A STATIC PDE APPROACH FOR MULTI-DIMENSIONAL EXTRAPOLATION USING FAST SWEEPING METHODS

TARIQ ASLAM [*], SONGTING LUO [†], AND HONGKAI ZHAO [‡]

Abstract. A static Partial Differential Equation (PDE) approach is presented for multi-dimensional extrapolation under the assumption that a level set function exists which separates the region of known values from the region to be extrapolated. Arbitrary orders of polynomial extrapolation can be obtained through solutions of a series of static linear PDEs. Fast sweeping methods of first and second orders are presented to solve the PDEs for constant, linear and quadratic extrapolation. Numerical examples are presented to demonstrate the approach.

Key words. Static PDE, Multi-dimensional extrapolation, Fast sweeping method

AMS subject classifications. 65N06, 49M25

1. Introduction.

A great number of computational physics applications require extrapolation methods. One particularly popular method utilized in level set methods [2] can be used to extrapolate data from one region, where the function is known, to another region where there is no data. A level set function demarcates the regions where the function is known and where extrapolation is needed. The original application of this type of extrapolation was the Ghost Fluid Method [7], where data is known in the "real" region and needs to be extrapolated into the "ghost" region. This was originally done with constant extrapolation in the normal direction to the interface. Linear and quadratic extrapolation were introduced in [2] and extended to cubic extrapolation in [9]. Fast marching and fast sweeping techniques were discussed in [1, 4] for constant extrapolation.

In terms of physics applications, the PDE approach to extrapolation was utilized in [9] for solving Laplace and heat conduction equations on arbitrary domains with applications to Stefan problems. The technique has also been applied to vaporization in two-phase fluid flow [23], bubble dynamics [5] and compressible reacting flow with phase changes [12]. It has also been extended to work on quadtree and octree grids [16] with applications to solving nonlinear Poisson-Boltzmann equations [17]. An excellent summary of parabolic and elliptic problems utilizing PDE extrapolation was given in [10].

The goal of this work is to develop a static PDE approach as general as the time-dependent approach proposed in [2, 9]. The main advantage of using the static PDE approach is computation efficiency. For the time-dependent approach, where the time is artificial, extrapolation or information is advected from the known region to the unknown region. Numerically one solves a series of linear advection equations. The time step is restricted by the Courant-Friedrichs-Lewy (CFL) condition for stable explicit time schemes, which can also be viewed as Jacobi iterations. In other words, it takes $O(d/\Delta t)$ time steps or iterations to advect extrapolation into the unknown region within a distance d from the known region, where the time step Δt has to be of the same order of the grid size Δx. Using the static PDE approach, efficient iterative schemes can be applied to solve the static linear PDEs. For example, the

[*]Los Alamos National Laboratory, Los Alamos, NM 87545. (aslam@lanl.gov).

[†]Department of Mathematics, Iowa State University, Ames, IA 50011. (luos@iastate.edu).

[‡]Department of Mathematics, University of California, Irvine, CA 92697-3875. (zhao@math.uci.edu). Research of Zhao is partially supported by NSF grant DMS-1115698.

fast sweeping method can be easily tailored for this task. The number of iterations is finite, and is usually small and almost independent of the grid size. In principle, one pass type of algorithms, such as fast marching method [21, 24] can also be used for solving these linear static advection PDEs. Sorting is not even needed in this case since the upwind direction is given a priori. However, depending on the geometry of the boundary of the known region, using a fixed upwind stencils may not work for the fast marching method. We will show numerical examples to demonstrate this situation in Section 4. Techniques from ordered upwind scheme [22] may be needed to adjust the local stencil at each grid. On the contrary, the fast sweeping method always uses fixed stencils and discretization schemes at each grid and can deal with high order discretization in the same easy and efficient fashion.

The outline of the paper is the following. In Section 2, the static PDE approach to solving constant, linear and quadratic extrapolation is presented. In Section 3, the fast sweeping methods of first and second orders are given. Section 4 provides numerical tests for several extrapolation methods and demonstrates efficiency and accuracy of the various schemes. Finally, we draw conclusions in the last section. In the appendix, extension of the static PDE approach to triangular meshes is presented.

2. Static PDE Approach to Multi-Dimensional Extrapolation . Assume a function u is given in a domain Ω_{in} that is defined by $\psi \leq 0$, and needs to be extrapolated to the remaining portion of the space $\Omega\backslash\Omega_{in}$ that is defined by $\psi > 0$, where ψ is a given level set function, e.g., the signed distance function from the interface $\Gamma = \overline{\Omega_{in}} \cap \overline{\Omega\backslash\Omega_{in}}$. Motivated by the time-dependent PDE approach introduced in [2], we consider a static PDE approach.

2.1. Constant Extrapolation. Constant extrapolation of the function u is done as a constant along a normal $\vec{\mathbf{n}}$ to the interface Γ. $\vec{\mathbf{n}}$ is defined with the level set function,

$$\vec{\mathbf{n}} = \frac{\nabla\psi}{|\nabla\psi|}. \tag{2.1}$$

The PDE used to achieve constant extrapolation is

$$H(\psi)\vec{\mathbf{n}} \cdot \nabla u = 0, \tag{2.2}$$

where $H(\psi)$ is the unit Heaviside function, i.e.,

$$H(\psi) = \begin{cases} 1 & \text{if } \psi > 0, \\ 0 & \text{if } \psi \leq 0. \end{cases} \tag{2.3}$$

2.2. Linear Extrapolation. Linear extrapolation in the normal direction is done in three steps. First, the directional derivative of u in the normal direction is defined as

$$u_{\mathbf{n}} = \vec{\mathbf{n}} \cdot \nabla u. \tag{2.4}$$

Then $u_{\mathbf{n}}$ can be extrapolated in a constant manner into the unknown region by the PDE

$$H(\psi)\vec{\mathbf{n}} \cdot \nabla u_{\mathbf{n}} = 0. \tag{2.5}$$

With $u_{\mathbf{n}}$ in the whole space, we can extrapolate u into the unknown region by solving the PDE

$$H(\psi)(\vec{\mathbf{n}} \cdot \nabla u - u_{\mathbf{n}}) = 0. \tag{2.6}$$

2.3. Quadratic Extrapolation. Quadratic extrapolation in the normal direction is done in four steps. First, the second directional derivative of u in the normal direction is defined as

$$u_{\mathbf{nn}} = \vec{\mathbf{n}} \cdot \nabla(\vec{\mathbf{n}} \cdot \nabla u) \tag{2.7}$$

Then $u_{\mathbf{nn}}$ is extrapolated into the unknown region in a constant manner with the PDE

$$H(\psi)\vec{\mathbf{n}} \cdot \nabla u_{\mathbf{nn}} = 0. \tag{2.8}$$

Next, with $u_{\mathbf{nn}}$ in the whole space, $u_{\mathbf{n}}$ can be extrapolated into the whole space in a linear manner with the PDE

$$H(\psi)(\vec{\mathbf{n}} \cdot \nabla u_{\mathbf{n}} - u_{\mathbf{nn}}) = 0. \tag{2.9}$$

Finally, the function u can be extrapolated to the unknown region via the PDE

$$H(\psi)(\vec{\mathbf{n}} \cdot \nabla u - u_{\mathbf{n}}) = 0. \tag{2.10}$$

2.4. Higher-order Extrapolation. It is clear that the pattern of the static PDE formulations can be extended to higher-order polynomial extrapolation. For instance in the n-th order extrapolation, the n-th order directional derivative of u is first computed in the region $\psi \leq 0$, then extrapolated in a constant fashion. After that, each successive lower-order directional derivative is integrated until u is calculated.

REMARK 1. *Here the Heaviside function is equivalent to a label to indicate where extrapolation is needed.*

3. Fast Sweeping Methods. In this section, we present the fast sweeping methods for solving the aforementioned PDEs in the general form,

$$\begin{aligned} &\mathbf{a}(\mathbf{x}) \cdot \nabla u(\mathbf{x}) = f(\mathbf{x}), \qquad && \mathbf{x} \in \Omega \backslash \Gamma, \\ &u(\mathbf{x}) = u_\Gamma(\mathbf{x}), && \mathbf{x} \in \Gamma, \end{aligned} \tag{3.1}$$

where $\mathbf{a}(\mathbf{x})$, $f(\mathbf{x})$, and $u_\Gamma(\mathbf{x})$ are given functions. For this linear hyperbolic PDE, boundary condition should be prescribed at the inflow boundary, i.e., the part of boundary where $\mathbf{a}(\mathbf{x}) \cdot \nu(\mathbf{x}) < 0,\ \mathbf{x} \in \Gamma$, where $\nu(\mathbf{x})$ is outward normal at the boundary. For the cases of extrapolation, $\mathbf{a}(\mathbf{x}) = \vec{\mathbf{n}}(\mathbf{x})$, $f(\mathbf{x}) = 0$, and the inflow boundary is the boundary of the given domain Ω_{in} or the zero level set of the level set function ψ. For simplicity, we present the fast sweeping methods on uniform meshes in two-dimensional (2-D) cases. Implementation on triangular meshes is discussed in Appendix A. Extension to higher dimension is straightforward. Fast sweeping methods are efficient iterative methods for solving hyperbolic type of PDEs, for example, the Hamilton-Jacobi equations. For previous work on the development of fast sweeping methods, please refer to [8, 13, 14, 19, 20, 27] and reference therein. For the linear advection equation (3.1), the fast sweeping method is even simpler since the upwind direction is give a priori by $\mathbf{a}(\mathbf{x})$. Hence the discretization scheme is linear, i.e., independent of the solution. The two key ingredients for the success of fast sweeping methods are: (1) an upwind discretization, and (2) systematic and alternating orderings that can cover all directions of characteristics. We first present a few upwind discretizations and then two choices of orderings.

Let us assume the domain is given by $[x_{min},\ x_{max}] \times [y_{min},\ y_{max}]$, and is discretized with a uniform mesh of size $I \times J$. The mesh size is denoted as h. At each grid point (i, j) (or (x_i, y_j)), for any function F, we denote $F_{i,j} = F(x_i, y_j)$. Also, assume that $\mathbf{a}(\mathbf{x}) = (a^x(x, y), a^y(x, y))$.

