Level-Set Method to Solve the two-dimensional (2D) Porous-Fisher-Stefan Model

Nizhum Rahmam, Alexander KY Tam and Matthew J Simpson

January 5, 2024

This document describes numerical methods used to solve the two-phase 2D Porous-Fisher-Stefan model.

1 Level-Set Method

We implement the level-set method [1–3] to compute solutions to the two-dimensional Porous-Fisher-Stefan model.

In this context, we consider the domain in which we solve the Porous-Fisher-Stefan model, denoted as $\Omega(t)$;, where $\partial\Omega(t)$ denotes the boundary separating the populated and partially vacant regions. The 2D Porous-Fisher-Stefan model for the population variable u is defined as follows:

$$\frac{\partial u}{\partial t} = u(1-u) + \frac{\partial}{\partial x}\left[u^m \frac{\partial u}{\partial x}\right] + \frac{\partial}{\partial y}\left[u^m \frac{\partial u}{\partial y}\right] \text{ on } \Omega(t)\,, \tag{1a}$$

$$u(x,y,t) = 0 \text{ on } \partial\Omega(t)\,, \quad u(x,y,0) = U(x,y) \text{ on } \Omega(0)\,, \tag{1b}$$

$$\mathcal{V} = -\kappa\, u^m\, \nabla u \cdot \hat{n} \text{ on } \partial\Omega(t)\,, \tag{1c}$$

where U is the initial population density. The normal velocity of the interface, denoted as $\mathcal{V}$, is determined by the Stefan condition, with the Stefan constant κ, and $\hat{n}$ represents the unit normal vector along the moving boundary. Here m is a positive constant.

To address the model (1), we introduce a new function, $\phi(x,y,t) = [u(x,y,t)]^{m+1}$, which allows us to transform the non-linear Stefan condition into a linear one at the moving boundary. Therefore, we get the following model:

$$\frac{\partial \phi}{\partial t} = (1+m)\,\phi\,(1-\phi^{1/(1+m)}) + \phi^{m/(1+m)}\left(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right) \text{ on } \Omega(t)\,, \tag{2a}$$

$$\phi(x,y,t) = 0 \text{ on } \partial\Omega(t)\,, \quad \phi(x,y,0) = \Phi(x,y) \text{ on } \Omega(0)\,, \tag{2b}$$

$$\mathcal{V} = -\frac{\kappa}{1+m}\nabla\phi\cdot\hat{n} \text{ on } \partial\Omega(t)\,, \tag{2c}$$

where we define $U = \Phi$.

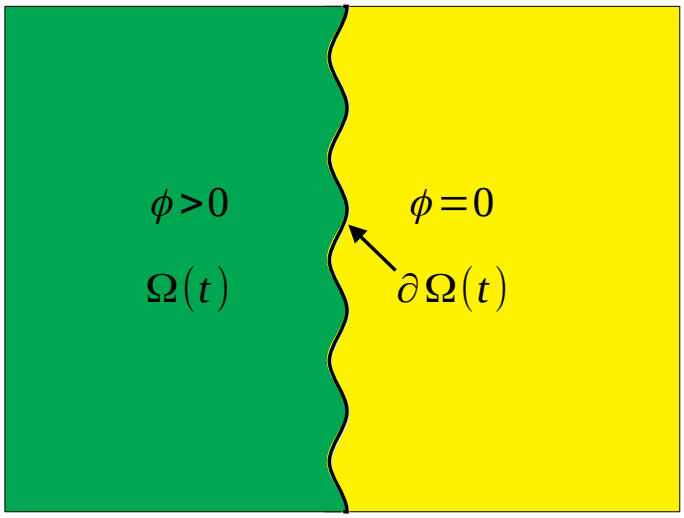


Figure 1: The two-dimensional computational domain for solving the Porous-Fisher-Stefan model using the level-set method. The function $\phi(x, y, t)$ is defined in the regions $\Omega(t)$ (shaded in green).

For solving equations (2), on a two-dimensional rectangular computational domain $\mathcal{D}$, we implement the numerical level-set method. In this method, the interface is embedded as the zero level-set using the scalar function $\varphi(x, y, t)$. The interface is given by

$$\partial\Omega(t) = \{x, y \mid \varphi(x, y, t) = 0\} \,, \tag{3}$$

where $\varphi(x, y, t)$ is defined everywhere in the computational domain, $\mathcal{D}$, and maintains the properties $\varphi < 0$ on $\Omega(t)$. In order to ensure that $\varphi = 0$ is maintained on the interface $\partial\Omega(t)$, the evolution of φ satisfies the following equation,

$$\frac{\partial\varphi}{\partial t} + F \mid \nabla\varphi \mid = 0, \tag{4}$$

where the scalar function $F(x, y, t)$ is defined for $(x, y) \in \mathcal{D}$. This function F represents the velocity $\mathcal{V}$ on the interface and is equal on the interface, such that $F = \mathcal{V}$ on $\partial\Omega(t)$. After introducing the level-set function, we rewrite (2) as follows:

$$\frac{\partial\phi}{\partial t} = (1+m)\,\phi\,(1-\phi^{1/(1+m)}) + \phi^{m/(1+m)}\left(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right) \text{ on } \varphi(x, y, t) < 0 \,, \tag{5a}$$

$$\frac{\partial\varphi}{\partial t} + F \mid \nabla\varphi \mid = 0 \text{ on } (x, y) \in \mathcal{D} \,, \tag{5b}$$

$$\phi(x, y, t) = 0 \text{ on } \varphi(x, y, t) = 0 \,, \quad \phi(x, y, 0) = \Phi(x, y) \text{ on } \varphi(x, y, 0) < 0 \,, \tag{5c}$$

$$F = -\frac{\kappa}{1+m}\nabla\phi\cdot\hat{n} \text{ on } \varphi(x, y, t) = 0 \,. \tag{5d}$$

