensional nozzle flow (fluids) problem where the Galerkin method results in a nine-dimensional summation, or more depending on boundary conditions. Other advantages to the gPC collocation method, as stated in [4, 87] are:

- The gPC collocation leads to solution of uncoupled deterministic problems where the gPC expansion is a post-processing step after the deterministic solutions at the collocation nodes have been found.
- The gPC can handle unbounded random variables, for example Gaussian or exponentially distributed random variables.

- Like the Galerkin form of the gPC, it is spectrally accurate, i.e. the algorithm gives exponential convergence.

The gPC algorithms have been applied to solve SDEs in many engineering disciplines such as: analysis of nonlinear integrated circuit response [80], various types of fluids problems [45, 57, 58, 62, 92], heat transfer [41, 82, 93], and others referred to by the authors of these references. Applications of the gPC methods applied to stochastic OC problems are rare. One such application is found in [11] where a nonlinear parabolic PDE-based OC problem is considered using "multi-grid" physical domain discretization. The difference between the work in this reference and the research developed in this document is that the former is not using a spectral method to solve the deterministic problems at the discretized random sample points. No references containing work similar to that described in this document have been found thus far.

2.4.3 Numerical Methods for Stochastic Differential Equations Summary.

This section began with a survey of commonly used techniques to solve stochastic optimal control problems. The discussion was then expanded to consider methods used to solve stochastic differential equations, which may be adapted to solve stochastic optimal control problems. Specifically, Generalized Polynomial Chaos methods were presented since they provide spectral representations of the random space that are analogous to the spectral representations of deterministic problems discussed earlier in the document and provide accurate approximations to the solution. Two forms of the Generalized Polynomial Chaos were summarized: Galerkin and Collocation. Again, parallels between the deterministic and stochastic versions of these algorithms are obvious, where the collocation approach is easier to implement while providing accuracy that is similar to the Galerkin method. The abbreviated version of the

collocation method was given at the end of the section, which will be adapted and applied to a sample stochastic optimal control problem in the next chapter.

2.5 Related Concepts Summary

This chapter began with a brief exposition on the historical evolution of OC theory. The general formulation of a continuous OC problem was posed along with the traditional solution technique applying CV theory and the PMP to transform the problem into an HBVP. Recognizing that analytically solving the resulting set of coupled nonlinear differential equations defining first-order optimality conditions may be impossible, the formulation provided motivation for discussing indirect and direct numerical solution methods.

Indirect methods seek to numerically solve the system of ODEs comprising the HBVP. As such, discussion of indirect methods presented focused primarily on numerical integration rules as implemented in the time-marching, shooting, and collocation methods. Direct methods, on the other hand, avoid analytical derivation of the necessary conditions for optimality like those in the HBVP by transcribing the original problem into a parameter optimization problem and applying NLP techniques. Basic concepts of formulating an NLP problem were presented and how direct shooting and collocation methods are cast as NLP problems.

Collocation and NLP concepts were then put together to form the basis for PSMs, which can be applied to both indirect HBVP and the direct NLP problem formulations. After general discussion of spectral methods highlighting basic concepts and available forms, particular attention was paid to the direct Gauss Pseudospectral Method where equations transcribing the optimal control problem to nonlinear programming problem, Karush-Kuhn-Tucker optimality conditions, equivalence of GPM and HBVP solutions, and GPM implementation in GPOPS were discussed.

Emphasis was then shifted to address stochastic solution methods. A summary of commonly used techniques, most notably MCS, to quantify uncertainty in SDEs ultimately led to presentation of the collocation form of the gPC, which is a spectral method analogous to the deterministic spectral methods presented in the chapter. The gPC uses orthogonal polynomial bases, chosen based on correlation between weighting functions of the assumed PDFs of the uncertain parameters and the polynomial bases, to approximate SDE solutions as functions of the uncertain parameters. The collocation form applies deterministic solution techniques at the collocation points to generate a set of deterministic solutions used in stochastic computations. Therefore, it seems possible to use GPOPS as the deterministic solver implementing the GPM to provide spectrally accurate approximations to OC problems, which have been shown to be equivalent to numerical solutions of the CV-based HBVP, in the gPC construct to extend the method to be applicable to stochastic optimization problems. Combining the GPM and gPC spectral methods into a hybrid algorithm will be presented in the next chapter.

III. Hybrid GPM-gPC Algorithm

The focus of this research is to determine if a method that integrates the GPM discussed in section 2.3.3.2 into the collocation form of the gPC discussed in section 2.4.2 is capable of solving nonlinear optimization problems by quantifying the effects random parameters have on the solutions. Using the GPM numerical solution package GPOPS in place of the differential equation solvers traditionally used in the gPC construct will provide a set of minimum cost numerical solutions to the sampled optimization problems that satisfy boundary, path, and bounded state and control constraints. Since these solutions are spectrally accurate and have been shown to be equivalent to numerical solutions of the HBVP, they are the best numerical approximations available to the true solution and are suitable for stochastic computations.

The algorithm presented in this chapter, as shown in figure 3.1 and discussed in the process steps, is an adaptation of the gPC collocation algorithm published in [86]. The gPC algorithm presented in the reference was originally written to solve SDEs by applying IVP and BVP differential equation routines to generate the deterministic solution set that is used in stochastic computations. Xiu's algorithm has been adapted to form a hybrid algorithm that is applicable to OC and TO problems by using GPOPS in place of the IVP and BVP solvers [86]. The remainder of this document will be focused on application of the hybrid algorithm combining the GPM and gPC collocation methods. For convenience, future use of the gPC abbreviation refers to the collocation form of the gPC and the term *hybrid algorithm* refers to the collocation form of the gPC where the GPM tool, GPOPS, is used as the deterministic OC solver in the gPC collocation construct. The hybrid algorithm is discussed in the following five steps.

Step #1: Choose a set of collocation nodes and quadrature weights, $\{\mathbf{p}_j, \alpha_j\}_{j=1}^{Q}$, in the finite probability space, Γ.

Since the hybrid algorithm will generate a spectral approximation of the solution to the stochastic problem, points that match both the quadrature rule evaluating the expansion coefficients given by (2.115) and the polynomial basis elements Φ_m should be used. For example, if Hermite orthogonal polynomial bases are chosen to approximate a Gaussian distribution as suggested in [87], [88], and Table 2.3, then Gauss-Hermite quadrature is an appropriate integration rule and will be most accurate using Gauss-Hermite nodes and weights. Furthermore, the Gauss-Hermite quadrature nodes will define the finite random domain Γ. The Gaussian distribution is supported on $\Omega = (-\infty, \infty)$ with an infinite number of possible realizations, but when the quadrature points are chosen, Γ will be supported on some bounded region in $\mathbb{R}$ with a finite number of points. The grid of Q points is constructed by choosing q_i collocation nodes and quadrature weights in each random domain, Γ_i, for $i = 1, \ldots, N$, and then taking the tensor product of the N individual sets. As mentioned in Chapter II, there is an abundance of MATLAB®functions available for download on the MathWorks™ website to generate quadrature nodes and weights. Descriptions of other quadrature point sets and rules can be found in [14], [33], and [81]. Each node in the grid is used as an input to the governing stochastic OC problem equations resulting in a set of deterministic problems that are solved in **Step #2** with existing numerical tools like GPOPS.

Step #2: Using each of the Q sample points, $\mathbf{p}_j$, solve the set of deterministic OC problems using GPOPS and build the observables array $\mathbf{z} = G(\mathbf{x})$, consisting of state trajectories, control solutions, and cost values as well as any other variables of interest such as costate trajectories, Hamiltonain values, terminal time, or any other variables

available in the GPOPS solution data.

Using the GPM to solve the Q deterministic OC problems at the collocation points is one reason why this hybrid algorithm is unique. Reference [86] states that any deterministic solver can be used, which for the author's purposes was an IVP solver. MATLAB®provides effective solvers, such as *ODE45* for IVPs and *BVP4c* for BVPs, that could also be used to solve the deterministic set of HBVP differential equations at the collocation points. However, using the direct GPM method coded in the GPOPS software provides spectrally accurate solutions to the deterministic sampled problems, if they exist, that are equivalent to numerical solutions of the HBVP differential equations, satisfy optimality conditions, minimize the performance index, and satisfy the boundary conditions while avoiding complexities of explicitly deriving and solving the HBVP differential equations mentioned in sections 2.2.2 and 2.3.1. Therefore, GPOPS, or similar PSM solvers, provides the best possible approximations to the Q deterministic problems whose solutions are used in **Step #3** to calculate the expansion coefficients.

Step #3: Evaluate the expansion coefficients using (2.116), which is restated below:

$$\hat{\mathbf{z}}_m = \sum_{j=1}^{Q} \mathbf{z}(\mathbf{p}_j)\Phi_m^c(\mathbf{p}_j)\alpha_j; \quad (m = 1, ..., M) \tag{3.1}$$

These expansion coefficients contain the information necessary to approximate the statistical properties of the solution constructed in **Step #4**.

Step #4: Build the output approximation function using (2.113), which is restated below:

$$\mathbb{P}_N^{P_c}\mathbf{z} \equiv \mathbf{z}_N^{P_c}(\mathbf{p}) = \sum_{m=1}^{M} \hat{\mathbf{z}}_m \Phi_m^c(\mathbf{p}) \tag{3.2}$$

The solution to the stochastic problem given by (3.2) is essentially a set of distribution functions that estimate the outputs (observables) as functions of the random inputs, the statistics of which are determined in **Step #5**.

Step #5: Evaluate the statistics of the approximate solution.

The statistical information about the solution is the desired output and is easily calculated using the expansion coefficients. Expected value, variance, and covariance functions describe the most likely solution to the stochastic problem, characterizes how the solution varies with random inputs, and describes the dependencies between the outputs, respectively. Equations for these properties are derived in [87] and are stated below.

The expected value ($\mathbb{E}$) is:

$$\mathbb{E}(\mathbf{z}(t)) \approx \mathbb{E}(\mathbf{z}_N^P(t)) = \int \left[\sum_{m=1}^{M} \hat{\mathbf{z}}_m(t) \Phi_m(\mathbf{p}) \right] \rho(\mathbf{p}) dp = \hat{\mathbf{z}}_1(t) \tag{3.3}$$

The variance (var) is:

$$var\left[\mathbf{z}(t)\right] \approx \sum_{m=2}^{M} \left[\hat{\mathbf{z}}_m^2(t)\right] \tag{3.4}$$

The covariance (cov) is:

$$cov\left[z_i(t), z_j(t)\right] \approx \sum_{m=2}^{M} \left[\hat{z}_{i,m}(t) \hat{z}_{j,m}(t)\right] \tag{3.5}$$

This process was implemented in MATLAB®and used to investigate an optimal control problem with state uncertainties in Chapter IV and a trajectory optimization problem with constraint uncertainties in Chapter V with the following research questions in mind.

- How can an OC or TO problem with uncertain parameters be formulated so

that it can be solved using the hybrid GPM-gPC algorithm?

- Can the algorithm be applied to problems with various types of boundary conditions and constraints?
- Can the algorithm be applied to problems where uncertain parameters effect different equations?
- How do the hybrid algorithm solutions compare with MCS solutions?
- Can the hybrid algorithm solve problems where the uncertain parameters do not all have the same assumed PDF and thus use a mix of polynomial bases?
- Can the hybrid algorithm be applied to real-world scenarios to provide useful information to the users?

This concise chapter has served as a bridge between the GPM and gPC theoretical concepts presented in sections 2.3.3.2 and 2.4.2 and their implementation into the hybrid algorithm shown in Figure 3.1 and discussed in the step-by-step process listed. Several research questions have also been posed that will be investigated by applying the hybrid GPM-gPC algorithm to two types of optimization problems in the following two chapters.

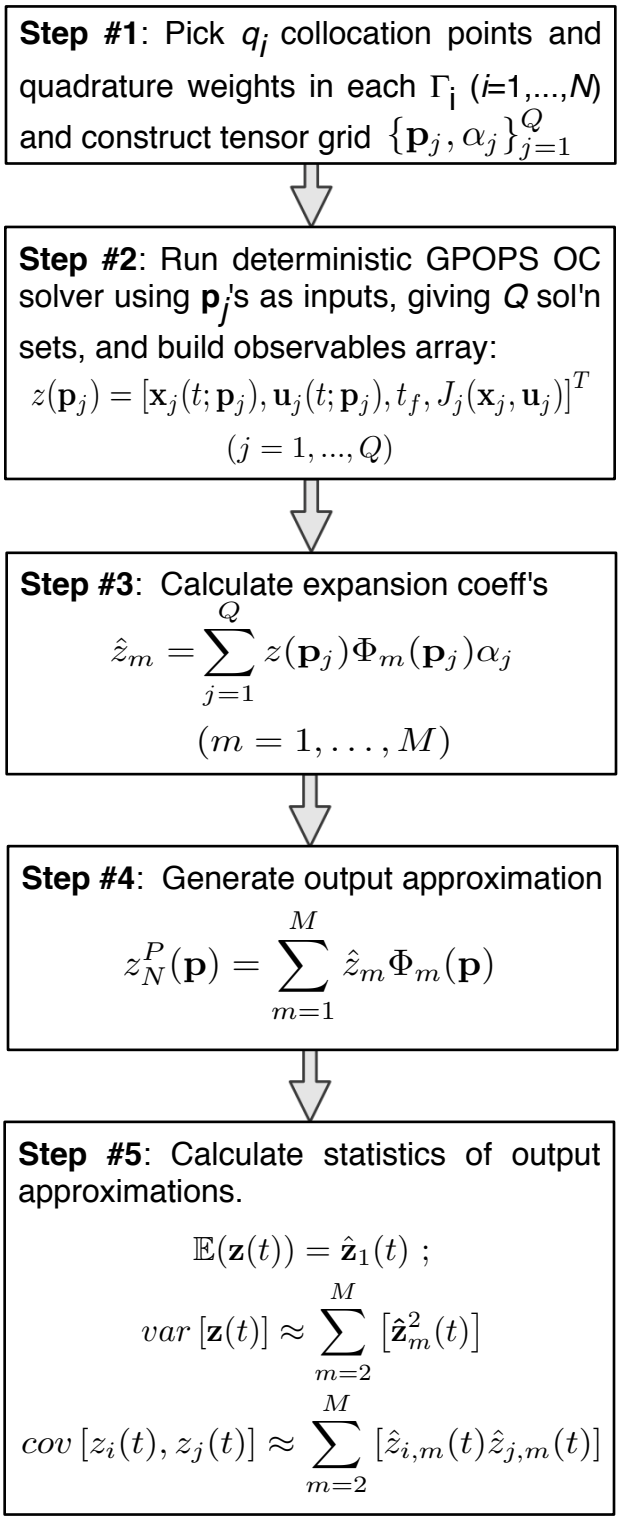


Figure 3.1. Hybrid algorithm combining GPM and gPC methods.

IV. Optimal Control Problem

In this chapter, the hybrid numerical algorithm combining the GPM and gPC methods outlined in Chapter III is applied to an OC problem that has nonlinear dynamics and performance index with multiplicative uncertainties in the state equations. The objective of this problem is to demonstrate the hybrid algorithm's ability to quantify the effects uncertain parameters have on OC solutions by applying it to a challenging nonlinear problem.

4.1 Problem Formulation

The OC concept demonstration problem is an adaptation of problem 5-2 in [54]. This deterministic baseline problem will be stated first in section 4.1.1 followed by modifications that were used to create the stochastic problem variant described in section 4.1.2. The stochastic problem is further developed following the gPC collocation modeling in section 2.4.2 resulting in a formulation that can be solved by applying the hybrid algorithm. The solution to the deterministic problem is included with the hybrid algorithm results in section 4.2 to compare with the expected value solution as a check that the hybrid algorithm solution is reasonable.

4.1.1 Deterministic Baseline Problem.

The deterministic baseline problem is stated as:

- Find $u(t) \in \mathbb{R}^1$ and $\mathbf{x}(t) \in \mathbb{R}^2$ that minimizes the cost functional (J):

$$J = \int_0^1 \frac{1}{2}[2x_1^2(t) + x_2^2(t) + u^2(t)]dt \tag{4.1}$$

- subject to:

$$\dot{x}_1(t) = x_2(t) \tag{4.2}$$

$$\dot{x}_2(t) = -x_1(t) + [1 - x_1^2(t)]x_2(t) + u(t) \tag{4.3}$$

- with boundary conditions:

$$\mathbf{x}(0) = [0, 0]^T \tag{4.4}$$

$$\mathbf{x}(1) = [1, 10]^T$$

GPOPS automatically formats the problem to be in the form of (2.69) - (2.72) by transforming $t \in [0, 1]$ to $\tau \in [-1, 1]$ using (2.73). This is a nonlinear, continuous-time, Lagrange-type OC problem without path and control constraints, and with fixed final state and time. It can be verified that forming the HBVP as described in section 2.2.2 results in a set of coupled ODEs that cannot be solved analytically, making it well suited for direct numerical solution in GPOPS.

4.1.2 Stochastic Problem.

Uncertainty is incorporated by modifying the dynamics (4.2) as indicated by the ($\rightarrow$), with random input parameters A and B as follows:

$$x_1(t) \rightarrow x_1(t) + Ax_1(t) = (1 + A)x_1(t) \tag{4.5}$$

$$x_2(t) \rightarrow x_2(t) + Bx_2(t) = (1 + B)x_2(t)$$

resulting in the modified stochastic dynamics:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) + Bx_2(t) \\ \dot{x}_2(t) &= -(x_1(t) + Ax_1(t)) + u(t) \\ &\quad +[1 - (x_1(t) + Ax_1(t))^2](x_2(t) + Bx_2(t)) \end{aligned} \tag{4.6}$$

Inputs A and B are assumed to be independent Gaussian RVs and come from the same distribution with zero mean, $\mu = 0$, and standard deviation, σ, of 0.1. These random inputs introduce uncertainties on the states transforming $\mathbf{x}(t)$, $u(t)$, and J into RVs. The cost functional (4.1), boundary conditions (4.4), and time domain remain the same in the stochastic problem. It should be noted that the objective of a stochastic OC problem is typically to find the control signal and state trajectories that minimize the expected value of the cost functional, written as:

$$J = \mathbb{E}\left[\int_{t_0}^{t_f} \frac{1}{2}(2x_1^2(t) + x_2^2(t) + u^2(t))dt\right] \tag{4.7}$$

In this case, quantification of the effects of uncertainties on $\mathbf{x}(t)$, $u(t)$, and $J(\mathbf{x}, u)$ solutions is sought since it is assumed that there is no way to control the uncertainties or minimize their effects. So, the cost functional as written in (4.1) is used when solving each of the Q sampled deterministic problems and the set of solutions is used to construct gPC approximations as functions of the random inputs to assess the effects of those uncertainties.

Continuing with the problem set-up, the following definitions are used following the development in section 2.4.2:

- **State variables**: $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ $(J = n = 2)$
- **Stochastic input parameters (random variables)**: $\mathbf{p} = (A, B) \in \mathbb{R}^2$ $(N = 2)$.

- **Outputs** or **observables**: the observables are chosen to be the state variables, control, and cost values. $\mathbf{z} = (x_1, x_2, u, J) \in \mathbb{R}^4$ $(K = 4)$

The PDFs (Gaussian) of A and B are denoted by $\rho_1(A)$ and $\rho_2(B)$. The joint PDF is given by (2.106) as:

$$\rho(\mathbf{p}) = \prod_{i=1}^{2} \rho_i(p_i) = \rho_1(A)\rho_2(B) \tag{4.8}$$

The sample space, Ω, is $(-\infty, \infty) \times (-\infty, \infty)$, but choosing the distributions with $\mu_A = \mu_B = 0$ and $\sigma_A = \sigma_B = 0.1$ results in A and B taking values in a 10σ window, $(-1, 1) \times (-1, 1)$, with greater than 99.99% probability. Defining the distributions in this way avoids non-existent OC problem solutions resulting from state cancellation in the dynamics as A and B approach -1.

The collocation points define the finite sample space. Since Gaussian PDFs are assumed, Hermite polynomial basis functions were chosen for the gPC expansions, as suggested in [87], [88], and Table 2.3, and collocated at Gauss-Hermite quadrature points. It's important to note that the statistical version of Hermite polynomials is used where the weighting function is $exp[-\frac{(x-\mu)^2}{2\sigma^2}]$ instead of the typical $exp[-x^2]$, implying that the traditional Gauss-Hermite quadrature points should be scaled by $\sqrt{2\sigma^2}$ and shifted by adding μ and the quadrature weights scaled by $(\sqrt{\pi})^{-1}$. Recall there are q points in each random dimension calculated by finding the roots of the q^{th} Hermite polynomial, H_q, and scaled accordingly. For this sample problem, q_1 and q_2 were chosen to be 21, resulting in $Q = 441$ collocation nodes shown in Fig. 4.1. Figure 4.1 shows that the resulting finite probability space, $\Gamma = \prod_{i=1}^{2} \Gamma_i$, as given in (2.107), is the tensor grid $[-0.8, 0.8] \times [-0.8, 8]$.

The one-dimensional polynomial spaces (2.108), were chosen to be represented by

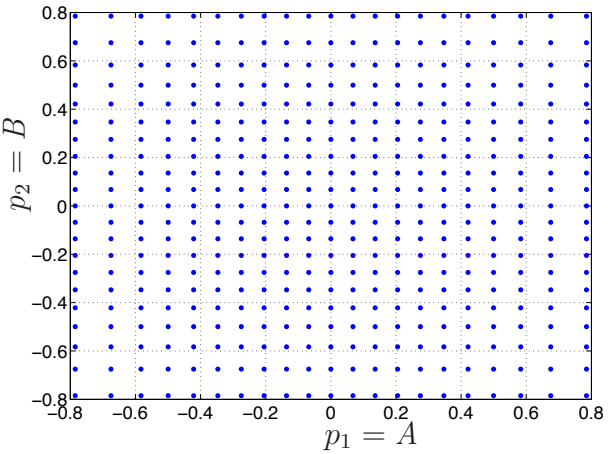


Figure 4.1. Finite domain, $\Gamma_1 \times \Gamma_2$, defined by the tensor grid of 441 collocation points used as random inputs A and B.

the following polynomial basis sets:

$$\{\phi_1\}_{d_1=0}^{5} = \{H_0, ..., H_5\} \tag{4.9}$$
$$\{\phi_2\}_{d_2=0}^{5} = \{H_0, ..., H_5\}$$

where H_i denotes the Hermite polynomial of the i^{th} order. Thus, the 2-dimensional polynomial space, defined using the tensor product in (2.110), is:

$$W_N^{P_c} \equiv \{\phi_1\}_{d_1=0}^{5} \bigotimes \{\phi_2\}_{d_2=0}^{5} \tag{4.10}$$

Selecting the entire set of basis functions yields 36 total basis elements, where each one-dimensional basis polynomial is fifth order and the order of the tensor product, P_c, is 10. The number of basis elements was chosen to provide balance between computational burden and lowest observed variances in the state approximations at the terminal time, indicating satisfaction of specified terminal boundary conditions.

