MULTIRESOLUTION STRATEGIES FOR THE NUMERICAL SOLUTION OF OPTIMAL CONTROL PROBLEMS

A Thesis
Presented to
The Academic Faculty

by

Sachin Jain

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
April 2008

MULTIRESOLUTION STRATEGIES FOR THE NUMERICAL SOLUTION OF OPTIMAL CONTROL PROBLEMS

Dedicated to my parents

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor and mentor Dr. Panagiotis Tsiotras for his constant individual attention, guidance, and support without which I couldn't have made it this far. His patience and kindness with his vast knowledge and broad interests in research played a crucial role in my development at Georgia Tech. I am very grateful to him for all the enormous amount of advise and encouragement that he has provided throughout the course of this work. I thoroughly enjoyed working under him and would like to thank him for making my experience at Georgia Tech a valuable one.

I would also like to thank my committee members, Dr. Hao-Min Zhou, Dr. Anthony J. Calise, Dr. J.V.R. Prasad, Dr. Magnus Egerstedt, and Dr. Ryan P. Russell for their advise and comments which have helped me carry this research much further than I could ever have by myself. I am very grateful to Dr. Zhou for his ever-extended assistance in my research. His immense knowledge on wavelets and evolution PDEs was very much appreciated. I would like to thank him for all his valuable advise and suggestions that helped me to move forward in my research work. I would like to express my deepest gratitude to Dr. Calise for his valuable comment on the initial guesses for solving trajectory optimization problems which helped me solve challenging optimal control problems; to Dr. Prasad for his course in "Optimal Guidance & Control" which laid the foundation for my work; to Dr. Egerstedt and Dr. Russell for their willingness to be a part of my thesis defense committee.

I would like to thank all my teachers here at Georgia Tech for their dedication and effort in instilling deep and fundamental understanding of course material. I would like to thank Dr. J. Jagoda (Associate Chair for Graduate Studies and Research) and the staff of the Aerospace Engineering office for always being ready with school related assistance.

I would like to acknowledge all my lab-mates of the Dynamics and Control Systems Lab (DCSL) for the hours of productive discussion, advise, and moral support. I would also like to thank all of my close friends who have supported me morally during these past few years.

I owe my greatest gratitude to my parents to whom this work is dedicated and who's love and sacrifices have made me worthy of this accomplishment. Thank you to my beloved family: my father, mother, sister, and brother-in-law for making even the most stressful of my days brighter. Their constant encouragement and great unconditional love gave me strength to get through this incredible journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

There exist many numerical techniques for solving optimal control problems but less work has been done in the field of making these algorithms run faster and more robustly. The main motivation of this work is to solve optimal control problems accurately in a fast and efficient way.

Optimal control problems are often characterized by discontinuities or switchings in the control variables. One way of accurately capturing the irregularities in the solution is to use a high resolution (dense) uniform grid. This requires a large amount of computational resources both in terms of CPU time and memory. Hence, in order to accurately capture any irregularities in the solution using a few computational resources, one can refine the mesh locally in the region close to an irregularity instead of refining the mesh uniformly over the whole domain. Therefore, a novel multiresolution scheme for data compression has been designed which is shown to outperform similar data compression schemes. Specifically, we have shown that the proposed approach results in fewer grid points in the grid compared to a common multiresolution data compression scheme.

The validity of the proposed mesh refinement algorithm has been verified by solving several challenging initial-boundary value problems for evolution equations in 1D. The examples have demonstrated the stability and robustness of the proposed algorithm. The algorithm adapted dynamically to any existing or emerging irregularities in the solution by automatically allocating more grid points to the region where the solution exhibited sharp features and fewer points to the region where the solution was smooth. Thereby, the computational time and memory usage has been reduced significantly, while maintaining an accuracy equivalent to the one obtained using a fine uniform mesh.

Next, a direct multiresolution-based approach for solving trajectory optimization problems is developed. The original optimal control problem is transcribed into a nonlinear programming (NLP) problem that is solved using standard NLP codes. The novelty of the proposed approach hinges on the automatic calculation of a suitable, nonuniform grid over which the NLP problem is solved, which tends to increase numerical efficiency and robustness. Control and/or state constraints are handled with ease, and without any additional computational complexity. The proposed algorithm is based on a simple and intuitive method to balance several conflicting objectives, such as accuracy of the solution, convergence, and speed of the computations. The benefits of the proposed algorithm over uniform grid implementations are demonstrated with the help of several nontrivial examples.

Furthermore, two sequential multiresolution trajectory optimization algorithms for solving problems with moving targets and/or dynamically changing environments have been developed. For such problems, high accuracy is desirable only in the immediate future, yet the ultimate mission objectives should be accommodated as well. An intelligent trajectory generation for such situations is thus enabled by introducing the idea of multigrid temporal resolution to solve the associated trajectory optimization problem on a non-uniform grid across time that is adapted to: (i) immediate future, and (ii) potential discontinuities in the state and control variables.

CHAPTER I

MOTIVATION AND LITERATURE SURVEY

1.1 A Brief History of Optimal Control

The objective of optimal control theory is to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criterion. The history of optimal control dates back to the 17th century when the calculus of variation originated (Fermat, Newton, Liebniz, and the Bernoullis). It is believed that the calculus of variation started with Pierre de Fermat (1601-1665) when in 1662 he postulated his principle that the light rays follow the minimum time paths [32, 62]. In the late XVII-th century, Bernoulli (1667-1748) used Fermat's ideas to solve a discrete-time version of the "brachistochrone" problem posed by Galileo Galilei (1564-1642) in the XVI-th century. The "brachistochrone" problem is to find the shape of a wire such that a bead sliding along it traverses the distance between the two end points in minimum time. Later Bernoulli challenged his colleagues to solve the continuous brachistochrone problem; not only did he solve it himself, but so did Leibniz, his brother James, l'Hospital, and Newton. Calculus of variations was futher developed by Euler (1707-1783) and Lagrange (1736-1813) who gave the first-order necessary conditions of optimality for minimizing or maximizing a functional. These conditions are commonly known as *Euler-Lagrange equations*. The next step was to look at the second variation, and Legendre (1752-1833) was the first one to do this, who found an additional necessary condition of optimality for a minimum. During the middle of XIX-th century, Jacobi (1804-1851) and Hamilton (1805-1865) showed that the partial derivatives of the performance index with respect to each parameter of a family of extremals (which today we call *states*) obey a certain differential equation. The equation is the *Hamilton-Jacobi* equation, which is the basis of *dynamic programming* developed by Bellman over 100 years later.

Weirstrass (1815-1897) put calculus of variations on a more rigorous basis and discovered his famous condition involving an "excess-function" which is the predecessor of the *maximum principle* of Pontryagin in this century. As pointed out by Sussmann [133], Weierstrass' condition, expressed in terms of the Hamiltonian, simply says that along the optimal curve the optimal control must maximize the Hamiltonian (where the classical definition of the Hamiltonian is used, which is opposite in sign from the one commonly used today). During this period, Clebsch (1833-1872) gave a sharper interpretation of Legendre's condition (the *Legendre-Clebsch condition*) which, in modern language, states that the second derivative matrix of the Hamiltonian with respect to the controls must be positive definite (assuming no active control or state constraints). Later Bolza (1857-1942) and Bliss (1876-1951) gave calculus of variations its present rigorous mathematical structure.

Dynamic programming, a new vision and an extension of Hamilton-Jacobi thoery, was developed by Bellman and his colleagues starting in the 1950s [12] which led to the *Hamilton-Jacobi-Bellman (HJB) equation.* The HJB equation is a partial differential equation which defines the optimal cost to go function, that is, the performance index value from current time to the end, on the optimal trajectory for the continuous time problems. In the middle of XX-th century, Pontryagin extended the calculus of variations to handle control variable inequality constraints, in particular, extended the necessary conditions derived by Weierstrass (1815-1897) to the cases where the control functions are bounded, enunciating his elegant *maximum principle* [29, 59, 60]. In optimal control terminology, it states that a minimizing path must satisfy Euler-Lagrange equations where the optimal controls maximize the Hamiltonian within their bounded region at each point along the path[1]. The maximum principle is inherent in dynamic programming since the HJB equation includes finding the controls (possibly bounded) that minimize the Hamiltonian at each point in the state space. A comprehensive introduction to calculus of variations and optimal control can be found in [33, 93], and for a more detailed historical perspective on the evolution of optimal control the reader is referred to [5, 32, 133].

[1]Pontryagin used the classical definition of the Hamiltonian, which is opposite in sign from the one commonly used today.

1.2 Motivation

The solution of general (realistic) trajectory optimization problems is a challenging task. Analytical solutions are seldom available or even possible. As a result, numerical methods must be employed in order to solve the trajectory optimization problems. However, the amount of numerical computation required for even a relatively simple problem is forbidding if it must be done by hand. This is why the calculus of variations and optimal control theory found very little use in engineering and applied science until the middle of XX-th century. The truly enabling element for the use of the optimal control theory was the digital computer, which became available commercially in 1950s. The development of economical, high-speed computers since then has dramatically changed the situation. These days, as will be discussed later in Section 1.4, there exist many numerical algorithms for solving optimal control problems but less work has been done in the field of making these algorithms run faster and more robustly. The main motivation of this work is to solve the optimal control problems accurately in a fast and efficient way.

Optimal control problems are often characterized by discontinuities or switchings in the control variables. One way of accurately capturing the irregularities in the solution is to use a high resolution (dense) uniform grid. This requires a large amount of computational resources both in terms of CPU time and memory. Hence, in order to accurately capture any irregularities in the solution using a few computational resources, one would like to refine the mesh locally in the region close to an irregularity instead of refining the mesh uniformly over the whole domain. To achieve this goal, we start by looking at what has been done in the field of partial differential equations (PDEs) for adaptive mesh refinement.

1.3 Adaptive Mesh Refinement for the Solution of Evolution PDEs

It is well known that the solution of evolution partial differential equations is often not smooth even if the initial data are smooth. For instance, shocks may develop in hyperbolic conservation laws. To capture discontinuities in the solution with high accuracy one needs to use a fine resolution grid. The use of a uniformly fine grid requires a large amount of computational resources in terms of both CPU time and memory. Hence, in order to solve evolution equations in a computationally efficient manner, the grid should adapt dynamically to reflect local changes in the solution.

Several adaptive gridding techniques for solving partial differential equations have been proposed in the literature. A nice survey of the early works on the subject can be found in [6, 138]. Currently, popular adaptive methods for solving PDEs are: (i) moving mesh methods [1, 2, 4, 7, 8, 38, 39, 50, 91, 98, 105, 106, 136], in which an equation is derived that moves a grid of a fixed number of finite difference cells or finite elements so as to follow and resolve any local irregularities in the solution; (ii) the so called "adaptive mesh refinement" method [10, 13, 14, 15], in which the mesh is refined locally based on the difference between the solutions computed on the coarser and the finer grids, and (iii) wavelet-based or multiresolution-based methods [3, 16, 68, 69, 75, 76, 87, 139, 140, 141], which take advantage of the fact that functions with localized regions of sharp transition can be very well compressed. Our proposed method falls under this latter category.

Mallat [102] formulated the basic idea of multiresolution analysis for orthonormal wavelets in $L^2(\mathbb{R})$. Harten [68, 69, 70] later proposed a general framework for multiresolution representation of data by integrating ideas from three different fields, namely, theory of wavelets, numerical solution of PDEs, and subdivision schemes. Recently, Alves et al. [3] proposed an adaptive multiresolution scheme, similar to the multiresolution approach proposed by Harten [68, 69] and Holmstrom [75] for solving hyperbolic PDEs. These approaches share similar underlying ideas. Namely, the first step is to interpolate the function values at the points belonging to a particular resolution level, from the corresponding points at the coarser level, and find the interpolative error at the points of that particular resolution level. Once this step has been performed for all resolution levels, all the points that have an interpolative error greater than a prescribed threshold are added to the grid, along with their neighboring points at the same level and the neighboring points at the next finer level. The main difference between these approaches is that in Harten's approach [68, 69], the solution for each time step is represented on the finest grid and one calculates the interpolative errors at all the points of the finest grid at each mesh refinement step. On the other hand, Holmstrom [75] and Alves et al. [3], compute the interpolative error only at the points that are in the adaptive grid. If a value that does not exist is needed, Holmstrom interpolates the required function value recursively from a coarser scale. Alternatively, Alves et al. [3] add to the grid the points that were used to predict the function values at all previously added points, in order to compute the interpolative error during the next mesh adaptation.

In this thesis, we propose a novel multiresolution scheme for data compression, which results in a higher compression rate compared to the multiresolution approach by Harten [68, 69, 70] for the same desired accuracy. Subsequently, we apply the proposed encoding scheme to solve initial-boundary value problems (IBVP) encountered in evolution PDEs and show that the proposed mesh refinement algorithm results in fewer points in the grid compared to the approach of Alves et al. [3].

Next, we give a literature survey on numerical methods for solving optimal control problems.

1.4 Numerical Methods for Solving Optimal Control Problems

As mentioned before, the solution of general (realistic) trajectory optimization problems is a challenging task. Analytical solutions are seldom available or even possible. As a result, most often than not, one resorts to numerical techniques [20, 21, 22, 24, 25, 26, 27, 28, 30, 52, 53, 56, 55, 67, 72, 104, 117, 118, 121, 124, 57, 73, 107, 100, 101, 36, 37, 131, 58]. Available numerical techniques for solving optimal control problems can be broadly divided into direct methods [20, 21, 22, 24, 25, 26, 27, 28, 30, 52, 53, 56, 55, 67, 72, 104, 117, 118, 121, 124] and indirect methods [57, 73, 107]. Indirect methods solve the necessary optimality conditions stated in terms of the adjoint differential equations, Pontryagin's minimum principle, and the associated transversality conditions. Direct methods, on the other hand, are based on discretizing the states and controls at a set of nodes, transforming the optimal control problem into a nonlinear programming (NLP) problem. The solution of the resulting NLP problem can be obtained using standard NLP solvers. A nice survey of available trajectory optimization methods can be found in [17] and [114]. Recently, hybrid methods that combine the analytic and numerical methods have also been proposed in the literature [36, 37] by Calise et al.

In recent years, direct transcription methods have become increasingly popular for solving trajectory optimization problems, the major reason being that in direct methods one does not require an analytic expression for the necessary conditions, which for complicated nonlinear dynamics can be intimidating. Moreover, incorporating state and control constraints is rather straightforward. Most importantly, experience has shown that direct methods tend to be more robust with respect to inaccurate initial guesses, hence they converge more easily. On the other hand, indirect methods result in more accurate overall solutions and provide more confidence in the (at least local) optimality of the obtained solution. Algorithms that aim at taking advantage of both direct and indirect methods by combining them into a single algorithm have been also proposed in the literature [132, 126].

Direct methods can be broadly classified as shooting methods [21, 27, 28, 30, 104, 121] and collocation methods [20, 22, 52, 53, 56, 55, 67, 72, 117, 118]. Shooting methods can be further subdivided into simple (or single) shooting methods [27, 30, 104, 121] and multiple shooting [21, 28] methods. In simple shooting the initial conditions, the final conditions, and the "parameters" make up the NLP variables. All states and controls are then represented using these NLP variables. The terminal conditions are the constraints and with each iteration of the NLP solver the trajectory is integrated and the terminal conditions evaluated. The fundamental difference between the simple shooting and multiple shooting methods is that the multiple shooting methods divide the time interval into multiple segments with there own initial conditions and which are integrated separately, that is on each segment the shooting is performed separately, and the values of the state variables at the junctions of these segments are also included as the optimization variables. Moreover, additional constraints are introduced enforcing continuity of the state from one segment to another. The effect of the controls is thus limited to corresponding segments, and the nonlinear effects of early controls on the latter parts of the trajectory is reduced. Hence, the multiple shooting technique is more robust compared to the simple shooting approach, where the small changes introduced early in the trajectory can propagate into very nonlinear changes at the end of the trajectory. However, in the case of multiple shooting, the number of NLP variables and constraints increases markedly over simple shooting implementations.

Direct collocation methods descretize the ordinary differential equations (ODEs), using collocation (or interpolation schemes) [120, 142] along with the introduction of collocation conditions as NLP constraints, together with the initial and terminal conditions. Direct collocation methods can be further subdivided into pseudospectral methods [52, 56, 55, 117, 118] and other collocation methods [20, 22, 53, 67, 72]. In a sense, "pseudospectral" is a synonym for "collocation" but the term "pseudospectral" is applied only when collocation is used with a basis of global functions like Chebyshev or Legenedre polynomials. The other difference between the pseudospectral and the rest of the collocation methods is that pseudospectral methods use differentiation, whereas typical collocation methods are based mainly on integration. In other words, pseudospectral methods rely on the discretization of the tangent bundle (roughly, the left-hand side of the differential equation, $\dot{x} = f(x, u, t)$), whereas most of collocation methods rely on the approximation of the vector field (the right-hand side).

Regardless of the particular method used, if a highly accurate solution is needed using one of the above mentioned direct methods, one must resort to the use of a high resolution (dense) grid. This choice results in a large amount of computational resources both in terms of CPU time and memory, especially if the resulting NLP problem is not sparse. Therefore, recent work has focus on the reduction of the high computational load associated with uniform grid discretizations. See, for instance, the work by Betts et al. [18, 20, 22], Ross and Fahroo [117, 118], Gong et al. [63], Binder et al. [24, 27, 25, 26], and Schlegel et al. [121].

The method of Betts et al. [18, 20, 22] selects the new grid points by solving an integer programming problem that minimizes the maximum discretization error (found by integrating the dynamics of the system) by subdividing the current grid. In [22], the authors computed the discretization error by comparing the solution with a more accurate estimate using two (half) steps and by keeping the control fixed. The authors also assumed that the order of discretization, which effects the addition of mesh points to any subinterval in their mesh refinement algorithm, is constant. However, during the course of optimization process the actual order may vary with each iteration because of the potential activation of path constraints. It has been shown in [23] that having the wrong value for the order of discretization can seriously impact the mesh refinement algorithm of [22]. In order to overcome this problem, Betts et al. [20] derived a formula for estimating the order reduction by comparing the behavior of the discretization errors on successive mesh refinement iterations. But since the estimated order reduction is very sensitive to the computed discretization errors, the authors in [20] use a highly accurate quadrature method, namely Romberg quadrature, with a tolerance close to machine precision for computing the discretization errors.

The pseudospectral knotting method introduced by Ross and Fahroo [117] breaks a single phase problem with discontinuities and switches in states, control, cost functional, or dynamic constraints into a multiple phase problem with the phase boundaries, termed as “knots” by the authors, as the point of discontinuities or switchings. This way states and controls are allowed to be discontinuous across the phase boundaries and the phase boundaries can be fixed or free. On each phase, the problem is solved using the Legendre pseudospectral method [52] or Chebyshev pseudospectral method [55], and the free knots are part of the optimization process. The knots where the states are assumed to be continuous but no continuity condition is imposed on the controls are termed as *soft knots*. The soft knots can handle problems with smooth data and non-smooth solutions (e.g. switches and corners). But as pointed out by Ross [116] “Soft knots do not increase the speed of the algorithm; they are expected to improve accuracy. Consequently, the introduction of soft knots in the grid might significantly slow the algorithm.” In order to improve the pseudospectral methods, Gong et al. [63] present an algorithm in which the user specifies the number of nodes to be increased in a particular phase, in case the error of the computed optimal control between two successive iterations is greater than a prescribed threshold. The authors of Ref. [63] use the gradient of the control to determine (approximately) the location of the knots. Binder et al [24, 27] use a wavelet-Galerkin approach to discretize the optimal control problem into an NLP problem. In Ref. [118], the authors use the domain transformation techniques for generating the adaptive grids.

Binder et al. [24, 25, 26] work in the wavelet space by using the wavelet-Galerkin approach to discretize the optimal control problem into an NLP problem and use the local error analysis of the states and the wavelet analysis of the control profile to add or remove the wavelet basis functions. In Ref. [27], the authors use a direct shooting approach, where the optimal control problem is converted into an NLP problem by parameterizing the control profile, combined with a wavelet analysis of the gradients of the Lagrangian function with respect to the parametrization functions at the optimal points in order to determine the regions that require refinement. For problems with state and/or control path constraints Schlegel et al. [121] use wavelet analysis of the control profile to determine the regions that require refinement.

In our continued effort on solving optimal control problems numerically [80, 81, 84], in this thesis, we have proposed a novel, fully automated, adaptive multiresolution-based trajectory optimization technique to solve optimal control problems quickly and accurately. The proposed technique does not require the solution of any secondary optimization problem for adding (or removing) points to the mesh, as done for instance, in Ref. [18, 20, 22]. Moreover, the criterion for deciding the region to refine the mesh is based on simple interpolations. Furthermore, the algorithm can add and remove points anywhere in the grid. Hence the grid can embrace any form depending on the irregularities in the solution, thus providing more flexibility in capturing any irregularities in the solution as opposed to the pseudospectral knotting method [117], where the grid on a particular phase is fixed.

Next, we give a literature survey on the numerical techniques for solving optimal control problems with moving targets and/or dynamically changing environments.

1.5 Trajectory Optimization for Moving Targets and/or Dynamically Changing Environments

A common line of attack for solving nonlinear trajectory optimization problems in real time [125, 100, 88, 144] is to break the problem into two phases: an offline phase and an online phase. The offline phase consists of solving the optimal control problem for various reference trajectories and storing these reference trajectories onboard for later online use. These reference trajectories are used to compute the actual trajectory online via a neighboring optimal feedback control strategy [31, 92, 130, 33] typically based on the linearized dynamics. This approach requires extensive ground-based analysis and onboard storage capabilities [94]. Moreover, perturbations around the reference trajectories might not be small, and therefore applying the linearized equations may not be appropriate.

To illustrate the previous point, consider the problem of finding the optimal control that will steer the system from point A to the target point B under certain path constraints at a minimum cost. If the target point B is far off, then there is no real advantage of finding the optimal trajectory online with high precision from the starting point till the end. As we continue to move from point A towards the target point B, we can get more accurate information about the surrounding environment (path constraints), which may be different from what was assumed at the beginning when the trajectory was optimized. Moreover, the path constraints and the terminal constraints may also change as the vehicle progresses towards point B. For example, the target point B may not be stationary. One way of handling this problem is to use the receding horizon approach [108, 11, 143], in which a trajectory that optimizes the cost function over a period of time, called the *planning horizon*, is designed. The trajectory is implemented over the shorter *execution time* and the optimization is performed again starting from the state that is reached at the end of the execution time. However, if the planning horizon length does not reach the target B, the trajectory found using this approach might not be optimal. One would like to solve the nonlinear trajectory optimization problem online for the whole time interval, but with high accuracy only near the current time. Recently, some work has been done in this direction by Kumar et al. [94] and Ross et al. [119]. Kumar and Seywald [94] proposed a dense-sparse discretization technique in which the trajectory is discretized by placing N_D dense nodes close to the current time and N_S sparse nodes for the rest of the trajectory. The state values at some future node are accepted as optimal and are prescribed as the initial conditions for the rest of the trajectory. The remainder of the trajectory is again discretized using a dense-sparse discretization technique, and the whole process is repeated again. The algorithm can be stopped by using any adhoc scheme, for example, it can be terminated when the density of the dense nodes is less than or equal to the density of the sparse nodes. Ross et al. [119] also proposed a similar scheme by solving the discretized NLP problem on a grid with a certain number of nodes and then propagate the solution from the prescribed initial condition by integrating the dynamics of the system for a specified interval of time. The values of the integrated states at the end of the integration interval are taken as the initial condition for solving the NLP problem for the rest of the trajectory, again on a grid with a fixed number of nodes. The whole process is repeated until the terminal conditions are met.

In this thesis, we present two algorithms that autonomously discretize the trajectory with more nodes (finer grid) near the current time (not necessarily uniformly placed) and use fewer nodes (coarser grid) for the rest of the trajectory, the latter to capture the overall trend. Furthermore, if the states or controls are irregular in the vicinity of the current time, the algorithm will automatically further refine the mesh in this region to capture the irregularities in the solution more accurately. The generated grid is fully adaptive and can embrace any form depending on the solution.

1.6 Organization of the Thesis

Since this work is multidisciplinary, every effort has been made to make this thesis self-contained. The thesis is organized as follows. Chapter 2 gives a brief introduction into wavelet multiresolution theory. In Chapter 3, we briefly describe the evolution equations, the difficulties encountered while solving the evolution equations, remedies for resolving these difficulties and also at the same time provide the reader with enough context to understand remarks made in the remainder of the thesis. In particular, we show that the solutions to the initial value problem for the conservation laws and Hamilton-Jacobi equations are not smooth in general, which is another motivation behind developing a novel multiresolution data compression algorithm described in Chapter 4. In Chapter 4, we present the proposed multiresolution scheme for data compression and compare the proposed scheme with the Harten's data compression scheme [68, 69, 70]. We show that the proposed algorithm results, in general, in a fewer number of grid points compared to Harten's approach [68, 69, 70]. In Chapter 5, we present a hierarchical multiresolution adaptive mesh refinement algorithm for the solution of evolution PDEs. The proposed grid adaptation method for the solution of evolution PDEs is then compared with the existing multiresolution schemes for the solution of evolution PDEs. This analysis is followed by several challenging numerical examples that show the robustness of the proposed approach and the advantages in terms of computational time compared to the uniform mesh case. Next, we move on to the optimal control part in Chapter 6. In Chapter 6, we first formulate the general optimal control problem and discretize the continuous optimal control problem into an NLP problem. We then present the multiresolution-based trajectory optimization algorithm followed by several nontrivial examples that show the robustness, efficiency and accuracy of the proposed algorithm. We conclude this chapter by giving advantages of the proposed algorithm over the current state-of-the-art adaptive algorithms for solving optimal control problems. In Chapter 7, we present two sequential trajectory optimization techniques for solving problems with moving targets and/or dynamically changing environments. Finally, the conclusions and several issues for the future study are proposed in Chapter 8.

CHAPTER II

WAVELET MULTIRESOLUTION THEORY

Wavelets and multiscale analysis have emerged in a number of different fields, from harmonic analysis and partial differential equations in pure mathematics, to signal and image processing in computer science and electrical engineering. Typically, a general function, signal, or image is broken up into linear combinations of translated and scaled versions of some simple, basic building blocks. Multiscale analysis comes with natural hierarchical structure obtained by only considering the linear combinations of building blocks up to a certain scale. This hierarchical structure is particularly suited for fast numerical implementations. In this chapter, we give a brief introduction into the theory of wavelets and multiresolution analysis. For details on any particular topic the reader is referred to the corresponding references.

2.1 Traditional Wavelets

The *traditional wavelets* are defined over the whole real line $\mathbb{R}$ and form two-parameter families of basis functions, which induce a multiresolution decomposition of $L^2(\mathbb{R})$ [34, 44, 102, 103]. This is the main property making wavelets attractive in applications. Specifically, wavelets induce the following nested sequence of subspaces,

$$\mathcal{V}_0 \subset \mathcal{V}_1 \subset \mathcal{V}_2 \cdots \subset \mathcal{V}_j \subset \mathcal{V}_{j+1} \subset \cdots \subset L^2(\mathbb{R}),$$

with the following properties.

Multiresolution Properties:

- $\bigcup_{j=0}^{\infty} \mathcal{V}_j$ is dense in $L^2(\mathbb{R})$, that is, $\overline{\bigcup_0^{\infty} \mathcal{V}_j} = L^2(\mathbb{R})$,
- $\bigcap_{j\geq 0} \mathcal{V}_j = 0$,
- $f(x) \in \mathcal{V}_j \Longleftrightarrow f(2x) \in \mathcal{V}_{j+1}, \forall\ j \geq 0$,
- $f(x) \in \mathcal{V}_j \Longleftrightarrow f(x - 2^{-j}k) \in \mathcal{V}_j, \forall\ j \geq 0$.

The "base" (or coarse-resolution) subspace $\mathcal{V}_0$ is spanned by integer translates of the *scaling function* ϕ:

$$\mathcal{V}_0 = \overline{\text{span}\{\phi(x-k)\}}, \quad k \in \mathbb{Z}. \tag{1}$$

The higher-resolution subspaces $\mathcal{V}_j$ are spanned by dilated versions of the scaling function:

$$\mathcal{V}_j = \overline{\text{span}\{2^{j/2}\phi(2^j x-k)\}}, \quad k \in \mathbb{Z},\ j \geq 0. \tag{2}$$

The orthogonal complement of $\mathcal{V}_j$ in the larger subspace $\mathcal{V}_{j+1}$ is denoted by $\mathcal{W}_j$ and it is spanned by the *wavelets*:

$$\mathcal{W}_j = \overline{\text{span}\{2^{j/2}\psi(2^j x-k)\}}, \quad k \in \mathbb{Z},\ j \geq 0, \tag{3}$$

where ψ is the *mother wavelet*, which spans the space $\mathcal{W}_0 = \mathcal{V}_1 \ominus \mathcal{V}_0$. Hence, we see that the traditional wavelets are characterized by the translation and dilation of a single function ψ. A pictorial representation of the subspaces $\mathcal{V}_j$ and $\mathcal{W}_j$ is given in Figure 1.

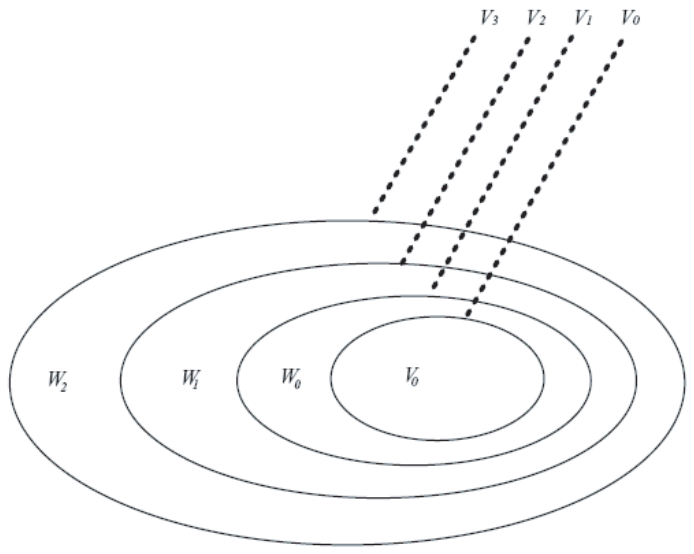


Figure 1: Pictorial representation of the subspaces $\mathcal{V}_j$ and $\mathcal{W}_j$.

For notational convenience, we define the two-parameter family of functions

$$\begin{aligned} \phi_{j,k}(x) &= 2^{j/2}\phi(2^j x-k), \quad j \geq 0,\ k \in \mathbb{Z}, & (4)\\ \psi_{j,k}(x) &= 2^{j/2}\psi(2^j x-k), \quad j \geq 0,\ k \in \mathbb{Z}. & (5) \end{aligned}$$

$L^2(\mathbb{R})$ can then be decomposed as

$$L^2(\mathbb{R}) = \mathcal{V}_0 \bigoplus_{j=0}^{+\infty} \mathcal{W}_j = \lim_{j\to\infty} \mathcal{V}_j, \tag{6}$$

that is, for all $f \in L^2(\mathbb{R})$,

$$\begin{aligned} f(x) &= \sum_{k\in\mathbb{Z}} c_{0,k}\phi_{0,k}(x) + \sum_{j\geq 0}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x) & (7)\\ &= \lim_{j\to\infty}\sum_{k\in\mathbb{Z}} c_{j,k}\phi_{j,k}(x), & (8)\end{aligned}$$

where

$$\begin{aligned} c_{j,k} &= \langle f, \phi_{j,k}\rangle_{L^2(\mathbb{R})} = \int_{-\infty}^{\infty} f(x)\phi_{j,k}(x)\mathrm{d}x, \quad j \geq 0,\ k \in \mathbb{Z}, & (9)\\ d_{j,k} &= \langle f, \psi_{j,k}\rangle_{L^2(\mathbb{R})} = \int_{-\infty}^{\infty} f(x)\psi_{j,k}(x)\mathrm{d}x, \quad j \geq 0,\ k \in \mathbb{Z}. & (10)\end{aligned}$$

The following fact is crucial for the approximating properties of wavelet decompositions.

Theorem 1 (Equivalent Characteristics [34])**.** *The following are equivalent:*

1. *The first μ moments of the wavelet ψ are zero, that is,*

$$\int x^\ell \psi(x)\mathrm{d}x = 0, \qquad \ell = 0, 1, \cdots, \mu - 1. \tag{11}$$

2. *All polynomials of degree up to $\mu - 1$ can be expressed as a linear combination of shifted scaling functions at any scale.*

Note that from the multiresolution properties, we have that $\phi(x) \in \mathcal{V}_0 \subset \mathcal{V}_1$. Hence, there exist coefficients h_k such that $\phi(x)$ satisfies

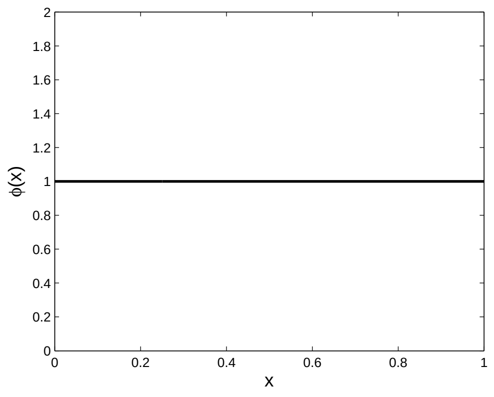
$$\phi(x) = \sum_k h_k \sqrt{2}\phi(2x - k), \quad k \in \mathbb{Z}. \tag{12}$$

Therefore, the scaling function is obtained by solving the above recursive equation (12). Now, since $\mathcal{W}_0 \subset \mathcal{V}_1$, and since the mother wavelet $\psi(x) \in \mathcal{W}_0$, there exist coefficients $\tilde{h}_k$ such that

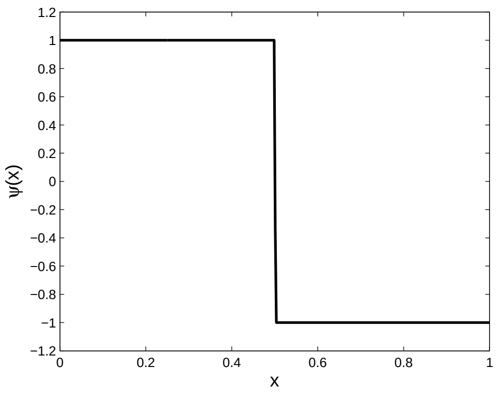
$$\psi(x) = \sum_k \tilde{h}_k \sqrt{2}\phi(2x - k), \quad k \in \mathbb{Z}. \tag{13}$$

Examples of some commonly used wavelets are Haar wavelets (Figure 2), Daubechies wavelets (Figure 3), symlets (Figures 4, 5), and coiflets (Figures 6, 7). It has been shown in the literature [34, 44] that the condition of orthogonality $\mathcal{V}_j \perp \mathcal{W}_j$ gives

$$\tilde{h}_k = (-1)^k h_{1-k}. \tag{14}$$

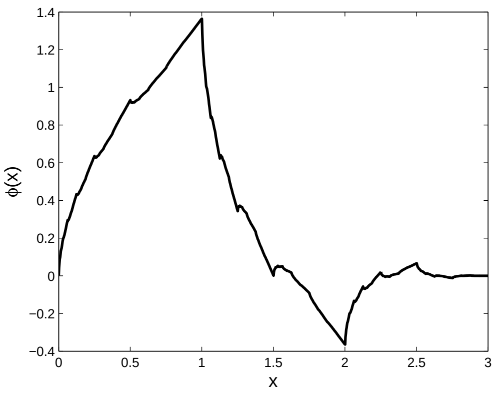


(a) Scaling function ($\phi(x)$).

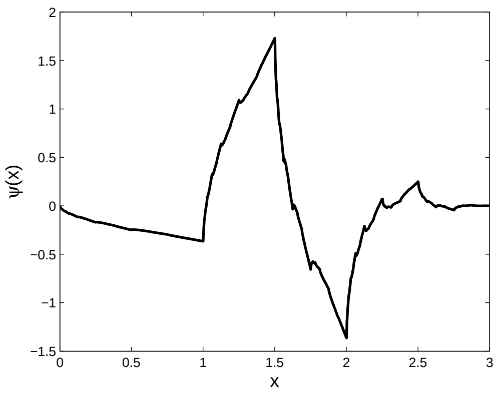


(b) Wavelet ($\psi(x)$).

Figure 2: Haar wavelets (Daubechies wavelets with $\mu = 1$).



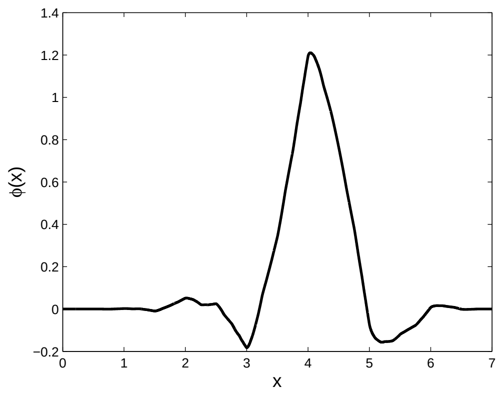
(a) Scaling function ($\phi(x)$).



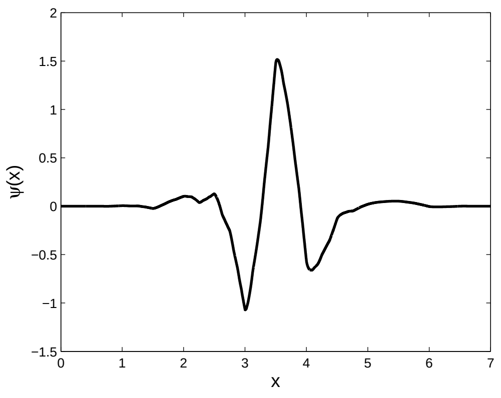
(b) Wavelet ($\psi(x)$).

Figure 3: Daubechies wavelets ($\mu = 2$).

In many applications, one never has to deal directly with the scaling functions or wavelets and only the coefficients h_k, $\tilde{h}_k$, $c_{j,k}$, and $d_{j,k}$ need to be considered. There exist following relationships between the coefficients h_k, $\tilde{h}_k$, $c_{j,k}$, and $d_{j,k}$ [34],

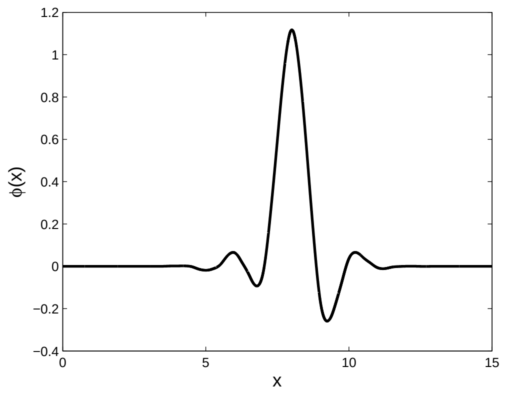


(a) Scaling function ($\phi(x)$).

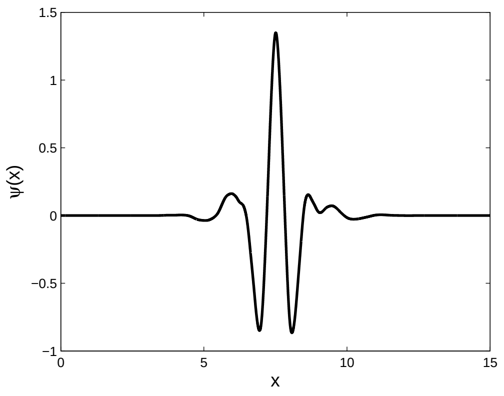


(b) Wavelet ($\psi(x)$).

Figure 4: Symlets ($\mu = 4$).



(a) Scaling function ($\phi(x)$).



(b) Wavelet ($\psi(x)$).

Figure 5: Symlets ($\mu = 8$).

From Fine Scale to Coarse Scale:

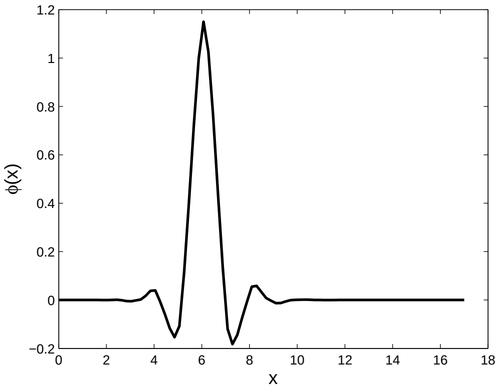
$$c_{j,k} = \sum_{\ell} h_{\ell-2k} c_{j+1,\ell}, \tag{15}$$

$$d_{j,k} = \sum_{\ell} \tilde{h}_{\ell-2k} c_{j+1,\ell}, \tag{16}$$

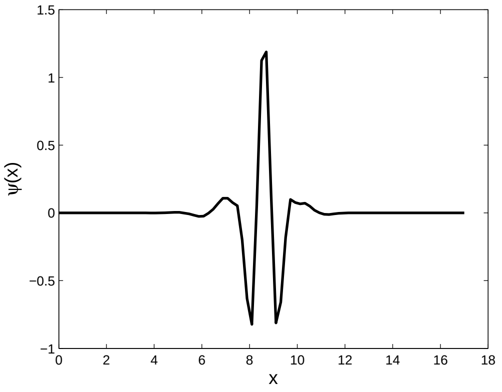
From Coarse Scale to Fine Scale:

$$c_{j+1,k} = \sum_{\ell} h_{k-2\ell} c_{j,\ell} + \sum_{\ell} \tilde{h}_{k-2\ell} d_{j,\ell}. \tag{17}$$

The traditional wavelets, discussed above, are usually constructed using Fourier techniques, although some traditional wavelets can be constructed without the use of Fourier

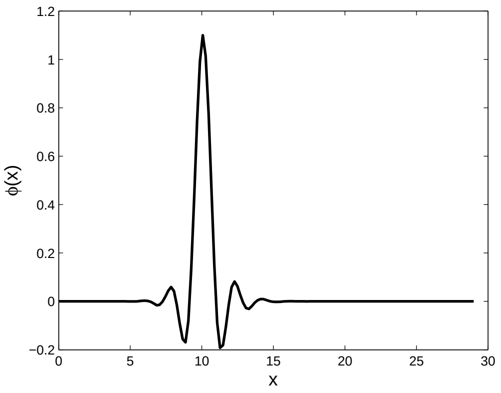


(a) Scaling function ($\phi(x)$).

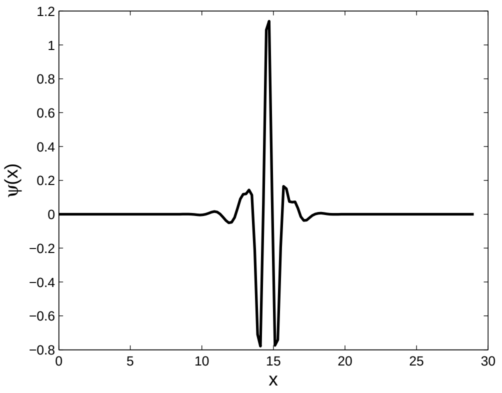


(b) Wavelet ($\psi(x)$).

Figure 6: Coiflets ($\mu = 3$).



(a) Scaling function ($\phi(x)$).



(b) Wavelet ($\psi(x)$).

Figure 7: Coiflets ($\mu = 5$).

techniques. Interpolating wavelets based on the interpolating subdivision scheme of Deslauriers and Dubuc [46], and independently discovered by Donoho [47] and Harten [68], are such an example and are discussed next.

2.2 Interpolating Wavelets

The interpolating wavelets are constructed on a set of dyadic grids of the form

$$\mathcal{V}_j = \{x_{j,k} \in \mathbb{R} : x_{j,k} = k/2^j,\ k \in \mathbb{Z}\}, \quad j \in \mathbb{Z}, \tag{18}$$

where j denotes the resolution level and k the spatial location. Note that since $x_{j,k} = x_{j+1,2k}$ it follows that $\mathcal{V}_j \subset \mathcal{V}_{j+1}$. Interpolating wavelets can be formally introduced through the interpolating subdivision scheme of Deslauriers and Dubuc [46], which considers the problem of building an interpolant $\hat{f}(x)$ on a grid $\mathcal{V}_{j+1}$ for a given data sequence $f(x_{j,k})$. Further, for simplicity of notations, we denote $f(x_{j,k})$ simply by $f_{j,k}$. Deslauriers and Dubuc defined a recursive procedure for interpolating the data $f_{j,k}$ to all dyadic points in between. The algorithm proceeds by interpolating the data $f_{j,k}$ to the points on a grid $\mathcal{V}_{j+1}$ which do not belong to $\mathcal{V}_j$. This procedure does not modify any of the existing data and thus can be repeated until the data are interpolated to all dyadic points up to the desired level of resolution. The interpolation is achieved by constructing local polynomials, $\hat{f}(x)$ of degree p, which uses $p+1$ closest points. For example, to find the value of the interpolant at location $x_{j+1,2k+1}$ we construct the polynomial of degree p based on the values of the function at locations $x_{j,k+\ell}$ $(\ell = -(p-1)/2, \dots, (p+1)/2)$ and evaluate it at location $x_{j+1,2k+1}$. Evaluating this polynomial at point $x_{j+1,2k+1}$ and substituting the values of polynomial coefficients expressed in terms of values $f_{j,k}$, we get that

$$\hat{f}(x_{j+1,2k+1}) = \sum_{\ell=-(p-1)/2}^{(p+1)/2} h_{j,k,\ell} f_{j,k+\ell}, \tag{19}$$

where $h_{j,k,\ell}$, $\ell = -(p-1)/2, \dots, (p+1)/2$, are the interpolating coefficients from even points $x_{j+1,2(k+\ell)}$ to odd point $x_{j+1,2k+1}$. The values of the interpolating coefficients are the same for the evenly spaced grid points. In other words, the interpolating coefficients are translation and dilation invariant for a uniform grid. For example, when the grid points are evenly spaced, we have

$$\{h_{j,k,\ell}\}_{\ell=0}^{1} = \left\{\frac{1}{2}, \frac{1}{2}\right\}, \tag{20}$$

for linear subdivision $(p = 1)$, and

$$\{h_{j,k,\ell}\}_{\ell=-1}^{2} = \left\{-\frac{1}{16}, \frac{9}{16}, \frac{9}{16}, -\frac{1}{16}\right\}, \tag{21}$$

for cubic subdivision $(p = 3)$. Examples of linear and cubic subdivisions are shown in Figure 8. In Figure 8, on the left, the linear subdivision step inserts new values in between the old values by averaging the two old neighbors, whereas on the right, cubic polynomials are used for every quad of old values to determine a new in between value.

The interpolating scaling function $\phi_{j,k}(x)$ is defined to be the result of running the subdivision scheme ad infinitum starting from a sequence $f_{j,\ell} = \delta_{\ell,k}$, where $\delta_{\ell,k}$ is the Kronecker

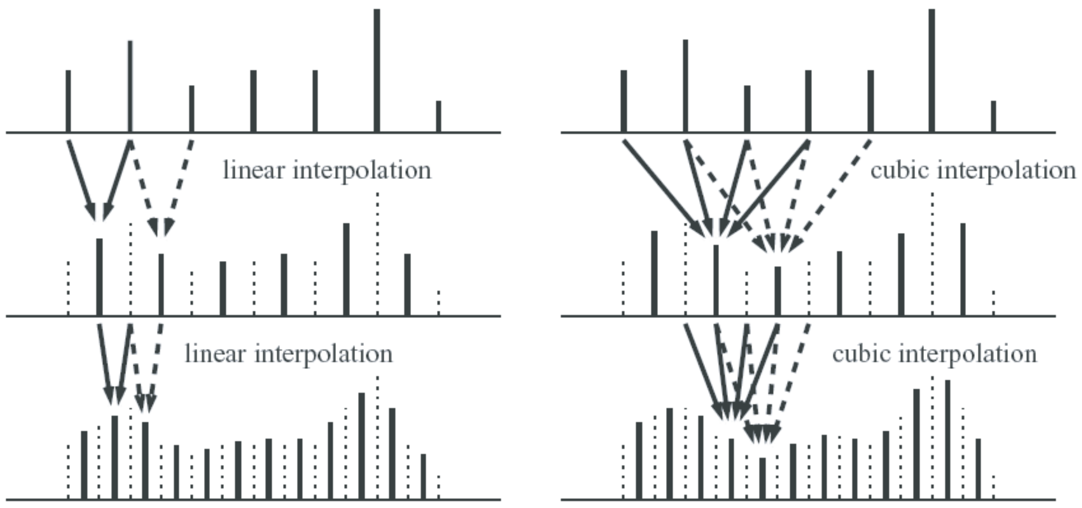


Figure 8: Examples of interpolating subdivision [135].

delta, and then performing the interpolating subdivision scheme up to an arbitrary high level of resolution. All scaling functions for the regularly spaced grid $\mathcal{V}_j$ are translates and dilates of one function $\phi(x) = \phi_{0,0}(x)$,

$$\phi_{j,k}(x) = \phi(2^j x - k), \tag{22}$$

called the *interpolating scaling function*, since $\phi(x)$ is interpolating in the sense that $\phi(0) = 1$ and $\phi(k) = 0$ for $k \neq 0$. The main feature of this approach is that the powerful properties such as approximation order and the connection with wavelets remain valid. The scaling function $\phi(x)$ resulting from the interpolating subdivision for different values of p, namely, 1, 3, 5, and 7 are shown in Figure 9.

Since the scaling functions are interpolating, then at a particular level j,

$$f(x) = \sum_k c_{j,k}\phi_{j,k}(x), \tag{23}$$

where $c_{j,k} = f_{j,k}$. Moreover, since $x_{j,k} = x_{j+1,2k}$, we have

$$c_{j,k} = c_{j+1,2k}. \tag{24}$$

Hence, if we set

$$d_{j,k}(x) = c_{j+1,2k+1} - \sum_\ell h_{j,k,\ell}c_{j+1,2(k+\ell)}, \tag{25}$$

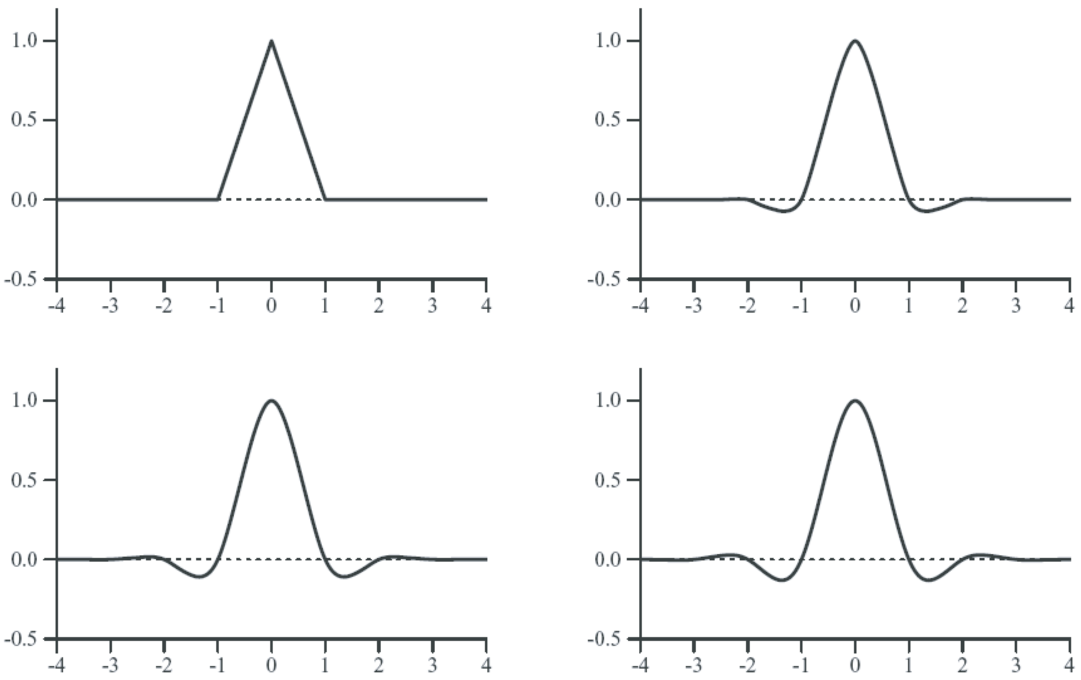


Figure 9: Scaling functions resulting from interpolating subdivision. Going from left to right, top to bottom, p is 1, 3, 5, 7 [135].

and

$$\psi_{j,k}(x) = \phi_{j+1,2k+1}(x), \tag{26}$$

then the forward wavelet transform can be written as

$$c_{j,k} = c_{j+1,2k}, \tag{27a}$$

$$d_{j,k} = c_{j+1,2k+1} - \sum_{\ell} h_{j,k,\ell} c_{j+1,2(k+\ell)}, \tag{27b}$$

while the inverse wavelet transform is given by

$$c_{j+1,2k} = c_{j,k}, \tag{28a}$$

$$c_{j+1,2k+1} = d_{j,k} + \sum_{\ell} h_{j,k,\ell} c_{j+1,2(k+\ell)}. \tag{28b}$$

While Mallat [102, 103] formulated the basic idea of multiresolution analysis for orthonormal wavelets in $L^2(\mathbb{R})$, Harten [70] later proposed a general framework for multiresolution representation of data by integrating ideas from three different fields, namely, theory of wavelets, numerical solution of PDEs, and subdivision schemes. His contribution to the theory of wavelets lies mainly in his extension of wavelets to nonuniform grids. The algorithm for constructing interpolating wavelets on a nonuniform grid is the same as described above, except that scaling functions and wavelets will not be dilates and translates of each other. In this case, the interpolating coefficients are location-dependent and are, in general, different. Further, in [134] Sweldens introduced a lifting scheme for constructing the wavelets that are not necessarily translates and dilates of each other and called such wavelets as *second generation wavelets*. Second generation wavelets maintain most of the useful properties of the traditional wavelets described above. For the sake of brevity, we will skip the details on the lifting scheme and the interested reader is referred to [134]. The interpolating wavelets defined on real line with evenly spaced grid are an example of traditional wavelets, while the extension to the irregular grids and intervals is an example of second-generation wavelets.

Next, we give a brief introduction to the evolution PDEs.

CHAPTER III

EVOLUTION PDES

Many problems in engineering and physics can be written in the form of an initial value problem (IVP) for an evolution equation,

$$\text{(IVP)}: \quad \begin{cases} u_t + f(u_{xx}, u_x, u, x) = 0 & \text{in } \mathbb{R} \times (0, \infty), \\ u = g & \text{on } \mathbb{R} \times \{t = 0\}, \end{cases} \tag{29}$$

where the function $f : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$, and the initial function $g : \mathbb{R} \to \mathbb{R}^m$ are given. The unknown is the function $u : \mathbb{R} \times [0, \infty) \to \mathbb{R}^m$. Such PDEs are often called *evolution equations*, the idea being that the solution evolves in time from a given initial configuration.

Our goal is to solve such PDEs. But what it means to "solve" a given PDE can be subtle, depending in large part on the particular structure of the problem at hand. The informal notion of a "well-posed problem" widely used in the study of PDEs captures many of the desirable features of what it means to solve a PDE.

A given problem for a PDE is said to be *well-posed* if

WP 1: the problem in fact has a solution;

WP 2: this solution is unique;

WP 3: the solution depends continuously on the data given in the problem.

Now it would be desirable to solve a PDE in such a way that WP 1 - WP 3 hold. But what is a solution? Should u be real analytic or at least a solution of a PDE of order k be at least k times continuously differentiable. Then at least all the derivatives which appear in the statement of the PDE will exist and be continuous, although maybe certain higher derivatives will not exist. A solution with this much smoothness is referred to as the *classical solution* of the PDE. In reality, it is not possible to solve many PDEs in a classical sense.

Hence, one looks for a wider class of candidates for solutions satisfying the well-posedness conditions WP 1 - WP 3. Such solutions are called *weak or generalized solutions.* Hence, our goal is to find numerically a weak solution to any well-posed evolution equation.

The multiresolution mesh refinement approach for solving evolution PDEs proposed in Chapter 5 will work for any evolution PDE, but the PDEs that are mainly of interest to us are nonlinear conservation laws and Hamilton-Jacobi (HJ) equations. The reason being that the IVP for the nonlinear conservation laws and the HJ equations do not, in general, have a smooth solution lasting for all times $t > 0$ even if the initial condition is smooth. Hence, in the next sections we will briefly discuss these equations, namely, we will describe why these equations can have non-smooth solutions, and define a notion of weak solution for both the nonlinear conservation laws and the HJ equations. The only purpose of this chapter is to familiarize the reader with the difficulties encountered while solving nonlinear conservation laws, HJ equations, and at the same time provide the reader with enough context to understand remarks made in the remainder of the thesis. Therefore, to keep things simple, for further analysis in this chapter we will assume $u : \mathbb{R} \times [0, \infty) \to \mathbb{R}$. But before going into the details of nonlinear scalar conservation laws and HJ equations, we briefly describe the method of characteristics for solving a basic nonlinear first order PDE which we will use to show as to why the nonlinear conservation laws and the HJ equations can have non-smooth solutions.

