Dijkstra-like Ordered Upwind Methods for Solving Static Hamilton-Jacobi Equations

by

Ken Alton

Bachelor of Science, University of Calgary, 2000

Master of Science, University of British Columbia, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

May 2010

Abstract

The solution of a static Hamilton-Jacobi Partial Differential Equation (HJ PDE) can be used to determine the change of shape in a surface for etching/deposition/lithography applications, to provide the first-arrival time of a wavefront emanating from a source for seismic applications, or to compute the minimal-time trajectory of a robot trying to reach a goal.

HJ PDEs are nonlinear so theory and methods for solving linear PDEs do not directly apply. An efficient way to approximate the solution is to emulate the causal property of this class of HJ PDE: the solution at a particular point only depends on values backwards along the characteristic that passes through that point and solution values always increase along characteristics. In our discretization of the HJ PDE we enforce an analogous causal property, that the solution value at a grid node may only depend on the values of nodes in its numerical stencil which are smaller. This causal property is related but not the same thing as an upwinding property of schemes for time dependent problems. The solution to such a discretized system of equations can be efficiently computed using a Dijkstra-like method in a single pass through the grid nodes in order of nondecreasing value.

We develop two Dijkstra-like methods for solving two subclasses of static HJ PDEs. The first method is an extension of the Fast Marching Method for isotropic Eikonal equations and it can be used to solve a class of axis-aligned anisotropic HJ PDEs on an orthogonal grid. The second method solves general convex static HJ PDEs on simplicial grids by computing stencils for a causal discretization in an initial pass through the grid nodes, and then solving the discretization in a second Dijkstra-like pass through the nodes.

This method is suitable for computing solutions on highly nonuniform grids, which may be useful for extending it to an error-control method based on adaptive grid refinement.

Table of Contents

Abstract . . . . . **ii**

Table of Contents . . . . . **iv**

List of Tables . . . . . **vii**

List of Figures . . . . . **viii**

Acknowledgments . . . . . **x**

1 Introduction **1**

1.1 Motivation 2

1.2 Simple Example Problem: Airplane Escape 3

1.3 Static Hamilton-Jacobi Equation 12

1.4 Ordered Upwind Methods 14

1.5 Contributions 15

1.6 Outline 16

2 Related Work **18**

2.1 Viscosity Solutions of Hamilton-Jacobi Equations 18

2.2 Numerical Methods for Hamilton-Jacobi Equations 19

2.3 Dynamic Programming and Optimal Control 21

2.4 Methods for Static Hamilton-Jacobi Equations 21

2.4.1 Fast Marching Methods 21

2.4.2 Ordered Upwind Methods 23

2.4.3 Sweeping Methods 24

2.5 Applications of Static Hamilton-Jacobi Equations 25

3 Framework 28
3.1 Discretization 28
3.2 Convergence 29
3.2.1 Consistency, Monotonicity, and Stability 30
3.2.2 Viscosity Solution and Proof of Convergence 31
3.3 Algorithm 33
3.3.1 Dijkstra-like Methods 34
3.3.2 Solution of Discretization 36

4 FMM for Axis-Aligned HJ PDEs 39
4.1 Introduction 39
4.1.1 The Fast Marching Method 41
4.2 Class of Hamiltonians 41
4.2.1 Connection to Osher's criterion 42
4.2.2 Example Hamiltonians 46
4.3 Discretization 51
4.3.1 Unique Update 53
4.3.2 Causality 55
4.3.3 Monotonicity 55
4.3.4 Consistency 56
4.3.5 Stability 58
4.4 Efficient Implementation of Update 61
4.4.1 Symmetry Node Elimination 62
4.4.2 Causality Node Elimination 65
4.4.3 Solution Elimination 66
4.5 Analytic Solutions 67
4.5.1 Update for $p = 1$ 68
4.5.2 Update for $p = 2$ 69
4.5.3 Update for $p = \infty$ 70
4.6 Experiments 71
4.6.1 Numerical Convergence Study 71

4.6.2 Asymmetric Anisotropic Problem 71
4.6.3 Anelliptic Elastic Wave Propagation 73
4.6.4 Two Robots 75

5 OUM with Monotone Node Acceptance for Convex HJ PDEs 77
5.1 Introduction 77
5.1.1 Dijkstra-like Methods 79
5.1.2 Related Work 80
5.2 Discretization 80
5.2.1 Monotonicity 82
5.2.2 Consistency 83
5.2.3 Unique Solution 84
5.2.4 Equivalence of Discretizations 86
5.3 Causality 92
5.3.1 Negative-gradient-acuteness 94
5.3.2 Anisotropy-angle-boundedness 95
5.3.3 Distance-ratio-boundedness 98
5.4 Algorithm 101
5.4.1 Computing the Update Set 104
5.4.2 Convergence 110
5.4.3 Complexity 111
5.4.4 Update Region 114
5.5 Experiments 116
5.5.1 Numerical Convergence Study 116
5.5.2 Nonuniform Grid 118
5.5.3 Seismic Imaging 121
5.5.4 Robot Navigation with Wind and Obstacles 122

6 Conclusion 126
6.1 Future Work 128

Bibliography 130

List of Tables

4.1 Numerical evidence of convergence of FMM for axis-aligned problems 72

5.1 Summary of MAOUM symbols 102
5.2 Errors for MAOUM on problem with elliptical speed function 117
5.3 Errors for MAOUM and AFOUM on problem with rectangular speed function 120

List of Figures

1.1 Two-robot coordinated optimal navigation problem 3
1.2 Navigating a robot with wind and obstacles 4
1.3 Viscosity solutions for the airplane escape problem 6
1.4 Approximate solution for the airplane escape problem 8
1.5 Orthogonal grids 14
1.6 Contours of first-arrival times of a seismic wave 16

2.1 Motion of two robots in a 2D world 26
2.2 Robot arms cooperating to optimally transport cargo 27

4.1 Contour plots of p-norms 47
4.2 Contour plots of transformed p-norms 49
4.3 Contour plots of mixed p-norm and asymmetric norm-like function 50
4.4 Neighborhood of x with $d = 2$ 52
4.5 Solution for asymmetric anisotropic problem 73
4.6 Anelliptic Hamiltonian and solution 74

5.1 Symbols used in the definition of δ-negative-gradient-acuteness 94
5.2 Symbols used in the definition of δ-anisotropy-angle-boundedness 96
5.3 Symbols used in the definition of distance-ratio-boundedness 99
5.4 Status of algorithm computing the update node set 104
5.5 (Continued) Status of algorithm computing the update node set 105
5.6 A sequence of uniformly-refined Maubach grids 117
5.7 Sequence of nonuniformly-refined Maubach grids 118

5.8 Error versus updates for rectangular speed function 119
5.9 Contours of first-arrival times of a seismic wave 122
5.10 Navigating a robot with wind and obstacles 125

Acknowledgments

I would like to thank my supervisor, Ian Mitchell, for his collaboration and support. Much of the progress I have made in developing the ideas in this thesis has been with the help of Ian. My supervisory committee, including Uri Ascher and Adam Oberman, gave me a great deal of insightful and constructive feedback, for which I am grateful. I would also like to thank Alex Vladimirsky who has been both a critic and mentor, and who no doubt has made my thesis much stronger.

I appreciate the time and effort spent by the examining committee to critique my thesis work and provide many useful suggestions and corrections. Thank you to the chair, Zinovy Reichstein, and the examiners, Robert Bridson and Philip Loewen.

I am greatly indebted to my parents, Val and Norm Alton, for their unwavering love and support throughout my studies. I doubt that I would have even applied for grad school without the appreciation for education they have instilled in me. I am also thankful to my aunt and uncle, Liz Watts and Warren Murschell, for being my family in Vancouver and having me over for Sunday dinners. My wife, Luiza, has been incredibly supportive for the four years since we met. She has cheered me on when things were going well and helped me bounce back when they were not. Finally, I would like to thank her parents for being very helpful and providing us with a second home in Vancouver.

This work has been supported by a grant from the National Science and Engineering Research Council of Canada.

Chapter 1

Introduction

The viscosity solution of a static Hamilton-Jacobi Partial Differential Equation (HJ PDE) has at least two illuminating interpretations. One of these is that the solution encodes for each location the first arrival time of a wavefront emanating from a source. Another interpretation is that the solution encodes for each starting location the minimal time for some dynamic system to arrive at a goal location. The connection between these two interpretations is hinted at in a version of Huygen's Principle (a minor modification of the definition in [31]):

> All points on a wavefront serve as point sources of secondary wavelets. After a short time the new position of the wavefront will be that of the surface tangent to those secondary wavelets.

A wavefront propagates by trying all possible directions from all possible point sources along the front. The sources and directions that push the wavefront forward the fastest are used to determine the future position of the wavefront. Over time, the wavefront propagates by trying all possible trajectories from the original source, and the position of the wavefront at a particular time is determined by all the places that can be reached in that time by the most efficient trajectories. In the context of light wave propagation, this is Fermat's Principal: the path taken between two points by a ray of light is the path that can be traversed in the least time. The

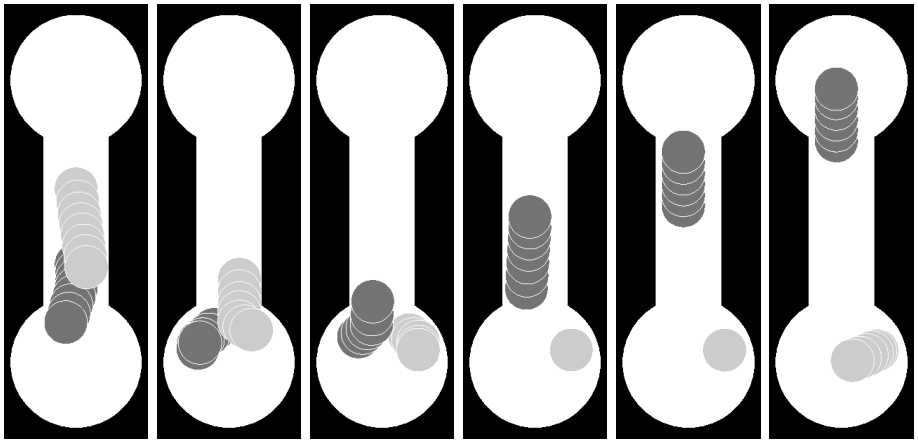
mathematical connection between these two interpretations is examined in [60].

1.1 Motivation

The two interpretations of the viscosity solution of a static HJ PDE suggest two applications. First, one may wish to know for a wavefront traveling in a medium when the wavefront first reaches various locations. Second, one may wish to compute optimal trajectories for a system to reach a goal. We solve both types of problems but we are particularly motivated by the latter, which are a type of optimal control problem. For example, one might ask: in what direction should I go in order to reach a goal location in the least possible time? An infinite number of such questions in sequence results in the larger question: what trajectory should I follow in order to reach the goal in the least possible time? One way of solving these optimal control problems is to formulate and solve an appropriate static HJ PDE.

As mentioned, the viscosity solution is a function that specifies for each possible starting location the minimal time to reach the goal. The viscosity solution is smallest in value at the goal and grows as one moves further out from the goal. From a particular location, to reach the goal as fast as possible one should move in the direction that allows the fastest possible descent of the viscosity solution. This is related to, but not the same thing as, moving in the direction of steepest descent, because one may be capable of moving faster in some directions than in others. If, from a particular starting location, one continually chooses to move in the direction that allows the fastest descent, one will eventually reach the goal via a trajectory of minimal time. The viscosity solution can be used to determine the optimal direction to move for the current state forming a feedback controller that results in optimal trajectories from any initial state.

This discussion is intended to provide an intuition of how viscosity solutions can be used to determine optimal trajectories. For a detailed mathematical analysis of how the viscosity solution of static HJ PDE is the solution to a related optimal control problem see [7] and [25, Chapters 3 and



(a) (b) (c) (d) (e) (f)

Figure 1.1: Two-robot coordinated optimal navigation problem. The joint goal is for the dark-colored robot to reach the center of the upper bulb and light-colored robot to reach the center of the lower bulb. Black indicates an obstacle region. The sequence shows the robots achieving their joint goal without collision from a particular initial state. The solution of an appropriate static HJ PDE allows quick determination of the optimal collision-free trajectories for both robots from any initial condition [1].

10]. Section 1.2 introduces this method of optimal control using a simple *plane escape* problem. We solve a two-robot coordinated optimal navigation problem in Section 4.6.4 by approximating the viscosity solution to a static HJ PDE on a uniform orthogonal grid. A resulting trajectory of the two robots is illustrated in Figure 1.1. We also solve a optimal robot navigation problem in Section 5.5.4 with wind and obtacles on a nonuniform grid with refinement around obstacles and the goal location. The grid, computed viscosity solution, and several resulting trajectories are shown in Figure 1.2.

1.2 Simple Example Problem: Airplane Escape

The following example problem can be solved more simply than the methods of this thesis, mainly because it involves traversal speeds that are homoge-

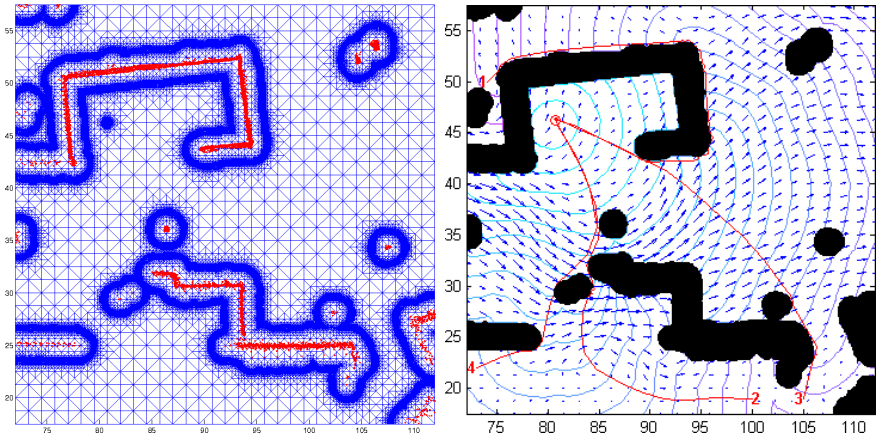


Figure 1.2: The problem of optimally-navigating a robot to a goal through wind and obstacles. The left shows the laser-rangefinder data of the obstacles and the grid refined in a band around the collision set and the goal. The right shows the collision set in solid black. The right includes the wind vector field, the contours of the computed viscosity solution, and four optimal trajectories from different starting locations to the goal.

neous throughout the domain. However, the example serves as an introduction to some notation, concepts, and issues involved in solving more complicated but related problems.

Imagine that a passenger is in an airplane that just crashed and he wants to get out as quickly as possible. There are two exits on the airplane: a rear exit and a front exit. The airplane crashed nose down so he can move faster towards the nose of the airplane than towards the tail. One of the exits may be partially blocked by debris which will take some extra time to clear. How does he decide which exit to go towards?

We let x be the distance from the tail along the longitudinal axis of the airplane. We use $x_t = 0$, x_r, x_f, and x_n to denote the x-coordinate of the tail, rear exit, front exit, and nose respectively. Let $f_t > 0$ and $f_n > 0$ be the speed at which the passenger can move towards the tail and nose of the airplane, respectively. Let g_r and g_f be the amount of time required just to open and get through the rear and front exit, respectively. The total time

$t_r(x)$ required for a passenger at position $x \in (x_t, x_n)$ to escape through the rear exit is

$$t_r(x) = g_r + \begin{cases} \frac{x - x_r}{f_t}, & x \geq x_r \\ \frac{x_r - x}{f_n}, & \text{otherwise} \end{cases} \tag{1.1}$$

and the total time $t_f(x)$ required for a passenger at position $x \in (x_t, x_n)$ to escape through the front exit is

$$t_f(x) = g_f + \begin{cases} \frac{x - x_f}{f_t}, & x \geq x_f \\ \frac{x_f - x}{f_n}, & \text{otherwise} \end{cases} \tag{1.2}$$

If $t_r < t_f$ he should go for the rear exit, if $t_f < t_r$ he should go for the front exit, and if $t_f = t_r$ either exit is fine.

This problem is simple to solve but for other problems we require more general methods. A more complicated version might have a speed $f(x, a)$ that depends not only on the direction of travel $a \in \{-1, +1\}$ but also on the current position x. Also $x \in \mathbb{R}^d$ might be a position in a d-dimensional space and $a \in \mathbb{R}^d$ such that $\|a\|_2 = 1$ may be a direction in that d-dimensional space.

Consider the function $u : (x_t, x_n) \to \mathbb{R}$ that gives the time required to exit the airplane for each passenger position $x \in (x_t, x_n)$, which is the minimum of the time required to escape through the front or rear exits. Using (1.1) and (1.2), we obtain

$$u(x) = \min(t_r(x), t_f(x)). \tag{1.3}$$

The graph of u for an instance of the problem is shown in the top-left of Figure 1.3. We can verify that function u satisfies the following differential equation for almost all x:

$$\max_{a \in \{-1,+1\}} -\frac{du(x)}{dx} a f(x, a) = 1, \qquad x \in \Omega \tag{1.4a}$$

$$u(x) = g(x), \quad x \in \partial\Omega, \tag{1.4b}$$

where $\Omega = (x_t, x_r) \cup (x_r, x_f) \cup (x_f, x_n)$ is the domain, $\partial\Omega = \{x_r, x_f\}$ is the domain boundary,

$$f(x, a) = f(a) = \begin{cases} f_t, & \text{if } a = -1, \\ f_n, & \text{if } a = +1, \end{cases}$$

and

$$g(x) = \begin{cases} g_r, & \text{if } x = x_r, \\ g_f, & \text{if } x = x_f. \end{cases}$$

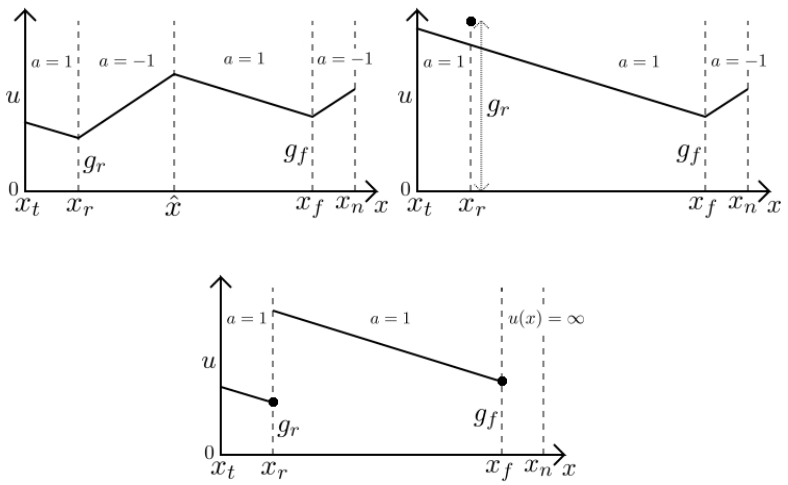


Figure 1.3: Viscosity solutions for instances of the airplane escape problem. We do not precisely specify the problem parameters for these instances because doing so is not necessary for the purpose of our illustration. The top-left shows the viscosity solution u to (1.4) and specifies the optimal control for the four regions separated by x_r, $\hat{x}$, and x_f. For this instance the problem parameters are such that the boundary conditions are compatible and the system is small-time-controllable (i.e., $f_t > 0$ and $f_n > 0$), so there is a continuous solution that satisfies the boundary conditions. The top-right plots the solution u to (1.3) for the case where the boundary conditions are incompatible because $u(x_r) < g_r$. The bottom plots the solution u to (1.3) for the case where the system is not small-time-controllable because $f_t = 0$.

We say that the differential equation holds for *almost* all x because there

may be some $\hat{x} \in \Omega$ at which the derivative $du(\hat{x})/dx$ does not exist. In the top-left of Figure 1.3 such an $\hat{x}$ can be seen, at which $u(\hat{x})$ forms the apex of the graph of u. The point $\hat{x}$ can be considered a decision boundary. For $x \in (x_r, \hat{x})$ the optimal action is $a = -1$ (move backward toward x_r), For $x \in (\hat{x}, x_f)$ the optimal action is $a = +1$ (move forward toward x_f), and if $x = \hat{x}$, the optimal action is $a \in \{-1, +1\}$ (move backward or forward). At the point $\hat{x}$ the time to escape through either exit is equal: $t_r(\hat{x}) = t_f(\hat{x})$. Because the derivative $du(\hat{x})/dx$ does not exist, there is no classical solution to (1.4). The u defined in (1.3) happens to be the (unique) weak solution of (1.4) called a viscosity solution [19]. The viscosity solution is defined in Section 3.2.2 but the theory of such solutions is not the focus of this thesis.

The boundary $\partial\Omega$ and the boundary condition (1.4b) require some special attention. Firstly, even though x_t and x_n form part of the physical boundary of Ω, we do not include x_t or x_n in $\partial\Omega$ and as a result do not enforce a boundary condition on x_t or x_n. We define u on (x_t, x_n) instead of $\mathbb{R}$ based on the dimensions of the airplane. In other problems, it may be appropriate to enforce boundary conditions on the exterior boundary instead of (or in addition to) the interior boundary of the domain.

Secondly, there is a potential issue with satisfying boundary conditions $u = g$ on $\partial\Omega$. For example, consider a problem instance for which, starting from the rear exit, the passenger can escape through the front exit faster than through the rear exit (perhaps the rear door is jammed shut):

$$u(x_r) = t_f(x_r) < t_r(x_r) = g_r.$$

Such an example is shown in Figure 1.3 top-right. In this case, either the boundary condition is not satisfied at x_r (i.e., $u(x_r) < g_r$); or we enforce the boundary condition, (1.3) is no longer satisfied, and the function u becomes discontinuous at x_r. When such an issue occurs, we say that the boundary conditions are *incompatible*. Since (1.3) is the correct solution to the physical problem, we want to ensure it holds. One approach is to assume compatible boundary conditions [11], so that the PDE and its boundary conditions can be satisfied by a continuous u. An alternative approach is

to use a modified definition of viscosity solution at the boundary, which is satisfied continuously [8], and corresponds to the correct solution for most physical problems. For example, the solution to (1.3) would satisfy the modified definition. To avoid complication we take the former approach and assume compatible boundary conditions, while acknowledging that the latter approach can sensibly handle incompatible boundary conditions for many practical problems.

Another issue with the solution may arise if $f_t = 0$ or $f_n = 0$. In this case u is discontinuous at x_r and x_f and $u(x) = \infty$ for some x (for example, see Figure 1.3 bottom). Such conditions complicate the analysis and approximation of viscosity solutions, and so we assume that $f_t > 0$ and $f_n > 0$. Such an assumption is called *small-time-controllability*.

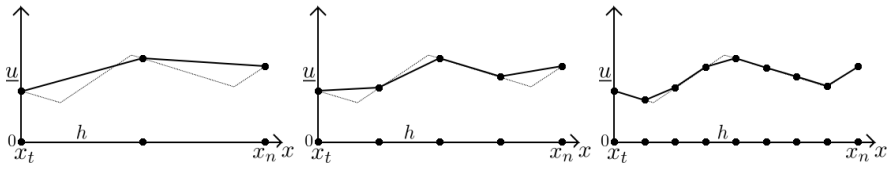


Figure 1.4: Approximate solution for an instance of the airplane escape problem. The problem parameters for this instance are identical to those for the top-left of Figure 1.3. The three figures show a progression of solutions $\underline{u}$ to (1.5) (i.e., approximations to the viscosity solution) for increasingly fine grids: $N \in \{3, 5, 9\}$. On these three figures u is depicted by a light thin dotted line to indicate how well $\underline{u}$ approximates u.

For this problem we have an exact closed form solution (1.3), but for most problems we are forced to approximate the solution on a grid using a numerical method. We demonstrate by constructing on a grid a discrete system of equations that is the analogue to (1.4). The solution $\underline{u}$ to the discrete system is used to approximate the solution u to (1.4) as shown in Figure 1.4. We let $\mathcal{X}$ be the set of grid nodes and $N = |\mathcal{X}|$ be the number of grid nodes. The spacing between grid nodes is $h = (x_n - x_t)/(N - 1)$.

Thus the set of grid nodes is

$$\mathcal{X} = \{x_t, x_t + h, x_t + 2h, \ldots, x_n - 2h, x_n - h, x_n\}.$$

The discrete boundary $\underline{\partial\Omega}$ contains any node $x \in \mathcal{X}$ that is less than h away from either x_r or x_f (so there may be as many as four boundary nodes):

$$\underline{\partial\Omega} = \{x \in \mathcal{X} \mid \min(|x - x_r|, |x - x_f|) < h\}.$$

For problems with dimension $d > 1$, $\underline{\partial\Omega}$ may be a discrete representation of a continuous boundary (see Figure 1.5). The set $\underline{\Omega} = \mathcal{X} \setminus \underline{\partial\Omega}$ is a discrete representation of the domain Ω.

The discrete analogue of (1.4) is

$$\max_{a \in \{-1,+1\}} \frac{\underline{u}(x) - \underline{u}(x + ah)}{h} f(x, a) = 1, \qquad x \in \underline{\Omega} \tag{1.5a}$$

$$\underline{u}(x) = u(x), \quad x \in \underline{\partial\Omega}. \tag{1.5b}$$

Note that (1.5b) defines the boundary conditions to be the exact solution u. For most problems we must approximate the boundary condition in (1.5b). The discretization results in a system of nonlinear equations which we must solve, one equation (1.5a) for each node $x \in \underline{\Omega}$. Luckily, this system is structured in such a way that it can be efficiently solved. We can solve (1.5a) for the node value $\underline{u}(x)$ based on the values of neighboring nodes:

$$\underline{u}(x) = \min_{a \in \{-1,+1\}} \left(\underline{u}(x + ah) + \frac{h}{f(x, a)} \right). \tag{1.6}$$

Note that $\underline{u}(x) > \underline{u}(x + ah)$, for all $x \in \underline{\Omega}$ for some $a \in \{-1, +1\}$. Also, if $\underline{u}(x) \leq \underline{u}(x + ah)$, $\underline{u}(x)$ is independent of $\underline{u}(x + ah)$, a property of the discrete equation we call *causality*. Nodes may only depend on neighboring nodes with smaller values for their own values. The information that determines the solution propagates between neighboring nodes from smaller-valued nodes to larger-valued nodes.

We describe three different algorithms that can be used to solve (1.5).

We let $\underline{v} : \mathcal{X} \to \mathbb{R}$ specify grid node values which are updated in the course of algorithm execution. All three algorithms begin by initializing $\underline{v}(x) \leftarrow u(x)$ for $x \in \underline{\partial\Omega}$, and $\underline{v}(x) \leftarrow \infty$ for $x \in \underline{\Omega}$. As each algorithm progresses, a node value $\underline{v}(x)$ is updated based on (1.6):

$$\underline{v}(x) \leftarrow \min_{a \in \{-1,+1\}} \left(\underline{v}(x + ah) + \frac{h}{f(x,a)} \right). \tag{1.7}$$

When each algorithm terminates, $\underline{v} = \underline{u}$ is the solution to (1.5).

The first algorithm is Gauss-Seidel iteration. It repeatedly visits all nodes in $\underline{\Omega}$ in a fixed order. Each time a node x is visited (1.7) is used to update $\underline{v}(x)$. The algorithm terminates when there is no change in $\underline{v}$ for an entire pass through $\mathcal{X}$. After $\mathcal{O}(N^2)$ updates (1.7), the algorithm terminates with $\underline{v} = \underline{u}$. There are $\mathcal{O}(N^2)$ updates because if the solution information flows counter to the fixed node order, that information can propagate to at most to one new node per pass through all N nodes.

A more efficient algorithm is a sweeping method. It performs two sweeps through the nodes in $\underline{\Omega}$: the first in left-to-right order and the second in right-to-left order. Each time a node x is visited (1.7) is used to update $\underline{v}(x)$. After two sweeps and $\mathcal{O}(N)$ node value updates, the algorithm terminates with $\underline{v} = \underline{u}$. There are at most $2N$ updates because solution information must flow uninterrupted to the left or right of each boundary node; thus, once the algorithm has performed sweeps in both directions the solution information has been propagated to all nodes.

Another efficient algorithm is a Dijkstra-like method (see Algorithm 1). It keeps a set $\mathcal{H}$ which is initialized to $\mathcal{X}$. The node $x \in \mathcal{H}$ with minimal $\underline{v}(x)$ is removed from $\mathcal{H}$ and its neighbors are updated using (1.7). The algorithm terminates after all nodes in $\mathcal{H}$ have been removed with $\underline{v} = \underline{u}$. It performs $\mathcal{O}(N)$ node value updates.

We note that the latter two methods are much more efficient than simple Gauss-Seidel iteration. This advantage also holds for more difficult practical problems. Sweeping and Dijkstra-like methods are compared in Section 2.4. In this thesis we use Dijkstra-like methods, which are described in detail in Section 3.3. It is important to note that Dijkstra-like methods can only be

used if the discrete equation is causal.

One can verify that the unique solution to (1.5) is actually $\underline{u}(x) = u(x)$ for $x \in \mathcal{X}$ for this example, as shown in Figure 1.3. In fact, the error in the approximation of u by $\underline{u}$ shown in the figure is solely due to the linear approximation of $\underline{u}$ between grid nodes. In typical problems additional errors would be introduced by approximation of the boundary condition, the inhomogeneous speed function $f(x, a)$, and the gradient of the solution.

We require the approximate solution $\underline{u}$ to converge to u as h approaches zero. This convergence appears to occur in Figure 1.3 for this problem. We examine convergence in detail in Section 3.2, but here we note that certain properties of the discrete equation (1.5a) are sufficient for convergence: namely, consistency, montonicity, and stability [8].

Recall that the original problem was to decide which direction to travel to escape the airplane in minimal time from a particular location x. An optimal direction of travel $a^*(x)$ for $x \in \Omega$ is given by

$$a^*(x) \in \underset{a \in \{-1,+1\}}{\operatorname{argmax}} -\frac{du(x)}{dx} a f(x, a).$$

If $du(x)/dx$ exists and is negative going forward is optimal. If $du(x)/dx$ exists and is positive going backward is optimal. If $du(x)/dx$ does not exist (e.g., at $\hat{x}$) moving in either direction is optimal. The mapping $a^* : \Omega \to \{-1, +1\}$ forms a feedback controller (or policy) that results in optimal trajectories from any initial state. An optimal trajectory $\zeta : [0, T] \to \overline{\Omega}$ from some initial position $x_0 \in \Omega$ is then defined by the ordinary differential equation

$$\frac{d\zeta(t)}{dt} = a^*(\zeta(t)) f(\zeta(t), a^*(\zeta(t))),$$

where $\zeta(0) = x_0$ and $t = T$ is the minimum time at which $\zeta(t) \in \partial\Omega$ (i.e., the trajectory exits the domain Ω). In practice, we compute a direction of travel (and trajectory) using the approximate function $\underline{u}$ in place of u.

1.3 Static Hamilton-Jacobi Equation

We characterize the category of problems we wish to solve. The problem described in the above section is a simple example of this type. The Dirichlet problem for a static HJ PDE is to find a function u, such that

$$H(x, Du(x)) = 0, \qquad x \in \Omega \tag{1.8a}$$

$$u(x) = g(x), \quad x \in \partial\Omega, \tag{1.8b}$$

where $\Omega \subset \mathbb{R}^d$ is a Lipschitz domain,[1] $\partial\Omega$ is the domain's boundary, $H : \Omega \times \mathbb{R}^d \to \mathbb{R}$ is the Hamiltonian function, $Du(x)$ is the gradient of u at x, and $g : \partial\Omega \to \mathbb{R}$ specifies the Dirichlet boundary condition. Let $\overline{\Omega} = \Omega \cup \partial\Omega$ be the closure of Ω.

An optimal continuous control problem which attempts to minimize the time to reach the boundary leads to the following Hamiltonian H in (1.8a) [60]:

$$H(x, q) = \max_{a \in \mathcal{A}}[(-q \cdot a) f(x, a)] - 1, \tag{1.9}$$

where $\mathcal{A} = \{a \in \mathbb{R}^d \mid \|a\| = 1\}$ is the set of unit vector controls, $a \in \mathcal{A}$ is a control specifying direction of travel, $x \in \overline{\Omega}$ is a state, $f : \Omega \times \mathcal{A} \to \mathbb{R}^+$ is a continuous function providing the finite positive speed of travel from each state x in each direction a, and $g : \partial\Omega \to \mathbb{R}$ gives the exit time penalty at each boundary state x. Note that f is positive, which means that small-time-controllability is assumed.

For all $x \in \Omega$, let the speed profile

$$\mathcal{A}_f(x) = \{taf(x, a) \mid a \in \mathcal{A} \text{ and } t \in \mathbb{R} \text{ such that } 0 \leq t \leq 1\} \tag{1.10}$$

be a closed convex set.[2] Because f is positive, $\mathcal{A}_f(x)$ contains the origin in

[1] A Lipschitz domain is a open bounded set whose boundary can be thought of locally as the graph of a Lipschitz continuous function

[2] Results based on a convex speed profile apply without loss of generality to nonconvex speed profiles as long as measurable (and not continuous) controls are assumed. A nonconvex speed profile can be replaced by its convex hull because a control $\tilde{a}$ can be imitated by alternating between two controls for which $\tilde{a}$ lies on their geodesic.

its interior. In an isotropic problem, $f(x,a)$ is independent of a for all x, i.e., $\mathcal{A}_f(x)$ is a hypersphere with the origin at its center. In such a problem, the Hamiltonian H reduces to

$$H(x,q) = \|q\|_2 f(x) - 1 \tag{1.11}$$

and (1.8) becomes the Eikonal equation:

$$\|Du(x)\|_2 = \frac{1}{f(x)}, \quad x \in \Omega \tag{1.12a}$$

$$u(x) = g(x), \quad x \in \partial\Omega. \tag{1.12b}$$

In an anisotropic problem, $f(x,a)$ depends on a, i.e., $\mathcal{A}_f(x)$ is a closed non-spherical but convex set. Since not all Hamiltonians H fit the control-theoretic formulation, more generally, for an isotropic problem the set of q solving $H(x,q) = 0$ is the surface of a origin-centered hypersphere; otherwise, the problem is anisotropic. The method in Chapter 5 assumes H has the form (1.9) and $\mathcal{A}_f(x)$ is a closed convex set. For the method in Chapter 4, H satisfies several assumptions listed in Section 4.2.

In general, it is impossible to find a classical solution u to the static Hamilton-Jacobi PDE (1.8) where u is differentiable for all x. We seek instead the viscosity solution (see definition in Section 3.2.2), a unique weak solution which under the above assumptions is continuous and almost everywhere differentiable [19].

Since we typically cannot solve for the viscosity solution analytically, we compute an approximate solution $\underline{u}$ using a grid with nodes forming both a discretization $\underline{\Omega}$ of Ω, and a discretization $\underline{\partial\Omega}$ of $\partial\Omega$. For example, in Chapter 4 we use an orthogonal grid like that shown in Figure 1.5 and Chapter 5 we use a simplicial grid like that shown on the left side of Figure 1.2. We note that a simplicial grid is significantly more difficult to implement than an orthogonal grid but allows for nonuniform refinement in the grid to better resolve certain features of the solution including non-trivial domain boundaries. Limited nonuniformity in orthogonal grids can be used to better resolve boundaries as shown in Figure 1.5.

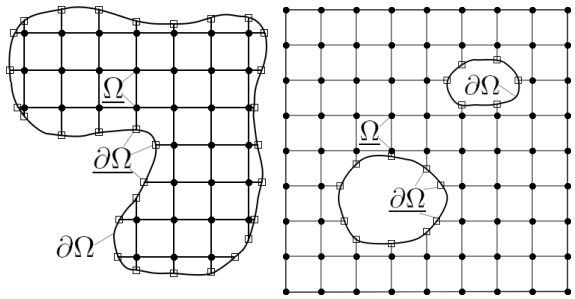


Figure 1.5: Orthogonal grids combining discretizations $\underline{\Omega}$ and $\underline{\partial\Omega}$. (a) Boundary conditions are given on the exterior of Ω. (b) Boundary conditions are given on the interior of Ω.

1.4 Ordered Upwind Methods

In order to use viscosity solutions of HJ PDEs for optimal control, we must develop efficient methods of computing them. An idea for such a method comes from their interpretation as the first arrival time of a propagating wavefront. A level contour of the solution is the position of the wavefront at a specific time. An intuitive and efficient method of approximating the solution on a grid is to compute the solution values at grid nodes in the order in which the wavefront passes through the grid nodes. The solution value at a particular grid node is based on values of neighboring grid nodes that are smaller, in the same way that the time at which the wavefront crosses any particular point is dependent on the earlier times the wavefront crosses nearby points in the direction from which it emanates. Dijkstra-like methods were developed in [55, 64] to approximate the solution to an isotropic static HJ PDE, also known as the Eikonal equation, in a single pass through the nodes of a grid in order of nondecreasing solution value. Ordered Upwind Methods (OUMs) [59, 60] are an extension of these Dijkstra-like methods that approximate the solution to static convex HJ PDEs in a single pass, but not exactly in order of nondecreasing solution value.

1.5 Contributions

Dijkstra-like methods require *causal* discretizations [60], for which the solution value at a node is independent of any solution value at another node that is greater than or equal to it. Dijkstra-like methods have been used to solve isotropic problems and some anisotropic problems. In this thesis we extend Dijkstra-like methods to solve anisotropic static HJ PDEs in two ways.

In the first extension, we analyze and implement a Dijkstra-like method for solving a class of axis-aligned anisotropic HJ PDEs on orthogonal grids. This extension is described in [4]. Our method is essentially the same as the Dijkstra-like Fast Marching Method (FMM) for isotropic problems [55], but we prove that it is applicable to all axis-aligned anisotropic HJ PDEs. Osher's fast marching criterion [47, 62] states that FMM can be used for a similar class of axis-aligned problems. Our criterion restricts Osher's criterion to rule out multiple solutions and loosens it to allow for nondifferentiable Hamiltonians. We prove the convergence properties of the resulting discretization, discuss implementation issues in detail, and report on experimental results of applying our method to problems such as two robot coordinated optimal control (e.g., Figure 1.1) and seismic anelliptic wavefront propagation [28].

The second extension is a Dijkstra-like method for solving general convex static HJ PDEs. This extension is described in paper submission [5]. Our method is distinct from the OUM described in [59, 60], which solves the same class of HJ PDEs. That method keeps track of an accepted front that defines the boundary of nodes which have computed values thus far, and is not strictly a Dijkstra-like method since the discretization is not necessarily solved in order of nondecreasing solution value. For that method the stencils are dynamically created using only nodes from the accepted front, whereas for our method the stencils are generated in an additional initial pass, resulting in a causal discretization that can be solved in a Dijkstra-like manner. Our method has the benefit that the stencil-width depends only on the local grid spacing, while for the OUM in [59, 60] the stencil width depends on the

global grid spacing. We propose and analyze several criteria to ensure that the stencil is causal and use these criteria to develop the initial pass that generates the stencils. Although we develop only a Dijkstra-like method, our criteria are generalized so that they can be used to build Dial-like methods, which solve the discrete equation simultaneously for buckets of nodes with similar values and may potentially be more easily implemented in parallel [23, 64, 65]. We test the Dijkstra-like method on several applications, including optimally navigating a robot to a goal through obstacles and a strong wind (see Figure 1.2), and computing the first arrival time for a seismic wave with elliptical but inhomogeneous propagation speed (see Figure 1.6).

