Journal of Scientific Computing (2021) 88:13
https://doi.org/10.1007/s10915-021-01531-x

A Rotating-Grid Upwind Fast Sweeping Scheme for a Class of Hamilton-Jacobi Equations

Christian Parkinson[1]

Received: 19 May 2020 / Revised: 13 April 2021 / Accepted: 18 May 2021

Abstract
We present a fast sweeping method for a class of Hamilton-Jacobi equations that arise from time-independent problems in optimal control theory. The basic method in two dimensions uses a four point stencil and is extremely simple to implement. We test our basic method against Eikonal equations in different norms, and then suggest a general method for rotating the grid and using additional approximations to the derivatives in different directions in order to more accurately capture characteristic flow. We display the utility of our method by applying it to relevant problems from engineering.

Keywords Fast sweeping method · Steady-state Hamilton-Jacobi equation · Upwind approximation · Rotating grid · Finite difference methods

Mathematics Subject Classification 65N06 · 35F30 · 49L20

1 Introduction

The general Hamilton-Jacobi (HJ) equation in d-dimensions is given by

$$H(x, \nabla\phi(x)) = 0, \quad x \in \Omega \tag{1}$$

where $\Omega \subset \mathbb{R}^d$ and $H : \Omega \times \mathbb{R}^d \to \mathbb{R}$ is the *Hamiltonian* function. Along with Eq. (1), one is often supplied boundary data $\phi(x) = g(x)$ on a set $\Gamma \subset \mathbb{R}^d$, which typically has dimension smaller than d. Common scenarios are $\Gamma = \partial\Omega$ or $\Gamma = \{x_0\}$, a single point. These equations have diverse application in fields including traffic modeling [30], medical imaging [32], path-planning [41], and dynamic visibility [27,34,56] to name a few.

The fast sweeping method is a type of finite difference scheme used to approximate (1). The basic strategy involves discretizing the domain and devising update rules

$$u_i = F_i(u_j|_{j\in N(i)}) \tag{2}$$

✉ Christian Parkinson
chparkin@math.arizona.edu

[1] Department of Mathematics, University of Arizona, Tucson, AZ 85721, USA

that locally approximate the equation at grid nodes i, where $N(i)$ is comprised of the nodes in some neighborhood of node i. Using these update rules, one sweeps through the domain in the Gauss-Seidel manner, iteratively updating the solution values at grid nodes until convergence. As far as this author can discern, the fast sweeping method was first used by Boué and Dupuis [10] and Zhao et al. [63]. Shortly afterwards, there was much work on developing fast sweeping methods for different types of Hamiltonians, and using different strategies for numerical approximation [24–26,57,64]. Subsequent effort was devoted to adapting fast sweeping methods to irregular grids [43,44], improving the accuracy [29,31], and extending them to other equations, such as conservation laws [19,20]. Luo and Zhao [33] provide a nice overview of fast sweeping methods, which we will refer to in Sect. 3.1.

Besides fast sweeping schemes, other grid-based methods used to approximate steady-state HJ equations can be largely divided into two categories. The first category is fast marching methods for monotonically advancing fronts, pioneered by Tsitsiklis [58]. These methods—as well as their generalization to ordered upwind methods—rely on a single-pass, Dijkstra-type algorithm to update the solution value at grid nodes as characteristics flow outward from boundary data [1,2,48–50]. Besides these, Bornemann and Rasch [9] proposed a variational method based on the Hopf-Lax formula. Their approach is to localize the HJ equation to finitely many simplices, approximate the solution with linear elements, and solve a discrete version of the Hopf-Lax formula. Their method is similar in spirit to fast marching methods in that it involves updating nodes in a specific order. However, it relies on a Gauss-Seidel iteration, rather than a single pass update. The second category is time-dependent methods. Osher showed that in many cases one can recast the steady-state HJ equation in a time-dependent manner [36]. There are very general methods which can approximate time-dependent HJ equations at high accuracy, and also allow for non-monotonic flow of information [23,38,51]. More recently, there has been increased interest in algorithms for numerical solutions of HJ equations which break the curse of dimensionality. These typically rely on Hopf-Lax or Lax-Oleinik type formulas for time-dependent HJ equations, and use optimization routines to approximate the solution at individual points [12,17,28]. However, due to the wide applicability and relative ease of both implementation and analysis, fast sweeping methods have remained a popular option for approximating solutions of steady-state HJ equations.

We present an exceedingly simple fast sweeping scheme for a class of Hamilton-Jacobi equations arising from optimal control theory. For simplicity of exposition, we develop our method in two spatial dimensions. The method applies in higher dimensions, though for dimensions $d > 3$, one will encounter the curse of dimensionality. In two dimensions, our most basic method includes a four-point stencil on a rectangular grid, using only the ordinary forward and backward difference operators. We then describe a general method for using rotated coordinates to improve the accuracy of the scheme. We implement our method with special application toward Eikonal equations in different norms, and also mention a few other applications. Because one of the strengths of our method is ease of implementation, we compare it with the Lax-Friedrichs sweeping scheme [25], another easily implementable method.

2 Hamilton-Jacobi Equations in Optimal Control Theory

We will address a specific class of Hamilton-Jacobi equations arising from deterministic optimal control theory. A basic problem in optimal control theory is to choose the best

control plan $\boldsymbol{a} : [0, T] \to A$ to steer a trajectory $\boldsymbol{x}$ obeying

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), \boldsymbol{a}(t), t), \quad 0 < t \leq T, \\ \boldsymbol{x}(0) &= x_0, \end{aligned} \tag{3}$$

to an optimal destination $\boldsymbol{x}(T)$. Here $A \subset \mathbb{R}^m$ is the set of admissible control actions and $f : \mathbb{R}^d \times \mathbb{R}^m \times [0, T] \to \mathbb{R}^d$ is a function describing the dynamics along the trajectory. The "optimal destination" is determined in view of a cost functional

$$C[\boldsymbol{x}(\cdot), \boldsymbol{a}(\cdot)] = g(\boldsymbol{x}(T)) + \int_0^T r(\boldsymbol{x}(t), \boldsymbol{a}(t), t)dt \tag{4}$$

that one wishes to minimize. The function $r : \mathbb{R}^d \times \mathbb{R}^m \times [0, T] \to \mathbb{R}$ accounts for a running cost along the trajectory, and $g : \mathbb{R}^d \to \mathbb{R}$ is the exit cost. While it is not necessary in all cases, we will assume that $r, g \geq 0$, which is common in many applications where cost cannot be negative. To analyze this problem using dynamic programming [7,8], one defines the value function $\phi : \mathbb{R}^d \times [0, T] \to \mathbb{R}$ by

$$\phi(x, t) = \inf_{\boldsymbol{x}(\cdot), \boldsymbol{a}(\cdot)} C_{x,t}[\boldsymbol{x}(\cdot), \boldsymbol{a}(\cdot)] \tag{5}$$

where $C_{x,t}[\boldsymbol{x}(\cdot), \boldsymbol{a}(\cdot)]$ is the remaining cost functional, restricted to trajectories $\boldsymbol{x}$ on the time interval $(t, T]$ and satisfying $\boldsymbol{x}(t) = x$. Thus ϕ is the optimal remaining cost for a trajectory that is at position x at time t. Under mild conditions on the data, this value function is the unique viscosity solution [15] of the terminal value Hamilton-Jacobi-Bellman equation [3,5]

$$\begin{aligned} &\phi_t(x, t) + \inf_{a \in A} \left\{ \langle f(x, a, t), \nabla\phi(x, t) \rangle + r(x, a, t) \right\} = 0, \\ &\phi(x, T) = g(x). \end{aligned} \tag{6}$$

Note that the viscosity solution of (6) should remain non-negative: by (5), ϕ is non-negative whenever r and g are non-negative.

We observe that (6) is of the form (1) if we consider generalized coordinates $\tilde{x} = (t, x)$ and $\nabla_{\tilde{x}} = (\partial_t, \nabla_x)$. In this case $\Omega = \mathbb{R}^d \times [0, T)$ and $\Gamma = \mathbb{R}^d \times \{T\}$. Thus this can be analyzed in the framework of the more general equation (1), but time-dependent equations like (6) are so ubiquitous in application that they are often analyzed independently. Indeed, in their two original papers, Crandall and Lions established the notion of viscosity solutions specifically for time-dependent Hamilton-Jacobi equations [14,15], and later the theory was extended to more general equations; see, for example, [13].

2.1 Our Class of Equations

We restrict our focus to a special class of optimal control problems. We consider the case that the dynamic function f does not depend explicitly on t, and the running cost function r does not depend explicitly on either t or $\boldsymbol{a}(\cdot)$. The removal of the explicit dependence on t is not a particularly stringent condition; this is very natural in many applications. Removing the dependence of r on $\boldsymbol{a}(\cdot)$ is a more serious restriction. For example, this will exclude essentially any problem from mathematical finance where the control variable could represent the fraction of capital one wishes to invest or the amount of goods a company would like to produce [42]. In this case, the cost and profit very explicitly depend on the value of the control variable. However, control problems of our type still have diverse application. Minimal-time path-planning [41] and reach avoid games [65] are two classical problems in applied optimal

control theory that fit into this framework. Otherwise, four of the five examples given by Evans [21, chap. 1] fall into this category. This includes the moon lander problem, optimally stopping a pendulum, and a model for growth of ant colonies originally proposed by Oster and Wilson [39].

When neither f nor r depend on t, one can neglect the time horizon T and formulate a steady-state Hamilton-Jacobi-Bellman equation for the value function. Given that r does not depend on $\boldsymbol{a}(\cdot)$, this takes the form

$$-r(x) = \inf_{a\in A}\{\langle f(x,a), \nabla\phi(x)\rangle\}, \tag{7}$$

or alternately

$$-r(x) = \inf_{a\in A}\left\{\sum_{\ell=1}^{d} f_\ell(x,a)\phi_{x_\ell}(x)\right\} \tag{8}$$

where $x = (x_1,\ldots,x_d)$ and $f(x,a) = (f_1(x,a),\ldots,f_d(x,a))$. We focus on numerical solutions for this equation with boundary data $\phi(x) = g(x)$ on a set $\Gamma \subset \mathbb{R}^d$. For example, in the case of optimal-time path-planning, we will take $\Gamma = \{x_f\}$, where $x_f \in \mathbb{R}^d$ is the desired ending point, and let $\phi(x_f) = 0$. This signifies that paths ending at the desired location incur no exit cost, while other paths are not admissible (i.e., they incur infinite cost).

Many classical Hamilton-Jacobi equations can be expressed in this form. Notably, the Eikonal equation

$$1 = v(x)\,|\nabla\phi(x)| \tag{9}$$

is of this form. The travel-time function for isotropic motion $\dot{\boldsymbol{x}}(t) = v(\boldsymbol{x}(t))\boldsymbol{a}(t)$, where $\boldsymbol{a}(\cdot)$ is a unit vector, is the viscosity solution of this equation, and in the case that $v(x) \equiv 1$, this yields a signed distance function [37]. Assuming $v > 0$, Eq. (9) can be re-written

$$-1/v(x) = \inf_{a\in\mathbb{S}^{d-1}}\left\{a\cdot\nabla\phi\right\} \tag{10}$$

whereupon casting the equation in the form (8) is accomplished by parameterizing the unit sphere $\mathbb{S}^{d-1}$. For example in dimension $d = 2$, we have

$$-1/v(x,y) = \inf_{a\in[0,2\pi)}\left\{\phi_x\cos(a) + \phi_y\sin(a)\right\}, \tag{11}$$

or in dimension $d = 3$,

$$-1/v(x,y,z) = \inf_{a,b}\left\{\phi_x\cos(a)\cos(b) + \phi_y\sin(a)\cos(b) + \phi_z\sin(b)\right\}, \tag{12}$$

where $(a,b) \in [0,2\pi)\times[-\pi/2,\pi/2]$ represent the xy-planar angle and the angle of inclination from the xy-plane, respectively. We return to Eikonal equations when testing our method in Sect. 3.2 and Sect. 4.1.

3 A Basic Fast Sweeping Scheme for (8)

As stated in Sect. 1, for simplicity of exposition, we will describe our fast sweeping scheme in dimension $d = 2$. We consider a rectangular domain $[x_{\min}, x_{\max}]\times[y_{\min}, y_{\max}]$ and a

uniform grid discritization with $I+1$ points in the x-direction, and $J+1$ points in the y-direction. Thus the grid is given by

$$\begin{aligned} x_i &= x_{\min} + i\Delta x, \quad \Delta x = \frac{x_{\max} - x_{\min}}{I}, \quad i = 0, 1, \ldots, I, \\ y_j &= y_{\min} + j\Delta y, \quad \Delta y = \frac{y_{\max} - y_{\min}}{J}, \quad j = 0, 1, \ldots, J. \end{aligned} \tag{13}$$

In two-dimensions, the equation of interest is

$$-r(x, y) = \inf_{a \in A} \Big\{ f_1(x, y, a)\phi_x(x, y) + f_2(x, y, a)\phi_y(x, y) \Big\}. \tag{14}$$

Let ϕ_{ij} be the numerical approximation to $\phi(x_i, y_j)$, and for a fixed $a \in A$, let $f_{\ell,ij}(a) = f_\ell(x_i, y_j, a)$ for $\ell = 1, 2$. Further let

$$\xi_{\ell,ij}(a) = \text{sign}(f_\ell(x_i, y_j, a)), \quad \ell = 1, 2. \tag{15}$$

Then the upwind approximations to the derivatives are given by

$$\begin{aligned} \Big(f_1(x, y, a)\phi_x(x, y)\Big)_{ij} &= \big|f_{1,ij}(a)\big| \frac{\phi_{i+\xi_{1,ij}(a),j} - \phi_{ij}}{\Delta x}, \\ \Big(f_2(x, y, a)\phi_y(x, y)\Big)_{ij} &= \big|f_{2,ij}(a)\big| \frac{\phi_{i,j+\xi_{2,ij}(a)} - \phi_{ij}}{\Delta y}. \end{aligned} \tag{16}$$

Supposing that a is the correct control value at the node (i, j), we can insert these approximations into (14) to arrive at

$$-r_{ij} = \big|f_{1,ij}(a)\big| \frac{\phi_{i+\xi_{1,ij}(a),j} - \phi_{ij}}{\Delta x} + \big|f_{2,ij}(a)\big| \frac{\phi_{i,j+\xi_{2,ij}(a)} - \phi_{ij}}{\Delta y}, \tag{17}$$

where $r_{ij} = r(x_i, y_j)$. Isolating ϕ_{ij}, we see that

$$\phi^*_{ij}(a) = \frac{r_{ij} + \frac{|f_{1,ij}(a)|}{\Delta x}\phi_{i+\xi_{1,ij}(a),j} + \frac{|f_{2,ij}(a)|}{\Delta y}\phi_{i,j+\xi_{2,ij}(a)}}{\frac{|f_{1,ij}(a)|}{\Delta x} + \frac{|f_{2,ij}(a)|}{\Delta y}} \tag{18}$$

is a first-order upwind approximation to Eq. (14), when a is the correct control value at node (i, j). This suggests the fast sweeping scheme detailed in Algorithm 1.

We include some comments regarding the algorithm. First, at each iteration, we sweep through the indices in alternating directions until all combinations of sweeping directions have been performed. Thus each iteration consists of four sweeps; in MATLAB notation:

(1) $i = 1 : I-1, \quad j = 1 : J-1,$
(2) $i = 1 : I-1, \quad j = J-1 : -1 : 1,$
(3) $i = I-1 : -1 : 1, \quad j = J-1 : -1 : 1,$
(4) $i = I-1 : -1 : 1, \quad j = 1 : J-1.$

Generally, in dimension d, there will be 2^d sweeps in each iteration. Second, it is important that we assign $\phi^n_{ij} \leftarrow \phi^{n-1}_{ij}$ at the beginning of each iteration and then operate only with ϕ^n_{ij}. This ensures that sweeping is carried out in the Gauss-Seidel sense: updating values, and then using the most recently updated values to resolve the ensuing values. Third, for the convergence criterion, we use the L^∞-norm so that the iteration halts when $\|\phi^n - \phi^{n-1}\| = \max_{ij} \left|\phi^n_{ij} - \phi^{n-1}_{ij}\right| \le \varepsilon$ for some prescribed tolerance ε, though other criteria could be used. Fourth, the scheme is fully upwind meaning that numerical characteristics flow away from the boundary set Γ. If Γ corresponds to the computational boundary, then information flows

Algorithm 1 A fast sweeping scheme to solve (14)

Initialization: Input boundary data (a function g and set Γ), a grid discretization as in (13), and a small error tolerance $\varepsilon > 0$. Initialize $\phi_{ij}^0 = g(x_i, y_j)$ for the grid nodes corresponding to Γ and $\phi_{ij}^0 = +\infty$ (or some large positive number) for all other grid nodes. Initialize $\phi_{ij}^1 = 0$ at all grid points, and $n = 1$.

while $\|\phi^n - \phi^{n-1}\| > \varepsilon$ **do**

Assign $\phi_{ij}^n \leftarrow \phi_{ij}^{n-1}$ for all (i, j).

for $i = 1$ **to** $I - 1$ **do**
for $j = 1$ **to** $J - 1$ **do**
For each $a \in A$, compute

$$\phi_{ij}^*(a) \leftarrow \frac{r_{ij} + \frac{|f_{1,ij}(a)|}{\Delta x}\phi^n_{i+\xi_{1,ij}(a),j} + \frac{|f_{2,ij}(a)|}{\Delta y}\phi^n_{i,j+\xi_{2,ij}(a)}}{\frac{|f_{1,ij}(a)|}{\Delta x} + \frac{|f_{2,ij}(a)|}{\Delta y}}.$$

Assign $\phi_{ij}^n \leftarrow \min\{\min_a \phi_{ij}^*(a), \phi_{ij}^{n-1}\}$
end for
end for

Repeat the above **for** loops, sweeping in alternating directions until all combinations of sweeping directions have been completed (a total of 4 sweeps).