3.1 Method of Characteristics

Consider the IVP for a basic nonlinear first-order PDE

$$G(D_x u, u, \mathbf{x}) = 0 \qquad \text{in } \mathcal{U}, \tag{30a}$$

$$u = g \qquad \text{on } \Gamma, \tag{30b}$$

where $\mathcal{U}$ is an open subset of $\mathbb{R}^2$, $\Gamma \subseteq \partial\mathcal{U}$, and $D_x u = [u_{x_1}, u_{x_2}]$. The function $G : \mathbb{R}^2 \times \mathbb{R} \times \overline{\mathcal{U}} \to \mathbb{R}$, and the initial function $g : \Gamma \to \mathbb{R}$ are given. The unknown is the function $u : \overline{\mathcal{U}} \to \mathbb{R}$. G, g are supposed to be smooth functions.

The basic idea behind the method of characteristics is to convert the PDE (30a) into a system of ODE's (called *characteristics*). Suppose we want to know the solution u of (30) at any point $\mathbf{x} \in \mathcal{U}$, that is, find $u(\mathbf{x})$. Then, the goal of the method is to find some curve lying within $\mathcal{U}$, connecting $\mathbf{x}$ with a point $\mathbf{x}^0 \in \Gamma$ and along which we can compute u. Since from (30b) we know $u(\mathbf{x}^0) = g(\mathbf{x}^0)$, the idea is to be able to compute u all along that curve, so in particular at $\mathbf{x}$.

To this end, let us suppose that the curve be parametrically described by the function

$$\mathbf{x}(s) = [x_1(s), x_2(s)], \tag{31}$$

where s lies in some subinterval of $\mathbb{R}$. Assume $u \in C^2$, and define

$$z(s) = u(\mathbf{x}(s)), \tag{32}$$

$$\mathbf{p}(s) = D_x u(\mathbf{x}(s)), \tag{33}$$

that is, $\mathbf{p}(s) = [p_1(s), p_2(s)]$, where

$$p_i(s) = u_{x_i}(s), \quad \text{for } i = 1, 2. \tag{34}$$

So $z(\cdot)$ gives the value of u along the curve and $\mathbf{p}(\cdot)$ records the values of the gradient $D_x u$. Then the following system of 5 first-order ODEs [54],

$$\begin{aligned}
\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}s}(s) &= -D_x G(\mathbf{p}(s), z(s), \mathbf{x}(s)) - D_z G(\mathbf{p}(s), z(s), \mathbf{x}(s))\mathbf{p}(s), && \text{(35a)}\\
\frac{\mathrm{d}z}{\mathrm{d}s}(s) &= D_p G(\mathbf{p}(s), z(s), \mathbf{x}(s)) \cdot \mathbf{p}(s), && \text{(35b)}\\
\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s}(s) &= D_p G(\mathbf{p}(s), z(s), \mathbf{x}(s)), && \text{(35c)}
\end{aligned}$$

comprise the characteristic equations of the nonlinear first-order PDE (30a). The functions $\mathbf{p}(\cdot) = [p_1(\cdot), p_2(\cdot)]$, $z(\cdot)$, $\mathbf{x}(\cdot) = [x_1(\cdot), x_2(\cdot)]$ are called the *characteristics.*

Theorem 2 (Structure of Characteristics [54])**.** *Let $u \in C^2(\mathcal{U})$ solve the nonlinear PDE (30a) in $\mathcal{U}$. Assume $\mathbf{x}(\cdot)$ solves the ODE (35c), where $\mathbf{p}(\cdot) = D_x u(\mathbf{x}(\cdot))$, $z(\cdot) = u(\mathbf{x}(\cdot))$. Then $\mathbf{p}(\cdot)$ solves the ODE (35a) and $z(\cdot)$ solves the ODE (35b) for those s such that $\mathbf{x}(s) \in \mathcal{U}$.*

Now we move on to the discussion of the nonlinear conservation laws and HJ equation.

3.2 Conservation Laws

3.2.1 Introduction

The class of conservation laws is a very important class of PDEs because as their name indicates, they include those equations that model conservation laws of physics (mass, momentum, energy etc.). Conservation laws are generally nonlinear.

Consider the IVP for the scalar conservation laws

$$u_t + F(u)_x = 0 \qquad \text{in } \mathcal{U} = \mathbb{R} \times (0, \infty), \tag{36a}$$

$$u = g \qquad \text{on } \Gamma = \mathbb{R} \times \{t = 0\}. \tag{36b}$$

Here, $F : \mathbb{R} \to \mathbb{R}$, $g : \mathbb{R} \to \mathbb{R}$ are given and $u : \mathbb{R} \times (0, \infty) \to \mathbb{R}$ is the unknown, $u = u(x, t)$. Equation (36a) is said to be in conservation form and is called a *conservation law.*

To understand the physical significance of the conservation laws, we integrate equation (36a) with respect to x and t from a to b and t_1 to t_2 respectively, where $a,\ b \in \mathbb{R}$ and $t_1,\ t_2 \in [0, \infty)$. Performing the integration with respect to t for the first term and with respect to x for the second term, we obtain

$$\int_a^b u(x, t_2)\mathrm{d}x - \int_a^b u(x, t_1)\mathrm{d}x = -\left(\int_{t_1}^{t_2} F(u(b, t))\mathrm{d}t - \int_{t_1}^{t_2} F(u(a, t))\mathrm{d}t\right). \tag{37}$$

Equation (37) is referred to as the *integral form of conservation law.* In (37), u is the density of the “conserved material” (whatever material the conservation law is conserving); $\int_a^b u(x, t_1)\mathrm{d}x$, $\int_a^b u(x, t_2)\mathrm{d}x$ are the amount of conserved material in the interval $[a, b]$ at times t_1, t_2 respectively; and $F(u)$ is vaguely defined to be the flux function. Then the physical interpretation of (37) is that the difference in the “amounts of material” entering and/or leaving the control volume $[a, b] \times [t_1,\ t_2]$ across the top and bottom, $t = t_1$ and $t = t_2$, is balanced by the amount of material entering/or leaving the sides, $x = a$ and $x = b$.

After giving a physical interpretation of the conservation laws, we derive the characteristics for conservation laws and show that the solution to the nonlinear conservation laws, in general, is not smooth.

3.2.2 Characteristics for Conservation Laws

For finding the characteristics of the scalar conservation law (36), we set $\mathbf{x}(s) = [x(s),\ t(s)]$, $\mathbf{p}(s) = [u_x(x(s), t(s)),\ u_t(x(s), t(s))]$. Then we have

$$G(\mathbf{p}(s), z(s), \mathbf{x}(s)) = u_t(x(s), t(s)) + F'(z(s))u_x(x(s), t(s)), \tag{38}$$

and consequently

$$\begin{aligned} D_p G &= [F'(z(s)),\ 1], && (39)\\ D_x G &= 0, && (40)\\ D_z G &= F''(z(s))u_x(x(s), t(s)). && (41) \end{aligned}$$

Hence, equation (35c) becomes

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}s}(s) &= F'(z(s)), \\ \frac{\mathrm{d}t}{\mathrm{d}s}(s) &= 1. \end{cases} \tag{42}$$

Therefore, $t(s) = s$, since $t(0) = 0$. In other words, we can identify the parameter s with the time t.

Equation (35b) becomes

$$\begin{aligned} \frac{\mathrm{d}z}{\mathrm{d}s}(s) &= D_p G \cdot p && (43)\\ &= F'(z(s))u_x(x(s), s) + u_t(x(s), s) && (44)\\ &= 0 \quad \text{by (36a).} && (45) \end{aligned}$$

Consequently,

$$z(s) = z^0 = g(x^0); \tag{46}$$

and (42) implies

$$x(s) = F'(g(x^0))s + x^0. \tag{47}$$

Thus the projected characteristic $\mathbf{x}(s) = (x(s), s) = (F'(g(x^0))s + x^0, s)$ $(s \geq 0)$ is a straight line, along which u is constant.

Remark 1 (Crossing characteristics.). But suppose now we apply the same reasoning to a different initial point $\hat{x}^0 \in \Gamma$, where $g(x^0) \neq g(\hat{x}^0)$. The projected characteristics may possibly then intersect at some time $t > 0$. Since Theorem 2 tells us $u = g(x^0)$ on the projected characteristic through x^0 and $u = g(\hat{x}^0)$ on the projected characteristic through $\hat{x}^0$, an apparent contradiction arises. The resolution is that the IVP (36) does not, in general, have a smooth solution, existing for all times $t > 0$.

The method of characteristics demonstrated that there does not in general exist a smooth solution of (36) existing for all times $t > 0$. Therefore, we look for a weak or generalized solution to (36).

3.2.3 Weak Solutions

Define the set of *test functions*, C_0^1 to be the set

$$\{v \in C^1 : \{(x,t) \in \mathbb{R} \times [0,\infty) : v(x,t) \neq 0\} \subset [a,b] \times [0,T] \text{ for some } a,\ b \text{ and } T\}. \tag{48}$$

If we multiply PDE (36a) by $v \in C_0^1$ and integrate with respect to x from $-\infty$ to ∞ and with respect to t from 0 to ∞, we get

$$\begin{aligned}
0 &= \int_0^\infty \int_{-\infty}^\infty [u_t + F(u)_x] v \ \mathrm{d}x\mathrm{d}t && (49)\\
&= \int_0^T \int_a^b [u_t + F(u)_x] v \ \mathrm{d}x\mathrm{d}t && (50)\\
&= \int_a^b \int_0^T u_t v \ \mathrm{d}t\mathrm{d}x + \int_0^T \int_a^b F(u)_x v \ \mathrm{d}x\mathrm{d}t && (51)\\
&= \int_a^b \left\{ \big[uv\big]_{t=0}^{t=T} - \int_0^T u v_t \ \mathrm{d}t \right\} \mathrm{d}x + \int_0^T \left\{ [F(u)v]_{x=a}^{x=b} - \int_a^b F(u) v_x \ \mathrm{d}x \right\} \mathrm{d}t && (52)\\
&= -\int_a^b u(x,0)v(x,0) \ \mathrm{d}x - \int_a^b \int_0^T u v_t \ \mathrm{d}t\mathrm{d}x - \int_0^T \int_a^b F(u) v_x \ \mathrm{d}x\mathrm{d}t, && (53)
\end{aligned}$$

since $v(x,T) = v(a,t) = v(b,t) = 0$. We note that since the support of v is contained in $[a,b] \times [0,T]$ and v is defined on $\mathbb{R} \times [0,\infty)$, $v(x,0)$ need not be zero. We can rewrite (49)-(53) as

$$0 = \int_0^\infty \int_{-\infty}^\infty [u v_t + F(u) v_x] \ \mathrm{d}x\mathrm{d}t + \int_{-\infty}^\infty g v(x,0) \ \mathrm{d}x. \tag{54}$$

The above equality was derived assuming that u is a smooth solution of (36), but the resulting formula has meaning even if u is only bounded. Hence, we define the notion of weak solution for the conservation laws as follows:

Definition 1. If u satisfies (54) for all $v \in C_0^1$, u is said to be a *weak solution* to IVP (36).

Next, we explain the concepts of "shocks" and "fans" with the help of some simple examples.

Example 1

Consider the Burger's equation

$$u_t + \left(\frac{1}{2}u^2\right)_x = 0, \tag{55}$$

with initial condition

$$u(x,0) = \begin{cases} 1, & x \leq 0, \\ 0, & x > 0. \end{cases} \tag{56}$$

The characteristic curves associated with the above IVP are shown in Figure 10(a). We see in Figure 10(a) that the characteristic curves associated with the above problem intersect. Hence, we need to consider a weak solution.

It is easy to verify that

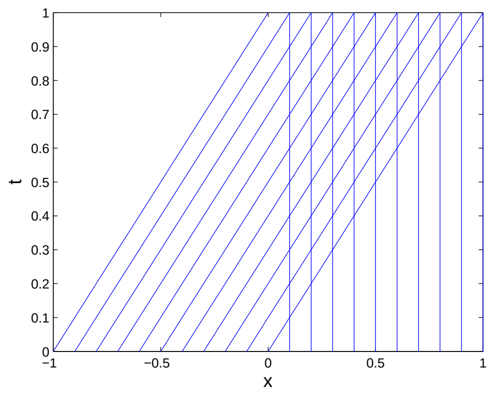
$$u(x,t) = \begin{cases} 1, & x \leq t/2, \\ 0, & x > t/2. \end{cases} \tag{57}$$

is a weak solution to IVP (55)-(56). The example demonstrates that a solution that is obviously not a classical solution can still be a weak solution. The weak solution given in (57) is associated with the characteristic curves given in Figure 10(b). The solution on the characteristics emanating from x, $x < 0$ is different from that on the characteristics emanating from x, $x > 0$. Hence, there is a discontinuity along the curve $x = t/2$.

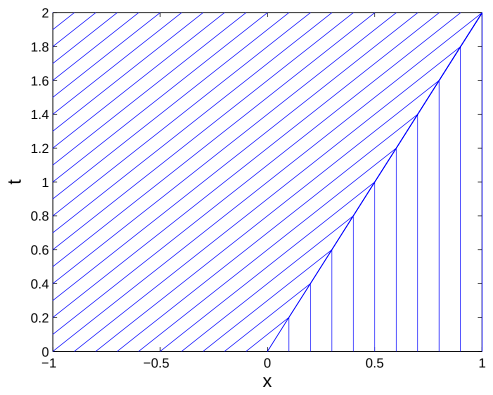
One should note that by the form of the solution (57) the discontinuity in the solution propagates along the curve $x = t/2$. Hence, the speed of propagation of the discontinuity is $\mathrm{d}x/\mathrm{d}t = \frac{1}{2}$.

A more formal definition of a shock will be given later in this chapter but for the time being we define the shock as follows.

Definition 2. A discontinuity of a piecewise continuous weak solution is called a *shock* if the characteristics on both sides of the discontinuity impinge on the discontinuity curve in the direction of increasing t.



(a) Characteristic curves associated with the IVP (55)-(56).



(b) Characteristic curves associated with the solution given in Example 1 to IVP (55)-(56).

Figure 10: Characteristic curves associated with the IVP (55)-(56) and the solution given in Example 1.

If we let $a_L = F'(u_L)$ and $a_R = F'(u_R)$, where u_L and u_R are the values of u on the left and right sides of the discontinuity, then a discontinuity will be a shock if

$$a_L > \sigma > a_R, \tag{58}$$

where σ is the speed of propagation of the discontinuity.

The discontinuity in the weak solution (57) of IVP considered in Example 1 is a shock (Figure 10(b)). In the above example, $a_L = 1$, $a_R = 0$ and $\sigma = \frac{1}{2}$, so the above inequality is satisfied.

Next, we consider another example.

Example 2

In this example, we again consider the Burger's equation (55) but with different initial condition

$$u(x,0) = \begin{cases} 0, & x \leq 0, \\ 1, & x > 0. \end{cases} \tag{59}$$

It can be easily shown that

$$u(x,t) = \begin{cases} 0, & x \leq t/2, \\ 1, & x > t/2, \end{cases} \tag{60}$$

and

$$u(x,t) = \begin{cases} 0, & x < 0, \\ x/t, & 0 \le x \le t, \\ 1, & x > t, \end{cases} \tag{61}$$

are both the weak solutions to the Burger's equation (55) with the initial condition given in (59). Hence, we see that the weak solutions to IVP for conservation laws are not unique.

The characteristics associated with the IVP given by Burger's equation (55) and initial condition (59) are shown in Figure 11. We see that because the slope of the characteristic curves for $x < 0$ is greater that the slope of the characteristic curves for $x > 0$, there is a region that has no characteristics. The solution (60) corresponds to filling in this region that has no characteristic curves with characteristics that come out of the curve $t = 2x$, as shown in Figure 12(a). Since the characteristics on either side of the curve $x = t/2$ emanate from the discontinuity, the discontinuity in solution (60) is not a shock.

The solution given in (61) corresponds to filling in the region that has no characteristic curves with a "fan" of characteristics as is shown in Figure 12(b). We saw that we were able to "fill in" the missing characteristics in at least two different ways that are compatible with the weak formulation of the problem. However, solutions found by filling in a region with a fan are the desired solutions (shown in the next section).

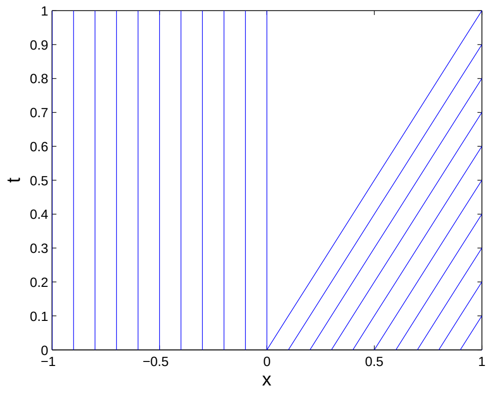
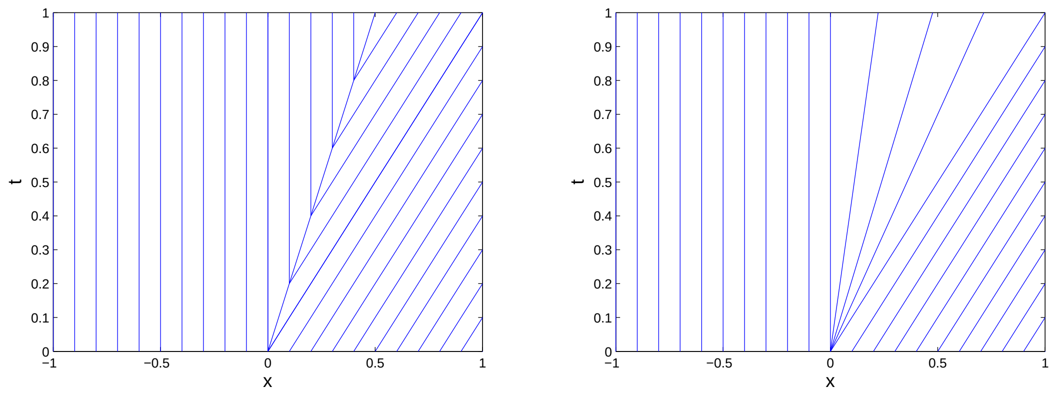


Figure 11: Characteristic curves associated with the IVP (55), (59).



(a) Characteristic curves associated with the solution (60) given in Example 2 to IVP (55), (59).

(b) Characteristic curves associated with the solution (61) given in Example 2 to IVP(55), (59).

Figure 12: Characteristic curves associated with the solutions given in Example 2 to IVP (55), (59).

3.2.4 Entropy Condition and Vanishing Viscosity Solution

Weak solutions to conservation laws can contain discontinuities that are due to a discontinuity in the initial condition or due to characteristics that cross each other, but any weak solution to an IVP (36) must satisfy across any jump discontinuity the following condition.

Proposition 1 (Rankine-Hugoniot condition [137]). *Let C be a smooth curve in $x-t$ space $(\mathbb{R}\times[0,\infty))$, $x_C = x_C(t)$, across which u, a weak solution to IVP (36), has a jump discontinuity. Let $P=(x_0,t_0)$, $t_0>0$, be any point on C and u_L and u_R be the values of u evaluated to the left and the right of P, respectively. Then*

$$(u_L-u_R)\frac{\mathrm{d}x_C}{\mathrm{d}t} = F(u_L)-F(u_R). \tag{62}$$

$\sigma = \frac{\mathrm{d}x_C}{\mathrm{d}t}$ is the speed of propagation of the discontinuity and equation (62) is referred to as the *jump condition* or the *Rankine-Hugoniot condition*. Observe that the speed σ and the values u_L, u_R, $F(u_L)$, $F(u_R)$ will generally vary along the curve C. The point is that even though these quantities may change, the expressions $\sigma(u_L-u_R)$ and $F(u_L)-F(u_R)$ must always exactly balance.

In addition, we saw that, in general, weak solutions to conservation laws are not unique. One way of choosing the correct solution is to choose the solutions that are limits of an associated viscous problem as the viscosity vanishes (which are generally called the *vanishing* *viscosity solutions*). Hence, we want solutions to equation (36a) that are limits of solutions to

$$u_t + F(u)_x = \nu u_{xx} \tag{63}$$

as $\nu \to 0$.

Proposition 2 ([137]). *If a vanishing viscosity solution exists, it is a weak solution.*

As shown in Section 3.2.2 using the method of characteristics that the solution u of the scalar conservation law (36a), whenever smooth, takes the constant value $z^0 = g(x^0)$ along the projected characteristic

$$\mathbf{x}(s) = (F'(g(x^0))s + x^0, s) \quad (s \geq 0). \tag{64}$$

Now we know that typically we will encounter the crossing of characteristics, and resultant discontinuities in the solution, if we move *forward* in time. However, we can hope that if we start at some point in $\mathbb{R} \times (0, \infty)$ and go *backwards* in time along a characteristic, we will not cross any others. In other words, let us consider the class of, say, piecewise-smooth weak solutions of (36) with the property that if we move backwards in t along any characteristic, we will not encounter any lines of discontinuity for u.

So now suppose at some point on a curve C of discontinuities that u has distinct left and right limits, u_L and u_R, and that the characteristic from the left and a characteristic from the right hit C at this point. Then in view of the above equation we have

$$F'(u_L) > \sigma > F'(u_R). \tag{65}$$

These inequalities are called the *entropy condition* (from a rough analogy with the thermodynamic principle that physical entropy cannot decrease as time goes forward).

Remark 2.
1. In Example 1: $F'(u_L) = 1,\ \sigma = \frac{1}{2}$ and $F'(u_R) = 0$, hence the solution satisfies the entropy condition.
2. In Example 2 with weak solution (60): $F'(u_L) = 0,\ s = \frac{1}{2}$ and $F'(u_R) = 1$, hence the solution does not satisfy the entropy condition.

3. In Example 2 with weak solution (61): the solution satisfies the entropy condition vacuously since there are no discontinuities in the solution.

In view of the entropy condition, we give a formal definition of shock as follows.

Definition 3. A curve of discontinuity for u is called a *shock* provided both the Rankine-Hugoniot and the entropy conditions are satisfied.

Proposition 3 ([137])**.** *Suppose that F is convex and that the solution to the IVP (36) satisfies the entropy condition across all jumps. Then the solution u is the unique viscosity solution to the IVP (36) that satisfies entropy condition and is a vanishing viscosity solution to IVP (36).*

The nonconvex analogue to the Entropy condition mentioned above is as follows:

Definition 4 ([137])**.** The solution to equation (36a) (where F is not necessarily convex), $u = u(x,t)$, containing a discontinuity is said to satisfy entropy condition if

$$\frac{F(u_L) - F(u)}{u_L - u} \geq \frac{F(u_R) - F(u_L)}{u_R - u_L} \tag{66}$$

for all u between u_L and u_R, where u_L and u_R are the solution values to the left and right of the discontinuity, respectively.

As for the case where F is convex, if F is not convex, the solution u is unique and is a vanishing viscosity solution if u satisfies the entropy condition (66) across all jumps.

3.2.5 Discrete Conservation Form

In order to solve the IVP for the conservation laws (36) numerically, we must write the conservation law (36a) in a discrete form. To this end, we assume that we are given a nonuniform grid of the form[1]

$$\begin{aligned}\mathsf{Grid} = \{&x_{j_i,k_i} : x_{j_i,k_i} = k_i/2^{j_i} \in [0,1],\ 0 \leq k_i \leq 2^{j_i},\ J_{\min} \leq j_i \leq J_{\max},\ \text{for } i = 0 \ldots N,\\ &\text{and } x_{j_i,k_i} < x_{j_{i+1},k_{i+1}},\ \text{for } i = 0 \ldots N-1\},\end{aligned} \tag{67}$$

[1]It should be noted that for solving IVP for the conservation laws (36) numerically the domain should be bounded, and hence, without loss of generality, we consider the nonuniform grid on the interval $[0, 1]$.

where $J_{\min},\ J_{\max} \in \mathbb{Z}_0^+$.

For simplicity of notations, we define the cell walls by

$$x_{j_{i-1/2},k_{i-1/2}} = \frac{x_{j_{i-1},k_{i-1}} + x_{j_i,k_i}}{2}, \quad x_{j_{i+1/2},k_{i+1/2}} = \frac{x_{j_i,k_i} + x_{j_{i+1},k_{i+1}}}{2}, \tag{68}$$

and we denote $u(x,t)$ evaluated at $x = x_{j,k}$ and $t = t_n$ by $u^n_{j,k}$, where $0 \le k \le 2^j,\ J_{\min} \le j \le J_{\max},\ n \in \mathbb{Z}_0^+$, $t_0 = 0$, $t_n = t_{n-1} + \Delta t_n$ for $n > 0$, and Δt_n is the time step based on the Courant-Friedrichs-Levy (CFL) condition [137].

The CFL condition asserts that the numerical waves should propagate at least as fast as the physical waves. This means that the numerical wave speed of $(x_{j_{i+1},k_{i+1}} - x_{j_i,k_i})/\Delta t_{n+1}$ must be at least as fast as the physical wave speed $|F'(u)|$. This leads us to the CFL time step restriction of

$$\Delta t_{n+1} < \frac{\min_{i=0,\dots,N-1}(x_{j_{i+1},k_{i+1}} - x_{j_i,k_i})}{\max_x\{|F'(u)|\}}. \tag{69}$$

The above equation (69) is usually enforced by choosing a CFL number α with

$$\Delta t_{n+1} \left(\frac{\max_x\{|F'(u)|\}}{\min_{i=0,\dots,N-1}(x_{j_{i+1},k_{i+1}} - x_{j_i,k_i})} \right) = \alpha, \tag{70}$$

and $0 < \alpha < 1$.

To ensure that shocks and other steep gradients move at the right speed, equation (36a) should be written in a discrete conservation form, that is, a form in which the rate of change of conserved quantities is equal to a difference of fluxes [97]. Hence, it has been shown in the literature [109], that equation (36a) should be approximated by

$$u^{n+1}_{j_i,k_i} = u^n_{j_i,k_i} - \frac{\Delta t_{n+1}}{x_{j_{i+1/2},k_{i+1/2}} - x_{j_{i-1/2},k_{i-1/2}}} (\mathcal{F}^n_{j_{i+1/2},k_{i+1/2}} - \mathcal{F}^n_{j_{i-1/2},k_{i-1/2}}), \tag{71}$$

where $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}} = \mathcal{F}(x_{j_{i\pm1/2},k_{i\pm1/2}}, t_n)$, and $\Delta t_{n+1}\mathcal{F}^n_{j_{i+1/2},k_{i+1/2}}$, $\Delta t_{n+1}\mathcal{F}^n_{j_{i-1/2},k_{i-1/2}}$ approximate the flux of material across the sides $x = x_{j_{i+1/2},k_{i+1/2}}$ and $x = x_{j_{i+1/2},k_{i+1/2}}$, respectively. The approximate fluxes are written as

$$\begin{aligned} \mathcal{F}^n_{j_{i+1/2},k_{i+1/2}} &= \mathcal{F}(u^n_{j_{i-\ell},k_{i-\ell}}, \dots, u^n_{j_{i+m},k_{i+m}}), & (72)\\ \mathcal{F}^n_{j_{i-1/2},k_{i-1/2}} &= \mathcal{F}(u^n_{j_{i-\ell-1},k_{i-\ell-1}}, \dots, u^n_{j_{i+m-1},k_{i+m-1}}), & (73) \end{aligned}$$

if $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}}$ depends on u at $\ell + m + 1$ points.

But how can it be guaranteed that the scheme picks out the correct entropy-satisfying weak solution? To answer this question, we first give the following definition.

Definition 5 (Monotone Scheme)**.** A difference scheme of the form

$$u^{n+1}_{j_i,k_i} = \mathcal{Q}(u^n_{j_{i-\ell-1},k_{i-\ell-1}}, \ldots, u^n_{j_{i+m},k_{i+m}}) \tag{74}$$

is said to be *monotone* if the function $\mathcal{Q}$ is a monotone increasing function with respect to each of its arguments.

Proposition 4 ([97, 123])**.** *A conservative, monotone scheme produces a solution that satisfies the entropy condition.*

In a nut shell, this means that to construct a viable numerical scheme for solving IVP (36), we only need to make sure that it is in conservation form and that it is a monotone increasing function of its arguments.

Next we move on to the discussion of HJ equations.

3.3 Hamilton-Jacobi Equations

3.3.1 Introduction

Consider the IVP for Hamilton-Jacobi (HJ) equation:

$$u_t + H(u_x, x) = 0 \qquad \text{in } \mathcal{U} = \mathbb{R} \times (0, \infty), \tag{75a}$$

$$u = g \qquad \text{on } \Gamma = \mathbb{R} \times \{t = 0\}, \tag{75b}$$

where the Hamiltonian $H : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and the initial function $g : \mathbb{R} \to \mathbb{R}$ are given. The unknown is the function $u : \mathbb{R} \times [0, \infty) \to \mathbb{R}$. In the next section, we derive the characteristics for the IVP (75).

3.3.2 Characteristics for the Hamilton-Jacobi Equation

As for the conservation laws, we set $\mathbf{x}(s) = [x(s),\ t(s)]$, $\mathbf{p} = [p_1(s),\ p_2(s)]$, where $p_1(s) = u_x(x(s), t(s))$ and $p_2(s) = u_t(x(s), t(s))$. Then we have

$$G(\mathbf{p}(s), z(s), \mathbf{x}(s)) = p_2(s) + H(p_1(s), x(s)). \tag{76}$$

Therefore,

$$D_pG = \left[\frac{\partial H}{\partial p_1}(p_1(s), x(s)),\ 1\right], \tag{77}$$

$$D_xG = \left[\frac{\partial H}{\partial x}(p_1(s), x(s)),\ 0\right], \tag{78}$$

$$D_zG = 0. \tag{79}$$

Thus equation (35c) becomes

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}s}(s) & = \frac{\partial H}{\partial p_1}(p_1(s), x(s)), \\ \frac{\mathrm{d}t}{\mathrm{d}s}(s) & = 1. \end{cases} \tag{80}$$

Hence, $t(s) = s$, since $t(0) = 0$. Again, as for the conservation laws, we can identify the parameter s with time t.

The equation (35a) for the case at hand reads

$$\begin{cases} \frac{\mathrm{d}p_1}{\mathrm{d}s}(s) & = -\frac{\partial H}{\partial x}(p_1(s), x(s)), \\ \frac{\mathrm{d}p_2}{\partial s}(s) & = 0; \end{cases} \tag{81}$$

the equation (35b) is

$$\frac{\mathrm{d}z}{\mathrm{d}s}(s) = \frac{\partial H}{\partial p_1}(p_1(s), x(s))p_1(s) + p_2(s) \tag{82}$$

$$= \frac{\partial H}{\partial p_1}(p_1(s), x(s))p_1(s) - H(p_1(s), x(s)) \quad \text{by (75a), (76).} \tag{83}$$

In summary, the characteristic equations for the HJ equation are

$$\frac{\mathrm{d}p_1}{\mathrm{d}s}(s) = -\frac{\partial H}{\partial x}(p_1(s), x(s)), \tag{84a}$$

$$\frac{\mathrm{d}z}{\mathrm{d}s}(s) = \frac{\partial H}{\partial p_1}(p_1(s), x(s))p_1(s) - H(p_1(s), x(s)), \tag{84b}$$

$$\frac{\mathrm{d}x}{\mathrm{d}s}(s) = \frac{\partial H}{\partial p_1}(p_1(s), x(s)). \tag{84c}$$

Equations (84a) and (84c) are called *Hamilton's equations*. Once $x(s)$, $p_1(s)$ have been found from the Hamilton's equations, $z(s)$ can be found from (84b).

As for conservation laws, the IVP for HJ equation (75) does not, in general, have a smooth solution u lasting for all times $t > 0$. To show this, we consider for simplicity

$$H(u_x, x) = H(u_x), \tag{85}$$

that is,

$$u_t + H(u_x) = 0 \qquad \text{in } \mathcal{U} = \mathbb{R} \times (0, \infty), \tag{86a}$$

$$u = g \qquad \text{on } \Gamma = \mathbb{R} \times \{t = 0\}. \tag{86b}$$

Hence, the associated Hamilton's equations are

$$\dot{x}(t) = \frac{\partial H}{\partial p_1}(p_1(t)), \tag{87a}$$

$$\dot{p_1}(t) = 0, \tag{87b}$$

with initial conditions

$$x(0) = y, \tag{88a}$$

$$p_1(0) = g'(y), \tag{88b}$$

for some fixed $y \in \Gamma$. Hence, the solution of (87)-(88) is

$$x(t) = y + t\frac{\partial H}{\partial p_1}(g'(y)), \tag{89a}$$

$$p_1(t) = g'(y). \tag{89b}$$

In the general form, we can write (89) as

$$x(t, y) = y + t\frac{\partial H}{\partial p_1}(g'(y)), \tag{90a}$$

$$p_1(t, y) = g'(y), \tag{90b}$$

for all $y \in \Gamma$.

Therefore, (84b) reduces to

$$\dot{z}(t) = g'(y)\frac{\partial H}{\partial p_1}(g'(y)) - H(g'(y)), \tag{91}$$

with $z(0) = g(y)$, which implies

$$z(t) = g(y) + t\left(g'(y)\frac{\partial H}{\partial p_1}(g'(y)) - H(g'(y))\right). \tag{92}$$

Now we suppose that the mapping from $y \to x(t, y)$ is one-to-one, then the candidate solution u produced by the method of characteristics is

$$u(x, t) = g(x^{-1}(t, y)) + t\left(g'(x^{-1}(t, y))\frac{\partial H}{\partial p_1}(g'(x^{-1}(t, y))) - H(g'(x^{-1}(t, y)))\right). \tag{93}$$

But since $(x(t),t) = \left(y + t\frac{\partial H}{\partial p_1}(g'(y)), t\right)$ $(t \geq 0)$ are straight lines, the characteristics may possibly intersect at some time $t > 0$. Hence, we see that the mapping from $y \to x(t,y)$ might not be one-to-one, and $x^{-1}(t,y)$ might be defined only for small $t > 0$. As a consequence, the function u is not globally defined by (93) which also implies that the solution to IVP for HJ equations does not in general have a smooth solution, existing for all times $t > 0$. Also it has been pointed out by Crandall et al. [43] that if H and g are assumed to be smooth and g be compactly supported, then (86) will typically have a unique C^2 solution u on some maximal time interval $0 \leq t < T$ for which $lim_{t \nearrow T} u(x,t)$ exists uniformly, but this limiting function might not be continuously differentiable. Thus, u_x might become discontinuous at $t = T$ (or "shocks" might form).

Hence, in the next section we consider the notion of "weak" solution for HJ equations.

3.3.3 Viscosity Solution

Consider the approximate problem:

$$\begin{cases} u_t^\nu + H(Du^\nu, x) - \nu u_{xx}^\nu = 0 & \text{in } \mathbb{R} \times (0,\infty), \\ u^\nu = g & \text{on } \mathbb{R} \times \{t = 0\}, \end{cases} \tag{94}$$

for $\nu > 0$. Equation (75) involves a fully nonlinear first order PDE, whereas (94) is an IVP for a quasilinear parabolic PDE, which is known to have a smooth solution [54]. The term νu_{xx}^ν in (94) in effect regularizes the HJ equation. The idea is that as $\nu \to 0$, the solution u^ν of (94) will converge to some sort of weak solution of (75). This technique is known as the method of *vanishing viscosity*. However, as $\nu \to 0$ we can expect to loose control over the various estimates of the function u^ν and its derivatives: these estimates depend strongly on the regularizing effect of νu_{xx}^ν and blow up as $\nu \to 0$. Hence, a unique limit solution may not exist. However, Evans in [54] has mentioned that in practice we can be at least sure that the family $\{u^\nu\}_{\nu>0}$ is bounded and equicontinuous on compact subsets of $\mathbb{R}^n \times [0,\infty)$. Now since the family $\{u^\nu\}_{\nu>0}$ is bounded and equicontinous on compact subsets of $\mathbb{R}^n \times [0,\infty)$, consequently the Arzela-Ascoli Compactness Criterion (Appendix B) ensures that

$$u^{\nu_j} \to u \quad \text{locally uniformly in } \mathbb{R} \times [0,\infty), \tag{95}$$

for some subsequence $\{u^{\nu_j}\}_{j=1}^{\infty}$ and some limit function

$$u \in C(\mathbb{R} \times [0, \infty)). \tag{96}$$

Now we can expect that u is some kind of solution of our IVP (75) but as we only know u is continuous, and have absolutely no information as to whether u_x and u_t exist in any sense, such an interpretation is difficult.

To show that such a u is a weak solution one way would have been to integrate by parts to throw the "hard-to-control" derivatives onto a fixed test function, and only then try to go to limits to discover a weak solution, as was done for the conservation laws. But since (75a) is fully nonlinear we cannot just integrate by parts to switch to differentiations on the test function. Hence, the idea is to put the derivatives onto any smooth function v, at the expense of certain inequalities holding. The solution that is built using this technique is called *viscosity solution*, in honor of the vanishing viscosity technique.

Definition 6 (Viscosity Solution)**.** A bounded, uniformly continuous function u is called a *viscosity solution* of IVP (75) for HJ equation provided:

i) $u = g$ on $\mathbb{R} \times \{t = 0\}$,

ii) for each $v \in C^{\infty}(\mathbb{R} \times (0, \infty))$

$$\begin{cases} \text{if } u - v \text{ has a local maximum at a point } (x_0, t_0) \in \mathbb{R} \times (0, \infty), \text{ then} \\ v_t(x_0, t_0) + H(v_x(x_0, t_0), x_0) \leq 0, \end{cases} \tag{97}$$

and

$$\begin{cases} \text{if } u - v \text{ has a local minimum at a point } (x_0, t_0) \in \mathbb{R} \times (0, \infty), \text{ then} \\ v_t(x_0, t_0) + H(v_x(x_0, t_0), x_0) \geq 0, \end{cases} \tag{98}$$

To verify that a given function u is a viscosity solution of the HJ equation (75), we must confirm that (97), (98) hold for all smooth functions v. Some of the interesting facts regarding the viscosity solutions taken from [54] are as follows:

1. If u is constructed using the vanishing viscosity method, it is a viscosity solution.

2. Any classical solution of (75) is also a viscosity solution.

3. Let u be a viscosity solution of (75) and suppose u is differentiable at some point $(x_0, t_0) \in \mathbb{R} \times (0, \infty)$. Then

$$u_t(x_0, t_0) + H(u_x(x_0, t_0), x_0) = 0. \tag{99}$$

For a more detailed discussion on the viscosity solutions, the reader is referred to [42, 41, 43, 54] where the existence and uniqueness of the viscosity solutions to IVP for HJ equations (75) is also shown.

3.3.4 Connection with Conservation Laws

The purpose of this section is to show a direct connection between the conservation laws and HJ equations in one spatial dimension, which will be utilized for solving the IVP to HJ equations numerically.

For simplicity, we consider the HJ equation

$$u_t + H(u_x) = 0, \tag{100}$$

which becomes

$$(u_x)_t + H(u_x)_x = 0, \tag{101}$$

after one takes a spatial derivative of the entire equation. Setting $v = u_x$ in the above equation results in

$$v_t + H(v)_x = 0, \tag{102}$$

which is a scalar conservation law. Thus in one spatial dimension a direct correspondence between HJ equations and conservation laws can be drawn. The solution v to a conservation law is the derivative of a solution u to a HJ equation. Conversely, the solution u to a HJ equation is the integral of a solution v to a conservation law. This allows us to point out a number of useful facts.

1. Since the integral of a discontinuity is a “kink”, or discontinuity in the first derivative, solutions to HJ equations can develop kinks in the solution even if the data are initially smooth.

2. The solutions to HJ equations cannot generally develop a discontinuity unless the corresponding conservation law develops a delta function. Thus solutions u to (75) are typically continuous.

3. Since the conservation laws can have non-unique solutions, entropy conditions are needed to pick out "physically" relevant solutions to equation (75) as well.

Hence, the successful numerical methodology for solving conservation laws (Section 3.2) can be applied for solving the IVP for HJ equations (75) [111].

3.4 Summary

In this chapter, we showed that the solution to the nonlinear conservation laws and the HJ equations do not have smooth solutions lasting for all times $t > 0$ and hence developed the notion of "weak" solutions for the nonlinear conservation laws and the HJ equations. We also stated the conditions that any numerical scheme for solving IVP for the conservation laws should satisfy for picking out the unique "physically" correct solution to the IVP for the conservation laws. A direct correspondence between the conservation laws and HJ equations in one spatial dimension was shown which allows us to use the successful numerical methodology of conservation laws for solving IVP for HJ equations.

As was shown for the conservation laws and HJ equations, the solution to (IVP), in general, do not have smooth solutions. Hence, in order to solve evolution equations in a computationally efficient manner, the grid should adapt dynamically to reflect local changes in the solution. Therefore, in the next chapter, we propose a novel multiresolution-based data compression algorithm.

CHAPTER IV

MULTIRESOLUTION DATA COMPRESSION

First, we give a brief overview on dyadic grids which are used in the proposed multiresolution data compression algorithm.

4.1 Dyadic grids

Since $\overline{\mathcal{D}} = [0,1]$, we consider dyadic grids of the form

$$\mathcal{V}_j = \{x_{j,k} \in [0,1] : x_{j,k} = k/2^j,\ 0 \le k \le 2^j\}, \quad J_{\min} \le j \le J_{\max}, \tag{103}$$

where j denotes the resolution level, k the spatial location, and $J_{\min},\ J_{\max} \in \mathbb{Z}_0^+$. We denote by $\mathcal{W}_j$ the set of grid points belonging to $\mathcal{V}_{j+1} \setminus \mathcal{V}_j$. Therefore,

$$\mathcal{W}_j = \{y_{j,k} \in [0,1] : y_{j,k} = (2k+1)/2^{j+1},\ 0 \le k \le 2^j - 1\}, \quad J_{\min} \le j \le J_{\max} - 1. \tag{104}$$

Hence, $x_{j+1,k} \in \mathcal{V}_{j+1}$ is given by

$$x_{j+1,k} = \begin{cases} x_{j,k/2}, & \text{if } k \text{ is even,} \\ y_{j,(k-1)/2}, & \text{otherwise.} \end{cases} \tag{105}$$

An example of a dyadic grid with $J_{\min} = 0$ and $J_{\max} = 5$ is shown in Figure 13.

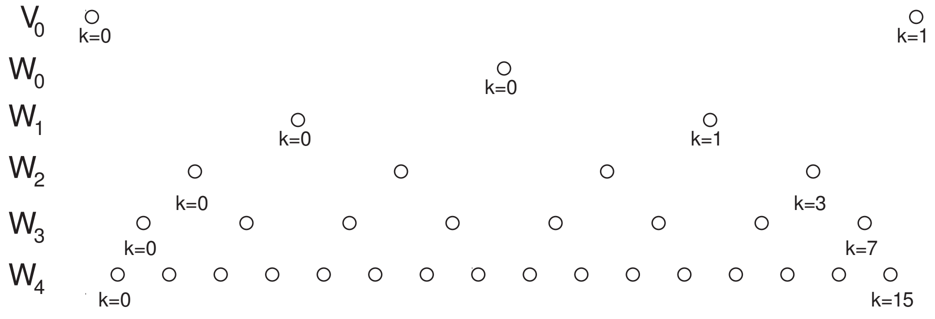


Figure 13: Example of a dyadic grid.

With a slight abuse of notation, we write $\mathcal{V}_{j+1} = \mathcal{V}_j \oplus \mathcal{W}_j$, although $\mathcal{W}_j$ is not an orthogonal complement of $\mathcal{V}_j$ in $\mathcal{V}_{j+1}$. The subspaces $\mathcal{V}_j$ are nested, $\mathcal{V}_{J_{\min}} \subset \mathcal{V}_{J_{\min}+1} \cdots \subset$

$\mathcal{V}_{J_{\max}}$, with $\lim_{J_{\max}\to\infty} \mathcal{V}_{J_{\max}} = \overline{\mathcal{D}}$. The sequence of subspaces $\mathcal{W}_j$ satisfy $\mathcal{W}_j \cap \mathcal{W}_\ell = \varnothing$ for $j \neq \ell$.

Next, we present a novel multiresolution scheme for data compression.

4.2 Encoding

Suppose $g : \overline{\mathcal{D}} \to \mathbb{R}$ is specified on a grid $\mathcal{V}_{J_{\max}}$,

$$\mathcal{U}_{J_{\max}} = \{g_{j,k} : x_{j,k} \in \mathcal{V}_{J_{\max}}\}, \tag{106}$$

where $g_{j,k} = g(x_{j,k})$. Let $\mathcal{I}^p(x; \mathcal{X}_{\mathsf{Grid}})$ denote *any* p-th order interpolation of $\mathcal{U} = \{g_{j,k} : x_{j,k} \in \mathcal{X}_{\mathsf{Grid}}\}$, where $\mathcal{X}_{\mathsf{Grid}} = \{x_{j_\ell,k_\ell}\}_{\ell=i}^{i+p} \subset \mathsf{Grid}$, where

$$\begin{aligned}\mathsf{Grid} =& \{x_{j_i,k_i} : x_{j_i,k_i} \in [0,1],\ 0 \le k_i \le 2^{j_i},\ J_{\min} \le j_i \le J_{\max},\ \text{for } i = 0 \ldots N, \\ & \text{and } x_{j_i,k_i} < x_{j_{i+1},k_{i+1}},\ \text{for } i = 0 \ldots N-1\} \subset \mathcal{V}_{J_{\max}},\end{aligned} \tag{107}$$

and $x \in [x_{j_i,k_i}, x_{j_{i+p},k_{i+p}}]$. In (107) Grid can be uniform or nonuniform. Then the encoding algorithm for compressing the signal g is as follows.

Encoding Algorithm

Step 1. Initialize an intermediate grid $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}}$, with function values $\mathcal{U}_{\text{int}} = \mathcal{U}_{\min}$, where $\mathcal{U}_{\min} = \{g_{J_{\min},k} : 0 \le k \le 2^{J_{\min}}\}$. Set $j = J_{\min}$.

Step 2. DO for $k = 0, \ldots, 2^j - 1$.

(a) Compute the interpolated function value $\hat{g}(y_{j,k}) = \mathcal{I}^p(y_{j,k}, \mathcal{X}_{\mathsf{Grid}_{\text{int}}})$.

(b) If the interpolative error coefficient at the point $y_{j,k}$,

$$d_{j,k} = |g(y_{j,k}) - \hat{g}(y_{j,k})| > \epsilon, \tag{108}$$

where ϵ is the prescribed threshold, then add $y_{j,k}$ to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and the corresponding function value $g(y_{j,k})$ to $\mathcal{U}_{\text{int}}$.

Step 3. Increment j by 1. If $j < J_{\max}$ goto Step 2, otherwise move on to the next step.

Step 4. Terminate the algorithm. The final nonuniform grid representing the compressed information is $\mathsf{Grid}_{\mathrm{M}} = \mathsf{Grid}_{\mathrm{int}}$ and the corresponding function values is the set $\mathcal{U}_{\mathrm{M}} = \mathcal{U}_{\mathrm{int}}$.

If we represent the above nonlinear encoding procedure by an operator M, then we can write

$$\mathcal{U}_{\mathrm{M}} = M\mathcal{U}_{\mathrm{J_{max}}}. \tag{109}$$

One should note that in Harten's approach [68, 69, 70] the points of a particular resolution level $\mathcal{W}_j$ are interpolated only from the corresponding points belonging to $\mathcal{V}_j$. In our approach, instead, we continuously keep on updating the grid, and the points $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ of level $\mathcal{W}_j$ are interpolated from the function values at the points in $\mathcal{V}_j \oplus \mathcal{W}_j$. Hence, by making use of the extra information from levels $\mathcal{W}_j$ – which in any case will be added to the adaptive grid – we are able to reduce the number of grid points in the final grid. This process results in higher compression factors, as will be shown in Section 4.6 via several examples.

In the next section, we briefly describe the techniques that will be used in this thesis for constructing $\mathcal{I}^p$.

4.3 Construction of the Interpolation Operator $\mathcal{I}^p$

The proposed encoding and decoding algorithms will work with many interpolations, like piecewise-polynomial interpolation, essentially nonoscillatory (ENO) interpolation, cubic-spline interpolation, trigonometric interpolation etc. [70]. For sake of brevity, we only describe the techniques for constructing $\mathcal{I}^p$ that are used in the thesis.

4.3.1 Piecewise-polynomial Interpolation

For piecewise-polynomial interpolation, the stencil $\mathcal{X}_{\mathsf{Grid}_{\mathrm{int}}}$ consists of the $p+1$ nearest points to x in $\mathsf{Grid}_{\mathrm{int}}$. By $p+1$ nearest points here we mean one neighboring point on the left of x, one neighboring point on the right of x and the remaining $p-1$ points are the points nearest to x in the set $\mathsf{Grid}_{\mathrm{int}}$. In case two points are at the same distance from x, that is, if a point on the left and a point on the right are equidistant to x, then we choose a point so as to equalize the number of points on both sides. For example, consider a grid at level $j = 3$ as shown in Figure 14. The set $\mathsf{Grid}_{\text{int}}$ consists of the solid circles. Let $p = 3$ and $x = x_{3,4}$, shown by an empty square in Figure 14. The whole process involves three steps. Step a: We include in the set $\mathcal{X}_{\mathsf{Grid}_{\text{int}}}$ one neighboring point on the left ($x_{3,2}$), shown by a left triangle and one neighboring point on the right ($x_{3,5}$), shown by a right triangle in Figure 14. Step b: We add to the set $\mathcal{X}_{\mathsf{Grid}_{\text{int}}}$ the point $x_{3,6}$ since the distance from x_{interp} to $x_{3,0}$ is greater than the distance of $x_{3,4}$ to $x_{3,6}$. Step c: To choose the last point, we notice that both points $x_{3,0}$ and $x_{3,8}$ are equidistant to $x_{3,4}$. In this case, we choose $x_{3,0}$ in order to equalize the number of points on both sides. Hence, our final set $\mathcal{X}_{\mathsf{Grid}_{\text{int}}}$ consists of points $x_{3,0},\ x_{3,2},\ x_{3,5}$, and $x_{3,6}$ as shown in Figure 14.

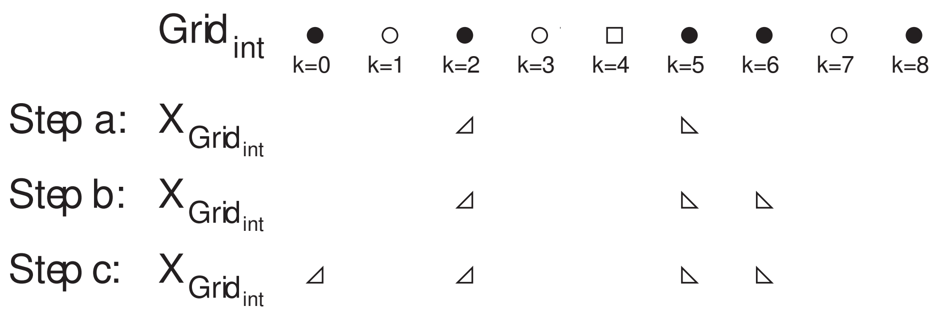


Figure 14: Demonstration of the procedure for finding the nearest points in piecewise polynomial interpolation.

Suppose p is even and suppose we have already chosen $p-2$ points based on the previous methodology, such that $(p-2)/2$ points are on the left of x and the remaining $(p-2)/2$ points are on the right of x. In case both points on the left and the right are equidistant to x we choose either of these points as the last point. Note that this situation will not arise if p is odd.

Once we have found the $p+1$ nearest points of the set $\mathcal{X}_{\mathsf{Grid}_{\text{int}}}$, we construct an interpolating polynomial $\hat{g}(x)$ of order p passing through these $p+1$ points. One may use Neville's algorithm (Appendix A.1.6) to construct the respective interpolating polynomials on the fly.

4.3.2 Essentially Non-Oscillatory Interpolation

For ENO interpolation, the stencil $\mathcal{X}_{\mathsf{Grid}_{\text{int}}}$ consists of one neighboring point on the left and one neighboring point on the right of x in the set $\mathsf{Grid}_{\text{int}}$, and the remaining $p-1$ points are selected from the set $\mathsf{Grid}_{\text{int}}$ that give the least oscillatory polynomial. Next, we briefly describe how the ENO interpolant is constructed. For more details on ENO interpolation the reader is referred to [70, 110].

To this end, let the grid $\mathsf{Grid}_{\text{int}}$ be given as in (107). Now define

$$D^+ g^n_{j_i,k_i} = \frac{g^n_{j_{i+1},k_{i+1}} - g^n_{j_i,k_i}}{x_{j_{i+1},k_{i+1}} - x_{j_i,k_i}}, \qquad D^- g^n_{j_i,k_i} = \frac{g^n_{j_i,k_i} - g^n_{j_{i-1},k_{i-1}}}{x_{j_i,k_i} - x_{j_{i-1},k_{i-1}}}. \tag{110}$$

Define the zeroth divided difference of g by

$$D^0_i g = g^n_{j_i,k_i}, \tag{111}$$

at each grid node x_{j_i,k_i}. The first divided difference of g are defined midway between grid nodes as

$$D^1_{i+1/2} g = \frac{D^0_{i+1} g - D^0_i g}{x_{j_{i+1},k_{i+1}} - x_{j_i,k_i}}. \tag{112}$$

The second divided differences are defined at the grid nodes as

$$D^2_i g = \frac{D^1_{i+1/2} g - D^1_{i-1/2} g}{x_{j_{i+1},k_{i+1}} - x_{j_{i-1},k_{i-1}}}, \tag{113}$$

while the third divided differences

$$D^3_{i+1/2} g = \frac{D^2_{i+1} g - D^2_i g}{x_{j_{i+2},k_{i+2}} - x_{j_{i-1},k_{i-1}}}, \tag{114}$$

are defined midway between the grid nodes.

The divided differences are then used to construct a polynomial of the form

$$\hat{g}(x) = Q_0 + Q_1(x) + Q_2(x) + Q_3(x). \tag{115}$$

Let the left neighboring point to x in $\mathsf{Grid}_{\text{int}}$ be x_{j_L,k_L}. Then define

$$Q_0 = g(x_{j_L,k_L}), \tag{116}$$

and

$$Q_1(x) = (D^1_{L+1/2} g)(x - x_{j_L,k_L}), \tag{117}$$

which gives linear interpolation of one neighboring point on the left and one neighboring point on the right of x in the set $\mathsf{Grid}_{\text{int}}$.

Now for the second-order accuracy we can include the next point to the left and use $D_L^2 g$, or we can include the next point to the right and use $D_{L+1}^2 g$. One would like to avoid interpolating near large variations such as discontinuities or steep gradients, since they cause overshoots in the interpolating function, leading to numerical errors in the approximation of the derivative. Thus, if $|D_L^2 g| \le |D_{L+1}^2 g|$, set $c = D_L^2 g$ and $L^\star = L - 1$; otherwise, set $c = D_{L+1}^2 g$ and $L^\star = L$. Then define

$$Q_2(x) = c(x - x_{j_L, i_L})(x - x_{j_{L+1}, k_{L+1}}). \tag{118}$$

If we stop here, that is, omitting the Q_3 term, we have a second-order accurate method for approximating $g(x)$.

To obtain the third-order accurate correction compare $|D^3_{L^\star+1/2} g|$ and $|D^3_{L^\star+3/2} g|$. If $|D^3_{L^\star+1/2} g| < |D^3_{L^\star+3/2} g|$, set $c^\star = |D^3_{L^\star+1/2} g|$ otherwise set $c^\star = |D^3_{L^\star+3/2} g|$. Then define

$$Q_3(x) = c^\star(x - x_{j_{L^\star}, k_{L^\star}})(x - x_{j_{L^\star+1}, k_{L^\star+1}})(x - x_{j_{L^\star+2}, k_{L^\star+2}}), \tag{119}$$

which is the third-order accurate correction to the approximation of $g(x)$.

Next, we give the decoding algorithm, that is, the algorithm for computing

$$\hat{\mathcal{U}}_{J_{\max}} = M^{-1} \mathcal{U}_{\mathrm{M}}. \tag{120}$$

4.4 Decoding

One way of decoding the information back from the compressed signal in nonlinear schemes is to keep track of the stencils that were used for interpolating the function values at a particular point while encoding the information and use the same stencils to decode the information from the compressed signal. An alternative way (described below) of decoding the information from the compressed signal is to follow the same approach as in the encoding algorithm.