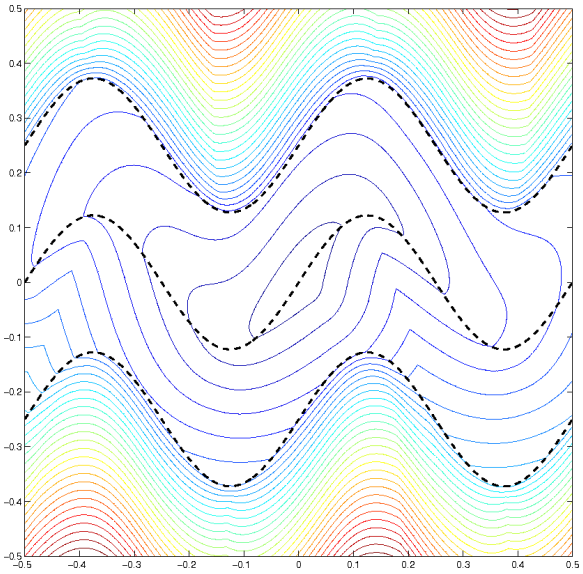


Figure 1.6: Contours of first-arrival times of a seismic wave.

1.6 Outline

Chapter 2 surveys related work to put our research into context. We discuss work from broader areas such as dynamic programming, optimal control,

and viscosity solutions. We narrow in on our subject by describing work that has been done on general numerical methods for time-dependent and static HJ PDEs, and in particular, single-pass methods for static HJ PDEs.

Chapter 3 details elements of our framework common to both algorithms for formulating and solving static HJ PDEs in a Dijkstra-like manner. We define the viscosity solution of a static HJ PDE that we wish to compute. We specify a format for the discretization and what types of properties are required so that the solution to the approximate problem converges to the viscosity solution as the grid is refined. Also, we point out what it means for the discretization to be causal. The chapter includes a definition of the basic Dijkstra-like method and a proof that it solves a causal discretized problem.

The analysis, implementation, and experimental testing of the Dijkstra-like method for solving axis-aligned anisotropic HJ PDEs is included in Chapter 4. That of the Dijkstra-like method with pre-computed stencil for solving general convex static HJ PDEs is in Chapter 5. In formulating the discretization and the algorithm, proving convergence, and proving that the algorithm solves the discretized system of equations, we attempt to follow the framework outlined in Chapter 3 as closely as is practical. In Chapter 5, we diverge slightly from the framework but the proofs still apply.

Chapter 6 ties together principles and observations from developing and testing the two Dijkstra-like methods and plots a course for developing these ideas further.

Chapter 2

Related Work

2.1 Viscosity Solutions of Hamilton-Jacobi Equations

The Dirichlet boundary-value problem for a static HJ PDE is to find a function $u : \overline{\Omega} \to \mathbb{R}$ satisfying (1.8), while the Cauchy initial-value problem for a time-dependent HJ PDE is to find a function $u : \overline{\Omega} \times \mathbb{R} \to \mathbb{R}$ such that

$$u_t + H(x, Du(x)) = 0, \qquad \text{in } \Omega \times (0, \infty) \tag{2.1a}$$

$$u(x, 0) = u_0(x), \quad \text{on } \Omega \times \{0\}, \tag{2.1b}$$

where $u_0 : \Omega \to \mathbb{R}$ specifies the initial value. In [17] both a boundary-value problem of a static HJ PDE and an initial-value problem of a time-dependent HJ PDE are described. In general, it is not possible to find a classical solution which is everywhere differentiable to an HJ PDE. The appropriate weak solution for the optimal control and wavefront first-arrival problems we seek to solve is differentiable almost everywhere and is called the viscosity solution. It is characterized for first-order HJ PDEs in [17]. Viscosity solution theory for more general second-order HJ PDEs is summarized in [19]. However, in our work we consider only first-order static HJ PDEs.

The viscosity solution of an HJ PDE is identical to the value function of a corresponding optimal control problem. For infinite-horizon discounted

control problems, this connection is discussed in great detail in [7]. The HJ PDEs for such problems include a forcing term which is absent for the undiscounted problems that we consider. The connection between viscosity solutions of time-dependent HJ PDEs and finite-horizon optimal control problems is examined in [25, Chapter 10].

2.2 Numerical Methods for Hamilton-Jacobi Equations

One cannot usually find an analytical solution to an HJ PDE so instead a numerical approximation is computed. Convergence of a finite difference scheme for time-dependent HJ PDEs is proved in [18]. In [8] it is proved that if a discretized scheme is monotone, stable, and consistent with a general second-order HJ PDE, the discretized solution will converge to the viscosity solution of the HJ PDE. The static first-order HJ PDEs we consider are a subset of the second-order HJ PDEs to which this theory applies. Our schemes, based on finite differences, mantain these properties so that convergence is assured (see Section 3.2).

Time-dependent HJ PDEs have the nice property that the solution at time t is dependent only on the solution at previous times. For this reason, a natural method for approximating the viscosity solution of a time-dependent HJ PDE is to discretize the PDE in space and time and march forward in time, calculating the approximate solution at one time step based on the solution at the previous time step, as is done in [18]. In [46] a connection is drawn between static and time-dependent HJ PDEs, by observing that one can solve a static HJ PDE by using the level set method to formulate and solve a related time-dependent HJ PDE. However, we focus on methods that solve static HJ PDEs directly because they are generally much more efficient computationally.

There are several different types of spatial discretizations that have been used to approximate the solution of HJ PDEs, both time-dependent and static.

Because of their simplicity, finite difference schemes have been very popular. Semi-Lagrangian schemes, in the category of finite difference schemes, directly discretize Bellman's dynamic programming principal and are studied in [26, 27, 63]. Eulerian finite difference schemes, which discretize the HJ PDE by numerically approximating the solution gradient, are used to solve static HJ PDEs in [53, 55, 70]. In [60] the equivalence between a particular semi-Lagragian scheme and a first-order upwind Eulerian finite difference scheme is proved.

Essentially non-oscillatory (ENO) and weighted ENO (WENO) are finite difference schemes that can produce higher-order accurate approximations of the solution where it is smooth while avoiding spurious oscillation at kinks in the solution, places where the solution is not differentiable. For solving time-dependent HJ PDEs, an ENO scheme is used in [48] and a WENO scheme is used in [35]. WENO schemes have been used to solve static HJ PDEs iteratively in [66, 68]. These methods were tested on several examples to demonstrate higher-order accurate convergence. However, convergence was not proven, possibly because the schemes are not monotone which makes the proof difficult. Furthermore, the solution of a first-order discretization is used as an initial guess to make convergence of the iterated method to the higher-order accurate solution more likely and faster.

A finite element discretization is used in [11] to solve static HJ PDEs. Discontinuous Galerkin (DG) methods have been used to solve a static HJ PDE with forcing term in [15] and to solve time-dependent HJ PDEs in [16, 33]. Second-order DG methods are developed to solve Eikonal equations in [42, 69].

In our research we use mainly first-order accurate semi-Lagrangian and Eulerian finite difference schemes because of their relative simplicity and guaranteed monotonicity and causality properties. In Chapter 5, we use grid refinement instead of higher-order accurate approximations to compute accurate solutions efficiently.

2.3 Dynamic Programming and Optimal Control

The classic book [9] introduced the ideas of a multi-stage decision process and the principle of optimality, popularizing the dynamic programming method. The optimal control problems we consider satisfy a continuous-time, continuous-state principal of optimality, which can be stated as a Hamilton-Jacobi-Bellman PDE [7, 25] with H as defined in (1.9). In [24] Dijkstra's Algorithm for finding the cost of the shortest paths on a graph is first described. Dijkstra's Algorithm is a dynamic programming method, and Dijkstra-like methods such as the Fast Marching Method (FMM) and Ordered Upwind Methods (OUMs) in general can be seen as modifications to the algorithm to approximate the cost of minimal-cost paths in a continuous space. We use Dijkstra-like methods to solve HJ PDEs. A modern reference [10] examines general dynamic programming algorithms for the purpose of solving discrete-time operations research problems and continuous-time optimal control problems.

2.4 Methods for Static Hamilton-Jacobi Equations

Our research looks at fast methods for directly solving static first-order HJ PDEs. Fast methods can be contrasted with slow methods like naive Gauss-Seidel iteration, which have asymptotic complexity $\mathcal{O}(N^2)$. However, fast methods are not appropriate for solving second-order HJ PDEs, which do not possess characteristics. Two main categories of methods for these problems are ordered upwind (or single-pass or label-setting) methods and sweeping (or iterative or label-correcting) methods. Our research focuses on methods in the first category. In particular, we create Dijkstra-like OUMs that construct the approximation in nondecreasing value order.

2.4.1 Fast Marching Methods

The first Dijkstra-like method for a first-order accurate semi-Lagrangian discretization of the isotropic Eikonal PDE on an orthogonal grid was developed in [63]. The Dijkstra-like FMM was later independently developed in [55]

for the first-order accurate upwind Eulerian finite-difference discretization of the same Eikonal PDE. FMM was then extended to handle higher-order accurate upwind discretizations on grids and unstructured meshes in $\mathbb{R}^d$ and on manifolds [39, 57, 58]. In [56] it was shown that Dijkstra's method on a uniform orthogonal grid produces the solution for the anisotropic maximum norm Eikonal equation. By solving an isotropic problem on a manifold and then projecting the solution into a subspace, FMM can solve certain anisotropic problems [58]. For example, if H is defined by (1.9) with a constant elliptical speed profile $\mathcal{A}_f(x) = \mathcal{A}_f$, then (1.8) can be solved by running isotropic FMM on an appropriately tilted planar manifold and then projecting away one dimension. Some anisotropic etching problems have also been solved using FMM [47].

The fact that correct operation of Dijkstra-like algorithms for approximating the Eikonal PDE requires the causality property that $\underline{u}(\underline{x})$ can be written only in terms of smaller-valued $\underline{u}$ at neighboring nodes was stated in [63], but a reader might incorrectly infer from further comments in that paper that such algorithms would not work for any unstructured grid or anisotropic problem. That FMM is applicable for any consistent, causal, finite-difference discretization of a general static convex HJ PDE on an orthogonal grid is stated in [56]. However, it is now understood that this criterion applies even more generally, since a Dijkstra-like method can be used to efficiently solve on a graph any nonlinear system of equations for which $\underline{u}(\underline{x})$ is dependent only on smaller-valued $\underline{u}$ at neighboring nodes. A sufficient criterion (see Section 4.2.1) under which FMM can be used for orthogonal, finite-difference discretizations of static HJ PDEs—now commonly referred to as "Osher's criterion"—is widely attributed to an unpublished work by Osher & Helmsen, but the earliest published description seems to be [47]. While it is stronger than the causality conditions described earlier, it is useful because it is stated as a condition on the analytic Hamiltonian instead of the equations created by the discretization. In this thesis, we likewise seek conditions under which FMM is applicable that are closer to the problem's definition than the algorithm's implementation.

A Dijkstra-like method with a semi-Langrangian update scheme for isotropic

equations was developed in [21]. The semi-Lagrangian update is distinct from that in [63] as the time step for the integration is fixed. Numerical tests indicate that the algorithm computes more accurate solutions than FMM with a finite difference scheme, but it is marginally more costly due to an expanded node neighborhood.

Dijkstra-like methods such as those described above have an asymptotic complexity of $\mathcal{O}(N \log N)$, because each of N nodes is visited and a priority queue ordered according to node value is used to determine the next node to visit in $\mathcal{O}(\log N)$ operations (see Section 3.3.1). There are modified versions of the FMM that claim to have $\mathcal{O}(N)$ asymptotic complexity. The Group Marching Method in [38] advances a group of grid points at a time, rather than sorting the node values to march forward a single node as is done in a Dijkstra-like method. An untidy priority queue (i.e. bucket sort) is used in [67] so that the next node to visit can be determined in constant time. The authors argue that the error introduced by the untidy priority queue will not be significant considering the discretization error. In [52] it is proven that a bucket width for this untidy priority queue can be chosen so that the error introduced is independent of the speed function f and the computational complexity is $\mathcal{O}((\hat{f}/\check{f}) \log N)$, where $\hat{f}$ is the maximum speed and $\check{f}$ the minimum speed. Both [38, 67] demonstrate their respective methods outperforming FMM on selected problems, but neither gives rigorous theoretical guarantees of this result for any class of problems.

2.4.2 Ordered Upwind Methods

The OUMs developed in [59, 60] can solve general convex anisotropic problems on unstructured grids with an asymptotic complexity only a constant factor (related to the degree of anisotropy) worse than FMM. FMM fails for these general problems because the neighboring simplex from which the characteristic approaches a node x may contain another node y such that causality does not hold: $\underline{u}(x) \leq \underline{u}(y)$. These OUMs avoid this difficulty by searching along the accepted front to find a set of neighboring nodes (which may not be direct neighbors of x) whose values have been accepted, and then

constructing a virtual simplex with these nodes from which to update $\underline{u}(x)$. Although this search along the active front does not degrade the asymptotic complexity, it does significantly increase the computational cost in practice. This effect can be partially mitigated by using nontrivial data structures, such as 2^d-trees, to speed up the search along the active front.

2.4.3 Sweeping Methods

An alternative to these single-pass algorithms are the sweeping algorithms, which are often even simpler to implement than FMM. Sweeping algorithms are also capable of handling anisotropic and even nonconvex problems. The simplest sweeping algorithm is to just iterate through the grid updating each node in a Gauss-Seidel (GS) fashion (so a new value for a node is used immediately in subsequent updates) until $\underline{u}$ converges. GS converges quickly if the node update order is aligned with the characteristics of the solution, so better sweeping algorithms [12, 22, 29, 36, 50, 51, 62, 70] alternate among a collection of static node orderings so that all possible characteristic directions will align with at least one ordering. It is argued in [70] that these methods achieve $\mathcal{O}(N)$ asymptotic complexity (assuming that the node orderings are already determined); however, unlike OUMs the constant depends on the particular problem. For practical grid resolutions on Eikonal problems with curved characteristics FMM outperforms sweeping methods despite the difference in asymptotic complexity [30, 34]. A method for improving the efficiency of sweeping methods by skipping over a node during a sweep when a value update is guaranteed not to change the value is described in [6].

There are also some sweeping algorithms which use dynamic node orderings; for example [6, 11, 49]. These algorithms attempt to approximate the optimal ordering generated by OUMs such as FMM without the overhead associated with sorting values exactly. These methods have been demonstrated in practice to be comparable to or better than OUMs for certain problems and grid resolutions. However, in general these methods may need to revisit nodes multiple times, because they do not compute the solution

in an order that exploits node value dependencies.

The algorithm described in [20] combines aspects of OUMs and sweeping methods. It computes solutions iteratively for a propagating band of grid nodes, but determines periodically which band nodes' values can no longer change and removes them from the band. This method is found to be more efficient than the Fast Sweeping method of [62, 70] for a problem with highly-curved characteristics but less efficient for a highly-anisotropic problem with straight characteristics. The asymptotic complexity of this method is not analyzed.

2.5 Applications of Static Hamilton-Jacobi Equations

The solution of an HJ PDE can be used in many contexts, including etching/deposition/lithography, seismic imaging, image processing, data visualization, computer vision, and optimal control. We list some applications of the solution to Eikonal and to non-Eikonal HJ PDEs. This is not a comprehensive list and is merely intended to demonstrate the wide applicability of HJ PDEs.

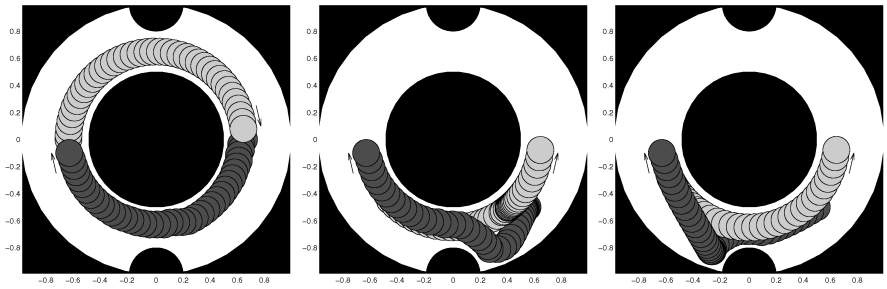
The first-arrival time of a front, which is the solution to an HJ PDE, can be computed in order to solve some etching/deposition/lithography and seismic imaging problems. A semiconductor deposition problem above a trench was solved in [55], and it was noted that other etching/deposition/lithography problems could be solved using the Eikonal equation. Non-Eikonal HJ PDE solutions were applied to some anisotropic etching problems in [47]. An anisotropic seismic imaging problem with elliptic speed profile was solved in [59].

Some image processing and data visualization problems have been solved using Eikonal equations. Euclidean distance maps, which are the solutions to Eikonal equations, were computed and used to generate skeletons for image processing in [22]. The unorganized point data obtained from a 3-dimensional scanner was visualized using a computation on a distance func-

tion which is the solution to an Eikonal equation in [70].

In [13] it was proven that an Eikonal equation could be formulated and solved to determine a surface from a shaded image (i.e., shape-from-shading) given certain light source and reflectance properties. Note that not all shape-from-shading problems can be addressed by solving Eikonal equations and [32] discusses techniques for shape-from-shading in greater generality. The solution of a first-order HJ PDE was used to solve a shape-from-shading problem in [53].

The Eikonal equation was solved to find optimal trajectories for isotropic control problems in [63]. This equation was used to solve several path planning problems including computing geodesics on a manifold in [40].

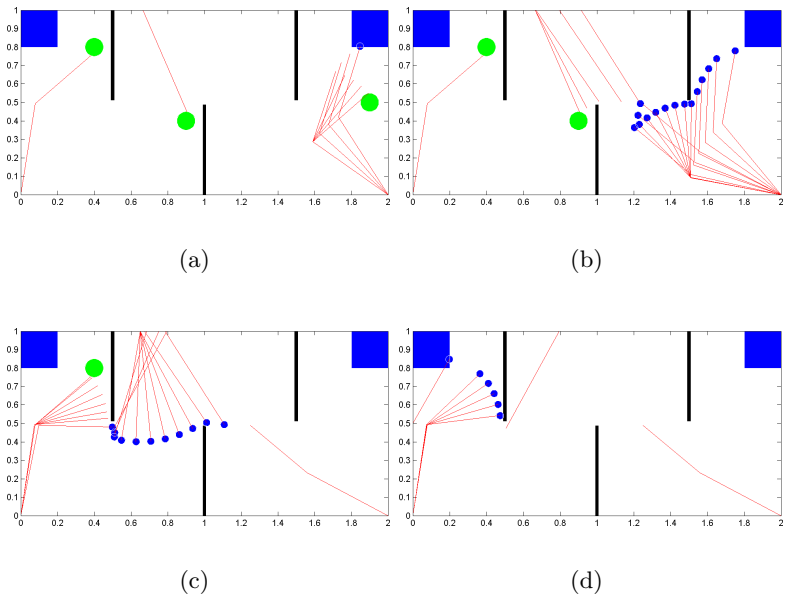


(a) (b) (c)

Figure 2.1: Motion of two robots in a 2D world. The light robot is constrained to a circular path, while the dark robot may move anywhere within the obstacle free space, where obstacles are shown in black. The light robot's goal is on the right, and the dark robot's goal is on the left. Arrows show direction of movement, and are placed near goal states.

Anisotropic HJ PDE solutions have been used to solve a multi-robot coordinated optimal control problem in [1] (see Figure 2.1). This application helped to motivate the extension of FMM to axis-aligned HJ PDEs described in Chapter 4. A sequence of HJ PDE problems can be solved to determine meeting locations and connecting trajectories for an optimal multi-robot multi-location rendezvous problem, as described in [2]. An example of such a problem where three robot arms cooperate to transport cargo as quickly

as possible in shown in Figure 2.2. The methods we describe in Chapters 4 and 5 can be used to solve the individual HJ PDE problems in the sequence.



(a) (b)

(c) (d)

Figure 2.2: Robot arms cooperating to optimally transport cargo from a pickup to a dropoff location. The sequence of figures is a time series with (a) showing the beginning of the robot arm motions and (d) showing the completion of the motions. The starting area for the end effector of each robot arm is indicated by a large circle. The square box on the top-right/top-left is the pickup/dropoff location for the cargo. A small circle at the end effector of a moving robot arm indicates that the robot is currently carrying the cargo. (a) Robot 1 moves from its starting location to cargo pickup area. (b) Robot 1 moves from the cargo pickup box to meet robot 2, while robot 2 moves from its starting location to meet robot 1. (c) Robot 2 moves from its meeting with robot 1 to meet robot 3, while robot 3 moves from its starting location to meet robot 2. (d) Robot 3 moves from its meeting with robot 2 to the cargo dropoff area.

Chapter 3

Framework

This chapter serves to unite the Dijkstra-like methods described in Chapters 4 and 5 under a single framework. Both methods involve devising for a particular grid a discrete system of equations that approximates the static HJ PDE (1.8). Whether or not we are using a Dijkstra-like method, we require our approximate solution computed on a grid to converge to the continuous viscosity solution of (1.8) as the grid spacing shrinks towards zero. We prove in Section 3.2 that consistency, monotonicity, and stability of the discrete equations implies convergence of the approximate solution [8]. For a Dijkstra-like method to be applied to solve the discrete equation, we require it to be causal [55, 60, 64]. We prove that a Dijkstra-like method solves such causal systems in Section 3.3. We present the proof of convergence and that of applicability of Dijkstra-like methods here for completeness.

3.1 Discretization

We denote the grid $\mathcal{G}$. The grid may be an orthogonal grid like that used in Chapter 4 and shown in Figure 1.5 or a simplicial grid like that used in Chapter 5 and shown on the left side of Figure 1.2. Take the discretized domain $\underline{\Omega}$ and the discretized boundary $\underline{\partial\Omega}$ to be disjoint sets of grid nodes and let $\mathcal{X} = \underline{\Omega} \cup \underline{\partial\Omega}$ be the set of all grid nodes. Let $\mathcal{N}(x)$ be the set of grid neighbors (i.e., the one-ring neighborhood) of node $x \in \mathcal{X}$. Let $\hat{h}^{\mathcal{G}}$ be the

maximum grid spacing of $\mathcal{G}$:

$$\hat{h}^{\mathcal{G}} = \max_{x \in \underline{\Omega},\, y \in \mathcal{N}(x)} \|x - y\|.$$

Let $\overrightarrow{\mathcal{Y}}^{\mathcal{G}}(x) \subset \mathcal{X}$ denote the numerical stencil of node x for grid $\mathcal{G}$. For example, $\overrightarrow{\mathcal{Y}}^{\mathcal{G}}(x) = \mathcal{N}(x)$ for standard FMM and our extension in Chapter 4. However, the method in Chapter 5, may require $\overrightarrow{\mathcal{Y}}^{\mathcal{G}}(x) \supset \mathcal{N}(x)$. For $x \in \partial\underline{\Omega}$, $\overrightarrow{\mathcal{Y}}^{\mathcal{G}}(x) = \emptyset$. Let $\overleftarrow{\mathcal{Y}}^{\mathcal{G}}(x) = \{y \in \underline{\Omega} \mid x \in \overrightarrow{\mathcal{Y}}^{\mathcal{G}}(y)\}$ be the set of nodes which have x in their stencil. Let $\hat{r}^{\mathcal{G}}(x) = \max_{y \in \overrightarrow{\mathcal{Y}}^{\mathcal{G}}(x)} \|x - y\|$ denote the the numerical stencil radius of node x for grid $\mathcal{G}$.

Let $\underline{H}(x, \mathcal{Y}, \phi, \mu) \in \mathbb{R}$ be the numerical Hamiltonian function with parameters $x \in \Omega$, stencil $\mathcal{Y} \subset \overline{\Omega}$, $\phi : \overline{\Omega} \to \mathbb{R}$, and $\mu \in \mathbb{R}$. The approximate Dirichlet problem corresponding to (1.8) is to find a function $\underline{u}^{\mathcal{G}} : \overline{\Omega} \to \mathbb{R}$ such that

$$\underline{H}(x, \overrightarrow{\mathcal{Y}}^{\mathcal{G}}(x), \underline{u}^{\mathcal{G}}, \underline{u}^{\mathcal{G}}(x)) = 0, \qquad x \in \underline{\Omega} \tag{3.1a}$$

$$\underline{u}^{\mathcal{G}}(x) = g(x), \quad x \in \partial\underline{\Omega}. \tag{3.1b}$$

and $\underline{u}^{\mathcal{G}}(x)$ satisfies a non-expansive interpolation,[1] for $x \in \overline{\Omega} \setminus \mathcal{X}$. For example, $\underline{H}$ could be a Godunov upwind finite difference scheme [53] as used in Chapter 4, or for H of form (1.9) a semi-Lagrangian scheme [59] as is used in Chapter 5. Excluding the boundary condition (3.1b) and interpolation, we have a system of $|\underline{\Omega}|$ nonlinear equations of the form (3.1a), one for each node $x \in \underline{\Omega}$. We wish to compute the $|\underline{\Omega}|$ values, $\underline{u}^{\mathcal{G}}(x)$, one for each node $x \in \underline{\Omega}$. We may omit $\mathcal{G}$ and write $\underline{u} = \underline{u}^{\mathcal{G}}$, $\overrightarrow{\mathcal{Y}}(x) = \overrightarrow{\mathcal{Y}}^{\mathcal{G}}(x)$, and $\overleftarrow{\mathcal{Y}}(x) = \overleftarrow{\mathcal{Y}}^{\mathcal{G}}(x)$ when no ambiguity results.

3.2 Convergence

For linear PDEs, [41] showed that for a consistent scheme, stability is necessary and sufficient for convergence. For nonlinear HJ PDEs, a numerical

[1] By non-expansive interpolation we mean the interpolated values are bounded by the data values, such as for linear interpolation.

scheme which is consistent, monotone, and stable implies that the solution $\underline{u}^{\mathcal{G}}$ of (3.1) converges to the viscosity solution u of (1.8) as the spacing between grid nodes goes to zero [8]. The monotonicity condition for the scheme is analogous to the comparison principle for the PDE (see Assumption 3.6). This relationship in the case of finite-difference schemes for degenerate elliptic PDEs (a superset of the first-order HJ PDEs we study) is examined in [45].

We follow the technique described in [8] to prove convergence in this section. We note that this proof technique applies to the more general set of degenerate elliptic PDEs. However, despite its generality it is a remarkably simple proof method, so we employ it here. We define consistency, monotonicity, and stability in Properties 3.2 to 3.4 and use these in the proof. Then, to show the convergence of the schemes in Chapter 4 and Chapter 5, we need only show that Properties 3.2 to 3.4 or their equivalents are satisfied.

3.2.1 Consistency, Monotonicity, and Stability

The following property states that the numerical stencil radius goes to zero at the same rate as the grid spacing.

Property 3.1. *Let $\hat{r}^{\mathcal{G}}(x) = \mathcal{O}(\hat{h}^{\mathcal{G}})$ for all $x \in \underline{\Omega}$.*

Roughly speaking, the numerical scheme is consistent if the numerical Hamiltonian $\underline{H}^{\mathcal{G}}$ approaches the Hamiltonian H as $\hat{h}^{\mathcal{G}} \to 0$. Let $C_b^\infty(\Omega)$ be the set of smooth, bounded functions on domain Ω.

Property 3.2 (Consistency). *Let Property 3.1 hold. Let $x \in \Omega$ and $\phi \in C_b^\infty(\Omega)$. For sequences $\mathcal{G}_k$, $x_k \in \underline{\Omega}_k$, and ξ_k, such that $x_k \to x$, $\hat{h}^{\mathcal{G}_k} \to 0$, and $\xi_k \to 0$ as $k \to \infty$,*

$$\lim_{k\to\infty} \underline{H}(x_k, \mathcal{Y}_k, \phi + \xi_k, \phi(x_k) + \xi_k) = H(x, D\phi(x)), \tag{3.2}$$

where $\mathcal{Y}_k = \overrightarrow{\mathcal{Y}}^{\mathcal{G}_k}(x_k)$.

The numerical scheme is monotone if an increase in the numerical Hamiltonian $\underline{H}$ implies a decrease in the function ϕ, all other quantities being equal. The monotonicity property below is stated as the contrapositive of this description.

Property 3.3 (Monotonicity). *Let $x \in \Omega$ and $\mu \in \mathbb{R}$. Let functions $\check{\phi} : \overline{\Omega} \to \mathbb{R}$ and $\hat{\phi} : \overline{\Omega} \to \mathbb{R}$ be such that $\check{\phi}(y) \leq \hat{\phi}(y)$ for all $y \in \mathcal{Y}$. Then, $\underline{H}(x, \mathcal{Y}, \check{\phi}, \mu) \geq \underline{H}(x, \mathcal{Y}, \hat{\phi}, \mu)$.*

Typically, stability of a numerical scheme serves to limit the extent to which the numerical scheme amplifies perturbations in the boundary conditions. We use an alternative definition of stability which is the same as that used in [8] and convenient for proving convergence. The numerical scheme is stable if for every grid $\mathcal{G}$ the solution $\underline{u}^{\mathcal{G}}$ to (3.1), is bounded independent of the grid spacing $\hat{h}^{\mathcal{G}}$.

Property 3.4 (Stability). *There exists $\hat{U} \in \mathbb{R}$ such for any $\mathcal{G}$, the solution $\underline{u}^{\mathcal{G}}$ to (3.1) satisfies $\min_{x \in \underline{\partial\Omega}} g(x) \leq \underline{u}^{\mathcal{G}} \leq \hat{U}$.*

3.2.2 Viscosity Solution and Proof of Convergence

We show that $\underline{u}^{\mathcal{G}}$ converges to the viscosity solution u of (1.8) as $\hat{h}^{\mathcal{G}} \to 0$. For the proof, we assume that Properties 3.1 to 3.4 hold.

We first define what it means for a function u to be a viscosity solution. An upper (respectively, lower) semi-continuous function u is a viscosity subsolution (respectively, supersolution) of (1.8) if for all $\phi \in C_b^\infty(\overline{\Omega})$ such that $u - \phi$ has a local maximum (respectively, minimum) at $x \in \Omega$ we have

$$H(x, D\phi(x)) \leq 0 \quad \text{(respectively, } H(x, D\phi(x)) \geq 0\text{)}. \tag{3.3}$$

A continuous function u is a viscosity solution if it is both a viscosity subsolution and a viscosity supersolution [19].

We proceed to show in Propositions 3.5 and 3.7 that $\underline{u}^{\mathcal{G}}$ converges to the viscosity solution as defined above. By Property 3.4 (i.e., stability), we have

bounded $\underline{u}^{\mathcal{G}}$. So, we define $\hat{u} \in B(\overline{\Omega})$ and $\check{u} \in B(\overline{\Omega})$ as

$$\hat{u}(x) = \limsup_{\substack{y \to x \\ \hat{h}^{\mathcal{G}} \to 0}} \underline{u}^{\mathcal{G}}(y) \quad \text{and} \quad \check{u}(x) = \liminf_{\substack{y \to x \\ \hat{h}^{\mathcal{G}} \to 0}} \underline{u}^{\mathcal{G}}(y). \tag{3.4}$$

Note that $\hat{u}$ is upper semi-continuous and $\check{u}$ is lower semi-conintuous.

Proposition 3.5. *$\hat{u}$ is a viscosity subsolution of (1.8) and $\check{u}$ is a viscosity supersolution of (1.8).*

Proof. We prove only that $\hat{u}$ is a viscosity subsolution of (1.8), as the second part of the proposition can be proved symmetrically. Let $\hat{x} \in \Omega$ be a local maximum of $\hat{u} - \phi$ for some $\phi \in C_b^\infty(\Omega)$. Without loss of generality, assume $\hat{x}$ is a strict local maximum [25, p. 542] and $(\hat{u} - \phi)(\hat{x}) = 0$. Then, there exist sequences $\mathcal{G}_k$ and $x_k \in \underline{\Omega}_k$ such that as $k \to \infty$,

$$\begin{gathered} \hat{h}^{\mathcal{G}_k} \to 0, \quad x_k \to \hat{x}, \quad \underline{u}^{\mathcal{G}_k}(x_k) \to \hat{u}(\hat{x}), \quad \textit{and} \\ (\underline{u}^{\mathcal{G}_k} - \phi)(x_k) \geq (\underline{u}^{\mathcal{G}_k} - \phi)(y) \text{ for all } y \in \mathcal{Y}_k, \end{gathered}$$

where $\mathcal{Y}_k = \overrightarrow{\mathcal{Y}}^{\mathcal{G}_k}(x_k)$.

Let $\xi_k = (\underline{u}^{\mathcal{G}_k} - \phi)(x_k)$. We have $\xi_k \to 0$ and $\phi(y) + \xi_k \geq \underline{u}^{\mathcal{G}_k}(y)$, for all $y \in \mathcal{Y}_k$. Consequently, by Property 3.3 (i.e., monotonicity) and the definition of $\underline{u}^{\mathcal{G}_k}$ we have

$$\begin{aligned} \underline{H}(x_k, \mathcal{Y}_k, \phi + \xi_k, \phi(x_k) + \xi_k) &= \underline{H}(x_k, \mathcal{Y}_k, \phi + \xi_k, \underline{u}^{\mathcal{G}_k}(x_k)) \\ &\leq \underline{H}(x_k, \mathcal{Y}_k, \underline{u}^{\mathcal{G}_k}, \underline{u}^{\mathcal{G}_k}(x_k)) = 0 \end{aligned}$$

Take the limit as $k \to \infty$, and use Property 3.2 (i.e., consistency) to get

$$0 \geq \lim_{k \to \infty} \underline{H}(x_k, \mathcal{Y}_k, \phi + \xi_k, \phi(x_k) + \xi_k) = H(x, D\phi(x))$$

Therefore, $\hat{u}$ is a viscosity subsolution. □

If H is continuous, (1.8) satisfies the comparison principle below [7]. This comparison principle roughly states that solutions of (1.8) are monotone in

the boundary values g and it is used in the proof of Proposition 3.7 below to show that $\underline{u}^{\mathcal{G}}$ converges to u, which is unique.

Assumption 3.6 (Comparison Principle). *For any bounded upper semi-continuous u^* and bounded lower semi-continuous u_*, which are a viscosity subsolution and supersolution, respectively, of (1.8), such that $u^*(x) \leq u_*(x)$ for all $x \in \partial\Omega$, we have $u^*(x) \leq u_*(x)$ for all $x \in \overline{\Omega}$.*

Proposition 3.7. *Let H be continuous. The function $\hat{u} = \check{u} = u$ is the unique viscosity solution of (1.8). As $\hat{h}^{\mathcal{G}} \to 0$, $\underline{u}^{\mathcal{G}}$ converges uniformly to u.*

Proof. By Proposition 3.5, $\hat{u}$ is an upper semi-continuous viscosity subsolution of (1.8) and $\check{u}$ is a lower semi-continuous viscosity supersolution of (1.8). It follows by the comparison principle that $\hat{u} \leq \check{u}$. But $\check{u} \leq \hat{u}$ by (3.4), so $u = \hat{u} = \check{u}$ is a viscosity solution of (1.8). Again, by the comparison principle, u must be the unique viscosity solution of (1.8). Therefore, by (3.4), $\underline{u}^{\mathcal{G}}$ converges uniformly to u as $\hat{h}^{\mathcal{G}} \to 0$. □

3.3 Algorithm

We note that (3.1) defines a potentially very large system of nonlinear equations, one equation for each node $x \in \underline{\Omega}$. A Dijkstra-like method can be used to solve this system very efficiently if for all x the solution value $\underline{u}(x)$ is dependent only on nodes in $\overrightarrow{\mathcal{Y}}(x)$ with smaller values. This property represents a causal relationship between node values. There is an information flow from nodes with smaller values to those with larger values. The causal relationship is meant to mimic that of the PDE (1.8) in which the solution u of (1.8) at x is completely defined using only values of u from states that are backwards along the characteristic line that passes through x, states whose u values must be less than $u(x)$.

We show that if the numerical scheme is causal, (3.1) can be solved using a Dijkstra-like algorithm in a single pass through the nodes in $\underline{\Omega}$ in order of increasing value $\underline{u}^{\mathcal{G}}$. Our argument is essentially the same as those in [55, 60, 64]. We require that the discrete equation has a unique solution.

Property 3.8 (Unique Solution). *Let $x \in \Omega$ and $\phi : \overline{\Omega} \to \mathbb{R}$. There exists a unique solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{Y}, \phi, \mu) = 0$.*

Causality of the numerical scheme means that if the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{Y}, \phi, \mu) = 0$ depends on the value $\phi(y)$ for $y \in \mathcal{Y}$, then $\tilde{\mu} > \phi(y)$. Causality is used in the proof of Proposition 3.13 that the Dijkstra-like algorithm solves (3.1).

Property 3.9 (Causality). *Let Property 3.8 hold. Let $\mu = \tilde{\mu}$ be the unique solution to $\underline{H}(x, \mathcal{Y}, \phi, \mu) = 0$. Let $\mathcal{Y}^- = \{y \in \mathcal{Y} \mid \phi(y) < \tilde{\mu}\}$. Then $\mu = \tilde{\mu}$ is also the unique solution to $\underline{H}(x, \mathcal{Y}^-, \phi, \mu) = 0$.*

The following corollary of Property 3.9 states that if adding a node y to the stencil causes the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{Y}, \phi, \mu) = 0$ to decrease, then the decreased solution must still be greater than the value $\phi(y)$. This corollary is more convenient than Property 3.9 to use in the proof of Lemma 3.12.

Corollary 3.10. *Let Property 3.9 be satisfied. Let $x, y \in \Omega$ and $\phi : \overline{\Omega} \to \mathbb{R}$. Let $\mu = \tilde{\mu}$ be the unique solution to $\underline{H}(x, \mathcal{Y}, \phi, \mu) = 0$. Let $\mu = \tilde{\mu}^y$ be the unique solution to $\underline{H}(x, \mathcal{Y} \cup \{y\}, \phi, \mu) = 0$. If $\tilde{\mu}^y < \tilde{\mu}$, then $\tilde{\mu}^y > \phi(y)$.*

Proof. We prove the contrapositive of the last statement instead of the statement itself: if $\phi(y) \geq \tilde{\mu}$, then $\tilde{\mu}^y \geq \tilde{\mu}$.

Let $\phi(y) \geq \tilde{\mu}$. Then,

$$\mathcal{Y}^- = \{\tilde{y} \in \mathcal{Y} \mid \phi(\tilde{y}) < \tilde{\mu}\} = \{\tilde{y} \in \mathcal{Y} \cup \{y\} \mid \phi(\tilde{y}) < \tilde{\mu}\}.$$

It follows by Properties 3.8 and 3.9, $\tilde{\mu}$ is the unique solution to all of $\underline{H}(x, \mathcal{Y}, \phi, \mu) = 0$, $\underline{H}(x, \mathcal{Y}^-, \phi, \mu) = 0$, and $\underline{H}(x, \mathcal{Y} \cup \{y\}, \phi, \mu) = 0$. So, $\tilde{\mu} = \tilde{\mu}^y$, which implies $\tilde{\mu}^y \geq \tilde{\mu}$. □

3.3.1 Dijkstra-like Methods

Algorithm 1 (p.35) outlines a standard Dijkstra-like method [60] for solving (3.1) in a single-pass through the grid nodes. The algorithm can become

Dijkstra's algorithm or FMM depending on the choice of the `Update` function. Consider, for example, the `Update` function in the context of optimal control, where we are computing the minimal cost over all possible paths. For Dijkstra's algorithm, `Update` computes $\underline{v}(y)$ as a simple minimization over the neighboring nodes of y of the path costs to y via each neighbor. For FMM, the `Update` function computes $\underline{v}(y)$ as a minimization over the neighboring simplices of y of the minimum path costs to y via each simplex.

