Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 196 (2004) 367–391
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

# Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations ☆

Chiu Yen Kao, Stanley Osher, Jianliang Qian *

*Department of Mathematics, University of California Los Angeles, Los Angeles, CA 90095, USA*

Received 4 August 2003; received in revised form 5 November 2003; accepted 10 November 2003

**Abstract**

We propose a simple, fast sweeping method based on the Lax–Friedrichs monotone numerical Hamiltonian to approximate viscosity solutions of arbitrary static Hamilton–Jacobi equations in any number of spatial dimensions. By using the Lax–Friedrichs numerical Hamiltonian, we can easily obtain the solution at a specific grid point in terms of its neighbors, so that a Gauss–Seidel type nonlinear iterative method can be utilized. Furthermore, by incorporating a group-wise causality principle into the Gauss–Seidel iteration by following a finite group of characteristics, we have an easy-to-implement, sweeping-type, and fast convergent numerical method. However, unlike other methods based on the Godunov numerical Hamiltonian, some computational boundary conditions are needed in the implementation. We give a simple recipe which enforces a version of discrete min–max principle. Some convergence analysis is done for the one-dimensional eikonal equation. Extensive 2-D and 3-D numerical examples illustrate the efficiency and accuracy of the new approach. To our knowledge, this is the first fast numerical method based on discretizing the Hamilton–Jacobi equation directly without assuming convexity and/or homogeneity of the Hamiltonian.


## 1. Introduction

The Hamilton–Jacobi equation arises in many applications such as geometrical optics, crystal growth, etching, computer vision, obstacle navigation, path planning, photolithography, and seismology. Viscosity solutions of these nonlinear differential equations usually develop singularities in their derivatives even with smooth initial conditions [3]. Numerically, in general, one looks for a consistent, convergent, e.g., monotone scheme to approximate such viscosity solutions [19].

In this paper, we focus on static Hamilton–Jacobi equations of the following form:

☆ Research supported by ONR MURI N00014-02-1-0720.
* Corresponding author. Tel.: +(310)-825-4746; fax: +(310)-206-6673.
*E-mail addresses:* ckao@math.ucla.edu (C.Y. Kao), sjo@math.ucla.edu (S. Osher), qian@math.ucla.edu (J. Qian).

0021-9991/$ - see front matter 
doi:10.1016/j.jcp.2003.11.007

$$\begin{cases} H(x, \nabla\phi(x)) = R(x), & x \in \Omega, \\ \phi(x) = q(x), & x \in \Gamma, \end{cases} \tag{1}$$

where $H$, $q$, and $R > 0$ are Lipschitz continuous, and $\Gamma$ is a subset of $\Omega$.

This kind of static, first-order nonlinear PDEs appears in many different applications. In the Dynamic Programming approach for infinite horizon optimal control problems, the value function of the optimized cost functional satisfies the so-called Hamilton–Jacobi–Bellman equation, a static equation having a convex Hamiltonian in the gradient variable. In the Dynamic Programming approach for differential games, the value function for the zero-sum game satisfies the so-called Hamilton–Jacobi–Isaacs equation, the resulting Hamiltonian being nonconvex. In the classical high frequency asymptotics for wave propagation in elastic solids, the phase function, a.k.a traveltime function in some applications, satisfies the so-called eikonal equation which is an indispensable element of the family of Hamilton–Jacobi equations. To be more specific, in an isotropic elastic solid, the traveltime satisfies the isotropic eikonal equation $|\nabla\phi| = 1$ which is of quadratic nonlinearity and thus has a convex Hamiltonian. However, in an anisotropic elastic solid, high frequency asymptotics gives rise to three different wave modes: one quasi-longitudinal wave and two quasi-shear waves. The Hamilton–Jacobi equation for quasi-longitudinal wave traveltime is convex in the gradient variable and homogeneous of degree one, but the Hamilton–Jacobi equation for one of the two shear waves is nonconvex in the gradient variable. In the semi-classical limit for Schröedinger equation, an eikonal equation arises as the Planck constant approaches zero. Therefore, it is of fundamental importance to design fast, accurate numerical schemes to solve resulting static Hamilton–Jacobi equations for the above applications.

Numerical methods for this type of equations can be roughly divided into three categories. The first class of methods are those that are based on the monotonicity of the solution along the characteristics [5,17,18,21]. The solutions are constructed by combining variations of the classical Dijkstra algorithm and heap-sort data structures. The complexity is $O(N \log N)$, where $N$ is the total number of grid points in the domain. So far, these methods can only handle convex, usually homogeneous-of-degree-one, Hamiltonians and become quite complicated with large initialized regions and cumbersome updating formulae if the Hamiltonian is not closely related to that of the eikonal equation, i.e., close to $H(x, \nabla\phi) = |\nabla\phi|$.

The second class of methods are those that rely on time-dependent Hamilton–Jacobi equations. The advantage of these methods is that higher-order schemes are easily derived. Osher [8] provided a rigorous link between static and time-dependent Hamilton–Jacobi equations. The zero-level set of the viscosity solution $\psi$ of

$$\psi_t(x,t) + H(x, \nabla\psi(x,t)) = 0 \tag{2}$$

with suitable initial conditions at a later time $t$ is the set of $x$ such that $\phi(x) = t$ of (1). Therefore one can first solve the time-dependent equation (2) by a localized level set formulation [9,11] with high-order approximations to partial derivatives [6,10] and recover the solution of (1) through finding zero-level sets. Another approach to obtaining a "time"-dependent Hamilton–Jacobi equation from the static Hamilton–Jacobi equation is using a so-called paraxial formulation by assuming that there is a preferred direction in the characteristic propagation. In [4], a paraxial formulation was first proposed for the eikonal equation $|\nabla\phi| = 1$. Later in [14,15], a paraxial formulation was proposed for convex Hamilton–Jacobi equations which is efficient in geophysical applications and optical instruments.

The third, and final, class of algorithms rely on iteration strategies. Rouy and Tourin [16] first used an upwind, monotone, and consistent discretization for $|\nabla\phi|$ to solve the discretized eikonal equation iteratively, and they also proved that the resulting algorithm converges to the viscosity solution. Boué and Dupuis [2] designed Markov chain-based algorithms for some deterministic control problems, in which Hamilton–Jacobi equations have affine and quadratic Hamiltonians, and they found out that Gauss–Seidel type iterations with appro-priate sweeping orders may lead to an $O(N)$ algorithm, where $N$ is the total

number of grid points in the domain. For more general convex Hamiltonians, Tsai et al. [20] combined such a Gauss–Seidel type iteration method with the monotone upwind Godunov Hamiltonian [1,10], and the resulting algorithm is very easy to implement and accurate. To obtain an efficient 3-D algorithm, Kao et al. [7] interpreted the monotone upwind Godunov Hamiltonian in terms of the Legendre transform so that a Gauss–Seidel type fast sweeping method can be easily designed. Computationally both of these two methods have $O(N)$ complexity. This has been proved in [22] for special cases, but the numerical evidence is convincing that this is true for general convex cases.

However, all of the above cited methods are designed for static Hamilton–Jacobi equations by assuming convexity and/or homogeneity of Hamiltonians. In this paper, we propose a new Gauss–Seidel sweeping type algorithm which is based on the Lax–Friedrichs Hamiltonian. It can handle both convex and non-convex Hamiltonians, no matter how complicated they are. Since the evaluation of nonlinear Hamiltonians $H$ uses data from the previous step, this makes the speed of the algorithm dramatically fast. The algorithm can deal with boundary conditions specified on arbitrary subsets $\Gamma$. Finally the overall algorithm is extremely easy to implement, using less than 100 lines of code.

## 2. The Lax–Friedrichs sweeping scheme

If a monotone scheme based on the Godunov Hamiltonian is applied to Eq. (1), then a nontrivial calculation involving minima and maxima needs to be carried out at each grid point to solve for a grid value in terms of its neighbors. This can be done without too much difficulty for convex Hamiltonians. For example, the ordered upwind method [18] updates the grid value using a minimization formula which essentially boils down to a version of Godunov type monotone schemes for convex and homogeneous Hamiltonians. To update the solution at each grid point, the ordered upwind method searches the whole "considered" front in order to find an approximately correct direction to satisfy the point-wise causality; thus this can involve an extensive, computationally costly search, and the resulting method has $O(N \log N)$ complexity, where $N$ is the total number of grid points. An optimal method of $O(N)$ complexity may be designed by following a group of characteristics at each iteration, so that a group-wise causality principle is satisfied. This led to the design of the fast sweeping methods proposed in [7,20,22], and these methods for convex Hamilton–Jacobi equations apparently have only $O(N)$ complexity and are simple to implement.

However, if the Hamiltonian in (1) is nonconvex, then the Godunov Hamiltonian gives rise to a formula involving minima and maxima which is extremely hard to carry out; therefore, in this case it is a nontrivial task to solve for a grid value in terms of its neighbors. Hence, we resort to using the Lax–Friedrichs Hamiltonian to avoid a complicated optimization process always needed for the Godunov Hamiltonian.

### 2.1. Lax–Friedrichs Hamiltonian

Our new numerical algorithm for static Hamilton–Jacobi equations

$$\begin{cases} H(x, \nabla\phi(x)) = R(x), & x \in \Omega, \\ \phi(x) = q(x), & x \in \Gamma \end{cases}$$

consists of an update formula based on the Lax–Friedrichs Hamiltonian and a sweeping process so that it can handle both convex and nonconvex cases.

For illustration purposes, we start with the 1-D case. The 1-D Lax–Friedrichs Hamiltonian is (dropping the obvious $x$ dependence on $H$)

$$\widetilde{H}^{\mathrm{LF}}(p^-, p^+) = H\left(\frac{p^+ + p^-}{2}\right) - \sigma_x \frac{p^+ - p^-}{2},$$

where $p = \partial\phi/\partial x$, $p^{\pm}$ are corresponding forward and backward difference approximations of $\partial\phi/\partial x$, and $[A, B]$ is the range of $p^{\pm}$; $\sigma_x$ is the artificial viscosity satisfying

$$\sigma_x \geqslant \max_{p\in[A,B]} \left|\frac{\partial H}{\partial p}\right|.$$

Consider an uniform discretization $\{x_i,\ i = 1,\ldots,m\}$ of $\Omega$ with grid size $\Delta x$. At mesh points $x_i$ we have the following numerical approximation:

$$\widetilde{H}^{\mathrm{LF}} = R,$$

which implies

$$H\left(x_i, \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x}\right) - \sigma_x \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{2\Delta x} = R(x_i).$$

In order to have a simple update formula, we solve for $\phi_i$ in the above,

$$\phi_i^{n+1} = \frac{\Delta x}{\sigma_x}\left(R(x_i) - H\left(x_i, \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x}\right)\right) + \frac{\phi_{i+1} + \phi_{i-1}}{2}. \tag{3}$$

We on purpose did not put superscripts on $\phi_{i+1}$ and $\phi_{i-1}$ since those superscripts depend on the sweeping directions of alternating Gauss–Seidel type iterations. If we sweep from left to right, $\phi_{i-1} = \phi_{i-1}^{n+1}$ and $\phi_{i+1} = \phi_{i+1}^{n}$ because we use the newest values for Gauss–Seidel iteration. Of course the opposite is true if we sweep from right to left.