Assign $n \leftarrow n + 1$

end while

return the values ϕ_{ij}^{end} for all (i, j)

into the domain. If Γ is contained in the computational domain, then characteristics will flow out of the computational boundary. In this case, no special considerations are necessary at the computational boundaries. The values at the boundary nodes will remain large, but will not affect the solution at interior nodes. In this way, our scheme is similar to Godunov-inspired methods such as [57]. In a different approach, Kao et al. [25] devise a sweeping method with a Lax-Friedrichs Hamiltonian, wherein added numerical diffusion will cause information to seep into the domain from the computational boundary, requiring special consideration. We will discuss the Lax-Friedrichs sweeping method in more detail later.

Perhaps the most important notes regard the minimization over $a \in A$, which takes place at each grid point in each sweep. A single iteration in two dimensions requires this minimization to be resolved roughly $4IJ$ times. Because of this, the shape of A is somewhat crucial to the algorithm. For example, in the Eikonal equation, we have $A = \mathbb{S}^1$, meaning this optimization is performed over a continuous set. One can either discretize the set and choose from finitely many values, or introduce an optimization routine of their choosing. Either way, this is likely to represent the largest computational burden. The minimization problem is nontrivial due to the switching between the values $\phi_{i\pm1,j}$ and $\phi_{i,j\pm1}$ for different values of $a \in A$. In some cases, one may be able to isolate regions of A where certain values are used, and thus simplify the problem. However, it is difficult to guarantee efficiency in general, since the computational complexity will depend on the fineness of the grid and resolution of the minimization problem.

The algorithm performs extraordinarily well when A is finite. For example, this occurs in bang-bang control problems, where the optimal controls switch between finitely many control values [52]. In this case, the minimization problem reduces to checking finitely many points, and the complexity depends only on the grid discretization. One application of this is in kinematic models for simple self-driving cars [18,46]. Takei and Tsai were the first to analyze this problem in the Hamilton-Jacobi setting [54,55], and they used a sweeping scheme much like ours. We will return to the example of self-driving cars in Sect. 5 where we display the utility of our method.

3.1 Upwinding, Monotonicity and Convergence

Luo and Zhao [33] discuss and analyze fast sweeping methods in some generality. In particular, they consider (1) with a Hamiltonian H that is

(i) continuous on $\Omega \times \mathbb{R}^n$,
(ii) convex and coercive in $\nabla\phi$,
(iii) compatible, in that $H(x, 0) \leq 0$ for $x \in \Omega$.

Under these conditions and some mild conditions on the boundary data g, they prove that if a fast sweeping scheme is consistent, monotone, and obeys a causality condition, then the approximate solution produced by the scheme will converge to the viscosity solution of the Hamilton-Jacobi equation under grid refinement.

An annoying but necessary facet of the theory of viscosity solutions is that orientation matters. Formally, the viscosity solution of $H(x, \nabla\phi(x)) = 0$ is the negative of the viscosity of $-H(x, \nabla\phi(x)) = 0$. Our orientation is reversed from that in [33] but modulo some sign changes and inequality flips, the analysis is the same. Our scheme is consistent to first order, as can be shown by a simple Taylor expansion. In our case, the monotonicity requirement is trivially satisfied since the update rule (18) is clearly non-decreasing in the values at the surrounding grid nodes. The causality condition states in essence that the characteristic flowing into grid node (i, j) is contained in the polygon formed by the nodes used for the finite difference approximations at (i, j). This is illustrated in Fig. 1, where the characteristic curve (blue) enters from the positive-x and positive-y direction, specifying that one should use nodes (i, j), $(i+1, j)$, $(i, j+1)$ to approximate $(\nabla\phi)_{ij}$. For us, the causality condition corresponds exactly to the upwind approximations (16). Note that because of the negative sign in the equation, the characteristic direction at (x, y) is $-f(x, y, a)$ when a is the correct control value at (x, y). Thus our scheme fits into their framework, and we have convergence to the viscosity solution of (7) as the grid parameters go to zero.

Determining the order of covergence is subtle. Classical proofs of convergence for numerical solutions of Hamilton-Jacobi equations depend not only on the order of local truncation error, but also on the regularity of the viscosity solution [4,6,53]. Typically one can guarantee convergence at order no less than $1/2$ when the scheme is consistent at order 1. However, one often sees full first-order convergence in regions where the solution is smooth [33], and in some cases, one can achieve higher order accuracy using techniques such as ENO or WENO schemes [23,38,51,62], though the application of these concepts to fast sweeping methods presents some challenges. We discuss this further in Sect. 3.3.

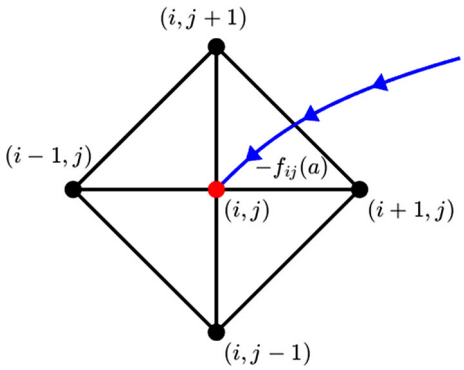


Fig. 1 The causality condition specifies that the nodes used to approximate $(\nabla\phi)_{ij}$ form a polygon containing the characteristic (blue) flowing into (i, j). Here, one would use nodes (i, j), $(i+1, j)$, $(i, j+1)$. The characteristic direction is given by $-f_{ij}(a)$ if a is the correct control value at the grid node (Color figure online)

3.2 Application of the Basic Method to Eikonal Equations

To empirically study error and convergence, we test our method on three different Eikonal equations:

$$1 = \|\nabla\phi(x)\|_p \tag{19}$$

where $p = 1, 2, \infty$. Given the boundary data $\phi(0) = 0$, we see that the unique (positive) viscosity solution of (19) is $\phi_p(x) = \|x\|_{p'}$ where $\frac{1}{p} + \frac{1}{p'} = 1$. This fact can be intuited from the ensuing optimal control problem, and essentially follows from the dual definition of the norm:

$$\|z\|_p = \sup_{\|a\|_{p'} \le 1} \langle z, a\rangle. \tag{20}$$

However, proving this in full generality is surprisingly intricate. A discussion of such equations is included in [35], and a full analysis is given in [11].

Each of these equations is solved by travel time function for a minimal-time path-planning problem of the form above. Indeed, consider the equation of motion

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{a}(t), \quad \boldsymbol{a}(\cdot) \in B_1^{(p')}, \tag{21}$$

where $B_1^{(p')}$ is the unit ball in the p'-norm (centered at the origin). If we pair this equation with the cost functional

$$C[\boldsymbol{x}(\cdot), \boldsymbol{a}(\cdot)] = \iota_0(\boldsymbol{x}(T)) + \int_0^T 1\, dt \tag{22}$$

where ι_0 is the convex indicator of the origin (0 at the origin; $+\infty$ elsewhere) and allow for infinite horizon time, then the Hamilton-Jacobi-Bellman equation for the value function is the p-norm Eikonal equation (19), and the optimal control plan steers the trajectory to the origin in the minimal possible time, where distance from the origin is computed in the p'-norm. In particular, since the unit ball has finitely many extreme points in the case that $p' = \infty$ or $p' = 1$, this leads to a bang-bang control problem for $p = 1$ or $p = \infty$.

In two-dimensions, Eq. (11) shows that the 2-norm Eikonal equation can be written in the form (14). We can write the other equations in this form as well. For $p = 1$, we have

$$-1 = \inf_{a_1, a_2 \in \{\pm 1\}} \Big\{ a_1\phi_x(x, y) + a_2\phi_y(x, y) \Big\} \tag{23}$$

and for $p = \infty$, we have

$$-1 = \inf_{a \in \{\pm e_1, \pm e_2\}} \Big\{ a_1 \phi_x(x, y) + a_2 \phi_y(x, y) \Big\}, \tag{24}$$

where in the latter equation, e_1, e_2 are the standard basis vectors, and $a = (a_1, a_2)$.

We would like derive the specific update formula (18) for each of these cases. For the ordinary Eikonal equation in the 2-norm, we find

$$\phi^{*,2}_{ij}(a) = \frac{1 + \frac{|\cos(a)|}{\Delta x} \phi^{n,2}_{i+\text{sign}(\cos(a)),j} + \frac{|\sin(a)|}{\Delta y} \phi^{n,2}_{i,j+\text{sign}(\sin(a))}}{\frac{|\cos(a)|}{\Delta x} + \frac{|\sin(a)|}{\Delta y}}. \tag{25}$$

and use the update $\phi^{n,2}_{ij} = \min\{\min_{a \in [0, 2\pi)} \phi^{*,2}_{ij}(a), \phi^{n-1,2}_{ij}\}$. To use this update, we will need to resolve the minimization over $a \in [0, 2\pi)$. To do so, we simply sample $a = 2\pi k/K$ for $k = 0, \ldots, K-1$ and choose the minimum from these finitely many points. In our tests, we fix $K = 400$. This will incur some small error. We discuss this briefly below.

For the 1-norm and ∞-norm equations, we can explictly write the update rule by considering all possible combinations of control variables. For the case $p = 1$, we have

$$\begin{aligned} \phi^{n,1}_{ij} = \min \Bigg\{ \phi^{n-1,1}_{ij}, & \frac{1 + \frac{1}{\Delta x}\phi^{n,1}_{i+1,j} + \frac{1}{\Delta y}\phi^{n,1}_{i,j+1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}}, \frac{1 + \frac{1}{\Delta x}\phi^{n,1}_{i-1,j} + \frac{1}{\Delta y}\phi^{n,1}_{i,j+1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}}, \\ & \frac{1 + \frac{1}{\Delta x}\phi^{n,1}_{i+1,j} + \frac{1}{\Delta y}\phi^{n,1}_{i,j-1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}}, \frac{1 + \frac{1}{\Delta x}\phi^{n,1}_{i-1,j} + \frac{1}{\Delta y}\phi^{n,1}_{i,j-1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}} \Bigg\}. \end{aligned} \tag{26}$$

In the $p = \infty$ case, the update is even simpler since one of a_1, a_2 in (24) is zero. Plugging the values into the general update formula (18) and clearing the denominator yields

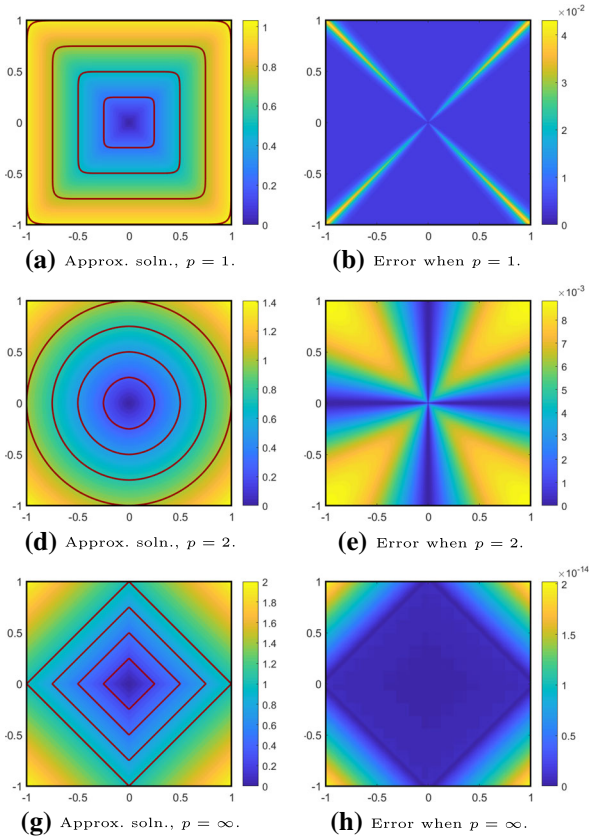
$$\phi^{n,\infty}_{ij} = \min \Big\{ \phi^{n-1,\infty}_{ij},\ \Delta x + \phi^{n,\infty}_{i+1,j},\ \Delta x + \phi^{n,\infty}_{i-1,j},\ \Delta y + \phi^{n,\infty}_{i,j+1},\ \Delta y + \phi^{n,\infty}_{i,j-1} \Big\}. \tag{27}$$

We note that (27) is perfectly satisfied by the exact solution $\phi_\infty(x, y) = \|(x, y)\|_1 = |x| + |y|$, and thus when $p = \infty$, our scheme will solve the equation exactly, so long as the origin is a grid node. Otherwise, the error in the approximation will only depend on the distance from the origin to the nearest grid node in each direction.

Using these update rules, and the boundary condition $\phi(0, 0) = 0$, we simulated Eq. (19) for $p = 1, 2, \infty$. The results are included in Fig. 2. Specifically, results for $p = 1$ are included in Fig. 2a–c; $p = 2$ in Fig. 2d–f; and $p = \infty$ in Fig. 2g–i. Recall again the exact solution $\phi_p(x, y) = \|(x, y)\|_{p'}$. The left most figure in each column shows contour plots of the approximate solutions $[-1, 1] \times [-1, 1]$ with a 401×401 grid, along with level sets of the approximate solutions. The middle figure in each column shows a contour plot of the error in the approximation. The right most figure includes the convergence table in each case. We note that there is a different scale in each plot.

When $p = 1$, the level sets should be perfect squares since these are balls in the ∞-norm. At the corners of those squares, the ordinary forward and backward difference operators cannot capture the sharp edges, which leads to some rounding off. Because of this, the error is large along the lines $y = \pm x$, and the order of convergence is roughly $1/2$; the minimal convergence rate guaranteed by the classical theory [4,53].

When $p = 2$, the maximum error is less than in the $p = 1$ case, and the error itself is more evenly spread throughout the entirety of each quadrant, rather than being focused along specific lines. The convergence rate here is roughly $3/4$, showing improved convergence behavior compared with the $p = 1$ case. An interesting note here is that along the lines $x = 0$



(a) Approx. soln., $p = 1$. **(b)** Error when $p = 1$.

I, J	L^∞ Err.	Conv.
50	1.4057e-01	—
100	9.3988e-02	0.5807
200	6.3636e-02	0.5626
400	4.3544e-02	0.5474
800	3.0049e-02	0.5352
1600	2.0872e-02	0.5257

(c) Conv. table for $p = 1$.

(d) Approx. soln., $p = 2$. **(e)** Error when $p = 2$.

I, J	L^∞ Err.	Conv.
50	4.3754e-02	—
100	2.6310e-02	0.7338
200	1.5464e-02	0.7666
400	8.9201e-03	0.7938
800	5.0668e-03	0.8160
1600	2.8431e-03	0.8336

(f) Conv. table for $p = 2$.

(g) Approx. soln., $p = \infty$. **(h)** Error when $p = \infty$.

I, J	L^∞ Err.	Conv.
50	1.7764e-15	—
100	1.7764e-15	0.0000
200	1.7764e-15	0.0000
400	2.0428e-14	-3.5236
800	4.2633e-14	-1.0614
1600	4.2633e-14	0.0000

(i) Conv. table for $p = \infty$.

Fig. 2 Approximation of $\|\nabla\phi\|_p = 1$ using our fast sweeping method. Plots display results from the 401×401 grid. Red lines are level sets of the solution (Color figure online)

and $y = 0$, the error is effectively zero. This is because the finite difference approximations are focused in those directions, and the cross sections of the exact solution in those directions are linear rays increasing outward from the origin. Thus, for example, when $x > 0$, the exact solution satisfies $\phi_2(x + \Delta x, 0) = \Delta x + \phi_2(x, 0)$, and our discretization captures this relationship with no error. We will return to this line of thought momentarily. Before doing so, we make a further remark regarding the discretization of the control set. Recall, the update rule for the 2-norm Eikonal equation requires that we resolve a minimization problem over $[0, 2\pi)$, and to do so we simply discretized the interval into $K = 400$ points and chose the minimum from the discrete set. We found empirically that error produced by approximating the control set is smaller than the error in the discrete derivative approximations. To test this, we instead resolved the minimization to a tolerance of 10^{-10} using built-in optimization routines in MATLAB. For a 400×400 grid, the approximate solution found using the exact minimization differed from that found using discrete minimization with $K = 400$ by only 1.4×10^{-5}, whereas the error between the exact solution and each of the approximate solutions was roughly 8.9×10^{-3}. It bears mentioning that when finding the exact minimum at every point, the algorithm required roughly 150 times the CPU time to resolve the solution. In general, as long as the minimization problem is solved so that the approximation using the

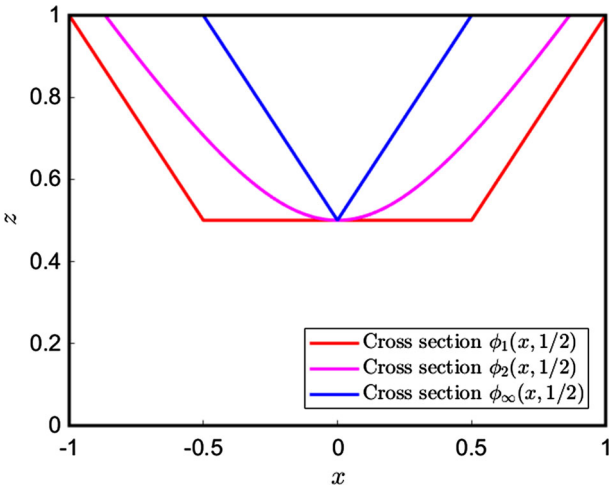


Fig. 3 Horizontal cross section of $\phi_1, \phi_2, \phi_\infty$ at $y = 1/2$

exact minimum and the approximation using an approximate minimum differ by no more than $O(\Delta x, \Delta y)$, then the approximation of the control set will not ruin convergence. Beyond that, one must choose how to balance accuracy and efficiency, as well as ease of implementation. To this last point, one of the strengths of this method *is* the ease of implementation, which is why it is particularly suited to problems where the minimization can be resolved explicitly (for example, bang-bang problems such as the 1-norm or ∞-norm Eikonal equation or the kinematics of the self-driving car presented in Sect. 5).