The `Update` function must satisfy a causality property in order for Algorithm 1 to terminate with a correct solution. `Update` must compute a node value $\underline{v}(y)$ based only on information from neighboring nodes with smaller values, so that $\underline{u}$ is computed in increasing order of $\underline{u}(y)$ [55, 63]. In Dijkstra's algorithm and FMM for a standard Euclidean-norm Eikonal equation on an orthogonal grid, this property is automatic.

```
1  foreach x ∈ Ω̲ do v̲(x) ← ∞
2  foreach x ∈ ∂Ω̲ do v̲(x) ← g(x)
3  H ← X
4  while H ≠ ∅ do
5      x ← argmin_{y∈H} v̲(y)
6      H ← H \ {x}
7      foreach y ∈ ←Y(x) ∩ H do
8          v̲(y) ← Update(y)
9      end
10 end
```

Algorithm 1: Standard Dijkstra-like method

The algorithm begins by initializing the *value* $\underline{v}(x)$ of each node x. If x is a boundary node, then $\underline{v}(x)$ is initialized to $g(x)$, otherwise it is initialized to ∞. The set $\mathcal{H}$ of *considered* nodes initially includes all nodes in $\mathcal{X}$. At the beginning of each **while** loop iteration, the node $x \in \mathcal{H}$ with the smallest value is *accepted*, it is removed from $\mathcal{H}$, and the values of all of the *considered* (i.e., still in $\mathcal{H}$) nodes in $\overleftarrow{\mathcal{Y}}(x)$ are *updated* using only the values of accepted nodes, including x. The **while** loop is repeated until $\mathcal{H}$ is empty or, in other words, all nodes have been accepted. The function call `Update`(y) returns

the solution $\mu = \tilde{\mu}$ to

$$\underline{H}(y, \overrightarrow{\mathcal{Y}}(y) \setminus \mathcal{H}, \underline{v}, \mu) = 0.$$

While the `Update` function in Algorithm 1 is determined by the underlying equation which we seek to solve, it is assumed that its execution time is independent of grid resolution and hence it does not affect the algorithm's asymptotic complexity. Dijkstra-like algorithms are usually described as being $\mathcal{O}(N \log N)$, where $N = |\mathcal{X}|$ is the number of grid nodes. This complexity is derived by noting that each node is removed from $\mathcal{H}$ once and in the usual binary min-heap implementation of $\mathcal{H}$ extraction of the minimum value node in line 5 costs $\mathcal{O}(\log |\mathcal{H}|) \leq \mathcal{O}(\log N)$. Note that the heap $\mathcal{H}$ need only sort considered nodes with finite values and typically the number of such nodes is much less than N.

Dijkstra-like methods are greedy algorithms: in Algorithm 1 the next node to remove from $\mathcal{H}$ is chosen greedily as the node in $\mathcal{H}$ with minimum value. This greedy selection of the next node yields a globally-correct method because of the causality of the discretization. It also yields a very efficient method because the value of each node in the grid is updated only a few times (no more than the number of neighboring nodes). The number of updates is in $\mathcal{O}(N)$ and it is not possible to compute the solution to (3.1) using asymptotically fewer updates.

3.3.2 Solution of Discretization

We prove that Algorithm 1 finds the solution to (3.1), by showing that when it terminates the node values $\underline{v}$ satisfy (3.1). We begin by proving a lemma to show that Algorithm 1 terminates. Let a subscript $i \geq 0$ represent the state of a variable at the beginning of the $(i+1)^{\text{st}}$ iteration of the **while** loop in Algorithm 1. For example, $\mathcal{H}_i$ is the state of $\mathcal{H}$ at the beginning of iteration $i+1$. From Line 5 and Line 6, we have $x_{i+1} = \operatorname{argmin}_{y \in \mathcal{H}_i} \underline{v}_i(y)$ and $\mathcal{H}_{i+1} = \mathcal{H}_i \setminus \{x_{i+1}\}$.

Lemma 3.11. *Let $N = |\mathcal{X}|$ be finite. Let $\overleftarrow{M} = \max_{x \in \underline{\Omega}} |\overleftarrow{\mathcal{Y}}(x)|$. If* `Update`

always terminates then Algorithm 1 terminates after N iterations of the **while** *loop and at most $N\overleftarrow{M}$ calls to* `Update`*.*

Proof. By Line 3 of Algorithm 1, $|\mathcal{H}_1| = N$. By Lines 5 and 6, $|\mathcal{H}_{i+1}| = |\mathcal{H}_i| - 1$. Thus, by the **while** loop condition, there are N iterations of the **while** loop. For every iteration of the **while** loop, there are at most $\overleftarrow{M}$ iterations of the **foreach** loop, each of which calls `Update` once. Therefore, Algorithm 1 terminates after at most $N\overleftarrow{M}$ calls to `Update`. □

The following lemma states that the nodes are accepted in nondecreasing value order. Let $\underline{v} : \mathcal{X} \to \mathbb{R}$ be the grid function after Algorithm 1 terminates.

Lemma 3.12. *Let Property 3.8 hold. Let Property 3.9 hold for all $x \in \mathcal{X}$ and $\mathcal{Y} = \overrightarrow{\mathcal{Y}}(x)$. For $1 \le i < j \le N+1$, we have $\underline{v}(x_i) \le \underline{v}(x_j)$.*

Proof. By Lemma 3.11, Algorithm 1 terminates. Note that once a node x_i is accepted, its value $\underline{v}(x_i) = \underline{v}_i(x_i)$ is fixed. Assume $\underline{v}(x_i) \le \underline{v}(x_{i+1})$, for $1 \le i \le N$. By induction on i, we have $\underline{v}(x_i) \le \underline{v}(x_j)$, for $1 \le i < j \le N+1$. We proceed by proving the inductive step.

Let i be such that $1 \le i \le N$. Let $y \in \mathcal{H}_i$. By Line 5, we have $\underline{v}(x_i) \le \underline{v}_{i-1}(y)$. First, consider the case $y \notin \overleftarrow{\mathcal{Y}}(x_i)$. Such y are not updated in the i^{th} iteration of the **while** loop, so $\underline{v}_i(y) = \underline{v}_{i-1}(y) \ge \underline{v}(x_i)$. Second, consider the case $y \in \overleftarrow{\mathcal{Y}}(x_i)$. In this case, by Line 8 and the definition of `Update`, $\mu = \underline{v}_i(y)$ is the unique solution to $\underline{H}(y, \overrightarrow{\mathcal{Y}}(y) \setminus \mathcal{H}_i, \underline{v}, \mu) = 0$. Because y is updated whenever a new node is added to $\overrightarrow{\mathcal{Y}}(y) \setminus \mathcal{H}$, we have $\overrightarrow{\mathcal{Y}}(y) \setminus \mathcal{H}_i = \tilde{\mathcal{Y}} \cup \{x_i\}$, where $\tilde{\mathcal{Y}}$ was the stencil last used to update y. By Corollary 3.10, if $\underline{v}_i(y) < \underline{v}_{i-1}(y)$, then $\underline{v}_i(y) > \underline{v}(x_i)$. So, for all $y \in \mathcal{H}_i$, $\underline{v}_i(y) \ge \underline{v}(x_i)$. By Line 5, $\underline{v}(x_{i+1}) = \min_{y \in \mathcal{H}_i} \underline{v}_i(y) \ge \underline{v}(x_i)$, which proves the inductive step. □

Proposition 3.13. *Let Property 3.8 hold. Let Property 3.9 hold for all $x \in \mathcal{X}$ and $\mathcal{Y} = \overrightarrow{\mathcal{Y}}(x)$. Then $\underline{u} = \underline{v}$ is the unique solution of (3.1).*

Proof. By Lemma 3.11, Algorithm 1 terminates. Let $x \in \mathcal{X}$. First, consider the case $x \in \underline{\partial\Omega}$. After executing Line 2, $\underline{v}_1(x) = g(x)$. Because $\overrightarrow{\mathcal{Y}}(x) = \emptyset$,

$x \notin \overleftarrow{\mathcal{Y}}(y)$ for any $y \in \mathcal{X}$. By the **foreach** loop condition x is not updated, and after Algorithm 1 terminates $\underline{v}(x) = \underline{v}_1(x) = g(x)$. Thus, $\underline{u}(x) = \underline{v}(x)$ is the unique solution to (3.1b). Second, consider the case $x \in \underline{\Omega}$. By Lemma 3.12, Line 8, and the definition of `Update`, $\mu = \underline{v}(x)$ is the unique solution to $\underline{H}(x, \mathcal{Y}^-, \underline{v}, \mu) = 0$, where $\mathcal{Y}^- = \{y \in \overrightarrow{\mathcal{Y}}(x) \mid \underline{v}(y) < \underline{v}(x)\}$. By Property 3.9, $\mu = \underline{v}(x)$ is also the unique solution to $\underline{H}(x, \overrightarrow{\mathcal{Y}}(x), \underline{v}, \mu) = 0$, which is (3.1a). □

Chapter 4

FMM for Axis-Aligned HJ PDEs

The material in this chapter is based on [4].

4.1 Introduction

The Fast Marching Method (FMM) [55, 64] has become a popular algorithm to use when solving the Dirichlet problem for an isotropic static HJ PDE $\|Du(x)\|_2 = c(x)$, also known as the Eikonal equation. While the isotropic case is the most common, there are applications which require solution of anisotropic HJ PDEs. Unfortunately, FMM produces a correct approximation only under certain causality conditions on the values of nodes and their neighbors. This limitation has motivated the development of a more generally applicable version of FMM called Ordered Upwind Methods (OUMs) [59] and also several works such as [36, 50, 62] on sweeping methods. However, OUMs are much more complex to implement than FMM, and sweeping methods can be much less efficient for problems with curved characteristics and practical grid sizes [30, 34].

Consequently, we have motivation to seek classes of anisotropic problems to which FMM might still be applied. One such class of problems was identified in [58] and includes the Eikonal equation where an energy norm re-

places the standard Euclidean norm (i.e., an elliptical speed profile replaces a spherical one). In [1] we identified another such class of problems. Because its characteristics are minimum time paths to the boundary, the Eikonal equation has often been proposed for robotic path planning; for example, see [40]. However, for some robots using the Euclidean norm in this equation is inappropriate. Consider a robot arm, where each joint has its own motor. If each motor can rotate at some maximum speed independent of the action of the other motors, then the action of the whole arm is best bounded in an approprately-scaled infinity norm. The corresponding Eikonal equation should use the dual Manhattan norm and is thus anisotropic. Other scenarios where such problems arise were considered in [1] and experimental evidence suggested that FMM would be successful on these problems.

As a group, the anisotropy in these problems is axis-aligned. In this chapter we describe a broader class of such axis-aligned problems (Section 4.2) and demonstrate that FMM can be applied to approximate their solution on axis-aligned orthogonal grids without modification of the algorithm beyond the local update function for a single node (Section 4.3). In Section 4.4 we propose some methods by which the local update's efficiency might be improved even if the grid and/or PDE lack symmetry. In Section 4.5 we provide analytic update formulas for the Eikonal equation with the $p = 1$, 2, and ∞ norms on variably-spaced orthogonal grids in any dimension. Lastly, the examples in Section 4.6 include an anelliptic wave propagation problem [28] and a new multirobot scenario.

Accurate robotic path planning is only required in cluttered environments where optimal paths—and hence the characteristics of the HJ PDE—are not straight. No alternative algorithm proposed approaches the simple implementation and guaranteed speed of FMM for these types of problems. Consequently, we set out in this chapter to characterize another class of anisotropic HJ PDEs for which FMM will work, and also to explore their efficient implementation. It should be noted that the update procedures discussed in this chapter can be applied to any of the sweeping algorithms without modification.

4.1.1 The Fast Marching Method

We compute an approximate solution $\underline{u}^{\mathcal{G}}$ on an axis-aligned orthogonal grid $\mathcal{G}$; for example, see Figure 1.5. We allow any axis-aligned orthogonal grid, including those with node spacing that varies between dimensions and within a single dimension; the latter capability makes it easier to more accurately manage an irregular boundary [30]. It is important that the orthogonal grid and the Hamiltonian H are aligned to the same axis. What it means for H to be aligned to an axis is formalized in Section 4.2.

Whenever we refer to a simplex of a node x, we mean a simplex specified by d neighbors of x, each in a distinct dimension. Since we are restricted to orthogonal grids, each simplex of x corresponds to a particular orthant.

A major contribution of this chapter is to demonstrate that for a class of static HJ PDEs with axis-aligned anisotropy, an `Update` function that is consistent with the PDE and satisfies the causality property can be defined; thus, FMM can be used. Because we restrict our modifications of Algorithm 1 (p.35) to the `Update` function, all of the results here can be used with other versions of FMM; for example, the $\mathcal{O}(N)$ algorithm described in [67], which uses an untidy priority queue for $\mathcal{H}$ to reduce the cost of `ExtractMin` and hence the whole algorithm.

4.2 Class of Hamiltonians

FMM can be extended to handle a class of axis-aligned anisotropic problems defined by a Hamiltonian H that satisfies Assumptions 4.3 to 4.6. We let $q, \tilde{q} \in \mathbb{R}^d$ and make the definitions:

Definition 4.1. *Write $q \trianglerighteq \tilde{q}$ if $q_j \tilde{q}_j \geq 0$ and $|q_j| \geq |\tilde{q}_j|$, for $1 \leq j \leq d$.*

Definition 4.2. *Write $q \triangleright \tilde{q}$ if (i) $q \neq 0$ and (ii) $q_j \tilde{q}_j \geq 0$ and $|q_j| > |\tilde{q}_j|$ or $q_j = \tilde{q}_j = 0$, for $1 \leq j \leq d$.*

The following assumptions are satisfied by H:

Assumption 4.3 (Continuity). $H \in C(\Omega \times \mathbb{R}^d)$.

Assumption 4.4 (Coercivity). *For all $x \in \Omega$, $H(x,q) \to \infty$ as $\|q\| \to \infty$.*

Assumption 4.5 (Strict Compatibility). *For all $x \in \Omega$, $H(x,0) < 0$.*

Assumption 4.6 (Strict One-Sided Monotonicity). *For all $x \in \Omega$, if $q \rhd \tilde{q}$, then $H(x,q) > H(x,\tilde{q})$.*

We note that Assumptions 4.3, 4.4, and 4.5 are similar to some assumed properties on the Hamiltonian in [11]. We usually consider a fixed $x \in \Omega$ and may write $H(q) = H(x,q)$ whenever no ambiguity results. When discussing properties of H they are in reference to the q parameter. The source of the *axis-aligned* description of the problem class is the strict one-sided monotonicity property of H.

4.2.1 Connection to Osher's criterion

Although there are earlier statements of the conditions on node values under which a Dijkstra-like algorithm can or cannot be used to solve the problem [55, 63], in this section we outline the connection between the properties described above and Osher's criterion [47] because the latter directly provides a condition on the Hamiltonian rather than on the solution values. In Section 4.3.2, we make the connection between Assumptions 4.3 to 4.6 and the earlier conditions.

Osher's fast marching criterion is defined in [47, 62] as

$$q_j \frac{\partial H(x,q)}{\partial q_j} \geq 0$$

for $1 \leq j \leq d$. The authors state there that as long as this criterion is satisfied, a simple fast marching algorithm based on a one-sided upwind finite-difference discretization can be applied to solve the problem. However, we use Assumptions 4.3 to 4.6 instead of Osher's criterion because Osher's criterion requires H to be differentiable so that $D_q H(x,q)$ exists, but we are interested in potentially nondifferentiable H (e.g., see Section 4.2.2). Note that strict one-sided monotonicity is applicable even when $D_q H(x,q)$ does not exist for all x.

Propositions 4.8, 4.9, and 4.11 explain the relationship between strict one-sided monotonicity of H (Assumption 4.6) and Osher's criterion. Proposition 4.8 shows that Assumption 4.6 implies one-sided monotonicity (Property 4.7). Then, Proposition 4.9 shows that Property 4.7 is the same as Osher's criterion as long as H is differentiable. Finally, Proposition 4.11 demonstrates that Property 4.7 with the addition of one-sided homogeneity (Property 4.10) implies Assumption 4.6.

Property 4.7 (One-Sided Monotonicity). *For all $x \in \Omega$, if $q \trianglerighteq \tilde{q}$, then $H(x, q) \geq H(x, \tilde{q})$.*

Proposition 4.8. *Let H be continuous (Assumption 4.3). Then strict one-sided monotonicity of H (Assumption 4.6) implies one-sided monotonicity of H (Property 4.7).*

Proof. Let H be strictly one-sided monotone. Let $q, \tilde{q} \in \mathbb{R}^d$ be such that $q \trianglerighteq \tilde{q}$. Let $\kappa \in \{-1, 1\}^d$ be such that

$$\kappa_j = \begin{cases} +1, & \text{if } q_j \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

and let $\epsilon > 0$. Note that $q + \epsilon\kappa \triangleright \tilde{q}$ and thus we have $H(q + \epsilon\kappa) > H(\tilde{q})$. Therefore, by the continuity of H, we have

$$H(q) = \lim_{\epsilon \to 0^+} H(q + \epsilon\kappa) \geq H(\tilde{q}).$$

□

Proposition 4.9. *Let H be continuous (Assumption 4.3) and let $D_q H(q)$ exist for all $q \in \mathbb{R}^d$. Then the following conditions on H are equivalent.*

(a) $q_j \frac{\partial H(q)}{\partial q_j} \geq 0$, for all j and $q \in \mathbb{R}^d$.

(b) H is one-sided monotone (Property 4.7).

Proof. We begin by proving that (a) implies (b). Let $q, \tilde{q} \in \mathbb{R}^d$ be such that $q \trianglerighteq \tilde{q}$. If $q = \tilde{q}$ then $H(q) = H(\tilde{q})$. Otherwise, define the function

$\bar{q} : [0,1] \to \mathbb{R}^d$ such that $\bar{q}(t) = \tilde{q} + t(q - \tilde{q})$ represents the line segment between $\tilde{q}$ and q parameterized by $t \in [0,1]$. Because $q \trianglerighteq \tilde{q}$ we have

$$(q - \tilde{q})_j \bar{q}_j(t) \geq 0$$

for $1 \leq j \leq d$ and for $t \in [0,1]$. Thus, by condition (a) we have

$$(q - \tilde{q})_j \frac{\partial H(\bar{q}_j(t))}{\partial \bar{q}_j(t)} \geq 0, \tag{4.1}$$

for $1 \leq j \leq d$ and for $t \in [0,1]$. We know that

$$\begin{aligned} H(q) &= H(\tilde{q}) + \int_0^1 \frac{d\bar{q}(t)}{dt} \cdot D_q H(\bar{q}(t)) dt \\ &= H(\tilde{q}) + \int_0^1 (q - \tilde{q}) \cdot D_q H(\bar{q}(t)) dt \\ &= H(\tilde{q}) + \int_0^1 \sum_{j=1}^{n} (q - \tilde{q})_j \frac{\partial H(\bar{q}_j(t))}{\partial \bar{q}_j(t)} dt \\ &\geq H(\tilde{q}). \end{aligned}$$

The first equality follows from integrating the change in H along the line segment connecting $\tilde{q}$ and q. The second equality is because the derivative $\frac{d\bar{q}(t)}{dt}$ is simply the vector $q - \tilde{q}$. The third equality breaks up the vector dot product into a sum of scalar products. The inequality results from (4.1) and the fact that an integral of a nonnegative function is nonnegative. Thus for all $q, \tilde{q}$ such that $q \trianglerighteq \tilde{q}$, including $q = \tilde{q}$, we have $H(q) \geq H(\tilde{q})$.

We now prove that (b) implies (a). Let $q \in \mathbb{R}^d$ and $1 \leq j \leq d$. Define the function $\sigma : \mathbb{R} \to \{-1, +1\}$ such that

$$\sigma(y) = \begin{cases} +1, & \text{if } y \geq 0 \\ -1, & \text{otherwise,} \end{cases}$$

let $\epsilon > 0$, and let e_j be the jth vector in the standard basis. Note that $q + \epsilon\sigma(q_j)e_j \trianglerighteq q$ and thus by (b) we have $H(q + \epsilon\sigma(q_j)e_j) - H(q) \geq 0$.

Consequently, by the existence of $D_q H(q)$ for all $q \in \mathbb{R}^d$ we have

$$q_j \frac{\partial H(q)}{\partial q_j} = \lim_{\epsilon \to 0^+} q_j \frac{H(q + \epsilon\sigma(q_j)e_j) - H(q)}{\epsilon\sigma(q_j)} \geq 0.$$

□

The following property is used to state Proposition 4.11.

Property 4.10 (One-sided Homogeneity). *For all $x \in \Omega$, $t \geq 0$, and $q \in \mathbb{R}^d$: $H(x, tq) - H(x, 0) = t(H(x, q) - H(x, 0))$.*

Proposition 4.11. *Let H satisfy Assumptions 4.3, 4.4, and 4.5, and let H be one-sided monotone (Property 4.7) and one-sided homogeneous (Property 4.10). Then H is strictly one-sided monotone (Assumption 4.6).*

Proof. Let $q \rhd \tilde{q}$. Then $q \trianglerighteq \tilde{q}$ and $H(q) \geq H(\tilde{q})$ by one-sided monotonicity.

First consider the case $\tilde{q} = 0$. Assume $H(q) = H(\tilde{q}) = H(0)$. By the one-sided homogeneity of H,

$$\lim_{t \to \infty} [H(tq) - H(0)] = \lim_{t \to \infty} [t(H(q) - H(0))] = 0.$$

But by the coercivity of H

$$\lim_{t \to \infty} [H(tq) - H(0)] = \infty,$$

since $\lim_{t \to \infty} \|tq\| = \infty$ and by compatability $H(0) < 0$. Thus we have a contradiction and it must be that $H(q) > H(\tilde{q})$.

Second, consider the case where $\tilde{q} \neq 0$. Let $\mathcal{J} = \{j \mid |q_j| > |\tilde{q}_j|\}$. By Definition 4.2, we have $\mathcal{J} \neq \emptyset$, and since $\tilde{q} \neq 0$ there is some $j \in \mathcal{J}$ such that $\tilde{q}_j \neq 0$. Define a scalar multiple of q:

$$\check{q} = tq = \left(\max_{j \in \mathcal{J}} \frac{|\tilde{q}_j|}{|q_j|} \right) q.$$

Since $|q_j| > |\tilde{q}_j|, \forall j \in \mathcal{J}$, we have $0 < t < 1$. Furthermore, for $j \in \mathcal{J}$,

$$|\check{q}_j| = \left(\max_{j \in \mathcal{J}} \frac{|\tilde{q}_j|}{|q_j|} \right) |q_j| \geq |\tilde{q}_j|,$$

while for $j \notin \mathcal{J}$,

$$\check{q}_j = tq_j = 0 = \tilde{q}_j.$$

Consequently, $|\check{q}_j| \geq |\tilde{q}_j|$ for $1 \leq j \leq d$. Also, since $t > 0$, we have $\check{q}_j \tilde{q}_j = tq_j \tilde{q}_j \geq 0$ for $1 \leq j \leq d$. This implies, by one-sided monotonicity of H, that $H(\check{q}) \geq H(\tilde{q})$. Moreover, by one-sided homogeneity of H,

$$H(\check{q}) - H(0) = H(tq) - H(0) = t(H(q) - H(0)).$$

It follows that

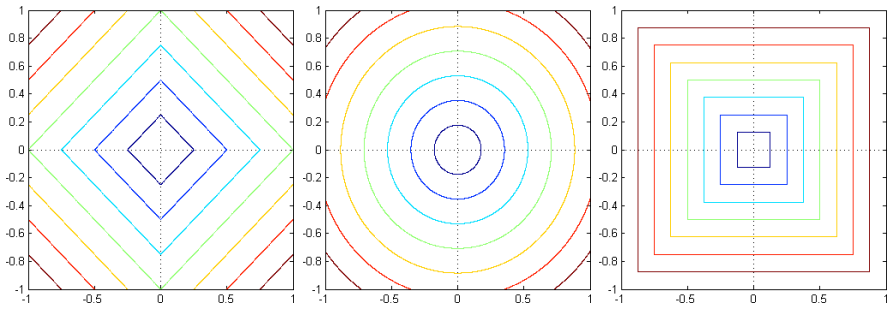
$$H(q) - H(0) = (H(\check{q}) - H(0))/t > H(\check{q}) - H(0),$$

since $0 < t < 1$ and $H(\check{q}) \geq H(0)$ by one-sided monotonicity. Therefore, $H(q) > H(\check{q}) \geq H(\tilde{q})$. □

We impose strict one-sided monotonicity on H because it guarantees a unique solution to a first-order upwind finite-difference discretization of (1.8a), as shown in Section 4.3.1. Simply imposing one-sided monotonicity on H or Osher's condition on differentiable H is not sufficient for a unique solution. However, Proposition 4.11 states that when H satisfies one-sided homogeneity in addition to one-sided monotonicity then it also satisfies strict one-sided monotonicity and there is a unique solution to the discretization. Moreover, by Propositions 4.9 and 4.11, when differentiable H satisfies one-sided homogeneity in addition to Osher's criterion then H also satisfies strict one-sided monotonicity and there is a unique solution to the discretization. Note that there exist conditions other than one-sided homogeneity, such as strict convexity, that in combination with Osher's criterion result in strict one-sided monotonicity of H.

4.2.2 Example Hamiltonians

A Hamiltonian H that satisfies Assumptions 4.3 to 4.6 encompasses a fairly broad range of anisotropic problems. We consider examples of such H. In



(a) $p = 1$ (b) $p = 2$ (c) $p = \infty$

Figure 4.1: Contour plots of $\|q\|_p$.

particular, we look at the case

$$H(x, q) = G(x, q) - 1, \tag{4.2}$$

where G is a p-norm or some variant, such as a linearly-tranformed p-norm or a mixed p-norm defined below. We must ensure that G is strictly one-sided monotone, which is not true of all norms.

The p-norm is a useful category of strictly one-sided monotone norms. Let a p-norm, $\|\cdot\|_p$, be defined by

$$\|q\|_p = \left(\sum_{j=1}^{d} |q_j|^p \right)^{1/p},$$

where $p \geq 1$. Commonly used p-norms, illustrated in Figure 4.1, are the Manhattan norm ($p = 1$), the Euclidean norm ($p = 2$), and the maximum norm ($p = \infty$).

Proposition 4.12. *$\|\cdot\|_p$ is strictly one-sided monotone, for $p \geq 1$.*

Proof. Let $q \rhd \tilde{q}$. We know that $|q_j| \geq |\tilde{q}_j| \geq 0$ for all j, such that $1 \leq j \leq d$. Furthermore, $|q_j| > |\tilde{q}_j|$ for at least one j, such that $1 \leq j \leq d$.

First, consider finite $p \geq 1$. By the properties of x^p, we have $|q_j|^p \geq$

$|\tilde{q}_j|^p \geq 0$ for all j. Furthermore, $|q_j|^p > |\tilde{q}_j|^p$ for at least one j. This, in turn, implies

$$\sum_{j=1}^{d} |q_j|^p > \sum_{j=1}^{d} |\tilde{q}_j|^p \geq 0.$$

It follows that

$$\|q\|_p = \left(\sum_{j=1}^{d} |q_j|^p \right)^{1/p} > \left(\sum_{j=1}^{d} |\tilde{q}_j|^p \right)^{1/p} = \|\tilde{q}\|_p.$$

Consider the case $p = \infty$ separately:

$$\|q\|_\infty = \lim_{p \to \infty} \left(\sum_{j=1}^{d} |q_j|^p \right)^{1/p} = \max_{1 \leq j \leq d} |q_j|.$$

We have $\max_{1 \leq j \leq d} |q_j| > \max_{1 \leq j \leq d} |\tilde{q}_j|$.

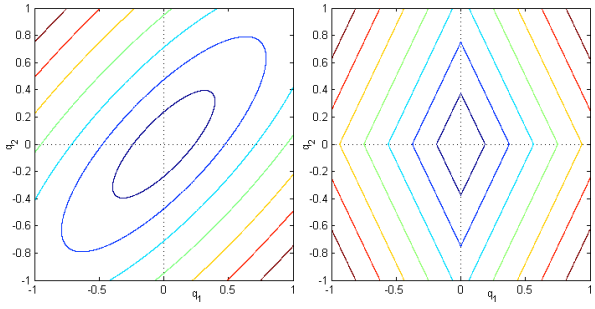
Therefore, for both the case where $p \geq 1$ is finite and $p = \infty$, $\|\cdot\|_p$ is strictly one-sided monotone. □

Define a linearly-transformed p-norm, $\|\cdot\|_{B,p}$, to be

$$\|q\|_{B,p} = \|Bq\|_p,$$

where $p \geq 1$ and B is a nonsingular $d \times d$ matrix. Note that B must be nonsingular so that $\|\cdot\|_{B,p}$ satisfies the properties of a norm such as definiteness and homogeneity. Such a norm is not strictly one-sided monotone in general. Figure 4.2(a) shows a simple example where a vector is rotated by $-\pi/4$ and scaled by 3 in the q_2-axis before the Euclidean norm is taken; i.e.,

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} \cos(-\pi/4) & -\sin(-\pi/4) \\ \sin(-\pi/4) & \cos(-\pi/4) \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -3/\sqrt{2} & 3/\sqrt{2} \end{bmatrix}. \tag{4.3}$$



(a) (b)

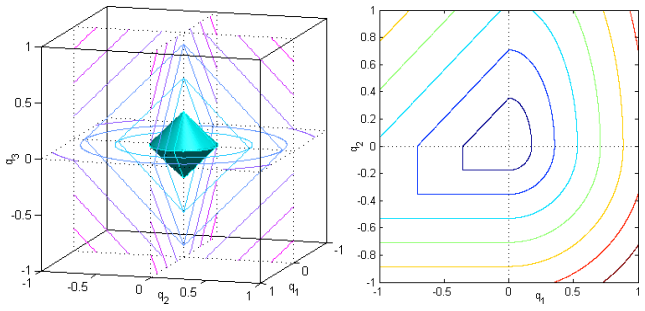
Figure 4.2: Contour plots of $\|Bq\|_p$. (a) is not strictly one-sided monotone: $p = 2$ and B is defined by (4.3). (b) is strictly one-sided monotone: $p = 1$ and B scales by 2 in the q_1-axis.

Let $q = (2, 2)^T$ and $\tilde{q} = (\sqrt{2}, 0)^T$. We have $q \trianglerighteq \tilde{q}$, but,

$$\|Bq\|_2 = \|(2\sqrt{2}, 0)^T\|_2 = \sqrt{8} < \sqrt{10} = \|(1, -3)^T\|_2 = \|B\tilde{q}\|_2.$$

Consequently, this particular linearly-transformed p-norm is not strictly one-sided monotone. However, in this case an inverse transformation B^{-1} of the grid coordinates will result in a strictly one-sided monotone p-norm, while maintaining the grid's orthogonality. More generally, we conjecture that if the Hamiltonian is of the form $H(q) = \tilde{H}(Bq)$, where B is a rotation (which may be followed by scaling) and $\tilde{H}$ satisfies Assumptions 4.3 to 4.6, a transformation of the grid coordinates by B^{-1} will result in a transformed H that also satisfies Assumptions 4.3 to 4.6, while maintaining the grid's orthogonality. More complex coordinate modifications might be possible but we have not yet adequately investigated conditions or procedures.

A scaled p-norm (Figure 4.2(b)) is a special case of a linearly-transformed p-norm. Such a norm scales the components of its argument before applying a p-norm, by restricting B to be a nonsingular diagonal matrix. It is simple to show that a scaled p-norm is strictly one-sided monotone, considering Proposition 4.12.



(a) (b)

Figure 4.3: Contour plots of $G(q)$. (a) mixed p-norm: G is defined by (4.4). (b) asymmetric norm-like function: G is defined by (4.5).

A mixed p-norm is a nested composition of p-norms and it is strictly one-sided monotone. The following is an example (Figure 4.3(a)) of a mixed p-norm that takes the Euclidean norm of the first 2 components and then takes the Manhattan norm of the result and the last component:

$$\begin{aligned} \|q\| &= \| \left(\| \left(q_1, q_2 \right) \|_2, q_3 \right) \|_1 \\ &= \sqrt{(q_1)^2 + (q_2)^2} + |q_3|. \end{aligned} \tag{4.4}$$

where $q = (q_1, q_2, q_3)$. This particular norm was used as a G function in [1] for a simple 2-robot coordinated optimal control problem (see Figure 2.1).

Finally, the one-sidedness of Assumption 4.6 allows G to be asymmetric, which is not permitted for a norm. An example of such an asymmetric norm-like function is shown in Figure 4.3(b) and is given by

$$G(q) = \begin{cases} \|B_a q\|_\infty, & \text{if } q_1 \leq 0 \text{ and } q_2 \leq 0, \\ \|B_b q\|_1, & \text{if } q_1 \leq 0 \text{ and } q_2 > 0, \\ \|B_c q\|_2, & \text{if } q_1 > 0 \text{ and } q_2 \leq 0, \\ \|B_d q\|_2, & \text{if } q_1 > 0 \text{ and } q_2 > 0, \end{cases} \tag{4.5}$$

where

$$B_a = \begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix} \quad B_b = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \quad B_c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B_d = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}.$$

We solve the anisotropic problem characterized by (4.5) as well as an anelliptic wave propagation problem and a multi-robot optimal path planning problem in Section 4.6. Other examples of G functions which satisfy strict one-sided monotonicity are some polygonal norms such as axis-aligned hexagonal or octagonal norms; however, we do not further investigate these options here.

4.3 Discretization

We define a discrete analogue of the Dirichlet problem (1.8). By describing the `Update` function in Algorithm 1, we also formalize the Dijkstra-like algorithm. Finally, we prove important properties of the numerical scheme and corresponding `Update` function to show that the solution to the discrete problem converges to the solution to (1.8) and that Algorithm 1 solves the discrete problem.

Let $x \in \Omega$. The stencil of x is shown in Figure 4.4. Let $x_j^{\pm} = x + h_j^{\pm} e_j$ be the stencil node in $\pm e_j$ direction, where e_j is the j^{th} vector in the standard orthogonal basis and $h_j^{\pm}$ is a signed distance along the j^{th} axis from x to $x_j^{\pm}$. Let $\mathcal{N} \subseteq \{x_j^{\pm} \mid 1 \leq j \leq d\}$ be the set of nodes used in the stencil. Let

$$\mathcal{K} = \{(\kappa_1, \kappa_2, \ldots, \kappa_d) \mid \kappa_j \in \{-1, +1\}, 1 \leq j \leq d\},$$

such that $\kappa \in \mathcal{K}$ represents one of the 2^d neighboring simplices of x. Note that we abuse notation by using $\kappa_j \in \{-1, +1\}$ as a superscript indexing $x_j^{\pm}$ or $h_j^{\pm}$.

We define the numerical Hamiltonian $\underline{H}$ as follows:

$$\underline{H}(x, \mathcal{N}, \phi, \mu) = \max_{\kappa \in \mathcal{K}} [H(x, \underline{D}^{\kappa}(x, \mathcal{N}, \phi, \mu))], \tag{4.6}$$

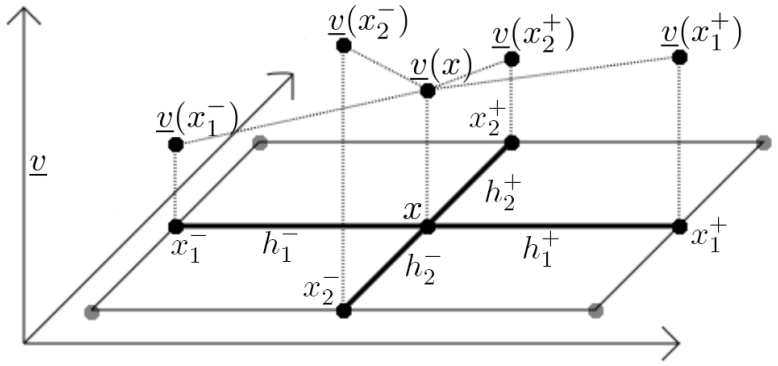


Figure 4.4: Neighborhood of x with $d = 2$

where H is as defined in Section 4.2 and

$$\underline{D}^{\kappa}(x, \mathcal{N}, \phi, \mu) = (\underline{D}_1^{\kappa}(x, \mathcal{N}, \phi, \mu), \underline{D}_2^{\kappa}(x, \mathcal{N}, \phi, \mu), \ldots, \underline{D}_d^{\kappa}(x, \mathcal{N}, \phi, \mu))$$

is a first-order, upwind, finite-difference gradient approximation from the simplex represented by κ; that is,

$$\underline{D}_j^{\kappa}(x, \mathcal{N}, \phi, \mu) = \begin{cases} \frac{\max(0, \mu - \phi(x_j^{\kappa_j}))}{-h_j^{\kappa_j}}, & \text{if } x_j^{\kappa_j} \in \mathcal{N} \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

for $1 \leq j \leq d$.

Given an axis-aligned orthogonal grid $\mathcal{G}$, the discretized Dirichlet problem is to find a function $\underline{u}^{\mathcal{G}} : \overline{\Omega} \to \mathbb{R}$, such that (3.1) is satisfied and $\underline{u}^{\mathcal{G}}(x)$ satisfies a non-expansive interpolation, for $x \in \overline{\Omega} \setminus \mathcal{X}$. For this discretization the stencil is the set of grid neighbors: $\overrightarrow{\mathcal{Y}}(x) = \overleftarrow{\mathcal{Y}}(x) = \mathcal{N}(x)$, for $x \in \underline{\Omega}$.

Let the `Update` function in Algorithm 1 be defined as follows. A call to `Update`(x) returns the solution $\mu = \tilde{\mu}$ to

$$\underline{H}(x, \mathcal{N}(x) \setminus \mathcal{H}, \underline{v}, \mu) = 0. \tag{4.8}$$

In this way it determines a node's value $\underline{v}(x) \leftarrow \tilde{\mu}$ given the values of its

accepted neighbors, $\underline{v}(x_j^{\pm})$.

Proposition 3.13 states that the grid function $\underline{v}$ that results from running Algorithm 1 is the unique solution of the discretized problem (3.1), provided that (4.8) has a unique solution (Property 3.8) and that the numerical scheme is causal (Property 3.9). We prove these properties in Sections 4.3.1 and 4.3.2. Also, the uniqueness of the solution to (4.8) and the fact that $H(\mu)$ is nondecreasing in μ (Lemma 4.14) are useful for using numerical root finders to implement `Update`.

Proposition 3.7 states that the solution $\underline{u}^{\mathcal{G}}$ converges to u the solution of (1.8), if the numerical scheme is consistent, monotone, and stable (Properties 3.2, 3.3, and 3.4 respectively). We include proofs of these properties in Sections 4.3.3, 4.3.4, and 4.3.5.

When we are varying only μ, it will be convenient to write $\underline{H}(\mu) = \underline{H}(x, \mathcal{N}, \phi, \mu)$ and $\underline{D}^{\kappa}(\mu) = \underline{D}^{\kappa}(x, \mathcal{N}, \phi, \mu)$. For the lemmas and theorems stated in this section we assume H satisfies Assumptions 4.3 to 4.6.

4.3.1 Unique Update

Let $\phi \in B(\overline{\Omega})$. Define

$$\check{\phi}_{\mathcal{N}} = \min_{x \in \mathcal{N}} \phi(x). \tag{4.9}$$

We show in Theorem 4.15 there is a unique solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{N}, \phi, \mu) = 0$ such that $\tilde{\mu} > \check{\phi}_{\mathcal{N}}$. In other words, we prove Property 3.8 (Unique Solution) and in addition a lower bound on the solution $\tilde{\mu}$. First, we prove two useful lemmas.

Lemma 4.13. *$\underline{H}(\mu)$ is strictly increasing on $\mu \geq \check{\phi}_{\mathcal{N}}$.*

Proof. Let $\mu_a > \mu_b \geq \check{\phi}_{\mathcal{N}}$. In order to prove the lemma, we must show $\underline{H}(\mu_a) > \underline{H}(\mu_b)$. Let $\kappa \in \mathcal{K}$ and $1 \leq j \leq d$. If $\mu_a > \underline{u}_j^{\kappa_j}$ then $\underline{D}_j^{\kappa}(\mu_a)\underline{D}_j^{\kappa}(\mu_b) \geq 0$ and $|\underline{D}_j^{\kappa}(\mu_a)| > |\underline{D}_j^{\kappa}(\mu_b)|$. On the other hand, if $\mu_a \leq \underline{u}_j^{\kappa_j}$ then $\underline{D}_j^{\kappa}(\mu_a) = \underline{D}_j^{\kappa}(\mu_b) = 0$. Also, there exists at least one $\kappa \in \mathcal{K}$ and $1 \leq j \leq d$ such that $\underline{D}_j^{\kappa}(\mu_a) \neq 0$, since $\mu_a > \check{\phi}_{\mathcal{N}}$. For such κ, $H(\underline{D}^{\kappa}(\mu_a)) > H(\underline{D}^{\kappa}(\mu_b))$, by strict one-sided monotonicity (Assumption 4.6). For all other κ, $H(\underline{D}^{\kappa}(\mu_a)) = H(\underline{D}^{\kappa}(\mu_b)) = H(0)$. Therefore, by (4.6), $\underline{H}(\mu_a) > \underline{H}(\mu_b)$. □

Lemma 4.14. *The numerical Hamiltonian $\underline{H}(\mu)$ satisfies the following.*

(a) $\underline{H}(\mu) = H(0) < 0$ *for* $\mu \leq \check{\phi}_{\mathcal{N}}$.

