THE THEORY OF FUNCTIONAL CONNECTIONS
A JOURNEY FROM THEORY TO APPLICATION

A Dissertation

by

HUNTER REED JOHNSTON

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Daniele Mortari
Committee Members,	John E. Hurtado
	Srinivas Vadali
	Yalchin Efendiev
Head of Department,	Srinivas Vadali

August 2021

Major Subject: Aerospace Engineering

ABSTRACT

The Theory of Functional Connections (TFC) is a general methodology for functional interpolation that can embed a set of user-specified linear constraints. The functionals derived from this method, called *constrained expressions*, analytically satisfy the imposed constraints and can be leveraged to transform constrained optimization problems to unconstrained ones. By simplifying the optimization problem, this technique has been shown to produce a numerical scheme that is faster, more accurate, and robust to poor initialization. The content of this dissertation details the complete development of the Theory of Functional Connections. First, the seminal paper on the Theory of Functional Connections is discussed and motivates the discovery of a more general formulation of the constrained expressions. Leveraging this formulation, a rigorous structure of the constrained expression is produced with associated mathematical definitions, claims, and proofs. Furthermore, the second part of this dissertation explains how this technique can be used to solve ordinary differential equations providing a wide variety of examples compared to the state-of-the-art. The final part of this work focuses on unitizing the techniques and algorithms produced in the prior sections to explore the feasibility of using the Theory of Functional Connections to solve real-time optimal control problems, namely optimal landing problems.

DEDICATION

To my mother and father.
And to the friends (C, L, & M) who have been there from the beginning,
and those who I've met along the way.

All things inevitably come to an end.
Some day the machine stops running.
We can share paths for a while, but
ultimately we all have our own
separate destinations.

— Unravel, *ColdWood Interactive*

ACKNOWLEDGMENTS

The path to completing this document involved not just numbers and equations but loving and caring human beings — family, friends, teachers, and mentors. Although I encountered many roadblocks, dead ends, and unfavorable terrain, you, knowingly or unknowingly, have propelled me. While I could easily fill this page with names, I restrain over the fear of forgetting just one. However, to those to who I am referring, you know who You are ...

Thank You,
and I love You.

Regardless, a few people were fundamental to my education and the completion of this document, and I would like to identify them by name specifically.

First, Dr. Daniele Mortari, my advisor and friend. Thank you for taking a chance to bring me in as one of your graduate students. My four years at Texas A&M were memorable, to say the least, and I will cherish the brainstorming session we've had, ALL of the meals we've shared, and our conversations about literature, life, and philosophy.

Second, my labmates and true friends, (soon to be Dr.) Carl Leake and Dr. Stoian Borissov. You both have given me unmatched support in dealing with the rough terrain of graduate school and graduate student life. Thanks not only for challenging me and providing unmatched feedback, but for also pulling me away from graduate life and distracting me with good food, good company, loud drums, and above all, many MANY "coffees."

Next, my colleagues from the University of Arizona, (also, soon to be Dr.) Enrico Schiassi and Dr. Roberto Furfaro. Thank you for your amazing collaboration on many projects and for welcoming me into your research group during my month-long visit to Arizona. Specifically, thank you Enrico for video chatting with me to watch F1 races throughout this crazy year of COVID.

Additionally, my committee members Drs. John E. Hurtado, Rao Vadali, and Yalchin Efendiev. Of the two I've been fortunate enough to take classes with, I would like to thank for their inspiration and guidance; your classes are two of my most memorable ones from my time at Texas A&M. Additionally, I thank Dr. Efendiev for the many Saturday mornings he spent with the TFC research group and his unmatched guidance and feedback.

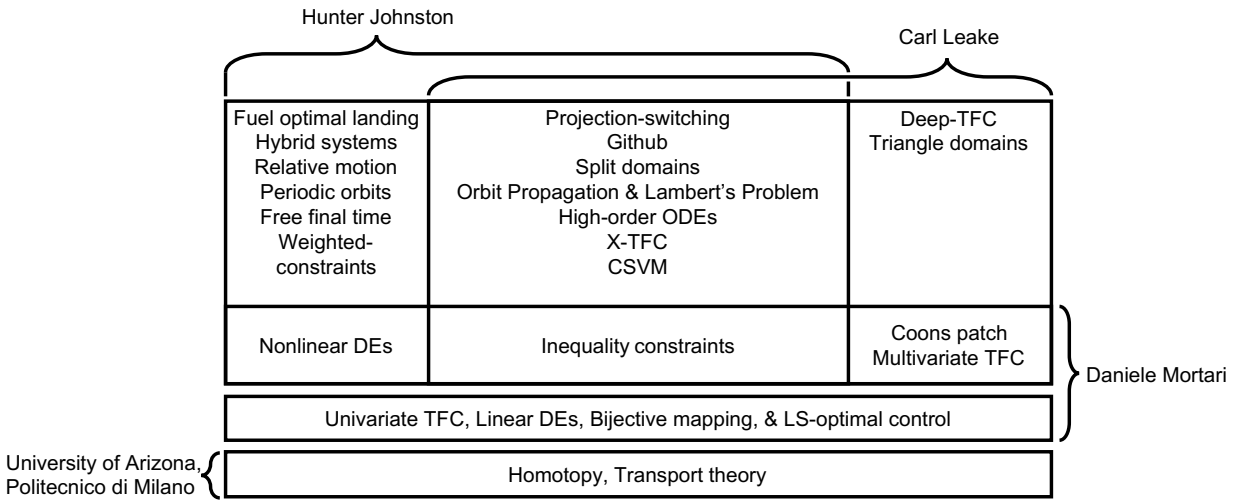
Lastly, I would like to thank my NASA/NSTRF collaborators, Drs. Chris D'Souza and Martin Lo (who was also my Visiting Technologist Experience host at JPL). Thank you for your support and guidance in both research and my career goals. Additionally, thank you for our long conversations when it seemed the world was falling down around us.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Daniele Mortari (advisor) and John E. Hurtado and Srinivas Vadali of the Department of Aerospace Engineering, and Yalchin Efendiev of the Department of Mathematics.

The Theory of Functional Connections was collaboratively developed by Daniele Mortari (advisor), Carl Leake (Ph.D. candidate), and Hunter Johnston (author/Ph.D. candidate). To clarify the major contributions of each, the following figure is included.



All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by teaching and research assistantships from Texas A&M University from August 2017 - August 2019, and by the NASA Space Technology Research Fellowship, Johnston [NSTRF 2019] Grant #: 80NSSC19K1149, from August 2019 - August 2021.

NOMENCLATURE

ELM	Extreme Learning Machine
FEM	Finite Element Method
LS-SVM	Least-Squares Support Vector Machine
NN	Neural Network
NSTRF	NASA Space Technology Research Fellowship
ODE	Ordinary differential equation
PDE	Partial differential equation
PMP	Pontryagin Minimum Principle
SVM	Support Vector Machine
TFC	Theory of Function Connections
TPBVP	two-point boundary-value problem
X-TFC	Extreme Theory of Functional Connections
c	Slope in the linear map for the independent variable that maps the basis function domain to the problem domain.
b	The square-root of the slope in the linear map for the independent variable that maps the basis function domain to the problem domain. $b^2 = c$
$\mathfrak{C}_i$	Constraint operator for the i-th constraint
δ_{ij}	Kronecker delta
$g(x)$	Free function $\mathbb{R} \mapsto \mathbb{R}$. Note that a superscript may be used to denote the free function for a specific dependent variable, e.g., $g^u(x)$ is the free function for the dependent variable u.
$\mathbb{J}$	Jacobian matrix of the loss vector function $\mathbb{L}$
$\kappa_i(x)$	Portion of the i-th constraint of the independent variable that does not contain the dependent variable.
$\mathbb{L}$	Loss vector function $\mathbb{R}^m \mapsto \mathbb{R}^n$

$\rho_i(x, g(x))$	Projection functional for the i-th constraint on the independent variable.
$\phi_i(x)$	Switching function for the i-th constraint on the independent variable.
$\mathbb{R}$	Field of real numbers
$\mathbb{S}_{ij}$	Support matrix
τ	Alternative definition of the basis function independent variable. Note, this is used when z is used as an independent variable.
$\mathbb{Z}^+$	Set of positive integers
z	Basis function independent variable. Note, this is replaced with τ in some cases.
$\mathbb{1}(x, x_1)$	Heaviside function, $\mathbb{R} \mapsto \mathbb{R}$
$\mathbb{1}_0(x)$	Heaviside function where $x_1 = 0$

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

LIST OF EXAMPLES

EXAMPLE Page

1. INTRODUCTION

The topics presented in this dissertation can be split into three distinct areas which flow from the general formulation of the Theory of Functional Connections (TFC) (Chapter 2 and Chapter 3) to its application to the solution of differential equations (Chapter 4) and finally leveraging the method to solve optimal control problems (Chapter 5), namely the energy-optimal landing (Chapter 6) and fuel-optimal landing (Chapter 7) problems. Ultimately, the goal of this work is to develop a fast, accurate, and robust numerical system to solve problems relevant in aerospace engineering; however, the development of TFC and its initial application to differential equations are vital stepping stones in this effort since each chapter is heavily reliant on those coming before.

Since this work covers the full journey from the initial theory first published by Mortari [3] in 2017 to applications in aerospace engineering, I have opted to provide multiple literature reviews directly before the chapters they pertain to. For example, Chapter 2 provides an overview of the mathematical concept of interpolation and how they have been utilized. Similarly, the beginning of Chapter 4 reviews current numerical techniques available to solve ordinary differential equations, and Chapter 5 provides background on the techniques to solve optimal control problems.

The following sections of this chapter provide a summary of the work in this dissertation. This is provided to give the reader insight into the structure of the document and highlight the new contributions made to current literature.

Part 1 — Theory

Chapter 2: An Introduction to the *Theory of Functional Connections*

This chapter introduces the reader to the original work on TFC, at that time, published simply as the Theory of Connections [3]. Through this review, TFC is presented in the broader context of interpolation to show this method is a generalized

interpolation scheme enabling functional interpolation. This provides the mathematical framework to generate functionals (functions of functions) that analytically satisfy all imposed linear constraints and represent the real-valued set of functions satisfying the constraints. Additionally, to familiarize the reader with the specific vocabulary of TFC and how the method is used, specific examples are provided with increasing complexity. The scope of these examples are two-fold as they 1) provide the reader with concrete, step-by-step derivations and 2) develop an understanding of the theory such that the general formulation of the univariate framework, provided in Chapter 3, is easily understood. After these examples, an ad-hoc approach is developed to handle inequality type constraints. Then the chapter concludes with a section highlighting how the functionals derived through the TFC framework can be over-constrained.

Chapter 3: A General Formulation of the Univariate *Theory of Functional Connections*

Leveraging the intuition of the TFC method provided in Chapter 2, this chapter provides a rigorous definition of TFC, and the terminology used and is an expanded version of the general formulation first published by Leake, Johnston, and Mortari [4]. Whereas Chapter 2 highlights the consistent structure of the interpolating functionals, this chapter utilizes this discovery to define the terms, identify their associated mathematical properties, and ultimately provide straightforward proofs on the existence and uniqueness of these functionals. These proofs have further implications when the expressions are used to solve differential equations, which is covered in Chapter 4. Moreover, the development in this section facilitates the generalization of TFC to n-dimensions.

Part 2 — Application

Chapter 4: Application to the Solution of Ordinary Differential Equations

As mentioned earlier, the use of TFC expressions to solve ordinary differential equations is one of the three major pillars of the work presented in this dissertation. Consequently, careful attention is paid to developing the numerical framework and consistent notation throughout to allow ease of implementation. Similar to the examples provided in Chapter 2 to derive the interpolating functionals, this chapter provides example solutions of differential equations starting with linear ordinary differential equations and culminates in the solution of systems of coupled, nonlinear ordinary differential equations. The examples presented (i.e., the Lane-Emden equations, perturbed orbit propagation, perturbed Lambert's problem, etc.) are meant to guide the reader in implementing the method and provide solutions to some relevant equations in the field of science and engineering. Following these examples, two unpublished additions to the numerical application of TFC are introduced. First, the method is adapted for the solution of hybrid systems — where the dynamics exhibit discrete jumps over the solution domain. Following this, a numerical technique to handle unspecified time, i.e., unknown final time problems, is introduced and highlighted with examples. Lastly, the author provides some numerical applications to problems of over-constrained differential equations.

Part 3 — Optimal Control

Chapter 5: Use for Real-time Optimal Controllers in Aerospace Systems

This chapter contains an overview of the current techniques to solve optimal control problems, emphasizing real-time implementation. After distinguishing between the direct and indirect methods to solve optimal control problems, the first-order necessary conditions for optimality are derived from first principles using the indirect method. This derivation is used as a background for the reader. It precisely shows where the

TFC approach fits into the solution of the resultant system of equations by analytically satisfying a portion of these equations. Additionally, this chapter serves as a high-level literature review for the specific problems presented in Chapter 6 and Chapter 7, where the indirect method and TFC are ultimately used to solve these problems. Finally, further insight is provided with comparisons between TFC, spectral, and collocation methods already studied in the context of optimal control theory.

Chapter 6: Energy-Optimal Landing

In this chapter, the full three degree-of-freedom energy-optimal landing problem is formulated according to two different TFC based schemes, Outer-loop and Single-loop. The Outer-loop relies on an external optimizer to solve for the final time in the problem (i.e., MATLAB `fsolve()`), while the Single-loop incorporates all necessary conditions into a single TFC scheme. First, these schemes are compared to the feedback solution for the constant gravity case of this problem to ensure the method's accuracy. With this said, both TFC schemes are formulated "blind" to the feedback form to study the effects of the free-final time on the algorithm. Finally, the two developed approaches are studied through a Monte Carlo simulation for varying initial conditions and compared to a similar implementation using a spectral method.

Chapter 7: Fuel-Optimal Landing

With the increasing interest in human spaceflight operations to the Moon, Mars, and possibly beyond, maximizing the amount of payload that can be landed on these bodies is of utmost importance. Optimizing the landing trajectory and minimizing fuel consumption over this landing sequence is one of the many avenues to achieve this. In all, this fuel-optimal landing problem is still an active area of research in the field of aerospace engineering. Therefore, this chapter is the culmination of the prior chapters, and herein, the three degree-of-freedom fuel-optimal landing problem is formulated and solved using the TFC framework. Similar to Chapter 6, all neces-

sary conditions are derived using the indirect method, which poses the problem as a nonlinear system of equations that is solved using the TFC framework. Ultimately, the resulting algorithm is used to solved for these trajectories and compared to current state-of-the-art, commercially available methods.

Chapter 8: Summary and Conclusions

While this dissertation stretches from the basis of the analytical method to embed constraints (the Theory of Functions Connections) to the numerical solution of optimal control problems, it was infeasible to include everything that has been done with regards to this new theory. Therefore, along with drawing the major takeaways of the work presented in this dissertation, I have also devoted the final section of this work (Section 8.2) to comment on the state of TFC at the date of publication. This includes a comprehensive list of all available publications with a summary of the major contributions and results. Additionally, I have also noted the current work in progress and the key problems moving forward. Therefore, after reading, this section can be leveraged by new researchers as a path to interesting and fruitful topics in the greater field of TFC.

Part 1
Theory

How beautiful this was, when it was new.

And how beautiful it still is, even

though time has made it different.

— Unravel, *ColdWood Interactive*

2. AN INTRODUCTION TO THE *THEORY OF FUNCTIONAL CONNECTIONS*

Interpolation is the mathematical process of estimating an unknown function's values within the range of k given data points, called constraints, provided by some unknown continuous process. Occasionally, in engineering and science, a function is expressed as data points, whether through sampling or experimentation. These data points represent a finite series (or reconstruction) of the governing process (function) at specific independent variable values. Given this data, it is often desired to estimate the function value at some point in between the given data. In another case, the function might be known but is defined by a complicated set of equations that are computationally inefficient to evaluate. In this context, it may be more desirable to approximate the function using a simpler function (with some associated interpolation error) that is easier to evaluate.

Our first mathematical understanding of interpolation can be traced back to elementary algebra. We were first introduced to interpolation when we looked for the numerical description of the line passing through two points (x_1, y_1) and (x_2, y_2). Recall the equation takes the form,

$$y(x) = y_1 + (y_2 - y_1)\frac{x - x_1}{x_2 - x_1}, \tag{2.1}$$

where x is a point along the domain. However, as we look to include more data points, we must substitute this linear interpolation method with other techniques such as polynomial interpolation, where the entire function is described by a polynomial, or spline interpolation, where the function is described by piecewise polynomials between data points. Regardless, these techniques provide us with general interpolation schemes to include a given number of points. As one specific example, a popular technique for polynomial interpolation is Lagrange

polynomials[1]

$$L_k(x) = \sum_{i=0}^{k} y_j \, \phi_j(x)$$

where the polynomial $L_k(x)$ passes through each set of $k+1$ data points (x_j, y_j), and $\phi_j(x)$ are polynomials based on the equation,

$$\phi_j(x) = \prod_{\substack{0 \le i \le k \\ i \ne j}} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)},$$

where $0 \le j \le k$. For example, if two data points are selected ($k = 1$), then the formula reduces to our simple description of a line,

$$L_2(x) = y_1\Big(\frac{x - x_2}{x_1 - x_2}\Big) + y_2\Big(\frac{x - x_1}{x_2 - x_1}\Big).$$

Creating the interpolating polynomial in this way makes it easy to see how the constraints of y_1 and y_2 are satisfied. The $\phi_j(x)$ terms multiplying the constraint terms act as continuous switches that evaluate to 1 at the constraint they are associated with and 0 when evaluated at all other constraints. In the case of the polynomial $L_2(x)$, we can see that the term multiplying y_1 is 1 when $x = x_1$ and is 0 when $x = x_2$. Furthermore, by simple algebraic manipulation, we can see that this equation is identical to Equation (2.1).

At this point, some questions may arise:

- What if we have data associated with derivatives as well?
- What if we are interested in all possible functions that interpolate these point and derivative values?
- What if the function of interest is based on a combination of data measurements?

[1]The author notes that the name "Lagrange polynomials" is an academic misnomer since the formula was actually first discovered by Edward Waring [5] in 1779, then by Leonhard Euler in 1783, and eventually Joseph-Louis Lagrange in 1795.

In general, a method that provides answers to these questions is interested in the interpolation of functions rather than just points: in other words, a method for "functional interpolation." Whereas Lagrange polynomials provide the polynomial expression that passes through all given points, the method of interest here is a functional[2] that represents all possible functions satisfying some given data set conditions, where these "conditions" are not limited to points. The questions mentioned and the search for a functional interpolation framework led to the development of the Theory of Functional Connections (TFC) [3] in the seminal paper by Mortari [3].

The foundation of this work is built on a straightforward method to derive analytical expressions (or functionals), which represent the set of all functions satisfying a specified combination of constraints. In his original paper, Mortari identified three unique ways to build these functionals, including linear, additive, and rational forms.

$$y(x, g(x)) = g(x)(x - x_0) + y_0 \qquad \text{(linear)}$$

$$y(x, g(x)) = g(x) + [y_0 - g(x_0)] \qquad \text{(additive)}$$

$$y(x, g(x)) = \frac{g(x)}{g(x_0)}\, y_0 \qquad \text{(rational)}$$

However, the additive form proved to be the most fruitful and therefore, the name the "Theory of Functional Connections" refers to functional interpolation using the additive form.[4] In this approach, the resulting functional was coined as a "constrained expression" since they constrain the functional to analytically satisfy the imposed constraints. Mortari's original

[2]Also known as a higher-order function or a function of functions.

[3]This theory was originally published under the name "Theory of Connections." However, this name conflicted with a specific theory in differential geometry and was not the most accurate description of the functional interpolation method. Therefore, in 2019, this name was changed to the "Theory of Functional Connections," to highlight the tie to functional interpolation and the fact that it provides *all* functions satisfying a set of linear constraints in rectangular domains of n-dimensional space.

[4]Note that linear, additive, and rational forms are equivalent through functional transformations. For example, by performing the logarithm of the rational formulation, an additive formulation is obtained. The additive formulation can also be recovered from the linear formulation by simply setting the function $g(x)$ in the additive formulation as $x\, g(x)$. Therefore, the additive form was adopted as the main formalism because of its simplicity.

work [3] provided examples of constraints in k points, constraints in k points and derivatives, and relative constraints. It hinted at the idea of linear constraints, something that this dissertation introduces along with a unified notation and associated claims. In all, the original work produced a generalized interpolation technique, as will soon be demonstrated. In fact, in the cases where only function values are considered, i.e., point constraints, it is easy to see that Lagrange polynomials are a specific case of the more general TFC.

While the idea of functional interpolation is not new, prior methods only existed for a class (or sub-class) of functions and not all of function space [6, 7, 8, 9]. More current techniques also include distributed approximating functions (DAFs) [10, 11], which use Hermite DAFs and Sinc DAFs. However, the theory discovered by Mortari [3] is the first interpolation technique not restricted to a specific class of functions. In the following section, a summary of the major points in this discovery is provided, along with a step-by-step development of the functional interpolation method called TFC. In all, what was discovered in this seminal paper is leveraged to develop a general technique to handle general linear constraints.

2.1 An introduction to constrained expressions

The idea for TFC started with an attempt to derive an expression for all functions passing through the specific point (x_0, y_0). Using algebra, one can easily define all *straight lines* with the equation, $y(x, m) = m(x - x_0) + y_0$, where $y(x_0) = y_0$ and m represents the constant value of the slope. Yet, the slope could be defined by a function, $m(x) : \mathbb{R} \to \mathbb{R}$, where the only restriction on $m(x)$ is it must be defined at x_0. By making this modification, the expression now becomes a functional, $y(x, m(x)) : \mathbb{R} \to \mathbb{R}$ that represents all functions that evaluate to y_0 at $x = x_0$. Although this functional always satisfies the constraints, and is thereby a valid constrained expression,[5] the derivation process did not provide a clear path to add multiple constraints. Therefore, a different approach is desired.

Said approach came from the realization that the *additive* form of the constrained expression describes all functions passing through the point defined earlier. Let $g(x) : \mathbb{R} \to \mathbb{R}$,

[5] A rigorous definition of a constrained expression is provided in Chapter 3.

be a user defined function that is defined at x_0, then the expression,

$$y(x, g(x)) = g(x) + (y_0 - g(x_0)), \tag{2.2}$$

produces a similar result to the constrained expression $y(x, m(x)) = m(x)(x - x_0) + y_0$, however, the function $g(x)$ appears linearly, which we will soon find to be invaluable. The next step was to determine the general methodology to derive Equation (2.2). Without changing the constrained expression, the latter term could be multiplied with the value 1, or in fact, any function $s(x)$ such that $s(x_0) = 1$. Let us define this function as simply $s(x) = 1$. Adding this to Equation (2.2) leads to,

$$y(x, g(x)) = g(x) + s(x)(y_0 - g(x_0))$$

Analyzing this equation, the term $y_0 - g(x_0)$ is constant for a a given $g(x)$ and is the only term containing information of the constraint point, let us denote this constant by η, and insert it into the equation and rearrange,

$$y(x, g(x)) = g(x) + s(x)\eta. \tag{2.3}$$

It becomes clear that in order to determine the coefficient η this equation must be evaluated at the constraint point (x_0, y_0). This realization was a pivotal moment in the discovery of the constrained expression, and it quickly followed that a general expression to Equation (2.3) could be written as,

$$y(x, g(x)) = g(x) + \sum_{j=1}^{k} s_j(x)\eta_j \tag{2.4}$$

where again $g(x) : \mathbb{R} \to \mathbb{R}$ is the free function. Additionally, the summation term is a linear combination of the functions, $s_j(x) : \mathbb{R} \to \mathbb{R}$, which we will call support functions, and the η_j coefficients, which we have already seen capture the constraint information. In fact, from this general expression we can quickly return to Equation (2.2). For this problem, the

number of constraints k is one, so the expression becomes

$$y(x, g(x)) = g(x) + s(x)\eta.$$

Evaluating the expression at the point (x_0, y_0), solving for η, and inserting it back into the expression above yields,

$$y(x, g(x)) = g(x) + \frac{s(x)}{s(x_0)}(y_0 - g(x_0)).$$

Defining $s(x) = 1$ this equation reduces to Equation (2.2). Finally, Equation (2.3) facilitates the derivation of constrained expressions for even more complicated sets of constraints.

2.2 Adding a second constraint

The next logical step is to find the constrained expression passing through two points. While in the previous derivation $s(x)$ was set loosely and without explanation, this example provides insight into how the support function, $s(x)$, must be chosen. Using Equation (2.4) as a template, let us derive an expression such that $y(x_1) = y_1$ and $y(x_2) = y_2$.

Example 2.1: Constraints at two points

Since there are two constraints, Equation (2.4) takes the form,

$$y(x, g(x)) = g(x) + s_1(x)\eta_1 + s_2(x)\eta_2. \tag{2.5}$$

Evaluating this expression at the two constraint points (e.g., for the first constraint, this means evaluating the right hand side of the equation at x_1 and setting it equal to y_1), leads to a system of equations,

$$y_1 = g(x_1) + s_1(x_1)\eta_1 + s_2(x_1)\eta_2$$
$$y_2 = g(x_2) + s_1(x_2)\eta_1 + s_2(x_2)\eta_2$$

where the only unknowns are the η_k coefficients. Writing these in vector-matrix form leads to a system of equations for these coefficients,

$$\begin{Bmatrix} y_1 - g(x_1) \\ y_2 - g(x_2) \end{Bmatrix} = \begin{bmatrix} s_1(x_1) & s_2(x_1) \\ s_1(x_2) & s_2(x_2) \end{bmatrix} \begin{Bmatrix} \eta_1 \\ \eta_2 \end{Bmatrix}.$$

By inverting the matrix composed of the support functions evaluated at the constraints, we can solve for the unknown coefficients η_1 and η_2. This highlights the major restriction on our definition of the support functions since to solve for η coefficients, the matrix *must* be invertible. In other words, the columns, and therefore the support functions, *must* be linearly independent.

Continuing with this example, by selecting $s_1(x) = 1$ and $s_2(x) = x$, which are linearly independent, the system of equations becomes,

$$\begin{Bmatrix} y_1 - g(x_1) \\ y_2 - g(x_2) \end{Bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{Bmatrix} \eta_1 \\ \eta_2 \end{Bmatrix}.$$

Solving this system yields the η_1 and η_2 values,

$$\begin{aligned} \eta_1 &= \frac{1}{x_2 - x_1}\Big(x_2[y_1 - g(x_1)] - x_1[y_2 - g(x_2)]\Big) \\ \eta_2 &= \frac{1}{x_2 - x_1}\Big([y_2 - g(x_2)] - [y_1 - g(x_1)]\Big) \end{aligned}$$

which can then be substituted into Equation (2.5) to produce the constrained expression,

$$\begin{aligned} y(x, g(x)) = g(x) &+ \frac{1}{x_2 - x_1}\Big(x_2[y_1 - g(x_1)] - x_1[y_2 - g(x_2)]\Big) \\ &+ \frac{x}{x_2 - x_1}\Big([y_2 - g(x_2)] - [y_1 - g(x_1)]\Big). \end{aligned}$$

While it may seem there is an excessive use of parenthesis, these are used to highlight that the terms $y_1 - g(x_1)$ and $y_2 - g(x_2)$ show up in the latter two terms, and thus, the equation can be rearranged by collecting on these two terms. Doing this leads to the familiar result obtained in the original derivation in Reference [3],

$$y(x, g(x)) = g(x) + \frac{x_2 - x}{x_2 - x_1}\Big(y_1 - g(x_1)\Big) + \frac{x - x_1}{x_2 - x_1}\Big(y_2 - g(x_2)\Big). \tag{2.6}$$

Using the constrained expression from Equation (2.6), it is easy to see that if this functional is evaluated at either x_1 or x_2, the corresponding constraint value of y_1 or y_2 is obtained regardless of the function $g(x)$. Further analyzing this equation, we might ask, what happens if we select the simplest expression for the free function such that $g(x) = 0$? If $g(x) = 0$, then Equation (2.6) reduces to

$$y = \frac{x_2 - x}{x_2 - x_1}\, y_1 + \frac{x - x_1}{x_2 - x_1}\, y_2,$$

which the reader may recognize as the Lagrange polynomial for two points discussed earlier. This result should come as no surprise since the original goal was to derive a functional that represents all possible functions passing through the given set of constraints, or in this simple case, points. In the context of our constrained expression, the Lagrange polynomial is the simplest interpolating function of the functional $y(x, g(x))$, when 1 and x are chosen as support functions. While this generalization is insightful, TFC should not be taken as a simple generalization of Lagrange polynomials. The following examples highlight that point constraints are merely the beginning of the theory.

2.3 The structure of the constrained expression

The prior example hinted at an interesting form of the constrained expression but did not give a mechanized method to arrive at the end result. This section explores Equation (2.6), specifically, and the terms dictating the constraints, to bring to light a structure within

the constrained expression that can be utilized to create the aforementioned mechanized method. Moreover, the said method will ultimately reveal itself to be a unified, consistent way to develop constrained expressions for many different types of constraints.

First, notice that the latter two terms in the constrained expression consist of two unique parts, 1) a term composed of only the support functions and their values at the constraint locations and 2) a term composed of the constraint condition and the function $g(x)$ evaluated at this constraint condition. As an example, consider the first of these terms from Equation (2.6),

$$\underbrace{\frac{x_2 - x}{x_2 - x_1}}_{\phi_1(x)} \underbrace{(y_1 - g(x_1))}_{\rho_1(x, g(x))}.$$

The first part of this structure we will call the switching function, $\phi_j(x)$. This function is defined such that it is equal to 1 when evaluated at the constraint it is referencing, and equal to 0 when evaluated at all other constraints. In our example, when evaluating the switching function, $\phi_1(x)$, at the constraint it is referencing it is equal to 1 (i.e., $\phi_1(x_1) = 1$), and when it is evaluated at the other constraints it is equal to 0 (i.e., $\phi_2(x_1) = \dfrac{x_1 - x_1}{x_2 - x_1} = 0$).

The second part of the structure, $\rho_1(x, g(x))$, is called the projection functional. In this case, the projection functional is simply the difference between the constraint value and the free function evaluated at that constraint; however, for more complex constraints this is not always the case. We choose the name projection functional because it "projects" the free function onto the set of functions that vanish at the constraint. Continuing with our example, the projection functional, $\rho_1(x, g(x))$, is simply the difference between the constraint $y(x_1) = y_1$ and the free function evaluated at the constraint point, $g(x_1)$. This structure is important, as it shows up in all other constraint types we consider. Additionally, notice what happens to the projection functional if $g(x)$ satisfies the constraint,

Property 1

The projection functionals for constraints at a point are always equal to zero if the free function, $g(x)$, is selected such that it satisfies the associated constraint.

This simply means that if $g(x)$ were defined such that $g(x) := y_1$, the entire term would reduce to 0. This property will be utilized in mathematical claims later in the dissertation.

Based on this structure, consider an alternative structure to Equation (2.4), which leverages the fact that the constrained expression can be built as a sum of switching functions and projection functionals expressed as,

$$y(x, g(x)) = g(x) + \sum_{j=1}^{k} \phi_j(x)\rho_j(x, g(x)). \tag{2.7}$$

First, based on their composition, the projection functionals, $\rho_j(x, g(x))$, are trivial to derive, but the switching functions, ϕ_j, require some attention. From the definition of the switching functions, these functions must go to 1 at their associated constraint and 0 at all other constraints. As a result, the following algorithm can be used to derive the switching functions for a set of k constraints:

Algorithm to derive the terms of Equation (2.7)

1. Choose the k linearly independent support functions, s_k.
2. Write each switching function as a linear combination of the support functions with k unknown coefficients.
3. Based on the switching function definition, write a system of equations to solve for the unknown coefficients.

To validate this approach, let us rederive the constrained expression from Example 2.1.

Example 2.2: Constraints at two points (Alternative derivation)

Given two constraints, Equation (2.7) takes the form,

$$y(x, g(x)) = g(x) + \phi_1(x)\rho_1(x, g(x)) + \phi_2(x)\rho_2(x, g(x)) \tag{2.8}$$

where the switching functions are of the form,

$$\phi_1(x) = s_i(x)\alpha_{i1} \quad \text{and} \quad \phi_2(x) = s_i(x)\alpha_{i2}$$

for some as yet unknown coefficients α_{ij}; note that in the previous expression, and throughout this book, the Einstein summation convention[a] is used to improve readability. Additionally, the projection functionals are,

$$\rho_1(x, g(x)) = y_1 - g(x_1) \quad \text{and} \quad \rho_2(x, g(x)) = y_2 - g(x_2).$$

Now, the definition of the switching function is used to come up with a set of equations. For example, the first switching function has the two equations,

$$\phi_1(x_1) = 1, \quad \phi_1(x_2) = 0.$$

The equations for all the switching functions can be combined into the compact form,

$$\begin{bmatrix} s_1(x_1) & s_2(x_1) \\ s_1(x_2) & s_2(x_2) \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) \\ \phi_1(x_2) & \phi_2(x_2) \end{bmatrix}.$$

This equation offers us our first visible connection to the original technique to derive constrained expressions. Notice that the support function matrix, i.e., the matrix composed of the support functions, is identical to the matrix multiplying the η coefficients in our prior example. Therefore, it still holds that the support functions must

be linearly independent. Therefore, and in order to mirror Example 2.1, let us define the support functions as, $s_1(x) = 1$ and $s_2(x) = x$, and the matrix of ϕ_j is identity by definition. Solving the system provides the values of the coefficients α_{ij}

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}^{-1} = \frac{1}{x_2 - x_1} \begin{bmatrix} x_2 & -x_1 \\ -1 & 1 \end{bmatrix}.$$

Substituting the constants back into the switching functions and simplifying yields,

$$\phi_1 = \frac{x_2 s_1(x) - s_2(x)}{x_2 - x_1} = \frac{x_2 - x}{x_2 - x_1} \quad \text{and} \quad \phi_2 = \frac{s_2(x) - x_1 s_1(x)}{x_2 - x_1} = \frac{x - x_1}{x_2 - x_1}.$$

Lastly, by substituting the switching functions along with the associated projection functionals back into Equation (2.8), the constrained expression becomes,

$$y(x, g(x)) = g(x) + \underbrace{\frac{x_2 - x}{x_2 - x_1}}_{\phi_1(x)} \underbrace{(y_1 - g(x_1))}_{\rho_1(x, g(x))} + \underbrace{\frac{x - x_1}{x_2 - x_1}}_{\phi_2(x)} \underbrace{(y_2 - g(x_2))}_{\rho_2(x, g(x))}.$$

[a] For example, $a_i b_i = \boldsymbol{a}^{\mathrm{T}} \boldsymbol{b}$ for the inner product.

The result is identical to Equation (2.6) and should come as no surprise as it is simply an exploitation of the structure of the constrained expression. At this point, it may be unclear the benefit of using Equation (2.7) to construct constrained expressions; however, the following section provides in-depth examples building up to general, linear-type constraints where the true power of the switching-projection notation will become obvious.

2.4 Examples using the switching-projection form of the constrained expression

While our motivating example in the prior section was vital to our understanding of the constrained expression and its underlying structure, it is limited to the application of constraints at a point. However, the insight and methodology built up in this example can be applied to various linear constraints. The following sections provide specific examples of the application of Equation (2.7). Admittedly, one could derive all of the following examples using the original form of the constrained expression, Equation (2.7), albeit with more difficulty.

2.4.1 Point and derivative constraints

In our first example, we take a small step by including derivative constraints into the constrained expression. The reader will see that this does not add any complexity when using the TFC approach.

Example 2.3: Point and derivative constraints

Consider the following set of point and derivative constraints defined by,

$$y(0) = 1, \quad y_x(1) = 2, \quad y(2) = 3,$$

where the notation $y_x := \frac{\mathrm{d}y}{\mathrm{d}x}$ is used for the derivative of the function $y(x)$ with respect to x. The projection functionals are immediate and can be written as,

$$\rho_1(x, g(x)) = 1 - g(0), \quad \rho_2(x, g(x)) = 2 - g_x(1), \quad \rho_3(x, g(x)) = 3 - g(2).$$

Now, the only terms that remain are the switching functions. Recall that our definition of the switching functions in terms of the support functions, $s_i(x)$, and the unknown coefficients, α_{ij}, is $\phi_j(x) = s_i(x)\alpha_{ij}$, and the expressions for the three switching functions are, $\phi_1 = s_i(x)\alpha_{i1}$, $\phi_2(x) = s_i(x)\alpha_{i2}$, and $\phi_3(x) = s_i\alpha_{i3}$. Now, this definition of

the switching function is used to come up with a set of equations. For example, the first switching function has the three equations,

$$\begin{aligned}
\phi_1(0) &= s_1(0)\alpha_{11} + s_2(0)\alpha_{21} + s_3(0)\alpha_{31} = 1 \\
\frac{\partial \phi_1}{\partial x}(1) &= s_{1_x}(1)\alpha_{11} + s_{2_x}(1)\alpha_{21} + s_{3_x}(1)\alpha_{31} = 0 \\
\phi_1(2) &= s_1(2)\alpha_{11} + s_2(2)\alpha_{21} + s_3(2)\alpha_{31} = 0.
\end{aligned}$$

where the reader should notice that the second equation involves the derivative of the switching function and is associated with the derivative constraint $y_x(1) = 2$. It is convenient to represent these equations in matrix form,

$$\begin{bmatrix} s_1(0) & s_2(0) & s_3(0) \\ s_{1_x}(1) & s_{2_x}(1) & s_{3_x}(1) \\ s_1(2) & s_2(2) & s_3(2) \end{bmatrix} \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \end{bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}.$$

Adding the expressions of the other two switching functions, the set of equations becomes,

$$\begin{bmatrix} s_1(0) & s_2(0) & s_3(0) \\ s_{1_x}(1) & s_{2_x}(1) & s_{3_x}(1) \\ s_1(2) & s_2(2) & s_3(2) \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Now, we can determine a valid expression of support functions that ensures the support matrix is non-singular. For example, if the support functions were chosen as $s_i(x) = (1, x, x^2)$, the 2^{nd} and 3^{rd} columns of the support matrix would be linearly dependent; hence, this is an invalid set. The simplest set of monomials that satisfies the requirement is $s_i(x) = (1, x^2, x^3)$. Using the defined support functions, the α_{ij}

coefficients can be derived as follows,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 3 \\ 1 & 4 & 8 \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 3 \\ 1 & 4 & 8 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{3}{4} & 2 & -\frac{3}{4} \\ -\frac{1}{2} & -1 & \frac{1}{2} \end{bmatrix}.$$

Substituting the constants back into the switching functions and simplifying yields,

$$\phi_1(x) = \frac{-2x^3 + 3x^2 + 4}{4}, \quad \phi_2(x) = -x^3 + 2x^2, \quad \phi_3(x) = \frac{2x^3 - 3x^2}{4}.$$

Finally, substituting the switching functions and projection functionals back into the constrained expression yields,

$$\begin{aligned} y(x, g(x)) = g(x) + \frac{-2x^3 + 3x^2 + 4}{4}\Big(1 - g(0)\Big) \\ + \Big(- x^3 + 2x^2\Big)\Big(2 - g_x(1)\Big) + \frac{2x^3 - 3x^2}{4}\Big(3 - g(2)\Big), \end{aligned} \tag{2.9}$$

It is simple to verify that regardless of how $g(x)$ is chosen, provided $g(x)$ is defined at the constraint points, Equation (2.9) always satisfies the given constraints.

2.4.2 Integral constraints

Moving forward, another constraint type of interest and one that can be easily incorporated using the TFC approach are integral constraints that include an integral over all or part of the domain. While the idea was first presented in Johnston and Mortari [12], this work relied on the original formulation. With the discovery of the switching-projection form, integral constraints become easier to embed.

Example 2.4: Integral constraints

Consider the function $y(x)$ subject to,

$$\int_0^3 y(x)\,\mathrm{d}x = 0 \quad \text{and} \quad \int_1^2 y(x)\,\mathrm{d}x = 2.$$

Following the same process as the prior example, first the projection functionals are determined. For this problem, the projection functions are merely the difference between the constraint value and free function evaluated over the integral. For this example,

$$\rho_1(x, g(x)) = -\int_0^3 g(\zeta)\,\mathrm{d}\zeta \quad \text{and} \quad \rho_2(x, g(x)) = 2 - \int_1^2 g(\zeta)\,\mathrm{d}\zeta.$$

where ζ is a "dummy" variable for the integration of the function, $g(x)$. As before, the switching functions are defined such that they are equal to 1 when evaluated at their associated integral constraint, and equal to 0 when evaluated at all other constraints. For this example,

$$\int_0^3 \phi_1(x)\,\mathrm{d}x = 1, \quad \int_1^2 \phi_1(x)\,\mathrm{d}x = 0,$$

for the first switching function, and

$$\int_0^3 \phi_2(x)\,\mathrm{d}x = 0, \quad \int_1^2 \phi_2(x)\,\mathrm{d}x = 1,$$

for the second switching function. Similar to the previous examples, the switching functions are chosen to be a linear combination of support functions. For the first switching function, this form yields,

$$\begin{aligned}\int_0^3 \phi_1(x)\,\mathrm{d}x &= \int_0^3 \Big(s_1(x)\alpha_{11} + s_2(x)\alpha_{21}\Big)\,\mathrm{d}x \\ &= \alpha_{11}\int_0^3 s_1(x)\,\mathrm{d}x + \alpha_{21}\int_0^3 s_2(x)\,\mathrm{d}x = 1\end{aligned}$$

where we can see that the unknown α_{ij} terms still appear linearly. The final step is to define the specific support functions, and evaluate them at the constraint conditions to populate the support matrix. For this example, let's choose the support functions $s_1(x) = 1$ and $s_2(x) = x^2$. Expressing the support functions in this way yields,

$$\begin{bmatrix} 3 & 9 \\ 1 & \frac{7}{3} \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 3 & \frac{9}{2} \\ 1 & \frac{3}{2} \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{7}{6} & \frac{9}{2} \\ \frac{1}{2} & -\frac{3}{2} \end{bmatrix},$$

The solution of this system yields the following switching functions,

$$\phi_1(x) = \frac{3x^2 - 7}{6} \quad \text{and} \quad \phi_2(x) = \frac{-3x^2 + 9}{2}.$$

Finally, substituting the switching functions and projection functionals back into the constrained expression given in Equation (2.7) produces,

$$y(x, g(x)) = g(x) - \frac{3x^2 - 7}{6} \int_0^3 g(\zeta)\ \mathrm{d}\zeta + \frac{-3x^2 + 9}{2} \Big(2 - \int_1^2 g(\zeta)\ \mathrm{d}\zeta \Big).$$

Again, it is easy to check this constrained expression to ensure that the constraints are met regardless of the value of $g(x)$. The inclusion of integral constraints leads to another property of projection functionals.

Property 2

The projection functions for integral constraints are always equal to zero if the free function is selected such that it satisfies the integral constraint.

For example, if $g(x)$ is selected such that $\int_1^2 g(\zeta)\ \mathrm{d}\zeta = 2$, then the second projection

function in this example becomes $\rho_2(x, g(x)) = 2 - \int_1^2 g(\zeta)\, \mathrm{d}\zeta = 0$.

2.4.3 Linear constraints

Taking our discussion on the derivation of constrained expressions a step further, the culmination of all prior examples is the linear constraint case. It is noted that by this definition, relative constraints such as $y(0) = y(1)$ are just a specific case of linear constraints. As mentioned earlier, the idea of embedding a general set of linear constraints is not new and was first teased in the seminal TFC paper [3]; however, the original form proved cumbersome when deriving constrained expressions of this type. In the following example, we highlight that these linear constraints can be embedded in the same way as the prior examples in the new generalized formulation.

Example 2.5: Linear constraints

For this example, let us consider the linear constraints,

$$y(0) = y(1) \quad \text{and} \quad 3 = \int_0^1 y(x)\, \mathrm{d}x + \pi y_x(0).$$

To generate a constrained expression, first the constraints are arranged such that the constants are collection on one side; for example,

$$0 = y(1) - y(0) \quad \text{and} \quad 3 = \int_0^1 y(x)\, \mathrm{d}x + \pi y_x(0).$$

By organizing the constraints in this manner, the projection functionals, again, are immediate. However, the author notes one extra step must be taken for the general linear constraints. The projection functionals take the form,

$$\rho_1(x, g(x)) = g(0) - g(1) \quad \text{and} \quad \rho_2(x, g(x)) = 3 - \int_0^1 g(\zeta)\, \mathrm{d}\zeta - \pi g_x(0),$$

where again ζ is the "dummy" variable for the integration of the free function.

The switching functions are again such that they are equal to 1 when evaluated with their associated constraint and equal to 0 when evaluated at all other constraints. However, the word "evaluation" in the previous sentence requires clarification. Here, evaluation means to replace the function, $y(x)$ in this case, with the switching function and remove any terms not multiplied by the switching function. For this example, this leads to

$$\phi_1(1) - \phi_1(0) = 1, \quad \int_0^1 \phi_1(x)\,\mathrm{d}x + \pi\frac{\partial\phi_1}{\partial x}(0) = 0,$$

for the first switching function, and

$$\phi_2(1) - \phi_2(0) = 0, \quad \int_0^1 \phi_2(x)\,\mathrm{d}x + \pi\frac{\partial\phi_2}{\partial x}(0) = 1,$$

for the second switching function. As in all prior examples, the switching functions are defined as a linear combination of support functions with unknown coefficients. Again, this can be written compactly in matrix form. For this example, let's choose the support functions $s_1(x) = 1$ and $s_2(x) = x$. Then the set of equations becomes,

$$\begin{bmatrix} 0 & 1 \\ 1 & \frac{1}{2}+\pi \end{bmatrix}\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & \frac{1}{2}+\pi \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{1}{2}-\pi & 1 \\ 1 & 0 \end{bmatrix}.$$

These coefficients are, as always, used to define the switching functions,

$$\phi_1(x) = -\frac{1}{2} - \pi + x, \quad \phi_2(x) = 1.$$

Lastly, substituting the switching functions and projection functionals back into the constrained expression form given in Equation (2.7) yields,

$$y(x, g(x)) = g(x) + \Big(-\frac{1}{2} - \pi + x \Big)\Big(g(0) - g(1)\Big) + \Big(3 - \int_0^1 g(\zeta)\, \mathrm{d}\zeta - \pi g_x(0)\Big).$$

By substituting this expression for $y(x)$ back into the constraints, one can verify that this constraint expression satisfies the constraints regardless of the choice of the free function $g(x)$. Therefore, we are lead to a similar property as those observed before.

Property 3

The projection functionals for linear constraints are always equal to zero if the free function is selected such that it satisfies the associated constraint.

It should be clear that Property 3 extends Property 1 and Property 2 to any linear constraints. For example, if $g(x)$ is selected such that $g(1) = g(0)$, then the first projection functional in this example becomes $\rho_1(x, g(x)) = g(1) - g(0) = 0$. Thus far, all examples have been for *scalar* univariate equations. In the following examples we will look into vector univariate equations where another interesting constraint case arises: component constraints.

2.4.4 Component constraints

Component constraints involve constraints across dependent variables. Mortari and Furfaro [13] first looked at these constraints and their application to solving systems of ordinary differential equations. The following example is used to highlight that the new, generalized, constrained expression with the switching-projection form can easily embed any set of linear component constraints.

Example 2.6: Component linear constraints

As with the prior constraint types, it is easiest to explore this constraint type through an example. Therefore, consider the vector function where the dependent variables x, y, and z are all functions of the independent variable t and are constrained by the following,

$$x(0) = 2y(0) + \int_{-1}^{+1} z(t)\,\mathrm{d}t \quad \text{and} \quad \dot{y}(0) = 2x(1) - z(1).$$

When handling component constraints, one must decide which dependent variables constrained expression the component constraint will be embedded. Regardless of which dependent variable is chosen, a valid constrained expression will be produced. For this example, let us choose to embed all constraints into the x-component (note: this could have also been done for the y-component or z-component). Doing this leads to the following constrained expressions,

$$\begin{aligned} x(t, g^x(t), g^y(t), g^z(t)) = g^x(t) &+\phi_1(t)\rho_1(t, g^x(t), g^y(t), g^z(t)) \\ &+\phi_2(t)\rho_2(t, g^x(t), g^y(t), g^z(t)) \\ y(t, g^y(t)) = g^y(t) & \\ z(t, g^z(t)) = g^z(t), & \end{aligned} \tag{2.10}$$

Now, the definition of the projection functionals become,

$$\begin{aligned} \rho_1(t, g^x(t), g^y(t), g^z(t)) &= g^x(0) - 2y(0, g^y(t)) - \int_{-1}^{+1} z(\zeta, g^z(\zeta))\,\mathrm{d}\zeta \\ \rho_2(t, g^x(t), g^y(t), g^z(t)) &= \dot{y}(0, g^y(t)) - 2g^x(1) + z(1, g^z(t)), \end{aligned}$$

where we can see that $g^x(t)$, which represents the free function used for the $x(t)$ constrained expression, is the only free function that shows up in the expressions.

Additionally, since the vector equation is a function of the independent variable t the dot operator is used to signify the derivative such that $\dot{y} := \frac{\mathrm{d}y}{\mathrm{d}t}$.

Similar to previous examples, the number of switching functions is equal to the number of constraints. The switching functions are derived by evaluating the conditions based on the applied constraints,

$$-\phi_1(0) = 1, \quad 2\phi_1(1) = 0$$
$$-\phi_2(0) = 0, \quad 2\phi_2(1) = 1.$$

The negative sign will be explained in greater detail in Chapter 3 and is based on the structure of constraints and projection functionals.

As in previous examples, the switching functions are chosen to be a linear combination of support functions. Let the support functions for this example be $s_1(t) = 1$ and $s_2(t) = t$. Then,

$$\begin{bmatrix} -1 & 0 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 2 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} -1 & 0 \\ 1 & \frac{1}{2} \end{bmatrix}.$$

where $\phi_1 = \alpha_1 1 + \alpha_2 1t$ and $\phi_2 = \alpha_1 2 + \alpha_2 2t$. Substituting these values into the constrained expressions shown in Equation (2.10) yields,

$$x(t, g^x(t), g^y(t), g^z(t)) = g^x(t) + (t-1)\Big(g^x(0) - 2y(0, g^y(t)) - \int_{-1}^{+1} z(\zeta, g^z(\zeta))\, \mathrm{d}\zeta\Big) + \frac{t}{2}\Big(\dot{y}(0, g^y(t)) - 2g^x(1) + z(1, g^z(t))\Big)$$
$$y(t, g^y(t)) = g^y(t)$$

$$z(t, g^z(t)) = g^z(t).$$

As with all prior examples, notice that regardless of how the free functions are chosen, these constrained expressions will always satisfy the constraints. In fact, Property 3 can be extended to component constraints.

Property 4

The projection functions for component constraints are always equal to zero if the free functions are selected such that they satisfy the component constraints.

For example, if $g^x(t)$, $g^y(t)$, and $g^z(t)$ are selected such that $\dot{g}^y(0) = 2g^x(1) - g^z(1)$, then the second projection function in this example becomes $\rho_2(t, g(t)) = \dot{g}^y(0) - 2g^x(1) + g^z(1) = 0$.

2.4.5 Mixed constraints

The methods for building constrained expressions shown in the previous four examples can be combined. However, special care must be taken when combining component constraints with the other types of constraints discussed earlier. The nuances of doing so are highlighted in this example.

Example 2.7: Mixed constraints

Consider the vector function where the dependent variables x and y are both functions of the independent variable t and are constrained by the following equations,

$$x(0) = 0, \quad y(0) = 0, \quad y(1) = y(2), \quad \text{and} \quad 4 = 2y(1) - \int_0^3 x(t)\, \mathrm{d}t.$$

Based on the previous examples, the four projection functions are defined,

$$\rho_1(t, g^x(t)) = -g^x(0), \quad \rho_3(t, g^y(t)) = g^y(1) - g^y(2),$$
$$\rho_2(t, g^y(t)) = -g^y(0), \quad \rho_4(t, g^x(t), g^y(t)) = 4 - 2y(1, g^y(t)) + \int_0^3 g^x(\zeta)\ \mathrm{d}\zeta.$$

As there are four constraints, there must also be four switching functions. Based on the constraints, the first must be associated with the x independent variable, and the second and third must be associated with the y independent variable. However, just as in the previous example, with the component constraint, there is freedom to choose where the constraint goes. How we have written $\rho_4(t, g^x(t), g^y(t))$, the constraint will be applied to the x-component, but it could have easily been applied to the y-component. The resulting constrained expressions are defined as,

$$x(t, g^x(t), g^y(t)) = g^x(t) + \phi_1^x(t)\rho_1(t, g^x(t)) + \phi_2^x(t)\rho_4(t, g^x(t), g^y(t))$$
$$y(t, g^y(t)) = g^y(t) + \phi_1^y(t)\rho_2(t, g^y(t)) + \phi_2^y(t)\rho_3(t, g^y(t)),$$

where the switching function equations are,

$$\phi_1^x(0) = 1, \quad -\int_0^3 \phi_1^x(t)\ \mathrm{d}t = 0$$
$$\phi_2^x(0) = 0, \quad -\int_0^3 \phi_2^x(t)\ \mathrm{d}t = 1$$
$$\phi_1^y(0) = 1, \quad \phi_1^y(2) - \phi_1^y(1) = 0$$
$$\phi_2^y(0) = 0, \quad \phi_2^y(2) - \phi_2^y(1) = 1.$$

Each switching function is again chosen to be a linear combination of support functions, where in this case the support functions are chosen as $s_1^x(t) = 1$ and $s_2^x(t) = t$ for $x(t)$ switching functions and $s_1^y(t) = 1$ and $s_2^y(t) = t$ for $y(t)$ switching functions. Thus, the

switching function can be concisely written as,

$$\begin{bmatrix} 1 & 0 \\ -3 & -\frac{9}{2} \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_3 \\ \alpha_2 & \alpha_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_5 & \alpha_7 \\ \alpha_6 & \alpha_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_1 & \alpha_3 \\ \alpha_2 & \alpha_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{2}{3} & -\frac{2}{9} \end{bmatrix} \qquad \begin{bmatrix} \alpha_5 & \alpha_7 \\ \alpha_6 & \alpha_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where $\phi_1^x(t) = \alpha_1 + \alpha_2 t$, $\phi_2^x(t) = \alpha_3 + \alpha_4 t$, $\phi_1^y(t) = \alpha_5 + \alpha_6 t$, and $\phi_2^y(t) = \alpha_7 + \alpha_8 t$. Substituting these values into the constrained expressions yields,

$$x(t, g^x(t), g^y(t)) = g^x(t) - \Big(1 - \frac{2}{3}t\Big)g^x(0) - \frac{2}{9}t\Big(4 - 2y(1, g^y(t)) + \int_0^3 g^x(\zeta)\, \mathrm{d}\zeta\Big)$$
$$y(t, g^y(t)) = g^y(t) - g^y(0) + t\Big(g^y(1) - g^y(2)\Big).$$

As in all previous examples, notice that regardless of how the free functions are chosen, the constraints will be satisfied exactly.

2.4.6 Infinite constraints

The derivation of constrained expression with infinite constraints was first solved by Johnston and Mortari [12] and requires greater attention to the selection of support functions. To understand this, first, consider a single infinite constraint on the value of the function as it approaches infinity,

$$\lim_{x \to \infty} y(x) = y_\infty.$$

When dealing with this single constraint, it should be straightforward to determine a simple constrained expression satisfying this constraint as,

$$y(x) = g(x) + \phi(x)\left(y_\infty - g(\infty)\right).$$

Here, the switching function can be simply defined as a constant value, $\phi(x) := 1$. As with all other types of constraints, the free function must be defined at the constraint. Therefore, $g(x)$ must be finite as $x \to \infty$. Additionally, as shown in the following example, the support functions must all be defined and finite at infinity.

Example 2.8: Infinite constraints

Consider a mixture of finite and infinite constraints as defined in the Falkner-Skan boundary layer equation [14],

$$y(0) = 0, \quad y_x(0) = 0, \quad \text{and} \quad y_x(\infty) = 1.$$

It follows that the projection functionals are,

$$\rho_1(x, g(x)) = -g(0), \quad \rho_2(x, g(x)) = -g_x(0), \quad \text{and} \quad \rho_3(x, g(x)) = 1 - g_x(\infty).$$

Let the support functions be,

$$s_1(x) = 1, \quad s_2(x) = x, \quad \text{and} \quad s_3(x) = \frac{x-1}{x+1}$$

Here, the selection of $s_3(x)$ is not arbitrary and is selected such that the last row of the support matrix is not zero and is therefore invertible. This leads to the system of equations.

$$\begin{bmatrix} s_1(0) & s_2(0) & s_3(0) \\ s_{1_x}(0) & s_{2_x}(0) & s_{3_x}(0) \\ s_{1_x}(\infty) & s_{2_x}(\infty) & s_{3_x}(\infty) \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which through matrix inversion leads to the solution of the α_{ij} coefficients

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 1 \\ 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

From this solution, the switching functions become,

$$\phi_1(x) = 1, \quad \phi_2(x) = \frac{1}{2} + \frac{x-1}{2(x+1)}, \quad \text{and} \quad \phi_3(x) = -\frac{1}{2} + x - \frac{x-1}{2(x+1)}$$

and the full constrained expression is ,

$$y(x, g(x)) = g(x) - g(0) + \Big(\frac{1}{2} + \frac{x-1}{2(x+1)}\Big)\Big(-g_x(0)\Big) + \Big(-\frac{1}{2} + x - \frac{x-1}{2(x+1)}\Big)\Big(1 - g_x(\infty)\Big)$$

With this example, we conclude our exploration of the implications and capabilities of the reformulation of the TFC approach spurred by the switching-projection form. These simply applied the techniques and loosely defined such terms as constrained expression, switching function, projection functional, etc., without much mathematical rigor. Chapter 3 looks to explicitly define all terms used; however, before doing this, it is important to highlight two other constrained types (inequality constraints and weighted-constraints), which are simply an extension of the constrained expression produced above.

2.5 Extension to inequality constraints

This section is referred to as an extension to inequality constraints since the following theory relies on the earlier sections. Inequality type constraints were first explored in John-

ston, Leake, Efendiev, and Mortari [15] and Johnston, Leake, and Mortari [16]; however, this dissertation provides invaluable updates from these two works.

To begin, let us consider a simple case with only one, continuous upper-bound inequality constraint defined on the domain $x \in [a, b]$. Let that constraint be given by the function $f_u(x)$ such that a function $y(x)$ satisfies this constraint if,

$$y(x) \leq f_u(x), \qquad \forall\ x \in [a, b].$$

For any given function $g(x)$, we can subtract off the sections of $g(x)$ that are larger than the inequality constraint $f_u(x)$ by using the Heaviside step function,

$$\mathbb{1}(z_1, z_2) = \begin{cases} 0 & \text{if } z_1 < 0 \\ z_2 & \text{if } z_1 = 0 \\ 1 & \text{if } z_1 > 0 \end{cases}$$

where the derivative of the Heaviside step function is exactly zero for all z_1. Furthermore, the Heaviside step function reduces to a simple step function if $z_2 = 0$, and in those cases will be defined as $\mathbb{1}_0(z_1) := \mathbb{1}(z_1, 0)$. The Heaviside step function can be thought of as the functional form of a gate or switch, and can be used to subtract off the difference between $f_u(x)$ and $g(x)$ when $g(x) > f_u(x)$, but does not affect $g(x)$ when $g(x) \leq f_u(x)$. Mathematically, this can be written as,

$$y(x, g(x)) = g(x) + [f_u(x) - g(x)]\mathbb{1}_0\Big(g(x) - f_u(x)\Big), \tag{2.11}$$

where $y(x, g(x))$ now represents the family of all possible functions that satisfy the inequality constraint. Another term can be added to Equation (2.11) to accommodate a lower bound

inequality constraint as well, $f_\ell(x)$. This is shown in Equation (2.12).

$$y(x, g(x)) = g(x) + [f_u(x) - g(x)]\mathbb{1}_0\Big(g(x) - f_u(x)\Big) + [f_\ell(x) - g(x)]\mathbb{1}_0\Big(f_\ell(x) - g(x)\Big) \quad (2.12)$$

2.5.1 Combining inequality and equality constraints

The technique to embed equality and inequality constraints builds on the formulation given in the earlier sections on the TFC approach to equality constraints. For a problem subject to equality and inequality constraints, let the TFC constrained expression for just the equality constraints be given by $\hat{y}(x, g(x))$. As per the univariate TFC, $\hat{y}(x, g(x))$ will represent the family of all possible functions that satisfy the equality constraints. Then, we exchange $g(x)$ in Equation (2.12) with $\hat{y}(x, g(x))$, as shown in Equation (2.13), to project $\hat{y}(x, g(x))$ onto the set of functions that satisfy the inequality constraints. It must be noted that this approach is limited to point equality constraints — derivative, integral, or component constraints cannot be combined with inequality constraints.

$$\begin{aligned} y(x, g(x)) = \hat{y}(x, g(x)) &+ [f_u(x) - \hat{y}(x, g(x))]\,\mathbb{1}_0\Big(\hat{y}(x, g(x)) - f_u(x)\Big) \\ &+ [f_\ell(x) - \hat{y}(x, g(x))]\,\mathbb{1}_0\Big(f_\ell(x) - \hat{y}(x, g(x))\Big) \end{aligned} \quad (2.13)$$

The resultant functional, $y(x, g(x))$, is now the TFC constrained expression representing all possible functions that satisfy both the equality constraints and inequality constraints of the problem.

Example 2.9: Numerical example of inequality constraints

Now, to analyze the expressions provided by Equations (2.12) and (2.13), a numerical test was constructed where the free function $g(x)$ and the inequality constraints were randomly generated through a linear expansion of m Chebyshev polynomials such

that,

$$g(x) = \sum_{i=0}^{m-1} a_i \, T_i(x), \tag{2.14}$$

where a_i are random coefficients $a_i \sim N(0,1)$ and $T_i(x)$ are the individual terms of the Chebyshev polynomials. Figure 2.1 shows Equation (2.12) subject to random inequality bounds and random values of $g(x)$. Furthermore, Figure 2.2 shows the application Equation (2.13) to both inequality and randomly generated equality point constraints. In both plots, the inequality constraints are shown as dotted black lines, the functions are shown as colored lines, and the three equality constraints are shown as black points.

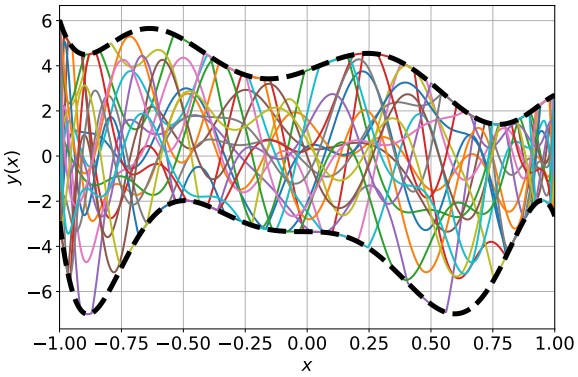


Figure 2.1: TFC constrained expression for inequality constraints only.

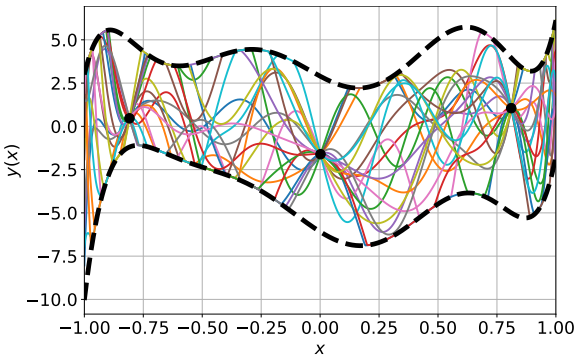


Figure 2.2: TFC constrained expression for equality and inequality constraints.

2.5.2 Keep-out zones

Using the univariate formulation of the TFC method subject to inequality constraints, a technique can be constructed for keep-out zones by augmenting Equation (2.12) or (2.13). This approach requires the constrained expression to be split into multiple constrained expressions for each possible path.

Example 2.10: Keep-out zones

As a simple example, let us consider solving all possible trajectories subject to upper and lower inequality constraints such that $f_u(x) = 1$, $f_\ell(x) = -1$ and avoiding an interior box defined by the coordinates, $A(-0.25, -0.25)$, $B(-0.25, 0.25)$, $C(0.25, 0.25)$, and $D(0.25, -0.25)$ as detailed in Figure 2.3.

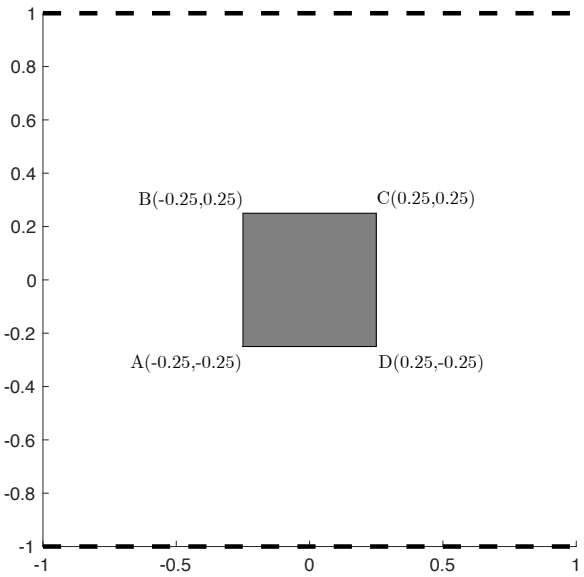


Figure 2.3: Keep-out box example.

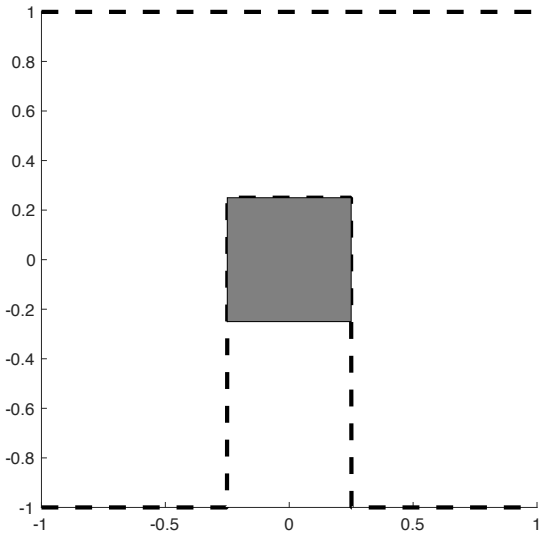


Figure 2.4: Upper path of keep-out box example.

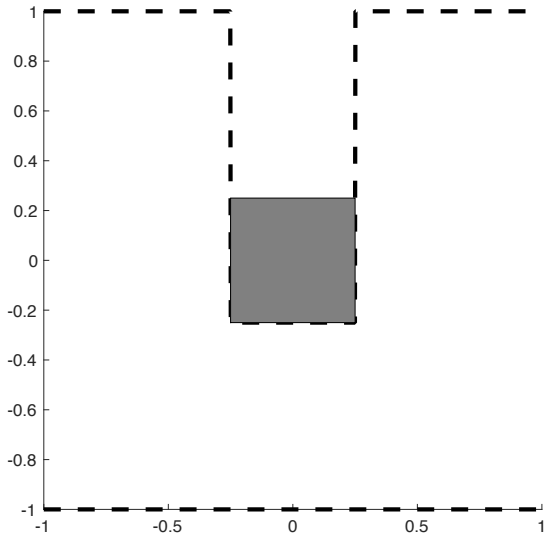


Figure 2.5: Lower path of keep-out box example.