The formulas for two and three dimensions can be obtained by similar procedures:

$$\begin{aligned}\phi_{i,j}^{n+1} =& \left(\frac{1}{\frac{\sigma_x}{\Delta x} + \frac{\sigma y}{\Delta y}}\right)\left(R - H\left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}, \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y}\right)\right)\\ &+ \left(\frac{1}{\frac{\sigma_x}{\Delta x} + \frac{\sigma_y}{\Delta y}}\right)\left(\sigma_x \frac{\phi_{i+1,j} + \phi_{i-1,j}}{2\Delta x} + \sigma_y \frac{\phi_{i,j+1} + \phi_{i,j-1}}{2\Delta y}\right),\end{aligned} \tag{4}$$

$$\begin{aligned}\phi_{i,j,k}^{n+1} = c\Bigg\{& R - H\left(\frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x}, \frac{\phi_{i,j+1,k} - \phi_{i,j-1,k}}{2\Delta y}, \frac{\phi_{i,j,k+1} - \phi_{i,j,k-1}}{2\Delta z}\right) + \sigma_x \frac{\phi_{i+1,j,k} + \phi_{i-1,j,k}}{2\Delta x}\\ &+ \sigma_y \frac{\phi_{i,j+1,k} + \phi_{i,j-1,k}}{2\Delta y} + \sigma_z \frac{\phi_{i,j,k+1} + \phi_{i,j,k-1}}{2\Delta z}\Bigg\},\end{aligned} \tag{5}$$

where $q = \partial\phi/\partial y$ and $r = \partial\phi/\partial z$,

$$c = \frac{1}{\frac{\sigma_x}{\Delta x} + \frac{\sigma_y}{\Delta_y} + \frac{\sigma_z}{\Delta_z}}.$$

and $\sigma_x$, $\sigma_y$ and $\sigma_z$ are artificial viscosities satisfying

$$\sigma_x \geqslant \max\left|\frac{\partial H}{\partial p}\right|, \quad \sigma_y \geqslant \left|\frac{\partial H}{\partial q}\right|, \quad \sigma_z \geqslant \max\left|\frac{\partial H}{\partial r}\right|.$$

We remark that no nonlinear inversion is required in the above formulae, therefore the algorithm is simple to implement, no matter how complicated the Hamiltonian might be.

### 2.2. Computational boundary condition

There is a major difference between the Godunov Hamiltonian and the Lax–Friedrichs Hamiltonian. The Godunov Hamiltonian is an upwind Hamiltonian while the Lax–Friedrichs Hamiltonian is not. Given that characteristics are assumed to flow out of the regions, the Godunov Hamiltonian will choose the grid points automatically to give reasonable results on the computational boundary. However, the Lax–Friedrichs Hamiltonian gives a solution depending on all of its neighbors in all Cartesian dimensions; therefore we have to specify carefully the values of points outside the computational domain; otherwise huge errors will be introduced for the points on the computational boundary and then propagate into the computational domain.

For simplicity, we detail boundary conditions in the 2-D case only. Consider a rectangular domain $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ with a uniform discretization $(x_i, y_j)$, $i = 0, 1, \ldots, m_1, m_1 + 1$, and $j = 0, 1, \ldots, m_2, m_2 + 1$, where $x_i = (i-1)\Delta z + x_{\min}$, $y_j = (j-1)\Delta y + y_{\min}$, $\Delta x = (x_{\max} - x_{\min})/(m_1 - 1)$ and $\Delta y = (y_{\max} - y_{\min})/(m_2 - 1)$.

On the four sides of the boundary we impose following conditions:

$$\begin{cases} \phi^{\text{new}}_{0,j} = \min(\max(2\phi_{1,j} - \phi_{2,j}, \phi_{2,j})\phi^{\text{old}}_{0,j}), \\ \phi^{\text{new}}_{m_1+1,j} = \min(\max(2\phi_{m_1,j} - \phi_{m_1-1,j}, \phi_{m_1-1,j})\phi^{\text{old}}_{m_1+1,j}), \\ \phi^{\text{new}}_{i,0} = \min(\max(2\phi_{i,1} - \phi_{i,2}, \phi_{i,2})\phi^{\text{old}}_{i,0}), \\ \phi^{\text{new}}_{i,m_2+1,j} = \min(\max(2\phi_{i,m_2} - \phi_{i,m_2-1}, \phi_{i,m_2-1}), \phi^{\text{old}}_{i,m_2+1}), \end{cases} \tag{6}$$

which combine extrapolation, maximization and minimization to calculate the values for points outside the computational domain.

The above formula is based on the following reasoning. For a source point not on the computational boundary, it is reasonable to do linear extrapolation, implying that $p^+ = p^-$ for the points on the computational boundary. For a source point on the computational boundary, we need to choose $\partial\phi/\partial n = 0$ in order to avoid inflows. For a well-posed problem, in the absence of physically prescribed boundary conditions, we must always have outflows on the computational boundary. That is why we choose

$$\left(\frac{\partial\phi}{\partial x}\right)^+ \simeq \frac{\phi_{2,j} - \phi_{1,j}}{\Delta x} = \left(\frac{\partial\phi}{\partial x}\right)^- \simeq \frac{\phi_{1,j} - \phi_{0,j}}{\Delta x}$$

when

$$\left(\frac{\partial\phi}{\partial x}\right)^+ \simeq \frac{\phi_{2,j} - \phi_{1,j}}{\Delta x} > 0$$

and

$$\left(\frac{\partial\phi}{\partial n}\right)^{\text{c}} \simeq \frac{\phi_{2,j} - \phi_{0,j}}{\Delta x}$$

when

$$\left(\frac{\partial\phi}{\partial x}\right)^+ \approx \frac{\phi_{2,j} - \phi_{1,j}}{\Delta x} \leqslant 0.$$

In addition, we approximate the viscosity solutions starting from some upper values; hence to ensure that our numerical approximation is decreasing after each iteration, we update the value only when it is less than its previous value.

### 2.3. Algorithm for Lax–Friedrichs sweeping

The following Lax–Friedrichs sweeping algorithm is very easy to implement. There are essentially three steps: initialization, alternating sweepings, and enforcing computational boundary conditions. We take the 2-D case for the sake of exposition. Suppose that we have a uniform discretization $(x_i, y_j)$, $i = 0, 1, \ldots, m_1, m_1 + 1$ and $j = 0, 1, \ldots, m_2, m_2 + 1$ as mentioned before. The algorithm is as follows.

1. *Initialization.* We assign exact values or interpolated values $\phi^0_{i,j}$ at grid points on or near the given boundary $\Gamma$, and these values are fixed during the iterations. At all other grid points, we assign large positive values $M$ to $\phi^0_{i,j}$, where $M$ should be larger than the maximum of the true solutions, and these values will be updated in the process of iterations.
2. *Alternating sweepings.* At iteration $n+1$, we calculate $\phi^{n+1}_{i,j}$ according to (4) at all grid points $(x_i, y_j)$ $1 \leqslant i \leqslant m_1$, $1 \leqslant j \leqslant m_2$ except for those which have assigned values and update $\phi^{n+1}_{i,j}$ only when it is less than its previous value $\phi^n_{i,j}$. Recall that this process needs to be done in alternating sweeping directions, which means that it needs four different sweeps in the 2-D case: (i) From lower left to upper right $i = 1 : m_1$, $j = 1 : m_2$; (ii) from lower right to upper left $i = m_1 : 1$, $j = 1 : m_2$; (iii) from upper left to lower right $i = 1 : m_1$, $j = m_2 : 1$; and (iv) from upper right to lower left $i = m_1 : 1$, $j = m_2 : 1$. In general, in the $d$-dimension case, we need $2^d$ alternating sweeps.
3. *Enforcing computational boundary conditions.* After each sweep, we enforce computational boundary conditions according to formula (6), trivially modified depending on which boundary we are at.
4. *Convergence test.* Given convergence criterion $\epsilon > 0$, check whether

$$\|\phi^{n+1} - \phi^n\|_{L_1} \leqslant \epsilon.$$

## 3. Properties for 1-D eikonal equation

For the 1-D eikonal equation, $H(p) = |p|$ and $R = c(x)$, we set $\Delta x = h$ and choose the optimal $\sigma_x = 1$. Thus

$$\phi_i = h\left(c(x_i) - \frac{|\phi_{i+1} - \phi_{i-1}|}{2h}\right) + \frac{\phi_{i+1} + \phi_{i-1}}{2}.$$

Moreover,

$$\phi_i = \begin{cases} hc(x_i) + \phi_{i+1} & \text{if } \phi_{i+1} \leqslant \phi_{i-1}, \\ hc(x_i) + \phi_{i-1} & \text{if } \phi_{i-1} < \phi_{i+1}. \end{cases}$$

It is easy to see that the enforced computational boundary condition will have no effect during the iteration process; in other words, the computational boundary condition is automatically satisfied in this 1-D case. Hence the Lax–Friedrichs sweeping gives exactly the same approximation as the upwind Godunov sweeping does, no matter how many source points we have.

In general, when $\sigma_x > 1$, we have the update formula

$$\phi_i = \begin{cases} \frac{h}{\sigma_x} + \left(\frac{1}{2} + \frac{1}{2\sigma_x}\right)\phi_{i+1} + \left(\frac{1}{2} - \frac{1}{2\sigma_x}\right)\phi_{i-1} & \text{if } \phi_{i+1} \leqslant \phi_{i-1}, \\ \frac{h}{\sigma_y} + \left(\frac{1}{2} + \frac{1}{2\sigma_x}\right)\phi_{i-1} + \left(\frac{1}{2} - \frac{1}{2\sigma_x}\right)\phi_{i+1} & \text{if } \phi_{i-1} < \phi_{i+1}. \end{cases}$$

We first analyze the convergence for the case with a single source point in the center of the domain $[-1, 1]$. Suppose that we have a discretization $x_i = \frac{i}{m}$, $-m - 1 \leqslant i \leqslant m + 1$, $x_{m+1}$ and $x_{-m-1}$ are points outside

the domain, and $\phi(0) = 0$, $x_0 = 0$. We sweep from left to right and then from right to left. The approximation after two sweeps on the left of the center is symmetric to the approximation after one sweep on the right. Without loss of generality, we can just discuss the approximations on the right of the center.

