Optimal Direction-Dependent Path Planning for Autonomous Vehicles

by

Alexander Shum

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2014

© Alexander Shum 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The focus of this thesis is optimal path planning. The path planning problem is posed as an optimal control problem, for which the viscosity solution to the static Hamilton-Jacobi-Bellman (HJB) equation is used to determine the optimal path. The Ordered Upwind Method (OUM) has been previously used to numerically approximate the viscosity solution of the static HJB equation for direction-dependent weights.

The contributions of this thesis include an analytical bound on the convergence rate of the OUM for the boundary value problem to the viscosity solution of the HJB equation. The convergence result provided in this thesis is to our knowledge the tightest existing bound on the convergence order of OUM solutions to the viscosity solution of the static HJB equation. Only convergence without any guarantee of rate has been previously shown.

Navigation functions are often used to provide controls to robots. These functions can suffer from local minima that are not also global minima, which correspond to the inability to find a path at those minima. Provided the weight function is positive, the viscosity solution to the static HJB equation cannot have local minima. Though this has been discussed in literature, a proof has not yet appeared. The solution of the HJB equation is shown in this work to have no local minima that is not also global. A path can be found using this method.

Though finding the shortest path is often considered in optimal path planning, safe and energy efficient paths are required for rover path planning. Reducing instability risk based on tip-over axes and maximizing solar exposure are important to consider in achieving these goals. In addition to obstacle avoidance, soil risk and path length on terrain are considered. In particular, the tip-over instability risk is a direction-dependent criteria, for which accurate approximate solutions to the static HJB equation cannot be found using the simpler Fast Marching Method.

An extension of the OUM to include a bi-directional search for the source-point path planning problem is also presented. The solution is found on a smaller region of the environment, containing the optimal path. Savings in computational time are observed.

A comparison is made in the path planning problem in both timing and performance between a genetic algorithm rover path planner and OUM. A comparison in timing and number of updates required is made between OUM and several other algorithms that approximate the same static HJB equation. Finally, the OUM algorithm solving the boundary value problem is shown to converge numerically with the rate of the proven theoretical bound.

Acknowledgements

I would like to first and foremost acknowledge my research supervisors Professor Kirsten Morris and Professor Amir Khajepour. Their guidance, encouragement and experience from both academic and non-academic perspectives have made this work possible. Thank you both. I would like to also extend my gratitude to the members of my thesis committee, Professor Dong Eui Chang, Professor Christopher Nielsen, and Professor Brian Ingalls for taking the time to review and improve this work. I would like to especially thank Professor Ian Mitchell for serving as my external committee member.

To my family who has supported and encouraged me, words can not express my gratitude. You've been there to share in my successes, and to pick up the pieces when I was not at my best. My work would not have been possible without you.

To my dearest V.T, you've given me new perspectives on life and helped me see the world outside academia and have genuinely made me a better person. Thank you.

To my friends in Applied Math at Waterloo who have helped to provide insight to my research problem, tackle coding and provided welcome distractions, thank you. In particular, I'd like to thank Michael Dunphy, Dhanaraja Kasinathan and Arman Tavakoli who treated my problem as if it was their own. To Clinton, Kexue, Wilten and Vivek, thanks for helping me to find a balance between my studies and everything else. To my research group: Amenda, Rasha, Amir, Sepideh and Tawsif, your input has been invaluable. To Ingle, Danny, Liz, Kitty, Steph, Brandon, Colin, Mercy and Mary, your encouragement and support away from Waterloo has helped me tremendously. Thank you.

To my colleagues at the Centre for Teaching Excellence, thank you for giving me the opportunity to discover and further my passion for teaching. Being exposed to teaching styles outside of the traditional mathematics classroom has been incredibly worthwhile, and has no doubt helped me to become a better teacher.

To the UW Mechatronics class of 2016: having the opportunity to teach your class (twice!) has been one of the greatest highlights during my time here. I wish you all the best in your studies and future careers.

Dedication

To Mom, Dad and Andrea

Table of Contents

List of Tables

List of Figures

Notation

Notation	First Described	Description
$\mathbb{R}^n$	Chapter 1	Euclidean n-dimensional space
Ω	Definition 1.0.1	An open, bounded path connected region of $\mathbb{R}^n$.
$\partial\Omega$	Definition 1.0.1	The boundary of Ω.
$\overline{\Omega}$	Definition 1.0.1	The closure of Ω: $\overline{\Omega} = \Omega \cup \partial\Omega$ is the workspace of possible locations of the robot.
O	Definition 1.0.2	The set of all (closed) obstacles $O = O_1 \cup ... \cup O_i$.
$\mathbf{x}_0$, $\mathbf{x}_f$	Definition 1.0.2	Initial and final point of the path.
$\mathbb{S}^{n-1}$	Section 3.1	The set of all direction vectors in $\mathbb{R}^n$: $\mathbb{S}^{n-1} = \{\mathbf{u} \in \mathbb{R}^n \| |\mathbf{u}| = 1\}$.
$\mathbf{x} \in \Omega$	Section 3.1	A possible position of the vehicle.
$\mathbf{y}(\cdot), \mathbf{y}_\mathbf{x}(\cdot)$	Section 3.1	Trajectory of a vehicle. The subscript $\mathbf{x}$ in $\mathbf{y}_\mathbf{x}(\cdot)$ denotes initial position $\mathbf{y}(0) = \mathbf{x}$.
$\mathbf{u}, \mathbf{u}(\cdot)$	Section 3.1	Direction $\mathbf{u} \in \mathbb{S}^{n-1}$ and control $\mathbf{u}(\cdot) : \mathbb{R}_+ \to \mathbb{S}^{n-1}$.
$g(\mathbf{x}, \mathbf{u})$	Definition 3.1.2	The positive weight function $g : \overline{\Omega} \times \mathbb{S}^{n-1} \to \mathbb{R}_+$ describing the environment.
$Cost(\mathbf{x}, \mathbf{u}(\cdot))$	Definition 3.1.2 and (3.2)	The cost of a trajectory $\mathbf{y}(\cdot)$ defined by $\mathbf{u}(\cdot)$ starting at $\mathbf{x} \in \Omega$ with weight g.
V	Definition 3.1.4	The value function $V(\mathbf{x})$ representing the minimum cost to reach $\mathbf{x}_f$ from $\mathbf{x}$.
G_{min}, G_{max}	(3.3)	The minimum and maximum of $g(\cdot, \cdot)$ achieved on $\overline{\Omega} \times \mathbb{S}^{n-1}$ respectively.
$g_{min}(\mathbf{x})$, $g_{max}(\mathbf{x})$	(3.3)	The minimum and maximum of $g(\mathbf{x}, \cdot)$ achieved on $\mathbb{S}^{n-1}$ at $\mathbf{x} \in \overline{\Omega}$.
Γ	Definition 3.1.8	Measure of global anisotropy: $\Gamma = \frac{G_{max}}{G_{min}}$.
$C^\infty(\Omega)$	Section 3.2	The set of infinitely continuously differentiable functions on Ω.
ϕ	Section 3.2	A test function $\phi \in C^\infty(\Omega)$.
$B_\delta(\mathbf{x})$	Section 3.3	Open ball of radius δ centered at $\mathbf{x}$.
$D^- f(\mathbf{x})$ ($D^+ f(\mathbf{x}_0)$)	Section 3.3 and Definition 3.3.1	The set of all subgradients (and supergradients) of f at $\mathbf{x}$.

Notation	First Described	Description
$\underline{V}(\overline{V})$	Definition 3.2.3 (Definition 3.2.4)	Subsolution (Supersolution) to static HJB.
$H(\mathbf{x},\mathbf{p})$	(3.7)	Hamiltonian $H:\Omega\times\mathbb{R}^n\to\mathbb{R}$, described (3.6), (3.13): $H(\mathbf{x},\nabla V)=0$.
$\mathbf{u}^*$, $\mathbf{u}^*(t)$, $\mathbf{y}^*(t)$	Definition 3.1.3	Optimal direction (at a location $\mathbf{x}\in\Omega$), control and trajectory respectively.
G	Definition 4.1.1	A grid with orthogonal elements, each with dimensions $\triangle x_1, \triangle x_2, ..., \triangle x_n$.
$\triangle x$	Section 4.1.1	The spacing of the discretization of the rectangular element in the x-direction.
X	Section 4.1	A (simplicial) mesh.
h_{max}	Definition 4.1.9	The longest edge length of a mesh X.
h_{min}	Definition 4.1.11	The shortest simplex height.
$\widetilde{V}$	Section 4.1	Approximated value function that solves an approximation of (3.5) or (3.6).
$\mathbf{x}_i$	Section 4.1	A vertex of a mesh X or a grid G.
$\mathbf{x}_i\mathbf{x}_j$	Section 4.1	An edge of a mesh X or a grid G.
$\mathbf{x}_j^{\mathbf{s}}$	Section 4.1.2	A vertex of simplex $\mathbf{s}$.
$\mathbf{AF}$	Section 4.3	The *Accepted Front* in the OUM algorithm.
$\mathbf{NF}(\mathbf{x}_i)$	Section 4.3 and (4.11)	The *Near Front* of $\mathbf{x}_i$ in the OUM algorithm.
$US(\mathbf{x}_i)$	Definition 5.2.4	Updating stencil, set of $(n-1)$-simplices that include update from OUM.
$\widetilde{H}[US,\phi](\mathbf{x}_i,\phi(\mathbf{x}_i))$	(5.6)	Numerical Hamiltonian, approximation of H.
$\mathbf{u}^*(\mathbf{x}_i)$ $(\widetilde{\mathbf{u}}^*(\mathbf{x}_i))$	Definitions 3.1.6 and 5.2.1	Characteristic direction and approximated charcteristic direction.
$\overline{\mathbf{NF}}(\mathbf{x}_i)$	Section 5.2	The *Near Front* of $\mathbf{x}_i$ right before $\mathbf{x}_i$ is labelled *Accepted*.
$\overline{\mathbf{AF}}(\mathbf{x}_i)$	Definition 5.2.9	The subset of $\mathbf{AF}$ that is directionally complete for $\mathbf{x}_i$ right before $\mathbf{x}_i$ is labelled *Accepted*.
$S(\mathbf{x}_i)$	Definition 5.2.7	Updating stencil that satisfies the four properties of Definition 5.2.7.
$\overline{\Omega}_X$	Section 5.2	Largest closed subset of $\mathbb{R}^n$ contained in X.

Notation	First Described	Description
$\hat{V}(\mathbf{x})$	(5.67)	A linearly interpolated function on its n-simplices where $\hat{V}(\mathbf{x}_i) = V(\mathbf{x}_i)$ for all $\mathbf{x}_i \in X$.

Chapter 1

Introduction

Path planning is a widely studied field of robotics and is required in many applications. Examples of continuous path planning problems include a rover traversing terrain, precision arm robots on an assembly line, a small robot taking pictures for a colonoscopy and unmanned aerial vehicles. In contrast, the travelling salesman problem (where one has to travel between a fixed number of points while minimizing the total distance travelled) is a discrete path planning problem.

While it is worthwhile to study the geometry and physical capabilities of a robot, knowledge of its surroundings have significant relevance in its performance of a specified task. In virtually all path planning problems, obstacles must be avoided, including physical obstructions and high-risk regions. Obstacles may prevent a robot from completing the required function in the most direct path. See Figure 1.1. It may be trivial for human operators to obtain a path that avoids obstacles by inspection. However, the objective is to achieve complete autonomy. For example in extraterrestrial exploration, instructions from Earth should be kept to a minimum, with navigation to depend entirely on the algorithms that are internally programmed. It is important that these algorithms are well-researched and reliable.

In the global path planning problem, the environment is assumed to be fully known (by means of a map or a satellite to relay pertinent information), including the locations and shapes of obstacles. The advantage of global path planning over local path planning (where the environment is discovered by traversing through it) is that an optimal path may be found. For rovers, the terrain and location of the sun are also assumed to be known. The Mars Reconnaissance Orbiter is a satellite that posseses a high resolution camera. The necessary information for global path planning could be relayed to the rovers

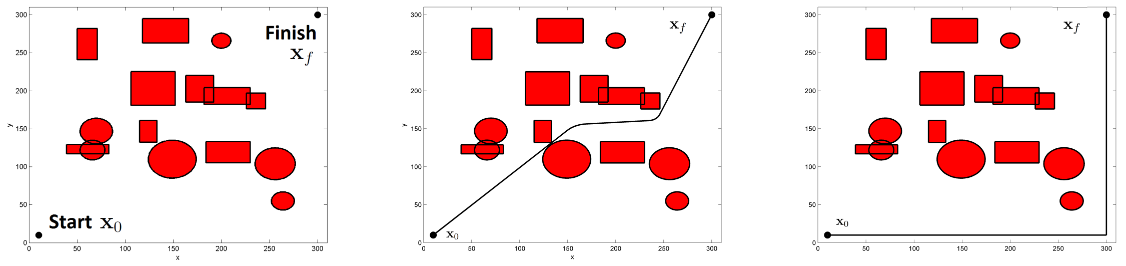


Figure 1.1: Global continuous path planning problem in $\mathbb{R}^2$ - Obstacles are shaded rectangles and circles. Left: environment/workspace, Centre: feasible path, Right: another feasible but not shortest path.

on the ground. For the remainder of the thesis, unless otherwise specified, all path planning problems will be assumed to be global and continuous. A definition of the global continuous path planning problem is presented.

Let $\Omega \subset \mathbb{R}^n$ be an open and bounded path-connected set, and let $\partial\Omega$ denote its boundary. The closure of Ω is denoted $\overline{\Omega} = \Omega \cup \partial\Omega$. The path planning problem will be solved on the region Ω.

Definition 1.0.1. *The* ***workspace*** $\overline{\Omega} \subset \mathbb{R}^n$ *is a closed and bounded set of possible locations of the robot.*

Let $\mathbf{x}_0, \mathbf{x}_f \in \Omega$ be the initial and final locations of the desired path respectively. A common objective is to avoid regions within Ω designated as obstacles. Let there be a finite number of obstacles i, such that $O_i \subset \overline{\Omega}$ is closed for all i. Define $O = O_1 \cup O_2 \cup ... \cup O_i \subset \overline{\Omega}$ the set of all obstacles. For a set $A \subset \mathbb{R}^n$, its complement is denoted A^c and is such that $A \cup A^c = \mathbb{R}^n$ and $A \cap A^c = \emptyset$.

Definition 1.0.2. *The* ***continuous path planning problem*** *is to find a path* $C : [0, T] \to \mathbb{R}^n$, $T > 0$ *such that* $C(0) = \boldsymbol{x}_0$, $C(T) = \boldsymbol{x}_f$ *and* $C(t)$ *does not encounter any obstacles. That is,* $C(t) \in \Omega \cap O^c$ *for all* $t \in [0, T]$.

Definition 1.0.3. *A solution to the continuous path planning problem is known as a* ***feasible path****.*

For surface vehicles such as rovers, the path is usually found on $\Omega \subset \mathbb{R}^2$, or on a surface. The focus of this work was to find a feasible path, rather than finding the lower level controls to follow the path. This is reasonable for rovers as they travel very slowly (5cm/sec, [56]), and can turn in place. Details on modelling the dynamics of the rover can be found in [37, 56, 88]. For aircrafts, $\Omega \subset \mathbb{R}^3$ is often used. Manipulator arms can have a larger number of coordinate dimensions according to the number of joints it possesses. Many approaches have been developed to find feasible paths for the path planning problem. In Chapter 2, several algorithms that solve the path planning problem are presented, including the artificial potential field [45], Rapidly-exploring Random Trees [52], optimal problems on graphs such as Dijkstra's algorithm [23], and genetic algorithm [2, 28]. The review is an introduction to some prevalent ideas found in literature. For a more comprehensive review, see [51].

Feasible paths are not unique in general; additional objectives such as minimizing time or distance traveled may be considered in an optimal path planning problem. In the context of the rovers used in extraterrestrial exploration, safe and efficient paths are important due to the high costs incurred. The optimal control problem is presented in Chapter 3. The path planning problem where aspects of the environment are modelled by a weight function can be posed as an optimal control problem. The control problem is reposed as a partial differential equation (PDE) of the value function in dynamic programming, known as the static Hamilton-Jacobi-Bellman (HJB) equation. Differentiable solutions to the static HJB equation may not exist on the entire workspace $\overline{\Omega}$, so a weaker notion of solution known as a viscosity solution [26] is presented.

The optimal path can be found using the solution to the static HJB equation as a navigation function [69] to guide the rover at each position. Navigation functions in general may have local minima that are not global minima. A feasible path for a robot at a configuration corresponding to a local minimum cannot be found [45]. Under a simple positivity assumption on the weight function, the solution of the static HJB equation is shown to have no local minima in Chapter 3. The result is that the optimal path is always found using the solution. Though this property has been previously described for the simpler Eikonal equation [59], a proof of this property for the viscosity solution has not yet been found in literature.

Two new criteria specific to rover path planning are modelled in Chapter 3: solar energy absorption and tip-over stability [63]. The latter depends on the travel direction of the rover. The tip-over stability was considered using Fast Marching Method (FMM) [58] by averaging the risk experienced in four predetermined directions. The risk was not accurately measured. Using a direction-dependent weight, the correct direction that minimizes the risk is found. Solutions to the static HJB equation with direction-dependent weights cannot in general be accurately approximated using FMM. A class of direction-dependent problems can be solved using FMM if the direction-dependence is aligned with the axes of a grid [4]. The solar energy abosrption is a new application to rover optimal path planning. These two criteria are combined additively with shortest length, obstacle avoidance and soil type.

It is often difficult to describe viscosity solutions analytically, so numerical approximations are obtained. In Chapter 4, a discretization of the region Ω known as a simplicial mesh are described. Several algorithms to approximate static HJB equations on simplicial meshes and grids including the Fast Marching Method (FMM) [47] and Ordered Upwind Method (OUM) [72], which have been used to solve both direction-independent and direction-dependent rover path planning problems respectively [76] are presented.

A novel algorithmic improvement of OUM [72], OUM-Bi-Directional (OUM-BD), is made to include a bi-directional search [66]. It is shown that with a slight modification to the anisotropic weight function, an equivalent problem can be stated, where the algorithm is initialized from $\mathbf{x}_0$ rather than $\mathbf{x}_f$. Both instances are executed concurrently, building individual fronts outwards, stopping when the solutions meet. A brief description of a bi-directional search used for Fast Marching Method is found in [15]. The Monotone Acceptance OUM (MAOUM) [5], Buffered Fast Marching Method [20] and the Fast Sweeping Method [41] are presented. These algorithms are compared in timing and computational effort in Chapter 6.

A different formulation of the path planning problem is to consider an optimal escape route out of a region. Rather than planning an optimal path from one point to another, an optimal path is planned from $\mathbf{x}_0$ inside Ω to (any point on) its boundary $\partial\Omega$. An exit-cost can be defined for each point on the boundary. The resulting static HJB equation is a boundary value problem (BVP); see Figure 1.2. An application is to find the best escape route to avoid catastrophic weather conditions in a given area.

In Chapter 5, the OUM solution is proven to converge to the viscosity solution of the static HJB equation BVP at a rate of at least $\mathcal{O}(\sqrt{h_{max}})$ as $h_{max} \to 0$ where h_{max} denotes the longest edge of the mesh. This is to our knowledge the tightest bound on error thus far for the Ordered Upwind Method. Only convergence without guarantee of rate has been previously shown [72]. The OUM convergence rate result is similar to [61] for FMM, which was proven for uniform unity weight function and zero boundary condition on a square grid. The proof shown here is for a different algorithm, namely OUM, that solves the optimal path planning problem for a more general class of weight and boundary functions on general unstructured meshes. A key step in the proof for the OUM method is the proof of existence of a directionally complete stencil. Similar ideas have been presented in [5]. The work in Chapter 5 has been submitted for publishing [75].

In Chapter 6, solutions and approximations to the HJB equation described in Chapters

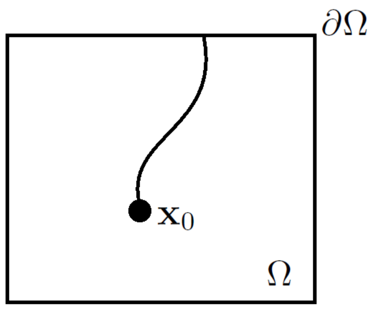


Figure 1.2: Minimizing the cost of a path from $\mathbf{x}_0 \in \Omega$ to the boundary $\partial\Omega$.

3 and 4 are presented. A comparison is made between FMM, FMM-BD (Bi-directional FMM), OUM, OUM-BD and a genetic algorithm rover path planner [28] in timing and performance. Comparisons are made between the OUM for timing with the methods described in Chapter 4. Numerical convergence of the OUM solution to the viscosity solution is also shown for a particular boundary value problem. The rate of convergence proved in Chapter 5 is confirmed numerically. The work in Chapters 3, 4 and 6 have been submitted for publishing [76].

Conclusions and directions for future work are discussed in Chapter 7.

Chapter 2

Path Planning Strategies

The algorithms presented in this chapter provide a non-optimal solution to the continuous path planning problem (Definitions 1.0.1 - 1.0.3). The review is not exhaustive, but it encompasses a wide range of ideas that are found in literature. Though the motivation for this research is path planning for rovers, the discussed algorithms can be used for many types of robots including robotic arm manipulators and aerial vehicles. Many of the algorithms presented in this chapter are related to work found in future chapters. A more detailed review on non-optimal path planning algorithms can be found in [51].

Considerable attention has been devoted in literature to the use of Model Predictive Control [31] to solve the finite-horizon discrete optimal control problem. At each time step t_k, a trajectory is found from the current state $\mathbf{y}(t_k)$ by solving a set of locally optimal problems, planning intermediate steps to the desired state $\mathbf{x}_f$. Only the first step of the calculated trajectory is executed, and the process is repeated. The result is not globally optimal [10]. Model Predictive Control has often been employed for linear and quadratic cost and constraint problems [6, 32]. With this simplification, most path planning problems in this approach consider only obstacle avoidance when modelling the environment [10, 12, 54, 86]. For Mixed-Integer Linear Programming used with Model Predictive Control [10, 11, 12, 55], obstacles are modelled using a set of linear constraints with binary variables that determine collision in a constrained optimization problem. Though fully nonlinear models have been considered, only locally optimal controls may be found. As well, verification of global minima is not straightforward [40]. A more complicated model for the dynamics can be considered using linear equality constraints. Modelling the features of the environment experienced by rovers cannot easily be done using these constraints.

For simplicity, all examples of the algorithms are presented for $\overline{\Omega} \subset \mathbb{R}^2$. All of the presented algorithms generalize easily to $\mathbb{R}^n$.

The artificial potential field will be presented in Section 2.1. Probabilistic methods that search workspaces with randomized variables are presented in Section 2.2. These include the Rapidly-exploring Random Trees (2.2.1) and genetic algorithms (2.2.2). Optimal path planning on discrete graphs including Dijkstra's algorithm will be presented in Section 2.3. The optimal solutions on graphs for these problems do not generally extend to the optimal solutions of continuous path planning problems.

2.1 Artificial Potential Fields

In the Artificial Potential Field method [45] fictitious force fields are used to model the workspace based on the position of the final goal point $\mathbf{x}_f$ and obstacle set O. The goal location acts as an attractor to the robot while obstacles repel it. The artificial potential function whose gradient gives the resultant field will be used to guide the robot. Define the shortest distance from a point $\mathbf{x} \in \Omega$ to an obstacle O_i,

$$\rho_i(\mathbf{x}) = \min_{\widetilde{\mathbf{x}} \in O_i} \|\mathbf{x} - \widetilde{\mathbf{x}}\| .$$

Define the attractive and repulsive functions $U_{att} : \overline{\Omega} \to \mathbb{R}_+$, and $U_{rep} : \overline{\Omega} \to \mathbb{R}_+$

$$U_{att}(\mathbf{x}) = \frac{1}{2} \|\mathbf{x} - \mathbf{x}_f\|^2 \tag{2.1}$$

and

$$U_{rep}(\mathbf{x}) = \sum_{i=1}^{N} U_{O_i}(\mathbf{x}), \tag{2.2}$$

where

$$U_{O_i}(\mathbf{x}) = \begin{cases} \frac{1}{2}\eta_i \left(\frac{1}{\rho_i(\mathbf{x})} - \frac{1}{\rho_{O_i}} \right)^2 & \text{if } \rho_i(\mathbf{x}) \le \rho_{O_i} \\ 0 & \text{if } \rho_i(\mathbf{x}) > \rho_{O_i} \end{cases} \tag{2.3}$$

and $\eta_i > 0$ and ρ_{O_i} are the weighting factor and radius of influence of the obstacle O_i respectively. The region in which U_{O_i} is non-zero is known as the region of influence of obstacle O_i. As the robot approaches an obstacle O_i, $U_{O_i}(\mathbf{x}) \to \infty$. Obstacles with overlapping regions of influence are resolved through addition (2.2). The artificial potential field function $U_{apf}(\mathbf{x}) : \overline{\Omega} \to \mathbb{R}_+$ is

$$U_{apf}(\mathbf{x}) = U_{att}(\mathbf{x}) + U_{rep}(\mathbf{x}).$$

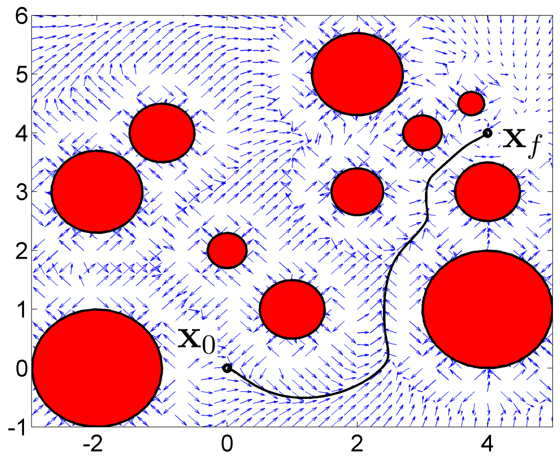


Figure 2.1: Artificial Potential Field - The obstacles are shaded. A path is planned from $\mathbf{x}_0$ to $\mathbf{x}_f$ following the field shown with arrows. No local minima were encountered by the path. Local minima may still exist in the field.

Since $U_{O_i} \geq 0$, $\mathbf{x}_f$ is a global minimum of U_{apf} if it lies outside the region of influence of the set of all obstacles O.

The gradient (where defined) of the artificial potential function is a force vector field used to direct the robot. To find a path, the direction $-\frac{\nabla U_{apf}}{\|\nabla U_{apf}\|}$ from the initial location $\mathbf{x}_0$ is followed until a local minimum is reached. If the local minimum is $\mathbf{x}_f$, then a feasible path has been found. Otherwise, a feasible path is not found and the robot has beome stuck. An example of artificial potential field is shown in Figure 2.1.

2.1.1 Local Minima

Since the path is found by following $-\frac{\nabla U_{apf}}{\|\nabla U_{apf}\|}$, the algorithm halts at points such that $\nabla U_{apf} = 0$. The position of the robot can be perturbed at local maxima and saddle points allowing the search to continue. This cannot be done for local minima, which arise in three settings. See Figure 2.2. The first is non-convex obstacles, which unless the goal lies within the convex hull can easily be remedied by considering the convex hull of an obstacle as a new obstacle. The second is through overlapping regions of influence of multiple obstacles (2.2). Finally, local minima can occur if an obstacle is in close proximity of the goal, and is

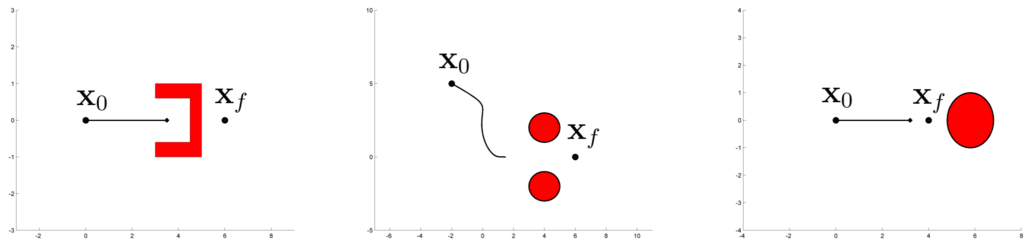


Figure 2.2: Artificial Potential Field Local Minima - Initial location $\mathbf{x}_0$ and desired final location $\mathbf{x}_f$. Shaded areas represent obstacles. Left: Non-Convex Obstacle, Centre: Multiple Obstacles, Right: Overpowering Obstacle

weighted to overpower U_{att}, shifting the minimum. A new artificial potential function was proposed in [82] to ensure that the attractive force of the goal overpowers the repulsive force of any obstacle.

Additional limitations of the original artificial potential field algorithm including oscillations in the presence of obstacles and narrow passages are summarized in [48].

2.1.2 Extensions and Limitations

Many advances have been proposed since the introduction of artificial potential fields. The artificial potential function is an example of a navigation function [69] that guides robots based on a vector field. Under a new formulation [69], local minima do not occur if all obstacles obey a property weaker than convexity known as star-shaped. Harmonic potential fields [16, 70] have been used to guarantee a potential function free of local minima, but the workspace is limited to maze-like environments. More recent advances [33] using a finite element method allow for more complicated shapes in workspace and obstacles. Additional methods exist to deal with local minima, such as random walks [53] and tangential fields [57], causing the robot to trace around obstacles. A virtual force [24] based on the location of the robot in the workspace has been implemented to escape local minima. In general, the artificial potential field needs only to be calculated locally, and can be used for dynamic environments with moving obstacles [62].

Optimal paths in general cannot be found with artificial potential fields. The robot reacts to obstacles only upon approach. Knowledge of obstacle placement is not used to obtain the shortest path. Genetic algorithm is used in the Evolutionary Potential Field method [81] to produce successively better potential functions according to factors such as path length, smoothness and obstacle avoidance. Local minima are still present and escape algorithms are used. The value function in the optimal control problem discussed in Chapter 3 is a navigation function that is free of local minima and capable of producing an optimal path.

2.2 Probabilistic Methods

Random variables are used in probabilistic methods to determine the feasibility of paths or connections in the workspace. In this section, Rapidly-exploring Random Trees and Genetic Algorithm are presented. In both algorithms, points in the workspace $\overline{\Omega}$ are randomly sampled, but the methods in which these points are used is quite different.

2.2.1 Rapidly-exploring Random Trees

In Rapidly-exploring Random Trees [87], the workspace $\overline{\Omega}$ is efficiently sampled by adding collision-free connections to a growing structure known as a tree. Some definitions from graph theory are required.

Definition 2.2.1. *A* ***graph*** $G(\mathcal{V}, \mathcal{E})$ *is an ordered pair of sets containing a set of vertices (points),* $\mathcal{V}$ *, and a set of edges,* $\mathcal{E}$ *whose elements are 2-element subsets of* $\mathcal{V}$*.*

Edges made from two vertices $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{V}$ may be denoted $\mathbf{x}_i\mathbf{x}_j \in \mathcal{E}$. If $\mathbf{x}_i\mathbf{x}_j \in \mathcal{E}$, $\mathbf{x}_i$ and $\mathbf{x}_j$ are said to be connected. The set of edges $\mathcal{E}$ describes the connectivity of the set $\mathcal{V}$ in the graph G.

Definition 2.2.2. *A* ***path*** *between two vertices* $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{V}$ *is a sequence of edges in* $\mathcal{E}$ *that connect* $\boldsymbol{x}_i$ *and* $\boldsymbol{x}_j$ *through any sequence of intermediate vertices in* $\mathcal{V}$*. A path between* $\boldsymbol{x}_i, \boldsymbol{x}_j$ *is* ***simple*** *if each vertex in the sequence appears only once.*

Definition 2.2.3. *A graph* $G(\mathcal{V}, \mathcal{E})$ *is* ***connected*** *if for every* $\boldsymbol{x}_i$*,*$\boldsymbol{x}_j \in \mathcal{V}$*, a path on* G *exists between* $\boldsymbol{x}_i$ *and* $\boldsymbol{x}_j$*.*

Definition 2.2.4. *A* ***tree*** $T(\mathcal{V}, \mathcal{E})$ *is a connected graph where for every two vertices* $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{V}$*, there is exactly one simple path between them.*

In the Rapidly-exploring Random Tree algorithm, a tree is built starting from the initial configuration $\mathbf{x}_0$. Let $d \in \mathbb{R}_+$, be the branch length parameter that is small relative to the size of the workspace $\overline{\Omega}$. Let T be a tree initialized with $\mathcal{V} = \{\mathbf{x}_0\}$ and $\mathcal{E} = \emptyset$. The algorithm is as follows.

1. Choose a random point $\mathbf{x}_r$ from a continuous uniform distribution on Ω.
2. Let $\mathbf{x}_c = \arg\min_{\mathbf{x}_k \in \mathcal{V}} \|\mathbf{x}_k - \mathbf{x}_r\|$ (Note $\mathbf{x}_c = \mathbf{x}_0$ in the first step).
3. If $\|\mathbf{x}_c - \mathbf{x}_r\| \leq d$, define $\mathbf{x}_{new} = \mathbf{x}_r$, otherwise define
$$\mathbf{x}_{new} = d\frac{\mathbf{x}_r - \mathbf{x}_c}{\|\mathbf{x}_r - \mathbf{x}_c\|} + \mathbf{x}_c.$$
4. If there is no collision between $\mathbf{x}_c$ and $\mathbf{x}_{new}$, add $\mathbf{x}_{new}$ to $\mathcal{V}$ and $\mathbf{x}_{new}\mathbf{x}_c$ to $\mathcal{E}$.
5. If $\|\mathbf{x}_{new} - \mathbf{x}_f\| \leq d$ and $\mathbf{x}_{new}\mathbf{x}_f$ does not encounter an obstacle, continue. Otherwise return to step 1.
6. Add $\mathbf{x}_f$ to $\mathcal{V}$ and $\mathbf{x}_{new}\mathbf{x}_f$ to $\mathcal{E}$. Terminate the algorithm.

The point $\mathbf{x}_r$ may be inside an obstacle, but this does not affect the performance of the algorithm. The output of the above algorithm is a tree that contains a simple path connecting $\mathbf{x}_0$ and $\mathbf{x}_f$. An example in $\mathbb{R}^2$ is shown in Figure 2.3. The feasible path is found by backtracking through the tree from $\mathbf{x}_f$. The size of the tree at termination is not fixed. A different path can be found each time the algorithm is executed, hence the produced path is not optimal. The algorithm is probabilistically complete. A feasible path is guaranteed after an infinite number of iterations. However, certain configurations of obstacles such as a narrow corridor can provide slow convergence. To improve convergence, a bi-directional search and a greedy heuristic have been implemented [49].

Recent improvements include Rapidly-exploring Random Tree-* [43] where costs can be introduced in the context of optimal path planning. The original algorithm is modified to redefine the connectivity $\mathcal{E}$ between vertices in $\mathcal{V}$ to lower the cost of paths between vertices of $\mathcal{V}$ and $\mathbf{x}_0$ according to a given cost function. Additional sampling results in closer-to-optimal paths. Asymptotic optimality, that the optimal path is found after infinite iterations, has been proven. Fast Marching Trees [39] algorithm is also asymptotically optimal. A tree is built by sampling nearby points and moving outwards. The edges corresponding to the best paths are kept. The algorithm is reminiscent of the expanding front used in the Fast Marching Method (Chapter 4).

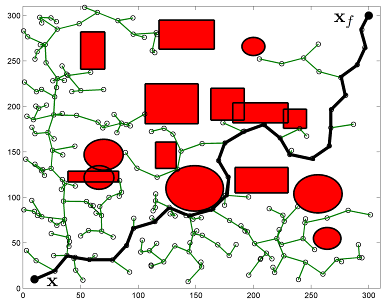


Figure 2.3: RRT Algorithm - The tree is completed after about 200 iterations. The feasible path is shown in bold with maximum branch length $d = 25$.

The Probabilistic Roadmap Method [44] is another sampling method that solves the path planning problem in two phases. In the construction phase, connections are tested for feasibility and a connectivity graph is built. In the query phase, the initial $\mathbf{x}_0$ and final $\mathbf{x}_f$ locations are added to the graph, and the shortest path is found using Dijkstra's algorithm [23] (described Section 2.3). Multiple path planning problems can concurrently be solved in the same workspace.

Though there are no guarantees of the optimal path being found in finite time, the algorithms described here provide ways to efficiently sample a large portion of the workspace $\overline{\Omega}$. Further iterations in asymptotically optimal algorithms improve the path.

2.2.2 Genetic Algorithm

In the binary genetic algorithm [60], the search space of paths from initial configuration $\mathbf{x}_0$ and final configuration $\mathbf{x}_f$ are discretized to a finite set represented by a string of binary digits (bits). In [2], the workspace $\overline{\Omega}$ is discretized by a fine square grid $\widetilde{\Omega}$ where each position on the grid is represented in bits based on location. Paths are limited to up, down, left and right motions between points. In [28], candidate paths $\mathbf{c}^k$ are stored as a binary representation of a sequence of n control points $\mathbf{x}_i \in \widetilde{\Omega}$: $\mathbf{c}^k = \mathbf{x}_1^k \mathbf{x}_2^k \cdots \mathbf{x}_n^k$ connecting $\mathbf{x}_0$ to $\mathbf{x}_f$. The actual paths are defined as cubic splines between the control points that lie in between the two fixed end-points $\mathbf{x}_0$ and $\mathbf{x}_f$. While the context of this presentation is path planning, the genetic algorithm is a versatile optimization method and can be easily applied to other problems. Each candidate solution has an associated cost known as a fitness function $f : \{\mathbf{c}^k\} \to \mathbb{R}$.

In each iteration of the algorithm, known as a generation, a population of $l \in \mathbb{Z}_+$ (where l is even) candidate solutions are created using the previous generation. These solutions are created in a manner that mimics the evolutionary processes: selection, crossover and mutation. The k-th candidate solution at generation t is denoted $\mathbf{c}_t^k$.

Selection is a mechanism for choosing which candidate solutions in a current generation is used in producing the next generation. Candidate solutions are selected to "reproduce" with probability based on its cost. Selection of a candidate solution does not remove it from the population. A candidate solution with lower cost will be chosen more often. In selection, two candidate solutions $\mathbf{c}_t^j$ and $\mathbf{c}_t^k$ are selected to undergo crossover and mutation, returning two new candidate solutions $\mathbf{c}_{t+1}^j$ and $\mathbf{c}_{t+1}^k$ for the next generation.

In crossover, a random bit in the string is chosen, and the subsequent digits of the two candidate solutions are swapped. For example, crossover on the fourth digit for candidates $\mathbf{c}_t^j = 0000000000$ and $\mathbf{c}_t^k = 1111111111$ would result in the new candidates $\mathbf{c}_{t+1}^j = 0000111111$ and $\mathbf{c}_{t+1}^k = 1111000000$. Crossover occurs with probability p_c. If crossover does not occur, then the new candidates are the same as the original candidates.

In mutation, a random bit in the string is chosen and altered with probability p_m. For example, a candidate $\mathbf{c}_t^k = 0111010101$ is mutated in its 5th digit to $\mathbf{c}_{t+1}^k = 0111110101$. Both candidates $\mathbf{c}_t^j$ and $\mathbf{c}_t^k$ chosen for reproduction undergo mutation.

The algorithm is terminated once a termination condition, determined by the user, is reached. Commonly used termination conditions include a limit on the number of iterations, a cost threshold for when a particular cost is reached, or a lack of progress in previous generations. Cost thresholds should be used in conjunction with other termination conditions if the optimal cost is not known.

The following is the simple binary genetic algorithm [60].

1. Initialize the population of candidates $\{\mathbf{c}_0^i\}$, $i = 1, ..., l$ for generation $t = 0$.

2. Calculate the cost of each candidate $f(\mathbf{c}_t^i)$, $i = 1, ..., l$.

3. Repeat the following steps until a new population of l candidates is produced.

   (a) Select two candidates $\mathbf{c}_t^j$ and $\mathbf{c}_t^k$ from the current population in generation t.

   (b) Perform crossover on $\mathbf{c}_t^j$ and $\mathbf{c}_t^k$ with probability p_c.

   (c) Perform mutation on each of $\mathbf{c}_t^j$ and $\mathbf{c}_t^k$ with probability p_m.

   (d) Add $\mathbf{c}_{t+1}^j$ and $\mathbf{c}_{t+1}^k$ to the $(t+1)$-th generation of candidates $\{\mathbf{c}_{t+1}^i\}$.

4. If the termination condition is met, terminate the algorithm and return $\mathbf{c}_{t+1}^{k*} \in \{\mathbf{c}_{t+1}^{i}\}$ with the lowest cost. Otherwise, increment t by 1 and go to Step 2.

The success of the binary genetic algorithm is largely dependent on the discretization of solutions into bits and the choice of the parameters p_c, p_m, l, n. The best candidate solution is often kept unchanged in each iteration, ensuring the cost of the best candidate is nonincreasing in subsequent generations. There are many variants of genetic algorithm with additional heuristics to improve performance [60]. Continuous genetic algorithms [36] have been developed to represent candidates using floating point variables rather than bits. The limitation in terms of the discretized space is now the internal precision used by the computer. Extensions include the use of line crossover, guided crossover and shrinking window mutation [68] which alter the candidates in the search-space directly, and decrease the effect of each operation as the algorithm progresses.

The rover path planning problem has been solved [28] using a genetic algorithm. Due to the versatility of the algorithm, a wide variety of factors including energy consumption and regeneration, the force-angle stability margin, soil type and obstacle avoidance were considered.

Unfortunately, there is no guarantee of optimality in finite time. It is difficult to estimate a cost threshold that would be close to the global minimum. In Chapter 6, a comparison of the rover path planner using a genetic algorithm [28] is made with methods described in Chapter 4.

2.3 Continuous Path Planning Using Graphs

In this section, the optimal path planning problem on graphs is considered. Optimal solutions on graphs do not in general extend to the optimal solutions of continuous path planning problems. Some examples are shown.

The workspace $\overline{\Omega}$ can be discretized by a graph $G(\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V}$ represent possible locations and edges $\mathcal{E}$ represent the possible directions of travel. Let the function $g : \mathcal{E} \to \mathbb{R}_+$ denote the weight associated with traversing an edge in $\mathcal{E}$.

Assume the graph G is connected and let $\mathbf{x}_f \in \mathcal{V}$ be the goal vertex. Let $\mathcal{N}(\mathbf{x}_i)$ denote the set of neighbouring vertices that have an edge in $\mathcal{E}$ connecting to $\mathbf{x}_i \in \mathcal{V}$ in G. The optimal cost-to-go function V at a vertex $\mathbf{x}_i$ is the lowest cost to reach the goal vertex $\mathbf{x}_f$. The vertices in G will be assigned between the following labels based on whether V has been finalized at those vertices.

Far - The vertices that have not yet been where the computation for V has not yet begun, and have tentative values $V(\mathbf{x}_i) = K$, where K is a large value.

Considered - The cost-to-go function has not yet been finalized at these vertices, but has tentative values $V < K$. These values are updated from neighbouring vertices where V has been finalized using an update function $C : \mathcal{V} \to \mathbb{R}$.

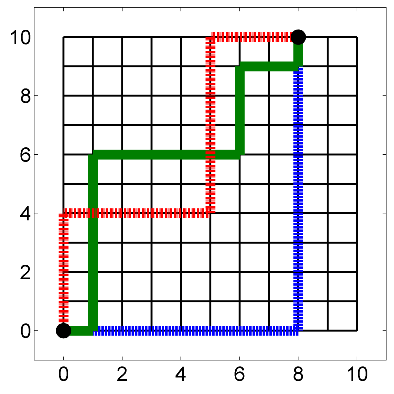
Accepted - The cost-to-go function V has been finalized at these vertices.

At any instant of the algorithm, each vertex of $\mathcal{V}$ must be assigned exactly one label. The relabelling of a vertex removes the old label. Dijkstra's algorithm is as follows.

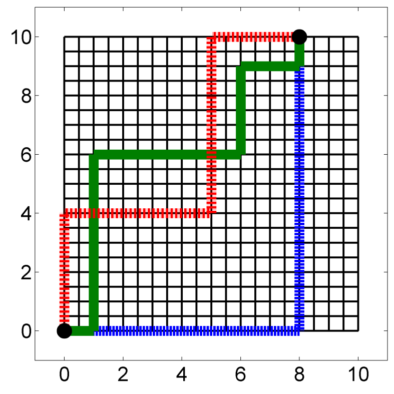
1. Label all vertices $\{\mathbf{x}_i\} \in \mathcal{V}$ *Far* with tentative values $V(\mathbf{x}_i) = K$.
2. Relabel the goal vertex $\mathbf{x}_f$ *Accepted* and set $V(\mathbf{x}_f) = 0$.
3. Relabel all the neighbouring vertices $\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_f)$ of $\mathbf{x}_f$ *Considered* and set $V(\mathbf{x}_i) = g(\mathbf{x}_i\mathbf{x}_f)$.
4. If there are no vertices labelled *Considered*, terminate the algorithm. Otherwise, find the vertex $\overline{\mathbf{x}}_i = \underset{\mathbf{x}_i \in Considered}{\arg\min} V(\mathbf{x}_i)$ and relabel $\overline{\mathbf{x}}_i$ *Accepted.*
5. Relabel the neighbours of $\overline{\mathbf{x}}_i$ with *Far* label *Considered.*
6. For all the neighbours of $\overline{\mathbf{x}}_i$, labelled *Considered* $\{\mathbf{x}_i\}$, let $C(\mathbf{x}_i) = V(\overline{\mathbf{x}}_i) + g(\mathbf{x}_i\overline{\mathbf{x}}_i)$. If $V(\mathbf{x}_i) > C(\mathbf{x}_i)$, update $V(\mathbf{x}_i) = C(\mathbf{x}_i)$. Otherwise, do nothing.
7. Go to step 4.

Paths are found by following the neighbouring vertex of greatest decrease on V from an initial vertex $\mathbf{x}_0 \in \mathcal{V}$ to the goal vertex $\mathbf{x}_f$. Many of the algorithms associated with optimal control presented in chapter 4 are based on Dijkstra's algorithm.

Though Dijkstra's algorithm solves a discrete problem, the algorithm can be applied to the continuous problem. An example of a graph G that discretizes the workspace $\overline{\Omega}$ is a grid with square elements. The vertices $\mathcal{V}$ are the grid points and the edges $\mathcal{E}$ connect them in a grid-like manner. The weighted edges can represent length if the edge does not encounter obstacles, and a large number otherwise. Though the solution of Dijkstra's algorithm is optimal for a discrete graph [23], the optimal paths of this discrete problem do not in general converge to the optimal path of the continuous problem as finer discretizations are



(a) Each edge has a weight of 1.



(b) Each edge has a weight of 0.5. The finer discretization does not improve the solution, but increases the number of optimal paths.

Figure 2.4: Dijkstra's algorithm on grid discretizing $\overline{\Omega}$ - Initial point $\mathbf{x}_0 = (0,0)$, goal point $\mathbf{x}_f = (8,10)$. The same optimal paths are shown for two different discretizations. There are many optimal paths each with length 18 (though only 3 are shown).

used [74]. For example, consider a path planning problem in $\mathbb{R}^2$ where the cost is defined by the length of the path. Let the workspace $\overline{\Omega} = [0,10]^2$ be discretized by the square grid G with vertices $\mathcal{V} = [0,10]^2 \cap \mathbb{Z}^2$. Each vertex (i,j) is connected to $(i \pm 1, j)$, $(i, j \pm 1)$, provided they are in $\mathcal{V}$. Let $\mathbf{x}_0 = (0,0)$, $\mathbf{x}_f = (8,10)$. See Figure 2.4. Refining the grid by splitting squares increases the number of optimal solutions, but has no effect on the cost.

A reduction in the cost of the optimal path is observed when diagonal connections are added. See Figure 2.5. Increasing the connectivity (directions of travel) will improve the results [64], which is the basis for the Fast Marching and Ordered Upwind Methods (presented in chapter 4). Despite the solution not converging to the optimal solution of the continuous path planning problem, Dijkstra's algorithm will yield a feasible path to the continuous path planning problem provided one exists along the edges of the graph. If one does not, a different graph must be used to discretize the workspace.

The discrete path planning problem on the edges of a graph has also been solved optimally by the A^* algorithm [35], where a heuristic to guide the search from the start goal to the end goal is implemented. Another algorithm D^* [77] is a replanning algorithm used to produce optimal paths in partially known environments.

