Journal of Computational Physics 274 (2014) 393–412

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A fast marching approach to multidimensional extrapolation

Jeremy O. McCaslin [*], Émilien Courtine, Olivier Desjardins

Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA

ARTICLE INFO

Article history:
Received 14 February 2014
Received in revised form 4 June 2014
Accepted 12 June 2014
Available online 19 June 2014

Keywords:
Fast marching method
Extrapolation
Hamilton–Jacobi equation
Eikonal equation
Level set method

ABSTRACT

A computationally efficient approach to extrapolating a data field with second order accuracy is presented. This is achieved through the sequential solution of non-homogeneous linear static Hamilton–Jacobi equations, which can be performed rapidly using the fast marching methodology. In particular, the method relies on a fast marching calculation of the distance from the manifold Γ that separates the subdomain Ω_{in} over which the quantity is known from the subdomain Ω_{out} over which the quantity is to be extrapolated. A parallel algorithm is included and discussed in the appendices. Results are compared to the multidimensional partial differential equation (PDE) extrapolation approach of Aslam (Aslam (2004) [31]). It is shown that the rate of convergence of the extrapolation within a narrow band near Γ is controlled by both the number of successive extrapolations performed and the order of accuracy of the spatial discretization. For m successive extrapolating steps and a spatial discretization scheme of order N, the rate of convergence in a narrow band is shown to be $\min(N+1, m+1)$. Results show that for a wide range of error levels, the fast marching extrapolation strategy leads to dramatic improvements in computational cost when compared to the PDE approach.

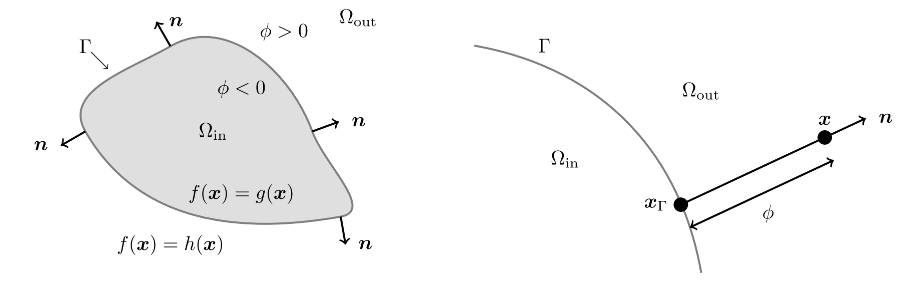
© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Many applications of computational science and engineering involve extrapolating data located in a subregion of a computational space to the rest of the simulation domain. This is especially pertinent to the propagation of discontinuous fronts, which is a significant component of phenomena such as multiphase flows [1–10], supersonic flows [11], reacting flows [12–14], multiphase electrohydrodynamics [15,16], crack propagation [17,18], and image processing [19–21]. Calculations involving discontinuous fronts often suffer from numerical artifacts such as nonphysical oscillations and low orders of convergence [22,23], unless relevant quantities are extrapolated or extended across the front in order to avoid differentiating across discontinuities. A typical way of achieving this is by performing a constant extrapolation through the solution of a Hamilton–Jacobi type equation, as is the case in the original ghost fluid method [22,24,25] and a variety of other level set applications [1,26,2]. Another approach that circumvents Courant–Friedrichs–Lewy (CFL) limitations of a partial differential equation (PDE) extrapolation is the fast marching method (FMM), developed by Sethian [27] and Adalsteinsson and Sethian [28] for the solution of static Hamilton–Jacobi equations in the context of level set methods [29]. Constant FMM extrapolation has been recently employed to improve the accuracy of the conservative level set method [9]. The fast marching approach is proven to be much faster than a PDE approach, but Aslam [30] has argued that it would suffer from lower accuracy and a reduced rate of convergence under mesh refinement in some instances.

[*] Corresponding author.
E-mail address: jom48@cornell.edu (J.O. McCaslin).

http://dx.doi.org/10.1016/j.jcp.2014.06.023
0021-9991/© 2014 Elsevier Inc. All rights reserved.



(a) Various parameters for the extrapolation. (b) Close-up version of image to the left.

Fig. 1. Schematics illustrating the extrapolation scenario.

Multidimensional extrapolation in the context of the PDE method has been described by Aslam [31] and subsequently used in application to the Stefan problem [32,33]. Despite its straightforward adaptivity to multidimensional extrapolation, the FMM has not yet been analyzed in detail for higher order extrapolations, which may be attributed to the aforementioned concerns regarding rate of convergence and accuracy. The goal of this study is to present a quantitative comparison between the PDE-based and FMM-based extrapolation methods, analyzing both the speed and accuracy of each method.

This paper is organized as follows: in the next section, the mathematical framework for multidimensional extrapolation is given, including a derivation of the expected order of accuracy, followed by discussions of both PDE and FMM solution approaches in Sections 2.3 and 2.4, respectively. Provided in Section 3 is a discussion of the PDE and FMM implementations used in this work. The PDE methodology is similar to the implementation of Aslam [31]. Parallel implementation of the FMM procedure for level set re-initialization and subsequent data extrapolation is provided in Appendix A and Appendix B, respectively. Section 4 presents multidimensional extrapolation results for the canonical test case used by Aslam [31]. Finally, this test case is also adapted to an additional shape in Section 5, revealing some subtleties of the method that are worth consideration. From these tests it is shown that the FMM provides a computationally efficient means of performing second order accurate extrapolation.

2. Mathematical formulation

2.1. Problem description

Consider a domain Ω that contains a continuous surface Γ, such that it divides Ω into an inner subdomain Ω_{in} and an outer subdomain Ω_{out}, as illustrated in Fig. 1(a). Then consider the function $g(\boldsymbol{x})$ to be extrapolated, defined for all $\boldsymbol{x} \in \Omega$. To simplify the discussion, the function $g(\boldsymbol{x})$ is considered here to be a scalar without loss of generality. There exists a signed distance level set $\phi(\boldsymbol{x}) = \pm\|\boldsymbol{x} - \boldsymbol{x}_\Gamma\|$, where $\boldsymbol{x}_\Gamma$ is the location on Γ that provides the minimum Euclidean distance from location $\boldsymbol{x}$, as shown in Fig. 1(b). The sign of ϕ is negative in the domain Ω_{in} and positive in the domain Ω_{out}. The level set ϕ is an auxiliary function to our discussion and, given ϕ, a smooth field of normal vectors is obtained from

$$\boldsymbol{n} = \frac{\nabla\phi}{\|\nabla\phi\|}, \tag{1}$$

oriented outward from Ω_{in}.

For any point $\boldsymbol{x} \in \Omega_{\text{out}}$, the function $h(\boldsymbol{x})$ is defined as the mth order Taylor Series expansion of g from $\boldsymbol{x}_\Gamma$, i.e.,

$$h(\boldsymbol{x}) = \sum_{k=0}^{m} \frac{\phi^k}{k!} \frac{\partial^k g}{\partial\phi^k}\bigg|_{\boldsymbol{x}_\Gamma}. \tag{2}$$

Note that the $\boldsymbol{x}$ dependence of ϕ has been dropped for simplicity. We will build our extrapolation of g in the form of the function $f(\boldsymbol{x})$, written as

$$f(\boldsymbol{x}) = \begin{cases} g(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \Omega_{\text{in}}, \\ h(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \Omega_{\text{out}}. \end{cases} \tag{3}$$

It is clear that $f(\boldsymbol{x})$ should be $\mathcal{C}^m$ continuous across Γ.

2.2. Expected accuracy

When obtaining f for a set of numerical data, it is clear from Eqs. (2) and (3) that the resulting accuracy of the extrapolated f field in Ω_{out} will depend on both m and the accuracy of the discrete representation of $\partial^k g/\partial\phi^k|_{\boldsymbol{x}_\Gamma}$. Assuming that $\boldsymbol{n}$ is horizontal and $\boldsymbol{x}_\Gamma = 0$ such that the distance from Γ is described by the variable x, we introduce a discrete upwind operator $\mathcal{U}$ that approximates the first derivative of a function with order of accuracy N such that

$$g_i' = \mathcal{U}(g_i) + \mathcal{O}(\Delta x^N), \tag{4}$$

where $g_i = g(x_i)$ is the value of g at location x_i, Δx is the characteristic mesh spacing, and the prime is used to denote $\partial/\partial x$. Recursively applying the operator $\mathcal{U}$ to higher derivatives of g leads to

$$g_i^{(k+1)} = \mathcal{U}\big(g_i^{(k)}\big) + \mathcal{O}\big(\Delta x^{N-k}\big), \tag{5}$$

where the superscript (k) is used to denote the order of the derivative in x.

Eqs. (2) and (3) provide $f(x_i)$ as a Taylor Series expansion about $x = 0$ with $m + 1$ terms, such that

$$f(x_i) = h(x_i) + \mathcal{O}\big(x_i^{m+1}\big). \tag{6}$$

The kth derivative $g^{(k)}(x)|_{x=0}$ necessary to obtain $h(x_i)$ via Eq. (2) is written as

$$g_0^{(k)} = \mathcal{U}\big(g_0^{(k-1)}\big) + \mathcal{O}\big(\Delta x^{N-(k-1)}\big), \tag{7}$$

where Eq. (5) has been used. Combining Eqs. (2), (6), and (7) leads to

$$f(x_i) = \sum_{k=0}^{m} \frac{x_i^k}{k!}\mathcal{U}\big(g_0^{(k-1)}\big) + \mathcal{O}\big(x_i^k \Delta x^{N-k+1}\big) + \mathcal{O}\big(x_i^{m+1}\big). \tag{8}$$

We rewrite Eq. (8) as

$$f(x_i) = \widetilde{h}(x_i) + \mathcal{O}\big(x_i^k \Delta x^{N-k+1}\big) + \mathcal{O}\big(x_i^{m+1}\big), \tag{9}$$

where $\widetilde{h}(x_i)$ is the discrete approximation to $h(x_i)$. Eq. (9) indicates that the rate of convergence will be controlled by either $\mathcal{O}(x_i^k \Delta x^{N-k+1})$ or $\mathcal{O}(x_i^{m+1})$. If $x_i \sim \Delta x$, which applies when extrapolating to a narrow band around Γ, Eq. (9) becomes

$$f(x_i) = \widetilde{h}(x_i) + \mathcal{O}\big(\Delta x^{\min(N+1,m+1)}\big). \tag{10}$$

This dictates that the convergence rate $\mathcal{R}_{\text{band}}$ of the f field within a narrow band around Γ is given by

$$\mathcal{R}_{\text{band}} = \min(N+1, m+1). \tag{11}$$

Obtaining f in the subregion Ω_{out} amounts to solving a boundary value problem in Ω_{out} based on boundary conditions on Γ. This can be accomplished in different ways, and this work compares the PDE-based extrapolation of Aslam [31] with the FMM-based extrapolation developed herein. The mathematical formalism for the two approaches is discussed below, followed by a discussion of the implemented equations when performing this procedure on numerical data.

2.3. Partial differential extrapolation

As done by Aslam [31], one way to obtain f is to solve a PDE in the subregion Ω_{out} to a steady state solution that has the desired properties. We outline the procedure here for completeness, first for constant, or 0th order, extrapolation, i.e., extrapolation that results in a field f that is $\mathcal{C}^0$ continuous across Γ. Next, Aslam's procedure [31] for mth order extrapolation that results in a field f that is $\mathcal{C}^m$ continuous across Γ is described. The descriptors "order" or "high order" when referencing the extrapolation refer to the order of m, and should not be confused with the order of accuracy of the spatial discretization used when discretizing the PDE in later sections.

2.3.1. Constant extrapolation

The f field can be constructed within Ω_{out} by solving in pseudo-time τ the Hamilton–Jacobi equation

$$\frac{\partial f}{\partial \tau} + \mathcal{H}(\nabla f) = 0, \tag{12}$$

where the Hamiltonian $\mathcal{H}$ is a function of the gradient of f. For constant extrapolation, $\mathcal{H}(\nabla f) = \nabla f \cdot \boldsymbol{n}$, leading to

$$\frac{\partial f}{\partial \tau} + \nabla f \cdot \boldsymbol{n} = 0. \tag{13}$$

The boundary of Ω_{out} is Γ, and the boundary and initial conditions for this PDE are

$$\begin{aligned} &f(\tau, \boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{for } \tau > 0,\ \forall \boldsymbol{x} \in \Gamma \\ &f(0, \boldsymbol{x}) = 0, \quad \forall \boldsymbol{x} \in \Omega_{\text{out}}. \end{aligned} \tag{14}$$

Note that the initial condition is arbitrary since the PDE is solved to steady state. The resulting f field within Ω_{out} will be constant along the direction of $\boldsymbol{n}$, and $f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ will be given by Eq. (3) with $m = 0$.