2 Numerical Algorithm

We address the system (5) within the confines of a two-dimensional computational domain denoted as $\mathcal{D}$, which spans the interval $[-L_x, L_x] \times [-L_y, L_y]$. Here, L_x and L_y

correspond to the respective positive widths along the x and y-directions. The solution is computed over a regularly spaced grid defined by coordinates (x_i, y_j). These grid points also referred to as nodes, are determined by the expressions $x_i = i\Delta$ and $y_j = j\Delta$, for $i = 0, ...N_x$, and $j = 0, ...N_y$. In this context, N_x and N_y represent the number of grid intervals in the x and y directions, while Δ is the equal constant grid spacing in both directions. In the temporal domain, we define $t_k = k\Delta t$, where $k = 0, ...N_t$, and Δt remains constant. The symbols $\phi^k_{i,j} = \phi(x_i, y_j, t_k)$, $\varphi^k_{i,j} = \varphi(x_i, y_j, t_k)$, and $F_{i,j} = F(x_i, y_j)$ are then utilized to represent the value of each quantity at discrete points in both time and space.

The approach for solving the Porous-Fisher-Stefan model through the level-set method encompasses the subsequent steps:

1. Initialise the problem by specifying the model parameters κ, along with the initial conditions $\phi_0(x)$ and $\varphi_0(x)$.

2. Progress in time to compute $\phi^{k+1}_{i,j}$, $\varphi^{k+1}_{i,j}$, and $F^{k+1}_{i,j}$ for $i = 0, ...N_x$, $j = 0, ...N_y$, and $k = 0, ..., N_t - 1$. At each time increment, execute the subsequent steps:

   (a) Determine the region $\Omega(t)$ and ascertain the location of the interface denoted by $\partial\Omega(t)$.

   (b) Solve the two-dimensional Porous-Fisher equation (5a) while adhering to the boundary condition (5c).

   (c) Calculate the interface velocity denoted as $\mathcal{V}$ by applying the Stefan condition (5d).

   (d) Solve the level-set equation (5b) to calculate $\varphi^{k+1}_{i,j}$, thereby updating the position of the interface.

   (e) (Optional) Reinitialise the level-set function as a signed-distance function within a suitable neighborhood of the interface.

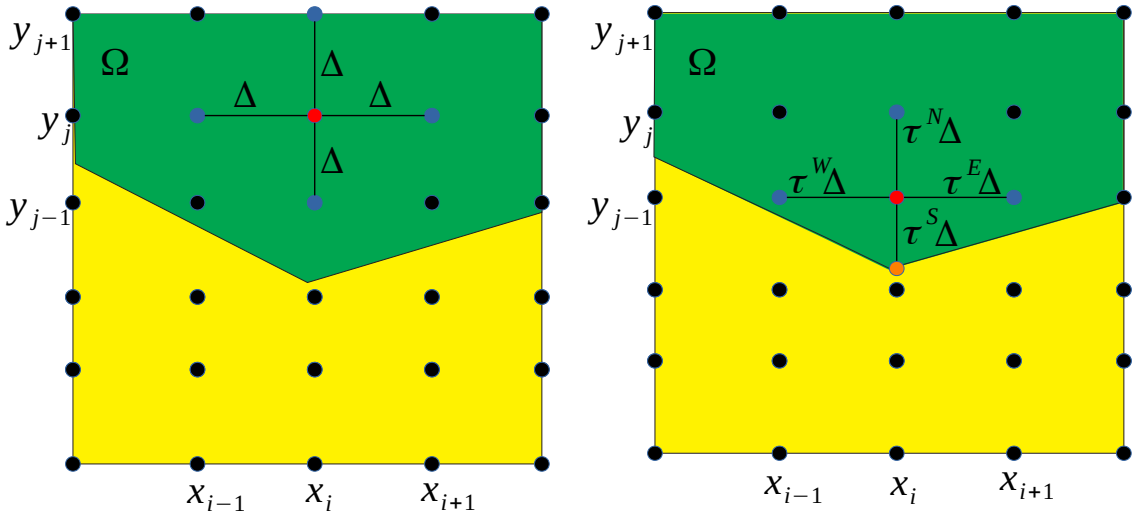
We execute this process by utilizing code written in Julia. A comprehensive explanation of the techniques employed for steps 2. (a) to 2. (e) can be found in the following subsections.

2.1 Domain and Interface

After specifying the parameters and initial conditions, the initial step involves identifying the domains $\Omega(t)$ and interface $\partial\Omega(t)$. For determining the domain $\Omega(t)$ associated with the function $\phi(x, y, t)$, all interior grid points (x_i, y_j) with $i = 1, ...N_x - 1$ and $j = 1, ...N_y - 1$ are categorized as either inside or outside $\Omega(t)$. Specifically, $(x_i, y_j) \in \Omega(t)$ if $\varphi(x_i, y_j, t) < 0$, and $(x_i, y_j) \notin \Omega(t)$ otherwise. To identify the interface $\partial\Omega(t)$, grid points (x_i, y_j) are singled out such that the sign of φ at any adjacent grid point is opposite.