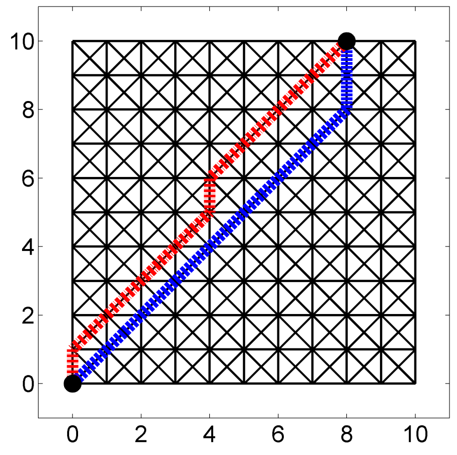


Figure 2.5: Dijkstra's algorithm discretizing $\overline{\Omega}$ with additional connectivity - Two of the optimal paths are shown. All optimal paths on the above graph have length $8\sqrt{2} + 2$. Adding more connections (directions of travel) improve the solution. In general, N many directions are required to obtain the optimal path between any two vertices, where N is the number of vertices in G.

Chapter 3

Path Planning using Optimal Control

In this chapter, the optimal path planning problem is formulated. The same continuous path planning problem presented in Definitions 1.0.1 - 1.0.3 is solved, but with additional considerations. Rather than finding any feasible path, the best path according to some criteria is found. Shortest paths while avoiding obstacles have been considered [46, 47]. In [4], a multi-robot path planning problem is solved using different norms.

The cost of a path is measured by a weighted path integral. The weights used in the optimal control problem can be dependent or independent of direction. Currents for underwater autonomous vehicles [64] and wind for ground and air vehicles [5] can be modelled as direction-dependent weights. Path planning considering a mix of continuous and discrete states has been performed [71].

New aspects of the environment for rover path planning are modelled. Using the force-angle stability margin [63], the risk due to tip-over is measured in the direction in which the rover is facing. In the previous direction-independent model [58], the weight was the average of the risk experienced in four predetermined directions. Directions that were unsafe for travel were omitted from the search. Modelling tipover risk as a direction-dependent weight provided more accuracy. Solar energy is also included here in the modelling of net consumed energy. As in previous formulations, the shortest path on a surface and obstacle avoidance are considered.

The optimal control problem in $\mathbb{R}^n$ of solving an ordinary differential equation while minimizing a cost is recast into a partial differential equation of n-spatial variables known as the static Hamilton-Jacobi-Bellman (HJB) equation [8, 26]. The solution to the HJB equation is analogous to a continuous version of the discrete cost-to-go function described in Dijkstra's algorithm. Solutions of the static HJB equation are rarely differentiable over the entire workspace. A weaker sense of solution that satisfies the HJB equation known as viscosity solutions is presented.

In the context of path planning, the value function that solves the HJB equation is used as a navigation function [69] to provide a direction of travel at each point to determine the optimal path. In Section 2.1, the artificial potential function was used as a navigation function to solve path planning problems. The artificial potential function encountered local minima, preventing a feasible path from being found. The value function that solves the static HJB equation will be shown to not possess any local minima given that the weight is assumed to be positive.

In Section 3.1, the control problem is stated. A brief introduction of the dynamic programming principle (DPP) and its equivalence to the HJB equation is given. An introduction to viscosity solutions is presented in Section 3.2. It is shown in Section 3.3 that the solution to the HJB equation cannot have local minima. Aspects that can be used to model the rover environment will be discussed in Section 3.4.

The new contributions described in this chapter include a proof that the viscosity solution of the HJB equation do not encounter local minima, as well as the modelling of tipover risk and solar power using direction-dependent weights in rover path planning.

3.1 Continuous Global Path Planning Problem

Recall the continuous global path planning problem in Definitions 1.0.1 - 1.0.3. In the context of vehicles, let $\mathbf{y} : \mathbb{R}_+ \to \mathbb{R}^n$ describe the trajectory of a vehicle, and let $\mathbf{u} : \mathbb{R}_+ \to \mathbb{S}^{n-1}$, where $\mathbb{S}^{n-1} = \{\mathbf{u} \in \mathbb{R}^n | \, \|\mathbf{u}\| = 1\}$ define the direction in which the vehicle is facing. The set of admissible controls is defined $\mathcal{U} = \{\mathbf{u}(\cdot) : \mathbb{R}_+ \to \mathbb{S}^{n-1} | \; \mathbf{u}(\cdot) \text{ is measurable}\}$. The path $\mathbf{y}$ traced out by the vehicle for $t \geq 0$ is

$$\dot{\mathbf{y}}(t) = \mathbf{u}(t), \quad \mathbf{y}(0) = \mathbf{x}_0, \quad \mathbf{x} \in \Omega. \tag{3.1}$$

Note that $\mathbf{y}(\cdot)$ denotes a trajectory (path), and not the controls for a specific vehicle to follow that trajectory. As rovers travel with a slow maximum speed (5cm/s), and are capable of turning in place [56], it is reasonable to assume that controls for a rover in $\mathbb{R}^2$ to follow the path in (3.1) can be found.

The control problem is to find a control $\mathbf{u}(\cdot) \in \mathcal{U}$ that plans a path from a start position, $\mathbf{x}_0 \in \Omega$ to a final position, $\mathbf{x}_f \in \Omega$ under the system (3.1). A trajectory $\mathbf{y}(\cdot)$ with initial point $\mathbf{x}_0$ may be written $\mathbf{y}_{\mathbf{x}_0}(\cdot)$.

Definition 3.1.1. *The* ***exit-time*** $T : \Omega \times \mathcal{U} \to \mathbb{R}_+$ *is*

$$T(\boldsymbol{x}_0, \boldsymbol{u}(\cdot)) = \inf\{t \in \mathbb{R}_+ |\ \boldsymbol{y}_{\boldsymbol{x}_0}(t) = \boldsymbol{x}_f\}.$$

The exit-time is the first time the vehicle reaches $\mathbf{x}_f$ (from $\mathbf{x}_0$) under the influence of the control $\mathbf{u}(\cdot)$.

Definition 3.1.2. *The* ***cost function*** $Cost : \overline{\Omega} \times \mathcal{U} \to \mathbb{R}_+$ *is*

$$Cost(\boldsymbol{x}_0, \boldsymbol{u}(\cdot)) = \int_0^{T(\boldsymbol{x}_0, \boldsymbol{u}(\cdot))} g(\boldsymbol{y}_{\boldsymbol{x}_0}(s), \boldsymbol{u}(s))ds, \tag{3.2}$$

where the ***weight function*** $g : \overline{\Omega} \times \mathbb{S}^{n-1} \to \mathbb{R}_+$ *is continuous.*

It is assumed that the weight function g is continuous and satisfies $g(\mathbf{x}, \mathbf{u}) > 0$ for all $\mathbf{x} \in \overline{\Omega}$ and $\mathbf{u} \in \mathbb{S}^{n-1}$. Since $\overline{\Omega} \times \mathbb{S}^{n-1}$ is a compact set, there exists $G_{min}, G_{max} \in \mathbb{R}_+$

$$0 < G_{min} < g_{min}(\mathbf{x}) \leq g(\mathbf{x}, \mathbf{u}) \leq g_{max}(\mathbf{x}) < G_{max} < \infty, \text{ for every } (\mathbf{x}, \mathbf{u}) \in \overline{\Omega} \times \mathbb{S}^{n-1}, \tag{3.3}$$

where $g_{min}(\mathbf{x}) = \min\limits_{\mathbf{u} \in \mathbb{S}^{n-1}} g(\mathbf{x}, \mathbf{u})$ and $g_{max}(\mathbf{x}) = \max\limits_{\mathbf{u} \in \mathbb{S}^{n-1}} g(\mathbf{x}, \mathbf{u})$.

The environment is assumed to be known and weighted with g. The case $g(\mathbf{x}, \mathbf{u}) \equiv 1$, for all $(\mathbf{x}, \mathbf{u}) \in \overline{\Omega} \times \mathbb{S}^{n-1}$ corresponds to the optimal control problem for shortest time (and hence shortest path, since the vehicle travels at constant speed).

Definition 3.1.3. *The control* $\boldsymbol{u}^*(\cdot) \in \mathcal{U}$ *is* ***optimal*** *if it minimizes the cost function (3.2) subject to (3.1).*

Definition 3.1.4. *The* ***value function*** $V : \overline{\Omega} \to \mathbb{R}$ *at* $\boldsymbol{x} \in \overline{\Omega}$ *is the cost associated with the optimal control* $\boldsymbol{u}^*(\cdot) \in \mathcal{U}$ *for reaching the final position* $\boldsymbol{x}_f$ *from* $\boldsymbol{x}$*,*

$$V(\boldsymbol{x}) = \inf_{\boldsymbol{u}(\cdot) \in \mathcal{U}} Cost(\boldsymbol{x}, \boldsymbol{u}(\cdot)) = Cost(\boldsymbol{x}, \boldsymbol{u}^*(\cdot)). \tag{3.4}$$

The control problem is often referred to as static since it is irrelevant of starting time and the weight does not change with respect to time. The value function satisfies the following continuous *Dynamic Programming Principle* (DPP).

Theorem 3.1.5. *[26, Section 10.3.2, Theorem 1] For every* $h > 0$*,* $t \geq 0$*, such that* $0 \leq t + h \leq T(\boldsymbol{x}, \boldsymbol{u}^*(\cdot))$*,*

$$V(\boldsymbol{y}_{\boldsymbol{x}}(t)) = \inf_{\boldsymbol{u}(\cdot) \in \mathcal{U}} \left\{ \int_t^{t+h} g(\boldsymbol{y}_{\boldsymbol{x}}(\tau), \boldsymbol{u}(\tau))d\tau + V(\boldsymbol{y}_{\boldsymbol{x}}(t+h)) \right\}. \tag{3.5}$$

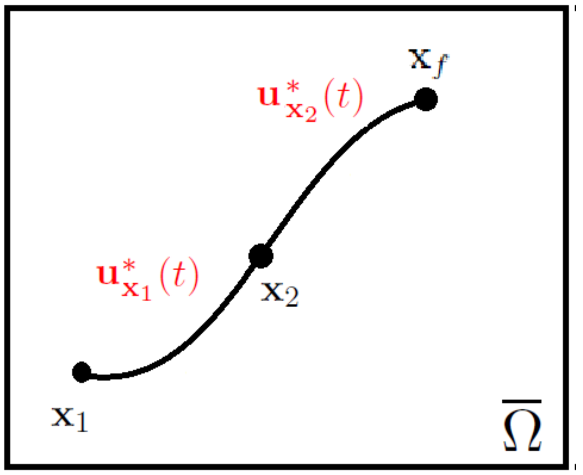

$\mathbf{x}_f$
$\mathbf{u}^*_{\mathbf{x}_2}(t)$
$\mathbf{u}^*_{\mathbf{x}_1}(t)$
$\mathbf{x}_2$
$\mathbf{x}_1$
$\overline{\Omega}$

Figure 3.1: Continuous Dynamic Programming Principle for $\overline{\Omega} \subset \mathbb{R}^2$ - For any $\mathbf{x}_2$ on the optimal path from $\mathbf{x}_1$ to $\mathbf{x}_f$, the portion of the path from $\mathbf{x}_1$ to $\mathbf{x}_2$ and $\mathbf{x}_2$ to $\mathbf{x}_f$ must also be optimal. As well, the concatenation of optimal paths from $\mathbf{x}_1$ to $\mathbf{x}_2$ and from $\mathbf{x}_2$ to $\mathbf{x}_f$ must be an optimal path from $\mathbf{x}_1$ to $\mathbf{x}_f$ on $\overline{\Omega}$.

The continuous DPP over the space $\overline{\Omega}$ states that a control is optimal between two states if and only if the same control is optimal over all intermediate states along the trajectory. See Figure 3.1. The value function $V(\mathbf{x})$ can be regarded as the lowest cost for $\mathbf{x} \in \overline{\Omega}$ to reach the point $\mathbf{x}_f$.

A partial differential equation in terms of the value function can be informally obtained [83]. Let the control $\mathbf{u}^*(\cdot)$ and trajectory $\mathbf{y}^*(\cdot)$ be optimal for (3.1) and (3.2).

$$\begin{aligned} V(\mathbf{y}^*(t)) &= \int_t^{t+h} g(\mathbf{y}^*(\tau), \mathbf{u}^*(\tau))d\tau + V(\mathbf{y}^*(t+h)) \\ 0 &\approx \int_t^{t+h} g(\mathbf{y}^*(t), \mathbf{u}^*(t))d\tau + V(\mathbf{y}^*(t+h) - V(\mathbf{y}^*(t)) \\ 0 &\approx g(\mathbf{y}^*(t), \mathbf{u}^*(t)) + \frac{V(\mathbf{y}^*(t+h)) - V(\mathbf{y}^*(t))}{h}. \end{aligned}$$

The static Hamilton-Jacobi-Bellman (HJB) is obtained in the limit as $h \to 0$.

$$\min_{\mathbf{u}\in\mathbb{S}^{n-1}} \{(\nabla V(\mathbf{x}) \cdot \mathbf{u}) + g(\mathbf{x}, \mathbf{u})\} = 0, \mathbf{x} \in \Omega \backslash \{\mathbf{x}_f\}, \tag{3.6}$$

$$V(\mathbf{x}_f) = 0,$$

where $\cdot$ in the above equation denotes the dot product in $\mathbb{R}^n$. The HJB (3.6) is often referred to as a source-point problem, for which all paths emanate from the single source $\mathbf{x}_f$. The Hamiltonian $H : (\Omega\backslash\{\mathbf{x}_f\}) \times \mathbb{R}^n \to \mathbb{R}$ for this problem is

$$H(\mathbf{x}, \mathbf{p}) = -\min_{\mathbf{u}\in\mathbb{S}^{n-1}} \{\mathbf{p} \cdot \mathbf{u} + g(\mathbf{x}, \mathbf{u})\}. \tag{3.7}$$

The optimal control is synthesized using the value function V [26, Chapter 10.3]. Combining (3.1) and (3.6),

$$\dot{\mathbf{y}}(t) = \mathbf{u}^*(t) = \arg\min_{\mathbf{u}\in\mathbb{S}^{n-1}}\{\nabla V(\mathbf{y}(t)) \cdot \mathbf{u} + g(\mathbf{y}(t), \mathbf{u})\}, \mathbf{y}(0) = \mathbf{x}_0, \mathbf{x}_0 \in \Omega \tag{3.8}$$

If the weight function is of the form $g(\mathbf{x}, \mathbf{u}) = g(\mathbf{x})$, then the HJB equation (3.6) reduces to the simpler Eikonal equation,

$$\|\nabla V(\mathbf{x})\| = g(\mathbf{x}), \tag{3.9}$$

with minimizing control

$$\mathbf{u}^*(t) = -\frac{\nabla V(\mathbf{y}(t))}{\|\nabla V(\mathbf{y}(t))\|}.$$

If the weight function g has dependence on direction $\mathbf{u}$, then the minimizing direction $\mathbf{u}^*$ (3.8) must be solved as a minimization problem.

Definition 3.1.6. *The **characteristic direction** $\boldsymbol{u}^* : \Omega \to \mathbb{S}^{n-1}$ of $\boldsymbol{x} \in \Omega$ is the optimizer of (3.7),*

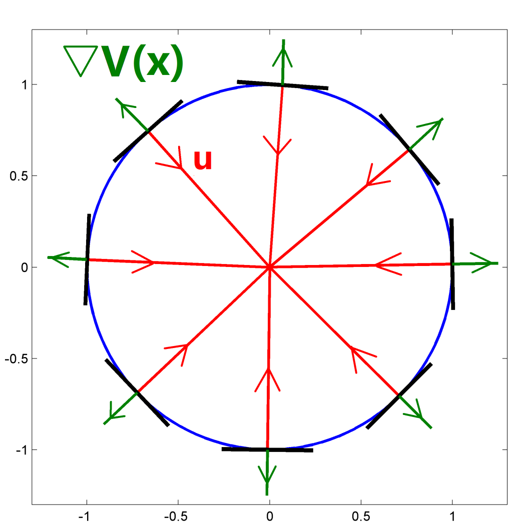
$$\boldsymbol{u}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{u}\in\mathbb{S}^{n-1}}\{\nabla V(\boldsymbol{x}) \cdot \boldsymbol{u} + g(\boldsymbol{x}, \boldsymbol{u})\}. \tag{3.10}$$

Definition 3.1.7. *The **speed profile** at $\boldsymbol{x}$ is*

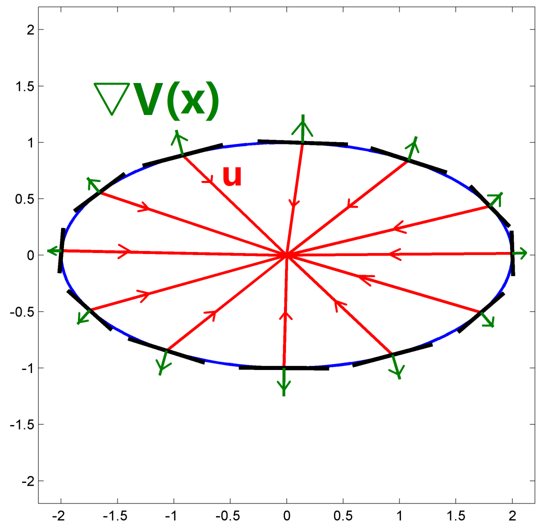
$$\mathcal{U}_g(\boldsymbol{x}) = \{\boldsymbol{u}/g(\boldsymbol{x}, \boldsymbol{u})|\ \boldsymbol{u} \in \mathbb{S}^{n-1}\}. \tag{3.11}$$

In $\mathbb{R}^2$, the speed profile is the shape traced out at $\mathbf{x}$ in polar coordinates with the speed $1/g(\mathbf{x}, \mathbf{u})$ as the radius and the angle corresponding to the direction $\mathbf{u}$ (see Figure 3.2). The minimizing characteristic direction $\mathbf{u}^*$ at $\mathbf{x}$ is unique provided that $\nabla V(\mathbf{x})$ is defined and the speed profile $\mathcal{U}_g(\mathbf{x})$ is convex [72]. If g does not depend on direction $\mathbf{u}$, then the speed profile is circular. See Figure 3.2.

Definition 3.1.8. *Let $\Gamma = \frac{G_{max}}{G_{min}}$, (where G_{min}, and G_{max} are defined in (3.3)) denote the **global anisotropy coefficient**.*



(a) The direction-independent speed profile in $\mathbb{R}^2$ is circular as the weight g does not change as direction $\mathbf{u}$ is varied. The gradient $\nabla V(\mathbf{x})$ and the characteristic direction $\mathbf{u}$ are collinear.



(b) A direction-dependent speed profile in $\mathbb{R}^2$, the weight g changes as the direction $\mathbf{u}$ is varied. The speed profile does not necessarily have to be elliptical, nor symmetrical about $\mathbf{x}$. Note the characteristic $\mathbf{u}$ is not necessarily collinear to $\nabla V(\mathbf{x})$.

Figure 3.2: Speed Profile at $\mathbf{x} \in \mathbb{R}^2$, $\mathcal{U}_g = \{\mathbf{u}/g(\mathbf{x}, \mathbf{u}) | \mathbf{u} \in \mathbb{S}^1\}$. The gradient $\nabla V(\mathbf{x})$ points outwards, perpendicular to the profile, while the corresponding characteristic $\mathbf{u}$ points inwards towards $\mathbf{x}$.

A bound on the angle between the characteristic direction $\mathbf{u}^*$ and gradient direction $\nabla V(\mathbf{x}_0)$ at $\mathbf{x}_0$ has been proven.

Lemma 3.1.9. *[72] Assume $\nabla V(\boldsymbol{x})$ exists. If $\boldsymbol{u}^*$ optimizes (3.6) at $\boldsymbol{x}$, and the weight $g(\boldsymbol{x}, \boldsymbol{u})$ satisfies the restrictions of (3.3), then*

$$\frac{\nabla V}{\|\nabla V\|} \cdot \boldsymbol{u}^* \leq -\Gamma^{-1}. \tag{3.12}$$

Proof. According to (3.6), (3.8), $\mathbf{u}^*$ satisfies

$$\nabla V \cdot \mathbf{u}^* + g(\mathbf{x}, \mathbf{u}^*) = 0.$$

Since $g(\mathbf{x}, \mathbf{u}) > 0$ for all $(\mathbf{x}, \mathbf{u}) \in \overline{\Omega} \times \mathbb{S}^{n-1}$, $\nabla V \cdot \mathbf{u}^* < 0$. Thus

$$0 = \frac{\nabla V \cdot \mathbf{u}^*}{g(\mathbf{x}, \mathbf{u}^*)} + 1 \leq \frac{\nabla V \cdot \left(-\frac{\nabla V}{\|\nabla V\|}\right)}{g(\mathbf{x}, -\frac{\nabla V}{\|\nabla V\|})} + 1 \leq -\frac{\|\nabla V\|}{G_{max}} + 1,$$

and hence

$$\nabla V \cdot \mathbf{u}^* \leq -\frac{\|\nabla V\| \, g(\mathbf{x}, \mathbf{u}^*)}{G_{max}} \leq -\frac{\|\nabla V\| \, G_{min}}{G_{max}} = -\|\nabla V\| \, \Gamma^{-1}. \square$$

Even for smooth g, the value function V may not be differentiable everywhere. A weaker sense of solution will be presented.

3.2 Viscosity Solutions

The static Hamilton-Jacobi equation for a general Hamiltonian $H : \Omega \times \mathbb{R}^n \to \mathbb{R}$, and boundary function $q : \partial\Omega \to \mathbb{R}$ is

$$H(\mathbf{x}, \nabla V) = 0, \mathbf{x} \in \Omega \tag{3.13}$$
$$V(\mathbf{x}) = q(\mathbf{x}), \mathbf{x} \in \partial\Omega$$

There are two common formulations of (3.13). In the first, the "boundary" $\partial\Omega$ is defined at a single point (or as a target set) on the interior of Ω and the solution V is found outwards on bounded subsets of $\mathbb{R}^n$. An example is (3.6) where $H(\mathbf{x}, \mathbf{p}) = -\min_{\mathbf{u} \in \mathbb{S}^{n-1}}\{\mathbf{p} \cdot \mathbf{u} + g(\mathbf{x}, \mathbf{u})\}$, $q(\mathbf{x}) = 0$ and $\partial\Omega = \{\mathbf{x}_f\}$.

The other formulation is the problem of $q(\mathbf{x})$ defined on the outer boundary $\partial\Omega$ of a bounded region in $\mathbb{R}^n$ and solving for V on the interior of that region. In this thesis, the two will be distinguished by calling the former the source-point problem and the latter the boundary value problem. The latter will be revisited in Chapter 5.

Even for smooth H, $\partial\Omega$ and q, the solution to (3.13) may not be differentiable on all of Ω. Viscosity solutions [17] are used to uniquely describe the solutions of HJB equations in a weak sense.

A similar discussion as the following on viscosity solutions can be found in [14, 26]. Let $\mathbf{x} \in \Omega = (-1, 1) \subset \mathbb{R}$, and the Hamiltonian of a system be

$$H(\mathbf{x}, \mathbf{p}) = |\mathbf{p}| - 1.$$

The static HJB equation $H(\mathbf{x}, V') = 0$ is

$$|V'(\mathbf{x})| = 1 \tag{3.14}$$

and boundary condition $q : \partial\Omega \to \mathbb{R}$ is given by $q(-1) = 0$ and $q(1) = 0$.

The only differentiable solutions of (3.14) on $(-1, 1)$ are $\mathbf{x} + \mathbf{b}$ and $-\mathbf{x} + \mathbf{b}$, $\mathbf{b} \in \mathbb{R}$, but neither of these solutions can simultaneously satisfy both boundary conditions. Hence there are no differentiable solutions that satisfy (3.14) on $[-1, 1]$. A more general sense of solution is considered. Let A be a closed and bounded subset of $\mathbb{R}^n$.

Definition 3.2.1. *A function $f : A \to \mathbb{R}$ is **Lipschitz-continuous** if there exists $C \in \mathbb{R}_+$ such that*

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq C \left\| \boldsymbol{x} - \boldsymbol{y} \right\|$$

for all $\boldsymbol{x}, \boldsymbol{y} \in A$.

Definition 3.2.2. *A **weak solution** to (3.13) is a Lipschitz-continuous function $V : \overline{\Omega} \to \mathbb{R}$ that satisfies (3.13) almost everywhere.*

A weak solution can have countably many points where ∇V is not defined. There are infinitely many weak solutions to (3.14), see Figure 3.3. To discuss a particular weak solution that has a particular connection to the optimal control problem presented previously, the following definitions are considered.

Let $C^\infty(\Omega)$ denote the space of infinitely continuously-differentiable functions on Ω. This will be the space of test functions used in the definition of viscosity solutions. The following definitions can be found in [8].

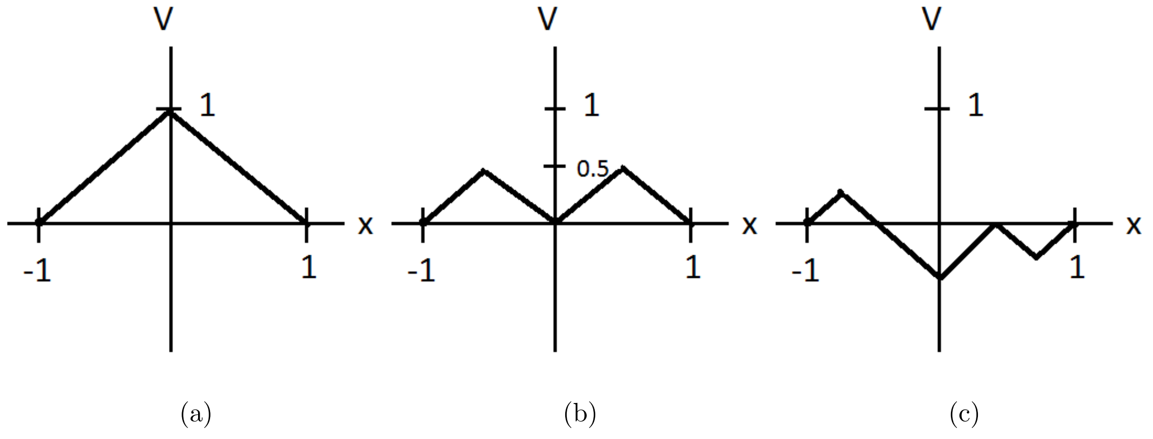


Figure 3.3: Weak solutions of (3.14). The left figure (a) is the viscosity solution of (3.14). The other two (b),(c) weak solutions are eliminated in the definition of viscosity solutions (Definitions 3.2.3 and 3.2.4).

Definition 3.2.3. *A function $\underline{V} : \overline{\Omega} \to \mathbb{R}$ is a **viscosity subsolution** of (3.13) if for any $\phi \in C^\infty(\Omega)$,*

$$H(\boldsymbol{x}_0, \nabla\phi(\boldsymbol{x}_0)) \le 0, \tag{3.15}$$

at any local maximum point $\boldsymbol{x}_0 \in \Omega$ of $\underline{V} - \phi$. See Figure 3.4a.

Definition 3.2.4. *A function $\overline{V} : \overline{\Omega} \to \mathbb{R}$ is a **viscosity supersolution** of (3.13) if for any $\phi \in C^\infty(\Omega)$,*

$$H(\boldsymbol{x}_0, \nabla\phi(\boldsymbol{x}_0)) \ge 0, \tag{3.16}$$

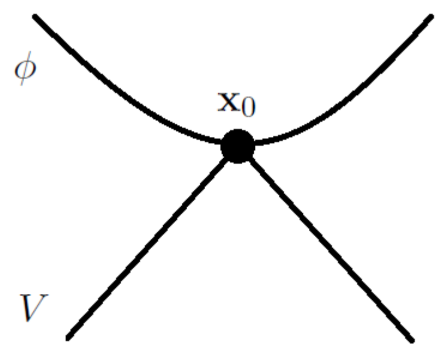
at any local minimum point $\boldsymbol{x}_0 \in \Omega$ of $\overline{V} - \phi$. See Figure 3.4b.

Definition 3.2.5. *A **viscosity solution** of the HJB (3.13) is both a viscosity subsolution and a viscosity supersolution of (3.13).*

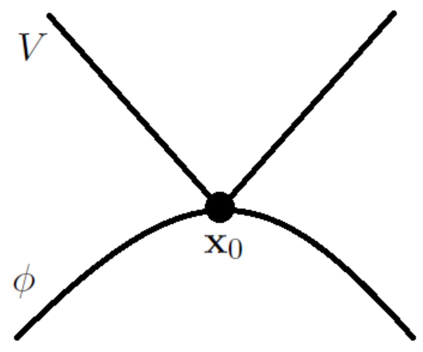
The weak solutions of (3.14) are now examined. Suppose a weak solution V_0 of (3.14) has a local minimum (for example Figure 3.3b and c) at $\mathbf{x}_0 \in (-1, 1)$. Let $\phi(\mathbf{x}) = -(\mathbf{x} - \mathbf{x}_0)^2$, then $V_0 - \phi$ has a local minimum at $\mathbf{x}_0$ (see Figure 3.4). On the other hand,

$$H(\mathbf{x}_0, \phi'(\mathbf{x}_0)) = |\phi'(\mathbf{x}_0)| - 1 = 0 - 1 < 0.$$

Hence V_0 does not satisfy the definition of a viscosity supersolution, which implies V_0 is not a viscosity solution. Note the only remaining weak solution is $V(\mathbf{x}) = 1 - |\mathbf{x}|$ which



(a) Viscosity Subsolution, $\underline{V}(\mathbf{x}) \leq \phi(\mathbf{x})$, $V(\mathbf{x}_0) = \phi(\mathbf{x}_0)$ ($V - \phi$ has a local maximum at $\mathbf{x}_0$), $H(\mathbf{x}_0, \nabla\phi(\mathbf{x}_0)) \leq 0$



(b) Viscosity Supersolution, $\overline{V}(\mathbf{x}) \geq \phi(\mathbf{x})$, $V(\mathbf{x}_0) = \phi(\mathbf{x}_0)$ ($V - \phi$ has a local minimum at $\mathbf{x}_0$), $H(\mathbf{x}_0, \nabla\phi(\mathbf{x}_0)) \geq 0$

Figure 3.4: Viscosity solutions at a point $\mathbf{x}_0$ such that $\nabla V(\mathbf{x}_0)$ is not defined.

can easily be verified to be a viscosity solution. The viscosity solution definition isolated a particular weak solution, which is in fact the value function (3.4) of the optimal control problem (3.6) [8, 26]. Note that if instead $H(\mathbf{x}, \mathbf{p}) = 1 - |\mathbf{p}|$, then the viscosity solution would be $V(\mathbf{x}) = |\mathbf{x}| - 1$.

Some standard definitions of viscosity solutions use $C^1(\Omega)$ as the space of test functions, for example [8], while others use $C^\infty(\Omega)$ [26]. It was shown in [18] that the space of test functions $C^1(\Omega)$ could be replaced with $C^\infty(\Omega)$ for an equivalent definition.

For general Hamiltonians, existence results are established using the vanishing viscosity argument [26], while uniqueness results for the viscosity solution V in (3.6) come from a comparison principle argument [8, Theorem IV.2.6].

3.3 Local Minina in Viscosity Solution to HJB

Recall that in Section 2.1, the artificial potential function [45] was used as a navigation function to find a path. The artificial potential function could encounter local minima, see Figure 2.2. The value function V (3.4), which is the viscosity solution of the HJB (3.6), is used as a navigation function in (3.8). It will be shown that V cannot have local minima on Ω.

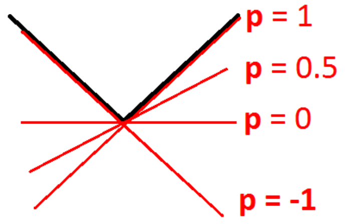


Figure 3.5: Several subgradients are shown for the function $f(\mathbf{x}) = |\mathbf{x}|$, $\mathbf{x} \in \mathbb{R}$. At $\mathbf{x} = 0$, the set of all subgradients is $D^- f(0) = [-1, 1]$.

Some definitions are required. Let $B_\delta(\mathbf{x})$ define the open ball around $\mathbf{x}$ with radius δ.

Definition 3.3.1. *The vector $\boldsymbol{p} \in \mathbb{R}^n$ is a **subgradient** of a function $f : \Omega \to \mathbb{R}$ at $\boldsymbol{x}_0 \in \Omega$ if there exists $\delta > 0$ such that for any $\boldsymbol{x} \in B_\delta(\boldsymbol{x}_0)$,*

$$f(\boldsymbol{x}) - f(\boldsymbol{x}_0) \geq \boldsymbol{p} \cdot (\boldsymbol{x} - \boldsymbol{x}_0).$$

Let $D^- f(\boldsymbol{x}_0)$ denote the set of all subgradients of f at $\boldsymbol{x}_0$.

Definition 3.3.2. *The vector $\boldsymbol{p} \in \mathbb{R}^n$ is a **supergradient** of a function $f : \Omega \to \mathbb{R}$ at $\boldsymbol{x}_0 \in \Omega$ if there exists $\delta > 0$ such that for any $\boldsymbol{x} \in B_\delta(\boldsymbol{x}_0)$,*

$$f(\boldsymbol{x}) - f(\boldsymbol{x}_0) \leq \boldsymbol{p} \cdot (\boldsymbol{x} - \boldsymbol{x}_0).$$

Let $D^+ f(\boldsymbol{x}_0)$ denote the set of all supergradients of f at $\boldsymbol{x}_0$.

If $\nabla f(\mathbf{x}_0)$ exists, $D^- f(\mathbf{x}_0) = \{\nabla f(\mathbf{x}_0)\}$. For example, consider $f(\mathbf{x}) = |\mathbf{x}|$, $\mathbf{x} \in \mathbb{R}$. See Figure 3.5. At $\mathbf{x} = 0$, $D^- f(0) = [-1, 1]$. A line with slope $2 \notin D^- f(0)$ touching $f(0)$ would cross f locally. Definition 3.3.1 would not be satisfied. Away from $x = 0$, $D^- f(0) = f'(0)$.

Lemma 3.3.3. *The point $\boldsymbol{x}_0 \in \Omega$ is a local minimum of f if and only if $\boldsymbol{0}$ is a subgradient of f at $\boldsymbol{x}_0$.*

The proof is a direct application of Definition 3.3.1.

The following lemma relates the set $D^- f(\mathbf{x}_0)$ to values of $\nabla \phi(\mathbf{x}_0)$ of the test functions in the viscosity supersolution definition given previously.

Lemma 3.3.4. *[8, Lemma II.1.7] A vector $\boldsymbol{p} \in D^- f(\boldsymbol{x}_0)$ if and only if there exists $\phi \in C^1(\Omega) \to \mathbb{R}$ such that $\nabla \phi(\boldsymbol{x}_0) = \boldsymbol{p}$, and $f - \phi$ has a local minimum at $\boldsymbol{x}_0$. Similarly, a vector $\boldsymbol{p} \in D^+ f(\boldsymbol{x}_0)$ if and only if there exists $\phi \in C^1(\Omega) \to \mathbb{R}$ such that $\nabla \phi(\boldsymbol{x}_0) = \boldsymbol{p}$, and $f - \phi$ has a local maximum at $\boldsymbol{x}_0$.*

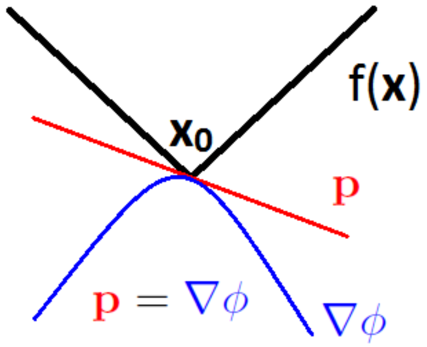


Figure 3.6: Lemma 3.3.4 for $f(\mathbf{x}) = |\mathbf{x}|$ - There exists $\phi(\mathbf{x}) \in C^1(\Omega)$ such that $\phi(\mathbf{x}) < f(\mathbf{x})$ and $\phi(\mathbf{x}_0) = f(\mathbf{x}_0)$ if and only if $\mathbf{p} \in D^- f(\mathbf{x}_0)$. Note that $\mathbf{p}$ is parallel to $\nabla\phi$.

See Figure 3.6 for an illustration for $f(x) = |x|$ where $\mathbf{p}(0) = \nabla\phi(0)$.

Theorem 3.3.5. *If V is a viscosity solution of the HJB equation (3.6) where the weight function satisfies assumption (3.3), then V has no local minima on $\Omega\backslash\partial\Omega$.*

Proof. It will be shown that $\mathbf{x}_0 \in \Omega\backslash\partial\Omega$ is not a local minimum. If $\nabla V(\mathbf{x}_0)$ exists, the proof is even simpler and $\nabla V(\mathbf{x}_0) \neq 0$ can be achieved by directly working with (3.6).

The viscosity solution V is a viscosity supersolution (Definition 3.2.4). If there is no $\phi \in C^\infty(\Omega)$ that satisfies Definition 3.2.4, then by Lemma 3.3.4, $D^- V(\mathbf{x}_0)$ is empty and does not contain $\mathbf{0}$. Hence by Lemma 3.3.3, $\mathbf{x}_0$ is not a local minimum. Now suppose such a $\phi \in C^\infty(\Omega)$ does exist. Hence,

$$H(\mathbf{x}_0, \nabla\phi) \geq 0,$$

or

$$-\min_{\mathbf{u}\in\mathbb{S}^{n-1}} \{\nabla\phi(\mathbf{x}_0) \cdot \mathbf{u} + g(\mathbf{x}_0, \mathbf{u})\} \geq 0.$$

and let $\mathbf{u}^* \in \mathbb{S}^{n-1}$ be the minimizer, where $\mathbb{S}^{n-1} = \{\mathbf{u} \in \mathbb{R}^n | \, \|\mathbf{u}\| = 1\}$

$$-\nabla\phi(\mathbf{x}_0) \cdot \mathbf{u}^* \geq g(\mathbf{x}_0, \mathbf{u}^*) \geq g_{min}(\mathbf{x}_0) > 0,$$

$$\|\nabla\phi(\mathbf{x}_0)\| \geq -\nabla\phi(\mathbf{x}_0) \cdot \mathbf{u}^* > 0.$$

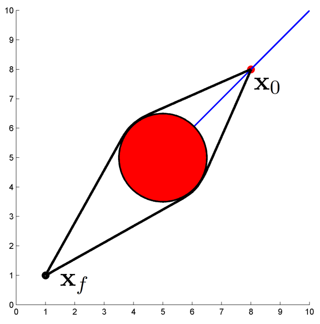


Figure 3.7: Consider the above path planning problem for shortest distance. Local Minima in the second argument of $Cost(\mathbf{x}_0, \mathbf{u}(\cdot))$ (3.2) do not imply minima of value function V at $\mathbf{x}_0$. Any point $\mathbf{x}_0$ on the line will have two cost-minimizing paths to $\mathbf{x}_f$. The value $V(\mathbf{x}_0)$ will be the cost of either path. Such $\mathbf{x}_0$ are not minimum of V.

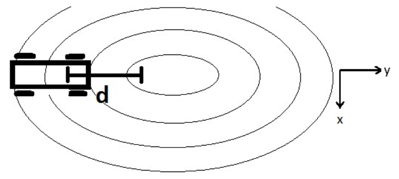
Thus $\nabla\phi(\mathbf{x}_0) \neq \mathbf{0}$ which implies by Lemma 3.3.4 that $\mathbf{0} \notin D^{-}V(\mathbf{x}_0)$. By Lemma 3.3.3, $\mathbf{x}_0$ is not a local minimum of V. Since $\mathbf{x}_0$ was an arbitrary point on $\Omega\backslash\{\mathbf{x}_f\}$, there are no local minima on $\Omega\backslash\{\mathbf{x}_f\}$. $\Box$

From Theorem 3.3.5, the solution to (3.8) will not encounter any local minima on $\Omega\backslash\{\mathbf{x}_f\}$. Local minima in $\mathcal{U}$ may exist for *Cost* at $\mathbf{x}_0$, such as in Figure 3.7 where multiple optimal controls $\mathbf{u}^*(\cdot)$ exist, but this does not imply $\mathbf{x}_0$ is a local minimum of V.

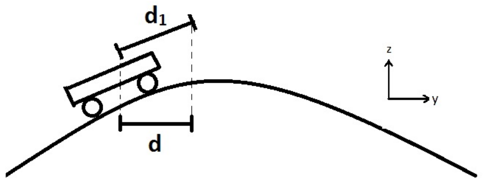
At local minima of navigation functions, no $\mathbf{u}$ exists that optimizes (3.8), since $g(\mathbf{x}, \mathbf{u}) > 0$. The same is true for strict local maxima and saddle points, but perturbing the solution (3.8) off of them will allow the path search to continue. Perturbing the solution off of a local minimum will return it to the minimum. Though local minima do not exist on $\Omega\backslash\partial\Omega$ for V, this does not imply the same result for a numerical approximation of V. The existence of local minima for numerical approximations of V are discussed in the next chapter.

3.4 Rover Environment Modelling

Aspects of the environment experienced by rovers that can be modelled using the weight function g are described in this section. The description will be given in $\mathbb{R}^2$. The weight function $g(\mathbf{x}, \mathbf{u}) \equiv 1$, for all $(\mathbf{x}, \mathbf{u}) \in \overline{\Omega} \times \mathbb{S}^1$ corresponds to the optimal control problem for shortest time (and hence shortest path, since the vehicle travels at constant speed). The shortest path may not always be the best.



(a) Level sets of terrain $z(\mathbf{x})$, two-dimensional view (xy-plane).



(b) Terrain z($\mathbf{x}$), three-dimensional view (yz-plane).

Figure 3.8: Path Length on Terrain - The rover is traversing a hill. The true distance traversed d_1 in the direction of greatest increase is larger than the distance d projected onto the xy-plane.

The optimal path planning problem formulated in Section 3.1 applies to path planning for any autonomous vehicle. Weights relevant to rovers including shortest path on terrain, obstacle avoidance, soil risk, solar energy input and tip-over stability risk are considered.

Path Length on Terrain - The weight $g(\mathbf{x}, \mathbf{u}) \equiv 1$ over all $\overline{\Omega} \times \mathbb{S}^1$ corresponds to the shortest path on $\overline{\Omega}$, however the rover must often traverse uneven terrain.

Definition 3.4.1. *The **terrain** $z : \overline{\Omega} \to \mathbb{R}$ is a continuously differentiable function representing the elevation experienced by the rover on the set $\overline{\Omega}$.*

The rover cannot move freely in the z-direction; its position is constrained to $(\mathbf{x}, z(\mathbf{x}))$ for $\mathbf{x} \in \overline{\Omega}$. Suppose the rover is on a hill. Travelling a small distance d in the direction of greatest increase of z on the xy-plane (since $\overline{\Omega} \subset \mathbb{R}^2$) would represent a greater distance d_1 on the terrain (see Figure 3.8). The shortest path problem on terrain when solved on $\overline{\Omega} \subset \mathbb{R}^2$ is dependent on direction. If the problem is solved directly on a three-dimensional mesh approximating the terrain $z(\mathbf{x})$, then the distance travelled (on the tangent plane) is the same in all directions. The workspace is extended to $\widetilde{\Omega} = \{(\mathbf{x}, z(\mathbf{x})) | \mathbf{x} \in \overline{\Omega}\}$. The weight $g(\mathbf{x}) = 1$ solved on $\widetilde{\Omega}$ is independent of direction.

Obstacles - Obstacles are regions in $\overline{\Omega}$ where travel is forbidden. These include impassable regions or high-risk zones such as quicksand. Recall $O \subset \overline{\Omega}$ is the set of obstacles. An obvious weight function is to use a large value M_O for obstacles, and 0 outside of obstacles. A buffer zone where the danger decreases gradually away from the obstacle can be included. Obstacle weights are independent of direction. Another method to model obstacles is to remove their corresponding regions from $\overline{\Omega}$.

Soil Risk - The soil risk is a measure of the difficulty to traverse a type of soil. For example, rocky soils are easier to traverse than soft sand. The weight can be independent of direction providing only a value for each type of soil. A weight that considers the risk associated with the slope experienced on a particular soil is dependent on direction.

Solar Energy - Energy absorbed by a solar panel is a source of power for rovers. For a panel parallel to the frame of the rover, the weight is independent of direction. The normal vector of the panel is $(-\nabla z(\mathbf{x}), 1)$. The sun, located at (x_s, y_s, z_s), will be at the same angle to the panel for a position $\mathbf{x}$ regardless of direction. The vector from the rover to the sun is

$$\mathbf{s}(\mathbf{x}) = (x_s, y_s, z_s) - (\mathbf{x}, z(\mathbf{x})),$$

while the angle between the vector normal to the tangent plane at $z(\mathbf{x})$ and $\mathbf{s}(\mathbf{x})$ is

$$\beta(\mathbf{x}) = \arccos\left(\frac{\mathbf{s}(\mathbf{x}) \cdot (-\nabla z(\mathbf{x}), 1)}{\|\mathbf{s}(\mathbf{x})\| \, \|(-\nabla z(\mathbf{x}), 1)\|}\right). \tag{3.17}$$

A smaller angle $\beta(\mathbf{x})$ will result in more direct sunlight reaching the panel (see Figure 3.9). If $\beta(\mathbf{x})$ is obtuse, then no direct sunlight will reach the panel. If the panel is not parallel to the rover frame, the weight becomes dependent on the direction in which the rover is facing.

Stability tip-over risk - The risk due to tip-over is dependent on direction. The rover is in a stable position if the downward projection of its centre of gravity onto the terrain lies within the convex hull defined by the rover's contact points (see Figure 3.10). Assume that the convex hull of the contact points made by the rover on the terrain is rectangular. A common model for rovers known as the rocker-bogie model [28] satisfies this assumption. The Mars rovers Spirit, Opportunity and Curiosity use the rocker-bogie model. Assume also that the centre of mass is at its centroid.

A suitable measure of tip-over risk is the force-angle stability margin described in [63]. Suppose the rover is at a position $\mathbf{x}$ facing a direction $\mathbf{u}$. Let the centre of gravity of

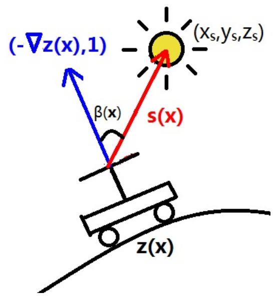


Figure 3.9: Solar energy definitions - The sun is located at (x_s, y_s, z_s). The function $\beta(\mathbf{x})$ is the angle in radians between the normal to the tangent plane $(-\nabla z(\mathbf{x}), 1)$ and the vector between the rover's panel and the sun $\mathbf{s}(\mathbf{x})$.

the rover be $\mathbf{p}_c = (x_c, y_c, z_c)$ and the four corners of the convex hull be labelled clockwise (viewed from above): $\mathbf{p}_i$, $i \in \{1, 2, 3, 4\}$ (see Figure 3.10a). Let the tip-over axes be

$$\mathbf{a}_i = \mathbf{p}_{i+1} - \mathbf{p}_i, \text{ for } i = \{1, 2, 3\}, \; \mathbf{a}_4 = \mathbf{p}_1 - \mathbf{p}_4.$$

and $\hat{\mathbf{a}}_i = \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|}$. The normal vector to tip-over axis $\mathbf{a}_i$ that intersects the centre of gravity is

$$\mathbf{l}_i = (\mathbf{I}_3 - \hat{\mathbf{a}}_i\hat{\mathbf{a}}_i{}^T)(\mathbf{p}_{i+1} - \mathbf{p}_c) \text{ for } i = \{1, 2, 3\}, \mathbf{l}_4 = (\mathbf{I}_3 - \hat{\mathbf{a}}_4\hat{\mathbf{a}}_4{}^T)(\mathbf{p}_1 - \mathbf{p}_c).$$

For $i \in \{1, 2, 3, 4\}$, let $\hat{\mathbf{l}}_i = \frac{\mathbf{l}_i}{\|\mathbf{l}_i\|}$ and let $\hat{\mathbf{f}}_g = [0, 0, -1]$. The force-angle stability measure corresponding to each tip-over axis is

$$\theta_i = \sigma_i \cos^{-1}(\hat{\mathbf{f}}_g \cdot \hat{\mathbf{l}}_i)$$

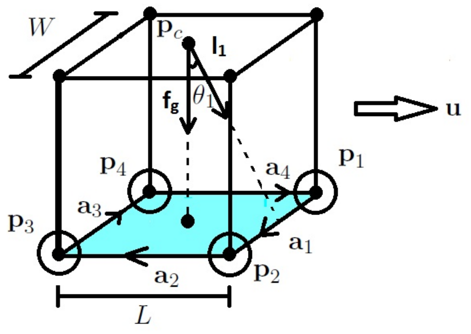
where

$$\sigma_i = \begin{cases} +1 & \text{if } \left(\hat{\mathbf{l}}_i \times \hat{\mathbf{f}}_g\right) \cdot \hat{\mathbf{a}}_i < 0 \\ -1 & \text{otherwise.} \end{cases}$$

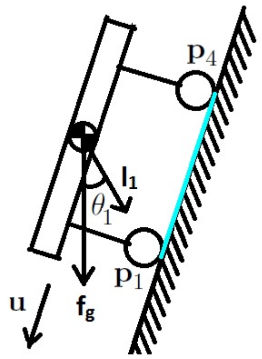
Instability occurs when θ_i is negative, indicating that the vectors $\hat{\mathbf{f}}_g$ and $\hat{\mathbf{l}}_i$ have switched sides (compared to when θ_i was positive), projecting the centre of gravity outside of the convex hull. The overall stability margin α at the position $\mathbf{x}$ with direction $\mathbf{u}$ is given by

$$\alpha(\mathbf{x}, \mathbf{u}) = \min_i \theta_i, \quad i \in \{1, 2, 3, 4\}. \tag{3.18}$$

The aspects of the environment described above will be used for path planning examples in Section 6.2, where the specific weights used will be described.