Decoding Algorithm

Step 1. Initialize $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}}$, with function values $\hat{\mathcal{U}}_{J_{\max}} = \mathcal{U}_{\text{int}} = \mathcal{U}_{\min}$, where $\mathcal{U}_{\min} = \{g_{J_{\min}, k} : 0 \le k \le 2^{J_{\min}}\}$. Set $j = J_{\min}$.

Step 2. DO for $k = 0, \dots, 2^j - 1$.

If $g(y_{j,k}) \in \mathcal{U}_{\mathrm{M}}$, then add $g(y_{j,k})$ to $\hat{\mathcal{U}}_{J_{\max}}$, $\mathcal{U}_{\mathrm{int}}$ and $y_{j,k}$ to $\mathsf{Grid}_{\mathrm{int}}$ otherwise add $\hat{g}(y_{j,k}) = \mathcal{I}^p(y_{j,k}, \mathcal{X}_{\mathsf{Grid}_{\mathrm{int}}})$ to $\hat{\mathcal{U}}_{J_{\max}}$.

Step 3. Increment j by 1. If $j < J_{\max}$ goto Step 2, otherwise move on to the next step.

Step 4. Terminate the algorithm.

It should be noted that at termination $\mathsf{Grid}_{\mathrm{int}} = \mathcal{V}_{J_{\max}}$.

4.5 Error Estimate

In this section, we derive an estimate for the error between the original signal $\mathcal{U}_{J_{\max}}$ and the decoded signal $\hat{\mathcal{U}}_{J_{\max}}$ obtained after encoding the original signal and then decoding the compressed signal $\mathcal{U}_M$.

Proposition 5. *Let $\mathcal{U}_{J_{\max}}$ be defined as in (106), and let $\hat{\mathcal{U}}_{J_{\max}} = M^{-1}\mathcal{U}_M$, where M^{-1} denotes the decoding algorithm described above. Then for $1 \le m < \infty$,*

$$E_m(g) = \|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_m = \left(\frac{1}{2^{J_{\max}} + 1} \sum_{k=0}^{2^{J_{\max}}} |g_{J_{\max},k} - \hat{g}_{J_{\max},k}|^m \right)^{\frac{1}{m}} \le \left(\frac{2^{J_{\max}} - 2^{J_{\min}}}{2^{J_{\max}} + 1} \right)^{\frac{1}{m}} \epsilon, \tag{121}$$

and

$$E_\infty(g) = \|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_\infty = \max_{0 \le k \le 2^{J_{\max}}} |g_{J_{\max},k} - \hat{g}_{J_{\max},k}| \le \epsilon. \tag{122}$$

Proof. First, we note that

$$|g_{J_{\min},k} - \hat{g}_{J_{\min},k}| = 0, \quad k = 0, \dots, 2^{J_{\min}}, \tag{123}$$

since $g_{J_{\min},k} \in \mathcal{U}_{\mathrm{M}}$ for all $k = 0, \dots, 2^{J_{\min}}$. Next, since the function values in the set $\hat{\mathcal{U}}_{J_{\max}}$ are interpolated directly only from the function values in $\mathcal{U}_{\mathrm{M}}$, we have a direct control over the error. Therefore,

$$|g(y_{j,k}) - \hat{g}(y_{j,k})| \le \epsilon, \quad k = 0, \dots, 2^j - 1, \tag{124}$$

for $j = J_{\min}, \dots, J_{\max} - 1$. Hence, we have

$$\|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_\infty = \max_{0 \le k \le 2^{J_{\max}}} |g_{j,k} - \hat{g}_{j,k}| \le \epsilon. \tag{125}$$

For $1 \leq m < \infty$, we have

$$\begin{aligned} & \|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_m^m & (126) \\ = \ & \frac{1}{2^{J_{\max}}+1}\left(\sum_{k=0}^{2^{J_{\min}}} |g_{J_{\min},k} - \hat{g}_{J_{\min},k}|^m + \sum_{j=J_{\min}}^{J_{\max}-1}\sum_{k=0}^{2^j-1} |g(y_{j,k}) - \hat{g}(y_{j,k})|^m\right) & (127) \\ \leq \ & \frac{\epsilon^m}{2^{J_{\max}}+1}\sum_{j=J_{\min}}^{J_{\max}-1}\sum_{k=0}^{2^j-1} 1 = \epsilon^m \frac{2^{J_{\max}} - 2^{J_{\min}}}{2^{J_{\max}}+1}. & (128) \end{aligned}$$

Consequently, for $1 \leq m < \infty$,

$$\|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_m \leq \left(\frac{2^{J_{\max}} - 2^{J_{\min}}}{2^{J_{\max}}+1}\right)^{\frac{1}{m}} \epsilon, \tag{129}$$

which completes the proof. □

Example 3

Consider $g_1 : \overline{\mathcal{D}} \to \mathbb{R}$,

$$g_1(x) = \begin{cases} 1, & \frac{1}{3} \leq x \leq \frac{2}{3}, \\ 0, & \text{otherwise}, \end{cases} \tag{130}$$

and $g_2 : \overline{\mathcal{D}} \to \mathbb{R}$,

$$g_2(x) = \begin{cases} 0, & 0 \leq x < \frac{1}{6}, \\ 1, & \frac{1}{6} \leq x < \frac{1}{3}, \\ 0, & \frac{1}{3} \leq x < \frac{1}{2}, \\ \sin(\pi x), & \frac{1}{2} \leq x < \frac{2}{3}, \\ 0, & \frac{2}{3} \leq x < \frac{5}{6}, \\ x, & \frac{5}{6} \leq x \leq 1. \end{cases} \tag{131}$$

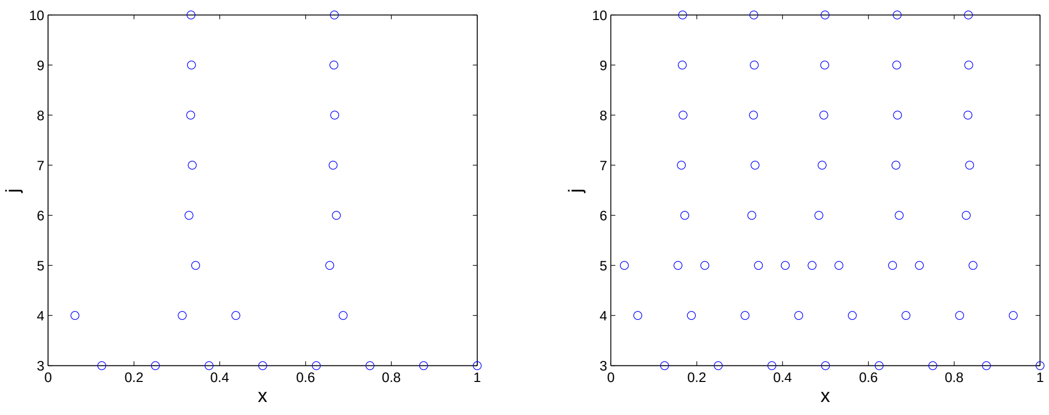
For this example, we consider a grid with $J_{\min} = 3$ and $J_{\max} = 10$ and use ENO interpolation for the encoding and the decoding algorithms described above. For both g_1 and g_2, the data compression factor

$$C = \frac{2^{J_{\max}} + 1 - N_{\mathrm{p}}}{2^{J_{\max}}+1} \times 100\%, \tag{132}$$

where N_{p} denotes the number of grid points, along with the decoding errors E_m, $m = 1, 2, \infty$, with an interpolating polynomial of degree $p = 3$ and different thresholds ϵ, are summarized in Table 1. First, we consider $\epsilon = 10^{-3}$ for both the functions g_1 and g_2. The proposed algorithm compressed the given signals g_1 and g_2 using only 25 and 52 points, respectively, The decoding errors E_m $(m = 1, 2, \infty)$ are well below the threshold for both these functions. The grid point distributions for both g_1 and g_2 are shown in Figure 15. Next, for g_2, we decrease the threshold to 10^{-7}. It is observed that the proposed encoding algorithm increased the number of points used for compressing the signal; once again, the decoding errors are below the prescribed threshold.

Table 1: Example 3. Data compression along with the decoding errors for the proposed approach.

	ϵ	C	E_{inf}	E_1	E_2
g_1	10^{-3}	97.56	0	0	0
g_2	10^{-3}	94.93	2.2741×10^{-5}	8.2103×10^{-7}	2.8111×10^{-6}
g_2	10^{-7}	93.85	9.0426×10^{-8}	3.7983×10^{-9}	1.2662×10^{-8}



(a) $g_1(x)$. (b) $g_2(x)$.

Figure 15: Example 3. Grid point distribution for $\epsilon = 1 \times 10^{-3}$.

4.6 Comparison with Existing Multiresolution-Based Approach

The proposed encoding algorithm results, in general, in a fewer number of grid points when compared to the Harten's multiresolution scheme [68, 69]. First, we explain why this is so and then we give several examples to demonstrate this fact.

In the encoding algorithm of existing approach [68, 69], one interpolates $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ only from the function values at the points belonging to $\mathcal{V}_j$ for $j = J_{\min} \dots J_{\max} - 1$, and only then, one adds to the adaptive grid, the points $y_{j,k}$ for all the pairs (j, k), such that $d_{j,k} > \epsilon$.

In the proposed method, we continuously keep on updating the adaptive grid instead. If the interpolative error coefficient at $y_{j,k}$, where $0 \leq k \leq 2^j - 1$ and $J_{\min} \leq j \leq J_{\max} - 1$, is greater than the prescribed threshold, we add $y_{j,k}$ to the adaptive grid. We use the newly added point also for interpolating the remaining points at level $\mathcal{W}_j$ and the levels below it. In other words, in the proposed approach $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ are interpolated from the function values at the points in $\mathcal{V}_j \oplus \mathcal{W}_j$ for $j = J_{\min}, \ldots, J_{\max} - 1$. Hence, by making use of the extra information from levels $\mathcal{W}_j$, which in any case will be added to the adaptive grid, we are able to reduce the number of grid points in the final grid.

We illustrate this fact with the help of several examples.

Example 4

We again consider the functions g_1 and g_2 given by (130) and (131), respectively, and a grid with $J_{\min} = 2$ and $J_{\max} = 10$. We compare the proposed encoding algorithm (using ENO interpolation) with the Harten's encoding algorithm (using ENO interpolation) and the Harten's encoding algorithm (using central interpolation). The number of grid points used by the proposed algorithm N_{p}, Harten's algorithm (using ENO interpolation) N_{Heno}, and Harten's algorithm (using central interpolation) N_{Hc} for different thresholds using interpolating polynomial of degree $p = 3$ are summarized in Table 2. For both

Table 2: Example 4. Comparison of the proposed decoding approach with Harten's approach.

	ϵ	N_{p}	N_{Heno}	N_{Hc}	$N_{\mathrm{p}}/N_{\mathrm{Heno}}$	$N_{\mathrm{p}}/N_{\mathrm{Hc}}$
g_1	10^{-3}	25	29	53	0.86	0.47
g_2	10^{-3}	52	58	108	0.90	0.48
g_2	10^{-7}	63	73	119	0.86	0.53

functions g_1 and g_2, we found that the proposed algorithm results in up to 14% fewer number of points in the compressed data compared to the Harten's approach (using ENO interpolation) and up to 53% fewer points compared to the Harten's approach (using central interpolation).

4.7 Summary

In this chapter, we have proposed a novel multiresolution scheme for data compression. The proposed data compression scheme is shown to outperform similar data compression schemes in the literature. Specifically, we have shown that the proposed approach results in fewer grid points when compared to a common adaptive grid approach.

Next, based on the proposed data compression scheme, we present an adaptive multiresolution technique for solving evolution PDEs .

CHAPTER V

SOLUTION OF IBVP FOR EVOLUTION EQUATIONS

5.1 Problem Statement

The IVP (29) that we are trying to solve is defined over all of $\mathbb{R}$, and hence has no physical boundary. Unfortunately, we can numerically approximate the solution only on a finite domain, so we must introduce boundaries and enforce some form of boundary conditions. Hence, we consider an initial-boundary value problem (IBVP) for an evolution equation:

$$\text{(IBVP)}: \quad \begin{cases} u_t + f(u_{xx}, u_x, u, x) = 0 & \text{in } \mathcal{D} \times (0, \infty), \\ u = g & \text{on } \overline{\mathcal{D}} \times \{t = 0\}, \end{cases} \tag{133}$$

where $\overline{\mathcal{D}} = \mathcal{D} \cup \partial\mathcal{D}$, with $\mathcal{D} \subset \mathbb{R}$ bounded. The function $f : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m \times \mathcal{D} \to \mathbb{R}^m$, and the initial function $g : \overline{\mathcal{D}} \to \mathbb{R}^m$ are given. The unknown is the function $u : \overline{\mathcal{D}} \times [0, \infty) \to \mathbb{R}^m$. The algorithm proposed in this section works for other boundary conditions as well, but for simplicity in the analysis below we only use *periodic*, *Dirichlet*, and *Neumann* boundary conditions. Without loss of generality, we will further assume that $\mathcal{D} = (0, 1)$.

In (IBVP) the initial function g can be irregular. Even if g is smooth, discontinuities such as shocks (in hyperbolic conservation laws) and kinks (in Hamilton-Jacobi equations) can develop in the solution u at some later time. Therefore, we would like to adapt the grid dynamically to any existing or emerging irregularities in the solution instead of using a fine mesh over the whole spatial and temporal domain. In the next section, we propose a novel grid refinement technique for solving (IBVP) in a computationally efficient manner.

5.2 Adaptive gridding

5.2.1 Grid Adaptation for the solution of (IBVP)

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in Eqs. (103) and (104),

$$\mathcal{V}_j = \{x_{j,k} \in [0, 1] : x_{j,k} = k/2^j,\ 0 \le k \le 2^j\}, \quad J_{\min} \le j \le J_{\max}, \tag{134}$$

$$\mathcal{W}_j = \{y_{j,k} \in [0,1] : y_{j,k} = (2k+1)/2^{j+1},\ 0 \le k \le 2^j - 1\}, \quad J_{\min} \le j \le J_{\max} - 1. \tag{135}$$

Assume we are given a nonuniform grid of the form

$$\begin{aligned}\mathsf{Grid} =& \{x_{j_i,k_i} : x_{j_i,k_i} \in [0,1],\ 0 \le k_i \le 2^{j_i},\ J_{\min} \le j_i \le J_{\max},\ \text{for } i = 0 \ldots N,\\ &\text{and } x_{j_i,k_i} < x_{j_{i+1},k_{i+1}},\ \text{for } i = 0 \ldots N-1\} \subset \mathcal{V}_{J_{\max}},\end{aligned} \tag{136}$$

For simplicity, we denote by $u^n_{j,k}$ the value of $u(x,t)$ evaluated at $x = x_{j,k}$ and $t = t_n$, where $0 \le k \le 2^j$, $J_{\min} \le j \le J_{\max}$, $n \in \mathbb{Z}_0^+$, $t_0 = 0$, $t_n = t_{n-1} + \Delta t_n$ for $n > 0$, and Δt_n is the time step based on the Courant-Friedrichs-Levy condition [137] for hyperbolic equations and the von Neumann condition [137] for all other evolution equations. The "top-down" approach of our algorithm allows one to add and remove points using the most recently updated information. To this end, suppose $u(x,t_n)$ is specified on the grid $\mathsf{Grid}_{\text{old}}$, with corresponding solution values $\mathcal{U}_{\text{old}} = \{u^n_{j,k} : x_{j,k} \in \mathsf{Grid}_{\text{old}}\}$, where $\mathsf{Grid}_{\text{old}}$ can be either regular or irregular[1]. We assume $\mathsf{Grid}_{\text{old}} \supseteq \mathcal{V}_{J_{\min}}$. Our aim is to find a new grid $\mathsf{Grid}_{\text{new}}$, by adding or removing points from $\mathsf{Grid}_{\text{old}}$, reflecting local changes in the solution. To this end, we initialize an intermediate grid $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}}$, with the function values $\mathcal{U}_{\text{int}} = \{u^n_{J_{\min},k} : u^n_{J_{\min},k} \in \mathcal{U}_{\text{old}},\ 0 \le k \le 2^{J_{\min}}\}$, and we set $j = J_{\min}$. The mesh refinement algorithm proceeds as follows:

Step 1. Find the points belonging to the intersection of $\mathcal{W}_j$ and $\mathsf{Grid}_{\text{old}}$, that is,

$$Y = \{y_{j,k_i} : y_{j,k_i} \in \mathcal{W}_j \cap \mathsf{Grid}_{\text{old}},\ \text{for } i = 1, \ldots, M,\ 1 \le M \le 2^j - 1\}. \tag{137}$$

If Y is empty goto Step 4 otherwise goto the next step.

Step 2. Set $i = 1$.

(a) Compute the interpolated function values at point $y_{j,k_i} \in Y$, $\hat{u}(y_{j,k_i})$, that is, $\hat{u}_\ell(y_{j,k_i}) = \mathcal{I}^p(y_{j,k_i}, \mathcal{X}_{\mathsf{Grid}_{\text{int}}})$, where $\hat{u}_\ell$ is the ℓth element of $\hat{u}$, for $\ell = 1, \ldots, m$.

(b) If at the point y_{j,k_i}[2],

$$d_{j,k_i}(u^n) = \max_{\ell=1,\ldots,m} |u_\ell(y_{j,k_i}, t_n) - \hat{u}_\ell(y_{j,k_i})| < \epsilon, \tag{138}$$

[1]Typically, $\mathsf{Grid}_{\text{old}}$ at time $t = 0$ is regular with $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\max}}$.

[2]Note that $u(y_{j,k}, t_n) \in \mathcal{U}_{\text{old}}$ for all $y_{j,k} \in Y$.

goto Step 2(f), otherwise add y_{j,k_i} to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and move on to the next step.

(c) Add to $\mathsf{Grid}_{\text{int}}$ N_1 points on the left and N_1 points on the right neighboring to the point y_{j,k_i} in $\mathcal{W}_j$. This step accounts for the possible displacement of any sharp features of the solution during the next time integration step. The value of N_1 dictates the frequency of mesh adaptation and is provided by the user. The larger the N_1, the smaller the frequency of mesh adaptation will be, at the expense of a larger number of grid points in the adaptive grid. Hence, there is a trade-off between the frequency of mesh adaptation and the number of grid points.

(d) Add to $\mathsf{Grid}_{\text{int}}$ $2N_2$ neighboring points at the next finer level $\{y_{j+1,2k_i+\ell}\}_{\ell=-N_2+1}^{N_2}$, where $1 \leq N_2 \leq 2N_1$. This step accounts for the possibility that the solution becomes steeper in this region. Our experience has shown that $N_2 = N_1$ is a good choice.

(e) Add the function values at all the newly added points to $\mathcal{U}_{\text{int}}$. If the function value at any of the newly added points is not known, we interpolate the function value at that point from the points in $\mathsf{Grid}_{\text{old}}$ and their function values in $\mathcal{U}_{\text{old}}$ using $\mathcal{I}^p(\cdot, \mathcal{X}_{\mathsf{Grid}_{\text{old}}})$.

(f) Increment i by 1. If $i \leq M$ goto Step 2(a), otherwise move on to the next step.

Step 3. Increment j by 1. If $j < J_{\max}$ goto Step 1, otherwise move on to the next step.

Step 4. Terminate the algorithm. The final nonuniform grid is $\mathsf{Grid}_{\text{new}} = \mathsf{Grid}_{\text{int}}$ and their corresponding function values is the set $\mathcal{U}_{\text{new}} = \mathcal{U}_{\text{int}}$.

Remark 3. Although the proposed grid adaptation algorithm will work for any interpolation technique, in this work we use ENO interpolation to avoid any unphysical interpolation of the data.

Remark 4. For sake of brevity, in the thesis, we work only with the point-value discretization of data but the proposed encoding and decoding algorithms (or the grid adaptation algorithm for solving PDEs) will also work for discretizations based on the cell-averages.

Next, we explain the proposed grid adaptation algorithm with the help of a simple example.

Example 5

Consider a dyadic grid $\mathcal{V}_4$ and the function

$$g(x) = \begin{cases} 1, & x = x_{4,k}, \\ 0, & \text{otherwise}, \end{cases} \tag{139}$$

with $k = 6$, so that g denotes an impulse located at $x = x_{4,6} = 0.375$. Let $J_{\min} = 0$, $J_{\max} = 4$, $p = 1$, $\epsilon = 0.1$, $N_1 = N_2 = 1$, and consider $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\max}}$. For this example the proposed grid adaptation algorithm is illustrated in Figure 16.

In Figure 16, the solid circles show the points belonging to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and those belonging to $\mathsf{Grid}_{\text{new}}$. The empty squares show the points that are being tested or have been tested for inclusion in $\mathsf{Grid}_{\text{int}}$. If the interpolative error coefficient at a point is greater than the prescribed threshold, then we show that point by a solid square. The left and the right neighbors are shown by left and right triangles, respectively. For reference, all points at that particular level are shown by empty circles.

First, we initialize $\mathsf{Grid}_{\text{int}}$ with $\mathcal{V}_{J_{\min}}$. Next, we check if the function value at the point $y_{0,0} \in \mathcal{W}_0$ can be interpolated from the nearest $p + 1 = 2$ points in $\mathsf{Grid}_{\text{int}}$, which in this case are the points $x_{0,0}$ and $x_{0,1}$. Since for this example $g(y_{0,0})$ can be interpolated from the points in $\mathsf{Grid}_{\text{int}}$, we do not include $y_{0,0}$ in $\mathsf{Grid}_{\text{int}}$. Next, we consider the level $\mathcal{W}_1$ and check the point $y_{1,0}$. Since $g(y_{1,0})$ can again be interpolated from the function values at points $x_{0,0}, x_{0,1} \in \mathsf{Grid}_{\text{int}}$, we do not include $y_{1,0}$ in $\mathsf{Grid}_{\text{int}}$, and move on to the next point $y_{1,1}$. For the same reason as before, we do not include this point and the point $y_{2,0}$ belonging to the next level $\mathcal{W}_2$. Moving further to $y_{2,1}$, we find that $g(y_{2,1})$ cannot be interpolated from the neighboring two points $x_{0,0}, x_{0,1} \in \mathsf{Grid}_{\text{int}}$. Hence, we include $y_{2,1}$ in $\mathsf{Grid}_{\text{int}}$ along with points $y_{2,0}, y_{2,2} \in \mathcal{W}_2$ and $y_{3,2}, y_{3,3} \in \mathcal{W}_3$. Next, we check point $y_{2,2}$. The nearest

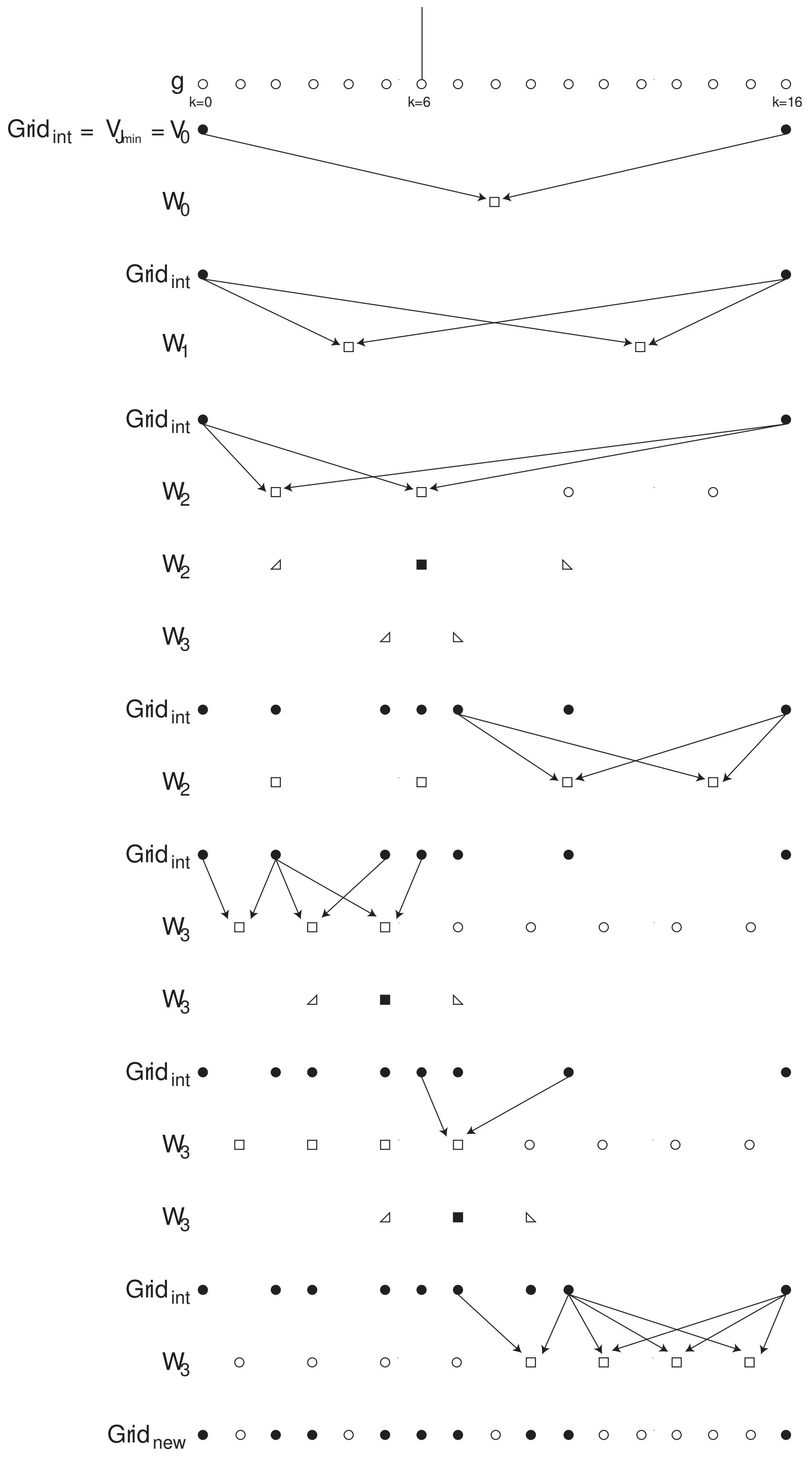


Figure 16: Demonstration of the proposed grid adaptation algorithm using Example 5.

points to $y_{2,2}$ in $\mathsf{Grid}_{\text{int}}$ are $y_{3,3}$ and $x_{0,1}$. Since $g(y_{2,2})$ can be interpolated from $y_{3,3}$ and $x_{0,1}$, we do not include $y_{2,2}$ in the grid. For the same reason, we do not include $y_{2,3}$. Now we move to the next level $\mathcal{W}_3$ and check points $y_{3,0}$ and $y_{3,1}$. Since both these points can be interpolated from the existing points in $\mathsf{Grid}_{\text{int}}$ we do not include these points in the grid. Subsequently we check $y_{3,2}$. Since $g(y_{3,2})$ cannot be interpolated from the nearest two points $y_{2,0}, y_{2,1} \in \mathsf{Grid}_{\text{int}}$ we include $y_{3,2}$ along with points $y_{3,1}$ and $y_{3,3}$ (which in any case is already present in $\mathsf{Grid}_{\text{int}}$) in $\mathsf{Grid}_{\text{int}}$. Moving on to the next point $y_{3,3}$, we see again that $g(y_{3,3})$ cannot be interpolated from the nearest two points $y_{2,1}, y_{2,2} \in \mathsf{Grid}_{\text{int}}$. Hence, we include $y_{3,3}$ along with $y_{3,2}$ (which in any case is already present in $\mathsf{Grid}_{\text{int}}$) and $y_{3,4}$ in $\mathsf{Grid}_{\text{int}}$. The next point in $\mathcal{W}_3$ is $y_{3,4}$. Since $g(y_{3,4})$ can be interpolated from the two nearest points $y_{3,3}, y_{2,2} \in \mathsf{Grid}_{\text{int}}$, we move on to the next point $y_{3,5}$. The nearest two points to $y_{3,5}$ in $\mathsf{Grid}_{\text{int}}$ are $y_{2,2}$, $x_{0,1}$, and since $g(y_{3,5})$ can be interpolated from these two points, we do not include $y_{3,5}$ in $\mathsf{Grid}_{\text{int}}$. For the same reason we do not add points $y_{3,6}$ and $y_{3,7}$. The final adaptive grid $\mathsf{Grid}_{\text{new}}$ is shown by the solid circles in Figure 16.

The adaptive grid generated using the previous algorithm depends on how we select points along the grid, that is, whether we move from left to right or from right to left across each level. It also depends on the location of the singularity. If the singularity is located in the middle, then it does not matter whether we move from left to right or from right to left. The result will be the same nonuniform grid. If on the other hand, the singularity is not in the middle, then the grid depends on the way in which we traverse across each level. To illustrate this fact, we again consider Example 5, but this time with $k = 1$. Hence, the impulse is located at $x = x_{4,1}$. If we go from left to right then the adaptive grid consists of the points $x_{4,0}$, $x_{4,1}$, $x_{4,3}$, $x_{4,5}$, $x_{4,16}$. If we go from right to left then the grid consists of the points $x_{4,0}, x_{4,1}, x_{4,3}, x_{4,16}$. Now let $k = 15$, which implies that the impulse is located at $x = x_{4,15}$. If we go from left to right then the grid consists of the points $x_{4,0}$, $x_{4,13}$, $x_{4,15}$, $x_{4,16}$, and if we go from right to left then the grid consists of the points $x_{4,0}, x_{4,11}, x_{4,13}, x_{4,15}, x_{4,16}$. Note that, in the proposed algorithm, it is not mandatory to traverse across a level only from the leftmost point or from the rightmost point. We can instead start from any point at that level, each time resulting in a different grid. This suggests that by using a suitable probability distribution function to choose the order in which the points at each particular level are selected, one may be able to further optimize the grid.

5.2.2 Grid Adaptation Approach of Alves et al. [3]

In this section, we briefly summarize the main idea underlying the grid adaptation approach of Alves et al. [3] which is based on the approach of Harten [68, 69]. For further details, the reader is referred to [3, 68, 69, 70].

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in equations (134) and (135) before. We explain pictorially, with the help of Example 5, how the approach of Alves et al. [3] works (see Figure 17). The symbols used in Figure 17 are the same as those used in Figure 16 except for a new symbol (a triangle facing down), which is used to show the parents of points in $\mathcal{W}_j$, that is, the points in $\mathcal{V}_j$ that are used for interpolating the function values at points in $\mathcal{W}_j$. In the approach of Alves et al., we first interpolate the function values at the points belonging to $\mathcal{W}_j$ from the corresponding points in $\mathcal{V}_j$ for $j = 0, \dots, 3$, respectively, as shown in Figure 17. Then we find the points which have interpolative error coefficients greater than the prescribed threshold ϵ. We see that points $y_{2,1},\ y_{3,2},\ y_{3,3}$ have interpolative error coefficients greater than the threshold (shown by solid squares) and add these points in the grid. Next, we add the points $y_{2,0}, y_{2,2}$ neighboring to $y_{2,1}$ in $\mathcal{W}_2$ (shown by level A) and the points $y_{3,2},\ y_{3,3}$ neighboring to $y_{2,1}$ in $\mathcal{W}_3$ (shown by level B). Similarly, we add points $y_{3,1},\ y_{3,3}$ neighboring to $y_{3,2}$ in $\mathcal{W}_3$ (shown by level C) and the points $y_{3,2},\ y_{3,4}$ neighboring to $y_{3,3}$ in $\mathcal{W}_3$ (shown by level D). Finally, we include the parents of all the points added previously to the adaptive grid (shown by levels $A_p,\ B_p,\ C_p$ and D_p) resulting in the nonuniform grid $\mathsf{Grid}_{\text{new}}$ shown in Figure 17.

5.2.3 Comparison with Existing Multiresolution-Based Approaches

The proposed grid adaptation algorithm results, in general, in a fewer number of grid points when compared to the grid adaptation algorithms of Harten [68, 69], Holmstrom [75], and the Alves et al. [3]. First, we explain why this is so and then we give several examples to demonstrate this fact.

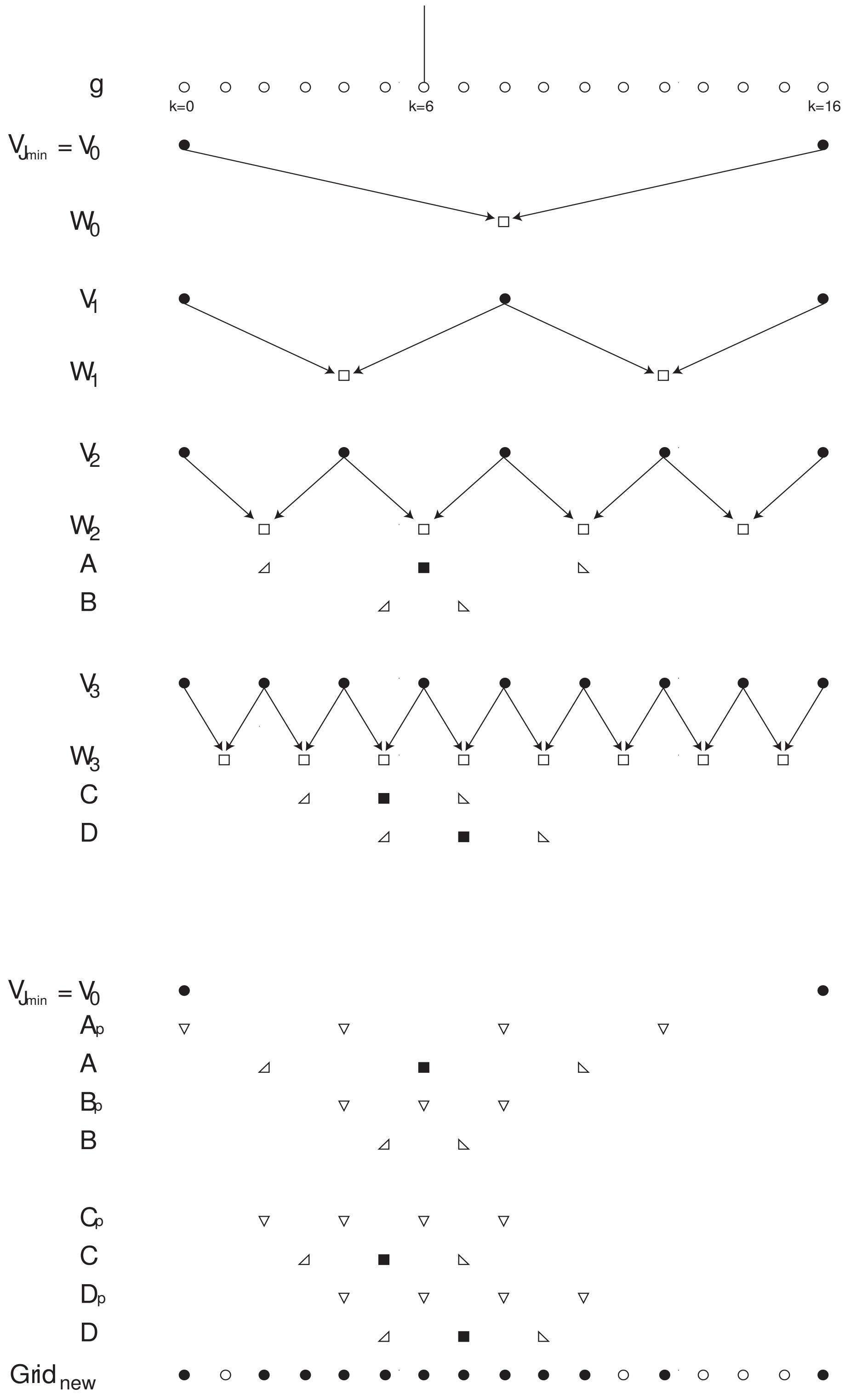


Figure 17: Demonstration of the grid adaptation approach of Alves et al. [3] using Example 5.

In the grid adaptation algorithm of existing approaches [68, 69, 75, 3], one interpolates $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ only from the function values at the points belonging to $\mathcal{V}_j$ for $j = J_{\min} \dots J_{\max} - 1$, and only then, one adds to the adaptive grid, the points $y_{j,k}$ along with the points $\{y_{j,k+\ell}\}_{\ell=-N_1}^{N_1}$ and $\{y_{j+1,2k+\ell}\}_{\ell=-N_2+1}^{N_2}$ for all the pairs (j, k), such that $d_{j,k} > \epsilon$. In the proposed method, we continuously keep on updating the adaptive grid instead. If the interpolative error coefficient at $y_{j,k}$, where $0 \leq k \leq 2^j - 1$ and $J_{\min} \leq j \leq J_{\max} - 1$, is greater than the prescribed threshold, we add $y_{j,k}$ to the adaptive grid, at the same time we also add to the adaptive grid the neighboring points at the same level $\{y_{j,k+\ell}\}_{\ell=-N_1}^{N_1}$, as well as the neighboring points at the next level $\{y_{j+1,2k+\ell}\}_{\ell=-N_2+1}^{N_2}$. We use the newly added points also for interpolating the remaining points at level $\mathcal{W}_j$ and the levels below it. In other words, in the proposed approach $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ are interpolated from the function values at the points in $\mathcal{V}_j \oplus \mathcal{W}_j \oplus \mathcal{W}_{j+1}$ for $J_{\min} \leq j \leq J_{\max} - 2$ and in $\mathcal{V}_j \oplus \mathcal{W}_j$ for $j = J_{\max} - 1$. Hence, by making use of the extra information from levels $\mathcal{W}_j$ (and $\mathcal{W}_{j+1}$), which in any case will be added to the adaptive grid, we are able to reduce the number of grid points in the final grid.

In Harten's approach, the solution at each time step is represented on the finest grid, and one encodes and decodes the solution at each time step in order to calculate the interpolative errors. In other words, the interpolative errors are computed at all points of the fine grid at each mesh refinement step. Holmstrom [75], on the other hand, calculates the interpolative error coefficients only at the points that are in the adaptive grid; if a function value is needed that does not exist in the present grid, the author interpolates the function value from a coarser scale recursively. In the algorithm of Alves et al. [3] one also adds to the grid the points that were used to predict the function values at all the previously added points in order to compute the interpolative error during the next mesh adaptation. Therefore, in the approach of Alves et al., when a point $y_{j,k}$ ($0 \leq k \leq 2^j - 1$ and $J_{\min} \leq j \leq J_{\max} - 1$), is added to the grid, one also include its parents, which were used to predict the function value at that point. The parents are not needed for approximating the given function to the prescribed accuracy, but are included just for calculating the interpolative error coefficient at the point $y_{j,k}$ during the next mesh adaptation. In the proposed algorithm, on the other hand, whenever a point is being checked for inclusion in the adaptive grid, we predict the function value at that point only from the points which already exist in the adaptive grid. Hence, if that point is inserted in the grid we do not need to add any extra points (i.e., its parents). This also alleviates the task of keeping track of the parents from the rest of the points as in the approach of Alves et al. and the task of recursively calculating the function values from the coarser resolution levels as is done in the approach of Holmstrom. Next we give several examples to compare the proposed grid adaptation approach with the algorithm of Alves et al. [3] for solving evolution PDEs.

Example 6

First we consider a very simple example. For this example we consider a dyadic grid $\mathcal{V}_4$ and the function

$$g(x) = \begin{cases} 1, & x = x_{4,k}, \\ 0, & \text{otherwise}, \end{cases} \tag{140}$$

with an impulse located at $x = k/2^4$, where $0 \leq k \leq 16$. Let $J_{\min} = 0$, $J_{\max} = 4$, $p = 1$, $\epsilon = 0.1$, and $N_1 = N_2 = 1$. Table 3 shows the number of grid points used by the proposed grid adaptation algorithm N_{p} and the number of points N_{A} used by the grid adaptation scheme of Alves et al. [3] for $k = 0, \ldots, 16$. We found that when the impulse is located at either the left boundary ($k = 0$) or the right boundary ($k = 16$) or in the middle of the domain ($k = 8$) both the approach of Alves et al. and the proposed approach result in the same grid. For all other cases, the grids generated are different. Moreover, we see that the proposed algorithm results in a fewer number of grid points. For this example, the proposed algorithm outperforms the algorithm of Alves et al. [3] by up to 33%.

Example 7

Next we again consider the functions g_1 and g_2 given by (130) and (131), respectively, and a grid with $J_{\min} = 2$ and $J_{\max} = 10$. This time we set $N_1 = N_2 = 1$ in the proposed grid adaptation algorithm. Table 4 gives the number of points used by the proposed grid adaptation algorithm N_{p} and the number of points N_{A} used by the grid adaptation scheme of Alves et al. [3]. For this example, we observe that the proposed grid adaptation algorithm

Table 3: Example 6. Comparison of the proposed algorithm with the algorithm of Alves et al.

k	N_{p}	N_{A}	$N_{\text{p}}/N_{\text{A}}$	k	N_{p}	N_{A}	$N_{\text{p}}/N_{\text{A}}$
0	9	9	1	9	6	9	0.67
1	5	6	0.83	10	9	12	0.75
2	7	9	0.78	11	6	9	0.67
3	6	8	0.75	12	11	12	0.92
4	11	12	0.92	13	5	7	0.71
5	6	9	0.67	14	7	9	0.78
6	9	12	0.75	15	4	6	0.67
7	6	9	0.67	16	9	9	1
8	13	13	1				

Table 4: Example 7. Comparison of the proposed algorithm with the algorithm of Alves et al.

	ϵ	N_{p}	N_{A}	$N_{\text{p}}/N_{\text{A}}$
g_1	10^{-3}	53	93	0.57
g_2	10^{-3}	108	185	0.58

outperforms the algorithm of Alves et al. [3] by up to 43%.

We are now ready to present the algorithm for solving the (IBVP) on an adaptive, nonuniform grid.

5.3 Numerical Solution of the IBVP for Evolution Equations

The numerical scheme for discretizing (IBVP) depends on $f(u_{xx}, u_x, u, x)$. The proposed grid adaptation algorithm will work for many numerically stable discretization schemes for (IBVP). We use different schemes for the numerical examples discussed in this chapter, depending on the problem. Hence, in the next section we only describe the techniques we use for calculating the spatial derivatives u_x and u_{xx} on the nonuniform grid and we state the numerical schemes in the examples themselves.

5.3.1 Calculation of Spatial Derivatives

To calculate the derivative u_x on the adaptive nonuniform grid $\mathsf{Grid}_{\text{new}}$ we use the weighted ENO (WENO) scheme [89, 90, 99, 110] on nonuniform grids. To this end, let the nonuniform grid be given as in (136). Now define

$$D^+ u^n_{j_i,k_i} = \frac{u^n_{j_{i+1},k_{i+1}} - u^n_{j_i,k_i}}{x_{j_{i+1},k_{i+1}} - x_{j_i,k_i}}, \qquad D^- u^n_{j_i,k_i} = \frac{u^n_{j_i,k_i} - u^n_{j_{i-1},k_{i-1}}}{x_{j_i,k_i} - x_{j_{i-1},k_{i-1}}}. \tag{141}$$

A third-order essentially nonoscillatory (ENO) approximation [71, 127, 128] to $(u_x^{\pm})^n_{j_i,k_i} = u_x^{\pm}(x_{j_i,k_i}, t_n)$ is given by one of the following expressions

$$((u_x^{\pm})^n_{j_i,k_i})_1 = \frac{v_1}{3} - \frac{7v_2}{6} + \frac{11v_3}{6}, \tag{142a}$$

or

$$((u_x^{\pm})^n_{j_i,k_i})_2 = -\frac{v_2}{6} + \frac{5v_3}{6} + \frac{v_4}{3}, \tag{142b}$$

or

$$((u_x^{\pm})^n_{j_i,k_i})_3 = \frac{v_3}{3} + \frac{5v_4}{6} - \frac{v_5}{6}, \tag{142c}$$

where for calculating $(u_x^-)^n_{j_i,k_i}$, we use $v_1 = D^-u^n_{j_{i-2},k_{i-2}}$, $v_2 = D^-u^n_{j_{i-1},k_{i-1}}$, $v_3 = D^-u^n_{j_i,k_i}$, $v_4 = D^-u^n_{j_{i+1},k_{i+1}}$, $v_5 = D^-u^n_{j_{i+2},k_{i+2}}$, and for calculating $(u_x^+)^n_{j_i,k_i}$, we use $v_1 = D^+u^n_{j_{i+2},k_{i+2}}$, $v_2 = D^+u^n_{j_{i+1},k_{i+1}}$, $v_3 = D^+u^n_{j_i,k_i}$, $v_4 = D^+u^n_{j_{i-1},k_{i-1}}$, $v_5 = D^+u^n_{j_{i-2},k_{i-2}}$. The basic idea behind a third-order ENO scheme is to choose either $((u_x^{\pm})^n_{j_i,k_i})_1$ or $((u_x^{\pm})^n_{j_i,k_i})_2$ or $((u_x^{\pm})^n_{j_i,k_i})_3$ for approximating $(u_x^{\pm})^n_{j_i,k_i}$ by choosing the smoothest possible polynomial interpolation of u.

It is reminded that a WENO approximation of $(u_x^{\pm})^n_{j_i,k_i}$ is a convex combination of the approximations in equations (142a), (142b) and (142c), that is,

$$(u_x^{\pm})^n_{j_i,k_i} = \sum_{\ell=1}^{3} \omega_\ell ((u_x^{\pm})^n_{j_i,k_i})_\ell, \tag{143}$$

where $0 \le \omega_\ell \le 1$ for $\ell = 1, 2, 3$ and $\omega_1 + \omega_2 + \omega_3 = 1$. The weights for fifth-order accuracy are given by [89, 110]

$$\omega_\ell = \frac{\alpha_\ell}{\alpha_1 + \alpha_2 + \alpha_3}, \qquad \ell = 1, 2, 3, \tag{144}$$

where,

$$\alpha_\ell = \frac{\bar{\alpha}_\ell}{(S_\ell + \delta)^2}, \qquad \ell = 1, 2, 3, \tag{145}$$

$$S_1 = \frac{13}{12}(v_1 - 2v_2 + v_3)^2 + \frac{1}{4}(v_1 - 4v_2 + 3v_3)^2, \tag{146}$$

$$S_2 = \frac{13}{12}(v_2 - 2v_3 + v_4)^2 + \frac{1}{4}(v_2 - v_4)^2, \tag{147}$$

$$S_3 = \frac{13}{12}(v_3 - 2v_4 + v_5)^2 + \frac{1}{4}(3v_3 - 4v_4 + v_5)^2, \tag{148}$$

and

$$\bar{\alpha}_1 = 0.1, \quad \bar{\alpha}_2 = 0.6, \quad \bar{\alpha}_3 = 0.3. \tag{149}$$

In (145) δ is used to prevent the denominator from becoming zero. In our computations, we have used $\delta = 10^{-6}$.

For the sake of brevity, we denote the cell walls by

$$x_{j_{i-1/2},k_{i-1/2}} = \frac{x_{j_{i-1},k_{i-1}} + x_{j_i,k_i}}{2}, \quad x_{j_{i+1/2},k_{i+1/2}} = \frac{x_{j_i,k_i} + x_{j_{i+1},k_{i+1}}}{2}. \tag{150}$$

In order to calculate $(u_{xx})^n_{j_i,k_i} = u_{xx}(x_{j_i,k_i}, t_n)$ on a nonuniform grid (136), we use the centered second difference scheme [137]

$$(u_{xx})^n_{j_i,k_i} = \frac{\left(\frac{u^n_{j_{i+1},k_{i+1}} - u^n_{j_i,k_i}}{x_{j_{i+1},k_{i+1}} - x_{j_i,k_i}} - \frac{u^n_{j_i,k_i} - u^n_{j_{i-1},k_{i-1}}}{x_{j_i,k_i} - x_{j_{i-1},k_{i-1}}} \right)}{x_{j_{i+1/2},k_{i+1/2}} - x_{j_{i-1/2},k_{i-1/2}}}. \tag{151}$$

5.3.2 Temporal Integration

Although the proposed grid adaptation algorithm of Section 5.2.1 will work for any numerically stable scheme, in this work we use the total variation diminishing (TVD) Runge-Kutta (RK) methods proposed by Shu and Osher in [110, 127] to increase the accuracy of the temporal discretization. While there are numerous RK schemes, these TVD RK schemes guarantee that no spurious oscillations are produced.

The basic first-order accurate TVD RK scheme is just the forward Euler method and is assumed to be TVD. Higher order accurate methods are obtained by sequentially taking Euler steps and combining the result with the initial data using a convex combination.

The second-order accurate TVD RK scheme is also known as the *midpoint rule.* First, an Euler step is taken to advance the solution to time $t_n + \Delta t_{n+1}$,

$$\frac{u^{n+1}_{j_i,k_i} - u^n_{j_i,k_i}}{\Delta t_{n+1}} + f\left((u_{xx})^n_{j_i,k_i}, (u^+_x)^n_{j_i,k_i}, (u^-_x)^n_{j_i,k_i}, x_{j_i,k_i}\right) = 0, \tag{152}$$

followed by a second Euler step to advance the solution to time $t_n + 2\Delta t_{n+1}$,

$$\frac{u^{n+2}_{j_i,k_i} - u^{n+1}_{j_i,k_i}}{\Delta t_{n+1}} + f\left((u_{xx})^{n+1}_{j_i,k_i}, (u^+_x)^{n+1}_{j_i,k_i}, (u^-_x)^{n+1}_{j_i,k_i}, x_{j_i,k_i}\right) = 0, \tag{153}$$

followed by an averaging step,

$$u^{n+1}_{j_i,k_i} = \frac{1}{2} u^n_{j_i,k_i} + \frac{1}{2} u^{n+2}_{j_i,k_i}, \tag{154}$$

that takes a convex combination of the initial data and the result of two Euler steps. The final averaging step produces the second-order accurate approximation to $u^{n+1}_{j_i,k_i}$.

The third-order accurate TVD RK scheme is as follows. First, an Euler step (152) is taken to advance the solution to time $t_n + \Delta t_{n+1}$, followed by a second Euler step (153) to advance the solution to time $t_n + 2\Delta t_{n+1}$, followed by an averaging step

$$u^{n+1/2}_{j_i,k_i} = \frac{3}{4}u^n_{j_i,k_i} + \frac{1}{4}u^{n+2}_{j_i,k_i}, \tag{155}$$

that produces an approximation to u at time $t_n + \frac{1}{2}\Delta t_{n+1}$ and at location x_{j_i,k_i}. Then another Euler step is taken to advance the solution to time $t_n + \frac{3}{2}\Delta t_{n+1}$,

$$\frac{u^{n+3/2}_{j_i,k_i} - u^{n+1/2}_{j_i,k_i}}{\Delta t_{n+1}} + f\left((u_{xx})^{n+1/2}_{j_i,k_i}, (u_x^+)^{n+1/2}_{j_i,k_i}, (u_x^-)^{n+1/2}_{j_i,k_i}, x_{j_i,k_i}\right) = 0, \tag{156}$$

followed by a second averaging step,

$$u^{n+1}_{j_i,k_i} = \frac{1}{3}u^n_{j_i,k_i} + \frac{2}{3}u^{n+3/2}_{j_i,k_i}, \tag{157}$$

that produces a third-order accurate approximation to $u^{n+1}_{j_i,k_i}$.

Now we are ready to give the algorithm for solving the IBVP for evolution equations (133).

5.3.3 Solution of the IBVP for Evolution PDEs

Based on the problem, the desired accuracy, and the computational hardware, we choose the minimum resolution level $J_{\min}$, the maximum resolution level $J_{\max}$, the threshold ϵ, the order of the interpolating polynomial p and the parameters N_1, N_2 required for the grid adaptation algorithm given in Section 5.2.1. The final time t_f is assumed to be given.

To solve (IBVP) on an adaptive grid, we first initialize $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\max}}$,

$$\mathcal{U}_{\text{old}} = \{g(x_{J_{\max},k})\}_{k=0}^{2^{J_{\max}}},$$

and set $t = 0$, $n = 0$[3]. Then the algorithm proceeds as follows:

[3]In case of hardware limitations, we suggest using $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\text{int}}}$, $\mathcal{U}_{\text{old}} = \{g(x_{J_{\text{int}},k})\}_{k=0}^{2^{J_{\text{int}}}}$, where $J_{\min} < J_{\text{int}} < J_{\max}$ is chosen based on the hardware limitations. Then, either if there are no discontinuities in the initial condition g or even if there are discontinuities in g that are self-sharpening, the algorithm will autonomously add points at higher resolution levels as we continue to move forward in time.

Step 1. Given $\mathsf{Grid}_{\text{old}}, \mathcal{U}_{\text{old}}$ find the new grid $\mathsf{Grid}_{\text{new}}$ and the function values at all the points in $\mathsf{Grid}_{\text{new}}, \mathcal{U}_{\text{new}} = \{u^n_{j,k} : x_{j,k} \in \mathsf{Grid}_{\text{new}}\}$, using the grid adaptation algorithm given in Section 5.2.1. The new grid $\mathsf{Grid}_{\text{new}}$ is the grid on which we will propagate the solution from time t to time $t_{\text{adapt}} = t + \Delta t_{\text{adapt}}$, where

$$\Delta t_{\text{adapt}} = \frac{N_1 \Delta x_{\min}}{\text{wave speed}}, \tag{158}$$

is the time after which the grid should be adapted again, calculated from the approximate time the solution will take to move N_1 grid points, and $\Delta x_{\min} = \min_{\mathsf{Grid}_{\text{new}}}(x_{j_{i+1},k_{i+1}} - x_{j_i,k_i})$. The reader is referred to [137, 110] for details on computing the wave speed. One can always use $\Delta t_{\text{adapt}} = \Delta t$ for the cases where the wave speed is either difficult or impossible to compute.

Step 2. Compute the solution at time $t = t_{n+1}$ at all the points belonging to $\mathsf{Grid}_{\text{new}}$, $\mathcal{U}_{\text{new}} = \{u^{n+1}_{j,k} : x_{j,k} \in \mathsf{Grid}_{\text{new}}\}$, using any numerically stable scheme, and increment n by 1. Keep on repeating this step while $t < t_{\text{adapt}}$. If $t \geq t_f$ terminate the algorithm.

Step 3. Reassign the sets: $\mathsf{Grid}_{\text{old}} \leftarrow \mathsf{Grid}_{\text{new}}, \mathcal{U}_{\text{old}} \leftarrow \mathcal{U}_{\text{new}}$. It should be noted that we do not interpolate the function values at the finest level during the mesh refinement process. In the proposed mesh refinement algorithm, we only check the retained points in $\mathsf{Grid}_{\text{old}}$ to further add and remove points in the grid. The interpolative error coefficients are computed only at the points $y_{j,k} \in \mathsf{Grid}_{\text{old}}$, and the solution $\mathcal{U}_{\text{old}}$ for all $y_{j,k} \in \mathsf{Grid}_{\text{old}}$ is known from the previous step.

Step 4. Goto Step 1.

Remark 5. As pointed out earlier, Δt_n is computed based on the Courant-Friedrichs-Levy (CFL) condition [137] for hyperbolic equations and the von Neumann condition [137] for all other evolution equations. For both CFL condition and the von Neumann condition Δt_n depends on $\Delta x_{\min}$. Hence, in the proposed algorithm Δt_n changes adaptively depending on $\Delta x_{\min}$, which also changes adaptively.

5.4 Numerical Examples

In this section, we present several examples to demonstrate the stability and robustness of our algorithm. These examples also illustrate the algorithm's ability to automatically capture and follow any existing or self-sharpening features of the solution that develop in time.

Example 8

First, we consider a nonlinear conservation law

$$u_t + (F(u))_x = 0. \tag{159}$$

For a specific example, we consider the inviscid Burgers' equation

$$u_t + \left(\frac{1}{2}u^2\right)_x = 0. \tag{160}$$

We use the same smooth initial condition and the Dirichlet boundary condition as in [3], that is,

$$g(x) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x), \quad u(0,t) = u(1,t) = 0, \tag{161}$$

to check the ability of the proposed algorithm to capture the shock. The solution is a wave that develops a very steep gradient and subsequently moves towards $x = 1$. Because of the zero boundary values, the wave amplitude diminishes with increasing time.