Denote $a = \left(\frac{1}{2} - \frac{1}{2\sigma_x}\right)$ and $b = \left(\frac{1}{2} + \frac{1}{2\sigma_x}\right)$. Let $(\phi^n)^+$ represent the solution by the sweep from left to right at the $n$th iteration and let $(\phi^n)^-$ represent the sweep from right to left at the $n$th iteration. Therefore, we can write down the update formula as the following linear system:

$$A_+(\phi^{n+1})^+ = B_+(\phi^n)^- + C$$

and

$$A_-(\phi^n)^- = B_-(\phi^n)^+ + C,$$

where

$$A_+ = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ -b & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & -b & 1 & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & 0 & -b & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 1 & -2 & 1 \end{bmatrix},$$

$$B_+ = \begin{bmatrix} 0 & a & 0 & \cdots & \cdots & \cdots & 0 \\ \cdots & 0 & a & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0 & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & 0 & a & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 & a \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix},$$

$$A_- = \begin{bmatrix} 1 & -a & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & -a & 0 & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & 1 & -a & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 & 1 & 0 \\ 0 & \cdots & \cdots & \cdots & 1 & -2 & 1 \end{bmatrix},$$

$$B_- = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ b & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & b & 0 & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & b & 0 & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0 & b & 0 & a \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 \end{bmatrix},$$

$$\phi^n = \begin{bmatrix} \phi_1^n \\ \phi_2^n \\ \vdots \\ \phi_m^n \\ \phi_{m+1}^n \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} \frac{h}{\sigma_x} \\ \frac{h}{\sigma_x} \\ \vdots \\ \frac{h}{\sigma_x} \\ \frac{h}{\sigma_x} \\ 0 \end{bmatrix}.$$

Thus the update formula can be written as

$$(\phi^{n+1})^+ = (A_+)^{-1}B_+(\phi^n)^- + (A_+)^{-1}C = \widehat{B_+}(\phi^n)^- + \widehat{C_+}$$

and

$$(\phi^n)^- = (A_-)^{-1}B_-(\phi^n)^+ + (A_-)^{-1}C = \widehat{B_-}(\phi^n)^+ + \widehat{C_-},$$

where

$$\widehat{B_+} = \begin{bmatrix} 0 & a & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & ab & a & \cdots & \cdots & \cdots & \cdots \\ 0 & ab^2 & ab & \cdots & \cdots & \cdots & \cdots \\ \cdots & ab^3 & ab^2 & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & 0 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & a & 0 & \cdots \\ 0 & ab^{m-2} & ab^{m-3} & \cdots & ab & a & 0 \\ 0 & ab^{m-1} & ab^{m-2} & \cdots & ab^2 & ab & a \\ 0 & -ab^{m-2} + 2ab^{m-1} & -ab^{m-3} + 2ab^{m-2} & \cdots & -ab + 2ab^2 & -a + 2ab & 2a \end{bmatrix},$$

$$\widehat{B_-} = \begin{bmatrix} ab & a^2b & a^3b & \cdots & \cdots & \cdots & a^{m-2}b & a^{m-1}b & 0 & a^m \\ b & ab & a^2b & a^3b & \cdots & \cdots & \cdots & a^{m-2}b & 0 & a^{m-1} \\ 0 & b & ab & a^2b & a^3b & \cdots & \cdots & a^{m-3}b & 0 & a^{m-2} \\ \cdots & ab^3 & ab^2 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & b & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a^2b & a^3b & \cdots & a^3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & ab & a^2b & \cdots & a^3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & b & ab & 0 & a^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & b & 0 & a \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & -b & (2-a)b & 0 & (2-a)a \end{bmatrix},$$

$$\widehat{C_+} = \begin{bmatrix} \frac{h}{\sigma_x} \\ \frac{h}{\sigma_x}(1+b) \\ \frac{h}{\sigma_x}(1+b+b^2) \\ \vdots \\ \frac{h}{\sigma_x}(1+b+\cdots+b^{m-3}+b^{m-2}) \\ \frac{h}{\sigma_x}(1+b+\cdots+b^{m-2}+b^{m-1}) \\ \frac{h}{\sigma_x}(2+b+\cdots+b^{m-2}+b^{m-1}) \end{bmatrix},$$

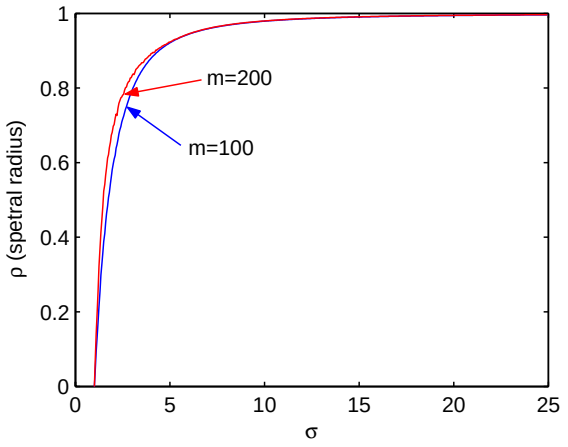



Fig. 1. Spectral radius of $\widehat{B_+}\widehat{B_-}$.

$$\widehat{C} = \begin{bmatrix} \frac{h}{\sigma_x}(1 + a + \cdots + a^{m-2} + a^{m-1}) \\ \frac{h}{\sigma_x}(1 + a + \cdots + a^{m-3} + a^{m-2}) \\ \frac{h}{\sigma_x}(1 + a + \cdots + a^{m-4} + a^{m-3}) \\ \vdots \\ \frac{h}{\sigma_x}(1 + a + a^2) \\ \frac{h}{\sigma_x}(1 + a) \\ \frac{h}{\sigma_x} \\ (1 - a)\frac{h}{\sigma_x} \end{bmatrix}.$$

The algorithm converges for any fixed $m$ if the spectral radius $\rho(\widehat{B_+}\widehat{B_-}) < 1$, since

$$(\phi^{n+1})^+ = \widehat{B_+}\widehat{B_-}(\phi^n)^+ + \widehat{B_+}\widehat{C}_- + \widehat{C_+}.$$

However, it is not obvious to estimate the spectral radius for $\widehat{B_+}\widehat{B_-}$ theoretically. Thus, in Fig. 1, we show the numerical estimation of the spectral radius for $m = 100$ and $m = 200$. As we can see, the spectral radius approaches 1 when $\sigma_x$ goes to infinity, but it vanishes rapidly as $\sigma_x$ decreases to 1.

## 4. Numerical simulation

We apply the Lax–Friedrichs sweeping scheme to Wulff crystal shape problems and traveltime calculations of elastic waves. However, its application is not limited to these two classes of problems. In each of the following example, we consider the iteration convergent if the $L_1$ norm of the difference of two successive iterations $\|\phi^{n+1} - \phi^n\|_{L_1}$ is less than $10^{-10}$. Generally, the algorithm converges within a few hundred iterations. Even though the presented algorithm needs more iterations than the sweeping methods based on the Godunov Hamiltonian, it is still very fast. This is because the Lax–Friedrichs sweeping scheme does not involve any nonlinear inversion at all, let alone a complicated procedure involving many "if" statements.

### 4.1. The Wulff crystal shape

The level set formulation of the Wulff crystal shape problem [12] is

$$\begin{cases} \psi_t + \gamma\left(\frac{\nabla\psi}{|\nabla\psi|}\right)|\nabla\psi| = 0, \quad x \in \mathbb{R}^d, \ t > 0, \\ \psi = 0, \quad x \in \Gamma, \end{cases} \tag{7}$$

where $\gamma$ is the normal speed, also known as the surface tension in the material science. The zero-level set of the viscosity solution $\psi$ of (7) at time $t$ is the viscosity solution $\phi(x,y) = t$ of the following static Hamilton–Jacobi equation [8]

$$\begin{cases} \gamma\left(\frac{\nabla\phi}{|\nabla\phi|}\right)|\nabla\phi| = 1, \quad x \in \mathbb{R}^d, \\ \phi = 0, \quad x \in \Gamma. \end{cases} \tag{8}$$

If $\Gamma$ is a collection of closed surfaces of codimension one, there is no difference between these two formulations. Moreover, if $\Gamma$ is more complicated, (8) gives both inward and outward propagation.

In [12], for 2-D cases $\gamma$ is given as $\gamma(v)$ where $v$ is the angle between the outward normal direction $\frac{\nabla\phi}{|\nabla\phi|}$ and $x$-axis with $-\pi < v \leqslant \pi$. Thus we have $\cos(v) = p/\sqrt{p^2+q^2}$ and $\sin(v) = q/\sqrt{p^2+q^2}$, where $p = \partial\phi/\partial x$ and $p = \partial\phi/\partial y$.

For 3-D cases, $\gamma = \gamma(v,\varphi) = \tilde{\gamma}(v)h(\varphi)$, where $v$ and $\varphi$ are the spherical coordinates: $-\pi < v \leqslant \pi$ and $-\frac{\pi}{2} < \varphi \leqslant \frac{\pi}{2}$. Thus we have

$$\cos(v) = \frac{p}{\sqrt{p^2+q^2}}, \quad \sin(v) = \frac{q}{\sqrt{p^2+q^2}},$$

$$\cos(\varphi) = \frac{\sqrt{p^2+q^2}}{\sqrt{p^2+q^2+r^2}}, \quad \cos(\varphi) = \frac{\sqrt{p^2+q^2}}{\sqrt{p^2+q^2+r^2}},$$

where $p = \partial\phi/\partial x$, $q = \partial\phi/\partial y$ and $r = \partial\phi/\partial z$.

Applying these trigonometric equalities to a given surface tension $\gamma(v)$ or $\gamma = \gamma(v,\varphi)$, we obtain a corresponding Hamilton–Jacobi equation. For example, if

$$\gamma(v) = 1 + \left|\sin\left(v + \frac{\pi}{2}\right)\right|,$$

then we have the corresponding Hamilton–Jacobi equation

$$\sqrt{p^2+q^2} + |p| = 1.$$

In each Wulff crystal shape problem, we specify the normal speed, obtain the corresponding Hamilton–Jacobi equation, and choose the artificial viscosity as small as possible to make the resulting Lax–Friedrichs scheme monotone. In some Wulff crystal shape problems, we have terms such as $\sqrt{p^2+q^2}$ and $\sqrt{p^2+q^2+r^2}$ in the denominator; we regularize them by adding a small quantity $\epsilon$, e.g. $\epsilon = 10^{-6}$ in order to avoid "dividing by zero".

First we apply the scheme to some 2-D problems with three different types of boundary conditions: (A) a single source point at the center of the domain: $\Gamma = \{(0,0)\}$ and $\phi(\Gamma) = 0$; (B) $\Gamma$ being a square and $\phi(\Gamma) = 0$; and (C) 100 random source points: $\Gamma = \{(x_i, y_i), \ 1 \leqslant i \leqslant 100\}$ so that $\phi(x_i, y_i) = 0$.