(a) Convex hull (shaded) of rover contact points with terrain. The rover has length L, and width W. The tip-over stability measure θ_1 is shown for the side $\mathbf{p}_1\mathbf{p}_2$. The vector $\mathbf{f}_g$ is pointing into the convex hull made by the contact points. Hence the stability measure is positive.



(b) A rover is on tilted ground. The projection of the centre of mass falls outside of the convex hull made of the contact points with the terrain, resulting in tip-over. The stability margin is the negative angle θ_1. A negative angle indicates that the orientation of the rover is unstable.

Figure 3.10: Tip-over Stability Risk

3.5 Conclusion

The optimal path planning problem was described as a control problem. An equivalent formulation was presented using the value function from dynamic programming. Even for smooth weights and boundary conditions, the solution to the static HJB equation (3.6) may not be smooth. A weaker sense of solution known as viscosity solutions was presented. It was shown that given a positive weight function, the value function did not possess any local minima that were not global minima. These local minima correspond to points where a path planner solver (3.8) would become stuck, for example with artificial potential functions.

Several rover path planning objectives were discussed including the tip-over stability risk and solar energy absorption. Soil risk, obstacle avoidance and shortest length on terrain were also discussed. These aspects will be explicitly modelled in simulations in Section 6.2. The only restriction on control given to the path planning problem was that it be measurable. This assumption is justified for rover path planning due to the slow maximum speed of the rover and its ability to turn in place. For vehicles where this is not reasonable, additional constraints can be imposed. One method would be to use a more realistic model for dynamics in the path planning problem (3.1). For example, a third dimension can be added to account for the angle of the rover, where the angular velocity is given a different fixed speed, separate from the translational speed. These dynamics along with a constraint on angular velocity based on turning radius is modelled in [78]. Raising the dimensionality of a problem results in a significant increase in computation of approximated solutions. For the rover path planning problem presented here, aspects of the environment are modelled. Travel is limited to either $\Omega \subset \mathbb{R}^2$ or on a surface, which allows for faster computation. Integral constraints as described in [50, 59] can be used to model limited fuel consumption, creating a more realistic model of the path planning problem.

Chapter 4

Approximate Solutions of Static HJB Equations

Analytic solutions to the static HJB (3.6) presented in Chapter 3 are often difficult to obtain, so approximate solutions are computed. The approximate solution is found on a discretization of the workspace. In each of the algorithms described in this chapter, an approximation of the value function $\widetilde{V}$ of V (3.4) is found on each point of the discretization. The solution is linearly interpolated between solved values to obtain a solution on all of Ω. One way to characterize the methods is by the scheme used. The scheme refers to the method in whch the equation used to compute $\widetilde{V}$ is approximated. A computation of the scheme is also known as an update. The following are two types of finite difference schemes used to find $\widetilde{V}$.

Definition 4.0.1. *A **semi-Lagrangian** finite difference scheme approximates the dynamic programming principle (3.5) by numerically approximating the characteristic direction of V.*

Definition 4.0.2. *An **Eulerian** finite difference scheme approximates the static Hamilton-Jacobi equation (3.13) by numerically approximating the gradient of V.*

The above definitions are consistent with the terms in literature. There are two semi-Lagrangian schemes [50] that are commonly used to approximate the dynamic programming principle (3.5). The difference between them is the manner in which the control (3.1) is approximated. In the first, the approximated control described in (3.1) is assumed to be held constant within each element of the discretization. These include the Ordered Upwind Method (OUM) [72], Monotone Acceptance OUM (MAOUM) [5], and Tsitsiklis' algorithm [80]. In the second semi-Lagrangian scheme, the approximated control is assumed to be held constant for a small fixed time $\triangle t$ and can change within an element of the discretization. Algorithms that use the second scheme include the Semi-Lagrangian Fast Marching Method [21] and the Buffered Fast Marching Method [20]. An interpolation from nearby points in the discretization is required to reconstruct the state in both cases. Both semi-Lagrangian schemes can be used to solve direction-dependent problems provided an appropriate algorithm is used: OUM or MAOUM for the first, and Buffered Fast Marching Method for the second.

Examples of Eulerian schemes include the Fast Marching Method (FMM) and Fast Sweeping Method (FSM) for Eikonal equations in [73, 89]. It was shown that a first-order Eulerian and a first-order semi-Lagrangian scheme used in [72] are equivalent for direction-independent problems for a particular type of discretization. The equivalence for problems with direction-dependence was analyzed in [3, 72]. An Eulerian OUM method was presented in [72].

Aside from the scheme used, methods can be characterized by the order in which the solution $\widetilde{V}$ is found on the points of the discretization. Iterative methods that solve HJB equations [8] have existed since the 1980s at the time the theory for viscosity solutions [19] was introduced. A naive iterative method to solve the DPP (3.5) is to use Gauss-Seidel iterations to compute the solution on vertices of the discretization in a certain order, repeating until convergence is achieved. A more efficient iterative method is the Fast Sweeping Method [41]. The approximate solution is computed at vertices in alternating predetermined orderings known as sweeps. The correct value at a vertex is calculated from vertices with already correct values in the direction of the characteristic. The original presentation [41] had first-order accuracy. Schemes with higher-order accuracy exist for FSM, including the Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory schemes, which are higher-order finite-difference schemes. In [85] a fifth-order scheme is used to solve the static HJB in which a first-order accurate solution is used as an initial guess.

Another group of methods differ from iterative methods by decoupling the dependencies of the solution at vertices, solving only for a narrow band of points of the solution at a time. Examples include the Fast Marching Method, which finds an approximate solution to the direction-independent Eikonal equation. Extensions have been proposed to extend the accuracy of FMM to second order [74] by using a (second-order) one-sided update and to third order [1] in two-dimensional problems by locally rotating the stencil by 45° to obtain more points for derivatives. The Fast Marching Method has been extended [47] to solve the geodesic problem on triangular discretizations in both two dimensions and on manifolds. In FMM, the value function is updated in the direction of the gradient.

For direction-independent weights, the gradient direction at a point is collinear with the characteristic direction (3.8).

For problems where the weight is dependent on direction, the characteristics of the PDE are no longer necessarily collinear with the gradient. A bound on the angle between the gradient and characteristic directions can be described based on properties of the weight (Lemma 3.1.9), [72]. Using this fact, algorithms such as Ordered Upwind Method (OUM) [72] and Monotone Acceptance OUM (MAOUM) [5] that search for the characteristic from a larger set of directions have been developed. Both algorithms achieve a similar decoupling of the solution on vertices and can be used to solve convex static HJB equations. In the Buffered Fast Marching Method (BFM), direction-dependent problems are solved by introducing a buffer set, where updates are kept local, iterating the solution within the buffer until convergence is reached.

Prior to discussing algorithms that solve the static HJB, the discretizations of the workspace, namely grids and simplicial meshes will be presented in Section 4.1. In Section 4.2, the FMM using simplicial meshes in $\mathbb{R}^2$ will be described. The OUM will be presented in Section 4.3. An algorithmic improvement of the OUM is proposed in Section 4.3.1. The MAOUM is described in Section 4.4. The Buffered Fast Marching and the Fast Sweeping Method will be briefly discussed in Sections 4.5 and 4.6 respectively. These algorithms will be timed in Section 6.3 on a benchmark problem.

In this chapter, a novel algorithmic improvement for the OUM to include a bi-directional search [66] (OUM-BD) is introduced. A similar approach is applied to FMM in Section 6.2 (and has been briefly described [15]), MAOUM and BFM. All of the algorithms will be presented for the source-point problem, where the approximated value function $\widetilde{V}$ will be built outwards from $\mathbf{x}_f$ with condition $\widetilde{V}(\mathbf{x}_f) = 0$.

4.1 Discretizations of the Workspace

The workspace $\overline{\Omega}$ will be discretized using a simplicial mesh or a grid and an approximation of V (3.5) will be found on the discretization. The approximate solution is denoted $\widetilde{V}$.

4.1.1 Uniform Grids

The region $\overline{\Omega}$ can be discretized using a rectangular grid G, where the solution of (3.6) is found on the vertices.

Definition 4.1.1. *A* ***grid*** *G is a discretization of $\overline{\Omega} \subset \mathbb{R}^n$ using orthogonal elements, where the dimension lengths are $\triangle x_1, ..., \triangle x_n$. Each grid point is represented by coordinates $(k_1 \triangle x_1, ..., k_n \triangle x_n)$ where $\boldsymbol{k} = (k_1, ..., k_n) \in \mathbb{Z}^n$.*

An square grid in $\mathbb{R}^2$ is an example, where all the elements are squares and $\triangle x = \triangle x_1 = \triangle x_2$. When discussing grids, the term vertices may be used interchangeably with grid points.

Grids are simple to implement but have a significant drawback. For the boundary value problem, if $\partial\Omega$ is not a rectangle, then a fine discretization may be required to accurately capture its details. The same is true for weights that vary over Ω. Non-uniform and adaptive grids [5, 73], where elements of different sizes have been used. Uniform square grids are used in the description of the Fast Sweeping Method and Buffered Fast Marching Method and have been implemented in Section 6.3. Another discretization known as a simplicial mesh is presented.

4.1.2 Simplicial Meshes

Simplicial meshes are used to discretize general regions of $\mathbb{R}^n$.

Definition 4.1.2. *A set of points $F = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_k\} \subset \mathbb{R}^n$ is* ***affinely independent*** *if the vectors $\{\boldsymbol{x}_1 - \boldsymbol{x}_0, \ \dots \ , \boldsymbol{x}_k - \boldsymbol{x}_0\}$ are linearly independent.*

Definition 4.1.3. *A* ***k-simplex*** *(plural k-simplices) is the convex hull of an affinely independent set of points $F = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_k\}$.*

Let the notation $\mathbf{x}_0\mathbf{x}_1 \cdots \mathbf{x}_k$ denote a k-simplex defined by the convex hull of F. The empty set is a -1-simplex, a single point $\mathbf{x}_0$ is a 0-simplex. The line segment $\mathbf{x}_0\mathbf{x}_1$ is a 1-simplex. The triangle $\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2$ is a 2-simplex and the tetrahedron $\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ is a 3-simplex. The notation $\mathbf{s}$ is used to denote an arbirary simplex of dimension 1 or higher. The notation $\mathbf{x}_i^{\mathbf{s}}$ is used to represent a vertex of $\mathbf{s}_k = \mathbf{x}_0^{\mathbf{s}}\mathbf{x}_1^{\mathbf{s}} \cdots \mathbf{x}_k^{\mathbf{s}}$. A 0-simplex is often referred to as a vertex, and a 1-simplex an edge.

Definition 4.1.4. *Suppose $\boldsymbol{s}$ is a k-simplex defined by the convex hull of F. An* ***m-face*** *$(-1 \leq m \leq k)$ of the k-simplex $\boldsymbol{s}$ is an m-simplex made of the convex hull of a subset of F containing $m + 1$ vertices.*

A 2-simplex (triangle) has 8 faces: the empty set (-1-face), three vertices (0-face), three edges (1-face), one triangle (2-face). The empty set is a -1-face of all simplices. A k-dimensional simplex has 2^{k+1} faces.

Definition 4.1.5. *A **simplicial mesh**, X is a set of simplices such that*

1. *any face of a simplex in X is also in X,*
2. *the intersection of two simplices $\boldsymbol{s}_1, \boldsymbol{s}_2 \in X$ is a face of X.*

Definition 4.1.6. *A k**-simplicial mesh** is a simplicial mesh where the highest simplex dimension in X is k.*

All lower dimensional faces of simplices in X also belong to X. Any overlap between two simplices in X must be a simplex of X. For a mesh $X \subset \mathbb{R}^2$, the discretization is a 2-simplicial mesh, where none of the triangles overlap except on edges. Similarly, edges only overlap at vertices. A simplicial mesh may simply be called a mesh in the remainder of this work.

Though each simplex is convex, the collection of all simplices in a simplicial mesh as a region is not necessarily convex.

Barycentric coordinates are used to uniquely describe a point $\mathbf{x}$ on the convex hull of a simplex $\mathbf{s}$ relative to its vertices. The set of barycentric coordinates for a k-simplex is

$$\Xi_k = \left\{ (\zeta_0, \zeta_1, ..., \zeta_k) \in \mathbb{R}^{k+1} \middle| \sum_{j=0}^{k} \zeta_j = 1, \zeta_j \in [0,1] \ \forall\ 0 \leq j \leq k \right\}. \tag{4.1}$$

Definition 4.1.7. *The **barycentric coordinates** of $\boldsymbol{x} \in \mathbb{R}^n$ belonging to a k-simplex $\boldsymbol{s}$ defined by the convex hull of F is a vector $\zeta = (\zeta_0, ..., \zeta_k) \in \Xi_k$ such that $\boldsymbol{x} = \sum_{i=0}^{k} \zeta_i \boldsymbol{x}_i^{\boldsymbol{s}}$.*

Definition 4.1.8. *A closed region $D \subset \mathbb{R}^n$ is **contained** in a mesh X if for every $\boldsymbol{x} \in D$, there exists an n-simplex $\boldsymbol{s}$ with vertices $\{\boldsymbol{x}_i^{\boldsymbol{s}}\}_{i=0}^{n} \subset X$ and barycentric coordinates $\zeta = (\zeta_0, ..., \zeta_n) \in \Xi_n$ such that $\boldsymbol{x} = \sum_{i=0}^{n} \zeta_i \boldsymbol{x}_i^{\boldsymbol{s}}$.*

See Figure 4.1. Under the assumptions in the previous definition, the vector $\zeta = (\zeta_0, ..., \zeta_k)$ is unique [72]. The computation of barycentric coordinates is shown for $\mathbb{R}^2$ but can be easily generalized to $\mathbb{R}^n$. For a point $\mathbf{x} \in \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2 \in X$, let $\zeta = (\zeta_0, \zeta_1, \zeta_2)$ such that

$$\mathbf{x} = \zeta_0 \mathbf{x}_0 + \zeta_1 \mathbf{x}_1 + \zeta_2 \mathbf{x}_2 \text{ and } \zeta_0 + \zeta_1 + \zeta_2 = 1.$$

The previous equations can be rearranged to yield the linear transformation $\mathbf{T}$ such that

$$\mathbf{T} \cdot \begin{pmatrix} \zeta_0 \\ \zeta_1 \end{pmatrix} = \mathbf{x} - \mathbf{x}_2, \text{ where } \mathbf{T} = \begin{pmatrix} \mathbf{x}_0 - \mathbf{x}_2, & \mathbf{x}_1 - \mathbf{x}_2 \end{pmatrix}.$$

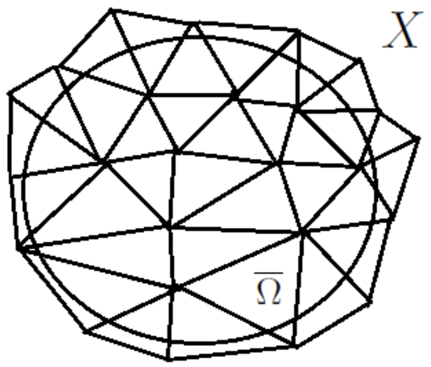


Figure 4.1: $\overline{\Omega}$ contained in X - An example $\overline{\Omega} \subset \mathbb{R}^2$ contained in a (2-simplicial) mesh X.

Because the columns of $\mathbf{T}$ are linearly independent (since F is affinely independent), $\mathbf{T}$ is invertible, and $\zeta = (\zeta_0, \zeta_1, \zeta_2)$ can be recovered from

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \end{pmatrix} = \mathbf{T}^{-1}(\mathbf{x} - \mathbf{x}_2) \text{ and } \zeta_2 = 1 - \zeta_0 - \zeta_1.$$

For any point $\mathbf{x} \in \mathbf{s} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2$, there exists a unique $\zeta = (\zeta_0, \zeta_1, \zeta_2)$ such that

$$\mathbf{x} = \zeta_0\mathbf{x}_0 + \zeta_1\mathbf{x}_1 + \zeta_2\mathbf{x}_2 \text{ where } \zeta_i \geq 0 \text{ and } \zeta_0 + \zeta_1 + \zeta_2 = 1.$$

The following definitions are used to describe properties of the mesh $X \subset \mathbb{R}^n$.

Definition 4.1.9. *The* ***maximum edge length*** h_{max} *of* $X \subset \mathbb{R}^n$ *is the longest edge (*1*-simplex) in* X*.*

Definition 4.1.10. *Let* $1 \leq k \leq n$*. A* ***neighbour*** *of a* $(k-1)$*-simplex in* X, $\boldsymbol{x}_0\boldsymbol{x}_1 \cdots \boldsymbol{x}_{k-1}$, *is any vertex* $\boldsymbol{x}_k \in X$ *such that* $\boldsymbol{x}_0\boldsymbol{x}_1 \cdots \boldsymbol{x}_k$ *forms a* k*-simplex iin* X*.*

Definition 4.1.11. *The* ***minimum simplex height*** h_{min} *of* X *is the shortest of all perpendicular distances between any* $(n-1)$*-simplex of* X *with its neighbours.*

If $n = 2$, then h_{min} is the shortest triangle height out of all triangles in X. Let $X \subset \mathbb{R}^n$ be an n-simplicial mesh with maximum edge length h_{max} and minimum simplex height h_{min}.

A uniform grid in $\mathbb{R}^2$ can be transformed into a simplicial mesh by splitting each rectangular element into two triangles by connecting either diagonal with an edge. For the case of a square grid in $\mathbb{R}^2$, $h_{max} = \sqrt{2}\triangle x$, and $h_{min} = \frac{\sqrt{2}}{2}\triangle x$.

Simplicial meshes can also be used to describe manifolds. For example in $\mathbb{R}^2$, a 2-simplicial mesh X_2 can be given a third coordinate based on a continuous function $\mathbf{z} : \mathbb{R}^2 \to \mathbb{R}$, maintaining the same connectivity with its edges. The augmented 2-simplicial mesh now exists in $\mathbb{R}^3$. Generalizations of simplicial meshes can also be made for closed surfaces, for example spheres and tori.

4.2 Fast Marching Method (FMM)

The Fast Marching Method (FMM) was first introduced by Sethian in [73] to approximate the value function V of the static HJB (3.6) on rectangular grids for problems where the weight g did not depend on direction. For such weights, the static HJB reduces to the Eikonal equation,

$$\begin{aligned} \|\nabla V(\mathbf{x})\| &= g(\mathbf{x}), \mathbf{x} \in \Omega, \\ V(\mathbf{x}) &= q(\mathbf{x}), \mathbf{x} \in \partial\Omega. \end{aligned} \tag{4.2}$$

The algorithm was extended to 2-simplicial meshes on $\mathbb{R}^2$ and manifolds [47], which are discussed here. In the source-point problem, the solution is computed outwards from the point $\mathbf{x}_f$, starting with the condition $\widetilde{V}(\mathbf{x}_f) = 0$. Recall that $\widetilde{V}(\mathbf{x}_i)$ is the approximate lowest cost to reach $\mathbf{x}_f$ from $\mathbf{x}_i$.

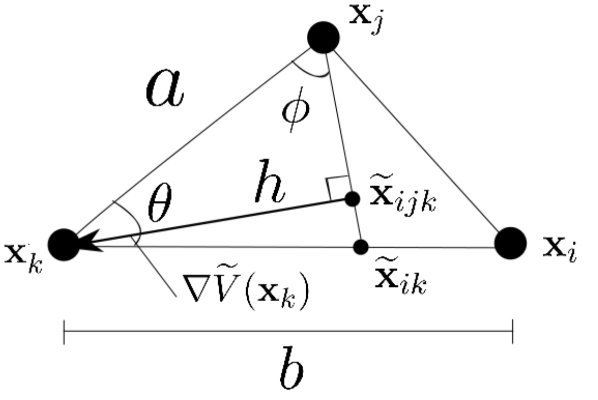
The same labels for vertices used in Dijkstra's algorithm from Section 2.3 are used. See Figure 4.3.

Far - The set of vertices in X where computation of $\widetilde{V}$ has not yet begun. These vertices have tentative values $\widetilde{V}(\mathbf{x}_i) = K$, where K is a large number.

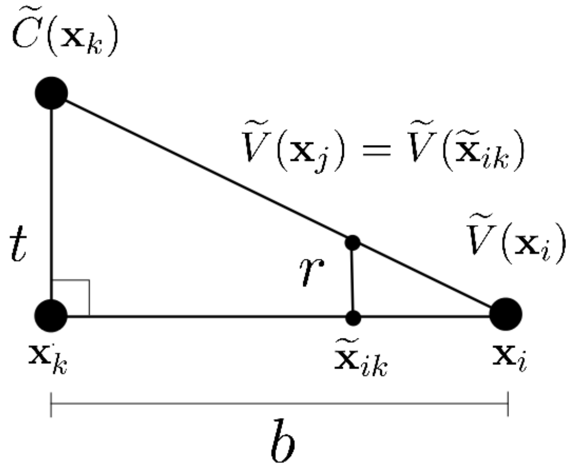
Considered - The set of vertices where $\widetilde{V}$ is being computed, but has yet to be finalized and $\widetilde{V} < K$. The computations are performed using an update function $\widetilde{C} : \{\mathbf{x}_i\} \in X \to \mathbb{R}_+$

Accepted - The set of vertices where $\widetilde{V}(\mathbf{x}_i)$ has been finalized.

The vertices are first labelled *Far*, then relabelled *Considered* and finally relabelled *Accepted*. The algorithm is complete when all the vertices of the mesh are labelled *Accepted*. These vertices labelled *Considered* must have at least one neighbour labelled *Accepted*.


$\mathbf{x}_j$
a
ϕ
θ
h
$\widetilde{\mathbf{x}}_{ijk}$
$\mathbf{x}_k$
$\mathbf{x}_i$
$\nabla\widetilde{V}(\mathbf{x}_k)$
$\widetilde{\mathbf{x}}_{ik}$
b

(a) The value at $\mathbf{x}_k$ is being updated from vertices $\mathbf{x}_i, \mathbf{x}_j$ labelled *Accepted*, with $\widetilde{V}(\mathbf{x}_j) \geq \widetilde{V}(\mathbf{x}_i)$.


$\widetilde{C}(\mathbf{x}_k)$
$\widetilde{V}(\mathbf{x}_j) = \widetilde{V}(\widetilde{\mathbf{x}}_{ik})$
t
r
$\widetilde{V}(\mathbf{x}_i)$
$\mathbf{x}_k$
$\widetilde{\mathbf{x}}_{ik}$
$\mathbf{x}_i$
b

(b) A tilted the triangle of Figure 4.2a using vertices $(\mathbf{x}_i, \widetilde{V}(\mathbf{x}_i))$, $(\mathbf{x}_j, \widetilde{V}(\mathbf{x}_j))$ and $(\mathbf{x}_k, \widetilde{C}(\mathbf{x}_k))$. The update $\widetilde{C}(\mathbf{x}_k)$ is found by solving for t.

Figure 4.2: The FMM update from an acute angle

An approximation of (4.2) known as an update formula for FMM will now be described for acute triangles. The value $\widetilde{V}$ for vertices labelled *Considered* is computed from vertices labelled *Accepted* using a function $\widetilde{C} : X \to \mathbb{R}$, and updated if $\widetilde{C}(\mathbf{x}_k) < \widetilde{V}(\mathbf{x}_k)$. Recall that for the Eikonal equation (4.2), the optimizing direction of (3.6) is in the negative direction of the gradient. To build an approximation $\widetilde{V}$ to (4.2), the gradient ∇V is approximated, $\nabla\widetilde{V}$. The update function $\widetilde{C}$ is now derived.

4.2.1 FMM for Acute Triangles

The following description from [47] is provided in $\mathbb{R}^2$ for a 2-simplicial mesh X. To approximate (4.2), let $\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k$ be a 2-simplex (triangle) and $\angle\mathbf{x}_i\mathbf{x}_k\mathbf{x}_j$ an acute angle. See Figure 4.2a. Suppose $\mathbf{x}_k$ is labelled *Considered*, and the value $\widetilde{C}(\mathbf{x}_k)$ is about to be computed. At least one of $\mathbf{x}_i$ and $\mathbf{x}_j$ must be labelled *Accepted*. If $\mathbf{x}_i$ is labelled *Accepted* and $\mathbf{x}_j$ is not, then (4.2) can be approximated by

$$\|\nabla V(\mathbf{x}_k)\| \approx \frac{\widetilde{C}(\mathbf{x}_k) - \widetilde{V}(\mathbf{x}_i)}{\|\mathbf{x}_k - \mathbf{x}_i\|} = g(\mathbf{x}_k),$$

and hence

$$\widetilde{C}(\mathbf{x}_k) := \|\mathbf{x}_k - \mathbf{x}_i\| \, g(\mathbf{x}_k) + \widetilde{V}(\mathbf{x}_i). \tag{4.3}$$

The gradient ∇V will not in general be aligned with the edges of the mesh. To obtain an accurate update, consider $\mathbf{x}_i$ and $\mathbf{x}_j$ both labelled *Accepted* and assume $\widetilde{V}(\mathbf{x}_j) \geq \widetilde{V}(\mathbf{x}_i)$. Let $\widetilde{\mathbf{x}}_{ik}$ be the point along $\mathbf{x}_i\mathbf{x}_k$ such that $\widetilde{V}(\widetilde{\mathbf{x}}_{ik}) = \widetilde{V}(\mathbf{x}_j)$. The update can be rewritten

$$\frac{\widetilde{C}(\mathbf{x}_k) - \widetilde{V}(\mathbf{x}_j)}{h} = g(\mathbf{x}_k),$$

where h is the perpendicular distance from the line segment $\mathbf{x}_i\widetilde{\mathbf{x}}_{ik}$ to vertex $\mathbf{x}_j$. Both $\widetilde{C}(\mathbf{x}_k)$ and h are unknown. Let $\widetilde{C}(\mathbf{x}_k) = \widetilde{V}(\mathbf{x}_i) + t$, and $r = \widetilde{V}(\mathbf{x}_j) - \widetilde{V}(\mathbf{x}_i)$. The update to find t (and hence $\widetilde{C}(\mathbf{x}_k)$) is defined

$$\frac{t - r}{h} = g(\mathbf{x}_k), \tag{4.4}$$

Let $a = \|\mathbf{x}_j - \mathbf{x}_k\|$ and $b = \|\mathbf{x}_i - \mathbf{x}_k\|$. Consider the tilted plane defined by the three points $(\mathbf{x}_i, \widetilde{V}(\mathbf{x}_i))$, $(\mathbf{x}_j, \widetilde{V}(\mathbf{x}_j))$ and $(\mathbf{x}_k, \widetilde{C}(\mathbf{x}_k))$. See Figure 4.2b. Using the sine and cosine laws,

$$h = a \sin\phi = a \frac{\|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\|}{\|\mathbf{x}_j - \widetilde{\mathbf{x}}_{ik}\|} \sin\theta \tag{4.5}$$

$$= \frac{a\, \|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\| \sin\theta}{\sqrt{a^2 + \|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\|^2 - 2a\, \|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\| \cos\theta}}. \tag{4.6}$$

By similar triangles $t/b = \frac{r}{\|\mathbf{x}_i - \widetilde{\mathbf{x}}_{ik}\|}$ and hence,

$$\|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\| = b - \|\mathbf{x}_i - \widetilde{\mathbf{x}}_{ik}\| = b - br/t = b(t - r)/t.$$

Substituting (4.6) into (4.4) and using $\|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\| = b(t - r)/t$, a quadratic in t is found:

$$(a^2 + b^2 - 2ab\cos\theta)t^2 + 2br(a\cos\theta - b)t + b^2(r^2 - g(\mathbf{x}_k)a^2 \sin^2\theta) = 0. \tag{4.7}$$

Solving the quadratic expression for t in (4.7) yields the update. The correct update direction $\nabla\widetilde{V}$ along $\mathbf{x}_k\widetilde{\mathbf{x}}_{ijk}$ should come from inside the triangle. The value solved in the above quadratic must satisfy the following inequality,

$$a\cos\theta \leq \|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\| = \frac{b(t - r)}{t} \leq \frac{a}{\cos\theta}. \tag{4.8}$$

The inequality can be interpreted geometrically as follows. Let $\widetilde{\mathbf{x}}_{ijk}$ be the point along $\mathbf{x}_j\widetilde{\mathbf{x}}_{ik}$ that forms a perpendicular with $\mathbf{x}_k\widetilde{\mathbf{x}}_{ijk}$. See Figure 4.2a. If $\|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\| = a\cos\theta$, then

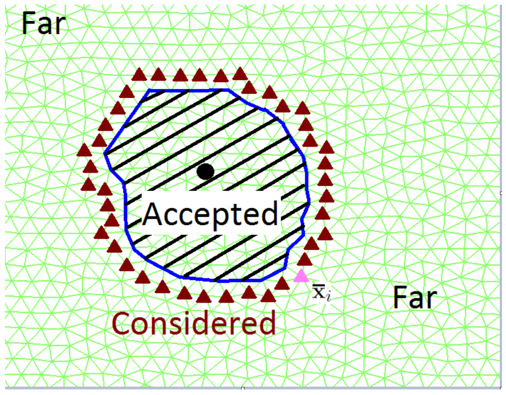


Figure 4.3: Fast Marching Method - Labels of vertices - $\mathbf{x}_f$: Black Dot, *Accepted*: Shaded area, *Considered*: Triangles, $\bar{\mathbf{x}}_i$: Next *Accepted* vertex, *Far*: Unmarked

$\mathbf{x}_k\widetilde{\mathbf{x}}_{ijk}$ is normal to $\mathbf{x}_j\widetilde{\mathbf{x}}_{ik}$ and collinear to $\mathbf{x}_k\widetilde{\mathbf{x}}_{ik}$. On the other hand, if $\|\mathbf{x}_k - \widetilde{\mathbf{x}}_{ik}\| = \frac{a}{\cos\theta}$, then $\mathbf{x}_k\widetilde{\mathbf{x}}_{ijk}$ is collinear to $\mathbf{x}_j\mathbf{x}_k$. If inequality (4.8) is not satisfied, then $\nabla\widetilde{V}$ does not lie inside the triangle. Finally, $\widetilde{C}(\mathbf{x}_k)$ must be larger than $\widetilde{V}(\mathbf{x}_j)$ (since $r < t$).

If $r < t$ and inequality (4.8) holds, then the update is

$$\widetilde{C}(\mathbf{x}_k) := t + \widetilde{V}(\mathbf{x}_i). \tag{4.9}$$

Otherwise the update must come from an edge of the triangle,

$$\widetilde{C}(\mathbf{x}_k) := \min\{bg(\mathbf{x}_k) + \widetilde{V}(\mathbf{x}_i), ag(\mathbf{x}_k) + \widetilde{V}(\mathbf{x}_j)\}. \tag{4.10}$$

The FMM algorithm for the source-point problem (4.2) with $\partial\Omega = \{\mathbf{x}_f\}$, $q(\mathbf{x}_f) = 0$ on a simplicial mesh of acute triangles is as follows. Relabelling a vertex removes the previous label.

1. Label all vertices $\{\mathbf{x}_i\}$ of mesh X *Far* with tentative value $\widetilde{V}(\mathbf{x}_i) = K$ where K a is large value.

2. Relabel $\mathbf{x}_f$ *Accepted* and set $\widetilde{V}(\mathbf{x}_f) = 0$.

3. Relabel the neighbours of $\mathbf{x}_f$ *Considered.* Compute $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$ from $\mathbf{x}_f$ using equation (4.3).

4. Relabel $\bar{\mathbf{x}}_i$ *Accepted.*

5. If all vertices are labelled *Accepted*, terminate. Otherwise, find vertex $\overline{\mathbf{x}}_i = \underset{\mathbf{x}_i \in \textit{Considered}}{\arg\min} \widetilde{V}(\mathbf{x}_i)$. If such a $\mathbf{x}_i$ is not unique, choose one.

6. Relabel all neighbours of $\overline{\mathbf{x}}_i$ with *Far* label to *Considered* label.

7. Compute $\widetilde{C}(\mathbf{x}_i)$ using (4.9) & (4.10) for all vertices neighbouring $\overline{\mathbf{x}}_i$. If $\widetilde{V}(\mathbf{x}_i) < \widetilde{C}(\mathbf{x}_i)$, then set $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$. Otherwise, do nothing.

8. Go to Step 4.

Rather than finding the solution iteratively, the solution $\widetilde{V}(\mathbf{x}_i)$ at a vertex $\mathbf{x}_i$ only depends on its neighbouring vertices that have a lower value of $\widetilde{V}$. The vertices are labelled *Accepted* in a nondecreasing manner.

Theorem 4.2.1 ([47])**.** *(Monotone Acceptance) If $\boldsymbol{x}_j$ is labelled Accepted after $\boldsymbol{x}_k$, then $\widetilde{V}(\boldsymbol{x}_j) \geq \widetilde{V}(\boldsymbol{x}_k)$.*

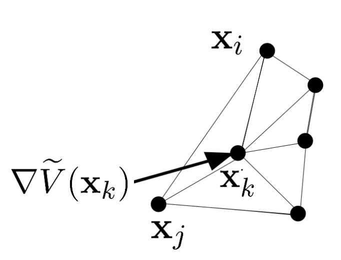
The Fast Marching Method obeys the following property.

Theorem 4.2.2 (Property 3.6 of [72])**.** *(Causality) Assume that $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_k$ are vertices on X that form an acute triangle. If $\nabla\widetilde{V}(\boldsymbol{x}_i)$ is defined, and the minimizing $\boldsymbol{u}^* = -\frac{\nabla\widetilde{V}(\boldsymbol{x}_i)}{\|\nabla\widetilde{V}(\boldsymbol{x}_i)\|}$ points into triangle $\boldsymbol{x}_i\boldsymbol{x}_j\boldsymbol{x}_k$, then $\widetilde{V}(\boldsymbol{x}_i) \geq \max\{\widetilde{V}(\boldsymbol{x}_j), \widetilde{V}(\boldsymbol{x}_k)\}$.*

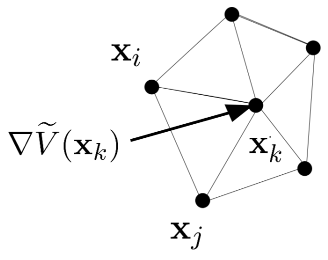
The FMM algorithm has complexity $\mathcal{O}(N \log N)$, where N is the number of vertices in the mesh [73]. The set of vertices labelled *Considered* is maintained in a minimum-heap binary array. Updates (4.9),(4.10) are repeated $\mathcal{O}(N)$ times and the sorting and maintaining the minimum-heap array requires $\mathcal{O}(\log N)$ operations. The FMM algorithm can be extended to meshes with obtuse triangles, but an additional step is required.

4.2.2 Obtuse Triangulations

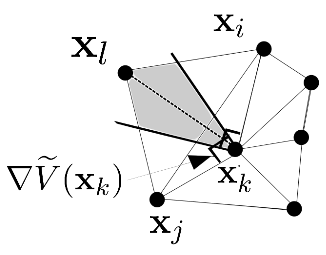
The update procedure described for acute triangles may be incorrect when used for obtuse triangles. In Figure 4.4a, $\nabla\widetilde{V}(\mathbf{x}_k)$ points into the obtuse triangle $\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k$. For small enough triangles where $\nabla V(\mathbf{x}_i) \approx \nabla V(\mathbf{x}_j) \approx \nabla V(\mathbf{x}_k)$, $\mathbf{x}_i$ is labelled *Accepted* after $\mathbf{x}_k$, $\widetilde{V}(\mathbf{x}_k)$ is updated with incorrect direction originating from edge $\mathbf{x}_i\mathbf{x}_j$. This cannot occur in an acute triangulation, see Figure 4.4b.



(a) Obtuse Angle Triangulation (Triangle $\mathbf{x}_i\mathbf{x}_k\mathbf{x}_j$ is obtuse)



(b) Acute Angle Triangulation



(c) Virtual Angle Projection, $\mathbf{x}_j\mathbf{x}_k\mathbf{x}_l$ acute

Figure 4.4: Obtuse and Acute Mesh Triangulations. The correct update is in the direction of $\nabla\widetilde{V}(\mathbf{x}_k)$.

To obtain a correct update direction, the obtuse angle $\angle\mathbf{x}_i\mathbf{x}_k\mathbf{x}_j$ is split using right angles with its sides. See Figure 4.4c. The region bounded by the two splittings is searched outwards from $\mathbf{x}_k$ until a vertex labelled *Accepted* is found ($\mathbf{x}_l$). The update for $\mathbf{x}_k$ is then computed using (4.9) through the virtual triangle $\mathbf{x}_j\mathbf{x}_k\mathbf{x}_l$. The angle $\angle\mathbf{x}_j\mathbf{x}_k\mathbf{x}_l$ is acute and will yield a correct update.

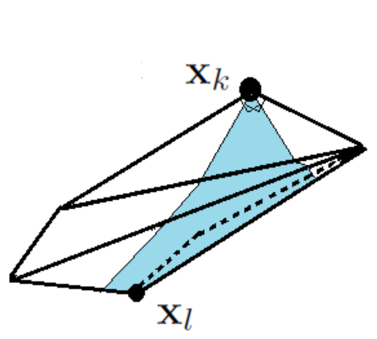
For a mesh approximating a surface in $\mathbb{R}^3$, the three-dimensional distances when splitting must be considered. See Figure 4.5. The middle portion will fold along the triangles in the mesh to obtain the true distance. When unfolded (see Figure 4.5b), the projected angle is identical to that on a 2-simplicial mesh.

Aside from finding a virtual acute triangle to calculate the update (4.9), (4.10), the algorithm is the same as before in the presence of obtuse triangles.

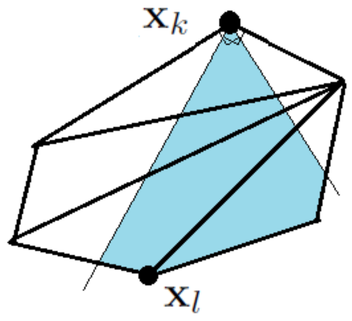
4.2.3 Finding the Path

The function $\widetilde{V}$ is extended from the vertices of X to $\overline{\Omega} \subset \mathbb{R}^n$ by linear interpolation. That is, if $\mathbf{x} \in \mathbf{s} = \mathbf{x}_0^{\mathbf{s}}\mathbf{x}_1^{\mathbf{s}} \cdots \mathbf{x}_n^{\mathbf{s}}$, and has barycentric coordinates such that

$$\mathbf{x} = \sum_{j=0}^{n} \zeta_j \mathbf{x}_j^{\mathbf{s}},$$



(a) The mesh in 3D containing an obtuse angle. The search for an update point requires compensating for folds in the mesh.



(b) Unfolding the mesh to account for true distances will produce a point to form an acute triangle with the correct update direction.

Figure 4.5: The unfolding process for obtuse triangles.

and

$$\widetilde{V}(\mathbf{x}) = \sum_{j=0}^{n} \zeta_j \widetilde{V}(\mathbf{x}_j^{\mathbf{s}}).$$

In the context of path planning, an approximated optimal control $\widetilde{\mathbf{u}}^*(\cdot)$ is found by solving (3.8), with V replaced with $\widetilde{V}$.

$$\dot{\mathbf{y}}(t) = \widetilde{\mathbf{u}}^*(t) = -\frac{\nabla\widetilde{V}(\mathbf{y}(t))}{\left\|\nabla\widetilde{V}(\mathbf{y}(t))\right\|},$$

$$\mathbf{y}(0) = \mathbf{x}_0.$$

It was shown in Theorem 3.3.5 that the true value solution V does not have any local minima. From Theorem 4.2.1, and the Fast Marching Method algorithm, it can easily be shown that $\widetilde{V}$ does not have any strict local minima aside from $\mathbf{x}_f$.

4.3 Ordered Upwind Method (OUM)

The Ordered Upwind Method (OUM) was introduced by Sethian and Vladimirsky in [72] to obtain an approximation $\widetilde{V}$ of V (3.5) on the vertices of a mesh X that discretizes

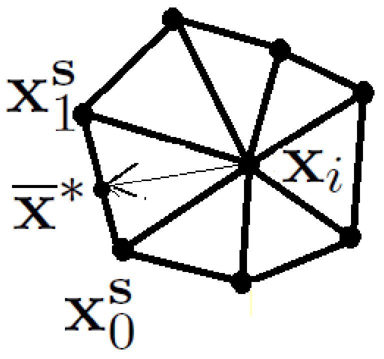

$\mathbf{x}_1^{\mathbf{s}}$
$\overline{\mathbf{x}}^*$
$\mathbf{x}_i$
$\mathbf{x}_0^{\mathbf{s}}$

Figure 4.6: OUM Update in $\Omega \subset \mathbb{R}^2$- The optimal trajectory at vertex $\mathbf{x}_i$ exits the edge $\mathbf{s}^* = \mathbf{x}_0^{\mathbf{s}}\mathbf{x}_1^{\mathbf{s}}$ at $\overline{\mathbf{x}}^*$. The point $\overline{\mathbf{x}}^*$ is found in terms of the barycentric coordinates ζ^* of $\mathbf{s}^*$ where $\overline{\mathbf{x}}^* = \sum_{j=0}^{1} \zeta_j^* \mathbf{x}_j^{\mathbf{s}}$.

$\overline{\Omega}$. The advantage of OUM over FMM [47] is that OUM can solve (3.5) for weights that depend on direction. The presentation of the algorithm will be given in $\mathbb{R}^n$.

As in FMM, the solution in OUM is again built outwards from $\mathbf{x}_f$ with $\widetilde{V}(\mathbf{x}_f) = 0$. The vertices of the mesh X are assigned the same labels *Accepted*, *Considered*, and *Far* as in the FMM algorithm with the same description. Vertices and $(n-1)$-simplices made of vertices with *Accepted* label are further classified.

Accepted Front - The vertices labelled *Accepted* that have a neighbour labelled *Considered.*

AF - The set of $(n-1)$-simplices made of *Accepted Front* vertices that have a neighbouring vertex (Definition 4.1.10) labelled *Considered.*

***Near Front** of* $\boldsymbol{x}_i$ ($\mathbf{NF}(\mathbf{x}_i)$) - Let $\mathbf{x}_i$ be labelled *Considered.* Then,

$$\mathbf{NF}(\mathbf{x}_i) = \left\{ \mathbf{s} \in \mathbf{AF} \middle| \ \exists\ \widetilde{\mathbf{x}} \in \mathbf{s} \middle| \ \|\widetilde{\mathbf{x}} - \mathbf{x}_i\| \leq \Gamma h_{max} \right\}, \tag{4.11}$$

where $\Gamma = \frac{G_{max}}{G_{min}}$ (Definition 3.1.8). See Figure 4.7. The set of $(n-1)$-simplices $\mathbf{AF}$ and $\mathbf{NF}(\mathbf{x}_i)$ change due to vertices being relabelled from *Far* to *Considered* to *Accepted.* It was proven in [72] that the minimizing direction for $\mathbf{x}_i$ must come from its *Near Front.*

The OUM uses a semi-Lagrangian scheme where the control is assumed to be held constant within each simplex, see Figure 4.6. Let $\mathbf{s}_{n-1} = \mathbf{x}_0^{\mathbf{s}} \cdots \mathbf{x}_{n-1}^{\mathbf{s}}$ be the $(n-1)$-simplex for which the optimal trajectory at $\mathbf{x}_i$ exits the n-simplex $\mathbf{s}_n = \mathbf{s}_{n-1}\mathbf{x}_i$ at $\overline{\mathbf{x}}^* \in \mathbf{s}_n$. See Figure 4.6. Let $\zeta^* = (\zeta_0^*, \zeta_1^*, ..., \zeta_{n-1}^*) \in \Xi_{n-1}$ be barycentric coordinates of $\overline{\mathbf{x}}^* \in \mathbf{s}^*$ such that

$$\overline{\mathbf{x}}^* = \sum_{j=0}^{n-1} \zeta_j^* \mathbf{x}_j^{\mathbf{s}}.$$

The distance $\tau_{\mathbf{s}}$ between $\mathbf{x}_i$ and $\overline{\mathbf{x}}^* \in \mathbf{s}^*$ and minimizing direction $\widetilde{\mathbf{u}}_{\mathbf{s}}^*(\mathbf{x}_i, \zeta^*)$ are

$$\tau_{\mathbf{s}}(\mathbf{x}_i, \zeta^*) := \left\| \sum_{j=0}^{n-1} \zeta_j^* \mathbf{x}_j - \mathbf{x}_i \right\| = \|\overline{\mathbf{x}}^* - \mathbf{x}_i\| \text{ and } \widetilde{\mathbf{u}}_{\mathbf{s}}^*(\mathbf{x}_i, \zeta^*) := \frac{\overline{\mathbf{x}}^* - \mathbf{x}_i}{\tau(\mathbf{x}_i, \zeta^*)} \tag{4.12}$$

respectively.

The update for $\mathbf{x}_i$ provided by the $(n-1)$-simplex $\mathbf{s}_{n-1} = \mathbf{x}_0^{\mathbf{s}}\mathbf{x}_1^{\mathbf{s}} \cdots \mathbf{x}_{n-1}^{\mathbf{s}}$ is a first-order approximation of (3.5),

$$\widetilde{C}_{\mathbf{s}_{n-1}}(\mathbf{x}_i) := \min_{\zeta \in \Xi_{n-1}} \left\{ \tau(\mathbf{x}_i, \zeta) g(\mathbf{x}_i, \mathbf{u}_\zeta) + \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\mathbf{x}_j^{\mathbf{s}}) \right\}, \tag{4.13}$$

where $\zeta = (\zeta_0, \zeta_1, ..., \zeta_{n-1}) \in \Xi_{n-1}$ (4.1). The optimizing direction is captured by updating $\mathbf{x}_i$ from its *Near Front* [72]. The update formula over all of the *Near Front of $\boldsymbol{x}_i$*, $\mathbf{NF}(\mathbf{x}_i)$ is

$$\widetilde{C}(\mathbf{x}_i) := \min_{\mathbf{s}_{n-1} \in \mathbf{NF}(\mathbf{x}_i)} \widetilde{C}_{\mathbf{s}_{n-1}}(\mathbf{x}_i). \tag{4.14}$$

The OUM algorithm will now be described. Recall that any vertex $\mathbf{x}_i \in X$ is labelled only one of *Accepted*, *Considered* or *Far* at any instant of the algorithm. Relabelling a vertex removes the previous label.

1. Label all vertices of X *Far*, with $\widetilde{V}(\mathbf{x}_i) = K$ (where K is large).

2. Relabel $\mathbf{x}_f$ *Accepted*, and let $\widetilde{V}(\mathbf{x}_f) = 0$.

3. Relabel all neighbours $\mathbf{x}_i$ of $\mathbf{x}_f$ to *Considered*. Compute the values $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$ according to (4.14).

4. Relabel $\overline{\mathbf{x}}_i = \arg\min_{\mathbf{x}_i \in \textit{Considered}} \widetilde{V}(\mathbf{x}_i)$ *Accepted*. If all vertices in X are labelled *Accepted*, terminate the algorithm.