(b) $\underline{H}(\mu) \to \infty$ *as* $\mu \to \infty$.

(c) $\underline{H}(\mu)$ *is nondecreasing on all* μ.

Proof. If $\mu \leq \check{\phi}_{\mathcal{N}}$ then by (4.7) and (4.9), we have $\underline{D}_j^{\kappa}(\mu) = 0$ for all $\kappa \in \mathcal{K}$, $1 \leq j \leq d$. By the strict compatibility of H, $H(\underline{D}^{\kappa}(v_j)) = H(0) < 0$, for all κ. By (4.6), we have $\underline{H}(\mu) = H(0) < 0$, for $\mu \leq \check{\phi}_{\mathcal{N}}$, proving (a).

Let $\kappa \in \mathcal{K}$ and $1 \leq j \leq d$. As $\mu \to \infty$, we have $\underline{D}_j^{\kappa}(\mu) \to \infty$ and $\|\underline{D}^{\kappa}(\mu)\| \to \infty$ for all $\kappa \in \mathcal{K}$, $1 \leq j \leq d$. By the coercivity of H, as $\mu \to \infty$, we have $H(\underline{D}^{\kappa}(\mu)) \to \infty$ for all $\kappa \in \mathcal{K}$. By (4.6), we have $\underline{H}(\mu) \to \infty$ as $\mu \to \infty$, proving (b).

Because $\underline{H}(\mu)$ is constant on $\mu \leq \check{\phi}_{\mathcal{N}}$ and by Lemma 4.13 increasing on $\mu \geq \check{\phi}_{\mathcal{N}}$, $\underline{H}(\mu)$ is nondecreasing on all μ, proving (c). □

Theorem 4.15. *There exists a unique solution* $\mu = \tilde{\mu}$ *to* $\underline{H}(\mu) = 0$ *such that* $\tilde{\mu} > \check{\phi}_{\mathcal{N}}$.

Proof. Each $\underline{D}_j^{\kappa}(\mu)$ is continuous on μ. Furthermore, by the continuity of H, $H(\underline{D}^{\kappa}(\mu))$ is continuous on μ for all κ. Since max is continuous, $\underline{H}(\mu)$ is continuous. By Lemma 4.14(a) and 4.14(b), $\underline{H}(\mu) < 0$ for $\mu \leq \check{\phi}_{\mathcal{N}}$ and $\underline{H}(\mu) \to \infty$ as $\mu \to \infty$. Therefore, by the Intermediate Value Theorem there exists a solution $\mu = \tilde{\mu}$ to $\underline{H}(\mu) = 0$, such that $\check{\phi}_{\mathcal{N}} < \tilde{\mu} < \infty$. Moreover, since $\underline{H}$ is strictly increasing on $\mu \geq \check{\phi}_{\mathcal{N}}$ by Lemma 4.13, the solution is unique. □

Remark 4.16. *We note that strict one-sided monotonicity (Assumption 4.6) of* H *is used to prove Lemma 4.13, and Lemma 4.13 is then used to show that the solution to* $\underline{H}(\mu) = 0$ *is unique. We might consider whether or not one-sided monotonicity (Property 4.7) of* H *is sufficient for a unique solution. However, Property 4.7 would not be sufficient to prove Lemma 4.13 and we would find that* $\underline{H}(\mu)$ *is only nondecreasing on* $\mu \geq \check{\phi}_{\mathcal{N}}$. *A solution to* $\underline{H}(\mu) = 0$ *would still be guaranteed but not necessarily unique in this case.*

Analogously, for differentiable H, Osher's criterion on H implies a solution that may not be unique unless H satisfies some additional property, such as one-sided homogeneity (Property 4.10) or convexity.

4.3.2 Causality

The following theorem states that $\underline{H}$ and the `Update` function are causal. The `Update` function is considered causal if the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{N}, \underline{u}, \mu) = 0$, does not change after all nodes with values greater than or equal to $\tilde{\mu}$ are removed from the stencil.

Theorem 4.17. *Property 3.9 (Causality) holds.*

Proof. Let Property 3.8 hold. Let $\mu = \tilde{\mu}$ be the unique solution to $\underline{H}(x, \mathcal{N}, \phi, \mu) = 0$. Let $\mathcal{N}^- = \{y \in \mathcal{N} \mid \phi(y) < \tilde{\mu}\}$. By (4.7), we have $D_j^\kappa(x, \mathcal{N}, \phi, \tilde{\mu}) = D_j^\kappa(x, \mathcal{N}^-, \phi, \tilde{\mu})$, for all $\kappa \in \mathcal{K}$, $1 \leq j \leq d$. Thus, $\underline{H}(x, \mathcal{N}^-, \phi, \tilde{\mu}) = \underline{H}(x, \mathcal{N}, \phi, \tilde{\mu}) = 0$. Therefore, $\mu = \tilde{\mu}$ is also the unique solution to $\underline{H}(x, \mathcal{N}^-, \phi, \mu) = 0$. □

4.3.3 Monotonicity

We show that $\underline{H}$ and the `Update` function are monotone in the neighbor's values. Monotonicity of $\underline{H}$ requires that if none of the stencil node values decreases, the numerical Hamiltonian $\underline{H}$ does not increase. We note that monotonicity does not require strict one-sided monotonicity of H, but rather one-sided monotonicity of H is sufficient.

Theorem 4.18. *Property 3.3 (Monotonicity) holds.*

Proof. Let $x \in \Omega$ and $\mu \in \mathbb{R}$. Let functions $\check{\phi} : \overline{\Omega} \to \mathbb{R}$ and $\hat{\phi} : \overline{\Omega} \to \mathbb{R}$ be such that $\check{\phi}(y) \leq \hat{\phi}(y)$ for all $y \in \mathcal{N}$. We have $\underline{D}^\kappa(x, \mathcal{N}, \check{\phi}, \mu) \trianglerighteq \underline{D}^\kappa(x, \mathcal{N}, \hat{\phi}, \mu)$, for all $\kappa \in \mathcal{K}$. Also, by Proposition 4.8, H satisfies one-sided monotonicity (Property 4.7). Thus,

$$H(\underline{D}^\kappa(x, \mathcal{N}, \check{\phi}, \mu)) \geq H(\underline{D}^\kappa(x, \mathcal{N}, \hat{\phi}, \mu)),$$

for all $\kappa \in \mathcal{K}$. Consequently, $\underline{H}(x, \mathcal{N}, \check{\phi}, \mu) \geq \underline{H}(x, \mathcal{N}, \hat{\phi}, \mu)$, proving Property 3.3 holds. □

The following corollary states that if none of the stencil node values decreases, the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{N}, \phi, \mu) = 0$ does not decrease.

Corollary 4.19. *Let Property 3.8 hold. Let $x \in \Omega$ and $\mu \in \mathbb{R}$. Let functions $\check{\phi} : \overline{\Omega} \to \mathbb{R}$ and $\hat{\phi} : \overline{\Omega} \to \mathbb{R}$ be such that $\check{\phi}(y) \leq \hat{\phi}(y)$ for all $y \in \mathcal{Y}$. Let $\mu = \mu_{\check{\phi}}$ be the unique solution to $\underline{H}(x, \mathcal{N}, \check{\phi}, \mu) = 0$ and $\mu = \mu_{\hat{\phi}}$ be the unique solution to $\underline{H}(x, \mathcal{N}, \hat{\phi}, \mu) = 0$, then $\mu_{\check{\phi}} \leq \mu_{\hat{\phi}}$.*

Proof. By Property 3.3, $\underline{H}(x, \mathcal{N}, \check{\phi}, \mu_{\check{\phi}}) = 0 \geq \underline{H}(x, \mathcal{N}, \hat{\phi}, \mu_{\check{\phi}})$. By Lemma 4.14(c), $\underline{H}(x, \mathcal{N}, \hat{\phi}, \mu)$ is nondecreasing on all μ, so in order that $\underline{H}(x, \mathcal{N}, \hat{\phi}, \mu_{\hat{\phi}}) = 0$, it must be that $\mu_{\check{\phi}} \leq \mu_{\hat{\phi}}$. □

4.3.4 Consistency

We show that the numerical Hamiltonian $\underline{H}$ is consistent with (4.2). First, we note that since $|h_j^{\pm}| \leq \hat{h}^{\mathcal{G}}$ for all $\mathcal{G}$, Property 3.1 is satisfied.

Lemma 4.20. *Let $x \in \Omega$ and $\phi \in C_b^{\infty}(\Omega)$. Let H be continuous in the first argument and satisfy Assumptions 4.3 to 4.6. Define $\hat{h} = \max_{1 \leq j \leq d} h_j^{\pm}$. Then*

$$\lim_{\substack{y \to x,\, \xi \to 0 \\ \hat{h} \to 0}} \underline{H}(y, \mathcal{N}, \phi + \xi, \phi(y) + \xi) = H(x, D\phi(x)). \tag{4.10}$$

Proof. Let ϕ, x, and H be as defined above. Let

$$D\phi(x) = (\partial_1 \phi(x), \partial_2 \phi(x), \ldots, \partial_d \phi(x)).$$

Let $\kappa \in \mathcal{K}$ and $1 \le j \le d$. We have by (4.7) and the smoothness of ϕ

$$\begin{aligned}\lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} D_j^{\kappa}(y,\mathcal{N},\phi+\xi,\phi(y)+\xi) &= \lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} \frac{\max(0,\phi(y)+\xi-\phi(x_j^{\kappa_j})-\xi)}{-h_j^{\kappa_j}} \\ &= \lim_{y\to x,\ \hat{h}\to 0} \frac{\min(0,\phi(y+h_j^{\kappa_j}e_j)-\phi(y))}{h_j^{\kappa_j}} \\ &= \begin{cases} \partial_j\phi(x), & \text{if } \kappa_j\partial_j\phi(x)\le 0, \\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

Define $\tilde{\kappa}$ as

$$\tilde{\kappa}_j = \begin{cases} +1, & \text{if } \partial_j\phi(x)\le 0, \\ -1, & \text{otherwise,}\end{cases} \tag{4.11}$$

for $1 \le j \le d$. We have

$$\lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} D^{\tilde{\kappa}}(y,\mathcal{N},\phi+\xi,\phi(y)+\xi) = D\phi(x). \tag{4.12}$$

By the continuity and strict one-sided monotonicity of H,

$$\lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} H(y, D^{\tilde{\kappa}}(y,\mathcal{N},\phi+\xi,\phi(y)+\xi)) \ge \lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} H(y, D^{\kappa}(y,\mathcal{N},\phi+\xi,\phi(y)+\xi)),$$

for all κ. Therefore, by the continuity of max and H

$$\begin{aligned}
&\lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} \underline{H}(y,\mathcal{N},\phi+\xi,\phi(y)+\xi)\\
&= \lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} \max_{\kappa\in\mathcal{K}} H(y,D^{\kappa}(y,\mathcal{N},\phi+\xi,\phi(y)+\xi))\\
&= \lim_{\substack{y\to x,\ \xi\to 0,\\ \hat{h}\to 0}} H(y,D^{\tilde{\kappa}}(y,\mathcal{N},\phi+\xi,\phi(y)+\xi))\\
&= H(x,D\phi(x)).
\end{aligned}$$

□

Proposition 4.21. *Let H be continuous in the first argument and satisfy Assumptions 4.3 to 4.6. Property 3.2 (Consistency) holds.*

Proof. Let H satisfy the properties specified above. Let Property 3.1 hold. Let $x \in \Omega$ and $\phi \in C_b^\infty(\Omega)$. Let $\mathcal{G}_k$, $x_k \in \underline{\Omega}_k$, and ξ_k be sequences such that $x_k \to x$, $\hat{h}^{\mathcal{G}_k} \to 0$, and $\xi_k \to 0$ as $k \to \infty$. Then, by Lemma 4.20,

$$\begin{aligned}
&\lim_{k\to\infty} \underline{H}(x_k,\mathcal{Y}_k,\phi+\xi_k,\phi(x_k)+\xi_k)\\
&= \lim_{\substack{x_k\to x,\ \xi_k\to 0\\ \hat{h}\to 0}} \underline{H}(x_k,\mathcal{N},\phi+\xi_k,\phi(x_k)+\xi_k) = H(x,D\phi(x)),
\end{aligned} \tag{4.13}$$

so Property 3.2 holds. □

4.3.5 Stability

We show that for any reasonable orthogonal grid $\mathcal{G}$, the solution $\underline{u}^{\mathcal{G}}$ to (3.1) is bounded and so Property 3.4 (Stability) holds. The following lemma implies that the magnitude of the slope in $\underline{u}^{\mathcal{G}}$ as measured between two neighbors is bounded.

Lemma 4.22. *Define*

$$\hat{K} = \sup_{x\in\Omega,\, H(x,p)\leq 0} \|p\|.$$

Let $x \in \Omega$ and $\phi \in B(\overline{\Omega})$. If $\mu = \tilde{\mu}$ is the unique solution to $\underline{H}(x, \mathcal{N}, \phi, \mu) = 0$ then

$$|D_j^\kappa(x, \mathcal{N}, \phi, \tilde{\mu})| \leq \hat{K},$$

for all $\kappa \in \mathcal{K}$ and $1 \leq j \leq d$.

Proof. Note that $\hat{K}$ exists and is finite by the coercivity of H. Let $\kappa \in \mathcal{K}$ and $1 \leq j \leq d$. Assume $|D_j^\kappa(x, \mathcal{N}, \phi, \tilde{\mu})| > \hat{K}$. By the definition of $\hat{K}$,

$$H(x, -|D_j^\kappa(x, \mathcal{N}, \phi, \tilde{\mu})|\kappa_j e_j) > 0.$$

Thus, by the one-sided monotonicity of H and (3.1),

$$\begin{aligned} H(x, D^\kappa(x, \mathcal{N}, \phi, \tilde{\mu})) &\geq H(x, -|D_j^\kappa(x, \mathcal{N}, \phi, \tilde{\mu})|\kappa_j e_j) \\ &> 0 = \underline{H}(x, \mathcal{N}, \phi, \tilde{\mu}), \end{aligned}$$

contradicting (4.6). Therefore, $|D_j^\kappa(x, \mathcal{N}, \phi, \tilde{\mu})| \leq \hat{K}$. □

We consider a grid reasonable if the minimum distance along grid lines from any node to a boundary node is bounded. Let $\underline{X} = (x_1, x_2, \ldots x_k)$ be a grid path or sequence of neighboring nodes, such that $x_l \in \mathcal{X}$ for $1 \leq l \leq k$ and $x_l \in \mathcal{N}(x_{l-1})$ for $2 \leq l \leq k$. Define the grid path length of $\underline{X}$ by

$$\omega(\underline{X}) = \sum_{l=2}^{k} \|x_l - x_{l-1}\|_2.$$

Define the minimum grid path length between $x \in \mathcal{X}$ and $x' \in \mathcal{X}$ to be

$$\breve{\omega}(x, x') = \min\{\omega(\underline{X}) \mid x_1 = x \text{ and } x_k = x'\}.$$

Finally, define the minimum node-to-boundary grid path length of $x \in \mathcal{X}$ as

$$\tilde{\omega}(x) = \min_{x' \in \underline{\partial\Omega}} \breve{\omega}(x, x').$$

Theorem 4.23. *Let $\mathcal{G}$ be an orthogonal discretization such that for all $x \in \underline{\Omega}$, $\tilde{\omega}(x) \leq \hat{W}$, for some constant $\hat{W}$. Let $\underline{u}^{\mathcal{G}}$ be the unique solution to (3.1). Then*

$$\min_{x' \in \underline{\partial\Omega}} g(x') \leq \underline{u}^{\mathcal{G}} \leq \hat{W}\hat{K} + \max_{x' \in \underline{\partial\Omega}} g(x'). \tag{4.14}$$

Proof. Let $x \in \underline{\Omega}$. Let $x' \in \underline{\partial\Omega}$ and $\underline{X} = (x_1, x_2, \dots x_k)$, such that $x_1 = x$, $x_k = x'$, and $\omega(\underline{X}) = \tilde{\omega}(x)$. Obtain a modified grid $\tilde{\mathcal{G}}$ for which $\underline{\tilde{\Omega}}$ includes all nodes in $\underline{\Omega}$ that are along the grid path $\underline{X}$:

$$\underline{\tilde{\Omega}} = \{x \mid x \in \underline{\Omega} \text{ and } x = x_j, \text{ for some } j \text{ such that } 1 \leq j \leq k\}$$

and $\underline{\partial\tilde{\Omega}}$ includes all other nodes in $\mathcal{X}$:

$$\underline{\partial\tilde{\Omega}} = \mathcal{X} \setminus \underline{\tilde{\Omega}}.$$

Define the boundary condition $\tilde{g} : \underline{\partial\tilde{\Omega}} \to \mathbb{R}$ for the modified discretization

$$\tilde{g}(x) = \begin{cases} g(x), & x \in \underline{\partial\Omega}, \\ \infty, & \text{otherwise.} \end{cases}$$

Let $\underline{u}^{\tilde{\mathcal{G}}}(x)$ be the unique solution to the modified problem at x. Note that $\underline{u}^{\tilde{\mathcal{G}}}(x) \leq \hat{W}\hat{K} + g(x')$, where $\hat{K}$ is given by Lemma 4.22. Also note, by the monotonicity of $\underline{H}$ (Corollary 4.19), we have $\underline{u}^{\mathcal{G}}(x) \leq \underline{u}^{\tilde{\mathcal{G}}}(x)$. Therefore, $\underline{u}^{\mathcal{G}}(x) \leq \hat{W}\hat{K} + \max_{x' \in \underline{\partial\Omega}} g(x')$, for all $x \in \mathcal{X}$.

For the lower bound, by Theorem 4.15, the unique solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{N}(x), \underline{u}, \mu) = 0$ must be such that $\tilde{\mu} > \breve{\underline{u}}$, where $\breve{\underline{u}}$ is the minimum neighbor value as in (4.9). By induction on the nodes of $\underline{\Omega}$, we find that for all $x \in \mathcal{X}$,

$$\underline{u}^{\mathcal{G}}(x) \geq \min_{x' \in \underline{\partial\Omega}} g(x').$$

Because of the non-expansive interpolation used to determine $\underline{u}^{\mathcal{G}}$, (4.14)

holds. □

4.4 Efficient Implementation of Update

This section describes three methods for improving the efficiency of the `Update` function: symmetry, causality, and solution elimination. Some of these methods are related to those found in [40, 70]. However, the efficiency gains from using these methods are not substantial for an already efficient implementation of Algorithm 1. In such an implementation the `Update` function only computes updates from those nodes that are accepted (i.e. that have been removed from $\mathcal{H}$) as in (4.8). Because the nodes are accepted in nondecreasing value order, causality elimination can not possibly remove any nodes from consideration in `Update`. Furthermore, only simplices that include the node x most recently removed from $\mathcal{H}$ are considered in computing the updated value. Then, $\underline{v}(x)$ gets assigned the minimum of this newly computed value and its previous value. Our experiments show that in many calls to `Update`, only a single simplex fits these criteria, and the fraction of updates for which only a single simplex fits the criteria grows as the grid is refined. For this reason, the techniques for eliminating nodes and simplices described in this section are largely ineffective.

However, for coarse grid resolutions and problems where characteristics intersect often, multiple simplices are considered by `Update` frequently enough that symmetry elimination, which is very cheap, significantly improves efficiency. In some cases, a node value update can be ignored altogether if the most-recently extracted node is eliminated by symmetry.

Despite the fact that the node and simplex elimination techniques described here are useful only in limited circumstances, we include them for theoretical interest and because they may be applied in other algorithms, such as iterative methods, that also require the `Update` function and for which the number of potential simplices to consider may be larger.

Efficiency can be gained by determining which stencil nodes $y \in \mathcal{N}$ have no influence on the solution and eliminating them from consideration. Let $\mathcal{K}_{\mathcal{N}}$ be the set of stencil simplices that can be formed by the neighbors in

$\mathcal{N}$. For example, in $d = 4$ dimensions, take

$$\mathcal{N} = \{x_2^{\pm}, x_3^{-}, x_4^{\pm}\} \quad \text{and}$$

Then we have

$$\mathcal{K}_{\mathcal{N}} = \{(0,-1,-1,-1),(0,+1,-1,-1),(0,-1,-1,+1),(0,+1,-1,+1)\}.$$

For $\kappa \in \mathcal{K}_{\mathcal{N}}$ and $1 \leq j \leq d$, $\kappa_j = 0$ indicates that $x_j^{\kappa_j}$ is not considered in computing the gradient approximation $\underline{D}^{\kappa}(\mu)$; that is, $\underline{D}_j^{\kappa}(x, \mathcal{N}, \phi, \mu) = 0$ if $\kappa_j = 0$ and $\underline{D}_j^{\kappa}$ satisfies (4.7) otherwise. We recognize that the computation of $\underline{H}$ can be simplified to only consider stencil simplices formed from the stencil nodes in $\mathcal{N}$:

$$\underline{H}(x, \mathcal{N}, \phi, \mu) = \max_{\kappa \in \mathcal{K}_{\mathcal{N}}} [H(x, \underline{D}^{\kappa}(x, \mathcal{N}, \phi, \mu))]. \tag{4.15}$$

In order to implement a more efficient `Update` function, first we attempt to reduce the set of stencil nodes from $\mathcal{N}$ to $\tilde{\mathcal{N}} \subseteq \mathcal{N}$ and second we solve

$$\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu) = 0 \tag{4.16}$$

for $\mu = \tilde{\mu}$.

4.4.1 Symmetry Node Elimination

We show how the considered stencil nodes $\mathcal{N}$ can be reduced by keeping only the neighbor with smaller value of a pair of opposite neighbors in the j^{th} dimension when (4.6) is symmetric in that dimension. This procedure is a generalization of those in [40, 70] to all axis-aligned anisotropic problems on unequally spaced grids. First, we introduce useful notation.

Let $q \in \mathbb{R}^d$. Let $T^i(q) \in \mathbb{R}^d$ be a reflection of q in the hyperplane orthogonal to the i^{th} axis, such that

$$T_j^i(q) = \begin{cases} -q_j, & \text{if } j = i, \\ q_j, & \text{otherwise,} \end{cases}$$

for $1 \leq j \leq d$. Let Φ_j indicate symmetry of (4.6) in the j^{th} dimension, as follows:

$$\Phi_j = \begin{cases} 1, & \text{if } |h_j^-| = |h_j^+| \text{ and for all } q \in \mathbb{R}^d,\ H(x,q) = H(x, T^j(q)), \\ 0, & \text{otherwise.} \end{cases}$$

In other words, $\Phi_j = 1$ if and only if the stencil spacing and H are symmetric in the j^{th} dimension.

Lemma 4.24. *Let j be such that $1 \leq j \leq d$. Let $\kappa \in \mathcal{K}_{\mathcal{N}}$ and $\kappa' = T^j(\kappa)$. If $\Phi_j = 1$ and $\phi(x_j^{\kappa'_j}) \leq \phi(x_j^{\kappa_j})$, then $H(x, \underline{D}^{\kappa'}(\mu)) \geq H(x, \underline{D}^{\kappa}(\mu))$, for all μ.*

Proof. Let j, κ, and κ' be as defined above. Let $\Phi_j = 1$ and $\phi(x_j^{\kappa'_j}) \leq \phi(x_j^{\kappa_j})$. Consider the components of $T^j(\underline{D}^{\kappa'}(\mu))$. We have

$$\begin{aligned} &T_j^j(\underline{D}^{\kappa'}(\mu))\underline{D}_j^{\kappa}(\mu) \\ &= -\underline{D}_j^{\kappa'}(\mu)\underline{D}_j^{\kappa}(\mu) \\ &= -\frac{\max(0, \mu - \phi(x_j^{\kappa'_j}))}{-h_j^{\kappa'_j}} \frac{\max(0, \mu - \phi(x_j^{\kappa_j}))}{-h_j^{\kappa_j}} \geq 0, \end{aligned}$$

since $h_j^{\kappa'_j} = -h_j^{\kappa_j}$. Furthermore,

$$\begin{aligned} |T_j^j(\underline{D}^{\kappa'}(\mu))| &= |-\underline{D}_j^{\kappa'}(\mu)| \\ &= \left| -\frac{\max(0, \mu - \phi(x_j^{\kappa'_j}))}{-h_j^{\kappa'_j}} \right| \\ &\geq \left| \frac{\max(0, \mu - \phi(x_j^{\kappa_j})}{-h_j^{\kappa_j}} \right| = |\underline{D}_j^{\kappa}(\mu)|, \end{aligned}$$

since $h_j^{\kappa'_j} = -h_j^{\kappa_j}$ and $\phi(x_j^{\kappa'_j}) \leq \phi(x_j^{\kappa_j})$. For $i \neq j$,

$$T_i^j(\underline{D}^{\kappa'}(\mu)) = \underline{D}_i^{\kappa'}(\mu) = \underline{D}_i^{\kappa}(\mu),$$

since $\kappa'_i = \kappa_i$. Consequently, $T^j(\underline{D}^{\kappa'}(\mu)) \trianglerighteq \underline{D}^{\kappa}(\mu)$.

Therefore, by the symmetry of H in the jth dimension and by the one-sided monotonicity of H (Property 4.7),

$$H(x, \underline{D}^{\kappa'}(\mu)) = H(x, T^j(\underline{D}^{\kappa'}(\mu))) \geq H(x, \underline{D}^{\kappa}(\mu)).$$

□

Theorem 4.25. *Let $\mathcal{N} \subseteq \{x_j^{\pm} \mid 1 \leq j \leq d\}$. Pick any $j \in \{1, 2, \ldots, d\}$ such that $\Phi_j = 1$, $x_j^- \in \mathcal{N}$, and $x_j^+ \in \mathcal{N}$. Pick any $\sigma \in \{-1, +1\}$ such that $\phi(x_j^{-\sigma}) \leq \phi(x_j^{\sigma})$. Let*

$$\tilde{\mathcal{N}} = \mathcal{N} \setminus \{x_j^{\sigma}\}.$$

Let $\mu = \mu_{\mathcal{N}}$ be the unique solution to $\underline{H}(x, \mathcal{N}, \phi, \mu) = 0$ and $\mu = \mu_{\tilde{\mathcal{N}}}$ be the unique solution to $\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu) = 0$. Then $\mu_{\tilde{\mathcal{N}}} = \mu_{\mathcal{N}}$.

Proof. Let $\kappa \in \mathcal{K}_{\mathcal{N}}$. First consider the case $\kappa_j = -\sigma$. We have $\kappa \in \mathcal{K}_{\tilde{\mathcal{N}}}$.

Second consider the case $\kappa_j = \sigma$. Let $\kappa' = T^j(\kappa)$ and note that $\kappa'_j = -\sigma$. Then, by Lemma 4.24, $H(x, \underline{D}^{\kappa'}(\mu)) \geq H(x, \underline{D}^{\kappa}(\mu))$, for all μ. In particular, $H(x, \underline{D}^{\kappa'}(\mu_{\mathcal{N}})) \geq H(x, \underline{D}^{\kappa}(\mu_{\mathcal{N}}))$.

In both cases, there exists $\tilde{\kappa} \in \mathcal{K}_{\tilde{\mathcal{N}}}$, such that $H(x, \underline{D}^{\tilde{\kappa}}(\mu_{\mathcal{N}})) \geq H(x, \underline{D}^{\kappa}(\mu_{\mathcal{N}}))$. Also, note that $\mathcal{K}_{\tilde{\mathcal{N}}} \subseteq \mathcal{K}_{\mathcal{N}}$. Consequently,

$$\begin{aligned}
\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu_{\mathcal{N}}) &= \max_{\kappa \in \mathcal{K}_{\tilde{\mathcal{N}}}} [H(x, \underline{D}^{\kappa}(\mu_{\mathcal{N}}))] \\
&= \max_{\kappa \in \mathcal{K}_{\mathcal{N}}} [H(x, \underline{D}^{\kappa}(\mu_{\mathcal{N}}))] \\
&= \underline{H}(x, \mathcal{N}, \phi, \mu_{\mathcal{N}}) = 0.
\end{aligned}$$

Therefore, $\mu = \mu_{\mathcal{N}}$ is the unique solution to $\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu) = 0$. □

An implementation of the `Update` function can use the result obtained in Theorem 4.25 to eliminate $x_j^{\pm} \in \mathcal{N}$ from consideration in solving (4.16) by exploiting symmetries in (4.6). We call this *symmetry elimination*.

Remark 4.26. *Theorem 4.25 can be generalized to a version that may eliminate the node with larger value in a pair of opposite nodes in the j^{th} dimension even if (4.6) is not symmetric in that dimension. We let $1 \leq j \leq d$*

such that $x_j^- \in \mathcal{N}$*, and* $x_j^+ \in \mathcal{N}$*. Let* $\sigma \in \{-1, +1\}$*. Node* $x_j^\sigma \in \mathcal{N}$ *may be eliminated from consideration if*

- $|h_j^{-\sigma}| \leq |h_j^\sigma|$*;*
- *for all* $q \in \mathbb{R}^d$ *such that* $\sigma q_j \geq 0$*,* $H(x, q) \geq H(x, T^j(q))$*;*
- *and* $\phi(x_j^{-\sigma}) \leq \phi(x_j^\sigma)$*.*

4.4.2 Causality Node Elimination

The causality of $\underline{H}$ can also be exploited to eliminate $x_j^\pm \in \mathcal{N}$ from consideration. This observation was used in two distinct but equivalent methods for analytically computing the `Update` from a single simplex to solve an isotropic Eikonal equation [40, 70]. We show with the following theorem that the condition $\underline{H}(\phi(y)) \geq 0$ can be checked to determine that a node $x_j^\pm$ is non-causal, i.e., that the solution $\mu = \tilde{\mu}$ to (4.16) is not dependent on the node y.

Theorem 4.27. *Let* $\mathcal{N} \subseteq \{x_j^\pm \mid 1 \leq j \leq d\}$*. Pick any* $\kappa \in \mathcal{K}_\mathcal{N}$ *and* $j \in \{1, 2, \ldots, d\}$*, such that* $\kappa_j \neq 0$ *and* $\underline{H}(x, \mathcal{N}, \phi, \phi(x_j^{\kappa_j})) \geq 0$*. Let*

$$\tilde{\mathcal{N}} = \mathcal{N} \setminus \{x_j^{\kappa_j}\}.$$

Let $\mu = \mu_\mathcal{N}$ *be the unique solution to* $\underline{H}(x, \mathcal{N}, \phi, \mu) = 0$ *and* $\mu = \mu_{\tilde{\mathcal{N}}}$ *be the unique solution to* $\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu) = 0$*. Then* $\mu_{\tilde{\mathcal{N}}} = \mu_\mathcal{N}$*.*

Proof. Let $\mathcal{N}$, κ, j, $\tilde{\mathcal{N}}$, $\mu_\mathcal{N}$ and $\mu_{\tilde{\mathcal{N}}}$ be as defined above. By Lemma 4.14(c), $\underline{H}(x, \mathcal{N}, \phi, \mu)$ is nondecreasing. Since

$$\underline{H}(x, \mathcal{N}, \phi, \phi(x_j^{\kappa_j})) \geq 0 = \underline{H}(x, \mathcal{N}, \phi, \mu_\mathcal{N}),$$

it must be that $\mu_\mathcal{N} \leq \phi(x_j^{\kappa_j})$. Note that $\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu)$ is identical to $\underline{H}(x, \mathcal{N}, \phi, \mu)$ except that $\underline{D}_j^\kappa(x, \mathcal{N}, \phi, \mu)$ is set to zero in $\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu)$. But for $\mu \leq \phi(x_j^{\kappa_j})$, we also have $\underline{D}_j^\kappa(x, \mathcal{N}, \phi, \mu) = 0$ in $\underline{H}(x, \mathcal{N}, \phi, \mu)$. Consequently, $\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu) = \underline{H}(x, \mathcal{N}, \phi, \mu)$ for $\mu \leq \phi(x_j^{\kappa_j})$. In particular,

$$\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu_\mathcal{N}) = \underline{H}(x, \mathcal{N}, \phi, \mu_\mathcal{N}) = 0.$$

Therefore, $\mu = \mu_{\mathcal{N}}$ is the unique solution to $\underline{H}(x, \tilde{\mathcal{N}}, \phi, \mu) = 0$. □

Theorem 4.27 states that the unique solution $\mu = \tilde{\mu}$ to (4.16) does not change when a non-causal node is removed from $\mathcal{N}$. This node removal can be repeated until all non-causal nodes have been removed and the solution $\tilde{\mu}$ will remain unchanged. We call this *causality elimination*. A binary or linear search through sorted neighbors' values can be used to determine the largest node value that might be causal. Note that causality elimination does not require symmetry in (4.16). However, the test for non-causality requires an evaluation of $\underline{H}$, which is more expensive than the comparison of two neighbors' values used for symmetry elimination.

4.4.3 Solution Elimination

After eliminating from consideration nodes in $\mathcal{N}$ using symmetry and causality elimination, we determine the solution $\mu = \tilde{\mu}$ to (4.16). Let

$$\check{\mu} = \min_{\kappa \in \mathcal{K}_{\mathcal{N}}} (\mu_\kappa), \tag{4.17}$$

where $\mu = \mu_\kappa$ is the unique solution to

$$H(x, \underline{D}^\kappa(\mu)) = 0. \tag{4.18}$$

We show with the following theorem that, instead of solving (4.16) directly, we can solve (4.18) for each $\kappa \in \mathcal{K}_{\mathcal{N}}$ and take the minimum such solution $\check{\mu}$. It can be shown that $H(x, \underline{D}^\kappa(\mu))$ is continuous and nondecreasing on μ and that (4.18) has a unique solution in an analogous but simpler manner to the proof of Theorem 4.15.

Theorem 4.28. *Let $\mu = \hat{\mu}$ be the unique solution to (4.16). Then $\hat{\mu} = \check{\mu}$.*

Proof. Let μ_κ, $\check{\mu}$ and $\hat{\mu}$ be as defined above. For any $\kappa \in \mathcal{K}_{\mathcal{N}}$, we know $\mu_\kappa \geq \check{\mu}$. Since $H(x, \underline{D}^\kappa(\mu))$ is nondecreasing on μ, it must be that $H(x, \underline{D}^\kappa(\mu)) \leq H(x, \underline{D}^\kappa(\mu_\kappa)) = 0$, for all $\mu \leq \mu_\kappa$. In particular, $H(x, \underline{D}^\kappa(\check{\mu})) \leq 0$. Furthermore, by the definition of $\check{\mu}$, there exists an $\check{\kappa} \in \mathcal{K}_{\mathcal{N}}$ such that $H(x, \underline{D}^{\check{\kappa}}(\check{\mu})) =$

0. Consequently,

$$\underline{H}(\check{\mu}) = \max_{\kappa \in \mathcal{K}_{\mathcal{N}}} H(x, \underline{D}^{\kappa}(\check{\mu})) = 0. \tag{4.19}$$

Therefore, $\hat{\mu} = \check{\mu}$ solves (4.16) and it is a unique solution by Theorem 4.15. □

We further show that we may be able to determine $\check{\mu}$ without solving (4.18) for each $\kappa \in \mathcal{K}_{\mathcal{N}}$. We demonstrate using the following theorem that if we have computed a solution $\mu = \mu_{\kappa}$ of (4.18) for some $\kappa \in \mathcal{K}_{\mathcal{N}}$, we can easily determine if $\mu_{\tilde{\kappa}} \geq \mu_{\kappa}$, where $\mu = \mu_{\tilde{\kappa}}$ is the solution to $H(x, \underline{D}^{\tilde{\kappa}}(\mu)) = 0$ for some $\tilde{\kappa} \in \mathcal{K}_{\mathcal{N}}$ such that $\tilde{\kappa} \neq \kappa$. Note we do not necessarily need to compute $\mu_{\tilde{\kappa}}$ to rule it out as a minimal solution.

Theorem 4.29. *Let $\kappa \in \mathcal{K}_{\mathcal{N}}$ and $\tilde{\kappa} \in \mathcal{K}_{\mathcal{N}}$. Let $\mu = \mu_{\kappa}$ be the unique solution to $H(x, \underline{D}^{\kappa}(\mu)) = 0$ and $\mu = \mu_{\tilde{\kappa}}$ be the unique solution to $H(x, \underline{D}^{\tilde{\kappa}}(\mu)) = 0$. Then $\mu_{\tilde{\kappa}} < \mu_{\kappa}$ if and only if $H(x, \underline{D}^{\tilde{\kappa}}(\mu_{\kappa})) > H(x, \underline{D}^{\kappa}(\mu_{\kappa}))$.*

Proof. Let μ_{κ} and $\mu_{\tilde{\kappa}}$ be as defined above. If $H(x, \underline{D}^{\tilde{\kappa}}(\mu_{\kappa})) > H(x, \underline{D}^{\kappa}(\mu_{\kappa})) = 0$, then the unique solution $\mu = \mu_{\tilde{\kappa}}$ to $H(x, \underline{D}^{\tilde{\kappa}}(\mu)) = 0$ must be such that $\mu_{\tilde{\kappa}} < \mu_{\kappa}$, since $H(x, \underline{D}^{\tilde{\kappa}}(\mu))$ is nondecreasing on μ. Similarily, if $H(x, \underline{D}^{\tilde{\kappa}}(\mu_{\kappa})) \leq H(x, \underline{D}^{\kappa}(\mu_{\kappa}))$ then the unique solution $\mu = \mu_{\tilde{\kappa}}$ to $H(x, \underline{D}^{\tilde{\kappa}}(\mu)) = 0$ must be such that $\mu_{\tilde{\kappa}} \geq \mu_{\kappa}$. □

The result of Theorem 4.29 can be used to eliminate simplices $\kappa \in \mathcal{K}_{\mathcal{N}}$ for which solutions to (4.18) are irrelevant to the computation. We call this process *solution elimination*.

4.5 Analytic Solutions

We provide analytic node value update equations for the cases where

$$H(q) = \|q\|_p - c \tag{4.20}$$

and $p = 1$, $p = 2$, or $p = \infty$. In these cases, there is an exact solution to (4.8). The equation for $p = 2$ fixes some errors in the appendix of [44].

In [1] we demonstrated that these cases could be handled by a Dijkstra-like method on an orthogonal grid and are useful for robotic applications. However, here we generalize the update equations to any dimension and grid spacing.

Let $\kappa \in \mathcal{K}_{\mathcal{N}}$. Let $\tilde{d}$ be the number of non-zero values in κ or the number of nodes in the simplex represented by κ. Note that κ might represent a lower-dimensional simplex: $1 \leq \tilde{d} \leq d$. Let $(v_1, v_2, \ldots, v_{\tilde{d}})$ be the list of simplex node values, one for each node x in the simplex and $(h_1, h_2, \ldots, h_{\tilde{d}})$ be the corresponding grid spacings. We are solving for μ. In order to use the analytic updates below, non-causal node values must already have been eliminated using causality elimination, so $\mu > \max_{1 \leq j \leq \tilde{d}} v_j$. In the case of Algorithm 1, any nodes that would be removed from consideration by causality elimination would not yet have been extracted from $\mathcal{H}$, and so the analytic updates below can be applied directly. In the following, indices j, l, j_1, and j_2 range from 1 to $\tilde{d}$ unless otherwise specified.