In order to accommodate these constraints, we can split the problem into two individual problems that follows the formulation of the earlier sections. Therefore, we consider the upper path defined in Figure 2.4 and the lower path defined in Figure 2.5. First, to satisfy the upper path, the lower boundary needs to be augmented by the function defining the box which takes the form,

$$f_\ell(x) = \begin{cases} -1, & \text{if} \quad x < -0.25 \\ 0.25, & \text{if} \quad -0.25 \leq x \leq 0.25 \\ -1, & \text{if} \quad x > 0.25 \end{cases}$$

It then follows that for the lower path, the upper boundary is augmented by the function of the lower portion of the box such that,

$$f_u(x) = \begin{cases} 1, & \text{if} \quad x < -0.25 \\ -0.25, & \text{if} \quad -0.25 \leq x \leq 0.25 \\ 1, & \text{if} \quad x > 0.25 \end{cases} \cdot$$

Searching over both constrained expressions produces all the possible trajectories around the object. This method is analyzed by expressing $g(x)$ by Equation (2.14) and again defining $\boldsymbol{\xi} \sim \mathcal{N}(0, I_{m\times m})$. In addition to the single box example (Figure 2.3), Figures 2.6 - 2.13 experiment with differing geometries and configurations of constraints.

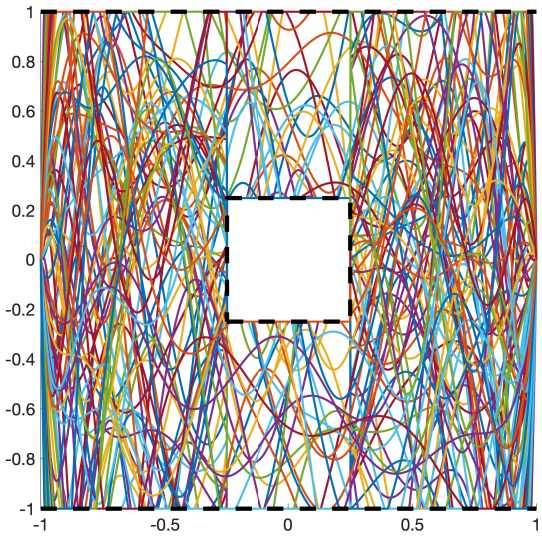


Figure 2.6: Single box.

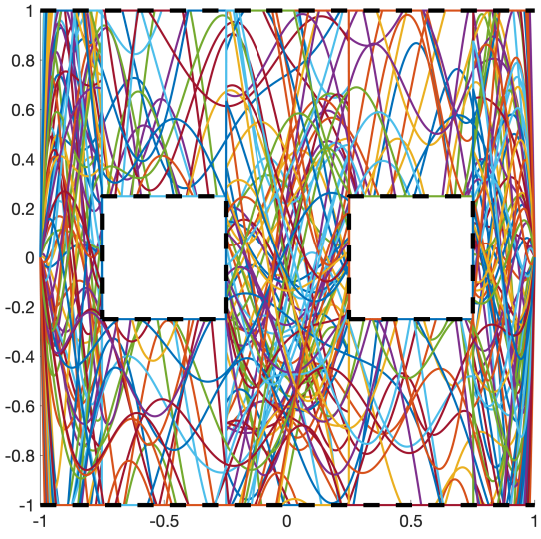


Figure 2.7: Two boxes horizontally arranged.

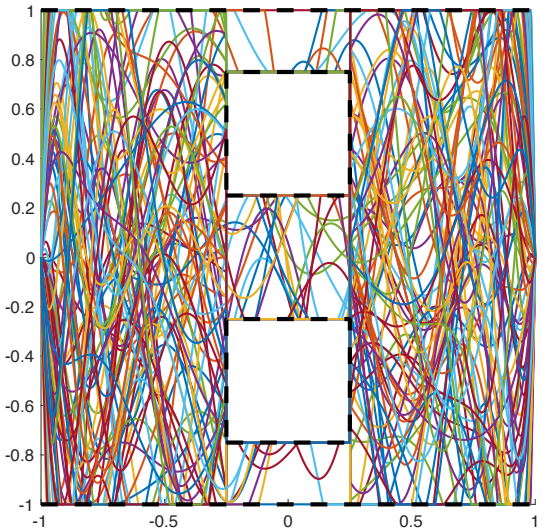


Figure 2.8: Two boxes vertically arranged.

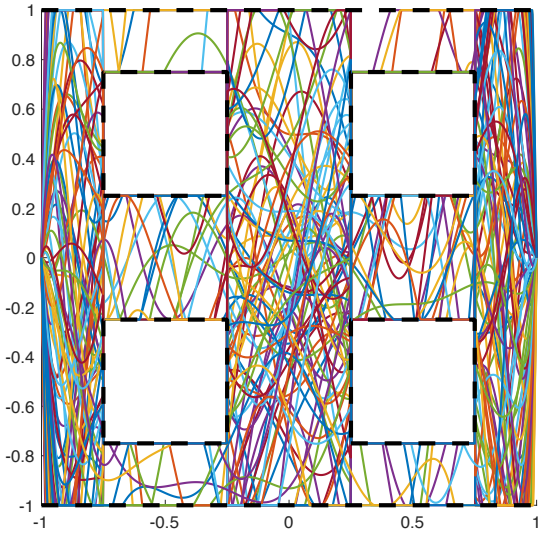


Figure 2.9: Four boxes.

Figures 2.6-2.9 provide multiple different keep-out box structures, including two horizontally arranged, two vertically arranged, and four equally spaced boxes.

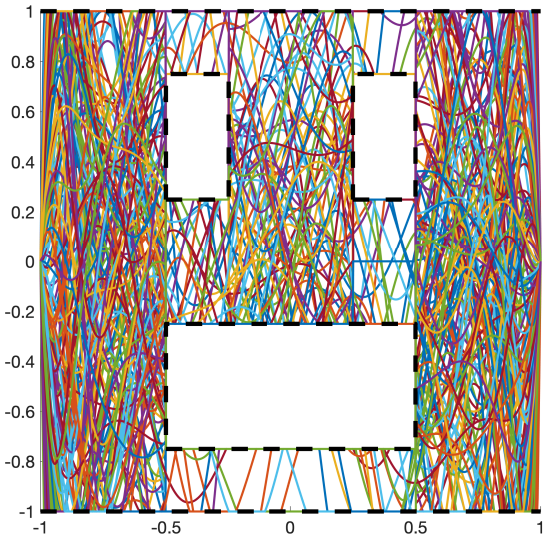


Figure 2.10: Different rectangles.

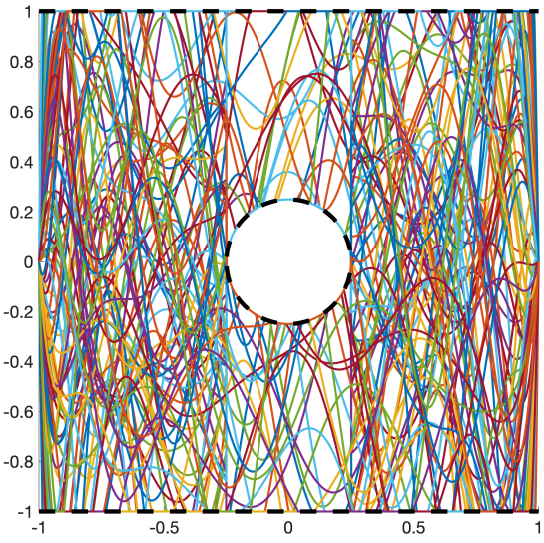


Figure 2.11: One circle.

Figures 2.10 through 2.13 look to push this method to unequally spaced rectangles, circles, and keep-out zones defined by an image. Two important things must be noted about this technique: 1) although the function defining the absolute lower and upper bounds in the example were constrained to $[-1, +1]$, these can be defined by any function similar to the bounds provided in Figures 2.1 and 2.2; 2) the major drawback of this method is that the search space scales with the number of possible trajectories, and therefore, the number of constrained expressions also increases. This implies that any optimization technique using this structure would produce the optimal trajectory for each path. As the number of paths increases drastically, this could become computationally expensive. Additionally, regardless of this method's flexibility, since it is only one-dimensional $y(x, g(x))$, it cannot be used in path planning problems. The next sections explores a two-dimensional, parametric space formulation where $x(t)$ and $y(t)$.

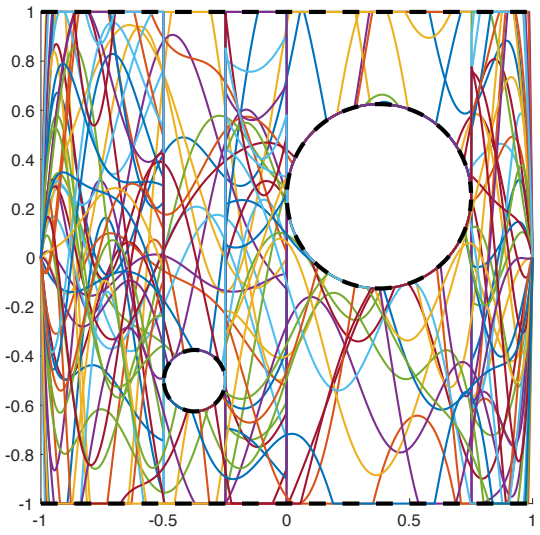


Figure 2.12: Two circles.

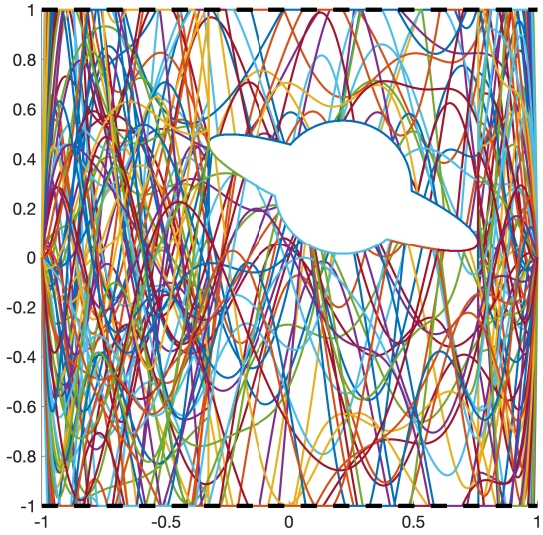


Figure 2.13: Random object.

2.5.3 Toward 2D inequality constraints

For this theory to be extended for path planning, the constrained expressions must be defined parametrically. For simplicity, let us consider a keep-out box defined by $x_\ell(t)$, $x_u(t)$, $y_\ell(t)$, and $y_u(t)$ as defined in Figure 2.14. In general, the keep-out zone could be dynamic; however, for now, let us consider the simple example of a static rectangular keep-out region. For this formulation let's define the path in terms of parametric variable t, using the func-

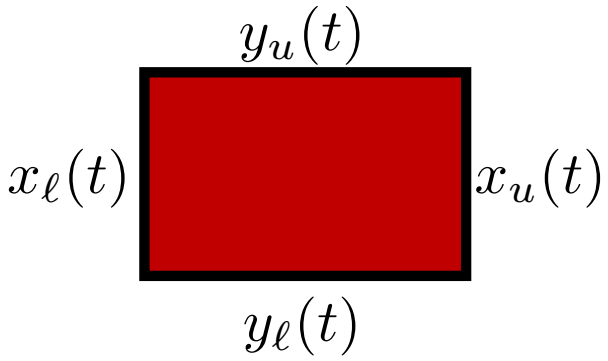


Figure 2.14: Conceptual keep-out box.

tionals $x(t, g(t))$ and $y(t, g(t))$ for x-position and y-position respectively. Associated with these two functions we use the free functions $g^x(t)$ and $g^y(t)$ in defining the constrained expression. Using the TFC method the constrained expression for $x(t, g^x(t))$ and $y(t, g^y(t))$ are as follows,

$$\begin{aligned} x(t, g^x(t)) = g^x(t) &+ \left[x_\ell(t) - g^x(t)\right]\left[\mathbb{1}_0(\varphi_1^y)\mathbb{1}_0(\varphi_2^y)\mathbb{1}_0(\varphi_3^x)\mathbb{1}_0(\varphi_2^x)\right] \\ &+ \left[x_u(t) - g^x(t)\right]\left[\mathbb{1}_0(\varphi_1^y)\mathbb{1}_0(\varphi_2^y)\mathbb{1}_0(-\varphi_3^x)\mathbb{1}_0(\varphi_1^x)\right], \end{aligned} \tag{2.15}$$

$$\begin{aligned} y(t, g^y(t)) = g^y(t) &+ \left[y_\ell(t) - g^y(t)\right]\left[\mathbb{1}_0(\varphi_1^x)\mathbb{1}_0(\varphi_2^x)\mathbb{1}_0(\varphi_3^y)\mathbb{1}_0(\varphi_2^y)\right] \\ &+ \left[y_u(t) - g^y(t)\right]\left[\mathbb{1}_0(\varphi_1^x)\mathbb{1}_0(\varphi_2^x)\mathbb{1}_0(-\varphi_3^y)\mathbb{1}_0(\varphi_1^y)\right]. \end{aligned} \tag{2.16}$$

The functions of φ (referred to as pseudo-switching functions due to their similarity with the true switching functions defined in the prior sections) are defined in Table 2.1, where c is replaced with either the component x or y. The constrained expressions in Equations (2.15)

Table 2.1: Pseudo-switching functions for Heaviside functions.

$\varphi_1^c(t)$	$\varphi_2^c(t)$	$\varphi_3^c(t)$
$c_u(t) - g^c(t)$	$g^c(t) - c_\ell(t)$	$\frac{c_u(t) + c_\ell(t)}{2} - g^c(t)$

and (2.16) are populated by three specific terms, and the interpretation for the x-component constrained expression is detailed below (note, the y-component constrained expression is of the same structure):

- The first term is the free functions for the x-component.
- The second term deals with the projection of the lower boundary and has four sigmoid functions as inputs to a 4-way AND gate that is true if and only if the following conditions are met:
  - $\varphi_1^y(t)$: the current path's y-position is less than $y_u(t)$
  - $\varphi_2^y(t)$: the current path's y-position is greater than $y_\ell(t)$
  - $\varphi_3^x(t)$: the current path's x-position is less than the average value of $x_u(t)$ and $x_\ell(t)$
  - $\varphi_2^x(t)$: the current path's x-position is greater than $x_\ell(t)$

  If these four conditions are true, than the current path is inside of the box and closer to the $x_\ell(t)$ line than the $x_u(t)$ line. In this case, the line is projected onto the $x_\ell(t)$ line by adding the difference between $x_\ell(t)$ and $g^x(t)$ to $g^x(t)$.

- The third term functions in a similar way to the second term, except in this case it deals with the projection of the upper boundary and has four sigmoid functions as inputs to a 4-way AND gate that is true if and only if the following conditions are met:
  - $\varphi_1^y(t)$: the current path's y-position is less than $y_u(t)$
  - $\varphi_2^y(t)$: the current path's y-position is greater than $y_\ell(t)$
  - $-\varphi_3^x(t)$: the current path's x-position is greater than the average value of $x_u(t)$ and $x_\ell(t)$
  - $\varphi_1^x(t)$: the current path's x-position is greater than $x_u(t)$

  If these four conditions are true, than the current path is inside of the box and closer to the $x_u(t)$ line than the $x_\ell(t)$ line. In this case, the line is projected onto the $x_u(t)$ line by adding the difference between $x_u(t)$ and $g^x(t)$ to $g^x(t)$.

Example 2.11: 2D inequality constraints

In order to analyze this technique, a numerical test was constructed using the constrained expressions given by Equations (2.15) and (2.16) where the terms $g^x(t)$ and $g^y(t)$ were defined according to Equation (2.14) with $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma^2 I_{m\times m})$ where $\sigma = 0.1$. The following tests show an example of single box path avoidance and multiple object path avoidance with boundary conditions such that $(x(t_0), y(t_0)) = (-1, -1)$ and $x(t_f), y(t_f)) = (+1, +1)$. The trajectories of both tests are shown in Figures 2.15 and 2.16. It can be seen in both tests the trajectories avoid the boundary displayed by the dashed black line. Additionally, the initial and final constraints on position are always met exactly. Going further, Figure 2.17 shows specifically the "smooth" trajectories that avoid the keep-out zones.

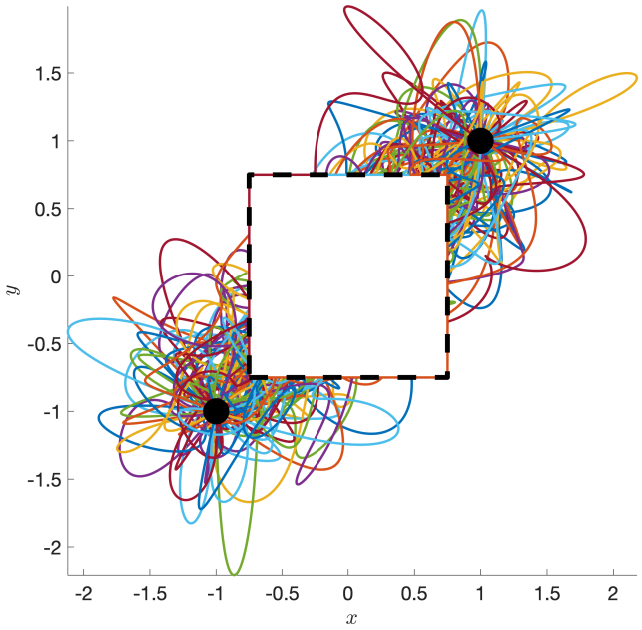


Figure 2.15: Keep-out box in parametric space.

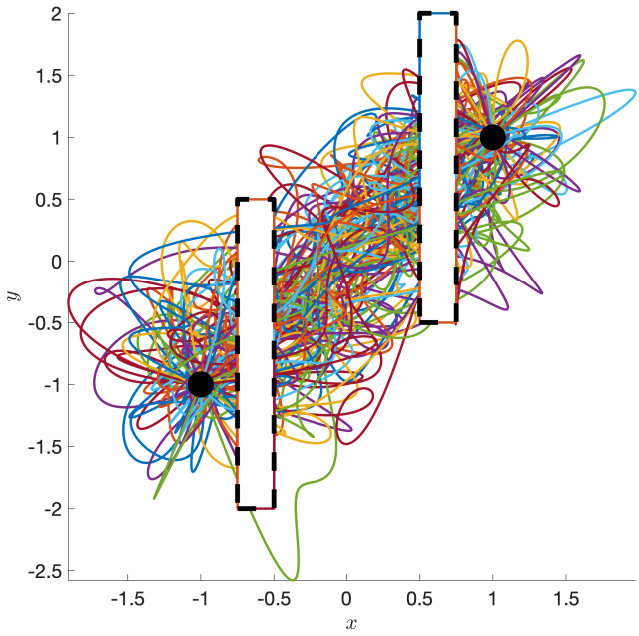


Figure 2.16: Multiple keep-out zones for parametric formulation.

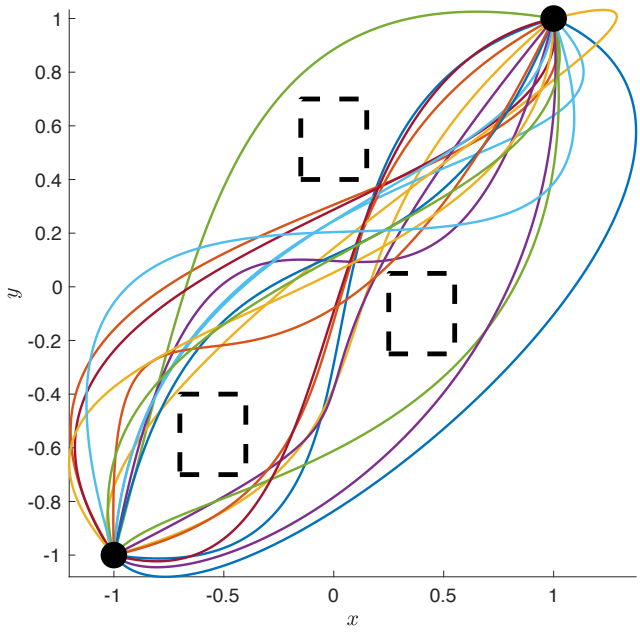


Figure 2.17: “Smooth” trajectories avoiding three box keep-out zones.

Although the test shows favorable results, there are potential issues when applying this formulation to optimization problems, namely path planning problems. For example, this method will try to project lines inside the box towards one of the corners. Since the Heaviside functions act as the switches in this problem, there could be cases where lines “snap" to the corners.

2.6 Over-constrained problems

As we have seen, the TFC framework can incorporate any linear constraints like those developed in the previous sections. Figure 2.18 provides an outline that distinguishes the TFC approach from classical methods in interpolation and least-squares. In the prior development of TFC, the number of the support function was equal to the number of constraints incorporated. It was shown to be a general interpolation approach that described *all* functions passing through k constraints. This section’s theory, highlighted in the grey box in Figure 2.18, combines this general interpolation method with a weighted least-squares technique for the constraints. Doing this allows for a constrained expression to be derived where

the number of constraints is greater than the number of support functions, $s_i(x)$, producing a *weighted* constrained expression. Using this expression produces a family of functions minimizing the weighted sum of squares of the constraints. This extension then provides the framework for the solution of over-constrained differential equations (a topic discussed in Chapter 4).

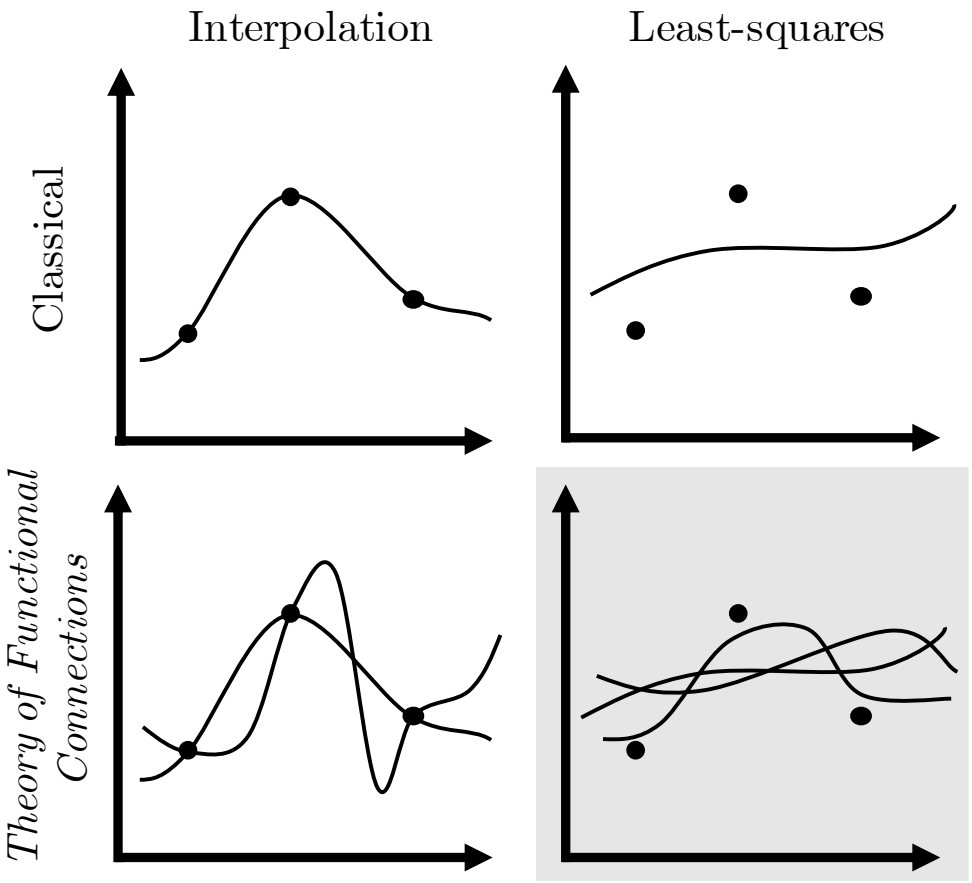


Figure 2.18: General illustration of classic and TFC approaches for interpolation and least-squares.

2.6.1 Two constraints in one degree of freedom

Consider a constrained expression such that,

$$y(x, g(x)) = g(x) + \varphi_1(x)\rho_1(x, g(x)) + \varphi_2(x)\rho_2(x, g(x)),$$

Note, in this expression, the $\varphi_i(x)$ is used for the switching functions because for the over-constrained cases these functions do not act like the switching functions, $\phi_i(x)$, discussed earlier. However, as will be seen later, the function $\varphi_i(x)$ can collapse to $\phi_i(x)$. Moving forward, it is desired that this function be subject to two constraints such that,

$$\begin{cases} y^{(i)}(x_1) = y_1^{(i)} \\ y^{(j)}(x_2) = y_2^{(j)} \end{cases} \qquad \text{where} \qquad i, j \in \mathbb{Z}.$$

First, consider the support function as $s(x)$ which will be evaluated at both constraint locations. Applying TFC produces an over-constrained system since there are two constraints but only one support function,

$$\begin{Bmatrix} s^{(i)}(x_1) \\ s^{(j)}(x_2) \end{Bmatrix} \begin{Bmatrix} \alpha_1 & \alpha_2 \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Therefore, this system can be solved by a weighted least-squares technique where W represents a diagonal matrix of the relative weights.

$$W\mathbb{S} \begin{Bmatrix} \alpha_1 & \alpha_2 \end{Bmatrix} = W$$

This system is then solved for the α coefficients just like in the traditional TFC approach,

$$\begin{Bmatrix} \alpha_1 & \alpha_2 \end{Bmatrix} = \left(\mathbb{S}^{\mathrm{T}} W \mathbb{S}\right)^{-1} \mathbb{S}^{\mathrm{T}} W$$

which leads to the expressions

$$\varphi_1(x) = s(x)\alpha_1 = \frac{s(x)s^{(i)}(x_1)w_1}{(s^{(i)}(x_1))^2 w_1 + (s^{(j)}(x_2))^2 w_2}$$
$$\varphi_2 = s(x)\alpha_2 = \frac{s(x)s^{(j)}(x_2)w_2}{(s^{(i)}(x_1))^2 w_1 + (s^{(j)}(x_2))^2 w_2}$$

The final constrained expression is realized as,

$$y(x, g(x)) = g(x) + \left[\frac{s(x)s^{(i)}(x_1)w_1}{(s^{(i)}(x_1))^2 w_1 + (s^{(j)}(x_2))^2 w_2}\right]\left(y_1^{(i)} - g^{(i)}(x_1)\right) + \left[\frac{s(x)s^{(j)}(x_2)w_2}{(s^{(i)}(x_1))^2 w_1 + (s^{(j)}(x_2))^2 w_2}\right]\left(y_2^{(j)} - g^{(j)}(x_2)\right). \tag{2.17}$$

Yet, there remains some conditions on the functions of $g(x)$ and $s(x)$. First, the function $g(x)$ must be differentiable up to the i-th and j-th derivative. Additionally, by analyzing terms $\varphi_1(x)$ and $\varphi_2(x)$, it can be seen that information on the constraints is lost when $\varphi_1(x)$ or $\varphi_2(x)$ becomes zero. Therefore, the support function $s(x)$ must be selected such that $s^{(i)}(x_1) \neq 0$ and $s^{(j)}(x_2) \neq 0$. Let us now consider the weighting scheme $w_1 = 1$ and $w_2 = 0$. In this case, Equation (2.17) reduces to a familiar form,

$$y(x, g(x)) = g(x) + \frac{s(x)}{s^{(i)}(x_1)}(y_1^{(i)} - g^{(i)}(x_1)),$$

which represents one constraint at one point. With this is example in mind, the following sections explore the characteristics of the *weighted* constrained expression for multiple points.

2.6.2 Weighted constraints at two points

Example 2.12: Weight-constrained expression for two points

As an example, let us consider constraints at two points such that,

$$y(x_1) = y_1 \quad \text{and} \quad y(x_2) = y_2,$$

which implies that $i = j = 0$. For these constraints, Equation (2.17) reduces to,

$$\begin{aligned} y(x, g(x)) = g(x) &+ \left[\frac{s(x)\, s(x_1)\, w_1}{s(x_1)^2\, w_1 + s(x_2)^2\, w_2}\right] \Big(y_1 - g(x_1)\Big) \\ &+ \left[\frac{s(x)\, s(x_2)\, w_2}{s(x_1)^2\, w_1 + s(x_2)^2\, w_2}\right] \Big(y_2 - g(x_2)\Big). \end{aligned}$$

The simplest definition of $s(x)$ such that $s(x_1) \neq 0$ and $s(x_2) \neq 0$ is $s(x) = 1$, leading to the equation,

$$y(x, g(x)) = g(x) + \left(\frac{w_1}{w_1 + w_2}\right) \Big(y_1 - g(x_1)\Big) + \left(\frac{w_2}{w_1 + w_2}\right) \Big(y_2 - g(x_2)\Big). \quad (2.18)$$

Analyzing this function, it can be seen that $g(x)$ is the only non-constant term in the equation and all other terms represent the relative weights of the prescribed constraints. Moreover, this equation represents every function that when evaluated at the constraint locations satisfies them relative to the prescribed weights w_1 and w_2. By setting $w_1 = 1$ and $w_2 = 0$, Equation (2.18) reduces to a constrained expression for one point,

$$y(x, g(x)) = g(x) + (y_1 - g(x_1)).$$

If $w_1 = 0$ and $w_2 = 1$ is selected, an equation satisfying $y(x_2) = y_2$ is obtained. This gives reason to believe that the weighted least-squares solution occupies the set of functions between these two absolute constraints. Keying in on this notion, let us explore the parametric

weight scheme,

$$W(\gamma) = \begin{bmatrix} 1-\gamma & 0 \\ 0 & \gamma \end{bmatrix}, \qquad \text{where} \qquad \gamma \in [0,1]. \tag{2.19}$$

Using these weights, Equation (2.18) becomes,

$$y(x, g(x)) = g(x) + (1-\gamma)\Big(y_1 - g(x_1)\Big) + \gamma\Big(y_2 - g(x_2)\Big). \tag{2.20}$$

Equation (2.20) was analyzed for multiple values of $g(x)$ over the range $x \in [-5, +5]$. The results in Figure 2.19 show that for each function, varying γ corresponds to translating the free function between the two prescribed constraints. Using $w_1 = w_2 = w$ (constraints

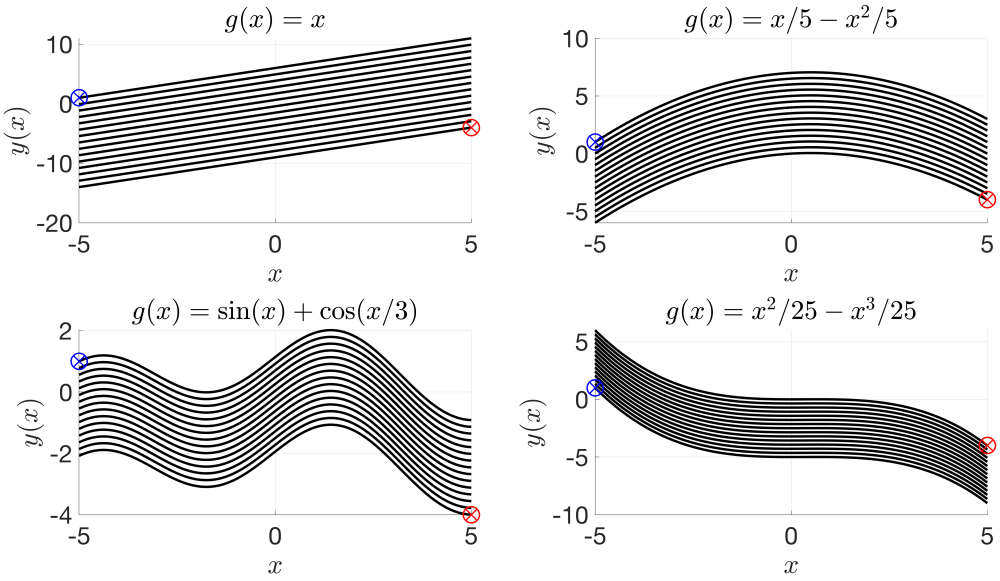


Figure 2.19: Analysis of Equation (2.20) for varying values of $g(x)$. It follows that as γ increases from 0 to 1, the function translates between the constraint conditions.

equally weighted), Equation (2.18) becomes,

$$y(x, g(x)) = g(x) + \frac{1}{2}(y_1 - g(x_1)) + \frac{1}{2}(y_2 - g(x_2)). \tag{2.21}$$

Analyzing this equation, it is expected that the constraint will be met with the same *relative* error for any function chosen for $g(x)$. Figure 2.20 shows the results of 20 randomly generated functions (left plot) and the constraint errors (right plot). In Figure 2.20, this "constraint

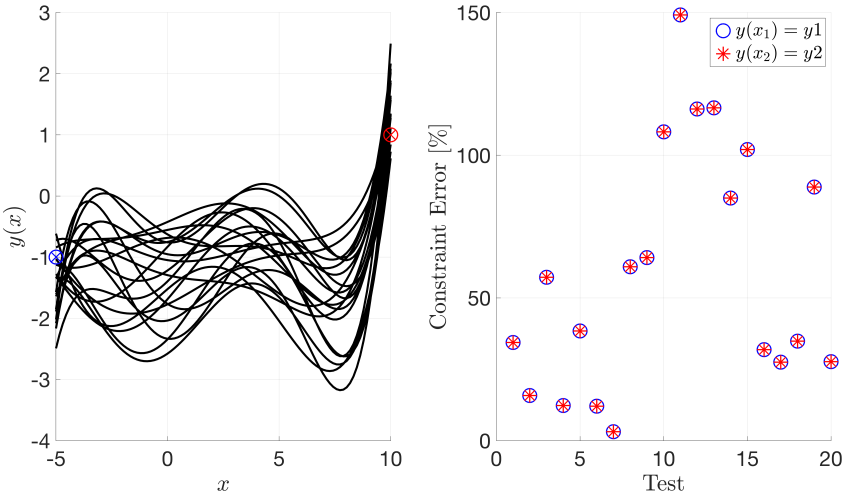


Figure 2.20: Analysis of Equation (2.21) for 20 randomly selected $g(x)$'s. The relative error between constraints is the same for every test.

error" is simply showing that since the projection functionals are equally weighted, the error from their imposed value (either y_1 or y_2) is the same.

2.6.3 Constraints on a function and its derivative

Example 2.13: Constraints on a function and its derivative

For further analysis, let us consider the case of constraints on a function and its derivative such that $i = 1$ and $j = 0$,

$$y_x(x_1) = y_{1_x} \quad \text{and} \quad y(x_2) = y_2,$$

which reduces Equation (2.17) to,

$$y(x, g(x)) = g(x) + \left[\frac{s(x)s_x(x_1)w_1}{(s_x(x_1))^2 w_1 + (s(x_2))^2 w_2}\right]\Big(y_{1_x} - g_x(x_1)\Big) + \left[\frac{s(x)s(x_2)w_2}{(s_x(x_1))^2 w_1 + (s(x_2))^2 w_2}\right]\Big(y_2 - g(x_2)\Big). \tag{2.22}$$

For this case, since $i = 1$ is the largest derivative, then s must be defined such that $s_x \neq 0$. The simplest case is to set $s = x$. Using this definition, Equation (2.22) becomes,

$$y = g + \left(\frac{w_1\, x}{w_1 + w_2\, x_2^2}\right)\Big(y_{1_x} - g_x(x_1)\Big) + \left(\frac{w_2\, x_2\, x}{w_1 + w_2\, x_2^2}\right)\Big(y_2 - g(x_2)\Big). \tag{2.23}$$

A similar test can be conducted for this case where W is defined according to Equation (2.19). For this particular case, let us define $g(x) = \sin(x) + \cos(x/3)$. Figure 2.21 shows the transformation from the initial derivative constraint to the final point constraint for various values of γ.

Additionally, Figure 2.21 shows the relative constraint error for each constraint as a function of the γ parameter. In the next section, this method will be applied to three constraints with two degrees of freedom.

2.6.4 Three constraints with two degrees of freedom

Now consider a constrained expression with two degrees of freedom defined as,

$$y(x, g(x)) = g(x) + \varphi_1(x)\rho_1(x, g(x)) + \varphi_2(x)\rho_2(x, g(x)), \tag{2.24}$$

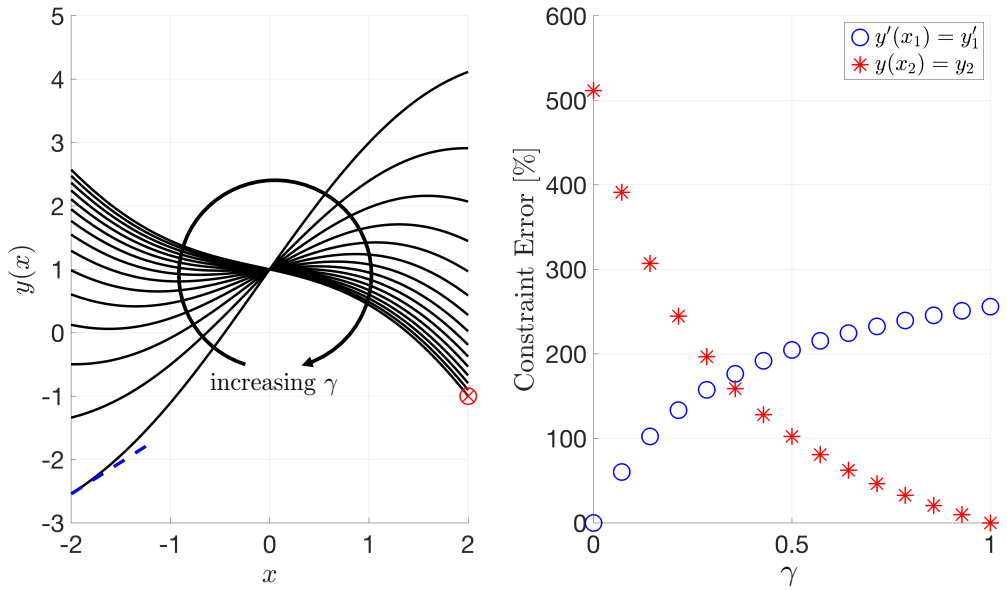


Figure 2.21: Equation (2.23) for varying weight values γ using the free function $g(x) = \sin x + \cos(x/3)$.

where $s_1(x)$ and $s_2(x)$ are assigned functions and the constraints are defined such that,

$$\begin{cases} y^{(i)}(x_1) = y_1^{(i)} \\ y^{(j)}(x_2) = y_2^{(j)} \qquad \text{where} \qquad i, j, k \in \mathbb{Z}. \\ y^{(k)}(x_3) = y_3^{(k)} \end{cases}$$

Applying these constraints leads to the system of equations,

$$W \begin{bmatrix} s_1^{(i)}(x_1) & s_2^{(i)}(x_1) \\ s_1^{(j)}(x_2) & s_2^{(j)}(x_2) \\ s_1^{(k)}(x_3) & s_2^{(k)}(x_3) \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \end{bmatrix} = W \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.25}$$

Although Equation (2.25) is expressed for three constraints, there is no upper limit on the number of constraints that can be incorporated.

Example 2.14: Three constraints with two degrees of freedom

Now, let us use the specific formulation, given by Equation (2.24), to derive an over-constrained expression with three point constraints. Incorporating these constraints ($i = j = k = 0$), the system of equations in Equation (2.25) reduces to,

$$W \begin{bmatrix} s_1(x_1) & s_2(x_1) \\ s_1(x_2) & s_2(x_2) \\ s_1(x_3) & s_2(x_3) \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \end{bmatrix} = W.$$

For this problem, let us define $s_1(x) = 1$, $s_2(x) = x$, and the diagonal weight matrix as,

$$W = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & w_3 \end{bmatrix}.$$

By solving the system using the weighted least-squares technique the over-constrained switching functions become,

$$\begin{aligned} \varphi_1(x) &= s_1(x)\alpha_{11} + s_2(x)\alpha_{21} = \frac{w_1}{D}\Big(w_2(x_2 - x)\Delta_{21} + w_3(x_3 - x)\Delta_{31}\Big) \\ \varphi_2(x) &= s_1(x)\alpha_{12} + s_2(x)\alpha_{22} = \frac{w_2}{D}\Big(w_1(x - x_1)\Delta_{21} + w_3(x_3 - x)\Delta_{32}\Big) \\ \varphi_3(x) &= s_1(x)\alpha_{13} + s_2(x)\alpha_{23} = \frac{w_3}{D}\Big(w_1(x - x_1)\Delta_{31} + w_2(x - x_2)\Delta_{32}\Big) \end{aligned}$$

such that $D := \Delta_{21}^2 w_1 w_2 + \Delta_{31}^2 w_1 w_3 + \Delta_{32}^2 w_2 w_3$ and $\Delta_{ij} := x_i - x_j$. Therefore, the over-constrained expression becomes

$$y(x, g(x)) = g(x) + \varphi_1(x)(y_1 - g(x_1)) + \varphi_2(x)(y_2 - g(x_2)) + \varphi_3(x)(y_3 - g(x_3)). \quad (2.26)$$

First, let us analyze the simplification when the weights are prescribed as $w_1 = w_2 = 1$

and $w_3 = 0$. Using these weights, Equation (2.26) reduces to,

$$y(x, g(x)) = g(x) + \left(\frac{x_2 - x}{x_2 - x_1}\right)(y_1 - g(x_1)) + \left(\frac{x - x_1}{x_2 - x_1}\right)(y_2 - g(x_2)) + [0](y_3 - g(x_3)),$$

which is the exact constrained expression obtained for the constraints $y(x_1) = y_1$ and $y(x_2) = y_2$ when using the methods developed earlier. Since the constraints are analytically embedded in Equation (2.26), the $g(x)$ function represents the solution space that satisfies the three constraints by weighted least-squares.

While this section simply introduces the over-constrained expression concept, in Section 4.10, we will look into using this framework to solve over-constrained differential equations.

3. A GENERAL FORMULATION OF THE UNIVARIATE *THEORY OF FUNCTIONAL CONNECTIONS*

This section rigorously defines the TFC constrained expression and provides some relevant proofs. First, the definition of a functional and properties of a functional are defined.

Definition 1

A functional, e.g. $f(x, g(x))$, has independent variable(s) and function(s) as inputs, and produces a function as an output.

Note that a function as defined here coincides with the computer science definition of a functional. One can think of a functional as a map for functions. That is, the functional takes a function, $g(x)$, as its input and produces a function, $f^*(x) = f(x, g(x))$ for any specified $g(x)$, as its output. Since this body of work is focused on constraint embedding, or in other words, functional interpolation, we will not concern ourselves with the domain/range of the input and output functions. Rather, we will discuss functionals only in the context of their potential input functions, hereon referred to as the domain of the functional, and potential output functions hereon referred to as the codomain of the functional.

Next, the definitions of injective, surjective, and bijective are extended from functions to functionals.

Definition 2

A functional, $f(x, g(x))$, is said to be injective if every function in its codomain is the image of at most one function in its domain.

Definition 3

A functional, $f(x, g(x))$, is said to be surjective if for every function in the codomain, $f^*(x)$, there exists at least one $g(x)$ such that $f^*(x) = f(x, g(x))$.

Definition 4

A functional, $f(x, g(x))$, is said to be bijective if it is both injective and surjective.

To elaborate, Figure 3.1 gives a graphical representation of each of these functionals, and examples of each of these functionals follow. Note that the phrase "smooth functions" is used here to denote continuous, infinitely differentiable, real-valued functions. Consider the functional $f(x, g(x)) = e^{-g(x)}$ whose domain is all smooth functions and whose codomain is all smooth functions. The functional is injective because for every $f^*(x)$ in the codomain there is at most one $g(x)$ that maps $f(x, g(x))$ to $f^*(x)$. However, the functional is not

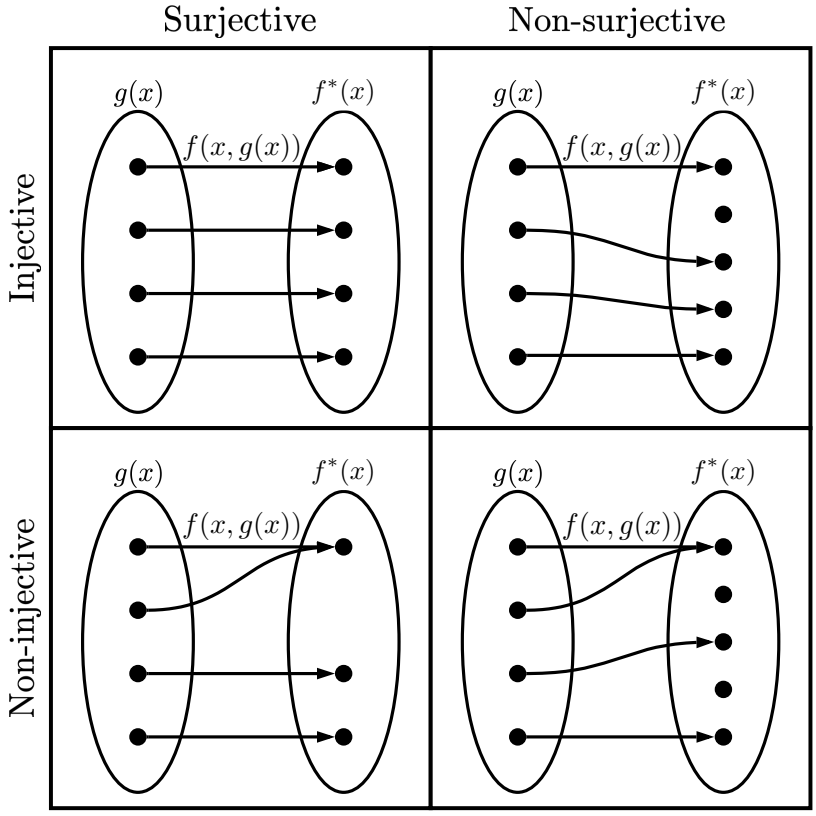


Figure 3.1: Graphical representation of injective and surjective functionals.

surjective, because the functional does not span the space of the codomain. For example, consider the desired output function $f^*(x) = -2$: there is no $g(x)$ that produces this output.

Next, consider the functional $f(x, g(x)) = g(x) - g(0)$ whose domain is all smooth functions and whose codomain is the set of all smooth functions $f^*(x)$ such that $f^*(0) = 0$. This functional is surjective because it spans the space of all smooth functions that are 0 when $x = 0$, but it is not injective. For example, the functions $g(x) = x$ and $g(x) = x + 3$ produce the same result, i.e., $f(x, x) = f(x, x + 3) = x$. Finally, consider the functional $f(x, g(x)) = g(x)$ whose domain is all smooth functions and whose codomain is all smooth functions. This functional is bijective, because it is both injective and surjective.

Also, the notion of projection is extended to functionals. Consider the typical definition of a projection matrix $P^n = P$ for some $n \in \mathbb{Z}^+$. In other words, when P operates on itself, it produces itself: a projection property for functionals can be defined similarly.

Definition 5

A functional is said to be a projection functional if it produces itself when operating on itself.

For example, consider a functional operating on itself, $f(x, f(x, g(x)))$. Then, if $f(x, f(x, g(x))) = f(x, g(x))$, then the functional is a projection functional. Note that proving $f(x, f(x, g(x))) = f(x, g(x))$ automatically extends to a functional operating on itself n times: for example, $f(x, f(x, f(x, g(x)))) = f(x, f(x, g(x))) = f(x, g(x))$, and so on.

Now that a functional and some properties of a functional have been defined, the notation used in the prior section can be leveraged to rigorously define TFC related concepts. First, it is useful to define the constraint operator, denoted by the symbol $\mathfrak{C}$.

Definition 6

The constraint operator, $\mathfrak{C}_i$, is a linear operator that, when operating on a function, returns the function evaluated at the i-th specified constraint.

As an example, consider the linear constraint $3 = 2y(2) + \pi y_{xx}(0)$, and suppose it is the first constraint in the set ($i = 1$). For this constraint, the constraint operator operates as

follows,

$$\mathfrak{C}_1[y(x)] = 2y(2) + \pi y_{xx}(0).$$

The constraint operator is a linear operator, as it satisfies the two properties of a linear operator:

1. $\mathfrak{C}_i[f(x) + g(x)] = \mathfrak{C}_i[f(x)] + \mathfrak{C}_i[g(x)]$
2. $\mathfrak{C}_i[ag(x)] = a\mathfrak{C}_i[g(x)]$

For example, again consider the linear constraint $3 = 2y(2) + \pi y_{xx}(0)$,

$$\mathfrak{C}_1[f(x) + g(x)] = \mathfrak{C}_1[f(x)] + \mathfrak{C}_1[g(x)] = 2f(2) + \pi f_{xx}(0) + 2g(2) + \pi g_{xx}(0)$$
$$\mathfrak{C}_1[af(x)] = a\mathfrak{C}_1[f(x)] = a\Big(2f(2) + \pi f_{xx}(0)\Big).$$

Naturally, the constraint operator has specific properties when operating on the support functions, switching functions, and projection functionals.

Property 5

The constraint operator acting on the support functions $s_j(x)$ produces the support matrix

$$\mathbb{S}_{ij} = \mathfrak{C}_i[s_j(x)].$$

For example, consider the two constraints, $y(1) = y(0)$ and $3 = 2y(2)+\pi y_{xx}(0)$. Applying the constraint operator,

$$\mathbb{S}_{ij} = \mathfrak{C}_i[s_j(x)] = \begin{bmatrix} \mathfrak{C}_1[s_1(x)] & \mathfrak{C}_1[s_2(x)] \\ \mathfrak{C}_2[s_1(x)] & \mathfrak{C}_2[s_2(x)] \end{bmatrix}$$
$$= \begin{bmatrix} s_1(1) - s_1(0) & s_2(1) - s_2(0) \\ 2s_1(2) + \pi s_{1_{xx}}(0) & 2s_2(2) + \pi s_{2_{xx}}(0) \end{bmatrix}.$$

In fact, the support matrix $\mathbb{S}_{ij}$ is simply the matrix multiplying the α_{ij}. Therefore, it follows that, $\mathbb{S}_{ij}\,\alpha_{jk} = \alpha_{ij}\,\mathbb{S}_{jk} = \delta_{ik}$, where δ_{ik} is the Kroneker delta, and the solution of the α_{ij} coefficients are precisely the inverse of the constraint operator operating on the support matrix.

Property 6

The constraint operator acting on the switching functions $\phi_j(x)$ produces the Kronecker delta.

$$\mathfrak{C}_i[\phi_j(x)] = \delta_{ij}$$

This property is just a mathematical restatement of the linguistic definition of the switching function given earlier. One can intuit this property from the switching function definition, since they evaluate to 1 at their specified constraint condition (i.e., $i = j$) and to 0 at all other constraint conditions (i.e., $i \neq j$).

Using this definition of the constraint operator, one can define the projection functional in a compact and precise manner.

Definition 7

Let $g(x)$ be the free function where $g(x) : \mathbb{R} \to \mathbb{R}$, and let $\kappa_i \in \mathbb{R}$ be the numerical portion of the i^{th} constraint. Then,

$$\rho_i(x, g(x)) = \kappa_i - \mathfrak{C}_i[g(x)].$$

Again, consider the linear constraint $3 = 2y(2) + \pi y_{xx}(0)$. The projection function is,

$$\begin{aligned}\rho_1(x, g(x)) &= \kappa_1 - \mathfrak{C}_1[g(x)] \\ &= 3 - 2g(2) - \pi g_{xx}(0).\end{aligned}$$

Moving forward, we look to leverage the definitions and properties of the TFC formulation to prove a few aspects of the TFC constrained expression that will be useful during numerical

implementation.

Claim 1

For any function, $f(x)$, satisfying the constraints, there exists at least one free function, $g(x)$, such that the TFC constrained expression $y(x, g(x)) = f(x)$.

Proof: As highlighted in Properties 1, 2, 3, and 4, the projection functionals are equal to zero whenever $g(x)$ satisfies the constraints. Thus, if $g(x)$ is a function that satisfies the constraints, then the constrained expression becomes,

$$\begin{aligned} y(x, g(x)) &= g(x) + \rho_i(x, g(x))\, \phi_i(x) \\ &= g(x) + 0\, \phi_i(x) \\ &= g(x). \end{aligned}$$

Hence, by choosing $g(x) = f(x)$, the constrained expression becomes $y(x, f(x)) = f(x)$. Therefore, for any function satisfying the constraints, $f(x)$, there exists at least one free function $g(x) = f(x)$, such that the constrained expression is equal to the function satisfying the constraints, i.e., $y(x, f(x)) = f(x)$. ■

Claim 2

The TFC univariate constrained expression is a projection functional.

Proof: To prove Claim 2, one must show that $y(x, y(x, g(x))) = y(x, g(x))$. By definition, the constrained expression returns a function that satisfies the constraints. In other words, for any $g(x)$, $y(x, g(x))$ is a function that satisfies the constraints. From Claim 1, if the free function used in the constrained expression satisfies the constraints, then the constrained expression returns that free function exactly. Hence, if the constrained expression functional is given itself as the free function, it will simply

return itself. ■

Claim 3

For a given function, $f(x)$, satisfying the constraints, the free function, $g(x)$, in the TFC constrained expression $y(x, g(x)) = f(x)$ is not unique. In other words, the TFC constrained expression is a surjective functional.

Proof: Consider the free function choice $g(x) = f(x)+\beta_j\, s_j(x)$ where β_j are scalar values on $\mathbb{R}$ and $s_j(x)$ are the support functions used to construct the switching functions $\phi_i(x)$.

$$y(x, g(x)) = g(x) + \phi_i(x)\, \rho_i(x, g(x)).$$

Substituting the chosen $g(x)$ yields,

$$y(x, g(x)) = f(x) + \beta_j\, s_j(x) + \phi_i(x)\, \rho_i(x, f(x) + \beta_j\, s_j(x)).$$

Now, according to Definition 7 of the projection functional,

$$y(x, g(x)) = f(x) + \beta_j\, s_j(x) + \phi_i(x)\Big(\kappa_i - \mathfrak{C}_i[f(x) + \beta_j\, s_j(x)]\Big).$$

Since the constraint operator $\mathfrak{C}_i$ is a linear operator,

$$y(x, g(x)) = f(x) + \beta_j s_j(x) + \phi_i(x)\Big(\kappa_i - \mathfrak{C}_i[f(x)] - \mathfrak{C}_i[s_j(x)]\beta_j\Big).$$

Since $f(x)$ is defined as a function satisfying the constraints, then $\mathfrak{C}_i[f(x)] = \kappa_i$, and,

$$y(x, g(x)) = f(x) + \beta_j s_j(x) - \phi_i(x)\mathfrak{C}_i[s_j(x)]\beta_j.$$

Now, according to Property 5 of the constraint operator, and by decomposing the switching functions $\phi_i(x)$,

$$y(x, g(x)) = f(x) + \beta_j \, s_j(x) - \alpha_{ki} \, s_k(x) \mathbb{S}_{ij} \, \beta_j.$$

Collecting terms results in,

$$y(x, g(x)) = f(x) + \beta_j \Big(\delta_{jk} - \alpha_{ki} \, \mathbb{S}_{ij} \Big) s_k(x).$$

However, $\mathbb{S}_{ki}\alpha_{ij} = \delta_{kj}$ because α_{ij} is the inverse of $\mathbb{S}_{ki}$. Therefore, by the definition of inverse, $\mathbb{S}_{ki}\alpha_{ij} = \alpha_{ki}\mathbb{S}_{ij} = \delta_{kj}$, and thus,

$$y(x, g(x)) = f(x) + \beta_j \Big(\delta_{jk} - \delta_{jk} \Big) s_k(x).$$

Simplifying yields the result,

$$y(x, g(x)) = f(x),$$

which is independent of the $\beta_j s_j(x)$ terms in the free function. Therefore, the free function is not unique. ■

Notice that the non-uniqueness of $g(x)$ depends on the support functions used in the constrained expression, which has an immediate consequence when using constrained expressions in optimization. If any terms in $g(x)$ are linearly dependent to the support functions used to construct the constrained expression, their contribution is negated and thus arbitrary. For some optimization techniques, it is critical that the linearly dependent terms that do not contribute to the final solution be removed; else, the optimization technique becomes impaired. For example, prior research focused on using this method to solve ODEs [17, 18] through a basis expansion of $g(x)$ and least-squares, and the basis terms linearly dependent

to the support functions had to be omitted from $g(x)$ to maintain full rank matrices in the least-squares.

The previous proofs coupled with the functional definitions and properties given earlier provide a more rigorous definition for the TFC constrained expression: the TFC constrained expression is a surjective, projection functional whose domain is the space of all real-valued functions that are defined at the constraints and whose codomain is the space of all real-valued functions that satisfy the constraints. It is surjective because it spans the space of all functions that satisfy the constraints, its codomain, based on Claim 1, but is not injective because Claim 3 shows that functions in the codomain are the image of more than one function in the domain: the functional is thus not bijective either because it is not injective. Moreover, the TFC constrained expression is a projection functional, as shown in Claim 2.

This formal definition of the univariate TFC is simple yet powerful, as its claims apply to any combination of the constraints introduced previously, and it can easily be extended to n-dimensions; The multivariate TFC is the topic of Carl Leake's dissertation [19] and was first introduced in Leake, Johnston, and Mortari [4]

Part 2
Application

Can you truly appreciate how special or beautiful something is if you don't know what it took to get it? If you never had to work for it?

— Unravel, *ColdWood Interactive*

4. APPLICATION TO THE SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

In the prior sections, we developed a technique to derive functionals, called constrained expressions, which represented all possible functions satisfying a given set of constraints. One of the obvious applications of these expressions is to the solution of differential equations. In general, differential equations (DEs) are used as numerical models to describe physical phenomena throughout engineering and science. The solution of these equations is vital for design, predictive modeling, and optimization, and therefore, fast and accurate solutions are vital.

In the following section, the process to solve these equations using the TFC framework is introduced and used to solve various differential equations of varying complexity. Furthermore, while this work focuses explicitly on the solution of ordinary differential equations, the technique is easily extended to partial differential equations and was first covered in detail in Leake, Johnston, and Mortari [4] and Schiassi et al. [20]. Again, for a complete development of multivariate TFC and the solution of partial differential equations, the reader is directed to the dissertation of Carl Leake [19].

Moving forward, we must first understand the two main approaches used to solve these types of problems. First, due to the structure of some types of problems, a differential equation can sometimes be solved analytically, and thereby, admit a closed-form solution. However, in most practical applications, the differential equations to be solved are complex, and numerical techniques become important when a solution, albeit approximated, is needed.

4.1 Analytical methods to solve ODEs

As mentioned above, some differential equations can be solved analytically to provide a closed-form solution to the equations. This solution is exact and suffers no associated error; however, these solutions are limited to a class of differential equations and do not encom-

pass all differential equations. For example, for first-order differential equations, analytical techniques exist for the solutions of classes such as directly integrable, linear, separable, homogeneous, exact, and Bernoulli, etc. In fact, resources, including References [21, 22], provide an extensive list of closed-form solutions to many classes of ordinary differential equations. However, the advancement and widespread use of computers has increased the emphasis on research towards solving these equations numerically. Additionally, since many numerical models are associated with complex differential equations, numerical solutions are sometimes the only available avenue to solve the problem.

4.2 Numerical methods to solve ODEs

The techniques to solve (or approximate) DEs are littered throughout literature, spanning almost all science, engineering, and mathematics fields. To understand how the TFC based method fits into the existing literature, let us look into the most popular numerical methods to solve ODEs, summarized in the following sections.

4.2.1 Runge-Kutta family

Some of the most widely used techniques are based on the Runge-Kutta family of integrators. Examples of these integrators include lower-order methods such as the Euler Method (first-order), Midpoint Method (second-order), and the Runge-Kutta Method (fourth-order) [23]. To highlight the general idea of these approaches, let us look at an example of solving the ODE, $y_x = f(x, y)$ subject to $y(x_0) = y_0$.

Low order Runge-Kutta methods

Methods based on the Runge-Kutta method are forward-propagation schemes that, in general, rely on estimating the next value of the solution (i.e., the $k + 1$ value) by an approximation involving the evaluation of the function $f(x, y)$ and some step size. The specific propagation equations for Euler, Midpoint, and Runge-Kutta methods are provided below:

Euler Method

$$y_{k+1} = y_k + hk_1 + \mathcal{O}(h^2)$$

Midpoint Method

$$y_{k+1} = y_k + hk_2 + \mathcal{O}(h^3)$$

Runge-Kutta Method (RK4)

$$y_{k+1} = y_k + \frac{1}{6}h\Big(k_1 + 2k_2 + 2k_3 + k_4\Big) + \mathcal{O}(h^5)$$

where k is the current time step, $k+1$ is the next time step, and h is the step size. Additionally, $\mathcal{O}$ signifies the truncation order and is omitted in the numerical solution. In these equations, the values of k_1, k_2, k_3, and k_4 are intermediate calculations based on the order of the method, and are as follows,

$$\begin{aligned}
k_1 &= f(x_k, y_k) \\
k_2 &= f(x_k + \frac{h}{2}, y_k + \frac{h}{2}k_1) \\
k_3 &= f(x_k + \frac{h}{2}, y_k + \frac{h}{2}k_2) \\
k_4 &= f(x_k + h, y_k + hk_3).
\end{aligned}$$

A typical approach to solving differential equations using the Runge-Kutta method is the RK45 technique, which combines an RK4 and RK5 method to adaptively select the step size h. This technique, called the Runge-Kutta-Fehlberg method, compares the difference between the value obtained from the 4th order and 5th order method to determine the optimal step size h_{best}. A summary of the RK45 algorithm is summarized below.

Runge-Kutta-Fehlberg method

4th order Runge-Kutta method

$$^{(4)}y_{k+1} = y_k + \frac{25}{216}hk_1 + \frac{1408}{2565}hk_3 + \frac{2197}{4104}hk_4 - \frac{1}{5}hk_5$$

5th order Runge-Kutta method

$$^{(5)}y_{k+1} = y_k + \frac{16}{135}hk_1 + \frac{6656}{12825}hk_3 + \frac{28561}{56430}hk_4 - \frac{9}{50}hk_5 + \frac{2}{55}hk_6$$

where the pre-superscripts denote the order of the method, and the coefficients are as follows,

$$\begin{aligned}
k_1 &= f(x_k, y_k) \\
k_2 &= f\Big(x_k + \frac{h}{4}, y_k + \frac{h}{4}k_1\Big) \\
k_3 &= f\Big(x_k + \frac{3h}{8}, y_k + \frac{3}{32}hk_1 + \frac{9}{32}hk_2\Big) \\
k_4 &= f\Big(x_k + \frac{12}{13}h, y_k + \frac{1932}{2197}hk_1 - \frac{7200}{2197}hk_2 + \frac{7296}{2197}hk_3\Big) \\
k_5 &= f\Big(x_k + h, y_k + \frac{439}{216}hk_1 - 8hk_2 + \frac{3680}{513}hk_3 - \frac{845}{4104}hk_4\Big) \\
k_6 &= f\Big(x_k + \frac{h}{2}, y_k - \frac{8}{27}hk_1 + 2hk_2 - \frac{3544}{2565}hk_3 + \frac{1859}{4104}hk_4 - \frac{11}{40}hk_5\Big).
\end{aligned}$$

The optimal step size is then defined by

$$h_{\text{opt}} = h_{\text{last}}\left(\frac{\varepsilon h_{\text{last}}}{2|\,^{(4)}y_{k+1} - {}^{(5)}y_{k+1}|}\right)^{1/4} \approx \frac{\text{desired error}}{\text{actual error}}.$$

The above technique is similar to what is implemented in algorithms such as MATLAB's `ode45()` [24] and the Python package SciPy's `scipy.integrate.ode()` [25]. In many numerical tests in this chapter, we will use the RK45 solution as the baseline to compare against the TFC method in terms of speed and accuracy.

4.2.2 Gauss-Jackson

Another technique widely used in the astrodynamics community is the Gauss-Jackson method, a multistep predictor-corrector method. First introduced in a 1924 paper by Jackson [26], this technique has been further studied in References [27, 28]. In general, this method is a summed form of the Stormer-Cowell integrator [29].

In order to understand the fundamentals of this method, consider the ordinary differential equation of the form $y_{xx} = f(x, y, y_x)$. The Gauss-Jackson technique first predicts the solution value $y(x)$ for the next step and evaluates the function $f(x, y, y_x)$ at this point. Then, this predicted function value is added to the backpoints, i.e., prior calculated points. A corrector formula is utilized to revise this set of data and refine the prediction of $y(x)$. The general implementation of these algorithms can be grouped into two methods, 1) Predict-Evaluate-Correct (PEC) and 2) Predict-Evaluate-Correct-Evaluate (PECE), where the latter performs a second evaluation step to increase accuracy. Furthermore, these processes can perform additional iterations to meet some tolerance.

The following example box provides a summary of the major equations in the Gauss-Jackson method.

Gauss-Jackson method

Consider the ordinary differential equation where $y_{xx} = f(x, y)$, subject to the initial conditions $y(x_0) = y_0$ and $y_x(x_0) = y_{0_x}$. The Gauss-Jackson correction and prediction formulas are as follows where $H.O.T$ stands for higher order terms.

Gauss-Jackson corrector formula

$$y_k = h^2\Big[\nabla^{-2} y_{k_{xx}} + \Big(\frac{1}{12} - \frac{1}{240}\nabla^2 - \frac{1}{240}\nabla^3 - \frac{221}{60480}\nabla^4 + \ldots + H.O.T.\Big) y_{k_{xx}}\Big]$$

Gauss-Jackson predictor formula

$$y_{k+1} = h^2\Big[\nabla^{-2} y_{k_{xx}} + \Big(\frac{1}{12} + \frac{1}{12}\nabla + \frac{19}{240}\nabla^2 + \frac{3}{40}\nabla^3 + \ldots + H.O.T.\Big) y_{k_{xx}}\Big]$$

where ∇ is the backwards difference operator such that $\nabla f_k = f_k - f_{k-1}$. The higher-order difference operators can be easily derived and are provided in Reference [28]. Additionally, the predication and correction of the first derivative, y_x, is given by the summed Adams method as,

$$y_{k_x} = h\Big[\nabla^{-1} - \frac{1}{2} - \frac{1}{12}\nabla - \frac{1}{24}\nabla^2 - \frac{19}{720}\nabla^3 - \frac{3}{160}\nabla^4 \ldots + H.O.T.\Big) y_{k_{xx}}\Big]$$
$$y_{k+1_x} = h\Big[\nabla^{-1} + \frac{1}{2} + \frac{5}{12}\nabla + \frac{3}{8}\nabla^2 + \frac{251}{720}\nabla^3 + \frac{95}{288}\nabla^4 + \ldots + H.O.T.\Big) y_{k_{xx}}\Big].$$

When solving differential equations using the Gauss-Jackson method (and other predictor -corrector methods), the main hurdle is initialization. Since the initial conditions are given at some epoch x_0, there are no backpoints, and these must be calculated before the algorithm is used. One way to initialize these backpoints is to use a single-step integrator such as the Runge-Kutta methods described in the prior section.

4.2.3 Modified Chebyshev-Picard Iteration

Modified Chebyshev-Picard Iteration [30, 31, 32] is a path-length integral approximation that has been recently proven to be highly effective. This technique has been successfully applied to initial- and boundary-value problems in orbit propagation. The following summarizes the main parts of the method.