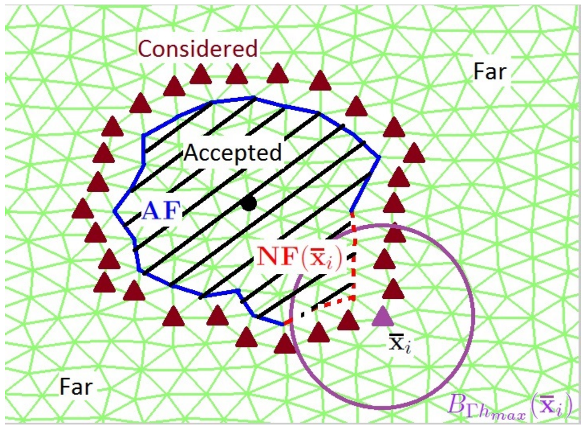

Considered
Far
Accepted
AF
$\mathbf{NF}(\overline{\mathbf{x}}_i)$
$\overline{\mathbf{x}}_i$
Far
$B_{\Gamma h_{max}}(\overline{\mathbf{x}}_i)$

Figure 4.7: Vertex labels in Ordered Upwind Method ($X \subset \mathbb{R}^2$) - Unmarked vertices have *Far* label. Vertices marked with a triangle have *Considered* label. Shaded vertices (including those on the border) have *Accepted* label. Vertex $\overline{\mathbf{x}}_i$ with *Considered* label is being updated. The ball $B_{\Gamma h_{max}}(\overline{\mathbf{x}}_i)$ has radius $\Gamma(\overline{\mathbf{x}}_i)h_{max}$ and centre $\overline{\mathbf{x}}_i$. The *Near Front* $\mathbf{NF}(\overline{\mathbf{x}}_i)$, shown dotted is the subset of edges in $\mathbf{AF}$ that contain a point within Γh_{max} of $\overline{\mathbf{x}}_i$ used in the computation (4.14).

5. Relabel all neighbouring vertices $\mathbf{x}_i$ of $\overline{\mathbf{x}}_i$ with *Far* label to *Considered*. For such $\mathbf{x}_i$, compute $\widetilde{C}(\mathbf{x}_i)$ using (4.14) and set $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$.

6. Recompute $\widetilde{C}(\mathbf{x}_i)$ for all other $\mathbf{x}_i$ with *Considered* label using (4.14) such that $\overline{\mathbf{x}}_i \in \mathbf{NF}(\mathbf{x}_i)$, using only the subset of $(n-1)$-simplices in $\mathbf{NF}(\mathbf{x}_i)$ that contain $\overline{\mathbf{x}}_i$. If $\widetilde{V}(\mathbf{x}_i) > \widetilde{C}(\mathbf{x}_i)$, then update $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$. Go to Step 4.

The different labels of Ordered Upwind Method are shown in Figure 4.7. The algorithm is terminated once all the vertices of X have been labelled *Accepted*. The solution produced by the OUM, $\widetilde{V}$ converges to the viscosity solution V of (3.6) as the mesh is refined ($h_{max} \to 0$) provided the quantity $\frac{h_{max}}{h_{min}}$ is bounded [72, Theorem 7.7].

To obtain an approximate optimal path, a minimization problem (3.8) is solved for $\mathbf{u}^*$ (replacing V with $\widetilde{V}$ computed by OUM) each time the path exits its current triangle and enters a new adjacent triangle. The path is not restricted to the edges of the mesh and is found from $\mathbf{y}(0) = \mathbf{x}_0$ and complete when $\mathbf{y}(T) = \mathbf{x}_f$ is reached. The OUM is not a strictly monotone acceptance algorithm in the sense of Theorem 4.2.1. Vertices are not necessarily labelled *Accepted* in nondecreasing value. A weaker property is observed. These properties will be stated and used in Chapter 5. Strict local minima away from $\mathbf{x}_f$ may exist in the approximation $\widetilde{V}$ computed by OUM. Situations in which this can arise are described in Section 6.1.3. In practice, such local minima disappear as the mesh is refined as $h_{max} \to 0$.

The computational complexity of OUM is $\mathcal{O}(\Gamma^{n-1} N \log N)$ [72]. Recall $\Gamma = \frac{G_{max}}{G_{min}}$ (3.3) and N is the number of vertices in the mesh X. As in FMM, the vertices labelled *Considered* are maintained in a minimum binary heap array [73]. The first entry of the array always contains the smallest value of the vertices labelled *Considered*. Only $\mathcal{O}(\log N)$ operations are required to maintain the heap ordering when a vertex is added, updated or removed. Additional implementation heuristics for OUM [72] will be discussed in Section 6.1, including the use of a local anisotropy coefficient for the *Near Front*.

4.3.1 Bi-Directional OUM

While the idea of a bi-directional search on discrete graphs is not new [66], bi-directional search has not been used previously with OUM. Instead of finding $\widetilde{V}$ on all of $\overline{\Omega}$, any region inside $\overline{\Omega}$ that contains the optimal path is sufficient to solve (3.8) to recover the optimal path. An obvious improvement to the OUM is to terminate the search when the initial point $\mathbf{x}_0$ is labelled *Accepted* (that is, a unidirectional search from $\mathbf{x}_f$). A likely smaller limiting region will be found by also considering the path planning problem where the roles of $\mathbf{x}_0$ and $\mathbf{x}_f$ in (3.1), (3.2) in Section 3.1 are reversed. Let $\mathbf{y}_{\mathbf{x}_0}$, $\mathbf{u}_{\mathbf{x}_0}$, $T_{\mathbf{x}_0}$, $Cost_{\mathbf{x}_0}$, $V_{\mathbf{x}_0}$ and $g_{\mathbf{x}_0}$ denote the trajectory, control, exit-time, cost, value function and weight respectively in the optimal control problem described in (3.1), (3.2) and (3.4).

Let

$$\begin{aligned} \dot{\mathbf{y}}_{\mathbf{x}_f}(t) &= \mathbf{u}_{\mathbf{x}_f}(t) \\ \mathbf{y}(0) &= \mathbf{x}_f, \quad \mathbf{x}_f \in \Omega. \end{aligned} \tag{4.15}$$

where $\mathbf{u}(\cdot) \in \mathcal{U}$. The objective is to reach $\mathbf{y}_{\mathbf{x}_f}(T_{\mathbf{x}_f}) = \mathbf{x}_0$, where $T_{\mathbf{x}_f}$ is the exit-time (Definition 3.1.1) for (4.15), while minimizing the cost function,

$$Cost_{\mathbf{x}_f}(\mathbf{x}_f, \mathbf{u}_{\mathbf{x}_f}(\cdot)) = \int_0^{T_{\mathbf{x}_f}(\mathbf{x}_f, \mathbf{u}_{\mathbf{x}_f}(\cdot))} g_{\mathbf{x}_f}(\mathbf{y}_{\mathbf{x}_f}(s), \mathbf{u}_{\mathbf{x}_f}(s)) ds \tag{4.16}$$

where $g_{\mathbf{x}_f} : \overline{\Omega} \times \mathbb{S}^{n-1}$ is the weight of (4.15). The value function $V_{\mathbf{x}_f} : \overline{\Omega} \to \mathbb{R}_+$ is

$$V_{\mathbf{x}_f}(\mathbf{x}) = \inf_{\mathbf{u}_{\mathbf{x}_f}(\cdot) \in \mathcal{U}} Cost_{\mathbf{x}_f}(\mathbf{x}, \mathbf{u}_{\mathbf{x}_f}(\cdot)).$$

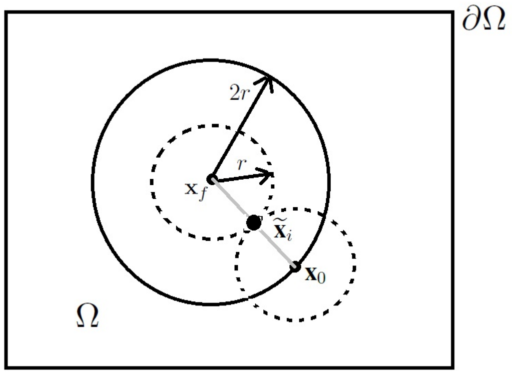


Figure 4.8: Bi-directional Search - Consider the shortest path problem. The solid circle is the level set of the single problem when $\mathbf{x}_0$ has been reached. The dotted circles are the respective level sets in each of the problems in the bi-directional problem. The value of the two level sets are the same. If the initial point $\mathbf{x}_0$ and final point $\mathbf{x}_f$ are far enough from $\partial\Omega$, the areas between the bi-directional search and original problem are reduced by a factor of 2. The optimal path (straight line) is shown in grey.

Suppose both problems (3.1) and (4.15) provide the same optimal path. See Figure 4.8. By the optimality principle, the point $\widetilde{\mathbf{x}}_i$ representing the lowest sum between the two problems must be on the optimal path. The points in OUM are finalized in the manner of an advancing front. The point $\widetilde{\mathbf{x}}_i$ is the instance for which the two fronts meet for the first time. The optimal path is recovered by solving (3.8), using their respective value functions and the initial condition $\widetilde{\mathbf{x}}_i$ for both problems. The following theorem is required to guarantee that an optimal path for one problem (4.15) is optimal for the other (3.1).

Theorem 4.3.1. *Assume $g_{\boldsymbol{x}_0}(\boldsymbol{x}, \boldsymbol{u}) = g_{\boldsymbol{x}_f}(\boldsymbol{x}, -\boldsymbol{u})$. Let $\boldsymbol{u}^*_{\boldsymbol{x}_0}(\cdot), \boldsymbol{u}^*_{\boldsymbol{x}_f}(\cdot) \in \mathcal{U}$ be optimal controls for the problems (3.1) and (4.15) respectively. Then the reversed controls $-\boldsymbol{u}^*_{\boldsymbol{x}_f}(T_{\boldsymbol{x}_f} - \cdot)$ and $-\boldsymbol{u}^*_{\boldsymbol{x}_0}(T_{\boldsymbol{x}_0} - \cdot)$ are also optimal controls in (3.1) and (4.15) respectively.*

Proof.

$$\begin{aligned}
V_{\mathbf{x}_f}(\mathbf{x}_f) &= Cost_{\mathbf{x}_f}(\mathbf{x}_f, \mathbf{u}^*_{\mathbf{x}_f}(\cdot)) \le Cost_{\mathbf{x}_f}(\mathbf{x}_f, -\mathbf{u}^*_{\mathbf{x}_0}(T_{\mathbf{x}_0} - \cdot)),\\
&= \int_0^{T_{\mathbf{x}_0}} g_{\mathbf{x}_f}(\mathbf{y}_{\mathbf{x}_0}(T_{\mathbf{x}_0} - s), -\mathbf{u}^*_{\mathbf{x}_0}(T_{\mathbf{x}_0} - s))ds\\
&= \int_{T_{\mathbf{x}_0}}^0 -g_{\mathbf{x}_f}(\mathbf{y}_{\mathbf{x}_0}(v), -\mathbf{u}^*_{\mathbf{x}_0}(v))dv,\\
&= \int_0^{T_{\mathbf{x}_0}} g_{\mathbf{x}_0}(\mathbf{y}_{\mathbf{x}_0}(s), \mathbf{u}^*_{\mathbf{x}_0}(s))ds = Cost_{\mathbf{x}_0}(\mathbf{x}_0, \mathbf{u}^*_{\mathbf{x}_0}(\cdot)) = V_{\mathbf{x}_0}(\mathbf{x}_0).
\end{aligned}$$

Similarly, $V_{\mathbf{x}_0}(\mathbf{x}_0) \le V_{\mathbf{x}_f}(\mathbf{x}_f)$. Therefore $V_{\mathbf{x}_0}(\mathbf{x}_0) = V_{\mathbf{x}_f}(\mathbf{x}_f)$. Hence $-\mathbf{u}^*_{\mathbf{x}_f}(T_{\mathbf{x}_f} - \cdot)$ and $-\mathbf{u}^*_{\mathbf{x}_0}(T_{\mathbf{x}_0} - \cdot)$ are optimal controls for (3.1) and (4.15) respectively. □

Furthermore, if $\mathbf{u}^*_{\mathbf{x}_0}(\cdot) = -\mathbf{u}^*_{\mathbf{x}_f}(T_{\mathbf{x}_f} - \cdot)$, then $T = T_{\mathbf{x}_0} = T_{\mathbf{x}_f}$ and the optimal path for both problems are the same. This may not be the case. If $\nabla V_{\mathbf{x}_0}(\mathbf{x}_0)$ or $\nabla V_{\mathbf{x}_f}(\mathbf{x}_f)$ are not defined, then multiple optimal paths may exist. See Figure 3.7.

The assumption $g_{\mathbf{x}_0}(\mathbf{x}, \mathbf{u}) = g_{\mathbf{x}_f}(\mathbf{x}, -\mathbf{u})$ in Theorem 4.3.1 is automatically satisfied for weights that are symmetric about its origin. The two problems (4.15) are solved concurrently with $g_{\mathbf{x}_0}(\mathbf{x}, \mathbf{u}) = g_{\mathbf{x}_f}(\mathbf{x}, -\mathbf{u})$. Let $\widetilde{T}$ be the time at which the fronts in the problems (4.15) meet. Then the control

$$\mathbf{u}^*(t) = \begin{cases} \mathbf{u}^*_{\mathbf{x}_0}(t), & t \in [0, \widetilde{T}] \\ -\mathbf{u}^*_{\mathbf{x}_f}(T - t), & t \in (\widetilde{T}, T] \end{cases}$$

is a solution to the optimal path planning problem for both problems in (4.15) as long as $y^*_{\mathbf{x}_0}(\widetilde{T}) = y^*_{\mathbf{x}_f}(T - \widetilde{T})$. There may be multiple points $\mathbf{x}^*$ where the fronts meet for the first time, for example in Figure 3.7. In such a case, $\mathbf{u}^*_{\mathbf{x}_0}$ and $-\mathbf{u}^*_{\mathbf{x}_f}$ are chosen from the same point $\mathbf{x}^*$.

The OUM-BD labels and algorithm are as follows. The labels $Far_{\mathbf{x}_0}$, $Far_{\mathbf{x}_f}$, $Considered_{\mathbf{x}_0}$, $Considered_{\mathbf{x}_f}$, $Accepted_{\mathbf{x}_0}$ and $Accepted_{\mathbf{x}_f}$ are the same labels as described in Section 4.3 for the problems (3.1) and (4.15) respectively.

1. Label all vertices in $\mathbf{x}_i \in X$ in both $Far_{\mathbf{x}_0}$ with $\widetilde{V}_{\mathbf{x}_0}(\mathbf{x}_i) = K$ and $Far_{\mathbf{x}_f}$ with $\widetilde{V}_{\mathbf{x}_f}(\mathbf{x}_i) = K$, where K is a large value.

2. Label $\mathbf{x}_f$ as $Accepted_{\mathbf{x}_0}$, setting $\widetilde{V}_{\mathbf{x}_0}(\mathbf{x}_f) = 0$ and label $\mathbf{x}_{\mathbf{x}_0}$ as $Accepted_{\mathbf{x}_f}$, setting $\widetilde{V}_{\mathbf{x}_f}(\mathbf{x}_0) = 0$.

3. Relabel the neighbours of $\mathbf{x}_f$ in *Considered*$_{\mathbf{x}_0}$ and the neighbours of $\mathbf{x}_0$ in *Considered*$_{\mathbf{x}_f}$ and update each from their respective *Accepted* vertex using (4.14).

4. Let $\overline{\mathbf{x}}_i$ be the vertex in $\{$*Considered*$_{\mathbf{x}_0} \cup$ *Considered*$_{\mathbf{x}_f}\}$ with minimum tentative value ($\widetilde{V}_{\mathbf{x}_0}$ or $\widetilde{V}_{\mathbf{x}_f}$). Let $k \in \{\mathbf{x}_0, \mathbf{x}_f\}$ be the *Considered*$_k$ label with the minimizing $\overline{\mathbf{x}}_i$.

5. Relabel $\overline{\mathbf{x}}_i$ from its respective *Considered*$_k$ label to its respective *Accepted*$_k$ label. Relabel the neighbours of $\overline{\mathbf{x}}_i$ with *Far*$_k$ label to *Considered*$_k$ label.

6. Update the vertices in *Considered*$_k$ affected by relabelling $\overline{\mathbf{x}}_i$ *Accepted*$_k$ using (4.14).

7. If no vertices have both *Accepted*$_{\mathbf{x}_0}$ and *Accepted*$_{\mathbf{x}_f}$ labels, then go to Step 4. Otherwise, let $\widetilde{\mathbf{x}}_i$ be the vertex labelled both *Accepted*$_{\mathbf{x}_0}$ and *Accepted*$_{\mathbf{x}_f}$. Repeat steps 4-6 until all the vertices of the triangles in which the update for $\mathbf{x}_i$ came from traverses in both problems become labelled *Accepted*. Then terminate.

To find the optimal path, the triangles of X which the optimal path traverses must be made entirely of vertices labelled *Accepted*. This is guaranteed in the last step. The optimal path is found by solving (3.8) using $\widetilde{V}_1$ and $\widetilde{V}_2$ with their respective problems and then connecting the two paths. The vertex $\widetilde{\mathbf{x}}_i$ may not lie on the actual optimal path of either problem in (4.15). However as $h_{max} \to 0$, $\widetilde{V}_{\mathbf{x}_0}$ and $\widetilde{V}_{\mathbf{x}_f}$ will converge to $V_{\mathbf{x}_0}$ and $V_{\mathbf{x}_f}$ [72, Theorem 7.7]. The two value functions can be solved independently using parallel processors with the termination condition in the last step of OUM-BD checked periodically.

For the Fast Marching Method, a brief description of a bi-directional search is provided in [15]. The OUM-BD can easily be modified to a bi-directional FMM algorithm, FMM-BD. The termination condition in Step 4 is replaced with "once the neighbours of $\widetilde{\mathbf{x}}_i$ have been labelled *Accepted*", and the update formula in Step 3 with the FMM update in [47].

In terms of computation saved, the number of vertices labelled *Accepted*$_k$ is less than the full problem. As well, the set of vertices labelled *Considered*$_k$ are smaller in each problem and have fewer elements for OUM-BD. Fewer operations are required to both maintain the minimum heap array as well as to solve the update formula (4.13).

The amount of computation saved is dependent on the problem, and location of $\mathbf{x}_0$ and $\mathbf{x}_f$ relative to the boundary of the workspace $\partial\Omega$. Examples are shown in Section 6.2 that demonstrate the effectiveness of the algorithm in terms of computation and time.

4.4 Monotone Acceptance Ordered Upwind Method (MAOUM)

The Monotone Acceptance Ordered Upwind Method (MAOUM) was developed by Alton and Mitchell [3], [5] and computes approximate solutions to the same class of Hamilton-Jacobi-Bellman equations as OUM on a simplicial mesh X.

The MAOUM algorithm is similar to the OUM algorithm. The difference is that the update is computed from a different set of $(n-1)$-simplices. Updates from the OUM are computed from a dynamically evolving *Near Front* of $\mathbf{x}_i$, $\mathbf{NF}(\mathbf{x}_i)$. On the other hand, the updates for MAOUM are from a predetermined set of $(n-1)$-simplices, $\mathcal{M}(\mathbf{x}_i)$ used in the update is found for each vertex $\mathbf{x}_i \in X$. The size of the set $\mathcal{M}(\mathbf{x}_i)$ depends on local mesh sizing, and a measure of local directional dependence

$$\Gamma(\mathbf{x}_i) = \frac{g_{max}(\mathbf{x}_i)}{g_{min}(\mathbf{x}_i)}.$$

The *Near Front* in the OUM algorithm depends on $\Gamma(\mathbf{x}_i)$, but also on the maximum edge length h_{max} over the entire mesh X.

In Section 4.2, the approximate characteristic direction for FMM on simplicial meshes at $\mathbf{x}_i$ came from an acute (but possibly virtual) angle. The acuteness of the angles resulted in monotone acceptance (Theorem 4.2.1) to be maintained [5, 72]. That is, vertices are labelled *Accepted* based on nondecreasing values of $\widetilde{V}$. In MAOUM, the maximum angle formed with the vertex to be updated $\mathbf{x}_i$ and any $(n-1)$-simplex $\mathbf{s} \in \mathcal{M}(\mathbf{x}_i)$ must be small enough to satisfy a similar bound defined by $\Gamma(\mathbf{x}_i)$. Let

$$\hat{\mathcal{U}}_{\mathbf{s}}(\mathbf{x}_i) = \left\{ \mathbf{u} \in \mathbb{S}^{n-1} \middle| \mathbf{x} \in \mathbf{s}, \mathbf{u} = \frac{\mathbf{x} - \mathbf{x}_i}{\|\mathbf{x} - \mathbf{x}_i\|} \right\}$$

where $\mathbb{S}^{n-1}$ denotes the set of unit directions in $\mathbb{R}^n$. Let $\upsilon^{\mathbf{s}}_{i,j}$ denote the angle between $\mathbf{u}_i, \mathbf{u}_j \in \hat{\mathcal{U}}_{\mathbf{s}}(\mathbf{x}_i)$. Define

$$\hat{\upsilon}_{\mathbf{s}} = \max_{i,j} \upsilon^{\mathbf{s}}_{i,j}.$$

Definition 4.4.1. *[5] A $(n-1)$-simplex $\boldsymbol{s}$ is* ***anisotropy angle bounded (AAB)*** *for $\boldsymbol{x}_i$ if*

$$\hat{\upsilon}_{\boldsymbol{s}} < \arcsin\left(\frac{1}{\Gamma(\boldsymbol{x}_i)}\right). \tag{4.17}$$

The presentation of Definition 4.4.1 has been simplified from its original presentation [5].

Let $\mathcal{N}(\mathbf{x}_i)$ denote the set of neighbouring vertices of $\mathbf{x}_i$. The sets Q and $\mathcal{Y}$ are subsets of vertices in X used to determine $\mathcal{M}(\mathbf{x}_i)$. The algorithm to compute the set $\mathcal{M}(\mathbf{x}_i)$ for each vertex is:

1. Let $Q = \mathcal{N}(\mathbf{x}_i)$ and $\mathcal{Y} = \{\mathbf{x}_i\}$.
2. If Q is empty, let $\mathcal{M}(\mathbf{x}_i)$ be the set of $(n-1)$-simplices on X made only of vertices from $\mathcal{Y}$ that are AAB with $\mathbf{x}_i$ and terminate. Otherwise, continue.
3. Let $\mathbf{x}_j$ be any element of Q. Remove $\mathbf{x}_j$ from Q and add it to $\mathcal{Y}$.
4. For each $(n-1)$-simplex $\mathbf{s} \in X$ made up of only vertices in $\mathcal{Y}$ including $\mathbf{x}_j$ but not $\mathbf{x}_i$, determine if $\mathbf{s}$ satisfies AAB for $\mathbf{x}_i$ (4.17).
    (a) If not, add the neighbour (Definition 4.1.10) of $\mathbf{s}$ that is not in $\mathcal{Y}$ (if one exists) to Q. Otherwise do nothing.
5. Go to step 2.

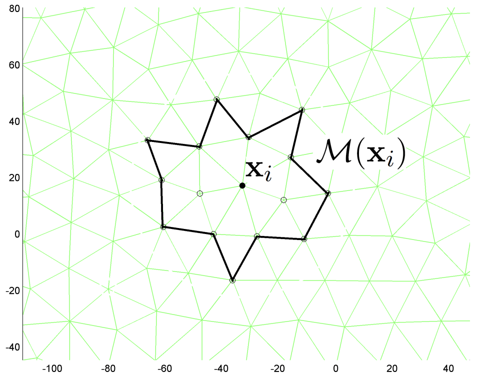
The output of the above algorithm is a set of $(n-1)$-simplices, $\mathcal{M}(\mathbf{x}_i)$ that are AAB with $\mathbf{x}_i$. For $\mathbf{x}_i$ far enough from the boundary $\mathbf{x}_i$, the set $\mathcal{M}(\mathbf{x}_i)$ is also directionally complete. That is, $\mathcal{M}(\mathbf{x}_i)$ contains all possible update directions in $\mathbb{S}^{n-1}$ for $\mathbf{x}_i$.

The MAOUM algorithm follows the same steps as the OUM algorithm, but instead of using the *Near Front* of $\mathbf{x}_i$, $\mathbf{NF}(\mathbf{x}_i)$ to update $\mathbf{x}_i$, the set of $(n-1)$-simplices $MU(\mathbf{x}_i) =$ *Accepted* $\cap \mathcal{M}(\mathbf{x}_i)$ is used. Therefore the MAOUM update is

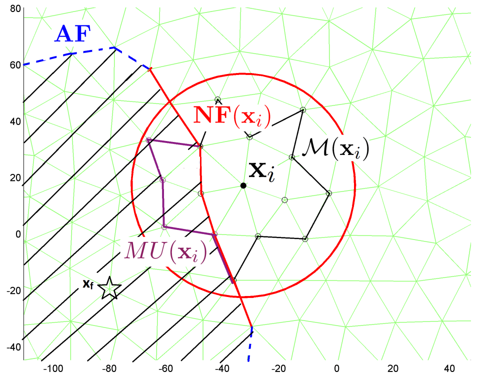
$$\widetilde{C}(\mathbf{x}_i) := \min_{\mathbf{s} \in MU(\mathbf{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \tau(\mathbf{x}_i, \zeta) g(\mathbf{x}_i, \mathbf{u}_\zeta) + \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\mathbf{x}_j^{\mathbf{s}}) \right\}. \tag{4.18}$$

The MAOUM update is used when a vertex $\mathbf{x}_i$ becomes labelled *Considered* and again when any $\mathbf{x}_j \in \mathcal{M}(\mathbf{x}_i)$ becomes labelled *Accepted*. In the second case, only $\mathbf{s} \in MU(\mathbf{x}_i)$ such that $\mathbf{x}_j \in \mathbf{s}$ need to be used for the update.

For meshes where $M = \frac{h_{max}}{h_{min}}$ is small ($M \approx 2$ to 3), the number of $(n-1)$-simplices used in the update of OUM ($\mathbf{NF}(\mathbf{x}_i)$) and MAOUM ($MU(\mathbf{x}_i)$) are comparable in size. See Figures 4.9b and 4.10. For meshes with a large value of M, the size of $\mathbf{NF}(\mathbf{x}_i)$ can be much larger. See Figure 4.11. If a mesh X has a large value of M, MAOUM may perform faster than OUM in computational speed on the same mesh X [5]. The computational complexity



(a) The precomputed set $\mathcal{M}(\mathbf{x}_i)$



(b) $MU(\mathbf{x}_i)$ consists of 5 edges, $\mathbf{NF}(\mathbf{x}_i)$ consists of 6 edges.

Figure 4.9: MAOUM and OUM Updates

of MAOUM is similar to that of OUM [5] given that $M = \frac{h_{max}}{h_{min}}$ is bounded, and is worse off only by a factor of global anisotropy. As a result of monotone acceptance (Theorem 4.2.1) and that vertices can be labelled *Accepted* only if it has a neighbour labelled *Accepted*, the approximate value function $\widetilde{V}$ computed by MAOUM cannot contain strict local minima away from $\mathbf{x}_f$.

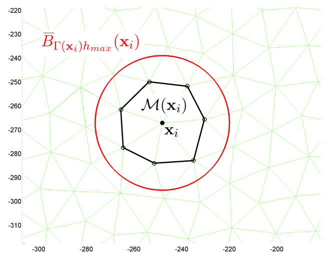
4.5 Buffered Fast Marching Method (BFM)

The Buffered Fast Marching (BFM) method was introduced by Cristiani and Falcone [20] to solve (3.5) approximately using a different semi-Lagrangian scheme than the scheme used in OUM and MAOUM. Rather than assuming the control is constant within each element of the discretization, the control is assumed to be held fixed for a small time $\triangle t$. The dynamic programming principle (3.5) is approximated

$$\widetilde{V}(\mathbf{x}_i) := \widetilde{V}(\mathbf{x}_i - \triangle t\mathbf{u}) - \triangle t g(\mathbf{x}, \mathbf{u}).$$

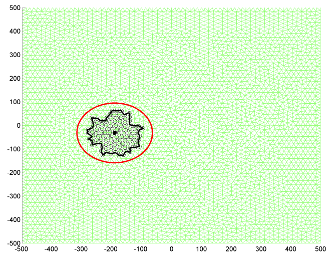
Since the state evolves at unit speed (3.1), the small time $\triangle t$ can be interpreted as a small distance $\triangle x$.

$$\widetilde{V}(\mathbf{x}_i) = \widetilde{V}(\mathbf{x}_i - \triangle x\mathbf{u}) - \triangle x g(\mathbf{x}, \mathbf{u}). \tag{4.19}$$

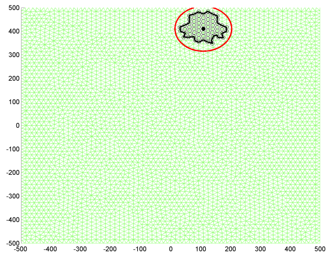


$\overline{B}_{\Gamma(\mathbf{x}_i)h_{max}}(\mathbf{x}_i)$
$\mathcal{M}(\mathbf{x}_i)$
$\mathbf{x}_i$

(a) $\Gamma(\mathbf{x}_i) = 1$

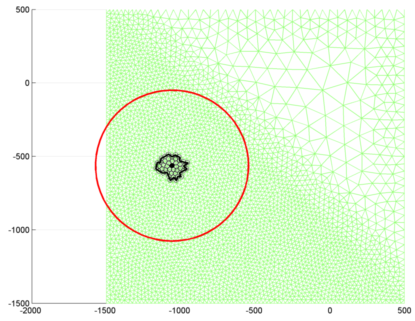


(b) $\Gamma(\mathbf{x}_i) = 5.3$

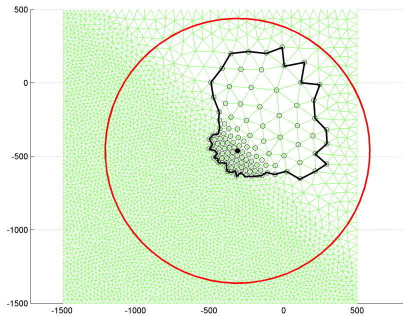


(c) $\Gamma(\mathbf{x}_i) = 3.9$

Figure 4.10: A mesh with small $M = \frac{h_{max}}{h_{min}}$ - MAOUM Update Set $\mathcal{M}(\mathbf{x}_i)$, and OUM, $\overline{B}_{\Gamma(\mathbf{x}_i)h_{max}}(\mathbf{x}_i)$. The computed stencil $\mathcal{M}(\mathbf{x}_i)$ for MAOUM is the outer edge of the points marked with black circles. The ratio $M = \frac{h_{max}}{h_{min}} \approx 2.5$. The vertex $\mathbf{x}_i$ is marked with a black circle.



(a) $\Gamma(\mathbf{x}_i) = 2.9$



(b) $\Gamma(\mathbf{x}_i) = 5.2$

Figure 4.11: A mesh with large $M = \frac{h_{max}}{h_{min}}$, $\mathcal{M}(\mathbf{x}_i)$ compared to $\overline{B}_{\Gamma(\mathbf{x}_i)h_{max}}(\mathbf{x}_i)$ used in OUM for $\Gamma(\mathbf{x}_i) = 2.9$ (left) and $\Gamma(\mathbf{x}_i) = 5.2$ (right). The ratio $M \approx 10$.

The BFM is an extension of the Semi-Lagrangian Fast Marching Method presented in [21], which like FMM, solves the Eikonal equation using the discretization (4.19).

The static HJB equation (3.6) is related to another PDE which solves the minimum cost to reach $\mathbf{x}_f$ for the dynamics given in (3.1). Let the function $W : \Omega \to [0, 1)$ be related to $V : \Omega \to \mathbb{R}_+$ in (3.6) by the Kruzkov transform,

$$W := 1 - e^{-V} \tag{4.20}$$

which when combined with (3.6) gives

$$W(\mathbf{x}) + \max_{\mathbf{u} \in \mathbb{S}^{n-1}} \left\{ -\frac{\mathbf{u}}{g(\mathbf{x}, \mathbf{u})} \cdot \nabla W(\mathbf{x}) \right\} = 1, \tag{4.21}$$

$$W(\mathbf{x}_f) = 0.$$

A similar transform can be made for the approximation $\widetilde{W}$.

$$\widetilde{W} := 1 - e^{-\widetilde{V}} \tag{4.22}$$

where $\widetilde{V}$ is given in (4.19). From (4.19) and (4.22),

$$\widetilde{W}(\mathbf{x}_i) = \min_{\mathbf{u} \in \mathbb{S}^{n-1}} \{ e^{-\triangle x g(\mathbf{x}, \mathbf{u})} (\widetilde{W}(\mathbf{x}_i - \triangle x \mathbf{u}) - 1) \} + 1, \tag{4.23}$$

which will be used as the update in the BFM algorithm.

The value function V is proven to be the unique viscosity solution of (3.6) using the Kruzkov transform (4.20) of V to W in [8, IV. Theorem 2.6]. A priori estimates in terms of rate of convergence of $\widetilde{W}$ to W have been established in [9]. These error estimates are directly applicable to the numerical approximation $\widetilde{V}$ (4.19) for this particular semi-Lagrangian scheme by using the inverse Kruzkov transform,

$$\widetilde{V} = -\ln(1 - \widetilde{W}). \tag{4.24}$$

The scheme is not the same as (4.13) used for OUM and MAOUM. The error estimates are not applicable to the approximate solutions computed by OUM and MAOUM.

The following update procedure is described in [20, 21] for $\overline{\Omega} \subset \mathbb{R}^2$. See Figure 4.12. A linear approximation is made on the circle of radius $\triangle x$ to find $\widetilde{W}(\mathbf{x}_i - \triangle x \mathbf{u})$.

Definition 4.5.1. *The* ***extended neighbours*** *of a grid point $\boldsymbol{x}_{i,j}$ in a grid G are*

$$\mathcal{EN}(\boldsymbol{x}_i) = \{ \boldsymbol{x}_{i\pm1,j}, \boldsymbol{x}_{i,j\pm1}, \boldsymbol{x}_{i\pm1,j\pm1}, \boldsymbol{x}_{i\pm1,j\mp1} \} \cap G. \tag{4.25}$$

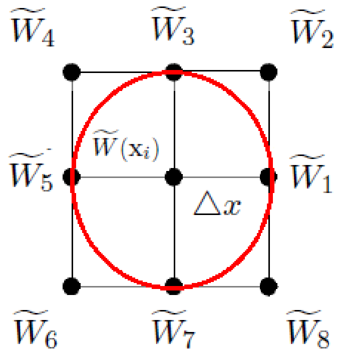


Figure 4.12: In each update for the Buffered Fast Marching method, the minimum is found on the circle, where each quadrant is defined by the plane approximated by the $\widetilde{W}$ values.

The intersection with G in the previous definition handles the edge cases. In each quadrant, let $f(x, y) = ax + by + c$, where a, b, c is defined by linear interpolation according to the value of $\widetilde{W}$ at the grid points. For example in the top-right quadrant,

$$a = \left(\frac{\widetilde{W}_2 - \widetilde{W}_3}{\triangle x}\right), \ b = \left(\frac{\widetilde{W}_2 - \widetilde{W}_1}{\triangle x}\right), \ c = \widetilde{W}_1 - \widetilde{W}_2 + \widetilde{W}_3.$$

By taking $\mathbf{u} = (\cos\theta, \sin\theta)$, $\theta \in [0, \pi/2]$, the value on the circle in the top-right quadrant is

$$\widetilde{W}(\mathbf{x}_i - \triangle x\mathbf{u}) = f(\triangle x \cos\theta, \triangle x \sin\theta) = a\triangle x \cos\theta + b\triangle x \sin\theta + c.$$

Similar formulas can be derived for the top-left quadrant using $\widetilde{W}_3, \widetilde{W}_4, \widetilde{W}_5$, the bottom-left quadrant using $\widetilde{W}_5, \widetilde{W}_6, \widetilde{W}_7$ and the bottom-right quadrant using $\widetilde{W}_7, \widetilde{W}_8, \widetilde{W}_1$. In the update (4.23), the term $\widetilde{W}(\mathbf{x}_i - \triangle x\mathbf{u})$ is approximated using three points yielding greater accuracy. However it is more computationally expensive than the equivalent FMM update [73] on square grids. The minimum of (4.23) is found over all four quadrants.

In addition to using the same definitions of *Accepted*, *Considered* and *Far* as in FMM, a fourth list is introduced known as *Buffer*.

Buffer - The set of vertices between the vertices labelled *Considered* and vertices labelled *Accepted* that are iterated using (4.23).

Characteristics of the solution are computed in the *Buffer* through iteration. This is a trade off to keep updates local as opposed to searching a larger set of $(n-1)$-simplices for the characteristic direction as in OUM and MAOUM.

The BFM algorithm is as follows. Recall that relabelling a vertex removes its previous label.

Initialization

1. Label all grid points of the grid G *Far*, and set $\widetilde{W} = 1$ at all grid points.
2. Set $\widetilde{W}(\mathbf{x}_f) = 0$, and relabel $\mathbf{x}_f$ *Accepted*.
3. Relabel the vertices of the extended neighbours (4.25) of $\mathbf{x}_f$, $\mathcal{EN}(\mathbf{x}_f)$, *Considered*.
4. Iterate the computation (4.23) in $\mathcal{EN}(\mathbf{x}_f)$ until $\widetilde{W}$ at all vertices labelled *Considered* has converged.

Main Loop

1. Let $\overline{\mathbf{x}}_i$ be the vertex with the smallest value of $\widetilde{W}$ with *Considered* label. Relabel $\overline{\mathbf{x}}_i$ *Buffer*. Relabel the vertices of $\mathcal{EN}(\overline{\mathbf{x}}_i)$ in *Far* to *Considered*.
2. Update (4.23) all the vertices in the *Buffer* once.
3. In a copy of the matrix, substitute $\widetilde{W} = 1$ for all the value of the nodes in *Considered*. Then iterate the computation (4.23) for all the vertices in the *Buffer* until the value $\widetilde{W}$ at all of its vertices converge.
4. Repeat the previous step, substituting $\widetilde{W} = \min_{\mathbf{x}_i \in Considered} \widetilde{W}(\mathbf{x}_i)$ for all values with *Considered* label and iterate the computation (4.23) until convergence.
5. Relabel all $\overline{\mathbf{x}}_i$ with *Buffer* label to *Accepted* label such that their values did not change in the previous two steps.
6. If there are still vertices labelled *Considered*, go to step 1, otherwise iterate (4.23) on all the vertices in the *Buffer* until convergence, and terminate.

To obtain $\widetilde{V}$, the inverse Kruzkov (4.24) transform is applied. The implementation of BFM has been limited to square grids. The extension to general simplicial meshes is non-trivial and requires additional computation to reconstruct $\widetilde{W}(\mathbf{x}_i - \triangle x \mathbf{u})$.

4.6 Fast Sweeping Method (FSM)

The Fast Sweeping Method (FSM) was originally proposed in [89] for numerically solving the Eikonal equation on a grid. It has since been extended to solve more general HJB equations [41, 42, 79]. The description for square grids in $\mathbb{R}^2$ with edge length $\triangle x$ will be given here. Let NX and NY denote the number of points on the grid in the x and y directions respectively. Let $\widetilde{V}^x_{min}$ and $\widetilde{V}^y_{min}$ denote the smaller of the values from neighbouring points along the x-axis and y-axis respectively. The local approximation of the Eikonal equation used to solve for $\widetilde{V}$ at $\mathbf{x}_{i,j}$ is

$$\widetilde{V}(\mathbf{x}_{i,j}) := \begin{cases} \min\{\widetilde{V}^x_{min}, \widetilde{V}^y_{min}\} + g_{i,j}\triangle x & \left\|\widetilde{V}^x_{min} - \widetilde{V}^y_{min}\right\| \geq g_{i,j}\triangle x \\ \frac{\widetilde{V}^x_{min}+\widetilde{V}^y_{min}+\sqrt{2g^2_{i,j}\triangle x^2-(\widetilde{V}^x_{min}-\widetilde{V}^y_{min})^2}}{2}, & \left\|\widetilde{V}^x_{min} - \widetilde{V}^y_{min}\right\| < g_{i,j}\triangle x \end{cases} \tag{4.26}$$

where $g_{i,j} = g(\mathbf{x}_{i,j})$, $\widetilde{V}^x_{min} = \min\{\widetilde{V}(\mathbf{x}_{i+1,j}), \widetilde{V}(\mathbf{x}_{i-1,j})\}$ and $\widetilde{V}^y_{min} = \min\{\widetilde{V}(\mathbf{x}_{i,j+1}), \widetilde{V}(\mathbf{x}_{i,j-1})\}$. Instead of finalizing the values of $\widetilde{V}$ one vertex at a time as in previous methods, all vertices have tentative values until the algorithm is finished.

The solution $\widetilde{V}$ at vertices is updated sequentially in a prescribed order known as a sweep. The sweeping orders P for $\mathbb{R}^2$ are

1. Outer Loop: $j = 1 : NY$, Inner Loop: $i = 1 : NX$
2. Outer Loop: $j = 1 : NY$, Inner Loop: $i = NX : -1 : 1$
3. Outer Loop: $j = NY : -1 : 1$, Inner Loop: $i = NX : -1 : 1$
4. Outer Loop: $j = NY : -1 : 1$, Inner Loop: $i = 1 : NX$

The first sweeping order is shown in Figure 4.13. By updating the points in this sequence, the characteristics (from $\mathbf{x}_f$) are updated to the right, then upwards. In higher dimensions $\mathbb{R}^n$, 2^n sweeping orders are used, ith n nested loops in each sweep.

The algorithm is as follows. Let ϵ be a small value.

1. Set $\widetilde{V} = \infty$ on all grid points.
2. Set $\widetilde{V}(\mathbf{x}_f) = 0$. Set $P = 1$.

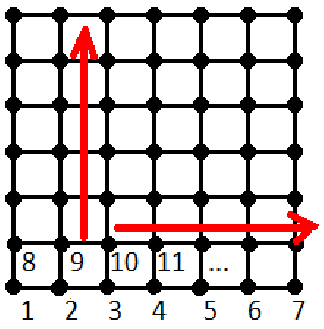


Figure 4.13: The first sweeping order Outer Loop: $j = 1 : NY$, Inner Loop: $i = 1 : NX$. Performing updates in this order will first further the characteristics to the right, then upwards.

3. Set $\widetilde{V}_{old} = \widetilde{V}$.
4. Perform a single sweep P, solving (4.26) for $\widetilde{V}$ at each point according to the order defined above.
5. If $\left\|\widetilde{V} - \widetilde{V}_{old}\right\|_{\infty} < \epsilon$, end. Otherwise, set $P = mod(P + 1, 4)$, and go to Step 3.

Typically, a small value of ϵ above machine precision is used.

The FSM has a computational complexity of $\mathcal{O}(N)$, where N is the number of grid points [89]. The number of sweeps required to reach convergence is independent of the number of points in the discretization. However, the constant in the computational complexity is dependent on the problem. If the characteristics are straight lines emanating outwards from a source point, then only four sweeps (once in each sweeping order) are required to achieve convergence. Suppose a particular characteristic of $\widetilde{V}$ is shown in Figure 4.14. In order to accurately obtain the values of $\widetilde{V}$ along that characteristic, 5 sweeps are required in the specific order as indicated in the figure. Since the algorithm has no information regarding the shape and curvature of the characteristics, the algorithm must cycle through the different sweeping orders, yielding a total of 13 sweeps (or 4 cycles of sweeps) for this example.

An extension has been proposed to extend Fast Sweeping Method to triangular meshes [79]. A preprocessing step is required to order the points in such a manner to mimic the sweeping directions. The Fast Sweeping Method can easily be adapted to solve anisotropic problems by replacing the update (4.26) with the update in OUM (4.13) or BFM (4.23).

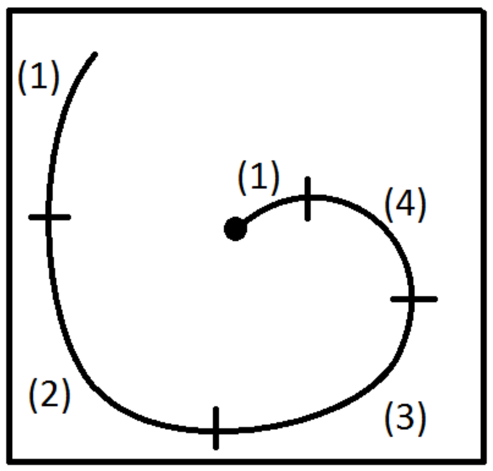


Figure 4.14: Starting from the centre, an example requiring 13 sweeps to achieve the correct solution $\widetilde{V}$ along this characteristic.

4.7 Conclusion

A number of algorithms that solve the dynamic programming principle (3.5) and HJB equation (3.6) approximately were presented. In each of the algorithms, the approximated value function $\widetilde{V}$ was found on the vertices of a discretization.

The Fast Marching Method can solve the Eikonal equation, which can only model direction-independent weights. The Ordered Upwind Method and Monotone Acceptance Ordered Upwind Method can solve problems with direction-dependent weights using an assumption that the control is constant within each element of a simplicial mesh. The Buffered Fast Marching Method is used to solve the static HJB equation with direction-dependent weights with a different assumption that the control is held constant on short intervals of time. The resulting approximated dynamic programming principle is different. Finally, an iterative algorithm, Fast Sweeping Method, which can handle both assumptions on control, was presented.

All of the algorithms presented in this chapter can be used for path planning. In path planning, the gradient of the linear interpolated value function $\nabla\widetilde{V}$ is used to solve (3.8) in place of ∇V, with initial condition $\mathbf{y}(0) = \mathbf{x}_0$. The resulting path is an approximation of the true path. The global minimum is at $\mathbf{x}_f$. The computation required is $\mathcal{O}(1/h_{max})$. As the discretization is refined, the path must traverse more elements. A proof of convergence of $\widetilde{V}$ to V as h_{max} approaches 0 for each of the algorithms discussed in this chapter can be found in the literature. The convergence of $\widetilde{V}$ to V however does not imply the convergence of $\nabla\widetilde{V}$ to ∇V, which is used in path planning (3.8). A convergence result of the gradients remains an open problem. Nonetheless, the numerically approximated gradient $\nabla\widetilde{V}$ can be shown to yield good results in path planning. Simulations involving the Fast Marching Method and Ordered Upwind Method are provided in Chapter 6.

A novel algorithmic improvement, OUM-BD was presented, combining the ideas of bi-directional search on a graph with OUM. It was shown that under a simple modification of the weight function, reversing the roles of $\mathbf{x}_0$ and $\mathbf{x}_f$ yielded an equivalent problem. The two problems are solved concurrently, stopping when they overlap. An area of future investigation is to apply heuristics such as in A^* [35] algorithm for graphs in the continuous setting for OUM. This has already been done for the Fast Marching Method [15].

Chapter 5

Convergence Rate of OUM

In this chapter, a convergence rate in the infinity-norm is shown for the solution provided by OUM on a bounded region of $\mathbb{R}^n$ with prescribed boundary values. Rather than finding the optimal path between one point to another as described previously (3.1), (3.2), the optimal path between a point in a region to the boundary with an exit cost is considered. See Figure 1.2.

An example of this problem is to find an optimal path to leave a building where the exit-cost corresponds to the difficulty of leaving at the boundary. Higher exit-costs would be assigned to walls, and lower costs to doors. With respect to rovers, an application is to find the quickest escape route out of a region that is about to experience an earthquake.

Additional applications that can be solved using OUM for prescribed boundary values include optimal escape from an airplane [3], area patrol and perimeter surveillance [30] and modelling folds in structural geology [38].

The convergence for the OUM problem without rate was shown in [72]. Though analytic convergence rate results for the OUM have not appeared in literature, a number of convergence rates have been proven for different discretization schemes to the same and similar equations. All convergence rates are stated for the infinity-norm. An optimal stopping problem was solved using a simplicial mesh [34] and shown to have a convergence rate of at leaster $\mathcal{O}(|\log h_{max}| \cdot \sqrt{h_{max}})$ as $h_{max} \to 0$. As described in Chapter 4, the semi-Lagrangian scheme used by OUM assumes that the control is held constant within a simplex. The convergence rate result will be shown for this semi-Lagrangian scheme.

An error bound $\mathcal{O}(\triangle t)$ has been proven for a different semi-Lagrangian scheme (4.19) if the controls are assumed to have bounded variation [9]. Analytic results for higher-order convergence rates for higher-order semi-Lagrangian schemes exist [27]. Iterative algorithms [8, 20] have been devised that use this discretization. A comparison between algorithms that use each Semi-Lagrangian scheme is given in Chapter 6. An error bound of $\mathcal{O}(\sqrt{\triangle t})$ for finite-difference schemes to the time-dependent Hamilton-Jacobi-Bellman equations was proven in [19]. An error bound of $\mathcal{O}(\sqrt{h_{max}})$ was shown for an approximation to the Eikonal equation with discontinuous weight function [22].