4.5.1 Update for $p = 1$

From (4.18) and (4.20) we have

$$\sum_j \left(\frac{|\mu - v_j|}{h_j} \right) = c.$$

Assume $\mu > \max_j v_j$ and multiply through by $\prod_l h_l$ to obtain

$$\sum_j \left(\prod_{l \neq j} h_l \right) \mu - \sum_j \left(\prod_{l \neq j} h_l \right) v_j = \left(\prod_l h_l \right) c.$$

Then solve for μ to get

$$\mu = \frac{\sum_j \left(\prod_{l \neq j} h_l \right) v_j + \prod_l h_l c}{\sum_j \prod_{l \neq j} h_l}.$$

4.5.2 Update for $p = 2$

From (4.18) and (4.20) we have

$$\sum_j \left(\frac{\mu - v_j}{h_j} \right)^2 = c^2.$$

Multiply through by $\prod_l h_l^2$ to get

$$\left[\sum_j \prod_{l \neq j} h_l^2 \right] \mu^2 - 2 \left[\sum_j \left(\prod_{l \neq j} h_l^2 \right) v_j \right] \mu + \sum_j \left(\prod_{l \neq j} h_l^2 \right) v_j^2 - \left(\prod_l h_l^2 \right) c^2 = 0.$$

Then, using the quadratic formula, solve for μ:

$$\begin{aligned} \mu &= \frac{2 \sum_j \left(\prod_{l \neq j} h_l^2 \right) v_j + \sqrt{\begin{array}{l} \left[2 \sum_j \left(\prod_{l \neq j} h_l^2 \right) v_j \right]^2 \\ - 4 \left[\sum_j \prod_{l \neq j} h_l^2 \right] \left[\sum_j \left(\prod_{l \neq j} h_l^2 \right) v_j^2 - \left(\prod_l h_l^2 \right) c^2 \right] \end{array}}}{2 \sum_j \prod_{l \neq j} h_l^2} \\ &= \frac{\sum_j \left(\prod_{l \neq j} h_l^2 \right) v_j + \prod_l h_l \sqrt{\begin{array}{l} \left(\sum_j \prod_{l \neq j} h_l^2 \right) c^2 \\ + \sum_{j_1} \sum_{j_2} \left(\prod_{l \neq j_1, j_2} h_l^2 \right) v_{j_1} v_{j_2} \\ - \sum_{j_1} \sum_{j_2} \left(\prod_{l \neq j_1, j_2} h_l^2 \right) v_{j_1}^2 \end{array}}}{\sum_j \prod_{l \neq j} h_l^2}. \end{aligned}$$

We only consider the larger of the two quadratic solutions since the alternative will result in $\mu \leq \max_j v_j$. The last two terms of the discriminant can be made more concise as follows:

$$
\begin{aligned}
&\sum_{j_1}\sum_{j_2}\left(\prod_{l\neq j_1,j_2} h_l^2\right) v_{j_1}v_{j_2} - \sum_{j_1}\sum_{j_2}\left(\prod_{l\neq j_1,j_2} h_l^2\right) v_{j_1}^2 \\
&= \sum_{j_1}\sum_{j_2\neq j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) v_{j_1}v_{j_2} + \sum_{j_1}\sum_{j_2=j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) v_{j_1}v_{j_2} \\
&\quad - \sum_{j_1}\sum_{j_2\neq j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) v_{j_1}^2 - \sum_{j_1}\sum_{j_2=j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) v_{j_1}^2 \\
&= 2\sum_{j_1}\sum_{j_2>j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) v_{j_1}v_{j_2} + \sum_{j}\left(\prod_{l\neq j} h_l^2\right) v_j^2 \\
&\quad - \sum_{j_1}\sum_{j_2>j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) (v_{j_1}^2 + v_{j_2}^2) - \sum_{j}\left(\prod_{l\neq j} h_l^2\right) v_j^2 \\
&= \sum_{j_1}\sum_{j_2>j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) 2v_{j_1}v_{j_2} - \sum_{j_1}\sum_{j_2>j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) (v_{j_1}^2 + v_{j_2}^2) \\
&= -\sum_{j_1}\sum_{j_2>j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right) (v_{j_1} - v_{j_2})^2.
\end{aligned}
$$

The simplified equation for μ becomes:

$$
\mu = \frac{\sum_j \left(\prod_{l\neq j} h_l^2\right) v_j + \prod_l h_l \sqrt{\begin{array}{l}\left(\sum_j \prod_{l\neq j} h_l^2\right) c^2 \\ \quad - \sum_{j_1}\sum_{j_2>j_1}\left(\prod_{l\neq j_1,j_2} h_l^2\right)(v_{j_1} - v_{j_2})^2\end{array}}}{\sum_j \prod_{l\neq j} h_l^2}.
$$

4.5.3 Update for $p = \infty$

From (4.18) and (4.20) we have

$$
\max_j \left(\frac{|\mu - v_j|}{h_j}\right) = c.
$$

Assume $\mu > \max_j v_j$ solve for μ to obtain

$$\mu = \min_j \left(v_j + h_j c \right).$$

The $p = \infty$ case is identical to the update formula for Dijkstra's algorithm for shortest path on a discrete graph.

4.6 Experiments

We conduct experiments to show numerical evidence that the result of Algorithm 1 (p.35) converges to the viscosity solution of (1.8), and to demonstrate types of anisotropic problems that can be solved. Throughout this section, the boundary conditions are $g(x) = 0$ for $x \in \partial\Omega$. For all experiments below, excluding that in Section 4.6.4, $\Omega = [-1, 1]^d \setminus \{O\}$ and $\partial\Omega = \{O\}$, where O is the origin. We discretize $[-1, 1]^d$ to have m uniformly-spaced nodes in each dimension, where m is odd to ensure that there is a node at the origin.

4.6.1 Numerical Convergence Study

We examine the difference between the solution to (3.1) and the solution to (1.8) for two simple Dirichlet problems. In particular, we look at how the absolute error changes as the grid spacing decreases toward zero. We take H to have the form in (4.2), where $G(Du(x)) = \|Du(x)\|_p$ and $p = 1$ or $p = 2$. There is no reason for testing $p = \infty$ because in this case Dijkstra's algorithm gives the exact solution, even on a coarse grid. We use the analytic node value update equations provided in Section 4.5. The approximation errors are summarized in Table 4.1.

4.6.2 Asymmetric Anisotropic Problem

For this anisotropic problem, H is as in (4.2), where G is defined by (4.5) (see Figure 4.3(b)). The four rectangular regions shown in black in Figure 4.5 are obstacles, where instead $H(x, q) = \|q\|_2 - c$ and $c \gg 1$, making it extremely costly for the system to traverse such states so that they will not

				$p=1$				$p=2$			
d	m	N	h	e_∞	r_∞^h	e_1	r_1^h	e_∞	r_∞^h	e_1	r_1^h
2	11	1.2e2	2.0e-1	2.2e-1		6.3e-2		1.2e-1		6.2e-2	
	21	4.4e2	1.0e-1	1.7e-1	.41	3.7e-2	.77	7.8e-2	.56	4.3e-2	.53
	41	1.7e3	5.0e-2	1.2e-1	.46	2.0e-2	.85	5.0e-2	.65	2.8e-2	.63
	81	6.6e3	2.5e-2	8.8e-2	.48	1.1e-2	.90	3.1e-2	.70	1.7e-2	.69
	161	2.6e4	1.3e-2	6.3e-2	.49	5.7e-3	.94	1.8e-2	.75	1.0e-2	.73
	321	1.0e5	6.3e-3	4.4e-2	.49	2.9e-3	.96	1.1e-2	.78	6.1e-3	.77
	641	4.1e5	3.1e-3	3.1e-2	.50	1.5e-3	.97	6.1e-3	.81	3.5e-3	.79
	1281	1.6e6	1.6e-3	2.2e-2	.50	7.6e-4	.98	3.4e-3	.83	2.0e-3	.82
3	11	1.3e3	2.0e-1	3.5e-1		1.2e-1		2.1e-1		1.2e-1	
	21	9.3e3	1.0e-1	2.6e-1	.43	6.9e-2	.78	1.4e-1	.58	8.4e-2	.57
	41	6.9e4	5.0e-2	1.9e-1	.47	3.9e-2	.85	8.7e-2	.66	5.4e-2	.65
	81	5.3e5	2.5e-2	1.3e-1	.49	2.1e-2	.89	5.3e-2	.72	3.3e-2	.70
	161	4.2e6	1.3e-2	9.5e-2	.50	1.1e-2	.92	3.1e-2	.76	2.0e-2	.74
4	11	1.5e4	2.0e-1	4.4e-1		1.7e-1		2.9e-1		1.8e-1	
	21	1.9e5	1.0e-1	3.2e-1	.45	9.8e-2	.78	1.9e-1	.60	1.2e-1	.58
	41	2.8e6	5.0e-2	2.3e-1	.48	5.5e-2	.83	1.2e-1	.67	7.7e-2	.66

Table 4.1: Errors of approximate solution computed by Algorithm 1 compared to exact solution of (1.8), where H is as in (4.2) and $G(Du(x)) = \|Du(x)\|_p$. The variables d, m, and N are the dimension, the number of nodes in each dimension, and the total number of nodes, respectively. Other variables are the spacing h between grid nodes, the $\mathcal{L}_\infty$-error e_∞, the $\mathcal{L}_\infty$ convergence rate r_∞^h, the $\mathcal{L}_1$-error e_1, and the $\mathcal{L}_1$ convergence rate r_1^h.

be on any optimal path from a non-obstacle state to the goal. In the `Update` function, we analytically computed to solution to (4.8) using the equations for updating from a single simplex given in Section 4.5. The number of nodes in each dimension is $m = 1281$. We plot the contours of $\underline{u}$ computed by Algorithm 1 in Figure 4.5. Note the asymmetric contours where the characteristics bend through gaps. The relationship between the shape of the contours of G in Figure 4.3(b) and those of $\underline{u}$ is explained by the duality articulated in [3, Proposition 2.7 and Appendix A].

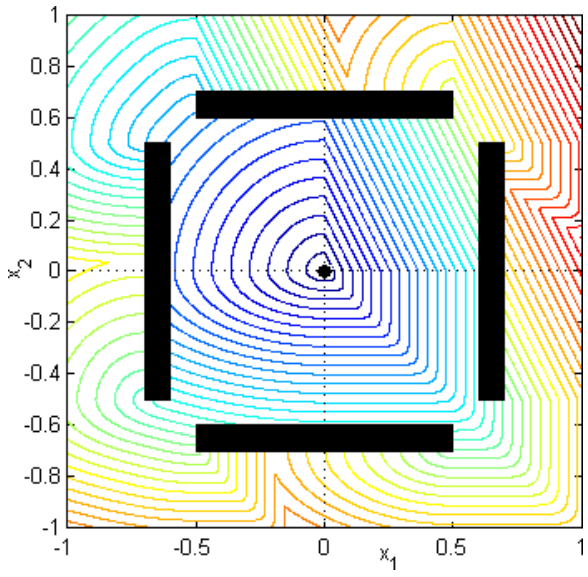


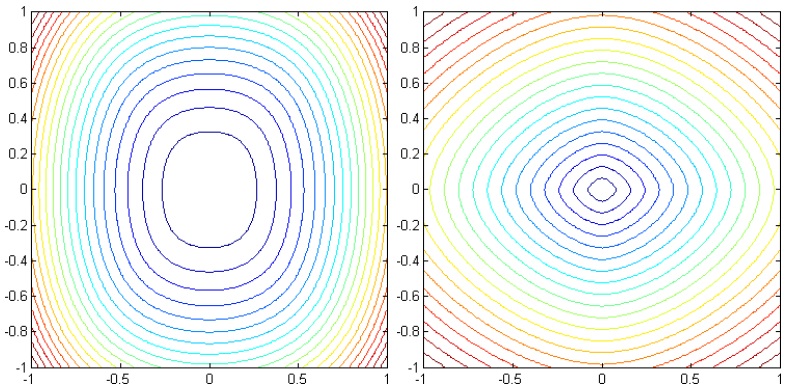
Figure 4.5: Contours of $\underline{u}$ computed for the anisotropic problem where Hamiltonian H is as in (4.2) and G is as in (4.5). The black circle at $O = (0, 0)$ indicates $\underline{\partial\Omega}$ and the black rectangles are obstacles. In these regions, $\underline{u}$ has purposefully not been computed.

4.6.3 Anelliptic Elastic Wave Propagation

We consider elastic wave propagation in transversely isotropic media with a vertical axis of symmetry (VTI media) as is done in [28]. In particular, we wish to find the arrival times of qP-waves propagating in two dimensions from a point source at the origin O. We solve the anisotropic HJ PDE given by defining the Hamiltonian

$$H(q) = \frac{1}{2}(q_1^2 + q_2^2)\left\{(a+l)q_1^2 + (c+l)q_2^2 + \sqrt{[(a-l)q_1^2 - (c-l)q_2^2]^2 + 4(f+l)^2 q_1^2 q_2^2}\right\} - 1. \tag{4.21}$$

This Hamiltonian is derived from the anisotropic Eikonal equation and the exact qP-wave phase velocity equation in [28]. The parameters $a = 14.47$,



(a) (b)

Figure 4.6: Using Algorithm 1 for computation of travel times of qP-waves in two-dimensional VTI media. (a) Contours of Hamiltonian H as given in (4.21). (b) Contours of approximate solution $\underline{u}$ computed by Algorithm 1.

$l = 2.28$, $c = 9.57$, and $f = 4.51$ are taken from [28].

The Hamiltonian H and the approximate solution $\underline{u}$ resulting from Algorithm 1 are shown in Figure 4.6. We have not shown analytically that (4.21) satisfies strict one-sided monotonicity for some range of parameters. However, the level sets of H as shown in Figure 4.6(a) indicate that H is strictly one-sided monotone for the given parameters. Furthermore, the level sets of H indicate that H is convex and computation of the derivative of H using the symbolic mathematics program Maple shows that H satisfies Osher's criterion for the given parameters. As a result, the analysis in this paper can be applied to the problem and Algorithm 1 can be used to compute the solution. Theorem 4.15 tells us that we can determine the solution to (4.8) uniquely. In the `Update` function, we used the bisection root-finding method to solve (4.8) numerically.

We used a grid of size 201×201. We computed the maximum relative error of $\underline{u}$ to be 0.0076, when compared to the travel-time computed with the group-velocity approximation for qP-waves presented in [28]. In turn, the group-velocity approximation is claimed to have a maximum error of

about 0.003, when compared to the true solution.

4.6.4 Two Robots

We consider the two-robot coordinated navigation problem illustrated in Figure 1.1. The circular robots are free to move independently in a 2-dimensional plane but may not collide with each other or the obstacles (black region). The dark-colored robot, called α, has a maximum speed of f_α. The light-colored robot, called β, has a maximum speed of f_β. The robots attempt to achieve a joint goal state in minimal time from any initial state in the domain without incurring collisions.

Let the state of robot α be $(x_1^\alpha, x_2^\alpha) \in \mathbb{R}^2$ and the state of robot β be $(x_1^\beta, x_2^\beta) \in \mathbb{R}^2$ so that the combined state of the two robots is

$$x = (x_1^\alpha, x_2^\alpha, x_1^\beta, x_2^\beta) \in \mathbb{R}^4.$$

The corresponding direction of travel for the combined system is

$$a = (a_1^\alpha, a_2^\alpha, a_1^\beta, a_2^\beta) \in \mathbb{R}^4,$$

where $\|a\|_2 = 1$. The speed function in non-collision states is

$$f(x,a) = f(a) = \min\left(\frac{f_\alpha}{\|(a_1^\alpha, a_2^\alpha)\|_2}, \frac{f_\beta}{\|(a_1^\beta, a_2^\beta)\|_2}\right).$$

The resulting speed profile is

$$\mathcal{A}_f = \{af(a) \mid \|a\| \leq 1\} = \{af(a) \mid F(af(a)) \leq 1\},$$

where

$$F(y) = \max\left(\frac{\|(y_1^\alpha, y_2^\alpha)\|_2}{f_\alpha}, \frac{\|(y_1^\beta, y_2^\beta)\|_2}{f_\beta}\right)$$

Proposition 2.7 of [3] states that we can use the dual of F to obtain

$$H(x,q) = G(x,q) - 1 = \left\|\left(f_\alpha\|(q_1^\alpha, q_2^\alpha)\|_2, f_\beta\|(q_1^\beta, q_2^\beta)\|_2\right)\right\|_1 - 1, \qquad (4.22)$$

where

$$q = (q_1^\alpha, q_2^\alpha, q_1^\beta, q_2^\beta).$$

For collision states we set $f(x, a) = f(a) = \epsilon \ll \min(f_\alpha, f_\beta)$, making it extremely slow for the two-robot system to traverse such states, so that they will not be on any optimal path to the joint goal.

We can compute $\underline{u}$ using Algorithm 1 since G is a mixed p-norm, and thus H satisfies Assumptions 4.3 to 4.6 (see Section 4.2.2). The domain Ω is discretized using a uniform orthogonal grid of $(81 \times 21)^2$ nodes. The resulting discrete equation (4.8) is quartic in μ, so it is difficult to solve for μ analytically. We implement the `Update` function using the bisection method to find the unique solution to (4.8) numerically.

Once we have determined $\underline{u}$ using Algorithm 1, we approximately solve the following ODE to determine an optimal collision-free trajectory from any initial state to the goal:

$$\begin{aligned} \frac{dx}{dt} &= \operatorname*{argmax}_{a \in \mathcal{A}} [(-q \cdot a) f(x, a)] \\ &= \left(w_\alpha q_1^\alpha, w_\alpha q_2^\alpha, w_\beta q_1^\beta, w_\beta q_2^\beta \right), \end{aligned} \tag{4.23}$$

where $q = D\underline{u}(x)$,

$$w_\alpha = \frac{f_\alpha}{\|(q_1^\alpha, q_2^\alpha)\|_2}, \quad \text{and} \quad w_\beta = \frac{f_\beta}{\|(q_1^\beta, q_2^\beta)\|_2}.$$

We discretize the ODE (4.23) using forward Euler. The gradient $D\underline{u}(x)$ is determined by a first-order finite difference scheme. At each time step, each robot moves at its maximum speed in the direction of the relevant components of the negative gradient. If the magnitude of the relevant components of the gradient fall below a small threshold the robot does not move at all, as is the case for the light-colored robot in Figures 1.1(d) and 1.1(e). In this case, the robot is not in the critical path to achieve the goal so it is not compelled to move. The optimal trajectories from a single starting condition are shown in Figure 1.1.

Chapter 5

OUM with Monotone Node Acceptance for Convex HJ PDEs

This material in this chapter is based on a submitted paper [5]. For the method described in this chapter, we assume H has the form (1.9) and $\mathcal{A}_f(x)$ is a closed convex set containing the origin in its interior.

5.1 Introduction

One potential benefit of single-pass methods is that it may be possible to estimate error and refine the grid to achieve a desired solution accuracy as the solution is computed outward from the boundary condition. Because a node value is not computed until those node values on which it depends have already been computed, there is the potential to control error early before it has been propagated unnecessarily to dependent grid nodes. We are not aware of any existing adaptive single-pass methods for static HJ PDEs that refine the grid to control error on the fly. However, we believe that such an adaptive method would create significantly nonuniform grids for many problems. The OUM in [59, 60] uses the global maximum grid spacing to determine the size of the update stencil of a node, which leads

to unreasonably large stencils in refined parts of the grid. Our attempts at modifying that OUM to use local grid spacing for the stencil showed some encouraging experimental evidence, but we have thus far been unable to prove theoretical convergence. Instead we developed an alternative method for which we can show convergence using a well-known consistency, monotonicity, and stability proof [8] (see Section 3.2). An added benefit of our method is that the discretization of the HJ PDE is δ-causal, such that the solution value at a grid node is dependent only on solution values that are smaller by at least δ, which is not true of the OUM in [59, 60].

We present Monotone Acceptance OUM (MAOUM), a two-pass Dijkstra-like method for which node values are accepted in nondecreasing order on a simplicial grid. We contrast MAOUM with the OUM introduced in [59, 60], which solves the same set of static convex HJ PDEs but does not necessarily accept node values monotonically. Because one of the defining features of that method is that a front of nodes with accepted values is maintained and the stencil is formed dynamically using only nodes from this front, we call their method Accepted Front OUM (AFOUM). MAOUM is a two-pass algorithm because the stencils for the discrete equation are precomputed in an initial pass through the nodes of the grid, as opposed to being computed in the same pass as the solution like in AFOUM. If $\delta > 0$, MAOUM can be modified to create a Dial-like method, for which nodes are sorted into buckets of width δ according to value and buckets of nodes are accepted in increasing value order [23, 64].

We present three hierarchical tests for the δ-causality of a node value update from a simplex and prove their validity in Section 5.3. We believe the first criterion, δ-Negative-Gradient-Acuteness, is nearly equivalent to the criterion for the applicability of Dijkstra-like and Dial-like methods to anisotropic problems given in [65, Section 4]. However, δ-Negative-Gradient-Acuteness was derived independently and does not require differentiability of the cost function (5.9). We use these tests to define MAOUM, verify that it solves the discretized equation, and analyze its asymptotic complexity in Section 5.4. Numerical examples are included in Section 5.5 to show that the algorithm is convergent, is not much less efficient/accurate than

AFOUM on uniform grids and is significantly more efficient/accurate than AFOUM for some examples with appropriately chosen grid refinement. We also demonstrate that MAOUM can be used to solve practical problems, such as computing the first-arrival time for a seismic wavefront or finding optimal paths for a robot to reach a goal while fighting a strong wind and avoiding obstacles.

5.1.1 Dijkstra-like Methods

We compute an approximate solution $\underline{u}^{\mathcal{G}}$ on a simplicial grid $\mathcal{G}$. Let s be a simplex with n_s vertices, and x_i^s be the i^{th} vertex of s where $1 \leq i \leq n_s$. Let $\underline{\mathcal{S}}$ be the set of grid simplices using neighboring grid nodes in $\mathcal{X}$. Define $\underline{\mathcal{S}}(\mathcal{R}) = \{s \in \underline{\mathcal{S}} \mid \text{ for } 1 \leq i \leq n_s,\, x_i^s \in \mathcal{R} \subset \mathbb{R}^d\}$, the set of grid simplices using neighboring grid nodes in $\mathcal{R}$. We may specify the number of vertices using a subscript: for example, $\underline{\mathcal{S}}_{\tilde{d}}(\mathcal{R})$ is the set of grid simplices with $\tilde{d}$ vertices. If left unspecified it is assumed $1 \leq n_s \leq d$, thereby excluding simplices with $d+1$ vertices.

MAOUM is an extension of Algorithm 1 (p.35) to handle HJ PDEs with convex anisotropy, while allowing for nonuniformity (defined using either a global or local mesh ratio) in the computational grid. For MAOUM, we use the semi-Lagrangian discretization from [60] to calculate $\texttt{Update}(y, s)$. This discretization generalizes the one for Eikonal equations from [63] to anisotropic HJ PDEs. It is discussed in detail in Section 5.2.

In order for MAOUM to maintain the capability of computing node values in a single pass, the update stencil for a node $x \in \mathcal{X}$ needs to be expanded beyond $\mathcal{N}(x)$ to handle many convex anisotropic problems. MAOUM includes an initial pass to compute this stencil for each x, resulting in a two-pass method. However, the asymptotic complexity for MAOUM is only increased from that of Algorithm 1 by a constant factor of $(\hat{\Psi}\rho)^d$, where d is the dimension, $\hat{\Psi}$ measures anisotropy in the equation, and ρ measures the degree of nonuniformity in the grid. This constant factor bounds above the number of nodes in a stencil. Although ρ factors into the asymptotic complexity, experiments demonstrate that grid nonuniformity does not appear

to have a large effect on the computational cost of MAOUM in practice.

5.1.2 Related Work

Like AFOUM, our method solves convex anisotropic problems on any simplicial grid. MAOUM does not maintain an accepted front but must perform an initial pass to compute the stencils for each node and store them for the second pass that computes the solution. The benefit of this extra initial computation and storage is that MAOUM does not need to determine the stencil based on a global grid spacing parameter, as is done with AFOUM. This results in an algorithm that can take better advantage of local refinements in the grid to perform updates from a smaller stencil.

Sweeping methods have been used to solve static convex Hamilton-Jacobi equations on semi-structured grids [14] and unstructured grids [11, 50]. These methods have the advantage that the update of node value depends only on immediate neighbors, so they naturally take advantage of local refinement in the grid to compute accurate updates. However, the discretizations used lack an easy way of determining dependencies in the solution, which makes it difficult to know if a part of the solution has finished being computed before the algorithm has terminated. On the other hand, for single-pass methods like OUMs it is clear when a particular part of the solution is complete and this might allow adaptive error estimation and grid refinement to be done as the solution is computed. For this reason, we develop an OUM that is suited to exploiting local refinement.

5.2 Discretization

MAOUM can be used on any type of simplicial grid, whether structured, semi-structured, or unstructured. We use the semi-Lagrangian discretization and notation from [60]. All norms are Euclidean unless otherwise stated. Let ζ be an n-vector barycentric coordinate such that

$$\sum_{i=1}^{n} \zeta_i = 1 \tag{5.1}$$

and $0 \leq \zeta_i$ for $1 \leq i \leq n$. Let Ξ_n be the set of all possible n-vector barycentric coordinates. For some simplex s, state $\tilde{x}_s \in s$ can be parameterized by $\zeta \in \Xi_{n_s}$, that is, $\tilde{x}_s(\zeta) = \sum_{i=1}^{n_s} \zeta_i x_i^s$. Let $\tilde{\Xi}_n$ be the set of n-vector barycentric coordinates without the constraint $0 \leq \zeta_i$ for $1 \leq i \leq d$. This looser definition of barycentric coordinates is used in Section 5.2.4.

Let $x \in \Omega$ and define

$$\tau_s(x, \zeta) = \|\tilde{x}_s(\zeta) - x\| \tag{5.2}$$

and

$$a_s(x, \zeta) = \frac{\tilde{x}_s(\zeta) - x}{\tau_s(x, \zeta)}. \tag{5.3}$$

Restrict $x \notin s$ for all x so that $\tau_s(x, \zeta) > 0$. We may write $\tau_s(\zeta) = \tau_s(x, \zeta)$ and $a_s(\zeta) = a_s(x, \zeta)$ when x is clear from context. We say that $a \in \mathcal{A}$ *intersects s from x* if the ray $x + ta$, with $t > 0$, intersects s. Note that $a_s(\zeta)$ intersects s from x if and only if $\zeta_i \geq 0$ for $1 \leq i \leq n_s$, a condition imposed on ζ above.

We define the numerical Hamiltonian $\underline{H}$ as follows:

$$\underline{H}(x, \mathcal{S}, \phi, \mu) = \max_{s \in \mathcal{S}} \max_{\zeta \in \Xi_{n_s}} \left\{ \frac{\mu - \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s)}{\tau_s(x, \zeta)} f(x, a_s(x, \zeta)) - 1 \right\}, \tag{5.4}$$

where $x \in \Omega$, $\mathcal{S}$ is a set of simplices s in $\overline{\Omega}$ such that $x \notin s$ and vectors $x_i^s - x$ are independent, $\phi : \overline{\Omega} \to \mathbb{R}$ is bounded, and $\mu \in \mathbb{R}$. Note that the second parameter of $\underline{H}$ is a set of stencil simplices instead of a set of stencil nodes as in previous chapters. This modification to the framework does not affect the relevance of the proofs of Propositions 3.7 (i.e., convergence) and 3.13 (i.e., solution of the discretized equations). When we are only varying μ, we write $\underline{H}(\mu) = \underline{H}(x, \mathcal{S}, \phi, \mu)$ for convenience.

Given a simplicial grid $\mathcal{G}$, the discretized Dirichlet problem is to find a function $\underline{u}^{\mathcal{G}} : \mathcal{X} \to \mathbb{R}$, such that

$$\underline{H}(x, \mathcal{S}(x), \underline{u}, \underline{u}(x)) = 0, \qquad x \in \underline{\Omega} \tag{5.5a}$$

$$\underline{u}(x) = g(x), \quad x \in \underline{\partial\Omega}, \tag{5.5b}$$

where $\mathcal{S}(x)$, *the update simplex set of* x, is chosen so that $\underline{H}$ is consistent and (5.5) is causal, allowing the solution to the discrete equation to be computed efficiently using a Dijkstra-like single-pass method. Section 5.3 discusses criteria on $\mathcal{S}(x)$ that guarantee causality. A significant part of Section 5.4 defines and analyzes the algorithm used to generate $\mathcal{S}(x)$.

Propositions 5.1 and 5.2 state that the numerical Hamiltonian $\underline{H}$ is monotone and consistent, which is important for the convergence of $\underline{u}$ to the solution u of the HJ PDE (1.8) as the grid spacing goes to zero [8] (see Section 3.2). A key property for monotonicity of $\underline{H}$ is that ζ_i are constrained to be nonnegative in (5.4). For consistency of $\underline{H}$, it it crucial that the simplices in $\mathcal{S}(x)$ encompass all possible directions $a \in \mathcal{A}$. Proposition 5.3 expresses μ in $\underline{H}(\mu) = 0$ explicitly, which is useful for implementing the `Update` function in Algorithm 2.

5.2.1 Monotonicity

$\underline{H}(x, \mathcal{S}, \phi, \mu)$ is monotone in the values $\phi(x_i^s)$, for $s \in \mathcal{S}$ and $1 \leq i \leq n_s$. Monotonicity of $\underline{H}$ requires that if none of $\phi(x_i^s)$ decrease, then $\underline{H}(x, \mathcal{S}, \phi, \mu)$ should not increase. Monotonicity of $\underline{H}$ comes naturally for the semi-Lagrangian form of (5.4). However, it is essential that $\zeta_i \geq 0$ for $1 \leq i \leq n_s$. A negative ζ_i can cause an increase in $\phi(x_i^s)$ to increase $\underline{H}(x, \mathcal{S}, \phi, \mu)$. This requirement is equivalent to making $a_s(\zeta)$ intersect s from x and makes the discrete equation $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ *upwinding* in the terminology of [60].

The following proposition is analogous to Property 3.3, but is stated for the formulation of $\underline{H}$ in (5.4) with a second parameter that is a set of simplices instead of a set of nodes.

Proposition 5.1. *Let* $x \in \underline{\Omega}$. *Let* $\check{\phi} : \overline{\Omega} \to \mathbb{R}$ *and* $\hat{\phi} : \overline{\Omega} \to \mathbb{R}$ *be functions such that* $\check{\phi}(x_i^s) \leq \hat{\phi}(x_i^s)$ *for all* $s \in \mathcal{S}$ *and* $1 \leq i \leq n_s$. *Then* $\underline{H}(x, \mathcal{S}, \check{\phi}, \mu) \geq \underline{H}(x, \mathcal{S}, \hat{\phi}, \mu)$.

Proof. Let $s \in \mathcal{S}$ and $1 \leq i \leq n_s$. Since $\check{\phi}(x_i^s) \leq \hat{\phi}(x_i^s)$, $\zeta_i \geq 0$ for $1 \leq i \leq n_s$,

and summation is monotone, we have

$$\mu - \sum_{i=1}^{n_s} \zeta_i \check{\phi}(x_i^s) \geq \mu - \sum_{i=1}^{n_s} \zeta_i \hat{\phi}(x_i^s).$$

Therefore, since f is positive, τ_s is positive, and max is monotone, we have that $\underline{H}(x, \mathcal{S}, \check{\phi}, \mu) \geq \underline{H}(x, \mathcal{S}, \hat{\phi}, \mu)$. □

5.2.2 Consistency

For the numerical Hamiltonian $\underline{H}$ to be consistent with the Hamiltonian H from (1.9), the set of simplices $\mathcal{S}$ must encompass all possible action directions in $\mathcal{A}$.

Definition. We say that $\mathcal{S}$ is *directionally-complete(DC) for* $x \in \Omega$ if for all $a \in \mathcal{A}$ there exists an $s \in \mathcal{S}$ such that a intersects s from x.

Proposition 5.2. *Let $x \in \Omega$, $y \in \Omega$, and $\phi \in C_b^\infty(\Omega)$. Let $\mathcal{S}(y)$ be DC for y and such that for $s \in \mathcal{S}(y)$, we have $y \notin s$. Define*

$$\hat{r}(y) = \max_{s \in \mathcal{S}(y), 1 \leq i \leq n_s} \|x_i^s - y\|.$$

Then

$$\lim_{\substack{y \to x,\, \xi \to 0 \\ \hat{r}(y) \to 0}} \underline{H}(y, \mathcal{S}(y), \phi + \xi, \phi(y) + \xi) = H(x, D\phi(x)). \tag{5.6}$$

Proof. By (5.4), (5.1), the smoothness of ϕ, the continuity of max and f,

(5.3), the DC for y of $\mathcal{S}(y)$, and (1.9)

$$
\begin{aligned}
&\lim_{\substack{y\to x,\ \xi\to 0\\ \hat{r}(y)\to 0}} \underline{H}(y,\mathcal{S}(y),\phi+\xi,\phi(y)+\xi)\\
&=\lim_{\substack{y\to x,\ \xi\to 0\\ \hat{r}(y)\to 0}} \max_{s\in\mathcal{S}(y)}\max_{\zeta\in\Xi_{n_s}}\left\{\frac{\phi(y)+\xi-\sum_{i=1}^{n_s}\zeta_i(\phi(x_i^s)+\xi)}{\tau_s(y,\zeta)}f(y,a_s(y,\zeta))-1\right\}\\
&=\lim_{y\to x,\ \hat{r}(y)\to 0}\max_{s\in\mathcal{S}(y)}\max_{\zeta\in\Xi_{n_s}}\left\{\frac{\phi(y)-\phi(\sum_{i=1}^{n_s}\zeta_i x_i^s)+\mathcal{O}\left(\hat{r}(y)^2\right)}{\tau_s(y,\zeta)}f(y,a_s(y,\zeta))-1\right\}\\
&=\max_{s\in\mathcal{S}(x)}\max_{\zeta\in\Xi_{n_s}}[(-D\phi(x)\cdot a_s(x,\zeta))f(x,a_s(x,\zeta))]-1\\
&=\max_{a\in\mathcal{A}}[(-D\phi(x)\cdot a)f(x,a)]-1\\
&=H(x,D\phi(x))
\end{aligned}
$$

□

In order to use Proposition 5.2 to prove that Property 3.2 holds, we must show that for all $x\in\Omega$, for sequences $\mathcal{G}_k$ and $x_k\in\underline{\Omega}_k$ such that $x_k\to x$ and $\hat{h}^{\mathcal{G}_k}\to 0$, $\mathcal{S}(x_k)$ is DC for x_k. We prove this property in Section 5.4.2.

5.2.3 Unique Solution

There exists a unique solution $\mu=\tilde{\mu}$ to the equation $\underline{H}(\mu)=0$. The value $\tilde{\mu}$ can be written explicitly in terms of the other quantities in $\underline{H}(\mu)$. This is a convenient form of the discretized equation for determining the solution $\underline{u}(y)$ at a node y in terms of the solution at its neighbors as is done in the `Update` function of Algorithm 2. It is the same form as the semi-Lagrangian update in [60]. The following proposition states that a property analogous to Property 3.8, but for the modified $\underline{H}$ in (5.4), is satisfied. In addition, it gives the explicit formula for the unique solution and states that an optimal $\hat{s}\in\mathcal{S}$ and $\hat{\zeta}\in\Xi_{n_s}$ in (5.4) remain optimal in the explicit formula.

Proposition 5.3. *The unique solution $\mu = \tilde{\mu}$ to $\underline{H}(\mu) = 0$ with $\underline{H}$ defined by (5.4) is given by*

$$\tilde{\mu} = \min_{s \in \mathcal{S}} \min_{\zeta \in \Xi_{n_s}} \left\{ \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} + \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s) \right\}. \tag{5.7}$$

Furthermore, if $\hat{s} \in \mathcal{S}$ and $\hat{\zeta} \in \Xi_{n_s}$ are maximizers in (5.4), then $\hat{s} \in \mathcal{S}$ and $\hat{\zeta} \in \Xi_{n_s}$ are minimizers in (5.7).

Proof. Let $s \in \mathcal{S}$ and $\zeta \in \Xi_{n_s}$. Define function $\underline{H}_\zeta^s(\mu) : \mathbb{R} \to \mathbb{R}$ to be

$$\underline{H}_\zeta^s(\mu) = \frac{\mu - \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s)}{\tau_s(\zeta)} f(x, a_s(\zeta)) - 1.$$

The function $\underline{H}_\zeta^s(\mu)$ is strictly increasing on μ, since it is linear with positive slope $f(x, a_s(\zeta))/\tau_s(\zeta)$. Furthermore, $\underline{H}_\zeta^s(\mu) = 0$ has a unique solution

$$\mu = \tilde{\mu}_\zeta^s = \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} + \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s).$$

Now define function $\lambda_\zeta^s(\mu) : \mathbb{R} \to \mathbb{R}$ to be

$$\lambda_\zeta^s(\mu) = \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} \underline{H}_\zeta^s(\mu) = \mu - \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s) - \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))}.$$

The function $\lambda_\zeta^s(\mu)$ is also strictly increasing on μ, since it is linear with a slope of 1. Note that $\mu = \tilde{\mu}_\zeta^s$ is also the unique solution to $\lambda_\zeta^s(\mu) = 0$. Because $\underline{H}_\zeta^s(\mu)$ and $\lambda_\zeta^s(\mu)$ are both increasing and $\tilde{\mu}_\zeta^s$ is the unique solution to both $\underline{H}_\zeta^s(\mu) = 0$ and $\lambda_\zeta^s(\mu) = 0$ for all $s \in \mathcal{S}$ and $\zeta \in \Xi_{n_s}$, the solution $\mu = \tilde{\mu}$ to

$$\underline{H}(\mu) = \max_{s \in \mathcal{S}} \max_{\zeta \in \Xi_{n_s}} \underline{H}_\zeta^s(\mu) = 0,$$

must also be the solution to

$$\max_{s\in\mathcal{S}} \max_{\zeta\in\Xi_{n_s}} \lambda_\zeta^s(\mu)$$
$$= \max_{s\in\mathcal{S}} \max_{\zeta\in\Xi_{n_s}} \left\{ \mu - \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s) - \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} \right\} = 0.$$

By rearranging the terms and negating both sides of this equation, we get the formula (5.7) for the unique solution $\mu = \tilde{\mu}$. A maximizer $\hat{s} \in \mathcal{S}$ and $\hat{\zeta} \in \Xi_{n_s}$ of $\max_{s\in\mathcal{S}} \max_{\zeta\in\Xi_{n_s}} \underline{H}_\zeta^s(\mu)$ is also a maximizer of $\max_{s\in\mathcal{S}} \max_{\zeta\in\Xi_{n_s}} \lambda_\zeta^s(\mu)$, which is a minimizer of the right-hand-side of (5.7). □

Remark 5.4. *This proposition still applies in the case where $\mathcal{S}$ contains just a single simplex and also in the case where Ξ_{n_s} is replaced by $\tilde{\Xi}_{n_s}$ (i.e., the constraint $\zeta_i \geq 0$ is removed).*

5.2.4 Equivalence of Discretizations

We examine the equivalence between the first-order semi-Lagrangian discretization (5.7) and an Eulerian discretization [59]. This equivalence is demonstrated in Proposition 5.7. The proof uses a simplified version of the equivalence, which is stated in Proposition 5.6. Proposition 5.7 is used in the proof of Theorem 5.8 that shows the discrete equation is causal as long as negative-gradient-acuteness is satisfied. It can also be used to extend the main results in this chapter to the equivalent Eulerian discretization. The analysis in this section is based on that in the appendix of [59].

Define the function $\eta_\phi^s : \Xi_{n_s} \to \mathbb{R}$, which is the objective function in (5.7):

$$\eta_\phi^s(x, \zeta) = \frac{\tau_s(x, \zeta)}{f(x, a_s(x, \zeta))} + \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s).$$

We typically deal with a fixed x and write $\eta_\phi^s(\zeta) = \eta_\phi^s(x, \zeta)$. Define the restriction of (5.7) to a single simplex s:

$$\tilde{\mu}_s = \min_{\zeta\in\Xi_{n_s}} \eta_\phi^s(\zeta). \tag{5.8}$$

The solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ is given by $\tilde{\mu} = \min_{s \in \mathcal{S}} \tilde{\mu}_s$.