When $p = \infty$, we noted earlier that our scheme should be exact. Indeed, we see that the level sets of the approximate solution are sharp-edged diamonds, exactly mirroring the level sets of $\phi_\infty(x, y) = |x| + |y|$. In this case, the error is near machine-ε, and thus the convergence table is not informative.

We remarked about the low error along the lines $x = 0$ and $y = 0$ in the $p = 2$ case, and the relationship between this low error and the cross sections of the exact solution along those lines. This remark very closely relates to the improved order of convergence for larger p. As p increases (and thus p' decreases), the cross sections of the exact solution $\phi_p(x) = \|x\|_{p'}$ in the vertical or horizontal directions more closely resemble the absolute value function, and thus can be captured more accurately by the finite difference approximations. This is seen in Fig. 3, where we have plotted horizontal cross sections of ϕ_1, ϕ_2 and ϕ_∞ at level $y = 1/2$. For $\phi_\infty(x, y) = |x| + |y|$, this cross section is exactly $|x| + 1/2$. For $\phi_2(x, y) = \sqrt{x^2 + y^2}$, the cross section is a smooth curve, which cannot be captured perfectly by our discretization, but is better approximated than the cross section of $\phi_1(x, y) = \max\{|x|, |y|\}$, which has two kinks. The accuracy of the method depends on how well these cross sections can be approximated, since any error in these approximations will propagate to other regions.

With this in mind, we note that for $\phi_1(x, y) = \max\{|x|, |y|\}$, while the cross sections in the horizontal and vertical direction have these two kinks, the cross sections in the diagonal directions $y = x_0 \pm x$ will look like absolute value functions. If we used first-order approximations to $\nabla\phi_1$ along these diagonals, we would perfectly capture these cross sections, and thus reconstruct the solution exactly. This suggests that we should rotate the grid and consider alternative approximations to $\nabla\phi_1$. Section 4 develops this idea.

3.3 Increasing Accuracy with WENO Approximations

We noted earlier that in some cases, one can increase the order of accuracy using (Weighted) Essentially Non-Oscillatory (WENO) schemes. The philosophy of ENO and WENO schemes—pioneered by Osher, Shu and Jiang, among others [23,38,51]—is to use multiple higher order approximations of ϕ_x and ϕ_y, and deftly combine the approximations so as to minimize oscillations in the numerical solution near kinks. These methods were originally developed for time-dependent Hamilton-Jacobi equation, but have since been adapted to fast sweeping methods. We demonstrate the application of the third-order WENO approximations to our method, following the work of Zhang et al. [62]. One could use higher order WENO approximations if desired.

The third-order WENO approximations to ϕ_x are given by

$$\begin{aligned}(\phi_x)_{ij}^{+} &= (1-w_x^{+})\left(\frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x}\right)+w_x^{+}\left(\frac{-\phi_{i+2,j}+4\phi_{i+1,j}-3\phi_{ij}}{2\Delta x}\right),\\ (\phi_x)_{ij}^{-} &= (1-w_x^{-})\left(\frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x}\right)+w_x^{-}\left(\frac{\phi_{i-2,j}-4\phi_{i-1,j}+3\phi_{ij}}{2\Delta x}\right),\end{aligned} \tag{28}$$

where the weights w_x^{+} and w_x^{-} are given by

$$\begin{aligned} w_x^{+} &= \frac{1}{1+2(r_x^{+})^2}, & r_x^{+} &= \frac{\varepsilon+(\phi_{i+2,j}-2\phi_{i+1,j}+\phi_{ij})^2}{\varepsilon+(\phi_{i+1,j}-2\phi_{ij}+\phi_{i-1,j})^2},\\ w_x^{-} &= \frac{1}{1+2(r_x^{-})^2}, & r_x^{-} &= \frac{\varepsilon+(\phi_{i-2,j}-2\phi_{i-1,j}+\phi_{ij})^2}{\varepsilon+(\phi_{i-1,j}-2\phi_{ij}+\phi_{i+1,j})^2}.\end{aligned} \tag{29}$$

Here ε is some small number which we fix at 10^{-6}. We define $(\phi_y)_{ij}^{+}$ and $(\phi_y)_{ij}^{-}$ analogously.

Notice that each of the divided differences in (28) is a second-order approximation to ϕ_x. The weighted averages—which favor the less oscillatory approximations—ensure that $(\phi_x)_{ij}^{+}$ and $(\phi_x)_{ij}^{-}$ are third-order approximations to ϕ_x in regions where ϕ is smooth. For a derivation and discussion of these formulas, see [51] and the references therein.

The question then becomes: how to include these approximations in a fast sweeping scheme? If we simply replace the finite difference approximations in (16) with $(\phi_x)_{ij}^{+}$ or $(\phi_x)_{ij}^{-}$ as appropriate, then we will not be able to isolate ϕ_{ij} and arrive at a simple update rule of the form (18). The idea presented by Zhang et al. [62] is to start from the update rule itself. Note that the update rule (18) gives ϕ_{ij} as a function of $\phi_{i\pm1,j}$ and $\phi_{i,j\pm1}$. A finite difference approximation exploits the formal relationship $\phi(x\pm\Delta x,y)\approx\phi(x,y)\pm\Delta x\phi_x(x,y)$. Thus to arrive at a higher order approximation of the form (18), we can replace $\phi_{i+1,j}$ with $\phi_{ij}+\Delta x(\phi_x)_{ij}^{+}$, and replace $\phi_{i-1,j}$ with $\phi_{ij}-\Delta x(\phi_x)_{ij}^{-}$, and similarly for $\phi_{i,j\pm1}$. Doing so results in the update rule

$$\phi_{ij}^{*}(a)=\frac{r_{ij}+\frac{|f_{1,ij}(a)|}{\Delta x}\left(\phi_{ij}+\Delta x\xi_{1,ij}(a)(\phi_x)_{ij}^{\xi_{1,ij}(a)}\right)+\frac{|f_{2,ij}(a)|}{\Delta y}\left(\phi_{ij}+\Delta y\xi_{2,ij}(a)(\phi_y)_{ij}^{\xi_{2,ij}(a)}\right)}{\frac{|f_{1,ij}(a)|}{\Delta x}+\frac{|f_{2,ij}(a)|}{\Delta y}}. \tag{30}$$

Using this update rule in Algorithm 1 yields a higher order approximation of (8).

Formally, the approximation is third-order accurate when the solution $\phi(x,y)$ is smooth. In practice, the convergence can be corrupted by non-smoothness of the solution, and by the non-monotone nature of higher order approximations, which affects the numerical causality.

Table 1 Error in solution of $\|\nabla\phi\|_2 = 1$ when using the Basic Method with (a) first-order approximations or (b) third-order WENO approximations

I, J	L^∞ Err.	L^∞ Conv.	L^1 Err.	L^1 Conv.
(a) Convergence table with first-order approximations to $\nabla\phi$				
50	4.3754e−02	–	9.7606e−02	–
100	2.6310e−02	0.7338	5.9553e−02	0.7128
200	1.5464e−02	0.7666	3.5451e−02	0.7484
400	8.9201e−03	0.7938	2.0691e−02	0.7768
(b) Convergence table with third-order WENO approximations to $\nabla\phi$				
50	9.0508e−03	–	2.0426e−02	–
100	4.4930e−03	1.0104	8.7373e−03	1.2252
200	2.2253e−03	1.0137	3.8868e−03	1.1686
400	1.0668e−03	1.0607	1.9013e−03	1.0316

Because of this last concern, when using the WENO approximations, it is crucial to seed the Gauss-Seidel iteration with a good initial guess ϕ^0_{ij}, rather than simply setting $\phi^0_{ij} = g_{ij}$ near the prescribed boundaries, and $\phi^0_{ij} = +\infty$ elsewhere. If one uses this crude initialization, it is easily checked in simple examples that (30) will not correctly propagate information from the boundaries. We suggest first running the basic scheme with the ordinary first-order approximations, and using the resulting solution to initialize the iteration that uses the WENO approximations.

We have carried out the implementation for two example problems. Both are of the form

$$r(x, y) = \|\nabla\phi(x, y)\|_2, \qquad \phi(0, 0) = 0. \tag{31}$$

In the first, we take $r(x, y) = 1$ so that it is the same 2-norm Eikonal equation as above, and the solution is given by $\phi(x, y) = \sqrt{x^2 + y^2}$, which has a kink at the origin. In the second, we take $r(x, y) = \sqrt{x^2 + y^2}$, in which case the exact solution is $\phi(x, y) = (x^2 + y^2)/2$ which is smooth throughout the domain. The results are summarized in Tables 1 and 2. In this case we report both the L^∞ and L^1 errors. In some cases, the L^1 error is more appropriate for evaluating the performance of WENO schemes, since the most significant errors can propagate along very small sets, whereas error remains small in the majority of the domain [62]. In Table 1, we see that for the Eikonal equation $\|\nabla\phi\|_2 = 1$, the non-smoothness of the solutions corrupts the effects of the WENO approximations, and while the errors are smaller and convergence rate is improved, we do not nearly have third-order convergence. By contrast, in Table 2 when the solution remains smooth, we do see a greatly improved rate of convergence which is near third-order as the grid refines.

4 A Rotating-Grid Fast Sweeping Scheme

In this section, we would like to append the basic algorithm with additional approximations to the gradient $\nabla\phi$ in directions that are not vertical and horizontal (with respect to the rectangular domain). In doing so, we can increase accuracy while maintaining a monotone scheme, since we do not use higher order approximations to the derivatives.

Table 2 Error in solution of $\|\nabla\phi\|_2 = \sqrt{x^2 + y^2}$ when using the Basic Method with (a) first-order approximations or (b) third-order WENO approximations

I, J	L^∞ Err.	L^∞ Conv.	L^1 Err.	L^1 Conv.
(a) Convergence table with first-order approximations to $\nabla\phi$				
50	4.0010e−02	–	8.0016e−02	–
100	2.0009e−02	0.9997	4.0014e−02	0.9998
200	1.0010e−02	0.9992	2.0014e−02	0.9995
400	5.0103e−03	0.9985	1.0014e−02	0.9990
(b) Convergence table with third-order WENO approximations to $\nabla\phi$				
50	2.3922e−03	–	5.4938e−03	–
100	1.1609e−03	1.0431	2.3126e−03	1.2483
200	1.5113e−04	2.9413	3.7584e−04	2.6213
400	3.9126e−05	1.9496	6.0658e−05	2.6314

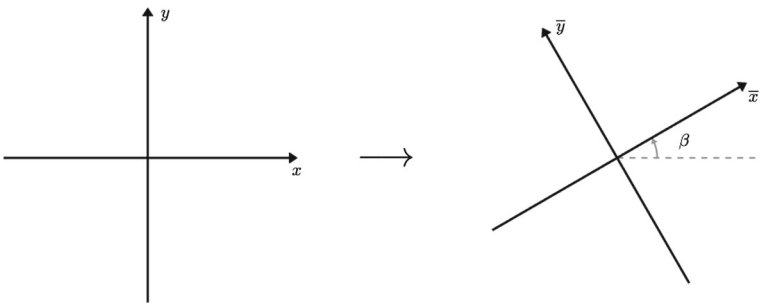


Fig. 4 Cartesian coordinates rotated by $\beta \in (0, \pi/2)$ in the counterclockwise direction

In order to accomplish this, we must first recast Eq. (14) in new coordinates $(\overline{x}, \overline{y})$, rotated versions of the standard Cartesian coordinates. Again, we describe this procedure in two dimensions. Here the extension to higher dimensions is not as straightforward but can still be accomplished in a somewhat principled, if tedious, manner. We discuss the three-dimensional implementation in Appendix A.

Suppose that $(\overline{x}, \overline{y})$ are the typical Cartesian coordinates, rotated counterclockwise by an angle $\beta \in (0, \pi/2)$, as pictured in Fig. 4. Note that it is sufficient to consider this range of angles; rotations by larger angles results in the same transformation up to renaming coordinates and flipping positive and negative directions. One easily verifies the relationship

$$\begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} = \begin{pmatrix} \cos(\beta) & \sin(\beta) \\ -\sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \longleftrightarrow \quad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}. \tag{32}$$

Thus the derivatives in the (x, y) directions can be expressed

$$\begin{aligned} \phi_x &= \frac{\partial \overline{x}}{\partial x}\,\phi_{\overline{x}} + \frac{\partial \overline{y}}{\partial x}\,\phi_{\overline{y}} = \cos(\beta)\phi_{\overline{x}} - \sin(\beta)\phi_{\overline{y}}, \\ \phi_y &= \frac{\partial \overline{x}}{\partial y}\,\phi_{\overline{x}} + \frac{\partial \overline{y}}{\partial y}\,\phi_{\overline{y}} = \sin(\beta)\phi_{\overline{x}} + \cos(\beta)\phi_{\overline{y}}. \end{aligned} \tag{33}$$

Inserting these representations into (14) yields

$$-r(\overline{x},\overline{y}) = \inf_{a\in A}\Big\{[\cos(\beta)f_1(\overline{x},\overline{y},a)+\sin(\beta)f_2(\overline{x},\overline{y},a)]\phi_{\overline{x}}(\overline{x},\overline{y}) + [\cos(\beta)f_2(\overline{x},\overline{y},a)-\sin(\beta)f_1(\overline{x},\overline{y},a)]\phi_{\overline{x}}(\overline{x},\overline{y})\Big\}. \tag{34}$$

Defining

$$\begin{aligned}\overline{f}_1(\overline{x},\overline{y},a) &= \cos(\beta)f_1(\overline{x},\overline{y},a)+\sin(\beta)f_2(\overline{x},\overline{y},a),\\ \overline{f}_2(\overline{x},\overline{y},a) &= \cos(\beta)f_2(\overline{x},\overline{y},a)-\sin(\beta)f_1(\overline{x},\overline{y},a),\end{aligned} \tag{35}$$

we arrive at

$$-r(\overline{x},\overline{y}) = \inf_{a\in A}\left\{\overline{f}_1(\overline{x},\overline{y},a)\phi_{\overline{x}}(\overline{x},\overline{y})+\overline{f}_2(\overline{x},\overline{y},a)\phi_{\overline{y}}(\overline{x},\overline{y})\right\}. \tag{36}$$

The idea is now to write the upwind finite difference approximations in the directions of $(\overline{x},\overline{y})$. Doing so shows that

$$\phi(\overline{x},\overline{y}) = \frac{r(\overline{x},\overline{y}) + \frac{|\overline{f}_1(\overline{x},\overline{y},a)|}{\Delta\overline{x}}\phi(\overline{x}+\overline{\xi}_1\Delta\overline{x},\overline{y}) + \frac{|\overline{f}_2(\overline{x},\overline{y},a)|}{\Delta\overline{y}}\phi(\overline{x},\overline{y}+\overline{\xi}_2\Delta\overline{y})}{\frac{|\overline{f}_1(\overline{x},\overline{y},a)|}{\Delta\overline{x}}+\frac{|\overline{f}_2(\overline{x},\overline{y},a)|}{\Delta\overline{y}}} \tag{37}$$

is a first-order, upwind approximation to (36) at the point $(\overline{x},\overline{y})$ when a is the correct control value, and $\overline{\xi}_\ell = \text{sign}(\overline{f}_\ell(\overline{x},\overline{y},a))$. Thus one could add this approximation into the sweeping scheme and use the update rule

$$\phi_{ij}^n = \min\left\{\phi_{ij}^{n-1}, \min_{a\in A}\phi_{ij}^*(a), \min_{a\in A}\overline{\phi}_{ij}^*(a)\right\}, \tag{38}$$

where $\overline{\phi}_{ij}^*(a)$ is computed from (37). However, this raises the question of how to evaluate (37) on the grid, since for example, $(\overline{x}\pm\Delta\overline{x},\overline{y})$ may not be grid nodes.

