J Sci Comput (2012) 51:313–348
DOI 10.1007/s10915-011-9512-4

An Ordered Upwind Method with Precomputed Stencil and Monotone Node Acceptance for Solving Static Convex Hamilton-Jacobi Equations

Ken Alton · Ian M. Mitchell

Received: 26 October 2009 / Revised: 24 June 2011 / Accepted: 28 June 2011 /
Published online: 29 July 2011

Abstract We define a δ-causal discretization of static convex Hamilton-Jacobi Partial Differential Equations (HJ PDEs) such that the solution value at a grid node is dependent only on solution values that are smaller by at least δ. We develop a Monotone Acceptance Ordered Upwind Method (MAOUM) that first determines a consistent, δ-causal stencil for each grid node and then solves the discrete equation in a single-pass through the nodes. MAOUM is suited to solving HJ PDEs efficiently on highly-nonuniform grids, since the stencil size adjusts to the level of grid refinement. MAOUM is a Dijkstra-like algorithm that computes the solution in increasing value order by using a heap to sort proposed node values. If $\delta > 0$, MAOUM can be converted to a Dial-like algorithm that sorts and accepts values using buckets of width δ. We present three hierarchical criteria for δ-causality of a node value update from a simplex of nodes in the stencil.

The asymptotic complexity of MAOUM is found to be $\mathcal{O}((\hat{\Psi}\rho)^d N \log N)$, where d is the dimension, $\hat{\Psi}$ is a measure of anisotropy in the equation, and ρ is a measure of the degree of nonuniformity in the grid. This complexity is a constant factor $(\hat{\Psi}\rho)^d$ greater than that of the Dijkstra-like Fast Marching Method, but MAOUM solves a much more general class of static HJ PDEs. Although ρ factors into the asymptotic complexity, experiments demonstrate that grid nonuniformity does not have a large effect on the computational cost of MAOUM in practice. Our experiments indicate that, due to the stencil initialization overhead, MAOUM performs similarly or slightly worse than the comparable Ordered Upwind Method presented in (Sethian and Vladimirsky, SIAM J. Numer. Anal. 41:323, 2003) for two examples on uniform meshes, but considerably better for an example with rectangular speed profile and significant grid refinement around nonsmooth parts of the solution. We

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

K. Alton · I.M. Mitchell (✉)
Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, Canada
e-mail: mitchell@cs.ubc.ca
url: http://www.cs.ubc.ca/~mitchell

K. Alton
e-mail: kalton@cs.ubc.ca

test MAOUM on a diverse set of examples, including seismic wavefront propagation and robotic navigation with wind and obstacles.

Keywords Ordered upwind methods · Anisotropic optimal control · Anisotropic front propagation · Hamilton-Jacobi equation · Viscosity solution · Dijkstra-like methods · Dial-like methods

1 Introduction

One interpretation of the viscosity solution to a static Hamilton-Jacobi Partial Differential Equation (HJ PDE) is the first arrival time of a propagating wavefront. A level contour of the solution is the position of the wavefront at a specific time. An intuitive and efficient method of approximating the solution on a grid is to compute the solution values at grid nodes in the order in which the wavefront passes through the grid nodes. The solution value at a particular grid node is based on values of neighboring grid nodes that are smaller, in the same way that the time at which the wavefront crosses any particular point is dependent on the earlier times the wavefront crosses nearby points in the direction from which it emanates. Dijkstra-like methods were developed in [23, 30] to approximate the solution to an isotropic static HJ PDE, also known as the Eikonal equation, in a single pass through the nodes of a grid in order of increasing solution value. Ordered Upwind Methods (OUMs) [26, 27] are an extension of these Dijkstra-like methods that approximate the solution to static convex HJ PDEs in a single pass.

One potential benefit of single-pass methods is that it may be possible to estimate error and refine the grid to achieve a desired solution accuracy as the solution is computed outward from the boundary condition. Because a node value is not computed until those node values on which it depends have already been computed, there is the potential to control error early before it has been propagated unnecessarily to dependent grid nodes. We are not aware of any existing adaptive single-pass methods for static HJ PDEs that refine the grid to control error on the fly. However, such an adaptive method would create significantly nonuniform grids for many problems. We develop a Dijkstra-like single-pass method which constructs causal stencils that extend beyond immediate neighbors but adjust to the local and directional grid spacing. Convergence can be shown using a well-known consistency, monotonicity, and stability proof [5]. A feature of our method is that the discretization of the HJ PDE is δ-causal (where $\delta \geq 0$ is a free parameter), such that the solution value at a grid node is dependent only on solution values that are smaller by more than δ.

We present Monotone Acceptance OUM (MAOUM), a two-pass Dijkstra-like method for which node values are accepted in nondecreasing order on a simplicial grid. We contrast MAOUM with the OUM introduced in [26, 27], which solves the same set of static convex HJ PDEs but does not necessarily accept node values monotonically. Because one of the defining features of their method is that a front of nodes with accepted values is maintained and the stencil is formed dynamically using only nodes from this front, we call their method Accepted Front OUM (AFOUM). MAOUM is a two-pass algorithm because the stencils for the discrete equation are precomputed in an initial pass through the nodes of the grid, as opposed to being computed in the same pass as the solution like in AFOUM. Because the stencil size and shape adjusts to local grid spacing, MAOUM is particularly suited to problems that benefit from nonuniform refinement in the grid (for an example, see Sect. 5.2). If $\delta > 0$, MAOUM can be modified to create a Dial-like method, for which nodes are sorted into buckets of width δ according to value and buckets of nodes are accepted in increasing value order [12, 30].

The rest of the introduction includes a definition of the problem, the computational grid, and the basic Dijkstra-like algorithm, as well as a discussion of related work. In Sect. 2 we present the discretized equation and the node value update equation, and we examine some of their useful properties. We develop three hierarchical tests for the δ-causality of a node value update from a simplex and prove their validity in Sect. 3. We use these tests to define MAOUM, verify that it solves the discretized equation, and analyze its asymptotic complexity in Sect. 4. Numerical examples are included in Sect. 5 to show empirically that the algorithm is convergent, is not much less efficient/accurate than AFOUM on uniform grids and is significantly more efficient/accurate than AFOUM for an example with appropriately chosen grid refinement. We also demonstrate that MAOUM can be used to solve practical problems, such as computing the first-arrival time for a seismic wavefront or finding optimal paths for a robot to reach a goal while fighting a strong wind and avoiding obstacles.

1.1 The Problem

The Dirichlet problem for a static HJ PDE is to find a function u such that

$$H(x, Du(x)) = 0, \qquad x \in \Omega, \tag{1.1a}$$

$$u(x) = g(x), \quad x \in \partial\Omega, \tag{1.1b}$$

where $Du(x)$ is the gradient of u at x, $\Omega \subset \mathbb{R}^d$ is a bounded Lipschitz domain, and $\partial\Omega$ is the domain's boundary.

An optimal continuous control problem which attempts to minimize the time to reach the boundary leads to the following Hamiltonian H in (1.1a) [27]:

$$H(x, p) = \max_{a \in \mathcal{A}} [(-p \cdot a) f(x, a)] - 1, \tag{1.2}$$

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

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

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

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

$$H(x, p) = \|p\|_2 f(x) - 1 \tag{1.3}$$

and (1.1) becomes the Eikonal equation. In an anisotropic problem, $f(x, a)$ depends on a for some x, i.e., $\mathcal{A}_f(x)$ is a closed non-spherical but convex set.

In general, it is impossible to find a classical solution u to the static Hamilton-Jacobi PDE (1.1) where u is differentiable for all x. We seek instead the viscosity solution, a unique weak solution which under the above conditions on $\mathcal{A}_f$ is continuous and almost everywhere differentiable [9].

1.2 Computational Grid

Since we typically cannot solve for the viscosity solution analytically, we compute an approximate solution $\underline{u}$ on a structured, semi-structured, or unstructured simplicial grid with nodes forming both a discretization $\underline{\Omega}$ of Ω, and a discretization $\underline{\partial\Omega}$ of $\partial\Omega$. Take $\underline{\Omega}$ and $\underline{\partial\Omega}$ to be disjoint sets and let $\mathcal{X} = \underline{\Omega} \cup \underline{\partial\Omega}$ be the set of all nodes in the grid. Let $\mathcal{N}(x)$ be the set of grid neighbors of node $x \in \mathcal{X}$. Let s be a simplex with n_s vertices, and x_i^s be the ith vertex of s where $1 \le i \le n_s$. Let $\underline{\mathcal{S}}$ be the set of grid simplices using neighboring grid nodes in $\mathcal{X}$. Define $\underline{\mathcal{S}}(\mathcal{R}) = \{s \in \underline{\mathcal{S}} \mid \text{for } 1 \le i \le n_s, x_i^s \in \mathcal{R} \subset \mathbb{R}^d\}$, the set of grid simplices using neighboring grid nodes in $\mathcal{R}$. We may specify the number of vertices using a subscript: for example, $\underline{\mathcal{S}}_d(\mathcal{R})$ is the set of grid simplices with d vertices. If left unspecified it is assumed $1 \le n_s \le d$, excluding simplices with $d+1$ vertices.

1.3 Dijkstra-like Methods

Algorithm 1 outlines a standard Dijkstra-like method [13] for solving isotropic static HJ PDEs on a simplicial grid, similar to the Fast Marching Method (FMM) [23]. MAOUM is a modification of this algorithm to handle HJ PDEs with convex anisotropy, while taking advantage of nonuniformity in the computational grid.

Informally, we refer to $\underline{v}(x)$ as the value of node x. When Algorithm 1 terminates, $\underline{v}$ is the approximate solution $\underline{u}$ to (1.1). The function call `Update`(y, s) returns a real number that replaces $\underline{v}(y)$ if it is less than the current value of $\underline{v}(y)$. In [23], the FMM algorithm employs an Eulerian finite-difference discretization to calculate `Update`(y, s). For MAOUM, we use the semi-Lagrangian discretization from [27] to calculate `Update`(y, s). This discretization generalizes that for Eikonal equations from [29] to anisotropic HJ PDEs. It is discussed in detail in Sect. 2 and the Appendix.

While the `Update` function in Algorithm 1 is determined by the underlying equation which we seek to solve, it is assumed that its execution time is independent of grid resolution and hence it does not affect the algorithm's asymptotic complexity. The `Update` functions in this paper maintain this property. Dijkstra-like algorithms are usually described as being $\mathcal{O}(N \log N)$, where $N = |\mathcal{X}|$ is the number of grid nodes. This complexity is derived by noting that each node is removed from $\mathcal{H}$ once and, in the usual binary min-heap implementation of $\mathcal{H}$, extraction of the minimum value node in line 1 costs $\mathcal{O}(\log |\mathcal{H}|) \le \mathcal{O}(\log N)$.

```
1  foreach x ∈ Ω̲ do v̲(x) ← ∞
2  foreach x ∈ ∂Ω̲ do v̲(x) ← g(x)
3  H ← X
4  while H ≠ ∅ do
5      x ← argmin_{y∈H} v̲(y)
6      H ← H \ {x}
7      foreach y ∈ N(x) ∩ H ∩ Ω̲ do
8          foreach s ∈ S̲(N(y) \ H) such that x ∈ s do
9              v̲(y) ← min(v̲(y), Update(y, s))
10         end
11     end
12 end
```

Algorithm 1: Standard Dijkstra-like method

For an efficient implementation a node $x \in \mathcal{H}$ need only be on the heap if $\underline{v}(x) < \infty$. Typically, the number of such nodes is much less than N.

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

1.4 Related Work

The first Dijkstra-like method for a first-order semi-Lagrangian discretization of the isotropic Eikonal PDE on an orthogonal grid was developed in [29]. The Dijkstra-like FMM was later independently developed in [23] for the first-order upwind Eulerian finite-difference discretization of the same Eikonal PDE. FMM was then extended to handle higher-order upwind discretizations on grids and unstructured grids in $\mathbb{R}^n$ and on manifolds [16, 24, 25]. By solving an isotropic problem on a manifold and then projecting the solution into a subspace, FMM can solve certain anisotropic problems [25]; for example, (1.2) with a constant elliptic speed profile $\mathcal{A}_f(x) = \mathcal{A}_f$ can be solved by running isotropic FMM on an appropriately tilted planar manifold and then projecting away one dimension. Some anisotropic etching problems have also been solved using FMM [19]. A class of axis-aligned anisotropic problems can be solved on a orthogonal grid using FMM [3].

AFOUM [26, 27] can solve general convex anisotropic problems on unstructured grids with an asymptotic complexity only a constant factor $\hat{\Upsilon}^{d-1}$ worse than FMM, where $\hat{\Upsilon}$ is a measure of anisotropy. FMM fails for these general problems because the neighboring simplex from which the characteristic approaches a node y may contain another node x such that causality does not hold: $\underline{v}(y) < \underline{v}(x)$. AFOUM avoids this difficulty by searching along the accepted front to find a set of neighboring nodes (which may not be direct neighbors of y) whose values have been accepted, and then constructing a virtual simplex with these nodes from which to update $\underline{v}(y)$.

Our method solves the same set of convex anisotropic problems on unstructured grids as AFOUM. MAOUM does not maintain an accepted front but must perform an initial pass to compute the stencils for each node and store them for the second pass that computes the solution. The benefit of this extra initial computation and storage is that MAOUM determines the stencil based on local grid spacing. This results in an algorithm that can take better advantage of local refinements in the grid to perform updates from a smaller stencil.

An alternative to these single-pass (or label-setting) algorithms are the sweeping (or label-correcting) algorithms, which are often even simpler to implement than FMM. Sweeping algorithms are also capable of handling anisotropic and even nonconvex problems. The simplest sweeping algorithm is to just iterate through the grid updating each node in a Gauss-Seidel (GS) fashion (so a new value for a node is used immediately in subsequent updates) until $\underline{v}$ converges. GS converges quickly if the node update order is aligned with the characteristics of the solution, so better sweeping algorithms [7, 11, 14, 21, 22, 28, 32] alternate among a collection of static node orderings so that all possible characteristic directions will align with at least one ordering. It is argued in [32] that these methods achieve $\mathcal{O}(N)$ asymptotic complexity (assuming that the node orderings are already determined); however, unlike

single-pass methods the number of sweeps necessary for convergence may depend on the problem.

There are also a number of sweeping algorithms which use dynamic node orderings; for example [4, 6, 20]. These algorithms attempt to approximate the optimal ordering generated by single-pass methods such as FMM without the overhead associated with managing an accurate queue. These methods have been demonstrated to be comparable to or better than single-pass methods for certain problems and grid resolutions. However, in general these methods may need to revisit nodes multiple times.

Sweeping methods have been used to solve static convex Hamilton-Jacobi equations on semi-structured grids [8] and unstructured grids [6, 22]. These methods have the advantage that the update of node value depends only on immediate neighbors, so they naturally take advantage of local refinement in the grid to compute accurate updates. However, the discretizations used lack an easy way of determining dependencies in the solution, which makes it difficult to know when the solution is computed correctly on a part of the domain before the algorithm has terminated. On the other hand, for single-pass methods like OUMs it is clear when the solution is computed correctly on a part of the domain and this might allow adaptive error estimation and grid refinement to be done as the solution is computed. For this reason, we develop an OUM algorithm that is suited to exploiting local refinement.

The algorithm described in [10] combines aspects of OUMs and sweeping methods. It computes solutions iteratively for a propagating band of grid nodes, but determines periodically which band nodes' values can no longer change and removes them from the band. This method is found to be more efficient than the Fast Sweeping method of [28, 32] for a problem with highly-curved characteristics but less efficient for a highly-anisotropic problem with straight characteristics. The asymptotic complexity of this method is not analyzed.