We show that η_ϕ^s is convex in ζ. This lemma is used in the proof of Proposition 5.7. Also, convexity is useful for applying convex numerical optimization algorithms to solve (5.7).

Lemma 5.5. *The function η_ϕ^s is convex.*

Proof. Define a function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$:

$$c(x, y) = \begin{cases} 0, & \text{if } y = 0, \\ \frac{\|y\|}{f\left(x, \frac{y}{\|y\|}\right)}, & \text{otherwise.} \end{cases} \tag{5.9}$$

Note that $\mathcal{A}_f(x) = \{y \mid c(x, y) \leq 1\}$ and that c is homogeneous in the second parameter: $c(x, ty) = tc(x, y)$ for $t \geq 0$. Thus, by the convexity of the set $\mathcal{A}_f(x)$, $c(x, y)$ is convex in y. We have

$$\begin{aligned} \eta_\phi^s(\zeta) &= \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} + \textstyle\sum_{i=1}^d \zeta_i \phi(x_i^s) \\ &= c(x, \tau_s(\zeta) a_s(\zeta)) + \textstyle\sum_{i=1}^d \zeta_i \phi(x_i^s) \\ &= c(x, \tilde{x}_s(\zeta) - x) + \textstyle\sum_{i=1}^d \zeta_i \phi(x_i^s). \end{aligned} \tag{5.10}$$

Since $c(x, y)$ is convex in y, $\tilde{x}_s$ is linear in ζ, and $\sum_{i=1}^d \zeta_i \phi(x_i^s)$ is linear in ζ, η_ϕ^s is convex. □

Proposition 5.6. *Let $x \in \Omega$ and $s \in \mathcal{S}$ and $n_s = d$. Conditions (a) and (b) on $\tilde{\mu}_s \in \mathbb{R}$ and $\hat{\zeta} \in \Xi_{n_s}$ are equivalent:*

(a)

$$\begin{aligned} &\max_{\zeta \in \tilde{\Xi}_d} \left\{ \frac{\tilde{\mu}_s - \sum_{i=1}^d \zeta_i \phi(x_i^s)}{\tau_s(\zeta)} f(x, a_s(\zeta)) \right\} \\ &= \frac{\tilde{\mu}_s - \sum_{i=1}^d \hat{\zeta}_i \phi(x_i^s)}{\tau_s(\hat{\zeta})} f(x, a_s(\hat{\zeta})) = 1, \end{aligned} \tag{5.11}$$

(b) there exists $q \in \mathbb{R}^d$ *satisfying*

$$\begin{aligned} &\max_{a \in \mathcal{A}} (-q \cdot a) f(x, a) \\ &= (-q \cdot a_s(\hat{\zeta})) f(x, a_s(\hat{\zeta})) = 1 \end{aligned} \tag{5.12}$$

and

$$\phi(x_i^s) = \tilde{\mu}_s + (x_i^s - x) \cdot q, \tag{5.13}$$

for i *such that* $1 \leq i \leq n_s$.

Proof. Since the vectors $\{x_i^s - x\}$ are independent, we define $q \in \mathbb{R}^d$ by

$$q = \begin{bmatrix} x_1^s - x \\ x_2^s - x \\ \vdots \\ x_d^s - x \end{bmatrix}^{-1} \begin{bmatrix} \phi(x_1^s) - \tilde{\mu}_s \\ \phi(x_2^s) - \tilde{\mu}_s \\ \vdots \\ \phi(x_d^s) - \tilde{\mu}_s \end{bmatrix}.$$

The above equation implies (5.13) holds for $1 \leq i \leq d$.

We prove that (a) and (b) on $\tilde{\mu}_s \in \mathbb{R}$ and $\hat{\zeta} \in \Xi_{n_s}$ are equivalent, by proving that (5.11) is equivalent to (5.12). By (5.13), (5.1), and (5.3),

$$\begin{aligned} \textstyle\sum_{i=1}^d \zeta_i \phi(x_i^s) &= \textstyle\sum_{i=1}^d \zeta_i [\tilde{\mu}_s + (x_i^s - x) \cdot q] \\ &= \tilde{\mu}_s + \left(\textstyle\sum_{i=1}^d \zeta_i x_i^s - x \right) \cdot q \\ &= \tilde{\mu}_s + \tau_s(\zeta) a_s(\zeta) \cdot q, \end{aligned}$$

for all $\zeta \in \tilde{\Xi}_d$. This equation can be rearranged and multiplied on both sides by $f(x, a_s(\zeta))$ to obtain

$$\frac{\tilde{\mu}_s - \sum_{i=1}^d \zeta_i \phi(x_i^s)}{\tau_s(\zeta)} f(x, a_s(\zeta)) = (-q \cdot a_s(\zeta)) f(x, a_s(\zeta)),$$

for all $\zeta \in \tilde{\Xi}_d$. Thus, (5.11) is equivalent to

$$\begin{aligned} &\max_{\zeta \in \tilde{\Xi}_d} [(-q \cdot a_s(\zeta)) f(x, a_s(\zeta))] \\ &= (-q \cdot a_s(\hat{\zeta})) f(x, a_s(\hat{\zeta})) = 1 \end{aligned} \tag{5.14}$$

We now prove that (5.14) is equivalent to (5.12). We first show that (5.14) implies (5.12). Note that $\hat{\zeta} \in \Xi_d$ is strictly feasible in $\tilde{\Xi}_d$. In other words, $\hat{\zeta}$ is a (not necessarily strict) local maximum of $(-q \cdot a_s(\zeta)) f(x, a_s(\zeta))$. Since $a_s(\zeta)$ is a continuous mapping, $a_s(\hat{\zeta})$ is a local maximum of $(-q \cdot a) f(x, a)$. Because $\mathcal{A}_f(x)$ is convex, a local maximum $\hat{a}$ of $(-q \cdot a) f(x, a)$ is a global maximum over all $a \in \mathcal{A}$. Thus, we have (5.12). But (5.12) implies (5.14), since $\mathcal{A} \supseteq \{a_s(\zeta) \mid \zeta \in \tilde{\Xi}_d\}$ and $\hat{\zeta} \in \Xi_d \subset \tilde{\Xi}_d$. Therefore, (5.12) is equivalent to (5.14), which is equivalent to (5.11). □

Proposition 5.6 shows in a simplified form the equivalence between the semi-Lagrangian and Eulerian discretizations. Note that there are two ways to find the unique $\tilde{\mu}_s$. The first, the semi-Lagrangian discretization, is to solve (5.11), which can be done by solving the unconstrained convex optimization problem

$$\tilde{\mu}_s = \min_{\zeta \in \tilde{\Xi}_{n_s}} \eta_\phi^s(\zeta).$$

The second, the Eulerian discretization, is to find $\tilde{\mu}_s$ by solving the nonlinear system of equations (5.13) and (5.12) for q and $\tilde{\mu}_s$.

Unfortunately, there may not be a $\hat{\zeta} \in \Xi_d$ that satisfies (5.11), or equivalently, satisfies condition (b) of Proposition 5.6. Also, the simplex s may have $1 \leq n_s < d$. Proposition 5.7 considers these more general cases. In the statement of Proposition 5.7, note that the maximum in (5.15) is over the set Ξ_{n_s} of barycentric coordinates within the simplex in contrast to the maximum of (5.11), which is over the set $\tilde{\Xi}_d$.

Proposition 5.7. *Let $x \in \Omega$ and $s \in \mathcal{S}$. Conditions (a) and (b) on $\tilde{\mu}_s \in \mathbb{R}$ and $\hat{\zeta} \in \Xi_{n_s}$ are equivalent:*

(a)

$$\begin{aligned} &\max_{\zeta \in \Xi_{n_s}} \left\{ \frac{\tilde{\mu}_s - \sum_{i=1}^d \zeta_i \phi(x_i^s)}{\tau_s(\zeta)} f(x, a_s(\zeta)) \right\} \\ &= \frac{\tilde{\mu}_s - \sum_{i=1}^d \hat{\zeta}_i \phi(x_i^s)}{\tau_s(\hat{\zeta})} f(x, a_s(\hat{\zeta})) = 1, \end{aligned} \tag{5.15}$$

(b) there exists $q \in \mathbb{R}^d$ satisfying (5.12) and

$$\phi(x_i^s) \begin{cases} = \tilde{\mu}_s + (x_i^s - x) \cdot q, & \text{if } \hat{\zeta}_i > 0, \\ \geq \tilde{\mu}_s + (x_i^s - x) \cdot q, & \text{if } \hat{\zeta}_i = 0, \end{cases} \tag{5.16}$$

for i such that $1 \leq i \leq n_s$.

We can make the proof of Proposition 5.7 below fit the form of the proof for Proposition 5.6 by making two adjustments. The first adjustment is to add $d - n_s$ phantom nodes x_i^s to s such that vectors $x_i^s - x$ remain independent but set $\hat{\zeta}_i = 0$. The second adjustment is to replace ϕ with a function $\tilde{\phi}$ such that $\tilde{\phi}(x_i^s) = \phi(x_i^s)$ for i such that $\hat{\zeta}_i > 0$ and $\hat{\zeta} \in \operatorname{argmin}_{\zeta \in \tilde{\Xi}_d} \eta_{\tilde{\phi}}^s(\zeta)$, where $\hat{\zeta} \in \Xi_{n_s}$ is the optimizer in (5.8). We let $q \in \mathbb{R}^d$ satisfy (5.13) with $\tilde{\phi}$ in place of ϕ, and use Proposition 5.6 to complete the proof.

Proof. Without loss of generality, assume $n_s = d$. If $n_s < d$, without affecting the proof result, we add $d - n_s$ phantom nodes x_i^s to s such that the vectors $\{x_i^s - x\}$ remain linearly independent. For i such that x_i^s is a phantom node, we set $\hat{\zeta}_i = 0$ and $\phi(x_i^s) = \infty$. By (5.1), there exists i such that $1 \leq i \leq n_s$ and $\hat{\zeta}_i > 0$. Without loss of generality, we assume $\hat{\zeta}_i > 0$ for $i = 1$.

First, we prove condition (a) implies condition (b). Assume $\tilde{\mu}_s \in \mathbb{R}$ and $\hat{\zeta} \in \Xi_{n_s}$ satisfy condition (a). We seek a function $\tilde{\phi}$ such that

$$\tilde{\phi}(x_i^s) \begin{cases} = \phi(x_i^s), & \text{if } \hat{\zeta}_i > 0, \\ \leq \phi(x_i^s), & \text{if } \hat{\zeta}_i = 0, \end{cases} \tag{5.17}$$

and

$$\hat{\zeta} \in \operatorname*{argmin}_{\zeta \in \tilde{\Xi}_d} \eta_{\tilde{\phi}}^s(\zeta). \tag{5.18}$$

By (5.10), because $c(x, \tilde{x}_s(\zeta) - x)$ is convex in ζ and $\sum_{i=1}^d \zeta_i \tilde{\phi}(x_i^s)$ is linear in ζ, such a function $\tilde{\phi}$ exists.

For $2 \leq i \leq d$, define the vector $\nu^i = (\nu_1^i, \ldots, \nu_d^i)$, where $\nu_1^i = -1$, $\nu_i^i = 1$, and $\nu_j^i = 0$ for j such that $j \notin \{1, i\}$. Let $\partial \eta_{\tilde{\phi}}^s(\zeta) / \partial \nu^i$ be the

directional subdifferential set of η_ϕ^s at ζ in direction ν^i. In order for (5.18) to be satisfied, $\tilde{\phi}$ must be defined such that

$$0 \in \frac{\partial \eta_{\tilde{\phi}}^s(\hat{\zeta})}{\partial \nu^i}, \tag{5.19}$$

for $2 \leq i \leq d$.

By (5.15) and Proposition 5.3, we have $\hat{\zeta} \in \operatorname{argmin}_{\zeta \in \Xi_d} \eta_\phi^s(\zeta)$. By Lemma 5.5, η_ϕ^s is convex. If $\hat{\zeta}_i > 0$, then $\hat{\zeta}$ is strictly feasible in Ξ_d in both directions ν^i and $-\nu^i$. On the other hand, if $\hat{\zeta}_i = 0$, then $\hat{\zeta}$ is strictly feasible in Ξ_d in direction ν^i, but $\hat{\zeta}$ is constrained in direction $-\nu^i$. In either case, the convexity of η_ϕ^s, the optimality of $\hat{\zeta}$, and the strict feasibility in Ξ_d in direction ν^i imply that $\max(\partial \eta_\phi^s(\hat{\zeta})/\partial \nu^i) \geq 0$. If $\hat{\zeta}$ is strictly feasible in Ξ_d in direction $-\nu^i$, it must be that $\min(\partial \eta_\phi^s(\hat{\zeta})/\partial \nu^i) \leq 0$. However, if $\hat{\zeta}$ is constrained in direction $-\nu^i$, it is possible that $\min(\partial \eta_\phi^s(\hat{\zeta})/\partial \nu^i) > 0$. In this case, we can define $\tilde{\phi}(x_i^s) < \phi(x_i^s)$, so that $\min(\partial \eta_{\tilde{\phi}}^s(\hat{\zeta})/\partial \nu^i) = 0$. With these observations and the definition of η_ϕ^s in mind, we define the function $\tilde{\phi} : \{x_i^s \mid 1 \leq i \leq d\} \to \mathbb{R}$:

$$\tilde{\phi}(x_i^s) = \begin{cases} \phi(x_i^s), & \text{if } \min\left(\frac{\partial \eta_\phi^s(\hat{\zeta})}{\partial \nu^i}\right) \leq 0, \\ \phi(x_1^s) - \|x_i^s - x_1^s\| \min\left(\frac{\partial \eta_\phi^s(\hat{\zeta})}{\partial \nu^i}\right), & \text{otherwise.} \end{cases} \tag{5.20}$$

By this definition, we have (5.19) for $2 \leq i \leq d$, which implies (5.18). Furthermore, (5.17) is satisfied by the definition of $\tilde{\phi}$. By (5.18), (5.17), (5.15), and Proposition 5.3 we have

$$\min_{\zeta \in \tilde{\Xi}_d} \eta_{\tilde{\phi}}^s(\zeta) = \eta_{\tilde{\phi}}^s(\hat{\zeta}) = \eta_\phi^s(\hat{\zeta}) = \min_{\zeta \in \Xi_d} \eta_\phi^s(\zeta) = \tilde{\mu}_s$$

Again by Proposition 5.3, we have

$$\max_{\zeta \in \tilde{\Xi}_d} \left\{ \frac{\tilde{\mu}_s - \sum_{i=1}^d \zeta_i \tilde{\phi}(x_i^s)}{\tau_s(\zeta)} f(x, a_s(\zeta)) \right\}$$
$$= \frac{\tilde{\mu}_s - \sum_{i=1}^d \hat{\zeta}_i \tilde{\phi}(x_i^s)}{\tau_s(\hat{\zeta})} = 1,$$

Since the vectors $\{x_i^s - x\}$ are independent, we define q satisfying

$$\tilde{\phi}(x_i^s) = \tilde{\mu}_s + (x_i^s - x) \cdot q,$$

for i such that $1 \leq i \leq n_s$. By (5.17), we have (5.16). Moreover, by Proposition 5.6, we have (5.12). Therefore, condition (b) holds.

Second, we prove condition (b) implies condition (a). Assume $\tilde{\mu}_s \in \mathbb{R}$ and $\hat{\zeta} \in \Xi_{n_s}$ satisfy condition (b). Let $q \in \mathbb{R}^d$ satisfy (5.12) and (5.16). Define $\tilde{\phi} : \{x_i^s \mid 1 \leq i \leq d\} \to \mathbb{R}$:

$$\tilde{\phi}(x_i^s) = \tilde{\mu}_s + (x_i^s - x) \cdot q.$$

By (5.16), we have (5.17). By (5.12) and Proposition 5.6, we have

$$\max_{\zeta \in \tilde{\Xi}_d} \left\{ \frac{\tilde{\mu}_s - \sum_{i=1}^d \zeta_i \tilde{\phi}(x_i^s)}{\tau_s(\zeta)} f(x, a_s(\zeta)) \right\}$$
$$= \frac{\tilde{\mu}_s - \sum_{i=1}^d \hat{\zeta}_i \tilde{\phi}(x_i^s)}{\tau_s(\hat{\zeta})} f(x, a_s(\hat{\zeta})) = 1.$$

Therefore, by (5.17) and the fact that $\hat{\zeta} \in \Xi_{n_s}$, we have (5.15). □

5.3 Causality

Causality means that if the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ depends on value $\phi(x_i^s)$ then it must be that $\tilde{\mu} > \phi(x_i^s)$. Causality allows Dijkstra-like algorithms to be used to compute the solution to (5.5) in a single pass through the nodes $x \in \mathcal{X}$ in order of increasing value $\underline{u}(x)$. For isotropic

problems, causality is satisfied if the direction vectors $(x_i^s - x)/\|x_i^s - x\|$ for $1 \leq i \leq n_s$, when considered pairwise, form non-obtuse angles [58]. Negative-gradient-acuteness (NGA) is a similar property for more general anisotropic problems.

Let $\delta \geq 0$. δ-causality means that if the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ depends on value $\phi(x_i^s)$ then it must be that $\tilde{\mu} > \phi(x_i^s) + \delta$. δ-causality allows Dial-like algorithms with buckets of width δ to be used to compute the solution to (5.5) in a single pass through the nodes $x \in \mathcal{X}$ in order of increasing bucket value.

Definition. Let $\check{\zeta}^s$ be a minimizer in (5.8). We say that (5.8) is *δ-causal for $x \in \Omega$ and $s \in \mathcal{S}$* if $\check{\zeta}_i^s > 0$ implies $\tilde{\mu}_s - \phi(x_i^s) > \delta$, for $1 \leq i \leq n_s$. When $\delta = 0$, we may say (5.8) is *causal for x* and s.

In Section 5.3.1 we define what it means for a simplex to be δ-NGA and prove that it implies δ-causality of the discrete equation. In [65] a criterion for the δ-causality of (3.1) was presented. We believe δ-NGA is equivalent to the criterion in [65], if the function c defined in (5.9) is differentiable in the second parameter y everywhere but at the origin. In Section 5.3.2 we define another property on a simplex, δ-anisotropy-angle-boundedness (δ-AAB), and show that it implies δ-NGA. δ-AAB is more specific than δ-NGA, but easy to implement in Algorithm 2. δ-NGA is general and more likely to have application beyond Algorithm 2. In Section 5.3.3 we define one last property on a simplex called distance-ratio-boundedness (DRB). When $\delta = 0$, DRB implies δ-AAB. DRB is the most specific condition and we use it to prove the correctness of Algorithm 3 in Theorem 5.17.

The properties δ-NGA and δ-AAB allow for general $\delta \geq 0$ so that Algorithm 2 can easily be converted into a Dial-like method. However, we set $\delta = 0$ for the description and discussion of Dijkstra-like Algorithm 2 in Section 5.4 and we test only the Dijkstra-like algorithm in Section 5.5. The property DRB is not generalized to $\delta \geq 0$, because it is used to prove the correctness of just the Dijkstra-like algorithm.

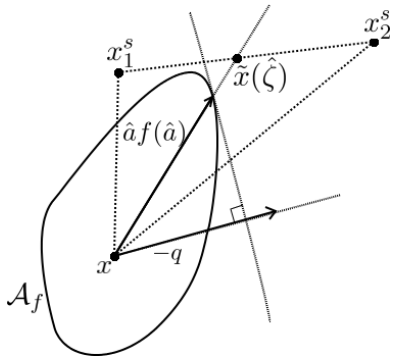


Figure 5.1: Depiction of symbols used in the definition of δ-NGA and the proof of Theorem 5.8.

5.3.1 Negative-gradient-acuteness

We show that if $\mathcal{S}$ is δ-negative-gradient-acute, the discrete equation $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ is δ-causal. Figure 5.1 is a geometric aid in understanding the definition of and proofs involving δ-negative-gradient-acuteness.

Definition. We say that $s \in \mathcal{S}$ is *δ-negative-gradient-acute (δ-NGA) for $x \in \Omega$* if: for all $q \in \mathbb{R}^d$ and $\hat{\zeta} \in \Xi_{n_s}$ such that (5.12) holds, we have $(x_i^s - x) \cdot (-q) > \delta$ for i such that $\hat{\zeta}_i > 0$. We say that $\mathcal{S}$ is *δ-NGA for x* if all $s \in \mathcal{S}$ are δ-NGA for x. When $\delta = 0$, we may say s (or $\mathcal{S}$) is *NGA for x*.

Theorem 5.8. *Let $s \in \mathcal{S}$ be δ-NGA for $x \in \Omega$. Let (5.8) hold and $\check{\zeta}$ be the minimizer in (5.8). Then (5.8) is δ-causal for x and s.*

Proof. Let $\hat{\zeta} = \check{\zeta}$. By Proposition 5.3, (5.15) holds. By Proposition 5.7, there exists $q \in \mathbb{R}^d$ such that (5.12) and (5.16) hold. Since $s \in \mathcal{S}$ is δ-NGA for x, we have $(x_i^s - x) \cdot (-q) > \delta$ for i such that $\hat{\zeta}_i > 0$. Therefore, by (5.16),

$$\tilde{\mu}_s > \tilde{\mu}_s + (x_i^s - x) \cdot q + \delta = \phi(x_i^s) + \delta,$$

for i such that $\hat{\zeta}_i > 0$. □

5.3.2 Anisotropy-angle-boundedness

A less general but more concrete way of ensuring causality of the discrete equation $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ is to bound the maximum angle between any two direction vectors in $\mathcal{A}$ that intersect a simplex $s \in \mathcal{S}$ from x. We show that δ-AAB of a simplex s implies δ-NGA, which in turn implies δ-causality. δ-AAB is easy to implement, so we use it in Algorithm 2. Figure 5.2 is a geometric aid in understanding the definition of and proofs involving δ-anisotropy-angle-boundedness.

Define $\check{f}(x) = \min_{a \in \mathcal{A}} f(x, a)$ and $\hat{f}(x) = \max_{a \in \mathcal{A}} f(x, a)$. Let

$$\Upsilon(x) = \hat{f}(x)/\check{f}(x)$$

be the local anisotropy coefficient. Note that $0 < \check{f}(x) \leq \hat{f}(x) < \infty$, so $1 \leq \Upsilon(x) < \infty$. Denote

$$a_i^s = (x_i^s - x)/\|x_i^s - x\|.$$

Define $\alpha_{i,j}^s$ to be the angle between a_i^s and a_j^s. Let

$$\hat{\alpha}_s = \max_{i,j} \alpha_{i,j}^s.$$

Let $r_s(x)$ be the minimum distance between x and any vertex of s:

$$r_s(x) = \min_i \|x_i^s - x\|. \tag{5.21}$$

Definition. We say that $s \in \mathcal{S}$ is *δ-anisotropy-angle-bounded (δ-AAB) for $x \in \Omega$* if $\delta \hat{f}(x)/r_s(x) \leq 1$ and

$$\hat{\alpha}_s < \arccos(\delta \hat{f}(x)/r_s(x)) - \arccos(1/\Upsilon(x)). \tag{5.22}$$

We say that $\mathcal{S}$ is *δ-AAB for x* if all $s \in \mathcal{S}$ are δ-AAB for x. When $\delta = 0$, we may say s (or $\mathcal{S}$) is *AAB for x*.

Theorem 5.9. *If $s \in \mathcal{S}$ is δ-AAB for $x \in \Omega$, then s is δ-NGA for x.*

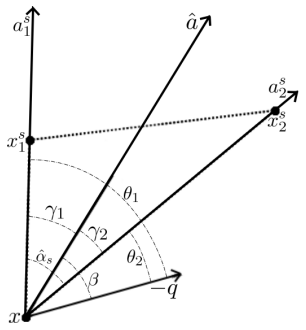


Figure 5.2: Depiction of symbols used in the definition of δ-AAB and the proof of Theorem 5.9.

The following proof builds on analysis done in [60, Section 3.4] that bounds the angle between the optimal action and the negative gradient of the viscosity solution u of (1.8).

Proof. Let $q \in \mathbb{R}^d$ be such that $\max_{a\in\mathcal{A}}(-q\cdot a)f(x,a) = 1$ and $\hat{a} \in \operatorname{argmax}_{a\in\mathcal{A}}(-q\cdot a)f(x,a)$ such that $\hat{a}$ intersects s from x. We have

$$(-q\cdot\hat{a})f(x,\hat{a}) \geq (-q\cdot(-q/\|-q\|))\, f\left(x,-q/\|-q\|\right) \geq \|q\|\check{f}(x).$$

Let β be the angle between $-q$ and $\hat{a}$. Since $\|\hat{a}\| \leq 1$, we have

$$\cos(\beta) = \frac{-q\cdot\hat{a}}{\|-q\|\|\hat{a}\|} \geq \frac{\check{f}(x)}{f(x,\hat{a})} \geq \frac{1}{\Upsilon(x)}. \tag{5.23}$$

Because s is δ-AAB for x and by (5.23), $\beta \leq \arccos(1/\Upsilon(x))$, we have

$$\hat{\alpha}_s + \beta < \arccos(\delta\hat{f}(x)/r_s(x)). \tag{5.24}$$

Let $1 \leq i \leq n_s$. Let γ_i be the angle between $\hat{a}$ and a_i^s. Since $\hat{a}$ intersects s from x, we have $\gamma_i \leq \hat{\alpha}_s$. Let θ_i be the angle between $-q$ and a_i^s. By (5.24), we have

$$\theta_i \leq \gamma_i + \beta \leq \hat{\alpha}_s + \beta < \arccos(\delta\hat{f}(x)/r_s(x)). \tag{5.25}$$

Since $\|\hat{a}\| \leq 1$, we have

$$\|q\|\hat{f}(x) = (-q \cdot (-q/\| - q\|))\hat{f}(x) \geq (-q \cdot \hat{a})f(x, \hat{a}) = 1,$$

and by (5.21), we have

$$0 \leq \frac{\delta}{\|x_i^s - x\|\|q\|} \leq \frac{\delta \hat{f}(x)}{r_s(x)} \leq 1,$$

where the final inequality is because s is δ-AAB. By this inequality and (5.25), it follows that

$$\cos(\theta_i) > \frac{\delta \hat{f}(x)}{r_s(x)} \geq \frac{\delta}{\|x_i^s - x\|\|q\|}.$$

Consequently, we have

$$(x_i^s - x) \cdot (-q) = \|x_i^s - x\|\| - q\| \cos(\theta_i) > \delta$$

for $1 \leq i \leq n_s$. Therefore, s is δ-NGA. □

We describe a way to increase the upper bound on $\hat{\alpha}_s$ for the δ-AAB of a simplex s. This modification may result in more simplices satisfying the definition, which may allow us to find a DC and δ-NGA set of update simplices $\mathcal{S}(x)$ occupying a smaller region around x and reduce the truncation error. First we define some simplex specific notation. Let $\check{f}_s(x) = \min_{a \in \check{\mathcal{A}}_s} f(x, a)$, where

$$\check{\mathcal{A}}_s = \{q \in \mathcal{A} \mid \operatorname*{argmax}_{a \in \mathcal{A}}(-q \cdot a)f(x, a) \text{ intersects } s \text{ from } x\}.$$

Let $\hat{f}_s(x) = \max_{a \in \hat{\mathcal{A}}_s} f(x, a)$, where

$$\hat{\mathcal{A}}_s = \{a \in \mathcal{A} \mid a \text{ intersects } s \text{ from } x\}.$$

Then let $\Upsilon_s(x) = \hat{f}_s(x)/\check{f}_s(x)$ be a simplex specific anisotropy coefficient. We have $\check{f}_s(x) \geq \check{f}(x)$ since $\check{\mathcal{A}}_s \subset \mathcal{A}$ and $\hat{f}_s(x) \leq \hat{f}(x)$ since $\hat{\mathcal{A}}_s \subset \mathcal{A}$. It

follows that

$$\Upsilon_s(x) = \hat{f}_s(x)/\check{f}_s(x) \leq \hat{f}(x)/\check{f}(x) = \Upsilon(x).$$

If it is possible to compute $\hat{f}_s(x)$ and $\check{f}_s(x)$, we can modify the definition of δ-AAB to use the loosened restriction $\delta\hat{f}_s(x)/r_s(x) \leq 1$ and

$$\hat{\alpha}_s < \arccos(\delta\hat{f}_s(x)/r_s(x)) - \arccos(1/\Upsilon_s(x)).$$

Theorem 5.9 still holds.

The definition of δ-AAB can be simplified if we restrict the problem in several ways. These restrictions serve to draw connections to previous work. Also, the restriction $\delta = 0$ is useful in the implementation of Algorithm 2. If we take $\Upsilon(x) = 1$, (5.22) becomes

$$\hat{\alpha}_s < \arccos(\delta\hat{f}(x)/r_s(x))$$

or, equivalently,

$$\cos(\hat{\alpha}_s) < \delta\hat{f}(x)/r_s(x).$$

This resembles a formula for the optimal bucket width δ for a Dial-like algorithm to solve the Eikonal equation derived in [65]. On the other hand, if we take $\delta = 0$, (5.22) becomes

$$\hat{\alpha}_s < \arcsin(1/\Upsilon(x)).$$

We use this condition in the Dijkstra-like Algorithm 2. Finally, if we take both $\Upsilon(x) = 1$ and $\delta = 0$, (5.22) becomes $\hat{\alpha}_s < \pi/2$. In the appendix of [60], it is shown that the slightly looser condition $\hat{\alpha}_s \leq \pi/2$ is sufficient for causality of (5.8).

5.3.3 Distance-ratio-boundedness

If the ratio of the minimum distance between x and any node in simplex s and the maximum distance between nodes in s is large enough then $s \in \mathcal{S}$ must be AAB for x. Proposition 5.12 provides a lower bound for this ratio that is sufficient for AAB. We use DRB in the proof of correctness of Algo-

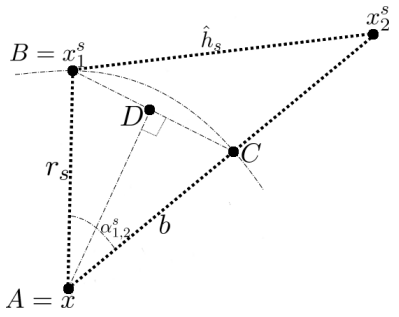


Figure 5.3: Depiction of symbols used in the definition of DRB and the proof of Lemma 5.10.

rithm 3 in the case when $\delta = 0$. We do not parameterize DRB by δ because it is difficult to determine a simple and tight lower bound on the ratio for general nonnegative δ. Figure 5.3 is a geometric aid in understanding the definition of and proofs involving distance-ratio-boundedness.

Let $\hat{h}_s$ be the maximum grid edge distance in s:

$$\hat{h}_s = \max_{i,j} \|x_i^s - x_j^s\|. \tag{5.26}$$

Lemma 5.10. *Let* $\hat{h}_s/(2r_s(x)) \leq 1$. *The inequality* $\hat{\alpha}_s \leq 2\arcsin(\hat{h}_s/(2r_s(x)))$ *holds.*

Proof. Let i, j be such that $1 \leq i \leq n_s$, $1 \leq j \leq n_s$, and $i \neq j$. Let $b = \min\{\|x_i^s - x\|, \|x_j^s - x\|\}$. Form an isosceles triangle with apex $A = x$ and the other two vertices $B = x + ba_i^s$, and $C = x + ba_j^s$. We bound the length of the base above:

$$\|B - C\| \leq \|x_i^s - x_j^s\| \leq \hat{h}_s.$$

By (5.21) we bound the length of either side below:

$$b = \|A - B\| = \|A - C\| \geq r_s(x).$$

We split the isosceles triangle ABC in half to obtain a right-angle triangle

with vertices $A = x$, $B = x + ba_i^s$, and $D = x + b(a_i^s + a_j^s)/2$. We have

$$\sin\left(\frac{\alpha_{i,j}^s}{2}\right) = \frac{\|B - D\|}{b} = \frac{\|B - C\|}{2b} \leq \frac{\hat{h}_s}{2r_s(x)}. \tag{5.27}$$

By the properties of the simplex s, $0 < \alpha_{i,j}^s/2 < \pi/2$. By (5.27), $\alpha_{i,j}^s \leq 2\arcsin(\hat{h}_s/(2r_s(x)))$ for any i and j. This implies that $\hat{\alpha}_s \leq 2\arcsin(\hat{h}_s/(2r_s(x)))$. □

Proposition 5.11. *Let $x \in \Omega$ and $s \in \mathcal{S}$. If $\hat{h}_s/(2r_s(x)) \leq 1$, $\delta\hat{f}(x)/r_s(x) \leq 1$, and*

$$2\arcsin(\hat{h}_s/(2r_s(x))) < \arccos(\delta\hat{f}(x)/r_s(x)) - \arccos(1/\Upsilon(x)) \tag{5.28}$$

then s is δ-AAB for x

Proof. By (5.28), Lemma 5.10, and (5.22) in the definition of δ-AAB, s is δ-AAB for x. □

Equation (5.28) can be simplified if we restrict the problem in several ways. These simplications are useful for building intuition about Proposition 5.11. Furthermore, the restriction $\delta = 0$ is used to form our definition of DRB. If we take $\Upsilon(x) = 1$, (5.28) becomes

$$2\arcsin(\hat{h}_s/(2r_s(x))) < \arccos(\delta\hat{f}(x)/r_s(x)).$$

On the other hand, if we take $\delta = 0$, (5.28) becomes

$$2\arcsin(\hat{h}_s/(2r_s(x))) < \arcsin(1/\Upsilon(x)).$$

From this condition, we can determine a lower bound $\Psi(x)$ on the ratio $r_s(x)/\hat{h}_s$:

$$\frac{r_s(x)}{\hat{h}_s} > \left[2\sin\left(\frac{\arcsin(1/\Upsilon(x))}{2}\right)\right]^{-1} = \Psi(x). \tag{5.29}$$

Finally, if we take both $\Upsilon(x) = 1$ and $\delta = 0$, (5.28) becomes $\hat{h}_s/r_s(x) < \sqrt{2}$, which implies $\hat{\alpha}_s < \pi/2$, the condition for AAB in this case.

Definition. We say that $s \in \mathcal{S}$ is *distance-ratio-bounded (DRB) for* $x \in \Omega$ if $\delta = 0$ and (5.29) holds. We say that $\mathcal{S}$ is *DRB for* x if all $s \in \mathcal{S}$ are DRB for x.

Proposition 5.12. *If $s \in \mathcal{S}$ is DRB for $x \in \Omega$ then s is AAB for x.*

Proof. By the definition of DRB, (5.29) holds, which is equivalent to (5.28), when $\delta = 0$. Note that $\Psi(x) \geq 1/\sqrt{2}$ (see Remark 5.13 below), which means that by (5.29), $\hat{h}_s/(2r_s(x)) < 1/\sqrt{2} \leq 1$. Also, since $\delta = 0$ we have $\delta \hat{f}(x)/r_s(x) = 0 \leq 1$. Therefore, by Proposition 5.11, s is AAB for x. □

Remark 5.13. *If we simplify the definition of DRB by replacing $\Psi(x)$ with $\Upsilon(x)$ in (5.29), Proposition 5.12 still holds. However, $\Psi(x) < \Upsilon(x)$, so using $\Psi(x)$ to define DRB results in a looser restriction on simplices. For $1 \leq \Upsilon(x) < \infty$, since $\arcsin(1/\Upsilon(x)) > 2\arcsin(1/(2\Upsilon(x)))$, we have $\Psi(x) < \Upsilon(x)$. When $\Upsilon(x) = 1$, $\Psi(x) = 1/\sqrt{2}$ and $\lim_{\Upsilon(x)\to\infty}[\Psi(x)/\Upsilon(x)] = 1$. Finally, for $1 \leq \Upsilon(x) < \infty$, $\Psi(x)$ increases as $\Upsilon(x)$ increases.*

5.4 Algorithm

In Algorithm 2 we define the MAOUM algorithm. Algorithm 2 solves the discrete system (3.1). For (3.1) to be well-defined $\mathcal{S}(x)$ must be determined. The update simplex set $\mathcal{S}(x)$ is chosen to ensure $\underline{H}(x, \mathcal{S}(x), \phi, \mu)$ is consistent and the discrete equation $\underline{H}(x, \mathcal{S}(x), \phi, \mu) = 0$ is causal. Let

$$\mathcal{S}(x) = \{s \in \underline{\mathcal{S}}(\overrightarrow{\mathcal{Y}}(x)) \mid s \text{ is AAB for } x\}, \tag{5.30}$$

where $\overrightarrow{\mathcal{Y}}(x)$ is the stencil or *update node set of* x. We update $\underline{v}(x)$ from simplex s using `Update`(x, s), which evaluates to the solution $\tilde{\mu}_s$ of (5.8), where $\phi(x_i^s) = \underline{v}(x_i^s)$. In Algorithm 2, the node x is included in $\overrightarrow{\mathcal{Y}}(x)$ and $\overleftarrow{\mathcal{Y}}(x)$ for convenience of proofs in Section 5.4.1. However, the algorithm is also correct if x is excluded from $\overrightarrow{\mathcal{Y}}(x)$ and $\overleftarrow{\mathcal{Y}}(x)$.

In Section 5.4.1 we describe how the subroutine `ComputeUpdateSet` defined in Algorithm 3 determines $\overrightarrow{\mathcal{Y}}(x)$ such that $\mathcal{S}(x)$ satisfies DC and AAB for x. We revisit the convergence of $\underline{u}$ to the solution u of the HJ PDE

symbol	type or definition	description
$\mathcal{X}$	$= \underline{\Omega} \cup \underline{\partial\Omega}$	set of all grid nodes
$\underline{\Omega}$	subset of $\mathcal{X}$	discretized domain
$\underline{\partial\Omega}$	subset of $\mathcal{X}$	discretized domain boundary
x	$\mathbb{R}^d$	domain point or grid node
$\mathcal{N}(x)$	$\mathcal{X} \to$ subset of $\mathcal{X}$	set of grid neighbors of node x
s	convex subset of $\mathbb{R}^d$	simplex
$\underline{\mathcal{S}}(\mathcal{R})$	set of simplices	grid simplices with all vertices in $\mathcal{R}$ and $1 \le n_s \le d$
$\overrightarrow{\mathcal{Y}}(x)$	$\mathcal{X} \to$ subset of $\mathcal{X}$	update node set of x, i.e., nodes upon which x might depend
$\mathcal{S}(x)$	$\mathcal{X} \to$ set of simplices	update simplex set of x
$\underline{H}(x, \mathcal{S}, \phi, \mu)$	see (5.4)	numerical (discrete) Hamiltonian
$\underline{u}(x)$	$\mathcal{X} \to \mathbb{R}$	solution of (5.5) at node x
$\underline{v}(x)$	$\mathcal{X} \to \mathbb{R}$	(temporary) value of node x
$\mathcal{H}$	subset of $\mathcal{X}$	min-value heap of unaccepted nodes
$\overleftarrow{\mathcal{Y}}(x)$	$\mathcal{X} \to$ subset of $\mathcal{X}$	dependent node set of x, i.e., nodes which might depend on x
$\underline{\mathcal{S}}_d$	set of simplices	grid simplices with $n_s = d$
$\texttt{Update}(x, s)$	$\mathcal{X} \times \underline{\mathcal{S}} \to \mathbb{R}$	value update of node x from simplex s, i.e., $\tilde{\mu}_s$ from (5.8) with $\phi(x_i^s) = \underline{v}(x_i^s)$
$\mathcal{Q}$	subset of $\mathcal{X}$	queue to become update nodes
$\mathcal{M}(s)$	$\underline{\mathcal{S}} \to$ subset of $\mathcal{X}$	set of neighbors of all vertex nodes in s
$\mathcal{B}(x, r)$	convex subset of $\mathbb{R}^d$	ball of radius r centered on x
$\mathcal{B}_1(x)$	convex subset of $\mathbb{R}^d$	ball centered on x defined in (5.31)
$\mathcal{B}_2(x)$	convex subset of $\mathbb{R}^d$	ball centered on x defined in (5.32)
$\Psi(x)$	$\Omega \to \mathbb{R}$	a measure of anisotropy from (5.29)

Table 5.1: Summary of symbols. The first group is used in defining the numerical problem, the second group is used in Algorithm 2, the third group is used only in Algorithm 3, and the forth group is used in proofs of algorithm correctness and complexity.