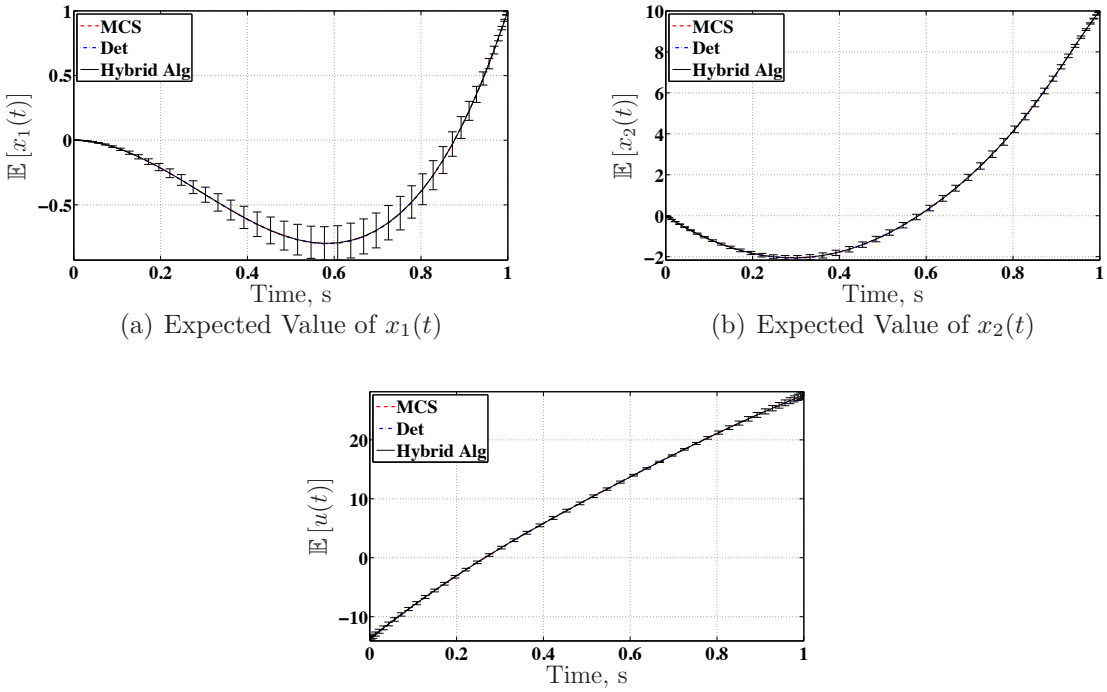
4.2 Results

Results of applying the hybrid GPM-gPC algorithm to the OC demonstration problem are presented in this section. Mean solutions and variances are shown for the state variables, $x_1(t)$, $x_2(t)$, and control, $u(t)$, as well as the covariances between these variables, and are compared to an MCS for verification. The variation of $J(\mathbf{x}(t), u(t), t; A, B)$ with respect to the random inputs is also shown.

The MCS was accomplished by generating random samples of the stochastic variables, (A, B), taken from Gaussian distributions with $\mu_A = \mu_B = 0$ and $\sigma_A = \sigma_B = 0.1$ as specified in the problem formulation. Each sample pair was subsequently inserted into the stochastic dynamics equations (4.6), creating a set of deterministic OC problems that were solved using GPOPS. The ensemble of solutions was used to calculate expected values (mean) and variances of the state variables and control and covariances between $x_1(t)$ and $u(t)$, $x_2(t)$ and $u(t)$, and $x_1(t)$ and $x_2(t)$. Convergence was checked after every 500 samples and was achieved at $16,500$ samples when the maximum change in expected values of the state trajectories from the previous interval was below an arbitrarily chosen tolerance of 5×10^{-5}. Expected value, variance, and covariance properties of the hybrid algorithm solution are compared to MCS results using the following Percent Difference (PD) equation:

$$PD = max \left| \frac{S_{MCS}(t) - S_{alg}(t)}{(S_{MCS}(t) + S_{alg}(t))/2} \right| \times 100 \tag{4.11}$$

where hybrid algorithm and MCS statistical quantities being compared, S_{alg} and S_{MCS}, are the expected values, variances, or covariances as appropriate. This comparison metric was chosen over the typical percent error because both hybrid algorithm and MCS results are approximations where fidelity can be increased, at the expense of computational burden, by increasing the numbers of sample points or by



(a) Expected Value of $x_1(t)$

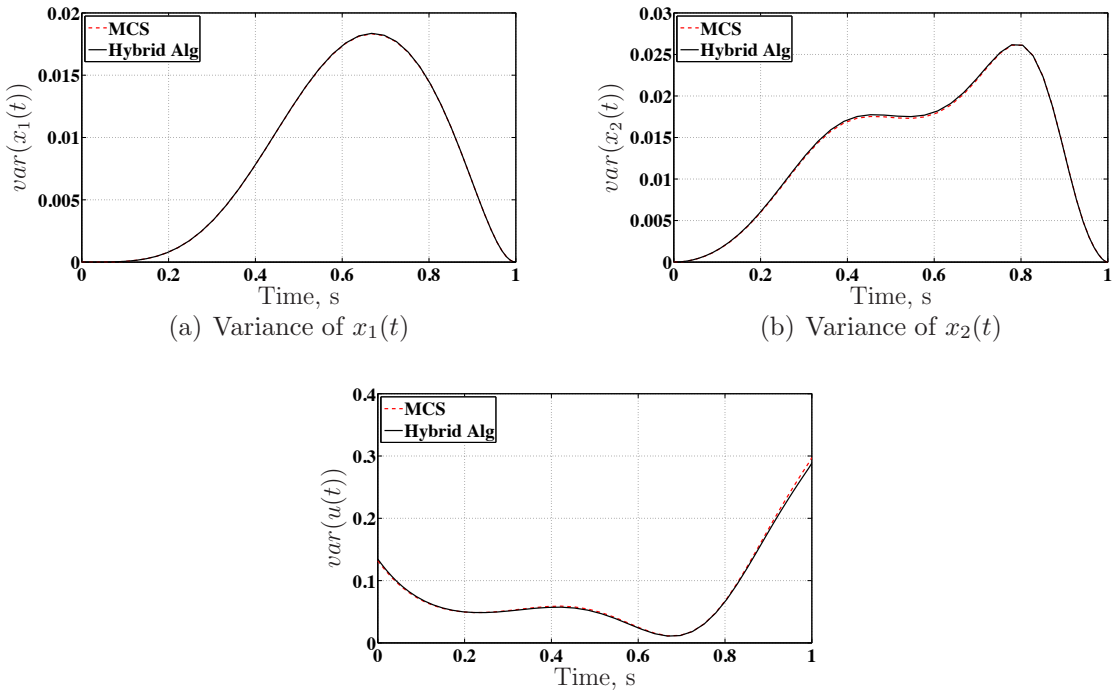
(b) Expected Value of $x_2(t)$

(c) Expected Value of $u(t)$

Figure 4.2. Hybrid algorithm, MCS, and deterministic expected value estimates of $x_1(t)$, $x_2(t)$, and $u(t)$.

increasing the polynomial order, and thus neither can be assumed to be a theoretical answer.

Figure 4.2 shows the mean solutions for $x_1(t)$, $x_2(t)$, and $u(t)$, respectively. The three curves on each of these plots show the hybrid algorithm expected value solution, the MCS expected value solution, and the deterministic solution. The hybrid algorithm expected value approximation for each state and control variable is given by the first coefficient in each expansion as stated in (3.3). Bars indicating the standard deviations are included to quantify how much the solution varies from the mean. The solution to the deterministic OC problem, given by (4.1) - (4.4), or equivalently by the modified dynamics (4.6) with $A = B = 0$, serves as another check of the mean solution. Since the stochastic inputs are unbiased and zero mean, it was expected that the hybrid algorithm mean solution would be close to the deterministic one, which is



(a) Variance of $x_1(t)$

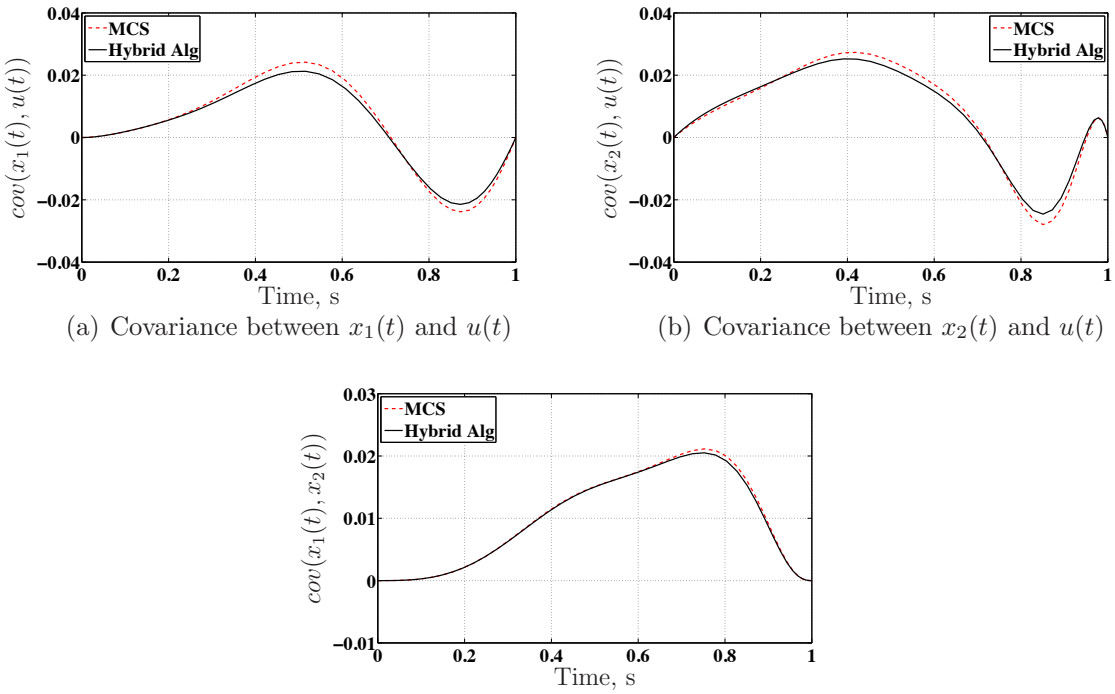
(b) Variance of $x_2(t)$

(c) Variance of $u(t)$

Figure 4.3. Hybrid algorithm and MCS variance estimates of $x_1(t)$, $x_2(t)$, and $u(t)$.

indeed the case. This check was possible by virtue of the problem formulation and may not be available in more general problems. These plots show that the hybrid algorithm and MCS expected value results closely match each other as quantified with PD calculations using (4.11). The PD in expected values of $x_1(t)$, $x_2(t)$, and $u(t)$ is 0.17%, 0.04%, and 0.02%, respectively. Both the hybrid algorithm and MCS solutions also match the deterministic solution as expected, and the desired terminal conditions $x_1(t_f)$ and $x_2(t_f)$ are satisfied.

Figure 4.3 shows the variances of the hybrid algorithm solutions of $x_1(t)$, $x_2(t)$, and $u(t)$, respectively, using (3.4). MCS results are also included for comparison. These plots show close agreement between the hybrid algorithm and MCS variance approximations with PD values of 0.17%, 1.236%, and 3.015% in variances of $x_1(t)$, $x_2(t)$, and $u(t)$, respectively. Desired terminal conditions, $x_1(t_f)$ and $x_2(t_f)$, are also satisfied indicated by zero variances at the terminal time.



(a) Covariance between $x_1(t)$ and $u(t)$

(b) Covariance between $x_2(t)$ and $u(t)$

(c) Covariance between $x_1(t)$ and $x_2(t)$

Figure 4.4. Hybrid algorithm and MCS covariance estimates between $x_1(t)$ and $u(t)$, $x_2(t)$ and $u(t)$, and $x_1(t)$ and $x_2(t)$.

Figure 4.4 shows the covariances between $x_1(t)$ and $u(t)$, $x_2(t)$ and $u(t)$, and $x_1(t)$ and $x_2(t)$, respectively, calculated using (3.5). Once again, MCS results are included for comparison. These plots further characterize the statistics of the solution by showing correlations between $x_1(t)$ and $u(t)$, $x_2(t)$ and $u(t)$, and $x_1(t)$ and $x_2(t)$. Since the dynamics in (4.6) are nonlinear and coupled in the state variables, it was expected that the output distributions would be correlated, as indicated by the non-zero covariances. There are more noticeable differences in these results where the maximum PD in covariances between $x_1(t)$ and $u(t)$ and $x_2(t)$ and $u(t)$ is approximately 14% and 4.3% in the covariance between $x_1(t)$ and $x_2(t)$. Even with more pronounced differences in the covariance estimates between the two methods, both sets of results follow the same general trend.

Closer agreement between the hybrid algorithm and MCS expected value, vari-

ance, and covariance estimates can be obtained by increasing the fidelity of the MCS. Decreasing the convergence tolerance by an order of magnitude, to 5×10^{-6}, yields between 28% and 67% improvement as can be seen in Table 4.1. The goal of running

Table 4.1. Difference between hybrid algorithm and MCS results

	16,500 Sample MCS[a]		74,500K Sample MCS[b]		
S	Max Difference ($\lvert S_{MCS} - S_{alg} \rvert$)	PD (%)	Max Difference ($\lvert S_{MCS} - S_{alg} \rvert$)	PD (%)	Change (%)
Mean					
$x_1(t)$	1.30×10^{-3}	0.1703	7.88×10^{-4}	0.1058	−37.87
$x_2(t)$	1.50×10^{-3}	0.0415	9.45×10^{-4}	0.0224	−45.93
$u(t)$	6.60×10^{-3}	0.02355	4.67×10^{-3}	0.0169	−28.36
Variance					
$x_1(t)$	3.20×10^{-5}	0.1742	4.40×10^{-5}	0.2396	37.54
$x_2(t)$	2.16×10^{-4}	1.236	1.05×10^{-4}	0.4021	−67.47
$u(t)$	8.80×10^{-3}	3.015	3.32×10^{-4}	1.142	−62.12
Covariance					
$x_1(t), u(t)$	3.0×10^{-3}	13.56	1.03×10^{-3}	5.039	−62.84
$x_2(t), u(t)$	3.6×10^{-3}	14.32	1.19×10^{-3}	5.203	−63.67
$x_1(t), x_2(t)$	7.74×10^{-4}	4.321	4.470×10^{-4}	2.396	−44.55

[a] MCS tolerance $= 5 \times 10^{-5}$, $16.5K$ samples, 137.8 minutes processing
[b] MCS tolerance $= 5 \times 10^{-6}$, $74.5K$ samples, 681.7 minutes processing

the second MCS was to gain insight into the cause of noticeable PD between the two methods, particularly in the covariance approximations. If the PD values did not change appreciably by increasing fidelity of the MCS, then it would be reasonable to conclude the parameters of the hybrid algorithm, such as number of collocation points and order of polynomial basis sets, would need to be adjusted to improve the quality of the hybrid algorithm solution. Conversely, if the correlation improved, then it could be concluded that the MCS convergence parameters could be better selected to provide better comparison. Table 4.1 shows that decreasing the MCS convergence tolerances by an order of magnitude significantly improved correlation with the hybrid algorithm, with most noticeable improvement in the covariance approximations, but

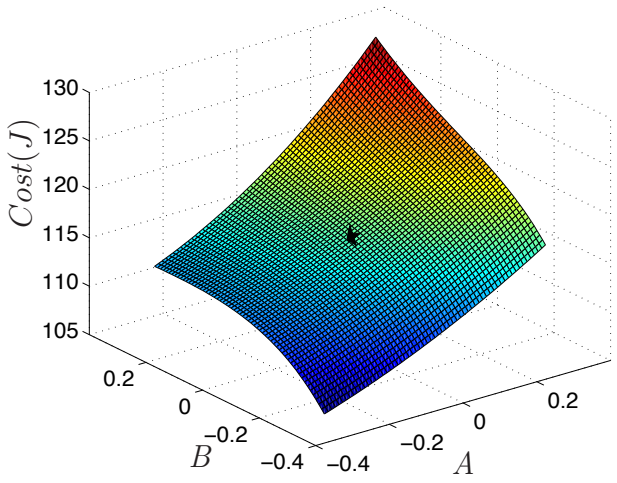


Figure 4.5. Hybrid algorithm cost approximation as function of uncertain inputs A and B.

more than quadrupled the number of random solutions required and corresponding processing time on an Apple MacBook Pro with a 2.2 gigahertz quad-core processor and 8 gigabytes of memory. Therefore, it's apparent that the hybrid algorithm's polynomial approximation using 441 collocation points, fifth-order one-dimensional polynomial bases, and 6.8 minutes of computation time yields results that are at least equivalent to, and probably more accurate than, MCS solving $74,500$ sampled OC problems.

Lastly, Figure 4.5 shows how the cost varies as a function of the random inputs. This plot was generated using a tensor product set of test points A and B ranging from -3σ to 3σ as inputs to the output approximation (3.2) of the cost. The surface plot of the gPC cost values represents a probability-like distribution function. The lowest cost is obtained when both A and B are less than zero. Conversely, the largest costs result when the random inputs are both greater than zero. The surface shows how the costs vary over the entire range. The star in the figure shows the hybrid algorithm estimation of the expected cost, which is also approximately equal to the deterministic cost. This plot is effective in visualizing the effects the uncertain parameters have

on the OC solution. Since the cost is a functional of the states and control, thus implicitly a functional of the uncertain parameters, increasing or decreasing cost resulting from realizations of the random parameters gives an indication of overall increase or decrease in magnitudes of the state and control vectors.

4.3 Summary

In this chapter, a nonlinear OC problem was considered where Gaussian random parameters were introduced into the state equations. Following the gPC description in section 2.4.2, the problem was formulated in a form suitable for application of the hybrid GPM-gPC algorithm presented in Chapter III. The output of the algorithm is a set of polynomial approximation functions for $x_1(t; A, B)$, $x_2(t; A, B)$, $u(t; A, B)$, and $J(x_1(t; A, B), x_2(t; A, B), u(t; A, B))$ whose expected values, variances, and covariances, are found using the expansion coefficients and describe the statistical properties of the solution. Comparing the statistical properties determined by application of the hybrid algorithm with the MCS results demonstrated that the hybrid algorithm effectively quantifies the effects of uncertainty with comparable accuracy while requiring dramatically fewer sample points and associated GPOPS deterministic solutions and computation time. The next chapter will apply the hybrid algorithm to another type of optimization problem, a trajectory optimization problem, that is representative of a real-world mission planning scenario, a target application for this research.

V. Trajectory Optimization Problem

A trajectory optimization problem, notionally sketched in Figure 5.1, is considered to demonstrate the ability of the hybrid GPM-gPC algorithm to quantify the effects of uncertain parameters on an optimal trajectory solution. The scenario was designed to represent an aircraft traveling to a desired target location through an environment where there are potential risks of lethal engagements. The objective was to find the optimal path in a two-dimensional space that takes a vehicle from an initial position to a target location while minimizing the probability that it will be killed by the threats whose locations are uncertain. It's assumed that at some point in time the center locations were perfectly known but have possibly moved in the time between intelligence gathering and mission planning according to an assumed probability distribution. The deterministic TO problem will be described first, which will form the basis for the subsequently discussed stochastic problem.

5.1 Problem Formulation

5.1.1 Deterministic Baseline Problem.

To begin adding details to the notional scenario in Figure 5.1, a two-dimensional grid of 50 x 50 nautical miles (NM) was chosen. Three threats with effective ring diameters of approximately 20 NM were arbitrarily placed in the grid at (10 NM, 30 NM), (25 NM, 15 NM), and (35 NM, 30 NM). The target was chosen to be located at (25 NM, 50 NM). Threat and target locations can be easily changed to accommodate user-supplied scenario specifications. Likewise, the diameters of the threat rings can be adjusted and are not constrained to be uniformly sized. Units of NM and knots (kts) were used since they are familiar to aircrews and mission planners who have interest in this type of problem.

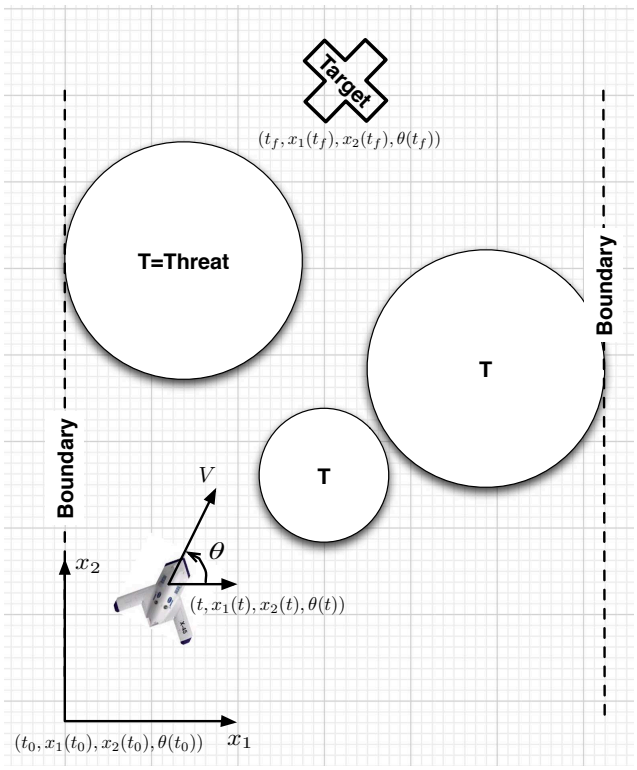


Figure 5.1. Notional sketch of the trajectory optimization problem

The vehicle dynamics are generically represented with the *Dubins* model in (5.1).

$$\begin{aligned} \dot{x}_1(t) &= V\cos(\theta(t)) \\ \dot{x}_2(t) &= V\sin(\theta(t)) \\ \dot{\theta}(t) &= u(t) \end{aligned} \tag{5.1}$$

This model was used to avoid adding the complexity of developing and coding a model for a specific vehicle's dynamics, which is not necessary to demonstrate the hybrid algorithm's performance. The state and control vectors are defined as $\mathbf{x}(t) = [x_1(t),\ x_2(t),\ \theta(t)]^T \in \mathbb{R}^3$ and $\mathbf{u}(t) = [\dot{\theta}(t)] \in \mathbb{R}^1$, where x_1 and x_2 describe the vehicle's position in the two-dimensional grid space and θ is the heading angle. The vehicle speed (V) is assumed to be a constant 470 kts, chosen based on a mission

profile of a generic large aircraft, and the control is bounded as:

$$|u(t)| \leq \frac{V}{R_{min}} \tag{5.2}$$

where the minimum turn radius (R_{min}) based on the constant speed is estimated to be 11,326 feet. The initial and terminal boundary conditions are:

$$\begin{aligned} \mathbf{x}(t_0) = &\quad \mathbf{x}(-1) = [0, 0, 1.1071]^T \\ \mathbf{x}(t_f) = &\quad \mathbf{x}(1) = [25, 50, free]^T \end{aligned} \tag{5.3}$$

where t_f is unspecified, or free. Lastly, admissible paths are restricted to lie within the boundaries of the grid, stated in the form of (2.72) as:

$$[-x_1(\tau),\ (x_1(\tau) - 50),\ -x_2(\tau),\ (x_2(\tau) - 50)]^T \leq \mathbf{0} \tag{5.4}$$

These boundaries were chosen to restrict the vehicle's allowable area of operations, thereby creating a scenario where the vehicle must navigate through the threat field rather than completely avoid it.

The most important element in setting up the TO problem is to appropriately choose a cost functional in the form of (2.69) that leads to the trajectory that allows the aircraft to reach the target while minimizing the vehicle's exposure to lethal threats and not prohibiting it from passing through higher threat areas of the space. The integrand of the Lagrange term in (2.69) must be chosen to define the areas of the space shown in Figure 5.1 that pose the greatest risk to the vehicle. In an obstacle avoidance problem, the threat shapes would be treated as path constraints, as investigated by Gong et al. [44] and Lewis et al. [59], making any trajectory that passes through those areas of the space inadmissible. Numerical solvers would only

return solutions that totally avoid those areas of the space or output a message stating that no optimal solution exists that satisfies these constraints. However, in this case, the threats are treated as regions of the space that represent varying probabilities that the vehicle will be killed, not as hard obstacles that should be totally avoided. The threat rings represent likelihood, or probabilities, that traveling into those areas would result in the vehicle being eliminated. Random elements, discussed later in this section, are introduced through uncertainty of the exact center locations of the threat rings, making it possible for the threat rings to overlap and impossible to completely avoid the threats. Therefore, the integrand of the cost functional, $g(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau; t_0, t_f)$, should be something that describes the probabilities of kill (POK) in the space, assuming that the highest probabilities will be near the center of the threat rings and decrease with distance away from the center. Minimizing such a functional will allow the vehicle to pass through regions of the space occupied by the threat rings while finding the least threatening path to the target. A bi-variate Gaussian PDF for each threat was used to describe the high-threat regions of the two dimensional space, written as:

$$g_i(\cdot) = \frac{1}{2\pi\sigma_{i,x_1}\sigma_{i,x_2}} exp\left(-\frac{1}{2}\left[\frac{(x_1(\tau)-\mu_{i,x_1})^2}{\sigma_{i,x_1}^2} + \frac{(x_2(\tau)-\mu_{i,x_2})^2}{\sigma_{i,x_2}^2}\right]\right) \tag{5.5}$$

where the mean values, μ_{i,x_1} and μ_{i,x_2}, and the standard deviations, σ_{i,x_1} and σ_{i,x_2}, define the center and variation in lethality of the i^{th} threat in the x_1 and x_2 directions for $i = 1, \ldots, 3$. Figure 5.2 depicts the high POK regions using the previously stated center locations and $\sigma_{i,x_1} = \sigma_{i,x_2} = 5$ NM to create rings with 20 NM effective diameters.