Fig. 2 illustrates the Wulff crystal shape in the case that $\gamma(v) = 1 + \left|\sin\left(v + \frac{\pi}{2}\right)\right|$ so that the Wulff shape is an ellipse. Fig. 2(1) shows ellipse contours with a single source point (type A boundary condition) at the center of the domain. Fig. 2(2) depicts the Wulff shapes for type B boundary condition so that we have both

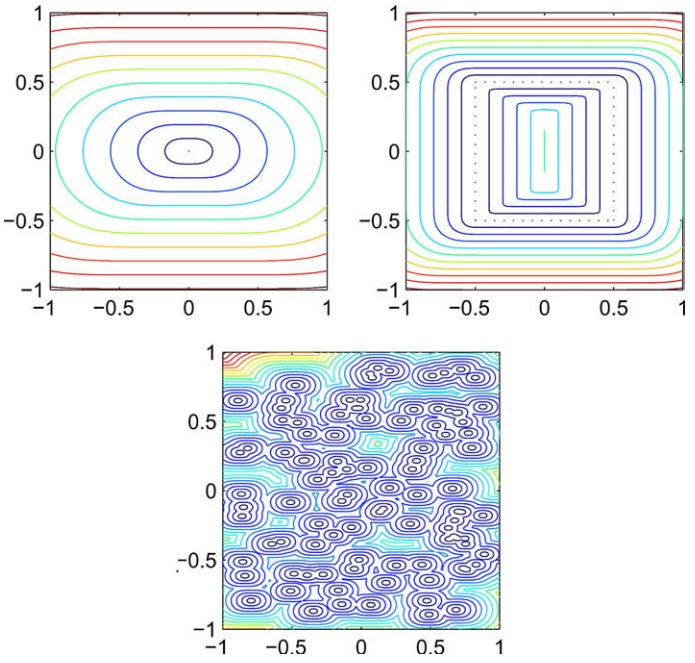



Fig. 2. $|p| + \sqrt{p^2 + q^2} = 1$, $\sigma_x = 2$, $\sigma_y = 1$. (1) A single source point: contour difference = 0.1, 200 × 200 grid, 55 iterations; (2) sources on a square: contour difference = 0.05, 200 × 200 grid, 31 iterations; (3) 100 random source points: contour difference = 0.02, 400 × 400 grid, 36 iterations.

inward and outward propagations. The outward propagation tends to smooth the kink and the contours gradually become ellipses, while the inward propagation makes the contours become vertical ellipses; however, they look more like rectangles because there is no enough space to propagate. Fig. 2(3) presents the Wulff shape for type C boundary condition: 100 random source points interact with each other to give rise to complicated contours.

In Figs. 3 and 4, $\gamma(v) = 1 + |\sin(\frac{3}{2}(v + \frac{\pi}{2}))|$ and $\gamma(v) = 1 + 3|\sin(\frac{3}{2}(v + \frac{\pi}{2}))|$, respectively, but the Wulff crystal shapes are triangles in both cases.

In Figs. 5 and 6, $\gamma(v) = |\cos(v)| + |\sin(v)|$ and $\gamma(v) = 1 + 3|\sin(2v)|$, respectively, and the Wulff crystal shapes are quadrilaterals in both cases.

From these simulations, we see that the more nonconvex the Haniiltonian, the sharper the facets are resolved numerically and the more iterations are needed for convergence.

Fig. 7 illustrates the Wulff shapes in the case that $\gamma(v) = 1 + |\sin(\frac{5}{2}(v + \frac{\pi}{2}))|$, and the corresponding crystal shape is a pentagon.

Fig. 8 shows the Wulff crystal shapes for $\gamma(v) = 1 + |\sin(3(v + \frac{\pi}{2}))|$, and the corresponding crystal shape is a hexagon.

Fig. 9 depicts the Wulff crystal shapes for $\gamma(v) = 1 + |\sin(\frac{7}{2}(v + \frac{\pi}{2}))|$, and the corresponding Wulff crystal shape is a heptahedron.

Fig. 10 presents the case that $\gamma(v) = 1 + |\sin(4v)|$, and the corresponding Wulff crystal shape is an octagon.

Next we apply the scheme to several 3-D examples. For ease of visualization, we only do simulations with a single source point in the center of the domain, even though we can handle very complicated boundary conditions. The contours are plotted with specified differences or values.

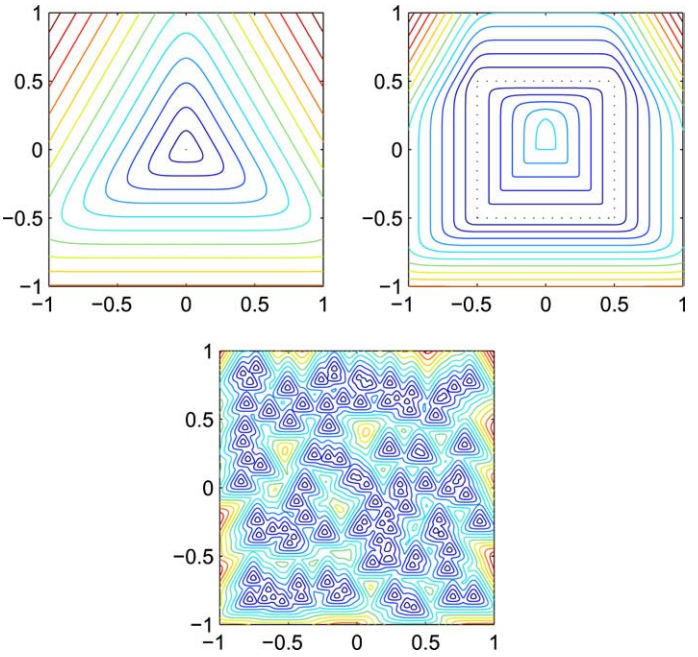



Fig. 3. $\sqrt{p^2+q^2}+\sqrt{\frac{(p^2+q^2)^{3/2}-(3p^2q-q^3)}{2\sqrt{p^2+q^2}}}=1$, $\sigma_x=\sigma_y=2$. (1) A single source point: contour difference = 0.1, $200\times200$ grid, 100 iterations; (2) Sources on a square: contour difference = 0.05, $200\times200$ grid, 68 iterations; (3) 100 random source points: contour difference = 0.02, $400\times400$ grid, 62 iterations.

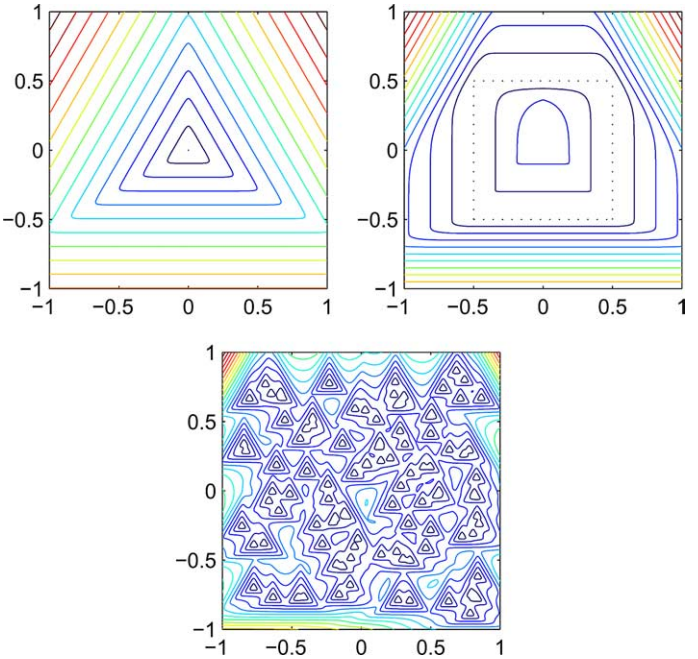



Fig. 4. $\sqrt{p^2+q^2}+\sqrt{\frac{(p^2+q^2)^{3/2}-(3p^2q-q^3)}{2\sqrt{p^2+q^2}}}=1$, $\sigma_x=\sigma_y=4$. (1) A single source point: contour difference = 0.1, $200\times200$ grid, 222 iterations; (2) Sources on a square: contour difference = 0.05, $200\times200$ grid, 137 iterations; (3) 100 random source points: contour difference = 0.02, $400\times400$ grid, 130 iterations.

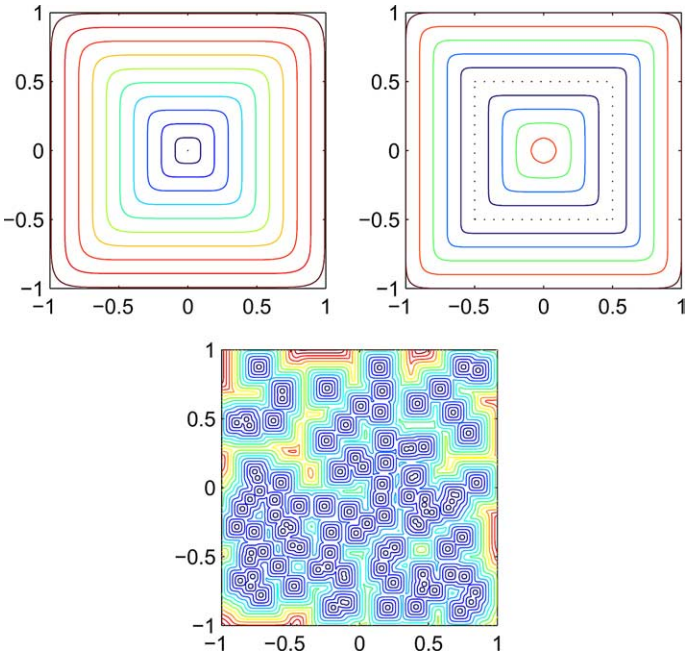



Fig. 5. $|p| + |q| = 1$, $\sigma_x = \sigma_y = 1$. (1) A single source point: contour difference = 0.1, $200 \times 200$ grid, 19 iterations; (2) Sources on a square: contour difference = 0.05, $200 \times 200$ grid, 3 iterations (3) 100 random source points: contour difference = 0.02, $400 \times 400$ grid, 16 iterations.