(1.8) as the grid spacing goes to zero in Section 5.4.2. In Section 5.4.3 we examine the computational complexity of MAOUM. One of the drawbacks of MAOUM is that the update node set $\overrightarrow{\mathcal{Y}}(x)$ must be stored for each x, which may require a significant amount of memory. In Section 5.4.4 we show that it is possible to instead store a superset of $\overrightarrow{\mathcal{Y}}(x)$ which has a compact

representation.

foreach $x \in \underline{\Omega}$ **do** $\underline{v}(x) \leftarrow \infty$
foreach $x \in \underline{\partial\Omega}$ **do** $\underline{v}(x) \leftarrow g(x)$
foreach $x \in \mathcal{X}$ **do** $\overleftarrow{\mathcal{Y}}(x) \leftarrow \{x\}$, $\overrightarrow{\mathcal{Y}}(x) \leftarrow \{x\}$
foreach $x \in \underline{\Omega}$ **do** `ComputeUpdateSet`(x)
$\mathcal{H} \leftarrow \mathcal{X}$
while $\mathcal{H} \neq \emptyset$ **do**
    $x \leftarrow \operatorname{argmin}_{y \in \mathcal{H}} \underline{v}(y)$
    $\mathcal{H} \leftarrow \mathcal{H} \setminus \{x\}$
    foreach $y \in \overleftarrow{\mathcal{Y}}(x) \cap \mathcal{H}$ **do**
        foreach $s \in \underline{\mathcal{S}}(\overrightarrow{\mathcal{Y}}(y) \setminus \mathcal{H})$ s.t. $x \in s$ and s is AAB for y **do**
            $\underline{v}(y) \leftarrow \min(\underline{v}(y), \texttt{Update}(y, s))$
        end
    end
end

Algorithm 2: Monotone Acceptance Ordered Upwind Method (MAOUM)

$\mathcal{Q} \leftarrow \mathcal{N}(x)$
while $\mathcal{Q} \neq \emptyset$ **do**
    $y \leftarrow \texttt{Pop}(\mathcal{Q})$
    $\overrightarrow{\mathcal{Y}}(x) \leftarrow \overrightarrow{\mathcal{Y}}(x) \cup \{y\}$
    $\overleftarrow{\mathcal{Y}}(y) \leftarrow \overleftarrow{\mathcal{Y}}(y) \cup \{x\}$
    foreach $s \in \underline{\mathcal{S}}_d(\overrightarrow{\mathcal{Y}}(x))$ s.t. $x \notin s$ and $y \in s$ **do**
        if s not AAB for x **then**
            $\mathcal{Q} \leftarrow \mathcal{Q} \cup \left(\mathcal{M}(s) \setminus \overrightarrow{\mathcal{Y}}(x)\right)$
        end
    end
end

Algorithm 3: `ComputeUpdateSet`(x)

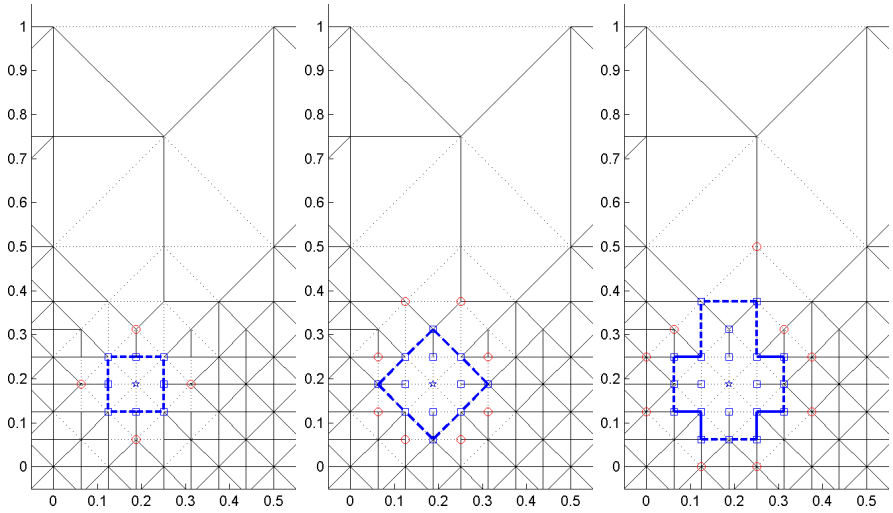


Figure 5.4: (Continued in following figure) The status of Algorithm 3 during different stages of the computation of the update node set of x. The star is node x. Squares are nodes in the update node set. Circles are nodes in $\mathcal{Q}$. Thin solid lines are grid edges that are AAB for x. Thin dotted lines are grid edges which are not AAB. Thick lines are grid edges on the frontier of the update node set. Thick solid lines are AAB and thick dashed lines are not AAB. The top-left shows the moment just after neighbors of x have been added to the update node set and the frontier edges for which they are vertices have failed the AAB test. As a result, all nodes opposite these edges have been added to $\mathcal{Q}$. Note that the order in which nodes are removed from $\mathcal{Q}$ is arbitrary, although our implementation uses first-in first-out order. Subsequent plots show subsequent but not sequential stages left to right.

5.4.1 Computing the Update Set

The status of Algorithm 3 during different stages of computing $\overrightarrow{\mathcal{Y}}(x)$ is shown in Figures 5.4 and 5.5. To achieve more accurate and efficient computations in locally-refined parts of the grid, we desire the maximum extent of the update node set, $\hat{r}(x) = \max_{y \in \overrightarrow{\mathcal{Y}}(x)} \|x - y\|$, to shrink towards zero as the local grid spacing goes to 0. By proving Theorem 5.17, we show that the subroutine call `ComputeUpdateSet`(x) (Algorithm 3) terminates in a finite

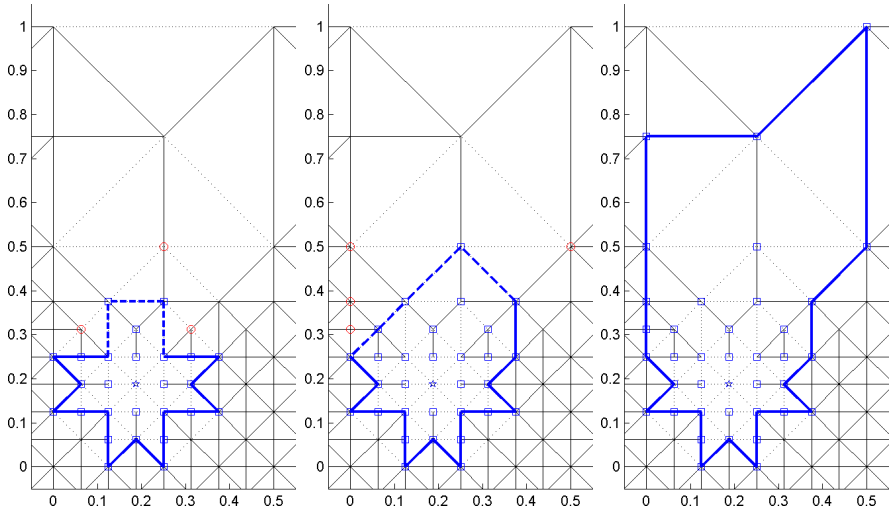


Figure 5.5: (Continued from previous figure) The lower-right shows the status at the termination of Algorithm 3. All grid edges on the frontier of the update node set have passed the AAB test. Subsequent plots show subsequent but not sequential stages left to right.

number of iterations with a bound on $\hat{r}(x)$ that varies linearly with the local grid resolution. We further show that for x not too near the boundary $\underline{\partial\Omega}$, $\mathcal{S}(x)$ defined in (5.30) is DC and NGA, which is sufficient for consistency and causality. First we define notation and prove some useful lemmas.

Let $\mathcal{Z} \subseteq \mathcal{X}$. The set $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$ contains the d-dimensional grid simplices with all vertex nodes in $\mathcal{Z}$. Define

$$\mathcal{U}_{\mathcal{Z}} = \bigcup_{s \in \underline{\mathcal{S}}_{d+1}(\mathcal{Z})} (s),$$

which is the d-dimensional set covered by the simplices in $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$.

Define

$$\mathcal{M}(s) = \{y \in \mathcal{X} \mid y \in \mathcal{N}(x_i^s) \text{ for all } i \text{ such that } 1 \leq i \leq n_s\}.$$

Define

$$\mathcal{F}_\partial(\mathcal{Z}) = \{s \in \underline{\mathcal{S}}_d(\mathcal{Z}) \mid \mathcal{M}(s) \setminus \mathcal{Z} \neq \emptyset \text{ or for all } j,\, x_j^s \in \underline{\partial\Omega}\}.$$

The set $\mathcal{F}_\partial(\mathcal{Z})$ contains the $(d-1)$-dimensional grid simplex faces with all vertex nodes in $\mathcal{Z}$, such that there is a neighbor of all vertex nodes which is not in $\mathcal{Z}$ or all vertex nodes are on the boundary of the grid. Define

$$\mathcal{U}_{\partial\mathcal{Z}} = \bigcup_{s \in \mathcal{F}_\partial(\mathcal{Z})} (s),$$

which is the $(d-1)$-dimensional surface covered by the simplices in $\mathcal{F}_\partial(\mathcal{Z})$.

Lemma 5.14. *Let $\partial\mathcal{U}_\mathcal{Z}$ be the boundary of $\mathcal{U}_\mathcal{Z}$. Then $\partial\mathcal{U}_\mathcal{Z} \subseteq \mathcal{U}_{\partial\mathcal{Z}}$.*

Proof. Since $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$ contains only grid simplices which do not overlap except at their boundary, $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$ is a partition of $\mathcal{U}_\mathcal{Z}$. It follows that any point $z \in \partial\mathcal{U}_\mathcal{Z}$ must be on a $(d-1)$-dimensional face s of just one d-dimensional simplex in $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$. We have $s \in \underline{\mathcal{S}}_d(\mathcal{Z})$. Either $\mathcal{M}(s) \setminus \mathcal{Z} \neq \emptyset$ or for all j, $x_j^s \in \underline{\partial\Omega}$. By definition $s \in \mathcal{F}_\partial(\mathcal{Z})$. Thus, $z \in \mathcal{U}_{\partial\mathcal{Z}} = \bigcup_{s \in \mathcal{F}_\partial(\mathcal{Z})}(s)$. □

For the following proofs we are only concerned with a single execution of Algorithm 3. Also, we are only considering the update node set $\overrightarrow{\mathcal{Y}}(x)$ and not the dependent node set $\overleftarrow{\mathcal{Y}}(x)$, so we abbreviate $\mathcal{Y} = \overrightarrow{\mathcal{Y}}(x)$, to make the notation less cluttered. We now prove a lemma stating that throughout the execution of Algorithm 3, $\mathcal{F}_\partial(\mathcal{Y})$ is DC for x. Let a subscript $i \geq 0$ represent the state of a variable at the beginning of the $(i+1)^{\text{st}}$ iteration of the **while** loop in Algorithm 3. For example, $\mathcal{Y}_i$ is the state of the update node set $\mathcal{Y}$ at the beginning of iteration $i+1$. From Lines 3 and 4 of Algorithm 3, we have $y_{i+1} = \texttt{Pop}(\mathcal{Q}_i)$ and $\mathcal{Y}_{i+1} = \mathcal{Y}_i \cup \{y_{i+1}\}$.

Lemma 5.15. *Suppose that $x \notin \underline{\partial\Omega}$. For $i \geq |\mathcal{N}(x)|$, $\mathcal{F}_\partial(\mathcal{Y}_i)$ is DC for x.*

Proof. Suppose that $x \in \mathcal{U}_{\mathcal{Y}_i} \setminus \partial\mathcal{U}_{\mathcal{Y}_i}$. Any path $\xi : [0,1] \to \Omega$, such that $\xi(0) = x$ and $\xi(1) \notin \mathcal{U}_{\mathcal{Y}_i}$ intersects $\partial\mathcal{U}_{\mathcal{Y}_i}$. In particular, any path ξ of the form $\xi(t) = x + tCa$, where $t \in [0,1]$, $C \in \mathbb{R}^+$ is a constant, $a \in \mathcal{A}$, and

$\xi(1) \notin \mathcal{U}_{\mathcal{Y}_i}$ intersects $\partial\mathcal{U}_{\mathcal{Y}_i}$. By Lemma 5.14, such a path also intersects $\mathcal{U}_{\partial\mathcal{Y}_i}$. Since this fact holds for any $a \in \mathcal{A}$ and $\mathcal{U}_{\partial\mathcal{Y}_i}$ is the union of simplices in $\mathcal{F}_\partial(\mathcal{Y}_i)$, $\mathcal{F}_\partial(\mathcal{Y}_i)$ is DC for x. We now prove by induction our supposition that $x \in \mathcal{U}_{\mathcal{Y}_i} \setminus \partial\mathcal{U}_{\mathcal{Y}_i}$.

Consider the base case: $i = |\mathcal{N}(x)|$. By Line 3 of Algorithm 2, and Lines 1, 3, and 4 of Algorithm 3, we have $\mathcal{Y}_i = \{x\} \cup \mathcal{N}(x)$. Thus, $\underline{\mathcal{S}}_{d+1}(\mathcal{Y}_i)$ includes all d-dimensional simplices which have x as a vertex. Since $x \notin \underline{\partial\Omega}$, we have $x \in \mathcal{U}_{\mathcal{Y}_i} \setminus \partial\mathcal{U}_{\mathcal{Y}_i}$.

Now consider the inductive step. Suppose that $x \in \mathcal{U}_{\mathcal{Y}_i} \setminus \partial\mathcal{U}_{\mathcal{Y}_i}$. We have $\mathcal{Y}_{i+1} = \mathcal{Y}_i \cup \{y_{i+1}\}$. Thus, $\underline{\mathcal{S}}_{d+1}(\mathcal{Y}_{i+1})$ includes just the d-dimensional simplices in $\underline{\mathcal{S}}_{d+1}(\mathcal{Y}_i)$ plus any other d-dimensional simplices which have y_{i+1} as a vertex and nodes in $\mathcal{Y}_i$ for all other vertices. It follows that $\underline{\mathcal{S}}_{d+1}(\mathcal{Y}_{i+1}) \supseteq \underline{\mathcal{S}}_{d+1}(\mathcal{Y}_i)$. Since, $x \in \mathcal{U}_{\mathcal{Y}_i} \setminus \partial\mathcal{U}_{\mathcal{Y}_i}$, we have $x \in \mathcal{U}_{\mathcal{Y}_{i+1}} \setminus \partial\mathcal{U}_{\mathcal{Y}_{i+1}}$. □

Let $\hat{h} = \max\{\|y - z\| \mid y \in \mathcal{X} \text{ and } z \in \mathcal{N}(y)\}$ be the maximum grid edge length. Let $\hat{h}(\mathcal{R}) = \max\{\|y - z\| \mid y \in \mathcal{R} \cap \mathcal{X} \text{ and } z \in \mathcal{N}(y)\}$ be the maximum length of grid edges with at least one end node in $\mathcal{R} \subset \mathbb{R}^d$. Let $\mathcal{B}(x, r)$ be a closed ball of radius r around point $x \in \mathbb{R}^d$. The following lemma establishes that we can define a ball centered on x with radius linear in the maximum length of grid edges within the ball. This concept is used to define local grid spacing in Theorem 5.17.

Lemma 5.16. *For all x and all $b \in \mathbb{R}^+$ there exists $\tilde{r} \in \mathbb{R}^+$ such that $0 < \tilde{r} = b\hat{h}(\mathcal{B}(x, \tilde{r})) < \infty$.*

Proof. We have

$$b\hat{h}(\mathcal{B}(x, 0)) = b \max\{\|x - y\| \mid y \in \mathcal{N}(x)\} > 0,$$

$b\hat{h}(\mathcal{B}(x, \tilde{r}))$ nondecreasing on $\tilde{r}$, and $\lim_{\tilde{r}\to\infty} b\hat{h}(\mathcal{B}(x, \tilde{r})) = b\hat{h} < \infty$. Therefore, there exists $\tilde{r}$ such that $0 < \tilde{r} < \infty$ and $\tilde{r} = b\hat{h}(\mathcal{B}(x, \tilde{r}))$. □

As allowed by the previous Lemma, choose $b = \Psi(x) + 1$ and define $\check{r}(x)$ to be the minimum $\tilde{r}$ such that $\hat{h}(\mathcal{B}(x, \tilde{r})) = \tilde{r}/(\Psi(x) + 1)$. Define

$$\mathcal{B}_1(x) = \mathcal{B}(x, \check{r}(x)\Psi(x)/(\Psi(x) + 1)) \tag{5.31}$$

and

$$\mathcal{B}_2(x) = \mathcal{B}(x, \check{r}(x)), \tag{5.32}$$

where $\Psi(x)$ is defined in (5.29). Since $\Psi(x) > 0$, we have $\mathcal{B}_1(x) \subset \mathcal{B}_2(x)$. Define $\mathcal{B}_1^C(x) = \Omega \backslash \mathcal{B}_1(x)$, and $\mathcal{B}_2^C(x) = \Omega \backslash \mathcal{B}_2(x)$. Let $\lambda(x) = \min_{y \in \mathcal{B}_1(x), z \in \mathcal{B}_2^C(x)} \|y - z\|$ be the minimum distance between $\mathcal{B}_1(x)$ and $\mathcal{B}_2^C(x)$. We have

$$\lambda(x) > \check{r}(x) - \check{r}(x)\Psi(x)/(\Psi(x)+1) = \check{r}(x)/(\Psi(x)+1).$$

When x is clear from the context, we may abbreviate $\mathcal{B}_1 = \mathcal{B}_1(x)$, $\mathcal{B}_2 = \mathcal{B}_2(x)$, $\mathcal{B}_1^C = \mathcal{B}_1^C(x)$, $\mathcal{B}_2^C = \mathcal{B}_2^C(x)$, and $\lambda = \lambda(x)$. Let $m(\mathcal{R}) = |\mathcal{X} \cap \mathcal{R}|$ be the number of grid nodes in $\mathcal{R} \subset \mathbb{R}^d$.

Theorem 5.17. *Let $x \in \underline{\Omega}$. Let $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$. The subroutine call* `ComputeUpdateSet`(x) *terminates before iteration $m(\mathcal{B}_2)$ of the* **while** *loop with $\hat{r}(x) \leq \check{r}(x)$. The set $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$ is DC for x and AAB for x.*

Section 5.4.2 explains why requiring that x be sufficiently far from the boundary does not impact convergence.

Proof. A node w may be added to $\mathcal{Y}$ at most once. Since $\mathcal{Y}_0 = \{x\}$, we have $|\mathcal{Y}_i| = i + 1$.

Suppose there exists $\check{i} \geq 1$, such that $y_{\check{i}} \in \mathcal{B}_2^C$ and for $1 \leq i < \check{i}$, $y_i \in \mathcal{B}_2$. In other words, $y_{\check{i}}$ is the first node outside of $\mathcal{B}_2$ to be added to $\mathcal{Y}$. By Lines 3 and 4 of Algorithm 3, $y_{\check{i}}$ must have entered $\mathcal{Q}$. So, by Lines 1 and 8, either $y_{\check{i}} \in \mathcal{N}(x)$ or $y_{\check{i}} \in \mathcal{M}(s)$ for $s \in \underline{\mathcal{S}}_d(\mathcal{B}_2)$. If $y_{\check{i}} \in \mathcal{N}(x)$, then

$$\|x - y_{\check{i}}\| \leq \hat{h}(\mathcal{B}_2) = \frac{\check{r}(x)}{\Psi(x)+1} < \check{r}(x),$$

which contradicts the supposition that $y_{\check{i}} \in \mathcal{B}_2^C$. So $y_{\check{i}} \notin \mathcal{N}(x)$, which means $y_{\check{i}} \in \mathcal{M}(s)$, for s such that $x_k^s \in \mathcal{B}_2 \cap \mathcal{X}$ for $1 \leq k \leq d$. Since

$$\|x_k^s - y_{\check{i}}\| \leq \hat{h}(\mathcal{B}_2) = \frac{\check{r}(x)}{\Psi(x)+1)} < \lambda,$$

it must be that $x_k^s \in \mathcal{B}_1^C$ for $1 \leq k \leq d$. By (5.21) and the definition of $\mathcal{B}_1^C$,

we have

$$r_s(x) > \frac{\check{r}(x)\Psi(x)}{\Psi(x)+1}.$$

Thus, by (5.26), since

$$\hat{h}_s \leq \hat{h}(\mathcal{B}_2) = \frac{\check{r}(x)}{\Psi(x)+1}$$

we have

$$\frac{r_s(x)}{\hat{h}_s} > \frac{\check{r}(x)\Psi(x)/(\Psi(x)+1)}{\hat{h}(\mathcal{B}_2)} = \Psi(x). \tag{5.33}$$

By definition, s is DRB for x. By Proposition 5.12, s is AAB for x and by the **if** condition in Line 7, $y_{\tilde{i}}$ did not enter $\mathcal{Q}$, which is a contradiction.

Thus, there does not exist $\check{i} \geq 1$, such that $y_{\tilde{i}} \in \mathcal{B}_2^C$. Because $|\mathcal{Y}_i| = i+1$, the algorithm terminates before iteration $m(\mathcal{B}_2)$ of the **while** loop. Let $\tilde{i}$ be the last **while** iteration. We have $|\mathcal{N}(x)| \leq \tilde{i} < m(\mathcal{B}_2)$ and $\hat{r}(x) = \max_{y\in\mathcal{Y}_{\tilde{i}}} \|x-y\| \leq \check{r}(x)$.

Consider each $(d-1)$-dimensional simplex face $s \in \mathcal{F}_\partial(\mathcal{Y}_{\tilde{i}})$. Because $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$, $x \notin \underline{\partial\Omega}$. So, since $\mathcal{N}(x) \subseteq \mathcal{Y}_{\tilde{i}}$, x is not a vertex of s. There exists $\check{j} \leq \tilde{i}$ such that $y_{\check{j}} = x_k^s$ for some k such that $1 \leq k \leq d$ and $x_k^s \in \mathcal{Y}_{\check{j}}$ for all k such that $1 \leq k \leq d$. In other words, $\check{j}$ is the first **while** iteration when all vertices of s are in $\mathcal{Y}$. By the **foreach** loop, if s is not AAB for x then $\mathcal{M}(s) \subset \mathcal{Q}_{\check{j}}$, which implies $\mathcal{M}(s) \subset \mathcal{Y}_{\tilde{i}}$, meaning $\mathcal{M}(s) \setminus \mathcal{Y}_{\tilde{i}} = \emptyset$. But since $s \in \mathcal{F}_\partial(\mathcal{Y}_{\tilde{i}})$, $\mathcal{M}(s) \setminus \mathcal{Y}_{\tilde{i}} \neq \emptyset$ or for all k, $x_k^s \in \underline{\partial\Omega}$. It follows that s is AAB or for all k, $x_k^s \in \underline{\partial\Omega}$. Since $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$, if for all k, $x_k^s \in \underline{\partial\Omega}$, then for all k, $x_k^s \in \mathcal{B}_1^C$. Thus, by (5.33), s is DRB for x, implying s is AAB for x. Therefore, we have $\mathcal{F}_\partial(\mathcal{Y}_{\tilde{i}})$ is AAB for x, and by Lemma 5.15, DC for x. □

The following corollary states that for x not too near the boundary $\underline{\partial\Omega}$, $\mathcal{S}(x)$ is DC for x and NGA for x. By Proposition 5.2, DC for x is needed for the consistency of the Hamiltonian $\underline{H}(x, \mathcal{S}(x), \phi, \mu)$. By Theorem 5.8, the fact that $\mathcal{S}(x)$ is NGA for x implies (5.8) is causal for x and $s \in \mathcal{S}(x)$.

Corollary 5.18. *Let $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$. Then $\mathcal{S}(x)$ is DC for x and NGA for x.*

Proof. By definition,

$$\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x)) \subseteq \underline{\mathcal{S}}_d(\overrightarrow{\mathcal{Y}}(x)) \subseteq \underline{\mathcal{S}}(\overrightarrow{\mathcal{Y}}(x)).$$

By the definition of $\mathcal{S}(x)$ in (5.30) and since, by Theorem 5.17, $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$ is AAB for x, we have $\mathcal{S}(x) \supseteq \mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$. By Theorem 5.17, $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$ is DC for x, so its superset $\mathcal{S}(x)$ is DC for x. By (5.30), $\mathcal{S}(x)$ is AAB for x, so by Theorem 5.9, $\mathcal{S}(x)$ is NGA for x. □

5.4.2 Convergence

In Section 5.2, we proved that $\underline{H}$ is monotone (Proposition 5.1) and consistent (Propositions 5.2) as long as $\mathcal{S}$ is DC for x. In Corollary 5.18, $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$ is a sufficient condition for $\mathcal{S}$ to be DC, but for x near the computational boundary the condition may not hold. However, the following propositions show that in the limit the update set does not intersect $\underline{\partial\Omega}$, so the scheme is consistent. Let $\check{r}^{\mathcal{G}}(x)$ and $\mathcal{B}_1^{\mathcal{G}}(x)$ be as defined above, but explicitly parameterized by the grid $\mathcal{G}$.

Proposition 5.19. *Property 3.1 (Stencil Radius) holds. Moreover,* $\check{r}^{\mathcal{G}}(x) = \mathcal{O}(\hat{h}^{\mathcal{G}})$*, for all* $x \in \underline{\Omega}$.

Proof. Let $x \in \underline{\Omega}$. By Theorem 5.17 and the definition of $\check{r}^{\mathcal{G}}(x)$, after executing Algorithm 3,

$$\hat{r}^{\mathcal{G}}(x) \leq \check{r}^{\mathcal{G}}(x) \leq (\Psi(x)+1)\hat{h}(\mathcal{B}_2) \leq (\hat{\Psi}+1)\hat{h}^{\mathcal{G}}.$$

□

Proposition 5.20. *Let* $x \in \Omega$ *and* $\phi \in C_b^\infty(\Omega)$*. For sequences* $\mathcal{G}_k$*,* $x_k \in \underline{\Omega}_k$*, and* ξ_k*, such that* $x_k \to x$*,* $\hat{h}^{\mathcal{G}_k} \to 0$*, and* $\xi_k \to 0$ *as* $k \to \infty$*,*

$$\lim_{k\to\infty} \underline{H}(x_k, \mathcal{S}_k, \phi + \xi_k, \phi(x_k) + \xi_k) = H(x, D\phi(x)), \tag{5.34}$$

where $\mathcal{S}_k = \mathcal{S}(\overrightarrow{\mathcal{Y}}^{\mathcal{G}_k}(x_k))$.

Proof. Let $x \in \Omega$ and $\phi \in C_b^\infty(\Omega)$. Let $\mathcal{G}_k$, $x_k \in \underline{\Omega}_k$, and ξ_k be sequences such that $x_k \to x$, $\hat{h}^{\mathcal{G}_k} \to 0$, and $\xi_k \to 0$ as $k \to \infty$. Since $\hat{h}^{\mathcal{G}_k} \to 0$ as $k \to \infty$, by Proposition 5.19, we have $\hat{r}^{\mathcal{G}_k}(x) \to 0$ and $\check{r}^{\mathcal{G}_k}(x) \to 0$, as $k \to \infty$. Also, because Ω is a bounded Lipschitz domain, $x \in \Omega$ is at some minimum distance $\epsilon > 0$ from $\partial\Omega$. So, there exists some $\tilde{k} > 0$ such that for all $k \geq \tilde{k}$, $\underline{\partial\Omega} \cap \mathcal{B}_1^{\mathcal{G}_k} = \emptyset$. By Theorem 5.17, for all $k \geq \tilde{k}$, $\mathcal{S}_k$ is DC for x_k. By the fact that $\hat{r}^{\mathcal{G}_k}(x) \to 0$ as $k \to \infty$ and by Proposition 5.2,

$$\begin{aligned}
&\lim_{k\to\infty} \underline{H}(x_k, \mathcal{S}_k, \phi + \xi_k, \phi(x_k) + \xi_k) \\
&= \lim_{\substack{y\to x,\, \xi\to 0 \\ \hat{r}(y)\to 0}} \underline{H}(y, \mathcal{S}(y), \phi + \xi, \phi(y) + \xi) = H(x, D\phi(x)),
\end{aligned} \tag{5.35}$$

so (5.34) holds. □

To use the convergence proof in Section 3.2 we must also show that $\underline{u}$ is bounded. Virtually the same proof as that in Section 4.3.5 can be used to prove $\underline{u}$ is bounded.

5.4.3 Complexity

We analyze the asymptotic computational complexity of Algorithm 2. We argue that executing the **while** loop in Algorithm 2 is more computationally complex than initialization before the **while** loop. Of the tasks performed during execution of the **while** loop, maintaining the nodes in value order using a heap determines the complexity of the **while** loop and, therefore, the complexity of Algorithm 2. Recall that $N = |\mathcal{X}|$ is the number of grid nodes.

To analyze computational complexity it is useful to prove a lemma bounding the maximum number of nodes in any update region or dependent region as $N \to \infty$. Let $\check{h} = \min\{\|y - z\| \mid y \in \mathcal{X} \text{ and } z \in \mathcal{N}(y)\}$ be the minimum grid edge length and let $\rho = \hat{h}/\check{h}$. Let $\overrightarrow{M} = \max_{x\in\underline{\Omega}} |\overrightarrow{\mathcal{Y}}(x) \cap \mathcal{X}|$ and $\overleftarrow{M} = \max_{x\in\mathcal{X}} |\overleftarrow{\mathcal{Y}}(x) \cap \underline{\Omega}|$ be the maximum number of nodes over all update node sets and dependent node sets, respectively. Let $\hat{\Psi} = \max_{x\in\underline{\Omega}} \Psi(x)$.

Lemma 5.21. *As $N \to \infty$, $\overrightarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$ and $\overleftarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$.*

Proof. By Theorem 5.17, after executing Algorithm 3,

$$\hat{r}(x) \leq \check{r}(x) \leq (\Psi(x)+1)\hat{h}(\mathcal{B}_2) \leq (\hat{\Psi}+1)\hat{h},$$

for all $x \in \underline{\Omega}$. Given $\check{h}$ is the minimum spacing between nodes, we bound above the number of nodes that can be packed into $\mathcal{B}(x, \hat{r}(x))$, for all x:

$$|\mathcal{B}(x,\hat{r}(x)) \cap \mathcal{X}| = \mathcal{O}\left(\left(\frac{\hat{r}(x)}{\check{h}}\right)^d\right) = \mathcal{O}\left(\left(\frac{\hat{\Psi}\hat{h}}{\check{h}}\right)^d\right) = \mathcal{O}((\hat{\Psi}\rho)^d)$$

Therefore, $\overrightarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$.

The symmetry $y \in \overleftarrow{\mathcal{Y}}(x)$ if and only if $x \in \overrightarrow{\mathcal{Y}}(y)$ is evident from Line 3 of Algorithm 2, and Lines 4 and 5 of Algorithm 3. Since $\hat{r}(x) \leq (\hat{\Psi}+1)\hat{h}$ for all $x \in \underline{\Omega}$, the same holds for the maximum extent of the dependent node set: $\max_{y \in \overleftarrow{\mathcal{Y}}(x)} \|x-y\| \leq (\hat{\Psi}+1)\hat{h}$ for all $x \in \mathcal{X}$. Following the same argument as above, $\overleftarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$. □

We first consider the computational cost of initializing the update regions of nodes $x \in \underline{\Omega}$ (and dependent regions of nodes $x \in \mathcal{X}$) using the subroutine Algorithm 3. In Line 4 of Algorithm 2, `ComputeUpdateSet` is called N times, once for each node x. For each call to `ComputeUpdateSet`, there are $\mathcal{O}(\overrightarrow{M})$ iterations of the **while** loop in Algorithm 3. For each **while** iteration, a **foreach** loop visits the $(d-1)$-dimensional simplex faces that include node y for a total of $\mathcal{O}(P_s)$ iterations, where $P_s = \max_{y \in \mathcal{X}} |\{s \in \underline{\mathcal{S}}_d \mid y \in s\}|$. Each iteration of the **foreach** loop performs a constant number of operations. Thus, the number of operations to execute `ComputeUpdateSet` N times is $\mathcal{O}(N\overrightarrow{M}P_s))$. Assuming P_s is bounded as $N \to \infty$, the computational complexity of initializing the update node sets is $\mathcal{O}(N\overrightarrow{M})$.

Now we examine the complexity of executing the **while** loop of Algorithm 2. For each iteration of the **while** loop, a node x is accepted. A node is accepted only once, so there are N iterations in total. For each iteration, a **foreach** loop visits the $\mathcal{O}(\overleftarrow{M})$ unaccepted nodes in the dependent node

set of x. For each such dependent node y, each neighbor of x must be tested for membership in $\overrightarrow{\mathcal{Y}}(y)$ to determine the update simplices in Line 10 at a cost of $\mathcal{O}(P_n \log \overrightarrow{M})$, where $P_n = \max_{x \in \mathcal{X}} |\mathcal{N}(x)|$ and $\overrightarrow{\mathcal{Y}}(y)$ is implemented as a self-balancing binary search tree. Also, for each y, $\mathcal{O}(P_s)$ updates are performed at Line 11 and the binary min-heap implementation of $\mathcal{H}$ must be updated at a complexity of $\mathcal{O}(\log N)$. Thus, the number of operations in executing the **while** loop is $\mathcal{O}(N\overleftarrow{M}(P_n \log \overrightarrow{M} + P_s + \log N))$. Assuming P_n and P_s are bounded as $N \to \infty$, the complexity is $\mathcal{O}(N\overleftarrow{M} \log N)$, which is determined by the heap update operations.

The complexity of executing the **while** loop of Algorithm 2 dominates the complexity of the initialization, including that of the calls to the subroutine Algorithm 3, which we determined above to be $\mathcal{O}(N\overrightarrow{M})$ and the initialization of the heap which is $\mathcal{O}(N \log N)$. So, by Lemma 5.21, the overall asymptotic complexity of Algorithm 2 is

$$\mathcal{O}(N\overleftarrow{M} \log N) = \mathcal{O}(N(\hat{\Psi}\rho)^d \log N).$$

Single-pass algorithms for isotropic problems [55, 64] and limited anisotropic problems such as that in [58] or in Chapter 4 have a complexity of $\mathcal{O}(N \log N)$. The extra $(\hat{\Psi}\rho)^d$ factor in the complexity of MAOUM is due to the number of nodes in the update node set, which in MAOUM has been expanded beyond direct grid neighbors. In [60] a complexity of $\mathcal{O}(N\hat{\Upsilon}^{d-1} \log N)$ is derived for AFOUM, where $\hat{\Upsilon} = \max_{x \in \underline{\Omega}} \Upsilon(x)$. As shown in Remark 5.13, $\hat{\Psi} < \hat{\Upsilon}$. However, AFOUM's complexity has a reduced exponent of $d - 1$ because the update node set lies on a lower dimensional accepted front.

The complexity claim from [60] does not consider ρ, even though it plays an important role when the grid is highly nonuniform. If we assume ρ is bounded as $N \to \infty$, then MAOUM's complexity is $\mathcal{O}(N\hat{\Psi}^d \log N)$. However, the optimal allocation of grid nodes for Algorithm 2 may cause the grid to become more nonuniform as $N \to \infty$. In Section 5.5 we examine the relationship between the solution accuracy and the computational cost experimentally on several problems with a progression of increasingly-refined uniform and nonuniform grids.

For practical grid sizes, it is often the CPU time spent computing node value updates which dominates the CPU time of executing the entire algorithm, despite the fact that the computational complexity of the heap updates dominates in asymptotic analysis. Computing a node value may involve iteratively solving a nonlinear equation or a numerical optimization which can be quite computationally intensive. For this reason, in Section 5.5 we use the number of updates as our measure of computational cost.

5.4.4 Update Region

One of the drawbacks of MAOUM is that $\overrightarrow{\mathcal{Y}}(x)$ and $\overleftarrow{\mathcal{Y}}(x)$ must be stored for each x, which may require a considerable amount of memory. It turns out that $\overrightarrow{\mathcal{Y}}(x)$ ($\overleftarrow{\mathcal{Y}}(x)$, repectively) can be stored in memory as a region $\overrightarrow{\mathcal{R}}(x) \supset \overrightarrow{\mathcal{Y}}(x)$ ($\overleftarrow{\mathcal{R}}(x) \supset \overleftarrow{\mathcal{Y}}(x)$, respectively) instead of a set of nodes. For now, we just discuss determining $\overrightarrow{\mathcal{R}}(x)$, since the identical representations and computations can be used for $\overleftarrow{\mathcal{R}}(x)$. At the end of this section, we discuss one important consideration for $\overleftarrow{\mathcal{R}}(x)$.

A simple example of such a region is the ball $\overrightarrow{\mathcal{R}}(x) = \mathcal{B}(x, \hat{r}(x))$. Such a representation is easy to update in Algorithm 3 by replacing Line 4 with

$$\hat{r}(x) \leftarrow \max(\hat{r}(x), \|x - y\|).$$

Also, it can be stored using just one number $\hat{r}(x)$ and it is easy to test whether $\|x_i^s - y\| \leq \hat{r}(y)$ in Line 10 of Algorithm 2. A ball may form a reasonably tight bound of $\overrightarrow{\mathcal{Y}}(x)$ when the mesh is uniform, but it can be much too large in the case where the mesh is highly nonuniform, since for a given x the mesh may be coarse in some directions while being highly refined in others.

For this reason we use a polygonal representation of $\overrightarrow{\mathcal{R}}(x)$, for which we choose a set of direction vectors $\mathcal{A}_p$ such that $n = |\mathcal{A}_p|$ is the number of directions and also the maximum number of sides in the polygon, and $\|a_i\|_2 = 1$ for $a_i \in \mathcal{A}_p$ and $1 \leq i \leq n$. For example, we may use $n = 3^d - 1$ direction vectors by taking all permutations of negative ones, zeroes, and ones except the zero vector and normalizing. If $d = 2$, this choice of $\mathcal{A}_p$

would represent an octagon using 8 direction vectors:

$$\mathcal{A}_p = \{(-1/\sqrt{2}, -1/\sqrt{2}), (-1, 0), (-1/\sqrt{2}, 1/\sqrt{2}), (0, -1),$$
$$(0, 1), (1/\sqrt{2}, -1/\sqrt{2}), (1, 0), (1/\sqrt{2}, 1/\sqrt{2})\}.$$

Roughly speaking, for each $x \in \underline{\Omega}$ and for each direction a_i we compute and store the maximum distance $\eta_i(x)$ of any $y \in \overrightarrow{\mathcal{Y}}(x)$ along the direction a_i. In other words,

$$\eta_i(x) = \max_{y \in \overrightarrow{\mathcal{Y}}(x)} [(y - x) \cdot a_i],$$

for $1 \leq i \leq n$. It is simple to update $\eta_i(x)$ in Algorithm 3 by replacing Line 4 with

$$\eta_i(x) \leftarrow \max(\eta_i(x), (y - x) \cdot a_i),$$

for $1 \leq i \leq n$. We determine whether $x_i^s \in \overrightarrow{\mathcal{R}}(x)$ in Line 10 of Algorithm 2 by checking the constraints

$$(x_i^s - y) \cdot a_i \leq \eta_i(y),$$

for $1 \leq i \leq n$.

There is a trade-off between the amount of data stored to represent $\overrightarrow{\mathcal{R}}(x)$, and the number of nodes $|\overrightarrow{\mathcal{R}}(x)|$, which relates to the number of times `Update`(y, s) is called. The above representation with $n = 3^d - 1$ direction vectors can in many cases form a reasonable approximation of $\overrightarrow{\mathcal{Y}}(x)$. This representation has worked well in tests we have performed with highly nonuniform meshes, even though it is limited to representing a convex region.

