Final Report for Period: 08/2007 - 07/2008 **Submitted on:** 09/16/2008

Principal Investigator: Tsiotras, Panagiotis . **Award ID:** 0510259

Organization: GA Tech Res Corp - GIT

Submitted By:

Tsiotras, Panagiotis - Principal Investigator

Title:

Wavelets in Control and Optimization

Project Participants

Senior Personnel

Name: Tsiotras, Panagiotis

Worked for more than 160 Hours: Yes

Contribution to Project:

Post-doc

Name: Kim, Byungmoon

Worked for more than 160 Hours: Yes

Contribution to Project:

From May 2006-December 2006 Mr. Kim worked as a research assistant in the project. From January 2007 to May 2007 he had a joint post-doctoral appointment between the school of Aerospace Engineering and the Math department at Georgia Tech. Mr. Kim developed the multi-resolution scheme for solving level-set methods.

Graduate Student

Name: Jain, Sachin

Worked for more than 160 Hours: Yes

Contribution to Project:

Graduate research assistanship (doctoral student)

Name: Jung, Dongwon

Worked for more than 160 Hours: Yes

Contribution to Project:

Graduate research assistanship (doctoral student)

Name: Bakolas, Efstathios

Worked for more than 160 Hours: No

Contribution to Project:

E. Bakolas was partially supported by this award. He worked in the development of the multi-resolution path-planning algorithm using wavelets.

Undergraduate Student

Name: Earl, Andrew

Worked for more than 160 Hours: No

Contribution to Project:

Name: Gloss, William

Worked for more than 160 Hours: No

Contribution to Project:

Name: Ozawa, Eiji
Worked for more than 160 Hours: No
Contribution to Project:

Name: Yu, Jason
Worked for more than 160 Hours: No
Contribution to Project:

Name: Hageman, Daniel
Worked for more than 160 Hours: No
Contribution to Project:
Performed undergraduate research with the PI

Name: Staub, Jeffery
Worked for more than 160 Hours: No
Contribution to Project:

Technician, Programmer

Other Participant

Research Experience for Undergraduates

Organizational Partners

Other Collaborators or Contacts

The PI initiated a collaboration with Prof. Haomin Zhou from the School of Mathematics at Georgia Tech. Prof. Zhou's expertise lie in level set methods, wavelet processing, and computational methods of pde's. The PI and the graduate students supported by this project have conducted weekly meetings with Prof. Zhou to exchange ideas and discuss the progress of this research. Zhou was also a member of the doctoral dissertation committee of one of the PI's students (S. Jain).

The PI has also started an informal collaboration with Prof. Magnus Egerstedt from the School of Electrical and Computer Engineering at Georgia Tech. Prof. Egerstedt prior experience is in optimal control and path and motion planning for mobile robots. Professor Egerstedt currently served as a thesis committee member for two of the graduate students involved in this project (S. Jain and E. Bakolas).

The PI has also continued his collaboration with Dr. Francois Chaplais of the Laboratory of Signals and Systems of the Ecole des Mines in Paris, France, which was initiated in 2003. This collaboration has led to two conference papers and one journal paper.

Activities and Findings

Research and Education Activities: (See PDF version submitted by PI at the end of the report)
Please see attached file.

Findings: (See PDF version submitted by PI at the end of the report)
Please see attached file.

Training and Development:
Sachin Jain received his doctoral dissertation in May 2008. The title of his dissertation was 'Multiresolution Strategies for the Numerical

Solution of Optimal Control Problems.' His studies were supported solely by this NSF award. During this project Sachin became familiar with adaptive gridding techniques, numerical methods for solving pde's and polynomial interpolation methods. His doctoral dissertation is one of very few (only?) works in the literature that have addressed multi-resolution trajectory optimization.

Mr. Dongwon (Thomas) Jung received his doctoral dissertation in December 2007. The title of his dissertation was Hierarchical Path Planning and Control of a Small Fixed-wing UAV: Theory and Experimental Validation." He was responsible for the design and construction of the UAV platform as well as the implementation of the control algorithms on the UAV autopilot. He designed both the software and hardware of the UAV. This included the on-board autopilot and associated electronics as well as the ground station. He also wrote the communication protocols between the UAV and the ground station. He designed the attitude and navigation filters and the low-level PID loops for the autopilot. He implemented the hierarchical multiresolution path planning scheme on the UAV autopilot and validated in a hardware-in-the-loop environment, which he also developed himself.

Mr. Efstathios Bakolas received his MS degree in Aerospace Engineering in May 2007. The title of his MS thesis was 'A Hierarchical On-Line Path Planning Scheme using Wavelets.' He is currently pursuing his Ph.D. degree in Aerospace Engineering at
Georgia Tech under the supervision of the PI.

Several undergraduate students have participated in the design and construction of the UAV. They gained valuable experience especially in the airframe construction and the validation of its aerodynamic properties.

This project has enabled the PI (P. Tsiotras) to be involved in a new research area that is not part of the traditional controls. He has gained an understanding of computational methods for solving pde's, graph theory, and signal processing techniques using wavelets. His involvement with this project has allowed him to apply new, nontraditional to control and optimization problems. His involvement with the hardware and software development of the UAV has complemented his theoretical skills. This project has also allowed the PI to interact and collaborate with faculty from Math and Electrical Engineering Departments at Georgia Tech, as well as with foreign researchers.

Outreach Activities:
The PI has disseminated the results of this research via the web.

The PI also investigated participation in the Georgia Intern-Fellowships for Teachers (GIFT) program. The Georgia Tech GIFT program assists Georgia Tech faculty in locating a local science or mathematics teacher to work in the PIs lab as well as it provides supports for translating their summer experience back to the teacher's classroom. However, his participation was not made possible.

Journal Publications

Jain, S., Tsiotras, P., and Zhou, H.-M., "Adaptive Multiresolution Mesh Refinement for the Solution of Evolution PDEs", SIAM Journal of Scientific Computing, p. , vol. , (2008). Accepted,

Jain, S. and Tsiotras, P., "Multiresolution-Based Direct Trajectory Optimization", AIAA Journal of Guidance Control, and Dynamics, p. 1424, vol. 31, (2008). Published, 10.2514/1.32220

Chaplais, F., Tsiotras, P. and Jung, D., "Redundant Wavelet Processing on the Half-Axis with Applications to Signal Denoising with Small Delays: Theory and Experiments", International Journal on Adaptive Control and Signal Processing, p. 447, vol. 20, (2006). Published, 10.1002/acs.911

Jung, D., Ratti, J., and Tsiotras, P., "Real-time Implementation and Validation of a New Hierarchical Path Planning Scheme for UAVs via Hardware-in-the-Loop Simulation", Journal of Intelligent and Robotic Systems, p. , vol. 52, (2008). Published, 10.1007/s10846-008-9255-0

Jain, S. and Tsiotras, P., "Sequential Multiresolution Trajectory Optimization Schemes for Problems with Moving Targets", AIAA Journal of Guidance, Control, and Dynamics, p. , vol. , (2008). Submitted,

Books or Other One-time Publications

Jung, D., Zhou, D., Fink, R., Williams, T., Moshe, J. and Tsiotras, P., "Design and Development of a Low-Cost Test-Bed for Undergraduate

Education in UAVs", (2005). Conference Proceedings, Published
Collection: 44th IEEE Conference on Decision and Control/European Control Conference ECC 2005
Bibliography: Seville, Spain, Dec. 12-15, 2005, pp. 2739-2744.

Jung, D., and Tsiotras, P.,, "Inertial Attitude and Position Reference System Development for a Small UAV", (2007). Conference Proccedings, Published
Collection: AIAA Infotech at Aerospace,
Bibliography: Rohnert Park, CA, May 7-10, 2007, AIAA Paper 07-2763

Jung, D., and Tsiotras, P., "Modelling and Hardware-in-the-Loop Simulation for a Small Unmanned Aerial Vehicle", (2007). Conference Proccedings, Published
Collection: AIAA Infotech at Aerospace
Bibliography: Rohnert Park, CA, May 7-10, 2007, AIAA Paper 07-2768.

Tsiotras, P., and Bakolas, E., "A Hierarchical On-Line Path-Planning Scheme Using Wavelets", (2007). Conference Proccedings, Published
Collection: European Control Conference
Bibliography: Kos, Greece, July 2-5, 2007, pp. 2806-2812

Tsiotras, P., "Multiresolution Hierarchical Path-Planning for Small UAVs", (2007). Conference Proccedings, Published
Collection: European Control Conference
Bibliography: Kos, Greece, July 2-5, 2007

Jain, S. and Tsiotras, P., "Multiresolution-Based Direct Trajectory Optimization", (2007). Conference Proccedings, Published
Collection: 46th IEEE Conference on Decision and Control
Bibliography: New Orleans, LA, Dec. 12-14, 2007, pp. 5991-5996

Jain, S. and Tsiotras, P., "Adaptive Multiresolution Mesh Refinement for the Solution of Evolution PDEs", (2007). Conference Proccedings, Published
Collection: 46th IEEE Conference on Decision and Control
Bibliography: New Orleans, LA, Dec. 12-14, 2007, pp. 3525-3530

Bakolas, E. and Tsiotras, P., "Multiresolution Path Planning Via Sector Decompositions Compatible to On-Board Sensor Data", (2008). Conference Proccedings, Published
Collection: AIAA Guidance, Navigation, and Control Conference (AIAA Paper 2008-7238)
Bibliography: Honolulu, HI, Aug. 18-21, 2008

Cowlagi, R. and Tsiotras, P., "Beyond Quadtrees: Cell Decomposition for Path Planning using the Wavelet Transform", (2007). Conference Proccedings, Published
Collection: 46th IEEE Conference on Decision and Control
Bibliography: New Orleans, LA, Dec. 12-14, 2007, pp. 1392-1397.

Jung, D., and Tsiotras, P., "Bank-to-Turn Control for a Small UAV using Backstepping and Parameter Adaptation", (2008). Conference Proceedings, Published
Collection: 7th IFAC World Congress
Bibliography: Seoul, South Korea, July 6-11, 2008, pp. 4406-4411.

Jung, D., and Tsiotras, P., "Multiresolution On-Line Path Planning for Small Unmanned Aerial Vehicles", (2008). Book, Published
Collection: American Control Conference
Bibliography: Seattle, WA, June~11-13, 2008, pp. 2744-2749.

Cowlagi, R., and Tsiotras, P., "Multiresolution Path Planning with Wavelets: A Local Replanning Approach", (2008). Conference Proceedings, Published

Collection: American Control Conference
Bibliography: Seattle, WA, June 11-13,
2008, pp. 1220-1225.

Jain, S., Tsiotras, P., and Velenis, E., "Optimal Feedback Velocity Profile
Generation for a Vehicle with Given Acceleration Limits: A Level Set Implementation", (2008). Conference Proceedings, Published
Collection: 16th Mediterranean Conference on Control and Automation
Bibliography: Ajaccio, Corsica,
France, June 25-26, 2008.

Jung, D., Ratti, J., and Tsiotras, P., "Real-time Implementation and Validation of a
New Hierarchical Path Planning Scheme for UAVs via Hardware-in-the-Loop Simulation", (2008). Conference Proceedings, Published
Editor(s): K. Valavalis
Collection: International Symposium of Unmanned Aerial Vehicles (UAV'08)
Bibliography: Orlando, FL, June 23-24, 2008.

Jung, D., and Tsiotras, P., "On-line Path Generation for Small Unmanned Aerial
Vehicles using B-Spline Path Templates", (2008). Conference Proceedings, Published
Collection: AIAA Guidance, Navigation, and Control Conference
Bibliography: Honolulu, HW, Aug.~18-21, 2008 (AIAA Paper 2008-7135)

Jain, S., and Tsiotras, P., "Sequential Multiresolution Trajectory Optimization for
Moving Targets", (2008). Conference Proceedings, Published
Collection: AIAA Guidance, Navigation, and Control Conference
Bibliography: Honolulu, HI, Aug. 18-21, 2008 (AIAA Paper 2008-6980)

Sachin Jain, "Multiresolution Strategies for the Numerical Solution of Optimal Control Problems", (2008). Thesis, Published
Collection: Ph.D. dissertation
Bibliography: School of Aerospace Engineering, Georgia Institutde of Technology, May 2008

Dongwon Jung, "Hierarchical Path Control of a Small Fixed-wing UAV: Theory and Experimental Validation", (2007). Thesis, Published
Collection: Ph.D. dissertation
Bibliography: School of Aerospace Engineering, Georgia Institute of Technology, December 2007

Efstathios Bakolas, "A Hierarchical On-Line Path Planning Scheme
using Wavelets", (2007). Thesis, Published
Collection: M.S. thesis
Bibliography: School of Aerospace Engineering, Georgia Institute of Technology, May 2007

Web/Internet Site

URL(s):
http://www.ae.gatech.edu/labs/dcsl/research-3.html
Description:
A summary of the project is provided. Major findings and publications are posted.

Other Specific Products

Product Type:
Software (or netware)
Product Description:
Developed software to perform path planning among obstacles and implemented it on a small microprocessor

Sharing Information:

A provisional patent has been filled through the Georgia Tech Office of Technology Licensing.

Product Type:

Instruments or equipment developed

Product Description:

We developed a small-scale unmanned aerial vehicle. This includes the airframe, the on-board autopilot, the ground station, and a hardware-in-the-loop (HIL) testing environment.

Sharing Information:

The platform and HIL architecture is available to other interested researchers to use in order to test their own algorithms.

Contributions

Contributions within Discipline:

The multiresolution algorithms proposed in this work provide superior numerical accuracy and faster execution speeds for solving optimal control problems. This is especially true for aerospace and mechanical applications, where state sensitivities, diverse time scales and uncertainty of the environment make control and trajectory generation extremely challenging using standard methods. The results of this research will advance the state-of-the art on numerical optimal control, especially for solving on-line optimal trajectories.

Contributions to Other Disciplines:

Our research will have an impact on the applied mathematics area since the techniques we have developed can be used for solving broader classes of pde's, not necessarily only those that appear in optimal control problems. The results of this research will have a broader impact on other areas such as image segmentation, medical imaging, signal encoding/decoding, denoising, etc. were multi-resolution can increase accuracy and speed up execution times. The approach to develop scalable algorithms can be used in the area of embedded control systems, where limited computational resources in terms of CPU speed and memory often hinder the implementation of 'out-of-the-box' control and path-planning algorithms.

Contributions to Human Resource Development:

Sachin Jain was supported by this NSF award towards the completion of this doctoral degree (awarded in May 2008).

Dongwon (Thomas) Jung was supported by this NSF current towards the completion of this doctoral degree (awarded in December 2007).

Efstathios Bakolas was partially supported by this NSF current towards the completion of this masters degree (awarded in May 2007)

Six undergraduate students have been involved in research.

The project has enabled the PI (P. Tsiotras) to be involved in new research areas and collaborate with faculty from other disciplines (Dr. Haomin Zhou from the Mathematics Dept and Dr. Magnus Egerstedt from Electrical and Computer Engineering).

Contributions to Resources for Research and Education:

As part of this project a team of undergraduare students was involved in the construction of a UAV platform and its associated autopilot. This UAV platform has become a central piece in the PI's lab for educating graduate and undergraduate students in UAV-related research.

Contributions Beyond Science and Engineering:

We do not expect that the technology developed under this project to have an immediate effect on regulatory policies. Just focusing on UAVs, there has been a long and tedious process during the last several years to come up with a consistent policy for just testing UAVs in controlled air-space. The FAA is still trying to iron out all potential conflicts. A formal blueprint is not expected before 2012. It is possible however, that the decisions of FAA and other government agencies tasked to prepare the list of regulations for operating UAVs will be affected by the currently accepted (or anticipated) level of intelligence of UAVs. In that respect, the algorithms developed in this work may have an impact in the final document. This remains to be seen, however.

It is very likely that the developed technology will find its way to commerical products. At the moment, we are working with a local Atlanta company (Guided Systems Technology) to port the developed path-planning algorithms to their UAV products.

Categories for which nothing is reported:

Organizational Partners

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.

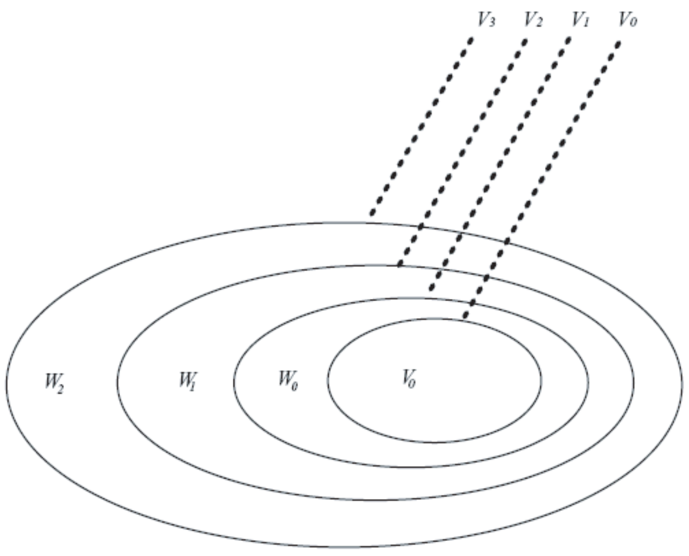


Figure 1: Pictorial representation of the subspaces $\mathcal{V}_j$ and $\mathcal{W}_j$.

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

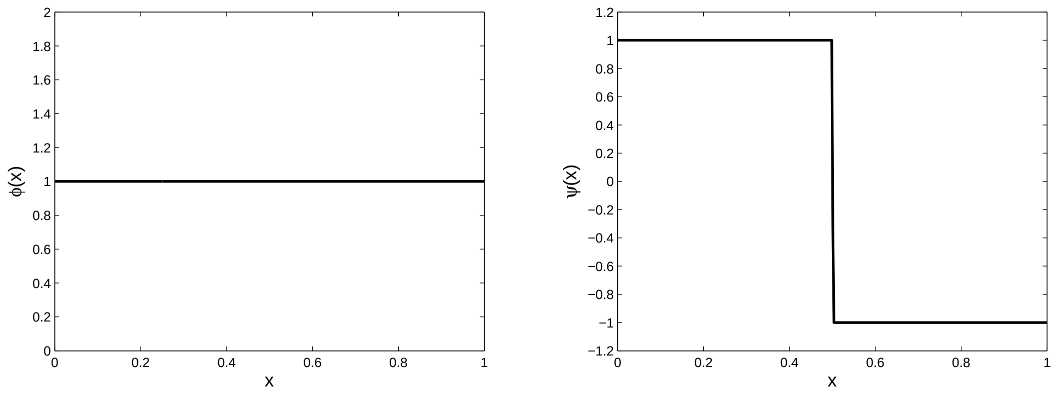
$$\phi(x) = \sum_k h_k\sqrt{2}\phi(2x-k), \quad k \in \mathbb{Z}. \tag{12}$$

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

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

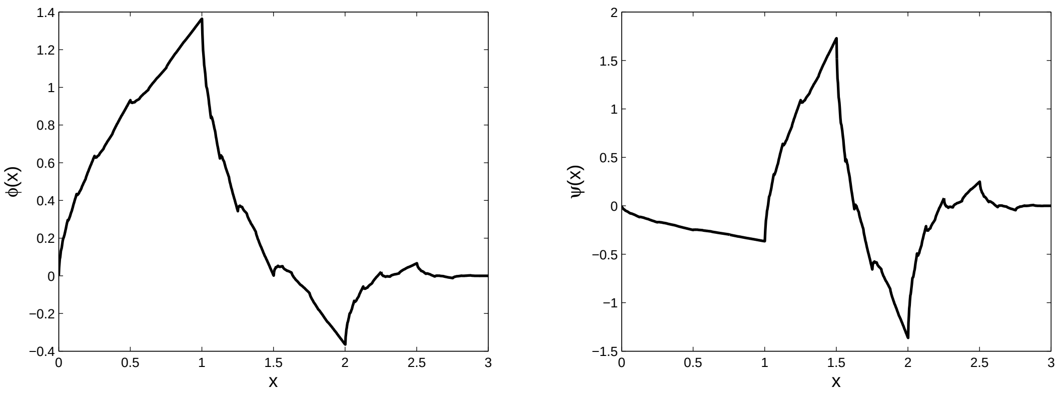
Examples of some commonly used wavelets are Haar wavelets (Figure 2), Daubechies wavelets (Figure 3), symlets (Figures 4, 5), and coiflets (Figures 6, 7). It has been shown in the literature [34, 44] that the condition of orthogonality $\mathcal{V}_j \perp \mathcal{W}_j$ gives

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



(a) Scaling function ($\phi(x)$). (b) Wavelet ($\psi(x)$).

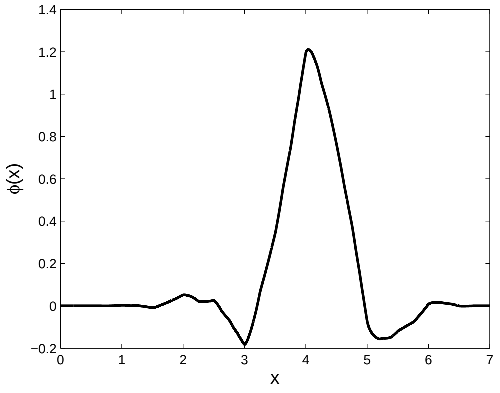
Figure 2: Haar wavelets (Daubechies wavelets with $\mu = 1$).



(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 ($\phi(x)$).

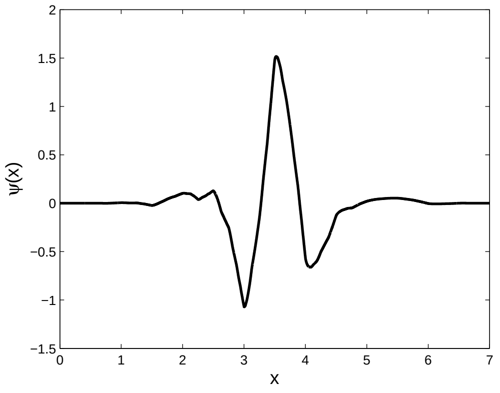
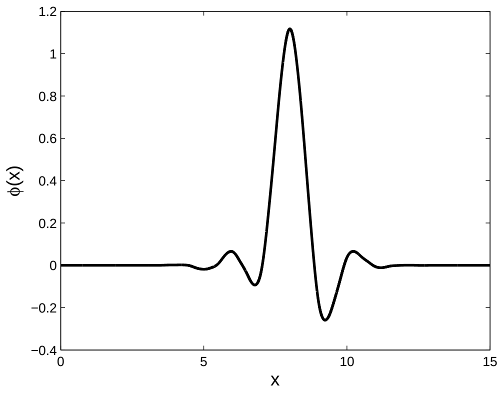

(b) Wavelet ($\psi(x)$).

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


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

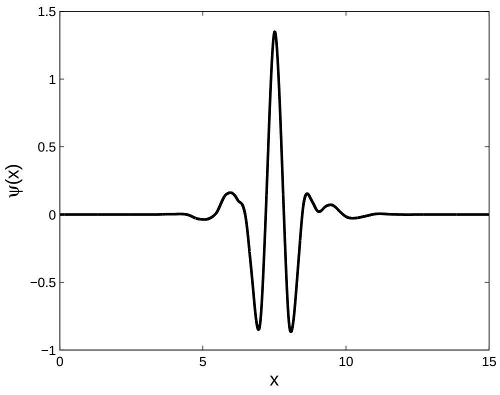

(b) Wavelet ($\psi(x)$).

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

From Fine Scale to Coarse Scale:

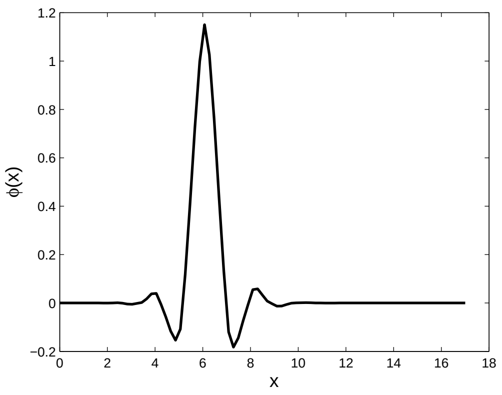
$$c_{j,k} = \sum_{\ell} h_{\ell-2k} c_{j+1,\ell}, \tag{15}$$

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

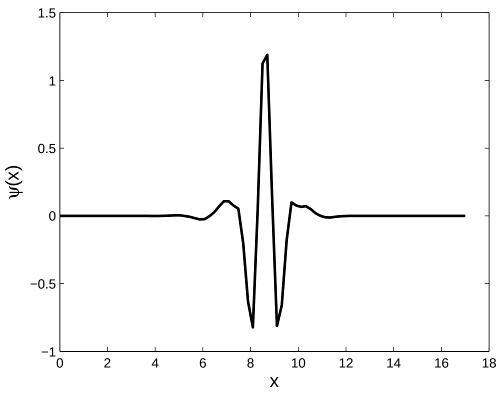
From Coarse Scale to Fine Scale:

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

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

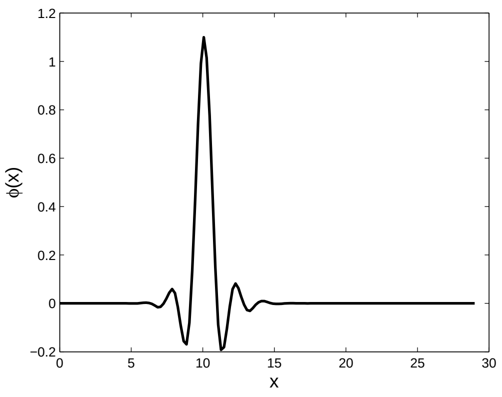


(a) Scaling function ($\phi(x)$).

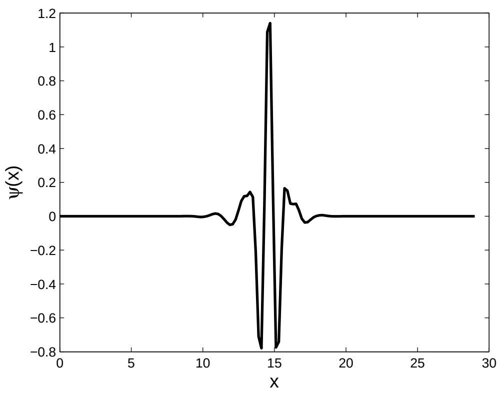


(b) Wavelet ($\psi(x)$).

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 $x_{j,k+\ell}$ ($\ell = -(p-1)/2, \ldots, (p+1)/2$) 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 $h_{j,k,\ell}$, $\ell = -(p-1)/2, \ldots, (p+1)/2$, 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

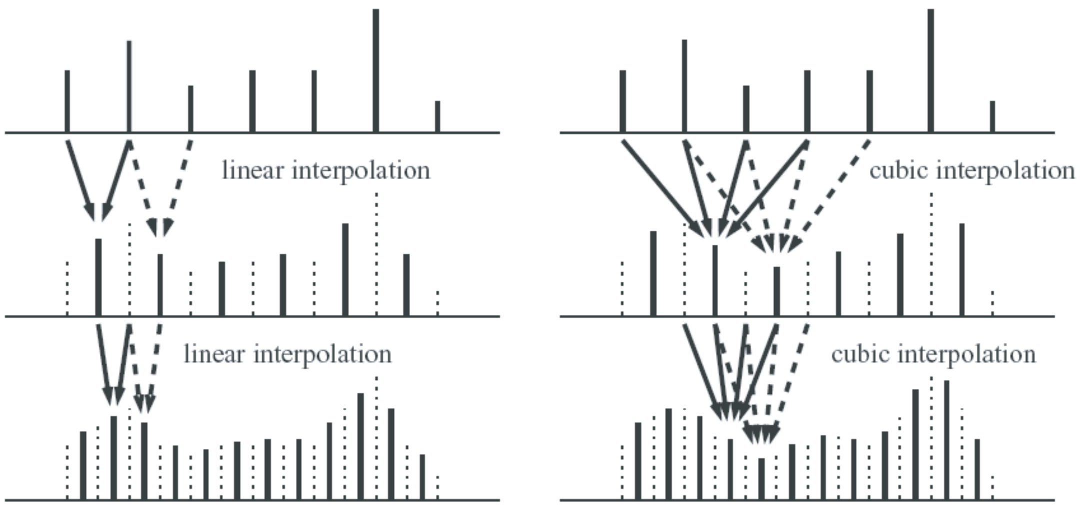


Figure 8: Examples of interpolating subdivision [135].

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

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

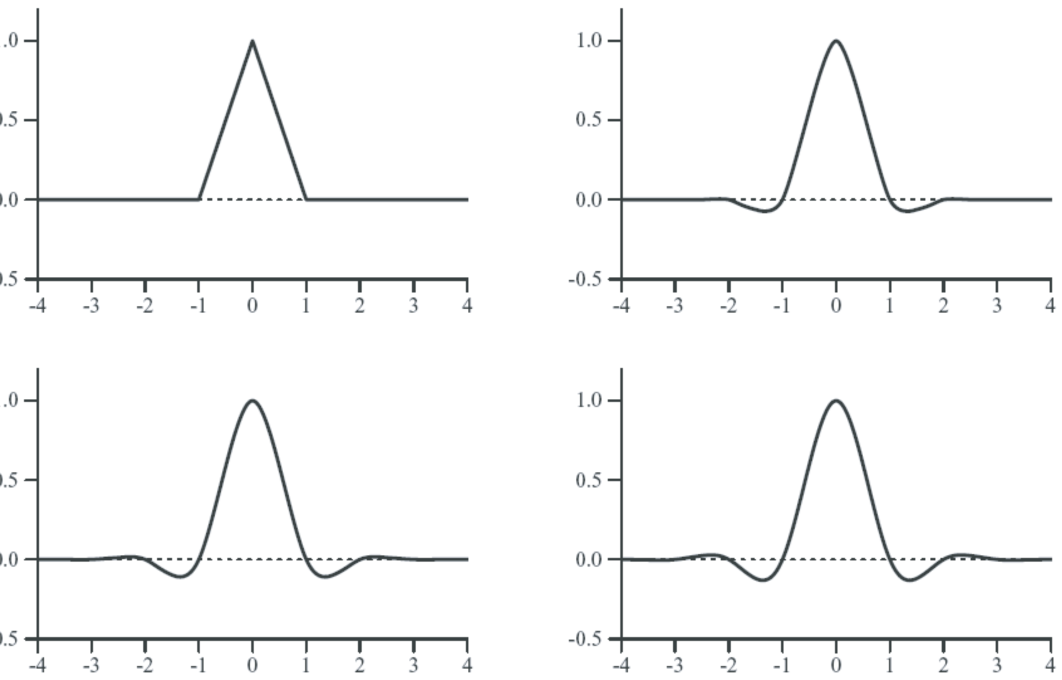


Figure 9: Scaling functions resulting from interpolating subdivision. Going from left to right, top to bottom, p is 1, 3, 5, 7 [135].

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],

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

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

$$D_pG = [F'(z(s)),\ 1], \tag{39}$$

$$D_xG = 0, \tag{40}$$

$$D_zG = F''(z(s))u_x(x(s), t(s)). \tag{41}$$

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

$$\frac{\mathrm{d}z}{\mathrm{d}s}(s) = D_pG \cdot p \tag{43}$$

$$= F'(z(s))u_x(x(s), s) + u_t(x(s), s) \tag{44}$$

$$= 0 \quad \text{by (36a)}. \tag{45}$$

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

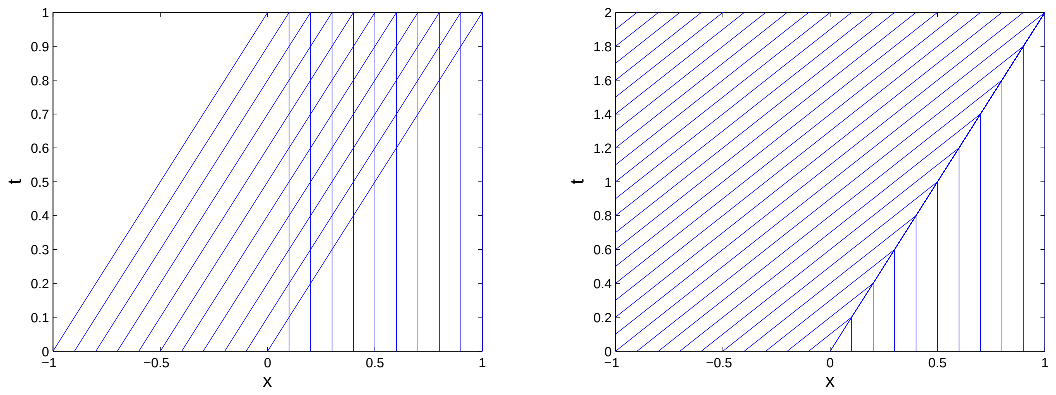
$$u(x,t) = \begin{cases} 1, & x \leq t/2, \\ 0, & x > t/2. \end{cases} \tag{57}$$

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

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

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).

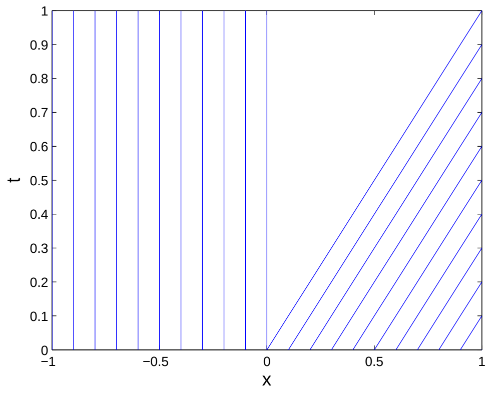
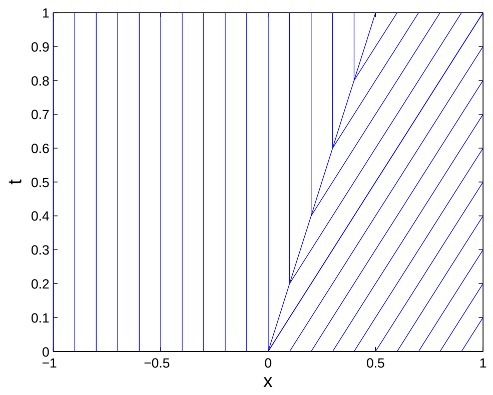
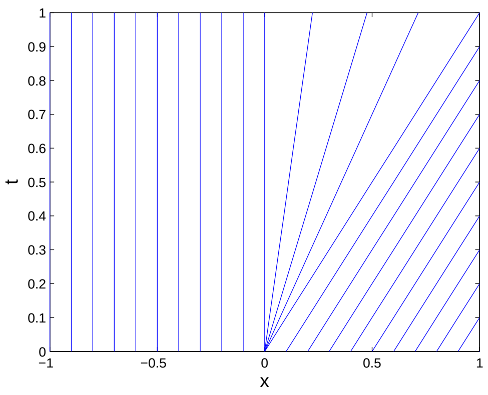


Figure 11: Characteristic curves associated with the IVP (55), (59).



(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

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

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

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

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

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

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

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

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,

$$\begin{aligned} D_pG &= \left[\frac{\partial H}{\partial p_1}(p_1(s), x(s)),\ 1\right], && (77)\\ D_xG &= \left[\frac{\partial H}{\partial x}(p_1(s), x(s)),\ 0\right], && (78)\\ D_zG &= 0. && (79)\end{aligned}$$

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

$$\begin{aligned} \frac{\mathrm{d}z}{\mathrm{d}s}(s) &= \frac{\partial H}{\partial p_1}(p_1(s), x(s))p_1(s) + p_2(s) && (82)\\ &= \frac{\partial H}{\partial p_1}(p_1(s), x(s))p_1(s) - H(p_1(s), x(s)) \quad \text{by (75a), (76).} && (83)\end{aligned}$$

In summary, the characteristic equations for the HJ equation are

$$\begin{aligned} \frac{\mathrm{d}p_1}{\mathrm{d}s}(s) &= -\frac{\partial H}{\partial x}(p_1(s), x(s)), && (84a)\\ \frac{\mathrm{d}z}{\mathrm{d}s}(s) &= \frac{\partial H}{\partial p_1}(p_1(s), x(s))p_1(s) - H(p_1(s), x(s)), && (84b)\\ \frac{\mathrm{d}x}{\mathrm{d}s}(s) &= \frac{\partial H}{\partial p_1}(p_1(s), x(s)). && (84c)\end{aligned}$$

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

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

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,

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

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.

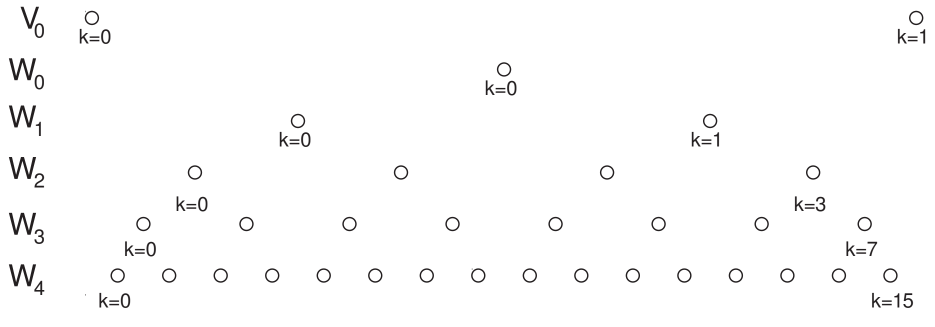


Figure 13: Example of a dyadic grid.

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

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

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

(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.

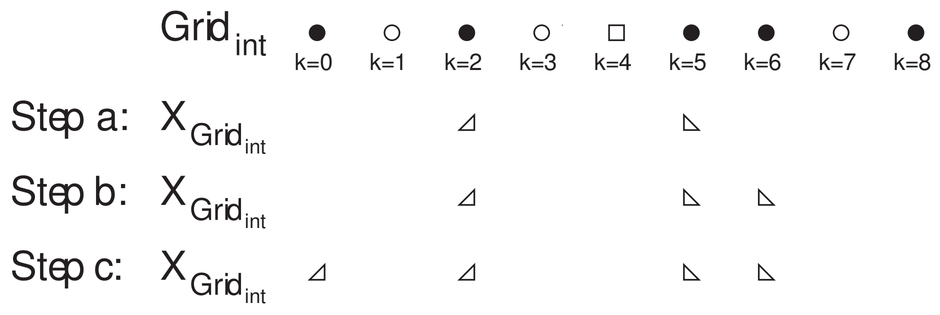


Figure 14: Demonstration of the procedure for finding the nearest points in piecewise polynomial interpolation.

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 $D^2_L g$, or we can include the next point to the right and use $D^2_{L+1} g$. 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 $|D^2_L g| \le |D^2_{L+1} g|$, set $c = D^2_L g$ and $L^\star = L - 1$; otherwise, set $c = D^2_{L+1} g$ 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

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

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

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

which completes the proof. □

Example 3

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

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

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

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

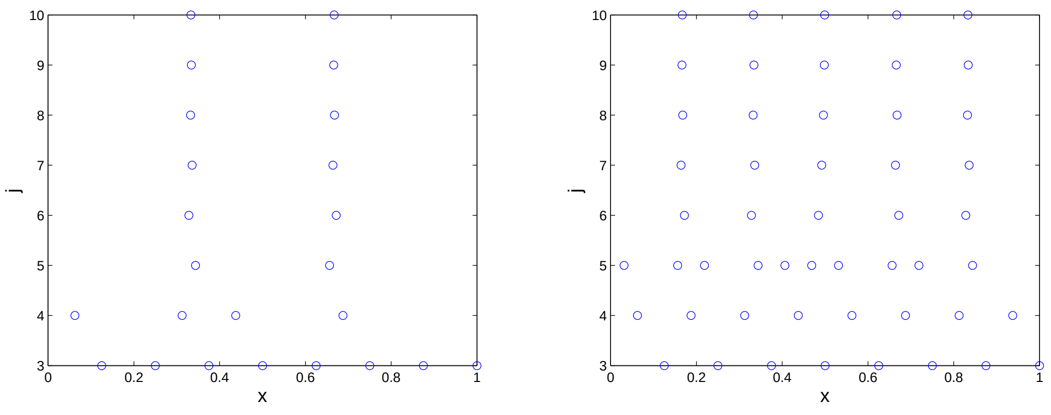
For this example, we consider a grid with $J_{\min} = 3$ and $J_{\max} = 10$ and use ENO interpolation for the encoding and the decoding algorithms described above. For both g_1 and g_2, the data compression factor

$$
C = \frac{2^{J_{\max}} + 1 - N_{\text{p}}}{2^{J_{\max}}+1} \times 100\%, \qquad (132)
$$

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 $j = J_{\min} \ldots J_{\max} - 1$, 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}	$N_{\text{p}}/N_{\text{Heno}}$	$N_{\text{p}}/N_{\text{Hc}}$
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

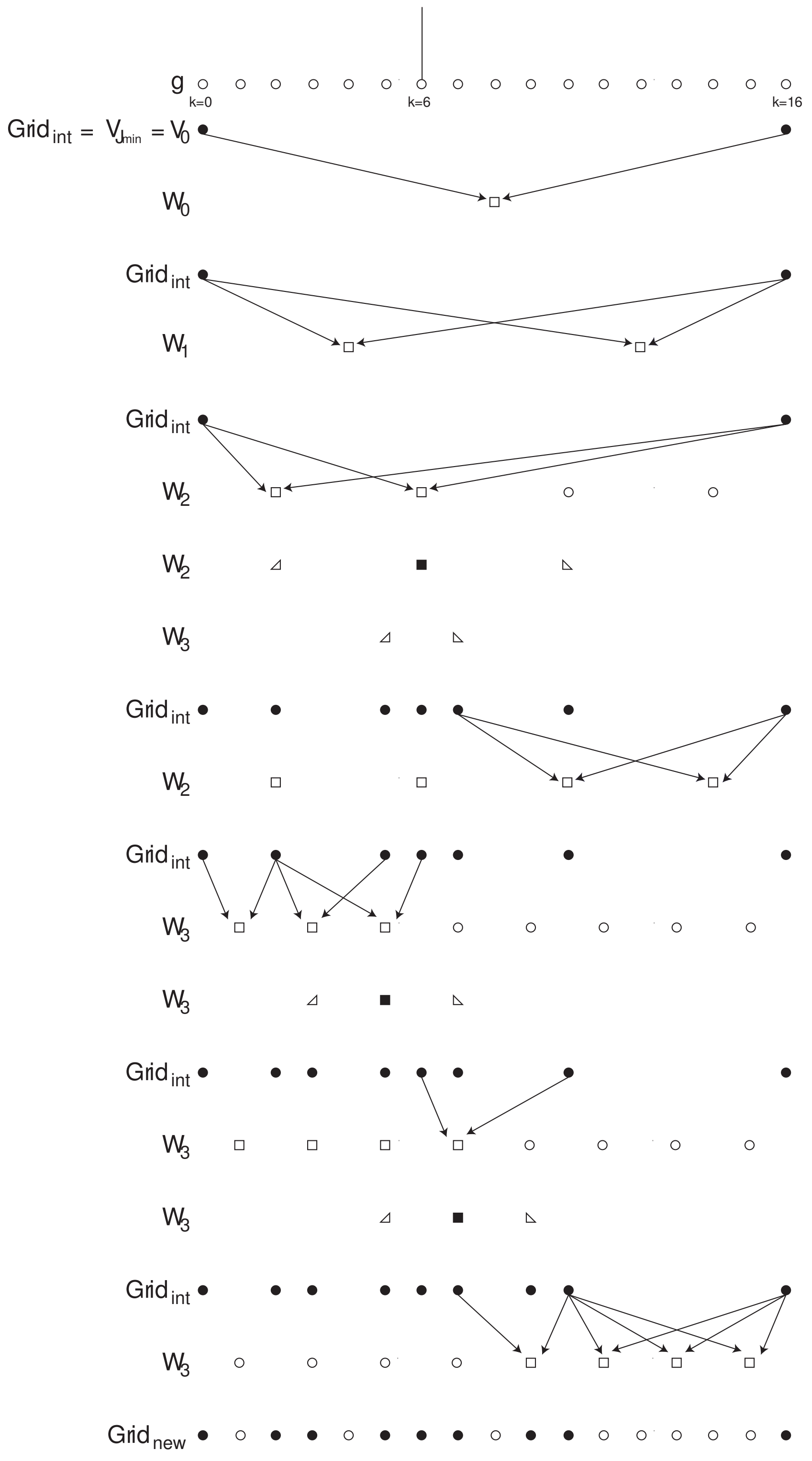


Figure 16: Demonstration of the proposed grid adaptation algorithm using Example 5.

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 $j = 0, \ldots, 3$, 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.

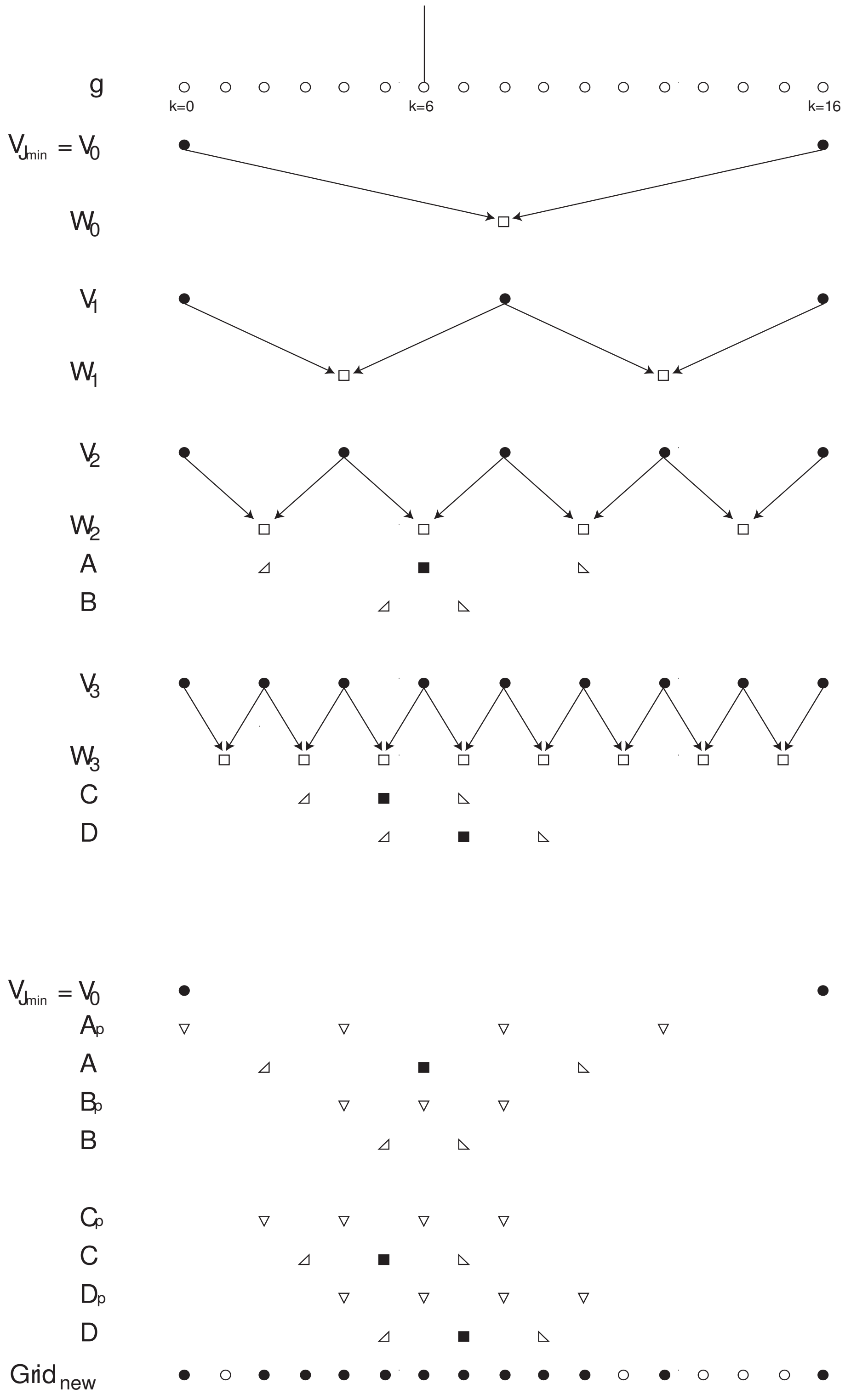


Figure 17: Demonstration of the grid adaptation approach of Alves et al. [3] using Example 5.

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 $0 \le k \le 2^j - 1$ and $J_{\min} \le j \le J_{\max} - 1$, 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 $J_{\min} \le j \le J_{\max} - 2$ 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 $y_{j,k}$ ($0 \le k \le 2^j - 1$ and $J_{\min} \le j \le J_{\max} - 1$), 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	$N_\mathrm{p}/N_\mathrm{A}$	k	N_p	N_A	$N_\mathrm{p}/N_\mathrm{A}$
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	$N_\mathrm{p}/N_\mathrm{A}$
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 $\mathsf{Grid}_\mathrm{new}$ 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.

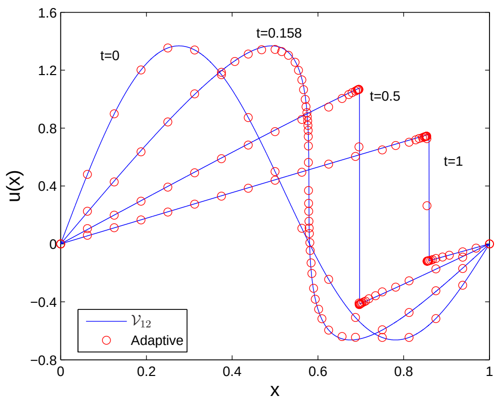
For solving (160)-(161), we use (71) along with the ENO-Roe scheme proposed by Shu and Osher [128] on a non-uniform grid for calculating the numerical flux functions $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}}$. For temporal integration, we use a third-order total variation diminishing (TVD) Runge–Kutta (RK) scheme [127]. The numerical solution at times $t = 0\,\mathrm{s}$, $t = 0.158\,\mathrm{s}$, $t = 0.5\,\mathrm{s}$ and $t = 1\,\mathrm{s}$ 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 $t = 0\,\mathrm{s}$, $t = 0.1\,\mathrm{s}$, $t = 0.158\,\mathrm{s}$ and $t = 1\,\mathrm{s}$. 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.

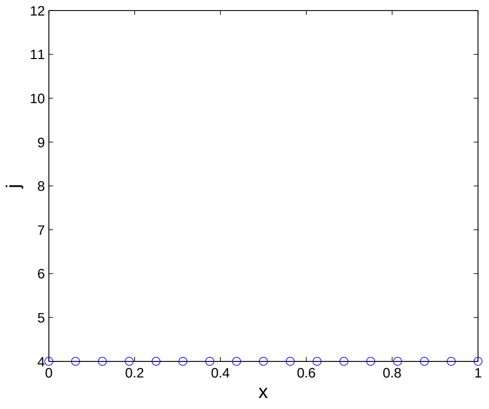
Table 5: Example 8. L_1 error and computational times for uniform vs. adaptive mesh.

	Uniform Mesh		Adaptive Mesh			
$J_{\max}$	N_p in $\mathcal{V}_{J_{\max}}$	t_{cpu} (s)	N_p at t_f in $\mathsf{Grid}_{\text{new}}$	$E_1(u)$	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

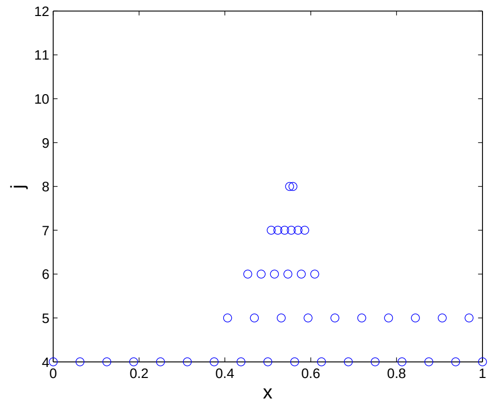
In the next two examples, we consider Hamilton-Jacobi equations, that is, evolution



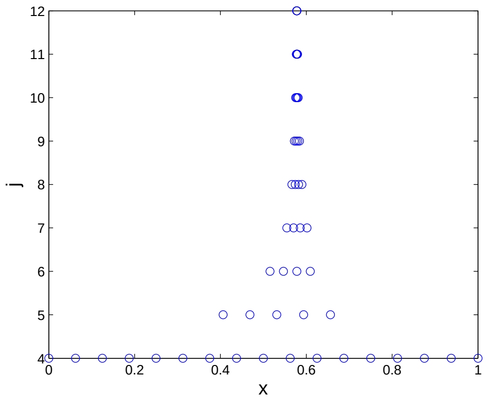
(a) Solution $u(x,t)$.



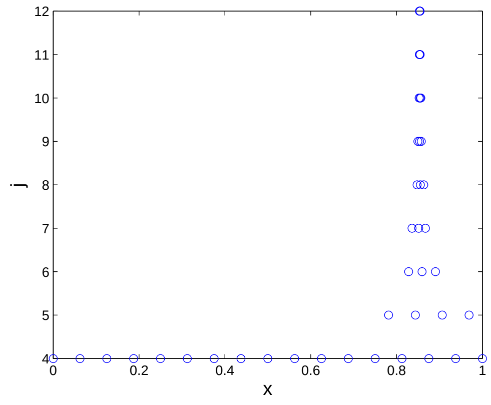
(b) Grid point distribution at $t = 0\,\mathrm{s}$.



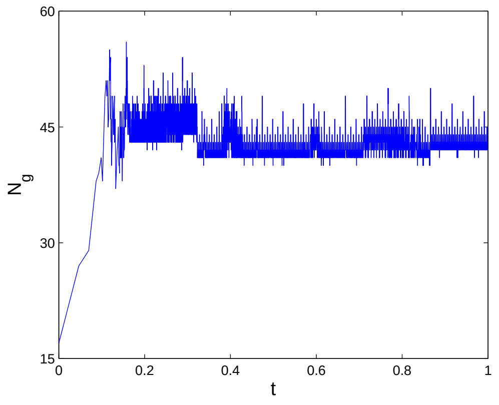
(c) Grid point distribution at $t = 0.1\,\mathrm{s}$.



(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,

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

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

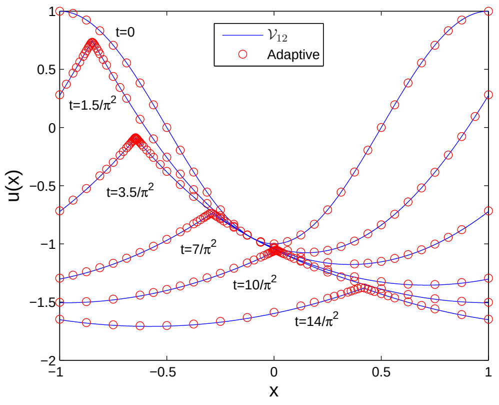
Next, we consider the Hamilton-Jacobi (HJ) equation with non-convex $f(u_x)$,

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

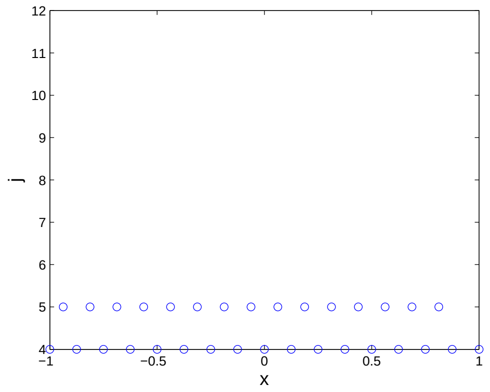
with

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

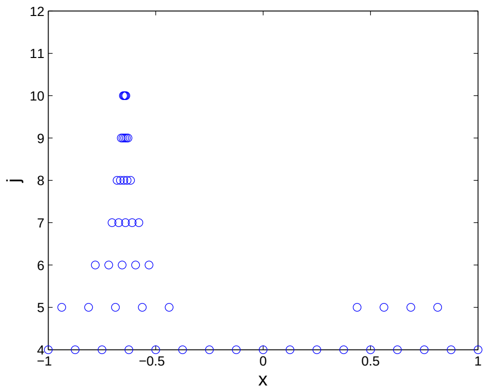
where α is a constant. We again use an LF scheme (163) for solving the IBVP (168)-(169). The derivatives u_x^+, u_x^- in the LF discretization are approximated using a WENO scheme and the temporal integration is performed using a third-order TVD RK scheme.



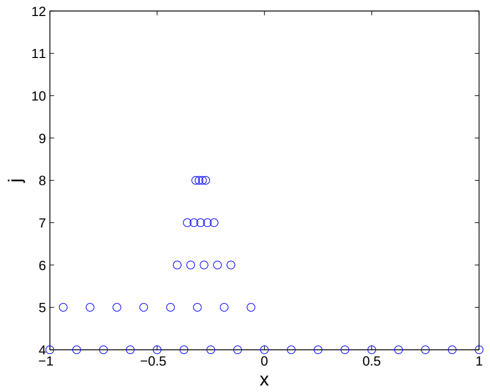
(a) Solution $u(x,t)$.



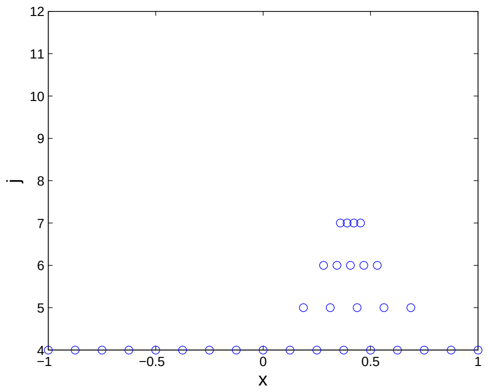
(b) Grid point distribution at $t = 0\,\text{s}$.



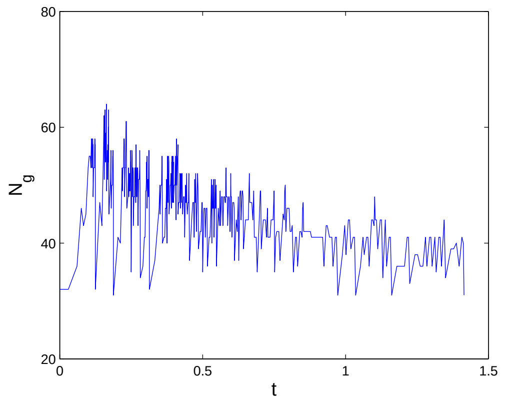
(c) Grid point distribution at $t = 3.5/\pi^2\,\text{s}$.



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



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



(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 $\mathsf{Grid}_{\rm new}$	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 $t = 0.2\,\text{s}$, $t = 0.13\,\text{s}$ 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 $t = 0.05\,\text{s}$, $t = 0.1\,\text{s}$, $t = 0.15\,\text{s}$, and $t = 0.2\,\text{s}$ and Lax's problem at times $t = 0.05\,\text{s}$, $t = 0.1\,\text{s}$, and $t = 0.13\,\text{s}$ 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_p in $\mathsf{Grid}_{\text{new}}$	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}

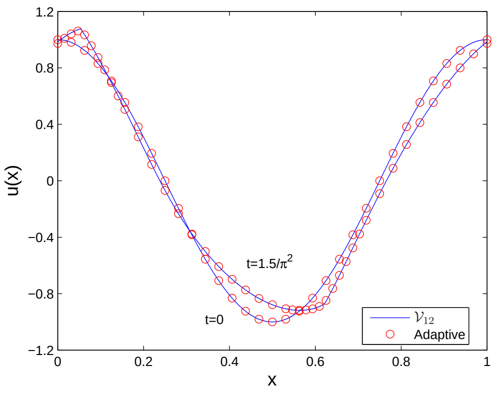
Table 11: Example 12. Lax's problem. L_1 errors at different times for $J_{\max} = 12$.

t	N_p in $\mathsf{Grid}_{\text{new}}$	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}

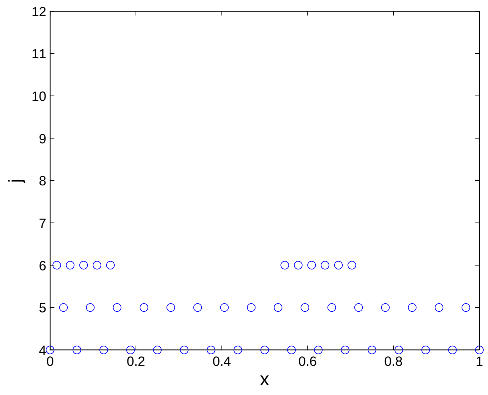
5.5 Summary

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.

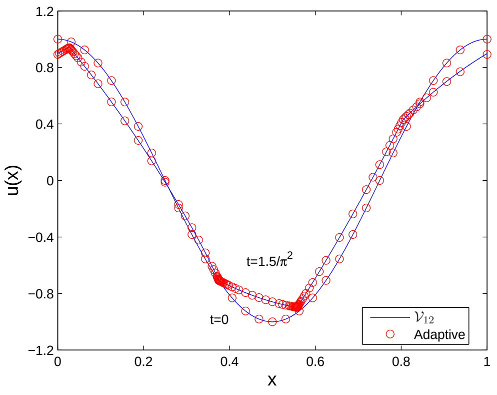
Next, we move on to our main motivation behind this work, that is, develop fast and efficient algorithms for solving optimal control problems.



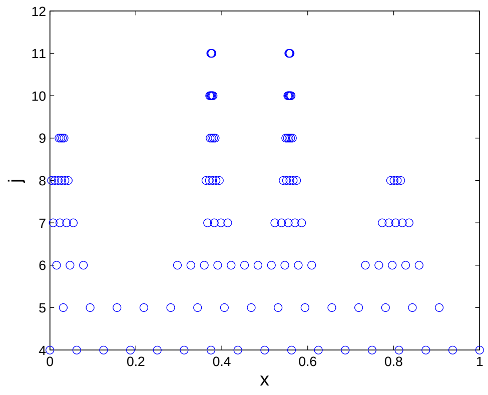
(a) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 0.5$.



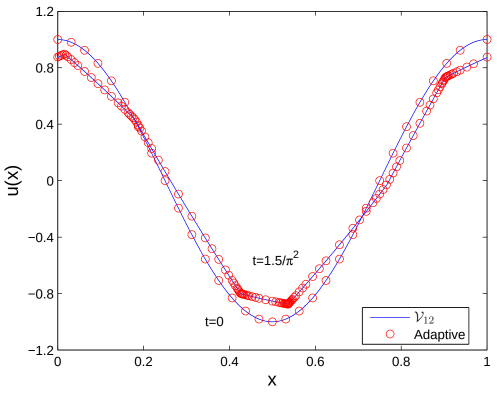
(b) Grid point distribution at $t = 1.5/\pi^2$ s for $\alpha = 0.5$.



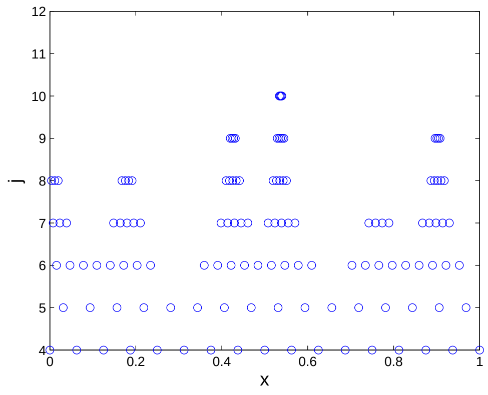
(c) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 1$.



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

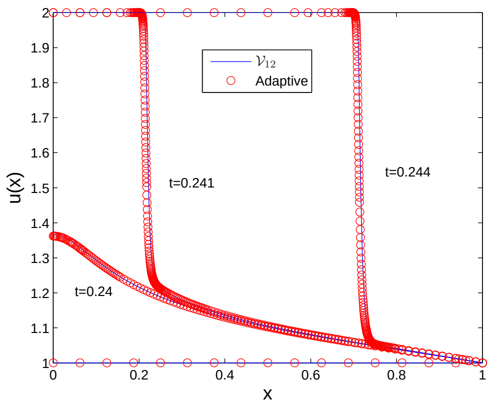


(e) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 1.5$.

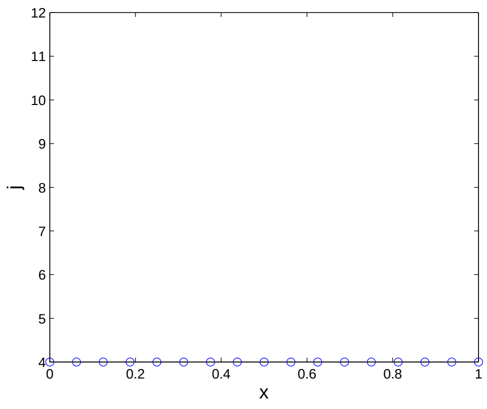


(f) Grid point distribution at $t = 1.5/\pi^2$ s for $\alpha = 1.5$.

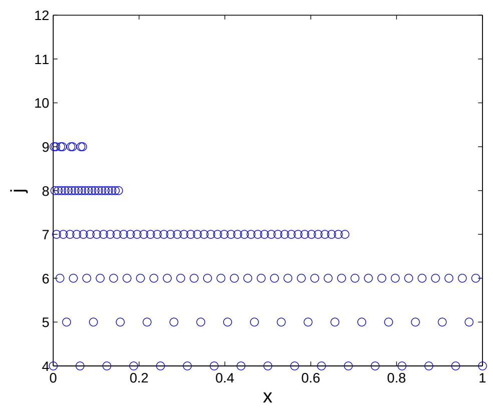
Figure 20: Example 10. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 0.001,\ N_1 = N_2 = 2$.



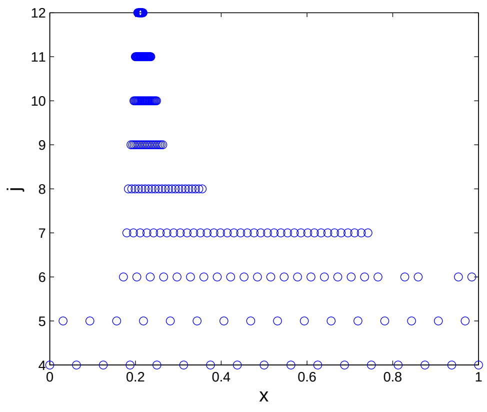
(a) Solution $u(x,t)$.



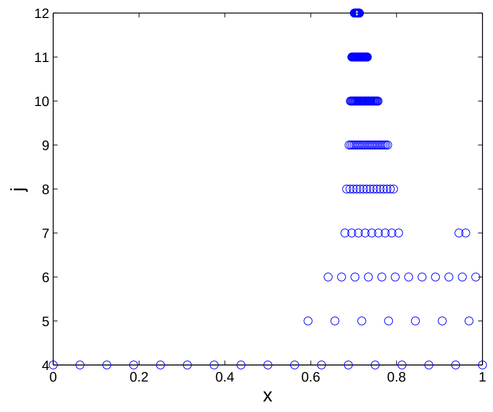
(b) Grid point distribution at $t = 0\,\mathrm{s}$.



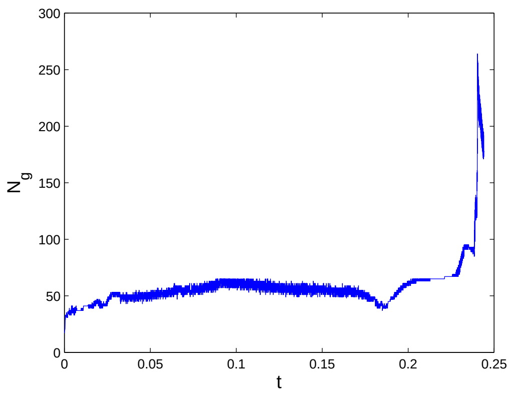
(c) Grid point distribution at $t = 0.24\,\mathrm{s}$.



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

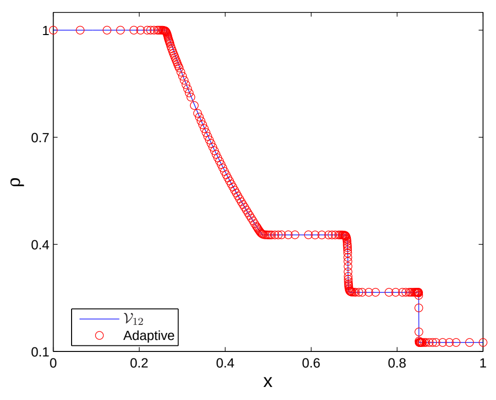


(e) Grid point distribution at $t = 0.244\,\mathrm{s}$.

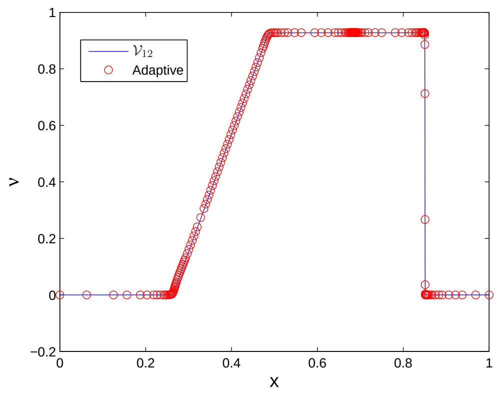


(f) Time evolution of the number of grid points.

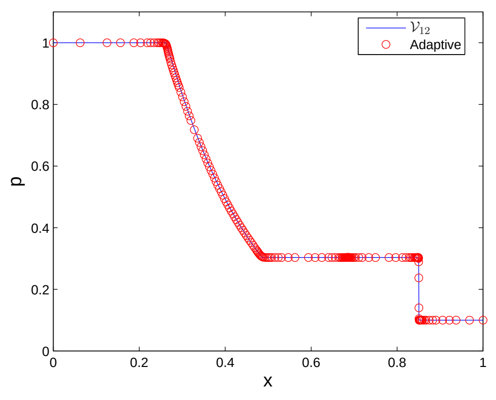
Figure 21: Example 11. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 10^{-5}/2^{J_{\max}-j},\ N_1 = N_2 = 1$.



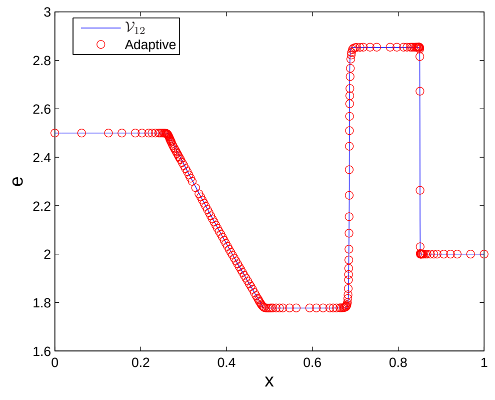
(a) Solution $\rho(x, 0.2)$.



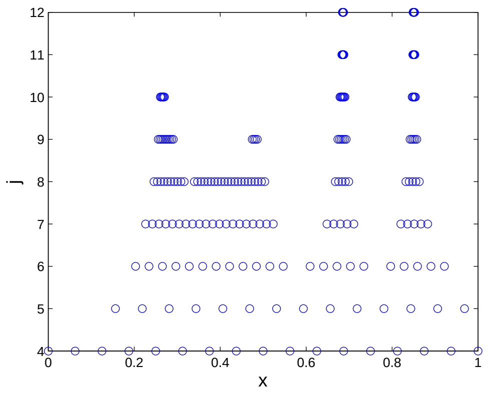
(b) Solution $\nu(x, 0.2)$.



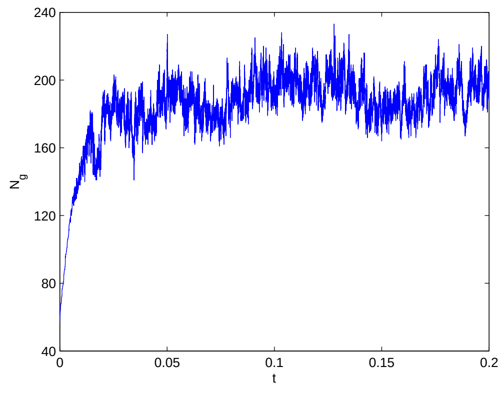
(c) Solution $p(x, 0.2)$.



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

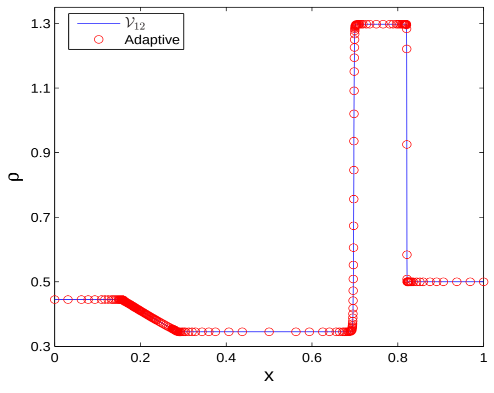


(e) Grid point distribution.

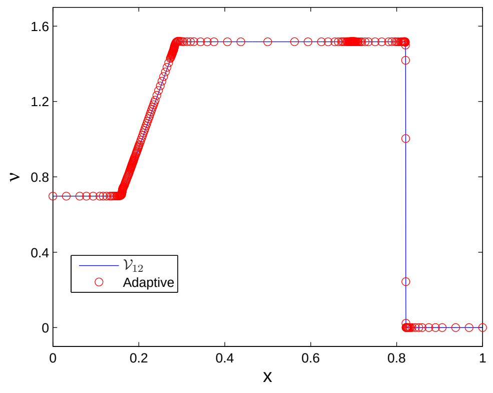


(f) Time evolution of grid points.

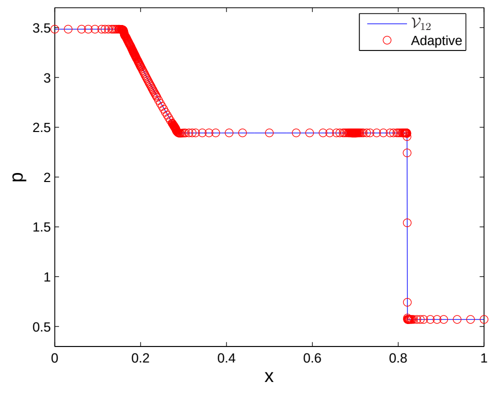
Figure 22: Example 12. Sod's problem. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 0.001,\ N_1 = N_2 = 2$.



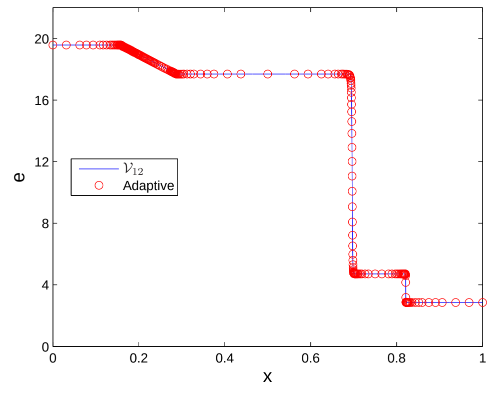
(a) Solution $\rho(x, 0.13)$.



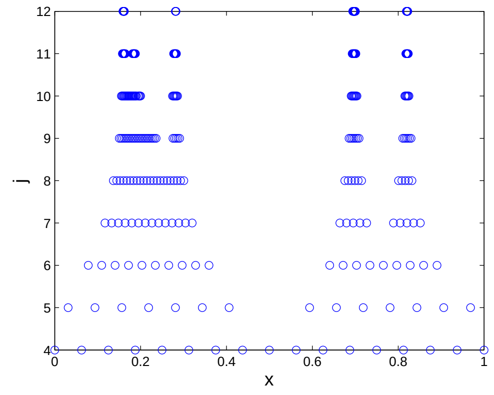
(b) Solution $\nu(x, 0.13)$.



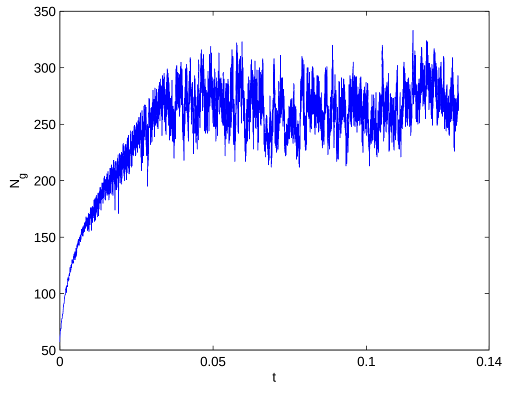
(c) Solution $p(x, 0.13)$.



(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

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

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,

$$\mathbf{x}(\tau_0) = \mathbf{x}_0, \tag{209}$$

$$e(\mathbf{x}(\tau_f), \tau_f) = 0. \tag{210}$$

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* $\operatorname{grad}(g_i)(\bar{\mathbf{x}})$ *for $i \in I$ and* $\operatorname{grad}(h_i)(\bar{\mathbf{x}})$ *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*

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

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

where $\operatorname{grad}(\cdot)$ *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:*

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

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

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

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

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

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

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

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

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

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

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]

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

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 $\mathsf{Grid}_{\text{int}}$ points belonging to the set $(\mathcal{V}_{\hat{J}} \cap [t_{j,k_i}, t_{j,k_i+1}]) \setminus \mathsf{Grid}_{\text{int}}$, 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 $\mathsf{Grid}_{\text{int}}$ the points $\{t_{\hat{J},k} : 2^{\hat{J}-j}k_i \leq k \leq 2^{\hat{J}-j}(k_i + 1)\} \setminus \mathsf{Grid}_{\text{int}}$.

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 $\mathsf{Grid}_{\text{iter}}$ and their function values in $\mathcal{X}_{\text{iter}}$ using $\mathcal{I}^p(\cdot, \mathcal{T}_{\mathsf{Grid}_{\text{iter}}}(\cdot))$.

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 $\mathsf{Grid}_{\text{new}} = \mathsf{Grid}_{\text{int}}$ and the corresponding function values are in the set $\Phi_{\text{new}} = \Phi_{\text{int}}$.

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 $\mathsf{Grid}_{\text{new}}$ (which will be the new initial guess $\mathcal{X}_{\text{iter}}$), reassign the set $\mathsf{Grid}_{\text{iter}}$ to $\mathsf{Grid}_{\text{new}}$, 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

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

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

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

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]

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

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 $h = \max_{0 \le i \le N}\{t_{i+1} - t_i\}$, 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

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

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 $|d_i(u^\star)| \le \epsilon$ 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

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

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

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

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

$$\begin{aligned}
\frac{\mathrm{d}h}{\mathrm{d}\tau} &= v, && (262)\\
\frac{\mathrm{d}v}{\mathrm{d}\tau} &= -g + \frac{T}{m}, && (263)\\
\frac{\mathrm{d}m}{\mathrm{d}\tau} &= -\frac{T}{I_{\mathrm{sp}}g}, && (264)
\end{aligned}$$

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 $J_{\mathrm{min}} = 3$ and $J_{\mathrm{max}} = 10$. 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

$$v(t_f) = 0.0165 \ (10^5 \text{ ft s}^{-1}), \tag{298}$$

$$\gamma(t_f) \quad \text{unspecified}, \tag{299}$$

$$\xi(t_f) = 0.75530/R \ (h(t_f) = 75,530 \text{ ft}), \tag{300}$$

$$\zeta(t_f) = 51.6912 \ (10^5 \text{ ft}). \tag{301}$$

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.

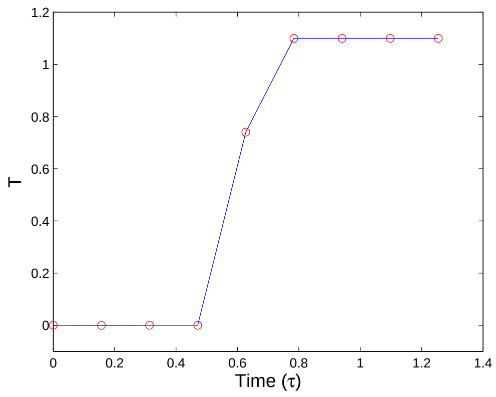
Binder et al. [24, 25, 26] work in the wavelet space by using the wavelet-Galerkin approach to discretize the optimal control problem into an NLP problem and use the local error analysis of the states and the wavelet analysis of the control profile to add or remove the wavelet basis functions. When one 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.

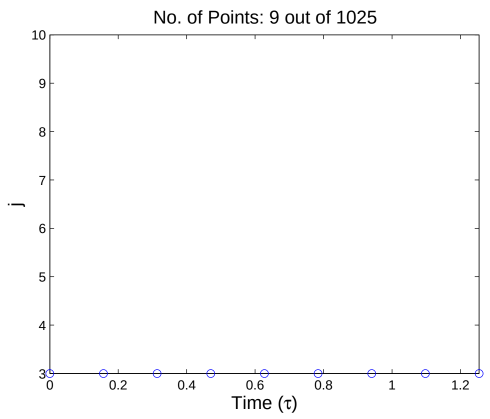
6.9 Summary

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.

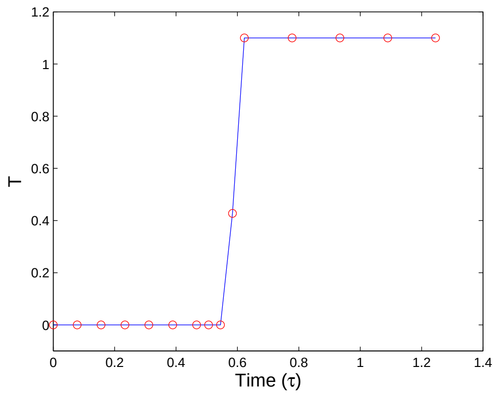
Next, we present two sequential trajectory optimization techniques for solving problems with moving targets and/or dynamically changing environments.



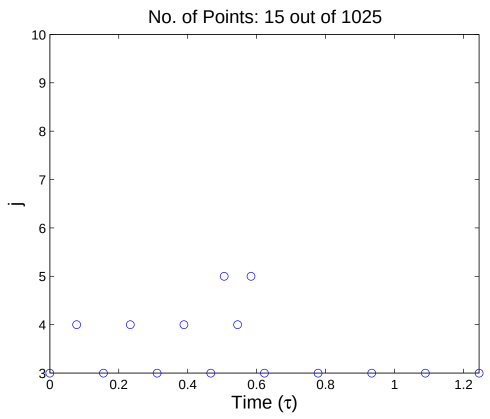
(a) Iteration 1: Time history of thrust T.



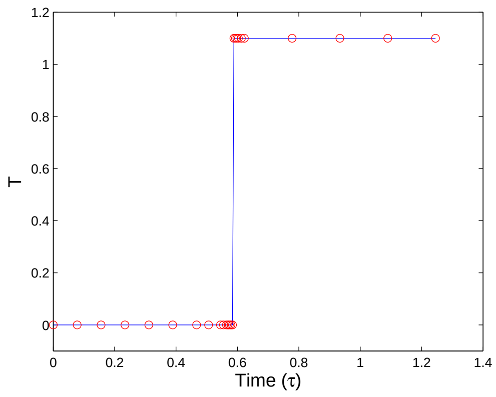
(b) Iteration 1: Grid point distribution.



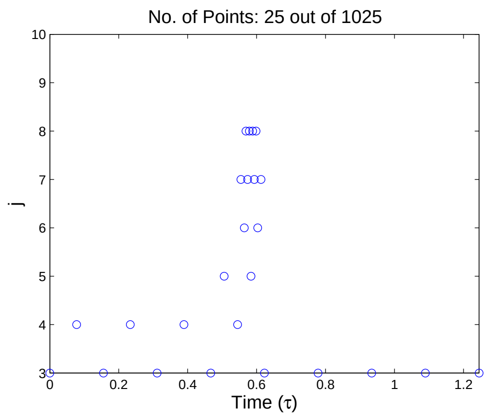
(c) Iteration 3: Time history of thrust T.



(d) Iteration 3: Grid point distribution.

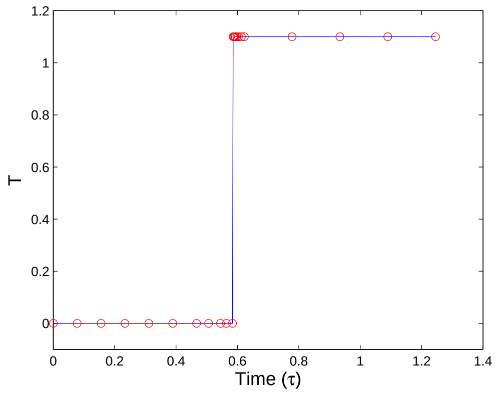


(e) Iteration 6: Time history of thrust T.

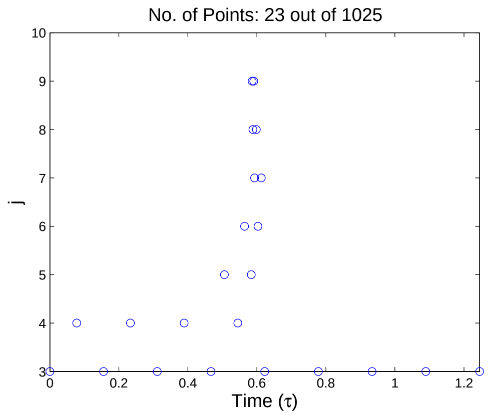


(f) Iteration 6: Grid point distribution.

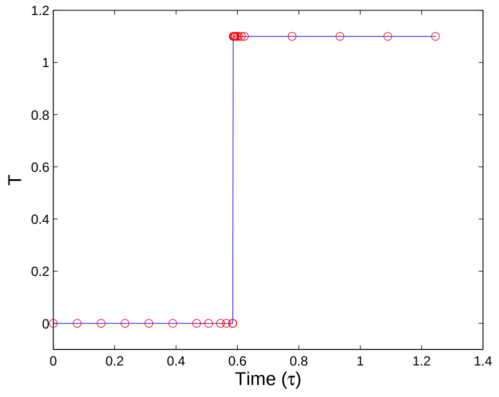
Figure 25: Example 13. Time history of thrust T along with the grid point distribution for iterations 1, 3, and 6.



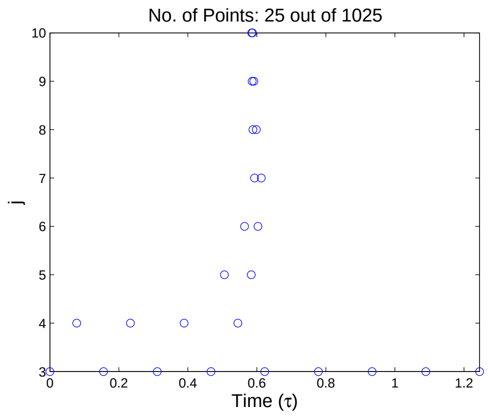
(a) Iteration 7: Time history of thrust T.



(b) Iteration 7: Grid point distribution.

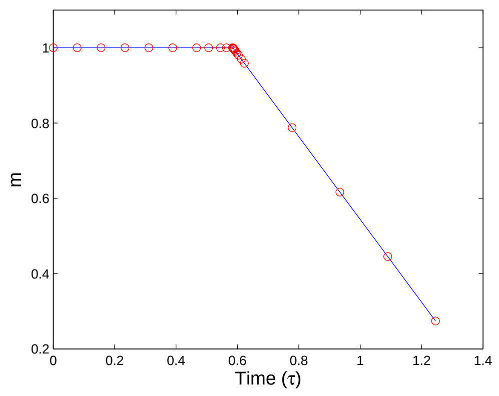


(c) Iteration 8: Time history of thrust T.

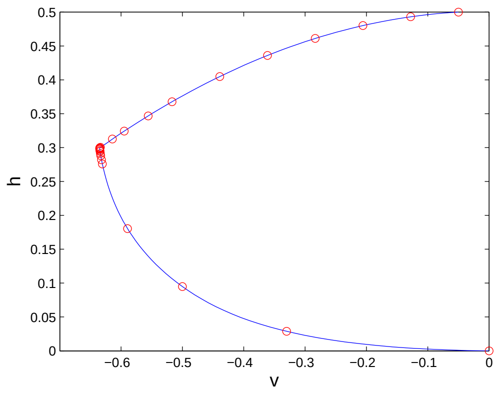


(d) Iteration 8: Grid point distribution.

Figure 26: Example 13. Time history of thrust T along with the grid point distribution for iterations 7 and 8.

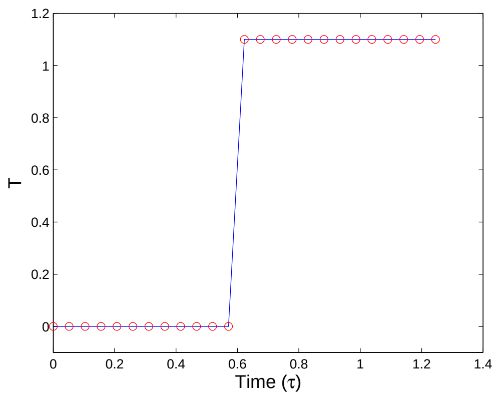


(a) Iteration 8: Time history of mass m.

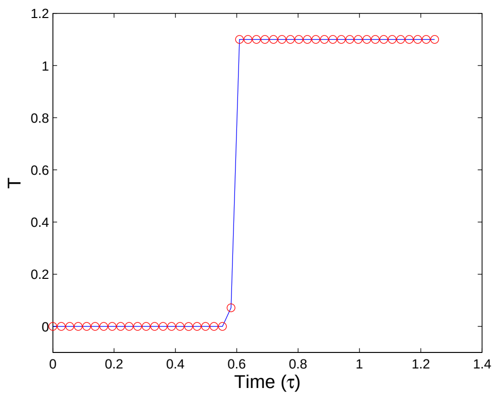


(b) Iteration 8: Phase portrait of v vs. h.

Figure 27: Example 13. Time history of mass m and the phase portrait of velocity v vs. altitude h for iteration 8.

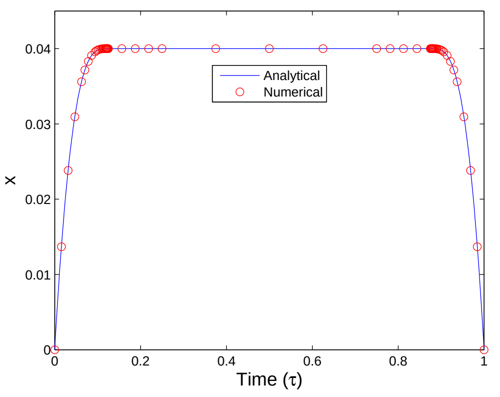


(a) No. of nodes used: 25.

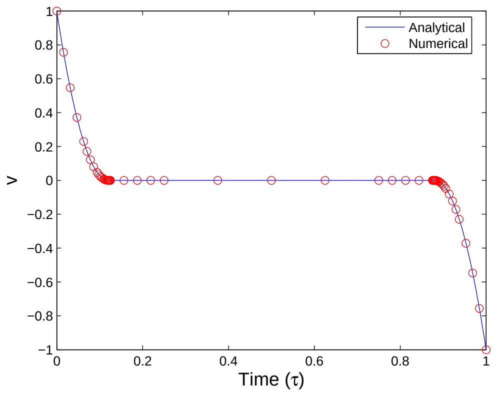


(b) No. of nodes used: 46.

Figure 28: Example 13. Time history of thrust T computed on a uniform mesh using an explicit fourth-order RK discretization.

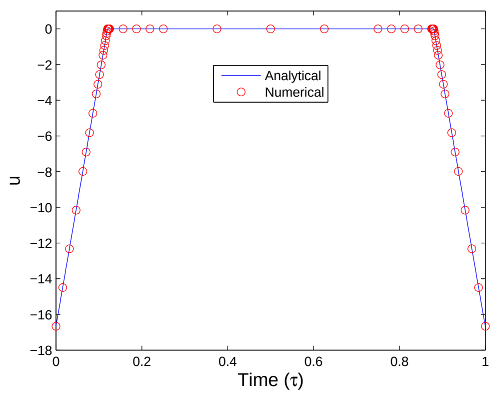


(a) Iteration 5: Time history of x.

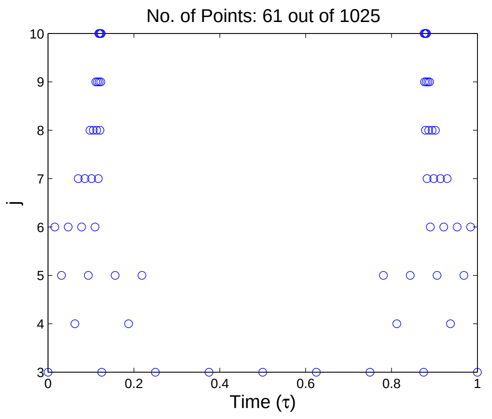


(b) Iteration 5: Time history of v.

Figure 29: Example 14: Time history of x, v at the final iteration of MTOA.

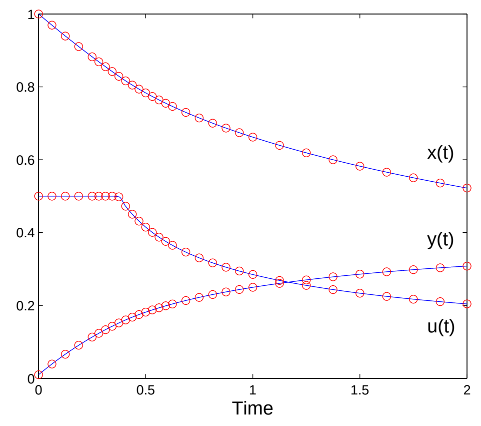


(a) Iteration 5: Time history of u.

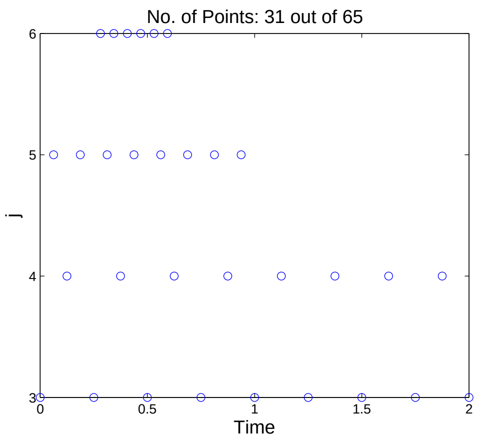


(b) Iteration 5: Grid point distribution.

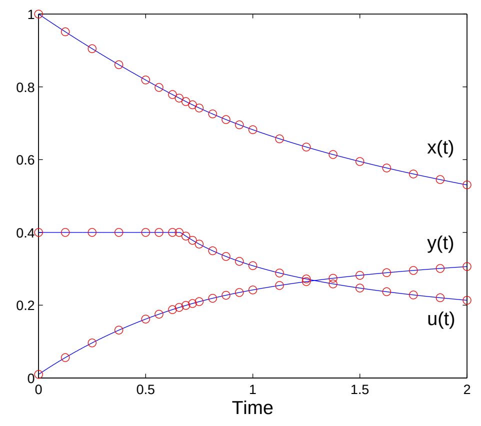
Figure 30: Example 14: Time history of u along with the grid point distribution at the final iteration of MTOA.



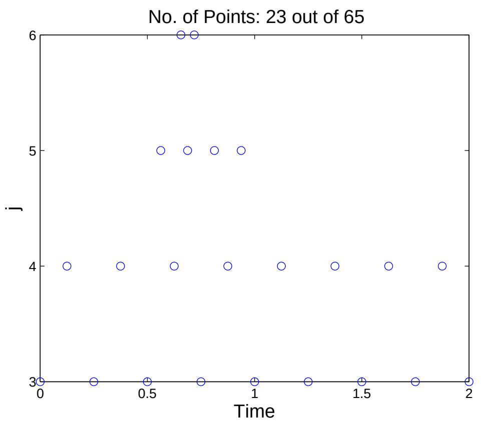
(a) Time history of x, y, and u for $u_{\max} = 0.5$.



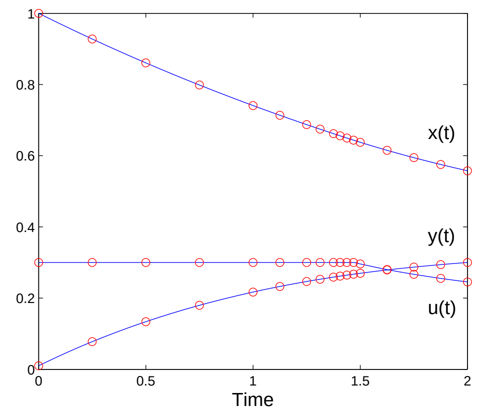
(b) Grid point distribution.



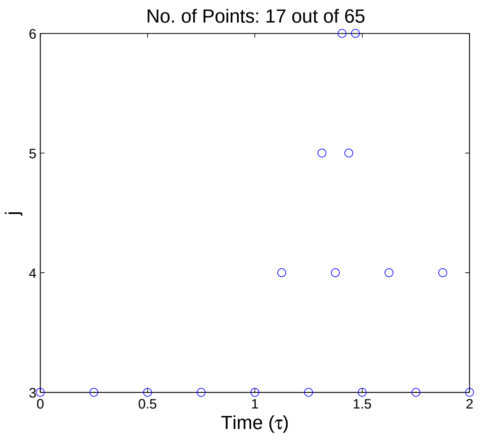
(c) Time history of x, y, and u for $u_{\max} = 0.4$.



(d) Grid point distribution.

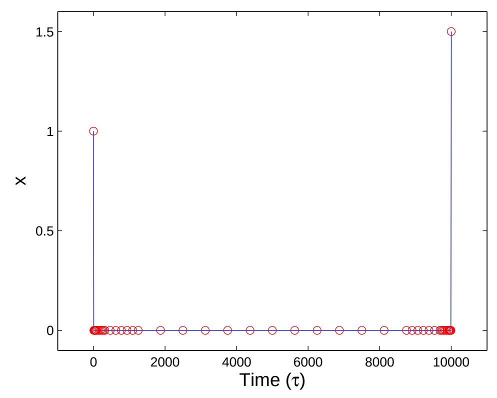


(e) Time history of x, y, and u for $u_{\max} = 0.3$.

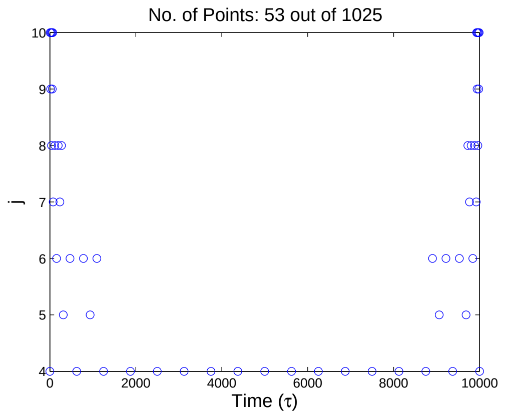


(f) Grid point distribution.

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.

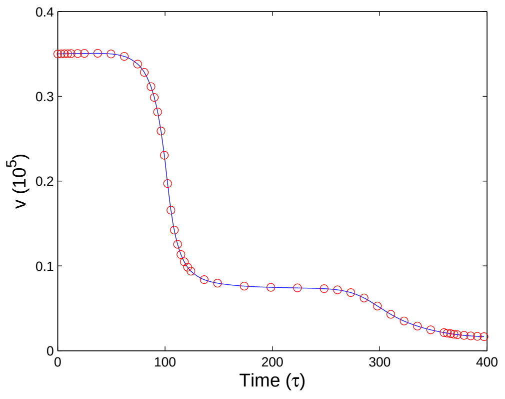


(a) Iteration 5: Time history of x.

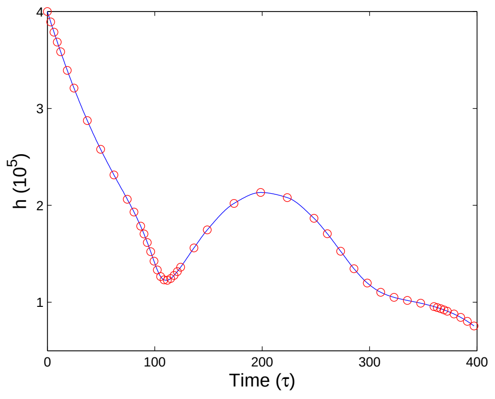


(b) Iteration 5: Grid point distribution.

Figure 32: Example 16: Time history of state x along with the grid point distribution at the final iteration of MTOA.

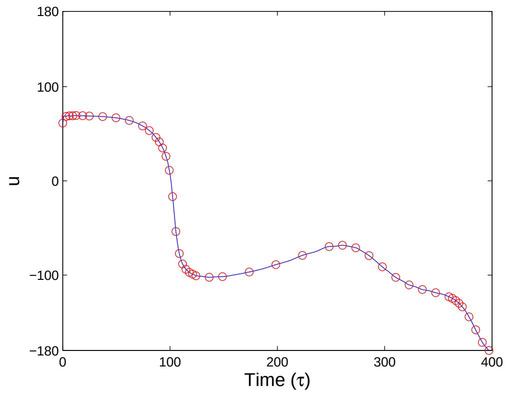


(a) Iteration 5: Time history of v.

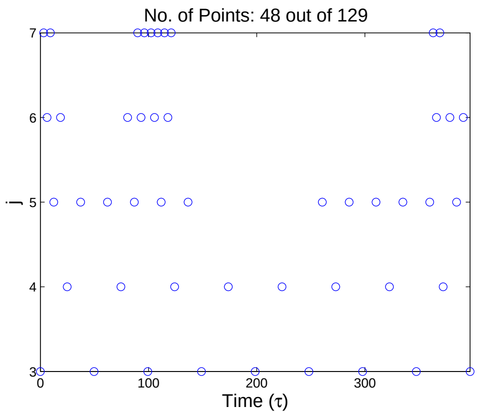


(b) Iteration 5: Time history of h.

Figure 33: Example 17: Problem A. Time histories of $v,\ h$ for $u_{\max} = 180$ at the final iteration of MTOA.

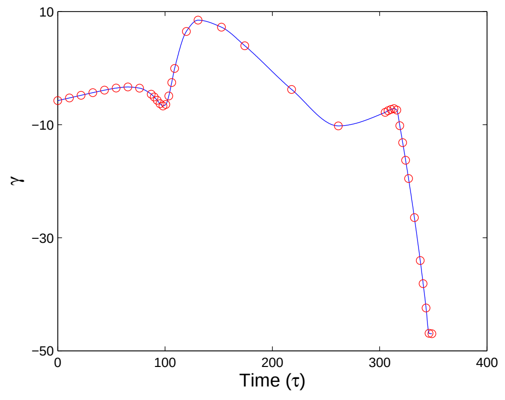


(a) Iteration 5: Time history of u.

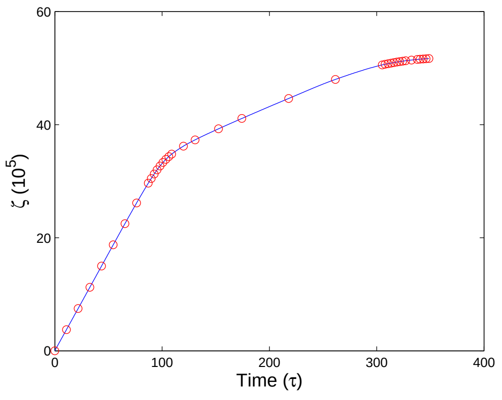


(b) Iteration 5: Grid point distribution.

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.

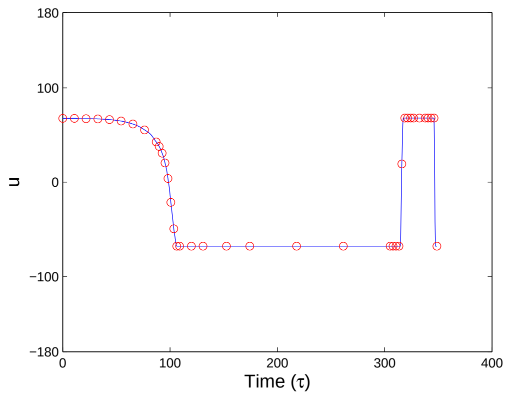


(a) Iteration 5: Time history of γ.

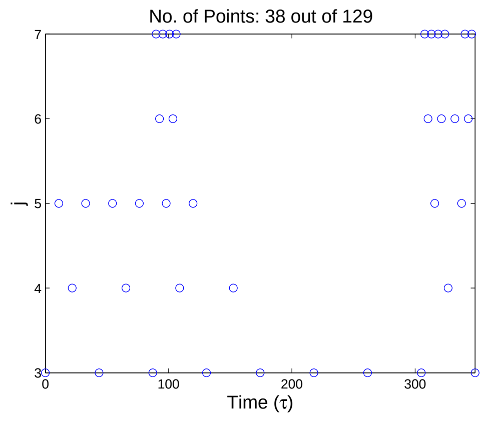


(b) Iteration 5: Time history of ζ.

Figure 35: Example 17: Problem A. Time histories of γ, ζ for $u_{\max} = 68$ at the final iteration of MTOA.

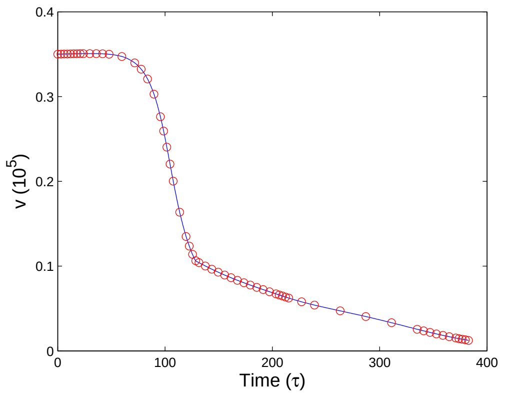


(a) Iteration 5: Time history of u.

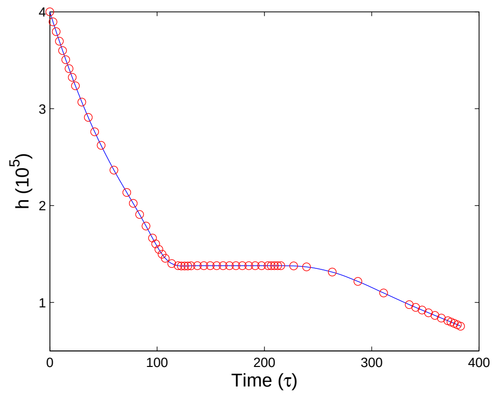


(b) Iteration 5: Grid point distribution.

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.

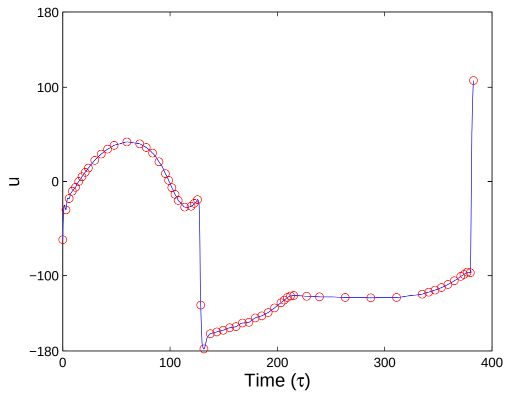


(a) Iteration 5: Time history of v.

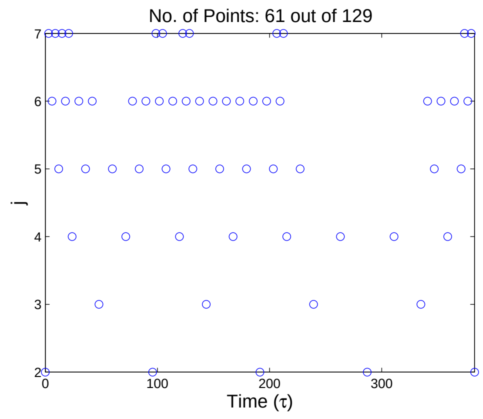


(b) Iteration 5: Time history of h.

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 $\Phi^i_{\text{iter}} = \{\phi_\ell(t_{j,k}) : \ell = 1, \dots, N_r,\ t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$.

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

$$\Phi_{\text{int}} = \{\phi_\ell(t_{J_{\min},k}) : \phi_\ell(t_{J_{\min},k}) \in \Phi^i_{\text{iter}},\ \forall\ t_{J_{\min},k} \in \mathcal{V}_{J_{\min}},\ \ell = 1, \dots, N_r\}, \tag{315}$$

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

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

$$\hat{T}_j = \{\hat{t}_{j,k_m} : \hat{t}_{j,k_m} \in \mathcal{W}_j \cap \mathsf{Grid}^i_{\text{iter}},\ \text{for } m = 1, \dots, N_{\hat{t}},\ 1 \le N_{\hat{t}} \le 2^j - 1\}. \tag{316}$$

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

$$\begin{aligned} \dot{x} &= V\cos\theta + u(x,y), && (318)\\ \dot{y} &= V\sin\theta + v(x,y), && (319) \end{aligned}$$

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,

$$\begin{aligned} u &= -Vy, && (320)\\ v &= 0, && (321) \end{aligned}$$

and the ship's velocity V is assumed to be unity. The path constraint is the width of the river, and we assume

$$0 \le x \le 6.8. \tag{322}$$

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 $\mathsf{Grid}^i_{\text{iter}} = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables ($\mathcal{X}^i_{\text{iter}}$). Fix $\bar{J} = J_{\min} - 1$. The proposed sequential trajectory optimization algorithm proceeds as follows:

Step 1. Solve the NLP problem on $\mathsf{Grid}^i_{\text{iter}}$ with the initial guess $\mathcal{X}^i_{\text{iter}}$ on the horizon $[t_0^i, 1]$. If $\mathsf{Grid}^i_{\text{iter}}$ 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

$$\begin{aligned}
R &= 209.0352 \ (10^5 \text{ ft}), \\
S/m &= 50,000 \ (10^{-5} \text{ ft}^2 \text{ slug}^{-1}), \\
\rho_0 &= 2.3769 \times 10^{-3} (\text{slug ft}^{-3}), \\
g &= 3.2172 \times 10^{-4} \ (10^5 \text{ ft s}^{-2}), \\
\beta &= 1/0.235 \ (10^{-5} \text{ ft}^{-1}).
\end{aligned}$$

The cost functional to be minimized that describes the total stagnation point convective heating per unit area is given by the integral

$$J(u) = \int_0^{\tau_f} 10v^3\sqrt{\rho}\,\mathrm{d}\tau. \tag{335}$$

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	`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

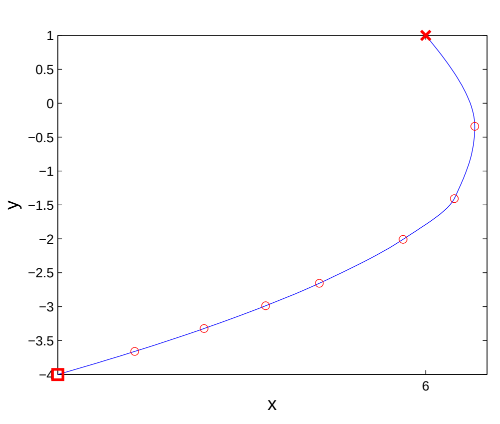
Both STOA I and STOA II have their own merits. STOA I will work for any user-specified time intervals ($\Delta\tau_{\mathrm{ro}}$), whereas the time intervals in STOA II are dyadic and fixed. On the other hand, STOA II takes full advantage of the multiresolution structure of the grid in the mesh refinement 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.