We designate these as irregular grid points (and points where φ has the same sign at all adjacent nodes as regular grid points). Figure (2) illustrates the contrast between regular and irregular grid points. In Figure (2)(a), the red node is a regular grid point because $\varphi_{i,j} < 0$, and $\varphi < 0$ at all adjacent (yellow) nodes. However, in Figure (2)(b), the red node is irregular because its southern neighbouring node (x_i, y_{j-1}) lies outside

$\Omega(t)$. To differentiate between regular and irregular grid points, we define $\tau_{i,j}^W$, $\tau_{i,j}^E$, $\tau_{i,j}^N$, and $\tau_{i,j}^S$ as the distances (relative to grid spacing) between the grid point (x_i, y_j) and a neighbouring node (western, eastern, southern, or northern, respectively) or the interface, whichever is smaller. For example, if $j \neq= 0$



(a) Regular grid point. (b) Irregular grid point.

Figure 2: Regular and irregular grid points within the computational domain are depicted. Black dots signify grid points, the red dot corresponds to the point (x_i, y_j), and blue dots denote nodes adjacent to (x_i, y_j) that lie within $\Omega(t)$. In the case of the irregular point shown in (b), the orange dot represents a ghost node situated on the interface $\partial\Omega$, specifically in the southern direction from (x_i, y_j). The domains and interfaces of populations $\phi(x, t)$ are also illustrated.

$$\tau_{i,j}^S = \begin{cases} 1 & \text{if} \quad \operatorname{sgn}(\varphi_{i,j}) = \operatorname{sgn}(\varphi_{i,j-1}) \\ \dfrac{\varphi_{i,j}}{\varphi_{i,j} - \varphi_{i,j-1}} & \textit{otherwise}, \end{cases} \tag{6}$$

In cases where the interface traverses between (x_i, y_j) and its southern neighbour, we employ linear interpolation at irregular grid points to determine the zero level-set ($\varphi = 0$). If such is the scenario, the position of the interface ghost node (represented by the orange dot in Figure (2)(b) at the zero level-set) is denoted as (x_i, y^G), where $y^G = y_j - \tau_{i,j}^S \Delta$. In situations where $j = 0$, we adjust (6) to accommodate periodic boundaries, resulting in the expression

$$\tau_{i,0}^S = \begin{cases} 1 & \text{if} \quad \operatorname{sgn}(\varphi_{i,j}) = \operatorname{sgn}(\varphi_{i,j-1}) \\ \dfrac{\varphi_{i,0}}{\varphi_{i,0} - \varphi_{i,N_y-1}} & \textit{otherwise}. \end{cases} \tag{7}$$

Equivalent interpolation formulations can be derived for $\tau_{i,j}^W$, $\tau_{i,j}^E$, and $\tau_{i,j}^N$ as needed, and their values are stored at each interior grid point in $\mathcal{D}$. These values facilitate the construction of finite difference methods for the Porous-Fisher equation and the extension velocity field, ensuring adherence to the shape of Ω and the location of the interface. Throughout the development of our numerical approaches, we assume that

the interface exists at a maximum of one location between adjacent grid points. In other words, there is at most one position between adjacent grid points where $\varphi = 0$. If Ω and the grid spacing result in multiple interface locations, we refine the grid until Δ is sufficiently small for only one to persist.

2.2 Porous-Fisher-Stefan Model

We employ the explicit finite-difference method of lines to solve the 2D Porous-Fisher equation. At every interior grid point in $\mathcal{D}$, we discretize the right-hand side of (5a) using second-order accurate finite-difference approximations for the derivatives. This results in a semi-discrete system of ordinary differential equations (ODEs) given by

$$\frac{\mathrm{d}\phi_{i,j}}{\mathrm{d}t} = \begin{cases} (1+m)\ \phi_{i,j}\ (1-\phi_{i,j}^{1/(1+m)}) + \ \phi_{i,j}^{m/(1+m)}\ (\nabla_d^2 \phi_{i,j}) & \text{if} \quad \varphi(x_i, y_j) \leq 0 \\ 0 & \text{if} \quad \varphi(x_i, y_j) > 0, \end{cases} \tag{8}$$

for $i = 0, ..., N_x$ and $j = 0, ..., N_y$. We use Dirichlet conditions $\phi_{0,j} = \phi_{N_x,j} = 1$. We use periodic conditions (on the cylinder) $\phi_{i,0} = \phi_{i,N_y-1}$ and $\phi_{i,N_y} = \phi_{i,1}$ and $u_f = 0$ on the interface. The term $\nabla_d^2 \phi_{i,j}$ denotes the second order central finite-difference approximation for the Laplacian, $\nabla^2 \phi = \phi_{xx} + \phi_{yy}$.

The finite-difference discretisation of $\nabla^2 \phi$ differs depending on whether x_i, y_i is regular or irregular.