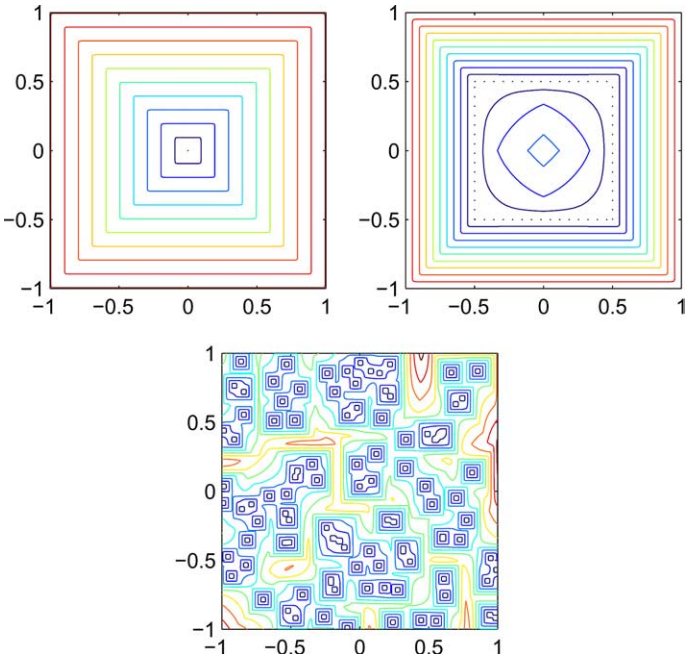



Fig. 6. $\sqrt{p^2 + q^2} + \frac{6|pq|}{\sqrt{p^2+q^2}} = 1$, $\sigma_x = \sigma_y = 4$. (1) A single source point: contour difference = 0.1, $200 \times 200$ grid, 134 iterations; (2) Sources on a square: contour difference = 0.05, $200 \times 200$ grid, 120 iterations; (3) 100 random source points: contour difference = 0.02, $400 \times 400$ grid, 119 iterations.

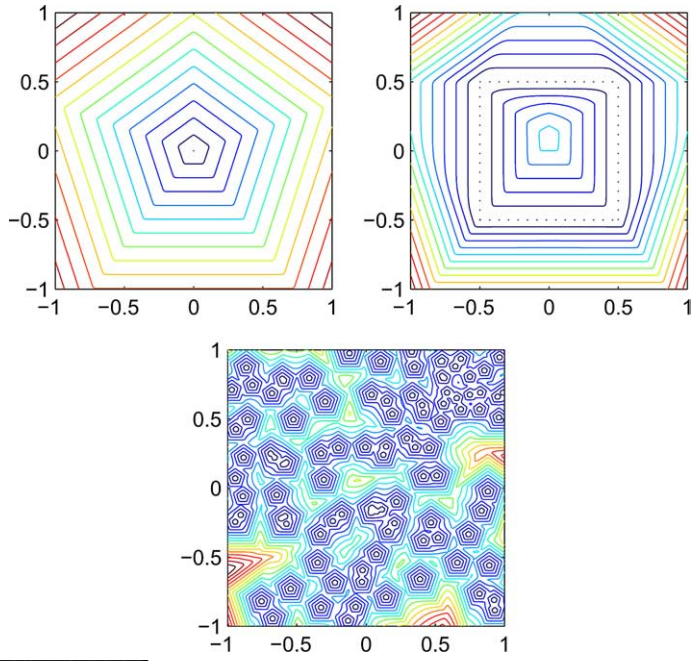



Fig. 7. $\sqrt{p^2+q^2}+\sqrt{\frac{(p^2+q^2)^{5/2}-(-5qp^4+10q^3p^2-q^5)}{2(p^2+q^2)^{3/2}}}=1$, $\sigma_x=\sigma_y=2$. (1) A source point: contour difference = 0.1, $200\times200$ grid, 93 iterations; (2) Sources on a square: contour difference = 0.05, $200\times200$ grid, 58 iterations; (3) 100 random source points: contour difference = 0.02, $400\times400$ grid, 61 iterations.

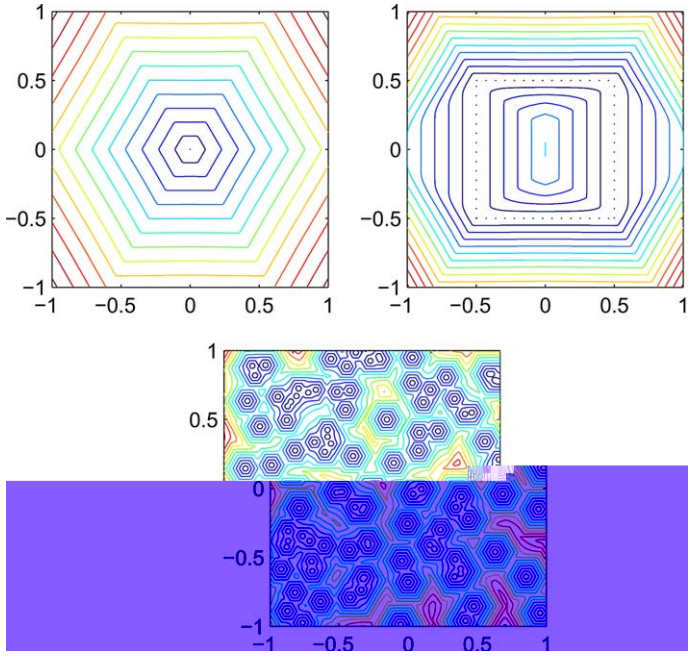



Fig. 8. $\sqrt{p^2+q^2}+|\frac{p^3-3pq^2}{p^2+q^2}|=1$, $\sigma_x=\sigma_y=2$. (1) A single source point: contour difference = 0.1, $200\times200$ grid, 88 iterations; (2) Sources on a square: contour difference = 0.05, $200\times200$ grid, 45 iterations; (3) 100 random source points: contour difference = 0.02, $400\times400$ grid, 58 iterations.

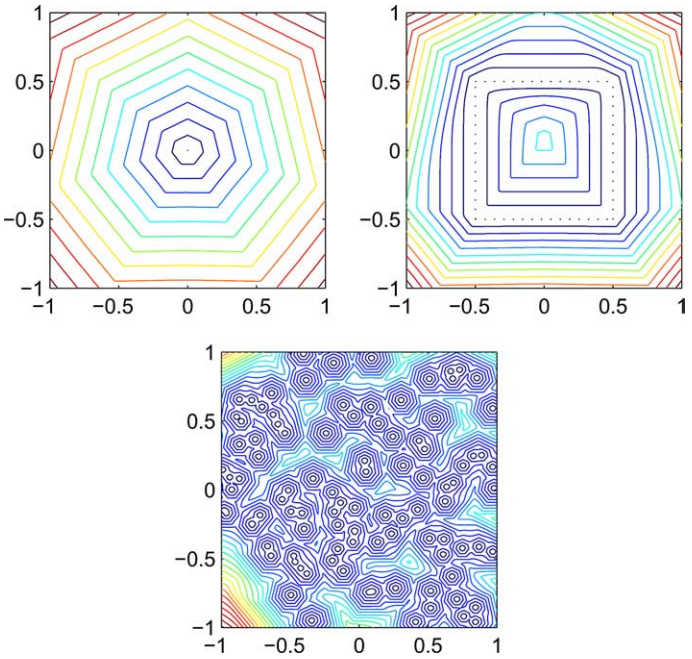



Fig. 9. $\sqrt{p^2+q^2}+\sqrt{\frac{(p^2+q^2)^{7/2}-(-q^7+21q^5p^2-35q^3p^4+7qp^6)}{2(p^2+q^2)^{5/2}}}=1$, $\sigma_x=\sigma_y=2$. (1) A single source point: contour difference = 0.1, 200 × 200 grid, 167 iterations; (2) Sources on a square: contour difference = 0.05, 200 × 200 grid, 53 iterations; (3) 100 random source points: contour difference = 0.02, 400 × 400 grid, 121 iterations.

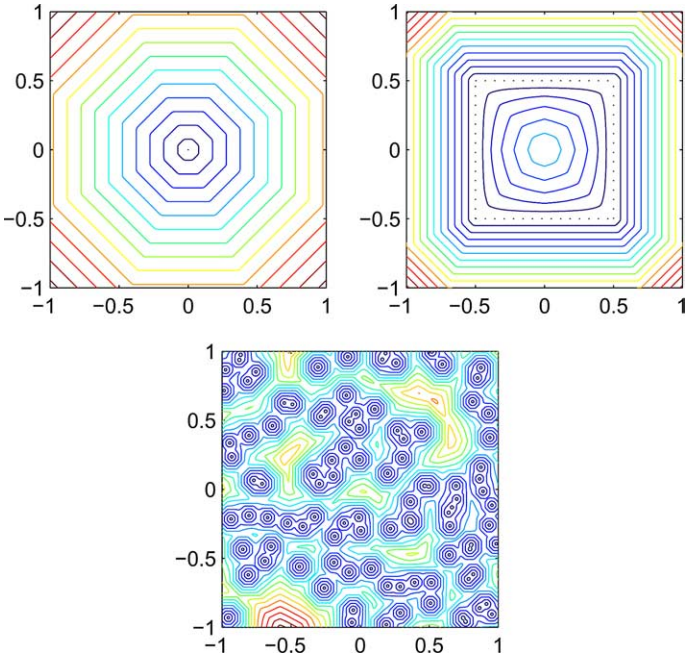



Fig. 10. $\sqrt{p^2+q^2}+\frac{|4pq(p^2-q^2)|}{(p^2-q^2)^{3/2}}=1$, $\sigma_x=\sigma_y=4$. (1) A single source point: contour difference = 0.1, 200 × 200 grid, 231 iterations; (2) Sources on a square: contour difference = 0.05, 200 × 200 grid, 163 iterations; (3) 100 random source points: contour difference = 0.02, 400 × 400 grid, 195 iterations.