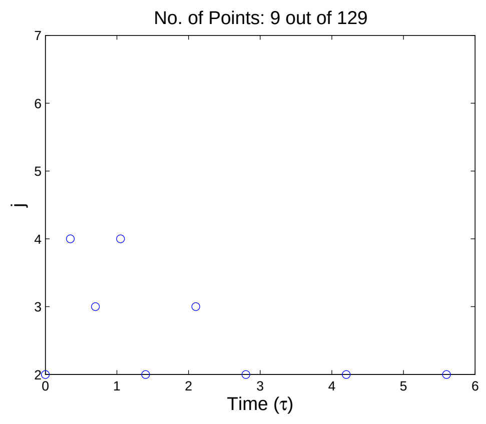
For both STOA I and STOA II, if the path constraints and the terminal constraints do not change drastically, the algorithm for each successive horizon converges pretty fast since the solution of the previous horizon is provided as an initial guess for solving the NLP problem on the current horizon. The 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.

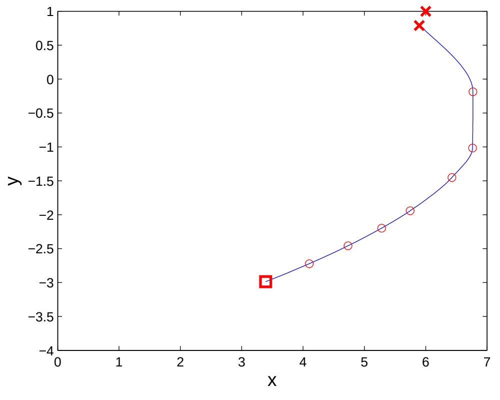
7.3 Summary

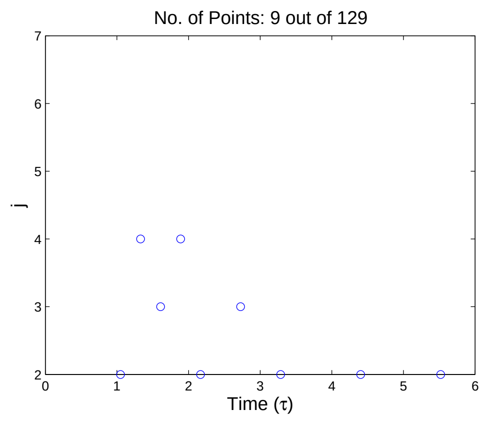
In this chapter, we have proposed two sequential trajectory optimization schemes to solve optimal control problems with moving targets and/or under dynamically changing environments in a fast and efficient way. The proposed algorithms autonomously discretize the trajectory with more nodes near the current 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.

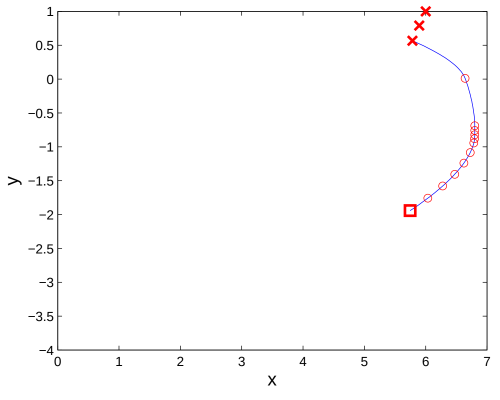


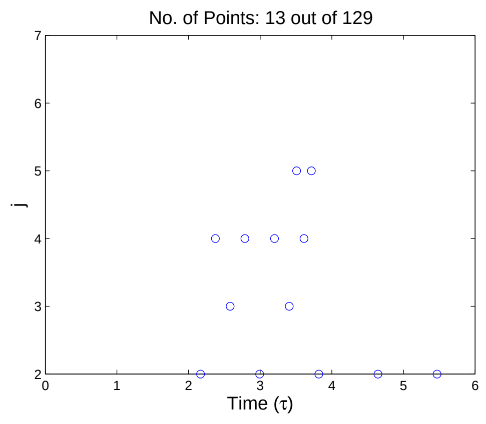
(b) Horizon 1. Grid point distribution.


(c) Horizon 2. Trajectory.



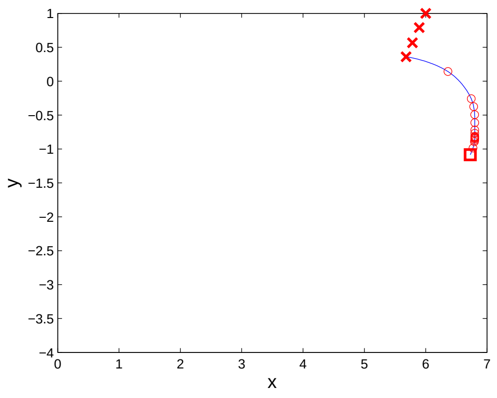
(d) Horizon 2. Grid point distribution.


(e) Horizon 3. Trajectory.

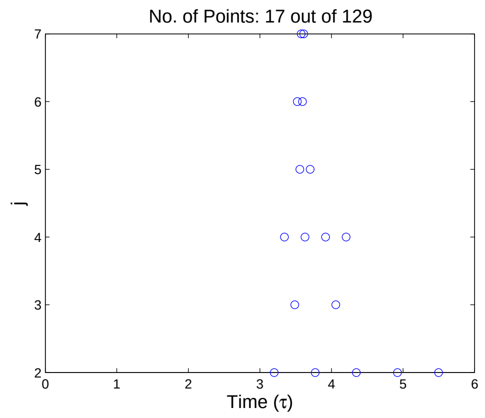


(f) Horizon 3. Grid point distribution.

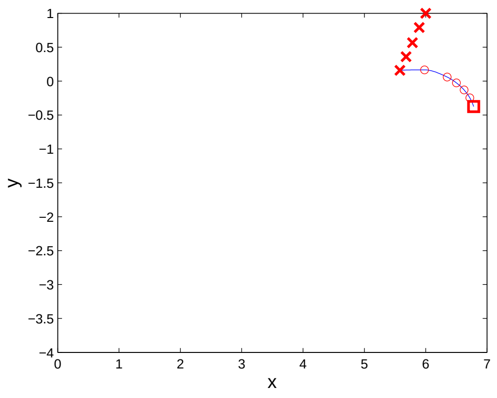
Figure 39: Example 18 (Target snapshots). Trajectory along with the grid point distributions for horizons 1, 2, and 3.



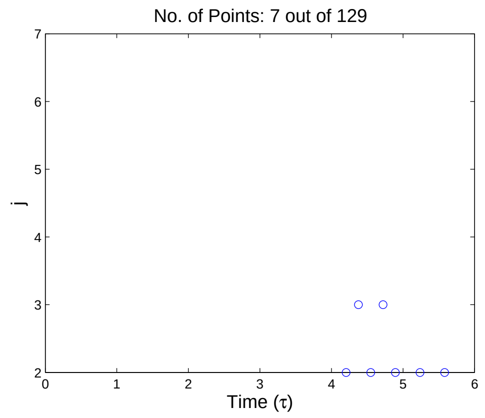
(a) Horizon 4. Trajectory.



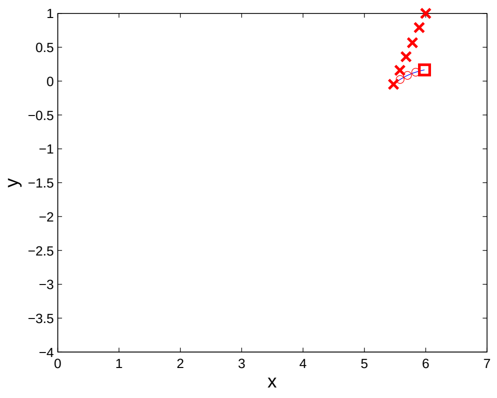
(b) Horizon 4. Grid point distribution.



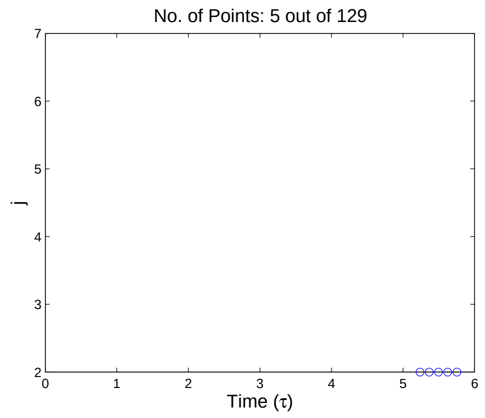
(c) Horizon 5. Trajectory.



(d) Horizon 5. Grid point distribution.

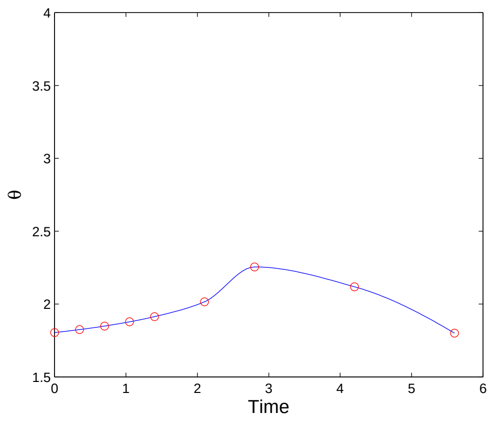


(e) Horizon 6. Trajectory.

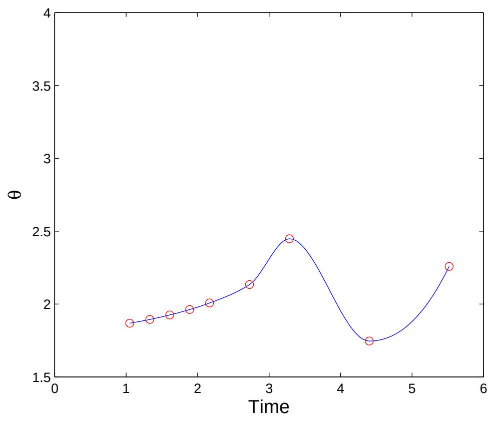


(f) Horizon 6. Grid point distribution.

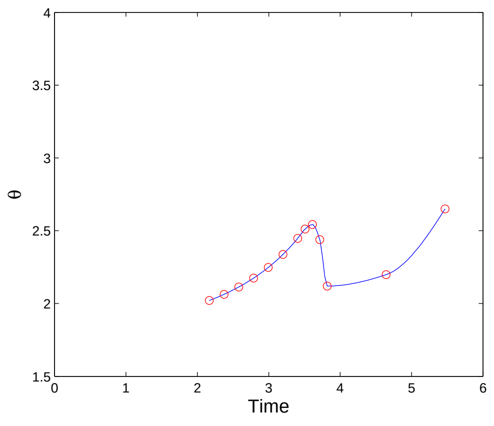
Figure 40: Example 18 (Target snapshots). Trajectory along with the grid point distributions for horizons 4, 5, and 6.



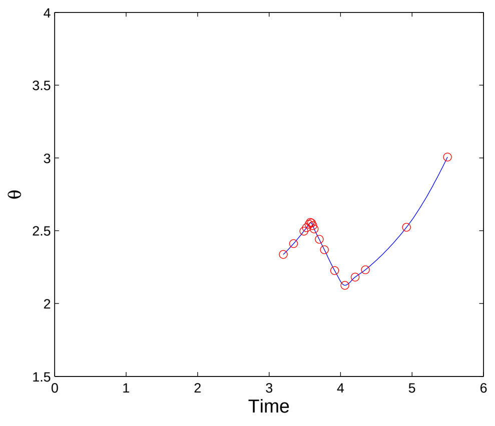
(a) Horizon 1.



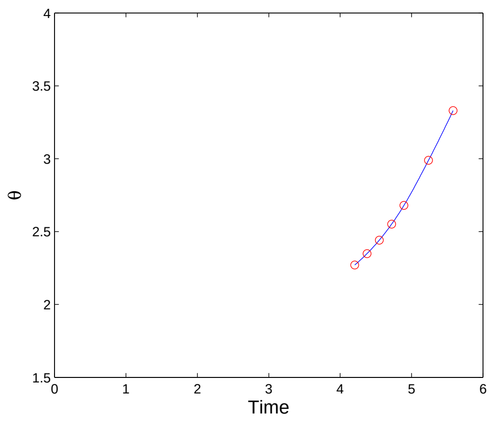
(b) Horizon 2.



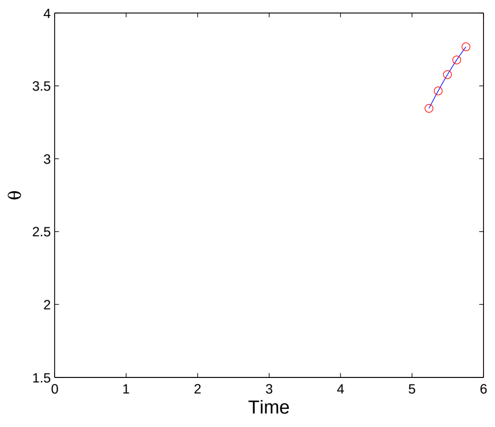
(c) Horizon 3.



(d) Horizon 4.

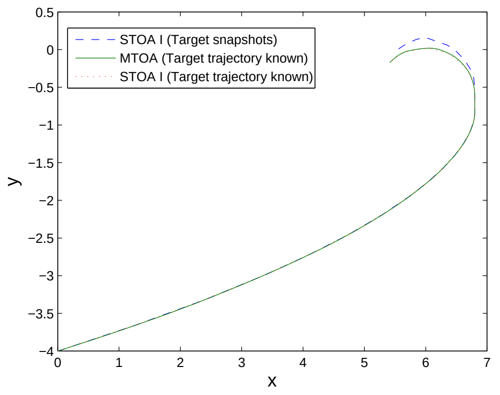


(e) Horizon 5.

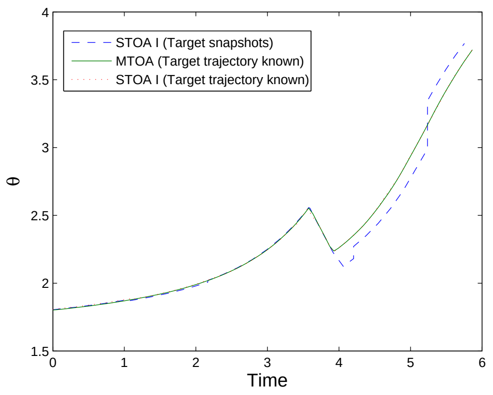


(f) Horizon 6.

Figure 41: Example 18 (Target snapshots). Time history of control θ for all horizons.

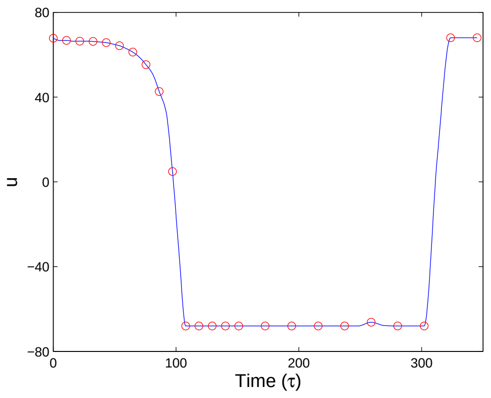


(a) Trajectory.

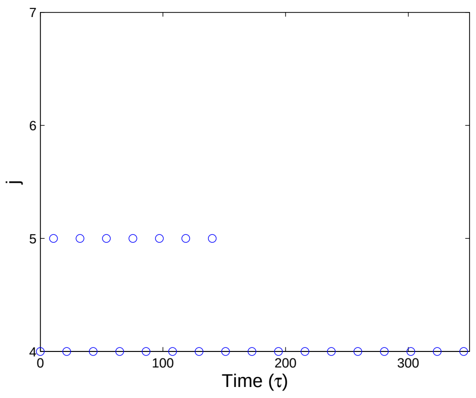


(b) Time history of control θ.

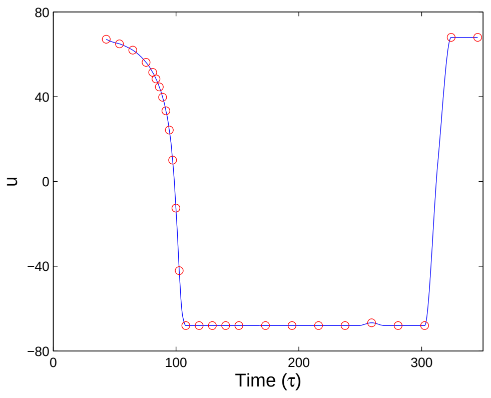
Figure 42: Example 18. Trajectory along with the time history of the control θ using three different multiresolution strategies.



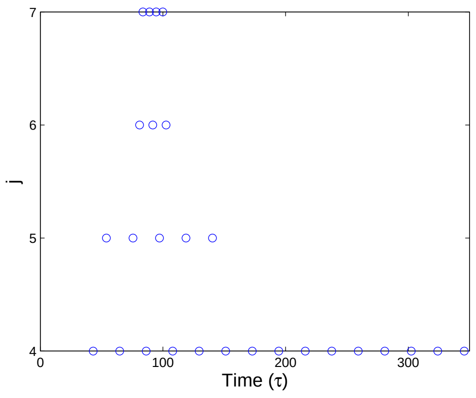
(a) Horizon 1. Control u.



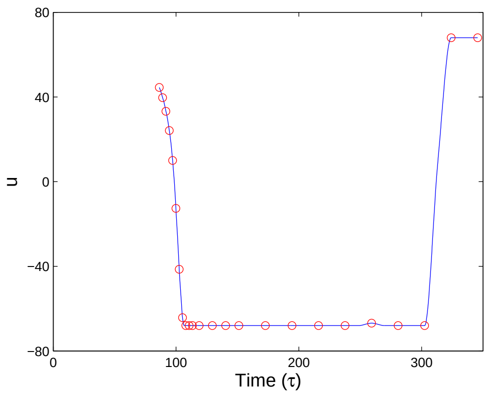
(b) Horizon 1. Grid point distribution.



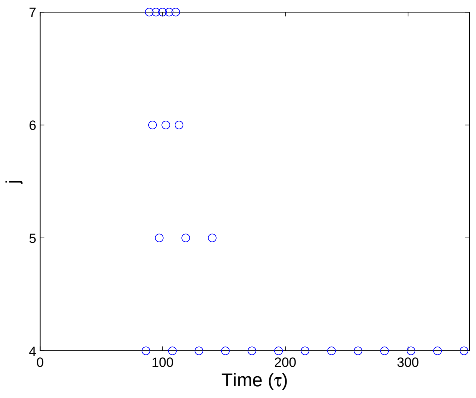
(c) Horizon 2. Control u.



(d) Horizon 2. Grid point distribution.

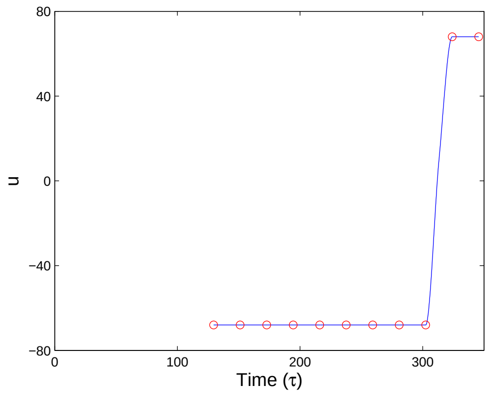


(e) Horizon 3. Control u.

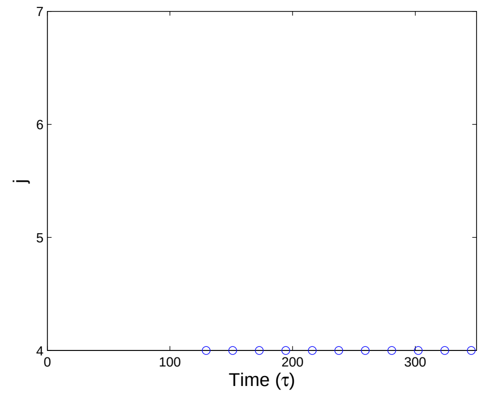


(f) Horizon 3. Grid point distribution.

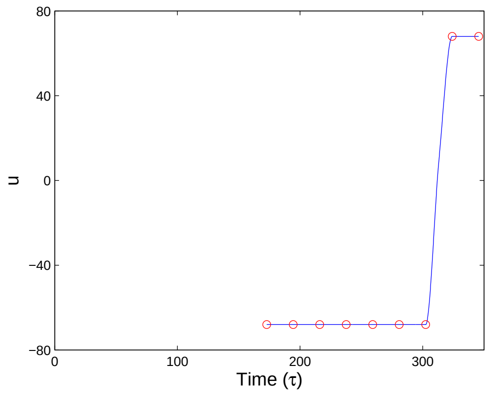
Figure 43: Example 19. Control time history and grid point distributions for horizons 1, 2, and 3.



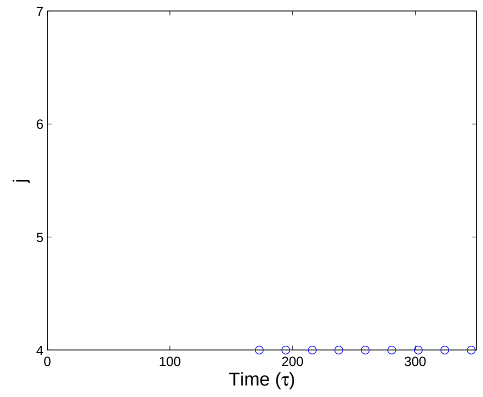
(a) Horizon 4. Control u.



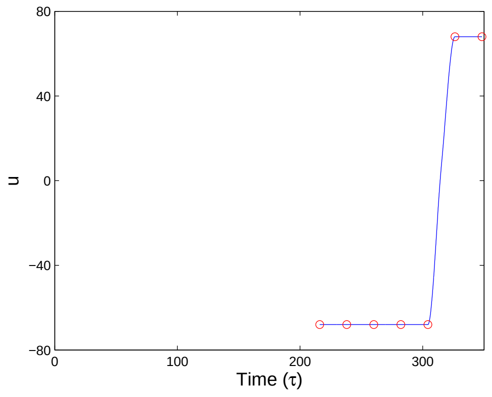
(b) Horizon 4. Grid point distribution.



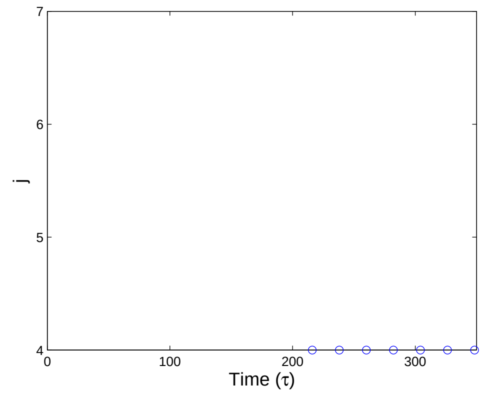
(c) Horizon 5. Control u.



(d) Horizon 5. Grid point distribution.

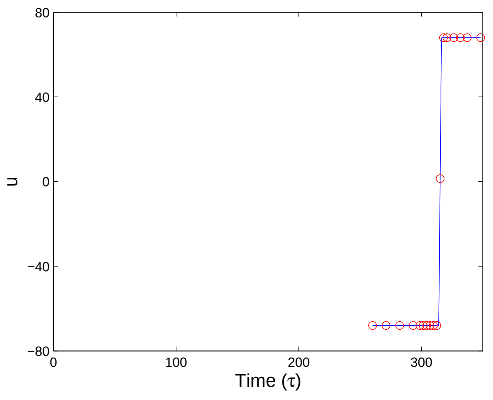


(e) Horizon 6. Control u.

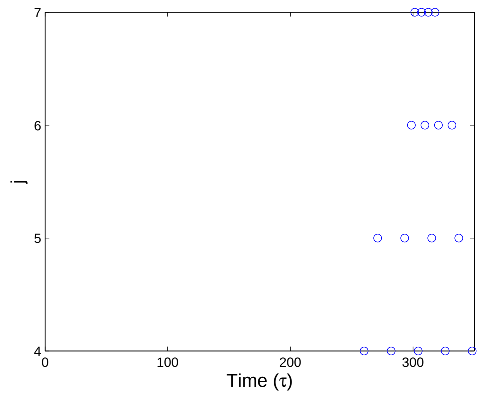


(f) Horizon 6. Grid point distribution.

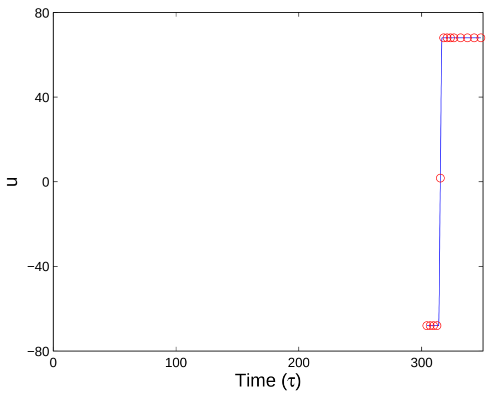
Figure 44: Example 19. Control time history and grid point distributions for horizons 4, 5, and 6.



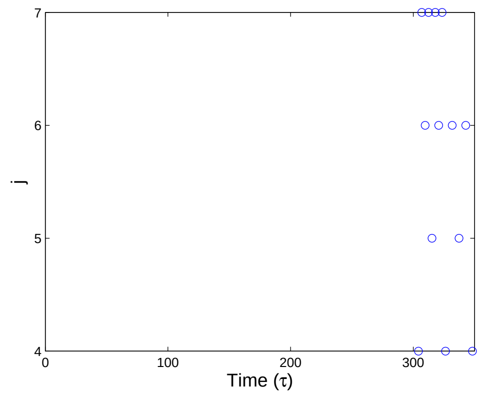
(a) Horizon 7. Control u.



(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 $x_0 \le \xi \le x_n$.

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

$$\begin{aligned} f(x) = f(x_0) &+ (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] && (359) \\ &+ (x - x_0)(x - x_1)(x - x_2) f[x, x_0, x_1, x_2]. && (360) \end{aligned}$$

Proceeding in this way we obtain

$$\begin{aligned} f(x) = f(x_0) &+ (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ &+ (x - x_0)(x - x_1)(x - x_2) f[x_0, x_1, x_2, x_3] + \cdots \\ &+ (x - x_0)(x - x_1) \cdots (x - x_n) f[x, x_0, x_1, \cdots, x_n]. && (361) \end{aligned}$$

This formula is called *Newton's form of interpolating polynomial.* Let

$$\begin{aligned} P_n(x) = f(x_0) &+ (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ &+ (x - x_0)(x - x_1)(x - x_2) f[x_0, x_1, x_2, x_3] + \cdots \\ &+ (x - x_0)(x - x_1) \cdots (x - x_{n-1}) f[x, x_0, x_1, \cdots, x_{n-1}]. && (362) \end{aligned}$$

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:

$$y_1^n(x) = 1 - \frac{n(n+1)}{2!}x^2 \quad + \quad \frac{(n-2)n(n+1)(n+3)}{4!}x^4 \tag{384}$$

$$- \quad \frac{(n-4)(n-2)n(n+1)(n+3)(n+5)}{6!}x^6 + \dots \tag{385}$$

$$y_2^n(x) = x - \frac{(n-1)(n+2)}{3!}x^3 + \frac{(n-3)(n-1)(n+2)(n+4)}{5!}x^5 - \dots \tag{386}$$

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

$$\text{Error}(h) = C_1 h^n + C_2 h^{n+m} + C_3 h^{n+2m} + \dots, \tag{393}$$

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]

$$\text{Error}(h) = C_1 h^2 + C_2 h^4 + C_3 h^6 + \dots. \tag{400}$$

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.

HIERARCHICAL PATH PLANNING AND CONTROL OF A SMALL FIXED-WING UAV: THEORY AND EXPERIMENTAL VALIDATION

A Thesis
Presented to
The Academic Faculty

by

Dongwon Jung

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

Georgia Institute of Technology
December 2007

HIERARCHICAL PATH PLANNING AND CONTROL OF A SMALL FIXED-WING UAV: THEORY AND EXPERIMENTAL VALIDATION

To my wife, whom I love,

and Elliot, my beloved son.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Panagiotis Tsiotras for providing me with the opportunity to study and conduct research at one of the best engineering schools in the country. His constant guidance and encouragement have motivated me to reach my utmost accomplishment towards professional maturity. For all your support and concern for me and my family, I am deeply indebted to you for potitive influence you have had on my life. In addition, I thank the other Ph.D. committee members Dr. Eric Corban, Dr. Eric Feron, Dr. Eric Johnson, and Dr. George Vachtsevanos for taking time from their busy schedules to review my thesis and providing valuable suggestions to improve the quality of the dissertation. I especially extend my appreciation to Dr. Eric Corban for your kind efforts and sincere concern in regards to my future plans.

This research work has been supported in part by NSF (award no. CMS-0510259) and by the Georgia Tech President's Undergraduate Research Awards (PURA) program. Their finalcial support is greatly appreciated. I would like to acknowledge all of my lab-mates and alumni of the Dynamic and Control Systems Lab, Dr. Haijun Shen, Dr. Xiping Zhang, Dr. Hyungjoo Yoon, Ancil Marshall, Sachin Jain, Atri Dutta, Efstathios Bakolas, Raghu Cowlagi, Dae-Min Cho for their help and valuable discussions from I have benefitted tremendously. In particular, I would like to give special thanks to Dr. Byungmoon Kim who has led me towards the spacecraft experimental research in my early days of Georgia Tech. In addition, I thank to Dr. Brian Wilson and Dr. Efstathios Velenis who spent time together at the office, Montgomery Knight building Room 111, listening to my problems, successes, and offering advice. During the early phase of this research work, I would like to acknowledge the contributions of Emmanuel J. Levy, Debao Zhou, Rebecca Fink, Terry Williams, Jonathan

Moshe, Andrew Earl, Eiji Ozawa, and Jayant Ratti with whom I was able to carry on this research work. Also, I thank to Dr. Ravi Doraiswami, PRC, Georgia Tech for advising on the the PCB fabrication and Iven Cornery, Atlanta RC club for willingly assisting me during the flight tests as a remote pilot.

This acknowledgement is incomplete until I mention other colleagues and alumni in the control group of AE, Jeong Hur, Dr. Bong-Jun Yang, Dr. Nakwan Kim, Dr. Suresh Kannan, Dr. Yoonghyun Shin, Dr. Ali Kutay, Dr. Ramachandra Sattigeri, Jincheol Ha, Seung-Min Oh, Yoko Watanabe, Nimrod Rooz, Jongki Moon, Keumjin Lee, Allen Wu, Clauss Christmann, Jonathan Muse, Kilsoo Kim, Henrik B. Christophersen, Wayne J. Pickell, and Jeongjun Im, with whom I have enjoyed the time at Georgia Tech and have learned from interactions. I give special thanks to Dr. Bong-Jun Yang who gave valuable comments and suggesstions on my dissertation. At this point I would like to thank all of my Korean fellow students in AE who have supported and encouraged me morally during the years of Georgia Tech. I would like to recognize the efforts of Vivian O'Neal, Andrew Carignan and the other dedicated workers in AE, who provided administrative supports and research assistance.

I would like to express my deep gratitude with all my heart to my parents. Even from thousands of miles away, your endless support and encouragement have strengthened me to complete this long journey for Ph.D. I am also thankful to my parents-in-law and other family members who have always wished for my success. I don't think I could have made the journey without their constant support throughout my whole academic years. A special thank must be given to my beloved son Elliot Hyunje Jung who was born on October 29^{th}, 2007. Since he has become our new family member as an unborn child, he has been a great source of inspiration to me, making even the most stressful days brigher. Last but certainly not least, I would like to give thanks to my lovely wife, Eun Ju "Agatha" Song. Your embrace lends me courage and strength, your smile makes me happy, your passion inspires me, your devotion spurs me to work hard, and your love fulfills me. Through you I have been centered in all respects of our upcoming life for a harmonious family. You are the best thing that has ever happened to me.

Dongwon “Thomas” Jung

November 13, 2007

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

$\{u_k\}$	Non-decreasing knot sequence.
α	Angle of attack.
b	Span length.
$\bar{c}$	Aerodynamic reference chord length.
$\bar{L}$	Aerodynamic moment component about the body x-axis.
$\bar{L}_w$	Aerodynamic roll moment about the wind x-axis.
$\bar{q}$	Dynamic pressure.
β	Side slip angle.
$\boldsymbol{e}_L$	Left bounding envelope of a two dimensional B-spline curve.
$\boldsymbol{e}_R$	Right bounding envelope of a two dimensional B-spline curve.
$\boldsymbol{\ell}_L$	Two dimensional left channel polygon.
$\boldsymbol{\ell}_R$	Two dimensional right channel polygon.
C_D	Drag coefficient.
C_ℓ	Rolling moment coefficient.
χ	Course angle, inertial speed heading.
C_L	Lift coefficient.
C_m	Pitching moment coefficient.
C_n	Yawing moment coefficient.
C_y	Side force coefficient.
D	Aerodynamic drag force.
δ_a	Aileron deflection angle.
$\delta(\cdot)$	Approach angle.
δ_e	Elevator deflection angle.
δ_r	Rudder deflection angle.
e_d	Cross track error with respect to the path.

ℓ	Control polygon of B-spline.
ϵ_p	White noise process for roll rate gyro random walk process.
ϵ_r	White noise process for yaw rate gyro random walk process.
ϵ_w	White noise process for the wind speed bias random walk process.
e_s	Along track error with respect to the path.
η_E	Measurement noise of GPS position fix along the East direction.
η_h	Measurement noise of the altitude.
η_N	Measurement noise of GPS position fix along the North direction.
η_p	Roll rate gyro measurement noise.
η_r	Yaw rate gyro measurement noise.
η_w	Measurement noise of the airspeed.
FLWT	Fast lifting wavelet transform.
g	Earth's gravity constant.
GPS	Global Positioning System.
GUI	Graphical User Interface.
h	Altitude of the UAV.
HILS	Hardware in the loop simulation.
J	Advance ratio.
J_x	Moment of inertia of mass in the body x-axis.
J_{xz}	Product of inertia of mass in terms of the body x-y plane.
J_y	Moment of inertia of mass in the body y-axis.
J_z	Moment of inertia of mass in the body z-axis.
κ	Curvature of the curve.
L	Aerodynamic lift force.
λ_ϕ	Time constant of the first-order roll angle dynamics.
M	Aerodynamic moment component about the body y-axis.
$\mathcal{C}_d$	Multiresolution cell decomposition.

$\mathcal{F}$	Obstacle free configuration space.
$\mathcal{G}$	Topological graph structure.
$\mathcal{M}$	Collection of m integer distinct risk measure level.
$\mathcal{N}(\mathsf{x}, r)$	Neighborhood of the location x with distance r.
$\mathcal{O}$	Obstacle space.
$\mathcal{W}$	Two dimensional world enviroment.
$\mathsf{cell}_{\mathcal{G}}(v)$	Center of the cell correponding to the node v of the graph $\mathcal{G}$.
$\mathsf{node}_{\mathcal{G}}(\mathsf{x})$	Node of the graph $\mathcal{G}$ corresponds to the location x.
rm	Risk measure function at the location $\mathsf{x} = (x, y)$.
M_w	Aerodynamic pitch moment about the wind y-axis.
N	Aerodynamic moment component about the body z-axis.
N_j^d	jth B-spline basis function of degree d.
ν	Auxiliary control input for backstepping controller.
N_w	Aerodynamic yaw moment about the wind z-axis.
Ω	Rotational speed of propeller.
ω	Heading rate of the UAV.
ω_e	Error state for the heading rate.
p	Angular velocity component about the body x-axis.
p_b	Roll rate gyro bias.
p^D	Down position of the airplane in the NED frame.
p^E	East position of the airplane in the NED frame.
ϕ	Roll attitude angle.
p^N	North position of the airplane in the NED frame.
ψ	Heading angle.
q	Angular velocity component about the body y-axis.
r	Angular velocity component about the body z-axis.
r_b	Yaw rate gyro bias.

R_p	Propeller radius.
S	Aerodynamic reference area.
s	Arc length parameter.
θ	Pitch attitude angle.
$\mathsf{\mathbf{U}}$	Velocity component in the body x-axis.
u_j^*	Greville abscissae.
V	Velocity component in the body y-axis.
V_T	Total airspeed in the wind x-axis.
V_w	Wind speed state.
W	Velocity component in the body z-axis.
$\widehat{\lambda}_\phi$	Estimate of the λ_ϕ.
$\widetilde{\chi}$	Error course angle.
X_A	Aerodynamic force component in the body x-axis.
X_T	Thrust force component in the body x-axis.
Y	Aerodynamic side force.
Y_A	Aerodynamic force component in the body y-axis.
Y_T	Thrust force component in the body y-axis.
Z_A	Aerodynamic force component in the body z-axis.
Z_T	Thrust force component in the body z-axis.

SUMMARY

Recently there has been a tremendous growth of research emphasizing control of unmanned aerial vehicles (UAVs) either in isolation or in teams. As a matter of fact, UAVs increasingly find their way into military and law enforcement applications (e.g., reconnaissance, remote delivery of urgent equipment/material, resource assessment, environmental monitoring, battlefield monitoring, ordnance delivery, etc.). This trend will continue in the future, as UAVs are poised to replace the human-in-the-loop during dangerous missions. Civilian applications of UAVs are also envisioned such as crop dusting, geological surveying, search and rescue operations, etc.

In this thesis we propose a new online multiresolution path planning algorithm for a small UAV with limited on-board computational resources. The proposed approach assumes that the UAV has detailed information of the environment and the obstacles only in its vicinity. Information about far-away obstacles is also available, albeit less accurately. The proposed algorithm uses the fast lifting wavelet transform (FLWT) to get a multiresolution cell decomposition of the environment, whose dimension is commensurate to the on-board computational resources. A topological graph representation of the multiresolution cell decomposition is constructed efficiently, directly from the approximation and detail wavelet coefficients. Dynamic path planning is sequentially executed for an optimal path using the $\mathcal{A}^*$ algorithm over the resulting graph. The proposed path planning algorithm is implemented on-line on a small autopilot. Comparisons with the standard $\mathcal{D}^*$-lite algorithm are also presented.

We also investigate the problem of generating a smooth, planar reference path from a discrete optimal path. Upon the optimal path being represented as a sequence of cells in square geometry, we derive a smooth B-spline path that is constrained inside a channel that is induced by the geometry of the cells. To this end, a constrained optimization problem is formulated by setting up geometric linear constraints as well as boundary conditions. Subsequently, we construct B-spline path templates by solving a set of distinct optimization problems. For application in UAV motion planning, the path templates are incorporated to replace parts of the entire path by the smooth B-spline paths. Each path segment is stitched together while preserving continuity to obtain a final smooth reference path to be used for path following control.

The path following control for a small fixed-wing UAV to track the prescribed smooth reference path is also addressed. Assuming the UAV is equipped with an autopilot for low level control, we adopt a kinematic error model with respect to the moving Serret-Frenet frame attached to a path for tracking controller design. A kinematic path following control law that commands heading rate is presented. Backstepping is applied to derive the roll angle command by taking into account the approximate closed-loop roll dynamics. A parameter adaptation technique is employed to account for the inaccurate time constant of the closed-loop roll dynamics during actual implementation.

Finally, we implement the proposed hierarchical path control of a small UAV on the actual hardware platform, which is based on an 1/5 scale R/C model airframe (Decathlon) and the autopilot hardware and software. Based on the hardware-in-the-loop (HIL) simulation environment, the proposed hierarchical path control algorithm has been validated through on-line, real-time implementation on a small micro-controller. By a seamless integration of the control algorithms for path planning, path smoothing, and path following, it has been demonstrated that the UAV equipped with a small autopilot having limited computational resources manages to accomplish the path control objective to reach the goal while avoiding obstacles with minimal human intervention.

CHAPTER I

INTRODUCTION

1.1 Motivation

For decades, unmanned aerial vehicles (UAVs) have been considered for replacing human pilots during various missions. The applications of UAVs are increasing at a fast pace in military and in law enforcement missions (e.g., reconnaissance, remote delivery of urgent equipment/material, resource assessment, environmental monitoring, battlefield monitoring, ordnance delivery, etc [28, 113, 93]). Civilian applications of UAVs are also envisioned (crop dusting, geological surveying, search and rescue operations, etc). In particular, UAVs play an important role in replacing the human-in-the-loop during dangerous missions in hazardous or hostile environment.

A typical mission of a UAV is characterized by the requirement of navigating through or close to way points specified by the mission profile. The way points are usually specified a priori, and the mission must be completed satisfying certain criteria provided by the mission scenario. These criteria are minimum time, minimum fuel, minimum danger exposure, and so on. Assuming the domain of interest is fully characterized a priori, an off-line trajectory planning optimization can be used to obtain a satisfactory solution. However, in case of UAVs flying over hostile environment, where the domain is not known completely beforehand, UAVs necessarily require more advanced navigation and guidance capabilities [143, 13, 18]. Both trajectory design (planning) and trajectory tracking (control) tasks should be completely automated so that UAVs fly safely while avoiding obstacles and minimizing the exposure to danger. In addition, pop-up threats that might be encountered during missions need to be appropriately addressed during trajectory design and execution.

In line with the previous observations, it is natural to consider the trajectory planning problem as having two distinct objectives: The planned trajectory must eventually reach a final goal (after passing through the way points) skirting known obstacles, and should take into account any local environment variation, such as pop-up threats, by an online update of the planned trajectory. The former is considered as a global trajectory planning algorithm and the latter is as a local trajectory planning algorithm. The global trajectory planning determines overall performance of the generated trajectory whereas the local planning guarantees a feasible trajectory that can be followed by the UAV under dynamic constraints. The idea of combining both a global and a local algorithm seamlessly when designing a whole trajectory is similar to what a human pilot does. Given a final destination, the pilot plans the overall trajectory to be followed toward the destination avoiding known obstacles. Because distant information provided to the pilot is possibly insufficient to the accuracy than proximal information, he puts less emphasis on the distant information and utilizes more reliable local information to decide a feasible trajectory that keeps the airplane safe and flyable. In this work we plan to develop trajectory design and trajectory tracking algorithms that imitate 'human intelligence' in order to successfully complete complicated missions assigned to autonomous vehicles.

Assuming perfect knowledge of the environment while designing the trajectory, the global trajectory planner yields a better solution than the local planner and avoids local minima. However, the computational cost tends to increase as the problem size gets bigger. In contrast, a local planner can reduce the number of computation, but at the cost of a solution which often results in a locally entrapped trajectory. Thus, the combination of these two methods offers the best compromise between feasibility and optimality, while minimizing the computational cost so that they could be implemented in realtime.

1.2 Literature Review

1.2.1 Geometric path planning

Guidance, navigation, and control of mobile agents has been an important research topic for several decades. The applications are enormous in a variety of areas such as marine craft [108, 40], underwater vehicles [3, 74], unicycle type mobile robots [66], and unmanned aerial vehicles [102, 5]. The navigation control algorithms reported in the literature implement way-point following schemes assuming the vehicles are given a sequence of way-points a priori in the region of interest. These way points are connected in order to generate smooth path segments [61, 122] in a manner that they preserve the continuity of the curvature between line and arc segments, while minimizing the maximum curvature on the curve. A series of cubic splines was employed in Ref. [56] to connect the straight line segments in a near-optimal manner, whereas the algorithm presented in Ref. [6] yields extremal trajectories that transition between straight-line path segments smoothly in a time-optimal fashion.

In two-dimensional vehicle motion subject to non-holonomic constraints, such as a unicycle type mobile robot, it has been proved in Ref. [38] and [17] that for given initial and final configurations $p_0 = (x_0, y_0, \theta_0)$ and $p_f = (x_f, y_f, \theta_f)$, respectively, the optimal shortest path exists and is one among six types of paths: the paths consisting of at most three parts which are either straight line segments or arcs of circle of radius R if the vehicle has a maximum turning rate constraint. This set of path elements is termed as "Dubins path" and has inspired many researchers on how to design a feasible path by connecting way-points. Based on this idea, Bui and Souères [21] proposed new optimality conditions on these paths by computing a partition space from the two-dimensional plane of the configuration space, Yang and Kapila [151] exploited parameter optimization methods to solve optimal path planning problems. A similar geometry-based path planning algorithm is also reported in Ref. [80]. More recently, a three-dimensional extension of Dubins path was developed in Ref. [126].

Although the path planning strategies discussed above yield the optimal (shortest) paths between way-points for non-holonomic vehicles and yield easy implementation, one should note that they do not explicitly deal with obstacles possibly existing along the path segments. This disadvantage becomes apparent in the case when the vehicle navigates in an environment filled with obstacles where the designed path segments could collide with the obstacles. As discussed in Ref. [80] it is then necessary to consider a path planning algorithm that takes into account obstacles avoidance.

1.2.2 Obstacle avoidance path planning

One technique for obstacle avoidance is to use the "artificial potential field" concept. Khatib [65] presented the earlier result of real-time obstacle avoidance for mobile robots using this concept. In this method, a mobile robot moves in a potential field such that the goal is modeled as an attractive force and the obstacles and other vehicles are modeled as repelling forces. In Ref. [9], local minima of a potential field are connected in order to build an incremental graph, and the graph is searched to attain the final goal configuration. In Refs. [85, 14], an artificial potential field serves as a local holonomic planner to avoid obstacles, of which the output is modified in a least-squares sense to generate actual commands for the desired motion. Although potential field methods can produce smooth and dynamically feasible trajectories at low computational cost, they have several drawbacks. Detecting and avoiding the obstacles are mostly influenced by proximate obstacles, and the vehicle can get trapped in a local minimum. Another issue arises from the fact that obstacles are not modeled as hard constraints. The potential fields are typically modeled as continuous and differentiable functions, leading to an imprecise description of the obstacle's shape and dimension. This restriction could result in unsatisfactory obstacle avoidance when the vehicle maneuvers through tight environment.

In contrast to the local potential field techniques, the so-called global path planning algorithms, by which the overall workspace is accounted for during motion planning, can overcome not only the local minima problem but also guarantee tight obstacle avoidance. These algorithms begin with first obtaining a workspace representation explicitly incorporating the obstacles. A basic and simplest representation is to decompose the workspace into cells via a regular grid [141], but more efficient representations have been developed to reduce the memory cost for storing representations: Quadtrees [60] have been utilized for two dimensional spaces, and oct-tree [54] for three dimensional spaces. Roadmap type representations are utilized to extract a skeleton graph from the workspace, which keeps the connectivity between the nodes of feature points. Voronoi diagram [69] and Freeways [19] are widely used representations for global motion planning problem. The visibility graph [84, 101] is another method for obtaining a graph by connecting the edges (or, vertices) of obstacles from a given start and end points. Collin's decompositions [81] or probabilistic roadmaps [64] also become efficient representation methods for global path planning.

Having the workspace represented by either a partition of finite cells (region) or a complete graph, a path planning problem is then reduced to finding a sequence of neighboring cells or nodes in the graph satisfying certain extremal criteria. Several optimal searching algorithms have been proposed in the literature. The use of dynamic programming was proposed in Ref. [130] for a robot manipulator. An evolutionary search method was successfully used to compute near-optimal paths through obstacles and dynamically changing environment [115, 55]. The randomized rapid-exploring heuristic search in Ref. [63] forms a subset of stochastic optimization [91] for a global path planning, in which the search tree is expanded online towards randomly generated target states. If a graph is obtained for workspace representation, the $\mathcal{A}^*$ algorithm [49], or the Dijkstra's algorithm [35], are the most common algorithms to find the shortest path from a given initial node to a given goal node on the graph.

Many different versions of this search algorithm have been presented in the literature for path planning of mobile robots [7, 111, 112, 29, 68, 152].

A path planning algorithm over a dynamic environment has also been proposed in the literature. In order to deal with a dynamically changing environment, it is necessary for the path planning algorithm to replan as often as possible in accordance with the new information about the environment. Several replanning strategies have been proposed for locally-directed wandering [110], local modification of initial path [72], and obstacle perimeter detouring [86]. The $\mathcal{D}^*$ algorithm, proposed by Stentz [135, 136], is capable of planning a path in unknown or partially known environment, as it adopts an efficient incremental search scheme to produce an optimal path. The incremental search scheme incorporates the information from previous searches to find a solution much faster than solving each iteration from scratch. Subsequently, Koenig and Likhachev proposed the Lifelong Planning $\mathcal{A}^*$ (LPA) [70] which has been derived from $\mathcal{A}^*$ and adopted for dynamic environment by a reusing scheme. Furthermore, Koenig and Likhachev presented a $\mathcal{D}^*$-lite algorithm, derived from the LPA algorithm, which implements the same planning strategy as $\mathcal{D}^*$ but is algorithmically different.

1.2.3 Dynamic path planning

Most of the search methods discussed above do not incorporate dynamic models of the vehicles or robots. This, in turn, results in certain local path segments that the vehicles are hard to follow under kinematic constraints. Several authors have proposed to smooth these segments by taking the kinematic constraints into account [7, 29], but no explicit analysis on the optimality variation due to local path smoothing has been given.

Recently, mathematical optimization has been adopted by many researchers to accommodate dynamic vehicle models in the path planning problems. Nonlinear programming was used in Refs. [114, 46] and the use of mixed integer linear programming (MILP) in the path planning problem was introduced in Refs. [124, 118, 117]. MILP allows inclusion of integer variables and discrete logic in a continuous linear optimization problem. These variables can be used to model logical constraints such as obstacle and collision avoidance, while dynamic equations of the vehicles are formulated as dynamic constraints. Although the MILP algorithm in trajectory planning can systemically handle *hard* obstacles of rectangular shapes over an entire workspace and provide a feasible trajectory if one exists, it can not handle the computational complexity for a large size trajectory planning problem, and the algorithm is not very useful in dynamically changing environment.

The computational complexity arising from the path planning over an entire workspace can be reduced by repeatedly solving a constrained optimization problem online over a finite planning horizon. This scheme is so called model predictive control (MPC), or receding horizon control (RHC), originated from the chemical process industry and has received great attention due to its broader applications over various research fields. A good survey on MPC/RHC can be found in Refs. [45, 97] and the relevant texts [24, 4]. In recent years, receding horizon control has been introduced to solve the problems of trajectory generation and collision avoidance of multiple vehicles: Fuller et al [43] presented a MPC-based trajectory generation method for a flight vehicle with nonlinear dynamics by using a feedforward nominal trajectory to pose a time-varying linear, convex optimization problem. This result is extended in Refs. [131, 77] to account for obstacle avoidance trajectory planning. Shim et al [67, 129] proposed a nonlinear model predictive tracking control (NMPTC) for planning a path under input and state constraints as well as collision avoidance. More recent publications regarding the implementation of NMPTC for UAVs are found in Refs. [128, 62]. The benefit of the MPC approach is that it can allow one to effectively handle multi-variable systems with inputs and states constraints while less computational cost requires.

On the other hand, an alternative receding horizon approach based on MILP was introduced in Refs. [13, 119]. In this approach, a discrete point mass model was employed as a finer dynamics model for a two-dimensional vehicle under the velocity and force limits, which correspond to state and control constraints, respectively. A finer level of optimization was executed over a fine resolution of planning horizon using MILP solver, whereas a coarser level of optimization for the shortest path was done by searching a visibility graph that was expanded from a goal point through edges of each *hard* obstacle. These two-step optimizations achieve the shortest path over a long time-scale while planning a dynamically feasible path over a short time-scale at the reduced computational cost. In the continuation papers, the authors of Refs. [12, 73] proposed a stable trajectory for highly complex environment by putting turning circles at each obstacle's corner, which takes into account the kinematic constraints of the maximum turning rate of vehicles. In addition, a safety condition of the algorithm was imposed by the form of loiter circles as a backup plan in case of the algorithm failure [123]. Although this algorithm has been successfully tested and executed for UAV guidance [146, 125], the algorithm has certain underlying weaknesses. First, because the discrete MILP solver incorporates linearized equations, in conjunction with binary variables as the constraints of turning circles, velocity and force limits, and so on, a better accuracy of solution could be obtained from using more binary variables to accommodate detailed constraint equations. This results in increased computation time that may not be suitable for realtime implementation. Next, since the obstacles are limited to rectangular shapes to apply the visibility graph search, various shapes of obstacles other than rectangles will require a conservative modeling of obstacles by polygons that completely enclose such obstacles, which might end up with a conservative path.

1.2.4 Multiresolution path planning

The path planning methodology that will be developed in this thesis will involve a multiresolution workspace representation. One benefit of such representation is that it is able to handle complex shapes of obstacles other than rectangles [60]. In this hierarchical representation, the workspace is recursively divided into cells of either white (obstacle-free region) or black (obstacle region). Since the detail partitioning is done along the boundary of the obstacles, an arbitrary obstacle shape can be incorporated into the path planning problem. Finding the shortest path around the obstacles is obtained by searching over the hierarchical data structure containing the connectivity information between cells [103]. Extensions of this standard quadtree data structure were pursued in Refs. [30, 148] to overcome the limitation of generating a nonoptimal solution when an obstacle is located on or near the boundary of a quad. In addition, triangular cell decompositions [52] instead of a square cell of the quadtree are known to provide an alternative method to get around such limitation. Another benefit of employing multiresolution based path planning is that it can substantially reduce the complexity of computation by adopting variable resolution grids [11]. The resolution is set high close to a vehicle and becomes less at far away distance. Taking care of the connectivity between the non-uniform size neighboring cells, the path planner could find a high accuracy initial path and replan continuously as the vehicle moves at reduced computation costs.

Recently, a wavelet-based path planning algorithm has been introduced to efficiently solve the path planning problem on a complex environment. In Refs. [47, 121], a trajectory was first parameterized using wavelet decomposition and then optimized for satisfying mission requirements using dynamic optimization methods. Based on multiresolution active modeling, the planned trajectory was characterized as high fidelity over the near-term and as approximate over the longer-term. The wavelet transform can also provide a fast decomposition of the environment at different levels of resolution [33] and it can be used to construct a hierarchical representation of the workspace as used in Ref. [104]. In a recent paper, Tsiotras and Bakolas [142] proposed to utilize the wavelet transform in a hierarchical on-line path planning problem, in which the workspace representation has high resolution close to the current position of the vehicle and low resolution far away. A topological graph is obtained from the wavelet decomposition online and repeatedly searched for the shortest path using Dijkstra's algorithm. The path planning algorithm is scalable, and can be implemented in real-time for various computational resources of the agent.

1.3 Outline of Thesis

Having presented the motivation and the related work in the literature, we now present the main objectives of this thesis. The outline of the overall organization of the thesis is shown in Fig. 1: In Chapter 2, we present the multiresolution path planning algorithm. We employ the fast lifting wavelet transform scheme to approximate a world by a multiresolution cell decomposition. An efficient algorithm for building the adjacency relationship is presented. In Chapter 3, the on-line path smoothing algorithm is presented. The path templates are comprised of a set of B-spline curves, which are computed from the off-line optimization. Given a discrete optimal path sequence, an on-line path smoothing algorithm is proposed to generate a smooth reference path to be used for path following control. Design of a nonlinear path following control law is presented in Chapter 4. We present a kinematic path following controller, which calculates the heading rate command input to the unicycle kinematics. The actual roll angle command for a fixed-wing UAV is derived from the heading rate command using backstepping technique, and the parameter adaptation scheme is incorporated to deal with the uncertainty of the time constant. In Chapter 5, hardware-in-the-loop (HIL) simulation results of the proposed multiresolution,

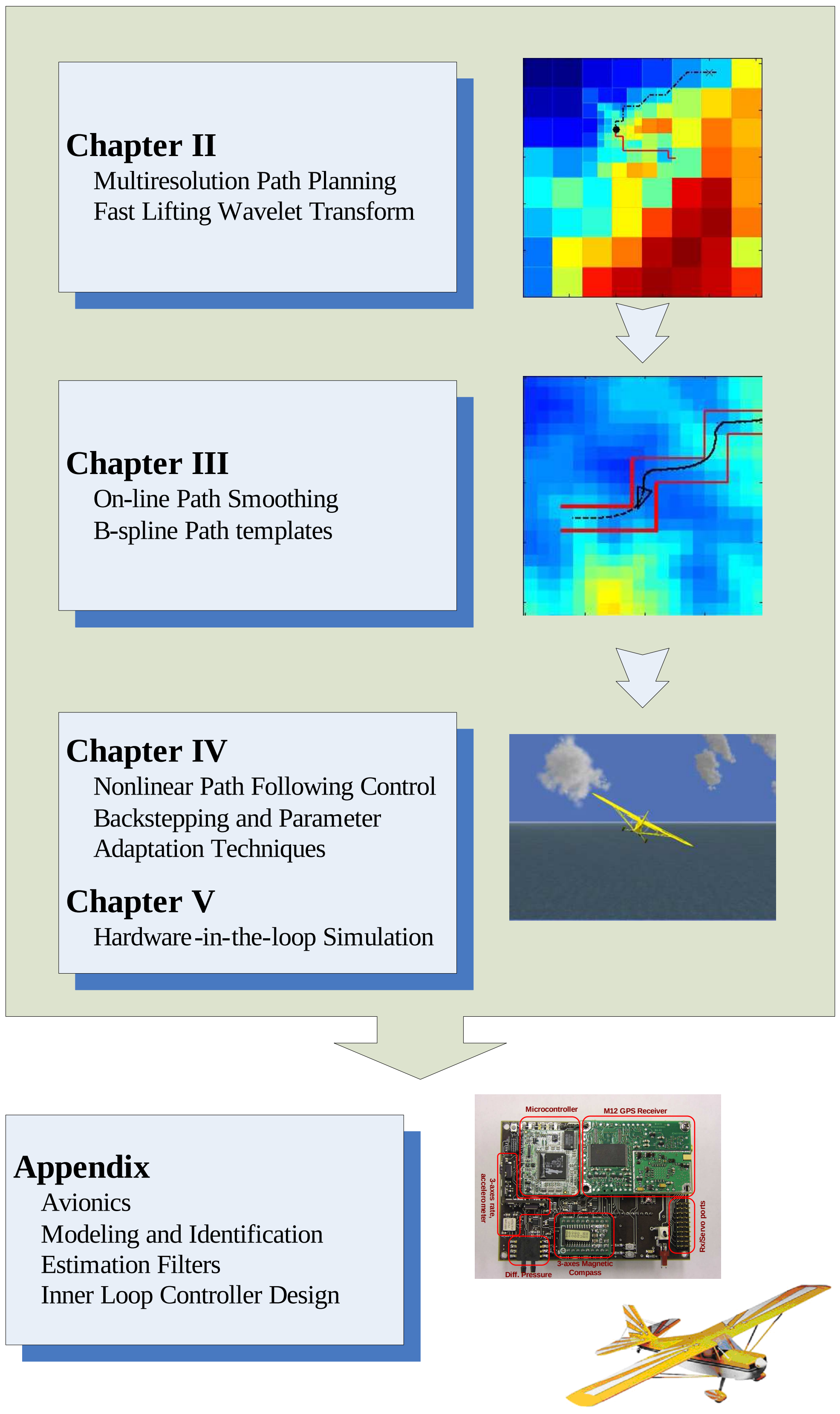


Figure 1: Thesis Outline.

hierarchical path control algorithm are shown. The proposed hierarchical path control algorithm is implemented on-board the actual hardware platform in real-time by a seamless integration of the hardware and the software. Finally, Chapter 6 summarizes the main results of this thesis, and enlists several recommendations for the future research.

1.4 Research Contributions

In this thesis we present both a theoretical development of the hierarchical path control algorithm and experimental results from the on-line implementation on the actual system. The following summary enlists the contribution of this work.

- Autonomous path planning for small UAVs imposes severe restrictions on control algorithm development, stemming from the limitations imposed by the on-board hardware and the requirement for on-line implementation. We propose a method to overcome this problem by using a new multiresolution path planning scheme. The fast lifting wavelet transform incorporating integer arithmetic is utilized to compute multiresolution representations of the environment, by taking into account the available computational resources of the agent. In addition, we show that the connectivity relationship of the graph $\mathcal{G}$ that is associated with the multiresolution representation can be constructed directly from the wavelet coefficients. Hence, the proposed multiresolution path planning algorithm is suitable for the on-line, on-board, and real-time implementation on a small UAV.

- For guidance and navigation control of UAVs, a complete solution which takes into account the equations of motion in conjunction with kinematic constraints is far from implementing in real-time, specially for small UAVs equipped with limited on-board hardware. We propose a path smoothing algorithm using a set of path templates. Instead of smoothing the entire path from an initial position to the goal position, we smooth the path segments over a finite planning horizon with respect to the current position of the UAV. This approach is somewhat similar to the receding horizon control adapted to the path generation purpose. By incorporating the path templates from off-line optimization, the proposed smoothing scheme has the advantage of minimal on-line computational cost since most of computation is done off-line.

- We present a nonlinear path following control algorithm, to regulate the error distances from a reference path. Based on the kinematic control law for unicycle-type mobile robot in Refs. [75, 76], a backstepping control law is proposed to compute the actual roll angle command for a fixed-wing UAV. In addition, in order to compensate for the inaccurate time constant of the roll angle closed-loop system, we propose to apply the parameter adaptation technique. The performance of the proposed path following control law is validated through the realistic simulation environment, showing that the applicability of the proposed algorithm on the actual system.

- The proposed hierarchical path control algorithm, which includes the path planning at the top level hierarchy, the path smoothing in the middle, and the path following at the bottom level, is experimentally validated through a realistic hardware-in-the-loop simulation environment. We describe the practical issues associated with the implementation of the proposed control algorithm, taking into consideration the actual hardware limitation. With a seamless integration of the hardware and the software, we demonstrate a real-time validation of the proposed hierarchical path control algorithm on-board the autopilot.

- The heart of the UAV platform is its autopilot, which consists of a flight control computer, sensors, actuators, communication devices and peripherals, along with the associated software. In order to provide maximum flexibility in control law development and implementation, we develop both the hardware and the low-level and high-level software in-house. We present the detail development of the autopilot hardware components and the subsystem integration process.

- A complete simulation environment for testing and validating UAV control systems is presented. With a comprehensive system modeling and experimental identification of a fixed-wing UAV, a high-fidelity simulation environment is built, which allows to incorporate a simulation-based testing during the development of the hardware and software. In addition, hardware-in-the-loop (HIL) simulation environment is developed for validating hardware and the software under realistic conditions. Details on developing user interface and subsystem modeling are also presented.

- A low cost inertial attitude and position reference system for a small UAV is presented. The estimation algorithms are simple, yet effective so that a microcontroller can execute these algorithms within a small time interval. We develop an algorithm that combines a complementary filter and a Kalman filter for the Euler attitude angles. A straightforward and innovative way of handling the data latency and the outage of a GPS sensor is introduced. Finally, a cascaded position estimation Kalman filter is designed utilizing the attitude estimates to lower the computational burden with a small performance loss.

CHAPTER II

MULTIRESOLUTION ON-LINE PATH PLANNING

2.1 Background

In a typical mission of a UAV, various sensors (e.g., cameras, radars, laser scanners, satellite imagery) having different range and resolution characteristics are employed to collect information about the environment the vehicle operates in. A computationally efficient path planning algorithm, specifically adopted for on-line implementation, should therefore choose the expedient information from all these sensors, and use the on-board computational resources to design the part of the path (spatial and temporal) that needs it most. In a nutshell, a computationally efficient algorithm suitable for *on-line* implementation should be characterized by a combination of short term tactics (reaction to unforeseen threats) with long-term strategy (planning towards the ultimate goal).

In this study, we assume a world environment $\mathcal{W} \subset \mathbb{R}^2$ that includes the obstacle space $\mathcal{O} \subset \mathcal{W}$ and the obstacle-free configuration space $\mathcal{F} = \mathcal{W} \setminus \mathcal{O}$ of all feasible states. We employ the wavelet transform to perform the required multiresolution decomposition of $\mathcal{W}$. The fast lifting wavelet transform (FLWT) offers a fast decomposition of a function at different levels of resolution, which can be twice as fast as the classical wavelet transform[1]. Furthermore, the FLWT integer arithmetic can be implemented to reduce the computational cost dramatically. This makes the FLWT especially suitable for processing data in a small micro-controller. The use of FLWT

[1]The computational complexity of the lifting scheme is still of order $\mathcal{O}(n)$ where n is the input data[145], however, the computational time may decrease by half according to wavelet basis.

has also the added benefit of allowing the construction of the associated cell connectivity relationship directly from the wavelet coefficients, thus eliminating the need for quadtree decomposition.

We employ the multiresolution path planning scheme to find an optimal path on the topological graph $\mathcal{G}$ induced by the multiresolution wavelet-based cell decomposition. Namely, the optimal path from an initial state to a final state may comprise of cells over different resolutions, depending on the distance from the current position of the agent. At the finer resolution level, the path is resolved through feasible states, hence avoiding obstacles. Multiresolution path planning is known to be more flexible, and computationally efficient for on-line implementation.

2.2 A Multiresolution Decomposition of $\mathcal{W}$

2.2.1 The 2D wavelet transform

The idea behind the wavelet transform is to represent a function $f \in \mathcal{L}^2(\mathbb{R})$ via a linear combination of elementary basis functions $\phi_{J,k}$ and $\psi_{j,k}$ as follows

$$f(x) = \sum_{k\in\mathbb{Z}} a_{J,k}\phi_{J,k}(x) + \sum_{j\geq J}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x), \tag{1}$$

where $\phi_{J,k}(x) = 2^{J/2}\phi(2^J x - k)$ and $\psi_{j,k} = 2^{j/2}\psi(2^j x - k)$. The choice of J determines the low resolution, or coarse approximation of f, spanned by the scaling function $\phi_{J,k}(x)$. The rest of $\mathcal{L}^2(\mathbb{R})$ is spanned by the wavelet functions $\psi_{j,k}(x)$ which provide the higher, or finer resolution details of the function. In other words, when analyzing the function f at the coarsest level (low resolution), only the most salient features of f will be revealed. Adding finer levels (high resolution) implies adding more and more details of the function f. The expansion (1) thus reveals the properties of f at different levels of resolution[22, 90]. In addition, in the ideal case both the scaling function and the wavelet function have compact support, i.e., they are non-zero only on a finite interval. This allows the wavelets to capture the localized features of the function f.

The wavelet transform can be readily extended to the two-dimensional case by introducing the following families of functions

$$\begin{aligned}
\Phi_{j,k,\ell}(x,y) &= \phi_{j,k}(x)\phi_{j,\ell}(y), && \text{(2a)}\\
\Psi^1_{j,k,\ell}(x,y) &= \phi_{j,k}(x)\psi_{j,\ell}(y), && \text{(2b)}\\
\Psi^2_{j,k,\ell}(x,y) &= \psi_{j,k}(x)\phi_{j,\ell}(y), && \text{(2c)}\\
\Psi^3_{j,k,\ell}(x,y) &= \psi_{j,k}(x)\psi_{j,\ell}(y). && \text{(2d)}
\end{aligned}$$

Given a function $f \in \mathcal{L}^2(\mathbb{R}^2)$ we can then write

$$f(x,y) = \sum_{k,\ell\in\mathbb{Z}} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j\geq J}\sum_{k,\ell\in\mathbb{Z}} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y) \tag{3}$$

where, for the case of orthonormal wavelets, the approximation coefficients are given by[2]

$$a_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Phi_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y, \tag{4}$$

and the detail coefficients by

$$d^i_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Psi^i_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{5}$$

The key property of wavelets used in this paper is the fact that the expansion (3) induces the following multiresolution decomposition of $\mathcal{L}^2(\mathbb{R}^2)$

$$\mathcal{L}^2(\mathbb{R}^2) = \mathcal{V}_J \oplus \mathcal{W}_J \oplus \mathcal{W}_{J+1} \oplus \cdots \tag{6}$$

where $\mathcal{V}_J = \overline{\operatorname{span}_{k,\ell\in\mathbb{Z}}\{\Phi_{J,k,\ell}\}}$ and $\mathcal{W}_j = \overline{\operatorname{span}_{k,\ell\in\mathbb{Z}}\{\Psi^1_{j,k,\ell}, \Psi^2_{j,k,\ell}, \Psi^3_{j,k,\ell}\}}$ for $j \geq J$.

In this study we use the Haar wavelet system for reasons that will become apparent shortly. The Haar scaling function

$$\phi(x) = \begin{cases} 1 & \text{if } x \in [0,1), \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

[2]In the more general case of biorthogonal wavelets, projections on the space spanned by the dual wavelets and dual scaling functions should be used in (4) and (5).

and the Haar wavelet function

$$\psi(x) = \begin{cases} 1 & \text{if } x \in [0, 1/2), \\ -1 & \text{if } x \in [1/2, 1), \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

have compact support on $[0, 1]$. Hence, each scaling function $\phi_{j,k}(x)$ and wavelet function $\psi_{j,k}(x)$ in the Haar system has support on the dyadic interval $I_{j,k} \overset{\triangle}{=} [k/2^j, (k+1)/2^j]$ of length $1/2^j$ and does not vanish in this interval [22, 149]. Similarly, we may associate the two-dimensional scaling function $\Phi_{j,k,\ell}$ and the wavelet function $\Psi^i_{j,k,\ell}$ $(i = 1, 2, 3)$ with the rectangular cell $c^j_{k,\ell} \overset{\triangle}{=} I_{j,k} \times I_{j,\ell}$.

2.2.2 Fast lifting wavelet transform (FLWT)

Implementing the wavelet transform in practice requires dealing with a discrete signal. The basic step in a typical discrete wavelet transform (DWT) involves the use of filter banks. Figure 2 shows a discrete signal a_n filtered by two complementary high- and low-pass (decomposition) filters $\bar{g}$ and $\bar{h}$ before it is down-sampled. The results of this operation are the next coarser approximation and detail coefficients a_{n-1} and d_{n-1}, each containing half as many samples as the input signal a_n. For the inverse transform, first the signals a_{n-1} and d_{n-1} are upsampled by inserting zeroes between every sample. Subsequently, the two signals are filtered by the low- and high-pass (reconstruction) filters $\tilde{g}$ and $\tilde{h}$, respectively, and then added together. This sequence of operations results in perfect reconstruction of the original signal a_n. Details of the filter bank implementation of wavelet transforms can be found, for instance, in Refs.[37] and [138].

The fast lifting wavelet scheme, originally introduced in Refs. [139] and [140], is a new method for building wavelets directly in the time domain, thus avoiding the use of Fourier analysis. Moreover, the scheme can be extended to construct the so-called second generation wavelets, which have certain benefits for handling boundary effects,

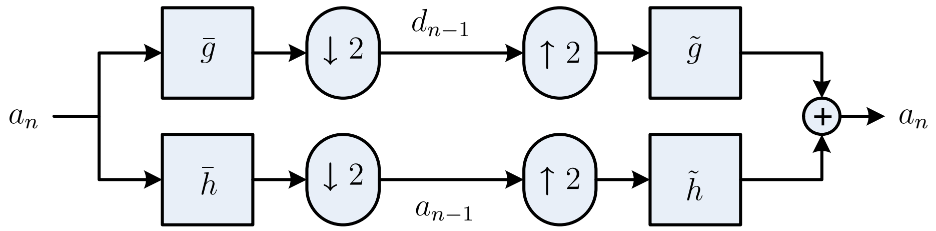


Figure 2: A typical one-stage two-band filter banks used for implementing the discrete wavelet transform.

irregular samples, and arbitrary weight functions[138].

A typical lifting decomposition scheme is depicted in Fig. 3. The first block in this decomposition splits the original signal a_n into two disjoint sets of samples containing the odd and the even indexed samples (Lazy wavelet). Because the even and odd subsets are correlated to each other locally, each signal is lifted by the opposite signal after passing through the corresponding operators $\mathcal{P}$ and $\mathcal{U}$ (the dual and primal lifting, or prediction and update, respectively). Finally, the results are normalized with the constants k_a and k_d, to end up with the approximation and detail coefficients, a_{n-1} and d_{n-1}, respectively.

For the case of the unnormalized Haar transform, the dual lifting does nothing more but calculate the difference

$$d_{n-1,k} = a_{n,2k+1} - a_{n,2k}, \tag{9}$$

whereas the primal lifting calculates the coarser approximation coefficients having the same average value as the original signal, by updating the even samples using the previously calculated detail signal as follows

$$a_{n-1,k} = a_{n,2k} + d_{n-1,k}/2. \tag{10}$$

It has been proved that all classical wavelet transforms can be implemented using the lifting scheme[33]. Most interestingly, the inverse transform is readily found by reversing the order of the operations and by flipping the signs.

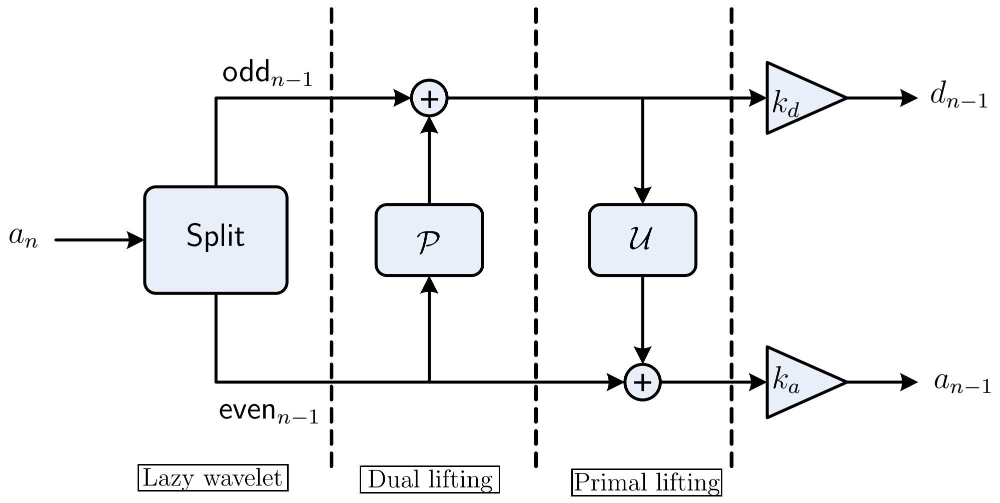


Figure 3: One step decomposition using the lifting scheme with the lazy wavelet.

The lifting scheme has a number of algorithmic advantages, such as faster computation speed (twice as fast as the usual discrete wavelet transform), *in-place* calculation of the coefficients (that saves memory), immediate inverse transform, generality for extension to irregular problems, etc. In particular, the lifting scheme is applicable to many applications where the input data consists of integer samples. Unlike the typical wavelet transform where floating number arithmetic is implicitly assumed, the lifting scheme can be easily modified to map integers to integers, and is readily reversible to allow perfect reconstruction[23]. This reconstruction is possible by adopting the sequential transform after modifying Eqs. (9) and (10) as follows[50]

$$\begin{aligned} d_{n-1,k} &= a_{n,2k+1} - a_{n,2k}, \\ a_{n-1,k} &= a_{n,2k} + \lfloor d_{n-1,k}/2 \rfloor, \end{aligned} \tag{11}$$

where, $\lfloor\cdot\rfloor$ is the rounding operator. In the sequel, we use the fast lifting Haar transform for two-dimensional signals of integer samples using a sequential 2D scheme; that is, we perform two successive one-dimensional transforms through the rows and then columns of the input data.

2.2.3 Wavelet decomposition of the risk measure

Without loss of generality, we let $\mathcal{W} = [0,1]\times[0,1]$, which is described using a discrete (fine) grid of $2^N \times 2^N$ dyadic points. The finest level of resolution $J_{\max}$ is therefore bounded by N. It follows from Eq. (3) and the accompanying discussions that the Haar wavelet decomposition at resolution level $J \geq J_{\min}$ is given by

$$f(x,y) = \sum_{k,\ell=0}^{2^J-1} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j=J}^{N-1}\sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y), \tag{12}$$

and it induces a cell decomposition of $\mathcal{W}$ of square cells of maximum size $1/2^J \times 1/2^J$.

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto \mathcal{M}$ that represents the "risk measure" at the location $\mathsf{x} = (x,y)$, where $\mathcal{M}$ is a collection of m integer distinct risk measure levels defined by

$$\mathcal{M} \triangleq \{M_i : M_1 < M_2 < \cdots < M_m\}. \tag{13}$$

The obstacle space $\mathcal{O}$ is defined as the space where the risk measure values exceed a certain threshold $\overline{M}$, that is,

$$\mathcal{O} = \{\mathsf{x} \in \mathcal{W} \mid \mathsf{rm}(\mathsf{x}) > \overline{M},\ \overline{M} \in \mathcal{M}\}. \tag{14}$$

For $\mathsf{x} \in \mathcal{F}$, we may think of $\mathsf{rm}(\mathsf{x})$ as an indication of the proximity of the agent to the obstacle space, or the probability of $\mathsf{x} \in \mathcal{O}$.

We construct approximation of $\mathcal{W}$ at distinct levels of resolution $J_{\min} \leq j \leq J_{\max}$ at ranges r_j from the current location of the agent $\mathsf{x}_0 = (x_0, y_0)$, in the sense that the resolution j is used for all points inside the neighborhood

$$\mathcal{N}(\mathsf{x}_0, r_j) \triangleq \{\mathsf{x} \in \mathcal{W} : \|\mathsf{x} - \mathsf{x}_0\|_\infty \leq r_j\}. \tag{15}$$

where $r_{J_{\max}} \leq r_j \leq r_{J_{\min}}$. By this, we imply that the finer resolution $J_{\max}$ is used for points close to the current location, and coarser resolutions at different levels are used elsewhere, according to the distance from the current location. Hence, the representation of $\mathcal{W}$ gets coarser farther away from the current location. Figure 4 illustrates this situation. The choice of $J_{\max}$ is determined by the requirement that at this level all cells can be resolved into either free or obstacle cells. The choice of $J_{\min}$ as well as the window span r_j are dictated by the on-board computational resources.

Let now $\mathcal{I}(j) \triangleq \{0, 1, 3, \cdots, 2^j - 1\}$ and let

$$\mathcal{K}(j) \triangleq \{k \in \mathcal{I}(j) \mid I_{j,k} \cap [x_0 - r_j, x_0 + r_j] \neq \varnothing\}, \tag{16a}$$

$$\mathcal{L}(j) \triangleq \{\ell \in \mathcal{I}(j) \mid I_{j,\ell} \cap [y_0 - r_j, y_0 + r_j] \neq \varnothing\}. \tag{16b}$$

Then the wavelet decomposition of rm, given by

$$\mathsf{rm}(x,y) = \sum_{k,\ell \in \mathcal{I}(J_{\min})} a_{J_{\min},k,\ell} \Phi_{J_{\min},k,\ell}(x,y) + \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\substack{k \in \mathcal{K}(j) \\ \ell \in \mathcal{L}(j)}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x,y), \tag{17}$$

induces, with a slight abuse of notation, the following multiresolution cell decomposition on $\mathcal{W}$

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}}, \tag{18}$$

where, ΔC_d^j is a union of cells $c^j_{k,\ell}$ of dimension $1/2^j \times 1/2^j$.

2.3 Multiresolution Graph Connectivity

2.3.1 Computation of adjacency list from the FLWT

In the previous section we described the construction of a multiresolution cell decomposition $\mathcal{C}_d$ of $\mathcal{W}$ in Eq. (18) using the FLWT. We now assign a topological graph $\mathcal{G} = (V, E)$ to $\mathcal{C}_d$ as follows. The nodes which belong to the set V represent the cells $c^j_{k,\ell}$ in $\mathcal{C}_d$ and the edges in the set E represent the connectivity relationship between these nodes. In this section we show that the connectivity of the graph $\mathcal{G}$ can be constructed directly from the wavelet coefficients. Equivalently, we compute the adjacency list of $\mathcal{G}$ directly from wavelet coefficients obtained from the FLWT.

Since the scaling function $\Phi_{j,k,\ell}$ and the wavelet functions $\Psi^i_{j,k,\ell}$ $(i = 1, 2, 3)$ of the 2D Haar wavelet are associated with square cells $c^j_{k,\ell}$, the corresponding approximation and nonzero detail coefficients encode the necessary information regarding the

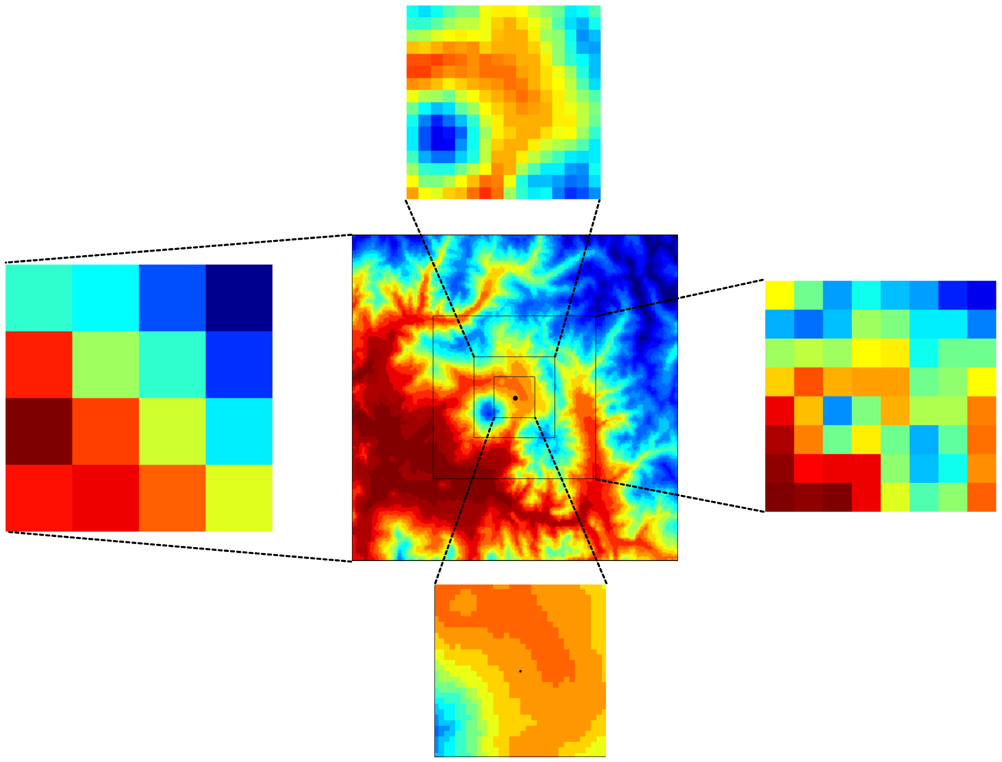


Figure 4: Multiresolution representation of the environment according to the distance from the current location of the agent.

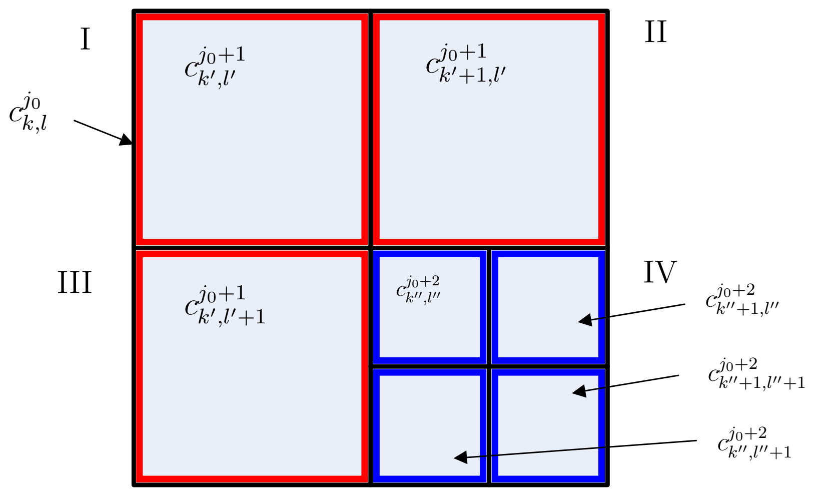


Figure 5: Multiresolution cell subdivision across different levels.

cell geometry (size and location). Recall that the approximation coefficients are the average values of the risk measure over the cells, and the detail coefficients determine the size of each cell. More specifically, consider a cell $c_{k,\ell}^{j_0}$ at level j_0, whose dimension is $1/2^{j_0} \times 1/2^{j_0}$ and is located at (k, ℓ). A cell will be called *independent* if it is associated with a non-zero approximation coefficient $a_{j_0,k,\ell}$, while the corresponding detail coefficients $d_{j,k,\ell}^i$ $(i = 1, 2, 3)$ at level $j_0 \le j \le J_{\max}$ are all zero. Otherwise, the cell is marked as a *parent* cell, and is subdivided into four *leaf* cells at level $j_0 + 1$. If a leaf cell cannot be subdivided further, it is classified as an independent cell. In Fig. 5, the top-most parent cell $c_{k,\ell}^{j_0}$ is subdivided into three independent cells at level $j_0 + 1$ with each non-zero approximation coefficient in the quadrant I, II, and III (all zero detail coefficients). For quadrant IV, the cell is further subdivided into four independent leaf cells at level $j_0 + 2$.

Assume that we are given the Haar wavelet transform of the risk measure function rm up to the level $J_{\min}$. The coarsest level of the cell dimension is set to $J_{\min}$. In Fig. 6, the initial coarse grid is drawn on the left. The agent is located at $\mathsf{x} = (x, y)$ and the high resolution horizon is given by r. Recalling expressions (16), we distinguish cells at distinct resolution levels, by starting from a coarse cell $c_{k,\ell}^{j_0}$, and by determining

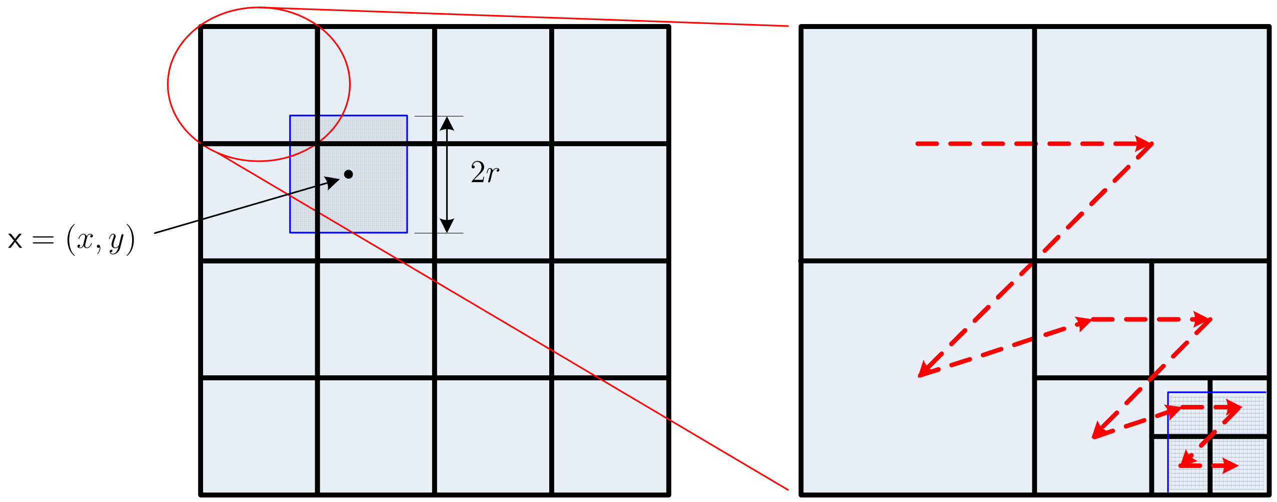


Figure 6: Recursive raster scan method for identifying independent cells.

if the cell either partially intersects or totally belongs to the set $\mathcal{N}(\mathsf{x}, r)$. The cell $c^{j_0}_{k,\ell}$ is easily ascertained to satisfy this property by choosing the indices such that $(k, \ell) \in (\mathcal{K}(j_0), \mathcal{L}(j_0))$. If the cell needs to be subdivided into higher resolution cells, the inverse fast lifting wavelet transform is first performed on the current cell (local reconstruction) in order to recover the four approximation coefficients at level $j_0 + 1$ and the corresponding detail coefficients. We then adopt the raster scan method[95] (zigzag search: I→II→III→IV) to examine each cell inside the parent cell overlapping with $\mathcal{N}(\mathsf{x}, r)$. This procedure is recursively repeated until the maximum resolution level $J_{\max}$ is reached. Figure 6 illustrates the recursive raster scan search. Once a cell is recognized as independent, we assign a node in the graph $\mathcal{G}$ with the node cost being the approximation coefficient that represents the average risk measure over the cell. In addition, the detail coefficients associated with the current cell are all set to zero; this will provide the necessary connectivity information between the cells later on.

After a cell has been identified as an independent cell, we search the adjacent cells in order to establish the adjacency relationship with the current cell. Recall that two cells c_i and c_j are adjacent if

$$\partial c_i \cap \partial c_j \neq \varnothing, \quad i \neq j,$$

where ∂c_i denotes the boundary of the cell c_i. For our case of square cells, this implies that two cells are adjacent only along the following eight directions: Left, top, right, bottom, and the four diagonal directions. Following the recursive raster search for cell identification, the adjacency search requires establishing links between two cells that have been identified as independent cells. Recalling that the raster search progresses from left to right and from top to down (zigzag progress) as illustrated in Fig. 6, we confine the adjacency search to the following directions: Left, top-left, top, and top-right from the current cell. By doing this, we render half of the links (for eight-connectivity) to be connected from the current cell, and the remaining links with the current cell will be connected as the recursive raster scan progresses to the next cells. In addition, because we deal with cells of different dimensions, it is required to devise a generic method to find the adjacency relationship between the cells.

Figure 7 illustrates the basic search direction of each leaf cell inside a parent cell. The dashed arrow points towards an external search region, that is, an adjacent cell could be found beyond the parent cell, whereas the solid arrow points towards an internal search region that belongs to the parent cell. In each search, we implicitly assume that the level of adjacent cells may vary from that of the parent cell to $J_{\max}$ (external connection), or from that of the current cell to $J_{\max}$ (internal connection).

A leaf cell inherits the search region from its parent cells, whose search direction ends up with one of the solid arrows in Fig. 7. Figure 8 shows this inheritance property. In Fig. 8 the current cell is chosen to be $c_{\mathrm{I}}^{j_0+2}$. This cell is a leaf cell of the parent cell $c_{\mathrm{IV}}^{j_0+1}$, which further becomes a leaf cell of the top-most parent cell $c_{k,l}^{j_0}$. The cell $c_{\mathrm{IV}}^{j_0+1}$ is located on the fourth quadrant inside the top-most parent cell $c_{k,l}^{j_0}$ so that the search region for $c_{\mathrm{IV}}^{j_0+1}$ ends up with the internal searches at the level j_0+1, whose adjacency search property is inherited to the cell $c_{\mathrm{I}}^{j_0+2}$ for left, top-left, and

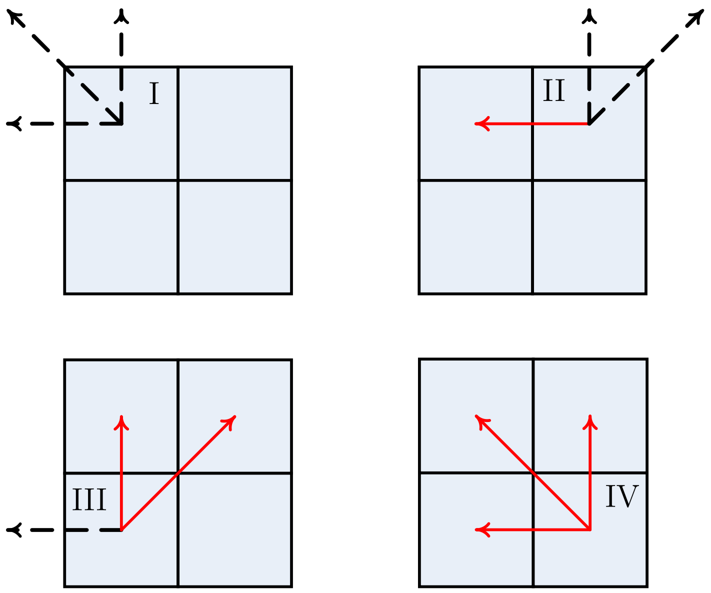


Figure 7: Basic connectivity properties with respect to the location of the leaf cell.

top direction searches. Having ascertained the basic search directions, we refine the adjacent search looking for opposite cells which must be independent and adjacent to the current cell. Because the opposite cells of the current cell could have different dimensions, we establish links by examining the associated detail coefficients of the opposite cells. Along the left search direction of $c_{\mathrm{I}}^{j_0+2}$, as illustrated in Fig. 8, one finds that only one independent cell at level $j_0 + 1$ is linked to $c_{\mathrm{I}}^{j_0+2}$.

The adjacency search algorithm refines its search to the higher levels if the opposite cell is not an independent cell, that is, if it is comprised of finer cells. This refinement subsequently forces a search of cells of the finer dimension (level) which are neighboring to the current cell. Subsequently, the detail coefficients of the opposite cells are examined in order to find the next finer cell that is adjacent to the current cell. For the top-left search direction of $c_{\mathrm{I}}^{j_0+2}$, as illustrated in Fig. 9(a), the search process initially examines the cell $c_{\mathrm{I}}^{j_0+1}$ located at the top-left corner of the current cell through the corresponding detail coefficient. Provided that the detail coefficient associated with the cell $c_{\mathrm{I}}^{j_0+1}$ takes a non-zero value, the cell is not an independent cell. Subsequently, the cell $c_{\mathrm{I}}^{j_0+1}$ is subdivided and the search process repeats at level

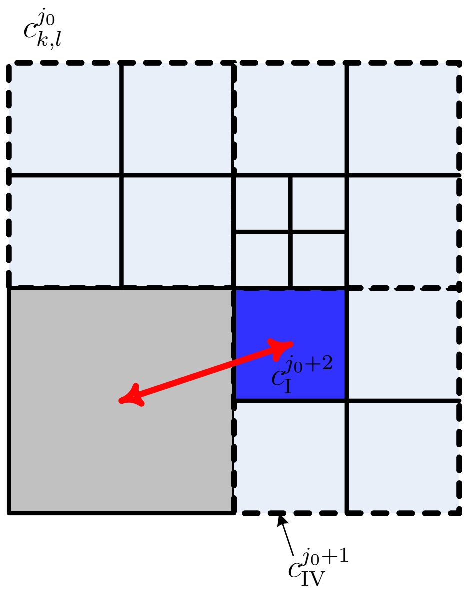


Figure 8: Searching an adjacent cell along the left search direction.

j_0+2 when the opposite cell to the current cell becomes an independent adjacent cell. In Fig. 9(a), since there exists no other independent cells along the top-left direction except the shaded one, a bidirectional link is established between the current and the opposite cells.

Similarly, for the top search direction, two cells at level $j_0 + 3$ and one at level j_0+2 are found to be independent and adjacent to the current cell. The bidirectional links are accordingly connected from the current cell $c_{\mathrm{I}}^{j_0+2}$ to those adjacent cells. Figure 9(b) depicts this situation. Finally, Fig. 10 shows an example of the graph structure obtained from the multiresolution cell decomposition associated with the wavelet coefficients. Without loss of generality the nodes are located at the center of each cell. The solid lines show the connectivity relationship between the cells.

A pseudo code implementation of the adjacency search algorithm is given in Fig. 11. Note that the subroutine Reconstruct_Scan is called recursively to identify independent cells intersecting with the set $\mathcal{N}(\mathsf{x}_0, r_j)$. After a cell identified as an independent cell, the adjacency search algorithm continues on establishing the links from the current cell to adjacent independent cells along Left, Left-Top, Top, and Right-Top

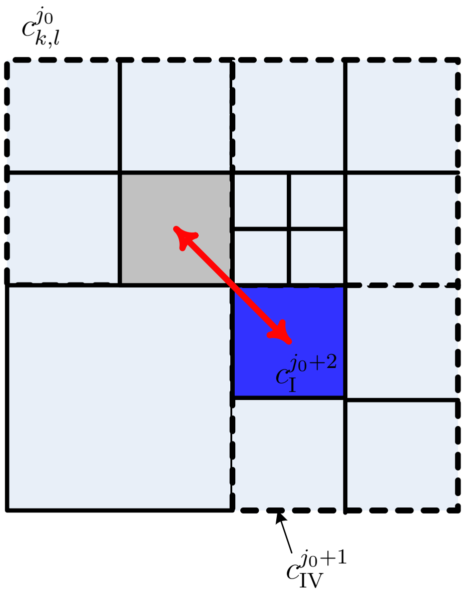


Figure 9: Refined adjacency search algorithm.

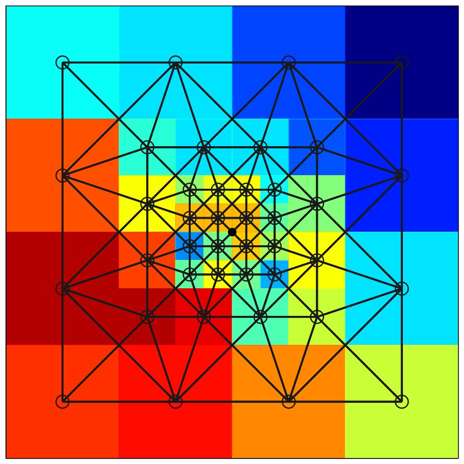


Figure 10: Connectivity relationship constructed from the multiresolution cell decomposition over three levels.

direction using the recursive refinements of search level as shown in Figs. 12 and 13.

Because the adjacency search algorithm uses recursive calls in both identification of cells and establishing links, obtaining the computational complexity for analytical expression is a non-trivial task. Rather, we estimate the computational complexity numerically. Suppose the input data to the algorithm are given by an $n \times n$ square image. The computational complexity of fast lifting wavelet transform is known to be $\mathcal{O}(N)$[145], where $N = n^2$. It should be noted that the computational cost can be tailored to the available computational resources of the agent due to multiresolution synthesis, however, it is rational to relate the computational complexity with the input data size N. For this purpose, we assumed two resolution levels, coarser level J_{rmmin} far away from the agent and finer level $J_{\max}$ close to the agent. Figure 14(a) shows the computational time in terms of various data size N in conjunction with a set of different sizes of high resolution window r. From Fig. 14 the numerical computational complexity of the adjacency search algorithm is deduced as a linear relationship with respect to the input data size N, or $\mathcal{O}(N)$. In contrast, Fig. 15 shows the computational complexity of the algorithm with respect to the window size r, which turns out to be a quadratic relationship $\mathcal{O}(r^2)$.

2.3.2 Cost assignment for $\mathcal{A}^*$ search

The $\mathcal{A}^*$ algorithm is a graph search algorithm that finds a path from an initial node to the goal node in the graph. The algorithm utilizes a *heuristic estimate* $h(v)$ that ranks each node v by a best cost estimate to reach the goal from the current node[49]. The algorithm visits the nodes in the order of the heuristic estimate, so the $\mathcal{A}^*$ algorithm is known as a best-first search algorithm. The key element of the $\mathcal{A}^*$ algorithm is that it expands each node from the priority queue that is ordered by (lower value has higher priority)

$$f(v) = g(v) + h(v), \tag{19}$$

```
BEGIN ADJACENCY SEARCH ALGORITHM
{
    Initialize graph structure $\mathcal{G} = (V, E)$;
    $\mathcal{C}_d \leftarrow$ Compute 2D FLWT($\mathcal{W}, J_{\min}$);
    $\big(\mathcal{K}(J_{\min}), \mathcal{L}(J_{\min})\big) \leftarrow \mathcal{N}(\mathsf{x}_0, r_{J_{\min}})$;
    for $(k = 0 : k_{J_{\min}})$ {
        for $(\ell = 0 : \ell_{J_{\min}})$ {
            if $(k, \ell) \in \big(\mathcal{K}(J_{\min}), \mathcal{L}(J_{\min})\big)$
                Reconstruct_Scan$(k, \ell, J_{\min}, \mathcal{C}_d)$;
            else
                AddNode_EstablishLinks$(k, \ell, J_{\min}, \mathcal{C}_d)$;
        }
    }
}
END ADJACENCY SEARCH ALGORITHM

Reconstruct_Scan($k$, $\ell$, $j$, $\mathcal{C}_d$)
{
    $\mathcal{C}_d \leftarrow$ Compute 2D Inverse FLWT$\big(\mathcal{C}_d(k, \ell; j)\big)$;
    $\big(\mathcal{K}(j+1), \mathcal{L}(j+1)\big) \leftarrow \mathcal{N}(\mathsf{x}_0, r_{j+1})$;
    for $(\bar{k} = 2k : 2k+1)$ {
        for $(\bar{\ell} = 2\ell : 2\ell+1)$ {
            if $(\bar{k}, \bar{\ell}) \in \big(\mathcal{K}(j+1), \mathcal{L}(j+1)\big)$
                Reconstruct_Scan$(\bar{k}, \bar{\ell}, j+1, \mathcal{C}_d)$;
            else
                AddNode_EstablishLinks$(\bar{k}, \bar{\ell}, j+1, \mathcal{C}_d)$;
        }
    }
}

AddNode_EstablishLinks($k$, $\ell$, $j$, $\mathcal{C}_d$)
{
    /* Add Node */
    $v \leftarrow c^j_{k,\ell} \sim a_{j,k,\ell}$;
    $V(\mathcal{G}) \leftarrow v$;
    Set $d^i_{j',k,\ell} \leftarrow 0 \ , \quad \forall j' \geq j$;

    /* Establish Links */
    LinkLeft$(k, \ell, j, \mathcal{C}_d)$;
    LinkLeftTop$(k, \ell, j, \mathcal{C}_d)$;
    LinkTop$(k, \ell, j, \mathcal{C}_d)$;
    LinkTopRight$(k, \ell, j, \mathcal{C}_d)$;
}
```

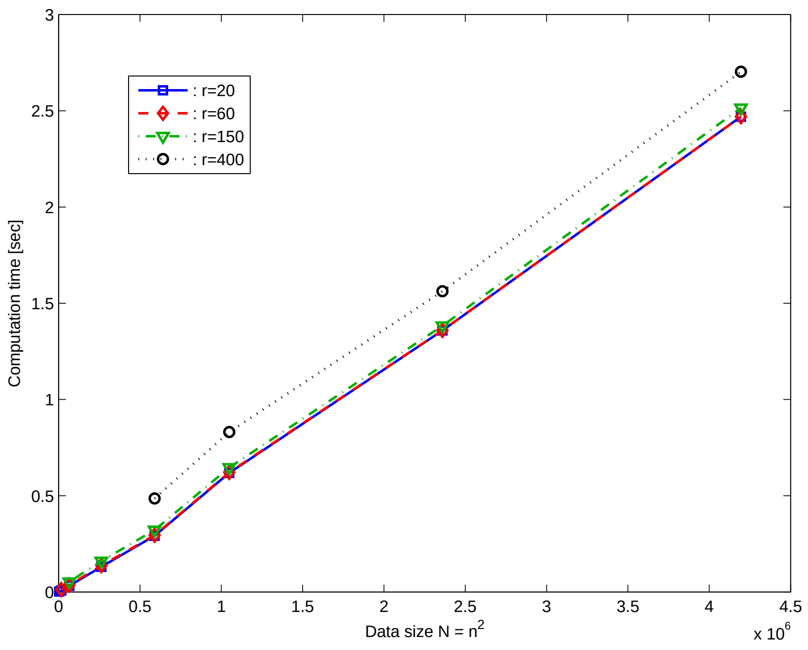
Figure 11: Pseudo-code implementation of the adjacency search algorithm.

```
LinkLeft($k$, $\ell$, $j$, $\mathcal{C}_d$)
{
    $c^j_{k,\ell} \leftarrow$ Get current cell($k, \ell, j, \mathcal{C}_d$);
    $j_s \leftarrow$ Determine basic search level($c^j_{k,\ell}$);
    $(k_s, \ell_s) \leftarrow$ Get parent cell index($c^j_{k,\ell}, j_s$);
    $k_s = k_s - 1$;    /* To examine left adjacent cell */
    LinkLeftRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}
LinkLeftRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$)
{
    $v \leftarrow c^j_{k,\ell}$;
    if ($d^1_{j_s,k_s,\ell_s} = 0$)    /* Horizontal detail */
        /* Adjacent independent cell */
        $u \leftarrow c^{j_s}_{k_s,\ell_s} \sim a_{j_s,k_s,\ell_s}$;
        $E(\mathcal{G}) \leftarrow (u,v),\ (v,u)$;  /* Bidirectional links */
    else
        /* Refine Search */
        $j_s \leftarrow j_s + 1$;
        $(k_s, \ell_s) \leftarrow$ Get refined left adjacent cell index($c^j_{k,\ell}, j_s$);
        LinkLeftRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}

LinkLeftTop($k$, $\ell$, $j$, $\mathcal{C}_d$)
{
    $c^j_{k,\ell} \leftarrow$ Get current cell($k, \ell, j, \mathcal{C}_d$);
    $j_s \leftarrow$ Determine basic search level($c^j_{k,\ell}$);
    $(k_s, \ell_s) \leftarrow$ Get parent cell index($c^j_{k,\ell}, j_s$);
    $k_s = k_s - 1$; $\ell_s = \ell_s - 1$;    /* To examine left-top adjacent cell */
    LinkLeftTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}
LinkLeftTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$)
{
    $v \leftarrow c_{j,k,\ell}$;
    if ($d^3_{j_s,k_s,\ell_s} = 0$)    /* Diagonal detail */
        /* Adjacent independent cell */
        $u \leftarrow c^{j_s}_{k_s,\ell_s} \sim a_{j_s,k_s,\ell_s}$;
        $E(\mathcal{G}) \leftarrow (u,v),\ (v,u)$;  /* Bidirectional links */
    else
        /* Refine Search */
        $j_s \leftarrow j_s + 1$;
        $(k_s, \ell_s) \leftarrow$ Get refined left-top adjacent cell index($c^j_{k,\ell}, j_s$);
        LinkLeftTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}
```

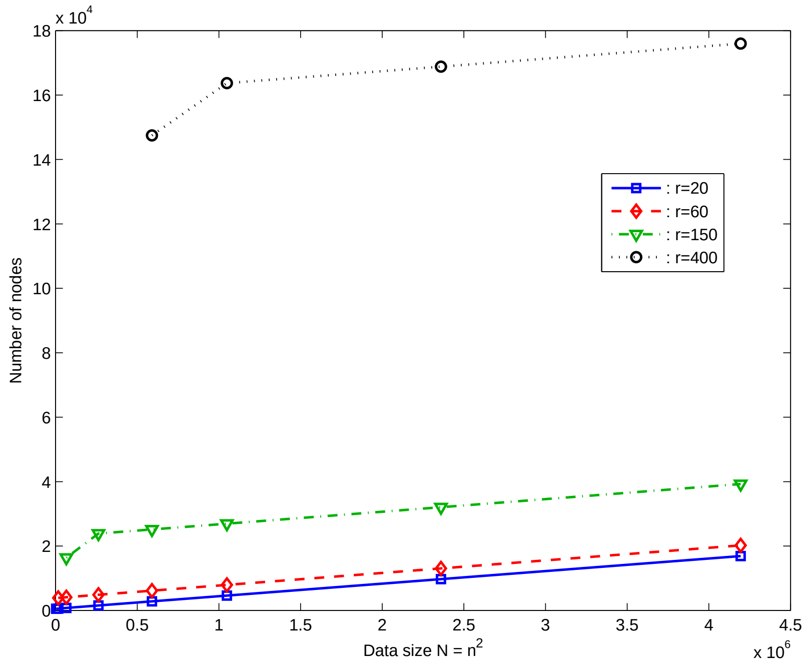
Figure 12: Pseudo-code implementation of the adjacency search algorithm: Recursive link connection.

```
LinkTop($k$, $\ell$, $j$, $\mathcal{C}_d$)
{
    $c^j_{k,\ell} \leftarrow$ Get current cell($k, \ell, j, \mathcal{C}_d$);
    $j_s \leftarrow$ Determine basic search level($c^j_{k,\ell}$);
    $(k_s, \ell_s) \leftarrow$ Get parent cell index($c^j_{k,\ell}, j_s$);
    $\ell_s = \ell_s - 1$;    /* To examine top adjacent cell */
    LinkTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}
LinkTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$)
{
    $v \leftarrow c^j_{k,\ell}$;
    if ($d^2_{j_s,k_s,\ell_s} = 0$)    /* Vertical detail */
        /* Adjacent independent cell */
        $u \leftarrow c^{j_s}_{k_s,\ell_s} \sim a_{j_s,k_s,\ell_s}$;
        $E(\mathcal{G}) \leftarrow (u, v),\ (v, u)$;  /* Bidirectional links */
    else
        /* Refine Search */
        $j_s \leftarrow j_s + 1$;
        $(k_s, \ell_s) \leftarrow$ Get refined top adjacent cell index($c^j_{k,\ell}, j_s$);
        LinkTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}

LinkRightTop($k$, $\ell$, $j$, $\mathcal{C}_d$)
{
    $c^j_{k,\ell} \leftarrow$ Get current cell($k, \ell, j, \mathcal{C}_d$);
    $j_s \leftarrow$ Determine basic search level($c^j_{k,\ell}$);
    $(k_s, \ell_s) \leftarrow$ Get parent cell index($c^j_{k,\ell}, j_s$);
    $k_s = k_s + 1$; $\ell_s = \ell_s - 1$;     /* To examine right-top adjacent cell */
    LinkRightTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}
LinkRightTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$)
{
    $v \leftarrow c_{j,k,\ell}$;
    if ($d^3_{j_s,k_s,\ell_s} = 0$)    /* Diagonal detail */
        /* Adjacent independent cell */
        $u \leftarrow c^{j_s}_{k_s,\ell_s} \sim a_{j_s,k_s,\ell_s}$;
        $E(\mathcal{G}) \leftarrow (u, v),\ (v, u)$;  /* Bidirectional links */
    else
        /* Refine Search */
        $j_s \leftarrow j_s + 1$;
        $(k_s, \ell_s) \leftarrow$ Get refined right-top adjacent cell index($c^j_{k,\ell}, j_s$);
        LinkRightTopRecurrence($c^j_{k,\ell}, k_s, \ell_s, j_s$);
}
```

Figure 13: Pseudo-code implementation of the adjacency search algorithm: Recursive link connection (continued).

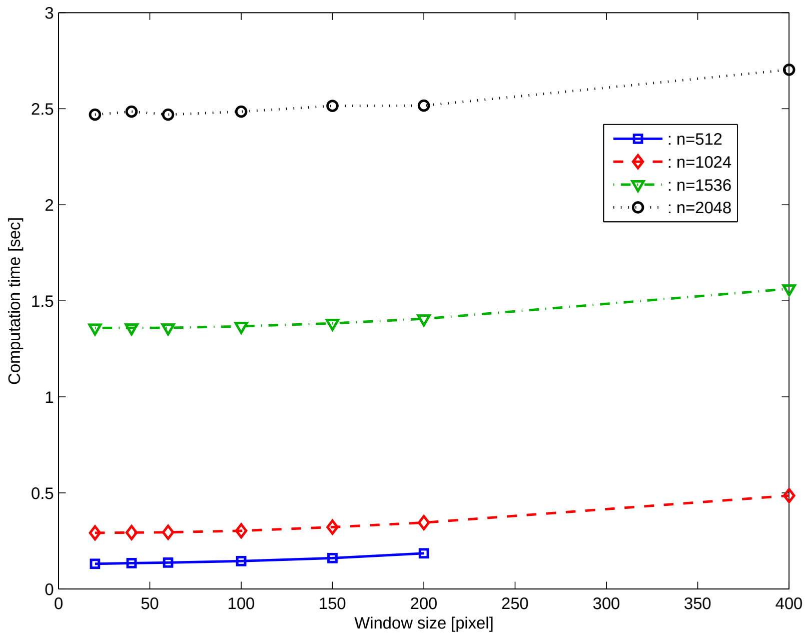


(a) Computational time

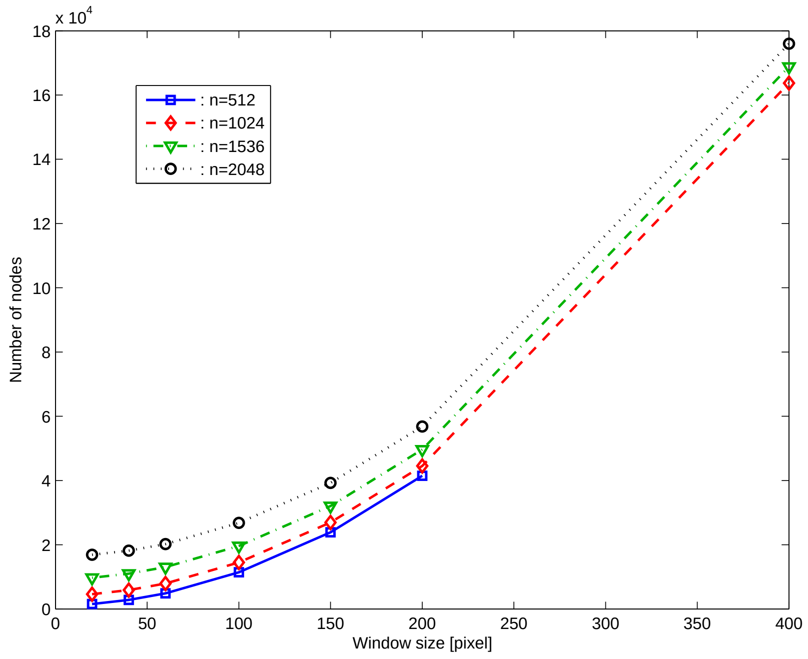


(b) Number of nodes

Figure 14: Computational cost for the adjacency search algorithm in terms of data size.