One final consideration is necessary in defining the cost functional. Looking at Figure 5.2, it is obvious that multiple trajectories may exist that have equal minimum POK, meaning that a unique solution may not exist, resulting in GPOPS failure to

return a solution. To minimize this possibility, the Mayer term in the cost functional should be chosen to drive the software to return the minimum POK path with shortest travel time to the target. Including the Mayer term in the cost functional gives:

$$J(\cdot) = \mathcal{W}t_f + \frac{t_f - t_0}{2} \int_{-1}^{1} \sum_{i=1}^{3} g_i(x_1(\tau), x_2(\tau), \tau; t_0, t_f) d\tau \tag{5.6}$$

where the weighting factor, $\mathcal{W}$, on t_f is used to make sure the Mayer and Lagrange terms are on the same order of magnitude to avoid having the minimum time requirement dominate the solution. For this problem, $\mathcal{W} = 10^{-5}$ was used.

Finally, the TO problem can be stated as: Minimize J (5.6) subject to the dynamics (5.1), control constraint (5.2), boundary conditions (5.3), and path constraints (5.4). This is a continuous-time Bolza-type TO problem that is nonlinear in both the cost functional and the state dynamics, with fixed final state and free final time, and constrained boundary conditions and control input. The solution to the problem described thus far will be referred to as the *deterministic* solution, i.e. the noise-free solution, which was found using GPOPS and is shown in Figure 5.2. The mean solution of the stochastic variant discussed in the next section will be compared to this solution and should closely match it since the uncertainties will be assumed to be zero mean.

5.1.2 Stochastic Problem.

The stochastic problem is formulated in this section by modifying the deterministic version in the previous section and then following the development in section 2.4.2. Uncertainty is added to the deterministic problem by introducing three random input parameters, A_1, A_2, and A_3, and making the following substitutions indicated by the

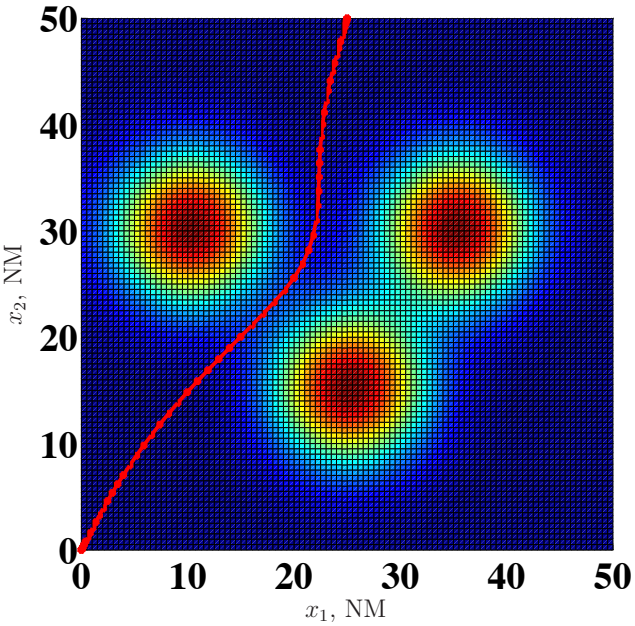


Figure 5.2. Threat ring locations and deterministic solution to the trajectory optimization problem

($\rightarrow$) into the Lagrange term, (5.5), of the cost functional, (5.6).

$$\begin{aligned} \mu_{i,x_1} &\rightarrow \mu_{i,x_1} + A_i \\ \mu_{i,x_2} &\rightarrow \mu_{i,x_2} + A_i \end{aligned}$$

The integrand of the modified cost functional becomes:

$$g_i(\cdot) = \frac{1}{2\pi\sigma_{i,x_1}\sigma_{i,x_2}} exp\left(-\frac{1}{2}\left[\frac{(x_1(\tau) - (\mu_{i,x_1} + A_i))^2}{\sigma_{i,x_1}^2} + \frac{(x_2(\tau) - (\mu_{i,x_2} + A_i))^2}{\sigma_{i,x_2}^2}\right]\right) \tag{5.7}$$

resulting in the modified cost functional:

$$J(\cdot) = \mathcal{W}t_f + \frac{t_f - t_0}{2}\int_{-1}^{1}\sum_{i=1}^{3} g_i(x_1(\tau), x_2(\tau), \tau; A_i, t_0, t_f)d\tau \tag{5.8}$$

where the A_i's are assumed to be independent RVs whose distributions will be discussed later in this section. The random inputs thus defined and written into (5.7)

and (5.8) introduce uncertainties on center locations of the threats, transforming J, $\mathbf{x}$, u, and t_f into RVs whose solutions will have statistical properties.

As in Chapter IV, the cost functional is not written with an expectation operator since quantification of the effects of uncertain parameters on $J(\mathbf{x})$, $\mathbf{x}(t)$, $u(t)$, and t_f solutions is sought and there is no way to control the uncertainties or minimize their effects. So, the cost functional as expressed by (5.7) and (5.8) is used when solving each of the Q sampled deterministic problems, and the set of solutions is used to construct gPC approximations of the states, control, cost, and final time as functions of the random inputs to assess the effects of those uncertainties. Additionally, the dynamics (5.1), control bounds (5.2), boundary conditions (5.3), and path constraints (5.4) remain the same with the addition of uncertainty.

Following the development in section 2.4.2, the following definitions are made to identify the key details needed to apply the hybrid algorithm:

- **State variables**: $\mathbf{x} = (x_1, x_2, \theta) \in \mathbb{R}^3$ $(J = n = 3)$
- **Stochastic input parameters (random variables)**: $\mathbf{p} = (A_1, A_2, A_3) \in \mathbb{R}^3$ $(N = 3)$.
- **Outputs** or **observables**: $\mathbf{z} = (x_1, x_2, \theta, u, t_f, J) \in \mathbb{R}^6$ $(K = 6)$

The PDFs of A_1, A_2, and A_3 are denoted by $\rho_1(A_1)$, $\rho_2(A_2)$, and $\rho_3(A_3)$, respectively. The joint PDF is given by (2.106) as:

$$\rho(\mathbf{p}) = \prod_{i=1}^{3} \rho_i(p_i) = \rho_1(A_1)\rho_2(A_2)\rho_3(A_3) \tag{5.9}$$

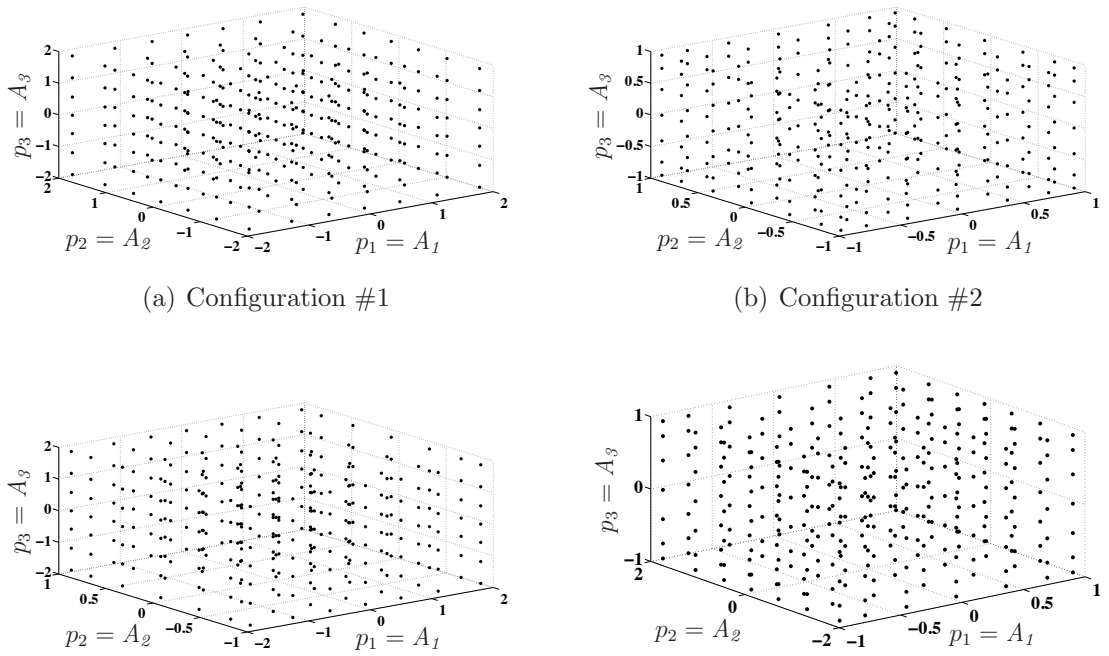
In this work, four combinations of the PDFs of A_1, A_2, and A_3 are considered as shown in Table 5.1, noting that the notation for Gaussian (normal) distributions is $\mathcal{N}(\mu, \sigma)$, and for uniform distributions is $\mathcal{U}(a, b)$.

Table 5.1. Combinations of PDFs $\rho_1(A_1)$, $\rho_2(A_2)$, and $\rho_3(A_3)$ and resulting finite sample space, Γ

Combination	$\rho_1(A_1)$	$\rho_2(A_2)$	$\rho_3(A_3)$	Γ
1	$\mathcal{N}(0, 0.5)$	$\mathcal{N}(0, 0.5)$	$\mathcal{N}(0, 0.5)$	$[-2, 2] \times [-2, 2] \times [-2, 2]$
2	$\mathcal{U}(-1, 1)$	$\mathcal{U}(-1, 1)$	$\mathcal{U}(-1, 1)$	$[-1, 1] \times [-1, 1] \times [-1, 1]$
3	$\mathcal{N}(0, 0.5)$	$\mathcal{U}(-1, 1)$	$\mathcal{N}(0, 0.5)$	$[-2, 2] \times [-1, 1] \times [-2, 2]$
4	$\mathcal{U}(-1, 1)$	$\mathcal{N}(0, 0.5)$	$\mathcal{U}(-1, 1)$	$[-1, 1] \times [-2, 2] \times [-1, 1]$

Hermite polynomial bases collocated at Gauss-Hermite quadrature points were used for gPC expansions associated with Gaussian distributions and Legendre polynomial bases collocated at Gauss-Legendre points were used for gPC expansions associated with uniform distributions as suggested in [87] and [88]. As discussed in Chapter IV, the statistical version of the Hermite polynomials was used where Gauss-Hermite quadrature points were scaled by $\sqrt{2\sigma^2}$ and shifted by μ and the quadrature weights scaled by $(\sqrt{\pi})^{-1}$. Additionally, the Gauss-Legendre quadrature weights were scaled such that the weights in the set sum to one so that the Legendre polynomials appropriately approximate uniform probability distributions. Recall there are q points in each random dimension, which are calculated by finding the roots of either the q^{th} Hermite polynomial, H_q, or the q^{th} Legendre polynomial, L_q, as dictated by the assumed PDF of the uncertain parameter, and scaled accordingly. The number of collocation points in the three random dimensions, q_1, q_2, and q_3, was chosen to be 7 to provide a balance between computation time and accuracy of the solution when compared to MCS, resulting in $Q = 343$ collocation nodes with the finite probability domains shown in Figure 5.3 and listed in Table 5.1.

The one-dimensional polynomial spaces were chosen, as given in (2.108), to be defined by the following Hermite (H) and Legendre (L) polynomial basis sets as shown in Table 5.2 for the four cases considered. In Table 5.2, the subscripts on H and L indicate the order of the polynomial basis elements. Thus, the 3-dimensional



(a) Configuration #1

(b) Configuration #2

(c) Configuration #3

(d) Configuration #4

Figure 5.3. Collocation points: Tensor grid of 343 points (7 in each dimension) used as random inputs A_1, A_2, and A_3.

polynomial space, defined using the tensor product in (2.110), is:

$$W_N^{P_c} \equiv \{\phi_1\}_{d_1=0}^{7} \bigotimes \{\phi_2\}_{d_2=0}^{7} \bigotimes \{\phi_3\}_{d_3=0}^{7} \tag{5.10}$$

Selecting the entire set of basis functions, $P_c = 21$ in (2.111), yields 512 total basis elements, where each one-dimensional basis set is at most seventh order and the highest order of the tensor product set is 21. The number of basis elements was chosen, all of which are used in computations, to provide a balance between computational burden and accuracy of the solution when compared to MCS.

This section has presented all of the details necessary to implement the hybrid GPM-gPC algorithm steps described in Chapter III. The results of applying the algorithm to the sample problem considered in this research are presented in the next section.

Table 5.2. One-Dimensional polynomial basis sets

Combination	Polynomial Basis Set		
	ϕ_1	ϕ_2	ϕ_3
1	$\{H_0, \ldots, H_7\}$	$\{H_0, \ldots, H_7\}$	$\{H_0, \ldots, H_7\}$
2	$\{L_0, \ldots, L_7\}$	$\{L_0, \ldots, L_7\}$	$\{L_0, \ldots, L_7\}$
3	$\{H_0, \ldots, H_7\}$	$\{L_0, \ldots, L_7\}$	$\{H_0, \ldots, H_7\}$
4	$\{L_0, \ldots, L_7\}$	$\{H_0, \ldots, H_7\}$	$\{L_0, \ldots, L_7\}$

5.2 Results

Results of applying the hybrid GPM-gPC algorithm to the TO concept demonstration problem are presented in this section. The expected value solutions and variances are plotted for $x_1(\tau)$, $x_2(\tau)$, $\theta(\tau)$, and $u(\tau)$ and tabulated for J and t_f for each of the configurations listed in Table 5.1. Covariances between the states $x_1(\tau)$ and $x_2(\tau)$, $x_1(\tau)$ and $\theta(\tau)$, and $x_2(\tau)$ and $\theta(\tau)$, as well as between the states and control, $x_1(\tau)$ and $u(\tau)$, $x_2(\tau)$ and $u(\tau)$, and $\theta(\tau)$ and $u(\tau)$, are also shown. The expected value, variance, and covariance results determined by the hybrid algorithm are compared to the MCS results for verification. Lastly, the hybrid algorithm solution statistics are used to estimate the probabilities that the vehicle will be killed during the course of the mission. Since this is a free final time problem, expected values, variances, and covariances are plotted versus the time vector τ. Stochastic computations evaluating the hybrid algorithm output function (3.2), expansion coefficients (3.1), and statistical quantities derived from the expansion coefficients are not possible in the time domain since the time points in the Q sampled solutions are not the same due to the GPOPS transformation from the τ domain to the time domain using (2.73). The τ vectors, however, remain consistent throughout regardless of the differing terminal times and allows for stochastic computations using a consistent time reference. In a fixed final time problem, these stochastic computations can be performed using the time domain.

As in Chapter IV, the MCS was accomplished by generating random samples of the stochastic variables, (A_1, A_2, A_3), taken from either Gaussian or uniform distributions as appropriate in accordance with the four cases specified in Table 5.1. Each sample was subsequently inserted into the Lagrange term (5.7) of the stochastic cost functional (5.8), creating a set of deterministic TO problems that were solved using GPOPS. The ensemble of solutions was used to calculate the MCS estimations of expected values (means), variances, and covariances. Convergence was checked after every 500 samples and was achieved when the maximum change in expected values of the state trajectories from the previous interval was below an arbitrarily chosen tolerance of 1×10^{-4}. Expected value, variance, and covariance properties of the hybrid algorithm solutions are compared to the MCS results using the the PD equation (4.11) introduced in Chapter IV and are presented in Table 5.3 discussed later in this section.

Expected value, or mean, approximations indicate the most likely solution to the TO problem. Figures 5.5 - 5.8 show the mean solutions of $x_1(\tau)$, $x_2(\tau)$, $\theta(\tau)$, and $u(\tau)$ for each of the four cases listed in Table 5.1. The three curves on each of these plots show the hybrid algorithm expected value solutions, the MCS expected value solutions, and the deterministic solution. The hybrid algorithm expected value approximation for each state and control variable is given by the first coefficient in each expansion as stated in (3.3). Bars indicating two standard deviations are included to quantify how much the solution varies from the mean. The solution to the deterministic TO problem defined by (5.1) - (5.6), or equivalently by (5.1) - (5.4) and the stochastic cost functional (5.8) with $A_i = 0$, serves as another check of the mean solution. Since the Gaussian stochastic inputs are unbiased and zero mean and the uniform stochastic inputs are symmetric about zero, it was expected that the hybrid algorithm mean solutions would be close to the deterministic ones as

shown in the expected value figures. This check was possible by virtue of the problem formulation and may not be available in more general problems. These plots show that the hybrid algorithm and MCS expected value results closely match each other as quantified with PD calculations using (4.11) and shown in Table 5.3. The PDs in expected values of $x_1(\tau)$, $x_2(\tau)$, $\theta(\tau)$, and $u(\tau)$ are generally on the order of 10^{-2} and at most around one percent in the $u(\tau)$ approximation in cases #3 and #4. Both the hybrid algorithm and the MCS solutions also match the deterministic solution as expected, and the desired terminal conditions $x_1(t_f)$ and $x_2(t_f)$ are satisfied.

Variance approximations to the stochastic problem indicate how much the solutions to the TO problem could deviate from the most likely solutions depending on the values that the uncertain parameters can assume. Figures 5.9 - 5.12 show the variances of the hybrid algorithm solutions of $x_1(\tau)$, $x_2(\tau)$, $\theta(\tau)$, and $u(\tau)$ for each of the four cases listed in Table 5.1, calculated using (3.4). MCS results are again included for comparison. These plots show close agreement between the hybrid algorithm and MCS variance approximations of $x_1(\tau)$, $x_2(\tau)$, $\theta(\tau)$, and $u(\tau)$ with PD values, as shown in Table 5.3, ranging from 0.901% to 4.55% and predominantly in the one to two percent range. Desired terminal conditions, $x_1(t_f)$ and $x_2(t_f)$ are also satisfied as indicated by zero variances at the terminal time.

Covariance estimations further characterize the distributions of the stochastic state and control solutions by providing indications of correlations between $x_1(\tau)$ and $x_2(\tau)$, $x_1(\tau)$ and $\theta(\tau)$, $x_2(\tau)$ and $\theta(\tau)$, $x_1(\tau)$ and $u(\tau)$, $x_2(\tau)$ and $u(\tau)$, and $\theta(\tau)$ and $u(\tau)$. Figures 5.13 - 5.16 show these covariance estimates approximated by the hybrid algorithm, calculated using (3.5), and compares them with MCS estimates for each of the four cases listed in Table 5.1. The covariance plots show that the state variables and control outputs are dependent even though the random inputs are assumed to be independent, which is as expected since the state trajectories and

uncertain parameters are related through the nonlinear cost functional (5.8). As with the expected value and variance estimates, these figures show agreement between the hybrid algorithm and the MCS results, which are confirmed quantitatively in Table 5.3, where the PD values are between 0.7 and 4.6 percent. Note that the table also shows two suspiciously high PD comparisons: 23.92% in the covariance between $x_1(\tau)$ and $\theta(\tau)$ in case #1 (Table 5.3(a)) and 54.16% in the covariance between $\theta(\tau)$ and $u(\tau)$ in case #4 (Table 5.3(d)). However, these apparently large differences between hybrid algorithm and MCS approximations are not evident in Figures 5.13(b) and 5.16(f). These exceptions are due to limitations of using (4.11) in cases where the difference between the quantities being compared (numerator) is very small but the values themselves are far enough apart such that the average (denominator) results in an unexpectedly high PD, even though significant differences are not obvious on the graphs. Therefore, it was concluded that these anomalies do not demonstrate appreciable disagreement between the hybrid algorithm and MCS and are a limitation of the PD metric.

Expected value and variance results for estimating the cost, J, and final time, t_f, are given in Table 5.4 and Table 5.5, respectively. The hybrid algorithm and MCS approximations for the expected values and variances in both variables closely match each other as shown by the small PDs given in the tables and are actually equal, to four decimal places, in expected value of the final time. Furthermore, the expected values of both J and t_f agree with the deterministic solution.

The expected value plots (Figures 5.5 - 5.8), along with the variance plots (Figures 5.9 - 5.12) and covariance plots (Figures 5.13 - 5.16) give the most likely solution to the stochastic trajectory optimization problem and describe the nature of the output distributions by quantifying the affects of the random parameters on the output solutions. Combined with the information in Table 5.3, these results show

that the hybrid algorithm provides solutions to the stochastic problems that are at least equivalent to the widely-accepted MCS method while significantly reducing the computational workload. Having a mix of probability distributions and thus a mix of polynomial basis sets also does not appear to have any effect on the ability of the hybrid algorithm to produce accurate solution approximations that correlate well with MCS approximations. Notes included in Table 5.3 show that the hybrid algorithm is able to produce equivalent approximations using seventh-order one-dimensional polynomial bases and 343 sampled deterministic solutions requiring five to six minutes of processing on an Apple MacBook Pro computer with a 2.2 gigahertz quad-core processor and 8 gigabytes of memory, whereas the MCS required between 17.5K and 34K sampled deterministic solutions and up to 6.5 hours of processing.

The most valuable product to mission planners and aircrews is not the set of statistics describing expected values, variances, and covariances and how well the hybrid algorithm results compare with the MCS results, but rather an estimation of the probability that they will suffer from lethal engagements with the threats while executing the mission. The discussion that follows leads to an estimate of the POK for the mission scenario by re-packaging the aforementioned results for case #1 where the uncertainties related to the center locations of the three threats are assumed to have zero mean Gaussian distributions with 0.5 NM standard deviations. The analysis that follows could be repeated for the other three cases, but does not add significantly new or noteworthy results and is thus omitted.

To do the evaluation, it was assumed no new information is available on the threat locations. Therefore, the best guess at a mission plan is to fly the mean trajectory shown in Figure 5.5, noting that if updated information were to become available, the hybrid output approximation function (3.2) could be evaluated to provide the crews with a new navigation solution with minimum POK. To assess minimum and

maximum expected POKs, the analysis was conducted as if the threats have moved in *best* and *worst* case scenarios while assuming the mean trajectory is followed. The *best case* was modeled by assuming the threats have moved away from the mean trajectory by the extreme values shown in Figure 5.3 by letting $A_1 = -2$, $A_2 = 2$, and $A_3 = 2$, resulting in the threat rings defined by the POK PDF, the integrand in (5.8), shown in Figure 5.17(a), effectively reducing potential exposure to the threats when compared to the nominal threat configuration. Conversely, the worst case was modeled by assuming that the threats have moved closer to the mean trajectory by the extreme values by letting $A_1 = 2$, $A_2 = -2$, and $A_3 = -2$, resulting in the threat rings shown in Figure 5.17(b), thus increasing the potential of fatal engagements. With these choices, the POK PDFs given in (5.7) can be written for the best case as:

$$\begin{aligned} g_1(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-8)^2}{25} + \tfrac{(x_2(\tau)-28)^2}{25}\right]\right) \\ g_2(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-27)^2}{25} + \tfrac{(x_2(\tau)-17)^2}{25}\right]\right) \\ g_3(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-37)^2}{25} + \tfrac{(x_2(\tau)-32)^2}{25}\right]\right) \end{aligned} \tag{5.11}$$

and for the worst case as:

$$\begin{aligned} g_1(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-12)^2}{25} + \tfrac{(x_2(\tau)-32)^2}{25}\right]\right) \\ g_2(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-23)^2}{25} + \tfrac{(x_2(\tau)-13)^2}{25}\right]\right) \\ g_3(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-33)^2}{25} + \tfrac{(x_2(\tau)-27)^2}{25}\right]\right) \end{aligned} \tag{5.12}$$

recalling that the full POK PDF is given by:

$$g(x_1(\tau), x_2(\tau), \tau) = \sum_{i=1}^{3} g_i(x_1(\tau), x_2(\tau), \tau) \tag{5.13}$$

The POK as a function of τ can be determined by integrating (5.13) over some

region of the two-dimensional space surrounding the mean trajectory at each time point. The region of the space was determined, based on the expected value trajectory shown in Figure 5.5 and variance results shown in Figure 5.9, by selecting the upper and lower bound trajectories, $\mathbf{x}_{ub}$ and $\mathbf{x}_{lb}$, as:

$$\begin{aligned} \mathbf{x}_{ub} = & \left(\mathbb{E}\left[x_1(\tau)\right] + 10\sqrt{var(x_1(\tau))}, \mathbb{E}\left[x_2(\tau)\right] + 10\sqrt{var(x_1(\tau))}\right) \\ \mathbf{x}_{lb} = & \left(\mathbb{E}\left[x_1(\tau)\right] - 10\sqrt{var(x_1(\tau))}, \mathbb{E}\left[x_2(\tau)\right] - 10\sqrt{var(x_1(\tau))}\right) \end{aligned} \tag{5.14}$$

This region is shown in Figure 5.17 and was arbitrarily chosen to provide conservative POK estimates to aircrews and mission planners while allowing for some flexibility in flight path. In this figure, notice that the trajectory windows are similar for the best and worst case scenarios and very small early in the mission profile where there is minimal threat to the vehicle necessitating little deviation from the minimum time trajectory. Thus, the best and worst case paths remain close to the expected value. As the vehicle moves into the center of the threat field, the threats pose greater risks to the vehicle resulting in greater variance in the path that should be flown to mitigate the risks. These observations are further evident in the POK calculation along the path.