Therefore, using the standard finite different stencil, Eq. (8) for $\varphi(x_i, y_j) \leq 0$ becomes,

$$\frac{\mathrm{d}\phi_{i,j}}{\mathrm{d}t} = (1+m)\ \phi_{i,j}\ (1-\phi_{i,j}^{1/(1+m)}) + \ \frac{\phi_{i,j}^{m/(1+m)}}{\Delta^2}\ (\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j})\ . \tag{9}$$

However, utilizing (9) at irregular grid points is impractical, as one or more grid points on the standard stencil may lie outside Ω for $\phi_{i,j}$. Instead, we introduce ghost nodes on the interface (such as the orange dot in the southern direction in Figure (2)(b)) to substitute for nodes outside Ω that would otherwise be part of the standard stencil. The density at these ghost nodes conforms to the interface condition $\phi = 0$. With our defined values of $\tau_{i,j}^K$, where $K \in W, E, N, S$ above, this leads to the utilization of the non-standard finite-difference stencil (9).

$$\begin{aligned} \frac{\mathrm{d}\phi_{i,j}}{\mathrm{d}t} &= (1+m)\ \phi_{i,j}\ (1-\phi_{i,j}^{1/(1+m)}) \\ &\quad + \ \phi_{i,j}^{m/(1+m)}\ \frac{2}{\Delta^2}\left(\frac{\phi_{i,j}^E}{\tau_{i,j}^E(\tau_{i,j}^E + \tau_{i,j}^W)} - \frac{\phi_{i,j}}{\tau^E \tau^W} + \frac{\phi_{i,j}^E}{\tau_{i,j}^W(\tau_{i,j}^E + \tau_{i,j}^W)} \ + \frac{\phi_{i,j}^N}{\tau_{i,j}^N(\tau_{i,j}^N + \tau_{i,j}^S)} \right. \\ &\quad \left. - \frac{\phi_{i,j}}{\tau^N \tau^S} + \frac{\phi_{i,j}^S}{\tau_{i,j}^S(\tau_{i,j}^N + \tau_{i,j}^S)} \right)\ , \end{aligned} \tag{10}$$

where the superscripts W, E, S and N denote the western, eastern, southern and northern neighbours to (x_i, y_j) respectively. The values $\phi_{i,j}^W$, $\phi_{i,j}^E$, $\phi_{i,j}^N$ and $\phi_{i,j}^S$ depend on whether the neighbouring nodes lie inside $\Omega(t)$. For example, we have

$$\phi_{i,j}^{W} = \begin{cases} \phi_{i-1,j} & \text{if} \quad \varphi(x_{i-1}, y_j) < 0 \\ 0 & \text{if} \quad \varphi(x_{i-1}, y_j) \leq 0, \end{cases} \tag{11}$$

and similar for $\phi_{i,j}^{E}$, $\phi_{i,j}^{N}$ and $\phi_{i,j}^{S}$. We note that if $\tau_{i,j}^{K} = 1$, then the irregular stencil (10) is identical to the regular stencil (9).

The approximation (10) can involve division by small quantities if any $\tau_{i,j}^{K}(K \in N, E, S, W)$ is small. If so, small time steps would be required in the method of lines to retain numerical stability. Following Morrow et al. [4], we circumvent this difficulty by setting $\phi_{i,j} = 0$ if any $\tau_{i,j}^{K} < \tau_{\partial\Omega}$ for some threshold $\tau_{\partial\Omega}$. In all computed solutions, we use $\tau_{\partial\Omega} = 0.01$

After formulating the system of ODEs (8) at all interior grid points within the computational domain, we perform time integration using the fifth-order Tsitouras Runge–Kutta method [5]. The integration is executed utilizing the built-in ODE solver in DifferentialEquationsjl [6]. In each time step of our algorithm, we solve to progress in time by an increment of Δt. The solver may take multiple intermediate time steps depending on the numerical stability of (8). We maintain this adaptive time-stepping for accuracy and apply an absolute tolerance of 1×10^{-3} and a relative tolerance of 1×10^{-6}.

2.3 Constructing the extension velocity Field

After updating the values of $\phi(x, y, t)$ according to the 2D Porous-Fisher equation, the subsequent task is to compute an extension velocity field, $F(x, y, t)$, crucial for evolving the interface. Constructing $F(x, y, t)$ involves a two-step process. Firstly, we must determine the normal speed on the interface, denoted as $\mathcal{V}$, by implementing the Stefan condition (5d). Subsequently, we extend this velocity to formulate the field defined for all $x \in \mathcal{D}$. Employing non-standard, second-order finite-differences, and linear interpolation facilitates the implementation of the Stefan condition. Following this, we extrapolate the velocity field in the direction orthogonal to the interface by solving a partial differential equation (PDE).

2.3.1 Computing the Interface Velocity

We enforce the Stefan condition at all ghost nodes on the interface (depicted as orange nodes in Figure (3) and subsequent figures). For the application of (5d), we implement second-order accurate finite differences for the first spatial derivatives of ϕ and φ, denoted as ϕ_x, ϕ_y, φ_x, and φ_y. The finite-difference scheme relies on the interface's location and the positions of adjacent nodes. Without loss of generality, we describe the procedure when the ghost node lies in the x-direction between two grid points, and the region $\Omega(t)$ lies directly west and north of the ghost node (refer to Figure 3). Although this configuration is not the sole possibility, alternative approximations can be devised when necessary.