Rotated finite differences are extensively used in computational wave mechanics. So-called rotated-staggered-grid methods were introduced by Saenger et al. [47] and are still being developed and improved today [16,22,45,59,61]. The philosophy of these methods is the same: using finite differences in multiple orientations will more accurately capture the upwind direction. Their strategy is to define a new grid corresponding to the points $(\overline{x},\overline{y})$ and keep track of solution values ϕ_{ij} and $\overline{\phi}_{ij}$ separately, while using both sets of values to approximate the derivatives on both grids. To this author's knowledge, the idea of fixing a square grid and computing approximations to $\nabla\phi$ in different directions has not been widely used in the context of fast sweeping methods. Takei et al. [54] suggest using approximations along different directions. However, in their case, the upwind direction is fixed (in analogy to our setup, they have f_1, f_2 independent of a) which simplifies the matter.

We would like to maintain a single grid (x_i, y_j). To do so, one could interpolate values of ϕ_{ij} to off grid values, and compute the upwind approximation in any direction β. This would be computationally expensive since, in order to maintain the Gauss-Seidel sweeping, this interpolation will need to be performed separately for every (i, j) using the newest updated values. Alternatively, we can choose particular values of β and $\Delta\overline{x}$, $\Delta\overline{y}$ such that the points $(\overline{x}\pm\Delta\overline{x},\overline{y})$, $(\overline{x},\overline{y}\pm\Delta\overline{y})$ fall on the grid.

Explicitly, rather than choosing β and the rotated grid parameters $(\Delta\overline{x},\Delta\overline{y})$, we choose natural numbers $(\hat{\imath},\hat{\jmath})$, and define $\beta = \arctan(\hat{\jmath}/\hat{\imath})$. We then let this β determine the grid rotation, so that the positive $\hat{x}$-direction is parallel with the vector $(\hat{\imath},\hat{\jmath})$. This is pictured in Fig. 5. Here we have used $(\hat{\imath},\hat{\jmath}) = (2, 1)$. As pictured, the nodes used to approximate $\phi_{\overline{x}}$ at

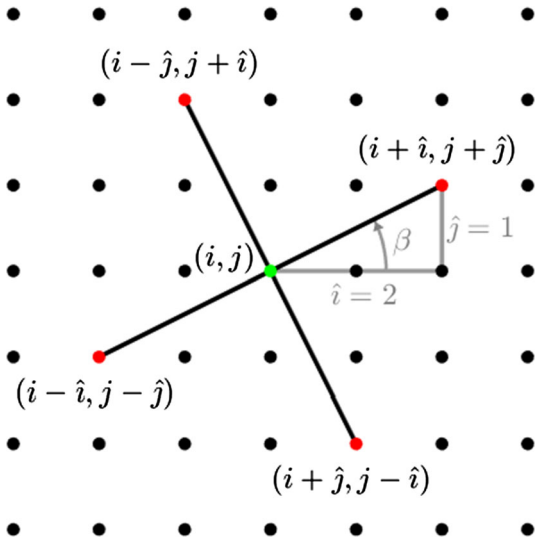


Fig. 5 Rotated stencil at (i, j) using the rotation determined by $(\hat{\imath}, \hat{\jmath}) = (2, 1)$

(i, j) will be $\{(i, j), (i+2, j+1)\}$ for the forward approximation, and $\{(i-2, j-1), (i, j)\}$ for the backward approximation. Similarly, the nodes used to approximate $\phi_{\overline{y}}$ at (i, j) will be $\{(i, j), (i-1, j+2)\}$ for the forward approximation and $\{(i+1, j-2), (i, j)\}$ for the backward approximation.

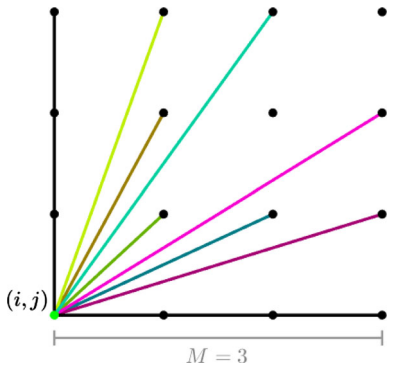
We note that as described, this will only work on a square grid ($\Delta x = \Delta y$). The extension to a non-square grid is a bit more complicated. In that case, there would be two rotation angles that rotate the x-axis and y-axis differently, and thus the resulting coordinate system would no longer be orthogonal. For the remainder of this document, we will assume that $\Delta x = \Delta y$ so that the rotation method works as described.

With these parameters $(\hat{\imath}, \hat{\jmath})$ determining the rotation, we define the new grid discretization parameter $\Delta s = \sqrt{(\hat{\imath}\Delta x)^2 + (\hat{\jmath}\Delta y)^2}$. Note that this Δs will take the place of $\Delta\overline{x}$, $\Delta\overline{y}$ in the case of a square grid. Thus we can translate Eq. (37) onto the grid:

$$\overline{\phi}^*_{ij}(a) = \frac{r_{ij}\Delta s + \left|\overline{f}_{1,ij}(a)\right| \phi_{i+\overline{\xi}_{1,ij}(a)\hat{\imath}, j+\overline{\xi}_{1,ij}(a)\hat{\jmath}} + \left|\overline{f}_{2,ij}(a)\right| \phi_{i-\overline{\xi}_{2,ij}(a)\hat{\jmath}, j+\overline{\xi}_{2,ij}(a)\hat{\imath}}}{\left|\overline{f}_{1,ij}(a)\right| + \left|\overline{f}_{2,ij}(a)\right|}, \quad (39)$$

which, one sees, is exactly analogous to (18), except that the coordinates are rotated and the grid parameters are equal. Inserting this approximation into (38) provides a new update rule that can be used in Algorithm 1. Of course, it is not necessary to limit oneself to a single rotation $(\hat{\imath}, \hat{\jmath})$. To further improve the scheme, one can choose as many pairs $(\hat{\imath}, \hat{\jmath})$ as desired, compute the rotated derivative approximations in each of these directions, and take the minimum over all such approximations. Since the stencil at each grid node will be larger, the scheme will require a larger layer of ghost nodes padding the computational boundary; otherwise, Algorithm 1 will operate in the exact same fashion, but with extra approximations included in the update rule. In general, if one imposes $1 \le \hat{\imath}, \hat{\jmath} \le M$, one should buffer the computational domain with M layers of grid nodes, and there will be some finite number $C(M)$ of distinct angles β created by different pairs $(\hat{\imath}, \hat{\jmath})$.[1]. This is pictured

[1] In fact, one has $C(M) = 2\left(\sum_{m=1}^{M} \varphi(m)\right) - 1$ where φ is the Euler totient function, as detailed in the Online Encyclopedia of Integer Sequences: http://oeis.org/A018805



(a) The possible rotation angles if $1 \le \hat{\imath}, \hat{\jmath} \le 3$.

M	$C(M)$
1	1
2	3
3	7
4	11
5	19
6	23
7	35
8	43
9	55
10	63

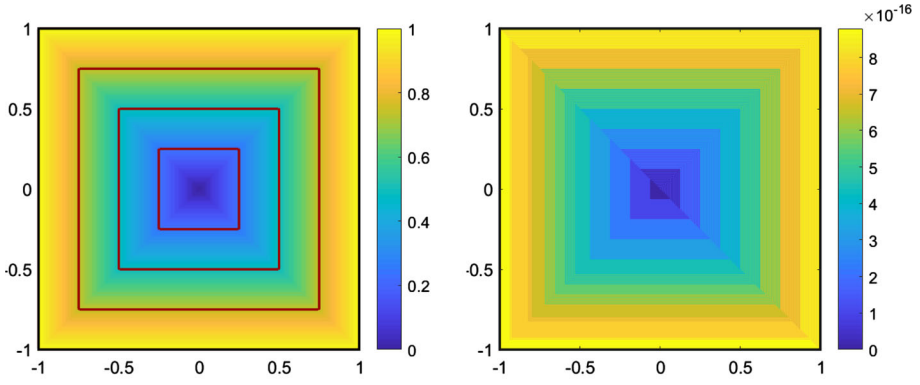
(b) Number of possible angles if $1 \le \hat{\imath}, \hat{\jmath} \le M$.

Fig. 6 If we restrict $1 \le \hat{\imath}, \hat{\jmath} \le M$ there will be some finite number $C(M)$ of distinct rotation angles $\beta = \arctan(\hat{\jmath}/\hat{\imath})$, each represented by a colored line (Color figure online)

in Fig. 6, where each colored line represents a distinct rotation angle β when $M = 3$. Fixing M, we propose two strategies for choosing different rotation angles: first, one could simply use every possible rotation angle. This may be computationally expensive since, for example, when $M = 5$, there are $C(M) = 19$ angles to consider. Accordingly, our second strategy will be to choose some fixed size subcollection at random. This will not be able to guarantee the same level of accuracy, but will be significantly cheaper computationally. It may also be better than choosing a fixed subcollection of angles since, in application, one may not be able to intuit the "principal" directions that need to be captured as we can for the Eikonal equations. Another possibility would be to change the rotation angle β for each grid point, perhaps accounting for the admissible control actions and possible upwind directions; this is essentially what is done by rotating the grid by $\beta = \pi/4$ for the 1-norm Eikonal equation below. To do so more generally, one would need to carefully analyze the particular update rule (39) for one's problem in order to determine a range of possible upwind directions. As presented, we fix the rotation angles β before each iteration.

Note that we will always use the ordinary forward and backward approximations in the (x, y) directions, and include approximations in other directions as desired. This is to establish a baseline. In this manner, using derivative approximations in additional directions can only improve upon the accuracy of the basic method presented in Algorithm 1.

It is natural to consider the optimal number of grid rotations—or similarly, the optimal width of a stencil—for a given problem. Unfortunately, it is difficult to address this point generally. In specific examples, the answer is simple. For example, in the 1-norm Eikonal equation, one can achieve an exact solution with a single grid rotation, as we demonstrate in the succeeding section, and thus additional rotations will offer no benefit. However, for the 2-norm Eikonal equation, each new rotation will serve to better capture the solution at certain points, since characteristics travel outward from the origin in every direction. For general steady-state HJB equations, one may not know the characteristic directions ahead of time, so while adding more rotations can do no worse than the basic scheme, the benefits may be marginal, and they come at the cost of increasing the computational burden. Accordingly, this point would need to be addressed on an *ad hoc* basis, and depends both on the problem and on the user's desire to balance the possibility of large accuracy gains against the increased computation.



(a) Approximate solution **(b)** Error in approximation.

Fig. 7 Numerical solution of $\|\nabla\phi\|_1 = 1$ with additional approximations to $\nabla\phi$ in the direction of $\beta = \pi/4$. Compare with Figs. 2a, 2b

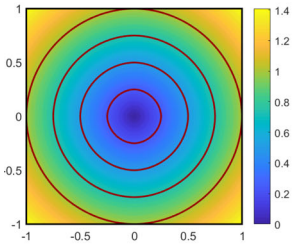
4.1 Application of the Rotating-Grid Method to Eikonal Equations

We apply the sweeping scheme with rotated derivative approximations to the Eikonal equation in the $p = 1$ and $p = 2$ norms. We remarked earlier that cross sections of the solution $\phi_1(x, y) = \max\{|x|, |y|\}$ along the diagonal lines $y = x_0 \pm x$ could be captured exactly by our scheme if we use the rotation $\beta = \pi/4$, which is the same as $(\hat{\imath}, \hat{\jmath}) = (1, 1)$. In this case, the rotated coefficients are $\overline{f}_1 = \frac{1}{\sqrt{2}}(a_1 + a_2)$ and $\overline{f}_2 = \frac{1}{\sqrt{2}}(a_2 - a_1)$, where $a_1, a_2 \in \{\pm 1\}$. Since one of these is zero, the update rule is

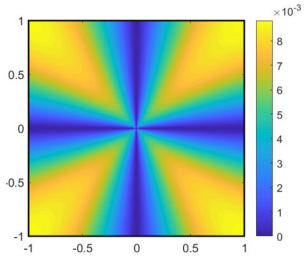
$$\begin{aligned}\phi_{ij}^{n,1} = \min\Bigg\{ & \phi_{ij}^{n-1,1}, \frac{1 + \frac{1}{\Delta x}\phi_{i+1,j}^{n,1} + \frac{1}{\Delta y}\phi_{i,j+1}^{n,1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}}, \frac{1 + \frac{1}{\Delta x}\phi_{i-1,j}^{n,1} + \frac{1}{\Delta y}\phi_{i,j+1}^{n,1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}}, \\ & \frac{1 + \frac{1}{\Delta x}\phi_{i+1,j}^{n,1} + \frac{1}{\Delta y}\phi_{i,j-1}^{n,1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}}, \frac{1 + \frac{1}{\Delta x}\phi_{i-1,j}^{n,1} + \frac{1}{\Delta y}\phi_{i,j-1}^{n,1}}{\frac{1}{\Delta x} + \frac{1}{\Delta y}}, \\ & \phi_{i+1,j+1}^{n,1} + \frac{\Delta s}{\sqrt{2}}, \phi_{i-1,j-1}^{n,1} + \frac{\Delta s}{\sqrt{2}} \\ & \phi_{i-1,j+1}^{n,1} + \frac{\Delta s}{\sqrt{2}}, \phi_{i+1,j-1}^{n,1} + \frac{\Delta s}{\sqrt{2}} \Bigg\}.\end{aligned} \tag{40}$$

We use this update rule in Algorithm 1 to solve $\|\nabla\phi\|_1 = 1$. The results are seen Fig. 7. We note that the level sets of the solution have sharp edges, as opposed to Fig. 2a, where they were rounded off. In this case, the error in the solution is on the order of machine-ε.

Next we solve $\|\nabla\phi\|_2 = 1$. Here, in contrast with $\|\nabla\phi\|_1 = 1$ or $\|\nabla\phi\|_\infty = 1$, we will never be able to solve the equation exactly with finitely many grid rotations. The solution will be resolved exactly along any line through the origin if we consider the derivatives in the direction along that line. We saw this in Fig. 2e; the error is approximately zero along the x-axis and y-axis. We see it further in Fig. 8. In that figure, we first solve $\|\nabla\phi\|_2 = 1$ using the basic method (Figs. 8a–c). We then compare this to results when using approximations to the derivatives in one additional direction (Fig. 8d–f), and three additional directions (Fig. 8g–i). As expected, we see that for a fixed I, J, the error only decreases as we incorporate additional appoximations to $\nabla\phi$ in different directions. Interestingly, the order of convergence appears



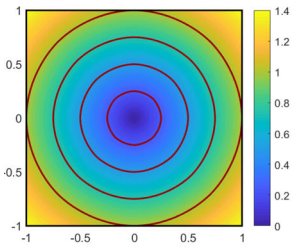
(a) Approx. soln., no additional directions.



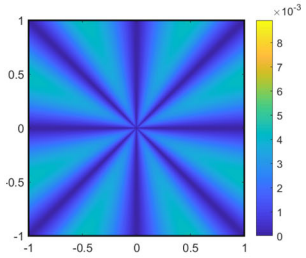
(b) Error in approx., no additional directions.

I, J	L^∞ Err.	Conv.
50	4.3754e-02	—
100	2.6310e-02	0.7338
200	1.5464e-02	0.7666
400	8.9201e-03	0.7938
800	5.0668e-03	0.8160
1600	2.8431e-03	0.8336

(c) Conv. table, no additional directions.



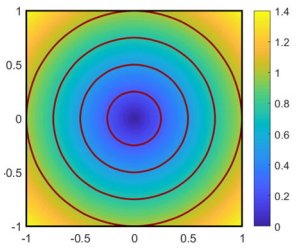
(d) Approx. soln., one additional direction.



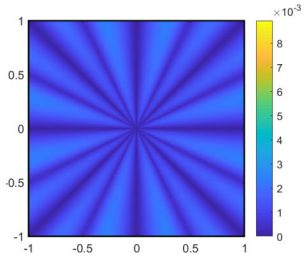
(e) Error in approx., one additional direction.

I, J	L^∞ Err.	Conv.
50	1.7901e-02	—
100	1.1567e-02	0.6300
200	7.2269e-03	0.6789
400	4.3888e-03	0.7192
800	2.6063e-03	0.7518
1600	1.5202e-03	0.7777

(f) Conv. table, one additional direction.



(g) Approx. soln., three additional directions.



(h) Error in approx., three additional directions.

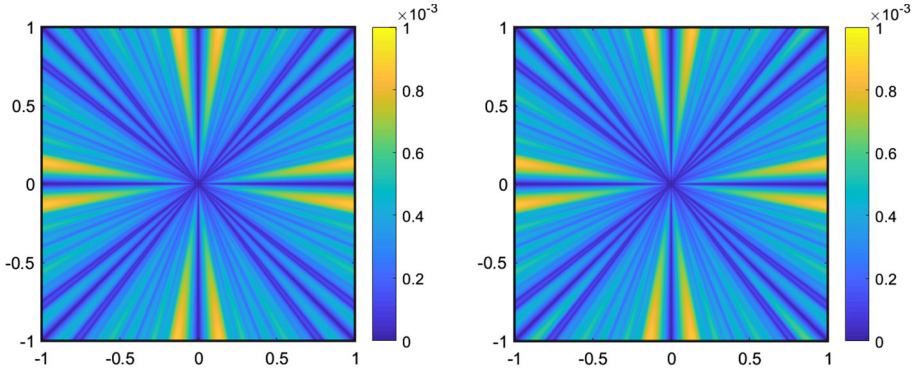
I, J	L^∞ Err.	Conv.
50	8.7787e-03	—
100	5.9351e-03	0.5647
200	3.8508e-03	0.6241
400	2.4134e-03	0.6741
800	1.4720e-03	0.7133
1600	8.7876e-04	0.7443

(i) Conv. table, three additional directions.