The POK as a function of τ was determined by constructing the Cumulative Distribution Function (CDF) of the POK PDF given by:

$$G(\tau_i) = \int_{x_{2,i}^{lb}}^{x_{2,i}^{ub}} \int_{x_{1,i}^{lb}}^{x_{1,i}^{ub}} \sum_{i=1}^{3} g_i(x_1(\tau), x_2(\tau), \tau) dx_1 dx_2 \tag{5.15}$$

where the counting index, i, refers to the i^{th} τ point and the limits of integration at

each τ_i point are:

$$\begin{aligned} x_{1,i}^{lb} &= \mathbb{E}\left[x_1(\tau_i)\right] - 10\sqrt{var(x_1(\tau_i))} \\ x_{1,i}^{ub} &= \mathbb{E}\left[x_1(\tau_i)\right] + 10\sqrt{var(x_1(\tau_i))} \\ x_{2,i}^{lb} &= \mathbb{E}\left[x_2(\tau_i)\right] - 10\sqrt{var(x_2(\tau_i))} \\ x_{2,i}^{ub} &= \mathbb{E}\left[x_2(\tau_i)\right] + 10\sqrt{var(x_2(\tau_i))} \end{aligned} \tag{5.16}$$

The results are shown in Figure 5.18. This figure shows that early in the mission, τ less than -0.6, the POK is essentially zero, then increases to a maximum of approximately 0.5% as the vehicle approaches the closest threat, with both scenarios resulting in similar threat levels. As the vehicle gets into the higher risk regions of the space, τ greater than 0.2, the best case and worst case POK estimates diverge where the best case shows a maximum POK of approximately 0.25% and the worst case peaks at about 1%. This information provides aircrews with estimates of how vulnerable they are to the threats at each point in time as they progress through the mission, although they would also be interested in knowing an overall mission POK estimate.

The mission POK was found by integrating the POK CDF (5.15) over the time domain for both best and worst cases, written as:

$$POK_{mission} = \int_{-1}^{1} G(\tau) d\tau \tag{5.17}$$

The integration was performed by applying Gaussian quadrature to the data presented in Figure 5.18 resulting in cumulative POK estimates shown in Figure 5.19. The anticipated mission POKs are read from the figure at the end time and range from 0.25% to 0.5%. This information, combined with the information contained in Figure 5.18, tells planners and crews what their expected overall mission POK is along with identifying the points during the mission where they are most vulnerable.

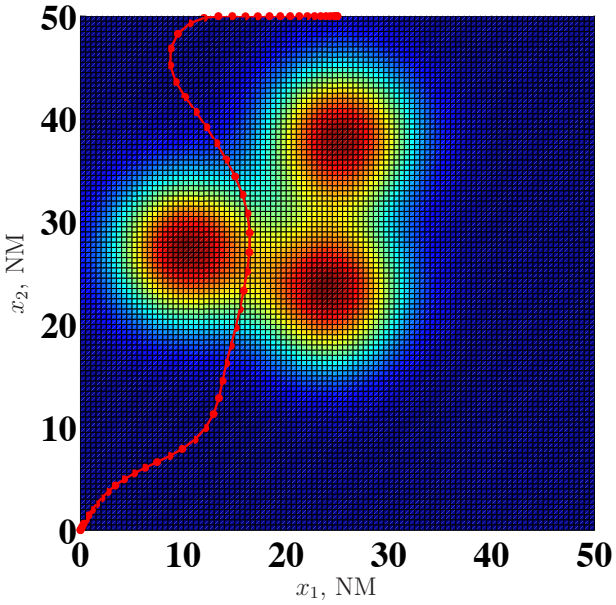


Figure 5.4. Threat ring locations and hybrid algorithm mean solution to the modified trajectory optimization problem used in POK analysis.

The POK evaluation was repeated for a more challenging threat layout to further demonstrate the ability of the hybrid algorithm to produce POK estimates that are meaningful to operators and planners. The locations of the threats were modified to yield greater interactions than was shown in Figure 5.17. The modified scenario was constructed by placing the threats at (10 NM, 27 NM), (24 NM, 23 NM), and (25 NM, 38 NM) as shown in Figure 5.4. The hybrid algorithm was applied to this scenario to estimate the expected value trajectory, also shown in Figure 5.4, and the variance information needed in the POK analysis. Comparison with an MCS is not presented but it's interesting to note that the hybrid algorithm generated solutions to this modified scenario in 26 minutes, whereas the MCS did not converge after 23 hours of processing generating an ensemble of 50K sample solutions.

To begin the POK analysis, the *best case* was once again modeled by assuming the threats have moved away from the mean trajectory by letting $A_1 = -2$, $A_2 = 2$, and $A_3 = 2$, resulting in the threat rings shown in Figure 5.20(a). The *worst case*

was also modeled as before by assuming that the threats have moved closer to the mean trajectory by letting $A_1 = 2$, $A_2 = -2$, and $A_3 = -2$, resulting in the threat rings shown in Figure 5.20(b). The modified POK PDFs, given by (5.7) for the best case are:

$$\begin{aligned} g_1(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-8)^2}{25} + \tfrac{(x_2(\tau)-25)^2}{25}\right]\right) \\ g_2(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-26)^2}{25} + \tfrac{(x_2(\tau)-25)^2}{25}\right]\right) \\ g_3(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-27)^2}{25} + \tfrac{(x_2(\tau)-40)^2}{25}\right]\right) \end{aligned} \tag{5.18}$$

and for the worst case are:

$$\begin{aligned} g_1(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-12)^2}{25} + \tfrac{(x_2(\tau)-29)^2}{25}\right]\right) \\ g_2(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-22)^2}{25} + \tfrac{(x_2(\tau)-21)^2}{25}\right]\right) \\ g_3(x_1(\tau), x_2(\tau), \tau) = &\quad \tfrac{1}{50\pi} exp\left(-\tfrac{1}{2}\left[\tfrac{(x_1(\tau)-23)^2}{25} + \tfrac{(x_2(\tau)-36)^2}{25}\right]\right) \end{aligned} \tag{5.19}$$

recalling that the full POK PDF is the sum of the three contributions given by (5.13).

The POK as a function of τ was determined by integrating (5.13) over the envelope surrounding the mean trajectory defined by:

$$\begin{aligned} \mathbf{x}_{ub} = &\quad \left(\mathbb{E}\left[x_1(\tau)\right] + 4\sqrt{var(x_1(\tau))}, \mathbb{E}\left[x_2(\tau)\right] + 4\sqrt{var(x_1(\tau))}\right) \\ \mathbf{x}_{lb} = &\quad \left(\mathbb{E}\left[x_1(\tau)\right] - 4\sqrt{var(x_1(\tau))}, \mathbb{E}\left[x_2(\tau)\right] - 4\sqrt{var(x_1(\tau))}\right) \end{aligned} \tag{5.20}$$

This region is shown in Figure 5.20 and was arbitrarily chosen to provide conservative POK estimates while allowing some flexibility in flight path. The results of the spatial integration are shown in Figure 5.21. This figure shows that early in the mission ($\tau < -0.6$) the POK is essentially zero, increases to a maximum of approximately 8% in the best case and 13% in the worst case as the vehicle traverses the high threat

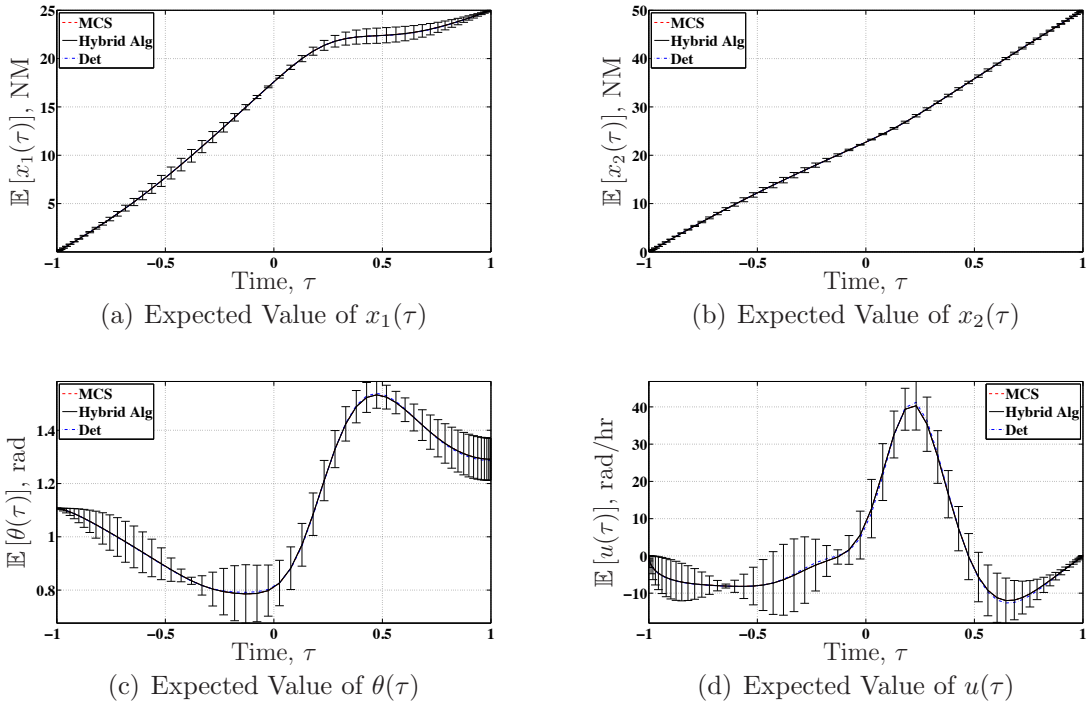
region of the space ($-0.6 < \tau < 0.4$), then becomes essentially zero again ($\tau > 0.4$) for the last part of the mission. This information provides aircrews with estimates of how vulnerable they are at each point in time during the mission and can be used to provide an overall mission POK estimate.

The mission POK estimate for the modified scenario was found by integrating the POK CDF (5.15) over the time domain for both best and worst cases using Gaussian quadrature applied to the data presented in Figure 5.21. The results are shown in Figure 5.22, which displays the mission POK as a cumulative total at each time point. The POK estimate covering the duration of the mission, which is read from the figure at the final time point, ranges between 3.3% and 6.8%. The combination of the information shown in Figures 5.21 and 5.22 tells mission planners and aircrews their probability of being killed based on their position along the planned trajectory and the total probability of being killed at each point in time.

5.3 Summary

The concept demonstration problem considered in this chapter was designed to be representative of a real-world mission scenario that is of interest to USSTRATCOM. It is a nonlinear TO problem designed to find the path through a two-dimensional space that minimizes the probability a vehicle will be killed by lethal threats whose locations are uncertain, to quantify the effects those uncertainties have on the solution by estimating the statistical properties, and to use the statistical properties to estimate the probability of the vehicle being killed during the mission. Uncertainties in the locations of the threats were modeled as either Gaussian or uniform random variables that entered the problem in the the cost functional. Following the gPC description in section 2.4.2, the problem was formulated in a form suitable for application of the hybrid GPM-gPC algorithm presented in Chapter III. The output

of the algorithm provided a set of polynomial approximation functions for the state variables, $\mathbf{x}(\tau; A_1, A_2, A_3)$, control, $u(\tau; A_1, A_2, A_3)$, cost, $J(A_1, A_2, A_3)$, and terminal time, $t_f(A_1, A_2, A_3)$, whose expected values, variances, and covariances, were found using the expansion coefficients and describe the statistical properties of the solution. Comparing the statistical properties determined by application of the hybrid algorithm with the MCS results demonstrated that the hybrid algorithm effectively quantifies the effects of uncertainty with comparable accuracy while requiring dramatically fewer sample points and associated GPOPS deterministic solutions and computation time. Probability of kill computations demonstrated that the hybrid algorithm results can be used to assess risks associated with a trajectory solution in a way that is meaningful to mission planners and aircrews.



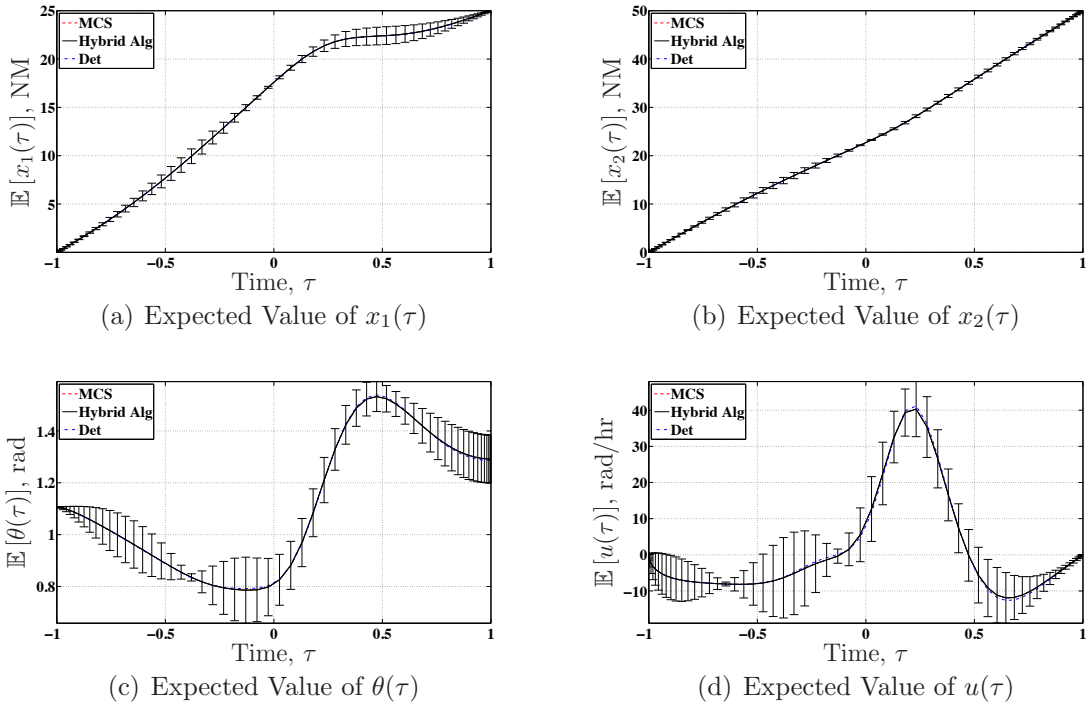
(a) Expected Value of $x_1(\tau)$

(b) Expected Value of $x_2(\tau)$

(c) Expected Value of $\theta(\tau)$

(d) Expected Value of $u(\tau)$

Figure 5.5. MCS, deterministic, and hybrid algorithm expected value estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #1.



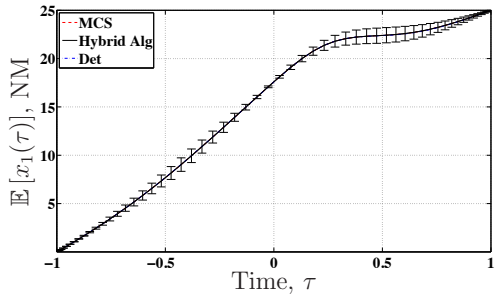
(a) Expected Value of $x_1(\tau)$

(b) Expected Value of $x_2(\tau)$

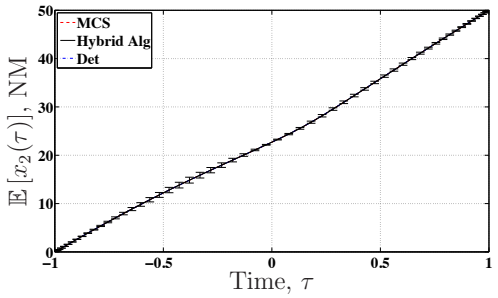
(c) Expected Value of $\theta(\tau)$

(d) Expected Value of $u(\tau)$

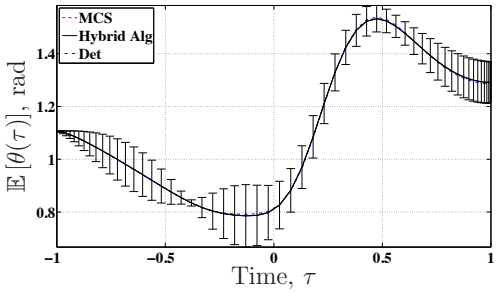
Figure 5.6. MCS, deterministic, and hybrid algorithm expected value estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #2.



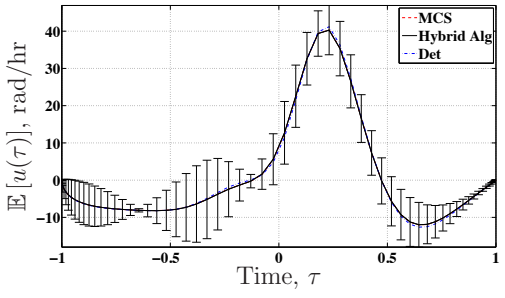
(a) Expected Value of $x_1(\tau)$



(b) Expected Value of $x_2(\tau)$

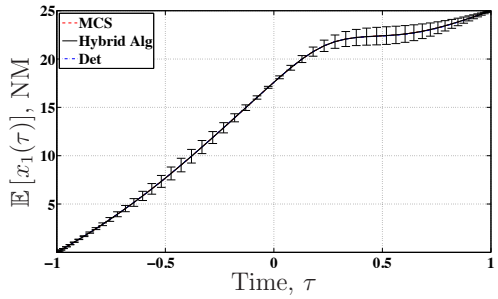


(c) Expected Value of $\theta(\tau)$

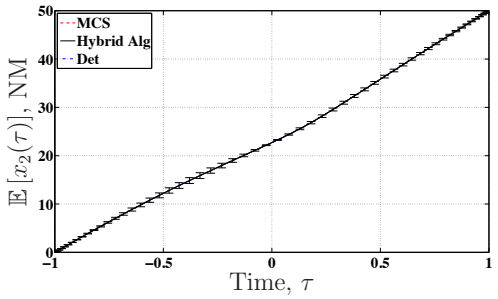


(d) Expected Value of $u(\tau)$

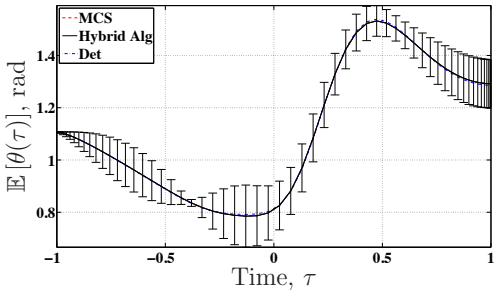
Figure 5.7. MCS, deterministic, and hybrid algorithm expected value estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #3.



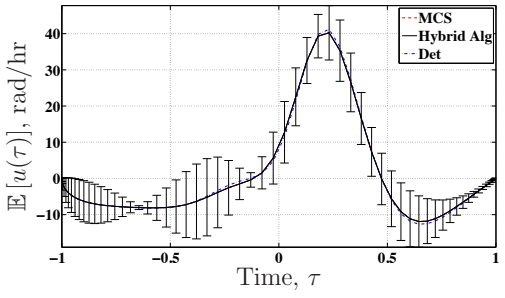
(a) Expected Value of $x_1(\tau)$



(b) Expected Value of $x_2(\tau)$

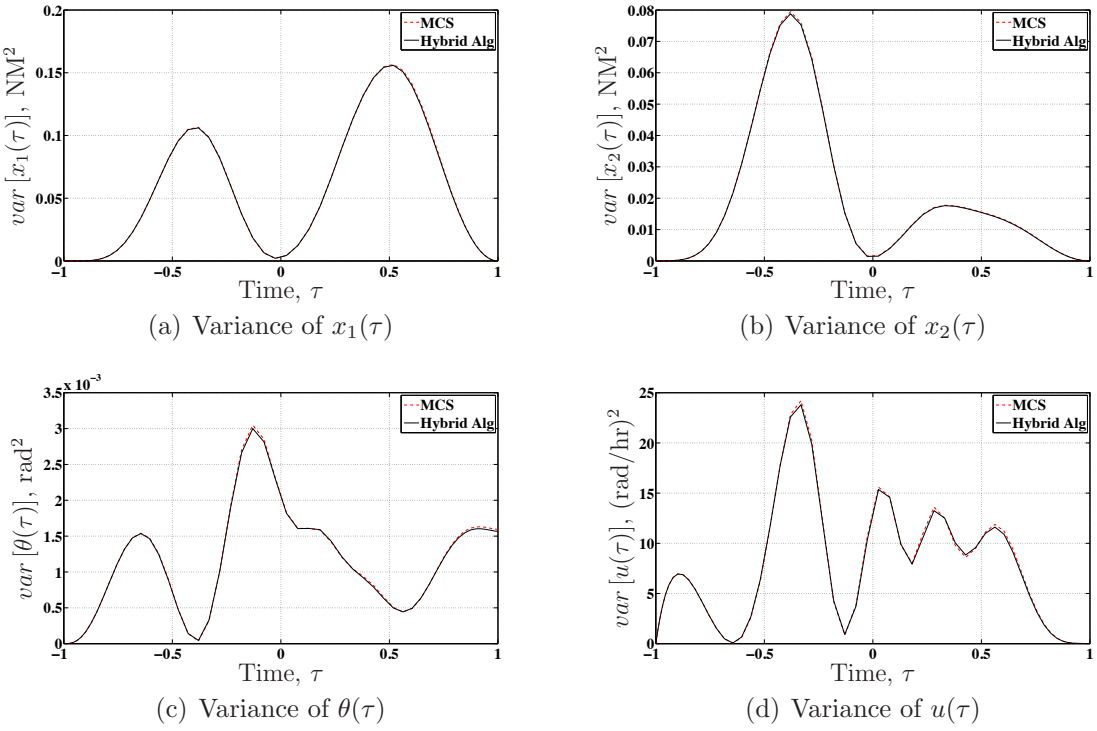


(c) Expected Value of $\theta(\tau)$



(d) Expected Value of $u(\tau)$

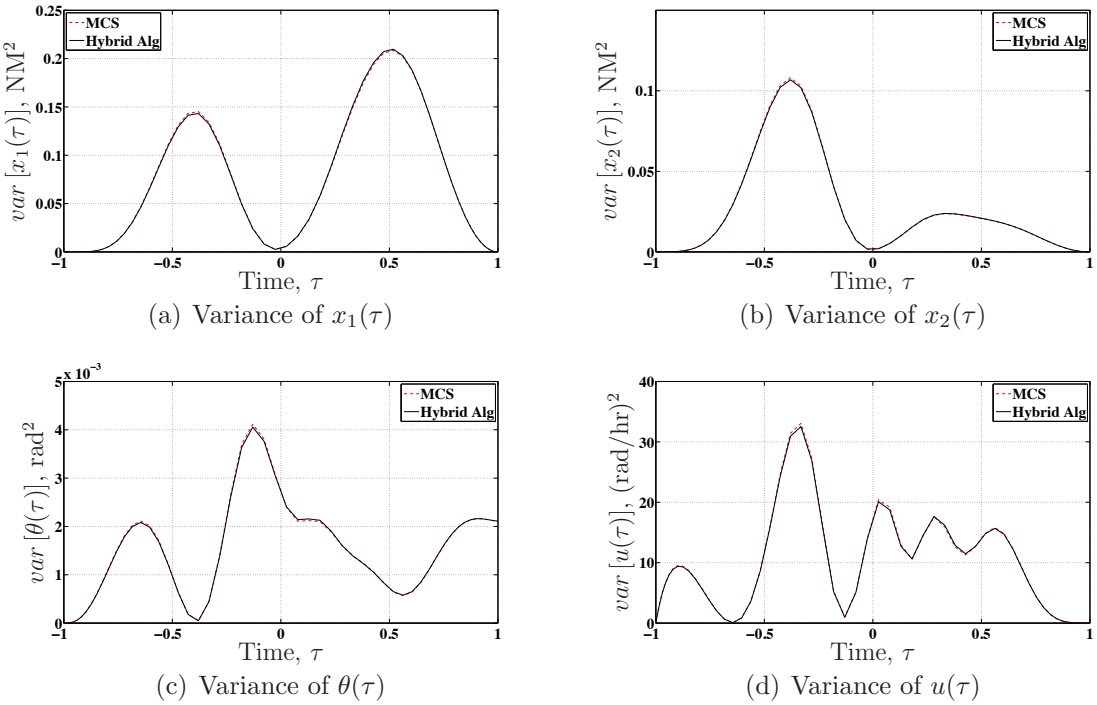
Figure 5.8. MCS, deterministic, and hybrid algorithm expected value estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #4.