For the most part, $\overleftarrow{\mathcal{R}}(x)$ can be computed and used in the same manner as $\overrightarrow{\mathcal{R}}(x)$, but there is one notable exception. In Line 9 of Algorithm 2, the set $\overleftarrow{\mathcal{R}}(x) \cap \mathcal{H}$ must be determined. We address this difficulty by storing the nodes in $\mathcal{H}$ in a 2^d-tree. It is fairly straightforward to search the 2^d-tree to determine which nodes are within a ball or polygon, for example. The ease with which the 2^d-tree can be searched is another important consideration in choosing a representation of $\overleftarrow{\mathcal{R}}(x)$ in our implementation.

5.5 Experiments

We present numerical experiments to test the convergence, computational cost, and accuracy of Algorithm 1 (p.35). We also demonstrate that MAOUM can be used to solve practical problems, such as a seismic imaging problem [60] and a robot navigation problem with obstacles and wind. The experiments indicate that MAOUM is particularly suited to problems in which the characteristics are highly-curved in some regions of Ω and straight or nearly straight elsewhere. For such problems it is more efficient to refine the grid only in the regions with curved characteristics.

For all of the experiments reported below $d = 2$, although Algorithm 2 can be used for any dimension. Let $x = (x_1, x_2)^T$ and $y = (y_1, y_2)^T$. We use a Maubach grid [43] for our simplicial grid. Examples of uniform Maubach grids are shown in Figure 5.6 and nonuniform Maubach grids in Figure 5.7. It is a semi-structured simplicial grid that can be adaptively refined and used in any dimension.

For the `Update` function we must solve (5.8), which involves finding the minimum over ζ of a convex function $\eta^s_{\underline{v}}(\zeta)$ for each s. We use the golden section optimization [37] when $d = 2$ and the projected subgradient method [61] when $d > 2$. These optimization methods are likely slower than more sophisticated alternatives that use gradient information but they are relatively simple to implement and can minimize all convex functions, even those for which the gradient is undefined in places.

5.5.1 Numerical Convergence Study

We demonstrate numerically that the output $\underline{v}$ of Algorithm 2 converges to the solution u of an anisotropic HJ PDE as the grid spacing goes to zero. For this experiment we use a series of increasingly fine uniform Maubach grids, such as those in Figure 5.6.

We use a homogeneous Hamiltonian (i.e. $H(x, q) = H(q)$) and $\mathcal{A}_f$ that is a non-grid aligned ellipse. For the purpose of implementation it is easiest to define $c(x, y) = c(y)$ from (5.9). Let $c(y) = \|By\|_2$, where B is the 2×2

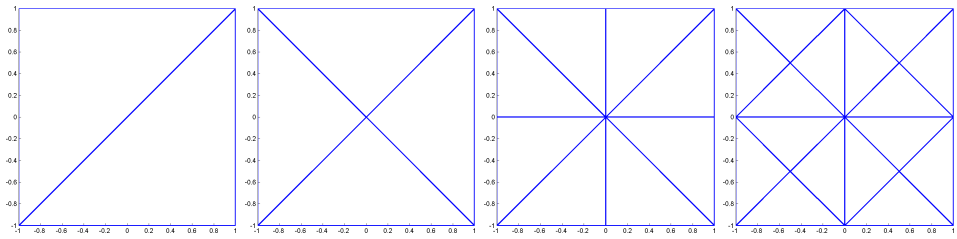


Figure 5.6: A sequence of uniformly-refined Maubach grids with 0, 1, 2, and 3 levels of refinement.

matrix

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \cos(\pi/6) & -\sin(\pi/6) \\ \sin(\pi/6) & \cos(\pi/6) \end{bmatrix}$$

The cost function c rotates y by $\pi/6$ counterclockwise around the origin and then scales it by 4 in the vertical axis before taking the Euclidean norm.

Level	N	M	h	Updates	e_∞	r_∞^h	e_1	r_1^h
10	1089	56.6	6.3e-2	32124	3.1e-2		2.9e-3	
12	4225	60.2	3.1e-2	134666	8.9e-3	1.8	1.2e-3	1.3
14	16641	62.1	1.6e-2	550726	3.8e-3	1.2	4.7e-4	1.3
16	66049	63.0	7.8e-3	2226532	1.8e-3	1.1	2.1e-4	1.2
18	263169	63.5	3.9e-3	8952636	8.2e-4	1.1	9.6e-5	1.1

Table 5.2: The problem has a homogeneous Hamiltonian and $\mathcal{A}_f$ that is a non-grid aligned ellipse with anisotropy $\Upsilon = 4$. The table shows grid spacing versus maximum errors and average errors of approximate solutions computed on a progression of uniform grids by MAOUM. *Level* is the level of grid refinement. M is the average over $x \in \underline{\Omega}$ of the number of nodes in the update node set. h is the distance between neighboring grid nodes in the x_1 and x_2 directions. *Updates* is the number of times `Update(`y, s`)` is called. e_∞ is the $\mathcal{L}_\infty$-error in the approximation $\underline{u}$ to the true solution u of (1.8). e_1 is the $\mathcal{L}_1$-error. The error convergence rates r_∞^h and r_1^h are computed with respect to the grid spacing h.

Let $\mathcal{D} = \{x \in \mathbb{R}^2 \mid c(x) \leq 0.4\}$. The domain for this problem is $\Omega = [-1,1]^2 \setminus \mathcal{D}$ and the boundary is $\partial\Omega = \partial\mathcal{D}$. The boundary conditions are given as $g(x) = c(x)$ for $x \in \mathcal{D}$. Notice that the boundary conditions are

defined throughout $\mathcal{D}$ and thus extend beyond $\partial\Omega$. We define the boundary to be the ellipse $\mathcal{D}$ to exclude errors caused by poor approximation of the solution u near the origin, where u is nondifferentiable. It is not necessary to give boundary conditions on the external boundary since all characteristics flow out of this boundary. The grid resolutions and corresponding errors are listed in Table 5.2.

5.5.2 Nonuniform Grid

To determine what benefit Algorithm 2 gains from refining a grid intelligently, we use an anisotropic HJ PDE which has a solution with kinks where the gradient is undefined. We run Algorithm 2 on a series of increasingly-fine uniform grids and a series of increasingly nonuniform grids, where newly added nodes are concentrated around the parts of the solution where the gradient is undefined. For comparison, we also run AFOUM on both the uniform and nonuniform grid series. For all four combinations of algorithm and grid series we plot the solution accuracy vs computational cost to see if Algorithm 2 performs better on a well-chosen nonuniform grid.

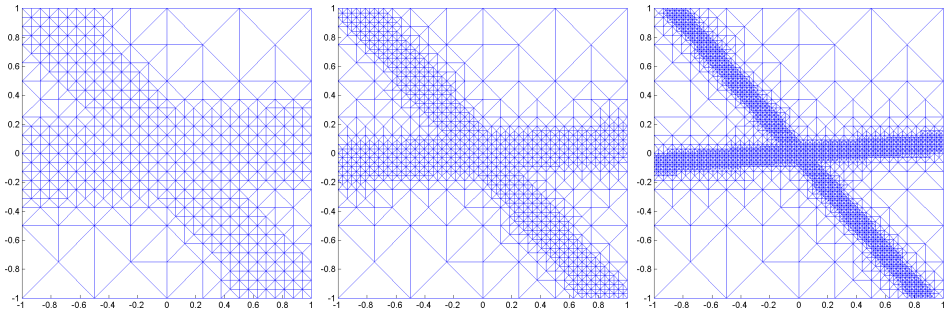


Figure 5.7: The sequence of nonuniformly-refined Maubach grids with 10, 12, and 14 levels of refinement used for the problem with homogeneous Hamiltonian and rectangular $\mathcal{A}_f$ that is not aligned with the grid lines.

We use a homogeneous Hamiltonian (i.e. $H(x,q) = H(q)$) and a rectangular $\mathcal{A}_f$ that is not aligned with the grid lines. Let $c(y) = \|By\|_\infty$, where

B is the 2×2 matrix

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \cos(\pi/8) & -\sin(\pi/8) \\ \sin(\pi/8) & \cos(\pi/8) \end{bmatrix}$$

The cost function c rotates y by $\pi/8$ counterclockwise around the origin and then scales it by 2 in the vertical axis before taking the maximum norm. The domain for this problem is $\Omega = [-1, 1]^2 \setminus O$ and the boundary is $\partial\Omega = O$, where O is the origin. The boundary conditions are given as $g(O) = 0$.

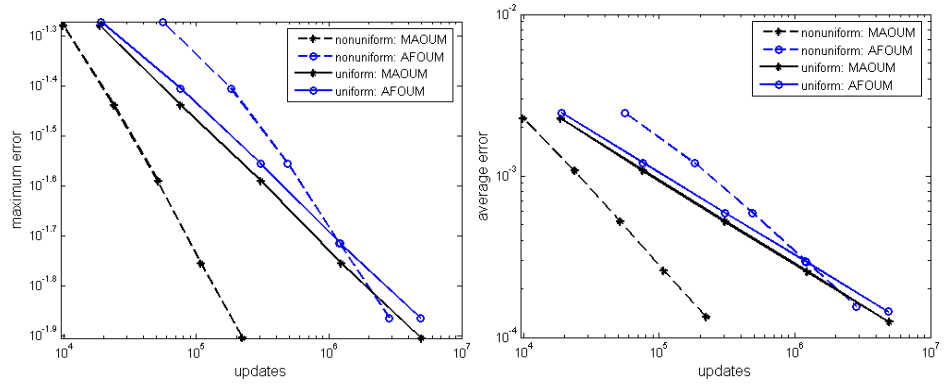


Figure 5.8: Error versus number of updates for problem with homogeneous Hamiltonian and $\mathcal{A}_f$ that is a non-grid aligned rectangle with length that is twice its width. The values plotted are from Table 5.3.

Part of the nonuniform grid series used is shown in Figure 5.7. The grids are refined within distance $2\check{h}\Upsilon$ of the two lines

$$x_2 = \frac{\sin(-\pi/8) + \frac{1}{2}\cos(-\pi/8)}{\cos(-\pi/8) - \frac{1}{2}\sin(-\pi/8)} x_1$$

and

$$x_2 = \frac{\sin(-\pi/8) - \frac{1}{2}\cos(-\pi/8)}{\cos(-\pi/8) + \frac{1}{2}\sin(-\pi/8)} x_1,$$

where $\check{h}$ is the minimum grid edge length after refinement is complete.

The results for all four combinations of algorithm and grid series are compared in Table 5.3 and Figure 5.8. To properly interpret the relative

			MAOUM			AFOUM		
Grid	Level	N	Updates	e_∞	r_∞^U	Updates	e_∞	r_∞^U
Nonuni	10	621	10126	5.3e-2		73560	5.3e-2	
	12	1443	24482	3.6e-2	0.41	243620	3.9e-2	0.26
	14	3051	53264	2.6e-2	0.45	657620	2.8e-2	0.35
	16	6305	111972	1.8e-2	0.51	1616153	1.9e-2	0.41
	18	12911	231618	1.2e-2	0.47	3809008	1.4e-2	0.40
Uni	10	1089	18744	5.3e-2		24754	5.3e-2	
	12	4225	76346	3.6e-2	0.26	97974	3.9e-2	0.22
	14	16641	308226	2.6e-2	0.25	390716	2.8e-2	0.25
	16	66049	1238754	1.8e-2	0.27	1561944	1.9e-2	0.27
	18	263169	4967030	1.2e-2	0.25	6248199	1.4e-2	0.25
				e_1	r_1^U		e_1	r_1^U
Nonuni	10	621	10126	2.3e-3		73560	2.4e-3	
	12	1443	24482	1.1e-3	0.84	243620	1.2e-3	0.59
	14	3051	53264	5.2e-4	0.93	657620	5.8e-4	0.71
	16	6305	111972	2.6e-4	0.95	1616153	2.9e-4	0.76
	18	12911	231618	1.3e-4	0.90	3809008	1.5e-4	0.75
Uni	10	1089	18744	2.3e-3		24754	2.4e-3	
	12	4225	76346	1.1e-3	0.53	97974	1.2e-3	0.52
	14	16641	308226	5.2e-4	0.52	390716	5.8e-4	0.51
	16	66049	1238754	2.5e-4	0.51	1561944	2.9e-4	0.51
	18	263169	4967030	1.2e-4	0.51	6248199	1.4e-4	0.51

Table 5.3: The problem has a homogeneous Hamiltonian and rectangular $\mathcal{A}_f$ that is not aligned with the grid lines. The table shows the number of updates versus maximum errors and average errors of approximate solutions computed on a progression of nonuniform and uniform grids by MAOUM and AFOUM. *Nonuni* indicates the use of a nonuniform grid, while *Uni* indicates a uniform grid. *Level* is the maximum level of grid refinement. *Updates* is the number of times `Update(`y, s`)` is called. e_∞ is the $\mathcal{L}_\infty$-error in the approximation $\underline{u}$ to the true solution u of (1.8). e_1 is the $\mathcal{L}_1$-error. The error convergence rates r_∞^U and r_1^U are computed with respect to Updates rather than h as in Table 5.2.

performance of MAOUM and AFOUM in Figure 5.8, one needs to understand the extra cost involved in the initialization of the update sets in Algorithm 3 of MAOUM. Between 40 and 50 percent of MAOUM's total run time was spent in Algorithm 3. If we consider the ratio of total run time

(including initialization) to number of updates, MAOUM took between 90 and 240 percent of the time per update of AFOUM.

After factoring in the initialization time for MAOUM, on the highly-nonuniform grids it runs nearly an order of magnitude faster than AFOUM resulting in similar approximation error. On uniform grids MAOUM runs slightly slower than AFOUM (but of the same order) resulting in similar error. MAOUM on highly-nonuniform grids runs nearly an order of magnitude faster than either method on uniform grids resulting in similar error. Note that this problem has been chosen to highlight the advantage of MAOUM in solving problems in which the characteristics are highly-curved in some regions of Ω and straight or nearly straight elsewhere.

5.5.3 Seismic Imaging

We run MAOUM on the seismic imaging example from the lower-right of Figure 6 in [60]. The domain for this problem is $\Omega = [-0.5, 0.5]^2 \setminus O$ and the boundary is $\partial\Omega = O$, where O is the origin. The boundary conditions are given as $g(O) = 0$. A wave propagates from the origin and passes through Ω, which is split into four layers by three vertically-shifted sinusoidal curves. The set $\mathcal{A}_f(x)$ is an ellipse with the longer axis aligned with the tangent of the sinusoidal curves at x_2. The dimensions of the elliptical $\mathcal{A}_f(x)$ are constant within a layer. The problem is to compute the first arrival time for the seismic wave. More details are specified in [60].

We test MAOUM on this problem using uniform Maubach grids from levels 13 to 18. The computed solution for levels 13 and 18 grids are shown in Figure 5.9. Preliminary experiments indicate that refining the grid along the sinusoidal layer boundaries does not improve accuracy significantly. We believe this is because the characteristics are curved to roughly the same degree throughout Ω. Localized grid refinement is most beneficial when characteristics are highly-curved in some parts of Ω and nearly straight elsewhere.

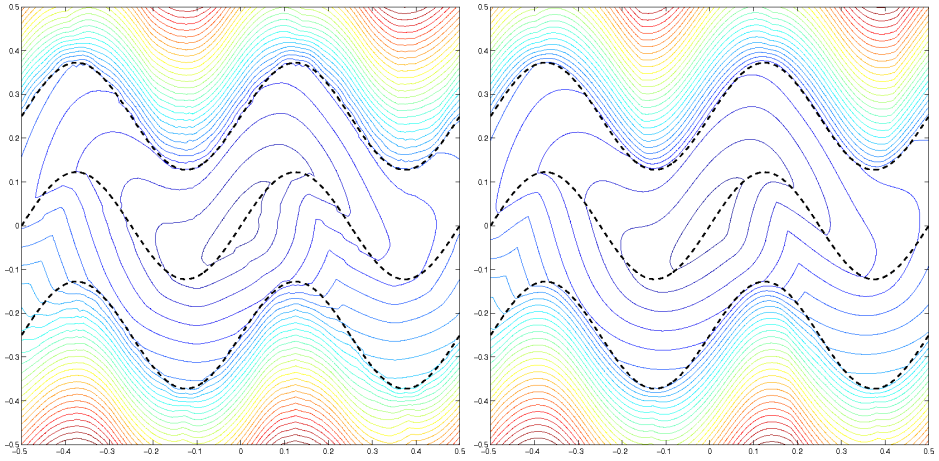


Figure 5.9: Contours of first-arrival times of a seismic wave computed using a uniform Maubach grid of level 13 with 8321 nodes (on left) and level 18 with 263169 nodes (on right).

5.5.4 Robot Navigation with Wind and Obstacles

An optimal time-to-reach problem with obstacles is a natural candidate for exploiting localized grid refinement. An optimal trajectory that must go around an obstacle to achieve the goal must closely track the obstacle boundary for some portion. Refining the grid to better resolve these obstacle boundaries should allow for a more accurate solution in portions of the domain that do not have an obstacle-free optimal path to the goal.

Although this example is not physically realistic, it does use data of suitable complexity for realistic scenarios and demonstrates MAOUM on a spatially-inhomogeneous anisotropic problem. The objective is for a robot to navigate from any location in the domain to a goal in optimal time. To make the task difficult the robot must avoid obstacles on its way to the goal and there is an inhomogeneous but static wind that pushes the robot. The goal is $x^* = (80.75, 46.25)^T$, $g(x^*) = 0$, and $\partial\Omega = \{x^*\}$. The domain is $\Omega = [72, 112] \times [17.5, 57.5] \setminus \partial\Omega$. The domain dimensions were chosen arbitrarily to obtain a sufficiently varied region of wind data (see below). The robot is circular with a radius of 1.1429. The obstacles are a

set of points obtained from a laser range finder map downloaded with the Saphira robot control system software. The same point data was used for an isotropic path planning problem in [1], but we map the data from the square domain $[-4000, -500] \times [-3500, 0]$ to Ω. The point obstacles are shown in Figure 5.10 (a). To prevent the robot from attempting to plan a path through obstacles, it moves at a very slow speed of $f(x, a) = 0.5$ for any $a \in \mathcal{A}$ and any $x \in \mathcal{C}$, where $\mathcal{C}$ is the set of states such that the robot is in collision with a point obstacle. The collision set $\mathcal{C}$ is depicted in solid black in Figure 5.10. The wind velocity is represented by a vector field shown in (b). We obtained the vector field from the `wind` arrays in Matlab.[1] We use bilinear interpolation between samples to obtain the wind vector function $\vec{f_w} : \Omega \to R^2$.

In the absence of wind, the robot moves with speed $f_r = 75.0$, resulting in an isotropic speed profile $\mathcal{A}_{f_r} = \{y \mid \|y\| \leq f_r\}$. Although not physically realistic, we shift the isotropic speed profile by wind velocity, so the anisotropic speed profile is

$$\mathcal{A}_f(x) = \{y \mid \|y - \vec{f_w}(x)\| \leq f_r\}.$$

Note that $f_r = 75.0 > \max_{x \in \Omega} \|\vec{f_w}(x)\|$, so $\mathcal{A}_f(x)$ contains the origin in its interior. In order to determine the cost $c(x, y)$ we find the intersection of vector y multiplied by a scalar b with $\partial \mathcal{A}_f(x)$ by solving the quadratic $\|by - \vec{f_w}(x)\|^2 = f_r^2$ for $b \in \mathbb{R}^+$. Then we have $f(x, y/\|y\|) = b\|y\|$ and

$$\begin{aligned} c(x, y) &= \frac{\|y\|}{f(x, y/\|y\|)} \\ &= \frac{1}{b} = \frac{\|y\|^2}{y \cdot \vec{f_w}(x) + \sqrt{(y \cdot \vec{f_w}(x))^2 - \|y\|^2(\|\vec{f_w}(x)\|^2 - f_r^2)}}. \end{aligned}$$

We note that an HJ PDE with the same isotropic control and advection component was derived and solved in [59].

[1] To load the wind data into Matlab type `load wind;`. The data is a 3D vector field, so we used only the 6^{th} page of the data and ignored any wind velocity component in the 3^{rd} dimension. In other words, we used `u(:,:,6)` and `v(:,:,6)` for the arrays of wind vector components and `x(:,:,6)` and `y(:,:,6)` for the arrays of spatial coordinates.

To compute an optimal trajectory from any initial location to x^*, we solve the characteristic ordinary differential equation (ODE)

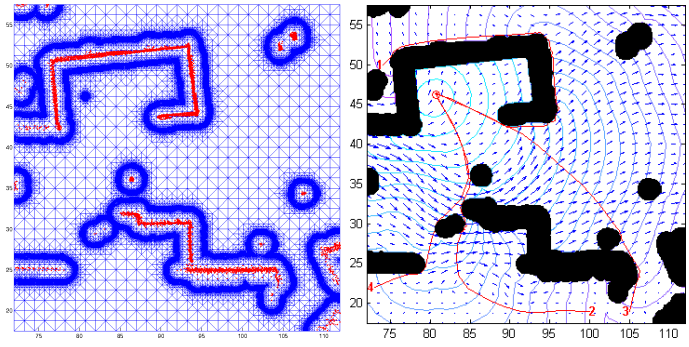
$$\frac{dx}{dt} = \vec{f}_w(x) - f_r \frac{D\underline{u}(x)}{\|D\underline{u}(x)\|} = f(x, \hat{a}(x))\hat{a}(x), \tag{5.36}$$

where

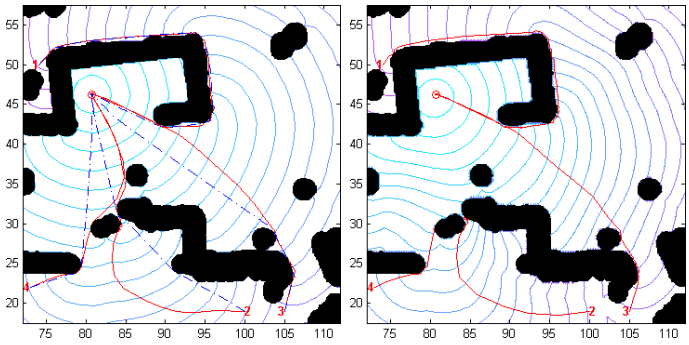
$$\hat{a}(x) \in \operatorname*{argmax}_{a \in \mathcal{A}} [(-D\underline{u}(x) \cdot a) f(x, a)].$$

Note that this is not gradient descent, because the gradient and the optimal characteristic will not generally align in anisotropic problems. To solve the ODE we used the function `ode23` in Matlab. We determine the derivative in (5.36) for any x, by first computing $D\underline{u}(x)$ as the gradient of the linearly interpolated $\underline{u}$ of the grid simplex containing x.

We use a Maubach grid that is additionally refined within a distance of $\check{h}\Upsilon$ of $\mathcal{C}$ and the goal x^*. The grid is uniformly refined to level 10 and then refined a further 8 levels near $\mathcal{C}$ and x^*. The resulting grid is shown in Figure 5.10 (a) and has 38728 nodes. We compute the time-to-reach function $\underline{u}$ for the anisotropic problem (i.e. with the wind) and the isotropic problem (i.e. without the wind) on the nonuniformly refined grid. Solution contours are shown in Figure 5.10 (b) and (c), respectively. Optimal trajectories for the anisotropic problem are shown in (b). Notice how trajectories 2 and 4 cut quickly across regions where the wind is strong and against the direction towards the goal. Contrast these trajectories with the straight line optimal trajectories for the isotropic problem in (c). We also solve the anistropic problem on a uniform level 15 Maubach grid of 33025 nodes. Solution contours and optimal trajectories are shown in (d). Although the solution contours are smoother away from $\mathcal{C}$ in this uniform case, the ODE computation gets stuck near $\mathcal{C}$ for trajectories 2 and 4, likely due to insufficient grid refinement and poor solution quality near $\partial\mathcal{C}$.



(a) (b)



(c) (d)

Figure 5.10: The problem of optimally-navigating a robot to a goal through wind and obstacles. The top-left shows the laser-rangefinder data of the obstacles and the grid refined in a band around the collision set $\mathcal{C}$ and the goal x^*. The other three figures show $\mathcal{C}$ in solid black. The top-right includes the wind vector field, the contours of the computed time-to-reach function, and four optimal trajectories from different starting locations to the goal. The lower-left compares the optimal trajectories computed with the wind and without. The contours are of the isotropic (i.e. without wind) time-to-reach function. The solid lines are trajectories with the wind and the dash-dotted lines are trajectories without the wind. The lower-right shows contours of the time-to-reach function and trajectories, computed using a level 15 uniform Maubach grid with roughly the same number of nodes. Note that for trajectories 2 and 4 the characteristic ODE computation of the optimal trajectories gets stuck near the boundary of $\mathcal{C}$ which is insufficiently resolved.

Chapter 6

Conclusion

We extended Dijkstra-like OUMs for solving static HJ PDEs in two important ways. For both extensions, we used a common framework (Chapter 3) to prove convergence of the approximate solution to the viscosity solution of the HJ PDE and to prove that the algorithm solves the system of discretized equations in a single-pass through the nodes of the grid. The consistency, monotonicity, and stability of the discretization were used to prove convergence (Section 3.2) following the technique of [8]. The causality of the discretization was used to prove a Dijkstra-like algorithm (Section 3.3) can solve the discrete equations in a single-pass in nondecreasing node value following a similar argument to those in [55, 64].

The first extension (Chapter 4) showed that the Dijkstra-like FMM can be used to solve a class of axis-aligned HJ PDEs on an orthogonal grid. This class (Section 4.2) is not limited to differentiable Hamiltonians as is the case for Osher's criterion on the applicability of FMM [47, 62]. We showed that a first-order upwind finite difference Eulerian discretization (Section 4.3) of the HJ PDE satisfies the required consistency, monotonicity, stability, and causality, as well as has properties that make it simple and efficient to update a node value using the discretized equation. These properties include the fact that $\underline{H}(\mu)$ is nondecreasing on μ and there is a unique solution $\mu = \tilde{\mu}$ to $\underline{H}(\mu) = 0$, where $\underline{H}$ is the numerical Hamiltonian and μ is a stand-in for the current node value of interest. There are also a number of ways that

structure in $\underline{H}$ can be exploited (Section 4.4) to remove from consideration neighboring nodes and simplices when solving $\underline{H}(\mu) = 0$. Our method can be used to solve some challenging problems (Section 4.6), including computing coordinated multi-robot optimal trajectories.

The second extension (Chapter 5) developed a Dijkstra-like method called MAOUM that can be used to solve general convex static HJ PDEs. We presented three simple criteria for the causality of the discretization of a static convex HJ PDE. MAOUM uses an initial pass through the nodes to compute and store extended stencils that ensure the discretization is causal. A second pass through the nodes computes the solution to the discretized equations in a Dijkstra-like fashion. This method solves the same class of problems as the original OUM (which we call AFOUM) described in [59, 60]. One advantage of MAOUM over AFOUM is that it uses local grid spacings to compute the stencil, making it more suitable for use on a grid with highly nonuniform grid spacing. A second advantage is that the discrete equations are causal in the sense of Property 3.9, and so once the stencils have been built, a simple Dijkstra-like method can be used to solve the discrete equations in nondecreasing value order. A third advantage is that the analysis of δ-causality in Section 5.3 allows modification of the method to a Dial-like algorithm with buckets of width δ. One last advantage is that for solving multiple problems with different boundary conditions g but the same Hamiltonian H the generated stencils can be computed once and reused for each problem. Disadvantages of the method include the extra pass required to compute the stencils and the necessity of storing the stencils for use in the second pass. Despite these disadvantages, in our experiments MAOUM performed similarly in terms of accuracy/computation cost to AFOUM for problems on uniform grids, and it performed considerably better for some problems which had highly curved characteristics and a more refined grid in only certain regions of the domain. MAOUM was applied to solve several problems, such as finding optimal paths for a robot to reach a goal while fighting a strong wind and avoiding obstacles.

6.1 Future Work

Algorithm 3 does not generate the smallest possible stencil $\overrightarrow{\mathcal{Y}}(x)$ such that $\mathcal{S}(x)$ satisfies DC and AAB for x. The last plot in Figure 5.5 shows a $\overrightarrow{\mathcal{Y}}(x)$ that is not minimal. Consequently, Algorithm 3 could be improved. Ideally, it would generate $\overrightarrow{\mathcal{Y}}(x)$ that is minimal without substantially increasing the computational cost.

We intend to develop an error-control adaptive-grid version of MAOUM. When a part of the approximate solution has just been computed we can measure the approximation error and test whether it falls within a pre-specified tolerance. If it does, that part of the solution can be accepted. If not, the grid can be refined in that local region, the stencils nearby can be recomputed, and the Dijkstra-like algorithm can be backed-up to recompute parts of the solution that may have changed due to the modified discretized equations. In this way, grid refinement would be used to control error as the solution is being constructed.

There is the question of how we might exploit heuristics and parallelism to compute the solution or parts of it more efficiently. If after computing the approximate solution, one plots a dependency graph that shows how grid nodes depend on others for their solution value, it is usually the case that any particular node is independent of most other nodes for its value. Accordingly, if we were to recompute the solution for that node, we could ignore all nodes of which it is independent. If we are only interested in computing the solution near a particular state or near a characteristic that passes through that state (as is often the case in optimal control problems), we may be able to devise a heuristic that helps us decide in advance which solution values might be relevant. This idea may allow us to develop a single-pass focused computation of the solution in the spirit of A* search for discrete graphs (see [54, page 97]).

Furthermore, the fact that node values are commonly independent of each other, especially when those values are very similar, makes the possibility of computing node values in parallel intriguing. One way to exploit this inherent potential for parallel computation may be to use a Dial-like

method [64] to solve a δ-causal discretization.

Bibliography

[1] K. Alton and I. Mitchell. Optimal path planning under different norms in continuous state spaces. In *Proceedings of the International Conference on Robotics and Automation*, pages 866–872, 2006.

[2] K. Alton and I. Mitchell. Efficient dynamic programming for optimal multi-location robot rendezvous. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2794–2799, 2008.

[3] K. Alton and I. M. Mitchell. Fast marching methods for a class of anisotropic stationary Hamilton-Jacobi equations. Technical Report TR-2006-27, Department of Computer Science, University of British Columbia, 2007.

[4] K. Alton and I. M. Mitchell. Fast marching methods for stationary Hamilton-Jacobi equations with axis-aligned anisotropy. *SIAM J. Numer. Anal.*, 43:363–385, 2008.

[5] K. Alton and I. M. Mitchell. An ordered upwind method with precomputed stencil and monotone node acceptance for solving static convex Hamilton-Jacobi equations, 2010. In revision for *J. Sci. Comput.*.

[6] S. Bak, J. McLaughlin, and D. Renzi. Some improvements for the fast sweeping method, 2010. Accepted to *SIAM J. Sci. Comput.*.

[7] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhauser, Boston and Basel and Berlin, 1997.

[8] G. Barles and P. E. Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptot. Anal.*, 4:271–283, 1991.

[9] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[10] D. P. Bertsekas. *Dynamic Programming and Optimal Control: Second Edition.* Athena Scientific, Belmont, Massachusetts, 2000.

[11] F. Bornemann and C. Rasch. Finite-element discretization of static Hamilton-Jacobi equations based on a local variational principle. *Comput. Vis. Sci.*, 9:57–69, 2006.

[12] M. Boue and P. Dupuis. Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control. *SIAM J. Numer. Anal.*, 36(3):667–695, 1999.

[13] A. R. Bruss. The Eikonal equation: Some results applicable to computer vision. *J. Math. Phys.*, 23:890–896, 1982.

[14] T. C. Cecil, S. J. Osher, and J. Qian. Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. *J. Comput. Phys.*, 213(2):458–473, Apr. 2006.

[15] Y. Chen and B. Cockburn. An adaptive high-order discontinuous Galerkin method with error control for the Hamilton-Jacobi equations. Part I: The one-dimensional steady state case. *J. Comput. Phys.*, 226 (1):1027–1058, September 2007.

[16] Y. Cheng and C.-W. Shu. A discontinuous Galerkin finite element method for directly solving the Hamilton-Jacobi equations. *J. Comput. Phys.*, 223:398–415, Apr. 2007.

[17] M. G. Crandall and P. L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *T. Am. Math. Soc.*, 277(1):1–42, May 1983.

[18] M. G. Crandall and P. L. Lions. Two approximations of solutions of Hamilton-Jacobi equations. *Math. Comput.*, 43(167):1–19, July 1984.

[19] M. G. Crandall, H. Ishii, and P. Lions. User's guide to viscosity solutions of second order partial differential equations. *Bull. Amer. Math. Soc.*, 27(1):1–67, 1992.

[20] E. Cristiani. A fast marching method for Hamilton-Jacobi equations modeling monotone front propagations. *J. Sci. Comput.*, 39(2):189–205, 2009.

[21] E. Cristiani and M. Falcone. Fast semi-Lagrangian schemes for the Eikonal equation and applications. *SIAM J. Numer. Anal.*, 45(5):1979–2011, 2007.

[22] P.-E. Danielsson. Euclidean distance mapping. *Comput. Graphics Image Process.*, 14(3):227–248, Nov. 1980.

[23] R. B. Dial. Algorithm 360: shortest-path forest with topological ordering. *Commun. ACM*, 12:632–633, 1969.

[24] E. W. Dijkstra. A note on two problems in connection with graphs. *Numer. Math.*, 1(1):269–271, Dec. 1959.

[25] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, Providence, Rhode Island, 1991.

[26] M. Falcone and R. Ferretti. Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations. *Numer. Math.*, 67:315–344, 1994.

[27] M. Falcone and R. Ferretti. Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods. *J. Comput. Phys.*, 175:559–575, 2002.

[28] S. Fomel. On anelliptic approximations for qP velocities in VTI media. *Geophysical Prospecting*, 52:247–259, April 2004.

[29] S. Fomel, S. Luo, and H. Zhao. Fast sweeping method for the factored Eikonal equation. *J. Comput. Phys.*, 228:6440–6455, September 2009.

[30] P. A. Gremaud and C. M. Kuster. Computational study of fast methods for the Eikonal equation. *SIAM J. Sci. Comput.*, 27:1803–1816, 2006.

[31] D. Halliday, R. Resnick, and J. Walker. *Fundamentals of Physics*. John Wiley and Sons, Inc., USA, 1997.

[32] B. K. P. Horn and M. J. Brooks, editors. *Shape from Shading*. The MIT Press, Cambridge, Massachusetts, 1989.

[33] C. Hu and C.-W. Shu. A discontinuous Galerkin finite element method for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21(2):666–690, 1999.

[34] S.-R. Hysing and S. Turek. The Eikonal equation: Numerical efficiency vs. algorithmic complexity on quadrilateral grids. In *Proceedings of Algoritmy 2005*, pages 22–31, 2005.

[35] G.-S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21(6):2126–2143, 2000.

[36] C. Y. Kao, S. Osher, and Y. Tsai. Fast sweeping methods for static Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 42(6):2612–2632, 2004-2005.

[37] J. Kiefer. Sequential minimax search for a maximum. *P. Am. Math. Soc.*, 4:502, 1953.

[38] S. Kim and D. Folie. The group marching method: An O(n) level set Eikonal solver. *SEG Technical Program Expanded Abstracts*, 70:2297–2300, 2000.

[39] R. Kimmel and J. A. Sethian. Fast marching methods on triangulated domains. *Proc. Natl. Acad. Sci.*, 95:8341–8435, 1998.

[40] R. Kimmel and J. A. Sethian. Optimal algorithm for shape from shading and path planning. *J. Math. Imaging Vision*, 14(3):237–244, May 2001.

[41] P. D. Lax and R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Comm. Pure Appl. Math.*, 9:267–293, 1956.

[42] F. Lia, C.-W. Shu, Y.-T. Zhang, and H. Zhao. A second order discontinuous Galerkin fast sweeping method for Eikonal equations. *J. Comput. Phys.*, 227:8191–8208, Sept. 2008.

[43] J. M. Maubach. Local bisection refinement for n-simplicial grids generated by reflection. *J. Sci. Comput.*, 16(1):210–227, January 1995.

[44] I. M. Mitchell and S. Sastry. Continuous path planning with multiple constraints. Technical Report UCB/ERL M03/34, UC Berkeley Engineering Research Laboratory, August 2003.

[45] A. M. Oberman. Convergent difference schemes for degenerate elliptic and parabolic equations: Hamilton-Jacobi equations and free boundary problems. *SIAM J. Numer. Anal.*, 44(2):879–895, 2006.

[46] S. Osher. A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations. *SIAM J. Math. Anal.*, 24(5):1145–1152, Sept. 1993.

[47] S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results. *J. Comput. Phys.*, 169(2):463–502, May 2001.

[48] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulation. *J. Comput. Phys.*, 79:12–49, 1988.

[49] L. C. Polymenakos, D. P. Bertsekas, and J. N. Tsitsiklis. Implementation of efficient algorithms for globally optimal trajectories. *IEEE Trans. Automat. Control*, 43:278–283, 1998.

[50] J. Qian, Y. Zhang, and H. Zhao. A fast sweeping method for static convex Hamilton-Jacobi equations. *J. Sci. Comput.*, 31(1-2):237–271, May 2007.

[51] J. Qian, Y. Zhang, and H. Zhao. Fast sweeping methods for Eikonal equations on triangulated meshes. *SIAM J. Numer. Anal.*, 45:83–107, 2007.

[52] C. Rasch and T. Satzger. Remarks on the O(n) implementation of the fast marching method. *IMA J. Numer. Anal.*, 29:806–813, 2009.

[53] E. Rouy and A. Tourin. A viscosity solution approach to shape-from-shading. *SIAM J. Numer. Anal.*, 29(3):867–884, 1992.

[54] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentic Hall, Upper Saddle River, New Jersey, 1995.

[55] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci.*, 93:1591–1595, 1996.

[56] J. A. Sethian. *Level Set Methods*. Cambridge University Press, Cambridge, UK, 1996.

[57] J. A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999.

[58] J. A. Sethian and A. Vladimirsky. Fast methods for Eikonal and related Hamilton-Jacobi equations on unstructured meshes. *Proc. Natl. Acad. Sci.*, 97(11):5699–5703, May 2000.

[59] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations. *Proc. Natl. Acad. Sci.*, 98(20):11069–11074, 2001.

[60] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. *SIAM J. Numer. Anal.*, 41(1):323–363, 2003.

[61] N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag, New York, USA, 1985.

[62] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 41(2):673–694, 2003.

[63] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. In *Proceedings of the 33rd Conference on Decision and Control*, pages 1368–1373, 1994.

[64] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. Automat. Control*, 40(9):1528–1538, 1995.

[65] A. Vladimirsky. Label-setting methods for multimode stochastic shortest path problems on graphs. *Math. Oper. Res.*, 33(4):821–838, November 2008.

[66] T. Xiong, M. Zhang, Y.-T. Zhang, and C.-W. Shu. Fifth order fast sweeping WENO scheme for static Hamilton-Jacobi equations with accurate boundary treatment, 2009. Submitted to *J. Sci. Comput.*.

[67] L. Yatziv, A. Bartesaghi, and G. Sapiro. O(n) implementation of the fast marching method. *J. Comput. Phys.*, 212(2):393–399, March 2006.

[68] Y.-T. Zhang, H.-K. Zhao, and J. Qian. High order fast sweeping methods for static Hamilton-Jacobi equations. *J. Sci. Comput.*, 29(1):25–56, Oct. 2006.

[69] Y.-T. Zhang, S. Chen, F. Li, H. Zhao, and C.-W. Shu. Uniformly accurate discontinuous Galerkin fast sweeping methods for Eikonal equations, 2009. Submitted to *SIAM J. Sci. Comput.*.

[70] H. Zhao. A fast sweeping method for Eikonal equations. *Math. Comp.*, 74(250):603–627, May 2004.