Figs. 11–14 demonstrate the results for

$$\gamma(v,\varphi) = h(\varphi)\tilde{\gamma}(v),$$

where

$$h(\varphi) = (1 + 2|\sin(\varphi)|)$$

and $\gamma$ is taken to be

$$\tilde{\gamma} = \left(1 + \left|\sin\left(\frac{3}{2}\left(v + \frac{\pi}{2}\right)\right)\right|\right),$$

$$\tilde{\gamma} = \left(1 + \left|\sin\left(2\left(v + \frac{\pi}{2}\right)\right)\right|\right),$$

$$\tilde{\gamma} = \left(1 + \left|\sin\left(\frac{5}{2}\left(v + \frac{\pi}{2}\right)\right)\right|\right),$$

and

$$\tilde{\gamma} = \left(1 + \left|\sin\left(3\left(v + \frac{\pi}{2}\right)\right)\right|\right),$$

respectively. The corresponding Hamilton–Jacobi equations are as follows:

$$(\sqrt{p^2+q^2+r^2} + 2|r|)\left(1 + 3\sqrt{\frac{(p^2+q^2)^{3/2} - (3p^2q - q^3)}{2(p^2+q^2)^{3/2}}}\right) = 1,$$

$$(\sqrt{p^2+q^2+r^2} + 2|r|)\left(1 + \frac{|2pq|}{p^2+q^2}\right) = 1,$$

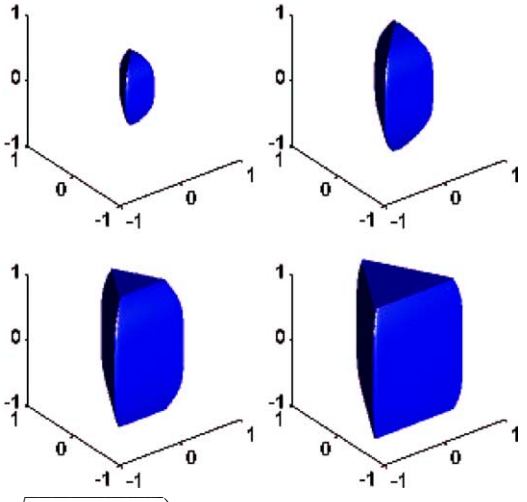


Fig. 11. $(\sqrt{p^2+q^2+r^2} + 2|r|)\left(1 + 3\sqrt{\frac{(p^2+q^2)^{3/2}-(3p^2q-q^3)}{2(p^2+q^2)^{3/2}}}\right) = 1$, $\sigma_x = \sigma_y = \frac{9}{2}$, $\sigma_z = \frac{27}{4}$, $100 \times 100 \times 100$ grid, 254 iterations; contour value = 0.2, 0.3, 0.4 and 0.5, respectively.

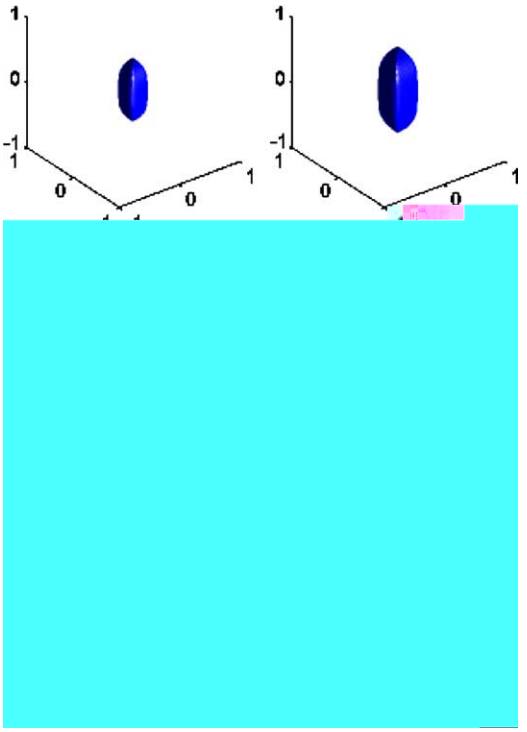


Fig. 12. $(\sqrt{p^2+q^2+r^2}+2|r|)\left(1+\frac{|2pq|}{p^2+q^2}\right)=1$, $\sigma_x=\sigma_y=\frac{5}{2}$, $\sigma_z=\frac{15}{2}$, $100\times100\times100$ grid, 97 iterations; contour value = 0.2, 0.25, 0.3, 0.35 and 0.4, respectively.

$$(\sqrt{p^2+q^2+r^2}+2|r|)\left(1+\sqrt{\frac{(p^2+q^2)^{5/2}-(-5qp^4+10q^3p^2-q^4)}{2(p^2+q^2)^{5/2}}}\right)=1,$$

and

$$(\sqrt{p^2+q^2+r^2}+2|r|)\left(1+\frac{|p^3-3pq^2|}{(p^2+q^2)^{3/2}}\right)=1.$$

Fig. 15 illustrates the Wulff shape for

$$\gamma(v,\varphi)=\left(1+2\left|\sin\left(\frac{3}{2}\left(\varphi+\frac{\pi}{2}\right)\right)\right|\right)\left(1+\left|\sin\left(\frac{5}{2}\left(v+\frac{\pi}{2}\right)\right)\right|\right),$$

and the resulting Hamilton–Jacobi equation is

$$(\sqrt{p^2+q^2+r^2})\left(1+\sqrt{2\left(1-\frac{3p^2r+3q^2r-r^3}{(p^2+q^2+r^2)^{3/2}}\right)}\right)\left(1+\sqrt{\frac{(p^2+q^2)^{5/2}-(-5qp^4+10q^3p^2-q^4)}{2(p^2+q^2)^{5/2}}}\right)=1,$$

the corresponding Wulff crystal shapes are pyramids.

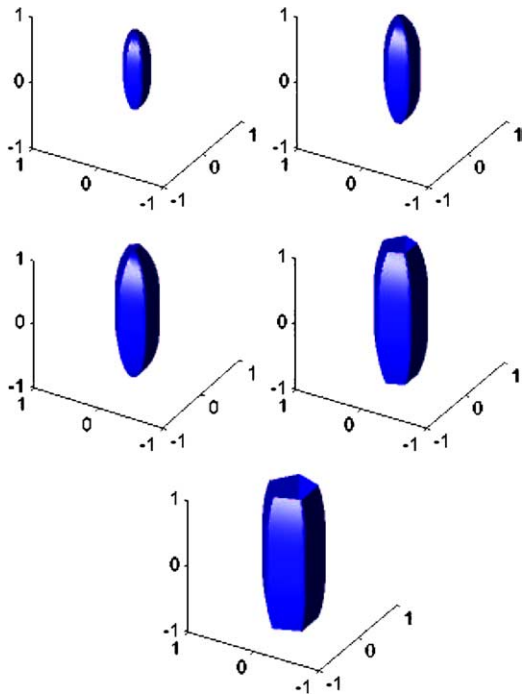


Fig. 13. $(\sqrt{p^2+q^2+r^2}+2|r|)\left(1+\sqrt{\frac{(p^2+q^2)^{5/2}-(-5qp^4+10q^3p^2-q^4)}{2(p^2+q^2)^{5/2}}}\right)=1$, $\sigma_x=\sigma_y=2$, $\sigma_z=3$, $100\times100\times100$ grid, 126 iterations; contour value = 0.2, 0.25, 0.3, 0.35 and 0.4, respectively.

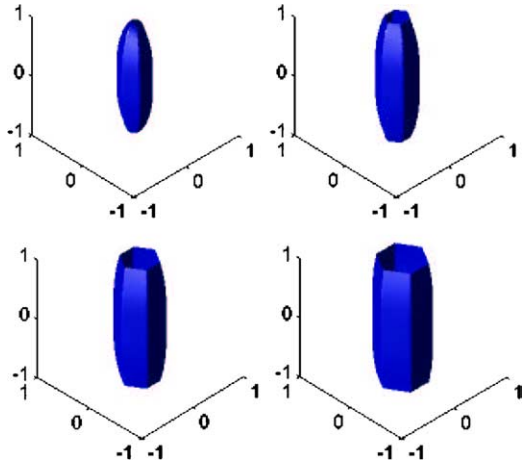


Fig. 14. $(\sqrt{p^2+q^2+r^2}+2|r|)\left(1+\frac{|p^3-3pq^2|}{(p^2+q^2)^{3/2}}\right)=1$, $\sigma_x=\sigma_y=2$, $\sigma_z=6$, $100\times100\times100$ grid, 136 iterations; contour value = 0.25, 0.3, 0.35 and 0.4, respectively.

Fig. 16 shows the Wulff shapes for

$$\gamma(v,\varphi)=\left(1+2\sqrt{\left|\sin\left(|\varphi|-\frac{\pi}{2}\right)\right|}\right)\left(1+\left|\sin\left(\frac{3}{2}\left(v+\frac{\pi}{2}\right)\right)\right|\right)$$

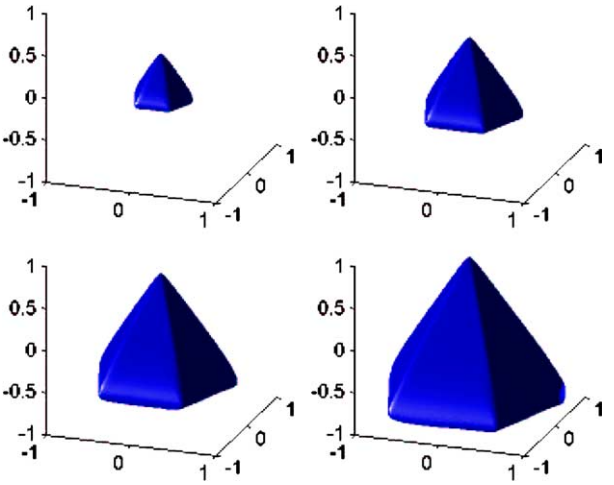


Fig. 15. $\gamma(v,\varphi) = \left(1+2\left|\sin\left(\frac{3}{2}\left(\varphi+\frac{\pi}{2}\right)\right)\right|\right)\left(1+\left|\sin\left(\frac{5}{2}\left(v+\frac{\pi}{2}\right)\right)\right|\right)$, $\sigma_x = \sigma_y = 3.5$, $\sigma_z = 4$, $100\times100\times100$ grid, 179 iterations; contour value $= 0.2$, 0.3, 0.4, and 0.5, respectively.

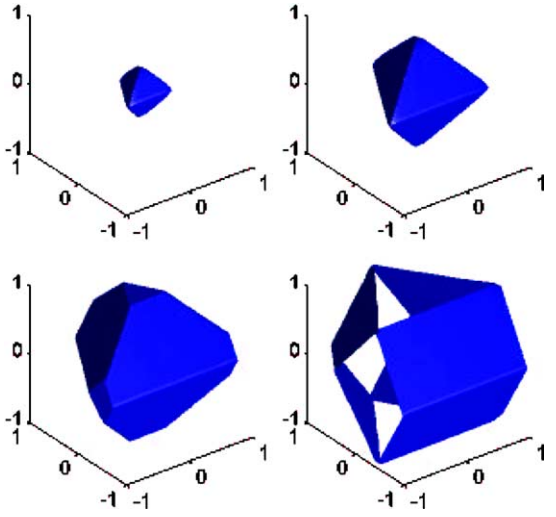


Fig. 16. $\gamma(v,\varphi) = \left(1+2\sqrt{\left|\sin\left(|\varphi|-\frac{\pi}{2}\right)\right|}\right)\left(1+\left|\sin\left(\frac{3}{2}\left(v+\frac{\pi}{2}\right)\right)\right|\right)$, $\sigma_x = \sigma_y = \sigma_z = 6$, $100\times100\times100$ grid, 91 iterations; contour value $= 0.2$, 0.4, 0.6 and 0.8, respectively.

and the resulting Hamilton–Jacobi equation is

$$\left(\sqrt{p^2+q^2+r^2}+\sqrt{2\sqrt{p^2+q^2+r^2}|\sqrt{3}|r|-\sqrt{p^2+q^2}|}\right)\left(1+3\sqrt{\frac{(p^2+q^2)^{3/2}-(3p^2q-q^3)}{2(p^2+q^2)^{3/2}}}\right)=1,$$

in this case the Wulff crystal shapes are bi-pyramids.