(a) Variance of $x_1(\tau)$ (b) Variance of $x_2(\tau)$

(c) Variance of $\theta(\tau)$ (d) Variance of $u(\tau)$

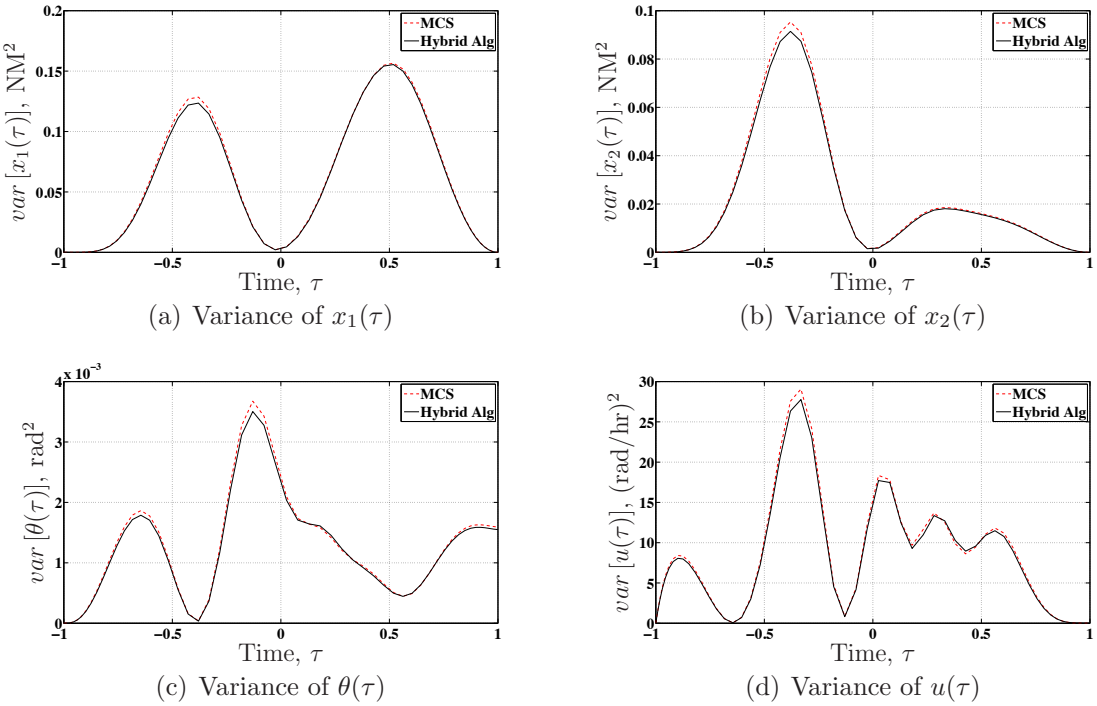
Figure 5.9. MCS and hybrid algorithm variance estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #1.



(a) Variance of $x_1(\tau)$ (b) Variance of $x_2(\tau)$

(c) Variance of $\theta(\tau)$ (d) Variance of $u(\tau)$

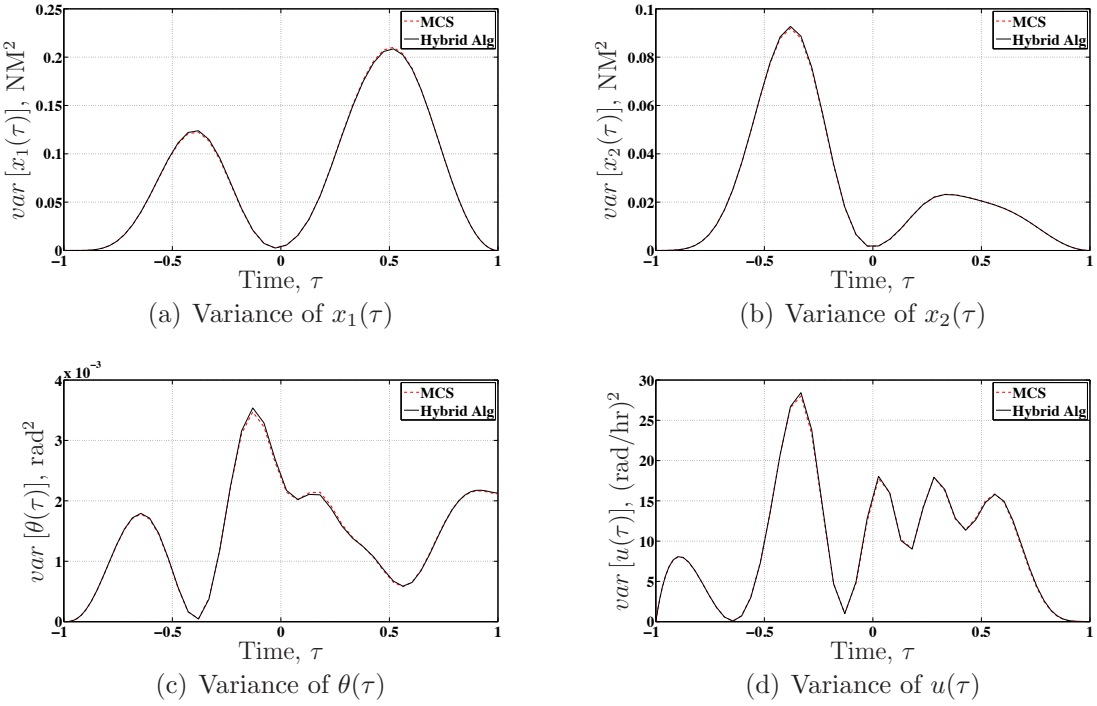
Figure 5.10. MCS and hybrid algorithm variance estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #2.



(a) Variance of $x_1(\tau)$ (b) Variance of $x_2(\tau)$

(c) Variance of $\theta(\tau)$ (d) Variance of $u(\tau)$

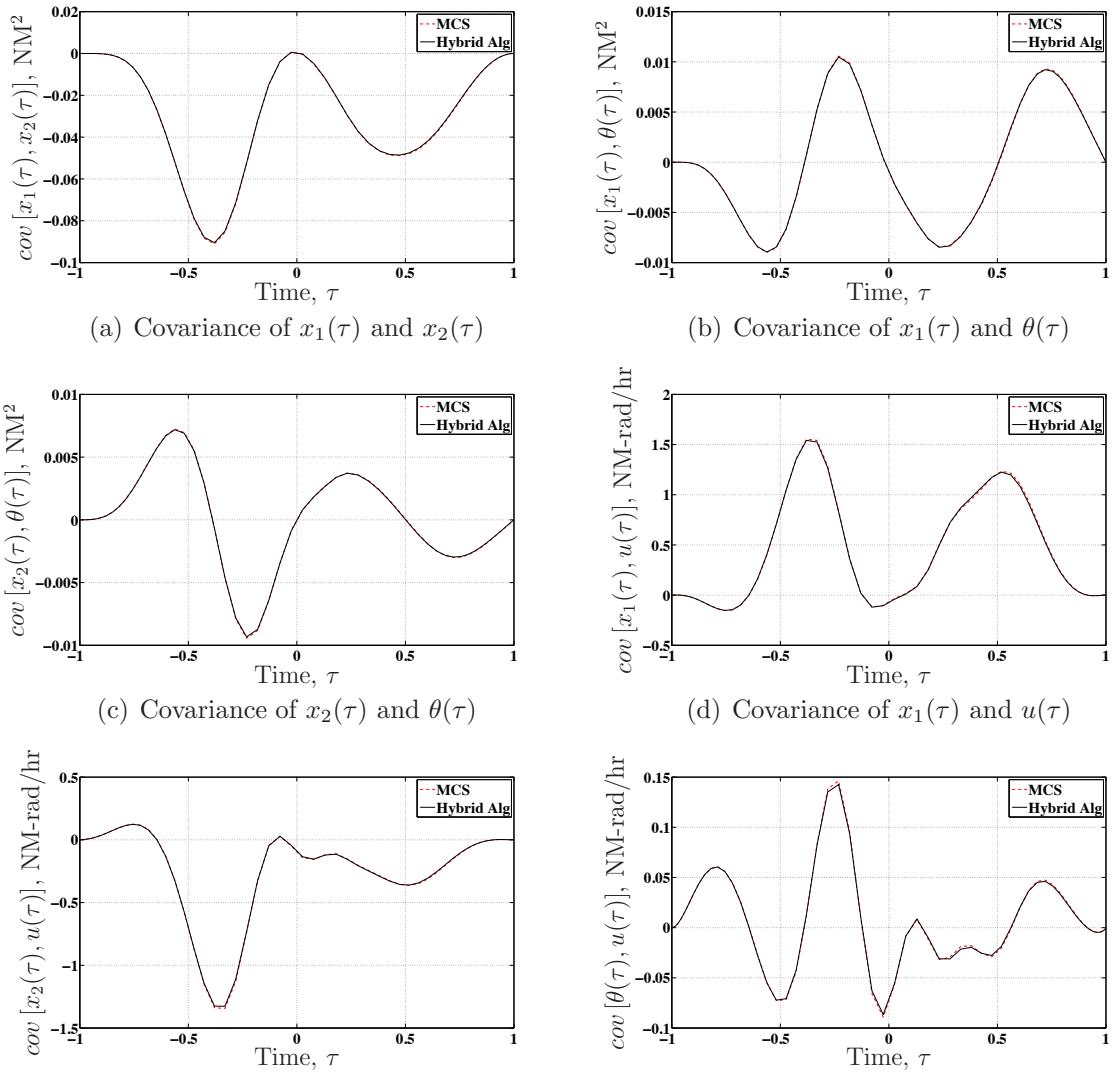
Figure 5.11. MCS and hybrid algorithm variance estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #3.



(a) Variance of $x_1(\tau)$ (b) Variance of $x_2(\tau)$

(c) Variance of $\theta(\tau)$ (d) Variance of $u(\tau)$

Figure 5.12. MCS and hybrid algorithm variance estimates of states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and control, $u(\tau)$, for configuration #4.



(a) Covariance of $x_1(\tau)$ and $x_2(\tau)$

(b) Covariance of $x_1(\tau)$ and $\theta(\tau)$

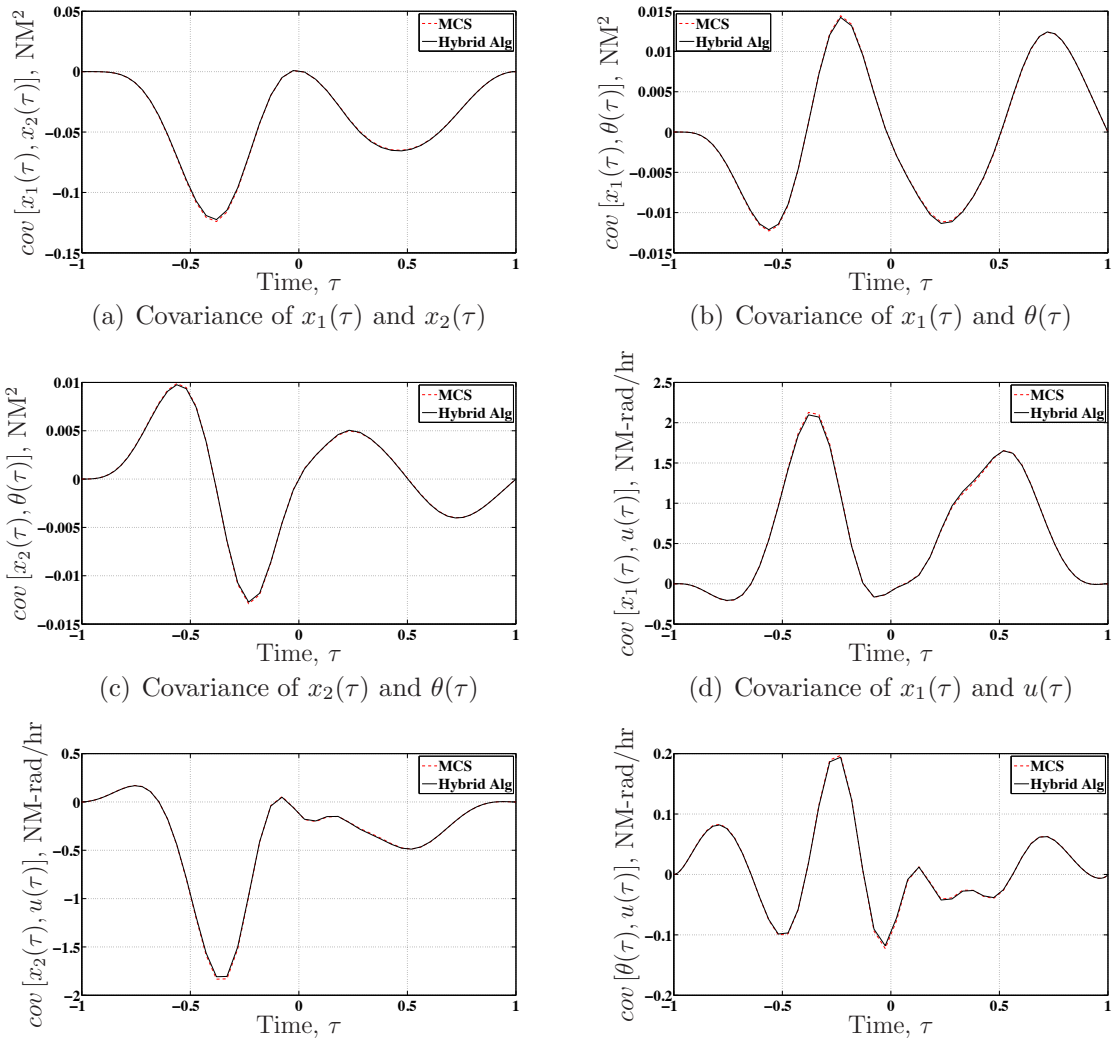
(c) Covariance of $x_2(\tau)$ and $\theta(\tau)$

(d) Covariance of $x_1(\tau)$ and $u(\tau)$

(e) Covariance of $x_2(\tau)$ and $u(\tau)$

(f) Covariance of $\theta(\tau)$ and $u(\tau)$

Figure 5.13. MCS and hybrid algorithm covariance estimates between the states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and between the states and control, $u(\tau)$, for configuration #1.



(a) Covariance of $x_1(\tau)$ and $x_2(\tau)$

(b) Covariance of $x_1(\tau)$ and $\theta(\tau)$

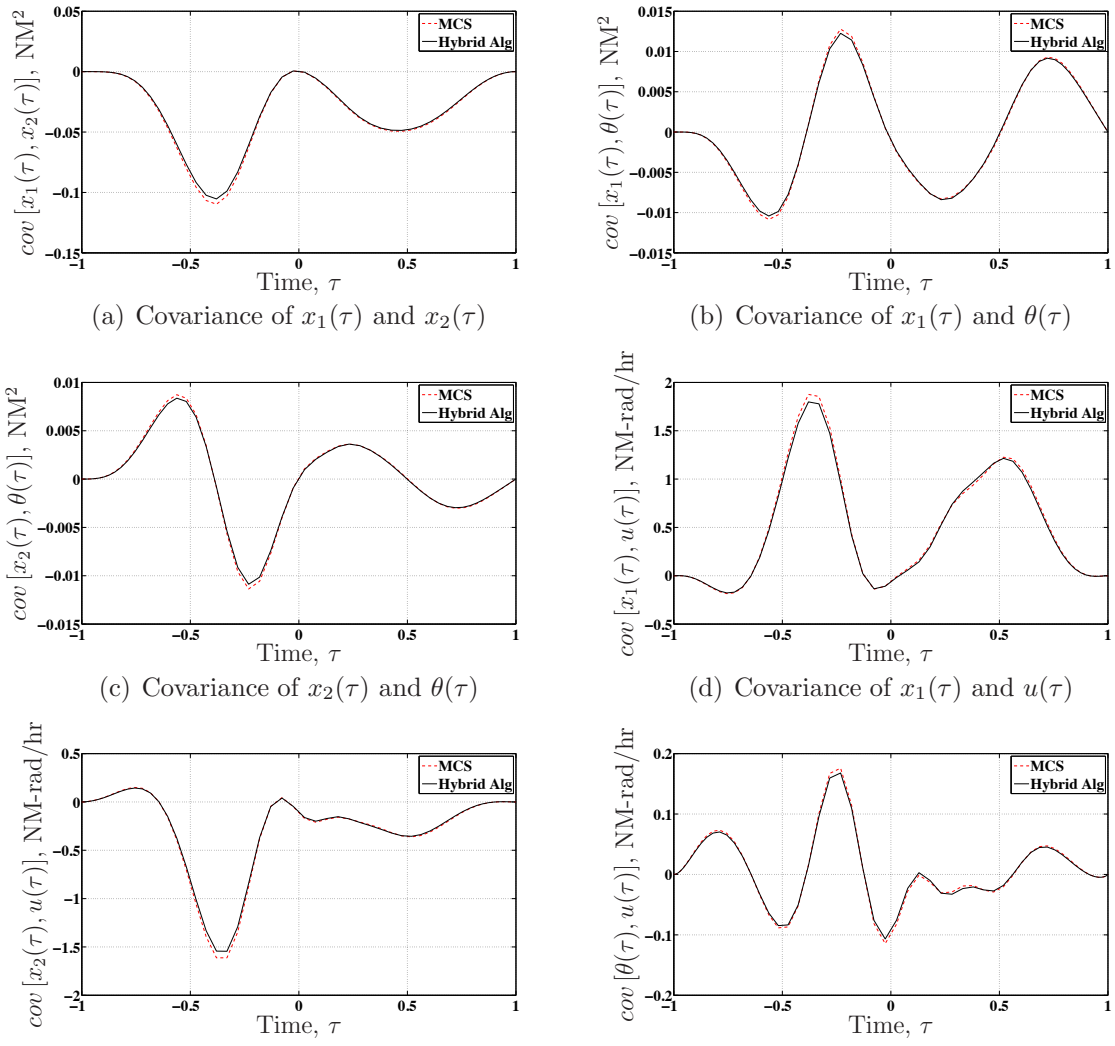
(c) Covariance of $x_2(\tau)$ and $\theta(\tau)$

(d) Covariance of $x_1(\tau)$ and $u(\tau)$

(e) Covariance of $x_2(\tau)$ and $u(\tau)$

(f) Covariance of $\theta(\tau)$ and $u(\tau)$

Figure 5.14. MCS and hybrid algorithm covariance estimates between the states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and between the states and control, $u(\tau)$, for configuration #2.



(a) Covariance of $x_1(\tau)$ and $x_2(\tau)$

(b) Covariance of $x_1(\tau)$ and $\theta(\tau)$

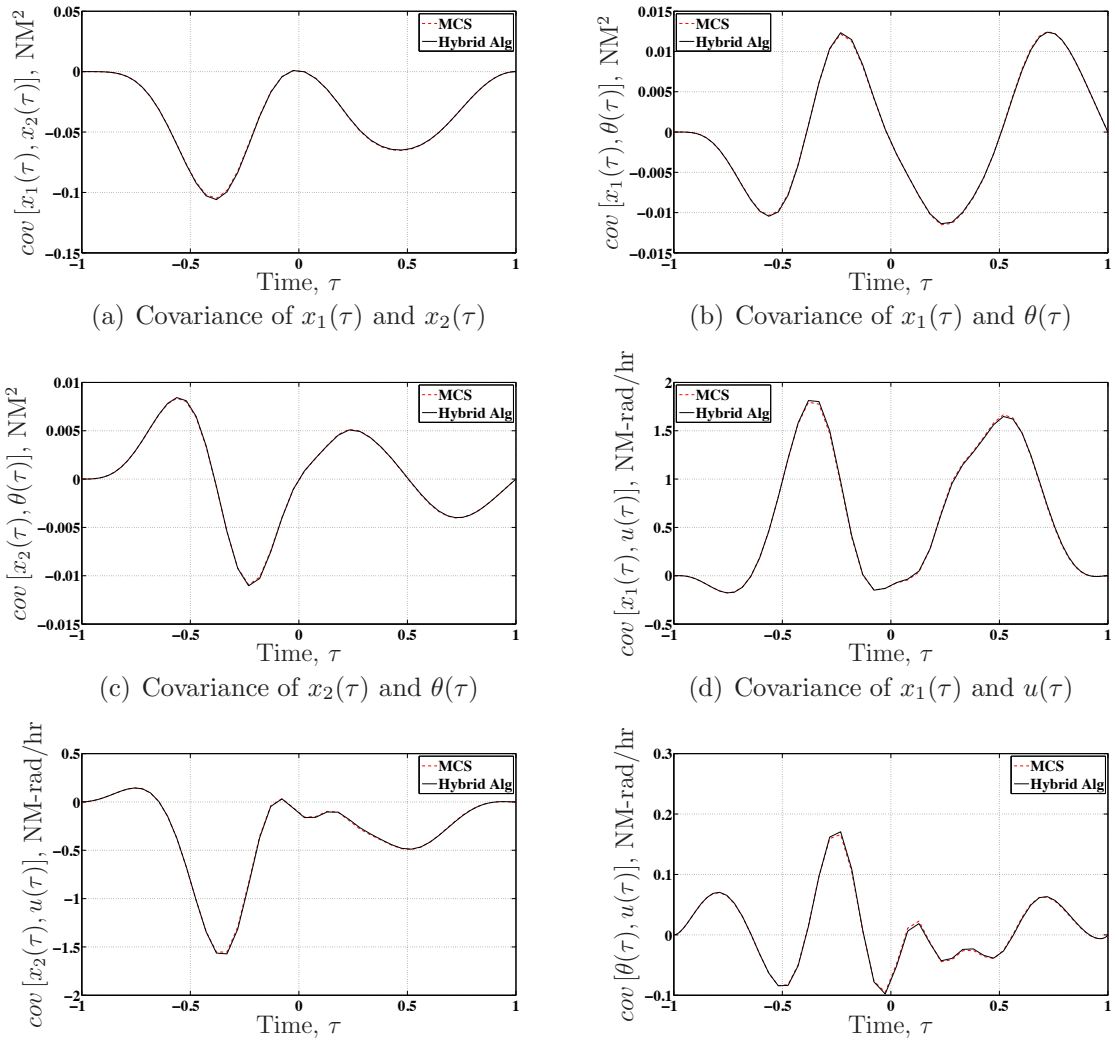
(c) Covariance of $x_2(\tau)$ and $\theta(\tau)$

(d) Covariance of $x_1(\tau)$ and $u(\tau)$

(e) Covariance of $x_2(\tau)$ and $u(\tau)$

(f) Covariance of $\theta(\tau)$ and $u(\tau)$

Figure 5.15. MCS and hybrid algorithm covariance estimates between the states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and between the states and control, $u(\tau)$, for configuration #3.