2.3.2. Higher order extrapolation

Consider again Eq. (13), but replace the quantity f with $f_n = \partial f/\partial n$ to get

$$\frac{\partial f_n}{\partial \tau} + \nabla f_n \cdot \boldsymbol{n} = 0, \tag{15}$$

subject to the new conditions

$$\begin{aligned} f_n(\tau, \boldsymbol{x}) &= \left.\frac{\partial g}{\partial n}\right|_{\Gamma} \quad \text{for } \tau > 0, \quad \forall \boldsymbol{x} \in \Gamma \\ f_n(0, \boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \Omega_{\text{out}}. \end{aligned} \tag{16}$$

After extrapolating f_n to Ω_{out} by solving Eq. (15) to steady state, we then solve

$$\frac{\partial f}{\partial \tau} + \nabla f \cdot \boldsymbol{n} = f_n \tag{17}$$

to steady state, subject to the conditions in Eq. (14). Having performed the additional extrapolation of f_n, the resulting f field within Ω_{out} will be such that $\partial f/\partial n$ is constant along the direction of $\boldsymbol{n}$, and $f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ will be given by Eq. (3) with $m = 1$. Naturally, one could construct a field f such that $f_{nn} = \partial^2 f/\partial n^2$ is constant along $\boldsymbol{n}$ by first extrapolating f_{nn} with a zero right-hand side, followed by extrapolation of f_n with f_{nn} as the right-hand side, and so on.

2.4. Fast marching extrapolation

An alternative way to construct the field f rather than converging a PDE to steady state is to directly solve the static Hamilton–Jacobi equation

$$\nabla f \cdot \nabla \phi = 0 \tag{18}$$

using an FMM. Indeed, if f represents the normal velocity of the interface Γ, then this equation is the basis for the well known velocity extension method [28,34,35] used in level set methods. The FMM is a numerical procedure, but its mathematical formalism that parallels the PDE approach is provided for consistency.

Recalling that $\boldsymbol{n} = \nabla\phi/\|\nabla\phi\|$, Eq. (18) is written as

$$\nabla f \cdot \boldsymbol{n} = 0, \tag{19}$$

which is the static form of Eq. (13). This equation is a boundary value problem subject to the boundary condition in Eq. (14), namely $f(\boldsymbol{x}) = g(\boldsymbol{x})\ \forall \boldsymbol{x} \in \Gamma$. Using an FMM to solve Eq. (19) in the region Ω_{out} subject to this boundary condition results in a field f within Ω_{out} that is constant along $\boldsymbol{n}$.

The fast marching solution of the static equation yields the same result (subject to different numerical errors) as PDE-based extrapolation, but without the cost of converging the time-dependent PDE to steady state. This approach is extended to higher order extrapolations in the same manner as the PDE approach, the difference being that the FMM is used to solve the successive Hamilton–Jacobi equations in their static form.

3. Numerical formulation

3.1. PDE-based extrapolation

Following the framework of Aslam [31], we introduce a Heaviside function $H_0(\boldsymbol{x})$ to account for the Dirichlet boundary condition on Γ when performing constant extrapolation numerically, defined as

$$H_0(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \Omega_{\text{out}}, \\ 1 & \text{otherwise.} \end{cases} \tag{20}$$

Thus, for constant PDE-based extrapolation, f is initialized as equal to the function g in Ω_{in}, and the steady-state solution of

$$\frac{\partial f}{\partial \tau} + \big(1 - H_0(\boldsymbol{x})\big)\nabla f \cdot \boldsymbol{n} = 0 \tag{21}$$

provides the extrapolated values $f = h$ in the region Ω_{out}, leaving the values $f = g$ in Ω_{in} unaltered. The resulting f field is constant along $\boldsymbol{n}$ within Ω_{out} and $\mathcal{C}^0$ continuous across Γ.

Extrapolations with $m > 0$ require the computation of $\partial^q g/\partial n^q$ for $q = m, m-1, \dots, 1$ within Ω_{in}, evaluated with second order central differences. As noted by Aslam [31], only points within Ω_{in} can be used in calculating $\partial^q g/\partial n^q$, so the region of points where $\partial^q g/\partial n^q$ can be computed decreases with increasing q. In general, we account for this by creating the function

$$\mathcal{B}(\boldsymbol{x}) = \left\lfloor \min\left(\frac{|x - x_\Gamma|}{\Delta x}, \frac{|y - y_\Gamma|}{\Delta y}, \frac{|z - z_\Gamma|}{\Delta z}\right) \right\rfloor \tag{22}$$

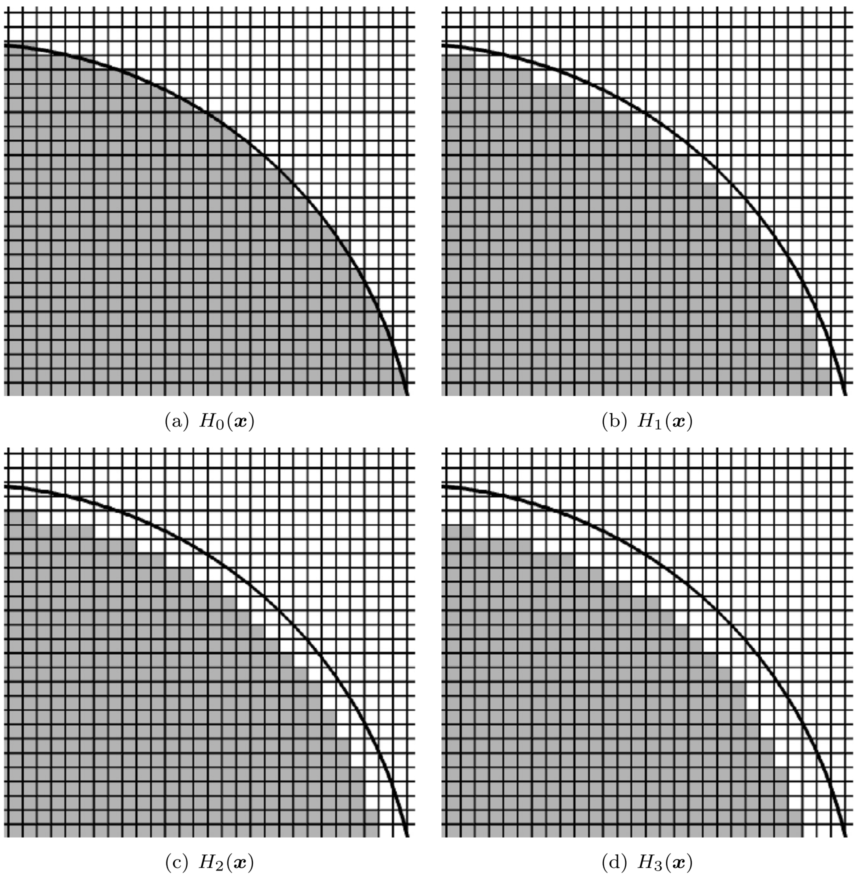

(a) $H_0(\boldsymbol{x})$
(b) $H_1(\boldsymbol{x})$
(c) $H_2(\boldsymbol{x})$
(d) $H_3(\boldsymbol{x})$

Fig. 2. The Heaviside function H_q defined by Eq. (23) and used in the qth extrapolation step. Cells shaded grey correspond to $H_q = 1$, and $H_q = 0$ otherwise. The interface Γ is shown by the thick black line.

within a band of cells that contains Γ, which keeps track of the minimum number of cells in a given direction between $\boldsymbol{x}$ and $\boldsymbol{x}_\Gamma$. This allows the Heaviside function $H_q(\boldsymbol{x})$ used in the qth extrapolation step to be succinctly defined as

$$H_q(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \mathcal{B}(\boldsymbol{x}) \le q, \\ 1 & \text{if } \mathcal{B}(\boldsymbol{x}) > q. \end{cases} \tag{23}$$

Fig. 2 shows $H_0(\boldsymbol{x})$, $H_1(\boldsymbol{x})$, $H_2(\boldsymbol{x})$, and $H_3(\boldsymbol{x})$ in the vicinity of Γ for a sample interface. Modifying the Hamiltonian to include a source term and utilizing this definition of $H_q(\boldsymbol{x})$, an mth order PDE-based extrapolation can be achieved by solving the $m+1$ equations

$$\frac{\partial f_q}{\partial \tau} + \big(1 - H_q(\boldsymbol{x})\big)(\nabla f_q \cdot \boldsymbol{n} - f_{q+1}) = 0 \quad \text{for } q = m, m-1, \ldots, 0, \tag{24}$$

where

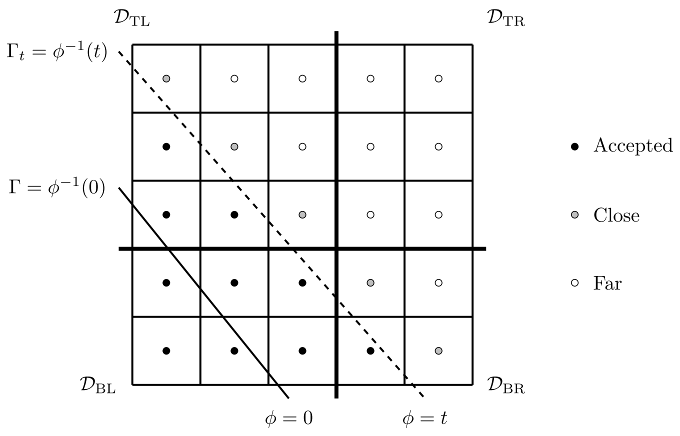
$$f_q = \nabla f_{q-1} \cdot \boldsymbol{n} \tag{25}$$

is the qth order differential of f projected onto the normal direction. Note that $f_{m+1} = 0$. The resulting f field will be such that $\partial^m f / \partial n^m$ is constant along $\boldsymbol{n}$ within Ω_{out} and $\mathcal{C}^m$ continuous across Γ.

Eq. (24) is discretized using second order upwind finite differences, where the upwind direction is the direction of the normal vector within each computational cell. The normals themselves are computed with second-order central differences from Eq. (1). A second-order Runge–Kutta (RK-2) scheme is used for temporal integration. Parallelization of the PDE approach is straightforward and only requires that the values of f_q be communicated at interprocessor boundaries between each substep of the RK-2 temporal integration and between each qth extrapolation of f_q for $q = m, m-1, \ldots, 0$.

3.2. FMM-based extrapolation

The fast marching extrapolation developed in the present work relies on a parallel FMM re-initialization of the signed distance level set function ϕ. The re-initialization procedure utilized herein is similar to the description of Herrmann [36], and



$\mathcal{D}_{\mathrm{TL}}$
$\mathcal{D}_{\mathrm{TR}}$
$\Gamma_t = \phi^{-1}(t)$
$\Gamma = \phi^{-1}(0)$
● Accepted
◉ Close
○ Far
$\mathcal{D}_{\mathrm{BL}}$
$\mathcal{D}_{\mathrm{BR}}$
$\phi = 0$
$\phi = t$

Fig. 3. Accepted (black circles), close (grey circles), and far (open circles) cells for organization of points for ϕ re-initialization. Thick black lines separate the computational domain into four subdomains $\mathcal{D}_{\mathrm{BL}}$, $\mathcal{D}_{\mathrm{BR}}$, $\mathcal{D}_{\mathrm{TL}}$, and $\mathcal{D}_{\mathrm{TR}}$ for each processor.

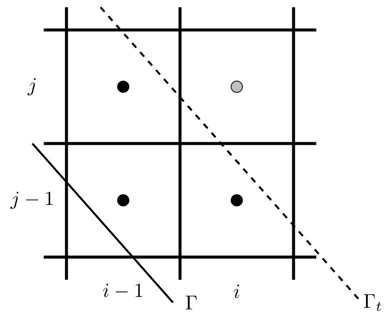


Fig. 4. Example 2D upwind ϕ calculation. The dashed line shows Γ, and the cell of consideration (i, j) is shaded in grey.

more in-depth discussions of the fast marching method can be found elsewhere [27,37–39]. The basis for ϕ re-initialization and subsequent f extrapolation is given here, while the details of their parallel implementation are provided in Appendix A and Appendix B, respectively.

3.2.1. Re-initialization of ϕ

The fast marching method can be used to efficiently solve the Eikonal equation

$$\|\nabla\phi\| = 1 \tag{26}$$

by ordered sweeping in the direction of normal characteristics, as opposed to the time-dependent nature of a Hamilton–Jacobi approach. The directionality of the FMM comes from an upwind finite difference discretization of $\|\nabla\phi\|$, where the location of the interface determines the upwind direction. Writing the interface $\Gamma = \{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$ as $\Gamma = \phi^{-1}(0)$, an iso-level of $\phi > 0$, denoted as Γ_t, can be written as $\Gamma_t = \phi^{-1}(t)$ for some $t > 0$, where t characterizes the progress of the algorithm. Re-initialization of ϕ amounts to sweeping the front Γ_t from the interface Γ to the rest of the computational domain, as illustrated in Fig. 3. Crucial to the FMM procedure are the three lists $\mathcal{A}$, $\mathcal{C}$, and $\mathcal{F}$, distinguishing each computational cell (i, j, k) as "accepted", "close", and "far", respectively. These lists are local to each processor and are of dynamic size $N_{\mathcal{A}}$, $N_{\mathcal{C}}$, and $N_{\mathcal{F}}$, respectively. Cells for which $\phi_{i,j,k}$ is set are included in $\mathcal{A}$, as shown in Fig. 3. Cells immediately adjacent to any cell in $\mathcal{A}$ but not included in $\mathcal{A}$ are included in $\mathcal{C}$, and all remaining cells are included in $\mathcal{F}$. Each cell in $\mathcal{C}$ is a candidate to be the closest cell to Γ_t, i.e., to have the minimum value of ϕ of all non-accepted cells.