Fig. 17 depicts the Wulff shapes for

$$\gamma(v,\varphi) = \left(1+2\sqrt{\left|\sin\left(|\varphi|-\frac{\pi}{2}\right)\right|}\right)\left(1+\left|\sin\left(\frac{5}{2}\left(v+\frac{\pi}{2}\right)\right)\right|\right),$$

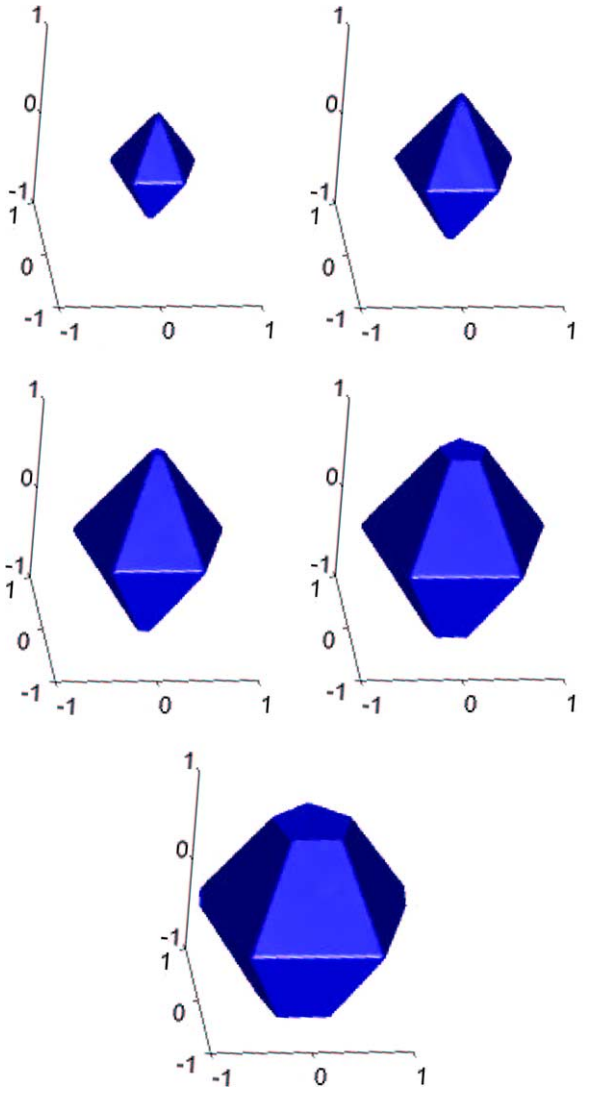


Fig. 17. $\gamma(v,\varphi) = \left(1 + 2\sqrt{\left|\sin\left(|\varphi| - \frac{\pi}{2}\right)\right|}\right)\left(1 + \left|\sin\left(\frac{5}{2}\left(v + \frac{\pi}{2}\right)\right)\right|\right)$, $\sigma_x = \sigma_y = 3.5$, $\sigma_z = 4$, $100 \times 100 \times 100$ grid, 54 iterations; contour value = 0.3, 0.4, 0.5, 0.6 and 0.7, respectively.

and the resulting Hamilton–Jacobi equation is

$$\left(\sqrt{p^2+q^2+r^2} + \sqrt{2\sqrt{p^2+q^2+r^2}|\sqrt{3}|r| - \sqrt{p^2+q^2}}\right)\left(1 + \sqrt{\frac{(p^2+q^2)^{5/2} - (-5qp^4 + 10q^3p^2 - q^4)}{2(p^2+q^2)^{5/2}}}\right) = 1,$$

in this case the Wulff crystal shapes are also bi-pyramids.

### 4.2. Traveltime computation for elastic wave propagation

In the high frequency asymptotics for linear elastic wave propagation, we need to compute traveltime functions for three different wave modes: the quasi-P and two quasi-S waves; see [13] and reference therein

for details. Here we consider a typical anisotropic elastic model, the transversely isotropic solid with horizontal symmetry. Then the quasi-P and the quasi-SV slowness surfaces are defined by the following quartic equation:

$$c_1p^4 + c_2p^2q^2 + c_3q^4 + c_4p^2 + c_5q^2 + 1 = 0,$$

where

$$c_1 = a_{11}a_{44}, \quad c_2 = a_{11}a_{33} + a_{44}^2 - (a_{13} + a_{44})^2, \quad c_3 = a_{33}a_{44}, \quad c_4 = -(a_{11} + a_{44}), \quad c_5 = -(a_{33} + a_{44}).$$

Here $a_{ij}$s are given elastic parameters. Substituting $p = \partial\phi/\partial x$ and $q = \partial\phi/\partial y$ into the above equation, we have a nonlinear Hamilton–Jacobi equation for the function $\phi$, the traveltime. Similarly, the quasi-SH slowness surface is defined by the equation

$$\tfrac{1}{2}(a_{11} - a_{12})p^2 + a_{44}q^2 = 1.$$

Since the model is transversely isotropic with horizontal symmetry, we may replace $p^2$ by $p^2 + r^2$ to obtain 3-D Hamilton–Jacobi equations, where $r = \partial\phi/\partial z$. Furthermore, we remark that the Hamilton–Jacobi equation for quasi-SV wave traveltime in this model has a nonconvex Hamiltonian; see [13] for results related to multivalued traveltime computation.

Fig. 18 presents wavefront contours for the three different wave modes in a homogeneous, transversely isotropic solid. Since the Hamilton–Jacobi equation for quasi-SV traveltime is nonconvex, the quasi-SV wavefront has cusps, corresponding to a class of multivalued solutions; see [13] for capturing those multivalued wavefronts. However, the concept of the viscosity solution underlying the Lax–Friedrichs sweeping scheme allows a single-valued solution only which essentially picks out the first-arrival traveltime and

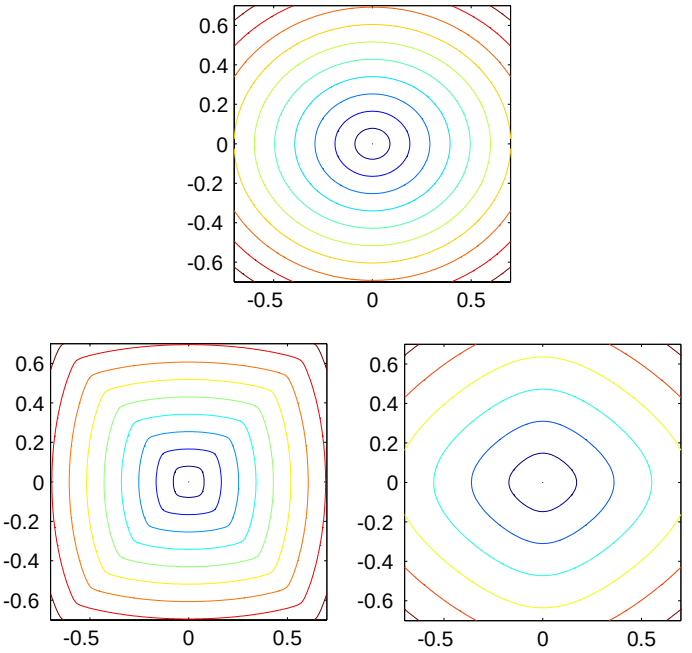
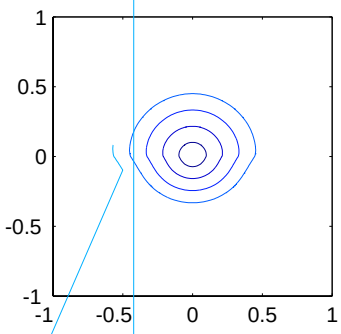



Fig. 18. $a_{11} = 15.0638$, $a_{33} = 10.8373$, $a_{13} = 1.6381$, $a_{44} = 3.1258$, and $a_{12} = 6.5616$; contour difference = 0.05, $200 \times 200$ grid. (1) quasi-SH: $\sigma_x = \sigma_y = 1.5$, 31 iterations; (2) quasi-SV: $\sigma_x = \sigma_y = 2$, 44 iterations; (3) quasi-P: $\sigma_x = \sigma_y = 3$, 50 iterations.




removes those cusps. This can be observed in Fig. 18(2), where we can see that kinks appear along the two diagonals.

Fig. 19 shows computational results for a model with two layers, so that the corresponding Hamilton–Jacobi equations have discontinuous coefficients; therefore, this model is used to test the stability and robustness of the sweeping scheme. As we can see from the figure, the Snell's law for anisotropic media is well enforced.

Fig. 20 demonstrates results for a 3-D transversely isotropic model with horizontal symmetry. As we can see from Fig. 20(2) and (3), the wavefront profiles along $y$-direction are circles as expected from the horizontal symmetry of the model.

### *4.3. Convergence test*

To validate the new Lax–Friedrichs sweeping scheme, we first apply the method to the eikonal equation $|\nabla\phi| = 1$ with a single source point at the center of the computational domain. In this case, we know the exact solution so that the convergence behavior of the method can be easily observed. Table 1 presents the $L_\infty$ errors between computed and exact solutions for different mesh sizes. As we can see, the errors indicate the apparent first-order convergence as the mesh size approaches zero.

Next we study the Lax–Friedrichs sweeping scheme for 2-D Wulff crystal shape examples with source points located at the center of the computational domain. In these cases, we do not know the exact solutions; therefore, we consider the computed solution with the mesh size $\frac{2}{1600}$ as a good approximation of the true solution and observe the $L_\infty$ error behaviors on coarser meshes. In Table 2, Figs. 2(1), 3(1), 5(1) and

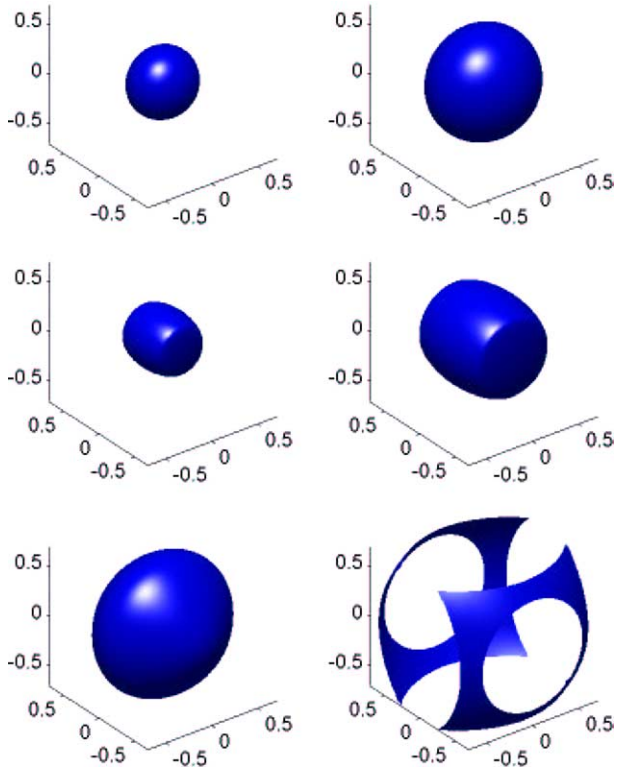


Fig. 20. $a_{11} = 15.0638$, $a_{33} = 10.8373$, $a_{13} = 1.6381$, $a_{44} = 3.1258$, and $a_{12} = 6.5616$; contour value $= 0.2$ and 0.3, $100 \times 100 \times 100$ grid; (1) quasi-SH: $\sigma_x = \sigma_y = \sigma_z = l.50$, 26 iterations; (2) quasi-SV: $\sigma_x = \sigma_y = \sigma_z = 1.50$, 24 iterations; (3) quasi-P: $\sigma_x = \sigma_y = \sigma_z = 1.50$, 26 iterations.