Fig. 8 Numerical solution of $\|\nabla\phi\|_2 = 1$ using our fast sweeping method with additional approximations to $\nabla\phi$ in different directions. Scale on error plots is fixed. Error is approximately zero in the directions of the derivative approximations

to slightly decrease when additional directions are included. However, we also note that when using three additional directions one only needs 51 grid points in each direction to achieve the same approximation error as the basic method with 401 points in each direction.

Finally, we solve the same equation using a 401×401 grid and all 19 grid rotations $\beta = \arctan(\hat{\jmath}/\hat{\imath})$ corresponding to $1 \le \hat{\imath}, \hat{\jmath} \le 5$. In Fig. 9a, we see that when using all 19 rotations, we achieve an approximation error of 8.7914×10^{-4}. In this case, the algorithm required 12 iterations to terminate, and each iteration requires 20 times the computation as in the basic method (since there are 20 total approximations to $\nabla\phi$ being computed). In Fig. 9b, we use the same 19 possible grid rotations, but for each iteration we choose only two rotations to use at random. We achieve similar approximation error: 8.7941×10^{-4}. The algorithm required 40 iterations to converge, but each iteration is 3 times as costly as in the basic method. Thus while there are roughly 3 times as many iterations, each iteration requires only 15% of the computation, meaning one can achieve similar approximation error with roughly



(a) Error in approximation when all 19 grid rotations are used in each iteration. Maximum error is 8.7914×10^{-4}.

(b) Error in approximation when each iteration uses 2 grid rotations chosen randomly from the 19 possibilities. Maximum error is 8.7941×10^{-4}

Fig. 9 Error in approximation using rotations $\beta = \arctan(\hat{j}/\hat{\imath})$ where $1 \le \hat{\imath}, \hat{j} \le 5$

half the computation. It should be mentioned that these results have some randomness, but the numbers presented are quite typical.

We note that Darbon and Osher [17] solve similar Eikonal equations using a variational method based on the Hopf-Lax formula. Their method is applicable in high dimensions and can resolve the solution with essentially no error. However, the method only applies to Hamiltonians which are state-independent: $H = H(\nabla\phi)$. Fast sweeping methods are more general, but suffer from the curse of dimensionality. We have included Eikonal equations as an example because they are the prototypical steady-state Hamilton-Jacobi equations.

4.2 Iteration Counts and Comparison with the Lax-Friedrichs Sweeping Scheme

One final consideration when weighing the efficiency of a sweeping scheme is the iteration count necessary for the scheme to converge. Accordingly, we include a brief discussion regarding the iteration counts for the algorithm with different derivative approximations. We note again that one of the primary strengths of our algorithm is its ease of implementation. One other fast sweeping method which shares this ease of implementation is the Lax-Friedrichs (LF) sweeping scheme devised by Kao, Osher and Qian [25]. In two dimensions, their scheme approximates the equation $H(x, y, \phi_x, \phi_y) = r(x, y)$ using the update rule

$$\phi_{ij}^* = \frac{r_{ij} - H\left(x_i, y_j, \frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x}, \frac{\phi_{i,j+1}-\phi_{i,j-1}}{2\Delta y}\right) + \sigma_x \frac{\phi_{i+1,j}+\phi_{i-1,j}}{2\Delta x} + \sigma_y \frac{\phi_{i,j+1}+\phi_{i,j-1}}{2\Delta y}}{\frac{\sigma_x}{\Delta x} + \frac{\sigma_y}{\Delta y}} \tag{41}$$

Intuitively, one arrives at this formula by using the centered difference approximations to ϕ_x and ϕ_y, and adding artificial viscosity at strength $O(\Delta x, \Delta y)$. Here σ_x and σ_y are the artificial viscosity coefficients; they are bounds on $\partial H/\partial\phi_x$ and $\partial H/\partial\phi_y$ respectively.

This method applies to general steady-state Hamilton-Jacobi equations, and is easily implemented regardless of how complicated the Hamiltonian may be. This is in contrast to other fast sweeping schemes, wherein the local update rule entails solving a nonlinear equation whose complexity depends on the Hamiltonian [43,44,62]. The tradeoff is that due

Table 3 The iteration counts for different versions of our algorithm and for the Lax-Friedrichs sweeping scheme when solving $\|\nabla\phi\|_2 = 1$

	Lax-Friedrichs		Basic		Basic+1		Basic+3	
I, J	Iter.	L^∞ Err.	Iter.	L^∞ Err.	Iter.	L^∞ Err.	Iter.	L^∞ Err.
50	34	1.0958e−01	1	4.3754e−02	5	1.7901e−02	5	8.7787e−03
100	43	6.1799e−02	1	2.6310e−02	8	1.1567e−02	7	5.9351e−03
200	59	3.4387e−02	1	1.5464e−02	14	7.2252e−03	10	3.8508e−03
400	91	1.8932e−02	1	8.9201e−03	24	4.3888e−03	18	2.4134e−03

Here Basic+1 designates the basic method appended with derivative approximations in one additional direction; Basic+3 designates the basic method appended with derivative approximations in three additional directions

to the diffusive nature of the LF numerical Hamiltonian, there is no causality condition being enforced, and consequently, a very large number of iterations are required for convergence.

We demonstrate this using the 2-norm Eikonal equation $\|\nabla\phi\|_2 = 1$ on $[-1, 1] \times [-1, 1]$. Note that because the characteristics are straight lines flowing out of the origin, our basic scheme, being fully upwind, converges in a single iteration. When we include additional approximations to the derivatives in rotated directions, this is no longer true. The scheme is still upwind, but there are multiple approximations to a given derivative which obey the causality condition, and alternate iterations may prefer different approximations, which means the algorithm requires more than one iteration to converge. The results are contained in Table 3. As seen in the table, the LF sweeping scheme requires significantly more iterations in order to converge, and results in a larger L^∞ error. As expected, the basic method converges in one iteration for any grid resolution. If we add derivative approximations in different directions, the algorithm no longer converges in one iteration, but empirically, we notice that when we add more approximations, fewer iterations are required. In all of these tests, the convergence criterion is $\max_{ij} \left|\phi_{ij}^n - \phi_{ij}^{n-1}\right| < 10^{-8}$.

It should be noted that, while the LF scheme requires more iterations, each iteration is more efficient since there is no minimization problem or nonlinear inversion. The LF scheme also applies to more general problems. However, in cases where the minimization in our scheme is easily resolved, it is likely to outperform the LF scheme both in terms of efficiency and accuracy. We see this with the last example in Sect. 5.

5 Other Applications

Lastly, we present two applications of our method to problems arising in engineering. First we consider the visibility problem. Here one could imagine placing cameras at fixed points in a domain. The cameras have omnidirectional view, but the view is occluded by obstacles. The problem is to find the region that is visible to the cameras.

This problem was first formulated using partial differential equations and the level set method by Tsai et al. [56]. However, that formulation involves a nonlocal equation. More recently, Oberman and Salvador were able to recast the problem in terms of a simple, local equation [34]. Specifically, supposing that $g : \mathbb{R}^d \to \mathbb{R}$ is the signed distance function to the obstacles (positive inside the obstacles) and $x^* \in \mathbb{R}^d$ is the vantage point, the visibility

function $\phi : \mathbb{R}^d \to \mathbb{R}$ satisfies

$$0 = \min\{\phi(x) - g(x),\ \langle x - x^*, \nabla\phi(x)\rangle\} \tag{42}$$

with the boundary condition $\phi(x^*) = g(x^*)$. The visibility set is then given by $\{\phi \le 0\}$. To include multiple vantage points, one solves (42) individually for each point, and combines the solution via minima and maxima to account for different scenarios (for example, the minimum of all such solutions will provide the set of points visible from at least one vantage point, while the maximum of all such solutions provides the set of points that are visible from all vantage points simultaneously).

Note that while Eq. (42) does not directly follow from an optimal control problem, it does fit into our framework. If one sets $\phi^0_{ij} = g_{ij}$ for the nodes closest to the vantage point (x^*, y^*) and $\phi^0_{ij} = -\infty$ at other nodes, one can use the update rule

$$\phi^*_{ij} = \frac{\frac{|x_i - x^*|}{\Delta x}\phi_{i-\text{sign}(x_i-x^*),j} + \frac{|y_j-y^*|}{\Delta y}\phi_{i,j-\text{sign}(y_i-y^*)}}{\frac{|x_i-x^*|}{\Delta x} + \frac{|y_j-y^*|}{\Delta y}}, \tag{43}$$

and iterate $\phi^n_{ij} = \max\{\phi^{n-1}_{ij}, g_{ij}, \phi^*_{ij}\}$. [Note that the upwind direction is reversed, which explains the slight deviations between these formulas and those above.] One can then use additional approximations to $\nabla\phi$ as desired. We used this update rule and applied Algorithm 1 with a 401×401 grid and with approximations to $\nabla\phi$ along the x-axis and y-axis as well as the $\beta = \pi/4$ direction. The results are seen in Fig. 10, where the yellow set represents the visible set, the black shapes are obstacles and the green dots are the vantage points. In this case, because there is no control variable, the upwind direction is fixed and characteristics are straight lines flowing away from the vantage points. Because of this simple geometry, the scheme requires only one iteration and values at grid nodes are resolved during one of the directional sweeps depending on where they lie relative to the vantage point. For example, if the vantage point is at grid node (i^*, j^*), then the forward-forward sweep will resolve all values ϕ_{ij} with $i > i^*$ and $j > j^*$. It should be noted that Oberman and Salvador also devised an upwind sweeping scheme that approximates (42) with one sweep in each direction by using interpolation to explicitly capture the exact upwind direction. Our method is not an improvement of theirs; we include this example only to demonstrate the diverse applicability of our method. For a full discussion of the visibility problem including rigorous analysis of (42), see [34].

Our final application is in time-optimal path planning for simple self-driving cars. This problem was first analyzed by Dubins [18] and Reeds and Shepp [46] in a purely geometric sense, and later analyzed in the Hamilton-Jacobi formulation by Takei, Tsai and others [40,54,55]. Let (x, y) denote the location of the center of mass of the vehicle and θ denote the orientation. If W is the maximum angular velocity of the car (which enforces a minimum turning radius) and d is the distance from the rear wheels—which drive the car—to the center of mass, then the kinematics are

$$\begin{aligned} \dot{x} &= v\cos(\theta) - \omega W d\sin(\theta), \\ \dot{y} &= v\sin(\theta) + \omega W d\cos(\theta), \\ \dot{\theta} &= W\omega, \end{aligned} \tag{44}$$

where $v, \omega \in [-1, 1]$ are normalized control variables representing tangential and angular velocity respectively [60].

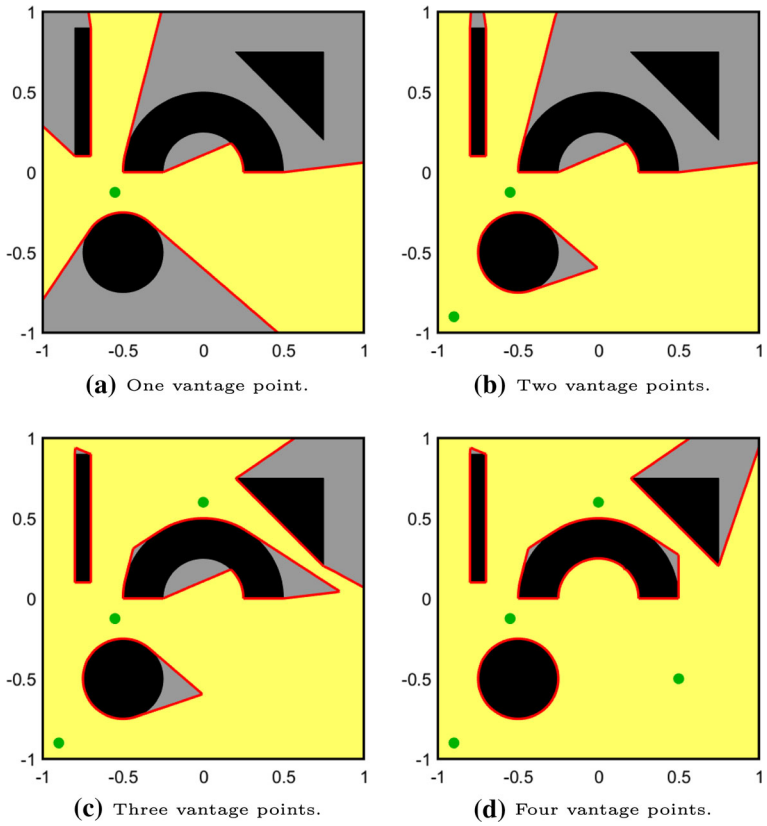


Fig. 10 Computing the visibility set using (42). The green dots represent the vantage points. The black shapes are obstacles. The yellow set is comprised of points visible from at least one vantage point. The grey set is the unobserved set (Color figure online)

With these kinematics, the optimal travel time function solves the Hamilton-Jacobi equation

$$-1 = \inf_{v,\omega} \left\{ [v\cos(\theta) - \omega W d \sin(\theta)]\phi_x + [v\sin(\theta) + \omega W d\cos(\theta)]\phi_y + \omega W \phi_\theta \right\}. \quad (45)$$

For a full derivation of this equation, we direct the reader to [55]; they consider the case that $d = 0$ so the car is simplified to a point mass, but otherwise the derivation is the same. One notes that the minimization is linear in (v, ω), and thus, since the minimization set $[-1, 1] \times [-1, 1]$ has finitely many extreme points, there are finitely many values that the pair (v, ω) will take. For technical reasons, one should allow $v \in \{-1, 1\}$ and $\omega \in \{-1, 0, 1\}$ [55].

Equation (45) fits directly into our framework. Discretizing (x_i, y_j, θ_k), Eq. (45) is approximated by the update rule

$$\begin{aligned}\phi^*_{ijk}(v,\omega) = \Big\{1 &+ \frac{|A_k(v,\omega)|}{\Delta x}\phi_{i+a_k(u,v),j,k} \\ &+ \frac{|B_k(v,\omega)|}{\Delta y}\phi_{i,j+b_k(v,\omega),k} \\ &+ \frac{|\omega|\,W}{\Delta\theta}\phi_{i,j,k+\mathrm{sign}(\omega)}\Big\} \\ &/\Big\{\frac{|A_k(v,\omega)|}{\Delta x} + \frac{|B_k(v,\omega)|}{\Delta y} + \frac{|\omega|\,W}{\Delta\theta}\Big\},\end{aligned} \tag{46}$$

where

$$\begin{aligned}A_k(v,\omega) &= v\cos(\theta_k) - \omega W d\sin(\theta_k),\\ B_k(v,\omega) &= v\sin(\theta_k) + \omega W d\cos(\theta_k),\\ a_k(v,\omega) &= \mathrm{sign}(v\cos(\theta_k) - \omega W d\sin(\theta_k)),\\ b_k(v,\omega) &= \mathrm{sign}(v\sin(\theta_k) + \omega W d\cos(\theta_k)).\end{aligned} \tag{47}$$

One can use this update rule in Algorithm 1 (accounting for three dimensions by performing 8 sweeps per iteration) with the boundary condition $\phi^0_{i^*,j^*,k^*} = 0$ for the desired ending configuration and $\phi^0_{ijk} = +\infty$ otherwise. Then ϕ_{ijk} will represent the approximate time needed to travel from grid node (i, j, k) to grid node (i^*, j^*, k^*) while obeying (44).