In Section 5.1, the boundary value problem will be presented. Following some definitions and results from [5], a numerical Hamiltonian will be defined and shown to be monotonic and consistent with the Hamiltonian of the problem in Section 5.2. Finally, in Section 5.3, the approximate solution provided by OUM is proven to converge to the viscosity solution of the static HJB with prescribed boundary values at a rate of at least $\mathcal{O}(\sqrt{h_{max}})$, as $h_{max} \to 0$. The result shown here is for a different algorithm, namely OUM. The proof is based on the FMM result in [61], but unlike the result in that reference, the weight can depend on position and direction and the boundary function can depend on position. The result holds on a simplicial mesh, which are better suited towards handling regions with complex geometries. A finer discretization may be required to obtain the same accuracy when the discretization is restricted to grids. Conclusions are provided in Section 5.4.

5.1 Boundary Value Problem

The presentation of the problem here is similar to that of Section 3.1. Though some of the notation is repeated, the problem presented here is different. The dynamics for a trajectory $\mathbf{y}(\cdot) : \mathbb{R}_+ \to \mathbb{R}^n$ and control $\mathbf{u}(\cdot) \in \mathcal{U}$ are the same as (3.1),

$$\dot{\mathbf{y}}(t) = \mathbf{u}(t), \mathbf{y}(0) = \mathbf{x}_0, \mathbf{x}_0 \in \Omega. \tag{5.1}$$

The control problem is to steer $\mathbf{y}(\cdot)$ from $\mathbf{x}_0 \in \Omega$ to any point on the boundary $\mathbf{x}_f \in \partial\Omega$. As before, the exit-time is the first time when the control problem is solved from $\mathbf{x}_0$ with a control $\mathbf{u}(\cdot)$.

Definition 5.1.1. *The* ***exit-time*** $T : \Omega \times \mathcal{U} \to \mathbb{R}_+$ *is the first time* $\boldsymbol{y}_{\boldsymbol{x}_0}(\cdot)$ *reaches* $\boldsymbol{x}_f \in \partial\Omega$ *under the influence of the control* $\boldsymbol{u}(\cdot)$*,*

$$T(\boldsymbol{x}_0, \boldsymbol{u}(\cdot)) = \inf\{t | \boldsymbol{y}_{\boldsymbol{x}_0}(t) \in \partial\Omega\}. \tag{5.2}$$

The cost for the boundary value problem is $Cost : \Omega \times \mathcal{U}$ is

$$Cost(\mathbf{x}_0, \mathbf{u}(\cdot)) = \int_0^{T(\mathbf{x}_0, \mathbf{u}(\cdot))} g(\mathbf{y}_{\mathbf{x}_0}(s), \mathbf{u}(s)) ds + q(\mathbf{y}_{\mathbf{x}_0}(T(\mathbf{x}_0, \mathbf{u}(\cdot)))), \text{ for } \mathbf{x}_0 \in \Omega \tag{5.3}$$

where $q : \partial\Omega \to \mathbb{R}$ is the boundary exit-cost and $g : \overline{\Omega} \times \mathbb{S}^{n-1} \to \mathbb{R}_+$ is the weight. The optimal control problem is to find a control $\mathbf{u}^*(\cdot)$ that minimizes (5.3).

As before the value function is the minimizing cost of the optimal control at $\mathbf{x} \in \overline{\Omega}$,

$$V(\mathbf{x}) = \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} Cost(\mathbf{x}, \mathbf{u}(\cdot)). \tag{5.4}$$

The dynamic programming principle (3.5) is satisfied. For $h > 0$, $t \geq 0$, such that $0 \leq t + h \leq T(\mathbf{x}, \mathbf{u}^*(\cdot))$,

$$V(\mathbf{y}(t)) = \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \int_t^{t+h} g(\mathbf{y}(\tau), \mathbf{u}(\tau))d\tau + V(\mathbf{y}(t+h)) \right\}. \tag{5.5}$$

The Hamiltonian is defined $H : \Omega \times \mathbb{R}^n \to \mathbb{R}$

$$H(\mathbf{x}, \mathbf{p}) = - \min_{\mathbf{u}\in\mathbb{S}^{n-1}} \{\mathbf{p} \cdot \mathbf{u} + g(\mathbf{x}, \mathbf{u})\}. \tag{5.6}$$

A similar static HJB to (3.6) but with boundary condition defined on the boundary of a region (rather than at a source point) is obtained,

$$\begin{gathered} \min_{\mathbf{u}\in\mathbb{S}^{n-1}} \{\nabla V(\mathbf{x}) \cdot \mathbf{u} + g(\mathbf{x}, \mathbf{u})\}, \mathbf{x} \in \Omega, \\ V(\mathbf{x}) = q(\mathbf{x}), \mathbf{x} \in \partial\Omega. \end{gathered} \tag{5.7}$$

Theorem 3.3.5 can be used again to assert that there are no local minima on Ω. In the context of path planning, the optimal escape trajectory for this boundary value problem can be found by solving (3.8), even though the final point $\mathbf{x}_f$ is not known beforehand.

5.1.1 Assumptions on the problem

A few assumptions are made on the boundary value problem (5.1), (5.3).

For V to be continuous on $\overline{\Omega}$, continuity between V on Ω and q on $\partial\Omega$ must be established. Let $L : \overline{\Omega} \times \overline{\Omega}$ be

$$L(\mathbf{x}_1, \mathbf{x}_2) = \inf_{\mathbf{u}(\cdot)\in\mathcal{U}} \left\{ \int_0^{\tau} g(\mathbf{y}_{\mathbf{x}_1}(s), \mathbf{u}(s))ds \;\middle|\; \mathbf{y}_{\mathbf{x}_1}(\tau) = \mathbf{x}_2, \mathbf{y}_{\mathbf{x}_1}(t) \in \overline{\Omega}, t \in (0, \tau) \right\}. \tag{5.8}$$

The function L is the lowest cost to reach $\mathbf{x}_2$ from $\mathbf{x}_1$, (ignoring the exit-cost on the boundary q if either $\mathbf{x}_1$ or $\mathbf{x}_2$ are on $\partial\Omega$).

Definition 5.1.2. *For (5.7), the exit-cost q is* ***compatible*** *(with the continuity of V) if*

$$q(\boldsymbol{x}_1) - q(\boldsymbol{x}_2) \leq L(\boldsymbol{x}_1, \boldsymbol{x}_2) \tag{5.9}$$

for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \partial\Omega$.

For example, consider $\overline{\Omega} = [-1, 1]$ with $g(x, u) \equiv 1$ and boundary conditions $q(-1) = 0$, and $q(1) = 3$. The HJB equation is

$$|V'(x)| = 1. \tag{5.10}$$

There are no weak solutions (Definition 3.2.2) V that satisfy both the boundary conditions q and (5.10). So q is not compatible (with the continuity of V).

The boundary value optimal control problem (5.1), (5.3) will be assumed to satisfy the following statements. Recall the definition of Lipschitz-continuity from Definition 3.2.1.

(P1) *Compatibility of the boundary function $q : \partial\Omega \to \mathbb{R}$.* The boundary function q is compatible with the continuity of V (Definition 5.1.2).

(P2) *The weight function g is positive and bounded.* A similar assumption (3.3) was used earlier.

$$0 < G_{min} \leq g_{min}(\mathbf{x}) \leq g(\mathbf{x}, \mathbf{u}) \leq g_{max}(\mathbf{x}) \leq G_{max} < \infty \tag{5.11}$$

for all $\mathbf{x} \in \overline{\Omega}$ and $\mathbf{u} \in \mathbb{S}^{n-1}$.

(P3) *Lipschitz-continuity in $\boldsymbol{x}$ of weight function g.* There exists $L_g > 0$ such that for $\mathbf{x}_1, \mathbf{x}_2 \in \overline{\Omega}$ and $\mathbf{u} \in \mathbb{S}^{n-1}$,

$$|g(\mathbf{x}_1, \mathbf{u}) - g(\mathbf{x}_2, \mathbf{u})| \leq L_g \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|. \tag{5.12}$$

(P4) *Convexity of region $\overline{\Omega}$.* For $\mathbf{x}_1, \mathbf{x}_2 \in \overline{\Omega}$,

$$\hat{\mathbf{x}} = \lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \in \overline{\Omega}, \qquad \text{for all } \lambda \in (0, 1).$$

(P5) *Convexity of Speed Profile $\mathcal{U}_g(\boldsymbol{x})$.* The set

$$\mathcal{U}_g(\mathbf{x}) = \left\{ \frac{\mathbf{u}}{g(\mathbf{x}, \mathbf{u})} \middle| \mathbf{u} \in \mathbb{S}^{n-1} \right\}$$

is convex. (This is needed to guarantee uniqueness of the optimizing direction in the approximated problem provided ∇V exists [5, 72])

Using the above assumptions, the Lipschitz-continuity of q can be shown.

Lemma 5.1.3. *The boundary function $q : \partial\Omega \to \mathbb{R}$ is Lipschitz-continuous.*

Proof. Let $\mathbf{x}_1, \mathbf{x}_2 \in \partial\Omega$ and $\overline{\mathbf{u}} = \frac{\mathbf{x}_2 - \mathbf{x}_1}{\|\mathbf{x}_2 - \mathbf{x}_1\|}$ be the control of the trajectory $\overline{\mathbf{y}}_{\mathbf{x}_1}(\cdot)$. From **(P4)**, $\overline{\mathbf{y}}_{\mathbf{x}_1}(\cdot)$ from $\mathbf{x}_1$ to $\mathbf{x}_2$ must lie entirely in $\overline{\Omega}$. If $\overline{\mathbf{y}}_{\mathbf{x}_1}(\cdot) \subset \Omega$ from $\mathbf{x}_1$ and $\mathbf{x}_2$, then (**P1**) yields

$$\begin{aligned} q(\mathbf{x}_1) - q(\mathbf{x}_2) &\leq L(\mathbf{x}_1, \mathbf{x}_2), \\ &\leq \int_0^{\|\mathbf{x}_1 - \mathbf{x}_2\|} g(\overline{\mathbf{y}}_{\mathbf{x}_1}(s), \overline{\mathbf{u}}(s)) ds, \\ &\leq G_{max} \|\mathbf{x}_1 - \mathbf{x}_2\| . \end{aligned}$$

Otherwise, $\overline{\Omega}$ is not strictly convex, and $\overline{\mathbf{y}}_{\mathbf{x}_1}(\cdot) \subset \partial\Omega$. For any $\epsilon > 0$, there exists a trajectory of length $(1+\epsilon)\|\mathbf{x}_1 - \mathbf{x}_2\|$ from $\mathbf{x}_1$ to $\mathbf{x}_2$ entirely in Ω except at $\mathbf{x}_1$ and $\mathbf{x}_2$. Let this trajecttory be denoted $\widetilde{\mathbf{y}}_{\mathbf{x}_1}(\cdot)$. Hence,

$$\begin{aligned} q(\mathbf{x}_1) - q(\mathbf{x}_2) &\leq L(\mathbf{x}_1, \mathbf{x}_2) \\ &\leq \int_0^{\|\mathbf{x}_1 - \mathbf{x}_2\|(1+\epsilon)} g(\widetilde{\mathbf{y}}_{\mathbf{x}_1}(s), \overline{\mathbf{u}}(s)) ds, \\ &\leq G_{max}(1+\epsilon) \|\mathbf{x}_1 - \mathbf{x}_2\| . \end{aligned}$$

The calculation for $q(\mathbf{x}_2) - q(\mathbf{x}_1)$ is identical. Hence q is Lipschitz-continuous. □

Note that a suitable Lipschitz constant for q is any value above G_{max}. Since q is Lipschitz-continuous over a compact subset of $\mathbb{R}^n$, it is continuous and hence there exists $q_{min}, q_{max} \in \mathbb{R}$ such that

$$q_{min} \leq q(\mathbf{x}) \leq q_{max}. \tag{5.13}$$

5.1.2 OUM for the Boundary Value Problem

Let the following statements hold for the mesh $X \subset \mathbb{R}^n$ on which the approximated solution $\widetilde{V}$ is found.

(M1) Assume $M > 0$ is fixed. All meshes discussed in this chapter satisfy $M \geq \frac{h_{max}}{h_{min}} \geq 1$. The value M characterizes the worst-case degeneracy for a mesh X, as well as even refinement of simplices in a mesh.

(M2) The region $\overline{\Omega}$ is contained (Definition 4.1.8) in the mesh X. See Figure 4.1.

(M3) The mesh X is bounded and has a finite number of vertices $\{\mathbf{x}_i\} \subset X$.

Denote X_j, $0 \leq j \leq n$ the set of j-simplices of X.

From (**M3**), there is a maximal set that can be defined using the barycentric coordinates of Ξ_n (4.1) and n-simplices $\mathbf{s} \in X \subset \mathbb{R}^n$. Define this set

$$\overline{\Omega}_X = \left\{ \bigcup_{\mathbf{s} \in X_n} \bigcup_{\zeta \in \Xi_n} \sum_{j=0}^{n} \zeta_j \mathbf{x}_j^{\mathbf{s}} \right\}. \tag{5.14}$$

From (**M2**), $\overline{\Omega} \subseteq \overline{\Omega}_X$.

The OUM can be used to find $\widetilde{V} : \{\mathbf{x}_i\} \subset X \to \mathbb{R}$, an approximation of V for the boundary value problem with only a slight modification from the algorithm presented in Section 4.3. Recall $\Omega^c = \mathbb{R}^n \backslash \Omega$. Steps 2 and 3 in the OUM algorithm Section 4.3 are replaced with

2. For vertices $\mathbf{x}_i \in X \cap \Omega^c$, let

   $$\hat{\mathbf{x}} = \arg \min_{\widetilde{\mathbf{x}} \in \partial\Omega} \|\mathbf{x}_i - \widetilde{\mathbf{x}}\|,$$

   and set $\widetilde{V}(\mathbf{x}_i) = q(\hat{\mathbf{x}})$. Relabel all vertices $\mathbf{x}_i \in X \cap \Omega^c$ *Accepted.*

3. Relabel all *Far* label neighbours $\mathbf{x}_i$ of vertices labelled *Accepted* to *Considered.* Compute the values $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$ according to (4.14).

All other steps of the OUM remain the same. See Figure 5.1. The other algorithms defined in Chapter 4 can be rewritten into the boundary value problem by setting the known boundary condition $q(\mathbf{x})$ on vertices on and outside the boundary of the region $\overline{\Omega}$.

The domain of the value function V and g are extended from $\overline{\Omega}$ to $\overline{\Omega}_X$. For $\mathbf{x} \in \overline{\Omega}^c \cap \overline{\Omega}_X$, let

$$\hat{\mathbf{x}} = \arg \min_{\widetilde{\mathbf{x}} \in \partial\Omega} \|\mathbf{x} - \widetilde{\mathbf{x}}\|, \text{ and } V(\mathbf{x}) = q(\hat{\mathbf{x}}), \text{ and } g(\mathbf{x}, \mathbf{u}) = g(\hat{\mathbf{x}}, \mathbf{u}).$$

As a result, H in (5.7) is now also defined on $\overline{\Omega}_X \times \mathbb{R}^n$.

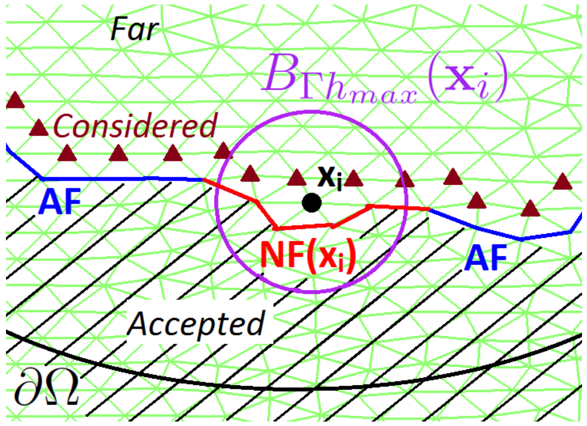


Figure 5.1: OUM Labels - An example for $\overline{\Omega} \subset \mathbb{R}^2$ for the boundary value problem. The vertex $\mathbf{x}_i$ with *Considered* label is updated from the set of directions provided by $\mathbf{NF}(\mathbf{x}_i)$. Vertices labelled *Accepted* are shaded, including vertices on the edges that make up **AF** and the *Near Front* of $\mathbf{x}_i$, $\mathbf{NF}(\mathbf{x}_i)$. Let $\Gamma = \frac{G_{max}}{G_{min}}$ and $B_{\Gamma h_{max}}(\mathbf{x}_i)$ have radius Γh_{max} and centre $\mathbf{x}_i$. Vertices labelled *Considered* are marked with a triangle. Vertices outside Ω are also labelled *Accepted*. Unmarked vertices are labelled *Far*.

5.2 Monotonicity and Consistency of the Numerical Hamiltonian

An approximation of the Hamiltonian H (5.6) on the vertices of a mesh X will be defined. It will be shown to be both monotonic and consistent with the exact Hamiltonian (5.6). The optimizing direction of the approximated problem from OUM must first be presented.

Let $\mathbf{x}_i$ be the vertex labelled *Considered* that is about to be relabelled *Accepted* in Step 4 of the OUM algorithm. Let $\overline{\mathbf{NF}}(\mathbf{x}_i)$ be the *Near Front* of $\mathbf{x}_i$ (4.11) at this instant. The distance $\tau_{\mathbf{s}}$ and direction $\widetilde{\mathbf{u}}_{\mathbf{s}}$ to $\mathbf{x} \in \mathbf{s} = \mathbf{x}_0^{\mathbf{s}}\mathbf{x}_1^{\mathbf{s}} \cdots \mathbf{x}_{n-1}^{\mathbf{s}}$ from $\mathbf{x}_i$ are defined

$$\tau_{\mathbf{s}}(\mathbf{x}_i, \zeta) := \left\| \sum_{j=0}^{n-1} \zeta_j \mathbf{x}_j^{\mathbf{s}} - \mathbf{x}_i \right\|, \text{ and } \widetilde{\mathbf{u}}_{\mathbf{s}}(\mathbf{x}_i, \zeta) := \frac{\sum_{j=0}^{n-1} \zeta_j \mathbf{x}_j^{\mathbf{s}} - \mathbf{x}_i}{\tau_{\mathbf{s}}(\mathbf{x}_i, \zeta)} \tag{5.15}$$

where $\zeta = (\zeta_0, \zeta_1, ..., \zeta_{n-1}) \in \Xi_{n-1}$ are the barycentric coordinates (4.1) of $\mathbf{x} \in \mathbf{s}$.

Definition 5.2.1. *The* ***approximated characteristic direction*** *at* $\boldsymbol{x}_i \in X \cap \Omega$ *from* *the OUM algorithm $\widetilde{\boldsymbol{u}}^*_{\boldsymbol{s}} : X \cap \Omega \times \Xi_{n-1} \to \mathbb{S}^{n-1}$ is the direction that minimizes*

$$\widetilde{V}(\boldsymbol{x}_i) = \min_{s \in \overline{\boldsymbol{NF}}(\boldsymbol{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ g(\boldsymbol{x}_i, \widetilde{\boldsymbol{u}}_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta))\tau_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta) + \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\boldsymbol{x}^{\boldsymbol{s}}_j) \right\}. \tag{5.16}$$

Let minimizers of (5.16) be $\boldsymbol{s}^*$, ζ^*, and let $\widetilde{\boldsymbol{x}} = \sum_{j=0}^{n-1} \zeta^*_j \boldsymbol{x}^{\boldsymbol{s}^*}_j$ be on $\boldsymbol{s}^* \in \overline{\boldsymbol{NF}}(\boldsymbol{x}_i)$. Then

$$\widetilde{\boldsymbol{u}}^*_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta^*) = \frac{\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i}{\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i\|}. \tag{5.17}$$

Definition 5.2.2. *[5, Section 2.2] The set of $(n-1)$-simplices $\mathcal{S} \subset X$ is **directionally complete** for a vertex $\boldsymbol{x}_i \in X$ if for all $\boldsymbol{u} \in \mathbb{S}^{n-1}$ there exists $\boldsymbol{x} \in \boldsymbol{s}$ where $\boldsymbol{s} \in \mathcal{S}$ such that*

$$\boldsymbol{u} = \frac{\boldsymbol{x} - \boldsymbol{x}_i}{\|\boldsymbol{x} - \boldsymbol{x}_i\|}.$$

Definition 5.2.3. *An **update** for a vertex $\boldsymbol{x}_i \in X$ from a set of $(n-1)$-simplices $\mathcal{S} \subset X$ is*

$$\min_{s \in \mathcal{S}} \min_{\zeta \in \Xi_{n-1}} \left\{ g(\boldsymbol{x}_i, \widetilde{\boldsymbol{u}}_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta))\tau_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta) + \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\boldsymbol{x}^{\boldsymbol{s}}_j) \right\}. \tag{5.18}$$

The update from OUM for $\mathbf{x}_i$ is the update for $\mathbf{x}_i$ over $\mathcal{S} = \overline{\mathbf{NF}}(\mathbf{x}_i)$ (5.16).

Definition 5.2.4. *An **updating stencil** of $\boldsymbol{x}_i$, $US(\boldsymbol{x}_i)$ is a set of $(n-1)$-simplices of X that contain the point $\widetilde{\boldsymbol{x}}$ on the $(n-1)$-simplex $\boldsymbol{s} \in US(\boldsymbol{x}_i)$ such that*

$$\widetilde{\boldsymbol{u}}^*_{\boldsymbol{s}}(\boldsymbol{x}_i) = \frac{\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i}{\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i\|}, \tag{5.19}$$

where $\widetilde{\boldsymbol{u}}^*_{\boldsymbol{s}}(\boldsymbol{x}_i)$ is the approximated characteristic direction at $\boldsymbol{x}_i$ (Definition 5.2.1) from the OUM.

For example, $\overline{\mathbf{NF}}(\mathbf{x}_i)$ is an updating stencil of $\mathbf{x}_i$.

The numerical Hamiltonian will now be defined.

Definition 5.2.5. *Let $\phi : \{\boldsymbol{x}_i\} \subset X \cap \Omega \to \mathbb{R}$. The **numerical Hamiltonian** of the OUM, $\widetilde{H} : X \cap \Omega \times \mathbb{R} \to \mathbb{R}$ is*

$$\widetilde{H}[US(\boldsymbol{x}_i), \phi[US(\boldsymbol{x}_i)]](\boldsymbol{x}_i, \mu) = -\min_{s \in US(\boldsymbol{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \frac{\sum_{j=0}^{n-1} \zeta_j \phi(\boldsymbol{x}^{\boldsymbol{s}}_j) - \mu}{\tau_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta)} + g(\boldsymbol{x}_i, \widetilde{\boldsymbol{u}}_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta)) \right\}, \tag{5.20}$$

where $US(\boldsymbol{x}_i) \subset X$ is an updating stencil of $\boldsymbol{x}_i \in X$, $\boldsymbol{s} = \boldsymbol{x}_0^{\boldsymbol{s}} \cdots \boldsymbol{x}_{n-1}^{\boldsymbol{s}} \in US(\boldsymbol{x}_i)$. The argument $\phi[US(\boldsymbol{x}_i)]$ of $\widetilde{H}$ denotes the use of the values of ϕ on the vertices that make up the $(n-1)$-simplices of $US(\boldsymbol{x}_i)$ in the optimization of (5.20).

Since $\mathbf{x}_i$ is an argument of $\widetilde{H}$, the notation for the numerical Hamiltonian $\widetilde{H}$ will be abbreviated $\widetilde{H}[US, \phi](\mathbf{x}_i, \mu)$.

The numerical HJB equation is

$$\min_{\mathbf{s} \in \overline{\mathbf{NF}}(\mathbf{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \frac{\sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\mathbf{x}_j^{\mathbf{s}}) - \widetilde{V}(\mathbf{x}_i)}{\tau_{\mathbf{s}}(\mathbf{x}_i, \zeta)} + g(\mathbf{x}_i, \widetilde{\mathbf{u}}_{\mathbf{s}}(\mathbf{x}_i, \zeta)) \right\} = 0, \tag{5.21}$$

and is equivalent to

$$\widetilde{H}[\overline{\mathbf{NF}}, \widetilde{V}](\mathbf{x}_i, \widetilde{V}(\mathbf{x}_i)) = 0.$$

The approximated value function $\widetilde{V}$ is said to satisfy (5.21) if the above is satisfied for all $\mathbf{x}_i \in X \cap \Omega$.

The following theorem states that the optimizing ζ^* and $\mathbf{s}^*$ in (5.16) is also optimal for (5.20). The statement **(P5)** in Section 5.1.1 is assumed in the proof to provide a unique solution to (5.21).

Theorem 5.2.6. *[5, Prop 5.3] The solution $\widetilde{\mu}$ to $\widetilde{H}[US, \widetilde{V}](\boldsymbol{x}_i, \widetilde{\mu}) = 0$ with $\widetilde{H}$ defined by (5.20) is unique, and is given by*

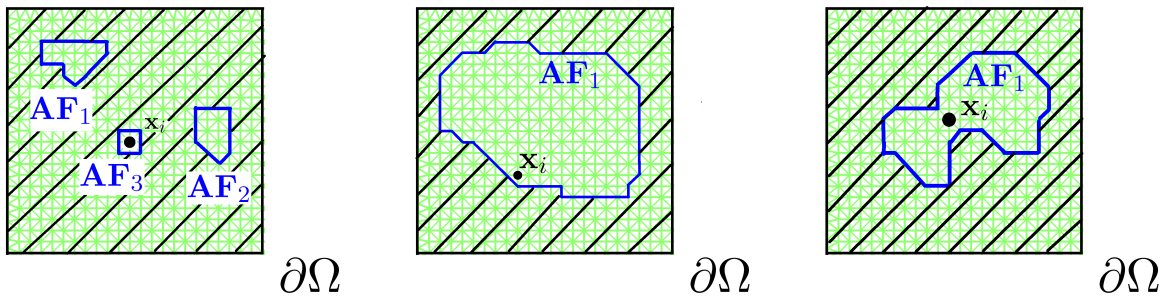
$$\widetilde{\mu} = \min_{\boldsymbol{s} \in US(\boldsymbol{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \tau(\boldsymbol{x}_i, \zeta) g(\boldsymbol{x}_i, \boldsymbol{u}_\zeta) + \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\boldsymbol{x}_j^{\boldsymbol{s}}) \right\}, \tag{5.22}$$

where $\boldsymbol{s} = \boldsymbol{x}_0^{\boldsymbol{s}} \boldsymbol{x}_1^{\boldsymbol{s}} \cdots \boldsymbol{x}_{n-1}^{\boldsymbol{s}} \in US(\boldsymbol{x}_i)$. Furthermore, $\boldsymbol{s}^* \in US(\boldsymbol{x}_i)$ and $\zeta^* = (\zeta_0^*, \zeta_1^*, ..., \zeta_{n-1}^*) \in \Xi_{n-1}$ minimize (5.22) if and only if $\boldsymbol{s}^*$ and ζ^* are minimizers of the expression in (5.20). Hence $\widetilde{\mu} = \widetilde{V}(\boldsymbol{x}_i)$.

Definition 5.2.7. *For every $\boldsymbol{x}_i \in X \cap \Omega$, let $S(\boldsymbol{x}_i)$ be an updating stencil such that*

1. *$\overline{\boldsymbol{NF}}(\boldsymbol{x}_i) \subseteq S(\boldsymbol{x}_i)$.*
2. *$S(\boldsymbol{x}_i)$ is directionally complete for $\boldsymbol{x}_i$.*
3. *For all $(n-1)$-simplices $\boldsymbol{s} \in S(\boldsymbol{x}_i)$, if a point $\boldsymbol{x} \in \boldsymbol{s}$, then*

$$\|\boldsymbol{x} - \boldsymbol{x}_i\| \leq (2\Gamma + 1) h_{max}.$$



(a) After relabelling $\mathbf{x}_i$ *Accepted*, $\mathbf{AF}_3$ is no longer part of $\mathbf{AF}$. Note that $\mathbf{AF}_1$ and $\mathbf{AF}_2$ are unaffected, and still contain only non-*Accepted* vertices in their interiors.

(b) After relabelling $\mathbf{x}_i$ *Accepted*, the other vertices in the interior of $\mathbf{AF}_1$ are still not yet *Accepted*

(c) Relabelling $\mathbf{x}_i$ *Accepted* splits $\mathbf{AF}_1$ into two regions, each only containing not yet *Accepted* vertices in their interiors.

Figure 5.2: Lemma 5.2.8: The vertices that lie in the interior of the region(s) contained by $\mathbf{AF}$ are not labelled *Accepted*. In each case above, the vertex $\mathbf{x}_i$ is about to be relabelled *Accepted*. After $\mathbf{x}_i$ has been relabelled *Accepted*, the updated $\mathbf{AF}$ must still contain only vertices that are not yet *Accepted* in its interior.

4. $\widetilde{H}[S(\boldsymbol{x}_i), \widetilde{V}[S(\boldsymbol{x}_i)]](\boldsymbol{x}_i, \widetilde{V}(\boldsymbol{x}_i)) = \widetilde{H}[\overline{\boldsymbol{NF}}(\boldsymbol{x}_i), \widetilde{V}[\overline{\boldsymbol{NF}}(\boldsymbol{x}_i)]](\boldsymbol{x}_i, \widetilde{V}(\boldsymbol{x}_i))$

An updating stencil $S(\mathbf{x}_i)$ satisfying Definition 5.2.7 will be constructed. For $r \in \mathbb{R}_+$, define

$$B_r(\mathbf{x}) = \{\widetilde{\mathbf{x}} \in \mathbb{R}^n | \, \|\mathbf{x} - \widetilde{\mathbf{x}}\| < r\}$$

and let $\overline{B}_r(\mathbf{x})$ denote its closure. Recall a set $A \subset \mathbb{R}^n$ has no holes if its complement A^c is connected.

Lemma 5.2.8. *Prior to each instance of Step 4 of the OUM algorithm, $(n-1)$-simplices of $\boldsymbol{AF}$ form the boundaries $\boldsymbol{AF}_j$ of j bounded open subsets $\Omega_{\boldsymbol{AF}_j} \subset \mathbb{R}^n$, such that each $\Omega^c_{\boldsymbol{AF}_j}$ is connected and $\bigcup_{k=1}^j \boldsymbol{AF}_j = \boldsymbol{AF}$.*

Furthermore, if $\boldsymbol{x}_i \in X \cap \Omega_{\boldsymbol{AF}_k}$, then

1. *the set of $(n-1)$-simplices $\boldsymbol{AF}_k$ is directionally complete for $\boldsymbol{x}_i$, and*

2. *$\boldsymbol{x}_i$ is not labelled Accepted.*

Proof:

The property that the set of $(n-1)$-simplices $\mathbf{AF}$ enclose a finite number of regions where only non-*Accepted* vertices lie in their interiors is described in the above lemma. Each case described in the proof is shown in Figure 5.2.

At the initialization (Steps 1-3) of the OUM algorithm, only $\mathbf{x}_i \in X \cap \Omega^c$ are labelled *Accepted*. Since $\overline{\Omega}$ is contained in X (**M2**)and convex (**P4**), $j = 1$ and $\mathbf{AF}_1 = \mathbf{AF}$ form a single boundary that encloses $\Omega_{\mathbf{AF}_1} \supseteq \Omega$. The set $\mathbf{AF}$ for $\mathbf{x}_i \in X \cap \Omega_{\mathbf{AF}_1} = X \cap \Omega$ is directionally complete. None of the vertices in $X \cap \Omega$ are labelled *Accepted*.

The *Accepted Front* and $\mathbf{AF}$ change only in Step 4 of the OUM. Proof by induction will be used. The result of the lemma is assumed to hold prior to step 4 of the OUM. Let vertex $\mathbf{x}_i \in X \cap \Omega_{\mathbf{AF}_k}$ be relabelled *Accepted* for some $1 \leq k \leq j$. All other regions $\Omega_{\mathbf{AF}_{j \neq k}}$ will remain unchanged. Only $\mathbf{AF}_k$ and $\Omega_{\mathbf{AF}_k}$ must be considered.

If $\mathbf{x}_i$ has no neighbours in $X \cap \Omega_{\mathbf{AF}_k}$, then the resulting $\Omega_{\mathbf{AF}_k}$ and $X \cap \Omega_{\mathbf{AF}_k}$ are both empty. The lemma is trivially satisfied.

If $\mathbf{x}_i$ has a neighbour in $X \cap \Omega_{\mathbf{AF}_k}$ which must be labelled *Considered*, then $\mathbf{x}_i$ is added to the *Accepted Front*. If $\Omega_{\mathbf{AF}_k}$ remains an open connected subset of $\mathbb{R}^n$, $\mathbf{x}_m \in X \cap \Omega_{\mathbf{AF}_k} \backslash \{\mathbf{x}_i\}$, $\mathbf{AF}_k$ remains directionally complete and $\mathbf{x}_m$ is not labelled *Accepted*. The lemma is satisfied.

Otherwise, $\Omega_{\mathbf{AF}_k}$ is no longer an open connected subset of $\mathbb{R}^n$. Thus, $\Omega_{\mathbf{AF}_k}$ has been split into $p \geq 2$ non-intersecting open connected regions $\Omega_{\mathbf{AF}_{k1}}$,...,$\Omega_{\mathbf{AF}_{k2}}$, $\Omega_{\mathbf{AF}_{kp}}$ with a subset of the resultant $\mathbf{AF}_{ki}$ as the boundary of each. Vertices $\mathbf{x}_m \in X \cap \Omega_{\mathbf{AF}_k} \backslash \{\mathbf{x}_i\}$ are still not labelled *Accepted*, and $\mathbf{AF}_{kl}$ is directionally complete for $\mathbf{x}_m \in \Omega_{\mathbf{AF}_{kl}}$. $\square$

Consider the OUM algorithm at the instant the vertex $\mathbf{x}_i \in X \cap \Omega$ is about to be relabelled *Accepted*. Recall the *Near Front* of $\mathbf{x}_i$ at this instant is denoted $\overline{\mathbf{NF}}(\mathbf{x}_i)$.

Definition 5.2.9. *Assume the OUM algorithm is at the instant that $\boldsymbol{x}_i$ labelled Considered be the vertex about to be relabelled Accepted. Let $\overline{\boldsymbol{AF}}(\boldsymbol{x}_i) \subseteq \boldsymbol{AF}$ be the set of $(n-1)$-simplices that contain $\boldsymbol{x}_i$ and satisfies Lemma 5.2.8.*

An updating stencil $S(\mathbf{x}_i)$ that satisfies Definition 5.2.7 will now be constructed for each $\mathbf{x}_i \in X \cap \Omega^c$. Two cases are considered.

Case 1: The set of $(n-1)$-simplices $\overline{\mathbf{AF}}(\mathbf{x}_i)$ is entirely inside $\overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i)$. Let

$$S(\mathbf{x}_i) = \overline{\mathbf{AF}}(\mathbf{x}_i) \cup \overline{\mathbf{NF}}(\mathbf{x}_i). \tag{5.23}$$

Case 2: The set of $(n-1)$-simplices $\overline{\mathbf{AF}}(\mathbf{x}_i)$ is not entirely inside $\overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i)$. Since $\mathbf{x}_i$ is labelled *Considered*, and the longest edge length of X is h_{max}, there exists a $(n-1)$-simplex $\mathbf{s} \in \overline{\mathbf{AF}}(\mathbf{x}_i)$ such that for all $\mathbf{x} \in \mathbf{s}$, $\mathbf{x} \in B_{2\Gamma h_{max}}(\mathbf{x}_i)$. Thus, $\overline{\Omega}_{\overline{\mathbf{AF}}(\mathbf{x}_i)} \cap \overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i)$ is a non-empty compact subset of $\mathbb{R}^n$.

Recall $\overline{\Omega}_X$ (5.14) is the largest region of $\mathbb{R}^n$ contained in X. If $\overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i) \subset \overline{\Omega}_X$, define $R(\mathbf{x}_i) \subset X$ the smallest n-simplicial mesh that contains $\overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i)$. Otherwise there exists $\mathbf{x} \in \overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i)$ such that $\mathbf{x} \notin X$. In this case, define $R(\mathbf{x}_i)$ to be the smallest n-simplicial mesh that contains $\overline{\Omega}_X \cap \overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i)$.

For both instances, define

$$\widetilde{R}(\mathbf{x}_i) = \left\{ \bigcup_{\mathbf{s} \in R(\mathbf{x}_i)} \bigcup_{\zeta \in \Xi_n} \sum_{j=0}^{n} \zeta_j \mathbf{x}_j^{\mathbf{s}} \right\}. \tag{5.24}$$

Let $\partial \widetilde{R}(\mathbf{x}_i)$ denote the boundary of $\widetilde{R}(\mathbf{x}_i)$.

Let $S_{\overline{\mathbf{AF}}\widetilde{R}}(\mathbf{x}_i)$ denote the $(n-1)$-simplices of X that form the boundary of the compact region $\overline{\Omega}_{\overline{\mathbf{AF}}(\mathbf{x}_i)} \cap \widetilde{R}(\mathbf{x}_i)$.

Finally for Case 2,

$$S(\mathbf{x}_i) = S_{\overline{\mathbf{AF}}\widetilde{R}}(\mathbf{x}_i) \cup \overline{\mathbf{NF}}(\mathbf{x}_i). \tag{5.25}$$

See Figure 5.3. The union with $\overline{\mathbf{NF}}(\mathbf{x}_i)$ ensures that the update for OUM is captured in $S(\mathbf{x}_i)$. Other $(n-1)$-simplices not part of $\overline{\mathbf{AF}}$ may be part of $\overline{\mathbf{NF}}(\mathbf{x}_i)$.

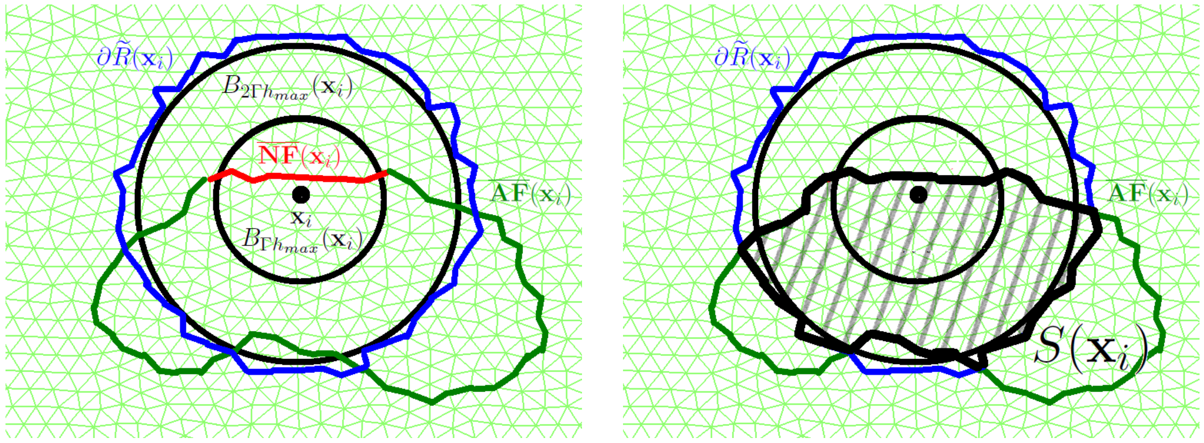
By construction, the updating stencil $S(\mathbf{x}_i)$ trivially satisfies the first three properties of Definition 5.2.7. Note if $\mathbf{x} \in \widetilde{R}(\mathbf{x}_i)$, then $\|\mathbf{x} - \mathbf{x}_i\| \leq (2\Gamma + 1)h_{max}$.

It remains to show the last property of Definition 5.2.7, that is, the optimizer over all of $S(\mathbf{x}_i)$ in (5.20) must come from $\overline{\mathbf{NF}}(\mathbf{x}_i)$. For $\mathbf{x}_i \in X \cap \Omega$, let $\widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_i}}$ be the minimum value on the *Accepted Front* $\mathbf{AF}$ just before $\mathbf{x}_i$ is labelled *Accepted*.

Lemma 5.2.10. *[72, Lemma 7.3(i) and (iii)] Assume the vertex $\boldsymbol{x}_i \in X$ is about to be labelled Accepted. Then*

1. $\widetilde{V}_{min}^{\boldsymbol{AF}_{\boldsymbol{x}_i}} + h_{min}G_{min} \leq \widetilde{V}(\boldsymbol{x}_i) \leq \widetilde{V}_{min}^{\boldsymbol{AF}_{\boldsymbol{x}_i}} + h_{max}G_{max}$.

2. *If $\boldsymbol{x}_i$ is labelled Accepted before $\boldsymbol{x}_j$ then* $\widetilde{V}_{min}^{\boldsymbol{AF}_{\boldsymbol{x}_i}} \leq \widetilde{V}_{min}^{\boldsymbol{AF}_{\boldsymbol{x}_j}}$.



$\partial\widetilde{R}(\mathbf{x}_i)$
$B_{2\Gamma h_{max}}(\mathbf{x}_i)$
$\mathbf{NF}(\mathbf{x}_i)$
$\mathbf{x}_i$
$B_{\Gamma h_{max}}(\mathbf{x}_i)$
$\mathbf{AF}(\mathbf{x}_i)$
$\partial\widetilde{R}(\mathbf{x}_i)$
$\mathbf{AF}(\mathbf{x}_i)$
$S(\mathbf{x}_i)$

Figure 5.3: Updating Stencil $S(\mathbf{x}_i)$ in $\mathbb{R}^2$ - Left: Edges of $\overline{\mathbf{NF}}(\mathbf{x}_i)$, $\overline{\mathbf{AF}}(\mathbf{x}_i)$ and $\partial\widetilde{R}(\mathbf{x}_i)$ are shown. Right: The updating stencil $S(\mathbf{x}_i)$ is the union of $\overline{\mathbf{NF}}(\mathbf{x}_i)$ with the boundary of the intersection of regions $\widetilde{R}(\mathbf{x}_i)$ with $\overline{\mathbf{AF}}(\mathbf{x}_i)$. Vertices strictly inside $S(\mathbf{x}_i)$ are not yet labelled *Accepted*. The updating stencil $S(\mathbf{x}_i)$ satisfies Definition 5.2.7.

Lemma 5.2.11. *Let* $\boldsymbol{s} = \boldsymbol{x}_0^{\boldsymbol{s}}\boldsymbol{x}_1^{\boldsymbol{s}}\cdots\boldsymbol{x}_{n-1}^{\boldsymbol{s}} \in X$ *and* $\zeta = (\zeta_0, \zeta_1, ..., \zeta_{n-1}) \in \Xi_{n-1}$ *such that* $\widetilde{\boldsymbol{x}} = \sum_{j=0}^{n-1}\zeta_j\boldsymbol{x}_j^{\boldsymbol{s}}$. *If* $\boldsymbol{x}_i \in X$ *is labelled Accepted before all of* $\boldsymbol{x}_0^{\boldsymbol{s}}$, $\boldsymbol{x}_1^{\boldsymbol{s}}$, ...,$\boldsymbol{x}_{n-1}^{\boldsymbol{s}}$ *and* $\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i\| > \Gamma h_{max}$*, then*

$$\widetilde{V}(\boldsymbol{x}_i) < \sum_{j=0}^{n-1}\zeta_j\widetilde{V}(\boldsymbol{x}_j^{\boldsymbol{s}}) + \|\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i\|\, g\left(\boldsymbol{x}_i, \frac{\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i}{\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}_i\|}\right). \tag{5.26}$$

Proof. The vertex $\mathbf{x}_i$ is labelled *Accepted* before each of the vertices of $\mathbf{s}$. From Lemma 5.2.10,

$$\begin{aligned}\widetilde{V}(\mathbf{x}_i) &\leq \widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_i}} + h_{max}G_{max},\\ &\leq \min\{\widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_0^{\mathbf{s}}}}, \widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_1^{\mathbf{s}}}}, ..., \widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_{n-1}^{\mathbf{s}}}}\} + \frac{G_{min}}{G_{min}}\cdot G_{max}h_{max},\\ &= \sum_{j=0}^{n-1}\zeta_j\min\{\widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_0^{\mathbf{s}}}}, \widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_1^{\mathbf{s}}}}, ..., \widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_{n-1}^{\mathbf{s}}}}\} + \Gamma h_{max}G_{min}.\end{aligned}$$

Since $h_{min}G_{min} > 0$, from part 1 of Lemma 5.2.10, $\widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_j^{\mathbf{s}}}} < \widetilde{V}(\mathbf{x}_j^{\mathbf{s}})$ for $j = 1, ..., n-1$. From (5.11), $G_{min} \leq g(\mathbf{x}, \mathbf{u})$ for any $\mathbf{x} \in \overline{\Omega}$ and $\mathbf{u} \in \mathbb{S}^{n-1}$. Finally, $\Gamma h_{max} < \|\widetilde{\mathbf{x}} - \mathbf{x}_i\|$ was assumed. Thus,

$$\widetilde{V}(\mathbf{x}_i) < \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\mathbf{x}_j^{\mathbf{s}}) + \|\widetilde{\mathbf{x}} - \mathbf{x}_i\| \, g\left(\mathbf{x}_i, \frac{\widetilde{\mathbf{x}} - \mathbf{x}_i}{\|\widetilde{\mathbf{x}} - \mathbf{x}_i\|}\right). \tag{5.27}$$

□ The inequality (5.27) is in the form (4.13). Hence $\widetilde{V}(\mathbf{x}_i)$ computed by the OUM algorithm (4.14) must be smaller than an update (5.18) from any $(n-1)$-simplex more than Γh_{max} away from $\mathbf{x}_i$ made of vertices labelled *Accepted* after $\mathbf{x}_i$.

The minimizing update in (5.16) from the *Near Front* $\overline{\mathbf{NF}}(\mathbf{x}_i)$ is the same as the update provided over all of $\mathbf{AF}$.

Lemma 5.2.12. *[72, Lemma 7.1] Let $\boldsymbol{x}_i$ be the vertex with Considered label that is about to be relabelled Accepted. Let*

$$\widetilde{\widetilde{W}}(\boldsymbol{x}_i) = \min_{s \in \boldsymbol{AF}} \min_{\zeta \in \Xi_{n-1}} \left\{ \tau_s(\boldsymbol{x}_i, \zeta) g(\boldsymbol{x}_i, \widetilde{\boldsymbol{u}}_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta)) + \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\boldsymbol{x}_j^{\boldsymbol{s}}) \right\}. \tag{5.28}$$

Then $\widetilde{\widetilde{W}}(\boldsymbol{x}_i) = \widetilde{V}(\boldsymbol{x}_i)$. The minimizing $\boldsymbol{s}^*$, ζ^* from (5.28) is the same as (5.16), despite $\overline{\boldsymbol{NF}}(\boldsymbol{x}_i) \subseteq \boldsymbol{AF}$.

Theorem 5.2.13. *Let $\widetilde{V} : X \to \mathbb{R}$ be computed by the OUM (5.16) on mesh X, weight function g and boundary function q. Then for $\boldsymbol{x}_i \in X \cap \Omega$,*

$$\widetilde{V}(\boldsymbol{x}_i) = \min_{\boldsymbol{s} \in S(\boldsymbol{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ g(\boldsymbol{x}_i, \widetilde{\boldsymbol{u}}_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta)) \tau_{\boldsymbol{s}}(\boldsymbol{x}_i, \zeta) + \sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\boldsymbol{x}_j^{\boldsymbol{s}}) \right\}. \tag{5.29}$$

A minimizing update from (5.29) over $S(\boldsymbol{x}_i)$ comes from $\overline{\boldsymbol{NF}}(\boldsymbol{x}_i)$.

Proof. Let the OUM algorithm be at the instant where vertex $\mathbf{x}_i$ with *Considered* label is about to be relabelled *Accepted*.

Recall Case 1, where $\overline{\mathbf{AF}}(\mathbf{x}_i)$ is entirely inside $\overline{B}_{2\Gamma h_{max}}(\mathbf{x}_i)$ and $S(\mathbf{x}_i) = \overline{\mathbf{AF}}(\mathbf{x}_i) \cup \overline{\mathbf{NF}}(\mathbf{x}_i)$. Since $\overline{\mathbf{AF}}(\mathbf{x}_i) \subseteq \mathbf{AF}$ and $\overline{\mathbf{NF}}(\mathbf{x}_i) \subseteq \mathbf{AF}$, $S(\mathbf{x}_i) \subseteq \mathbf{AF}$. By Lemma 5.2.12, $\overline{\mathbf{NF}}(\mathbf{x}_i)$ must contain a minimizing update.

Recall Case 2, where $S(\mathbf{x}_i) = S_{\overline{\mathbf{AF}}\widetilde{R}}(\mathbf{x}_i) \cup \overline{\mathbf{NF}}(\mathbf{x}_i)$. By Lemma 5.2.12, the minimizing update (5.28) is not from $(n-1)$-simplices $\mathbf{s} \in \overline{\mathbf{AF}}(\mathbf{x}_i)$ that are entirely outside $B_{\Gamma h_{max}}(\mathbf{x}_i)$. The $(n-1)$-simplices of $\overline{\mathbf{AF}}(\mathbf{x}_i)$ that contain a point inside $B_{\Gamma h_{max}}(\mathbf{x}_i)$ belong to $\overline{\mathbf{NF}}(\mathbf{x}_i)$.