Table 1
The 2D eikonal case: errors and convergence order

| d$x$ | $L_\infty$-error | Convergence order |
|---|---|---|
| $\frac{2}{50}$ | 0.062286 | |
| $\frac{2}{100}$ | 0.036946 | 0.743490 |
| $\frac{2}{200}$ | 0.019846 | 0.896570 |
| $\frac{2}{400}$ | 0.010416 | 0.930047 |
| $\frac{2}{800}$ | 0.005378 | 0.953660 |
| $\frac{2}{1600}$ | 0.002750 | 0.967638 |

7(1) correspond to ellipse, triangle, quadrilateral, and pentagon Wulff crystal shapes, respectively. The errors listed in Table 2 also indicate first-order convergence.

Table 3 presents the numbers of iterations for 2-D Wulff crystal shape examples, and the result indicates that in some cases, for example, Fig. 5(1), we have $O(N)$ complexity since the corresponding Hamiltonian is separable.

However, in general, we have an algorithmic complexity better than $O(N^2)$ but worse than $O(N)$.

Table 2
The errors of 2-D Wulff crystal shape examples

| Example | Fig. 2(1) | Fig. 3(1) | Fig. 5(1) | Fig. 7(1) |
|---|---|---|---|---|
| $\left\|\phi_{\frac{2}{50}} - \phi_{\frac{2}{1600}}\right\|_\infty$ | 0.074627 | 0.142362 | 0.119700 | 0.037861 |
| $\left\|\phi_{\frac{2}{100}} - \phi_{\frac{2}{1600}}\right\|_\infty$ | 0.042410 | 0.089958 | 0.073120 | 0.020970 |
| $\left\|\phi_{\frac{2}{200}} - \phi_{\frac{2}{1600}}\right\|_\infty$ | 0.022857 | 0.053107 | 0.042770 | 0.013404 |
| $\left\|\phi_{\frac{2}{400}} - \phi_{\frac{2}{1600}}\right\|_\infty$ | 0.011111 | 0.028248 | 0.022670 | 0.007039 |
| $\left\|\phi_{\frac{2}{800}} - \phi_{\frac{2}{1600}}\right\|_\infty$ | 0.004121 | 0.011427 | 0.009180 | 0.002419 |

Table 3
The number of iterations for 2-D Wulff crystal shape examples

| d$x$ | Fig. 2(1) | Fig. 3(1) | Fig. 5(1) | Fig. 7(1) |
|---|---|---|---|---|
| $\frac{2}{50}$ | 28 | 65 | 16 | 75 |
| $\frac{2}{100}$ | 39 | 79 | 18 | 80 |
| $\frac{2}{200}$ | 55 | 100 | 19 | 93 |
| $\frac{2}{400}$ | 86 | 140 | 20 | 117 |
| $\frac{2}{800}$ | 146 | 229 | 20 | 202 |
| $\frac{2}{1600}$ | 262 | 404 | 20 | 354 |

Table 4 illustrates the computational results for some 3-D examples, and the errors also indicate first-order convergence.

Table 5 shows the number of iterations for these 3-D examples; once again, the observed algorithmic complexity is much better than $O(N^2)$. If the Hamiltonian is smooth, for example, in the anisotropic wave propagation problem, then Table 5 indicates that the complexity is $O(N \log N)$ on average.

Table 4
The errors of 3-D examples

| Example | Fig. 17 | Fig. 20 quasi-SH | Fig. 20 quasi-P | Fig. 20 quasi-SV |
|---|---|---|---|---|
| $\left\|\phi_{\frac{2}{50}} - \phi_{\frac{2}{300}}\right\|_\infty$ | 0.139730 | 0.060470 | 0.061429 | 0.033068 |
| $\left\|\phi_{\frac{2}{100}} - \phi_{\frac{2}{300}}\right\|_\infty$ | 0.091490 | 0.026058 | 0.027040 | 0.014548 |
| $\left\|\phi_{\frac{2}{150}} - \phi_{\frac{2}{300}}\right\|_\infty$ | 0.067460 | 0.013513 | 0.014183 | 0.007627 |
| $\left\|\phi_{\frac{2}{200}} - \phi_{\frac{2}{300}}\right\|_\infty$ | 0.055150 | 0.006913 | 0.007311 | 0.003930 |
| $\left\|\phi_{\frac{2}{250}} - \phi_{\frac{2}{300}}\right\|_\infty$ | 0.045040 | 0.002810 | 0.002988 | 0.001606 |

Table 5
The number of iterations for 3-D examples

| d$x$ | Fig. 17 | Fig. 20 quasi-SH | Fig. 20 quasi-P | Fig. 20 quasi-SV |
|---|---|---|---|---|
| $\frac{2}{50}$ | 42 | 21 | 21 | 19 |
| $\frac{2}{100}$ | 52 | 26 | 26 | 24 |
| $\frac{2}{150}$ | 90 | 31 | 31 | 29 |
| $\frac{2}{200}$ | 109 | 36 | 36 | 34 |
| $\frac{2}{250}$ | 131 | 41 | 41 | 39 |
| $\frac{2}{300}$ | 206 | 46 | 47 | 44 |

## 5. Conclusion

In this paper, we proposed a simple, fast sweeping method based on the Lax–Friedrichs Hamiltonian to approximate viscosity solutions of static Hamilton–Jacobi equations. By using the Lax–Friedrichs Hamiltonian, we can handle much more complicated Hamiltonians, including both convex and general non-convex cases. By using the sweeping method, we are able to follow a group of characteristics at each iteration to speed up the algorithm. Unlike other sweeping methods based on the Godunov Hamiltonian, we need to specify a computational boundary condition for our scheme. In order to have no inflow at the boundary, we have derived a min–max formula to impose appropriate computational boundary conditions. Some properties of the approximation for the 1-D eikonal equation were analyzed. We illustrated the efficiency and accuracy of the approach with extensive numerical examples in 2- and 3-D cases. Currently, we are applying the method to differential games and we will report the result elsewhere.

## References

[1] M. Bardi, S. Osher, The nonconvex multi-dimentional Riemann problem for Hamilton–Jacobi equations, SIAM J. Math. Anal. 22 (2) (1991) 344–351.
[2] M. Boué, P. Dupuis, Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control, SIAM J. Numer. Anal. 36 (3) (1999) 667–695.
[3] M.G. Crandall, P.L. Lions, Two approximations of solutions of Hamilton–Jacobi equations, Math. Comput. 43 (1984) 1–19.
[4] S. Gray, W. May, Kirchhoff migration using eikonal equation travel-times, Geophysics 59 (5) (1994) 810–817.
[5] J. Helmsen, E. Puckett, P. Colella, M. Dorr, Two new methods for simulating photolithography development in 3d. in: SPIE 2726, 1996, pp. 253–261.
[6] G.-S. Jiang, D. Peng, Weighted ENO schemes for Hamilton–Jacobi equations, SIAM J. Sci. Comput. 21 (6) (2000) 2126–2143 (electronic).
[7] C.Y. Kao, S. Osher, R. Tsai, Fast sweeping methods for Hamilton–Jacobi equations, UCLA CAM Reports (02-66), SINUM (submitted).
[8] S. Osher, A level set formulation for the solution of the Dirichlet problem for Hamilton–Jacobi equations, SIAM J. Math. Anal. 24 (5) (1993) 1145–1152.
[9] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49.
[10] S. Osher, C.-W. Shu, High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations, SIAM J. Numer. Anal. 28 (4) (1991) 907–922.
[11] D. Peng, B. Merriman, S. Osher, H.-K. Zhao, M. Rang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438.
[12] D. Peng, S. Osher, B. Merriman, H.-K. Zhao, The geometry of Wulff crystal shapes and its relations with Riemann problems, in: Nonlinear Partial Differential Equations (Evanston, IL, 1998), American Mathematical Society, Providence, RI, 1999, pp. 251–303.
[13] J. Qian, L.T. Cheng, S. Osher, A level set based Eulerian approach for anisotropic wave propagations, Wave Motion 37 (2003) 365–379.
[14] J. Qian, W.W. Symes, Paraxial eikonal solvers for anisotropic quasi-P travel times, J. Comput. Phys. 174 (1) (2001) 256–278.
[15] J. Qian, W.W. Symes, Finite-difference quasi-P traveltimes for anisotropic media, Geophysics 67 (2002) 147–155.
[16] E. Rouy, A. Tourin, A viscosity solutions approach to shape-from-shading, SIAM J. Numer. Anal. 29 (3) (1992) 867–884.
[17] J.A. Sethian, Fast marching level set methods for three dimensional photolithography development, in: SPIE 2726, 1996, pp. 261–272.
[18] J.A. Sethian, A. Vladimirsky, Ordered upwind methods for static Hamilton–Jacobi equations, Proc. Natl. Acad. Sci. USA 98 (20) (2001) 11069–11074 (electronic).
[19] P.E. Souganidis, Approximation schemes for viscosity solutions of Hamilton–Jacobi equations, J. Differential Equations 59 (1) (1985) 1–43.
[20] Y.-H.R. Tsai, L.-T. Cheng, S. Osher, H.-K. Zhao, Fast sweeping algorithms for a class of Hamilton–Jacobi equations, SIAM J. Numer. Anal. 41 (2) (2003) 659–672.
[21] J. Tsitsiklis, Efficient algorithms for globally optimal trajectories, IEEE Trans. Autom. Control 40 (9) (1995) 1528–1538.
[22] H.-K. Zhao, Fast sweeping method for eikonal equations I: Distance function. Available from: <www.math.uci.edu/~zhao>, 2002. Under review, SINUM.