Modified Chebyshev-Picard Iteration

Given a differential equation

$$y_x = f(x, y)$$

where $y(x_0) = y_0$, the domain is first transformed to that of the closed interval of the Chebyshev polynomials [-1, +1],

$$x = \omega_1 + \omega z \quad \omega_1 = \frac{x_f + x_0}{2} \quad \omega_2 = \frac{x_f - x_0}{2}.$$

This transformation allows us to rewrite the differential equation as,

$$y_z = q(z, y) = \omega_2 f(\omega_1 + \omega_2 z, y).$$

The solution to this equation is provided by Picard iteration where,

$$y^i(z) = y_0 + \int_{-1}^{z} q(s, y^{i-1}(s))\, \mathrm{d}s \quad i = 1, 2, \ldots$$

Next, the state y^i and the integrand are approximated by a sum of Chebyshev polynomials with unknown coefficients discretized at $(N + 1)$ Chebyshev-Gauss-Lobatto (CGL) nodes such that,

$$z_j = \cos\left(\frac{j\pi}{N}\right) \quad j = 0, 1, 2 \ldots, N$$

The forcing function is approximated by Chebyshev polynomials through,

$$\begin{aligned} q(z, y^{i-1}(z)) &\approx \sum_{k=0}^{k=N}{}' F_k^{i-1} T_k(z) \\ &\equiv \frac{1}{2} F_0^{i-1} T_0(z) + F_1^{i-1} T_1(z) + F_2^{i-1} T_2(z) + \ldots F_N^{i-1} T_N(z). \end{aligned}$$

The discrete orthogonality of Chebyshev polynomials [33] allows for the direct computation of F_k,

$$\begin{aligned} F_k^{i-1} &= \frac{2}{N}\sum_{j=0}^{N}{}''q(z_j, y^{i-1}(z_j))T_k(z_j) \\ &= \frac{1}{N}q(z_0, y^{i-1}(z_0))T_k(z_0) + \\ &\quad \frac{2}{N}q(z_1, y^{i-1}(z_1))T_k(z_1) + \ldots + \frac{1}{N}q(z_N, y^{i-1}(z_N))T_k(z_N) \end{aligned}$$

where $\sum'$ denotes that the first term is halved and $\sum''$ represents that both the first and last terms are halved. Plugging this into the Picard iteration equation leads to,

$$y^i = y_0 + \sum_{r=0}^{N}{}'F_r^{i-1}\int_{-1}^{z} T_r(s)\,\mathrm{d}s \equiv \sum_{k=0}^{N}{}'\beta_k^i T_k(z)$$

where the updated equations for the coefficients are derived in detail in Reference [34] and summarized below,

$$\begin{aligned} \beta_k^i &= \frac{1}{2k}(F_{k-1}^{i-1} - F_{k+1}^{i-1}) \quad k = 1, 2, \ldots, N-1 \\ \beta_N^i &= \frac{F_{N-1}^{i-1}}{2N} \\ \beta_0^i &= 2y_0 + 2\sum_{k=1}^{k=N}(-1)^{k+1}\beta_k^i \end{aligned}$$

4.2.4 Collocation and Spectral Methods

The previously mentioned methods are based on low-order Taylor expansions, which limit the step size that can be used to propagate the solution. Additionally, a common weakness of all methods based on low-order Taylor expansion is that they are not effective in enforcing algebraic constraints. Therefore, recent research has looked for other numerical schemes.

4.2.4.1 Collocation methods

One of these numerical schemes is the collocation method [35, 36, 37]. In this method, the solution components are approximated by piecewise polynomials on a mesh. The mesh is made up of a number of points in the domain (called collocation points), and the problem is solved by minimizing the residual of the differential equation at the collocation points. In general, this reduces to computing the unknown coefficients of the polynomial functions. The approximation to the solution must satisfy the constraint conditions and the differential equation at the collocation points in each mesh subinterval. In the collocation methods, the placement of the collocation points is not arbitrary. A modified Newton-type method, known as quasi-linearization, is then used to solve the nonlinear equations for the polynomial coefficients. The mesh is then refined by equally distributing the estimated error over the whole interval, and therefore, an initial estimation of the solution across the mesh is required. In general, this method numerically approximates the differential equation and the specified constraints.

4.2.4.2 Spectral methods

On the other hand, spectral methods [38] model the differential equation's solution by a sum of "basis functions" with unknown coefficients that are solved according to the specific differential equation. The differential equation is then approximated by 1) discretizing the domain and 2) solving the resulting algebraic equations of the differential equation and specified constrained at these nodes. In general, this method benefits from being less computationally expensive than approaches like collocation methods, but it suffers from accuracy problems when applied to complex geometries such as discontinuities. Furthermore, spectral methods are the most similar to the TFC approach since they both are an "assumed" solution method. In both techniques, we assume the form of the solution (i.e., Chebyshev orthogonal polynomials) and solve for unknown coefficients that minimize the residual of the differential equation. The key difference between spectral methods and the TFC method is

in spectral methods, the constraints have to be introduced into the numerical scheme and therefore have associated error, whereas, in the TFC method, the constraints are satisfied analytically via the constrained expression.

4.2.5 Machine Learning

With the current boom in machine learning and artificial intelligence spurred by the increasing capabilities of computers, researchers have looked to apply these algorithms to the numerical solution of differential equations. This method is similar to the spectral method; however, the "basis functions" are replaced with neural networks (NNs) and paired with a multitude of optimization algorithms to solve the problem. In fact, various authors have explored the feasibility of using Neural Networks (NNs) to solve ODEs and PDEs.

The basis of this work leverages two main ideas. First, the *Universal Approximation Theorem* [39, 40], which states that NNs are universal approximators, and therefore, can potentially represent the function that is the solution of a given differential equation [39, 41] as the number of neurons go to infinity. Using these ideas, in 1995, Chen and Chen [42] were able to show that NNs could approximate nonlinear operators. Furthering this work, Pinkus [43] and Lu et al. [44] detailed a function and its partial derivatives that could simultaneously and uniformly be approximated with a single layer NN with a sufficiently large number of hidden neurons.

Of importance to the topic of this dissertation, for ODEs, multiple NN-based solutions have been proposed, including Yang et al. [45] Legendre Neural Networks (LeNNs), Sun et al. [46] Bernstein Neural Network (BNNs), and Mall and Chakraverty [47] Chebyshev Neural Network (CNNs). All of these techniques use single-layer NNs where the activation functions are Legendre, Bernstein, or Chebyshev polynomials, respectively. The network is trained via the Extreme Learning Machine (ELM) algorithm, proposed by Huang et al. [48][1]. The ELM algorithm is used for single-hidden layer feed-forward networks where the

[1]The author notes that the method of Legendre, Bernstein, or Chebyshev Neural Networks paired with the ELM algorithm is exactly the method defined by the spectral method by simply using Legendre, Bernstein, or Chebyshev polynomials.

hidden input weights and biases are randomly selected, and the output weights are solved via least-squares. To satisfy the problem constraint, a constraint penalty is added to the loss function minimized during the training phase.

4.3 The TFC method to solve ODEs

As we will soon see, the TFC method shares a similar approach to the collocation method, spectral method, and ELMs. However, the distinction is that the constraints are embedded analytically before the numerical approximation step. In summary, this will provide us with two unique advantages, 1) the constraints are always satisfied analytically, and 2) the loss functions only deal with the differential equation to be solved. In general, the TFC method is planted between the two general methods (analytical and numerical) to solve differential equations. This can be easily visualized in the diagram of Figure 4.1. In the prior section,

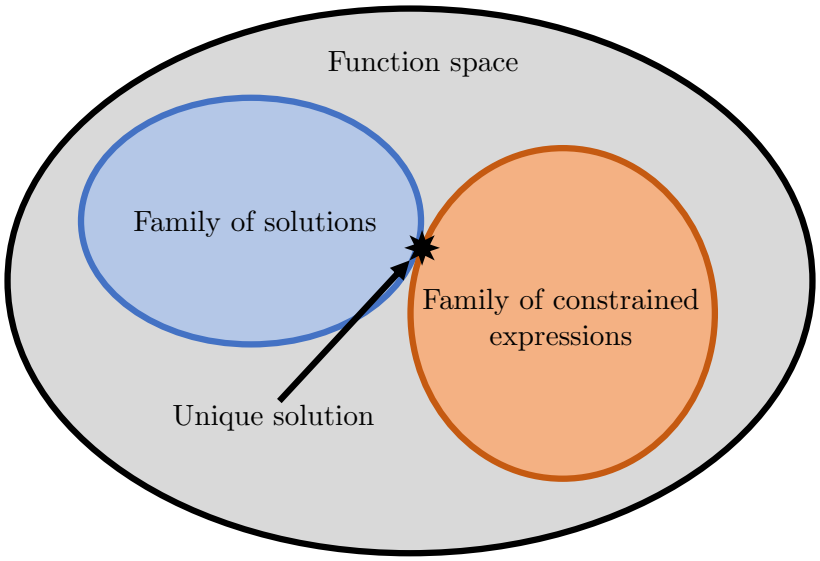


Figure 4.1: Diagram of function space associated with the solution of a ordinary differential equation. Note: this figure is used for conceptual purposes and is not a rigorous mathematical description. For example, in the solution of some differential equations, there could be more than one, or even infinite intersection points depending on the nature of the differential equation.

we discussed the solution of DEs through analytical techniques. The analytical method is

represented by the blue oval, where a family of solutions is provided. The unique solution (the black star) is then determined by applying the constraints to the differential equation. On the other hand, numerical solutions (excluding IVPs) must search the entire function space to find a unique solution. Conversely, the TFC method solves the problem in the opposite sequence of the analytical approach. First, the candidate solution is constructed by using a constrained expression. The constrained expression represents a reduction of the function space to a set only the functions satisfying the DE's constraints. Then, the codomain of the constrained expression is used to find the unique solution of the differential equation. In another sense, if we assume that our free function, $g(x)$, covers the function space of the solution, then the constrained expression is projecting this function into a reduced set of the constraints, i.e., the orange oval. It should be clear from this discussion that the solution of the differential equation is dependent on the definition of $g(x)$.

To further understand these concepts, let us consider a general differential equation,

$$F\Big(x, y, \frac{\mathrm{d}y}{\mathrm{d}x}, \frac{\mathrm{d}^2y}{\mathrm{d}x^2}, \cdots, \frac{\mathrm{d}^ny}{\mathrm{d}x^n}\Big) = 0, \tag{4.1}$$

subject to n linear constraints. Using the TFC framework, the first step is to derive the switching and projection functions of Equation (2.7). By doing this, the constraints of Equation (4.1) are decoupled from the solution of the differential equation, and the differential equation is transformed into,

$$\tilde{F}\Big(x, g, \frac{\mathrm{d}g}{\mathrm{d}x}, \frac{\mathrm{d}^2g}{\mathrm{d}x^2}, \cdots, \frac{\mathrm{d}^ng}{\mathrm{d}x^n}\Big) = 0, \tag{4.2}$$

where the solution to this "differential equation"[2] is obtained by finding the function $g(x)$ satisfying Equation (4.2). In order to solve this new equation, four major steps must be taken: 1) define the free function $g(x)$ and 2) determine the derivatives of the free function

[2]The use of quotations around the word differential equation is used because the resulting expression is: 1) technically not a differential equation and 2) cannot be solved using the analytical techniques to solve differential equations. To date, this type of equation has not been rigorously defined.

$g(x)$ 3) discretize the domain, and 4) solve the resulting algebraic equation. The following sections elaborate on these steps.

4.3.1 Defining the free function

For our definition of the free function, we will allow the domain of this function, z, to be different from the differential equation problem domain x. Ultimately, we will need to map between the domains with some function $z = z(x)$; however, allowing for different basis and problem domains is necessary in most cases since some numerical bases are defined on closed domains, e.g., Chebyshev orthogonal polynomials are defined on $z \in [-1, +1]$. This will be made clear in Section 4.3.2.

In selecting a free function, we are essentially looking for the best (differentiable) function approximator. A simple definition of $g(x)$ could be the monomial expansion of m terms,

$$g(x) = \sum_{k=0}^{m-1} a_k \, z^k, \tag{4.3}$$

where a_k are coefficients and z is simply the independent variable. According to Claim 3, the terms linearly dependent to the support functions used in the constrained expression must be removed. While this definition is valid, a linear combination of orthogonal polynomials can be leveraged for their advantageous numerical properties.

Consider the definition of Chebyshev polynomials of the first kind,

$$g(x) = \sum_{k=0}^{m-1} a_k \, T_k(z), \tag{4.4}$$

where again a_k are coefficients and $T_k(z)$ are the Chebyshev polynomials terms. Again, Claim 3 must be considered in this expansion. It has been shown that Chebyshev polynomials of the first kind produce a function that minimizes the maximum error in its application. In fact, these polynomials are part of a special class well suited for function approximation [49]. Furthermore, this expansion also provides a simple way to estimate the solution's accuracy

by observing the size of the coefficients of latter terms in the expansion (i.e., the coefficients of the highest-order terms), which is justified by the convergence properties of Chebyshev polynomials. An even better approximation is obtained by comparing the sets of coefficients obtained when the number of basis terms is varied [50].

Additionally, the Legendre orthogonal polynomials, defined as,

$$g(x) = \sum_{k=0}^{m-1} a_k \, L_k(z), \tag{4.5}$$

where a_k are coefficients and L_k are polynomial terms, are another important expansion, which has been used extensively in function approximation and the solution of differential equations with beneficial error properties for the approximation of smooth functions [51]. In fact, both orthogonal polynomials types mentioned have been extensively used in spectral methods [38].

Moreover, our definition of $g(x)$ can even extend to machine learning where the function is defined as a neural network where we would express

$$g(x) = \mathcal{N}(z, \theta),$$

where the architecture is based on the independent variable z and trainable parameters θ, such as the weights and the biases. A complete study of the use of neural networks is out of the scope of this work, and interested readers are directed to Leake and Mortari [52] for a more detailed look into applying TFC in this field.

In addition to the general neural networks, one specific architecture has shown promising results which is based on the theory of the ELM [48]. ELMs are a single-layer feed-forward NN where in the univariate definition,

$$g(x) = \sum_{k=0}^{m-1} a_k \, \sigma \left(w_k z + b_k \right). \tag{4.6}$$

In this equation, m is the number of hidden neurons, i.e., similar to the number of basis functions, and σ is a user-defined activation function, e.g., sigmoid, tanh, swish, etc. The terms w_k and b_k are the associated weights and biases for the nodes and are selected randomly according to any continuous probability distribution proven in Theorems 2.1 and 2.2 in G.-B. Huang et al. [48]. Therefore, it makes the unknown coefficients, a_k, linear in the form of Equation (4.6) similar to Equations (4.3), (4.4), and (4.5).

Moving forward we will only consider the free function defined in terms of the Chebyshev polynomials Equation (4.4), Legendre polynomials Equation (4.5), and ELMs, Equation (4.6). Since all functions are linear in their unknown coefficients, a_k, let us write the general expansion as,

General Basis Expansion

$$g(x) = \boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{h}(z) \quad \text{where} \quad z = z(x) \tag{4.7}$$

where $\boldsymbol{\xi} = \{a_0, \cdots, a_k, \cdots, a_{m-1}\}^{\mathrm{T}}$ and $\boldsymbol{h}(z)$ is a vector function of the m functions.

4.3.2 Derivatives of the free function

In most cases the domain of the free function will not coincide with the domain of the problem. For example, for the orthogonal polynomials mentioned, the domain is defined for $z \in [-1, +1]$ and most of the time it is desirable to scale the input which may be different than our problem domain, $x \in [x_0, x_f]$. Therefore, these functions must be linearly mapped to the independent variable x. This can be done using the equations,

$$z = z_0 + \frac{z_f - z_0}{x_f - x_0}(x - x_0) \quad \longleftrightarrow \quad x = x_0 + \frac{x_f - x_0}{z_f - z_0}(z - z_0), \tag{4.8}$$

where x_f represents the upper integration limit. The subsequent derivatives of the free function are defined as,

$$\frac{\mathrm{d}^n g}{\mathrm{d}x^n} = \boldsymbol{\xi}^{\mathrm{T}} \frac{\mathrm{d}^n \boldsymbol{h}(z)}{\mathrm{d}z^n} \left(\frac{\mathrm{d}z}{\mathrm{d}x} \right)^n,$$

where by defining,

$$c := \frac{\mathrm{d}z}{\mathrm{d}x} = \frac{z_f - z_0}{x_f - x_0} \tag{4.9}$$

the expression can be simplified to,

Derivatives of the free function

$$\frac{\mathrm{d}^n g}{\mathrm{d}x^n} = c^n \boldsymbol{\xi}^{\mathrm{T}} \frac{\mathrm{d}^n \boldsymbol{h}(z)}{\mathrm{d}z^n},$$

which defines all mappings of the free function. By defining the free function according to the form of Equation (4.7), our transformed differential equation, Equation (4.2), that was derived earlier reduces to,

$$\tilde{F}(x, \boldsymbol{\xi}) = 0. \tag{4.10}$$

Next, the problem domain, x, must be discretized to eventually solve for the unknown coefficients and ultimately solve the differential equation. Therefore, a specific discretized scheme is needed.

4.3.3 Discretization of the domain

Since the ultimate goal is to solve Equation (4.1) computationally, the problem domain (and therefore the basis function domain) must be discretized. In the case of defining $g(x)$ using an ELM, the discretization can simply be selected as uniformly spaced points. However, when using Chebyshev and Legendre orthogonal polynomials, the discretization scheme is slightly more involved. For these polynomials, the optimal discretization scheme is Chebyshev-Gauss-Lobatto nodes [53, 54]. For $N+1$ points, the discrete points are calculated as,

Discretization scheme for Chebyshev-Gauss-Lobatto nodes

$$z_j = -\cos\left(\frac{j\pi}{N}\right) \quad \text{for} \quad j = 0, 1, 2, \cdots, N. \tag{4.11}$$

Compared with the uniform distribution, this distribution results in a much slower increase of the condition number of the matrix to be inverted in the least-squares as the number of basis functions, m, increases. The nodes can be realized in the problem domain through the relationship provided in Equation (4.9).

By discretizing the domain according to the specific free function used, Equation (4.10) becomes a system of equations that is linear if Equation (4.1) is linear and nonlinear if Equation (4.1) is nonlinear. This can be written as a loss vector at the discretized points,

$$\mathbb{L}(\boldsymbol{\xi}) = \begin{Bmatrix} \tilde{F}(x_0, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_k, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_f, \boldsymbol{\xi}) \end{Bmatrix} = \mathbf{0} \tag{4.12}$$

where x_k, and therefore z_k, are defined by Equation (4.8) and Equation (4.11).

4.3.4 Solving the resulting algebraic equation

For a linear differential equation F (and therefore a linear differential equation $\tilde{F}$), the constrained expression and its derivatives will show up linearly, and therefore, will remain linear in the unknown $\boldsymbol{\xi}$ term. This leads to the form,

$$\mathbb{A}\boldsymbol{\xi} + \boldsymbol{b} = 0, \tag{4.13}$$

where the matrix $\mathbb{A}$ is composed of a linear combination of the terms linear in the unknown coefficients. Written in terms of the loss function $\tilde{F}$, $\mathbb{A}$ is simply the Jacobian of the loss

vector Equation (4.12),

$$\mathbb{J}(\boldsymbol{\xi}) = \frac{\partial \mathbb{L}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} \dfrac{\partial \tilde{F}(x_0, \boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \\ \vdots \\ \dfrac{\partial \tilde{F}(x_k, \boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \\ \vdots \\ \dfrac{\partial \tilde{F}(x_N, \boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \end{bmatrix}.$$

Since the loss function is linear in $\boldsymbol{\xi}$, it will be independent of $\boldsymbol{\xi}$. Additionally, the vector $\boldsymbol{b}$ is simply the loss vector evaluated at $\boldsymbol{\xi} = \mathbf{0}$,

$$\boldsymbol{b} = \mathbb{L}(\mathbf{0}) = \begin{Bmatrix} \tilde{F}(x_0, \mathbf{0}) \\ \vdots \\ \tilde{F}(x_k, \mathbf{0}) \\ \vdots \\ \tilde{F}(x_f, \mathbf{0}) \end{Bmatrix}.$$

Therefore, Equation (4.13) can also be realized as,

$$\mathbb{J}(\mathbf{0})\, \boldsymbol{\xi} = -\mathbb{L}(\mathbf{0}). \tag{4.14}$$

In these linear cases, Equation (4.14) can be solved directly using any available least-squares technique. A summary of these numerical schemes are provided in Appendix B. However, in the case of a nonlinear differential equations, Equation (4.12) will be nonlinear in the $\boldsymbol{\xi}$ coefficients. This system can be solved by an iterative least-squares method similar to Equation (4.14); however, now a multivariate Newton's method is used to solve the nonlinear system for the change in the $\boldsymbol{\xi}$ parameter denoted by $\Delta\boldsymbol{\xi}$,

$$\mathbb{J}(\boldsymbol{\xi}_i)\, \Delta\boldsymbol{\xi}_i = -\mathbb{L}(\boldsymbol{\xi}_i) \tag{4.15}$$

Parameter update equations

The parameter update of $\boldsymbol{\xi}$ is provided by,

$$\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi}_i + \Delta\boldsymbol{\xi}_i$$

where the $\Delta\boldsymbol{\xi}_i$ can be defined using classic least-squares,

$$\Delta\boldsymbol{\xi}_i = -\Big(\mathbb{J}(\boldsymbol{\xi}_i)^{\mathrm{T}}\mathbb{J}(\boldsymbol{\xi}_i)\Big)^{-1}\mathbb{J}(\boldsymbol{\xi}_i)^{\mathrm{T}}\mathbb{L}(\boldsymbol{\xi}_i),$$

or any other least-squares technique provided in Appendix B. This process is repeated until some stopping criteria are met. The original work on the solution of nonlinear differential equations by Mortari, Johnston, and Smith [18] used the L_2 norm of the loss function and the L_2 norm of the least-squares step ($\Delta\boldsymbol{\xi}$) such that it was below some tolerance, ε, according to the following equations,

$$L_2[\mathbb{L}(\boldsymbol{\xi}_i)] < \varepsilon \qquad \text{or} \qquad L_2[\Delta\boldsymbol{\xi}_i] < \varepsilon.$$

However, the work presented in this dissertation utilizes a slightly different stopping condition to reduces computational overhead such that,

$$\max[\mathbb{L}(\boldsymbol{\xi}_i)] < \varepsilon \qquad \text{or} \qquad \max[\Delta\boldsymbol{\xi}_i] < \varepsilon.$$

In all, the solution of a linear versus a nonlinear ordinary differential equation is reduced simply to the difference between Equation (4.14) and Equation (4.15), where the linear case only requires "one" iteration compared to the nonlinear equations. This similarity is highlighted in Section 4.5.1, where the problem is formulated according to both notations.

Additionally, since the constraints are embedded in the constrained expression before

forming the loss vector, the numerical scheme does not change between boundary conditions. In other words, an initial-value problem is solved in the same manner as a boundary-value problem. We will soon see the power of this when applying TFC to the solution of boundary-value problems.

4.3.5 The TFC roadmap

Before moving to our numerical examples, it is useful to summarize the entire process of solving differential equations using the TFC approach. This is provided in the flowchart in Figure 4.2, where the process is summarized with all major equations. First, given the dif-

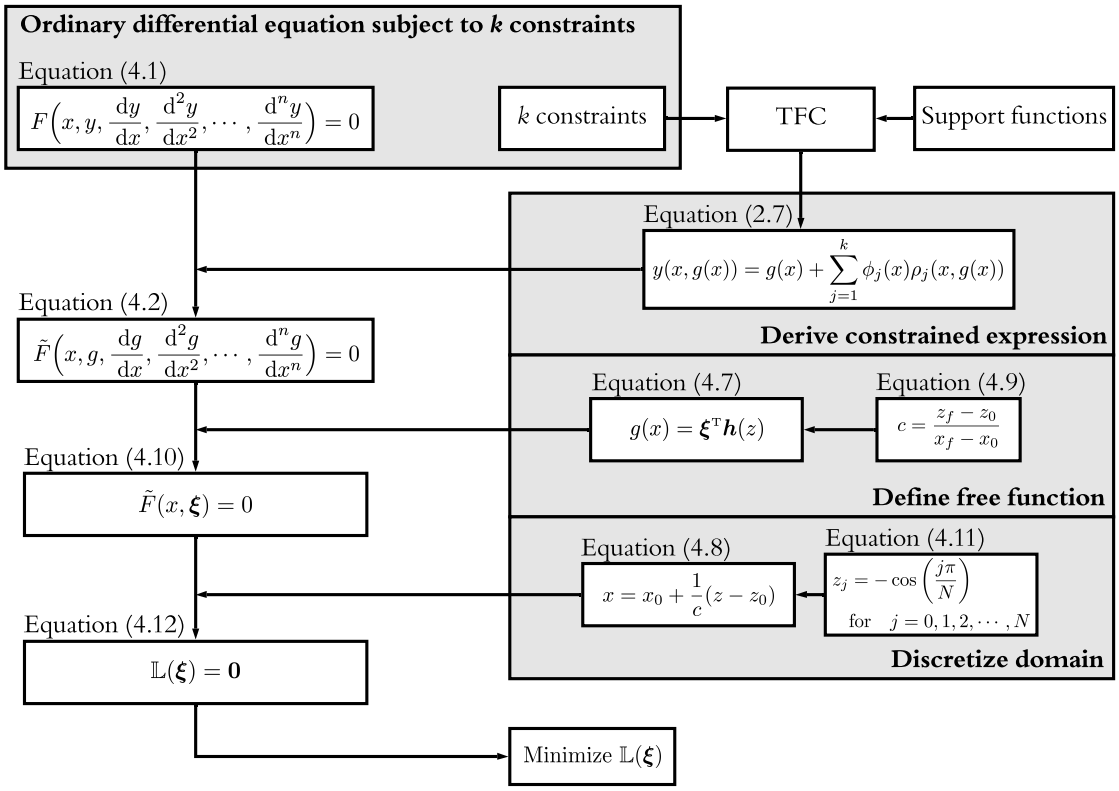


Figure 4.2: Flowchart of the TFC method applied to solving an ordinary differential equation in the form of Equation (4.1).

ferential equation, Equation (4.1), subject to k constraints, we embed these constraints into the constrained expression, Equation (2.7), by selecting acceptable support functions and

deriving the projection functionals and switching functions. The constrained expression and its derivative are substituted into Equation (4.1), which transforms the differential equation subject to k constraints to one which is unconstrained and denoted by $\tilde{F}(x, \boldsymbol{\xi})$, Equation (4.2). After this, the free function $g(x)$ is expressed by one of the many function approximation methods discussed in Section 4.3.1 using Equation (4.7). By doing this, the differential equation is transformed into an algebraic equation with the unknown vector $\boldsymbol{\xi}$. Next, we discretize the basis function domain according to Equation (4.11) when using Chebyshev or Legendre polynomials, and uniformly when using ELMs, and connect these to the problem domain by Equation (4.8). By evaluating Equation (4.10), the loss function, at these discretization nodes and stacking them in a loss vector we are led to Equation (4.12). Finally, Equation (4.12) is minimized using least-squares or nonlinear least-squares, depending on the linearity of the original differential equation, Equation (4.1). Note, we are not limited to least-squares techniques, and in fact, any numerical minimization scheme can be used to solve the system $\mathbb{L}(\boldsymbol{\xi}) = \mathbf{0}$. With that said, the work in this dissertation focuses specifically on least-squares techniques for numerical simplicity and speed advantages. However, with the increasing complexity of problems, least-squares can become prohibitive, and the use of different optimizers is an area of future research summarized in Section 8.1.2.

4.4 Numerical Implementation

To demonstrate how the TFC approach is used to solve differential equations, we will start with two simple examples covering a linear initial-value problem (Section 4.5) and a nonlinear boundary-value problem (Section 4.6). These problems provide the full derivation and explicitly provide the Jacobian of the loss vector directly in the text for clarity. After these problems, all analytical Jacobians are not provided directly in the main text but collected in Appendix D. Following this, a brief discussion is provided on how systems of differential equations (or a subclass, vector equations) can be solved in the same manner. Lastly, Section 4.8 discusses two adjustments to the theory to solve problems with discontinuous dynamics and unknown final times. Additionally, all numerical results were produced

on a MacBook Pro (2016) macOS Version 10.15, with a 3.3 GHz Dual-Core Intel® Core™ i7 and with 16 GB of RAM.

4.5 Lane-Emden equation

As a motivating example, let us consider the Lane-Emden equation where,

$$y_{xx} + \frac{2}{x} y_x + y^a = 0 \quad \text{such that} \quad (x > 0, a \geq 0) \quad \text{subject to:} \quad \begin{cases} y(0) = 1 \\ y_x(0) = 0 \end{cases} \tag{4.16}$$

For this differential equation, an exact solution exists for $a = 0$, 1, and 5. We can see, regardless of the value of a, the constrained expression will be the same. Therefore, whether the equation is linear or nonlinear does not affect the derivation of the constrained expression. This should be obvious since the TFC approach decouples the problem's constraints from the solution of the differential equation. Using the theory developed earlier, the constrained expression for this problem can be solved by defining the projection functionals as,

$$\rho_1(x, g(x)) = 1 - g(0) \qquad \text{and} \qquad \rho_2(x, g(x)) = -g_x(0)$$

and the switching functions are determined by choosing the support functions $s_1(x) = 1$ and $s_2(x) = x$ and solving for the coefficients α_{ij},

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where it can easily be seen that $\alpha_{ij} = \delta_{ij}$. Thus, the switching functions are $\phi_1(x) = 1$ and $\phi_2(x) = x$, and the final constrained expression is,

$$y(x, g(x)) = g(x) + (1 - g(0)) + x(-g_x(0)). \tag{4.17}$$

The simplicity of this expression is due in part to the second-order initial value constraints. See Appendix C for a summary of the associated switching functions and projection functionals for other typical constraint cases. The constrained expression, Equation (4.17), always satisfies the constraints of Equation (4.16).

Now, by defining $g(x)$ according to Equation (4.7), the constrained expression and its derivatives can be written as a linear function of the unknown coefficients,

$$y(x, \boldsymbol{\xi}) = \Big(\boldsymbol{h} - \boldsymbol{h}(z_0) - x\, c\boldsymbol{h}_z(z_0)\Big)^{\mathrm{T}} \boldsymbol{\xi} + 1 \tag{4.18}$$

$$y_x(x, \boldsymbol{\xi}) = \Big(c\boldsymbol{h}_z - c\boldsymbol{h}_z(z_0)\Big)^{\mathrm{T}} \boldsymbol{\xi} \tag{4.19}$$

$$y_{xx}(x, \boldsymbol{\xi}) = \Big(c^2\boldsymbol{h}_{zz}\Big)^{\mathrm{T}} \boldsymbol{\xi} \tag{4.20}$$

In the following sections, we will use our description of the constrained expression to solve each case of the Lane-Emden equation.

4.5.1 Linear differential equations

First, let us consider the solution of the linear differential equation associated with setting $a = 0$ in the Lane-Emden equation,

$$y_{xx} + \frac{2}{x} y_x + 1 = 0 \quad \text{subject to:} \begin{cases} y(0) = 1 \\ \\ y_x(0) = 0 \end{cases} .$$

This equation is singular at the initial value of $x = 0$ due to the coefficient function $\frac{2}{x}$. However, we can avoid this by multiplying both sides of the equation by the variable x. Hence, the differential equation becomes,

$$x\, y_{xx} + 2\, y_x + x = 0,$$

which when evaluated at $x = 0$ gives us the initial derivative constraint. By substituting the constrained expression into the differential equation, we are left with an algebraic equation with unknowns $\boldsymbol{\xi}$,

$$\Big[x\, c^2 \boldsymbol{h}_{zz} + 2\left(c\boldsymbol{h}_z - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} = -x, \tag{4.21}$$

where the coefficient c comes from our mapping between the basis function domain and problem domain (recall Equation (4.9)). Now, by discretizing the domains, Equation (4.21) can be written as a linear system of equation such that,

$$\begin{bmatrix} \Big[x_0\, c^2 \boldsymbol{h}_{zz}(z_0) + 2\left(c\boldsymbol{h}_z(z_0) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_k\, c^2 \boldsymbol{h}_{zz}(z_k) + 2\left(c\boldsymbol{h}_z(z_k) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_f\, c^2 \boldsymbol{h}_{zz}(z_f) + 2\left(c\boldsymbol{h}_z(z_f) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \end{bmatrix} \boldsymbol{\xi} = \begin{Bmatrix} -x_0 \\ \vdots \\ -x_k \\ \vdots \\ -x_f \end{Bmatrix}$$

which is of the form $A\boldsymbol{x} = \boldsymbol{b}$ and can be solved with any least-squares technique. While the construction of this linear system was straightforward, there is another formalization that will be consistent between linear and nonlinear differential equations. To realize this, consider rewriting the differential equation as the loss function,

$$\tilde{F} = \Big[x\, c^2 \boldsymbol{h}_{zz} + 2\left(c\boldsymbol{h}_z - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} + x = 0$$

which can be written as a loss vector which is the discretization of $\tilde{F}$ at the collocation nodes,

$$\mathbb{L}(\boldsymbol{\xi}) = \begin{Bmatrix} \tilde{F}(x_0, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_k, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_f, \boldsymbol{\xi}) \end{Bmatrix} = \begin{Bmatrix} \Big[x_0\, c^2 \boldsymbol{h}_{zz}(z_0) + 2\left(c\boldsymbol{h}_z(z_0) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} + x_0 \\ \vdots \\ \Big[x_k\, c^2 \boldsymbol{h}_{zz}(z_k) + 2\left(c\boldsymbol{h}_z(z_k) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} + x_k \\ \vdots \\ \Big[x_f\, c^2 \boldsymbol{h}_{zz}(z_f) + 2\left(c\boldsymbol{h}_z(z_f) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} + x \end{Bmatrix} = \mathbf{0}$$

with the Jacobian term of,

$$\mathbb{J}(\boldsymbol{\xi}) = \begin{bmatrix} \Big[x_0\, c^2 \boldsymbol{h}_{zz}(z_0) + 2\left(c\boldsymbol{h}_z(z_0) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_k\, c^2 \boldsymbol{h}_{zz}(z_k) + 2\left(c\boldsymbol{h}_z(z_k) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_f\, c^2 \boldsymbol{h}_{zz}(z_f) + 2\left(c\boldsymbol{h}_z(z_f) - c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \end{bmatrix}$$

where the equation,

$$\mathbb{J}(\boldsymbol{\xi} = \mathbf{0})\boldsymbol{\xi} = -\mathbb{L}(\boldsymbol{\xi} = \mathbf{0}) \quad \Longleftrightarrow \quad A\boldsymbol{x} = \boldsymbol{b};$$

however, this is the same as the first iteration of the nonlinear least-squares approach. Therefore, writing all problems (linear or nonlinear) using the loss function and Jacobian allows us to use the same process and simplify notation.

Next, for the Lane-Emden equation where $a = 1$, the loss function becomes

$$\tilde{F} = \Big[x\, c^2 \boldsymbol{h}_{zz} + 2\left(c\boldsymbol{h}_z - c\boldsymbol{h}_z(z_0)\right) + x\left(\boldsymbol{h} - \boldsymbol{h}(z_0) - x\, c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} = 0$$

making the loss vector,

$$\mathbb{L}(\boldsymbol{\xi}) = \begin{Bmatrix} \tilde{F}(x_0, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_k, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_f, \boldsymbol{\xi}) \end{Bmatrix}$$

$$= \begin{Bmatrix} \Big[x_0\, c^2 \boldsymbol{h}_{zz}(z_0) + 2\left(c\boldsymbol{h}_z(z_0) - c\boldsymbol{h}_z(z_0)\right) + x_0\left(\boldsymbol{h}(z_0) - \boldsymbol{h}(z_0) - x_0\, c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} \\ \vdots \\ \Big[x_k\, c^2 \boldsymbol{h}_{zz}(z_k) + 2\left(c\boldsymbol{h}_z(z_k) - c\boldsymbol{h}_z(z_0)\right) + x_k\left(\boldsymbol{h}(z_k) - \boldsymbol{h}(z_0) - x_k\, c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} \\ \vdots \\ \Big[x_f\, c^2 \boldsymbol{h}_{zz}(z_f) + 2\left(c\boldsymbol{h}_z(z_f) - c\boldsymbol{h}_z(z_0)\right) + x_f\left(\boldsymbol{h}(z_f) - \boldsymbol{h}(z_0) - x_f\, c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \boldsymbol{\xi} \end{Bmatrix}$$

with Jacobian,

$$\mathbb{J}(\boldsymbol{\xi}) = \begin{bmatrix} \Big[x_0\, c^2 \boldsymbol{h}_{zz}(z_0) + 2\left(c\boldsymbol{h}_z(z_0) - c\boldsymbol{h}_z(z_0)\right) + x_0\left(\boldsymbol{h}(z_0) - \boldsymbol{h}(z_0) - x_0\, c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_k\, c^2 \boldsymbol{h}_{zz}(z_k) + 2\left(c\boldsymbol{h}_z(z_k) - c\boldsymbol{h}_z(z_0)\right) + x_k\left(\boldsymbol{h}(z_k) - \boldsymbol{h}(z_0) - x_k\, c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_f\, c^2 \boldsymbol{h}_{zz}(z_f) + 2\left(c\boldsymbol{h}_z(z_f) - c\boldsymbol{h}_z(z_0)\right) + x_f\left(\boldsymbol{h}(z_f) - \boldsymbol{h}(z_0) - x_f\, c\boldsymbol{h}_z(z_0)\right)\Big]^{\mathrm{T}} \end{bmatrix}.$$

.

4.5.2 Nonlinear ordinary differential equations

Now, let us consider the nonlinear cases of the Lane-Emden equation where,

$$y_{xx} + \frac{2}{x} y_x + y^a = 0 \quad \text{such that} \quad (x > 0, a \geq 2) \quad \text{subject to:} \quad \begin{cases} y(0) = 1 \\ y_x(0) = 0 \end{cases}$$

Again, since the constraints are the same as the linear instance of the differential equation, the constrained expression is the same as in Equation (4.18). Now, the approach is exactly the same as the linear cases. First, we form the loss vector such that,

$$\tilde{F} = x\, y_{xx} + 2y_x + x\, y^a = 0$$

where for clarity the terms y, y_x, and y_{xx} are not expanded. These equations are defined by Equation (4.18), Equation (4.19), and Equation (4.20), respectively. This produces the loss vector,

$$\mathbb{L}(\boldsymbol{\xi}) = \begin{Bmatrix} \tilde{F}(x_0, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_k, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_f, \boldsymbol{\xi}) \end{Bmatrix} = \begin{Bmatrix} x_0\, y_{xx}(x_0, \boldsymbol{\xi}) + 2y_x(x_0, \boldsymbol{\xi}) + x_0\, y^a(x_0, \boldsymbol{\xi}) \\ \vdots \\ x_k\, y_{xx}(x_k, \boldsymbol{\xi}) + 2y_x(x_k, \boldsymbol{\xi}) + x_k\, y^a(x_k, \boldsymbol{\xi}) \\ \vdots \\ x_f\, y_{xx}(x_f, \boldsymbol{\xi}) + 2y_x(x_f, \boldsymbol{\xi}) + x_f\, y^a(x_f, \boldsymbol{\xi}) \end{Bmatrix}$$

Additionally, it follows that the Jacobian is,

$$\mathbb{J}(\boldsymbol{\xi}) = \begin{bmatrix} \Big[x_0\, c^2 \boldsymbol{h}_{zz}(z_0) + 2\Big(c\boldsymbol{h}_z(z_0) - c\boldsymbol{h}_z(z_0)\Big) + x_0\, a\, y^{a-1}(x_0, \boldsymbol{\xi})\Big(\boldsymbol{h}(z_0) - \boldsymbol{h}(z_0) - x_0\, c\boldsymbol{h}_z(z_0)\Big)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_k\, c^2 \boldsymbol{h}_{zz}(z_k) + 2\Big(c\boldsymbol{h}_z(z_k) - c\boldsymbol{h}_z(z_0)\Big) + x_k\, a\, y^{a-1}(x_k, \boldsymbol{\xi})\Big(\boldsymbol{h}(z_k) - \boldsymbol{h}(z_0) - x_k\, c\boldsymbol{h}_z(z_0)\Big)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[x_f\, c^2 \boldsymbol{h}_{zz}(z_f) + 2\Big(c\boldsymbol{h}_z(z_f) - c\boldsymbol{h}_z(z_0)\Big) + x_f\, a\, y^{a-1}(x_f, \boldsymbol{\xi})\Big(\boldsymbol{h}(z_f) - \boldsymbol{h}(z_0) - x_f\, c\boldsymbol{h}_z(z_0)\Big)\Big]^{\mathrm{T}} \end{bmatrix}$$

.

Again, following the same process, the nonlinear least-squares method is used to update the $\boldsymbol{\xi}$ coefficient vector and ultimately solve the differential equation. In the proceeding section, we look at the accuracy obtained for this problem.

4.5.3 Numerical results of the Lane-Emden equation

The Lane-Emden equation has an analytical solution for the following values of a,

$$a = \begin{cases} 0 & \longrightarrow \quad y = 1 - \dfrac{x^2}{6} \\ 1 & \longrightarrow \quad y = \dfrac{\sin(x)}{x} \\ 5 & \longrightarrow \quad y = \dfrac{1}{\sqrt{1 + \dfrac{x^2}{3}}} \end{cases}$$

In the following examples, we will solve this differential equation for these values of a to directly compare with the analytical solution. This will allow us to analyze the accuracy of the TFC method compared to others in the literature.

Example 4.1: Lane-Emden ($a = 0$)

In this example, the Lane-Emden equation is solved for $a = 0$ on the domain $x \in [0, 10]$. The results given in Figure 4.3 detail the TFC method's accuracy compared to the spectral method using either Chebyshev polynomials or ELMs with the sigmoid function. Figure 4.4 compares the TFC method to spectral both expressed using Chebyshev polynomials to directly quantify the maximum accuracy of these two methods. Finally, Figure 4.5 provides a speed versus accuracy comparison of the techniques mentioned above along with the RK45 technique using SciPy's `scipy.integrate.solve_ivp` algorithm [25].

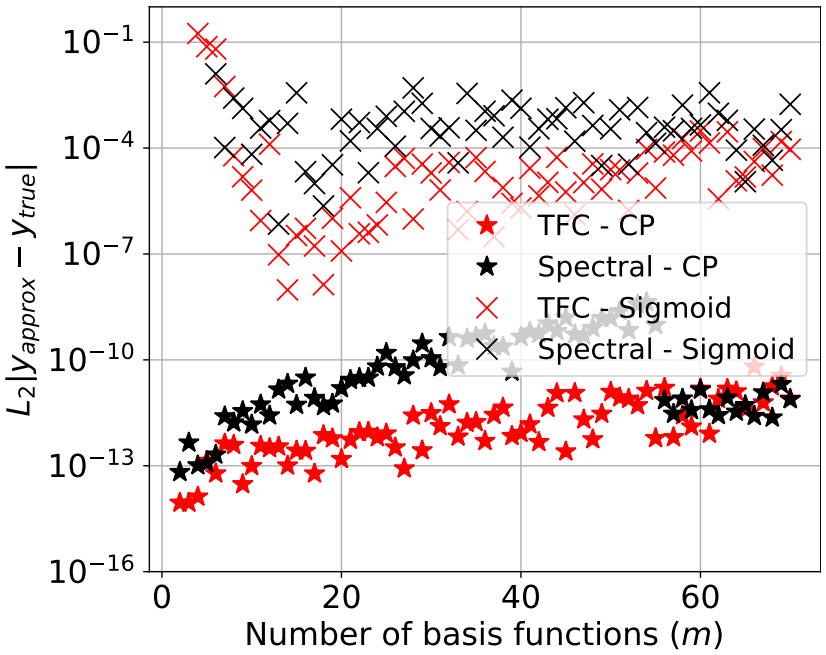


Figure 4.3: Accuracy of TFC and spectral method for varying number and types of basis functions for the Lane-Emdem equation ($a = 0$).

Looking at the TFC based solutions given in Figure 4.3, it can be seen that the orthogonal polynomial definition of the free function provides dramatic accuracy gain at a lower number of terms. Furthermore, even by adding basis terms, the ELM based free function (sigmoid) does no match the accuracy of the Chebyshev orthogonal polynomials. In fact, for the solution of ordinary differential equations, ELMs are never more accurate than the orthogonal basis set.

Looking at the comparison of the TFC method with the spectral method given in Figure 4.4, we can see a slight accuracy gain when using TFC versus a spectral method that increases as the number of basis functions increases. However, this gain of accuracy at higher basis functions is misleading because overall, both methods lose accuracy with this increase. This can be explained by looking at the analytical solution for $a = 0$, which is a quadratic polynomial. This means that an expression (either

with the spectral method or TFC) based on the orthogonal polynomials should have the best solution at $m = 2$. Any terms past this only contribute noise to the solution of the differential equations. Regardless, at $m = 2$, the TFC method is about an order of magnitude more accurate than the spectral method.

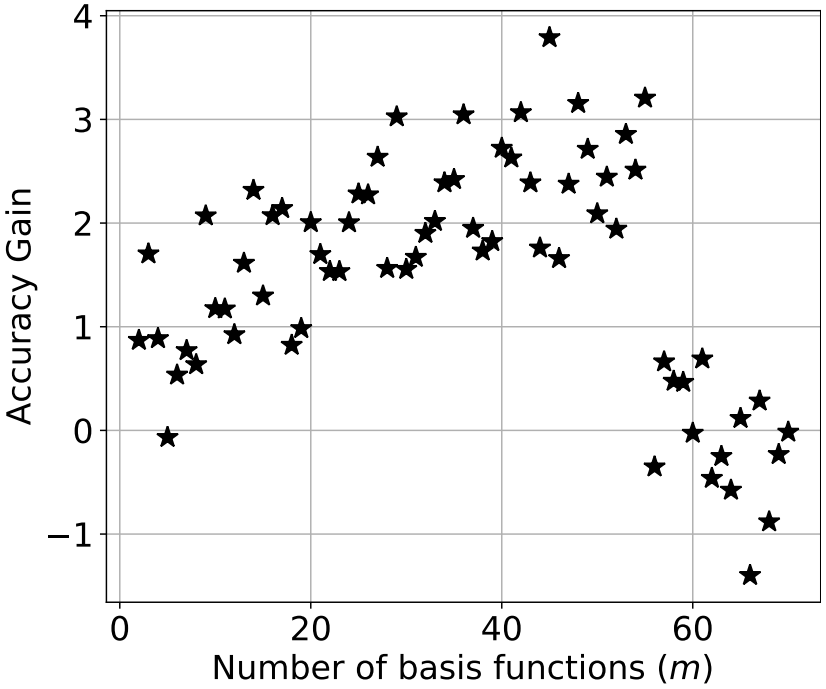


Figure 4.4: Accuracy gain of TFC vs. spectral for the Solution of Lane-Emdem ($a = 0$). The accuracy gain is quantified in terms of $\log_{10}(\frac{\text{spectral method error}}{\text{TFC error}})$ and therefore, the y-axis is by orders of magnitude. For example, when this value is greater than zero, TFC is more accurate, and vice-versa.

Figure 4.5 shows that when more solution accuracy is needed, the RK45 method requires more time to solve the problem, while the spectral and TFC method see little change in computation time. However, comparing spectral and TFC method, there seems to be little difference in accuracy versus speed, with TFC maintaining only a slight advantage.

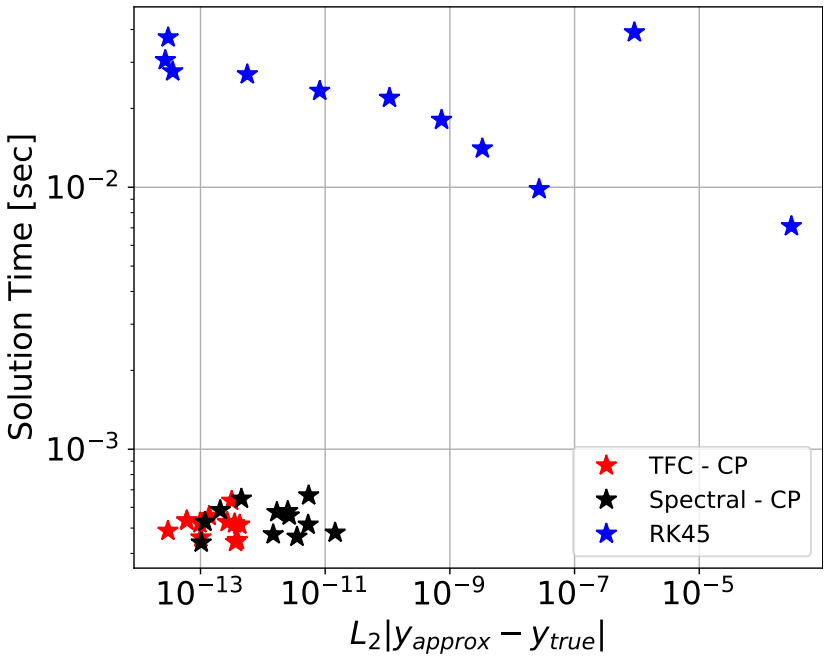


Figure 4.5: Timed solution of Lane-Emdem ($a = 0$).

Example 4.2: Lane-Emden ($a = 1$)

In this example, the Lane-Emden equation is solved for $a = 1$ on the domain $x \in [0, 10]$. The results given in Figures 4.6-4.7 compare the TFC method to both spectral method and ELMs based on the number of basis terms used. Additionally, Figure 4.8 provides a speed versus accuracy comparison of the techniques mentioned above along with the RK45 technique using SciPy's `scipy.integrate.solve_ivp` algorithm [25].

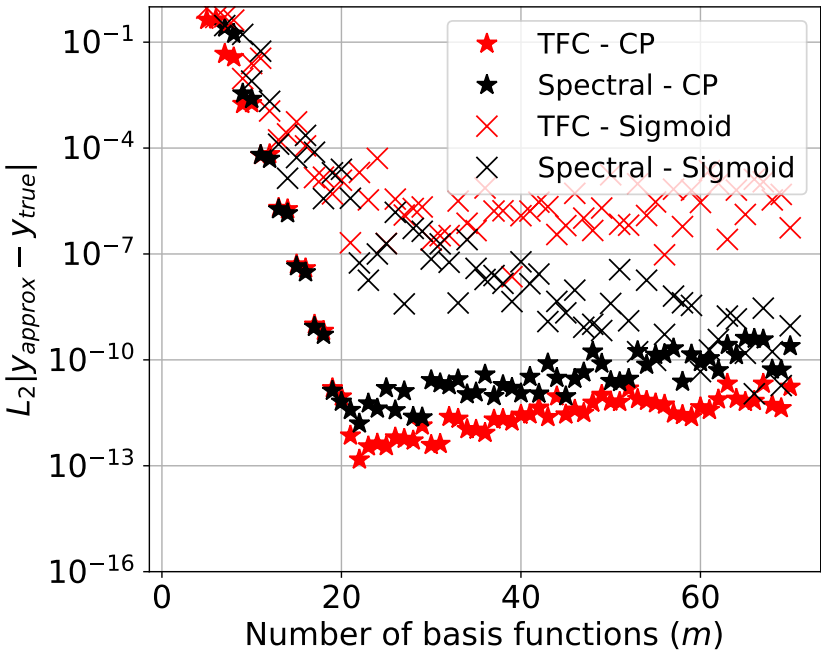


Figure 4.6: Accuracy of TFC and spectral method for varying number and types of basis functions for the Lane-Emdem equation ($a = 1$).

Looking at the TFC based solutions given in Figure 4.6, it can be seen that the orthogonal polynomial definition of the free function quickly reaches a minimum at 22 basis terms. Furthermore, even by adding basis terms, the ELM-based free functions (sigmoid) do not match the Chebyshev orthogonal polynomials' accuracy.

Looking at the comparison of the TFC method with the spectral method in Figure 4.7, we can see that the TFC method is always more accurate than the spectral method when more than 20 basis terms are used. However, at lower basis terms, TFC and the spectral method are comparable in terms of accuracy.

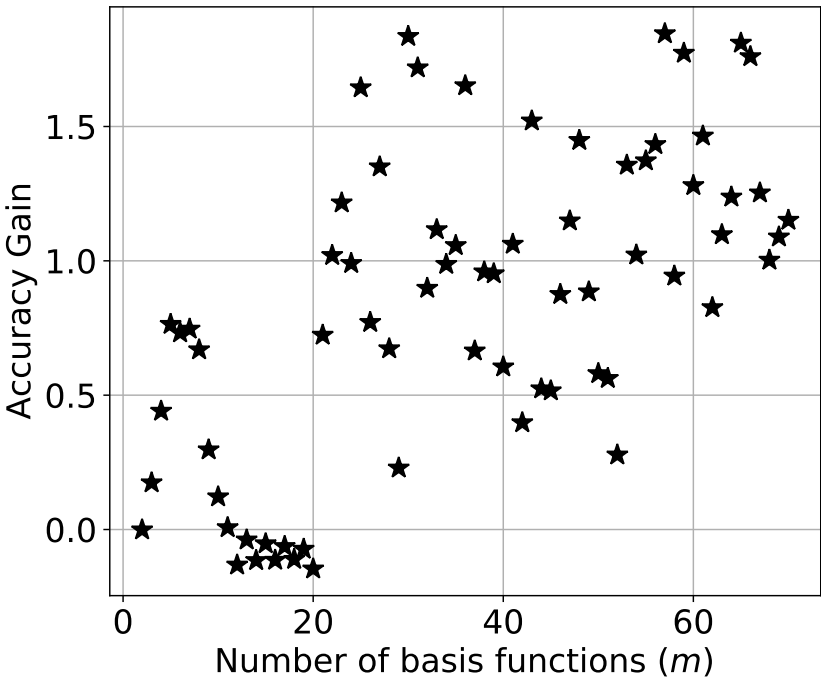


Figure 4.7: Accuracy gain of TFC vs. spectral for the solution of Lane-Emdem ($a = 1$). The accuracy gain is quantified in terms of $\log_{10}(\frac{\text{spectral method error}}{\text{TFC error}})$, and therefore, the y-axis is by orders of magnitude. For example, when this value is greater than zero, TFC is more accurate, and vice-versa.

Figure 4.8 shows that when more solution accuracy is needed, the RK45 method requires more time to solve the problem, while the spectral and TFC method see little change in computation time. In this case, TFC is slightly more accurate and faster than the spectral method.

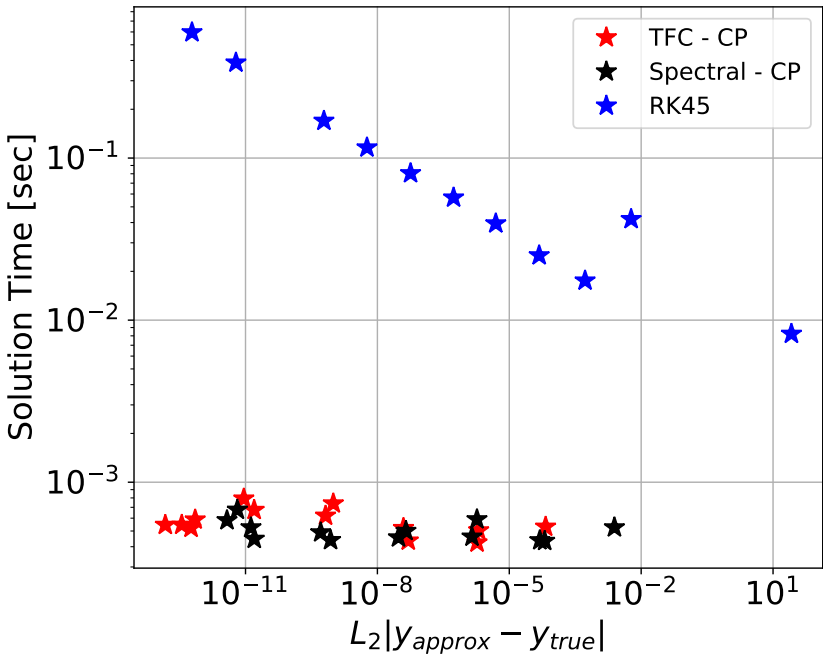


Figure 4.8: Timed solution of Lane-Emdem ($a = 1$).

Example 4.3: Lane-Emden ($a = 5$)

In this example, the Lane-Emden equation is solved for $a = 5$ on the domain $x \in [0, 10]$. The results given in Figures 4.9-4.10 compare the TFC method to the spectral method with a varying number of basis terms. Additionally, Figure 4.11 provides a speed versus accuracy comparison of the techniques mentioned above, along with the RK45 technique.

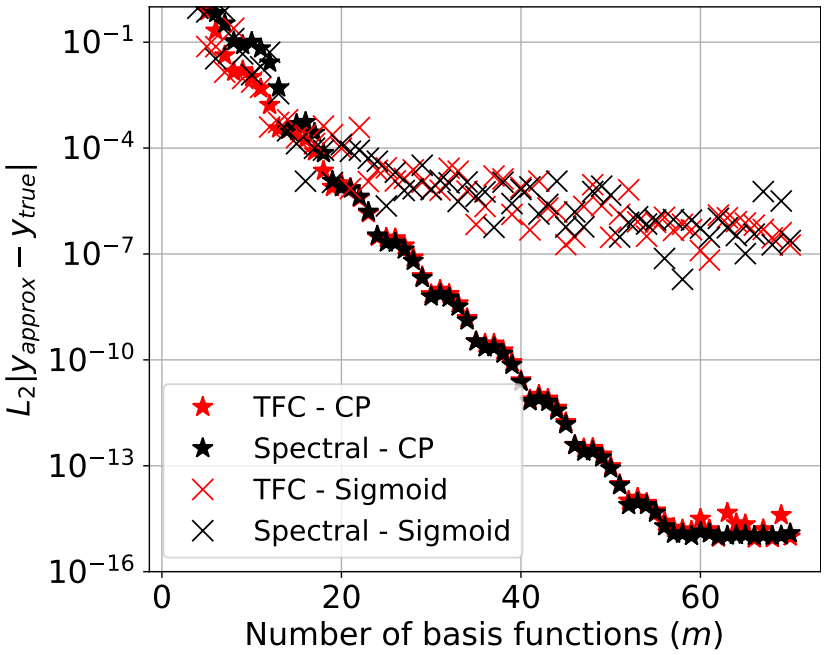


Figure 4.9: Accuracy of TFC and spectral method for varying number and types of basis functions for the Lane-Emdem equation ($a = 5$).

Looking at the TFC based solutions given in Figure 4.9, it can be seen that the orthogonal polynomial definition of the free function quickly reaches a minimum at 62 basis terms. Furthermore, even by adding basis terms, the ELM-based free functions do not match the Chebyshev orthogonal polynomials' accuracy. In fact, the solution with the sigmoid function is seven orders of magnitude less accurate.

Next, comparing the TFC method with the spectral method in Figure 4.10, we can see a slight accuracy gain for the TFC method until about 20 terms, where spectral and TFC method are the same in terms of accuracy. Then, around 60 terms, the spectral method has a slight accuracy gain.

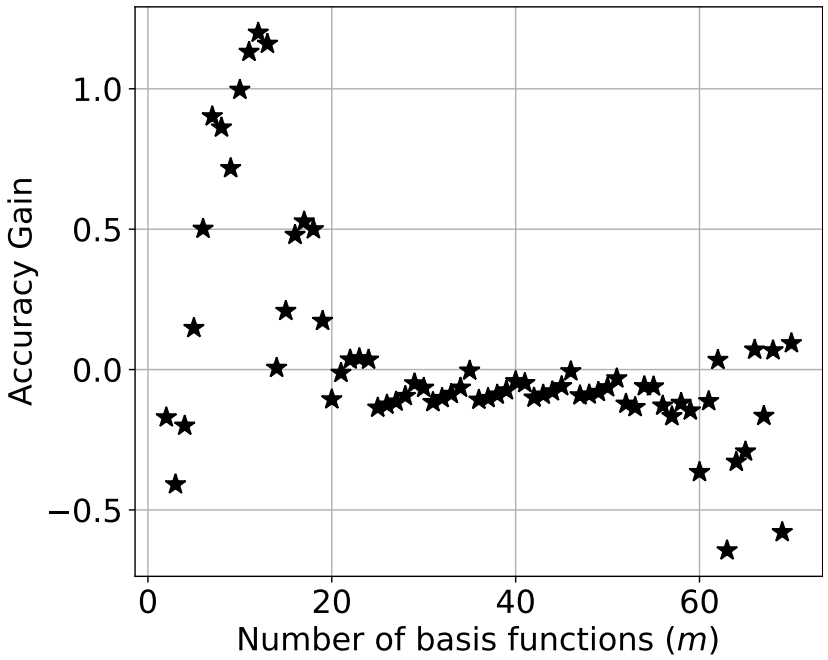


Figure 4.10: Accuracy gain of TFC vs. spectral for the solution of Lane-Emdem ($a = 5$). The accuracy gain is quantified in terms of $\log_{10}(\frac{\text{spectral method error}}{\text{TFC error}})$, and therefore, the y-axis is by orders of magnitude. For example, when this value is greater than zero, TFC is more accurate, and vice-versa.

In Figure 4.11, we can see when more solution accuracy is needed, the RK45 method requires more time to solve the problem; however, in this case, we do see a similar trend in the spectral and TFC method where the speed is reduced for more accurate solutions. As for the comparison between spectral and TFC method, in this test, the TFC method is slightly faster to converge.

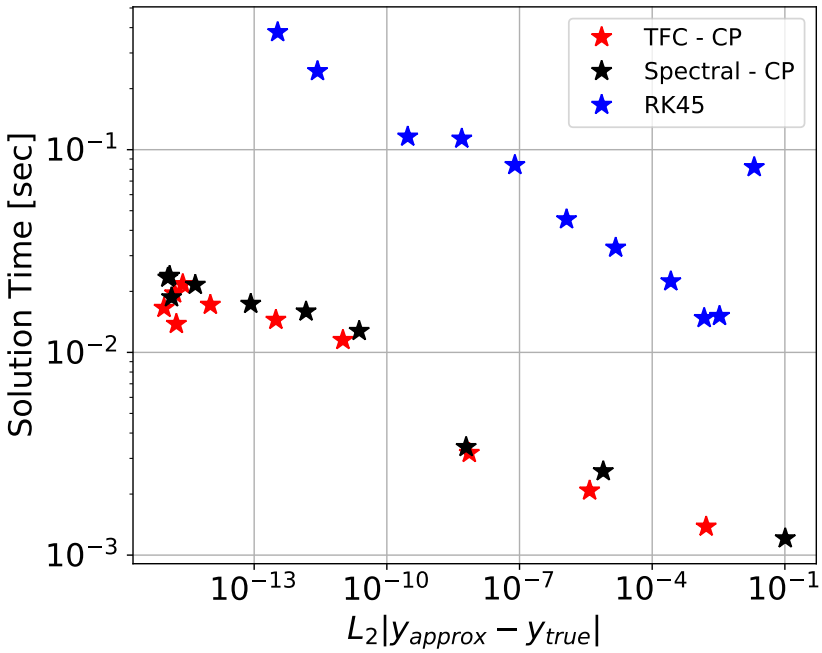


Figure 4.11: Timed solution of Lane-Emdem ($a = 5$).

4.6 Boundary-value problem

From Section 4.5.1 and Section 4.5.2 is was observed that solving nonlinear differential equations with TFC is the same as solving linear differential equations with one exception: the nonlinear case requires multiple iterations to solve for $\boldsymbol{\xi}$. In fact, the TFC approach is a unified approach to solve differential equations, meaning that the solution method is the same regardless of the constraints. This property results from the constrained expression, which decouples the differential equation constraints from the dynamics. To highlight this, let's consider the solution of a two-point boundary value problem,

$$y_{xx} + yy_x = f(x) \quad \text{subject to:} \quad \begin{cases} y(0) = 0 \\ y(\pi) = 0 \end{cases} \tag{4.22}$$

such that $f(x) = e^{-2x}\sin(x)\Big(\cos(x) - \sin(x)\Big) - 2e^{-x}\cos(x)$. Using the our generalized theory, the projection functionals are,

$$\rho_1(x, g(x)) = -g(0) \qquad \text{and} \qquad \rho_2(x, g(x)) = -g(\pi).$$

Again, the switching functions are determined by choosing the support functions $s_1 = 1$ and $s_2 = x$ and solving for the coefficients α_{ij},

$$\begin{bmatrix} 1 & 0 \\ 1 & \pi \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & \pi \end{bmatrix}^{-1} = \frac{1}{\pi} \begin{bmatrix} \pi & 0 \\ -1 & 1 \end{bmatrix}$$

which leads to the switching functions,

$$\phi_1(x) = \frac{\pi - x}{\pi} \qquad \text{and} \qquad \phi_2(x) = \frac{x}{\pi}.$$

The constrained expression in terms of $\boldsymbol{\xi}$ is

$$\begin{aligned} y(x, \boldsymbol{\xi}) &= \Big(\boldsymbol{h} - \frac{\pi - x}{\pi}\boldsymbol{h}(z_0) - \frac{x}{\pi}\boldsymbol{h}(z_f)\Big)^{\mathrm{T}}\boldsymbol{\xi} \\ y_x(x, \boldsymbol{\xi}) &= \Big(c\,\boldsymbol{h}_z + \frac{1}{\pi}\boldsymbol{h}(z_0) - \frac{1}{\pi}\boldsymbol{h}(z_f)\Big)^{\mathrm{T}}\boldsymbol{\xi} \\ y_{xx}(x, \boldsymbol{\xi}) &= \Big(c^2\,\boldsymbol{h}_{zz}\Big)^{\mathrm{T}}\boldsymbol{\xi}. \end{aligned}$$

Just like the Lane-Emden initial-value problem, the constraints are embedded, and we have a transformed differential equation subject to no constraints. Therefore, the last step is to form the loss vector and Jacobian and solve for the coefficients using our nonlinear least-squares method. Therefore, it should now be clear by this example that the process of solving the differential equations is unaffected by different constraint types. For completeness, the

associated loss function, loss vector, and Jacobian are provided below.

$$\tilde{F} = y_{xx} + y\, y_x - f(x) = 0$$

$$\mathbb{L}(\boldsymbol{\xi}) = \begin{Bmatrix} \tilde{F}(x_0, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_k, \boldsymbol{\xi}) \\ \vdots \\ \tilde{F}(x_f, \boldsymbol{\xi}) \end{Bmatrix} = \begin{Bmatrix} y_{xx}(x_0, \boldsymbol{\xi}) + y(x_0, \boldsymbol{\xi})\, y_x(x_0, \boldsymbol{\xi}) - f(x_0) \\ \vdots \\ y_{xx}(x_k, \boldsymbol{\xi}) + y(x_k, \boldsymbol{\xi})\, y_x(x_k, \boldsymbol{\xi}) - f(x_k) \\ \vdots \\ y_{xx}(x_f, \boldsymbol{\xi}) + y(x_f, \boldsymbol{\xi})\, y_x(x_f, \boldsymbol{\xi}) - f(x_f) \end{Bmatrix}$$

$$\mathbb{J}(\boldsymbol{\xi}) = \begin{bmatrix} \Big[c^2\, \boldsymbol{h}_z z(z_0) + y(x_0, \boldsymbol{\xi})\Big(c\, \boldsymbol{h}_z(z_0) + \frac{1}{\pi}\boldsymbol{h}(z_0) - \frac{1}{\pi}\boldsymbol{h}(z_f)\Big) + y_x(x_0, \boldsymbol{\xi})\Big(\boldsymbol{h}(z_0) - \frac{\pi - x_0}{\pi}\boldsymbol{h}(z_0) - \frac{x_0}{\pi}\boldsymbol{h}(z_f)\Big)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[c^2\, \boldsymbol{h}_z z(z_k) + y(x_k, \boldsymbol{\xi})\Big(c\, \boldsymbol{h}_z(z_k) + \frac{1}{\pi}\boldsymbol{h}(z_0) - \frac{1}{\pi}\boldsymbol{h}(z_f)\Big) + y_x(x_k, \boldsymbol{\xi})\Big(\boldsymbol{h}(z_k) - \frac{\pi - x_k}{\pi}\boldsymbol{h}(z_0) - \frac{x_k}{\pi}\boldsymbol{h}(z_f)\Big)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[c^2\, \boldsymbol{h}_z z(z_f) + y(x_f, \boldsymbol{\xi})\Big(c\, \boldsymbol{h}_z(z_f) + \frac{1}{\pi}\boldsymbol{h}(z_0) - \frac{1}{\pi}\boldsymbol{h}(z_f)\Big) + y_x(x_f, \boldsymbol{\xi})\Big(\boldsymbol{h}(z_f) - \frac{\pi - x_f}{\pi}\boldsymbol{h}(z_0) - \frac{x_f}{\pi}\boldsymbol{h}(z_f)\Big)\Big]^{\mathrm{T}} \end{bmatrix}$$

.