3.1. First-order Approximation. We present a first-order version of the fast sweeping method. At point (i, j), without loss of generality, let us assume $a^x_{i,j} > 0,\ a^y_{i,j} > 0$. The fast sweeping method uses one-sided first-order finite difference to approximate the gradient ∇u. That is,

$$(u_x)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h}, \quad (u_y)_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{h}. \tag{3.2}$$

By substituting (3.2) into (3.1), we obtain the discretized PDE

$$a^x_{i,j}\frac{u_{i,j} - u_{i-1,j}}{h} + a^y_{i,j}\frac{u_{i,j} - u_{i,j-1}}{h} = f_{i,j}, \tag{3.3}$$

which can be reformulated to get a local updating formula for $u_{i,j}$, i.e.,

$$u_{i,j} = \frac{f_{i,j} + a^x_{i,j}u_{i-1,j}/h + a^y_{i,j}u_{i,j-1}/h}{a^x_{i,j}/h + a^y_{i,j}/h}. \tag{3.4}$$

3.2. Second-order Approximation. Here we present two versions of second-order schemes. One is a simple direction by direction upwind second-order scheme. The other one is an upwind second-order scheme with compact stencils.

3.2.1. Direction by direction second-order upwind scheme. An easy direction by direction version is as follows. At point $(i,\ j)$ with $a^x_{i,j} > 0,\ a^y_{i,j} > 0$, we use one-sided second-order finite difference to approximate u_x, u_y. That is,

$$\begin{aligned}(u_x)_{i,j} &= \frac{-2/3u_{i,j} + 2u_{i-1,j} - 1/2u_{i-2,j}}{h},\\ (u_y)_{i,j} &= \frac{-2/3u_{i,j} + 2u_{i,j-1} - 1/2u_{i,j-2}}{h}.\end{aligned} \tag{3.5}$$

By substituting (3.5) into (3.1), we obtain the discretized PDE

$$a^x_{i,j}\frac{\frac{-2}{3}u_{i,j} + 2u_{i-1,j} - \frac{1}{2}u_{i-2,j}}{h} + a^y_{i,j}\frac{\frac{-2}{3}u_{i,j} + 2u_{i,j-1} - \frac{1}{2}u_{i,j-2}}{h} = f_{i,j}, \tag{3.6}$$

from which we can get a local updating formula for $u_{i,j}$, i.e.,

$$u_{i,j} = \frac{f_{i,j} + a^x_{i,j}(2u_{i-1,j} - \frac{1}{2}u_{i-2,j})/h + a^y_{i,j}(2u_{i,j-1} - \frac{1}{2}u_{i,j-2})/h}{-2a^x_{i,j}/(3h) - 2a^y_{i,j}/(3h)}. \tag{3.7}$$

Following the same procedure, the one-sided finite difference approximation, such as (3.2) or (3.5), can be easily obtained for other cases of $\mathbf{a}(\mathbf{x})$, and hence the local updating formula, such as (3.4) or (3.7).

3.2.2. Second-order upwind scheme with compact stencils. Here we give a second-order scheme with compact stencils following the formulation developed in [3]. The key idea is to use the original PDE (3.1) to provide more relations and hence reduce the number of needed stencils to achieve certain accuracy. Differentiating (3.1) we get

$$\nabla\mathbf{a}(\mathbf{x}) \cdot \nabla u(\mathbf{x}) + D^2u(\mathbf{x}) \cdot \mathbf{a}(\mathbf{x}) = \nabla f(\mathbf{x}), \tag{3.8}$$

where $\nabla\mathbf{a}$ denotes the Jacobian of $\mathbf{a}$ and D^2u denotes the Hessian of u. The Taylor expansion of $u(\mathbf{x})$ to the second order is

$$u(\mathbf{x} + \delta\mathbf{x}) = u(\mathbf{x}) + \delta\mathbf{x} \cdot \nabla u(\mathbf{x}) + \frac{1}{2}\langle\delta\mathbf{x}, D^2u(\mathbf{x}) \cdot \delta\mathbf{x}\rangle + O(\|\delta\mathbf{x}\|^3). \tag{3.9}$$

By writing $\delta\mathbf{x}$ as

$$\delta\mathbf{x} = (\delta\mathbf{x}\cdot\hat{\mathbf{a}})\hat{\mathbf{a}} + (\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})\hat{\mathbf{a}}^{\perp}, \tag{3.10}$$

(3.9) can be written as

$$\begin{aligned} u(\mathbf{x}+\delta\mathbf{x}) =& u(\mathbf{x}) + \delta\mathbf{x}\cdot\nabla u(\mathbf{x}) + \frac{1}{2}(\delta\mathbf{x}\cdot\hat{\mathbf{a}})^2\langle\hat{\mathbf{a}}, D^2u(\mathbf{x})\cdot\hat{\mathbf{a}}\rangle \\ &+ (\delta\mathbf{x}\cdot\hat{\mathbf{a}})(\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})\langle\hat{\mathbf{a}}^{\perp}, D^2u(\mathbf{x})\cdot\hat{\mathbf{a}}\rangle \\ &+ \frac{1}{2}(\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})^2\langle\hat{\mathbf{a}}^{\perp}, D^2u(\mathbf{x})\cdot\hat{\mathbf{a}}^{\perp}\rangle + O(\|\delta\mathbf{x}\|^3), \end{aligned} \tag{3.11}$$

where $\hat{\mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}$ and $\|\hat{\mathbf{a}}^{\perp}\| = 1, \hat{\mathbf{a}}^{\perp} \perp \hat{\mathbf{a}}$. From (3.8) we have

$$D^2u(\mathbf{x})\cdot\hat{\mathbf{a}} = \frac{\nabla f(\mathbf{x})}{\|\mathbf{a}(\mathbf{x})\|} - \frac{\nabla\mathbf{a}(\mathbf{x})\cdot\nabla u(\mathbf{x})}{\|\mathbf{a}(\mathbf{x})\|}. \tag{3.12}$$

By substituting (3.12) into (3.11), we have

$$\begin{aligned} u(\mathbf{x}+\delta\mathbf{x}) =& u(\mathbf{x}) + \delta\mathbf{x}\cdot\nabla u(\mathbf{x}) + \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}})^2}{2\|\mathbf{a}\|}(\hat{\mathbf{a}}\cdot\nabla f - \hat{\mathbf{a}}\cdot(\nabla\mathbf{a}\cdot\nabla u)) \\ &+ \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}})(\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})}{\|\mathbf{a}\|}(\hat{\mathbf{a}}^{\perp}\cdot\nabla f - \hat{\mathbf{a}}^{\perp}\cdot(\nabla\mathbf{a}\cdot\nabla u)) \\ &+ \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})^2}{2}\langle\hat{\mathbf{a}}^{\perp}, D^2U\cdot\hat{\mathbf{a}}^{\perp}\rangle + O(\|\delta\mathbf{x}\|^3) \\ =& u(\mathbf{x}) + \left(\delta\mathbf{x} - \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}})^2}{2\|\mathbf{a}\|}(\hat{\mathbf{a}}\cdot\nabla\mathbf{a}) - \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}})(\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})}{\|\mathbf{a}\|}(\hat{\mathbf{a}}^{\perp}\cdot\nabla\mathbf{a})\right)\cdot\nabla u(\mathbf{x}) \\ &+ \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})^2}{2}\langle\hat{\mathbf{a}}^{\perp}, D^2U\cdot\hat{\mathbf{a}}^{\perp}\rangle + \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}})^2}{2\|\mathbf{a}\|}\hat{\mathbf{a}}\cdot\nabla f + \frac{(\delta\mathbf{x}\cdot\hat{\mathbf{a}})(\delta\mathbf{x}\cdot\hat{\mathbf{a}}^{\perp})}{\|\mathbf{a}\|}\hat{\mathbf{a}}^{\perp}\cdot\nabla f \\ &+ O(\|\delta\mathbf{x}\|^3). \end{aligned} \tag{3.13}$$

Notice that the only unknowns in the Taylor expansion (3.13) are $\nabla u(\mathbf{x})$ and $\langle\hat{\mathbf{a}}^{\perp}, D^2u(\mathbf{x})\cdot\hat{\mathbf{a}}^{\perp}\rangle$. Therefore, in 2-D, at each grid point (i,j), without loss of generality, assuming the upwind direction $a^x_{i,j} > 0, a^y_{i,j} > 0$, by applying the Taylor expansion (3.13) to $u_{i-1,j}, u_{i,j-1}, u_{i-1,j-1}$, we get three linear equations of the three unknowns, $\nabla u(\mathbf{x})$ and $\langle\hat{\mathbf{a}}^{\perp}, D^2u(\mathbf{x})\cdot\hat{\mathbf{a}}^{\perp}\rangle$. By solving the three linear equations, the three unknowns are computed in terms of $u_{i,j}, u_{i-1,j}, u_{i,j-1}, u_{i-1,j-1}$. The solutions give an approximation of $\nabla u(\mathbf{x})$ at (i,j), which is further substituted into the linear PDE (3.1) to get the discretized PDE that can be solved to obtain a local updating formula of $u_{i,j}$. For example, let us denote $\nabla u = (\alpha, \beta)$, $\langle\hat{\mathbf{a}}^{\perp}, D^2u(\mathbf{x})\cdot\hat{\mathbf{a}}^{\perp}\rangle = \gamma$ and assume $\mathbf{a}$ is constant locally. Then using (3.13), we have the following three equations

$$\begin{aligned} u_{i-1,j} &= u_{i,j} - \alpha h + \frac{h^2 {a^y_{i,j}}^2}{2}\gamma + F^x, \\ u_{i,j-1} &= u_{i,j} - \beta h + \frac{h^2 {a^x_{i,j}}^2}{2}\gamma + F^y, \\ u_{i-1.j-1} &= u_{i,j} - \alpha h - \beta h + \frac{h^2(a^x_{i,j} - a^y_{i,j})^2}{2}\gamma + F^{xy}, \end{aligned} \tag{3.14}$$

with

$$F^x = \frac{h^2 {a^x_{i,j}}^2}{2\|\mathbf{a}_{i,j}\|}\hat{\mathbf{a}}_{i,j}\cdot\nabla f_{i,j} - \frac{h^2 a^x_{i,j}a^y_{i,j}}{\|\mathbf{a}_{i,j}\|}\hat{\mathbf{a}}^{\perp}_{i,j}\cdot\nabla f_{i,j},$$
$$F^y = \frac{h^2 {a^y_{i,j}}^2}{2\|\mathbf{a}_{i,j}\|}\hat{\mathbf{a}}_{i,j}\cdot\nabla f_{i,j} + \frac{h^2 a^x_{i,j}a^y_{i,j}}{\|\mathbf{a}_{i,j}\|}\hat{\mathbf{a}}^{\perp}_{i,j}\cdot\nabla f_{i,j},$$
$$F^{xy} = \frac{h^2 (a^x_{i,j}+a^y_{i,j})^2}{2\|\mathbf{a}_{i,j}\|}\hat{\mathbf{a}}_{i,j}\cdot\nabla f_{i,j} + \frac{h^2 ({a^x_{i,j}}^2-{a^y_{i,j}}^2)}{\|\mathbf{a}_{i,j}\|}\hat{\mathbf{a}}^{\perp}_{i,j}\cdot\nabla f_{i,j}.$$