We believe that δ-Negative-Gradient-Acuteness, our first criterion for δ-causality of a node value update, is nearly equivalent to the criterion for the applicability of Dijkstra-like and Dial-like methods to anisotropic problems given in [31, Sect. 4]. However, δ-Negative-Gradient-Acuteness was derived independently and does not require differentiability of the cost function (2.11).

2 Discretization

We use the semi-Lagrangian discretization and notation from [27]. All norms are Euclidean unless otherwise stated. Let ζ be an n-vector barycentric coordinate such that

$$\sum_{i=1}^{n} \zeta_i = 1 \tag{2.1}$$

and $\zeta_i \geq 0$ for $1 \leq i \leq n$. Let Ξ_n be the set of all possible n-vector barycentric coordinates. State $\tilde{x}_s \in s$ can be parameterized by $\zeta \in \Xi_{n_s}$, that is, $\tilde{x}_s(\zeta) = \sum_{i=1}^{n_s} \zeta_i x_i^s$.

Let $x \in \Omega$ and define

$$\tau_s(x, \zeta) = \|\tilde{x}_s(\zeta) - x\| \tag{2.2}$$

and

$$a_s(x, \zeta) = \frac{\tilde{x}_s(\zeta) - x}{\tau_s(x, \zeta)}. \tag{2.3}$$

Restrict $x \notin s$ for all x so that $\tau_s(x, \zeta) > 0$. We may write $\tau_s(\zeta) = \tau_s(x, \zeta)$ and $a_s(\zeta) = a_s(x, \zeta)$ when x is clear from the context. We say that $a \in \mathcal{A}$ *intersects* s *from* x if the ray

$x + ta$, with $t > 0$, intersects s. Note that $a_s(\zeta)$ intersects s from x if and only if $\zeta_i \geq 0$ for $1 \leq i \leq n_s$, a condition imposed on ζ above.

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

$$\underline{H}(x, \mathcal{S}, \phi, \mu) = \max_{s \in \mathcal{S}} \max_{\zeta \in \Xi_{n_s}} \left\{ \frac{\mu - \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s)}{\tau_s(x, \zeta)} f(x, a_s(x, \zeta)) - 1 \right\}, \tag{2.4}$$

where $x \in \Omega$, $\mathcal{S}$ is a set of simplices s in $\Omega \cup \partial\Omega$ such that $x \notin s$ and vectors $x_i^s - x$ are independent, $\phi : \Omega \to \mathbb{R}$ is bounded, and $\mu \in \mathbb{R}$. When we are only varying μ, we write $\underline{H}(\mu) = \underline{H}(x, \mathcal{S}, \phi, \mu)$ for convenience.

The discretized Dirichlet problem is to find a function $\underline{u} : \mathcal{X} \to \mathbb{R}$, such that

$$\underline{H}(x, \mathcal{S}(x), \underline{u}, \underline{u}(x)) = 0, \qquad x \in \underline{\Omega}, \tag{2.5a}$$

$$\underline{u}(x) = g(x), \quad x \in \partial\underline{\Omega}, \tag{2.5b}$$

where $\mathcal{S}(x)$, *the update simplex set of* x, is chosen so that $\underline{H}$ is consistent and the solution to the discrete equation can be computed efficiently using a Dijkstra-like single-pass method. Note that in the definition of $\underline{H}$ in (2.4) we use the continuous domain Ω for the proof of consistency (Proposition 2.2), while in (2.5a) we restrict the domain to the discrete $\underline{\Omega}$, which poses no technical difficulties. Excluding the boundary condition (2.5b), we have a system of $|\underline{\Omega}|$ nonlinear equations of the form (2.5a), one for each node $x \in \underline{\Omega}$. We wish to compute the $|\underline{\Omega}|$ values $\underline{u}(x)$, one for each node $x \in \underline{\Omega}$.

Propositions 2.1 and 2.2 state that the numerical Hamiltonian $\underline{H}$ is monotone and consistent, which is important for the convergence of $\underline{u}$ to the viscosity solution u of the HJ PDE (1.1) as the grid spacing goes to zero [5]. A key property for monotonicity of $\underline{H}$ is that ζ_i are constrained to be nonnegative in (2.4). For consistency of $\underline{H}$, it is crucial that the simplices in $\mathcal{S}(x)$ encompass all possible directions $a \in \mathcal{A}$. Proposition 2.3 expresses μ in $\underline{H}(\mu) = 0$ explicitly, which is useful for constructing Algorithm 2 to solve (2.5). All propositions in this section are proved in the Appendix.

2.1 Monotonicity

$\underline{H}(x, \mathcal{S}, \phi, \mu)$ is monotone in the values $\phi(x_i^s)$, for $s \in \mathcal{S}$ and $1 \leq i \leq n_s$. Monotonicity of $\underline{H}$ requires that if none of $\phi(x_i^s)$ decrease, then $\underline{H}(x, \mathcal{S}, \phi, \mu)$ should not increase. Monotonicity of $\underline{H}$ comes naturally for the semi-Lagrangian form of (2.4). However, it is essential that $\zeta_i \geq 0$ for $1 \leq i \leq d$. A negative ζ_i can cause an increase in $\phi(x_i^s)$ to increase $\underline{H}(x, \mathcal{S}, \phi, \mu)$. This requirement is equivalent to making $a_s(\zeta)$ intersect s from x and makes the discrete equation (2.5) *upwinding* in the terminology of [27].

Proposition 2.1 *Let $x \in \underline{\Omega}$. Let $\check{\phi} : \Omega \to \mathbb{R}$ and $\hat{\phi} : \Omega \to \mathbb{R}$ be functions such that $\check{\phi}(x_i^s) \leq \hat{\phi}(x_i^s)$ for all $s \in \mathcal{S}$ and $1 \leq i \leq n_s$. Then $\underline{H}(x, \mathcal{S}, \check{\phi}, \mu) \geq \underline{H}(x, \mathcal{S}, \hat{\phi}, \mu)$.*

2.2 Consistency

For the numerical Hamiltonian $\underline{H}$ to be consistent with the Hamiltonian H from (1.2), the set of simplices $\mathcal{S}$ must encompass all possible action directions in $\mathcal{A}$.

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

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

Algorithm 2: Monotone Acceptance Ordered Upwind Method (MAOUM)

Proposition 2.2 *Let $\phi : \Omega \to \mathbb{R}$ be smooth, $x \in \Omega$, and $y \in \Omega$. Let $\mathcal{S}(y)$ be DC for y and such that for $s \in \mathcal{S}(y)$, we have $y \notin s$. Define $\hat{r}(y) = \max_{s\in\mathcal{S}(y),1\le i\le n_s} \|x_i^s - y\|$. Then*

$$\lim_{y\to x,\, \hat{r}(y)\to 0} \underline{H}(y, \mathcal{S}(y), \phi, \phi(y)) = H(x, D\phi(x)). \tag{2.6}$$

2.3 Unique Solution

There exists a unique solution $\mu = \tilde{\mu}$ to the equation $\underline{H}(\mu) = 0$. The value $\tilde{\mu}$ can be written explicitly in terms of the other quantities in $\underline{H}(\mu)$. This is a convenient form of the discretized equation for determining the solution $\underline{u}(x)$ at a node x in terms of the solution at its neighbors, as is done in the `Update` function of Algorithm 2. It is the same form as the semi-Lagrangian update in [27].

Proposition 2.3 *The unique solution to $\underline{H}(\mu) = 0$ with $\underline{H}$ defined by (2.4) is given by*

$$\mu = \tilde{\mu} = \min_{s\in\mathcal{S}} \min_{\zeta\in\Xi_{n_s}} \left\{ \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} + \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s) \right\}. \tag{2.7}$$

2.4 Convex Update Objective Function

Define the function $\eta_\phi^s : \Xi_{n_s} \to \mathbb{R}$, which is the objective function in (2.7):

$$\eta_\phi^s(\zeta) = \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} + \sum_{i=1}^{n_s} \zeta_i \phi(x_i^s). \tag{2.8}$$

Define the restriction of (2.7) to a single simplex s:

$$\tilde{\mu}_s = \min_{\zeta\in\Xi_{n_s}} \eta_\phi^s(\zeta). \tag{2.9}$$

The solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ is

$$\tilde{\mu} = \min_{s \in \mathcal{S}} \tilde{\mu}_s. \tag{2.10}$$

We show that η_ϕ^s is convex in ζ. The lemma is useful for proving the truth of Proposition 3.2, although this proof is not included here (see [1, Chap. 5] for the proof). Also, convexity is useful when applying numerical optimization algorithms to solve (2.7).

Lemma 2.4 *The function η_ϕ^s is convex.*

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

$$c(x, y) = \begin{cases} 0, & \text{if } y = 0, \\ \frac{\|y\|}{f(x, \frac{y}{\|y\|})}, & \text{otherwise.} \end{cases} \tag{2.11}$$

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

$$\begin{aligned} \eta_\phi^s(\zeta) &= \frac{\tau_s(\zeta)}{f(x, a_s(\zeta))} + \sum_{i=1}^d \zeta_i \phi(x_i^s) \\ &= c(x, \tau_s(\zeta) a_s(\zeta)) + \sum_{i=1}^d \zeta_i \phi(x_i^s) \\ &= c(x, \tilde{x}_s(\zeta) - x) + \sum_{i=1}^d \zeta_i \phi(x_i^s). \end{aligned} \tag{2.12}$$

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

3 Causality

Causality means that if the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ depends on value $\phi(x_i^s)$ then it must be that $\tilde{\mu} > \phi(x_i^s)$. Causality allows Dijkstra-like algorithms to be used to compute the solution to (2.5) in a single pass through the nodes $x \in \mathcal{X}$ in order of increasing value $\underline{u}(x)$. For isotropic problems, causality is satisfied if the direction vectors $(x_i^s - x)/\|x_i^s - x\|$ for $1 \le i \le n_s$, when considered pairwise, form non-obtuse angles [25]. Negative-gradient-acuteness (NGA) is a similar property for more general anisotropic problems.

Let $\delta \ge 0$. δ-causality means that if the solution $\mu = \tilde{\mu}$ to $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ depends on value $\phi(x_i^s)$ then it must be that $\mu > \phi(x_i^s) + \delta$. δ-causality allows Dial-like algorithms with buckets of width δ to be used to compute the solution to (2.5) in a single pass through the nodes $x \in \mathcal{X}$ in order of increasing bucket value.

In Sect. 3.1 we define what it means for a simplex to be δ-NGA and prove that it implies δ-causality of the discrete equation. In [31] a criterion for the δ-causality of (2.5) was presented. We believe δ-NGA is equivalent to the criterion in [31], if the function defined in (2.11) is differentiable everywhere but at the origin. In Sect. 3.2 we define another property

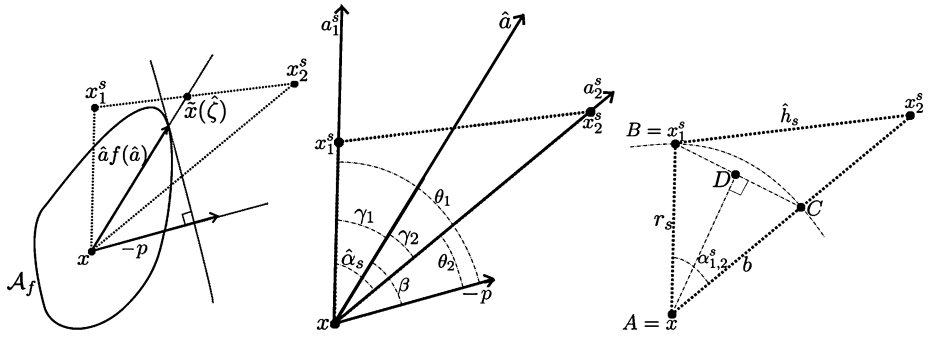

Fig. 1 Depiction of symbols used in the definition of (**a**) δ-NGA and the proof of Theorem 3.1 (**b**) δ-AAB and the proof of Theorem 3.3 (**c**) DRB and the proof of Lemma 3.4

on a simplex, δ-anisotropy-angle-boundedness (δ-AAB), and show that it implies δ-NGA. δ-AAB is easier to implement in Algorithm 2, while δ-NGA is more likely to have application beyond Algorithm 2. In Sect. 3.3 we define one last property on a simplex called distance-ratio-boundedness (DRB). When $\delta = 0$, DRB implies δ-AAB. DRB is used to prove the correctness of Algorithm 3 in Theorem 4.4. Note that 0-AAB and DRB are independent of specific speed profile $\mathcal{A}_f(x)$ as long as the degree of anisotropy remains the same.

The properties δ-NGA and δ-AAB allow for general $\delta \geq 0$ so that Algorithm 2 can be easily converted into a Dial-like method. However, we set $\delta = 0$ for the description and discussion of Dijkstra-like Algorithm 2 in Sect. 4 and we test only the Dijkstra-like algorithm in Sect. 5. The property DRB is not generalized to $\delta \geq 0$, because it is used to prove the correctness of just the Dijkstra-like algorithm. In the future we plan to implement and test a Dial-like version of the algorithm using δ-AAB.

3.1 Negative-Gradient-Acuteness

We show that if $\mathcal{S}$ is δ-negative-gradient-acute, the discrete equation $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ is δ-causal. Figure 1(a) is a geometric aid to understanding the definition of and proofs involving δ-negative-gradient-acuteness.

Definition Let $\check{\zeta}$ be a minimizer in (2.9). We say that (2.9) is *δ-causal for $x \in \Omega$* and $s \in \mathcal{S}$ if $\check{\zeta}_i > 0$ implies $\tilde{\mu}_s - \phi(x_i^s) > \delta$, for $1 \leq i \leq d$. When $\delta = 0$, we may say (2.9) is *causal for x* and s.

Definition We say that $s \in \mathcal{S}$ is *δ-negative-gradient-acute (δ-NGA) for $x \in \Omega$* if for all $p \in \mathbb{R}^d$ and $\hat{\zeta} \in \Xi_{n_s}$ such that

$$\max_{a \in \mathcal{A}} (-p \cdot a) f(x, a) = (-p \cdot a_s(\hat{\zeta})) f(x, a_s(\hat{\zeta})) = 1 \tag{3.1}$$

holds, we have $(x_i^s - x) \cdot (-p) > \delta$ for i such that $\hat{\zeta}_i > 0$. We say that $\mathcal{S}$ is *δ-NGA for x* if all $s \in \mathcal{S}$ are δ-NGA for x. When $\delta = 0$, we may say s (or $\mathcal{S}$) is *NGA for x*.