(a) Computational time



(b) Number of nodes

Figure 15: Computational cost for the adjacency search algorithm in terms of window size.

where the cost $g(v)$ is the actual cost of the path up to v, i.e. the sum of the edge costs from the initial node, and $h(v)$ is the heuristic estimate at v. When a node u is expanded, the adjacent nodes to the current node are exploited. Let v be the adjacent node, then it follows that we evaluate the actual cost $g(v)$ to see if the transition from u to v results in lower cost than any other transitions to v. The algorithm then sets a back-pointer $\pi(v)$ by its preceding node u. This process iterates until the goal node is reached and no other nodes have a lower cost to the goal.

The $\mathcal{A}^*$ algorithm is complete in the sense that it is always guaranteed to find a solution if a solution exists. In addition, if the heuristic function $h(v)$ is admissible, that is, it uses an underestimate of the actual cost of reaching the goal, then $\mathcal{A}^*$ is optimal. Details about the implementation of the $\mathcal{A}^*$ algorithm can be found, for instance, in Ref. [32].

To the cell decomposition (18) we associate each node $v \in \mathcal{G}$ o a cell $c^j_{k,\ell}$. Moreover, since $\mathcal{G}$ is a topological graph, we may associate each node v with some point $\mathsf{x} \in c^j_{k,\ell}$. Without loss of generality, we choose the center of the cell. Let $\mathsf{cell}_{\mathcal{G}}(v)$ denote the center of the corresponding cell. If $\mathsf{x} \in c^j_{k,\ell}$ we will write $v = \mathsf{node}_{\mathcal{G}}(\mathsf{x})$.

To each directed edge (u, v) of $\mathcal{G}$ we assign an edge cost, given as

$$\mathcal{J}(u, v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \alpha \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, \tag{20}$$

where $\alpha \geq 0$ is a weight constant. The first term in (20) is proportional to the probability that the target node is close to obstacles, while the second term penalizes the (Euclidean) distance between $\mathsf{cell}_{\mathcal{G}}(u)$ and $\mathsf{cell}_{\mathcal{G}}(v)$.

Suppose now that we are given a path of $q + 1$ consecutive, adjacent nodes in $\mathcal{G}$ as follows

$$\mathcal{P} = (v_0, v_1, \cdots, v_q). \tag{21}$$

We can then assign a traversal cost to each node in the path $\mathcal{P}$, induced by

$$g(v_i) = g(v_{i-1}) + \mathcal{J}(v_{i-1}, v_i), \qquad i = 1, \cdots, q. \tag{22}$$

The value of $g(v_k)$ represents the (accumulated) cost of the path from v_0 to v_k $(k \leq q)$, i.e. the weight of the edges followed up to v_k. We use the following heuristic estimate

$$h(v) = \|\mathsf{cell}_{\mathcal{G}}(v) - \mathsf{cell}_{\mathcal{G}}(v_f)\|_\infty, \tag{23}$$

where $v_f = \mathsf{node}_{\mathcal{G}}(\mathsf{x}_f)$.

The $\mathcal{A}^*$ algorithm then finds a path that minimizes the cost in (22) to the goal node, or determines that such a path does not exist.

2.4 *Multiresolution Path Planning*

2.4.1 Multiresolution path planning algorithm

The proposed multiresolution path planning algorithm proceeds as follows. Starting from $\mathsf{x}(t_0) = \mathsf{x}_0$ at time $t = t_0$, we construct using the approach of Section 2.2 a cell decomposition $\mathcal{C}_d(t_0)$ of $\mathcal{W}$. A topological graph, and the adjacency list of its nodes are obtained using the approach of Section 2.3. Let the corresponding graph be $\mathcal{G}(t_0)$ and let $v_1^0 \in \mathcal{G}(t_0)$ and $v_f^0 \in \mathcal{G}(t_0)$ be the initial and the goal nodes such that $v_1^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_0)$ and $v_f^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_f)$, respectively. Using the $\mathcal{A}^*$ algorithm we compute a path $\mathcal{P}(t_0)$ of free and mixed nodes from v_1^0 to v_f^0 in $\mathcal{G}(t_0)$ assuming that such a path exists. Let $\mathcal{P}(t_0)$ be given by an ordered sequence of $l_0 + 1$ nodes as follows

$$\mathcal{P}(t_0) = (v_0^0,\ v_1^0, \cdots,\ v_{l_0-1}^0,\ v_{l_0}^0 = v_f^0). \tag{24}$$

It is assumed that v_1^0 is a free node owing to the high resolution representation of $\mathcal{W}$ close to x_0. The agent subsequently moves from v_0^0 to v_1^0. Let now t_1 be the time the agent is at the location $\mathsf{x}(t_1) = \mathsf{cell}_{\mathcal{G}(t_0)}(v_1^0)$ and let $\mathcal{C}_d(t_1)$ be the multiresolution cell decomposition of $\mathcal{W}$ around $\mathsf{x}(t_1)$ with a corresponding topological graph $\mathcal{G}(t_1)$. Applying again the $\mathcal{A}^*$ algorithm we compute a (perhaps new) path in $\mathcal{G}(t_1)$ from $v_0^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1))$ to $v_f^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}_f)$ if such a path exists. Let $\mathcal{P}(t_1)$ be given by the ordered sequence of $l_1 + 1$ nodes as follows

$$\mathcal{P}(t_0) = (v_0^1,\ v_1^1, \cdots,\ v_{l_1-1}^1,\ v_{l_1}^1 = v_f^1). \tag{25}$$

```
BEGIN PATH PLANNING ALGORITHM
{
    $i = 0$;
    $\mathsf{x}(t_i) \leftarrow \mathsf{x}_0$;
    while $\|\mathsf{x}(t_i) - \mathsf{x}_f\|_\infty \geq 1/2^{J_{\max}}$
    {
        compute $\mathsf{rm}(\mathsf{x}, i)$ for all $\mathsf{x} \in \mathcal{W}$;
        construct $\mathcal{C}_d(i)$ at level $J_{\min}$;
        construct $\mathcal{G}(i) = \big(E(i), V(i)\big)$;
        $v_1^i \leftarrow \mathsf{node}_{\mathcal{G}(i)}\big(\mathsf{x}(t_i)\big)$;
        $v_f^i \leftarrow \mathsf{node}_{\mathcal{G}(i)}(\mathsf{x}_f)$;
        $\mathcal{P}(i) \leftarrow$ Astar$\big(v_1^i, v_f^i, \mathcal{G}(i)\big)$;
        if $\mathcal{P}(i) = \varnothing$
            report FAILURE; break;
        $\mathsf{x}(t_{i+1}) \leftarrow \mathsf{cell}_{\mathcal{G}(i)}(v_2^i)$;
        Move to $\mathsf{x}(t_{i+1})$;
        $i \leftarrow i + 1$;
    }
}
END PATH PLANNING ALGORITHM
```

Figure 16: Pseudo-code implementation of proposed multiresolution path planning scheme.

The agent subsequently moves to v_1^1 at location $\mathsf{x}(t_2) = \mathsf{cell}_{\mathcal{G}(t_1)}(v_1^1)$ at time t_2.

In general, assume the agent is at location $\mathsf{x}(t_i)$ at time t_i. We construct a multiresolution decomposition $\mathcal{C}_d(t_i)$ of $\mathcal{W}$ around $\mathsf{x}(t_i)$ with a corresponding graph $\mathcal{G}(t_i)$. The $\mathcal{A}^*$ algorithm yields a path $\mathcal{P}(t_i)$ in $\mathcal{G}(t_i)$ of length $l_i + 1$,

$$\mathcal{P}(t_i) = (v_0^i,\ v_1^i, \cdots,\ v_{l_i-1}^i,\ v_{l_i}^i = v_f^i), \tag{26}$$

where $v_0^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}(t_i))$ and $v_f^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}_f)$. This iteration process terminates at some time t_f when $\|\mathsf{x}(t_f) - \mathsf{x}_f\|_\infty < 1/2^{J_{\max}}$. At the last step the agent moves from $\mathsf{x}(t_f)$ to x_f. A pseudo code implementation of the multiresolution path planning algorithm is given in Fig. 16. Note that the actual path followed by the agent is given by the sequence of nodes $\big\{\mathsf{node}_{\mathcal{G}(t_0)}\big(\mathsf{x}(t_0)\big), \mathsf{node}_{\mathcal{G}(t_1)}\big(\mathsf{x}(t_1)\big), \cdots, \mathsf{node}_{\mathcal{G}(t_f)}\big(\mathsf{x}(t_f)\big)\big\}$.

2.4.2 $\mathcal{D}^*$-lite path planning algorithm

The $\mathcal{D}^*$ algorithm has been originally proposed by Stentz[135, 136] for planning a path in unknown or partially known environment. Prior to $\mathcal{D}^*$, several replanning strategies have been proposed to solve dynamic planning problems for locally-directed wandering[110], local modification of initial path[72], and obstacle perimeter detouring[86]. Although these methods are complete, they are suboptimal and computationally inefficient. On the contrary, $\mathcal{D}^*$ produces an optimal path by adopting an efficient incremental search scheme to reduce the time required to replan. In particular, $\mathcal{D}^*$ is more appropriate when dealing with an environment having a large number of states, by reusing information from the previous search to find the solution at the next step. Koenig and Likhachev introduced Lifelong Planning $\mathcal{A}^*$ (LPA)[70] which employs heuristic estimate like $\mathcal{A}^*$, while reusing information from previous searches to find a solution much faster than solving each iteration from scratch. Furthermore, Koenig and Likhachev presented the $\mathcal{D}^*$-lite algorithm, derived from the LPA algorithm, which implements the same planning strategy as $\mathcal{D}^*$ but is algorithmically different. The $\mathcal{D}^*$-lite algorithm simplifies the maintenance of priority queues, thus it does not use complicated conditional statements, thus ending up with shorter codes than the original $\mathcal{D}^*$ algorithm. In the sequel, we employ the $\mathcal{D}^*$-lite algorithm to the path planning problem on a non-trivial environment. By comparing the $\mathcal{D}^*$-lite algorithm with the multiresolution path planning algorithm, we discuss the benefits and shortfalls of using the multiresolution path planning algorithm over the $\mathcal{D}^*$-lite algorithm.

We apply the Haar wavelet transform up to resolution level $J \geq J_{\min}$ to obtain the wavelet decomposition of rm, given by

$$\mathsf{rm}(x,y) = \sum_{k,\ell=0}^{2^J-1} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j=J}^{N-1}\sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y). \tag{27}$$

A uniform cell decomposition $\mathcal{C}_d^J$ at level J on $\mathcal{W}$ is induced from Eq. (27), and is comprised of cells $c^J_{k,\ell}$ of dimension $1/2^J \times 1/2^J$. We adopt the eight-connectivity relationship between the cells. The connectivity relationship is easily found by book-keeping the location of each cell through the indices k and ℓ. It should be noted that the adjacency relationship will remain the same throughout the replanning, but the edge costs will change incrementally to incorporate the information from the previous step.

Suppose the agent is equipped only with a proximity sensor that senses the environment close to the current location with high accuracy. That is, the sensor provides information about classification of the neighboring environment by a free region or obstacle region. Let $\mathcal{S}(i)$ be the known region up to $t = t_i$ using a sensor with the range r_J, defined by

$$\mathcal{S}(i) = \mathcal{S}(i-1) \cup \mathcal{N}(\mathsf{x}_i, r_J) \tag{28}$$

where x_i is the current location of the agent at $t = t_i$ and $\mathcal{N}(\mathsf{x}_i, r_J)$ represents the effective sensory region at that moment. In Eq. (28) it is assumed that the agent navigates an initially unknown environment while updating the map from the collected information. In order to take this process into consideration for replanning, we assign a conditional cost to each edge (u, v) which depends on the relative location of the edges to $\mathcal{S}$ as follows,

$$\mathcal{J}(u,v) = \begin{cases} \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, & \text{if } u,\ v \in \mathcal{S}, \\ \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, & \text{if } u,\ v \in \mathcal{W} \setminus \mathcal{S}. \end{cases} \tag{29}$$

It follows that for the edges outside $\mathcal{S}$ we simply impose the traversal cost between nodes considering the uniform size of cells. With the traversal cost, a general path planning algorithm such as Dijkstra's or $\mathcal{A}^*$ simply computes a shortest path from an initial node to the goal node which might pass through obstacles outside $\mathcal{S}$. Nevertheless, whenever the map is updated using contingent information from the sensor, we accordingly update the corresponding edge costs by appending the obstacle cost to each edge as given in (29).

The main $\mathcal{D}^*$-lite path planning algorithm proceeds as follows. From the uniform cell decomposition and the corresponding graph, we solve for an initial path from $v_0 = v_{\text{start}}$ to v_{goal} assuming only the distance cost for edge weights. Let v_1 be the node next to v_0 in the path. The agent subsequently moves from v_0 to v_1. At time $t = t_1$ when the agent is located at v_1, the algorithm continues to scan the graph for changed edge costs. If any edge costs have changed, then the algorithm updates the corresponding edge weights. Subsequently, a new path is computed from v_1 to v_{goal}, by incorporating the updated edge weights. It should be noted that if no edge costs have changed the agent moves to the successive node v' in the previous path that has the minimum cost $\mathcal{J}(v_{\text{last}}, v') + g(v')$.

Similar to $\mathcal{A}^*$, the $\mathcal{D}^*$-lite algorithm also incorporates a heuristic estimate to choose the nodes from a priority queue. However, as the agent detects changes in the edge costs, the priority queue should be reordered to render itself consistent. This might be an expensive task, so instead of reordering the priority queue every time, Koenig and Likhachev utilizes a *dynamic heuristic constant* k_m[71] to keep the priority queue unaltered regardless of the change of the edge costs. This reduces the computational overhead, resulting in faster execution. The iteration terminates at some time t_l when the goal node is reached. A pseudo code implementation of the $\mathcal{D}^*$-lite incremental path planning algorithm is given in Fig. 16. Note that the actual path followed by the agent is given by the sequence of nodes $\{v_0 = v_{\text{start}}, v_1, \cdots, v_{l-1}, v_l = v_{\text{goal}}\}$.

2.5 Simulation Results

2.5.1 Simulation results for the proposed algorithm

In this section we present simulation results of the proposed algorithm for a non-trivial scenario. The environment $\mathcal{W}$ is an actual topographic (elevation) map of a US state, shown in Fig. 18. The environment is assumed to be square of dimension

```
BEGIN $\mathcal{D}^*$-LITE PATH REPLANNING ALGORITHM
{
    $i = 0$;
    compute $\mathsf{rm}(\mathsf{x})$ for all $\mathsf{x} \in \mathcal{W}$;
    construct $\mathcal{C}_d$ at level $J_{\max}$;
    construct $\mathcal{G} = \big(E, V\big)$;
    $v_{\text{start}} \leftarrow \mathsf{node}_{\mathcal{G}}(\mathsf{x}_0)$;
    $v_{\text{goal}} \leftarrow \mathsf{node}_{\mathcal{G}}(\mathsf{x}_f)$                                        ;
    Initialize();
    ComputeShortestPath$(v_{\text{start}}, v_{\text{goal}}, \mathcal{G})$;
    $v_i \leftarrow v_{\text{start}}$;
    $v_{\text{last}} \leftarrow v_i$;
    while $(v_i \neq v_{\text{goal}})$
    {
        $i \leftarrow i + 1$;
        $v_i = \text{argmin}_{v' \in Adj(v_{\text{last}})}\big(\mathcal{J}(v_{\text{last}}, v') + g(v')\big)$;
        Move to $v_i$;
        Scan graph for changed edge costs;
        If any edge costs changed;
        {
            $k_m \leftarrow k_m + h(v_{\text{last}}, v_i)$;
            $v_{\text{last}} \leftarrow v_i$;
            For all directed edges $(u, v) \in E$ with changed edge costs
            {
                Update the edge cost $\mathcal{J}(u, v)$;
                UpdateVertex$(v)$;
            }
            ComputeShortestPath$(v_i, v_{\text{goal}}, \mathcal{G})$;
        }
    }
}
END $\mathcal{D}^*$-LITE PATH REPLANNING ALGORITHM
```

Figure 17: Pseudo-code implementation of $\mathcal{D}^*$-lite path planning scheme.

128×128 units. Hence the finest possible resolution is $N = 7$. Taking into account the available memory of the micro-controller, we choose the fine level as $J_{\max} = 6$ and the coarse level as $J_{\min} = 3$. This makes the total number of nodes in the graph not to exceed the maximum count of 256 that corresponds to the maximum allowable variable size of the micro-controller. The ranges at distinct levels of resolution are selected $(r_6, r_5, r_4) = (8, 15, 30)$ units from the current location. It follows that the fine resolution at level $J = 6$ is used inside an area of 16×16 units around the current location of the agent, and the coarser resolutions at levels $J = 5$ and $J = 4$ are used inside 30×30 and 60×60 units.

The objective of the UAV is to follow a path from the initial position to the final position while circumventing the obstacles over a certain elevation threshold. Since the on-line path planning problem at the finest resolution is computationally prohibitive, the proposed algorithm accommodates the need of the on-line implementation for the micro-controller by limiting the amount of the information to process, thus computing an immediate path with high accuracy within the allowable time scale of the micro-controller.

The results from the multiresolution path planning algorithm are shown in Fig. 18. Specifically, Fig. 18 shows the evolution of the path at different time steps as the agent moves to the final destination. Figure 18(a) shows the agent's position at time step $t = t_5$ along with the best proposed path to the final destination by a dashed-dot line at that time. The actual path followed by the agent is drawn by a solid line. Similarly, Fig. 18(b) shows the agent's position at time step $t = t_{21}$. As seen in Fig. 18(c), the actual path followed by the agent differs from the one predicted in either Figs. 18(a) or 18(b). This is due to the fact that at time t_5 and t_{21} the agent does not have complete information for upcoming location up to confident level. In particular, Fig. 18(b) shows that as the agent gets closer to the obstacle, it recognizes the presence of obstacles and redirects the path to avoid the obstacle. Finally, the agent reaches the final destination x_f in a collision free manner, as seen in Fig. 18(c).

2.5.2 Simulation results for the $\mathcal{D}^*$-lite algorithm

In this section we present simulation results of the $\mathcal{D}^*$-lite path planning algorithm for the same environment used in the previous simulation. It is assumed that the agent navigates over the unknown environment, while updating the map with the information gathered from a proximity sensor. We adopt a uniform cell decomposition of cell size the $J_{\max} = 6$ which is the same to the finest level used in the previous section. The range of the proximity sensor is chosen $r_6 = 7$, thus resulting in a high resolution window by a 7×7 square grid.

The results from the $\mathcal{D}^*$-lite path planning algorithm are shown in Fig. 19. Also, Fig. 19 shows the evolution of the path at different time steps as the agent moves to the final destination. At each step, the best proposed path is drawn by a dashed-dot line and the actual path followed by the agent is drawn by a solid line. As seen in Fig. 19(c), the actual path differs significantly from the one predicted in either Figs. 19(a) or 19(b). This is attributed to the fact that the environment is unknown a priori, and the initial path is computed using the distance cost outside the high resolution horizon. Hence, as shown in Fig. 19(a), the agent is unable to anticipate the existence of the obstacles outside the high resolution horizon. Nonetheless, as the agent gets closer to the obstacles, the $\mathcal{D}^*$-lite algorithm effectively replans the entire path avoiding the obstacles, reaching the final destination.

2.6 Comparison

The proposed multiresolution path planning algorithm was written in C code and implemented on an on-board autopilot equipped with a Rabbit RCM-3400 micro-controller. Because the micro-controller has limited computational resources (10,000 instructions per second, and 512 KB RAM for handling variables), the code has been written giving special attention not only to the accuracy of the output, but

Table 1: Computational cost of the proposed algorithm by the on-board autopilot.

Multiresolution cell decomposition using FLWT	452 [msec]
Construct the connectivity relationship and $\mathcal{G}$	292 [msec]
Compute a path using $\mathcal{A}^*$ employing the binary heap	202 [msec]
Average number of nodes of each $\mathcal{G}$	~200

also to the computational speed during implementation. Specifically, most of the computations for the proposed algorithm is done using integer arithmetic. Given a risk measure rm of integer samples, the integer fast lifting wavelet transform provides the approximation and detail coefficients that are used to construct the adjacency relationship between cells. The $\mathcal{A}^*$ algorithm is then called to find the shortest path in the graph associated with the wavelet decomposition.

Table 1 shows the computational cost of the proposed path planning algorithm using the on-board autopilot. One step of the path planning iteration takes 946 [msec] for execution. With the knowledge of the execution time of the proposed algorithm, we actually choose to implement the proposed path planning algorithm on-line every three seconds. Hence, the autopilot manages not only to execute the basic tasks such as data acquisition and processing, inner loops control, and etc., but also to plan a path in a seamless manner.

We compared the computational costs between the proposed multiresolution path planning algorithm and the $\mathcal{D}^*$-lite algorithm, using different simulation results for several cases. The simulations were carried out on an IBM-PC (Pentium M 2.0 GHz, 1 GB RAM), based on codes written in C for implementing both path planning algorithms. The proposed path planning algorithm accomplishes the path planning objective of reaching the goal in a fewer number of iterations, as shown in Table 2, than the $\mathcal{D}^*$-lite algorithm. This is due to the fact that the proposed algorithm effectively manages the information of the coarser resolutions so that a preferred path is computed over the approximation of $\mathcal{W}$. The $\mathcal{D}^*$-lite algorithm, however,

Table 2: The computational cost comparison between the multiresolution path planning v/s the $\mathcal{D}^*$-lite.

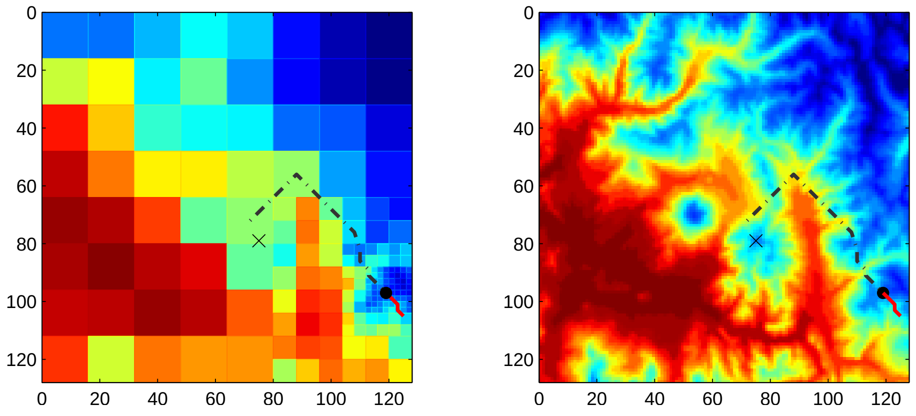
Items	Scenario I		Scenario II		Scenario III		Scenario IV		Scenario V	
	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet
# iteration	35	31	93	50	61	40	123	52	44	43
# nodes in $\mathcal{G}$	4096	201	4096	194	4096	192	4096	185	4096	194
Data processing [msec]	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03
Adjacency search [msec]	-	0.977	-	0.987	-	0.969	-	0.94	-	0.958
$\mathcal{A}^*$ search [msec]	-	0.1	-	0.125	-	0.094	-	0.138	-	0.066
Init. $\mathcal{D}^*$-lite search [msec]	1.87	-	2.03	-	2.03	-	1.87	-	2.03	-
$\mathcal{D}^*$-lite update [msec]	4.1	-	23.8	-	11.3	-	43.9	-	12.7	-
Total Comp. time [msec]	5.97	33.387	25.83	55.6	13.33	42.53	45.77	56.056	14.73	44.032
Computational cost (%)	17.8	100	46.46	100	31.35	100	81.65	100	33.45	100
Memory cost (%)	2037.8	100	2111.3	100	2133.3	100	2214.1	100	2111.3	100

relies on the information at finer resolution which is unveiled up to the current time, thus requiring the agent to explore the environment and to replan the path along the movement of the agent. In the worst case, the total number of iterations by the $\mathcal{D}^*$-lite algorithm increases significantly (e.g. Scenario IV) because of the existence of unknown obstacles.

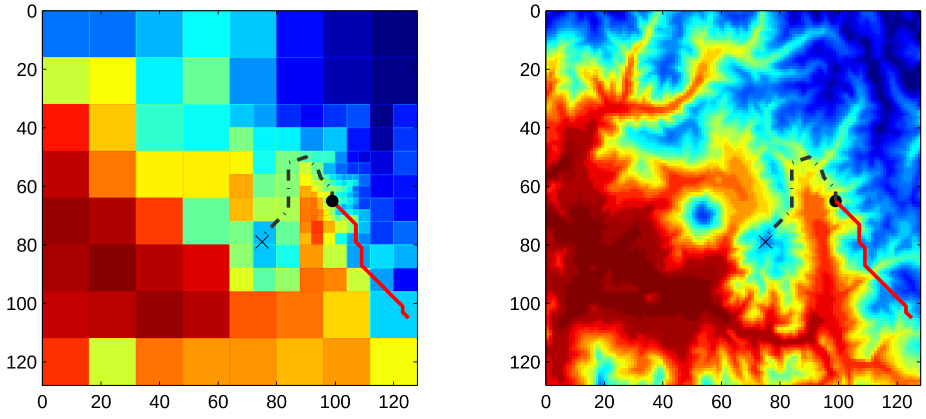
The total computation time of the proposed algorithm is obtained by adding the computation times throughout each iteration. For the $\mathcal{D}^*$-lite algorithm, the total computation time consists of the time for initializing and updating, which is shown to be smaller than that of the proposed algorithm. The $\mathcal{D}^*$-lite algorithm is computationally efficient in the sense that it reuses information from the previous step. In contrast, most of the computations in the proposed algorithm are devoted to the construction of the adjacency list at each planning step, as shown in Table 2. The performance of the proposed algorithm can thus be improved by using, say, four-connectivity instead of eight-connectivity during the adjacency search algorithm. This will possibly halve the computation time with little performance degradation. By the inherent benefit from the multiresolution decomposition, the proposed algorithm requires little memory as shown in Table 2 compared to $\mathcal{D}^*$-lite. For on-line, on-board path planning, the proposed algorithm has the advantages of scalability according to the available on-board computational resources.

2.7 Summary

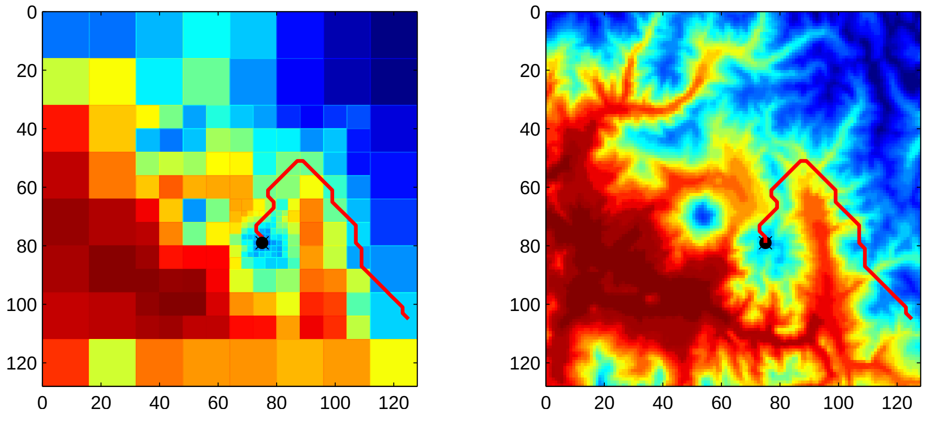
Autonomous path planning for small UAVs imposes severe restrictions on control algorithm development, stemming from the limitations imposed by the on-board hardware and the requirement for on-line implementation. In this chapter we have proposed a method to overcome this problem by using a new multiresolution path planning scheme. The algorithm computes at each step a multiresolution representation of the environment using the fast lifting wavelet transform. By utilizing most of integer arithmetic of FLWT, the computational cost is significantly reduced. The idea is to employ high resolution close to the agent (where is needed most), and a coarse resolution at large distances from the current location of the agent. As an added benefit, the connectivity relationship of the resulting cell decomposition can be computed directly from the nonzero detail coefficients of the wavelet transform. The algorithm is scalable and can be tailored to the available computational resources of the agent.



(a) $t = t_5$

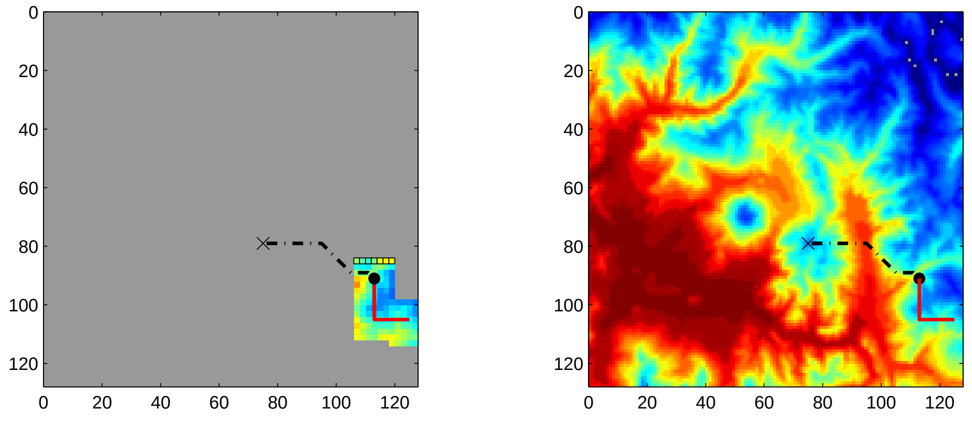


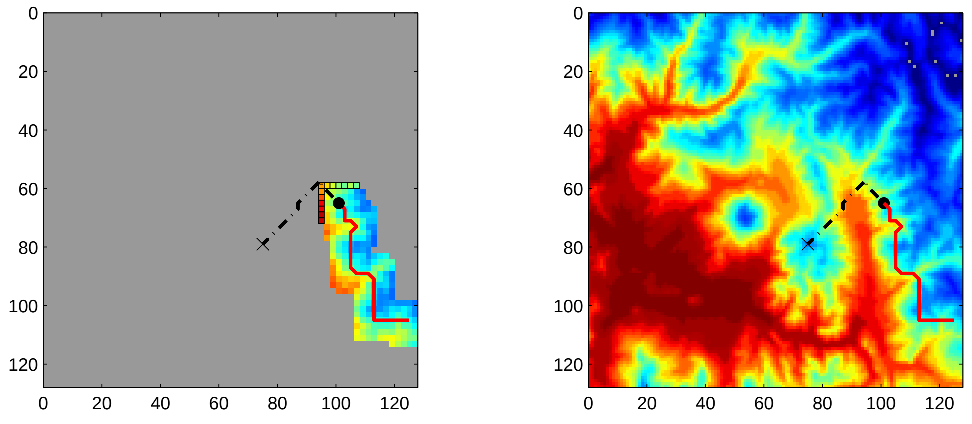
(b) $t = t_{21}$



(c) $t = t_f$

Figure 18: Path evolution and replanning. Dashed-dot lines represent the currently tentative optimal path obtained from the $\mathcal{A}^*$ algorithm, based on the available multiresolution approximation of the environment at different time steps. Solid lines reveal the actual path followed by the agent.


(a) $t = t_{14}$


(b) $t = t_{30}$

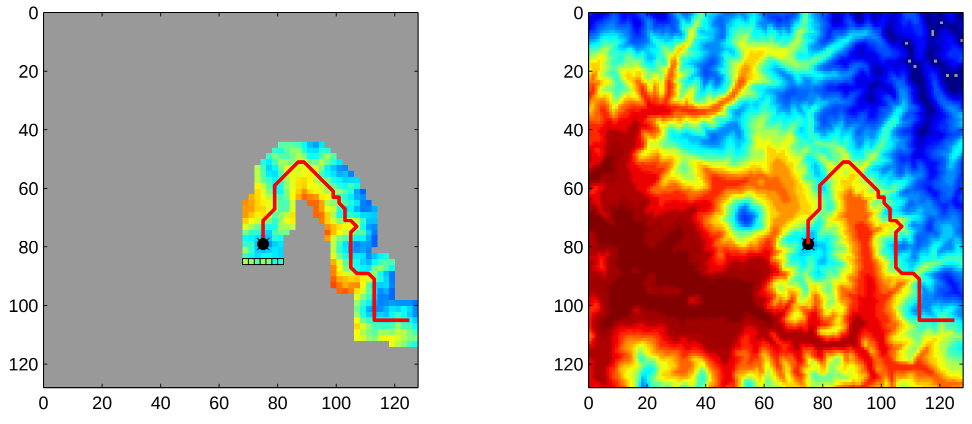

(c) $t = t_f$

Figure 19: Path evolution and replanning using the $\mathcal{D}^*$-lite algorithm. Dashed-dot lines show the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the high resolution area. The actual path followed by the agent is drawn by solid lines.

CHAPTER III

ON-LINE PATH SMOOTHING USING PATH TEMPLATES

3.1 Introduction

Guidance and navigation control of mobile agents has been an important research topic for several decades. In particular, for unmanned aerial vehicles (UAVs), the ability of fully automated guidance and navigation control allows the UAVs to accomplish missions under various circumstances with minimal human intervention. As a matter of fact, because of the stringent operational requirements and the restrictions imposed on UAVs by autonomy, safety, efficiency, etc., a complete solution to fully automated guidance and navigation control of UAVs is challenging. Many researchers suggest that breaking the problem into several subproblems such as path planning, trajectory smoothing, trajectory tracking, and etc., makes it easy to solve by a hierarchical control structure [93, 10, 94].

It is assumed that a planned path is given by a series of way-points from the top-level path planner. Traditionally, a simple implementation of the way-points tracking control is used for guidance purpose. A better implementation of the guidance and navigation control incorporates a smooth path by taking into account the dynamic constraints of UAVs. The way-points can then be connected to generate smooth path segments, which preserves the continuity of curvature between line and arc segments while minimizing the maximum curvature on the curve [61, 122]. In Ref. [6], the authors proposed a dynamic trajectory smoothing algorithm by which the path segments in straight-lines are smoothed to yield an extremal trajectory with explicit consideration of the kinematic constraint of a fixed-wing UAV.

In contrast to the explicit consideration of the kinematic constraint of the UAV, a spline-based path generation method has been widely adopted when computing the smooth, dynamically feasible trajectory for UAVs. A series of cubic splines was employed in Ref. [56] to connect straight line segments in a near-optimal manner. The authors in Ref. [147] presented an implicit time-parameterization of the trajectory using a B-spline representation. Designing an obstacle-avoiding B-spline path has been dealt with by Berglund et al.[15], whereas the real-time modifications of a spline path is proposed in Ref. [39]. The advantage of employing the (B-)splines in generating a smooth path is dictated by the fact that the path can be represented by using smaller number of parameters than using complete description of path. Accordingly, both for path optimization and for on-line implementation, it is straightforward to deal with small number of parameters when generating paths that are subject to the given obstacle constraints at the minimum computational cost.

The obstacle avoidance path planning problem using B-splines involves a constrained optimization such that the path should not only avoid forbidden regions but also become a flyable trajectory. In Ref. [87], a polygonal channel comprised of piecewise polylines serves as constraint equations while a B-spline curve is optimized using quadratic programming. This has been made possible by adopting tight linear envelopes for splines [88], by which a B-spline is represented as an approximate bounding polygon. In their approach, one dimensional B-splines are utilized to describe a smooth path subject to the channel constraints. In this paper, we extend the results in Ref. [88] in two dimensions, thus incorporating a two dimensional B-spline curve instead of a B-spline function. To this end, we formulate an optimization problem similar to the channel problem in Ref. [87] with constraints being given as geometric constraints.

Incorporating a high-level path planning algorithm such as $\mathcal{D}^*$-lite, we present a path smoothing algorithm using a set of path templates. Instead of smoothing the entire path from an initial position to the goal position, we smooth the path segments over a finite planning horizon with respect to the current position of the UAV. This approach is somewhat similar to implementing a receding horizon concept in the path generation stage. To the problem of trajectory generation and collision avoidance, the receding horizon concept has been successfully implemented showing that it is an effective way to reduce the computational cost [43, 67, 129]. Each segment of B-spline curves are then stitched together to the B-spline curve that corresponds to the next path sequence, thus overall path remains smooth. Although the explicit dynamics of the UAV are not directly dealt with for smoothing the path segments, this approach has the advantage of having minimal on-line computational cost since most of computation is done off-line.

3.2 Tight Envelope for B-spline curves

3.2.1 Tight envelope for B-spline function

An one-dimensional spline function b is expressed by

$$b(u) = \sum_{j=0}^{m} b_j N_j^d(u) \tag{30}$$

where b_j are control points and $N_j^d(u)$ are the B-spline basis functions of degree d which are defined over a non-decreasing knot sequence $\{u_k\}$ such that $u_0 \leq u_1 \leq \cdots \leq u_{m+d+1}$. The number of knots is determined by the sum of the number of control points $(m+1)$ and the B-spline order $(d+1)$. The first and the last knots of the sequence should have multiplicity $(d+1)$ for a B-spline to pass the the first and the last control points, in such that $u_0 = u_1 = \cdots = u_d$ and $u_{m+1} = u_{m+2} = \cdots = u_{m+d+1}$, respectively. The B-spline basis functions are computed by the well-known *Cox-de*

Boor recursion formulas [34] from the degree 0 to d as follows,

$$N_j^0(u) = \begin{cases} 1 & \text{if } u_j \le u < u_{j+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{31a}$$

$$N_j^d(u) = \frac{u - u_j}{u_{j+d} - u_j} N_j^{d-1}(u) + \frac{u_{u+d+1} - u}{u_{j+d+1} - u_{j+1}} N_{j+1}^{d-1}(u). \tag{31b}$$

The B-spline basis function has several useful properties. Among them, it is well known that B-spline basis function has local support [109], that is $N_j^d(u)$ is a non-zero polynomial over a knot span $[u_j,\ u_{j+d+1})$, or given any span $[u_j,\ u_{j+1})$, at most $(d+1)$ B-spline basis functions of degree d are non-zero. This local support property is important to curve design, since it allows to modify the B-spline curve locally without changing the entire shape. Figure 20 shows the cubic B-spline basis functions of degree $d = 3$ over the knot sequence $u \in [0,\ 1]$.

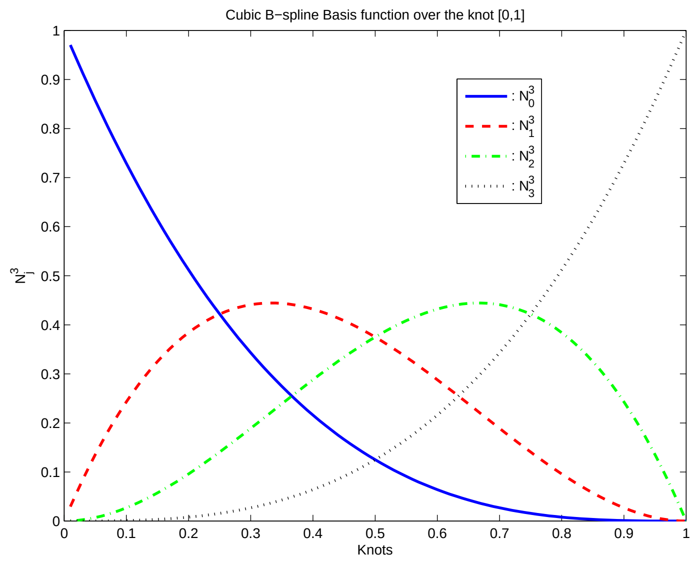


Figure 20: B-spline basis functions N_j^3 over the knot $u \in [0,\ 1]$.

Let the control polygon of the B-spline ℓ be defined by piecewise line segments connecting the control points b_j at the Greville abscissae[48],

$$u_j^* = \sum_{i=j+1}^{j+d} u_i / d, \tag{32}$$

such that at each Greville abscissae it satisfies $\ell(u_j^*) = b_j$. Accordingly, an envelope of the B-spline specifies a bound on the distance between b and its control polygon ℓ. This envelope should provide a good estimate of the shape of the B-spline by approximating the spline using less information. Although a number of bounds for splines are proposed in the literature, the bound derived in Ref. [99, 88] is known to be a tight, quantitative bound. Hence, it is possible to approximate the B-splines by piecewise linear envelopes, as the envelopes carry most of salient information about the curve itself. In light of this, the envelopes have advantage of being incorporated in the B-spline optimization problem, thus simplifying the analysis.

The envelopes in Ref. [88] are expressed in terms of the weighted second difference of the control points,

$$\Delta_2 b_j \triangleq b'_{j+1} - b'_j \qquad \text{where} \;\; b'_j = \frac{b_j - b_{j-1}}{u_j^* - u_{j-1}^*}, \tag{33}$$

and by the non-negative and convex functions over the interval $[u_k^*, u_{k+1}^*]$ $(k = 0, 1, \cdots, m)$ as follows,

$$\beta_{ki} = \begin{cases} \sum_{j=i}^{\overline{k}} (u_j^* - u_i^*) N_j^d(u) & i > k, \\ \sum_{j=\underline{k}}^{i} (u_i^* - u_j^*) N_j^d(u) & j \leq k, \end{cases} \tag{34}$$

where, $\underline{k}$ and $\overline{k}$ denote the index of the first and the last B-spline basis functions which are nonzero over the corresponding interval, respectively. Subsequently, the distance between the spline function b and its control polygon ℓ is calculated as follows,

$$b - \ell = \sum_{i=\underline{k}}^{\overline{k}} \Delta_2 b_i \beta_{ki}. \tag{35}$$

Furthermore, by choosing the maximum and minimum variation of the weighted second difference as $\Delta_i^- = \min\{0, \Delta_2 b_i\}$, $\Delta_i^+ = \max\{0, \Delta_2 b_i\}$, we obtain the maximum offsets in both positive and negative direction with respect to the control polygon, which, in turn, become the upper and lower bounds of the spline function with respect to the control polygon,

$$\ell + \sum_{i=\underline{k}}^{\overline{k}} \Delta_i^- \beta_{ki} \le b \le \ell + \sum_{i=\underline{k}}^{\overline{k}} \Delta_i^+ \beta_{ki}. \tag{36}$$

Since the β_{ki}'s are non-negative and convex function over the corresponding interval $[u_k^*,\ u_{k+1}^*]$, the maximum function values occur at each end of the interval, i.e. at each Greville abscissae. (See Fig. 21). Then the piecewise linear functions $\underline{e}$ and

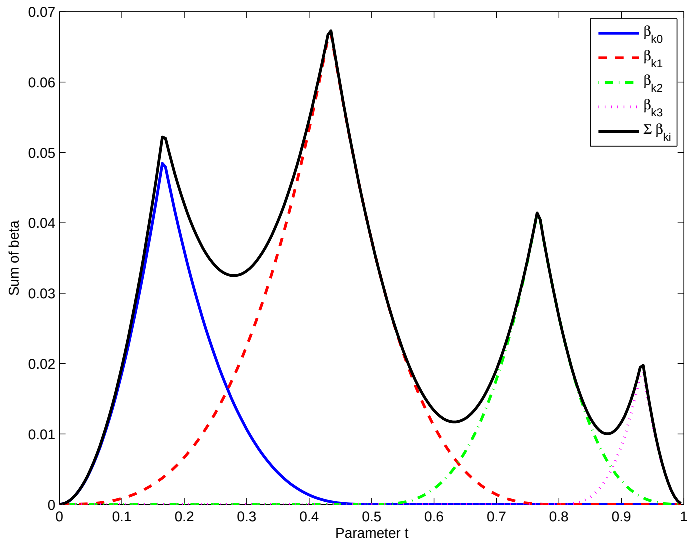


Figure 21: Non-negative and convex functions β_{ki}.

$\overline{e}$ that interpolate their values at each Greville abscissa can be employed to provide tight envelopes of the spline function,

$$\begin{aligned} \underline{e} &= \ell + \mathcal{L}\left(\sum \Delta_i^- \beta_{ki}(u_k^*), \sum \Delta_i^- \beta_{k+1,i}(u_{k+1}^*) \right), \\ \overline{e} &= \ell + \mathcal{L}\left(\sum \Delta_i^+ \beta_{ki}(u_k^*), \sum \Delta_i^+ \beta_{k+1,i}(u_{k+1}^*) \right), \end{aligned} \tag{37}$$

where $\mathcal{L}(\cdot,\cdot)$ denotes a linear interpolation between two values. Therefore, the maximal bounds from the B-spline function to its control polygon are obtained in a simple form,

$$\underline{e} \le b \le \overline{e}. \tag{38}$$

Figure 22 shows a cubic B-spline function b over the knot sequence $u \in [0,\ 1]$. The bounding envelopes $\underline{e}$ and $\overline{e}$ are drawn by dotted and dashed-dot lines, respectively.

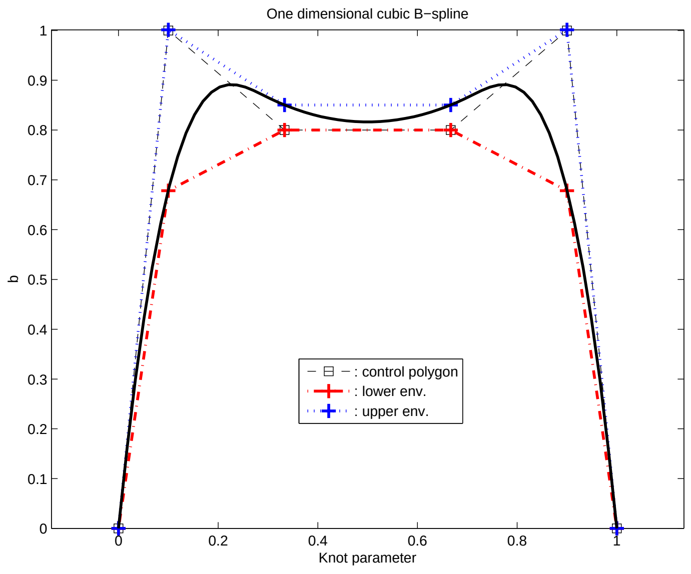


Figure 22: One dimensional cubic B-spline bounding envelopes.

3.2.2 Tight envelope for planar B-spline curves

A two dimensional planar B-spline curve $\boldsymbol{b}(u) = [b^1(u)\ b^2(u)]^\mathsf{T}$ is expressed in terms of the B-spline basis functions,

$$\boldsymbol{b}(u) = \sum_{j=0}^{m} \boldsymbol{b}_j N_j^d(u), \tag{39}$$

where, $\boldsymbol{b}_j = [b_j^1\ b_j^2]^\mathsf{T}$ are the control points. It is also assumed that the B-spline curve is clamped at the first and last control points by assigning the $(d+1)$ multiple knots at each the first and last knots.

At each Greville abscissa u_k^*, the one-dimensional bound by Eq. (38) constitutes a two-dimensional bounding box, whose ith axis is determined by the one-dimensional envelope as

$$\underline{e}^i(u_k^*) \leq b^i(u_k^*) \leq \overline{e}^i(u_k^*) \qquad i = 1, 2. \tag{40}$$

Let this axis-aligned bounding box be denoted by S^k, then the curve segment $\boldsymbol{b}(u)$, $u \in [u_k^*, u_{k+1}^*]$, lies in a convex combination of S^k and the consecutive box S^{k+1} owing to the linearity of $\underline{e}$ and $\overline{e}$, which is denoted by

$$H^k = \mathcal{L}(S^k,\ S^{k+1}). \tag{41}$$

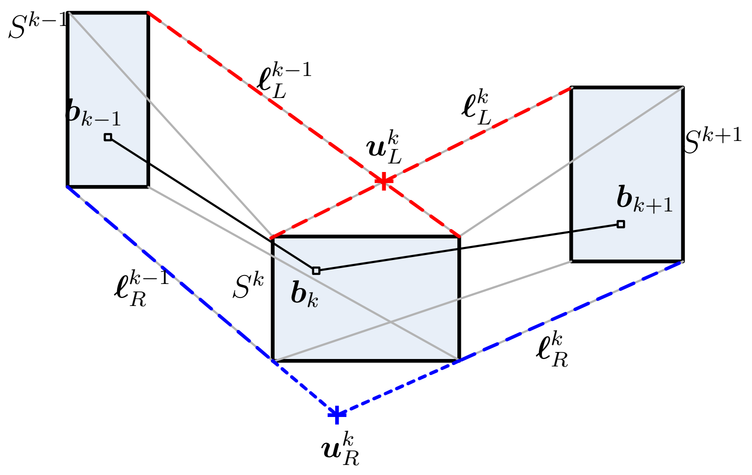


Figure 23: Constructing the envelope of a planar curve from neighboring bounding boxes.

Note that H_k is rendered as a convex hull of S^k and S^{k+1}, which is circumscribed by the edges of S^k and S^{k+1}, and lines connecting the corners of S^k and S^{k+1}. Let $\boldsymbol{v}_i^k$, $i = 1, \cdots, 4$, be the line segments connecting corresponding corners of S^k and S^{k+1}. Hence, $\boldsymbol{v}_1^k$ connects the lower left corner of S^k to the lower left corner of S^{k+1}, $\boldsymbol{v}_2^k$ connects the lower right corner of S^k to the lower right corner of S^{k+1}, and so on. Figure 23 shows these line segments. As mentioned above, the convex hull H^k over the knot $u \in [u_k^*, u_{k+1}^*]$ consists of parts of the edges of S^k and S^{k+1} and exactly two extra line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$ chosen among $\boldsymbol{v}_i^k$. The line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$ are separated by the line that connects the control points $\boldsymbol{b}_k$ and $\boldsymbol{b}_{k+1}$, thus $\boldsymbol{\ell}_L^k$ is denoted a left envelope line segment and $\boldsymbol{\ell}_R^k$ is a right envelope line segment. These line segments $\boldsymbol{\ell}_L^k$ and $\boldsymbol{\ell}_R^k$, $k = 0, \cdots, m$ are joined together to form piecewise linear envelopes of the B-spline curve $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$, respectively. It might be the case, however, where two line segments do not intersect each other such as the line segments $\boldsymbol{\ell}_R^{k-1}$ and $\boldsymbol{\ell}_R^k$ in Fig. 23. In order to form piecewise linear envelopes by a set of line segments, those line segments are extended to find the intersection point $\boldsymbol{u}_R^k$. Consequently, the envelopes $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$ are determined by a set of line segments between the intersection points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$. Figure 24 shows an example of two-dimensional bounding envelopes of the given B-spline curve, which reveals that the entire B-spline curve stays inside the envelopes of $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$.

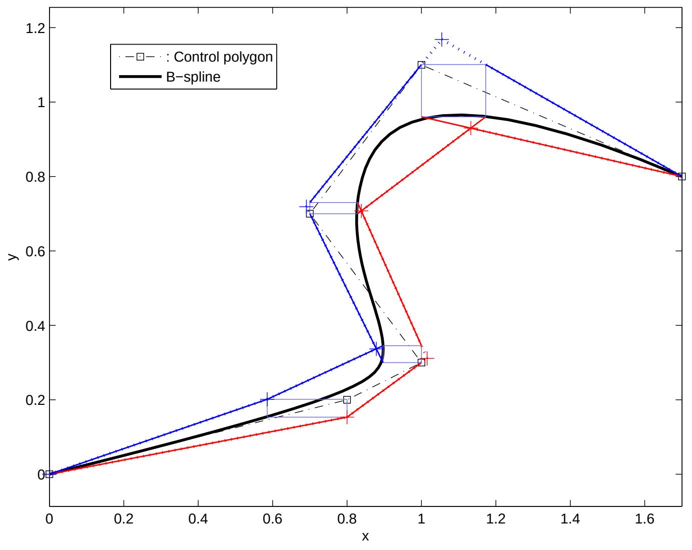


Figure 24: Bounding envelopes $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$ of two-dimensional cubic B-spline .

3.3 Obstacle avoidance path optimization

3.3.1 Channel constraints for obstacle avoidance

We formulate an optimization problem to calculate a smooth curve which avoids a prescribed obstacle region. We adopt a B-spline curve, due to the inherent smoothness property of B-spline, as a reference path to the UAV. Hence, the path given by a B-spline curve is rendered a flyable path by the agent. In particular, we let the path be placed inside a feasible channel, thus avoiding the obstacle region. The channel separates the obstacle and feasible regions by two distinct polygonal lines to yield geometric constraints for optimization problem.

In Ref. [87], linear inequality constraints are incorporated in the B-spline function optimization. Given input channel polygon, the inequality expressions are formulated in conjunction with the lower and upper envelopes $\underline{e}$ and $\overline{e}$ of the B-spline function such that the B-spline function should stay between the polygons. On the other hand, because we deal with a planar B-spline curve in this research, the channel constraints are formulated as geometric constraints. Hence, the given geometric channel should contain the envelopes of the B-spline curve.

To this end, we first introduce a signed distance-map function $f(\boldsymbol{x};\boldsymbol{\ell})$, $\boldsymbol{x} \in \mathbb{R}^2$ with respect to a polygonal line $\boldsymbol{\ell}$ to provide a metric for geometric constraints of the channel problem as follows,

$$f(\boldsymbol{x};\boldsymbol{\ell}) \triangleq s \cdot \min\{d_1,\ d_2\}, \tag{42}$$

where, $d_1 \in \mathcal{D}_1 = \{\|\boldsymbol{x} - \boldsymbol{c}_i\|,\ i = 0, \cdots, q\}$ is the distance from $\boldsymbol{x}$ to a corner point $\boldsymbol{c}_i$ of the polygonal line $\boldsymbol{\ell}_i$, $d_2 \in \mathcal{D}_2 = \{d(\boldsymbol{x}, \boldsymbol{\ell}_i),\ i = 0, \cdots, q-1\}$ is the perpendicular distance from $\boldsymbol{x}$ to the line segment $\boldsymbol{\ell}_i$ that connects two consecutive corner points $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$, and s is a sign value which decides the location of the point $\boldsymbol{x}$ with respect to $\boldsymbol{\ell}$ as follows,

$$s = \begin{cases} +1, & \boldsymbol{x} \in \mathcal{O}, \\ 0, & \boldsymbol{x} \in \boldsymbol{\ell}, \\ -1, & \boldsymbol{x} \notin \mathcal{O}. \end{cases} \tag{43}$$

Figure 25 shows this distance-map function value with respect to the given zig-zag shape polygonal line. Far-away points from the line have bigger values, whereas points close to the line yield smaller values. The information about the relative location of the points with respect to the polygonal line is determined by its sign.

In order to formulate the inequality constraints as similar to those in Ref. [87], first of all, it follows from Fig 23 and accompanying discussions that the envelopes of a planar B-spline curve are characterized by the feature points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$ of the envelopes $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$, respectively. Hence, if all these points are placed inside the feasible region, then each bounding box at $u = u_k^*$ of the B-spline curve will be completely contained in the feasible channel. Let $\boldsymbol{\ell}_L$ and $\boldsymbol{\ell}_R$ be the polygonal lines representing the obstacle boundaries, then the feature points $\boldsymbol{u}_L^k$ and $\boldsymbol{u}_R^k$ at each

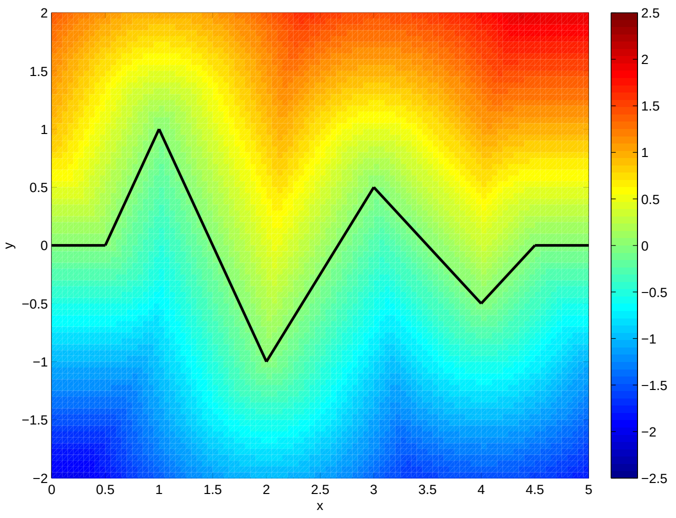


Figure 25: Signed distance map for an arbitrary polygonal line. The feasible region is characterized by the negative function values.

Greville abscissa $u = u_k^*$ should satisfy the following inequality relations,

$$f(\boldsymbol{u}_L^k; \boldsymbol{\ell}_L) \leq 0, \tag{44a}$$

$$f(\boldsymbol{u}_R^k; \boldsymbol{\ell}_R) \leq 0, \tag{44b}$$

where $k = 0, \cdots, m$.

Meanwhile, recall the fact that the envelopes of the B-spline curve are determined by the convex hull of each bounding box. Then the concave corner points of the obstacle boundaries, as marked by triangles in Fig. 26, should be excluded from the envelopes of the B-spline curve. To this end, we formulate inequality expressions utilizing the distance-map function in conjunction with the concave corner points and the piecewise linear envelopes as follow,

$$f(\boldsymbol{c}_l^{L_v}; \boldsymbol{e}_L) \geq 0 \tag{45a}$$

$$f(\boldsymbol{c}_m^{R_v}; \boldsymbol{e}_R) \geq 0 \tag{45b}$$

where, $l = 1, \cdots, n_{L_v}$ and n_{L_v} is the number of concave corner points of $\boldsymbol{\ell}_L$. Similarly, $m = 1, \cdots, n_{R_v}$ and n_{R_v} is the number of concave corner points of $\boldsymbol{\ell}_R$. Note that the

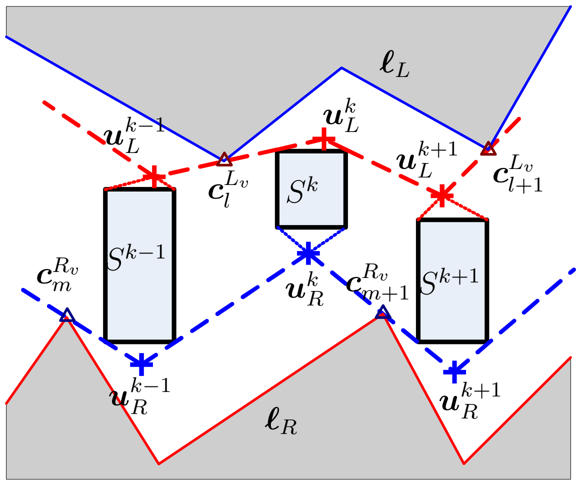


Figure 26: Geometric constraints formulation. The channel is given by two polylines $\boldsymbol{\ell}_L$ and $\boldsymbol{\ell}_R$, the envelope of the B-spline is drawn by the dashed lines, which is supposed to stay inside the channel.

positive condition in Eqs. (45) implies that the corner points are located outside the boundary envelopes of the B-spline curve. Consequently, the inequality constraints in Eqs. (44) and (45) ensure that the envelope of the B-spline stays between the channel, as depicted in Fig. 26.

3.3.2 Smooth curve optimization

We consider designing a smooth curve using a quartic B-spline, whose basis functions are computed from Eq. (31b) as degree 4 polynomials in terms of the knot parameter u. Hence, the quartic B-spline basis function preserves its continuity up to the third order derivative, thus resulting in the continuity of the derivative of curvature $\mathrm{d}/\mathrm{d}u(\kappa)$. Without loss of generality, the knot parameter is selected $u \in [0, 1]$, and the first and last knots have multiplicity 5 such as $u_0 = \cdots = u_4 = 0$ and $u_{m+1} = \cdots = u_{m+5} = 1$. Accordingly, the B-spline curve will be clamped at, or pass through, the first and last control points.

For the optimization, we manipulate $(m + 1)$ control points of the B-spline as $\boldsymbol{b}_j = [b_j^1 \; b_j^2]^\mathsf{T}$ $(j = 0, \cdots, m)$, which have direct influence on the shape of the curve. The number of control points is chosen along the complexity of the curve shape. In general, the curve shape is closely related to the given channel geometry, subsequently the number of control points can be opted for the minimum number required to get a B-spline curve inside the channel. The knot sequence is initially given arbitrary non-decreading numbers in $(0, 1)$ by taking into account the number of control points, then can be altered with knot insertion if the envelopes of the B-spline curve need to be refined [87].

Two different performance indices are adopted to compute curves for attaining distinct optimization goals. In order to keep the B-spline curve as close as possible to a straight line, for which $\Delta_2 \boldsymbol{b}_j = 0$, we employ the cost function

$$\mathcal{J}_1 = \sum_{i=1}^{m-1} (\Delta_2 \boldsymbol{b}_j)^{\mathsf{T}} (\Delta_2 \boldsymbol{b}_j), \tag{46}$$

which implicitly minimizes the curvature variation of B-spline curve, thus resulting in a smooth B-spline curve. On the other hand, we suppose that the arc length of the B-spline curve is approximately captured by the total length of the control polygon $\boldsymbol{\ell}$. Hence, for the shortest path, we employ the cost function as the sum of the length of the piecewise control polygon,

$$\mathcal{J}_2 = \sum_{i=0}^{m-1} \|\boldsymbol{\ell}_i\|_2. \tag{47}$$

The constraints for the optimization problem is comprised of both equality constraints and inequality constraints. The equality constraints stipulate boundary conditions for position, heading, and curvature at each end point at $u_0 = 0$ and $u_{m+5} = 1$ as follows,

$$\boldsymbol{b}(0) = \boldsymbol{p}_0 \ , \qquad \boldsymbol{b}(1) = \boldsymbol{p}_f, \tag{48a}$$

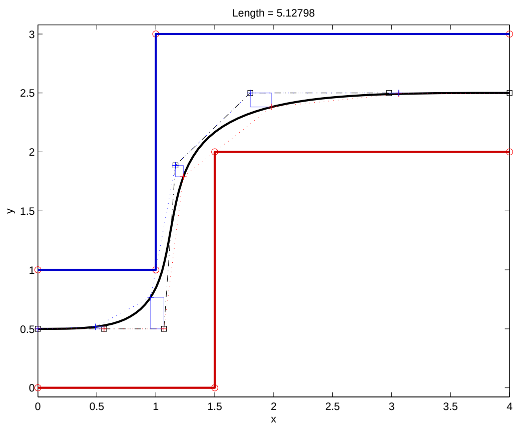
$$\psi(0) = \psi_0 \ , \qquad \psi(1) = \psi_f, \tag{48b}$$

$$\kappa(0) = \kappa_0 \ , \qquad \kappa(1) = \kappa_f, \tag{48c}$$

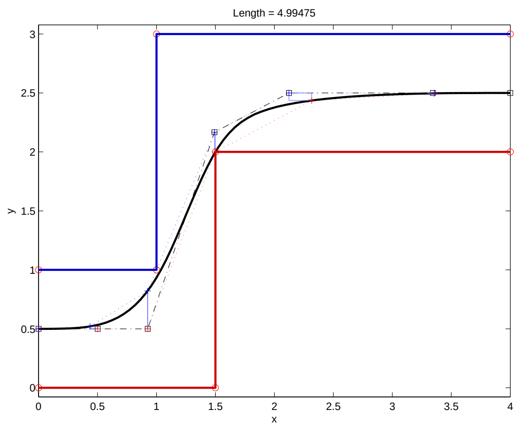
where $\psi(u)$ and $\kappa(u)$ are the heading and the curvature of the B-spline curve at each knot parameter. Inequality constraints are obtained from Eqs. (44) and (45) as the channel constraints.

The path optimization problem is formulated as follows. Given a knot sequence $\{u_k\}$, two polygonal lines for channel geometry, and boundary conditions for each end point, find a B-spline curve which minimizes the cost function in Eqs. (46) or (47) subject to the equality constraints in Eqs. (48) and the inequality constraints in Eqs. (44) and (45).

Figure 27(a) shows the optimization result using the cost function in Eq. (46). The constructed quartic B-spline curve is drawn by a solid line, and the envelopes are drawn by dashed lines. The B-spline as well as the envelopes stay inside the specified channel polygon. For the case of the shortest path using the cost function in Eq. (47), Fig. 27(b) reveals that the computed B-spline curve is rendered shorter than the previous case.



(a) A smooth curve in the channel



(b) A shortest curve in the channel

Figure 27: Two optimization results.

3.4 Construct path templates for different channels

In this section we propose to construct path templates to be utilized for on-line path smoothing. The templates contain a set of planar B-spline curves which are then used for local path segments to avoid obstacles. An obstacle-free discrete path sequence is provided by a high-level path planner [59], which constructs a channel as the obstacle-free region delimited by polygonal lines. Taking into consideration all possible combinations of path sequences, we solve a number of path optimization problems subject to different channel constraints, computing a set of quartic B-splines. The path templates are used for on-line path smoothing in conjunction with the high-level path planning algorithm in order to generate a smooth path which avoids obstacles.

3.4.1 Path rules within a finite horizon

Suppose a world environment $\mathcal{W} \subset \mathbb{R}^2$ is decomposed into a uniform cell decomposition which consists of square cells $c_{k,\ell}$ of dimension $1/2^J \times 1/2^J$ at resolution level J. We adopt the four-connectivity between cells, hence the high-level path planner computes an optimal path as a sequence of cells from the current cell to the goal cell. The path sequence is written in a *path word* by which the transitions toward the North, South, East, and West directions between two cells can be encoded by `N`, `S`, `E`, and `W`, respectively. We specify the range of interest within four-cell horizon from the current cell, as shown in Fig. 28. If the goal cell is located outside the horizon, then an optimal path sequence is computed in a such manner that the path eventually passes through one of the cells at the horizon boundary. Let a local path sequence from the current cell to reach the boundary cells be a *local path instance*. If a path is supposed to visit each cell only once, the number of possible combinations for local path instance turns out to be finite. In addition, taking advantage of the symmetry about the x-axis (East direction) and y-axis (North direction), we investigate the local path instances restricted on one quadrant of the 7×7 cell grid.

Without loss of generality, we consider the local path instances on the first quadrant, as shown in Fig. 29. From the square cell geometry, it follows that we can also take advantage of the symmetry about the diagonal axis. Hence, as shown in Fig. 29,

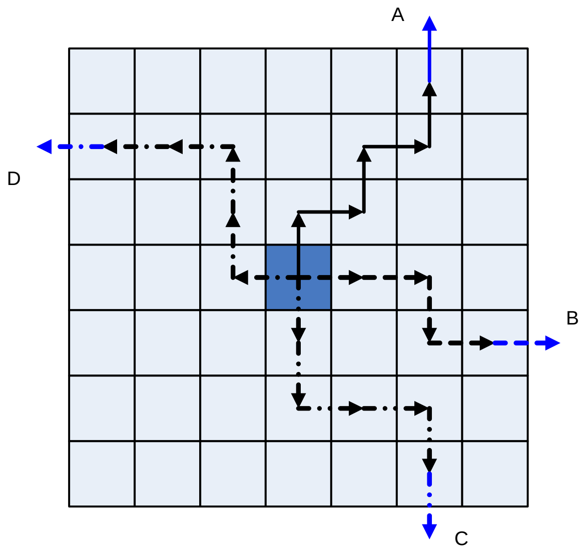


Figure 28: Examples of path sequences starting from the current cell at the center. We adopt the four-connectivity between cells. The goal cell is supposed to be located beyond the horizon. Possible path sequences are given as examples of the optimal path. The path A is written by `NENEN`$\cdots$, the path B is `EESE`$\cdots$, the path C is `SSEES`$\cdots$, and the path D is `WNNWW`$\cdots$.

two path instances of `ENNEN` and `NEENE` are similar to each other so that they are considered to be derived from the same local path instance.

Subsequently, we only investigate the local path instances from the current cell to one of the top boundary cells $c_{0,3}$, $c_{1,3}$, $c_{2,3}$, and $c_{3,3}$ shown in Fig. 29. Any local path instances can then be derived from these path instances by applying the symmetric operations along the horizontal, vertical, diagonal axes. Among all possible combinations of path sequences to reach the top boundary cells, we describe the necessary path rules to determine a unique local path instance, which are enlisted as follows,

1. *(Terminal conditions)* Suppose a local path instance is restricted inside the first quadrant, that is, it never goes outside the horizon before it reaches a terminal cell on the top boundary. The terminal cell should be one of the top boundary cells, except the cell $c_{3,3}$. The reason for this is attributed to the four-connectivity between cells; The path instance that reaches $c_{3,3}$ necessarily

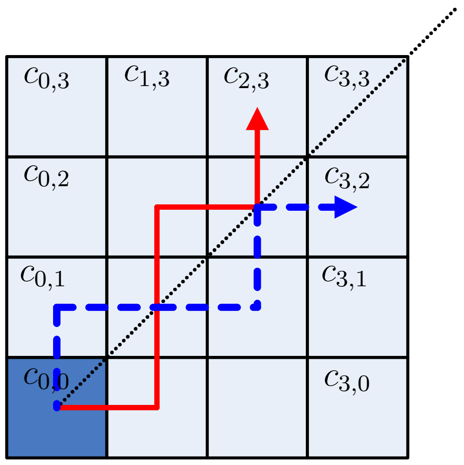


Figure 29: Local path instances on the first quadrant. From the additional symmetry about the diagonal axis, it is possible to transform the path instance drawn in dashed line (`NEENE`) to the path instance drawn in solid line (`ENNEN`). Path rules are given in order to determine unique path instances reaching the cell at the top boundary.

passes the adjacent boundary cell either $c_{2,3}$ or $c_{3,2}$, thus we regard the path instances to the cell $c_{3,3}$ as a subset of the path instances to $c_{2,3}$ and exclude from the consideration. In order to come up with a path sequence that reaches the terminal cell, the corresponding path word should have certain numbers of occurrence of `N`, `S`, `E`, and `W` satisfying the following conditions,

$$\sum(\#\mathtt{N} - \#\mathtt{S}) = 3,$$

$$\sum(\#\mathtt{E} - \#\mathtt{W}) = \begin{cases} 0 & \text{for } c_{0,3}, \\ 1 & \text{for } c_{1,3}, \\ 2 & \text{for } c_{2,3}. \end{cases}$$

2. *(Self-avoiding path)* The path must visit each cell exactly once, and never intersect itself. From this rule, we explicitly prevent a pathological case such as a cyclic loop from being considered in the templates. This type of path can be described by a self-avoiding walk [89] on a 4×4 cell grid. The total number of self-avoiding walks on an $m \times n$ grid, which starts from a corner and ends at the opposite corner by only horizontal and vertical steps, is computed using

Table 3: Number of self-avoiding walks on an $m \times n$ grid

m, n	2	3	4	5
2	2	4	8	16
3	4	12	38	125
4	8	38	184	976
5	16	125	976	8512

a recurrence relation [42]. Table 3 gives the first few numbers of such walks for small m and n. Similarly, the number of candidate of self-avoiding paths from the current cell $c_{0,0}$ to the top boundary cells is calculated from Table 3 utilizing recurrence relationship. Among these candidates, only certain number of self-avoiding path will be considered in the path templates.

3. *(Path optimality)* The high-level path planning algorithm is supposed to provide an optimal path sequence. The optimal path sequence is calculated as such that it minimizes the accumulated transition cost from the current node to the goal node. Typically, an directed edge cost is assigned to each transition of `N`, `S`, `E`, `W`, taking into account the cost associated with the cells as follows,

$$\mathcal{J}(u,v) = f(v) + \alpha g(u,v), \tag{49}$$

where, f is a positive obstacle cost associated with the target cell v, g is (Euclidean) distance cost between u and v, and $\alpha \geq 0$ is a weight constant. Consider now the path word `ENW` which represents the transition among four cells u, v, w, and z in order. The accumulated cost for this transition is computed by

$$\mathcal{J}(u,v) + \mathcal{J}(v,w) + \mathcal{J}(w,z) > \mathcal{J}(u,z), \tag{50}$$

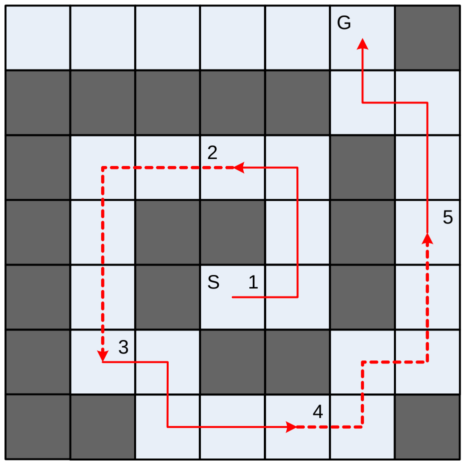
which turns out to be greater than the direct transition cost from u to z by a single path sequence `N`. It follows that the transition `ENW` is not an optimal, and neither of `ESW`, `NES`, `NWS`, etc. Consequently, we disregard any non-optimal path sequence when investigating the candidates of self-avoiding paths.

When establishing the previous path rules, we assumed that the local path instance necessarily ends up the top boundary cells, without escaping the quadrant. In certain cases, however, the path sequence might be rather provided in such way that it comes to cross the quadrants. In other words, the path sequence which starts from the current cell at the center of grid, is comprised of cells in more than two quadrants. If this is the case, we can infer that the path sequence will finally exit the finite horizon after passing through at least two quadrants, which makes it difficult to take advantage of the symmetry of the templates. In particular, the number of templates will increase, if one wants to consider all possibilities of inter-quadrant transitions, thus losing the benefit of using templates. In order to retain the symmetry of templates, we append additional cells between the quadrants to be considered as terminal cells other than the top boundary cells. Applying the path rules to a 7×7 cell grid (thus 4×4 cell grid for the first quadrant), only the cell $c_{0,2}$ (See Fig. 29) can be considered as an additional terminal cell. The other cells on the axis can not be terminal cells since the local path instances reaching them would conflict with the path rules. Then any possible local path instances starting from the center cell to $c_{0,2}$, certainly satisfying the path rules, are appended to the path templates.

The path templates for local path instances on the first quadrant are summarized in Table 4. Figure 30 shows an example of utilizing the path templates on a given path sequence. The starting cell is located in the middle area, where the path sequence is computed avoiding the shaded obstacle cells. In order to reach the goal cell, five local path instances are required as they are connected one another at one cell. Since each local path instance is written in path word, the corresponding path templates are incorporated with required symmetry operations, which are horizontal(H), vertical(V), diagonal(D) reflections, in order to adopt the templates to the actual path words.

Table 4: Path templates for local path instances on the first quadrant.

Destination cell	Path words			
$c_{0,3}$	NNN	ENNWN	EENNWWN	
$c_{1,3}$	NNEN	NENN	ENNN	EENNWN
$c_{2,3}$	NNEEN	NENEN	ENNEN	
	NEENN	ENENN	EENNN	
$c_{0,2}$	ENNW	EENNWW		



Path #	Path words	Template	Operations
1	ENNW	ENNW	-
2	WWSSS	EENNN	H, V
3	ESEE	NENN	H, D
4	ENENN	ENENN	-
5	NNWN	NNEN	V

Figure 30: Example incorporating the path templates on a complex path sequence. Five local path instances are are connected each other to reach the goal cell. The actual path words are equivalently recovered from the path templates with corresponding symmetry operations, which are horizontal(H), vertical(V), diagonal(D) reflections.

3.4.2 Construct B-spline path templates

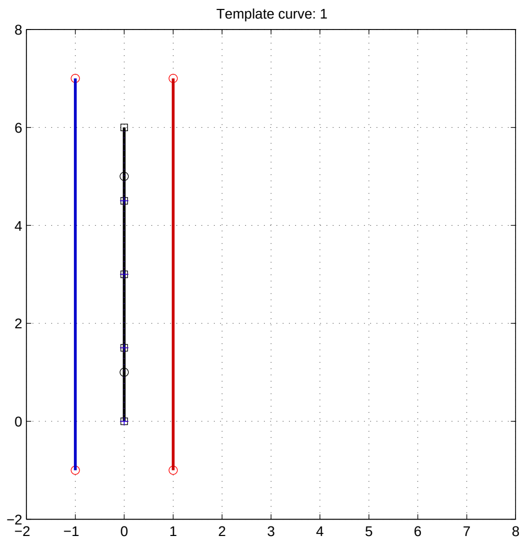
The B-spline path templates are composed of a set of B-spline curves which is supposed to be placed inside the channels. By assumption of the high resolution representation of $\mathcal{W}$, the cells of the optimal path sequence are regarded as a feasible region for the agent to fly safely. Hence, the shape of the channel is determined from the optimal path sequence by taking into account the square cell geometry, as the border lines around the cells of the local path instance are joined together to yield a channel polygon consisting of polylines (Left, Right) for the channel constraints. The boundary conditions of each B-spline curve are chosen, for the convenience sake, such that the B-spline curve starts from the center of the first cell of the local path instance, ends at the center of the last cell of the local path instance. The heading angles at each end of the curve is chosen such that the tangent vector at the point directs toward the center of the next adjacent cell, whereas the curvature values are set to be all zero. Hence, we manage to solve the optimization problem discussed in Section 3.3 for minimizing one of the cost in Eqs. (47) and (46), or combination of both. Figure 31 shows the results of optimization for path templates using B-splines corresponding each local path instance shown in Table 4.

3.5 On-line path smoothing algorithm

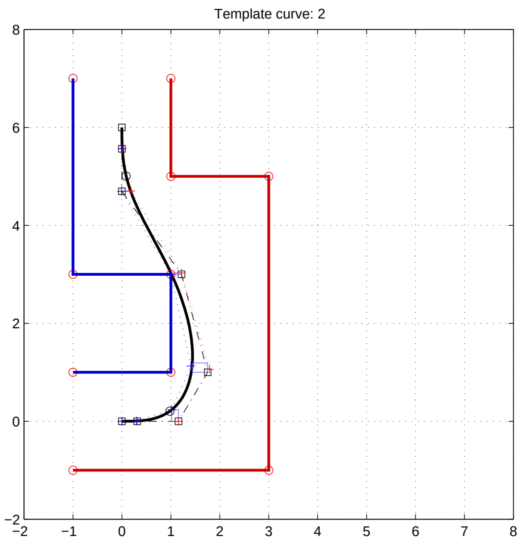
In this section, we present an on-line path smoothing algorithm incorporating the B-spline path templates. Given a discrete path sequence from a high-level path planner, each instance of the B-spline path templates becomes a part of the smooth path. Hence, the on-line path smoothing algorithm connects these path segments resulting in the smooth path over the entire section.

3.5.1 Stitching the path segments

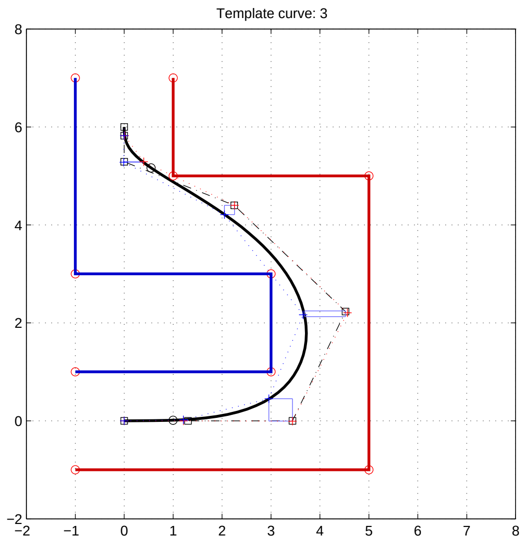
Along with the earlier discussion, two B-spline curves which are chosen from the path templates meet each other at one junction point. Due to the different boundary conditions with respect to tangent direction at each end of the curves, it follows that a heading angle discontinuity occurs at the junction point. Hence, in order to get a smooth path segment over two consecutive B-spline curves, we should stitch them with a transient B-spline curve, hence preserving the continuity property over distinct B-spline curves. To this end, we let $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ be the points on the leading and the following B-spline curves, respectively, other than the end points. Suppose the transient curve intersects these points at its own end points, then it follows that preserving continuity condition over distinct B-spline curves is assured by imposing the continuity conditions at these points. Hence, the transient curve, which is supposed



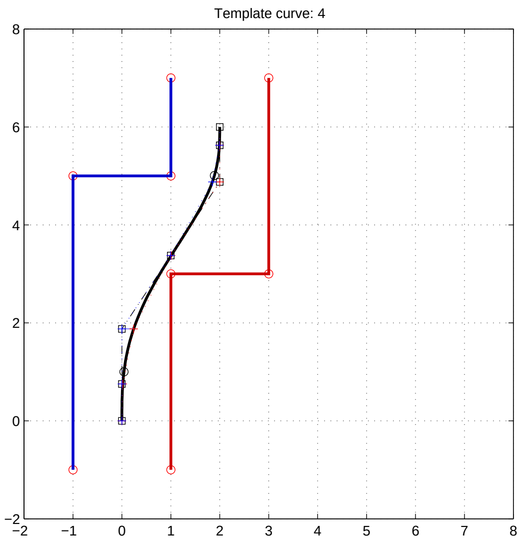
(a) `NNN`



(b) `ENNWN`

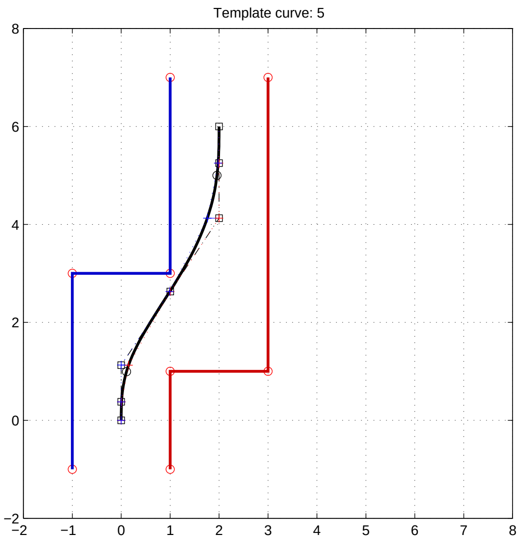


(c) `EENNWWN`

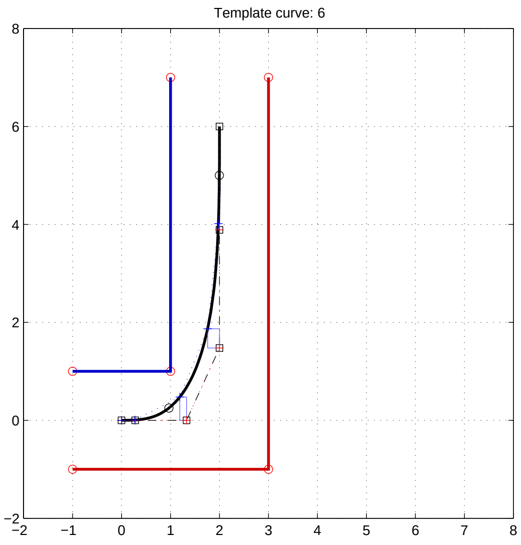


(d) `NNEN`

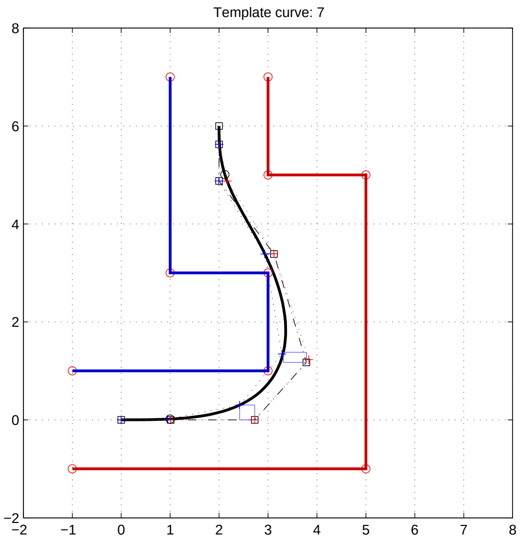
Figure 31: Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.



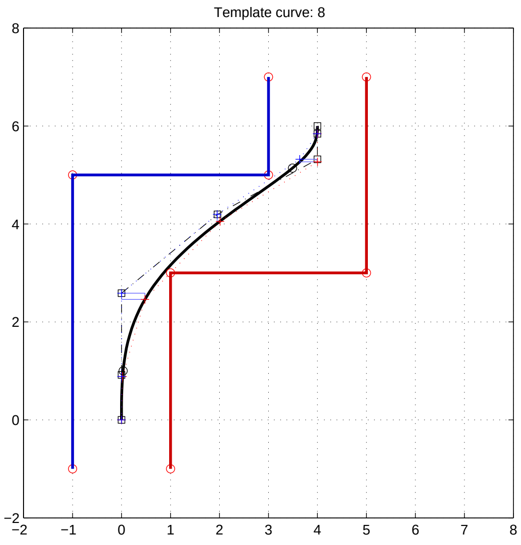
(e) NENN



(f) ENNN

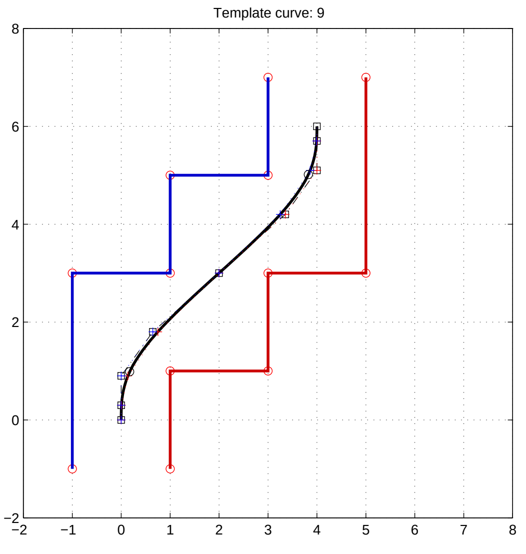


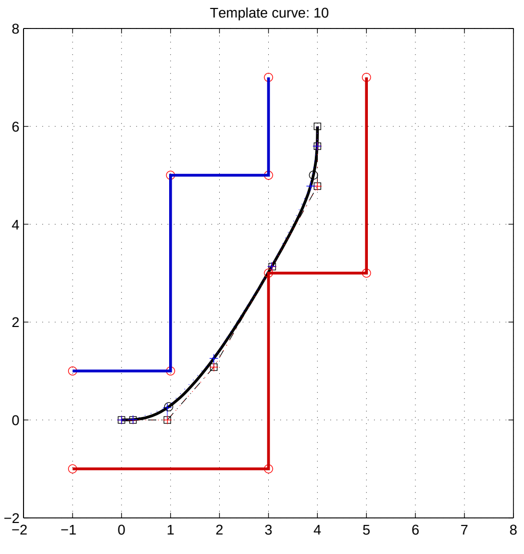
(g) EENNWN

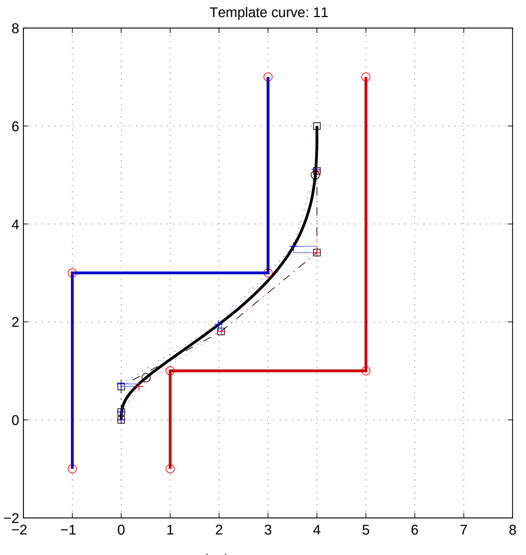


(h) NNEEN

Figure 31: (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.


(i) NENEN


(j) ENNEN


(k) NEENN

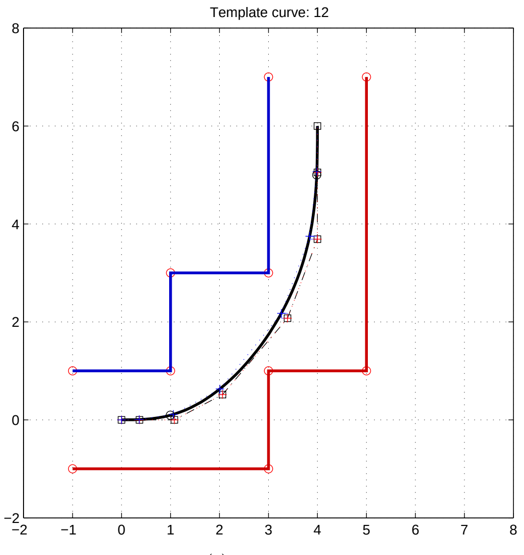
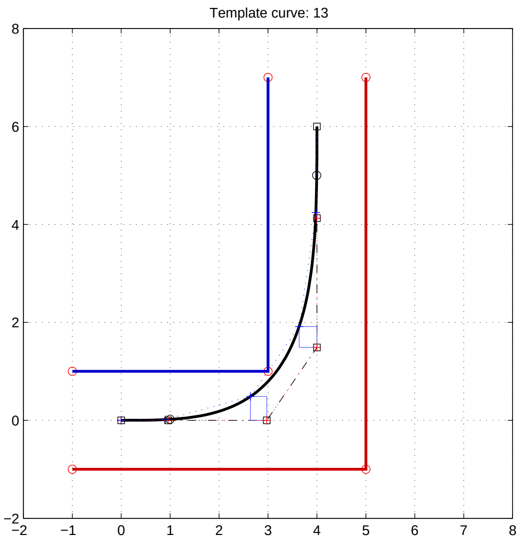
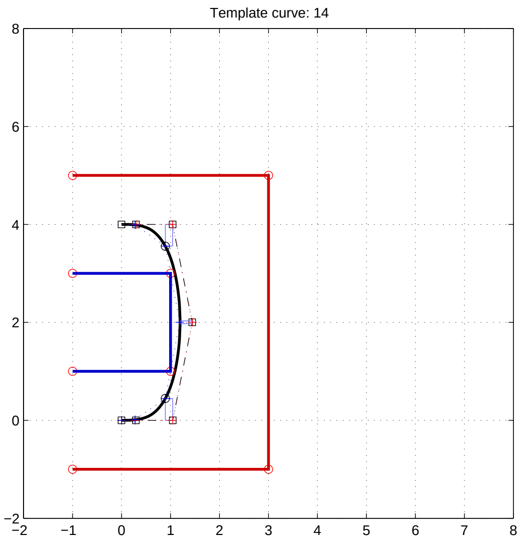

(l) ENENN

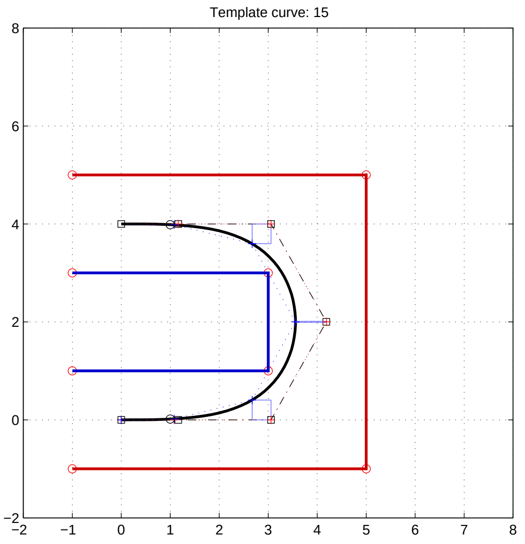
Figure 31: (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.



(m) EENNN



(n) ENNW



(o) EENNWW

Figure 31: (Continued) Path templates results from the path optimization using B-spline curves. Each plot corresponds to the local path instance in Table 4.

to stitch two distinct B-spline curves, satisfies the continuity conditions not only for position but also for heading angle and curvature value at each end.

The minimum order of the B-spline curve that satisfies the continuity conditions described above is four. Subsequently, we employ a transient cubic B-spline curve which is defined on a fixed clamped knot vector, $u = [0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1]$. Hence, the unknown parameters for determining this transient B-spline curve are six control points, among which the first three control points are related to the boundary conditions associated with the leading B-spline curve, and the rest are related to the boundary conditions associated with the following B-spline curve. Suppose the boundary conditions at the intersection point with the leading B-spline curve are given as follows. The intersection point is specified by $\boldsymbol{p}_a = [x_a, y_a]$, the heading angle at $\boldsymbol{p}_a$ is given ψ_a, and the curvature value is given κ_a. Let the control points be given $\boldsymbol{p}_i = [x_i, y_i]$, $i = 0, \cdots, 5$, on which we impose the following boundary conditions,

$$x_a = x_0, \tag{51a}$$

$$y_a = y_0, \tag{51b}$$

$$\psi_a = \tan^{-1}\left(\frac{y_0'}{x_0'}\right). \tag{51c}$$

where, x_0' and y_0' are the derivative with respect to the knot parameter at the control point $\boldsymbol{p}_0$. Note that, with the adopted clamped knot vector above, the cubic B-spline basis functions evaluated at the knot $u = 0$ are computed as follows,

	b_0	b_1	b_2	b_3	$\cdots$
$N_i^3(0)$	1	0	0	0	
$N_i^{3\prime}(0)$	-9	9	0	0	$\cdots$
$N_i^{3\prime\prime}(0)$	54	-81	27	0	

(52)

Hence, the first derivative at $\boldsymbol{p}_0$ is evaluated with the knot parameter $u = 0$ as follows,

$$x_0' = -9x_0 + 9x_1, \tag{53a}$$

$$y_0' = -9y_0 + 9y_1. \tag{53b}$$

From Eqs. (51c) and (53), it follows that

$$x_1 = x_a + R_a \cos\psi_a, \tag{54a}$$

$$y_1 = y_a + R_a \sin\psi_a, \tag{54b}$$

where R_a is the distance between $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ as a design parameter. In addition, we impose the curvature boundary condition as follows

$$\kappa_a = \frac{x_0' y_0'' - y_0' x_0''}{\left(x_0'^2 + y_0'^2\right)^{3/2}}, \tag{55}$$

where, x_0'' and y_0'' are the second derivative with respect to the knot parameter at $\boldsymbol{p}_0$. Using Eqs. (52) and (53) after algebraic manipulation results in the following expression regarding x_2 and y_2,

$$-x_2 \sin\psi_a + y_2 \cos\psi_a = 3R_a^2 \kappa_a + y_a \cos\psi_a - x_a \sin\psi_a. \tag{56}$$

In order to solve for x_2 and y_2, we adopt an additional equation in terms of x_2 and y_2 which determines a unique solution of x_2 and y_2 as follows. Suppose $\boldsymbol{p}_2^{\perp}$ is the point at the distance $2R_a$ along the line $\overline{\boldsymbol{p}_0\boldsymbol{p}_1}$ from $\boldsymbol{p}_0$ (See Fig. 32). The control point $\boldsymbol{p}_2$ can then be chosen uniquely by imposing the additional condition such that the projection of $\boldsymbol{p}_2$ onto the line $\overline{\boldsymbol{p}_0\boldsymbol{p}_1}$ ends up with the design point $\boldsymbol{p}_2^{\perp}$. It follows that

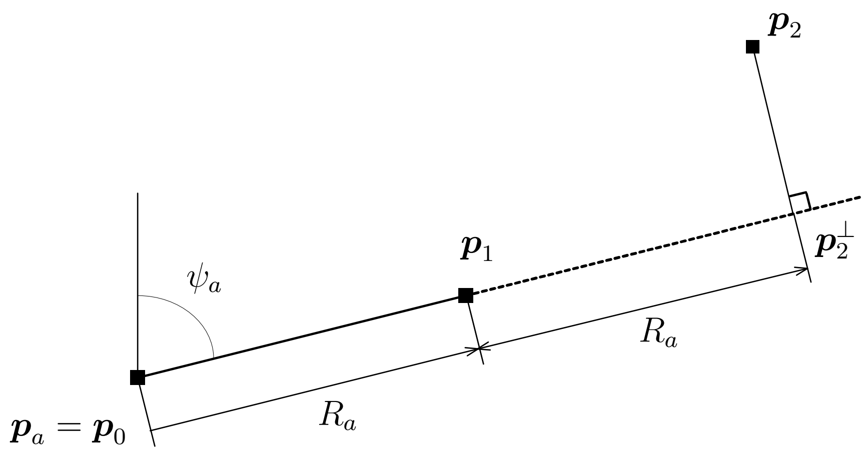

$\boldsymbol{p}_2$
$\boldsymbol{p}_1$
ψ_a
$\boldsymbol{p}_2^{\perp}$
R_a
R_a
$\boldsymbol{p}_a = \boldsymbol{p}_0$

Figure 32: Determine the unique $\boldsymbol{p}_2$ in terms of $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ in conjunction with the design parameter R_a.

with extra algebraic manipulation, we obtain the additional equation as follow,

$$x_2 \cos\psi_a + y_2 \sin\psi_a = 2R + x_a \cos\psi_a + y_a \psi_a. \tag{57}$$

In a similar manner, we impose the boundary conditions at the end of the transient B-spline as follows. With the cubic B-spline basis functions evaluated at the knot $u = 1$,

$$\begin{array}{c|ccccc} & \cdots & x_2 & x_3 & x_4 & x_5 \\ \hline N_i^3(1) & & 0 & 0 & 0 & 1 \\ N_i^{3\prime}(1) & \cdots & 0 & 0 & -9 & 9 \\ N_i^{3\prime\prime}(1) & & 0 & 27 & -81 & 54 \end{array} \tag{58}$$

and the given boundary conditions such that the intersection point is specified by $\boldsymbol{p}_b = [x_b, y_b]$, the heading angle at $\boldsymbol{p}_b$ is given ψ_b, and the curvature value is given κ_b, we obtain the following formulas to determine the three control points,

$$x_5 = x_b, \tag{59a}$$

$$y_5 = y_b, \tag{59b}$$

$$x_4 = x_b - R_b \cos\psi_b, \tag{59c}$$

$$y_1 = y_b - R_b \sin\psi_b, \tag{59d}$$

$$-x_3 \sin\psi_b + y_3 \cos\psi_b = 3R_b^2\kappa_b + y_b \cos\psi_b - x_b \sin\psi_b, \tag{59e}$$

$$x_3 \cos\psi_b + y_3 \sin\psi_b = -2R_b + x_b \cos\psi_b + y_b \sin\psi_b. \tag{59f}$$

Figure 33 shows an example of stitching the two B-spline curve derived from the path templates by a transient cubic B-spline.

3.5.2 Simulation results of the on-line path smoothing algorithm

In this section we present simulation results of the on-line path smoothing in conjunction with the $\mathcal{D}^*$-lite path planning algorithm. It is assumed that the agent navigates over the unknown environment, while updating the map with the information gathered from a proximity sensor. The world data is given by 256× 256 pixels. We adopt

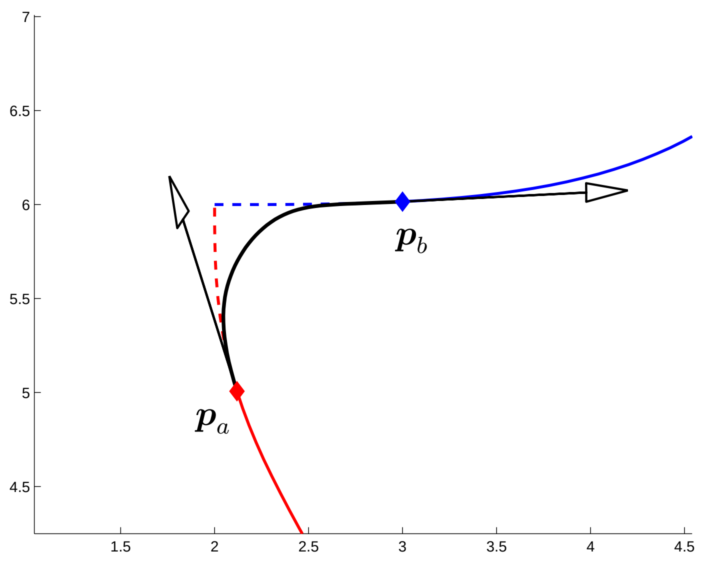


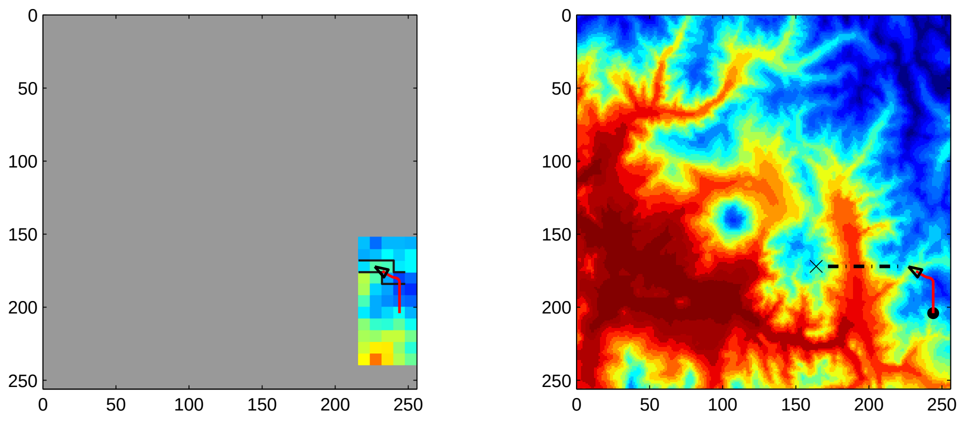
Figure 33: Example of stitching the two B-spline curve with a cubic B-spline curve

a uniform cell decomposition of cell size 8×8 pixels at the level $J = 5$. The range of the proximity sensor is chosen to be $r_5 = 28$, thus resulting in the high resolution window by a 7×7 square cell grids.

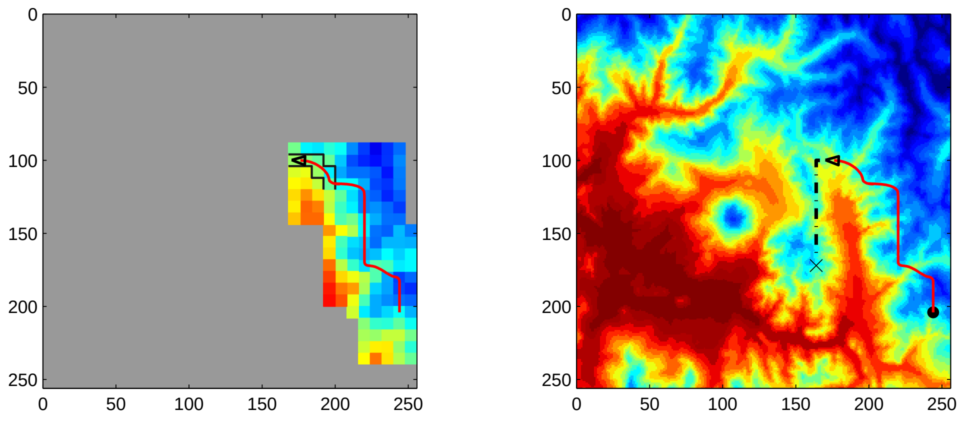
The results from the on-line path smoothing algorithm with the $\mathcal{D}^*$-lite path planning algorithm are shown in Fig. 34. Specifically, Fig. 34 shows the evolution of the path at different time steps as the agent moves to the final destination. At each step, the best proposed path is drawn by a dashed-dot line and the actual path followed by the agent is drawn by a solid line. In each step a channel is drawn by thin polylines that correspond to the discrete path sequence. The actual path is constructed by joining the smooth path segments derived from the B-spline path templates. The $\mathcal{D}^*$-lite algorithm updates occur whenever the agent approaches closed to the end of each path segment, giving a (possibly new) path sequence. Subsequently, the previous curve and the newly derived B-spline curve are stitched with a transient B-spline curve. This process is repeated until the UAV reaches the final destination, as shown in Fig. 34(c)

3.6 Summary

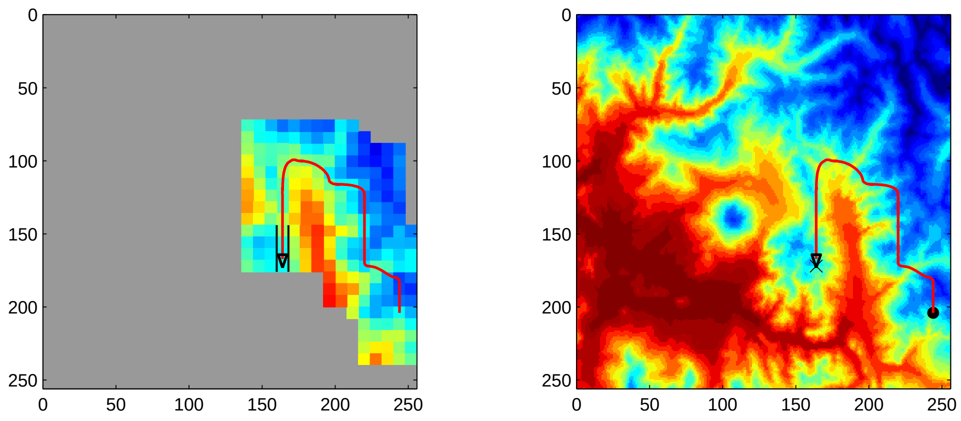
In this chapter, we presented the on-line path smoothing algorithm incorporating the path templates for generating a smooth path derived from the high-level path planner. The path templates are comprised of a set of B-spline curves, which have been obtained from the off-line optimization in the manner that each path instance stays inside the prescribed channel, hence the path efficiently avoids obstacles outside channels. In conjunction with the high-level path planning algorithm, the on-line implementation of the proposed algorithm involves finding the corresponding path segments and stitching them together while preserving the continuity of the curve. The simulation results with the $\mathcal{D}^*$-lite path planning algorithm validates the effectiveness of the proposed algorithm, having minimal on-line computational cost to get a smooth path for UAV to fly along.



(a) $t = t_{14}$



(b) $t = t_{30}$



(c) $t = t_f$

Figure 34: On-line path smoothing in conjunction with replanning using $\mathcal{D}^*$-lite algorithm. Dashed-dot lines represent the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the high resolution area. Actual path followed by the agent is derived from the B-spline path templates, which are represented by solid lines.

CHAPTER IV

PATH FOLLOWING CONTROL USING BACKSTEPPING AND PARAMETER ADAPTATION

4.1 Introduction

Modern UAVs are envisioned to replace human pilots in various missions. In order to accomplish these missions with minimal human intervention, the operation of a UAV should be fully automated from the top level of path planning, down to the inner control loop level. At the top level of the control hierarchy, the path planning algorithm computes a rough approximation of the optimal path toward the goal. A path generation algorithm then smooths the path yielding a dynamically feasible, flyable path, by taking into account the kinematic constraints of the UAV. Finally, the path following algorithm is responsible for guiding the UAV to stay close to the designed path.

Various control approaches in the literature have been proposed to address the path following problem: Niculescu [102] introduced a lateral track control law, Park et al. [106] proposed a simple, yet effective, nonlinear control logic and demonstrated it experimentally, Ren and Beard [116] considered the problem of constrained trajectory tracking, Nelson et al. [100] proposed a path following control using vector fields to guide the UAV on the desired path, and Rysdyk [120] proposed a guidance law based on the ‘good helmsman’ behavior.

Motivated by the method proposed in Ref. [96], kinematic control laws have been used to regulate the distance error from the reference path for unicycle-type mobile robots [75, 76]. The key aspect of the proposed algorithms in Refs. [75, 76] is that the control laws explicitly incorporate the controlled motion of a ‘virtual target’ to be tracked along the path. In this chapter, we present a nonlinear path following algorithm, which is an extension of the one by [75] for UAV path following control. We apply a standard backstepping technique to compute the roll angle command from the heading rate command of the kinematic control law. A parameter adaptation technique is then proposed to compensate for the inaccurate time constant of the roll closed loop, yielding robust performance during controller implementation. The path following control algorithm is validated through a high-fidelity hardware-in-the-loop simulation (HILS) environment to show the applicability of the presented algorithm on the actual system.

4.2 Problem Description

A fixed-wing UAV is equipped with a low-level autopilot with on-board sensors that provides feedback control for attitude angles, air speed, and altitude. In a typical search mission, the air speed and the altitude are kept constant, so that the UAV stays inside the safe flying envelope. Suppose that the inertial speed (V) and the course angle (χ; inertial speed heading) are directly measured using an on-board GPS sensor. A simplified kinematic model in the two dimensional plane is given as follows,

$$\begin{aligned} \dot{x} &= V\cos\chi, \\ \dot{y} &= V\sin\chi, \\ \dot{\chi} &= \omega, \end{aligned} \tag{60}$$

where (x, y) denotes the inertial position, and ω is the heading rate of the UAV. By expressing the equations of motion in terms of the ground speed and the course angle, the effect of the wind velocity on the dynamics is removed. Furthermore, by using inertial measurements for path planning control, wind disturbance is naturally rejected so that the performance of path following controller can be improved significantly. For a fixed-wing UAV at a banked-turn maneuver with no sideslip, and assuming

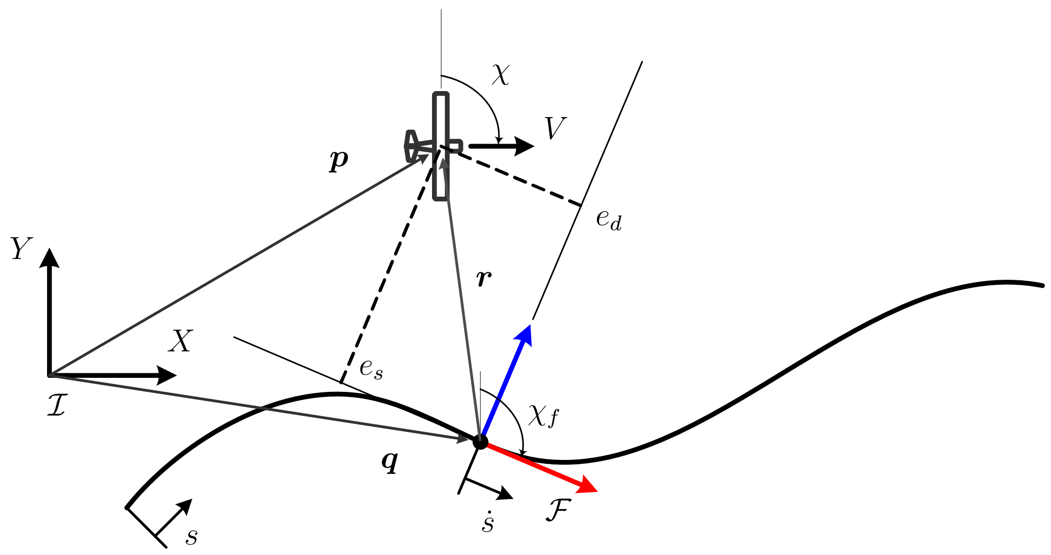


Figure 35: Definition of the Serret-Frenet frame for the path following problem.

that $|\phi| < \phi_{\max} < \pi/2$, the heading rate ω is induced by the roll angle ϕ with the inertial speed V as follows,

$$\omega = \frac{g}{V} \tan\phi, \tag{61}$$

where the roll angle is assumed to be controlled by an inner PID loop. Properly tuned PID gains result in closed-loop roll dynamics which resemble a first-order system,

$$\dot{\phi} = \frac{1}{\lambda_\phi}(\phi_c - \phi), \tag{62}$$

where ϕ_c is the roll angle command and $\lambda_\phi > 0$ is the time constant.