We can solve (3.14) to obtain α, β, γ,

$$\begin{aligned}
\alpha &= (1-\frac{a^y_{i,j}}{2a^x_{i,j}})\frac{u_{i,j}-u_{i-1,j}}{h} + \frac{a^y_{i,j}}{2a^x_{i,j}}\frac{u_{i,j-1}-u_{i-1,j-1}}{h} \\
&\quad + \frac{-2a^x_{i,j}F^x - a^y_{i,j}F^{xy} + a^y_{i,j}F^x + a^y_{i,j}F^y}{-2ha^x_{i,j}}, \\
\beta &= (1-\frac{a^x_{i,j}}{2a^y_{i,j}})\frac{u_{i,j}-u_{i,j-1}}{h} + \frac{a^x_{i,j}}{2a^y_{i,j}}\frac{u_{i-1,j}-u_{i-1.j-1}}{h} \\
&\quad + \frac{-2a^y_{i,j}F^y - a^x_{i,j}F^{xy} + a^x_{i,j}F^x + a^x_{i,j}F^y}{-2ha^y_{i,j}}, \\
\gamma &= \frac{u_{i-1,j}+u_{i,j-1}-u_{i-1,j-1}-u_{i,j}+F^{xy}-F^x-F^y}{h^2 a^x_{i,j}a^y_{i,j}}.
\end{aligned} \tag{3.15}$$

By rewriting α, β as $\alpha = g_1 u_{i,j} + g_2, \beta = g_3 u_{i,j} + g_4$ and substituting them into the PDE (3.1), we have the discretized PDE

$$a^x_{i,j}(g_1 u_{i,j} + g_2) + a^y_{i,j}(g_3 u_{i,j} + g_4) = f_{i,j}, \tag{3.16}$$

where,

$$\begin{aligned}
g_1 &= (1-\frac{a^y_{i,j}}{2a^x_{i,j}})\frac{1}{h}, \\
g_2 &= (1-\frac{a^y_{i,j}}{2a^x_{i,j}})\frac{-u_{i-1,j}}{h} + \frac{a^y_{i,j}}{2a^x_{i,j}}\frac{u_{i,j-1}-u_{i-1,j-1}}{h} + \frac{-2a^x_{i,j}F^x - a^y_{i,j}F^{xy} + a^y_{i,j}F^x + a^y_{i,j}F^y}{-2ha^x_{i,j}}, \\
g_3 &= (1-\frac{a^x_{i,j}}{2a^y_{i,j}})\frac{1}{h}, \\
g_4 &= (1-\frac{a^x_{i,j}}{2a^y_{i,j}})\frac{-u_{i,j-1}}{h} + \frac{a^x_{i,j}}{2a^y_{i,j}}\frac{u_{i-1,j}-u_{i-1.j-1}}{h} + \frac{-2a^y_{i,j}F^y - a^x_{i,j}F^{xy} + a^x_{i,j}F^x + a^x_{i,j}F^y}{-2ha^y_{i,j}}.
\end{aligned}$$

Hence we have the local updating formula for $u_{i,j}$ by solving (3.16) as

$$u_{i,j} = \frac{f_{i,j} - a^x_{i,j}g_2 - a^y_{i,j}g_4}{g_1 a^x_{i,j} + g_3 a^y_{i,j}}. \tag{3.17}$$

From the first two equations in (3.15), we can see that the scheme is not monotone and when $\mathbf{a}_{i,j}$ is close to the grid lines, i.e., either $a^x_{i,j} \to 0$ or $a^y_{i,j} \to 0$, the coefficients may go unbounded. As discussed in [3], at each point (i,j), the stencils must be chosen more carefully according to the upwind direction $\mathbf{a}_{i,j}$. At each point (i,j), the local mesh is divided into eight regions with equal inner angles (see Figure 3.1 (a)). If the upwind direction $\mathbf{a}_{i,j}$ is in region 1 (see Figure 3.1 (b)), then we choose stencil $\{(i,j),(i-1,j),(i,j-1),(i-1,j-1)\}$; if the upwind direction $\mathbf{a}_{i,j}$ is in region 2 (see Figure 3.1 (c)), then we choose stencil $\{(i,j),(i-1,j-1),(i,j-1),(i+1,j-1)\}$; and if the upwind direction $\mathbf{a}_{i,j}$ is in region 3 (see Figure 3.1 (d)), then we choose stencil $\{(i,j),(i-1,j),(i-1,j-1),(i-1,j+1)\}$. For other regions, the stencils can be chosen in a similar way.

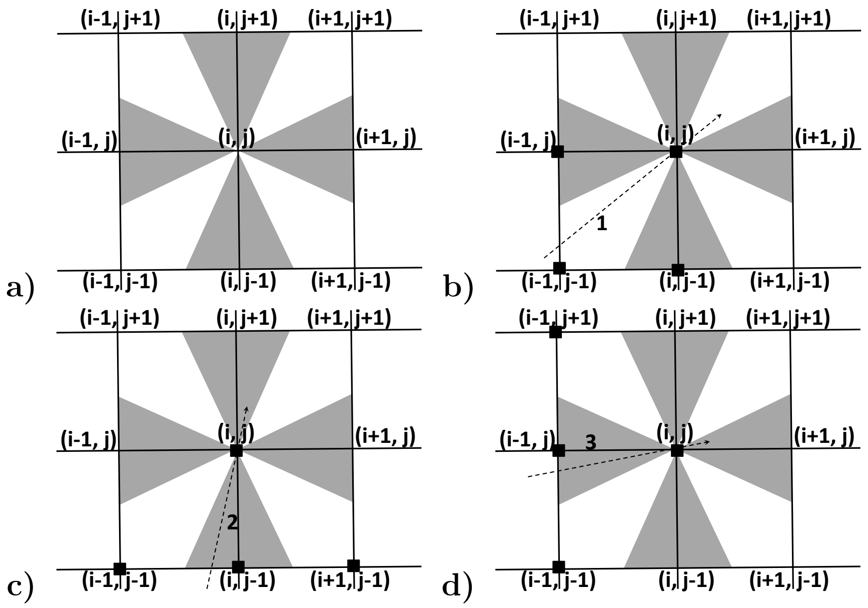

FIG. 3.1. *Local mesh and stencils for compact second-order scheme: (a) partition of the local mesh into* 8 *regions with equal inner angles; (b), (c) and (d) stencils (denoted as dark squares) used corresponding to the locations and directions of the characteristics (denoted as dashed arrow).*

3.3. Ordering.

Systematic and alternating orderings that can cover all directions of characteristics effectively is another crucial factor for the efficiency of the fast sweeping method. For the first-order upwind scheme and direction by direction second-order upwind scheme on rectangular grids, alternating the following four orderings in 2-D during Gauss-Seidel iterations is effective since information arrives at a grid point through one of the four quadrants.

$$\begin{aligned} &(1)\ i=1:I,\ j=1:J;\ (2)\ i=1:I,\ j=J:1;\\ &(3)\ i=I:1,\ j=1:J;\ (4)\ i=I:1,\ j=J:1. \end{aligned} \tag{3.18}$$

On the other hand, for the second-order upwind scheme with compact stencils, using the ordering introduced in [19] for triangular meshes is more effective since information arrives at a grid point through one of the eight sectors illustrated in Figure 3.1. The idea is to choose a few reference points, e.g., the four corners points of the computational domain. All grid points are ordered by the distance to each reference point in both increasing and decreasing order. Guass-Seidel iterations will be carried out according to these orderings alternately.

3.4. Schematic sketch of the fast sweeping method. With both upwind schemes and systematic orderings available, We summarize the fast sweeping methods below.

1. Initialization: impose the exact value of u at grid points on or near Γ, which are fixed during iterations; assign arbitrary values at all other points.
2. Gauss-Seidal iterations: sweep the whole mesh following alternating orderings: At each point (i, j), update $u_{i,j}$ with (3.4) or (3.7) or (3.17), One iteration corresponds to sweeping all the grid points once.
3. Termination: stop the iterations if two successive iterations have small changes, i.e., $|u^k - u^{k-1}|_\infty \leq \delta$ for some stopping criterion $\delta > 0$.

Narrow band implementation. In applications, the extrapolation is often needed only in a narrow band adjacent to the known region. The narrow band is typically defined by the level set function, e.g., a signed distance function. An easy modification of the fast sweeping method described above can serve the purpose. First we collect all grid points in the narrow band and create a list by a simple one pass region growing algorithm: starting with any point in the narrow band, we add its neighbors that are in the narrow band to the list, then we further add the neighbors of the newly added points that are in the narrow band but not in the current list yet to the updated list. We continue this process until no more points are added to the list. Since there is no regular data structure for points in the list, distance to a few references points is used to provide alternating orderings. The fast sweeping methods will be applied to the grid points on the list only, i.e., the PDEs will be solved numerically on the narrow band only.