(a) Covariance of $x_1(\tau)$ and $x_2(\tau)$

(b) Covariance of $x_1(\tau)$ and $\theta(\tau)$

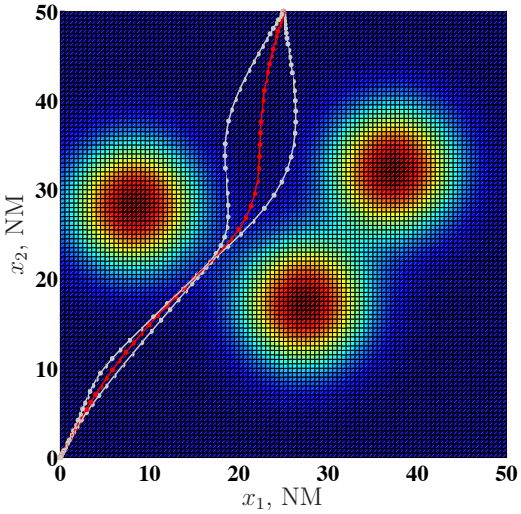
(c) Covariance of $x_2(\tau)$ and $\theta(\tau)$

(d) Covariance of $x_1(\tau)$ and $u(\tau)$

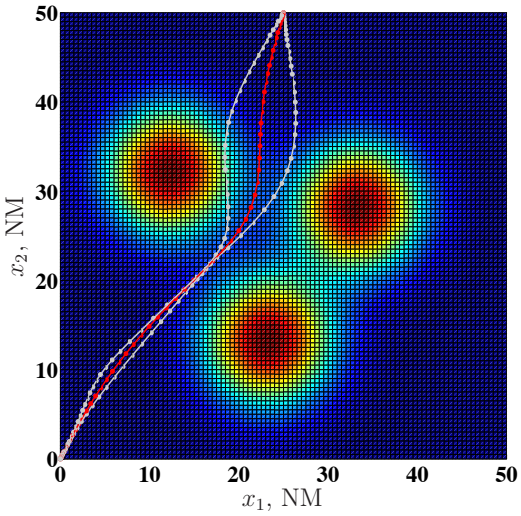
(e) Covariance of $x_2(\tau)$ and $u(\tau)$

(f) Covariance of $\theta(\tau)$ and $u(\tau)$

Figure 5.16. MCS and hybrid algorithm covariance estimates between the states, $x_1(\tau)$, $x_2(\tau)$, and $\theta(\tau)$, and between the states and control, $u(\tau)$, for configuration #4.



(a) Best case threat PDF configuration



(b) Worst case threat PDF configuration

Figure 5.17. Trajectory window with threat rings defined by POK PDF.

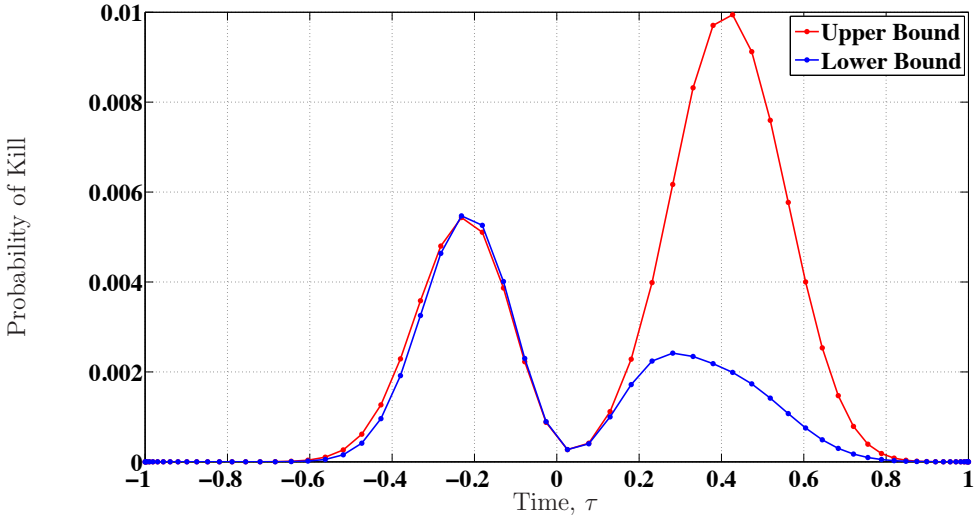


Figure 5.18. POK estimates at each time point for best and worst case scenarios.

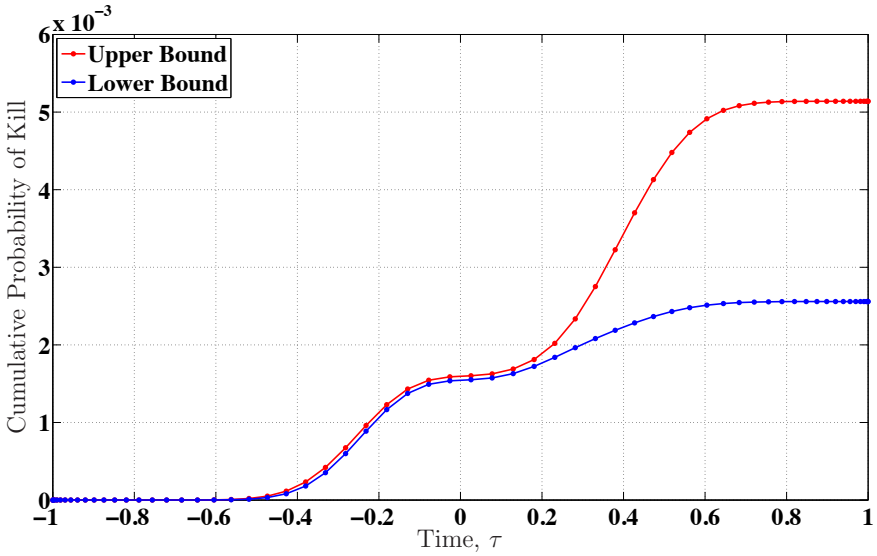
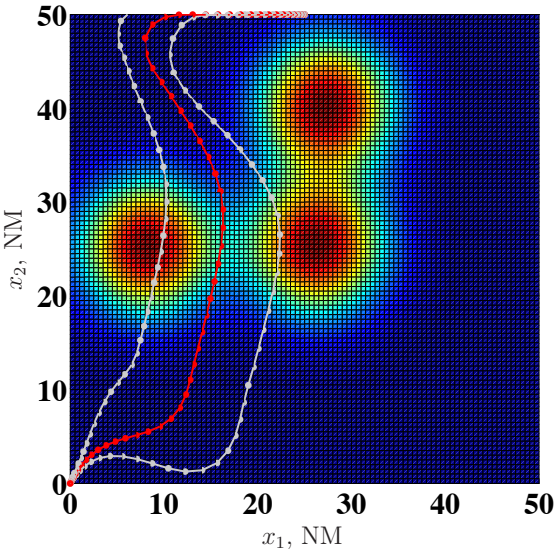
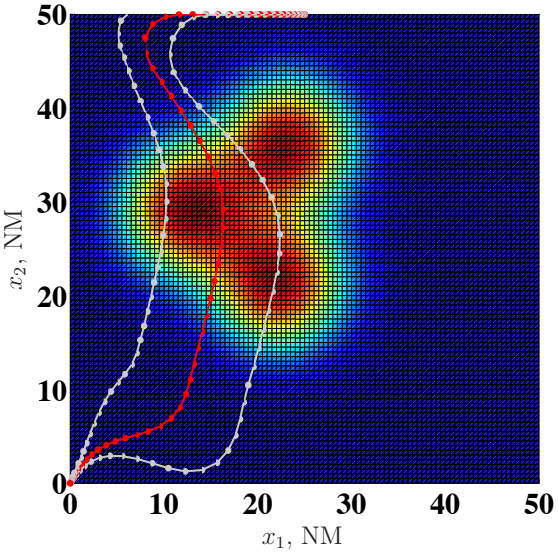


Figure 5.19. Cumulative POK estimates for best and worst case scenarios.



(a) Best case threat PDF configuration



(b) Worst case threat PDF configuration

Figure 5.20. Trajectory window and threat rings defined by POK PDF in modified scenario.

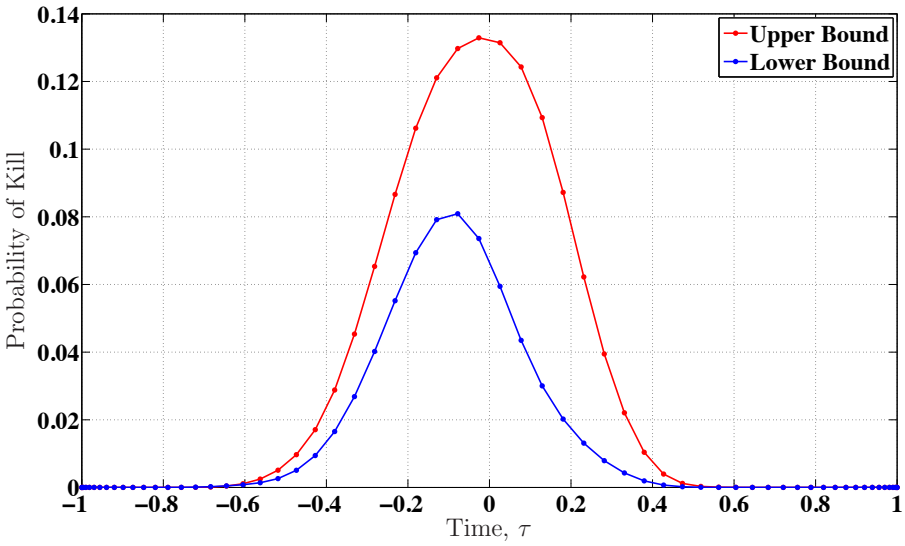


Figure 5.21. POK estimates at each time point for best and worst case scenarios using modified threat locations.

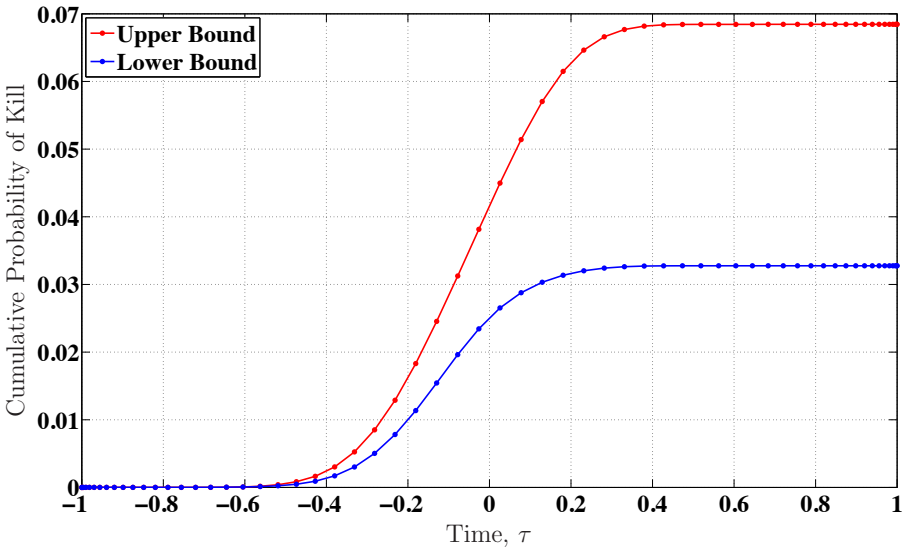


Figure 5.22. Cumulative POK estimates for best and worst case scenarios using modified threat locations.

Table 5.3. Difference between hybrid algorithm and MCS results

(a) Case #1

S	Max Difference ($\lvert S_{MCS} - S_{alg} \rvert$)	PD (%)
Mean		
$x_1(\tau)$	5.825E^{-3}	0.0262
$x_2(\tau)$	1.950E^{-3}	0.01027
$\theta(\tau)$	9.214E^{-4}	0.09503
$u(\tau)$	8.522E^{-2}	0.2409
Variance		
$x_1(\tau)$	1.403E^{-3}	1.126
$x_2(\tau)$	6.815E^{-4}	0.901
$\theta(\tau)$	4.798E^{-5}	1.587
$u(\tau)$	0.4126	1.719
Covariance		
$x_1(\tau), u(\tau)$	2.332E^{-2}	2.528
$x_2(\tau), u(\tau)$	1.702E^{-2}	1.275
$\theta(\tau), u(\tau)$	3.689E^{-3}	2.554
$x_1(\tau), x_2(\tau)$	6.25E^{-4}	0.7332
$x_1(\tau), \theta(\tau)$	1.638E^{-4}	23.92
$x_2(\tau), \theta(\tau)$	1.149E^{-4}	1.224

MCS: 30K samples, 387.1 minutes processing
Hybrid algorithm: 6.5 minutes processing

(b) Case #2

S	Max Difference ($\lvert S_{MCS} - S_{alg} \rvert$)	PD (%)
Mean		
$x_1(\tau)$	5.205E^{-3}	0.02332
$x_2(\tau)$	1.217E^{-3}	0.003761
$\theta(\tau)$	8.404E^{-4}	0.07754
$u(\tau)$	0.1004	0.4471
Variance		
$x_1(\tau)$	2.002E^{-3}	1.138
$x_2(\tau)$	1.228E^{-3}	1.143
$\theta(\tau)$	5.992E^{-5}	1.467
$u(\tau)$	0.5126	1.563
Covariance		
$x_1(\tau), u(\tau)$	3.088E^{-2}	1.481
$x_2(\tau), u(\tau)$	2.496E^{-2}	1.373
$\theta(\tau), u(\tau)$	4.702E^{-3}	3.908
$x_1(\tau), x_2(\tau)$	1.586E^{-3}	1.287
$x_1(\tau), \theta(\tau)$	2.062E^{-4}	1.437
$x_2(\tau), \theta(\tau)$	1.721E^{-4}	1.343

MCS: 17.5K samples, 221.8 minutes processing
Hybrid algorithm: 5.6 minutes processing

(c) Case #3

S	Max Difference ($\lvert S_{MCS} - S_{alg} \rvert$)	PD (%)
Mean		
$x_1(\tau)$	3.7E^{-3}	0.04095
$x_2(\tau)$	2.374E^{-3}	0.01718
$\theta(\tau)$	8.359E^{-4}	0.07706
$u(\tau)$	0.1379	1.09
Variance		
$x_1(\tau)$	4.977E^{-3}	4.002
$x_2(\tau)$	3.879E^{-3}	4.153
$\theta(\tau)$	1.713E^{-4}	4.772
$u(\tau)$	1.293	4.55
Covariance		
$x_1(\tau), u(\tau)$	7.787E^{-2}	4.238
$x_2(\tau), u(\tau)$	6.936E^{-2}	4.396
$\theta(\tau), u(\tau)$	7.935E^{-3}	4.614
$x_1(\tau), x_2(\tau)$	4.4E^{-3}	4.092
$x_1(\tau), \theta(\tau)$	5.251E^{-4}	4.193
$x_2(\tau), \theta(\tau)$	4.911E^{-4}	4.415

MCS: 34K samples, 453.2 minutes processing
Hybrid algorithm: 6.6 minutes processing

(d) Case #4

S	Max Difference ($\lvert S_{MCS} - S_{alg} \rvert$)	PD (%)
Mean		
$x_1(\tau)$	6.796E^{-3}	0.03057
$x_2(\tau)$	1.718E^{-3}	0.00749
$\theta(\tau)$	1.185E^{-3}	0.1093
$u(\tau)$	0.1359	1.071
Variance		
$x_1(\tau)$	1.691E^{-3}	0.8180
$x_2(\tau)$	9.097E^{-4}	1.207
$\theta(\tau)$	6.581E^{-5}	2.021
$u(\tau)$	0.5254	2.231
Covariance		
$x_1(\tau), u(\tau)$	2.985E^{-2}	2.01
$x_2(\tau), u(\tau)$	2.377E^{-2}	1.822
$\theta(\tau), u(\tau)$	4.704E^{-3}	54.16
$x_1(\tau), x_2(\tau)$	1.214E^{-3}	1.228
$x_1(\tau), \theta(\tau)$	2.018E^{-4}	1.774
$x_2(\tau), \theta(\tau)$	1.354E^{-4}	1.325

MCS: 17.5K samples, 234.6 minutes processing
Hybrid algorithm: 5.1 minutes processing

Table 5.4. Hybrid algorithm and MCS expected value and variance of J

Case	$\mathbb{E}[J]$ Alg	$\mathbb{E}[J]$ MCS	PD (%)	var(J) Alg	var(J) MCS	PD (%)
1	1.6156E^{-5}	1.6160E^{-5}	2.4756E^{-2}	3.5976E^{-13}	3.5961E^{-13}	4.1703E^{-2}
2	1.6136E^{-5}	1.6155E^{-5}	0.1177	4.8037E^{-13}	4.9090E^{-13}	2.1683
3	1.6148E^{-5}	1.6144E^{-5}	2.4774E^{-2}	3.6575E^{-13}	3.7227E^{-13}	1.7669
4	1.6162E^{-5}	1.6163E^{-5}	6.1872E^{-3}	4.7421E^{-13}	4.7438E^{-13}	3.5843E^{-2}

Deterministic solution of J: 1.6164E^{-5}

Table 5.5. Hybrid algorithm and MCS expected value and variance of t_f

Case	$\mathbb{E}[t_f]$ Alg	$\mathbb{E}[t_f]$ MCS	PD (%)	var(t_f) Alg	var(t_f) MCS	PD (%)
1	0.1231	0.1231	0	1.5542E^{-7}	1.5493E^{-7}	0.3158
2	0.1231	0.1231	0	2.0915E^{-7}	2.0793E^{-7}	0.5850
3	0.1231	0.1231	0	1.6921E^{-7}	1.7553E^{-7}	3.6665
4	0.1231	0.1231	0	1.8987E^{-7}	1.9238E^{-7}	1.3133

Deterministic solution of t_f: 0.1230 hours

VI. Conclusions and Recommendations

A hybrid numerical algorithm was presented in this dissertation which combines the GPM with the collocation form of the gPC method to quantify the effects of stochastic parameters on OC and TO problem solutions. The GPM and gPC have both been shown to be spectrally accurate numerical methods for solving deterministic optimal control problems and stochastic differential equations, respectively. The gPC method samples the random space using collocation nodes, which are then inserted into the differential equations creating a set of deterministic differential equation problems that are solved using standard ODE and PDE solvers. The resulting set of deterministic solutions are used in stochastic computations to characterize the distribution of the solution by constructing a polynomial representation of the output as a function of uncertain parameters. If OC and TO problems are considered in the context of the HBVP formulation, an indirect transcription of the optimization problem into a differential equation problem, and the GPM is thought of as a numerical method that produces an approximate solution to the optimization problem that is equivalent to an approximate pseudospectral solution to the HBVP, then it is reasonable to theorize that a numerical GPM solver such as GPOPS could be used in place of the deterministic differential equation solvers traditionally used in the gPC algorithm. Thus, the hybrid algorithm considered in this document integrates the GPOPS deterministic OC problem solver into the gPC construct thereby providing a set of spectrally accurate numerical solutions to the sampled problems that satisfy boundary, path, and bounded state and control constraints while minimizing the cost functional and are suitable for stochastic computations, thus extending the current and well-established gPC method of solving SDEs to be applicable to TO and OC problems. The goal of the research presented in this document was to demonstrate that the hybrid algorithm combining the GPM and gPC methods can effectively be

applied to quantify the effects of uncertainty in stochastic optimization problems.

6.1 Conclusions

The hybrid GPM-gPC algorithm was applied to two nonlinear concept demonstration problems to demonstrate the ability of the hybrid algorithm to solve nonlinear optimization problems with uncertain parameters.

The first problem, an adaptation of a textbook problem, was an OC problem with nonlinear dynamics and cost functional, fixed final state and time, and multiplicative Gaussian uncertainties effecting the state variable dynamics equations. The solution to the problem was a set of polynomial functions approximating state variables, control, and cost solutions whose expected values, variances, and covariances, were found using the expansion coefficients to describe the statistical properties of the solution. Comparing the statistical properties determined by application of the hybrid algorithm with the MCS results demonstrated close agreement between the two methods. The error differences between the two methods did however bring into question whether closer agreement would result if either the gPC parameters, number of collocation points and order of polynomial approximation, or the MCS convergence tolerance were adjusted. A second MCS with lower convergence tolerance was conducted resulting in differences between the two methods decreasing by 28 to 67%, leading to the conclusion that the hybrid algorithm was actually providing more accurate approximations. The results presented show that the hybrid algorithm is able to quantify the effects the Gaussian parameters on the optimal solution of a very challenging nonlinear problem while requiring dramatically fewer sample points and associated GPOPS deterministic solutions and computation time.

The second concept demonstration problem was designed to be representative of a real-world mission scenario that has USSTRATCOM interest. The scenario was

constructed to find the path through a two-dimensional space that minimizes the POK by lethal threats whose locations are uncertain, to quantify the effects those uncertainties have on the trajectory solution by estimating the statistical properties, and to use the statistical properties to estimate the probability of the vehicle being killed during the mission. It is a TO problem that is nonlinear in the state equations and cost functional, has fixed final state and free final time specifications, and incorporates additive uncertainties that effect the cost functional. Four cases were evaluated where the uncertainties in the threat locations were modeled as random variables with either all Gaussian, all uniform, or a mix of the two distributions. Similar to the first concept demonstration problem, the algorithm generated a set of polynomial approximation functions for the state variables, control, cost, and terminal time whose expected values, variances, and covariances, were calculated using the expansion coefficients to describe the statistical properties of the solution. Comparing the statistical properties determined by application of the hybrid algorithm with the MCS results again demonstrated that the hybrid algorithm effectively quantifies the effects of uncertainty with comparable accuracy to the MCS while requiring dramatically fewer sample points, GPOPS calls, and computation time. Additionally, mixing probability distributions, and thus polynomial bases, did not seem to cause any appreciable change in algorithm performance requiring modifications to numbers of collocation points or polynomial order, or unfavorable comparison with the MCS results. Probability of kill computations, which really was a repackaging of the hybrid algorithm's statistical data outputs, demonstrated that the results can be presented in a form that can be used by mission planners and aircrews to asses risks associated with a mission trajectory solution.

Considering that two types of nonlinear optimal control problems were investigated with fixed boundary conditions in the first problem and a mix of fixed and free

boundary conditions in the second, and with uncertainties effecting different equations in the problem formulations, it is apparent that inserting the GPM into the gPC construct results in a new algorithm that is capable of solving a variety of nonlinear optimal control problems with uncertain parameters. The results presented have demonstrated that the method is able to generate solutions to the stochastic TO and OC problems by determining the expected value solutions for the state variables, control, cost, terminal time and characterize the probabilistic information about the solutions that agrees with MCS results, thus characterizing how the optimal solution changes with uncertainty. Therefore, the hybrid GPM-gPC algorithm developed herein extends gPC methods by providing a tool with great potential for uncertainty quantification in optimal control and trajectory optimization problems.

6.2 Recommendations for Future Research

The concept demonstration problems considered in this work were designed to be manageable within the confines of the limited processing capabilities that were available. The results presented indicate that processing requirements in terms of computation time and number of sample points required to solve the demonstration problems was reasonable, but slight changes to these problems, such as adding additional state equations, increasing the number of random dimensions, or widening the PDF windows of the uncertain parameters, causes dramatic increases in computational burden that quickly exceeds the capabilities of available resources. Therefore, further investigation should first focus on improving computational efficiency of the algorithm by determining guidelines for choosing the numbers of polynomial basis elements combined with incorporation of sparse grids to reduce the number of collocation points and GPOPS calls. Calling GPOPS 441 times in the first problem, and 343 times in the second, while using fifth- and seventh-order polynomial bases, respec-

tively, was not overly time consuming, but keeping in mind that GPOPS uses an NLP solver, which is an iterative routine in itself, to calculate state and control solutions that meet optimality conditions, it's obvious that as the number of states, controls, and random inputs grow the computational burden will increase exponentially.