Theorem 3.1 *Let $s \in \mathcal{S}$ be δ-NGA for $x \in \Omega$ and let $n_s = d$. Let (2.9) hold. Let $\check{\zeta}$ be a minimizer in (2.9) such that $\check{\zeta}_i > 0$ for $1 \le i \le d$. Then (2.9) is δ-causal for x and s.*

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

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

It follows that for $1 \le i \le d$

$$\phi(x_i^s) = \tilde{\mu}_s + (x_i^s - x) \cdot p. \tag{3.2}$$

By (3.2), (2.1), and (2.3),

$$\begin{aligned} \sum_{i=1}^{d} \zeta_i \phi(x_i^s) &= \sum_{i=1}^{d} \zeta_i [\tilde{\mu}_s + (x_i^s - x) \cdot p] \\ &= \tilde{\mu}_s + \left(\sum_{i=1}^{d} \zeta_i x_i^s - x \right) \cdot p \\ &= \tilde{\mu}_s + \tau_s(\zeta) a_s(\zeta) \cdot p. \end{aligned} \tag{3.3}$$

By (3.3) and because $\check{\zeta}$ is a minimizer in (2.9),

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

On the other hand, by (2.9) and (3.3),

$$1 \ge \max_{\zeta \in \Xi_d} \left\{ \frac{\tilde{\mu}_s - \sum_{i=1}^{d} \zeta_i \phi(x_i^s)}{\tau_s(\zeta)} f(x, a_s(\zeta)) \right\} = \max_{\zeta \in \Xi_d} [(-p \cdot a_s(\zeta)) f(x, a_s(\zeta))].$$

So,

$$\check{\zeta} \in \operatorname*{argmax}_{\zeta \in \Xi_d} [(-p \cdot a_s(\zeta)) f(x, a_s(\zeta))].$$

Since $\check{\zeta}_i > 0$ for $1 \le i \le d$, $\check{\zeta}$ is strictly feasible. In other words, $\check{\zeta}$ is a local maximum of $(-p \cdot a_s(\zeta)) f(x, a_s(\zeta))$. Since $a_s(\zeta)$ is a continuous mapping, $a_s(\check{\zeta})$ is a local maximum of $(-p \cdot a) f(x, a)$. Because $\mathcal{A}_f(x)$ is convex, any local maximum $\hat{a}$ of $(-p \cdot a) f(x, a)$ must be a global maximum over all $a \in \mathcal{A}$. Thus, we have

$$\max_{a \in \mathcal{A}} (-p \cdot a) f(x, a) = (-p \cdot a_s(\check{\zeta})) f(x, a_s(\check{\zeta})) = 1.$$

This equation is just (3.1) with $\check{\zeta}$ in place of $\hat{\zeta}$. Since $s \in \mathcal{S}$ is δ-NGA for x and $\check{\zeta}_i > 0$ for $1 \le i \le d$, we have $(x_i^s - x) \cdot (-p) > \delta$ for $1 \le i \le d$. Therefore, by (3.2), for $1 \le i \le d$

$$\tilde{\mu}_s > \tilde{\mu}_s + (x_i^s - x) \cdot p + \delta = \phi(x_i^s) + \delta. \qquad \square$$

Proposition 3.2 *Let $s \in \mathcal{S}$ be δ-NGA for $x \in \Omega$. Let (2.9) hold and $\check{\zeta}$ be the minimizer in (2.9). Then (2.9) is δ-causal for x and s.*

Proposition 3.2 is slightly more general than Theorem 3.1 in that the simplex s may have $1 \leq n_s < d$ and it does not require $\check{\zeta}_i > 0$ for all i such that $1 \leq i \leq n_s$. For example, in 3 dimensions s could be an edge instead of a triangle, or $\tilde{x}_s(\zeta)$ could be on an edge of a triangular s. However, we can make Proposition 3.2 fit the form of the proof for Theorem 3.1 by making some adjustments. We do not include a formal proof because we believe the proof of Theorem 3.1 concisely illuminates the reason why δ-NGA implies δ-causality. The proof of Proposition 3.2 is more complicated but not much more explanatory and is included in [1, Chap. 5].

3.2 Anisotropy-Angle-Boundedness

A less general but more concrete way of ensuring causality of the discrete equation $\underline{H}(x, \mathcal{S}, \phi, \mu) = 0$ is to bound the maximum angle between any two direction vectors in $\mathcal{A}$ that intersect a simplex $s \in \mathcal{S}$ from x. We show that δ-AAB of a simplex s implies δ-NGA, which in turn implies δ-causality. δ-AAB is easy to implement, so we use it in Algorithm 2. Figure 1(b) is a geometric aid to understanding the definition of and proofs involving δ-anisotropy-angle-boundedness.

Define $\check{f}(x) = \min_{a\in\mathcal{A}} f(x,a)$ and $\hat{f}(x) = \max_{a\in\mathcal{A}} f(x,a)$. Let $\Upsilon(x) = \hat{f}(x)/\check{f}(x)$ be the local anisotropy coefficient. Note that $0 < \check{f}(x) \leq \hat{f}(x) < \infty$, so $1 \leq \Upsilon(x) < \infty$. Denote $a_i^s = (x_i^s - x)/\|x_i^s - x\|$. Define $\alpha_{i,j}^s$ to be the angle between a_i^s and a_j^s. Let $\hat{\alpha}_s = \max_{i,j} \alpha_{i,j}^s$. Let $r_s(x)$ be the minimum distance between x and any vertex of s:

$$r_s(x) = \min_i \|x_i^s - x\|. \tag{3.4}$$

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

$$\hat{\alpha}_s < \arccos(\delta \hat{f}(x)/r_s(x)) - \arccos(1/\Upsilon(x)). \tag{3.5}$$

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

Theorem 3.3 *If $s \in \mathcal{S}$ is δ-AAB for $x \in \Omega$, then s is δ-NGA for x.*

The following proof builds on analysis done in [27, Sect. 3.4] that bounds the angle between the optimal action and the negative gradient of the viscosity solution u of (1.1).

Proof Let $p \in \mathbb{R}^d$ and $\hat{a} \in \mathcal{A}$ be such that $\hat{a}$ intersects s from x and

$$\max_{a\in\mathcal{A}}(-p \cdot a) f(x,a) = (-p \cdot \hat{a}) f(x,\hat{a}) = 1.$$

We have

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

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

$$\cos(\beta) = \frac{-p \cdot \hat{a}}{\|-p\|\|\hat{a}\|} \geq \frac{\check{f}(x)}{f(x,\hat{a})} \geq \frac{1}{\Upsilon(x)}. \tag{3.6}$$

Because s is δ-AAB for x and by (3.6), $\beta \leq \arccos(1/\Upsilon(x))$, we have

$$\hat{\alpha}_s + \beta < \arccos(\delta \hat{f}(x)/r_s(x)). \tag{3.7}$$

Let $1 \leq i \leq n_s$. Let γ_i be the angle between $\hat{a}$ and a_i^s. Since $\hat{a}$ intersects s from x, we have $\gamma_i \leq \hat{\alpha}_s$. Let θ_i be the angle between $-p$ and a_i^s. By (3.7), we have

$$\theta_i \leq \gamma_i + \beta \leq \hat{\alpha}_s + \beta < \arccos(\delta \hat{f}(x)/r_s(x)). \tag{3.8}$$

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

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

and by (3.4), we have

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

By this inequality and (3.8), it follows that $\cos(\theta_i) > \delta \hat{f}(x)/r_s(x) \geq \delta/(\|x_i^s - x\|\|p\|)$. Consequently, we have $(x_i^s - x) \cdot (-p) = \|x_i^s - x\|\|-p\| \cos(\theta_i) > \delta$ for $1 \leq i \leq n_s$. Therefore, s is δ-NGA. □

Remark 1 We describe a way to consider the shape of the speed profile and not just the anisotropy coefficient $\Upsilon(x)$ when determining whether a simplex is δ-NGA. The upper bound on $\hat{\alpha}_s$ for the δ-AAB of a simplex s can often be increased. First we define some simplex specific notation. Let $\check{f}_s(x) = \min_{a \in \check{\mathcal{A}}_s} f(x,a)$, where

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

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

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

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

If it is possible to compute $\hat{f}_s(x)$ and $\check{f}_s(x)$, we can modify the definition of δ-AAB to use the loosened restrictions $\delta \hat{f}_s(x)/r_s(x) \leq 1$ and $\hat{\alpha}_s < \arccos(\delta \hat{f}_s(x)/r_s(x)) - \arccos(1/\Upsilon_s(x))$, and Theorem 3.3 still holds. This modification may result in more simplices satisfying the definition, which may allow us to find a DC and δ-NGA set of update simplices $\mathcal{S}(x)$ occupying a smaller region around x and thereby reduce the truncation error and computation cost.

The definition of δ-AAB can be simplified if we restrict the problem in several ways. If we take $\Upsilon(x) = 1$, (3.5) becomes $\hat{\alpha}_s < \arccos(\delta \hat{f}(x)/r_s(x))$ or, equivalently, $\cos(\hat{\alpha}_s) <$

$\delta \hat{f}(x)/r_s(x)$. This resembles a formula for the optimal bucket width δ for a Dial-like algorithm to solve the Eikonal equation derived in [31]. On the other hand, if we take $\delta = 0$, (3.5) becomes $\hat{\alpha}_s < \arcsin(1/\Upsilon(x))$. We use this condition in the Dijkstra-like Algorithm 2. Finally, if we take both $\Upsilon(x) = 1$ and $\delta = 0$, (3.5) becomes $\hat{\alpha}_s < \pi/2$. In the appendix of [27], it is shown that the slightly looser condition $\hat{\alpha}_s \leq \pi/2$ is sufficient for causality of (2.9).

3.3 Distance-ratio-boundedness

If the ratio of the minimum distance between x and any node in simplex s and the maximum distance between nodes in s is large enough then $s \in \mathcal{S}$ must be AAB for x. Proposition 3.6 provides a lower bound for this ratio that is sufficient for AAB. We use DRB in the proof of correctness of Algorithm 3 in the case when $\delta = 0$. We do not parameterize DRB by δ because it is difficult to determine a simple and tight lower bound on the ratio for general positive δ. Figure 1(c) is a geometric aid to understanding the definition of and proofs involving distance-ratio-boundedness.

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

$$\hat{h}_s = \max_{i,j} \|x_i^s - x_j^s\|. \tag{3.9}$$

Lemma 3.4 *Let $\hat{h}_s/(2r_s(x)) \leq 1$. The inequality $\hat{\alpha}_s \leq 2\arcsin(\hat{h}_s/(2r_s(x)))$ holds.*

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

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

By (3.4) we bound the length of either side below:

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

We split the isosceles triangle ABC in half to obtain a right-angle triangle with vertices $A = x$, $B = x + ba_i^s$, and $D = x + b(a_i^s + a_j^s)/2$. We have

$$\sin\left(\frac{\alpha_{i,j}^s}{2}\right) = \frac{\|B - D\|}{b} = \frac{\|B - C\|}{2b} \leq \frac{\hat{h}_s}{2r_s(x)}. \tag{3.10}$$

By the properties of the simplex s, $0 < \alpha_{i,j}^s/2 < \pi/2$. By (3.10), $\alpha_{i,j}^s \leq 2\arcsin(\hat{h}_s/(2r_s(x)))$ for any i and j. This implies that $\hat{\alpha}_s \leq 2\arcsin(\hat{h}_s/(2r_s(x)))$. □

Proposition 3.5 *Let $x \in \Omega$ and $s \in \mathcal{S}$. If $\delta \hat{f}(x)/r_s(x) \leq 1$ and*

$$2\arcsin(\hat{h}_s/(2r_s(x))) < \arccos(\delta \hat{f}(x)/r_s(x)) - \arccos(1/\Upsilon(x)) \tag{3.11}$$

then s is δ-AAB for x

Proof By (3.11), Lemma 3.4, and (3.5) in the definition of δ-AAB, s is δ-AAB for x. □

Equation (3.11) can be simplified if we restrict the problem in several ways. If we take $\Upsilon(x) = 1$, (3.11) becomes $2\arcsin(\hat{h}_s/(2r_s(x))) < \arccos(\delta \hat{f}(x)/r_s(x))$. On the other hand, if we take $\delta = 0$, (3.11) becomes $2\arcsin(\hat{h}_s/(2r_s(x))) < \arcsin(1/\Upsilon(x))$. From this condition, we can determine a lower bound $\Psi(x)$ on the ratio $r_s(x)/\hat{h}_s$:

$$\frac{r_s(x)}{\hat{h}_s} > \left[2\sin\left(\frac{\arcsin(1/\Upsilon(x))}{2}\right)\right]^{-1} = \Psi(x). \tag{3.12}$$

Finally, if we take both $\Upsilon(x) = 1$ and $\delta = 0$, (3.11) becomes $\hat{h}_s/r_s(x) < \sqrt{2}$, which implies $\hat{\alpha}_s < \pi/2$, the condition for AAB in this case.

Definition We say that $s \in \mathcal{S}$ is *distance-ratio-bounded (DRB) for* $x \in \Omega$ if $\delta = 0$ and (3.12) holds. We say that $\mathcal{S}$ is *DRB for* x if all $s \in \mathcal{S}$ are DRB for x.

Proposition 3.6 *If $s \in \mathcal{S}$ is DRB for $x \in \Omega$ then s is AAB for x.*

Proof By the definition of DRB, (3.12) holds, which is equivalent to (3.11), when $\delta = 0$. Since $\delta \hat{f}(x)/r_s(x) = 0 \leq 1$, by Proposition 3.5 s is AAB for x. □

Remark 2 If we simplify the definition of DRB by replacing $\Psi(x)$ with $\Upsilon(x)$ in (3.12), Proposition 3.6 still holds. However, $\Psi(x) < \Upsilon(x)$, so using $\Psi(x)$ to define DRB results in a looser restriction on simplices. For $1 \leq \Upsilon(x) < \infty$, since $\arcsin(1/\Upsilon(x)) > 2\arcsin(1/(2\Upsilon(x)))$, we have $\Psi(x) < \Upsilon(x)$. When $\Upsilon(x) = 1$, $\Psi(x) = 1/\sqrt{2}$ and $\lim_{\Upsilon(x)\to\infty}[\Psi(x)/\Upsilon(x)] = 1$. Finally, for $1 \leq \Upsilon(x) < \infty$, $\Psi(x)$ increases as $\Upsilon(x)$ increases.

4 Algorithm

In Algorithm 2 we define MAOUM. Algorithm 2 solves the discrete system (2.5). For (2.5) to be well-defined $\mathcal{S}(x)$ must be determined. The update simplex set $\mathcal{S}(x)$ is chosen to ensure $\underline{H}(x, \mathcal{S}(x), \phi, \mu)$ is consistent and the discrete equation $\underline{H}(x, \mathcal{S}(x), \phi, \mu) = 0$ is causal. Let

$$\mathcal{S}(x) = \{s \in \underline{\mathcal{S}}(\overrightarrow{\mathcal{Y}}(x)) \mid s \text{ is AAB for } x\}, \tag{4.1}$$

```
1  Q ← N(x)
2  while Q ≠ ∅ do
3      y ← Pop(Q)
4      Y→(x) ← Y→(x) ∪ {y}
5      Y←(y) ← Y←(y) ∪ {x}
6      foreach s ∈ S_d(Y→(x)) such that x ∉ s and y ∈ s do
7          if s not AAB for x then
8              Q ← Q ∪ (M(s) \ Y→(x))
9          end
10     end
11 end
```

Algorithm 3: `ComputeUpdateSet(x)`

Table 1 Summary of symbols. The *first group* is used in defining the numerical problem, the *second group* is used in Algorithm 2, the *third group* is used only in Algorithm 3, and the *fourth group* is used in the analysis of algorithm correctness and complexity