To approximate ϕ_x, we generally apply a second-order accurate finite-difference approximation. In cases where the grid point to the left of the ghost node is within Ω (as illustrated in Figure 3), this approximation takes the following form

$$\phi_x = \frac{1}{\Delta}\left[\frac{(2\tau_{i,j}^E + \tau_{i,j}^W)\, u_f}{\tau_{i,j}^E(\tau_{i,j}^E + \tau_{i,j}^W)} - \frac{(\tau_{i,j}^E + \tau_{i,j}^W)\, \phi_{i,j}}{\tau_{i,j}^E \tau_{i,j}^W} + \frac{\tau_{i,j}^E \phi_{i,j}^W}{\tau_{i,j}^W(\tau_{i,j}^E + \tau_{i,j}^W)}\right] \tag{12}$$

The value $\phi_{i,j}^W$ is contingent upon whether the node (x_{i-1}, y) lies within Ω. It is important to note that if $\tau_{i,j}^E = \tau_{i,j}^W = 1$, then (12) represents the standard second-order one-sided difference for ϕ_x. In cases where $(x_{i-1}, y_j) \in \Omega$ and $(x_{i-2}, y_j) \in \Omega$, we employ the standard formula

$$\phi_x = \frac{3\phi_f - 4\phi_{i-1,j} + \phi_{i-2,j}}{2\Delta}. \tag{13}$$

This situation is depicted in Figure 4(a). If $(x_{i-1}, y_j) \notin \Omega$, we assign ϕ_x a value of 0. In cases where $(x_{i-1}, y_j) \in \Omega$ but $(x_{i-2}, y_j) \notin \Omega$ (as illustrated in Figure 4(b)), we modify (13) to derive an appropriate non-standard three-point stencil. The formula is determined by evaluating (12) at the point (x_{i-1}, y_j) with $\tau_{i-i,j}^E = 1$ and $\tau_{i-i,j}^W = u_f$. If $\tau_{i-i,j}^W < \tau_{\partial\Omega}$, we instead set ϕ_x to 0.

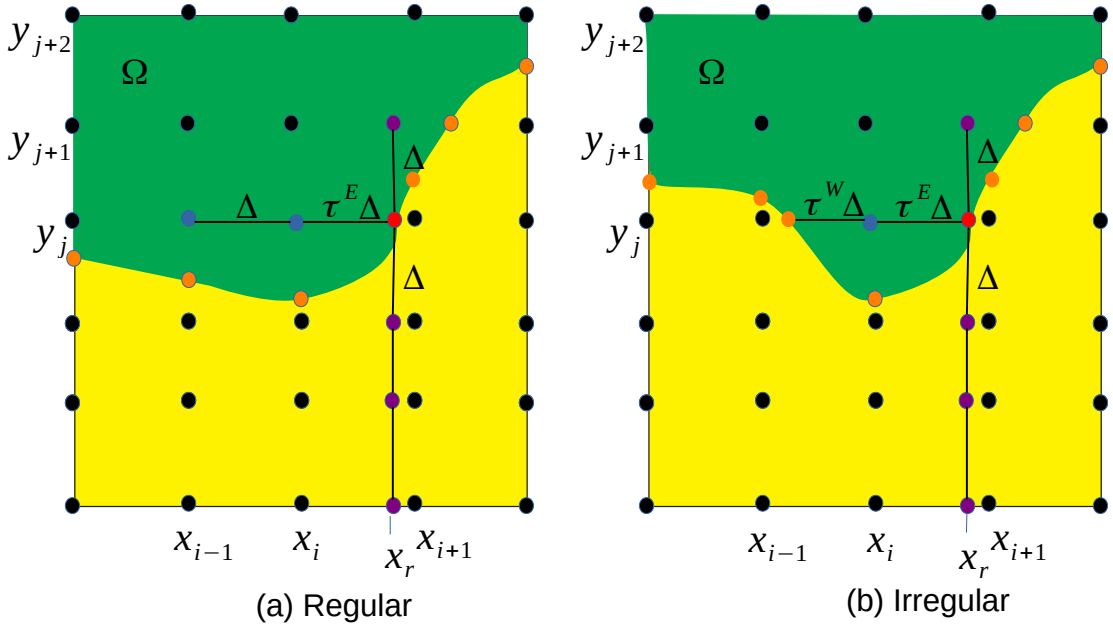
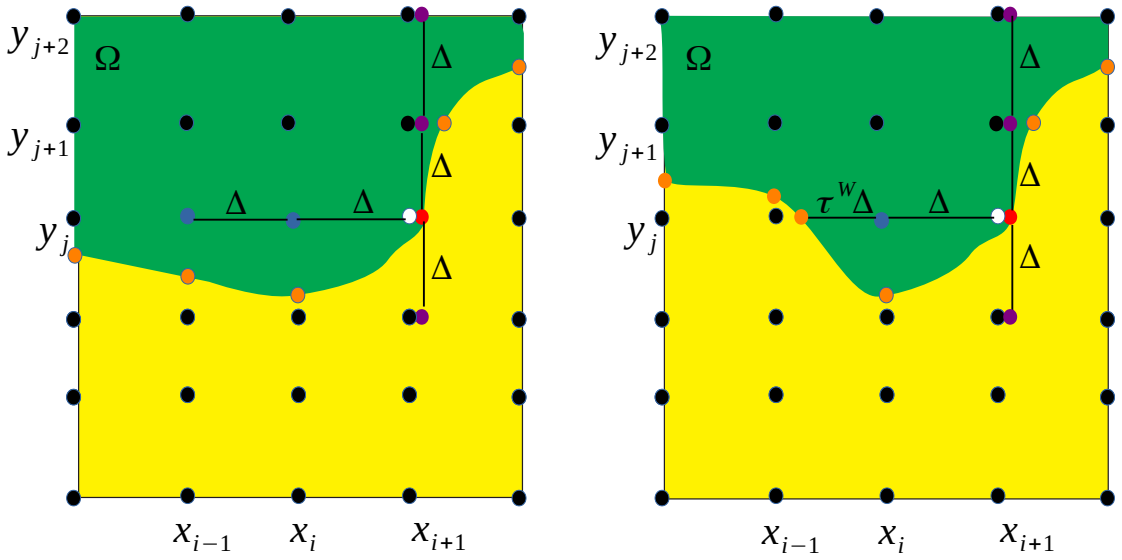