Consider now a UAV flying along a planar path as shown in Fig. 35. Denote the inertial position of the UAV by $\boldsymbol{p} = [x\ y]^\mathsf{T} \in \mathbb{R}^2$. The geometric path is defined in terms of the arc-length parameter s. For any given s, the inertial position of the point on the path associated with s is denoted by $\boldsymbol{q}(s) \in \mathbb{R}^2$, at which the Serret-Frenet frame is attached, and moves along the path with speed $\dot{s}$. The x-axis of the Serret-Frenet frame is aligned with the tangent vector to the path at $\boldsymbol{q}(s)$ and has an angle $\chi_f(s)$ with respect to the inertial frame $\mathcal{I}$. Let the error vector $\boldsymbol{e}$ in the Serret-Frenet frame be decomposed in the along-track error e_s and the cross-track error e_d. Then the inertial error vector $\boldsymbol{r} = \boldsymbol{p} - \boldsymbol{q}(s)$ expressed in the Serret-Frenet frame is obtained by

$$\boldsymbol{e} = \mathbf{R}^{\mathsf{T}}(\chi_f)\big(\boldsymbol{p} - \boldsymbol{q}(s)\big), \tag{63}$$

where,

$$\mathbf{R}(\chi_f) = \begin{bmatrix} \cos\chi_f & -\sin\chi_f \\ \sin\chi_f & \cos\chi_f \end{bmatrix}, \tag{64}$$

is the rotation matrix from the Serret-Frenet frame to the inertial frame.

By differentiating Eq. (63) with respect to time, it follows that

$$\dot{\boldsymbol{e}} = \dot{\mathbf{R}}^{\mathsf{T}}(\chi_f)\big(\boldsymbol{p} - \boldsymbol{q}(s)\big) + \mathbf{R}^{\mathsf{T}}(\chi_f)\big(\dot{\boldsymbol{p}} - \dot{\boldsymbol{q}}(s)\big), \tag{65}$$

where,

$$\dot{\mathbf{R}}(\chi_f) = \mathbf{R}(\chi_f)\mathbf{S}(\dot{\chi}_f), \tag{66}$$

where, $\mathbf{S}(\dot{\chi}_f)$ is the skew-symmetric matrix,

$$\mathbf{S}(\dot{\chi}_f) = \begin{bmatrix} 0 & -\dot{\chi}_f \\ \dot{\chi}_f & 0 \end{bmatrix}. \tag{67}$$

It follows from Eq. (60) that

$$\dot{\boldsymbol{p}} = \mathbf{R}(\chi) \begin{bmatrix} V \\ 0 \end{bmatrix}. \tag{68}$$

Notice that $\dot{\boldsymbol{q}}$ is the time derivative of the point $\boldsymbol{q}(s)$, whose speed is represented by $\dot{s}$ when expressed in the Serret-Frenet frame. It follows that

$$\dot{\boldsymbol{q}} = \mathbf{R}(\chi_f) \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix}. \tag{69}$$

Subsequently, by substituting Eqs. (68) and (69) in Eq. (65), we obtain,

$$\dot{\boldsymbol{e}} = -\mathbf{S}(\dot{\chi}_f)\boldsymbol{e} + \mathbf{R}(\chi - \chi_f) \begin{bmatrix} V \\ 0 \end{bmatrix} - \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix}, \tag{70}$$

where $\mathbf{R}^{\mathsf{T}}(\chi_f)\mathbf{R}(\chi) = \mathbf{R}(\chi - \chi_f)$.

Let now the error course angle $\widetilde{\chi}$ be defined by

$$\widetilde{\chi} \triangleq \chi - \chi_f. \tag{71}$$

Hence, we obtain the time derivative of $\widetilde{\chi}$,

$$\dot{\widetilde{\chi}} = \dot{\chi} - \dot{\chi}_f = \omega - \dot{\chi}_f, \tag{72}$$

where

$$\dot{\chi}_f \triangleq \frac{\mathrm{d}\chi_f}{\mathrm{d}t} = \frac{\mathrm{d}\chi_f}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}t} = \kappa(s)\dot{s}, \tag{73}$$

and $\kappa(s)$ is the curvature of the path at $\boldsymbol{q}(s)$.

The error kinematic model of a fixed-wing UAV for the path following problem with respect to the Serret-Frenet frame is summarized as follows,

$$\begin{aligned}
\dot{e}_s &= V\cos\widetilde{\chi} - (1 - \kappa(s)e_d)\dot{s}, && \text{(74a)}\\
\dot{e}_d &= V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}, && \text{(74b)}\\
\dot{\widetilde{\chi}} &= \omega - \kappa(s)\dot{s}. && \text{(74c)}
\end{aligned}$$

From Eqs. (74), the path following problem reduces into a problem of driving the errors to zero as the UAV approaches the given path. In Ref. [96], the point $\boldsymbol{q}$ is found by projecting $\boldsymbol{p}$ on the path, assuming the projection is well defined. Hence, the control law derived in [96] requires a stringent restriction on the initial position of $\boldsymbol{q}$ in order to avoid singularities. In contrast, Lapierre and Soetanto[75] proposed to employ a moving Serret-Frenet frame along the path, which effectively provides an extra control parameter $\dot{s}$ allowing $\boldsymbol{q}(s)$ to evolve along the error states. This control parameter then mitigates the stringent restriction on the initial condition arose in [96].

4.3 Path Following Controller Design

In this section we present a nonlinear path following control logic, which steers the UAV to the reference path with an inaccurately known time constant λ_ϕ. Beginning with the derivation of a kinematic control law for the heading rate command, we employ backstepping to derive a roll control command for a fixed-wing UAV, which, in turn, induces an equivalent control effort by the kinematic control law. In addition, we apply a parameter adaptation technique to deal with the inaccurately known time constant.

4.3.1 Kinematic controller design

Following a similar approach as in Ref. [75], we first derive a kinematic control law for the heading rate. We introduce a bounded differentiable function with respect to the cross-track error e_d, as follows

$$\delta(e_d) = -\chi^\infty \frac{e^{2ke_d} - 1}{e^{2ke_d} + 1} \tag{75}$$

where, $k > 0$ and $\chi^\infty \in (0, \pi/2)$[96]. This function is so-called the *approach angle*, since it provides the desired relative course transition of the UAV to the path as a function of the cross-track error e_d. When the cross-track error e_d is zero, the approach angle vanishes, thus imposing the condition that the course angle of the UAV must be tangential to the path. The positive constant k sets the effectiveness of the transient maneuver during approach. The approach angle provides a behavior similar to that of a 'good helmsman'[120] since it satisfies the following condition:

$$e_d \sin\big(\delta(e_d)\big) \leq 0, \qquad \text{for all } e_d. \tag{76}$$

This condition steers the UAV to the path along the correct direction (turn left when the UAV is on the right side of the path, and turn right in the opposite situation).

The kinematic controller for path following using the error kinematic model in Eq. (74) is given below.

Proposition 1 *Consider the kinematic error model of the UAV described in Eq. (74) and the approach angle $\delta(e_d)$ defined as in Eq. (75). Assume that the speed of the UAV is non-negative and suppose that the reference path is parameterized by the arc-length s, and that at each s the variables κ, e_s, e_d, and χ_f are well defined. Then, the following kinematic control law,*

$$\begin{aligned}
\omega &= -k_\omega\big(\widetilde{\chi} - \delta(e_d)\big) + \kappa(s)\dot{s} + \delta'(e_d)\big(V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}\big) \\
&\quad - \frac{e_d V}{\gamma}\left(\frac{\sin\widetilde{\chi} - \sin\big(\delta(e_d)\big)}{\widetilde{\chi} - \delta(e_d)}\right), && \text{(77a)} \\
\dot{s} &= k_s e_s + V\cos\widetilde{\chi}, && \text{(77b)}
\end{aligned}$$

where k_s, k_ω and γ are positive constants, asymptotically drives e_s, e_d, and $\widetilde{\chi}$ towards zero.

Proof Let V_1 be a positive definite and radially unbounded Lyapunov function[96],

$$V_1 = \frac{1}{2\gamma}\big(e_s^2 + e_d^2\big) + \frac{1}{2}\big(\widetilde{\chi} - \delta(e_d)\big)^2. \tag{78}$$

In the Lyapunov candidate function adopted, the first term captures the distance between the UAV and the path, and the second term intends to steer the error course angle $\widetilde{\chi}$ to the approach angle $\delta(e_d)$, which forces the UAV to follow a desired transition profile.

Differentiating V_1 with respect to time, we get

$$\begin{aligned}
\dot{V}_1 &= \frac{1}{\gamma}\big(e_s\dot{e}_s + e_d\dot{e}_d\big) + \big(\widetilde{\chi} - \delta(e_d)\big)\big(\dot{\widetilde{\chi}} - \dot{\delta}(e_d)\big) \\
&= \frac{1}{\gamma}\Big\{e_s\big(V\cos\widetilde{\chi} - (1-\kappa(s)e_d)\dot{s}\big) + e_d\big(V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}\big)\Big\} \\
&\quad + \big(\widetilde{\chi} - \delta(e_d)\big)\Big\{\omega - \kappa(s)\dot{s} - \delta'(e_d)\big(V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}\big)\Big\} \\
&= \frac{1}{\gamma}e_s(V\cos\widetilde{\chi} - \dot{s}) + \frac{1}{\gamma}e_d V\sin\big(\delta(e_d)\big) \\
&\quad + \big(\widetilde{\chi} - \delta(e_d)\big)\Big\{\omega - \kappa(s)\dot{s} - \delta'(e_d)\big(V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}\big) + \frac{e_d V}{\gamma}\frac{\sin\widetilde{\chi} - \sin\big(\delta(e_d)\big)}{\widetilde{\chi} - \delta(e_d)}\Big\}.
\end{aligned} \tag{79}$$

Using the control law for ω and $\dot{s}$ from Eqs. (77) yields,

$$\dot{V}_1 = -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 \leq 0. \tag{80}$$

Note that V_1 is radially unbounded, hence the set $\Omega_c = \{V_1(e_s, e_d, \widetilde{\chi}) \leq c\}$ is a compact, positively invariant set. Let E_1 be the set of all points in Ω_c where $\dot{V}_1 = 0$. The set E_1 is given by $E_1 = \{(e_s, e_d, \widetilde{\chi}) \in \Omega_c | e_s = e_d = 0 \text{ and } \widetilde{\chi} = \delta\}$. Noticing that $\delta(\cdot)$ is a function of the cross-track error e_d, one can easily verify that any point starting from the set E_1 will remain in the set, or the set E_1 is an invariant set. Hence, by the LaSalle's invariance principle, we conclude that every trajectory starting in Ω_c approaches E_1 as $t \to \infty$. Therefore,

$$\lim_{t\to\infty} e_s = 0, \;\; \lim_{t\to\infty} e_d = 0, \;\text{ and }\; \lim_{t\to\infty} \widetilde{\chi} = \delta(e_d) \to 0, \tag{81}$$

since $e_d \to 0$ and $\delta(e_d) \to 0$. ∎

Remarks. The objective of the closed loop system is to steer the error states towards zero, so that the UAV follows the virtual target moving on the path. It should be noted that the approach angle plays an important role, in steering the UAV to the reference path by constructing a vector field with respect to a reference point on the path to guide the UAV to the path, as shown in Fig. 36. As mentioned earlier, the moving reference frame in conjunction with the extra control command $\dot{s}$ helps the errors converge to zero. In particular, the term $k_s e_s$ in Eq. (77b) ensures the convergence of e_s to zero, providing the momentum induced by the moving virtual target. Figure 36 shows the vector fields with respect to the corresponding reference points on the path, which takes into account the relative momentum induced by the virtual target (drawn by dashed arrows). During implementation, the new location of the virtual target on the path is propagated using the expression in Eq. (77b) via a numerical integration scheme, such as the forward Euler method. Hence, if the reference path is well defined in terms of the arc-length parameter s, then the

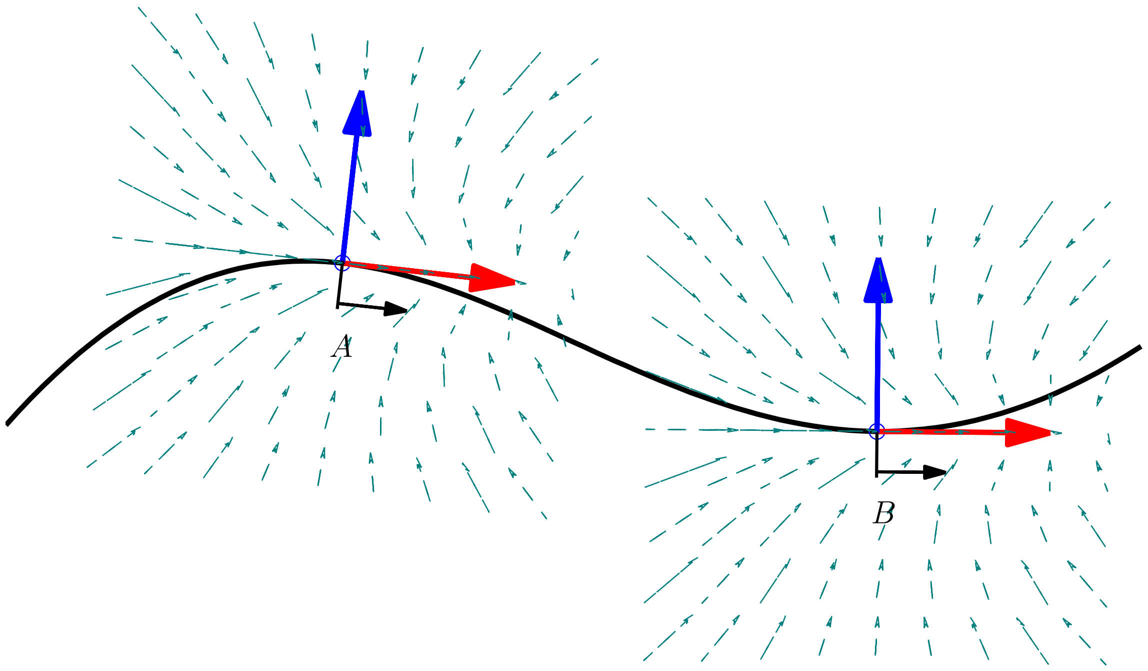


Figure 36: Local vector fields generated with respect to two distinct points on the path.

corresponding variables e_s, e_d, χ_f, and κ as also well defined, and thus the proposed path following algorithm is well-posed.

4.3.2 Roll angle command via backstepping

In this section, we apply backstepping to compute the roll control command from the heading rate command of the kinematic controller of the previous section.

For a fixed-wing UAV flying at a constant altitude, a roll angle command is often used for heading control. A simple way to compute the roll angle command directly from the heading rate command ω_d is using the kinematic relationship in Eq. (61),

$$\phi_d = \tan^{-1}\left(\frac{V\omega_d}{g}\right). \tag{82}$$

Note that the actual response of the roll angle differs from the desired value of Eq. (82), but it can be approximated by a first-order system described in Eq. (62). Taking into account the approximated model of Eq. (62), one can design a steering logic for the roll angle command ϕ_c in order to achieve $\omega \to \omega_d$ via $\phi \to \phi_d$. Following the standard backstepping technique [134], we introduce an auxiliary control input ν for the roll angle by augmenting the error model in Eq. (74) with $\dot{\phi} = \nu$. Thus, the system is given as follows,

$$\begin{aligned}
\dot{e}_s &= V\cos\widetilde{\chi} - (1-\kappa(s)e_d)\dot{s}, && \text{(83a)}\\
\dot{e}_d &= V\sin\widetilde{\chi} - \kappa(s)e_s\dot{s}, && \text{(83b)}\\
\dot{\widetilde{\chi}} &= \frac{g}{V}\tan\phi - \kappa(s)\dot{s}, && \text{(83c)}\\
\dot{\phi} &= \nu, && \text{(83d)}
\end{aligned}$$

where the auxiliary control input ν is associated with the roll command by

$$\nu = \frac{1}{\lambda_\phi}\big(\phi_c - \phi\big). \tag{84}$$

Suppose that the desired heading rate ω_d is obtained from Eq. (77a) and let $\omega_e \triangleq \omega - \omega_d = (g/V)\tan\phi - \omega_d$ be an error state for the heading rate, whose time derivative is given as follows

$$\dot{\omega}_e = \frac{g}{V}\dot{\phi}\sec^2\phi - \dot{\omega}_d. \tag{85}$$

Proposition 2 *Consider the kinematic error model of the UAV described in Eq. (83) and the approach angle $\delta(e_d)$ defined in Eq. (75). Assume that the speed of the UAV is non-negative and suppose that the reference path is parameterized by the arc-length s, and at each s the variables κ, e_s, e_d, and χ_f are well defined. Then, the control input,*

$$\nu = \frac{V}{g\sec^2\phi}\bigg(- k_e\omega_e - \widetilde{\chi} + \delta(e_d) + \dot{\omega}_d\bigg), \tag{86}$$

where k_e is a positive constant and ω_d is given in Eq. (77a), asymptotically drives e_s, e_d, and $\widetilde{\chi}$ towards zero, while $\omega \to \omega_d$.

Proof Let V_2 be an augmented Lyapunov candidate function given by,

$$V_2 = V_1 + \frac{1}{2}\omega_e^2. \tag{87}$$

where V_1 is given in Eq. (78). Differentiating with respect to time, one obtains

$$\begin{aligned}\dot{V}_2 &= \dot{V}_1 + \omega_e \dot{\omega}_e \\ &= \frac{1}{\gamma} e_s (V \cos \widetilde{\chi} - \dot{s}) + \frac{1}{\gamma} e_d V \sin\big(\delta(e_d)\big) \\ &\quad + \big(\widetilde{\chi} - \delta(e_d)\big)\bigg\{ \frac{g}{V}\tan\phi + \omega_d - \omega_d - \kappa(s)\dot{s} - \delta'(e_d)\big(V \sin\widetilde{\chi} - \kappa(s) e_s \dot{s}\big) \\ &\qquad + \frac{e_d V}{\gamma} \frac{\sin\widetilde{\chi} - \sin\big(\delta(e_d)\big)}{\widetilde{\chi} - \delta(e_d)} \bigg\} + \omega_e \bigg(\frac{g}{V}\dot{\phi}\sec^2\phi - \dot{\omega}_d \bigg).\end{aligned} \tag{88}$$

By the definition of ω_e, and substituting the desired ω_d from Eq. (77a) and $\dot{s}$ from Eq. (77b), the term inside of the bracket collapses to

$$\dot{V}_2 = -\frac{k_s}{\gamma} e_s^2 + \frac{e_d V}{\gamma} \sin\big(\delta(e_d)\big) - k_\omega \big(\widetilde{\chi} - \delta(e_d)\big)^2 + \omega_e \bigg(\frac{g}{V}\dot{\phi}\sec^2\phi + \widetilde{\chi} - \delta(e_d) - \dot{\omega}_d \bigg). \tag{89}$$

Choosing $\dot{\phi}$ through the auxiliary control input in Eq. (86), that is,

$$\dot{\phi} = \nu = \frac{V}{g \sec^2 \phi} \bigg(-k_e \omega_e - \widetilde{\chi} + \delta(e_d) + \dot{\omega}_d \bigg), \tag{90}$$

results in,

$$\dot{V}_2 = -\frac{k_s}{\gamma} e_s^2 + \frac{e_d V}{\gamma} \sin\big(\delta(e_d)\big) - k_\omega \big(\widetilde{\chi} - \delta(e_d)\big)^2 - k_e \omega_e^2 \leq 0. \tag{91}$$

The last inequality implies, in particular, that $e_s, e_d, \widetilde{\chi}$ and ω_e are bounded. Furthermore, $\delta(e_d)$ is also bounded from Eq. (75) since e_d is bounded. It is also easy to show, using Eq. (77a), that ω_d is bounded. To this end, notice that all signals in the rhs of Eq. (77a) except the last are bounded. Moreover, since the last term is arranged by

$$\frac{\sin\widetilde{\chi} - \sin\big(\delta(e_d)\big)}{\widetilde{\chi} - \big(\delta(e_d)\big)} = \operatorname{sinc}\bigg(\frac{1}{2}\big(\widetilde{\chi} - \delta(e_d)\big)\bigg) \cos\bigg(\frac{1}{2}\big(\widetilde{\chi} + \delta(e_d)\big)\bigg) \tag{92}$$

and the sinc function is bounded, the last term in the rhs of Eq. (77a) is also bounded. Since ω_d and ω_e are bounded, it follows that $|\phi| < \pi/2$ and the control Eq. (86) is well defined.

To complete the proof, note that V_2 is radially unbounded, hence the set $\Omega_d = \{V_2(e_s, e_d, \widetilde{\chi}, \omega_e) \leq c\}$ is a compact, positively invariant set. Let E_2 be the set of all points in Ω_d where $\dot{V}_2 = 0$. The set E_2 is given by $E_2 = \{e_s = e_d = 0,\ \widetilde{\chi} = \delta, \text{and } (g/V)\tan\phi = \omega_d\}$. Noticing that $\delta(\cdot)$ is a function of the cross-track error e_d, one can easily verify that any point starting from the set E_2 will remain in the set, i.e., E_2 is an invariant set. By the LaSalle's invariance principle, we have that every trajectory starting inside the set Ω_d approaches E_2 as $t \to \infty$. Therefore, $\lim_{t\to\infty} e_s = 0$, $\lim_{t\to\infty} e_d = 0$, and $\lim_{t\to\infty} \widetilde{\chi} = 0$ since $\delta(e_d) \to 0$. We also have $(g/V)\tan\phi \to \omega_d$ as $t \to \infty$. ∎

4.3.3 Parameter adaptation

Note that the actual roll angle command is computed from Eq. (62) and (86),

$$\phi_c = \lambda_\phi \nu + \phi, \tag{93}$$

where, λ_ϕ is the time constant of the system.

In practice, λ_ϕ is determined by the characteristics of the UAV airframe and PID gains, thus it is not known accurately. The actual roll command $\widehat{\phi}_c$ is then computed by

$$\widehat{\phi}_c = \widehat{\lambda}_\phi \nu + \phi, \tag{94}$$

where $\widehat{\lambda}_\phi$ is an estimate of λ_ϕ.

In order to compensate for any uncertainty in λ_ϕ, we apply a parameter adaptation technique. To this end, let V_3 be a candidate Lyapunov function defined by

$$V_3 \triangleq V_2 + \frac{1}{2}\frac{1}{k_a \lambda_\phi}\widetilde{\lambda}_\phi^2, \tag{95}$$

where V_2 is given in Eq. (87) and $\widetilde{\lambda}_\phi \triangleq \lambda_\phi - \widehat{\lambda}_\phi$ is the parameter estimate error. Differentiating with respect to time, we get

$$\begin{aligned}\dot{V}_3 &= \dot{V}_2 + \frac{1}{k_a\lambda_\phi}\widetilde{\lambda}_\phi \dot{\widetilde{\lambda}}_\phi \\ &= -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 \\ &\quad + \omega_e\bigg(\frac{g}{V}\dot{\phi}\sec^2\phi + \widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\bigg) + \frac{1}{k_a\lambda_\phi}\widetilde{\lambda}_\phi\dot{\widetilde{\lambda}}_\phi.\end{aligned} \tag{96}$$

The actual value of $\dot{\phi}$ is calculated from Eq. (62) in conjunction with the actual roll command in Eq. (94) and ν given in Eq. (86). Subsequently, substituting the actual $\dot{\phi}$ into Eq. (96), we get

$$\begin{aligned}\dot{V}_3 = &-\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 - \frac{\widehat{\lambda}_\phi}{\lambda_\phi}k_e\omega_e^2 \\ &+ \omega_e\frac{\widetilde{\lambda}_\phi}{\lambda_\phi}\big(\widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\big) + \frac{1}{k_a\lambda_\phi}\widetilde{\lambda}_\phi\dot{\widetilde{\lambda}}_\phi.\end{aligned} \tag{97}$$

Choose $\dot{\widetilde{\lambda}}_\phi$ as,

$$\dot{\widetilde{\lambda}}_\phi = -k_a\omega_e\big(\widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\big), \tag{98}$$

where k_a is a positive constant. It follows that

$$\dot{V}_3 = -\frac{k_s}{\gamma}e_s^2 + \frac{e_d V}{\gamma}\sin\big(\delta(e_d)\big) - k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)^2 - \frac{\widehat{\lambda}_\phi}{\lambda_\phi}k_e\omega_e^2 \leq 0. \tag{99}$$

Assuming λ_ϕ is constant, the parameter update law is readily obtained from Eq. (98) as follows,

$$\dot{\widehat{\lambda}}_\phi = k_a\omega_e\big(\widetilde{\chi} - \delta(e_d) - \dot{\omega}_d\big). \tag{100}$$

Proposition 3 *Let the control law in Eqs. (77) and (86). With the parameter update law given by Eq. (100) the actual roll command of Eq. (94) guarantees that the signals e_s, e_d, and $\widetilde{\chi}$ asymptotically tend to zero, while $(g/V)\tan\phi \rightarrow \omega_d$.*

Proof In order to prove the proposition notice that $\lim_{t\to\infty} V_3(t)$ exists since V_3 is bounded from below and is non-increasing. It therefore suffices to show that $\dot{V}_3$ is uniformly continuous, in which case we can invoke Barbalat's lemma to ensure that $\lim_{t\to\infty} \dot{V}_3(t) = 0$.

To this end, consider the expression of $\dot{V}_3$ in Eq. (99), from which it follows that e_d, e_s, $\widetilde{\chi}$, ω_e, $\widehat{\lambda}_\phi$, and $\widetilde{\lambda}_\phi$ are all bounded. As with the proof of Proposition 2, it follows from (77b) that $\dot{s}$ is bounded, and from (77a) that ω_d is bounded as well. The boundedness of ω_d and ω_e imply that $|\phi| < \pi/2$. Using (83) we conclude that $\dot{e}_s, \dot{e}_d, \dot{\widetilde{\chi}}$ are also bounded. Differentiating Eq. (77a) with respect to time, one can show that all terms are bounded, while the last term is also bounded since the derivative of the sinc function is bounded (see also Eq. (92)). Hence, it can be shown that $\dot{\omega}_d$ is bounded. Furthermore, $\delta(e_d)$ and its time derivative $\dot{\delta}(e_d) = \delta'(e_d)\dot{e}_d$ are bounded. A straightforward calculation shows that

$$\dot{\omega}_e = \frac{\widehat{\lambda}_\phi}{\lambda_\phi}\big(- k_e\omega_e - \widetilde{\chi} + \delta(e_d)\big) - \frac{\widetilde{\lambda}_\phi}{\lambda_\phi}\dot{\omega}_d. \tag{101}$$

From the previous equation it follows that $\dot{\omega}_e$ is bounded. Furthermore, $\dot{\widehat{\lambda}}_\phi$ is also bounded from (100). Consider now the expression for $\ddot{V}_3$, given by

$$\begin{aligned}\ddot{V}_3 = &-\frac{k_s}{\gamma}e_s\dot{e}_s + \frac{V\dot{e}_d}{\gamma}\sin\big(\delta(e_d)\big) + \frac{Ve_d}{\gamma}\dot{\delta}(e_d)\cos\big(\delta(e_d)\big) \\ &- 2k_\omega\big(\widetilde{\chi} - \delta(e_d)\big)\big(\dot{\widetilde{\chi}} - \dot{\delta}(e_d)\big) - \frac{\widehat{\lambda}_\phi}{\lambda_\phi}k_e\omega_e\dot{\omega}_e,\end{aligned} \tag{102}$$

where, all expressions in the rhs of the equation have been shown to be bounded. It follows that $\ddot{V}_3$ is bounded and hence $\dot{V}_3$ is uniformly continuous. Applying Barbalat's lemma, it follows that $\dot{V}_3 \to 0$ as $t \to 0$. ∎

4.4 Simulation results

This section illustrates the performance of the derived path-following control law. The reference path is given by a quartic B-spline over a non-decreasing knot parameter u which is monotonic to the arc-length s. Hence, we can compute $\boldsymbol{q}(s)$, $\chi_f(s)$, and $\kappa(s)$ in terms of the knot parameter as follows,

$$\chi_f(s) = \tan^{-1}\frac{y'}{x'}\,, \quad \kappa(s) = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}}. \tag{103}$$

where $(\cdot)'$ and $(\cdot)''$ are the derivatives with respect to u. The knot parameter is propagated along with Eq. (77b) as follows,

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\mathrm{d}s}{\mathrm{d}t}\Big/\frac{\mathrm{d}s}{\mathrm{d}u} = \dot{s}/\sqrt{x'^2 + y'^2}. \tag{104}$$

The parameters used in the simulations are shown in Table 5. We present the results from two simulations. In the first case, we calculate the roll angle command from

Table 5: Simulation parameters.

V = 20 [m/s]	$k = 0.01$
$k_s = 0.4$	$k_\omega = 0.001$
$\lvert\phi_c^{\max}\rvert = \pi/6$	$k_e = 1.1$
$k_a = 0.7$	$\gamma = 4000$
$\lambda_\phi \approx 1.1$	

Eq. (93) using a time constant $\lambda_\phi = 0.5$ which is known inaccurate. Without parameter adaptation, as shown in Fig. 37, we observe a sluggish and low damped response of the actual trajectory. The error variables are shown in Fig. 38, and the command inputs are shown in Fig. 39.

In the second case, we calculate the roll angle command from Eq. (94) with the parameter update law of Eq. (100). Figures 40-43 show the results. The error states tend to zero asymptotically as shown in Fig. 41. As the command inputs are shown in Fig. 42, the roll angle command is limited within $\pm\pi/6$, hence the UAV is unable to exactly follow the path where the curvature exceeds the maximum curvature achievable by the UAV at the speed of 20 [m/sec]. Nevertheless, the path following control law forces the UAV converge to the path asymptotically after a short transient. Finally, Fig. 43 displays the time history of the estimate of λ_ϕ.

4.5 Summary

In this chapter a nonlinear path following control law has been developed for a small UAV using backstepping of the heading rate command. The kinematic control law realizes cooperative path following control such that the motion of a virtual target is controlled by an extra control input to help the convergence of the error variables. A roll command that gives rise to the desired heading rate has been derived by taking into account the inaccurate system time constant with parameter adaptation scheme. From a high-fidelity hardware-in-the-loop (HIL) simulation results, the presented algorithm is validated for the applicability to the actual UAV.

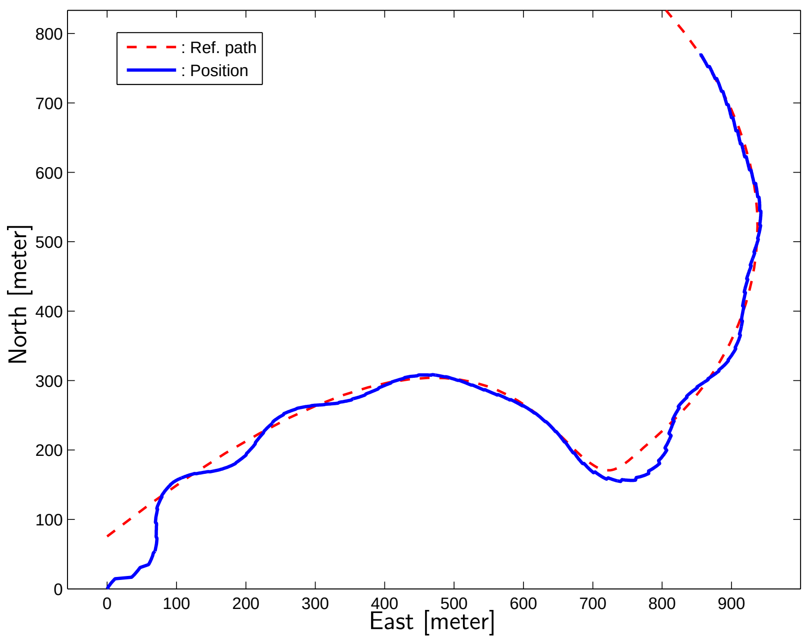
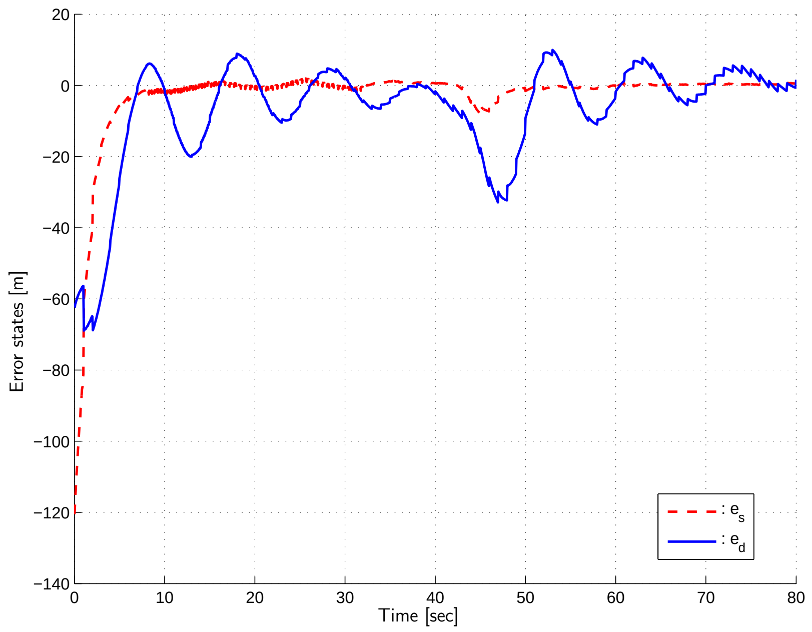
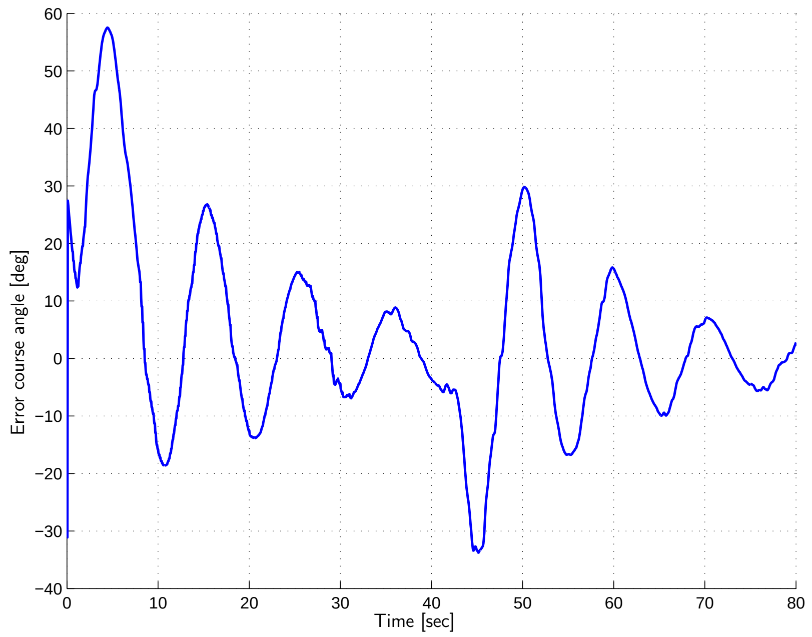


Figure 37: Reference path and actual trajectory of the UAV without parameter adaptation.

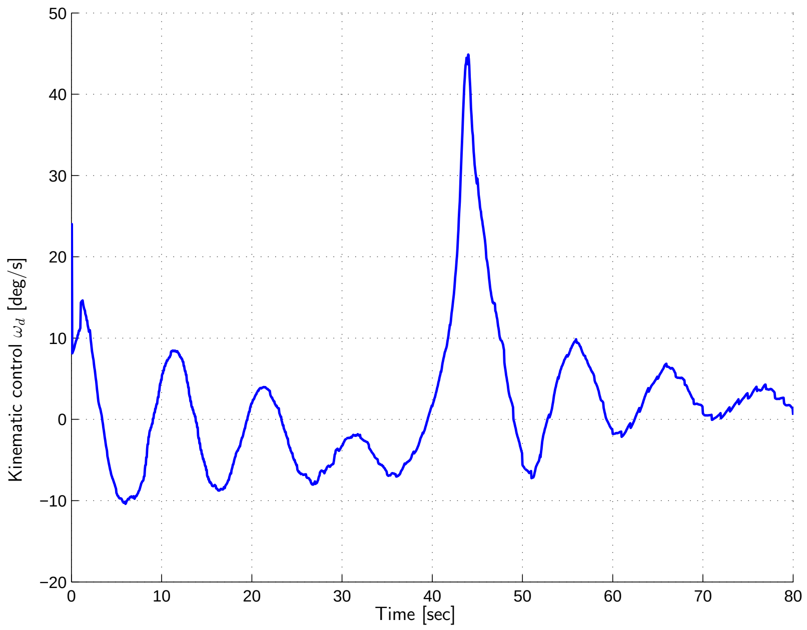


(a) Track errors

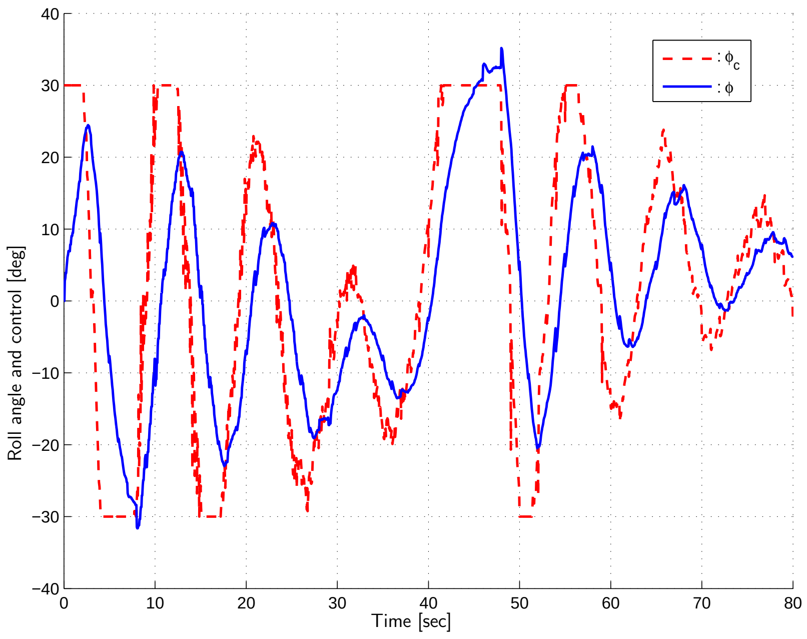


(b) Course angle error

Figure 38: Error states without parameter adaptation.



(a) Heading rate command



(b) ϕ v/s ϕ_c

Figure 39: Command inputs without parameter adaptation.

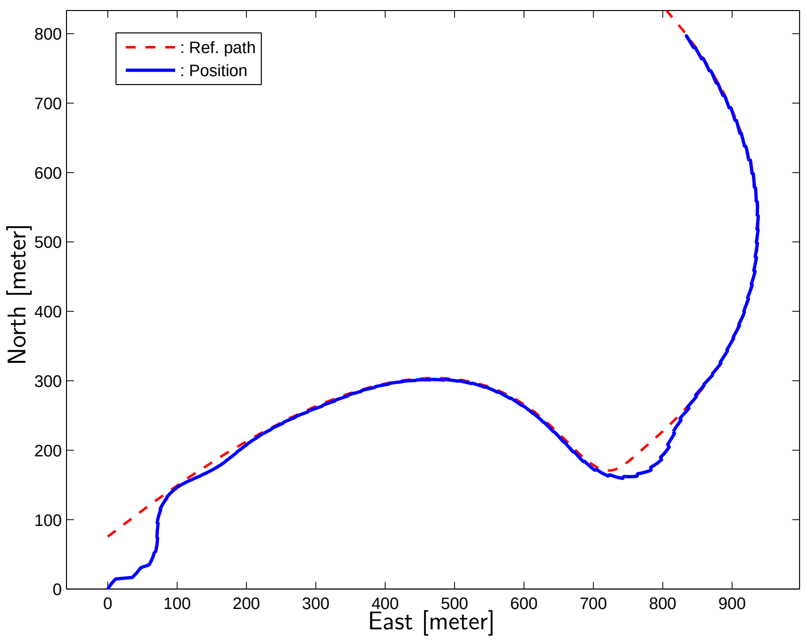
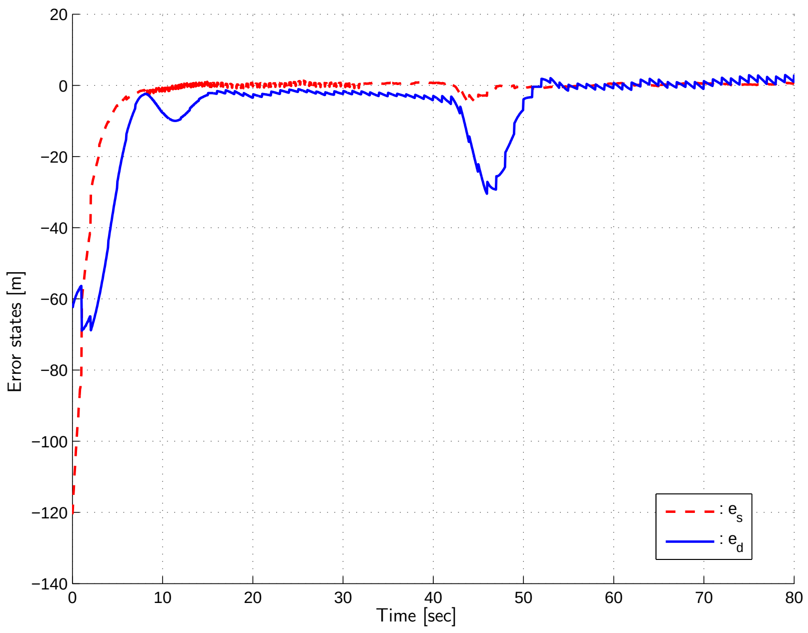
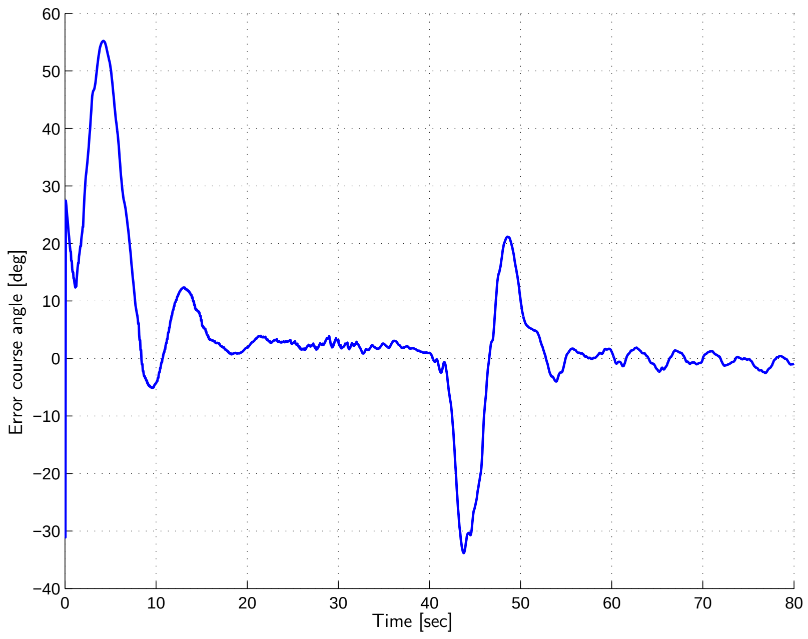


Figure 40: Reference path and actual trajectory of the UAV with parameter adaptation.

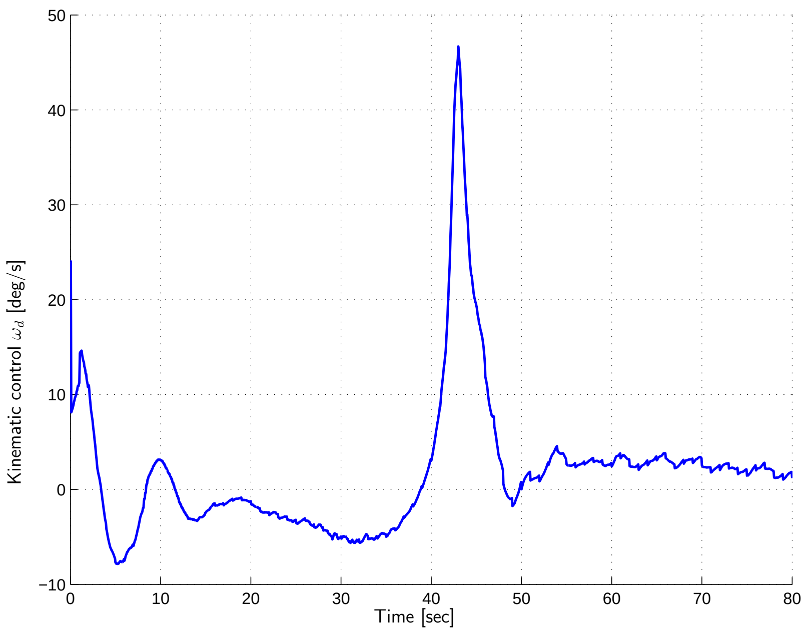


(a) Track errors

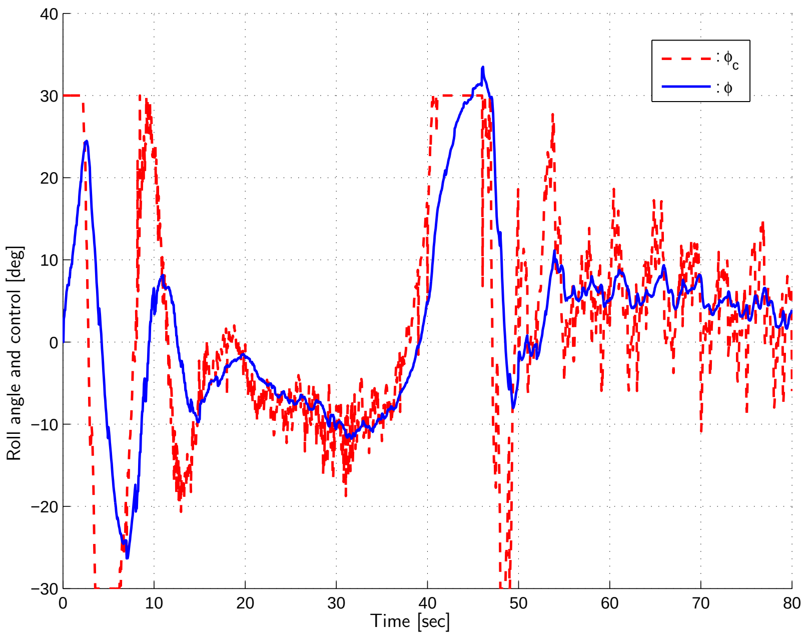


(b) Course angle error

Figure 41: Error states with parameter adaptation.



(a) Heading rate command



(b) ϕ v/s ϕ_c

Figure 42: Command inputs with parameter adaptation.

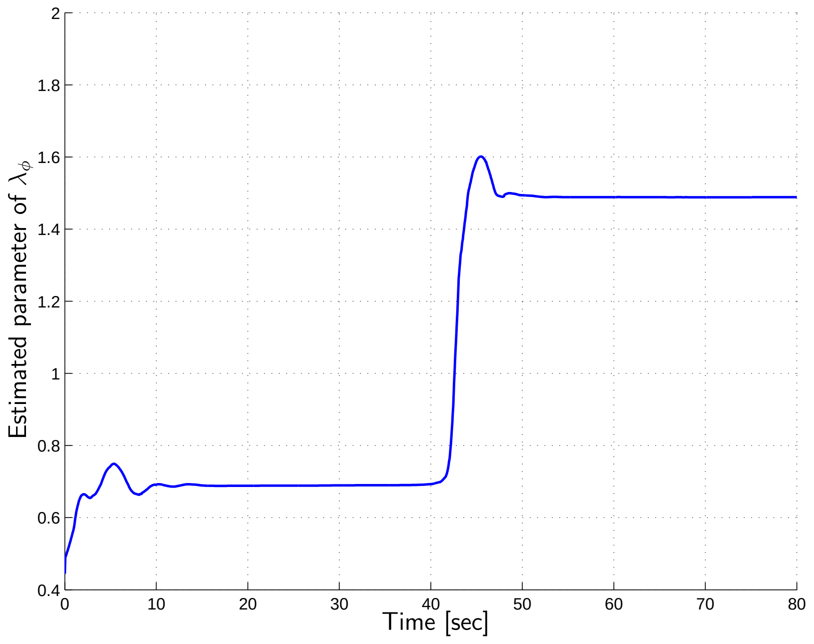


Figure 43: Parameter estimate of λ_ϕ.

CHAPTER V

REAL TIME IMPLEMENTATION OF THE HIERARCHICAL PATH CONTROL ALGORITHM USING HARDWARE-IN-THE-LOOP SIMULATION

In this chapter we present on-line, real-time hardware-in-the-loop simulation results of the hierarchical path control algorithm using a small micro-controller. The control hierarchy, as depicted in Fig. 44, consists of path planning (Chapter 2), path smoothing (Chapter 3), and path following control (Chapter 4).

At each stage of the control hierarchy, the required commands for the next stage are calculated, which finally drives the control surfaces of the fixed-wind UAV. The information regarding the world is initially given to the autopilot by a two dimensional elevation map. With the given initial position of the UAV, the user can choose any arbitrary goal position for the UAV to fly safely. Details of the implementation are discussed in the sequel.

5.1 Hardware description

A UAV research platform, as shown in Fig. 45, has been developed to support the UAV research. The development of the hardware and software was done completely in-house to have a full access to the entire system. The on-board autopilot is equipped with a micro-controller, sensors and actuators, along with the communication devices. The micro-controller adopted in this research was chosen to be a Rabbit RCM-3400 module which runs at 30 MHz clock speed equipped with 512 KB RAM for data and 512 KB ROM for program.

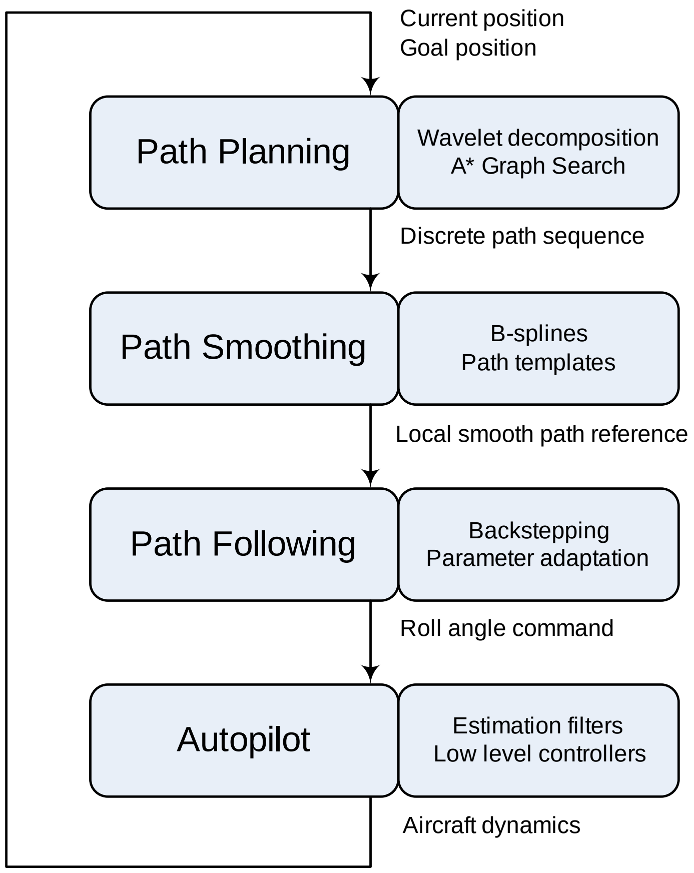


Figure 44: Block diagram for control hierarchy of the proposed path control algorithm.

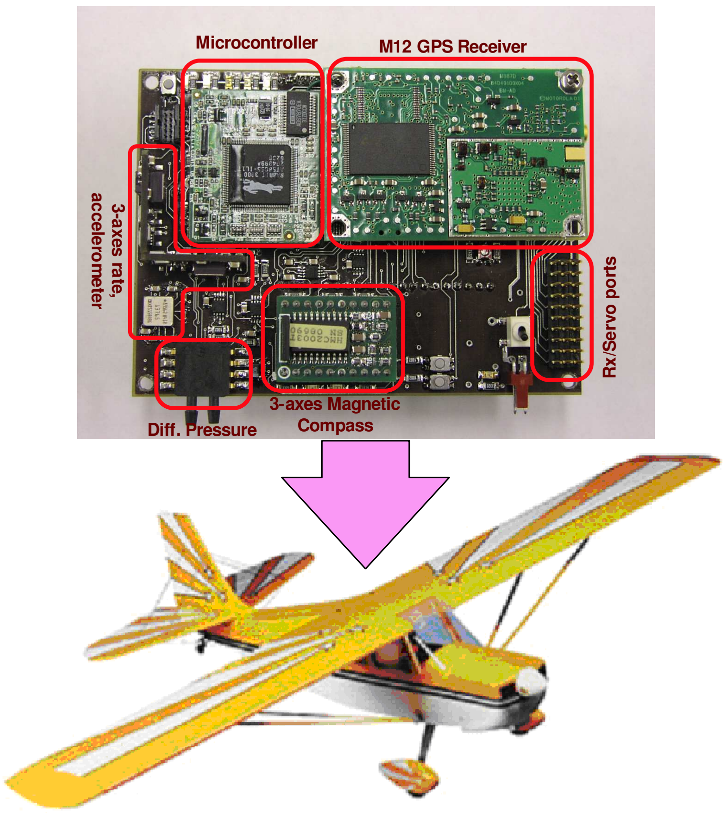


Figure 45: A small fixed-wing UAV equipped with an autopilot for hierarchical path planning control.

5.2 Hardware-in-the-loop simulation environment

A realistic hardware-in-the-loop simulation (HILS) environment has also been developed to validate the UAV autopilot hardware and software development utilizing Matlab® and Simulink®. A full 6-DOF nonlinear aircraft model is used in conjunction with a linear approximation of the aerodynamic forces and moments, along with Earth gravitational (WGS-84) and magnetic field models. Detailed models for the sensors and actuators have also been incorporated. Four independent computer systems are used in the hardware-in-the-loop simulation (HILS) as illustrated in Fig. 46. A 6-DOF simulator, the flight visualization computer, the autopilot micro-controller, and the ground station computer console are involved in the simulation. Further details about the UAV platform, autopilot and HILS set-up are described in the

Appendices.

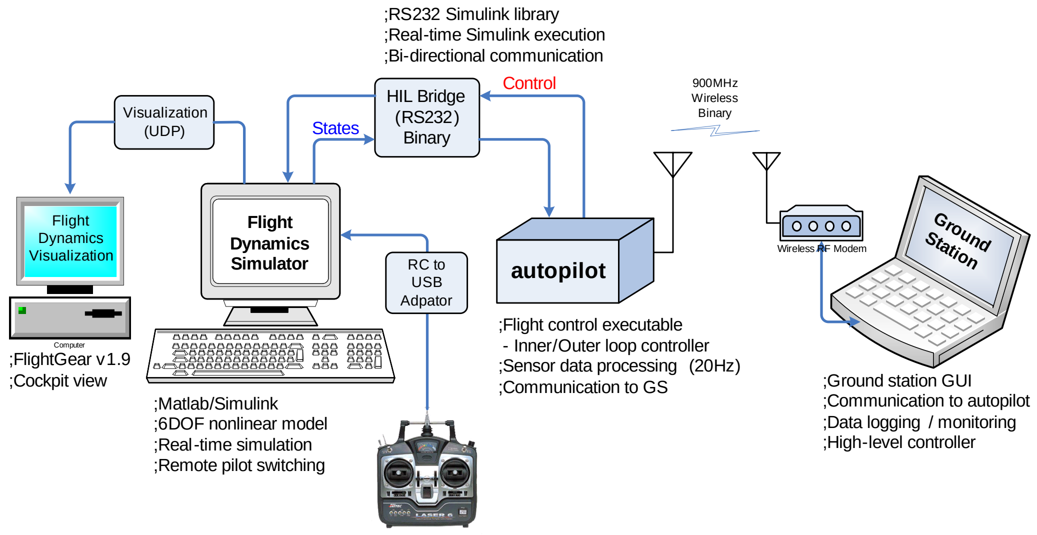
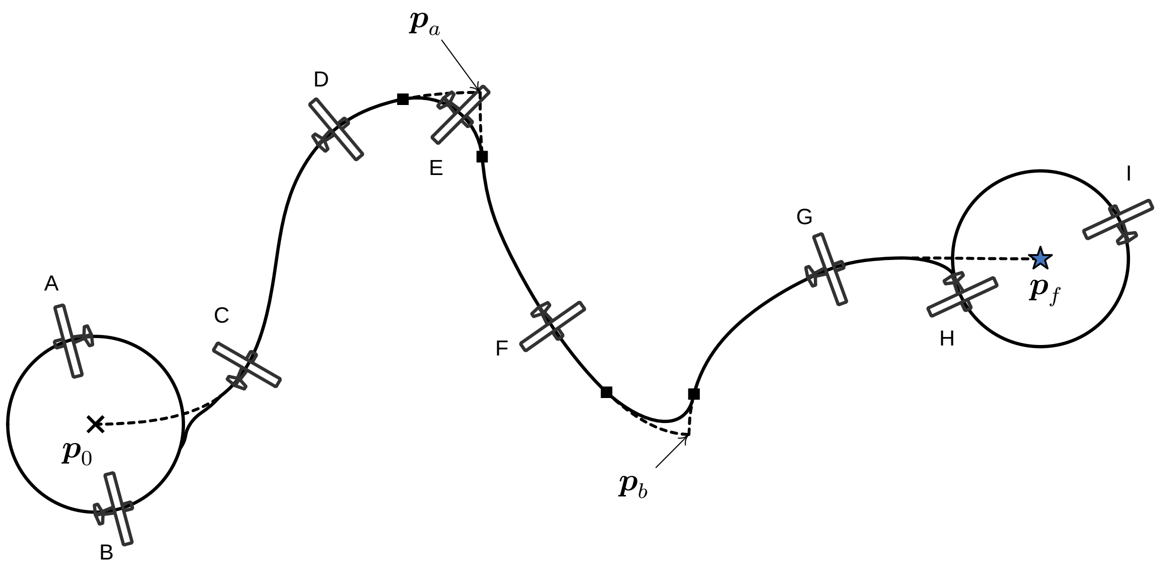


Figure 46: High fidelity hardware-in-the-loop simulation (HILS) environment that enables rapid testing of the proposed path planning algorithm.

5.3 *Simulation scenario*

The environment $\mathcal{W}$ is the elevation map of a certain area in US state. The environment is assumed to be square of dimension 128×128 units, which corresponds to 9600×9600 meters in actual size. Taking into account the available memory of the micro-controller, we choose the fine resolution level as $J_{\max} = 6$ and the coarser resolution level as $J_{\min} = 3$. Cells at the fine resolution have dimensions 150×150 meters, which is slightly larger than the minimum turning radius of the fixed-wing UAV. The minimum turning radius is approximately calculated for the UAV flying at the constant speed of $V_T = 20$ [m/sec] with the bounded roll angle $|\phi| \leq 30°$, resulting in $R_{\min} \approx 70$ [meter].

The objective of the UAV is to follow a path from the initial position to the final position while circumventing the obstacles over a certain elevation threshold. Figure 47 illustrates the detail on-line implementation of the proposed path control algorithm. Initially, the UAV is loitering around the initial position $\boldsymbol{p}_0$ until when



Step	Task description
A	Initially, the UAV is loitering around the initial position with the circle radius R_l
B	Calculate the the first path segment from $\boldsymbol{p}_0$ to $\boldsymbol{p}_a$
C	Break away from the loiter circle, start to follow the first path segment
D	Calculate the second path segment from $\boldsymbol{p}_a$ to $\boldsymbol{p}_b$, and a transient path connecting two paths
E	UAV is on the transient path
F	Calculate the third path segment, and a transient path
G	UAV is approaching the goal position, no path calculated
H	The goal is reached, end of the path control, get on the loitering circle
I	UAV is loitering around the goal position $\boldsymbol{p}_f$

Figure 47: Illustration of the on-line implementation of the proposed hierarchical path control algorithm.

a local path segment from $\boldsymbol{p}_0$ to $\boldsymbol{p}_a$ is computed (Step A,B). Subsequently, the path following controller is engaged to follow the path (Step C,D). At step D, the UAV replans to compute a new path from the intermediate location $\boldsymbol{p}_a$ to the goal, resulting in the second local path segments from $\boldsymbol{p}_a$ to $\boldsymbol{p}_b$. The first and second path segments are stitched by a transient B-spline path assuring the continuity condition at each intersection points (marked by black squares). This process iterates until the final position $\boldsymbol{p}_f$ is reached (Step H), when the UAV engages to loiter around the goal location.

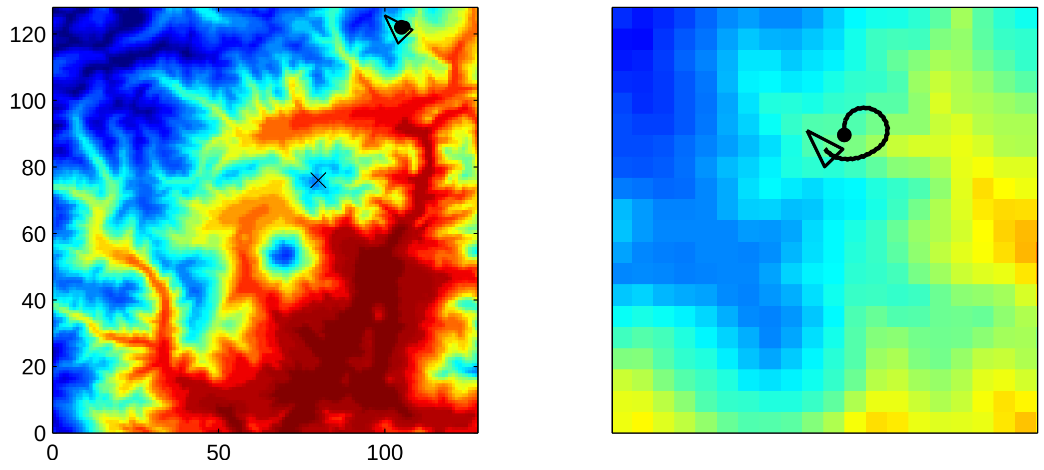
5.4 Simulation results

Figure 48 shows the simulation results of the hierarchical path control implementation. Specifically, figures on the right side show the close-up view of the simulation. The channels are drawn by polygonal lines, and the UAV smoothly follows the reference path avoiding the possible obstacles outside the channels. Consequently, the UAV reaches the final destination in a collision free manner, as seen in Fig. 48(o).

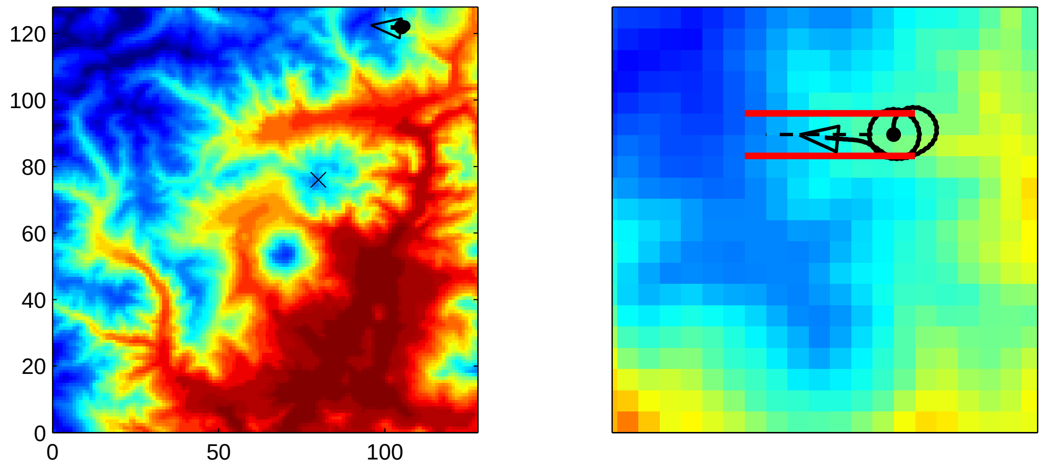
Figure 49 shows a 3D screen shot during the simulation. The ground track of the followed path is displayed showing that the UAV is avoiding the obstacles (for this case, it is the high elevation region) effectively.

5.5 Summary

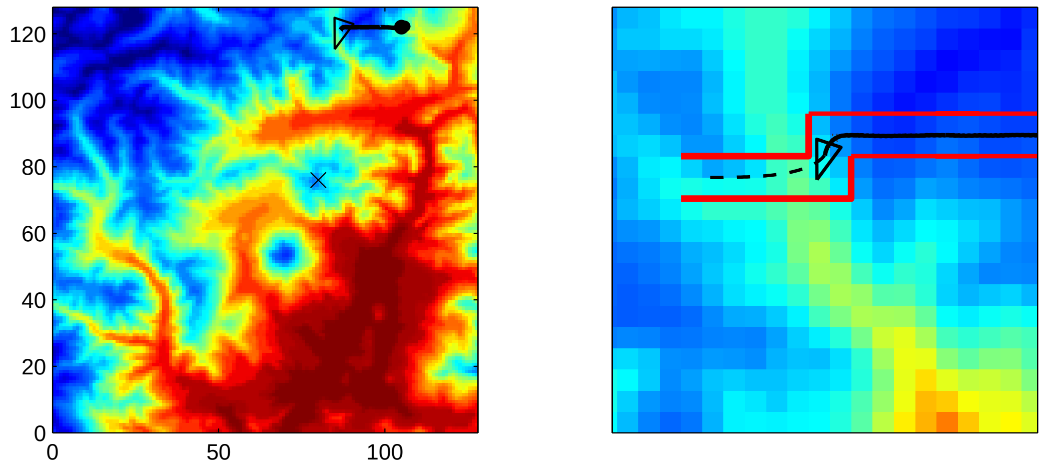
We implement the hierarchical path control of a small UAV on the actual hardware platform. Based on the high fidelity hardware-in-the-loop (HIL) simulation environment, the proposed hierarchical path control algorithm has been validated through the on-line, real-time implementation on a small micro-controller. By a seamless integration of the control algorithms for path planning, path smoothing, and path following, it has been demonstrated that the UAV equipped with a small autopilot having limited computational resources manages to accomplish the path control objective to reach the goal while avoiding obstacles with minimal human intervention.



(a) $t = 27.5$ [sec]

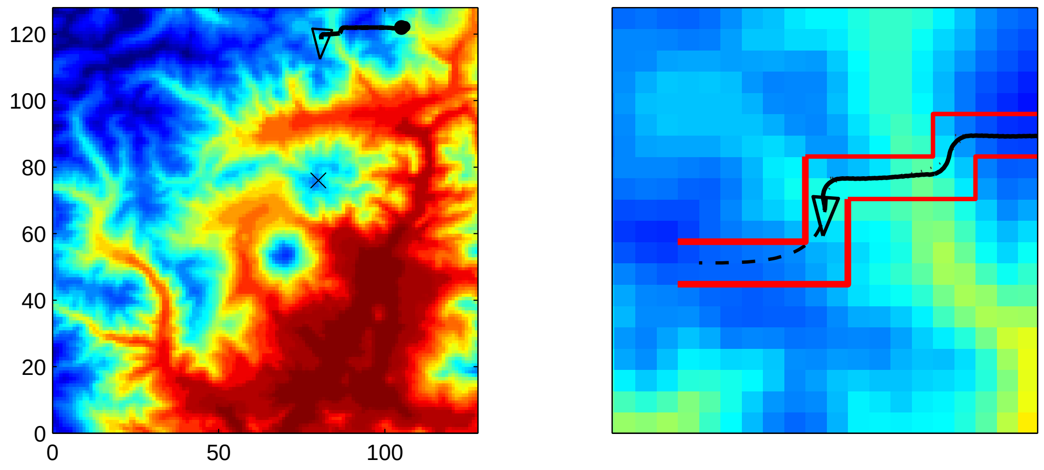


(b) $t = 64.5$ [sec]

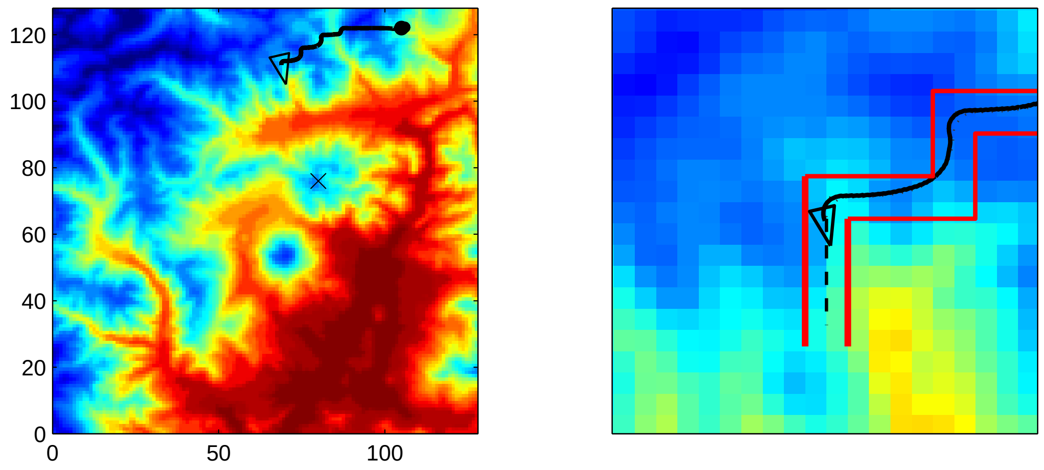


(c) $t = 126.5$ [sec]

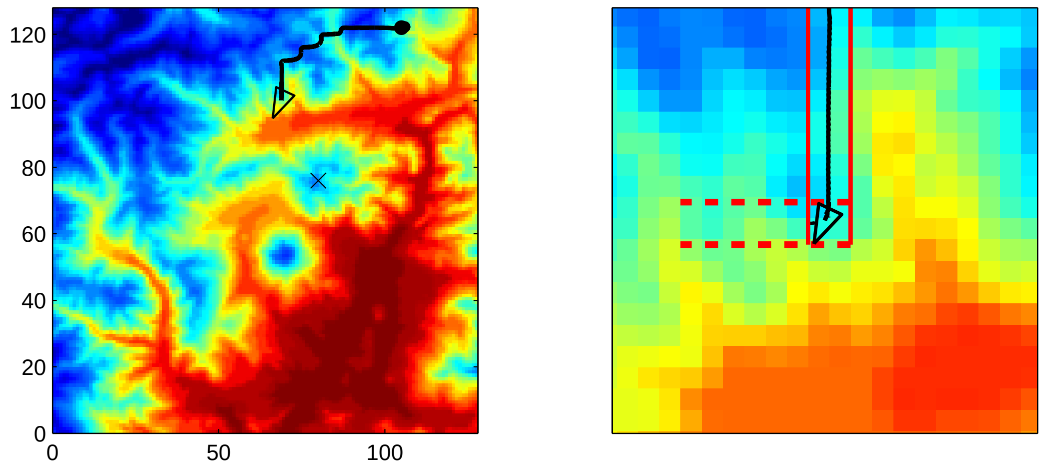
Figure 48: Simulation results of the hierarchical path control implementation. Figures on the right show the close-up view of the simulation. At each instant the channel is drawn by polygonal lines, where the smooth path segment from the path templates stays. The actual path followed by the UAV is drawn on top of the reference path.



(d) $t = 156.5$ [sec]

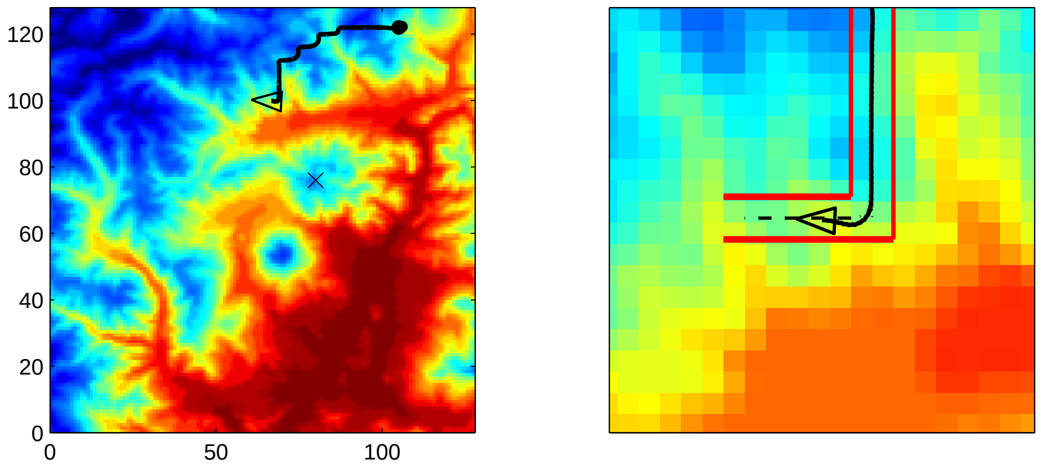


(e) $t = 222.0$ [sec]

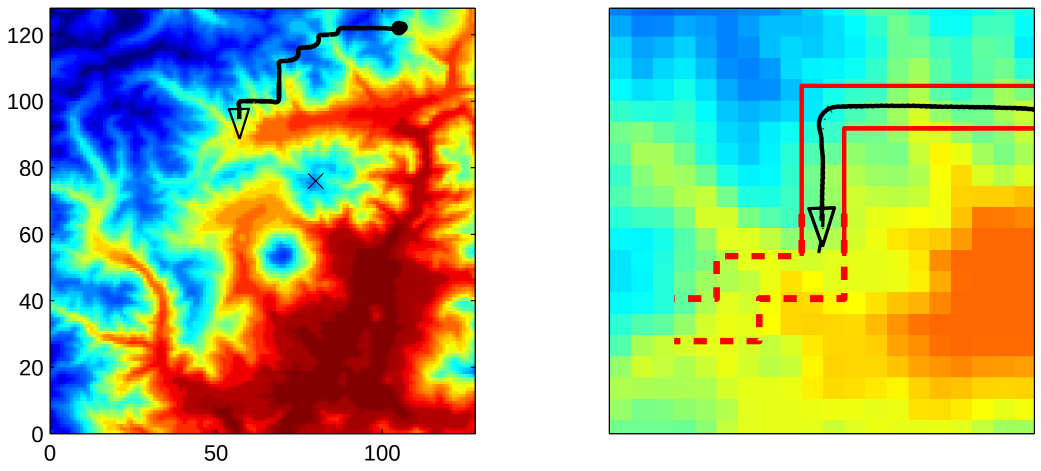


(f) $t = 265.0$ [sec]

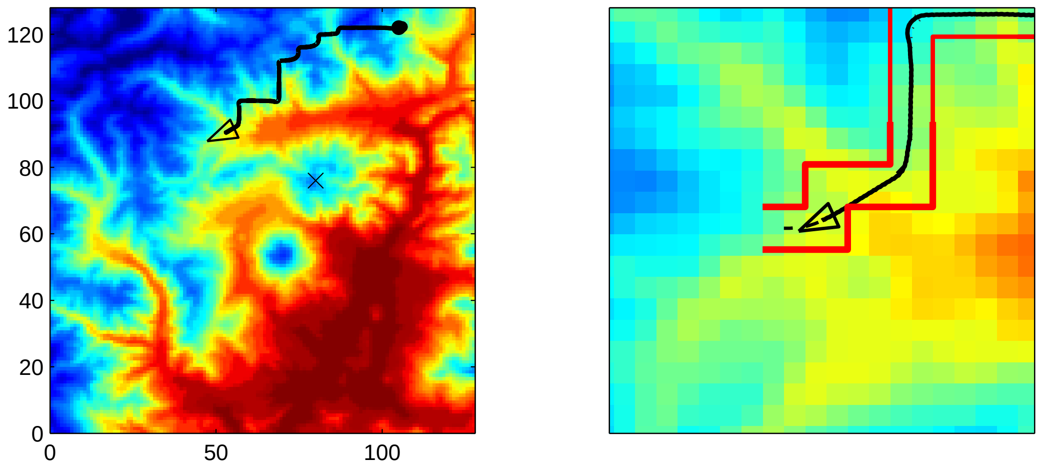
Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)



(g) $t = 274.0$ [sec]

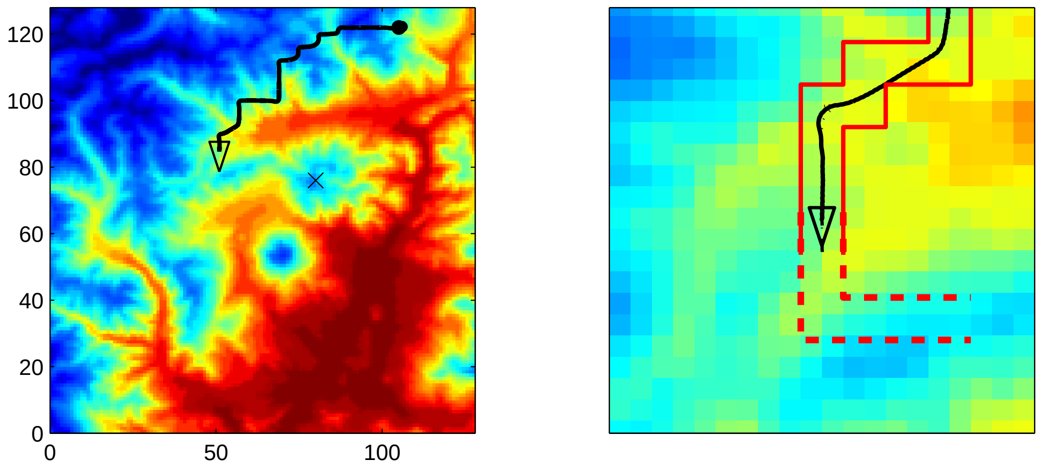


(h) $t = 333.0$ [sec]

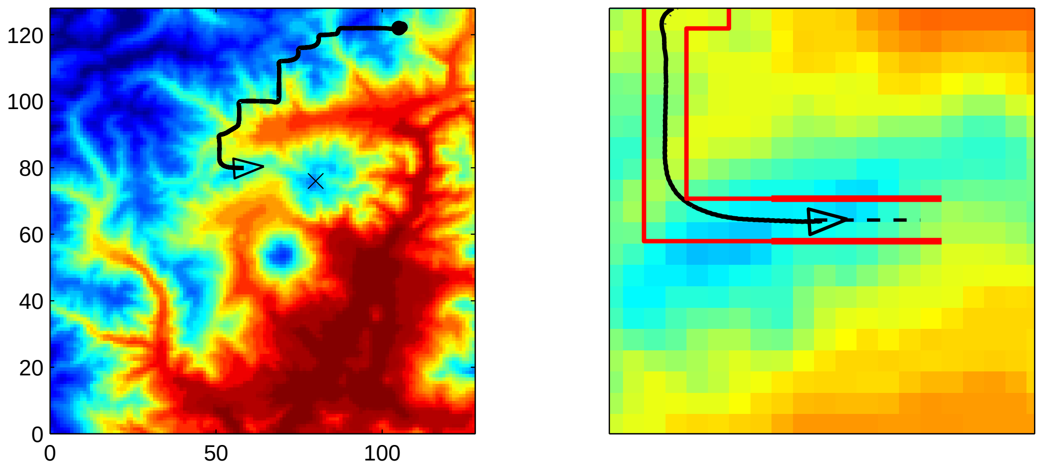


(i) $t = 358.5$ [sec]

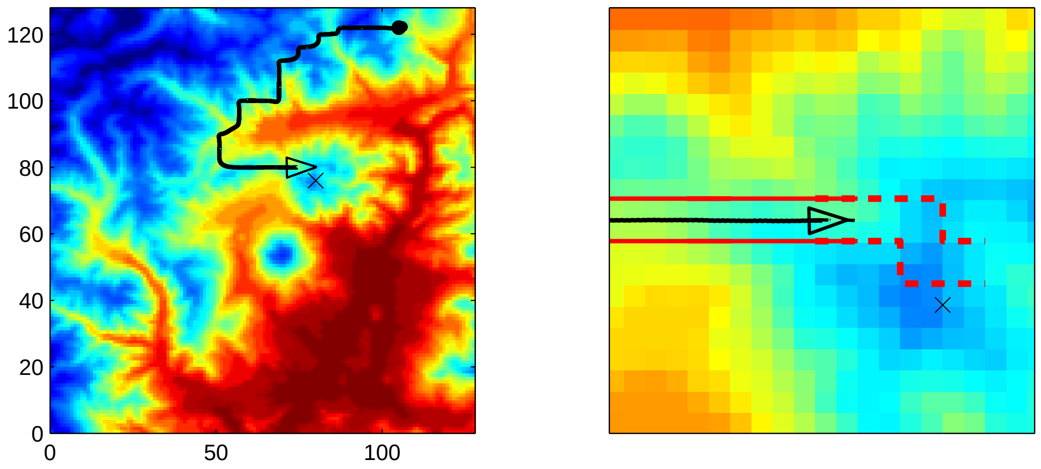
Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)



(j) $t = 386.5$ [sec]

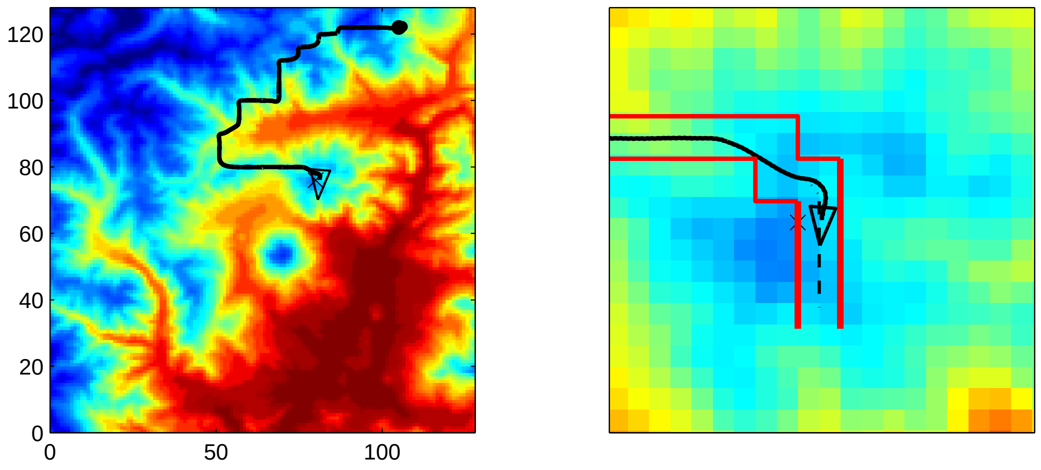


(k) $t = 429.0$ [sec]

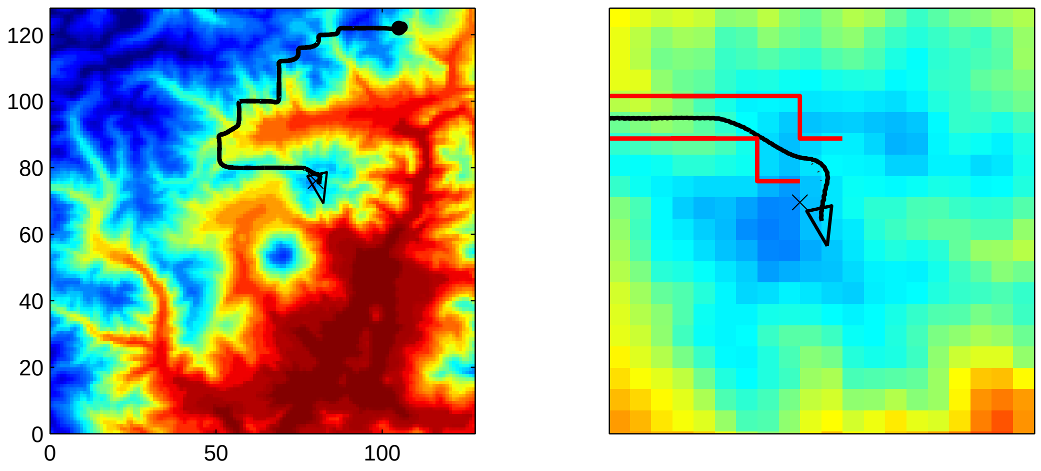


(l) $t = 492.5$ [sec]

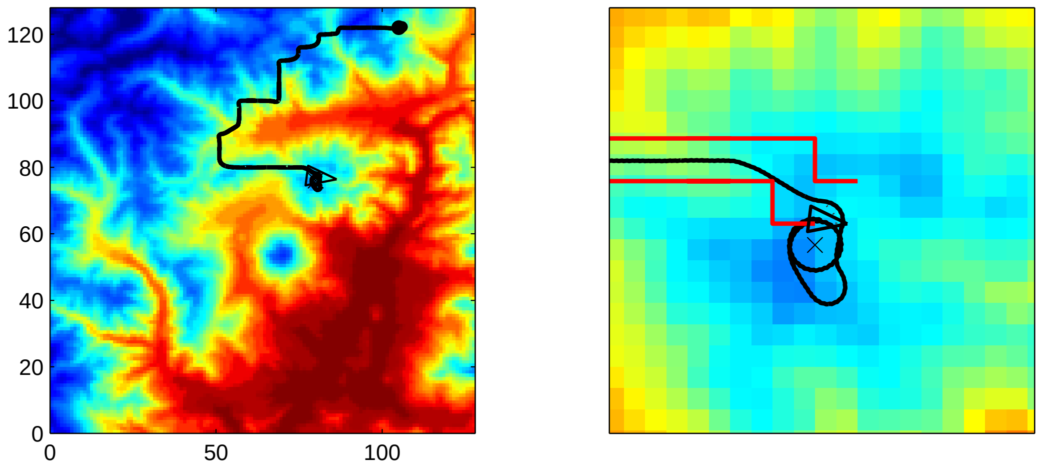
Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)



(m) $t = 528.0$ [sec]



(n) $t = 532.0$ [sec]



(o) $t = 591.5$ [sec]

Figure 48: Simulation results of the hierarchical path control implementation. (cont'd)

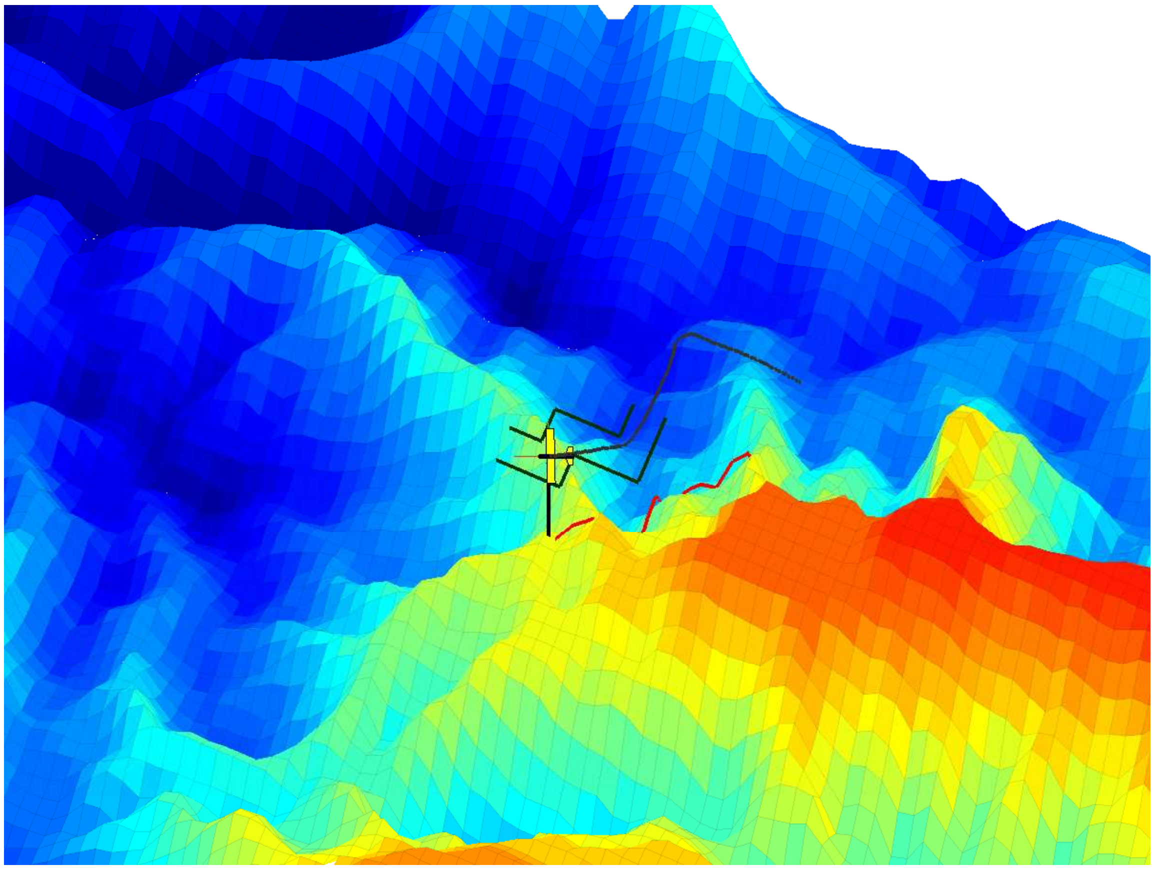


Figure 49: A 3D screen shot during the simulation. The ground track of the followed path is displayed showing that the UAV is avoiding the obstacles (for this case, it is the high elevation region).

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

Autonomous control for small UAVs imposes severe restrictions on the control algorithm development, stemming from the limitations imposed by the on-board hardware and the requirement for on-line implementation. In this thesis we have proposed a new hierarchical control scheme for the navigation and guidance of a small UAV for obstacle avoidance. The multi-stage control hierarchy for a complete path control algorithm is comprised of several control steps: Top-level path planning, mid-level path smoothing, and bottom-level path following controls. In each stage of the control hierarchy, the limitation of the on-board computational resources has been taken into account to come up with a practically feasible control solution. We have validated these developments in realistic non-trivial scenarios.

In Chapter 2 we proposed a multiresolution path planning algorithm. The algorithm computes at each step a multiresolution representation of the environment using the fast lifting wavelet transform. The main idea is to employ high resolution close to the agent (where is needed most), and a coarse resolution at large distances from the current location of the agent. It has been shown that the proposed multiresolution path planning algorithm provides an on-line path solution which is most reliable close to the agent, while ultimately reaching the goal. In addition, the connectivity relationship of the corresponding multiresolution cell decomposition can be computed directly from the the approximation and detail coefficients of the FLWT. The path planning algorithm is scalable and can be tailored to the available computational resources of the agent.

The on-line path smoothing algorithm incorporating the path templates is presented in Chapter 3. The path templates are comprised of a set of B-spline curves, which have been obtained from solving the off-line optimization problem subject to the channel constraints. The channel is closely related to the obstacle-free high resolution cells over the path sequence calculated from the high-level path planner. The obstacle avoidance is implicitly dealt with since each B-spline curve is constrained to stay inside the prescribed channel, thus avoiding obstacles outside the channel. By the affine invariance property of B-spline, each component in the B-spine path templates can be adapted to the discrete path sequence obtained from the high-level path planner. We have shown that the smooth reference path over the entire path can be calculated on-line by utilizing the path templates and path stitching scheme. The simulation results with the $\mathcal{D}^*$-lite path planning algorithm validates the effectiveness of the on-line path smoothing algorithm. This approach has the advantage of minimal on-line computational cost since most of computations are done off-line.

In Chapter 4 a nonlinear path following control law has been developed for a small fixed-wing UAV. The kinematic control law realizes cooperative path following so that the motion of a virtual target is controlled by an extra control input to help the convergence of the error variables. We applied the backstepping to derive the roll command for a fixed-wing UAV from the heading rate command of the kinematic control law. Furthermore, we applied parameter adaptation to compensate for the inaccurate time constant of the roll closed-loop dynamics. The proposed path following control algorithm is validated through a high-fidelity 6-DOF simulation of a fixed-wing UAV using a realistic sensor measurement, which verifies the applicability of the proposed algorithm to the actual UAV.

Finally, the complete hierarchical path control algorithm proposed in this thesis is validated thorough a high-fidelity hardware-in-the-loop simulation environment using the actual hardware platform. From the simulation results, it has been demonstrated that the proposed hierarchical path control law has been successfully applied for path control of a small UAV equipped with an autopilot that has limited computational resources.

6.2 Future Research

In this section, several possible extensions of the work presented in this thesis are outlined.

6.2.1 Reusable graph structure

The proposed path planning algorithm involves calculating the multiresolution cell decomposition and the corresponding graph structure at each of iteration. Hence, the connectivity graph $\mathcal{G}(t)$ changes as the agent proceeds toward the goal. Subsequently, let $\mathsf{x} \in \mathcal{W}$ be a state (location) which corresponds to nodes of two distinct graphs as follows

$$\mathsf{cell}_{\mathcal{G}(t_i)}(v_m^i) = \mathsf{cell}_{\mathcal{G}(t_j)}(v_n^j), \qquad i \neq j \tag{105}$$

By the respective $\mathcal{A}^*$ search on those graphs, the agent might be rendered to visit x at different time steps of t_i and t_j, $i \neq j$. As a result, a cyclic loop with respect to x is formed for the agent to repeat this pathological loop, while never reaching the goal. Although it has been presented that maintaining a visited set might be a means of avoiding such pathological situations[142], it turns out to be a trial-and-error scheme is not a systemical approach. Rather, suppose that we could employ a unified graph structure over the entire iteration, which retains the information from the previous search. Similar to the $\mathcal{D}^*$-lite path planning algorithm, the incremental search over the graph by reusing the previous information results in not only overcoming the pathological situation but also reducing the computational time. In contrast to $\mathcal{D}^*$ or $\mathcal{D}^*$-lite algorithms where a uniform graph structure is employed, a challenge lies in building the unified graph structure from a multiresolution cell decomposition. Specifically, it includes a dynamic, multiresolution scheme for constructing the graph connectivity between nodes at different levels. The unified graph structure will evolve itself as the agent moves, while updating nodes and edges associated with the multiresolution cell decomposition from the FLWT. If this is the case, we might be able to adapt the proposed path planning algorithm to an incremental search algorithm, hence taking advantages of both the efficient multiresolution connectivity (due to the FLWT) and the fast computation (due to the incremental search by using the previous information).

6.2.2 Kinodynamically feasible trajectory generation using B-splines

In this thesis, we utilized a B-spline representation of the reference path for the path smoothing purpose. In general, B-spline curves provide only information in spatial terms without taking into account evolution in terms of time, thus they are not useful to represent state references that are dependent on time. This aspect imposes severe restrictions on designing control algorithms for time critical applications. On the other hand, several authors in the literature [147, 39] proposed to adapt B-spline to explicitly deal with the time variable to design a smooth trajectory subject to the kinematic constraints of the vehicles. Let us consider the following unicycle-like kinematic equations of motion,

$$\dot{x} = v\cos\psi, \tag{106a}$$

$$\dot{y} = v\sin\psi, \tag{106b}$$

$$\dot{\psi} = \omega, \tag{106c}$$

Suppose the reference trajectory can be represented by a B-spline curve in terms of the curve parameter u,

$$\mathbf{r} = \big(x_d(u), y_d(u)\big). \tag{107}$$

It follows from the basic theory of differential geometry for curves[25] that the speed of the agent is determined by,

$$v(u) = \sqrt{\left(\frac{\mathrm{d}x_d}{\mathrm{d}u}\right)^2 + \left(\frac{\mathrm{d}y_d}{\mathrm{d}u}\right)^2}, \tag{108}$$

and the curvature is calculated as

$$\kappa(u) = \frac{x''_d y'_d - y''_d x'_d}{(x'^2_d + y'^2_d)^{3/2}}, \tag{109}$$

where $(\cdot)'$ and $(\cdot)''$ denote the first and second order derivative with respect to u, respectively. The angular velocity can be determined using Eqs. (106) as

$$\omega = \frac{\ddot{y}\dot{x} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}, \tag{110}$$

and can be represented using Eqs (108) and (109) as follows,

$$\omega = \kappa(u)v(u) \tag{111}$$

The kinematic constraints of the vehicles are given by

$$|\omega| \leq \omega_{\max}, \tag{112}$$

and $v \in [v_{\min}, v_{\max}]$, using Eqs. (111) and (112), we get

$$v_{\min} \leq v \leq \left(\frac{\omega_{\max}}{\kappa}\right). \tag{113}$$

Subsequently, an off-line optimization problem can then be formulated taking into consideration the kinematic constraints shown in Eqs. (112) and (113). The path templates which explicitly deal with the kinodynamical constraints such as the bounded flight speed or the minimum turning radius are constructed from a set of off-line optimization problems.

6.2.3 Trajectory tracking controller design using differential flatness

In this thesis, we proposed the path following control law for tracking a reference path. Although the path following control algorithm was successfully implemented to follow a smooth reference path, no explicit consideration of vehicle dynamics and the kinematic constraints of a fixed-wing UAV was given. Furthermore, a non-zero constant speed of the UAV was assumed, hence it is not possible for the UAV to track the time-stamped trajectory using the proposed path following controller. In order to take into account the trajectory tracking capability of the UAV in conjunction with the vehicle dynamics, we consider a simplified kinematic model in the two dimensional plane as follows,

$$\dot{x} = u_1 \cos \chi, \tag{114a}$$

$$\dot{y} = u_1 \sin \chi, \tag{114b}$$

$$\dot{\chi} = u_2, \tag{114c}$$

where (x, y) denotes the inertial position and χ is the inertial heading angle. The control inputs $\mathbf{u} = [u_1 \ u_2]^\mathsf{T}$ consist of the forward flight speed and the heading rate of the UAV, respectively. Given a reference trajectory, we design a control law for the flight speed and the heading rate commands. To this end, we let $\mathbf{z} = [z_1 \ z_2]^\mathsf{T}$ be an output of the system,

$$z_1 = x, \qquad z_2 = y. \tag{115}$$

It follows from Eqs. (114a) and (114b) that

$$\tan \chi = \frac{\dot{y}}{\dot{x}}, \tag{116}$$

which shows that the system state $\mathbf{x} = [x \ y \ \chi]^\mathsf{T}$ is represented by the output as follows,

$$\mathbf{x} = \begin{bmatrix} z_1 \\ z_2 \\ \tan^{-1}\left(\frac{\dot{z}_2}{\dot{z}_1}\right) \end{bmatrix} \tag{117}$$

The vehicle forward flight speed is obtained from Eqs. (114a) and (114b), which yields the control input u_1 expressed in terms of the output as follows,

$$u_1 = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{118}$$

By differentiating Eq. (116) with respect to time and using trigonometry, we obtain the expression for u_2 in terms of the output

$$u_2 = \frac{\ddot{y}\dot{x} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}. \tag{119}$$

The control vector $\mathbf{u}$ is represented by the output as follows,

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{z}_1^2 + \dot{z}_2^2} \\ \frac{\ddot{z}_2\dot{z}_1 - \dot{z}_2\ddot{z}_1}{\dot{z}_1^2 + \dot{z}_2^2} \end{bmatrix} \tag{120}$$

Therefore, the system in Eq. (114) is proved differentially flat in conjunction with the flat output $\mathbf{z} = [x\ y]^{\mathsf{T}}$, since each state and control input can be expressed in terms of the flat output and its higher derivatives,

$$\mathbf{x} = \mathcal{X}(z_1, z_2, \dot{z}_1, \dot{z}_2), \tag{121a}$$

$$\mathbf{u} = \mathcal{W}(z_1, z_2, \dot{z}_1, \dot{z}_2, \ddot{z}_1, \ddot{z}_2) \tag{121b}$$

Now, given a desired trajectory in terms of $\mathbf{x}(t)$, $\dot{\mathbf{x}}_d(t)$, $\ddot{\mathbf{x}}_d(t)$, we can design a trajectory tracking control law in the flat space $\mathcal{E} = \{\mathcal{X}, \mathcal{W}\}$, explicitly dealing with the input constraints of the forward flight speed, as specified by

$$v_{\min} \leq u_1 \leq v_{\max}, \tag{122}$$

and the bounded heading rate as follows,

$$|u_2| \leq \omega_{\max}. \tag{123}$$

6.2.4 Path planning in three dimensions

The proposed path planning algorithm is based on the assumption that the UAV is restricted to navigate through a two dimensional environment, that is, the UAV flies at constant altitude while avoiding in-plane obstacles. On the other hand, by allowing the UAV to change altitude during the mission, we can take the advantage of three dimensional maneuvers of the UAV to plan a realistic three dimensional path that avoids 3D obstacles effectively. To this end, without loss of generality, we let $\mathcal{W} = [0,1] \times [0,1] \times [0,1]$. We assume that we are given a 3D function $\mathsf{RM} : \mathcal{W} \mapsto \mathcal{M}$ that represent the risk measure at the location $\mathsf{x} = (x, y, z)$. This three dimensional function is tailored to capture not only the aspect of 3D risk measure depending on the operational altitude but also the attainable maneuverability of the UAV between distinct altitudes. We apply the multidimensional FLWT on this function, which results in the the following decomposition,

$$\mathsf{RM}(x,y,z) = \sum_{k,\ell,m=0}^{2^J-1} a_{J,k,\ell,m}\Phi_{J,k,\ell,m}(x,y,z) + \sum_{i=1}^{8}\sum_{j=J}^{N-1}\sum_{k,\ell,m=0}^{2^j-1} d^i_{j,k,\ell,m}\Psi^i_{j,k,\ell,m}(x,y,z), \tag{124}$$

where $\Phi_{J,k,\ell,m}(x,y,z)$ and $\Psi^i_{j,k,\ell,m}(x,y,z)$, $i = 1, \cdots, 8$ are families of function, which are derived from a linear combination of both the scaling function $\phi(\cdot)$ and the wavelet function $\psi(\cdot)$ in each coordinate. Subsequently, we are able to obtain the multiresolution, three dimensional cell decomposition of $\mathcal{W}$, which is further associated with a topological graph structure with a connectivity information. It is then imperative that we extend the approach discussed in Chapter 2 in order to construct the connectivity relationship of the graph by utilizing the corresponding approximation and detail coefficients of the FLWT. Consequently, the rest of 3D path planning collapses to finding an optimal sequence of cells in the graph structure in conjunction with standard graph search methods such as $\mathcal{A}^*$ or Dijkstra's algorithm.

APPENDIX A

UAV AVIONICS DESCRIPTION

A.1 System Architecture

The overall architecture of the UAV system is shown in Fig. 50. The main subsystems are the autopilot, the ground station, and the interconnection between the two. The on-board autopilot is equipped with a micro-controller, sensors and actuators, along with the communication devices that allow full functionality for autonomous control. The micro-controller provides data acquisition, processing, and communication with the ground station. It also runs the main control software. Table 6 shows the detail specification of the micro-controller. The on-board sensors include angular rate sensors for three axes, accelerometers for three axes, a three-axis magnetic compass, a GPS sensor, an engine RPM sensor, absolute and differential pressure sensor, battery voltage sensor, and temperature sensor.

Table 6: Specifications of the Rabbit RCM-3400 micro-controller module.

Parameters	Values
Clock speed	29.4MHz
Programmable memory	512KByte
Data memory	512KByte
Analog Inputs	Eight channels Single-Ended, 11bit resolution
Serial ports	Six configurable asynchronous/SPI SDLC
Timers	Ten independent 8bit timers
Pulse-Width Modulators	four independent 10bit free-running counters
Power consumption	Max. 100mA at 3.3V operation
Size	$1.37'' \times 1.16'' \times 0.31''$

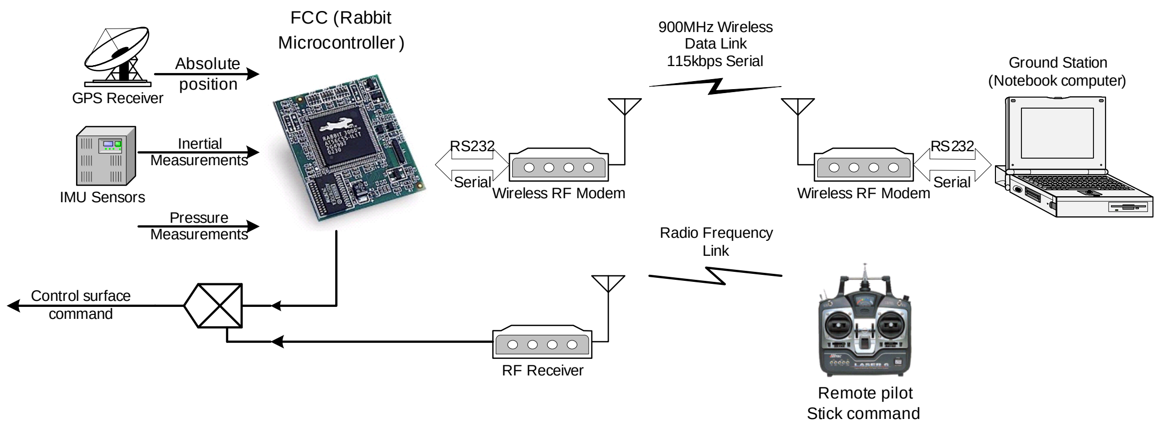


Figure 50: System architecture of the UAV test-bed.

A.2 Autopilot

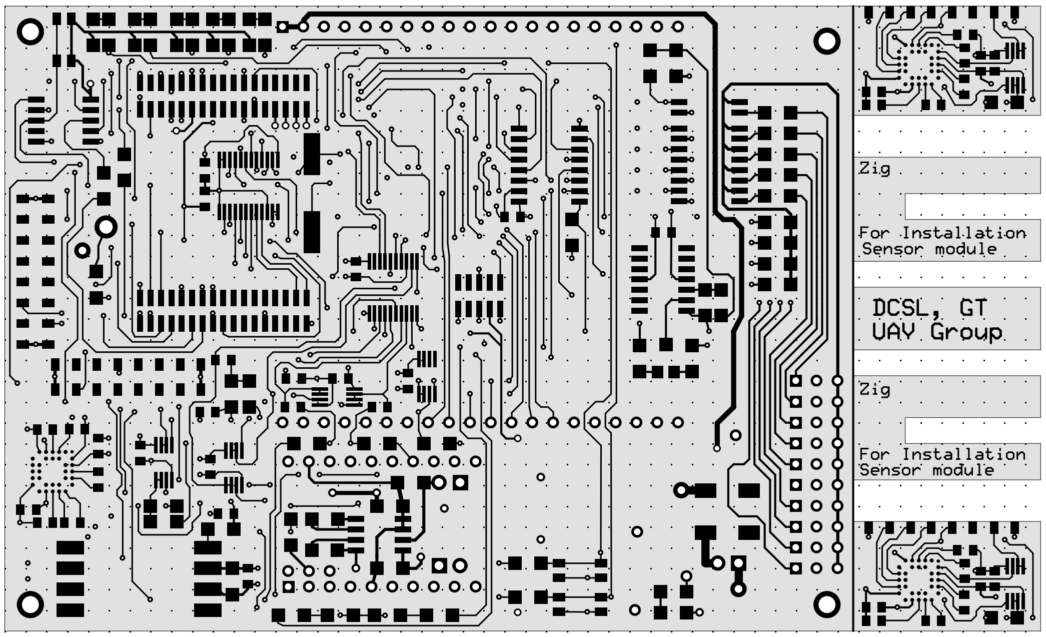
The autopilot box contains all hardware components, such as the micro-controller, all sensor ICs, signal conditioning circuitry, data acquisition devices, and the wireless modem board.

A.2.1 Sensor Board

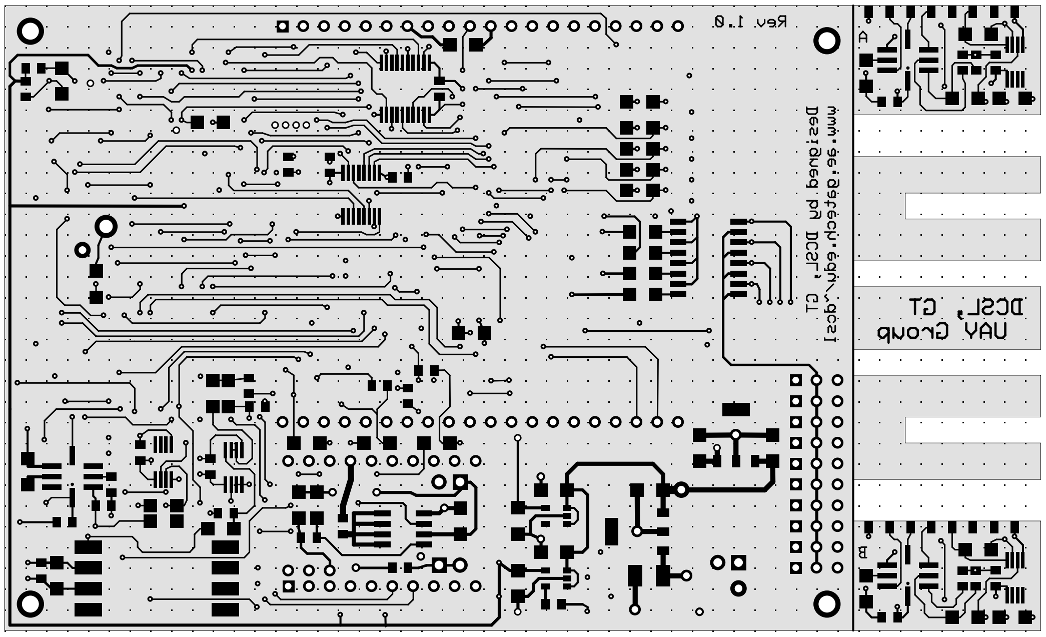
The microprocessor, sensors and associated electronics were integrated on a custom-designed and fabricated four-layer 5″ by 3″ printed circuit board (PCB). Figure 51 shows the detail layout of the four-layer sensor board for the top and bottom layers. The sensor board is equipped with three single-chip rate gyros, three two-axis accelerometers, a three-axis magnetometer, two pressure sensors, and a GPS receiver interfacing to the micro-controller module. It also includes the power regulating circuitry that supplies power for all electronic components. Figure 52 shows the functional diagram of the sensor board and Fig. 53 shows the top and side views of the completed sensor board with all components assembled.