Once efficiency improvements have been considered, further research should be conducted using a high-performance workstation with multiple processors and higher memory capacity enabling expansion of the problem formulations. Although GPOPS is not written for parallel processing, the hybrid algorithm code evaluating the expansion coefficients and output functions was able to take advantage of the available multi-core processor and parallel processing functionality in MATLAB®to provide noticeable improvement in processing time. Further implementation on a multiprocessor computer is expected to provide even more improvements in processing time and enable problem expansions. The first problem would be suitable for considering time-varying random parameters effecting the state equations. Further developing the algorithm to quantify the effects of time-varying uncertainties such as wind gusts or sensor noise could eventually lead to applying the hybrid algorithm to real- or near-real-time state estimation and control problems. The second problem should be expanded by adding three additional random elements to more realistically model potential threat movement in the two-dimensional space. This expansion was attempted, but using the same seven points, seventh-order polynomial bases, and tensor product spaces resulted in the number of collocation points increasing from 343 to 117,649 and the number of basis polynomials increasing from 512 to 262,144, which was beyond the capabilities of available computers. On a more powerful computer, additional threats could also be added as desired to represent other mission scenarios USSTRATCOM may encounter.

With improved computational efficiency, high-performance computer resources,

and algorithm enhancements to solve problems with time-varying random inputs, the algorithm could then be applied to MAV research. MAV control problems are challenging because the small, lightweight aircraft are inherently nonlinear and vulnerable to exogenous disturbances such as wind gusts and turbulence. Vehicle models are still being developed through theoretical analysis and both wind tunnel and flight testing since traditional aerodynamic theory does not adequately describe low Reynolds number aerodynamics. Low-cost sensors used on these vehicles have more measurement noise and errors than higher-fidelity sensors. Therefore, MAV problems are stochastic in nature with model uncertainties, measurement noise, and time-varying random disturbances. The hybrid algorithm may provide an effective tool to solve nonlinear MAV optimization problems while including these stochastic elements.

Applying the hybrid GPM-gPC algorithm to other classes of nonlinear OC problems with uncertain parameters is also recommended as future research. Two types of problems where the hybrid algorithm may be effective in quantifying the effects of uncertain parameters on optimal solutions are *bifurcation* and *chaotic* problems. Bifurcation occurs in a dynamical system when a "quantitative change of parameters" leads to a "qualitative change in system properties" [76]. In the context of applying the hybrid algorithm to stochastic OC problems, uncertain parameters effecting the state equations may result in a *qualitative change of system properties* where solutions to the set of deterministically sampled problems used in stochastic computations may result in clustering around more than one apparent expected value. This clustering would indicate discontinuity in the functional relationship between the approximate solution and the uncertain parameters, suggesting that effective application of the hybrid algorithm may require a method for determining the points in the finite probability space where bifurcation occurs and application of the hybrid algorithm over each corresponding sub-domain.

The second class of nonlinear OC problems involves applying the hybrid algorithm to chaotic systems where uncertainty is included. Chaotic behavior is seen in strongly nonlinear systems and is characterized by "output[s] that are extremely sensitive to initial conditions" where very small changes in the initial conditions result in radically different state trajectories [76]. Optimal control problems could be considered where uncertainties are incorporated in initial conditions or in both initial conditions and parameters effecting the dynamics equations to investigate the ability of the hybrid algorithm to quantify the effects of the uncertainties on the solutions. A comparison should be made between the output approximation function evaluated at several test points drawn from the assumed probability distributions of the uncertain parameters and deterministic solutions based on those test points to verify the accuracy of the hybrid algorithm approximation. Additionally, since system responses are extremely sensitive to small changes in initial conditions, a convergence analysis should be performed to determine if exponential convergence is preserved since the approximate solutions may not be smoothly dependent on the uncertain parameters. Investigating applying the hybrid GPM-gPC algorithm to these two classes of nonlinear problems in conjunction with the previously recommended research will further extend the gPC method to be a useful tool in determining the effects uncertainties have on OC problem solutions.

The work presented in this document is a first step in combining GPM and gPC numerical methods to consider nonlinear optimal control problems effected by stochastic elements. The results have demonstrated that the hybrid algorithm is capable of quantifying the effects of uncertainties in problems that are meaningful to the USAF and shows great potential for further applications to solve challenging control problems.

6.3 Sponsor Acknowledgement

This research was sponsored by the Joint Functional Component Command for Global Strike, United States Strategic Command, and supports their efforts to increase efficiency in their mission planning tools and process.

Appendix A. Hybrid GPM-gPC Code

Selected lines of MATLAB®code are given in this appendix to show how the steps of the hybrid GPM-gPC algorithm presented in Chapter III were implemented to solve the TO problem in Chapter V. Similar code was used to produce the results presented for the OC problem in Chapter IV.

1.1 Collocation Nodes

Step #1: Calculate collocation nodes and quadrature weights $\{\mathbf{p}_j, \alpha_j\}_{j=1}^{Q}$:

Listing A.1. Calculate collocation nodes and quadrature weights

```
% Define constants needed to set up the TO problem
constants.v=470;  % Cruise speed in Kt
constants.ft2Nm=1/6076.11549;  % Convert ft to nautical miles
constants.deg2rad=pi/180; % Convert degrees to radians
constants.Rmin=11326*constants.ft2Nm; % Assumed turn radius ...
   converted to NM
% Properties of the 2-variate Gaussian PDF used to model threat ...
   rings
constants.rho1=0.0; constants.rho2=0.0; constants.rho3=0.0;
constants.sigt1x1=5; constants.sigt2x1=5; constants.sigt3x1=5;
constants.sigt1x2=5; constants.sigt2x2=5; constants.sigt3x2=5;
% Center locations of threats
constants.c1x1=10; constants.c1x2=30;
constants.c2x1=25; constants.c2x2=15;
constants.c3x1=35; constants.c3x2=30;

N=3; % Specify number of random dimensions (random parameters)
q=7; % Specify number of collocation points in each random ...
   dimension
Q=q^N; % Total number of collocation points
d1=7; d2=7; d3=7; P=21; % 1-D polynomial order (d_i) and N-D ...
   polynomial overall order (P)
mu=0; dev=0.5; % Define properties of Gaussian distributions

% For Gaussian distributions
    [a,wa]=hermquadpts(q1); % hermquadpts function from Mathworks ...
       website
    [b,wb]=hermquadpts(q2);
    [c,wc]=hermquadpts(q3);

    a=a.*sqrt(2*dev^2)+mu;  % scale points
    b=b.*sqrt(2*dev^2)+mu;
```

```
    c=c.*sqrt(2*dev^2)+mu;

    wa=wa./sqrt(pi); % scale quadrature weights
    wb=wb./sqrt(pi);
    wc=wc./sqrt(pi);

% For uniform distributions
    [a,wa]=legquadpts(q1); % legquadpts function from Mathworks ...
       website
    [b,wb]=legquadpts(q2);
    [c,wc]=legquadpts(q3);

    lb=-1; % lower bound of uniform distribution
    ub=1;  % upper bound of uniform distribution

    a=((a./2) +(1/2)).*(ub-lb)+lb; % scale points
    b=((b./2) +(1/2)).*(ub-lb)+lb;
    c=((c./2) +(1/2)).*(ub-lb)+lb;

    wa=wa./sum(wa); % scale weights
    wb=wb./sum(wb);
    wc=wc./sum(wc);

% Construct tensor product point and weight sets
count=1;
for ii=1:q1
    for jj=1:q2
        for kk=1:q3
        qc(count,:)=[a(ii),b(jj),c(kk)]; % array of Q points
        wtc(count,:)=[wa(ii),wb(jj),wc(kk)];
        wcprod(count)=wa(ii)*wb(jj)*wc(kk); % array of Q weights
        count=count+1;
        end
    end
end
```

1.2 Apply GPOPS to Deterministic Sample Problems

Step #2: Using each of the Q collocation points $\mathbf{p}_j$, solve the set of deterministic OC problems using GPOPS:

Listing A.2. Solve Q deterministic problems using GPOPS

```
for ii=1:Q
    % Seperate samples for each dynamics equation
        ac=qc(ii,1);
    bc=qc(ii,2);
    cc=qc(ii,3);
```

```
    % Call the main GPOPS script
    MyProblemMain;
    % Save state trajectory data
    x1data=[x1data;x1out'];
    x2data=[x2data;x2out'];
    x3data=[x3data;x3out'];
    % Save control data
    udata=[udata;uout'];
    % Save cost data
    cost=[cost;costout];
    % Save time data
    tdata=[tdata;tout'];
end
```

1.3 Expansion Coefficients

Step #3: Evaluate the expansion coefficients using (3.1):

$$\hat{z}_m = \sum_{j=1}^{Q} z(\mathbf{p}_j)\Phi_m^c(\mathbf{p}_j)\alpha_j; \quad (m = 1, ..., M)$$

Listing A.3. Evaluate expansion coefficients

```
% Build the Hermite polynomial (H) or Legendre polynomial (P) ...
   basis functions and evaluate at all combinations of a, b, and c

% Calculate polynomial normaliztion factors from Xiu's book [86]
% For Gaussian distributions
for ii=0:d1
        int1(ii+1)=factorial(ii);
end

for ii=0:d2
        int2(ii+1)=factorial(ii);
end

for ii=0:d3
        int3(ii+1)=factorial(ii);
end

% For uniform distributions
for ii=0:d1
        int1(ii+1)=1/(2*ii+1);
end

for ii=0:d2
```

```
        int2(ii+1)=1/(2*ii+1);
end

for ii=0:d3
        int3(ii+1)=1/(2*ii+1);
end

% Build polynomial bases
% For Gaussian distributions...use Hermite polynomial basis ...
   functions (hermite function from Mathworks website)
for ii=1:Q
    m=1;
    for jj=0:d1
        for kk=0:d2
            for ll=0:d3
            h1(ii,m)=(1/((2)^(jj/2)))*hermite(jj,(qc(ii,1)-mu)/(...
               sqrt(2*dev^2)))/sqrt(int1(jj+1));
            h2(ii,m)=(1/((2)^(kk/2)))*hermite(kk,(qc(ii,2)-mu)/(...
               sqrt(2*dev^2)))/sqrt(int2(kk+1));
            h3(ii,m)=(1/((2)^(ll/2)))*hermite(ll,(qc(ii,3)-mu)/(...
               sqrt(2*dev^2)))/sqrt(int3(ll+1));

                if jj+kk+ll<=P
                    Phi(ii,m)=h1(ii,m)*h2(ii,m)*h3(ii,m);
                    m=m+1;
                end
            end
        end
    end
end

% For uniform distributions...use Legendre polynomial basis ...
   functions (LegendrePoly function from Mathworks website)
for ii=1:Q
    m=1;
    for jj=0:d1
        for kk=0:d2
            for ll=0:d3
            h1(ii,m)=polyval(LegendrePoly(jj),2*((qc(ii,1)-lb)/(ub...
               -lb))-1)/sqrt(int1(jj+1));
            h2(ii,m)=polyval(LegendrePoly(kk),2*((qc(ii,2)-lb)/(ub...
               -lb))-1)/sqrt(int2(kk+1));
            h3(ii,m)=polyval(LegendrePoly(ll),2*((qc(ii,3)-lb)/(ub...
               -lb))-1)/sqrt(int3(ll+1));

                if jj+kk+ll<=P
                    Phi(ii,m)=h1(ii,m)*h2(ii,m)*h3(ii,m);
                    m=m+1;
                end
            end
        end
    end
```

```
end

[rphi cphi]=size(Phi);
M=cphi;

% Calculate expansion coefficients
for ii=1:M
    parfor jj=1:Q % Parallel processing speeds up evaluation of ...
       expansion coefficients
        tt=Phi(jj,ii)*wcprod(jj);

        zj(jj)=cost(jj)*tt;
        zt(jj)=tdata(jj,Nnodes+2)*tt;

        z1(jj,:)=x1data(jj,:).*tt;
        z2(jj,:)=x2data(jj,:).*tt;
        z3(jj,:)=x3data(jj,:).*tt;
        zu(jj,:)=udata(jj,:).*tt;
        end

        % Expansion coefficients for states x_1, x_2, and x_3
        zhat1(ii,:)=sum(w1);
        zhat2(ii,:)=sum(w2);
        zhat3(ii,:)=sum(w3);
        % Expansion coefficients for control u
        zhatu(ii,:)=sum(wu);
        % Expansion coefficients for cost J
        zhatj(ii,:)=sum(wj);
        % Expansion coefficients for terminal time t_f
        zhatt(ii,:)=sum(wt);
end
```

1.4 Output Approximation Function

Step #4: Build the output approximation function using (3.2):

$$\mathbb{P}_N^{P_c} z \equiv z_N^{P_c}(\mathbf{p}) = \sum_{m=1}^{M} \hat{z}_m \Phi_m^c(\mathbf{p})$$

Listing A.4. Build output approximation function

```
% Construct polynomials and evaluate points of interest a_out, ...
   b_out, and c_out

% For Gaussian distributions...use Hermite polynomial basis ...
   functions (hermite function from Mathworks website)
```

```
m=1;
for jj=0:d1
    for kk=0:d2
        for ll=0:d3

          h11(m)=(1/((2)^(jj/2)))*hermite(jj,(aout-mu)/sqrt(2*dev...
              ^2))/sqrt(int1(jj+1));
          h22(m)=(1/((2)^(kk/2)))*hermite(kk,(bout-mu)/sqrt(2*dev...
              ^2))/sqrt(int2(kk+1));
          h33(m)=(1/((2)^(ll/2)))*hermite(ll,(cout-mu)/sqrt(2*dev...
              ^2))/sqrt(int3(ll+1));

            if jj+kk+ll<=P
                Phiout(m)=h11(m)*h22(m)*h33(m);
                m=m+1;
            end
        end
    end
end

% For uniform distributions...use Legendre polynomial basis ...
   functions (LegendrePoly function from Mathworks website)
m=1;
for jj=0:d1
    for kk=0:d2
        for ll=0:d3

          h11(m)=polyval(LegendrePoly(jj),aout)/sqrt(int1(jj+1));
          h22(m)=polyval(LegendrePoly(kk),bout)/sqrt(int2(kk+1));
          h33(m)=polyval(LegendrePoly(ll),cout)/sqrt(int3(ll+1));

            if jj+kk+ll<=P
                Phiout(m)=h11(m)*h22(m)*h33(m);
                m=m+1;
            end
        end
    end
end

parfor ii=1:M % Parallel processing speeds up evaluation of output...
    approximation
        z1(ii,:)=zhat1(ii,:).*Phiout(ii);
        z2(ii,:)=zhat2(ii,:).*Phiout(ii);
        z3(ii,:)=zhat3(ii,:).*Phiout(ii);
        zu(ii,:)=zhatu(ii,:).*Phiout(ii);
        zj(ii)=zhatj(ii)*Phiout(ii);
    zt(ii)=zhatt(ii)*Phiout(ii);
end

% Output approximation of states x_1, x_2, and x_3
zout1=sum(z1);
zout2=sum(z2);
```

```
zout3=sum(z3);
% Output approximation of control u
zoutu=sum(zu);
% Output approximation of cost J
zoutj=sum(zj);
% Output approximation of terminal time t_f
zoutt=sum(zt);
```

1.5 Statistics of the Solution

Step #5: Evaluate the statistics of the approximate solution:

- Expected value (3.3):

$$\mathbb{E}(\mathbf{z}(t)) \approx \mathbb{E}(\mathbf{z}_N^P(t)) = \int \left[\sum_{m=1}^{M} \hat{\mathbf{z}}_m(t) \Phi_m(\mathbf{p}) \right] \rho(\mathbf{p}) dp = \hat{\mathbf{z}}_1(t)$$

- Variance (3.4):

$$var\left[\mathbf{z}(t)\right] \approx \sum_{m=2}^{M} \left[\hat{\mathbf{z}}_m^2(t)\right]$$

- Covariance (3.5):

$$cov\left[z_i(t), z_j(t)\right] \approx \sum_{m=2}^{M} \left[\hat{z}_{i,m}(t) \hat{z}_{j,m}(t)\right]$$

Listing A.5. Evaluate statistics

```
% Expected values of hybrid algorithm solution
% Expected value of states x_1, x_2, and x_3
MeanX1gPC=zhat1(1,:);
MeanX2gPC=zhat2(1,:);
MeanX3gPC=zhat3(1,:);
% Expected value of control u
MeanUgPC=zhatu(1,:);
% Expected value of cost J
MeanJgPC=zhatj(1);
% Expected value of terminal time t_f
MeantfgPC=zhatt(1);

% Expected values of MCS solution
```

```
% Expected value of states x_1, x_2, and x_3
MeanX1MC=mean(x1MC);
MeanX2MC=mean(x2MC);
MeanX3MC=mean(x3MC);
% Expected value of control u
MeanUMC=mean(uMC);
% Expected value of cost J
MeanJMC=mean(costMC);
% Expected value of terminal time t_f
Meantf=mean(tMC(:,end));

% Variances of hybrid algorithm solution
% Variances of state estimates x_1, x_2, and x_3
VarX1gPC=sum(zhat1(2:end,:).^2);
VarX2gPC=sum(zhat2(2:end,:).^2);
VarX3gPC=sum(zhat3(2:end,:).^2);
% Variances of control estimate u
VarUgPC=sum(zhatu(2:end,:).^2);
% Variances of cost estimate J
VarJgPC=sum(zhatj(2:end).^2);
% Variances of terminal time estimate t_f
VartfgPC=sum(zhatt(2:end).^2);

% Variances of MCS solution
% Variances of state estimates x_1, x_2, and x_3
VarX1MC=var(x1MC);
VarX2MC=var(x2MC);
VarX3MC=var(x3MC);
% Variances of control estimate u
VarUMC=var(uMC);
% Variances of cost estimate J
VarJMC=var(costMC);
% Variances of terminal time estimate t_f
VartfMC=var(tMC(:,end));

% Covariance of hybrid algorithm solution
% Covariances between state estimates x_1, x_2, and x_3
CovX1X2gPC=sum(what1(2:end,:).*what2(2:end,:));
CovX1X3gPC=sum(what1(2:end,:).*what3(2:end,:));
CovX2X3gPC=sum(what2(2:end,:).*what3(2:end,:));
% Covariances between state estimates x_1, x_2, and x_3 and ...
   control estimate u
CovX1UgPC=sum(what1(2:end,:).*whatu(2:end,:));
CovX2UgPC=sum(what2(2:end,:).*whatu(2:end,:));
CovX3UgPC=sum(what3(2:end,:).*whatu(2:end,:));

% Covariances of MCS solution
for ii=1:QMC
    for jj=1:numtime
        c1(ii,jj)=x1MC(ii,jj)-MeanX1MC(jj);
        c2(ii,jj)=x2MC(ii,jj)-MeanX2MC(jj);
        c3(ii,jj)=x3MC(ii,jj)-MeanX3MC(jj);
```

```
        c4(ii,jj)=uMC(ii,jj)-MeanUMC(jj);
    end
end

% Covariances between state estimates x_1, x_2, and x_3
CovX1X2MC=mean(c1.*c2);
CovX1X3MC=mean(c1.*c3);
CovX2X3MC=mean(c2.*c3);
% Covariances between state estimates x_1, x_2, and x_3 and ...
   control estimate u
CovX1UMC=mean(c1.*c4);
CovX2UMC=mean(c2.*c4);
CovX3UMC=mean(c3.*c4);
```

The code in this appendix forms the basis for a main MATLAB®script that was used that calls GPOPS (Appendix B) and MCS (Appendix C) routines where indicated.

Appendix B. GPOPS Set-Up Code

The GPOPS set-up files used to produce results presented in Chapter V are shown in this appendix. This representative code shows the GPOPS configuration used to produce solutions at the collocation points (hybrid algorithm) and at the randomly sampled points (MCS). See [72] for detailed descriptions about these files.

2.1 GPOPS Main Script: MyProblemMain.m

Listing B.1. GPOPS main script

```
 clc
clear setup limits guess

global qc ac bc cc Nnodes constants t PK xinit xfinal

x10 = 0;
x1f = 25;

x20 = xinit(2);
x2f = xfinal(2);

x30 = atan(x2f/x1f);
x3f = [];

x1min = 0;
x1max = 50;

x2min = x1min;
x2max = x1max;

[xgrid,ygrid,PK]=killrings(constants,ac,bc,cc,x1min,x1max,x2min,...
   x2max);

constants.tmin=0;
constants.tmax=1;

x3min = -2*pi;
x3max = 2*pi;

tinit=0;
tfinal=[];
```

```
param_min = [];
param_max = [];
path_min = [];
path_max = [];

event_min = [x10; x20; x30; x1f; x2f];
event_max = [x10; x20; x30; x1f; x2f];

duration_min = constants.tmin;
duration_max = constants.tmax;

iphase = 1;
limits(iphase).nodes = Nnodes;  %60;
limits(iphase).time.min = [0 constants.tmin];
limits(iphase).time.max = [0 constants.tmax];
limits(iphase).state.min(1,:) = [x1min x1min x1min];
limits(iphase).state.max(1,:) = [x1max x1max x1max];
limits(iphase).state.min(2,:) = [x2min x2min x2min];
limits(iphase).state.max(2,:) = [x2max x2max x2max];
limits(iphase).state.min(3,:) = [x3min x3min x3min];
limits(iphase).state.max(3,:) = [x3max x3max x3max];


limits(iphase).control.min    = -470/(11326/6076.11549);
limits(iphase).control.max    = 470/(113266076.11549);

limits(iphase).parameter.min  = param_min;
limits(iphase).parameter.max  = param_max;

limits(iphase).path.min       = path_min;
limits(iphase).path.max       = path_max;

limits(iphase).event.min      = event_min;
limits(iphase).event.max      = event_max;

limits(iphase).duration.min    = [];
limits(iphase).duration.max    = [];

guess(iphase).time             = [0; 0];

guess(iphase).state(:,1)      = [x10; x1f];
guess(iphase).state(:,2)      = [x20; x2f];
guess(iphase).state(:,3)      = [x30; x3f];

guess(iphase).control         = [0; 0];
guess(iphase).parameter       = [];

clear x10 x20 x30 x1f x2f x3f x1min x1max x2min x2max x3min x3max
clear param_min param_max path_min path_max event_min event_max
clear duration_min duration_max iphase

setup.name  = 'My-Problem';
```

```
setup.funcs.cost = 'MyProblemCost';
setup.funcs.dae = 'MyProblemDae';
setup.funcs.event = 'MyProblemEvent';
setup.limits = limits;
setup.guess = guess;
setup.linkages = [];
setup.derivatives = 'automatic';
setup.direction = 'decreasing'; %increasing
setup.autoscale = 'on';

output = gpops(setup);
solution = output.solution;

x1out=solution.state(:,1);
x2out=solution.state(:,2);
x3out=solution.state(:,3);
p1out=solution.costate(:,1);
p2out=solution.costate(:,2);
p3out=solution.costate(:,3);
uout=solution.control;
tout=solution.time;
Hamout=solution.Hamiltonian;
costout=solution.Mayer_cost+solution.Lagrange_cost;

%-------------------------------------
% END: script MyProblemMain.m
%-------------------------------------
```

2.2 GPOPS Cost Function: MyProblemCost.m

Listing B.2. GPOPS cost function

```
%-------------------------------------
% BEGIN: function MyProblemCost.m
%-------------------------------------
function [Mayer,Lagrange,DMayer,DLagrange]=MyProblemCost(sol)
global constants ac bc cc PK t Jscl

t0 = sol.initial.time;
x0 = sol.initial.state;
tf = sol.terminal.time;
xf = sol.terminal.state;
t  = sol.time;
x  = sol.state;
u  = sol.control;
p  = sol.parameter;

Mayer = 0.00001*tf;

% Sampled values (collocation or random) are passed into this ...
   funciton as ac, bc, cc
```

```
c1x=constants.c1x1+ac; c1y=constants.c1x2+ac;
c2x=constants.c2x1+bc; c2y=constants.c2x2+bc;
c3x=constants.c3x1+cc; c3y=constants.c3x2+cc;

% Threat #1
f11=(1/(2*pi*constants.sigt1x1*constants.sigt1x2*sqrt(1-constants....
   rho1^2)));
f12=1/(2*(1-constants.rho1^2));
t11=(x(:,1)-c1x).^2./(constants.sigt1x1^2);
t12=(x(:,2)-c1y).^2./(constants.sigt1x2^2);
t13=(2*constants.rho1*(x(:,1)-c1x).*(x(:,2)-c1y))./(constants....
   sigt1x1*constants.sigt1x2);

PK1=f11.*exp(-f12.*(t11+t12-t13));

% Threat #2
f21=(1/(2*pi*constants.sigt2x1*constants.sigt2x2*sqrt(1-constants....
   rho2^2)));
f22=1/(2*(1-constants.rho2^2));
t21=(x(:,1)-c2x).^2./(constants.sigt2x1^2);
t22=(x(:,2)-c2y).^2./(constants.sigt2x2^2);
t23=(2*constants.rho2*(x(:,1)-c2x).*(x(:,2)-c2y))./(constants....
   sigt2x1*constants.sigt2x2);

PK2=f21.*exp(-f22.*(t21+t22-t23));

% Threat #3
f31=(1/(2*pi*constants.sigt3x1*constants.sigt3x2*sqrt(1-constants....
   rho3^2)));
f32=1/(2*(1-constants.rho3^2));
t31=(x(:,1)-c3x).^2./(constants.sigt3x1^2);
t32=(x(:,2)-c3y).^2./(constants.sigt3x2^2);
t33=(2*constants.rho3*(x(:,1)-c3x).*(x(:,2)-c3y))./(constants....
   sigt3x1*constants.sigt3x2);

PK3=f31.*exp(-f32.*(t31+t32-t33));

Lagrange = (PK1+PK2+PK3)/Jscl;

%-------------------------------------
% END: function MyProblemCost.m
%-------------------------------------
```

2.3 GPOPS Differential Algebraic Equations Function: MyProblemDAE.m

Listing B.3. GPOPS DAE function

```
%-------------------------------------
% BEGIN: function MyProblemDae.m
%-------------------------------------
function dae = MyProblemDae(sol);
```

```
global ac bc constants

t = sol.time;
x = sol.state;
u = sol.control;


x1dot = constants.v.*cos(x(:,3));
x2dot = constants.v.*sin(x(:,3));
x3dot = u;

path = [];
dae = [x1dot x2dot x3dot path];

%-------------------------------------
% END: function MyProblemDae.m
%-------------------------------------
```

2.4 GPOPS Event Function: MyProblemEvent.m

Listing B.4. GPOPS Event Function

```
%-------------------------------------
% BEGIN: function MyProblemEvent.m
%-------------------------------------
function event = MyProblemEvent(sol);
global constants

t0 = sol.initial.time;
x0 = sol.initial.state;
tf = sol.terminal.time;
xf = sol.terminal.state;

event = [x0(1,:); x0(2,:); x0(3,:); xf(1,:); xf(2,:)];

%-------------------------------------
% END: function MyProblemEvent.m
%-------------------------------------
```

Appendix C. Monte-Carlo Simulation Code

Selected lines of MATLAB®code are given in this appendix to show how the MCS results were generated for comparison with the hybrid GPM-gPC results in Chapter V. A similar version was used to generate the MCS results presented in Chapter IV.