Figure 3: Finite-difference stencils used to calculate the interface normal speed for grid points with $\tau_{i,j}^E > \tau_{\partial\Omega}$. Black dots represent nodes of the computational grid. The red dots signify the ghost node where $\mathcal{V}$ is computed using the Stefan condition, and the orange dots represent other ghost nodes on the interface. Blue nodes indicate grid points utilized in the stencils for ϕ_x, while pink dots denote ghost nodes used in the stencils for ϕ_y.



(a) Standard one-sided stencil (b) Non-standard one-sided stencil

Figure 4: Finite-difference stencils are used for calculating the interface normal speed at grid points with $\tau^E_{i,j} > \tau_{\partial\Omega}$. Ghost nodes on the interface, marked in orange and red, are where the speed is computed using the Stefan condition. Blue nodes represent grid points utilized in the stencils for ϕ_x, while pink dots indicate ghost nodes used in the stencils for ϕ_y. The white dot denotes the grid point close to the interface, where we assume $\phi_{i,j} = u_f$. Black dots represent nodes of the computational grid.

Following the approximation of ϕ_x, the subsequent task is to calculate the derivatives of the level-set function, namely φ_x and φ_y. For computing φ_x, we initially employ second-order central finite differences at the two grid points neighbouring the interface ghost node.

$$(\varphi_{i,j})_x = \frac{\varphi_{i+1,j} - \varphi_{i-1,j}}{2\Delta} \tag{14a}$$

$$(\varphi_{i+1,j})_x = \frac{\varphi_{i+2,j} - \varphi_{i,j}}{2\Delta} \tag{14b}$$

To estimate the value of φ_x at the interface ghost node, we subsequently apply linear interpolation to achieve

$$\varphi_x = \tau^E_{i,j}(\varphi_{i+1,j})_x + (1 - \tau^E_{i,j})(\varphi_{i,j})_x \ . \tag{15}$$

For the estimation of φ_y, we introduce two supplementary ghost nodes positioned directly above and below the interface ghost node (depicted as pink dots in Figures 3 and 4). Subsequently, we employ linear interpolation to approximate φ at these nodes, resulting in

$$\varphi_{r,j+1} = \tau^E_{i,j}\ \varphi_{i+1,j+1} + (1 - \tau^E_{i,j})\varphi_{i,j+1} \ , \tag{16a}$$

$$\varphi_{r,j-1} = \tau^E_{i,j}\ \varphi_{i+1,j-1} + (1 - \tau^E_{i,j})\varphi_{i,j-1} \ . \tag{16b}$$

The first derivative approximation is then obtained using the central scheme

$$\varphi_y = \frac{\varphi_{r,j+1} - \varphi_{r,j-1}}{2\,\Delta}\ . \tag{17}$$

In the scenario where the ghost node along the interface is aligned in the x-direction, linear interpolation is required to approximate ϕ_y. To execute a one-sided difference, we introduce two additional ghost nodes positioned directly above the interface's ghost node (as illustrated in Figures (3) and (4)), leading to

$$\phi_{r,j} = \phi_f\ , \tag{18a}$$
$$\phi_{r,j+1} = \tau^E_{i,j}\ \phi_{i+1,j+1} + (1 - \tau^E_{i,j})\phi_{i,j+1}\ , \tag{18b}$$
$$\phi_{r,j+2} = \tau^E_{i,j}\ \phi_{i+1,j+2} + (1 - \tau^E_{i,j})\phi_{i,j+2}\ . \tag{18c}$$

The subsequent one-sided finite-difference approximation at the second order is as follows:

$$\phi_y = \frac{3\phi_f - 4\phi_{r,j+1} + \phi_{r,j+2}}{2\Delta}. \tag{19}$$

The approximation provided in equation (19) assumes that both ghost nodes (x_r, y_{j+1}) and (x_r, y_{j+2}) are situated within Ω, indicating that $\varphi_{r,j+1} < 0$ and $\varphi_{r,j+2} < 0$. The value of $\varphi_{r,j+1}$ can be validated by applying (16a), and the value of $\varphi_{r,j+2}$ can be estimated through similar linear interpolation

$$\varphi_{r,j+2} = \tau^E_{i,j}\ \varphi_{i+1,j+2} + (1 - \tau^E_{i,j})\varphi_{i,j+2}\ . \tag{20}$$