The update from $(n-1)$-simplex $\mathbf{s} \in S(\mathbf{x}_i) \cap \partial\widetilde{R}(\mathbf{x}_i)$ $(n-1)$-simplices outside $B_{2\Gamma h_{max}}(\mathbf{x}_i)$ will be shown to be at least the update from OUM. Because the vertices of $\mathbf{s}$ lie on or inside

$\overline{\mathbf{AF}}(\mathbf{x}_i)$, they must either be on the *Accepted Front* or not yet *Accepted* (Lemma 5.2.8). The $(n-1)$-simplex $\mathbf{s}$ is outside $B_{2\Gamma h_{max}}(\mathbf{x}_i)$. Three cases are considered.

1. If none of the vertices of $\mathbf{s}$ have been labelled *Accepted*, Lemma 5.2.11 applies. The update for $\mathbf{x}_i$ from $\mathbf{s} \in S(\mathbf{x}_i) \cap \partial\widetilde{R}(\mathbf{x}_i)$ is greater than $\widetilde{V}(\mathbf{x}_i)$ from OUM.

2. If the vertices of $\mathbf{s}$ are all on the *Accepted Front*, then $\mathbf{s} \in \mathbf{AF}$ and Lemma 5.2.12 applies. The update from $\mathbf{s}$ is at least $\widetilde{V}(\mathbf{x}_i)$.

3. If at least one but not all the vertices of the $(n-1)$-simplex $\mathbf{s}$ are on the *Accepted Front*, then the remaining vertices must be labelled *Considered* (a neighbour is labelled *Accepted*). Let the vertices labelled *Accepted* and *Considered* $\mathbf{s}$ be denoted by $\{\mathbf{x}_1^{\mathbf{sa}}, ..., \mathbf{x}_l^{\mathbf{sa}}\}$ and $\{\mathbf{x}_1^{\mathbf{sc}}, ..., \mathbf{x}_k^{\mathbf{sc}}\}$ respectively. Let $\mathbf{s}$ be rewritten $\mathbf{s} = \mathbf{x}_1^{\mathbf{sa}} \cdots \mathbf{x}_l^{\mathbf{sa}} \mathbf{x}_1^{\mathbf{sc}} \cdots \mathbf{x}_k^{\mathbf{sc}}$ where $l+k = n$ since $\mathbf{s}$ has n vertices. Let $\zeta = (\zeta_1^{\mathbf{sa}}, ..., \zeta_l^{\mathbf{sa}}, \zeta_1^{\mathbf{sc}}, ..., \zeta_k^{\mathbf{sc}})$ be the barycentric coordinates for $\mathbf{x} \in \mathbf{s}$. By Lemma 5.2.10, for all $1 \leq j \leq k$,

$$\begin{aligned}\widetilde{V}(\mathbf{x}_i) &\leq \widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_i}} + h_{max} G_{max}, \\ &\leq \widetilde{V}_{min}^{\mathbf{AF}_{\mathbf{x}_j^{\mathbf{sc}}}} + h_{max} \cdot \frac{G_{min}}{G_{min}} \cdot G_{max}, \\ &< \widetilde{V}(\mathbf{x}_j^{\mathbf{sc}}) + \Gamma h_{max} G_{min}.\end{aligned}$$

For all $1 \leq j \leq k$, and $1 \leq m \leq l$, $\mathbf{x}_j^{\mathbf{sc}}$ is labelled *Considered* and $\mathbf{x}_m^{\mathbf{sa}}$ is on its *Near Front* $\overline{\mathbf{NF}}(\mathbf{x}_j^{\mathbf{sc}})$. Thus,

$$\begin{aligned}\widetilde{V}(\mathbf{x}_j^{\mathbf{sc}}) &\leq \widetilde{V}(\mathbf{x}_m^{\mathbf{sa}}) + \left\|\mathbf{x}_m^{\mathbf{sa}} - \mathbf{x}_j^{\mathbf{sc}}\right\| g\left(\mathbf{x}_j^{\mathbf{sc}}, \frac{\mathbf{x}_m^{\mathbf{sa}} - \mathbf{x}_j^{\mathbf{sc}}}{\left\|\mathbf{x}_m^{\mathbf{sa}} - \mathbf{x}_j^{\mathbf{sc}}\right\|}\right) \\ &\leq \widetilde{V}(\mathbf{x}_m^{\mathbf{sa}}) + \Gamma h_{max} G_{min} \\ \widetilde{V}(\mathbf{x}_m^{\mathbf{sa}}) &\geq \widetilde{V}(\mathbf{x}_j^{\mathbf{sc}}) - \Gamma h_{max} G_{min} > \widetilde{V}(\mathbf{x}_i) - 2\Gamma h_{max} G_{min}.\end{aligned}$$

Consider the update for $\mathbf{x}_i$ using (4.13) from $(n-1)$-simplex $\mathbf{s}$. For $\zeta \in \Xi_{n-1}$,

$$
\begin{aligned}
&\sum_{j=0}^{n-1} \zeta_j \widetilde{V}(\mathbf{x}_j^{\mathbf{s}}) + \tau_{\mathbf{s}}(\mathbf{x}_i, \zeta) g(\mathbf{x}_i, \mathbf{u}_{\mathbf{s}}(\mathbf{x}_i, \zeta)) \\
=& \left(\sum_{m=1}^{l} \zeta_m^{\mathbf{sa}} \widetilde{V}(\mathbf{x}_m^{\mathbf{sa}}) \right) + \left(\sum_{j=1}^{k} \zeta_j^{\mathbf{sc}} \widetilde{V}(\mathbf{x}_j^{\mathbf{sc}}) \right) + \tau_{\mathbf{s}}(\mathbf{x}_i, \zeta) g(\mathbf{x}_i, \mathbf{u}_{\mathbf{s}}(\mathbf{x}_i, \zeta)), \\
>& \sum_{m=1}^{l} \zeta_m^{\mathbf{sa}} (\widetilde{V}(\mathbf{x}_i) - 2\Gamma h_{max} G_{min}) + \sum_{j=1}^{k} \zeta_j^{\mathbf{sc}} (\widetilde{V}(\mathbf{x}_i) - \Gamma h_{max} G_{min}) + 2\Gamma h_{max} G_{min} \\
\geq& \widetilde{V}(\mathbf{x}_i),
\end{aligned}
$$

since $\sum_{m=1}^{l} \zeta_m^{\mathbf{sa}} + \sum_{j=1}^{k} \zeta_j^{\mathbf{sc}} = 1$.

Therefore $\mathbf{s} \in S(\mathbf{x}_i) \cap \partial \widetilde{R}(\mathbf{x}_i)$ provides an update larger than OUM. Recall $S(\mathbf{x}_i) \backslash \partial \widetilde{R}(\mathbf{x}_i) \subseteq$ **AF**. By Lemma 5.2.12, a minimizing direction (5.29) in $S(\mathbf{x}_i)$ must always come from the *Near Front* $\overline{\mathbf{NF}}(\mathbf{x}_i)$ of $\mathbf{x}_i$. □

By Theorems 5.2.6 and 5.2.13,

$$
\widetilde{H}[S(\mathbf{x}_i), \widetilde{V}[S(\mathbf{x}_i)]](\mathbf{x}_i, \widetilde{V}(\mathbf{x}_i)) = \widetilde{H}[\overline{\mathbf{NF}}(\mathbf{x}_i), \widetilde{V}[\overline{\mathbf{NF}}(\mathbf{x}_i)]](\mathbf{x}_i, \widetilde{V}(\mathbf{x}_i)).
$$

Therefore, for $\mathbf{x}_i \in X \cap \Omega$, an updating stencil $S(\mathbf{x}_i)$ that satisfies the properties of Definition 5.2.7 always exists.

The monotonicity and consistency of the numerical Hamiltonian using $S(\mathbf{x}_i)$ is shown.

Theorem 5.2.14. *(Monotonicity) For $\underline{\phi}, \overline{\phi} : X \to \mathbb{R}$ that satisfy $\underline{\phi}(\boldsymbol{x}_j) \leq \overline{\phi}(\boldsymbol{x}_j)$ for all $\boldsymbol{x}_j \in X \cap \Omega$, and $\phi(\boldsymbol{x}_i) = \overline{\phi}(\boldsymbol{x}_i) = \underline{\phi}(\boldsymbol{x}_i) \in \mathbb{R}$,*

$$
\widetilde{H}[S, \underline{\phi}](\boldsymbol{x}_i, \phi(\boldsymbol{x}_i)) \geq \widetilde{H}[S, \overline{\phi}](\boldsymbol{x}_i, \phi(\boldsymbol{x}_i)).
$$

Proof.

$$
\begin{aligned}
\widetilde{H}[S, \underline{\phi}](\mathbf{x}_i, \phi(\mathbf{x}_i)) &= - \min_{\mathbf{s} \in S(\mathbf{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \frac{\sum_{k=0}^{n-1} \zeta_k \underline{\phi}(\mathbf{x}_k^{\mathbf{s}}) - \phi(\mathbf{x}_i)}{\tau_{\mathbf{s}}(\mathbf{x}, \zeta)} + g(\mathbf{x}_i, \widetilde{\mathbf{u}}_{\mathbf{s}}(\mathbf{x}, \zeta)) \right\}, \\
&\geq - \min_{\mathbf{s} \in S(\mathbf{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \frac{\sum_{k=0}^{n-1} \zeta_k \overline{\phi}(\mathbf{x}_k^{\mathbf{s}}) - \phi(\mathbf{x}_i)}{\tau_{\mathbf{s}}(\mathbf{x}, \zeta)} + g(\mathbf{x}_i, \widetilde{\mathbf{u}}_{\mathbf{s}}(\mathbf{x}, \zeta)) \right\}, \\
&= \widetilde{H}[S, \overline{\phi}](\mathbf{x}_i, \phi(\mathbf{x}_i)).
\end{aligned}
$$

□

The error between the Hamiltonian H (5.6) and numerical Hamiltonian $\widetilde{H}$ (5.20) will be shown to be $\mathcal{O}(h_{max})$ as $h_{max} \to 0$. A similar proof was used in [5, Prop 2.2] for the Monotone Acceptance OUM.

Theorem 5.2.15. *(Consistency) There exists $C_1 \in \mathbb{R}_+$ (not dependent on h_{max}) for all $\boldsymbol{x}_i \in X \cap \Omega$ and $\phi \in C^2(\Omega)$, such that*

$$|\widetilde{H}[S,\phi](\boldsymbol{x}_i, \phi(\boldsymbol{x}_i)) - H(\boldsymbol{x}_i, \nabla\phi)| \leq C_1 \left\|\nabla^2\phi\right\|_2 h_{max}.$$

where $\|A\|_2$ denotes the largest magnitude of the eigenvalues of $A \in \mathbb{R}^{n\times n}$.

Proof. Let $\phi \in C^2(\Omega)$ and $\mathbf{x}_i \in X \cap \Omega$. Taylor's theorem is used. There exists $\mathbf{c} \in \mathbb{R}^n$ on the straight line between $\mathbf{x}_i$ and $\mathbf{x}$ such that

$$\nabla\phi(\mathbf{x}_i)\cdot(\mathbf{x}-\mathbf{x}_i) = \phi(\mathbf{x}) - \phi(\mathbf{x}_i) - \frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T\nabla^2\phi(\mathbf{c})(\mathbf{x}-\mathbf{x}_i). \tag{5.30}$$

Let $\zeta = (\zeta_0, \zeta_1, ..., \zeta_{n-1}) \in \Xi_{n-1}$ (4.1) and $\mathbf{x} \in \mathbf{s}$ where $\mathbf{x} = \sum_{j=0}^{n-1}\zeta_j\mathbf{x}_j^{\mathbf{s}}$ and $\mathbf{s} = \mathbf{x}_0^{\mathbf{s}}\mathbf{x}_1^{\mathbf{s}}\cdots\mathbf{x}_{n-1}^{\mathbf{s}}$. For $0 \leq j \leq n-1$, there exists $\mathbf{c}_j \in \mathbb{R}^n$ on the straight line between $\mathbf{x}_j^{\mathbf{s}}$ and $\mathbf{x}$ such that

$$\phi(\mathbf{x}_j^{\mathbf{s}}) = \phi(\mathbf{x}) + \nabla\phi(\mathbf{x})^T(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x}) + \frac{1}{2}(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x})^T\nabla^2\phi(\mathbf{c}_j)(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x}),$$

and so

$$\sum_{j=0}^{n-1}\zeta_j\phi(\mathbf{x}_j^{\mathbf{s}}) = \sum_{j=0}^{n-1}\zeta_j\phi(\mathbf{x}) + \zeta_j\nabla\phi(\mathbf{x})^T(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x}) + \frac{\zeta_j}{2}(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x})^T\nabla^2\phi(\mathbf{c}_j)(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x}).$$

From (4.1) $\sum_{j=0}^{1}\zeta_j = 1$, $\sum_{j=0}^{n-1}\zeta_j\nabla\phi(\mathbf{x})^T(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x}) = \nabla\phi(\mathbf{x})^T(\mathbf{x}-\mathbf{x}) = 0$. Thus

$$\sum_{j=0}^{n-1}\zeta_j\phi(\mathbf{x}_j^{\mathbf{s}}) = \phi(\mathbf{x}) + \frac{\zeta_j}{2}(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x})^T\nabla^2\phi(\mathbf{c}_j)(\mathbf{x}_j^{\mathbf{s}}-\mathbf{x}). \tag{5.31}$$

Since $S(\mathbf{x}_i)$ is directionally complete, the characteristic direction $\mathbf{u}^*$ (3.10) of the Hamiltonian $H(\mathbf{x}_i, \nabla\phi)$ (5.6) at $\mathbf{x}_i$ can be described by a point $\mathbf{x}^*$ on $(n-1)$-simplex $\mathbf{s}^* \in S(\mathbf{x}_i)$ with barycentric coordinates $\zeta^* = (\zeta_0^*, \zeta_1^*, ..., \zeta_{n-1}^*)$,

$$\mathbf{u}_{\mathbf{s}^*}^*(\mathbf{x}_i, \zeta^*) = \frac{\sum_{j=0}^{n-1}\zeta_j^*\mathbf{x}_j^{\mathbf{s}^*} - \mathbf{x}_i}{\tau_{\mathbf{s}^*}(\mathbf{x}_i, \zeta^*)} = \frac{\mathbf{x}^* - \mathbf{x}_i}{\tau_{\mathbf{s}^*}(\mathbf{x}_i, \zeta^*)}. \tag{5.32}$$

Similarly, let $\widetilde{\mathbf{s}}^* \in S(\mathbf{x}_i)$ and $\widetilde{\zeta}^* = (\widetilde{\zeta}_0^*, \widetilde{\zeta}_1^*, ..., \widetilde{\zeta}_{n-1}^*)$ optimize the numerical Hamiltonian $\widetilde{H}[S,\phi](\mathbf{x}_i, \phi(\mathbf{x}_i))$ (5.20). The approximated characteristic direction (5.17) is

$$\widetilde{\mathbf{u}}^*_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*) = \frac{\sum_{j=0}^{n-1} \widetilde{\zeta}_j^* \mathbf{x}_j^{\widetilde{\mathbf{s}}^*} - \mathbf{x}_i}{\tau_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)} = \frac{\widetilde{\mathbf{x}}^* - \mathbf{x}_i}{\tau_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)}, \tag{5.33}$$

where $\widetilde{\mathbf{x}}^* = \sum_{j=0}^{n-1} \widetilde{\zeta}_j^* \mathbf{x}_j^{\widetilde{\mathbf{s}}^*} \in \widetilde{\mathbf{s}}^* \in S(\mathbf{x}_i)$.

From (5.6),

$$-H(\mathbf{x}_i, \nabla\phi) = \min_{\mathbf{u} \in \mathbb{S}^{n-1}} \{\nabla\phi(\mathbf{x}_i) \cdot \mathbf{u} + g(\mathbf{x}_i, \mathbf{u})\}.$$

Using (5.30),

$$-H(\mathbf{x}_i, \nabla\phi) = \min_{\mathbf{s} \in S(\mathbf{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \nabla\phi(\mathbf{x}_i) \cdot \frac{\mathbf{x} - \mathbf{x}_i}{\tau_{\mathbf{s}}(\mathbf{x}_i, \zeta)} + g(\mathbf{x}_i, \mathbf{u}_{\mathbf{s}}(\mathbf{x}_i, \zeta)) \right\}.$$

$$= \min_{s \in S(\mathbf{x}_i)} \min_{\zeta \in \Xi_{n-1}} \left\{ \frac{\phi(\mathbf{x}) - \phi(\mathbf{x}_i) - \frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^T \nabla^2\phi(\mathbf{c})(\mathbf{x} - \mathbf{x}_i)}{\tau_{\mathbf{s}}(\mathbf{x}_i, \zeta)} + g(\mathbf{x}_i, \mathbf{u}_{\mathbf{s}}(\mathbf{x}_i, \zeta)) \right\}, \tag{5.34}$$

which is minimized with $\mathbf{s}^* \in S(\mathbf{x}_i)$ and $\zeta^* \in \Xi_{n-1}$. Because $S(\mathbf{x}_i)$ is directionally complete, replacing the minimizing $\mathbf{s}^*$ and ζ^* in (5.34) with $\widetilde{\mathbf{s}}^* \in S(\mathbf{x}_i)$, $\widetilde{\zeta}^* \in \Xi_{n-1}$ yields an equal or greater expression. Using (5.31),

$$\begin{aligned} - H(\mathbf{x}_i, \nabla\phi) \leq \frac{1}{\tau_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)} \cdot (\sum_{j=0}^{n-1} \widetilde{\zeta}_j^* \phi(\mathbf{x}_j^{\widetilde{\mathbf{s}}^*}) - \phi(\mathbf{x}_i) - ... \\ ... - \frac{\widetilde{\zeta}_j^*}{2}(\mathbf{x}_j^{\widetilde{\mathbf{s}}^*} - \widetilde{\mathbf{x}}^*)^T \nabla^2\phi(\widetilde{\mathbf{c}}_j^*)(\mathbf{x}_j^{\widetilde{\mathbf{s}}^*} - \widetilde{\mathbf{x}}^*) - \frac{1}{2}(\widetilde{\mathbf{x}}^* - \mathbf{x}_i)^T \nabla^2\phi(\widetilde{\mathbf{c}}^*)(\widetilde{\mathbf{x}}^* - \mathbf{x}_i)) + g(\mathbf{x}_i, \widetilde{\mathbf{u}}_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)), \end{aligned} \tag{5.35}$$

where $\widetilde{\mathbf{c}}^*$ is on the line segment between $\widetilde{\mathbf{x}}_i^*$ and $\mathbf{x}_i$ and $\widetilde{\mathbf{c}}_j^*$ is on the line segment between $\widetilde{\mathbf{x}}^*$ and $\mathbf{x}_j^{\widetilde{\mathbf{s}}^*}$ for $j = 0, 1, ..., n-1$. The two Hamiltonians (5.6) and (5.20) will be compared. From (5.20) and (5.35),

$$\begin{aligned} &\widetilde{H}[S,\phi](\mathbf{x}_i, \phi(\mathbf{x}_i)) - H(\mathbf{x}_i, \nabla\phi) \\ &= -\frac{\sum_{j=0}^{n-1} \widetilde{\zeta}_j^* \phi(\mathbf{x}_j^{\widetilde{\mathbf{s}}^*}) - \phi(\mathbf{x}_i)}{\tau_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)} - g(\mathbf{x}_i, \widetilde{\mathbf{u}}_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)) - H(\mathbf{x}_i, \nabla\phi), \\ &\leq -\frac{\sum_{j=0}^{n-1} \frac{\widetilde{\zeta}_j^*}{2}(\mathbf{x}_j^{\widetilde{\mathbf{s}}^*} - \widetilde{\mathbf{x}}^*)^T \nabla^2\phi(\mathbf{c}_j^*)(\mathbf{x}_j^{\widetilde{\mathbf{s}}^*} - \widetilde{\mathbf{x}}^*) + \frac{1}{2}(\widetilde{\mathbf{x}}^* - \mathbf{x}_i)^T \nabla^2\phi(\mathbf{c}^*)(\widetilde{\mathbf{x}}^* - \mathbf{x}_i)}{\tau_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)}. \end{aligned}$$

The point $\widetilde{\mathbf{x}}^* \in \widetilde{\mathbf{s}}^* \in S(\mathbf{x}_i)$ is at most $(2\Gamma+1)h_{max}$ from $\mathbf{x}_i$ and at most h_{max} away from any of the vertices of $\widetilde{\mathbf{s}}^*$. The distance $\tau_{\widetilde{\mathbf{s}}^*}(\mathbf{x}_i, \widetilde{\zeta}^*)$ is at least the minimum simplex height h_{min}. Recall M from (**M1**) satisfies $1 \le \frac{h_{max}}{h_{min}} \le M$. Thus,

$$\begin{aligned}
&\widetilde{H}[S,\phi](\mathbf{x}_i, \phi(\mathbf{x}_i)) - H(\mathbf{x}_i, \nabla\phi) \\
&\le \frac{1}{h_{min}} \left(\frac{\sum_{j=0}^{n-1} \zeta_j^*}{2} \left\|\nabla^2\phi\right\|_2 h_{max}^2 + \frac{1}{2}\left\|\nabla^2\phi\right\|_2 (2\Gamma+1)^2 \right) h_{max}^2, \\
&\le \frac{M}{2}\left\|\nabla^2\phi\right\|_2 (1+(2\Gamma+1)^2) h_{max}.
\end{aligned}$$

Recall that $-\widetilde{H}[S,\phi](\mathbf{x}_i, \phi(\mathbf{x}_i))$ is minimized (5.33) with $\widetilde{\mathbf{s}}^* \in S(\mathbf{x}_i)$, $\widetilde{\zeta}^* \in \Xi_{n-1}$. From (5.30), (5.31), and $\mathbf{s}^* \in S(\mathbf{x}_i)$, $\zeta^* \in \Xi_{n-1}$ not necessarily minimizers of $-\widetilde{H}[S,\phi](\mathbf{x}_i, \phi(\mathbf{x}_i))$,

$$\begin{aligned}
&= H(\mathbf{x}_i, \nabla\phi) - \widetilde{H}[S,\phi](\mathbf{x}_i, \phi(\mathbf{x}_i)), \\
&\le \frac{1}{\tau_{\mathbf{s}^*}(\mathbf{x}_i, \zeta^*)} \left((\sum_{j=0}^{n-1} \frac{\zeta_j^*}{2} (\mathbf{x}_j^{\mathbf{s}^*} - \mathbf{x}^*)^T \nabla^2\phi(\mathbf{c}_j^*)(\mathbf{x}_j^{\mathbf{s}^*} - \mathbf{x}^*)) + \frac{1}{2}(\mathbf{x}^* - \mathbf{x}_i)^T \nabla^2\phi(\mathbf{c})(\mathbf{x}^* - \mathbf{x}_i) \right), \\
&\le \frac{M}{2}\left\|\nabla^2\phi\right\|_2 (1+(2\Gamma+1)^2) h_{max},
\end{aligned}$$

where $\mathbf{c}^*$ is on the line segment from $\mathbf{x}^*$ and $\mathbf{x}_i$ and for $j = 0, 1, ..., n-1$, $\mathbf{c}_j^*$ is on the line segment from $\mathbf{x}_j^{\mathbf{s}^*}$ and $\mathbf{x}^*$. The theorem is proved with $C_1 = \frac{M}{2}(1+(2\Gamma+1)^2)$. $\square$

5.3 OUM Error Bound

Boundedness and Lipschitz continuity of both V and $\widetilde{V}$ over $\overline{\Omega}_X \subset \mathbb{R}^n$ (5.14) are required to show the main result of a bound on convergence of the OUM. The standing assumptions **(P1) - (P5)** and **(M1) - (M3)** will be used. The following lemma is an exercise in [13] and a similar proof to that given below can be found in [7].

Lemma 5.3.1. *The set $\overline{\Omega} \subset \mathbb{R}^n$ is convex. Let $\boldsymbol{x} \in \mathbb{R}^n$, then $\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}\in\overline{\Omega}} \|\boldsymbol{x} - \boldsymbol{z}\|$ is unique, and satisfies*

$$\langle \boldsymbol{x} - \boldsymbol{z}^*, \boldsymbol{w} - \boldsymbol{z}^* \rangle \le 0, \text{ for all } \boldsymbol{w} \in \overline{\Omega}. \quad (5.36)$$

Proof

Uniqueness: If $\mathbf{x} \in \overline{\Omega}$, then $\mathbf{z}^* = \mathbf{x}$ is trivially the unique minimizer.

Otherwise, $\mathbf{x} \in \overline{\Omega}^c$. The contrapositive of the statement will be shown. Assume $\mathbf{z}_1, \mathbf{z}_2 \in \overline{\Omega}$ and $\mathbf{z}_1 \neq \mathbf{z}_2$ both minimize $\min_{\mathbf{z}\in\overline{\Omega}} \|\mathbf{x}-\mathbf{z}\|$. Let $\hat{\mathbf{z}} = \frac{1}{2}\mathbf{z}_1 + \frac{1}{2}\mathbf{z}_2$. Consider

$$\begin{aligned}
0 &< \|(\mathbf{x}-\mathbf{z}_1) - (\mathbf{x}-\mathbf{z}_2)\|^2, \\
&= 2\|\mathbf{x}-\mathbf{z}_1\|^2 + 2\|\mathbf{x}-\mathbf{z}_2\|^2 - \|(\mathbf{x}-\mathbf{z}_1) + (\mathbf{x}-\mathbf{z}_2)\|^2, \\
&= 2\|\mathbf{x}-\mathbf{z}_1\|^2 + 2\|\mathbf{x}-\mathbf{z}_2\|^2 - 4\|\mathbf{x}-\hat{\mathbf{z}}\|^2,
\end{aligned}$$

Since $\|\mathbf{x}-\mathbf{z}_1\| = \|\mathbf{x}-\mathbf{z}_2\|$,

$$\|\mathbf{x}-\hat{\mathbf{z}}\| < \|\mathbf{x}-\mathbf{z}_1\| \text{ and } \|\mathbf{x}-\hat{\mathbf{z}}\| < \|\mathbf{x}-\mathbf{z}_2\|.$$

Since $\hat{\mathbf{z}} = \frac{1}{2}\mathbf{z}_1 + \frac{1}{2}\mathbf{z}_2 \in \overline{\Omega}^c$, $\overline{\Omega}$ is not convex.

Proof of (5.36): If $\mathbf{x} \in \overline{\Omega}$, then $\mathbf{z}^* = \mathbf{x}$ and $\langle \mathbf{x}-\mathbf{z}^*, \mathbf{w}-\mathbf{z}^*\rangle = 0$. Otherwise $\mathbf{x} \in \overline{\Omega}^c$. The contrapositive will again be shown. Assume there exists $\mathbf{w} \in \Omega$ such that

$$\langle \mathbf{x}-\mathbf{z}^*, \mathbf{w}-\mathbf{z}^*\rangle > 0.$$

Let $\mathbf{w}_\alpha = (1-\alpha)\mathbf{z}^* + \alpha\mathbf{w}$ with $\alpha \in [0,1]$,

$$\begin{aligned}
\|\mathbf{x}-\mathbf{w}_\alpha\|^2 &= \|(1-\alpha)(\mathbf{x}-\mathbf{z}^*) + \alpha(\mathbf{x}-\mathbf{w})\|^2, \\
&= (1-\alpha)^2\|\mathbf{x}-\mathbf{z}^*\|^2 + \alpha^2\|\mathbf{x}-\mathbf{w}\|^2 + 2(1-\alpha)(\alpha)\langle \mathbf{x}-\mathbf{z}^*, \mathbf{x}-\mathbf{w}\rangle,
\end{aligned}$$

Differentiating, and evaluating the expression at $\alpha = 0$,

$$\begin{aligned}
\frac{d}{d\alpha}\|\mathbf{x}-\mathbf{w}_\alpha\|^2\Big|_{\alpha=0} &= -2\|\mathbf{x}-\mathbf{z}^*\|^2 + 2\langle \mathbf{x}-\mathbf{z}^*, \mathbf{x}-\mathbf{w}\rangle, \\
&= -2\langle \mathbf{x}-\mathbf{z}^*, \mathbf{w}-\mathbf{z}^*\rangle \\
&< 0,
\end{aligned}$$

since $\langle \mathbf{x}-\mathbf{z}^*, \mathbf{w}-\mathbf{z}^*\rangle > 0$.

For $\alpha > 0$ small enough,

$$\|\mathbf{x}-\mathbf{w}_\alpha\| < \|\mathbf{x}-\mathbf{w}_\alpha\|\Big|_{\alpha=0}.$$

Since $|\mathbf{x}-\mathbf{w}_\alpha| = |\mathbf{x}-\mathbf{z}^*|$ for $\alpha = 0$.

$$\|\mathbf{x}-\mathbf{w}_\alpha\| < \|\mathbf{x}-\mathbf{z}^*\|.$$

Hence $\mathbf{w}_\alpha \in \overline{\Omega}^c$. But since $\mathbf{w}, \mathbf{z}^* \in \overline{\Omega}$, and $\mathbf{w}_\alpha = (1-\alpha)\mathbf{z}^* + \alpha\mathbf{w} \in \overline{\Omega}^c$, $\overline{\Omega}$ is not convex. Therefore for convex $\overline{\Omega}$, $\langle \mathbf{x}-\mathbf{z}^*, \mathbf{w}-\mathbf{z}^*\rangle \leq 0$ for all $\mathbf{w} \in \overline{\Omega}$. □

Lemma 5.3.2. *The value function V in (5.5) is Lipschitz over $\overline{\Omega}_X$. That is, there exists $L_V > 0$ such that for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \overline{\Omega}_X$,*

$$|V(\boldsymbol{x}_1) - V(\boldsymbol{x}_2)| \leq L_V \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| .$$

Proof: Three cases are considered.

Case 1: $\mathbf{x}_1, \mathbf{x}_2 \in \overline{\Omega}_X \cap \Omega^c$. This is an exercise in [13, Exercise 2.8d]. The proof is as follows. Let

$$\widetilde{\mathbf{x}}_1 = \underset{\overline{\mathbf{x}}_1 \in \partial\Omega}{\arg\min} \|\mathbf{x}_1 - \overline{\mathbf{x}}_1\| \text{ and } \widetilde{\mathbf{x}}_2 = \underset{\overline{\mathbf{x}}_2 \in \partial\Omega}{\arg\min} \|\mathbf{x}_2 - \overline{\mathbf{x}}_2\| .$$

From Lemma 5.3.1,

$$\langle \mathbf{x}_1 - \widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2 - \widetilde{\mathbf{x}}_1 \rangle \leq 0, \text{ and } \langle \mathbf{x}_2 - \widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2 \rangle \leq 0,$$

$$\begin{gathered} \langle \mathbf{x}_2 - \widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2 \rangle - \langle \mathbf{x}_1 - \widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2 \rangle \leq 0, \\ \langle \widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2 - (\mathbf{x}_1 - \mathbf{x}_2), \widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2 \rangle \leq 0. \end{gathered}$$

By the Cauchy-Schwartz inequality,

$$\begin{aligned} \|\widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2\|^2 &\leq \langle \mathbf{x}_1 - \mathbf{x}_2, \widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2 \rangle \\ &\leq \|\mathbf{x}_1 - \mathbf{x}_2\| \|\widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2\| , \\ \|\widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2\| &\leq \|\mathbf{x}_1 - \mathbf{x}_2\| . \end{aligned}$$

By Lemma 5.1.3, q is Lipschitz-continuous with constant $2G_{max}$, therefore

$$|V(\mathbf{x}_1) - V(\mathbf{x}_2)| \leq |q(\widetilde{\mathbf{x}}_1) - q(\widetilde{\mathbf{x}}_2)| \leq 2G_{max} \|\widetilde{\mathbf{x}}_1 - \widetilde{\mathbf{x}}_2\| \leq 2G_{max} \|\mathbf{x}_1 - \mathbf{x}_2\| .$$

Case 2: $\mathbf{x}_1, \mathbf{x}_2 \in \Omega$. This is shown in [83, Lemma 2.2.7]. Suppose $\mathbf{x}_1, \mathbf{x}_2 \in \Omega$. Using (5.5), let $\overline{\mathbf{u}}(t) = \frac{\mathbf{x}_2 - \mathbf{x}_1}{\|\mathbf{x}_2 - \mathbf{x}_1\|}$ and $\tau = \|\mathbf{x}_2 - \mathbf{x}_1\|$,

$$\begin{aligned} V(\mathbf{x}_1) &\leq \int_0^\tau g(\mathbf{y}_{\mathbf{x}_1}(s), \overline{\mathbf{u}}(s)) ds + V(\mathbf{x}_2), \\ V(\mathbf{x}_1) - V(\mathbf{x}_2) &\leq G_{max} \|\mathbf{x}_1 - \mathbf{x}_2\| , \\ V(\mathbf{x}_2) &\leq \int_0^\tau g(\mathbf{y}_{\mathbf{x}_2}(s), -\overline{\mathbf{u}}(s)) ds + V(\mathbf{x}_1), \\ V(\mathbf{x}_2) - V(\mathbf{x}_1) &\leq G_{max} \|\mathbf{x}_1 - \mathbf{x}_2\| . \end{aligned}$$

Case 3: $\mathbf{x}_1 \in \Omega$ and $\mathbf{x}_2 \in \Omega^c$: Let $\widetilde{\mathbf{x}}_2 = \arg\min_{\widetilde{\mathbf{x}} \in \partial\Omega} \|\mathbf{x}_2 - \widetilde{\mathbf{x}}\|$, and $\overline{\mathbf{u}} = \frac{\widetilde{\mathbf{x}}_2 - \mathbf{x}_1}{\|\widetilde{\mathbf{x}}_2 - \mathbf{x}_1\|}$. By **(P4)**, $V(\mathbf{x}_2) = q(\widetilde{\mathbf{x}}_2)$ and the definition of V,

$$\begin{aligned} V(\mathbf{x}_1) - V(\mathbf{x}_2) = V(\mathbf{x}_1) - q(\widetilde{\mathbf{x}}_2) &\leq \int_0^{\tau} g(\mathbf{y}_{\mathbf{x}_1}(s), \overline{\mathbf{u}}(s))ds + q(\widetilde{\mathbf{x}}_2) - q(\widetilde{\mathbf{x}}_2) \\ &\leq G_{max} \|\mathbf{x}_1 - \widetilde{\mathbf{x}}_2\| \leq G_{max} \|\mathbf{x}_1 - \mathbf{x}_2\|, \end{aligned}$$

where the last line comes from the use of Lemma 5.3.1 in

$$\begin{aligned} \|\mathbf{x}_1 - \mathbf{x}_2\|^2 &= \|\mathbf{x}_1 - \widetilde{\mathbf{x}}_2\|^2 + \|\mathbf{x}_2 - \widetilde{\mathbf{x}}_2\|^2 - 2\langle \mathbf{x}_1 - \widetilde{\mathbf{x}}_2, \mathbf{x}_2 - \widetilde{\mathbf{x}}_2 \rangle, \\ &\geq \|\mathbf{x}_1 - \widetilde{\mathbf{x}}_2\|^2. \end{aligned}$$

Observe that for $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \overline{\Omega}$ (5.8),

$$L(\mathbf{a}, \mathbf{b}) \leq L(\mathbf{a}, \mathbf{c}) + L(\mathbf{c}, \mathbf{b}). \tag{5.37}$$

Let $\mathbf{y}_{\mathbf{x}_1}(\tau_1) \in \partial\Omega$ be the point the optimal trajectory starting at $\mathbf{x}_1$ reached $\partial\Omega$. Using the compatibility of q from assumption (**P1**) and (5.9),

$$\begin{aligned} V(\mathbf{x}_2) - V(\mathbf{x}_1) &= q(\widetilde{\mathbf{x}}_2) - L(\mathbf{x}_1, \mathbf{y}_{\mathbf{x}_1}(\tau_1)) - q(\mathbf{y}_{\mathbf{x}_1}(\tau_1)) \\ &\leq L(\widetilde{\mathbf{x}}_2, \mathbf{y}_{\mathbf{x}_1}(\tau_1)) - L(\mathbf{x}_1, \mathbf{y}_{\mathbf{x}_1}(\tau_1)) \\ &\leq L(\widetilde{\mathbf{x}}_2, \mathbf{x}_1) \leq G_{max} \|\mathbf{x}_1 - \widetilde{\mathbf{x}}_2\| \leq G_{max} \|\mathbf{x}_1 - \mathbf{x}_2\|. \end{aligned}$$

Combining cases 1-3, for $\mathbf{x}_1, \mathbf{x}_2 \in \overline{\Omega}_X$,

$$|V(\mathbf{x}_1) - V(\mathbf{x}_2)| \leq 2G_{max} \|\mathbf{x}_1 - \mathbf{x}_2\|.$$

The lemma is satisfied for $L_V = 2G_{max}$. □

Lemma 5.3.3. *[83, Lemma 2.2.9] Let $\boldsymbol{x} \in \overline{\Omega}_X$. Let $\widetilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{z} \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{z}\|$. The value function V is bounded by*

$$q_{min} \leq V(\boldsymbol{x}) \leq G_{max} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\| + q_{max}.$$

Proof. For $\mathbf{x} \in \Omega$, let $\widetilde{\mathbf{u}} = \frac{\mathbf{x} - \widetilde{\mathbf{x}}}{\|\mathbf{x} - \widetilde{\mathbf{x}}\|}$. By the optimality principle (5.5),

$$V(\mathbf{x}) \leq \int_0^{|\mathbf{x} - \widetilde{\mathbf{x}}|} g(\mathbf{y}_{\mathbf{x}}(s), \widetilde{\mathbf{u}}(s))ds + q(\widetilde{\mathbf{x}}) \leq G_{max} \|\mathbf{x} - \widetilde{\mathbf{x}}\| + q_{max},$$

and

$$V(\mathbf{x}) = \int_0^T g(\mathbf{y}^*_{\mathbf{x}}(s), \mathbf{u}^*(s))ds + q(\mathbf{y}^*(T)) \geq G_{min} \|\mathbf{x} - \widetilde{\mathbf{x}}\| + q_{min} \geq q_{min}.$$

where $\mathbf{y}^*_{\mathbf{x}}$ and $\mathbf{u}^*$ are the optimal trajectory (with initial condition $\mathbf{x}$) and control. For $\mathbf{x} \in \overline{\Omega}_X \cap \Omega^c$, $V(\mathbf{x}) = q(\hat{\mathbf{x}})$ for some $\hat{\mathbf{x}} \in \partial\Omega$. Hence, $q_{min} \leq V(\mathbf{x}) \leq q_{max} \leq G_{max} \|\mathbf{x} - \widetilde{\mathbf{x}}\| + q_{max}$.
□

The following two lemmas show the Lipschitz-continuity and boundedness for the approximated value function $\widetilde{V}$.

Lemma 5.3.4. *[72, Lemma 7.5] For $\widetilde{V} : X \to \mathbb{R}$ obtained by the Ordered Upwind Method, and $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$, there exists $L_{\widetilde{V}} > 0$ such that*

$$|\widetilde{V}(\boldsymbol{x}_i) - \widetilde{V}(\boldsymbol{x}_j)| \leq L_{\widetilde{V}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| .$$

A possible Lipschitz constant for $\widetilde{V}$ is $L_{\widetilde{V}} = M^2 G_{max}$ [72], where M is described in (**M1**). Similar proof from case 1 and case 3 of Lemma 5.3.2 is valid with a restriction of $\mathbf{x} \in X$ and function L (5.8) is replaced with $\widetilde{L} : X \times X \to \mathbb{R}$,

$$\widetilde{L}(\mathbf{x}_1, \mathbf{x}_2) = \inf_{\mathbf{u}(\cdot)\in\widetilde{\mathcal{U}}} \left\{ \int_0^\tau g(\mathbf{y}_{\mathbf{x}_1}(s), \mathbf{u}(s))ds \;\middle|\; \mathbf{y}_{\mathbf{x}_1}(\tau) = \mathbf{x}_2, \mathbf{y}_{\mathbf{x}_1}(t) \in \overline{\Omega}, t \in (0, \tau) \right\}. \tag{5.38}$$

where

$$\widetilde{\mathcal{U}} = \left\{ \widetilde{\mathbf{u}}(\cdot) \in \mathcal{U} \middle| \widetilde{\mathbf{u}}(t) = \widetilde{\mathbf{u}}_i \text{ while } \mathbf{y}(t) \in \mathbf{s}_i \in X \right\}.$$

Lemma 5.3.5. *[72, Lemma 7.2] Let $\boldsymbol{x} \in \boldsymbol{s} \in X$ and $\widetilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{z}\in\partial\Omega} \|\boldsymbol{x} - \boldsymbol{z}\|$. Then*

$$q_{min} \leq \widetilde{V}(\boldsymbol{x}) \leq G_{max} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\| + q_{max}.$$

The proof is shown in [72] for $\mathbf{x} \in \overline{\Omega}$ and is trivial for $\mathbf{x} \in \overline{\Omega}^c$.

The following lemma provides a limit on the magnitudes of the subgradients (Definition 3.3.1) and supergradients (Definition 3.3.2) for a Lipschitz-continuous function. Let $A \subset \mathbb{R}^n$ be closed and bounded.

Lemma 5.3.6. *Let $f : A \to \mathbb{R}$ be Lipschitz-continuous with Lipschitz constant $C \in \mathbb{R}_+$. If $\boldsymbol{p} \in D^- f(\boldsymbol{x}_0) \cup D^+ f(\boldsymbol{x}_0)$, then*

$$\|\boldsymbol{p}\| \leq C.$$

Proof. Let $\delta > 0$. Then for any $\mathbf{b} \in \mathbb{S}^{n-1}$, $0 < \tilde{\delta} < \delta$, consider the Lipschitz continuity of f (Definition 3.2.1) and subgradient $\mathbf{p} \in D^- f(x_0)$ (Definition 3.3.1) with $\mathbf{x} = \mathbf{x}_0 + \tilde{\delta}\mathbf{b}$,

$$C \left\| \mathbf{x}_0 + \tilde{\delta}\mathbf{b} - \mathbf{x}_0 \right\| \geq |f(\mathbf{x}_0 + \tilde{\delta}\mathbf{b}) - f(\mathbf{x}_0)| \geq \mathbf{p} \cdot (\mathbf{x}_0 + \tilde{\delta}\mathbf{b} - \mathbf{x}_0),$$

$$C \left\| \tilde{\delta}\mathbf{b} \right\| \geq \mathbf{p} \cdot \tilde{\delta}\mathbf{b},$$

and hence

$$C \geq \mathbf{p} \cdot \mathbf{b}. \tag{5.39}$$

Since (5.39) holds for all $\mathbf{b} \in \mathbb{S}^{n-1}$, the inequality is true in particular for $\mathbf{b} = \frac{\mathbf{p}}{\|\mathbf{p}\|}$. Hence

$$\|\mathbf{p}\| \leq C.$$

The proof for the set of all supergradients, $D^+ f(\mathbf{x}_0)$, is similar. For $\mathbf{p} \in D^+ f(\mathbf{x}_0)$ (Definition 3.3.2), and Lipschitz continuity of f (Definition 3.2.1), with $\mathbf{x} = \mathbf{x}_0 + \tilde{\delta}\mathbf{b}$,

$$-C \left\| \tilde{\delta}\mathbf{b} \right\| \leq |f(\mathbf{x}_0 + \tilde{\delta}\mathbf{b}) - f(\mathbf{x}_0)| \leq \mathbf{p} \cdot (\tilde{\delta}\mathbf{b}),$$

$$-C \leq \mathbf{p} \cdot \mathbf{b}.$$

Hence,

$$\mathbf{p} \cdot (-\mathbf{b}) \leq C,$$

which again implies $\|\mathbf{p}\| \leq C$. □

The following lemma states that any point on the boundary $\partial\Omega$ must be at most h_{max} away from its nearest vertex of X.

Lemma 5.3.7. *If $\boldsymbol{x} \in \partial\Omega$, there exists $\boldsymbol{x}_i \in X \cap \Omega^c$ such that*

$$\|\boldsymbol{x} - \boldsymbol{x}_i\| \leq h_{max}. \tag{5.40}$$

Proof. From assumption (**M2**), $\overline{\Omega}$ is contained in X and so the point $\mathbf{x}$ belongs to a simplex of X. Therefore, the distance between $\mathbf{x}$ and its nearest vertex $\mathbf{x}_i$ is less than the length of the longest edge with length h_{max}. Therefore $\mathbf{x}_i$ must be at most h_{max} away. □.

A similar notion of viscosity solution for the approximated value function $\widetilde{V}$ is presented.

It is now shown that the approximated value function $\widetilde{V}$ is in a sense a viscosity solution for the numerical HJB equation (5.21).

Definition 5.3.8. *Let* $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\partial\Omega} \|\boldsymbol{x}_i - \boldsymbol{x}\|$. *A* ***subsolution of the numerical HJB equation*** *(5.21)* $\underline{\widetilde{V}} : \{\boldsymbol{x}_i\} \subset X \to \mathbb{R}$ *satisfies*

$$\begin{cases} \underline{\widetilde{V}}(\boldsymbol{x}_i) \leq q(\hat{\boldsymbol{x}}) & \text{for } \boldsymbol{x}_i \in X \cap \Omega^c, \\ \widetilde{H}[\overline{\boldsymbol{NF}}, \underline{\widetilde{V}}](\boldsymbol{x}_i, \underline{\widetilde{V}}(\boldsymbol{x}_i)) \leq 0 & \text{for } \boldsymbol{x}_i \in X \cap \Omega. \end{cases}$$

Definition 5.3.9. *Let* $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\partial\Omega} \|\boldsymbol{x}_i - \boldsymbol{x}\|$. *A* ***supersolution of the numerical HJB equation*** *(5.21)* $\overline{\widetilde{V}} : \{\boldsymbol{x}_i\} \subset X \to \mathbb{R}$ *satisfies*

$$\begin{cases} \overline{\widetilde{V}}(\boldsymbol{x}_i) \geq q(\hat{\boldsymbol{x}}) & \text{for } \boldsymbol{x}_i \in X \cap \Omega^c, \\ \widetilde{H}[\overline{\boldsymbol{NF}}, \overline{\widetilde{V}}](\boldsymbol{x}_i, \overline{\widetilde{V}}(\boldsymbol{x}_i)) \geq 0 & \text{for } \boldsymbol{x}_i \in X \cap \Omega. \end{cases}$$

Definition 5.3.10. *A* ***solution of the numerical HJB equation*** *(5.21)* $\widetilde{V}$ *is both a subsolution and a supersolution of the numerical HJB equation (5.21).*

For solutions of the numerical HJB equation,

$$\widetilde{V}(\mathbf{x}_i) = q(\hat{\mathbf{x}}) \text{ for } \hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\partial\Omega} \|\mathbf{x} - \mathbf{x}_i\| \text{ for } \mathbf{x}_i \in X \cap \Omega^c$$

and

$$\widetilde{H}[\overline{\mathbf{NF}}, \widetilde{V}](\mathbf{x}_i, \widetilde{V}(\mathbf{x}_i)) = 0 \text{ for } \mathbf{x}_i \in X \cap \Omega.$$

By Theorem 5.2.6, the approximate value function $\widetilde{V}$ produced by the OUM algorithm is a solution of the numerical HJB equation. Hence, it is both a subsolution and supersolution of the numerical HJB equation.

Theorem 5.3.11. *Let* $V : \overline{\Omega}_X \to \mathbb{R}$ *be a viscosity solution of (5.7) and* $\widetilde{V} : \{\boldsymbol{x}_i\} \subset X \to \mathbb{R}$ *be a solution of the numerical HJB equation (5.21). There exist* $C, h_0 > 0$*, (both independent of* h_{max}*) such that*

$$\max_{\boldsymbol{x}_i\in X} |V(\boldsymbol{x}_i) - \widetilde{V}(\boldsymbol{x}_i)| \leq C\sqrt{h_{max}}, \tag{5.41}$$

for every $\boldsymbol{x}_i \in X$ *and* $h_{max} \leq h_0$.