Based on Godunov's method, upwind finite difference stencils are used to compute $\phi_{i,j,k}$ within each close cell in $\mathcal{C}$ by solving the equation

$$\max\big(D_x^-\phi_{i,j,k}, -D_x^+\phi_{i,j,k}, 0\big)^2 + \max\big(D_y^-\phi_{i,j,k}, -D_y^+\phi_{i,j,k}, 0\big)^2 + \max\big(D_z^-\phi_{i,j,k}, -D_z^+\phi_{i,j,k}, 0\big)^2 - 1 = 0, \tag{27}$$

where $D^{\pm}_{x,y,z}$ are first order upwind finite difference notations, defined as $D_x^+\phi_{i,j,k} = (\phi_{i+1,j,k} - \phi_{i,j,k})/\Delta x$ and $D_y^-\phi_{i,j,k} = (\phi_{i,j,k} - \phi_{i,j-1,k})/\Delta y$, for example. Prior to solving Eq. (27) within each cell in $\mathcal{C}$, ϕ is set to ∞ in any adjacent cell not included in $\mathcal{A}$ and in all far cells included in $\mathcal{F}$, i.e., Eq. (27) will select only accepted cells for the upwind calculation. Solving for the roots of Eq. (27) provides two solutions for $\phi_{i,j,k}$. In most cases, one root will be smaller than the neighboring values and the other larger. The causal nature of the FMM is maintained by taking the larger of the roots, and this gives the proper value of $\phi_{i,j,k}$. A two-dimensional example is shown in Fig. 4. The grey cell (i, j) is the close cell at which $\phi_{i,j}$ is being computed. Based on the location of Γ_t, the values of neighboring accepted cells $\phi_{i-1,j}$ and $\phi_{i,j-1}$ have already been computed and will be used in the calculation of $(\nabla\phi)_{i,j}$. In this case, Eq. (27) becomes

$$\left(D_x^-\phi_{i,j}\right)^2+\left(D_y^-\phi_{i,j}\right)^2-1=0, \tag{28}$$

and keeping the larger of the two roots provides the solution for $\phi_{i,j}$.

Once ϕ has been computed in all close cells, a heap sort is then performed on the list $\mathcal{C}$, which is a balanced binary-tree structure that orders $\mathcal{C}$ by increasing $|\phi|$. The heap sort will keep the cell with the minimum $|\phi|$ value as the $\mathcal{C}(1)$ entry [21,38] and leads to a theoretical $\mathcal{O}(N\log N)$ operation count for the serial FMM on a domain of N cells. After being sorted, cell $\mathcal{C}(1)$ is added to list $\mathcal{A}$ and any of its adjacent far neighbors are removed from $\mathcal{F}$ and added to $\mathcal{C}$. Eq. (27) is again solved for all cells in $\mathcal{C}$ by setting $\phi=\infty$ for all adjacent cells not included in $\mathcal{A}$ and in all far cells, the heap sort algorithm will reorder $\mathcal{C}$, and the process is repeated. The entire procedure is performed separately for regions corresponding to $\phi<0$ and $\phi\geq 0$. A detailed discussion of our parallel implementation of ϕ re-initialization is provided in Appendix A.

3.2.2. Redistribution of f

Having solved $\|\nabla\phi\|=1$ via Eq. (27), the re-initialized ϕ field can be used to solve the $m+1$ redistribution equations

$$\nabla f_q\cdot\nabla\phi=f_{q+1}\quad \text{for } q=m,m-1,\dots,0, \tag{29}$$

which are the static form of Eq. (24) implemented for PDE-based extrapolation. Note that the form of Eq. (29) does not account for the boundary cells that are designated by $H_q(\boldsymbol{x})=1$; in practice, cells with $H_{q,i,j,k}=1$ are skipped during f redistribution (see Appendix B for details). One iteration of Eq. (29) is approximated by

$$\begin{aligned}&S_x^-\left(D_x^-\phi_{i,j,k}\cdot D_x^- f_{q,i,j,k}\right)+S_x^+\left(D_x^+\phi_{i,j,k}\cdot D_x^+ f_{q,i,j,k}\right)\\&\quad+S_y^-\left(D_y^-\phi_{i,j,k}\cdot D_y^- f_{q,i,j,k}\right)+S_y^+\left(D_y^+\phi_{i,j,k}\cdot D_y^+ f_{q,i,j,k}\right)\\&\quad+S_z^-\left(D_z^-\phi_{i,j,k}\cdot D_z^- f_{q,i,j,k}\right)+S_z^+\left(D_z^+\phi_{i,j,k}\cdot D_z^+ f_{q,i,j,k}\right)=f_{q+1,i,j,k},\end{aligned} \tag{30}$$

where the switches $S_x^\pm$ are defined as

$$\begin{aligned}S_x^+&=\begin{cases}1 & \text{if } \max(D_x^-\phi_{i,j,k},-D_x^+\phi_{i,j,k},0)=-D_x^+\phi_{i,j,k},\\0 & \text{otherwise}\end{cases}\\S_x^-&=\begin{cases}1 & \text{if } \max(D_x^-\phi_{i,j,k},-D_x^+\phi_{i,j,k},0)=D_x^-\phi_{i,j,k},\\0 & \text{otherwise.}\end{cases}\end{aligned} \tag{31}$$

The switches $S_y^\pm$ and $S_z^\pm$ are defined accordingly, and these switches ensure that only the adjacent cells with a smaller $|\phi|$ value used in solving Eq. (27) are again used to solve the redistribution equation. In terms of the example in Fig. 4, Eq. (30) becomes

$$f_{q,i,j}\left(\frac{(\phi_{i,j}-\phi_{i-1,j})}{\Delta x^2}+\frac{(\phi_{i,j}-\phi_{i,j-1})}{\Delta y^2}\right)-\left(f_{q,i-1,j}\frac{(\phi_{i,j}-\phi_{i-1,j})}{\Delta x^2}+f_{q,i,j-1}\frac{(\phi_{i,j}-\phi_{i,j-1})}{\Delta y^2}\right)=f_{q+1,i,j}. \tag{32}$$

All values of ϕ are known, $f_{q,i-1,j}$ and $f_{q,i,j-1}$ have already been computed, and $f_{q+1,i,j}$ was computed in the previous $q+1$ extrapolation step, so $f_{q,i,j}$ is readily available from Eq. (32). A significant benefit of this approach is that if the upwind stencils used to re-initialize ϕ are stored, the computational cost of successive redistributions is minimal.

An important detail that warrants special attention arises when performing high order extrapolations: the FMM propagates information in the normal direction starting from Γ, while FMM data extrapolation propagates information in the normal direction starting from the region where $H_q=1$. Thus, extrapolations with $m>0$ require propagating information in the direction opposite the normal in a few cells near the interface (see Fig. 2). Numerically this means that within these few cells near Γ, the stencils for re-initializing ϕ are not equivalent to the stencils for extrapolating f_q. The way that we account for this and the details of the parallel implementation of f redistribution are provided in Appendix B.

4. Comparison between FMM and PDE extrapolation

In this section, the predicted rates of convergence of the extrapolations are verified, and the error level and computational cost of the PDE and FMM procedures are compared. Using the test case of Aslam [31], convergence over the entire domain as well as in a narrow band is considered. The two-dimensional computational domain is defined by $x\in[-\pi,+\pi]$ and $y\in[-\pi,+\pi]$, and the level set is given as

$$\phi=r-R, \tag{33}$$

where $r=\sqrt{x^2+y^2}$. The radius of the circle is set to be $R=2$. The scalar field $g(x,y)$ is set to be

$$g(x,y)=\cos(x)\sin(y). \tag{34}$$

The simple form of ϕ in Eq. (33) allows to solve analytically $\nabla f_q\cdot\boldsymbol{n}=f_{q+1}$ for $q=m,m-1,\dots,0$. The solutions for $m=0,1,2,3$ are shown in Fig. 5. For the PDE and FMM extrapolations, second order central differences are used to compute normal vectors from Eq. (1) based on the prescribed ϕ in Eq. (33).

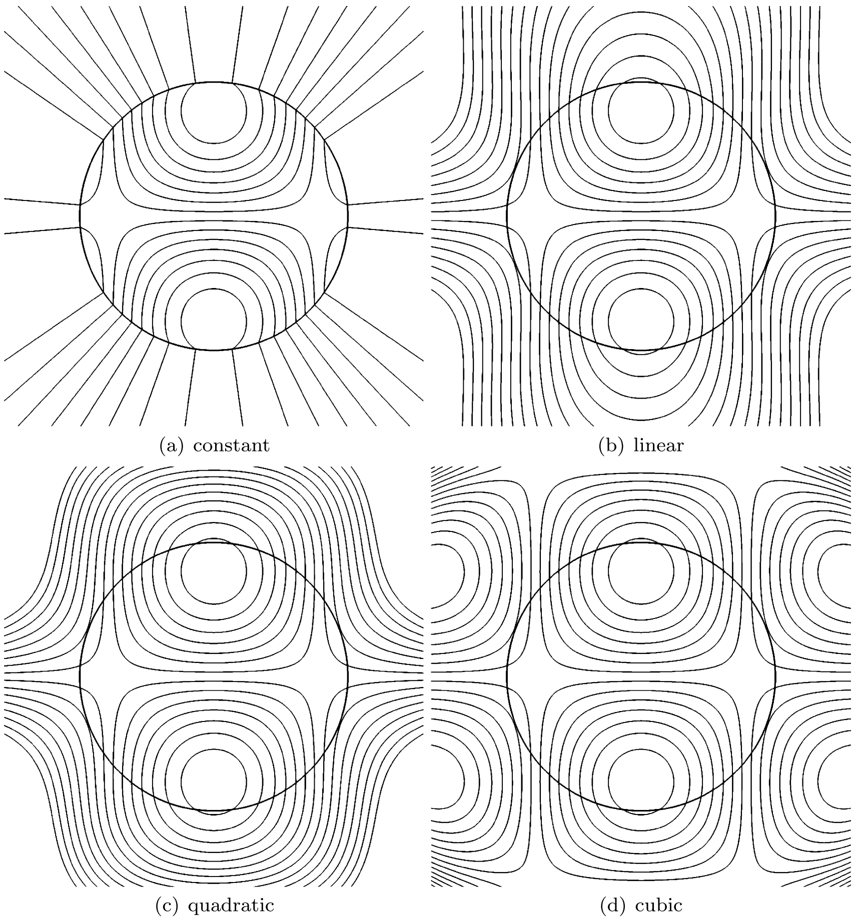


Fig. 5. Exact solution to the test case described by Eqs. (33) and (34). $\phi = 0$ (thick line), contours of f in increments of 0.127 (thin lines).

4.1. Global convergence

Table 1 gives the ratio of the FMM error L_{F}^{p} to the PDE error L_{P}^{p} for $p = 1$, 2, and ∞. The PDE approach proves to be slightly more accurate for all cases except for cubic, due to the first order nature of the upwind stencils used in the FMM. The rates of convergence under mesh refinement $\mathcal{R}_{\mathrm{F,M}}^{p}$ are also provided, observing a range of convergence near first order for both PDE and FMM. Overall, both FMM and PDE appear to be performing very similarly for this test case.

4.2. Narrow band convergence

In many applications of multidimensional extrapolation, it is only necessary to extend a given quantity into a narrow band containing the interface. Performing the same extrapolations into a region $5\Delta x$ from the interface yields a convergence rate $\mathcal{R}_{\mathrm{band}}$ that follows the predicted value in Eq. (11). As shown in Figs. 6(a) and 6(b), the first order nature of the FMM limits it to second order convergence, and the PDE is limited to third order convergence since it relies on second order accurate schemes. For the case of cubic extrapolation with $m = 3$, the PDE's $\mathcal{O}(\Delta x^3)$ error associated with the second order upwind discretization will limit the rate of convergence, despite the $\mathcal{O}(\Delta x^4)$ nature of the $m = 3$ extrapolation. This is seen in Fig. 6(a), which begins to show $\mathcal{R}_{\mathrm{band}} = 3$ for the cubic extrapolation on the 1600×1600 and 3200×3200 grids.

It is evident from comparison of the L^2 values in Figs. 6(a) and 6(b) that the accuracy of the PDE approach relative to FMM improves when only a narrow band is considered, as the first order errors associated with the FMM are most predominant in the cells immediately adjacent to the interface. This effect could be reduced by utilizing a high-order interpolation to initialize the fast marching process, but this is not investigated in this study. Confirmation that first order errors are what prevent third order convergence for quadratic FMM extrapolation is given in Fig. 6(c), which shows that the PDE behaves identically to the FMM when a first order upwind is used to discretize ∇f.

4.3. Cost comparison

Tests on meshes of 800^2 and smaller were performed on a single Intel Xeon dual 6-core X5670 CPU (12 threads), while the larger meshes were run on 8 CPUs (96 threads). For the PDE method, the pseudo-time step $\Delta\tau$ was set to $\Delta\tau = 0.3\Delta x$

Table 1
Error ratios and rates of convergence for FMM and PDE extrapolation from a circle.