Example 4.4: Solution to two-point boundary-value problem

In this example, the two-point boundary-value problem given by Equation (4.22). The results given in Figures 4.12-4.13 compare the TFC method to both spectral method and ELMs based on the number of basis terms used. Additionally, Figure 4.14 provides a speed versus accuracy comparison of the techniques mentioned above along a 4th order collocation algorithm with the control of residuals from SciPy's `scipy.integrate.solve_bvp` algorithm [25].

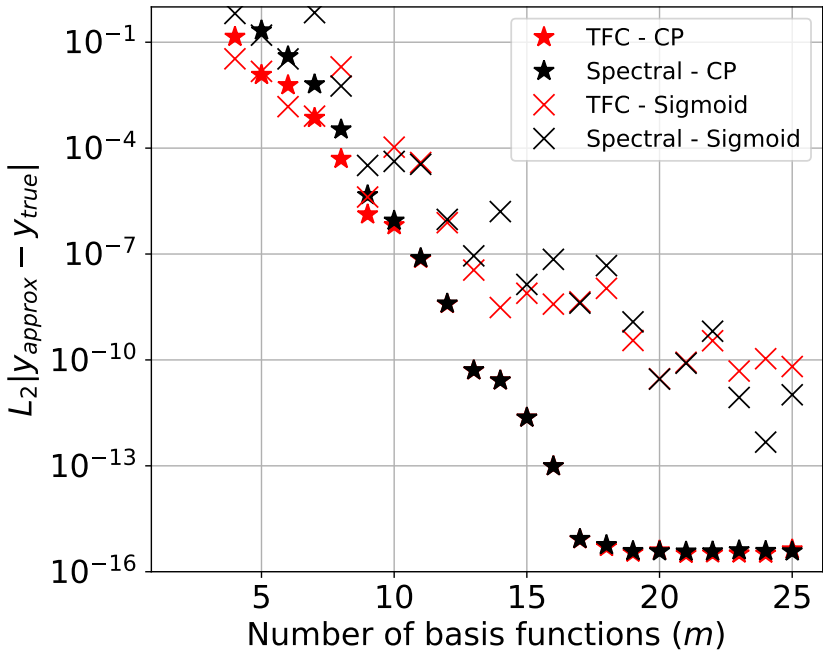


Figure 4.12: Accuracy of TFC and spectral method for varying number and types of basis functions for the boundary-value problem.

Looking at the TFC based solutions given in Figure 4.12, it can be seen that similar to the solutions of the Lane-Emden differential equation, the orthogonal polynomial definition of the free function is superior. Additionally, at 22 Chebyshev basis terms, both TFC and spectral method reach a minimum with respect to solution error. Further analysis shows that that the ELM based free functions are at least 3 orders of magnitude less accurate than the orthogonal polynomials; it is clear that using TFC with orthogonal polynomials to solve ordinary differential equations is the preferred approach. Therefore, after this example, all following examples will utilize Chebyshev or Legendre polynomials as the free function.

Next, the comparison of the TFC method with the spectral method is given in Figure 4.13. We can see a slight accuracy gain for the TFC method until about ten terms, where the spectral and TFC method are the same in terms of accuracy. Then,

around 20 terms, the TFC method has a slight accuracy gain.

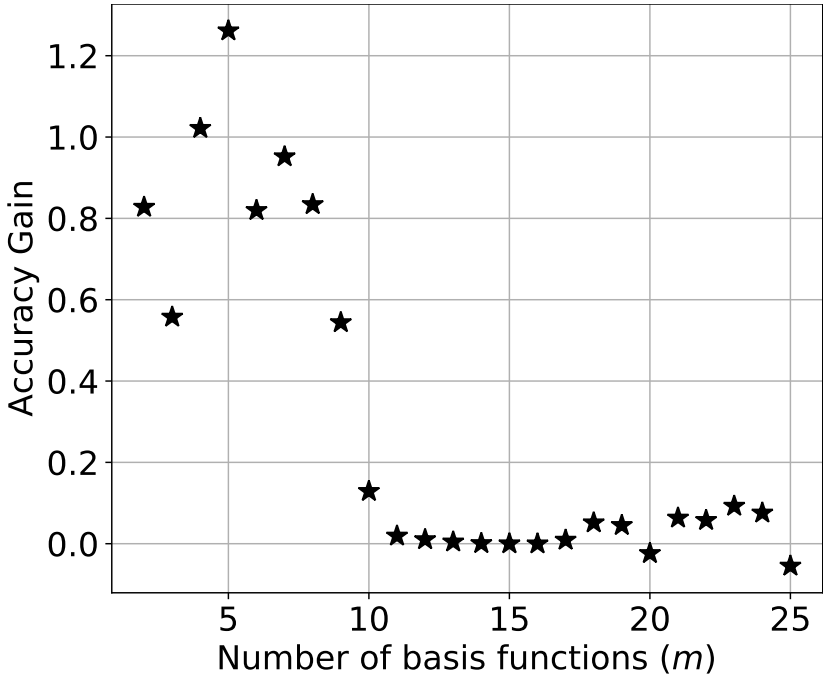


Figure 4.13: Accuracy gain of TFC vs. spectral method for the solution of the simple boundary-value problem. The accuracy gain is quantified in terms of $\log_{10}(\frac{\text{spectral method error}}{\text{TFC error}})$ and therefore, the y-axis is by orders of magnitude.

Finally, in Figure 4.14, a comparison of computation time is given for all of the previous techniques along with the RK45 method. Again, when more solution accuracy is needed, the RK45 method paired with a shooting method requires more time to solve the problem. Additionally, the maximum accuracy obtained from this method is on the order of 10^{-11}. For the spectral and TFC based methods, we notice only a slight increase in computation time with increasing accuracy, and the TFC method is slightly faster.

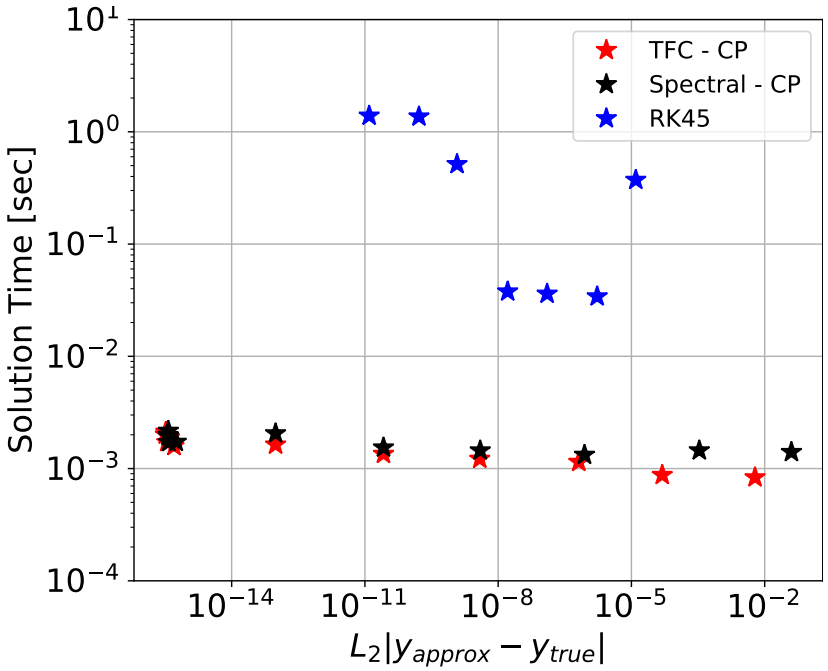


Figure 4.14: Timed Solution of BVP.

4.7 Solving systems of ordinary differential equations

The process discussed to solve single differential equations can directly be used to solve systems of differential equations. In general, we can consider a vector function $\boldsymbol{v}(t) : \mathbb{R} \to \mathbb{R}^n$ where $\boldsymbol{v}(t) = \{v_1(t), v_2(t), \cdots, v_n(t)\}^{\mathrm{T}}$ where $v_i : \mathbb{R} \to \mathbb{R}$ or in a vector-sense, the components of the vector. This vector function is subject to some set of differential equations and constraints imposed on the v_i components. Therefore, just as we have done in the single differential equation examples, a system of differential equations can be solved by deriving the constrained expressions for the n component functions according to the theory provided in Chapters 2 and 3. In fact, if constraints are shared between components, the theory can easily incorporate these constraints (see Example 2.4.4). Finally, these constrained expressions can be parameterized by defining n free functions and creating a system of algebraic equations that must then be discretized and solved as usual.

4.8 Two major extensions for use in optimal control problems

Until now, we have dealt with ordinary differential equations where 1) the free function $g(x)$ is expressed as an orthogonal polynomial set that can accurately and completely describe the solution and 2) the integration range was explicitly stated (i.e., the initial and final time of the problems were known). However, in many optimal control problems, we run into two scenarios that cause issues with the standard framework. Thus, extra theory must be developed to handle it; however, the tools and concepts developed in the earlier sections make this task an effortless step forward.

4.8.1 A hybrid systems approach*

First, we need to adapt the constrained expression for use in hybrid systems. The original adaptation was spurred by the problem of bang-bang control structure inherent in the fuel optimal landing problem solved in Johnston et al. [2] and explored in more detail in Johnston and Mortari [1]. By definition, hybrid systems are dynamical systems governed by a time-sequence of differential equations, either linear or nonlinear. A simple example is a bouncing ball where the motion is described by a sudden variation (or jump) in the dynamics when the ball impacts the ground, shown in Figure 4.15. These systems become even more common in the study of control problems where a dynamical system is controlled by discrete controls (e.g., bang-bang control). In fact, these are considered a special case of hybrid systems called variable structure systems (VSS), and the study of the control of these systems is named variable structure control (VSC) [55].

Initial-value problems for these systems can be easily solved by propagating the initial conditions over the domain of the first differential equation in the sequence. The final conditions can then be used as the initial conditions for the next differential equation, and the process can be repeated indefinitely (ignoring any accumulation of numerical error).

*Reprinted (along with revisions and updates unique to this dissertation) by permission from Elsevier the Journal of Computational and Applied Mathematics "Least-squares solutions of boundary-value problems in hybrid systems," Johnston, H. and Mortari, D., 2021, J. Comput. Appl. Math., 393, 113524, Copyright 2021, [1]

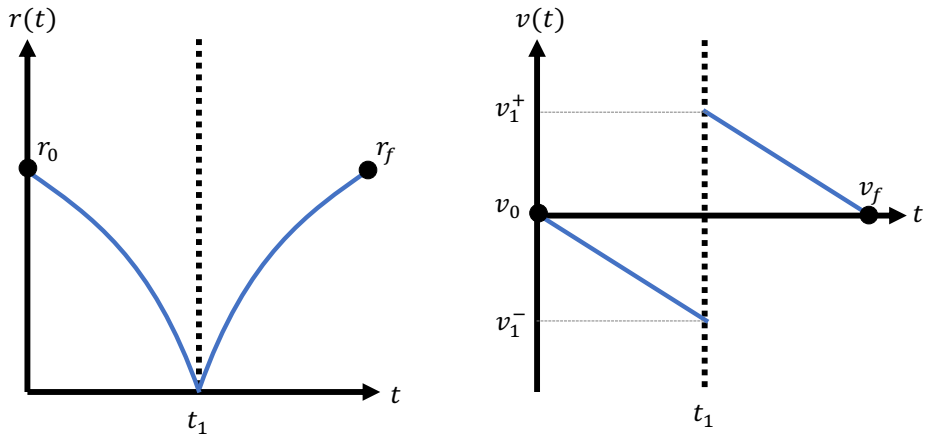


Figure 4.15: Graphical representation of the bouncing ball hybrid system. Reprinted with permission from [1].

However, boundary-value problems do not offer this luxury and will be the main focus of the proceeding section. The study of these problems is not new, and numerical techniques to solve these problems have existed since the 1960s, based on the shooting method [56, 57, 58, 59] detailed in Figure 4.16. In these approaches, the interval is divided over multiple sub-intervals, and the boundary-value problem is converted to multiple initial-value problems. The unknown boundary conditions are then solved by minimizing the DE residuals and the residuals of function and derivative continuities connecting all sub-intervals. In practice, root solving techniques (bisection, Newton's method, etc.) are used to minimize all residuals. In general, even when two subsequent linear differential equations are connected, solutions based on a shooting method requires an initial guess of the unknown parameters that are used to iterate until the solution is obtained. Note that the convergence is dictated by the initial guess [60], and it is not guaranteed. Regardless, studies have been conducted to quantify these methods' error once an approximation is obtained [61, 62].

Other techniques for solving these problems include finite difference and finite element methods. A finite difference method where the differential equation is approximated by a difference equation that converts the problem into a system of equations that are solved using linear algebra techniques. On the other hand, in finite element methods (collocation,

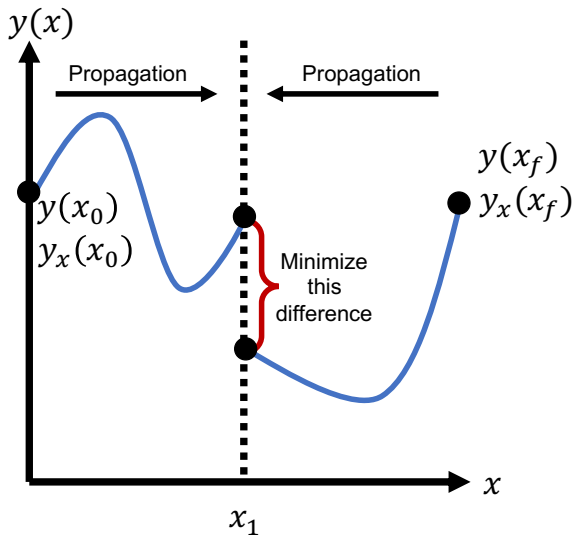


Figure 4.16: Graphical representation of shooting method. Reprinted with permission from [1].

Galerkin, etc.) [63], the problem is split into smaller parts called finite elements. Simple approximated equations are used to model these elements. These elements are then assembled into a larger system of equations that model the entire problem. The finite difference and finite element method's major drawback is the number of subdivisions needed to capture large variations in the solution.

The simplest example of a hybrid system is a differential equation with a discrete jump in the dynamic behavior at a single point along the domain. When solving a two-point BVP according to these dynamics, not only must the solution satisfy the boundary condition, but it must also preserve the C^1 continuity over the jump. The differential equation associated with the single switch in dynamics can be expressed in its explicit form by,

$$\begin{cases} {}^{(1)}F(x, y, y_x, y_{xx}) = 0 & \text{for } x \leq x_1 \\ {}^{(2)}F(x, y, y_x, y_{xx}) = 0 & \text{for } x > x_1 \end{cases} \qquad \text{subject to: } \begin{cases} y(x_0) = y_0 \\ y(x_f) = y_f \end{cases}$$

where $x_1 \in (x_0, x_f)$ and ${}^{(1)}F(x, y, y_x, y_{xx})$ and ${}^{(2)}F(x, y, y_x, y_{xx})$ are both functions of the independent variable x, the function y, and its derivatives. For this system, a separate

constrained expression for each segment must be derived. Additionally, at the boundary of the differential equations, in this case x_1, continuity must be enforced. Figure 4.17 depicts the constrained expression over the two differential equation segments. This can be

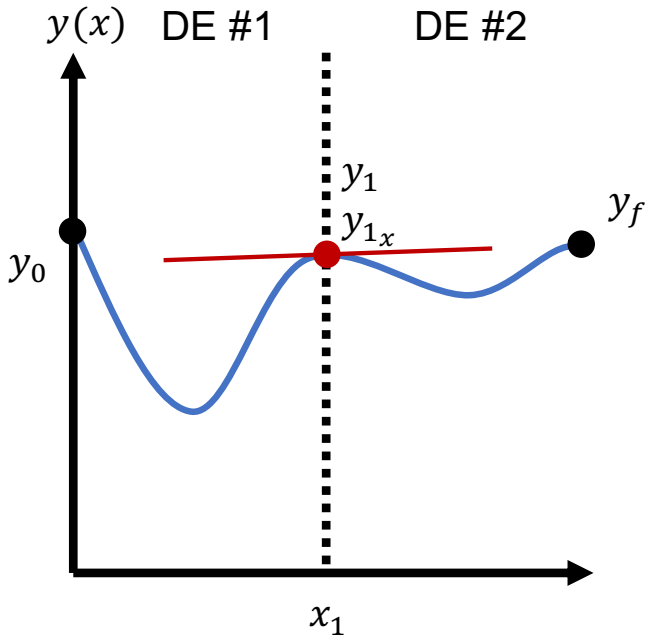


Figure 4.17: Illustration of piecewise TFC approach enforcing C^1 continuity over two segments. Reprinted with permission from [1].

done by considering each segment independently and introducing two new unknown values $y(x_1) = y_1$ and $y_x(x_1) = y_{1_x}$, which are the value and derivative of the function at the intersection. Therefore, the constrained expression over the first segment must be written for an initial value, final value, and initial derivative, while the constrained expression over the second segment must be written for an initial value, initial derivative, and final value. Using the theory already developed (and using monomial support functions), these constrained

expressions take the form,

$$
\begin{aligned}
{}^{(1)}y(x, {}^{(1)}g(x)) = {}^{(1)}g(x) &+ {}^{(1)}\phi_1(x)\Big(y_0 - {}^{(1)}g(x_0)\Big) \\
&+ {}^{(1)}\phi_2(x)\Big(y_1 - {}^{(1)}g(x_1)\Big) + {}^{(1)}\phi_3(x)\Big(y_{1_x} - {}^{(1)}g_x(x_1)\Big) \qquad (4.23)
\end{aligned}
$$

$$
\begin{aligned}
{}^{(2)}y(x, {}^{(2)}g(x)) = {}^{(2)}g(x) &+ {}^{(2)}\phi_1(x)\Big(y_1 - {}^{(2)}g(x_1)\Big) \\
&+ {}^{(2)}\phi_2(x)\Big(y_{1_x} - {}^{(2)}g_x(x_1)\Big) + {}^{(2)}\phi_3(x)\Big(y_f - {}^{(2)}g(x_f)\Big) \qquad (4.24)
\end{aligned}
$$

where the switching functions are provided below,

$$
\begin{aligned}
{}^{(1)}\phi_1(x) &= \frac{1}{(x_1 - x_0)^2}\Big(x_1^2 - 2x_1x + x^2\Big) \\
{}^{(1)}\phi_2(x) &= \frac{1}{(x_1 - x_0)^2}\Big(x_0(x_0 - 2x_1) + 2x_1x - x^2\Big) \\
{}^{(1)}\phi_3(x) &= \frac{1}{x_1 - x_0}\Big(x_0x_1 - (x_0 + x_1)x + x^2\Big)
\end{aligned}
$$

$$
\begin{aligned}
{}^{(2)}\phi_1(x) &= \frac{1}{(x_f - x_1)^2}\Big(x_f(x_f - 2x_1) + 2x_1x - x^2\Big) \\
{}^{(2)}\phi_2(x) &= \frac{1}{x_f - x_1}\Big(- x_fx_1 + (x_f + x_1)x - x^2\Big) \\
{}^{(2)}\phi_3(x) &= \frac{1}{(x_f - x_1)^2}\Big(x_1^2 - 2x_1x + x^2\Big).
\end{aligned}
$$

The major result of the constrained expressions derived in Equations (4.23) and Equation (4.24) is that for all finite values of y_1 and y_{1_x}, C^1 continuity is satisfied. However, this formulation comes with one caveat. Since y_1 and y_{1_x} were considered arbitrary, they are free parameters that must be solved for when solving the differential equation. Therefore, for numerical implementation, this causes the number of parameters to be solved to scale with the number of segments in the hybrid system. We will find that this is not a major issue for ordinary differential equations.

4.8.1.1 Generalization for n segments

Suppose the problem is subject to n jumps in dynamics as detailed in Figure 4.18. This case is the generalization of the problem presented in Section 4.8.1. Additionally, this gen-

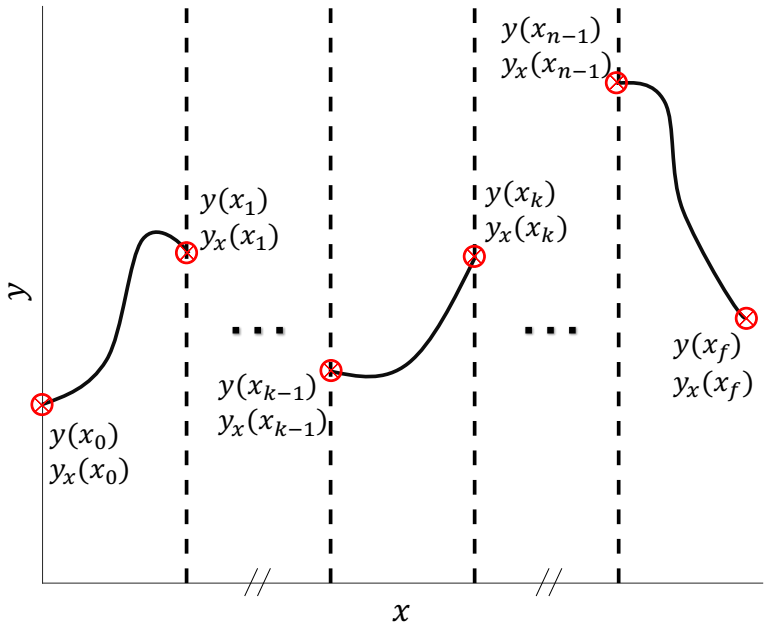


Figure 4.18: Illustration of segmented TFC approach to enforce C^1 continuity over n segments. Reprinted with permission from [1].

eralization necessitates the introduction of another set of switching functions that can be derived using the TFC method. Since our future applications will focus on optimal control problems governed by second-order dynamics, we will consider each segment constrained on both sides by point and derivative constraints. The constrained expression for this constraint type produces the equation,

$$
\begin{aligned}
{}^{(k)}y(x, {}^{(k)}g) = {}^{(k)}g(x) + {}^{(k)}\phi_1(x)\Big({}^{(k-1)}\beta - {}^{(k)}g(x_{k-1})\Big) + {}^{(k)}\phi_2(x)\Big({}^{(k)}\beta - {}^{(k)}g(x_k)\Big) \\
+ {}^{(k)}\phi_3(x)\Big({}^{(k-1)}\beta_x - {}^{(k)}g_x(x_{k-1})\Big) + {}^{(k)}\phi_4(x)\Big({}^{(k)}\beta_x - {}^{(k)}g_x(x_k)\Big),
\end{aligned}
$$

where $k = 0, 1, \cdots, n$, and ${}^{(k-1)}\beta$, ${}^{(k)}\beta$, ${}^{(k-1)}\beta_x$, and ${}^{(k)}\beta_x$ are the value and derivative continuity constraints when $0 < k < n$. The conditions at $k = 0$ and $k = n$ are defined by the boundary constraints. In this equation, the switching functions (when selecting the support functions as $s_1 = 1$, $s_2 = x$, $s_3 = x^2$, and $s_4 = x^3$) become,

$$
\begin{aligned}
{}^{(k)}\phi_1(x) &= \frac{1}{(x_k - x_{k-1})^3}\Big(- x_k^2(3x_{k-1} - x_k) + 6x_{k-1}x_k x - 3(x_{k-1} + x_k)x^2 + 2x^3\Big) \\
{}^{(k)}\phi_2(x) &= \frac{1}{(x_k - x_{k-1})^3}\Big(- x_{k-1}^2(x_{k-1} - 3x_k) - 6x_{k-1}x_k x + 3(x_{k-1} + x_k)x^2 - 2x^3\Big) \\
{}^{(k)}\phi_3(x) &= \frac{1}{(x_k - x_{k-1})^2}\Big(- x_{k-1}x_k^2 + x_k(2x_{k-1} + x_k)x - (x_{k-1} + 2x_k)x^2 + x^3\Big) \\
{}^{(k)}\phi_4(x) &= \frac{1}{(x_k - x_{k-1})^2}\Big(- x_{k-1}^2 x_k + x_{k-1}(x_{k-1} + 2x_k)x - (2x_{k-1} + x_k)x^2 + x^3\Big),
\end{aligned}
$$

where x_k denotes the boundaries of the segments. Lastly, by expressing the free function in the form of Equation (4.7) and discretizing the domains, the generalization can be written in a compact block diagonal matrix of the form,

$$
\boldsymbol{y} = \begin{bmatrix} \mathbb{A}_1 & \vdots & \mathbb{A}_2 \end{bmatrix} \Xi + \mathbb{B}
$$

where the terms of this equation are,

$$
\mathbb{A}_1 = \begin{bmatrix}
{}^{(1)}H & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\mathbf{0} & \cdots & {}^{(k)}H & \cdots & \mathbf{0} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\mathbf{0} & \cdots & \mathbf{0} & \cdots & {}^{(n)}H
\end{bmatrix}.
$$

In this block diagonal matrix, the terms of ${}^{(k)}H$ are matrices of the terms multiplied by ${}^{(k)}\boldsymbol{\xi}$. For example, the term ${}^{(1)}H$ is simply,

$$
{}^{(1)}H = \begin{bmatrix} \Big[\boldsymbol{h}(z_0) - {}^{(1)}\phi_1(x_0)\boldsymbol{h}(z_0) - {}^{(1)}\phi_2(x_0)\boldsymbol{h}(z_1) - {}^{(1)}\phi_3(x_0)\, c\, \boldsymbol{h}_z(z_0) - {}^{(1)}\phi_4(x_0)\, c\, \boldsymbol{h}_z(z_1)\Big]^{\mathrm{T}} \\ \vdots \\ \Big[\boldsymbol{h}(z_1) - {}^{(1)}\phi_1(x_1)\boldsymbol{h}(z_0) - {}^{(1)}\phi_2(x_1)\boldsymbol{h}(z_1) - {}^{(1)}\phi_3(x_1)\, c\, \boldsymbol{h}_z(z_0) - {}^{(1)}\phi_4(x_1)\, c\, \boldsymbol{h}_z(z_1)\Big]^{\mathrm{T}} \end{bmatrix}
$$

Next,

$$\mathbb{A}_2 = \begin{bmatrix}
{}^{(1)}\Phi_2 & {}^{(1)}\Phi_4 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
{}^{(2)}\Phi_1 & {}^{(2)}\Phi_3 & {}^{(2)}\Phi_2 & {}^{(2)}\Phi_4 & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & {}^{(k-1)}\Phi_2 & {}^{(k-1)}\Phi_4 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & {}^{(k)}\Phi_1 & {}^{(k)}\Phi_3 & {}^{(k)}\Phi_2 & {}^{(k)}\Phi_4 & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & {}^{(k+1)}\Phi_1 & {}^{(k+1)}\Phi_3 & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & {}^{(n-1)}\Phi_1 & {}^{(n-1)}\Phi_3 & {}^{(n-1)}\Phi_2 & {}^{(n-1)}\Phi_4 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & {}^{(n)}\Phi_1 & {}^{(n)}\Phi_3
\end{bmatrix},$$

and

$$ {}^{(k)}\Phi_j = \left\{ {}^{(k)}\phi_j(x_{k-1}) \quad \dots {}^{(k)}\phi_j(x_k) \right\}^{\mathrm{T}}, $$

is used for the switching functions ${}^{(k)}\phi_j(x)$ evaluated at the discretization points. Lastly,

$$ \mathbb{B} = \left\{ {}^{(1)}\Phi_1^{\mathrm{T}} \quad {}^{(1)}\Phi_3^{\mathrm{T}} \quad \mathbf{0}^{\mathrm{T}} \quad {}^{(n)}\Phi_2^{\mathrm{T}} \quad {}^{(n)}\Phi_4^{\mathrm{T}} \right\}^{\mathrm{T}}, $$

which is a vector associated with the boundary constraints. For this system, the unknown vector is,

$$ \begin{aligned} \Xi = \{ & {}^{(1)}\boldsymbol{\xi}^{\mathrm{T}} \quad \dots \quad {}^{(k)}\boldsymbol{\xi}^{\mathrm{T}} \quad \dots \quad {}^{(n)}\boldsymbol{\xi}^{\mathrm{T}} \\ & {}^{(1)}\beta \quad {}^{(1)}\beta_x \quad \dots \quad {}^{(k-1)}\beta \quad {}^{(k-1)}\beta_x \quad {}^{(k)}\beta \quad {}^{(k)}\beta_x \quad \dots \quad {}^{(n-1)}\beta \quad {}^{(n-1)}\beta_x \}^{\mathrm{T}}. \end{aligned} $$

Since this is a linear set of equations all subsequent derivatives are the derivatives of the individual components. The d-th order derivative of $\boldsymbol{y}$ becomes,

$$ \boldsymbol{y}^{(d)} = \begin{bmatrix} \mathbb{A}_1^{(d)} & \vdots & \mathbb{A}_2^{(d)} \end{bmatrix} \Xi + \mathbb{B}^{(d)}, $$

which is also a block diagonal matrix.

Moving forward, numerical examples are provided for two cases: 1) a hybrid system governed by a linear to nonlinear differential equation sequence and 2) the one-dimensional convection-diffusion equation. The solution of the convection-diffusion highlights that this technique can also be applied outside of hybrid systems—specifically when the dynamics of two regions in the differential equation behavior drastically different.

4.8.1.2 Linear-to-nonlinear differential equation sequence

Consider a second-order linear-nonlinear DE sequence such that,

$$y_{xx} + y(y_x)^a = -e^{\pi-2x} + e^{\pi/2-x} \quad \text{subject to: } \begin{cases} y(0) = \dfrac{9}{10} + \dfrac{1}{10}e^{\pi/2}(5 - 2e^{\pi/2}) \\ y(\pi) = e^{-\pi/2} \end{cases} \tag{4.25}$$

where the parameter a is determined by,

$$a = \begin{cases} 0 & \text{for } x \leq \pi/2 \\ 1 & \text{for } x > \pi/2 \end{cases}.$$

At the switch, $x_1 = \pi/2$, the differential equation changes from an linear differential equation to a nonlinear differential equation. This differential equation has the unique solution defined by,

$$y(x) = \begin{cases} = -\dfrac{1}{5}e^{\pi-2x} + \dfrac{1}{2}e^{\pi/2-x} + \dfrac{9\cos(x) + 7\sin(x)}{10} & \text{for } x \leq \pi/2 \\ = e^{\pi/2-x} & \text{for } x > \pi/2 \end{cases}.$$

Since the sequence has a nonlinear differential equation (over the second segment), an iterative least-squares approach is necessary. For this, we define the residual of the differential equation as the loss functions such that,

$$^{(1)}\tilde{F} = {}^{(1)}y_{xx} + {}^{(1)}y - e^{\pi/2} + e^{\pi/2-x} \tag{4.26}$$

$$^{(2)}\tilde{F} = {}^{(2)}y_{xx} + {}^{(2)}y\,{}^{(2)}y_x - e^{\pi/2} + e^{\pi/2-x} \tag{4.27}$$

where ${}^{(1)}y$, ${}^{(1)}y_{xx}$, ${}^{(2)}y$, ${}^{(2)}y_x$, and ${}^{(2)}y_{xx}$ are defined by the constrained expressions given by Equations (4.23)-(4.24), which have the unknown parameters ${}^{(1)}\boldsymbol{\xi}$, ${}^{(2)}\boldsymbol{\xi}$, y_1, and y_{1_x}. By substituting these equations into Equations (4.26) and (4.27) and taking the partials with respect to the unknown parameters, a Jacobian can be derived and ultimately used to solve

the differential equations. The analytical partials that form the Jacobian are provided in Appendix D.1.

Example 4.5: Results of linear-nonlinear differential equation sequence

Just like all other problems, this system can be solved using an iterative least-squares approach. However, an initial guess must be provided for the iterative least-squares. In the case of BVPs using the TFC method, the initial parameters can be determined by connecting the boundary constraints using a straight line. The line initial guess is adopted here in all the hybrid numerical tests provided. Therefore, the initial estimate of y_1 and y_{1_x} is automatically determined by this initialization. For this problem,

$$\Xi_0 = \left\{ \mathbf{0}^{\mathrm{T}} \quad \mathbf{0}^{\mathrm{T}} \quad \frac{y(\pi)-y(0)}{2} + y(0) \quad \frac{y(\pi)-y(0)}{\pi} \right\}^{\mathrm{T}}.$$

A visualization of this initial guess compared to the true solution is provided in Figure (4.19).

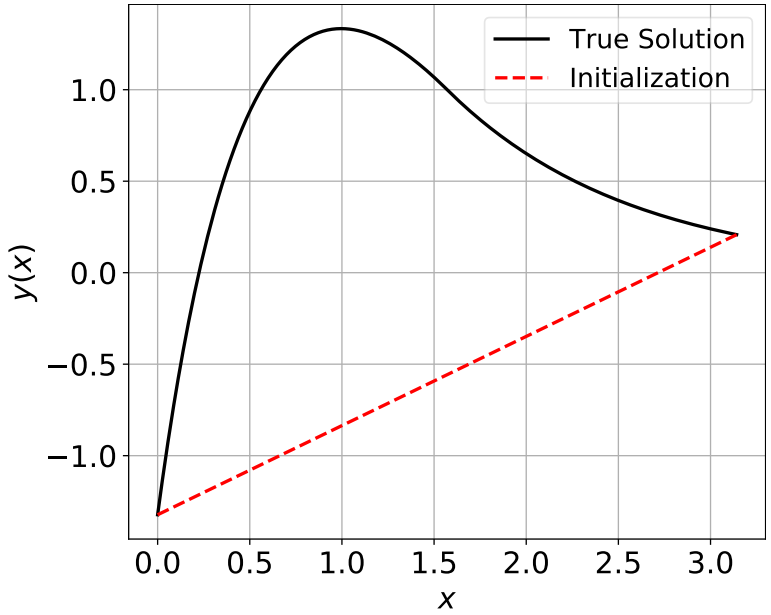


Figure 4.19: Initial guess and true solution for the linear-nonlinear sequence. Reprinted with permission from [1].

For the DE presented in Equation (4.25), $N = 100$ and $m = 16$ basis functions were used for each segment. The solution reached machine error accuracy in 15 iterations. The results of this numerical test are shown in Figures 4.20 and 4.21. The results show the function, its first two derivatives, and the associated absolute errors compared to the analytical solution.

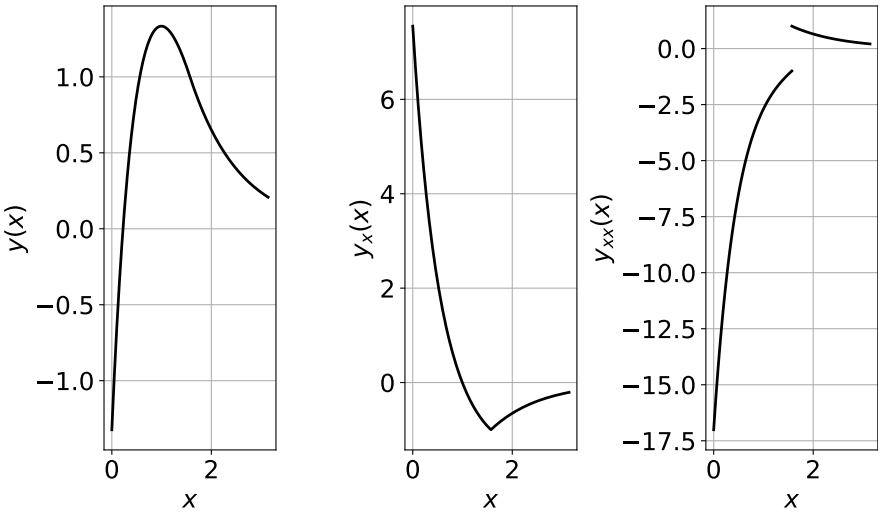


Figure 4.20: Solution of linear-nonlinear differential equation sequence. Reprinted with permission from [1].

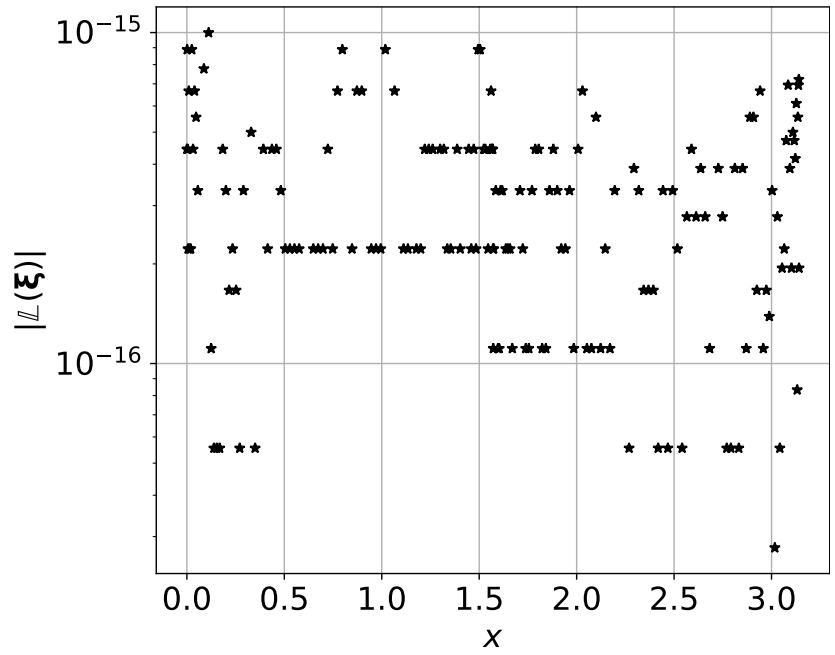


Figure 4.21: Absolute error of solution of linear-nonlinear differential equation sequence. Reprinted with permission from [1].

4.8.1.3 1D convection-diffusion equation

This technique doesn't just apply to hybrid systems. In fact, the concept of splitting the problem domain can be utilized when the dynamics exhibit transient behavior. To further highlight this concept, consider the example of the one-dimensional convection-diffusion equation defined by the differential equation,

$$y_{xx} - \mathrm{Pe}\, y_x = 0 \quad \text{subject to:} \begin{cases} y(0) = 1 \\ y(1) = 0 \end{cases} \tag{4.28}$$

with analytical solution

$$y = \frac{1 - e^{\mathrm{Pe}(x-1)}}{1 - e^{-\mathrm{Pe}}}.$$

In these equations, Pe is the Peclet number defined by the equation,

$$\text{Pe} = \frac{uL}{k} = \text{RePr} \approx \frac{\text{heat transported}}{\text{heat conducted}},$$

where u is the fluid velocity, L is the characteristic length, k is the thermal diffusivity of the fluid, Re is the Reynolds number, and Pr is the Prandtl number. We are interested in the behavior of the solution as the Peclet number increases, as shown in Figure 4.22. As

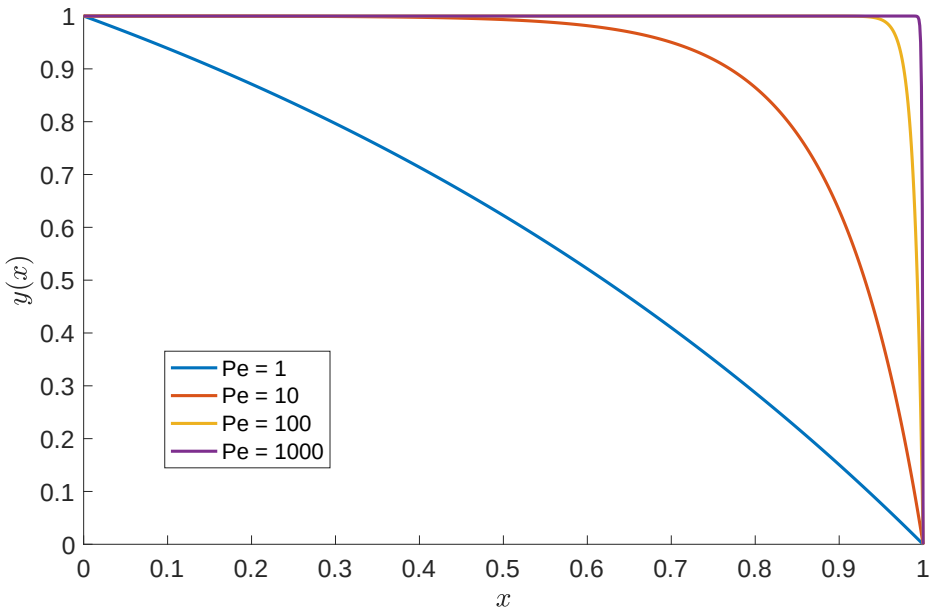


Figure 4.22: Solution of the 1D convection-diffusion equation for varying values of the Peclet number. As the Peclet number increases, the solution exhibits sharp transient behavior close to the endpoint. Reprinted with permission from [1].

Pe increases to around 100, the function begins to have a sharp transient behavior near the end of the domain. In order to solve this problem, let us consider a TFC solution where the domain is split into two segments such that the switch is defined at some value $x_1 \in (0, 1)$.

The constrained expression follow as,

$$
\begin{aligned}
{}^{(1)}y(z,{}^{(1)}\boldsymbol{\xi}) = \Big(\boldsymbol{h}(z) - {}^{(1)}\phi_1(z)\boldsymbol{h}(z_0) - {}^{(1)}\phi_2(z)\boldsymbol{h}(z_f) - {}^{(1)}\phi_3(z)\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\,{}^{(1)}\boldsymbol{\xi} \\
+ {}^{(1)}\phi_1(z)y_0 + {}^{(1)}\phi_2(z)y_1 + {}^{(1)}\phi_3(z)\frac{y_{1_x}}{{}^{(1)}c}
\end{aligned}
$$

$$
\begin{aligned}
{}^{(2)}y(z,{}^{(2)}\boldsymbol{\xi}) = \Big(\boldsymbol{h}(z) - {}^{(2)}\phi_1(z)\boldsymbol{h}(z_0) - {}^{(2)}\phi_2(z)\boldsymbol{h}_z(z_0) - {}^{(2)}\phi_3(z)\boldsymbol{h}(z_f)\Big)^{\mathrm{T}}\,{}^{(2)}\boldsymbol{\xi} \\
+ {}^{(2)}\phi_1(z)y_1 + {}^{(2)}\phi_2(z)\frac{y_{1_x}}{{}^{(2)}c} + {}^{(2)}\phi_3(z)y_f
\end{aligned}
$$

where the segment domains are defined in the basis domain z and ${}^{(1)}x \in [0, x_1] \rightarrow {}^{(1)}z \in [z_0, z_f]$ and ${}^{(2)}x \in [x_1, x_f] \rightarrow {}^{(2)}z \in [z_0, z_f]$. Since we have written the constrained expression in the basis domain, the derivative constraints must be divided by the mapping coefficient ${}^{(1)}c$ and ${}^{(2)}c$ to account for this. Next, the switching functions are defined as,

$$
\begin{aligned}
{}^{(1)}\phi_1(z) &= \frac{1}{(z_f - z_0)^2}\Big(z_f^2 - 2z_f z + z^2\Big) \\
{}^{(1)}\phi_2(z) &= \frac{1}{(z_f - z_0)^2}\Big(z_0(z_0 - 2z_f) + 2z_f z - z^2\Big) \\
{}^{(1)}\phi_3(z) &= \frac{1}{z_f - z_0}\Big(z_0 z_f - (z_0 + z_f)z + z^2\Big)
\end{aligned}
$$

for the first segment's constrained expression, and as

$$
\begin{aligned}
{}^{(2)}\phi_1(z) &= \frac{1}{(z_f - z_0)^2}\Big(z_f(z_f - 2z_0) + 2z_0 z - z^2\Big) \\
{}^{(2)}\phi_2(z) &= \frac{1}{z_f - z_0}\Big(- z_f z_0 + (z_f + z_0)z - z^2\Big) \\
{}^{(2)}\phi_3(z) &= \frac{1}{(z_f - z_0)^2}\Big(z_0^2 - 2z_0 z + z^2\Big)
\end{aligned}
$$

for the first segment's constrained expression. Now, we can rewrite the differential equation given by Equation (4.28) as,

$$\begin{cases} \text{for } x \leq x_1 : {}^{(1)}\tilde{F} = {}^{(1)}c^2\,{}^{(1)}y_{xx} - \mathrm{Pe}\,{}^{(1)}c\,{}^{(1)}y_x = 0 \\ \text{for } x \geq x_1 : {}^{(2)}\tilde{F} = {}^{(2)}c^2\,{}^{(2)}y_{xx} - \mathrm{Pe}\,{}^{(2)}c\,{}^{(2)}y_x = 0 \end{cases}.$$

To solve this differential equation, we could simply select the value of x_1 based on intuition and proceed with the same process as described earlier. However, it is highly likely that the selected value of x_1 will not be optimal and should therefore be a value that is optimized. Two methods exist to determine this value. The first method involves combining the TFC approach with an outer-loop optimizer (i.e., `fsolve`, a genetic algorithm, etc.) to solve for x_1. In this method, the TFC method supplies the estimated solution accuracy through the differential equation residuals, and the outer-loop optimizes the value of x_1 to minimize the residual. The second method is to include the solution of x_1 inside the TFC method. This can be realized by a single coefficient, since ${}^{(1)}c$ and ${}^{(2)}c$ are connected through the value x_1 by the equations

$$\begin{aligned} {}^{(1)}c &= \frac{z_f - z_0}{x_1 - x_0} \\ {}^{(2)}c &= \frac{z_f - z_0}{x_f - x_1} \end{aligned}$$

and ${}^{(2)}c$ can be rewritten in terms of ${}^{(1)}c := \bar{c}$

$${}^{(2)}c = \frac{\bar{c}(z_f - z_0)}{\bar{c}(x_f - x_0) - z_f + z_0} = \frac{\bar{c}\Delta z}{\bar{c} - \Delta z}$$

and

$$\frac{\partial\,{}^{(2)}c}{\partial \bar{c}} = -\frac{\Delta z^2}{(\bar{c} - \Delta z)^2}.$$

This reduces the mapping coefficient to a single parameter that can be plugged into the

constrained expressions and differential equation; however, doing so forces the system of equations to be nonlinear. The loss functions become,

$$\bar{\mathbb{L}}(\Xi) = \begin{bmatrix} {}^{(1)}\mathbb{L}(\Xi) & {}^{(2)}\mathbb{L}(\Xi) \end{bmatrix}^{\mathrm{T}}$$

with the unknown vector defined as,

$$\Xi = \begin{bmatrix} {}^{(1)}\boldsymbol{\xi} & {}^{(2)}\boldsymbol{\xi} & y_1 & y_{1_x} & \bar{c} \end{bmatrix}.$$

Additionally, the terms of the total loss vector and Jacobian are provided in Appendix D.2 for completeness.

Adaptation for other numerical techniques

While the equations above show the nonlinear least-squares approach to solve the problem, the equations can easily be adapted where an external optimizer handles the estimation of the optimal x_1 location ($\bar{c}$ in the above equations). By removing the unknown $\bar{c}$ from the equations, we are left with a linear set of equations defined by the same loss function and the updated unknown vector,

$$\Xi = \begin{bmatrix} {}^{(1)}\boldsymbol{\xi} & {}^{(2)}\boldsymbol{\xi} & y_1 & y_{1_x} \end{bmatrix}$$

By defining this TFC method where the input is x_1 (or $\bar{c}$) and the output is some function of the loss vector (in this case we use $\max|\mathbb{L}(\Xi)|$), a suite of optimizers can be leveraged.

Example 4.6: Results of the 1D convection-diffusion equation

For the numerical solution of the 1D convection-diffusion equation three methods where used: 1) a nonlinear least-squares (NLS) approach, and two approaches relying on the adaptation discuss earlier, 2) a differential evolution algorithm (DEvo) utilizing

SciPy's `optimize.differential_evolution()` and 3) SciPy's `optimize.fsolve()` algorithm. In all cases, $N = 200$ discretization points were used per segment and the basis functions were taken to the 190^{th} degree term ($m = 187$ basis functions). Additionally, the problem was solved for a range of Peclet numbers from 10^2 to 10^6 with a convergence criteria of $\varepsilon = 1 \times 10^{-13}$ and an initial guess of $x_1 = 0.75$. The results are captured in Table 4.1. For numerical stability of the algorithms, the upper bound of x_1 in the NLS and `fsolve()` approaches was set to be 0.9990 while the DEvo was set to 0.9999990.

In general, it can be seen that the algorithms have similar maximum errors; however, the location of the estimated x_1 value differs considerably, and the computation time of the NLS approach is two orders of magnitude faster than the differential evolution algorithm. The difference in x_1 is because x_1 is a numerical construct based on solving the differential equation. This value does not show up naturally in the equations, and therefore there is a potential of many local minima. This is very evident in the solution of the problem for $Pe = 10^2$ and 10^3 where the `fsolve` algorithm simply chooses the initial guess as the best solution, yet has similar accuracy to the other two methods.

Table 4.1: Solution for convention-diffusion equation using traditional TFC with nonlinear least-squares and with a genetic algorithm to solve for x_1 over a span of Peclet numbers. In all test cases, the number of points was $N = 200$ for each segment and the basis functions were taken to the 190^{th} degree term ($m = 187$ basis functions). Reprinted with permission from [1].

Type	Pe	max |Error|	max $\lvert L(\Xi)\rvert$	x_1	Computation time [s]
NLS	10^2	5.13×10^{-15}	7.28×10^{-12}	0.91127	1.10
NLS	10^3	5.36×10^{-14}	4.66×10^{-10}	0.91827	0.91
NLS	10^4	4.97×10^{-13}	5.96×10^{-8}	0.99000	0.53
NLS	10^5	4.22×10^{-12}	7.63×10^{-6}	0.99900	0.91
NLS	10^6	3.10×10^{-11}	6.10×10^{-4}	0.99900	3.82
DEvo	10^2	5.53×10^{-15}	4.15×10^{-12}	0.98374	9.25
DEvo	10^3	4.46×10^{-14}	2.95×10^{-10}	0.87589	10.88
DEvo	10^4	1.65×10^{-13}	2.20×10^{-8}	0.90771	11.10
DEvo	10^5	3.307×10^{-12}	3.53×10^{-6}	0.99838	10.50
DEvo	10^6	3.94×10^{-11}	2.66×10^{-4}	0.99945	9.72
`fsolve`	10^2	4.88×10^{-15}	4.81×10^{-12}	0.75000	2.11
`fsolve`	10^3	4.71×10^{-14}	2.60×10^{-10}	0.75000	1.54
`fsolve`	10^4	3.68×10^{-13}	2.09×10^{-8}	0.92199	4.51
`fsolve`	10^5	4.21×10^{-12}	4.10×10^{-6}	0.99900	1.54
`fsolve`	10^6	3.11×10^{-11}	5.79×10^{-4}	0.99900	1.75

4.8.2 Dealing with unspecified time and nonlinear constraints

Suppose we are faced with a problem that involves solving a differential equation subject to both linear and nonlinear boundary constraints along with an unknown final time. These conditions are typical of optimal control problems; therefore, let us consider a simple controls

problem,

$$\dot{x} = \alpha x + \beta u$$

$$\dot{u} = \beta x - \alpha u$$

subject to $x(0) = x_0$, $x(t_f) = x_f$, where t_f is unknown[3]. Additionally, the system must satisfy the algebraic constraint at the final time

$$\frac{1}{2}\Big(x^2(t_f) - u^2(t_f)\Big) - \frac{\alpha}{\beta}x(t_f)u(t_f) = 0.$$

Now, since the final time, t_f, is unknown, let us write the entire problem in the basis function domain z and map to the problem domain t using the parameter c from Equation (4.9). Therefore, the system of equations to be solved becomes,

$$F^x = c\,x_z - \alpha x - \beta u = 0 \tag{4.29}$$

$$F^u = c\,u_z - \beta x + \alpha u = 0 \tag{4.30}$$

$$f = \frac{1}{2}\Big(x^2(t_f) - u^2(t_f)\Big) - \frac{\alpha}{\beta}x(t_f)u(t_f) = 0 \tag{4.31}$$

Now, we use the developed method, but we write all constrained expressions in the z domain such that,

$$x(z, g^x(z)) = g^x(z) + \frac{z_f - z}{z_f - z_0}\Big(x_0 - g^x(z_0)\Big) + \frac{z - z_0}{z_f - z_0}\Big(x_f - g^x(z_f)\Big)$$

$$u(z, g^u(z)) = g^u(z)$$

where the function of $u(z)$ has no linear constraints and becomes solely a function of the free function, $g_u(z)$. By discretizing Equations (4.29), (4.30), and (4.31), we can construct

[3]Note, this system of equation is derived from the optimal control problem $\min J = \int_0^{t_f} \frac{1}{2}(x^2 + u^2)\,\mathrm{d}t$ subject to the dynamics $\dot{x} = \alpha x + \beta u$ constrained such that $x(0) = x_0$ and $x(t_f) = x_f$.

our typical loss vectors for each function,

$$\mathbb{L}^x = \left\{ F^x(z_0, \boldsymbol{\xi}_x, \boldsymbol{\xi}_u, c) \quad \dots \quad F^x(z_k, \boldsymbol{\xi}_x, \boldsymbol{\xi}_u, c) \quad \dots \quad F^x(z_f, \boldsymbol{\xi}_x, \boldsymbol{\xi}_u, c) \right\}^{\mathrm{T}}$$
$$\mathbb{L}^u = \left\{ F^u(z_0, \boldsymbol{\xi}_x, \boldsymbol{\xi}_u, c) \quad \dots \quad F^u(z_k, \boldsymbol{\xi}_x, \boldsymbol{\xi}_u, c) \quad \dots \quad F^u(z_f, \boldsymbol{\xi}_x, \boldsymbol{\xi}_u, c) \right\}^{\mathrm{T}}$$
$$\mathbb{L}^f = f(z_f, \boldsymbol{\xi}_x, \boldsymbol{\xi}_u)$$

which is collected in a total loss vector,

$$\mathbb{L} = \left\{ \mathbb{L}^{x\mathrm{T}} \quad \mathbb{L}^{y\mathrm{T}} \quad \mathbb{L}^f \right\}^{\mathrm{T}}.$$

In this problem, not only are the coefficients $\boldsymbol{\xi}_x$ and $\boldsymbol{\xi}_u$ unknowns, but the final time is also unknown, which is captured in our mapping parameter c, such that, $\Xi = \{\boldsymbol{\xi}_x, \boldsymbol{\xi}_u, c\}^{\mathrm{T}}$. Therefore, our Jacobian will also be populated by partial derivatives with respect to Ξ. The derivation of the Jacobian is left to the reader.

We rely again on the nonlinear least-squares approach to solve the problem since the final equation is nonlinear in the variables x and u. Yet, note that c defines a domain length and can never be negative. Therefore, let us change the definition for this variable such that $b^2 := c$. By doing this, we avoid the time domain parameter becoming negative, and the vector of unknowns becomes $\Xi = \{\boldsymbol{\xi}_x, \boldsymbol{\xi}_u, b\}^{\mathrm{T}}$. In summary, this simply changes Equations (4.29) and (4.30) to,

$$F^x = b^2\, x_z - \alpha x - \beta u = 0$$
$$F^u = b^2\, u_z - \beta x + \alpha u = 0,$$

in the development provided above.

Adaptation for other numerical techniques

Additionally, similar to our solution of the convection-diffusion equation in Section 4.8.1.3, we can remove the unknown value of t_f (which is related to b^2), creating a linear system of equations to be solved in $\mathbb{L}^x$ and $\mathbb{L}^u$. After solving the system, $|\mathbb{L}^f|$ can be used as the function to be minimized.

Example 4.7: Solution to free-final time problem

In this example, we have defined the coefficients as $\alpha = \beta = 1$, with the boundary conditions set as $x(0) = 1$ and $x(t_f) = 1$. Furthermore, all numerical systems were discretized with $N = 35$ points and used basis function up to the 30^{th} degree term (28 basis functions for $x(t)$ and 30 basis functions for $y(t)$). Lastly, the tolerance on the algorithms was set to $\varepsilon = 2.22 \times 10^{-16}$ and were initialized with $\boldsymbol{\xi}_x = \mathbf{0}$, $\boldsymbol{\xi}_u = \mathbf{0}$, and $t_f = 1$. For reference, the solution of $x(t)$ and $y(t)$ is highlighted in Figure 4.23.

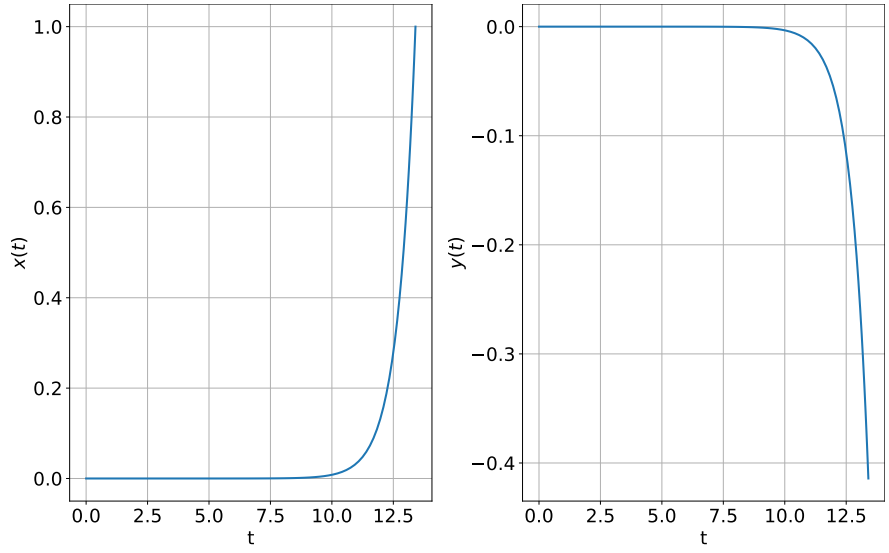


Figure 4.23: Time histories of the state.

The results of this test are provided in Table 4.2 where it can be seen that the

`fsolve` method is the most accurate in terms of $\max|\mathbb{L}(\Xi)|$ and $\max|H(t)|$, which is the Hamiltonian[a], parameter used to derive the problem and should be zero for all times. However, we can see that all methods differ at the sixth digit of the cost, which is defined as,

$$\text{Cost} = \frac{1}{2}\int_0^{t_f} \left(x^2(t) + u^2(t)\right) \mathrm{d}t,$$

and should be minimized in our case. Yet, where these solutions drastically differ is in the solution time, where the NLS approach is two orders of magnitude faster than the other approaches. This should be obvious since the NLS is the simplest approach to solving the problem. Additionally, since the cost isn't as sensitive to the final time, we see a large range of solutions for t_f. In all, the major consideration becomes a trade-off between solution accuracy versus computational time. In Chapters 6 and 7, we will take a deeper look into this regarding optimal control problems.

Table 4.2: Comparison of optimization scheme to solve the free final time problems.

Type	$\max\|L(\Xi)\|$	$\max\|H(t)\|$	Cost	t_f	Iterations	Comp. Time [s]
NLS	8.36×10^{-14}	8.36×10^{-14}	0.206**91**	11.13663	23	0.0457
DEvo	9.99×10^{-16}	5.55×10^{-17}	0.206**78**	13.92129	77	2.879
`fsolve`	5.18×10^{-16}	2.17×10^{-16}	0.206**82**	13.24100	62	2.520

[a]In this example the Hamiltonian is simply stated without definition. In Chapter 5, this term will be defined more rigorously.

4.9 A Solution of Lyapunov and Halo Orbits

According to Poincaré, "periodic orbits" provide the only gateway into the otherwise impenetrable domain of nonlinear dynamics. With the advent of space exploration, periodic

orbits have become an indispensable part of missions in space. The amazing fish-like Apollo orbit was the first three-body orbit used for space missions. The second three-body orbit used for space missions was the Halo orbit, discovered by Robert Farquhar in his Ph.D. thesis [64] under John Breakwell [65]. In 1978, Farquhar convinced NASA and led the International Sun-Earth Explorer 3 mission (ISEE3) to study the Sun from a Halo orbit around the Earths L1 Lagrange point. Farquhars original idea was to place a satellite in Halo orbit around the Lunar L2 for telecommunication support for the backside of the Moon. Today, this is indeed part of NASAs planned return of humans to the Moon in the next few years.

Typically, the standard method for computing periodic orbits is the differential correction method (also called the shooting method), as presented by Kathleen Howell [66]. One begins with an approximate solution obtained typically from normal form expansions. Using the variational equation, the guess solution is iteratively corrected for periodicity. Assuming the initial guess is in a reasonable basin of attraction to a periodic orbit, the process converges to a periodic orbit. In Hamiltonian systems, periodic orbits occur in 1-parameter families. Often, there are multiple families nearby. Hence, the convergence may not always lead to the desired orbit. Moreover, control over the specific features of the periodic orbit, such as its period or energy, requires additional work, for example, using continuation methods to reach the exact orbit desired. Using TFC, a simpler formulation and more efficient algorithm for finding periodic orbits is possible.

4.9.1 System dynamics

The circular-restricted three-body problem is a dynamical model used to describe the motion of a particle $\boldsymbol{r} = \{x, y, z\}^{\mathrm{T}}$ of negligible mass under the influence of a primary body of mass m_1 and secondary body of mass m_2. Furthermore, the orbits of m_1 and m_2 are subject to circular motion about the system's barycenter and lie in the x-y plane; the total system is depicted in Figure 4.24. Following this, the system can be non-dimensionalized by the following scaled units; unit mass is defined as $m_1 + m_2$; unit length is taken as the separation between m_1 and m_2; the unit time is chosen such that the orbits of m_1 and m_2

about the system's barycenter is 2π. By following these steps, the system can be reduced to a single parameter called the mass parameter, μ, where,

$$\mu = \frac{m_2}{m_1 + m_2}$$

From this, we define the terms μ_1 and μ_2 as

$$\mu_1 = 1 - \mu \quad \text{and} \quad \mu_2 = \mu.$$

Using this definition of the system, the equations of motion can be derived in the rotating

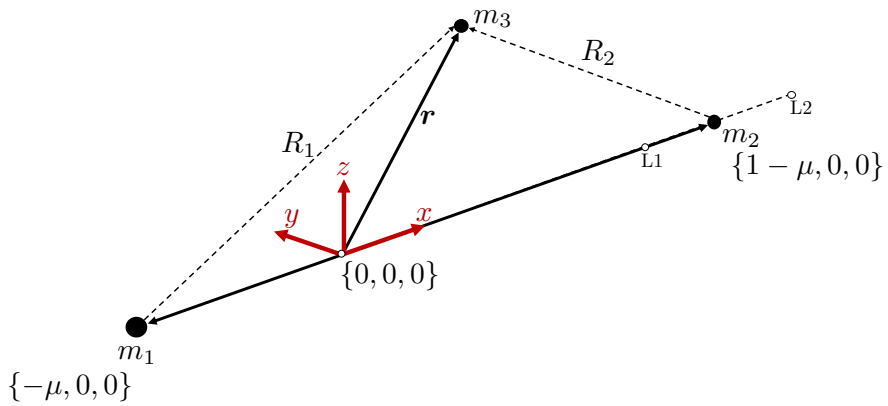


Figure 4.24: Schematic of the circular restricted three-body problem where the secondary body m_2 orbits around m_1 in a circular orbit. The third-body whose mass is $m_3 \ll m_2 < m_1$ is negligible and at a distance R_1 from m_1, R_2 from m_2, and $\boldsymbol{r}$ from the origin, which is the system barycenter (the system's center of mass).

frame leading to the following system of equations,

$$\begin{aligned} \ddot{x} - 2\dot{y} &= \frac{\partial \Omega}{\partial x} \\ \ddot{y} + 2\dot{x} &= \frac{\partial \Omega}{\partial y} \\ \ddot{z} &= \frac{\partial \Omega}{\partial z} \end{aligned} \tag{4.32}$$

Additionally, Ω is defined as,

$$\Omega(x, y, z) := \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{R_1} + \frac{\mu}{R_2} + \frac{1}{2}(1-\mu)\mu$$

where $R_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$ and $R_2 = \sqrt{(x+\mu-1)^2 + y^2 + z^2}$ are the distances to the primaries. Furthermore, the equations of motion are Hamiltonian and independent of time, and thus have an energy integral of motion E, where in the celestial mechanics community the Jacobi constant is used which is $C := -2E$ and given as,

$$C = 2\Omega - (\dot{x} + \dot{y} + \dot{z}) = (x^2 + y^2) + 2\frac{1-\mu}{R_1} + 2\frac{\mu}{R_2} + (1-\mu)\mu - (\dot{x} + \dot{y} + \dot{z}) \tag{4.33}$$

Moving forward, we will look to solve the dynamics defined by the system of equations in Equation (4.32) such that the orbit is at a fixed energy level (or rather Jacobi constant) using Equation (4.33). For our implementation, it is useful to define the residuals of these equations,

$$0 = F_x := \ddot{x} - 2\dot{y} - \frac{\partial \Omega}{\partial x} \tag{4.34}$$

$$0 = F_y := \ddot{y} + 2\dot{x} - \frac{\partial \Omega}{\partial y} \tag{4.35}$$

$$0 = F_z := \ddot{z} - \frac{\partial \Omega}{\partial z} \tag{4.36}$$

$$0 = F_c := (x^2 + y^2) + 2\frac{1-\mu}{R_1} + 2\frac{\mu}{R_2} + (1-\mu)\mu - (\dot{x} + \dot{y} + \dot{z}) - C. \tag{4.37}$$

Next, we look to generate analytical expressions for the states to guarantee a periodic orbit.

First, since in the problem the orbital period is unknown, the problem represents an unknown final time problem where we can define the problem domain as $t \in [0, T]$ where T is the period of the orbit and the basis domain is $\tau \in [-1, +1]$.[4] The final time (or the orbital period T) can be parameterized in the same manner as Section 4.8.2.

[4]Note, we have used τ here in place of z because in the common notation for this problem, z represents the z-component of the position of the body m_3.