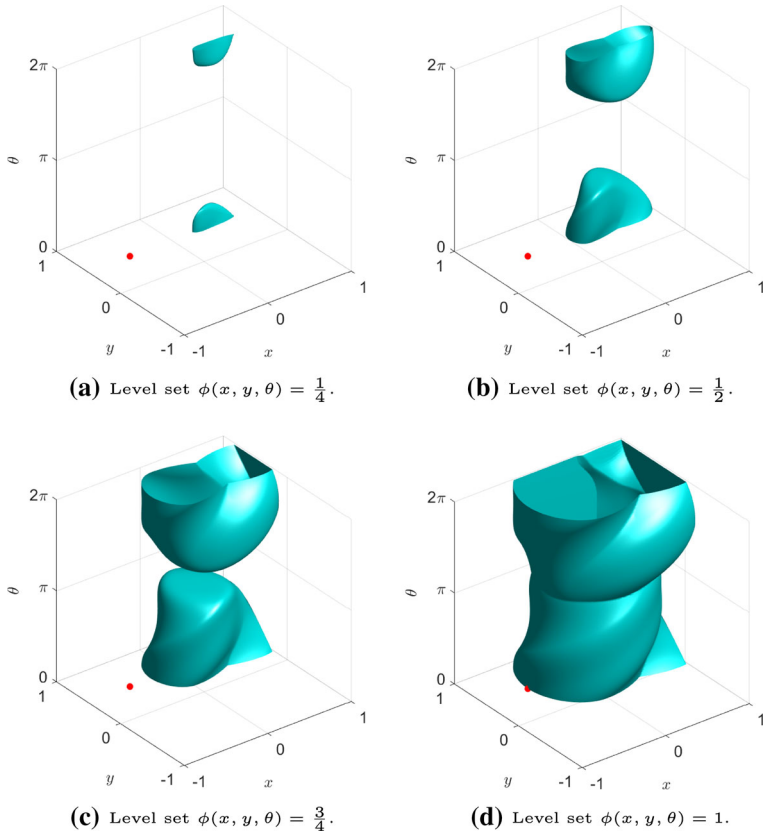
In three dimensions, it is less obvious how to incorporate grid rotations in a fully principled manner. We discuss this further in Appendix A. One approach is to restrict ourselves to rotations of the xy-plane while keeping the θ-axis fixed. In doing so, we can again trade (x, y) for $(\overline{x}, \overline{y})$ exactly as in the two-dimensional case. Using this strategy, if the rotation angle is $\beta = \arctan(\hat{\jmath}/\hat{\imath})$, the new update rule is

$$\begin{aligned}\overline{\phi}^*_{ijk}(v,\omega) = \Big\{\Delta s\ &+ \big|\overline{A}_k(v,\omega)\big|\,\phi_{i+\hat{\imath}\overline{a}_k(u,v),j+\hat{\jmath}\overline{a}_k(u,v),k} \\ &+ \big|\overline{B}_k(v,\omega)\big|\,\phi_{i-\hat{\imath}\overline{b}_k(v,\omega),j+\hat{\jmath}\overline{b}_k(v,\omega),k} \\ &+ \frac{\Delta s\,|\omega|\,W}{\Delta\theta}\phi_{i,j,k+\mathrm{sign}(\omega)}\Big\} \\ &/\Big\{|A_k(v,\omega)| + |B_k(v,\omega)| + \frac{\Delta s\,|\omega|\,W}{\Delta\theta}\Big\},\end{aligned} \tag{48}$$

where $\Delta s = \sqrt{(\hat{\imath}\Delta x)^2 + (\hat{\jmath}\Delta y)^2}$ as before, and

$$\begin{aligned}\overline{A}_k(v,\omega) &= v\cos(\theta_k+\beta) - \omega W d\sin(\theta_k+\beta),\\ \overline{B}_k(v,\omega) &= v\sin(\theta_k+\beta) + \omega W d\cos(\theta_k+\beta),\\ \overline{a}_k(v,\omega) &= \mathrm{sign}(v\cos(\theta_k+\beta) - \omega W d\sin(\theta_k+\beta)),\\ \overline{b}_k(v,\omega) &= \mathrm{sign}(v\sin(\theta_k+\beta) + \omega W d\cos(\theta_k+\beta)).\end{aligned} \tag{49}$$

We used these formulas on a $201\times201\times201$ discretization of $[-1,1]\times[-1,1]\times[0,2\pi]$ to compute the travel-time function for this control problem when the ending configuration is $(\frac{1}{2}, \frac{1}{2}, 0)$ meaning the car should end at $(x_f, y_f) = (\frac{1}{2}, \frac{1}{2})$ facing in the positive x-direction. In all these tests, the convergence criterion is $\max_{ijk}|\phi^n_{ijk} - \phi^{n-1}| < 10^{-4}$. The results in Figs. 11 and 12 were generated using three additional directions to approximate ϕ_x, ϕ_y: the directions of $\beta = \arctan(1/2), \arctan(1), \arctan(2/1)$. One way to evaluate the results is to



(a) Level set $\phi(x, y, \theta) = \frac{1}{4}$. **(b)** Level set $\phi(x, y, \theta) = \frac{1}{2}$.

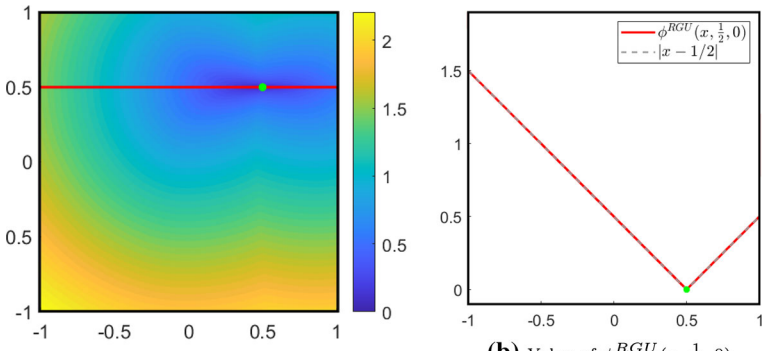
(c) Level set $\phi(x, y, \theta) = \frac{3}{4}$. **(d)** Level set $\phi(x, y, \theta) = 1$.

Fig. 11 Level sets (cyan) of the travel-time function $\phi(x, y, \theta)$ with ending point $(\frac{1}{2}, \frac{1}{2}, 0)$. Plotted in red is the point $(-\frac{1}{2}, \frac{1}{2}, 0)$. This point should have a travel time of 1, and indeed the level set $\phi(x, y, \theta) = 1$ includes the point (Color figure online)

compare them against known values of the travel-time function. For example, anywhere along the line $(x, \frac{1}{2}, 0)$, the optimal travel time is $|x - \frac{1}{2}|$ since the optimal path simply requires pulling forward or reversing into the final configuration. Accordingly, on the level set plots in Fig. 11, we plot the point $(-\frac{1}{2}, \frac{1}{2}, 0)$ in red. This point should satisfy $\phi(-\frac{1}{2}, \frac{1}{2}, 0) = 1$ and indeed, it seems to approximately lie in the level set $\phi(x, y, \theta) = 1$ (Fig. 11d). Likewise, in Fig. 12, we display the contours of $\phi(x, y, 0)$ which show the values of the travel-time function given that the car is facing in the positive x-direction. Using these, we can directly compare values of $\phi(x, \frac{1}{2}, 0)$ and $|x - \frac{1}{2}|$ and the results line up very well.

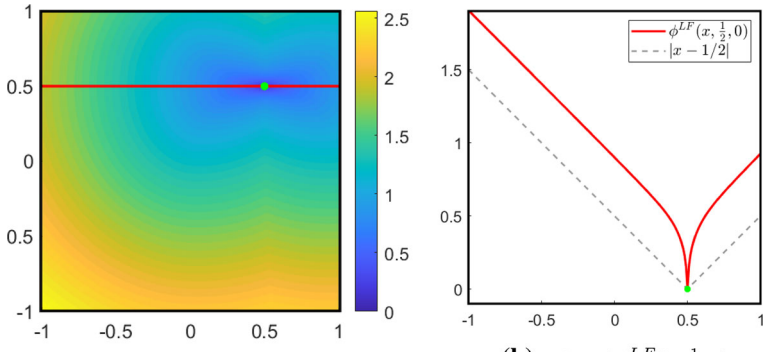
Again, we compare our results to those of the Lax-Friedrichs (LF) sweeping scheme [25]. Because the LF scheme includes artificial viscosity, it has trouble resolving the value function in the neighborhood surrounding the source point (x_f, y_f, θ_f). Indeed, we computed the solution of the same problem using the LF scheme. Values analogous to those in Fig. 12 are displayed in Fig. 13. We note there is some error in the values of $\phi(x, \frac{1}{2}, 0)$. We also notice that the solution suggested by the LF scheme takes larger values throughout the domain, which hints that the optimal travel time is being overestimated.

Table 4 lists the iteration counts for different grid resolutions, and different solution methods. In the table, "Basic" denotes the basic scheme, and "Basic+M" denotes the basic scheme



(a) Contours of $\phi^{RGU}(x, y, 0)$. **(b)** Value of $\phi^{RGU}(x, \frac{1}{2}, 0)$.

Fig. 12 Contour plot of the approximate travel-time function $\phi^{RGU}(x, y, 0)$ with ending point $(\frac{1}{2}, \frac{1}{2}, 0)$ [green], computed using our Rotating Grid Upwind (RGU) method. Along the line $(x, \frac{1}{2}, 0)$ [red] the exact solution value is $|x - 1/2|$, and our results match these values (Color figure online)



(a) Contours of $\phi^{LF}(x, y, 0)$. **(b)** Value of $\phi^{LF}(x, \frac{1}{2}, 0)$.

Fig. 13 Contour plot of the approximate travel-time function $\phi^{LF}(x, y, 0)$ with ending point $(\frac{1}{2}, \frac{1}{2}, 0)$ [green], computed using the Lax-Friedrichs (LF) method. Along the line $(x, \frac{1}{2}, 0)$ [red] the exact solution value is $|x - 1/2|$, and due to the diffusive nature of the scheme, the approximate solution incurs some error (Color figure online)

appended with M grid rotations. A first note is that for this problem, including additional derivative approximations in different directions lowers the number of iterations required for our algorithm to converge. Due to diffusivity, the LF method requires vastly more iterations. In this case, the LF iterations are no more or less efficient than those of our method. Problems where the control values can be resolved explicitly are well-suited to our method. For problems of this type, our method is very likely to ourperform the LF method and is equally easy to implement. It bears repeating that the LF method is more generally applicable and easier to implement for problems with very complicated Hamiltonians [25].

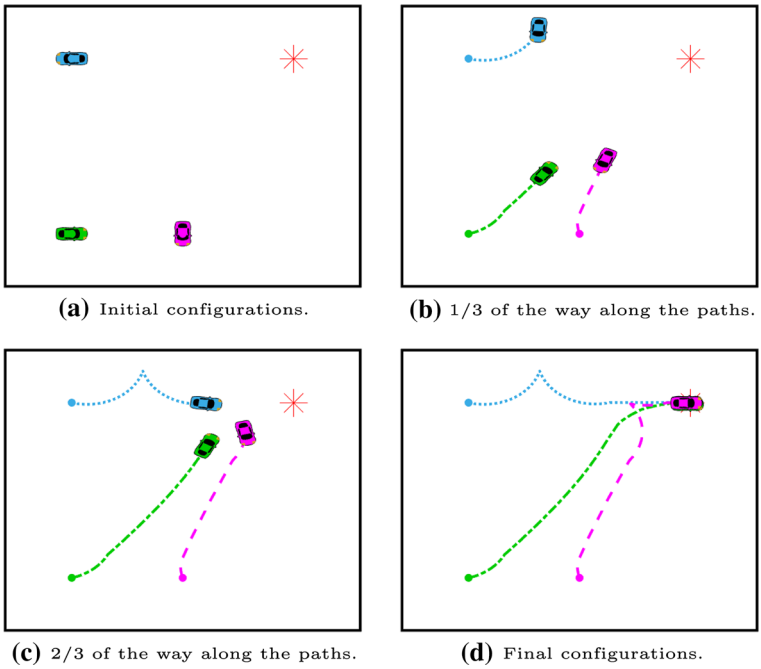
Another way one can verify the results of these simulations is to compute the actual paths given by the control problem. Having computed the travel-time function ϕ, one can determine optimal trajectories by integrating (44) using control values

$$\begin{aligned} v &= -\text{sign}(\phi_x \cos\theta + \phi_y \sin\theta), \\ \omega &= -\text{sign}(-d\phi_x \sin\theta + d\phi_y \cos\theta + \phi_\theta). \end{aligned} \tag{50}$$

Table 4 The iteration counts for different versions of our algorithm, and for the Lax-Friedrichs sweeping scheme, when resolving the optimal travel-time function for the simple self-driving car

Iteration counts: self-driving car example				
I, J, K	LF	Basic	Basic+1	Basic+3
50	99	17	16	17
100	187	25	22	21
200	309	32	26	24

Here Basic+1 designates the basic method appended with derivative approximations in one additional direction; Basic+3 designates the basic method appended with derivative approximations in three additional directions



(a) Initial configurations. **(b)** 1/3 of the way along the paths. **(c)** 2/3 of the way along the paths. **(d)** Final configurations.

Fig. 14 Optimal paths for cars with initial configurations $(-\frac{1}{2}, \frac{1}{2}, \pi)$ [blue], $(-\frac{1}{2}, -\frac{1}{2}, 0)$ [green], and $(0, -\frac{1}{2}, \frac{5\pi}{4})$ [pink]. Final configuration is $(\frac{1}{2}, \frac{1}{2}, 0)$ [red star] (Color figure online)

Some optimal paths are seen in Fig. 14. In those plots, the final location is marked by the red star, and the initial locations are marked by colored dots. The positions of the vehicles are displayed at several points along their respective optimal trajectories. Note, these optimal paths were computed independently and are simply plotted on top of each other; the paths will require different amounts of time to traverse and there is no interaction between the cars. The results appear to agree with a theoretical result of Reeds and Shepp [46] that states that optimal trajectories consist of straight lines and arcs of circles of minimum radius.

6 Conclusion and Discussion

Fast sweeping methods provide a simple and robust framework for numerical solutions of steady-state Hamilton-Jacobi equations. We have developed a fast sweeping scheme for a class of Hamilton-Jacobi equations arising from steady-state optimal control problems wherein the running cost is independent of the control variables. Our method is exceedingly simple to implement and applies to a wide range of problems. We tested our method against Eikonal equations in different norms, and demonstrated how one can use WENO approximations to improve accuracy. We then suggested a general method for maintaining a square grid, but using approximations to derivatives in rotated directions, so as to more accurately capture the information flow along characteristics. We compare our method against the Lax-Friedrichs method [25] and demonstrate that in some cases, our method is preferable. Finally, we demonstrated the utility of our method by applying it to two problems arising from engineering applications.

There are several ways in which our method could be modified or adjusted for other scenarios. We suggest two such modifications now. First, a further exploration of the efficacy of WENO approximations in conjunction with our method could prove interesting. In Sect. 3.3, we demonstrated one method for including WENO approximations, following [62]. However, especially when the solution was non-smooth, we did not achieve the full increase in accuracy that one may desire. It is possible that one could improve this with a closer analysis of the scheme near the point source. One may also try to include WENO approximations with the grid rotations. This is likely to be difficult due to the different sizes of the rotated grid parameters $\Delta\overline{x}$, $\Delta\overline{y}$ which may skew convergence results, so one would need to be cautious. Second, when using a single grid rotation with angle $\beta = \pi/4$, we are essentially using a 9 point stencil for local derivative approximations, which yields a structured triangulation of the domain. It would be interesting to modify the method for unstructured and/or triangulated domains such as those in [44]. In these domains, our method may provide a simpler update rule for Eikonal equations, though a careful analysis would be required.

Acknowledgements The author thanks Andrea Bertozzi and Stanley Osher for reading an early version of this manuscript, and for several valuable conversations and suggestions, especially regarding the example of optimal path planning for self-driving cars. The author also thanks two anonymous reviewers for helpful comments and suggestions which improved the manuscript.

Declarations Data sharing not applicable to this article as no datasets were generated or analyzed during the current study. The author has no conflicts of interest to declare that are relevant to the content of this article.

A Appendix: 3D Implementation

In this appendix, we briefly describe the implementation of the rotating-grid method in three dimensions. In this case, the equation of interest is

$$-r(x, y, z) = \inf_{a \in A} \left\{ f_1(x, y, z, a)\phi_x + f_2(x, y, z, a)\phi_y + f_3(x, y, z, a)\phi_z \right\}. \quad (51)$$

The extension of the basic method to 3D is straightforward. We discretize the domain into (x_i, y_j, z_k) with uniform grid parameters Δx, Δy and Δz. Then following the work in Sect. 3, we arrive at the local upwind approximation

$$\phi^*_{ijk}(a) = \frac{r_{ijk} + \frac{|f_{1,ijk}(a)|}{\Delta x}\phi_{i+\xi_{1,ijk}(a),j,k} + \frac{|f_{2,ijk}(a)|}{\Delta y}\phi_{i,j+\xi_{2,ijk}(a),k} + \frac{|f_{3,ijk}(a)|}{\Delta z}\phi_{i,j,k+\xi_{3,ijk}(a)}}{\frac{|f_{1,ijk}(a)|}{\Delta x} + \frac{|f_{2,ijk}(a)|}{\Delta y} + \frac{|f_{3,ijk}(a)|}{\Delta z}}, \tag{52}$$

where $\xi_{\ell,ijk}(a) = \text{sign}(f_{\ell,ijk}(a))$ as before. One can then use the update rule $\phi^n_{ijk} = \min\{\phi^{n-1}_{ijk}, \min_{a\in A}\phi^*_{ijk}(a)\}$, while performing 8 sweeps per iteration to account for all combinations of sweeping directions.