Δx	$L^1_\mathrm{F}/L^1_\mathrm{P}$	$L^2_\mathrm{F}/L^2_\mathrm{P}$	$L^\infty_\mathrm{F}/L^\infty_\mathrm{P}$	$\mathcal{R}^1_\mathrm{F}$	$\mathcal{R}^1_\mathrm{P}$	$\mathcal{R}^2_\mathrm{F}$	$\mathcal{R}^2_\mathrm{P}$	$\mathcal{R}^\infty_\mathrm{F}$	$\mathcal{R}^\infty_\mathrm{P}$
$2\pi/100$	4.48	4.35	1.85						
$2\pi/200$	4.18	4.16	1.89	0.96	0.87	0.99	0.93	0.93	0.96
$2\pi/400$	4.77	4.24	1.64	1.07	1.26	1.08	1.11	1.05	0.85
$2\pi/800$	4.84	4.22	1.87	1.05	1.07	1.02	1.01	0.88	1.07
$m = 0$									
$2\pi/100$	1.64	1.69	1.47						
$2\pi/200$	1.61	1.66	1.16	1.10	1.07	1.09	1.06	1.00	0.66
$2\pi/400$	1.60	1.65	1.38	1.08	1.07	1.05	1.04	0.75	1.00
$2\pi/800$	1.61	1.66	1.17	1.06	1.06	1.06	1.07	1.07	0.83
$m = 1$									
$2\pi/100$	1.44	1.37	1.26						
$2\pi/200$	1.42	1.37	1.26	1.04	1.03	1.03	1.02	0.98	0.98
$2\pi/400$	1.44	1.40	1.43	1.07	1.08	1.07	1.10	1.07	1.25
$2\pi/800$	1.45	1.41	1.34	1.04	1.05	1.05	1.07	1.12	1.02
$m = 2$									
$2\pi/100$	0.95	0.94	0.92						
$2\pi/200$	0.86	0.83	0.77	1.42	1.27	1.45	1.27	1.46	1.21
$2\pi/400$	0.85	0.78	0.71	1.20	1.18	1.26	1.18	1.34	1.23
$2\pi/800$	0.85	0.77	0.66	1.12	1.11	1.16	1.14	1.37	1.26
$m = 3$									

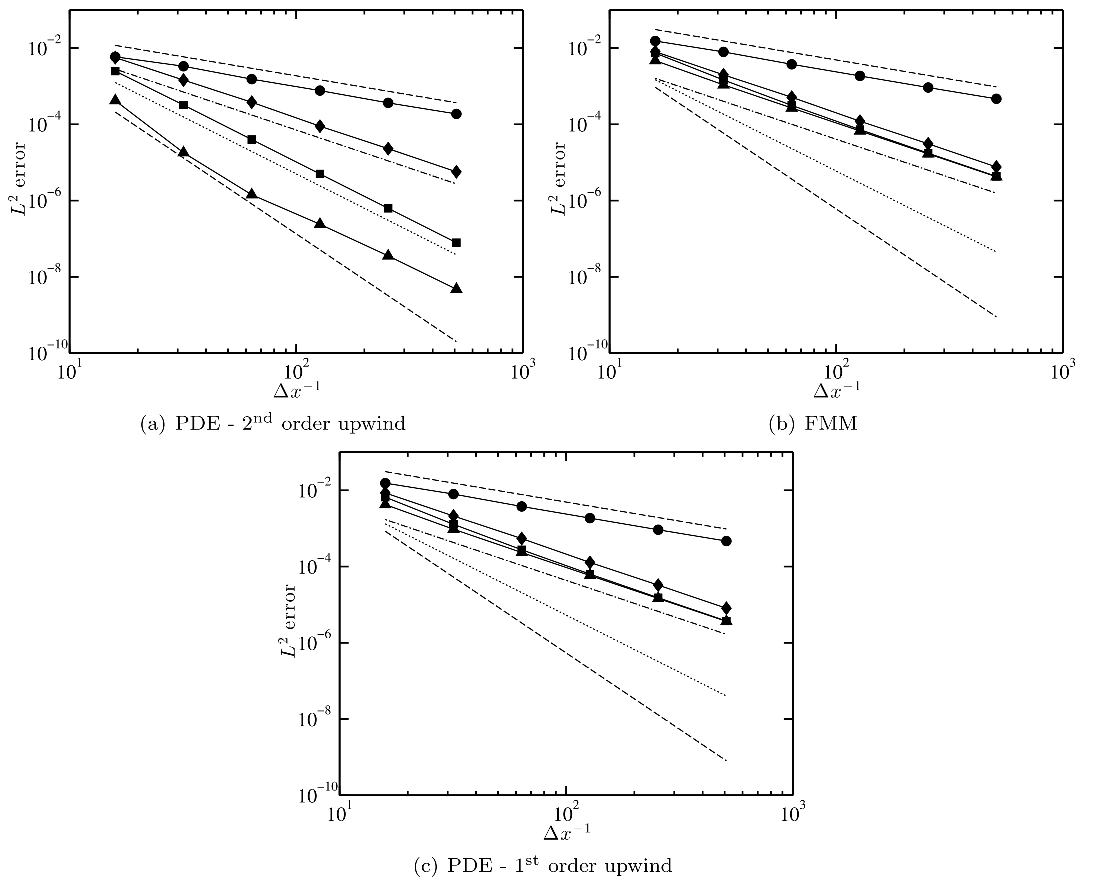


Fig. 6. Convergence in a band of points $5\Delta x$ from the interface. Constant (●), linear (◆), quadratic (■), cubic (▲), 1st order (upper dashed line), 2nd order (dash-dotted line), 3rd order (dotted line), 4th order (lower dashed line).

Table 2
Timing ratios for extrapolation to the full domain.

Δx	$T_{\text{PDE}}/T_{\text{FMM}}$			
	$m=0$	$m=1$	$m=2$	$m=3$
$2\pi/100$	10.09	15.58	17.58	16.61
$2\pi/200$	14.61	21.90	26.74	22.23
$2\pi/400$	40.04	36.24	59.96	35.25
$2\pi/800$	127.49	183.01	181.91	133.37

Table 3
Timing ratios for extrapolation into a band of $5\Delta x$.

Δx	$T_{\text{PDE}}/T_{\text{FMM}}$			
	$m=0$	$m=1$	$m=2$	$m=3$
$2\pi/100$	8.25	12.77	12.97	10.22
$2\pi/200$	6.21	7.97	11.05	9.96
$2\pi/400$	4.63	8.13	9.72	8.59
$2\pi/800$	5.83	5.65	11.09	8.94
$2\pi/1600$	9.98	15.21	17.56	14.03
$2\pi/3200$	18.65	33.34	33.51	18.27

Table 4
L^2 error ratios for extrapolation into a band of $5\Delta x$.

Δx	$L^2_{\text{FMM}}/L^2_{\text{PDE}}$			
	$m=0$	$m=1$	$m=2$	$m=3$
$2\pi/100$	2.524	1.3546	3.2772	11.9886
$2\pi/200$	2.8565	1.3947	4.9682	39.1061
$2\pi/400$	2.8723	1.3913	8.8461	144.8141
$2\pi/800$	2.5897	1.3797	16.524	358.4859
$2\pi/1600$	2.662	1.3623	32.095	684.3938
$2\pi/3200$	2.5489	1.3736	63.349	1110.746

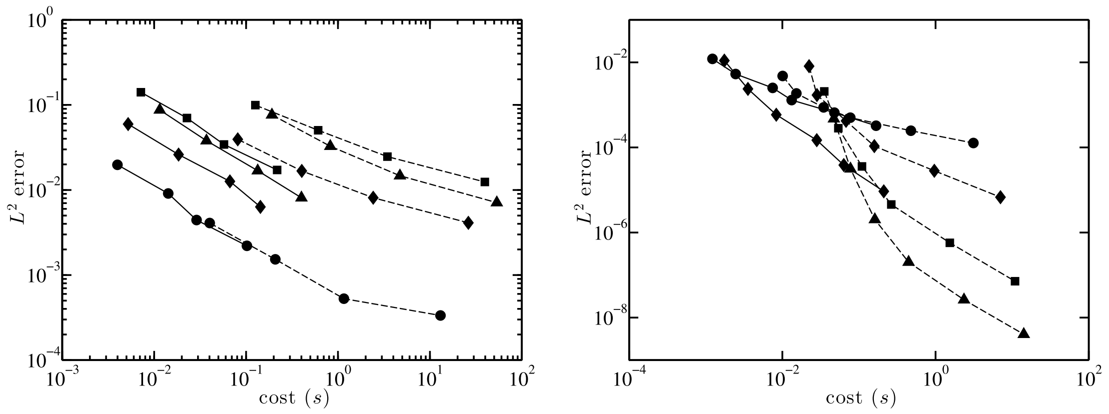
for stability reasons, and the convergence criterion was set to 10^{-9}. The ability of the FMM to extrapolate a data field in a single step as opposed to the iterative nature of the Hamilton–Jacobi equation leads to dramatic improvements in computational cost. Table 2 shows the ratio of PDE to FMM time spent in extrapolation to the whole domain, showing factors of nearly 200 speed-up for the FMM approach. If instead a narrow band is considered, the improvements shown in Table 3 are significantly more modest. However, even in the best case scenario for the PDE approach, a factor of 4 increase in time makes the FMM approach very attractive.

Fig. 7 shows the L^2 error as a function of computational cost for both FMM and PDE extrapolations. Fig. 7(a) for the full domain shows similar error levels for both methods, while the cost of the FMM is significantly reduced. Results in Fig. 7(b) for the narrow band show similar error levels and much lower FMM cost up to linear extrapolation, while the PDE drives errors much lower with quadratic and cubic extrapolation at the sake of increased computational cost. When compared to a first order upwind PDE approach, Fig. 7(c) shows that the FMM achieves the same level of error at a significantly reduced cost, as expected. These results are made more visible through direct comparison between Tables 3 and 4, which shows that, up to $m = 1$, error levels of the two methods are essentially the same, but the cost of the FMM is significantly less. We conclude that if error levels $\sim 10^{-5}$ are acceptable, then linear FMM extrapolation is clearly optimal, while lower error requirements benefit from quadratic and cubic PDE extrapolation.

It is worth mentioning that for a given problem, the mesh size is usually fixed, due to constraints like the resolution requirements of a fluid flow solver, for example. If second order narrow-band convergence is sufficient for a given application, then the FMM should always be used, as it provides the same accuracy as the PDE and is dramatically faster for any given mesh. If faster rates of convergence are required, then the PDE should be used, or second order upwind operators can be combined with second order interpolants to make the FMM formally second order accurate [21,34]. Note that this would increase the complexity of the parallel implementation.

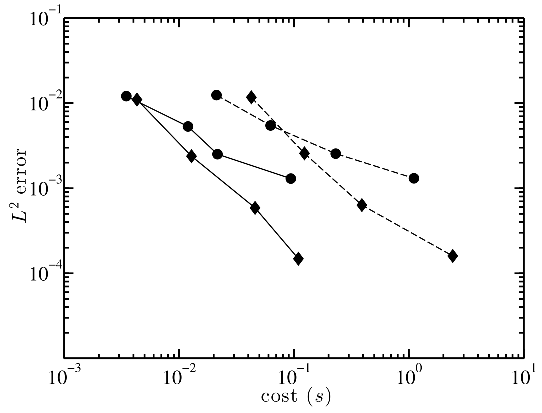
5. Extrapolation from an elliptical shape

In this section, we test a variation of Aslam's case [31] that reveals subtleties of multidimensional extrapolation and addresses questions regarding the directionality of the characteristics for a given problem. Convergence over the entire domain as well as in a narrow band is again considered. For this new test case, an alternative way of computing high order differentials is proposed that may reduce the error level for high order extrapolations.



(a) Extrapolation to full domain. FMM (solid line), PDE - 2nd order upwind (dashed line).

(b) Extrapolation to a band of $5\Delta x$. FMM (solid line), PDE - 2nd order upwind (dashed line)



(c) Extrapolation to a band of $5\Delta x$. FMM (solid line), PDE - 1st order upwind (dashed line)

Fig. 7. L^2 error as a function of computational cost of extrapolation. Constant (●), linear (◆), quadratic (■), cubic (▲).

5.1. Normal orientation

In this example the computational domain is the same as for the circle, and the level set is given as $\widetilde{\phi} = r - R$, where r is now defined as $r = \sqrt{(x/a)^2 + (y/b)^2}$. The radius is $R = 2$, with $a = 1.2$ and $b = 0.8$. The initial scalar field $g(x, y)$ is again provided by Eq. (34). It is important to realize that for this elliptic shape, $\widetilde{\phi}$ is not a true signed distance function. Indeed, a field of normals obtained from $\widetilde{\boldsymbol{n}} = \nabla\widetilde{\phi}/\|\nabla\widetilde{\phi}\|$ have the property $\nabla\widetilde{\boldsymbol{n}} \cdot \widetilde{\boldsymbol{n}} \neq 0$, which means the normals turn when going from one iso-level of $\widetilde{\phi}$ to the next. This is in contrast to the ϕ obtained by projecting a point $\boldsymbol{x}$ onto Γ such that $\|\boldsymbol{x} - \boldsymbol{x}_\Gamma\|$ in minimized, which is done via a Newton–Raphson method to a high degree of accuracy to obtain the analytical solutions for $m = 0, 1, 2, 3$ shown in Fig. 8. The difference between normals obtained from the true signed distance function ϕ and those obtained from the elliptic $\widetilde{\phi}$ is illustrated in Fig. 9. Extrapolations based on $\widetilde{\phi}$ do not converge, while extrapolations based on ϕ obtained from the projection method show the same behavior as the circle test case. This is evidenced by the global convergence in Table 5 that mimics the global results for the circle and the narrow-band convergence shown in Fig. 10 that follows Eq. (11).