Symbol	Type or definition	Description
$\mathcal{X}$	$= \underline{\Omega} \cup \underline{\partial\Omega}$	set of all grid nodes
$\underline{\Omega}$	subset of $\mathcal{X}$	discretized domain
$\underline{\partial\Omega}$	subset of $\mathcal{X}$	discretized domain boundary
x	$\mathbb{R}^d$	domain point or grid node
$\mathcal{N}(x)$	$\mathcal{X} \to$ subset of $\mathcal{X}$	set of grid neighbors of node x
s	convex subset of $\mathbb{R}^d$	simplex
$\underline{\mathcal{S}}(\mathcal{R})$	set of simplices	grid simplices with all vertices in $\mathcal{R}$ and $1 \le n_s \le d$
$\overrightarrow{\mathcal{Y}}(x)$	$\mathcal{X} \to$ subset of $\mathcal{X}$	update node set of x, i.e., nodes upon which x might depend
$\mathcal{S}(x)$	$\mathcal{X} \to$ set of simplices	update simplex set of x
$\underline{H}(x, \mathcal{S}, \phi, \mu)$	see (2.4)	numerical (discrete) Hamiltonian
$\underline{u}(x)$	$\mathcal{X} \to \mathbb{R}$	solution of (2.5) at node x
$\underline{v}(x)$	$\mathcal{X} \to \mathbb{R}$	(temporary) value of node x
$\mathcal{H}$	subset of $\mathcal{X}$	min-value heap of unaccepted nodes
$\overleftarrow{\mathcal{Y}}(x)$	$\mathcal{X} \to$ subset of $\mathcal{X}$	dependent node set of x, i.e., nodes which might depend upon x
$\underline{\mathcal{S}}_d$	set of simplices	grid simplices with $n_s = d$
`Update`(x, s)	$\mathcal{X} \times \underline{\mathcal{S}} \to \mathbb{R}$	value update of node x from simplex s, i.e., $\tilde{\mu}_s$ from (2.9) with $\phi(x_i^s) = \underline{v}(x_i^s)$
$\mathcal{Q}$	subset of $\mathcal{X}$	queue to become update nodes
$\mathcal{M}(s)$	$\underline{\mathcal{S}} \to$ subset of $\mathcal{X}$	set of neighbors of all vertex nodes in s
$\mathcal{B}(x, r)$	convex subset of $\mathbb{R}^d$	ball of radius r centered on x
$\mathcal{B}_1(x)$	convex subset of $\mathbb{R}^d$	ball centered on x defined in (4.3)
$\mathcal{B}_2(x)$	convex subset of $\mathbb{R}^d$	ball centered on x defined in (4.4)
$\Psi(x)$	$\Omega \to \mathbb{R}$	a measure of anisotropy from (3.12)

where $\overrightarrow{\mathcal{Y}}(x)$ is the stencil or *update node set of* x. Let $\underline{v}(x)$ be the temporary value of node x. We update $\underline{v}(x)$ from simplex s using `Update`(x, s), which evaluates to the solution $\tilde{\mu}_s$ of (2.9), where $\phi(x_i^s) = \underline{v}(x_i^s)$. The dependent node set $\overleftarrow{\mathcal{Y}}(x)$ is the set of nodes that is dependent on x for their updates: $\overleftarrow{\mathcal{Y}}(x) = \{y \in \underline{\Omega} \mid x \in \overrightarrow{\mathcal{Y}}(y)\}$. Note that $\overrightarrow{\mathcal{Y}}(x)$ is not symmetric, meaning that it is usually not the case that $\overrightarrow{\mathcal{Y}}(x) = \overleftarrow{\mathcal{Y}}(x)$. In Algorithm 2, the node x is included in $\overrightarrow{\mathcal{Y}}(x)$ and $\overleftarrow{\mathcal{Y}}(x)$, which does not need to be true for the correctness of the algorithm, but it does not hurt and is done for convenience of proofs in Sect. 4.1.

In Sect. 4.1 we describe how the subroutine `ComputeUpdateSet` defined in Algorithm 3 determines $\overrightarrow{\mathcal{Y}}(x)$, such that $\mathcal{S}(x)$ satisfies DC and AAB for x. We note that Algorithm 3 as defined does not result in the smallest possible such set $\overrightarrow{\mathcal{Y}}(x)$. In Sect. 4.2 we explain how Algorithm 2 computes the solution $\underline{u}$ to (2.5) in a single pass (after initialization) over the nodes in $\mathcal{X}$. Because $\underline{H}(x, \mathcal{S}(x), \phi, \mu) = 0$ is causal, when Algorithm 2 terminates, $\underline{v}(x) = \underline{u}(x)$ for all $x \in \mathcal{X}$. We revisit the convergence of $\underline{u}$ to the solution u of the HJ PDE (1.1) as the grid spacing goes to zero in Sect. 4.3. In Sect. 4.4 we examine the

computational and storage complexity of MAOUM. Table 1 summarizes the symbols used in these algorithms and their analysis.

4.1 Computing the Update Set

The status of Algorithm 3 during different stages of computing $\overrightarrow{\mathcal{Y}}(x)$ is shown in Fig. 2. To achieve more accurate and efficient computations in locally-refined parts of the grid, we desire the maximum extent of the update node set, $\hat{r}(x) = \max_{y \in \overrightarrow{\mathcal{Y}}(x)} \|x - y\|$, to shrink towards zero as the local grid spacing goes to 0. Section 4.3 discusses why this property is also needed for convergence. By proving Theorem 4.4, we show that the subroutine call `ComputeUpdateSet`(x) given in Algorithm 3 terminates in a finite number of iterations with a bound on $\hat{r}(x)$ that varies linearly with the local grid resolution. We further show that a subset of $\underline{\mathcal{S}}(\overrightarrow{\mathcal{Y}}(x))$ is DC and AAB for x not too near the boundary $\underline{\partial\Omega}$. Corollary 4.5 states that for x not too near the boundary $\underline{\partial\Omega}$, $\mathcal{S}(x)$ defined in (4.1) is DC and NGA, which is sufficient for consistency and causality. First we define notation and prove some useful lemmas.

Let $\mathcal{Z} \subseteq \mathcal{X}$. The set $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$ contains the d-dimensional grid simplices with all vertex nodes in $\mathcal{Z}$. Define $\mathcal{U}_{\mathcal{Z}} = \bigcup_{s \in \underline{\mathcal{S}}_{d+1}(\mathcal{Z})}(s)$, which is the d-dimensional set covered by the simplices in $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$.

Define $\mathcal{F}_\partial(\mathcal{Z}) = \{s \in \underline{\mathcal{S}}_d(\mathcal{Z}) \mid \mathcal{M}(s) \setminus \mathcal{Z} \neq \emptyset \text{ or for all } j, x_j^s \in \underline{\partial\Omega}\}$. The set $\mathcal{F}_\partial(\mathcal{Z})$ contains the $(d-1)$-dimensional grid simplex faces with all vertex nodes in $\mathcal{Z}$, such that there is a neighbor of all vertex nodes which is not in $\mathcal{Z}$ or all vertex nodes are on the boundary of the grid. Define $\mathcal{U}_{\partial\mathcal{Z}} = \bigcup_{s \in \mathcal{F}_\partial(\mathcal{Z})}(s)$, which is the $(d-1)$-dimensional surface covered by the simplices in $\mathcal{F}_\partial(\mathcal{Z})$.

Lemma 4.1 *Let $\partial\mathcal{U}_{\mathcal{Z}}$ be the boundary of $\mathcal{U}_{\mathcal{Z}}$. Then $\partial\mathcal{U}_{\mathcal{Z}} \subseteq \mathcal{U}_{\partial\mathcal{Z}}$.*

Proof Since $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$ contains only grid simplices which do not overlap except at their boundary, $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$ is a partition of $\mathcal{U}_{\mathcal{Z}}$. It follows that any point $z \in \partial\mathcal{U}_{\mathcal{Z}}$ must be on a $(d-1)$-dimensional face of at least one d-dimensional simplex in $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$. Furthermore, there exists such a $(d-1)$-dimensional face $s \in \underline{\mathcal{S}}_d(\mathcal{Z})$ such that either $\mathcal{M}(s) \setminus \mathcal{Z} \neq \emptyset$ or for all $j, x_j^s \in \underline{\partial\Omega}$. If there did not then z would necessarily be in the interior of $\mathcal{U}_{\mathcal{Z}}$, contradicting the assumption that $z \in \partial\mathcal{U}_{\mathcal{Z}}$. By definition $s \in \mathcal{F}_\partial(\mathcal{Z})$. Thus, $z \in \mathcal{U}_{\partial\mathcal{Z}} = \bigcup_{s \in \mathcal{F}_\partial(\mathcal{Z})}(s)$. □

Lemma 4.2 *If $x \in \underline{\Omega} \setminus \underline{\partial\Omega}$ and $\mathcal{N}(x) \subseteq \mathcal{Z}$, then $\mathcal{F}_\partial(\mathcal{Z})$ is DC for x.*

Proof Let $x \in \underline{\Omega} \setminus \underline{\partial\Omega}$ and $\mathcal{N}(x) \subseteq \mathcal{Z}$. Then $\underline{\mathcal{S}}_{d+1}(\mathcal{Z})$ includes all d-dimensional simplices which have x as a vertex. Since $x \notin \underline{\partial\Omega}$, we have $x \in \mathcal{U}_{\mathcal{Z}} \setminus \partial\mathcal{U}_{\mathcal{Z}}$.

Any path $\xi : [0, 1] \to \Omega$, such that $\xi(0) = x$ and $\xi(1) \notin \mathcal{U}_{\mathcal{Z}}$ intersects $\partial\mathcal{U}_{\mathcal{Z}}$. In particular, any path ξ of the form $\xi(t) = x + tCa$, where $t \in [0, 1]$, $C \in \mathbb{R}^+$ is a constant, $a \in \mathcal{A}$, and $\xi(1) \notin \mathcal{U}_{\mathcal{Z}}$ intersects $\partial\mathcal{U}_{\mathcal{Z}}$. By Lemma 4.1, such a path also intersects $\mathcal{U}_{\partial\mathcal{Z}}$. Since this fact holds for any $a \in \mathcal{A}$ and $\mathcal{U}_{\partial\mathcal{Z}}$ is the union of simplices in $\mathcal{F}_\partial(\mathcal{Z})$, $\mathcal{F}_\partial(\mathcal{Z})$ is DC for x. □

Let

$$\hat{h} = \max\{\|y - z\| \mid y \in \mathcal{X} \text{ and } z \in \mathcal{N}(y)\} \tag{4.2}$$

be the maximum grid edge length for the entire grid. Let $\hat{h}(\mathcal{R}) = \max\{\|y - z\| \mid y \in \mathcal{R} \cap \mathcal{X} \text{ and } z \in \mathcal{N}(y)\}$ be the maximum length of grid edges with at least one end node in $\mathcal{R} \subset \mathbb{R}^d$. Let $\mathcal{B}(x, r)$ be a closed ball of radius r around point $x \in \mathbb{R}^d$. The following

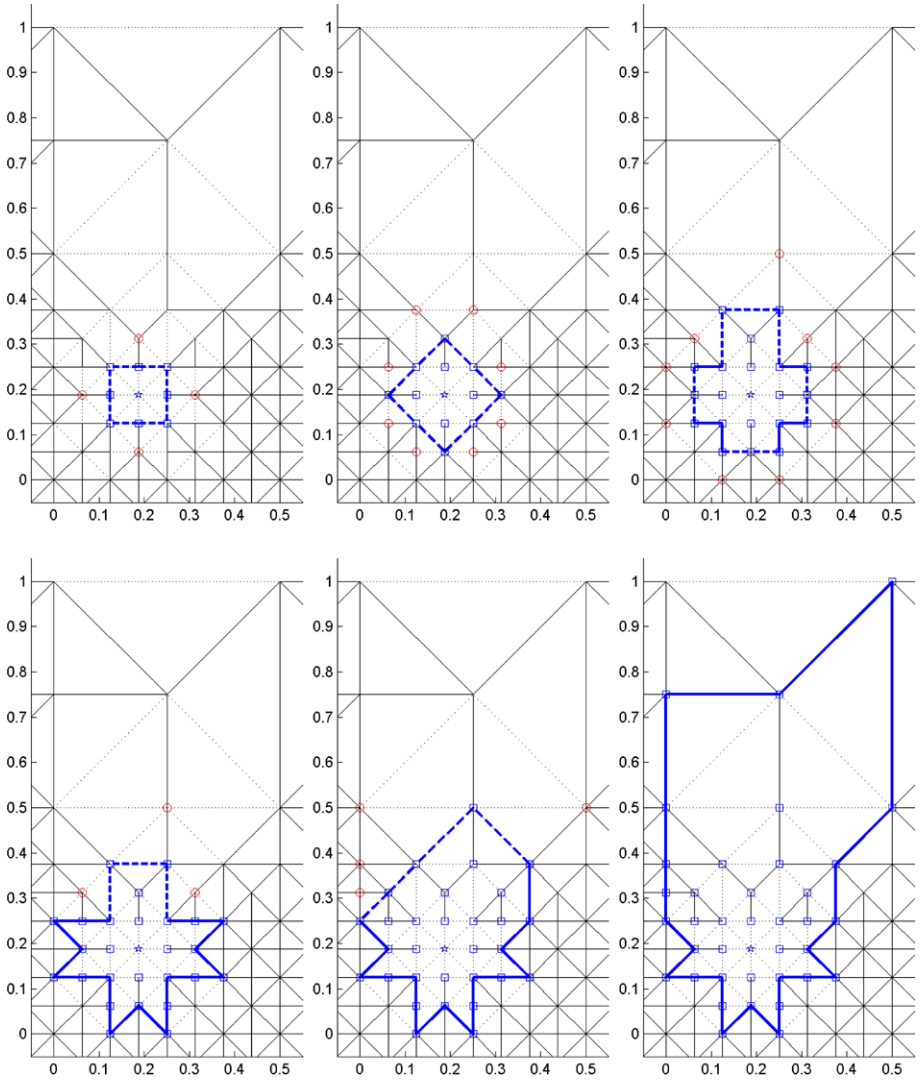

Fig. 2 The status of Algorithm 3 during different stages of the computation of the update node set of x. The *star* is node x. *Squares* are nodes in the update node set. *Circles* are nodes in $\mathcal{Q}$. *Thin solid lines* are grid edges that are AAB for x. *Thin dotted lines* are grid edges which are not AAB. *Thick lines* are grid edges on the frontier of the update node set. *Thick solid lines* are AAB and *thick dashed lines* are not AAB. The *top-left* shows the moment just after neighbors of x have been added to the update node set and the frontier edges for which they are vertices have failed the AAB test. As a result, all nodes opposite these edges have been added to $\mathcal{Q}$. Note that the order in which nodes are removed from $\mathcal{Q}$ is arbitrary, although our implementation uses first-in first-out order. Subsequent plots show subsequent but not sequential stages *left* to *right* and *top* to *bottom*. The *bottom-right* shows the status at the termination of Algorithm 3. All grid edges on the frontier of the update node set have passed the AAB test. Note that all AAB simplices on and within this frontier are part of the update simplex set $\mathcal{S}(x)$ (see Remark 3)

lemma establishes that we can define a ball centered on x with radius linear in the maximum length of grid edges within the ball. This concept is used to define local grid spacing in Theorem 4.4.

Lemma 4.3 *For all x and all $b \in \mathbb{R}^+$ there exists $\tilde{r} \in \mathbb{R}^+$ such that $0 < \tilde{r} = b\hat{h}(\mathcal{B}(x,\tilde{r})) < \infty$.*

Proof We have $b\hat{h}(\mathcal{B}(x,0)) = b\max\{\|x-y\| \mid y \in \mathcal{N}(x)\} > 0$, $b\hat{h}(\mathcal{B}(x,\tilde{r}))$ nondecreasing on r̃, and $\lim_{\tilde{r}\to\infty} b\hat{h}(\mathcal{B}(x,\tilde{r})) = b\hat{h} < \infty$. Therefore, there exists r̃ such that $0 < \tilde{r} < \infty$ and $\tilde{r} = b\hat{h}(\mathcal{B}(x,\tilde{r}))$. □

As allowed by the previous Lemma, choose $b = \Psi(x) + 1$ where $\Psi(x)$ is defined in (3.12) and define $\check{r}(x)$ to be the minimum r̃ such that $\hat{h}(\mathcal{B}(x,\tilde{r})) = \tilde{r}/(\Psi(x)+1)$. Define

$$\mathcal{B}_1(x) = \mathcal{B}(x, \check{r}(x)\Psi(x)/(\Psi(x)+1)) \tag{4.3}$$

and

$$\mathcal{B}_2(x) = \mathcal{B}(x, \check{r}(x)). \tag{4.4}$$

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

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

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

For the proof of Theorem 4.4 below we are only concerned with a single execution of Algorithm 3. Also, we are only considering the update node set $\overrightarrow{\mathcal{Y}}(x)$ and not the dependent node set $\overleftarrow{\mathcal{Y}}(x)$, so we abbreviate $\mathcal{Y} = \overrightarrow{\mathcal{Y}}(x)$, to make the notation less cluttered. Let a subscript $i \geq 0$ represent the state of a variable at the beginning of the $(i+1)^{\text{st}}$ iteration of the **while** loop in Algorithm 3. For example, $\mathcal{Y}_i$ is the state of the update node set $\mathcal{Y}$ at the beginning of iteration $i+1$. From Lines 3 and 4 of Algorithm 3, we have $y_{i+1} = \texttt{Pop}(\mathcal{Q}_i)$ and $\mathcal{Y}_{i+1} = \mathcal{Y}_i \cup \{y_{i+1}\}$.