Listing C.1. MCS code

```
% Initialize variables for data storage
% States x_1, x_2, and x_3
x1MC=[]; x2MC=[]; x3MC=[];
% Control u
uMC=[];
% Cost J
costMC=[];
% Time
tMC=[];
% Point counter for plotting
ptcount=[];

% Initialize variables for convergence checks
meanOldX1=zeros(1,Nnodes+2); meanOldX2=zeros(1,Nnodes+2); ...
   meanOldX3=ones(1,Nnodes+2);

convStop=0;
chkPt=500;
numPts= 100000;

% Generate arrays of random samples for a, b, c
randsampG=randn(numPts,3);  % Gaussian
randsampU=rand(numPts,3);  % Uniform

% Initialize counter
ii = 1;

while convStop==0 && ii<=numPts

        % For Gaussian distributions...scale points into specified...
            distribution
        ac=mu+dev*randsampG(ii,1);
        bc=mu+dev*randsampG(ii,2);
        cc=mu+dev*randsampG(ii,3);

        % For uniform distributions...scale points into specified ...
           domain [lb,ub]
```

```
        ac=(lb)+(((ub)-(lb))*randsampU(ii,1));
        bc=(lb)+(((ub)-(lb))*randsampU(ii,2));
        cc=(lb)+(((ub)-(lb))*randsampU(ii,3));

        % Call GPOPS main script
    MyProblemMain;

    % Save GPOPS outputs
        % States x_1, x_2, and x_3
        x1MC=[x1MC;x1out'];
        x2MC=[x2MC;x2out'];
        x3MC=[x3MC;x3out'];

    % Control u
    uMC=[uMC;uout'];

        % Time
    tMC=[tMC;tout'];

        % Cost J
    costMC=[costMC;costout];

        % Point counter
    ptcount=[ptcount;ii];

    % Check convergence
        if mod(ii,500)==0

                convChkX1=(abs(meanOldX1-mean(x1MC)));
                convChkX2=(abs(meanOldX2-mean(x2MC)));
                convChkX3=(abs(meanOldX3-mean(x3MC)));

            if max(convChkX1)<=0.0001 && max(convChkX2)<=0.0001 &&...
                max(convChkX3)<=0.0001 && ii<numPts
                convStop=1;
            end

            % Save data for next convergence check
            meanOldX1=mean(x1MC);
            meanOldX2=mean(x2MC);
            meanOldX3=mean(x3MC);
        end

    ii=ii+1;
end

    % Save data to .mat files
    % States x_1, x_2, and x_3
    save x1MC x1MC
    save x2MC x2MC
    save x3MC x3MC
```

```
% Control u
save uMC uMC

    % Cost J
save costMC costMC

    % Time
save tMC tMC


save ptcount ptcount
```

Bibliography

1. "DIDO, A Pseudospectral Method for Solving Optimal Control Problems", 2006. MIT Online Course Lecture (16.323 Lecture 17).

2. Abate, Gregg. "Micro Aerial Vehicle (MAV) Research at AFRL". PowerPoint Presentation, June 2008. Presentation given at the International Symposium on Unmanned Aerial Vehicles.

3. Babuska, Ivo and Panagiotis Chatzipantelidis. "On solving elliptic stochastic partial differential equations". *Computer Methods in Applied Mechanics and Engineering*, 191(37-38):4093–4122, 2002.

4. Babuska, Ivo, Fabio Nobile, and Ra'ul Tempone. "A Stochastic Collocation Method for Elliptic Partial Differential Equations with Random Input Data". *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007.

5. Babuska, Ivo, Raúl Tempone, and Georgios E. Zouraris. "Galerkin Finite Element Approximations of Stochastic Elliptic Partial Differential Equations". *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

6. Becerra, Victor M. "Solving Optimal Control Problems with State Constraints Using Nonlinear Programming and Simulation Tools". *IEEE Transactions on Education*, 47(3):377–384, 2004.

7. Benson, David. *A Gauss Pseudospectral Transcription for Optimal Control*. Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

8. Benson, David A., Geoffrey T. Huntington, Tom P. Thorvaldsen, and Anil V. Rao. "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method". *Journal of Guidance, Control, and Dynamics*, 29(6):1435–1440, 2006.

9. Betts, John T. "Survey of Numerical Methods for Trajectory Optimization". *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

10. Borzi, A. and G. von Winckel. "SPQUAD: Computes the sparse grid quadrature abscissae and weights on an orthotope/hyperrectangle using the Clenshaw-Curtis rule", 2008. URL `http://www.ing.unisannio.it/borzi/spquad.m`.

11. Borzi, A. and G. von Winckel. "Multigrid Methods and Sparse-Grid Collocation Techniques for Parabolic Optimal Control Problems with Random Coefficients". *SIAM Journal on Scientific Computing*, 31(3):2172–2192, 2009.

12. Boyce, William E. and Richard C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. Wiley, New York, 1992. ISBN 0-471-50998-1.

13. Boyd, John P. "Multipole Expansions and Pseudospectral Cardinal Functions: A New Generalization of the Fast Fourier Transform". *Journal of Computational Physics*, 103(1):184–186, 1992.

14. Boyd, John P. *Chebyshev and Fourier Spectral Methods*. Dover, New York, 2001. ISBN 0-486-41183-4.

15. Brown, Robert G. and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. Wiley, New York, 1996. ISBN 978-0471128397.

16. Bryson, A. E. "Optimal control-1950 to 1985". *IEEE Control Systems Magazine*, 16(3):26–33, 1996.

17. Bryson, Arthur E. *Dynamic Optimization*. Addison-Wesley, Menlo Park, CA, 1999. ISBN 978-0201597905.

18. Budhiraja, Amarjit, Lingji Chen, and Chihoon Lee. "A survey of numerical methods for nonlinear filtering problems". *Physica D: Nonlinear Phenomena*, 230(1-2):27–36, 2007.

19. Burden, Richard L. and J. Douglas Faires. *Numerical Analysis*. Thompson Brooks/Cole, Belmont, CA, 2005. ISBN 978-0-534-39200-0.

20. Burl, Jeffrey B. *Linear Optimal Control*. Prentice Hall, Menlo Park, CA, 1998. ISBN 978-0201808681.

21. Burns, John A. "Notes on Calculus of Variations with Modern Applications to Control Theory, Numerical Methods, and Differential Equations", 2009.

22. Crespo, Luis G. and Jian Q. Sun. *Stochastic Optimal Control Via Bellman's Principle*. Technical Report NAS 1.26:212419, NASA/CR-2003-212419, NIA Report No. 2003-04, 2003.

23. Deb, Manas K., Ivo M. Babuska, and J. Tinsley Oden. "Solution of stochastic partial differential equations using Galerkin finite element techniques". *Computer Methods in Applied Mechanics and Engineering*, 190(48):6359–6372, 2001.

24. Dormand, J. R. and P. J. Prince. "A family of embedded Runge-Kutta formulae". *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

25. Eldred, M. S. "Recent Advances in Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Analysis and Design". *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. May 2009.

26. Elnagar, G., M. A. Kazemi, and M. Razzaghi. "The pseudospectral Legendre method for discretizing optimal control problems". *IEEE Transactions on Automatic Control*, 40(10):1793–1796, 1995.

27. Fahroo, Fariba and I. Michael Ross. "Trajectory Optimization by Indirect Spectral Collocation Methods". *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, 123–129. August 2000.

28. Fahroo, Fariba and I. Michael Ross. "Costate Estimation by a Legendre Pseudospectral Method". *Journal of Guidance, Control, and Dynamics*, 24(2):270–277, 2001.

29. Fahroo, Fariba and I. Michael Ross. "Direct Trajectory Optimization by Chebyshev Pseudospectral Method". *Journal of Guidance, Control, and Dynamics*, 25(1):160–166, 2002.

30. Fahroo, Fariba and I. Michael Ross. "Advances in Pseudospectral Methods for Optimal Control". *AIAA Guidance, Navigation and Control Conference and Exhibit*. August 2008.

31. Feng, Xiangbo, Kenneth A. Loparo, and Yuguang Fang. "Optimal State Estimation for Stochastic Sytstems: An Information Theoretic Approach". *IEEE Transactions on Automatic Control*, 42(6):771–785, 1997.

32. Ferguson, James. "A Brief Survey of the History of the Calculus of Variations and its Applications", 2004. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:math/0402357`.

33. Fornberg, Bengt. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, New York, 1996. ISBN 0-521-49582-2.

34. Frauenfelder, Philipp, Christoph Schwab, and Radu Alexandru Todor. "Finite elements for elliptic problems with stochastic coefficients". *Computer Methods in Applied Mechanics and Engineering*, 194(2-5):205–228, 2005.

35. Garg, Divya, Michael Patterson, Anil V. Rao, William W. Hager, David A. Benson, and Geoffrey T. Huntington. "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods". *Automatica*, 46(11):1843–1851, 2010.

36. Garg, Divya, Michael A. Patterson, Camila Francolin, Christopher L. Darby, Geoffrey T. Huntington, William W. Hager, and Anil V. Rao. "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Pseudospectral Method". *Computational Optimization and Applications*, 49(2):335–358, 2011.

37. Garg, Divya, Michael A. Patterson, William W. Hager, Anil V. Rao, David Benson, and Geoffrey T. Huntington. "An Overview of Three Pseudospectral Methods for the Numerical Solution of Optimal Control Problems". *AAS/AIAA Astrodynamics Specialist Conference*. August 2009.

38. Gerstner, Thomas and Michael Griebel. "Numerical Integration Using Sparse Grids". *Numerical Algorithms*, 18(3-4):209–234, 1998.

39. Ghanem, R. G. and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer-Verlag, New York, 1991. ISBN 0-387-97456-3.

40. Ghanem, Roger. "Ingredients for a general purpose stochastic finite elements implementation". *Computer Methods in Applied Mechanics and Engineering*, 168(1-4):19–34, 1999.

41. Ghanem, Roger. "Stochastic Finite Elements With Multiple Random Non-Gaussian Properties." *Journal of Engineering Mechanics*, 125(1):26–40, 1999.

42. Gill, Philip E., Walter Murray, and Michael A. Saunders. "User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming", 2008. URL `http://www.cam.ucsd.edu/~peg/papers/sndoc7.pdf`.

43. Girardeau, Pierre. "A comparison of sample-based Stochastic Optimal Control methods". *ArXiv e-prints*, 2010. URL `http://adsabs.harvard.edu/abs/2010arXiv1002.1812G`.

44. Gong, Qi, Wei Kang, N.S. Bedrossian, F. Fahroo, P. Sekhavat, and K. Bollino. "Pseudospectral Optimal Control for Military and Industrial Applications". *46th IEEE Conference on Decision and Control*, 4128 –4142. December 2007. ISSN 0191-2216.

45. Gottlieb, David and Dongbin Xiu. "Galerkin Method for Wave Equations with Uncertain Coefficients". *Communications in Computational Physics*, 3(2):505–518, 2008.

46. Grigoriu, Mircea. "Simulation of stationary non-Gaussian translation processes". *Journal of Engineering Mechanics*, 124(2):121–126, 1998.

47. Hanc, Jozef. "The Original Euler's Calculus-of-Variations Method: Key to Lagrangian Mechanics for Beginners". *European Journal of Physics*, Submitted 2004. URL `http://lcib.rutgers.edu/~james/CalcOfVar.pdf`.

48. Harmon, Frederick G. "Air Force Institute of Technology Class Notes for MECH 622", 2009. Personal notes and class materials from AFIT MECH 622.

49. Holsapple, Raymond, Ram Venkataraman, and David Doman. "A modified simple shooting method for solving two-point boundary value problems". *2003 IEEE Aerospace Conference Proceedings*, volume 6, 2783–2790. 2003.

50. Huntington, Geoffrey T. *Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control Problems*. Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

51. Kalman, Rudolph E. "A New Approach to Linear Filtering and Prediction Problems". *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35–45, 1960.

52. Kalman, Rudolph E. and Richard S. Bucy. "New Results in Linear Filtering and Prediction Theory". *Transactions of the ASME-Journal of Basic Engineering*, 83:95–108, 1961.

53. Kincaid, David and Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. American Mathematical Society, Pacific Grove, CA, 2002. ISBN 978-0-8218-4788-6.

54. Kirk, Donald E. *Optimal Control Theory: An Introduction*. Dover Publications, Mineola, NY, April 2004. ISBN 0486434842.

55. Kreyszig, Erwin. "On the calculus of variations and its major influences on the mathematics of the first half of our century I". *The American Mathematical Monthly*, 101(7):674–678, 1994.

56. Kushner, Harold J. "Numerical Methods for Stochastic Control Problems in Continuous Time". *SIAM Journal of Control and Optimization*, 28(5):999–1048, 1990.

57. Le Maître, Olivier P., Omar M. Knio, Habib N. Najm, and Roger G. Ghanem. "A Stochastic Projection Method for Fluid Flow: I. Basic Formulation". *Journal of Computational Physics*, 173(2):481–511, 2001.

58. Le Maître, Olivier P., Matthew T. Reagan, Habib N. Najm, Roger G. Ghanem, and Omar M. Knio. "A Stochastic Projection Method for Fluid Flow: Random Process". *Journal of Computational Physics*, 181(1):9–44, 2002.

59. Lewis, L.R., I.M. Ross, and Qi Gong. "Pseudospectral motion planning techniques for autonomous obstacle avoidance". *46th IEEE Conference on Decision and Control*, 5997 –6002. December 2007. ISSN 0191-2216.

60. Liu, Yunyun. *The Pseudospectral Chebyshev Method for Two-Point Boundary Value Problems*. Master's thesis, Simon Fraser University, 1992.

61. Luenberger, David G. and Yinyu Ye. *Linear and Nonlinear Programming*. Springer, New York, 2008. ISBN 978-0-387-74502-2.

62. Mathelin, Lionel and M. Yousuff Hussaini. *A Stochastic Collocation Algorithm for Uncertainty Analysis*. Technical Report NASA/CR-2003-212153, 2003.

63. Maybeck, Peter S. *Stochastic Models, Estimation, and Control Volume 1*. Academic Press, New York, 1979. ISBN 0-12-480701-1.

64. McEneaney, William M. "Dynamics and Control". PowerPoint Presentation, March 2009. Briefing given on 4 Mar 2009 at AFOSR Spring Review.

65. Moler, Cleve B. *Numerical Computing with MATLAB*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2004. ISBN 978-0-898715-60-6.

66. Nobile, Fabio, Raul Tempone, and Clayton Webster. *The analysis of a sparse grid stochastic collocation method for partial differential equations with high-dimensional random input data*. Technical Report SAND2007-8093, December 2007.

67. Novak, Kyle. "Numerical Methods for Partial Differential Equations - AFIT MATH 676 Course Notes", 2009.

68. Ogundare, B. S. "On the Pseudo-Spectral Method of Solving Linear Ordinary Differential Equations". *Journal of Mathematics and Statistics*, 5(2):136–140, 2009.

69. Pontryagin, L. S., V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Wiley, New York, 1962. ISBN 978-2881240775.

70. Pulch, Roland. "Polynomial Chaos for Boundary Value Problems of Dynamical Systems". *Applied Numerical Mathematics (to appear)*, 2009.

71. Rao, Anil V. "A Survey of Numerical Methods for Optimal Control". *AAS/AIAA Astrodynamics Specialist Conference*. August 2009.

72. Rao, Anil V., David Benson, Geoffrey T. Huntington, Camila Francolin, Christopher L. Darby, Michael Patterson, and Ilyssa Sanders. "User's Manual for GPOPS Version 3.0: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using Pseudospectral Methods", 2010.

73. Rogers, L. C. G. "Pathwise Stochastic Optimal Control". *SIAM Journal on Control and Optimization*, 46(3):1116–1132, 2007.

74. Russell, Stuart and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2002. ISBN 978-0137903955.

75. Siouris, George M. *An Engineering Approach to Optimal Control and Estimation Theory*. Wiley, New York, 1996. ISBN 0-471-12126-6.

76. Slotine, Jean-Jacques E. and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, Upper Saddle River, NJ, 1991. ISBN 0-13-040890-5.

77. Smolyak, S. A. "Quadrature and Interpolation Formulas for Tensor Products of Certain Classes". *Soviet Mathematics - Doklady*, 4:240–243, 1963.

78. von Stryk, O. and R. Bulirsch. "Direct and Indirect Methods for Trajectory Optimization". *Annals of Operations Research*, 37(1):357–373, 1992.

79. Sussmann, H. J. and J. C. Willems. "300 years of optimal control: from the brachystochrone to the maximum principle". *IEEE Control Systems Magazine*, 17(3):32–44, 1997.

80. Tao, Jun, Xuan Zeng, Wei Cai, Yangfeng Su, Dian Zhou, and Charles Chiang. "Stochastic Sparse-grid Collocation Algorithm (SSCA) for Periodic Steady-State Analysis of Nonlinear System with Process Variations". *Proceedings of the 2007 Asia and South Pacific Design Automation Conference*, 474–479. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 1-4244-0629-3.

81. Trefethen, Lloyd N. *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics, Philadelphia, 2001. ISBN 978-0-898714-65-4.

82. Wan, Xiaoliang and George Em Karniadakis. "Multi-Element Generalized Polynomial Chaos for Arbitrary Probability Measures". *SIAM Journal on Scientific Computing*, 28(3):901–928, 2006.

83. Wiener, Norbert. "The Homogeneous Chaos". *American Journal of Mathematics*, 60(4):897–936, 1938.

84. Wiener, Norbert. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. Wiley, New York, 1949.

85. Willems, Jan C. "1696: the Birth of Optimal Control". *Proceedings of the 35th IEEE Conference on Decision and Control*, 1586–1587. December 1996.

86. Xiu, Dongbin. "Efficient Collocational Approach for Parametric Uncertainty Analysis". *Communications in Computational Physics*, 2(2):293–309, 2007.

87. Xiu, Dongbin. "Fast Numerical Methods for Stochastic Computations: A Review". *Communications in Computational Physics*, 5(2-4):242–272, 2009.

88. Xiu, Dongbin. *Numerical Methods for Stochastic Computations*. Princeton University Press, Princeton, NJ, 2010. ISBN 978-0-691-14212-8.

89. Xiu, Dongbin and Jan S. Hesthaven. "High-Order Collocation Methods for Differential Equations with Random Inputs". *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2006.

90. Xiu, Dongbin and George Em Karniadakis. "Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos". *Computer Methods in Applied Mechanics and Engineering*, 191(43):4927–4948, 2002.

91. Xiu, Dongbin and George Em Karniadakis. "The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations". *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.

92. Xiu, Dongbin and George Em Karniadakis. "Modeling uncertainty in flow simulations via generalized polynomial chaos". *Journal of Computational Physics*, 187(1):137–167, 2003.

93. Xiu, Dongbin and George Em Karniadakis. "A new stochastic approach to transient heat conduction modeling with uncertainty". *International Journal of Heat and Mass Transfer*, 46(24):4681–4693, 2003.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22–03–2012	Dissertation	Aug 2008 — Mar 2012

4. TITLE AND SUBTITLE

Hybrid Solution of Stochastic Optimal Control Problems Using Gauss Pseudospectral Method and Generalized Polynomial Chaos Algorithms

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Gerald C. Cottrill, Lt Col, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/DS/ENY/12-11

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

USSTRATCOM Joint Functional Component Command for Global Strike
Paul A. Blue, Branch Chief, JF523
901 SAC Blvd, Suite BB18
Offutt AFB, NE 68113-6300
(402) 294-1497; paul.blue@stratcom.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

USSTRATCOM/JFCC-GS JF52

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Two numerical methods, Gauss Pseudospectral Method and Generalized Polynomial Chaos Algorithm, were combined to form a hybrid algorithm for solving nonlinear optimal control and optimal path planning problems with uncertain parameters. The algorithm was applied to two concept demonstration problems: a nonlinear optimal control problem with multiplicative uncertain elements and a mission planning problem sponsored by USSTRATCOM. The mission planning scenario was constructed to find the path that minimizes the probability of being killed by lethal threats whose locations are uncertain to statistically quantify the effects those uncertainties have on the flight path solution, and to use the statistical properties to estimate the probability that the vehicle will be killed during mission execution. The results demonstrated that the method is able to effectively characterize how the optimal solution changes with uncertainty and that the results can be presented in a form that can be used by mission planners and aircrews to assess risks associated with a mission profile.

15. SUBJECT TERMS

Optimal Control, Stochastic Optimal Control, Generalized Polynomial Chaos, Gauss Pseudospectral Method

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col Frederick Harmon
U	U	U	UU	189	19b. TELEPHONE NUMBER *(include area code)* (937)255–3636x7478, frederick.harmon@afit.edu