Proof. If $\mathbf{x}_i \in X \cap \overline{\Omega}^c$, then

$$|V(\mathbf{x}_i) - \widetilde{V}(\mathbf{x}_i)| = |q(\hat{\mathbf{x}}) - q(\hat{\mathbf{x}})| = 0,$$

where $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\partial\Omega} \|\mathbf{x}_i - \mathbf{x}\|$. The bound (5.41) is satisfied. Otherwise, $\mathbf{x}_i \in X \cap \overline{\Omega}$. Since $\overline{\Omega} \subset \mathbb{R}^n$ is bounded, define

$$d_\Omega := \max_{\mathbf{x},\widetilde{\mathbf{x}}\in\partial\Omega} \|\mathbf{x} - \widetilde{\mathbf{x}}\|, \tag{5.42}$$

$$C_0 := \max\{L_V, L_{\widetilde{V}}, |q_{min}|, d_\Omega G_{max} + |q_{max}|\}, \tag{5.43}$$

where $L_V, L_{\widetilde{V}}, |q_{min}|, G_{max}$, and $|q_{max}|$ are from Lemmas 5.3.2, 5.3.3, 5.3.4, 5.3.5.

Two parameters λ and ϵ are used to determine the error bound. For $0 < \lambda < 1$ and $\epsilon > 0$, define $\Phi : \overline{\Omega} \times X \to \mathbb{R}$

$$\Phi(\mathbf{x}, \mathbf{x}_i) = \lambda V(\mathbf{x}) - \widetilde{V}(\mathbf{x}_i) - \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\epsilon}. \tag{5.44}$$

Let $\overline{\mathbf{x}} \in \overline{\Omega}$ and $\overline{\mathbf{x}}_i \in X$ maximize Φ, over the compact set $\overline{\Omega} \times X$. Let

$$M_{\epsilon,\lambda} = \max_{\mathbf{x}\in\overline{\Omega},\mathbf{x}_i\in X} \Phi(\mathbf{x}, \mathbf{x}_i) = \Phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_i). \tag{5.45}$$

Choose λ such that

$$\lambda = 1 - \frac{2}{G_{min}} \left(\frac{C_1}{\epsilon} h_{max} + C_0 L_g \epsilon \right),$$

which implies

$$(1 - \lambda) = \frac{2}{G_{min}} \left(\frac{C_1}{\epsilon} h_{max} + C_0 L_g \epsilon \right), \tag{5.46}$$

where L_g is defined in (5.12), and $C_1 = \frac{M(1+(2\Gamma+1)^2)}{2}$ is defined in Theorem 5.2.15 with M in (**M1**) and $\Gamma = \frac{G_{max}}{G_{min}}$. For $\mathbf{x}_i \in X \cap \overline{\Omega}$, $V(\mathbf{x}_i) \le C_0$ from Lemma 5.3.3,

$$V(\mathbf{x}_i) - \widetilde{V}(\mathbf{x}_i) \le (1 - \lambda)V(\mathbf{x}_i) + M_{\epsilon,\lambda} \le C_0(1 - \lambda) + M_{\epsilon,\lambda}. \tag{5.47}$$

The parameter $\lambda \in (0, 1)$. Note 0, 1 are excluded. It is sufficient to pick h_{max} in (5.46) small enough such that $0 < (1 - \lambda) < 1$ is satisfied. The parameter h_0 is chosen to satisfy this. The result of the theorem will be true with $\epsilon = \sqrt{h_{max}}$. Setting (5.46) less than 1, with $\epsilon = \sqrt{h_{max}}$ yields

$$h_{max} < \frac{G_{min}^2}{4(C_1 + C_0 L_g)^2}.$$

Let $h_0 = \min\{\frac{G_{min}^2}{4(C_1+C_0L_g)^2}, 1\}$. It will be shown that for $h_{max} < h_0$, the conclusion of the theorem is true.

The point $\overline{\mathbf{x}}$ must belong to Ω or $\partial\Omega$, while $\overline{\mathbf{x}}_i$ must belong to $X \cap \Omega$ or $X \cap \Omega^c$. An outline of the remainder of proof is as follows.

1. Show that at most only one of $\overline{\mathbf{x}}$ and $\overline{\mathbf{x}}_i$ may be in Ω. The case $\overline{\mathbf{x}} \in \partial\Omega$ and $\overline{\mathbf{x}}_i \in X \cap \Omega^c$ is possible.
2. Find an upper bound for $M_{\epsilon,\lambda}$ in (5.47) given the restriction in the first step.
3. Find an upper bound on the error in (5.47) by choosing the parameter ϵ in (5.44) in terms of h_{max}.
4. Outline the proof of (5.47) for $\widetilde{V}(\mathbf{x}_i) - V(\mathbf{x}_i)$.

Step 1: Define $\phi : \overline{\Omega} \to \mathbb{R}$,

$$\phi(\mathbf{x}) = \frac{1}{\lambda}\left(M_{\epsilon,\lambda} + \widetilde{V}(\overline{\mathbf{x}}_i) + \frac{\|\mathbf{x} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}\right) \text{ and so } \nabla\phi(\mathbf{x}) = \frac{1}{\lambda}\left(\frac{\mathbf{x} - \overline{\mathbf{x}}_i}{\epsilon}\right). \tag{5.48}$$

From (5.44) and (5.45), it can be shown that $V(\mathbf{x}) \leq \phi(\mathbf{x})$ for all $\mathbf{x} \in \overline{\Omega}$ and $V(\overline{\mathbf{x}}) = \phi(\overline{\mathbf{x}})$. Therefore, $V - \phi$ has a local maximum at $\overline{\mathbf{x}}$. Therefore by Lemma 3.3.4, $\mathbf{p} = \nabla\phi(\overline{\mathbf{x}}) \in D^+V(\overline{\mathbf{x}})$. By Lemma 5.3.6, $\|\nabla\phi(\overline{\mathbf{x}})\|$ is bounded by the Lipschitz constant $L_V \leq C_0$. By (5.43), and (5.48),

$$\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\| \leq \lambda \|\nabla\phi(\overline{\mathbf{x}})\| \epsilon \leq C_0\epsilon.$$

From (5.46), and using $0 < \lambda < 1$,

$$(1 - \lambda) > \frac{1}{G_{min}}\left(\frac{C_1}{\epsilon}h_{max} + \lambda L_g \|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|\right). \tag{5.49}$$

Define $\psi : \overline{\Omega}_X \to \mathbb{R}$,

$$\psi(\mathbf{x}_i) = -M_{\epsilon,\lambda} + \lambda V(\overline{\mathbf{x}}) - \frac{\|\overline{\mathbf{x}} - \mathbf{x}_i\|^2}{2\epsilon}, \text{ and so } \nabla\psi(\mathbf{x}_i) = \frac{\overline{\mathbf{x}} - \mathbf{x}_i}{\epsilon}.$$

Let $\mathbf{u}^*_{\overline{\mathbf{x}}_i}$ optimize (3.7) for arguments $\overline{\mathbf{x}}_i$ and $\nabla\psi(\overline{\mathbf{x}}_i)$. From (5.49) and $G_{min} \leq g(\mathbf{x}, \mathbf{u})$ for all $\mathbf{x} \in \overline{\Omega}$, $\mathbf{u} \in \mathbb{S}^{n-1}$, the Lipschitz-continuity of g and definitions of $\nabla\phi$ and $\nabla\psi$,

$$\begin{aligned}
(1 - \lambda)g(\overline{\mathbf{x}}_i, \mathbf{u}^*_{\overline{\mathbf{x}}_i}) &> \frac{C_1}{\epsilon}h_{max} + \lambda(g(\overline{\mathbf{x}}, \mathbf{u}^*_{\overline{\mathbf{x}}_i}) - g(\overline{\mathbf{x}}_i, \mathbf{u}^*_{\overline{\mathbf{x}}_i})), \\
g(\overline{\mathbf{x}}_i, \mathbf{u}^*_{\overline{\mathbf{x}}_i}) - \lambda g(\overline{\mathbf{x}}, \mathbf{u}^*_{\overline{\mathbf{x}}_i}) &> \frac{C_1}{\epsilon}h_{max}, \\
\frac{\overline{\mathbf{x}} - \overline{\mathbf{x}}_i}{\epsilon} \cdot \mathbf{u}^*_{\overline{\mathbf{x}}_i} + g(\overline{\mathbf{x}}_i, \mathbf{u}^*_{\overline{\mathbf{x}}_i}) - \lambda\left(\frac{1}{\lambda} \cdot \frac{\overline{\mathbf{x}} - \overline{\mathbf{x}}_i}{\epsilon} \cdot \mathbf{u}^*_{\overline{\mathbf{x}}_i} + g(\overline{\mathbf{x}}, \mathbf{u}^*_{\overline{\mathbf{x}}_i})\right) &> \frac{C_1}{\epsilon}h_{max},
\end{aligned}$$

$$\nabla\psi(\overline{\mathbf{x}}_i)\cdot\mathbf{u}^*_{\overline{\mathbf{x}}_i} + g(\overline{\mathbf{x}}_i,\mathbf{u}^*_{\overline{\mathbf{x}}_i}) - \lambda(\nabla\phi(\overline{\mathbf{x}})\cdot\mathbf{u}^*_{\overline{\mathbf{x}}_i} + g(\overline{\mathbf{x}},\mathbf{u}^*_{\overline{\mathbf{x}}_i})) > \frac{C_1}{\epsilon}h_{max}. \tag{5.50}$$

Since $\mathbf{u}^*_{\overline{\mathbf{x}}_i}$ is not necessarily the maximizer of $H(\overline{\mathbf{x}},\nabla\phi(\overline{\mathbf{x}}))$,

$$-\lambda(\nabla\phi(\overline{\mathbf{x}})\cdot\mathbf{u}^*_{\overline{\mathbf{x}}_i} + g(\overline{\mathbf{x}},\mathbf{u}^*_{\overline{\mathbf{x}}_i})) \le \lambda H(\overline{\mathbf{x}},\nabla\phi(\overline{\mathbf{x}})). \tag{5.51}$$

It will now be shown that at most one of $\overline{\mathbf{x}}_i$ or $\overline{\mathbf{x}}$ can be in Ω. Following (5.50) and using (3.7), (5.51), $\mathbf{u}^*_{\overline{\mathbf{x}}_i}$ is the optimizer of $H(\overline{\mathbf{x}}_i,\nabla\psi(\overline{\mathbf{x}}_i))$,

$$-H(\overline{\mathbf{x}}_i,\nabla\psi(\overline{\mathbf{x}}_i)) + \lambda H(\overline{\mathbf{x}},\nabla\phi(\overline{\mathbf{x}})) > \frac{C_1}{\epsilon}h_{max}. \tag{5.52}$$

Case 1: Let $\overline{\mathbf{x}}\in\Omega$. From Definition 3.2.3, $H(\overline{\mathbf{x}},\nabla\phi(\overline{\mathbf{x}}))\le 0$. From (5.52),

$$H(\overline{\mathbf{x}}_i,\nabla\psi(\overline{\mathbf{x}}_i)) < -\frac{C_1}{\epsilon}h_{max}. \tag{5.53}$$

Since $\psi(\mathbf{x}_i)\le\widetilde{V}(\mathbf{x}_i)$ for all $\mathbf{x}_i\in X$, $\psi(\overline{\mathbf{x}}_i)=\widetilde{V}(\overline{\mathbf{x}}_i)$, by Theorem 5.2.14 and Definition 5.2.7,

$$\widetilde{H}[\overline{\mathbf{NF}},\widetilde{V}](\overline{\mathbf{x}}_i,\widetilde{V}(\overline{\mathbf{x}}_i)) = \widetilde{H}[S,\widetilde{V}](\overline{\mathbf{x}}_i,\widetilde{V}(\overline{\mathbf{x}}_i)) \le \widetilde{H}[S,\psi](\overline{\mathbf{x}}_i,\psi(\overline{\mathbf{x}}_i)). \tag{5.54}$$

It will be shown that $\overline{\mathbf{x}}_i\in X\cap\Omega^c$ using proof by contrapositive. Since $\widetilde{V}$ is a solution to the numerical HJB equation (5.21), it is a supersolution of the numerical HJB equation (Definition 5.3.9). If $\overline{\mathbf{x}}_i\in X\cap\Omega$,

$$\widetilde{H}[\overline{\mathbf{NF}},\widetilde{V}](\overline{\mathbf{x}}_i,\widetilde{V}(\overline{\mathbf{x}}_i)) = \widetilde{H}[S,\widetilde{V}](\overline{\mathbf{x}}_i,\widetilde{V}(\overline{\mathbf{x}}_i)) \ge 0 \tag{5.55}$$

holds. Furthermore if $\overline{\mathbf{x}_i}\in X\cap\Omega$, Theorem 5.2.15 must also hold. That is, since $\|\nabla^2\psi\|_2 = \frac{1}{\epsilon}$,

$$|H(\overline{\mathbf{x}}_i,\nabla\psi(\overline{\mathbf{x}}_i)) - \widetilde{H}[S,\psi](\overline{\mathbf{x}}_i,\psi(\overline{\mathbf{x}}_i))| \le \frac{C_1}{\epsilon}h_{max}. \tag{5.56}$$

It will be shown (5.55) and (5.56) cannot be simultaneously true, implying $\overline{\mathbf{x}}_i\in X\cap\Omega^c$. If (5.55) is true, then by (5.54), $\widetilde{H}[S,\psi](\overline{\mathbf{x}}_i,\psi(\overline{\mathbf{x}}_i))\ge 0$. By (5.53),

$$H(\overline{\mathbf{x}}_i,\nabla\psi(\overline{\mathbf{x}}_i)) - \widetilde{H}[S,\psi](\overline{\mathbf{x}}_i,\psi(\overline{\mathbf{x}}_i)) < -\frac{C_1}{\epsilon}h_{max}.$$

Therefore (5.56) is false.

Otherwise, if (5.56) were true, using (5.53),

$$H(\overline{\mathbf{x}}_i,\nabla\psi(\overline{\mathbf{x}}_i)) - \widetilde{H}[S,\psi](\overline{\mathbf{x}}_i,\psi(\overline{\mathbf{x}}_i)) \ge -\frac{C_1}{\epsilon}h_{max} > H(\overline{\mathbf{x}}_i,\nabla\psi(\overline{\mathbf{x}}_i)).$$

Hence with (5.54),

$$\widetilde{H}[\overline{\mathbf{NF}}, \widetilde{V}](\overline{\mathbf{x}}_i, \widetilde{V}(\overline{\mathbf{x}}_i)) = \widetilde{H}[S, \widetilde{V}](\overline{\mathbf{x}}_i, \widetilde{V}(\overline{\mathbf{x}}_i)) \leq \widetilde{H}[S, \psi](\overline{\mathbf{x}}_i, \psi(\overline{\mathbf{x}}_i)) < 0.$$

Therefore (5.55) is false. Hence $\overline{\mathbf{x}}_i \in X \cap \Omega^c$.

Case 2: If $\overline{\mathbf{x}}_i \in X \cap \Omega$, from Theorem 5.2.15,

$$\widetilde{H}[S, \psi](\overline{\mathbf{x}}_i, \psi(\mathbf{x}_i)) - H(\overline{\mathbf{x}}_i, \nabla\psi(\overline{\mathbf{x}}_i)) \leq \frac{C_1}{\epsilon} h_{max}. \tag{5.57}$$

From (5.54), and $\widetilde{V}$ is a supersolution of the numerical HJB equation (5.21),

$$\widetilde{H}[S, \psi](\overline{\mathbf{x}}_i, \psi(\overline{\mathbf{x}}_i)) \geq \widetilde{H}[S, \widetilde{V}](\overline{\mathbf{x}}_i, \widetilde{V}(\overline{\mathbf{x}}_i)) = \widetilde{H}[\overline{\mathbf{NF}}, \widetilde{V}](\overline{\mathbf{x}}_i, \widetilde{V}(\overline{\mathbf{x}}_i)) \geq 0$$

From (5.52) and (5.57),

$$\frac{C_1}{\epsilon} h_{max} + \widetilde{H}[S, \psi](\overline{\mathbf{x}}_i, \psi(\overline{\mathbf{x}}_i)) - \lambda H(\overline{\mathbf{x}}, \nabla\phi(\overline{\mathbf{x}})) < \frac{C_1}{\epsilon} h_{max}. \tag{5.58}$$

Since $\overline{\mathbf{x}}_i \in \Omega \cap X$, $\widetilde{H}[S, \psi](\overline{\mathbf{x}}_i, \psi(\overline{\mathbf{x}}_i)) \geq 0$, from (5.58), and $0 < \lambda < 1$,

$$H(\overline{\mathbf{x}}, \nabla\phi(\overline{\mathbf{x}})) > 0,$$

which implies by Definition 3.2.3 of the viscosity subsolution, $\overline{\mathbf{x}} \in \partial\Omega$. Hence at most one of $\overline{\mathbf{x}}$ and $\overline{\mathbf{x}}_i$ can belong to Ω.

Step 2: An upper bound on $M_{\epsilon,\lambda}$ will be found. The cases in this step are the same as Step 1. The scenario of $\overline{\mathbf{x}}$ and $\overline{\mathbf{x}}_i$ both being outside Ω is considered in Case 1 below.

Case 1: $\overline{\mathbf{x}} \in \overline{\Omega}$, $\overline{\mathbf{x}}_i \in X \cap \Omega^c$.

Let $\check{\mathbf{x}} = \arg\min_{\mathbf{x} \in \partial\Omega} \|\overline{\mathbf{x}}_i - \mathbf{x}\|$. Let $\underline{\mathbf{x}}_i$ be the point on the line from $\overline{\mathbf{x}}$ and $\overline{\mathbf{x}}_i$ intersecting $\partial\Omega$. For $\overline{\mathbf{x}} \in \partial\Omega$, $\underline{\mathbf{x}}_i = \overline{\mathbf{x}}$. Since $\overline{\Omega}$ is convex, by Lemma 5.3.1, the angle between vectors $\underline{\mathbf{x}}_i - \check{\mathbf{x}}$ and $\overline{\mathbf{x}}_i - \check{\mathbf{x}}$ is nonacute.

$$\begin{aligned}
\|\underline{\mathbf{x}}_i - \overline{\mathbf{x}}_i\|^2 &= \|\underline{\mathbf{x}}_i - \check{\mathbf{x}}\|^2 + \|\check{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2 - 2\langle \underline{\mathbf{x}}_i - \check{\mathbf{x}}, \overline{\mathbf{x}}_i - \check{\mathbf{x}} \rangle, \\
&\geq \|\underline{\mathbf{x}}_i - \check{\mathbf{x}}\|^2 . \\
\|\underline{\mathbf{x}}_i - \overline{\mathbf{x}}_i\| &\geq \|\underline{\mathbf{x}}_i - \check{\mathbf{x}}\| .
\end{aligned}$$

Recall $\underline{\mathbf{x}}_i$ is on the line from $\overline{\mathbf{x}}$ to $\overline{\mathbf{x}}_i$, $\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\| = \|\overline{\mathbf{x}} - \underline{\mathbf{x}}_i\| + \|\underline{\mathbf{x}}_i - \overline{\mathbf{x}}_i\|$. With the triangle inequality,

$$\begin{aligned}
\|\overline{\mathbf{x}} - \underline{\mathbf{x}}_i\| + \|\underline{\mathbf{x}}_i - \overline{\mathbf{x}}_i\| &\geq \|\overline{\mathbf{x}} - \underline{\mathbf{x}}_i\| + \|\underline{\mathbf{x}}_i - \check{\mathbf{x}}\| , \\
\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\| &\geq \|\overline{\mathbf{x}} - \check{\mathbf{x}}\| .
\end{aligned}$$

By the Lipschitz-continuity of V with constant C_0, $0 < \lambda < 1$, the boundedness of $|\widetilde{V}| \le C_0$, and and since $\widetilde{V}$ is a supersolution to the numerical HJB equation (5.21), $\widetilde{V}(\overline{\mathbf{x}}_i) \ge q(\check{\mathbf{x}})$,

$$\begin{aligned} M_{\epsilon,\lambda} &= \lambda V(\overline{\mathbf{x}}) - \widetilde{V}(\overline{\mathbf{x}}_i) - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}, \\ &= \lambda(V(\overline{\mathbf{x}}) - \widetilde{V}(\overline{\mathbf{x}}_i)) - (1-\lambda)\widetilde{V}(\overline{\mathbf{x}}_i) - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}, \\ &\le \lambda(V(\overline{\mathbf{x}}) - q(\check{\mathbf{x}})) + (1-\lambda)C_0 - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}, \end{aligned} \tag{5.59}$$

If $\overline{\mathbf{x}} \in \Omega$, $V(\check{\mathbf{x}}) = q(\check{\mathbf{x}})$, from (5.59),

$$M_{\epsilon,\lambda} \le \lambda(V(\overline{\mathbf{x}}) - V(\check{\mathbf{x}})) + (1-\lambda)C_0 - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}. \tag{5.60}$$

Otherwise $\overline{\mathbf{x}} \in \partial\Omega$, and $V(\overline{\mathbf{x}}) = q(\overline{\mathbf{x}})$, from (5.59),

$$M_{\epsilon,\lambda} \le \lambda(q(\overline{\mathbf{x}}) - q(\check{\mathbf{x}})) + (1-\lambda)C_0 - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}. \tag{5.61}$$

The Lipschitz continuity of both q and V with constant C_0 in (5.60) and (5.61) and $\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\| \ge \|\overline{\mathbf{x}} - \check{\mathbf{x}}\|$ yield

$$M_{\epsilon,\lambda} \le C_0\lambda \|\overline{\mathbf{x}} - \check{\mathbf{x}}\| + (1-\lambda)C_0 - \frac{\|\overline{\mathbf{x}} - \check{\mathbf{x}}\|^2}{2\epsilon}, \tag{5.62}$$

which is quadratic in $\|\overline{\mathbf{x}} - \check{\mathbf{x}}\|$. The quadratic is maximized with $\|\overline{\mathbf{x}} - \check{\mathbf{x}}\| = \epsilon\lambda C_0$. Thus, with $0 < \lambda < 1$,

$$M_{\epsilon,\lambda} \le (1-\lambda)C_0 + \frac{C_0^2\epsilon}{2}. \tag{5.63}$$

Case 2: $\overline{\mathbf{x}} \in \partial\Omega$, $\overline{\mathbf{x}}_i \in X \cap \Omega$.

From Lemma 5.3.7, there exists $\hat{\mathbf{x}}_i \in X \cap \Omega^c$ such that

$$\|\overline{\mathbf{x}} - \hat{\mathbf{x}}_i\| \le h_{max}. \tag{5.64}$$

Let $\widetilde{\mathbf{x}} = \arg\min_{\mathbf{x}\in\partial\Omega} \|\hat{\mathbf{x}}_i - \mathbf{x}\|$. Using $0 < \lambda < 1$, $\widetilde{V}(\hat{\mathbf{x}}_i) \ge q(\widetilde{\mathbf{x}})$, $V(\mathbf{x}) \le q(\mathbf{x})$ for $\mathbf{x} \in \partial\Omega$, Lipschitz-continuity of q and $\widetilde{V}$ both with constant C_0,

$$\begin{aligned}
M_{\epsilon,\lambda} &= \lambda V(\overline{\mathbf{x}}) - \widetilde{V}(\overline{\mathbf{x}}_i) - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}, \\
&= \lambda(V(\overline{\mathbf{x}}) - \widetilde{V}(\overline{\mathbf{x}}_i)) - (1-\lambda)\widetilde{V}(\overline{\mathbf{x}}_i) - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}, \\
&\leq \lambda(q(\overline{\mathbf{x}}) - q(\widetilde{\mathbf{x}}) + q(\widetilde{\mathbf{x}}) - \widetilde{V}(\overline{\mathbf{x}}_i)) + (1-\lambda)C_0 - \frac{|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}, \\
&\leq \lambda C_0 \|\overline{\mathbf{x}} - \widetilde{\mathbf{x}}\| + \lambda(\widetilde{V}(\hat{\mathbf{x}}_i) - \widetilde{V}(\overline{\mathbf{x}}_i)) + (1-\lambda)C_0 - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}, \\
&\leq C_0(\|\overline{\mathbf{x}} - \widetilde{\mathbf{x}}\| + \|\hat{\mathbf{x}}_i - \overline{\mathbf{x}}_i\|) + (1-\lambda)C_0 - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon},
\end{aligned}$$

Using the triangle inequality $\|\hat{\mathbf{x}}_i - \overline{\mathbf{x}}_i\| \leq \|\hat{\mathbf{x}}_i - \widetilde{\mathbf{x}}\| + \|\widetilde{\mathbf{x}} - \overline{\mathbf{x}}\| + \|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|$,

$$M_{\epsilon,\lambda} \leq (1-\lambda)C_0 + C_0(\|\overline{\mathbf{x}} - \widetilde{\mathbf{x}}\| + \|\hat{\mathbf{x}}_i - \widetilde{\mathbf{x}}\| + \|\widetilde{\mathbf{x}} - \overline{\mathbf{x}}\| + \|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|) - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}.$$

By Lemma 5.3.1, and the cosine law, $\|\overline{\mathbf{x}} - \widetilde{\mathbf{x}}\| \leq \|\overline{\mathbf{x}} - \hat{\mathbf{x}}_i\|$. From the definition of $\widetilde{\mathbf{x}}$, $\|\hat{\mathbf{x}}_i - \widetilde{\mathbf{x}}\| \leq \|\overline{\mathbf{x}} - \hat{\mathbf{x}}_i\|$. Therefore,

$$M_{\epsilon,\lambda} \leq (1-\lambda)C_0 + 3C_0 \|\overline{\mathbf{x}} - \hat{\mathbf{x}}_i\| + C_0 \|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\| - \frac{\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|^2}{2\epsilon}.$$

From (5.64) and maximizing over the quadratic $\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\|$ as before with $\|\overline{\mathbf{x}} - \overline{\mathbf{x}}_i\| = C_0\epsilon$,

$$M_{\epsilon,\lambda} \leq (1-\lambda)C_0 + 3C_0 h_{max} + \frac{C_0^2\epsilon}{2}. \tag{5.65}$$

Step 3: The upper bound of $M_{\epsilon,\lambda}$ in (5.65) is larger than (5.63). Using (5.47),

$$\begin{aligned}
V(\mathbf{x}_i) - \widetilde{V}(\mathbf{x}_i) &\leq C_0(1-\lambda) + M_{\epsilon,\lambda}, \\
&\leq 2C_0(1-\lambda) + 3C_0 h_{max} + \frac{C_0^2\epsilon}{2}, \\
&\leq 2C_0 \frac{2}{G_{min}} \left(\frac{C_1}{\epsilon} h_{max} + C_0 L_g \epsilon \right) + 3C_0 h_{max} + \frac{C_0^2\epsilon}{2}, \\
&\leq \max\left\{ \frac{4C_0C_1}{G_{min}}, \frac{4C_0^2 L_g}{G_{min}} + \frac{C_0^2}{2} \right\} \left(\frac{h_{max}}{\epsilon} + \epsilon \right) + 3C_0 h_{max},
\end{aligned}$$

Since $\epsilon = \sqrt{h_{max}}$ is a global minimum of $(\frac{h_{max}}{\epsilon} + \epsilon)$,

$$V(\mathbf{x}_i) - \widetilde{V}(\mathbf{x}_i) \leq 2 \max\left\{\frac{4C_0C_1}{G_{min}}, \frac{4C_0^2L_g}{G_{min}} + \frac{C_0^2}{2}\right\}\sqrt{h_{max}} + 3C_0h_{max}.$$

Setting $C = (4\max\{\frac{2C_0C_1}{G_{min}}, \frac{2C_0^2L_g}{G_{min}} + \frac{C_0^2}{2}\} + 3C_0)$, if $h_{max} < h_0 < 1$, and hence $h_{max} < \sqrt{h_{max}}$,

$$V(\mathbf{x}_i) - \widetilde{V}(\mathbf{x}_i) \leq C\sqrt{h_{max}}.$$

Step 4: To complete the proof,

$$\sup_{\mathbf{x}_i \in X}\left\{\widetilde{V}(\mathbf{x}_i) - V(\mathbf{x}_i)\right\} \leq C\sqrt{h_{max}}. \tag{5.66}$$

must be shown. A symmetrical argument using V a viscosity supersolution of (3.6), and $\widetilde{V}$ a subsolution of the numerical HJB equation (5.21) can be used. Hence,

$$\sup_{\mathbf{x}_i \in X} |\widetilde{V}(\mathbf{x}_i) - V(\mathbf{x}_i)| \leq C\sqrt{h_{max}}$$

for $h_{max} \leq h_0$. $\square$

The convergence over the continuous solution is now shown. Recall $\overline{\Omega}_X \supseteq \overline{\Omega}$ (5.14) is the largest closed region in $\mathbb{R}^n$ contained in X. Define $\hat{V} : \overline{\Omega}_X \to \mathbb{R}$

$$\hat{V}(\mathbf{x}) = \sum_{j=0}^{n} \zeta_j V(\mathbf{x}_j^{\mathbf{s}}) \text{ for } \mathbf{x} = \sum_{j=0}^{n} \zeta_j \mathbf{x}_j^{\mathbf{s}}. \tag{5.67}$$

On $\mathbf{x}_i \in X$, $V(\mathbf{x}_i) = \hat{V}(\mathbf{x}_i)$ are equal.

Lemma 5.3.12. *There exists $D_1 > 0$ for all $\boldsymbol{x} \in \overline{\Omega}_X$, such that*

$$|V(\boldsymbol{x}) - \hat{V}(\boldsymbol{x})| \leq D_1 h_{max}. \tag{5.68}$$

Proof. Let $\zeta \in \Xi_n$ and $\mathbf{x} \in \mathbf{s}$ such that $\mathbf{x} = \sum_{j=0}^{n} \zeta_j \mathbf{x}_j^{\mathbf{s}}$. Using $V(\mathbf{x}_i) = \hat{V}(\mathbf{x}_i)$ for all vertices $\mathbf{x}_i \in X$, $\sum_{j=0}^{n} \zeta_j = 1$ and V is Lipschitz-continuous with constant $2G_{max}$ (Theorem 5.3.2),

$$\begin{aligned}
|V(\mathbf{x}) - \hat{V}(\mathbf{x})| &= \Big|\sum_{j=0}^{n} \zeta_j \left(V(\mathbf{x}) - \hat{V}(\mathbf{x}_j)\right)\Big| \\
&\leq \sum_{j=0}^{n} \zeta_j |V(\mathbf{x}) - V(\mathbf{x}_j)| \\
&\leq \sum_{j=0}^{n} \zeta_j (2G_{max}\|\mathbf{x} - \mathbf{x}_j\|) \\
&\leq 2G_{max}h_{max}.
\end{aligned}$$

Hence $D_1 = 2G_{max}$. $\square$

Corollary 5.3.13. *There exists $D_2 > 0$ for all $\boldsymbol{x} \in \overline{\Omega}_X$ such that*

$$|V(\boldsymbol{x}) - \widetilde{V}(\boldsymbol{x})| \leq D_2\sqrt{h_{max}}, \tag{5.69}$$

for $h_{max} < h_0$ as described in Theorem 5.3.11.

Proof. Let $\zeta \in \Xi_n$ and $\mathbf{x} \in \mathbf{s}$ such that $\mathbf{x} = \sum_{j=0}^{n} \zeta_j \mathbf{x}_j^{\mathbf{s}}$. From Lemma 5.3.12, and Theorem 5.3.11,

$$-D_1 h_{max} + \hat{V}(\mathbf{x}) \leq V(\mathbf{x}) \leq D_1 h_{max} + \hat{V}(\mathbf{x}), \tag{5.70}$$

$$D_1 h_{max} - \hat{V}(\mathbf{x}) \geq -V(\mathbf{x}) \geq -D_1 h_{max} - \hat{V}(\mathbf{x}), \tag{5.71}$$

For $\mathbf{x} \in \overline{\Omega}_X$, $\widetilde{V}(\mathbf{x}) = \sum_{j=0}^{n} \zeta_j \widetilde{V}(\mathbf{x}_j^{\mathbf{s}})$. From (5.70),

$$\begin{aligned}
V(\mathbf{x}) - \widetilde{V}(\mathbf{x}) &\leq D_1 h_{max} + \hat{V}(\mathbf{x}) - \widetilde{V}(\mathbf{x}) \\
&= D_1 h_{max} + \sum_{j=0}^{n} \zeta_j (V(\mathbf{x}_j^{\mathbf{s}}) - \widetilde{V}(\mathbf{x}_j^{\mathbf{s}})) \\
&\leq D_1 h_{max} + C\sqrt{h_{max}} \\
&\leq (D_1 + C)\sqrt{h_{max}},
\end{aligned}$$

for $h_{max} < h_0$. The proof for $\widetilde{V}(\mathbf{x}) - V(\mathbf{x})$ is symmetrical using (5.71). Hence $D_2 = D_1 + C$. $\square$

5.4 Conclusion

The rate of convergence of the approximate solution provided by OUM with prescribed boundary values was shown to be at least $\mathcal{O}(\sqrt{h_{max}})$. The basic idea of the proof was an extension of that in [61]. A key step was to show the existence of a directionally complete stencil, discussed in [5]. This led to a proof that the numerical Hamiltonian for the OUM was both consistent and monotonic. The consistency and monotonicity of the numerical Hamiltonian was used in a proof similar to the comparison principle in which the error bound is shown. One extension of this work would be to provide a convergence rate proof for the single-source point formulation of the static HJB. This is complicated by the lack of an obvious directionally complete stencil near the source point that yields a consistent solution.

Numerical evidence [72] indicates that convergence of the approximating solution computed by the OUM to the exact solution may occur at a rate of $\mathcal{O}(h_{max})$ for some examples in the boundary value problem. It would interesting to establish conditions under which OUM does converge with rate $\mathcal{O}(h_{max})$, but it is believed that a different technique than used here is required.

Chapter 6

Simulations

To illustrate the theory presented in the previous chapters, simulations were performed. The algorithms described in this section were programmed in MATLAB®. All timed computations were performed using `tic` and `toc` commands from MATLAB on a Lenovo ThinkPad E520 Laptop with Intel® Core ™ i5 -2430M CPU Processor (2.4 GHz $\times$ 2) with 4GB RAM. The mesh discretizations were generated using Mesh2D [25] available from MATLAB Central. The meshes were manipulated using functions from the Numerical Tours Package available online [65].

The optimal path planning problem presented in Chapter 3 considering rover objectives (Section 3.4) was solved numerically using FMM, OUM and OUM-BD presented in Chapter 4. An approximated solution $\widetilde{V}$ of the dynamic programming principle (3.5) was found. Solving the ordinary differential equation (3.8), the optimal paths were recovered. The optimal paths were compared to paths generated by a genetic algorithm based rover path planner [28] using the cost function (3.2) for the continuous problem. Three examples were considered. The first was a cluttered environment. The second considered tipover-stability risk while all of the environment weights including terrain and solar energy were considered in the third example.

A comparison was made between the algorithms presented in Chapter 4 that could handle direction-dependent weights. The time required and number of updates were compared on several mesh refinements on the same problem.

In Chapter 5, a convergence rate of $\mathcal{O}(\sqrt{h_{max}})$ was proven for the Ordered Upwind Method in the context of a boundary value problem. The analytically proven convergence rate will be shown numerically. Additional experiments demonstrating numerical convergence rate of OUM were presented in [5, 72].

The chapter is outlined as follows. In Section 6.1, aspects of the implementation of the Ordered Upwind Method will be described. In Section 6.2, the rover path planning problem will be considered, and the weights introduced in Chapter 3 will be used. In Section 6.3, algorithms described in Chapter 4 that could handle direction-dependent weights will be compared in timing and number of updates using discretizations with different-sized elements. Finally, the rate of convergence for OUM will be shown numerically in Section 6.4 for the types of problems described in Chapter 5. Some conclusions are presented in Section 6.5.

6.1 Implementation of Ordered Upwind Method

The following algorithmic improvements suggested in [72] were used in the implementation of both OUM and OUM-BD.

6.1.1 Local Anisotropy Coefficient

Rather than using the global anisotropy coefficient $\Gamma = \frac{G_{max}}{G_{min}}$ (Definition 3.1.8) as the search radius in the *Near Front* of $\mathbf{x}_i$ (4.11), a local anisotropy coefficient was used. At each $\mathbf{x}_i \in X$, define

$$\Gamma(\mathbf{x}_i) = \frac{g_{max}(\mathbf{x}_i)}{g_{min}(\mathbf{x}_i)},$$

where

$$g_{max}(\mathbf{x}) = \max_{\mathbf{u}\in\mathbb{S}^{n-1}} g(\mathbf{x},\mathbf{u}) \text{ and } g_{min}(\mathbf{x}) = \min_{\mathbf{u}\in\mathbb{S}^{n-1}} g(\mathbf{x},\mathbf{u}).$$

The minimizing update (4.14) is shown using Lemma 3.1.9 to come from within $\Gamma(\mathbf{x}_i)h_{max}$ of $\mathbf{x}_i$ [72]. This computational improvement was especially beneficial for problems where $\Gamma(\mathbf{x}_i)$ in the weight g was not constant throughout the workspace $\overline{\Omega}$. For direction-independent problems, $\Gamma(\mathbf{x}_i) = 1$, whereas Γ is larger when $G_{min} \neq G_{max}$.

6.1.2 Doubling the Search Radius

The value $\widetilde{V}(\mathbf{x}_i)$ at $\mathbf{x}_i$ is updated in two possible instances during the OUM algorithm. The first is when $\mathbf{x}_i$ is labelled *Considered*, and the second is when a vertex $\mathbf{x}_j \in B_{\Gamma(\mathbf{x}_i)h_{max}}(\mathbf{x}_i)$

is relabelled *Accepted*. In order to eliminate the update in the second case, the *Near Front* of $\mathbf{x}_i$ was widened to

$$\hat{\mathbf{NF}}(\mathbf{x}_i) = \left\{ \mathbf{s} \in \mathbf{AF} \middle| \; \exists \; \widetilde{\mathbf{x}} \in \mathbf{s}, \|\widetilde{\mathbf{x}} - \mathbf{x}_i\| \leq 2\Gamma(\mathbf{x}_i) h_{max} \right\}$$

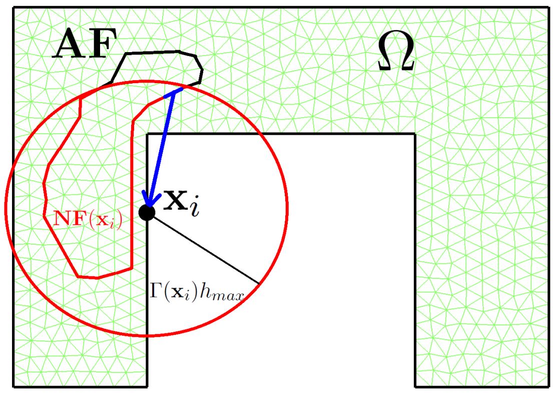
in the initial update. It was discussed in [72, Lemma 3.4,7.3] that if the boundary function $q(\mathbf{x})$ is compatible (Definition 5.9) the update cannot come from further away than $2\Gamma(\mathbf{x}_i)h_{max}$. Updating $\mathbf{x}_i$ from $\hat{\mathbf{NF}}(\mathbf{x}_i)$ when it is labelled *Considered* will include the approximate characteristic direction. Though this does not lead to a substantial reduction in the number of updates (4.13), the search to detect $\mathbf{x}_j \in B_{\Gamma(\mathbf{x}_i)h_{max}}(\mathbf{x}_i)$ being *Accepted* is no longer required. Since the update may potentially come from farther away than in the original algorithm, the calculated values may be less accurate. The resulting $\widetilde{V}$ has been shown numerically [72] to attain the same order of accuracy.

6.1.3 Non-Convex Regions and Varying Weight Function

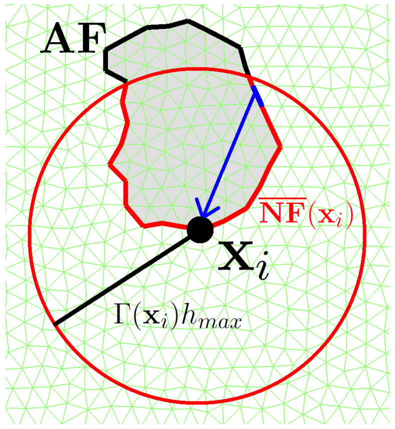
Recall in the Ordered Upwind Method, a vertex $\mathbf{x}_i$ is updated from its *Near Front* $\mathbf{NF}(\mathbf{x}_i)$ (4.14) in Steps 5 and 6 of the OUM algorithm. If the workspace $\overline{\Omega}$ is not convex, an update from an $(n-1)$-simplex $\mathbf{s} \in \mathbf{NF}(\mathbf{x}_i)$ may travel outside of $\overline{\Omega}$ (see Figure 6.1a).

Obstacles in the weight function g will exhibit the same behaviour. If the update from a $(n-1)$-simplex travels through n-simplices of X made of vertices labelled *Far*, then the update may be incorrect as the weight of intermediate points are not considered. See Figure 6.1. This behaviour is not limited to obstacles, but can occur updates travel through a region where the weight function g experiences large variations.

In the implementation of OUM, a check is performed to determine if an update from a vertex on the *Near Front* would intersect with these problematic simplices. If an intersection occurs, then those vertices are omitted in the construction of the *Near Front*. For known obstacles and boundaries, these simplices can be determined ahead of time separately. Excluding these vertices ensures that the (incorrect) update cannot come from them. The problem is naturally resolved in the limit as $h_{max} \to 0$ for Lipschitz-continuous boundaries for non-convex domains and obstacles since the search radius of the *Near Front* of $\mathbf{x}_i$ decreases linearly with h_{max}. See Figure 6.2. For implementation purposes, the h_{max} required to avoid such situations may be small, especially if $\Gamma(\mathbf{x}_i)$ is large. Strict local minima in $\widetilde{V}$ can occur at a vertex $\mathbf{x}_i$ of X if the minimizing update is from a non-neighbouring edge $\mathbf{s}$, while its neighbours have smaller *Near Front* radii that exclude $\mathbf{s}$. Local minima can arise in both nonconvex domains and a large variation in the weight function g, but disappear as the mesh X is refined.



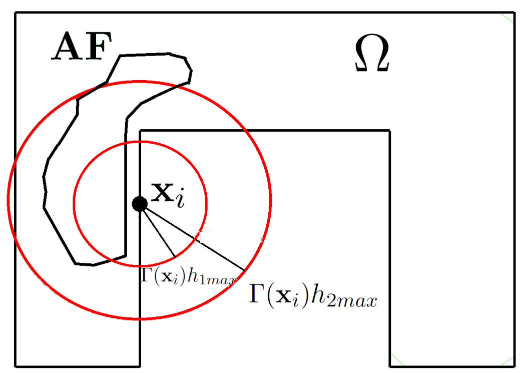
(a) In the source-point formulation above, the region $\overline{\Omega}$ is not convex. Updates from edges where the approximations exit $\overline{\Omega}$ are not valid.



AF
$\overline{\mathbf{NF}}(\mathbf{x}_i)$
$\mathbf{X}_i$
$\Gamma(\mathbf{x}_i)h_{max}$

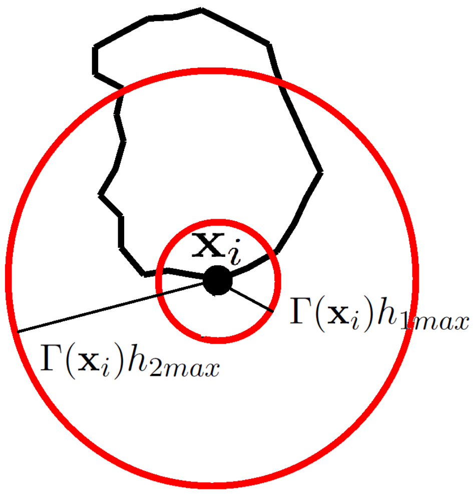
(b) The vertices inside the obstacle (shown shaded) are not labelled *Accepted*. The search radius $\Gamma(\mathbf{x}_i)h_{max}$ for the update $\widetilde{V}(\mathbf{x}_i)$ at $\mathbf{x}_i$ is large enough that the *Near Front* of $\mathbf{x}_i$, $\mathbf{NF}(\mathbf{x}_i)$, includes edges that traverse the obstacle, yielding a possibly incorrect update.

Figure 6.1: Incorrect updates in $\mathbb{R}^2$ - Obstacle regions and non-convex Ω.



AF
Ω
$\mathbf{x}_i$
$\Gamma(\mathbf{x}_i)h_{1max}$
$\Gamma(\mathbf{x}_i)h_{2max}$

(a) Using a finer discretization that reduces h_{max} as in Figure 6.1.



$\mathbf{x}_i$
$\Gamma(\mathbf{x}_i)h_{1max}$
$\Gamma(\mathbf{x}_i)h_{2max}$

(b) A finer discretization with smaller h_{max} may not include edges that enter the obstacle as did the larger circle seen in Figure 6.1.

Figure 6.2: Incorrect updates in $\mathbb{R}^2$ - As $h_{max} \to 0$ the problem is naturally resolved for Lipschitz continuous weight and boundary. A small h_{max} may be required. Instead, these simplices (edge) were removed when building the *Near Front*. Note $0 < h_{1max} < h_{2max}$.

The same issue does not occur when updates are only considered from neighbouring simplices such as in the Fast Marching, Fast Sweeping and Buffered Fast Marching Methods. Updates travel only between neighbouring simplices.

6.2 Comparison of OUM with FMM and Genetic Algorithm on Rover Objectives

All path planning problems were executed on the square workspace $\overline{\Omega} = [-500, 500] \times [-500, 500] \subset \mathbb{R}^2$ discretized by a mesh X made of 16765 vertices and 32888 triangles. The FMM and FMM-BD (only for direction-independent problems), OUM, OUM-BD, and a genetic algorithm (GA) path planner for rovers [28] were compared. For the FMM, FMM-BD, OUM, OUM-BD algorithms, the gradients to obtain the path were approximated linearly within each triangle of the solution based on the values of $\widetilde{V}$.

In the GA path planner [28], candidate paths were defined using cubic splines between a set of control points in $\overline{\Omega}$. All candidate paths had ends fixed at $\mathbf{x}_0$ and $\mathbf{x}_f$. A minimization was performed over the placement of control points according to the cost (3.2). The best results were obtained when 6-8 control points were used [28]. In the following examples, 8 control points were used with a population of 50 candidate paths. In the initialization of the 50 candidate paths, 9 were initialized randomly, 7 were the same straight line connecting $\mathbf{x}_0$ and $\mathbf{x}_f$ and the remaining paths were each single arcs with varying curvature. The crossover factor and mutation probabilities were chosen to be 0.9 and 0.1 respectively. The best candidate path was returned after 100 generations. Since a different path may be produced each time the algorithm is executed, both the average cost and the cost of the best path over 5 trials are presented.

The start and final positions were $\mathbf{x}_0 = (450, -450)$ and $\mathbf{x}_f = (-450, 450)$ respectively in each example. The minimizations in (3.8) and (4.13) were performed using the golden section search [67, Chapter 10.2]. Each value presented in Table 6.2 was averaged over 5 trials. The performance for all three examples was measured on the optimal path of the continuous cost function (3.2) using the trapezoidal rule to evaluate the integral, not the approximated value funtion $\widetilde{V}(\mathbf{x}_0)$. See Table 6.1. For examples 2 and 3, the rover is assumed to be rectangular with width 6 units and length 3 units.

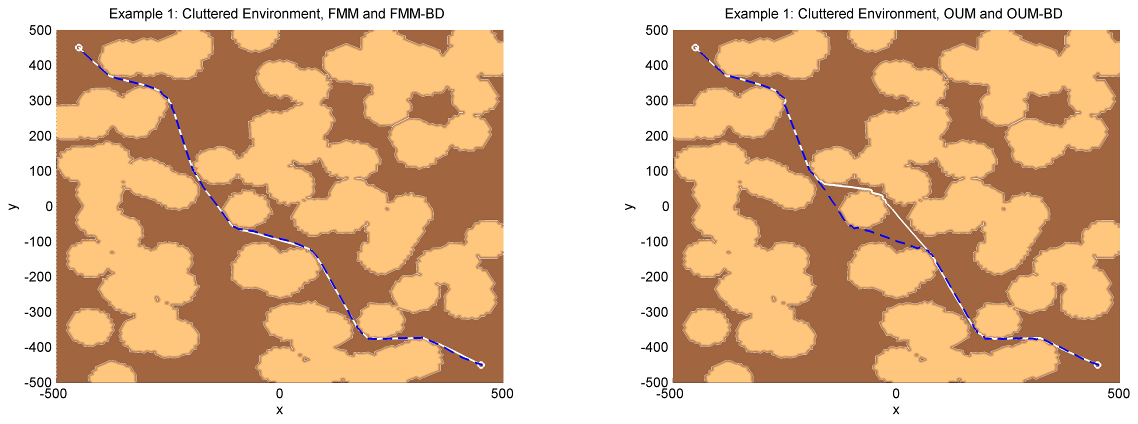
Example 1, Cluttered Environment - A hundred circular obstacles with radius 30 were generated with random locations in $\overline{\Omega}$. Let the set of obstacle regions be denoted O. The weight used to model obstacles was $g_o(\mathbf{x}) = 10^6$ for $\mathbf{x} \in O$, with linear decay to $g_o(\mathbf{x}) = 0$

Cost	**FMM**	**FMM-BD**	**OUM**	**OUM-BD**	**GA**
Example 1	1404.1	1405.2	1421.5	1417.8	NFP
Example 2	-	-	61.38	61.43	61.87 (61.79)
Example 3	-	-	796370	794120	820300 (818940)

Table 6.1: Performance - Costs for GA in all examples were averaged over 5 trials, with the bracketed value being the lowest cost of the 5 trials. Example 1: Path length with obstacle avoidance, Example 2: Tip-over stability risk, Example 3: All components. NFP indicates that no feasible path was found. All costs were measured using the produced path using (3.2) on the continuous weight g and not the value obtained by the value function $\widetilde{V}$.