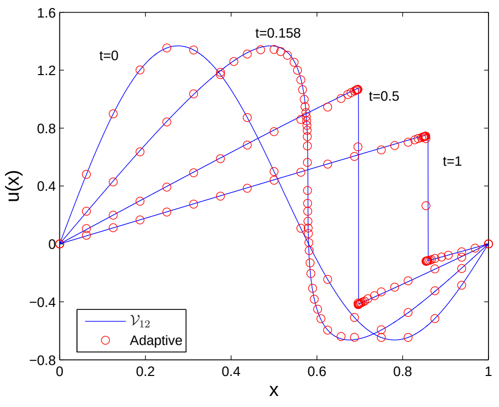
For solving (160)-(161), we use (71) along with the ENO-Roe scheme proposed by Shu and Osher [128] on a non-uniform grid for calculating the numerical flux functions $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}}$. For temporal integration, we use a third-order total variation diminishing (TVD) Runge–Kutta (RK) scheme [127]. The numerical solution at times $t = 0\,\text{s}$, $t = 0.158\,\text{s}$, $t = 0.5\,\text{s}$ and $t = 1\,\text{s}$ using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 18(a). The other parameters used in the grid adaptation procedure are $p = 3$, $\epsilon = 0.01$, $N_1 = N_2 = 1$. Figure 18 also shows the grid point distribution in the adaptive mesh at times $t = 0\,\text{s}$, $t = 0.1\,\text{s}$, $t = 0.158\,\text{s}$ and $t = 1\,\text{s}$. We see that as the shock continues to develop, the algorithm adds points at the finer levels of resolution in the region where the shock is developing, and removes points from the regions where the solution is getting smoother. Similar conclusions can be drawn by looking at the time evolution of the number of grid points (Figure 18(f)). We observe that the number of grid points increases as the shock continues to develop, and once the solution is smooth everywhere except for the region of the shock the number of grid points is pretty much steady, the number of grid points oscillates about a mean value of 43. This shows that the proposed strategy uses only the grid points that are actually necessary to attain a given precision, and the algorithm is able to add and remove points when and where is needed.

A comparison of CPU times for the uniform and adaptive grids, along with the L_1 error ($E_1(u)$) between the solution of the proposed multiresolution algorithm and the fine grid solution evaluated at grid $\mathcal{V}_{J_{\max}}$ and the number of grid points used by the proposed algorithm at the final time step for different $J_{\max} = 8, 9, 10, 11, 12$ are summarized in Table 5. We observe a major speed up in the computational time compared to the uniform mesh, and the speed-up factors increase at an approximate rate of two. The proposed approach results in speed-up factors that are higher than those reported in [3]. For scale $J_{\max} = 12$ the speed-up factor using the proposed approach is 63.7, which is about 27% higher than the one reported in [3]. It is reminded that we chose $N_1 = 1$ and Alves et al. chose $N_1 = 2$ which implies that in our case mesh refinement was performed twice as many times as was performed in [3] for the same problem and even then the speed-up factor is 27% higher than the one reported in [3]. The L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at times $t = 0.158\,\mathrm{s}$, $t = 0.5\,\mathrm{s}$ and $t = 1\,\mathrm{s}$ for $J_{\max} = 12$ have been summarized in Table 6.

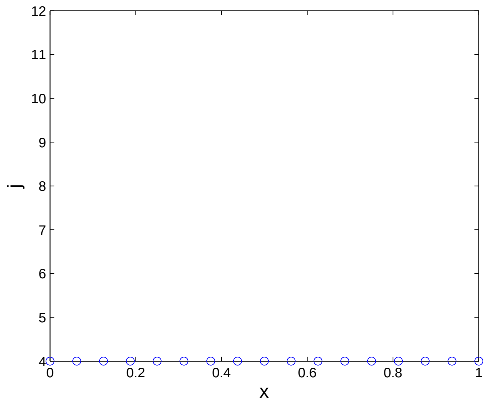
Table 5: Example 8. L_1 error and computational times for uniform vs. adaptive mesh.

	Uniform Mesh		Adaptive Mesh			
$J_{\max}$	N_p in $\mathcal{V}_{J_{\max}}$	t_{cpu} (s)	N_p at t_f in $\mathsf{Grid}_{\mathrm{new}}$	$E_1(u)$	t_{cpu} (s)	Speed Up
8	$2^8 + 1 = 257$	2.7106	31	7.1991×10^{-3}	0.5835	4.6454
9	$2^9 + 1 = 513$	9.3851	34	7.1717×10^{-3}	1.2737	7.3684
10	$2^{10} + 1 = 1025$	36.6631	37	7.7397×10^{-3}	2.6622	13.7717
11	$2^{11} + 1 = 2049$	223.8606	40	7.7220×10^{-3}	6.0399	37.0636
12	$2^{12} + 1 = 4097$	804.9415	43	7.8012×10^{-3}	12.6301	63.7320

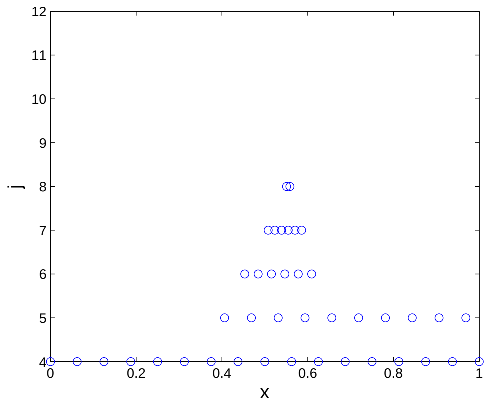
In the next two examples, we consider Hamilton-Jacobi equations, that is, evolution



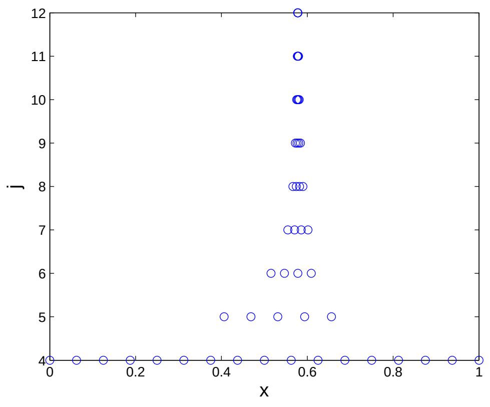
(a) Solution $u(x,t)$.



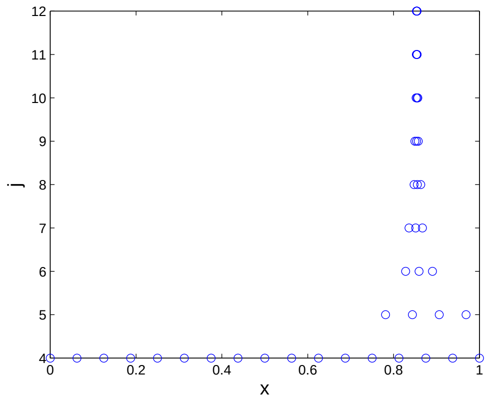
(b) Grid point distribution at $t = 0\,\mathrm{s}$.



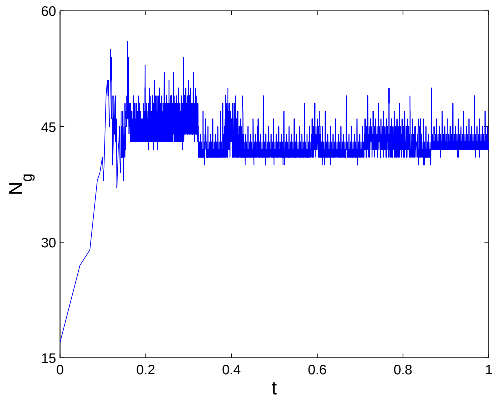
(c) Grid point distribution at $t = 0.1\,\mathrm{s}$.



(d) Grid point distribution at $t = 0.158\,\mathrm{s}$.



(e) Grid point distribution at $t = 1\,\mathrm{s}$.



(f) Time evolution of the number of grid points.

Figure 18: Example 8. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 1, \epsilon = 0.01,\ N_1 = N_2 = 1$.

Table 6: Example 8. L_1 errors at different times for $J_{\max} = 12$.

t	$N_{\rm p}$ in $\mathsf{Grid}_{\rm new}$	C (%)	$E_1(u)$
0.158	49	98.80	8.2093×10^{-3}
0.5	42	98.97	9.3552×10^{-3}
1	43	98.95	7.8012×10^{-3}

equations as in (133), where

$$f(u_{xx}, u_x, u, x) = f(u_x). \tag{162}$$

For discretizing $f(u_x)$ we use the Lax-Friedrich's (LF) scheme [43, 110, 111, 128]

$$f(u_x) = \hat{f}^{\rm LF}(u_x^-, u_x^+) = f\left(\frac{u_x^+ + u_x^-}{2}\right) - \frac{1}{2}\alpha^x(u_x^+ - u_x^-), \tag{163}$$

where, $\alpha^x = \max_{u_x \in I^x} |f_1(u_x)|$, f_1 is the partial derivative of f with respect to u_x, $I^x = [u_x^{\min}, u_x^{\max}]$, and the minimum and the maximum values of u_x are identified by considering all the values of u_x^- and u_x^+ on the nonuniform grid.

Example 9

First, we consider the HJ equation with convex $f(u_x)$ taken from [111]

$$u_t + \frac{(u_x + 1)^2}{2} = 0, \tag{164}$$

with the initial condition and the periodic boundary condition as in [111], that is,

$$g(x) = -\cos \pi x, \quad u(-1, t) = u(1, t), \quad -1 \le x < 1. \tag{165}$$

For solving the problem using the proposed algorithm, we first convert the above mentioned problem from $x \in [-1, 1]$ to $\hat{x} \in [0, 1]$ by using a simple change of variables $x = 2\hat{x} - 1$. With a slight abuse of notation, we denote $\hat{x}$ by x, and hence, the problem (164)-(165) transforms to

$$u_t + \frac{(\frac{1}{2}u_x + 1)^2}{2} = 0, \tag{166}$$

with the initial condition and the periodic boundary condition,

$$g(x) = -\cos \pi(2x - 1), \quad u(0, t) = u(1, t). \tag{167}$$

The derivatives u_x^+, u_x^- in the LF discretization are approximated using a WENO scheme and the temporal integration is performed using a third-order TVD RK scheme. The numerical solution at times $t = 0\,\mathrm{s}$, $t = 1.5/\pi^2\,\mathrm{s}$, $t = 3.5/\pi^2\,\mathrm{s}$, $t = 7/\pi^2\,\mathrm{s}$, $t = 10/\pi^2\,\mathrm{s}$, and $t = 14/\pi^2\,\mathrm{s}$ using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 19(a). The other parameters used in the grid adaptation algorithm are $p = 3$, $\epsilon = 0.001$, $N_1 = N_2 = 2$. Figure 19 also shows the grid point distribution in the adaptive mesh at times $t = 0\,\mathrm{s}$, $t = 3.5/\pi^2\,\mathrm{s}$, $t = 7/\pi^2\,\mathrm{s}$, and $t = 14/\pi^2\,\mathrm{s}$. We see that as the kink continues to develop the algorithm adds points at the finer levels of resolution in the region where the kink is developing, and removes points from the regions where the solution is getting smoother and smoother. As the HJ equation (166) continues to evolve further in time, the discontinuity in the first derivative of the solution is smoothing out and as a result the algorithm starts removing points from the finer levels of resolution. This, again, demonstrates that the proposed strategy uses only the grid points that are actually necessary to attain a given precision, and the algorithm is able to add and remove points when and where is needed. The L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at times $t = 1.5/\pi^2\,\mathrm{s}$, $t = 3.5/\pi^2\,\mathrm{s}$, $t = 7/\pi^2\,\mathrm{s}$, $t = 10/\pi^2\,\mathrm{s}$, and $t = 14/\pi^2\,\mathrm{s}$ for $J_{\max} = 12$ have been summarized in Table 7.

Table 7: Example 9. L_1 errors at different times for $J_{\max} = 12$.

t	N_p in $\mathsf{Grid}_\mathrm{new}$	C (%)	$E_1(u)$
$1.5/\pi^2$	52	98.73	1.7603×10^{-3}
$3.5/\pi^2$	50	98.78	4.2828×10^{-3}
$7/\pi^2$	39	99.05	5.4194×10^{-3}
$10/\pi^2$	44	98.93	6.0113×10^{-3}
$14/\pi^2$	31	99.24	6.5118×10^{-3}

Example 10

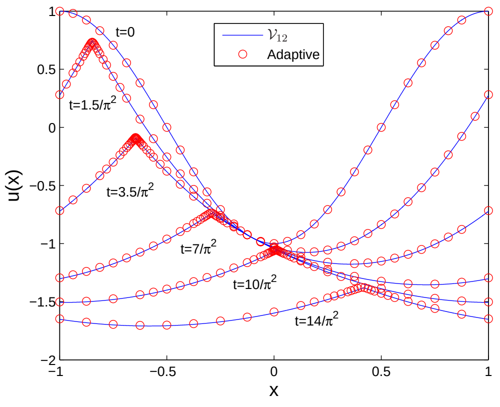
Next, we consider the Hamilton-Jacobi (HJ) equation with non-convex $f(u_x)$,

$$u_t - \cos(\alpha u_x + 1) = 0, \tag{168}$$

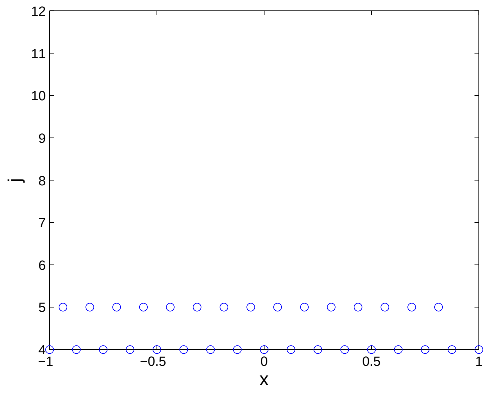
with

$$g(x) = -\cos\pi(2x-1), \quad u(0,t) = u(1,t), \tag{169}$$

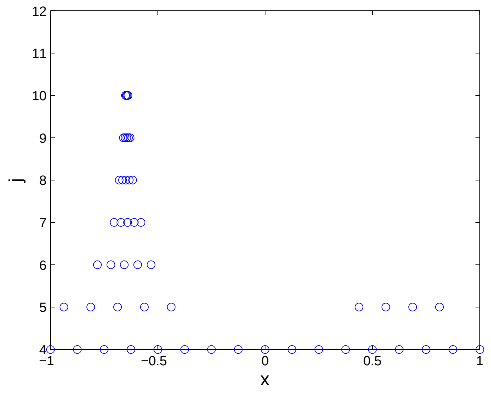
where α is a constant. We again use an LF scheme (163) for solving the IBVP (168)-(169). The derivatives u_x^+, u_x^- in the LF discretization are approximated using a WENO scheme and the temporal integration is performed using a third-order TVD RK scheme.



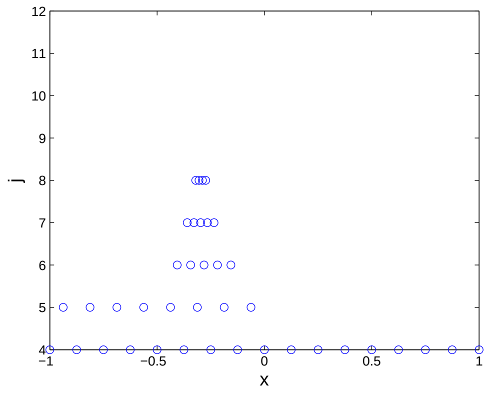
(a) Solution $u(x,t)$.



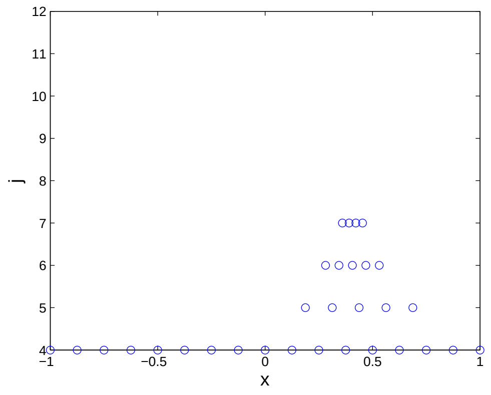
(b) Grid point distribution at $t = 0\,$s.



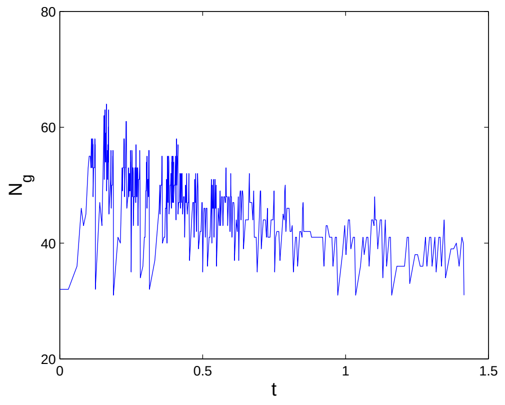
(c) Grid point distribution at $t = 3.5/\pi^2\,$s.



(d) Grid point distribution at $t = 7/\pi^2\,$s.



(e) Grid point distribution at $t = 14/\pi^2\,$s.



(f) Time evolution of the grid points.

Figure 19: Example 9. Parameters used in the simulation are $J_{\text{min}} = 4,\ J_{\text{max}} = 12,\ p = 3, \epsilon = 0.001,\ N_1 = N_2 = 2.$

The choice of $\alpha = 0.5$ results in the commonly used test problem for 1-D HJ equations given in [111]. In order to make the problem more interesting and challenging, in this work, we consider two more choices for α, namely, $\alpha = 1$ and $\alpha = 1.5$. The choices $\alpha = 1$ and $\alpha = 1.5$ result in more kinks in the solution at time $t = 1.5/\pi^2$ s. The numerical solutions for all the cases at time $t = 1.5/\pi^2$ s using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ along with the corresponding grid point distributions are shown in Figure 20. The other parameters used in the grid adaptation algorithm are $p = 3$, $\epsilon = 0.001$, $N_1 = N_2 = 2$. The solutions at $t = 1.5/\pi^2$ s for $\alpha = 0.5$, 0.1, and 1.5 have two, four, and six kinks respectively. We once again observe that the proposed algorithm is able to capture all the kinks in the solutions accurately and efficiently by adding points at the finer resolution levels in the region of kinks, while resolving the smoother regions using only the points at the coarse resolution levels. The L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at time $t = 1.5/\pi^2$ s, for $\alpha = 0.5,\ 1,\ 1.5$ and $J_{\max} = 12$ have been summarized in Table 8.

Table 8: Example 10. L_1 errors at different times for $J_{\max} = 12$.

α	$N_{\rm p}$ in Grid$_{\rm new}$	C (%)	$E_1(u)$
0.5	44	98.93	1.1259×10^{-3}
1	120	97.07	8.8733×10^{-4}
1.5	125	96.95	5.6106×10^{-4}

Example 11

Consider the scalar reaction-diffusion problem that appears in combustion problems [77, 113]

$$u_t - u_{xx} - \frac{Re^{\delta}}{a\delta}(1 + a - u)e^{-\delta/u} = 0, \tag{170}$$

$$u_x(0,t) = 0, \quad u(1,t) = 1, \quad u(x,0) = 1. \tag{171}$$

The solution u represents the temperature of a reactant in a chemical system, a is the heat release, δ is the activation energy, and R is the reaction rate. For small times the temperature gradually increases from unity with a "hot spot" forming at $x = 0$. After some finite time, ignition occurs and the temperature at $x = 0$ jumps rapidly from near unity to near $1 + a$. A flame front then forms and propagates towards $x = 1$ with a speed proportional to $e^{a\delta}/2(1 + a)$. In real problems, a is close to unity and δ is large, thus the flame front moves exponentially fast after the ignition. We use the same problem parameters as in [77, 113] namely, $a = 1$, $R = 5$, and $\delta = 30$. This is the same problem as the one given in [1], except for the value of the parameter δ, which in [1] is taken to be 20. Instead, we consider $\delta = 30$ as in [77, 113], since the flame layer in this case is much thinner, and higher mesh adaptation is required. We use (151) to discretize u_{xx}, and use a third-order TVD RK scheme for temporal integration. To illustrate how we apply Neumann boundary condition on a nonuniform grid we again consider a grid of the form (136). To apply the Neumann boundary condition, $u_x(0,t) = 0$, we introduce a fictitious node $x_{j_{-1},k_{-1}} = -x_{j_1,k_1}$, which lies outside the physical domain[4], and approximate the boundary condition by

$$(u_x)^n_{j_0,k_0} = \frac{u^n_{j_1,k_1} - u^n_{j_{-1},k_{-1}}}{x_{j_1,k_1} - x_{j_{-1},k_{-1}}} = 0, \tag{172}$$

which implies $u^n_{j_{-1},k_{-1}} = u^n_{j_1,k_1}$. Hence, at the boundary $x = 0$, equation (151) reduces to

$$(u_{xx})^n_{j_0,k_0} = \frac{2\left(\frac{u^n_{j_1,k_1}-u^n_{j_0,k_0}}{x_{j_1,k_1}-0} - \frac{u^n_{j_0,k_0}-u^n_{j_{-1},k_{-1}}}{0-x_{j_{-1},k_{-1}}}\right)}{x_{j_1,k_1} - x_{j_{-1},k_{-1}}} = \frac{2(u^n_{j_1,k_1} - u^n_{j_0,k_0})}{(x_{j_1,k_1})^2}. \tag{173}$$

The numerical solutions at times $t = 0\,\text{s}$, $t = 0.24\,\text{s}$, $t = 0.241\,\text{s}$ and $t = 0.244\,\text{s}$ using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 21(a). The other parameters used in the grid adaptation algorithm are $p = 3$, $\epsilon = 10^{-5}/2^{J_{\max}-j}$, $N_1 = N_2 = 1$. One of the main challenges of this problem is the fact that one needs to use a very small time step to capture the transition layer during the time of ignition. This is achieved automatically by the proposed algorithm since the algorithm is adaptive both in time and space. As the mesh gets refined, Δt_n in the proposed algorithm for the solution of evolution PDEs given in Section 5.3.3 also decreases. We see from Figure 21(f) that for time $t < 0.195\,\text{s}$ the proposed algorithm found the solution using only about 50 to 65 points. Starting from $t = 0.195\,\text{s}$ to $t = 0.2385\,\text{s}$, the algorithm slowly increased the number of points to around 95 points and, thereafter, efficiently added points at finer levels starting at $t = 0.2385\,\text{s}$. As the points from finer grid levels are being added, the algorithm automatically decreases the time step and is able to capture the transition layer during the time of ignition. The

[4]Note that $x_{j_0,k_0} = 0$.

L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at times $t = 0.24\,\mathrm{s}$, $t = 0.241\,\mathrm{s}$ and $t = 0.244\,\mathrm{s}$ for $J_{\max} = 12$ have been summarized in Table 9.

Table 9: Example 11. L_1 errors at different times for $J_{\max} = 12$.

t	N_p in $\mathsf{Grid}_\mathrm{new}$	C (%)	$E_1(u)$
0.24	137	96.66	4.6714×10^{-4}
0.241	227	94.46	3.4834×10^{-3}
0.244	180	95.61	3.2098×10^{-3}

Example 12

Finally, we consider a Riemann initial value problem (shock tube) for the Euler equations of gas dynamics, as follows

$$u_t + f(u)_x = 0, \tag{174}$$

$$u(x,0) = \begin{cases} u_\mathrm{L}, & x < 0.5, \\ u_\mathrm{R}, & x > 0.5, \end{cases} \tag{175}$$

where

$$u = [\rho\ m\ E]^\mathrm{T}, \tag{176}$$

$$f(u) = \nu u + [0\ p\ p\nu]^\mathrm{T}, \tag{177}$$

ρ, m, E are the gas density, momentum, total energy per unit volume, respectively, $\nu = m/\rho$ is the velocity, and

$$p = (\gamma - 1)\left(E - \frac{\rho\nu^2}{2}\right), \tag{178}$$

is the pressure. In (178) γ is the ratio of specific heat, which takes the usual value of 1.4 (for air). We consider the two well-known problems, namely, Sod's problem [129], the initial data for which is given by

$$u_\mathrm{L} = [1\ 0\ 2.5]^\mathrm{T}, \quad u_\mathrm{R} = [0.125\ 0\ 0.25]^\mathrm{T}, \tag{179}$$

and Lax's problem [96], the initial data for which is given by

$$u_\mathrm{L} = [0.445\ 0.698\ 8.82]^\mathrm{T}, \quad u_\mathrm{R} = [0.5\ 0\ 1.4275]^\mathrm{T}. \tag{180}$$

We use the characteristic numerical scheme given in [128, 110] for solving this problem. The basic idea behind the characteristic scheme is to transform the nonlinear system (174) to a system of (nearly) independent scalar conservation laws, and discretize each scalar conservation law independently in an upwind biased fashion. Then we transform the discretized system back to the original variables. We use the ENO-Roe fix (ENO-RF) scheme [128] on a non-uniform grid for obtaining the numerical flux function $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}}$ in the scalar field, and we use a third-order TVD RK scheme for temporal integration.

The numerical solution of the density $\rho(x,t)$, the velocity $\nu(x,t)$, the pressure $p(x,t)$, the internal energy per unit mass $e(x,t)$ ($e = p/(\gamma-1)\rho$), and the grid point distributions in the adaptive mesh for Sod's problem and Lax's problem at times $t = 0.2$ s, $t = 0.13$ s respectively, using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 22 and Figure 23 respectively. The other parameters used in the grid adaptation procedure are $p = 3$, $\epsilon = 0.001$ and $N_1 = N_2 = 2$. Figures 22, 23 also show the time evolution of the number of grid points for both Sod's and Lax's problems. The L_1 errors ($E_1(\rho)$, $E_1(m)$, $E_1(E)$) along with the number of grid points used by the proposed algorithm for solving Sod's problem at times $t = 0.05$ s, $t = 0.1$ s, $t = 0.15$ s, and $t = 0.2$ s and Lax's problem at times $t = 0.05$ s, $t = 0.1$ s, and $t = 0.13$ s for $J_{\max} = 12$ are summarized in Table 10 and Table 11, respectively.

Table 10: Example 12. Sod's problem. L_1 errors at different times for $J_{\max} = 12$.

t	$N_{\rm p}$ in $\mathsf{Grid}_{\rm new}$	C (%)	$E_1(\rho)$	$E_1(m)$	$E_1(E)$
0.05	212	94.83	1.0300×10^{-4}	1.1859×10^{-4}	2.9885×10^{-4}
0.1	189	95.39	2.8712×10^{-4}	3.2164×10^{-4}	8.3684×10^{-4}
0.15	173	95.78	4.9362×10^{-4}	5.4437×10^{-4}	1.4215×10^{-3}
0.2	195	95.24	7.8443×10^{-4}	8.1571×10^{-4}	2.1954×10^{-3}

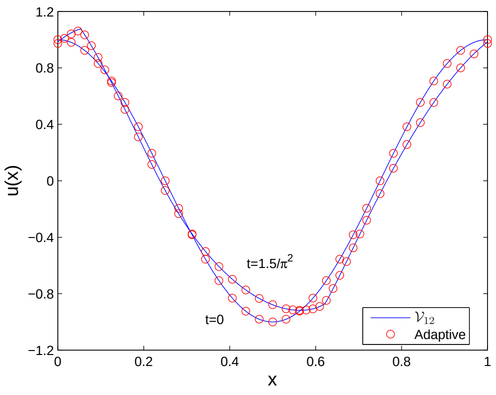
Table 11: Example 12. Lax's problem. L_1 errors at different times for $J_{\max} = 12$.

t	$N_{\rm p}$ in $\mathsf{Grid}_{\rm new}$	C (%)	$E_1(\rho)$	$E_1(m)$	$E_1(E)$
0.05	272	93.36	5.6092×10^{-5}	1.2380×10^{-4}	7.7312×10^{-4}
0.1	270	93.41	1.8641×10^{-4}	4.1361×10^{-4}	3.3487×10^{-3}
0.13	267	93.48	2.7005×10^{-4}	5.9612×10^{-4}	4.9735×10^{-3}

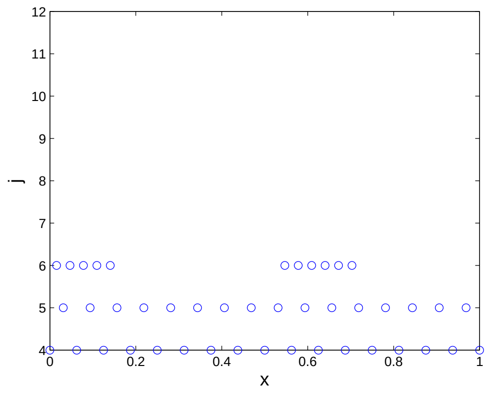
5.5 Summary

In this chapter, we have proposed a novel multiresolution grid adaptation algorithm for solving evolution equations. The proposed algorithm for solving evolution PDEs is adaptive both in space and time. The algorithm is shown to outperform similar grid adaptation schemes in the literature. Several examples have demonstrated the stability and robustness of the proposed algorithm. In all examples considered, the algorithm adapted dynamically to any existing or emerging irregularities in the solution, by automatically allocating more grid points to the region where the solution exhibited sharp features and fewer points to the region where the solution was smooth. As a result, the computational time and memory usage can be reduced significantly, while maintaining an accuracy equivalent to the one obtained using a fine uniform mesh.

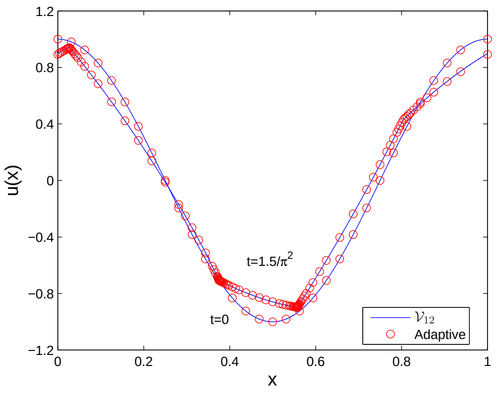
Next, we move on to our main motivation behind this work, that is, develop fast and efficient algorithms for solving optimal control problems.



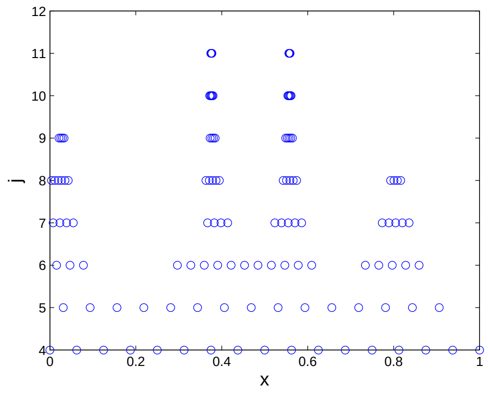
(a) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 0.5$.



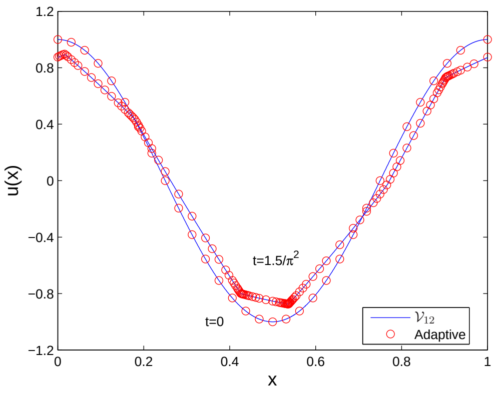
(b) Grid point distribution at $t = 1.5/\pi^2$ s for $\alpha = 0.5$.



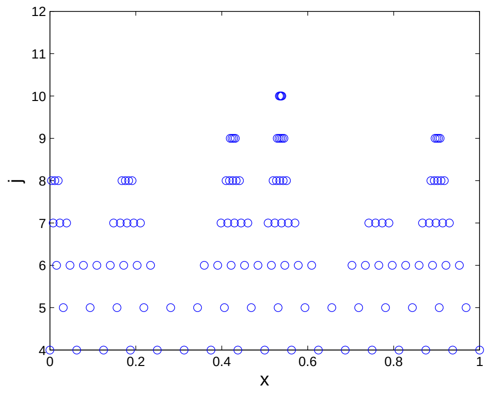
(c) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 1$.



(d) Grid point distribution at $t = 1.5/\pi^2$ s for $\alpha = 1$.

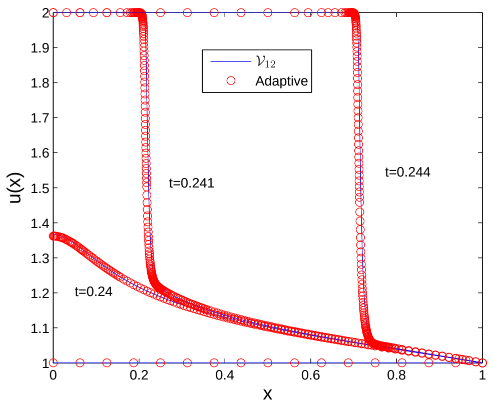


(e) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 1.5$.

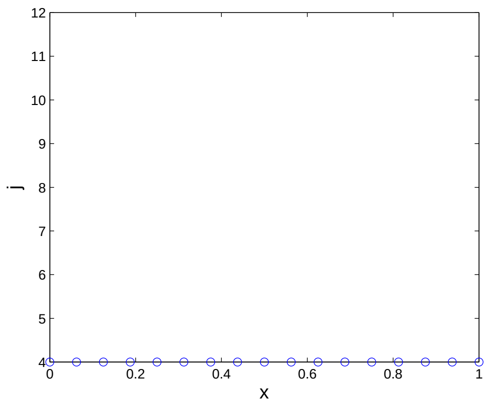


(f) Grid point distribution at $t = 1.5/\pi^2$ s for $\alpha = 1.5$.

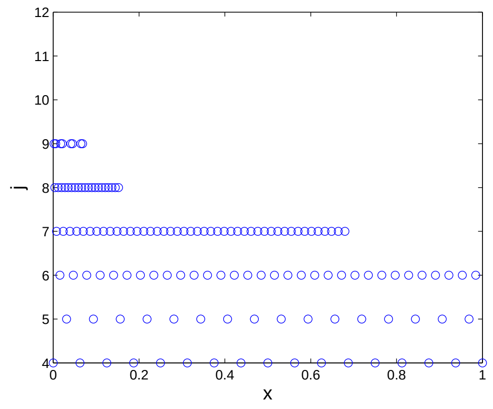
Figure 20: Example 10. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 0.001,\ N_1 = N_2 = 2$.



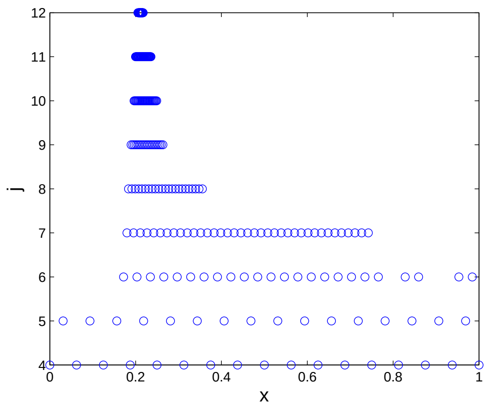
(a) Solution $u(x,t)$.



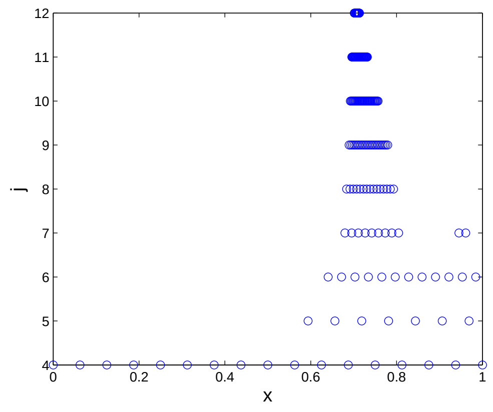
(b) Grid point distribution at $t = 0\,\mathrm{s}$.



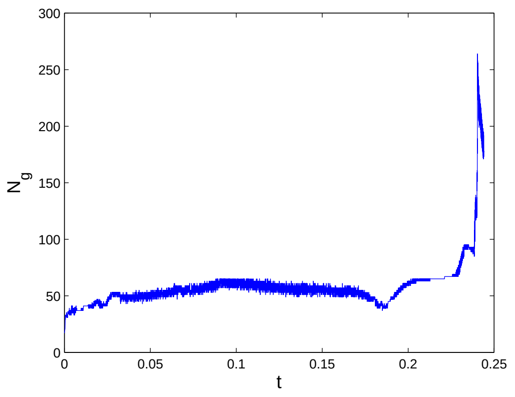
(c) Grid point distribution at $t = 0.24\,\mathrm{s}$.



(d) Grid point distribution at $t = 0.241\,\mathrm{s}$.

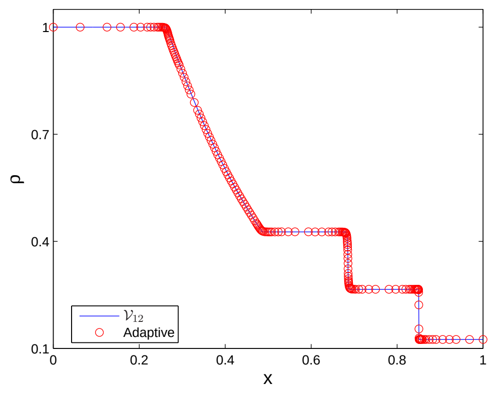


(e) Grid point distribution at $t = 0.244\,\mathrm{s}$.

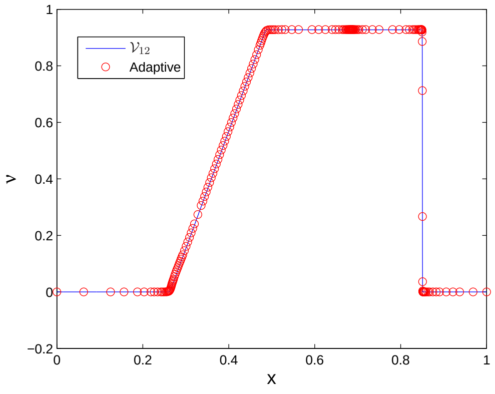


(f) Time evolution of the number of grid points.

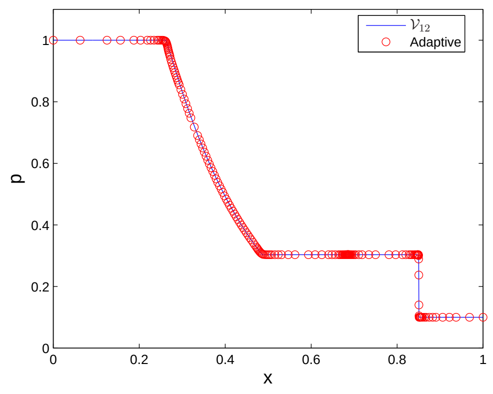
Figure 21: Example 11. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 10^{-5}/2^{J_{\max}-j},\ N_1 = N_2 = 1.$



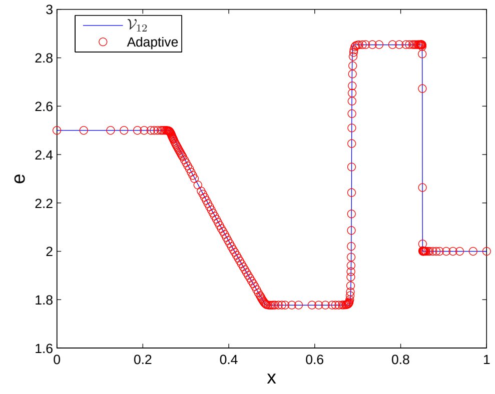
(a) Solution $\rho(x, 0.2)$.



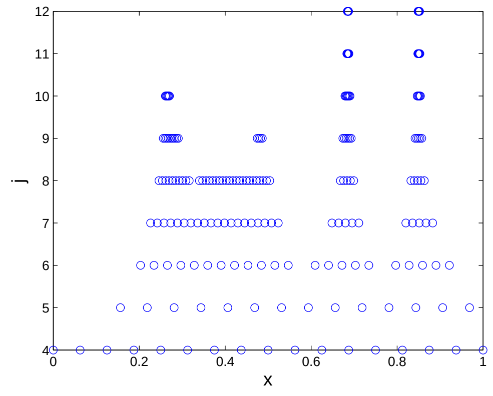
(b) Solution $\nu(x, 0.2)$.



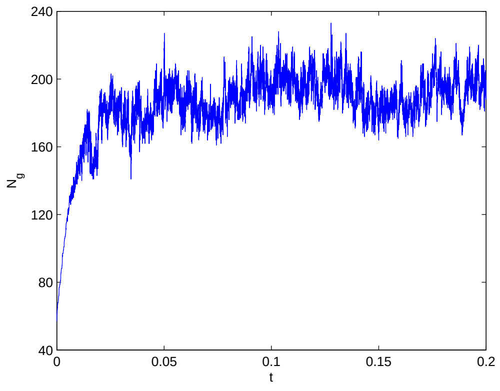
(c) Solution $p(x, 0.2)$.



(d) Solution $e(x, 0.2)$.

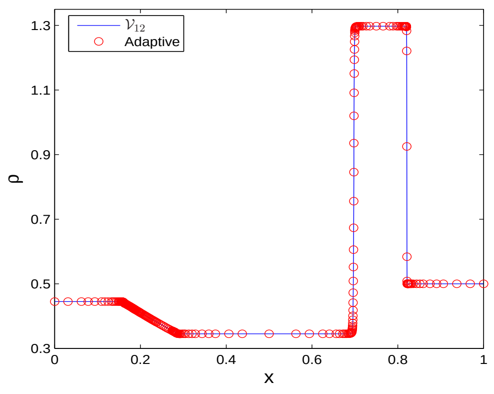


(e) Grid point distribution.

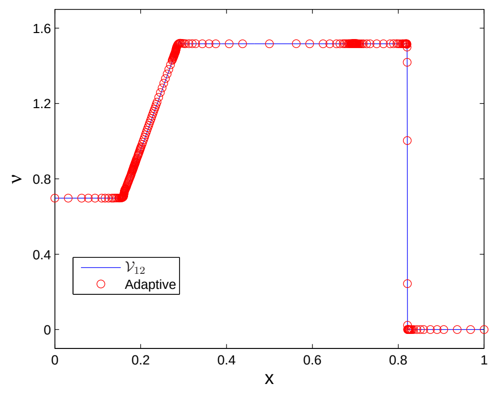


(f) Time evolution of grid points.

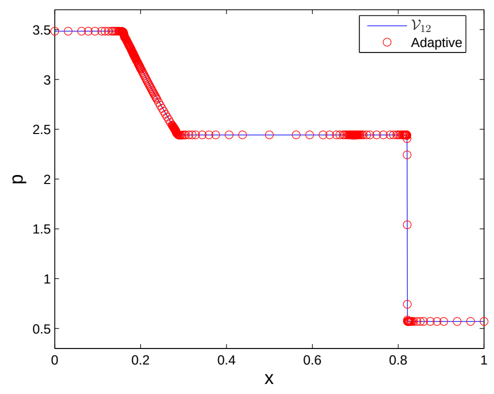
Figure 22: Example 12. Sod's problem. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 0.001,\ N_1 = N_2 = 2$.



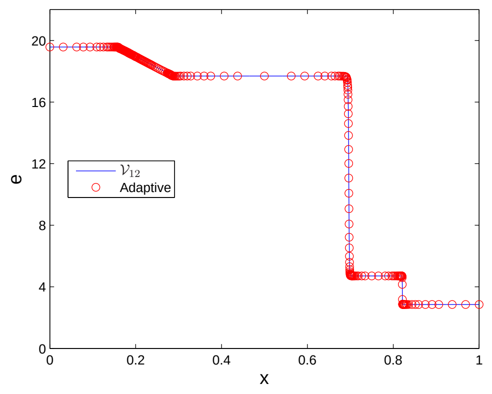
(a) Solution $\rho(x, 0.13)$.



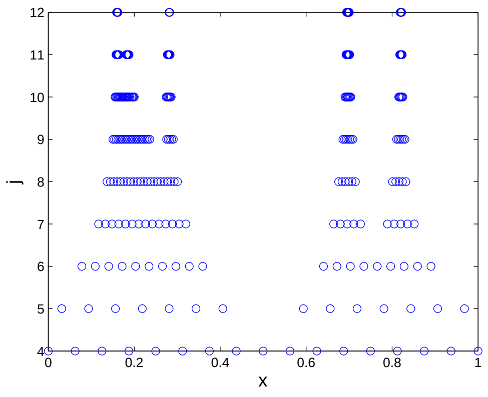
(b) Solution $\nu(x, 0.13)$.



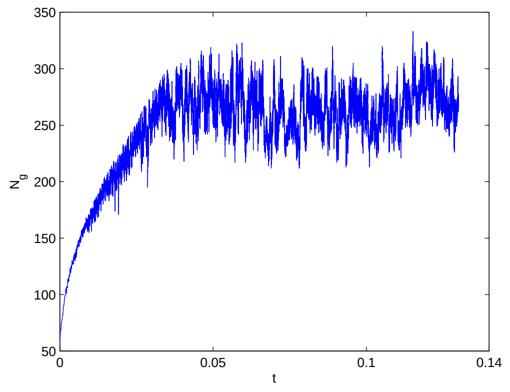
(c) Solution $p(x, 0.13)$.



(d) Solution $e(x, 0.13)$.



(e) Grid point distribution.



(f) Time evolution of grid points.

Figure 23: Example 12. Lax's problem. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 0.001,\ N_1 = N_2 = 2$.

CHAPTER VI

OPTIMAL CONTROL

6.1 Problem Formulation

Consider the following optimal control problem with Bolza cost functional, which we call the primal problem and denote it by P.

6.1.1 Primal Problem P

The problem is to determine the state $\mathbf{x}(\cdot)$ and the control $\mathbf{u}(\cdot)$ that minimize the Bolza cost functional,

$$J = e(\mathbf{x}(\tau_f), \tau_f) + \int_{\tau_0}^{\tau_f} L(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)\mathrm{d}\tau, \tag{181}$$

where $e : \mathbb{R}^{N_x} \times \mathbb{R}_+ \to \mathbb{R}$, $\tau \in [\tau_0, \tau_f]$, $\mathbf{x} : [\tau_0, \tau_f] \to \mathbb{R}^{N_x}$, $\mathbf{u} : [\tau_0, \tau_f] \to \mathbb{R}^{N_u}$, $L : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times [\tau_0, \tau_f] \to \mathbb{R}$, subject to the state dynamics

$$\dot{\mathbf{x}}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau), \tag{182}$$

the boundary conditions

$$\mathbf{x}(\tau_0) = \mathbf{x}_0, \quad \mathbf{e}_f(\mathbf{x}(\tau_f), \tau_f) = 0, \tag{183}$$

where $\mathbf{e}_f : \mathbb{R}^{N_x} \times \mathbb{R}_+ \to \mathbb{R}^{N_e}$, and the constraints

$$\mathbf{C}_{\mathrm{u}}(\mathbf{u}(\tau)) \leq 0, \quad \mathbf{C}_{\mathrm{x}}(\mathbf{x}(\tau)) \leq 0, \quad \mathbf{C}_{\mathrm{xu}}(\mathbf{x}(\tau), \mathbf{u}(\tau)) \leq 0, \tag{184}$$

where $\mathbf{C}_{\mathrm{u}} : \mathbb{R}^{N_u} \to \mathbb{R}^{N_{C_{\mathrm{u}}}}$, $\mathbf{C}_{\mathrm{x}} : \mathbb{R}^{N_x} \to \mathbb{R}^{N_{C_{\mathrm{x}}}}$, $\mathbf{C}_{\mathrm{xu}} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_{C_{\mathrm{xu}}}}$. The initial time τ_0 is assumed to be given and the final time τ_f can be fixed or free.

Next, we define the dual problem.

6.1.2 Problem P^λ

For the sake of simplicity, in this section, we denote the inequality constraints by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{\mathrm{u}} \\ \mathbf{C}_{\mathrm{x}} \\ \mathbf{C}_{\mathrm{xu}} \end{bmatrix} \leq 0, \tag{185}$$

where $\mathbf{C} : \mathbb{R}^{N_u} \times \mathbb{R}^{N_x} \to \mathbb{R}^{N_C}$, $N_C = N_{C_\mathrm{u}} + N_{C_\mathrm{x}} + N_{C_\mathrm{xu}}$. Now by adjoining the dynamic constraints (182), the constraints (185), and the boundary condition (183) to J, we obtain the augmented performance index,

$$J' = e(\mathbf{x}(\tau_f), \tau_f) + \nu^T \mathbf{e}_f(\mathbf{x}(\tau_f), \tau_f) + \int_{\tau_0}^{\tau_f} (L + \lambda^T(\mathbf{f} - \dot{\mathbf{x}}) + \mu^T \mathbf{C})\mathrm{d}\tau, \tag{186}$$

where $\lambda \in \mathbb{R}^{N_x}$, $\nu \in \mathbb{R}^{N_e}$, $\mu \in \mathbb{R}^{N_C}$, and

$$\mu_i = \begin{cases} 0, & C_i < 0, \\ > 0, & C_i = 0, \end{cases} \tag{187}$$

for $i = 1, \ldots, N_C$.

The Hamiltonian is defined to be

$$H = L + \lambda^T \mathbf{f} + \mu^T \mathbf{C}. \tag{188}$$

Let us define

$$E(\mathbf{x}(\tau_f), \tau_f) = e(\mathbf{x}(\tau_f), \tau_f) + \nu^T \mathbf{e}_f(\mathbf{x}(\tau_f), \tau_f). \tag{189}$$

Hence,

$$J' = E(\mathbf{x}(\tau_f), \tau_f) + \int_{\tau_0}^{\tau_f} (H - \lambda^T \dot{\mathbf{x}})\mathrm{d}\tau. \tag{190}$$

Now consider the integral

$$\begin{aligned} \int_{\tau_0}^{\tau_f} \lambda^T(\tau)\dot{\mathbf{x}}(\tau)\mathrm{d}\tau &= \lambda^T(\tau)\mathbf{x}(\tau)|_{\tau_0}^{\tau_f} - \int_{\tau_0}^{\tau_f} \dot{\lambda}^T(\tau)\mathbf{x}(\tau)\mathrm{d}\tau & (191) \\ &= \lambda^T(\tau_f)\mathbf{x}(\tau_f) - \lambda^T(\tau_0)\mathbf{x}(\tau_0) - \int_{\tau_0}^{\tau_f} \dot{\lambda}^T(\tau)\mathbf{x}(\tau)\mathrm{d}\tau. & (192) \end{aligned}$$

Hence,

$$J' = E(\mathbf{x}(\tau_f), \tau_f) - \lambda^T(\tau_f)\mathbf{x}(\tau_f) + \lambda^T(\tau_0)\mathbf{x}(\tau_0) + \int_{\tau_0}^{\tau_f} (H + \dot{\lambda}^T \mathbf{x})\mathrm{d}\tau. \tag{193}$$

The first variation of J' is given by

$$\begin{aligned} \delta J' &= \frac{\partial E}{\partial \mathbf{x}(\tau_f)}\mathrm{d}\mathbf{x}(\tau_f) + \frac{\partial E}{\partial \tau_f}\delta\tau_f - \lambda^T(\tau_f)\delta\mathbf{x}(\tau_f) & (194) \\ &\quad + \int_{\tau_0}^{\tau_f} (H_x \delta\mathbf{x} + H_u \delta\mathbf{u} + \dot{\lambda}^T \delta\mathbf{x})\mathrm{d}\tau + \int_{\tau_f}^{\tau_f + \delta\tau_f} L\mathrm{d}\tau, & (195) \end{aligned}$$

where

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{x}(\tau_f)} &= \left[\frac{\partial E}{\partial x_1(\tau_f)}, \dots, \frac{\partial E}{\partial x_{N_x}(\tau_f)}\right], && (196)\\
\mathrm{d}\mathbf{x}(\tau_f) &= \delta\mathbf{x}(\tau_f) + \dot{\mathbf{x}}(\tau_f)\delta\tau_f = \delta\mathbf{x}(\tau_f) + \mathbf{f}|_{\tau=\tau_f}\delta\tau_f, && (197)\\
H_x &= \left[\frac{\partial H}{\partial x_1}, \dots, \frac{\partial H}{\partial x_{N_x}}\right]^T, && (198)\\
H_u &= \left[\frac{\partial H}{\partial u_1}, \dots, \frac{\partial H}{\partial u_{N_u}}\right]^T. && (199)
\end{aligned}$$

Since

$$\int_{\tau_f}^{\tau_f+\delta\tau_f} L\mathrm{d}\tau = L|_{\tau=\tau_f}\delta\tau_f, \tag{200}$$

therefore,

$$\delta J' = \left[\frac{\partial E}{\partial \mathbf{x}(\tau_f)} - \lambda^T(\tau_f)\right]\delta\mathbf{x}(\tau_f) + \left[\frac{\partial E}{\partial \tau_f} + \frac{\partial E}{\partial \mathbf{x}(\tau_f)}\mathbf{f}|_{\tau=\tau_f} + L|_{\tau=\tau_f}\right]\delta\tau_f + \int_{\tau_0}^{\tau_f}((H_x + \dot{\lambda}^T)\delta\mathbf{x} + H_u\delta\mathbf{u})\mathrm{d}t. \tag{201}$$

Recall that a necessary condition for a minimum is that the first variation of J' be zero, that is,

$$\delta J' = 0. \tag{202}$$

Hence, the necessary conditions for optimality are as follows:

$$\begin{aligned}
\dot{\lambda}^T &= -H_x, && (203)\\
H_u &= 0 && (204)\\
\lambda^T(t_f) &= \frac{\partial E}{\partial x(t_f)}, && (205)
\end{aligned}$$

and

$$\frac{\partial E}{\partial \tau_f} + \frac{\partial E}{\partial \mathbf{x}(\tau_f)}f|_{\tau=\tau_f} + L|_{\tau=\tau_f} = 0. \tag{206}$$

Using (205), equation (206) can be written as

$$\frac{\partial E}{\partial \tau_f} + \lambda^T(t_f)f|_{\tau=\tau_f} + L|_{\tau=\tau_f} = 0, \tag{207}$$

Equations (203)-(205) are known as *Euler-Lagrange equations* and (207) is known as the *transversality condition.*

Hence, problem P reduces to the dual problem of determining $\mathbf{x}$, $\mathbf{u}$, λ, ν, and μ from the Euler-Lagrange equations (203)-(205), the transversality condition (207), the state dynamics,

$$\dot{\mathbf{x}}(\tau) = f(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau), \tag{208}$$

and the boundary conditions,

$$\begin{aligned} \mathbf{x}(\tau_0) &= \mathbf{x}_0, & (209) \\ e(\mathbf{x}(\tau_f), \tau_f) &= 0. & (210) \end{aligned}$$

As noted earlier, it is very difficult to find an analytic solution to the above mentioned optimal control problems P or P^λ, therefore the problems must be solved numerically. For this reason, the optimal control problems P or P^λ must be discretized to an NLP problem using certain kind of discretization, say for example, Runge-Kutta (RK) discretizations (discussed later in Sections 6.3 and 6.4). A discretization method is said to be *direct* if it refers to the discretization of problem P and *indirect* if it refers to the discretization of problem P^λ.

The indirect methods, as seen from above, require one to solve the necessary optimality conditions stated in terms of the adjoint differential equations and the associated transversality conditions, which for complicated nonlinear dynamics can be intimidating. On the other hand, direct methods are simply based on discretizing the states and the controls at a set of nodes, transforming the optimal control problem into an NLP problem. Moreover, direct methods tend to be more robust to the initial guesses, hence they converge more easily. Therefore, in this work, we discretize problem P directly without finding any analytic expressions for the necessary conditions using RK discretizations as described in Section 6.3.

Before we transcribe the optimal control problem into an NLP problem, we give a very brief introduction to nonlinear programming. For further reading, the reader is referred to a very nice text on nonlinear programming by Bazaraa et al. [9].

6.2 Introduction to Nonlinear Programming

A NLP problem is to minimize $f(\mathbf{x})$ subject to

$$\mathbf{g}(\mathbf{x}) \leq 0, \tag{211}$$

$$\mathbf{h}(\mathbf{x}) = 0, \tag{212}$$

where $\mathbf{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^\ell$. The above problem should be solved for the values of the variables $x_1, \ldots, x_n$ that satisfy the constratints (211), (212) and meanwhile minimize the function f.

The function f is usually called the *objective function*, or the *criterion function*. The constraints (211) are called *inequality constraints* and the constraints (212) are called the *equality constraints*. A vector $\mathbf{x}$ satisfying all the constraints (211), (212) is called a *feasible solution* to the problem. The collection of all such solutions forms the *feasible region*. The nonlinear programming (NLP) problem, then, is to find a feasible point $\bar{\mathbf{x}}$ such that $f(\mathbf{x}) \geq f(\bar{\mathbf{x}})$ for each feasible point $\mathbf{x}$. Such a point $\bar{\mathbf{x}}$ is called an *optimal solution* to the problem.

Next, we give the necessary conditions for $\bar{\mathbf{x}}$ to be an optimal solution to the above mentioned NLP problem.