5.2. High order differential formulation

The turning characteristics of the ellipse for which $\nabla\widetilde{\boldsymbol{n}} \cdot \widetilde{\boldsymbol{n}} \neq 0$ raise to question the computation of high order differentials that are the source terms for high order extrapolations. For $q \geq 2$ in Eq. (25), there is flexibility in how f_q is computed. For example, f_2 could be computed as

$$f_2 = \nabla f_1 \cdot \boldsymbol{n} = \nabla(\nabla f \cdot \boldsymbol{n}) \cdot \boldsymbol{n}, \tag{35}$$

which is the approach of Aslam [31]. An alternative is

$$f_2 = \boldsymbol{n}^{\mathsf{T}} \cdot \nabla(\nabla f) \cdot \boldsymbol{n}, \tag{36}$$

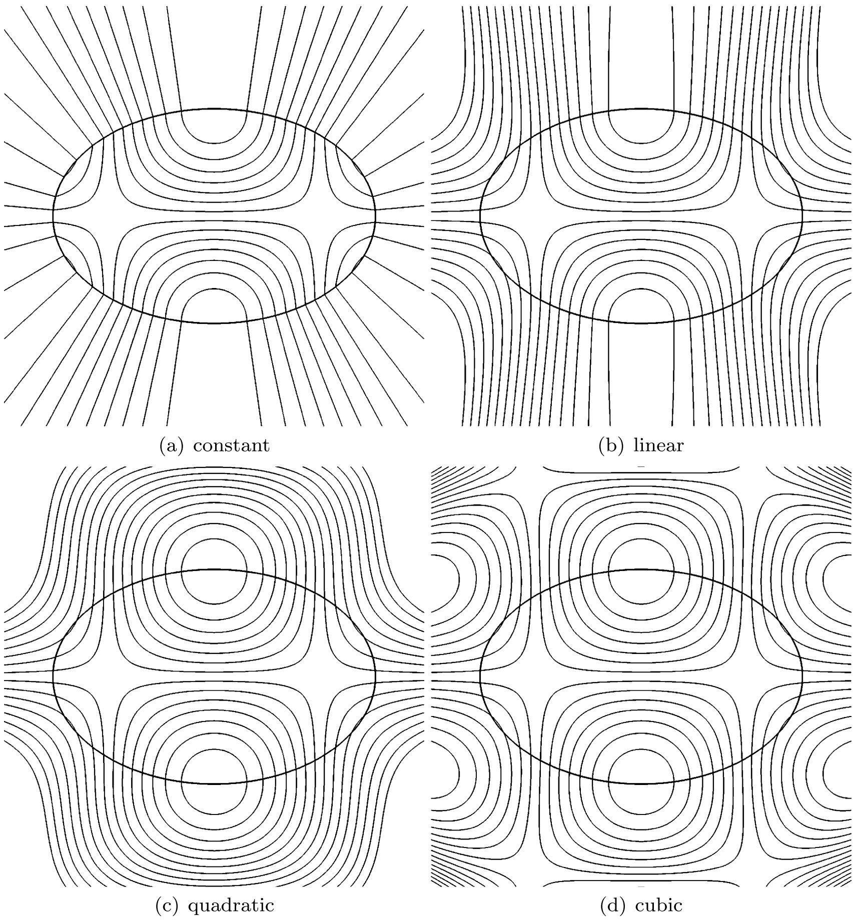


Fig. 8. Exact solution to the elliptic test case. $\phi = 0$ (thick line), contours of f in increments of 0.127 (thin lines).

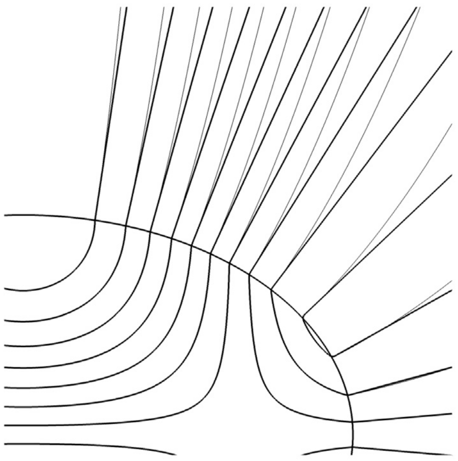


Fig. 9. Normals obtained from $\boldsymbol{n} = \nabla\phi / \|\nabla\phi\|$ (thick lines) and from $\widetilde{\boldsymbol{n}} = \nabla\widetilde{\phi} / \|\nabla\widetilde{\phi}\|$ (thin lines).

where $\nabla(\nabla f)$ is the Hessian of f. Similarly, two possible ways of writing f_3 are

$$f_3 = \nabla f_2 \cdot \boldsymbol{n} = \nabla\big(\nabla(\nabla f \cdot \boldsymbol{n}) \cdot \boldsymbol{n}\big) \cdot \boldsymbol{n} \tag{37}$$

and

$$f_3 = \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k} n_i n_j n_k, \tag{38}$$

Table 5
Error ratios and rates of convergence for FMM and PDE extrapolation. Normals obtained from the analytic ϕ based on the projection method.

Δx	L^1_F/L^1_P	L^2_F/L^2_P	L^∞_F/L^∞_P	$\mathcal{R}^1_F$	$\mathcal{R}^1_P$	$\mathcal{R}^2_F$	$\mathcal{R}^2_P$	$\mathcal{R}^\infty_F$	$\mathcal{R}^\infty_P$
$2\pi/100$	5.07	4.83	2.68						
$2\pi/200$	6.50	5.95	3.04	1.07	1.43	1.12	1.42	1.08	1.26
$2\pi/400$	8.99	8.39	2.25	1.06	1.52	1.04	1.54	1.11	0.68
$2\pi/800$	7.89	6.59	1.98	0.98	0.79	1.01	0.66	1.02	0.84
$m=0$									
$2\pi/100$	1.65	1.51	1.34						
$2\pi/200$	1.68	1.56	1.13	1.15	1.18	1.19	1.23	1.47	1.23
$2\pi/400$	1.68	1.56	1.14	1.06	1.06	1.05	1.05	0.85	0.86
$2\pi/800$	1.66	1.54	1.16	1.00	0.99	1.00	0.98	0.85	0.87
$m=1$									
$2\pi/100$	1.54	1.42	1.31						
$2\pi/200$	1.50	1.39	1.30	1.02	0.98	1.01	0.98	0.98	0.98
$2\pi/400$	1.49	1.39	1.28	1.04	1.04	1.03	1.03	0.98	0.96
$2\pi/800$	1.49	1.38	1.28	1.00	0.99	0.99	0.99	0.99	0.99
$m=2$									
$2\pi/100$	1.22	1.14	0.89						
$2\pi/200$	1.26	1.16	0.86	1.18	1.23	1.20	1.23	1.31	1.27
$2\pi/400$	1.26	1.15	0.80	1.15	1.15	1.17	1.15	1.20	1.09
$2\pi/800$	1.26	1.14	0.84	1.05	1.06	1.06	1.05	0.99	1.06
$m=3$									

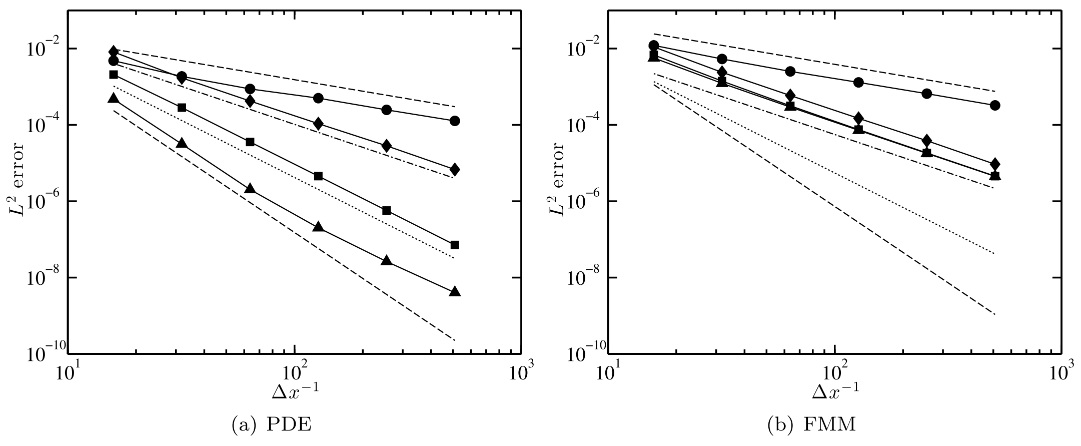


Fig. 10. Convergence in a band of points $5\Delta x$ from the interface. Normals obtained from the analytic ϕ based on the projection method. Constant (●), linear (◆), quadratic (■), cubic (▲), 1st order (upper dashed line), 2nd order (dash-dotted line), 3rd order (dotted line), 4th order (lower dashed line).

where Einstein's notation is used in Eq. (38). Upon expanding Eqs. (35) and (37) it is clear that they are only equal to their respective counterparts, Eqs. (36) and (38), if $\nabla \boldsymbol{n} \cdot \boldsymbol{n} = 0$. In general, f_q can be obtained by computing a qth order tensor that is the exact differential of f and projecting onto the normal q times. Consequently, Eqs. (36) and (38) do not involve differentiating $\boldsymbol{n}$, which may be desirable when $\boldsymbol{n}$ is computed numerically and contains some level of error.

Many realistic level set applications rely on normals derived from a ϕ field obtained from FMM re-initialization [5,7,9,15], and it is of interest to test convergence when using such normals. Fig. 11 shows that extrapolation using an FMM-based ϕ does not alter the observed convergence as compared to when the exact ϕ obtained from the projection method is used, so long as f_2 and f_3 for quadratic and cubic extrapolation are computed via Eqs. (36) and (38). However, repeated differentiation of $\boldsymbol{n}$ to form the differential as in Eq. (37) does affect the convergence at the cubic level for PDE extrapolation, as shown in Fig. 12. We conclude that the generalized tensor-projection method proposed herein for computing high order differentials is beneficial.

6. Conclusion

A fast marching framework for multidimensional extrapolation has been provided. Second order FMM extrapolation is achieved through the sequential solution of non-homogeneous static Hamilton–Jacobi equations and compared to solutions from the pre-existing PDE-based framework [31]. Two numerical examples based on prescribed signed distance level set

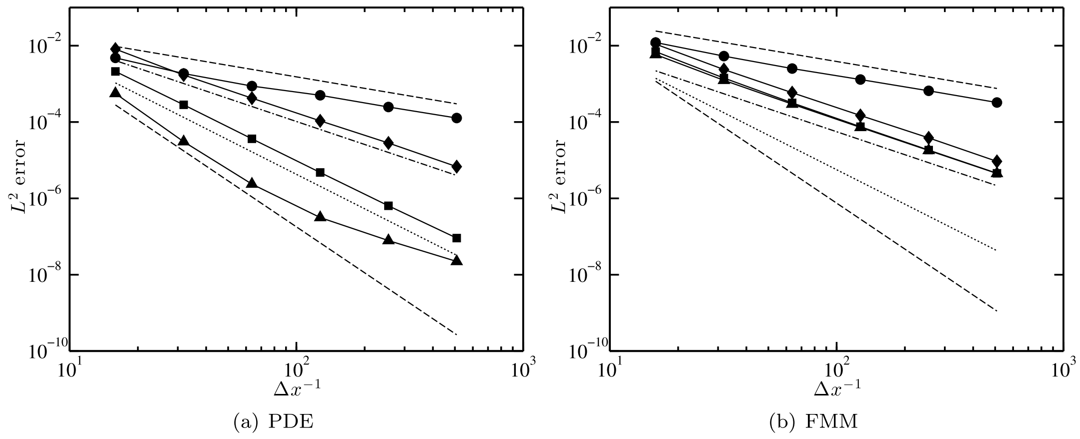


Fig. 11. Convergence in a band of points 5Δx from the interface. Normals obtained from a fast marching solution for ϕ, differentials computed via Eqs. (36) and (38). Constant (●), linear (♦), quadratic (■), cubic (▲), 1st order (upper dashed line), 2nd order (dash-dotted line), 3rd order (dotted line), 4th order (lower dashed line).