Theorem 4.4 *Let $x \in \underline{\Omega}$. Let $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$. The subroutine call* `ComputeUpdateSet(`x`)` *terminates before iteration $m(\mathcal{B}_2)$ of the* **while** *loop with $\hat{r}(x) \leq \check{r}(x)$. The set $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$ is DC for x and AAB for x.*

Section 4.3 explains why requiring that x be sufficiently far from the boundary by assuming $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$ does not impact convergence.

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

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

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

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

it must be that $x_k^s \in \mathcal{B}_1^C$ for $1 \le k \le d$. By (3.4) and the definition of $\mathcal{B}_1^C$, we have $r_s(x) > \check{r}(x)\Psi(x)/(\Psi(x)+1)$. Thus, by (3.9), since $\hat{h}_s \le \hat{h}(\mathcal{B}_2) = \check{r}(x)/(\Psi(x)+1)$ we have

$$\frac{r_s(x)}{\hat{h}_s} > \frac{\check{r}(x)\Psi(x)/(\Psi(x)+1)}{\hat{h}(\mathcal{B}_2)} = \Psi(x). \tag{4.5}$$

By definition, s is DRB for x. By Proposition 3.6, s is AAB for x and by the **if** condition in Line 8, $y_{\check{\imath}}$ did not enter $\mathcal{Q}$, which is a contradiction.

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

Consider each $(d-1)$-dimensional simplex face $s \in \mathcal{F}_\partial(\mathcal{Y}_{\tilde{\imath}})$. Because $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$, $x \notin \underline{\partial\Omega}$. So, since $\mathcal{N}(x) \subseteq \mathcal{Y}_{\tilde{\imath}}$, x is not a vertex of s. There exists $\check{\jmath} \le \tilde{\imath}$ such that $y_{\check{\jmath}} = x_k^s$ for some k such that $1 \le k \le d$ and $x_k^s \in \mathcal{Y}_j$ for all k such that $1 \le k \le d$. In other words, $\check{\jmath}$ is the first **while** iteration when all vertices of s are in $\mathcal{Y}$. By the **foreach** loop, if s is not AAB for x then $\mathcal{M}(s) \subset \mathcal{Q}_{\check{\jmath}}$, which implies $\mathcal{M}(s) \subset \mathcal{Y}_{\tilde{\imath}}$, meaning $\mathcal{M}(s) \setminus \mathcal{Y}_{\tilde{\imath}} = \emptyset$. But by definition, $\mathcal{M}(s) \setminus \mathcal{Y}_{\tilde{\imath}} \ne \emptyset$ or for all k, $x_k^s \in \underline{\partial\Omega}$. It follows that s is AAB or for all k, $x_k^s \in \underline{\partial\Omega}$. Since $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$, if for all k, $x_k^s \in \underline{\partial\Omega}$, then for all k, $x_k^s \in \mathcal{B}_1^C$. Thus, by (4.5), s is DRB for x, implying s is AAB for x. Therefore, we have $\mathcal{F}_\partial(\mathcal{Y}_{\tilde{\imath}})$ is AAB for x, and by Lemma 4.2, DC for x. □

The following corollary states that for x not too near the boundary $\underline{\partial\Omega}$, $\mathcal{S}(x)$ is DC for x and NGA for x. By Proposition 2.2, DC for x implies the consistency of the Hamiltonian $\underline{H}(x, \mathcal{S}(x), \phi, \mu)$. By Proposition 3.2, NGA for x implies the causality of the discrete equation $\underline{H}(x, \mathcal{S}(x), \phi, \mu) = 0$.

Corollary 4.5 *Let $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$. Then $\mathcal{S}(x)$ is DC for x and NGA for x.*

Proof By definition, $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x)) \subseteq \underline{\mathcal{S}}_d(\overrightarrow{\mathcal{Y}}(x)) \subseteq \underline{\mathcal{S}}(\overrightarrow{\mathcal{Y}}(x))$. By the definition of $\mathcal{S}(x)$ and since, by Theorem 4.4, $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$ is AAB for x, we have $\mathcal{S}(x) \supset \mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$. By Theorem 4.4, $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$ is DC for x, so its superset $\mathcal{S}(x)$ is DC for x. By definition $\mathcal{S}(x)$ is AAB for x, so by Theorem 3.3, $\mathcal{S}(x)$ is NGA for x. □

Remark 3 Note that nodes are never removed from $\overrightarrow{\mathcal{Y}}(x)$ in Algorithm 3. This subroutine call `ComputeUpdateSet`(x) expends most of its effort adding nodes to $\overrightarrow{\mathcal{Y}}(x)$ such that the frontier $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$ is DC and AAB; however, by (4.1) the update simplex set $\mathcal{S}(x)$ contains not just the simplices in $F_\partial(\overrightarrow{\mathcal{Y}}(x))$, but also all simplices within this frontier which are AAB. In particular, any zero dimensional simplex is automatically AAB, so direct updates from all nodes in $\overrightarrow{\mathcal{Y}}(x)$ are always possible even if those nodes are not part of any higher dimensional AAB simplex.

Remark 4 Although the update simplex set $\mathcal{S}(x)$ contains many simplices interior to $\mathcal{F}_\partial(\overrightarrow{\mathcal{Y}}(x))$, during the execution of the main Algorithm 2 MAOUM will not use updates

from a simplex whose nodes are in $\overrightarrow{\mathcal{Y}}(x)$ but which is not AAB, even if all nodes of that simplex have known value. This behavior is in contrast to AFOUM, which will use updates from any simplex on the accepted front. Because MAOUM may ignore an update from a non-AAB but closer simplex with known node values which AFOUM uses, it is possible that MAOUM will display higher local truncation error than AFOUM on problems with curved characteristics (such as those with inhomogeneous cost). MAOUM could be modified to use all simplices whose nodes are in $\overrightarrow{\mathcal{Y}}(x)$ and have known value in its updates, at the cost of abandoning its precomputed stencil property.

4.2 Discrete Solution

If the discrete equation $\underline{H}(x, \mathcal{S}(x), \phi, \mu) = 0$ is causal, then Algorithm 2 computes the solution to (2.5) in a single pass (not including initialization) of the nodes in $\mathcal{X}$. Causality is the property that allows Dijkstra's algorithm to be used to compute a minimal path to a goal node on a weighted graph in a single pass over the nodes of the graph. Both the semi-Lagrangian discretization in [30] and the Eulerian finite differences discretization in [23] of the Eikonal equation are causal and a single-pass algorithm can be used to find a solution to the respective discretized systems of equations. Our argument for the applicability of Algorithm 2 to solving (2.5) is based on those in [23, 30].

All nodes in $\mathcal{X}$ are added to $\mathcal{H}$ in Line 5. When a node $x \in \mathcal{X}$ is removed from $\mathcal{H}$ we say that x is *accepted*. A node x is accepted just once in Line 7, since x is never re-added to $\mathcal{H}$ once it has been removed. Once x is accepted, its value $\underline{v}(x)$ is fixed.

We argue that the nodes are accepted in non-decreasing order. This is because the node x that currently has the smallest value $\underline{v}(x)$ is accepted in Line 7. However, by Theorem 3.1, any node y whose value is updated in Line 11, must not have a smaller value than x, that is $\underline{v}(y) \geq \underline{v}(x)$. So the next node to be accepted in Line 7 must not have a smaller value than $\underline{v}(x)$. It follows by induction on the iterations of the **while** loop that the nodes are accepted in non-decreasing value order.

When a node x is accepted, its final value $\underline{v}(x)$ has been calculated using only simplices $s \in S(x)$ such that all vertex nodes x_i^s are already accepted. But since the nodes are accepted in non-decreasing value order, $\underline{v}(x)$ has been calculated using all simplices $s \in S(x)$ such that all vertex nodes x_i^s have final values $\underline{v}(x_i^s)$ such that $\underline{v}(x_i^s) \leq \underline{v}(x)$. By the causality of the discrete equation, we have $\underline{H}(x, \mathcal{S}(x), \underline{v}, \underline{v}(x)) = 0$.

4.3 Convergence

For convergence of $\underline{u}$ to u, it is important that for all $x \in \mathcal{X}$ the maximum extent of the update node set, $\hat{r}(x)$, goes to zero as the grid spacing goes to zero. A more rigorous treatment of convergence can be found in [1].

Propositions 2.1 and 2.2 state that $\underline{H}$ is monotone and consistent. Noting that the update simplex set $\mathcal{S}(x)$ always contains the immediate neighbours of x (see Remark 3) it can be shown that $\underline{u}$ is uniformly bounded independent of the grid spacing $\hat{h}$ (a straightforward extension of [1, Theorem 4.23], since the simplex update set of MAOUM is always a superset of the update set used in the axis-aligned anisotropy algorithm in that theorem). Consequently, we can prove the convergence of $\underline{u}$ to the solution u of the HJ PDE (1.1) as the grid spacing goes to zero [5]. However, consistency of $\underline{H}(x, \mathcal{S}, \phi, \mu)$ requires that $\mathcal{S}$ be DC for x. In Corollary 4.5, $\underline{\partial\Omega} \cap \mathcal{B}_1 = \emptyset$ is a sufficient condition for $\mathcal{S}$ to be DC, but for x near the computational boundary the condition may not hold.

Fortunately, if we examine the requirements for convergence [5] carefully, we see that the problem goes away. If for any $x \in \Omega$ there is a sequence of grids $\mathcal{X}_k$ and grid nodes $x_k \in \mathcal{X}_k$, such that the grid spacing of $\mathcal{X}_k$ goes to 0, $x_k \to x$, and $\mathcal{S}(x_k)$ is DC for x_k, then we can use Proposition 2.2 to prove convergence. It can be shown that such sequences exist. For example, consider $x \in \Omega$ to be at distance $\epsilon > 0$ from $\partial\Omega$. For a grid with small enough spacing, there exists a node y close to x and $\check{r}(y)$ such that the radius $\check{r}(y)\Psi(y)/(\Psi(y)+1)$ of $\mathcal{B}_1(y)$ is sufficiently small so that $\underline{\partial\Omega} \cap \mathcal{B}_1(y) = \emptyset$.

Although $\underline{u}$ converges to u, for many problems $\overrightarrow{\mathcal{Y}}(x)$ is not symmetric (i.e., in general $\overrightarrow{\mathcal{Y}}(x) \neq \overleftarrow{\mathcal{Y}}(x)$) and the approximate solution $\underline{u}$ may not be *geometric* in the sense that it is possible that $\underline{u}(x) < \underline{u}(y)$ even though at a coarser scale the node values are increasing in the y to x direction. In other words, x may be accepted before y even if more generally the order of node acceptance is roughly in the y to x direction. Furthermore, for nodes near $\partial\Omega$ this effect may persist as $h \to 0$. However, the local range over which $\underline{u}$ may not be geometric is bounded by $\hat{r}(x)$, which shrinks to 0 as $h \to 0$.

4.4 Complexity

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

To analyze computational complexity it is useful to prove a lemma bounding the maximum number of nodes in any update region or dependent region as $N \to \infty$. Let $\rho = \hat{h}/\check{h}$, where $\hat{h}$ is the maximum grid edge length defined in (4.2) and $\check{h} = \min\{\|y - z\| \mid y \in \mathcal{X} \text{ and } z \in \mathcal{N}(y)\}$ is minimum grid edge length. Let $\overrightarrow{M} = \max_{x \in \underline{\Omega}} |\overrightarrow{\mathcal{Y}}(x) \cap \mathcal{X}|$ and $\overleftarrow{M} = \max_{x \in \mathcal{X}} |\overleftarrow{\mathcal{Y}}(x) \cap \underline{\Omega}|$ be the maximum number of nodes over all update node sets and dependent node sets, respectively. Let $\hat{\Psi} = \max_{x \in \underline{\Omega}} \Psi(x)$.

Lemma 4.6 *As $N \to \infty$, $\overrightarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$ and $\overleftarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$.*

Proof By Theorem 4.4, after executing Algorithm 3,

$$\hat{r}(x) \leq (\Psi(x)+1)\hat{h}(\mathcal{B}_2) \leq (\hat{\Psi}+1)\hat{h},$$

for all $x \in \underline{\Omega}$. Given $\check{h}$ is the minimum spacing between nodes, we bound above the number of nodes that can be packed into $\mathcal{B}(x, \hat{r}(x))$, for all x:

$$|\mathcal{B}(x, \hat{r}(x)) \cap \mathcal{X}| = \mathcal{O}\left(\left(\frac{\hat{r}(x)}{\check{h}}\right)^d\right) = \mathcal{O}\left(\left(\frac{\hat{\Psi}\hat{h}}{\check{h}}\right)^d\right) = \mathcal{O}((\hat{\Psi}\rho)^d).$$

Therefore, $\overrightarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$.

The symmetry $y \in \overleftarrow{\mathcal{Y}}(x)$ if and only if $x \in \overrightarrow{\mathcal{Y}}(y)$ is evident from Line 3 of Algorithm 2, and Lines 4 and 5 of Algorithm 3. Since $\hat{r}(x) \leq (\hat{\Psi}+1)\hat{h}$ for all $x \in \underline{\Omega}$, the same holds for the maximum extent of the dependent node set: $\max_{y \in \overleftarrow{\mathcal{Y}}(x)} \|x - y\| \leq (\hat{\Psi}+1)\hat{h}$ for all $x \in \underline{\Omega}$. Following the same argument as above, $\overleftarrow{M} = \mathcal{O}((\hat{\Psi}\rho)^d)$. □

We first consider the computational cost of initializing the update regions of nodes $x \in \underline{\Omega}$ (and dependent regions of nodes $x \in \mathcal{X}$) using the subroutine Algorithm 3. In Line 4 of Algorithm 2, `ComputeUpdateSet` is called N times, once for each node x. For each call to `ComputeUpdateSet`, there are $\mathcal{O}(\overrightarrow{M})$ iterations of the **while** loop in Algorithm 3. For each **while** iteration, a **foreach** loop visits the $(d-1)$-dimensional simplex faces that include node y for a total of $\mathcal{O}(P_s)$ iterations, where $P_s = \max_{y \in \mathcal{X}} |\{s \in \underline{\mathcal{S}}_d \mid y \in s\}|$. Each iteration of the **foreach** loop performs a constant number of operations. Thus, the number of operations to execute `ComputeUpdateSet` N times is $\mathcal{O}(N\overrightarrow{M}P_s))$. Assuming P_s is bounded as $N \to \infty$, the computational complexity of initializing the update node sets is $\mathcal{O}(N\overrightarrow{M})$.