Theorem 3 (Karush-Kuhn-Tucker (KKT) Necessary Conditions [9])**.** *For the above stated problem let $\bar{\mathbf{x}}$ be a feasible solution, and let $I = \{i : g_i(\bar{x}) = 0\}$. Suppose that f and g_i for $i \in I$ are differentiable at $\bar{\mathbf{x}}$, and suppose that each g_i for $i \notin I$ is continuous at $\bar{\mathbf{x}}$, and that each h_i for $i = 1, \ldots, \ell$ is continuously differentiable at $\bar{\mathbf{x}}$. Further, suppose that* $\mathrm{grad}(g_i)(\bar{\mathbf{x}})$ *for $i \in I$ and* $\mathrm{grad}(h_i)(\bar{\mathbf{x}})$ *for $i = 1, \ldots, \ell$ are linearly independent. If $\bar{\mathbf{x}}$ solves the problem locally, then unique scalars $\tilde{\mu}_i$ for $i \in I$ and $\tilde{\lambda}_i$ for $i = 1, \ldots, \ell$ exist such that*

$$\mathrm{grad}(f)(\bar{\mathbf{x}}) + \sum_{i \in I} \tilde{\mu}_i \mathrm{grad}(g_i)(\bar{\mathbf{x}}) + \sum_{i=1}^{\ell} \tilde{\lambda}_i \mathrm{grad}(h_i)(\bar{\mathbf{x}}) = 0, \tag{213}$$

$$\tilde{\mu_i} \geq 0, \qquad i \in I, \tag{214}$$

where $\mathrm{grad}(\cdot)$ *denotes the gradient of the function in the parantheses. In addition to the above assumptions, if each g_i for $i \notin I$ is also differentiable at $\bar{\mathbf{x}}$, then the KKT conditions* *can be written in the following equivalent form:*

$$\mathrm{grad}(f)(\bar{\mathbf{x}}) + \sum_{i=1}^{m} \tilde{\mu}_i \mathrm{grad}(g_i)(\bar{\mathbf{x}}) + \sum_{i=1}^{\ell} \tilde{\lambda}_i \mathrm{grad}(h_i)\bar{\mathbf{x}}) = 0, \tag{215}$$

$$\tilde{\mu}_i g_i(\bar{\mathbf{x}}) = 0, \qquad i = 1, \dots, m, \tag{216}$$

$$\tilde{\mu_i} \geq 0, \qquad i = 1, \dots, m. \tag{217}$$

In the next section, we transcribe the optimal control problem P into an NLP problem.

6.3 NLP Formulation: Discretizations on Dyadic Grids

All discretizations of the state dynamics, constraints and performance index in this chapter will be performed on (nonuniform) grids induced by dyadic grids (103) and (104):

$$\mathcal{V}_j = \{t_{j,k} \in [0,1] : t_{j,k} = k/2^j,\ 0 \leq k \leq 2^j\}, \qquad J_{\min} \leq j \leq J_{\max}, \tag{218}$$

$$\mathcal{W}_j = \{\hat{t}_{j,k} \in [0,1] : \hat{t}_{j,k} = (2k+1)/2^{j+1},\ 0 \leq k \leq 2^j - 1\}, \qquad J_{\min} \leq j \leq J_{\max} - 1. \tag{219}$$

For simplicity, we denote $\mathbf{x}$ and $\mathbf{u}$ evaluated at $t_{j,k}$ by $\mathbf{x}_{j,k}$ and $\mathbf{u}_{j,k}$ respectively. Using the transformation

$$\tau = t\,\Delta\tau + \tau_0, \tag{220}$$

where $\Delta\tau = \tau_f - \tau_0$ we can express the trajectory optimization problem stated in Section 6.1 on the unit interval $t \in [0,1]$ in terms of the new independent variable t. Hence, the original trajectory optimization problem reduces to the minimization of the following cost functional

$$J = e(\mathbf{x}(1), \tau_f) + \Delta\tau \int_0^1 L(\mathbf{x}(t), \mathbf{u}(t), t)\mathrm{d}t, \tag{221}$$

subject to the state dynamics

$$\frac{1}{\Delta\tau}\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t], \tag{222}$$

where $\mathbf{x} : [0,1] \to \mathbb{R}^{N_x}$, $\mathbf{u} : [0,1] \to \mathbb{R}^{N_u}$, the boundary conditions

$$\mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{e}_f(\mathbf{x}(1), \tau_f) = 0, \tag{223}$$

and constraints

$$\mathbf{C}_{\mathrm{u}}(\mathbf{u}(\tau)) \leq 0, \quad \mathbf{C}_{\mathrm{x}}(\mathbf{x}(\tau)) \leq 0, \quad \mathbf{C}_{\mathrm{xu}}(\mathbf{x}(\tau), \mathbf{u}(\tau)) \leq 0. \tag{224}$$

We convert the above mentioned optimal control problem into an NLP problem using a Runge-Kutta (RK) discretization. To this end, let a nonuniform grid of the form

$$\begin{aligned}\mathsf{G} = \{t_{j_i,k_i} : t_{j_i,k_i} \in [0,1],\ 0 \le k_i \le 2^{j_i},\ J_{\min} \le j_i \le J_{\max},\ \text{for } i = 0,\dots,N,\\ \text{and } t_{j_i,k_i} < t_{j_{i+1},k_{i+1}},\ \text{for } i = 0,\dots,N-1\}.\end{aligned} \tag{225}$$

Then a q-stage RK method for discretizing Eq. (222) is given by [17, 18]

$$\mathbf{x}_{j_{i+1},k_{i+1}} = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i}\Delta\tau \sum_{\ell=1}^{q} \beta^{\ell}\mathbf{f}^{\ell}_{j_i,k_i}, \tag{226}$$

where $\mathbf{f}^{\ell}_{j_i,k_i} = \mathbf{f}(\mathbf{y}^{\ell}_{j_i,k_i}, \mathbf{u}^{\ell}_{j_i,k_i}, t^{\ell}_{j_i,k_i})$, $\mathbf{y}^{\ell}_{j_i,k_i}$, $\mathbf{u}^{\ell}_{j_i,k_i}$, $t^{\ell}_{j_i,k_i}$ are the intermediate state, control, and time variables on the interval $[t_{j_i,k_i}, t_{j_{i+1},k_{i+1}}]$, given by

$$\mathbf{y}^{\ell}_{j_i,k_i} = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i}\Delta\tau \sum_{m=1}^{q} \alpha^{\ell,m}\mathbf{f}^{m}_{j_i,k_i}, \tag{227}$$

where $h_{j_i,k_i} = t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}$, $t^{\ell}_{j_i,k_i} = t_{j_i,k_i} + h_{j_i,k_i}\rho^{\ell}$, $\mathbf{u}^{\ell}_{j_i,k_i} = \mathbf{u}(t^{\ell}_{j_i,k_i})$, for $1 \le \ell \le q$, and q is referred to as the *stage*. In these expressions $\rho^{\ell}, \beta^{\ell}, \alpha^{\ell,m}$ are known constants with $0 \le \rho^1 \le \rho^2 \le \cdots \le 1$. The scheme is explicit if $\alpha^{\ell,m} = 0$ for $m \ge \ell$ and implicit otherwise. The coefficients $\rho^{\ell}, \beta^{\ell}, \alpha^{\ell,m}$ can be written in a convenient way using the Butcher diagram [35] as shown in Figure 24. Some common examples of q-stage RK methods are the trapezoidal method ($q = 2$), the Hermite-Simpson method ($q = 3$), and the classical fourth-order RK method ($q = 4$) [17, 18, 35].

$$\begin{array}{c|ccc} \rho^1 & \alpha^{11} & \dots & \alpha^{1q} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^q & \alpha^{q1} & \dots & \alpha^{qq} \\ \hline & \beta^1 & \dots & \beta^q \end{array}$$

Figure 24: Butcher diagram.

Using Eq. (226), the defects of discretization are given by

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - h_{j_i,k_i}\Delta\tau \sum_{\ell=1}^{q} \beta^{\ell}\mathbf{f}^{\ell}_{j_i,k_i}, \tag{228}$$

for $i = 0,\dots,N-1$. For discretizing the cost functional (221), we introduce a new state z such that

$$\dot{z}(t) = \Delta\tau L(\mathbf{x}(t), \mathbf{u}(t), t)\mathrm{d}t, \qquad z(0) = 0. \tag{229}$$

Using a q-stage RK method for discretizing Eq. (229) yields

$$z_{j_{i+1},k_{i+1}} = z_{j_i,k_i} + h_{j_i,k_i}\Delta\tau \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i}, \tag{230}$$

where $L^\ell_{j_i,k_i} = L(\mathbf{y}^\ell_{j_i,k_i}, \mathbf{u}^\ell_{j_i,k_i}, t^\ell_{j_i,k_i})$, $i = 0,\dots,N-1$. Hence, we have

$$z_{j_N,k_N} = z_{j_0,k_0} + \Delta\tau \sum_{i=0}^{N-1} h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i}. \tag{231}$$

Since $z(0) = z_{j_0,k_0} = 0$, the cost functional (221) in discretized form can be written as follows

$$J = e(\mathbf{x}_{j_N,k_N}, \tau_f) + \Delta\tau \sum_{i=0}^{N-1} \left(h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i} \right). \tag{232}$$

Let us now define the following sets

$$\mathbf{X} = \{\mathbf{x}_{j_0,k_0}, \dots, \mathbf{x}_{j_N,k_N}\},$$

$$\mathbf{U} = \{\mathbf{u}_{j_0,k_0}, \dots, \mathbf{u}_{j_N,k_N}\},$$

$$\tilde{\mathsf{G}} = \{t^\ell_{j_i,k_i} \in [0,1] : t^\ell_{j_i,k_i} \notin \mathsf{G},\ 0 \le i < N,\ 1 \le \ell \le q\},$$

$$\tilde{\mathbf{X}} = \{\mathbf{y}^\ell_{j_i,k_i} : t^\ell_{j_i,k_i} \in \tilde{\mathsf{G}}\},$$

$$\tilde{\mathbf{U}} = \{\mathbf{u}^\ell_{j_i,k_i} : t^\ell_{j_i,k_i} \in \tilde{\mathsf{G}}\}.$$

As a result of the discretization, the optimal control problem reduces to the NLP problem of finding the variables $\mathbf{X},\ \mathbf{U},\ \tilde{\mathbf{U}},\ \tau_f$, that minimize

$$J = e(\mathbf{x}_{j_N,k_N}, \tau_f) + \Delta\tau \sum_{i=0}^{N-1} \left(h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i} \right), \tag{233}$$

subject to the following constraints

$$\zeta_i = 0, \quad i = 0,\dots,N-1, \tag{234}$$

$$\mathbf{x}_{j_0,k_0} = \mathbf{x}_0, \tag{235}$$

$$\mathbf{e}_f(\mathbf{x}_{j_N,k_N}, \tau_f) = 0, \tag{236}$$

$$\mathbf{C}_{\mathrm{u}}(\mathbf{U}, \tilde{\mathbf{U}}) \le 0, \tag{237}$$

$$\mathbf{C}_{\mathrm{x}}(\mathbf{X}, \tilde{\mathbf{X}}) \le 0, \tag{238}$$

$$\mathbf{C}_{\mathrm{xu}}(\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{U}, \tilde{\mathbf{U}}) \le 0. \tag{239}$$

Remark 6. It is well known [64, 49] that RK discretizations for optimal control problems need to satisfy additional assumptions in order to obtain consistent approximations. Henceforth, we will therefore assume that the following conditions hold:

1. If the optimal control problem does not have any constraints, or if the optimal control problem has only pure control constraints then by RK discretizations we mean RK discretizations that satisfy the conditions in Ref. [64].
2. Alternatively, if the optimal control problem has only pure control constraints, the coefficients of the RK scheme satisfy the conditions given in Ref. [49] or Ref. [64].
3. If the optimal control problem has state or mixed state/control constraints, then by RK discretizations we mean either Euler, Trapezoidal, or Hermite-Simpson discretization.

The restriction to the above mentioned schemes stems from the fact that the convergence of these schemes for optimal control problems has been demonstrated in the literature [64, 49, 48, 19, 18]. Nonetheless, we point out that the proposed mesh refinement approach will work with any RK discretization for which the convergence for the optimal control problems can be shown, using either uniform or non-uniform meshes.

In the next section, we give examples of the RK discretizations used in this work.

6.4 Examples of Runge-Kutta Discretization

Four common examples of q-stage RK methods are Euler method ($q = 1$), trapezoidal method ($q = 2$), Hermite-Simpson method ($q = 3$), and classical fourth-order RK method ($q = 4$). The Euler discretization is first-order accurate, whereas the trapezoidal discretization is second-order accurate, and Hermit-Simpson and classical RK discretization are both fourth-order accurate.

6.4.1 Euler Method

An explicit Euler method is a 1-stage RK scheme with the following parameters

The defects of discretization for an explicit Euler scheme are as follows

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - h_{j_i,k_i}\Delta\tau\mathbf{f}(\mathbf{x}_{j_i,k_i}, \mathbf{u}_{j_i,k_i}, t_{j_i,k_i}), \tag{240}$$

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

for $i = 1, \ldots, N-1$.

6.4.2 Trapezoidal Method

Trapezoidal method is a 2-stage implicit RK scheme with the following parameters

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

The defects for the trapezoidal discretization are given by

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - \Delta\tau \frac{h_{j_i,k_i}}{2}(\mathbf{f}_{j_i,k_i} + \mathbf{f}_{j_{i+1},k_{i+1}}), \tag{241}$$

where

$$\mathbf{f}_{j_i,k_i} = \mathbf{f}(\mathbf{x}_{j_i,k_i}, \mathbf{u}_{j_i,k_i}, t_{j_i,k_i}),$$

for $i = 1, \ldots, N-1$.

6.4.3 Hermite-Simpson Method

Let us consider a 3-stage RK scheme with the following parameters

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 5/24 & 1/3 & -1/24 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

It has been indicated in [35] that this scheme is equivalent to the implicit Hermite-Simpson (HS) scheme (Appendix A.3), the defects of discretization for which are given by

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - \Delta\tau \frac{h_{j_i,k_i}}{6}[\mathbf{f}_{j_i,k_i} + 4\mathbf{f}_{j_{i+1/2},k_{i+1/2}} + \mathbf{f}_{j_{i+1},k_{i+1}}], \tag{242}$$

where

$$\mathbf{f}_{j_i,k_i} = \mathbf{f}(\mathbf{x}_{j_i,k_i}, \mathbf{u}_{j_i,k_i}, t_{j_i,k_i}),$$

$$\mathbf{f}_{j_{i+1/2},k_{i+1/2}} = \mathbf{f}(\mathbf{x}_{j_{i+1/2},k_{i+1/2}}, \mathbf{u}_{j_{i+1/2},k_{i+1/2}}, t_{j_{i+1/2},k_{i+1/2}}),$$

$$\mathbf{x}_{j_{i+1/2},k_{i+1/2}} = \frac{1}{2}[\mathbf{x}_{j_i,k_i} + \mathbf{x}_{j_{i+1},k_{i+1}}] + \Delta\tau\frac{h_{j_i,k_i}}{8}[\mathbf{f}_{j_i,k_i} - \mathbf{f}_{j_{i+1},k_{i+1}}],$$
$$t_{j_{i+1/2},k_{i+1/2}} = \frac{t_{j_i,k_i} + t_{j_{i+1},k_{i+1}}}{2}, \quad \mathbf{u}_{j_{i+1/2},k_{i+1/2}} = \mathbf{u}(t_{j_{i+1/2},k_{i+1/2}}),$$

for $i = 1, \dots, N-1$.

6.4.4 Classical Runge-Kutta Method

The Butcher diagram for the fourth-order explicit RK scheme ($q = 4$) is

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Hence, the defects of discretization for a fourth-order RK scheme are as follows

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - \frac{1}{6}(\mathbf{r}_1 + 2\mathbf{r}_2 + 2\mathbf{r}_3 + \mathbf{r}_4), \tag{243}$$

where

$$\begin{aligned} \mathbf{r}_1 &= h_{j_i,k_i}\Delta\tau\mathbf{f}(\mathbf{x}_{j_i,k_i}, \mathbf{u}_{j_i,k_i}, t_{j_i,k_i}), \\ \mathbf{r}_2 &= h_{j_i,k_i}\Delta\tau\mathbf{f}(\mathbf{x}_{j_i,k_i} + \frac{1}{2}\mathbf{r}_1, \mathbf{u}_{j_{i+1/2},k_{i+1/2}}, t_{j_{i+1/2},k_{i+1/2}}), \\ \mathbf{r}_3 &= h_{j_i,k_i}\Delta\tau\mathbf{f}(\mathbf{x}_{j_i,k_i} + \frac{1}{2}\mathbf{r}_2, \mathbf{u}_{j_{i+1/2},k_{i+1/2}}, t_{j_{i+1/2},k_{i+1/2}}), \\ \mathbf{r}_4 &= h_{j_i,k_i}\Delta\tau\mathbf{f}(\mathbf{x}_{j_i,k_i} + \mathbf{r}_3, \mathbf{u}_{j_{i+1},k_{i+1}}, t_{j_{i+1},k_{i+1}}), \end{aligned}$$
$$t_{j_{i+1/2},k_{i+1/2}} = \frac{t_{j_i,k_i} + t_{j_{i+1},k_{i+1}}}{2}, \quad \mathbf{u}_{j_{i+1/2},k_{i+1/2}} = \mathbf{u}(t_{j_{i+1/2},k_{i+1/2}}),$$

for $i = 1, \dots, N-1$.

Since the trajectory optimization problem can have discontinuities and switchings in the states and the controls, one way to accurately capture these discontinuities and switchings in the solution is to solve the NLP problem on a very fine mesh. However, this will require a lot of computational resources in terms of both CPU time and memory. Therefore, in order to accurately capture the irregularities in the solution and alleviate these problems, we will only refine the mesh locally in the region of the irregularity using the multiresolution-based mesh refinement algorithm described in Chapter 4.

We are now ready to present the proposed multiresolution-based trajectory optimization algorithm.

6.5 *Multiresolution Trajectory Optimization*

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in Eqs. (218) and (219). Let G be a nonuniform grid as given in (67)), then by $\mathcal{I}^p(\cdot;\mathcal{T}_{\mathsf{G}}(\cdot))$ we denote the p-th order essentially nonoscillatory (ENO) interpolation (see Section 4.3.2) of $\mathcal{U} = \{g_{j,k} : t_{j,k} \in \mathcal{T}_{\mathsf{G}}(t)\}$, where $\mathcal{T}_{\mathsf{G}}(t) = \{t_{j_m,k_m}\}_{m=i}^{i+p} \subseteq \mathsf{G}$, $0 \leq i \leq N-p-1$.

To proceed with the algorithm, we first choose the minimum resolution level $J_{\min}$ based on the minimum time step required to achieve the desired accuracy in the regions of the solution where no constraints are active[1], the threshold ϵ, which should be at least of the order of $h_{J_{\min}}$, where $h_{J_{\min}} = 1/2^{J_{\min}}$ (the significance of ϵ and reason for such a choice of ϵ which will be clear shortly), and pick the maximum resolution level $J_{\max}$. The proposed MTOA involves the following steps. First, we transcribe the continuous trajectory optimization problem into an NLP problem using a q-stage RK discretization as described in the previous section. We use trapezoidal discretization for the first iteration and switch to a high-order discretization for subsequent iterations. Next, we set $\texttt{iter} = \texttt{1}$, initialize $\mathsf{Grid}_{\text{iter}} = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables. Let us denote the set of initial guesses by $\mathcal{X}_{\text{iter}}$. The proposed Multiresolution Trajectory Optimization Algorithm (MTOA) then proceeds as follows:

Multiresolution Trajectory Optimization Algorithm (MTOA)

1. Solve the NLP problem on $\mathsf{Grid}_{\text{iter}}$ with the initial guess $\mathcal{X}_{\text{iter}}$. If $\mathsf{Grid}_{\text{iter}}$ has points from the level $\mathcal{W}_{J_{\max}-1}$, terminate.

2. **Mesh refinement**.

(a)-i. If the problem has either pure state constraints or mixed constraints on the states

[1]The minimum time step required to achieve a desired accuracy in the regions of the solution where no constraints are active can be calculated using the well-known error estimation formulas for RK schemes [64, 19, 65, 66].

and controls, set $\Phi_{\text{iter}} = \{\mathbf{x}_{j,k}, \mathbf{u}_{j,k} : t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$ and $N_r = N_x + N_u$.

(a)-ii. If the optimal control problem does not have any constraints, or if only pure control constraints are present, set $\Phi_{\text{iter}} = \{\mathbf{u}_{j,k} : t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$ and $N_r = N_u$.

(a)-iii. In case no controls are present in the problem, set $\Phi_{\text{iter}} = \{\mathbf{x}_{j,k} : t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$ and $N_r = N_x$.

In the following, let Φ_{iter} denote the set constructed in Step (a) of the algorithm, that is, let $\Phi_{\text{iter}} = \{\phi_\ell(t_{j,k}) : \ell = 1, \ldots, N_r,\ t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$.

(b) Initialize an intermediate grid $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}-1}$, with function values

$$\Phi_{\text{int}} = \{\phi_\ell(t_{J_{\min},k}) \in \Phi_{\text{iter}},\ 0 \le k \le 2^{J_{\min}},\ \ell = 1, \ldots, N_r\}, \qquad (244)$$

and set $j = J_{\min} - 1$.

i. Find the points that belong to the intersection of $\mathcal{W}_j$ and $\mathsf{Grid}_{\text{iter}}$

$$\hat{T}_j = \{\hat{t}_{j,k_i} : \hat{t}_{j,k_i} \in \mathcal{W}_j \cap \mathsf{Grid}_{\text{iter}},\ \text{for } i = 1, \ldots, N_{\hat{t}},\ 1 \le N_{\hat{t}} \le 2^j - 1\}. \quad (245)$$

If $\hat{T}_j$ is empty go to Step 2(c), otherwise go to the next step.

ii. Set $i = 1$.

A. Compute the interpolated function values at $\hat{t}_{j,k_i} \in \hat{T}_j$,

$$\hat{\phi}_\ell(\hat{t}_{j,k_i}) = \mathcal{I}^p(\hat{t}_{j,k_i}, \mathcal{T}_{\mathsf{Grid}_{\text{int}}}(\hat{t}_{j,k_i})),$$

where $\hat{\phi}_\ell$ is the ℓth element of $\hat{\phi}$, for $\ell = 1, \ldots, N_r$.

B. Calculate the interpolative error coefficient d_{j,k_i} at the point $\hat{t}_{j,k_i}$[2]

$$d_{j,k_i}(\phi) = \max_{\ell=1,\ldots,N_r} d_{j,k_i}(\phi_\ell) = \max_{\ell=1,\ldots,N_r} |\phi_\ell(\hat{t}_{j,k_i}) - \hat{\phi}_\ell(\hat{t}_{j,k_i})|. \qquad (246)$$

If the value of d_{j,k_i} is below the threshold ϵ, then reject $\hat{t}_{j,k_i}$ and goto Step 2(b)iiE, otherwise add $\hat{t}_{j,k_i}$ to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and move on to the next step.

[2]Note that $\phi_\ell(\hat{t}_{j,k}) \in \Phi_{\text{iter}}$ for all $\hat{t}_{j,k} \in \hat{T}_j$ and $\ell = 1, \ldots, N_r$.

C. Add to $\mathsf{Grid}_{\mathrm{int}}$ points belonging to the set $(\mathcal{V}_{\hat{J}} \cap [t_{j,k_i}, t_{j,k_i+1}]) \setminus \mathsf{Grid}_{\mathrm{int}}$, where $\hat{J} = \min\{j + \hat{j}, J_{\max}\}$, $\hat{j} = 2$ if $\texttt{iter} = \texttt{1}$ and $\hat{j} \geq 2$ if $\texttt{iter} > \texttt{1}$. Here $\hat{j}$ is the number of finer levels from which the points be added to the grid for refinement. In particular, we add to the intermediate grid $\mathsf{Grid}_{\mathrm{int}}$ the points $\{t_{\hat{J},k} : 2^{\hat{J}-j}k_i \leq k \leq 2^{\hat{J}-j}(k_i + 1)\} \setminus \mathsf{Grid}_{\mathrm{int}}$.

D. Add the function values at all the newly added points to Φ_{int}. If the function value at any of the newly added points is not known, interpolate the function value at that point from the points in $\mathsf{Grid}_{\mathrm{iter}}$ and their function values in $\mathcal{X}_{\mathrm{iter}}$ using $\mathcal{I}^p(\cdot, \mathcal{T}_{\mathsf{Grid}_{\mathrm{iter}}}(\cdot))$.

E. Increment i by 1. If $i \leq N_{\hat{t}}$ goto Step 2(b)iiA, otherwise move on to the next step.

iii. Set $j = j + 1$. If $j < J_{\max}$ go to Step 2(b)i, otherwise move on to the next step.

(c) Terminate the mesh refinement algorithm. The final nonuniform grid is $\mathsf{Grid}_{\mathrm{new}} = \mathsf{Grid}_{\mathrm{int}}$ and the corresponding function values are in the set $\Phi_{\mathrm{new}} = \Phi_{\mathrm{int}}$.

3. Set $\texttt{iter} = \texttt{iter} + \texttt{1}$. If the number of points and the level of resolution remain the same after the mesh refinement procedure, terminate. Otherwise interpolate the NLP solution found in Step 1 on the new mesh $\mathsf{Grid}_{\mathrm{new}}$ (which will be the new initial guess $\mathcal{X}_{\mathrm{iter}}$), reassign the set $\mathsf{Grid}_{\mathrm{iter}}$ to $\mathsf{Grid}_{\mathrm{new}}$, and go to Step 1.

The order of the interpolating polynomial p can be taken to be one less than the order of the RK discretization of the differential equations. This choice of p is dictated by the error analysis given in the next section, which considers the case with no constraints. It is reminded that under the presence of constraints, the order of the RK discretization for optimal control problems may be less than the order of the RK discretization used for the differential equations [64]. The subsequent analysis, albeit heuristic, elucidates the motivation behind the proposed approach and the previous choice of p. Although a more rigorous analysis is required to justify the recommended choice for the order of the interpolating polynomials (hence the order of the RK discretization as well), nonetheless, in all numerical examples we considered, choosing the interpolating polynomial according to the previous criterion turned out to be adequate, irrespective of the presence (or not) of the constraints.

6.6 Rationale of Proposed Multiresolution Scheme

In this section we outline the main idea behind the multiresolution mesh refinement algorithm. In the process, we also provide rough estimates on the error one expects to obtain by following the proposed approach. To keep the notation as simple as possible, the subsequent discussion will be restricted to the case of a scalar-valued control function u. Furthermore, we will consider a problem without state and control constraints, so that the refinement algorithm is performed based on the (scalar-valued) control histories (case 2(a)-ii of MTOA with $N_u = N_r = 1$).

The key idea behind the proposed mesh refinement algorithm is based on the fact that the interpolative error coefficient in Step 2(b)ii-B of the MTOA, and for a sufficiently fine grid, provides a good measure of the local smoothness of the function u. To see why this is so, consider a function u which, at $t = \bar{t}$, has $\nu \geq -1$ continuous derivatives[3], but it has a jump discontinuity in its $(\nu + 1)$th derivative. Locally around any point $t \neq \bar{t}$ the function u can be approximated accurately by a polynomial, say $\hat{u}$, of degree ν. Furthermore, in the neighborhood of $\bar{t}$, any interpolating polynomial of degree at least $\nu+1$ will induce an error that is proportional of the jump discontinuity of $u^{(\nu+1)}$.

The proposed algorithm uses the information of the local interpolation error in (246) to locally refine the grid, if necessary. In particular, at the locations when the solution is smooth (hence it can be accurately interpolated by neighboring points) no further refinement is performed. At those locations where the function is not smooth, grid points are added to reduce the interpolation error below a certain threshold.

To this end, let the final grid at a certain iteration step be given by the points $\mathsf{G} = \{t_0, t_1, \ldots, t_N\}$. For each point $t_i \in \mathsf{G}$, $(0 \leq i \leq N)$, let $\mathcal{T}_{\mathsf{G}}(t_i) = \{\tau_0^i, \tau_1^i, \ldots, \tau_p^i\} \in \mathsf{G}\backslash\{t_i\}$ with $\tau_0^i < \cdots < \tau_p^i$, be the stencil of $p + 1$ points that are used to interpolate the function

[3]This notation implies that for $\nu = -1$ the function is discontinuous.

u in the interval $[\tau_0^i, \tau_p^i]$, according to the discussion in the previous section. That is, let $\hat{u}$ be the unique polynomial of degree p, such that (see Appendix A.1.1)

$$\hat{u}(\tau_m^i) = u(\tau_m^i), \qquad 0 \le m \le p, \;\; 0 \le i \le N, \tag{247}$$

and

$$u(t) = \hat{u}(t) + u[\tau_0^i, \dots, \tau_p^i, t]\, \Pi_{m=0}^{p}(t - \tau_m^i), \qquad \tau_0^i \le t \le \tau_p^i. \tag{248}$$

Moreover, if u is sufficiently smooth (i.e., is continuously differentiable at least $\nu \ge p+1$ times) in the interval $[\tau_0^i, \tau_p^i]$ then

$$u[\tau_0^i, \dots, \tau_p^i, t] = \frac{u^{(p+1)}(\xi)}{(p+1)!}, \qquad \tau_0^i \le \xi \le \tau_p^i. \tag{249}$$

It then follows from (248) that

$$d_i(u) = |u(t_i) - \hat{u}(t_i)| \approx |u^{(p+1)}|(h_i)^{(p+1)}, \quad (\nu \ge p+1), \tag{250}$$

where $h_i = \max_{0 \le m \le p-1}(\tau_{m+1}^i - \tau_m^i)$. In (250) the notation "$\approx$" indicates a term of the same order of magnitude. Similarly, the notation "$\lesssim$" will be used to indicate a term dominated by an expression of a known order of magnitude.

If, on the other hand, u has a jump discontinuity in its $(\nu+1)$ derivative and $\nu < p+1$, then [51, 68]

$$u[\tau_0^i, \dots, \tau_p^i, t] \approx \frac{[[u^{(\nu+1)}]]}{(h_i)^{p-\nu}}, \tag{251}$$

where $[[u^{(\nu+1)}]]$ denotes the jump at the discontinuity of the $(\nu+1)$th derivative of u inside the interval $[\tau_0^i, \tau_p^i]$. It follows from (248) that, in this case, we have the estimate

$$d_i(u) = |u(t_i) - \hat{u}(t_i)| \approx [[u^{(\nu+1)}]](h_i)^{(\nu+1)}, \quad (\nu < p+1). \tag{252}$$

It has been shown in Ref. [64] that, under appropriate smoothness and coercivity hypotheses [64], and assuming that the solution $u^\star$ of the continuous optimal control problem (221)-(223) is at least $\nu = p-1$ continuous differentiable, the following estimate holds

$$\max_{0 \le i \le N} |\mathbf{x}_i - \mathbf{x}^\star(t_i)| + \max_{0 \le i \le N} |u_i - u^\star(t_i)| \lesssim h^{p+1} + h^p \int_0^1 \omega(u^{\star(p)}, [0,1]; t, h)\, \mathrm{d}t, \tag{253}$$

for sufficiently small $h = \max_{0\leq i\leq N}\{t_{i+1} - t_i\}$, and where $\omega(v, [a, b]; t, h)$ denotes the local modulus of continuity of the function v, defined by [122]

$$\omega(v, [a, b]; t, h) = \sup\{|v(\sigma_1) - v(\sigma_2)| : \sigma_1, \sigma_2 \in [t - h/2, t + h/2] \cap [a, b]\}. \tag{254}$$

In (253) it has been assumed that the optimal solution $(\mathbf{x}_i, u_i)$ of the discrete problem (233)-(236) is computed using a $p+1$-th order RK scheme satisfying the Hager conditions of Ref. [64].

The MTOA estimates the last term in (253) using the local interpolating error for the control. To see why this is true, re-write the last term in (253) as follows

$$\begin{aligned}\int_0^1 \omega(u^{\star(p)}, [0, 1]; t, h)\,\mathrm{d}t &= \sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \omega(u^{\star(p)}, [0, 1]; t, h)\,\mathrm{d}t \\ &= \sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \omega(u^{\star(p)}, [t_i', t_i'']; t, h)\,\mathrm{d}t\end{aligned} \tag{255}$$

where $t_i' = 3t_i/2 - t_{i+1}/2$ and $t_i'' = 3t_{i+1}/2 - t_i/2$. Using the definition of the modulus of continuity (254) and the estimate (251), we have that

$$\omega(u^{\star(p)}, [t_i', t_i'']; t, h) \leq \sup_{\sigma_1,\sigma_2\in[t_i',t_i'']} |u^{\star(p)}(\sigma_1) - u^{\star(p)}(\sigma_2)| \approx [[u^{\star(p)}]]. \tag{256}$$

It follows that

$$\int_{t_i}^{t_{i+1}} \omega(u^{\star(p)}, [t_i', t_i'']; t, h)\,\mathrm{d}t \lesssim h_i^{1-p} d_i(u^\star). \tag{257}$$

Since the MTOA ensures the bound $|d_i(u^\star)| \leq \epsilon$ we finally get the estimate (recall that $\sum_{i=0}^{N} h_i \approx 1$)

$$h^p \int_0^1 \omega(u^{\star(p)}, [0, 1]; t, h)\,\mathrm{d}t \lesssim \epsilon. \tag{258}$$

It follows that

$$\max_{0\leq i\leq N} |\mathbf{x}_i - \mathbf{x}^\star(t_i)| + \max_{0\leq i\leq N} |u_i - u^\star(t_i)| \lesssim h^{p+1} + \epsilon. \tag{259}$$

Given now the general grid in (225) it follows from (259) that if we chose $\epsilon \approx h_{J_{\min}}^{(p+1)}$, where $h_{J_{\min}} = t_{J_{\min},k+1} - t_{J_{\min},k} = 1/2^{J_{\min}}$, $0 \leq k \leq 2^{J_{\min}} - 1$, we get an estimate of the form

$$\max_{0\leq i\leq N} |\mathbf{x}_{j_i,k_i} - \mathbf{x}^\star(t_{j_i,k_i})| + \max_{0\leq i\leq N} |u_{j_i,k_i} - u^\star(t_{j_i,k_i})| \lesssim h_{J_{\min}}^{p+1}. \tag{260}$$

Remark 7. As pointed out by Hager [64], the error in the discrete controls u_i may be one or more orders larger that the error obtained if the control were computed by the minimization of the Hamiltonian and by using the discrete state/costate pair instead. Hence, ideally, the approximation order in the right-hand-side of (253) will be one or more order less that $p+1$ even if a $p+1$-th RK order is used. The interested reader may refer to Ref. [64] for further details in regards to this observation. Since here we are only interested in rough error estimates, the exact order of convergence for the discrete controls is immaterial for the overall analysis (for example, use a higher order RK-scheme if needed).

6.7 Numerical Examples

In this section we provide several examples to demonstrate the robustness and efficiency of the proposed approach for the solution of optimal control problems. For all cases, we have used SNOPT [61] to solve the resulting NLP problem (233)-(239). SNOPT is an NLP solver, which is based on sequential quadratic programming (SQP). All computations were performed in MATLAB on a Pentium IV machine with a 3 GHz processor and 2 GB of RAM. In all the examples below, and unless stated otherwise, a linear function has been used as an initial guess for the first iteration of MTOA.

Example 13

First, we consider the Moon landing problem, taken from Ref. [55]. The control problem is formulated as maximizing the final mass, and hence minimizing

$$J = -m(\tau_f). \tag{261}$$

The equations of motion are given by

$$\frac{\mathrm{d}h}{\mathrm{d}\tau} = v, \tag{262}$$

$$\frac{\mathrm{d}v}{\mathrm{d}\tau} = -g + \frac{T}{m}, \tag{263}$$

$$\frac{\mathrm{d}m}{\mathrm{d}\tau} = -\frac{T}{I_{\mathrm{sp}}g}, \tag{264}$$

where the state variables $h,\ v,\ m$ are altitude, velocity, and mass respectively. Control is provided by the thrust T, which is bounded by

$$0 \le T \le T_{\max}. \tag{265}$$

The final time τ_f is free. The other parameters in the problem are g, the gravity of the Moon, and I_{sp}, the specific impulse of the spacecraft engine. The normalized parameters for the problem were chosen the same as in [55]:

$$\frac{T_{\max}}{m_0 g} = 1.1, \quad \frac{I_{\text{sp}} g}{v_0} = 1, \quad \frac{h(0)}{h_0} = 0.5,$$
$$\frac{v(0)}{v_0} = -0.05, \quad \frac{m(0)}{m_0} = 1,$$

for any given set of initial conditions h_0, v_0, and m_0. Therefore, we have the following normalized initial conditions:

$$h(0) = 0.5, \quad v(0) = -0.05, \quad m(0) = 1.0. \tag{266}$$

For soft landing, we must have

$$\begin{aligned} h(0) &= 0.5, & (267)\\ v(0) &= -0.05, & (268)\\ m(0) &= 1.0. & (269) \end{aligned}$$

In addition, for a physical meaningful trajectory, we must have

$$m(\tau_f) > 0. \tag{270}$$

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 10$. The threshold used for this problem was $\epsilon = 10^{-4}$. We used the fourth-order explicit RK scheme ($q = 4$) for a high order discretization in MTOA. The algorithm terminated in 8 iterations and the overall CPU time taken by MTOA to solve this problem was 5.1 seconds. Because of the space constraints, we show the time history of thrust T along with the grid point distribution only for iterations $1,\ 3,\ 6,\ 7,$ and 8 (Figure 25 and Figure 26). The grid point distributions in Figure 25 and Figure 26 show that with each iteration the algorithm adds points at finer resolution levels, and as a result the solution is getting more and more accurate. Moreover, as the solution gets more and more accurate, the algorithm also removes points at the coarser levels from the region where the solution is getting smoother. The grid point distribution at iteration 8 (Figure 26(d)) again shows that the regions where the solution is smooth are well represented by the coarse resolution levels; the higher resolution levels are needed only near the switching points in the thrust T, thus illustrating the efficiency of the proposed algorithm. From Figure 26(c), we see that the algorithm was accurately able to capture the switching in the control using only two points. It should be noted that the algorithm used only 25 points out of 1025 points of the grid $\mathcal{V}_{10}$ for calculating the final solution. One should also discern that the algorithm used 25 points at iteration 6 whereas used 23 points at iteration 7. At iteration 7, the algorithm removed some points at the coarser resolution levels and added points at the finer resolution level $\mathcal{W}_8$. This clearly demonstrates that the proposed strategy uses only the grid points that are actually necessary to attain a given precision, and the algorithm is able to add and remove points when and where is needed. The time history of mass m along with the phase portrait of the velocity v vs. the altitude h for the last iteration are shown in Figure 27.

For comparison, we also solved the same problem using a fourth-order explicit RK scheme on a uniform grid with same number of nodes as used by MTOA at the final iteration, that is, on a uniform mesh with 25 nodes. The algorithm terminated in 2.1 seconds. Since, the switching in the control is not captured accurately (see Figure 28(a)) using a uniform mesh with 25 nodes, we gradually increased the number of nodes in the uniform mesh and resolved the problem using the same linear initial guess, until the CPU time taken by the algorithm was approximately equal to the CPU time taken by MTOA. We ended up using 46 nodes in the uniform mesh. The algorithm terminated in 5.1 seconds. The control found using a uniform mesh with 46 nodes is shown in Figure 28(b). Hence, we see that MTOA was able to capture the switching in the control with more accuracy for the same CPU time as for the uniform mesh case with 46 nodes.

Example 14

We first consider a simple minimum-energy problem with a second-order state variable inequality constraint, taken from Ref. [33]. Since the analytic solution for this problem is known, we can infer the *absolute* accuracy of the solution provided by the proposed MTOA.

The problem is to find the control $u(t)$ that minimizes the cost function

$$J = \frac{1}{2}\int_0^1 u^2(t)\,\mathrm{d}t, \tag{271}$$

subject to the dynamics

$$\dot{x} = v, \tag{272}$$

$$\dot{v} = u, \tag{273}$$

initial and final conditions

$$x(0) = x(1) = 0, \tag{274}$$

$$v(0) = -v(1) = 1, \tag{275}$$

and the path constraint

$$x(t) \leq 0.04. \tag{276}$$

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 10$. The threshold used was $\epsilon = 10^{-4}$. We used the implicit HS scheme for a high order discretization in MTOA. The algorithm terminated in 5 iterations. The time histories of the states x and v at the final iteration are shown in Figure 29. The time history of the control u along with the grid point distribution at the final iteration are shown in Figure 30. It should be noted that the proposed algorithm used only 61 points out of the maximum of 1025 points at the finest resolution grid $\mathcal{V}_{10}$. Since the analytic solution of this problem is known [33] the absolute error can be computed for all cases. The errors in the computed solution along with the number of grid points (N_{iter}) used by the algorithm at each iteration are shown in Table 12. As shown in Table 12, the numerical solution converges to the analytic solution and, with each iteration, the errors are decreasing roughly by an order of magnitude.

The overall CPU time taken by MTOA to solve this problem was 5.6 seconds. For comparison, we also solved the same problem using a Hermite-Simpson discretization on a uniform grid with the same number of points as in MTOA at the final iteration, that is, on a uniform mesh with 61 nodes. The algorithm terminated in 2.5 seconds with the errors shown in Table 13. Since the errors in the solution using a uniform mesh with 61 nodes

Table 12: Example 14: No. of grid points along with the error in the computed optimal cost at each iteration.

Iteration	N_{iter}	$\|x - x^\star\|_{L^\infty}$	$\|v - v^\star\|_{L^\infty}$	$\|u - u^\star\|_{L^\infty}$	$\|J - J^\star\|$
1	9	4.0×10^{-2}	1.5×10^{-1}	1.7×10^{0}	Failed
2	15	1.3×10^{-4}	2.1×10^{-3}	1.3×10^{-1}	5.7×10^{-3}
3	29	3.9×10^{-6}	5.9×10^{-5}	3.0×10^{-3}	4.6×10^{-4}
4	45	3.1×10^{-7}	1.4×10^{-5}	5.2×10^{-4}	6.6×10^{-6}
5	61	3.0×10^{-8}	1.6×10^{-6}	5.6×10^{-5}	3.3×10^{-8}

Table 13: Example 14: No. of grid points and error for uniform mesh.

N_1	$\|x - x^\star\|_{L^\infty}$	$\|v - v^\star\|_{L^\infty}$	$\|u - u^\star\|_{L^\infty}$	$\|J - J^\star\|$
61	3.7×10^{-6}	1.4×10^{-4}	1.4×10^{-1}	2.7×10^{-4}
131	1.7×10^{-6}	9.5×10^{-5}	2.7×10^{-2}	6.0×10^{-5}

are larger than the ones achieved using MTOA, we also gradually increased the number of nodes in the uniform mesh and resolved the problem using the same linear initial guess, until either the errors were of the same order of magnitude as the ones obtained using the MTOA or the CPU time taken by the algorithm was approximately equal to the CPU time taken by MTOA. This process ended up in a uniform mesh of 131 nodes. The algorithm terminated in 5.7 seconds and the final errors are shown in Table 13. These results show a typical trend we observed in all examples we tested, and which demonstrate the efficacy of the MTOA : higher accuracy for the same CPU time or a smaller number of grid points and CPU time for the same accuracy, compared to uniform grid implementations.

Example 15

Here we consider a problem derived from the control of a chemical reaction [22, 40]. The problem is to maximize the final amount of product y during a two-stage chemical reaction, $x \to y \to z$, by a proper choice of the rate coefficient $u(t)$. The amount of waste product z formed does not influence x and y, and since the magnitude of z is of no interest, we may consider only the reaction rates for x, y, which are given by

$$\dot{x} = -ux, \tag{277}$$

$$\dot{y} = ux - \rho u^k y, \tag{278}$$

where ρ, k are positive constants. For this example we consider the same parameters as in

[22, 40]

$$\rho = 2.5, \quad k = 1.5, \quad t_f = 2, \tag{279}$$

and initial conditions

$$x(0) = 1, \tag{280}$$

$$y(0) = 0.01. \tag{281}$$

The allowable control must lie within the range

$$0.1 \leq u(t) \leq u_{\max}. \tag{282}$$

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 6$ for three different choices of $u_{\max}$, namely, $u_{\max} = 0.5$, $u_{\max} = 0.4$, and $u_{\max} = 0.3$. The threshold used in the simulations was $\epsilon = 10^{-4}$. We used the implicit HS scheme for a high order discretization in MTOA. The algorithm terminated in four iterations for all cases. The time history of the states x, y and the control u, along with the grid point distribution for different values of $u_{\max}$ at the final iteration of MTOA are shown in Figure 31. The final states $x(2)$ and $y(2)$ (rounded off to five decimal places), the overall CPU time taken by MTOA to solve the problem, and the number of nodes used at the final iteration of MTOA (N_f) for the previous three values of $u_{\max}$, are summarized in Table 14. The values of $x(2)$ and $y(2)$ are the same as those reported in Ref. [40].

We also solved the same problem using the HS discretization on a uniform grid having the same number of points as used by MTOA at the final iteration, that is, on a uniform mesh with N_f nodes. The CPU times (t_{CPU}) used by the algorithm for all the cases are summarized in Table 15. The values for both $x(2)$ and $y(2)$ were accurate up to five decimal places for the case $u_{\max} = 0.4$. Since either of the two values $x(2)$ or $y(2)$ was not of the same accuracy for the remaining two cases, we resolved the problem for the cases $u_{\max} = 0.5$ and $u_{\max} = 0.2$ with a larger number of nodes in the uniform mesh (using again a linear initial guess). We repeated this process until the values for both the states at the final time coincided to five decimal places to the solution given in Table 14. The result was a uniform mesh of 55 and 20 nodes for the cases $u_{\max} = 0.5$ and $u_{\max} = 0.3$

respectively. These observations, along with the corresponding CPU times are reported in Table 15. The uniform mesh implementation required more points to obtain the same accuracy. The corresponding CPU times were comparable for this example for both uniform and non-uniform mesh implementations. The reader should be reminded however that the uniform grid solutions were obtained by calling SNOPT only once (assuming convergence was possible). Hence *per iteration* the CPU time for the MTOA is indeed smaller, as expected.

Table 14: Example 15: No. of nodes used by MTOA at the final iteration, overall CPU time taken by MTOA, and final states for three different values of $u_{\max}$.

$u_{\max}$	N_f	t_{CPU} (sec)	$x(2)$	$y(2)$
0.5	31	6.2	0.52222	0.30813
0.4	23	3.8	0.53051	0.30611
0.3	17	1.7	0.55765	0.30013

Table 15: Example 15: Uniform mesh.

$u_{\max}$	N	t_{CPU} (sec)	Error	N	t_{CPU} (sec)	Error
0.5	31	3	10^{-5}	55	6.9	10^{-6}
0.4	23	1.7	10^{-6}	-	-	-
0.3	17	1.0	10^{-5}	20	1.3	10^{-6}

Example 16

In this example we investigate the performance of MTOA to a "hyper-sensitive" problem, taken from Ref. [18]. As pointed out in Ref. [115, 18] this problem is extremely difficult to solve using indirect methods. The problem is to minimize

$$J = \int_0^{10000} (x^2(t) + u^2(t))\, \mathrm{d}t, \tag{283}$$

subject to

$$\dot{x} = -x^3 + u, \tag{284}$$

and

$$y(0) = 1, \tag{285}$$

$$y(10000) = 1.5. \tag{286}$$

We solved this problem on a grid with $J_{\min} = 4$ and $J_{\max} = 10$. The threshold used was $\epsilon = 10^{-4}$. We used the implicit HS scheme for a high order discretization in MTOA. MTOA terminated in 5 iterations and the overall CPU time taken by MTOA to solve this problem was 17.5 seconds. The final nonuniform grid (shown in Fig. 32(b)) included 53 nodes. The time history of the state x is shown in Figure 32(a).

For comparison, we also solved the same problem using a HS discretization on a uniform grid having the same number of nodes as used by MTOA at the final iteration, that is, on a uniform mesh with 53 nodes. The algorithm terminated after 43.7 seconds; the value of the optimal cost found was an order of magnitude larger than the optimal cost found using MTOA. These results show, again, the superiority of the MTOA over uniform grid implementations. For this example, the uniform grid implementation not only took more than twice the CPU time of MTOA, but also returned a solution that was far worse than the one obtained from MTOA.

Example 17

As our final example we consider the realistic problem of optimizing the re-entry trajectory of an Apollo-type vehicle [112]. This is a benchmark problem in trajectory optimization that is known to be very challenging owing to its sensitivity in terms of the initial guesses.

The equations of motion during the flight of the vehicle through the Earth's atmosphere are as follows:

$$\begin{aligned}
\dot{v} &= -\frac{S}{2m}\rho v^2 c_{\mathrm{D}}(u) - \frac{g\sin\gamma}{(1+\xi)^2}, \\
\dot{\gamma} &= \frac{S}{2m}\rho v c_{\mathrm{L}}(u) + \frac{v\cos\gamma}{R(1+\xi)} - \frac{g\cos\gamma}{v(1+\xi)^2}, \\
\dot{\xi} &= \frac{v}{R}\sin\gamma, \\
\dot{\zeta} &= \frac{v}{1+\xi}\cos\gamma,
\end{aligned}$$

where v is the velocity, γ is the flight path angle, $\xi = h/R$ is the normalized altitude, h is the altitude above the Earth's surface, R is the Earth's radius, and ζ is the distance on the Earth's surface of a trajectory of an Apollo-type vehicle. The control variable is the angle of attack u. For the lift and drag the following relations hold:

$$c_{\mathrm{D}} = c_{\mathrm{D}_0} + c_{\mathrm{DL}} \cos u, \tag{287}$$

$$c_{\mathrm{D}_0} = 0.88, \tag{288}$$

$$c_{\mathrm{DL}} = 0.52, \tag{289}$$

$$c_{\mathrm{L}} = c_{\mathrm{L}_0} \sin u, \tag{290}$$

$$c_{\mathrm{L}_0} = -0.505. \tag{291}$$

The air density is assumed to satisfy the relationship, $\rho = \rho_0 e^{-\beta R \xi}$. The values of the constants are

$$\begin{aligned} R &= 209.0352 \ (10^5 \text{ ft}), \\ S/m &= 50,000 \ (10^{-5} \text{ ft}^2 \text{ slug}^{-1}), \\ \rho_0 &= 2.3769 \times 10^{-3} (\text{slug ft}^{-3}), \\ g &= 3.2172 \times 10^{-4} \ (10^5 \text{ ft s}^{-2}), \\ \beta &= 1/0.235 \ (10^{-5} \text{ ft}^{-1}). \end{aligned}$$

The cost functional to be minimized that describes the total stagnation point convective heating per unit area is given by the integral

$$J = \int_0^{t_f} 10 v^3 \sqrt{\rho} \, \mathrm{d}t. \tag{292}$$

The vehicle is to be maneuvered into an initial position favorable for the final splashdown in the Pacific. Data at the moment of entry are

$$v(0) = 0.35 \ (10^5 \text{ ft s}^{-1}), \tag{293}$$

$$\gamma(0) = -5.75 \text{ deg}, \tag{294}$$

$$\xi(0) = 4/R \ (h(0) = 400,000 \text{ ft}), \tag{295}$$

$$\zeta(0) = 0 \ (10^5 \text{ ft}). \tag{296}$$

Pesch [112] considered two situations for the given problem, one with constraints on control and the other one with constraints on the state. We consider both of these problems in the sequel.

Problem A. Problem A imposes a control inequality constraint that limits the deceleration of the vehicle:

$$|u| \leq u_{\max}, \quad u_{\max} > 0. \tag{297}$$

The data prescribed at the unspecified terminal time t_f for Problem A are as follows

$$\begin{aligned}
v(t_f) &= 0.0165 \ (10^5 \text{ ft s}^{-1}), & (298)\\
\gamma(t_f) & \quad \text{unspecified}, & (299)\\
\xi(t_f) &= 0.75530/R \ (h(t_f) = 75,530 \text{ ft}), & (300)\\
\zeta(t_f) &= 51.6912 \ (10^5 \text{ ft}). & (301)
\end{aligned}$$

We solved this problem for all the cases considered by Pesch [112], and the results obtained using MTOA vindicate the proposed algorithm. For the sake of brevity, here we only give the results for the cases when $u_{\max} = 180$ and $u_{\max} = 68$.

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 7$. The threshold used for this problem was $\epsilon = 10^{-2}$. We used the implicit HS scheme for a high order discretization in MTOA. The algorithm for both cases terminated after 5 iterations and the overall CPU times taken by MTOA to solve the problem for both cases were 111.2 seconds and 125.6 seconds, respectively. The time histories of the velocity (v) and altitude above the Earth's surface (h) at the final iteration of MTOA for $u_{\max} = 180$ are shown in Figure 33. The time history of the angle of attack (u) along with the grid point distribution at the final iteration of MTOA for $u_{\max} = 180$ are shown in Figure 34. The time histories of the flight-path angle (γ) and the distance on the Earth's surface (ζ) at the final iteration of MTOA for $u_{\max} = 68$ are shown in Figure 35. The time history of the angle of attack (u) along with the grid point distribution at the final iteration of MTOA for $u_{\max} = 68$ are shown in Figure 36.

We also solved this problem for both the previous two cases on a grid with $J_{\min} = 3$ and $J_{\max} = 6$, but this time we uniformly refined the mesh after each iteration. The reason for choosing $J_{\max} = 6$ is because this problem could not be solved on a uniform grid finer than $\mathcal{V}_6$ because of hardware limitations. The CPU times taken by the algorithm for both the cases are shown in Table 16. We solved the problem using MTOA with the same parameters as before, but this time with $J_{\max} = 6$. MTOA terminated within four iterations for both cases. The overall CPU times taken by MTOA along with the number of nodes used by MTOA at the final iteration (N_f) are given in Table 16. As shown in Table 16, the MTOA outperformed the standard uniform grid implementation for this problem in terms of CPU time.

Table 16: Example 17: Uniform mesh vs. MTOA.

Problem	Uniform mesh		MTOA	
	N	t_{CPU} (sec)	N_f	t_{CPU} (sec)
A ($u_{\max} = 180$)	65	241.1	39	76.6
A ($u_{\max} = 68$)	65	174.9	30	94.2
B ($\xi_{\max} = 0.0066$)	65	265.4	41	60.0

Problem B. For this problem we impose a constraint to reduce re-ascent after the first dip into the atmosphere, that is, we have

$$\xi \leq \xi_{\max}, \quad \xi_{\max} > 0. \tag{302}$$

The data prescribed at the unspecified terminal time t_f for this problem are

$$v(t_f) = 0.01239929 \ (10^5 \text{ ft s}^{-1}), \tag{303}$$

$$\gamma(t_f) = -26.237124 \text{ deg}, \tag{304}$$

$$\xi(t_f) = 0.75530/R \ (h(t_f) = 75530 \text{ ft}), \tag{305}$$

$$\zeta(t_f) = 51.10198 \ (10^5 \text{ ft}). \tag{306}$$

Again, we solved this problem for all the cases considered by Pesch [112]. The results obtained using MTOA, once again, justify the proposed algorithm. For the sake of brevity, below we only give the results for the case when $\xi_{\max} = 0.0066$. We solved this problem on a grid with $J_{\min} = 2$ and $J_{\max} = 7$. We used the implicit HS scheme for a high order discretization in MTOA. The threshold used for this problem was $\epsilon = 10^{-2}$. The algorithm terminated after 6 iterations and the overall CPU time taken by MTOA to solve this problem was 235.2 seconds. The time histories of the velocity (v) and altitude above the Earth's surface (h) at the final iteration of MTOA are shown in Figure 37. The time history of the angle of attack (u) along with the final grid point distribution are shown in Figure 38.

When we attempted to solve the same problem on a uniform mesh with 33 nodes (with the same linear initial guess) using HS discretization, the algorithm failed to converge. Increasing the number of nodes to 65 nodes and using again the same linear initial guess did not help. We therefore solved this problem again on a grid with $J_{\min} = 2$ and $J_{\max} = 6$, but this time we progressively refined the mesh *uniformly* after each iteration. The value of $J_{\max} = 6$ was chosen because the problem could not be solved on a uniform grid finer than $\mathcal{V}_6$ owing to hardware limitations. The CPU time taken by the algorithm in this case is given in Table 16. Once again, we solved the problem using MTOA with the same parameters as before but this time with $J_{\max} = 6$. MTOA terminated in 5 iterations. The overall CPU time taken by MTOA along with the number of nodes used by MTOA at the final iteration (N_f) are also given in Table 16. These results again show the benefits of the MTOA in terms of accuracy and speed when compared with uniform grid implementations for this problem.

Next, we give the advantages of MTOA over the existing adaptive techniques for solving optimal control problems.