3.5. Convergence Analysis. Here we provide a convergence study for the first-order scheme defined in Section 3.1. In Section 2, the extrapolation problem is formulated as a series of static linear hyperbolic PDEs in the form of (3.1), where the inflow boundary is the zero level set of the level set function $\psi(\mathbf{x})$ and $\mathbf{a}(\mathbf{x}) = \vec{\mathbf{n}}(\mathbf{x}) = \frac{\nabla\psi(\mathbf{x})}{|\nabla\psi(\mathbf{x})|}$. Assume $\nabla\psi(\mathbf{x})$ is continuous and $0 < c < |\nabla\psi| < C < \infty$ for $\psi \geq 0$ with c and C some constants. The computation domain Ω_h is discretized by a Cartesian grid with grid size h. For simplicity we present the analysis for 2D. Extension to higher dimensions is straightforward.

Define $\Omega_h^+ = \{(x_i, y_j) \in \Omega_h | \psi_{i,j} > 0\}$, $\Omega_h^- = \{(x_i, y_j) \in \Omega_h | \psi_{i,j} \leq 0\}$, and $\Gamma_h^\pm$ to be the set of boundary grid points, i.e., $\Gamma_h^- = \{(x_i, y_j) \in \Omega_h^-$ and at least one of its four neighbors $(x_{i\pm1}, y_j), (x_i, y_{j\pm1})$ belongs to $\Omega_h^+\}$ and $\Gamma_h^+ = \{(x_i, y_j) \in \Omega_h^+$ and at least one of its four neighbors $(x_{i\pm1}, y_j), (x_i, y_{j\pm1})$ belongs to $\Omega_h^-\}$. The value of $u_{i,j}$ is prescribed at Γ_h^-. Let $\mathbf{A}$ denote the matrix of the linear system for $\{u_{i,j}, (x_i, y_j) \in \Omega_h^+\}$ corresponding to the first-order upwind scheme defined by (3.3) with given inflow boundary condition for $u_{i,j}, (x_i, y_j) \in \Gamma_h^-$. We prove the following statements about the linear system.

THEOREM 3.1. *The discretized linear system has the following properties:*

(1) Maximum principle holds, i.e., if $f_{i,j} = 0$, the maximum or minimum value of $u_{i,j}$ is attained at the boundary Γ_h^-.
(2) $\boldsymbol{A}$ is a non-singular M-matrix.
(3) The fast sweeping method for the linear system converges.

Proof. (1) We first show $\max_{(x_i,y_j)\in\Omega_h^+\cup\Gamma_h^-} u_{i,j} = \max_{(x_i,y_j)\in\Gamma_h^-} u_{i,j}$. Suppose the maximum value of $u_{i,j}$ is attained at $(x_m, y_n) \in \Omega_h^+$. Without loss of generality assume $a^x_{m,n} \geq 0, a^y_{m,n} \geq 0$. From the upwind scheme we have

$$u_{m,n} = \frac{a^x_{m,n} u_{m-1,n}/h + a^y_{m,n} u_{m,n-1}/h}{a^x_{m,n}/h + a^y_{m,n}/h}.$$

So $u_{m-1,n} = u_{m,n-1} = u_{m,n}$. If either $(x_{m-1}, y_n) \in \Gamma^-_h$ or $(x_m, y_{n-1}) \in \Gamma^-_h$, the result is proved. Otherwise, the argument can be continued all the way to reach a point on the boundary Γ^-_h. Similarly, one can show $\min_{(x_i,y_j)\in\Omega^+_h\cup\Gamma^-_h} u_{i,j} = \min_{(x_i,y_j)\in\Gamma^-_h} u_{i,j}$.

(2) There are several equivalent definitions of an M-matrix [11, 18]. Denote $\mathbf{A} = (a_{i,j})$, due to the upwind scheme, $a_{i,j} \leq 0,\ i \neq j$. We show that $\mathbf{A}$ is non-singular and $\mathbf{A}^{-1} \geq 0$ [11]: (a) denote $\vec{\mathbf{u}}$ to be the vector of values $u_{i,j}$ at $(x_i, y_j) \in \Omega^+_h$ corresponding to $f_{i,j} = 0, (x_i, y_j) \in \Omega^+_h$ and $u_{i,j} = 0, (x_i, y_j) \in \Gamma^-_h$, i.e., $\mathbf{A}\vec{\mathbf{u}} = \vec{0}$. Without loss of generality, assume $u_{m,n} = \max_{i,j} u_{i,j} > 0$, from the the maximum principle proved above, it contradicts to the fact that $u_{i,j} = 0, (x_i, y_j) \in \Gamma^-_h$. So $\vec{\mathbf{u}} = \vec{0}$. Hence $\mathbf{A}$ is non-singular; and (b) we show that every element in any column of $\mathbf{A}^{-1}$ is non-negative. The j-th column of $\mathbf{A}^{-1}$ corresponds to a solution $\vec{\mathbf{u}} = \{u_{i,j} | (x_i, y_j) \in \Omega^+_h\}$ with $u_{i,j} = 0, (x_i, y_j) \in \Gamma^-_h$ and $\mathbf{A}\vec{\mathbf{u}} = \vec{\mathbf{e}}_j$, where $\vec{\mathbf{e}}_j$ is a column vector with j-th element equal to 1 and all other elements equal to 0. In another word, $\mathbf{u}$ corresponds to a discrete Green's function with zero Dirichlet boundary condition and a source at some grid point $(x_m, y_n) \in \Omega^+_h$, i.e., $f_{i,j} = 0$ if $(i,j) \neq (m,n)$ and $f_{m,n} = 1$, where (m,n) is indexed as j. Suppose $u_{p,q} = \min_{i,j} u_{i,j} < 0$, then it is easy to see $(p,q) \neq (m,n)$. Then apply the argument for maximum principle one can easily show $\min_{(x_i,y_j)\in\Gamma^-_h} u_{i,j} = u_{p,q} < 0$, which is a contradiction.

(3) Since the fast sweeping method is a Gauss-Seidel iteration with alternating orderings in each iteration, it converges for an M-matrix [11, 25].
□

Remark 2. *Although the fast sweeping algorithm is a type of Gauss-Seidel iteration, the use of different orderings during the iterations can greatly accelerate the convergence for linear or nonlinear systems arising from upwind discretization for hyperbolic problems. Once correct causality is enforced in the discretization and during the update at each grid point, alternating orderings allow efficient propagation of information along characteristics in all directions.*

Remark 3. *For the linear hyperbolic PDEs for the extrapolation problem, the main complication comes from the geometry of the inflow boundary Γ. Near the concave part of the boundary, the characteristics following $\frac{\nabla\psi}{|\nabla\psi|}$ converge. Near the convex part, the characteristics diverge. These scenarios are analogous to shocks and rarefactions for the hyperbolic conservation law. For the first-order upwind monotone scheme, there is an explicit numerical viscosity with coefficient of grid size h (by Taylor expansion of the finite difference scheme). So it can handle both cases. However, the divergence of characteristics near the convex part of the boundary can make one-pass algorithm fail as shown in numerical example 4 and Figure 4.4 in the next Section. This phenomenon can be explained from linear algebra point of view, the discretized system can not be transformed to a triangular system no matter what ordering is used for the grid points, which is also equivalent to the fact that the discretized problem becomes locally elliptic near the convex part of the boundary due to the numerical viscosity. Since the fast sweeping method is an iterative method, it works efficiently in all scenarios.*

The following theorem shows error estimate for the first-order upwind scheme. For simplicity, we state the result in 2D and extension to higher dimension is straightforward.

THEOREM 3.2. *Assume ψ is the signed distance function to a smooth interface Γ. Let $u(\boldsymbol{x})$ be the solution to* (3.1) *with $\boldsymbol{a}(\boldsymbol{x}) = \nabla\psi(\boldsymbol{x})/|\nabla\psi(\boldsymbol{x})|, f(\boldsymbol{x}) = 0$ and smooth boundary condition, and $u^h_{i,j}$ be the corresponding numerical solution to* (3.3). *For any point (x_i, y_j) in the region where $\boldsymbol{a}(\boldsymbol{x})$ is smooth,*

$$|u^h_{i,j} - u(x_i, y_j)| \le Ch \quad \text{as } h \to 0, \tag{3.19}$$

where $0 < C < \infty$ is some constant which depends on Γ and ψ.

Proof. Denote

$$e_{i,j} = u^h_{i,j} - u(x_i, y_j).$$

We order all grid points in Ω^+_h by layers such that the k-th layer contains all the points with $(k-1)h < \psi(x_i, y_j) \le kh$. We prove the error estimate (3.19) by induction. Assume

$$E_k = \max_{(k-1)h<\psi(x_m,y_n)\le kh} |e_{m,n}|$$

is attained at point (x_i, y_j), i.e., $E_k = |e_{i,j}|$. And let us assume $a^x_{i,j} \ge 0, a^y_{i,j} \ge 0$ without loss of generality. We have

$$a^x_{i,j}(e_{i,j} - e_{i-1,j}) + a^y_{i,j}(e_{i,j} - e_{i,j-1}) = T_{i,j}, \tag{3.20}$$

where $T_{i,j} = O(h^2)$ is the local truncation error and the constant in $O(\cdot)$ depends on the second derivatives of $u(\mathbf{x})$ which are bounded by the smoothness assumption. Then

$$e_{i,j} = \frac{a^x_{i,j}}{a^x_{i,j} + a^y_{i,j}} e_{i-1,j} + \frac{a^y_{i,j}}{a^x_{i,j} + a^y_{i,j}} e_{i,j-1} + \frac{T_{i,j}}{a^x_{i,j} + a^y_{i,j}},$$

which implies

$$|e_{i,j}| \le \frac{a^x_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i-1,j}| + \frac{a^y_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i,j-1}| + \frac{|T_{i,j}|}{a^x_{i,j} + a^y_{i,j}}.$$

Since ${a^x_{i,j}}^2 + {a^y_{i,j}}^2 = 1$, $a^x_{i,j} + a^y_{i,j} \ge 1$. We have the following cases:

(a) If both (x_{i-1}, y_j) and (x_i, y_{j-1}) are in the $(k-1)$-th layer, then $E_k \le E_{k-1} + O(h^2)$.