Now we examine the complexity of executing the **while** loop of Algorithm 2. For each iteration of the **while** loop, a node x is accepted. A node is accepted only once, so there are N iterations in total. For each iteration, a **foreach** loop visits the $\mathcal{O}(\overleftarrow{M})$ unaccepted nodes in the dependent node set of x. For each such dependent node y, each neighbor of x must be tested for membership in $\overrightarrow{\mathcal{Y}}(y)$ to determine the update simplices in Line 10 at a cost of $\mathcal{O}(P_n \log \overrightarrow{M})$, where $P_n = \max_{x \in \mathcal{X}} |\mathcal{N}(x)|$ and $\overrightarrow{\mathcal{Y}}(y)$ is implemented as a self-balancing binary search tree. Also, for each y, $\mathcal{O}(P_s)$ updates are performed at Line 11 and the binary min-heap implementation of $\mathcal{H}$ must be updated at a complexity of $\mathcal{O}(\log N)$. Thus, the number of operations in executing the **while** loop is $\mathcal{O}(N\overleftarrow{M}(P_n \log \overrightarrow{M} + P_s + \log N))$. Assuming P_n and P_s are bounded as $N \to \infty$, the complexity is $\mathcal{O}(N\overleftarrow{M} \log N)$, which is determined by the heap update operations.

The complexity of executing the **while** loop of Algorithm 2 dominates the complexity of the initialization, including that of the calls to the subroutine Algorithm 3, which we determined above to be $\mathcal{O}(N\overrightarrow{M})$ and the initialization of the heap which is $\mathcal{O}(N \log N)$. So, by Lemma 4.6, the overall asymptotic complexity of Algorithm 2 is

$$\mathcal{O}(N\overleftarrow{M} \log N) = \mathcal{O}(N(\hat{\Psi}\rho)^d \log N).$$

Single-pass algorithms for isotropic problems [23, 30] and limited anisotropic problems [3, 25] have a complexity of $\mathcal{O}(N \log N)$. The extra $(\hat{\Psi}\rho)^d$ factor in the complexity of MAOUM is due to the number of nodes in the update node set, which in MAOUM has been expanded beyond direct grid neighbors. In [27] a complexity of $\mathcal{O}(N\hat{\Upsilon}^{d-1} \log N)$ is derived for AFOUM, where $\hat{\Upsilon} = \max_{x \in \underline{\Omega}} \Upsilon(x)$. As shown in Remark 2, $\hat{\Psi}$ is smaller than $\hat{\Upsilon}$. However, AFOUM's complexity has a reduced exponent of $d-1$ because the update node set lies on the lower dimensional accepted front.

The complexity claim from [27] does not consider ρ, even though it plays an important role when the grid is highly nonuniform. If we assume ρ is bounded as $N \to \infty$, then MAOUM's complexity is $\mathcal{O}(N\hat{\Psi}^d \log N)$. However, the optimal allocation of grid nodes for Algorithm 2 may cause the grid to be more nonuniform as $N \to \infty$. In Sect. 5 we examine the relationship between the solution accuracy and the computational cost experimentally on several problems with a progression of increasingly-refined uniform and nonuniform grids.

For practical grid sizes, it is often the CPU time spent computing node value updates which dominates the CPU time of executing the entire algorithm, despite the fact that the computational complexity of the heap updates dominates in asymptotic analysis. Computing a node value may involve iteratively solving a nonlinear equation or a numerical optimization which can be quite computationally intensive. For this reason, in Sect. 5 we use the number of updates as our measure of computational cost.

The asymptotic storage complexity for MAOUM is dominated by that of storing the sets $\overrightarrow{\mathcal{Y}}(x)$ and $\overleftarrow{\mathcal{Y}}(x)$ for each node $x \in \mathcal{X}$, which is $\mathcal{O}(N(\overrightarrow{M} + \overleftarrow{M})) = \mathcal{O}(N(\hat{\Psi}\rho)^d) = \mathcal{O}(N)$. Asymptotically this is the same as AFOUM but the factor of $(\hat{\Psi}\rho)^d$ does have a significant impact on practical memory requirements which is reported in Sect. 5.

One of the drawbacks of MAOUM is that the sets $\overrightarrow{\mathcal{Y}}(x)$ and $\overleftarrow{\mathcal{Y}}(x)$, must be precomputed and stored for each x. While this stencil precomputation does not affect the asymptotic computational and storage complexity as shown above, it is a significant portion of the computational and storage cost in practice (see Sect. 5). The computational and storage overhead may become less of a concern if the same HJ PDE is repeatedly solved on the same grid with different boundary conditions, allowing the precomputed stencils to be reused several times. Also, stencils could be reused to solve a different HJ PDE with the same anisotropy $\Upsilon(x)$ on the same grid. Furthermore, in order to reduce storage requirements, it is possible to store a reasonably tight superset of $\overrightarrow{\mathcal{Y}}(x)$ and $\overleftarrow{\mathcal{Y}}(x)$ with a compact polygonal representation [1, Chap. 5]. In preliminary experiments we found that the number of stencil nodes was not greatly increased, which is important for keeping local truncation error and update counts down.

5 Experiments

We present numerical experiments to test the convergence, computational cost, and accuracy of Algorithm 2. We also demonstrate that MAOUM can be used to solve practical problems, such as a seismic imaging problem [27] and a robot navigation problem with obstacles and wind. The experiments indicate that MAOUM is particularly suited to problems in which the characteristics are highly-curved in some regions of Ω and straight or nearly straight elsewhere. For such problems it is more efficient to refine the grid only in the regions with curved characteristics.

Although we assume continuity of function f for the theoretical consistency of the numerical Hamiltonian $\underline{H}$, one of the examples in Sect. 5.3 and the example in Sect. 5.4 have discontinuous f. Despite this property, our experiments indicate that the respective numerical solutions are converging. For all experiments reported below $d = 2$ and $\Omega \subset \mathbb{R}^2$. However, Algorithm 2 can be used for problems in any dimension. In the following, let $x = (x_1, x_2)^T \in \mathbb{R}^2$ and $y = (y_1, y_2)^T \in \mathbb{R}^2$. Note that the boundary conditions are defined only on internal boundaries in the following examples, which is not the case in general. In this section we use the notation $\partial\Omega$ to specify the internal boundary where the boundary conditions are defined, which differs from the usual use of $\partial\Omega$ as the topological boundary of Ω.

We use a Maubach grid [18] in our implementation but MAOUM works with any simplicial grid (even unstructured or obtuse grids). Examples of uniform Maubach grids are shown in Fig. 3 and nonuniform Maubach grids in Fig. 4. It is a semi-structured simplicial grid that can be adaptively refined and used in any dimension.

For the `Update` function we must solve (2.9), which involves finding the minimum of a convex function $\eta_{\underline{v}}^s$ for each s. We use the golden section optimization [15] which is sufficient for $d = 2$. This optimization method is likely slower than more sophisticated alternatives that use derivative information but it is very simple to implement and can minimize all convex functions in one dimension, even those for which the derivative is sometimes undefined.

For the experiments below we use implementations of MAOUM and AFOUM that include some minor optimizations. Our implementation of both MAOUM and AFOUM does

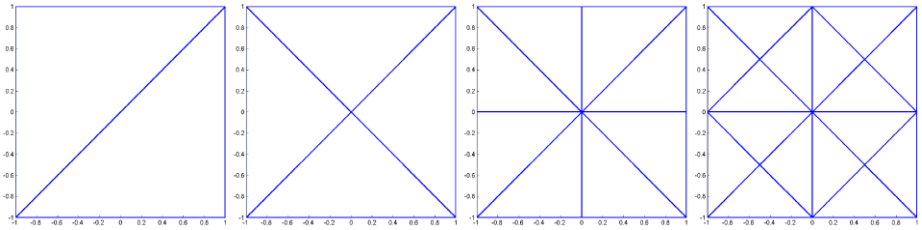

Fig. 3 A sequence of uniformly-refined Maubach grids with 0, 1, 2, and 3 levels of refinement

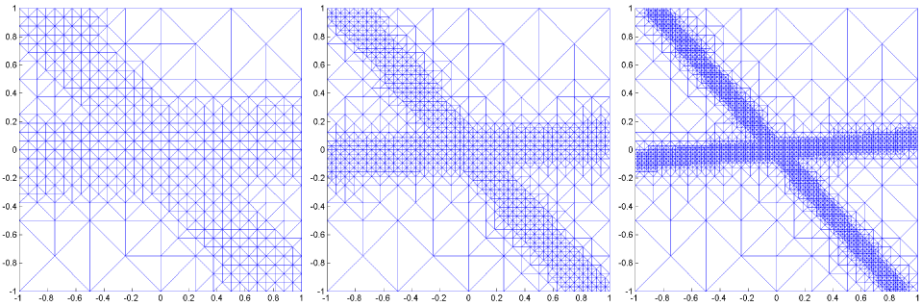

Fig. 4 The sequence of nonuniformly-refined Maubach grids with 10, 12, and 14 levels of refinement used for the problem with homogeneous Hamiltonian and $\mathcal{A}_f$ that is a non-grid aligned rectangle with length that is twice its width

not perform an update for node x from a grid edge s with vertices x_1^s and x_2^s in the update set when $(x - x_1^s)$ and $(x - x_2^s)$ are collinear. Furthermore, our implementation of AFOUM does a local visibility check when deciding whether to update x from a segment s on the accepted front. If x is behind s in the sense that s is an edge of a grid triangle with all vertices accepted and any line segment with end points at x and s cuts through the interior of the triangle, then AFOUM does not perform the update. However, our implementation does not perform a global visibility check that tests if all line segments with end points at x and s pass through some segment other than s on the accepted front. These implementation optimizations could be generalized when solving problems with $d > 2$ using MAOUM or AFOUM.

In order to ensure that MAOUM indeed computes the solution to (2.5) in a single pass, we have computed the residual of (2.5) for all of the examples below and have not found a case where it is significantly larger than machine epsilon.

5.1 Convergence

We demonstrate numerically that the output $\underline{v}$ of MAOUM converges to the solution u of an anisotropic HJ PDE as the grid spacing goes to zero. For this experiment we use a series of increasingly fine uniform Maubach grids, such as those in Fig. 3.

We use a homogeneous Hamiltonian (i.e. $H(x, p) = H(p)$) and $\mathcal{A}_f$ that is a non-grid aligned ellipse. For the purpose of implementation it is easiest to define $c(x, y) = c(y)$ from (2.11). Let $c(y) = \|By\|_2$, where B is the 2×2 matrix

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \cos(\pi/6) & -\sin(\pi/6) \\ \sin(\pi/6) & \cos(\pi/6) \end{bmatrix}$$

Table 2 The problem has a homogeneous Hamiltonian and $\mathcal{A}_f$ that is a non-grid aligned ellipse with anisotropy $\Upsilon = 4$. The table shows grid spacing h versus maximum errors and average errors of approximate solutions computed on a progression of uniform grids by MAOUM. *Level* is the level of grid refinement. M is the average over $x \in \underline{\Omega}$ of the number of nodes in the update node set. h is the grid spacing in the horizontal and vertical directions. *Updates* is the number of times `Update(y, s)` is called. e_∞ is the $\mathcal{L}_\infty$-error in the approximation $\underline{u}$ to the true solution u of (1.1). e_1 is the $\mathcal{L}_1$-error. The error convergence rates r_∞^h and r_1^h are computed with respect to the grid spacing h

Level	N	M	h	Updates	e_∞	r_∞^h	e_1	r_1^h
10	1089	56.6	6.3e−2	36750	3.1e−2		2.9e−3	
12	4225	60.2	3.1e−2	152080	8.9e−3	1.8	1.2e−3	1.3
14	16641	62.1	1.6e−2	619062	3.8e−3	1.2	4.7e−4	1.3
16	66049	63.0	7.8e−3	2498574	1.8e−3	1.1	2.1e−4	1.2
18	263169	63.5	3.9e−3	10040504	8.2e−4	1.1	9.6e−5	1.1

The cost function c rotates y by $\pi/6$ counterclockwise around the origin and then scales it by 4 in the vertical axis before taking the Euclidean norm.

Let $\mathcal{D} = \{x \in \mathbb{R}^2 \mid c(x) \leq 0.4\}$. The domain for this problem is $\Omega = [-1, 1]^2 \setminus \mathcal{D}$ and the boundary is $\partial\Omega = \partial\mathcal{D}$. The boundary conditions are given as $g(x) = c(x)$ for $x \in \mathcal{D}$. Notice that the boundary conditions are defined throughout $\mathcal{D}$ and thus extend beyond $\partial\Omega$. We also allow $\mathcal{S}(x)$ to extend beyond $\partial\Omega$. We define the boundary to be the ellipse $\mathcal{D}$ to exclude errors caused by poor approximation of the solution u near the origin, where u is not differentiable. It is not necessary to give boundary conditions on the external boundary of Ω since all characteristics flow out of this boundary. The grid resolutions and corresponding errors are listed in Table 2.

5.2 Nonuniform Grid

To determine what benefit MAOUM gains from refining a grid intelligently, we use an anisotropic HJ PDE which has a solution with kinks where the gradient is undefined. We run Algorithm 2 on a series of increasingly-fine uniform grids and a series of increasingly nonuniform grids, where newly added nodes are concentrated around the parts of the solution where the gradient is undefined. For comparison, we run also AFOUM on both the uniform and nonuniform grid series. For all four combinations of algorithm and grid series we plot the solution error vs computational cost to see if MAOUM performs better on a well-chosen nonuniform grid.

We use a homogeneous Hamiltonian (i.e. $H(x, p) = H(p)$) and $\mathcal{A}_f$ that is a non-grid aligned rectangle. Let $c(y) = \|By\|_\infty$, where B is the 2×2 matrix

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \cos(\pi/8) & -\sin(\pi/8) \\ \sin(\pi/8) & \cos(\pi/8) \end{bmatrix}$$

The cost function c rotates y by $\pi/8$ counterclockwise around the origin and then scales it by 2 in the vertical axis before taking the maximum norm. The domain for this problem is $\Omega = [-1, 1]^2 \setminus O$ and the boundary is $\partial\Omega = O$, where O is the origin. The boundary conditions are given as $g(O) = 0$.

Part of the nonuniform grid series used is shown in Fig. 4. The grids are refined within distance $2\check{h}\Upsilon$ of the two lines

$$x_2 = \frac{\sin(-\pi/8) + \frac{1}{2}\cos(-\pi/8)}{\cos(-\pi/8) - \frac{1}{2}\sin(-\pi/8)} x_1$$

Table 3 The problem has a homogeneous Hamiltonian and $\mathcal{A}_f$ that is a non-grid aligned rectangle with length that is twice its width. The table shows the number of updates versus maximum errors (*top*) and average errors (*bottom*) of approximate solutions computed on a progression of nonuniform and uniform grids by MAOUM and AFOUM. *Nonuni* indicates the use of a nonuniform grid, while *Uni* indicates a uniform grid. *Level* is the maximum level of grid refinement. *Updates* is the number of times `Update(y, s)` is called. e_∞ is the $\mathcal{L}_\infty$-error in the approximation $\underline{u}$ to the true solution u of (1.1). e_1 is the $\mathcal{L}_1$-error. Because we use both uniform and nonuniform grids, the error convergence rates r_∞^U and r_1^U are computed with respect to Updates rather than h as in Table 2. If we computed the rates for uniform grids with respect to h instead, they would be approximately double those listed in the table because Updates is proportional to N (which is $\mathcal{O}(1/h^2)$), but in that case we could no longer sensibly compare the rates for uniform and nonuniform grids

Grid	Level	N	MAOUM			AFOUM		
			Updates	e_∞	r_∞^U	Updates	e_∞	r_∞^U
Nonuni	10	621	9908	5.3e−2		55509	5.4e−2	
	12	1443	23756	3.6e−2	0.42	182632	3.9e−2	0.26
	14	3051	51488	2.6e−2	0.45	490629	2.8e−2	0.35
	16	6305	108008	1.8e−2	0.51	1203283	1.9e−2	0.41
	18	12911	223248	1.2e−2	0.47	2837945	1.4e−2	0.40
Uni	10	1089	18668	5.3e−2		19040	5.4e−2	
	12	4225	76184	3.6e−2	0.26	75887	3.9e−2	0.22
	14	16641	307868	2.6e−2	0.25	303647	2.8e−2	0.25
	16	66049	1238000	1.8e−2	0.27	1216300	1.9e−2	0.27
	18	263169	4965536	1.2e−2	0.25	4869733	1.4e−2	0.25
				e_1	r_1^U		e_1	r_1^U
Nonuni	10	621	9908	2.3e−3		55509	2.5e−3	
	12	1443	23756	1.1e−3	0.85	182632	1.2e−3	0.60
	14	3051	51488	5.2e−4	0.94	490629	5.9e−4	0.72
	16	6305	108008	2.6e−4	0.95	1203283	3.0e−4	0.77
	18	12911	223248	1.3e−4	0.90	2837945	1.6e−4	0.75
Uni	10	1089	18668	2.3e−3		19040	2.5e−3	
	12	4225	76184	1.1e−3	0.53	75887	1.2e−3	0.52
	14	16641	307868	5.2e−4	0.52	303647	5.9e−4	0.51
	16	66049	1238000	2.5e−4	0.51	1216300	2.9e−4	0.51
	18	263169	4965536	1.2e−4	0.51	4869733	1.4e−4	0.51

and

$$x_2 = \frac{\sin(-\pi/8) - \frac{1}{2}\cos(-\pi/8)}{\cos(-\pi/8) + \frac{1}{2}\sin(-\pi/8)} x_1,$$

where $\check{h}$ is the minimum grid edge length after refinement is complete.