Since we are looking for periodic orbits, we can utilize the constrained expression to satisfy the following constraints,

$$r_i(\tau_0) = r_i(\tau_f) = \alpha_i \quad \text{and} \quad \frac{\mathrm{d}r_i}{\mathrm{d}\tau}(\tau_0) = \frac{\mathrm{d}r_i}{\mathrm{d}\tau}(\tau_f) = \frac{\beta_i}{b^2}$$

where we define $\boldsymbol{r}(\tau) := \{r_x(\tau), r_y(\tau), r_z(\tau)\}^{\mathrm{T}} = \{x(\tau), y(\tau), z(\tau)\}^{\mathrm{T}}$. Since the trajectory must return to the initial state at some period T. The constrained expressions for the three components of position are as follows,

$$\begin{aligned} r_i(\tau, g_i(\tau)) = g_i(\tau) &+ \phi_1(\tau)\Big(\alpha_i - g(\tau_0)\Big) + \phi_2(\tau)\Big(\alpha_i - g(\tau_f)\Big) \\ &+ \phi_3(\tau)\Big(\frac{\beta_i}{b^2} - g_\tau(\tau_0)\Big) + \phi_4(\tau)\Big(\frac{\beta_i}{b^2} - g_\tau(\tau_f)\Big) \end{aligned} \tag{4.38}$$

where

$$\begin{aligned} \phi_1(\tau) &= \frac{1}{4}\Big(2 - 3\tau + \tau^3\Big) & \phi_2(\tau) &= \frac{1}{4}\Big(2 + 3\tau - \tau^3\Big) \\ \phi_3(\tau) &= \frac{1}{4}\Big(1 - \tau - \tau^2 + \tau^3\Big) & \phi_4(\tau) &= \frac{1}{4}\Big(-1 - \tau + \tau^2 + \tau^3\Big). \end{aligned}$$

By their definition, the projection functionals follow as,

$$\begin{aligned} \rho_1(x, g_i(x)) &= \alpha_i - g_i(\tau_0) & \rho_2(x, g_i(x)) &= \alpha_i - g_i(\tau_f) \\ \rho_3(x, g_i(x)) &= \frac{\beta_i}{b^2} - g_{\tau_i}(\tau_0) & \rho_4(x, g_i(x)) &= \frac{\beta_i}{b^2} - g_{\tau_i}(\tau_f). \end{aligned}$$

Then, as usual, the constrained expressions defined by Equation (4.38) are used to evaluate the three differential equations and one algebraic equation given in Equations (4.34), (4.35), (4.36), and (4.37) at the discretization points, which are ultimately used to construct a loss vector of the residuals of these equations.

$$\mathbb{L}_i(\Xi) = \left\{\tilde{F}_i(\tau_0, \Xi), \quad \ldots, \quad \tilde{F}_i(\tau_k, \Xi), \quad \ldots, \quad \tilde{F}_i(\tau_f, \Xi)\right\}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}_{N\times 1}$$

with the total loss vector of

$$\mathbb{L}(\Xi) = \left\{ \mathbb{L}_x^{\mathrm{T}}(\Xi), \;\; \mathbb{L}_y^{\mathrm{T}}(\Xi), \;\; \mathbb{L}_z^{\mathrm{T}}(\Xi), \;\; \mathbb{L}_c^{\mathrm{T}}(\Xi) \right\}^{\mathrm{T}} = \mathbf{0}_{4N\times 1}^{\mathrm{T}}$$

where the unknown vector is defined as,

$$\Xi = \left\{ \boldsymbol{\xi}_x^{\mathrm{T}}, \;\; \boldsymbol{\xi}_y^{\mathrm{T}}, \;\; \boldsymbol{\xi}_z^{\mathrm{T}}, \;\; \boldsymbol{\alpha}^{\mathrm{T}}, \;\; \boldsymbol{\beta}^{\mathrm{T}}, \;\; b \right\}^{\mathrm{T}} = \mathbf{0}_{(3m+7)\times 1}^{\mathrm{T}}.$$

4.9.2 Numerical Test

We consider the Earth-Moon system with the parameters given in Table 4.3. Additionally,

Table 4.3: Earth-Moon system parameters

Variable	Value
Earth mass m_1 [kg]	5.9724×10^{24}
Moon mass m_2 [kg]	7.346×10^{22}

for the TFC implementation, the parameters used are summarized in Table 4.4.

Table 4.4: TFC algorithm parameters

Variable	Value
N [number of points]	140
m [basis terms]	130
ε [tolerance]	2.22×10^{-16}
Maximum iterations	20

For all numerical tests, the unknown vector must be initialized. First, the terms $\boldsymbol{\xi}_x$, $\boldsymbol{\xi}_y$, and $\boldsymbol{\xi}_z$ were all initialized by a null vector, which ultimately represents the simplest

interpolating expression for the state variables. This initialization represents the worst-case scenario when there is no estimation of the trajectory. Next, the other unknown values of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and b (which are associated with the position, velocity, and the period of the orbit) were initialized using Richardson's third-order analytical method for Halo-type periodic motion [67].

This initialization was used to find the first orbit of the specified Jacobi constants. For the following orbits, the desired Jacobi constant was incrementally increased, and the converged values from the prior Jacobi constant level were used to initialize each step.

This same process was utilized for the differential corrector method, which was implemented as a point of comparison to TFC. In the differential corrector inner-loop, the desired Jacobi constant was obtained by an iterative least-squares approach to update the initial guess.

Example 4.8: Lyapunov orbits around L1 & L2 Lagrange points

First, the method was used to explore the computation of Lyapunov orbits, which lie in the x-y plane, or rather in the plane of the two primaries. For our test, the Lyapunov orbits were computed over a range of Jacobi constants, starting close to the equilibrium point's specific energy levels up to a Jacobi constant of 2.92. The associated trajectories for the orbits around L1 and L2 are provided in Figure 4.25 and Figure 4.28.

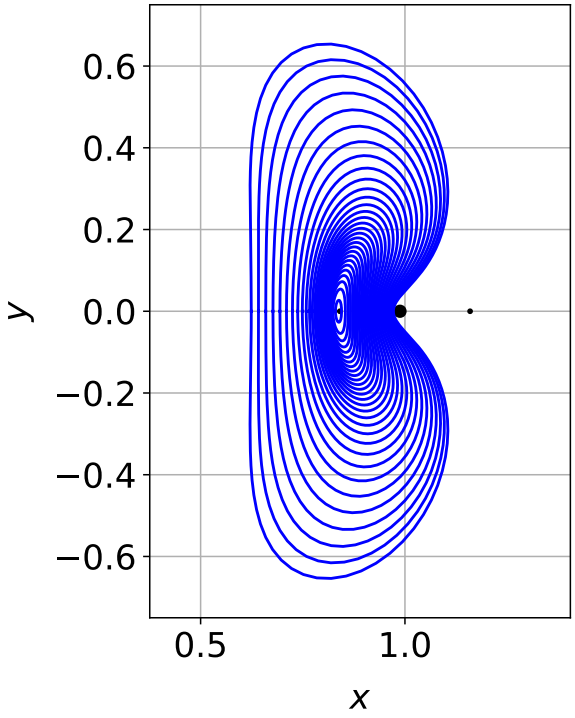


Figure 4.25: Lyapunov orbits for the Earth-Moon system for Jacobi constant values ranging from the energy of L1 to 2.92.

Additionally, a comparison with the differential corrector method (Reference [66]) is provided in terms of speed and accuracy. Figure 4.26 compares the residuals for both methods where it can be seen that the TFC approach is around 2 orders of magnitude more accurate than the differential corrector at higher Jacobi constants. Furthermore, the computation of the TFC solution is slightly faster, a little over 0.25 seconds in the extreme case, as displayed in Figure 4.27.

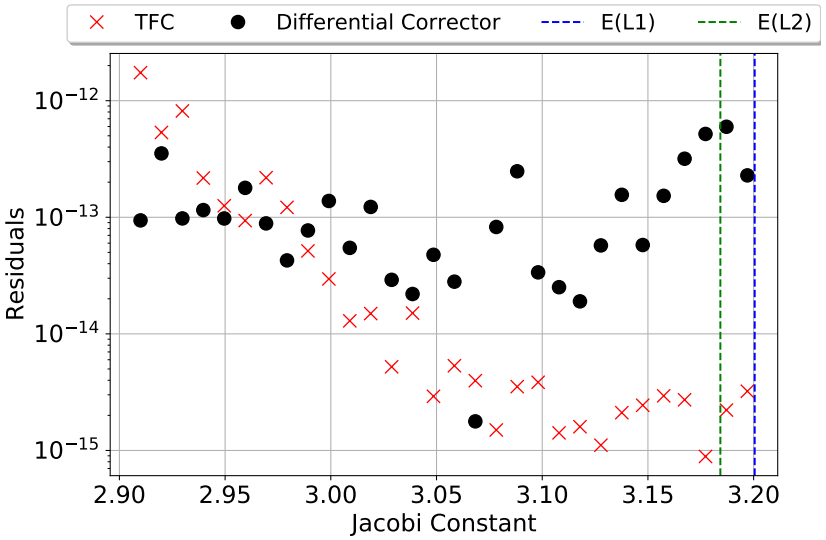


Figure 4.26: Maximum residuals of the loss vector for the TFC method solving for the trajectories plotted in Fig. 4.25 compared to that of the differential corrector. The lines of E(L1) and E(L2) represent the energy of the L1 and L2 Lagrange points respectively. The TFC approach has a slight accuracy advantage (an order-of-magnitude) as compared to the differential corrector method at higher Jacobi constants.

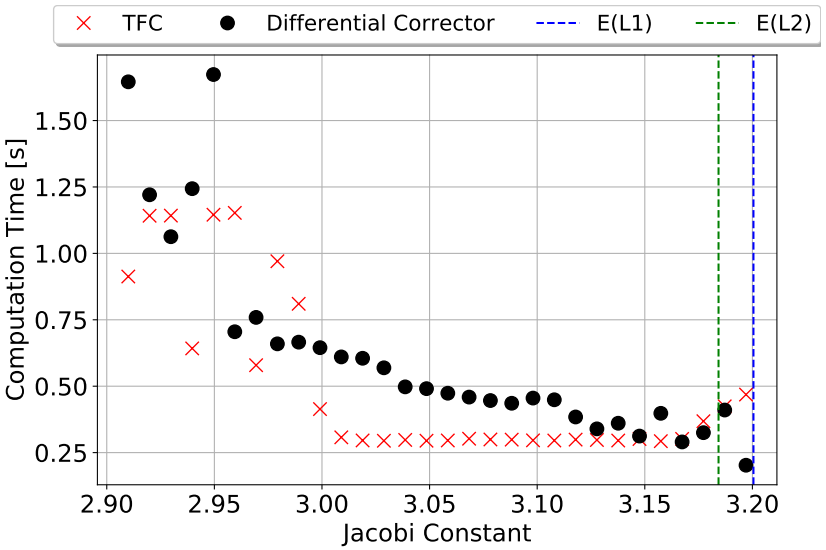


Figure 4.27: Computational time of the the TFC method for the trajectories plotted in Fig. 4.25 compared to that of the differential corrector. The TFC method holds a slight speed gain over the differential corrector.

Similar to the test for the L1 Lagrange point, Figure 4.28 displays the computed trajectories around the L2 Lagrange point. Additionally, like Figures 4.26 and 4.27, the accuracy and computation time for these tests are provided in Figures 4.29 and 4.30.

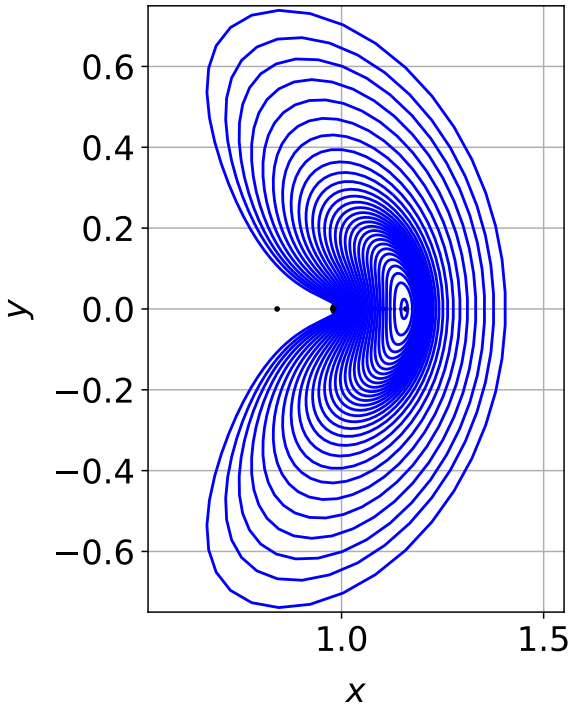


Figure 4.28: Lyapunov orbits for the Earth-Moon system for Jacobi constant values ranging from the energy of L2 to 2.92.

.

In Figure 4.29, the TFC method is more accurate, albeit only slightly. At a Jacobi constant level of about 3.00 and above, the differential corrector method does not converge, as shown by the jump in accuracy. At this Jacobi constant level, the TFC method's accuracy starts to decrease before failing to converge at the Jacobi constant value of 2.92.

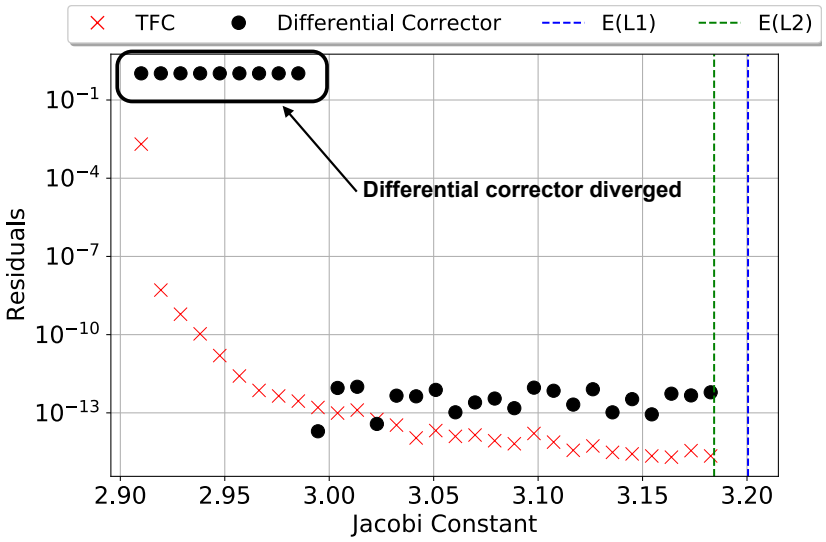


Figure 4.29: Maximum residuals of the loss vector for the TFC method solving for the trajectories plotted in Fig. 4.28 as compared to the differential corrector. For the trajectories around L2, the differential corrector diverged around a Jacobi constant level of 3.00, while the TFC method was able to solve the problem with diminishing accuracy. The black box highlights the diverged cases.

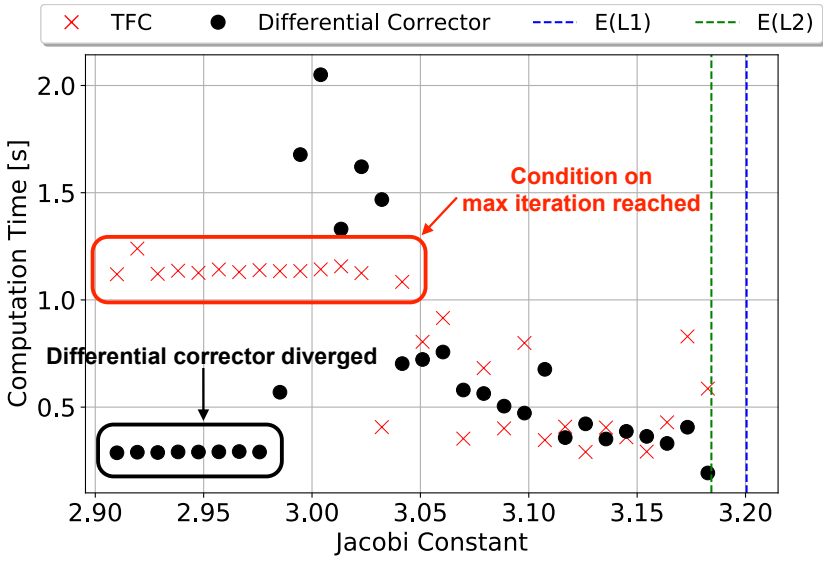


Figure 4.30: Computational time of the TFC method for the trajectories plotted in Fig. 4.28 compared to that of the differential corrector. Again, the black box highlights where the differential corrector diverged. Additionally, the red box shows where the TFC method reached its maximum allowed iterations of 20. These cases are correlated to the reduction of accuracy seen in Fig. 4.29.

Example 4.9: Halo Orbits around L1 & L2 Lagrange points

Next, the proposed technique was utilized to compute Halo orbits around L1 and L2. These orbits differ from Lyapunov orbits because they are *not* restricted to the x-y plane and become three-dimensional. In fact, this family of orbits is a bifurcation of the Lyapunov orbits computed in the previous section and are characterized by "northern" and "southern" bifurcations. However, when using the TFC method to compute these Halo orbits, the only thing that changes is the initialization of the $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and b parameters. First, we look at the computation of the "northern" family of Halo orbits around L1 and L2 as plotted in Figure 4.31.

In these plots, we can see that around the L1 equilibrium point, the method converged to Lyapunov orbits for higher Jacobi constants. However, as the Jacobi constant decreases below 3.025, the method does not converge to a periodic orbit, as shown by

the increase in residuals around the Jacobi constant value of 3.025. In fact, at the other L2 equilibrium point, the method converges for all Jacobi constant values; however, at values below 3.025, the solution jumps to a circular orbit around the Moon—therefore, these orbits were not plotted.

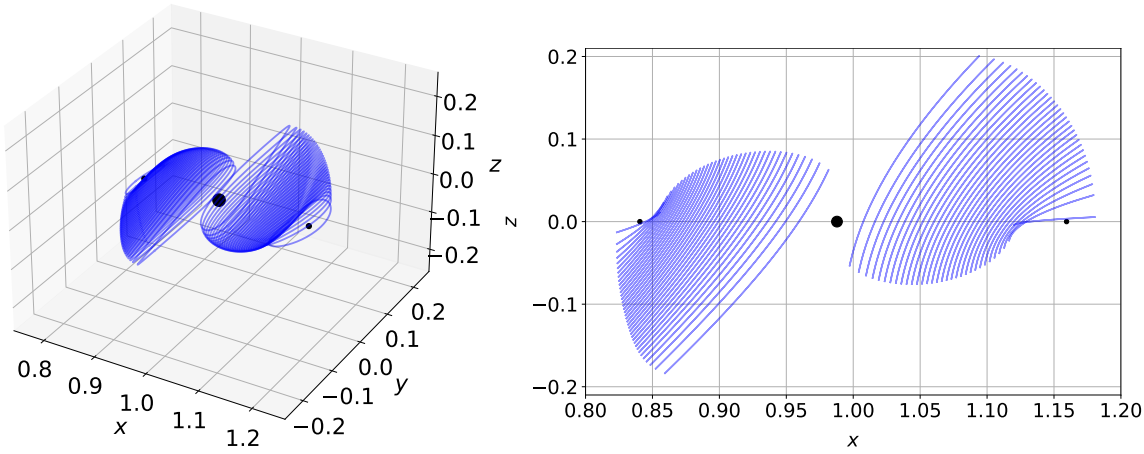


Figure 4.31: Halo orbits of the "northern" bifurcation around both L1 and L2 Lagrange points.

Like the Lyapunov orbit tests, the loss vector's maximum residual was recorded and is plotted in 4.32. All converged solutions were on the order of $\mathcal{O}(10^{-14})$. The convergence for Halo orbits took longer, with some cases taking 8 seconds, while on average, the solution time was around 2 seconds, as shown in Figure 4.33.

Like the "northern" Halo orbits plotted in Figure 4.31, the Halo orbits of the "southern" bifurcation were computed with similar findings and, therefore, omitted from this paper for brevity.

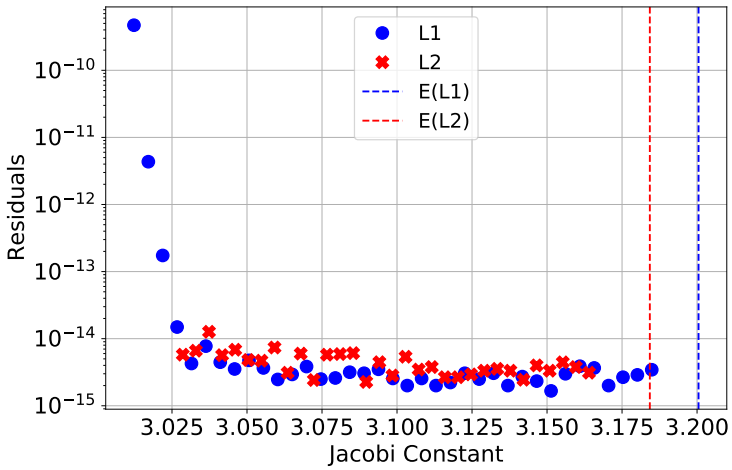


Figure 4.32: Maximum residuals of the loss vector for the TFC method solving for the trajectories plotted in Fig. 4.31. For almost all cases, the solution accuracy is on the order, $\mathcal{O}(10^{-14})$. However, around a Jacobi constant level of 3.025, the accuracy decreases for orbits around L1. The solutions for orbits around L2 lower than 3.025 are not plotted because, while they converged to a valid period orbit with high accuracy, it was not a Halo-type orbit.

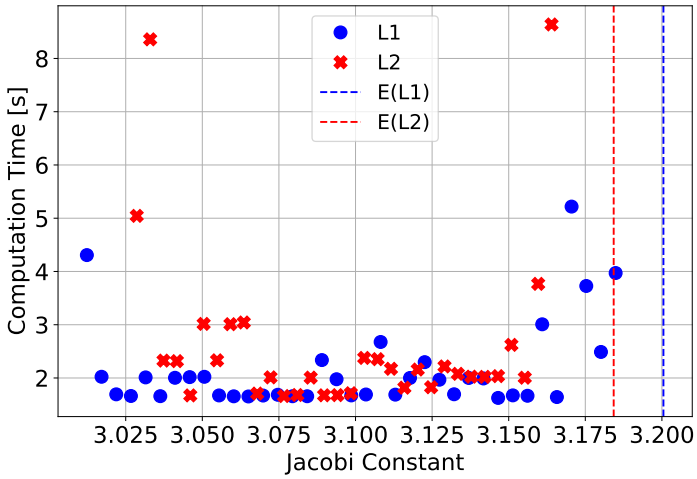


Figure 4.33: Computational time of the TFC method for the solution of "northern" Halo orbits around L1 and L2 plotted in Fig. 4.31. At first glance, it can easily be seen that the computation of these orbits too about twice as long to compute as the Lyapunov orbits. One cause of increased computation time is that the system of equations increased since more points and basis functions were need in the computation of these orbits.

4.10 Over-constrained differential equations

In the following section, we revisit the theory developed in Section 2.6 and apply some of the over-constrained expressions to specific applications. First, a problem considering the interpolation of a trajectory based on noisy measurements augmented by a differential equation is explored. After this, a differential equation is analyzed by solving the continuous transformation from an initial-value problem to a boundary-value problem.

4.10.1 Merging data with dynamics

Consider a scenario where a trajectory is observed multiple times over its path. One question may arise about how this observational data (subject to measurement noise) can be incorporated along with the dynamical model to predict the object's actual path. The following example considers the merging of data with dynamics by using an over-constrained expression.

Example 4.10: Merging data with dynamics

Consider a trajectory governed by the following differential equation,

$$y_{xx} + 2y_x + y = 0,$$

with the analytical solution of the form $y(x) = e^{-x}(c_1 x + c_2)$, which was used to check the final answer and create the true trajectory. Additionally, assume that an object is "observed" under the influence of this dynamical system at three points $x = [-1, -0.5, +1]$, and these measurements are subject to normally distributed noise such that,

$$y(-1) = \mathcal{N}(y_1^{true}, \sigma_1^2), \quad y(-0.5) = \mathcal{N}(y_2^{true}, \sigma_2^2), \quad \text{and} \quad \mathcal{N}(y_3^{true}, \sigma_3^2).$$

To solve this problem, we can utilize Equation (2.26) from Section 2.6.4. Additionally, since the measurement data has associated accuracy in terms of σ_1, σ_2, and σ_3, the weight matrix is defined by the variances such that,

$$W = \begin{bmatrix} \sigma_1^{-2} & 0 & 0 \\ 0 & \sigma_2^{-2} & 0 \\ 0 & 0 & \sigma_3^{-2} \end{bmatrix}.$$

For the given problem, we assume $\sigma_1 = \sigma_2 = \sigma_3 = 1$ and $y_1^{true} = 5$, $y_2^{true} = 4.515$, and $y_3^{true} = 2$. Using the development in Section 2.6.4 this differential equation can incorporate information from all three observations even though the differential equation is only second order. Furthermore, after this step, the process to solve the differential equation is exactly the same as all prior examples and has been omitted for brevity.

For this specific test, a Monte Carlo simulation of $10,000$ trials was conducted to determine the space that the function $y(x)$ could occupy given the "observation" uncertainty and subject to the governing dynamics of the differential equation. Figure 4.34 shows the solution.

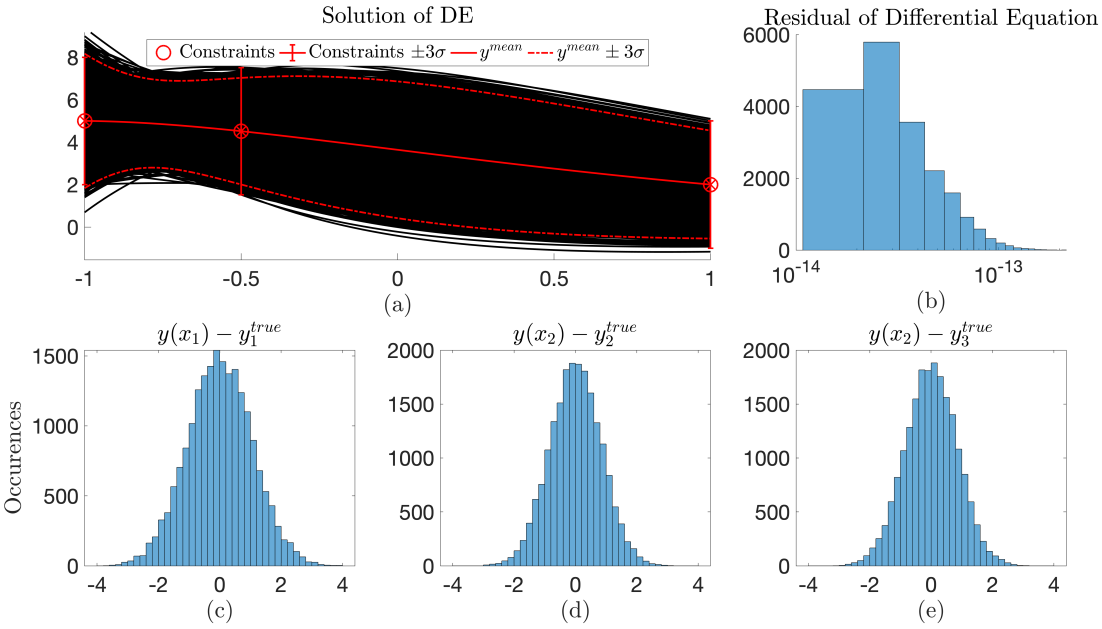


Figure 4.34: Monte Carlo test for 10,000 trials. Plot (a) shows the differential equation solution space given the observation uncertainty. Plot (b) highlights the residuals of the differential equation over the entire simulation. It can be seen that the residuals of all solutions are between 10^{-13} to 10^{-14}. Plots (c), (d), and (e) display the distribution of the constraint points around the true value. Note, these values are sampled from the solutions of the differential equation and not the constraints specified in the constrained expression.

A probability bound for the differential equation can be produced through this test, along with an estimated mean. For all solutions, the differential equation residuals remained less than 10^{-13} verifying the accuracy of the method. Additionally, an interesting result of this test is in the final estimated solutions. This is most evident when observing the solution trajectories near the constraint points, $x = -0.5$ and $x = +1$; it can be seen that the 3σ of the differential equation is less than that of 3σ associated with the constraints. This happens because the loss function in the TFC method minimizes the residuals of the differential equation; in the over-constrained TFC method, the residuals are minimized simultaneously with the weighted least-

squares of the observations (or constraints).

4.10.2 Initial to boundary value problem transformation

The development of the over-constrained expression led to this question: if a differential equation can be solved with more constraints than its order, what is the connection between an initial- and boundary-value problem?

Example 4.11: Initial to boundary value problem transformation

Consider the second-order, linear differential equation given by,

$$y_{xx} + \left[\cos(3x^2) - 3x + 1\right] y_x + \left[6\sin(4x^2) - e^{\cos(3x)}\right] y = 2\frac{[1-\sin(3x)](3x-\pi)}{4-x}$$

subject to the three constraints

$$y(-1) = -2, \qquad y_x(-1) = -2, \qquad \text{and} \qquad y(+1) = 2.$$

By using Equation (2.25) from Section 2.6.4 and defining $s_1(x) = 1$ and $s_2(x) = x$, we can write,

$$W \begin{bmatrix} 1 & x_0 \\ 0 & 1 \\ 1 & x_f \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \end{bmatrix} = W.$$

Next, before solving this system, let us also define the weight matrix as,

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1-\gamma & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

where γ is a weight parameter transforming the problem from IVP to BVP as $\gamma \in [0, 1]$. Now, solving for the system we get the pseudo-switching functions,

$$\varphi_1 = s_1\alpha_{11} + s_2\alpha_{21} = \frac{1}{1+4\gamma-\gamma^2}\Big((1+\gamma) - 2\gamma x\Big)$$
$$\varphi_2 = s_1\alpha_{12} + s_2\alpha_{22} = \frac{1}{1+4\gamma-\gamma^2}\Big((1-\gamma)^2 + (1-\gamma^2)x\Big)$$
$$\varphi_3 = s_1\alpha_{13} + s_2\alpha_{23} = \frac{1}{1+4\gamma-\gamma^2}\Big(-\gamma(\gamma-3) + 2\gamma x\Big)$$

so the total over-constrained expression takes the form,

$$y(x, g(x)) = g(x) + \varphi_1(x)\Big(-2 - g(-1)\Big) + \varphi_2(x)\Big(-2 - g_x(-1)\Big) + \varphi_3(x)\Big(2 - g(1)\Big).$$

Again, by utilizing the numerical techniques discussed earlier, we can define the free function, plug the resulting expression into the differential equation to create our loss function, discretize the domain at the collocation nodes, and solve the system via least-squares. Figure 4.35 shows this transformation "surface" along with the residuals of the differential equation for validation of the method. Figure 4.36 shows that the mean residual over all of the γ values are on the order of 10^{-14} with a standard deviation on the same order.

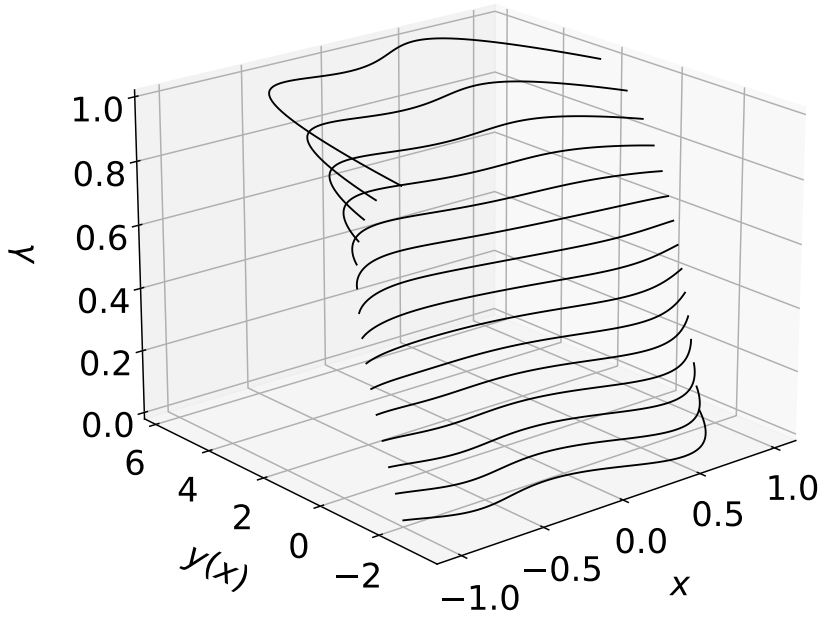


Figure 4.35: IVP to BVP differential equation parametric transformation. These plots shows the solution of the differential equation, $y(x)$, continuously morphing from IVP constraints to BVP constraints.

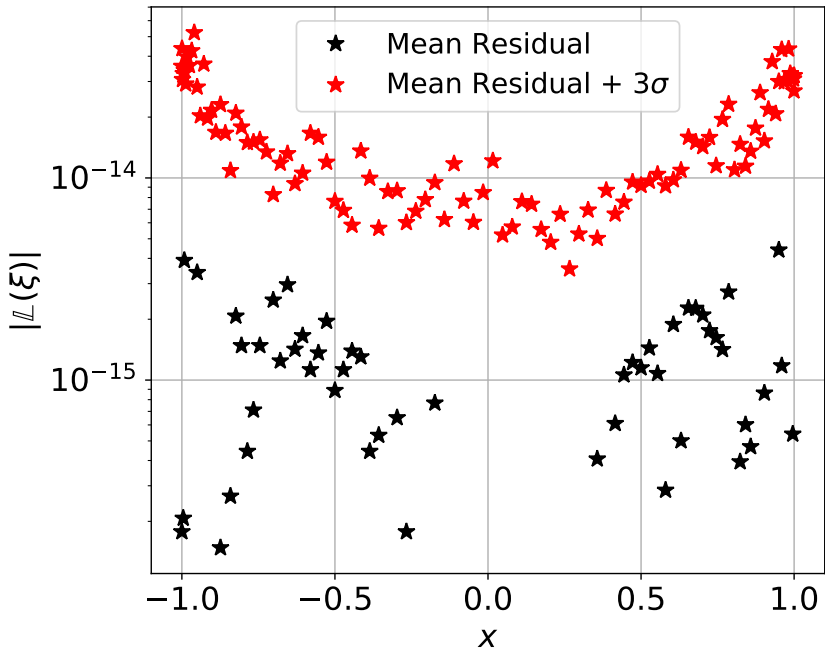


Figure 4.36: Residuals of loss vectors for IVP to BVP differential equation parametric transformation. In all cases, the residual of the differential equation is on the order of 10^{-14}.

Part 3
Optimal Control

Some days feel warm no matter how cold
they are, and some things are fun no
matter how old you are, and sometimes
you wish a visit could just last forever...

— Unravel, *ColdWood Interactive*

5. USE FOR REAL-TIME OPTIMAL CONTROLLERS IN AEROSPACE SYSTEMS

Over the previous sections, we have explored the Theory of Functional Connections to build the constrained expression and solve differential equations subject to constraints. In this section, we will take everything we have learned thus far and explore its application to the field of optimal control, and specifically, real-time optimal control, which is an active field of research. It should be clear from the examples given in Chapter 4 that TFC is an effective method to solve differential equations.

Transitioning from theoretic equations to the physical world, many problems arise affecting the accuracy and robustness of controllers, including unmodelled dynamics and sensor measurement noise, which can result in a deviation from the desired optimal trajectory. Classically, this problem is overcome by deriving a closed-loop controller that tracks the optimal reference trajectory (e.g., Mars Science Laboratory guidance [68]). While the closed-loop controller may be optimal in following the reference trajectory, it will be sub-optimal in the global problem since a disturbance in the state should redefine the full optimal trajectory. Solving for the new optimal solution would involve computing a single-open loop trajectory consisting of the optimal state and optimal control program history. However, as mentioned above, disturbances and measurement noise will cause a deviation from this solution. Therefore, this computation would have to be done during each guidance cycle of the computer allowing for an updated solution based on the state.

The difference between the two methods mentioned above is easily visualized with a simple example provided in Figure 5.1. Consider some optimal control problem where it is desired that an object's trajectory goes from point A to point B subject to some cost function. Over the course of the trajectory, the true path can deviate from the reference trajectory due to such things as unmodelled dynamics, disturbances, etc. In practice, the control for this reference trajectory is followed until the next guidance cycle, signified by the black box in Figure 5.1. At this point, sensors provide some information on the state,

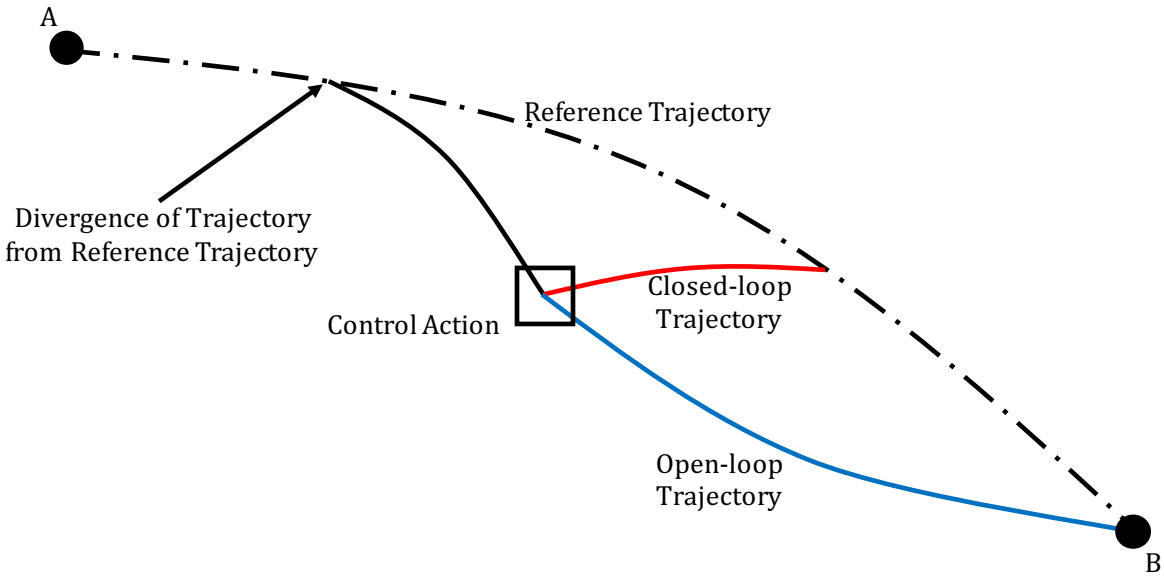


Figure 5.1: Trajectory going from Point A to Point B. The dashed line represents the reference trajectory. In this situation, the true trajectory deviates from the reference trajectory. At the guidance computer cycle, the closed-loop controller acts optimally to return the trajectory (red line) to the reference trajectory. On the other hand, the open-loop solution provides the optimal path from the new point and the resulting trajectory follows this path (blue line).

i.e., position, velocity, etc., and a control action is determined. In the case of a closed-loop control law, the computed control action will be the one that optimally returns the object to the reference trajectory. Conversely, an open-loop control law will recompute a new optimal trajectory from the current state, producing a trajectory that could be drastically different reference trajectory.

Contrary to the example in Figure 5.1, in actual implementation, the frequency of the guidance cycle is drastically higher, and therefore, the control is updated more often. For example, it is reported all guidance functions on the Mars Science Laboratory [68] are within 60 to 70 Hz ($\sim$ 14 to 17 ms). While other applications, this can exceed 100 Hz. Additionally, in the case of the open-loop solution, this implies that a new solution must be computed at this frequency.

Clearly, to enable such technology, real-time solutions must be obtained as quickly as

possible to implement the recomputed optimal trajectory and control. With the exponential increase in computational power, this computation has become more feasible for onboard implementation, and researchers have started to explore the possibility of rapid and real-time trajectory generation for guidance application [69, 70, 71] through open-loop solutions. Additionally, the issues associated with real-time optimal control have also been recently explored in Reference [71]. Overall, the idea is to generate an optimal feedback control that can be constructed by continuously generating computational open-loop optimal trajectories quickly and efficiently [72, 69, 70].

Therefore, with this being the ultimate goal, the following sections will focus on studying the solution of the single open-loop optimal control problems using the TFC approach, where we are interested in determining the limits of the method's speed and accuracy. By developing a fast, accurate, and robust solver, this smaller algorithm can be eventually incorporated into the larger problem, as mentioned above. In the following section, we will discuss the current techniques to solve the open-loop optimal control problem.

5.1 Techniques to solve optimal control problems: direct vs. indirect method

Usually, two methods are available to solve optimal control problems, direct and indirect methods. Direct methods are based on discretizing the continuous states and controls to transform the continuous problem into a nonlinear programming (NLP) problem [73, 74, 75]. The latter can be cast as a finite constrained optimization problem that can be solved via any of the available numerical algorithms that have the potential to find a local minimum, e.g., trust-region method [76]. Whereas direct methods have been applied to solve a large variety of optimal control problems [77, 78, 79, 80], the general NLP problem is considered NP-hard, i.e., non-deterministic polynomial-time hard. NP-hard problems imply that the required amount of computational time needed to find the optimal solution does not have a predetermined bound, i.e., a bound cannot be determined a priori. NP-hard problems are such that the computational time necessary to converge to the solution is not known. As a consequence, the lack of assured convergence may result in questioning the reliability

of the proposed approach. Since for optimal, closed-loop space guidance, most problems require computing numerical solutions onboard and in real-time; general algorithms that solve NLP problems cannot be reliably implemented. More recently, researchers have been experimenting with transforming optimal control problems from a general non-convex formulation into a convex optimization problem [81, 82]. Here, the goal is to take advantage of the assured convex convergence properties. Indeed, convex optimization problems are shown to be computationally tractable as their related numerical algorithms guarantee convergence to a globally optimal solution in polynomial time. The general convex methodology requires that the optimal guidance problem is formulated as convex optimization whenever appropriate or convexification techniques are applied to transform the problem from a non-convex problem into a convex one. Such methodologies have been proposed and applied to solve optimal guidance and control via the direct method in a large variety of problems including, planetary landing [81, 82], entry atmospheric guidance [83, 84], rocket ascent guidance [85], and low thrust [86].

Alternatively, a second approach to solve optimal control and guidance problems, called the indirect method, has been generally applied to various optimal control problems. This approach applies optimal control theory (i.e., Pontryagin Minimum Principle, PMP) to formally derive the first-order necessary conditions that must be satisfied by the optimal solution (state and control). The problem is cast as a two-point boundary value problem (TPBVP) that must be solved to determine the time evolution of state and costate from which the control generally depends. For general nonlinear problems, the necessary conditions result in a complicated set of equations and conditions. Additionally, the resulting TPBVP tends to be highly sensitive to the initial guess on the costates making the problem very hard to solve. Although indirect methods are known to yield more accurate optimal solutions, they are tough to implement and tend to be less used in practice with respect to direct methods. For this problem, we attempt to alleviate the sensitivity of initialization by TFC constrained expressions.

In the next section, we will look at the derivation of the TPBVP from the indirect method, starting with first principles. Additionally, we will explore how the TFC constrained expression reduces the number of algebraic equations to be solved.

5.2 Summary of the indirect method

To thoroughly understand the application of the TFC method to solve optimal control problems, a basic understanding of optimal control theory, and more specifically, the indirect method based on the calculus of variation, is needed. For the reader's convenience, the mathematical foundation for a general optimal control problem is provided in this section. For an extensive look into a plethora of optimal control problem types solved using the indirect method, the reader is directed to "Applied Optimal Control" by Bryson and Ho [87].

In general, a continuous-time dynamical optimization problem can be posed as a minimization of the cost functional (known as the Bolza Problem),

$$J = \Phi\left(\boldsymbol{x}(t_f), t_f\right) + \int_{t_0}^{t_f} \mathcal{L}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) \, \mathrm{d}t \tag{5.1}$$

where $\boldsymbol{x}(t)$ is the state vector and $\boldsymbol{u}(t)$ is the control vector, both a function of the independent variable of time, t. In this formulation, Φ is a function is the cost associated with the terminal state values and $\mathcal{L}$ is cost over the trajectory. In addition to Equation (5.1), the states' dynamics are governed by a general nonlinear equation,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) \tag{5.2}$$

with the boundary constraints

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{5.3}$$

$$\Psi\left(\boldsymbol{x}(t_f), t_f\right) = \boldsymbol{0}.$$

By adjoining the system of differential equations given by Equation (5.2) with the Lagrange multiplier functions $\boldsymbol{\lambda}(t)$, called the costate functions, the augmented cost function becomes,

$$J_a = \Phi(\boldsymbol{x}(t_f), t_f) + \boldsymbol{\nu}^{\mathrm{T}}\Psi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} \Big(\mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{\lambda}^{\mathrm{T}}(t)\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) - \boldsymbol{\lambda}^{\mathrm{T}}(t)\dot{\boldsymbol{x}}\Big)\,\mathrm{d}t \tag{5.4}$$

In optimal control theory, the first two terms in the integral are defined as the scalar function H called the Hamiltonian,

$$H(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t), t) = \mathcal{L}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) + \boldsymbol{\lambda}^{\mathrm{T}}(t)\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{5.5}$$

Substituting Equation (5.5) into Equation (5.4) and dropping the function arguments for clarity yields,

$$J_a = \Phi + \boldsymbol{\nu}^{\mathrm{T}}\Psi + \int_{t_0}^{t_f} \Big(H - \boldsymbol{\lambda}^{\mathrm{T}}\dot{\boldsymbol{x}}\Big)\,\mathrm{d}t.$$

Consider the variation of the augmented cost function J_a about the optimal solution of $J(\boldsymbol{x}^*, \boldsymbol{u}^*, t_f^*)$ where the $(^*)$ signifies the optimal solution,

$$\begin{aligned}\delta J_a = {} & \frac{\partial \Phi}{\partial \boldsymbol{x}^*(t_f^*)}^{\mathrm{T}}\,\mathrm{d}\boldsymbol{x}_f^* + \frac{\partial \Phi}{\partial t_f^*}\,\mathrm{d}t_f + \boldsymbol{\nu}^{\mathrm{T}}\frac{\partial \Psi}{\partial \boldsymbol{x}^*(t_f^*)}\,\mathrm{d}\boldsymbol{x}_f + \boldsymbol{\nu}^{\mathrm{T}}\frac{\partial \Psi}{\partial t_f^*}\,\mathrm{d}t_f + \mathrm{d}\boldsymbol{\nu}^{\mathrm{T}}\Psi(\boldsymbol{x}^*(t_f^*), t_f^*) \\ & + \int_{t_0}^{t_f^*} \Big[\frac{\partial H}{\partial \boldsymbol{x}^*(t)}^{\mathrm{T}}\delta\boldsymbol{x} + \frac{\partial H}{\partial \boldsymbol{u}^*(t)}^{\mathrm{T}}\delta\boldsymbol{u} + \frac{\partial H}{\partial \boldsymbol{\lambda}^*}^{\mathrm{T}}\delta\boldsymbol{\lambda} - \delta\boldsymbol{\lambda}^{\mathrm{T}}\dot{\boldsymbol{x}}^* - \boldsymbol{\lambda}^{*\mathrm{T}}\delta\dot{\boldsymbol{x}}^*\Big]\,\mathrm{d}t \\ & + \Big[H(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, t_f^*) - \boldsymbol{\lambda}^{*\mathrm{T}}(t_f^*)\dot{\boldsymbol{x}}^*(t_f^*)\Big]\,\mathrm{d}t_f.\end{aligned}$$

Collecting terms and rewriting $\boldsymbol{\lambda}^{*\mathrm{T}}\delta\dot{\boldsymbol{x}}^*$ using integration by parts leads to,

$$\begin{aligned}\delta J_a = {} & \Big[\frac{\partial \Phi}{\partial \boldsymbol{x}^*(t_f^*)} + \frac{\partial \Psi}{\partial \boldsymbol{x}^*(t_f^*)}^{\mathrm{T}}\boldsymbol{\nu}\Big]^{\mathrm{T}}\,\mathrm{d}\boldsymbol{x}_f + \Big[\frac{\partial \Phi}{\partial t_f^*} + \boldsymbol{\nu}^{\mathrm{T}}\frac{\partial \Psi}{\partial t_f^*}\Big]\,\mathrm{d}t_f + \mathrm{d}\boldsymbol{\nu}^{\mathrm{T}}\Psi(\boldsymbol{x}^*(t_f^*), t_f^*) \\ & - \boldsymbol{\lambda}^{*\mathrm{T}}\delta\boldsymbol{x}|_{t_0}^{t_f^*} + \int_{t_0}^{t_f^*} \Big\{\Big[\frac{\partial H}{\partial \boldsymbol{x}^*} + \dot{\boldsymbol{\lambda}}^*\Big]^{\mathrm{T}}\delta\boldsymbol{x} + \frac{\partial H}{\partial \boldsymbol{u}^*}^{\mathrm{T}}\delta\boldsymbol{u} + \Big[\frac{\partial H}{\partial \boldsymbol{\lambda}^*} - \dot{\boldsymbol{x}}^*\Big]^{\mathrm{T}}\delta\boldsymbol{\lambda}\Big\}\,\mathrm{d}t \\ & + H(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, t_f^*)\,\mathrm{d}t_f - \boldsymbol{\lambda}^{*\mathrm{T}}(t_f^*)\dot{\boldsymbol{x}}^*(t_f^*)\,\mathrm{d}t_f\end{aligned}$$

Now, to simplify the problem further, we need to consider the admissible variation of the state vector, $\delta\boldsymbol{x}(t)$, shown in Figure 5.2 Here we can define the skew variation, $\mathrm{d}\boldsymbol{x}_f$, as,

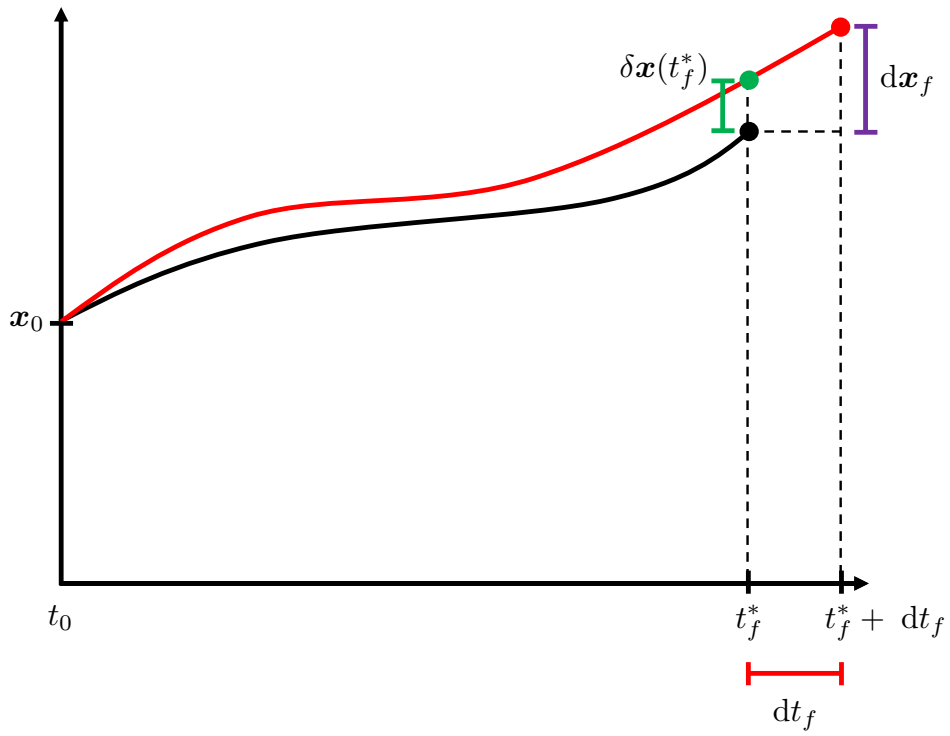


Figure 5.2: Graphical representation of the admissible variation, $\delta\boldsymbol{x}(t_f^*)$, which is the state's variation with respect to the optimal trajectory's (black line) final condition, $\boldsymbol{x}_f^*$.

$$\mathrm{d}\boldsymbol{x}_f = \delta\boldsymbol{x}(t_f^*) + \Big(\dot{\boldsymbol{x}}^*(t_f) + \overbrace{\delta\dot{\boldsymbol{x}}(t_f)}^{\text{neglect second order term}}\Big)\,\mathrm{d}t_f$$

which we simplify to,

$$\mathrm{d}\boldsymbol{x}_f = \delta\boldsymbol{x}(t_f^*) + \dot{\boldsymbol{x}}^*(t_f)\,\mathrm{d}t_f.$$

Using this relationship along with the fact that for $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ the variation of the state at the initial condition is equal to zero and thus $\boldsymbol{\lambda}^{*\mathrm{T}}\delta\boldsymbol{x}\big|_{t_0}^{t_f^*} = \boldsymbol{\lambda}^{*\mathrm{T}}\delta\boldsymbol{x}(t_f^*)$, we can simplify the

expression of δJ_a to,

$$\begin{aligned}\delta J_a =& \Big[\frac{\partial \Phi}{\partial \boldsymbol{x}^*(t_f^*)} + \frac{\partial \Psi}{\partial \boldsymbol{x}^*(t_f^*)}^{\mathrm{T}} \boldsymbol{\nu} - \boldsymbol{\lambda}^*(t_f^*)\Big]^{\mathrm{T}} \mathrm{d}\boldsymbol{x}_f + \Big[\frac{\partial \Phi}{\partial t_f^*} + \boldsymbol{\nu}^{\mathrm{T}} \frac{\partial \Psi}{\partial t_f^*} + H(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, t_f^*)\Big] \mathrm{d}t_f \\ &+ \mathrm{d}\boldsymbol{\nu}^{\mathrm{T}} \Psi(\boldsymbol{x}^*(t_f^*), t_f^*) + \int_{t_0}^{t_f^*} \Big\{ \Big[\frac{\partial H}{\partial \boldsymbol{x}^*} + \dot{\boldsymbol{\lambda}}^*\Big]^{\mathrm{T}} \delta \boldsymbol{x} + \frac{\partial H}{\partial \boldsymbol{u}^*}^{\mathrm{T}} \delta \boldsymbol{u} + \Big[\frac{\partial H}{\partial \boldsymbol{\lambda}^*} - \dot{\boldsymbol{x}}^*\Big]^{\mathrm{T}} \delta \boldsymbol{\lambda} \Big\} \mathrm{d}t.\end{aligned}$$

The extrema of this equation can be found by finding the conditions such that δJ_a is equal to zero. In order for this to occur, the square bracketed terms must go to zero. From this, we are lead to a set of equations that must be satisfied simultaneously that are referred to as the first-order necessary conditions for optimality (since they are based on the first variation of the augmented cost function). Dropping the (*) notation, the optimal solution is defined by the following set of differential and algebraic equations,

$$\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} \tag{5.6}$$

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} \tag{5.7}$$

$$\frac{\partial H}{\partial \boldsymbol{u}} = \mathbf{0} \tag{5.8}$$

$$\Psi(\boldsymbol{x}(t_f), t_f) = \mathbf{0} \tag{5.9}$$

$$\boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} + \frac{\partial \Psi}{\partial \boldsymbol{x}(t_f)}^{\mathrm{T}} \boldsymbol{\nu} \tag{5.10}$$

$$H(t_f) + \frac{\partial \Phi}{\partial t_f} + \boldsymbol{\nu}^{\mathrm{T}} \frac{\partial \Psi}{\partial t_f} = 0 \tag{5.11}$$

By looking at our definition of the Hamiltonian, Equation (5.5), the first necessary condition simply reiterates the dynamics of the system, $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, t)$. Furthermore, Equation (5.3) constrains the initial values, Equations (5.7) is a differential equation governing the costate values, and Equation (5.8) is the necessary condition for the control vector. Finally, Equations (5.9), (5.10), and (5.11) are necessary for the following cases and are sometimes referred to as transversality conditions,

- For constraints on the final state and/or time, Equation (5.9) must be satisfied.

- For the components of $\boldsymbol{x}(t_f)$ that are unconstrained (or free), Equation (5.10) is used to determine the final value of the associated costate, i.e., $\boldsymbol{\lambda}(t_f)$.
- For unconstrained (or free) final time, Equation (5.11) must also be satisfied.

In all problems, Equations (5.6), (5.7), and (5.8) will always be applicable, while Equations (5.9), (5.10), and (5.11) are dictated by the constraints of the final state and time according to the bullet points above.

5.3 Addition of control inequality constraint

It can be seen from the prior section that the first-order necessary conditions derived from the indirect method rely on the formulation of the Hamiltonian, H, based on the cost function (a functional of Φ and $\mathcal{L}$), along with any terminal constraints (Ψ). In many problems, as is the case with the fuel-optimal landing problem explored in Chapter 7, it is necessary to constrain the control by some function. Therefore, consider the constraint,

$$\boldsymbol{C}(\boldsymbol{u}(t), t) \leq \boldsymbol{0},$$

where $\boldsymbol{C}$ is a vector function. The method to apply this constraint is to adjoin the constraint to Equation (5.5),

$$H = \mathcal{L} + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{f} + \boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{C}$$

where $\boldsymbol{\mu}$ are Lagrange multipliers that have the requirement,

$$\mu_i \quad \begin{cases} \leq 0, & C_i = 0, \\ = 0, & C_i < 0 \end{cases}$$

where i denotes the specific constraint. By doing this, the only equation that changes in our prior derivation is Equation (5.8) since $\boldsymbol{C}$ is a function of the control variable. It follows

that,

$$\mathbf{0} = \frac{\partial H}{\partial \boldsymbol{u}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} + \boldsymbol{\lambda}^{\mathrm{T}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} + \boldsymbol{\mu}^{\mathrm{T}} \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{u}}. \tag{5.12}$$

In general, Equation (5.12) defines the set of conditions for the control based on the inequality constraints, $\boldsymbol{C}$, and the state and costate values. We will revisit the application of inequality constraints in Chapter 7.

5.4 Adjustment using the TFC approach and constrained expressions

In general, through the indirect method, the optimal control problem is converted into a two-point boundary-value problem, Equations (5.6) and (5.7), with additional linear and nonlinear constraints, Equations (5.3), (5.8), (5.9), (5.10), and (5.11). In the case of control constraints described in Section 5.3, Equation (5.8) is replaced by Equation (5.12). In all, these equations represent the first-order necessary conditions that must be satisfied simultaneously.

As it should be clear from the development of the TFC approach in Sections 2 and 3, the benefit of this method is the ability to analytically embed linear constraints. Of the necessary conditions, the initial value constraint, Equation (5.3), and any linear terminal constraints, Equation (5.9), can be easily embedded into a constrained expression for the state. To distinguish between the linear and nonlinear components of Ψ, let Ψ be the composition of the linear and nonlinear portions,

$$\Psi = \begin{Bmatrix} \Psi_{\ell} \\ \Psi_{n\ell} \end{Bmatrix} = \mathbf{0}$$

where the linear terms Ψ_{ℓ} are embedded into the state constrained expressions and $\Psi_{n\ell}$ replaces the Ψ term in Equations (5.9), (5.10), and (5.11). Doing this reduces the length of the $\boldsymbol{\nu}$ coefficient vector, and therefore, reduces the search space of the numerical optimization algorithm. However, in most cases, and both landing problems presented in this work, the terminal constraints are all linear, and thus, the Ψ term can be eliminated. The result of

the application of the TFC constrained expression is summarized in the following equations.

$$
\begin{aligned}
\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} \quad &\longrightarrow \quad \dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} \\
\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} \quad &\longrightarrow \quad \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} \\
\frac{\partial H}{\partial \boldsymbol{u}} = \mathbf{0} \quad &\longrightarrow \quad \frac{\partial H}{\partial \boldsymbol{u}} = \mathbf{0} \\
\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \quad &\longrightarrow \quad -- \\
\Psi(\boldsymbol{x}(t_f), t_f) = \mathbf{0} \quad &\longrightarrow \quad -- \\
\boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} + \frac{\partial \Psi}{\partial \boldsymbol{x}(t_f)}^{\mathrm{T}} \boldsymbol{\nu} \quad &\longrightarrow \quad \boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} \\
H(t_f) + \frac{\partial \Phi}{\partial t_f} + \boldsymbol{\nu}^{\mathrm{T}} \frac{\partial \Psi}{\partial t_f} = 0 \quad &\longrightarrow \quad H(t_f) + \frac{\partial \Phi}{\partial t_f} = 0
\end{aligned}
$$

5.5 Connection with the existing literature and difference between local and global collocation methods

Over the past few decades, optimal control and trajectory optimization have been very active and interconnected fields of research. Solving optimal control problems is becoming increasingly important in developing G&C algorithms that can effectively enable system autonomy and autonomous operations. Indeed, the recently coined term *computational guidance and control* [88] refers to a paradigm shift in which computation has a central role in defining and executing G&C functions for aerospace systems. Newly defined algorithms tend to rely extensively on onboard computation, where numerical algorithms replace closed-loop G&C and closed-loop predefined laws. Indeed, the vast majority of optimal control problems of interest for space systems do not have a closed-form solution and must rely on numerical methods. The latter are generally divided into two classes, i.e., direct and indirect methods.

Direct methods, sometimes called to as *direct transcription methods* [89], refer to a class of numerical optimal control methodologies where the continuous optimal control problem is transcribed into an NLP optimization problem via proper approximation of the state and/or control. The most fundamental direct method is the single or multiple shooting method (e.g.,

[90]), where the control is parametrized using a specified functional form, and the equations of motions are satisfied by direct integration. The resulting NLP minimizes the discretized cost function subject to path and/or interior-point constraints.

In contrast, the alternative and more popular class of direct methods is the *direct collocation* method. Here, both state and control are approximated using a defined functional form (e.g., a linear combination of Chebyshev polynomials). Such methods are generally divided into *local and global collocation*. Local collocation divides the interval into many subintervals and enforces continuity across the interfaces. The resulting problem is further discretized using Runge-Kutta (implicit) methods (e.g., References [91, 92]) or orthogonal collocation methods, where the collocation points are selected as roots of a family of orthogonal polynomials (e.g., References [93, 94]). Conversely, global collocation methods employ *global* polynomials to approximate state and control with collocation executed at specified points across the desired time interval.

The most popular set of global collocation methods for optimal control are named *pseudospectral* methods. Indeed, there are different ways to approximate state and control. Historically, the first class of pseudospectral methods were developed by expanding state and control in a set of Chebyshev polynomials of degree N [74, 95]. Eventually, this approach was abandoned in favor of a linear combination of Lagrange polynomials using alternative collocation points such as Gauss-Lobatto [96] and Gauss-Lobatto-Radau [97, 98]. Such formulations were preferred mainly because the isolation condition was automatically satisfied [99] and yielded simpler conditions for collocation.

Many advancements have been made to develop both theory and practical implementation of pseudospectral methods for direct transcription of optimal control problems. Theoretical understanding in the convergence properties and connection with indirect methods [100, 101, 102, 103, 104, 105] coupled with pseudospectral algorithmic advancements to deal with a large class of smooth and non-smooth problems [75, 97, 106, 107, 108] has been paving the way to the potential application of such approaches for real-time implementation

[72, 69, 70]. Importantly, a new class of adaptive pseudospectral methods capable of automatically determining the number of segments and order of polynomial expansion has been recently developed [73]. Such an approach eventually led to the development of the GPOPS-II numerical platform [109], which has been widely employed in trajectory optimization and control in a few applications such as low-thrust [78], solar sail [110], and rocket ascent [79]. An in-depth review of pseudospectral methods applied to optimal control can be found in [99, 111, 112].

On the other end, indirect methods rely on developing the first-order necessary conditions by directly applying PMP or by the calculus of variations. The necessary conditions result in a TPBVP that must be generally resolved by application of numerical techniques such as single and multiple shooting methods [113, 114], orthogonal collocation [115], or pseudospectral methods [116]. The proposed method falls under this category, as the optimal guidance problem is cast as TPBVP that is solved via TFC.

At first glance, the proposed technique might seem similar to some of the above mentioned numerical schemes, namely, collocation methods [54] and indirect pseudospectral methods [116]. This similarity is because the free function $g(t)$ is approximated using orthogonal polynomials discretized over the local or global domain, depending on the selected technique. However, there is a fundamental difference and a numerical benefit that the TFC approach adds, which is absent in previously developed techniques. For example, in indirect orthogonal collocation methods, the state and costates are parameterized using piecewise polynomial functions, transforming the problem into a nonlinear system of equations that must be solved.

Similarly, in indirect pseudospectral methods, the global spectral approach mandates that the state and costate are expanded via some basis functions. While it is true that the function $g(t)$ may be defined in the same fashion, the fundamental difference lies in how the TFC approach handles the problem's constraints: by analytically embedding them through the use of constrained expressions. In both local and global spectral methods, such constraints become part of the optimization scheme. In contrast, the TFC approach analytically reduces

the search space of the solution to those that only satisfy the constraints. As a result, a simpler optimization scheme can be employed to find the solution.

To further highlight the differences, consider the differential equation to be solved in Equation (5.13),

$$F\left(t, y, \dot{y}, \ddot{y}\right) = 0 \qquad \text{subject to:} \quad \begin{cases} y(t_0) = y_0 \\ \dot{y}(t_0) = \dot{y}_0 \\ y(t_f) = y_f \\ \dot{y}(t_f) = \dot{y}_f \end{cases} \tag{5.13}$$

using the spectral method. Let the function $y(t)$ be defined in the same way as the $g(t)$ function in the TFC formulation such that,

$$y(t) = \boldsymbol{\zeta}^{\mathrm{T}} \boldsymbol{h}(z).$$

The key difference is that this description does not satisfy the constraints which must be enforced by the following equations,

$$\begin{aligned} y(t_0) &= y_0 = \boldsymbol{\zeta}^{\mathrm{T}} \boldsymbol{h}(z_0) \\ y(t_f) &= y_f = \boldsymbol{\zeta}^{\mathrm{T}} \boldsymbol{h}(z_f) \\ \dot{y}(t_0) &= \dot{y}_0 = \boldsymbol{\zeta}^{\mathrm{T}} c\boldsymbol{h}_z(z_0) \\ \dot{y}(t_f) &= \dot{y}_f = \boldsymbol{\zeta}^{\mathrm{T}} c\boldsymbol{h}_z(z_f). \end{aligned}$$

Then, to solve the problem, these equations must be appended to the residual of the differential equation,

$$F(t, \boldsymbol{\zeta}) = 0$$

Notice that to solve for the unknown $\boldsymbol{\zeta}$ coefficient vector, all five equations must be solved simultaneously. In other words, the solution of the constraints are now coupled to the

solution of the dynamics, and the coefficients of $\boldsymbol{\zeta}$ contribute to the constraint satisfaction, which will have numerical approximation error. Therefore, $\boldsymbol{\xi}$ from the TFC development is not the same as $\boldsymbol{\zeta}$ defined through the spectral method.

It should now be clear that the major novelty when solving optimal control problems is the analytical constraint satisfaction that reduces the system of equations. Since this technique is applied before numerically approximating the solution using orthogonal polynomials, there is no numerical error associated with enforcing the boundary conditions. Importantly, the constraints and dynamics are decoupled. Additionally, the constraint satisfaction is independent of how $g(t)$ is expressed, and therefore, the proposed formulation allows for a wide range of potential approximation of the free function. It is worth noting that in pseudospectral optimal control, the selection of the weighted interpolating functions is essential for convergence, and such functions are intimately connected with the problem's boundary conditions [97, 117, 111]. The TFC approach decouples the two problems and only relies on the convergence properties of the selected family of functions which approximate $g(t)$.

6. ENERGY-OPTIMAL LANDING

The energy-optimal landing problem is an important step in our study of the TFC method for real-time optimal control. While mathematically simpler than the fuel-optimal landing problem, it provides a real problem for testing the algorithms. In the simplest formulation, the acceleration due to gravity, $\boldsymbol{a}_g$, is considered constant (which is the cause for the terminal descent phase of landing). For this case, a feedback solution can be derived based on the calculated time-to-go function and can be solved for a problem formulated in state-space (as is the following example) [118]. The feedback law is defined as,

$$\boldsymbol{u} = -\frac{6}{t_{go}^2}\boldsymbol{r} - \frac{4}{t_{go}}\boldsymbol{v} - \boldsymbol{a}_g$$

where $\boldsymbol{u}$ is the control acceleration, $\boldsymbol{r}$ and $\boldsymbol{v}$ are the position and velocity states respectively, and time-to-go is t_{go}.