(b) If only one of (x_{i-1}, y_j), (x_i, y_{j-1}) is in the $(k-1)$-th layer (assume it is (x_{i-1}, y_j)), then

$$\begin{aligned} |e_{i,j}| &\le \frac{a^x_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i-1,j}| + \frac{a^y_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i,j-1}| + \frac{|T_{i,j}|}{a^x_{i,j} + a^y_{i,j}} \\ &\le \frac{a^x_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i-1,j}| + \frac{a^y_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i,j}| + \frac{|T_{i,j}|}{a^x_{i,j} + a^y_{i,j}} \\ &\Rightarrow \end{aligned}$$

$$\frac{a^x_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i,j}| \le \frac{a^x_{i,j}}{a^x_{i,j} + a^y_{i,j}} |e_{i-1,j}| + \frac{|T_{i,j}|}{a^x_{i,j} + a^y_{i,j}},$$

which implies

$$E_k \le |e_{i-1,j}| + |T_{i,j}|/a^x_{i,j} \le E_{k-1} + |T_{i,j}|/a^x_{i,j}.$$

Since $\psi_{i-1,j} < \psi_{i,j-1}$ and

$$\psi_{i-1,j} = \psi_{i,j} - a^x_{i,j} h + O(h^2), \quad \psi_{i,j-1} = \psi_{i,j} - a^y_{i,j} h + O(h^2),$$

we have $a^x_{i,j} > a^y_{i,j} - O(h)$. So $a^x_{i,j} > \frac{1}{\sqrt{2}} - O(h) > \frac{1}{2}$ if h is small enough. So $E_k \le E_{k-1} + 2|T_{i,j}|$.

(c) If both (x_{i-1}, y_j) and (x_i, y_{j-1}) are not in the $(k-1)$-th layer, let us assume $a^x_{i,j} \ge \frac{1}{\sqrt{2}} \ge a^y_{i,j} \ge 0$ without loss of generality. From (3.20), we have

$$a^x_{i,j}|e_{i,j} - e_{i-1,j}| + a^y_{i,j}|e_{i,j} - e_{i,j-1}| = O(h^2),$$

which implies

$$e_{i-1,j} = e_{i,j} + O(h^2). \tag{3.21}$$

Also we have

$$\psi_{i-1,j} = \psi_{i,j} - a^x_{i,j} h + O(h^2) \le \psi_{i,j} - \frac{h}{\sqrt{2}} + O(h^2) < \psi_{i,j} - \frac{h}{2} \tag{3.22}$$

for h small enough. For $e_{i-1,j}$ at point (x_{i-1}, y_j), it satisfies a relation as (3.20) with two neighbors coming from the upwind discretization determined by $(a^x_{i-1,j}, a^y_{i-1,j})$, for example,

$$a^x_{i-1,j}(e_{i-1,j} - e_{i-2,j}) + a^y_{i-1,j}(e_{i-1,j} - e_{i-1,j-1}) = T_{i-1,j},$$

which implies from (3.21)

$$a^x_{i-1,j}(e_{i,j} - e_{i-2,j}) + a^y_{i-1,j}(e_{i,j} - e_{i-1,j-1}) = O(h^2). \tag{3.23}$$

If either both (x_{i-2}, y_j) and (x_{i-1}, y_{j-1}) are or one of them is in $(k-1)$-th layer, then it is reduced to case (a) or (b) above and the proof is completed. Otherwise we repeat the previous step: if $a^x_{i-1,j} \ge a^y_{i-1,j}$, we consider $e_{i-2,j}$ at point (x_{i-2}, y_j) and have

$$e_{i-2,j} = e_{i,j} + O(h^2) \quad \text{and} \quad \psi_{i-2,j} < \psi_{i-1,j} - \frac{h}{2} < \psi_{i,j} - h < (k-1)h;$$

otherwise we consider $e_{i-1,j-1}$ at point (x_{i-1}, y_{j-1}) and have

$$e_{i-1,j-1} = e_{i,j} + O(h^2) \quad \text{and} \quad \psi_{i-1,j-1} < \psi_{i,j} - h < (k-1)h$$

for h small enough. Hence we get $E_k = |e_{i,j}| \le E_{k-1} + O(h^2)$.

Therefore, for any layer $k \ge 1$,

$$E_k = kO(h^2).$$

Since ψ is the signed distance function, for a domain of $O(1)$ size, there are at most $O(1/h)$ layers in total. Hence the global maximum error is $O(h)$.

□

Remark 4. *If Γ is convex, then $\boldsymbol{a}(\boldsymbol{x}) = \nabla\psi(\boldsymbol{x})$ is smooth everywhere. So the above error bound is true everywhere. If Γ is concave, then $\boldsymbol{a}(\boldsymbol{x})$ is smooth up to the shocks of the distance function where $\nabla\psi(\boldsymbol{x})$ is discontinuous. Also the above result can be extended to a general level set function $\psi(\boldsymbol{x})$ as long as $|\psi(\boldsymbol{x})| > c > 0$ for some constant c. The only difference is the number of layers in a bounded domain and the case (c) in the above proof may have to go through a few more but a finite number (depending on c) of recursive relations.*

4. Numerical Tests .

We present more numerical implementation details and a few examples to demonstrate both efficiency and accuracy of our approach. In all the examples, level set function, ψ, is the signed distance to the boundary between known and unknown region. Level set function is negative inside the known region.

4.1. Numerical Implementations.

For constant extrapolation in the normal direction, we solve (2.2) with the first-order fast sweeping method. The normal components $\vec{\mathbf{n}}$ are computed with the third-order Weighted Essentially Non-Oscillatory (WENO) finite difference approximations (e.g., see [26]).

For linear extrapolation in the normal direction, we solve (2.5) and (2.6) with the first-order fast sweeping method. $u_{\mathbf{n}} = \vec{\mathbf{n}} \cdot \nabla u$ is first computed with the third-order WENO approximations in the region $\psi \leq -band_1$. Then it is extrapolated in a constant manner to the whole domain by solving the modified equation (2.5), i.e.,

$$H(\psi + band_1)\vec{\mathbf{n}} \cdot \nabla u_{\mathbf{n}} = 0. \tag{4.1}$$

After that, we solve (2.6) to extrapolate u into the whole domain. Here $band_1 = h$.

For quadratic extrapolation in the normal direction, we solve (2.8), (2.9) and (2.10) with the second-order fast sweeping methods. $u_{\mathbf{nn}} = \vec{\mathbf{n}} \cdot \nabla(\vec{\mathbf{n}} \cdot \nabla u)$ is first computed in the region $\psi \leq -band_2$. Then it is extrapolated in a constant manner to the whole domain by solving the modified equation (2.8), i.e.,

$$H(\psi + band_2)\vec{\mathbf{n}} \cdot \nabla u_{\mathbf{nn}} = 0. \tag{4.2}$$

After that, $u_{\mathbf{n}}$ is computed by solving the modified equation (4.1). Finally u is computed by solving (2.10). Here $band_2 = h$.

Note that the Heaviside function serves as the indicator of unknown and known regions in the PDEs. In the numerical implementation, the PDEs only need to be solved in the unknown region where $H \equiv 1$.

REMARK 5. *For the computation of the derivatives in $\vec{\mathbf{n}}$, $u_{\mathbf{n}}$, $u_{\mathbf{nn}}$, besides the third-order WENO approximations, we also implement the centered finite differences. For constant and linear extrapolation, second-order centered finite difference is used ($band_1 = h$). For quadratic extrapolation, fourth-order centered finite difference is used ($band_1 = 2h$, $band_2 = 2\sqrt{2}h$). And the results show desired accuracy of extrapolation as in the following examples.*

4.2. Numerical Examples.

We choose $\delta = 10^{-9}$ as the stopping criterion for the fast sweeping methods. In computation of $\vec{\mathbf{n}}$, $u_{\mathbf{n}}$, $u_{\mathbf{nn}}$, we implement both the third-order WENO approximations and the centered finite difference approximations. Numerical results including the accuracy, the order of convergence and the number of iterations are recorded. One iteration corresponds to sweeping the whole mesh once.

Example 1: The computational domain is $[-\pi,\ \pi]\times[-\pi,\ \pi]$, the level set function is $\psi = \sqrt{x^2+y^2} - 2$. The function u is initially given as

$$u = \begin{cases} 0, & \text{if } \psi > 0, \\ \cos(x)\sin(y), & \text{if } \psi \leq 0. \end{cases} \tag{4.3}$$

Table 4.1 shows the accuracy of constant, linear and quadratic extrapolation. Figure 4.1 shows the contour plots of the numerical solutions. From Table 4.1, we observe first, second and third order convergence for constant, linear and quadratic extrapolation inside a narrow band of size $3h$ next to the interface Γ, respectively. And the number of iterations is almost constant as the mesh is refined.

Table 4.1

Numerical accuracy for extrapolation in a neighborhood of size $3h$ *near the interface* Γ*. The numbers of iterations correspond to solving the different PDEs: constant extrapolation: equation* (2.2)*; linear extrapolation:* (4.1) *and* (2.6)*; and quadratic extrapolation:* (4.2), (4.1) *and* (2.10).