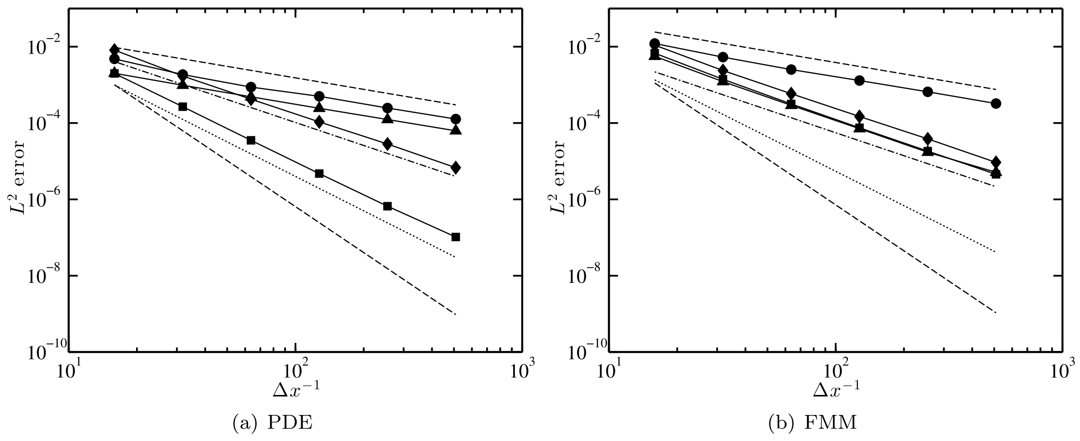


Fig. 12. Convergence in a band of points 5Δx from the interface. Normals obtained from a fast marching solution for ϕ, differentials computed via Eqs. (35) and (37). Constant (●), linear (♦), quadratic (■), cubic (▲), 1st order (upper dashed line), 2nd order (dash-dotted line), 3rd order (dotted line), 4th order (lower dashed line).

fields are used to compare the PDE and FMM approaches, the first being extrapolation from a circle used by Aslam [31]. Our predicted narrow-band convergence is verified on meshes ranging from 100^2 to 3200^2 points, demonstrating that the rate of convergence is controlled by the minimum between the order of the extrapolation itself and the order of accuracy of the spatial discretization. Timing analysis shows that the FMM approach offers dramatic computational cost improvements compared to the iterative nature of PDE extrapolation. If second order convergence in a narrow band is sufficient for a given application, then the FMM should always be used, as it provides the same accuracy as the PDE at a significantly reduced cost. Higher rates of convergence will require the use of the PDE method or a modified FMM that has higher spatial accuracy. Next, an ellipse is presented, and results suggest that computing high order differentials through a more general tensor-projection method is preferred to the method of Aslam, as it eliminates the need to repeatedly differentiate the normal vector field. Finally, appendices are provided that give a detailed parallel implementation of the FMM-based extrapolation developed in the present work.

Appendix A. Parallel implementation of ϕ re-initialization

To keep the implementation general, we denote a computational cell as $\mathcal{X}$. To initialize the points in the domain prior to the FMM re-initialization process, each processor searches through its computational subdomain $\mathcal{D}$ and tags all cells $\mathcal{X}$ as either close or far, as done in Algorithm 1 in the plus direction, i.e., in regions with $\phi \geq 0$. Once cells adjacent to the interface are added to $\mathcal{C}$ and all others added to $\mathcal{F}$, the FMM process can begin. The serial algorithm for $\phi \geq 0$ performed by each processor on its local computational subdomain $\mathcal{D}$ is presented in Algorithm 2. This serial algorithm relies on Algorithm 3, which determines the orientation of $\mathcal{X}$ with respect to its accepted neighbors (or Γ itself, for the case of initialization when $\mathcal{A}$ is empty). Algorithm 3 then calls Algorithm 4, which computes the minimum $\phi(\mathcal{X})$ based on all possible upwind configurations.

Algorithm 1 Procedure to initialize the close and far lists in subdomain $\mathcal{D}$ where $\phi \geq 0$.

```
function FMM_CLOSE_FAR($\phi, \mathcal{C}, \mathcal{F}, N_\mathcal{C}, N_\mathcal{F}$)        ▷ in: $\phi$
                                                                ▷ out: $\phi$, $\mathcal{C}$, $\mathcal{F}$, $N_\mathcal{C}$, $N_\mathcal{F}$
    $N_\mathcal{C} \leftarrow 0$                                    ▷ number of close cells
    $N_\mathcal{F} \leftarrow 0$                                    ▷ number of far cells
    for all $\mathcal{X} \in \mathcal{D} : \phi(\mathcal{X}) \geq 0$ do          ▷ loop over processor subdomain
        if $\mathcal{X}$ is adjacent to $\Gamma$ then                ▷ determine if cell includes $\Gamma$
            $N_\mathcal{C} \leftarrow N_\mathcal{C} + 1$
            $\mathcal{C}(N_\mathcal{C}) \leftarrow \mathcal{X}$                       ▷ if so, add to list of close cells
        else
            $N_\mathcal{F} \leftarrow N_\mathcal{F} + 1$
            $\mathcal{F}(N_\mathcal{F}) \leftarrow \mathcal{X}$                       ▷ if not, add to list of far cells
        end if
    end for
    initialize $\mathcal{A}$ to an empty list
    call FMM_GET_NEIGHBORS($\phi, \mathcal{A}, \mathcal{C}$)       ▷ compute $\phi(\mathcal{X})$ $\forall \mathcal{X} \in \mathcal{C}$ based on $\Gamma$
end function
```

Algorithm 2 Serial FMM re-initialization of ϕ for a processor on subdomain $\mathcal{D}$.

```
function SERIAL_FMM_REINIT($\phi, \mathcal{A}, N_\mathcal{A}$)                  ▷ in: $\phi$
                                                                ▷ out: $\phi$, $\mathcal{A}$, $N_\mathcal{A}$
    $N_\mathcal{A} \leftarrow 0$                                    ▷ number of accepted cells
    call FMM_CLOSE_FAR($\phi, \mathcal{C}, \mathcal{F}, N_\mathcal{C}, N_\mathcal{F}$)     ▷ initialize the close and far lists
    while $N_\mathcal{C} > 0$ do                                ▷ process all remaining close cells
        call the heap sort on $\mathcal{C}$                        ▷ order close cells by increasing $\phi$
        $\mathcal{X} \leftarrow \mathcal{C}(1)$                          ▷ obtain the cell that is closest to $\Gamma_t$
        $N_\mathcal{A} \leftarrow N_\mathcal{A} + 1$
        $\mathcal{A}(N_\mathcal{A}) \leftarrow \mathcal{X}$                       ▷ add it to the accepted list
        delete $\mathcal{C}(1)$                                 ▷ remove it from the close list
        $N_\mathcal{C} \leftarrow N_\mathcal{C} - 1$
        for all $\mathcal{X}'$ adjacent to $\mathcal{X}$ do          ▷ loop over cells adjacent to newly accepted cell
            if $\mathcal{X}' \in \mathcal{F}$ then                     ▷ is adjacent cell currently tagged as far?
                $N_\mathcal{C} \leftarrow N_\mathcal{C} + 1$
                $\mathcal{C}(N_\mathcal{C}) \leftarrow \mathcal{X}'$                  ▷ if so, add to close list
                delete $\mathcal{F}(N_\mathcal{F})$
                $N_\mathcal{F} \leftarrow N_\mathcal{F} - 1$                    ▷ remove from far list
            end if
        end for
        call FMM_GET_NEIGHBORS($\phi, \mathcal{A}, \mathcal{C}$)   ▷ compute $\phi(\mathcal{X})$ $\forall \mathcal{X} \in \mathcal{C}$ based on updated $\mathcal{A}$
    end while
end function
```

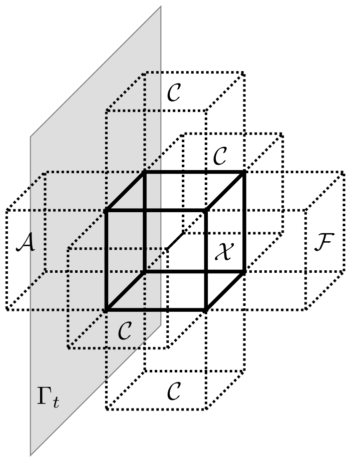
Algorithm 3 Procedure to compute ϕ.

```
function FMM_GET_NEIGHBORS($\phi, \mathcal{A}, \mathcal{C}$)                  ▷ in: $\phi$, $\mathcal{A}$, $\mathcal{C}$
                                                                ▷ out: $\phi$
    for all $\mathcal{X}$ included in $\mathcal{C}$ do                ▷ loop through list of close cells
        $N_{stc} \leftarrow 0$                                 ▷ number of possible neighboring cells
        for all $\mathcal{X}'$ adjacent to $\mathcal{X}$ do          ▷ loop over cells adjacent to cell of consideration
            if $\mathcal{X}' \in \mathcal{A}$ then                     ▷ is neighbor an accepted cell?
                $N_{stc} \leftarrow N_{stc} + 1$
                $\phi_{stc}(N_{stc}) \leftarrow \phi(\mathcal{X}')$               ▷ if so, store the $\phi$ of that neighbor
            else if $\mathcal{X}' \in \Gamma$ then            ▷ is neighbor the interface itself? (initialization step)
                $N_{stc} \leftarrow N_{stc} + 1$
                $\phi_{stc}(N_{stc}) \leftarrow 0$                    ▷ if so, store the $\phi$ of that neighbor as 0
            end if
        end for
        call FMM_UPWIND_CALC($\phi, \mathcal{S}, N_{stc}, \phi_{stc}$)   ▷ compute $\phi(\mathcal{X})$ based on $N_{stc}$ and $\phi_{stc}$
    end for
end function
```

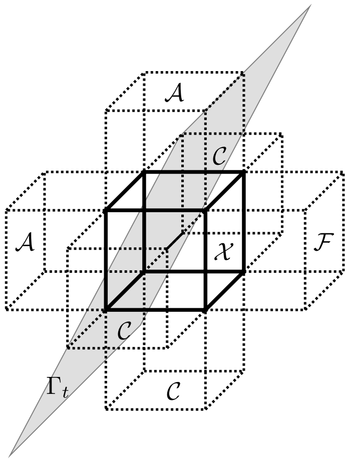
Algorithm 4 tests all possible upwind stencils based on N_{stc} accepted neighbors, followed by the subsequent selection of ϕ based on the stencil that provides the minimum ϕ value. This is because when interface topology is complex, cells can lie between multiple fronts. This becomes rather detailed in 3D, since as many as 6 adjacent neighbors are possible. Ultimately, the computation can be broken into 1D, 2D, and 3D calculations, as shown in Fig. 13. As in Fig. 13(a), a 1D calculation simply computes $\phi(\mathcal{X})$ by adding the cell size Δ of the computational mesh to the ϕ value of the single adjacent accepted cell. A 2D calculation amounts to computing the normal distance from $\mathcal{X}$ to the linear reconstruction of Γ_t by 2 in-plane adjacent accepted cells, as shown in Fig. 13(b). The third calculation involves computing the normal distance from $\mathcal{X}$ to the planar reconstruction of Γ_t from 3 adjacent accepted cells, as shown in Fig. 13(c).

Algorithm 4 Procedure to determine the FMM upwind stencil and resulting ϕ.

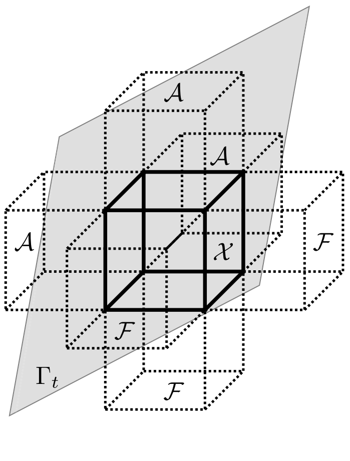
```
function FMM_UPWIND_CALC($\phi, \mathcal{S}, N_{stc}, \phi_{stc}$)        ▷ in: $N_{stc}, \phi_{stc}$
                                                                          ▷ out: $\phi, \mathcal{S}$
    if $N_{stc} == 0$ then
        $\phi(\mathcal{X})$ already set
    else if $N_{stc} == 1$ then
        $\phi(\mathcal{X}) \leftarrow \phi_{stc}(1) + \Delta$             ▷ 1D calculation
    else if $N_{stc} == 2$ then
        if neighbors lie on opposite sides of $\mathcal{X}$ then
            $\phi(\mathcal{X}) \leftarrow \min(\phi_{stc}(1) + \Delta, \phi_{stc}(2) + \Delta)$   ▷ minimum of 2 1D calculations
        else
            $\phi(\mathcal{X}) \leftarrow$ normal distance to linear reconstruction of $\Gamma_t$ from $\phi_{stc}(1:2)$   ▷ 2D calculation
        end if
    else if $N_{stc} == 3$ then
        if any two neighbors lie on opposite sides of $\mathcal{X}$ then
            compute all possible linear recons. of $\Gamma_t$ from pairs of neighbors
            $\phi(\mathcal{X}) \leftarrow$ minimum normal distance to possible $\Gamma_t$'s   ▷ 2D calculations
        else
            $\phi(\mathcal{X}) \leftarrow$ normal distance to planar recons. of $\Gamma_t$ from $\phi_{stc}(1:3)$   ▷ 3D calculation
        end if
    else if $N_{stc} == 4$ then
        if both sets of neighbors lie on opposite sides of $\mathcal{X}$ then
            compute all possible linear recons. of $\Gamma_t$ from pairs of neighbors
            $\phi(\mathcal{X}) \leftarrow$ minimum normal distance to possible $\Gamma_t$'s   ▷ 2D calculations
        else
            compute all possible planar recons. of $\Gamma_t$ from triads of neighbors
            $\phi(\mathcal{X}) \leftarrow$ minimum normal distance to possible $\Gamma_t$'s   ▷ 3D calculations
        end if
    else if $N_{stc} == 5$ then
        compute 4 possible planar recons. of $\Gamma_t$ from triads of neighbors
        $\phi(\mathcal{X}) \leftarrow$ minimum normal distance to possible $\Gamma_t$'s   ▷ 3D calculations
    else if $N_{stc} == 6$ then
        compute 8 possible planar recons. of $\Gamma_t$ from triads of neighbors
        $\phi(\mathcal{X}) \leftarrow$ minimum normal distance to possible $\Gamma_t$'s   ▷ 3D calculations
    end if
    $\mathcal{S}(\mathcal{X}) \leftarrow$ stencil for $N_{stc}$ neighbors used   ▷ remember stencil used for upwind (see Appendix B)
end function
```



(a) Stencil with 1 accepted neighbor to the left, leading to a 1D upwind calculation.