The results for all four combinations of algorithm and grid series are compared in Table 3 and Fig. 5. Note that in order to compare results from uniform and nonuniform grids we use the number of updates rather than the grid spacing as the independent variable in the plots and rate calculations. To properly interpret the relative performance of MAOUM and AFOUM in Fig. 5, one needs to understand the extra cost involved in the initialization of the update sets in Algorithm 3 of MAOUM. Between 45 and 54 percent of MAOUM's

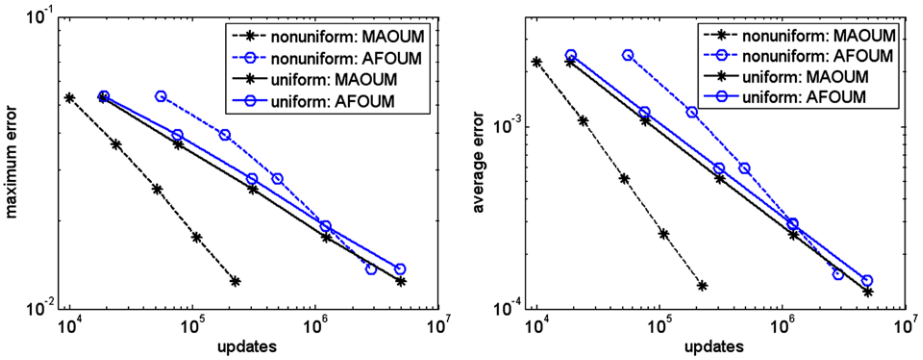

Fig. 5 Error versus number of updates for the problem with homogeneous Hamiltonian and $\mathcal{A}_f$ that is a non-grid aligned rectangle with length that is twice its width. The values plotted are from Table 3

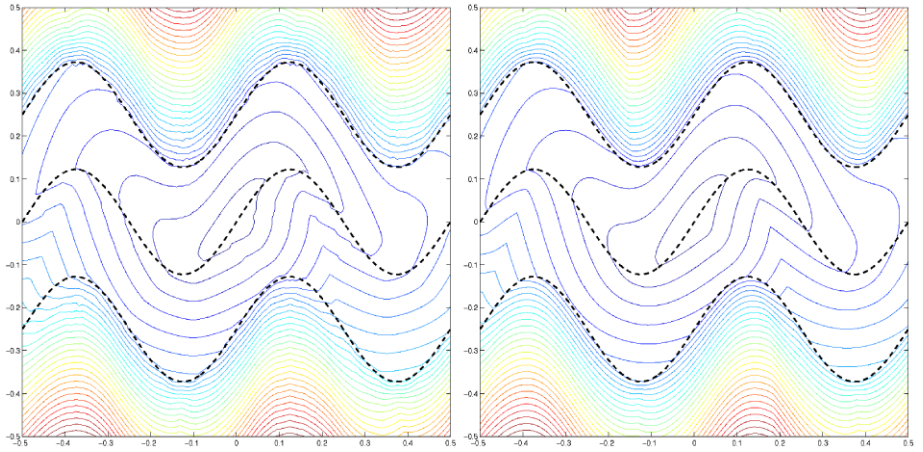

Fig. 6 Contours of first-arrival times of a seismic wave for the layered example computed using a uniform Maubach grid of level 13 with 8321 nodes (*on left*) and level 18 with 263169 nodes (*on right*)

total run time was spent in Algorithm 3. If we consider the ratio of total run time (including initialization) to number of updates, MAOUM took between 122 and 168 percent of the time per update of AFOUM. Consequently, AFOUM gets nearly the same error as MAOUM in significantly less time for uniform grids but MAOUM clearly wins for nonuniform grids. The memory footprint of MAOUM was as much as 3.0 times that of AFOUM, because of the overhead of storing computed stencils.

5.3 Seismic Imaging

We consider the seismic imaging examples from the top-left and bottom-right of Fig. 6 in [27]. The domain for this problem is $\Omega = [-0.5, 0.5]^2 \setminus O$ and the boundary is $\partial\Omega = O$, where O is the origin. The boundary conditions are given as $g(O) = 0$. A wave propagates from the origin and passes through Ω, which is split into four layers by three vertically-shifted sinusoidal curves. The problem is to compute the first arrival time for the seismic wave.

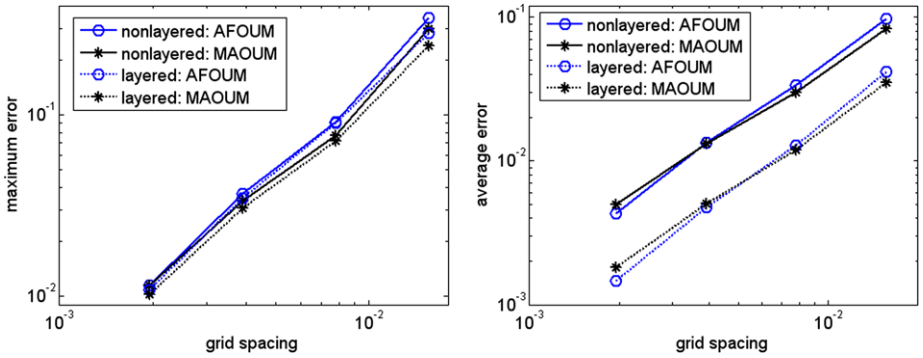

Fig. 7 Error versus grid spacing h for first-arrival time of a seismic wave. The values plotted are from Table 4

The set $\mathcal{A}_f(x)$ is an ellipse with the long axis aligned with the tangent of the sinusoidal curves at x_1 and hence the problem is both anisotropic and inhomogeneous. The dimensions of the elliptical $\mathcal{A}_f(x)$ are constant within a layer. The layers in the top-left example are homogeneous, although $\mathcal{A}_f(x)$ is still inhomogeneous in x_1. In all layers, this example has elliptical $\mathcal{A}_f(x)$ with a length of 0.8 and width of 0.2. We call this example *nonlayered*. On the other hand, the layers in the bottom-right example are inhomogeneous. Moving through the layers in the positive x_2 direction, the lengths/widths of the elliptical $\mathcal{A}_f(x)$ are 0.8/0.2, 1.0/1.0, 3.0/1.0, and 0.8/0.2. We call this example *layered*. More details can be found in [27].

We test MAOUM on the layered example using uniform Maubach grids from levels 13 to 18. The computed solution for levels 13 and 18 grids are shown in Fig. 6. Experiments indicate that refining the grid along the sinusoidal layer boundaries does not improve accuracy significantly. We believe this is because the characteristics are curved to roughly the same degree throughout Ω due to the inhomogeneity of $\mathcal{A}_f(x)$. Localized grid refinement is most beneficial when characteristics are highly-curved in some parts of Ω and nearly straight elsewhere.

We compare MAOUM and AFOUM on both the nonlayered and layered seismic examples by running all four algorithm/example combinations on a series of uniform Maubach grids with even levels from 12 to 18. Since we do not have an analytic solution to these examples, the error is computed relative to the solutions computed by AFOUM on a level-20 Maubach grid. The solution error vs grid spacing for all four algorithm/example combinations is plotted in Fig. 7. For these examples on the range of uniform grid resolutions tested both algorithms produce similar results, although MAOUM has a much larger memory footprint. For the nonlayered example, the total run time (including initialization) of MAOUM was between 1.17 and 1.22 times that of AFOUM. The memory footprint of MAOUM was as much as 4.5 times that of AFOUM. For the layered example, the total run time (including initialization) of MAOUM was between 0.71 and 0.79 times that of AFOUM. The memory footprint of MAOUM was as much as 3.3 times that of AFOUM. For both examples between 26 and 29 percent of MAOUM's total run time was spent in Algorithm 3.

Despite the discontinuities in the speed function for the layered case, the average error is significantly larger for the nonlayered case because the layered case has less anisotropy in two of the four layers, and is in fact isotropic in one layer. It is not clear whether the slightly smaller average error displayed by AFOUM on the finer grids in both the layered and nonlayered cases is a manifestation of the potentially larger local truncation error of

Table 4 The problem is to calculate the first-arrival time of a seismic wave. The table shows the grid spacing h versus maximum errors (*top*) and average errors (*bottom*) of approximate solutions computed on a progression of uniform grids by MAOUM and AFOUM for the nonlayered and layered example. *Nonlay*/*Lay* indicates the nonlayered/layered example from the top-left/bottom-right of Fig. 6 in [27] is being solved. *Level* is the maximum level of grid refinement. h is the grid spacing in the horizontal and vertical directions. *Updates* is the number of times `Update(y, s)` is called. e_∞ is the $\mathcal{L}_\infty$-error in the approximation $\underline{u}$ to the solution computed by AFOUM on a level-20 grid. e_1 is the $\mathcal{L}_1$-error. The error convergence rates r_∞^h and r_1^h are computed with respect to the grid spacing h

Example	Level	N	h	MAOUM			AFOUM		
				Updates	e_∞	r_∞^h	Updates	e_∞	r_∞^h
Nonlay	12	4225	1.6e−2	152906	3.0e−1		136869	3.4e−1	
	14	16641	7.8e−3	622304	7.7e−2	2.0	546776	9.1e−2	1.9
	16	66049	3.9e−3	2510808	3.4e−2	1.2	2184552	3.7e−2	1.3
	18	263169	2.0e−3	10078844	1.1e−2	1.5	8745903	1.2e−2	1.7
Lay	12	4225	1.6e−2	102962	2.4e−1		138878	2.8e−1	
	14	16641	7.8e−3	411006	7.2e−2	1.8	548440	9.1e−2	1.6
	16	66049	3.9e−3	1641828	3.1e−2	1.2	2181326	3.5e−2	1.4
	18	263169	2.0e−3	6557579	1.0e−2	1.6	8700919	1.1e−2	1.7
					e_1	r_1^h		e_1	r_1^h
Nonlay	12	4225	1.6e−2	152906	8.4e−2		136869	9.7e−2	
	14	16641	7.8e−3	622304	3.0e−2	1.5	546776	3.3e−2	1.5
	16	66049	3.9e−3	2510808	1.3e−2	1.2	2184552	1.3e−2	1.3
	18	263169	2.0e−3	10078844	5.0e−3	1.4	8745903	4.3e−3	1.6
Lay	12	4225	1.6e−2	102962	3.5e−2		138878	4.1e−2	
	14	16641	7.8e−3	411006	1.2e−2	1.6	548440	1.3e−2	1.7
	16	66049	3.9e−3	1641828	5.0e−3	1.3	2181326	4.7e−3	1.4
	18	263169	2.0e−3	6557579	2.0e−3	1.5	8700919	1.5e−3	1.7

MAOUM (see Remark 4). These two examples are inhomogeneous and hence have curved characteristics; however, the error is measured against a higher resolution AFOUM solution, which may bias the results. What is clear is that any such increase in error is relatively small for these examples.

5.4 Robot Navigation with Wind and Obstacles

An optimal time-to-reach problem with obstacles is a natural candidate for exploiting localized grid refinement. An optimal trajectory that must go around an obstacle to achieve the goal will closely track the obstacle boundary for some portion. Refining the grid to better resolve these obstacle boundaries should allow for a more accurate solution in portions of the domain that do not have an obstacle-free optimal path to the goal. Although this specific example is not physically realistic, it does use data of suitable complexity for realistic scenarios and demonstrates MAOUM on a spatially-inhomogeneous anisotropic problem.

The objective is for a robot to navigate from any location in the domain to a goal in optimal time. To make the task difficult the robot must avoid obstacles on its way to the goal and there is an inhomogeneous but static wind that pushes the robot. The goal is $x^* = (80.75, 46.25)^T$, $g(x^*) = 0$, and $\partial\Omega = \{x^*\}$ The domain is $\Omega = [72, 112] \times$

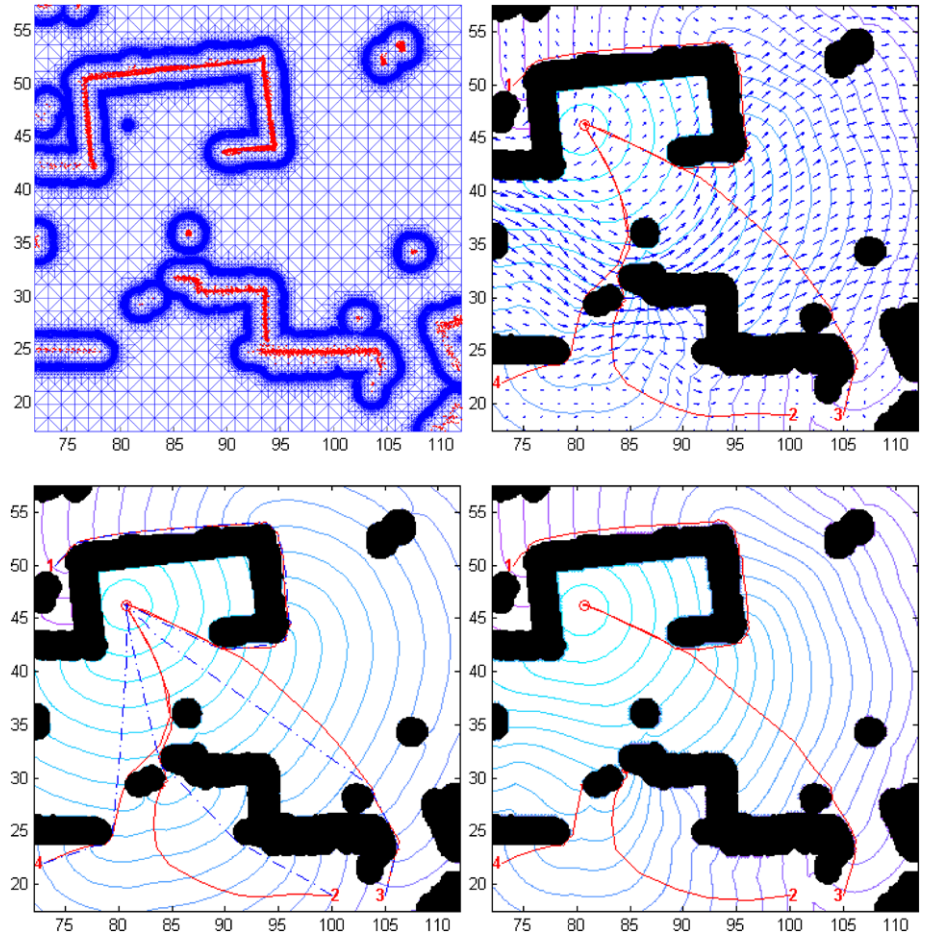

Fig. 8 The problem of navigating a robot with wind and obstacles. The *top-left* shows the laser-rangefinder data (*red*) of the obstacles and the grid (*blue*) refined in a band around the collision set $\mathcal{C}$ and the goal x^*. The other three figures show $\mathcal{C}$ in *solid black*. The *top-right* includes the wind vector field, the contours of the computed time-to-reach function, and four optimal trajectories from different starting locations to the goal. The *bottom-left* compares the optimal trajectories computed with the wind and without. The contours are of the isotropic (i.e. without wind) time-to-reach function. The *solid lines* are trajectories with the wind and the *dash-dotted lines* are trajectories without the wind. The *bottom-right* shows contours of the time-to-reach function and trajectories, computed using a level 15 uniform Maubach grid with roughly the same number of nodes. Note that for trajectories 2 and 4 the characteristic ODE computation of the optimal trajectories gets stuck near the boundary of $\mathcal{C}$, which is insufficiently resolved by the uniform grid