Mesh	201×201	401×401	801×801	1601×1601
Third-Order WENO				
constant extrapolation				
L_∞ error	1.591E-2	7.585E-3	4.241E-3	2.093E-3
Conv. Order	—	1.069	0.839	1.019
# Iter	9	9	9	9
linear extrapolation				
L_∞ error	4.329E-3	1.079E-3	2.881E-4	7.387E-5
Conv. Order	—	2.004	1.905	1.963
# Iter	9, 9	9, 9	9, 9	9, 9
quadratic extrapolation				
L_∞ error	3.126E-4	3.736E-5	4.832E-6	5.975E-7
Conv. Order	—	3.065	2.951	3.016
# Iter	17, 17, 17	17, 17, 17	17, 17, 17	17, 17, 17
quadratic extrapolation (with compact upwind second-order scheme)				
L_∞ error	9.779E-4	1.450E-4	1.631E-5	2.042E-6
Conv. Order	—	2.754	3.152	2.998
# Iter	8, 8, 8	8, 8, 8	8, 8, 8	8, 8, 8
Centered finite difference				
constant extrapolation				
L_∞ error	1.591E-2	7.585E-3	4.241E-3	2.093E-3
Conv. Order	—	1.069	0.839	1.019
# Iter	5	5	5	5
linear extrapolation				
L_∞ error	5.711E-3	1.413E-3	3.892E-4	9.666E-5
Conv. Order	—	2.015	1.860	2.010
# Iter	5, 5	7, 5	5, 5	5, 5
quadratic extrapolation				
L_∞ error	1.070E-3	1.386E-4	1.809E-5	2.274E-6
Conv. Order	—	2.949	2.938	2.992
# Iter	28, 28, 17	28, 28, 17	28, 28, 17	28, 28, 17
quadratic extrapolation (with compact upwind second-order scheme)				
L_∞ error	1.977E-3	2.541E-4	3.214E-5	4.040E-6
Conv. Order	—	2.960	2.983	2.992
# Iter	8, 8, 8	8, 8, 8	8, 8, 8	8, 8, 8

Table 4.2 shows the comparison of CPUtime between the time-dependent approach [2] and the static approach for constant and linear extrapolation. For the time-dependent approach, we use second-order upwind finite difference scheme and second-order Runge-Kutta method as in [2]. The PDE must be solved to the steady state. In the numerical implementation, the computation stops when the change between two successive time steps is less than δ. For the static approach, we use second-order upwind scheme (3.7) for the purpose of comparison. It is clear that the time-dependent approach requires much more CPUtime than the static approach.

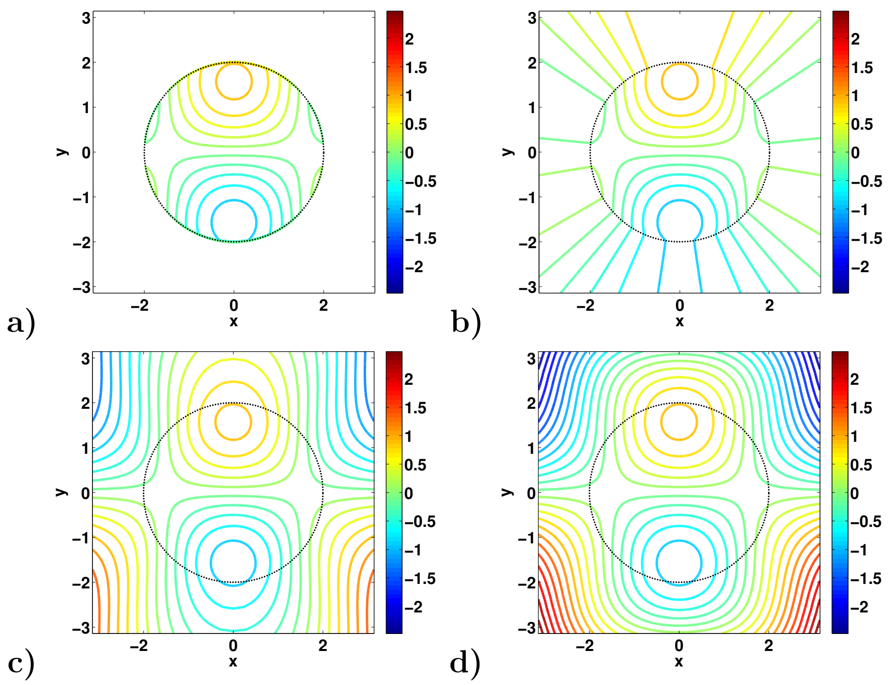

Fig. 4.1. *Contour plots of the numerical solutions: (a) initial u, (b) constant extrapolation, (c) linear extrapolation, and (d) quadratic extrapolation. Black dots indicate the interface* Γ. *Mesh:* 401×401.

Table 4.2

CPUtime (seconds) comparison between the time-dependent approach (TD) in [2] and the static approach. Same order of accuracy is obtained in the $3h$ *band near the interface for both approaches.*

Mesh	101×101	201×201	401×401	801×801
Centered finite difference, constant extrapolation				
CPUtime(Static)	0.01	0.03	0.10	0.51
CPUtime(TD)	0.11	0.68	6.35	68.96
Centered finite difference, linear extrapolation				
CPUtime(Static)	0.02	0.06	0.28	1.54
CPUtime(TD)	0.23	1.45	13.43	142.92

Both approaches are implemented with C codes on a Desktop.

Example 2: The setup is the same as in Example 1, except that the level set function is given as $\psi = \min\{\sqrt{(x-0.8)^2+y^2}-1, \sqrt{(x+0.8)^2+y^2}-1\}$. Table 4.3 shows the performance of the fast sweeping methods for constant, linear and quadratic extrapolation. Figure 4.2 shows the contour plots of the numerical solutions. From Table 4.3, we observe again that the number of iterations changes very little as the mesh is refined.

Example 3: This example demonstrates narrow band implementation described in Section 3.4. Four corner points of the computational domain are chosen as the reference points for alternating orderings. Figure 4.3 shows the results of extrapolation into a narrow band of size $10h$ near the interface Γ.

Example 4: Here we use a simple example to demonstrate that one pass algo-

TABLE 4.3

The numbers of iterations correspond to solving the different PDEs: constant extrapolation: equation (2.2)*; linear extrapolation:* (4.1) *and* (2.6)*; and quadratic extrapolation:* (4.2), (4.1) *and* (2.10).

Mesh	201×201	401×401	801×801	1601×1601
Third-order WENO				
constant extrapolation				
# Iter	21	17	19	21
linear extrapolation				
# Iter	21, 21	17, 17	19, 19	21, 21
quadratic extrapolation				
# Iter	25, 25, 25	23, 23, 23	25, 25, 25	25, 25, 25
quadratic extrapolation (with compact upwind second-order scheme)				
# Iter	12, 12, 12	14, 14, 14	14, 14, 14	12, 12, 12
Centered finite difference				
constant extrapolation				
# Iter	21	17	19	21
linear extrapolation				
# Iter	33, 21	29, 17	31, 19	33, 21
quadratic extrapolation				
# Iter	42, 41, 25	39, 37, 23	41, 39, 25	45, 43, 25
quadratic extrapolation (with compact upwind second-order scheme)				
# Iter	12, 12, 12	12, 12, 12	12, 12, 12	12, 12, 12

rithm with fixed upwind stencils does not work. For simplicity we use the first-order upwind scheme (3.4) as an example. At any given point, at most two of its four immediate neighbors in the upwind direction are used to determine its value. So the value at this point can get the correct value only if its upwind neighbors have correct values already. If a one pass algorithm with fixed stencil has to work, there has to be an ordering of all grid points such that every grid point can only depend on grid points that either have assigned value (from initialization) or have been updated correctly. Figure 4.4(a) shows a scenario where two grid points, **A** and **B**, next to the interface depend on each other. According to the upwind direction, shown by arrows in the figure, value at point **B** and **C** are required to determine the value at **A**. Similarly the value at point **A** and **D** are required to determine the value at **B**. In another word, the values at **A** and **B** are coupled in the upwind scheme and can not be solved one by one as in one pass algorithms. Figure 4.4(b) exactly verifies this scenario by enforcing a one pass condition in the sweeping algorithm for constant extrapolation. We update values only at those grid points where their upwind neighbors either have assigned value (from initialization) or have been updated correctly. We see that extrapolation get stuck immediately at the four points where the interface is tangent to the grid lines which is exactly demonstrated in Figure 4.4(a). Therefore the stencils must be modified on the fly, such as those used in the ordered upwind scheme [22]. On the other hand, the fast sweeping method is an iterative method which handles this situation naturally and effectively as investigated in [15]. The fast sweeping method does not only allow the fast propagation of information along characteristics by using upwind scheme, causality and alternating orderings during the iterations but also have an effective built in relaxation mechanism that can handle more complicated

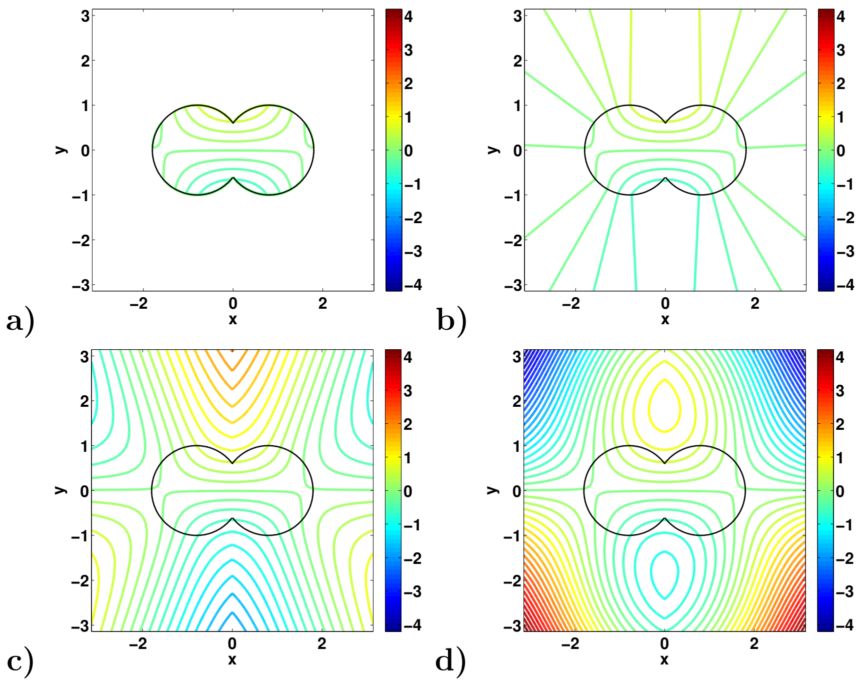

FIG. 4.2. *Contour plots of the numerical solutions: (a) initial u, (b) constant extrapolation, (c) linear extrapolation, and (d) quadratic extrapolation. Black dots indicate the interface* Γ. *Mesh:* 401×401.

situation when one pass algorithms fail.