Timings	**FMM**	**FMM-BD**	**OUM**	**OUM-BD**	**GA**
Example 1	17.19	5.99	33.60	11.68	2741
Example 2	-	-	85.99	55.70	6510
Example 3	-	-	43.56	37.95	5621

Table 6.2: Time required in seconds - All timings were averaged over 5 trials. Example 1: Path length with obstacle avoidance, the timing for GA was for 8 control points, Example 2: Tip-over stability risk, Example 3: All components. The timings presented for FMM, FMM-BD, OUM and OUM-BD included the time required to find the approximate optimal path in addition to computing $\widetilde{V}$.



(a) FMM in solid line, FMM-BD in dotted line. (b) OUM in solid line, OUM-BD in dotted line.

Figure 6.3: *Example 1*: Cluttered Environment - The GA planner was unable to find a feasible path. The costs of the paths are approximately the same between FMM, FMM-BD, OUM, OUM-BD. The difference in paths between OUM and OUM-BD is attributed to there being more than one optimal path. The section where they do not overlap has approximately the same length, which is reminiscent of the situation in Figure 3.7. See Tables 6.1 and 6.2.

at $\epsilon = 1$ from O. The weight function used was

$$g(\mathbf{x}) = 1 + g_o(\mathbf{x}).$$

Obstacle locations are shown in Figure 6.3.

The weight associated with this example did not depend on direction. All of FMM, FMM-BD, OUM and OUM-BD were compared. The cost (path length) between the paths were approximately the same. See Figure 6.3 and Table 6.1. The GA planner was executed using 8 control points (as described above) and using 25 control points. All other parameters were the same. A feasible path was not found by GA in either case (over any of the 5 trials for each). Timings are shown in Table (6.2). The OUM is slower than both the FMM and FMM-BD, as expected. The OUM-BD was only slightly slower than the FMM algorithm. The time required for the genetic algorithm planner (shown for 8 control points) is much higher, despite not producing a feasible path.

Example 2 - Tip-over stability risk - Recall that $z : \Omega \to \mathbb{R}$ denotes the terrain experienced by the rover (Definition 3.4.1). Let $\nabla z(\mathbf{x})_\perp$ be one of the level set directions of z at $\mathbf{x}$ such that $\nabla z(\mathbf{x}) \cdot \nabla z(\mathbf{x})_\perp = 0$. Using the force-angle stability margin α as defined in (3.18), let $m(\mathbf{x}) = \alpha\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})_\perp}{|\nabla z(\mathbf{x})_\perp|}\right)$ and $n(\mathbf{x}) = \alpha\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})}{|\nabla z(\mathbf{x})|}\right)$. Though there are two possible level set directions, due to symmetry of the rectangular rover, both level set directions yield the same stability margin $m(\mathbf{x})$.

The tip-over risk weight $g_{risk} : \overline{\Omega} \times \mathbb{S}^1 \to \mathbb{R}_+$ used was

$$g_{risk}(\mathbf{x}, \mathbf{u}) = \frac{1}{n(\mathbf{x})}\sqrt{1 + \left(\frac{n(\mathbf{x})^2}{m(\mathbf{x})^2} - 1\right)\left(\frac{\nabla z(\mathbf{x})}{|\nabla z(\mathbf{x})|} \cdot \mathbf{u}\right)^2}. \tag{6.1}$$

The resulting speed profile (3.11) $\mathcal{U}_{g_{risk}}(\mathbf{x})$ is an ellipse. The stability margins in directions $\nabla z(\mathbf{x})_\perp$ and $\nabla z(\mathbf{x})$ were

$$g_{risk}\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})_\perp}{|\nabla z(\mathbf{x})_\perp|}\right) = \frac{1}{n(\mathbf{x})} \quad \text{and} \quad g_{risk}\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})}{|\nabla z(\mathbf{x})|}\right) = \frac{1}{m(\mathbf{x})}$$

respectively. The maximum and minimum stability margin $\alpha(\mathbf{x}, \cdot)$ over $\mathbb{S}^1$ corresponded to directions $\nabla z(\mathbf{x})_\perp$ and $\nabla z(\mathbf{x})$ dependent on the length and the width of the rover. As the rover approached tip-over, g_{risk} became large. Finally, if either $m(\mathbf{x})$ or $n(\mathbf{x})$ were negative, then the corresponding weight in (6.1) was given the value 10^6.

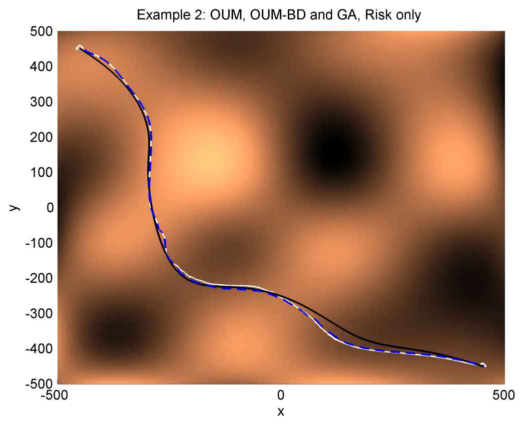
The weight used in example 2 was

$$g(\mathbf{x}, \mathbf{u}) = g_{risk}(\mathbf{x}, \mathbf{u}).$$

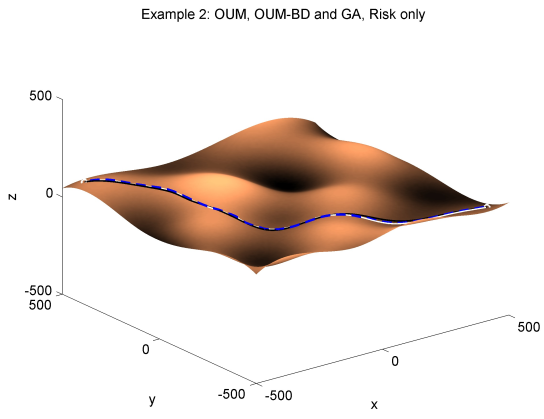
The terrain is shown in Figure 6.4. Since the weight was dependent on direction, FMM cannot be used. Note the difference between the safest path calculated using the OUM approach and the shortest path on the terrain in Figure 6.4c and 6.4d. A comparison is made with GA using the same weight $g_{risk}(\mathbf{x}, \mathbf{u})$. Though the costs of the paths were similar between OUM, OUM-BD and GA (see Table 6.1), the time required to find the path with GA was much higher. See Table 6.2.

Example 3 - Soil risk, solar energy and path length on terrain were considered in addition to obstacle avoidance, and tip-over stability risk as described in examples 1 and 2. The weight $g_{sr} : \overline{\Omega} \to \mathbb{R}_+$ used to model soil risk was independent of direction. A value was assigned for each type of soil. The soil and obstacle weight maps from [28] in Figure 6.5 were used. The solar panel was chosen to be parallel to the frame of the rover. The solar energy weight $g_{so} : \overline{\Omega} \to \mathbb{R}_+$ used was

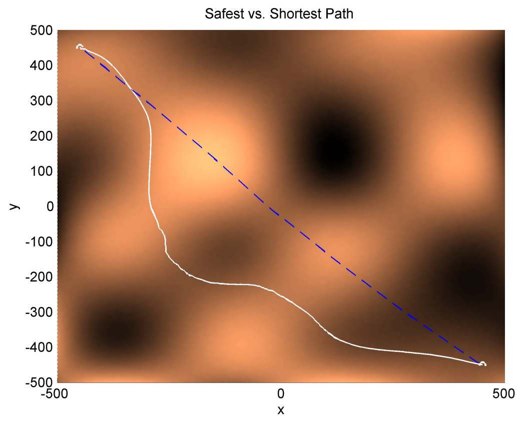
$$g_{so}(\mathbf{x}) = \begin{cases} (\beta(\mathbf{x}))^4, & \text{if } \beta(\mathbf{x}) \leq \frac{\pi}{2} \\ \left(\frac{\pi}{2}\right)^4, & \text{if } \beta(\mathbf{x}) > \frac{\pi}{2}. \end{cases} \tag{6.2}$$



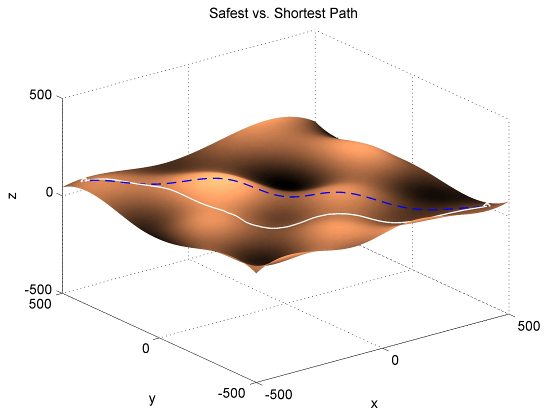
(a) OUM-BD - White, OUM - Dotted, GA - Black



(b) OUM-BD - White, OUM - Dotted, GA - Black

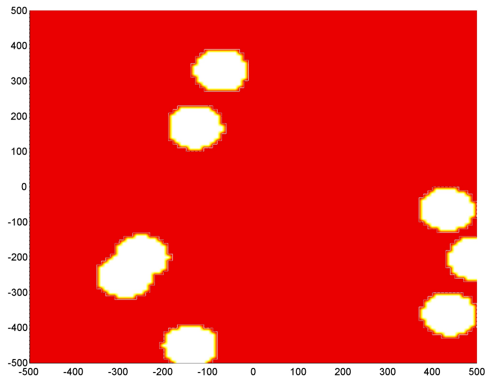


(c) Safest Path (OUM-BD) - Solid, Shortest Path (FMM-BD) - Dotted

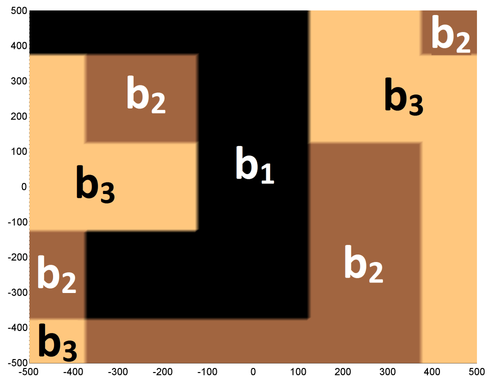


(d) Safest Path (OUM-BD) - Solid, Shortest Path (FMM-BD) - Dotted

Figure 6.4: *Example 2*: Tip-over stability risk. The paths found using OUM and OUM-BD were virtually identical. The safest path is quite different from the shortest path.



(a) Obstacles $g_o(\mathbf{x})$



(b) Soil Weight $g_{sr}(\mathbf{x})$, $b_1 = 0.5$, $b_2 = 1.5$ and $b_3 = 2.5$

Figure 6.5: *Example 3* - Weight maps [28].

A quartic was chosen to mimic a drastic drop in energy absorbed for an angle $\beta(\mathbf{x}) > \pi/4$.

The weight function used in example 3 was

$$g(\mathbf{x}, \mathbf{u}) = 1 + a_o g_o(\mathbf{x}) + a_{sr} g_{sr}(\mathbf{x}) + a_{so} g_{so}(\mathbf{x}) + a_{risk} g_{risk}(\mathbf{x}, \mathbf{u}) \tag{6.3}$$

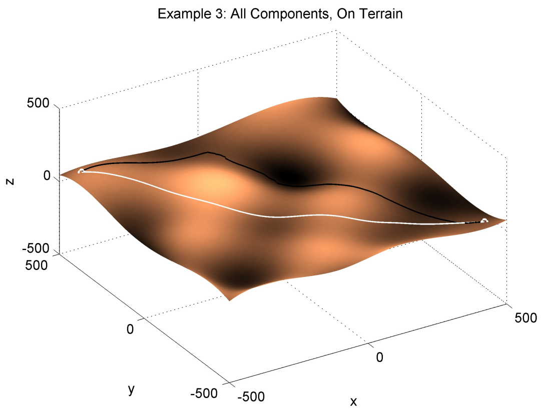
where $a_o, a_{sr}, a_{so}, a_{risk} \geq 0$ were chosen to reflect the relative importance of the various weights.

The parameters $a_{sr} = 100$, $a_{risk} = 1000$, $a_{so} = 10$ were chosen. The paths are shown in Figure 6.6. The cost and timings are presented in Tables 6.1 and 6.2 respectively. The GA path planner was unable to find a path with lower cost on the other side of the obstacles.

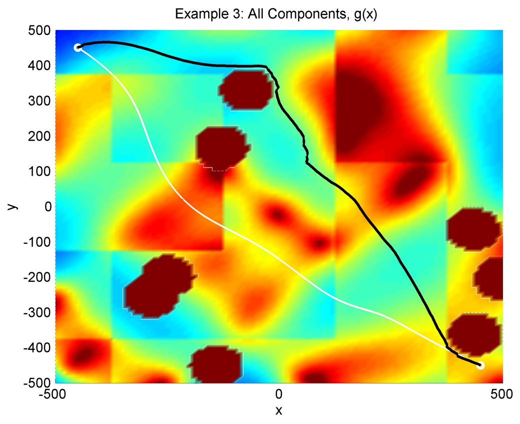
The regions $\overline{\Omega}$ for which $\widetilde{V}$ is found using OUM-BD are shown in Figure 6.7 for each example. In terms of timing, OUM-BD was always faster than OUM.

6.3 Comparison of Static HJB equation Algorithms

The Ordered Upwind Method (OUM), Monotone Acceptance OUM (MAOUM), Buffered Fast Marching (BFM) and Fast Sweeping Method (FSM) were compared for a problem with

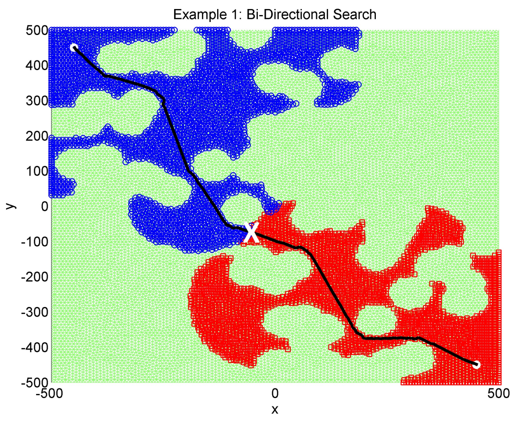


(a) OUM-BD/OUM (visually identical) - Black, GA - White superimposed on the terrain.

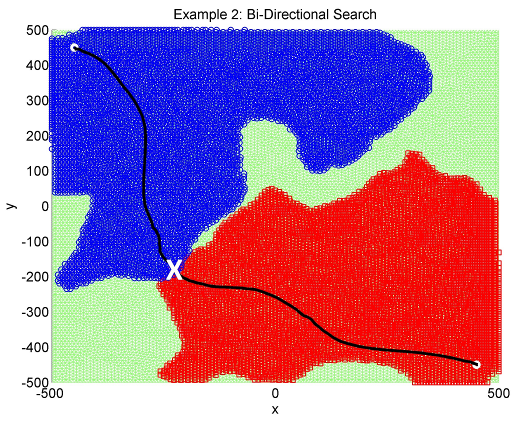


(b) OUM-BD/OUM (visually identical) - Black, GA - White superimposed on the sum of the direction-independent weights.

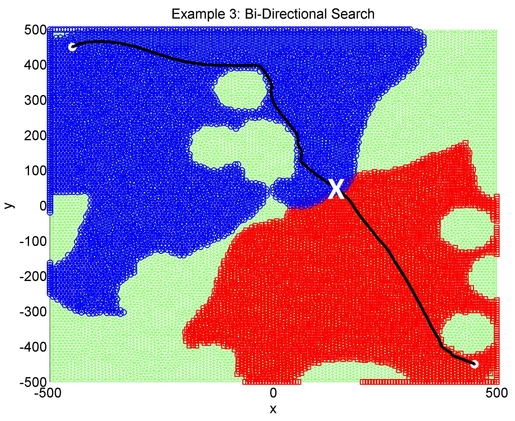
Figure 6.6: *Example 3* - Optimal path with all terms included in weight $g(\mathbf{x}, \mathbf{u})$: $a_{sr} = 100$ $(b_1 = 1, b_2 = 2, b_3 = 3)$, $a_{risk} = 1000$, $a_{so} = 10$, with obstacles $(g_o(\mathbf{x}) = 10^6)$. The rover has width 6 units and length 3 units. Though OUM-BD and OUM computed visually identical paths, OUM-BD was faster. The genetic algorithm appears stuck in a local minimum, unable to find a path with lower cost on the other side of the obstacles.



(a) Example 1: 5359 of 16765 Vertices (68% savings)



(b) Example 2: 12410 of 16765 Vertices (26% savings)



(c) Example 3: 11401 of 16765 Vertices (32% savings)

Figure 6.7: Bi-directional Fronts: The vertices in OUM-BD labelled *Accepted* in each example with the approximate optimal path found. The front found from $\mathbf{x}_f = (-450, 450)$, and the front found from $\mathbf{x}_0 = (450, -450)$ are both shown. The point at which the fronts met is labelled with a white X. The unmarked areas show the computation that was saved.

direction-dependent weight. The problem was solved on a workspace of $\overline{\Omega} = [-500, 500] \times [-500, 500]$, with $\partial\Omega = \{\mathbf{x}_f\}$ and $\mathbf{x}_f = (-450, 450)$. The weight used was

$$g(\mathbf{x}, \mathbf{u}) = \sqrt{1 + (\nabla z(\mathbf{x}) \cdot \mathbf{u})^2}$$

where $z : \mathbb{R}^2 \to \mathbb{R}$ is defined

$$z(x, y) = 900 \sin\left(\frac{\pi}{500}x\right) \sin\left(\frac{\pi}{500}y\right),$$

and $q(\mathbf{x}_f) = 0$.

The BFM and FSM algorithms presented in Chapter 4 compute approximated solutions on uniform square grids, while OUM and MAOUM provide approximated solutions on simplicial meshes. For each simplicial mesh used for OUM and MAOUM, a square grid with similar properties was used for BFM and FSM. See Table 6.3. For a simplicial mesh made from diagonals of a square grid, the maximum edge length is $h_{max} = \sqrt{2}\triangle x$ and minimum simplex height is $h_{min} = \frac{\triangle x}{\sqrt{2}}$, the simplicial meshes and square grids have similar h_{max} values, number of vertices and $M = \frac{h_{max}}{h_{min}}$. In MAOUM, the time taken to execute the algorithm is presented without the time required to compute the stencil $\mathcal{M}(\mathbf{x}_i)$. To solve problems where the weight is dependent on direction using FSM, the update is changed to (4.13) using neighbours on the grid. The termination condition on the error between iterations for FSM was chosen to be

$$|\widetilde{V} - \widetilde{V}_{old}|_\infty < \epsilon = 10^{-14}.$$

The timings and number of updates required to obtain $\widetilde{V}$ for each algorithm are presented in Tables 6.4 and 6.5 respectively. The time required by all algorithms were comparable despite the additional computation required by OUM and MAOUM to process the simplicial meshes. The OUM and MAOUM also required significantly fewer updates than BFM and FSM to compute a solution on a discretization with similar properties.

	Mesh Vertices	Gridpoint Vertices	Mesh h_{max}	Grid $\sqrt{2}\triangle x$	Mesh M
Discretization 1	1121	1089	44.19	44.19	2.50
Discretization 2	4289	4225	24.07	22.09	2.77
Discretization 3	16765	16900	11.99	10.96	3.08
Discretization 4	33147	33124	8.761	7.86	3.36

Table 6.3: Discretizations of $\overline{\Omega}$ - The parameter $M = \frac{h_{max}}{h_{min}}$ provides a measure of consistency in size of the elements in the discretization. A simplicial mesh made of square elements with side length $\triangle x$ split on the diagonals will have a value of $M = 2$ and $h_{max} = \sqrt{2}\triangle x$. The parameters M, h_{max} and h_{min} of the simplicial meshes are shown for comparison.

	OUM	MAOUM	BFM	FSM
Discretization 1	27.13	4.701	30.33	55.91
Discretization 2	70.38	29.38	64.39	69.05
Discretization 3	244.29	242.8	279.4	232.6
Discretization 4	518.27	807.9	932.9	492.7

Table 6.4: Timings of Algorithms in Seconds. The timings between the algorithms are of the same order of magnitude, despite OUM and MAOUM being solved on a 2-simplicial mesh. In BFM and FSM, the majority of the time is spent calculating the updates. In OUM and MAOUM, fewer updates are required, but additional computation is required to determine which updates to perform.

	OUM (4.14)	MAOUM (4.14)	BFM (4.23)	FSM (4.14)
Discretization 1	29251	41573	196672	126324
Discretization 2	122721	192023	598878	304200
Discretization 3	461608	800856	2255836	1149200
Discretization 4	946560	1621924	4360206	2252432

Table 6.5: Update count for Algorithms. Fewer updates are required for the OUM and MAOUM, since the dependencies of the solution at vertices are in some sense decoupled. Only a narrow band of points are solved at a time in both algorithms. For iterative algorithms such as BFM and FSM, additional updates are required as the algorithms do not use information regarding possible directions of the characteristic, resulting in additional computation.

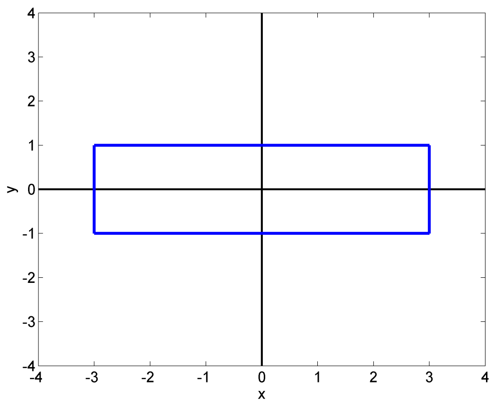


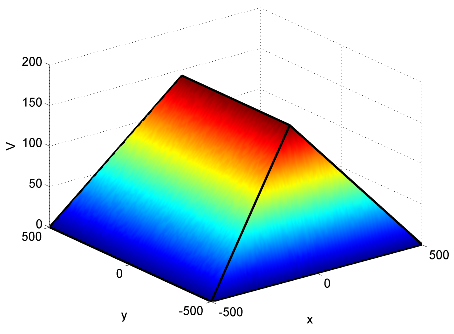
Figure 6.8: Rectangular speed profile $\mathcal{U}_g(\mathbf{x})$ with length 6 in the x-direction and 2 in the y-direction. The distance (radius) from $\mathbf{x}$ to the speed profile for a given angle θ in polar coordinates corresponds to the value $\frac{1}{g(\mathbf{x},\mathbf{u})}$ for the direction $\mathbf{u} = (\cos\theta, \sin\theta)$.

6.4 Numerical Convergence of OUM

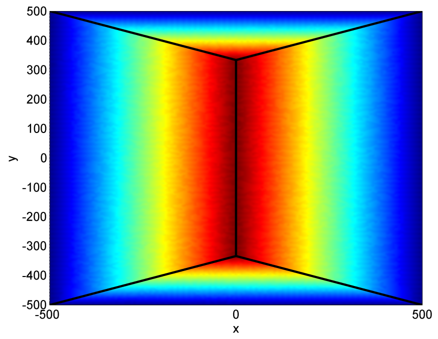
In Chapter 5, a result was proven that the OUM algorithm for boundary value problems must converge at a rate of at least $\mathcal{O}(\sqrt{h_{max}})$. An example of the error computed using OUM for the boundary value problem is shown below. For $\overline{\Omega} = [-500, 500] \times [-500, 500]$, $\partial\Omega = \{(x, y) \in \overline{\Omega} | |x| = 500 \text{ or } |y| = 500\}$, the weight g used corresponded to a rectangular speed profile centered about $\mathbf{x}$ with dimensions 6 in the x-direction and 2 in the y-direction. See Figure 6.8. The boundary function was $q(\mathbf{x}) = 0$ for $\mathbf{x} \in \partial\Omega$. Recall that for a direction $\mathbf{u}$, the radius of the speed profile $\mathcal{U}_g(\mathbf{x})$ (3.11) corresponds to $\frac{1}{g(\mathbf{x},\mathbf{u})}$. The same weight was used for all $\mathbf{x} \in \Omega$.

For this simple problem, the analytic solution is made up of the concatenation of 4 planes: $y + z = 500$, $x + 3z = 500$, $-y + z = 500$ and $-x + 3z = 500$ within Ω. See Figure 6.9a. Black lines have been used to indicate where the planes are defined. On all of these lines, the gradient ∇V is not defined. The plot of the error for one of the discretizations is presented in Figure 6.9c. The error plot for all discretiztaions had the same general shape, with the error appearing only near areas where ∇V was not defined.

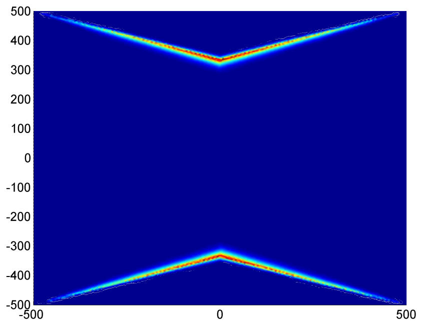
The error values are given in Table 6.6 and a plot is provided in Figure 6.10. The numerical rate of convergence k_p was measured using the L^p norm for $p = 1$ and $p = \infty$.



(a) The exact solution V: a three-dimensional view.



(b) The exact solution V: a two-dimensional (birds-eye) view.



(c) The error between the approximated solution $\widetilde{V}$ and true solution V is greatest at points where ∇V is not defined.

Figure 6.9: The exact solution V for the static HJB problem with rectangular speed profile is the concatenation of four planes. In (a) and (b), the separation between the planes is marked by black lines. The gradient ∇V is not defined on those lines.

Vertices	Triangles	h_{max}	L^1-Error	L^∞-Error
4289	8256	24.07	3.93×10^5	10.54
16765	32888	11.99	1.95×10^5	7.45
33147	65652	8.761	1.17×10^5	5.90
66291	131300	6.438	9.85×10^4	5.36
263597	524632	3.483	5.02×10^4	3.80
1051261	2097400	1.785	2.56×10^4	2.74

Table 6.6: Accuracy of OUM for a Boundary Value Problem - The OUM was used to solve the static HJB problem with a rectangular profile on six meshes. Each was measured against the exact solution.

The error formula

$$\left\|\widetilde{V} - V\right\|_{L^p} = Ch_{max}^k,$$

where V is the true solution and $\widetilde{V}$ is the approximated solution computed using OUM, was used. Hence

$$\ln\left\|\widetilde{V} - V\right\|_{L^p} = k\ln(h_{max}) + \ln(C).$$

For $p = 1$, the error was measured by finding the error at vertices, then the volume double integral was computed to obtain the L^1-norm. This was easy since V was made of four planes. For the L^∞-norm, the maximum error over all vertices was found. Using `polyfit` in MATLAB for the data points above, rates of convergence of $k_1 = 1.050$ and $k_\infty = 0.519$ were obtained for L^1-error and L^∞-error respectively. Numerical evidence in [72] suggests that for some examples, the convergence rate of OUM can occur at a faster rate of $\mathcal{O}(h_{max})$.

6.5 Conclusion

The implementation of OUM was discussed, including two suggested improvements in [72]. As well, the update from incorrect simplices as a result of nonconvex workspace Ω and obstacles were removed in the building of the *Near Front*.

The optimal path planning problem was solved considering rover objectives in the weight function. The FMM, FMM-BD, OUM and OUM-BD were compared against a genetic algorithm path planner [28] in both timing and performance. Using the approximated value function $\widetilde{V}$, the optimal path was found using (3.8). The path planning method suggested in [28] did not find a feasible path in a cluttered environment. As expected, the

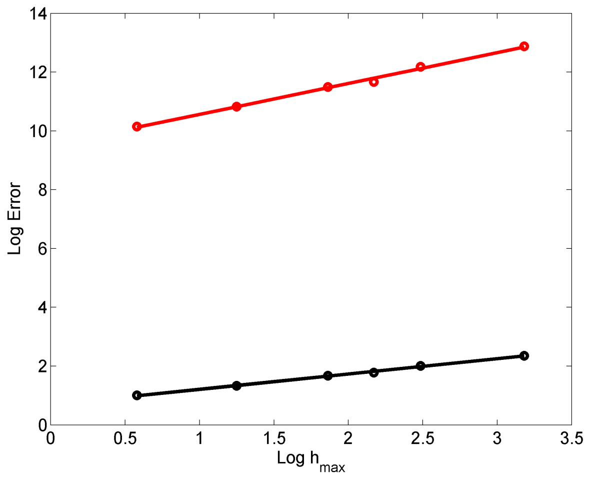


Figure 6.10: L^1 and L^∞-error for OUM Convergence Example - L^1-error shown in red (above), L^∞-error shown in black (below). The numerical convergence rates measured for L^1 and L^∞-error were $k_1 = 1.050$ and $k_\infty = 0.519$ respectively.

computational time of OUM and OUM-BD were slower than FMM and FMM-BD respectively to obtain paths with approximately the same cost. Significantly more time was required by the genetic algorithm path planner to obtain a path of comparable cost in the other examples.

Four algorithms were presented in Chapter 4 that provide an approximation of V in (3.5) for direction-dependent weights. The time and number of updates required for varying mesh refinements were shown. The timings in all four algorithms were comparable, despite the additional computation required to perform calculations on a simplicial mesh in OUM and MAOUM. It should be noted that further improvements in timing could be achieved by reprogramming all algorithms in `C`.

The convergence rate for OUM was numerically tested on a problem with a homogeneous weight defined using a rectangular speed profile. The L_∞-error converged at a rate of $\mathcal{O}(h_{max}^{1/2})$. The numerical results confirmed the convergence rate proved in Chapter 5. In numerical tests shown in [5, 72], the convergence occured at a faster rate for some problems. It would be interesting to develop conditions or determine a class of problems for which a faster convergence rate could be achieved.

Chapter 7

Conclusions and Future Work

Optimal path planning problems were solved approximately on discretizations of a workspace. Two novel improvements to the modelling of rover optimal path planning were considered: solar energy input and tip-over stability risk. Safe paths are important in rover path planning because of the high costs incurred in unmanned extraterrestrial exploration. The safest path was shown to be significantly different from the shortest path. Another area of future investigation is to allow for time-dependence in the weight function as done in [84]. As a rover traverses the environment, the urgency to recharge the solar panels becomes essential. Including a moving sun $(x_s(t), y_s(t), z_s(t))$ in the solar energy weight is realistic as rover speeds are slow and paths are planned across large environments. Negative weights would allow for more accuracy in the modelling of energy such as in energy regeneration from downhill travel. Of the algorithms programmed in Chapter 6, only the genetic algorithm can handle weights where some directions are negative. The discussed algorithms that approximately solve the static HJB require weights cannot handle negative weights. Integral constraints that limit the fuel used could also be implemented using an extension of FMM or OUM such as in [50, 59].

The work on path planning in this thesis focused upon finding a path, rather than finding the controls for a specific vehicle to follow the path. As such, the only restriction on control given to the path planning problem was that it be measurable. For rover path planning, where the rover has slow maximum speed and and can turn in place, controls can be found to follow such a path. For vehicles where this is not reasonable, additional constraints can be imposed. One method would be to use a more realistic model for dynamics in the path planning problem (3.1). A constraint on turning radius was considered in [78] where an additional dimension was given to the angle of the car.

Raising the dimensionality of a problem results in a significant increase in computation of approximated solutions.

A variety of different algorithms that found an approximation to the dynamic programming principle (3.5) were programmed and compared in timing. The FMM can be used to solve an approximation of the HJB equation for positive, continuous, direction-independent weights. The solution found by OUM yielded approximately the same path, but was slightly slower than FMM. Thus, for problems where either approach could be used, FMM is preferable. A novel bi-directional search for OUM, OUM-BD specific to path planning was presented. Depending on the problem, computational savings of 25% to 70% were observed in terms of finalized vertices on a mesh. The OUM-BD search algorithm could be implemented using parallel processors which would increase its efficiency. The OUM-BD was approximately 100 times faster than a rover path planner [28] that used genetic algorithm on the problems tested and the path produced yielded better results according to the cost function. The GA is capable of handling more general weights, which include negative weights. However, the faster computational times with FMM and OUM would allow for re-planning, therefore allowing the algorithm to be executed with real-time responses towards moving obstacles, and a moving sun. The computational times could be further reduced with coding in `C`. Investigation in the use of heuristics such as in A^* [35] to guide the search in conjunction with OUM-BD would reduce the domain on which the solution $\widetilde{V}$ was required even further. Unfortunately, consistent and admissible heuristics may not be easily obtained for OUM. Depending on the frequency of calculation, OUM-BD could be programmed into a FPGA and used in real-time path planning.

For FMM, and OUM, a proof of convergence of the approximated value function $\widetilde{V}$ to the value function V as h_{max} approaches 0 can be found in the literature. A proof of convergence for $\nabla\widetilde{V}$ to ∇V, which is used in path planning (3.8) remains an open problem. In practice, the approximation of $\nabla\widetilde{V}$ yields good results when used for path planning, as was shown in the path planning simulations in Chapter 6.

Local minima are known to exist in some navigation functions such as the artificial potential function. At locations that are local minima but not global minima, the numerical path finder (3.8) would become stuck. The value function used in optimal path planning was shown to not have any local minima (aside from a global minimum).

A performance comparison was made between four algorithms that could find solutions to the static HJB equation. Both timing and the number of updates required to produce the approximate value function were compared. As the BFM and FSM algorithms have iterative components, the OUM and MAOUM required significantly fewer updates. However, for OUM and MAOUM, computational effort and storage were required to determine which updates to compute. As well, the OUM and MAOUM were executed on simplicial meshes, which also increased the amount of storage and computation was required. Simplicial meshes have the benefit of handling complex geometries. From the implementations used, OUM and FSM are the leading candidates with regard to speed. If OUM was programmed to make use of a grid geometry for grids and grid-like meshes, it would become even faster.

The rate of convergence of the approximate solution provided by OUM to the viscosity solution for a problem with prescribed boundary values on a region was proven to be at least $\mathcal{O}(\sqrt{h_{max}})$. The basic idea of the proof was an extension of a result in [61]. A key step was to show the existence of a directionally complete stencil, discussed in [5]. This led to a proof that the numerical Hamiltonian for the OUM was both consistent and monotonic. One extension of this work would be to provide a convergence rate proof for the single-source point formulation of the problem. This is complicated by the lack of an obvious directionally complete stencil that would yield a consistent solution over vertices near the source point. The source-point formulation admits additional error near the source point [29, 79] due to a lack of continuous directions to perform the update. The numerical error is usually measured away from the source point. It is likely that the technique used here can be applied to the source-point problem for vertices that are far enough away from the source point.

Numerical evidence illustrated in Chapter 6 indicates that convergence of the approximate solution computed by the OUM occurs at a rate $\mathcal{O}(\sqrt{h_{max}})$. It would be interesting to establish conditions or determine a class of problems under which OUM converges with rate $\mathcal{O}(h_{max})$ as described numerically in [72]. It is believed that a different technique than used here is required.

References

[1] S. Ahmed, S. Bak, J. McLaughlin, and D. Renzi. A third order accurate fast marching method for the Eikonal equation in two dimensions. *SIAM J. Sci. Comput.*, 33:2402–2420, 2011.

[2] I. Al-Taharwa, A. Sheta, and M. Al-Weshah. A mobile robot path planning using genetic algorithm in static environment. *Journal of Computer Science*, 4(4):341–344, 2008.

[3] K. Alton. *Dijkstra-like Ordered Upwind Methods for Solving Static Hamilton-Jacobi Equations*. PhD thesis, University of British Columbia, 2010.

[4] K. Alton and I. Mitchell. Fast marching methods for stationary Hamilton-Jacobi equations with axis-aligned anisotropy. *SIAM J. Numer. Anal.*, 47:363–385, 2008.

[5] K. Alton and I. Mitchell. An ordered upwind method with precomputed stencil and monotone node acceptance for solving static convex Hamilton-Jacobi equations. *Journal of Scientific Computing*, 51:313–348, 2012.

[6] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *Int. J. Vehicle Autonomous Systems*, 8:190–216, 2010.

[7] T. Angell. Notes on convex sets. http://www.math.udel.edu/

[8] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhäuser, 1997.

[9] M. Bardi and M. Falcone. Discrete approximation of the minimal time function for systems with regular optimal trajectories. *Analysis and Optimation of Systems: Lecture Notes in Control and Information Sciences*, 144:103–112, 1990.

[10] J. Bellingham, A. Richards, and J. How. Receding horizon control of autonomous aerial vehicles. In *American Control Conference*, 2002.

[11] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *Proceedings of the American Control Conference*, 2006, pages 2831–2837, 2006.

[12] L. Blackmore, M. Ono, and B.C. Williams. Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27:1080–1094, 2011.

[13] J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. Springer, 2005.

[14] A. Bressan. Viscosity solutions of Hamilton-Jacobi equations and optimal control problems. https://www.math.psu.edu/bressan/PSPDF/hj.pdf, June 2011.

[15] Z. Clawson, A. Chacon, and A. Vladimirsky. Causal domain restriction for Eikonal equations. *Submitted to SIAM J. of Scientific Computing*, 2013.

[16] C. I. Connolly, J. B. Burns, and R. Weiss. Path planning using Laplace's equation. *Proc. IEEE Int. Conf. Robotics Automat.*, May 13-18:2102–2106, 1990.

[17] M. G. Crandall and P. L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277:1–42, 1983.

[18] M.G. Crandall, L.C. Evans, and P.L. Lions. Some properties of viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 282:487–502, 1984.

[19] M.G. Crandall and P.L. Lions. Two approximations of solutions of Hamilton-Jacobi equations. *Mathematics of Computation*, 43:1–19, 1984.

[20] E. Cristiani. A fast marching method for Hamilton-Jacobi equations modeling monotone front propagation. *J Sci Comput*, 39:189–205, 2009.

[21] E. Cristiani and M. Falcone. Fast semi-Lagrangian schemes for the Eikonal equation and applications. *SIAM J. Numer. Anal.*, 45:1979–2011, 2007.

[22] K. Deckelnick and C. Elliott. Uniqueness and error analysis for Hamilton-Jacobi equations with discontinuities. *Interfaces and Free Boundaries*, 6:329–349, 2004.

[23] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[24] F.G. Ding, J. Peng, X.Q. Bian, and H.J. Wang. AUV local path planning based on virtual potential field. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, 2005.

[25] D. Engwirda. MESH2D - Automatic mesh generation. MATLAB Central, 2011.

[26] L.C. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 2nd edition, 2010.

[27] M. Falcone and R. Ferretti. Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations. *Numer. Math*, 67:315–344, 1994.

[28] S. Fallah, B. Yue, O. Vahid-Araghi, and A. Khajepour. Energy management of planetary rovers using a fast feature-based path planning and hardware-in-the-loop experiments. *IEEE Transactions on Vehicular Technology*, 62:2389 – 2401, 2013.

[29] S. Fomel, S. Luo, and H. Zhao. Fast sweeping method for the factored Eikonal equation. *Journal of Computational Physics*, 228:6440–6455, 2009.

[30] E. Frew. Combining area patrol, perimeter surveillance, and target tracking using ordered upwind methods. In *IEEE International Conference on Robotics and Automation, Kobe, Japan*, pages 3123–3128, 2009.

[31] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice - a survey. *Automatica*, 25:335–348, 1989.

[32] M.R. Garcia, C. Vilas, L.O. Santos, and A.A. Alonso. A robust multi-model predictive control for distributed parameter systems. *Journal of Process Control*, 22:60–71, 2012.

[33] S. Garrido, L. Moreno, D. Blanco, and F. Monar. Robotic motion using harmonic functions and finite elements. *J Intell Robot Syst*, 59:57–73, 2010.

[34] R. Gonzalez and E. Rofman. On deterministic control problems: An approximation procedure for the optimal cost I. the stationary problem. *SIAM J. Control and Optimization*, 23:242–266, 1985.

[35] P.E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC4 4*, 2:100–107, 1968.

[36] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms, 2nd Edition.* Wiley Publishing, 2004.

[37] S. Hayati, R. Volpe, P. Backes, J. Balaram, R. Welch, R. Ivlev, G. Tharp, S. Peters, T. Ohm, R. Petras, and S. Laubach. The Rocky 7 rover: A Mars sciencecraft prototype. *Proceedings - IEEE International Conference on Robotics and Automation*, 3:2458–2464, 1997.

[38] O. Hjelle and A. Petersen. A Hamilton-Jacobi framework for modeling folds in structural geology. *Math Geosci*, 43:741–761, 2011.

[39] L. Janson and M. Pavone. Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions. *Preprint; Arxiv: cs.RO, 1306.3532.*, 2013.

[40] T. A. Johansen. *Selected Topics on Constrained and nonlinear control: Introduction to Nonlinear Model predictive control and moving horizon estimation.* STU Bratislava and NTNU Trondheim, 2011.

[41] C. Kao, S. Osher, and Y. Tsai. Fast sweeping methods for static Hamilton-Jacobi equations. *SIAM Journal on Numerical Analysis*, 42:2612–2632, 2005.

[42] C.Y. Kao, S. Osher, and J. Qian. Legendre-transform-based fast sweeping methods for static HamiltonJacobi equations on triangulated meshes. *Journal of Computational Physics*, 227:10209–10225, 2008.

[43] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30:846–894, 2011.

[44] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[45] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[46] R. Kimmel and J. Sethian. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision*, 14:237–244, 2001.

[47] R. Kimmel and J.A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95:8431–8435, 1998.

[48] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. *Proceedings of the IEEE - International Conference on Robotics and Automation*, 2:1398–1404, 1991.

[49] J. Kuffner. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2000.

[50] A. Kumar and A. Vladimirsky. An efficient method for multiobjective optimal control and optimal control subject to integral constraints. *Journal of Computational Mathematics*, 28:517–551, 2010.

[51] S. LaValle. *Planning Algorithms*. Cambridge, 2006.

[52] S. LaValle and J. Kuffner. Randomized kinodynamic planning. *Int'l Journal of Robotics Research*, 5:473–479, 2001.

[53] J. Lee, Y. Nam, S. Hong, and W. Cho. New potential functions with random force algorithms using potential field method. *J Intell Robot Syst*, 66:303–319, 2012.

[54] M. Lepetic, G. Klancar, I. Skarjanc, D. Matko, and B. Potocnik. Time optimal path planning considering acceleration limits. *Robotics and Autonomous Systems*, 45:199–210, 2003.

[55] C. Ma and R. Miller. MILP optimal path planning for real-time applications. In *Proceedings of the 2006 American Control Conference*, pages 4945–4950, Minneapolis, Minnesota, June 2006.

[56] M. Maimone, J. Biesiadecki, E. Tunstel, Y. Cheng, and C. Leger. *Intelligence for Space Robotics: Surface navigation and mobility intelligence on the Mars Exploration Rovers*. TSI Press, 2006.

[57] D. Maravall and J. de Lope. Integration of artificial potential field theory and sensory-based search in autonomous navigation. In *IFAC, 15th Triennial World Congress, Barcelona, Spain*, 2002.

[58] J. Miro, G. Dumonteil, C. Beck, and G. Dissanayake. A kyno-dynamic metric to plan stable paths over uneven terrain. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 294 – 299, 2010.

[59] I. Mitchell and S. Sastry. Continuous path planning with multiple constraints. *IEEE Conference on Decision and Control*, 42:5502–5507, 2003.

[60] M. Mitchell. *An Introduction to Genetic Algorithm.* MIT Press, 1998.

[61] R. Monneau. Introduction to the Fast Marching Method. Technical report, Centre International de Mathématiques Pures et Appliqués, 2010.

[62] Y.S. Nam, B. H. Lee, and N. Y. Ko. An analytic approach to moving obstacle avoidance using an artificial potential field. In *Intelligent Robots and Systems 1005, RSJ International Conference*, 1995.

[63] E. Papadopoulos and D. Rey. A new measure of tipover stability margin for mobile manipulators. *Proc. IEEE Int. Conf. Robotics and Automation*, 4:3111–3116, 1996.

[64] C. Pêtrès, Y. Pailhas, P. Patrón, Y. Petillot, J. Evans, and D. Lane. Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23:331 – 341, 2007.

[65] G. Peyré. Dijkstra and fast marching algorithms. http://www.ceremade.dauphine.fr/~peyre/numerical-tour/tours/fastmarching_0_implementing/, 2011.

[66] I. Pohl. Bi-directional search. *Machine Intelligence*, 6:127–140, 1971.

[67] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, 2007.

[68] K. Rasheed, H. Hirsh, and A. Gelsey. A genetic algorithm for continuous design space search. *Artificial Intelligence in Engineering*, 11:295–305, 1997.

[69] E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, Oct 1992.

[70] K. Sato. Deadlock-free motion planning using Laplace potential field. *Advanced Robotics*, 7:449–461, 1993.

[71] J. Sethian and A. Vladimirsky. Ordered upwind methods for hybrid control. *Hybrid Systems: Computation and Control, Lecture Notes in Computer Science*, 2289:393–406, 2002.

[72] J. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. *SIAM J. Numer. Anal.*, 41:325–363, 2003.

[73] J.A. Sethian. A marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93:1591–1595, 1996.

[74] J.A. Sethian. *Level Set Methods and Fast Marching Methods: Computer Vision and Material Sciences*. Cambridge University Press, 1999.

[75] A. Shum, K. Morris, and A. Khajepour. Convergence rate for the ordered upwind method. *In Preparation.*, 2014.

[76] A. Shum, K. Morris, and A. Khajepour. Direction-dependent path planning for autonomous rovers. *Submitted to Robotics and Autonomous Systems*, 2014.

[77] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. In *Robotics and Automation*, pages 3310–3317, 1994.

[78] R. Takei and R. Tsai. Optimal trajectories of curvature constrained motion in HamiltonJacobi formulation. *Journal of Scientific Computing*, 54:622–644, 2013.

[79] Y. Tsai, L. Cheng, S. Osher, and H. Zhao. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *SIAM J. Numer. Anal. 41*, 2:673–694, 2003.

[80] J. N. Tsitsiklis. Efficient algorithms for global optimal trajectories. *IEEE Transactions on Automatic Control*, 40:1528–1538, 1995.

[81] P. Vadakkepat, K.C. Tan, and W.M. Liang. Evolutionary artificial potential fields and their application in real time robot path planning problem. In *Proceedings of the 2000 Congress on Evolutionary Computation.*, volume 1, pages 256–263, 2000.

[82] J. Vanualailai, B. Sharma, and S. Nakagiri. An asymptotically stable collision-avoidance system. *International Journal of Non-Linear Mechanics*, 43:925–932, 2008.

[83] A. Vladimirsky. *Fast Methods for Static Hamilton-Jacobi Partial Differential Equations*. PhD thesis, University of Califoria, Berkeley, 2001.

[84] A. Vladimirsky. Static PDEs for time-dependent control problems. *Interfaces and Free Boundaries*, 8/3:281–300, 2006.

[85] T. Xiong, M. Zhang, Y.T. Zhang, and C.W. Shu. Fast sweeping fifth order WENO scheme for static Hamilton-Jacobi equations with accurate boundary treatment. *SIAM J. Sci. Comput.*, 45:514–536, 2010.

[86] B. Xu, A. Kurdila, and D. J. Stillwell. A hybrid receding horizon control method for path planning in uncertain environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[87] K. Yang, S. Gan, and S. Sukkarieh. An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments. *J Intell Robot Syst*, 57:101–122, 2010.

[88] Y. Yokokohji, S. Chaen, and T. Yoshikawa. Evaluation of traversability of wheeled mobile robots on uneven terrains by a fractal terrain model. *Advanced Robotics*, 21(1-2):121–142, 2007.

[89] H. Zhao. A fast sweeping method for Eikonal equations. *Mathematics of Computation*, 74:603–627, 2004.