A.2.2 Inertial Sensors

Three ADXRS150 angular rate sensors from Analog Devices provide three-axis body-fixed angular rate measurements. Measurements of linear accelerations in all three



(a) Top layer design



(b) Bottom layer design

Figure 51: Sensor board design layout. The board is 5″ by 3″ printed circuit board (PCB). Four layers include the power plane and the ground plane (Not shown above).

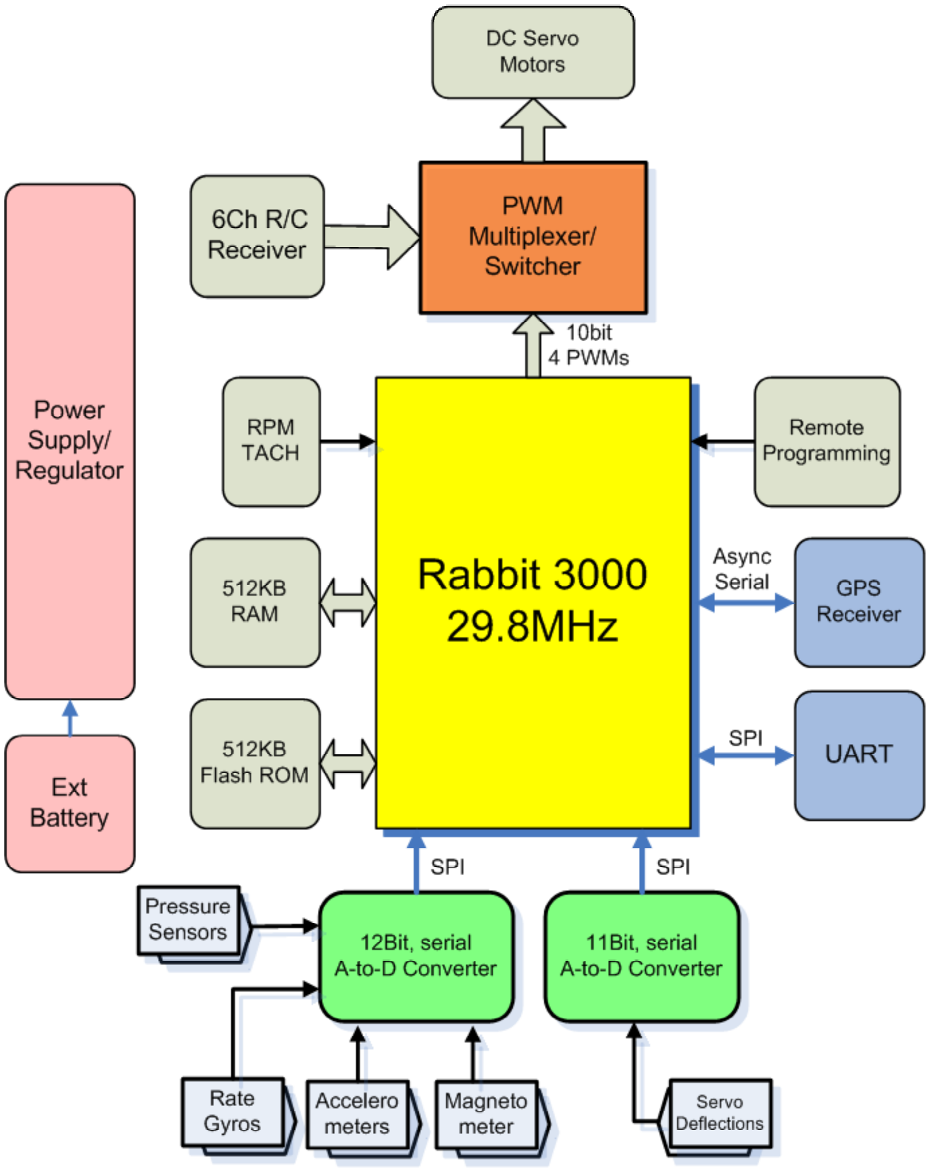
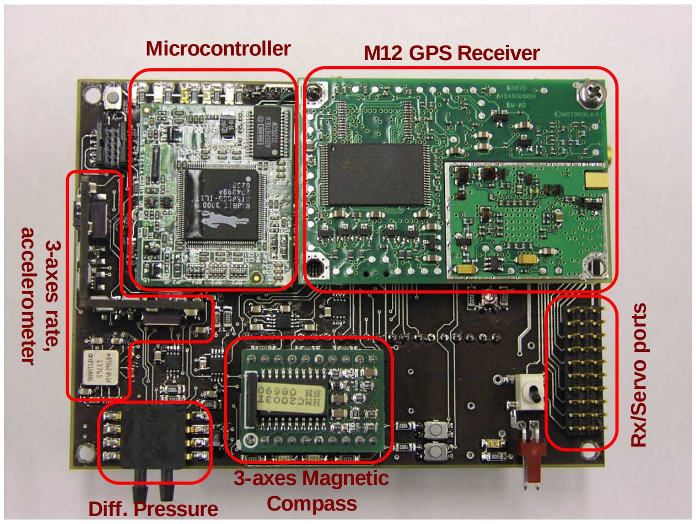
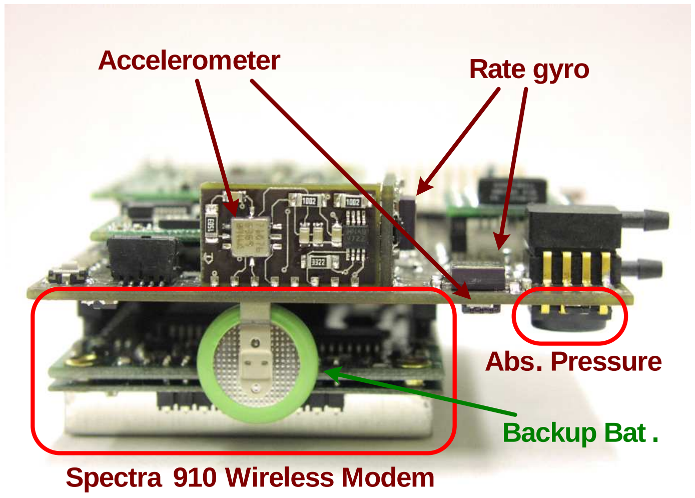


Figure 52: Sensor board functional block diagram.



(a) Top view



(b) Side view

Figure 53: Assembled autopilot hardware.

Table 7: Inertial sensors specifications

Parameters	Values	Remarks
MEMS angular rate sensor, Analog Device ADXRS150		
Dynamic range	± 150 °/sec	
Sensitivity	12.5$\pm$10% mV/°/sec	
Sensitivity nonlinearity	0.1% FS	from the best fit line
Rate Noise Density	0.05 °/sec/$\sqrt{\text{Hz}}$	
Bandwidth	DC to 2 kHz	
MEMS linear acceleration sensor, Analog Device ADXL202		
Dynamic range	± 2 g	1 g = 9.81 m/sec^2
Sensitivity	312 mV/g	
Sensitivity nonlinearity	0.2% FS	from the best fit line
Acceleration Noise Density	200 μg/$\sqrt{\text{Hz}}$	
3dB Bandwidth	DC to 6 kHz	Maximum
3-axis magnetometer, Honeywell HMC2003 module		
Dynamic range	± 2 gauss	
Sensitivity nonlinearity	0.5% FS	from the best fit line
Resolution	40 μgauss	
Bandwidth	1 kHz	

axes are provided by three ADXL202 dual-axis accelerometers from Analog Devices. A three-axis magnetometer module HMC2003 from Honeywell Solid State Electronics Center (SSEC) is employed to obtain absolute orientation angles with respect to the Earth by sensing Earth's magnetic field. Table 7 displays the detail specification of the inertial sensors. A GPS receiver (Motorola OnCore M12) has been used to provide absolute position of the UAV in the Earth-fixed Earth-centered (EFEC) coordinate frame. The output data of the GPS sensor is directly connected to a serial port on the micro-controller using the standard NMEA format or Motorola's native binary format at a rate of 1 Hz.

A.2.3 Other Sensors

An MPXV5004D differential pressure sensor that can measure pressures up to 3.92 kPa was used in conjunction with a custom-made pitot-tube, attached under the left

Table 8: Specifications of the autopilot sensors.

Sensors	Range	Resolution	1-σ noise
Accelerometer	± 2 g	0.004 g	0.025 g
Rate gyro	± 150 °/sec	0.1 °/sec	0.4 °/sec
Magnetometer	± 2 gauss	1.22 mgauss	4 mgauss
Absolute pressure	Above sea level	2.75 m	3 m
Differential pressure	79.2 m/sec	1.40 m/sec	1.5 m/sec
Servo Position	± 60 deg	0.5 deg	

wing to obtain the airspeed. The altitude of the airplane is obtained from the pressure differential referred by the ground level, as it is measured during flight via an MPXAZ4115 pressure sensor.

Engine thrust can be approximately calculated from the knowledge of the engine RPM. The engine RPM is measured by attaching two very small magnets ($1/4''$ diameter) on the back plate of the spinner, and by using a non-contact hall-effect sensor that is fixed on the cowling of the airplane. The hall sensor generates electrical pulses whenever the magnet passes in front of it as the propeller spins. By measuring the time interval between each pulse the micro-controller can calculate the engine/propeller speed with a resolution of 1 rpm.

The airplane's control surfaces are actuated with the help of a series of DC servo motors. To obtain command input information for model identification purposes we should have accurate knowledge of the deflection angles of all the aerodynamic surfaces (elevator, rudder, ailerons) as well as the throttle setting. These are obtained by measuring the voltage of the potentiometer connected by mechanical link to each DC servo motor. This approach allows to measure the control surfaces deflections with a resolution of 0.5 [deg]. Details from the calibration of potentiometers for accurate deflection angles are given in Section A.4.3.

Table 8 summarizes the specifications, operational range, resolution, and noise performance of the autopilot sensors.

A.2.4 Communication Modem

The UAV has two main remote communication links (a third, independent link which is used to provide live video is not described here). The first link (RF band) uses the standard communication channel between the remote control (Futaba) and the airplane. The second link provides the main data communication backbone between the airplane and the ground station (see Section A.3). These two links are kept completely separate for safety reasons. A Spectra 910 wireless modem was utilized to set up a data communication link between the autopilot and the ground station. The Spectra 910 operates in the license-exempt 900 MHz frequency band utilizing frequency-hopping spread-spectrum, and is capable of providing reliable wireless data transfer up to distance of 25 miles LOS under ideal conditions (at maximum transmitting power). The interface with the micro-controller is achieved via a standard RS-232 serial communication at a maximum baud rate of 115200 bps.

A.2.5 Servo Motor Control

The micro-controller has four independent PWM outputs that generate reference command to the motors in pulse form with a varying pulse width according to the desired position. The frequency of the pulse was identified by 75 [Hz], and the duty-ratio (the ratio between [On] time versus [Off] time of the pulse) changes from 5% to 15% for the maximum allowable positions in positive and negative direction, respectively.

To achieve seamless integration (as well as switching back and forth) between autopilot and remote control action the native signal commands from the R/C receiver are merged with the PWM generated output from the micro-controller using a multiplexer. Switching of the multiplexer is being toggled by the remote pilot using a switch on the Futaba transmitter.

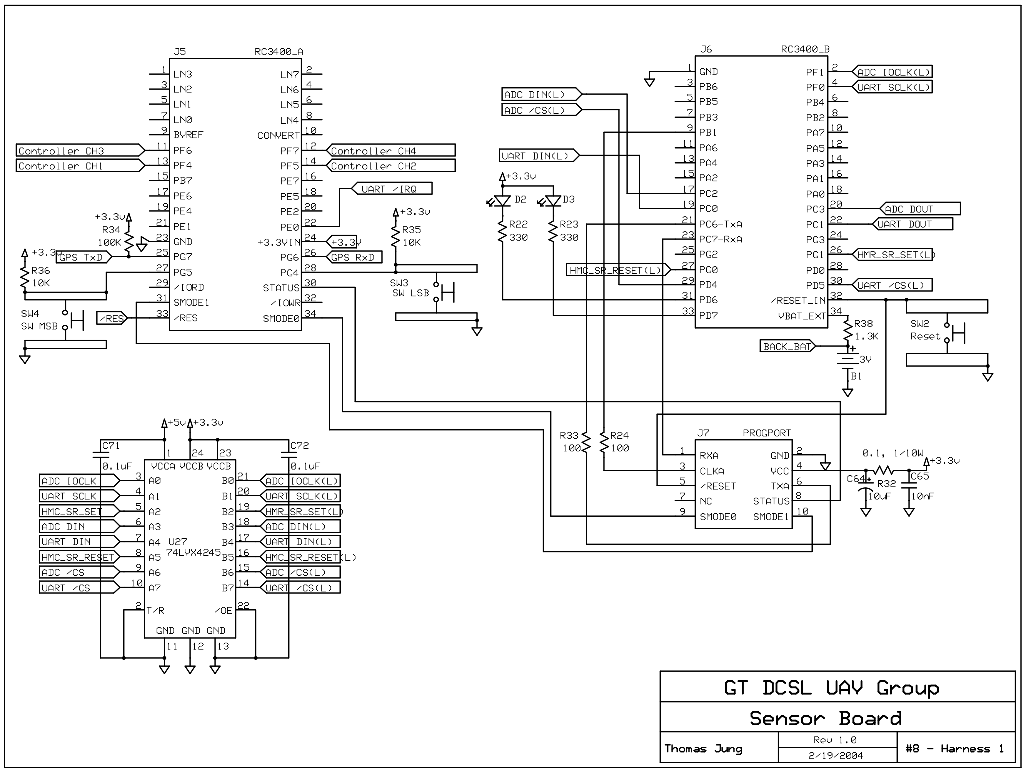
A.2.6 Schematics

In this section, detail electronic designs of the autopilot are presented. The RCM-3400 micro-controller module is interfaced through 64 pins (34 pins$\times$2), as shown in Fig. 54(a). The micro-controller has seven ports (PA-PG) which are associated with various functionality of the micro-controller, allowing the interface to external peripherals. The autopilot adopts three independent drop-down power regulators for supplying 3.0V, 3.3V, and 5.0V, as shown in Fig. 54(b). Figures 55-56 show the detail interface circuits for the inertial sensors. The interface circuits for two pressure sensors are shown in Fig. 57(a), and the serial interface for both the M12 GPS module and the Spectra 910 is depicted in Fig. 57(b).

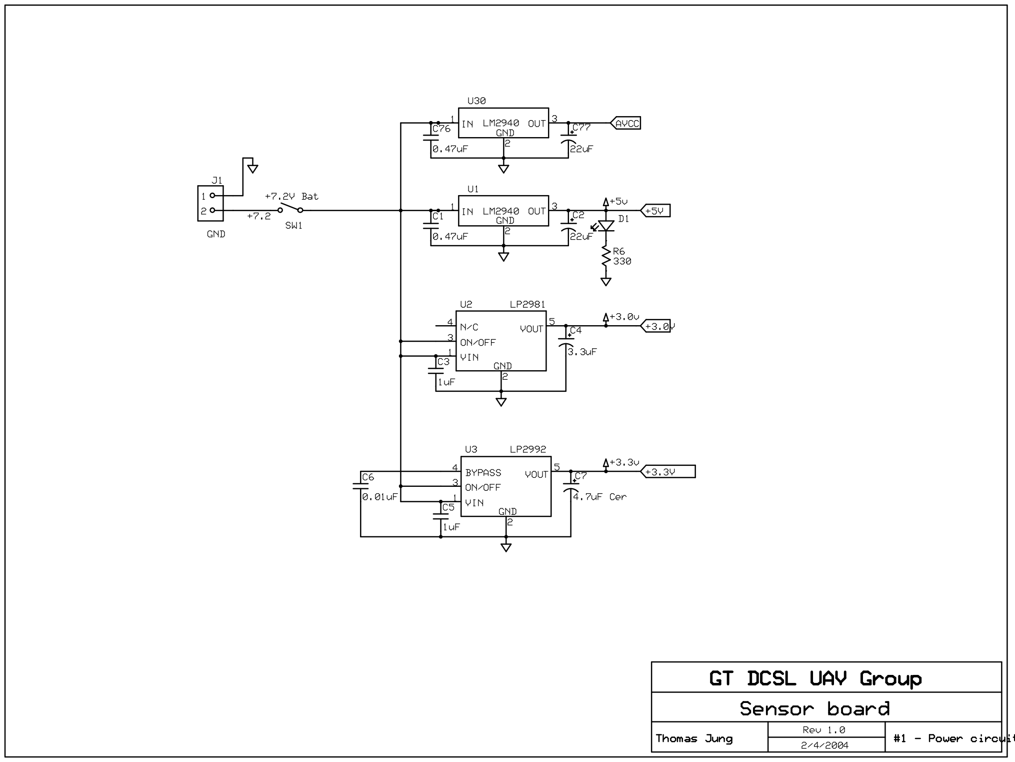
Figure 58(b) shows the schematic diagram for the control switching capability between the autopilot and the remote pilot stick command using a multiplexer.

A.3 Ground Station

The ground station consists of a laptop computer with a wireless communication modem. The laptop runs a Windows-based Graphical User Interface (GUI) program developed in-house, shown in Fig. 59. The ground station program provides real-time flight information by displaying all relevant system parameters, sensor readings, etc. A graphical dashboard representing a virtual horizon, altitude and speed has been adopted in the GUI panel to show all information graphically. A map of the area of the UAV's operation can be overlaid on the map panel in order to provide the user with the navigational details of the UAV via GPS data. The ground station program is also capable of coordinating the autonomous flight of the airplane by providing high level navigation control command via way-points on the map specified by the operator.

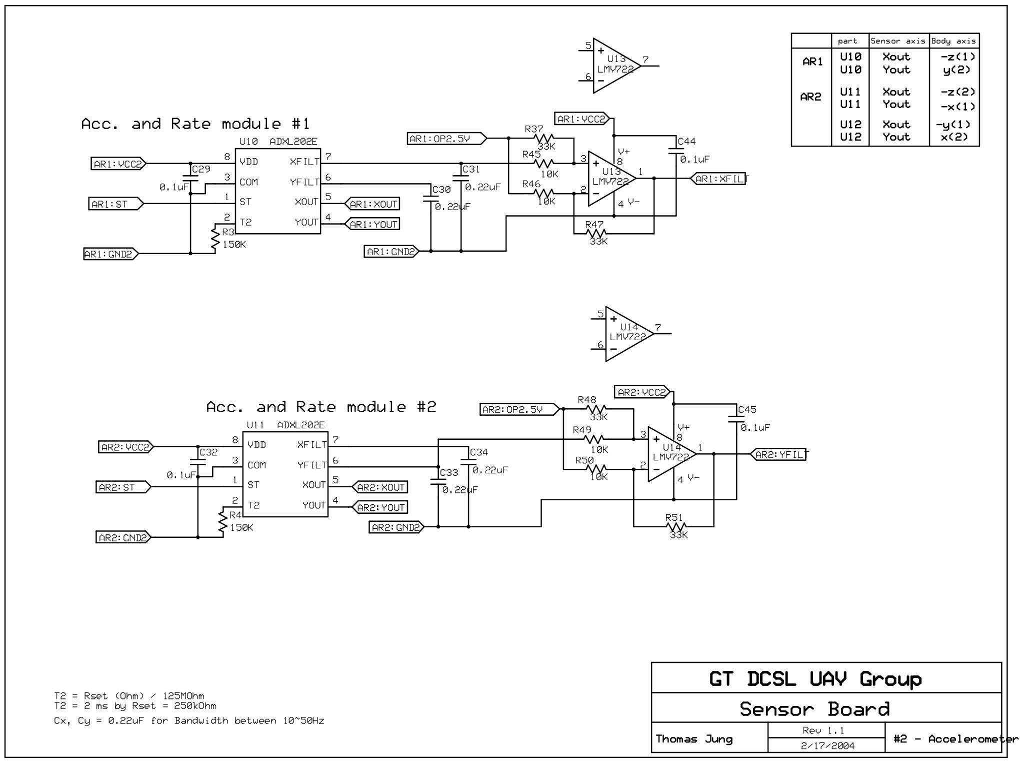


(a) Micro-controller interface circuits.



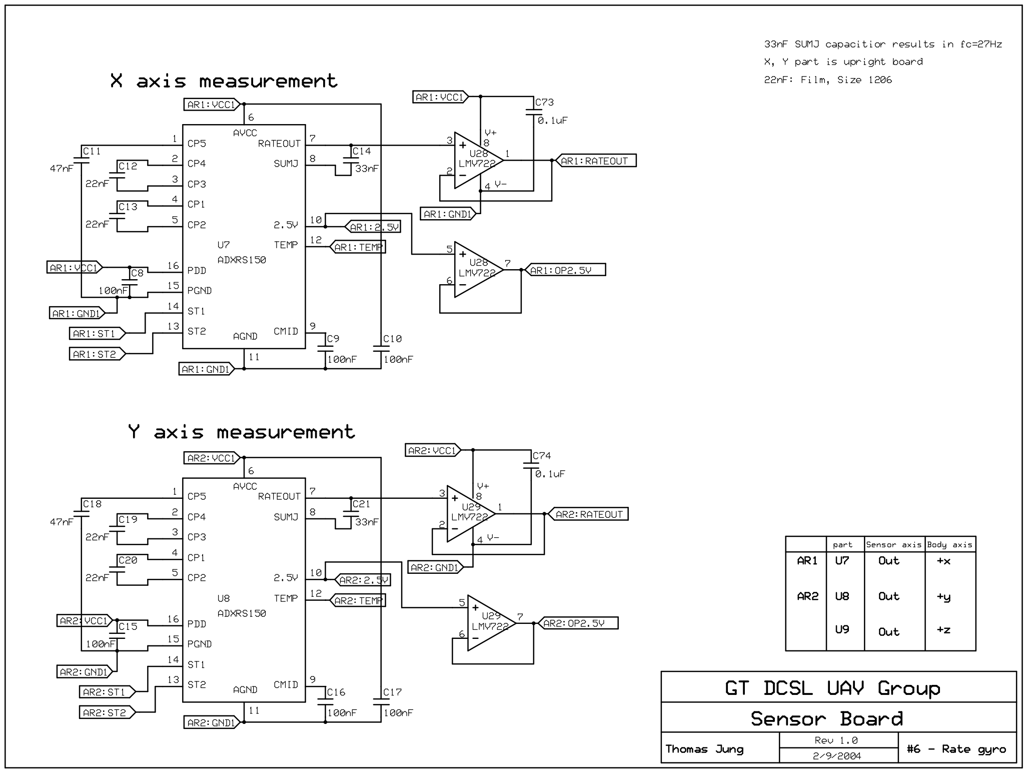
(b) Power regulators circuits.

Figure 54: Schematic diagrams of the designed autopilot: Power circuit and the micro-controller interface.



	part	Sensor axis	Body axis
AR1	U10	Xout	-z(1)
	U10	Yout	y(2)
AR2	U11	Xout	-z(2)
	U11	Yout	-x(1)
	U12	Xout	-y(1)
	U12	Yout	x(2)

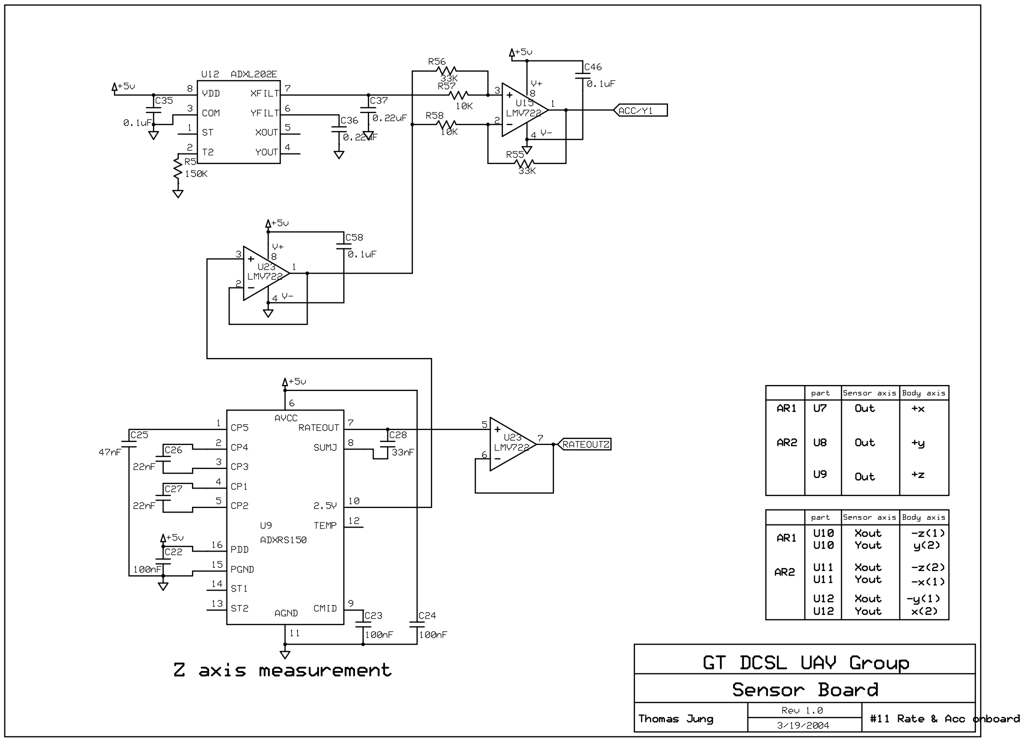
(a) ADXL202 interface circuits.



	part	Sensor axis	Body axis
AR1	U7	Out	+x
AR2	U8	Out	+y
	U9	Out	+z

(b) ADXRS150 interface circuits.

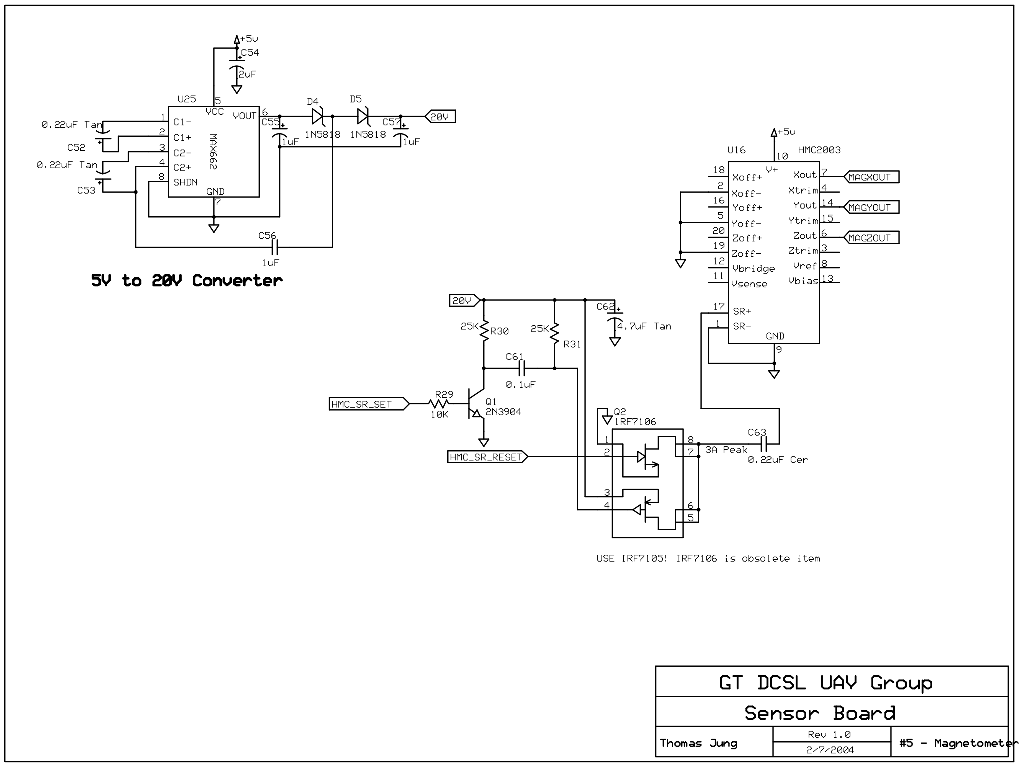
Figure 55: Schematic diagrams of the designed autopilot: The rate sensors and the accelerometers interface.



	part	Sensor axis	Body axis
AR1	U7	Out	+x
AR2	U8	Out	+y
	U9	Out	+z

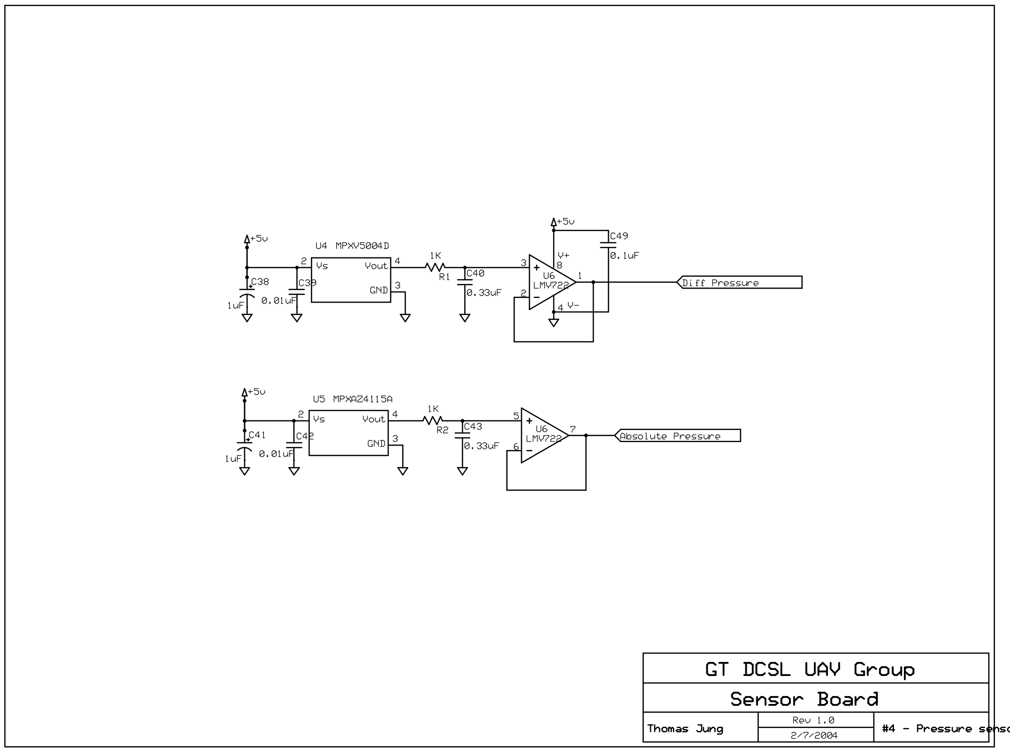
	part	Sensor axis	Body axis
AR1	U10	Xout	-z(1)
	U10	Yout	y(2)
AR2	U11	Xout	-z(2)
	U11	Yout	-x(1)
	U12	Xout	-y(1)
	U12	Yout	x(2)

(a) ADXL202 and ADXRS150 interface circuits.

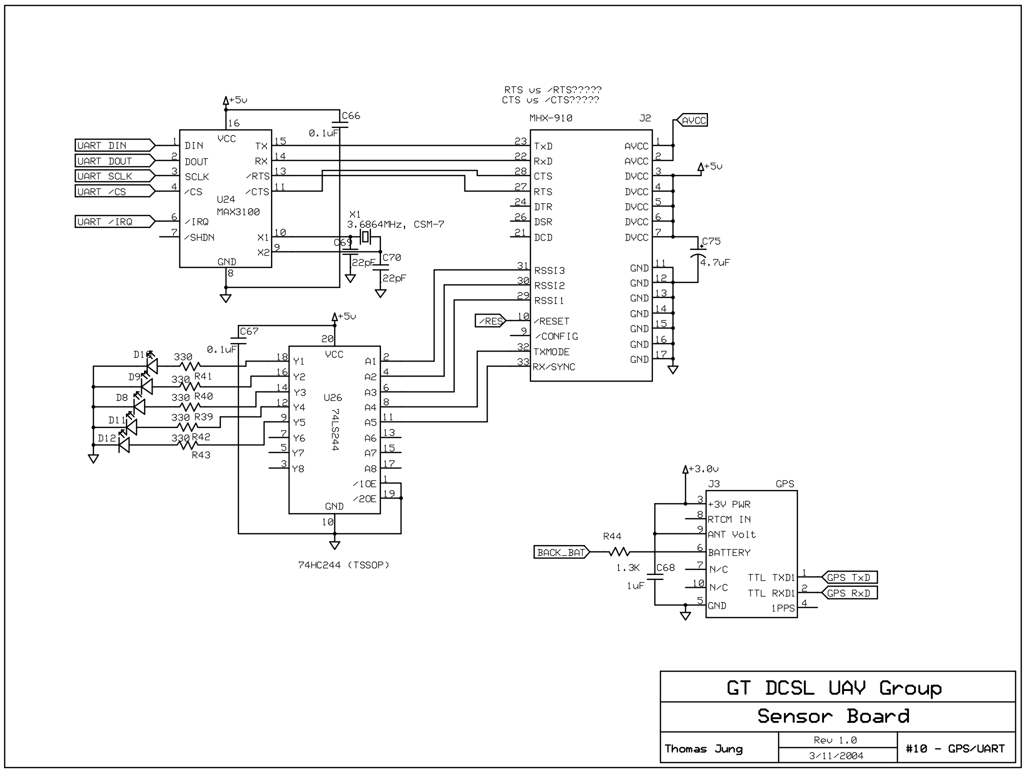


(b) HMC2003 interface circuits.

Figure 56: Schematic diagrams of the designed autopilot: Z-axis rate and acceleration, 3-axis magnetometer interface.

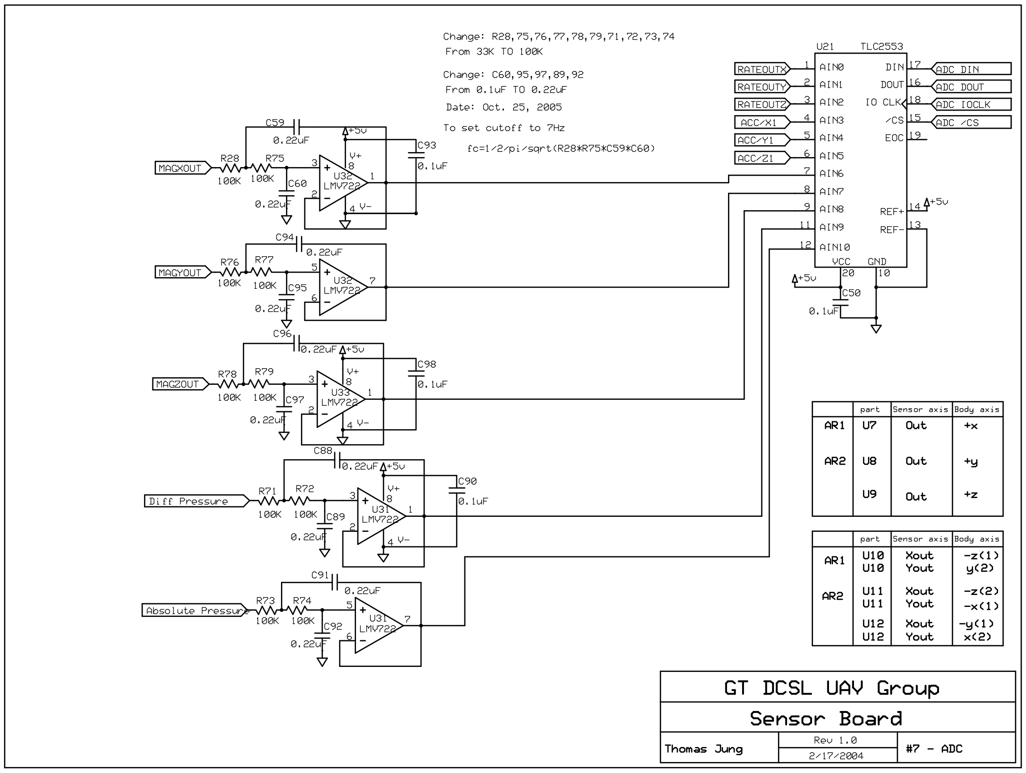


(a) MPXV5004D and MPXAZ4115 pressure sensor interface circuits.



(b) M12 oncore GPS receiver and the Spectra 910 interface circuits.

Figure 57: Schematic diagrams of the designed autopilot: Pressure sensors and the GPS interface.


(a) Analog-to-Digital converter interface circuits.

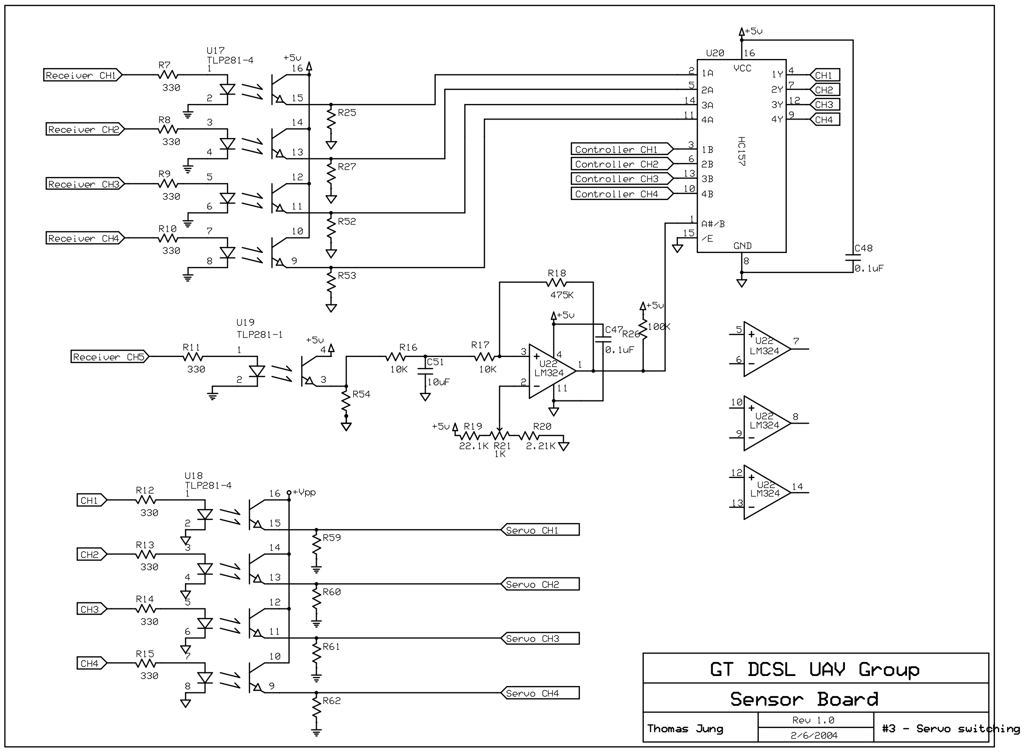
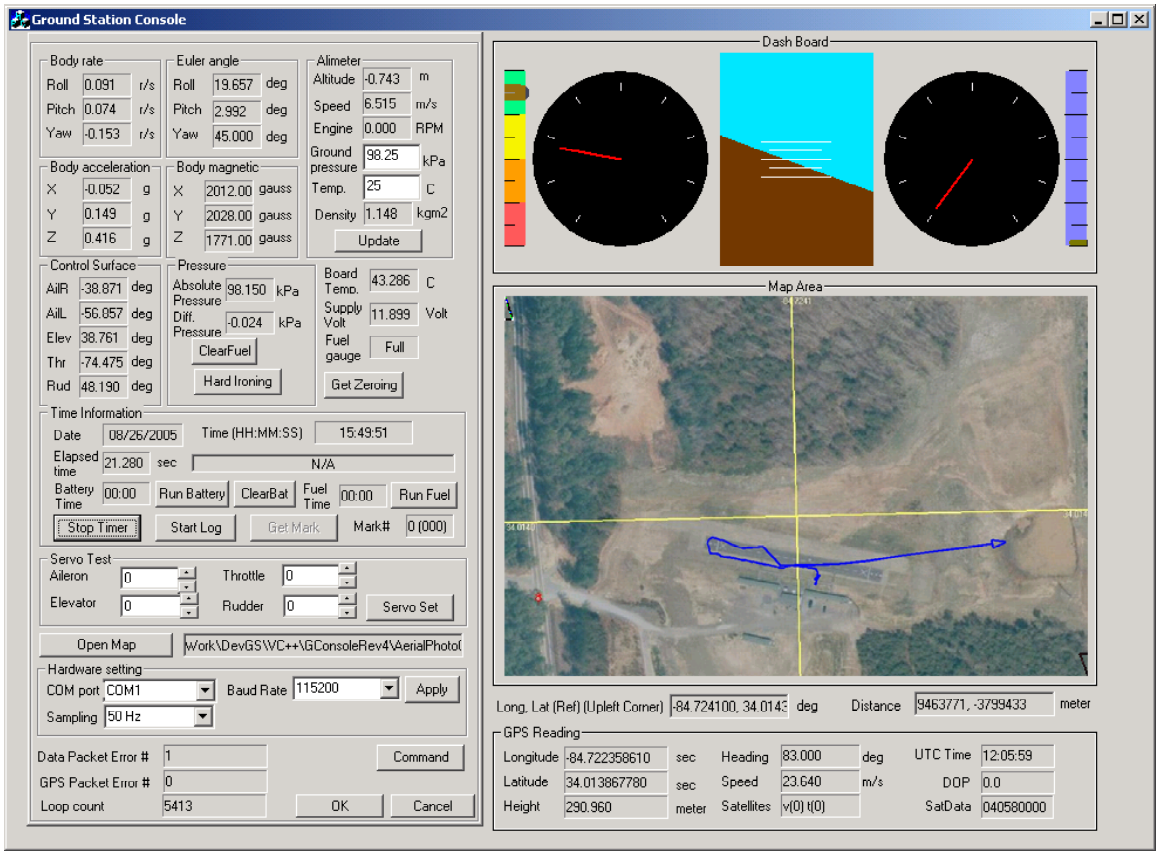

(b) Servo switching circuits.

Figure 58: Schematic diagrams of the designed autopilot: A-to-D converter and the servo switching interface.



Body rate

Roll	0.091	r/s
Pitch	0.074	r/s
Yaw	-0.153	r/s

Euler angle

Roll	19.657	deg
Pitch	2.992	deg
Yaw	45.000	deg

Alimeter

Altitude	-0.743	m
Speed	6.515	m/s
Engine	0.000	RPM
Ground pressure	98.25	kPa
Temp.	25	C
Density	1.148	kgm2

Body acceleration

X	-0.052	g
Y	0.149	g
Z	0.416	g

Body magnetic

X	2012.00	gauss
Y	2028.00	gauss
Z	1771.00	gauss

Control Surface

AilR	-38.871	deg
AilL	-56.857	deg
Elev	38.761	deg
Thr	-74.475	deg
Rud	48.190	deg

Pressure

Absolute Pressure	98.150	kPa
Diff. Pressure	-0.024	kPa

Board Temp.	43.286	C
Supply Volt	11.899	Volt
Fuel gauge	Full	

Time Information

Date 08/26/2005 Time (HH:MM:SS) 15:49:51
Elapsed time 21.280 sec N/A
Battery Time 00:00 Fuel Time 00:00
Mark# 0 (000)

Servo Test

Aileron 0 Throttle 0
Elevator 0 Rudder 0

Open Map Work\DevGS\VC++\GConsoleRev4\AerialPhoto(

Hardware setting

COM port COM1 Baud Rate 115200
Sampling 50 Hz

Data Packet Error #	1
GPS Packet Error #	0
Loop count	5413

Long, Lat (Ref) (Upleft Corner) -84.724100, 34.0143 deg Distance 9463771, -3799433 meter

GPS Reading

Longitude	-84.722358610	sec	Heading	83.000	deg	UTC Time	12:05:59
Latitude	34.013867780	sec	Speed	23.640	m/s	DOP	0.0
Height	290.960	meter	Satellites	v(0) t(0)		SatData	040580000

Figure 59: The ground station GUI program.

A.4 Hardware Evaluation/Calibration

A.4.1 Inertial Sensor Calibration

A static test was performed to determine the initial biases and the static noise level of each sensor. The sensor outputs were measured while the autopilot was completely stationary, and by performing a statistical analysis on the recorded data over time, the initial biases and 1-σ noise levels were obtained. The static noise characteristics of all sensor correlated favorably with the specification provided by the sensor manufacturer, and summarized in Table 8.

The actual scale factors for each accelerometer can be found by taking two measurements with the accelerometer's measurement axis pointed directly towards $(+1g)$ or opposite (-1g) to the Earth. The scale factors can then be found from the difference of two measurements factored by the known gravity change $(2g)$.

The scale factors of the angular velocity sensors were found by placing the autopilot on a three-axis rotational platform [57]. The platform is equipped with a high-performance angular rate gyro and an inertial measurement unit (IMU) which is capable of measuring angular velocities and linear accelerations in all three-axes with an accuracy better than 0.03 [deg/sec], and 0.001 [g], respectively. After the autopilot was securely mounted on the platform, the platform was set in motion while both signals from the autopilot and from the high-performance platform sensors were recorded. The rate sensor outputs were then compared, and a least square fit was employed to find the best scale factor of the autopilot rate sensors. Figure 60 shows the result from this approach. In addition, Fig. 61 shows the validation of estimated scale factors and biases for the accelerometers on this platform. From the plots, it is asserted that the correlation between the two sets of signals is satisfactory for our purposes.

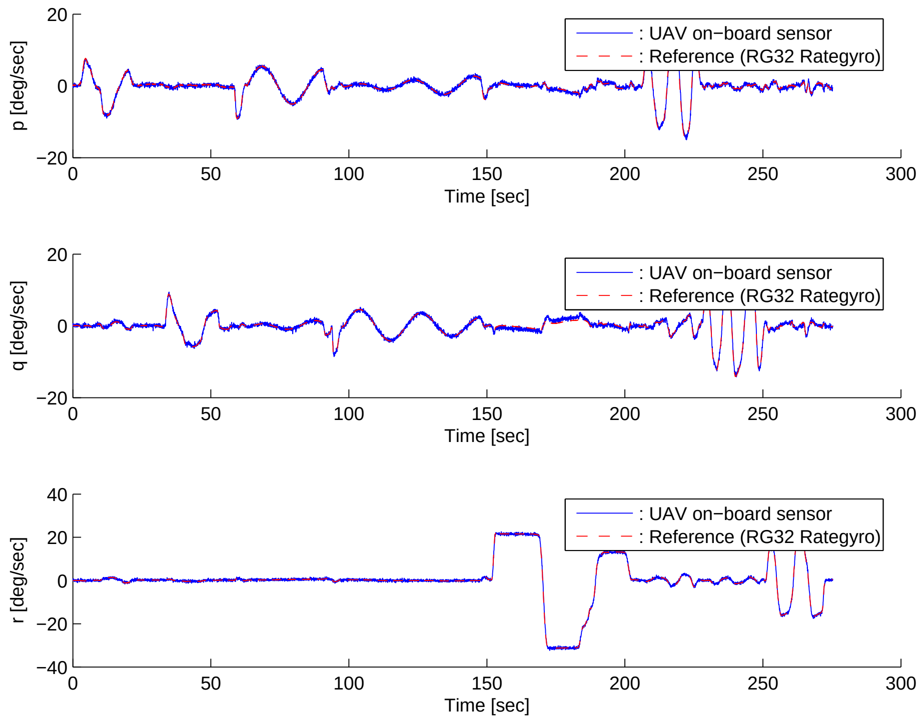


Figure 60: Angular rate calibration results.

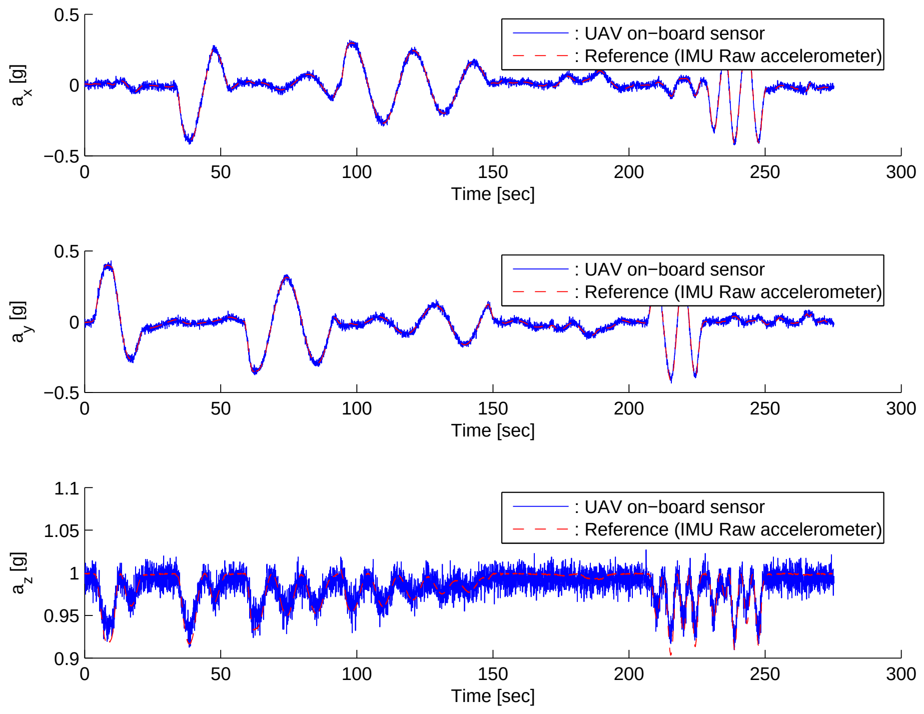


Figure 61: Accelerometer calibration results.

A.4.2 Magnetometer Calibration

The magnetometer can provide absolute orientation and it is not affected by motion constraints. On the other hand, it is susceptible to magnetic disturbances from nearby permanent magnets or ferrous materials that locally distort the Earth magnetic field. Magnetic distortion can be categorized as hard iron or soft iron effects [27]. These effects become evident as the magnetometer is rotated in the horizontal plane. By plotting the two measured signals in the body-axis frame, the hard iron distortion appears as a shift of the origin in the phase plot (X_h vs. Y_h), whereas soft iron effects appear as a distortion of a circle to an ellipse in the X_h vs. Y_h plane. Figure 62 shows the hard and soft iron distortions and the compensated magnetometer outputs.

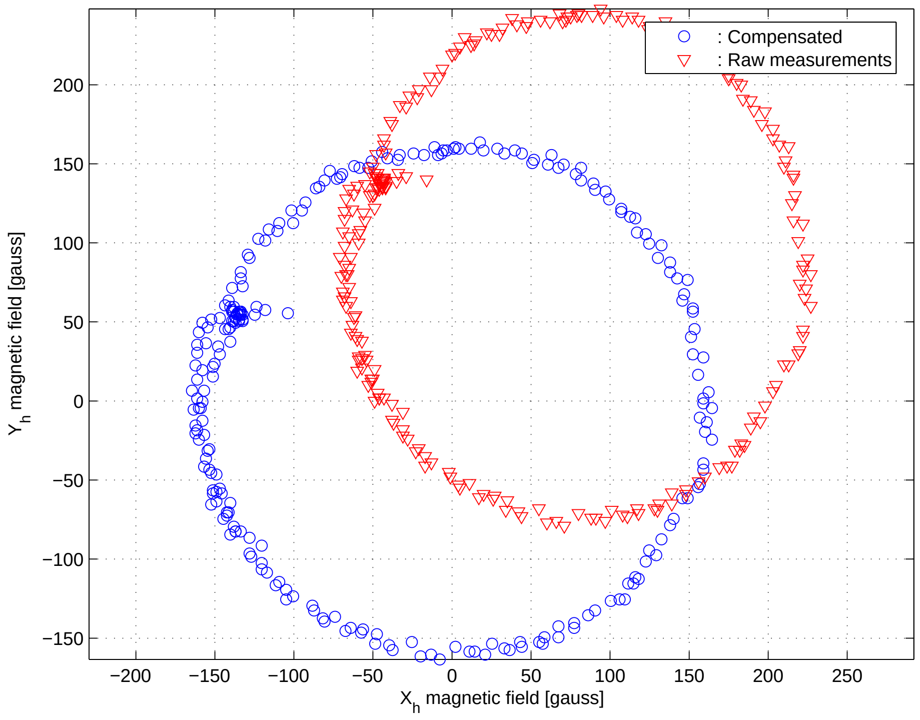


Figure 62: Effect of hard and soft iron disturbances and compensated magnetometer measurements.

Table 9: Maximum deflection angles for each control surface.

Control surface	+Range	-Range
Aileron δ_a	21.8 deg	-21 deg
Elevator δ_e	28 deg	-29.6 deg
Rudder δ_r	16 deg	-20.7 deg
Throttle δ_t	Full open (1)	Full closed (0)

A.4.3 Control Surface Deflection Calibration

One can get the actual angle of the servo motor from the corresponding voltage level of the external potentiometer linked to the DC servo motors. Therefore, the deflection angles for each control surface can be determined from each servo's position. An angle meter was used to set the actual deflection angles by a specific amount, while measuring the voltage output from the potentiometer. After several commands were applied to the servo motor, the conversion factors from potentiometer voltage level to actual deflection angle for each control surface were found. In addition, the maximum allowable deflections for each control surface were determined experimentally as well. The results are summarized in Table 9.

A.5 Summary

We have summarized the efforts undertaken to design and build a low-cost autopilot for a small UAV. The focus from the very start has been to design and assemble as much of the hardware and electronics in house as possible. This choice was opted for in order to achieve the full accessibility to the entire system, so the developed autopilot satisfies not only the stringent size and weight constraints to fit in a small UAV but also it can provide full functionality for an autonomous UAV operation.

APPENDIX B

INERTIAL ATTITUDE AND POSITION REFERENCE SYSTEM DEVELOPMENTS

B.1 Attitude estimation

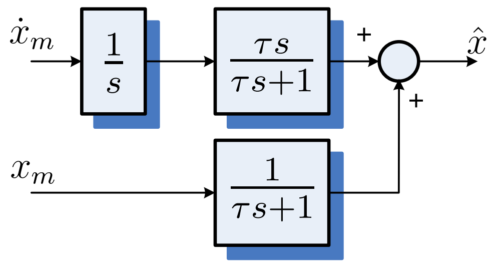
The sensors involved in a strapdown attitude and heading reference system are rate gyros, accelerometers and magnetometers. These sensors measure the three-axis angular rates, three-axis apparent acceleration (gravity minus inertial acceleration), and Earth's magnetic field with respect to the body frame. In order to obtain the best estimate of the attitude angles from the available sensors, it is imperative to blend these measurements in a seamless manner by taking into account the different signal specifications for each sensor.

B.1.1 Complementary filter

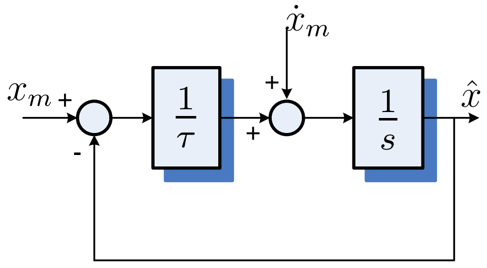
Complementary filters have been widely used to combine two independent noisy measurements of the same signal, where each measurement is corrupted by different types of spectral noise[20]. The filter provides an estimate of the true signal by employing two complementary high-pass and low-pass filters. Figure 63(a) shows the case of using a complementary filter to obtain an estimate $\hat{x}(t)$ of $x(t)$ from the two measurements $x_m(t)$ and $\dot{x}_m(t)$. Notice that $x_m(t)$ is the measurement of the signal with predominantly high-frequency noise $n_1(t)$ and $\dot{x}_m(t)$ is the measurement with low-frequency noise $n_2(t)$ as follows

$$x_m(t) = x(t) + n_1(t) \quad \text{and} \quad \dot{x}_m(t) = \dot{x}(t) + n_2(t). \tag{125}$$

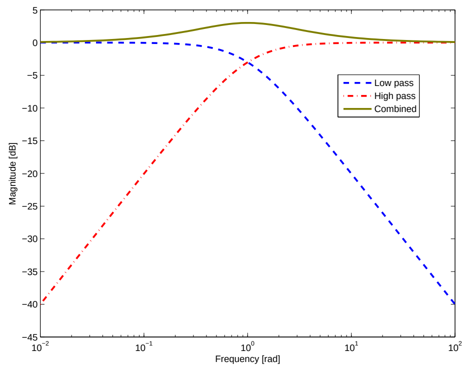
From Fig. 63(a), it is apparent that the Laplace transform of the estimate can be written as



(a) Direct complementary filter



(b) Indirect complementary filter



(c) Frequency magnitude plot for the output of a complementary filter

Figure 63: Two different schemes for the implementation of the complementary filter.

$$\hat{X}(s) = \underbrace{\frac{1}{\tau s + 1} X(s) + \frac{\tau s}{\tau s + 1} X(s)}_{\text{Signal terms}} + \underbrace{\frac{1}{\tau s + 1} N_1(s) + \frac{\tau s}{\tau s + 1}\left(\frac{1}{s} N_2(s)\right)}_{\text{Noise terms}} \tag{126}$$

The noise terms in both channels are effectively suppressed by the first order low- and high- pass filter with the time constant τ. The frequency response plot shown in Fig. 63(c) illustrates the contribution of each frequency channel to the output, where the cutoff frequency is chosen as 1 [rad/sec]. The time constant τ is selected according to the noise characteristics of each channel such that the estimate $\hat{x}$ is obtained by integrating $\dot{x}_m$ over frequencies $\omega \gg 1/\tau$, whereas for frequencies $\omega \ll 1/\tau$, $\hat{x}$ tracks x_m. At frequencies near $1/\tau$ the estimated output $\hat{x}$ is a combination of the two channels, which appears as a hump in Fig. 63(c). The estimate approximates the true signal faithfully over most of the frequency range.

In order to implement a complementary filter on a micro-controller, a discrete version of the high-pass and low-pass filters should be written in software by taking into consideration the sampling period of the micro-controller. The filter coefficients of the discrete filters are related to the cutoff frequency τ, which forces the user to recalculate the coefficients of both filters, if necessary. Instead, an alternative form of the complementary filter can be used as depicted in Fig. 63(b). The filter transfer function remains the same as in Eq. (126) but the feedback structure of the filter simplifies the filter implementation on a micro-controller. It also allows easy tuning for acceptable performance when low cost sensors are used. In addition, this feedback structure can be easily adapted to deal with multiple measurements. Figure 64 illustrates the case when using two low frequency channels. A tuning parameter $\alpha_k \in [0, 1]$ sets the relative weight between the signals x_{m_1} and x_{m_2}. By choosing the more reliable measurements the filter takes advantage of multiple measurements to calculate the best estimate.

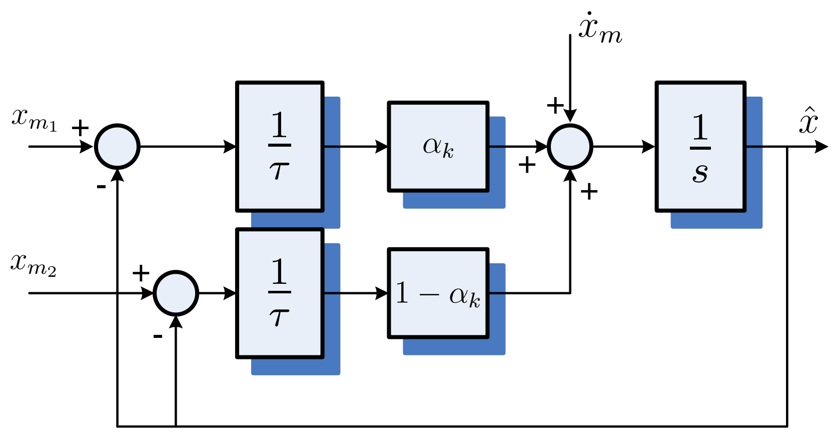


Figure 64: Multiple measurements augmentation in the indirect complementary filter.

B.1.2 Pitch and heading angle estimation

The low frequency dominant pitch angle is directly calculated from accelerometer outputs because the accelerometer is able to measure the gravity vector minus the inertial acceleration (the apparent acceleration, $\vec{g}-\vec{a}_I$) with respect to the body axes. In steady-state flight conditions, the accelerometers output mostly the gravity vector since the inertial acceleration is negligible at the steady state. Then the pitch angle is calculated from the accelerometer output $\mathbf{a} = [a_x \; a_y \; a_z]^\mathsf{T}$ as follows

$$\theta_L = -\sin^{-1}\left(\frac{a_x}{g}\right). \tag{127}$$

The low frequency dominant heading angle is determined by two different sources: the GPS sensor and the magnetometer. The GPS sensor used in this research provides an absolute heading information ψ_{GPS} at a low rate (1 Hz), the output of the three-axis magnetometer $\mathbf{m} = [m_x \; m_y \; m_z]^\mathsf{T}$ with respect to the body axes provides a heading measurement ψ_L at a much higher rate according to the following relationship[26]

$$\psi_L = \begin{cases} \pi - \tan^{-1}(\overline{m}_y/\overline{m}_x) & \text{if } \overline{m}_x < 0, \\ 2\pi - \tan^{-1}(\overline{m}_y/\overline{m}_x) & \text{if } \overline{m}_x > 0,\, \overline{m}_y > 0, \\ -\tan^{-1}(\overline{m}_y/\overline{m}_x) & \text{if } \overline{m}_x > 0,\, \overline{m}_y < 0, \\ \pi/2 & \text{if } \overline{m}_x = 0,\, \overline{m}_y < 0, \\ 3\pi/2 & \text{if } \overline{m}_x = 0,\, \overline{m}_y > 0, \end{cases} \tag{128}$$

where $\overline{m}_x$ and $\overline{m}_y$ are the projected magnetic field components on the horizontal plane that can be calculated by transforming $\mathbf{m}$ through the rotation matrix $\mathcal{C}(\phi,\theta) \triangleq \big(\mathcal{C}_1(\phi)\mathcal{C}_2(\theta)\big)^{\mathsf{T}}$. If the pitch angle θ and the roll angle ϕ are not available, their estimates $\hat{\theta}$ and $\hat{\phi}$ can be used instead, to yield

$$\begin{aligned} \overline{m}_x &= m_x \cos\hat{\theta} + m_y \sin\hat{\phi}\sin\hat{\theta} + m_z \cos\hat{\phi}\sin\hat{\theta} \\ \overline{m}_y &= m_y \cos\hat{\phi} - m_z \sin\hat{\phi} \end{aligned} \tag{129}$$

The high frequency dominant pitch and heading angles are inferred from the attitude kinematics equations,

$$\begin{aligned} \dot{\theta} &= q\cos\hat{\phi} - r\sin\hat{\phi} \\ \dot{\psi} &= (q\sin\hat{\phi} + r\cos\hat{\phi})/\cos\hat{\theta} \end{aligned} \tag{130}$$

where, $\boldsymbol{\omega} = [p\ q\ r]^{\mathsf{T}}$ is the onboard rate gyro measurement.

Figure 65 illustrates the block diagram for the combined complementary filters for pitch and heading angle estimation. As discussed earlier, the filters can be tuned for acceptable performance via the parameters τ_θ and τ_ψ for pitch and heading angle, respectively. The relative weight for the heading angle between the magnetometer and the GPS sensor is imposed by the parameter $\alpha_\psi \in [0\ ,1]$ in order to put more emphasis on the measurement that seems to be close to the true heading. A detailed description regarding the adaptive tuning of α_ψ is given later.

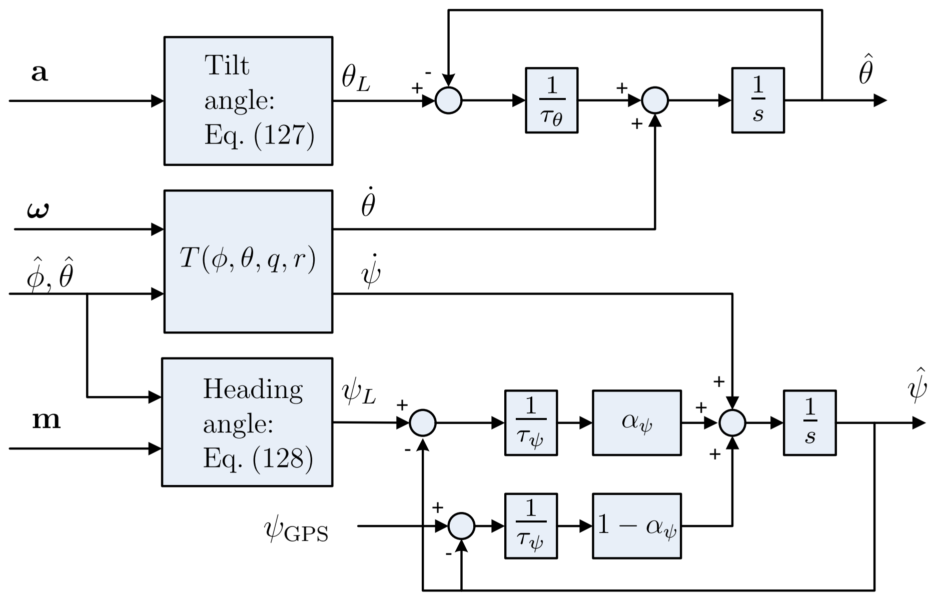


Figure 65: Entire complementary filter setup for pitch and heading angles.

B.1.3 Roll angle estimation

The roll angle can also be estimated from a complementary filter using high frequency dominant information for $\dot{\phi}$ via the attitude kinematics and low frequency dominant information from the kinematic relationship of the airplane at a banked condition. As illustrated in Fig. 66, if an airplane is in a purely banked, coordinated turn con-

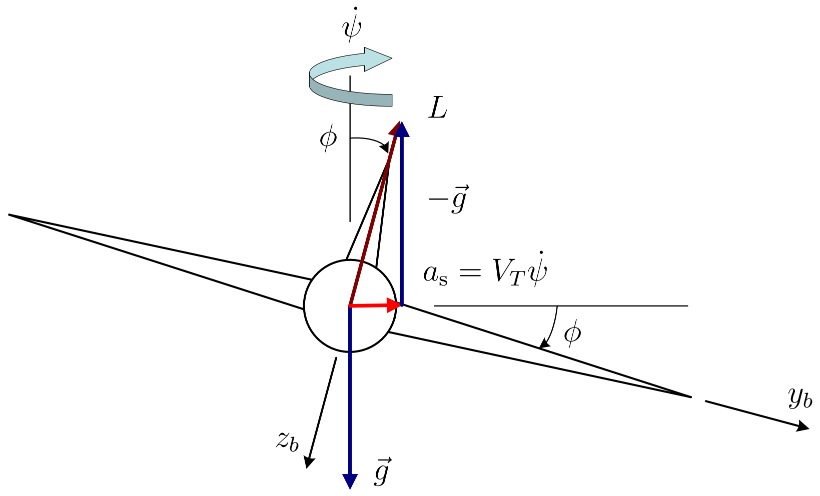


Figure 66: Kinematic relation at a truly banked turn condition.

dition (no side acceleration along body y-axis), then the roll angle is approximately computed by the following relation assuming no wind[41],

$$\sin\phi = \frac{\dot{\psi} V_T}{g} \tag{131}$$

where V_T is the flight speed and g is the gravitational acceleration. This equation can be further approximated using $\sin\phi \approx \phi$ and $\dot{\psi} \approx r$ as

$$\phi = \frac{r V_T}{g}. \tag{132}$$

Then the low frequency dominant roll angle is approximated from Eq. (132) with the yaw rate from the yaw gyro and the flight speed from the pitot tube. In reality, the estimate from Eq. (132) tends to be biased since it utilizes the direct gyro output which is vulnerable to drift. Over a long period of time the estimate will deviate owing to this yaw rate bias[105]. To compensate for this bias, a Kalman filter was designed as follows.

In general, the fast roll dynamics of the airplane allow to use a linear approximation for the roll kinematics $\dot{\phi} = p$. The roll rate gyro measurement p_m is assumed to be corrupted by the roll rate bias p_b as well as measurement noise η_p,

$$p_m = p + p_b + \eta_p.$$

The biases for both roll and yaw gyros are modeled as random walk processes driven by Gaussian white noise processes. It follows that the equations of the filter dynamics are given by

$$\begin{aligned} \dot{\phi} &= p_m - p_b - \eta_p, \\ \dot{p}_b &= \epsilon_p, \\ \dot{r}_b &= \epsilon_r. \end{aligned} \tag{133}$$

The measurement model of the Kalman filter includes the yaw gyro output and the rate of heading angle change induced from the heading angle measurement by the

GPS sensor. The yaw rate gyro measurement r_m contains a bias r_b and measurement noise η_r

$$r_m = r + r_b + \eta_r. \tag{134}$$

Hence the yaw rate measurement is related to the roll angle, using Eq. (132), as follows,

$$r_m = \frac{g}{V_T}\phi + r_b + \eta_r. \tag{135}$$

On the other hand, because the GPS sensor provides the heading angle at one second interval, the rate of heading angle change $\dot{\psi}$ is calculated through numerical differentiation. It follows from Eq. (131) that

$$\dot{\psi}_m = \frac{g\cos\hat{\phi}}{V_T^*}\phi + \eta_{\dot{\psi}}, \tag{136}$$

where, V_T^* is the flight speed obtained from the GPS sensor, $\eta_{\dot{\psi}}$ is the measurement noise, and $\hat{\phi}$ is the current roll angle estimate. Equations (135) and (136) become the measurement model for roll angle Kalman filter. The Kalman filter is implemented in a discrete format on the micro-controller using a sampling period Δt:

- **Time update**
  - Project ahead

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \Phi_k \hat{\mathbf{x}}_{k-1} + [\ \Delta t p_m^k \quad 0 \quad 0\ ]^\mathsf{T}, \\ \mathbf{P}_k^- &= \Phi_k \mathbf{P}_{k-1} \Phi_k^\mathsf{T} + \mathbf{Q}_k, \end{aligned} \tag{137}$$

where,

$$\Phi_k = \begin{bmatrix} 1 & -\Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\phi}_k \\ \hat{p}_{b_k} \\ \hat{r}_{b_k} \end{bmatrix}.$$

- **Measurement update**

– Compute Kalman gain

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} \big(\mathbf{H}_k^\star \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} + \mathbf{R}_k^\star\big)^{-1}, \tag{138}$$

– Update estimate with measurements

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k^\star - \mathbf{H}_k^\star \hat{\mathbf{x}}_k^-), \tag{139}$$

– Compute error covariance for updated estimate

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k^\star)\mathbf{P}_k^-, \tag{140}$$

where, $\star = s,\ f$.

Note that the measurement update for GPS is done once every second at which the new $\dot{\psi}_m$ becomes available, whereas the measurement update for the yaw rate gyro occurs at a high update rate. These two measurement updates are coordinated such that each update is completed whenever the corresponding measurement becomes available: a fast update for r_{m_k} and a slow update for $\dot{\psi}_{m_k}$ as follows,

- Fast update

$$\mathbf{z}_k^f = r_{m_k}, \qquad \mathbf{H}_k^f = \begin{bmatrix} \dfrac{g}{V_T} & 0 & 1 \end{bmatrix}, \tag{141}$$

- Slow update

$$\mathbf{z}_k^s = \dot{\psi}_{m_k}, \qquad \mathbf{H}_k^s = \begin{bmatrix} \dfrac{g}{V_T}\cos\hat{\phi}_k^- & 0 & 1 \end{bmatrix}. \tag{142}$$

The process noise covariance matrix $\mathbf{Q}_k$ and measurement noise covariance matrix $\mathbf{R}_k$ are determined from the noise characteristics of each signal by assuming that the noise processes are uncorrelated to each other, as follows

$$\begin{aligned} \mathbf{Q}_k &= \mathrm{diag}\Big(E\big[\ \bar{\eta}_p^2 \quad \bar{\epsilon}_p^2 \quad \bar{\epsilon}_r^2\ \big]\Big), \\ \mathbf{R}_k^f &= E\big[\bar{\eta}_r^2\big], \qquad \mathbf{R}_k^s = E\big[\bar{\eta}_{\dot{\psi}}^2\big], \end{aligned} \tag{143}$$

where the overbar variables represent the discrete noise sequences at the sampling period of Δt such that they have equal noise strength as the continuous noise process, and $E[\cdot]$ calculates the mean-square noise strength.

B.1.4 Dealing with GPS latency and GPS lock

The GPS receiver employed in this research has an inherent data latency, which causes the output of the GPS sensor to be delayed by a certain amount of time. The position and velocity output from the GPS sensor at the kth time step are processed internally based on the satellite range measurements at the epoch of the $(k-1)$th time step. Combining the latency due to the internal data processing along with the communication latency, the time delay of the position output is observed to be about 0.1 [sec] and the delay of the velocity output about 1.1 [sec][98]. The GPS sensor also provides the heading angle based on the internal velocity estimate. Hence, the measurement of ψ_{GPS} is also delayed by 1.1 [sec]. Unless this delay is properly compensated, substantial errors will arise during the estimation process.

Figure 67 illustrates graphically the issue of delay. Assume that the measurement is delayed by N samples. Then the measurement $\mathbf{z}_k^*$ at $t = t_k$ represents the state of the N prior sample $\mathbf{z}_k^* = \mathbf{H}_{k-N}^{\star}\mathbf{x}_{k-N}$. Several arrangements to incorporate delays into the Kalman filter framework have been suggested in the literature[8, 79]. One approach is to simply recalculate the complete time-trajectory of the filter throughout the delayed period when a delayed measurement is received. The large memory cost for storing the intermediate states over the delayed period and the corresponding increased computational cost prevent delayed measurements from being incorporated directly in a real-time estimation algorithm. Another approach to account for delayed

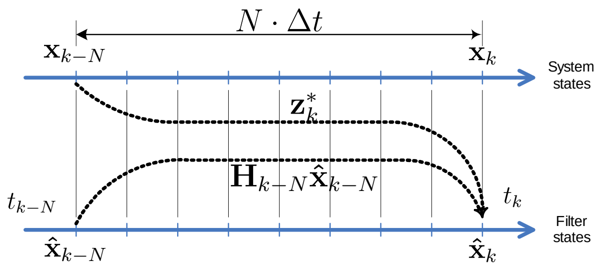


Figure 67: System with a delayed measurement due to sensor latency.

measurements is to make use of the state estimate corresponding to the delayed measurement at $t = t_{k-N}$, i.e., $\hat{\mathbf{x}}_{k-N}$ in a buffer. When the delayed measurement $\mathbf{z}_k^*$ becomes available at the time $t = t_k$, an innovation that is calculated from the delayed measurement and the delayed state estimate in the buffer is blended with the current estimate state in the standard Kalman measurement update as follows,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k^* - \mathbf{H}_{k-N}\hat{\mathbf{x}}_{k-N}). \tag{144}$$

Compared to the case of a wrong innovation calculated from the delayed measurement and the current state estimate $\hat{\mathbf{x}}_k^-$ (no delay compensation) as in Eq. (139), this approach forces the correct innovation to be used in the measurement update and then yields a better estimate. It is, however, a sub-optimal solution[79].

The estimation filters presented in the previous section assumed that the GPS measurement is always available. However, the GPS output is locked when a GPS outage occurs, failing to provide successive information. The situation gets even worse if the UAV performs aggressive motion such as a sharp turn at high speed. During such an outage, the GPS output is held at the previous valid output. Figure 68 demonstrates the real measured GPS data when the UAV performs a sharp turn by a remote pilot. The plot shows that at $t = 2711.2$ [sec] the GPS heading angle is held to the previous value of -90 [deg]. The value is kept until $t = 2714.2$ [sec] when a new (possibly valid) heading angle is provided by the GPS sensor. Because the attitude estimation algorithm presented earlier utilizes the heading angle measurement from the GPS sensor, an incorrect heading information from the GPS sensor due to a GPS outage would result in a wrong estimation of the heading angle in the complementary filter. Moreover, this would lead to a wrong estimation of the roll angle from the Kalman filter.

One possible remedy to this is to use intermitently the heading information from the magnetic compass for a short time period (i.e., during GPS outage). Even though any local magnetic distortion due to the existence of ferrous materials near the magnetic compass yields a small offset in the magnetic heading information, the magnetic

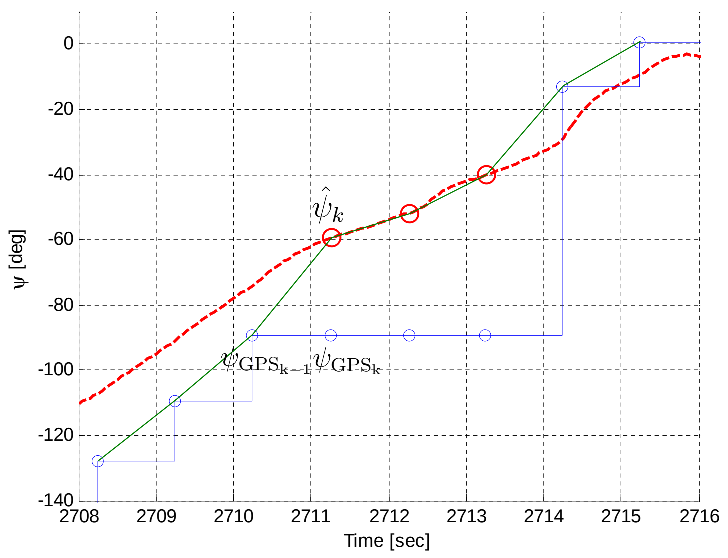


Figure 68: GPS momentary outage during an aggressive maneuver.

heading information can provide continuous heading measurement to the filtering algorithm even when a GPS outage occurs. Whenever the complementary filter detects the GPS heading angle being held constant, it first modifies the weighting parameter α_ψ to put more emphasis on the magnetic heading information. As the α_ψ approaches one, the magnetic heading information is used more exclusively in the complementary filter, while the GPS heading information is ignored. A update logic for a new weighting parameter α_ψ^* is simply given by

$$\alpha_\psi^* = \alpha_\psi + (1 - \alpha_\psi)\lambda \tag{145}$$

where, $\lambda \geq 1$ controls the rate of change of weighting parameter. Once the GPS is back to normal operation, the weighting parameter will restore the specified value for a normal operation. After the complementary filter computes the heading estimate during a GPS outage, the Kalman filter can make use of the heading estimate from the complementary filter in order to obtain the rate of change of the heading angle. Figure 68 describes the use of heading estimate from the complementary filter to compute the rate of heading angle change during a GPS outage. Notice that at

$t = 2711.2$ sec, a GPS outage occurs and the heading angle is held fixed. Then the Kalman filter switches to using the heading estimate $\hat{\psi}_k$ instead of the false heading $\psi_{\mathrm{GPS_k}}$ to calculate the rate of change of the heading angle as follows,

$$\dot{\psi}_k^* = (\hat{\psi}_k - \psi_{\mathrm{GPS}_{k-1}}). \tag{146}$$

The use of estimated heading continues until the GPS sensor yields correct heading information at $t = 2714.2$ [sec].

B.1.5 Attitude filter validation

The previous algorithm was written in C codes and implemented as an S-function in the Matlab/Simulink environment. This enables the actual C codes to be validated for any errors and tuned before used for the UAV. A complete non-linear 6-DOF Simulink model[58] is used to simulate the full dynamics of the UAV. The inertial sensor measurements are emulated to have close correlation to the real sensors used in the UAV in terms of signal specifications and noise characteristics. The gain parameters for the pitch and heading complementary filters were chosen as

$$\tau_\theta = 2, \quad \tau_\psi = 0.5, \quad \alpha_\psi = 0.4.$$

The process noise covariance matrix and the noise covariance matrix for the Kalman filter were carefully chosen in consideration of the noise characteristics of the sensors as follows

$$\begin{aligned} \mathbf{Q}_k &= \mathrm{diag}\Big(\left[\begin{array}{ccc} 0.02^2 & 0.001^2 & 0.01^2 \end{array}\right]\Big), \\ \mathbf{R}_k^f &= 0.02^2, \qquad \mathbf{R}_k^s = 0.05^2. \end{aligned} \tag{147}$$

Two internal PID controller loops were designed for roll angle control and pitch angle control with the associated stability augmentation dampers. The filter outputs are fed back to the corresponding PID controllers and validated in the closed loop. Figure 69 shows the comparison between true values and estimated values.

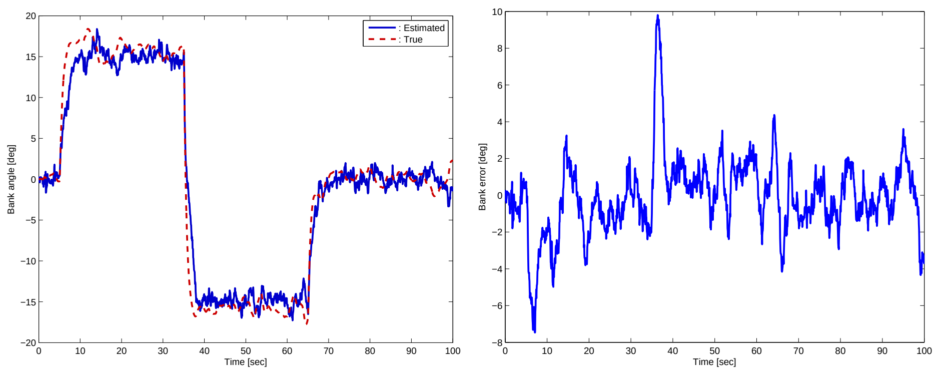
In Figs. 69(a) and 69(c) a doublet roll angle reference command was used to excite the lateral motion of the airplane, while the pitch attitude is held constant at zero. Both the complementary filter and the Kalman filter operate properly. However, a transient time lag in the estimation process is observed when the UAV changes its orientation quickly, which leads to the increased estimation error shown in the right side of Fig. 69 during the transients. Nonetheless, the filter converges to the correct angle after the UAV comes into steady state. Figure 69(b) shows the pitch angle estimation when a doublet pitch angle reference command was used to excite the longitudinal motion of the UAV, while the roll angle is controlled to zero.

B.2 Position Estimation

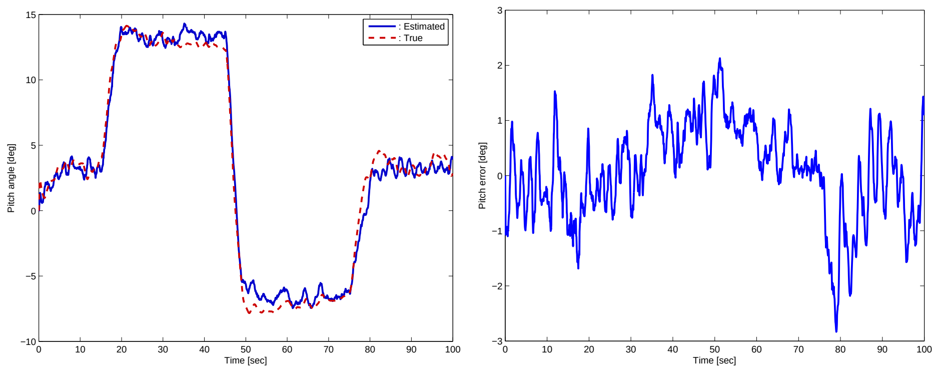
In this section a filter for estimating the absolute position of the UAV in the north-east-down (NED) inertial reference frame is developed. A typical attitude heading reference system/inertial navigation system (AHRS/INS) makes use of the accelerometer output in conjunction with the full equations of motion to propagate the inertial position and velocity from acceleration measurements using a Kalman filter. However, the dimension of this complete Kalman filter is too large to be implemented on a micro-controller and run in real-time. Instead of using the full equations of motion, the navigation equations are used to propagate the position from the flight speed measurement. The position filter is cascaded with the attitude filters so as to allow separate filter tuning and at the same time to reduce the computational cost with minimal loss of performance.

B.2.1 Filter formulation

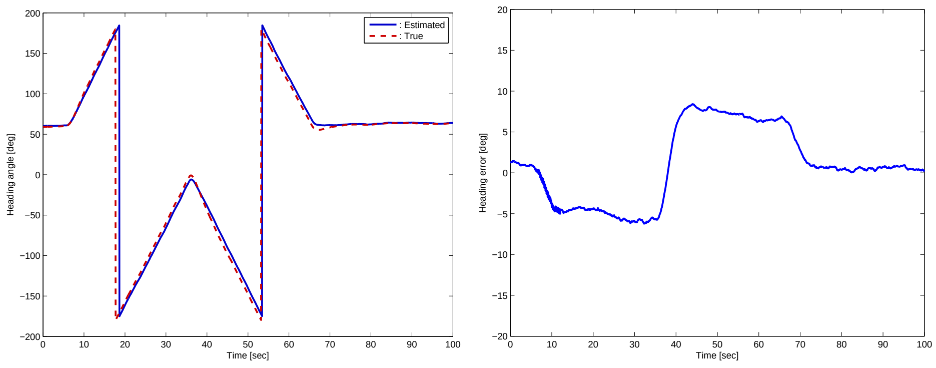
The navigation equations of a 6-dof airplane[137] are used to obtain the position filter equation. Using the rotational transform matrix from the body-axes ($\mathcal{B}$) to the



(a) Roll angle filter performance



(b) Pitch angle filter performance



(c) Yaw angle filter performance

Figure 69: Attitude estimation filters validation.

inertial frame ($\mathcal{N}$), the navigation equations are given by

$$\begin{bmatrix} \dot{p}^N \\ \dot{p}^E \\ \dot{p}^D \end{bmatrix} = \begin{bmatrix} v^N \\ v^E \\ v^D \end{bmatrix} = \begin{bmatrix} {}^{\mathcal{B}}\mathbf{C}^{\mathcal{N}} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} U \\ V \\ W \end{bmatrix}. \tag{148}$$

Assume that the angle of attack and side slip angle are small, the velocity components expressed in terms of the body-axes are approximated by

$$U \approx V_T, \qquad V, W \approx 0. \tag{149}$$

The flight speed V_T, if it is measured from a pitot tube, includes the inertial speed (relative ground speed) and the wind speed. The relative ground speed is only to be integrated to propagate the inertial position. The wind speed is dependent on the flight condition, so is added to the position filter as an extra state to account for the pitot speed measurement. A random walk model for the wind speed variation is assumed, driven by a Gaussian white noise process as follows,

$$\dot{V}_w = \epsilon_w. \tag{150}$$

The speed measurement V_{T_m} from the pitot tube contains the inertial speed, the wind speed, and the measurement noise η_w,

$$V_{T_m} = V_T + V_{\mathrm{w}} + \eta_w. \tag{151}$$

The attitude angle information used in Eq. (148) is provided separately by the attitude filters. This cascaded configuration of the attitude and position filters reduces the complexity arising from the coupling of the variables. It therefore yields a simple position estimation filter with minimal order. It follows from Eq. (148), (149), and (151) , that the equations of filter dynamics are given by

$$\begin{bmatrix} \dot{p}^N \\ \dot{p}^E \\ \dot{p}^D \\ \dot{V}_w \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi \\ \cos\theta\sin\psi \\ -\sin\theta \\ 0 \end{bmatrix} (V_{T_m} - V_w) + \begin{bmatrix} \nu_N \\ \nu_E \\ \nu_D \\ \epsilon_w \end{bmatrix}, \tag{152}$$

where, $[\nu_N \ \nu_E \ \nu_D]^\mathsf{T}$ is the process noise vector of η_w being projected onto the NED frame.

The measurement model for the position estimation filter is described as follows. The North position p_m^N and the East position measurements p_m^E are provided by the GPS sensor at an update rate of 1 Hz,

$$\begin{aligned} p_m^N &= p^N + \eta_N, \\ p_m^E &= p^E + \eta_E, \end{aligned} \tag{153}$$

where η_N and η_E are the measurement noise for the North and East directions, which are assumed to be the Gaussian.

Altitude information with good accuracy is attainable using a barometric altimeter. The one used in our UAV platform has minimum three-meter resolution and a higher update rate. The down position measurement p_m^D with a corresponding Gaussian noise is obtained by,

$$p_m^D = -h + \eta_h. \tag{154}$$

Having multiple measurements at different update rates, the discrete position Kalman filter implementation at a specified sampling period Δt on a micro-controller is given as follows.

- **Time update**
  - Project ahead

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \Phi_k \hat{\mathbf{x}}_{k-1} + V_w [\ \Delta t \sin\hat{\theta}_k \cos\hat{\psi}_k \quad \Delta t \cos\hat{\theta}_k \sin\hat{\psi}_k \quad -\Delta t \sin\hat{\theta}_k \quad 0\ ]^\mathsf{T}, \\ \mathbf{P}_k^- &= \Phi_k \mathbf{P}_{k-1} \Phi_k^\mathsf{T} + \mathbf{Q}_k, \end{aligned} \tag{155}$$

where,

$$\Phi_k = \begin{bmatrix} 1 & 0 & 0 & -\Delta t \cos\hat{\theta}_k \cos\hat{\psi}_k \\ 0 & 1 & 0 & -\Delta t \cos\hat{\theta}_k \sin\hat{\psi}_k \\ 0 & 0 & 1 & \Delta t \sin\hat{\theta}_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \hat{\mathbf{x}}_k = \begin{bmatrix} \hat{p}_k^N \\ \hat{p}_k^E \\ \hat{p}_k^D \\ \hat{V}_{w_k} \end{bmatrix}.$$

- **Measurement update**
  - Compute Kalman gain

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} \left(\mathbf{H}_k^\star \mathbf{P}_k^- \mathbf{H}_k^{\star\mathsf{T}} + \mathbf{R}_k^\star \right)^{-1}, \tag{156}$$

  - Update estimate with measurements

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k^\star - \mathbf{H}_k^\star \hat{\mathbf{x}}_k^-), \tag{157}$$

  - Compute error covariance for updated estimate

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k^\star) \mathbf{P}_k^-, \tag{158}$$

  where, $\star = s,\ f$.

Each update is performed whenever the corresponding measurement becomes available: a fast update and a slow update.

- Fast update

$$\mathbf{z}_k^f = -h_k, \qquad \mathbf{H}_k^f = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, \tag{159}$$

- Slow update

$$\mathbf{z}_k^s = \begin{bmatrix} p_{m_k}^N \\ p_{m_k}^E \end{bmatrix}, \qquad \mathbf{H}_k^s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{160}$$

The process noise covariance matrix $\mathbf{Q_N}$ and measurement noise covariance matrix $\mathbf{R_N}$ are determined from the noise characteristics of each signal,

$$\begin{aligned} \mathbf{Q_N} &= \operatorname{diag}\Big(E\big[\ \bar{\nu}_N^2 \quad \bar{\nu}_E^2 \quad \bar{\nu}_h^2 \quad \bar{\epsilon}_w^2\ \big]\Big), \\ \mathbf{R}_\mathbf{N}^f &= E\big[\bar{\eta}_h^2\big], \qquad \mathbf{R}_\mathbf{N}^s = \operatorname{diag}\Big(E\big[\ \bar{\eta}_N^2 \quad \bar{\eta}_E^2\ \big]\Big), \end{aligned} \tag{161}$$

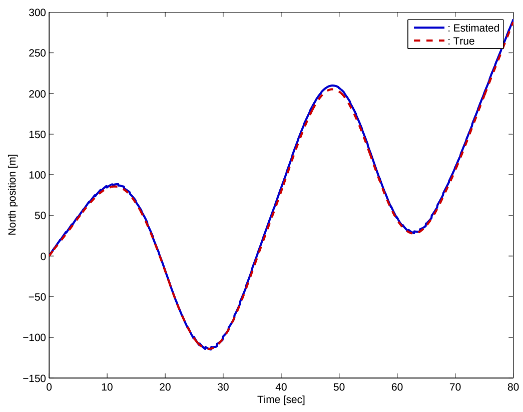
where the overbar variables represent the discrete noise sequences at a sampling period of Δt having equal noise strength as the continuous noise process, and $E[\cdot]$ calculates the mean-square noise strength.

B.2.2 Navigation filter validation

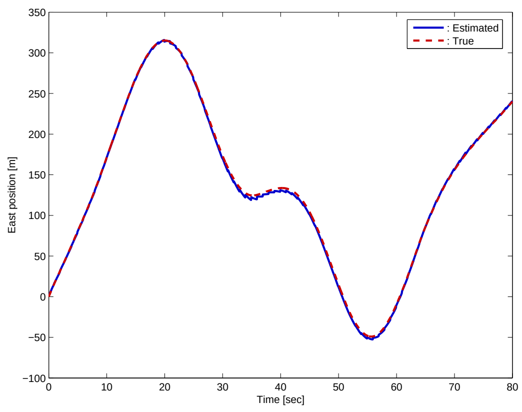
The navigation filter was written in C code as an S-function of the Matlab/Simulink® environment, as described in Section B.1.5. The process covariance matrix and the noise covariance matrix for the navigation filter were carefully chosen based on the noise characteristics of the sensors as follows

$$\begin{aligned} \mathbf{Q_N} &= \operatorname{diag}\Big(\big[\ 2^2 \quad 2^2 \quad 2^2 \quad 0.01^2\ \big]\Big), \\ \mathbf{R}_\mathbf{N}^f &= 2^2, \qquad \mathbf{R}_\mathbf{N}^s = \operatorname{diag}\Big(\big[\ 3^2 \quad 3^2\ \big]\Big). \end{aligned} \tag{162}$$

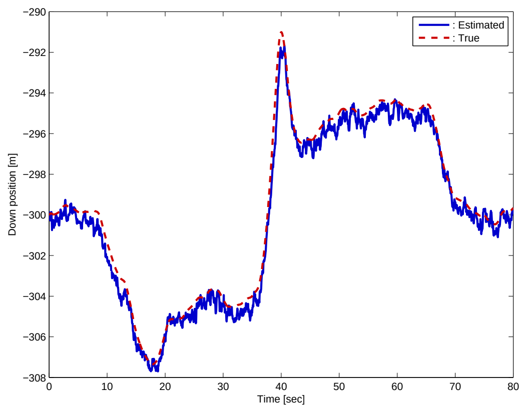
An open loop steering command for the UAV to perform an eight-shape maneuver was used, which in turn results in a doublet bank angle command as a reference to the roll PID controller. Figure 70 shows the performance of the navigation filter. Overall, the navigation filter works very well by providing a series of estimation values for the main control loop at a high rate (20 Hz) despite the low update rate of the GPS sensor (1 Hz). The North and East position estimates appear to have certain corrections periodically, which can be explained by the fact that during no GPS output update, the heading estimation from the attitude filters is used to propagate the position estimation. However, the time lag of the heading output from the attitude filters causes the navigation filter to use non-ideal heading information during propagation of the states. This results in a drifted position estimate at one second after when



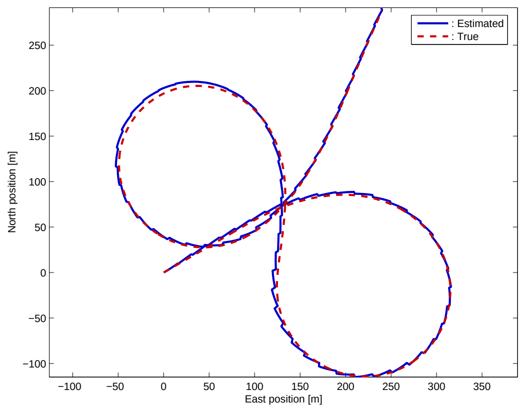
(a) North position



(b) East position



(c) Down position



(d) North-East position

Figure 70: Navigation filter validation.

a new GPS measurement becomes available. The navigation Kalman filter works to update the filter states according to the new GPS measurement in order to dump out the drift and to provide the best estimate of position at all times. In addition, the altitude of the UAV is controlled by the altitude PID controller loop along with the doublet reference command, which enables the position filter to be validated with respect to the down position. The Kalman filter works properly while estimating the vertical position despite the noisy barometric altitude measurement.

B.3 Summary

A simple, yet effective attitude and position estimation algorithm has been developed for use with a low-cost UAV autopilot. Utilizing a complementary filter for estimating the pitch and heading angles, dramatically reduces the computational burden. A minimal dimension Kalman filter estimates the roll angle. An algorithm for handling GPS lock is given and is tested to show the feasibility of real-time implementation with delayed GPS measurements. A cascaded position filter is also derived, and it is shown to be effective at incorporating the slow GPS output in order to provide a high update rate position solution. Results from both simulation and hardware validation show that the execution times of the estimation algorithms are well within the capability of the micro-controller (about 10 [msec] for the attitude filters and 3 [msec] for the navigation filter).

APPENDIX C

MODELING AND HARDWARE-IN-THE-LOOP SIMULATION

C.1 Equations of Motion of a Fixed-Wing Aircraft

The standard 6-dof equations of motion for a conventional aircraft are used for modeling and simulation of a small UAV. Flat Earth approximation[137] provides a reasonable modeling assumption when the vehicle operates over a small area. The body-axes equations are as follows:

Force equations:

$$\dot{U} = rV - qW - g\sin\theta + (X_A + X_T)/m, \tag{163a}$$

$$\dot{V} = -rU + pW + g\sin\phi\cos\theta + (Y_A + Y_T)/m, \tag{163b}$$

$$\dot{W} = qU - pV + g\cos\phi\cos\theta + (Z_A + Z_T)/m, \tag{163c}$$

Moment equations:

$$J_x\dot{p} - J_{xz}(\dot{r} + pq) + (J_z - J_y)qr = \bar{L}, \tag{164a}$$

$$J_y\dot{q} + (J_x - J_z)pr + J_{xz}(p^2 - r^2) = M, \tag{164b}$$

$$J_z\dot{r} - J_{xz}(\dot{p} - qr) + (J_y - J_x)pq = N, \tag{164c}$$

Kinematic equations:

$$\dot{\phi} = p + \tan\theta(q\sin\phi + r\cos\phi), \tag{165a}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi, \tag{165b}$$

$$\dot{\psi} = (q\sin\phi + r\cos\phi)/\cos\theta, \tag{165c}$$

Navigation equations:

$$\begin{aligned}
\dot{p}^N &= U\mathrm{c}\theta\mathrm{c}\psi + V(-\mathrm{c}\phi\mathrm{s}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi) + W(\mathrm{s}\phi\mathrm{s}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{c}\psi), && \text{(166a)}\\
\dot{p}^E &= U\mathrm{c}\theta\mathrm{s}\psi + V(\mathrm{c}\phi\mathrm{c}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi) + W(-\mathrm{s}\phi\mathrm{c}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi), && \text{(166b)}\\
\dot{p}^D &= -U\mathrm{s}\theta + V\mathrm{s}\phi\mathrm{c}\theta + W\mathrm{c}\phi\mathrm{c}\theta, && \text{(166c)}
\end{aligned}$$

where the definition of each variable is found in Ref. [137].

The aerodynamic forces and moments are obtained from the dimensionless aerodynamic coefficients at a given flight condition as follows,

$$X_A = \bar{q}SC_X, \quad Y_A = \bar{q}SC_Y, \quad Z_A = \bar{q}SC_Z, \tag{167a}$$

$$\bar{L} = \bar{q}SbC_\ell, \quad M = \bar{q}S\bar{c}C_m, \quad N = \bar{q}SbC_n. \tag{167b}$$

The analysis of the aerodynamic behavior of the aircraft, however, is better understood in the stability axes, or wind axes, system. In particular, the aerodynamic forces are easily handled in the wind axes, which is given in terms of the angle of attack α and the sideslip angle β. Hence, the force equations in wind-axes are introduced as follows[137],

$$\begin{aligned}
m\dot{V}_T &= T\cos(\alpha+\alpha_T)\cos\beta - D + mg_1, && \text{(168a)}\\
m\dot{\beta}V_T &= -T\cos(\alpha+\alpha_T)\sin\beta - C_w + mg_2 - mV_Tr_s, && \text{(168b)}\\
m\dot{\alpha}V_T\cos\beta &= -T\sin(\alpha+\alpha_T) - L + mg_3 + mV_T(q\cos\beta - p_s\sin\beta), && \text{(168c)}
\end{aligned}$$

where, V_T is total air speed, p_s and r_s are the pitch and yaw rates projected on the stability axes, m is the mass of the vehicle, and g_1, g_2, and g_3 are the projections of the gravitational acceleration on the wind axes system. The aerodynamic forces are lift (L), drag (D), and cross wind force (C_w) while T is the thrust force exerted on the aircraft.

C.1.1 Aerodynamic coefficients for forces and moments

The aerodynamic forces and moments have a complex dependence on a number of variables resulting in complicated nonlinear correlation between each variable. Building up the aerodynamic forces and moments as a linear sum of contributing components provides a mathematically convenient way of representing the aerodynamic forces and moments for a specified flight condition. By the same token, dimensionless aerodynamic coefficients, which are associated with the stability and control derivatives in terms of independent parameters have been widely used to represent the aerodynamic characteristic of an aircraft:

$$\begin{aligned}
C_D &= C_{D0} + \frac{(C_L - C_{L0})^2}{\pi e AR} + |C_{D_{\delta e}}\delta_e| + |C_{D_{\delta_a}}\delta_a| + |C_{D_{\delta r}}\delta_r|, & \text{(169a)}\\
C_Y &= C_{y_\beta}\beta + C_{y_{\delta_a}}\delta_a + C_{y_{\delta_r}}\delta_r + \frac{b}{2V_T}(C_{y_p}p_s + C_{y_r}r_s), & \text{(169b)}\\
C_L &= C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e + \frac{\bar{c}}{2V_T}(C_{L_{\dot{\alpha}}}\dot{\alpha} + C_{L_q}q), & \text{(169c)}\\
C_m &= C_{m0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e + \frac{\bar{c}}{2V_T}(C_{m_{\dot{\alpha}}}\dot{\alpha} + C_{m_q}q), & \text{(169d)}\\
C_\ell &= C_{\ell_\beta}\beta + C_{\ell_{\delta_a}}\delta_a + C_{\ell_{\delta_r}}\delta_r + \frac{b}{2V_T}(C_{\ell_p}p_s + C_{\ell_r}r_s), & \text{(169e)}\\
C_n &= C_{n_\beta}\beta + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + \frac{b}{2V_T}(C_{n_p}p_s + C_{n_r}r_s), & \text{(169f)}
\end{aligned}$$

where, e is the Oswald coefficient, AR is the main wing aspect ratio.

C.1.2 Numerical modeling of aerodynamic coefficients

The stability and control derivatives shown in Eqs. (169) determine the aerodynamic characteristic of the aircraft, and can be estimated or identified through wind tunnel tests or via flight tests. Despite the fact that the wind tunnel test gives accurate results, a complete solution for all derivatives is not always feasible from the wind tunnel test alone. Hence, it is beneficial to obtain these derivatives from empirical data, if possible. The United States Air Force (USAF) has developed the stability and control compendium (DATCOM)[150], a large collection of information combining both classical aerodynamic analysis and experimental data. The digital DATCOM[16] is the digital version of DATCOM, a computer program which was originally written in Fortran and ported afterwards to ANSI C. It incorporates the classical aerodynamic equations with empirical corrections from experimental data for various aircraft to compute the aerodynamic stability and control derivatives of the aircraft. The DDATCOM is useful in predicting the stability and control derivatives for preliminary aircraft designs or developing a high fidelity flight simulation[44]. The DDATCOM takes an input file containing the aircraft configuration and geometric parameters, the flight condition, the mass properties, and so on. The input configuration file of the UAV airframe used in this research is shown in Table 10.

Table 10: Digital DATCOM input configuration file of the 1/5 scale Decathlon.

Properties	value	units
Mass (m)	5.6132	[kg]
Aerodynamic reference area (S)	0.6558	[m^2]
Longitudinal reference length ($\bar{c}$)	0.3215	[m]
Lateral reference length (b)	2.04	[m]
Free stream airspeed (V_T)	20	[m/sec]
Flight path angle (γ)	0	[deg]
Altitude (h)	300	[m]

The geometric configuration of the UAV airframe was accurately measured and entered in the DDATCOM input file. This included the total weight, the shape of the wing/tail airfoil section, a cylindrical modeling of the fuselage section, the center of gravity location, the placement of wing/tails, and etc. Figure 71(a) shows the UAV airframe, a commercial Decathlon R/C model, and Fig. 71(b) shows the detailed CAD model of the airframe. With the geometric data supplied, the DDATCOM computes the static stability derivatives, the dynamic stability derivatives, and control derivatives for both the aileron and the elevator. The control derivatives associated with the rudder and the stability derivatives associated with the yawing moment are not computed from the DDATCOM. These derivatives can be estimated using the method proposed in Ref. [132]. The stability and control derivatives calculated from

(a)

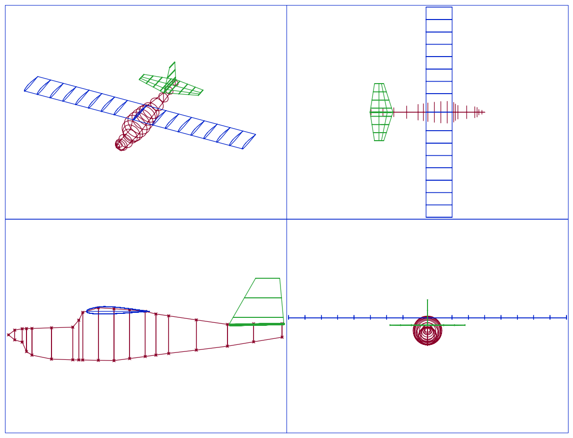

(b)

Figure 71: Geometric modeling of the 1/5 scale Decathlon used as an input to the DDATCOM program.

the DDATCOM are summarized in Table 11.

Table 11: Static and dynamic stability derivatives and control derivatives estimated from the geometry of the 1/5 scale Decathlon.

C_{D_0}	C_{L_0}	C_{L_α}	$C_{L_{\dot\alpha}}$	C_{L_q}	C_{y_β}	C_{y_p}	C_{y_r}
0.028	0.062	5.195	1.22	4.589	-0.2083	0.0057	0.0645*
C_{m_0}	C_{m_α}	$C_{m_{\dot\alpha}}$	C_{m_q}	C_{ℓ_β}	C_{ℓ_p}	C_{ℓ_r}	C_{n_β}
0.0598	-0.9317	-2.897	-5.263	-0.0377	-0.4625	0.0288	0.0116
C_{n_p}	C_{n_r}	$C_{D_{\delta_e}}$	$C_{D_{\delta_a}}$	$C_{D_{\delta_r}}$	$C_{L_{\delta_e}}$	$C_{y_{\delta_a}}$	$C_{y_{\delta_r}}$
-0.0076	-0.0276	0.0418	0.0†	0.0†	0.2167	0.0†	0.1096*
$C_{m_{\delta_e}}$	$C_{\ell_{\delta_a}}$	$C_{\ell_{\delta_r}}$	$C_{n_{\delta_r}}$	$C_{n_{\delta_a}}$			
-0.8551	-0.2559	0.0085†	0.0035	-0.0216†			

† : approximately zero assumed, * : estimation from the Reference [132].

C.1.3 Mass properties modeling

The mass properties can be properly estimated from a computational analysis with a help of CAD program. In contrast to the a CAD modeling, the mass properties have been identified experimentally based on the actual platform. The total weight of the UAV was measured using a precise scale, and the center of gravity location was carefully determined in the body axes with respect to the base point at the center of propeller. The mass moment of inertia about each principal axis were identified

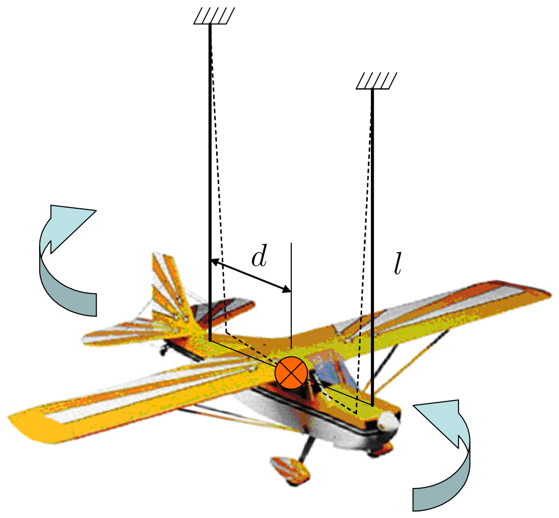


Figure 72: Torsional pendulum experimental setup for identifying the moment of inertia.

experimentally using the formula[53]

$$J_\star = \frac{gT_m^2 d^2 m}{4\pi^2 l}, \quad \star = x, y, z \tag{170}$$

where, d is the distance from the center of gravity to the point at which the suspension wire is vertically connected, T_m is the period of a small perturbed rotational motion captured by the rate gyros, and l is the length of the wire from the ceiling. Figure 72 illustrates the detail torsional pendulum experimental setup. After careful alignment of the aircraft to the horizontal plane and equal distance for each anchor point to the center of gravity, the moments of inertia were identified in the direction of x-, y-, z- of the body axes, as J_x, J_y, and J_z, respectively. Table 12 summarizes the final mass properties from experiments.

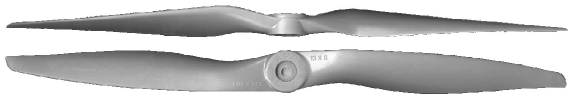
Table 12: Mass properties identified from experiments

Properties	symbols	values
Mass	m	5.6132 [kg]
Center of mass location in body axes	$[r_x \;\; r_y \;\; r_z]$	$[-0.47 \;\; 0 \;\; -0.012]$ [m]
Moment of inertia	$[J_x \;\; J_y \;\; J_z]$	$[0.4497 \;\; 0.5111 \;\; 0.8470]$ $[\mathrm{kg} \cdot \mathrm{m}^2]$

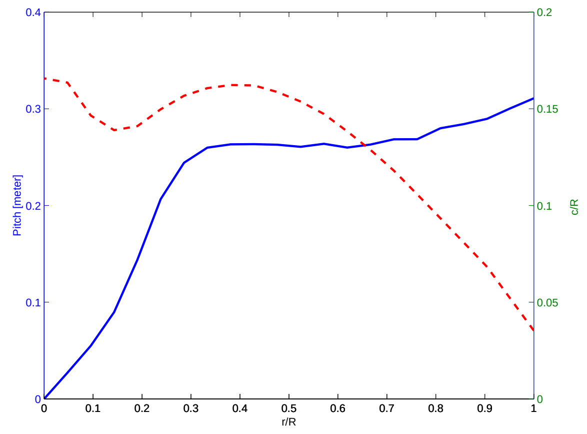
C.1.4 Engine and propeller modeling

The UAV is powered by a two-stroke internal combustion engine with a fixed pitch propeller. Since the thrust force is always acting on the airplane to overcome the drag even during steady level flight, comprehensive understanding on the thrust producing mechanism for the propeller-engine enables a faithful mathematical model for the simulation. The model engine is an O.S. FX 91 that delivers the maximum 2.8 horse power (HP) at a rotational speed of 16000 [rpm] with a propeller of 13 inches in diameter and 8 inches in propeller pitch distance. As a complete modeling of the engine with the propeller is beyond our scope, a simplified modeling of the thrust force exerted on the aircraft is given as follows. Assume that the engine is overpowered providing any required power to the propeller. Hence, the thrust is solely dependent on the propeller aerodynamics.

The aerodynamics of the propeller can be understood by a combination of the lift and drag force along its cross-sectional plane. The analysis of the aerodynamic forces along the propeller is based on the geometry of the propeller. The geometry for an APC model propeller of 13 inches in diameter by 8 inches in pitch distance is shown in Fig. 73. The static thrust can be computed from a program, JavaProp[92], which is written based on the blade element theory[78, 2]. With the geometry of the propeller is specified a priori, it calculates the produced thrust in terms of the rotating speed of the propeller. On the other hand, an experiment was conducted to identify the actual static thrust at different throttle command $\delta_t \in [0,\ 1]$. Figure 74 illustrates a simple experimental setup using a spring scale to directly measure the static thrust. The static thrust was approximately gauged at the steady state with a RPM reading at the given throttle command. Figure 75 shows the measured RPM v/s input throttle command. While the solid line represents actual measurements, the dashed line represents the best fitting curve that is obtained from a polynomial



(a) Horizonal and vertical view.