Example 5: The setup is the same as in Example 1, except that the level set function is the signed distance function to a triangle (see Figure 4.5). In this case, the interface Γ is not smooth. The derivatives are computed with third-order WENO approximations. Figure 4.5 shows the contour plots of the numerical solutions.

5. Conclusion.

We developed the fast sweeping method for the static PDE based extrapolation which can be applied to a variety of physics problems often solved in conjunction with level set techniques. The proposed method is efficient, in that they scale linearly with the size of the grid, and eliminate the need to integrate in time to steady state as was done in the original approach [2]. Although parallelization techniques were not discussed here, approaches in [28, 6] can easily be applied to gain further efficiency.

Acknowledgment
The authors would like to thank the anonymous referees for helpful suggestions on this work.

Appendix A. Implementations on Triangular Meshes.

In this section, we present numerical implementations of the constant and linear extrapolation with first-order fast sweeping method on a triangular mesh. Assume that the computational domain is discretized by a triangular mesh. Given any point $\mathbf{C} = (x_{\mathbf{C}}, y_{\mathbf{C}})$ with its two neighbors $\mathbf{A} = (x_{\mathbf{A}}, y_{\mathbf{A}})$ and $\mathbf{B} = (x_{\mathbf{B}}, y_{\mathbf{B}})$ on the same simplex $\Delta\mathbf{CAB}$ (see Figure A.1), and assume that the characteristic passing through $\mathbf{C}$ falls in $\Delta\mathbf{CAB}$. By applying linear Taylor expansion within $\Delta\mathbf{CAB}$, we have

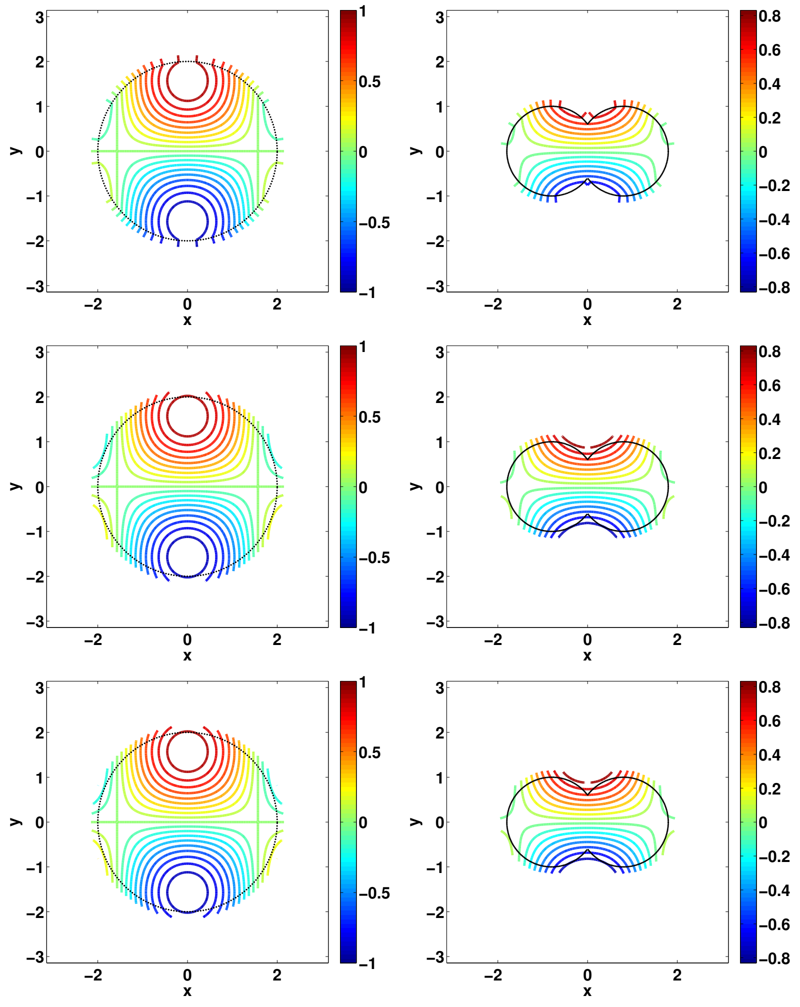

FIG. 4.3. *Contour plots of the numerical solutions with extrapolation to a narrow band of size $10h$ near Γ: from top to bottom: constant, linear and quadratic extrapolation, respectively. Left: setup as in Example 1; Right: setup as in Example 2. Black dots indicate the interface Γ. Mesh size: $h = \pi/200$.*

$$\nabla u(\mathbf{C}) \approx \mathbf{P}^{-1}\begin{pmatrix} \frac{u(\mathbf{C})-u(\mathbf{A})}{l_b} \\ \frac{u(\mathbf{C})-u(\mathbf{B})}{l_a} \end{pmatrix}, \text{ with } \mathbf{P} = \begin{pmatrix} (x_{\mathbf{C}}-x_{\mathbf{A}})/l_b, & (y_{\mathbf{C}}-y_{\mathbf{A}})/l_b \\ (x_{\mathbf{C}}-x_{\mathbf{B}})/l_a, & (y_{\mathbf{C}}-y_{\mathbf{B}})/l_a \end{pmatrix}, \tag{A.1}$$

where $l_a = |\mathbf{C}-\mathbf{B}|$, and $l_b = |\mathbf{C}-\mathbf{A}|$. If denoting $\mathbf{P}^{-1} = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}$, we have

$$\begin{aligned} &\nabla u(\mathbf{C}) \approx \begin{pmatrix} g_1 u(\mathbf{C}) + g_2 \\ g_3 u(\mathbf{C}) + g_4 \end{pmatrix}, \\ &g_1 = p_{11}/l_b + p_{12}/l_a, \quad g_2 = -(u(\mathbf{A})p_{11}/l_b + u(\mathbf{B})p_{12}/l_a), \\ &g_3 = (p_{21}/l_b + p_{22}/l_a), \quad g_4 = -(u(\mathbf{A})p_{21}/l_b + u(\mathbf{B})p_{22}/l_a). \end{aligned} \tag{A.2}$$

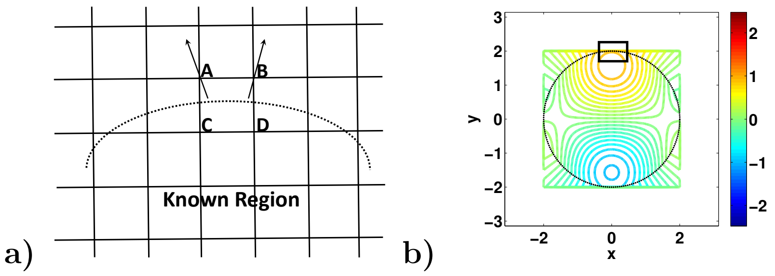

FIG. 4.4. *Numerical test for one pass algorithm. (a) demonstration of upwind stencils for two special points next to the interface (the part inside the rectangle of (b)). Black dots indicate the interface. Arrows indicate upwind directions. (b) numerical test for one pass algorithm for constant extrapolation.*

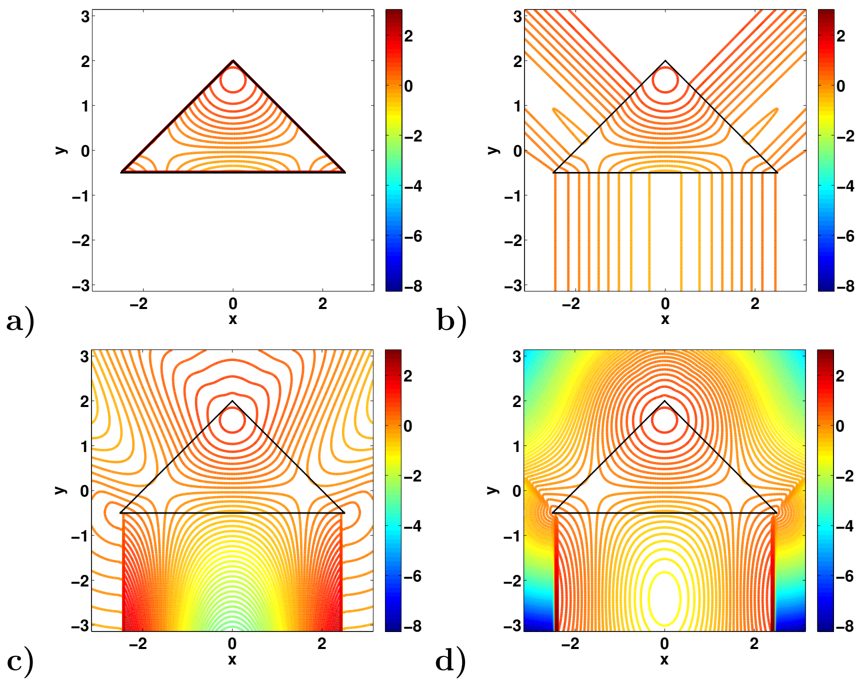

FIG. 4.5. *Contour plots of the numerical solutions: (a) initial u, (b) constant extrapolation, (c) linear extrapolation, and (d) quadratic extrapolation. Black dots indicate the interface* Γ. *Mesh:* 401×401.

We substitute $\nabla u(\mathbf{C})$ into the PDE (3.1) to have the discretized PDE on $\Delta\mathbf{CAB}$,

$$a^x(x_{\mathbf{C}}, y_{\mathbf{C}})(g_1 u(\mathbf{C}) + g_2) + a^y(x_{\mathbf{C}}, y_{\mathbf{C}})(g_3 u(\mathbf{C}) + g_4) = f(\mathbf{C}), \tag{A.3}$$

which provides a local updating formula for $u(\mathbf{C})$,

$$u(\mathbf{C}) = \frac{f(\mathbf{C}) - a^x(x_{\mathbf{C}}, y_{\mathbf{C}})g_2 - a^y(x_{\mathbf{C}}, y_{\mathbf{C}})g_4}{a^x(x_{\mathbf{C}}, y_{\mathbf{C}}) + g_1 + a^y(x_{\mathbf{C}}, y_{\mathbf{C}})g_3}. \tag{A.4}$$

With the local solver (A.4), we can simply use the fast sweeping algorithm summarized in Section 3.4. By choosing a few reference points, triangular meshes are ordered according to the distance to these reference points as in [19]. We apply the fast

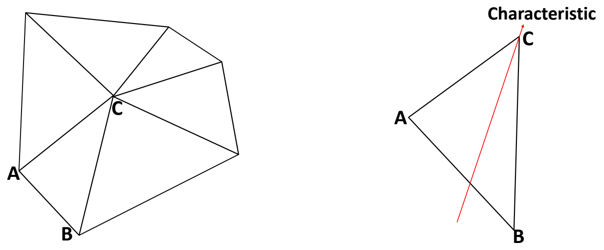

FIG. A.1. *Illustrations of local triangular mesh: left: local mesh; right: a simplex where the characteristic passes through* **c** *falls into.*

TABLE A.1
Numerical accuracy for extrapolation in a neighborhood of size 3h near the interface Γ.