Conversely, another feedback solution exists (although not used in this work) for this problem called Zero-Effort-Miss/Zero-Effort-Velocity (ZEM/ZEV) [119, 120, 121]. In this approach, ZEM is the associated error in the final distance to the landing site if no control action is taken, and ZEV is the error on final velocity again under zero control effort. This formulation collapses to the expression,

$$\boldsymbol{u} = \frac{6}{t_{go}^2}\mathbf{ZEM} - \frac{2}{t_{go}}\mathbf{ZEV}$$

Moving forward, it is important to know that TFC is by nature an open-loop optimal controller since, in practice, the problem would be solved at every computer cycle to update the trajectory. The feedback solution is only valid for a constant gravity vector, $\boldsymbol{a}_g$; however, since the TFC development is general, it can be easily adjusted to solve for any gravitational model.

Although not presented here, the interested reader is directed to the application of this technique to both small and large planetary bodies presented in Reference [122].

6.1 Dynamical model

For the problem of energy-optimal pinpoint landing on large bodies (e.g., the Moon or Mars) the governing system dynamics during the powered descent phase can be modeled as follows,

$$\begin{aligned} \dot{\boldsymbol{r}} &= \boldsymbol{v} \\ \dot{\boldsymbol{v}} &= \boldsymbol{a}_g + \boldsymbol{u}, \end{aligned}$$

where $\boldsymbol{r}$ and $\boldsymbol{v}$ are position and velocity vectors of the lander with respect to the landing site fixed frame. Additionally, $\boldsymbol{u} = \frac{T}{m}$ is associated with the thrust acceleration of the lander and is used to determine the thrust control T for the current spacecraft mass m. The dynamics of the mass state are governed by the equation,

$$\dot{m} = -\alpha T$$

where $\alpha = 1/v_{ex}$, with v_{ex} being the effective exhaust velocity of the rocket engine. However, since the mass dynamics are independent of the spacecraft position and velocity, and the spacecraft acceleration is the control variable, the mass state and, in turn, the thrust value can are determined after the optimal trajectory is computed. Furthermore, acceleration due to gravity, $\boldsymbol{a}_g$, is considered constant since this problem deals with the terminal descent phase. For this problem, the initial and final position and velocity, and initial mass are given:

$$\begin{cases} \boldsymbol{r}(0) = \boldsymbol{r}_0 \\ \boldsymbol{v}(0) = \boldsymbol{v}_0 \end{cases}, \qquad \begin{cases} \boldsymbol{r}(t_f) = \boldsymbol{r}_f \\ \boldsymbol{v}(t_f) = \boldsymbol{v}_f \end{cases}.$$

The objective is to minimize the energy, which can be realized by minimizing the control used while satisfying the problem's dynamics constraints. Therefore, the problem can be posed as,

Optimization problem statement

$$\begin{aligned} \underset{t_f, \boldsymbol{u}}{\text{minimize}} \quad & \Gamma t_f + \frac{1}{2}\int_{t_0}^{t_f} \boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}\, \mathrm{d}\tau \\ \text{subject to} \quad & \dot{\boldsymbol{r}} = \boldsymbol{v}, \quad \dot{\boldsymbol{v}} = \boldsymbol{a}_g + \boldsymbol{u}, \\ & \boldsymbol{r}(0) = \boldsymbol{r}_0, \quad \boldsymbol{v}(0) = \boldsymbol{v}_0, \\ & \boldsymbol{r}(t_f) = \boldsymbol{r}_f, \quad \boldsymbol{v}(t_f) = \boldsymbol{v}_f \end{aligned}$$

where $\Phi(t_f) = \Gamma\, t_f$ is the terminal cost parameter for the final time. Γ is a scalar weight parameter on the final time and represents a trade-off between the minimum-time and minimum-energy problem. For example, if $\Gamma = 0$, we recover the minimum energy cost function.

6.2 First-order necessary conditions

Applying the PMP, the Hamiltonian takes the following form,

$$H = \mathcal{L} + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{f}$$

which can be expanded as,

$$H = \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u} + \boldsymbol{\lambda}_r^{\mathrm{T}}\boldsymbol{v} + \boldsymbol{\lambda}_v^{\mathrm{T}}(\boldsymbol{a}_g + \boldsymbol{u}).$$

Applying the first-order necessary conditions, the optimal control action is realized by,

$$\frac{\partial H}{\partial \boldsymbol{u}} = \boldsymbol{u} + \boldsymbol{\lambda}_v = 0 \quad \longrightarrow \boldsymbol{u} = -\boldsymbol{\lambda}_v.$$

It can be seen that the vector $\boldsymbol{u}$ is opposite of the costate $\boldsymbol{\lambda}_v$, and therefore, we can replace this costate term directly with the control in all following equations. The additional first-order conditions lead to

$$\begin{aligned} \dot{\boldsymbol{r}} &= \frac{\partial H}{\partial \boldsymbol{\lambda}_r} = \boldsymbol{v} & \qquad \dot{\boldsymbol{\lambda}}_r &= -\frac{\partial H}{\partial \boldsymbol{r}} = \boldsymbol{0} \\ \dot{\boldsymbol{v}} &= \frac{\partial H}{\partial \boldsymbol{\lambda}_v} = \boldsymbol{a}_g + \boldsymbol{u} & \qquad \dot{\boldsymbol{\lambda}}_v &= -\frac{\partial H}{\partial \boldsymbol{v}} = -\boldsymbol{\lambda}_r \end{aligned}$$

and as mentioned, the differential equation associated with $\dot{\boldsymbol{\lambda}}_v$ can be written as,

$$\dot{\boldsymbol{u}} = \boldsymbol{\lambda}_r.$$

Lastly, since the problem is posed as a free final time problem, the transversality condition is given by,

$$H(t_f) + \frac{\partial \Phi}{\partial t_f} = 0$$

which reduces to

$$H(t_f) = -\Gamma.$$

Collecting all equations, a constrained, differential systems of equations is formed which must be satisfied simultaneous to obtain an optimal solution,

First-order necessary conditions

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{6.1}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a}_g + \boldsymbol{u} \tag{6.2}$$

$$\dot{\boldsymbol{\lambda}}_r = \boldsymbol{0} \tag{6.3}$$

$$\dot{\boldsymbol{u}} = \boldsymbol{\lambda}_r \tag{6.4}$$

$$H(t_f) + \Gamma = 0 \tag{6.5}$$

subject to the constraints $\boldsymbol{r}(t_0) = \boldsymbol{r}_0$, $\boldsymbol{v}(t_0) = \boldsymbol{v}_0$, $\boldsymbol{r}(t_f) = \boldsymbol{r}_f$, and $\boldsymbol{v}(t_f) = \boldsymbol{v}_f$.

The following section reformulates the system of equations defined by Equations (6.1), (6.2), (6.3), (6.4), (6.5) using the techniques developed in the prior sections.

6.3 Solving the problem via the TFC

Through the use of TFC, Equations (6.1) through (6.5) can be reduced. First, using the TFC approach, Equation (6.1) is redundant, since the constrained expression will always satisfy this condition. Furthermore, Equations (6.3) and (6.4) can be combined since Equation (6.3) shows $\boldsymbol{\lambda}_r$ must be constant. Therefore, these two equations can be replaced by the equation,

$$u_i(t, \boldsymbol{\xi}_{u_i}) = \boldsymbol{h}_u^{\mathrm{T}} \boldsymbol{\xi}_{u_i}, \quad \text{for} \quad i = 1, 2, 3 \tag{6.6}$$

where $\boldsymbol{h}_u$ consists of the constant and linear terms of the selected basis set. Lastly, the boundary constraints are fully handled by the TFC constrained expressions of the following form,

$$\begin{aligned} r_i(t, \boldsymbol{\xi}_i) = \Big(\boldsymbol{h}(z) - \phi_1(t)\boldsymbol{h}(z_0) - \phi_2(t)\boldsymbol{h}(z_f) - \phi_3(t)c\boldsymbol{h}_z(z_0) - \phi_4(t)c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}} \boldsymbol{\xi}_i \\ + \phi_1(t)r_{0_i} + \phi_2(t)r_{f_i} + \phi_3(t)v_{0_i} + \phi_4(t)v_{f_i}. \end{aligned} \tag{6.7}$$

Therefore, the first-order necessary conditions reduce to,

$$\dot{v}_i = a_{g_i} + u_i \tag{6.8}$$

$$0 = -\frac{1}{2}u_i^2(t_f) + u_i(t_f)a_{g_i} + \Gamma, \tag{6.9}$$

where the state and control are written in terms of the TFC constrained expressions. In general, the unknowns of this system are the coefficients related to the state $\boldsymbol{\xi}_i$ and control $\boldsymbol{\xi}_u$ along with the final time t_f. Both state and control unknowns appear linearly in the

system of equations; however, the final time appears nonlinearly through the transversality equation, Equation (6.9), and can be handled in two different ways. The first method uses an Outer-loop optimizer that solves for the mapping parameter, i.e., optimizes the final time with the transversality condition. In contrast, the inner TFC loop solves the least-squares problem of Equation (6.8). The second method leverages the theory developed in Section 4.8.2, where the mapping parameter (which is a function of t_f) is solved alongside the other unknowns in a single loop. This method, however, requires an implementation of a nonlinear least-squares approach. While this section has merely summarized the relevant equations, Sections 6.3.1 and 6.3.2 discuss in detail how each method can be applied to the energy-optimal landing problem. Lastly, various tests are conducted to determine the accuracy, speed, and robustness of both techniques. The findings of these tests will help us in our study of the more complex problem of fuel-optimal landing in the following chapter.

6.3.1 Outer-loop optimizer

Using the constrained expression given by Equation (6.7), for the Outer-loop method, the constrained expression is written in the problem domain (i.e., in terms of time), and thus, the switching functions are,

$$
\begin{aligned}
\phi_1(t) &= \frac{1}{\Delta t^3}\Big(- t_f^2(3t_0 - t_f) + 6t_0 t_f t - 3(t_0 + t_f)t^2 + 2t^3\Big)\\
\phi_2(t) &= \frac{1}{\Delta t^3}\Big(- t_0^2(t_0 - 3t_f) - 6t_0 t_f t + 3(t_0 + t_f)t^2 - 2t^3\Big)\\
\phi_3(t) &= \frac{1}{\Delta t^2}\Big(- t_0 t_f^2 + t_f(2t_0 + t_f)t - (t_0 + 2t_f)t^2 + t^3\Big)\\
\phi_4(t) &= \frac{1}{\Delta t^2}\Big(- t_0^2 t_f + t_0(t_0 + 2t_f)t - (2t_0 + t_f)t^2 + t^3\Big)
\end{aligned}
$$

Equation (6.7) and its derivatives for this method are,

$$
\begin{aligned}
r_i(t, \boldsymbol{\xi}_i) = \Big(\boldsymbol{h}(z) - \phi_1\boldsymbol{h}(z_0) - \phi_2\boldsymbol{h}(z_f) - \phi_3 c\boldsymbol{h}_z(z_0) - \phi_4 c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\boldsymbol{\xi}_i\\
+ \phi_1 r_{0_i} + \phi_2 r_{f_i} + \phi_3 v_{0_i} + \phi_4 v_{f_i}. \qquad (6.10)
\end{aligned}
$$

Substituting Equation (6.10) and its second derivative, i.e. acceleration, and the definition of the control, Equation (6.6), into Equation (6.8), the loss functions becomes,

$$F_i(t,\Xi) = \left(c^2\boldsymbol{h}_{zz}(z) - \ddot{\phi}_1(t)\boldsymbol{h}(z_0) - \ddot{\phi}_2(t)\boldsymbol{h}(z_f) - \ddot{\phi}_3(t)\boldsymbol{h}_z(z_0) - \ddot{\phi}_4(t)\boldsymbol{h}_z(z_f)\right)^{\mathrm{T}}\boldsymbol{\xi}_i + \ddot{\phi}_1(t)r_{0_i} + \ddot{\phi}_2(t)r_{f_i} + \ddot{\phi}_3(t)v_{0_i} + \ddot{\phi}_4(t)v_{f_i} - a_{g_i} - \boldsymbol{h}_u^{\mathrm{T}}\boldsymbol{\xi}_{u_i} = 0 \quad (6.11)$$

where the loss vector becomes,

$$\mathbb{L} = \begin{Bmatrix}\mathbb{L}_1^{\mathrm{T}} & \mathbb{L}_2^{\mathrm{T}} & \mathbb{L}_3^{\mathrm{T}}\end{Bmatrix}^{\mathrm{T}}_{3N\times 1},$$

where

$$\mathbb{L}_i = \begin{Bmatrix}F_i(t_0,\Xi) & \dots & F_i(t_k,\Xi) & \dots & F_i(t_f,\Xi)\end{Bmatrix}^{\mathrm{T}},$$

and the unknown vector is then,

$$\Xi = \begin{Bmatrix}\boldsymbol{\xi}_1^{\mathrm{T}} & \boldsymbol{\xi}_2^{\mathrm{T}} & \boldsymbol{\xi}_3^{\mathrm{T}} & \boldsymbol{\xi}_{u_1}^{\mathrm{T}} & \boldsymbol{\xi}_{u_2}^{\mathrm{T}} & \boldsymbol{\xi}_{u_3}^{\mathrm{T}}\end{Bmatrix}^{\mathrm{T}}_{(3m+6)\times 1}.$$

It should be seen that the loss function, given by Equation (6.11), is linear, and therefore the loss vector is a linear system of equations. The terms of this linear system are provided in Appendix D.3. Additionally, given this linear system, any available least-squares technique can be used to solve for the unknown coefficients. Next, once these coefficients are solved, Equation (6.9) is enforced using any available root solving technique (the numerical results used `NumPy's` $fsolve()$ algorithm). This process is repeated until the tolerance on the inner and outer residuals are met.

6.3.2 Single-loop approach

For the single-loop approach using TFC, we take advantage of the fact that the mapping coefficient, of Equation (4.9), is a function of the final time t_f. Next, the parameter is redefined such that it cannot be negative: $b^2 := c$. Then, by converting the dynamics and

constraints into the basis function domain, i.e., $z \in [-1, 1]$ for Chebyshev and Legendre polynomials, this parameter can simply be included in the optimization loop and solved simultaneously with the $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_{u_i}$ coefficients. However, in all cases, an unknown final time will appear nonlinearly, and therefore, a nonlinear least-squares will be required regardless of whether or not the original system is linear.

The first step in this method is to write the whole problem in the basis function domain. This in turn will introduce new switching functions (and for clarity will be labeled as ${}^{z}\phi$), which are,

$$
\begin{aligned}
{}^{z}\phi_1(z) &= \frac{1}{\Delta z^3}\Big(- z_f^2(3z_0 - z_f) + 6z_0 z_f z - 3(z_0 + z_f)z^2 + 2z^3\Big) \\
{}^{z}\phi_2(z) &= \frac{1}{\Delta z^3}\Big(- z_0^2(z_0 - 3z_f) - 6z_0 z_f z + 3(z_0 + z_f)z^2 - 2z^3\Big) \\
{}^{z}\phi_3(z) &= \frac{1}{\Delta z^2}\Big(- z_0 z_f^2 + z_f(2z_0 + z_f)z - (z_0 + 2z_f)z^2 + z^3\Big) \\
{}^{z}\phi_4(z) &= \frac{1}{\Delta z^2}\Big(- z_0^2 z_f + z_0(z_0 + 2z_f)z - (2z_0 + z_f)z^2 + z^3\Big),
\end{aligned}
$$

such that $\Delta z := z_f - z_0$. This change is also reflected in the constrained expression for the state,

$$
\begin{aligned}
r_i(z, \boldsymbol{\xi}) = \Big(\boldsymbol{h}(z) - {}^{z}\phi_1 \boldsymbol{h}(z_0) - {}^{z}\phi_2 \boldsymbol{h}(z_f) - {}^{z}\phi_3 \boldsymbol{h}_z(z_0) - {}^{z}\phi_4 \boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}} \boldsymbol{\xi}_i \\
+ {}^{z}\phi_1 r_{0_i} + {}^{z}\phi_2 r_{f_i} + {}^{z}\phi_3 \frac{v_{0_i}}{b^2} + {}^{z}\phi_4 \frac{v_{f_i}}{b^2}. \qquad (6.12)
\end{aligned}
$$

Hence, the need to divide the velocity constraints by the modified mapping parameter, b in Equations (6.12). Next, our definition of $\boldsymbol{u}$ remains unchanged and is defined by Equation (6.6). Following the current definition of the state and costate, the differential equation of Equation (6.8) becomes,

$$
\begin{aligned}
F_i(z, \Xi) = b^4\Big[\Big(\boldsymbol{h}_{zz}(z) - {}^{z}\phi_{1_{zz}} \boldsymbol{h}(z_0) - {}^{z}\phi_{2_{zz}} \boldsymbol{h}(z_f) - {}^{z}\phi_{3_{zz}} \boldsymbol{h}_z(z_0) - {}^{z}\phi_{4_{zz}} \boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}} \boldsymbol{\xi}_i \\
+ {}^{z}\phi_{1_{zz}} r_{0_i} + {}^{z}\phi_{2_{zz}} r_{f_i} + {}^{z}\phi_{3_{zz}} \frac{v_{0_i}}{b^2} + {}^{z}\phi_{4_{zz}} \frac{v_{f_i}}{b^2}\Big] - a_{g_i} - \boldsymbol{h}_u^{\mathrm{T}} \boldsymbol{\xi}_{u_i} = 0
\end{aligned}
$$

and the loss function associated with Equation (6.9) can be written in terms of the unknowns as,

$$\mathbb{L}_H(\Xi) = -\frac{1}{2}\sum_{j=1}^{3}\left(u_j^2(z_f)\right) + \sum_{j=1}^{3}\left(u_j(z_f)a_{g_j}\right) + \Gamma = 0$$

with the augmented loss function

$$\mathbb{L} = \left\{\mathbb{L}_1^{\mathrm{T}} \quad \mathbb{L}_2^{\mathrm{T}} \quad \mathbb{L}_3^{\mathrm{T}} \quad \mathbb{L}_H\right\}^{\mathrm{T}}_{(3N+1)\times 1}$$

where

$$\mathbb{L}_i = \left\{F_i(z_0,\Xi) \quad \dots \quad F_i(z_k,\Xi) \quad \dots \quad F_i(z_f,\Xi)\right\}^{\mathrm{T}}.$$

The unknown vector is then,

$$\Xi = \left\{\boldsymbol{\xi}_1^{\mathrm{T}} \quad \boldsymbol{\xi}_2^{\mathrm{T}} \quad \boldsymbol{\xi}_3^{\mathrm{T}} \quad \boldsymbol{\xi}_{u_1}^{\mathrm{T}} \quad \boldsymbol{\xi}_{u_2}^{\mathrm{T}} \quad \boldsymbol{\xi}_{u_3}^{\mathrm{T}} \quad b\right\}^{\mathrm{T}}_{(3m+7)\times 1}$$

The partial derivatives of the loss functions are provided in Appendix D.4, and nonlinear least-squares is used to update the unknowns.

6.4 Parameter initialization

Finally, the last consideration before solving the problem using either method is to initialize the unknown parameters. In the Outer-loop method detailed in Section 6.3.1, the inner-loop is a linear system, and therefore, $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_{u_i}$ do not need to be initialized. However, an estimate of the final time t_f is needed: for all numerical tests, this value was chosen to be one in the scaled time.

Next, for the single-loop method, all variables must be initialized since the system is nonlinear. As observed in the earlier section, the simplest initialization of the unknowns associated with the state constrained expression is to set them equal to zero. This is equivalent to connecting the boundary value problem with the simplest interpolating expression (i.e,.

$g(x) = 0$).

Although the initialization scheme for the unknowns associated with the control expression could follow the same process, more is known about their potential solution, which can be leveraged. Following this thought, we can initialize the parameters assuming the initial control is opposite the spacecraft velocity,

$$\boldsymbol{u}_0 = -\frac{\boldsymbol{v}_0}{||\boldsymbol{v}_0||}$$

and the final control value is assumed to be in the direction opposite of the initial position vector,

$$\boldsymbol{u}_f = -\frac{\boldsymbol{r}_0}{||\boldsymbol{r}_0||}$$

Using these two equations, the values of $\boldsymbol{\xi}_{u_i} = \left\{ a_{0_i} \quad a_{1_i} \right\}^{\mathrm{T}}$ become,

$$\begin{aligned} a_{0_i} - a_{1_i} &= -\frac{v_{0_i}}{\left(\sum_{j=1}^{3} v_{0_i}^2\right)^{1/2}} = -\mathbb{V}_{0_i} \\ a_{0_i} + a_{1_i} &= -\frac{r_{0_i}}{\left(\sum_{j=1}^{3} r_{0_i}^2\right)^{1/2}} = -\mathbb{R}_{0_i}. \end{aligned}$$

Solving this linear system yields,

$$\begin{aligned} a_{0_i} &= -\frac{1}{2}(\mathbb{R}_{0_i} + \mathbb{V}_{0_i}) \\ a_{1_i} &= -\frac{1}{2}(\mathbb{R}_{0_i} - \mathbb{V}_{0_i}). \end{aligned}$$

6.5 Results

First, the two proposed methods are compared to the known feedback solution presented in Reference [118] to validate the TFC method's accuracy. After this, a Monte Carlo simulation is constructed to test the Single-loop and Outer-loop method over a range of initial conditions to determine expected speed and robustness.

For the numerical test presented in this section, the problem was scaled by the initial conditions. The unit length, ℓ^*, and unit time, t^*, where calculated by the following equations,

$$\ell^* = \max\left(|\boldsymbol{r}_0|\right)$$
$$t^* = \frac{\ell^*}{\max\left(|\boldsymbol{v}_0|\right)}.$$

Example 6.1: Comparison to known feedback solution

Table 6.1: Problem parameters for numerical test.

(a) Problem specific values.

Variable	Value
$\boldsymbol{r}_0$	$\{500000, 100000, 50000\}^{\mathrm{T}}$ [ft]
$\boldsymbol{v}_0$	$\{-3000, 0, 0\}^{\mathrm{T}}$ [ft/s]
$\boldsymbol{a}_g$	$\{0, 0, -5.31\}^{\mathrm{T}}$ [ft/s^2]
Γ	0 and 100

(b) TFC parameters.

Variable	Value
N [# points]	100
m [# basis functions]	60
ε [tolerance]	2.22×10^{-16}

This comparison test shows that both TFC based methods solve the problem with almost identical results to the feedback solution. Furthermore, the TFC based method produces identical results regardless of whether the Outer-loop or Single-loop method is used. The tingle-loop method is an order of magnitude faster. Compared to the spectral method, for these specific test cases, TFC is slightly slower with regard to computation time. Looking at Tables 6.2 and 6.3, the difference is close to 10 milliseconds.

Table 6.2: Single case energy-optimal landing for $\Gamma = 0$.

Parameter	TFC		Spectral		Feedback
Method	Outer-loop	Single-loop	Outer-loop	Single-loop	–
t_f [sec]	406.03	406.03	406.03	406.04	406.03
Cost	19004.12	19004.12	19004.12	19004.12	19004.19
Comp. Time [s]	2.63	0.097	2.00	0.087	–
Iterations	19	15	19	11	–
$\max \lvert\mathbb{L}\rvert$	3.3×10^{-16}	2.2×10^{-16}	4.2×10^{-16}	4.3×10^{-16}	–

However, this test did not provide the full picture and was used as a first test to compare all of the results with those published in Reference [118]. Regardless, this test shows that the Single-loop approach should be the focus of further testing where TFC and the spectral method are used to solve the problem over a wide range of initial conditions.

Table 6.3: Single case energy-optimal landing for $\Gamma = 100$.

Parameter	TFC		Spectral		Feedback
Method	Outer-loop	Single-loop	Outer-loop	Single-loop	–
t_f [sec]	301.05	301.05	301.05	301.05	301.05
Cost	52569.32	52569.32	52569.32	52569.32	52568.53
Comp. Time [s]	2.64	0.124	1.92	0.111	–
Iterations	16	18	16	14	–
$\max \lvert\mathbb{L}\rvert$	4.4×10^{-16}	2.8×10^{-16}	4.4×10^{-16}	3.3×10^{-16}	–

Example 6.2: Monte Carlos simulation for varying initial conditions

Test Setup: For this test, the TFC parameters and acceleration due to gravity remained the same as the example above. Furthermore, for the following test, we have only considered the pure energy-optimal problem such that $\Gamma = 0$.

Next, to span a large variety of initial conditions, the following process was used to define an initial position ellipse and associated velocity. Recall, the equation of the radius of an ellipse is defined as,

$$R_{\text{ellipse}} = \frac{ab}{\sqrt{a^2 \sin^2(\alpha) + b^2 \cos^2(\alpha)}}$$

In our case, we define these parameters,

$$\alpha = \mathcal{U}(0, 2\pi), \quad a = 1000\,[\text{m}], \quad b = 500\,[\text{m}].$$

Using this, the sample ellipse was centered $2,000$ meters up-range and with an elevation of $1,500$ meters. This is simply the point $(-2000, 0, 1500)$. Finally, our initial conditions can be written using the following equations, where $\text{SF} = \mathcal{U}(0, 1)$ is a scale factor used to span the whole area of the ellipse.

$$\boldsymbol{r}_0 = \begin{cases} -2,000 + \text{SF} \cdot R_{\text{ellipse}} \cos(\alpha) \\ \text{SF} \cdot R_{\text{ellipse}} \sin(\alpha) \\ 1,500 + \mathcal{U}(-100, 100) \end{cases} \quad [\text{m}] \quad \boldsymbol{v}_0 = \begin{cases} 100 \cos(\beta) \\ 100 \sin(\beta) \\ -75 + \mathcal{U}(-10, 10) \end{cases} \quad [\text{m/s}]$$

where $\beta = \mathcal{U}\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. The following results compare the accuracy, speed, and robustness of the Single-loop approach for both TFC and spectral method. Recall, the Single-loop method solves all first-order necessary conditions simultaneously, albeit forcing the method to become nonlinear.

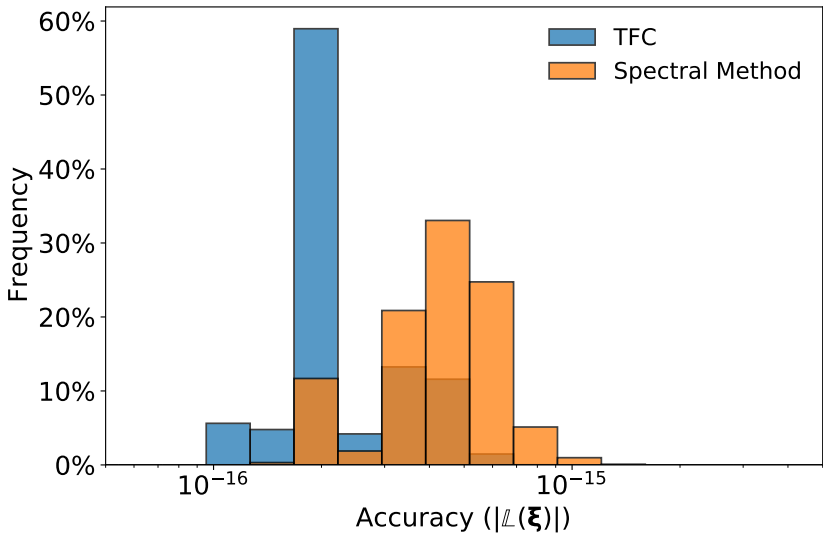


Figure 6.1: Histogram of the maximum residual of the loss vector.

Over the 10,000 trial Monte Carlo simulation, the TFC method failed on three accounts or 0.03% of the time, and the spectral method failed 279 times or 2.79% of the time. Figure 6.1 is a histogram of the methods' error, which shows that TFC is consistently more accurate.

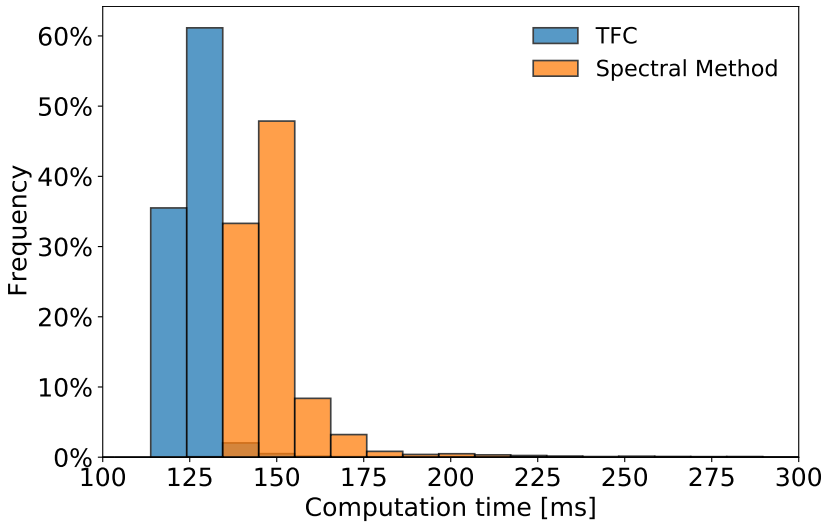


Figure 6.2: Histogram of the computation time of both methods.

Next, Figure 6.2 quantifies the computation time associated with both approaches. It can be seen that TFC produces a solution about 25-50 ms faster than the spectral method. This observation makes sense when analyzing Figure 6.3, where it can be seen that TFC usually takes between 18-20 iterations, while the spectral method can take up to 30 iterations.

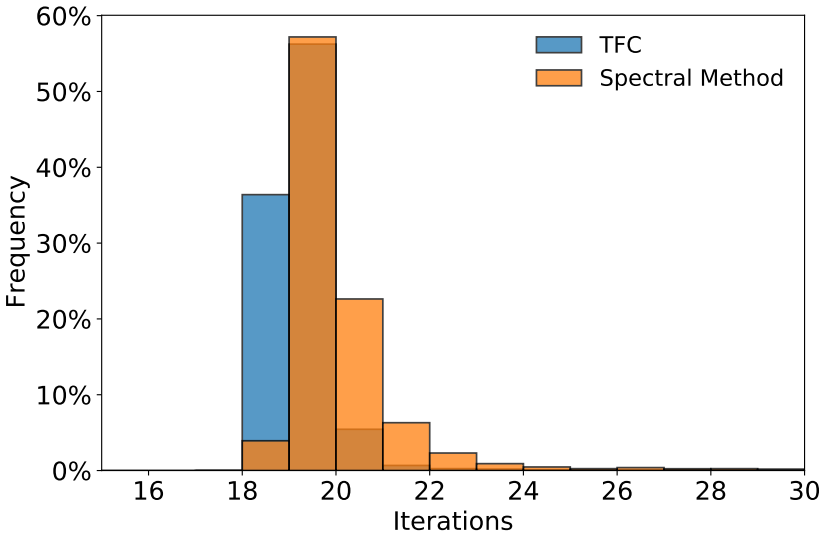


Figure 6.3: Histogram of the number of iterations.

6.6 Conclusions

In this section, we applied TFC to the 3D energy-optimal landing problem, which has a known feedback solution for constant acceleration due to gravity $\boldsymbol{a}_g$. While the TFC algorithm's implementation is relatively straightforward due to the simplicity of the optimal control problem, it gives us a major stepping stone forward in quantifying the accuracy, robustness, and speed of the TFC technique to solve realistic optimization problems. Moving forward, we will leverage what was learned from this example to make decisions in the fuel-optimal landing problem in the following section. The major takeaways from this problem are:

Major takeaways from energy-optimal landing tests

1. The Single-loop TFC approach requires the mapping parameter to show up non-linearly in the dynamics. This can cause sensitivity due to initialization, which reduces the algorithm's robustness.

2. The Outer-loop approach allows for any numerical scheme to be paired with TFC, which increases the applicability and, as seen in the prior example, can lead to increased robustness.

7. FUEL-OPTIMAL LANDING*

The fuel-optimal (or propellant-efficient) landing is the natural extension from our solution of the energy-optimal landing problem presented in Chapter 6. This problem now introduces the mass state as another dynamic equation and inequality constraints on the spacecraft's thrust. While ultimately, we are interested in the full six-degree-of-freedom (6-DOF) solution, this 3-DOF is the natural next step where the attitude dynamics are not considered. This problem's solution is the subject of many studies, as mentioned in the literature review presented at the beginning of Chapter 5. Of the techniques discussed, Lu [123] has looked to solve this problem using the indirect method, which reduces the problem to a shooting method, and Acikmese and Ploen [81] and Blackmore et al. [82] have reformulated the problem via convex optimization to derive a solution.

7.1 Dynamical model

For the problem of powered descent pinpoint landing guidance on large bodies (e.g., the Moon or Mars) the governing system dynamics during the powered descent phase can be modeled as follows,

$$\begin{aligned} \dot{\boldsymbol{r}} &= \boldsymbol{v} \\ \dot{\boldsymbol{v}} &= \boldsymbol{a}_g + \frac{\boldsymbol{T}}{m} \\ \dot{m} &= -\alpha T \end{aligned} \tag{7.1}$$

where the spacecraft's state is defined by the position $\boldsymbol{r}$, velocity $\boldsymbol{v}$, and mass m. Additionally, $\alpha = 1/v_{ex}$, where v_{ex} is the effective exhaust velocity of the rocket engine that is considered

*Reprinted (along with revisions and updates unique to this dissertation) by permission from Springer Nature Customer Service Centre GmbH: Springer Nature The Journal of the Astronautical Sciences "Fuel-Efficient Powered Descent Guidance on Large Planetary Bodies via Theory of Functional Connections," Johnston, H., Schiassi, E., Furfaro, R. et al., 2020, J Astronaut Sci 67, 15211552, Copyright 2020, [2]

constant [81, 123], $T = ||\boldsymbol{T}||$, and $\boldsymbol{T} = T\,\hat{\boldsymbol{t}}$ is the thrust and it is constrained as follows:

$$0 \leq T_{min} \leq T \leq T_{max}$$

$$||\hat{\boldsymbol{t}}|| = 1.$$

Furthermore, $\boldsymbol{a}_g$ is the gravity acceleration, which is also considered constant. As stated in Reference [123], this assumption is justified for short flights, as is the case for the landing's powered descent phase. A summary of the reference frame for this problem is given in Figure 7.1. For the landing problem, the boundary consists of initial and final constraints on the

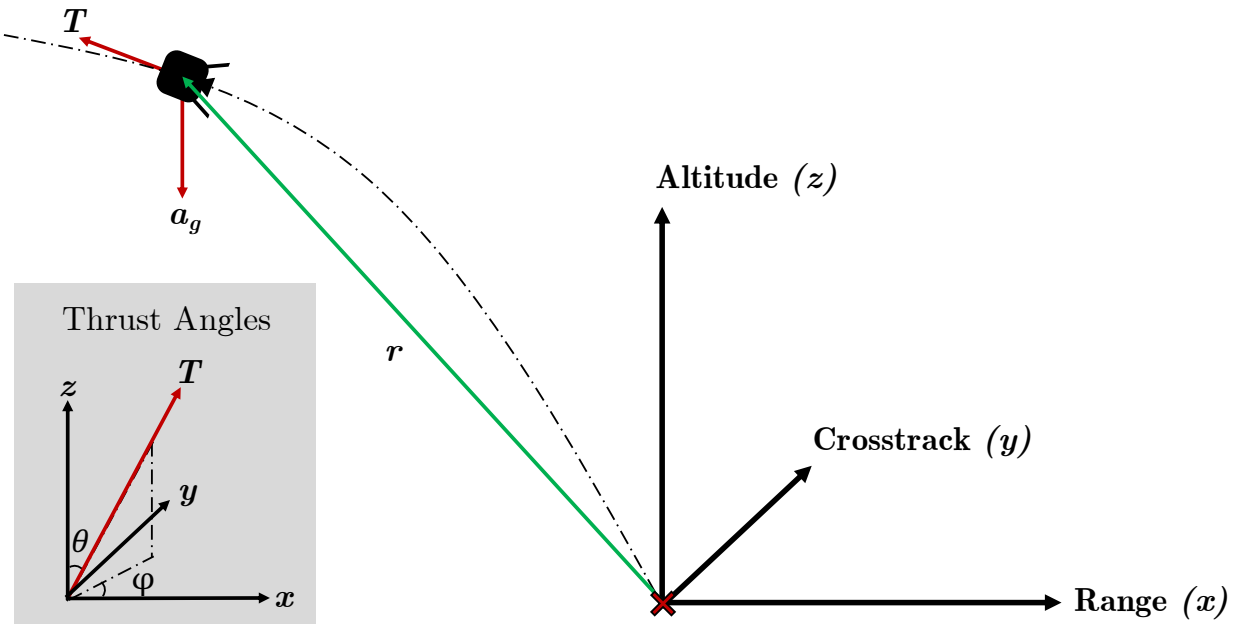


Figure 7.1: Coordinate frame definition for optimal powered descent pinpoint landing problem. Reprinted with permission from [2].

position and velocity state and an initial constraint on the mass state,

$$\begin{cases} \boldsymbol{r}(0) = \boldsymbol{r}_0 \\ \boldsymbol{v}(0) = \boldsymbol{v}_0 \end{cases}, \qquad \begin{cases} \boldsymbol{r}(t_f) = \boldsymbol{r}_f \\ \boldsymbol{v}(t_f) = \boldsymbol{v}_f \end{cases}, \qquad \text{and} \qquad m(0) = m_0.$$

In all, the objective is to minimize the mass of the propellant used while satisfying the dynamics constraints of the problem. Therefore, the problem can be posed as,

Optimization problem statement

$$\begin{aligned}
&\underset{t_f,T}{\text{minimize}} \quad \alpha \int_0^{t_f} T \, \mathrm{d}\tau \\
\text{subject to} \quad &\dot{\boldsymbol{r}} = \boldsymbol{v}, \quad \dot{\boldsymbol{v}} = \boldsymbol{a}_g + \frac{\boldsymbol{T}}{m}, \quad \dot{m} = -\alpha T \\
&0 \leq T_{min} \leq T \leq T_{max}, \quad ||\hat{\boldsymbol{t}}|| = 1 \\
&\boldsymbol{r}(0) = \boldsymbol{r}_0, \quad \boldsymbol{v}(0) = \boldsymbol{v}_0, \quad m(0) = m_0 \\
&\boldsymbol{r}(t_f) = \boldsymbol{r}_f, \quad \boldsymbol{v}(t_f) = \boldsymbol{v}_f
\end{aligned}$$

7.2 First-order necessary conditions

From our definition of the optimization problem, we next apply the indirect method by applying the PMP dictates that the Hamiltonian takes the following form [87],

$$H = \mathcal{L} + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{f} + \boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{C}$$

which can be expanded to,

$$H = \alpha T + \boldsymbol{\lambda}_r^{\mathrm{T}} \boldsymbol{v} + \boldsymbol{\lambda}_v^{\mathrm{T}} \left(\boldsymbol{a}_g + \frac{T}{m} \hat{\boldsymbol{t}} \right) - \lambda_m \alpha T + \mu_1 (T - T_{\max}) + \mu_2 (T_{\min} - T) \tag{7.2}$$

where $T - T_{\max} \leq 0$ and $T_{\min} - T \leq 0$ and $\mu_1 > 0, \mu_2 > 0$. According to PMP, the optimal thrust solution is one that minimizes the Hamiltonian. Because both the thrust T and mass m are both non-negative, $\hat{\boldsymbol{t}}$ should be in the opposite direction of of the velocity costate, i.e., $\hat{\boldsymbol{t}} = -\frac{\boldsymbol{\lambda}_v}{||\boldsymbol{\lambda}_v||}$. This is what in Lawden's theory [124] is called *primer's vector*. Thus Equation (7.2) can be rewritten as,

$$H = \alpha T + \boldsymbol{\lambda}_r^{\mathrm{T}} \boldsymbol{v} + \boldsymbol{\lambda}_v^{\mathrm{T}} \boldsymbol{a}_g - \frac{T}{m} ||\boldsymbol{\lambda}_v|| - \lambda_m \alpha T + \mu_1 (T - T_{\max}) + \mu_2 (T_{\min} - T)$$

Now, to determine optimal thrust magnitude, we impose that the partial derivative of the Hamiltonian with respect to the thrust (i.e., the control) is equal to zero, which is of the form of Equation (5.8),

$$S := \frac{\partial H}{\partial T} = \underbrace{\alpha - \frac{1}{m}||\boldsymbol{\lambda}_v|| - \alpha\lambda_m}_{\sigma} + \mu_1 - \mu_2 = 0$$

where there are three conditions that result in $S = 0$:

1. if $\mu_1 = \mu_2 = 0 \quad (T_{\min} < T < T_{\max}) \quad$ then $\quad \sigma = 0$

2. if $\mu_1 = 0,\ \mu_2 > 0 \quad (T = T_{\min}) \quad$ then $\quad \sigma - \mu_2 = 0 \quad \rightarrow \quad \sigma = \mu_2 > 0$

3. if $\mu_1 > 0,\ \mu_2 = 0 \quad (T = T_{\max}) \quad$ then $\quad \sigma + \mu_1 = 0 \quad \rightarrow \quad \sigma = -\mu_1 < 0$

Finally, one can conclude that the thrust magnitude has the following program:

$$T = \begin{cases} = T_{\max} & \text{if} \quad \sigma < 0 \\ = T_{\min} & \text{if} \quad \sigma > 0 \end{cases}$$

It has been demonstrated in Reference [123] that the singular case $\sigma = 0$ corresponds to a constant thrust perpendicular to the gravity vector, which is generally not possible for a powered descent problem. Therefore, a singular arc is not part of the sought optimal solution. Furthermore, it is straightforward to show that σ changes signs at most twice and is derived in detailed in Reference [123]. Consequently, the thrust magnitude can switch between min-max twice at the most. That is, in the most general case, the thrust magnitude has a max-min-max profile. Hence, we can write the thrust magnitude as a function of time with t_1 and t_2 as parameters, where t_1 and t_2 are the times where the switches happen, i.e., $T = T(t; t_1, t_2)$. This result implies that thrust is constant between switches, and therefore, the solution of Equation (7.1) is a piecewise linear function in terms of t_1 and t_2 detailed by

the following equation,

$$m(t;t_1,t_2) = \begin{cases} \text{if } t \leq t_1 & : m_0 - \alpha\Big[T_{\max}(t-t_0)\Big] \\ \text{if } t_1 \leq t \leq t_2 & : m_0 - \alpha\Big[T_{\max}(t_1-t_0) - T_{\min}(t-t_1)\Big] \\ \text{else} & : m_0 - \alpha\Big[T_{\max}(t_1-t_0) - T_{\min}(t_2-t_1) - T_{\max}(t-t_2)\Big]. \end{cases}$$

In addition to these conditions, we are left with the first-order necessary conditions for the costates as given by Equation (5.7),

$$\begin{aligned} \dot{\boldsymbol{\lambda}}_r &= -\frac{\partial H}{\partial \boldsymbol{r}} &&= \mathbf{0} \\ \dot{\boldsymbol{\lambda}}_v &= -\frac{\partial H}{\partial \boldsymbol{v}} &&= -\boldsymbol{\lambda}_r \\ \dot{\lambda}_m &= -\frac{\partial H}{\partial m} &&= -\frac{T}{m^2}||\boldsymbol{\lambda}_v||. \end{aligned}$$

Finally, since the final mass state is unconstrained, Equation (5.10) implies that,

$$\lambda_m(t_f) = 0,$$

and likewise, since the final time of the problem is unknown, Equations (5.11) leads to the condition on the final value of the Hamiltonian.

$$H(t_f) = 0.$$

In fact, since the Hamiltonian is not an explicit function of time, the partial derivative with respect to time is zero (i.e., $\frac{\partial H}{\partial t} = 0$), which implies a stronger condition, that the Hamiltonian should be zero for all time,

$$H(t) = 0.$$

We will take these conditions and look to apply the TFC method to solve all of the equations simultaneously.

7.3 Solving the problem via the TFC

With the simplifications introduced in the previous section, the following nonlinear set of equations must be solved to find the optimal state and thrust program,

First-order necessary conditions

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{7.3}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a}_g - \beta(t)\frac{\boldsymbol{\lambda}_v}{||\boldsymbol{\lambda}_v||} \tag{7.4}$$

$$\dot{\boldsymbol{\lambda}}_r = \boldsymbol{0} \tag{7.5}$$

$$\dot{\boldsymbol{\lambda}}_v = -\boldsymbol{\lambda}_r \tag{7.6}$$

$$\dot{\lambda}_m = -\frac{T(t;t_1,t_2)}{m^2}||\boldsymbol{\lambda}_v|| \tag{7.7}$$

$$H(t_f) = 0 = \alpha T(t_f;t_1,t_2) + \boldsymbol{\lambda}_v^{\mathrm{T}}(t_f)\left(\boldsymbol{a}_g - \beta(t_f)\frac{\boldsymbol{\lambda}_v(t_f)}{||\boldsymbol{\lambda}_v(t_f)||}\right) \tag{7.8}$$

where we define

$$\beta(t) := \frac{T(t;t_1,t_2)}{m(t)}$$

and Equations (7.3), (7.4), and (7.7) are subject to

$$\boldsymbol{r}(0) = \boldsymbol{r}_0, \quad \boldsymbol{v}(0) = \boldsymbol{v}_0, \quad \boldsymbol{r}(t_f) = \boldsymbol{r}_f, \quad \boldsymbol{v}(t_f) = \boldsymbol{v}_f, \quad \lambda_m(t_f) = 0.$$

It must be noted that λ_m only shows up in Equation (7.7), and can therefore be solved independently. Since the transversality condition gives $\lambda_m(t_f) = 0$, Equation (7.7) can be solved by back propagation or by simply using the TFC method.

Since this problem's solution exhibits a bang-bang profile for thrust, the original formulation of the TFC method (i.e., as used in the Outer-loop method of the energy optimal

landing problem in Section 6.3.1) must be adjusted to accommodate switching behavior in the control. In general, this can be labeled as a hybrid system because the dynamical behavior is governed by both continuous dynamics (when the thruster is firing) and discrete dynamics (when the thrust jumps). The general theory for this extension to hybrid systems has been developed in Section 4.8.1 but is also fully developed in the following equations. Additionally, a few equations are redundant and can be removed completely via the TFC constrained expression to further simplify the solution of this nonlinear system of equations. As done in the last section, the differential equation expressed by Equation (7.3) is unnecessary and can be disregarded. Similarly, the equations for $\dot{\boldsymbol{\lambda}}_r$ and $\dot{\boldsymbol{\lambda}}_v$ can be simplified. First, let us express the vector equations as three scalar equations, each where the index i represents the individual components. Using this notation, we can expand $\boldsymbol{\lambda}_v$ such that,

$$\lambda_{v_i} = a_{0_i} + a_{1_i} z = \boldsymbol{h}_\lambda^{\mathrm{T}} \boldsymbol{\xi}_{\lambda_i}, \quad \text{for} \quad i = 1, 2, 3$$

which satisfies Equations (7.5-7.6) through

$$\begin{aligned} \dot{\lambda}_{v_i} &= c_\lambda \lambda'_{v_i} = c_\lambda a_{1_i} \\ -\dot{\lambda}_{v_i} &= \lambda_{r_i} = -c_\lambda a_{1_i}. \end{aligned}$$

This process reduces the problem to the solution of a single differential equation expressed by Equation (7.4) and an algebraic equation for the Hamiltonian at the final time given by Equation (7.8). Rewriting the differential equation in indicial notation and collecting all terms on one side, a loss function based on the residuals of the differential equation can be defined,

$$\mathbb{L}_i = a_i - a_{g_i} + \beta(t)\, \lambda_{v_i} \left(\sum_{j=1}^{3} \lambda_{v_j}^2 \right)^{-1/2} \qquad \text{for} \qquad i = 1, 2, 3 \tag{7.9}$$

where $a_i := \dot{v}_i$ (or simply the acceleration of the spacecraft). Now, the only step left is to construct a constrained expression for the state variables. In the above derivation of

the thrust structure, we have shown that the thrust switches at most twice, leading to a max-min-max profile. Therefore, the function $\beta(t)$ in Equation (7.9) jumps twice along the solution trajectory. This switching causes three distinct differential equations that cannot be solved with a single polynomial expansion over the entire domain, as was done for the energy-optimal guidance in Chapter 6. Therefore, a new formulation for the TFC approach has been developed to handle these hybrid systems [1]. This process allows for the continuity between each segment of the domain. As shown in Figure 7.2, it is apparent that all sub-

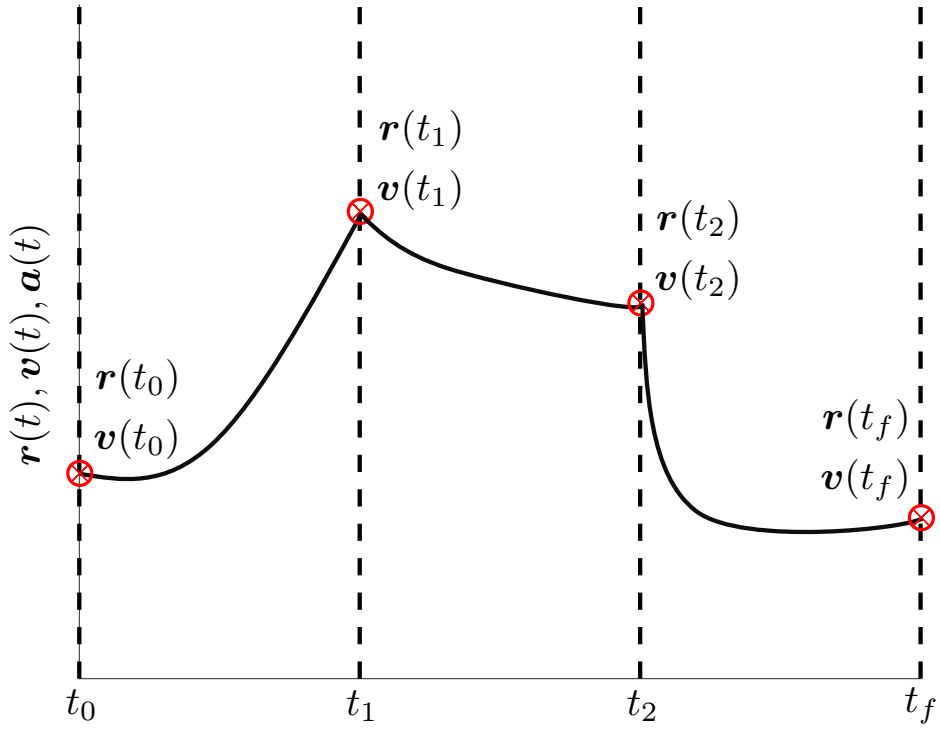


Figure 7.2: Visual representation of piece-wise approach using the TFC method. In this derivation, the constrained expressions maintain continuity of position and velocity through embedded relative constraints. Reprinted with permission from [2].

domains share the same constraint conditions (i.e, the initial and final position and velocity are constrained). Therefore, a single constraint expression can be derived for the case of arbitrary constraint locations and then incorporated into the sub-domains. The constrained

expression for this specific case was derived in Section 6.3.1 and it is captured by Equation (6.10). Consequently, the position, velocity, and acceleration constrained expression can be expressed as,

$$\begin{aligned} r_i(t, g_i(t)) = g_i(t) + \phi_1(t)(r_{0_i} - g_i(t_0)) + \phi_2(t)(r_{f_i} - g_i(t_f)) \\ + \phi_3(t)(v_{0_i} - \dot{g}_i(t_0)) + \phi_4(t)(v_{f_i} - \dot{g}_i(t_f)) \end{aligned} \tag{7.10}$$

$$\begin{aligned} v_i(t, g_i(t)) = \dot{g}_i(t) + \dot{\phi}_1(t)(r_{0_i} - g_i(t_0)) + \dot{\phi}_2(t)(r_{f_i} - g_i(t_f)) \\ + \dot{\phi}_3(t)(v_{0_i} - \dot{g}_i(t_0)) + \dot{\phi}_4(t)(v_{f_i} - \dot{g}_i(t_f)) \\ a_i(t, g_i(t)) = \ddot{g}_i(t) + \ddot{\phi}_1(t)(r_{0_i} - g_i(t_0)) + \ddot{\phi}_2(t)(r_{f_i} - g_i(t_f)) \\ + \ddot{\phi}_3(t)(v_{0_i} - \dot{g}_i(t_0)) + \ddot{\phi}_4(t)(v_{f_i} - \dot{g}_i(t_f)) \end{aligned} \tag{7.11}$$

The switching functions are the same as those used in the Outer-loop method for solving the energy-optimal landing problem (this is because they share the same constraint conditions) and are defined by switching functions of Section 6.3.1. In these switching functions, t_0 and t_f must be replaced with the respective segment's initial and final time, e.g., for the first segment $t \in [t_0, t_1]$.

The constrained expression detailed by Equations (7.10-7.11) can be used as a template to write the constrained expressions for each segment of the solution trajectory. In order to explicitly identify the segment, the pre-superscript notation will be used. For example, ${}^{(1)}r_i$ describes the position constrained expression for the first segment defined on $t \in [t_0, t_1]$. For this problem, $s = 1$ (where s is used to denote the segment) is defined on $t \in [t_0, t_1]$, $s = 2$ is defined on $t \in [t_1, t_2]$, and $s = 3$ is defined on $t \in [t_2, t_f]$. Using this formulation, the constrained expressions of position for each segment are,

$$\begin{aligned} {}^{(1)}r_i(t, g_i(t)) = {}^{(1)}g_i(t) + {}^{(1)}\phi_1(t)\left(r_{0_i} - {}^{(1)}g_i(t_0)\right) + {}^{(1)}\phi_2(t)\left(r_{1_i} - {}^{(1)}g_i(t_f)\right) \\ + {}^{(1)}\phi_3(t)\left(v_{0_i} - {}^{(1)}\dot{g}_i(t_0)\right) + {}^{(1)}\phi_4(t)\left(v_{1_i} - {}^{(1)}\dot{g}_i(t_f)\right) \end{aligned}$$

$$
\begin{aligned}
{}^{(2)}r_i(t, g_i(t)) = {}^{(2)}g_i(t) + {}^{(2)}\phi_1(t)\left(r_{1_i} - {}^{(2)}g_i(t_0)\right) + {}^{(2)}\phi_2(t)\left(r_{2_i} - {}^{(2)}g_i(t_f)\right) \\
+ {}^{(2)}\phi_3(t)\left(v_{1_i} - {}^{(2)}\dot{g}_i(t_0)\right) + {}^{(2)}\phi_4(t)\left(v_{2_i} - {}^{(2)}\dot{g}_i(t_f)\right)
\end{aligned}
$$

$$
\begin{aligned}
{}^{(3)}r_i(t, g_i(t)) = {}^{(3)}g_i(t) + {}^{(3)}\phi_1(t)\left(r_{2_i} - {}^{(3)}g_i(t_0)\right) + {}^{(3)}\phi_2(t)\left(r_{f_i} - {}^{(3)}g_i(t_f)\right) \\
+ {}^{(3)}\phi_3(t)\left(v_{2_i} - {}^{(3)}\dot{g}_i(t_0)\right) + {}^{(3)}\phi_4(t)\left(v_{f_i} - {}^{(3)}\dot{g}_i(t_f)\right)
\end{aligned}
$$

where the derivative of these functions follow the form of Equations (7.10-7.11). This allows us to collect the unknown $\boldsymbol{\xi}_i$ vectors and write the constrained expression in the form,

$$
\begin{aligned}
{}^{(1)}r_i(t, {}^{(1)}\boldsymbol{\xi}_i) = {}^{(1)}\Big(\boldsymbol{h}(z) - \phi_1(t)\boldsymbol{h}(z_0) - \phi_2(t)\boldsymbol{h}(z_f) - \phi_3(t)c\boldsymbol{h}_z(z_0) - \phi_4(t)c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\,{}^{(1)}\boldsymbol{\xi}_i \\
+ {}^{(1)}\phi_1(t)r_{0_i} + {}^{(1)}\phi_2(t)r_{1_i} + {}^{(1)}\phi_3(t)v_{0_i} + {}^{(1)}\phi_4(t)v_{1_i}
\end{aligned}
$$

$$
\begin{aligned}
{}^{(2)}r_i(t, {}^{(2)}\boldsymbol{\xi}_i) = {}^{(2)}\Big(\boldsymbol{h}(z) - \phi_1(t)\boldsymbol{h}(z_0) - \phi_2(t)\boldsymbol{h}(z_f) - \phi_3(t)c\boldsymbol{h}_z(z_0) - \phi_4(t)c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\,{}^{(2)}\boldsymbol{\xi}_i \\
+ {}^{(2)}\phi_1(t)r_{1_i} + {}^{(2)}\phi_2(t)r_{2_i} + {}^{(2)}\phi_3(t)v_{1_i} + {}^{(2)}\phi_4(t)v_{2_i}
\end{aligned}
$$

$$
\begin{aligned}
{}^{(3)}r_i(t, {}^{(3)}\boldsymbol{\xi}_i) = {}^{(3)}\Big(\boldsymbol{h}(z) - \phi_1(t)\boldsymbol{h}(z_0) - \phi_2(t)\boldsymbol{h}(z_f) - \phi_3(t)c\boldsymbol{h}_z(z_0) - \phi_4(t)c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\,{}^{(3)}\boldsymbol{\xi}_i \\
+ {}^{(3)}\phi_1(t)r_{2_i} + {}^{(3)}\phi_2(t)r_{f_i} + {}^{(3)}\phi_3(t)v_{2_i} + {}^{(3)}\phi_4(t)v_{f_i}
\end{aligned}
$$

Along with the linear unknowns in ${}^{(s)}\boldsymbol{\xi}_i$, the equations share linear unknowns in $r_{1_i}, v_{1_i}, r_{2_i}, v_{2_i}$ which serve as the embedded relative constraints between adjacent segments. With this new formulation, we now have three separate loss functions based on the residual of the differential equation over each segment (s) which are as follows,

$$
{}^{(s)}F_i\left(t, \Xi\right) = {}^{(s)}a_i - a_{g_i} + \beta(t)\,\lambda_{v_i}\left(\sum_{j=1}^{3}\lambda_{v_j}^2\right)^{-1/2}.
$$

Note that although the costate constrained expressions do not need to be split into separate domains, special attention must be paid to discretizing the equations according to the segment time ranges. Again, to solve for the unknown $\boldsymbol{\xi}_i$ parameters, a nonlinear least-squares technique was used, which requires computing the partials of the loss function with respect to all of the unknowns. All partial derivatives for each segment and each unknown are provided in Appendix D.5.

In addition to the loss functions for the problem dynamics given by Equation (7.4), a loss function associated with the transversality conditions for the Hamiltonian is defined as,

$$\mathbb{L}_H\left(t_f, \Xi\right) = \alpha T_{\max} + \sum_{i=1}^{3} \lambda_{v_i}(t_f) a_{g_i} - \beta(t_f) \left(\sum_{i=1}^{3} \lambda_{v_i}^2(t_f) \right)^{\frac{1}{2}}.$$

The partial derivatives of this function are also provided in Appendix D.5. Next, by discretizing the domain over N points, these loss functions can be organized into the loss vector,

$$\mathbb{L} = \left\{ {}^{(1)}\mathbb{L}_1^{\mathrm{T}} \quad {}^{(1)}\mathbb{L}_2^{\mathrm{T}} \quad {}^{(1)}\mathbb{L}_3^{\mathrm{T}} \quad {}^{(2)}\mathbb{L}_1^{\mathrm{T}} \quad {}^{(2)}\mathbb{L}_2^{\mathrm{T}} \quad {}^{(2)}\mathbb{L}_3^{\mathrm{T}} \quad {}^{(3)}\mathbb{L}_1^{\mathrm{T}} \quad {}^{(3)}\mathbb{L}_2^{\mathrm{T}} \quad {}^{(3)}\mathbb{L}_3^{\mathrm{T}} \quad \mathbb{L}_H \right\}^{\mathrm{T}}_{(\{9N+1\}\times 1)}$$

where

$${}^{(s)}\mathbb{L}_i = \left\{ {}^{(s)}F_i(t_0, \Xi) \quad \dots \quad {}^{(s)}F_i(t_k, \Xi) \quad \dots \quad {}^{(s)}F_i(t_f, \Xi) \right\}^{\mathrm{T}}.$$

Additionally, the vector of unknowns takes the form,

$$\Xi = \left\{ {}^{(1)}\boldsymbol{\xi}_1^{\mathrm{T}} \quad {}^{(1)}\boldsymbol{\xi}_2^{\mathrm{T}} \quad {}^{(1)}\boldsymbol{\xi}_3^{\mathrm{T}} \quad {}^{(2)}\boldsymbol{\xi}_1^{\mathrm{T}} \quad {}^{(2)}\boldsymbol{\xi}_2^{\mathrm{T}} \quad {}^{(2)}\boldsymbol{\xi}_3^{\mathrm{T}} \quad {}^{(3)}\boldsymbol{\xi}_1^{\mathrm{T}} \quad {}^{(3)}\boldsymbol{\xi}_2^{\mathrm{T}} \quad {}^{(3)}\boldsymbol{\xi}_3^{\mathrm{T}} \right.$$
$$\left. \boldsymbol{\xi}_{\lambda_1}^{\mathrm{T}} \quad \boldsymbol{\xi}_{\lambda_2}^{\mathrm{T}} \quad \boldsymbol{\xi}_{\lambda_3}^{\mathrm{T}} \quad \boldsymbol{r}_1^{\mathrm{T}} \quad \boldsymbol{v}_1^{\mathrm{T}} \quad \boldsymbol{r}_2^{\mathrm{T}} \quad \boldsymbol{v}_2^{\mathrm{T}} \right\}^{\mathrm{T}}_{(9m+18)}.$$

In general, the structure of the Jacobian is,

$$\mathbb{J} = \begin{bmatrix} {}^{(1)}J_{\boldsymbol{\xi}} & \mathbf{0}_{(3N\times 3m)} & \mathbf{0}_{(3N\times 3m)} & {}^{(1)}J_{\boldsymbol{\xi}_\lambda} & {}^{(1)}J_{r_1,v_1} & \mathbf{0}_{(3N\times 6)} \\ \mathbf{0}_{(3N\times 3m)} & {}^{(2)}J_{\boldsymbol{\xi}} & \mathbf{0}_{(3N\times 3m)} & {}^{(2)}J_{\boldsymbol{\xi}_\lambda} & {}^{(2)}J_{r_1,v_1} & {}^{(2)}J_{r_2,v_2} \\ \mathbf{0}_{(3N\times 3m)} & \mathbf{0}_{(3N\times 3m)} & {}^{(3)}J_{\boldsymbol{\xi}} & {}^{(3)}J_{\boldsymbol{\xi}_\lambda} & \mathbf{0}_{(3N\times 6)} & {}^{(3)}J_{r_2,v_2} \\ \mathbf{0}_{(1\times 3m)} & \mathbf{0}_{(1\times 3m)} & \mathbf{0}_{(1\times 3m)} & J_H & \mathbf{0}_{(1\times 6)} & \mathbf{0}_{(1\times 6)} \end{bmatrix}_{(\{9N+1\}\times\{9m+18\})} . \quad (7.12)$$

Finally, using Equation (7.12) along with the augmented loss functions and unknown vector, an iterative least-squares is used to find Ξ.

7.3.1 Jacobian properties and sparsity

From the prior equations, it should be evident that the Jacobian defined by Equation (7.12) will need to be inverted. Therefore, Figure 7.3 is provided as a visual aid to highlight the sparsity structure of this Jacobian. In addition to this structure, another property of this matrix is that the elements dealing with continuity, Jacobian terms ${}^{(1)}J_{r_1,v_1}$, ${}^{(2)}J_{r_1,v_1}$, ${}^{(2)}J_{r_2,v_2}$, and ${}^{(3)}J_{r_2,v_2}$, highlighted in the right side of Figure 7.3, are parameter independent (i.e., they are only a function of the $\phi(t)$ terms, or rather time) and therefore are constant and need only to be computed once per TFC loop.

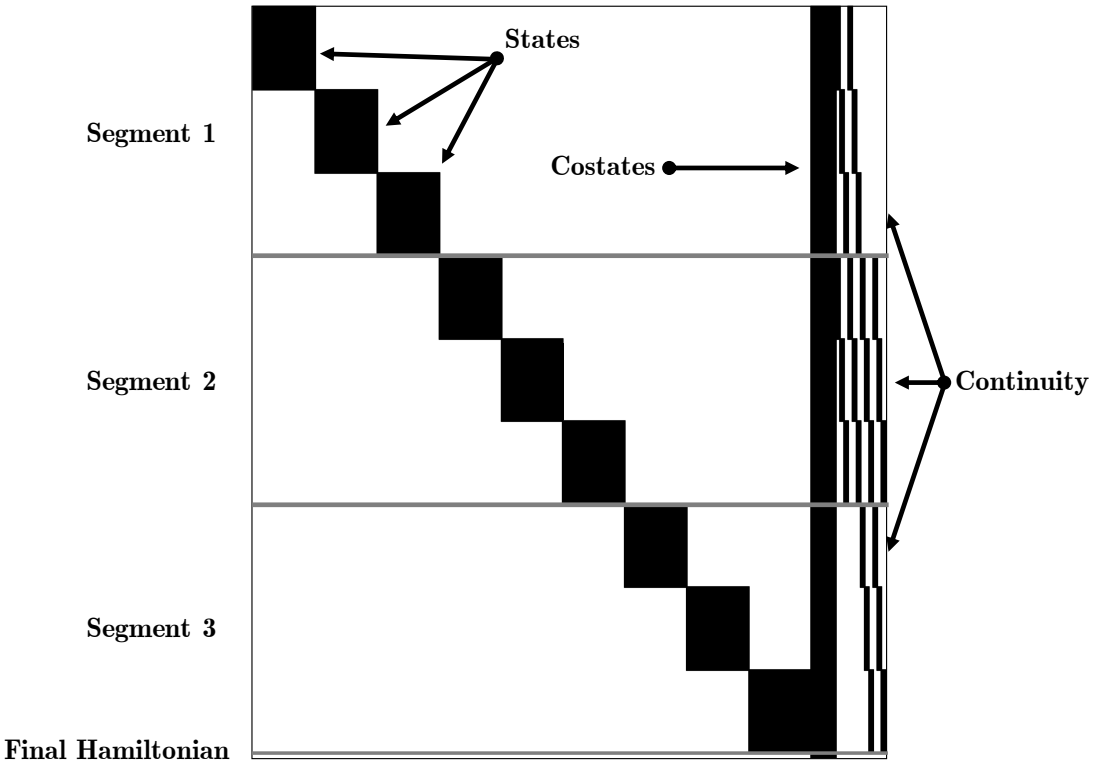


Figure 7.3: Visual representation of the Jacobian matrix to be inverted where the black elements represent the nonzero entries. Reprinted with permission from [2].