(b) Geometry of an APC propeller (chord length and pitch distance).

Figure 73: APC model airplane propeller 13″ by 8″.

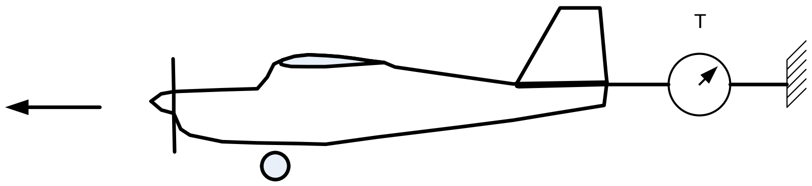


Figure 74: Experimental setup for identifying static thrust force.

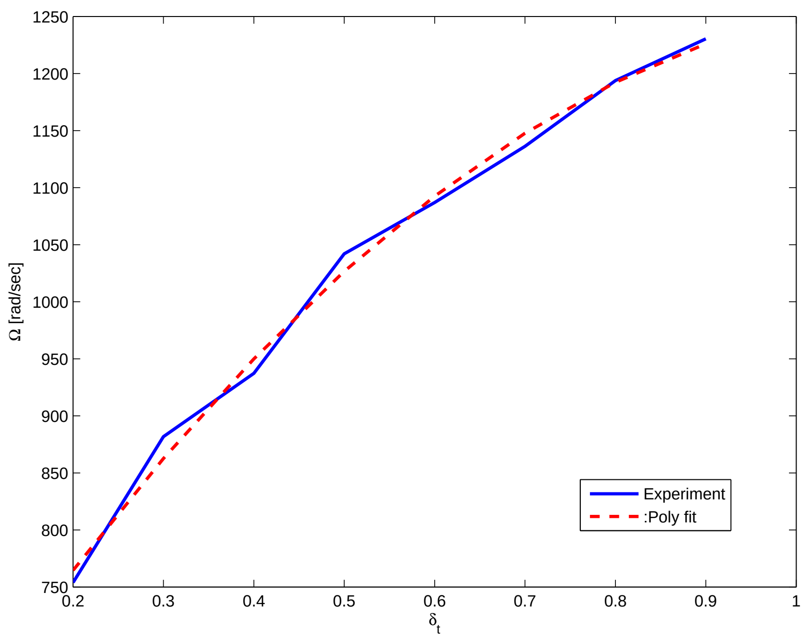


Figure 75: Rotating speed of propeller v/s the throttle command.

fit in least-square sense as follows,

$$\Omega = -534.8\delta_t^2 \ + \ 1247.5\delta_t \ + \ 536.5 \quad \text{[rad/sec]}. \tag{171}$$

Figure 76 compares the actual static thrust measured with the calculated thrust from the JavaProp. Figure 76 shows that the estimated thrust has similar trend as such that the thrust increases quadratically as the rotational speed increases. The shortage of thrust by the experiment can be attributed to the fact that the downstream after the propeller is hindered by the fuselage cross section, which affects to reduce the mass flow rate thus less thrust generated. By a simple correction, the numerical thrust can be scaled to the actual thrust by the following relation, which was also plotted by a dashed line in Fig. 76.

$$T_{\text{ex}} = 1.2407 \cdot T_{\text{th}} - 8.4742 \quad \text{[N]}. \tag{172}$$

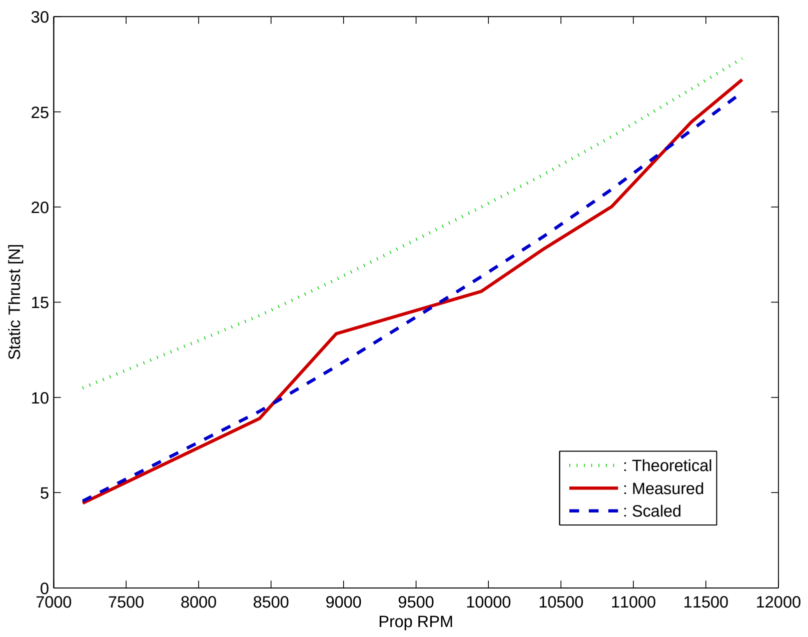


Figure 76: Static thrust comparison for a model propeller 13″ by 8″

The dynamic thrust, on the other hand, is quite complicated to be estimated directly from the propeller geometry. In general, the thrust force during flights depends not only on the rotational speed of the propeller but also on the forward flight speed (incoming wind speed). The forward flight speed reduces an effective angle of attack to the airfoil cross section, thus eventually reducing generated thrust force. This aspect will continue until a certain combination of RPM and forward flight speed at which no thrust is generated and changes its sign for an opposite direction (wind mill effect). In order to take into account this aspect, the advance ratio (J) should be introduced to incorporate the effect of forward flight speed as follows,

$$J = \frac{\pi V_T}{\Omega R_p}, \tag{173}$$

where, R_p is the propeller radius in meters and Ω the rotational speed in [rad/sec].

From the JavaProp, the thrust coefficient (c_T) was obtained in terms of the advance ratio (J) as shown in Fig. 77. This dimensionless coefficient is related to the

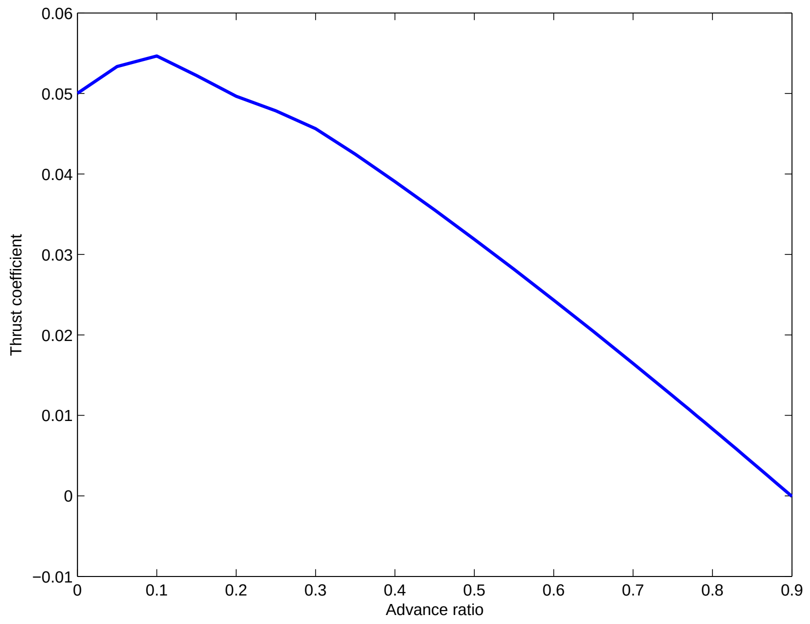


Figure 77: Thrust coefficient c_T v/s advance ratio for the selected propeller.

actual thrust by the following relation,

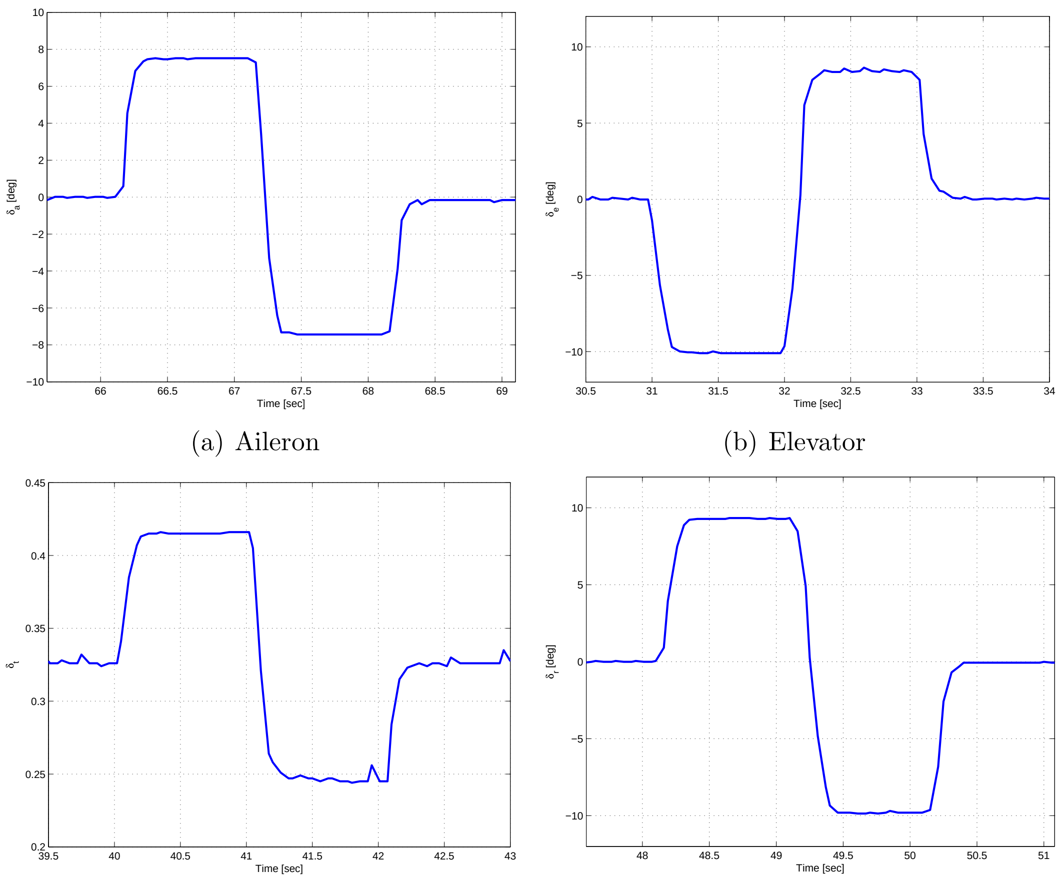
$$T_{\text{th}} = \frac{4}{\pi^2}\rho\Omega^2 R_p^4 c_T \tag{174}$$

where, ρ is the air density. Hence, the actual thrust can be estimated from Eqs. (172) and (174) using an approximation of c_T at a given advance ratio value that is computed from the flight speed and the rotational speed of propeller at each instant.

In addition, the dynamic characteristic of thrust generation is modeled by a transfer function from throttle command (δ_t) to the thrust output. The transfer function is experimentally identified by a first order low pass filter with the time constant 0.3333 [sec], which lumps together various effects such as servo motor dynamics, engine dynamics, and propeller aerodynamics.

C.1.5 Actuator modeling

The control surfaces of the UAV are actuated by four identical R/C servo motors. Each servo arm is coupled to the corresponding control surface via mechanical links.



(a) Aileron (b) Elevator

(c) Throttle (d) Rudder

Figure 78: Various doublet responses for dynamic modeling of the control surfaces.

Hence, in order to obtain the dynamic characteristics of the control surface deflection, several doublet responses were recorded for each control surface. Figure 78 shows these responses. Deflection angles of the control surfaces are measured by the potentiometers linked to the servo motor. It appears that an R/C servo motor nearly performs a dead beat control action. Thus, a first-order low pass filter combined with both slew rate limit and control saturation results in a good actuator model. Table 13 summarizes the results of dynamic modeling of the control surfaces.

C.2 Estimation of Aerodynamic Angles

The angle of attack and the sideslip angle are significant states describing the aerodynamics of the airplane. Specifically, these angles are required for identification of

Table 13: Results of dynamic modeling of each control surface identified by experiments.

Control surface	+Range	-Range	Slew rate
Aileron δ_a	21.8 deg	-21 deg	106.5 deg/sec
Elevator δ_e	28 deg	-29.6 deg	129.4 deg/sec
Rudder δ_r	16 deg	-20.7 deg	141.2 deg/sec
Throttle δ_t	Full open (1)	Full closed (0)	2.5 /sec

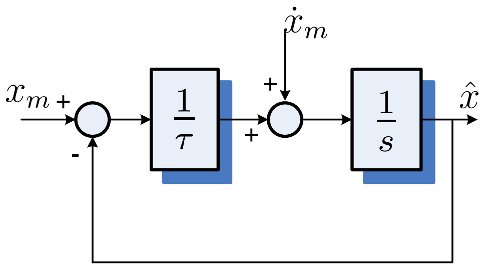


Figure 79: A complementary filter in the feedback form.

aerodynamic force and moment coefficients. For exact measurements, it is necessary to install appropriate air data sensors on-board the aircraft. However, it is sometimes difficult to install such apparatuses on a small UAV. Another approach, used here, is to determine the aerodynamic angles using inertial sensor measurements. In this section, the aerodynamic angles are estimated from raw inertial acceleration measurements using an approach similar to the one presented in Refs. [31, 51].

C.2.1 Filter formulation

Complementary filters have been widely used for combining two independent noisy measurements of a same signal, where each measurement is corrupted by different types of spectral noise[20]. Figure 79 shows a complementary filter that uses two measurements $x_m(t)$ and $\dot{x}_m(t)$ to obtain an estimate $\hat{x}(t)$ of $x(t)$. The time constant τ is selected according to the noise characteristics of each channel such that the estimate $\hat{x}$ is contributed by integration of $\dot{x}_m$ over frequencies $\omega \gg 1/\tau$, whereas for frequencies $\omega \ll 1/\tau$, $\hat{x}$ tracks x_m.

In order to employ a complementary filter in the estimation of the aerodynamic angles, one needs to find the aerodynamic angles and their derivatives computed from the inertial measurements. Using a small angle approximation for α and β during a steady state flight condition, the body z-axis accelerometer output a_m^z measures approximately the thrust force and the aerodynamic force in the wind-axes as follows

$$ma_m^z \cong F_T \sin(\alpha + \alpha_T) + L. \tag{175}$$

It follows from Eq. (168c) that the time derivative of the angle of attack is obtained by

$$\dot{\alpha}_m = \frac{g - a_m^z}{V_T} + q. \tag{176}$$

The thrust force components in Eq. (175) can be further neglected from the assumption of small angle of attack and the thrust vector aligned to the body x-axis. Knowing that the lift force L is represented by the dimensionless coefficient of Eq (169c), one can obtain the angle of attack from the accelerometer output and the aircraft states using the following relation

$$\alpha_m = \frac{1}{C_{L_\alpha}}\Big(\frac{m}{\bar{q}S}a_m^z - \frac{\bar{c}C_{L_{\dot{\alpha}}}}{2V_T^2}(g - a_m^z) - C_{L_0} - C_{L_{\delta_e}}\delta_e - \frac{\bar{c}q}{2V_T}(C_{L_{\dot{\alpha}}} + C_{L_q})\Big). \tag{177}$$

It should be noted that a priori knowledge of the stability and control derivatives used in Eq. (177) is important to get a correct estimate of the angle of attack. These parameters can be supplied by the method discussed in Section C.1.1.

For the sideslip angle estimation, the body y-axis accelerometer output a_m^y measures approximately the thrust force and the aerodynamic force as follows

$$ma_m^y \cong F_T \cos(\alpha + \alpha_T)\sin\beta + C_w, \tag{178}$$

where C_w is the cross-wind force component. With a small angle approximation of α and β during a steady-state flight condition at small θ, it follows that $g_2 \approx g\sin\phi$ and $r_s \approx r$ in Eq. (168b). Hence the time derivative of the sideslip angle yields

$$\dot{\beta}_m = \frac{1}{V_T}\big(g\sin\phi - a_m^y\big) - r. \tag{179}$$

The thrust force components in Eq. (178) can also be neglected along the same reason discussed above. Assuming that the cross-wind force component C_w in Eq. (168b) is related to the body force component Y_A by $C_w \cong -Y_A$, it follows from Eq. (169b) that the sideslip angle is obtained from the accelerometer output and the aircraft states using the following relation

$$\beta_m = -\frac{1}{C_{y_\beta}}\Big(\frac{m}{\bar{q}S}a_m^y + C_{y_{\delta_r}}\delta_r + \frac{b}{2V_T}\big(C_{y_p}p + C_{y_r}r\big)\Big). \tag{180}$$

The estimates of the angle of attack and the sideslip angle are then computed from Eqs. (176),(177),(179) and (180) using the complementary filters, as illustrated in Fig. 79.

C.3 Identification for Aerodynamic Force/Moment Coefficients

In this section, the stability and control derivatives of the UAV are identified from the actual flight test data. In contrast to the discussion in Section C.1.1, the aerodynamic coefficients with respect to the body-axes system will be dealt with in the following formulation due to the fact that measurements are obtained via body-fixed inertial sensors. As discussed already, these coefficients are assumed to be linear in terms of each contributing component, so a linear parameterization is adopted as the identification model structure.

C.3.1 A linear parameterization

The measurements for model identification are provided by the on-board sensor suite. Among these sensors, accelerometers and rate gyros capture the dynamic behavior of the UAV, which yields the aerodynamic force coefficients expressed in the body axes as follows,

$$C_X = \frac{m}{\bar{q}S}a_m^x - \frac{T}{\bar{q}S}, \quad C_Y = \frac{m}{\bar{q}S}a_m^y, \quad C_Z = \frac{m}{\bar{q}S}a_m^z. \tag{181}$$

where the thrust force was assumed to be only exerted along the body x-axis and is modeled as a function of the rotating speed of the propeller and the forward flight speed as discussed in Section C.1.4, as follws

$$T = f(\rho,\ \Omega,\ V_T). \tag{182}$$

The aerodynamic moment coefficients are obtained as follows

$$C_\ell = \frac{\bar{L}}{\bar{q}Sb}, \quad C_m = \frac{M}{\bar{q}S\bar{c}}, \quad C_n = \frac{N}{\bar{q}Sb}, \tag{183}$$

where $\bar{L}$, M and N are calculated from Eqs. (164a), (164b) and (164c) using the body rate measurements $\boldsymbol{\omega} = [p\ q\ r]^\mathsf{T}$ and the corresponding numerical differentiation $\dot{\boldsymbol{\omega}} = [\dot{p}\ \dot{q}\ \dot{r}]^\mathsf{T}$.

A linear regression model structure was opted for by taking into account the decoupled longitudinal and lateral dynamics of the aircraft. Hence, decoupled longitudinal and lateral derivatives are associated with each aerodynamic coefficient as follows,

$$\begin{aligned}
C_X &= C_{x_0} + C_{x_\alpha}\alpha + C_{x_q}\frac{\bar{c}}{2V_T}q + C_{x_{\delta_e}}\delta_e && \text{(184a)}\\
C_Y &= C_{y_0} + C_{y_\beta}\beta + C_{y_p}\frac{b}{2V_T}p + C_{y_r}\frac{b}{2V_T}r + C_{y_{\delta_a}}\delta_a + C_{y_{\delta_r}}\delta_r && \text{(184b)}\\
C_Z &= C_{z_0} + C_{z_\alpha}\alpha + C_{z_q}\frac{\bar{c}}{2V_T}q + C_{z_{\delta_e}}\delta_e && \text{(184c)}\\
C_\ell &= C_{\ell_0} + C_{\ell_\beta}\beta + C_{\ell_p}\frac{b}{2V_T}p + C_{\ell_r}\frac{b}{2V_T}r + C_{\ell_{\delta_a}}\delta_a + C_{\ell_{\delta_r}}\delta_r && \text{(184d)}\\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{\bar{c}}{2V_T}q + C_{m_{\delta_e}}\delta_e && \text{(184e)}\\
C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_T}p + C_{n_r}\frac{b}{2V_T}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r && \text{(184f)}
\end{aligned}$$

The stability and control derivatives are unknown parameters to be identified, so the model structure in Eqs. 184 yields the following linear parameterization for each aerodynamic coefficient,

$$z = \boldsymbol{\theta}^\mathsf{T}\mathbf{x}, \tag{185}$$

where, z is a measurement scalar and can be obtained from Eq. (181) or (183) for each aerodynamic coefficient, $\boldsymbol{\theta}$ is a unknown parameter vector, and $\mathbf{x}$ is a regressor matrix of which entries are comprised of the measurements from sensors along with the right-hand-side of Eqs. (184).

The problem of batch identification is to find $\boldsymbol{\theta}$ that satisfies Eq. (185). A best fit is obtained using a batch process utilizing a set of measurements $z(k)$ and $\mathbf{x}(k)^\mathsf{T}$ as follows

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(1)^\mathsf{T} \\ \mathbf{x}(2)^\mathsf{T} \\ \vdots \\ \mathbf{x}(n)^\mathsf{T} \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(n) \end{bmatrix}. \tag{186}$$

The best fit is obtained from the least-square optimization problem,

$$\min_{\boldsymbol{\theta}} \|\mathbf{z} - \mathbf{X}\boldsymbol{\theta}\|^2, \tag{187}$$

whose solution is

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}\mathbf{z}. \tag{188}$$

C.3.2 Identification from flight test data

Flight test data were collected during maneuvers initiated from a trim condition of steady and straight level flight by exciting each control surface. Doublet-like open-loop commands to each control surface were issued by a remote pilot for each case. Upon the assumption of decoupled longitudinal and lateral dynamics, the aileron or rudder doublets were executed for identifying the lateral coefficients (C_Y, C_ℓ, and C_n) whereas the elevator doublet commands were executed for identifying the longitudinal coefficients (C_m, C_X, and C_Z). The sensor measurements were sampled at 20 [Hz] and were processed a posteriori to remove measurement noises or biases. For estimating the aerodynamic angles and attitude angles for building the regressor matrices, the estimation filters which were discussed in Section C.2 and Appendix B were utilized for off-line calculation of those variables.

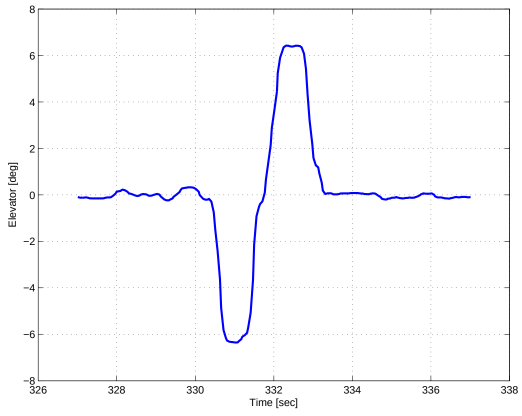
Two typical sets of measured (or estimated) states for longitudinal and lateral identification are shown in Figs. 80 and 81. The initial regression models given in Eq. (184) were first attempted for each identification, however, after several trials it was realized that slight modifications on the regression model structures in Eq. (184) result in better identification results. The modified regression model is then rearranged as follows,

$$
\begin{aligned}
C_X &= C_{x_0} + C_{x_q}\frac{\bar{c}}{2V_T}q + C_{x_{\delta_e}}\delta_e, && \text{(189a)}\\
C_Y &= C_{y_0} + C_{y_\beta}\beta + C_{y_p}\frac{b}{2V_T}p + C_{y_r}\frac{b}{2V_T}r, && \text{(189b)}\\
C_Z &= C_{z_0} + C_{z_\alpha}\alpha + C_{z_{\dot\alpha}}\frac{\bar{c}}{2V_T}\dot\alpha + C_{z_q}\frac{\bar{c}}{2V_T}q, && \text{(189c)}\\
C_\ell &= C_{\ell_0} + C_{\ell_\beta}\beta + C_{\ell_p}\frac{b}{2V_T}p + C_{\ell_{\delta_a}}\delta_a, && \text{(189d)}\\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\dot\alpha}}\frac{\bar{c}}{2V_T}\dot\alpha + C_{m_q}\frac{\bar{c}}{2V_T}q + C_{m_{\delta_e}}\delta_e, && \text{(189e)}\\
C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_T}p + C_{n_r}\frac{b}{2V_T}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r. && \text{(189f)}
\end{aligned}
$$

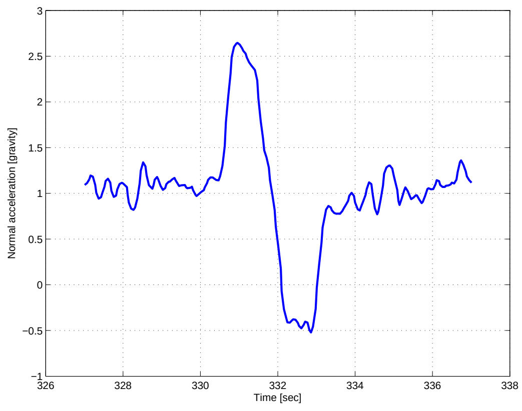
Several data sets have been selected for each identification process to assure the correctness of the results. Subsequently, the identified parameters are found to be consistent in each case within a reasonable bound of the estimation error. The identified parameter estimates for longitudinal coefficients are shown in Tables 14-16. Note that the error value given inside the brackets represents the 1-σ standard deviation of the estimation error which provides the confidence interval of the estimate. The longitudinal coefficients result mostly in good agreement as the numerical models were validated against experiment data taken from different maneuvers shown in Figs. 82-84. Tables 17-19 summarize the identified parameter estimates for lateral coefficients.

C.4 Hardware in the Loop Simulation Development

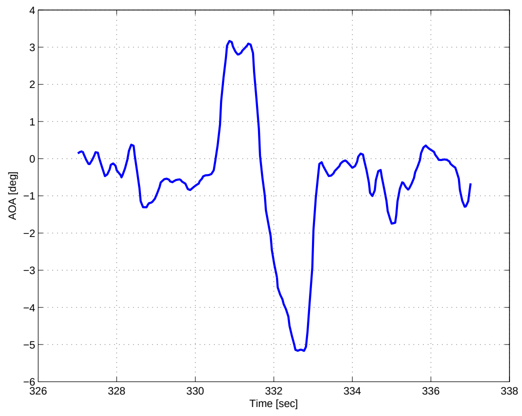
This section describes the details of developing a realistic simulation environment based on Matlab/ Simulink®. A complete 6-DOF nonlinear aircraft model with a



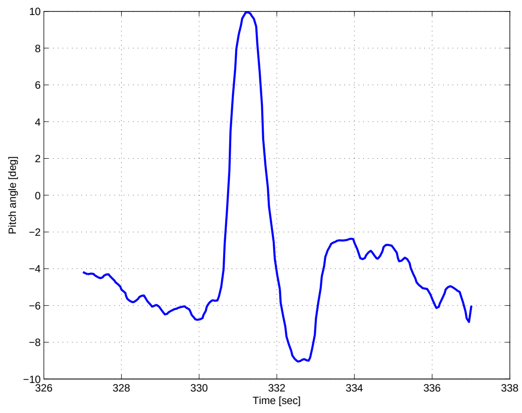
(a) Elevator command



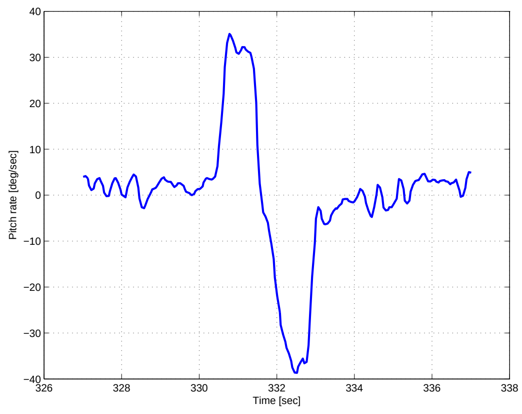
(b) Normal acceleration



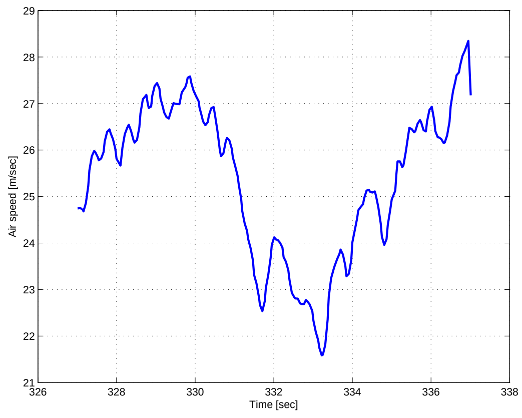
(c) Angle of attach



(d) Pitch angle

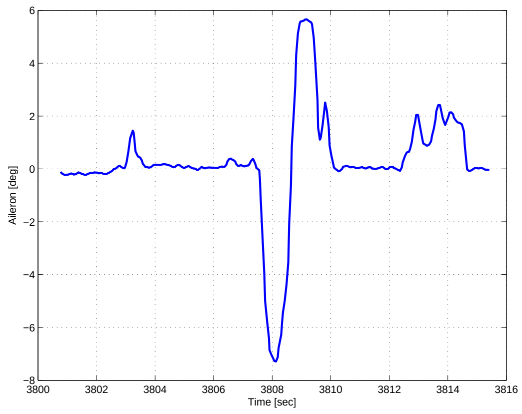


(e) Pitch rate

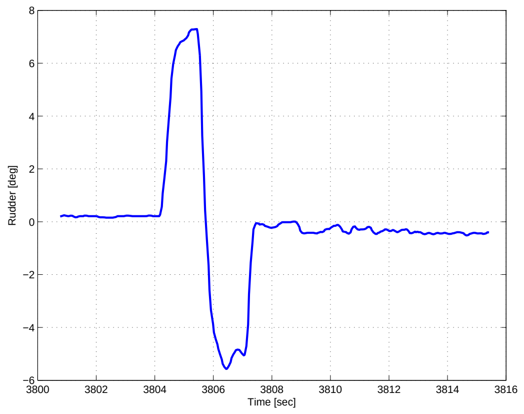


(f) Air speed

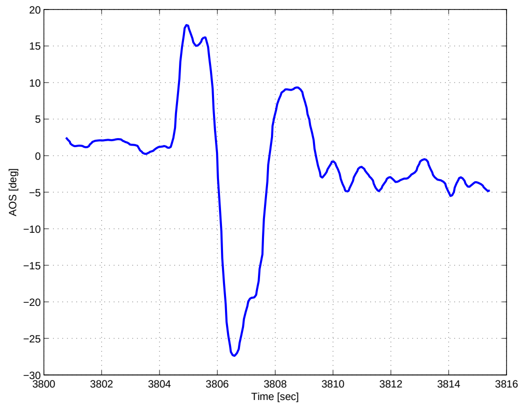
Figure 80: Measured state variables during a longitudinal maneuver.



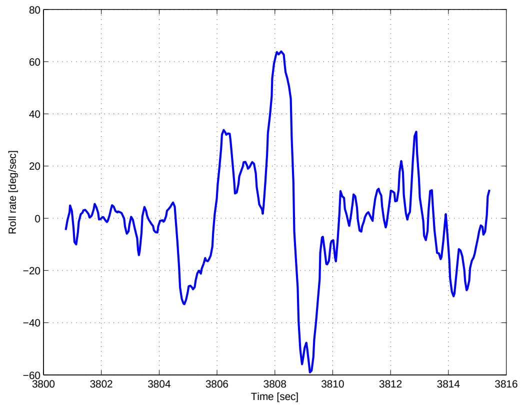
(a) Aileron command



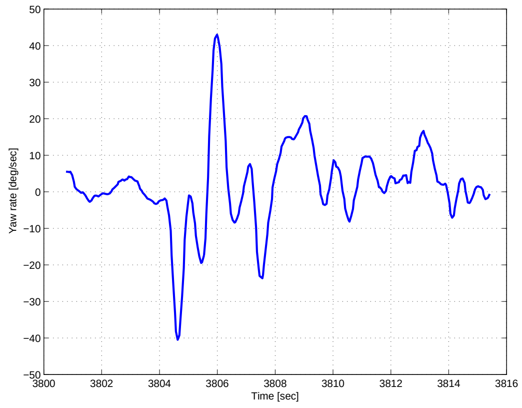
(b) Rudder command



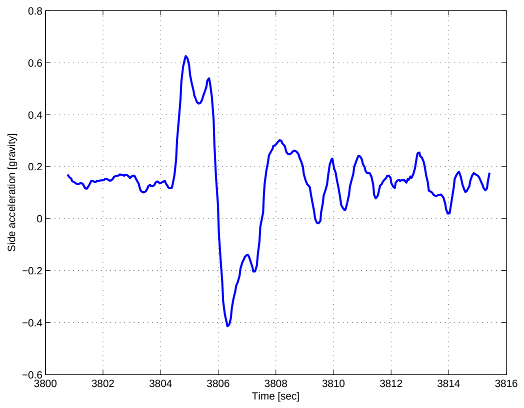
(c) Sideslip angle



(d) Roll rate



(e) Yaw rate



(f) Side acceleration

Figure 81: Measured state variables during a lateral maneuver.

Table 14: Body x-axis aerodynamic force coefficient (C_X) identification results.

Estimates [Error, %]	C_{x0}	C_{x_q}	$C_{x_{\delta_e}}$
Case 1	-9.441e-3 [3.1]	-13.64 [7.9]	-0.5413 [7.4]
Case 2	-9.576e-3 [3.1]	-14.66 [5.6]	-0.6207 [4.7]
Case 3	-7.331e-3 [6.1]	-19.46 [7.4]	-0.8219 [6.2]
Case 4	7.348e-3 [6.2]	-15.34 [6.3]	-0.6543 [5.3]
Case 5	-7.747e-3 [5.6]	-17.38 [5.7]	-0.7881 [4.7]

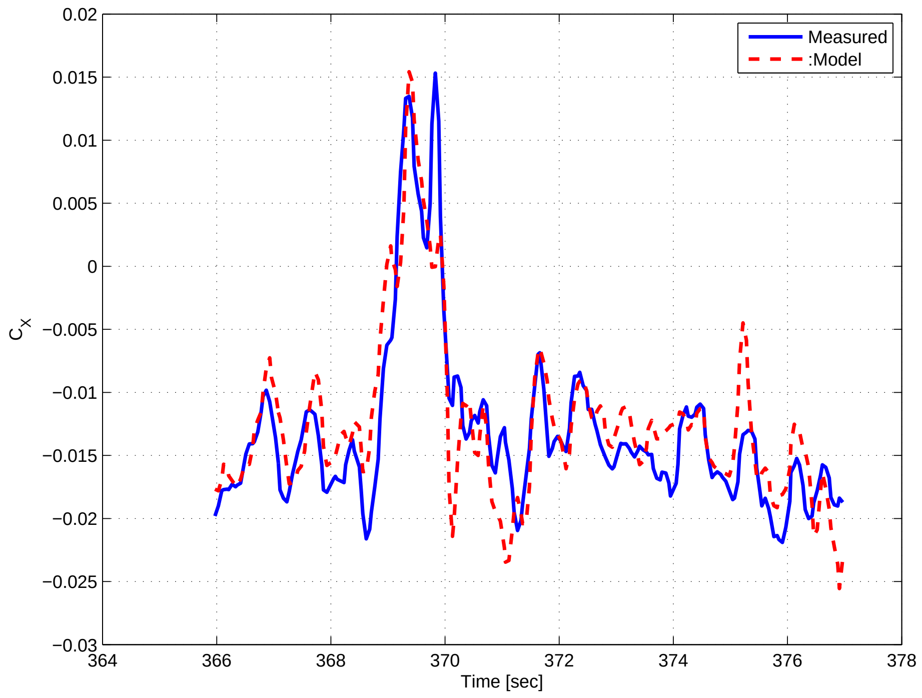


Figure 82: Validation for the force coefficient C_X.

Table 15: Body z-axis aerodynamic force coefficient (C_Z) identification results.

Estimates [Error, %]	C_{z0}	C_{z_α}	C_{z_q}	$C_{z_{\dot{\alpha}}}$
Case 1	-0.2522 [2.3]	-2.746 [8.0]	-42.03 [9.0]	81.3 [4.9]
Case 2	-0.2588 [1.4]	-2.874 [5.9]	-43.94 [7.4]	77.91 [4.7]
Case 3	-0.2670 [1.2]	-2.848 [4.6]	-43.86 [5.7]	63.4 [4.7]
Case 4	-0.2309 [3.0]	-2.561 [9.3]	-49.08 [9.3]	64.67 [8.9]
Case 5	-0.2609 [1.4]	-3.173 [5.2]	-45.84 [8.6]	72.71 [5.9]

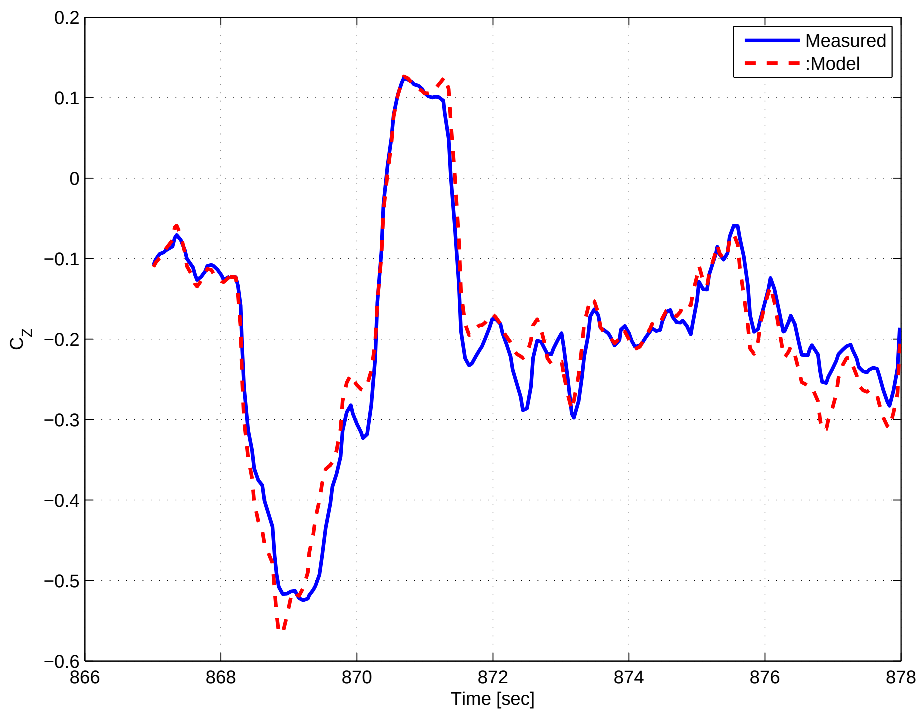


Figure 83: Validation for the force coefficient C_Z.

Table 16: Body y-axis aerodynamic moment coefficient (C_m) identification results.

Estimates [Error, %]	C_{m_0}	C_{m_α}	C_{m_q}	$C_{m_{\dot\alpha}}$	$C_{m_{\delta_e}}$
Case 1	-1.09e-3 [13.1]	-0.1963 [5.1]	-3.962 [13.1]	4.941 [4.6]	-0.2639 [7.4]
Case 2	-1.58e-3 [12.6]	-0.2073 [6.6]	-3.758 [17.6]	5.069 [6.0]	-0.2442 [9.8]
Case 3	-2.15e-3 [14.7]	-0.2757 [6.4]	-2.664 [18]	4.281 [8.2]	-0.2063 [7.2]
Case 4	-1.46e-3 [28.3]	-0.2636 [6.0]	-4.323 [14.7]	5.224 [5.4]	-0.244 [6.3]
Case 5	-5.80e-4 [104.8]	-0.1976 [9.6]	-4.567 [19.5]	5.747 [6.2]	-0.2086 [11.2]

Table 17: Body y-axis aerodynamic force coefficient (C_Y) identification results.

Estimates [Error, %]	C_{y_0}	C_{y_β}	C_{y_p}	C_{y_r}
Case 1	-2.645e-2 [2.8]	-0.1275 [5.7]	1.144 [10.1]	1.039 [10.2]
Case 2	-2.946e-2 [1.9]	-0.1485 [4.7]	0.5335 [14.0]	0.8015 [12.5]
Case 3	-2.566e-2 [3.6]	-0.1825 [3.7]	0.6247 [18.6]	0.7222 [15.4]
Case 4	-2.597e-2 [4.5]	-0.1634 [5.2]	1.151 [16.1]	0.8016 [16.4]
Case 5	-2.786e-2 [3.3]	-0.1561 [4.5]	1.036 [11.7]	0.8627 [15.2]

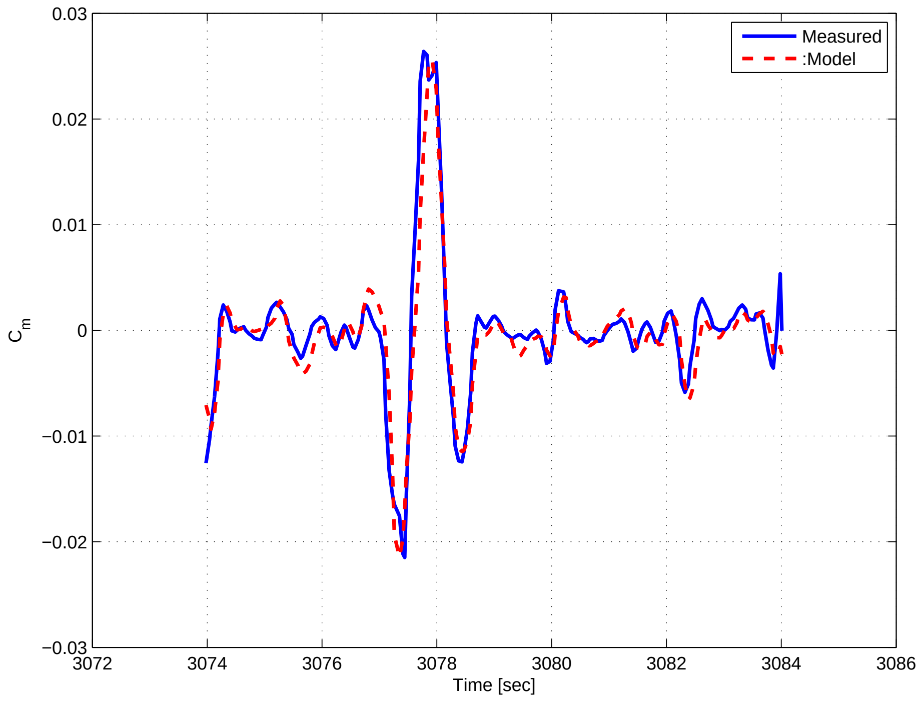


Figure 84: Validation for the moment coefficient C_m.

Table 18: Body x-axis aerodynamic moment coefficient (C_ℓ) identification results.

Estimates [Error, %]	C_{ℓ_0}	C_{ℓ_β}	C_{ℓ_p}	$C_{\ell_{\delta_a}}$
Case 1	-3.652e-4 [10.1]	-1.305e-2 [4.9]	-1.239e-2 [36.7]	-1.362e-2 [15.2]
Case 2	-2.629e-4 [17.2]	-9.19e-3 [8.2]	-2.566e-2 [25.5]	-1.517e-2 [18.5]
Case 3	3.90e-4 [11.9]	-7.61e-3 [6.6]	-3.499e-2 [19.1]	-1.752e-2 [16.8]

Table 19: Body z-axis aerodynamic moment coefficient (C_n) identification results.

Estimates [Error, %]	C_{n_0}	C_{n_β}	C_{n_p}
Case 1	-7.571e-4 [11.1]	1.179e-2 [9.4]	-6.768e-2 [15.2]
Case 2	2.180e-4 [31.4]	5.986e-3 [16.7]	-0.1077 [10.5]
Case 3	-3.244e-4 [24.9]	1.029e-2 [13.7]	-3.968e-2 [27.1]
Case 4	1.822e-4 [53.3]	1.005e-2 [13.9]	-9.096e-2 [15.7]
Estimates [Error, %]	C_{n_r}	$C_{n_{\delta_a}}$	$C_{n_{\delta_r}}$
Case 1	-0.1322 [12.3]	-3.065e-2 [14.3]	-5.691e-2 [9.1]
Case 2	-6.577e-2 [22.5]	-4.902e-2 [10.2]	-4.107e-2 [11.9]
Case 3	-8.447e-2 [20.6]	-1.712e-2 [26.8]	-3.841e-2 [14.6]
Case 4	-9.061e-2 [17.3]	-4.827e-2 [13.0]	-5.221e-2 [10.5]

linear approximation of the aerodynamic forces and moments is used to simulate realistic dynamic behavior of the aircraft. The nonlinear aircraft model also involves the detail modeling of subsystems such as sensors and actuators using experimental data. In addition, the external pilot command input and the flight visualization enable the simulation to be used as a virtual flight test.

The actual autopilot hardware is placed inside the simulation loop in order to test and validate both the hardware and the on-board software. Four independent computer systems were used in the hardware-in-the-loop (HIL) simulation as illustrated in Fig. 85: the 6-DOF simulator, the flight visualization computer, the autopilot micro-controller, and the ground station computer console. A detail description for this setup is given below.

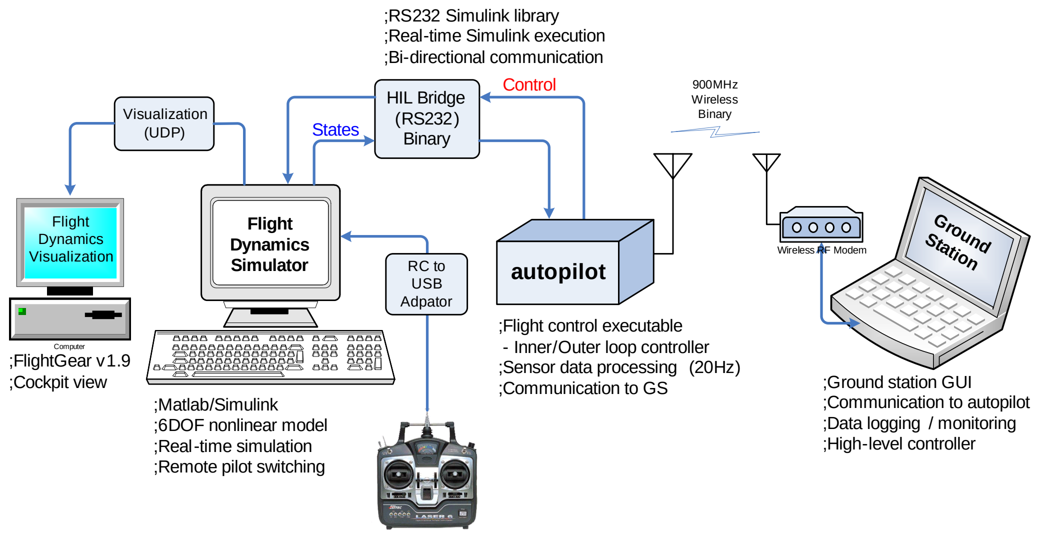


Figure 85: High fidelity hardware-in-the-loop (HIL) simulation environment.

C.4.1 A 6-DOF simulator

A full 6-DOF nonlinear model for the aircraft was built in Matlab/Simulink$^{®}$ environment. The 12 states differential equations of motion in Eqs. (163)-(166) are utilized in conjunction with an approximation of the aerodynamic forces and moments via component buildup. The stability and control derivatives are either obtained in the manner as discussed in Section C.1.2 or Section C.3. The stability and control derivatives from the geometry of the aircraft can be chosen during the early phase of development, however, once these derivatives have been identified experimentally, the identified values are used for realistic simulation results. The 6-DOF nonlinear model becomes more realistic by employing a standard atmospheric model, an Earth and gravity model (WGS-84), and an Earth magnetic field model[144]. The output states from the simulator are processed to emulate real sensors accounting for sensor latency, random walk bias, and measurement noise. After digitized according to the word size of the micro-controller (12 bit, 4096 steps), the sensor values are transmitted to the autopilot via a serial communication. In a similar manner, the navigation states are processed for data latency and measurement noise and encapsulated in a binary GPS packet identical to the actual GPS sensor output. This packet is also transmitted to the autopilot at low update rate (1 Hz) via the serial communication.

In the HIL environment, the autopilot functions identically as in a real flight test, while computing control commands to each control actuator. These commands are sent back to the 6-DOF simulator in order to drive the actuator model of the control surface. Each control surface was modeled as a first order system with a rate limiter and saturation limit, whose parameters were identified experimentally. Consequently, a simulation loop is formed between the 6-DOF simulator (software) and the autopilot (hardware) exchanging the simulated data back and forth.

C.4.2 User interface

An R/C transmitter is connected to the 6-DOF simulator for recording remote pilot stick commands: Four channels for control surface commands ($\delta_a, \delta_e, \delta_t$ and δ_r) and two auxiliary commands for toggling between autonomous control mode and remote pilot mode. The remote pilot stick commands can override the autopilot control command at any time by switching the commands to the actuator models at user's choice. In this case, a simulation for an open loop maneuver can be conducted along the remote pilot input to validate the dynamic characteristic of the aircraft model.

In order to visualize the simulation, we adopted the use of FlighGear[1], an open-source flight simulator for a visualization tool. The FlightGear is a flight simulator framework, which has been widely used in various research environment[107, 127, 36, 133]. Although it provides a total simulation environment in conjunction with the internal flight dynamics models, we used the FlightGear as a visualization tool combining with the 6-DOF simulator already developed. By doing this, the FlightGear receives the simulated states from the simulator at a fixed update rate to constantly refresh the virtual scenes that have been generated along the user's convenient view angles.

The ability to record the pilot stick command with visualizing via the FlightGear allows the simulation environment to replace a real experiment to save time and effort. An open loop behavior of the UAV can be tested and validated by a remote pilot via the user friendly input device, and the closed loop control performance can be verified and demonstrated by conducting virtual experiments in advance to the real flight test. Figure 86 shows a computer screen shot taken during the HIL simulation visualizing the dynamic behavior of the UAV, while displaying the corresponding state variables in the scopes.

C.4.3 Data communication and synchronization

The 6-DOF simulator exchanges the sensor and control command signals with the autopilot via a serial communication between the two. The sensor output data include 16 different sensor outputs from rate gyros, accelerometers, magnetometers, and etc., and are packed into a custom-designed binary packet. The binary packet begins with a designated header to distinguish between different packets and ends by a trailer for data consistency check. The serial communication is configured for a

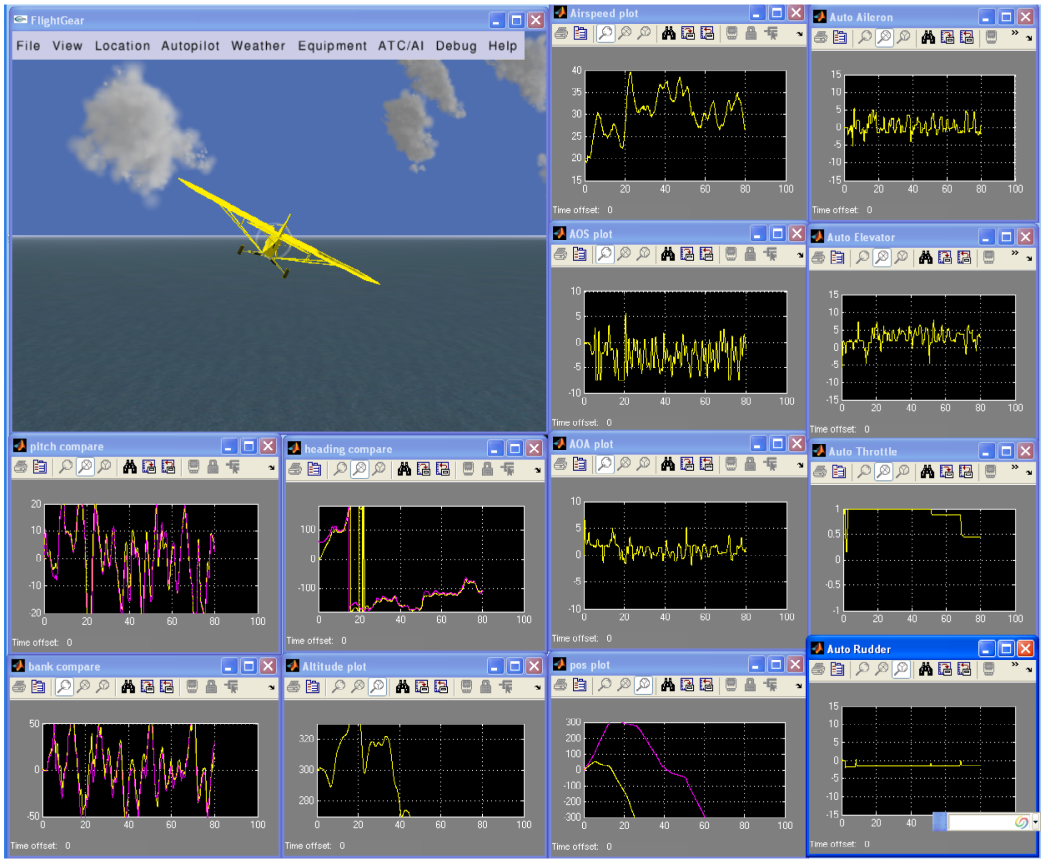


Figure 86: Hardware-in-the-loop simulation screen shot

baud rate 115200 bps, which allows communicating by transmitting the sensor packet of 39 bytes at 20 Hz as well as the GPS packet of 54 bytes at 1 Hz update rate. The control command data from the autopilot are composed of four PWM commands for each actuator, resulting in a binary packet of 11 bytes in length associated with header and trailer to be transmitted at 20 Hz update rate. The HIL bridge shown in Fig. 86 was then incorporated into the 6-DOF Simulink model to handle the bidirectional communication with a help of the serial communication block toolbox[82]. In addition, a dedicated S-function block is utilized to send simulated output to the FlightGear[144]. The S-function block bundles the simulated data with a user defined protocol (UDP) for small size of a data packet to ensure less communication overhead over the network. Real time visualization is then made possible unless the network latency is significant. To minimize the network latency, one should locate two hosts (for 6-DOF simulation and 3-D visualization) within a local area network (LAN), or if the computer resources permits one can configure to use a single computer for visualizing and running the 6-DOF simulation on the same machine.

Because the multiple systems are involved in the HIL simulation, it is all important to keep the execution timing among these systems properly synchronized during the entire simulation period for data compatibility among them. Besides, a real-time execution is necessary for the simulation to be in accord with the pilot input and the visualization at correct time step. Although the 6-DOF simulator was written in a Simulink model running at non real-time, a real time execution was accomplished by utilizing the Realtime block toolbox[83] in the simulation. This toolbox is based on the simple concept that if the current hardware cycle time is lower than the desired simulation step time, the block simply hold the current execution until the desired time step is triggered. For a real time simulation, the basic simulation time step of the 6-DOF simulator is carefully chosen at 100 [Hz] taking into consideration both the computational overhead and the faithful simulation results for accurate dynamic characteristics of the UAV. A different sampling rate at 20 [Hz] was utilized for the serial communication block and the FlightGear S-function block. The synchronization of the data was implicitly dealt with in a manner that the main simulation loop of the 6-DOF simulator polls each communication block at a fixed interval during the simulation, whereas the autopilot and the FlightGear receive the packets passively.

C.5 Summary

A hardware-in-the-loop simulation environment has been built to validate hardware and software development of the autopilot for a small UAV. A full 6-DOF nonlinear dynamic model has been used in conjunction with the linear approximation of the aerodynamic forces and moments. A batch parameter identification method has been used to determine the stability and control derivatives of the UAV using flight test data. Detailed models for the sensors and actuators were incorporated in the simulation along with the capability of real-time 3-D visualization and external pilot interface. The hardware-in-the-loop simulation is an indispensable tool for performing simulated flight tests in support of avionics development and validation of control laws for small UAVs with minimal cost and effort.

APPENDIX D

DESIGN OF INNER CONTROL LOOPS

D.1 Decoupled linear model

For the design purpose of inner loop controls of the UAV, we employ a decoupled linear model, which has been built from stability and control derivatives. The stability and control derivatives of the UAV utilized in this research have been experimentally identified through the approach described in Appendix C. The resulting linear model is composed of two linear systems which describe the longitudinal and lateral dynamics of the UAV:

Longitudinal dynamics

$$\dot{x}_{\text{Long}} = A_{\text{Long}} x_{\text{Long}} + B_{\text{Long}} u_{\text{Long}}, \tag{190}$$

where the state vector x_{Long} and the control input u_{Long} are defined by

$$x_{\text{Long}} = [\Delta\alpha \; \Delta q \; \Delta v_T \; \Delta\theta]^{\mathsf{T}}, \qquad u_{\text{Long}} = [\Delta\delta_e \; \Delta\delta_t]^{\mathsf{T}},$$

and the symbol Δ denotes a perturbed representation of the corresponding variables. The system matrices were obtained from the stability and control derivatives,

$$A_{\text{Long}} = \begin{bmatrix} -7.1264 & 0.9497 & -0.0485 & 0.0 \\ -122.8676 & -4.2920 & -0.0261 & 0.0 \\ 4.1512 & -0.0398 & -0.1935 & -9.81 \\ 0 & 1.0000 & 0.0 & 0.0 \end{bmatrix}, \tag{191a}$$

$$B_{\text{Long}} = \begin{bmatrix} -0.2953 & -0.0174 \\ -83.4069 & 0.8221 \\ -1.3732 & 6.1044 \\ 0.0 & 0 \end{bmatrix}. \tag{191b}$$

Lateral dynamics

$$\dot{x}_{\text{Lat}} = A_{\text{Lat}} x_{\text{Lat}} + B_{\text{Lat}} u_{\text{Lat}}, \tag{192}$$

where the state vector x_{Lat} and the control input u_{Lat} are defined by

$$x_{\text{Lat}} = [\Delta\beta \ \Delta\phi \ \Delta p \ \Delta r]^{\mathsf{T}}, \quad u_{\text{Lat}} = [\Delta\delta_a \ \Delta\delta_r]^{\mathsf{T}},$$

and the system matrices were obtained from the identified stability and control derivatives as follows,

$$A_{\text{Lat}} = \begin{bmatrix} -0.3301 & 0.4897 & 0.0570 & -0.9939 \\ 0.0 & 0.0 & 1.0 & 0.0570 \\ -35.4688 & 0.0 & -16.8528 & 0.1524 \\ 2.7315 & 0.0 & -1.1105 & -0.5340 \end{bmatrix}, \tag{193a}$$

$$B_{\text{Lat}} = \begin{bmatrix} 0.0 & 0.1494 \\ 0.0 & 0 \\ -182.2395 & 9.2613 \\ -9.4423 & -7.7841 \end{bmatrix}. \tag{193b}$$

The servo actuators were modeled as first-order systems with a corner frequency of 10 [rad/sec] as identified experimentally. By convention, the positive control surface deflections are supposed to generate negative aerodynamic moments about each corresponding axis (roll moment by aileron, pitch moment by elevator, and yaw moment by rudder). In order to use a negative feedback structure of the classical control design method, we define different control inputs u_a, u_e, and u_r for aileron, elevator, and rudder, respectively. Subsequently, we take into account not only the actuator dynamics but also the sign convention of control inputs to come up with the actuator models as follows,

$$\Delta\delta_a = \tfrac{-10}{s+10}\Delta u_a, \tag{194a}$$

$$\Delta\delta_e = \tfrac{-10}{s+10}\Delta u_e, \tag{194b}$$

$$\Delta\delta_t = \tfrac{10}{s+10}\Delta u_t, \tag{194c}$$

$$\Delta\delta_r = \tfrac{-10}{s+10}\Delta u_r. \tag{194d}$$

D.2 Longitudinal controllers design

In this section, we describe a detail design of inner loop controllers for the longitudinal dynamics of the UAV. The longitudinal controllers comprise a pitch controller and a speed controller, which calculate the elevator command input (Δu_e) and the throttle command input (Δu_t), respectively.

D.2.1 Pitch controller

We design the pitch controller starting from a pitch damper. The open-loop transfer function from the elevator command input (Δu_e) to the pitch rate (Δq) is obtained from Eqs. (191) and (194b) as follows,

$$\frac{\Delta q}{\Delta u_e} = \frac{834.07s(s+6.645)(s+0.2394)}{(s+10)(s+5.721 \pm 10.706i)(s+0.085 \pm 0.6141i)}, \tag{195}$$

which reveals the natural frequency of the phugoid mode is 0.6199 [rad/sec] (damping ratio is 0.1371) and the natural frequency of the short period mode is 12.1391 [rad/sec] (damping ratio is 0.4713). It also shows the natural frequency of the servo actuator by 10 [rad/sec] (damping ratio is 1.0). Figure 87 shows a root locus plot of this transfer function with a unity negative feedback of the pitch rate with a positive gain k_q, which shows that the short period poles get poorly damped as k_q increases towards infinity. In order to improve the damping characteristics of the short period mode, a lead compensator is added in the feedback path with a compensator pole of $s = -16$ to minimize the influence of the compensator pole on the open loop plant.

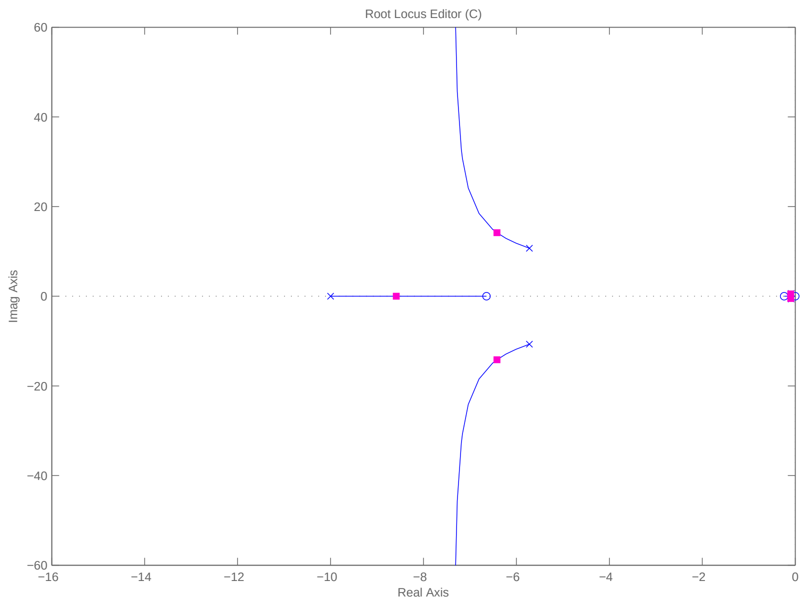


Figure 87: Root locus plot of the transfer function in Eq. (195) with a unity negative feedback of the pitch rate Δq with the positive gain k_q. As the gain increases, the short period mode gets poorly damped.

The pole/zero ratio is carefully chosen by eight, hence the compensator is given by

$$G_q(s) = k_q \frac{s+2}{s+16}, \tag{196}$$

where the compensator gain is chosen $k_q = 0.101$ in order for the short period mode damping to be close to the critical damping ratio. The closed loop transfer function from a commanded pitch rate (Δq_r) to the pitch rate (Δq) after closing the lead compensator is calculated as follows,

$$\frac{\Delta q}{\Delta q_r} = \frac{834.07s(s+16)(s+0.2394)(s+6.645)}{(s+9.145 \pm 12.196i)(s+9.578 \pm 3.844i)(s+0.083 \pm 0.599i)}. \tag{197}$$

Figure 88 shows the root locus plot incorporating the designed lead compensator in the feedback path. With the lead compensator, the phugoid mode does not change much, yet the short period mode become properly damped as resulting in the natural frequency of 15.2441 [rad/sec] and the damping ratio of 0.6.

Having designed the pitch damper, we design a pitch attitude controller that forms an outer loop of the system given in Eq. (197). To this end, a transfer function from

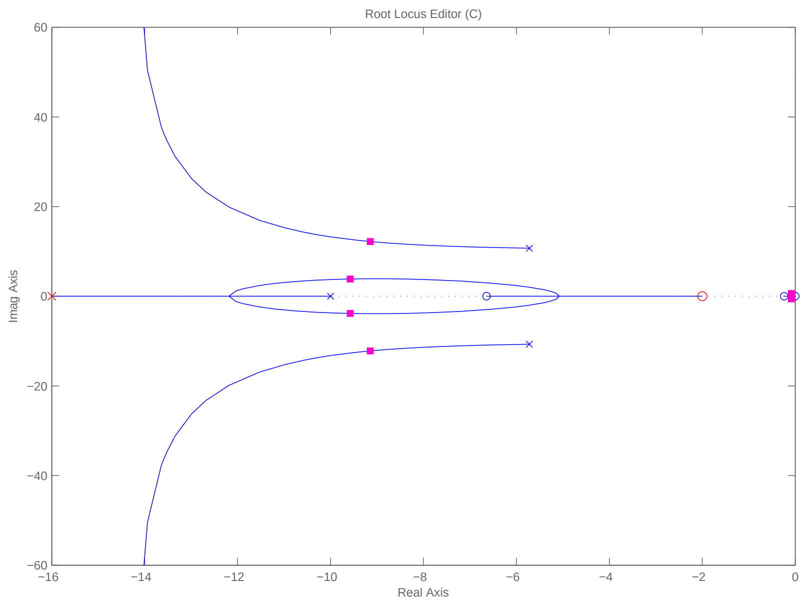


Figure 88: Root locus plot of the closed-loop system incorporating the lead compensator of the pitch rate. The closed-loop poles and zeros are shown, and the squares represent the location of the poles at $k_q = 0.101$.

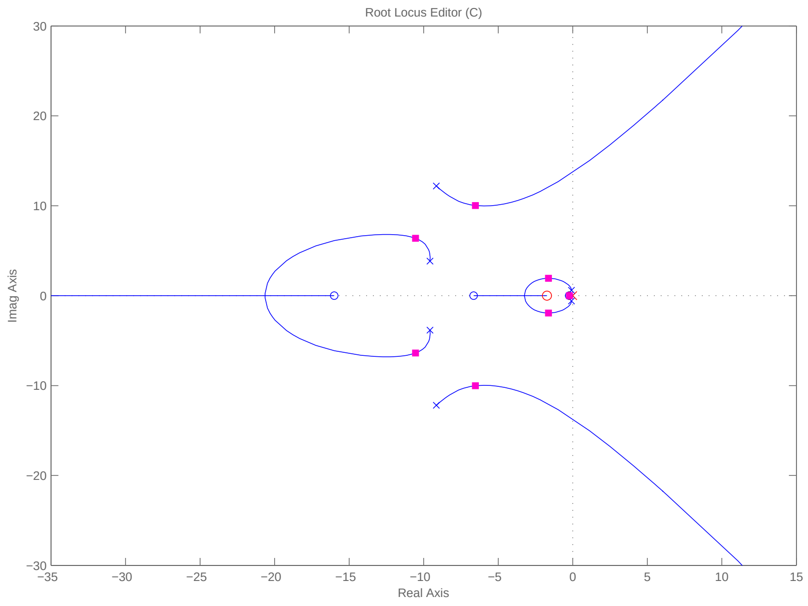
the commanded pitch rate (Δq_r) to the pitch angle ($\Delta\theta$) is obtained by

$$\frac{\Delta\theta}{\Delta q_r} = \frac{834.07(s+16)(s+0.2394)(s+6.645)}{(s+9.145\pm 12.196i)(s+9.578\pm 3.844i)(s+0.083\pm 0.599i)}. \tag{198}$$

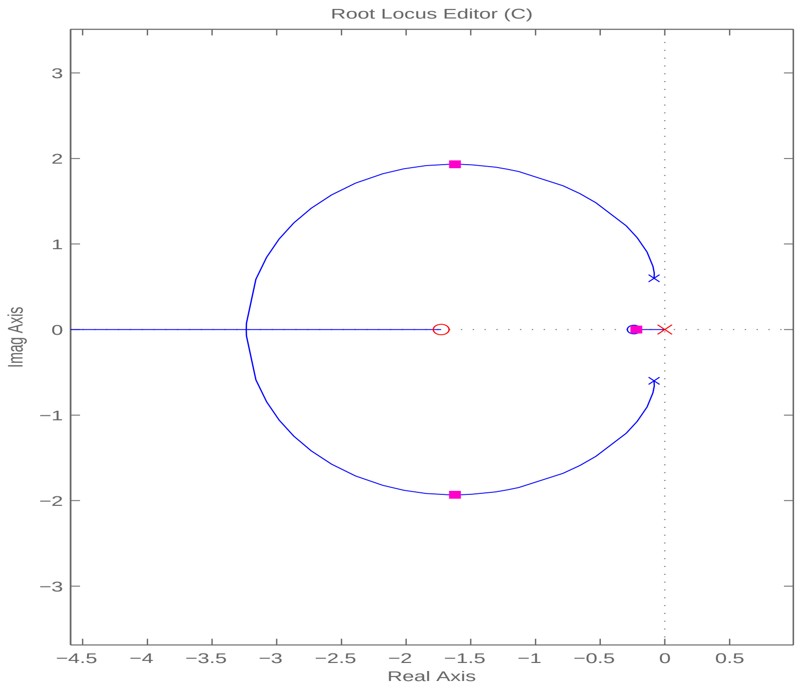
The transfer function appears to be a *Type-0* system that has no integrator pole in the characteristic equation. Hence, in order to remove a steady state error of the pitch angle we design a proportional integral (PI) controller. The PI controller with the zero located at $s = -1.73$ is given as follows,

$$G_\theta(s) = k_\theta \frac{s+1.73}{s}, \tag{199}$$

yields the phugoid poles are located at $s \approx -1.6 \pm 1.9i$ of the closed-loop system with the proportional gain $k_\theta = 0.855$, which provides a critical damping of phugoid mode (see Fig. 89(b) for the root locus plot of the closed-loop system). Figure 89(a) shows the entire root locus plot of the closed-loop system, where the short period poles are also properly damped. In addition, the Bode plot of the closed-loop system is displayed in Fig. 90, which shows that the closed-loop system has the gain margin of 12.4 [dB] and the phase margin of 51.6 [deg].



(a) Entire display of the root locus plot of the closed-loop system.



(b) Magnified display of the root locus plot showing the phugoid poles around the origin.

Figure 89: Root locus plot of the closed-loop system with the pitch angle PI controller as k_θ varies. The squares represent the closed-loop poles at $k_\theta = 0.855$.

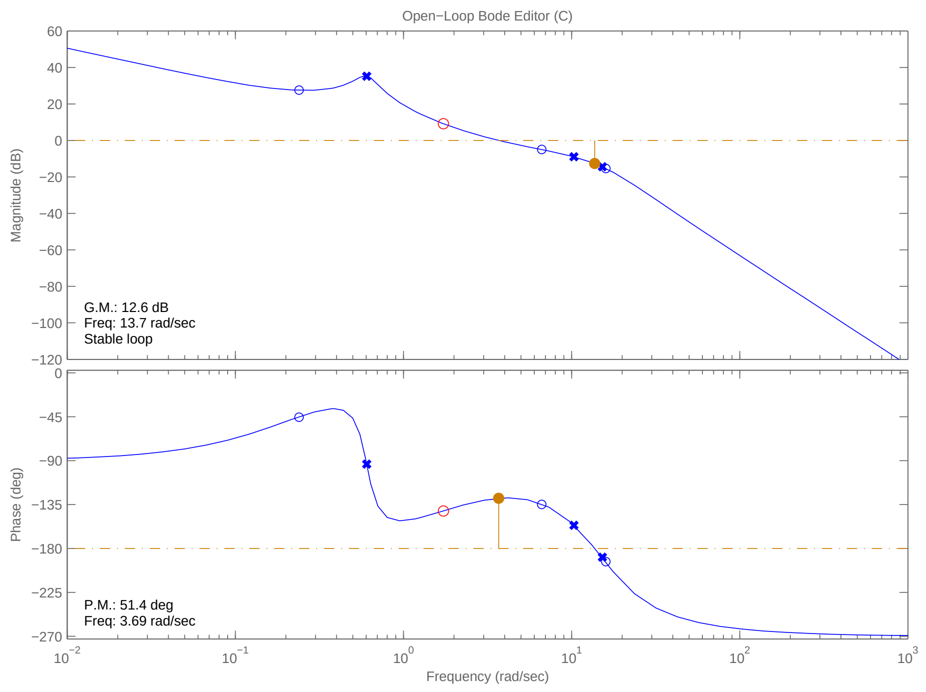


Figure 90: Bode plot of the closed-loop system with the pitch angle PI controller.

The final closed-loop transfer function from the pitch angle reference command ($\Delta\theta_r$) to the pitch angle ($\Delta\theta$) becomes,

$$\frac{\Delta\theta}{\Delta\theta_r} = \frac{713.13(s+0.24)(s+1.73)(s+6.65)(s+16)}{(s+0.22)(s+1.67\pm1.94i)(s+6.41\pm10.01i)(s+10.62\pm6.42i)}, \quad (200)$$

which reveals that the natural frequency of the phugoid mode ends up with 2.5539 [rad/sec] (damping ratio 0.6541), and the natural frequency of the short period mode becomes 11.8813 [rad/sec] (damping ratio 0.5392). Figure 91 illustrates the final implementation of the pitch attitude controller including the pitch damper. The proportional term of the PI controller is fed from the feedforward signal $\Delta\theta_e$ as drawn by a dashed line and is added together with integral term. A step response of the closed-loop system by a unit pitch reference command is shown by a dashed line in Fig. 92. The performance of the PI controller is given by an overshoot 120% and a zero steady state error with the settling time of 2.9 [sec].

By the PI implementation discussed above, the integrator pole of the controller introduces an additional zero in the closed-loop system. Excessive overshoot arises

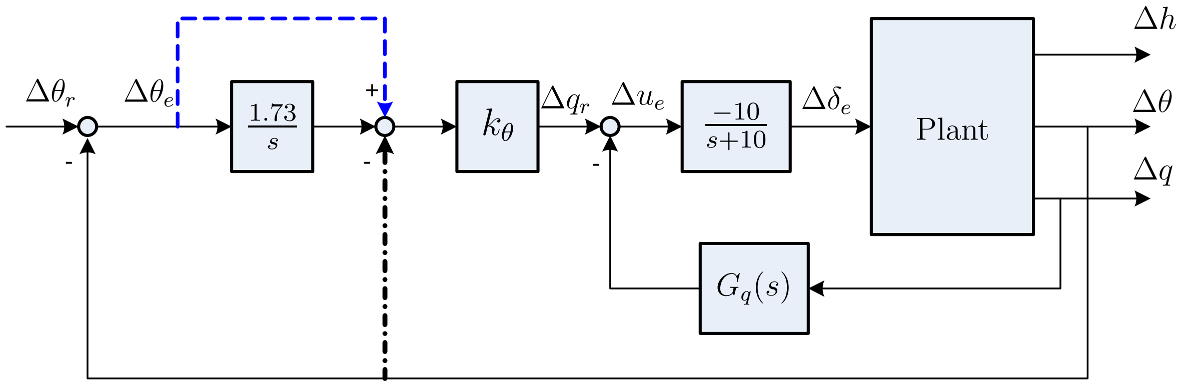


Figure 91: Block diagram of the closed-loop pitch angle controller. The dashed line denotes the original PI controller implementation, whereas the dashed-dot line denotes an alternative PI implementation with closed-loop zero removed.

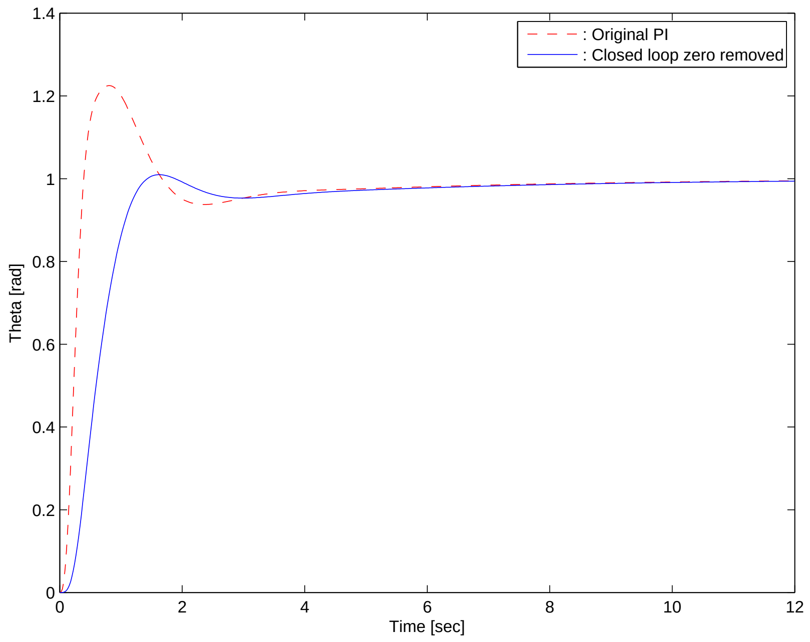


Figure 92: Step response of the closed-loop system by a unit pitch reference command.

from this closed-loop zero, thus degrading the overall performance (see Fig.92 for the dashed response). On the other hand, an alternative PI implementation which does not induce a closed-loop zero can be incorporated in the manner that the proportional term of the PI is fed from the feedback signal of $\Delta\theta$[137] as shown in Fig. 91 by a dashed-dot line. By this implementation, it is easy to verify that the characteristic equation of the closed-loop remains same, while the zero induced from the integrator pole is removed from the closed-loop system. The step response of this implementation is shown in Fig. 92, where the overshoot of the step response is reduced at the cost of slow rise time while attaining approximately same settling time. Consequently, since the latter PI implementation (no closed-loop zero) results in the tight control bound with a smooth transition, it is likely to be a better choice for the actual implementation.

D.2.2 Altitude Controller

After closing the inner loop pitch controllers discussed in the previous section, we obtain an open-loop plant for the purpose of altitude controller design. Incorporating the PI controller implementation that removes the closed-loop zero, we obtain a transfer function from the pitch reference angle command ($\Delta\theta_r$) to the altitude (Δh) as follows,

$$\frac{\Delta h}{\Delta\theta_r} = \frac{-87.37(s+0.14)(s+16)(s+39.42)(s-49.59)}{s(s+0.20)(s+1.76\pm 1.93i)(s+6.44\pm 9.95i)(s+10.51\pm 6.61i)}. \tag{201}$$

It follows from Eq. (201) that due to a non-minimum zero in the transfer function, it is anticipated that an excessive proportional feedback gain might cause instability of the system. Hence, in order to improve the performance of the transient response while ensuring the stability of the system, a lead compensator is adopted as follows,

$$G_{hL}(s) = 0.3\frac{s+2.5}{s+20}, \tag{202}$$

which results in the gain margin 13.1 [dB] and the phase margin 79.8 [deg] of the closed-loop system. The lead compensator improves the transient response, however, it reduces the low frequency gain by -27 dB. As a result, a steady-state error to a unit ramp input will grow accordingly. In order to boost the low frequency gain and thus reduce the steady-state error, a PI compensator is adopted, albeit it adds small phase lag in the higher frequency range, as follows

$$G_h(s) = k_h \frac{(s + 0.495)}{s}, \tag{203}$$

where the proportional gain of the PI controller is chosen $k_h = 1.22$.

Figure 94 is the root locus plot of the altitude feedback using the lead and the PI compensators as k_h varies. With the chosen gain $k_h = 1.22$, the final closed-loop transfer function from the altitude reference command (Δh_r) to the altitude (Δh) yields,

$$\frac{\Delta h}{\Delta h_r} = \frac{-31.98(s+0.14)(s+0.495)(s+2.5)(s+16)(s+39.42)(s-49.59)}{s(s+0.15)(s+0.54 \pm 0.51i)(s+1.0 \pm 1.96i)(s+6.49 \pm 10.14i)(s+10.71 \pm 6.49i)(s+19.99)}, \tag{204}$$

where the phugoid poles are located at $s = -0.54 \pm 0.51i$, which gives the natural frequency of 0.742 [rad/sec] with the damping ratio of 0.731. The gain and phase margin plot of the closed-loop system is shown in Fig. 93, giving the margins by 9.31 [dB] and 45 [deg], respectively. From the step response shown in Fig. 95, the closed-loop system has a fast rise time but an undesirable overshoot. It follows from the arguments in the previous section, the excessive overshoot is caused by the induced closed-loop zero due to the PI controller. This overshoot is then relaxed by implementing the other PI controller, where the altitude signal is fed back before/after the the integrator block. The closed-loop system ends up with the same closed-loop characteristic equations yet the closed-loop zero removed. Hence, as shown in Fig. 95, the overshoot performance is improved by 110% compared to 130% with no closed-loop zero removal, by sacrificing the transient performance with the increased rise time about 2.5 [sec].

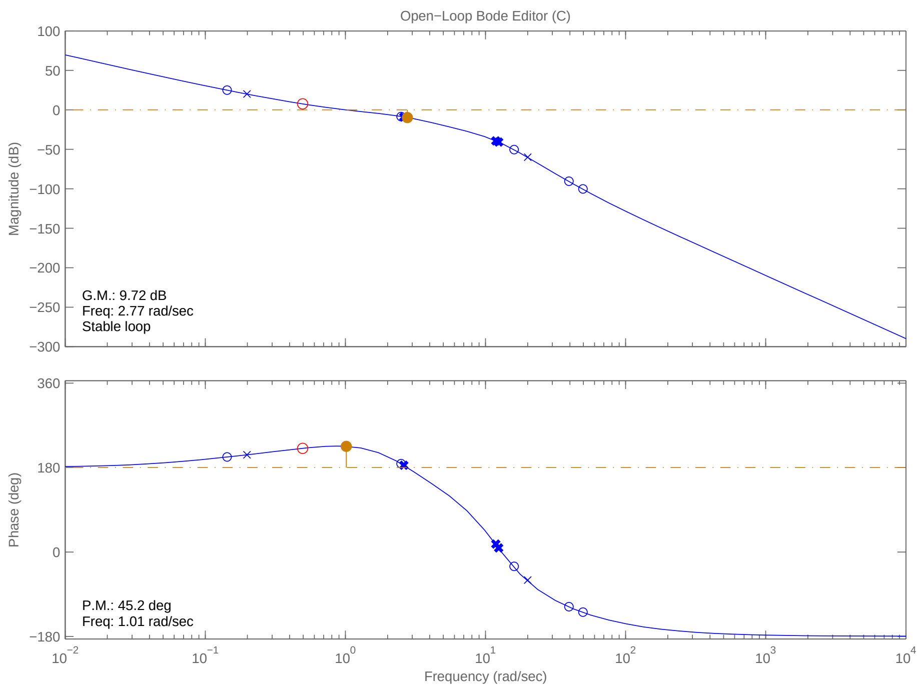
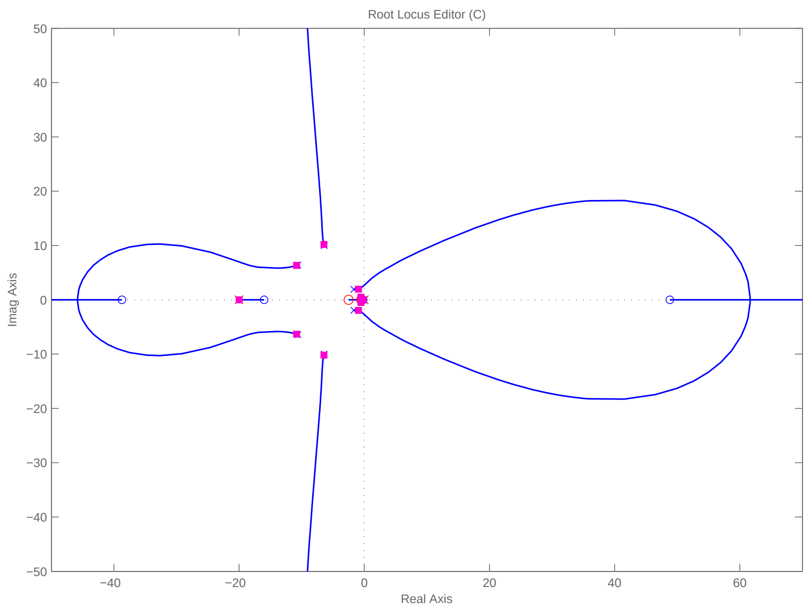


Figure 93: Bode plot of the closed-loop system with the lead compensator.

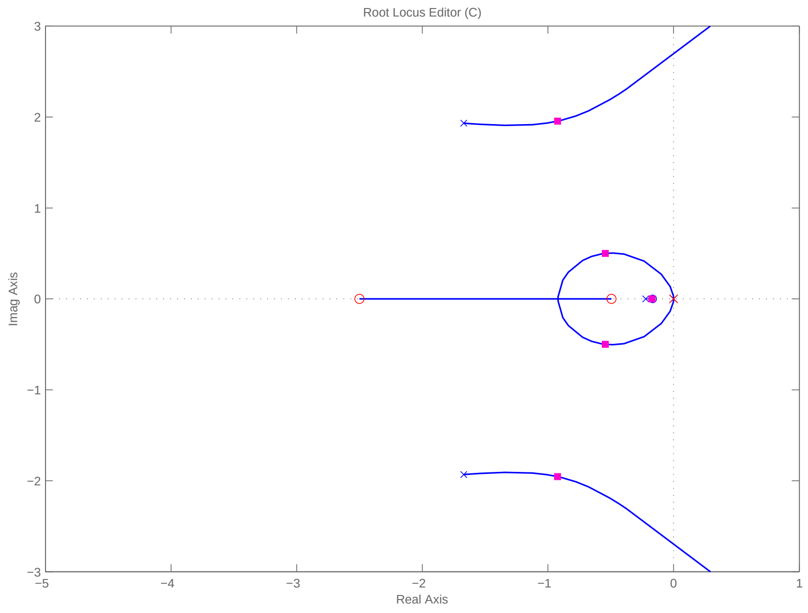
D.2.3 Integrator antiwindup

The altitude controller computes the pitch reference command for the inner loop pitch angle controller. In order to prevent the reference command from exceeding the acceptable range of the pitch controller, we put a saturation block after the altitude controller. The saturation limit for the pitch reference is chosen as $|\Delta\theta_{c_{\max}}| < 30$ [deg], thus the actual pitch angle of the vehicle is controlled within a linear region of operation. Because we introduced a voluntary saturation limit to the output of the altitude controller, the integral control action of the altitude controller might be saturated whenever a large altitude error occurs. Subsequently, the accumulated integral error terms cannot be removed until when the altitude error gets small, resulting in a substantial overshoot.

The solution to this problem is the integrator antiwindup, which turns off the integral action as soon as the saturation occurs. Figure 96 illustrates a block diagram that implements the integrator antiwindup technique using a nonlinearity. As soon



(a) Entire display of the root locus plot of the altitude closed-loop system.



(b) Magnified display of the root locus plot showing the phugoid poles around the origin.

Figure 94: Root locus plot of the closed-loop system using a lead and PI compensators as k_h varies. The squares represent the closed-loop poles at $k_h = 1.22$.

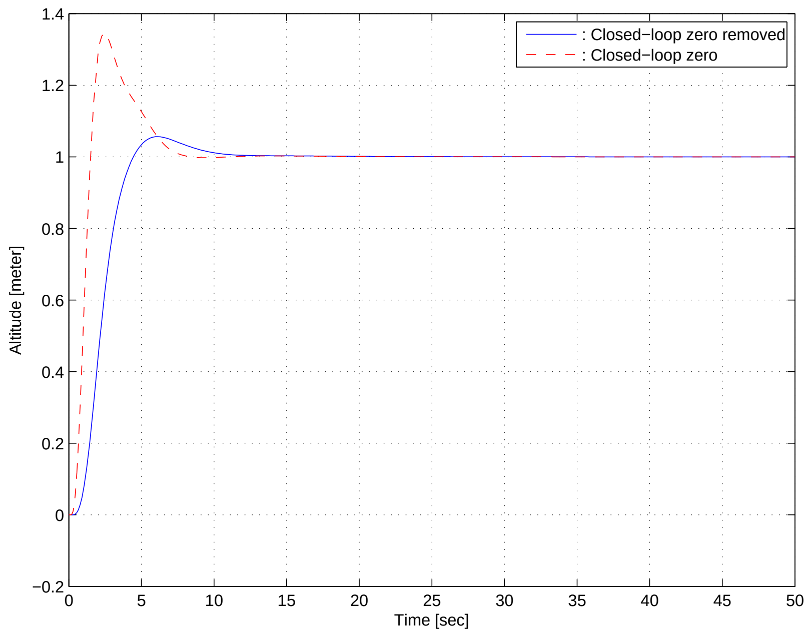


Figure 95: Step response of the closed-loop system by a unit altitude reference command.

as the saturation occurs, the feedback loop around the integrator reacts rapidly to keep e_1 at zero. The antiwindup gain K_a should be chosen to be large enough so that the antiwindup circuit is capable of following e while keeping the output from saturating.

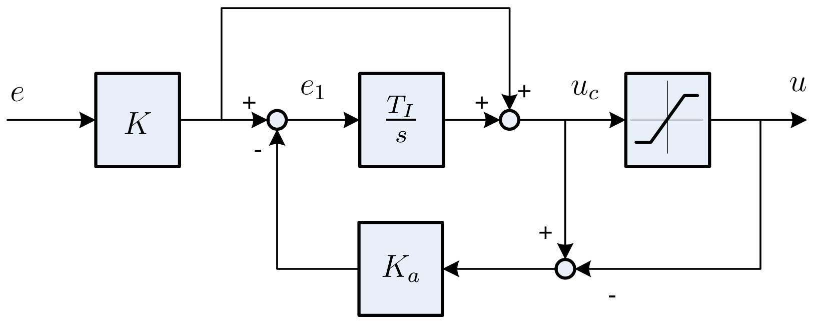


Figure 96: Integrator antiwindup technique with a single saturation nonlinearity.

Figure 97 shows the entire block diagram of the closed-loop altitude controller incorporating the integrator antiwindup scheme. Figure 98 shows the step response of the closed-loop system with/without the antiwindup scheme. It should be noted that the altitude controller implemented with the antiwindup element has substantially

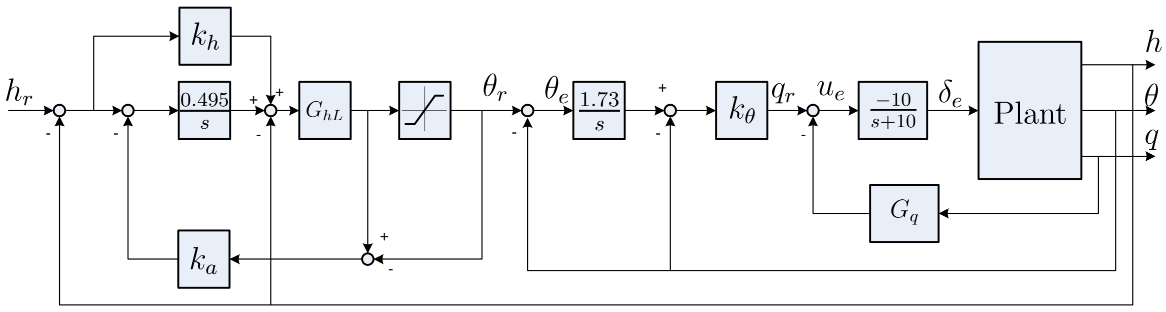


Figure 97: Block diagram of the final closed-loop altitude controller.

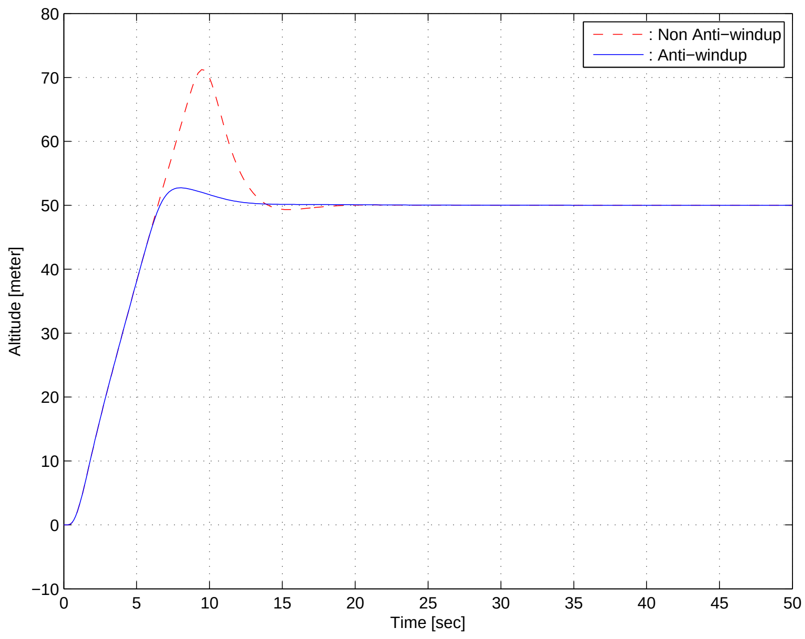


Figure 98: Step response of the closed-loop system with the altitude controller, comparing the cases of with/without antiwindup scheme.

less overshoot compared to the response of the original PI controller implementation.

D.2.4 Airspeed controller

An open-loop transfer function from the throttle command input (Δu_t) to the airspeed (ΔV_T) is obtained including the actuator dynamics using Eqs. (191) and (194c) as follows,

$$\frac{\Delta V_T}{\Delta u_t} = \frac{61.0413(s+0.07)(s+5.67 \pm 10.75i)}{(s+10)(s+0.09 \pm 0.64i)(s+5.72 \pm 10.71i)}. \tag{205}$$

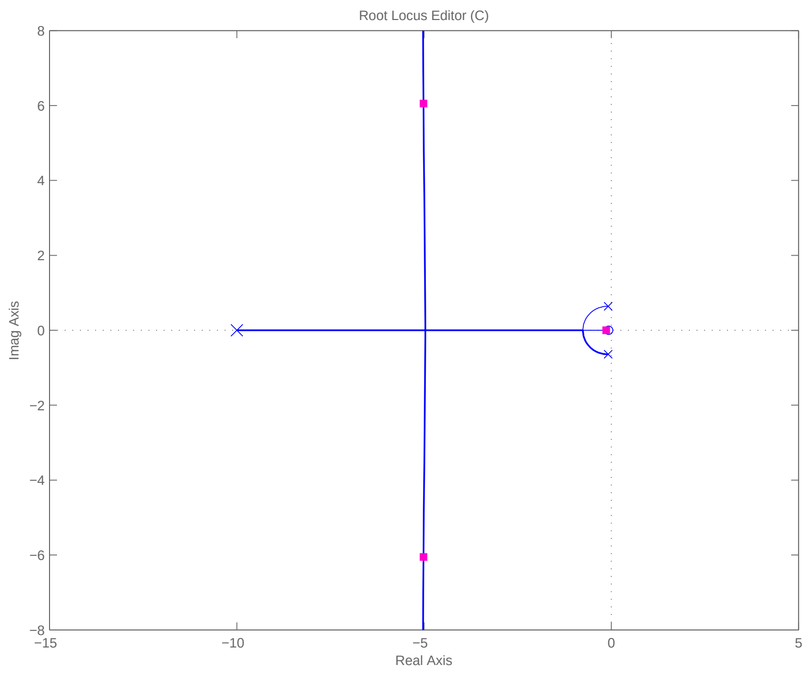


Figure 99: Root locus plot of the simplified speed control plant

Since the short period mode poles appears to be located close to the complex zeros, they could be cancelled out for the sake of simplicity. The simple transfer function is then given,

$$\frac{\Delta V_T}{\Delta u_t} = \frac{61.0413(s+0.07)}{(s+10)(s+0.09 \pm 0.64i)}. \tag{206}$$

As shown in the root locus plot (see Fig. 99), due to the zero close to the origin ($s = -0.07$), the DC gain of the plant is relatively small,

$$K_p = \lim_{s \to 0} \frac{\Delta V_T(s)}{\Delta u_t(s)} = 1.023, \tag{207}$$

which results in a large steady state error $1/(1+K_p) \approx 0.5$ when using the unity negative feedback scheme. In order to reduce the steady state error while improving the performance of the speed controller, a PI controller was adopted with PI zero at $s = -1.53$,

$$G_{V_T}(s) = k_{V_T} \frac{s+1.53}{s}. \tag{208}$$

Figure 100 shows the root locus plot of the closed-loop system with the airspeed controller. The PI zero at $s = -1.53$ in conjunction with the chosen gain $k_{V_T} = 0.773$

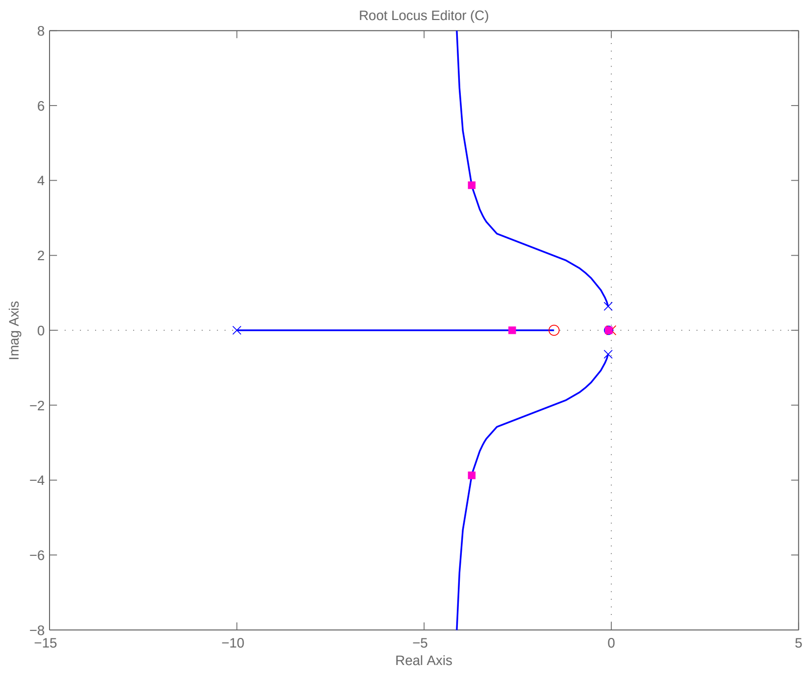


Figure 100: Root locus plot of the closed-loop system incorporating the PI speed controller. The closed-loop poles and zeros are shown, and the squares represent the closed-loop poles at $k_{V_T} = 0.773$

pushes the conjugate poles around the origin to $s = -1.92 \pm 2.13i$ to yield a damping ratio of 0.671. Figure 101 shows the step response of closed-loop system with the designed PI controller. The step response of the closed-loop system with the closed-loop zero removal PI implementation is also shown in Fig. 101, as the overshoot performance is improved at the cost of reduced rise time.

D.3 Lateral controllers design

In this section, we present a detail design of inner loop controllers for lateral-directional dynamic motion of the UAV. The roll and yaw dynamics are not, in general, decoupled from each other, the augmented control system design in the sequel should be incorporated with multivariable system analysis, for two control inputs of aileron and rudder, and two or more lateral outputs.

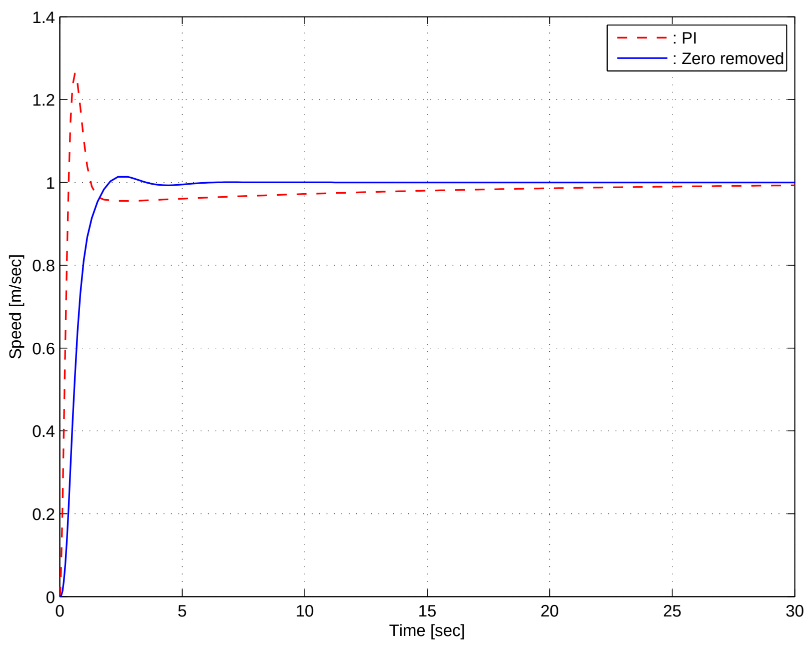


Figure 101: Step response of the closed-loop system by a unit speed reference command.

D.3.1 Yaw damper

The purpose of the yaw damper is to use the rudder to generate a yawing moment that opposes any yaw rate that builds up from the dutch roll mode. In a coordinated steady-state turn where the yaw rate is non-zero constant, the use of the yaw damper is found to be contradicting to the turn maneuver. A washout filter is used to compromise the tendency of turning and the yaw rate damping, by which the yaw rate signal due to the coordinated turn is differentiated and vanishes during steady-state turning condition. In contrast, the yaw rate signal due to the dutch-roll dynamics, which is assumed to be relatively high-frequency, is utilized in the yaw damper to suppress the dutch-roll mode. The wash out filter is given in the form of a first order high pass filter as follow,

$$G_{\mathrm{w}}(s) = \frac{\Delta r_{\mathrm{w}}}{\Delta r} = \frac{\tau_{\mathrm{w}} s}{1 + \tau_{\mathrm{w}} s}, \tag{209}$$

where the time constant τ_{w} is determined by $\tau w = 1.0$ by taking into account the natural frequency of the inherent dutch-roll mode of the UAV.

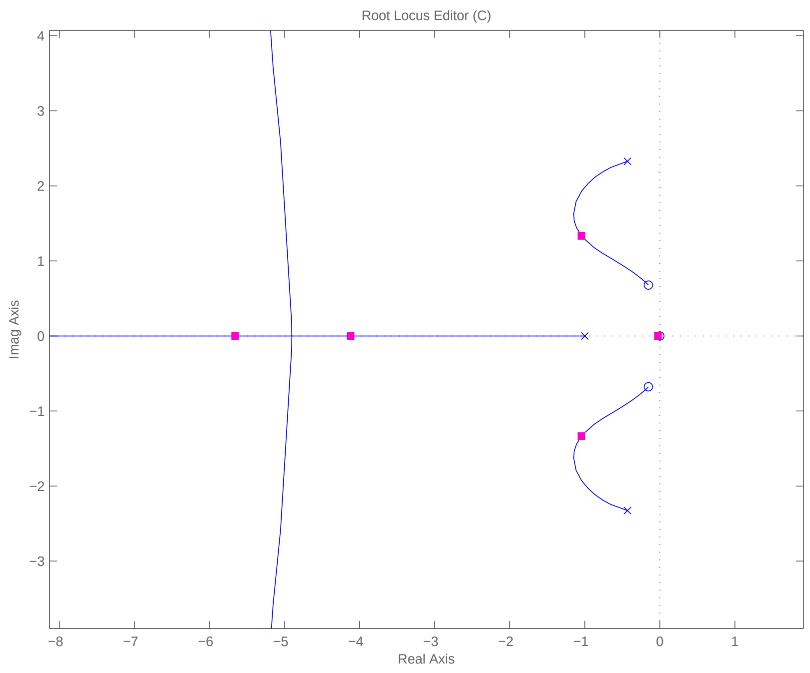


Figure 102: Root locus plot of the closed-loop system with the designed yaw washout damper. The squares represent the closed-loop poles at $k_r = 0.27$.

Figure 102 shows the root locus plot of the closed-loop system with the yaw washout damper in the feedback path. The washout gain is chosen $k_r = 0.27$, hence the dominant dutch-roll poles end up with $s = -1.0 \pm 1.55i$ with the damping ratio of 0.543. Figure 103 shows the performance of the yaw damper in conjunction with the washout filter. The aileron doublet command is used to excite the dutch-roll motion, and the yaw damper suppresses the residual roll rate effectively as well as the residual yaw rate, from the roll and yaw dutch roll coupling.

D.3.2 Roll controllers

The roll controllers consist of the roll rate damper and the roll angle controller. The roll-damping loop is less critical and is closed first. An open loop transfer function from the aileron command input (Δu_a) to the roll rate output (Δp) is obtained from using Eqs. (193) and (194a) as follows,

$$\frac{\Delta p}{\Delta u_a} = \frac{1822.4(s-0.027)(s+0.449\pm2.131i)}{(s+0.064)(s+0.364\pm2.465i)(s+10)(s+16.926)}, \tag{210}$$

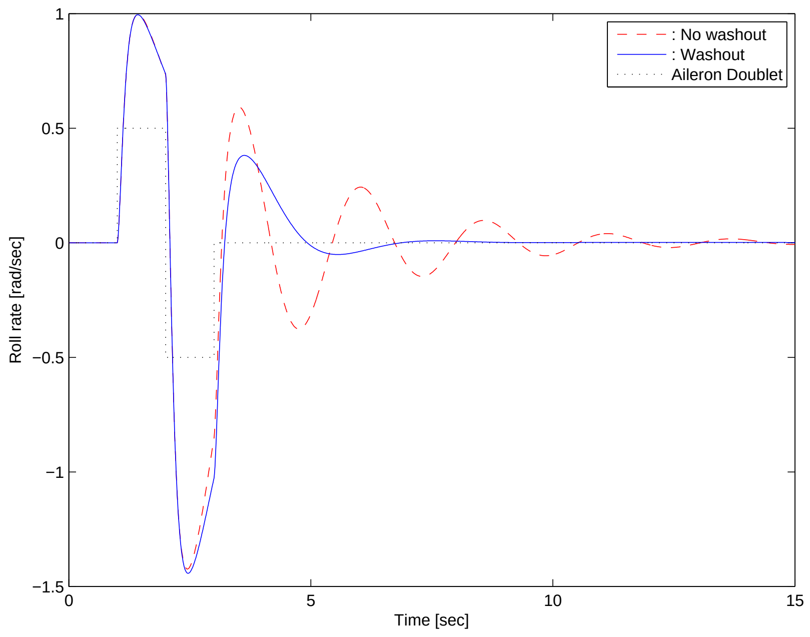


Figure 103: The effectiveness of the yaw damper to suppress the dutch roll mode oscillation.

which reveals that the plant has a non-minimum phase zero, thus excessive gain will lead to a unstable closed-loop pole. Figure 104 shows the root locus plot of the unity feedback with positive roll damping gain k_p, which indicates that a large gain will push the spiral pole to the right half plane to end up with a unstable pole. Consequently, the gain of the roll damper is chosen $k_p = 0.075$ to make the closed-loop system stable.

The closed-loop transfer function from the commanded roll rate (Δp_r) to the roll rate (Δp) after closing the roll damper is calculated as follows,

$$\frac{\Delta p}{\Delta p_r} = \frac{1822.4(s - 0.027)(s + 0.93 \pm 0.97i)(s + 5.02 \pm 1.13i)}{(s + 0.048)(s + 1.0 \pm 1.55i)(s + 4.77 \pm 1.05i)(s + 10)(s + 17.12)}. \tag{211}$$

With the designed roll damper, we design a roll angle controller that forms an outer loop of the system given in Eq. (211). In general, the roll angle is not measured directly from the inertial sensors but estimated from the attitude Kalman filter as developed in Appendix B. Since the filter introduces the time latency, the delay effect of the filter should be taken into consideration in designing the roll angle controller.

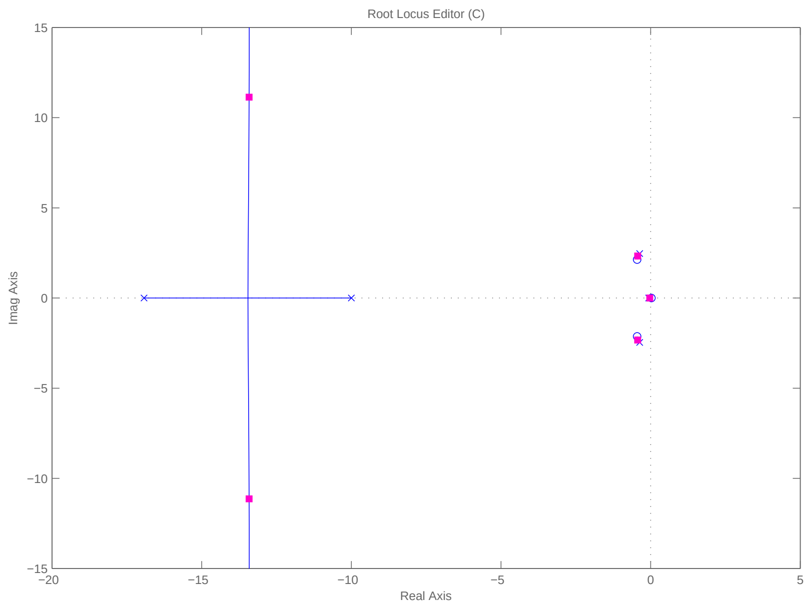


Figure 104: Root locus plot of the closed-loop system with the roll damper. The squares represent the closed-loop poles at $k_p = 0.075$.

Subsequently, the filter is modelled by a first-order low-pass filter from the actual bank angle ($\Delta\phi$) to the estimated bank angle ($\Delta\widehat{\phi}$) as follows,

$$G_f(s) = \frac{\Delta\widehat{\phi}(s)}{\Delta\phi(s)} = \frac{1}{\tau_f s + 1}, \tag{212}$$

where the time constant τ_f is chosen as $\tau_f = 0.5$ which corresponds the 0.5 [sec] time latency identified a priori.

After closing the roll damping loop, the open loop plant from the roll rate command input (Δp_r) to the estimated roll angle ($\Delta\widehat{\phi}$) is obtained as follows,

$$\frac{\Delta\widehat{\phi}}{\Delta p_r} = \frac{3655.6(s + 0.91 \pm 0.98i)(s + 5.02 \pm 1.06i)}{(s + 0.048)(s + 1.00 \pm 1.55i)(s + 2)(s + 4.77 \pm 1.05i)(s + 10)(s + 17.12)}. \tag{213}$$

Because the plant has no integrator (type 0), a PI controller is designed to remove the steady state error as follows,

$$G_\phi(s) = k_\phi \frac{s + 0.05}{s}. \tag{214}$$

Figure 105 shows the root locus plot of the closed-loop system with the roll angle PI controller. Because the closed-loop system has two dominant modes around the origin, the gain k_ϕ is carefully chosen $k_\phi = 0.128$, after evaluating the time response of the closed-loop system. Figure 106 shows the step response of the roll angle closed-loop system. In addition, the Bode plot of the closed-loop system is displayed in Fig. 107, which shows that the closed-loop system has the gain margin of 10.6 [dB] and the phase margin of 63.3 [deg].

Figure 108 shows the entire block diagram of the closed-loop system for lateral controllers.