If $\varphi_{r,j+1} < 0$ and $\varphi_{r,j+2} < 0$, we employ Eq.(19). However, in the scenario where $\varphi_{r,j+1} < 0$ but $\varphi_{r,j+2} \geq 0$, we introduce an additional ghost node (as depicted in Figure (5)) and establish the following:

$$\tau^G = \frac{\varphi_{r,j+1}}{(\varphi_{r,j+1} - \varphi_{r,j+2})}\ , \tag{21}$$

as the distance from the ghost node at x_r, y_{j+1} to the interface, scaled by Δ. When dealing with ghost points significantly distant from the interface, i.e., when $\tau^G \geq \tau_{\partial\Omega}$, we employ the following approximation:

$$\phi_y = \frac{1}{\Delta}\left[-\frac{(2+\tau_G)\phi_f}{1+\tau^G} + \frac{(1+\tau^G)\phi_{r,j+1}}{\tau^G} - \frac{\phi_f}{\tau^G(1+\tau^G)}\right] \tag{22}$$

To prevent division by small quantities in (22), we set ϕ_y to 0 when $\tau^G < \tau_{\partial\Omega}$. A similar strategy can be employed when Ω is situated below the interface, where $\varphi_{r,j+1} \geq 0$ and $\varphi_{r,j-1} < 0$.

In cases where both $\varphi_{r,j-1} \geq 0$ and $\varphi_{r,j+1} \geq 0$, we assign ϕ_y a value of 0. Following the numerical approximations for ϕ_x, ϕ_y, φ_x, and φ_y at the interface, we apply the Stefan condition to compute $\mathcal{V} = -\frac{\kappa}{(m+1)}(\phi_x\varphi_x + \phi_y\varphi_y)$ and record the normal speed at each interface ghost node.

In summary, the process for calculating the velocity at a ghost node on the interface is as follows:

1. Identify whether the ghost node is positioned in the x or y direction between two grid points and establish which point (left or right/bottom or top) is located within $\Omega(t)$.

2. Calculate the first derivative of ϕ in the direction aligned with the interface ghost point's placement, utilizing a second-order accurate one-sided finite-difference method. This entails employing either (12) or (13), depending on the interface's location.

3. Estimate φ_x and φ_y at the ghost node through central differencing and linear interpolation, as described in (15) and (17).

4. Calculate the first derivative of u in the direction perpendicular to the grid using a second-order accurate one-sided difference. This process depends on the interface's position and requires the application of either (19) or (22).

5. After computing all the derivatives of ϕ and φ, record the ghost node's position and calculate the velocity as follows:

$$\mathcal{V} = -\frac{\kappa}{m+1}(\phi_x\varphi_x + \phi_y\varphi_y) \ .$$

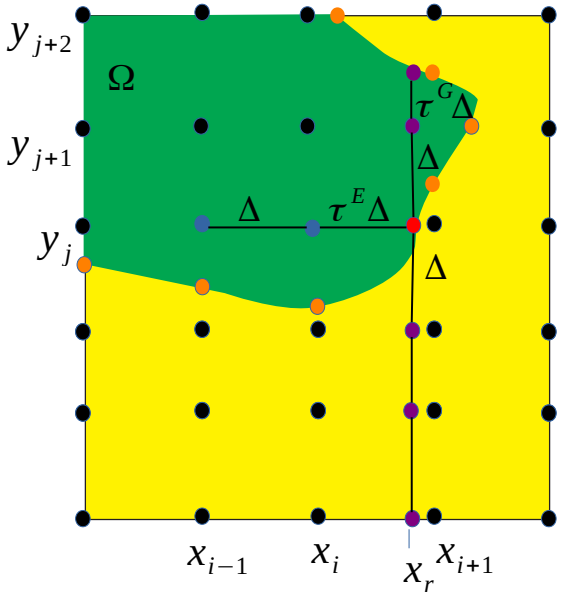


Figure 5: Finite-difference stencil utilized to calculate ϕ_y, with $\varphi_{r,j+1} < 0$ but $\varphi_{r,j+2} \geq 0$. The derivative is computed at the red ghost node, and pink nodes indicate the stencils used for computing ϕ_y and φ_y. Black and yellow dots represent nodes in the computational grid, while orange dots denote other ghost nodes on the interface.

2.3.2 Velocity Extrapolation Away From Interface

To derive the extension velocity field, defined throughout $\mathcal{D}$, we extrapolate $\mathcal{V}$ away from the interface in the orthogonal direction. This is achieved by solving the partial differential equation (PDE) [7]:

$$\frac{\partial F}{\partial \Upsilon} + \nabla F \cdot \hat{n} = 0, \tag{23}$$

subject to the boundary condition $F = \mathcal{V}$ on $\partial\Omega$, where $\hat{n}$ is the unit outward normal to the interface. To obtain F, we first solve (23) with the initial condition $F(x, 0) = 0$ everywhere except $\partial\Omega$. This extrapolates the velocity in the outward normal direction. We then use the solution to (23) as the initial condition and solve

$$\frac{\partial F}{\partial \Upsilon} + \nabla F \cdot (-\hat{n}) = 0, \tag{24}$$

to extrapolate in the inward normal direction. To solve (23) and (24) on $\mathcal{D}$, the unit outward normal vector on the interface can be extended to the entire domain via the level-set function, $\hat{n} = \nabla\varphi/|\nabla\varphi|$.