In theory, introducing a rotation in $\mathbb{R}^3$ is not too different from introducing a rotation in $\mathbb{R}^2$. One can choose an orthogonal matrix $U = [\, u_1 \mid u_2 \mid u_3 \,]$ whose columns represent the directions of the new axes, and set

$$\begin{pmatrix} \overline{x} \\ \overline{y} \\ \overline{z} \end{pmatrix} = U^t \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad \text{so that} \qquad \nabla\phi = U\overline{\nabla}\phi, \tag{53}$$

where ∇ represents the gradient in the original coordinates, and $\overline{\nabla}$ represents the gradient with respect to the new coordinates. Plugging this representation of $\nabla\phi$ into (51) and denoting $f := (f_1, f_2, f_3)$ gives

$$-r = \inf_{a\in A}\left\{\langle f(a), U\overline{\nabla}\phi\rangle\right\} = \inf_{a\in A}\left\{\langle U^t f(a), \overline{\nabla}\phi\rangle\right\}. \tag{54}$$

Thus, defining $\overline{f}_\ell(\overline{x},\overline{y},\overline{z},a) = \langle u_\ell, f(\overline{x},\overline{y},\overline{z},a)\rangle$ for $\ell = 1,2,3$, the rotated equation is

$$-r(\overline{x},\overline{y},\overline{z}) = \inf_{a\in A}\left\{\overline{f}_1(\overline{x},\overline{y},\overline{z},a)\phi_{\overline{x}} + \overline{f}_2(\overline{x},\overline{y},\overline{z},a)\phi_{\overline{y}} + \overline{f}_3(\overline{x},\overline{y},\overline{z},a)\phi_{\overline{z}}\right\}. \tag{55}$$

Now the question arises of how to discretize this equation. Similar to the 2D formulation, we would like to avoid defining a new grid, but rather restrict ourselves to rotations which allow us to use the already-defined grid points to approximate derivatives in different directions. There is a practical complication to address here. As demonstrated above, in 2D it is sufficient to choose a grid point $(\hat{\imath}, \hat{\jmath})$ and rotate the grid so that the $\overline{x}$-axis points at $(\hat{\imath}, \hat{\jmath})$. Having done so, the new $\overline{y}$-axis points at $(-\hat{\jmath}, \hat{\imath})$ as shown in Fig. 5. However, in 3D, there are infinitely many rotations which fix the $\overline{x}$-axis in a specified direction. Thus, in analogy to the 2D scenario, there needs to be a principled manner by which to point the $\overline{x}$-axis toward a desired grid point $(\hat{\imath}, \hat{\jmath}, \hat{k})$ while ensuring that the $\overline{y}$- and $\overline{z}$-axes are still pointed toward other grid points, so as to avert the need to define a new grid.

We suggest two ways for doing this. The first, which is simpler but not as general, is to restrict oneself to rotations which fix one of the axes, as we did in the example of time-optimal path planning for self-driving cars in Sect. 5. Here one chooses $(\hat{\imath}, \hat{\jmath})$ as before, and also specifies which axis is to remain fixed. In doing so, the 3D implementation is effectively reduced to a 2D implementation, since one of the derivatives follows through the computation without changing.

The second method can handle general rotations, but is slightly more difficult to describe. Here, we suggest choosing a grid point $(\hat{\imath}, \hat{\jmath}, \hat{k})$ and using the rotation which orients the $\overline{x}$-axis toward $(\hat{\imath}, \hat{\jmath}, \hat{k})$ by viewing it as the image of the x-axis under two successive rotations: first, a rotation by $\beta = \arctan(\hat{\jmath}/\hat{\imath})$ about the z-axis, and then a rotation by $\gamma = \arctan(\hat{k}/\sqrt{\hat{\imath}^2 + \hat{\jmath}^2})$ about the line $\hat{\imath}x + \hat{\jmath}y = 0$. This is illustrated in Fig. 15 where the black lines are the original axes, the red lines are the axes resulting from the first rotation (note, the z-axis is unchanged under the first rotation), and the blue lines are the new axes after both rotations.

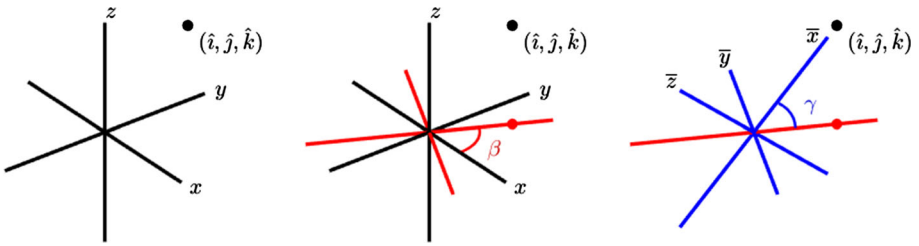


Fig. 15 We orient $\overline{x}$ to point at $(\hat{\imath}, \hat{\jmath}, \hat{k})$ by viewing it as the image of the x-axis under successive rotations: first a rotation by β about the z-axis, and then a rotation by γ about the line $\hat{\imath}x + \hat{\jmath}y = 0$

In terms of the old coordinates, the new orthogonal coordinates are given by

$$\begin{aligned} &\text{positive } \overline{x} \text{ direction: } (\hat{\imath}, \hat{\jmath}, \hat{k}), \\ &\text{positive } \overline{y} \text{ direction: } (-\hat{\jmath}, \hat{\imath}, 0), \\ &\text{positive } \overline{z} \text{ direction: } (-\hat{\imath}\hat{k}, -\hat{\jmath}\hat{k}, \hat{\imath}^2 + \hat{\jmath}^2). \end{aligned} \tag{56}$$

As formulated in (54), the columns of U are normalized versions of the three vectors in (56). Using these, we can write the upwind approximations to the derivatives $\phi_{\overline{x}}$, $\phi_{\overline{y}}$ and $\phi_{\overline{z}}$ necessary to approximate (55). Indeed,

$$\begin{aligned} \left(\overline{f}_1(\overline{x}, \overline{y}, \overline{z}, a)\phi_{\overline{x}}\right)_{ijk} &= \left|\overline{f}_{1,ijk}(a)\right| \frac{\phi_{i+\hat{\imath}\overline{\xi}_{1,ijk}(a),\, j+\hat{\jmath}\overline{\xi}_{1,ijk}(a),\, k+\hat{k}\overline{\xi}_{1,ijk}(a)} - \phi_{ijk}}{\Delta\overline{x}}, \\ \left(\overline{f}_2(\overline{x}, \overline{y}, \overline{z}, a)\phi_{\overline{y}}\right)_{ijk} &= \left|\overline{f}_{2,ijk}(a)\right| \frac{\phi_{i-\hat{\jmath}\overline{\xi}_{2,ijk}(a),\, j+\hat{\imath}\overline{\xi}_{2,ijk}(a),\, k} - \phi_{ijk}}{\Delta\overline{y}}, \\ \left(\overline{f}_3(\overline{x}, \overline{y}, \overline{z}, a)\phi_{\overline{z}}\right)_{ijk} &= \left|\overline{f}_{3,ijk}(a)\right| \frac{\phi_{i-\hat{\imath}\hat{k}\overline{\xi}_{3,ijk}(a),\, j-\hat{\jmath}\hat{k}\overline{\xi}_{3,ijk}(a),\, k+(\hat{\imath}^2+\hat{\jmath}^2)\overline{\xi}_{3,ijk}(a)} - \phi_{ijk}}{\Delta\overline{z}} \end{aligned} \tag{57}$$

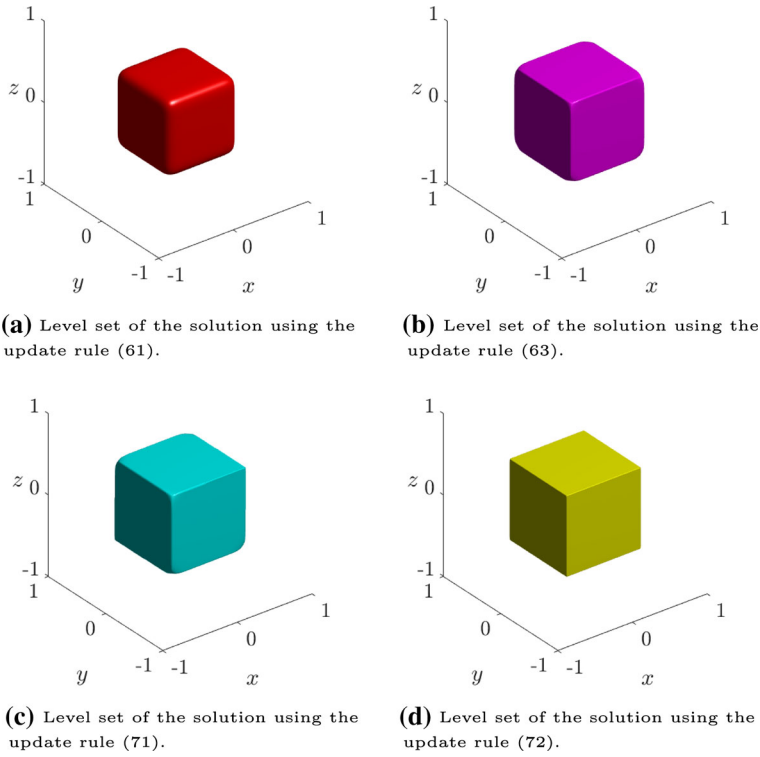
where, as before, $\overline{\xi}_{\ell,ijk}(a) = \operatorname{sign}\left(\overline{f}_{\ell,ijk}(a)\right)$, and the new grid parameters are given by

$$\begin{aligned} \Delta\overline{x} &= \sqrt{(\hat{\imath}\Delta x)^2 + (\hat{\jmath}\Delta y)^2 + (\hat{k}\Delta z)^2}, \\ \Delta\overline{y} &= \sqrt{(\hat{\jmath}\Delta x)^2 + (\hat{\imath}\Delta y)^2}, \\ \Delta\overline{x} &= \sqrt{(\hat{\imath}\hat{k}\Delta x)^2 + (\hat{\jmath}\hat{k}\Delta y)^2 + ((\hat{\imath}^2 + \hat{\jmath}^2)\Delta z)^2}. \end{aligned} \tag{58}$$

We will demonstrate both of these methods of implementation using the 1-norm Eikonal equation as an example; first for its simplicity, and second because the level sets of the solution have sharp edges which allow us to easily verify the results. In 3D, the 1-norm Eikonal equation is given by

$$-1 = -|\phi_x| - |\phi_y| - |\phi_z| = \inf_{a_i \in \{\pm 1\}} \left\{ a_1\phi_x + a_2\phi_y + a_3\phi_z \right\}. \tag{59}$$

We use the boundary condition $\phi(0, 0, 0) = 0$. The solution is $\phi(x, y, z) = \max\{|x|, |y|, |z|\}$, and the level sets of this solution are perfect cubes. In all cases, we use a $201 \times 201 \times 201$ discretization of $[-1, 1]^3$ and display the level set $\phi(x, y, z) = 1/2$ which should be a



(a) Level set of the solution using the update rule (61).

(b) Level set of the solution using the update rule (63).

(c) Level set of the solution using the update rule (71).

(d) Level set of the solution using the update rule (72).

Fig. 16 The 1/2-level set of the approximate solution to $\|\nabla\phi\|_1 = 1$ resulting from our method when using the update rules (61) (no rotated directions), (63) (rotations which keep one axis fixed), (71) (an additional rotation to resolve the corners along $(1, -1, 1)$), and (72) (rotations to capture all the corners)

cube centered at the origin of side length 1. We successively build better approximations of the solution by including additional approximations to the derivatives in different rotated directions.

Applying the basic method, we find that the local upwind approximation to the solution is

$$\phi^*_{ijk}(a) = \frac{1 + \frac{1}{\Delta x}\phi_{i+\text{sign}(a_1),j,k} + \frac{1}{\Delta y}\phi_{i,j+\text{sign}(a_2),k} + \frac{1}{\Delta z}\phi_{i,j,k+\text{sign}(a_3)}}{\frac{1}{\Delta x} + \frac{1}{\Delta y} + \frac{1}{\Delta z}}. \tag{60}$$

Then the local update rule for the iteration is

$$\phi^n_{ijk} = \min\left\{\phi^{n-1}_{ijk}, \ \min_a \phi^*_{ijk}(a)\right\}. \tag{61}$$

A level set of the solution produced by this update rule is displayed in red in Fig. 16a. Note the rounding along the edges and at the corners.

Next, we implement the first method for incorporating grid rotations, wherein we use rotations which keep one axis fixed. Since the level sets of the solution are cubes, we will use rotations of $\beta = \pi/4$ in attempt to capture the edges. We will implement three rotations, alternately keeping the x-, y-, or z-axis fixed. The local upwind approximations are then

$$\phi^{*,z}_{ijk}(a) = \frac{1 + \frac{|a_1+a_2|}{\sqrt{2}\Delta s_{xy}}\phi_{i+\text{sign}(a_1+a_2),j+\text{sign}(a_1+a_2),k} + \frac{|a_2-a_1|}{\sqrt{2}\Delta s_{xy}}\phi_{i-\text{sign}(a_2-a_1),j+\text{sign}(a_2-a_1),k} + \frac{1}{\Delta z}\phi_{i,j,k+\text{sign}(a_3)}}{\frac{|a_1+a_2|}{\sqrt{2}\Delta s_{xy}} + \frac{|a_2-a_1|}{\sqrt{2}\Delta s_{xy}} + \frac{1}{\Delta z}},$$

$$\phi^{*,y}_{ijk}(a) = \frac{1 + \frac{|a_1+a_3|}{\sqrt{2}\Delta s_{xz}}\phi_{i+\text{sign}(a_1+a_3),j,k+\text{sign}(a_1+a_3)} + \frac{1}{\Delta y}\phi_{i,j+\text{sign}(a_2),k} + \frac{|a_3-a_1|}{\sqrt{2}\Delta s_{xz}}\phi_{i-\text{sign}(a_3-a_1),j,k+\text{sign}(a_3-a_1)}}{\frac{|a_1+a_3|}{\sqrt{2}\Delta s_{xz}} + \frac{1}{\Delta y} + \frac{|a_3-a_1|}{\sqrt{2}\Delta s_{xz}}},$$

$$\phi^{*,x}_{ijk}(a) = \frac{1 + \frac{1}{\Delta x}\phi_{i+\text{sign}(a_1),j,k} + \frac{|a_2+a_3|}{\sqrt{2}\Delta s_{yz}}\phi_{i,j+\text{sign}(a_2+a_3),k+\text{sign}(a_2+a_3)} + \frac{|a_3-a_2|}{\sqrt{2}\Delta s_{yz}}\phi_{i,j-\text{sign}(a_3-a_2),k+\text{sign}(a_3-a_2)}}{\frac{1}{\Delta x} + \frac{|a_2+a_3|}{\sqrt{2}\Delta s_{yz}} + \frac{|a_3-a_2|}{\sqrt{2}\Delta s_{yz}}}, \tag{62}$$

where the superscript denotes the axis that is fixed, $\Delta s_{xy} = \sqrt{(\Delta x)^2 + (\Delta y)^2}$ and similarly for Δs_{xz} and Δs_{yz}. The update rule for the iteration is then

$$\phi^n_{ijk} = \min\left\{\phi^{n-1}_{ijk},\ \min_a \phi^{*}_{ijk}(a),\ \min_a \phi^{*,x}_{ijk}(a),\ \min_a \phi^{*,y}_{ijk}(a),\ \min_a \phi^{*,z}_{ijk}(a)\right\}. \tag{63}$$

The level set of the solution created using this update rule is seen in magenta in Fig. 16b. Note that while the edges are captured fairly sharply, the corners are still rounded off.

In order to capture the corners sharply, we need to consider derivative approximations in the directions pointing toward the corners. It is a happy coincidence in 2D, that we can use a single rotation to capture all four corners of the square, since the vectors $(1, 1)$ and $(1, -1)$ are orthogonal. In 3D, the vectors that point to alternate corners of the cube are no longer orthogonal; for example, $(1, 1, 1)$ is not orthogonal to $(1, -1, 1)$. Thus the rotation which captures the corners along the directions $(\pm 1, \pm 1, \pm 1)$, will not capture any of the other corners. Hence, if we want to capture all corners, we need to use four separate rotated approximations to the derivatives.