D.4 Discrete implementation

For the discrete implementation, the control sampling rate is first determined by 20 [Hz] taking into account the computational throughput of the RCM-3400 micro-controller. Note that the dynamics of the UAV is much slower than 20 [Hz], we assume that at this sampling rate the most dynamic characteristics are captured by the on-board sensors. The discrete implementation of the controller is computed by incorporating the bilinear mapping (Tustin's method) as follows,

$$s = \frac{2}{T}\frac{z-1}{z+1}, \tag{215}$$

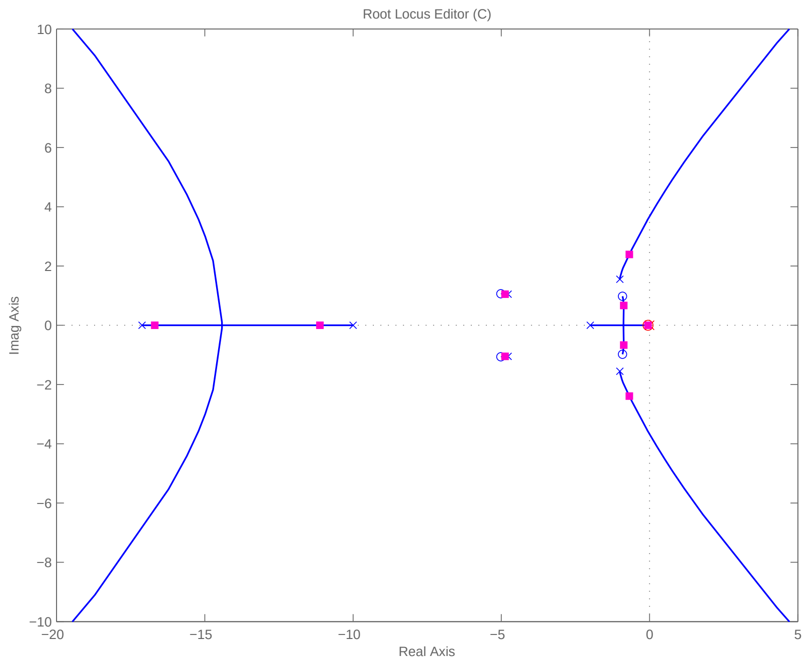
where $T = 0.05$ is used for the 20Hz sampling frequency.

The discrete version of the controller, as an example by the roll angle PI controller, can be obtained as follows,

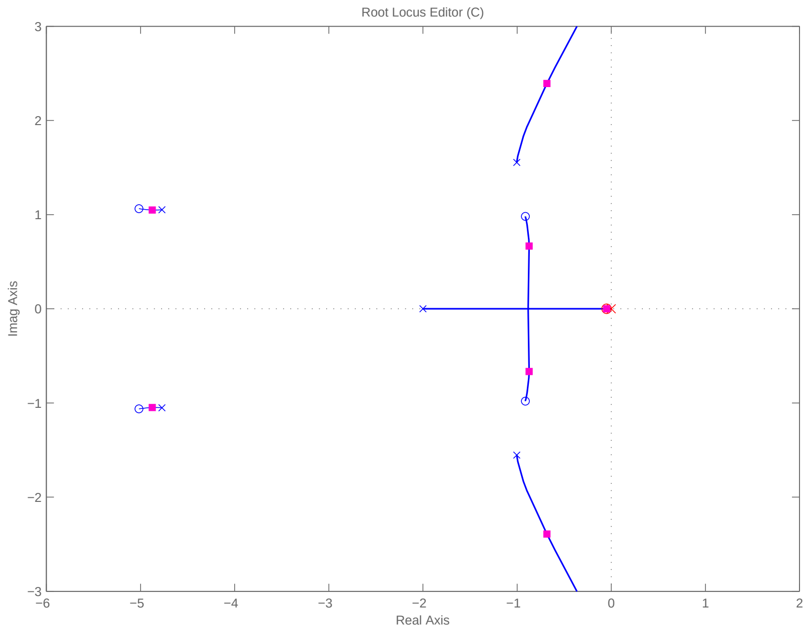
$$\begin{aligned}\frac{\Delta p_r}{\Delta \phi_e} &= k_\phi \frac{s + T_I}{s}\bigg|_{s=40\frac{z-1}{z+1}} \\ &= \frac{40k_\phi(z-1) + k_\phi T_I(z+1)}{40(z-1)} \\ &= \frac{(k_\phi + 0.025k_\phi T_I) + (-k_\phi + 0.025k_\phi T_I)z^{-1}}{1 - z^{-1}}.\end{aligned} \tag{216}$$

Consequently, the control at time step k is obtained from the inverse z-transform,

$$\Delta p_r[k] = \Delta p_r[k-1] + (k_\phi + 0.025k_\phi T_I)\Delta\phi_e[k] + (-k_\phi + 0.025k_\phi.T_I)\Delta\phi_e[k-1] \tag{217}$$



(a) Entire display of the root locus plot of the roll angle closed-loop system.



(b) Magnified display of the root locus plot showing two dominant modes around the origin.

Figure 105: Root locus plot of the closed-loop system with the PI roll angle controller as k_ϕ varies. The squares represent the closed-loop poles at $k_\phi = 0.128$.

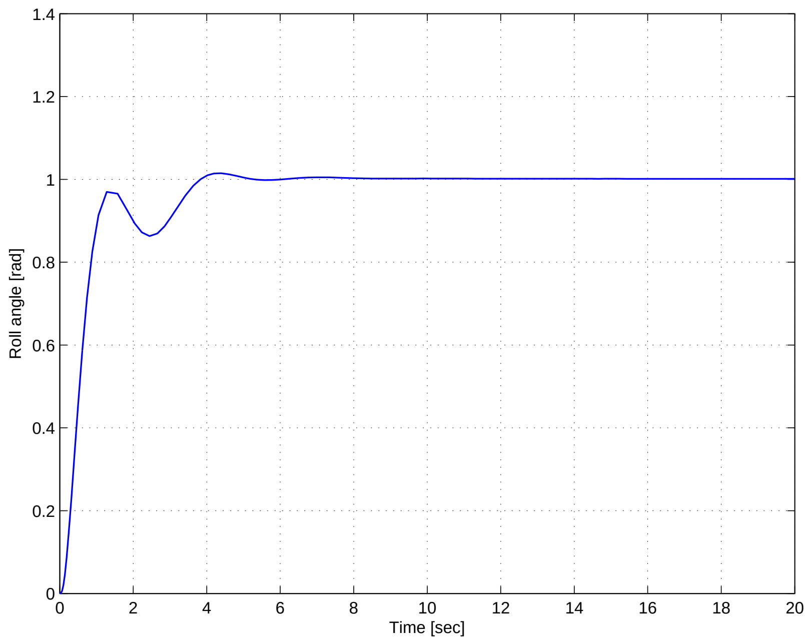


Figure 106: Step response of the closed-loop system by a unit roll angle reference command.

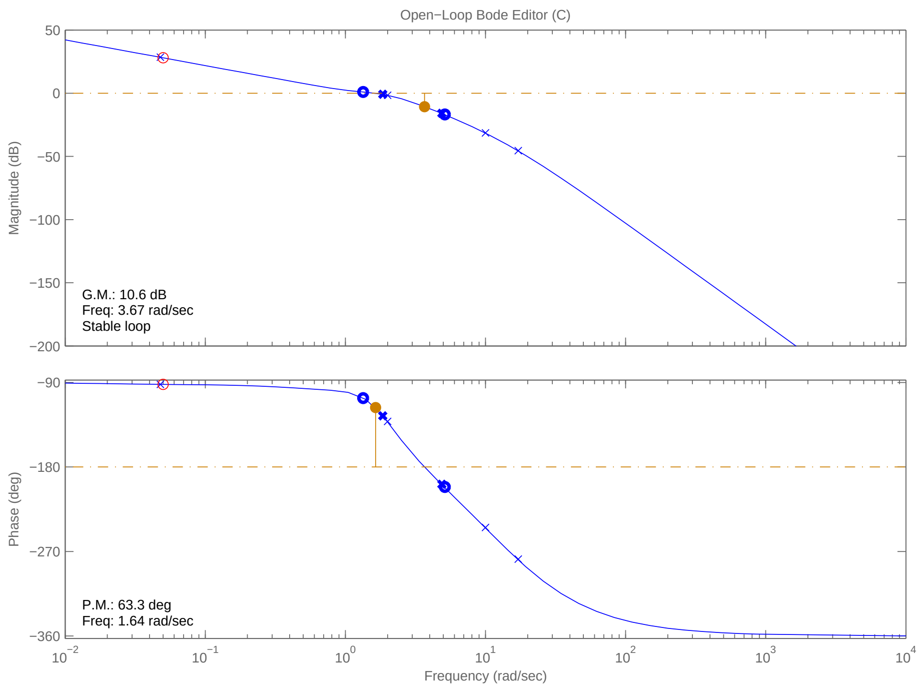


Figure 107: Bode plot of the closed-loop system with the roll angle PI controller.

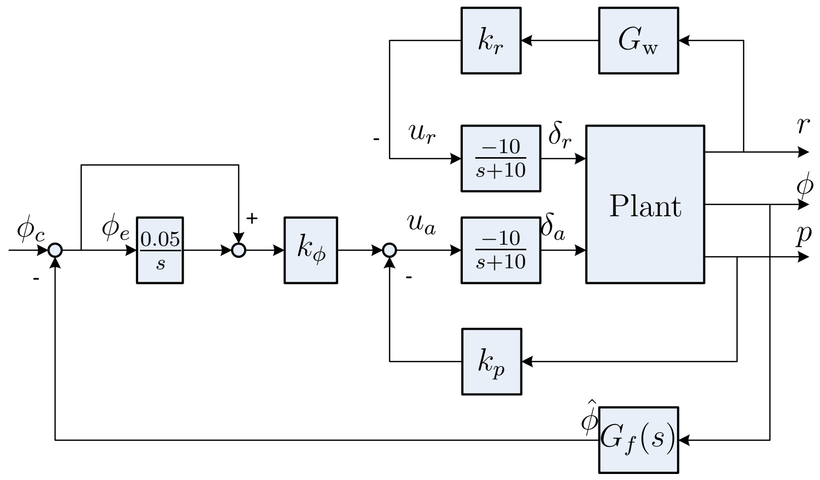


Figure 108: Entire block diagram for the lateral controllers

REFERENCES

[1] "FlightGear Flight Simulator," Oct. 2007. http://www.flightgear.org/.

[2] Adkins, C. N. and Liebeck, R. H., "Design of Optimum Propellers," *Journal of Propulsion and Power*, vol. 10, no. 5, pp. 676–682, 1994.

[3] Aguiar, A. P. and Pascoal, A. M., "Way-point Tracking of Underactuated AUVs in the Presence of Ocean Currents," in *Proceedings of the 10th Mediterranean Conference on Control and Automation*, (Lisbon, Portugal), July 2002.

[4] Allgöwer, F. and Zheng, A., *Nonlinear Model Predictive Control*, vol. 26 of *Progress in Systems and Control Theory.* Basel, Germany: Birkhäuser Verlag, 2000.

[5] Anderson, E. P. and Beard, R. W., "An Algorithmic Implementation of Constrained Extremal Control for UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Monterey, Canada), Aug. 2002. AIAA 2002-4470.

[6] Anderson, E. P., Beard, R. W., and McLain, T. W., "Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles," *IEEE Transactions on Control Systems Technology*, vol. 13, pp. 471–477, May 2005.

[7] Arinaga, S., Nakajima, S., Okabe, H., Ono, A., and Kanayama, Y., "A Motion Planning Method for an AUV," in *Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology*, (Monterey, CA), pp. 477–484, June 1996.

[8] Bak, M., Larsen, T., Norgaard, M., Andersen, N., Poulsen, N., and Ravn, O., "Location Estimation using Delayed Measurements," in *Proceedings of the 5th International Workshop on Advanced Motion Control (AMC98)*, pp. 180–185, June-July 1998.

[9] Barraquand, J., Langlois, B., and Latombe, J.-C., "Numerical Potential Field Techniques for Robot Path Planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 224–240, Mar.-Apr. 1992.

[10] Beard, R. W., McLain, T. W., Goodrich, M., and Anderson, E. P., "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 911–922, Dec. 2002.

[11] Behnke, S., “Local Multiresolution Path Planning,” in *RoboCup 2003: Robot Soccer World Cup VII*, vol. 3020 of *Lecture Notes in Computer Science*, pp. 332–343, Berlin: Springer, 2004.

[12] Bellingham, J., Kuwata, Y., and How, J., “Stable Receding Horizon Trajectory Control for Complex Environments,” in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Austin, TX), Aug. 2003. AIAA 2003-5635.

[13] Bellingham, J., Richards, A., and How, J. P., “Receding Horizon Control of Autonomous Aerial Vehicles,” in *Proceedings of the American Control Conference*, (Anchorage, AK), pp. 3741–3746, May 2002.

[14] Bemporad, A., Luca, A. D., and Oriolo, G., “Local Incremental Planning for a Car-Like Robot Navigating among Obstacles,” in *Proceedings of IEEE International Conference on Robotics and Automation*, (Minneapolis, MN), pp. 1205–1211, Apr. 1996.

[15] Berglund, T., Jonsson, H., and Söderkvist, I., “An Obstacle-avoiding Minimum Variation B-spline Problem,” in *Proceedings of International Conference on Geometric Modeling and Graphics*, pp. 156–161, July 2003.

[16] Blake, W. B., “Prediction of Fighter Aircraft Dynamic Derivatives using Digital Datcom,” in *Third Applied Aerodynamics Conference*, (Colorado Springs, CO), Oct. 1985. AIAA-1985-4070.

[17] Boissonnat, J. D., Cérézo, A., and Leblond, J., “Shortest Paths of Bounded Curvature in the Plane,” in *Proceedings of 9^{th} IEEE International Conference on Robotics and Automation*, 1992.

[18] Bortoff, S. A., “Path planning for UAVs,” in *Proceedings of the American Control Conference*, (Chicago, IL), pp. 364–368, 2000.

[19] Brooks, R. A., “Solving the Find-Path Problem by Good Representation of Free Space,” *IEEE Transactions on System, Man, and Cybernetics*, vol. 13, no. 3, pp. 190–193, 1983.

[20] Brown, R. G. and Hwang, P. Y. C., *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises and Solutions.* New York, NY: John Wiley & Sons, 3rd ed., 1997.

[21] Bui, X.-N. and Souères, P., “Shortest Path Synthesis for Dubins Nonholonomic Robot,” in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, (San Diego, CA), pp. 2–7, May 1994.

[22] Burrus, C. S., Gopinath, R. A., and Guo, H., *Introduction to Wavelets and Wavelet Transforms.* Upper Saddle River, New Jersey: Prentice Hall, 1998.

[23] Calderbank, A. R., Daubechies, I., Sweldens, W., and Yeo, B.-L., "Wavelet Transforms That Map Integers to Integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, 1998.

[24] Camacho, E. F. and Bordons, C., *Model Predictive Control.* London, UK: Springer-Verlag, 1999.

[25] Carmo, M. D., *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.

[26] Caruso, M. J., "Application of Magnetic Sensors for Low Cost Compass Systems." Honeywell technical article. http://www.ssec.honeywell.com/magnetic/.

[27] Caruso, M. J., "Applications of Magnetoresistive Sensors in Navigation Systems." Honeywell technical article. http://www.ssec.honeywell.com/magnetic/.

[28] Chandler, P. and Pachter, M., "Research Issues in Autonomous Control of Tactical UAVs," in *Proceedings of the American Control Conference*, (Philadelphia, PA), pp. 394–398, 1998.

[29] Chandler, P. R., Rasmussen, S., and Pachter, M., "UAV Cooperative Path Planning," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Denver, CO), Aug. 2000. AIAA-2000-4370.

[30] Chen, D. Z., Szczerba, R. J., and Uhran, J. J., "A Framed-Quadtree Approach for Determining Euclidean Shortest Paths in a 2-D Environment," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 668–681, 1997.

[31] Colgren, R. D. and Martin, K. E., "Flight Test Validation of Sideslip Estimation using Inertial Accelerations," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Denver, CO), Aug. 2000. AIAA-2000-4448.

[32] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms.* The MIT Press, 2nd ed., Sept. 2001.

[33] Daubechies, I. and Sweldens, W., "Factoring Wavelet Transform into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[34] de Boor, C., *A Practical Guide to Splines*, vol. 27 of *Applied mathematical sciences.* New York, NY: Springer-Verlag, 1978.

[35] Dijkstra, E., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[36] Dimock, G. A., Deters, R. W., and Selig, M. S., "Icing Scenarios with the Icing Encounter Flight Simulator," in *the AIAA 41st Aerospace Sciences Meeting and Exhibit*, (Reno, NV), Jan. 2003. AIAA-2003-23.

[37] Donoho, D. L., "Smooth Wavelet Decompositions with Blocky Coefficient Kernels," in *Recent Advances in Wavelet Analysis*, pp. 1–43, Academic Press, 1993.

[38] Dubins, L. E., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[39] Dyllong, E. and Visioli, A., "Planning and Real-time Modifications of a Trajectory Using Spline Techniques," *Robotica*, vol. 21, pp. 475–482, 2003.

[40] Encarnacao, P. and Pascoal, A., "Combined Trajectory Tracking and Path Following: An Application to the Coordinated Control of Autonomous Marine Craft," in *Proceedings of the 40th IEEE Conference on Decision and Control*, (Orlando, FL), pp. 964–969, Dec. 2001.

[41] Etkin, B. and Reid, L. D., *Dynamics of Flight: Stability and Control*. New York, NY: John Wiley and Sons, 3rd ed., 1996.

[42] Finch, S. R., *Mathematical Constants*, ch. 5, pp. 331–339. Cambridge, England: Cambridge University Press, 2003.

[43] Fuller, J., Seto, D., and Meisner, R., "Optimization-Based Control for Flight Vehicles," in *AIAA Guidance, Navigation, and Control Conference*, (Denver, CO), Aug. 2000. AIAA 2000-4055.

[44] Galbraith, B., "DATCOM Predicted Aerodynamic Model," July 2004. http://www.holycows.net.

[45] Garcia, C. E., Prett, D. M., and Morari, M., "Model Predictive Control: Theory and Practice – A Survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[46] Geiger, B. R., Horn, J. F., DeLullo, A. M., Long, L. N., and Niessner, A. F., "Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, CO), Aug. 2006. AIAA 2006-6199.

[47] Godbole, D., Samad, T., and Gopal, V., "Active Multi-Model Control for Dynamic Maneuver Optimization of Unmanned Air Vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1257–1262, 2000.

[48] Greville, T. N. E., *Theory and Applications of Spline Functions*. New York, NY: Academinc press, 1968.

[49] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[50] Heer, V. K. and Reinfelder, H.-E., "A Comparison of Reversible Methods for Data Compression," in *Medical Imaging IV: Image Processing*, vol. Proc. SPIE Vol. 1233, pp. 354–365, 1990.

[51] Heller, M., Myschik, S., Holzapfel, F., and Sachs, G., "Low-cost Approach based on Navigation Data for Determining Angles of Attack and Sideslip for Small Aircraft," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Austin, TX), Aug. 2003. AIAA-2003-5777.

[52] Hwang, J. Y., Kim, J. S., Lim, S. S., and Park, K. H., "A Fast Path Planning by Path Graph Optimization," *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, vol. 33, no. 1, pp. 121–128, 2003.

[53] Inman, D. J., *Engineering Vibration*. Englewood Cliffs, New Jersey: Prentice Hall, 1996.

[54] Jackins, C. L. and Tanimoto, S. L., "Oct-tree and Their Use in Representing Three Dimensional Objects," *Computer Graphics and Information Processing*, vol. 14, no. 3, pp. 249–270, 1980.

[55] Jia, D. and Vagners, J., "Parallel Evolutionary Algorithms for UAV Path Planning," in *AIAA 1st Intelligent Systems Technical Conference*, (Chicago, IL), Sept. 2004. AIAA 2004-6230.

[56] Judd, K. B. and McLain, T. W., "Spline Based Path Planning for Unmanned Air Vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Montreal, Canada), Aug. 2001. AIAA 2001-4238.

[57] Jung, D. and Tsiotras, P., "A 3-DoF Experimental Test-Bed for Integrated Attitude Dynamics and Control Research," in *AIAA Guidance, Navigation, and Control Conference*, (Austin, Texas), 2003. AIAA 2003-5331.

[58] Jung, D. and Tsiotras, P., "Modelling and Hardware-in-the-loop Simulation for a Small Unmanned Aerial Vehicle," in *AIAA Infotech at Aerospace*, (Rohnert Park, CA), May 2007. AIAA Paper 07-2763.

[59] Jung, D. and Tsiotras, P., "Multiresolution On-Line Path Planning for Small Unmanned Aerial Vehicles," in *American Control Conference*, 2008. submitted.

[60] Kambhampati, S. and Davis, L. S., "Multiresolution Path Planning for Mobile Robots," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 135–145, Sept. 1986.

[61] Kanayama, Y. and Hartman, B. I., "Smooth Local Path Planning for Autonomous Vehicles," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1265–1270, May 1989.

[62] Kang, Y. and Hedrick, J. K., "Design of Nonlinear Model Predictive Controller for a Small Fixed-Wing Unmanned Aerial Vehicle," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, CO), Aug. 2006. AIAA-2006-6685.

[63] Karata, T. and Bullo, F., "Randomized Searches and Nonlinear Programming in Trajectory Planning," in *Proceedings of the 40th IEEE Conference on Decision and Control*, (Orlando, FL), pp. 5032–5037, Dec. 1994.

[64] Kavraki, L. and Latombe, J.-C., "Randomized Preprocessing of Configuration for Fast Path Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, (San Diego, CA), pp. 2138–2145, May 1994.

[65] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 396–404, 1986.

[66] Kim, B. and Tsiotras, P., "Time-Invariant Stabilization of a Unicycle-Type Mobile Robot: Theory and Experiments," in *IEEE Conference on Control Applications*, (Ancorage, AL), pp. 443–448, Sept. 2000.

[67] Kim, H. J., Shim, D. H., and Sastry, S., "Nonlinear Model Predictive Tracking Control for Rotorcraft-based Unmanned Aerial Vehicles," in *Proceedings of the American Control Conference*, (Anchorage, AK), pp. 3576–3581, May 2002.

[68] Kim, J., Pearce, R. A., and Amato, N. M., "Extracting Optimal Paths from Roadmaps for Motion Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 2424–2429, Sept. 2003.

[69] Klein, R., *Concrete and Abstract Voronoi Diagrams*, vol. 400 of *Lecture notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1989.

[70] Koenig, S. and Likhachev, M., "Incremental $\mathcal{A}^*$," in *Advances in Neural Information Processing Systems*, pp. 1539–1546, 2002.

[71] Koenig, S. and Likhachev, M., "$\mathcal{D}^*$ Lite," in *Proceedings of the National Conference of Artificial Intelligence*, pp. 476–483, 2002.

[72] Korf, R. E., "Real-Time Heuristic Search," *Artificial Intelligence*, vol. 42, pp. 189–211, 1990.

[73] Kuwata, Y. and How, J. P., "Stable Trajectory Design for Highly Constrained Environments using Receding Horizon Control," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 902–907, June-July 2004.

[74] Lapierre, L., Soetanto, D., and Pascoal, A., "Nonlinear Path Following with Applications to the Control of Autonomous Underwater Vehicles," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, (Maui, HI), pp. 1256–1261, Dec. 2003.

[75] Lapierre, L. and Soetanto, D., "Nonlinear Path-following Control of an AUV," *Ocean Engineering*, vol. 34, pp. 1734–1744, 2007.

[76] Lapierre, L., Zapata, R., and Lepinay, P., "Combined Path-following and Obstacle Avoidance Control of a Wheeled Robot," *The International Journal of Robotics Research*, vol. 26, pp. 361–375, Apr. 2007.

[77] Lapp, T. and Singh, L., "Model Predictive Control Based Trajectory Optimization for Nap-of-the-Earth (NOE) Flight Including Obstacle Avoidance," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 891–896, June-July 2004.

[78] Larrabee, E. E., "Practical Design of Minimum Induced Loss Propellers," in *Society of Automotive Engineers, Business Aircraft Meeting and Exposition*, (Wichita, KS), Apr. 1979.

[79] Larsen, T., Andersen, N., Ravn, O., and Poulsen, N., "Incorporation of Time Delayed Measurements in a Discrete-time Kalman Filter," in *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 4, pp. 3972–3977, Dec. 1998.

[80] Larson, R. A., Pachter, M., and Mears, M. J., "Path Planning by Unmanned Air Vehicles for Engaging an Integrated Radar Network," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (San Francisco, CA), Aug. 2005. AIAA 2005-6191.

[81] Latombe, J.-C., *Robot Motion Planning.* Boston, MA: Kluwer Academic Publishers, 1991.

[82] Leonardo Daga, "RS232 Blockset for Simulink," Dec. 2004. http://digilander.libero.it/LeoDaga/Simulink/RS232Blockset.htm.

[83] Leonardo Daga, "RT Blockset for Simulnk," Dec. 2004. http://digilander.libero.it/LeoDaga/Simulink/RTBlockset.htm.

[84] Lozano-Perez, T. and Wesley, M. A., "An Algorithm for Planning Collision-Free Path Among Polyhedral Obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[85] Luca, A. D. and Oriolo, G., "Local Incremental Planning for Nonholonomic Mobile Robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, (San Diego, CA), pp. 104–110, May 1994.

[86] Lumelsky, V. and Stepanov, A., "Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment," *IEEE Transactions on Automatic Control*, vol. 31, pp. 1058–1063, Nov. 1986.

[87] Lutterkort, D. and Peters, J., "Smooth Paths in a Polygonal Channel," in *Proceedings of the fifteenth Annual Symposium on Computational Geometry*, (Miami Beach, FL), pp. 316–321, 1999.

[88] Lutterkort, D. and Peters, J., "Tight Linear Envelopes for Splines ," *Numerische Mathematik*, vol. 89, pp. 735–748, Oct. 2001.

[89] Madras, N. and Slade, G., *The Self-Avoiding Walk*. Boston, MA: Birkhäuser, 1993.

[90] Mallat, S. G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 674–693, July 1989.

[91] Marti, K. and Qu, S., "Path Planning for Robots by Stochastic Optimization Methods," *International Journal of Intelligent and Robotic Systems*, vol. 22, pp. 117–127, June 1998.

[92] Martin Hepperle, "JavaProp - Design and Analysis of Propellers." http://www.mh-aerotools.de/airfoils/javaprop.htm.

[93] McLain, T., Chandler, P., and Pachter, M., "A Decomposition Strategy for Optimal Coordination of Unmanned Air Vehicles," in *Proceedings of the American Control Conference*, (Chicago, IL), pp. 369–373, 2000.

[94] McLain, T. W. and Beard, R. W., "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 1, pp. 150–161, 2005.

[95] Mertz, P. and Gray, F., "A Theory of Scanning and Its Relation to the Characteristics of the Transmitted Signal in Telephotography and Television," *Bell System Technical Journal*, vol. 13, pp. 464–515, 1934.

[96] Micaelli, A. and Samson, C., "Trajectory Tracking for Unicycle-type and Two-Steering-Wheels Mobile Robots," Tech. Rep. 2097, INRIA, Sophia-Antipolis, Nov. 1993.

[97] Morari, M. and Lee, J., "Model Predictive Control: Past, Present, and Future," *Computers and Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[98] Motorola, Inc., *Motorola GPS Products - Oncore User's Guide*, Aug. 2002. Revision 5.0.

[99] Nairn, D., Peters, J., and Lutterkort, D., "Sharp, Quantitative Bounds on the Distance Between a Polynomial Piece and its Bézier Control Polygon," *Computer Aided Geometric Design*, vol. 16, pp. 613–631, 1999.

[100] Nelson, D. R., Barber, D. B., McLain, T. W., and Beard, R. W., "Vector Field Path Following for Small Unmanned Air Vehicles," in *Proceedings of the 2006 American Control Conference*, (Minneapolis, MN), pp. 5788–5794, June 2006.

[101] Neus, M. and Maouche, S., "Motion Planning using the Modified Visibility Graph," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, (Tokyo, Japan), pp. 651–655, Oct. 1999.

[102] Niculescu, M., "Lateral Track Control Law for Aerosonde UAV," in *39th AIAA Aerospace Sciences Meeting and Exhibit*, (Reno, NV), Jan. 2001. A01-16013.

[103] Noborio, H., Naniwa, T., and Arimoto, S., "A Quadtree-Based Path-Planning Algorithm for a Mobile Robot," *Journal of Robotic Systems*, vol. 7, no. 4, pp. 555–574, 1990.

[104] Pai, D. K. and Reissell, L.-M., "Multiresolution Rough Terrain Motion Planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 19–33, 1998.

[105] Park, S., *Avionics and Control System Development for Mid-Air Rendezvous of Two Unmanned Aerial Vehicles.* Ph.d. thesis, Massachusetts Institute of Technology, Boston, MA, Feb. 2004.

[106] Park, S., Deyst, J., and How, J. P., "A New Nonlinear Guidance Logic for Trajectory Tracking," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Providence, RI), Aug. 2004. AIAA-2004-4900.

[107] Perry, A. R., "The FlightGear Flight Simulator," in *USENIX Annual Technical Conference*, UseLinux SIG sessions, (Boston, MA), June-July 2004.

[108] Pettersen, K. Y. and Lefeber, E., "Way-point Tracking Control of Ships," in *Proceedings of the 40th IEEE Conference on Decision and Control*, (Orlando, FL), pp. 940–945, Dec. 2001.

[109] Piegl, L. and Tiller, W., *The NURBS Book.* Monographs in Visual Communication, Berlin Heidelberg: Springer-Verlag, 2 ed., 1997.

[110] Pirzadeh, A. and Snyder, W., "A Unified Solution to Coverage and Search in Explored and Unexplored Terrains using Indirect Control," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2113–2119, May 1990.

[111] Podscedkowski, L., "Path Planner for Nonholonomic Mobile Robot with Fast Replanning Procedure," in *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, (Leuven, Belgium), pp. 3588–3593, May 1998.

[112] Podsedkowski, L., Nowakowski, J., Idzikowski, M., and Visvary, I., "Modified $\mathcal{A}^*$ Algorithm Suitable for Online Car-like Mobile Robot Control," in *Proceedings of the First Workshop on Robot Motion and Control*, pp. 235–240, June 1999.

[113] Prasanth, R., Bošković, J., Li, S.-M., and Mehra, R., "Initial Study of Autonomous Trajectory Generation for Unmanned Aerial Vehicles," in *Proceedings of the 40^{th} IEEE Conference on Decision and Control*, (Orlando, FL), pp. 640–645, 2001.

[114] Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T., "Dynamic Optimization Strategies for Three-Dimensional Conflict Resolution of Multiple Aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 4, pp. 586–594, 2004.

[115] Rathbun, D., Kragelund, S., Pongpunwattana, A., and Capozzi, B., "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments," in *Proceedings of the 21^{st} Digital Avionics Systems Conference*, vol. 2, pp. 8D2(1)–8D2(12), 2002.

[116] Ren, W. and Beard, R. W., "Trajectory Tracking for Unmanned Air Vehicles with Velocity and Heading Rate Constraints," *IEEE Transactions on Control Systems Technology*, vol. 12, pp. 706–716, Sept. 2004.

[117] Richards, A. and How, J., "Aircraft Trajectory Planning with Collision Avoidance using Mixed Integer Linear Programming," in *Proceedings of the American Control Conference*, (Anchorage, AK), pp. 1936–1940, 2002.

[118] Richards, A., How, J., Schouwenaars, T., and Feron, E., "Plume Avoidance Maneuver Planning Using Mixed Integer Linear Programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Montreal, Canada), Aug. 2001. AIAA-2001-4091.

[119] Richards, A., Kuwata, Y., and How, J., "Experimental Demonstrations of Real-time MILP Control," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Austin, TX), Aug. 2003. AIAA 2003-5802.

[120] Rysdyk, R., "UAV Path Following for Constant Line-Of-Sight," in *2nd AIAA Unmanned Unlimited Conference and Workshop and Exhibit*, (San Diego, CA), Sept. 2003. AIAA-2003-6626.

[121] Samad, T., Gorinevsky, D., and Stoffelen, F., "Dynamic Multiresolution Route Optimization for Autonomous Aircraft," in *Proceedings of the IEEE*

International Symposium on Intelligent Control, (Mexico City, Mexico), pp. 13–18, Sept. 2001.

[122] Scheuer, A. and Laugier, C., "Planning Sub-Optimal and Continuous-Curvature Paths for Car-Like Robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Victoria, B. C., Canada), pp. 25–31, Oct. 1998.

[123] Schouwenaars, T., How, J., and Feron, E., "Receding Horizon Path Planning with Implicit Safety Guarantees," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 5576–5581, June-July 2004.

[124] Schouwenaars, T., Moor, B. D., Feron, E., and How, J., "Mixed Integer Programming for Multi-Vehicle Path Planning," in *Proceedings of the European Control Conference*, (Porto, Portugal), pp. 2603–2608, 2001.

[125] Schouwenaars, T., Valenti, M., Feron, E., and How, J., "Implementation and Flight Test Results of MILP-based UAV Guidance," in *2005 IEEE Conference on Aerospace*, pp. 1–13, Mar. 2005.

[126] Shanmugavel, M., Tsourdos, A., Zbikowski, R., and White, B. A., "3D Dubins Sets Based Coordinated Path Planning for Swarm of UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, CO), Aug. 2006. AIAA 2006-6211.

[127] Shaw, A., Barnes, D., and Summers, P., "Landmark Recognition for Localisation and Navigation of Aerial Vehicles," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 42– 47, Oct. 2003.

[128] Shim, D. H., Chung, H., Kim, H. J., and Sastry, S., "Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (San Francisco, CA), Aug. 2005. AIAA 2005-6478.

[129] Shim, D. H., Kim, H. J., and Sastry, S., "Decentralized Nonlinear Model Predictive Control of Multiple Flying Robots," in *Proceedings of the 42^{nd} IEEE Conference on Decision and Control*, (Maui, HI), pp. 3621–3626, Dec. 2003.

[130] Shin, K. and McKay, N., "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 491–500, 1986.

[131] Singh, L. and Fuller, J., "Trajectory Generation for a UAV in Urban Terrain, using Nonlinear MPC," in *Proceedings of the American Control Conference*, (Arlington, VA), pp. 2301–2308, June 2001.

[132] Smetana, F. O., *Computer Assisted Analysis of Aircraft Performance, Stability, and Control.* New York: McGraw-Hill, 1984.

[133] Sorton, E. F. and Hammaker, S., "Simulated Flight Testing of an Autonomous Unmanned Aerial Vehicle Using FlightGear," in *Infotech@Aerospace*, (Arlington, VA), Sept. 2005. AIAA 2005-7083.

[134] Spooner, J. T., Maggiore, M., Raúl Ordóñez, and Passino, K. M., *Stable Adaptive Control and Estimation for Nonlinear Systems*. New York, NY: A John Wiley & Sons, Inc., 2002.

[135] Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310–3317, May 1994.

[136] Stentz, A., "Map-based Strategies for Robot Navigation in Unknown Environments," in *Proceedings AAAI 1996 Spring Symposium on Planning with Incomplete Information for Robot Problems*, (Menlo Park, CA), pp. 110–116, 1996.

[137] Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*. Hobokren, NJ: John Wiley & Sons, 2nd ed., 2003.

[138] Sweldens, W. and Schröder, P., "Building Your Own Wavelets at Home," in *Wavelets in Computer Graphics*, pp. 15–87, ACM SIGGRAPH Course notes, 1996.

[139] Sweldens, W., "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions," in *Wavelet Applications in Signal and Image Processing III*, pp. 68–79, 1995.

[140] Sweldens, W., "The Lifting Scheme: A Construction of Second Generation Wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, 1997.

[141] Thorpe, C. E., "Path Relaxation: Path Planning for a Mobile Robot," in *National Conference on Artificial Intelligence*, 1984.

[142] Tsiotras, P. and Bakolas, E., "A Hierarchical On-Line Path Planning Scheme using Wavelets," in *Proceedings of the European Control Conference*, (Kos, Greece), July 2007.

[143] Twigg, S., Calise, A., and Johnson, E., "On-line Trajectory Optimization for Autonomous Air Vehicles," in *AIAA Guidance, Navigation, and Control Conference*, (Austin, TX), pp. AIAA 2003–5522, 2003.

[144] Unmanned Dynamics, http://www.u-dynamics.com/, *AeroSim User's Guide - Aeronautical Simulation Blockset Ver. 1.2.*

[145] Uytterhoeven, G., Roose, D., and Bultheel, A., "Wavelet transforms using lifting scheme," technical report ita-wavelets report wp 1.1, Katholieke Universiteit Leuven, Belgium, Apr. 1997.

[146] Valenti, M., Schouwenaars, T., Kuwata, Y., Feron, E., and How, J., “Implementation of a Manned Vehicle - UAV Mission System,” in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Providence, RI), Aug. 2004. AIAA 2004-5142.

[147] Vázquez G., B., Sossa A., J. H., and Díaz-de-León S., J. L., “Auto Guided Vehicle Control Using Expanded Time B-splines,” in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, (San Antonio, TX), pp. 2786–2791, Oct. 1994.

[148] Vörös, J., “Low-cost Implementation of Distance Maps for Path Planning using Matrix Quadtrees and Octrees,” *Robotics and Computer Integrated Manufacturing*, vol. 17, pp. 447–459, 2001.

[149] Walnut, D. F., *An Introduction to Wavelet Analysis.* Boston: Birkhäuser, 2002.

[150] Williams, J. E. and Vukelich, S. R., “The USAF Stability and Control DATCOM,” tech. rep., McDonnell Douglas Astronautics Company, St Louis, MO, 1979. AFFDL-TR-79-3032.

[151] Yang, G. and Kapila, V., “Optimal Path Planning for Unmanned Air Vehicles with Kinematic and Tactical Constraints,” in *Proceedings of the 41st IEEE Conference on Decision and Control*, (Las Vegas, NV), pp. 1301–1306, Dec. 2002.

[152] Yang, H. I. and Zhao, Y. J., “Trajectory Planning for Autonomous Aerospace Vehicles amid Known Obstacles and Conflicts,” *Journal of Guidance, Control, and Dynamics*, vol. 27, pp. 997–1008, Nov.-Dec. 2004.

VITA

Dongwon Jung was born in Chungju, Korea, on January 1973. He received a bachelor's and a master's degrees at the Seoul National University, Seoul, Korea in 1998 and 2000, respectively. In Fall 2001, he began his studies for a Ph.D degree in the School of Aerospace Engineering at Georgia Institute of Technology, Atlanta, Georgia, USA.

His current research interest is to develop control algorithms for embedded systems that take into account the limited hardware resources, including multiresolution path planning, path generation/tracking, and validation of control algorithms through experiments.

A Hierarchical On-Line Path Planning Scheme using Wavelets

A Thesis
Presented to the Academic Faculty

by

Efstathios Bakolas

In partial fulfillment
of the requirements for the degree of
Master of Science
in Aerospace Engineering

Georgia Institute of Technology
School of Aerospace Engineering

School of Aerospace Engineering
Atlanta, Georgia 30332-0150 U.S.A.
May, 2007

A Hierarchical On-Line Path Planning Scheme using Wavelets

Approved by:

Date Approved: March 26, 2007

Epigraph

"... instead of the great number of precepts of which logic is composed, I believed that the four following would prove perfectly sufficient for me, provided I took the firm and unwavering resolution never in a single instance to fail in observing them.

The first was never to accept anything for true which I did not clearly know to be such; that is to say, carefully to avoid precipitancy and prejudice, and to comprise nothing more in my judgement than what was presented to my mind so clearly and distinctly as to exclude all ground of doubt.

The second, to divide each of the difficulties under examination into as many parts as possible, and as might be necessary for its adequate solution.

The third, to conduct my thoughts in such order that, by commencing with objects the simplest and easiest to know, I might ascend by little and little, and, as it were, step by step, to the knowledge of the more complex; assigning in thought a certain order even to those objects which in their own nature do not stand in a relation of antecedence and sequence.

And the last, in every case to make enumerations so complete, and reviews so general, that I might be assured that nothing was omitted."

"Discourse on the Method of Rightly Conducting One's Reason and of Seeking Truth in the Sciences",
Descartes René,
translation of Discours de la méthode,
Gutenberg Project at http://www.gutenberg.org.

Acknowledgements

I wish to express my appreciation to Dr. P. Tsiotras, Dr. M. Egerstedt and Dr. E. Feron, members of my Master Thesis committee, for their true interest in evaluating this work.

Furthermore, I would like to thank the A. Onassis Public Benefit Foundation, of which I have been a scholarship recipient for the last two years. This research work has been supported in part by NSF (award no. CMS-0510259) and ARO (award no. W911NF-05-1-0331). Their financial support is greatly appreciated.

Finally, I owe the greatest debt of gratitude to my family for the moral and emotional support they have provided me, especially during the years of my academic studies. Their affluent help has been a strong motivation and source of inspiration for continuing to fight for my ideals and ambitions. For these reasons I dedicate this work to them as the least token of appreciation.

Contents

List of Tables

List of Figures

Summary

The main objective of this thesis is to present a new path planning scheme for solving the shortest (collision-free) path problem for an agent (vehicle) operating in a partially known environment. We present two novel algorithms to solve the planning problem. For both of these approaches we assume that the agent has detailed knowledge of the environment and the obstacles only in the vicinity of its current position. Far away obstacles or the final destination are only partially known and may even change dynamically at each instant of time. The path planning scheme is based on information gathered on-line by the available on-board sensor devices. The solution minimizes the total length of the path with respect to a metric that includes actual path length, along with a risk-induced metric. In order to obtain an approximation of the whole configuration space at different levels of fidelity we use a wavelet approximation scheme. In the first proposed algorithm, the path-planning problem is solved using a multi-resolution cell decomposition of the environment obtained from the wavelet transform. In the second algorithm, we extend the results of the the first one by using the multiresolution representation of the environment in conjunction with a conformal mapping to polar coordinates. By performing the cell decomposition in polar coordinates, we can naturally incorporate sector-like cells that are adapted to the data representation collected by the on-board sensor devices.

Our approach is motivated by many typical navigation problems for autonomous vehicles, where a plethora of sensory devices used (e.g., cameras, radars, laser scanners, satellite imagery) usually have different ranges and resolutions. The proposed algorithms provide a computationally efficient path planning scheme which is able to combine the information provided by all the sensors in such a way that the computational resources are used on that part of the path (spatial and temporal) that needs them most.

The proposed planning scheme takes full advantage of any local information around the agent's current position. This allows the construction of a directed weighted graph of the approximated free space, the dimension of which is adapted to the on-board computational resources. By searching this graph we find the desired shortest path to the final destination using the Dijkstra's algorithm, provided that such a path exists. The path is a finite ordered sequence of visiting points corresponding to a polygonal line connecting the initial and goal destination. When dynamic constraints on the agent's motion are taken into consideration this polygonal line may be dynamically infeasible. Therefore, we introduce an *on-line* scheme which generates a collision-free, smooth trajectory. The trajectory passes sufficiently close to the points of the path at specified instants of time (time scheduling). For this scheme we employ a kinematic ground vehicle model to describe the agent's equations of motion and then apply some ideas of the input/output linearization theory (IOL) for MIMO systems to accomplish the trajectory generation and time scheduling tasks.

Finally, several simulations are presented to test the efficiency of the proposed path planning and trajectory generation schemes using non-trivial scenarios.

Chapter 1

Introduction

1.1 Introduction

The path planning problem has been under intense investigation for many years in several research areas, ranging from operation research, vehicle navigation, network optimization, etc. In the area of the autonomous vehicle navigation a major distinction between different path planning problems is based on whether the information about the operating environment is known *a priori* or is updated with time. In the first case, the problem is known as a *static* path planning problem whereas, the second case corresponds to a *dynamic* problem (see [1]). From an application point of view, dynamic problems are more interesting than static ones since autonomous vehicles typically have on-board sensors that obtain information about the operating environment dynamically. On the other hand, dynamic problems are more challenging, since they require the use of algorithms that can be implemented *on-line*. In the past, this kind of implementation was restricted by hardware limitations, and therefore the path designers had to rely more or less on *off-line* solutions. With the recent significant advances in computing technology, real time implementation of path planning algorithms has become easier. As a result, new path planning schemes which can be implemented "on the fly" have appeared over the past decade. Path planning schemes using feedback control laws and the D^* algorithm are some of the most significant examples of this era. In this work we deal with the vehicle navigation problem where the vehicle (e.g., ground vehicle, UAV) is operating inside a partially known environment $\mathcal{W}$.

In the next section we introduce the reader to some basic concepts of the path planning problem and present some significant results that have appeared in the literature and are relevant to our problem.

1.2 The Path Planning Problem

The general framework for the path planning problem is as follows: *given a topological space $\mathcal{F} \subset W$ of admissible states $\mathsf{x}_{\mathcal{F}}$, find a path connecting the prescribed initial state $\mathsf{x}_0 \in \mathcal{F}$ with the destination state $\mathsf{x}_f \in \mathcal{F}$ such that the path lies completely inside $\mathcal{F}$.* The construction of such a path can take place using either a continuous representation of $\mathcal{W}$ or a discrete one. Seeing the problem from the geometrical point of view (no kinematic or dynamic constraints) a sufficient condition for a solution path to exist is that the space $\mathcal{F}$ is polygonally (or path) connected, i.e. that for any two configuration inside $\mathcal{F}$ there exist a path which lies completely inside $\mathcal{F}$ connecting these two configurations. In discrete topologies like finite graph structures comprised of nodes, a sufficient condition is that, for any two nodes in $\mathcal{F}$, there exists a sequence of nodes, adjacent to each other, connecting these two nodes.

1.2.1 Planning Inside Continuous Spaces

In applications where dynamic constraints are taken into consideration the most natural approach is to work with continuous spaces. Then the path planning can be treated as a two boundary value problem (the fixed final point and free final time problem) where application of standard control techniques, such as optimal control theory (see [2]), can be employed.

We define the configuration space $\mathcal{W} \subset \mathbb{R}^n$ to be the space that contains all the possible states x of the agent such that

$$\mathcal{W} = \mathcal{F} \oplus \mathcal{O}, \tag{1.1}$$

where $\mathcal{F}$ denote the space of all admissible states $\mathsf{x}_{\mathcal{F}}$ known as the free configuration space and $\mathcal{O}$ the obstacle configuration space which contains all the unfeasible states. Finally, we denote with $\mathcal{U}$ the function space of all admissible inputs u which take values in the space $U \subset \mathbb{R}^m$. Let's assume that the dynamics that govern the motion of the agent are given in the form

$$\dot{\mathsf{x}}(t) = f(\mathsf{x}(t), u(t)), \quad \mathsf{x}(0) = \mathsf{x}_0 \;\; t \geq 0 \tag{1.2}$$

where we consider the time invariant case for simplicity. We additionally assume that the space $\mathcal{F}$ is open and connected and the function f satisfies a Lipschitz condition for all $\mathsf{x} \in \mathcal{F}$. The condition on $\mathcal{F}$ being a region (open and connected subset of $\mathbb{R}^n$) is equivalent to the polygonally connected requirement (see [3]) introduced earlier whereas the Lipschitz condition on f is needed so that a solution path always exist (but may not be unique, see [4]). Then the path finding problem is the one of finding a control input function $u \in \mathcal{U}_{[0,t_f]}$ that drives the agent from the initial state x_0 to the final state x_f in finite time t_f, while the whole trajectory from x_0 to x_f lies entirely on $\mathcal{F}$.

If an optimal feedback control cannot be derived for our problem then the optimal control solution may not be plausible from the application point of view. This is due to the sensitivity that may be exhibited to variations of the initial and final configurations or to any environment parameters. One popular technique to proceed is the potential function method (see [5, 6]). The potential functions are scalar functions $\phi : \mathcal{F} \mapsto \mathbb{R}$ which attains their global minimum at the final destination state. These sufficiently smooth functions, known also as navigation functions, provide control laws in feedback form which result in smooth collision free trajectories which may given in closed mathematical expressions (analytic solutions). For these reasons navigation function are very efficient for real time implementation. The feedback control law is given by

$$u(t) = -K\nabla\phi(\mathsf{x}(t)) \tag{1.3}$$

where $\nabla\phi(\mathsf{x})$ denotes the gradient vector of the navigation function at state x where $K \in \mathbb{R}$ is a gain matrix. With $\dot{\mathsf{x}}(t) = f(\mathsf{x}(t), -K\nabla\phi(\mathsf{x}(t))) = \tilde{f}(\mathsf{x}(t))$ be the closed loop dynamics, the trajectory $\mathsf{x}(t)$ of the agent at some time t is now given by

$$\mathsf{x}(t) = \mathsf{x}_0 + \int_0^t \tilde{f}(\mathsf{x}(\tau))\mathrm{d}\tau. \tag{1.4}$$

A drawback of this approach is the existence of local minima or even stationary points which may not allow the agent to reach its destination even though a solution path may exist. Under some specific assumptions on the geometry of the working environment, the structure of obstacles, and the agent's governing equations of motion, results that guarantee that the trajectory of an agent driven by a feedback control law induced by a potential function converges to the desired destination were first presented in [7, 8]. The approach used in the latter method is the construction of a function which attains its minimum value at the destination state. The final state in this case is the unique isolated minimum of $\phi(\mathsf{x})$ which additionally corresponds to an asymptotically stable equilibrium of the closed loop dynamics. Thus, when the agent's configuration is arbitrarily close to the destination state, the agent practically halts there. However, the construction of the potential function in arbitrary geometries of either the configuration or obstacle space without local minima other than the destination state is not an easy task and general results have not been found. Additionally, the task of finding an appropriate potential function becomes more complicated as the complexity of the agent's governing equations is increased.

1.2.2 Planning Inside Discrete Spaces

Due to the existence of many shortest path and network minimization algorithms, planning inside discrete spaces has become very popular in many path planning applications. The discrete space $\mathcal{X}$, which is the equivalent of the continuous configuration $\mathcal{W}$, is defined as the countable collection of all possible states x such that

$$\mathcal{X} = \mathcal{X}_{\text{free}} \oplus \mathcal{X}_{\text{unfeasible}}$$

where $\mathcal{X}_{\text{free}}$ and $\mathcal{X}_{\text{unfeasible}}$ denote the corresponding subcollections of all feasible and unfeasible states $\mathsf{x} \in \mathcal{X}$ respectively. Let now x_k be the state at some time t_k, then the state at the next time step t_{k+1} is given by

$$\mathsf{x}_{k+1} = f(\mathsf{x}_k, u_k) \tag{1.5}$$

where u_k is an action applied at time t_k to the agent with $u_k \in \mathcal{U}$, where $\mathcal{U}$ denotes the countable collection of all available admissible inputs, and f is a state transition function

$$f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$$

which behaves as the 'dynamics' of this discrete system. The problem then becomes one of finding the appropriate sequence of actions u_i which will drive the agent form an initial state $\mathsf{x}_0 \in \mathcal{X}$ to the final destination $\mathsf{x}_f \in \mathcal{X}$ after a finite number of steps.

A common method to pass from the real environment (continuous space) to the discrete world is by using cell-based approximations of the environment and then employ a graph representation scheme. By transcribing the free space $\mathcal{F}$ on a graph $\mathcal{G}$ the problem reduces to one of finding a sequence of adjacent nodes in the graph $\mathcal{G}$ from the starting node to the destination node. These nodes form a connected sequence of nodes of the graph, provided that such a sequence exists. In case where more than one feasible solutions exist, optimization criteria determine the one that will qualify for the agent's path. The problem is therefore reduced to a network minimization or a graph search problem. Algorithms that find a solution to the graph search problem or prove that the problem has no solution are called *exact* or *complete* algorithms (e.g. Dijkstra's algorithm, A^* algorithm, see [9, 10]). These path planning algorithms will find the solution even when the polygonal connected condition for the entire free space $\mathcal{F}$ is not satisfied, provided of course that a solution exists for the specific choice of the initial and final node. Furthermore, in case where there does not exist a polygonally connected subset of $\mathcal{F}$ that contains the initial and final nodes, then Dijkstra's algorithm and the A^* algorithm will determine that the specific problem is infeasible. Another category of planning algorithms are the heuristic algorithms which generate a solution while aiming at lower computational complexity (see [6, 5]).

All the algorithms we mentioned in the previous paragraph are mostly used to solve static problems. On the other hand, dynamic problems require appropriate modifications to the original static path planning schemes. The basic idea is to deal with the dynamic problem as a sequence of different static problems. Each one of these static problems corresponds to the planning problem inside the environment representation of the specific time step (continuous replanning). A more straightforward approach for the dynamic problem is the D^* algorithm (or Stentz's algorithm) which is presented in [11]. The D^* algorithm is the first path planning algorithm for discrete spaces which deals exclusively with the dynamic problem while having significant advantages over the other available dynamic schemes (e.g. reduced computational complexity, optimality in the global sense). In Stentz's algorithm the operating environment is assumed to be partially known and therefore the transition costs between nodes of the corresponding graph are originally unknown. The agent's sensors provide information about the agent's immediate environment and this information is processed in a heuristic way. This allows the construction of the true operating map with reduced computational complexity. Subsequently, the algorithm uses a graph search approach, which can be characterized as a dynamic extension of Dijkstra's algorithm, in order to find the optimal solution in a complete way. Improved versions of the D^* algorithm, namely the delayed D^* and the D^* lite algorithms, are presented in [12, 13] respectively.

An important issue that should be dealt within the discrete approach is the "smoothness" requirement of the generated path, since in realistic situations the vehicle under dynamic constraints has to follow a smooth trajectory. In the literature (see [14]) one can find modified shortest path algorithms which assign an additional penalty cost when the angle between two successive arcs of the path is larger than a threshold angle $\theta_{\max}$. By imposing this "smoothness" constraint, the sequence of nodes that solve the network minimization problem form a smooth enough polygonal line joining x_0 and x_f. Then, one can directly proceed to the trajectory planning problem by determining the linear and angular velocities required to drive the agent to the goal destination. The resultant trajectory has to be close enough, in a pointwise sense, to the polygonal path that the graph search algorithm found. This requirement is a direct consequence of the polygonal connected requirement for the space $\mathcal{F}$ since no other curve except from the polygonal path is guaranteed to lie completely inside $\mathcal{F}$.

The most important difficulty of the graph search approach lies in the dimensionality of the problem. To elucidate this point, let us assume that we implement an on line path planning scheme where all the data of the environment are available at all times. On the other hand, the available computational resources are limited due to hardware limitations. It is obvious that the higher the amount of data processed, the more accurate the solution path will be. Thus, the dimension of $\mathcal{G}$ (i.e., the number of nodes) becomes very large as the fidelity of the approximation of $\mathcal{F}$ and/or $\mathcal{W}$ increases. Furthermore, the increase in the number of the nodes of the graph is very likely to increase the adjacency relations between different nodes. In all cases, the graph search task becomes more demanding and the problem more computationally complex. Sooner or later, the computational resources on-board the agent reach their limit; hence, an accurate solution cannot be generated as often as desired, and therefore the agent capacity to deal with rapidly changing situations is also limited.

1.3 Thesis Motivation and Objectives of The Proposed Path-Planning Schemes

1.3.1 Introduction

From the previous observations, a dynamic path planning scheme which can handle changes in the vicinity of the agent while requiring a reduced amount of computational resources will be useful in many applications in the field of UAV, UGV navigation and robot motion planning. One straightforward approach to deal with this problem is to design a path planning algorithm based solely on local information of the configuration space. In that case replanning is easier than with global methods; there exist obstacle configurations nonetheless, where such local methods are not guaranteed to find a solution even if such a solution exists. This situation is depicted in Fig. 1.1 where the agent starting from 'A' fails to find the route to the goal destination 'B' when the visibility/exploration horizon of its sensors are not sufficient large. As the exploration horizon is increased, the construction of a collision-free path becomes more likely. The increase of the exploration horizon, however, comes at the expense of an increase of the required on-board computational resources. Therefore, a path planning algorithm based on local information may not be efficient in practice unless the initial and final states are sufficiently close to each other (see [1]). On the other hand, global path-planning algorithms in realistic situations, where the agent's computational resources are limited, are restricted to use coarse representations of the whole environment. Thus, the global information approach fails to take into consideration the high fidelity information in the vicinity of the agent's current position. Subsequently, the real time implementation of a global path-planning scheme becomes problematic since the agent is not appropriately sensitive to crucial changes in its immediate environment.

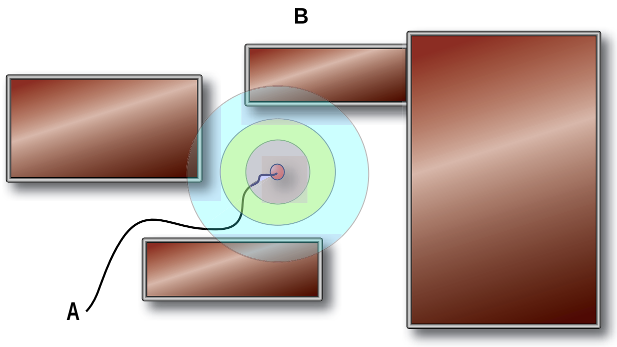


Figure 1.1: A path generated by a short visibility/exploration horizon planning scheme.

1.3.2 Thesis Motivation

Strong motivation for this thesis is the design of a path planning scheme where the operating environment of the vehicle is assumed to be only partially known. This motivation comes from many applications of autonomous vehicle navigation problems, where the assumption of global knowledge of the environment is an overconservative approach. One can think of an unmanned aerial vehicle operating in a hostile environment with the mission to identify an enemy target, where areas of high risk are not known *a priori*. Based on the information gathered by the on-board sensor devices, the UAV should find the route to the destination while avoiding to get close to areas of high risk. It is evident that its ability to react immediately to an obstacle or popup threat is crucial to the success of its mission. In this case, as with many other applications of vehicle navigation, different sensor devices can provide information of the environment in the vicinity of the vehicle. The information obtained by different devices should be weighted appropriately, since the sensors, (cameras, radars, laser scanners, satellite imagery, etc.) have different ranges and resolutions.

1.3.3 Thesis Objectives

In this work, we introduce two hybrid local/global path planning algorithms that use district levels of fidelity (resolution) of the environment at different distances from the agent's current position. The first planning scheme uses cell decompositions of the agent's operating environment constructed using wavelets and standard quadtree decompositions. The second scheme, finds the solution path using sector cell approximations. This is a more natural approach given the sector like topology of the areas scanned *on-line* by the on-board sensors of the vehicle. In the sequel, when we will use the term the "*path planning scheme*" we will refer to the general idea of multiresolution path planning lying behind both of these algorithms. In cases where we want to distinguish between the two, we do so explicitly.

The goal of the proposed *path planning scheme* is to find a sequence of points in the world $\mathcal{W}$ (operating environment of the vehicle) that the agent should visit in order to reach the final destination x_f (perhaps not accurately known a priori). The initial state x_0 is assumed to be prescribed whereas the final time is a free parameter of the problem. All the points of the sequence should lie in a polygonally connected subspace of the free configuration space $\mathcal{F}$. This requirement is needed so that the polygonal line that connects the points of the sequence lies completely inside $\mathcal{F}$.

Since the resultant path of the *planning scheme* is a finite sequence of ordered points, the next step is to introduce a methodology for generating a smooth trajectory under dynamic constraints. We employ a kinematic ground vehicle model to obtain the equations of motion of the agent, and additionally we introduce a trajectory generating scheme using some basic ideas from input/output linearization theory (IOL) for MIMO systems (see [15]). This scheme produces a smooth curve passing sufficiently close to the visiting points of the path at specified instants of time (time scheduling).

1.3.4 The Multiresolution Hierarchical Approach

The success of a multi-resolution path planning algorithm hinges on its ability to compute the obstacle boundaries well in advance and with sufficient accuracy, by keeping a balance between an overconservative approximation of the environment and the computation of a collision-free path.

In this work we use wavelets to obtain multiresolution approximations of the configuration space at different distances from the vehicle. The computational complexity of the wavelet transform is of order $O(n)$ where n is the input data [16] while even than the Fast Fourier Transform has complexity of order $O(n \log_2 n)$. Therefore the wavelet transform provides a very fast decomposition of the environment at different levels of resolution. From the output of the wavelet transform we construct cell-based approximations (rectangular or sector-like) of the whole environment $\mathcal{W}$ having high/fine resolution close to the current position of the vehicle and low/coarse resolution far away. The number of resolution levels, their scale, and range can all be readily adapted at each time step to yield graph representations that are commensurable to the available on-board computational resources.

We employ the hierarchical path planning principle to find the optimal path on a topological graph $\mathcal{G}$ induced by the previous cell decomposition. Namely, the path may contain mixed nodes at all resolution levels except the finer resolution level, where it is assumed that nodes can be confidently resolved as either free or occupied. Mixed nodes, on the other hand are not known with certainty whether they belong to the free or the obstacle space. Hierarchical path planning is known to be more flexible than methods that search only through free nodes [17]. In hierarchical path planning the mixed nodes are subsequently resolved to free or occupied nodes, as the agent gets closer to the obstacles and more information about their shape and location becomes available.

1.3.5 Literature Review

Several multi-resolution or hierarchical algorithms have been proposed in the literature for path planning [18, 19] The majority of those use some form of quadtree decomposition of the environment. One drawback of quadtree-based decompositions is that a finer resolution is used close to the boundaries of all obstacles, regardless of their distance from the agent. This tends to waste computational resources. One of the central references in the context of quadtree-based cell decompositions is perhaps [19], where the authors present a hierarchical path planning scheme based on a multistage quadtree decomposition. Both free and mixed nodes are included in the search, which is conducted using the A^* algorithm. A path to the target is first computed using a coarse grid and subsequently refined using information from higher resolution levels, uniformly along the path. Even though this technique is efficient and easy to implement, it fails to take full advantage of the local information around the agent. Wavelets for multi-resolution decomposition of the environment have also been used in [18]. The approach in [18] combines a more efficient model for the local behavior of the approximation, with improved computational characteristics, compared to the one proposed in [19]. The main emphasis in [19] is to construct a smooth path. This is easily achieved by the information provided by the detail coefficients in the wavelet expansion. The smoothness requirement is then embedded in the transition cost of the agent. The reference most closely related to our approach is [20]. Therein, the author also uses the idea of coarse/fine grid at close/far distances from the current location of the agent in order to avoid the demerits of uniform grids or standard quadtrees. Nonetheless, no connection with wavelets is attempted. In addition, the multiresolution scheme in [20] requires a rather careful handling of the cell connectivity at the boundaries between two different resolution levels. This is handled automatically in our approach.

1.4 Thesis Overview

In this section we briefly describe each chapter in this thesis by and we point out interrelationships between these chapters.

Chapter 2: Technical background

We discuss some basic notions from topology, graph theory and wavelet approximation theory and we present Dijkstra's algorithm and the elementary theory of the quadtree decomposition algorithm. These topics are important for a deeper understanding of the subsequent chapters. A reader familiar with these concepts can skip this chapter in a first reading.

Chapter 3: Multiresolution Path planning Using Rectangular Cell Decompositions

We introduce the first path planning scheme of this thesis. The proposed algorithm constructs the shortest path based on rectangular cell decompositions of the world space $\mathcal{W}$ induced by a *Haar* wavelet multiresolution approximation scheme of a risk measure function defined over $\mathcal{W}$.

Chapter 4: Multiresolution Path Planning Using Sector Decomposi-

tions

We present the second path planning scheme of this thesis, which is a natural extension of the algorithm presented in Chapter 3. Now the planning takes place in sector-cell decompositions of the world $\mathcal{W}$. This approach is compatible with the specific form of data obtained by on-board sensors of an autonomous vehicle.

Chapter 5: Time Scheduling and Smooth Trajectory Generation

We discuss ways to generate a smooth trajectory passing sufficiently close to the visiting points given by the *path planning scheme* presented in the previous two chapters. We wish this closeness requirement to be satisfied at specific time instants. We first introduce a kinematic model to describe the equations of motion of the agent, and we subsequently specify an appropriate speed profile to accomplish the trajectory generation and time scheduling tasks. Ideas from input/output linearization theory for MIMO systems are employed to achieve this objective.

Chapter 6: Simulation Results

We present simulation results to test the efficiency of the two proposed planning algorithms and the trajectory generating and time scheduling scheme.

Chapter 7: Conclusions and Future Work

We discuss the conclusions from our approach to the path planning problem. Ideas and possible extensions are also briefly discussed.

Chapter 2

Technical background

2.1 Introduction

In this chapter we present some basic notions that we will use extensively in this work. First, we present the basic theory of cell decomposition schemes, which are typically used in robot motion planning application [5, 6] and we describe how we incorporate the cell decomposition approach in our problem. Then we introduce the elementary theory of the wavelet transform. In this work we employ wavelet schemes to obtain cell decompositions of the working environment with special localized attributes. Furthermore, we present the basic ideas behind the Dijkstra's algorithm which we use to solve the shortest path problem. For a more detailed presentation of the shortest path problem the reader is encouraged to consult the suggested references relevant to this topic [21, 9].

2.2 Cell Decomposition

An m-cell decomposition $\mathcal{C}_d$ of $\mathcal{W}$ is a finite collection of m cells

$$\mathcal{C}_d = \{c_i \in \mathcal{W} : \; i = 1, \ldots, m\} \tag{2.1}$$

with the following properties:

1. $\mathcal{W} = \bigcup_{i=1}^{m} c_i$

2. $\mathrm{int}(c_i) \bigcap \mathrm{int}(c_j) = \varnothing$

A cell decomposition algorithm generates a collection of cells by creating sequences of nested cells. One popular algorithm to accomplish this decomposition is the quadtree decomposition algorithm. The goal of this algorithm is the construction of a collection of square cells which contains only empty or full cells. In order to present the way the algorithm constructs these sequences, let us suppose that initially we have only one mixed cell c_1 that encloses the world $\mathcal{W}$ with ℓ_1 be the length of each of its sides. Then by bisecting each of the sides of the cell c_1 we take four new cells c_i, with $i = 1, \ldots, 4$ with corresponding side length $\ell_i = \ell_1/2$. The cell c_1 is the parent node and c_i are the children nodes of the corresponding expansion tree of depth $k = 2$. Since the aim of the cell decomposition algorithm is to generate only empty or full cells the algorithm will continue to subdivide each mixed cell creating four new cells with sides of length $\ell_i = \ell_1/2^{k-1}$, where k the depth of the corresponding expansion tree. The algorithm will terminate either when there no mixed cells or the resulting expansion tree reaches a predetermined depth.

Given two cell decompositions $\mathcal{C}_d$ and $\mathcal{C}'_d$ of $\mathcal{W}$ we say that $\mathcal{C}'_d$ is a *finer, or higher resolution* decomposition of $\mathcal{W}$ than $\mathcal{C}_d$ if and only if for every cell $c_i \in \mathcal{C}_d$ there exists an integer $p_i > 1$ such that $c_i = \bigcup_{l=1}^{p_i} c'_l$ where $c'_l \in \mathcal{C}'_d$.

We may define the following three categories of cells:

1. empty cells, when $c_i \cap \mathcal{O} = \varnothing$
2. mixed cells, when $c_i \cap \mathcal{O} \neq \varnothing$
3. full cells, when $c_i \subseteq \mathcal{O}$.

We will say that two cells c_i and c_j are adjacent if

$$\partial c_i \cap \partial c_j \neq \varnothing, \quad i \neq j, \tag{2.2}$$

where ∂c_i denotes the boundary of the cell c_i. A cell decomposition of a square environment is presented in Fig. 2.1. The corresponding to this decomposition free, mixed and full cells are given in the Table 2.1.

2.3 Graph Representation

To a cell decomposition $\mathcal{C}_d$ we will associate a directed graph $\mathcal{G} = (V, E)$ with nodes V and edges E, known as the connectivity graph, such that:

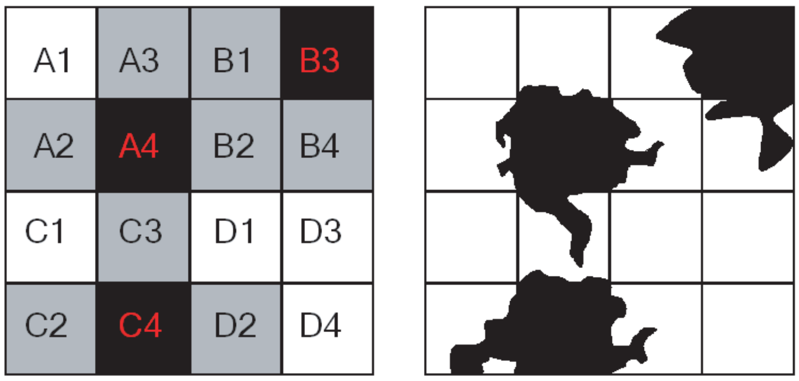


Figure 2.1: Quadtree decomposition scheme. Right figure depicts the world $\mathcal{W}$ where the white areas correspond to $\mathcal{F}$ and black to $\mathcal{OB}$. In the left figure we see the obtained cell decomposition after we apply the quadtree algorithm.

Table 2.1: Cell characterization for the decomposition depicted in Fig. 2.1.

Full Cells	A4, B3, C4
Mixed Cells	A2, A3, B1, B2, B3, B4, C2, C3, D2
Free Cells	A1, C1, D1, D2, D3

1. The nodes of $\mathcal{G}$ correspond to the free and mixed cells of $\mathcal{C}_d$
2. The edges of $\mathcal{G}$ correspond to cells that are adjacent to each other

It is easy to see that $\mathcal{G}$ is a topological graph [5].

In this work we use the 8-connectivity scheme to define adjacency relationships between nodes. This is an immediate result of the equation (2.2). This adjacency scheme is presented in Fig. 2.2. To each ordered pair inside E we associate a cost of transition. In our *path planning scheme* the order of a transition is important for the cost assignment procedure, i.e the connectivity graph is in our case a directed graph.The procedure of cost assignment is described in detail in the subsequent chapters. Finally, the reader interested in a deeper understanding of topics from graph theory should refer to [22, 23, 24].

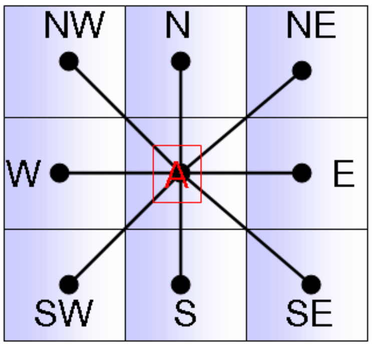


Figure 2.2: The 8-connectivity adjacency scheme.

2.4 The 2D Wavelet Transform

The idea behind the theory of the wavelet transform is to represent a function $f \in \mathcal{L}_2(\mathbb{R})$ as a summation of elementary basis functions $\phi_{J,k}$ and $\psi_{j,k}$ as follows

$$f(x) = \sum_{k\in\mathbb{Z}} a_{J,k}\phi_{J,k}(x) + \sum_{j\geq J}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x), \tag{2.3}$$

where $\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k)$ and $\psi_{j,k} = 2^{j/2}\psi(2^j x - k)$. In the ideal case both $\phi(x)$ (scaling function) and $\psi(x)$ (mother wavelet) have compact support or they decay very fast outside a small interval so they can capture localized features of f. The first summation in (2.3) gives a low resolution, or coarse, approximation of f. The second term in (2.3) gives the difference (details) between the original function and its low resolution approximation. For example, when analyzing a signal at the coarsest level (low resolution) only the general, most salient, features of the signal will be revealed. The index j denotes the resolution level. For each increasing index j, a higher, or finer resolution term is added, which adds more and more details. The expansion (2.3) thus reveals the properties of f at different levels of resolution [25, 26, 27].

This idea can be readily extended to the two-dimensional case by introducing the following families of functions

$$\Phi_{j,k,\ell}(x,y) = \phi_{j,k}(x)\phi_{j,\ell}(y) \tag{2.4}$$

$$\Psi^1_{j,k,\ell}(x,y) = \phi_{j,k}(x)\psi_{j,\ell}(y) \tag{2.5}$$

$$\Psi^2_{j,k,\ell}(x,y) = \psi_{j,k}(x)\phi_{j,\ell}(y) \tag{2.6}$$

$$\Psi^3_{j,k,\ell}(x,y) = \psi_{j,k}(x)\psi_{j,\ell}(y) \tag{2.7}$$

Given a function $f \in \mathcal{L}_2(\mathbb{R}^2)$ we can then write

$$\begin{aligned} f(x,y) = &\sum_{k,\ell\in\mathbb{Z}} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) \\ &+ \sum_{i=1}^{3}\sum_{j\geq J}\sum_{k,\ell\in\mathbb{Z}} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y) \end{aligned} \tag{2.8}$$

where, for the case of orthonormal wavelets the approximation coefficients are given by

$$a_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\phi_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{2.9}$$

and the detail coefficients by

$$d^i_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Psi^i_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{2.10}$$

In the more general case, biorthogonal wavelets projections on the space spanned by the dual wavelets and dual scaling functions should be used in (2.9) and (2.10). The key property of wavelets used in this work is the fact that the expansion (2.8) induces the following decomposition of $\mathcal{L}_2(\mathbb{R}^2)$

$$\mathcal{L}^2(\mathbb{R}^2) = \mathcal{V}_J \oplus \mathcal{W}^{detail}_J \oplus \mathcal{W}^{detail}_{J+1} \oplus \cdots \tag{2.11}$$

where $\mathcal{V}_J = \overline{\text{span}}\{\phi_{J,k,\ell}\}_{k,\ell\in\mathbb{Z}}$ and similarly $\mathcal{W}^{detail}_j = \overline{\text{span}}\{\psi^1_{j,k,\ell}, \psi^2_{j,k,\ell}, \psi^3_{j,k,\ell}\}_{k,\ell\in\mathbb{Z}}$ for $j \geq J$.

By using the Haar family of wavelets, each scaling function $\phi_{j,k}(x)$ and wavelet function $\psi_{j,k}(x)$ in the Haar system is supported on the dyadic interval $I_{j,k} \triangleq [k/2^j, (k+1)/2^j]$ of length $1/2^j$ and does not vanish in this interval [25, 28]. Subsequently, and via the tensor product in (2.4), we may associate the functions $\Phi_{j,k,\ell}$ and $\Psi^i_{j,k,\ell}$ $(i = 1, 2, 3)$ in the 2D case with the square cell $c^j_{k,\ell} \triangleq I_{j,k} \times I_{j,\ell}$.

2.5 Description of Dijkstra's Algorithm

A popular algorithm to find the shortest path between two nodes u and v of a graph $\mathcal{G}$ is the Dijkstra's algorithm. Dijkstra's algorithm is considered as one of the most efficient algorithms when a path between two nodes of a graph is needed (see [1]). Even though it is based on a greedy strategy the Dijkstra's algorithm always finds the optimal solution, provided that such a solution exists. Additionally, depending on the sparsity of the adjacency matrix and the particular implementation it is computationally more appealing than other standard shortest path algorithms, like the Bellman-Ford algorithm, see [9].

Below, we shall briefly present the basic principles of Dijkstra's algorithm. Given a graph $\mathcal{G}$, we associate to a transition from the node v_{i-1} to the node v_i of this graph a nonnegative cost $\mathcal{J}(v_{i-1}, v_i)$. The exact cost assignment procedure shall be described in the subsequent chapters where we introduce the multiresolution *path planning scheme*. The goal of the algorithm is to find the sequence of $1 + m_d$ nodes $\mathcal{P} = (v_0 = s, v_1, \ldots, v_{m_d} = d)$, that the agent has to visit, starting from the initial node s until reaches the destination node d, while at the same the total cost

$$\mathcal{H}(d) = \sum_{j=1}^{m_d} \mathcal{J}(v_{j-1}, v_j) \tag{2.12}$$

is minimized whenever $v_0 = s$. We call $\mathcal{P}$ the shortest path. It is clear that the solution path $\mathcal{P}$ inside the graph $\mathcal{G}$ is a function of the initial and final nodes. When part of the optimal path $\mathcal{P}$ is known we define the optimal weight $\hat{\mathcal{H}}(u_k, u_{k+n})$ to be the sum

$$\hat{\mathcal{H}}(v_k, v_{k+n}) = \sum_{j=k+1}^{n+k} \mathcal{J}(v_{j-1}, v_j), \tag{2.13}$$

where all the nodes appearing in the transition costs of the sum are elements of the optimal part of the path. When v_k is the starting node s then we denote the optimal weight between s and the node $v_{k+n} = u$ as $\hat{\mathcal{H}}(u)$. Dijkstra's algorithm finds $\hat{\mathcal{H}}(u)$ for all $u \in V$ until the value of $\hat{\mathcal{H}}(d)$ is known. This is accomplished with a help of a list S which contains all the nodes u for which $\hat{\mathcal{H}}(u)$ is known. At each iteration the algorithm picks one element w from V which is not in S and is closest to the finite set S. The node w is then an element of the list S. The way that the algorithm assigns the optimal value to the last node w which was inserted in the list S is based on the *principle of relaxation.* To see how the *principle of relaxation* works we let u, w be two adjacent nodes and assume that the set $S_x(u,w) \overset{\triangle}{=} \{x \in V : x \text{ is adjacent to both}\, u, w\}$ is nonempty. Let $x \in S_x(u,w)$, then the principle is defined by the following inequality

$$\mathcal{H}(u,v) \leq \mathcal{H}(u,x) + \mathcal{H}(x,v). \tag{2.14}$$

In the case where $u \in S$ the *principle of relaxation* states that $\mathcal{H}(u,w) = \hat{\mathcal{H}}(u,w)$ provided that the inequality (2.14) holds for every node $x \in S_x$. Otherwise $\hat{\mathcal{H}}(u,v) = \min_{x \in S_x}\{\mathcal{H}(u,x) + \mathcal{H}(x,v)\}$. For details the reader can refer to [9].

Chapter 3

Multiresolution Path planning Using Rectangular Cell Decompositions

3.1 Introduction

In this chapter we introduce an innovative approach to the path planning problem using ideas from wavelet theory. With this approach the agent is capable of reacting immediately to situations where the collision avoidance requirement is violated (i.e an obstacle is revealed to be in the agent's vicinity) or a threat suddenly appears (popup threat) while approaching the goal destination. The proposed algorithm assumes that far away obstacles or threats should not have a large effect on the vehicle's *immediate* motion, since either these regions will never be visited by the agent or more accurate and reliable information about them will become available when the agent gets closer to them while approaching the final destination. So high resolution information is needed only for the area in the vicinity of the agent's current position. This allows one to construct a graph with fewer nodes than the case where the whole environment is represented in a uniform fashion. One should note that the approach of using a different resolution to approximate the working environment is also based in practical issues since in many typical navigation problem a plethora of sensory devices used (e.g., cameras, radars, laser scanners, satellite imagery) have different ranges and resolutions. A computationally efficient path planning algorithm should be able to combine the information provided by all these sensors in such a way that the computational resources are used on that part of the path (spatial and temporal) that needs them most. Additionally, information about far away obstacles is taken into some consideration providing the planning scheme with a mechanism which "pulls" the agent to the final destination (long-term strategy).

The resultant solution-path of this algorithm is the one that minimizes the total length of the path with respect to a metric that includes actual path length along with a risk-induced metric. The risk-induced metric depends on the available environment representation and the way is defined is presented in detail in Section 3.3.

3.2 Wavelet decomposition of the risk measure

Without loss of generality, we take $\mathcal{W} = [0,1] \times [0,1]$, which is described using a discrete (fine) grid of $2^N \times 2^N$ dyadic points. The finest level of resolution $J_{\max}$ is therefore bounded by N. It follows from the previous discussion that the Haar wavelet decomposition of a function f defined over $\mathcal{W}$ at resolution level $J \geq J_{\min}$

$$\begin{aligned} f(x,y) = & \sum_{k,\ell=0}^{2^{J_{\min}-1}} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(x,y) \\ & + \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J-1} \sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x,y) \end{aligned} \tag{3.1}$$

induces a cell decomposition of $\mathcal{W}$ of square cells of size $1/2^J \times 1/2^J$.

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto [0,1]$ that represents the "risk measure" at the location $\mathsf{x} = (x,y)$. For instance, one may choose

$$\mathsf{rm}(\mathsf{x}) = \begin{cases} (d_{\max} - \min_{\mathsf{y}\in\mathcal{O}} \|\mathsf{x}-\mathsf{y}\|_\infty)/d_{\max}, & \text{if } \mathsf{x} \in \mathcal{F}, \\ 1, & \text{if } \mathsf{x} \in \mathcal{O}, \end{cases} \tag{3.2}$$

where $d_{\max} \overset{\triangle}{=} \max_{\mathsf{x}\in\mathcal{F}} \min_{\mathsf{y}\in\mathcal{O}} \|\mathsf{x}-\mathsf{y}\|_\infty$. Alternatively, one may think of rm as the probability that $(x,y) \in \mathcal{O}$.

Let us also assume that we have m distinct risk measure levels, say $M_1 < M_2 < \cdots < M_m$ where $M_i \in [0,1]$, $i = 1,\ldots,m$.

We will use the $\|\cdot\|_\infty$ norm to measure distances in $\mathcal{W}$. Consequently, all points within range r from the current location of the agent are given by

$$\mathcal{N}(\mathsf{x}, r) \triangleq \{\mathsf{y} \in \mathcal{W} : \|\mathsf{x} - \mathsf{y}\|_\infty \leq r\}. \tag{3.3}$$

Suppose now that we are given the desired levels of resolution of $\mathcal{W}$ as $J_{\min} \leq j \leq J_{\max}$ where $J_{\min}, J_{\max} \in \{1, \ldots, N\}$, with corresponding ranges $r(j)$ from the agent's current location. By this we mean that we wish all points $\mathsf{y} \in \mathcal{N}(\mathsf{x}, r(J_{\max}))$ to be described by resolution $J_{\max}$, all points $\mathsf{y} \in \mathcal{N}(\mathsf{x}, r(j-1)) \backslash \mathcal{N}(\mathsf{x}, r(j))$ to be described by resolution j, where $J_{\min} < j \leq J_{\max}$, and all points $\mathsf{y} \notin \mathcal{N}(\mathsf{x}, r(J_{\min} + 1))$ to be described by resolution $J_{\min}$. Since we require finer resolution closer to the agent we assume, of course, that $r(j-1) > r(j)$. The situation is depicted in Fig. 3.1.

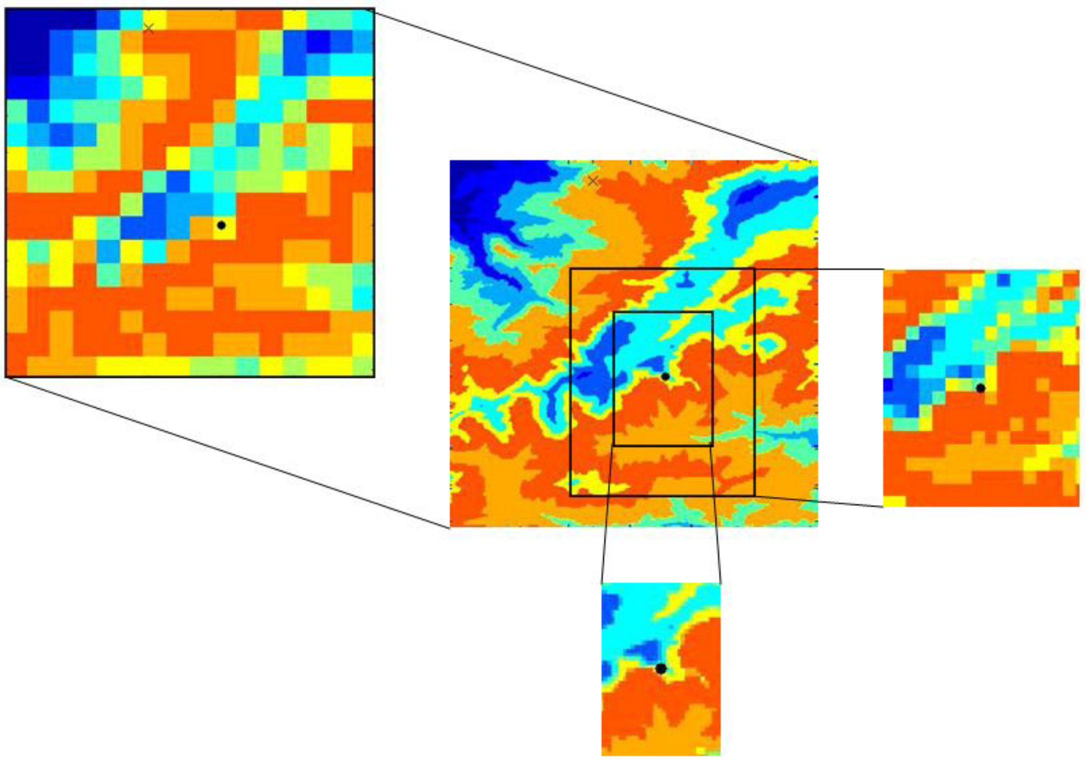


Figure 3.1: Multiresolution representation of the environment according to the distance from the current location of the agent.

The choice of $J_{\max}$ is dictated by the requirement that at this level all cells can be resolved into either free or occupied cells. The choice of $J_{\min}$ as well as the values of $r(j)$ are typically dictated by the on-board computational resources.

We obtain the distinct resolution levels at the given required distances from the current location of the agent by applying the Haar wavelet transform to rm. The use of Haar wavelets is mainly dictated by the choice of the norm in (4.5). To this end, let $\mathcal{I}(j) \triangleq \{0, 1, \ldots, 2^j - 1\}$ and let

$$\mathcal{K}(j) \triangleq \{k \in \mathcal{I}(j) : I_{j,k} \cap [x_0 - r(j), x_0 + r(j)] \neq \varnothing\},$$
$$\mathcal{L}(j) \triangleq \{k \in \mathcal{I}(j) : I_{j,\ell} \cap [y_0 - r(j), y_0 + r(j)] \neq \varnothing\}.$$

The wavelet decomposition of rm, given by

$$\begin{aligned} \mathsf{rm}(x,y) = & \sum_{k,\ell \in \mathcal{I}(J_{\min})} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(x,y) \\ & + \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\substack{k \in \mathcal{K}(j) \\ \ell \in \mathcal{L}(j)}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x,y) \end{aligned} \tag{3.4}$$

induces, via a slight abuse of notation, the following cell decomposition on $\mathcal{W}$

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}}. \tag{3.5}$$

where, ΔC_d^j is a union of cells $c^j_{k,\ell}$ of dimension $1/2^j \times 1/2^j$.

3.3 Cost Assignment

Each cell $c^j_{k,\ell}$ in the cell decomposition has a value $\mathsf{val}(c^j_{k,\ell}) \in \{M_1, \ldots, M_N\}$, which for the case of Haar wavelets is the weighted average of the risk measure function over the cell. Following the hierarchical approach, we divide the cells in three categories: free cells if $\mathsf{val}(c^j_{k,\ell}) < m_1$, full cells if $\mathsf{val}(c^j_{k,\ell}) > m_2$, and mixed cells if $m_1 \leq \mathsf{val}(c^j_{k,\ell}) \leq m_2$, where $m_1, m_2 \in \{M_1, \ldots, M_N\}$ are given.

To the cell decomposition $\mathcal{G}$ having as nodes $V(\mathcal{G})$ all the cells with $\mathsf{val}(c^j_{k,\ell}) \leq m_2$. The edges $E(\mathcal{G})$ of $\mathcal{G}$ correspond to the adjacency relationships of $V(\mathcal{G})$, as usual. Clearly, there is an one-to-one correspondence between the elements of $V(\mathcal{G})$ and the free and mixed cells of $\mathcal{C}_d$. We write $v \sim c^j_{k,\ell}$ to denote this correspondence. Moreover, since $\mathcal{G}$ is a topological graph we may associate each node $v \in V(\mathcal{G})$ with any point $\mathsf{x} \in c^j_{k,\ell}$. Without loss of generality we choose the center of the cell. Let $\mathsf{cell}_{\mathcal{G}}(v)$ denote the center of the corresponding cell in this case. Finally, if $\mathsf{x} \in c^j_{k,\ell}$ we will write $v = \mathsf{node}_{\mathcal{G}}(\mathsf{x})$ where $v \sim c^j_{k,\ell}$.

To each edge $(u, v) \in E(\mathcal{G})$ we assign a cost $\mathcal{J}(u, v)$, which is the cost of transitioning from node u to node v. We may use the transition cost as follows

$$\mathcal{J}(u, v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)). \tag{3.6}$$

That is, the cost of transitioning from node u to the adjacent node v depends only on the risk measure of the final node, v and is independent of the starting node u. This situation is depicted in Fig. 3.2. Note that, in general, $\mathcal{J}(u, v) \neq \mathcal{J}(v, u)$.

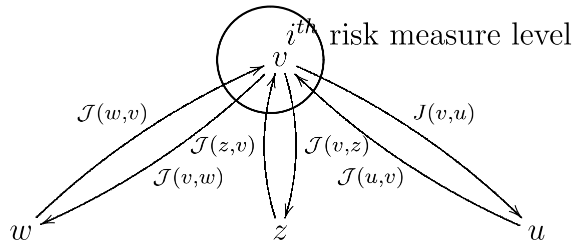


Figure 3.2: Cost assignment for each node u of the directed graph $\mathcal{G}$.

Another alternative would be to choose the transition cost so as to also penalize the (euclidean) distance between $\mathsf{cell}_{\mathcal{G}}(u)$ and $\mathsf{cell}_{\mathcal{G}}(v)$. In this case the cost becomes

$$\mathcal{J}(u, v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \alpha \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2. \tag{3.7}$$

where $\alpha \geq 0$ is a weight constant. The larger the α the more emphasis we place on a shorter path.

Suppose now that we are given a path of q consecutive, adjacent nodes in $\mathcal{G}$ as follows $\mathcal{P} = (v_0, v_1, \ldots, v_q)$. We can then assign a cost to each node in the path $\mathcal{P}$, induced by the two-node transitioning cost, iteratively, via

$$\mathcal{H}(v_i) = \mathcal{H}(v_{i-1}) + \mathcal{J}(v_{i-1}, v_i), \quad i = 1, \ldots, q. \tag{3.8}$$

The value of $\mathcal{H}(v_k)$ represents the (accumulated) cost of the path from v_0 to v_k ($k \leq q$). The shortest path problem is then to find a path that minimizes the accumulated cost from the initial to the destination node, or determine that such a path does not exist.

3.4 Multiresolution Path planning

The proposed multiresolution path planning algorithm proceeds as follows. Starting from $\mathsf{x}(t_0) = \mathsf{x}_0$ at time $t = t_0$, we construct using the approach of Chapter 2, a cell decomposition $\mathcal{C}_d(t_0)$ of $\mathcal{W}$. Let the corresponding graph be $\mathcal{G}(t_0)$ and let $v_1^0 \in \mathcal{G}(t_0)$ and $v_f^0 \in \mathcal{G}(t_0)$ be the initial and final nodes, respectively such that $v_1^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_0)$ and $v_f^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_f)$. Using Dijkstra's algorithm (or any other similar algorithm) we find a path $\mathcal{P}(t_0)$ in $\mathcal{G}(t_0)$ of free and mixed nodes from v_1^0 to v_f^0 assuming that such a path exists. Note that areas far away from the agent's current position are likely to be represented by cells of relative high size.Let the path $\mathcal{P}(t_0)$ be given by the ordered sequence of l_0 nodes as follows

$$\mathcal{P}(t_0) = (v_1^0, v_2^0, \cdots, v_{l_0-1}^0, v_{l_0}^0 = v_f^0).$$

It is assumed that v_2^0 is free owing to the high resolution decomposition of $\mathcal{W}$ close to x_0. The agent subsequently moves from v_1^0 to v_2^0. Let now t_1 be the time the agent is at the location $\mathsf{x}(t_1) = \mathsf{cell}_{\mathcal{G}(t_0)}(v_2^0)$ and let $\mathcal{C}_d(t_1)$ be the multiresolution cell decomposition of $\mathcal{W}$ around $\mathsf{x}(t_1)$ with corresponding topological graph $\mathcal{G}(t_1)$. Applying again Dijkstra's algorithm we find a (perhaps new) path in $\mathcal{G}(t_1)$ from $v_1^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1))$ to $v_f^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}_f)$ if such a path exists. Let $\mathcal{P}(t_1)$ be given by the ordered sequence of l_1 nodes as follows

$$\mathcal{P}(t_0) = (v_1^1, v_2^1, \cdots, v_{l_1-1}^1, v_{l_1}^1 = v_f^1).$$

The agent subsequently moves to $\mathsf{x}(t_2) = \mathsf{cell}_{\mathcal{G}(t_1)}(v_2^1)$ at time t_2.

In general, assume the agent is at location $\mathsf{x}(t_i)$ at time t_i. We construct a multiresolution decomposition $\mathcal{C}_d(t_i)$ of $\mathcal{W}$ around $\mathsf{x}(t_i)$ with corresponding graph $\mathcal{G}(t_i)$. Dijkstra's algorithm yields a path $\mathcal{P}(t_i)$ in $\mathcal{G}(t_i)$ of mixed and free noes of length l_i,

$$\mathcal{P}(t_i) = (v_1^i, v_2^i, \cdots, v_{l_i-1}^i, v_{l_i}^i = v_f^i),$$

where $v_1^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}(t_i))$ and $v_f^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}_f)$ if such a path exists. The process is continued until some time t_f when $\|\mathsf{x}(t_f) - \mathsf{x}_f\| < 1/2^{J_{\max}}$, at which time the algorithm terminates. At the last step the agent moves from $\mathsf{x}(t_f)$ to x_f.

Note that the actual path $\mathsf{x}(t_0), \mathsf{x}(t_1), \ldots, \mathsf{x}(t_f)$ followed by the agent is given by the sequence of nodes $\mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}(t_0)), \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1)), \ldots, \mathsf{node}_{\mathcal{G}(t_f)}(\mathsf{x}(t_f))$. Since the connectivity graph $\mathcal{G}(t)$ changes at each time step, it is therefore possible that the same state x may be visited twice since it may correspond to nodes of two distinct graphs. That is, it is possible that

$$\mathsf{cell}_{\mathcal{G}(t_i)}(v_k^i) = \mathsf{cell}_{\mathcal{G}(t_j)}(v_m^j), \quad i \neq j. \tag{3.9}$$

This will cause the agent to repeat the previous (optimal) decision ending up in a continuous loop. In order to avoid such pathological situations, we maintain a list $\mathsf{L}_{\mathsf{Visited}} = \{\mathsf{x}(t_0), \mathsf{x}(t_1), \ldots, \mathsf{x}(t_i)\}$ of all visited states up to the current time step t_i. At the next time step t_{i+1} we remove from $V(\mathcal{G}(t_{i+1}))$ all nodes v such that

$$\mathsf{cell}_{\mathcal{G}(t_{i+1})}(v) \in \mathsf{L}_{\mathsf{Visited}}. \tag{3.10}$$

A pseudo-code implementation of the above algorithm is given in Fig. 3.3.

```
BEGIN PATH PLANNING ALGORITHM
{
    $i = 0$;
    $\mathsf{x}_i \longleftarrow \mathsf{x}_0$;
    $\{\mathsf{L}_{\mathsf{Visited}}\}$ = $\{\varnothing\}$;
    (while $\|\mathsf{x}_f - \mathsf{x}_i\| > \epsilon$)
    {
      compute $\mathsf{rm}(\mathsf{x}, i)$ for all $\mathsf{x} \in \mathcal{W}$;
      construct $\mathcal{C}_d(i)$;
      construct $\mathcal{G}(i) = (E(i), V(i))$;
      (if $\mathsf{L}_{\mathsf{Visited}}$ is nonempty)
        for $v \in V(i)$
          $\mathsf{x}_v(i) = \mathsf{cell}_{\mathcal{G}(i)}(v)$;
          if $\mathsf{x}_v(i) \in \mathsf{L}_{\mathsf{Visited}}$
            extract $v$ from $V(i)$;
            for all $u$ adjacent to $v$
              remove $(v, u)$ from $E(i)$;
          end if;
      end if;
      $v_1^i \longleftarrow \mathsf{node}_{\mathcal{G}(i)}(\mathsf{x}_0)$;
      $v_f^i \longleftarrow \mathsf{node}_{\mathcal{G}(i)}(\mathsf{x}_f)$;
      $\mathcal{P}(i) \longleftarrow$ Dijkstra$(v_1^i, v_f^i, V(i), E(i))$;
      if $\mathcal{P}(i) = \{\varnothing\}$
        report FAILURE;
        break;
      $\mathsf{x}_i(1) = \mathsf{cell}_{\mathcal{G}(i)}(v_1^i)$;
      $\mathsf{x}_i(2) = \mathsf{cell}_{\mathcal{G}(i)}(v_2^i)$;
      $\{\mathsf{L}_{\mathsf{Visited}}\} \longleftarrow \{\mathsf{L}_{\mathsf{Visited}}\} \bigoplus \{\mathsf{x}_i(1), \mathsf{x}_i(2)\}$;
      $\mathsf{x}_{i+1} \longleftarrow \mathsf{x}_i(2)$;
      $i \longleftarrow (i + 1)$;
      $\mathsf{x}_0 \longleftarrow \mathsf{x}_i(2)$;
    }
}
END PATH PLANNING ALGORITHM
```

Figure 3.3: Pseudo-code implementation of proposed multiresolution path planning scheme.

Chapter 4

Multiresolution Path Planning Using Sector Decompositions

4.1 Introduction

In path-planning problems information about the environment is obtained using either on-board or off-board sensors. Some of this information is provided off-line and some is gathered on-line. Furthermore, most typical sensor devices provide sector-like representations of the environment (see Fig. 4.1). This type of information is not in the most efficient form for the majority of planning algorithms, which employ rectangle or square cell approximations, typically using quadtrees [19, 17, 29]. Such approximations are not compatible to the sector based representations obtained by most sensor devices.

In this chapter we present a planning algorithm as an extension of the results of the previous chapter. In the proposed path planning scheme we employ a conformal mapping to devise a hybrid local/global path planning algorithm using sector cell decompositions instead of decompositions that employ only rectangular or square cells. Sector cells are compatible to the on-board sensors and thus process the data more efficiently, in a manner that does not contradict its original sector-based form. We provide approximations with special localized attributes by combining efficiently data from sensors of different resolutions and ranges. Furthermore, in the algorithm of Chapter 3 the whole environment was assumed to be known *á priori* and the wavelet approximation scheme allowed us to plan the path

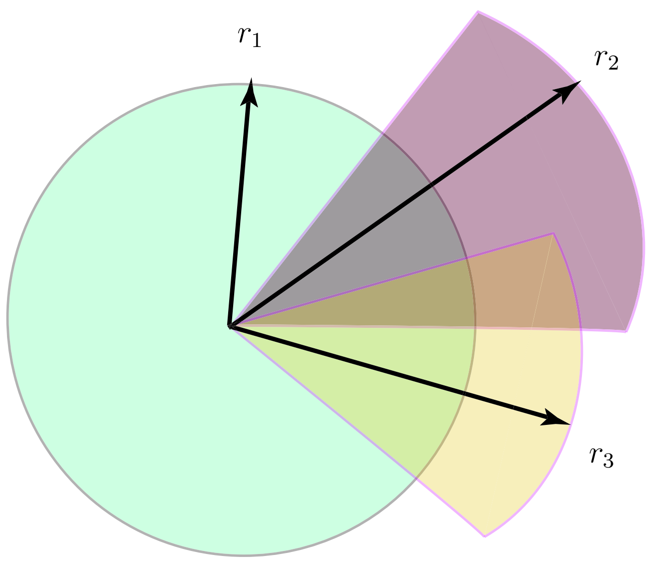


Figure 4.1: Sensors have different ranges, fields of view and resolution. Ideally, the algorithm that processes this data should conform to this topology.

using only a small fraction of the available information. This results in a reduced number of computations that can be handled by the available computational resources on-board the vehicle. Contrary to the planning algorithm of Chapter 3, the proposed methodology in this chapter employs on-line data of the agent's immediate environment as it is obtained by the on-board sensors. This approach is the most natural way to deal with the navigation problem of an autonomous vehicle operating in an unexplored environment, for which no prior knowledge is available.

In the next section we introduce a methodology for obtaining sector cell decompositions given rectangular cell decompositions using conformal mapping. In Section 4.3 we describe the way we represent hierarchically the new world space $\mathcal{W}$ in the new coordinate system induced by the conformal mapping. The new multiresolution decomposition scheme of the risk measure in the new coordinate system is described in Section 4.4. Finally, in Section 4.5 we present in detail the new sector-based multiresolution path planning algorithm.

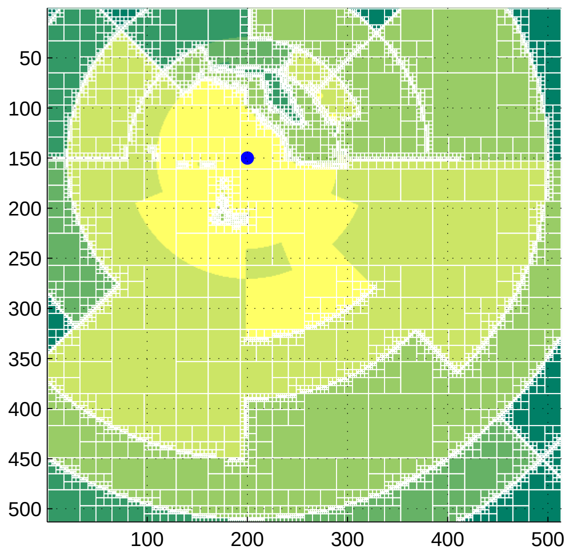


Figure 4.2: A cell decomposition based on the available sector approximation of the environment obtained by the on-board sensor devices of the agent (denoted with the blue dot). In order to resolve the geometry of the arc-boundary of each sector the standard quadtree algorithm generates a large number of cells at close to the boundaries of these arcs.

4.2 From Rectangular to Sector Cell Decompositions

It is assumed that the available information of the surrounding area of the agent is given by the superposition of circular or conical sectors obtained by different sensor devices as depicted in Fig. 4.1. This information about $\mathcal{W}$ needs to be processed by the path planner to compute a collision free path. In order to do so, the path-planning algorithm typically computes a cell decomposition of the environment. As can be easily observed by Fig. 4.2, if rectangular cells are used (as with any quadtree-based approach) the algorithm may waste computational resources by subdividing the cells in order to resolve the sector boundaries.

A simple way to overcome this difficulty is to employ a conformal mapping to map the sector cells to rectangular cells in a new coordinate system. The latter approach is proposed in this chapter. The motivation for this idea is simple. Let

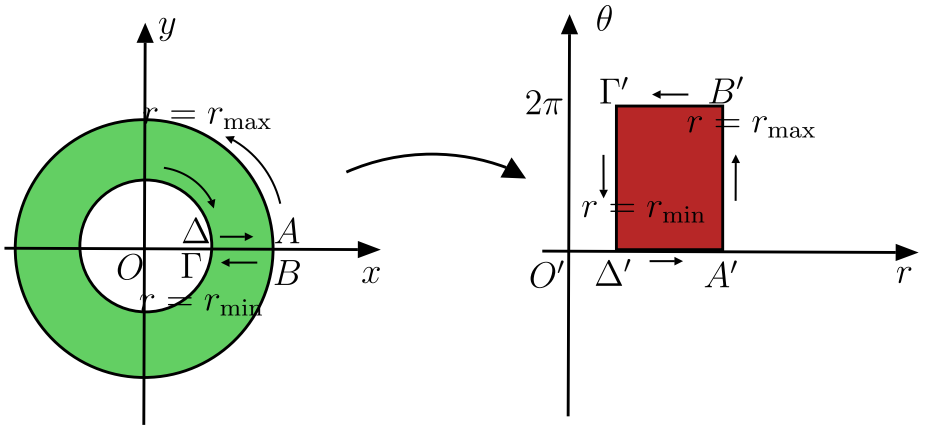


Figure 4.3: A cut annulus in the (x, y) plane is mapped to a rectangle in the (r, θ) plane using a polar (conformal) mapping.

us assume that $\mathcal{R}$ is a sector domain in the (x, y) plane specified by the radii $r_{\min}$ and $r_{\max}$ and the angles $\theta_{\min}$ and $\theta_{\max}$. Mapping this domain to the (r, θ) plane using the polar transformation one obtains a rectangular domain $\mathcal{R}'$ defined by the same radii and angles. If the angle varies from $\theta_{\min} = 0$ up to $\theta_{\max} = 2\pi$ the whole annulus cut defined by the radii $r_{\min}$ and $r_{\max}$ is mapped to a rectangular cell in the (r, θ) plane as shown in Fig. 4.3.

More precisely, recall that the polar coordinates r, θ are related to x and y via the equations

$$x = r\cos\theta, \qquad y = r\sin\theta, \tag{4.1}$$

where the *Jacobian* of the transformation is given by

$$J = \frac{\partial(x, y)}{\partial(r, \theta)} = r,$$

and thus $J > 0$ provided that $r > 0$. Thus, the inverse transformation $(x, y) \mapsto (r, \theta)$ exists for $r > 0$.

The closed curve defined by the union of the circles $C_1 = \{(x, y) \in \mathbb{R}^2,\ x^2 + y^2 = r_{\min}^2\}$, $C_2 = \{(x, y) \in \mathbb{R}^2,\ x^2 + y^2 = r_{\max}^2\}$ and the line segment $L = \{(x, y) \in \mathbb{R},\ r_{\min} \leq x \leq r_{\max}\}$ (traversed twice), maps to a rectangle $r = r_{\min}, r = r_{\max}, \theta = 0, \theta = 2\pi$ in the (r, θ) plane. Alternatively, the area inside a rectangle defined by $r = r_{\min}, r = r_{\max}, \theta = \theta_{\min}, \theta = \theta_{\max}$ where $0 \leq \theta_{\min} \leq \theta_{\max} \leq 2\pi$ is mapped via the inverse transformation to the intersection of two sectors defined by $r_{\min}, \theta_{\min}$ and $r_{\max}, \theta_{\max}$ respectively in the (x, y) plane.

Thus, given a sector cell decomposition in the (x, y) plane, a rectangle cell decomposition can be obtained in the (r, θ) plane via the polar coordinate transformation (4.1). Conversely, if we have a multiresolution approximation of the rectangular domain in the (r, θ) plane, then by applying the inverse mapping, the rectangular area can be approximated by a collection of cells forming a sector domain in the (x, y) plane, as seen in Fig. 4.4.

4.3 World Space in the New Coordinate System

Let f be a function $f : \mathfrak{Domain}(f) = \mathcal{W} \mapsto \mathbb{R}$ and let $\mathfrak{Image}_f(\mathcal{W}) = \{f(x) : x \in \mathcal{W}\} \subseteq \mathbb{R}$. In order to map the set $\mathcal{W} \times \mathfrak{Image}_f(\mathcal{W})$ to the polar coordinate system we proceed as follows.

First, we discretize $\mathcal{W}$ using a uniform grid of dimension $2^N \times 2^N$. For each point (x_i, y_i), $(1 \leq i \leq 2^N)$ we compute the corresponding (r_i, θ_i) point in polar form using the following equations:

$$\begin{aligned} r_i &= \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}, \\ \theta_i &= \operatorname{atan2}(y_i - y_0, x_i - x_0), \end{aligned} \tag{4.2}$$

where (x_0, y_0) is the agent's current position and (r_i, θ_i) is the distance and angle position with respect to the agent. Let $\phi_{(x_0,y_0)} : \mathbb{R}^2 \mapsto \mathbb{R} \times S^1$. We can then associate to each pair (r_i, θ_i) the function value $f(x_i, y_i)$, that is, $f'(r_i, \theta_i) = f(\phi_{(x_0,y_0)})(x_i, y_i)) = f(x_i, y_i)$.

The next step is to obtain a multiresolution (rectangular) cell approximation of the function f' in $\mathcal{W}'$. As we shall see in the next section, the wavelet transform provides us with such a multiresolution cell based approximation in any rectangular domain. Based on this cell decomposition we can proceed with the design of our path planning algorithm working entirely in the $\mathcal{W}'$ domain by applying the wavelet-based path planning algorithm of Chapter 3.

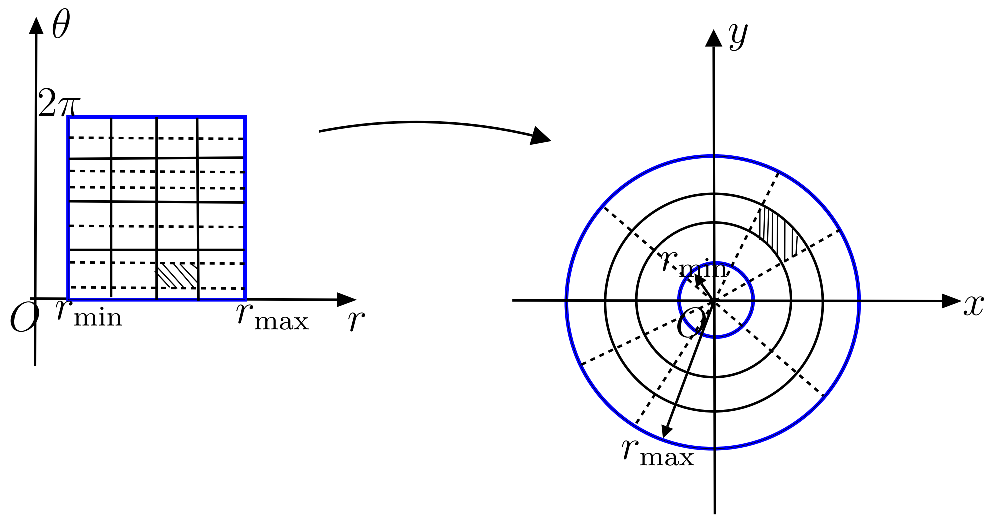


Figure 4.4: A multiresolution approximation of the rectangular domain in (r, θ) system defined by the radii $r_{\min}$ and $r_{\max}$ under the inverse conformal mapping gives a multiresolution sector approximation of an annulus cut defined by the same radii.

4.4 A Multiresolution Decomposition Scheme of the Risk Measure

Without loss of generality, in the work we take $\mathcal{W} = [0,1] \times [0,1]$. Using the conformal mapping of section 4.2, $\mathcal{W}$ is mapped to $\mathcal{D}'$ in the polar coordinate system. The new domain however, is not rectangular. Assuming without loss of generality (renormalize $\mathcal{W}$, if necessary) that $\mathcal{D}' \subset [0,1] \times [0,1]$, we let $[0,1] \times [0,1] \backslash \mathcal{D}'$ lie completely inside the obstacle configuration space $\mathcal{O}'$ in the (r, θ) domain. Thus, by adding this artificial obstacle corresponding to the boundary of $\mathcal{D}'$ we can assume that the world $\mathcal{W}'$ in the polar coordinate system to be again $\mathcal{W}' = [0,1] \times [0,1]$.

We describe $\mathcal{W}'$ using a discrete (fine) grid of $2^N \times 2^N$ dyadic points. The finest level of resolution $J_{\max}$ is therefore bounded by N. It follows from the previous discussion that the Haar wavelet decomposition of a function f' defined over $\mathcal{W}'$

at resolution level $J \geq J_{\min}$, given by,

$$\begin{aligned} f'(r,\theta) = &\sum_{k,\ell=0}^{2^{J_{\min}}-1} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(r,\theta) \\ &+ \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J-1} \sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(r,\theta) \end{aligned} \tag{4.3}$$

induces a cell decomposition of $\mathcal{W}'$ of square cells of size $1/2^J \times 1/2^J$ in the (r,θ) coordinate system.