$[17.5, 57.5] \setminus \partial\Omega$. The robot is circular with a radius of 1.1429. The obstacles are a set of points obtained from a laser range finder map downloaded with the Saphira robot control system software [17]. The same point data was used for a isotropic path planning problem in [2], but we map the data from the square domain $[-4000, -500] \times [-3500, 0]$ to Ω. The point obstacles are shown in Fig. 8 (top-left). To prevent the robot from attempting to plan a path through obstacles, it moves at a very slow speed of $f(x, a) = 0.5$ for any $a \in \mathcal{A}$ and any $x \in \mathcal{C}$, where $\mathcal{C}$ is the set of states such that the robot is in collision with a point obstacle. The collision set $\mathcal{C}$ is depicted in black in Fig. 8. The wind velocity is represented by a vector

field shown in Fig. 8 (top-right). We used a vector field from the wind arrays in Matlab to obtain the wind velocity vector function $\vec{f}_w : \Omega \to R^2$.[1]

In the absence of wind, the robot moves with speed $f_r = 75.0$, resulting in an isotropic speed profile $\mathcal{A}_{f_r} = \{y \mid \|y\| \le f_r\}$. Although not physically realistic, we shift the isotropic speed profile by the local wind velocity, so the anisotropic speed profile is

$$\mathcal{A}_f(x) = \{y \mid \|y - \vec{f}_w(x)\| \le f_r\}.$$

Note that $f_r = 75.0 > \max_{x\in\Omega} \|\vec{f}_w(x)\|$, so $\mathcal{A}_f(x)$ contains the origin in its interior. In order to determine the cost $c(x, y)$, we first find the point by lying on $\partial\mathcal{A}_f(x)$ by solving the quadratic $\|by - \vec{f}_w(x)\|^2 = f_r^2$ for $b \in \mathbb{R}^+$:

$$b = \frac{y \cdot \vec{f}_w(x) + \sqrt{(y \cdot \vec{f}_w(x))^2 - \|y\|^2(\|\vec{f}_w(x)\|^2 - f_r^2)}}{\|y\|^2}.$$

Then since $f(x, y/\|y\|) = b\|y\|$, we have

$$c(x, y) = \frac{\|y\|}{f(x, y/\|y\|)} = \frac{1}{b}.$$

We note that an HJ PDE with the same form of isotropic control and advection component was derived and solved in [26].

To compute an optimal trajectory $\xi(\cdot)$ from a starting location z to x^*, we solve the characteristic ordinary differential equation (ODE)

$$\frac{d\xi(t)}{dt} = f(x, a^*)a^* = \vec{f}_w(x) - f_r\frac{p}{\|p\|} \tag{5.1}$$

with initial condition $\xi(0) = z$, where $a^* \in \operatorname{argmax}_{a\in\mathcal{A}}[(-p \cdot a)f(x, a)]$, $p = D\underline{u}(x)$, and $x = \xi(t)$. Note that this is not gradient descent, because the gradient and the optimal characteristic will not generally align in anisotropic problems. To solve the ODE we used the function `ode23` in Matlab. In order to determine $f(x, a^*)a^*$ in (5.1) for any x, we first approximate $p = D\underline{u}(x)$ as the constant gradient of the linearly interpolated $\underline{u}$ in the grid simplex containing x.

We use a Maubach grid that is additionally refined within a distance $\check{h}\Upsilon$ of $\mathcal{C}$ and the goal x^*. The grid is uniformly refined to level 10 and then refined a further 8 levels near $\mathcal{C}$ and x^*. The resulting grid is shown in Fig. 8 (top-left) and has 38728 nodes. We compute the time-to-reach function $\underline{u}$ for the anisotropic problem (i.e. with the wind) and the isotropic problem (i.e. without the wind) on the nonuniformly refined grid. Solution contours are shown in Figs. 8 (top-right) and (bottom-left), respectively. Optimal trajectories for the anisotropic problem are shown in Fig. 8 (top-right). Notice how trajectories 2 and 4 minimize the distance traveled through regions where the wind is strong and blowing away from the goal. Contrast these trajectories with the straight line optimal trajectories for the isotropic problem in Fig. 8 (bottom-left). We also solve the anisotropic problem on a uniform level 15 Maubach grid of 33025 nodes. Solution contours and optimal trajectories are

[1] To load the wind data into Matlab type `load wind;`. The data is a 3D vector field. We used only the 6th page of the data and ignored any component in the 3rd dimension. In other words, we used `u(:,:,6)` and `v(:,:,6)` for the arrays of wind vector components and `x(:,:,6)` and `y(:,:,6)` for the arrays of spatial coordinates. This discrete data was linearly interpolated to form $\vec{f}_w$.

shown in Fig. 8 (bottom-right). Although the solution contours are smoother away from $\mathcal{C}$ in this uniform grid case, the ODE computation gets stuck near $\mathcal{C}$ for trajectories 2 and 4, likely due to insufficient grid refinement and poor solution quality near $\mathcal{C}$.

6 Conclusion

We presented three simple criteria for the δ-causality of the discretization of a static convex HJ PDE. We showed that with respect to a node, δ-anisotropy-angle-boundedness of a simplex implies δ-negative-gradient-acuteness of the simplex, which in turn implies δ-causality of the equation for the node value update from the simplex. We defined the MAOUM algorithm, which in the initial pass through the grid determines a causal and consistent set of discretized equations by computing stencils for each node which are directionally-complete and anisotropy-angle-bounded. The second pass solves the causal discretized equations in a Dijkstra-like fashion. This pass is essentially FMM with a first-order semi-Lagrangian update and an enlarged stencil. In comparison to AFOUM, the extra computation and storage for the precomputed stencil allows MAOUM to accept node values monotonically and to use a stencil size which adjusts to the local grid refinement. The latter property makes MAOUM efficient for solving static convex HJ PDEs on highly nonuniform grids. This strength of MAOUM was demonstrated in a problem with a homogeneous rectangular speed profile and in a robot navigation problem involving wind and obstacles.

We hope in future work to investigate inhomogeneous problems more fully. The lack of an analytic solution to the seismic imaging problem in section 5.3 confounded our attempts to experimentally measure the difference in local truncation error between AFOUM and MAOUM under inhomogeneity. It would be very useful to develop a benchmark problem with smoothly varying (and perhaps adjustable) inhomogeneity and an analytic solution, in order to better study the effects of stencil and grid adaptation in algorithms such as these.

Although demonstrated only in 2D with Maubach grids, the algorithms described here will work in any dimension on any simplicial grid. However, Algorithm 3 does not generate the smallest possible stencil $\overrightarrow{\mathcal{Y}}(x)$ such that $\mathcal{S}(x)$ satisfies DC and AAB for x; for example, Fig. 2 (bottom-right) shows a $\overrightarrow{\mathcal{Y}}(x)$ that is not minimal. Consequently, Algorithm 3 could be improved. Ideally, it would generate a $\overrightarrow{\mathcal{Y}}(x)$ that is minimal without substantially increasing the computational cost.

MAOUM is best suited to solving problems with a suitably-refined grid which have highly-curved characteristics in some regions of the domain and straight or nearly straight characteristics elsewhere. However, in many cases it is not obvious where the grid should be refined without approximating the solution first. We plan as future work a single-pass method that estimates error in the solution and refines the grid appropriately as the solution is being computed. Another potential future project is to modify MAOUM to be a Dial-like algorithm with bucket width δ, which may be more suitable for parallelization [30]. An open question is how to choose δ for a given problem and grid. This leads to an investigation of the tradeoff between bucket width and stencil size [31]. A larger bucket width would likely make the algorithm more parallelizable, but the accompanying large stencil size would lead to greater truncation error.

Acknowledgements We would like to thank Professor Alexander Vladimirsky for many constructive comments about our work, and in particular for pointing out connections to [31] and the reasons for the larger error in the nonlayered case of the seismic examples in Sect. 5.3 taken from [27]. We would also like to thank Professor Hongkai Zhao for his work as external examiner on the first author's thesis [1].

Appendix: Discretization Proofs

We prove the Propositions of Sect. 2, demonstrating important properties of the discrete equation such as monotonicity and consistency. These proofs are not a novel contribution, but can be helpful for a more complete understanding of the convergence properties of the discretization.

A.1 Monotonicity

We prove Proposition 2.1, showing $\underline{H}(x, \mathcal{S}, \phi, \mu)$ is monotone in $\phi(x_i^s)$.

Proof Let $s \in \mathcal{S}(x)$ and $1 \le i \le n_s$. Since $\check{\phi}(x_i^s) \le \hat{\phi}(x_i^s)$, $\zeta_i \ge 0$ for $1 \le i \le n_s$, and summation is monotone, we have

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

Therefore, since f is positive, τ_s is positive, and max is monotone, we have from (2.4) that $\underline{H}(x, \mathcal{S}, \check{\phi}, \mu) \ge \underline{H}(x, \mathcal{S}, \hat{\phi}, \mu)$. □

A.2 Consistency

We prove Proposition 2.2, showing that the numerical Hamiltonian $\underline{H}$ is consistent with the Hamiltonian H, where consistency is defined by (2.6).

Proof By (2.4), the smoothness of ϕ and the continuity of max and f, (2.3), the DC for y of $\mathcal{S}(y)$, and (1.2)

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

□

A.3 Unique Solution

Lastly, we prove Proposition 2.3, showing the unique solution to $\underline{H}(\mu) = 0$ is given by (2.7).

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

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

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

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

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

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

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

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

must also be the solution to

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

By negating both sides of this equation and rearranging the terms, we get the formula (2.7) for the unique solution $\mu = \tilde{\mu}$. □

References

1. Alton, K.: Dijkstra-like ordered upwind methods for solving static Hamilton-Jacobi equations. PhD thesis, University of British Columbia (2010)
2. Alton, K., Mitchell, I.: Optimal path planning under different norms in continuous state spaces. In: Proceedings of the International Conference on Robotics and Automation, pp. 866–872 (2006)
3. Alton, K., Mitchell, I.M.: Fast marching methods for stationary Hamilton-Jacobi equations with axis-aligned anisotropy. SIAM J. Numer. Anal. **43**, 363–385 (2008)
4. Bak, S., McLaughlin, J., Renzi, D.: Some improvements for the fast sweeping method. SIAM J. Sci. Comput. (2010). doi:10.1137/090749645
5. Barles, G., Souganidis, P.E.: Convergence of approximation schemes for fully nonlinear second order equations. Asymptot. Anal. **4**, 271–283 (1991)
6. Bornemann, F., Rasch, C.: Finite-element discretization of static Hamilton-Jacobi equations based on a local variational principle. Comput. Vis. Sci. **9**, 57–69 (2006)
7. Boue, M., Dupuis, P.: Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control. SIAM J. Numer. Anal. **36**(3), 667–695 (1999)
8. Cecil, T.C., Osher, S.J., Qian, J.: Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. J. Comput. Phys. **213**, 458–473 (2006)
9. Crandall, M.G., Ishii, H., Lions, P.: User's guide to viscosity solutions of second order partial differential equations. Bull. Am. Math. Soc. **27**(1), 1–67 (1992)
10. Cristiani, E.: A fast marching method for Hamilton-Jacobi equations modeling monotone front propagations. J. Sci. Comput. **39**(2), 189–205 (2009)
11. Danielsson, P.-E.: Euclidean distance mapping. Comput. Graph. Image Process. **14**(3), 227–248 (1980)

12. Dial, R.B.: Algorithm 360: shortest-path forest with topological ordering. Commun. ACM **12**, 632–633 (1969)
13. Dijkstra, E.W.: A note on two problems in connection with graphs. Numer. Math. **1**, 269–271 (1959)
14. Kao, C.Y., Osher, S., Tsai, Y.: Fast sweeping methods for static Hamilton-Jacobi equations. SIAM J. Numer. Anal. **42**(6), 2612–2632 (2004–2005)
15. Kiefer, J.: Sequential minimax search for a maximum. Proc. Am. Math. Soc. **4**, 502–506 (1953)
16. Kimmel, R., Sethian, J.A.: Fast marching methods on triangulated domains. Proc. Natl. Acad. Sci. USA **95**, 8341–8435 (1998)
17. Konolige, K.: Saphira robot control system (2011). http://www.ai.sri.com/konolige/saphira/
18. Maubach, J.M.: Local bisection refinement for n-simplicial grids generated by reflection. J. Sci. Comput. **16**(1), 210–227 (1995)
19. Osher, S., Fedkiw, R.P.: Level set methods: An overview and some recent results. J. Comput. Phys. **169**(2), 463–502 (2001)
20. Polymenakos, L.C., Bertsekas, D.P., Tsitsiklis, J.N.: Implementation of efficient algorithms for globally optimal trajectories. IEEE Trans. Autom. Control **43**, 278–283 (1998)
21. Qian, J., Zhang, Y., Zhao, H.: Fast sweeping methods for Eikonal equations on triangulated meshes. SIAM J. Numer. Anal. **45**, 83–107 (2007)
22. Qian, J., Zhang, Y.-T., Zhao, H.-K.: A fast sweeping method for static convex Hamilton-Jacobi equations. J. Sci. Comput. **31**, 237–271 (2007)
23. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. USA **93**, 1591–1595 (1996)
24. Sethian, J.A.: Fast marching methods. SIAM Rev. **41**(2), 199–235 (1999)
25. Sethian, J.A., Vladimirsky, A.: Fast methods for Eikonal and related Hamilton-Jacobi equations on unstructured meshes. Proc. Natl. Acad. Sci. USA **97**(11), 5699–5703 (2000)
26. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations. Proc. Natl. Acad. Sci. USA **98**(20), 11069–11074 (2001)
27. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. SIAM J. Numer. Anal. **41**(1), 323–363 (2003)
28. Tsai, Y.-H.R., Cheng, L.-T., Osher, S., Zhao, H.-K.: Fast sweeping algorithms for a class of Hamilton-Jacobi equations. SIAM J. Numer. Anal. **41**(2), 673–694 (2003)
29. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. In: Proceedings of the 33rd Conference on Decision and Control, pp. 1368–1373 (1994)
30. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Trans. Autom. Control **40**(9), 1528–1538 (1995)
31. Vladimirsky, A.: Label-setting methods for multimode stochastic shortest path problems on graphs. Math. Oper. Res. **33**(4), 821–838 (2008)
32. Zhao, H.: A fast sweeping method for Eikonal equations. Math. Comput. **74**(250), 603–627 (2004)