We will describe the rotation that captures the corners in the directions of $(\hat{\imath}, \hat{\jmath}, \hat{k}) = (1, -1, 1)$, detailing every step along the way. To orient the $\overline{x}$-axis toward $(1, -1, 1)$, we first rotate about the z-axis by an angle of $\beta = -\pi/4$, and then about the line $x = y$ by an angle of $\gamma = \arctan(1/\sqrt{2})$. The matrix that accomplishes this transformation is

$$U = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{2} & -1/\sqrt{6} \\ -1/\sqrt{3} & 1/\sqrt{2} & 1/\sqrt{6} \\ 1/\sqrt{3} & 0 & 2/\sqrt{6} \end{pmatrix} \tag{64}$$

Following the computations above, the new grid directions are

$$\begin{aligned} &\text{positive } \overline{x} \text{ direction: } (1, -1, 1), \\ &\text{positive } \overline{y} \text{ direction: } (1, 1, 0), \\ &\text{positive } \overline{z} \text{ direction: } (-1, 1, 2), \end{aligned} \tag{65}$$

and the rotated coefficient functions—which in this case depend only on a—are

$$\overline{f}_1(a) = \frac{1}{\sqrt{3}}(a_1 - a_2 + a_3), \qquad \overline{f}_2(a) = \frac{1}{\sqrt{2}}(a_1 + a_2), \qquad \overline{f}_3(a) = \frac{1}{\sqrt{6}}(-a_1 + a_2 + 2a_3), \tag{66}$$

and thus the rotated equation is

$$-1 = \inf_{a_i \in \{\pm 1\}} \left\{ \left(\frac{a_1 - a_2 + a_3}{\sqrt{3}} \right) \phi_{\overline{x}} + \left(\frac{a_1 + a_2}{\sqrt{2}} \right) \phi_{\overline{y}} + \left(\frac{-a_1 + a_2 + 2a_3}{\sqrt{6}} \right) \phi_{\overline{z}} \right\}. \tag{67}$$

At grid points (x_i, y_j, z_k), the upwind derivative approximations are given by

$$\begin{aligned}
(\overline{f}_1(a)\phi_{\overline{x}})_{ijk} &= \frac{|a_1 - a_2 + a_3|}{\sqrt{3}} \frac{\phi_{i+\text{sign}(a_1-a_2+a_3),\, j-\text{sign}(a_1-a_2+a_3),k+\text{sign}(a_1-a_2+a_3)} - \phi_{ijk}}{\Delta \overline{x}}, \\
(\overline{f}_2(a)\phi_{\overline{y}})_{ijk} &= \frac{|a_1 + a_2|}{\sqrt{2}} \frac{\phi_{i+\text{sign}(a_1+a_2),\, j+\text{sign}(a_1+a_2),k} - \phi_{ijk}}{\Delta \overline{y}}, \\
(\overline{f}_3(a)\phi_{\overline{z}})_{ijk} &= \frac{|-a_1 + a_2 + 2a_3|}{\sqrt{6}} \frac{\phi_{i-\text{sign}(-a_1+a_2+2a_3),\, j+\text{sign}(-a_1+a_2+2a_3),k+2\text{sign}(-a_1+a_2+2a_3)} - \phi_{ijk}}{\Delta \overline{z}},
\end{aligned} \tag{68}$$

where the new grid parameters are given by

$$\begin{aligned}
\Delta \overline{x} &= \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}, \\
\Delta \overline{y} &= \sqrt{(\Delta x)^2 + (\Delta y)^2}, \\
\Delta \overline{z} &= \sqrt{(\Delta x)^2 + (\Delta y)^2 + (2\Delta z)^2}.
\end{aligned} \tag{69}$$

Plugging these into the equation gives the upwind approximation to the equation at grid points:

$$\begin{aligned}
\phi^{*,+-+}_{ijk}(a) = \Bigg\{ & 1 + \frac{|a_1 - a_2 + a_3|}{\sqrt{3}\Delta x} \phi_{i+\text{sign}(a_1-a_2+a_3),\, j-\text{sign}(a_1-a_2+a_3),k+\text{sign}(a_1-a_2+a_3)} \\
& + \frac{|a_1 + a_2|}{\sqrt{2}\Delta \overline{y}} \phi_{i+\text{sign}(a_1+a_2),\, j+\text{sign}(a_1+a_2),k} \\
& + \frac{|-a_1 + a_2 + 2a_3|}{\sqrt{6}\Delta \overline{z}} \phi_{i-\text{sign}(-a_1+a_2+2a_3),\, j+\text{sign}(-a_1+a_2+2a_3),k+2\text{sign}(-a_1+a_2+2a_3)} \Bigg\} \\
& \Bigg/ \left\{ \frac{|a_1 - a_2 + a_3|}{\sqrt{3}\Delta \overline{x}} + \frac{|a_1 + a_2|}{\sqrt{2}\Delta \overline{y}} + \frac{|-a_1 + a_2 + 2a_3|}{\sqrt{6}\Delta \overline{z}} \right\},
\end{aligned} \tag{70}$$

where the superscript $+-+$ denotes the fact that the $\overline{x}$-axis points at the corners along the line parallel to $(1, -1, 1)$. Finally, we can include this approximation, and iterate using the update rule

$$\begin{aligned}
\phi^n_{ijk} = \min \Big\{ & \phi^{n-1}_{ijk}, \quad \min_a \phi^*_{ijk}(a), \quad \min_a \phi^{*,x}_{ijk}(a), \quad \min_a \phi^{*,y}_{ijk}(a), \quad \min_a \phi^{*,z}_{ijk}(a), \\
& \min_a \phi^{*,+-+}_{ijk}(a) \Big\}
\end{aligned} \tag{71}$$

In doing so, we will capture all of the edges of the level set fairly well, and perfectly capture the corners in the directions of $(1, -1, 1)$. This is demonstrated by the cyan level set in Fig. 16c. Note that the remaining corners, along the directions $(1, 1, 1)$, $(1, 1, -1)$ and $(1, -1, -1)$ are still rounded, while the corners along $(1, -1, 1)$ are sharp.

Table 5 Maximal error in our approximation to the solution of $\|\nabla\phi\|_1 = 1$ resulting from the update rules (61) (no rotated directions), (63) (rotations which keep one axis fixed), (71) (an additional rotation to resolve the corners along $(1, -1, 1)$), and (72) (rotations to capture all the corners)

Update rule	(61)	(63)	(71)	(72)
Max error	1.0429e−01	4.2424e−02	3.9865e−02	1.4433e−14

Each approximation was computed on a $201 \times 201 \times 201$ grid. Notice that when all corners and edges are accounted for, the solution is accurate to machine precision

Finally, if we want to perfectly capture all corners, we simply need to devise similar upwind approximations $\phi_{ijk}^{*,+++}(a)$, $\phi_{ijk}^{*,++-}(a)$, $\phi_{ijk}^{*,+--}(a)$, and include these using the update rule

$$\phi_{ijk}^{n} = \min\Big\{\phi_{ijk}^{n-1},\ \min_a \phi_{ijk}^{*}(a),\ \min_a \phi_{ijk}^{*,x}(a),\ \min_a \phi_{ijk}^{*,y}(a),\ \min_a \phi_{ijk}^{*,z}(a),\ \min_a \phi_{ijk}^{*,+++}(a),\ \min_a \phi_{ijk}^{*,++-}(a),\ \min_a \phi_{ijk}^{*,+--}(a)\min_a,\ \phi_{ijk}^{*,+--}(a)\Big\} \tag{72}$$

The level set of the solution resulting from this update rule is shown in yellow in Fig. 16d. In this case, the level set is (to machine precision) a perfect cube, with all corners and edges sharp.

Lastly, Table 5 documents the maximal error in the numerical solution resolved using each update rule (61),(63),(71),(72). As expected, including more approximations to the derivatives in additional directions only improves the accuracy. In the last trial, when we perfectly capture all edges and corners, the method is accurate to machine precision.

References

1. Alton, K., Mitchell, I. M.: Optimal path planning under different norms in continuous state spaces. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, pp. 866–872 (2006)
2. Alton, K., Mitchell, I.M.: Fast marching methods for stationary Hamilton-Jacobi equations with axis-aligned anisotropy. SIAM J. Numer. Anal. **47**(1), 363–385 (2009)
3. Bardi, M., Capuzzo-Dolcetta, I.: Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations. Modern Birkhäuser Classics, Birkhäuser Boston (2008)
4. Barles, G., Souganidis, P.E.: Convergence of approximation schemes for fully nonlinear second order equations. Asymp. Anal. **4**, 271–283 (1991)
5. Barles, G.: An Introduction to the Theory of Viscosity Solutions for First-Order Hamilton-Jacobi Equations and Applications, pp. 49–109. Springer, Berlin (2013)
6. Barles, G., Jakobsen, E.R.: Error bounds for monotone approximation schemes for Hamilton-Jacobi-Bellman equations. SIAM J. Numer. Anal. **43**(2), 540–558 (2005)
7. Bellman, R.: The theory of dynamic programming. Technical report, Rand Corp, Santa Monica, CA (1954)
8. Bellman, R.: Adaptive Control Processes: A Guided Tour. Karreman Mathematics Research Collection. Princeton Legacy Library, Princeton University Press, Princeton (1961)
9. Bornemann, F., Rasch, C.: Finite-element discretization of static Hamilton-Jacobi equations based on a local variational principle. Comput. Vis. Sci. **9**(2), 57–69 (2006)
10. Boué, 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)
11. Caffarelli, L.A., Crandall, M.G.: Distance functions and almost global solutions of Eikonal equations. Comm. Partial Differential Equations **35**(3), 391–414 (2010)

12. Chow, Y.T., Darbon, J., Osher, S., Yin, W.: Algorithm for overcoming the curse of dimensionality for state-dependent Hamilton-Jacobi equations. J. Comput. Phys. **387**, 376–409 (2019)
13. Crandall, M.G., Ishii, H., Lions, P.-L.: User's guide to viscosity solutions of second order partial differential equations. Bull. Am. Math. Soc. **27**(1), 1–67 (1992)
14. Crandall, M.G., Lions, P.-L.: Two approximations of solutions of Hamilton-Jacobi equations. Math. Comput. **43**(167), 1–19 (1984)
15. Crandall, M.G., Lions, P.-L.: Viscosity solutions of Hamilton-Jacobi equations. Trans. Am. Math. Soc. **277**(1), 1–42 (1983)
16. Danggo, M.Y., Mungkasi, S.: A staggered grid finite difference method for solving the elastic wave equations. J. Phys: Conf. Ser. **909**, 012047 (2017)
17. Darbon, J., Osher, S.: Algorithms for overcoming the curse of dimensionality for certain Hamilton-Jacobi equations arising in control theory and elsewhere. Research in the Mathematical Sciences **3**(1), 19 (2016)
18. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Am. J. Math. **79**(3), 497–516 (1957)
19. Engquist, B., Froese, B.D., RichardTsai, Y.-H.: Fast sweeping methods for hyperbolic systems of conservation laws at steady state. J. Comput. Phys. **255**, 316–338 (2013)
20. Engquist, B., Froese, B.D., Richard Tsai, Y.-H.: Fast sweeping methods for hyperbolic systems of conservation laws at steady state II. J. Comput. Phys. **286**, 70–86 (2015)
21. Evans, L.C.: An introduction to mathematical optimal control theory version 0.2. *Lecture notes available online*
22. Gao, K., Huang, L.: An improved rotated staggered-grid finite-difference method with fourth-order temporal accuracy for elastic-wave modeling in anisotropic media. J. Comput. Phys. **350**, 361–386 (2017)
23. Jiang, G.-S., Peng, D.: Weighted ENO schemes for Hamilton-Jacobi equations. SIAM J. Sci. Comput. **21**(6), 2126–2143 (2000)
24. Kao, C.Y., Navasca, C., Osher, S.: The Lax-Friedrichs sweeping method for optimal control problems in continuous and hybrid dynamics. Nonlinear Anal. Theory Methods Appl. **63**(5), 1561–1572 (2005). Invited Talks from the Fourth World Congress of Nonlinear Analysts (WCNA 2004)
25. Kao, C.Y., Osher, S., Qian, J.: Lax-Friedrichs sweeping scheme for static Hamilton-Jacobi equations. J. Comput. Phys. **196**(1), 367–391 (2004)
26. Kao, C.-Y., Osher, S., Tsai, Y.-H.: Fast sweeping methods for static Hamilton-Jacobi equations. SIAM J. Numer. Anal. **42**(6), 2612–2632 (2005)
27. Kao, C.-Y., Tsai, R.: Properties of a level set algorithm for the visibility problems. J. Sci. Comput. **35**, 170–191 (2008)
28. Lin, A.T., Chow, Y.T., Osher, S.J.: A splitting method for overcoming the curse of dimensionality in Hamilton-Jacobi equations arising from nonlinear optimal control and differential games with applications to trajectory generation. Commun. Math. Sci. **16**(7), 1 (2018)
29. Luo, S.: A uniformly second order fast sweeping method for Eikonal equations. J. Comput. Phys. **241**, 104–117 (2013)
30. Luo, S., Leung, S., Qian, J.: An adjoint state method for numerical approximation of continuous traffic congestion equilibria. Commun. Comput. Phys. **10**, 11 (2011)
31. Luo, S., Qian, J., Burridge, R.: High-order factorization based high-order hybrid fast sweeping methods for point-source Eikonal equations. SIAM J. Numer. Anal. **52**(1), 23–44 (2014)
32. Luo, S., Qian, J., Stefanov, P.: Adjoint state method for the identification problem in SPECT: Recovery of both the source and the attenuation in the attenuated x-ray transform. SIAM J. Imag. Sci. **7**(2), 696–715 (2014)
33. Luo, S., Zhao, H.: Convergence analysis of the fast sweeping method for static convex Hamilton-Jacobi equations. Res. Math. Sci. **3**(1), 35 (2016)
34. Oberman, A., Salvador, T.: A partial differential equation obstacle problem for the level set approach to visibility. J. Sci. Comput. **82**(1), 14 (2020)
35. Oberman, A.M., Takei, R., Vladimirsky, A.: Homogenization of metric Hamilton-Jacobi equations. Multiscale Model. Simul. **8**(1), 269–295 (2009)
36. Osher, S.: A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations. SIAM J. Math. Anal. **24**(5), 1145–1152 (1993)
37. Osher, S., Fedkiw, R.P.: *Level set methods and dynamic implicit surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer (2003)
38. Osher, S., Shu, C.-W.: High order essentially non-oscillatory schemes for Hamilton-Jacobi equations. SIAM J. Numer. Anal. **28**(4), 907–922 (1991)
39. Oster, G.F., Wilson, E.O.: Caste and ecology in the social insects. Princeton University Press, Princeton (1978)

40. Parkinson, C., Bertozzi, A. L., Osher, S. J.: A Hamilton-Jacobi formulation for time-optimal paths of rectangular nonholonomic vehicles. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 4073–4078 (2020)
41. Parkinson, C., Arnold, D., Bertozzi, A.L., Chow, Y.T., Osher, S.: Optimal human navigation in steep terrain: a Hamilton-Jacobi-Bellman approach. Commun. Math. Sci. **17**(1), 227–242 (2019)
42. Pham, H.: Continuous-time Stochastic Optimal Control and Optimization with Financial Applications, 1st edn. Springer-Verlag, Berlin Heidelberg (2009)
43. Qian, J., Zhang, Y.-T., Zhao, H.-K.: A fast sweeping method for static convex Hamilton-Jacobi equations. J. Sci. Comput. **31**(1–2), 237–271 (2007)
44. Qian, J., Zhang, Y.-T., Zhao, H.-K.: Fast sweeping methods for Eikonal equations on triangular meshes. SIAM J. Numer. Anal. **45**(1), 83–107 (2007)
45. Qin, L., Sui-Bo, M., Bin, Z., Wei, Z.: An improved rotated staggered grid finite difference scheme in coal seam. Appl. Geophys. (2019)
46. Reeds, J.A., Shepp, L.A.: Optimal paths for a car that goes both forwards and backwards. Pacific J. Math. **145**(2), 367–393 (1990)
47. Saenger, E.H., Gold, N., Shapiro, S.A.: Modeling the propagation of elastic waves using a modified finite-difference grid. Wave Motion **31**(1), 77–92 (2000)
48. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. **93**(4), 1591–1595 (1996)
49. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations. Proc. Natl. Acad. Sci. **98**(20), 11069–11074 (2001)
50. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. SIAM J. Numer. Anal. **41**(1), 325–363 (2003)
51. Shu., C.-W.: High order numerical methods for time dependent Hamilton–Jacobi equations. In *Mathematics and computation in imaging science and information processing*, pp. 47–91. World Scientific, New York (2007)
52. Sonneborn, L.M., Van Vleck, F.S.: The bang-bang principle for linear control systems. J. Soc. Ind. Appl. Math. Ser. A: Control **2**(2), 151–159 (1964)
53. Souganidis, P.E.: Approximation schemes for viscosity solutions of Hamilton-Jacobi equations. J. Differential Equations **59**(1), 1–43 (1985)
54. Takei, R., Tsai, R., Shen, H., Landa, Y.: A practical path-planning algorithm for a simple car: a Hamilton-Jacobi approach. In *Proceedings of the 2010 American Control Conference*, pp. 6175–6180 (2010)
55. Takei, R., Tsai, R.: Optimal trajectories of curvature constrained motion in the Hamilton-Jacobi formulation. J. Sci. Comput. **54**(2), 622–644 (2013)
56. Tsai, Y.-H.R., Cheng, L.-T., Osher, S., Burchard, P., Sapiro, G.: Visibility and its dynamics in a PDE based implicit framework. J. Comput. Phys. **199**(1), 260–290 (2004)
57. Richard Tsai, Y.-H., 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)
58. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Trans. Autom. Control **40**(9), 1528–1538 (1995)
59. Wang, K., Peng, S., Yongxu, L., Cui, X.: The velocity-stress finite-difference method with a rotated staggered grid applied to seismic wave propagation in a fractured medium. Geophysics **85**(2), T89–T100 (2020)
60. Weiguo, W., Chen, H., Woo, P.-Y.: Time optimal path planning for a wheeled mobile robot. J. Robot. Syst. **17**(11), 585–591 (2000)
61. Yang, L., Yan, H., Liu, H.: Optimal rotated staggered-grid finite-difference schemes for elastic wave modeling in TTI media. J. Appl. Geophys. **122**, 40–52 (2015)
62. Zhang, Y.-T., Zhao, H.-K., Qian, J.: High order fast sweeping methods for static Hamilton-Jacobi equations. J. Sci. Comput. **29**(1), 25–56 (2006)
63. Zhao, H.-K., Osher, S., Merriman, B., Kang, M.: Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. Comput. Vis. Image Underst. **80**(3), 295–314 (2000)
64. Zhao, H.: A fast sweeping method for Eikonal equations. Math. Comput. **74**(250), 603–627 (2005)
65. Zhou, Z., Ding, J., Huang, H., Takei, R., Tomlin, C.: Efficient path planning algorithms in reach-avoid problems. Automatica **89**, 28–36 (2018)