7.3.2 Initialization of parameters

An initial estimate of the parameters is needed to initialize the iterative least-squares process. Since the problem is a boundary-value problem, the first guess for ${}^{(s)}\boldsymbol{\xi}_i$, $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{v}_1$, and $\boldsymbol{v}_2$ can be determined by simply connecting the initial and final position with a straight line and using this trajectory for a least-squares fitting of the constrained expressions describing the ${}^{(s)}r_i$ terms. Next, since λ_{v_i} is related to the thrust direction, it can be assumed,

$$\boldsymbol{\lambda}_{v_0} = \frac{\boldsymbol{v}_0}{||\boldsymbol{v}_0||},$$

similar to that presented in Reference [123] (Equation (51) in the text) However, the initialization of $\boldsymbol{\lambda}_r = \mathbf{0}$ will cases issues in the TFC method because this involves setting $\boldsymbol{\xi}_\lambda$

coefficients to zeros. Therefore, in this dissertation, the coefficients are initialized using,

$$\boldsymbol{\lambda}_{v_f} = -\frac{\boldsymbol{r}_0}{||\boldsymbol{r}_0||}.$$

7.4 Summary of Algorithm

Overall, the TFC method was used as ab inner-loop function to minimize the residuals of the first-order necessary conditions subject to a prescribed thrust profile $T(t; t_1, t_2)$, i.e., the switching times, t_1 and t_2, and the final time, t_f, are assumed to be known by the TFC-based inner-loop routine. Consequently, an outer-loop routine has been developed to optimize the three time parameters t_1, t_2, t_f given the L_2-norms of the residual of the first-order conditions, and the Hamiltonian over the first two segments (here, MATLAB's [125] `fsolve` was used). In other words, the following minimization problem needs to be solved for $t_1, t_2, \text{and} t_f$,

$$\min_{t_1,t_2,t_f} \boldsymbol{F}(t_1, t_2, t_f) = \left[\max|\mathbb{L}|, \qquad \max|\,^{(1)}H(t)|, \qquad \max|\,^{(2)}H(t)|\right]^{\mathrm{T}}, \tag{7.13}$$

where $\mathbb{L}$ is the loss function of the inner TFC loop, and $^{(1)}H(t)$ and $^{(2)}H(t)$ are the Hamiltonian values over the first and second segment, respectively, evaluated using the inner loop converged parameters. A flow chart of the relevant inputs and outputs is provided in Figure 7.4.

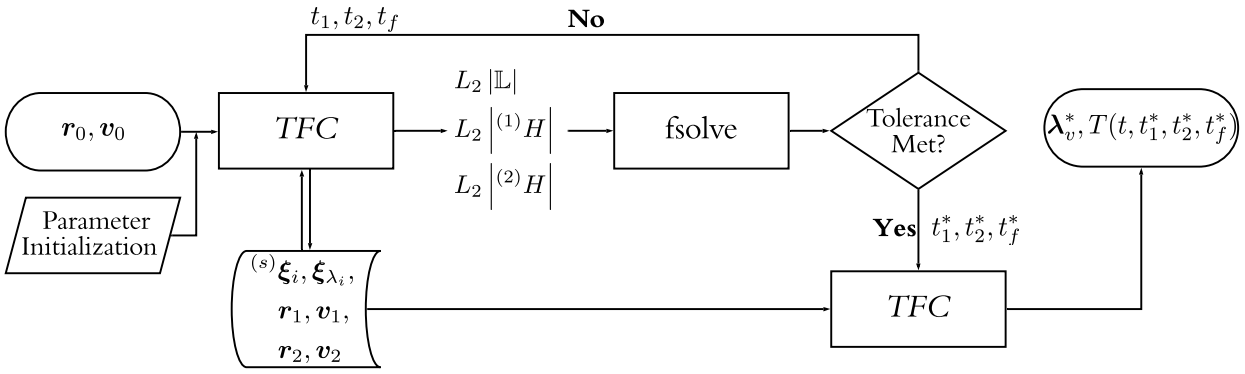


Figure 7.4: Summary of the full algorithm used with the TFC approach. Reprinted with permission from [2].

Following the process given in Figure 7.4, the initial conditions $\boldsymbol{r}_0$ and $\boldsymbol{v}_0$ along with initial guesses for t_1, t_2, and t_f are fed into the TFC method to minimize $\mathbb{L}$. The converged parameters are used to evaluate the Hamiltonian over the first and second segments. Using the norm of these quantities, `fsolve` is used to solve the minimization problem given in Equation (7.13). If the tolerance of the outer loop is met (in all tests, the step and function tolerance of `fsolve` were set to 4.4×10^{-16}), the t_1, t_2, and t_f are considered optimal, and the TFC loop is ran one more time to compute the optimal trajectory.[1]

7.5 Results

The proposed method was validated using two specific test cases based on selected initial conditions defining a powered descent guidance scenario for landing on Mars. In Example 7.1, the algorithm is tested on initial conditions where the optimal trajectory is characterized by a min-max thrust profile. Furthermore, in Example 7.2, the case where the optimal thrust profile is max-min-max is studied. In both cases, the results were compared with GPOPS-II solutions. The algorithm was fully implemented in MATLAB R2019*a*, and therefore not optimized for speed,

[1]It must be noted that these low tolerances were used to quantify the baseline for speed and accuracy of the method. For implementation, the accuracy needed can be used to tune the tolerance and increase the algorithm's computational speed.

Similar to the energy-optimal landing problem in Chapter 6, the problem was scaled by the initial conditions for the numerical implementation. The unit length, ℓ^*, and unit time, t^*, where calculated by the following equations,

$$\ell^* = \max\left(|\boldsymbol{r}_0|\right)$$
$$t^* = \frac{\ell^*}{\max\left(|\boldsymbol{v}_0|\right)}.$$

7.5.1 Constant Test Parameters

We consider the trajectory optimization problem for a spacecraft performing powered descent for a pinpoint landing on Mars. The gravitational field is assumed constant, as generally, the powered descent starts below 1.5 km. For the numerical test, the lander parameters have been assumed to be similar to the ones presented in Reference [81] and reported in Table 7.1. Thrust magnitude bounds and the α parameter are defined as

Table 7.1: Constant parameters used in test cases. Reprinted with permission from [2].

Variable	Value
$\boldsymbol{a}_g$ [m/s^2]	$\{0, \quad 0, \quad -3.7114\}^{\mathrm{T}}$
I_{sp} [s]	225
g_0 [m/s^2]	9.807
$\overline{T}$ [N]	$3,100$
N_T [-]	6
ϕ_T [deg]	27

follows:

$$T_{\min} = 0.3\overline{T}N_T\cos\phi_T \approx 4,971.81 \ \ [\mathrm{N}]$$
$$T_{\max} = 0.8\overline{T}N_T\cos\phi_T \approx 13,258.18\ [\mathrm{N}]$$

where $\overline{T}$ is the maximum thrust for a single engine, N_T is the number of thrusters in the lander, and ϕ_T is the cant angle of the thrusters with respect to the lander, and

$$\alpha = \frac{1}{I_{\text{sp}}\, g_0\, \cos\phi_T} \approx 5.0863 \cdot 10^{-4}\ [\text{s/m}],$$

where I_{sp} is the engines' specific impulse and g_0 is Earth's gravitational constant.

Example 7.1: Test 1: Min-Max Trajectory

For Test 1, initial conditions were selected such that the optimal thrust profile would be min-max, i.e., switch between minimum thrust to maximum thrust. Table 7.2 defines the boundary conditions for this test case, and Figure 7.5 provides the converged trajectory using the TFC approach.

Table 7.2: Boundary conditions for min-max trajectory profile test case. Reprinted with permission from [2].

Variable	Initial	Final
$\boldsymbol{r}$ [m]	$\left\{-900,\quad 10,\quad 1500\right\}^{\mathrm{T}}$	$\left\{0,\quad 0,\quad 0\right\}^{\mathrm{T}}$
$\boldsymbol{v}$ [m/s]	$\left\{30,\quad -10,\quad -70\right\}^{\mathrm{T}}$	$\left\{0,\quad 0,\quad 0\right\}^{\mathrm{T}}$
m [kg]	1905	-

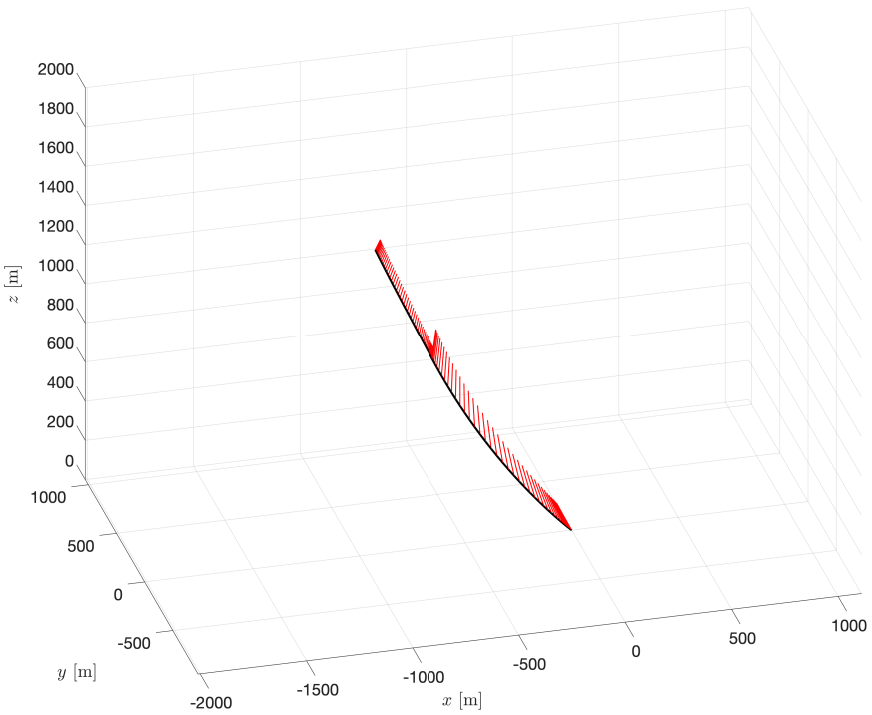


Figure 7.5: Landing trajectory for min-max thrust profile based on initial conditions, $\boldsymbol{r}_0 = \{-900,\ 10,\ 1500\}^{\mathrm{T}}$[m], $\boldsymbol{v}_0 = \{30,\ -10,\ -70\}^{\mathrm{T}}$ [m/s], $m_0 = 1905$ [kg]. Reprinted with permission from [2].

In addition to the trajectory, component plots of the position, velocity, and acceleration are provided in Figure 7.6. Furthermore, this figure also plots the residual of the governing differential equations for mass and acceleration to quantify the method's accuracy. It can be seen that the TFC residual is about $\mathcal{O}(10^{-11})$ or less for the whole solution domain.

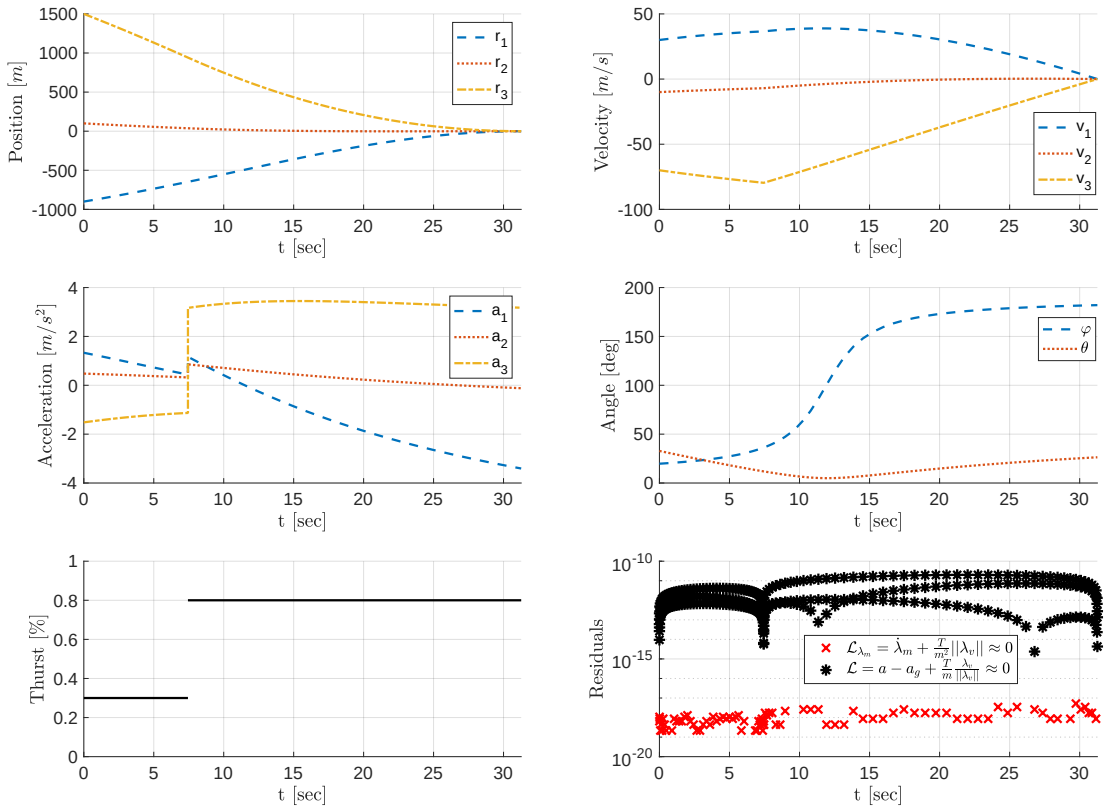


Figure 7.6: TFC solution of the min-max thrust profile case. The solution is presented in terms of the position, velocity, acceleration, and residuals of the differential equations. Reprinted with permission from [2].

The accuracy of this approach was also compared to results obtained using GPOPS-II [126] and is quantified in terms of the converged parameters, the L_2-norms of the Hamiltonian, and propellant mass used. Moreover, to further justify the accuracy of the solution, the converged parameters of initial costate values and switching times for each method were propagated using MATLAB's `ode45` with a tolerance of 2.2×10^{-14} to check the final position and velocity error and also the final error of the λ_m term. The tabulated values of this test are provided in Table 7.3. In this test, `fsolve` iterated 27 times with each TFC inner-loop averaging 76 ms, resulting in a total execution time of 2.1 seconds within the MATLAB implementation. Further, during this test, the

TFC method converged in about 6 iterations every function call. Additionally, as a last point of comparison, the time histories of the Hamiltonian for both methods are plotted in Figure 7.7.

Table 7.3: Converged parameters for the TFC and GPOPS-II solution for the min-max trajectory test case. The values $||\boldsymbol{r}(t_f)||$, $||\boldsymbol{v}(t_f)||$, and $\lambda_m(t_f)$ were determined by propagating both TFC and GPOPS-II converged solutions in order to have a one-to-one comparison on the accuracy of the converged solutions. Reprinted with permission from [2].

Variable	TFC	GPOPS-II [126]
$L_2[\mathbb{L}]$	$1.036 \cdot 10^{-10}$	–
$L_2[H]$	$5.488 \cdot 10^{-11}$	$1.064 \cdot 10^{-3}$
m_{used} [kg]	179.447	179.447
t_1 [s]	7.4430	7.4430
t_f [s]	31.2623	31.2623
$\|\|\boldsymbol{r}(t_f)\|\|$ [m]	$2.886 \cdot 10^{-9}$	$1.535 \cdot 10^{-2}$
$\|\|\boldsymbol{v}(t_f)\|\|$ [m]	$3.166 \cdot 10^{-10}$	$7.649 \cdot 10^{-4}$
$\lambda_m(t_f)$ [s]	$4.496 \cdot 10^{-14}$	$-4.193 \cdot 10^{-7}$

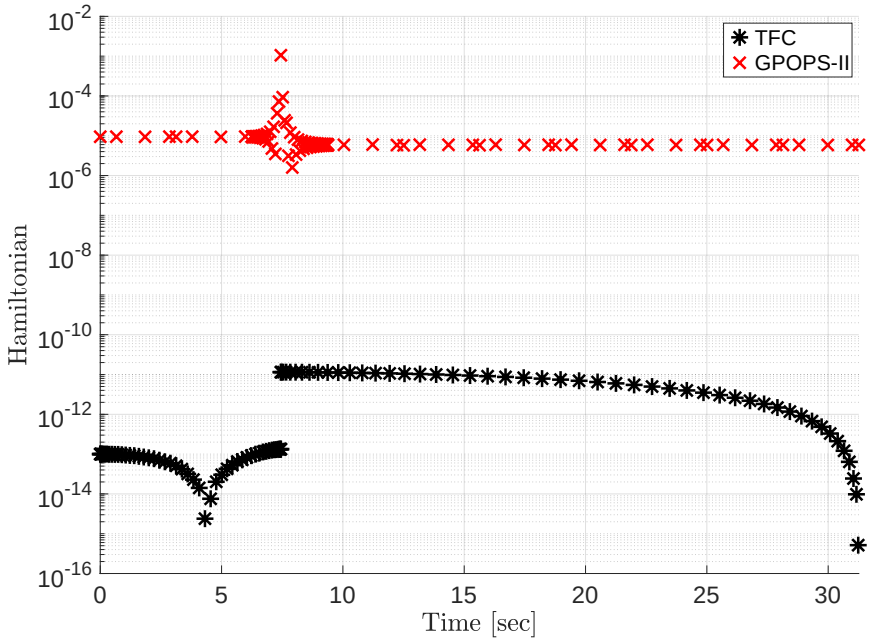


Figure 7.7: Comparison of Hamiltonian for TFC and GPOPS-II converged solutions for the min-max trajectory. Reprinted with permission from [2].

Example 7.2: Test 2: Max-Min-Max Trajectory

In test case 2, the initial conditions were specified such that the optimal solution exhibited a max-min-max profile, i.e., the thrust switches twice, max-min and min-max. The boundary conditions for this case are provided in Table 7.4, whereas Figure 7.8 reports the shape of the trajectory computed using the TFC-based algorithm.

Table 7.4: Boundary conditions for max-min-max trajectory profile test case. Reprinted with permission from [2].

Variable	Initial	Final
$\boldsymbol{r}$ [m]	$\{-200, \quad 100, \quad 1500\}^{\mathrm{T}}$	$\{0, \quad 0, \quad 0\}^{\mathrm{T}}$
$\boldsymbol{v}$ [m/s]	$\{85, \quad 50, \quad -65\}^{\mathrm{T}}$	$\{0, \quad 0, \quad 0\}^{\mathrm{T}}$
m [kg]	1905	-

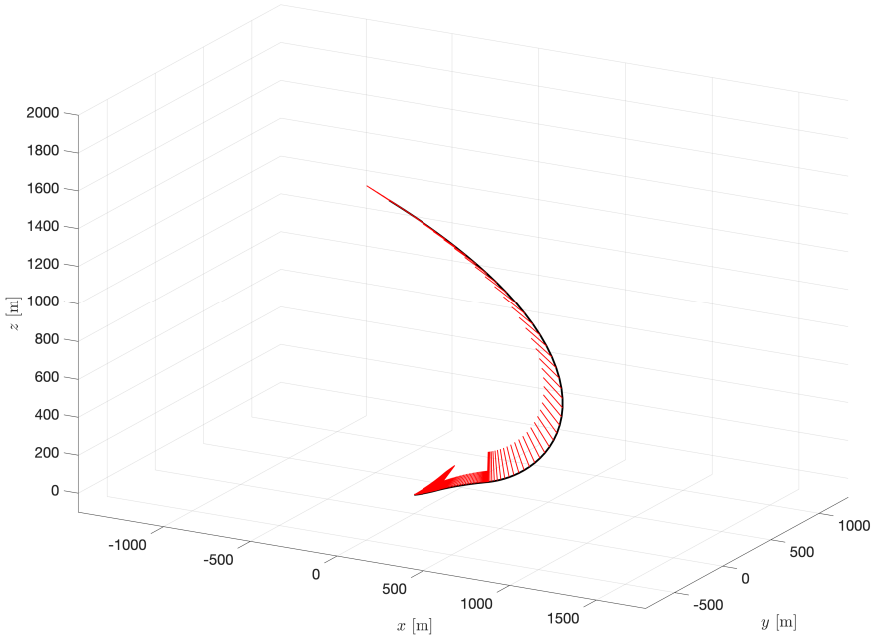


Figure 7.8: Landing trajectory for max-min-max thrust profile based on initial conditions, $\boldsymbol{r}_0 = \{-200, \quad 100, \quad 1500\}^{\mathrm{T}}$[m], $\boldsymbol{v}_0 = \{85, \quad -50, \quad -65\}^{\mathrm{T}}$ [m/s], $m_0 = 1905$ [kg]. Reprinted with permission from [2].

Again, the TFC solution history is reported for each component of position, velocity, and acceleration in Figure 7.9. The error is quantified by the residual of the governing equation of motion and the mass costate equation. It can be seen that the TFC residual is $\mathcal{O}(10^{-12})$ or less for the whole solution domain.

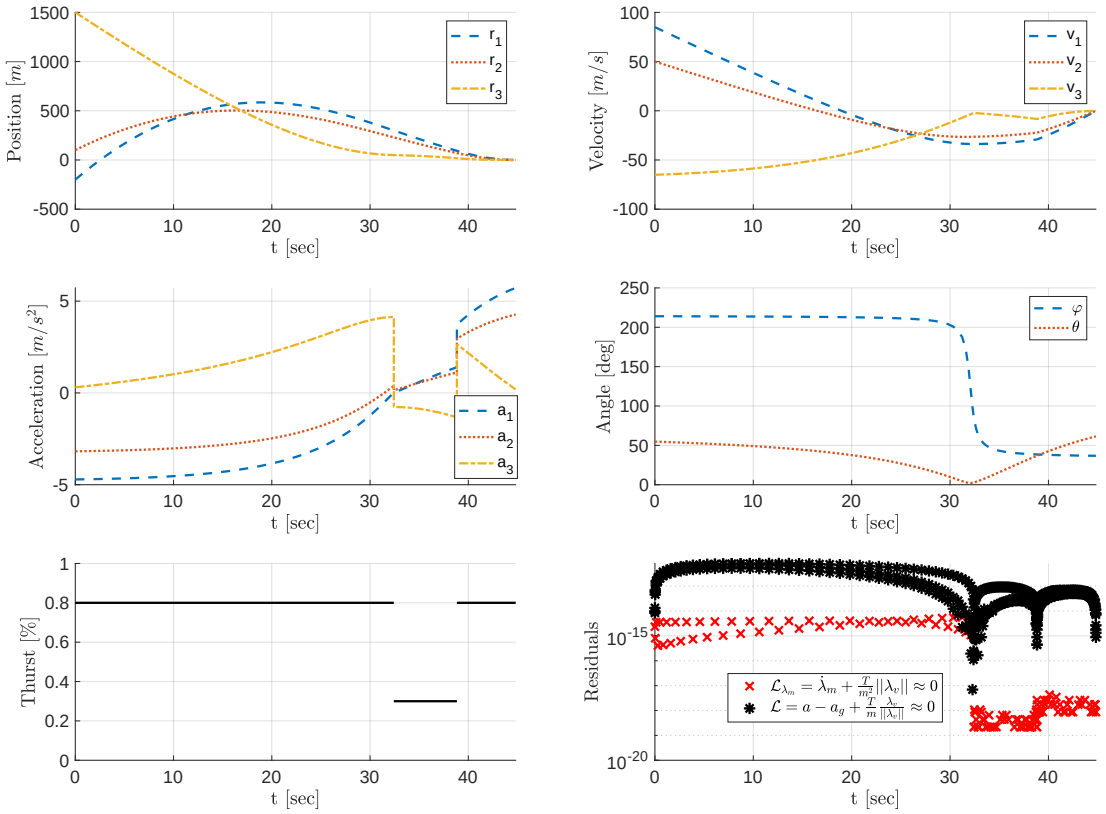


Figure 7.9: TFC solution of the max-min-max thrust profile case. The solution is presented in terms of the position, velocity, acceleration, and residuals of the differential equations. Reprinted with permission from [2].

Similar to test case 1, the solution is compared with the one obtained via GPOPS-II [126] for all converged parameters, which now includes another switching time, t_2. It can be seen that the magnitude of associated errors is similar to those presented in Section 7.1. In this test, `fsolve` iterated 32 times with each TFC inner-loop averaging 81 ms, resulting in a total execution time of 2.6 seconds within the MATLAB implementation. Further, during this test, the TFC method converged in about 3 iterations every `fsolve` function call. Lastly, the propagated comparison to GPOPS is provided in Table 7.5, and the Hamiltonian of the two methods is plotted as a function of time in Figure 7.10 to highlight the optimality of both solutions.

Table 7.5: Converged parameters for the TFC and GPOPS-II solution for the max-min-max trajectory test case. The values $||\boldsymbol{r}(t_f)||$, $||\boldsymbol{v}(t_f)||$, and $\lambda_m(t_f)$ were determined by propagating both TFC and GPOPS-II converged solutions in order to have a one-to-one comparison on the accuracy of the converged solutions. Reprinted with permission from [2].

Variable	TFC	GPOPS-II [126]
$L_2[\mathbb{L}]$	$5.654 \cdot 10^{-12}$	–
$L_2[H]$	$8.686 \cdot 10^{-8}$	$6.418 \cdot 10^{-3}$
m_{used} [kg]	275.205	275.206
t_1 [s]	32.418	32.417
t_2 [s]	38.838	38.833
t_f [s]	44.823	44.823
$\|\|\boldsymbol{r}(t_f)\|\|$ [m]	$8.330 \cdot 10^{-10}$	$1.350 \cdot 10^{-1}$
$\|\|\boldsymbol{v}(t_f)\|\|$ [m]	$2.812 \cdot 10^{-11}$	$2.077 \cdot 10^{-2}$
$\lambda_m(t_f)$ [s]	$-8.815 \cdot 10^{-15}$	$-7.354 \cdot 10^{-6}$

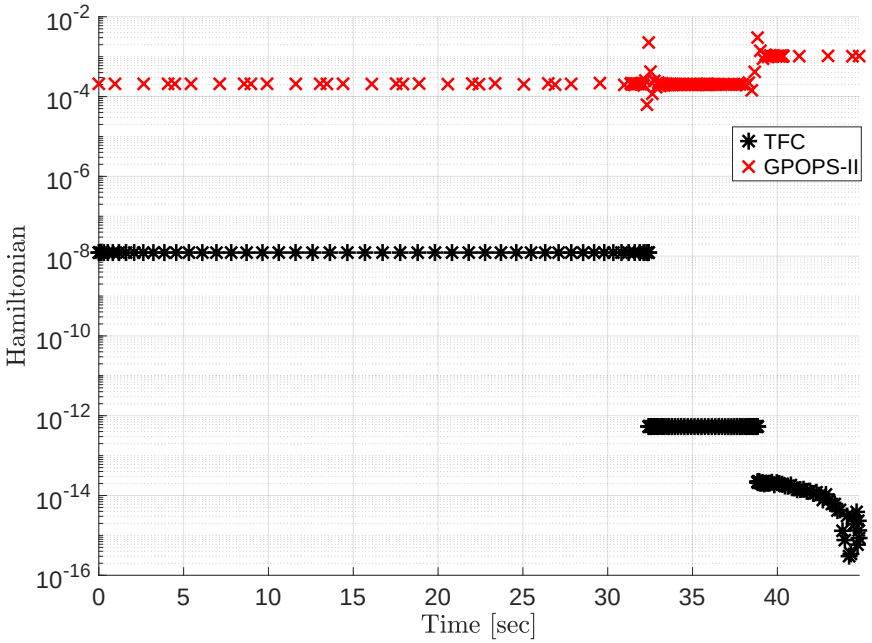


Figure 7.10: Comparison of Hamiltonian for TFC and GPOPS-II converged solutions for the max-min-max trajectory. Reprinted with permission from [2].

7.6 Major findings and conclusions of results

In all, the current implementation of TFC to the fuel-optimal landing problem *cannot* be used in real-time applications. While the accuracy and speed, once written to a compiled language, are acceptable, the algorithm's robustness is the limiting factor. For example, the Monte Carlo test conducted in Chapter 6 could not be run for this algorithm. This and other conclusions are summarized below:

Major takeaways from fuel-optimal landing tests

1. As illustrated in Figure 7.4, the proposed TFC-based algorithm requires an efficient implementation of the iterative least-square together with a root-finding algorithm (e.g., Trust-Region-Dogleg algorithm [127] as implemented in the `fsolve` routine in MATLAB).

2. As reported in the numerical tests presented in Examples 7.1 and 7.2, the `fsolve`

routine iterates for up to 32 times with an upper bound on the execution time of about 2.6 seconds to generate one optimal trajectory, using MATLAB.

- It is known that the MATLAB programming language is about 10 times slower than a C++ executable, which is usually employed to run algorithms on the spacecraft onboard microprocessor.
- Therefore, a computational time gain of at least one order of magnitude is expected, thus making the algorithm attractive for real-time implementations with regards to speed.

3. While the problem was solved with acceptable speed and accuracy, the robustness to the initialization of the times t_1, t_2, and t_f caused convergence issues that are not acceptable for real-time implementation.

   - In this dissertation, two specific cases were solved for the fuel-optimal landing problem but "hand-tuning" was necessary for reliable convergence.
   - Future work could look remove the necessity of the outer-loop; however, from other studies on free final time problems, this problem may be sufficiently complex such that a single-loop least-squares, like that of Chapter 6, will not work.

4. A major concern of this technique may be the trade-off in the amount of work in formulating the problem (and especially computing the constrained expression) compared to other optimization packages. While these terms are formulated analytically, the TFC GitHub [128] provides a framework such after forming the loss vector, the Jacobian terms are computed through automatic differentiation and do not require analytical formulation.

8. SUMMARY AND CONCLUSIONS

The work presented is entitled "A Journey from Theory to Application," because it represents a single route through the dense landscape of the Theory of Functional Connections. I have surely not observed, recorded, and studied all aspects along the way. However, this section is my way of creating a map for future work. Through examples presented, the reader should be familiar with the theory and how it is currently applied. To further aid the reader, the code for most of the problems and examples in this dissertation can be found for free on the TFC GitHub [128]. Moving forward with this section, I look to summarize the major results of this journey along with many potential ideas I have explored.

The main route of this dissertation began with discussing the fundamentals of TFC and the process to derive *constrained expressions*, which are the heart of the method. For a given set of linear constraints, the constrained expression is a functional that represents *all* functions analytically satisfying the constraints, parameterized by the free function $g(x)$. While a method to derive these constrained expressions was provided in the original work on TFC (Reference [3]), this dissertation presents a new formulation that exploits the main structure shared by all constrained expressions. This structure, named the switching-projection form, 1) gives a more intuitive approach to derive constrained expression, 2) provides a straightforward and general framework for the derivation of linear type constraints, 3) allows for a plethora of mathematical insights and associated claims on existence and non-uniqueness, and 4) provides a simple and elegant extension to n-dimensional constrained expressions. In fact, readers interested in the latter point and their application to partial differential equations are directed to Carl Leake's dissertation: "The Multivariate Theory of Functional Connections: An n-Dimensional Constraint Embedding Technique Applied to Partial Differential Equations" [19]. In addition to many detailed examples that derive constrained expressions, the first part also provides preliminary insight for an ad-hoc method allowing inequality constraints and some discussion on over-constrained expressions. While the for-

mer has been implemented in multiple numerical solutions, the latter topic was an academic exploration that spurred from the realization that constrained expressions could also be derived using a weight least-squares approach and allow for more constraints than the number of support functions used in the derivation. In all, this topic was marginally studied, and the usefulness and potential applications are not well understood.

Following the derivation of constrained expression, the second part of this dissertation focused on applying these functionals to the solution of ODEs. Compared to other numerical techniques, the one based on TFC splits the problem into two separate parts: 1) the constraints and 2) the dynamics. As should be clear from the prior sections, the TFC approach allows for the differential equation constraints to be *analytically* embedded in the constrained expressions. In general, this process transforms the differential equation from a constrained optimization problem into an *unconstrained* optimization problem. Next, by using the constrained expression associated with the differential equation constraints, and by 1) defining the free function, $g(x)$ as some know basis with unknown coefficients and 2) discretizing the domain, the problem is again transformed into an algebraic equation that can be solved with any optimization technique, where $\mathbb{L}(\boldsymbol{\xi}) = \mathbf{0}$. While in this dissertation, almost all problems were solved with a linear or nonlinear least-squares, except for free final time problems where `fsolve` or differential evolution algorithms were also used, much fruitful research remains in the study of this technique paired with other numerical schemes. In fact, TFC *is not* by itself a numerical scheme, but rather an analytical technique to reduce the computational overhead of numerically approximating the constraints.

In this part, the approach to solve differential equations was highlighted by numerous examples, starting with a simple initial-value problem and ending with complex cases such as systems of differential equations with terminal algebraic constraints and an unknown domain length. In fact, the latter examples of part two of this dissertation focused on unique corner cases that are relevant in ODEs, including 1) a technique for split domain problems and its application to 2) hybrid systems (differential equations with jumps in dynamics), 3) unknown

domain length, (free final time) problems relevant in optimal control, and 4) the computation of periodic orbits, which constrained expressions provide a simple and elegant approach to tackle. Finally, some examples of the application of over-constrained constrained expressions were provided.

The final part of this dissertation leveraged the prior sections to solve specific aerospace engineering problems, namely terminal descent spacecraft landing on large planetary bodies. These problems were formulated using the indirect method, where the optimal control problem is transformed into a set of differential and algebraic equations that much be solved simultaneously. While this approach is known to produce more optimal solutions than the direct method, the indirect method has a few major drawbacks: 1) the size of the system is doubled with the incorporation of the costates (Lagrange multipliers), and 2) that these costates are highly sensitive to initialization. Therefore, in practice, the indirect method is used less often. Furthermore, while many other numerical approaches exist to solve these types of problems, the motivation to use TFC was that the constrained expressions would provide 1) added robustness to initialization and 2) faster solution speeds. The benefits are not as drastic as first hypothesized for the two problems studies, energy-optimal and fuel-optimal landing. While the TFC solution to the energy-optimal landing did show increased robustness, speed, and accuracy over the spectral method, the solution to the fuel-optimal landing problem lacked robustness and could only be solved for particular cases. In its current state, the TFC algorithm is not quite robust enough. Future improvements could still lead to a technique that could be leveraged to solve trajectories on-board and in real-time by recomputing the optimal trajectory at every computer guidance cycle.

First, the energy-optimal landing problem was analyzed for constant gravity cases. This problem has an analytical feedback solution and was used to evaluate the accuracy of the TFC method versus the spectral method and highlight the benefits of TFC. These results showed that the method built with TFC was more accurate, faster, and more robust to poor initialization. Moving forward, the lessons learned from the energy-optimal problem were

translated to the fuel-optimal landing problem with one distinct difference: in the energy optimal problem, the final time was solved using a single-loop approach where all the TFC parameters were solved for simultaneously. However, it was found that this method only works for a selection of problems[1], including problems where the domain has more than one segment due to the dynamics' switching behavior, as seen in the fuel-optimal landing problem. For this reason, the fuel-optimal landing problem was solved using an inner- and outer-loop approach where the TFC method solved the problem for the fixed time cases, i.e., where the switching times and the final time where specified (t_1, t_2, t_f), and an outer-loop was used to determine the values of these times. The drawback of this is that the algorithm relies on an external optimizer and increases computation time; in this problem, MATLAB's `fsolve` algorithm was unitized. While two specific cases were solved, showing that a solution can be obtained using TFC, the algorithm is not fit for implementation as a real-time controller in its current state. The current issues with this algorithm include 1) the lack of robustness to the initialization of t_1, t_2, and t_f, 2) the inability to solve the problem with a priori knowledge of the control structures, i.e., max, min-max, or max-min-max thrust arcs, and 3) no guarantees on the convergence of the algorithm.

To remedy these concerns, more research needs to be done to identify other optimization techniques that could be used in both the inner- and outer-loops of the algorithms. Additionally, the entirety of this work focuses on solving the problems derived using the indirect method. This leaves the area of direct optimization completely untouched and ripe for exploration.

8.1 Future research

Based on the discussion above, I have chosen to include this section to discuss the current and most fruitful paths in the study of TFC related to the topics covered in this dissertation. In this section, I look to provide key insight into topics most likely to yield widespread

[1]The author has found that this approach also does not work for many problems in trajectory optimization, e.g., minimum-time orbit transfer with a solar sail [129].

improvements to the technique and its applications.

8.1.1 In search of a free function

At the heart of TFC is the constrained expression, which can describe all functions satisfying a set of constraints. The reader should recall that the constrained expression has a free function, $g(x)$, which does not affect the constraints. In numerical applications such as solving differential equations or optimal control problems, the free function must be numerically approximated. Therefore, the representation of the free function is vital in the overall ability to solve problems; however, an in-depth study of this topic is lacking in this dissertation—along with the entire body of research of TFC.

While in this dissertation I mainly focused on the Legendre and Chebyshev orthogonal polynomials, other papers on TFC have looked into using Extreme Learning Machines [20] (mentioned briefly in Chapter 4) and Neural Networks (Deep-TFC) [52] to approximate the free function. However, the work on Deep-TFC has only used fully connected NNs up to this point, and the study of different NN architectures is an active area of research.

According to all of the research conducted to date, orthogonal polynomials for most problems are highly effective and produce solutions near machine-level precision. However, when dealing with complex problems, e.g., Naiver-Stokes equations or PDEs with sharp gradients, the Neural Network approach is more accurate. In general, the only benefit of using ELMs is in the low memory case for the solution of PDEs where the number of basis functions is reduced.

Regardless, there is major promise with the study of particular definitions of $g(x)$ leveraging some a priori knowledge of the problem dynamics. To explain this concept and shed light on a potential area of further research, consider a boundary-value problem in trajectory design that includes many revolutions (or orbits) and dynamics that are not purely Keplerian (there are perturbations due to third-body effects, the sun, etc.). In this case, to accurately determine a solution, the function of $g(x)$ must capture both the periodicity of the orbit and the orbit changes due to perturbation. One idea to solve this problem would be to

use a hybrid basis composed of terms to individually capture the periodic and non-periodic portions individually.

8.1.2 Other optimization schemes

Next, as mentioned in the previous section, TFC *is not a numerical optimization technique*, but rather an analytical method than can be coupled with any optimization scheme that can solve $\mathbb{L}(\boldsymbol{\xi}) = \mathbf{0}$. In this dissertation, along with every paper other than Deep-TFC [52], the optimization scheme used to determine the $\boldsymbol{\xi}$ coefficients of the free function $g(x) = \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{h}(x)$ were based on a simple linear or nonlinear least-squares. This was done for two reasons: 1) the simplicity of the method and the fact that 2) most problems did not require a more complex method. Outside of this dissertation, along with least-squares, Leake [19] studied the use of three other optimizers, including Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm, Adam (a first-order gradient-based optimization of stochastic objective functions), and constrained support vector machines for the solution of differential equations.

However, just as I have discussed with the definition of the free function, an exploration of a wide range of numerical optimization techniques should be the focus of future work in the application of TFC. For the increasing complexity of problems, this will also be a necessity. Above all, TFC can reduce the set of admissible functions and has the potential to speed up many optimization techniques.

One of the potential areas of research is pairing TFC with other optimization schemes within direct optimization. In this dissertation and all other work utilizing TFC, optimal control problems were solved using the indirect method. Similar to how convex optimization is used to convert nonconvex problems into convex problems to assure convergence with NLP solvers, there is potential that the TFC constrained expressions can be used to complement current NLP solvers.

8.2 Additional Literature on TFC

In this section, I look to provide the reader with the most up-to-date capabilities of the theory and the many areas not covered in this dissertation. In all, I hope that the text is a springboard for interested researchers that provides references to all prior work and gives a clear path to more fruitful studies in this area. The list below provides a short description of each paper's contribution along with the links (the PDF file provides clickable links).

8.2.1 Functional Interpolation

- Mortari, D. The Theory of Connections: Connecting Points. *Mathematics* **2017**, 5(4), 57; [**Link**]

  This is the seminal paper on the Theory of Functional Connections. The work presented explores the fundamental idea of functional interpolation using an additive formulation. Constraint interpolation is introduced for points, derivatives, and linear combinations of them. The additive form of functional interpolation is the basis for all subsequent works.

- Johnston, H., Leake, C., Efendiev, Y., and Mortari, D. Selected Applications of the Theory of Connections: A Technique for Analytical Constraint Embedding. *Mathematics* **2019**, 7(6), 537; [**Link**]

  This paper highlights the utility of TFC by introducing various problems that can be solved using this framework, including (1) analytical linear constraint optimization, (2) the brachistochrone problem, (3) over-constrained differential equations; (4) inequality constraints; and (5) triangular domains.

- Mortari, D. and Leake, C. The Multivariate Theory of Connections. *Mathematics* **2019**, 7(3), 296; [**Link**]

  This paper extends the univariate TFC, introduced by Mortari in 2017, to

the multivariate case on rectangular domains with detailed attention to the bivariate case. Although this article's focus is on two-dimensional spaces, the nal section introduces the multivariate TFC, validated by a mathematical proof; this section describes how to write constrained expressions on rectangular domains for an arbitrary number of constraints with arbitrary order derivatives in n-dimensions. In all, this last section was the first iteration of what is later presented in "The Multivariate Theory of Functional Connections: Theory, Proofs, and Application in Partial Differential Equations."

- Wang, Y. and Topputo, F. A Homotopy Method Based on Theory of Functional Connections. *arXiv* **2019**; [**Link**]

  A method for solving zero-finding problems is developed by tracking homotopy paths, which define connecting channels between an auxiliary problem and the objective problem. Current algorithms success relies heavily on empirical knowledge, as the homotopy paths must be selected manually. This work introduces a homotopy method based on TFC. The TFC-based method implicitly defines infinite homotopy paths, from which the most promising ones are selected. A two-layer continuation algorithm is devised, where the first layer tracks the homotopy path by monotonously varying the continuation parameter, while the second layer recovers possible failures and resorts to a TFC representation of the homotopy function. Compared to pseudo-arclength methods, the proposed TFC-based method retains the simplicity of direct continuation while allowing for flexible path switching.

- Leake, C., Johnston, H., and Mortari, D. The Multivariate Theory of Functional Connections: Theory, Proofs, and Application in Partial Differential Equations. *Mathematics* **2020**, 8(8), 1303; [**Link**]

This article exploits constrained expressions' underlying functional structure to ease their derivation and provides mathematical proofs regarding their properties. Furthermore, the extension of the technique to and proofs in n-dimensions is immediate through a recursive application of the univariate formulation.

- Mortari, D. and Arnas, D. Bijective Mapping Analysis to Extend the Theory of Functional Connections to Non-Rectangular 2-Dimensional Domains. *Mathematics* **2020**, 8(9), 1593; [**Link**]

  This work presents an initial analysis of using bijective mappings to extend TFC to non-rectangular, two-dimensional domains. Specically, this manuscript proposes three different mapping techniques: 1) complex mapping, 2) the projection mapping, and 3) polynomial mapping. In that respect, an accurate least-squares approximated inverse mapping is also developed for those mappings with no closed-form inverse.

- Mortari, D. and Furfaro, R. Univariate Theory of Functional Connections Applied to Component Constraints, *Math. Comput. Appl.* **2021**, 26(1), 9; [**Link**]

  This work presents a methodology to derive analytical functionals, with embedded linear constraints among the components of a vector (e.g., coordinates) that is a function a single variable (e.g., time). This work prepares the background necessary for the indirect solution of optimal control problems via the application of the Pontryagin Maximum Principle. The methodology presented is part of the univariate Theory of Functional Connections that has been developed to solve constrained optimization problems. To increase the clarity and practical aspects of the proposed method, the work is mostly presented via examples of applications than via rigorous mathematical denitions and proofs.

8.2.2 Solution of Differential Equations

- Mortari, D. Least-Squares Solution of Linear Differential Equations. *Mathematics* **2017**, 5(4), 48; [**Link**]

  This is the first work utilizing the TFC method to solve linear ordinary differential equations. Herein, the constrained expressions from the TFC framework are used to embed the differential equation constraints, and the free function is defined by Chebyshev and Legendre polynomials. The process converts a differential equation subject to constraints to a linear system of equations that is solved via linear least-squares. The method is thus a unified way to solve initial-, boundary-, and multi-value problems.

- Johnston, H. and Mortari, D. Linear Differential Equations Subject to Relative, Integral, and Infinite Constraints. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2018**, 167, AAS 18-273, pp. 3107-3121, Snowbird, UT, August 19-23, 2018; [**Link**]

  This study looks into extending TFC to incorporate relative, integral, and innite constraints in the solution of differential equations. The results obtained by this method are then compared in terms of speed and accuracy with the solution provided by the `Chebfun` toolbox and are shown to be more accurate with reduced computation time (two orders of magnitude). The new TFC switching-projection form in this dissertation updates the results of this paper.

- Johnston, H. and Mortari, D. Weighted Least-Squares Solutions of Over-Constrained Differential Equations. Proceedings of the *International Academy of Astronautics SciTech Forum* **2018**, AAS 18-812, Moscow, Russia, November 13-15, 2018; [**Link**]

The main purpose of this paper was to explore the ability to derive over-constrained expressions. These constrained expressions satisfy the constraints subject to some relative weighting. They can be used to solve over-constrained differential equations, i.e., it is desired to incorporate more measurements than the order of the differential equation. The contents of this have been refreshed and are included in this dissertation.

- Mortari, D., Johnston, H., and Smith, L. High accuracy least-squares solutions of nonlinear differential equations, *Journal of Computational and Applied Mathematics* **2019**, Vol. 352, pp. 293-307; [**Link**]

  The techniques developed in Mortari's "Least-Squares Solution of Linear Differential Equations" are extended to nonlinear differential equations by implementing a nonlinear least-squares method. This technique is compared to MATLAB's `ode45` and the `Chebfun` package. Additionally, the paper provides the initial scheme to handle long propagation times and is tested on the simple and duffing oscillator.

- Leake, C., Johnston, H., Smith, L., and Mortari, D. Analytically Embedding Differential Equation Constraints into Least Squares Support Vector Machines Using the Theory of Functional Connections. *Mach. Learn. Knowl. Extr.* **2019**, 1(4), 1058-1083; [**Link**]

  This work merges least-squares support vector machines (LS-SVM) with TFC to produced a technique called constrained SVMs (CSVM). In general, TFC is shown to be slightly faster (by an order of magnitude or less) and more accurate (by multiple orders of magnitude) than the LS-SVM and CSVM approaches. Therefore, this technique is not recommended for use. However, this article was an important step towards integrating TFC with machine learning algorithms.

- Johnston, H., Leake, C., and Mortari. D. An Analysis of the Theory of Functional Connections Subject to Inequality Constraints. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2019**, AAS 19-732, Portland, ME, August 11-15, 2019; [**Link**]

  This paper is the first work that incorporates inequality constraints into the TFC framework. The work shows how to extend the original theory to problems subject to equality and inequality constraints for one- and two-dimensions. All of the work in this paper has been updated in this dissertation.

- Johnston, H. and Mortari, D. Least-squares solutions of boundary-value problems in hybrid systems. *arXiv* **2019**; [**Link**]

  This paper looks to apply the mathematical framework of TFC to the solution of boundary-value problems arising from hybrid systems (or a sequence of different differential equations). The approach developed in this work derives an analytical constrained expression for the entire range of a hybrid system, enforcing both the boundary conditions and the continuity conditions across the sequence of differential equations. This reduces the solution space of the hybrid system to only admissible solutions. This technique is widely used throughout this dissertation and enables the solution of problems such as fuel-optimal landing.

- Leake, C. and Mortari, D. Deep Theory of Functional Connections: A New Method for Estimating the Solutions of Partial Differential Equations. *Mach. Learn. Knowl. Extr.* **2020**, 2(1), 37-55; [**Link**]

  This article uses neural networks as the free function in TFC constrained expressions to estimate the solutions of PDEs. Neural networks are not plagued by the same computational curse-of-dimensionality that occurs

when using a linear expansion of basis functions as the free function. Neither are they typically trained via least-squares, which is also memory intensive. This new methodology, called Deep-TFC, is advantageous when estimating the solutions of complex PDEs, such as Navier-Stokes, and has broader impacts outside of differential equation solutions: the article's contents can be used to apply constraints to neural networks, which has multiple applications throughout the machine learning community.

- Johnston, H., Leake, C., and Mortari, D. Least-Squares Solutions of Eighth-Order Boundary Value Problems Using the Theory of Functional Connections. *Mathematics* **2020**, 8(3), 397; [**Link**]

  This paper shows how to obtain highly accurate solutions of eighth-order boundary-value problems of linear and nonlinear ordinary differential equations. The results highlight that the TFC approach does not lose accuracy based on the order of the differential equation and all problems were solved with error on the order of $\mathcal{O}(10^{-13} - 10^{-16})$. In all problems, TFC outperformed current literature by at least four orders of magnitude.

8.2.3 Optimization and Optimal Control

- Mai, T. and Mortari, D. Theory of functional connections applied to nonlinear programming under equality constraints. *arXiv* **2019**; [**Link**]

  This paper introduces an efficient approach to solve quadratic programming problems subject to equality constraints via TFC. This is done without using the traditional Lagrange multipliers approach, and the solution is provided in closed-form for two distinct constrained expressions (satisfying the equality constraints). The unknown optimization variable is then the free vector $\boldsymbol{g}$ introduced by TFC. The solution to the general nonlinear programming problem is obtained by Newtons method. Each iteration

involves the second-order Taylor approximation, starting from an initial vector $\boldsymbol{x}_0$, which is a solution of the equality constraint. Numerical results are provided, which compare the speed and accuracy of this approach to MATLABs `quadprog`. Finally, a convergence analysis of NLP using TFC is provided.

- Drozd, K., Furfaro, R., and Mortari, D. Constrained Energy-Optimal Guidance in Relative Motion via Theory of Functional Connections and Rapidly-Explored Random Trees. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2019**, AAS 19-662, Portland, ME, August 11-15, 2019; [**Link**]

  This is a preliminary study that explores using TFC as a fast and reliable TPBVP solver for kinodynamic sample-based motion planners, like RRTs. A trajectory for a deputy satellite that is energy-optimal, successfully rendezvous with a chief satellite, and is governed by the Clohessy-Wiltshire equations of motion (relative motion) is computed. Within the RRT process, multiple solutions from the many TPBVPs solved via TFC are strung together to form a trajectory that also avoids keep-out-zones.

- Furfaro, R. and Mortari, D. Least-squares Solution of a Class of Optimal Guidance Problems via Theory of Connections, *ACTA Astronautica*, **2020**, Vol. 168, pp. 92-103; [**Link**]

  This paper is the first application of TFC to solve the TPBVPs derived from the indirect method of optimal control. The examples solved in this work include a class of optimal guidance problems, including energy-optimal landing on planetary bodies (where time is fixed for the TFC loop) and fixed-time optimal intercept for a target-interceptor scenario.

- Johnston, H., Schiassi, E., Furfaro, R. and Mortari, D. Fuel-Efficient Powered Descent Guidance on Large Planetary Bodies via Theory of Functional Connections. *J*

Astronaut Sci **2020**; [**Link**]

This paper presents a new approach to solve the fuel-efficient powered descent guidance problem on large planetary bodies with no atmosphere (e.g., Moon or Mars). The problem is formulated using the indirect method, which casts the optimal guidance problem as a system of nonlinear two-point boundary value problems that are solved with TFC. In general, the technique produces solutions with error on the order of $\mathcal{O}(10^{-10})$. The results of this paper are contained in Chapter 7 of this dissertation.

- Schiassi, E., DAmbrosio, A., Johnston, H., Furfaro, R., Curti, F., and Mortari, D. Complete Energy Optimal Landing on Small and Large Planetary Bodies via Theory of Functional Connections. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2020**, AAS 20-557, Lake Tahoe, CA, August 9-13, 2020; [**Link**]

  This paper proposes a unified approach to solve the energy optimal landing on a planetary body (e.g., planet, asteroid, comet, etc.). The method accurately computes the energy optimal landing trajectories, including the optimal time of flight, with a computation time on the order of 10-100 milliseconds, using MATLAB. The algorithms developed from this theory are validated for the landing nal descent phase in Gaspra and Bennu asteroids and Mars.

- Schiassi, E., DAmbrosio, A., Johnston, H., De Florio, M., Drozd, K., Furfaro, R., Curti, F., and Mortari, D. Physics-Informed Extreme Theory of Functional Connections Applied to Optimal Orbit Transfer. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2020**, AAS 20-524, Lake Tahoe, CA, August 9-13, 2020; [**Link**]

  This paper looks to solve a class of trajectory optimization problems using the TFC framework with the free function defined as a single-layer NN.

This technique, referred to as X-TFC, is used to solve the system of differential equations derived through the indirect method of optimal control. The problems studied include the Feldbaum problem, minimum time orbit transfer, and maximum radius orbit transfer.

8.2.4 Astrodynamics

- Johnston, H. and Mortari. D. The Theory of Connections Applied to Perturbed Lamberts Problem. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2018**, AAS 18-282, Snowbird, UT, August 19-23, 2018; [**Link**]

  This paper formulates the perturbed Lambert's problem, a boundary-value problem, in the TFC framework such that the method uses an unperturbed solution as the baseline (or initial guess) and looks to add all perturbations simultaneously with the constrained expression. The results and theory of this paper are dated, and the major issue with this work is that the constrained expressions capturing the perturbations are added to the numerical solution of the unperturbed Lambert's solver. This causes numerical issues and is remedied by only using the unperturbed Lambert's solution as an initial guess to a constrained expression describing the full solution. The updated approach to solve this problem is provided in "Evaluation of transfer costs in the Earth-Moon system using the Theory of Functional Connections."

- Johnston, H. and Mortari. D. Orbit Propagation via the Theory of Functional Connections. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2019**, AAS 19-736, Portland, ME, August 11-15, 2019; [**Link**]

  Spurring from the study of Lambert's problem, this paper investigates the accuracy of TFC applied to the perturbed orbit propagation (initial-value) problem. The method is analyzed for accuracy and convergence behavior

and is compared with the `ode113` propagator and the F & G method. This paper shows that TFC is comparable to other techniques but is better suited for boundary-value problems.

- de Almeida Jr., A. K., Johnston, H., Leake, C., and Mortari. D. Evaluation of transfer costs in the Earth-Moon system using the Theory of Functional Connections. Proceedings of the *AAS/AIAA Astrodynamics Specialist Conference* **2020**, AAS 20-596, Lake Tahoe, CA, August 9-13, 2020; [**Link**]

  This paper uses TFC to analyze the mission design space of the two-impulse maneuver Earth-Moon orbit transfer problem by evaluating ΔV as a function of time of flight and other parameters, like the points of application of the thrusts. Transfers from low-Earth orbit to the L1 Lagrange point and near-Earth orbit to a near-Moon orbit are analyzed as functions of the departure position and the time of ight. Furthermore, the inuence of perturbations due to the gravitational attraction of the Sun is also investigated.

- Johnston, H., Lo, M., and Mortari, D. A Functional Interpolation Method to Compute Period Orbits in the Circular Restricted Three-Body Problem. Proceedings of the *31st AAS/AIAA Space Flight Mechanics Meeting* **2021**, AAS 21-257, Virtual, February 1-4, 2021; [**Link**]

  In this paper, we develop a method to solve for periodic orbits, i.e. Lyapunov and Halo orbits, using a functional interpolation scheme called the Theory of Functional Connections (TFC). Using this technique, a periodic constraint is analytically embedded into the TFC constrained expression. By doing this, the system of differential equations governing the three-body problem is transformed into an unconstrained optimization problem where simple numerical schemes can be used to find a solution, e.g. non-

linear least-squares. This allows for a simpler numerical implementation with comparable accuracy and speed to the traditional differential corrector method.

8.2.5 Transport Theory

- De Florio, M. Accurate Solutions of the Radiative Transfer Problem via Theory of Connections. Thesis for: *MSc in Energy and Nuclear Engineering* **2019**; [**Link**]

  In this thesis, a new approach to solve a class of radiative transfer problems is presented using TFC to solve the linear one-point boundary-value problem derived from the Boltzmann integrodifferential equation for radiative transfer. The proposed algorithm resides in the category of numerical methods for the solution of transport equations and is accurate and suitable for applications in atmospheric science and remote sensing.

- De Florio, M., Schiassi, E., Furfaro, R., Ganapol, B.D., and Mostacci, D. Solutions of Chandrasekhar's Basic Problem in Radiative Transfer via Theory of Functional Connections. *Journal of Quantitative Spectroscopy and Radiative Transfer*, p.107384. **2020**; [**Link**]

  In this paper, Chandrasekhar's problem in radiative transfer is solved using TFC. The method is designed to efficiently and accurately solve the linear boundary-value problem arising from the angular discretization of the integrodifferential Boltzmann equation for radiative transfer. The proposed algorithm falls under the category of numerical methods for the solution of radiative transfer equations. The accuracy of this new method is tested by benchmark comparison for Mie and Haze L scattering laws.

8.2.6 Physics-Informed Neural Networks

- Schiassi, E., Leake, C., De Florio, M., Johnston, H., Furfaro, R., and Mortari, D. Extreme Theory of Functional Connections: A Physics-Informed Neural Network Method

for Solving Parametric Differential Equations. *arXiv* **2020**; [**Link**]

This article uses a single layer neural network (NN), or more precisely an Extreme Learning Machine (ELM), as the free function in TFC constrained expressions to estimate the solutions of DEs. The results show that X-TFC achieves high accuracy with low computational time but is never more accurate than the original TFC formulation with orthogonal polynomials for simple problems, nor more accurate than Deep-TFC for complex problems.

- Schiassi, E., D'Ambrosio, A., De Florio, M., Furfaro, R., and Curti, F. Physics-Informed Extreme Theory of Functional Connections Applied to Data-Driven Parameters Discovery of Epidemiological Compartmental Models. *arXiv* **2020**; [**Link**]

  This paper utilizes the X-TFC framework, which combines TFC with the Physics-Informed Neural Networks (PINN) framework for data-driven parameters discovery of problems modeled via ordinary differential equations (ODEs). In particular, this work focuses on the capability of X-TFC in solving inverse problems to estimate the parameters governing the epidemiological compartmental models via a deterministic approach. The epidemiological compartmental models treated in this work are Susceptible Infectious Recovered (SIR), Susceptible Exposed Infectious Recovered (SEIR), and Susceptible Exposed Infectious Recovered Susceptible (SEIRS). The results show that these problems can be accurately solved with low computational times under the influence of unperturbed and perturbed data.

REFERENCES

[1] H. Johnston and D. Mortari, "Least-squares solutions of boundary-value problems in hybrid systems," 2021.

[2] H. Johnston, E. Schiassi, R. Furfaro, and D. Mortari, "Fuel-efficient powered descent guidance on large planetary bodies via theory of functional connections," *The Journal of the Astronautical Sciences*, vol. 67, no. 4, pp. 1521–1552, 2020.

[3] D. Mortari, "The Theory of Connections: Connecting Points," *MDPI Mathematics*, vol. 5, no. 4, 2017.

[4] C. Leake, H. Johnston, and D. Mortari, "The multivariate theory of functional connections: Theory, proofs, and application in partial differential equations," *Mathematics*, vol. 8, no. 8, 2020.

[5] E. Waring, "Problems concerning interpolations. by edward waring, m. d. f. r. s. and of the institute of bononia, lucasian professor of mathematics in the university of cambridge," *Philosophical Transactions of the Royal Society of London*, vol. 69, pp. 59–67, 1779.

[6] N. Lam, "Spatial interpolation methods: a review," *American Cartographer*, vol. 10, pp. 129–149, 01 1983.

[7] J. Li and A. D. Heap, "A review of comparative studies of spatial interpolation methods in environmental sciences: Performance and impact factors," *Ecological Informatics*, vol. 6, no. 3, pp. 228 – 241, 2011.

[8] T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: interpolation methods in medical image processing," *IEEE Transactions on Medical Imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[9] J. Steffensen, *Interpolation*. Chelsea Publishing Company, 1950.

[10] D. K. Hoffman, G. W. Wei, D. S. Zhang, and D. J. Kouri, "Interpolating distributed approximating functionals," *Phys. Rev. E*, vol. 57, pp. 6152–6160, May 1998.

[11] G. Wei, H. Wang, D. J. Kouri, M. Papadakis, I. A. Kakadiaris, and D. K. Hoffman, "On the mathematical properties of distributed approximating functionals," *Journal of Mathematical Chemistry*, vol. 30, no. 1, pp. 83–107, 2001.

[12] H. Johnston and D. Mortari, "Linear differential equations subject to relative, integral, and infinite constraints," in *2018 AAS/AIAA Astrodynamics Specialist Conference Snowbird, UT, August 19–23, 2018*, AAS/AIAA, 2018.

[13] D. Mortari and R. Furfaro, "Theory of connections applied to first-order system of ordinary differential equations subject to component constraints," in *2018 AAS/AIAA Astrodynamics Specialist Conference Snowbird, UT, August 19–23, 2018*, vol. 167, pp. 3041–3056, AAS/AIAA, 2018.

[14] V. M. F. B.Sc and M. S. W. Skan, "Solutions of the boundary-layer equations," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 12, p. 865896, Nov 1931.

[15] H. Johnston, C. Leake, Y. Efendiev, and D. Mortari, "Selected Applications of the Theory of Connections: A Technique for Analytical Constraints Embedding," *MDPI Mathematics*, vol. 7, no. 6, 2019.

[16] H. Johnston, C. Leake, and D. Mortari, "An analysis of the theory of functional connections subject to inequality constraints," in *2019 AAS/AIAA Astrodynamics Specialist Conference Portland, ME, August 11–15, 2019*, AAS/AIAA, 2019.

[17] D. Mortari, "Least-Squares Solution of Linear Differential Equations," *MDPI Mathematics*, vol. 5, no. 4, 2017.

[18] D. Mortari, H. Johnston, and L. Smith, "High Accuracy Least-squares Solutions of Nonlinear Differential Equations," *Journal of Computational and Applied Mathematics*, vol. 352, pp. 293 – 307, 2019.

[19] C. Leake, "The Multivariate Theory of Functional Connections: An n-Dimensional Constraint Embedding Technique Applied to Partial Differential Equations." PhD Dissertation, Texas A&M University, 2021.

[20] E. Schiassi, C. Leake, M. De Florio, H. Johnston, R. Furfaro, and D. Mortari, "Extreme theory of functional connections: A physics-informed neural network method for solving parametric differential equations," *arXiv preprint arXiv:2005.10632*, 2020.

[21] F. Schwarz, *Algorithmic lie theory for solving ordinary differential equations*. Chapman & Hall/CRC, 01 2007.

[22] A. Polyanin and V. Zaitsev, *Handbook of Exact Solutions for Ordinary Differential Equations*. Chapman & Hall/CRC, 10 2002.

[23] J. Dormand and P. Prince, "A Family of Embedded Runge-Kutta Formulae," *J. Comp. Appl. Math.*, vol. 6, pp. 19–26, 1980.

[24] L. F. Shampine and M. W. Reichelt, "The matlab ode suite," *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997.

[25] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[26] J. Jackson, "Note on the numerical integration of $d^2x/dt^2 = f(x,t)$," *Monthly Not. Roy. Astron. Soc.*, vol. 84, pp. 602–6067, 1924.

[27] H. Jeffreys and B. Jeffreys, *The Gauss-Jackson Method*, vol. 84. Cambridge University Press, 1988.

[28] M. M. Berry and L. M. Healy, "Implementation of Gauss-Jackson integration for orbit propagation," The Journal of the Astronautical Sciences, vol. 52, no. 3, pp. 351–357, 2004.

[29] M. M. Berry, *A variable-step double-integration multi-step integrator*. PhD thesis, Virginia Tech, 2004.

[30] X. Bai and J. L. Junkins, "Modified Chebyshev-Picard Iteration Methods for Orbit Propagation," The Journal of the Astronautical Sciences, vol. 58, no. 4, pp. 583–613, 2011.

[31] J. L. Junkins, A. B. Younes, R. Woollands, and X. Bai, "Picard Iteration, Chebyshev Polynomials, and Chebyshev Picard Methods: Application in Astrodynamics," The Journal of the Astronautical Sciences, vol. 60, pp. 623–653, December 2015.

[32] J. Reed, A. B. Younes, B. Macomber, J. L. Junkins, and D. J. Turner, "State Transition Matrix for Perturbed Orbital Motion using Modified Chebyshev Picard Iteration," *The Journal of the Astronautical Sciences*, vol. 6, pp. 148–167, 2015. doi: 10.1007/s40295-015-0051-3.

[33] L. Fox and I. Parker, *Chebyshev Polynomials in Numerical Analysis*. London, UK: Oxford University Press, 1972.

[34] X. Bai, *Modified Chebyshev-Picard Iteration Methods for Solution of Initial Value and Boundary Value Problems*. PhD thesis, Texas A&M University, 2010.

[35] T. A. Elgohary, L. Dong, J. L. Junkins, and S. N. Alturi, "Time Domain Inverse Problems in Nonlinear Systems Using Collocation & Radial Basis Functions," Computer Modeling in Engineering & Sciences, vol. 100, no. 1, pp. 59–84, 2014.

[36] *An RBF-Collocation Algorithm for Orbit Propagation*, 2015.

[37] K. Wright, "Chebyshev Collocation Methods for Ordinary Differential Equations," *The Computer Journal*, vol. 6, no. 1, pp. 358–365, 1964.

[38] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*. Society for Industrial and Applied Mathematics, 1977.

[39] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[40] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[41] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, p. 879892, 2006.

[42] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 911–917, 1995.

[43] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, p. 143195, 1999.

[44] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *arXiv preprint arXiv:1907.04502*, 2019.

[45] Y. Yang, M. Hou, and J. Luo, "A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods," *Advances in Difference Equations*, vol. 2018, no. 1, p. 469, 2018.

[46] H. Sun, M. Hou, Y. Yang, T. Zhang, F. Weng, and F. Han, "Solving Partial Differential Equation Based on Bernstein Neural Network and Extreme Learning Machine Algorithm," *Neural Processing Letters*, vol. 50, no. 2, pp. 1153–1172, 2019.

[47] S. Mall and S. Chakraverty, "Single Layer Chebyshev Neural Network Model for Solving Elliptic Partial Differential Equations," *Neural Processing Letters*, vol. 45, no. 3, pp. 825–840, 2017.

[48] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, " Extreme learning machine: Theory and applications ," *Neurocomputing*, vol. 70, pp. 489–501, May 2006.

[49] A. Gil, J. Segura, and N. Temme, *Numerical Methods for Special Functions*. Society for Industrial and Applied Mathematics, 1 2007.

[50] C. Lanczos, *Applied Analysis*. New York: Dover Publications, Inc., 1957.

[51] N. Liu, *Theory and Applications and Legendre Polynomials and Wavelets*. University of Toledo, 2008.

[52] C. Leake and D. Mortari, "Deep theory of functional connections: A new method for estimating the solutions of partial differential equations," *Machine Learning and Knowledge Extraction*, vol. 2, no. 1, pp. 37–55, 2020.

[53] C. Lanczos, *Applied Analysis*, p. 504. New York: Dover Publications, Inc., 1957.

[54] K. Wright, "Chebyshev Collocation Methods for Ordinary Differential Equations.," *The Computer Journal*, vol. 6, no. 1, pp. 358–365, 1964. Issue 4.

[55] Edwards, C., et al.(Eds.), *Advances in Variable Structure and Sliding Mode Control*, ch. ..., p. ... Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[56] D. D. Morrison, J. D. Riley, and J. F. Zancanaro, "Multiple shooting method for two-point boundary value problems," *Commun. ACM*, vol. 5, pp. 613–614, Dec. 1962.

[57] G. J. Lastman, "A shooting method for solving two-point boundary-value problems arising from non-singular bang-bang optimal control problems," *International Journal of Control*, vol. 27, no. 4, pp. 513–524, 1978.

[58] M. Osborne, "On shooting methods for boundary value problems," *Journal of Mathematical Analysis and Applications*, vol. 27, no. 2, pp. 417 – 433, 1969.