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto [0,1]$ that represents the "risk measure" at the location $\mathsf{x} = (x,y)$. For instance, one may choose

$$\mathsf{rm}(\mathsf{x}) = \begin{cases} (d_{\max} - \min_{\mathsf{y}\in\mathcal{O}} \|\mathsf{x}-\mathsf{y}\|_2)/d_{\max}, & \text{if } \mathsf{x} \in \mathcal{F}, \\ 1, & \text{if } \mathsf{x} \in \mathcal{O}, \end{cases} \tag{4.4}$$

where $d_{\max} \overset{\triangle}{=} \max_{\mathsf{x}\in\mathcal{F}} \min_{\mathsf{y}\in\mathcal{O}} \|\mathsf{x}-\mathsf{y}\|_2$. Alternatively, one may think of rm as the probability that $(x,y) \in \mathcal{O}$. The function rm can be defined over $\mathcal{W}'$, i.e $\mathsf{rm}' : \mathcal{W}' \mapsto [0,1]$.

All points within range ρ from the current location of the agent, when expressed in the (r,θ) system, are given by

$$\mathcal{N}(\rho) \overset{\triangle}{=} \{(r,\theta) \in \mathcal{W}' : |r| \leq \rho,\ \theta \in [-\theta_{\min}, \theta_{\max}]\}. \tag{4.5}$$

The region that corresponds to $\mathcal{N}(\rho)$ in the (r,θ) coordinate system is a strip of width equal to ρ and height $\theta_{\max} - \theta_{\min}$. For simplicity, in this work we will assume that $\theta_{\min} = -\pi$ and $\theta_{\max} = \pi$.

Suppose now that we are given the desired levels of resolution of $\mathcal{W}'$ as $J_{\min} \leq j \leq J_{\max}$ where $J_{\min}, J_{\max} \in \{1,\ldots,N\}$, with corresponding ranges ρ_j from the agent's current location. By this we mean that we wish all points $(r,\theta) \in \mathcal{N}(\rho_{J_{\max}})$ to be described by resolution $J_{\max}$, all points $(r,\theta) \in \mathcal{N}(\rho_{j-1}) \backslash \mathcal{N}(\rho_j)$ to be described by resolution j, where $J_{\min} < j \leq J_{\max}$, and all points $(r,\theta) \notin \mathcal{N}(\rho_{J_{\min}+1})$ to be described by resolution $J_{\min}$. Since we require finer resolution closer to the agent we assume, of course, that $\rho_{j-1} > \rho_j$.

The choice of $J_{\max}$ is dictated by the requirement that at this level all cells should be resolved into either free or occupied cells. The choice of $J_{\min}$ as well as the values of ρ_j are typically dictated by the sensor specifications and/or the on-board computational resources.

We obtain the distinct resolution levels at the given required distances from the current location of the agent by applying the Haar wavelet transform to rm'. To this end, let $\mathcal{I}(j) \triangleq \{0, 1, \ldots, 2^j - 1\}$ and let

$$\begin{aligned}
\mathcal{K}(j) &\triangleq \{k \in \mathcal{I}(j) : I_{j,k} \cap [0, \rho_j] \neq \varnothing\}, \\
\mathcal{L}(j) &\triangleq \{k \in \mathcal{I}(j) : I_{j,\ell} \cap [\theta_{\min}, \theta_{\max}] \neq \varnothing\}.
\end{aligned}$$

The wavelet decomposition of rm', given by

$$\begin{aligned}
\mathsf{rm}'(r,\theta) = &\sum_{k,\ell \in \mathcal{I}(J_{\min})} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(r,\theta) \\
&+ \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\substack{k \in \mathcal{K}(j) \\ \ell \in \mathcal{L}(j)}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(r,\theta)
\end{aligned} \tag{4.6}$$

induces, via a slight abuse of notation, the following cell decomposition on $\mathcal{W}'$

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}} \tag{4.7}$$

where, ΔC_d^j is a union of square cells $c^j_{k,\ell}$ in the (r,θ) domain of dimension $1/2^j \times 1/2^j$. Furthermore, by applying the inverse mapping based on the analysis of section 4.2 we obtain a sector decomposition $\mathcal{S}_d$ of $\mathcal{W}$, where $\mathcal{S}_d$ is defined as

$$\mathcal{S}_d = \Delta S_d^{J_{\min}} \oplus \cdots \oplus \Delta S_d^{J_{\max}} \tag{4.8}$$

where, ΔS_d^J is the image under the inverse conformal mapping of ΔC_d^J, with $J_{\min} \leq J \leq J_{\max}$.

4.5 Sector-based Multiresolution Path Planning

The proposed multiresolution path planning algorithm takes place completely in the (r,θ) coordinate system. The agent in this system remains at the origin at all times. Let $\mathsf{x}' = (r,\theta)$. Assuming therefore that $\mathsf{x}' = (0,0)$ at $t = t_0$, we construct using the approach of Section 4.4, a cell decomposition $\mathcal{C}_d(t_0)$ of $\mathcal{W}'$. Denote by $\mathcal{G}(t_0)$ be the corresponding connectivity graph. Using Dijkstra's algorithm (or any other similar algorithm) we then find a *tentative* path $\mathcal{P}(t_0)$ in $\mathcal{G}(t_0)$ of free and mixed nodes which connects the initial node to the final node in $\mathcal{G}(t_0)$. The path $\mathcal{P}(t_0)$ is therefore an ordered sequence of nodes

$$\mathcal{P}(t_0) = (v_1^1, v_2^1, \cdots, v_{\ell_1 - 1}^1, v_{\ell_1}^1 = v_f^1). \tag{4.9}$$

where $v_i^1 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_i')$ and $\mathsf{x}_i' = \mathsf{cell}_{\mathcal{G}(t_0)}(v_i^1)$ is a representative, arbitrary point of the cell that corresponds to node v_i^1. For simplicity, we will assume that x_i' is the center of the cell.

The first cell in the sequence (4.9) is the cell that contains the origin and the final cell in the sequence is the cell that contains the final destination (r_f, θ_f) provided that such a sequence exists. Since we assume high resolution inside $\mathcal{N}(\rho_{J_{\max}})$ it is natural to assume that the first two cells in $\mathcal{C}_d(t_0)$ corresponding to v_1^1 and v_2^1 are free for a feasible path. The agent subsequently moves from $\mathsf{cell}_{\mathcal{G}(t_0)}(v_1^1)$ to $\mathsf{cell}_{\mathcal{G}(t_0)}(v_2^1)$. This means that only the points x_1' and x_2' will actually be visited by the agent. In order to find the subsequent points of the actual path to be followed we apply a continuous replanning scheme. This is accomplished by constructing a cell decomposition $\mathcal{C}_d(t_k)$ at each next time step $t_k > t_0$, $(k = 1, 2, \ldots)$, and by inserting in the list of the visiting points only the point $\mathsf{x}_k' = \mathsf{cell}_{\mathcal{G}(t_k)}(v_2^k)$ which corresponds to the second node of the tentative path $\mathcal{P}(t_k)$. We repeat the process until the goal is inside the second cell of $\mathcal{P}(t_k)$.

Since the approach does not eliminate the possibility that the same point may be revisited during the subsequent replanning, thus resulting in an endless loop, we overcome this issue by keeping a list of all points already visited and by removing the corresponding nodes form the graph $\mathcal{G}(t_k)$ at each time step t_k. A pseudo-code implementation of the above algorithm is given in Fig. 4.5.

```
BEGIN PATH PLANNING ALGORITHM
{
    $\mathsf{x}'_0 = 0$;
    $i = 0$;
    $\mathsf{x}_i \longleftarrow \mathsf{x}_0$;
    $\{\mathsf{L}_{\mathsf{Visited}}\}$ = $\{\varnothing\}$;
    (while $\|\mathsf{x}_f - \mathsf{x}_i\| > \epsilon$)
    {
      compute $\mathsf{rm}(\mathsf{x}, i)$ for all $\mathsf{x} \in \mathcal{W}$;
      compute $\mathsf{rm}(\mathsf{x}', i)$ for all $\mathsf{x}' \in \mathcal{W}'$ via conformal mapping;
      construct $\mathcal{C}_d(i)$ on $\mathcal{W}'$ topology ;
      construct $\mathcal{G}(i) = (E(i), V(i))$;
      (if $\mathsf{L}_{\mathsf{Visited}}$ is nonempty)
        for $v \in V(i)$
          $\mathsf{x}'_v(i) = \mathsf{cell}_{\mathcal{G}(i)}(v)$;
          if $\mathsf{x}_v(i) \in \mathsf{L}_{\mathsf{Visited}}$
            extract $v$ from $V(i)$;
            for all $u$ adjacent to $v$
              remove $(v, u)$ from $E(i)$;
          end if;
      end if;
      $v_1^i \longleftarrow \mathsf{node}_{\mathcal{G}(i)}(\mathsf{x}'_0)$;
      $v_f^i \longleftarrow \mathsf{node}_{\mathcal{G}(i)}(\mathsf{x}'_f)$;
      $\mathcal{P}(i) \longleftarrow$ Dijkstra$(v_1^i, v_f^i, V(i), E(i))$;
      if $\mathcal{P}(i) = \{\varnothing\}$
        report FAILURE;
        break;
      $\mathsf{x}'_i(1) = \mathsf{cell}_{\mathcal{G}(i)}(v_1^i)$;
      $\mathsf{x}'_i(2) = \mathsf{cell}_{\mathcal{G}(i)}(v_2^i)$;
      $\{\mathsf{L}_{\mathsf{Visited}}\} \longleftarrow \{\mathsf{L}_{\mathsf{Visited}}\} \bigoplus \{\mathsf{x}'_i(1), \mathsf{x}'_i(2)\}$;
      $\mathsf{x}'_{i+1} \longleftarrow \mathsf{x}'_i(2)$;
      $i \longleftarrow (i + 1)$;
      compute $\mathsf{x}_{i+1}$ via inverse conformal mapping;
      $\mathsf{x}_0 \longleftarrow \mathsf{x}_{i+1}$;
    }
}
END PATH PLANNING ALGORITHM
```

Figure 4.5: Pseudo-code implementation of proposed multiresolution path planning scheme.

Chapter 5

Time Scheduling and Smooth Trajectory Generation

5.1 Introduction

As we have seen in Chapters 3 and 4 the output of the proposed *path planning scheme* at each time step t_j is a path $\mathcal{P}(t_j)$. This path is defined as a finite ordered sequence of visiting points x_i, where i belongs to some index set $\mathcal{I}(t_j)$. These points form at each time t_j a polygonal line which lies completely inside the free space $\mathcal{F}$. The engineering problem that subsequently appears is to derive control laws that result in this polygonal trajectory curve. The difficulty of this problem is due to the dynamic constraints imposed by the equations of motion of the agent. These constraints make the tracking of the polygonal ,and thus non smooth, trajectory a “mission impossible”. In this section we examine ways to generate smooth trajectories when the equations of motion of the agent are specified by a unicycle kinematic model (ground vehicle model). Our objective is that the generated trajectory passes sufficiently close to the x_i points at specified instants of time (time scheduling).

5.2 Unicycle Kinematic Model under Dynamic Extension

The unicycle kinematic model is given by the following equations:

$$\dot{x}(t) = v(t)\cos(\theta(t)) \tag{5.1}$$

$$\dot{y}(t) = v(t)\sin(\theta(t)) \tag{5.2}$$

$$\dot{\theta}(t) = \omega(t) \tag{5.3}$$

where x, y are the coordinates of the unicycle, θ is the orientation of the vehicle (velocity vector orientation) and v and ω are the control inputs of the system. Specifically, v is the speed of the vehicle defined by the following equation

$$v(t) = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}. \tag{5.4}$$

Furthermore, ω is the angular velocity of the agent due to the change of vehicle orientation. Since the output we wish to track is the position of the agent we choose the outputs of our system to be

$$h_1(t) = x(t) \tag{5.5}$$

$$h_2(t) = y(t) \tag{5.6}$$

The system can be written in the form $\dot{\mathsf{x}}(t) = f(\mathsf{x}) + g(\mathsf{x})u$ with

$$\mathsf{x}(t) = \begin{pmatrix} \mathsf{x}_1(t) \\ \mathsf{x}_2(t) \\ \mathsf{x}_3(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix},$$

$$f(\mathsf{x}) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad g(\mathsf{x}) = \begin{pmatrix} \cos(\mathsf{x}_3) & \sin(\mathsf{x}_3) & 0 \\ 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}, \quad u(t) = \begin{pmatrix} v \\ \omega \end{pmatrix},$$

where the output is written in the form

$$y(t) = h(\mathsf{x}(t))$$

where

$$h = \begin{pmatrix} h_1(t) \\ h_2(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}.$$

Our goal then is to find the admissible inputs v, ω which achieve asymptotic tracking of the system output to a desired reference signal

$$\tilde{h}_{\text{ref}} = \begin{pmatrix} x_{\text{ref}}(t) \\ y_{\text{ref}}(t) \end{pmatrix}.$$

Since the system in (5.1)-(5.3) is nonlinear, in order to achieve asymptotic tracking our first thought is to try to apply input/output feedback linearization (i.e. IOL, see [15]). Unfortunately, the relative degrees r_1, r_2 for the specific choice of inputs and outputs are found to be ill defined. To see this, let $\mathcal{L}_\phi z(x) = \frac{\partial z}{\partial x}\phi(x)$ denote the Lie derivative of z with respect to ϕ along ϕ (where ϕ and z are sufficiently smooth functions). Then, after routine calculations we have for the first output

$$\mathcal{L}_{\text{col}_1(g)} h_1(\mathsf{x}) = \cos(\mathsf{x}_3) \tag{5.7}$$

$$\mathcal{L}_{\text{col}_2(g)} h_1(\mathsf{x}) = 0 \tag{5.8}$$

and for the second one

$$\mathcal{L}_{\text{col}_1(g)} h_2(\mathsf{x}) = \sin(\mathsf{x}_3) \tag{5.9}$$

$$\mathcal{L}_{\text{col}_2(g)} h_2(\mathsf{x}) = 0 \tag{5.10}$$

where the notation $\text{col}_1(g)$ is used for the i^{th} column of g. Therefore, the decoupling matrix corresponding to (5.1)-(5.3) is given by

$$\alpha = \begin{pmatrix} \mathcal{L}_{\text{col}_1(g)} h_1 & \mathcal{L}_{\text{col}_2(g)} h_2 \\ \mathcal{L}_{\text{col}_1(g)} h_1 & \mathcal{L}_{\text{col}_2(g)} h_2 \end{pmatrix} = \begin{pmatrix} \cos(\mathsf{x}_3) & 0 \\ \sin(\mathsf{x}_3) & 0 \end{pmatrix}. \tag{5.11}$$

where the notation

From (5.11) we observe that the matrix α is always singular and thus we cannot find a region in $\mathbb{R}^3$ where both r_1 and r_2 are non zero. Therefore, application of output tracking via input/output feedback linearization (IOL) theory is impossible.

A proposed methodology in the literature (see [15]) to overcome this problem is to achieve well-defined relative degree via dynamic extension. The idea is to transform the system dynamics in such a way that all the Lie derivatives in the expressions (5.7) - (5.10) vanish. In that case, both r_1 and r_2 are at least equal to one. The most desirable case is to achieve $r_1 + r_2 = n$, where n is the dimension of the new system dynamics. In that case, the system is said to have trivial zero dynamics and therefore no internal instability phenomena appear. The term zero dynamics describes these modes of the system that do not allow the original dynamical system to be brought into a fully linear and controllable form via state feedback and coordinate transformation. If however, $2 \leq r_1 + r_2 < n$ then because the decoupling matrix is nonsingular our system can be separated into two subsystems in cascade form after the application of the feedback linearization theory. The first one is a controllable linear system, whereas the second subsystem is formed by the remaining zero dynamics which cannot be reached directly by the input. Explicit or implicit interconnections between these two subsystems are possible. Thus, in order to guarantee that the system does not exhibit any internal instability the remaining zero dynamics have to be stable in the sense of Lyapunov.

In our case, we proceed by considering the speed v to be an additional state of the system dynamics and

$$\tilde{v} = \frac{dv}{dt} \tag{5.12}$$

to be the new control input component in place of v, or in other words the speed v(new state) is the output of an integrator driven by the new control input $\tilde{v}$. Then, the system dynamics (5.1)-(5.3)become:

$$\dot{x}(t) = v(t)\cos(\theta(t)) \tag{5.13}$$

$$\dot{y}(t) = v(t)\sin(\theta(t)) \tag{5.14}$$

$$\dot{\theta}(t) = w(t) \tag{5.15}$$

$$\dot{v}(t) = \tilde{v}(t) \tag{5.16}$$

The system can be written in the form $\dot{\tilde{\mathsf{x}}}(t) = \tilde{f}(\tilde{\mathsf{x}}(t)) + \tilde{g}(\tilde{\mathsf{x}}(t))\tilde{u}(t)$ with

$$\tilde{\mathsf{x}}(t) = \begin{pmatrix} \tilde{\mathsf{x}}_1(t) \\ \tilde{\mathsf{x}}_2(t) \\ \tilde{\mathsf{x}}_3(t) \\ \tilde{\mathsf{x}}_4(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \\ v(t) \end{pmatrix},$$

$$\tilde{f}(\tilde{\mathsf{x}}(t)) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \tilde{g}(\tilde{\mathsf{x}}(t)) = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}^{\mathrm{T}}, \ \tilde{u}(t) = \begin{pmatrix} \tilde{v}(t) \\ w(t) \end{pmatrix},$$

where the output is as before written in the form

$$\tilde{y}(t) = \tilde{h}(\tilde{\mathsf{x}}(t))$$

where $\tilde{h} = \begin{pmatrix} \tilde{h}_1(t) \\ \tilde{h}_2(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$.

Repeating the Lie derivatives calculations for the dynamically extended system we get for the first output

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = 0 \tag{5.17}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = 0 \tag{5.18}$$

and for the second one

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = 0 \tag{5.19}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = 0 \tag{5.20}$$

so the relative degrees r_1 and r_2 are well defined and both of them greater than or equal to one. Therefore, we can continue the calculations for higher order Lie derivatives to obtain for the first output

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = \cos(\tilde{\mathsf{x}_3})) \tag{5.21}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = -\tilde{\mathsf{x}_4}\sin(\tilde{\mathsf{x}_3}) \tag{5.22}$$

and for the second one

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = \sin(\tilde{\mathsf{x}_3}) \tag{5.23}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = \tilde{\mathsf{x}_4}\cos(\tilde{\mathsf{x}_3}) \tag{5.24}$$

Therefore the corresponding decoupling matrix to the dynamically extended unicycle (5.13)-(5.16) is given by

$$\tilde{\alpha} = \begin{pmatrix} \mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1 & \mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2 \\ \mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1 & \mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2 \end{pmatrix} = \begin{pmatrix} \cos(\tilde{\mathsf{x}_3}) & -\tilde{\mathsf{x}_4}\sin(\tilde{\mathsf{x}_3}) \\ \sin(\tilde{\mathsf{x}_3}) & \tilde{\mathsf{x}_4}\cos(\tilde{\mathsf{x}_3}) \end{pmatrix}. \tag{5.25}$$

We observe that the matrix $\tilde{\alpha}(\tilde{\mathsf{x}})$ is non singular for all $\tilde{\mathsf{x}} \in \mathbb{R}^4$ with $\tilde{\mathsf{x}}_4 \neq 0$ and furthermore, the relative degrees are $\tilde{r}_1 = \tilde{r}_2 = 2$. Thus, we have $r_1 + r_2 = n = 4$ and the system is input/output linearizable with trivial zero dynamics. Notice that the condition $\tilde{\mathsf{x}}_4 \neq 0$ in our specific choice of the system outputs is a smoothness requirement for the curve that the agent has to follow.

Finally, the control input vector that achieves the output tracking, provided that $\tilde{\mathsf{x}}_4 \neq 0$, is given by

$$\tilde{u}(t) = \tilde{a}^{-1}(\tilde{\nu}(t) - b(\tilde{\mathsf{x}}(t))) \tag{5.26}$$

where

$$\tilde{b}(\tilde{\mathsf{x}}(t)) = \begin{pmatrix} \mathcal{L}^2_{\tilde{f}}\tilde{h}_1(\tilde{\mathsf{x}}(t)) \\ \mathcal{L}^2_{\tilde{f}}\tilde{h}_2(\tilde{\mathsf{x}}(t)) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

and

$$\tilde{\nu}(t) = \begin{pmatrix} \tilde{\nu}_1(t) \\ \tilde{\nu}_2(t) \end{pmatrix}$$

with

$$\tilde{\nu}_1(t) = -k_{11}(\tilde{h}_1(\tilde{\mathsf{x}}(t)) - x_{\text{ref}}(t)) - k_{12}(\mathcal{L}_{\tilde{f}}\tilde{h}_1(\tilde{\mathsf{x}}(t)) - \dot{x}_{\text{ref}}(t)) + \ddot{x}_{\text{ref}}(t) \tag{5.27}$$

$$\tilde{\nu}_2(t) = -k_{21}(\tilde{h}_2(\tilde{\mathsf{x}}(t)) - y_{\text{ref}}(t)) - k_{22}(\mathcal{L}_{\tilde{f}}\tilde{h}_2(\tilde{\mathsf{x}}(t)) - \dot{y}_{\text{ref}}(t)) + \ddot{y}_{\text{ref}}(t). \tag{5.28}$$

In the expressions (5.27)-(5.28) $k_{11}, k_{12}, k_{21}, k_{22}$ are constants chosen such that the binomials

$$P_1(s) = s^2 + k_{12}s + k_{11} \tag{5.29}$$

$$P_2(s) = s^2 + k_{21}s + k_{22} \tag{5.30}$$

$$\tag{5.31}$$

are Hurwitz.

5.3 Reference Signal Specification

As we have described in the previous section, the output of our system is the (x, y) coordinates of the agent's position. Additionally, the path $\mathcal{P}$ of the *path planning scheme* we have presented so far is comprised of a finite number of visiting points X_i in order to reach the goal destination. Therefore, the desired system output should be a curve $\vec{r}(\xi) = (x_{\mathrm{ref}}(\xi), y_{\mathrm{ref}}(\xi))$, where ξ is the curve parameter, which passes through or arbitrary close to those points. Additionally, we know that the space $\mathcal{W}$ is a polygonally connected space and thus the polygonal line that connects all the points X_i in $\mathcal{P}$ lies completely in $\mathcal{W}$. Based on this observation, the curve $\vec{r}(\xi)$ we select as the desired path of the vehicle should be a smooth curve which at the same time remains as close as possible to the original polygonal line. The smoothness requirement of the curve is imposed by the tracking scheme we are employing. Furthermore, the requirement to stay close to the polygonal lines comes from the collision avoidance constraint which requires the generated trajectory to lie at all times inside $\mathcal{W}$. The most natural way to proceed is to try to approximate the polygonal line (or part of it as we shall see later) using either a polynomial or a spline interpolation scheme. The degree of the polynomial used and the number of points lying on the polygonal line for the interpolation is correlated to the closeness of the the reference curve to the polygonal line we wish to approximate.

Designing our reference curve to be close to the polygonal line may be an overconservative approach since a trajectory which lies inside the free world $\mathcal{W}$ with more plausible characteristics (e.g. a curve with less sharp corners) may exist. However, the generation of a collision free trajectory which is at the same time compatible with the system dynamics and the admissible inputs in the framework of the path planning problems we are dealing with in this work is still an open problem. Here we tacitly assume that there exist admissible inputs v and w such that the true system response under those inputs is the reference curve

$$\vec{r}(\xi) = (x_{\mathrm{ref}}(\xi), y_{\mathrm{ref}}(\xi)).$$

In order to proceed, we shall employ some basic theory of differential geometry for curves (see [30]). The velocity of the agent is given by

$$\vec{V}(\xi) = \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} = \left\| \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} \right\| \vec{\mathbb{T}}(\xi) \tag{5.32}$$

where

$$\vec{\mathbb{T}}(\xi) = \frac{1}{\left\| \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} \right\|} \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi}$$

is the unit tangent vector of the curve $\vec{r}(\xi)$.

The speed of the agent is the norm of the velocity, i.e.

$$v(\xi) = \left\| \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} \right\| = \sqrt{\left(\frac{\mathrm{d}x(\xi)}{\mathrm{d}\xi}\right)^2 + \left(\frac{\mathrm{d}y(\xi)}{\mathrm{d}\xi}\right)^2}. \tag{5.33}$$

We can alternatively express the speed as $v = \frac{\mathrm{d}s}{\mathrm{d}\xi}$ where s is the arc length of the curve between the points $\vec{r}(\xi)$ and $\vec{r}(\xi_0)$ which is given by

$$s(\xi) = \int_{\xi_0}^{\xi} \left\| \frac{\mathrm{d}\vec{r}(z)}{\mathrm{d}z} \right\| \mathrm{d}z. \tag{5.34}$$

At this point we observe that the speed v, which appears either as one of the inputs in the original unicycle equations (5.1)-(5.3) or as one of the states in the dynamically extended model equations (5.13)-(5.16), is solely specified by the $\dot{x}, \dot{y}$ quantities. Thus, given a perfect tracking capacity scheme as the one we employ, the speed is prescribed by the reference outputs $x_{\mathrm{ref}}, y_{\mathrm{ref}}$ (or more precisely by the time derivatives of the them). In other words, after any transient phenomena have disappeared from the system response, we expect the speed to be

$$v(\xi) = \sqrt{\left(\frac{\mathrm{d}x_{\mathrm{ref}}(\xi)}{\mathrm{d}\xi}\right)^2 + \left(\frac{\mathrm{d}y_{\mathrm{ref}}(\xi)}{\mathrm{d}\xi}\right)^2}.$$

Therefore, the speed depends only on the choice of the curve parameter ξ. On the other hand, the geometric curve that passes through the points x_i of the path $\mathcal{P}$ has infinitely many different parameterizations. The question that rises in this context is how we choose the curve parameter ξ.

The way to proceed is simple. Since we expect from the vehicle not only to pass sufficiently close to the points x_i of the path $\mathcal{P}(t_j)$ but additionally, to achieve this at instants of time t_j that we specify. Therefore, the curve parametrization ξ has to specify a velocity profile compatible with this requirement. Let us assume, without loss of generality, that we wish our agent to move with constant unit speed throughout the whole route to the goal destination. In that case, the curve has to be parameterized by the natural length s (intrinsic parametrization of the curve), where the parameter s was defined in (5.34). If x_i and x_{i+1} are two successive points in $\mathcal{P}(t_j)$, then the time $t_{\mathsf{x}_i \to \mathsf{x}_{i+1}}$ required for the agent to move from x_i and x_{i+1} is given by the arc length of the resulting path

$$\Delta s_{\mathsf{x}_i \to \mathsf{x}_{i+1}} = \int_{s_i}^{s_{i+1}} \left\| \frac{\mathrm{d}\vec{r}(z)}{\mathrm{d}z} \right\| \mathrm{d}z = s_{i+1} - s_i$$

since always $\left\|\frac{\mathrm{d}\vec{r}(s)}{\mathrm{d}s}\right\| = 1$. In other words, in this case there is no distinction between time and intrinsic parametrization of the curve.

We can extend this approach in the case where we wish the agent to move with varying speed other different parts of the path. A straightforward approach is to assign a higher speed when the vehicle is inside areas of high risk measure rm and lower ones otherwise. Thus, let m be the number of the district levels of the risk measure $M_1, M_2, \ldots, M_m$ over the world $\mathcal{W}$ and let $v \in [v_{\min}, v_{\max}]$ where $v_{\min} \geq \epsilon > 0$ where the tolerance ϵ is sufficiently large to guarantee that the speed never equals zero due to the constraint that the matrix $\tilde{\alpha}$ in 5.25 is non singular. We then divide $[v_{\min}, v_{\max}]$ also to m district levels and by employing a linear model we associate to the path that lies inside the region

$$\mathcal{D}_i \overset{\triangle}{=} \{(x, y) \in \mathcal{W} : M_{i-1} \leq \mathsf{rm}(x, y) \leq M_i\}$$

the speed value

$$v_i = v_{\min} + \left(i - \frac{3}{2}\right) \frac{v_{\max} - v_{\min}}{m - 1}. \tag{5.35}$$

The parametrization of the corresponding part of the curve lying in $\mathcal{D}_i$ is consequently

$$\xi = \frac{s}{v_i} \tag{5.36}$$

since then $\frac{\mathrm{d}s}{\mathrm{d}\xi} = v_i$.

5.4 Trajectory Generation and Time Scheduling Over a Receding Horizon

Let us assume that $\mathsf{x}(t_i)$ is the agent's current position and let $\mathcal{P}(t_i)$ be the solution path (i.e. ordered sequence of m_j distinct points x_j with $j = 1, \ldots, m_j$) to the goal destination x_f based on the environment representation of time t_i. Since $\mathsf{x}(t_i)$ is the first point in $\mathcal{P}(t_i)$, we approximate the polygonal line connecting the first three points, namely $\mathsf{x}(t_i) = \mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3$ of the path $\mathcal{P}(t_i)$ with a smooth curve $\vec{r}(\xi)$ which is the reference output of our system as described in the Section 5.3 . Then we execute the trajectory tracking scheme presented in Section 5.2 for time $t_{1\to 2}$, i.e. the time required for the agent to move from $\mathsf{x}(t_i)$ to x_2 which is given by

$$t_{1\to 2} = \int_{s_1}^{s_2} \frac{\mathrm{d}s}{v(s)}. \tag{5.37}$$

Assume that $\mathsf{x}_1 = \vec{r}(\xi_1) \in \mathcal{D}_1$ and $\mathsf{x}_2 = \vec{r}(\xi_2) \in \mathcal{D}_2$ and let $\vec{r}(\xi_{12})$ be the point corresponding to the intersection $\mathcal{D}_1 \bigcap \mathcal{D}_2 \bigcap \vec{r}(\xi)$ for $\xi \in (\xi_1, \xi_2)$. Since the agent moves for the most part with constant velocity v_1 or v_2 as long as it is inside $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, the time $t_{1\to 2}$ can be estimated by

$$t_{1\to 2} \simeq \int_{s_1}^{s_{12}} \frac{\mathrm{d}s}{v_1} + \int_{s_{12}}^{s_2} \frac{\mathrm{d}s}{v_2}. \tag{5.38}$$

and because inside $\mathcal{D}_1$ and $\mathcal{D}_2$ the s and ξ parameters are connected by the relation 5.36 the previous expression reduces to

$$t_{1\to 2} \simeq \xi_2 - \xi_1. \tag{5.39}$$

After the execution of trajectory tracking scheme over a time interval $t_{1\to 2}$ the agent's new configuration is $\mathsf{x}(t_{i+1})$ where $t_{i+1} = t_i + t_{1\to 2}$. At this time instant we compute the new path $\mathcal{P}(t_{i+1})$ using the *path planning scheme* of Chapters 3 or 4, where the point $\mathsf{x}(t_{i+1})$ is now the first point of $\mathcal{P}(t_{i+1})$. We simply repeat the procedure already described until after some finite time t_f the agent's configuration is arbitrarily close to the goal destination x_f, i.e.

$$\|\mathsf{x}(t_f) - \mathsf{x}_f\| \leq \epsilon \tag{5.40}$$

for some sufficiently small positive ϵ.

When one tries to implement this approach should be very careful since one of the assumptions of the tracking scheme we use was that the reference signal is twice differentiable. Thus, the curve parametrization should not result in discontinuous jumps of the speed (i.e. discontinuity of $\left\|\frac{\mathrm{d}r(\xi)}{\mathrm{d}\xi}\right\|$). However, the agent's transition from $\mathcal{D}_1$ to $\mathcal{D}_2$ results in a discontinuous jump on the agent's velocity. In order to eliminate this pathology induced by the curve parametrization (5.36) we propose a new parametrization for the part of the curve that corresponds to the transition from $\mathcal{D}_1$ to $\mathcal{D}_2$. This parametrization is given by

$$\xi = \frac{2s}{v_1(1 - \tanh(\frac{s - s_{12}}{\delta})) + v_2(1 + \tanh(\frac{s - s_{12}}{\delta}))}, \tag{5.41}$$

where δ is a sufficiently small positive number.

The time scheduling scheme presented in this section can be seen as an implicit way to impose input constraints. Actually, the way we approach the problem is to prescribe the speed with which the agent moves along the curve. Since perfect tracking in realistic simulation may not be achieved as fast as we want, the time scheduling procedure may not be perfectly accurate. However, it is a design issue to achieve at least a velocity profile $v(t)$ such that for the part of the trajectory which lies inside the area $\mathcal{D}_i$ the average actual velocity over the time period $\Delta t_{\mathcal{D}_i}$ (i.e period for which the agent remains inside the area $\mathcal{D}_i$) is sufficiently close to the v_i value given by (5.35), i.e

$$\left| \frac{\int_{\Delta t_{\mathcal{D}_i}} v(t)\mathrm{d}t}{\Delta t_{\mathcal{D}_i}} - v_i \right| \leq \varepsilon_0 \tag{5.42}$$

for some sufficiently small positive ε_0.

Alternatively, in problems where the time scheduling is not an important design objective we can alternatively impose constrains on the angular velocity control input using similar ideas as those presented in this section. In particular, the angular velocity using equations (5.1)-(5.3) can be written as

$$\omega = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} \tag{5.43}$$

provided that $\dot{x}^2 + \dot{y}^2 \neq 0$. Since the speed of the agent is given by (5.33) and additionally the curvature of the trajectory curve is

$$\kappa = \frac{|\ddot{y}\dot{x} - \ddot{x}\dot{y}|}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{5.44}$$

the angular velocity can be written as

$$\omega = \mathrm{sign}(\ddot{y}\dot{x} - \ddot{x}\dot{y})\frac{\kappa}{\sqrt{v}}. \tag{5.45}$$

Therefore, the magnitude of ω is given by

$$|\omega| = \frac{\kappa}{\sqrt{v}}. \tag{5.46}$$

Because the curvature κ is an invariant of the curve under arbitrary curve parametrization ξ, the only way to impose constraints over the magnitude of the angular velocity is by finding an appropriate profile for the speed of the vehicle along the solution path. To this end, let us assume that the magnitude of the angular velocity should satisfy the following condition

$$|\omega| \leq \omega_{\max} \tag{5.47}$$

where $v \in [v_{\min}, v_{\max}]$ as in Section 5.3. Then, equations (5.46) and (5.47) imply that

$$\left(\frac{\kappa}{\omega_{\max}}\right)^2 \leq v \leq v_{\max} \tag{5.48}$$

or equivalently

$$\left(\frac{\kappa}{\omega_{\max}}\right)^4 \leq \dot{x}^2 + \dot{y}^2 \leq v_{\max}^2. \tag{5.49}$$

In case we want to incorporate the time scheduling task to the above problem formulation, then we expect the appearance of conflicts between the design objectives and the problem constraints. Therefore, the solution of such a problem requires a more complex and powerful methodology compared to the time scheduling scheme presented so far.

Chapter 6

Simulation Results

6.1 Introduction

In this chapter we present simulation results of the proposed *path planning scheme* and the smooth trajectory generation and time scheduling scheme.

For the first planning algorithm we present the results for two non-trivial scenarios. In both cases, the environment is assumed to be a square of dimension 512×512 units. Hence $N = 9$ is the finest resolution possible. For simplicity, for both scenarios only two levels of resolution have been chosen to represent the environment. Inside an area of 100×100 unit cells we employ a high resolution approximation and outside this area we employ a low resolution approximation of $\mathcal{W}$.

All the above assumptions hold in the second planning scheme. The only difference here is that the high resolution area corresponds to the area inside a disk of radius 100 pixels. This is the area of 100×512 unit cells in the (r, θ) plane when it is mapped to the (x, y) plane. Outside this area we employ a low resolution approximation of $\mathcal{W}'$.

Simulation for the smooth trajectory and time scheduling scheme are presented for all the scenarios of the two algorithms.

6.2 Simulation Results of the First Scenario for the First Path Planning Scheme

In the first scenario, the environment $\mathcal{W}$ is an actual topographic (elevation) map of a certain US state with fractal-like characteristics, shown in Fig. 6.1. The blue color in this figure corresponds to areas of obstacles whereas the initial configuration of the agent is denoted by A and the desired final configuration is denoted by B. The objective is for the agent (e.g., a UAV) to follow a path from A to B while flying as low as possible, and below a certain elevation threshold. Areas with bright colors in Fig. 6.1 correspond to areas of low risk (elevation in this case) and darker colors correspond to areas of high risk (elevation in this case) that should be avoided. Solving the path-planning problem on-line at this resolution is computationally prohibitive.

In order to apply the proposed multiresolution scheme we choose five distinct risk measure levels $M_1, \ldots, M_5$, equally spaced between 0 and 1. The two levels with the highest values ($M_4 = 0.75$ and $M_5 = 1$) denote the obstacle space; that is, $m_2 = 0.75$. The rest three levels $M_1, \ldots, M_3$ denote feasible states. Level M_1 represents the unoccupied, most desirable states and we thus chose $m_1 = M_1$.

The results from the multiresolution path-planning algorithm using a fine resolution level $J_{\max} = 5$, and a low resolution at level $J_{\min} = 3$ are shown in Fig. 6.2. Specifically, Fig. 6.2 shows the evolution of the path at different time steps as the agent moves to the final destination. Figure 6.2(a) shows the agent's position at time step $t = t_{15}$ along with the best proposed path to the final destination at that time. Similarly, Fig. 6.2(b) shows the agent's position at time step $t = t_{50}$ along with the best proposed path to the final destination at that time. As seen in Fig. 6.2(c), the actual path followed by the agent differs significantly from the one predicted in either Figs. 6.2(a) or 6.2(b). This happens because at time t_{15} and t_{50} the agent does not have complete information outside the high resolution zone, and the predicted path actually penetrates the obstacle space $\mathcal{O}$. At time t_{50}, for example, the agent – being far from any obstacle – fails to anticipate the upcoming collision. As the agent gets closer to the obstacle however, and new information is gathered, the existence of the obstacle forces the agent to redirect its path. The agent reaches the final destination x_f in a collision free manner, as seen in Fig. 6.2(c). The actual path followed lies inside areas with a low elevation level, which verifies the optimal nature of the path.

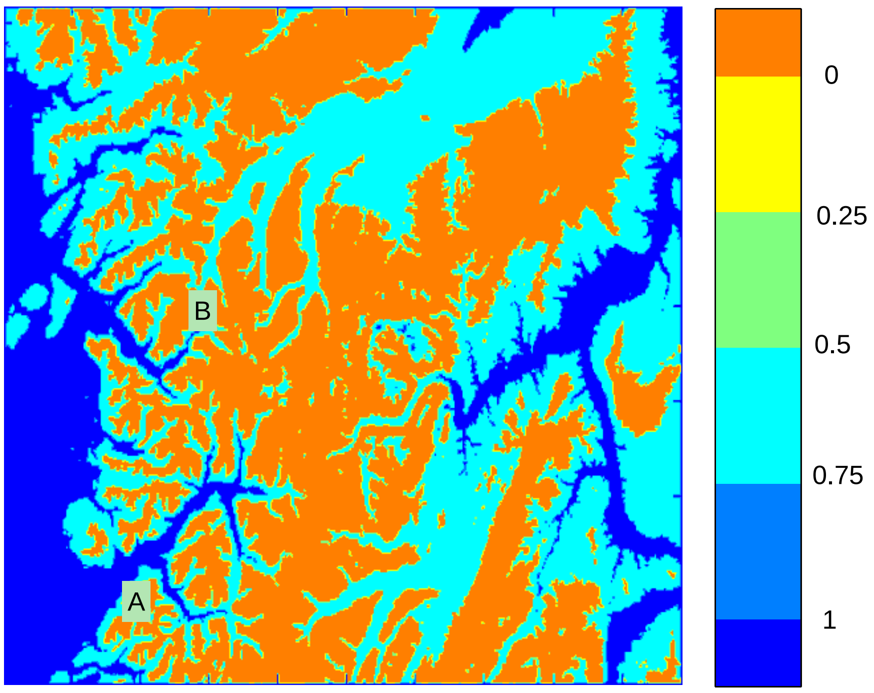
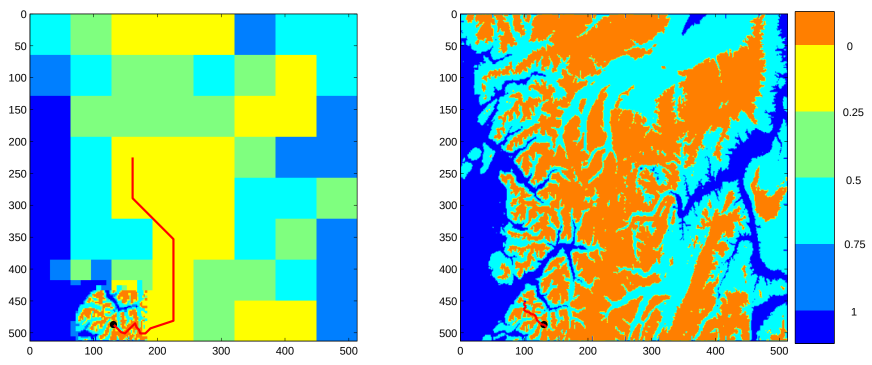
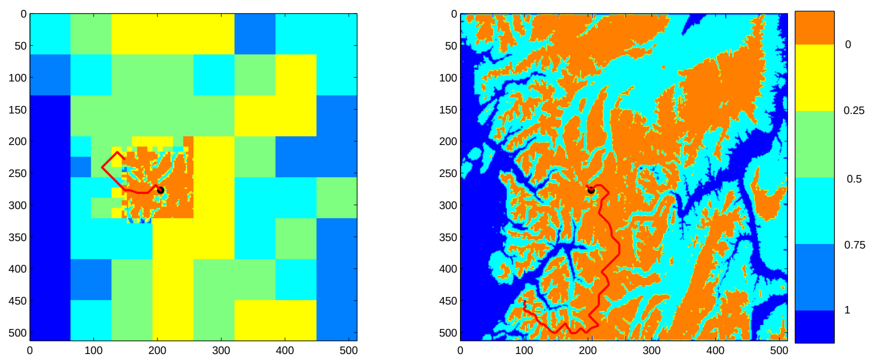


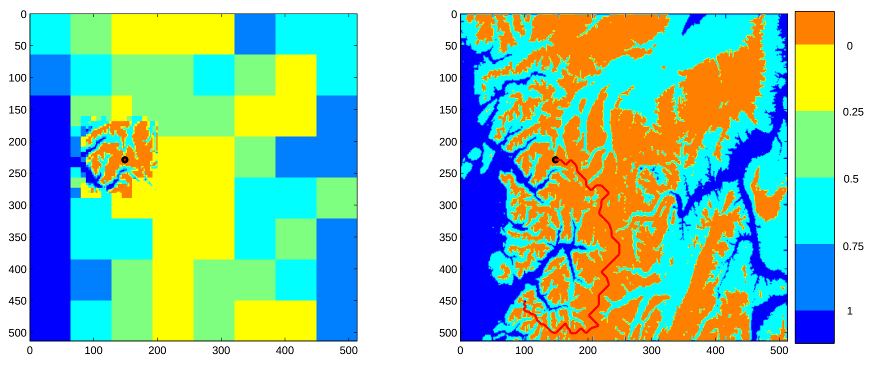
Figure 6.1: Plot of risk measure (elevation) for the whole configuration space using a 512×512 unit cell resolution.



(a) $t = 15$



(b) $t = 50$



(c) $t = 62$

Figure 6.2: Path evolution and replanning at time $t = t_{15}$, $t = t_{50}$ and $t = t_f$.

6.3 Simulation Results of the Second Scenario for the First Path Planning Scheme

In the previous scenario the cost to be minimized along the path is derived solely from the risk measure shown in Fig. 6.1. When the environment is very fragmented, this cost may result in excessively long, meandering paths. To avoid this problem for a cluttered environment as the one shown in Fig. 6.3 we add an additional term that also penalizes the total length of the path. This allows the agent to prefer short paths in the euclidean sense. Four distinct risk measure levels M_1, M_2, M_3, M_4 are chosen for the scenario shown in Fig. 6.3, as follows $M_1 = 0, M_2 = 0.25, M_3 = 0.75$, and $M_4 = 1$. Here we let again $M_2 = 0.75$ and $M_1 = 0$. Hence values above 0.75 denote obstacles, shown in red color in Fig. 6.3. The final path obtained using the proposed multiresolution path-planning algorithm is shown in Fig. 6.4.

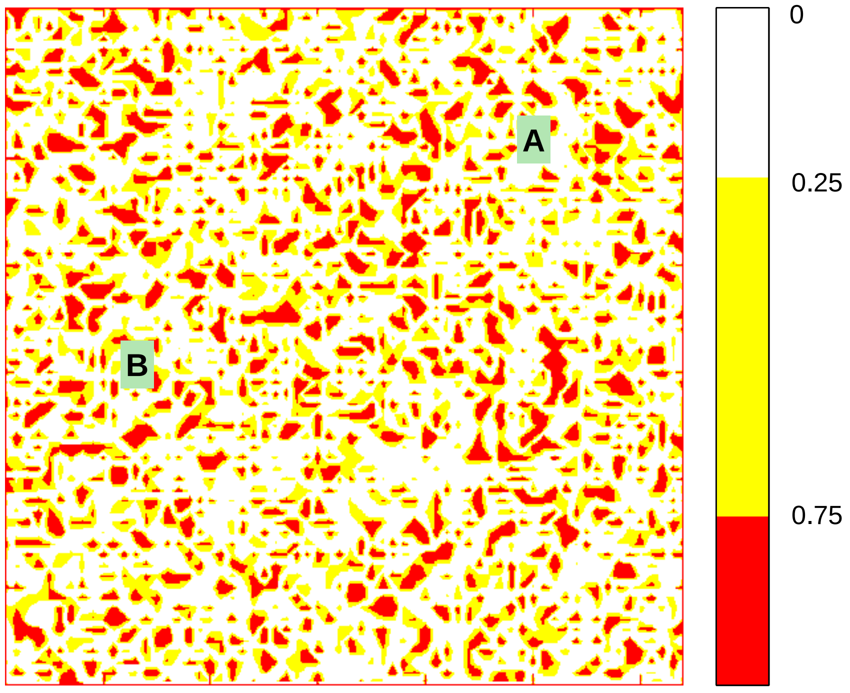


Figure 6.3: Plot of the risk measure function for the second scenario. Areas with red color correspond to the obstacle space. The point ‘A’ denotes the initial state x_0 and point ‘B’ denotes the final state x_f.

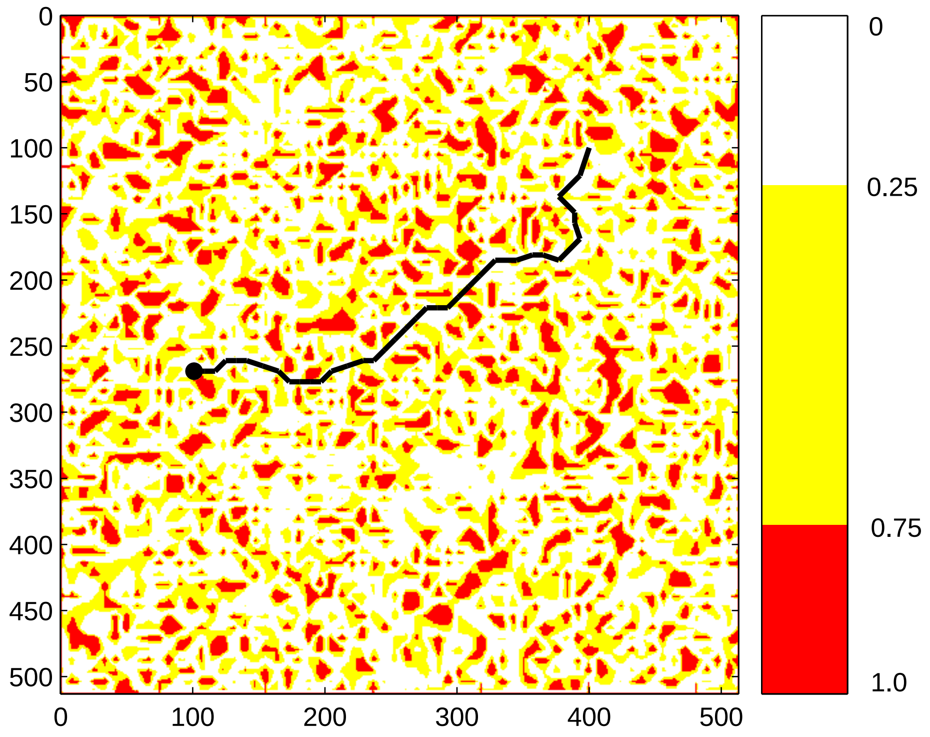


Figure 6.4: Final path for the second scenario. For such highly fragmented environments it is advisable to also include a penalty on the euclidean distance between successive nodes of the path.

6.4 Simulation Results of the Second Path Planning Scheme

In this section we present simulation results of the second proposed algorithm for a non-trivial scenario. The environment is assumed to be square of dimension 512×512 units. Hence $N = 9$ is the finest resolution possible. For simplicity, only two levels of resolution have been chosen to represent the environment. Inside an area of 100×100 unit cells in the (r, θ) system we employ a high resolution approximation and outside this area we employ a low resolution approximation of $\mathcal{W}'$.

The initial and final positions of the agent are also shown in this figure. The objective is for the agent (e.g., a UAV) to follow a path from A to B while flying as low as possible, and below a certain elevation threshold. Areas with bright colors in Fig. 6.6 correspond to areas of low risk (elevation in this case) and darker colors correspond to areas of high risk (elevation in this case) that should be avoided. Solving the path-planning problem on-line at this resolution is computationally prohibitive.

In order to apply the proposed multiresolution scheme we choose six distinct risk measure levels $M_1, \ldots, M_6$, spaced between 0 and 1. The two levels with the highest values ($M_5 = 0.9$ and $M_6 = 1$) denote the obstacle space; that is, $m_2 = 0.9$. The rest four levels $M_1, \ldots, M_4$ denote feasible states. Level M_1 represents the unoccupied, most desirable states and we thus chose $m_1 = M_1$.

The results from the multiresolution path-planning algorithm using a fine resolution level $J_{\max} = 5$, and a low resolution at level $J_{\min} = 3$ are shown in Fig. 6.5. Specifically, Fig. 6.6 shows the evolution of the path at different time steps as the agent moves to the final destination.

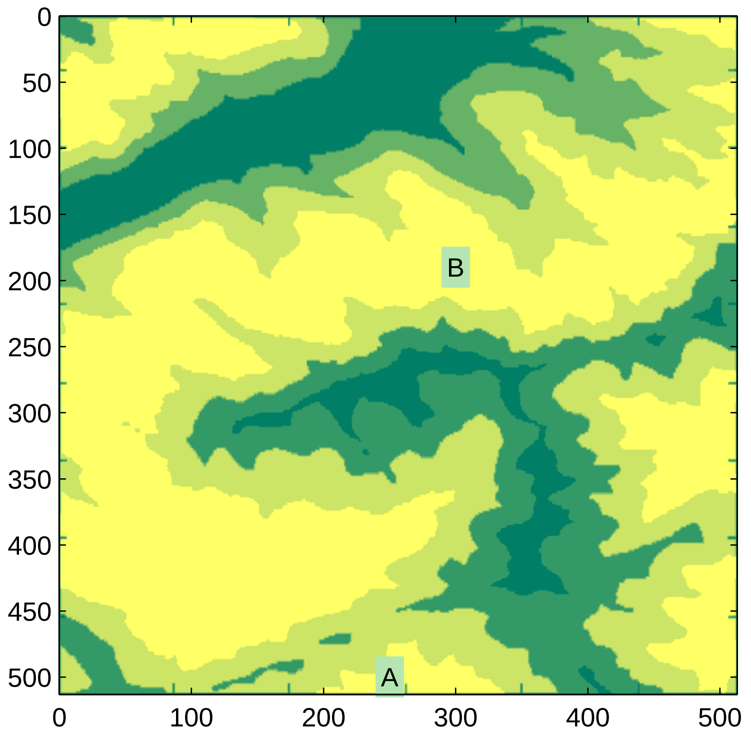
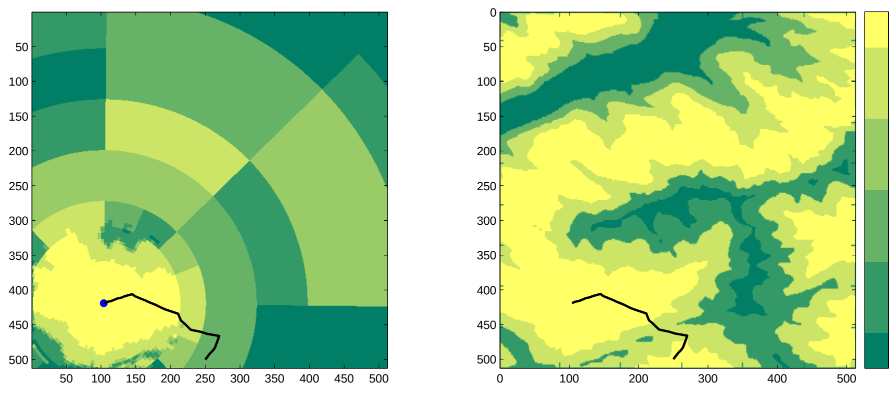
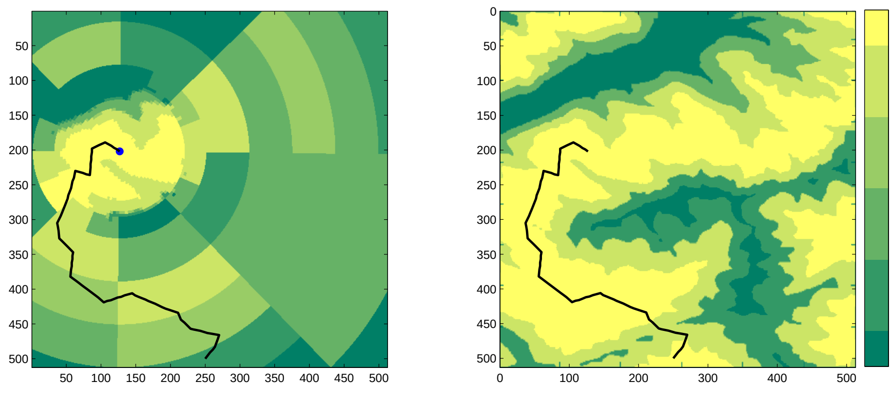


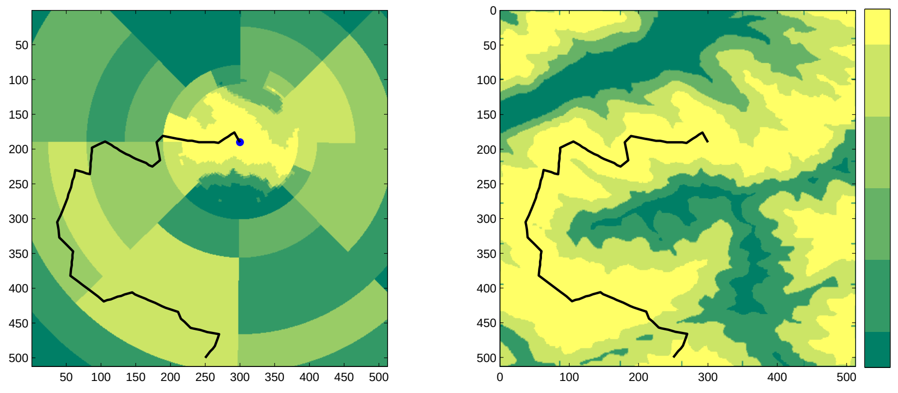
Figure 6.5: Plot of the original risk measure function where dark green area correspond to areas of high risk whereas the yellow ones to low risk areas.



(a) $t = 10$



(b) $t = 22$



(c) $t = t_f$

Figure 6.6: Path evolution. Figures show the actual path followed by the agent which finally reaches the final destination at $t = t_f$.

6.5 Simulation Results of Trajectory and Time Scheduling Scheme

In this section we present the results of the trajectory generation and scheduling scheme we presented in Chapter 5. Actually, we shall reproduce the results of the simulation for all the scenarios of the two proposed path planning algorithms, presented in the previous two sections, using the kinematic unicycle model.

To simplify the computations we require in all cases the vehicle to move with constant unit speed as long as it is inside the free world $\mathcal{F}$. The reference output signals (reference curve) correspond to a smooth approximation of the polygonal line of the path $\mathcal{P}$. In Figures 6.7-6.9 we see the simulation results on the first scenario of the first planning scheme. In Figures 6.10-6.12 we see the simulation results on the scenario with the environment full of obstacles of the first planning scheme. Finally, in Figures 6.13-6.15 the simulation results of the trajectory generation scheme for the scenario of the second planning scheme are presented.

We observe that the generated trajectory in all cases achieves the closeness requirement to the final path $\mathcal{P}$ (polygonal line) that the planning schemes give us. This guarantees that the resulting smooth curve lies completely inside the free world $\mathcal{F}$. Additionally, the trajectory at the specified time instants t_j (time scheduling) passes sufficiently close to the visiting points x_i of the corresponding available path $\mathcal{P}(t_j)$. Furthermore, the implicit constraint on the agent's speed is achieved in a satisfactory degree in the sense of condition (5.42) with ε_0 be in the order of 10^{-3}. Any deviation or oscillation phenomena that appear are due to the transient response of the system during the error regulation procedure. These are mostly due to the *on-line* implementation nature of our scheme since a small deviation in the initial conditions (especially the ones for the orientation θ) from those that correspond to the reference curve may result in a transient response for each time we execute the trajectory generation scheme. This transient response is decaying very fast and it is a design issue to make it practically disappear.

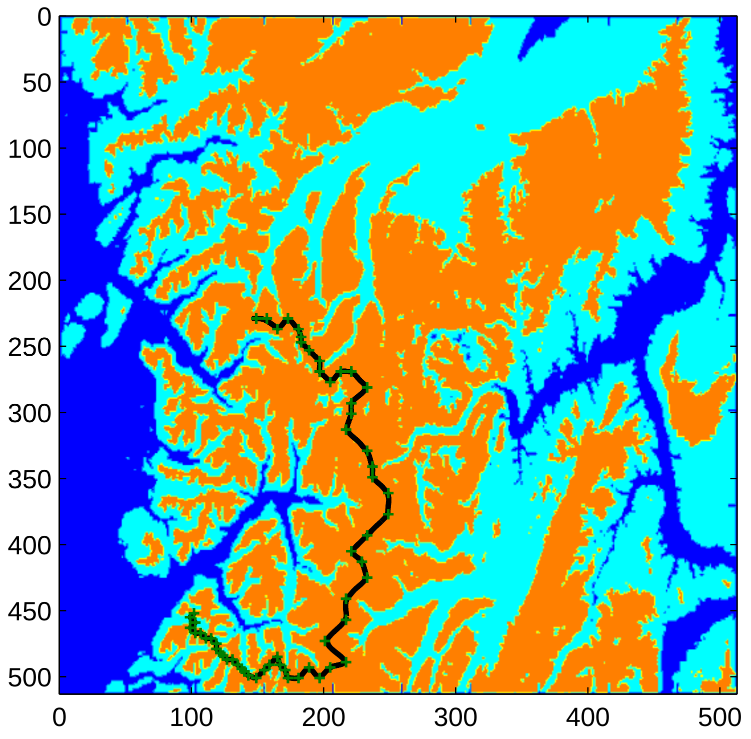


Figure 6.7: Plot of the *on-line* generated smooth trajectory passing close to the x_i points of the corresponding path $\mathcal{P}(t_j)$ generated by the first planning scheme (denoted with '+') at specified instants of time t_j for the 1st scenario.

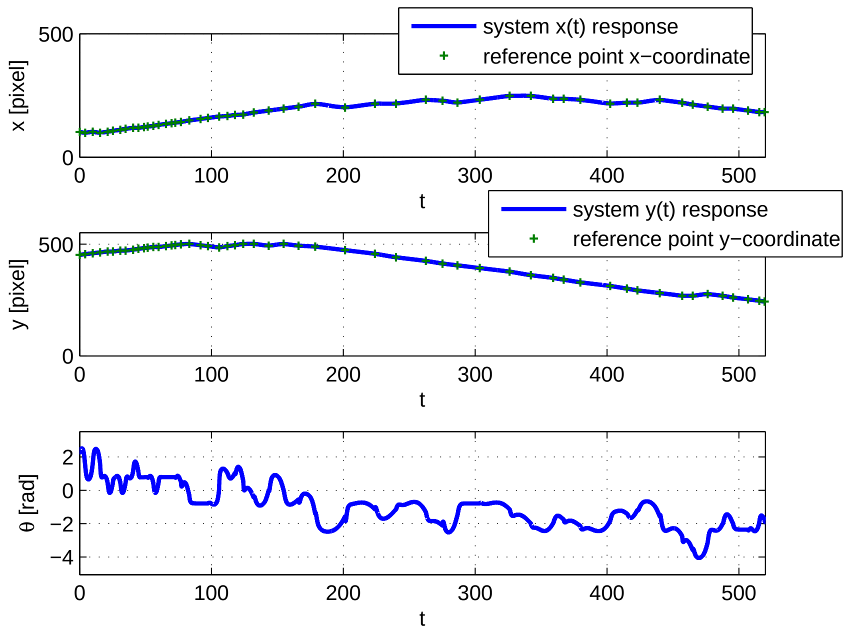


Figure 6.8: Plot of the states evolution of the system under the feedback law.

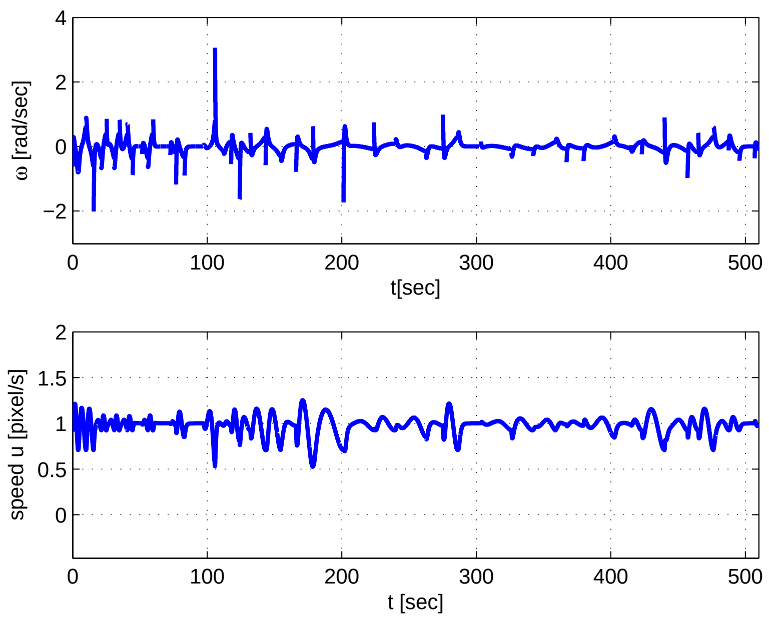


Figure 6.9: Plot of the input components (i.e. velocity v and angular velocity ω) versus time.

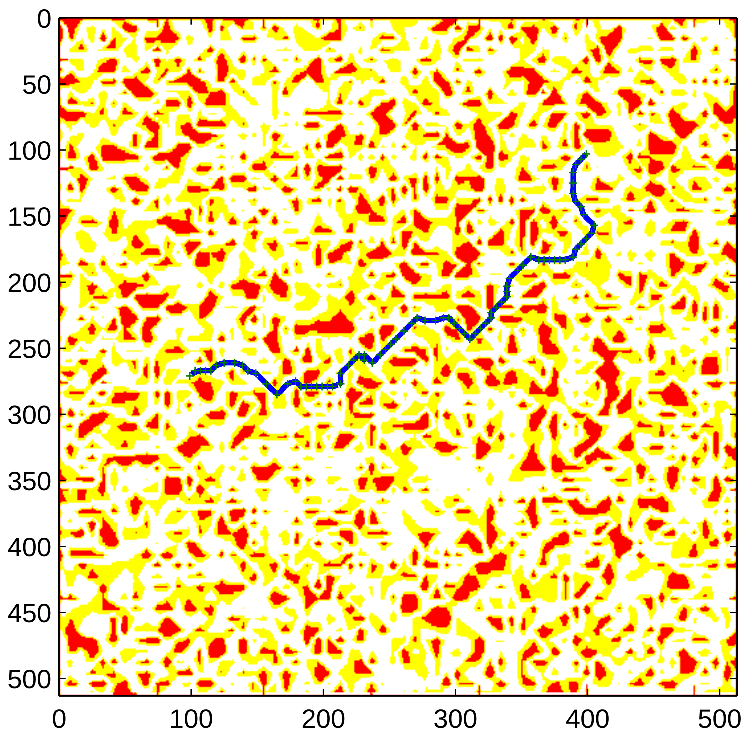


Figure 6.10: Plot of the *on-line* generated smooth trajectory passing close to the x_i points of the corresponding path $\mathcal{P}(t_j)$ generated by the planning scheme (denoted with '+') at specified instants of time t_j for the fragmented environment scenario.

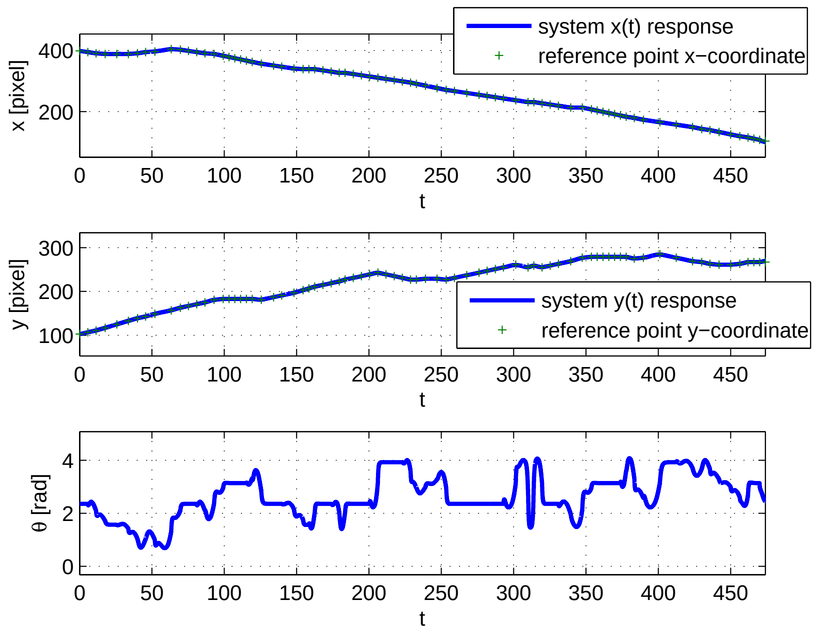


Figure 6.11: Plot of the states evolution of the system under the feedback law.

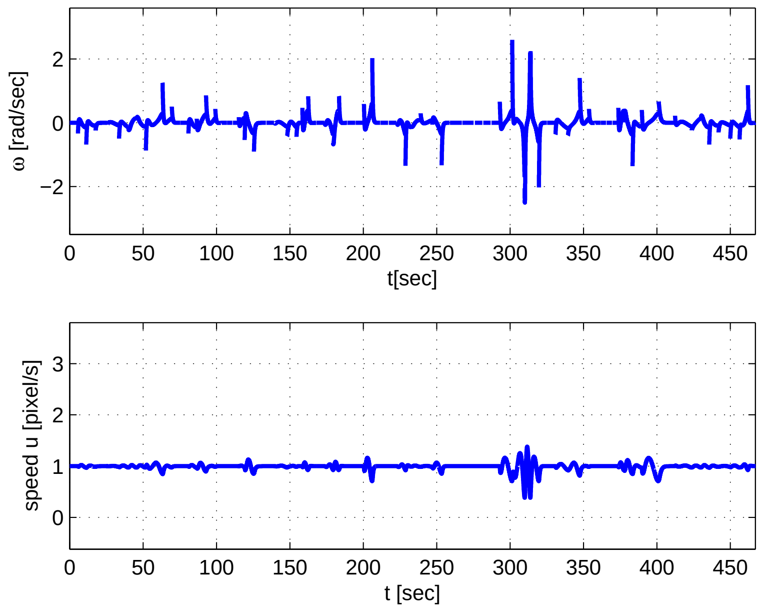


Figure 6.12: Plot of the input components (i.e. velocity v and angular velocity ω) versus time.

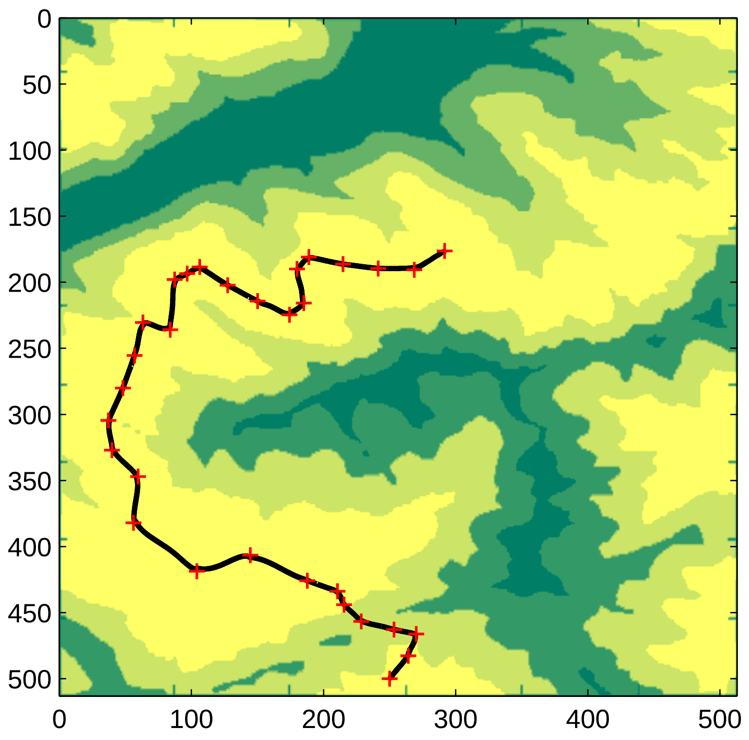


Figure 6.13: Plot of the *on-line* generated smooth trajectory passing close to the x_i points of the corresponding path $\mathcal{P}(t_j)$ generated by the second planning scheme (denoted with '+') at specified instants of time t_j.

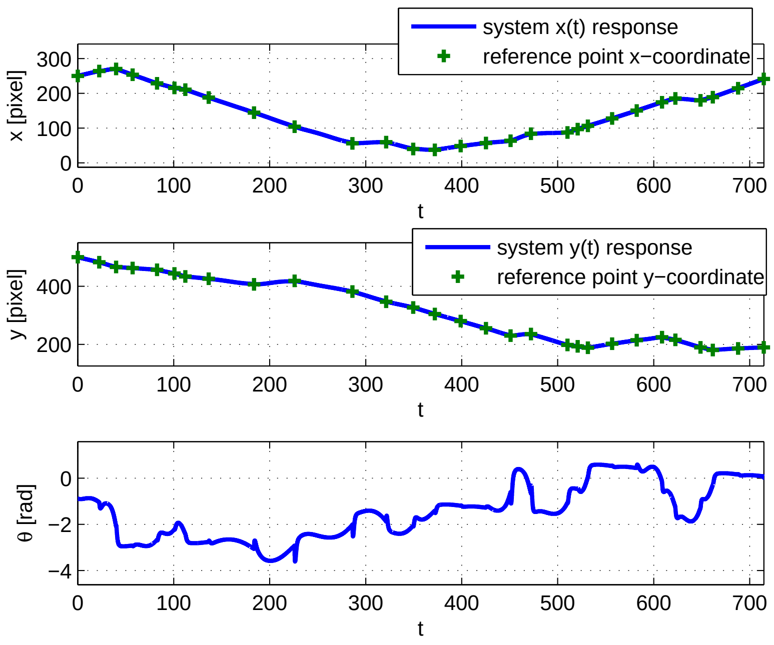


Figure 6.14: Plot of the states evolution of the system under the feedback law.

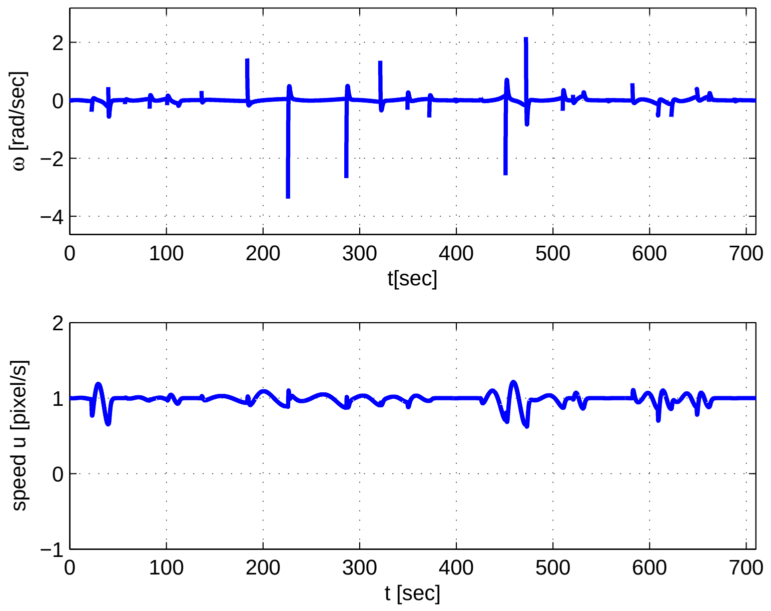


Figure 6.15: Plot of the input components (i.e. velocity v and angular velocity ω) versus time.

Chapter 7

Conclusions and Future Work

In this work we have proposed an *on-line* hierarchical *path planning scheme* for navigating an autonomous agent inside an unknown environment based on information obtained by its available on-board sensor devices. The idea is to use a higher resolution close to the agent where is needed most, and a coarser resolution at large distances from the current location of the agent. This is motivated by the natural observation that for on-line implementations it is not prudent from a computational point of view to compute a solution with great accuracy over large ranges of over a very long time horizon. The *planning scheme* is scalable and can be tailored to the available computational resources of the agent. An extension of the original idea motivated by practical consideration is also presented. In this extension, a sector-based multiresolution decomposition of the environment is computed at each step. This decomposition is adapted to the on-board sensor data using the wavelet transform in conjunction with a conformal mapping to new (polar) coordinates. The multiresolution approach allows the agent to blend information arising from different sensors at different ranges and resolutions. Furthermore, some dynamical considerations for the agent comes into play by employing a kinematic unicycle model. In order to produce smooth trajectories compatible with the dynamic constraints we introduce a smooth trajectory generation and time scheduling scheme using some basic ideas of IOL theory for MIMO systems.

The proposed methodology can be further expanded towards different directions. One important issue is to investigate how it is possible to reduce the computations required to obtain a solution at a given instant of time if previous information is used appropriately. Additionally, the problem of smooth trajectory generation and time scheduling can be examined for more complicated kinematic or even dynamic models for the agent under control input constraints (especially for the angular velocity). In order to obtain a solution for the constrained problem the range and resolution of the information obtained by the agent's sensors may have to be appropriately modified. In this case, the robustness of the proposed scheme under these possible modifications is another crucial issue which is strongly correlated with how efficient the path planning scheme is for real time implementation. These possible extensions of the baseline methodology presented in this thesis will be addressed in the future along with the exploration of other research horizons in the framework of the path planning problem for autonomous vehicles.

Bibliography

[1] Y. K. Hwang and N. Ahuga, "Gross motion-planning–a survey," *ACM Computing Surveys*, vol. 24, no. 3, pp. 219–291, 1992.

[2] M. Athans and P. Falb, *Optimal Control, An Introduction to the Theory and Its Application.* New York: Dover, 2007.

[3] R. Bartle, *The Elements of Real Analysis.* New York: Wiley Sons Inc., second ed., 1976.

[4] H. Khalil, *Nonlinear Systems.* New Jersey: Prentice Hall, third ed., 2002.

[5] J. Latombe, *Robot Motion Planning.* Boston, MA: Kluwer Academic Publishers, 1991.

[6] S. Lavalle, *Planning Algorithms.* New York, NY: Cambridge University Press, 2006.

[7] D. Koditschek, "Exact robot navigation by means of potential functions: Some topological considerations," in *IEEE Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1–6, 1987.

[8] D. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," in *Advances Appl. Math.*, vol. 11, pp. 412–442, 1990.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms.* McGraw Hill and MIT Press, second ed., 2001.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," in *IEEE Transactions on Systems Science and Cybernetics SSC4)*, pp. 100– 107, 1968.

[11] A. Stentz, “Optimal and efficient path planning for partially-known environments,” in *IEEE Proceedings of the IEEE International Conference of Robotics and Automation*, pp. 3310–3317, 1994.

[12] A. Ferguson, D. Stentz, “The delayed d* algorithm for efficient path replanning,” in *IEEE Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2045– 2050, 2005.

[13] M. Koenig, S. Likhachev, “Improved fast replanning for robot navigation in unknown terrain,” in *IEEE Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 968– 975, 2002.

[14] S. Murphey, S. Uryasev, and M. Zabarankin, “Trajectory optimization in a threat environment,” *Research Report 2003-9*, pp. 22–45, 2003.

[15] A. Isidori, *Nonlinear Control Systems.* Berlin: Springer - Verlang, third ed., 1995.

[16] I. Daubechies and W. Sweldens, “Factoring wavelets transforms into lifting steps,” *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, 1998.

[17] T. Lozano-Perez and M. Wesley, “Automatic planning for planning collision-free paths among polyhedral obstacles,” in *IEEE Trans. Syst., Man, Cybern.*, vol. 11, pp. 681–698, 1981.

[18] D. Pai and L. Reissell, “Multiresolution rough terrain motion planning,” in *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 19–33, 1998.

[19] S. Kambhampati and L. Davis, “Multiresolution path planning for mobile robots planning for mobile robots,” *IEEE Journal of Robotics and Automation*, pp. 135–145, 1986.

[20] S. Behnke, “Local multiresolution path planning,” *Lecture Notes in Computer Science*, 2004.

[21] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Cambridge, MA: Athena Scientific, 1995.

[22] B. Bollobas, *Modern Graph Theory.* New York: Springer, 1998.

[23] C. Godsil and G. Royle, *Algebraic Graph Theory.* New York: Springer, 2001.

[24] R. Diestel, *Graph Theory.* New York: Springer, third edition ed., 2006.

[25] C. Burrus, R. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms.* New Jersey, NJ: Prentice Hall, 1998.

[26] S. Mallat, "A theory for multiresolution signal decomposition, the wavelet representation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, 1989.

[27] A. Cohen, *Numerical Analysis of Wavelet Methods*, vol. 32. Amsterdam: Elsever Science, 2003.

[28] D. Walnut, *An Introduction to Wavelet Analysis*, vol. 32. Boston: Birkhauser, 2003.

[29] P. Tsiotras and E. Bakolas, "A hierarchical on-line path-planning scheme using wavelets," in *European Control Conference*, (Kos, Greece), July 2–5 2007.

[30] M. Do Carmo, *Differential Geometry of Curves and Surfaces.* New Jersey: Prentice Hall, 1976.

Findings

The advantages of multi-resolution and wavelet-based strategies were demonstrated in the following application areas.

Multiresolution Grid Generation for Hyperbolic PDE's: We developed a novel multiresolution adaptive mesh refinement algorithm for solving general pde's of hyperbolic type (e.g., Hamilton-Jacobi equations). Hyperbolic pde's may exhibit non-smooth solutions even if the initial data is smooth. We have investigated adaptive gridding techniques using multiresolution ideas in order to represent solutions of partial differential equations of the Hamilton-Jacobi type. The motivation for investigating HJ pde's stems from (i) their use in deriving optimal feedback problems and (ii) the fact that often the solution of a pde of the Hamilton-Jacobi type is not smooth, even if the given initial conditions are smooth.

To capture the discontinuity and/or nonsmoothness in the solution we need to use a high resolution grid, locally, close to this discontinuity. Hence in order to solve these problems in a computationally efficient way, the computational grid should adapt in space and in time to reflect local changes in the solution. The proposed algorithm dynamically adapts the grid to any existing or emerging irregularities in the solution, by refining the grid only at places where the solution exhibits sharp features. The net result is a grid with a fewer number of nodes (about 30% less for one-dimensional problems), when compared to adaptive grids generated by other techniques.

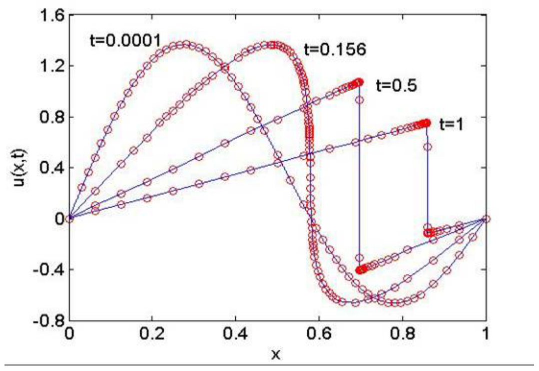
Table 1 shows an error comparison between the use of a uniform and a non-uniform grid for one of the examples in [1].

Table 1: L_1 error and computational times for uniform vs. adaptive mesh.

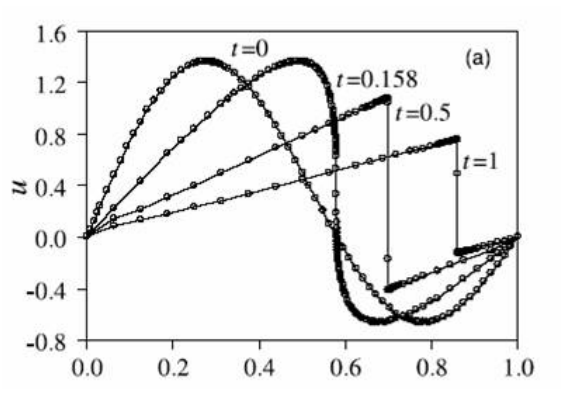
	Uniform Mesh		Adaptive Mesh			
$J_{\max}$	N_{g} in $\mathcal{V}_{J_{\max}}$	t_{cpu} (s)	N_{g} at t_f in $\mathsf{Grid}_{\mathrm{new}}$	$E_1(u)$	t_{cpu} (s)	Speed Up
8	$2^8+1=257$	2.7106	31	7.1991×10^{-3}	0.5835	4.6454
9	$2^9+1=513$	9.3851	34	7.1717×10^{-3}	1.2737	7.3684
10	$2^{10}+1=1025$	36.6631	37	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

Additional details of the approach with more examples can be found in [1, 2] and [3].

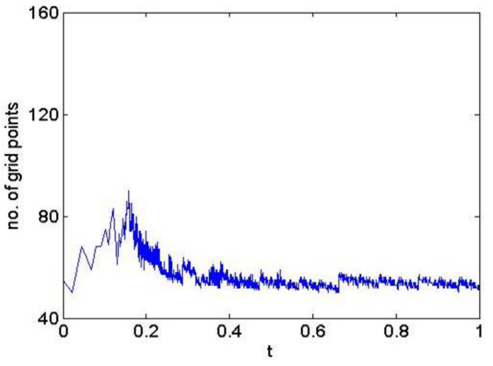
Multiresolution Trajectory Optimization: For direct trajectory optimization problems, one typically transcribes the optimal control problem into a nonlinear programming (NLP) problem and then one solves the resulting NLP problem on a given grid. Good accuracy requires a dense (fine) grid, whereas computational speed requires a thin (coarse) grid. Real-time or nearly real-time calculation of optimal controllers necessitates fast computations with minimum degradation in accuracy. We have developed a novel multiresolution-based approach for solving trajectory optimization problems. The original optimal control problem is solved using a direct method, thereby being transcribed into a nonlinear programming (NLP) problem that is solved using standard NLP codes. The novelty of the proposed approach (Multiresolution Trajectory Optimization Algorithm - MTOA) hinges on the automatic calculation of a suitable, nonuniform grid over which the NLP problem is subsequently solved. This 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 conflicting objectives, such as accuracy of the solution, convergence, and speed of computations. The benefits of the proposed algorithm over uniform grid implementations are demonstrated with the help of



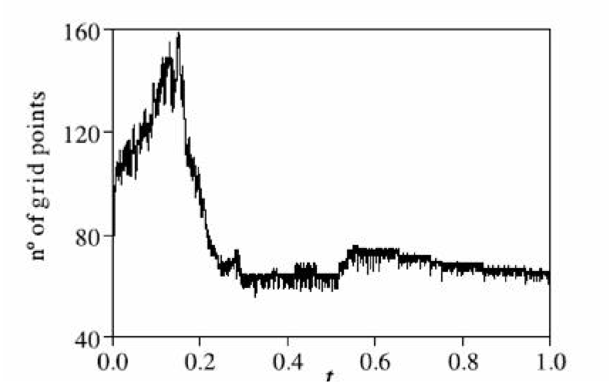
(a)



(b)



(c)



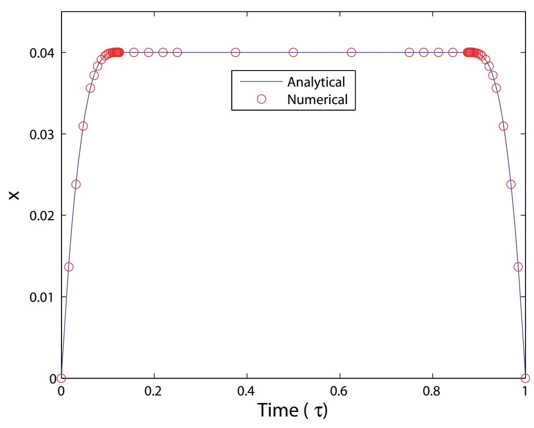
(d)

Figure 1: Comparison of our method (left column) with the one in (M.A. Alves, P. Cruz, A. Mendes, F.D. Magalhaes, F.T. Pinho, and P.J. Oliveira, "Adaptive Multiresolution Approach for Solution of Hyperbolic PDEs," *Comput. Methods Appl. Mech. Eng.*, 191 (2002), pp. 3909–3928) (right column); Figures (a) and (b) show the solution obtained with each method; Figures (c) and (d) compare the corresponding number of grid points. We obtain the same accuracy albeit using fewer grid points.

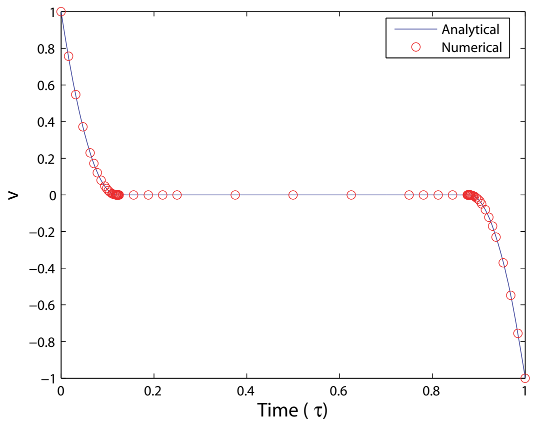
several nontrivial examples. Figures 2 and 3 show typical grids resulting from the application of MTOA.

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.

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

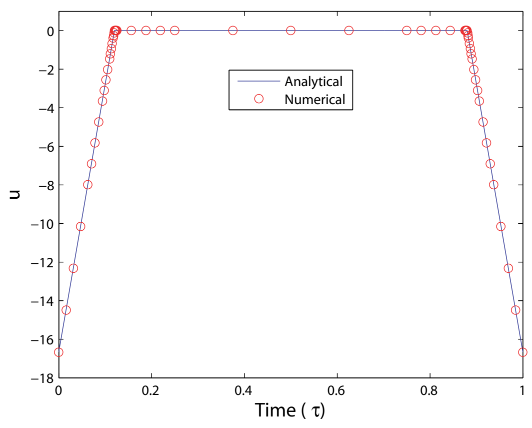


(a) Iteration 8: Time history of x.

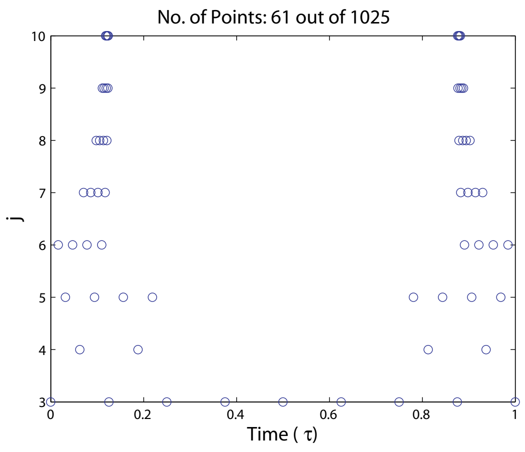


(b) Iteration 8: Time history of v.

Figure 2: Time history of $x,\ v$ at the final iteration of MTOA.



(a) Iteration 8: Time history of u.



(b) Iteration 8: Grid point distribution.

Figure 3: Time history of u along with the grid point distribution at the final iteration of MTOA.

of the interpolating polynomials, the maximum/minimum time steps, etc) can be chosen during implementation and to yield consistent approximations. Details can be found in [4, 5].

As a natural extension of the baseline algorithm, we have also 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 paper confirm these observations. Given their simplicity and efficiency, the proposed techniques offer a potential for online implementation. Details can be found in [6, 7].

Path-Planning: For path planning problems, we have used wavelets to perform hierarchical multi-resolution for navigating an autonomous agent inside an environment full of obstacles. Our method uses higher resolution close to the agent (where it is needed most) and a coarser resolution at large distances from the current location of the agent. Since the range and resolution levels can be chosen by the user, the algorithm results in search graphs of known a priori complexity. The approach is motivated by the observation that for on-line implementation it is not prudent from a computational point of view to find a solution with great accuracy over large ranges and over a very long time horizon.

The approach allows the construction of a directed weighted graph the dimension of which can be adapted to the on-board computational resources. By searching this graph we find the desired shortest path to the final destination using Dijkstra's algorithm, provided that such a path exists.

The algorithm is scalable and can be tailored to the available computational resources of the agent. Furthermore, we have developed an algorithm for computing the adjacency matrix directly from the wavelet decomposition, without the need to perform an additional quadtree decomposition on the reconstructed function [8, 9, 10]. Our numerical experiments have shown that this approach speeds up the whole path-planning process.

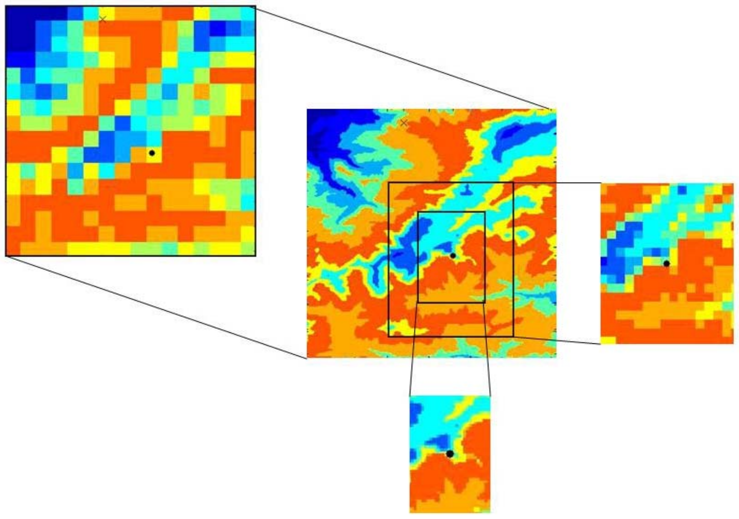
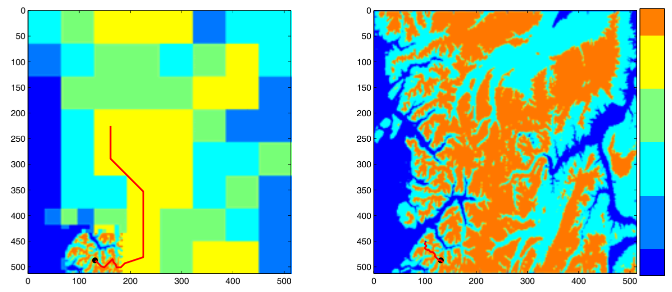


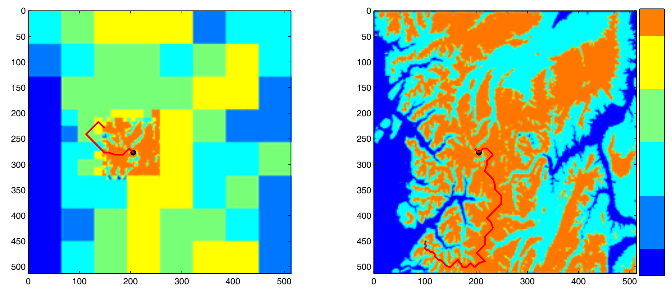
Figure 4: Multiresolution representation of the environment according to the distance from the current location of the agent.

We have applied this algorithm and have solved path-planning problems in challenging environments of realistic data (topographic maps) as well as in cluttered environments with fractal-like characteristics (see Fig. 5). Details can be found in [11, 10, 12].

In [13] we have extended the results of [11], by employing a conformal mapping to devise a hybrid local/global path planning algorithm using sector cell decompositions instead of decompositions that employ only rectangular or square cells. Sector cells are compatible to the on-board sensors, and thus process the data more efficiently, in a manner that does not contradict its original sector-based form. We provide approximations with special localized attributes by combining efficiently data from sensors of different resolutions and ranges. Furthermore, in the work of [11] the whole environment was assumed to be known a priori and the wavelet approximation scheme allowed us to plan the path using only a small fraction of the available information.

In [14] a path planning algorithm is proposed that constructs a cell decomposition at several levels of


(a) $t = t_{15}$


(b) $t = t_{50}$

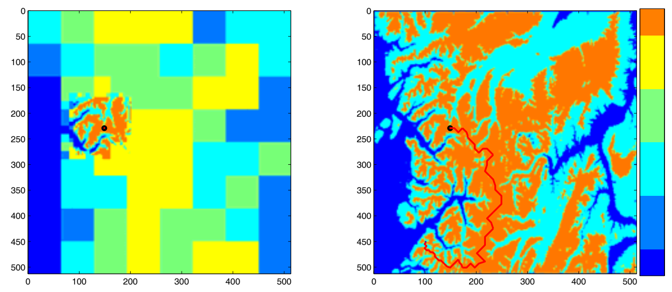

(c) $t = t_f$

Figure 5: Path evolution and replanning. Figures on the left show the currently tentative optimal path obtained from Dijkstra's algorithm, based on the available multiresolution approximation of the environment at different time steps. Figures on the right show the actual path followed by the agent.

resolution (cell sizes) and constructs an optimal path to the destination from the current location of the agent. At each step the algorithm iteratively refines a coarse approximation to the path through local replanning. The replanning process uses previous information to refine the original cell channel in the immediate area of the path. This is done efficiently using the wavelet coefficients. Numerical tests show a speed-up of an order of magnitude over the baseline algorithm with minimal impact on the overall optimality of the resulting path. A comparative study with the well-known D* algorithm is also provided.

Multiresolution for Level Set Methods: We have developed a new multi-resolution gridding scheme to accelerate level set methods for solving pdes. Level set (LS) methods have been widely used for solving certain types of partial differential equations appearing in many applications, such as computing optimal feedback strategies, image segmentation, computer graphics, etc. In most LS

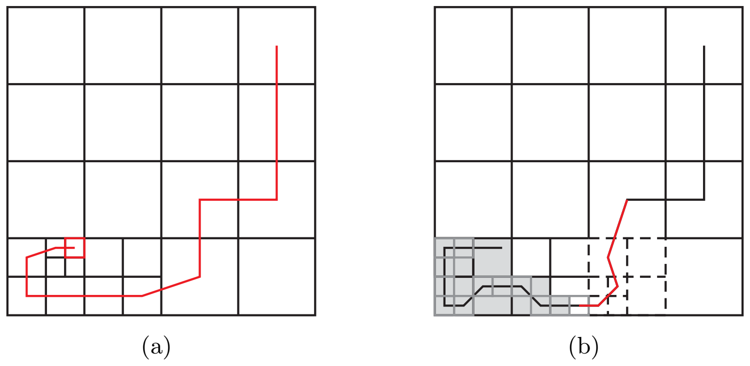


Figure 6: Schematic illustration of the local replanning algorithm. From [14].

applications on-line processing is not critical. For control applications, on the other hand, one needs to be able to solve certain equations in real-time or nearly real-time. Feedback control using video signals, for instance, requires fast and efficient segmentation of a sequence of images at 30 Hz or faster. We have developed an algorithm that implements an adaptive grid *along the interface* of the level set curve; this is in addition to existing level set implementations that generate adaptive grids only normal to the interface, such as narrow band methods. The idea is to use an adaptive grid whose resolution depends on the curvature of the contour. While the resulting grid is adaptively refined along the interface at the locations where the curvature is high, it remains coarse along the interface at the locations where the curvature is low. The grid also remains coarse in the region far away from the interface, thus saving a significant amount of computations. Numerical examples show that about 10 time steps often suffice to obtain high-quality segmentation if the curvature penalty is not high. The result was a significant speed-up of computations when applied to image segmentation problems. Details can be found in [15, 16].

Experimental Demonstration: A UAV platform has been developed that is used to implement the hierarchical, wavelet-based path-planning algorithm. The development of the *hardware and software was done completely in-house by Georgia Tech graduate and undergraduate students.* The overall architecture of the UAV system is shown in Fig. 7.

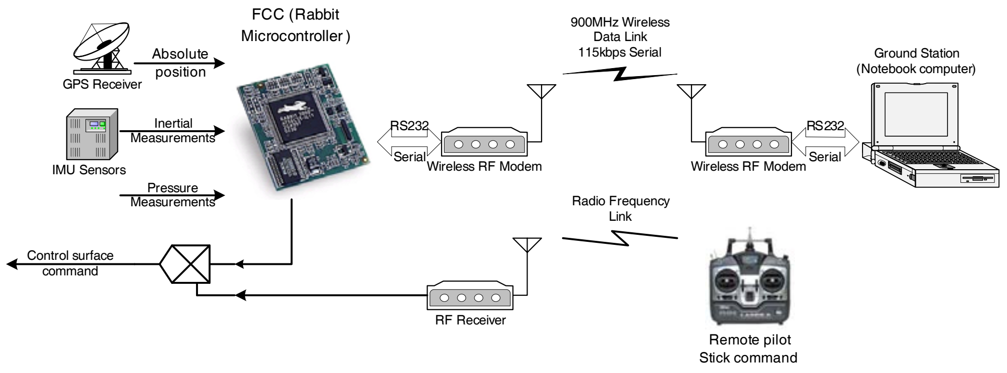


Figure 7: Hardware system configuration schematic of the UAV test bed. All hardware integration, electronics and associated software was developed in-house by Georgia Tech students. A hardware-in-the-loop simulation set-up was also developed to enable rapid testing of the control laws.

The main subsystems are the autopilot, the ground station, and the interconnection between the two. The on-board autopilot is equipped with a micro-controller, sensors and actuators, along with the communication devices that allow full functionality for autonomous control. The microcontroller (Rabbit RCM-3400 at 30 MHz, 512 KB RAM) provides data acquisition, processing, and communication with the ground station. It also runs the main control software. The on-board sensors include three-axis angular rate sensors, three-axis accelerometers, a three-axis magnetic compass, a GPS sensor, an engine RPM sensor, absolute and differential pressure sensors, battery voltage, fuel level and temperature sensors. Details can be found in [17] and [18].

A very detailed nonlinear 6-dof model of the platform was developed. The aerodynamic derivatives of the UAV airframe were computed and linearized models about trim positions were used for low level (PID) control design. We designed the navigation and guidance estimation filters that process the measurements from the on board sensors. In order to reduce the computational overheard, two cascades of complementary and Kalman filters have been developed; this reduces significantly the computational cost when compared to the implementation of a centralized, fully populated extended Kalman filter.

A hardware-in-the-loop (HIL) simulation environment was developed to support and validate the UAV autopilot hardware and software development. The HIL simulation incorporates a high-fidelity dynamic model that includes the sensor and actuator models, from the identified parameters from experiments. A user-friendly graphical interface that incorporates external stick commands and 3-D visualization of thevehicle's motion completes the simulation environment. The hardware-in-the-loop setup is an indispensable tool for rapid certification of both the avionics hardware and the control software, while performing simulated flight tests with minimal cost and effort.

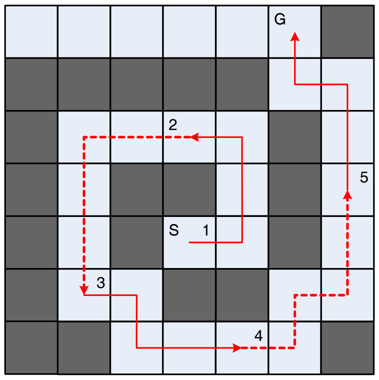
Owing to the limited computational resources of the UAV, we have followed an offline/online approach to generate smooth, reference paths, given a family of discrete optimal paths. In conjunction with a path representation by a finite sequence of square cells, the generated path is supposed to stay inside a feasible channel, while minimizing certain performance criteria. Constrained optimization problems are formulated subject to geometric (linear) constraints, as well as boundary conditions in order to generate a library of B-spline path templates. As an application to the vehicle motion planning, the path templates are incorporated to represent local segments of the entire path as geometrically smooth curves, which are then joined with one another to generate a reference path to be followed by a closed-loop tracking controller. The on-line path generation algorithm incorporates the path templates such that continuity and smoothness are preserved when switching from one template to another along the path. The approach is shown pictorially in Fig. 8 Combined with the $\mathcal{D}^*$-lite path planning algorithm, the proposed algorithm provides a complete solution to the obstacle-free path generation problem in a computationally efficient manner, suitable for real-time implementation. The approach is explained in detail in [22].

We have used this UAV platform to test the multiresolution path planning algorithms developed in this work. Details can be found in [19, 9, 18, 20, 21, 23].

References

[1] S. Jain, P. Tsiotras, and H.-M. Zhou, "Adaptive multiresolution mesh refinement for the solution of evolution PDEs," *SIAM Journal of Scientific Computing*, (accepted) 2008.

[2] S. Jain and P. Tsiotras, "Adaptive multiresolution mesh refinement for the solution of evolution pdes," in *46th IEEE Conference on Decision and Control*, (New Orleans, New Orleans LA), Dec. 12-14 2007.



Path #	Path words	Templates	Operations
1	`ENNW`	`ENNW`	-
2	`WWSSS`	`EENNN`	H, V
3	`ESEE`	`NENN`	H, D
4	`ENENN`	`ENENN`	-
5	`NNWN`	`NNEN`	V

Figure 8: Example incorporating the path templates on a complex path sequence. Five local path instances are connected to one another in order to reach the goal cell. The actual path words are recovered from the path templates with the corresponding symmetry operations of the horizontal (H), vertical (V), or diagonal (D) reflections.

[3] S. Jain, *Multiresolution Strategies for the Numerical Solution of Optimal Control Problems.* Ph.D. Thesis, Aerospace Engineering,, Georgia Institute of Technology, Atlanta, GA, May 2008.

[4] S. Jain and P. Tsiotras, "Multiresolution-based direct trajectory optimization," in *46th IEEE Conference on Decision and Control*, (New Orleans, LA), pp. 5991–5996, Dec. 12–14 2007.

[5] S. Jain and P. Tsiotras, "Multiresolution-based direct trajectory optimization," *Journal of Guidance Control, and Dynamics*, vol. 31, no. 5, pp. 1424–1436, 2008.

[6] S. Jain and P. Tsiotras, "Sequential multiresolution trajectory optimization for moving targets," in *AIAA Guidance, Navigation, and Control Conference*, (Honolulu, HI), Aug. 18-21 2008. AIAA Paper 2008-6980.

[7] S. Jain and P. Tsiotras, "Sequential multiresolution trajectory optimization schemes for problems with moving targets," *Journal of Guidance, Control, and Dynamics*, 2008. (under review).

[8] R. Cowlagi and P. Tsiotras, "Beyond quadtrees: Cell decomposition for path planning using the wavelet transform," in *46th IEEE Conference on Decision and Control*, (Orleans, LA), pp. 1392–1397, Dec. 12–14 2007.

[9] D. Jung and P. Tsiotras, "Multiresolution on-line path planning for small unmanned aerial vehicles," in *American Control Conference*, (Seattle, WA), pp. 2744–2749, June 11-13 2008.

[10] P. Tsiotras, "Multiresolution hierarchical path-planning for small UAVs,," in *European Control Conference*, (Kos, Greece), July 2-5 2007.

[11] P. Tsiotras and E. Bakolas, "A hierarchical on-line path-planning scheme using wavelets," in *European Control Conference*, (Kos, Greece), pp. 2806–2812, July 2–5 2007.

[12] E. Bakolas, "A hierarchical on-line path planning scheme using wavelets," M.S. Thesis, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, March 2006.

[13] E. Bakolas and P. Tsiotras, "Multiresolution path planning via sector decompositions compatible to on-board sensor data," in *AIAA Guidance, Navigation, and Control Conference*, (Honolulu, HI), 2008. AIAA Paper 2008-7238.

[14] R. Cowlagi and P. Tsiotras, "Multiresolution path planning with wavelets: A local replanning approach," in *American Control Conference*, (Seattle, WA), pp. 1220–1225, June 11-13 2008.

[15] B. Kim, P. Tsiotras, and H.-M. Zhou, "Image segmentation using a fast multi-resolution level set method," Technical Report, Georgia Institute of Technology, Atlanta, GA, November 2006.

[16] B. Kim and P. Tsiotras, "Image segmentation on cell-center sampled quadtree and octree grids," in *Wavelet Applications in Industrial Processing VI*, (San Jose, CA), Jan. 18–22 2009.

[17] D. Jung, D. Zhou, R. Fink, T. Williams, J. Moshe, and P. Tsiotras, "Design and development of a low-cost test-bed for undergraduate education in UAVs," in *44th IEEE Conference on Decision and Control/European Control Conference ECC 2005*, (Seville, Spain), pp. 2739–2744, Dec. 12–15 2005.

[18] D. Jung and P. Tsiotras, "Inertial attitude and position reference system development for a small UAV," in *AIAA Infotech at Aerospace*, (Rohnert Park, CA), May 7–10 2007. AIAA Paper 07-2763.

[19] D. Jung, *Hierarchical Path Planning and Control of a Small Fixed-Wing UAV: Theory and Experimental Validation.* Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, Dec. 2007.

[20] D. Jung and P. Tsiotras, "Modelling and hardware-in-the-loop simulation for a small unmanned aerial vehicle," in *AIAA Infotech at Aerospace*, (Rohnert Park, CA), May 7–10 2007. AIAA Paper 07-2768.

[21] D. Jung, J. Ratti, and P. Tsiotras, "Real-time implementation and validation of a new hierarchical path planning scheme for UAVs via hardware-in-the-loop simulation," in *International Symposium of Unmanned Aerial Vehicles (UAV'08)*, (Orlando, FL), June June 23-24 2008.

[22] D. Jung and P. Tsiotras, "On-line path generation for small unmanned aerial vehicles using B-spline path templates," in *AIAA Guidance, Navigation, and Control Conference*, (Honolulu, HI), Aug. 18-21 2008. AIAA Paper 2008-7135.

[23] D. Jung and P. Tsiotras, "Bank-to-turn control for a small UAV using backstepping and parameter adaptation," in *17th IFAC World Congress*, (Seoul, South Korea), pp. 4406–4411, July 6-11 2008.

Research and Education Activities

Research Activities

The overall objective of this research was to apply multi-resolution and wavelet-based techniques to solve the following problems:

- Development of adaptive gridding in the time-domain in order to enhance the numerical accuracy and execution speed of trajectory generators for dynamical systems. Optimal control problems can be cast as parameter optimization (nonlinear programming) problems via direct transcription and then solved on the grid points. The computational complexity and accuracy of the solution depends on the number and distribution of these grid points. By developing adaptive gridding algorithms we will be able to increase speed and enhance robustness.

- Development of multiresolution methods for the solution of Hamilton-Jacobi pdes for solving optimal feedback control problems. Often the solution of a pde of the Hamilton-Jacobi type is not smooth, even if the given initial condition is smooth. To capture the discontinuity in the solution we need to use a high resolution grid. Hence in order to solve these problems in a computationally efficient way, the computational grid should adapt to reflect local changes in the solution.

- Development of multiresolution decomposition and reconstructions of the environment to reduce the computational complexity for optimal path-generation in the presence of obstacles. For the case of UAVs in particular, one requires increased automation, beyond the traditional control systems. The autonomous operation of UAVs requires both trajectory design (planning) and trajectory tracking (control) tasks to be completely automated. If these tasks are to be computed on-line *exactly* this seems to be a unsurmountable undertaking using current computer technology. On the other hand, it is rarely necessary to be able to compute the route of an aircraft to within seconds or meters several hours into the future. High accuracy is desirable (and possible) only for the immediate future. Of course, it is not enough to only address the near-term problem; the overall mission objectives should be accommodated, although further out segments of the route may be optimized with progressively lesser accuracy. A multi-scale algorithmic hierarchy suitable for on-line implementation thus emerges. Namely, computation with high accuracy locally, around the current position and for the immediate future, and computation of decreased accuracy elsewhere and for longer time horizons.

- Application of multiresolution strategies on level set (LS) methods for solving Hamilton-Jacobi equations for optimal feedback control and image segmentation problems. The most significant advantage of the level set formulation is that it can easily handle the topological changes of the interfaces, such as splitting, merging, appearing and vanishing. Level set methods have been widely used in many applications including computer vision, image processing, material science, fluid dynamics, control, and medical science. Level set methods can be used to track the boundaries of obstacles in the environment. The level set approach can be used to deal with emerging new obstacles or with changes of existing obstacles as the vehicle moves along its path. The accuracy and speed of current LS methods can be improved by using multiresolution methods (e.g., adaptive gridding) along the interface of the zero level set, as well as normal to the interface (traditional narrow band method).

- Development of a small UAV platform in order to experimentally validate the proposed multi-resolution approaches, by performing autonomous navigation and control. The main challenge here is the (intentionally) limited computational resources on board this small UAV. The autopilot microprocessor memory and speed requirement do not allow the implementation of algorithms that do not scale well with the number of grid points or the number of constraints. This notion of "hardware-driven" control algorithm development is central throughout this work.

Education Activities

One of the students (E. Bakolas), who was partially supported from this project, completed the requirements for an MS degree in Aerospace Engineering in the Spring of 2007. Mr. Bakolas successfully defended his MS thesis in front of a committee composed of the PI, Prof. Eric Feron (AE) and Prof. Magnus Egerstedt (ECE) on March 26, 2007. The title of his thesis was "A Hierarchical On-Line Path Planning Scheme using Wavelets." Mr. Bakolas is currently enrolled in the Ph.D. program at Georgia Tech.

A second student (S. Jain), who was fully supported from this NSF project, completed his Ph.D. degree in Aerospace Engineering in the Spring of 2008. Mr. Jain successfully defended his Ph.D. dissertation on March 7, 2008 in front of a committee composed of the PI, Prof. H.-M. Zhou (Math), Prof. JVR Prasad (AE), Prof. Anthony Calise (AE), Prof. Ryan Russell (AE), and Prof. Magnus Egerstedt (ECE). The title of this doctoral dissertation was "Multiresolution Strategies for the Numerical Solution of Optimal Control Problems."

A third student (D. Jung), who was fully supported from this NSF project, completed his Ph.D. degree in Aerospace Engineering in the Fall of 2008. Mr. Jung successfully defended his Ph.D. dissertation on October 30, 2007 in front of a committee composed of the PI, Prof. G. Vachtsevanos (ECE), Prof. Eric Johnson (AE), Dr. Eric Corban (GST), and Prof. Eric Feron (AE). The title of this doctoral dissertation was "Hierarchical Path Planning and Control of a Small Fixed-wing UAV: Theory and Experimental Validation."

Six undergraduate students were involved with the design of the UAV platform, the ground station, and the UAV autopilot.