(b) Stencil with 2 accepted neighbors, to the left and above, leading to a linear reconstruction of Γ_t.



(c) Stencil with 3 accepted neighbors, to the left, above, and behind, leading to a planar reconstruction of Γ_t.

Fig. 13. Examples of the 1D, 2D, and 3D calculations involved in the upwind FMM calculation of ϕ. Cell $\mathcal{X}$ where ϕ is being computed is indicated by dark lines, and the grey plane indicates Γ_t. Adjacent cells are shown as $\mathcal{A}$ (accepted), $\mathcal{C}$ (close), or $\mathcal{F}$ (far).

The intricacy of the parallel FMM is that when a computational cell $\mathcal{X}$ is determined to have the minimum local ϕ of all candidate cells, i.e., $\mathcal{X} = \mathcal{C}(1)$, this does not mean that $\mathcal{X}$ has the minimum ϕ of all candidate cells globally. Consequently, processors are required to communicate accepted cells at interprocessor boundaries to the ghost cells of neighboring processors. If a received ϕ value is smaller than a previously accepted ϕ, then a rollback step is required to ensure that the final global list of accepted cells increases monotonically. Utilizing all algorithms required for the serial FMM along with the parallel rollback step in Algorithm 5, the parallel FMM procedure is provided in Algorithm 6.

Algorithm 5 Parallel rollback of cells in $\mathcal{A}$.

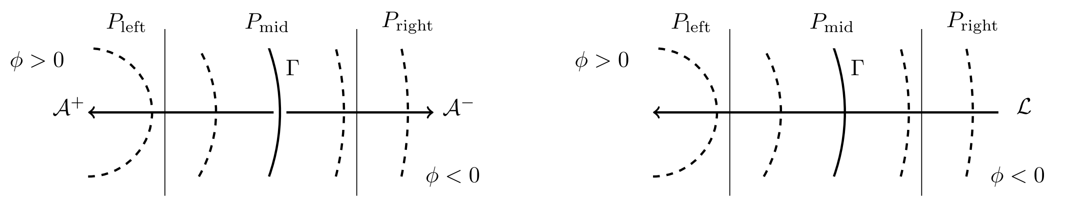
```
function PARALLEL_ROLLBACK($\phi, \mathcal{A}, \mathcal{C}, \mathcal{F}, N_{\mathcal{A}}, N_{\mathcal{C}}, N_{\mathcal{F}}$)    ▷ in: $\phi$, $\mathcal{A}$, $\mathcal{C}$, $\mathcal{F}$, $N_{\mathcal{A}}$, $N_{\mathcal{C}}$, $N_{\mathcal{F}}$
                                                        ▷ out: $\phi$, $\mathcal{A}$, $\mathcal{C}$, $\mathcal{F}$, $N_{\mathcal{A}}$, $N_{\mathcal{C}}$, $N_{\mathcal{F}}$
  while messages are available for receipt do
    receive $\mathcal{X}$ and $\phi_{\text{nbrs}}(\mathcal{X})$    ▷ receive all $\phi$ values from neighbor procs
    if $\phi_{\text{nbrs}}(\mathcal{X}) < \phi(\mathcal{X})$ then    ▷ if received value is smaller than local value, might need to roll back
      while $N_{\mathcal{A}} > 0$ do    ▷ if less than currently accepted value, roll back
        $\mathcal{X}' \leftarrow \mathcal{A}(N_{\mathcal{A}})$    ▷ access most recently accepted cell
        if $\phi(\mathcal{X}') < \phi_{\text{nbrs}}(\mathcal{X})$ then    ▷ is most recently accepted value smaller than received value?
          if $\mathcal{X}'$ is not a ghost cell, exit    ▷ if so, exit loop and check for another message
        end if
        delete $\mathcal{A}(N_{\mathcal{A}})$    ▷ if not, remove $\mathcal{X}'$ from $\mathcal{A}$
        $N_{\mathcal{A}} \leftarrow N_{\mathcal{A}} - 1$
        $N_{\mathcal{C}} \leftarrow N_{\mathcal{C}} + 1$
        $\mathcal{C}(N_{\mathcal{C}}) \leftarrow \mathcal{X}'$    ▷ and add it to $\mathcal{C}$
        call the heap sort on $\mathcal{C}$    ▷ re-sort the heap
      end while
      if $\mathcal{X} \in \mathcal{C}$ then    ▷ is cell in question already tagged as close?
        $\phi(\mathcal{X}) \leftarrow \phi_{\text{nbrs}}(\mathcal{X})$    ▷ if so, overwrite local value with received value
        call the heap sort on $\mathcal{C}$    ▷ re-sort the heap
      else if $\mathcal{X} \in \mathcal{F}$ then    ▷ is cell in question currently tagged as far?
        $\phi(\mathcal{X}) \leftarrow \phi_{\text{nbrs}}(\mathcal{X})$    ▷ if so, overwrite the local value
        delete $\mathcal{F}(N_{\mathcal{F}})$    ▷ remove it from the far list
        $N_{\mathcal{F}} \leftarrow N_{\mathcal{F}} - 1$
        $N_{\mathcal{C}} \leftarrow N_{\mathcal{C}} + 1$
        $\mathcal{C}(N_{\mathcal{C}}) \leftarrow \mathcal{X}$    ▷ add it to the close list
        call the heap sort on $\mathcal{C}$    ▷ re-sort the heap
      end if
    end if
  end while
end function
```

Algorithm 6 Parallel FMM re-initialization of ϕ.

```
function PARALLEL_FMM_REINIT($\phi, \mathcal{A}, N_{\mathcal{A}}$)    ▷ in: $\phi$
                                                        ▷ out: $\phi$, $\mathcal{A}$, $N_{\mathcal{A}}$
  $N_{\mathcal{A}} \leftarrow 0$    ▷ number of accepted cells
  call FMM_CLOSE_FAR($\phi, \mathcal{C}, \mathcal{F}, N_{\mathcal{C}}, N_{\mathcal{F}}$)    ▷ initialize the close and far lists
  $N_{\mathcal{C}}^{\text{all}} \leftarrow$ parallel sum of $N_{\mathcal{C}}$    ▷ sum close cells over all processors
  while $N_{\mathcal{C}}^{\text{all}} > 0$ do    ▷ begin the global loop
    while $N_{\mathcal{C}} > 0$ do    ▷ begin the local loop
      if message(s) is waiting for receipt then    ▷ have neighbors sent messages?
        call PARALLEL_ROLLBACK($\phi, \mathcal{A}, \mathcal{C}, \mathcal{F}, N_{\mathcal{A}}, N_{\mathcal{C}}, N_{\mathcal{F}}$)    ▷ if so, modify lists if necessary
      end if
      call the heap sort on $\mathcal{C}$    ▷ order close cells by increasing $\phi$
      $\mathcal{X} \leftarrow \mathcal{C}(1)$    ▷ obtain the cell that is closest to $\Gamma_t$
      $N_{\mathcal{A}} \leftarrow N_{\mathcal{A}} + 1$
      $\mathcal{A}(N_{\mathcal{A}}) \leftarrow \mathcal{X}$    ▷ add it to the accepted list
      delete $\mathcal{C}(1)$    ▷ remove it from close list
      $N_{\mathcal{C}} \leftarrow N_{\mathcal{C}} - 1$
      if $\mathcal{X}$ is at an interprocessor boundary then
        send $\phi(\mathcal{X})$ to buffer    ▷ communicate newly accepted cell to neighbor ghost cells
      end if
      for all $\mathcal{X}'$ adjacent to $\mathcal{X}$ do    ▷ loop over cells adjacent to newly accepted cell
        if $\mathcal{X}' \in \mathcal{F}$ then    ▷ is adjacent cell currently tagged as far?
          $N_{\mathcal{C}} \leftarrow N_{\mathcal{C}} + 1$
          $\mathcal{C}(N_{\mathcal{C}}) \leftarrow \mathcal{X}'$    ▷ if so, add to close list
          delete $\mathcal{F}(N_{\mathcal{F}})$
          $N_{\mathcal{F}} \leftarrow N_{\mathcal{F}} - 1$    ▷ remove from far list
        end if
      end for
      call FMM_GET_NEIGHBORS($\phi, \mathcal{A}, \mathcal{C}$)    ▷ compute $\phi(\mathcal{X})\ \forall \mathcal{X} \in \mathcal{C}$ based on updated $\mathcal{A}$
    end while
    $N_{\mathcal{C}}^{\text{all}} \leftarrow$ parallel sum of $N_{\mathcal{C}}$    ▷ sum close cells over all processors
  end while
end function
```

Appendix B. Parallel implementation of f redistribution

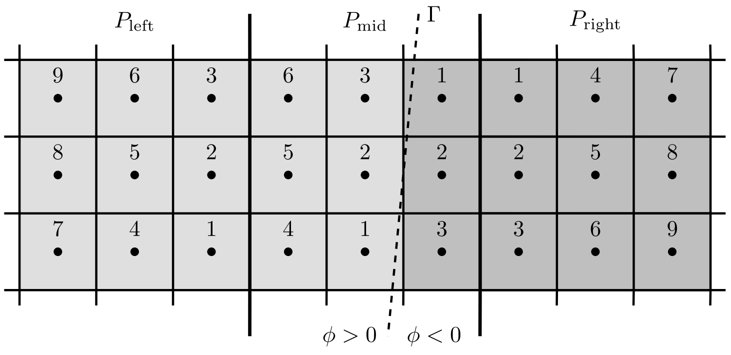
In performing high order extrapolations, $H_q(\boldsymbol{x}) = 1$ defines the region where f_q can be computed, as explained in Section 3.1. Thus, discretely populating Ω_{out} through an mth order extrapolation requires populating cells that lie on both sides



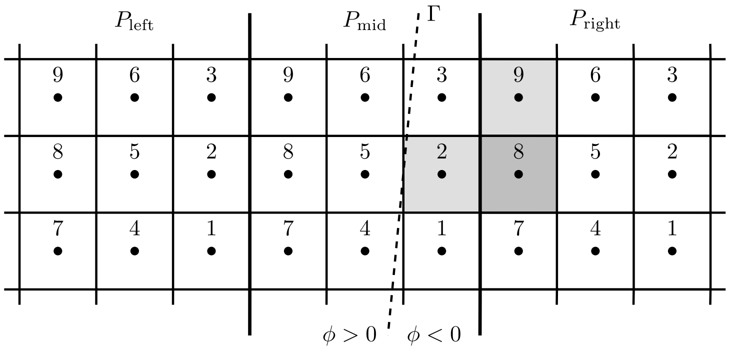
(a) The plus list map $\mathcal{A}^+$ ranges from $\min(\phi)$ to $\max(\phi)$ away from the interface, and the minus list map $\mathcal{A}^-$ ranges from $\max(\phi)$ to $\min(\phi)$ away from the interface.

(b) Combining $\mathcal{A}^+$ and $\mathcal{A}^-$ into a single list map $\mathcal{L}$, ranging from $\min(\phi)$ to $\max(\phi)$.

Fig. 14. Combining each processor's ordered list maps in the plus and minus directions into one list map for the extension routine. The domain is divided among three processors P_{left}, P_{mid}, and P_{right}.



(a) List maps $\mathcal{A}^\pm$ and orders $\mathcal{M}^\pm$ prior to merging. Dark grey indicates cells contained in $\mathcal{A}^-$, and light grey indicates cells contained in $\mathcal{A}^+$. Indices $1 - 9$ give the value of $\mathcal{M}^\pm(\mathcal{X})$.



(b) List map $\mathcal{L}$ and order $\mathcal{M}$ after merging. Indices $1 - 9$ give the value of $\mathcal{M}(\mathcal{X})$. The value f_q is being updated at $\mathcal{M}(\mathcal{X}) = 8$ by processor P_{right} (dark grey) with stencil points indicated by light grey. Processor P_{right} must wait for processor P_{mid} to communicate f_q at the cell corresponding to $\mathcal{M}(\mathcal{X}) = 2$.

Fig. 15. An example domain divided among three processors P_{left}, P_{mid}, and P_{right}, demonstrating the merging of $\mathcal{A}^\pm$ and $\mathcal{M}^\pm$ into $\mathcal{L}$ and $\mathcal{M}$.

of Γ if $m > 0$. Since the boundary domain for f_q decreases with increasing q and the outer domain increases, it is convenient to introduce Ω^q_{out} to represent the outer region that must be populated for a given q, defined as

$$\Omega^q_{\text{out}} = \left\{ \boldsymbol{x} \in (\Omega_{\text{in}} \cup \Omega_{\text{out}}) : H_q(\boldsymbol{x}) = 0 \right\}. \tag{B.1}$$

Non-shaded cells in Fig. 2 comprise Ω^q_{out}.