6.8 Advantages of the Proposed Multiresolution Trajectory Optimization Algorithm over the Existing Methods

First we show the advantages of the proposed multiresolution trajectory optimization algorithm over the current state-of-the-art algorithms for solving optimal control problems, namely, the algorithm of Betts et al. [18, 20, 22] and the pseudospectral knotting method [117], and then compare the proposed algorithm with the methods of Binder et al. [24, 25, 26, 27] and Schlegel et al. [121].

6.8.1 Advantages over the Method of Betts et al. [18, 20, 22]

The method of Betts et al. [18, 20, 22] selects the new grid points by solving an integer programming problem, that minimizes the maximum discretization error (found by integrating the dynamics of the system) by subdividing the current grid. In [22], the discretization error is computed by comparing the solution with a more accurate estimate using two (half) steps and by keeping the control fixed. The authors also assumed that the order of discretization, which effects the addition of mesh points to any subinterval in their mesh refinement algorithm, is constant. However, during the course of optimization process the actual order may vary with each iteration because of the potential activation of path constraints. It has been shown in [23] that having the wrong value for the order of discretization can seriously impact the mesh refinement algorithm of [22]. In order to overcome this problem, Betts et al. [20] derived a formula for estimating the order reduction by comparing the behavior of the discretization errors on successive mesh refinement iterations. But since the estimated order reduction is very sensitive to the computed discretization errors, the authors in [20] use a highly accurate quadrature method, namely Romberg quadrature (Appendix A.2), with a tolerance close to machine precision for computing the discretization errors.

Briefly, the mesh refinement method of Betts et al. [18, 20] comprises of three main steps. First, to interpolate all the states and controls using B-splines required for integrating the dynamics of the system. Second, integrate all the states using a highly accurate quadrature method, namely Romberg quadrature, with a tolerance close to machine precision in order to find the discretization errors. Third, solve an integer programming problem for refining the mesh.

Solving an integer programming problem for just refining the mesh on top of the NLP problem required for solving the optimal control problem can be computationally expensive. The proposed technique allows us to bypass solving any kind of secondary optimization problem for adding points to the mesh. Only simple interpolations are needed to refine the mesh, which can be done on the fly. Furthermore, the proposed mesh refinement algorithm does not involve any integrations, as opposed to the highly accurate integrations (Romberg quadratures) used by Betts et al, which again can be computationally expensive for nonlinear dynamics. Finally, the algorithm of [18, 20, 22] can only add points to the grid, whereas MTOA is capable of not only adding points to the grid but also removing points from the grid when and where is needed. Moreover both the operations of adding and removing points can be done in a single step.

6.8.2 Advantages over the Pseudospectral Knotting Method [117, 63]

The pseudospectral knotting method introduced by Ross and Fahroo [117] breaks a single phase problem with discontinuities and switches in states, control, cost functional, or dynamic constraints into a multiple phase problem with the phase boundaries, termed as "knots" by the authors, as the point of discontinuities or switchings. This way states and controls are allowed to be discontinuous across the phase boundaries and the phase boundaries can be fixed or free. On each phase, the problem is solved using the Legendre pseudospectral method [52] or Chebyshev pseudospectral method [55], and the free knots are part of the optimization process. The knots where the states are assumed to be continuous but no continuity condition is imposed on the controls are termed as *soft knots*. The soft knots can handle problems with smooth data and non-smooth solutions (e.g. switches and corners). But as pointed out by Ross [116] "Soft knots do not increase the speed of the algorithm; they are expected to improve accuracy. Consequently, the introduction of soft knots in the grid might significantly slow the algorithm." In the pseudospectral knotting method, one needs to know a priori the approximate number and location of singularities in the solution. These may not be known beforehand for most problems. One needs to know the number of irregularities in order to add that many soft knots in the optimization problem. Furthermore, the soft knots break the problem into multiple phases and each phase is solved using a particular number of grid points assigned by the user for that phase before starting the algorithm. Therefore, the user needs to know a priori the approximate locations of the singularities, without which, the user will not be able to assign correctly, the number of nodes for a particular phase. These facts are illustrated with the help of a simple example.

Let us assume an optimal control problem which has two switchings in the control and neither the number of switchings nor their approximate locations is known a priori before solving the problem. Now suppose the user adds only one soft knot. In that case, the pseudospectral knotting method will not be able to capture accurately both the switchings in the control since one soft knot can capture only one switching. On the other hand, adding too many soft knots will result in the unnecessary increase in the size of the optimization problem.

Now we assume that somehow it is known that there are going to be two switchings in the control. In this case, the user can add two free soft knots in the optimization problem, which will break the problem into three phases. But now the question is to add how many nodes in each of the phases, which can not be answered correctly unless the user knows the approximate location of the switching beforehand. Say for example, the user solves the optimization problem using pseudospectral knotting method with equal number of grid points N for all the phases and suppose after solving the problem it appears that the switchings take place at $0.1\tau_f$ and $0.95\tau_f$. Hence, the problem was solved by using N points for the phase $[0, 0.1\tau_f]$, N points for the phase $[0.1\tau_f, 0.95\tau_f]$, and N points for the phase $[0.95\tau_f, \tau_f]$. It may happen that the chosen value of N was redundant for the first and third phases, whereas N might have not been adequate for the second phase. While taking N to be sufficiently large for all the phases will increase the size of the NLP problem considerably.

In order to improve the pseudospectral methods, Gong et al. [63] present an algorithm in which the user specifies the number of nodes to be increased in a particular phase, in case the error of the computed optimal control between two successive iterations is greater than a prescribed threshold. The authors of Ref. [63] use the gradient of the control to determine (approximately) the location of the knots. Since the number of nodes that will be increased on a particular phase is assigned by the user a priori even before starting the code, the algorithm faces the same problem as discussed before for the case of pseudospectral knotting method.

On the other hand, the proposed multiresolution trajectory optimization algorithm is fully autonomous. The user need not know a priori the number nor the approximate locations of the irregularities in the solution. The proposed MTOA will automatically detect the regions in the solution that are nonsmooth and it will add points accordingly when and where is needed.

Furthermore, the nonuniform grids of pseudospectral methods result in grid distributions that remain fixed for each phase, since the location of the nodes are dictated by the zeros of the first derivative of the Legendre or Chebyshev polynomials (Appendix A.1), irrespective of the location of the soft knots [118]. Our algorithm uses a grid that is fully adaptive, embracing any form depending on the irregularities in the solution. This provides more flexibility in capturing any irregularities in the solution.

6.8.3 Advantages over the Algorithms of Binder et al. [24, 25, 26, 27] and Schlegel et al. [121]

From all previous references in this area the work of Binder et al. [24, 25, 26, 27], and Schlegel et al. [121] are the closest – at least in spirit – to the approach proposed in this thesis.

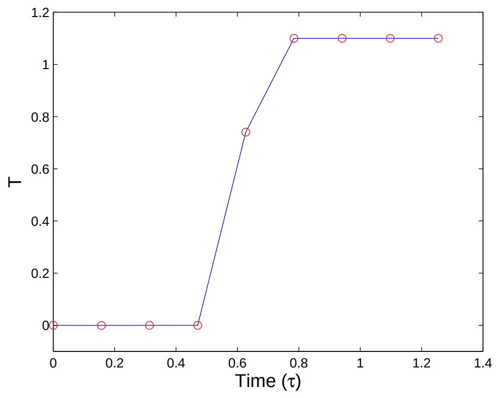
Binder et al. [24, 25, 26] work in the wavelet space by using the wavelet-Galerkin approach to discretize the optimal control problem into an NLP problem and use the local error analysis of the states and the wavelet analysis of the control profile to add or remove the wavelet basis functions. When one uses wavelet-Galerkin methods, multiplication in the physical space becomes convolution in the wavelet space, which is very costly as it is hard to compute convolutions efficiently. Whereas, by using the proposed mesh refinement technique, we always work in the physical domain and at the same time take advantage of one of the main properties of the wavelets - the multiresolution properties (see Chapter 2). Moreover, operations like multiplication and differentiation are fast in the physical domain as compared to the wavelet domain. Furthermore, nonlinearities can be handled with ease.

Binder et al. in [27] use the direct shooting approach, where the optimal control problem is converted into an NLP problem by parametrization of the control profiles, combined with a wavelet analysis of the gradients of the Lagrangian function with respect to the parametrization functions at the optimal points to determine the regions that require refinement. In order to improve this method further for problems with state and/or control path constraints Schlegel et al. [121] use wavelet analysis of the control profile to determine the regions that require refinement. Using wavelet analysis only to determine the regions of irregularities in the solution results in additional computational overhead, as one needs to transform back and forth between the physical and wavelet domain. In addition, one needs to interpolate the function values at the finest level every time one needs to perform the wavelet transform, which also results in additional computational overhead. Whereas the proposed mesh refinement technique uses only the retained points in the grid for further adding and removing points from the grid, and hence there is no need of interpolating the function values at the finest level.

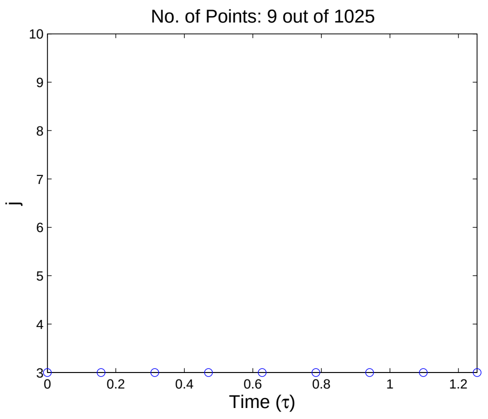
6.9 Summary

In this chapter we have proposed a novel multiresolution-based approach for direct trajectory optimization. The algorithm automatically, and with minimal effort, generates a nonuniform grid that reduces the discretization error with each iteration. As a result, one is able to capture the solution accurately and efficiently using a relatively small number of points. All the transition points in the solution (for example, bang-bang subarcs, or entry and exit points associated with state or mixed constraints) are captured with high accuracy. The convergence of the algorithm can be enhanced by initializing the algorithm on a coarse grid having a small number of variables. Once a converged solution is attained, the grid can be further refined by increasing the accuracy locally, only at the vicinity of those points that cannot be accurately interpolated by neighboring points in the grid. The methodology thus provides a compromise between robustness with respect to initial guesses, intermediate and final solution accuracy, and execution speed. These observations are supported by several numerical examples of challenging trajectory optimization problems. The proposed multiresolution trajectory optimization algorithm has been shown to have several advantages over the current state-of-the-art methods for solving the optimal control problems.

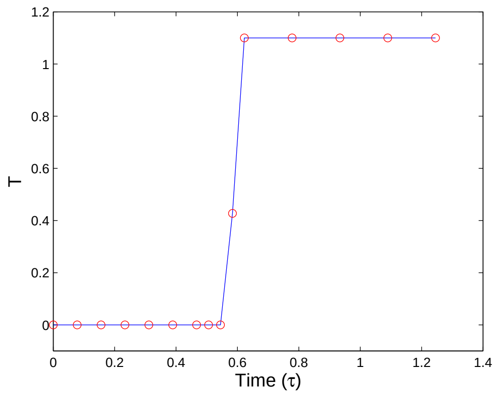
Next, we present two sequential trajectory optimization techniques for solving problems with moving targets and/or dynamically changing environments.



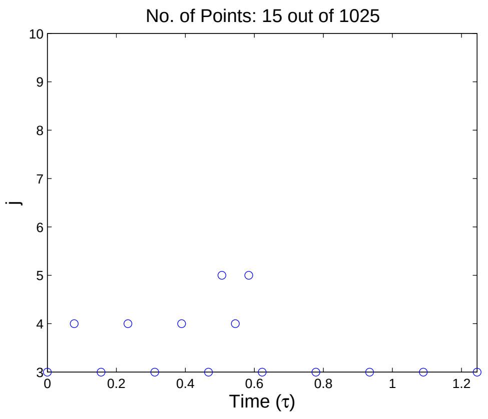
(a) Iteration 1: Time history of thrust T.



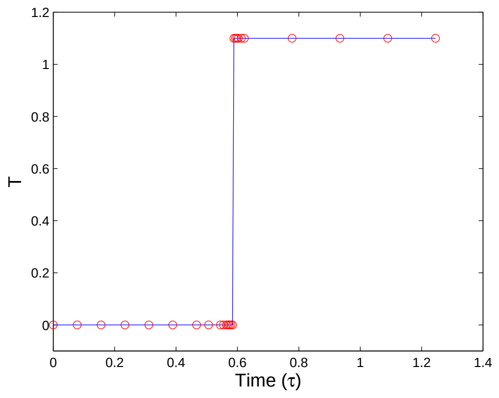
(b) Iteration 1: Grid point distribution.



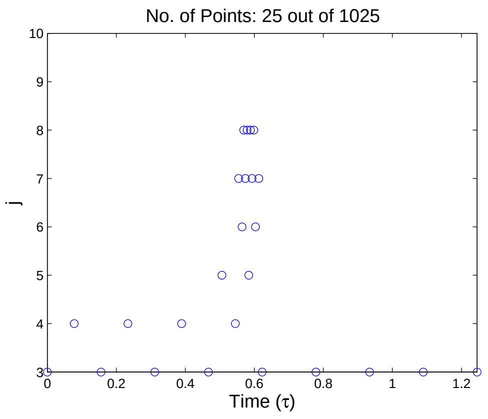
(c) Iteration 3: Time history of thrust T.



(d) Iteration 3: Grid point distribution.

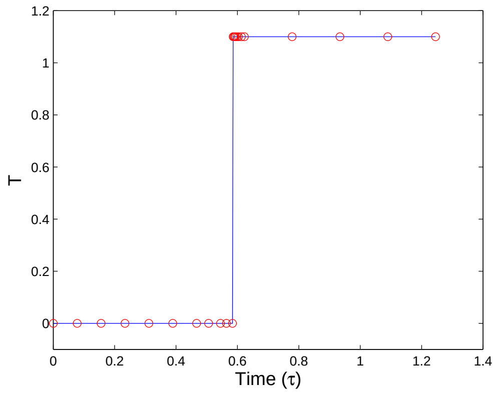


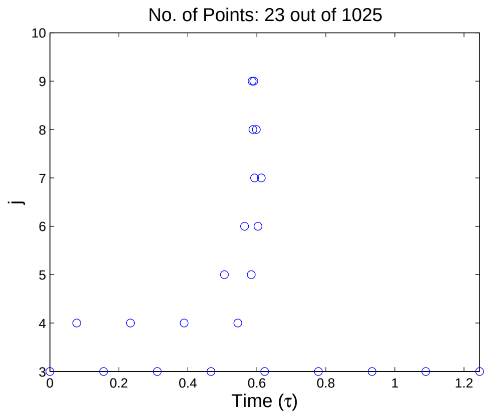
(e) Iteration 6: Time history of thrust T.



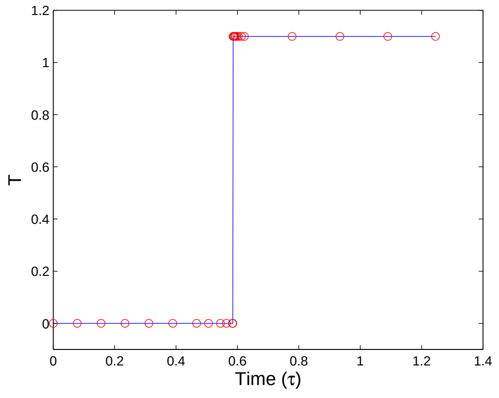
(f) Iteration 6: Grid point distribution.

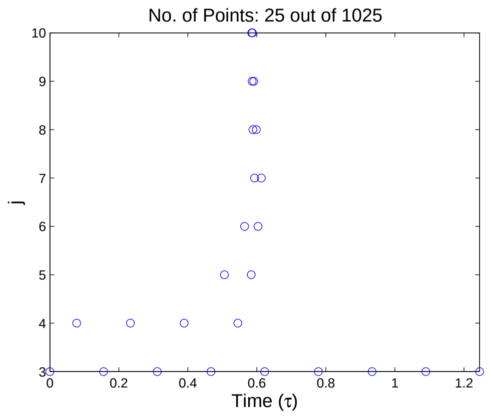
Figure 25: Example 13. Time history of thrust T along with the grid point distribution for iterations 1, 3, and 6.


(a) Iteration 7: Time history of thrust T.



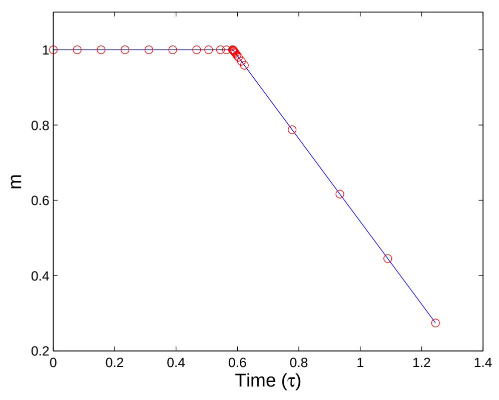
(b) Iteration 7: Grid point distribution.


(c) Iteration 8: Time history of thrust T.



(d) Iteration 8: Grid point distribution.

Figure 26: Example 13. Time history of thrust T along with the grid point distribution for iterations 7 and 8.


(a) Iteration 8: Time history of mass m.

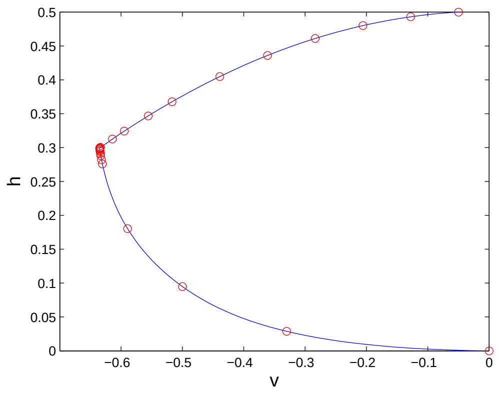
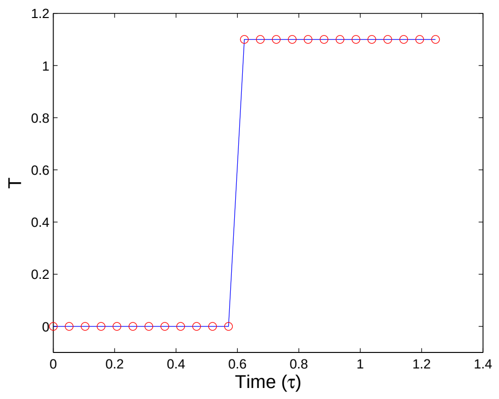
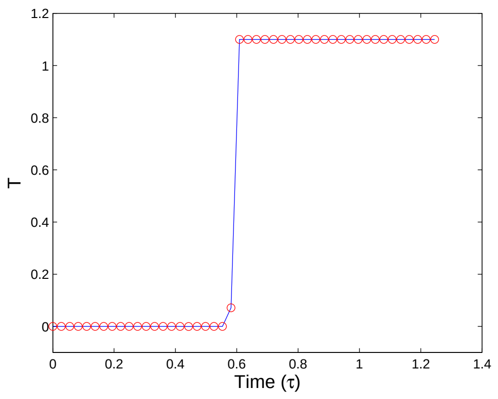

(b) Iteration 8: Phase portrait of v vs. h.

Figure 27: Example 13. Time history of mass m and the phase portrait of velocity v vs. altitude h for iteration 8.

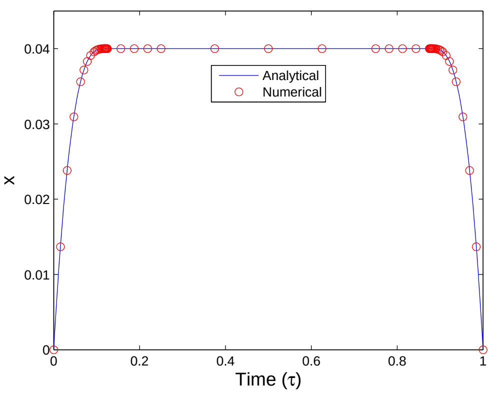


(a) No. of nodes used: 25.

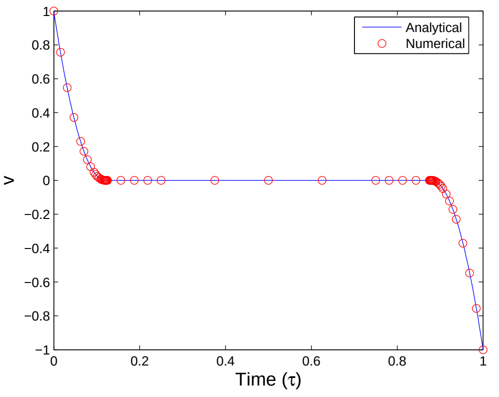


(b) No. of nodes used: 46.

Figure 28: Example 13. Time history of thrust T computed on a uniform mesh using an explicit fourth-order RK discretization.

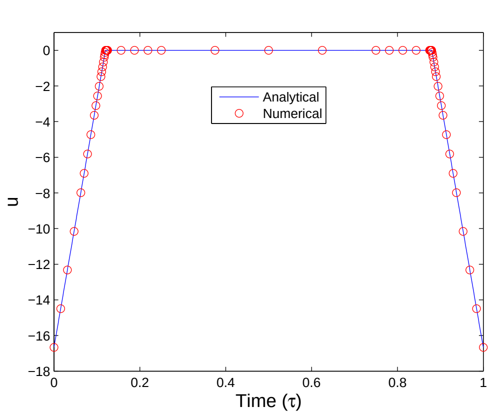


(a) Iteration 5: Time history of x.

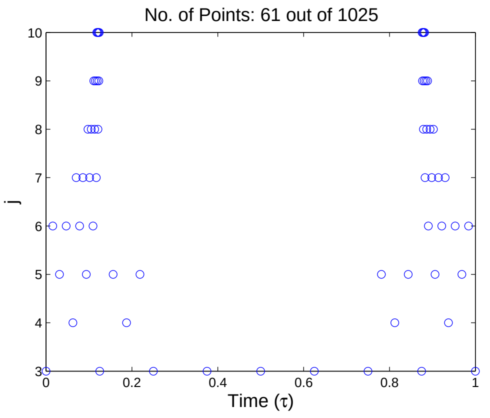


(b) Iteration 5: Time history of v.

Figure 29: Example 14: Time history of x, v at the final iteration of MTOA.

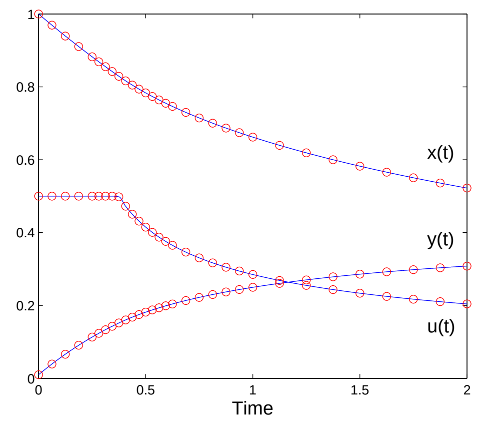


(a) Iteration 5: Time history of u.

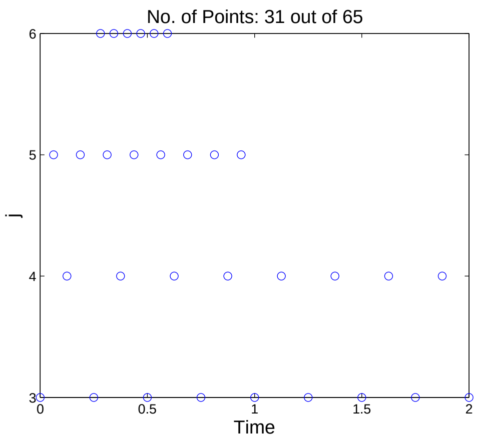


(b) Iteration 5: Grid point distribution.

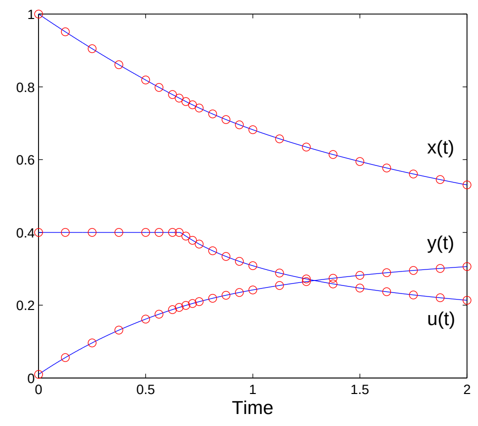
Figure 30: Example 14: Time history of u along with the grid point distribution at the final iteration of MTOA.



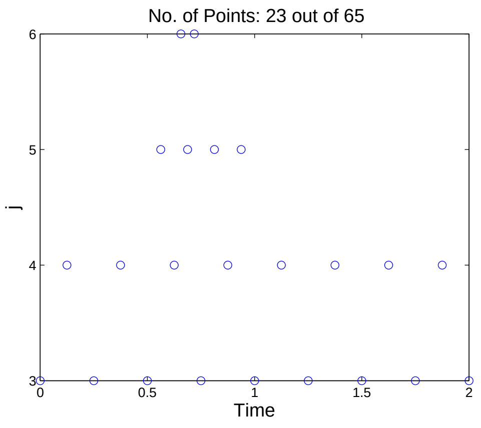
(a) Time history of x, y, and u for $u_{\max} = 0.5$.



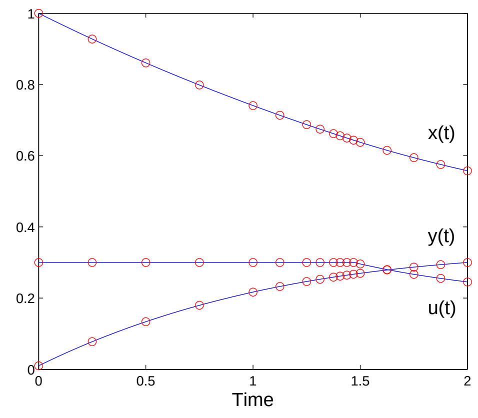
(b) Grid point distribution.



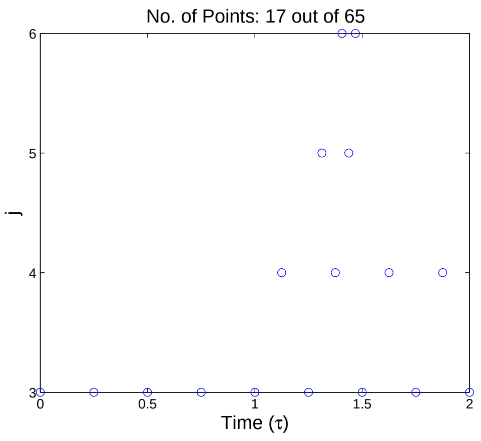
(c) Time history of x, y, and u for $u_{\max} = 0.4$.



(d) Grid point distribution.

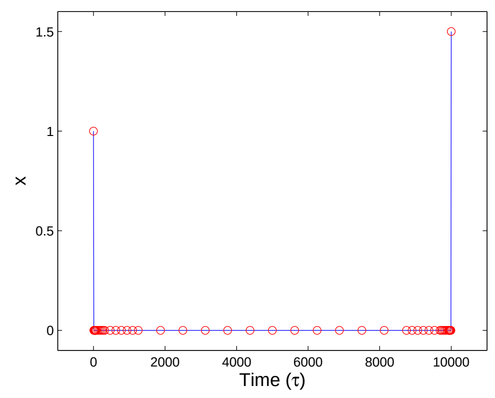


(e) Time history of x, y, and u for $u_{\max} = 0.3$.

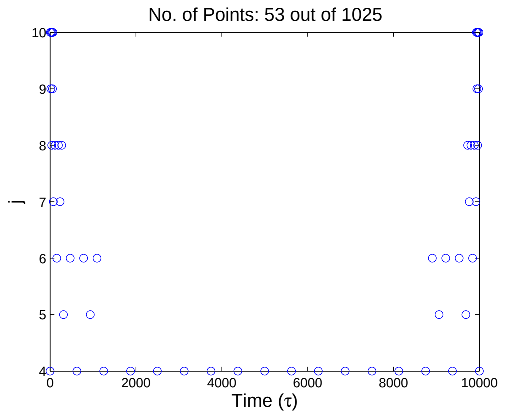


(f) Grid point distribution.

Figure 31: Example 15: Time history of states x, y and control u along with the grid point distributions for different $u_{\max}$ at the final iteration of MTOA.

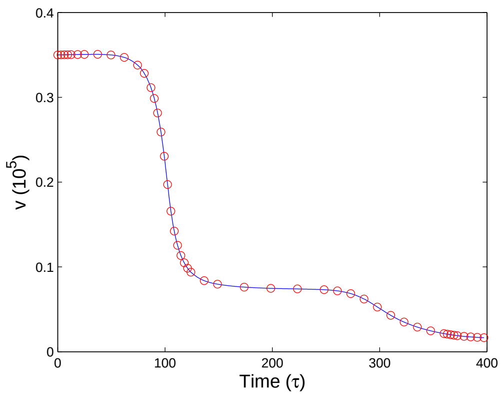


(a) Iteration 5: Time history of x.

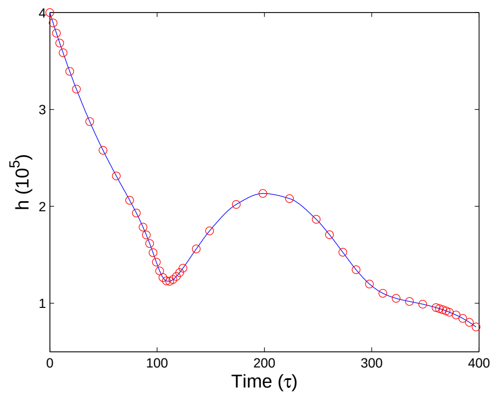


(b) Iteration 5: Grid point distribution.

Figure 32: Example 16: Time history of state x along with the grid point distribution at the final iteration of MTOA.

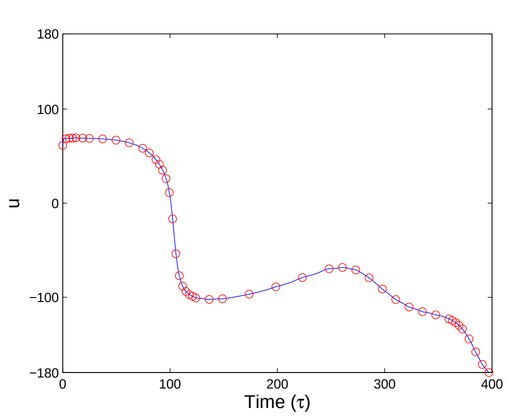


(a) Iteration 5: Time history of v.

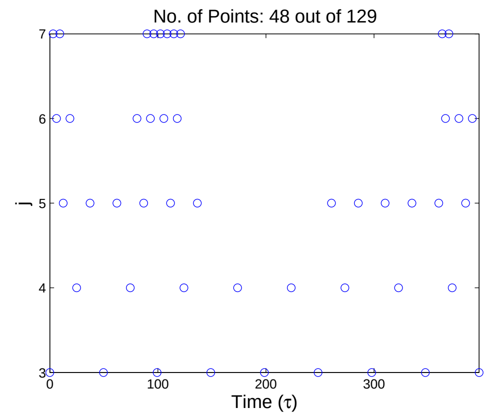


(b) Iteration 5: Time history of h.

Figure 33: Example 17: Problem A. Time histories of $v,\ h$ for $u_{\max} = 180$ at the final iteration of MTOA.

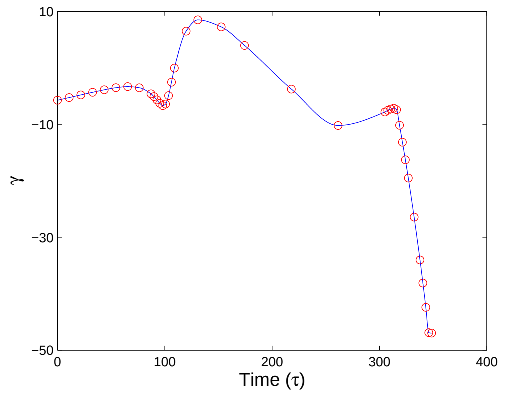


(a) Iteration 5: Time history of u.

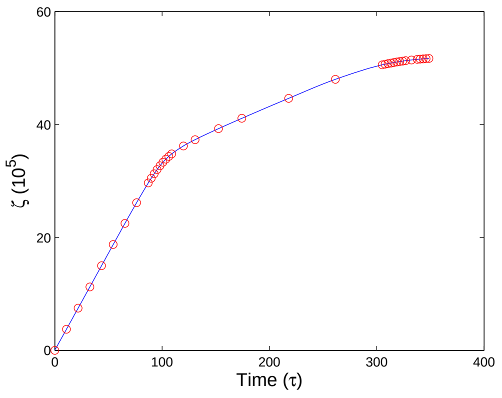


(b) Iteration 5: Grid point distribution.

Figure 34: Example 17: Problem A. Time history of u along with the grid point distribution for $u_{\max} = 180$ at the final iteration of MTOA.

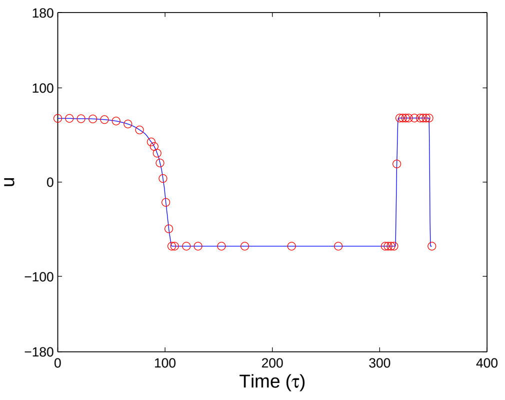


(a) Iteration 5: Time history of γ.

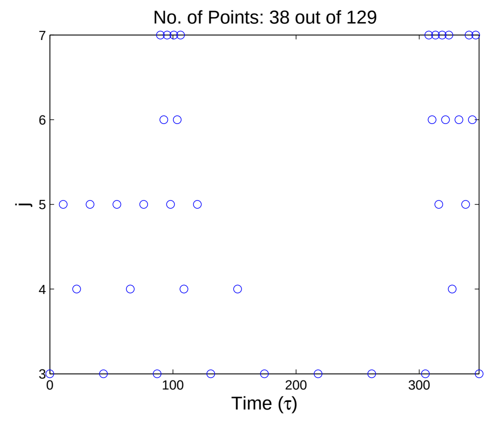


(b) Iteration 5: Time history of ζ.

Figure 35: Example 17: Problem A. Time histories of γ, ζ for $u_{\max} = 68$ at the final iteration of MTOA.

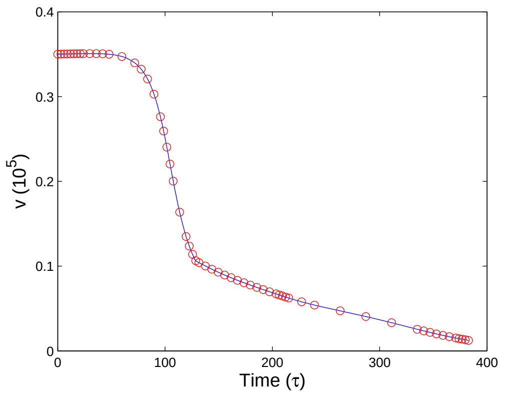


(a) Iteration 5: Time history of u.

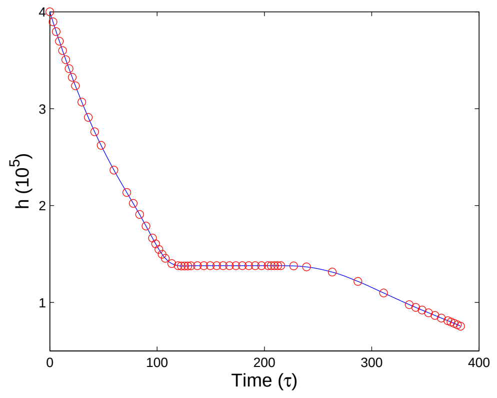


(b) Iteration 5: Grid point distribution.

Figure 36: Example 17: Problem A. Time history of u along with the grid point distribution for $u_{\max} = 68$ at the final iteration of MTOA.

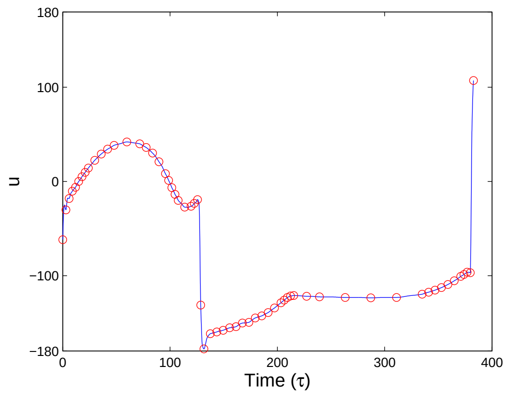


(a) Iteration 5: Time history of v.

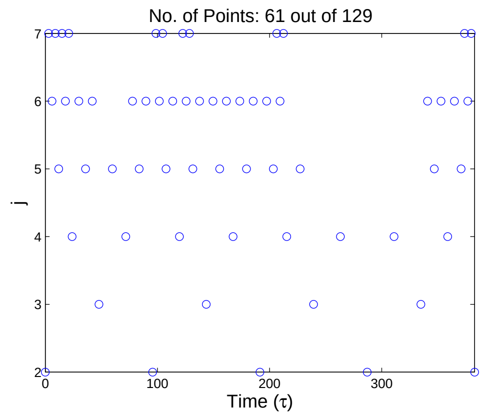


(b) Iteration 5: Time history of h.

Figure 37: Example 17: Problem B. Time histories of v, h for $\xi_{\max} = 0.0066$ at the final iteration of MTOA.



(a) Iteration 5: Time history of u.



(b) Iteration 5: Grid point distribution.

Figure 38: Example 17: Problem B. Time history of u for $\xi_{\max} = 0.0066$ along with the grid point distribution at the final iteration of MTOA.

CHAPTER VII

SEQUENTIAL MULTIRESOLUTION TRAJECTORY OPTIMIZATION FOR PROBLEMS WITH MOVING TARGETS AND/OR DYNAMICALLY CHANGING ENVIRONMENT

7.1 Problem Formulation

We wish to determine the state $\mathbf{x}(\cdot)$ and the control $\mathbf{u}(\cdot)$ that minimize the Bolza cost functional,

$$J = e(\mathbf{x}(\tau_f), \tau_f) + \int_{\tau_0}^{\tau_f} L(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)\mathrm{d}\tau, \tag{307}$$

where $e : \mathbb{R}^{N_x} \times \mathbb{R}_+ \to \mathbb{R}$, $\tau \in [\tau_0, \tau_f]$, $\mathbf{x} : [\tau_0, \tau_f] \to \mathbb{R}^{N_x}$, $\mathbf{u} : [\tau_0, \tau_f] \to \mathbb{R}^{N_u}$, $L : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times [\tau_0, \tau_f] \to \mathbb{R}$, subject to the state dynamics

$$\dot{\mathbf{x}}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau), \tag{308}$$

the state and control constraints

$$\mathbf{C}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) \leq 0, \tag{309}$$

where $\mathbf{C} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times [\tau_0, \tau_f] \to \mathbb{R}^{N_c}$, the initial condition

$$\mathbf{x}(\tau_0) = \mathbf{x}_0, \tag{310}$$

and the terminal constraint

$$\mathbf{e}_f(\mathbf{x}(\tau_f), \tau_f) = 0, \tag{311}$$

where $\mathbf{e}_f : \mathbb{R}^{N_x} \times [\tau_0, \infty) \to \mathbb{R}^{N_e}$. The initial time τ_0 is assumed to be given and the final time τ_f can be fixed or free.

Note that the functions $\mathbf{C}$ and $\mathbf{e}_f$ are assumed to be given at time t_0, but may change as the vehicle moves from $\mathbf{x}_0$ to $\mathbf{x}(\tau_f)$. This change is *not known a priori* so it cannot be modeled via the explicit time-dependence of $\mathbf{C}$ and $\mathbf{e}_f$ in (309) and (311).

7.2 Sequential Trajectory Optimization

In order to solve an optimal control problem with a moving target and/or a dynamically changing environment, in this chapter we present two sequential trajectory optimization algorithms. The basic idea behind the proposed algorithms is to solve the trajectory optimization problem at hand over the horizon $[\tau_0^1, \tau_f^1]$, and as we continue to move forward in time, we re-solve the optimization problem again on the new horizons $[\tau_0^i, \tau_f^i]$, where $i = 2, \ldots, N_H$, using the solution of the previous horizon as an initial guess. Here $\tau_0^1 = \tau_0$, $\tau_0^{i-1} < \tau_0^i < \tau_f^{i-1}$, $i = 2, \ldots, N_H$, and N_H is the number of horizons. If the final time is fixed, then

$$\tau_f^1 = \tau_f^2 = \cdots = \tau_f^{N_H} = \tau_f. \tag{312}$$

For further analysis, let

$$\Delta\tau_{\text{ro}}^i = \tau_0^{i+1} - \tau_0^i, \qquad i = 1, \ldots, N_H - 1, \tag{313}$$

be the time interval after which we re-optimize the trajectory. The value of $\Delta\tau_{\text{ro}}^i$ can be the same or different for all $i = 1, \ldots, N_H - 1$. For the case when $\Delta\tau_{\text{ro}}^i$ are all the same for $i = 1, \ldots, N_H - 1$, that is, $\tau_{\text{ro}}^i = \tau_{\text{ro}}$, for all $i = 1, \ldots, N_H - 1$, and the final time is fixed, the number of horizons is given by

$$N_H = \lfloor (\tau_f - \tau_0)/\Delta\tau_{\text{ro}}) \rfloor. \tag{314}$$

Next, we present the sequential trajectory optimization algorithm STOA I.

7.2.1 Sequential Trajectory Optimization Algorithm I (STOA I)

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in Eqs. (218) and (219). We first choose the minimum resolution level $J_{\min}$ based on the minimum time step required to achieve the desired accuracy in the regions of the solution where no constraints are active[1], the threshold $\epsilon(t)$ (the significance of which will be clear shortly), and the maximum resolution level $J_{\max}$. Then the proposed STOA I involves the following steps. First, we transcribe

[1]The minimum time step required to achieve a desired accuracy in the regions of the solution where no constraints are active can be calculated using the well-known error estimation formulas for RK schemes [64, 19, 65, 66].

the continuous trajectory optimization problem into an NLP problem using a q-stage RK discretization as described in Section 6.3. We use trapezoidal discretization for the first iteration and switch to a high-order discretization for subsequent iterations. Next, we set $i = 1$, $\texttt{iter} = 1$, initialize $\mathsf{Grid}^i_{\text{iter}} = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables. In the following, the interpolation operator $\mathcal{I}^p$ is constructed using ENO interpolations (see Section 4.3.2). Let us denote the set of initial guesses by $\mathcal{X}^i_{\text{iter}}$. The proposed sequential trajectory optimization algorithm then proceeds as follows:

STOA I

Step 1. Solve the NLP problem on $\mathsf{Grid}^i_{\text{iter}}$ with the initial guess $\mathcal{X}^i_{\text{iter}}$ on the horizon $[\tau^i_0, \tau^i_f]$. If $\mathsf{Grid}^i_{\text{iter}}$ has points from the level $\mathcal{W}_{J_{\max}-1}$, go to Step 4.

Step 2. **Mesh refinement**.

(a) i. If the problem either has pure state constraints or mixed constraints on states and controls, set $\Phi^i_{\text{iter}} = \{\mathbf{x}_{j,k}, \mathbf{u}_{j,k} : t_{j,k} \in \mathsf{Grid}^i_{\text{iter}}\}$, $N_r = N_x + N_u$.

ii. If the optimal control problem does not have any constraints or only pure control constraints are present, set $\Phi^i_{\text{iter}} = \{\mathbf{u}_{j,k} : t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$, $N_r = N_u$.

iii. In case no controls are present in the problem, set $\Phi^i_{\text{iter}} = \{\mathbf{x}_{j,k} : t_{j,k} \in \mathsf{Grid}^i_{\text{iter}}\}$, $N_r = N_x$.

In the following, let Φ^i_{iter} denote the set constructed in Step 2a of the algorithm, that is, let $\Phi^i_{\text{iter}} = \{\phi_\ell(t_{j,k}) : \ell = 1, \ldots, N_r,\ t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$.

(b) Initialize an intermediate grid $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}-1}$, with function values

$$\Phi_{\text{int}} = \{\phi_\ell(t_{J_{\min},k}) : \phi_\ell(t_{J_{\min},k}) \in \Phi^i_{\text{iter}},\ \forall\ t_{J_{\min},k} \in \mathcal{V}_{J_{\min}},\ \ell = 1, \ldots, N_r\}, \tag{315}$$

and set $j = J_{\min} - 1$.

i. Find the points that belong to the intersection of $\mathcal{W}_j$ and $\mathsf{Grid}^i_{\text{iter}}$

$$\hat{T}_j = \{\hat{t}_{j,k_m} : \hat{t}_{j,k_m} \in \mathcal{W}_j \cap \mathsf{Grid}^i_{\text{iter}},\ \text{for } m = 1, \ldots, N_{\hat{t}},\ 1 \leq N_{\hat{t}} \leq 2^j - 1\}. \tag{316}$$

If $\hat{T}_j$ is empty go to Step 2c otherwise go to the next step.

ii. Set $m = 1$.

A. Compute the interpolated function values at point $\hat{t}_{j,k_m} \in \hat{T}_j$, $\hat{\phi}_\ell(\hat{t}_{j,k_m}) = \mathcal{I}^p(\hat{t}_{j,k_m}, \mathcal{T}_{\mathsf{Grid}_{\text{int}}}(\hat{t}_{j,k_m}))$, where $\hat{\phi}_\ell$ is the ℓ-th element of $\hat{\phi}$, for $\ell = 1, \ldots, N_r$.

B. Calculate the interpolative error coefficient d_{j,k_m} at the point $\hat{t}_{j,k_m}$,[2]

$$d_{j,k_m}(\phi) = \max_{\ell=1,\ldots,N_r} d_{j,k_m}(\phi_\ell) = \max_{\ell=1,\ldots,N_r} |\phi_\ell(\hat{t}_{j,k_m}) - \hat{\phi}_\ell(\hat{t}_{j,k_m})|.$$

If the value of d_{j,k_m} is below the threshold $\epsilon(\hat{t}_{j,k_m})$, then reject $\hat{t}_{j,k_m}$ and go to Step 2(b)iiF, otherwise add $\hat{t}_{j,k_m}$ to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and move on to the next step.

C. Add to $\mathsf{Grid}_{\text{int}}$ N_{neigh} points on the left and N_{neigh} points on the right of the point $\hat{t}_{j,k_m}$ in $\mathcal{W}_j$.

D. If $N_{\text{neigh}} = 0$ add to $\mathsf{Grid}_{\text{int}}$ points belonging to the set

$$(\mathcal{V}_{\hat{J}} \cap [t_{j,k_m}, t_{j,k_m+1}]) \setminus \mathsf{Grid}_{\text{int}},$$

else add to $\mathsf{Grid}_{\text{int}}$ points belonging to the set

$$(\mathcal{V}_{\hat{J}} \cap [\hat{t}_{j,k_m - N_{\text{neigh}}}, \hat{t}_{j,k_m + N_{\text{neigh}}}]) \setminus \mathsf{Grid}_{\text{int}}.$$

Here $\hat{J} = \min\{j + \hat{j}, J_{\max}\}$, where $\hat{j} = 2$ if $\texttt{iter} = 1$ else $\hat{j} \geq 2$, $\hat{j}$ is the number of finer levels from which the points be added to the grid for refinement.

E. Add the function values at all the newly added points to Φ_{int}. If the function value at any of the newly added points is not known, we interpolate the function value at that point from the points in $\mathsf{Grid}^i_{\text{iter}}$ and their function values in Φ^i_{iter} using $\mathcal{I}^p(\cdot, \mathcal{T}_{\mathsf{Grid}^i_{\text{iter}}}(\cdot))$.

F. Increment m by 1. If $m \leq N_{\hat{t}}$ go to Step 2(b)iiA, otherwise move on to the next step.

[2]Note that $\phi_\ell(\hat{t}_{j,k}) \in \Phi^i_{\text{iter}}$ for all $\hat{t}_{j,k} \in \hat{T}_j$ and $\ell = 1, \ldots, N_r$.

iii. Set $j = j + 1$. If $j < J_{\max}$ go to Step 2(b)i, otherwise go to Step 2c.

(c) Terminate. The final nonuniform grid is $\mathsf{Grid}_{\text{new}} = \mathsf{Grid}_{\text{int}}$ and the corresponding function values are in the set $\Phi_{\text{new}} = \Phi_{\text{int}}$.

Step 3. Set $\texttt{iter} = \texttt{iter} + 1$. If the number of points and the level of resolution remain the same after the mesh refinement procedure, terminate. Otherwise, interpolate the NLP solution found in Step 1 on the new mesh $\mathsf{Grid}_{\text{new}}$, which will be the new initial guess $\mathcal{X}^i_{\text{iter}}$. Reassign the set $\mathsf{Grid}^i_{\text{iter}}$ to $\mathsf{Grid}_{\text{new}}$, and go to Step 1.

Step 4. New horizon:

(a) Set $\mathsf{Grid}^i = \mathsf{Grid}^i_{\text{iter}}$.

(b) Increment i by 1.

(c) Set $\tau^i_0 = \tau^{i-1}_0 + \Delta\tau^{i-1}_{\text{ro}}$.

(d) Terminate if $\tau^i_0 \geq \tau^{i-1}_f$, otherwise set $\texttt{iter} = 1$, $\mathsf{Grid}^i_{\text{iter}} = \mathcal{V}_{J_{\min}}$.

(e) Interpolate the solution of the previous horizon $[\tau^{i-1}_0, \tau^{i-1}_f]$ given on Grid^{i-1} to $\mathsf{Grid}^i_{\text{iter}}$, which will be our new initial guess $\mathcal{X}^i_{\text{iter}}$ for Step 1[3].

(f) Update information about the path constraints and the terminal constraints.

Step 5. Go to Step 1.

Remark 8. Although the STOA I will work for any form of $\epsilon(t)$, we recommend using the following form,

$$\epsilon(t) = \hat{\epsilon} E(\max\{0, t - t^{i+1}_0\}), \tag{317}$$

where $\hat{\epsilon}$ be at least of order $h_{J_{\min}} = 1/2^{J_{\min}}$, and $E : [0, 1 - t^{i+1}_0] \to \mathbb{R}^+$ is such that $E(0) = 1$. For example, one may choose $E(t) = e^{\beta(\max\{0, t - t^{i+1}_0\})}$, where $\beta \in \mathbb{R}^+$, for $t \in [0, 1]$, and $i = 1, \ldots, N_H$. This choice implies that the threshold is constant, is equal to $\hat{\epsilon}$ for $t \in [0, t^{i+1}_0]$, and it varies with time for $t \in (t^{i+1}_0, 1]$. Such a choice stems from the fact

[3]It should be noted that although $\mathsf{Grid}^i_1 = \mathsf{Grid}^{i-1}_1$ on the transformed domain $[0, 1]$ but both the grids Grid^{i-1}_1 and Grid^i_1 correspond to different time intervals, that is, $[\tau^{i-1}_0, \tau^{i-1}_f]$ and $[\tau^i_0, \tau^i_f]$ respectively.

that the solution should be calculated with high precision till the initial time of the next horizon.

We demonstrate the above algorithm with the help of a simple, yet practical example, in which the terminal condition is assumed to be changing with time.

Example 18

Consider the Zermelo's problem taken from Ref. [33]. A ship must travel through a region of strong currents. The equations of motion of the ship are

$$\dot{x} = V\cos\theta + u(x,y), \tag{318}$$

$$\dot{y} = V\sin\theta + v(x,y), \tag{319}$$

where θ is the heading angle of the ship's axis relative to the (fixed) coordinate axes, (x, y) represent the position of the ship, V is the magnitude of the ship's velocity relative to the water, and (u, v) are the velocity components of the current in the x and y directions, respectively. The magnitude and direction of the currents are assumed to be,

$$u = -Vy, \tag{320}$$

$$v = 0, \tag{321}$$

and the ship's velocity V is assumed to be unity. The path constraint is the width of the river, and we assume

$$0 \leq x \leq 6.8. \tag{322}$$

The problem is to steer the ship in such a way so as to minimize the time necessary to go from a given point A to another given point B. For this specific example, we assume the coordinates of point A to be

$$x_{\mathrm{A}} = x(0) = 0, \qquad y_{\mathrm{A}} = y(0) = -4. \tag{323}$$

The target B is assumed to be moving. However, the trajectory of point B is not known in advance. Initially, the coordinates of B are taken to be as follows

$$x_{\mathrm{B}} = x(\tau_f) = 6, \qquad y_{\mathrm{B}} = y(\tau_f) = 1. \tag{324}$$

We assume (Step 4f of STOA I) that the information about the target is updated every time before the re-optimization is done on a new horizon. We also assume that the trajectory of the target is given by

$$x_{\mathrm{B}}(\tau) = 6 - 0.1\tau, \qquad y_{\mathrm{B}}(\tau) = 1 - 0.2\tau. \tag{325}$$

Hence, on each horizon H_i, where $i = 2, \ldots, N_H$, we have the following terminal constraints,

$$x(\tau_f^i) = 6 - 0.1\tau_0^i, \qquad x(\tau_f^i) = 1 - 0.2\tau_0^i. \tag{326}$$

For the sake of simplicity, and so that the proposed algorithm terminates in a finite number of iterations, we assume that if $\tau_0^i \geq 5$, for some $i \in [1, N_H]$, then

$$x(\tau_f^m) = 6 - 0.1\tau_0^i, \qquad y(\tau_f^m) = 1 - 0.2\tau_0^i, \tag{327}$$

for all $m = i, \ldots, N_H$.

We solved this problem on a grid with $J_{\min} = 2$ and $J_{\max} = 7$ for each horizon with $\epsilon(\tau) = 0.01e^{10\max\{0,\tau-\tau_0^{i+1}\}}$, where $i = 1, \ldots, N_H$. The other parameters used in the simulation are $p = 3$ and $N_{\text{neigh}} = 0$. A fourth-order implicit Hermite-Simpson scheme [82] was used as a high-order scheme for discretizing the continuous optimal control problem into an NLP problem.

To solve this problem, we let $\Delta\tau_{\text{ro}}^i \approx 1\,\text{sec}$ ($i = 1, \ldots, N_H - 1$). One way for finding the initial conditions $(x(\tau_0^i), y(\tau_0^i))$ for the next horizon (H_i) is to integrate the dynamics of the system using the control found on the previous horizon (H_{i-1}) for a duration of $\Delta\tau_{\text{ro}}^i$ seconds and then use the integrated states at the end of the interval $[\tau_0^{i-1}, \tau_0^i]$ as the initial conditions for solving the NLP problem on the new horizon (H_i). For this example, we picked the initial time τ_0^i for each horizon H_i, $i = 1, \ldots, N_H$, as follows. For the first horizon we set $\tau_0^1 = 0$ and for subsequent horizons we choose

$$\tau_0^i = \min_{\tau}\{\tau \in \mathsf{Grid}_\tau^{i-1} : \tau \geq \tau_0^{i-1} + 0.95\}, \tag{328}$$

where $i = 2, \ldots, N_H$,

$$\mathsf{Grid}_\tau^{i-1} = \{\tau : \tau = (\tau_f^{i-1} - \tau_0^{i-1})t_{j,k} + \tau_0^{i-1}, \ \forall\ t_{j,k} \in \mathsf{Grid}^{i-1}\}. \tag{329}$$

The algorithm terminated after solving the problem on 6 horizons. The number of iterations taken by the algorithm before the algorithm terminated on each horizon ($\texttt{iter}_f$), the maximum resolution level reached on each horizon (J_f), the number of nodes used by the algorithm at the final iteration on each horizon (N_f), along with the initial and the final times for all the horizons are shown in Table 17.

Table 17: Example 18. Target snapshots.