[59] S. M. Filipov, I. D. Gospodinov, and I. Faragó, "Replacing the finite difference methods for nonlinear two-point boundary value problems by successive application of the linear

shooting method," *Journal of Computational and Applied Mathematics*, vol. 358, pp. 46 – 60, 2019.

[60] R. Weiss, "The convergence of shooting methods," *BIT Numerical Mathematics*, vol. 13, pp. 470–475, Dec 1973.

[61] P. Marzulli and G. Gheri, "Estimation of the global discretization error in shooting methods for linear boundary value problems," *Journal of Computational and Applied Mathematics*, vol. 28, pp. 309 – 314, 1989.

[62] P. Marzulli, "Global error estimates for the standard parallel shooting method," *Journal of Computational and Applied Mathematics*, vol. 34, no. 2, pp. 233 – 241, 1991.

[63] J. N. Reddy, "An Introduction to the Finite Element Method," *Journal of Pressure Vessel Technology*, vol. 111, pp. 348–349, 08 1989.

[64] R. W. Farquhar, *The Control and Use of Libration-Point Satellites*. PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford University, Stanford, California, 1968.

[65] J. V. Breakwell and J. V. Brown, "The 'halo'family of 3-dimensional periodic orbits in the earth-moon restricted 3-body problem," *Celestial mechanics*, vol. 20, no. 4, pp. 389–404, 1979.

[66] K. Connor Howell, "Three-dimensional, periodic, 'halo'orbits," *Celestial mechanics*, vol. 32, no. 1, pp. 53–71, 1984.

[67] D. L. Richardson, "Analytic construction of periodic orbits about the collinear points," *Celestial mechanics*, vol. 22, no. 3, pp. 241–253, 1980.

[68] G. Singh, A. M. SanMartin, and E. C. Wong, "Guidance and control design for powered descent and landing on mars," in *2007 IEEE Aerospace Conference*, pp. 1–8, IEEE, 2007.

[69] A. V. Rao and W. W. Hager, "Mesh-generation method for real-time optimal control using adaptive gaussian quadrature collocation," in *2018 AIAA Guidance, Navigation, and Control Conference*, p. 0848, 2018.

[70] M. E. Dennis, W. W. Hager, and A. V. Rao, "Computational method for optimal guidance and control using adaptive gaussian quadrature collocation," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 9, pp. 2026–2041, 2019.

[71] I. M. Ross and F. Fahroo, "Issues in the real-time computation of optimal control," *Mathematical and computer modelling*, vol. 43, no. 9-10, pp. 1172–1188, 2006.

[72] I. M. Ross, P. Sekhavat, A. Fleming, and Q. Gong, "Optimal feedback control: foundations, examples, and experimental results for a new approach," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 2, pp. 307–321, 2008.

[73] C. L. Darby, W. W. Hager, and A. V. Rao, "An hp-adaptive pseudospectral method for solving optimal control problems," *Optimal Control Applications and Methods*, vol. 32, no. 4, pp. 476–502, 2011.

[74] F. Fahroo and I. M. Ross, "Direct trajectory optimization by a chebyshev pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 160–166, 2002.

[75] I. M. Ross and F. Fahroo, "Pseudospectral knotting methods for solving nonsmooth optimal control problems," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 3, pp. 397–405, 2004.

[76] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming," *Mathematical programming*, vol. 89, no. 1, pp. 149–185, 2000.

[77] S. Josselyn and I. M. Ross, "Rapid verification method for the trajectory optimization of reentry vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 3, pp. 505–508, 2003.

[78] K. F. Graham and A. V. Rao, "Minimum-time trajectory optimization of multiple revolution low-thrust earth-orbit transfers," *Journal of Spacecraft and Rockets*, vol. 52, no. 3, pp. 711–727, 2015.

[79] A. T. Miller and A. V. Rao, "Rapid ascent-entry vehicle mission optimization using hp-adaptive gaussian quadrature collocation," in *AIAA Atmospheric Flight Mechanics Conference*, p. 0249, 2017.

[80] X. Jiang, S. Li, and R. Furfaro, "Integrated guidance for mars entry and powered descent using reinforcement learning and pseudospectral method," *Acta Astronautica*, vol. 163, pp. 114–129, 2019.

[81] B. Acikmese and S. R. Ploen, "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[82] L. Blackmore, B. Acikmese, and D. P. Scharf, "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing using Convex Optimization," *Journal of guidance, control, and dynamics*, vol. 33, no. 4, pp. 1161–1171, 2010.

[83] Z. Wang and M. J. Grant, "Constrained trajectory optimization for planetary entry via sequential convex programming," in *AIAA Atmospheric Flight Mechanics Conference*, p. 3241, 2016.

[84] Z. Wang and M. J. Grant, "Autonomous entry guidance for hypersonic vehicles by convex optimization," *Journal of Spacecraft and Rockets*, vol. 55, no. 4, pp. 993–1006, 2018.

[85] K. Zhang, S. Yang, and F. Xiong, "Rapid ascent trajectory optimization for guided rockets via sequential convex programming," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, p. 0954410019830268, 2019.

[86] Z. Wang and M. J. Grant, "Minimum-fuel Low-thrust Transfers for Spacecraft: A Convex Approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 5, pp. 2274–2290, 2018.

[87] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control : Optimization, Estimation, and Control*. Hemisphere Pub. Corp., New York, rev. printing. ed., 1975.

[88] P. Lu, "Introducing computational guidance and control," 2017.

[89] J. T. Betts and W. P. Huffman, "Mesh refinement in direct transcription methods for optimal control," *Optimal Control Applications and Methods*, vol. 19, no. 1, pp. 1–21, 1998.

[90] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*, pp. 65–93, Springer, 2006.

[91] A. Schwartz and E. Polak, "Consistent approximations for optimal control problems based on runge–kutta integration," *SIAM Journal on Control and Optimization*, vol. 34, no. 4, pp. 1235–1269, 1996.

[92] W. W. Hager, "Runge-kutta methods in optimal control and the transformed adjoint system," *Numerische Mathematik*, vol. 87, no. 2, pp. 247–282, 2000.

[93] G. Reddien, "Collocation at gauss points as a discretization in optimal control," *SIAM Journal on Control and Optimization*, vol. 17, no. 2, pp. 298–306, 1979.

[94] A. L. Herman and B. A. Conway, "Direct optimization using collocation based on high-order gauss-lobatto quadrature rules," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 3, pp. 592–599, 1996.

[95] J. Vlassenbroeck and R. Van Dooren, "A chebyshev technique for solving nonlinear optimal control problems," *IEEE transactions on automatic control*, vol. 33, no. 4, pp. 333–340, 1988.

[96] G. Elnagar, M. A. Kazemi, and M. Razzaghi, "The pseudospectral legendre method for discretizing optimal control problems," *IEEE transactions on Automatic Control*, vol. 40, no. 10, pp. 1793–1796, 1995.

[97] F. Fahroo and I. M. Ross, "Pseudospectral methods for infinite-horizon nonlinear optimal control problems," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 4, pp. 927–936, 2008.

[98] D. Garg, M. A. Patterson, C. Francolin, C. L. Darby, G. T. Huntington, W. W. Hager, and A. 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*, vol. 49, no. 2, pp. 335–358, 2011.

[99] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[100] Q. Gong, I. M. Ross, W. Kang, and F. Fahroo, "Connections between the covector mapping theorem and convergence of pseudospectral methods for optimal control," *Computational Optimization and Applications*, vol. 41, no. 3, pp. 307–335, 2008.

[101] Q. Gong, F. Fahroo, and I. M. Ross, "Spectral algorithm for pseudospectral methods in optimal control," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 460–471, 2008.

[102] W. Kang, Q. Gong, I. M. Ross, and F. Fahroo, "On the convergence of nonlinear optimal control using pseudospectral methods for feedback linearizable systems," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 17, no. 14, pp. 1251–1277, 2007.

[103] W. Kang, I. M. Ross, and Q. Gong, "Pseudospectral optimal control and its convergence theorems," in *Analysis and design of nonlinear control systems*, pp. 109–124, Springer, 2008.

[104] W. W. Hager, H. Hou, and A. V. Rao, "Convergence rate for a gauss collocation method applied to unconstrained optimal control," *Journal of Optimization Theory and Applications*, vol. 169, no. 3, pp. 801–824, 2016.

[105] W. W. Hager, H. Hou, S. Mohapatra, A. V. Rao, and X.-S. Wang, "Convergence rate for a radau hp collocation method applied to constrained optimal control," *Computational Optimization and Applications*, vol. 74, no. 1, pp. 275–314, 2019.

[106] I. M. Ross and F. Fahroo, "Legendre pseudospectral approximations of optimal control problems," in *New trends in nonlinear dynamics and control and their applications*, pp. 327–342, Springer, 2003.

[107] G. T. Huntington and A. V. Rao, "Comparison of global and local collocation methods for optimal control," *Journal of guidance, control, and dynamics*, vol. 31, no. 2, pp. 432–436, 2008.

[108] Y. M. Agamawi, W. W. Hager, and A. V. Rao, "Mesh refinement method for solving bang-bang optimal control problems using direct collocation," in *AIAA Scitech 2020 Forum*, p. 0378, 2020.

[109] M. A. Patterson and A. V. Rao, "Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1, pp. 1–37, 2014.

[110] A. Peloni, A. V. Rao, and M. Ceriotti, "Automated trajectory optimizer for solar sailing (atoss)," *Aerospace Science and Technology*, vol. 72, pp. 465–475, 2018.

[111] I. M. Ross and M. Karpenko, "A review of pseudospectral optimal control: From theory to flight," *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.

[112] A. V. Rao, "Trajectory optimization: a survey," in *Optimization and optimal control in automotive systems*, pp. 3–21, Springer, 2014.

[113] H. B. Keller, *Numerical solution of two point boundary value problems*, vol. 24. SIaM, 1976.

[114] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, vol. 12. Springer Science & Business Media, 2013.

[115] S. Oh and R. Luus, "Use of orthogonal collocation method in optimal control problems," *International Journal of Control*, vol. 26, no. 5, pp. 657–673, 1977.

[116] F. Fahroo and I. Ross, "Trajectory optimization by indirect spectral collocation methods," in *Astrodynamics specialist conference*, p. 4028, 2000.

[117] F. Fahroo and I. M. Ross, "Advances in pseudospectral methods for optimal control," in *AIAA guidance, navigation and control conference and exhibit*, p. 7309, 2008.

[118] C. D'Souza, *An optimal guidance law for planetary landing*, pp. 1376–1381. American Institute of Aeronautics and Astronautics, 1997.

[119] R. Furfaro, S. Selnick, M. Cupples, and M. Cribb, "Non-linear sliding guidance algorithms for precision lunar landing," *Advances in the Astronautical Sciences*, vol. 140, pp. 945 – 964, 2011.

[120] B. Ebrahimi, M. Bahrami, and J. Roshanian, "Optimal sliding-mode guidance with terminal velocity constraint for fixed-interval propulsive maneuvers," *Acta Astronautica*, vol. 62, no. 10, pp. 556 – 562, 2008.

[121] Y. Guo, M. Hawkins, and B. Wie, "Applications of generalized zero-effort-miss/zero-effort-velocity feedback guidance algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 3, pp. 810–820, 2013.

[122] E. Schiassi, A. D'Ambrosio, H. Johnston, R. Furfaro, F. Curti, and D. Mortari, "Complete energy optimal landing on small and large planetary bodies via theory of functional connections," in *AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, August 9-13, 2020*, AAS/AIAA, 2020.

[123] P. Lu, "Propellant-Optimal Powered Descent Guidance," *Journal of Guidance, Control, and Dynamics*, vol. 41, April 2018.

[124] D. F. Lawden, *Optimal trajectories for space navigation*, vol. 3. Butterworths, 1963.

[125] MATLAB, *version 9.6.0 (R2019a)*. Natick, Massachusetts: The MathWorks Inc., 2019.

[126] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Trans. Math. Softw.*, vol. 41, pp. 1:1–1:37, Oct. 2014.

[127] A. Conn, N. Gould, and P. L. Toint, "Trust-region methods. mps-siam series on optimization siam and mps," *Society for Industrial and Applied Mathematics: Philadelphia, PA, USA*, 2000.

[128] C. Leake and H. Johnston, "TFC: A Functional Interpolation Framework," 2020.

[129] E. Schiassi, A. D'Ambrosio, H. Johnston, M. D. Florio, K. Drozd, R. Furfaro, F. Curti, and D. Mortari, "Physics-informed extreme theory of functional connections applied to optimal orbit transfer," in *AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, August 9-13, 2020*, AAS/AIAA, 2020.

[130] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, and S. Wanderman-Milne, "JAX: composable transformations of Python+NumPy programs," 2018.

[131] R. Frostig, M. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," in *SysML Conference*, 2018.

APPENDIX A

ORTHOGONAL BASIS FUNCTIONS

Since the proposed method uses a set of basis functions, a summary of the candidate orthogonal polynomial basis functions is provided.

A.1 Chebyshev

Chebyshev Orthogonal Polynomials (CP) of the first kind, $T_k(z)$, are defined on the domain $z \in [-1, +1]$ and are generated using the recursive function,

$$T_{k+1} = 2\,z\,T_k - T_{k-1} \qquad \text{starting from:} \begin{cases} T_0 = & 1 \\ T_1 = & z \end{cases} \tag{A.1}$$

All derivatives of CP can be computed recursively, starting from

$$\frac{\mathrm{d}T_0}{\mathrm{d}z} = 0, \quad \frac{\mathrm{d}T_1}{\mathrm{d}z} = 1 \qquad \text{and} \qquad \frac{\mathrm{d}^d T_0}{\mathrm{d}z^d} = \frac{\mathrm{d}^d T_1}{\mathrm{d}z^d} = 0 \quad (\forall\, d > 1),$$

while the subsequent derivatives of Equation (A.1) are given for $k \geq 1$,

$$\begin{aligned}
\frac{\mathrm{d}T_{k+1}}{\mathrm{d}z} &= 2\left(T_k + z\,\frac{\mathrm{d}T_k}{\mathrm{d}z}\right) - \frac{\mathrm{d}T_{k-1}}{\mathrm{d}z} \\
\frac{\mathrm{d}^2 T_{k+1}}{\mathrm{d}z^2} &= 2\left(2\,\frac{\mathrm{d}T_k}{\mathrm{d}z} + z\,\frac{\mathrm{d}^2 T_k}{\mathrm{d}z^2}\right) - \frac{\mathrm{d}^2 T_{k-1}}{\mathrm{d}z^2} \\
&\vdots \qquad\qquad\qquad \vdots \qquad\qquad \vdots \\
\frac{\mathrm{d}^d T_{k+1}}{\mathrm{d}z^d} &= 2\left(d\,\frac{\mathrm{d}^{d-1} T_k}{\mathrm{d}z^{d-1}} + z\,\frac{\mathrm{d}^d T_k}{\mathrm{d}z^d}\right) - \frac{\mathrm{d}^d T_{k-1}}{\mathrm{d}z^d}; \quad (\forall\, d \geq 1).
\end{aligned}$$

In particular,

$$T_k(-1) = (-1)^k, \quad \left.\frac{\mathrm{d}T_k}{\mathrm{d}z}\right|_{z=-1} = (-1)^{k+1}\,k^2, \quad \left.\frac{\mathrm{d}^2 T_k}{\mathrm{d}z^2}\right|_{z=-1} = (-1)^k\,\frac{k^2\,(k^2-1)}{3}$$

and

$$T_k(1) = 1, \qquad \left.\frac{\mathrm{d}T_k}{\mathrm{d}z}\right|_{z=1} = k^2, \qquad \left.\frac{\mathrm{d}^2T_k}{\mathrm{d}z^2}\right|_{z=1} = \frac{k^2\,(k^2-1)}{3}.$$

A.2 Legendre

Legendre Orthogonal Polynomials (LeP), $L_k(z)$, are defined on the domain $z \in [-1, +1]$ and are generated using the recursive function,

$$L_{k+1} = \frac{2k+1}{k+1}\,z\,L_k - \frac{k}{k+1}\,L_{k-1} \qquad \text{starting:} \begin{cases} L_0 = & 1 \\ L_1 = & z \end{cases} \tag{A.2}$$

All derivatives of LeP can be computed recursively, starting from

$$\frac{\mathrm{d}L_0}{\mathrm{d}z} = 0, \quad \frac{\mathrm{d}L_1}{\mathrm{d}z} = 1 \qquad \text{and} \qquad \frac{\mathrm{d}^dL_0}{\mathrm{d}z^d} = \frac{\mathrm{d}^dL_1}{\mathrm{d}z^d} = 0 \quad (\forall\, d > 1),$$

while the subsequent derivatives of Equation (A.2) for $k \geq 1$, can be computed in cascade,

$$\begin{array}{rcll} \dfrac{\mathrm{d}L_{k+1}}{\mathrm{d}z} & = & \dfrac{2k+1}{k+1}\left(L_k + z\dfrac{\mathrm{d}L_k}{\mathrm{d}z}\right) & -\dfrac{k}{k+1}\dfrac{\mathrm{d}L_{k-1}}{\mathrm{d}z} \\ \dfrac{\mathrm{d}^2L_{k+1}}{\mathrm{d}z^2} & = & \dfrac{2k+1}{k+1}\left(2\dfrac{\mathrm{d}L_k}{\mathrm{d}z} + z\dfrac{\mathrm{d}^2L_k}{\mathrm{d}z^2}\right) & -\dfrac{k}{k+1}\dfrac{\mathrm{d}^2L_{k-1}}{\mathrm{d}z^2} \\ \vdots & & \vdots & \vdots \\ \dfrac{\mathrm{d}^dL_{k+1}}{\mathrm{d}z^d} & = & \dfrac{2k+1}{k+1}\left(d\dfrac{\mathrm{d}^{d-1}L_k}{\mathrm{d}z^{d-1}} + z\dfrac{\mathrm{d}^dL_k}{\mathrm{d}z^d}\right) & -\dfrac{k}{k+1}\dfrac{\mathrm{d}^dL_{k-1}}{\mathrm{d}z^d}; \quad (\forall\, d \geq 1). \end{array}$$

A.3 Laguerre

Laguerre Orthogonal Polynomials (LaP), $L_k(z)$, are defined on the domain $z \in [0, \infty)$ and are generated using the recursive function,

$$L_{k+1}(z) = \frac{2k+1-z}{k+1}\,L_k(z) - \frac{k}{k+1}\,L_{k-1}(z) \qquad \text{starting:} \begin{cases} L_0 = & 1 \\ L_1 = & 1-z \end{cases}$$

All derivatives of LaP can be computed recursively, starting from

$$\frac{\mathrm{d}L_0}{\mathrm{d}z} = 0, \quad \frac{\mathrm{d}L_1}{\mathrm{d}z} = -1 \qquad \text{and} \qquad \frac{\mathrm{d}^d L_0}{\mathrm{d}z^d} = \frac{\mathrm{d}^d L_1}{\mathrm{d}z^d} = 0 \quad (\forall\, d > 1),$$

then

$$\begin{aligned}
\frac{\mathrm{d}L_{k+1}}{\mathrm{d}z} &= \frac{2k+1-z}{k+1}\frac{\mathrm{d}L_k}{\mathrm{d}z} - \frac{1}{k+1}L_k - \frac{k}{k+1}\frac{\mathrm{d}L_{k-1}}{\mathrm{d}z} \\
\frac{\mathrm{d}^2 L_{k+1}}{\mathrm{d}z^2} &= \frac{2k+1-z}{k+1}\frac{\mathrm{d}^2 L_k}{\mathrm{d}z^2} - \frac{2}{k+1}\frac{\mathrm{d}L_k}{\mathrm{d}z} - \frac{k}{k+1}\frac{\mathrm{d}^2 L_{k-1}}{\mathrm{d}z^2} \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
\frac{\mathrm{d}^d L_{k+1}}{\mathrm{d}z^d} &= \frac{2k+1-z}{k+1}\frac{\mathrm{d}^d L_k}{\mathrm{d}z^d} - \frac{d}{k+1}\frac{\mathrm{d}^{d-1} L_k}{\mathrm{d}z^{d-1}} - \frac{k}{k+1}\frac{\mathrm{d}^d L_{k-1}}{\mathrm{d}z^d}
\end{aligned}$$

A.4 Hermite

There are two Hermite Orthogonal Polynomials (HP), the probabilists, indicated by $E_k(z)$ defined on the domain $z \in (-\infty, \infty)$, and the physicists, indicated by $H_k(z)$ also defined on the domain $z \in (-\infty, \infty)$. They both are generated using recursive functions.

The probabilistists are defined as

$$E_{k+1}(z) = z\,E_k(z) - kE_{k-1}(z) \qquad \text{starting:} \begin{cases} E_0(z) = 1 \\ E_1(z) = z \end{cases}$$

All derivatives can be computed recursively, starting from

$$\frac{\mathrm{d}E_0}{\mathrm{d}z} = 0, \quad \frac{\mathrm{d}E_1}{\mathrm{d}z} = 1 \qquad \text{and} \qquad \frac{\mathrm{d}^d E_0}{\mathrm{d}z^d} = \frac{\mathrm{d}^d E_1}{\mathrm{d}z^d} = 0 \quad (\forall\, d > 1),$$

then

$$\begin{aligned}
\frac{\mathrm{d}E_{k+1}}{\mathrm{d}z} &= E_k + z\frac{\mathrm{d}E_k}{\mathrm{d}z} - k\frac{\mathrm{d}E_{k-1}}{\mathrm{d}z} \\
\frac{\mathrm{d}^2E_{k+1}}{\mathrm{d}z^2} &= 2\frac{\mathrm{d}E_k}{\mathrm{d}z} + z\frac{\mathrm{d}^2E_k}{\mathrm{d}z^2} - k\frac{\mathrm{d}^2E_{k-1}}{\mathrm{d}z^2} \\
\vdots &\quad\quad \vdots \\
\frac{\mathrm{d}^dE_{k+1}}{\mathrm{d}z^d} &= d\frac{\mathrm{d}^{d-1}E_k}{\mathrm{d}z^{d-1}} + z\frac{\mathrm{d}^dE_k}{\mathrm{d}z^d} - k\frac{\mathrm{d}^dE_{k-1}}{\mathrm{d}z^d}
\end{aligned}$$

The physicists are defined as

$$H_{k+1}(z) = 2z\,H_k(z) - 2k\,H_{k-1}(z) \qquad \text{starting:} \begin{cases} H_0(z) = & 1 \\ H_1(z) = & 2z \end{cases}$$

All derivatives can be computed recursively, starting from

$$\frac{\mathrm{d}H_0}{\mathrm{d}z} = 0, \quad \frac{\mathrm{d}H_1}{\mathrm{d}z} = 2 \qquad \text{and} \qquad \frac{\mathrm{d}^dH_0}{\mathrm{d}z^d} = \frac{\mathrm{d}^dH_1}{\mathrm{d}z^d} = 0 \quad (\forall\, d > 1),$$

then

$$\begin{aligned}
\frac{\mathrm{d}H_{k+1}}{\mathrm{d}z} &= 2H_k + 2z\frac{\mathrm{d}H_k}{\mathrm{d}z} - 2k\frac{\mathrm{d}H_{k-1}}{\mathrm{d}z} \\
\frac{\mathrm{d}^2H_{k+1}}{\mathrm{d}z^2} &= 4\frac{\mathrm{d}H_k}{\mathrm{d}z} + 2z\frac{\mathrm{d}^2H_k}{\mathrm{d}z^2} - 2k\frac{\mathrm{d}^2H_{k-1}}{\mathrm{d}z^2} \\
\vdots &\quad\quad \vdots \\
\frac{\mathrm{d}^dH_{k+1}}{\mathrm{d}z^d} &= 2d\frac{\mathrm{d}^{d-1}H_k}{\mathrm{d}z^{d-1}} + 2z\frac{\mathrm{d}^dH_k}{\mathrm{d}z^d} - 2k\frac{\mathrm{d}^dH_{k-1}}{\mathrm{d}z^d}
\end{aligned}$$

A.5 Fourier Basis

The Fourier Series (FS) is defined on the domain $z \in [-\pi, \pi]$; however, it does not have a recursive generating function like the other basis sets. In general, the FS can be written as

$$g(z) = \frac{1}{2}a_0 + \sum_{k=1}^{m} \Big(a_k\, \cos(kz) + b_k\, \sin(kz) \Big)$$

The derivatives are of the following based on the order d, where $d > 0$

$$\frac{\mathrm{d}^d g(z)}{\mathrm{d}z^d} = \begin{cases} k^d \sum_{k=1}^{m} \Big(a_k \cos(kz) + b_k \sin(kz) \Big) & \mod (d, 4) = 0 \\ k^d \sum_{k=1}^{m} \Big(-a_k \sin(kz) + b_k \cos(kz) \Big) & \mod (d, 4) = 1 \\ k^d \sum_{k=1}^{m} \Big(-a_k \cos(kz) - b_k \sin(kz) \Big) & \mod (d, 4) = 2 \\ k^d \sum_{k=1}^{m} \Big(a_k \sin(kz) - b_k \cos(kz) \Big) & \mod (d, 4) = 3 \end{cases}$$

APPENDIX B

LINEAR LEAST-SQUARES METHODS

There are different numerical techniques to compute the linear least-squares (LS) solution of $A\,\boldsymbol{\xi} = \boldsymbol{b}$. These are:

- The Moore-Penrose inverse,

$$\boldsymbol{\xi} = (A^{\mathrm{T}}\,A)^{-1}\,A^{\mathrm{T}}\,\boldsymbol{b}.$$

- QR decomposition,

$$A = Q\,R \qquad \rightarrow \qquad \boldsymbol{\xi} = R^{-1}\,Q^{\mathrm{T}}\,\boldsymbol{b},$$

  where Q is an orthogonal matrix and R an upper triangular matrix.

- SVD decomposition,

$$A = U\,\Sigma\,V^{\mathrm{T}} \qquad \rightarrow \qquad \boldsymbol{\xi} = A^{+}\,\boldsymbol{b} = V\,\Sigma^{+}\,U^{\mathrm{T}}\,\boldsymbol{b}$$

  where U and V are two orthogonal matrices, and where Σ^{+} is the pseudo-inverse of Σ, which is formed by replacing every non-zero diagonal entry by its reciprocal and transposing the resulting matrix.

- Cholesky decomposition,

$$A^{\mathrm{T}}A\,\boldsymbol{\xi} = U^{\mathrm{T}}U\boldsymbol{\xi} = A^{\mathrm{T}}\,\boldsymbol{b} \qquad \rightarrow \qquad \boldsymbol{\xi} = U^{-1}\left(U^{-\mathrm{T}}A^{\mathrm{T}}\,\boldsymbol{b}\right),$$

  where U is a upper triangular, and consequently, U^{-1} and $U^{-\mathrm{T}}$ are easy to compute.

One can reduce the condition number of the matrix to be inverted by scaling the columns of A,

$$A\left(SS^{-1}\right)\boldsymbol{\xi} = (AS)\left(S^{-1}\boldsymbol{\xi}\right) = B\,\boldsymbol{\eta} = \boldsymbol{b} \;\rightarrow\; \boldsymbol{\xi} = S\,\boldsymbol{\eta} = S\,(B^{\mathrm{T}}B)^{-1}B^{\mathrm{T}}\boldsymbol{b},$$

where S is the $m \times m$ scaling diagonal matrix whose diagonal elements are the inverse of the norms of the corresponding columns of A: $s_{kk} = |\boldsymbol{a}_k|^{-1}$ or the maximum absolute value, $s_{kk} = \max_i |a_{ki}|$.

In this dissertation, the least-squares problem is solved using two methods: (1) the SVD decomposition introduced above (2) a combination of QR decomposition and the previously mentioned scaling, called the scaled QR approach. This approach performs the QR decomposition of the scaled matrix,

$$B = A\,S = Q\,R \qquad \rightarrow \qquad \boldsymbol{\xi} = S\,R^{-1}\,Q^{\mathrm{T}}\,\boldsymbol{b}.$$

A weighted LS solution can be obtained by introducing an $n \times n$ diagonal matrix of weights, W. This technique exactly follows the Moore-Penrose inverse, however, the weight matrix W allows for unequal emphasis given to the fitting of the solution,

$$W\,A\,\boldsymbol{\xi} = W\,\boldsymbol{b} \qquad \rightarrow \qquad \boldsymbol{\xi} = (A^{\mathrm{T}}\,W^2\,A)^{-1}\,A^{\mathrm{T}}\,W\,\boldsymbol{b}.$$

Furthermore, it can also be shown that a simple scaling of the rows of A is equivalent to weighted LS.

APPENDIX C

SOME COMMON CONSTRAINED EXPRESSIONS

Point and derivative

Constraints:

$$y(x_0) = \kappa_1 \quad \text{and} \quad y_x(x_0) = \kappa_2$$

Projection functionals:

$$\rho_1(x, g(x)) = \kappa_1 - g(x_0) \quad \text{and} \quad \rho_2(x, g(x)) = \kappa_2 - g_x(x_0)$$

Switching functions:

$$\phi_1(x) = 1 \quad \text{and} \quad \phi_2(x) = x - x_0$$

Initial and final point

Constraints:

$$y(x_0) = \kappa_1 \quad \text{and} \quad y(x_f) = \kappa_2$$

Projection functionals:

$$\rho_1(x, g(x)) = \kappa_1 - g(x_0) \quad \text{and} \quad \rho_2(x, g(x)) = \kappa_2 - g(x_f)$$

Switching functions:

$$\phi_1 = \frac{x_f - x}{x_f - x_0} \quad \text{and} \quad \phi_2 = \frac{x - x_0}{x_f - x_0}$$

Initial point and final point/derivative

Constraints:

$$y(x_0) = \kappa_1, \quad y(x_f) = \kappa_2, \quad \text{and} \quad y_x(x_f) = \kappa_3$$

Projection functionals:

$$\rho_1(x, g(x)) = \kappa_1 - g(x_0), \quad \rho_2(x, g(x)) = \kappa_2 - g(x_f) \quad \text{and} \quad \rho_3(x, g(x)) = \kappa_3 - g_x(x_f)$$

Switching functions:

$$\begin{aligned}
\phi_1(x) &= \frac{1}{(x_f - x_0)^2}\Big(x_f^2 - 2x_f x + x^2\Big) \\
\phi_2(x) &= \frac{1}{(x_f - x_0)^2}\Big(x_0(x_0 - 2x_f) + 2x_f x - x^2\Big) \\
\phi_3(x) &= \frac{1}{x_f - x_0}\Big(x_0 x_f - (x_0 + x_f)x + x^2\Big)
\end{aligned}$$

Initial point/derivative and final point

Constraints:

$$y(x_0) = \kappa_1, \quad y_x(x_0) = \kappa_2, \quad \text{and} \quad y(x_f) = \kappa_3$$

Projection functionals:

$$\rho_1(x, g(x)) = \kappa_1 - g(x_0), \quad \rho_2(x, g(x)) = \kappa_2 - g_x(x_0) \quad \text{and} \quad \rho_3(x, g(x)) = \kappa_3 - g(x_f)$$

Switching functions:

$$\begin{aligned}
\phi_1(x) &= \frac{1}{(x_f - x_0)^2}\Big(x_f(x_f - 2x_0) + 2x_0 x - x^2\Big) \\
\phi_2(x) &= \frac{1}{x_f - x_0}\Big(- x_f x_0 + (x_f + x_0)x - x^2\Big) \\
\phi_3(x) &= \frac{1}{(x_f - x_0)^2}\Big(x_0^2 - 2x_0 x + x^2\Big)
\end{aligned}$$

Initial point/derivative and final point/derivative

Constraints:

$$y(x_0) = \kappa_1, \quad y(x_f) = \kappa_2, \quad y_x(x_0) = \kappa_3, \quad \text{and} \quad y_x(x_f) = \kappa_4$$

Projection functionals:

$$\rho_1(x, g(x)) = \kappa_1 - g(x_0) \qquad \rho_3(x, g(x)) = \kappa_3 - g_x(x_0)$$
$$\rho_2(x, g(x)) = \kappa_2 - g(x_f) \qquad \rho_4(x, g(x)) = \kappa_4 - g_x(x_f)$$

Switching functions:

$$\phi_1(x) = \frac{1}{(x_f - x_0)^3}\Big(- x_f^2(3x_0 - x_f) + 6x_0x_fx - 3(x_0 + x_f)x^2 + 2x^3\Big)$$
$$\phi_2(x) = \frac{1}{(x_f - x_0)^3}\Big(- x_0^2(x_0 - 3x_f) - 6x_0x_fx + 3(x_0 + x_f)x^2 - 2x^3\Big)$$
$$\phi_3(x) = \frac{1}{(x_f - x_0)^2}\Big(- x_0x_f^2 + x_f(2x_0 + x_f)x - (x_0 + 2x_f)x^2 + x^3\Big)$$
$$\phi_4(x) = \frac{1}{(x_f - x_0)^2}\Big(- x_0^2x_f + x_0(x_0 + 2x_f)x - (2x_0 + x_f)x^2 + x^3\Big)$$

APPENDIX D

ANALYTICAL TERMS FOR SELECTED PROBLEMS

The analytical terms of this section are provided for completeness; however, in code, these terms are handled through JAX [130, 131] and the TFC toolbox (TFC GitHub) [128] where all of the partial derivatives are taken by automatic differentiation.

D.1 Linear-Nonlinear differential equation Jacobian terms from Section 4.8.1.2

$$\mathbb{J}(\Xi) = \begin{bmatrix} \frac{\partial^{(1)}\tilde{F}(x_0,\Xi)}{\partial^{(1)}\boldsymbol{\xi}} & \mathbf{0}_{1\times m} & \frac{\partial^{(1)}\tilde{F}(x_0,\Xi)}{\partial y_1} & \frac{\partial^{(1)}\tilde{F}(x_0,\Xi)}{\partial y_{1_x}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^{(1)}\tilde{F}(x_1,\Xi)}{\partial^{(1)}\boldsymbol{\xi}} & \mathbf{0}_{1\times m} & \frac{\partial^{(1)}\tilde{F}(x_1,\Xi)}{\partial y_1} & \frac{\partial^{(1)}\tilde{F}(x_1,\Xi)}{\partial y_{1_x}} \\ \mathbf{0}_{1\times m} & \frac{\partial^{(2)}\tilde{F}(x_1,\Xi)}{\partial^{(2)}\boldsymbol{\xi}} & \frac{\partial^{(2)}\tilde{F}(x_1,\Xi)}{\partial y_1} & \frac{\partial^{(2)}\tilde{F}(x_1,\Xi)}{\partial y_{1_x}} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{1\times m} & \frac{\partial^{(2)}\tilde{F}(x_f,\Xi)}{\partial^{(2)}\boldsymbol{\xi}} & \frac{\partial^{(2)}\tilde{F}(x_f,\Xi)}{\partial y_1} & \frac{\partial^{(2)}\tilde{F}(x_f,\Xi)}{\partial y_{1_x}} \end{bmatrix} \tag{D.1}$$

For this problem all terms of Equation (D.1) are provided below:

$$
\begin{aligned}
\frac{\partial\,{}^{(1)}\tilde{F}}{\partial\,{}^{(1)}\boldsymbol{\xi}} &= \Big[c^2\,\boldsymbol{h}_{zz}(z) - {}^{(1)}\phi_{1_{xx}}\boldsymbol{h}(z_0) - {}^{(1)}\phi_{2_{xx}}\boldsymbol{h}(z_1) - {}^{(1)}\phi_{3_{xx}}c\,\boldsymbol{h}_z(z_1) \\
&\qquad + \boldsymbol{h}(z) - {}^{(1)}\phi_1\boldsymbol{h}(z_0) - {}^{(1)}\phi_2\boldsymbol{h}(z_1) - {}^{(1)}\phi_3 c\,\boldsymbol{h}_z(z_1)\Big]^{\mathrm{T}} \\
\frac{\partial\,{}^{(1)}\tilde{F}}{\partial y_1} &= {}^{(1)}\phi_{2_{xx}}(x) + {}^{(1)}\phi_2(x) \\
\frac{\partial\,{}^{(1)}\tilde{F}}{\partial y_{1_x}} &= {}^{(1)}\phi_{3_{xx}}(x) + {}^{(1)}\phi_3(x) \\
\frac{\partial\,{}^{(2)}\tilde{F}}{\partial\,{}^{(2)}\boldsymbol{\xi}} &= \Big[c^2\,\boldsymbol{h}_{zz}(z) - {}^{(2)}\phi_{1_{xx}}\boldsymbol{h}(z_1) - {}^{(2)}\phi_{2_{xx}}c\,\boldsymbol{h}(z_1) - {}^{(2)}\phi_{3_{xx}}\boldsymbol{h}(z_f) \\
&\qquad + {}^{(2)}y_x\Big(\boldsymbol{h}(z) - {}^{(2)}\phi_1\boldsymbol{h}(z_1) - {}^{(2)}\phi_2 c\,\boldsymbol{h}(z_1) - {}^{(2)}\phi_3\boldsymbol{h}(z_f)\Big) \\
&\qquad + {}^{(2)}y\Big(c\,\boldsymbol{h}(z) - {}^{(2)}\phi_{1_x}\boldsymbol{h}(z_1) - {}^{(2)}\phi_{2_x}c\,\boldsymbol{h}(z_1) - {}^{(2)}\phi_{3_x}\boldsymbol{h}(z_f)\Big)\Big]^{\mathrm{T}} \\
\frac{\partial\,{}^{(2)}\tilde{F}}{\partial y_1} &= {}^{(2)}\phi_{1_{xx}}(x) + {}^{(2)}y\,{}^{(2)}\phi_{1_x}(x) + {}^{(2)}y_x\,{}^{(2)}\phi_1(x) \\
\frac{\partial\,{}^{(2)}\tilde{F}}{\partial y_{1_x}} &= {}^{(2)}\phi_{2_{xx}}(x) + {}^{(2)}y\,{}^{(2)}\phi_{2_x}(x) + {}^{(2)}y_x\,{}^{(2)}\phi_2(x)
\end{aligned}
$$

where Ξ is the vector of unknown coefficients such that,

$$
\Xi = \left\{ {}^{(1)}\boldsymbol{\xi}^{\mathrm{T}} \quad {}^{(2)}\boldsymbol{\xi}^{\mathrm{T}} \quad y_1 \quad y_{1_x} \right\}^{\mathrm{T}},
$$

D.2 Convection-diffusion equation from Section 4.8.1.3

The Jacobian is of the form,

$$
\mathbb{J}(\Xi) = \begin{bmatrix}
\frac{\partial\,{}^{(1)}\mathbb{L}}{\partial\,{}^{(1)}\boldsymbol{\xi}} & \mathbf{0}_{N\times m} & \frac{\partial\,{}^{(1)}\mathbb{L}}{\partial y_1} & \frac{\partial\,{}^{(1)}\mathbb{L}}{\partial y_{1_x}} & \frac{\partial\,{}^{(1)}\mathbb{L}}{\partial \bar{c}} \\
\mathbf{0}_{N\times m} & \frac{\partial\,{}^{(2)}\mathbb{L}}{\partial\,{}^{(1)}\boldsymbol{\xi}} & \frac{\partial\,{}^{(2)}\mathbb{L}}{\partial y_1} & \frac{\partial\,{}^{(2)}\mathbb{L}}{\partial y_{1_x}} & \frac{\partial\,{}^{(2)}\mathbb{L}}{\partial \bar{c}}
\end{bmatrix}
$$

where the following equations are the detailed Jacobian terms from the convection-diffusion equation from Section 4.8.1.3. For clarity, the constrained expressions are,

$$\begin{aligned}
{}^{(1)}y(z,{}^{(1)}\boldsymbol{\xi}) &= \Big(\boldsymbol{h}(z) - {}^{(1)}\phi_1(z)\boldsymbol{h}(z_0) + {}^{(1)}\phi_2(z)\boldsymbol{h}(z_f) + {}^{(1)}\phi_3(z)\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\,{}^{(1)}\boldsymbol{\xi} \\
&\quad + {}^{(1)}\phi_1(z)y_0 + {}^{(1)}\phi_2(z)y_1 + {}^{(1)}\phi_3(z)\frac{y_{1_x}}{{}^{(1)}c} \\
{}^{(2)}y(z,{}^{(2)}\boldsymbol{\xi}) &= \Big(\boldsymbol{h}(z) - {}^{(2)}\phi_1(z)\boldsymbol{h}(z_0) + {}^{(2)}\phi_2(z)\boldsymbol{h}_z(z_0) + {}^{(2)}\phi_3(z)\boldsymbol{h}(z_f)\Big)^{\mathrm{T}}\,{}^{(2)}\boldsymbol{\xi} \\
&\quad + {}^{(2)}\phi_1(z)y_1 + {}^{(2)}\phi_2(z)\frac{y_{1_x}}{{}^{(2)}c} + {}^{(2)}\phi_3(z)y_f
\end{aligned}$$

where the loss vectors of each segment are,

$${}^{(1)}\mathbb{L}(\Xi) = \begin{Bmatrix} {}^{(1)}\tilde{F}(z_0,\Xi) \\ \vdots \\ {}^{(1)}\tilde{F}(z_f,\Xi) \end{Bmatrix} = \begin{Bmatrix} \bar{c}^2\,{}^{(1)}y_{xx}(z_0,\Xi) - \mathrm{Pe}\,\bar{c}\,{}^{(1)}y_x(z_0,\Xi) \\ \vdots \\ \bar{c}^2\,{}^{(1)}y_{xx}(z_f,\Xi) - \mathrm{Pe}\,\bar{c}\,{}^{(1)}y_x(z_f,\Xi) \end{Bmatrix}$$

and

$${}^{(2)}\mathbb{L}(\Xi) = \begin{Bmatrix} {}^{(2)}\tilde{F}(z_0,\Xi) \\ \vdots \\ {}^{(2)}\tilde{F}(z_f,\Xi) \end{Bmatrix} = \begin{Bmatrix} \left(\frac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)^2 {}^{(2)}y_{xx}(z_0,\Xi) - \mathrm{Pe}\left(\frac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right) {}^{(2)}y_x(z_0,\Xi) \\ \vdots \\ \left(\frac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)^2 {}^{(2)}y_{xx}(z_f,\Xi) - \mathrm{Pe}\left(\frac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right) {}^{(2)}y_x(z_f,\Xi) \end{Bmatrix}.$$

The following equations are the Jacobians of the loss vectors with respect to the unknowns:

$$\frac{{}^{(1)}\mathbb{L}(\Xi)}{\partial\,{}^{(1)}\boldsymbol{\xi}} = \begin{bmatrix} \left[\bar{c}^2\frac{\partial\,{}^{(1)}y_{zz}}{\partial\,{}^{(1)}\boldsymbol{\xi}}(z_0) - \mathrm{Pe}\,\bar{c}\,\frac{\partial\,{}^{(1)}y_z}{\partial\,{}^{(1)}\boldsymbol{\xi}}(z_0)\right]^{\mathrm{T}} \\ \vdots \\ \left[\bar{c}^2\frac{\partial\,{}^{(1)}y_{zz}}{\partial\,{}^{(1)}\boldsymbol{\xi}}(z_f) - \mathrm{Pe}\,\bar{c}\,\frac{\partial\,{}^{(1)}y_z}{\partial\,{}^{(1)}\boldsymbol{\xi}}(z_f)\right]^{\mathrm{T}} \end{bmatrix}$$

$$\frac{{}^{(1)}\mathbb{L}(\Xi)}{\partial y_1} = \begin{Bmatrix} \bar{c}^{2\,(1)}\phi_{2_{zz}}(z_0) - \mathrm{Pe}\,\bar{c}\,{}^{(1)}\phi_{2_z}(z_0) \\ \vdots \\ \bar{c}^{2\,(1)}\phi_{2_{zz}}(z_f) - \mathrm{Pe}\,\bar{c}\,{}^{(1)}\phi_{2_z}(z_f) \end{Bmatrix}$$

$$\frac{{}^{(1)}\mathbb{L}(\Xi)}{\partial y_{1_x}} = \begin{Bmatrix} \bar{c}\,{}^{(1)}\phi_{3_{zz}}(z_0) - \mathrm{Pe}\,{}^{(1)}\phi_{3_z}(z_0) \\ \vdots \\ \bar{c}\,{}^{(1)}\phi_{3_{zz}}(z_f) - \mathrm{Pe}\,{}^{(1)}\phi_{3_z}(z_f) \end{Bmatrix}$$

$$\frac{{}^{(1)}\mathbb{L}(\Xi)}{\partial \bar{c}} = \begin{Bmatrix} 2\bar{c}\,{}^{(1)}y_{zz}(z_0) - {}^{(1)}\phi_{3_{zz}}(z_0)y_{1_x} - \mathrm{Pe}\,{}^{(1)}y_z(z_0) + \mathrm{Pe}\dfrac{{}^{(1)}\phi_{3_z}(z_0)y_{1_x}}{\bar{c}} \\ \vdots \\ 2\bar{c}\,{}^{(1)}y_{zz}(z_f) - {}^{(1)}\phi_{3_{zz}}(z_f)y_{1_x} - \mathrm{Pe}\,{}^{(1)}y_z(z_f) + \mathrm{Pe}\dfrac{{}^{(1)}\phi_{3_z}(z_f)y_{1_x}}{\bar{c}} \end{Bmatrix}$$

$$\frac{{}^{(2)}\mathbb{L}(\Xi)}{\partial\,{}^{(2)}\boldsymbol{\xi}} = \begin{bmatrix} \left[\left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)^2 \dfrac{\partial\,{}^{(2)}y_{zz}}{\partial\,{}^{(2)}\boldsymbol{\xi}}(z_0) - \mathrm{Pe}\left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)\dfrac{\partial\,{}^{(2)}y_z}{\partial\,{}^{(2)}\boldsymbol{\xi}}(z_0)\right]^{\mathrm{T}} \\ \vdots \\ \left[\left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)^2 \dfrac{\partial\,{}^{(2)}y_{zz}}{\partial\,{}^{(2)}\boldsymbol{\xi}}(z_f) - \mathrm{Pe}\left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)\dfrac{\partial\,{}^{(2)}y_z}{\partial\,{}^{(2)}\boldsymbol{\xi}}(z_f)\right]^{\mathrm{T}} \end{bmatrix}$$

$$\frac{{}^{(2)}\mathbb{L}(\Xi)}{\partial y_1} = \begin{Bmatrix} \left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)^2 {}^{(2)}\phi_{1_{zz}}(z_0) - \mathrm{Pe}\left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right){}^{(2)}\phi_{1_z}(z_0) \\ \vdots \\ \left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right)^2 {}^{(2)}\phi_{1_{zz}}(z_f) - \mathrm{Pe}\left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right){}^{(2)}\phi_{1_z}(z_f) \end{Bmatrix}$$

$$\frac{{}^{(2)}\mathbb{L}(\Xi)}{\partial y_{1_x}} = \begin{Bmatrix} \left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right){}^{(2)}\phi_{2_{zz}}(z_0) - \mathrm{Pe}\,{}^{(2)}\phi_{2_z}(z_0) \\ \vdots \\ \left(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\right){}^{(2)}\phi_{2_{zz}}(z_f) - \mathrm{Pe}\,{}^{(2)}\phi_{2_z}(z_f) \end{Bmatrix}$$

$$\frac{{}^{(2)}\mathbb{L}(\Xi)}{\partial \bar{c}} = \begin{Bmatrix} -\dfrac{\Delta z^2}{(\bar{c}-\Delta z)^2}\left[2\Big(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\Big){}^{(2)}y_{zz}(z_0) - {}^{(2)}\phi_{2_{zz}}(z_0)y_{1_x} - \mathrm{Pe}\,{}^{(2)}y_z(z_0) + \mathrm{Pe}\dfrac{{}^{(2)}\phi_{2_z}(z_0)y_{1_x}}{\Big(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\Big)}\right] \\ \vdots \\ -\dfrac{\Delta z^2}{(\bar{c}-\Delta z)^2}\left[2\Big(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\Big){}^{(2)}y_{zz}(z_f) - {}^{(2)}\phi_{2_{zz}}(z_f)y_{1_x} - \mathrm{Pe}\,{}^{(2)}y_z(z_f) + \mathrm{Pe}\dfrac{{}^{(2)}\phi_{2_z}(z_f)y_{1_x}}{\Big(\dfrac{\bar{c}\Delta z}{\bar{c}-\Delta z}\Big)}\right] \end{Bmatrix}$$

D.3 Terms for Outer-loop approach in the energy optimal landing problem from Section 6.3.1

By discretizing the domain the linear system becomes,

$$\begin{bmatrix} \mathbb{A} & \mathbf{0}_{N\times m} & \mathbf{0}_{N\times m} & -\mathbb{C} & \mathbf{0}_{N\times 2} & \mathbf{0}_{N\times 2} \\ \mathbf{0}_{N\times m} & \mathbb{A} & \mathbf{0}_{N\times m} & \mathbf{0}_{N\times 2} & -\mathbb{C} & \mathbf{0}_{N\times 2} \\ \mathbf{0}_{N\times m} & \mathbf{0}_{N\times m} & \mathbb{A} & \mathbf{0}_{N\times 2} & \mathbf{0}_{N\times 2} & -\mathbb{C} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\xi}_1 \\ \boldsymbol{\xi}_2 \\ \boldsymbol{\xi}_3 \\ \boldsymbol{\xi}_{u_1} \\ \boldsymbol{\xi}_{u_2} \\ \boldsymbol{\xi}_{u_3} \end{Bmatrix} = - \begin{Bmatrix} \mathbb{B}_1 \\ \mathbb{B}_2 \\ \mathbb{B}_3 \end{Bmatrix}$$

where $\mathbb{A}$, $\mathbb{B}_i$, $\mathbb{C}$ are defined as,

$$\mathbb{A} = \begin{bmatrix} \left(c^2\boldsymbol{h}_{zz}(z_0) - \ddot{\phi}_1(t_0)\boldsymbol{h}(z_0) - \ddot{\phi}_2(t_0)\boldsymbol{h}(z_f) - \ddot{\phi}_3(t_0)c\boldsymbol{h}_z(z_0) - \ddot{\phi}_4(t_0)c\boldsymbol{h}_z(z_f)\right)^{\mathrm{T}} \\ \vdots \\ \left(c^2\boldsymbol{h}_{zz}(z_k) - \ddot{\phi}_1(t_k)\boldsymbol{h}(z_0) - \ddot{\phi}_2(t_k)\boldsymbol{h}(z_f) - \ddot{\phi}_3(t_k)c\boldsymbol{h}_z(z_0) - \ddot{\phi}_4(t_k)c\boldsymbol{h}_z(z_f)\right)^{\mathrm{T}} \\ \vdots \\ \left(c^2\boldsymbol{h}_{zz}(z_f) - \ddot{\phi}_1(t_f)\boldsymbol{h}(z_0) - \ddot{\phi}_2(t_f)\boldsymbol{h}(z_f) - \ddot{\phi}_3(t_f)c\boldsymbol{h}_z(z_0) - \ddot{\phi}_4(t_f)c\boldsymbol{h}_z(z_f)\right)^{\mathrm{T}} \end{bmatrix}$$

$$\mathbb{B}_i = \begin{bmatrix} \ddot{\phi}_1(t_0)r_{0_i} + \ddot{\phi}_2(t_0)r_{f_i} + \ddot{\phi}_3(t_0)v_{0_i} + \ddot{\phi}_4(t_0)v_{f_i} - a_{g_i} \\ \vdots \\ \ddot{\phi}_1(t_k)r_{0_i} + \ddot{\phi}_2(t_k)r_{f_i} + \ddot{\phi}_3(t_k)v_{0_i} + \ddot{\phi}_4(t_k)v_{f_i} - a_{g_i} \\ \vdots \\ \ddot{\phi}_1(t_f)r_{0_i} + \ddot{\phi}_2(t_f)r_{f_i} + \ddot{\phi}_3(t_f)v_{0_i} + \ddot{\phi}_4(t_f)v_{f_i} - a_{g_i} \end{bmatrix} \quad \mathbb{C} = \begin{bmatrix} \boldsymbol{h}_u^{\mathrm{T}}(z_0) \\ \vdots \\ \boldsymbol{h}_u^{\mathrm{T}}(z_k) \\ \vdots \\ \boldsymbol{h}_u^{\mathrm{T}}(z_f) \end{bmatrix}.$$

D.4 Single-loop approach Jacobian terms in the energy optimal landing problem from Section 6.3.2

The partial derivatives for the state loss function, $\mathbb{L}_i$, when $i = j$ are,

$$\begin{aligned}\frac{\partial \mathbb{L}_i}{\partial \boldsymbol{\xi}_j} &= b^4\Big(\boldsymbol{h}_{zz}(z) - {}^z\phi_{1_{zz}}\boldsymbol{h}_0 - {}^z\phi_{2_{zz}}\boldsymbol{h}_f - {}^z\phi_{3_{zz}}c\boldsymbol{h}_z(z_0) - {}^z\phi_{4_{zz}}c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\\ \frac{\partial \mathbb{L}_i}{\partial \boldsymbol{\xi}_{u_j}} &= \boldsymbol{h}_u^{\mathrm{T}}.\end{aligned}$$

If $i \neq j$

$$\begin{aligned}\frac{\partial \mathbb{L}_i}{\partial \boldsymbol{\xi}_j} &= \mathbf{0}_{N\times m}\\ \frac{\partial \mathbb{L}_i}{\partial \boldsymbol{\xi}_{u_j}} &= \mathbf{0}_{N\times 2}\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \mathbb{L}_i}{\partial b} = 4b^3\Big[\Big(\boldsymbol{h}_{zz}(z) - {}^z\phi_{1_{zz}}\boldsymbol{h}_0 - {}^z\phi_{2_{zz}}\boldsymbol{h}_f - {}^z\phi_{3_{zz}}c\boldsymbol{h}_z(z_0) - {}^z\phi_{4_{zz}}c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}\\ + {}^z\phi_{1_{zz}}r_{0_i} + {}^z\phi_{2_{zz}}r_{f_i}\Big]\\ + 2b\Big[{}^z\phi_{3_{zz}}v_{0_i} + {}^z\phi_{4_{zz}}v_{f_i}\Big].\end{aligned}$$

Similarly, the partial derivatives for $\mathbb{L}_H$ are,

$$\begin{aligned}\frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_j} &= \mathbf{0}_{1\times m}\\ \frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_{u_j}} = \frac{\partial \mathbb{L}_H}{\partial u_j}\frac{\partial u_j}{\partial \boldsymbol{\xi}_{u_j}} &= \boldsymbol{h}_u^{\mathrm{T}}\Big(a_{g_j} - u_j(z_f)\Big)\\ \frac{\partial \mathbb{L}_H}{\partial b} &= 0.\end{aligned}$$

Combining these into a single Jacobian term leads to,

$$\mathbb{J} = \begin{bmatrix} \frac{\partial \mathbb{L}_1}{\partial \boldsymbol{\xi}_1} & \mathbf{0}_{N\times m} & \mathbf{0}_{N\times m} & \frac{\partial \mathbb{L}_1}{\partial \boldsymbol{\xi}_{u_1}} & \mathbf{0}_{N\times 2} & \mathbf{0}_{N\times 2} & \frac{\partial \mathbb{L}_1}{\partial b} \\ \mathbf{0}_{N\times m} & \frac{\partial \mathbb{L}_2}{\partial \boldsymbol{\xi}_2} & \mathbf{0}_{N\times m} & \mathbf{0}_{N\times 2} & \frac{\partial \mathbb{L}_2}{\partial \boldsymbol{\xi}_{u_2}} & \mathbf{0}_{N\times 2} & \frac{\partial \mathbb{L}_2}{\partial b} \\ \mathbf{0}_{N\times m} & \mathbf{0}_{N\times m} & \frac{\partial \mathbb{L}_3}{\partial \boldsymbol{\xi}_3} & \mathbf{0}_{N\times 2} & \mathbf{0}_{N\times 2} & \frac{\partial \mathbb{L}_3}{\partial \boldsymbol{\xi}_{u_3}} & \frac{\partial \mathbb{L}_1}{\partial b} \\ \mathbf{0}_{1\times m} & \mathbf{0}_{1\times m} & \mathbf{0}_{1\times m} & \frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_1} & \frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_2} & \frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_3} & \frac{\partial \mathbb{L}_H}{\partial b} \end{bmatrix}_{(3N+1)\times(3m+7)}$$

with the augmented loss function and unknown vector defined as

$$\mathbb{L} = \begin{Bmatrix} \mathbb{L}_1^{\mathrm{T}} & \mathbb{L}_2^{\mathrm{T}} & \mathbb{L}_3^{\mathrm{T}} & \mathbb{L}_H \end{Bmatrix}^{\mathrm{T}}_{(3N+1)\times 1}$$
$$\Xi = \begin{Bmatrix} \boldsymbol{\xi}_1^{\mathrm{T}} & \boldsymbol{\xi}_2^{\mathrm{T}} & \boldsymbol{\xi}_3^{\mathrm{T}} & \boldsymbol{\xi}_{u_1}^{\mathrm{T}} & \boldsymbol{\xi}_{u_2}^{\mathrm{T}} & \boldsymbol{\xi}_{u_3}^{\mathrm{T}} & b \end{Bmatrix}^{\mathrm{T}}_{(3m+7)\times 1}.$$

D.5 Fuel-Optimal Landing from Section 7.3

In the fuel-optimal landing problem the analytical partial derivatives of the state loss function are:

$$\frac{\partial^{(s)}\mathbb{L}_i}{\partial^{(s)}\boldsymbol{\xi}_i} = {}^{(s)}\Big(c^2\boldsymbol{h}_{zz} - \ddot{\phi}_1(t)\boldsymbol{h}(z_0) - \ddot{\phi}_2(t)\boldsymbol{h}(z_f) - \ddot{\phi}_3(t)c\boldsymbol{h}_z(z_0) - \ddot{\phi}_4(t)c\boldsymbol{h}_z(z_f)\Big)^{\mathrm{T}}$$
$$\frac{\partial^{(1)}\mathbb{L}_i}{\partial r_{1_i}} = {}^{(1)}\ddot{\phi}_2(t)$$
$$\frac{\partial^{(1)}\mathbb{L}_i}{\partial v_{1_i}} = {}^{(1)}\ddot{\phi}_4(t)$$
$$\frac{\partial^{(2)}\mathbb{L}_i}{\partial r_{1_i}} = {}^{(2)}\ddot{\phi}_1(t), \qquad \frac{\partial^{(2)}\mathbb{L}_i}{\partial v_{1_i}} = {}^{(2)}\ddot{\phi}_3(t)$$
$$\frac{\partial^{(2)}\mathbb{L}_i}{\partial r_{2_i}} = {}^{(2)}\ddot{\phi}_2(t), \qquad \frac{\partial^{(2)}\mathbb{L}_i}{\partial v_{2_i}} = {}^{(2)}\ddot{\phi}_4(t)$$
$$\frac{\partial^{(3)}\mathbb{L}_i}{\partial r_{2_i}} = {}^{(3)}\ddot{\phi}_1(t)$$
$$\frac{\partial^{(3)}\mathbb{L}_i}{\partial v_{2_i}} = {}^{(3)}\ddot{\phi}_3(t).$$

For the costate portion, if $i = j$

$$\frac{\partial \mathbb{L}_i}{\partial \boldsymbol{\xi}_{\lambda_i}} = \beta(t) \left[\left(\sum_{j=1}^{3} \lambda_{v_j}^2 \right)^{-1/2} - \lambda_{v_i}^2 \left(\sum_{j=1}^{3} \lambda_{v_j}^2 \right)^{-3/2} \right] \boldsymbol{h}_\lambda^{\mathrm{T}}$$

if $i \neq j$

$$\frac{\partial \mathbb{L}_i}{\partial \boldsymbol{\xi}_{\lambda_j}} = \beta(t) \left[-\lambda_{v_i} \lambda_{v_j} \left(\sum_{j=1}^{3} \lambda_{v_j}^2 \right)^{-3/2} \right] \boldsymbol{h}_\lambda^{\mathrm{T}}.$$

For the loss function associated with the transversality conditions for the Hamiltonian, $\mathbb{L}_H$, the only non-zero partial is with respect to $\boldsymbol{\xi}_\lambda$, which is defined by

$$\frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_{\lambda_i}} = \left[a_{g_i} - \beta(t_f)\, \lambda_{v_i}(t_f) \left(\sum_{j=1}^{3} \lambda_{v_j}^2(t_f) \right)^{-1/2} \right] \boldsymbol{h}_\lambda^{\mathrm{T}}(t_f).$$

The augmented loss functions for the discretized points become

$$\mathbb{L} = \left\{ {}^{(1)}\mathbb{L}_1^{\mathrm{T}} \quad {}^{(1)}\mathbb{L}_2^{\mathrm{T}} \quad {}^{(1)}\mathbb{L}_3^{\mathrm{T}} \quad {}^{(2)}\mathbb{L}_1^{\mathrm{T}} \quad {}^{(2)}\mathbb{L}_2^{\mathrm{T}} \quad {}^{(2)}\mathbb{L}_3^{\mathrm{T}} \quad {}^{(3)}\mathbb{L}_1^{\mathrm{T}} \quad {}^{(3)}\mathbb{L}_2^{\mathrm{T}} \quad {}^{(3)}\mathbb{L}_3^{\mathrm{T}} \quad \mathbb{L}_H \right\}^{\mathrm{T}}_{(\{9N+1\}\times 1)}$$

with the unknown vector

$$\Xi = \Big\{ {}^{(1)}\boldsymbol{\xi}_1^{\mathrm{T}} \quad {}^{(1)}\boldsymbol{\xi}_2^{\mathrm{T}} \quad {}^{(1)}\boldsymbol{\xi}_3^{\mathrm{T}} \quad {}^{(2)}\boldsymbol{\xi}_1^{\mathrm{T}} \quad {}^{(2)}\boldsymbol{\xi}_2^{\mathrm{T}} \quad {}^{(2)}\boldsymbol{\xi}_3^{\mathrm{T}} \quad {}^{(3)}\boldsymbol{\xi}_1^{\mathrm{T}} \quad {}^{(3)}\boldsymbol{\xi}_2^{\mathrm{T}} \quad {}^{(3)}\boldsymbol{\xi}_3^{\mathrm{T}}$$
$$\boldsymbol{\xi}_{\lambda_1}^{\mathrm{T}} \quad \boldsymbol{\xi}_{\lambda_2}^{\mathrm{T}} \quad \boldsymbol{\xi}_{\lambda_3}^{\mathrm{T}} \quad \boldsymbol{r}_1^{\mathrm{T}} \quad \boldsymbol{v}_1^{\mathrm{T}} \quad \boldsymbol{r}_2^{\mathrm{T}} \quad \boldsymbol{v}_2^{\mathrm{T}} \Big\}^{\mathrm{T}}_{(9m+18)}.$$

All partials can be combined into one augmented matrix,

$$\mathbb{J} = \begin{bmatrix} {}^{(1)}J_{\boldsymbol{\xi}} & \mathbf{0}_{(3N\times 3m)} & \mathbf{0}_{(3N\times 3m)} & {}^{(1)}J_{\boldsymbol{\xi}_\lambda} & {}^{(1)}J_{r_1,v_1} & \mathbf{0}_{(3N\times 6)} \\ \mathbf{0}_{(3N\times 3m)} & {}^{(2)}J_{\boldsymbol{\xi}} & \mathbf{0}_{(3N\times 3m)} & {}^{(2)}J_{\boldsymbol{\xi}_\lambda} & {}^{(2)}J_{r_1,v_1} & {}^{(2)}J_{r_2,v_2} \\ \mathbf{0}_{(3N\times 3m)} & \mathbf{0}_{(3N\times 3m)} & {}^{(3)}J_{\boldsymbol{\xi}} & {}^{(3)}J_{\boldsymbol{\xi}_\lambda} & \mathbf{0}_{(3N\times 6)} & {}^{(3)}J_{r_2,v_2} \\ \mathbf{0}_{(1\times 3m)} & \mathbf{0}_{(1\times 3m)} & \mathbf{0}_{(1\times 3m)} & J_H & \mathbf{0}_{(1\times 6)} & \mathbf{0}_{(1\times 6)} \end{bmatrix}_{(\{9N+1\}\times\{9m+18\})} \tag{D.2}$$

The terms of Equation (D.2) are defined by the following equations:

$$
{}^{(s)}J_{\boldsymbol{\xi}} = \begin{bmatrix} \frac{\partial {}^{(s)}\mathbb{L}_1}{\partial {}^{(s)}\boldsymbol{\xi}_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial {}^{(s)}\mathbb{L}_2}{\partial {}^{(s)}\boldsymbol{\xi}_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial {}^{(s)}\mathbb{L}_3}{\partial {}^{(s)}\boldsymbol{\xi}_3} \end{bmatrix}_{(3N \times 3m)}, \quad {}^{(s)}J_{\boldsymbol{\xi}_\lambda} = \begin{bmatrix} J_{\boldsymbol{\xi}_{\lambda_{11}}} & J_{\boldsymbol{\xi}_{\lambda_{12}}} & J_{\boldsymbol{\xi}_{\lambda_{13}}} \\ J_{\boldsymbol{\xi}_{\lambda_{21}}} & J_{\boldsymbol{\xi}_{\lambda_{22}}} & J_{\boldsymbol{\xi}_{\lambda_{23}}} \\ J_{\boldsymbol{\xi}_{\lambda_{31}}} & J_{\boldsymbol{\xi}_{\lambda_{32}}} & J_{\boldsymbol{\xi}_{\lambda_{33}}} \end{bmatrix}_{(3N \times 6)}
$$

$$
{}^{(1)}J_{r_1,v_1} = \begin{bmatrix} {}^{(1)}\ddot{\phi}_2 & \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_4 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & {}^{(1)}\ddot{\phi}_2 & \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_4 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_2 & \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_4 \end{bmatrix}_{(3N \times 6)}
$$

$$
{}^{(2)}J_{r_1,v_1} = \begin{bmatrix} {}^{(1)}\ddot{\phi}_1 & \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & {}^{(1)}\ddot{\phi}_1 & \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_1 & \mathbf{0} & \mathbf{0} & {}^{(1)}\ddot{\phi}_3 \end{bmatrix}_{(3N \times 6)}
$$

$$
{}^{(2)}J_{r_2,v_2} = \begin{bmatrix} {}^{(2)}\ddot{\phi}_2 & \mathbf{0} & \mathbf{0} & {}^{(2)}\ddot{\phi}_4 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & {}^{(2)}\ddot{\phi}_2 & \mathbf{0} & \mathbf{0} & {}^{(2)}\ddot{\phi}_4 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & {}^{(2)}\ddot{\phi}_2 & \mathbf{0} & \mathbf{0} & {}^{(2)}\ddot{\phi}_4 \end{bmatrix}_{(3N \times 6)}
$$

$$
{}^{(3)}J_{r_2,v_2} = \begin{bmatrix} {}^{(3)}\ddot{\phi}_1 & \mathbf{0} & \mathbf{0} & {}^{(3)}\ddot{\phi}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & {}^{(3)}\ddot{\phi}_1 & \mathbf{0} & \mathbf{0} & {}^{(3)}\ddot{\phi}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & {}^{(3)}\ddot{\phi}_1 & \mathbf{0} & \mathbf{0} & {}^{(3)}\ddot{\phi}_3 \end{bmatrix}_{(3N \times 6)}
$$

$$
J_H = \begin{bmatrix} \frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_{\lambda_1}}, & \frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_{\lambda_2}}, & \frac{\partial \mathbb{L}_H}{\partial \boldsymbol{\xi}_{\lambda_3}} \end{bmatrix}_{(1 \times 6)}.
$$