(Nodes, Elements)	$(362, 661)$	$(1452, 2774)$	$(5809, 11366)$	$(23227, 45946)$
constant extrapolation				
L_∞ error	1.910E-1	1.017E-1	5.100E-2	2.470E-2
Conv. Order	—	0.909	0.996	1.046
# Iter	8	10	10	14
linear extrapolation				
L_∞ error	1.762E-1	3.808E-2	1.2201E-2	2.550E-3
Conv. Order	—	2.210	1.642	2.258
# Iter	8, 8	10, 10	12, 10	14, 14

sweeping method for Examples 1 and 2 on triangular meshes. The computation domain is now given as a disk centered at the origin with radius π. And the domain is discretized with triangular mesh, e.g., see Figure A.2.

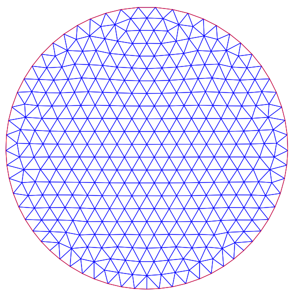

FIG. A.2. *A Triangular mesh.*

Table A.1 shows the accuracy for constant and linear extrapolation for Example 1. The order of convergence is expected as in Example 1. For linear extrapolation, for simplicity, we compute $u_{\mathbf{n}}$ directly with u initially given as in Example 1.

Figure A.3 (Example 1) and Figure A.4 (Example 2) show plots of numerical solutions by constant and linear extrapolation.

REFERENCES

[1] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, 148(1):2–22, 1999.

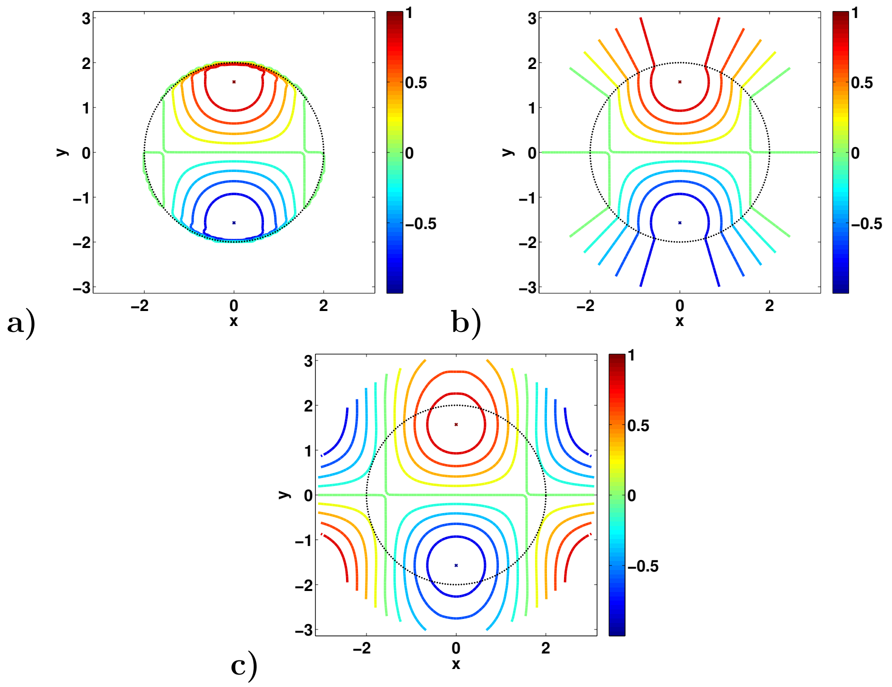

FIG. A.3. *Contour plots of the numerical solutions: (a) initial u, (b) constant extrapolation, and (c) linear extrapolation. Mesh: 5809 nodes and 11366 elements. Black dots indicate the interface* Γ.

[2] T. D. Aslam. A partial differential equation approach to multidimensional extrapolation. *J. Comput. Phys.*, 193(1):349–355, 2004.

[3] J. D. Benamou, S. Luo, and H.-K. Zhao. A Compact Upwind Second Order Scheme for the Eikonal Equation. *J. Comput. Math.*, 28:489–516, 2010.

[4] R. Bridson. *Fluid Simulation for Computer Graphics*. CRC Press, 2008.

[5] E. Can and A. Prosperetti. A level set method for vapor bubble dynamics. *J. Comput. Phys.*, 231(4):1533–1552, 2012.

[6] M. Detrixhe, F Gibou, and C. Min. A parallel fast sweeping method for the Eikonal equation. *J. Comput. Phys.*, 237:46–55, 2013.

[7] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152(2):457–492, 1999.

[8] S. Fomel, S. Luo, and H.-K. Zhao. Fast sweeping method for the factored eikonal equation. *J. Comput. Phys.*, 228(17):6440–6455, 2009.

[9] F. Gibou and R. Fedkiw. A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem. *J. Comput. Phys.*, 202(2):557–601, 2005.

[10] F. Gibou, C. Min, and R. Fedkiw. High Resolution Sharp Computational Methods for Elliptic and Parabolic Problems in Complex Geometries. *J. Sci. Comput.*, 54(2-3):369–413, 2013.

[11] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, 1997.

[12] R. W. Houim and K. K. Kuo. A ghost fluid method for compressible reacting flows with phase change. *J. Comput. Phys.*, 235:865–900, 2013.

[13] C. Y. Kao, S. Osher, and Y. H. Tsai. Fast sweeping method for static Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 42:2612–2632, 2005.

[14] S. Luo. A uniformly second order fast sweeping method for eikonal equations. *J. Comput. Phys.*, 241:104–117, 2013.

[15] S. Luo and H.-K. Zhao. The Convergence Study of Fast Sweeping Method. *Preprint*.

[16] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *J. Comput. Phys.*, 225(1):300–321, 2007.

[17] M. Mirzadeh, M. Theillard, and F. Gibou. A second-order discretization of the nonlinear

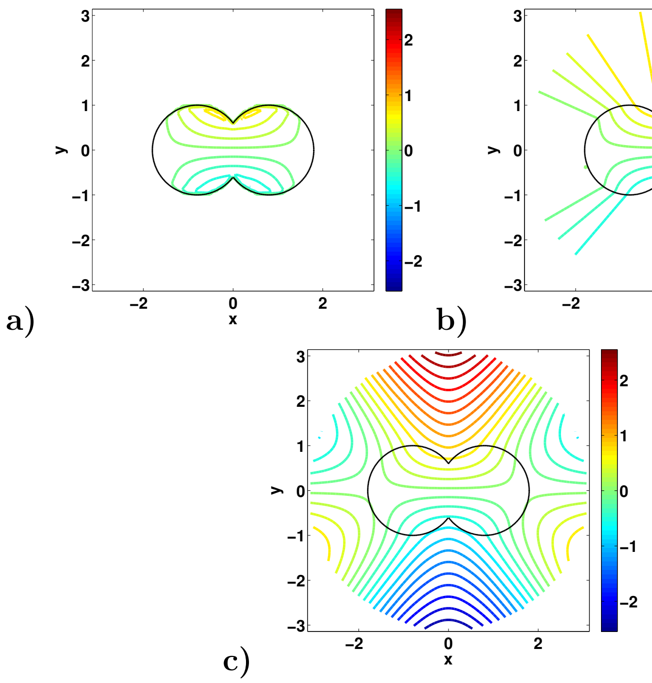

Fig. A.4. *Contour plots of the numerical solutions: (a) initial* u*, (b) constant extrapolation, and (c) linear extrapolation. Mesh: 5809 nodes and 11366 elements. Black dots indicate the interface* Γ*.*

Poisson–Boltzmann equation over irregular geometries using non-graded adaptive Cartesian grids. *J. Comput. Phys.*, 230(5):2125–2140, 2011.

[18] G. Poole and T. Boullion. A survey on M-matrices. *SIAM Review*, 16(4):419–427, 1974.

[19] J. Qian, Y.-T. Zhang, and H.-K. Zhao. A fast sweeping methods for static convex Hamilton-Jacobi equations. *J. Sci. Comput.*, 31(1/2):237–271, 2007.

[20] J. Qian, Y.-T. Zhang, and H.-K. Zhao. Fast sweeping methods for eikonal equations on triangulated meshes. *SIAM J. Numer. Anal.*, 45:83–107, 2007.

[21] J. A. Sethian. A Fast Marching Level Set Method for Monotonically Advancing Fronts. *Proceedings of the National Academy of Sciences*, 93(4), 1996.

[22] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms. *SIAM J. Numer. Anal.*, 41:325–363, 2003.

[23] S. Tanguy, T. Menard, and A. Berlemont. A Level Set Method for vaporizing two-phase flows. *J. Comput. Phys.*, 221(2):837–853, 2007.

[24] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. Auto. Control*, 40:1528–1538, 1995.

[25] R. S. Varga. *Matrix Iterative Analysis (Springer Series in Computational Mathematics)*. Springer Berlin Heidelberg, 2009.

[26] Y.-T. Zhang, H.-K. Zhao, and J. Qian. High order fast sweeping methods for static Hamilton-Jacobi equations. *J. Sci. Comput.*, 29:25–56, 2006.

[27] H.-K. Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74:603–627, 2005.

[28] H.-K. Zhao. Parallel Implementations of the Fast Sweeping Method. *J. Comput. Math.*, 25:421–429, 2007.