Horizon	$\texttt{iter}_f$	J_f	N_f	τ_0	τ_f
H_1	3	4	9	0	5.6018
H_2	3	4	9	1.0503	5.5198
H_3	4	5	13	2.1677	5.4687
H_4	6	7	17	3.1993	5.4965
H_5	2	3	7	4.2043	5.5818
H_6	1	2	5	5.2374	5.7538

The computed trajectory found using the proposed algorithm, along with the grid point distributions for different horizons are shown in Figures 39 and 40. In these figures, the initial point A is depicted by a square and the target point B is depicted by a cross. As pointed out earlier, the target B is assumed to be non-stationary, and for convenience of the reader, in Figures 39, 40 all the previous locations of B are also shown in addition to the current position of target B. The optimal controls found for all the horizons are shown in Figure 41. From Figures 39(a), 41(a), we see that the proposed algorithm used only 9 points out of 129 points of the grid $\mathcal{V}_7$ for solving the given problem on the first horizon $[0, \tau_f]$. The grid point distribution 39(b) shows that the points from the finer resolution levels $\mathcal{V}_3$, $\mathcal{V}_4$ are concentrated only near the initial time. On the second horizon, we assume that the target B has moved to the new location. From Figures 39(c), 39(d), and 41(b), we again find that the algorithm used only 9 points for discretizing the trajectory and the points from the finer levels of resolution $\mathcal{V}_3$, $\mathcal{V}_4$ are again clustered near the current time. For the third horizon, the algorithm used 13 points to find the optimal solution. From the grid point distribution in Figure 39(f), it is evident that the algorithm started adding points from the finer resolution level, $\mathcal{V}_5$, near the location where there should be a switching in the control, since the ship is approaching the shore. Moving on to the fourth horizon, we see that, as the boat is approaching the shore, there should be a switching in the control.

Hence, in order to capture this control switching, the algorithm further added points at the finer resolution levels $\mathcal{V}_6$, and $\mathcal{V}_7$, as can be observed from the grid point distribution for the fourth horizon (Figure 40(b)). For the fifth and sixth horizons, the algorithm used only 7 and 5 points respectively for computing the optimal solution. Since on the sixth horizon, we had $\tau_0^6 > 5$, the target was further assumed to be stationary located at

$$x(\tau_f^m) = 6 - 0.1\tau_0^5, \qquad y(\tau_f^m) = 1 - 0.2\tau_0^5, \tag{330}$$

for all $m = 5, \ldots, N_H$. Hence, the algorithm terminated after solving the problem on the sixth horizon. The overall CPU time taken by STOA I to solve this problem was 5.1 seconds. The combined trajectory and the control found on different horizons is shown in Figure 42.

Next, we incorporate the information of the trajectory profile of the target (325) in the optimal control problem itself. Since the trajectory profile of the target is assumed to be given for the optimal control problem at hand, the resulting problem can be solved in one go using MTOA (see Section 6.5). The results found using MTOA are shown in Figure 42 and the overall CPU time taken by MTOA to solve this problem was 9.5 seconds. The minimum time (τ_f) to steer the ship from point A to the target point B found using MTOA is $\tau_f = 5.8637$. We also solved the same problem using STOA I. For comparison purposes the results found using STOA I are again shown in Figure 42. The number of iterations taken by the algorithm before the algorithm terminated on each horizon ($\texttt{iter}_f$), the maximum resolution level reached on each horizon (J_f), the number of nodes used by the algorithm at the final iteration on each horizon (N_f), along with the initial and the final times for all the horizons are shown in Table 18. The overall CPU time to solve the problem using STOA I was 6.3 seconds. Hence, we see that the cost found by solving the problem using MTOA is less by 5^{-4} than the cost found using STOA I for the problem when the trajectory profile of the target is assumed to be known. However, we see that the overall CPU time taken by STOA I is about two-thirds of the overall CPU time taken by MTOA to solve the same problem.

Table 18: Example 18. Target trajectory known.

Horizon	$\texttt{iter}_f$	J_f	N_f	τ_0	τ_f
H_1	3	4	9	0	5.9065
H_2	3	4	9	1.1075	5.8256
H_3	3	4	11	2.2870	5.8648
H_4	6	7	17	3.4051	5.8643
H_5	1	2	5	4.4810	5.8642
H_6	1	2	5	5.5184	5.8642

7.2.2 Sequential Trajectory Optimization Algorithm II (STOA II)

In this section, we present yet another sequential trajectory optimization scheme referred to as STOA II, which takes full advantage of the multiresolution structure of the grid in the mesh refinement procedure so that the previously computed information is retained, while moving from one horizon to the next. In order to avoid notational complexities, and without loss of generality, we will assume in this section that the time interval of interest is the unit interval $t \in [0,1] = [\tau_0, \tau_f]$. Transformation (220) can be used to convert any optimal control problem from the domain $[\tau_0, \tau_f]$ to $[0,1]$.

Consider again a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in Eqs. (218) and (219). We choose the parameters $J_{\min}$, $J_{\max}$, and $\epsilon(t)$ as for the STOA I. Then the proposed STOA II involves the following steps. First, we transcribe the continuous trajectory optimization problem into an NLP problem using a q-stage RK discretization as described in the previous section. We use trapezoidal discretization for the first iteration and switch to a high-order discretization for subsequent iterations. Next, we set $i = 1$, $\texttt{iter} = 1$, $t_0^i = 0$, initialize $\mathsf{Grid}_{\text{iter}}^i = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables ($\mathcal{X}_{\text{iter}}^i$). Fix $\bar{J} = J_{\min} - 1$. The proposed sequential trajectory optimization algorithm proceeds as follows:

Step 1. Solve the NLP problem on $\mathsf{Grid}_{\text{iter}}^i$ with the initial guess $\mathcal{X}_{\text{iter}}^i$ on the horizon $[t_0^i, 1]$. If $\mathsf{Grid}_{\text{iter}}^i$ has points from the level $\mathcal{W}_{J_{\max}-1}$, go to Step 4.

Step 2. Find $\mathsf{Grid}_{\text{new}}$ using the mesh refinement step (Step 2) of STOA I.

Step 3. Set $\texttt{iter} = \texttt{iter} + 1$. If the number of points and the level of resolution remain the same after the mesh refinement procedure then terminate, otherwise interpolate the NLP solution found in Step 1 on the new mesh $\mathsf{Grid}_{\text{new}}$, which will be our new initial guess $\mathcal{X}^i_{\text{iter}}$, reassign the set $\mathsf{Grid}^i_{\text{iter}}$ to $\mathsf{Grid}_{\text{new}}$, and go to Step 1.

Step 4. New horizon.

(a) Set $\mathsf{Grid}^i = \mathsf{Grid}^i_{\text{iter}}$.

(b) Increment i by 1.

(c) Set $t^i_0 = t_{\bar{J},i-1}$.

(d) If $i = 2^{\bar{J}} + 1$ terminate, else go to the next step.

(e) Set $\mathsf{Grid}^{i-} = \{t : t \in \mathsf{Grid}^{i-1} \text{ and } t \geq t_{\bar{J},i-1}\}$.

(f) If the number of points in the set $\{\mathsf{Grid}^{i-} \cap \mathcal{V}_{J_{\min}-1}\}$ is less than $p+1$, set $J_{\min} = J_{\min} + 1$.

(g) Set $\texttt{iter} = 1$, $\mathcal{V}_j = \mathcal{V}_j \setminus (\mathcal{V}_j \cap [0, t_{\bar{J},i-1}))$ (where $j = J_{\min} - 1, \ldots, J_{\max}$), and $\mathcal{W}_j = \mathcal{W}_j \setminus (\mathcal{W}_j \cap [0, t_{\bar{J},i-1}))$ (where $j = J_{\min} - 1, \ldots, J_{\max} - 1$). Find $\mathsf{Grid}_{\text{new}}$ using the mesh refinement step (Step 2) of STOA I with $\mathsf{Grid}^i_{\text{iter}} = \mathsf{Grid}^{i-}$.

(h) Increment $\texttt{iter}$ by 1 and reassign the set $\mathsf{Grid}^i_{\text{iter}}$ to $\mathsf{Grid}_{\text{new}}$.

(i) Interpolate the NLP solution given on Grid^{i-} to $\mathsf{Grid}^i_{\text{iter}}$, which will be our new initial guess $\mathcal{X}^i_{\text{iter}}$ for Step 1.

(j) Update the information about the path constraints and the terminal constraints.

Step 5. Go to Step 1.

Remark 9. Although STOA II will work for any form of the threshold $\epsilon(t)$, we recommend choosing

$$\epsilon(t) = \hat{\epsilon} E(\max\{0, t - t_{\bar{J},i}\}), \tag{331}$$

where $\hat{\epsilon}$ is at least of order $h_{J_{\min}} = 1/2^{J_{\min}}$, and $E : [0, 1 - t_{\bar{J},i}] \to \mathbb{R}^+$ such that $E(0) = 1$, $t \in [0, 1]$, and $i = 1, \ldots, N_H$. This choice implies that the threshold is constant and is equal to $\hat{\epsilon}$ for $t \in [0, t_{\bar{J},i}]$ and varies with time for $t \in (t_{\bar{J},i}, 1]$. Such a choice stems from the fact that the solution should be calculated with high precision till the initial time of the next horizon, which in this case would be $t_{\bar{J},i}$.

Example 19

In this example, we again consider the re-entry guidance problem of an Apollo-type vehicle taken from Ref. [112]. The equations of motion during the flight of the vehicle through the Earth's atmosphere are as follows:

$$\begin{aligned}
\dot{v} &= -\frac{S}{2m}\rho v^2 c_{\mathrm{D}}(u) - \frac{g\sin\gamma}{(1+\xi)^2},\\
\dot{\gamma} &= \frac{S}{2m}\rho v c_{\mathrm{L}}(u) + \frac{v\cos\gamma}{R(1+\xi)} - \frac{g\cos\gamma}{v(1+\xi)^2},\\
\dot{\xi} &= \frac{v}{R}\sin\gamma,\\
\dot{\zeta} &= \frac{v}{1+\xi}\cos\gamma,
\end{aligned}$$

where v is the velocity, γ is the flight path angle, $\xi = h/R$ is the normalized altitude, h is the altitude above the Earth's surface, R is the Earth's radius, and ζ is the distance on the Earth's surface of a trajectory of an Apollo-type vehicle. The control variable is the angle of attack u. For the lift and drag the following relations hold:

$$c_{\mathrm{D}} = c_{\mathrm{D}_0} + c_{\mathrm{DL}}\cos u, \quad c_{\mathrm{D}_0} = 0.88, \quad c_{\mathrm{DL}} = 0.52, \tag{332}$$

$$c_{\mathrm{L}} = c_{\mathrm{L}_0}\sin u, \quad c_{\mathrm{L}_0} = -0.505. \tag{333}$$

The air density is assumed to satisfy

$$\rho = \rho_0 e^{-\beta R\xi}. \tag{334}$$

The values of the constants are

$$\begin{aligned}
R &= 209.0352\ (10^5\ \mathrm{ft}),\\
S/m &= 50,000\ (10^{-5}\ \mathrm{ft}^2\ \mathrm{slug}^{-1}),\\
\rho_0 &= 2.3769\times 10^{-3}(\mathrm{slug}\ \mathrm{ft}^{-3}),\\
g &= 3.2172\times 10^{-4}\ (10^5\ \mathrm{ft}\ \mathrm{s}^{-2}),\\
\beta &= 1/0.235\ (10^{-5}\ \mathrm{ft}^{-1}).
\end{aligned}$$

The cost functional to be minimized that describes the total stagnation point convective heating per unit area is given by the integral

$$J(u) = \int_0^{\tau_f} 10v^3\sqrt{\rho}\,\mathrm{d}\tau. \tag{335}$$

The vehicle is to be maneuvered into an initial position favorable for the final splashdown in the Pacific. The data at the moment of entry are

$$v(0) = 0.35 \ (10^5 \text{ ft s}^{-1}), \qquad \gamma(0) = -5.75 \ \text{deg}, \tag{336}$$

$$\xi(0) = 4/R \ (h(0) = 400,000 \text{ ft}), \qquad \zeta(0) = 0 \ (10^5 \text{ ft}). \tag{337}$$

The data prescribed at the unspecified terminal time t_f for this problem are

$$v(\tau_f) = 0.0165 \ (10^5 \text{ ft s}^{-1}), \qquad \gamma(\tau_f) \text{ unspecified}, \tag{338}$$

$$\xi(\tau_f) = 0.75530/R \ (h(t_f) = 75530 \text{ ft}), \qquad \zeta(\tau_f) = 51.6912 \ (10^5 \text{ ft}). \tag{339}$$

The angle of attack is constrained to be between $\pm 68\,\text{deg}$, that is,

$$|u| \leq 68\,\text{deg}\,. \tag{340}$$

We have used STOA II to solve this problem with $J_{\min} = 4$, and $J_{\max} = 7$. The threshold used for this problem was

$$\epsilon(t) = 0.01e^{7\max\{0, t - t_{3,i}\}}, \quad i = 1, \ldots, N_H. \tag{341}$$

The other parameters used in the simulation for the mesh refinement step were $p = 3$ and $N_{\text{neigh}} = 1$. A fourth-order implicit Hermite-Simpson scheme [82] was used as a high-order scheme for discretizing the continuous optimal control problem into an NLP problem. The algorithm terminated after solving the problem on 8 horizons and the overall CPU time taken by the algorithm was 41.2 seconds, out of which 22 seconds were used to compute the solution on the first horizon H_1. For sake of brevity, we only show the time histories of the control u, along with the grid point distribution for different horizons, in Figures 43, 44, and 45. The number of iterations taken by the algorithm before the algorithm terminated on each horizon ($\texttt{iter}_f$), the maximum resolution level reached on each horizon (J_f), and the number of nodes used by the algorithm at the final iteration on each horizon (N_f) are shown in Table 19.

Table 19: Example 19.

Horizon	$\texttt{iter}_f$	J_f	N_f
H_1	2	5	24
H_2	2	7	27
H_3	1	7	24
H_4	1	4	11
H_5	1	4	9
H_6	1	4	7
H_7	3	7	17
H_8	1	7	13

7.2.3 STOA I vs. STOA II

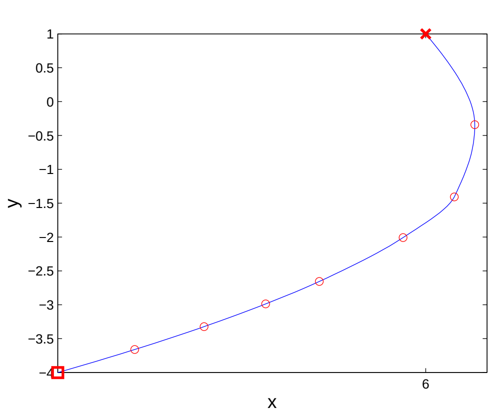
Both STOA I and STOA II have their own merits. STOA I will work for any user-specified time intervals ($\Delta\tau_{\mathrm{ro}}$), whereas the time intervals in STOA II are dyadic and fixed. On the other hand, STOA II takes full advantage of the multiresolution structure of the grid in the mesh refinement procedure. Most of the nodes in the grid for the new horizon are the nodes from the grid of the previous horizon. In STOA II most of the points of Grid_1^i consist of the points belonging to $\mathsf{Grid}^{i-} \subset \mathsf{Grid}^{i-1}$, for which the solution is already known. Hence, none of the previously computed information is lost while going from one horizon to the next. Therefore, in order to provide an initial guess $\mathcal{X}^i$ for starting the NLP solver on horizon H_i, the function values only at few additional points in the vicinity of the current time need to be interpolated from the solution found on the grid Grid^{i-1} during the previous horizon H_{i-1}. Moreover, in STOA I the algorithm always begins to iterate from the coarsest grid $\mathcal{V}_{J_{\min}}$. In STOA II, since most of the points of Grid_1^i consist of the points belonging to Grid^{i-}, the algorithm need not necessarily start from the coarsest grid, and in fact Grid_1^i may have nodes from finer scales resulting in faster convergence.

For both STOA I and STOA II, if the path constraints and the terminal constraints do not change drastically, the algorithm for each successive horizon converges pretty fast since the solution of the previous horizon is provided as an initial guess for solving the NLP problem on the current horizon. The CPU times achieved using the current implementation show the merits of the proposed algorithms in terms of speed. We should mention at this point that since all the computations presented in this chapter were carried out in

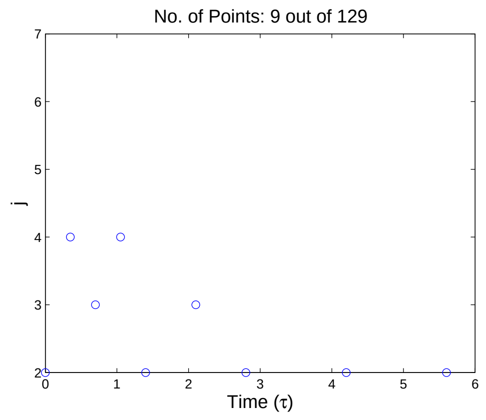
MATLAB, the reported CPU times can be significantly reduced by coding the algorithms in C or FORTRAN.

7.3 Summary

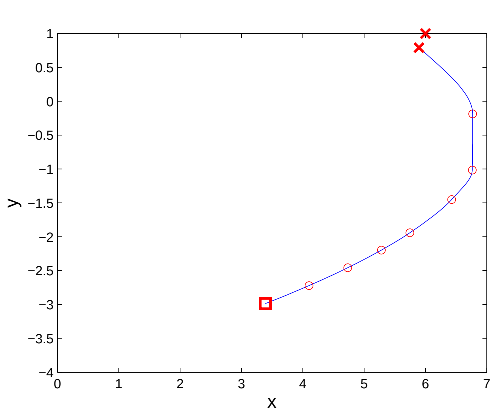
In this chapter, we have proposed two sequential trajectory optimization schemes to solve optimal control problems with moving targets and/or under dynamically changing environments in a fast and efficient way. The proposed algorithms autonomously discretize the trajectory with more nodes near the current time (not necessarily uniformly placed) while using a coarser grid for the rest of the trajectory in order to capture the overall trend. Moreover, if the states or the controls are irregular at a certain future time, the mesh is further refined automatically at those locations as well. The final grid point distributions for all the horizons and for both the examples considered in this chapter confirm these observations. Given their simplicity and efficiency, the proposed techniques offer a potential for online implementation for solving problems with moving targets and dynamically changing environments.



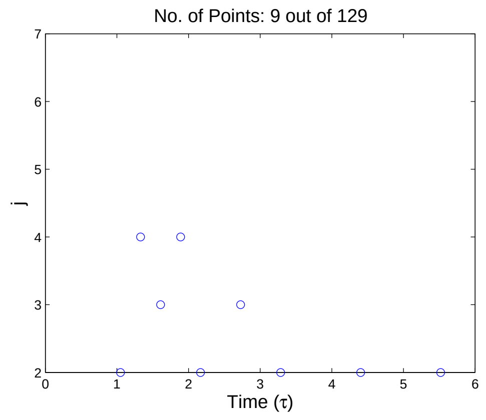
(a) Horizon 1. Trajectory.



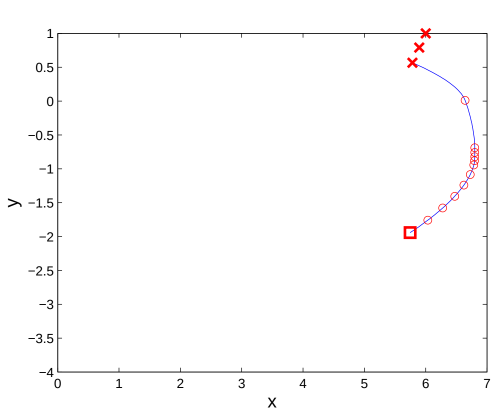
(b) Horizon 1. Grid point distribution.



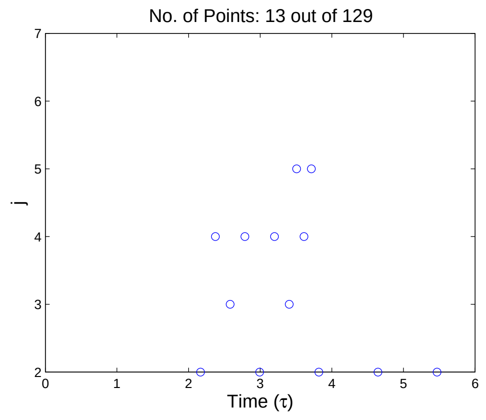
(c) Horizon 2. Trajectory.



(d) Horizon 2. Grid point distribution.

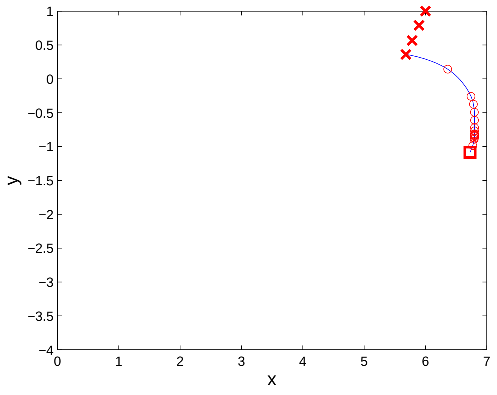


(e) Horizon 3. Trajectory.

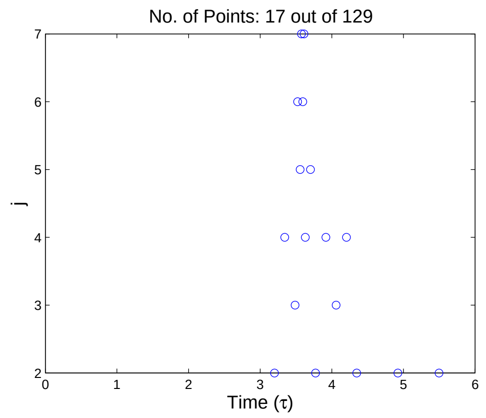


(f) Horizon 3. Grid point distribution.

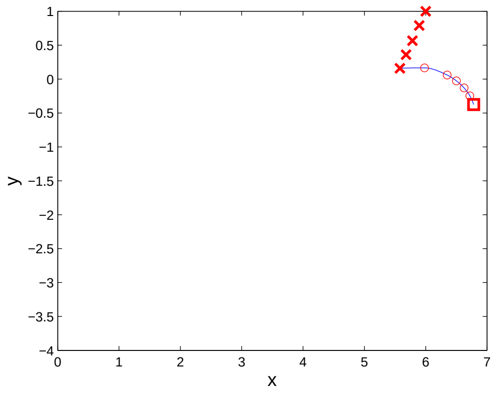
Figure 39: Example 18 (Target snapshots). Trajectory along with the grid point distributions for horizons 1, 2, and 3.



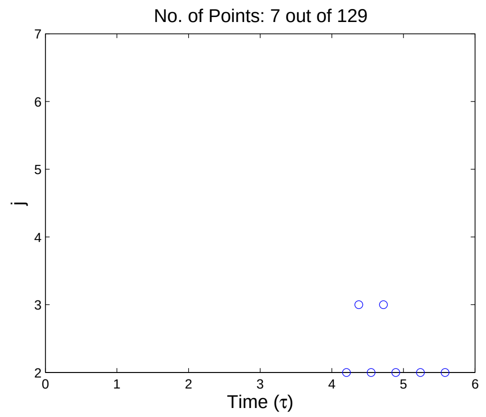
(a) Horizon 4. Trajectory.



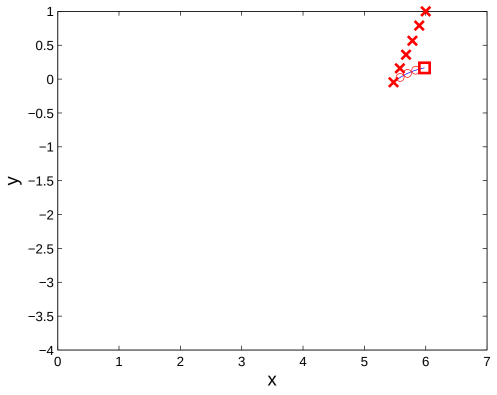
(b) Horizon 4. Grid point distribution.



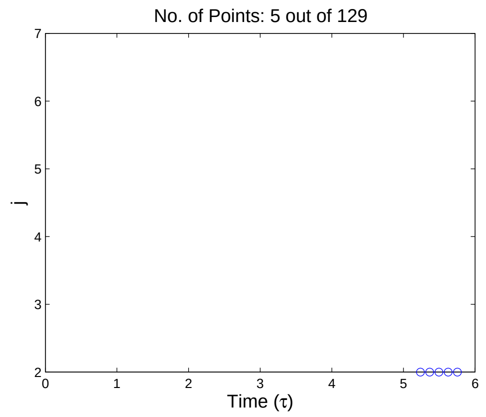
(c) Horizon 5. Trajectory.



(d) Horizon 5. Grid point distribution.

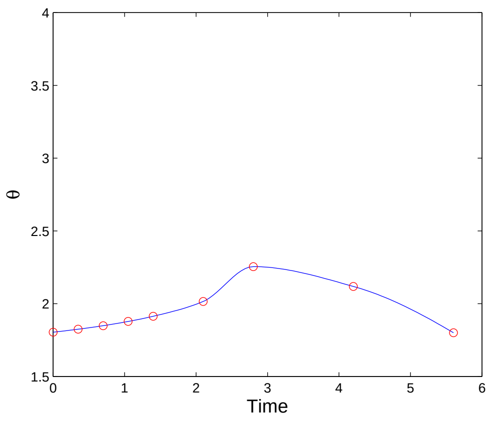


(e) Horizon 6. Trajectory.

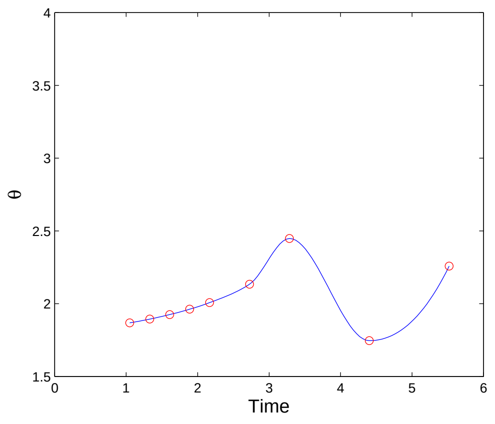


(f) Horizon 6. Grid point distribution.

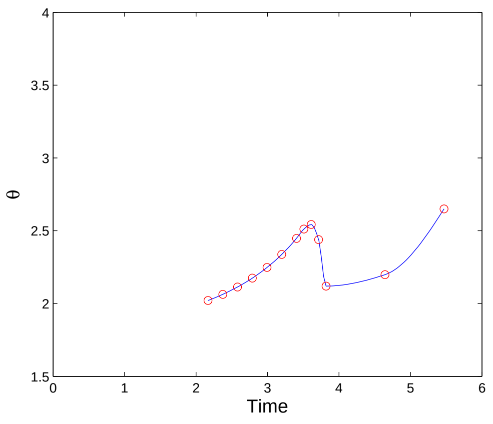
Figure 40: Example 18 (Target snapshots). Trajectory along with the grid point distributions for horizons 4, 5, and 6.



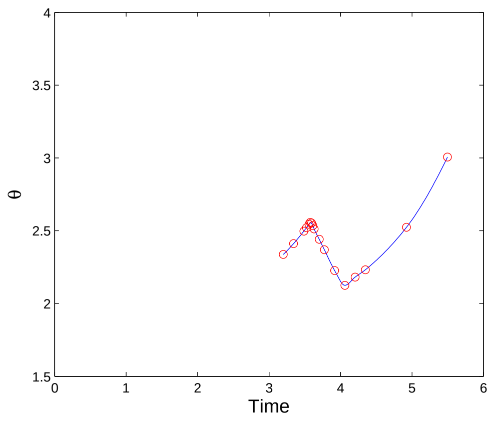
(a) Horizon 1.



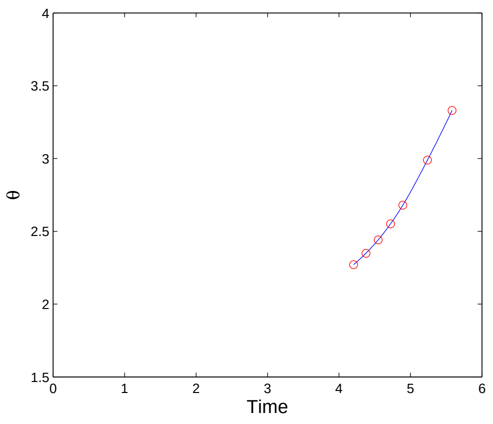
(b) Horizon 2.



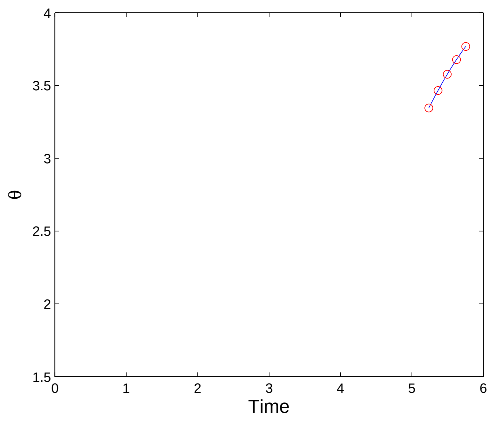
(c) Horizon 3.



(d) Horizon 4.

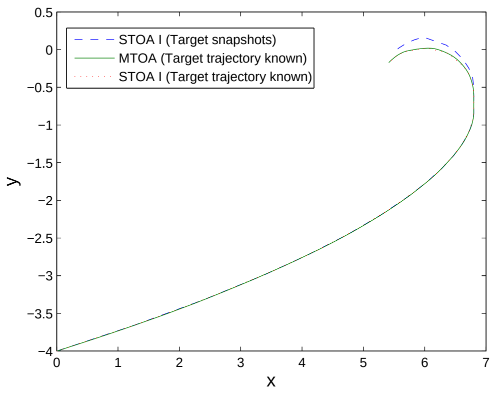


(e) Horizon 5.

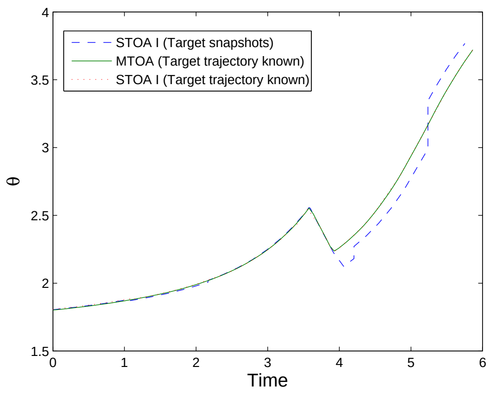


(f) Horizon 6.

Figure 41: Example 18 (Target snapshots). Time history of control θ for all horizons.

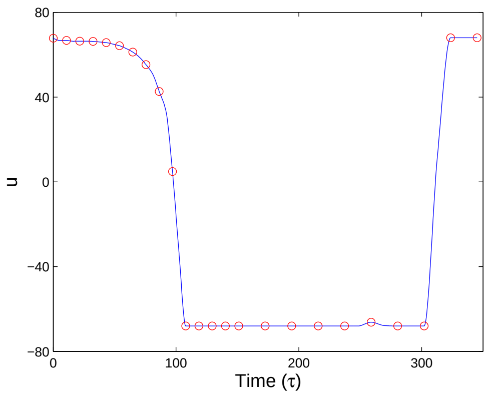


(a) Trajectory.

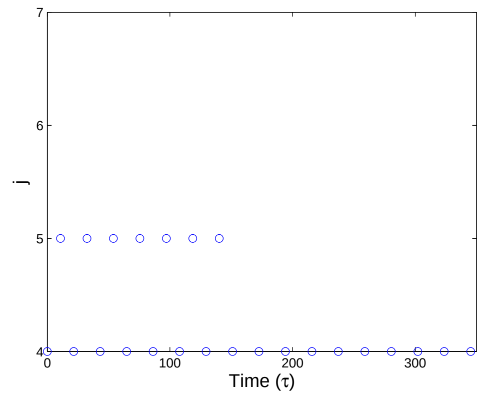


(b) Time history of control θ.

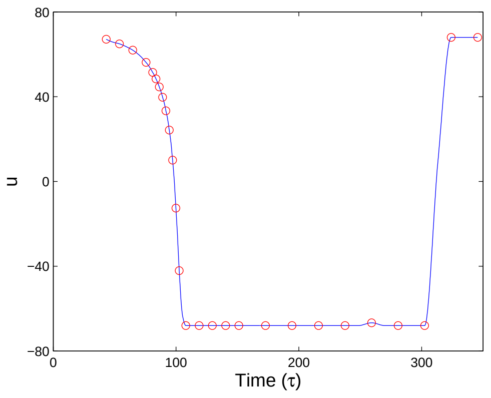
Figure 42: Example 18. Trajectory along with the time history of the control θ using three different multiresolution strategies.



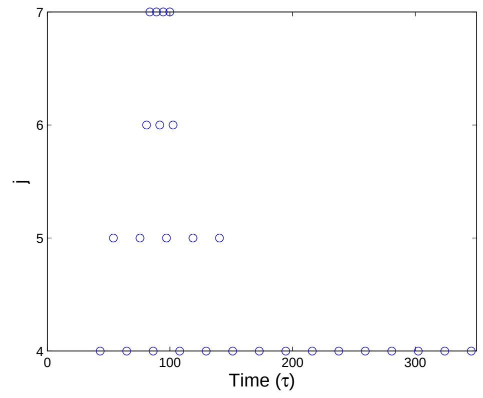
(a) Horizon 1. Control u.



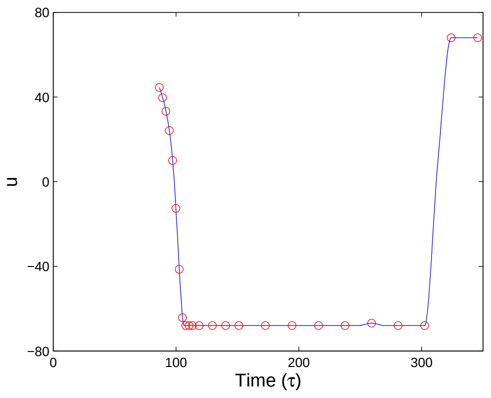
(b) Horizon 1. Grid point distribution.



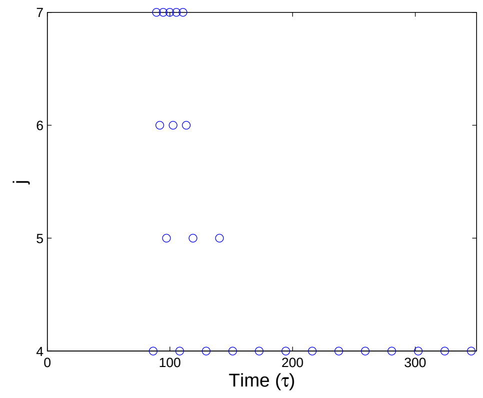
(c) Horizon 2. Control u.



(d) Horizon 2. Grid point distribution.

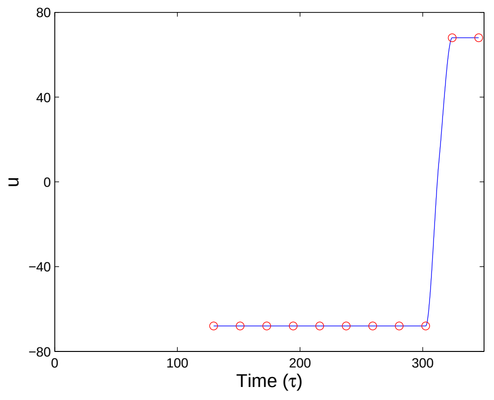


(e) Horizon 3. Control u.

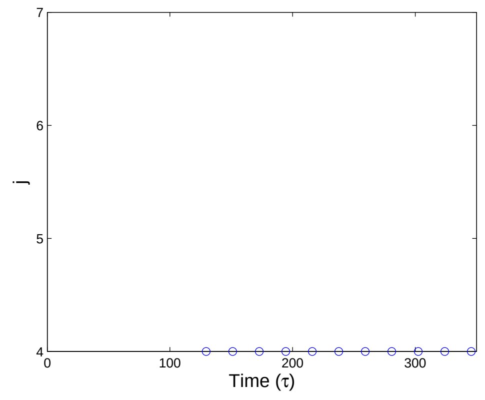


(f) Horizon 3. Grid point distribution.

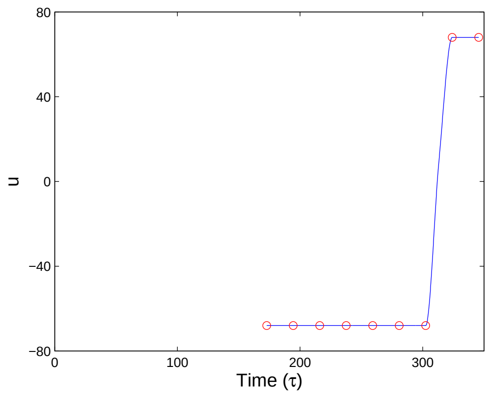
Figure 43: Example 19. Control time history and grid point distributions for horizons 1, 2, and 3.



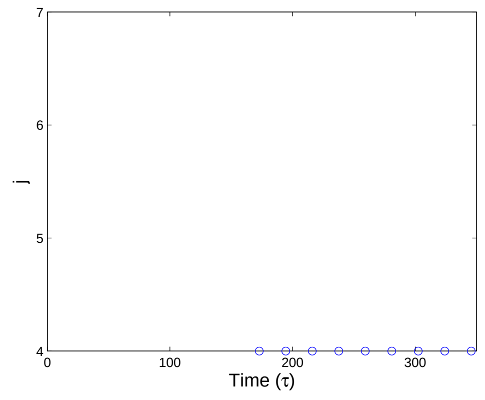
(a) Horizon 4. Control u.



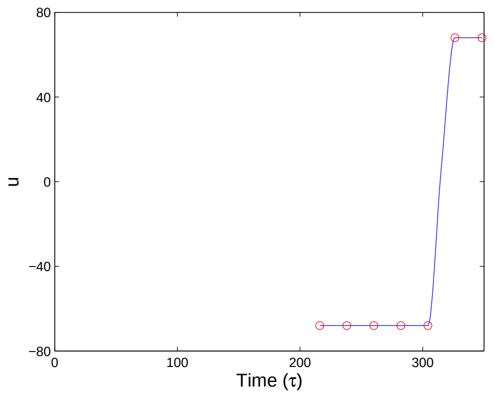
(b) Horizon 4. Grid point distribution.



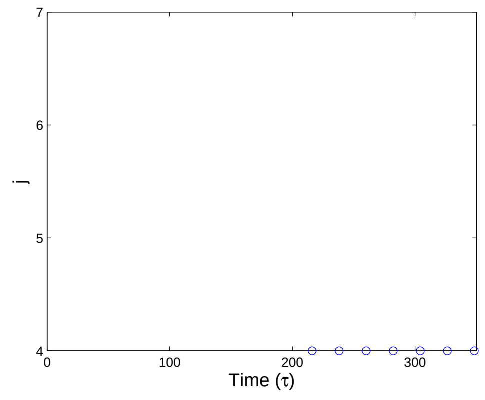
(c) Horizon 5. Control u.



(d) Horizon 5. Grid point distribution.

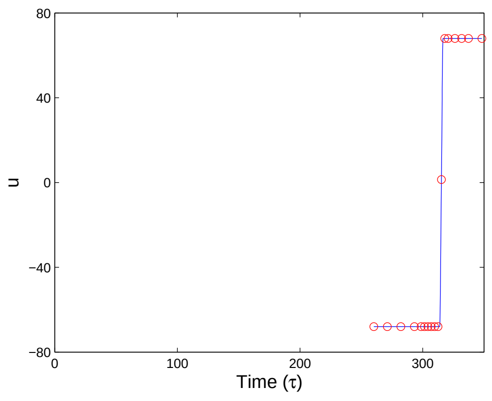


(e) Horizon 6. Control u.

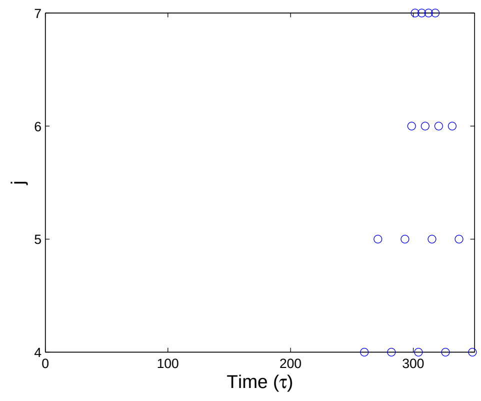


(f) Horizon 6. Grid point distribution.

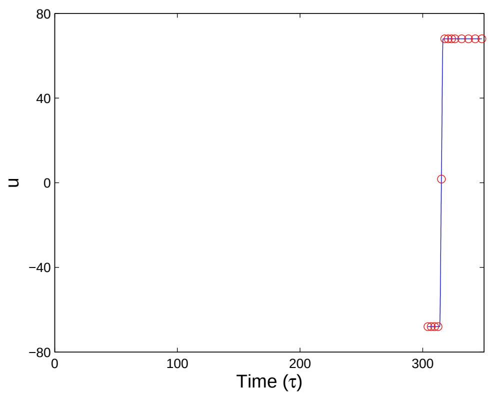
Figure 44: Example 19. Control time history and grid point distributions for horizons 4, 5, and 6.



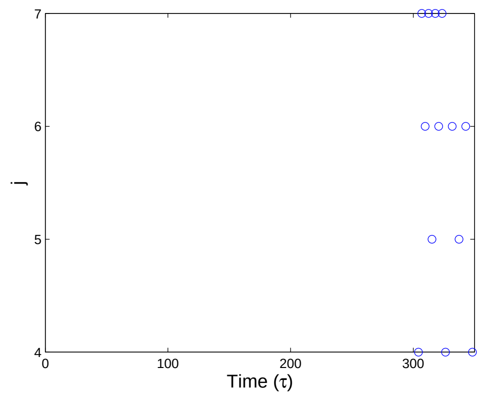
(a) Horizon 7. Control u.



(b) Horizon 7. Grid point distribution.



(c) Horizon 8. Control u.



(d) Horizon 8. Grid point distribution.

Figure 45: Example 19. Control time history and grid point distributions for horizons 7 and 8.

CHAPTER VIII

THESIS CONTRIBUTION, CONCLUSIONS, AND FUTURE WORK

8.1 Conclusions and Contributions

8.1.1 Hierarchical Multiresolution Adaptive Mesh Refinement for the Solution of Evolution PDEs

It is well-known that the solution of evolution partial differential equations is often not smooth even if the initial data are smooth. For instance, shocks may develop in hyperbolic conservation laws and kinks in Hamilton-Jacobi equations. To capture discontinuities and irregularities in the solution with high accuracy one needs to use a fine resolution grid. The use of a uniformly fine grid requires a large amount of computational resources in terms of both CPU time and memory. Hence, in order to solve evolution equations in a computationally efficient manner, several adaptive gridding techniques for solving partial differential equations have been proposed in the literature. Currently, popular adaptive methods for solving PDEs are: (i) moving mesh methods [1, 2, 4, 7, 8, 38, 39, 50, 91, 98, 105, 106, 136], in which an equation is derived that moves a grid of a fixed number of finite difference cells or finite elements so as to follow and resolve any local irregularities in the solution; (ii) the so called "adaptive mesh refinement" method [10, 13, 14, 15], in which the mesh is refined locally based on the difference between the solutions computed on the coarser and the finer grids, and (iii) wavelet-based or multiresolution-based methods [3, 16, 68, 69, 75, 76, 87, 139, 140, 141], which take advantage of the fact that functions with localized regions of sharp transition can be very well compressed. Our proposed method falls under this latter category.

Recently, Alves et al. [3] proposed an adaptive multiresolution scheme, similar to the multiresolution approach proposed by Harten [68, 69] and Holmstrom [75] for solving hyperbolic PDEs. These approaches share similar underlying ideas. Namely, the first step is to interpolate the function values at the points belonging to a particular resolution level, from the corresponding points at the coarser level, and find the interpolative error at the points of that particular resolution level. Once this step has been performed for all resolution levels, all the points that have an interpolative error greater than a prescribed threshold are added to the grid, along with their neighboring points at the same level and the neighboring points at the next finer level. The main difference between these approaches is that in Harten's approach [68, 69], the solution for each time step is represented on the finest grid and one calculates the interpolative errors at all the points of the finest grid at each mesh refinement step. On the other hand, Holmstrom [75] and Alves et al. [3], compute the interpolative error only at the points that are in the adaptive grid. If a value that does not exist is needed, Holmstrom interpolates the required function value recursively from a coarser scale. Alternatively, Alves et al. [3] add to the grid the points that were used to predict the function values at all previously added points, in order to compute the interpolative error during the next mesh adaptation.

In this work and Ref. [85, 86], we have proposed a novel multiresolution scheme for data compression, which results in a higher compression rate compared to the multiresolution approach by Harten [68, 69, 70] for the same desired accuracy. Subsequently, we developed an encoding scheme to solve initial-boundary value problems (IBVP) encountered in evolution PDEs. The proposed multiresolution scheme for data compression works with any of the interpolation techniques mentioned in Ref. [70]. One of the key features of our algorithm is that it is a "top-down" (from coarse to fine scale) approach, and we use the most recently updated information to make predictions. Moreover, our interpolations are not restricted to the use of only the retained points at the coarser level, but also use the retained points at the same level (and even the next finer level in the case of solving PDEs). This allows for a more accurate interpolation which, in turn, leads to fewer points in the final grid. In the proposed algorithm, we continuously keep on updating the grid as we go from the coarsest level to finer levels. If the interpolative error at a point that belongs to a particular level is greater than the prescribed threshold, we add that point to the grid. In the case of solving PDEs, we also add the neighboring points at the same level and the neighboring points at the next level to the grid. We make use of the fact that the point at which the interpolative error is greater than a prescribed threshold, this point is added to the grid and, in addition, it can be used to predict the remaining points at the same level and the levels below it. Moreover, for refining the mesh for solving evolution PDEs we predict the function value at a particular point only from the points that are already present in the grid, hence we avoid recursive interpolations from the coarser scales as is done by Holmstrom [75]. At the same time we do not need to add any extra points to the grid that are required just for computing the interpolative errors at the next mesh refinement step, as is done by Alves et al. [3].

Several examples have demonstrated the stability and robustness of the proposed algorithm. In all examples considered, the algorithm adapted dynamically to any existing or emerging irregularities in the solution, by automatically allocating more grid points to the region where the solution exhibited sharp features and fewer points to the region where the solution was smooth. As a result, the computational time and memory usage can be reduced significantly, while maintaining an accuracy equivalent to the one obtained using a fine uniform mesh. We observed speed-up factors of up to 64 (for $J_{\max} = 12$) when compared to the uniform mesh implementation. We also found that the speed-up factors increased at an approximate rate of 2 with the increase in the resolution level. At the same time, we have observed savings of up to 43% in terms of the number of grid points and a gain of about 27% in terms of speed-up factors compared to the approach of Alves et al. [3].

8.1.2 Trajectory Optimization Using Multiresolution Techniques

In this work and Ref. [79, 82], we have proposed a novel multiresolution-based approach for solving optimal control problems. As mentioned before, the solution of general (realistic) trajectory optimization problems is a challenging task. Analytical solutions are seldom available or even possible. In all numerical methods for the solution of trajectory optimization problems one needs to compromise between accuracy of the solution, robustness in terms of convergence, and execution speed. The use of a high resolution (dense) grid to accurately capture any discontinuities or switchings in the state or control variables requires a large amount of computational resources both in terms of CPU time and memory. Moreover, a large grid results in a large number of the variables to optimize, which in turn, can lead to ill-conditioning. MTOA automatically and inexpensively generates a grid that reduces the discretization error with each iteration. As a result, one is able to capture the solution accurately and efficiently using only a few nodes. The algorithm can handle state constraints, control constraints, and mixed constraints with ease. All the transition points in the solution (for example, bang-bang subarcs, or entry and exit points associated with state or mixed constraints) are captured with high accuracy. The convergence of the algorithm can be enhanced by initializing the algorithm on a coarse grid having a small number of variables. Once a converged solution is attained, the grid can be further refined by increasing the accuracy locally, only at the vicinity of those points that cannot be accurately interpolated by neighboring points in the grid. The methodology thus provides a compromise between robustness with respect to initial guesses, intermediate and final solution accuracy, and execution speed. These observations are supported by several numerical examples of challenging trajectory optimization problems.

Compared to prior similar results in this area [22, 20, 118, 63, 24, 27, 121] the algorithm proposed in this thesis has several advantages. First, we avoid the solution of a secondary optimization problem for adding points to the mesh as in Ref. [18, 20, 22]. Only simple interpolations are needed to refine the mesh, which can be done on the fly. Furthermore, our algorithm does not involve any integrations, as opposed to the highly accurate integrations (Romberg quadratures) required in the method by Betts et al. [20], which again can be computationally expensive for nonlinear dynamics. Finally, our algorithm is capable of not only adding points to the grid but also removing points from the grid when and where is needed. Moreover, both the operations of adding and removing points can be done in a single step. In the pseudospectral knotting method of Ross et al. [118, 63], one needs to know a priori the approximate number and location of singularities in the solution. These may not be known beforehand for most problems. The number of nodes to be added to a particular phase must be defined by the user before starting the algorithm. In our algorithm the user need not know a priori the number nor the locations of the irregularities in the solution. The algorithm will automatically detect the regions in the solution that are nonsmooth and it will add points accordingly when and where is needed. Furthermore, the nonuniform grids of pseudospectral methods result in grid distributions that remain fixed for each phase, since the location of the nodes are dictated by the zeros of the first derivative of the Legendre or Chebyshev polynomials, irrespective of the location of the soft knots [118]. Our algorithm uses a grid that is fully adaptive, embracing any form depending on the irregularities in the solution. This provides more flexibility in capturing any irregularities in the solution.

From all previous references in this area the work of Binder et al. [24, 27], and Schlegel et al. [121] are the closest – at least in spirit – to the approach proposed in the current work. These references use wavelet-based ideas to locate possible singularities in the solution and refine the grid locally. However, since these references work solely in the wavelet domain, they may lead to an increase of the overall computational overhead, as one needs to transform back and forth between the physical and wavelet domain. We avoid this issue altogether by always working in the physical domain. Nonetheless, by working with dyadic grids we still take advantage of the major advantage of the wavelet-based analysis, that is, multiresolution functional representations [70, 103].

8.1.3 Multiresolution Trajectory Optimization Schemes for Problems with Moving Targets and/or Dynamically Changing Environments

Next, we move on to the optimal control problems with moving targets and/or dynamically changing environments. A common line of attack for solving nonlinear trajectory optimization problems in real time [125, 100, 88, 144] is to break the problem into two phases: an offline phase and an online phase. The offline phase consists of solving the optimal control problem for various reference trajectories and storing these reference trajectories onboard for later online use. These reference trajectories are used to compute the actual trajectory online via a neighboring optimal feedback control strategy [31, 92, 130, 33] typically based on the linearized dynamics. This approach requires extensive ground-based analysis and onboard storage capabilities [94]. Moreover, perturbations around the reference trajectories might not be small, and therefore applying the linearized equations may not be appropriate.

In order to overcome the above mentioned problems, Kumar and Seywald [94] proposed to solve the nonlinear trajectory optimization problem online for the whole time interval, but with high accuracy only near the current time. Kumar and Seywald [94] proposed a dense-sparse discretization technique in which the trajectory is discretized by placing N_D dense nodes close to the current time and N_S sparse nodes for the rest of the trajectory. The state values at some future node are accepted as optimal and are prescribed as the initial conditions for the rest of the trajectory. The remainder of the trajectory is again discretized using a dense-sparse discretization technique, and the whole process is repeated again. The algorithm can be stopped by using any adhoc scheme, for example, it can be terminated when the density of the dense nodes is less than or equal to the density of the sparse nodes. Ross et al. [119] also proposed a similar scheme by solving the discretized NLP problem on a grid with a certain number of nodes and then propagate the solution from the prescribed initial condition by integrating the dynamics of the system for a specified interval of time. The values of the integrated states at the end of the integration interval are taken as the initial condition for solving the NLP problem for the rest of the trajectory, again on a grid with a fixed number of nodes. The whole process is repeated until the terminal conditions are met.

In this thesis and Ref. [83, 78], we have developed two sequential trajectory optimization schemes that autonomously discretize the trajectory with more nodes (finer grid) near the current time (not necessarily uniformly placed) and use fewer nodes (coarser grid) for the rest of the trajectory, the latter to capture the overall trend. Furthermore, if the states or controls are irregular in the vicinity of the current time, the algorithm will automatically further refine the mesh in this region to capture the irregularities in the solution more accurately. The generated grid is fully adaptive and can embrace any form depending on the solution. Given their simplicity and efficiency, the proposed techniques offer a potential for online implementation for solving problems with moving targets and dynamically changing environments. However, we would like to point that the neighboring optimal feedback control strategy is more robust compared to solving the nonlinear programming problem for trajectory generation.

8.2 Future Work

This work lays the foundation for solving optimal control problems using multiresolution techniques in a fast and efficient way. Apart from what has been done in this dissertation, there are many directions and unsolved problems which are worth investigating in the future.

8.2.1 Mesh Refinement

The adaptive grid generated in the proposed multiresolution-based algorithm for data compression and the solution of evolution PDEs, MTOA, STOA I, and STOA II depends on how we select points along the grid, that is, whether we move from left to right or from right to left across each level. It also depends on the location of the singularity. If the singularity is located in the middle, then it does not matter whether we move from left to right or from right to left. The result will be the same nonuniform grid. If on the other hand, the singularity is not in the middle, then the grid depends on the way in which we traverse across each level. This suggests that by using a suitable probability distribution function to choose the order in which the points at each particular level are selected, one may be able to further optimize the grid.

The threshold ϵ in the proposed multiresolution-based algorithm for data compression and the solution of evolution PDEs, MTOA, STOA I, and STOA II is level independent. During the course of this work, it was observed that the interpolative error coefficients decrease with the increase in the resolution level. The reduction in the interpolation error is because of the decrease in the distance between the interpolating points as we go to finer and finer levels. Hence, the future work should investigate the possible use of level dependent threshold for solving both evolution PDEs and optimal control problems, which will again help in optimizing the grid further.

8.2.2 Multiresolution Mesh Refinement for the Solution of Evolution PDEs

Follow-up work should concentrate on extending the proposed multiresolution approach for solving evolution PDEs to multiple dimensions. It is expected that the savings in terms of CPU time and the number of grid points observed for the single spatial dimension case will be greater in multiple dimensions. One approach in this direction is to work directly with interpolating functions in higher dimensions and then follow the same approach as for the one-dimensional case. That is, use the error between the actual and interpolated values from neighboring points to determine which points to retain in the grid and which to remove. The challenge is to find a consistent way of selecting neighboring points. Another idea is to proceed in a dimension by dimension fashion. That is, to compute the interpolative error coefficients at a particular point using an interpolation operator based on function values in the intermediate grid along one direction while keeping the other coordinates fixed and repeating the same for all directions.

In the proposed multiresolution mesh refinement scheme for the solution of evolution PDEs, the values of the parameters N_1 and N_2 are considered to be constant across the spatial as well as temporal domain. Future work should focus on the adaptation of these parameters in order to further optimize the grid.

In this work, Δt_n is computed based on the Courant-Friedrichs-Levy (CFL) condition [137] for hyperbolic equations and the von Neumann condition [137] for all other evolution equations. For both CFL condition and the von Neumann condition Δt_n depends on $\Delta x_{\min}$. Hence, in the proposed algorithm Δt_n changes adaptively depending on $\Delta x_{\min}$, which also changes adaptively. Therefore, a potential extension of this work is to incorporate different values of Δt_n for different grid levels which might further speed up the computations.

8.2.3 Multiresolution Trajectory Optimization Algorithm

A preliminary error analysis shows that the effect of the proposed multiresolution scheme is somewhat akin to a local control of the tolerance of the Runge-Kutta integration error. The error analysis also provides guidelines on how certain parameters needed in the algorithm (e.g., the order of the interpolating polynomials, the maximum/minimum time steps, etc) can be chosen for its correct implementation and to yield consistent approximations. Future work should focus on more quantitative measures for the selection of these parameters, and well as on providing explicit error bounds both for the unconstraint case as well as for more general cases that include path constraints.

Another problem of interest is to investigate the possible use of different grids for different variables. In the current implementation of MTOA, the grid for all the states and controls is the same even if the refinement is done based only on controls. The use of different grids for different variables should have benefits in terms of the optimality of the grid and hence should speed up the computations. But at the same time, the use of different grids for different variables might affect the sparsity of the NLP problem being solved which plays a crucial role in solving a NLP problem.

8.2.4 Applications of Sequential Trajectory Optimization Algorithms

There are several applications in which the proposed Sequential Trajectory Optimization Algorithms might prove advantageous and are worth investigating, for example, aircraft emergency landing and low thrust trajectory generation.

In an aircraft, once an emergency condition arises, effective generation of a safe trajectory (and then following this trajectory) becomes crucial to a safe landing. From the pilots point of view, emergency trajectory generation is defined as the determination of a course of action with sufficient detail to describe immediate aircraft dynamic states and required control activities to ensure a safe landing. Emergency trajectory generation requires a high level of detail in the near-term and a long time-scale to avoid generating a trajectory that is later found to be lacking. Generation of a detailed emergency trajectory can therefore be viewed as a task that may prevent problems such as taking too long to land (important in smoke and fire situations) or requiring extreme maneuvers to intercept the localizer and glideslope (important in situations with degraded aircraft stability and maneuverability).