Recall that after performing the FMM to re-distance ϕ in the previous section, all cells are eventually added to the list of accepted cells $\mathcal{A}$, which is an ordered list map local to each processor that corresponds to the order in which cells are computed during the FMM. This is done in both the plus and minus directions from the interface, because the FMM propagates information from $\boldsymbol{x}_\Gamma$ outwards. The plus and minus lists generated during re-initialization in both directions are referred to as $\mathcal{A}^+$ and $\mathcal{A}^-$, respectively. The minus list ranges from $\max(\phi)$ to $\min(\phi)$ in the region where $\phi \le 0$, and the plus list ranges from $\min(\phi)$ to $\max(\phi)$ in the region where $\phi \ge 0$, as indicated in Fig. 14(a). The local lists $\mathcal{A}^-$ and $\mathcal{A}^+$ are

Algorithm 7 Store $\mathcal{S}$ based on upwind ϕ calculation.

```
function STORE_STENCIL($\phi, \mathcal{L}, \mathcal{M}, \mathcal{S}^{+\phi}, \mathcal{S}^{-\phi}$)    ▷ in: $\phi$, $\mathcal{L}$, $\mathcal{M}$
                                                                                                   ▷ out: $\mathcal{S}^{+\phi}$, $\mathcal{S}^{-\phi}$
    for $n = 1 \to N$ do
        $\mathcal{X} \leftarrow \mathcal{L}(n)$
        $N^+ \leftarrow 0$                                         ▷ number of neighbors to compute $\|\nabla\phi\|^+_{\mathcal{X}}$
        $N^- \leftarrow 0$                                         ▷ number of neighbors to compute $\|\nabla\phi\|^-_{\mathcal{X}}$
        for all $\mathcal{X}'$ adjacent to $\mathcal{X}$ do                     ▷ loop through all adjacent cells
            if $\mathcal{M}(\mathcal{X}') < \mathcal{M}(\mathcal{X})$ and
                $\phi(\mathcal{X}') < \phi(\mathcal{X})$ then                    ▷ ensure causality in plus direction
                $N^+ \leftarrow N^+ + 1$                            ▷ store the neighbor
                $\phi^+(N^+) \leftarrow \phi(\mathcal{X}')$                    ▷ store $\phi$ value of neighbor
            else if $\mathcal{M}(\mathcal{X}') > \mathcal{M}(\mathcal{X})$ and
                    $\phi(\mathcal{X}') > \phi(\mathcal{X})$ then                ▷ ensure causality in minus direction
                $N^- \leftarrow N^- + 1$                            ▷ store the neighbor
                $\phi^-(N^-) \leftarrow -\phi(\mathcal{X}')$                   ▷ store $-\phi$ value of neighbor
            end if
        end for
        call FMM_UPWIND_CALC($\phi, \mathcal{S}^{+\phi}, N^+, \phi^+$)          ▷ obtain stencil for $\|\nabla\phi\|^{+\phi}_{\mathcal{X}} = 1$
        call FMM_UPWIND_CALC($\phi, \mathcal{S}^{-\phi}, N^-, \phi^-$)          ▷ obtain stencil for $\|\nabla\phi\|^{-\phi}_{\mathcal{X}} = 1$
    end for
end function
```

Algorithm 8 Extrapolation of f_q in the plus direction.

```
function PLUS_EXTRAP($\phi, f_{q+1}, H_q, f_q$)                              ▷ in: $\phi$, $H_q$, $f_{q+1}$
                                                                           ▷ out: $H_q$, $f_q$
    for $n = 1 \to N$ do
        $\mathcal{X} \leftarrow \mathcal{L}(n)$
        if $H_q(\mathcal{X}) == 1$ then
            cycle                                                ▷ cell $\mathcal{X}$ already updated or $\notin \Omega^q_{\text{out}}$
        end if
        $N^+ \leftarrow 0$                                             ▷ number of neighbors to compute $\|\nabla\phi\|^+_{\mathcal{X}}$
        for all $\mathcal{X}'$ adjacent to $\mathcal{X}$ for which $\mathcal{S}^{+\phi}(\mathcal{X}) == 1$ do          ▷ stored in Algorithm 7
            while $H_q(\mathcal{X}') == 0$ do                        ▷ does stencil point belong to different processor?
                wait to receive $f_q(\mathcal{X}')$                    ▷ if so, wait to receive $f_q$ from neighbor
                $H_q(\mathcal{X}') \leftarrow 1$                      ▷ once $f_q$ is received, mark stencil point as ready for use
            end while
            $N^+ \leftarrow N^+ + 1$
            $\phi^+(N^+) \leftarrow \phi(\mathcal{X}')$                              ▷ store $\phi$ value at stencil point
            $f^+_q(N^+) \leftarrow f_q(\mathcal{X}')$                               ▷ store $f_q$ value at stencil point
        end for
        compute $f_q(\mathcal{X})$ from $N^+$, $\phi^+(1:N^+)$, $f^+_q(1:N^+)$, $f_{q+1}(\mathcal{X})$     ▷ update $f_q$ (Eq. (30))
        $H_q(\mathcal{X}) \leftarrow 1$                                         ▷ mark cell as updated
        if $\mathcal{X}$ is at an interprocessor boundary then
            send $f_q(\mathcal{X})$ to buffer                      ▷ communicate updated $f_q$ to neighbor processors
        end if
    end for
end function
```

combined into a single local list map $\mathcal{L}$ that is ordered from $\min(\phi)$ to $\max(\phi)$ on all processor domains, as indicated in Fig. 14(b). For N cells stored in $\mathcal{L}$, the location in computational space $\mathcal{X}$ of any cell $n \in \{1, 2, \ldots, N\}$ can be accessed via

$$\mathcal{X} = \mathcal{L}(n). \tag{B.2}$$

Similarly, the order n for any cell at location $\mathcal{X}$ can be obtained from the spatial function $\mathcal{M}$ through

$$n = \mathcal{M}(\mathcal{X}). \tag{B.3}$$

Fig. 15(a) shows an example domain divided among three processors, demonstrating $\mathcal{M}^+$ and $\mathcal{M}^-$ on each processor prior to merging them into a single $\mathcal{M}$, which is shown in Fig. 15(b).

Once $\mathcal{L}$ is created, the local stencils $\mathcal{S}^{\pm\phi}(\mathcal{X})$ used for extrapolation are obtained. The stencil $\mathcal{S}^{+\phi}$ contains the entire set of switches $S^{\pm}_{x,y,z}$ (defined in Section 3.2.2) for extrapolating in the direction of increasing values of ϕ, i.e., $\mathcal{S}^{+\phi}$ determines which accepted neighbors should be used. The stencil $\mathcal{S}^{-\phi}$ contains the same information for extrapolation toward decreasing ϕ values. If, for example, $\mathcal{S}^{+\phi}(\mathcal{X})$ contains the neighbor to the left and the cell to the left of cell $\mathcal{X}$ is a ghost cell, then the processor containing cell $\mathcal{X}$ must wait until the ghost cell is updated and communicated by the processor that controls it before it can be used to compute $f_q(\mathcal{X})$. An example of such a scenario is shown in Fig. 15(b). The stencils $\mathcal{S}^{\pm\phi}$ are stored in Algorithm 7 based on the upwind stencil for cell $\mathcal{X}$ that solves $\|\nabla\phi\|_{\mathcal{X}} = 1$, *according to the causal nature now determined by* $\mathcal{L}$, $\mathcal{M}$, *and* ϕ. This is an important distinction, as it allows us to perform extrapolations with $m > 0$ that require propagating information in a direction opposite to the interfacial normal (see the end of Section 3.2.2).

After the stencils $\mathcal{S}^{\pm\phi}$ are stored, f_q can be extrapolated as in Algorithm 8, which outlines the procedure in the direction of increasing ϕ. Note that the flag function $H_q(\mathcal{X})$ which denotes boundary cells is updated from 0 to 1 whenever $f_q(\mathcal{X})$ is updated. Extrapolation in the direction of decreasing ϕ follows the same algorithm, but the outermost loop index n goes from $N \to 1$ rather than from $1 \to N$, and the neighbors $\mathcal{X}'$ are determined by $\mathcal{S}^{-\phi}$ rather than $\mathcal{S}^{+\phi}$.

References

[1] M. Sussman, P. Smereka, S. Osher, A level set method for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.
[2] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438.
[3] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (1) (2002) 83–116.
[4] D. Enright, S. Marschner, R. Fedkiw, Animation and rendering of complex water surfaces, in: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, vol. 21, ACM Press, New York, NY, USA, 2002, p. 736.
[5] O. Desjardins, V. Moureau, H. Pitsch, An accurate conservative level set/ghost fluid method for simulating turbulent atomization, J. Comput. Phys. 227 (18) (2008) 8395–8416.
[6] O. Desjardins, H. Pitsch, A spectrally refined interface approach for simulating multiphase flows, J. Comput. Phys. 228 (5) (2009) 1658–1677.
[7] M. Owkes, O. Desjardins, A discontinuous Galerkin conservative level set scheme for interface capturing in multiphase flows, J. Comput. Phys. 249 (1) (2013) 275–302.
[8] O. Desjardins, J.O. McCaslin, M. Owkes, P. Brady, Direct numerical and large-eddy simulation of primary atomization in complex geometries, At. Sprays 23 (11) (2013) 1001–1048.
[9] J.O. McCaslin, O. Desjardins, A localized re-initialization equation for the conservative level set method, J. Comput. Phys. 262 (2014) 408–426.
[10] J.O. McCaslin, O. Desjardins, Numerical investigation of gravitational effects in horizontal annular liquid–gas flow, Int. J. Multiph. Flow (2014), forthcoming.
[11] R.K. Shukla, C. Pantano, J.B. Freund, An interface capturing method for the simulation of multi-phase compressible flows, J. Comput. Phys. 229 (19) (2010) 7411–7439.
[12] J. Zhu, J.A. Sethian, Projection methods coupled to level set interface techniques, J. Comput. Phys. 102 (1) (1992) 128–138.
[13] C.W. Rhee, L. Talbot, J.A. Sethian, Dynamical behaviour of a premixed turbulent open V-flame, J. Fluid Mech. 300 (1995) 87–115.
[14] H. Pitsch, O. Desjardins, G. Balarac, M. Ihme, Large-eddy simulation of turbulent reacting flows, Prog. Aerosp. Sci. 44 (6) (2008) 466–478.
[15] B. Van Poppel, O. Desjardins, J. Daily, A ghost fluid, level set methodology for simulating multiphase electrohydrodynamic flows with application to liquid fuel injection, J. Comput. Phys. 229 (20) (2010) 7977–7996.
[16] Y. Lin, Two-phase electro–hydrodynamic flow modeling by a conservative level set model, Electrophoresis 34 (5) (2013) 736–744.
[17] A. Gravouil, N. Moës, T. Belytschko, Non-planar 3D crack growth by the extended finite element and level sets, part II: Level set update, Int. J. Numer. Methods Eng. 53 (11) (2002) 2569–2586.
[18] N. Moës, A. Gravouil, T. Belytschko, Non-planar 3D crack growth by the extended finite element and level sets, part I: Mechanical model, Int. J. Numer. Methods Eng. 53 (11) (2002) 2549–2568.
[19] S. Osher, R.P. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2) (2001) 463–502.
[20] V.A. Luminita, T.F. Chan, A multiphase level set framework for image segmentation using the Mumford and Shah model, Int. J. Comput. Vis. 50 (3) (2002) 271–293.
[21] D. Chopp, Recent advances in the level set method, in: Handbook of Biomedical Image Analysis, Kluwer Academic, 2005, pp. 201–256, Ch. 4.
[22] R.P. Fedkiw, T.D. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), J. Comput. Phys. 152 (2) (1999) 457–492.
[23] T.D. Aslam, A level-set algorithm for tracking discontinuities in hyperbolic conservation laws: I. Scalar equations, J. Comput. Phys. 167 (2) (2001) 413–438.
[24] R.P. Fedkiw, T.D. Aslam, S. Xu, The ghost fluid method for deflagration and detonation discontinuities, J. Comput. Phys. 154 (2) (1999) 393–427.
[25] F. Losasso, R.P. Fedkiw, S. Osher, Spatially adaptive techniques for level set methods and incompressible flow, Comput. Fluids 35 (10) (2006) 995–1010.
[26] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving Stefan problems, J. Comput. Phys. 135 (1) (1997) 8–29.
[27] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. Natl. Acad. Sci. 93 (4) (1996) 1591–1595.
[28] D. Adalsteinsson, J.A. Sethian, The fast construction of extension velocities in level set methods, J. Comput. Phys. 148 (1) (1999) 2–22.
[29] 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.
[30] T.D. Aslam, A level set algorithm for tracking discontinuities in hyperbolic conservation laws II: systems of equations, J. Sci. Comput. 19 (1–3) (2003) 37–62.
[31] T.D. Aslam, A partial differential equation approach to multidimensional extrapolation, J. Comput. Phys. 193 (1) (2004) 349–355.
[32] C. Min, F. Gibou, A second order accurate level set method on non-graded adaptive cartesian grids, J. Comput. Phys. 225 (1) (2007) 300–321.
[33] S. Tanguy, T. Ménard, A. Berlemont, A level set method for vaporizing two-phase flows, J. Comput. Phys. 221 (2) (2007) 837–853.
[34] D.L. Chopp, Another look at velocity extensions in the level set method, SIAM J. Sci. Comput. 31 (5) (2009) 3255–3273.
[35] J. Strain, A fast modular semi-Lagrangian method for moving interfaces, J. Comput. Phys. 161 (2) (2000) 512–536.
[36] M. Herrmann, A domain decomposition parallelization of the fast marching method, in: Center for Turbulence Research Annual Research Briefs, 2003, pp. 213–225.
[37] J.A. Sethian, Theory, algorithms, and applications of level set methods for propagating interfaces, Acta Numer. 5 (1996) 309–395.
[38] J.A. Sethian, Fast marching methods, SIAM Rev. 41 (1999) 199–235.
[39] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1999.