Another example is the low thrust trajectory generation. Constructing the trajectory for a spacecraft as it transfers from a low earth orbit to a mission orbit is characterized by large time scales. Since the thrust applied to the vehicle is small in comparison to the weight of the spacecraft, the duration of the trajectroy can be very long. If a problem is solved from the initial time to the final time in one go, the resulting NLP problem might go substantially large to meet reasonable accuracy requirements. Lot of computational resources might be required for solving such a problem with high accuracy. Proposed sequential trajectory optimization algorithms might prove advantageous in solving such problems by solving several small scale problems with high accuracy only near the current time.

APPENDIX A

NUMERICAL ANALYSIS

A.1 Polynomials and Interpolation

The general form of an n-th degree polynomial is

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \tag{342}$$

where n denotes the degree of the polynomial and a_0 to a_n are constant coefficients. There are $n+1$ coefficients, so $n+1$ discrete data points are required to obtain unique values for the coefficients.

Definition 7 (Interpolation)**.** An interpolating approximation to a function $f(x)$ is an expression $P_n(x)$, whose $n+1$ degrees of freedom are determined by the requirement that the "interpolant" agrees with $f(x)$ at each point of a set of $n+1$ interpolation points,

$$P_n(x_i) = f(x_i), \qquad i = 0, 1, 2, \ldots, n. \tag{343}$$

When a polynomial of degree n, $P_n(x)$, is fit exactly to a set of $n+1$ discrete data points $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$, the polynomial has no error at the data points themselves. However, at the locations between the data points, there is an error which is defined by

$$\text{Error}(x) = |f(x) - P_n(x)|. \tag{344}$$

It is shown later in Appendix A.1.1 that if f is sufficiently smooth (i.e., is continuously differentiable at least $n+1$ times) in the interval $[x_0, x_n]$ then the error term is given by

$$\text{Error}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \Pi_{i=0}^{n} (x - x_i), \tag{345}$$

where $x_0 \leq \xi \leq x_n$.

Next, we briefly describe the polynomials and the interpolation techniques used in this work for interpolating a given data set $(x_i, f(x_i))$, for $i = 0, 1, \ldots, n$.

A.1.1 Divided Difference Polynomials

A *divided difference* is defined as the ratio of the difference in the function values at two points divided by the difference in the values of the corresponding independent variable. Thus, the first divided at point i is defined as

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}. \tag{346}$$

The second divided difference is defined as

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}. \tag{347}$$

Similar expansions can be obtained for divided differences of any order. Also note that

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{f(x_i) - f(x_{i+1})}{x_i - x_{i+1}} = f[x_{i+1}, x_i]. \tag{348}$$

Approximating polynomials for nonequally spaced data can be constructed using divided differences. Let $(x_0, f(x_0)),\ (x_1, f(x_1)), \ldots,\ (x_n, f(x_n))$ be the given $n+1$ points. Then the divided difference of orders $1, 2, \ldots, n$ are defined by the relations

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}, \tag{349}$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}, \tag{350}$$

$$\vdots$$

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0}. \tag{351}$$

By definition, $f[x_i] = f(x_i)$, for $i = 0, \ldots, n$. Furthermore,

$$f[x, x_0] = \frac{f(x) - f(x_0)}{x - x_0}, \tag{352}$$

therefore

$$f(x) = f(x_0) + (x - x_0) f[x, x_0]. \tag{353}$$

Again from the definition of divided differences we have

$$f[x, x_0, x_1] = \frac{f[x, x_0] - f[x_0, x_1]}{x - x_1}, \tag{354}$$

which gives

$$f[x,x_0] = f[x_0,x_1] + (x-x_1)f[x,x_0,x_1]. \tag{355}$$

Substituting the value of $f[x,x_0]$ from (355) in (353) we get

$$f(x) = f(x_0) + (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x,x_0,x_1]. \tag{356}$$

Similarly

$$f[x,x_0,x_1,x_2] = \frac{f[x,x_0,x_1] - f[x_0,x_1,x_2]}{x-x_2}, \tag{357}$$

and therefore

$$f[x,x_0,x_1] = f[x_0,x_1,x_2] + (x-x_2)f[x,x_0,x_1,x_2]. \tag{358}$$

Substituting the value of $f[x,x_0,x_1]$ in (356) we get

$$f(x) = f(x_0) \;+\; (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x_1,x_2] \tag{359}$$

$$+\; (x-x_0)(x-x_1)(x-x_2)f[x,x_0,x_1,x_2]. \tag{360}$$

Proceeding in this way we obtain

$$\begin{aligned} f(x) = f(x_0) \;&+\; (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x_1,x_2] \\ &+\; (x-x_0)(x-x_1)(x-x_2)f[x_0,x_1,x_2,x_3] + \cdots \\ &+\; (x-x_0)(x-x_1)\cdots(x-x_n)f[x,x_0,x_1,\cdots,x_n]. \end{aligned} \tag{361}$$

This formula is called *Newton's form of interpolating polynomial.* Let

$$\begin{aligned} P_n(x) = f(x_0) \;&+\; (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x_1,x_2] \\ &+\; (x-x_0)(x-x_1)(x-x_2)f[x_0,x_1,x_2,x_3] + \cdots \\ &+\; (x-x_0)(x-x_1)\cdots(x-x_{n-1})f[x,x_0,x_1,\cdots,x_{n-1}]. \end{aligned} \tag{362}$$

Hence, the interpolating error is given by

$$|f(x) - P_n(x)| = f[x,x_0,x_1,\cdots,x_n]\Pi_{i=0}^{n}(x-x_i). \tag{363}$$

Moreover, if f is sufficiently smooth (i.e., is continuously differentiable at least $n+1$ times) in the interval $[x_0,x_n]$ then [45]

$$f[x,x_0,x_1,\cdots,x_n] = \frac{f^{(n+1)}(\xi)}{(n+1)!}, \qquad x_0 \le \xi \le x_{n+1}. \tag{364}$$

A.1.2 Lagrange Interpolating Polynomial

A Lagrange interpolating polynomial of degree n is given by

$$P_n(x) = \sum_{i=0}^{n} C_i(x) f(x_i), \tag{365}$$

where $C_i(x)$ are polynomials of degree n which satisfy the condition

$$C_i(x_j) = \delta_{ij}, \tag{366}$$

where δ_{ij} is the Kronecker δ-function and are defined as

$$C_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}. \tag{367}$$

The n factors for $(x - x_j)$ insure that $C_i(x)$ vanishes at all the interpolation points except x_i. As opposed to forward-difference polynomials, the interpolating points x_i for Lagrange interpolation can be evenly spaced or unevenly spaced.

A.1.3 Hermite's Interpolating Polynomial

The interpolating formulas, considered so far, make use of only function values. We now give an interpolation formula in which both the function and its first derivative values are to be assigned at each point of interpolation, that is,

$$\begin{aligned} P_{2n+1}(x_i) &= f_i, && (368) \\ P'_{2n+1}(x_i) &= f'_i, && (369) \end{aligned}$$

for $i = 0, 1, \ldots, n$. This is referred to as Hermite's interpolation formula and such a polynomial is given by

$$P_{2n+1}(x) = \sum_{i=0}^{n} U_i(x) f_i + \sum_{i=0}^{n} V_i(x) f'_i, \tag{370}$$

where

$$\begin{aligned} U_i(x) &= [1 - 2C'_i(x_i)(x - x_i)][C_i(x)]^2, && (371) \\ V_i(x) &= (x - x_i)[C_i(x)]^2, && (372) \end{aligned}$$

and $C_i(x)$ are given by (367).

A.1.4 Chebyshev Polynomial

Chebyshev polynomials are eigenfunctions of the following differential equation

$$(1-x^2)\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - x\frac{\mathrm{d}y}{\mathrm{d}x} + \lambda y = 0, \tag{373}$$

with eigenvalue $\lambda = n^2$. There are two solutions which are given as series by:

$$y_1(x) = 1 - \frac{n^2}{2!}x^2 + \frac{(n-2)n^2(n+2)}{4!}x^4 - \frac{(n-4)(n-2)n^2(n+2)(n+4)}{6!}x^6 + \dots \tag{374}$$

and

$$y_2(x) = x - \frac{(n-1)(n+1)}{3!}x^3 + \frac{(n-3)(n-1)(n+1)(n+3)}{5!}x^5 - \dots \tag{375}$$

When n is a non-negative integer, one of these series will terminate, giving a polynomial solution. If $n \geq 0$ is even, then the series for y_1 terminates at x^n. If n is odd, then the series for y_2 terminates at x^n. These polynomials are known as the *Chebyshev polynomials*. (In fact, polynomial solutions are also obtained when n is a negative integer, but these are not the new solutions, since the Chebyshev equation is invariant under the substitution of n by $-n$.)

Now, for $n = 0$,

$$y_1(x) = 1, \tag{376}$$

and if we take $x = \cos(t)$, then we have

$$y_1(\cos(t)) = 1 = \cos(0 \cdot t). \tag{377}$$

For $n = 1$,

$$y_2(x) = x, \tag{378}$$

and if we again take $x = \cos(t)$, then

$$y_2(\cos(t)) = \cos(t). \tag{379}$$

Similarly, for $n = 2$,

$$y_1(\cos(t)) = 1 - \frac{4}{2}\cos^2(t) = \cos(2t), \tag{380}$$

and so on. Hence, Chebyshev polynomial $P_n(x)$ is given by the following equation

$$P_n(\cos(t)) = \cos(nt), \qquad \forall \;\; n. \tag{381}$$

A.1.5 Legendre Polynomial

Legendre polynomials are the eigenfunctions of the following differential equation

$$(1-x^2)\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - 2x\frac{\mathrm{d}y}{\mathrm{d}x} + \lambda y = 0, \tag{382}$$

with an eigenvalue $\lambda = n(n+1)$. The above equation can be written as

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[(1-x^2)\frac{\mathrm{d}y}{\mathrm{d}x}\right] + n(n+1)y = 0. \tag{383}$$

The above differential equation again has two solutions which are given as series by:

$$y_1^n(x) = 1 - \frac{n(n+1)}{2!}x^2 + \frac{(n-2)n(n+1)(n+3)}{4!}x^4 \tag{384}$$

$$- \frac{(n-4)(n-2)n(n+1)(n+3)(n+5)}{6!}x^6 + \dots \tag{385}$$

$$y_2^n(x) = x - \frac{(n-1)(n+2)}{3!}x^3 + \frac{(n-3)(n-1)(n+2)(n+4)}{5!}x^5 - \dots \tag{386}$$

Hence, the general solution for an integer n is then given by the *Legendre polynomials*,

$$P_n(x) = c_n \begin{cases} y_1^n(x), & n \text{ even}, \\ y_2^n(x), & n \text{ odd}, \end{cases} \tag{387}$$

where c_n is chosen so as to yield the normalization $P_n(1) = 1$. Or, alternatively, we can write

$$P_n(x) = \begin{cases} y_1^n(x)/y_1^n(1), & n \text{ even}, \\ y_2^n(x)/y_2^n(1), & n \text{ odd}. \end{cases} \tag{388}$$

A.1.6 Neville's Algorithm

Neville's algorithm is equivalent to a Lagrange polynomial. It is based on a series of linear interpolations. The data do not have to be in monotonic order, or in any structured order. The main advantage of Neville's algorithm is that it can be very easily programmed for a computer. Moreover, it will be seen that none of the prior work must be redone, as it would have to be redone to evaluate Lagrange interpolating polynomials.

Given $(n+1)$ data points $(x_i, f(x_i))$, for $i = 0, \dots, n$, where the values of x need not necessarily be equally spaced, then to find the value of f corresponding to any given value of x we proceed iteratively as follows: obtain a first approximation to $f(x)$ by considering the first two points only; then obtain its second approximation by considering the first three points, and so on. We denote the different interpolation polynomials by $\Delta(x)$ (with suitable subscripts) so that at the first stage of approximation, we have

$$\Delta_{01}(x) = \frac{x - x_1}{x_0 - x_1} f_0 + \frac{x - x_0}{x_1 - x_0} f_1. \tag{389}$$

Similarly, we can form $\Delta_{12},\ \Delta_{23}, \ldots$.

Next, we form Δ_{012} by considering the first three points

$$\Delta_{012}(x) = \frac{x - x_2}{x_0 - x_2} \Delta_{01}(x) + \frac{x - x_0}{x_2 - x_0} \Delta_{12}(x). \tag{390}$$

In the same fashion, we can obtain $\Delta_{123},\ \Delta_{234}, \ldots$. Continuing this way, at the nth stage of approximation we obtain

$$\Delta_{012\ldots n}(x) = \frac{x - x_n}{x_0 - x_n} \Delta_{012\ldots n-1}(x) + \frac{x - x_0}{x_n - x_0} \Delta_{123\ldots n}(x). \tag{391}$$

The Neville's algorithm for a third degree interpolation has been summarized in Table 20.

x	f			
x_0	$f(x_0)$			
		$\Delta_{01}(x)$		
x_1	$f(x_1)$		$\Delta_{012}(x)$	
		$\Delta_{12}(x)$		$\Delta_{0123}(x)$
x_2	$f(x_2)$		$\Delta_{123}(x)$	
		$\Delta_{23}(x)$		
x_3	$f(x_3)$			

Table 20: Neville's algorithm for a third-degree interpolation.

Next, we briefly describe the Romberg integration algorithm.

A.2 Romberg Quadrature

When the functional form of the error of a numerical algorithm is known, the error can be estimated by evaluating the algorithm for two different increment sizes. The error estimate can be used both for error control and extrapolation.

Consider a numerical algorithm which approximates an exact calculation with an error that depends on the grid size h. Let us denote the exact calculation by f_{exact} and the approximation by $f(h)$, which also depends on the grid size h. Thus,

$$f_{\text{exact}} = f(h) + \text{Error}(h), \tag{392}$$

where

$$\text{Error}(h) = C_1 h^n + C_2 h^{n+m} + C_3 h^{n+2m} + \ldots, \tag{393}$$

n is the order of the leading error term and m is the increment in the order of the following error terms. Applying the algorithm at two increment sizes, $h_1 = h$ and $h_2 = h/R$, gives

$$f_{\text{exact}} = f(h) + C_1 h^n + \mathcal{O}(h^{n+m}). \tag{394}$$

$$f_{\text{exact}} = f(h/R) + C_1 \left(\frac{h}{R}\right)^n + \mathcal{O}(h^{n+m}). \tag{395}$$

Subtracting (395) from (394) gives

$$0 = f(h) - f(h/R) + C_1 h^n - C_1 \left(\frac{h}{R}\right)^n + \mathcal{O}(h^{n+m}). \tag{396}$$

Solving (396) for the leading error terms in (394) and (395) yields

$$\text{Error}(h) = C_1 h^n = \frac{R^n}{R^n - 1}(f(h/R) - f(h)), \tag{397}$$

$$\text{Error}(h/2) = C_1 (h/2)^n = \frac{1}{R^n - 1}(f(h/R) - f(h)). \tag{398}$$

The error estimates can be added to the approximate results to yield an improved approximation. This process is called *extrapolation.* Adding (398) to (395) gives

$$\text{Extrapolated value} = f(h/R) + \frac{1}{R^n - 1}(f(h/R) - f(h)) + \mathcal{O}(h^{n+m}). \tag{399}$$

The error of the extrapolated value is $\mathcal{O}(h^{n+m})$.

When extrapolation is applied to numerical integration by the trapezoidal rule where the successive increment size is one-half of the preceding increment size, that is, $R = 2$, the result is called *Romberg integration.*

The error of the composite trapezoidal rule has the form [74]

$$\text{Error}(h) = C_1 h^2 + C_2 h^4 + C_3 h^6 + \ldots. \tag{400}$$

Thus, the basic algorithm is $\mathcal{O}(h^2)$, so $n = 2$. The following error terms increase in order in increments of 2. Hence, the error estimation formula (398) becomes

$$\text{Error}(h/2) = \frac{1}{2^n - 1}(f(h/2) - f(h)). \tag{401}$$

For the trapezoidal rule itself, $n = 2$, and equation (401) becomes

$$\text{Error}(h/2) = \frac{1}{3}(I(h/2) - I(h)), \tag{402}$$

where $I(h)$ denotes the integral approximation using the trapezoidal rule with step size h. Applying the extrapolation formula (399) for $R = 2$ gives

$$I(h, h/2) = I(h/2) + \text{Error}(h/2) + \mathcal{O}(h^4). \tag{403}$$

Equation (403) shows that the result $I(h, h/2)$ obtained by extrapolating the $\mathcal{O}(h^2)$ trapezoidal rule is $\mathcal{O}(h^4)$.

If two extrapolated $\mathcal{O}(h^4)$ values are available $I(h, h/2)$, $I(h/2, h/4)$, which requires three $\mathcal{O}(h^2)$ trapezoidal rule results $I(h)$, $I(h/2)$, $I(h/4)$, those two values $I(h, h/2)$, $I(h/2, h/4)$ can be extrapolated to obtain an $\mathcal{O}(h^6)$ value $I(h, h/2, h/4)$ by applying (401) with $n = 4$ to estimate the $\mathcal{O}(h^4)$ error, and adding that error term to the more accurate $\mathcal{O}(h^4)$ value. Successively higher-order extrapolations can be performed until round-off error masks any further improvements. Each successive higher-order extrapolation begins with an additional application of the $\mathcal{O}(h^2)$ trapezoidal rule, which is then combined with the previously extrapolated values to obtain the next higher-order extrapolated result.

With the notation described above, the Romberg integration can be summarized in the tabular form as follows (Table 21).

Table 21: Romberg quadrature.

$I(h)$			
	$I\left(h, \frac{1}{2}h\right)$		
$I\left(\frac{1}{2}h\right)$		$I\left(h, \frac{1}{2}h, \frac{1}{4}h\right)$	
	$I\left(\frac{1}{2}h, \frac{1}{4}h\right)$		$I\left(h, \frac{1}{2}h, \frac{1}{4}h, \frac{1}{8}h\right)$
$I\left(\frac{1}{4}h\right)$		$I\left(\frac{1}{2}h, \frac{1}{4}h, \frac{1}{8}h\right)$	
	$I\left(\frac{1}{4}h, \frac{1}{8}h\right)$		
$I\left(\frac{1}{8}h\right)$			

A.3 Derivation of the Defects of Hermite-Simpson Discretization

The usage of Hermite-Simpson combination dates back at least to Kunz (1957) [95] and the collocation interpretation was provided by Weiss [142].

Consider for simplicity the scalar version of (222)

$$\dot{x} = \Delta\tau f(x(t), u(t), t), \tag{404}$$

where $t \in [t_{j_i,k_i}, t_{j_{i+1},k_{i+1}}] \subset \mathsf{G}$, where G be a non-uniform grid of the form (225). As before we denote, $x(t_{j_i,k_i})$ by x_{j_i,k_i}. Then we can write the state $x(t)$ on the segment $[t_{j_i,k_i}, t_{j_{i+1},k_{i+1}}]$, using the cubic Hermite interpolating polynomial (370), with $n = 1$, as

$$x(t) = U_i(t)x_{j_i,k_i} + U_{i+1}(t)x_{j_{i+1},k_{i+1}} + \Delta\tau V_i(t)f_{j_i,k_i} + \Delta\tau V_{i+1}(t)f_{j_{i+1},k_{i+1}}. \tag{405}$$

Therefore,

$$\begin{aligned} x_{j_{i+1/2},k_{i+1/2}} =& U_i(t_{j_{i+1/2},k_{i+1/2}})x_{j_i,k_i} + U_{i+1}(t_{j_{i+1/2},k_{i+1/2}})x_{j_{i+1},k_{i+1}} \\ &+ \Delta\tau V_i(t_{j_{i+1/2},k_{i+1/2}})f_{j_i,k_i} + \Delta\tau V_{i+1}(t_{j_{i+1/2},k_{i+1/2}})f_{j_{i+1},k_{i+1}}, \end{aligned} \tag{406}$$

where $x_{j_{i+1/2},k_{i+1/2}} = x(t_{j_{i+1/2},k_{i+1/2}})$ and

$$t_{j_{i+1/2},k_{i+1/2}} = \frac{t_{j_i,k_i} + t_{j_{i+1},k_{i+1}}}{2}. \tag{407}$$

For finding $U_i(t)$, $U_{i+1}(t)$, $V_i(t)$ and $V_{i+1}(t)$, we first find

$$C_i(t) = \frac{t - t_{j_{i+1},k_{i+1}}}{t_{j_i,k_i} - t_{j_{i+1},k_{i+1}}}, \tag{408}$$

$$C_i'(t) = \frac{1}{t_{j_i,k_i} - t_{j_{i+1},k_{i+1}}}, \tag{409}$$

$$C_{i+1}(t) = \frac{t - t_{j_i,k_i}}{t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}}, \tag{410}$$

$$C_{i+1}'(t) = \frac{1}{t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}}. \tag{411}$$

Hence,

$$\begin{aligned} U_i(t) &= [1 - 2C_i'(t_{j_i,k_i})(t - t_{j_i,k_i})][C_i(t)^2] \\ &= \left[1 - 2\frac{(t - t_{j_i,k_i})}{t_{j_i,k_i} - t_{j_{i+1},k_{i+1}}}\right]\left[\frac{t - t_{j_{i+1},k_{i+1}}}{t_{j_i,k_i} - t_{j_{i+1},k_{i+1}}}\right]^2 \\ &= (t_{j_i,k_i} - t_{j_{i+1},k_{i+1}} - 2t + 2t_{j_i,k_i}]\frac{(t - t_{j_{i+1},k_{i+1}})^2}{(t_{j_i,k_i} - t_{j_{i+1},k_{i+1}})^3} \\ &= (3t_{j_i,k_i} - t_{j_{i+1},k_{i+1}} - 2t)\frac{(t - t_{j_{i+1},k_{i+1}})^2}{(t_{j_i,k_i} - t_{j_{i+1},k_{i+1}})^3}. \end{aligned} \tag{412}$$

Therefore,

$$\begin{aligned}
U_i(t_{j_{i+1/2},k_{i+1/2}}) &= U_i(t_{j_i,k_i} + h_{j_i,k_i}/2) \\
&= -\left[3t_{j_i,k_i} - t_{j_{i+1},k_{i+1}} - 2\left(t_{j_i,k_i} + \frac{h_{j_i,k_i}}{2}\right)\right] \frac{\left(t_{j_i,k_i} + \frac{h_{j_i,k_i}}{2} - t_{j_{i+1},k_{i+1}}\right)^2}{h^3_{j_i,k_i}} \\
&= -(3t_{j_i,k_i} - t_{j_{i+1},k_{i+1}} - 2t_{j_i,k_i} - h_{j_i,k_i})\frac{h^2_{j_i,k_i}}{4h^3_{j_i,k_i}} \\
&= -\frac{1}{4h_{j_i,k_i}}(t_{j_i,k_i} - t_{j_{i+1},k_{i+1}} - h_{j_i,k_i}) \\
&= -\frac{1}{4h_{j_i,k_i}}(-h_{j_i,k_i} - h_{j_i,k_i}) \\
&= \frac{1}{2}. \qquad (413)
\end{aligned}$$

Next we find

$$\begin{aligned}
U_{i+1}(t) &= [1 - 2C'_{i+1}(t_{j_i,k_i})(t - t_{j_{i+1},k_{i+1}})][C_{i+1}(t)^2] \\
&= \left[1 - 2\frac{t - t_{j_{i+1},k_{i+1}}}{t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}}\right]\left[\frac{t - t_{j_i,k_i}}{t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}}\right]^2 \\
&= (t_{j_{i+1},k_{i+1}} - t_{j_i,k_i} - 2t + 2t_{j_{i+1},k_{i+1}})\frac{(t - t_{j_i,k_i})^2}{(t_{j_{i+1},k_{i+1}} - t_{j_i,k_i})^3}. \qquad (414)
\end{aligned}$$

Therefore,

$$\begin{aligned}
U_{i+1}(t_{j_{i+1/2},k_{i+1/2}}) &= U_{i+1}(t_{j_i,k_i} + h_{j_i,k_i}/2) \\
&= \frac{1}{h^3_{j_i,k_i}}\left[3t_{j_{i+1},k_{i+1}} - t_{j_i,k_i} - 2(t_{j_i,k_i} + \frac{h_{j_i,k_i}}{2})\right]\left(t_{j_i,k_i} + \frac{h_{j_i,k_i}}{2} - t_{j_i,k_i}\right)^2 \\
&= \frac{1}{h^3_{j_i,k_i}}(3t_{j_{i+1},k_{i+1}} - t_{j_i,k_i} - 2t_{j_i,k_i} - h_{j_i,k_i})\frac{h^2_{j_i,k_i}}{4} \\
&= \frac{1}{4h_{j_i,k_i}}(3h_{j_i,k_i} - h_{j_i,k_i}) \\
&= \frac{1}{2}. \qquad (415)
\end{aligned}$$

Next we find

$$\begin{aligned}
V_i(t) &= (t - t_{j_i,k_i})[C_i(t)]^2 \\
&= (t - t_{j_i,k_i})\left[\frac{t - t_{j_{i+1},k_{i+1}}}{t_{j_i,k_i} - t_{j_{i+1},k_{i+1}}}\right]^2. \qquad (416)
\end{aligned}$$

Therefore,

$$\begin{aligned} V_i(t_{j_{i+1/2},k_{i+1/2}}) &= V_i(t_{j_i,k_i} + h_{j_i,k_i}/2) \\ &= \frac{h_{j_i,k_i}}{2}\left[\frac{t_{j_i,k_i} + \frac{h_{j_i,k_i}}{2} - t_{j_{i+1},k_{i+1}}}{h_{j_i,k_i}}\right]^2 \\ &= \frac{h_{j_i,k_i}}{2}\left[\frac{-\frac{h_{j_i,k_i}}{2}}{h_{j_i,k_i}}\right]^2 \\ &= \frac{h_{j_i,k_i}}{8}. \end{aligned} \tag{417}$$

Next we find

$$\begin{aligned} V_{i+1}(t) &= (t - t_{j_{i+1},k_{i+1}})[C_{i+1}(t)]^2 \\ &= (t - t_{j_{i+1},k_{i+1}})\left[\frac{t - t_{j_i,k_i}}{t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}}\right]^2. \end{aligned} \tag{418}$$

Therefore,

$$\begin{aligned} V_{i+1}(t_{j_{i+1/2},k_{i+1/2}}) &= V_{i+1}(t_{j_i,k_i} + h_{j_i,k_i}/2) \\ &= \left(t_{j_i,k_i} + \frac{h_{j_i,k_i}}{2} - t_{j_{i+1},k_{i+1}}\right)\left(\frac{t_{j_i,k_i} + \frac{h_{j_i,k_i}}{2} - t_{j_i,k_i}}{h_{j_i,k_i}}\right)^2 \\ &= \frac{1}{4}\left(-h_{j_i,k_i} + \frac{h_{j_i,k_i}}{2}\right) \\ &= -\frac{h_{j_i,k_i}}{8}. \end{aligned} \tag{419}$$

Hence, (406) reduces to

$$\begin{aligned} x_{j_{i+1/2},k_{i+1/2}} &= \frac{1}{2}x_{j_i,k_i} + \frac{1}{2}x_{j_{i+1},k_{i+1}} + \Delta\tau\frac{h_{j_i,k_i}}{8}f_{j_i,k_i} - \Delta\tau\frac{h_{j_i,k_i}}{8}f_{j_{i+1},k_{i+1}} \\ &= \frac{1}{2}(x_{j_i,k_i} + x_{j_{i+1},k_{i+1}}) + \Delta\tau\frac{h_{j_i,k_i}}{8}(f_{j_i,k_i} - f_{j_{i+1},k_{i+1}}). \end{aligned} \tag{420}$$

Once we have found $x_{j_{i+1/2},k_{i+1/2}}$, we compute[1]

$$f_{j_{i+1/2},k_{i+1/2}} = f(x_{j_{i+1/2},k_{i+1/2}}, u_{j_{i+1/2},k_{i+1/2}}, t_{j_{i+1/2},k_{i+1/2}}), \tag{421}$$

where $u_{j_{i+1/2},k_{i+1/2}} = u(t_{j_{i+1/2},k_{i+1/2}})$, and integrate across the segment using Simpson's quadrature rule [74],

$$\int_{t_i}^{t_{i+1}} \dot{x}\mathrm{d}t = x_{j_{i+1},k_{i+1}} - x_{j_i,k_i} = \Delta\tau\frac{h_{j_i,k_i}/2}{3}(f_{j_i,k_i} + 4f_{j_{i+1/2},k_{i+1/2}} + f_{j_{i+1},k_{i+1}}). \tag{422}$$

[1]It should be noted that $u_{j_{i+1/2},k_{i+1/2}}$ is an optimization parameter (NLP variable), hence we do not calculate $u_{j_{i+1/2},k_{i+1/2}}$.

Hence, the defects of Hermite-Simpson discretization are given by

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - \Delta\tau \frac{h_{j_i,k_i}}{6}[\mathbf{f}_{j_i,k_i} + 4\mathbf{f}_{j_{i+1/2},k_{i+1/2}} + \mathbf{f}_{j_{i+1},k_{i+1}}], \tag{423}$$

where

$$\mathbf{f}_{j_i,k_i} = \mathbf{f}(\mathbf{x}_{j_i,k_i}, \mathbf{u}_{j_i,k_i}, t_{j_i,k_i}),$$

$$\mathbf{f}_{j_{i+1/2},k_{i+1/2}} = \mathbf{f}(\mathbf{x}_{j_{i+1/2},k_{i+1/2}}, \mathbf{u}_{j_{i+1/2},k_{i+1/2}}, t_{j_{i+1/2},k_{i+1/2}}),$$

$$\mathbf{x}_{j_{i+1/2},k_{i+1/2}} = \frac{1}{2}[\mathbf{x}_{j_i,k_i} + \mathbf{x}_{j_{i+1},k_{i+1}}] + \Delta\tau \frac{h_{j_i,k_i}}{8}[\mathbf{f}_{j_i,k_i} - \mathbf{f}_{j_{i+1},k_{i+1}}],$$

$$t_{j_{i+1/2},k_{i+1/2}} = \frac{t_{j_i,k_i} + t_{j_{i+1},k_{i+1}}}{2}, \quad \mathbf{u}_{j_{i+1/2},k_{i+1/2}} = \mathbf{u}(t_{j_{i+1/2},k_{i+1/2}}),$$

for $i = 1, \ldots, N-1$.

APPENDIX B

ARZELO-ASCOLI COMPACTNESS CRITERION

Definition 8 (Uniformly Equicontinuous Functions)**.** Let $\{f_k\}_{k=1}^{\infty}$ be a sequence of functions from $X \subset \mathbb{R}^n \to \mathbb{R}$. The sequence $\{f_k\}_{k=1}^{\infty}$ is uniformly equicontinuous if for every $\epsilon > 0$, $\exists\, \delta > 0$ such that for all k and all $x, y \in X$ with $|x-y| < \delta$ we have $|f_k(x)-f_k(y)| < \epsilon$.

Theorem 4 (Arzelo-Ascoli Compactness Criterion [54])**.** *Suppose that* $\{f_k\}_{k=0}^{\infty}$ *is a sequence of functions from* $\mathbb{R}^n \to \mathbb{R}$*, such that*

$$|f_k(x)| \leq M \qquad (k = 1, \ldots,\ x \in \mathbb{R}^n) \tag{424}$$

for some constant M*, and that* $\{f_k\}_{k=1}^{\infty}$ *are uniformly equicontinuous. Then there exists a subsequence* $\{f_{k_j}\}_{j=1}^{\infty} \subseteq \{f_k\}_{k=1}^{\infty}$ *and a continuous function* f*, such that* $f_{k_j} \to f$ *uniformly on compact subsets of* $\mathbb{R}^n$*.*

REFERENCES

[1] Adjerid, S. and Flaherty, J. E., "A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations," *SIAM Journal on Numerical Analysis*, vol. 23, pp. 778–795, 1986.

[2] Adjerid, S. and Flaherty, J., "A moving mesh finite element method with local refinement for parabolic partial differential equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 56, pp. 3–26, 1986.

[3] Alves, M. A., Cruz, P., Mendes, A., Magalhães, F. D., Pinho, F. T., and Oliveira, P. J., "Adaptive multiresolution approach for solution of hyperbolic PDEs," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 3909–3928, 2002.

[4] Arney, D. C. and Flahetry, J. E., "A two-dimensional mesh moving technique for time dependent partial differential equations," *Journal of Computational Physics*, vol. 67, pp. 124–144, 1986.

[5] Athans, M. and Falb, P. L., *Optimal Control*. NY: McGraw-Hill Book Company, 1966.

[6] Babuška, I., Chandra, J., and Flaherty, J. E., eds., *Adaptive Computational Methods for Partial Differential Equations*. Philadelphia, PA: SIAM, 1983.

[7] Baines, M. J., *Moving Finite Elements*. New York: Oxford University Press, 1994.

[8] Baines, M. and Wathen, A., "Moving finite element methods for evolutionary problems. I. Theory," *Journal of Computational Physics*, vol. 79, pp. 245–269, 1988.

[9] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M., *Nonlinear Programming: Theory and Algorithms*. NY: John Wiley & Sons, Inc., 1993.

[10] Bell, J., Berger, M. J., Saltzman, J., and Welcome, M., "Three-dimensional adaptive mesh refinement for hyperbolic conservation laws," *SIAM Journal on Scientific Computing*, vol. 15, pp. 127–138, 1994.

[11] Bellingham, J., Kuwata, Y., and How, J., "Stable receding horizon trajectory control for complex environments," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003. AIAA 2003-5635.

[12] Bellman, R. E., *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[13] Berger, M. J. and Colella, P., "Local adaptive mesh refinement for shock hydrodynamics," *Journal of Computational Physics*, vol. 82, pp. 64–84, 1989.

[14] Berger, M. J. and Leveque, R., "Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems," *SIAM Journal on Numerical Analysis*, vol. 35, pp. 2298–2316, 1998.

[15] Berger, M. J. and Oliger, J., "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, pp. 484–512, 1984.

[16] Bertoluzza, S., "Adaptive wavelet collocation method for the solution of Burger's equation," *Transport Theory and Statistical Physics*, vol. 25, pp. 339–352, 1996.

[17] Betts, J. T., "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 193–207, 1998.

[18] Betts, J. T., *Practical Methods for Optimal Control using Nonlinear Programming.* Philadelphia, PA: SIAM, 2001.

[19] Betts, J. T., Biehn, N., and Campbell, S. L., "Convergence of nonconvergent irk discretizations of optimal control problems with state inequality constraints," *SIAM Journal on Scientific Computing*, vol. 23, pp. 1981–2007, 2002.

[20] Betts, J. T., Biehn, N., Campbell, S. L., and Huffman, W. P., "Compensating for order variation in mesh refinement for direct transcription methods," *Journal of Computational and Applied Mathematics*, vol. 125, pp. 147–158, 2000.

[21] Betts, J. T. and Huffman, W. P., "Trajectory optimization on a parallel processor," *Journal of Guidance, Control, and Dynamics*, vol. 14, pp. 431–439, 1991.

[22] Betts, J. T. and Huffman, W. P., "Mesh refinement in direct transcriptioin methods for optimal control," *Optimal Control Applications & Methods*, vol. 19, pp. 1–21, 1998.

[23] Biehn, N., Campbell, S. L., Jay, L., and Westbrook, T., "Some comments on DAE theory for IRK methods and trajectory optimization," *Journal of Computational and Applied Mathematics*, vol. 120, pp. 109–131, 2000.

[24] Binder, T., Blank, L., Dahmen, W., and Marquardt, W., "Grid refinement in multiscale dynamic optimization," Tech. Rep. LPT-2000-11, RWTH Aachen, Aachen, Germany, 2000.

[25] Binder, T., Blank, L., Dahmen, W., and Marquardt, W., "Iterative algorithms for multiscale state estimation, Part 1: Concepts," *Journal of Optimization Theory and Applications*, vol. 111, pp. 501–527, 2001.

[26] Binder, T., Blank, L., Dahmen, W., and Marquardt, W., "Iterative algorithms for multiscale state estimation, Part 2: Numerical investigations," *Journal of Optimization Theory and Applications*, vol. 111, pp. 529–551, 2001.

[27] Binder, T., Cruse, A., Villar, C. A. C., and Marquardt, W., "Dynamic optimization using a wavelet based adaptive control vector parametrization strategy," *Computers and Chemical Engineering*, vol. 24, pp. 1201–1207, 2000.

[28] Bock, H. G. and Plitt, K. J., "A multiple shooting algorithm for direct solution of optimal control problems," in *Proceedings of the 9th IFAC World Congress*, (Budapest, Hungary), pp. 242–247, 1984.

[29] Boltyanski, V. G., "Maximum principle in theory of optimal processes," *Doklady Akademii nauk SSSR (Proceedings of Academy of Sciences, U.S.S.R.)*, vol. 119, pp. 1070–1073, 1958. (Russian).

[30] Brauer, G. L., Cornick, D. E., and Stevenson, R., "Capabilities and applications of the program to optimize simulated trajectories (POST)," Tech. Rep. NASA CR-2770, NASA, Feb 1977.

[31] Breakwell, J. V., Speyer, J. L., and Bryson, A. E., "Optimization and control of nonlinear systems using the second variation," *SIAM Journal on Control*, vol. 1, pp. 193–223, 1963.

[32] Bryson, A. E., "Optimal control - 1950 to 1985," *IEEE Control Systems Magazine*, vol. 16, pp. 26–33, 1996.

[33] Bryson, A. E. and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*. Washington: Hemisphere Publishing Corporation, 1975.

[34] Burrus, C. S., Gopinath, R. A., and Guo, H., *Introduction to Wavelets and Wavelet Transforms*. NJ: Prentice Hall, 1998.

[35] Butcher, J. C., "Implicit Runge–Kutta processes," *Mathematics of Computation*, vol. 18, pp. 50–64, 1964.

[36] Calise, A. J. and Brandt, N., "Generation of launch vehicle abort trajectories using a hybrid optimization method," *Journal of Guidance, Control, and Dynamics*, vol. 27, pp. 930–937, 2004.

[37] Calise, A. J., Melamed, N., and Lee, S., "Design and evaluation of a three-dimensional optimal ascent guidance algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 867–875, 1998.

[38] Carlson, N. N. and Miller, K., "Design and application of a gradient weighted moving finite element code I: In one dimension," *SIAM Journal on Scientific Computing*, vol. 19, pp. 728–765, 1998.

[39] Ceniceros, H. D. and Hou, T. Y., "An efficient dynamically adaptive mesh for potentially singular solutions," *Journal of Computational Physics*, vol. 172, pp. 609–639, 2001.

[40] Citron, S. J., *Elements of Optimal Control*. New York: Hotl, Rinehart and Winston, Inc., 1969.

[41] Crandall, M. G., Evans, L. C., and Lions, P. L., "Some properties of viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 282, pp. 487–502, 1984.

[42] Crandall, M. G. and Lions, P. L., "Viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, pp. 1–42, 1983.

[43] Crandall, M. and Lions, P., "Two approximations of solutions of Hamilton-Jacobi equations," *Mathematics of Computation*, vol. 43, pp. 1–19, 1984.

[44] Daubechies, I., *Ten Lectures on Wavelets.* PA: Society for Industrial and Applied Mathematics, 1992.

[45] de Boor, C., *A Practical Guide to Splines*, vol. 27 of *Applied Mathematical Sciences.* New York: Springer-Verlag, 1978.

[46] Deslauriers, G. and Dubuc, S., "Symmetric iterative interpolation processes," *Constructive Approximation*, vol. 5, pp. 49–68, 1989.

[47] Donoho, D. L., "Interpolating wavelet transforms," tech. rep., Department of Statistics, Stanford University, Stanford, CA, 1992.

[48] Dontchev, A. L. and Hager, W. W., "The euler approximation in state constrained optimal control," *Mathematics of Computation*, vol. 70, pp. 173–203, 2000.

[49] Dontchev, A. L., Hager, W. W., and Veliov, V. M., "Second-order runge-kutta approximations in control constrained optimal control," *SIAM Journal on Numerical Analysis*, vol. 38, pp. 202–226, 2000.

[50] Dorfi, E. A. and Drury, L. O., "Simple adaptive grids for 1-d initial value problems," *Journal of Computational Physics*, vol. 69, pp. 175–195, 1987.

[51] Drikakis, D. and W., R., *High Resolution Methods for Incompressible and Low-Speed Flows.* New York: Springer, 2004.

[52] Elnagar, G., Kazemi, M. A., and Razzaghi, M., "The pseudospectral Legendre method for discretizing optimal control problems," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1793–1796, 1995.

[53] Enright, P. J. and Conway, B. A., "Discrete approximation to optimal trajectories using direct transcription and nonlinear programming," *Journal of Guidance, Control, and Dynamics*, vol. 15, pp. 994–1002, 1992.

[54] Evans, L. C., *Partial Differential Equations.* Providence, Rhode Island: American Mathematical Society, 1998.

[55] Fahroo, F. and Ross, I. M., "Direct trajectory optimization by a Chebyshev pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 160–166, 2002.

[56] Fahroo, F. and Ross, I., "A spectral patching method for direct trajectory optimization," *Journal of the Astronautical Sciences*, vol. 48, pp. 269–286, 2000.

[57] Fahroo, F. and Ross, I., "Trajectory optimization by indirect spectral collocation methods," in *AIAA/AAS Astrodynamics Specialist Conference*, (Denver, CO), pp. 123–129, August 2000.

[58] Feeley, T. S. and Speyer, J. L., "Techniques for developing approximate optimal advanced launch system guidance," *Journal of Guidance, Control, and Dynamics*, vol. 17, pp. 889–896, 1994.

[59] Gamkrelidze, R. V., "Discovery of the maximum principle," *Journal of Dynamical and Control Systems*, vol. 5, pp. 437–451, 1999.

[60] Gelfand, I. M. and Fomin, S. V., *Calculus of Variations.* Prentice-Hall, N.J., rev. english ed., 1963. Translated and edited from Russian by R.A. Silverman.

[61] Gill, P. E., Murray, W., and Saunders, M. A., *User's Guide for SNOPT Version 6, A Fortran Package for Large-Scale Nonlinear Programming.* Stanford, CA, 2002.

[62] Goldstine, H. H., *A History of the Calculus of Variations from the 17th through the 19th Century.* Springer-Verlag, 1980.

[63] Gong, Q. and Ross, I. M., "Autonomous pseudospectral knotting methods for space mission optimization," in *16th AAS/AIAA Space Flight Mechanics Meeting*, no. AAS 06-151, 2006.

[64] Hager, W. W., "Runge-kutta methods in optimal control and the transformed adjoint system," *Numerische Mathematik*, vol. 87, pp. 247–282, 2000.

[65] Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer-Verlag, New York, 1987.

[66] Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer-Verlag, New York, 1991.

[67] Hargraves, C. R. and Paris, S. W., "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, pp. 338–342, 1987.

[68] Harten, A., "Adaptive multiresolution schemes for shock computations," *Journal of Computational Physics*, vol. 115, pp. 319–338, 1994.

[69] Harten, A., "Multiresolution algorithms for the numerical solution of hyperbolic conservation laws," *Comm. Pure Appl. Math.*, vol. 48, pp. 1305–1342, 1995.

[70] Harten, A., "Multiresolution representation of data: A general framework," *SIAM Journal on Numerical Analysis*, vol. 33, no. 3, pp. 1205–1256, 1996.

[71] Harten, A., Enquist, B., Osher, S., and Chakravarthy, S., "Uniformly high-order accurate essentially non-oscillatory schemes III," *Journal of Computational Physics*, vol. 71, pp. 231–303, 1987.

[72] Herman, A. L. and Conway, B. A., "Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules," *Journal of Guidance, Control, and Dynamics*, vol. 19, pp. 592–599, 1996.

[73] Hodges, D. H., Bless, R. R., and Seywald, H., "A finite element method for the solution of state-constrained optimal control problems," *Journal of Guidance, Control and Dynamics*, vol. 18, pp. 1036–1043, 1995.

[74] Hoffman, J. D., *Numerical Methods for Engineers and Scientists.* NY: Marcel Dekker, Inc., 2001.

[75] Holmström, M., "Solving hyperbolic PDEs using interpolating wavelets," *SIAM Journal on Scientific Computing*, vol. 21, pp. 405–420, 1999.

[76] Holmström, M. and Waldén, J., "Adaptive wavelet methods for hyperbolic PDEs," *Journal of Scientific Computing*, vol. 13, no. 1, pp. 19–49, 1998.

[77] Huang, W., Ren, Y., and Russell, R. D., "Moving mesh methods based on moving mesh partial differential equations," *Journal of Computational Physics*, vol. 113, pp. 279–290, 1994.

[78] Jain, S. and Tsiotras, P., "Multiresolution trajectory optimization schemes for problems with moving targets and dynamically changing environment," *Journal of Guidance, Control, and Dynamics*. To be Submitted.

[79] Jain, S. and Tsiotras, P., "Trajectory optimization using multiresolution techniques," *Journal of Guidance, Control, and Dynamics*. Accepted.

[80] Jain, S. and Tsiotras, P., "A solution of the time-optimal Hamilton-Jacobi-Bellman equation on the interval using wavelets," in *Proceedings of 43rd IEEE Conference on Decision and Control*, (Paradise Island, Bahamas), pp. 2728–2733, 2004.

[81] Jain, S. and Tsiotras, P., "The method of reflection for solving the time-optimal Hamilton-Jacobi-Bellman equation on the interval," in *13th IEEE Mediterranean Conference on Control and Automation*, (Limassol, Cyprus), pp. 1062–1067, 2005.

[82] Jain, S. and Tsiotras, P., "Multiresolution-based direct trajectory optimization," in *Proceedings of 46th IEEE Conference on Decision and Control*, (New Orleans, LA), pp. 5991–5996, 2007.

[83] Jain, S. and Tsiotras, P., "Sequential nonlinear trajectory optimization for moving targets," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008. Submitted.

[84] Jain, S., Tsiotras, P., and Velenis, E., "Optimal feedback velocity profile generation for a vehicle with given acceleration limits: A level set implementation," in *13th IEEE Mediterranean Conference on Control and Automation*, 2008. Submitted.

[85] Jain, S., Tsiotras, P., and Zhou, H.-M., "A hierarchical multiresolution adaptive mesh refinement for the solution of evolution PDEs," *SIAM Journal on Scientific Computing*. Submitted.

[86] Jain, S., Tsiotras, P., and Zhou, H.-M., "Adaptive multiresolution mesh refinement for the solution of evolution PDEs," in *Proceedings of 46th IEEE Conference on Decision and Control*, (New Orleans, LA), pp. 3525–3530, 2007.

[87] Jameson, L., "A wavelet-optimized, very high order adaptive grid and order numerical method," *SIAM Journal on Scientific Computing*, vol. 19, pp. 1980–2013, 1998.

[88] Jardin, M. R. and Bryson, A. E., "Neighboring optimal aircraft guidance in winds," *Journal of Guidance, Control, and Dynamics*, vol. 24, pp. 710–715, 2001.

[89] Jiang, G. S. and Peng, D., "Weighted ENO schemes for Hamilton-Jacobi equations," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 2126–2143, 2000.

[90] Jiang, G. S. and Shu, C.-W., "Efficient implementation of weighted ENO schemes," *Journal of Computational Physics*, vol. 126, no. 1, pp. 202–228, 1996.

[91] Johnson, I. W., Wathen, A. J., and Wathen, M. J., "Moving finite element methods for evolutionary problems. II. Applications," *Journal of Computational Physics*, vol. 79, pp. 270–297, 1988.

[92] Kelly, H. J., "An optimal guidance approximation theory," *IEEE Transactions on Automatic Control*, vol. 9, pp. 375–380, 1964.

[93] Kirk, D. E., *Optimal Control Theory: An Introduction.* Prentice-Hall, N.J., 1970.

[94] Kumar, R. R. and Seywald, H., "Dense-sparse discretization for optimization and real-time guidance," *Journal of Guidance, Control, and Dynamics*, vol. 19, pp. 501–503, 1996.

[95] Kunz, K. S., *Numerical Analysis.* New York: McGraw-Hill, 1957.

[96] Lax, P. D., "Weak solutions of nonlinear hyperbolic equations and their numerical computation," *Communications on Pure and Applied Mathematics*, vol. 7, pp. 195–206, 1954.

[97] LeVeque, R., *Numerical Methods for Conservation Laws.* Boston: Birhäuser Verlag, 1992.

[98] Li, R. and Tang, T., "Moving mesh discontinuous Galerkin method for hyperbolic conservation laws," *Journal of Scientific Computing*, vol. 27, pp. 347–363, 2006.

[99] Liu, X. D., Osher, S., and Chan, T., "Weighted essentially non-oscillatory schemes," *Journal of Computational Physics*, vol. 115, no. 1, pp. 200–212, 1994.

[100] Lu, P., "Regulation about time-varying trajectories: Precision entry guidance illustrated," *Journal of Guidance, Control, and Dynamics*, vol. 22, pp. 784–790, 1999.

[101] Lu, P., "Predictor-corrector entry guidance for low lifting vehicles," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, no. AIAA 2007-6425, 2007.

[102] Mallat, S. G., "Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$," in *Transactions of the American Mathematical Society*, vol. 315, pp. 69–87, 1989.

[103] Mallat, S. G., "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intellegence*, vol. 2, pp. 674–693, 1989.

[104] Meder, D. S. and McLaughlin, J. R., "A generalized trajectory simulation system," in *Summer Computer Simulation Conference, Washington, D.C., July 12-14, 1976, Proceedings. (A77-24576 09-66) Montvale, N.J., AFIPS Press, 1976, p. 366-372.* (Mankin, J. B., Gardner, R. H., and Shugart, H. H., eds.), pp. 366–372, 1976.

[105] Miller, K., "Moving finite elements II," *SIAM Journal on Numerical Analysis*, vol. 18, pp. 1033–1057, 1981.

[106] Miller, K. and Miller, R. N., "Moving finite elements I," *SIAM Journal on Numerical Analysis*, vol. 18, pp. 1019–1032, 1981.

[107] Oberle, H. J. and Grimm, W., "BNDSCO - A program for the numerical solution of optimal control problems," Tech. Rep. DLR IB/515-89/22, Institute for Flight System Dynamics, German Aerospace Research Establishment, Oberpfaffenhofen, Germany, 1989.

[108] Ohtsuka, T., "Quasi-newton-type continuation method for nonlinear receding horizon control," *Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 685–692, 2002.

[109] Osher, S. and Fedkiw, R. P., "Level set methods: An overview and some recent results," *Journal of Computational Physics*, vol. 169, no. 2, pp. 436–502, 2001.

[110] Osher, S. and Fedkiw, R. P., *Level Set Methods and Dynamic Implicit Surfaces.* New York: Springer, 2003.

[111] Osher, S. and Shu, C.-W., "High order essentially non-oscillatory schemes for Hamilton-Jacobi equations," *SIAM Journal on Numerical Analysis*, vol. 28, pp. 902–921, 1991.

[112] Pesch, H. J., "Real-time computation of feedback controls for constrained optimal control problems. part 2: A correction method based on multiple shooting," *Optimal Control Applications & Methods*, vol. 10, pp. 147–171, 1989.

[113] Petzold, L. R., "Observations on an adaptive moving grid method for one-dimensional systems for partial differential equations," *Applied Numerical Mathematics*, vol. 3, pp. 347–360, 1987.

[114] Polak, E., "An historical survey of computational methods in optimal control," *SIAM Review*, vol. 15, no. 2, pp. 553–584, 1973.

[115] Rao, A. V. and Mease, K. D., "Eigenvector approximate dichotomic basis method for solving hyper-sensitive optimal control problems," *Optimal Control Applications and Methods*, vol. 20, pp. 59–77, 1999.

[116] Ross, I. M., "User's manual for DIDO (ver. pr.1β): A matlab application package for solving optimal control problems," Tech. Rep. 04-01.0, Naval Postgraduate School, Monterey, CA, 2004.

[117] Ross, I. M. and Fahroo, F., "Pseudospectral knotting methods for solving optimal control problems," *Journal of Guidance, Control, and Dynamics*, vol. 27, pp. 397–405, 2004.

[118] Ross, I. M., Fahroo, F., and Strizzi, J., "Adaptive grids for trajectory optimization by pseudospectral methods," in *AAS/AIAA Spaceflight Mechanics Conference*, (Ponce, Puerto Rico), pp. 649–668, 2003.

[119] Ross, I. M., Gong, Q., and Sekhavat, P., "Low-thrust, high accuracy trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 921–933, 2007.

[120] Russell, R. D. and Shampine, L. F., "A collocation method for boundary value problems," *Numerische Mathematik*, vol. 19, pp. 13–36, 1972.

[121] Schlegel, M., Stockmann, K., Binder, T., and Marquardt, W., "Dynamic optimization using adaptive control vector parameterization," *Computers and Chemical Engineering*, vol. 29, pp. 1731–1751, 2005.

[122] Sendov, B. and Popov, V. A., *The Averaged Moduli of Smootheness.* Pure and Applied Mathematics, Chichester: John Whiley & Sons, 1988.

[123] Sethian, J. A., *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* NY: Cambridge University Press, 1999.

[124] Seywald, H., "Trajectory optimization based on differential inclusion," *Journal of Guidance, Control and Dynamics*, vol. 17, pp. 480–487, 1994.

[125] Seywald, H. and Cliff, E. M., "Neighboring optimal control based feedback law for the advanced launch system," *Journal of Guidance, Control, and Dynamics*, vol. 17, pp. 1154–1162, 1994.

[126] Shen, H. and Tsiotras, P., "Time-optimal control of axi-symmetric spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 22, no. 5, pp. 682–694, 1999.

[127] Shu, C.-W. and Osher, S., "Efficient implementation of essentially non-oscillatory shock capturing schemes," *Journal of Computational Physics*, vol. 77, pp. 439–471, 1988.

[128] Shu, C.-W. and Osher, S., "Efficient implementation of essentially non-oscillatory shock capturing schemes II," *Journal of Computational Physics*, vol. 83, pp. 32–78, 1989.

[129] Sod, G. A., "A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws," *Journal of Computational Physics*, vol. 27, pp. 1–31, 1978.

[130] Speyer, J. L. and Bryson, A. E., "A neighboring optimum feedback control scheme based on estimated time-to-go with application to re-entry flight paths," *AIAA Journal*, vol. 6, pp. 769–776, 1968.

[131] Speyer, J. L. and Crues, E. Z., "Approximate optimal atmospheric guidance law for aeroassisted plane-change maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 13, pp. 792–802, 1990.

[132] Stryk, O. v. and Bulirsch, R., "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, pp. 357–373, 1992.

[133] Sussmann, H. J. and Willems, J. C., "300 years of optimal control: From the brachystochrone to the maximum principle," *IEEE Control Systems Magazine*, vol. 17, pp. 32–44, 1997.

[134] Sweldens, W., "The lifting scheme: A construction of second generation wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, pp. 511–546, 1998.

[135] Sweldens, W. and Schröder, P., "Building your own wavelets at home," in *Wavelets in Computer Graphics*, ACM SIGGRAPH Course Notes, pp. 15–87, 1996.

[136] Tang, H. and Tang, T., "Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws," *SIAM Journal on Numerical Analysis*, vol. 41, pp. 487–515, 2003.

[137] Thomas, J. W., *Numerical Partial Differential Equations.* NY: Springer, 1995.

[138] Thompson, J. F., "A survey of dynamically-adaptive grids in the numerical solution of partial differential equations," *Applied Numerical Mathematics*, vol. 1, pp. 3–27, 1985.

[139] Vasilyev, O. V., "Solving multi-dimensional evolution problems with localized structures using second generation wavelets," *International Journal of Computational Fluid Dynamics*, vol. 17, no. 2, pp. 151–168, 2003.

[140] Vasilyev, O. V. and Bowman, C., "Second-generation wavelet collocation method for the solution of partial differential equations," *Journal of Computational Physics*, vol. 165, pp. 660–693, 2000.

[141] Waldén, J., "Filter bank methods for hyperbolic PDEs," *Journal of Numerical Analysis*, vol. 36, pp. 1183–1233, 1999.

[142] Weiss, R., "The application of implicit Runge-Kutta and collocation methods to boundary value problems," *Mathematics of Computation*, vol. 28, pp. 449–464, 1974.

[143] Williams, P., "Application of pseudospectral methods for receding horizon control," *Journal of Guidance, Control, and Dynamics*, vol. 27, pp. 310–314, 2004.

[144] Yan, H., Fahroo, F., and Ross, I. M., "Real-time computation of neighboring optimal control laws," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002. AIAA 2002-4657.