Thus, if $\nabla\varphi$ is known, we can rewrite both (23) and (24) as the linear advection equation

$$\frac{\partial F}{\partial \Upsilon} + a\frac{\partial F}{\partial x} + b\frac{\partial F}{\partial y} = 0. \tag{25}$$

To address 23, we determine $a = \varphi_x/|\nabla\varphi|$ and $b = \varphi_y/|\nabla\varphi|$. To solve 24 and extrapolate in the inwards normal direction, we employ $a = -\varphi_x/|\nabla\varphi|$ and $b = -\varphi_y/|\nabla\varphi|$.

The solution to 25 involves the method of lines. We approximate derivatives of x and y using first-order upwind finite differences. At interior grid points, we estimate derivatives of φ using central schemes:

$$[\varphi_x]_{i,j} = \frac{\varphi_{i+1,j} - \varphi_{i-1,j}}{2\Delta}, \quad [\varphi_y]_{i,j} = \frac{\varphi_{i,j+1} - \varphi_{i,j-1}}{2\Delta}. \tag{26}$$

Assuming $\partial\Omega$ is sufficiently distant from the domain boundaries, we omit boundary schemes for $[\varphi_x]_{i,j}$. For periodic boundaries $j = 0, N_y$, we apply

$$[\varphi_y]_{i,0} = [\varphi_y]_{i,N_y} = \frac{\varphi_{i,1} - \varphi_{i,N_y-1}}{2\Delta}. \tag{27}$$

Equations 26 and 27 suffice to determine the advection coefficients a and b in 25. Subsequently, employing the method of lines, we utilize the semi-discrete form:

$$\frac{\mathrm{d}\varphi_{i,j}}{\mathrm{d}\Upsilon} = a\left[\nabla_x F\right]_{i,j} + b\left[\nabla_y F\right]_{i,j}, \tag{28}$$

where ∇_x and ∇_y are first-order upwind approximations to the first derivatives. For regular grid points, $\nabla_x F$ takes the form:

$$[\nabla_x F]_{i,j} = \begin{cases} (F_{i+1,j} - F_{i,j})/\Delta & \text{if} \quad a < 0 \\ (F_{i,j} - F_{i-1,j})/\Delta & \text{if} \quad a \geq 0. \end{cases} \tag{29}$$

For irregular grid points, if the sign of a or b necessitates differencing across the interface, we construct these derivatives using the computed speed on the interface ghost node. For instance, if the interface lies east of grid point (x_i, y_j) and $a < 0$, then we use

$$[\nabla_x F]_{i,j} = \frac{\mathcal{V}_{i,j}^{i+1,j} - F_{i,j}}{\tau_{i,j}^{E}\Delta}. \tag{30}$$

If $\tau^E_{i,j} < \tau_{\partial\Omega}$, we set $F_{i,j} = \mathcal{V}^{i+1,j}_{i,j}$, and $[\nabla_x F]_{i,j} = [\nabla_y F]_{i,j} = 0$. Again, this is to prevent division by small quantities in 30. Other cases at irregular grid points are handled similarly. When solving 23 and 24, we assume that the interface $\partial\Omega$ sufficiently distant from the left and right boundaries of $\mathcal{D}$. Thus, we apply the Dirichlet boundary conditions $F_{0,j} = F_{N_x,j} = 0$. Where necessary, we apply periodic conditions at the upper and lower boundaries in the upwind schemes (analogous to 29 and 30, but for $[\nabla_y F]_{i,j}$). The implementation of periodic boundary conditions follows similar methods as outlined previously.

We integrate in 'pseudo-time' Υ is performed using DifferentialEquations.jl, employing the fifth-order Tsitouras Runge–Kutta method with an absolute tolerance of 1×10^{-3} and relative tolerance of 1×10^{-6}. For both equations 23 and 24, we compute solutions until $\Upsilon = 20\Delta\Upsilon$, where $\Delta\Upsilon = \Delta/5$. This ensures that the velocity field $F(x,t)$ is accurate in a neighbourhood of the interface. Consequently, the zero level-set of φ remains an implicit description of the interface position after solving the level-set equation.

2.4 Solving the Level-Set Equation

The level-set equation 4 can be expressed as a multidimensional Hamilton–Jacobi partial differential equation (PDE) given by

$$\frac{\partial\varphi}{\partial t} + H\left(\nabla\varphi\right) = 0, \tag{31}$$

where $H = F|\nabla\varphi|$ is the Hamiltonian. These equations share similarities with hyperbolic conservation laws, necessitating analogous numerical methods. We adopt the second-order scheme outlined by [8], which is then transformed into a non-staggered scheme through the averaging method introduced by [9]. The Lin–Tadmor method is derived from the Nessyahu–Tadmor scheme designed for hyperbolic PDEs [10]. This method, characterized as a Godunov-type central scheme, eliminates the need for solving an approximate Riemann problem and avoids upwind differencing.

The numerical approach presented by [8] is a fully-discrete method designed to solve 31 for φ^{k+1} on a staggered grid. This method employs subscripts outside the brackets to denote partial differentiation.

$$\begin{aligned}
\varphi^{k+1}_{i+1/2,j+1/2} = {} & \frac{1}{4}\left(\varphi^k_{i,j} + \varphi^k_{i+1,j} + \varphi^k_{i,j+1} + \varphi^k_{i+1,j+1}\right) \\
& + \frac{\Delta}{16}\left[\left(\varphi^k_{i,j}\right)_x - \left(\varphi^k_{i+1,j}\right)_x + \left(\varphi^k_{i,j+1}\right)_x - \left(\varphi^k_{i+1,j+1}\right)_x\right] \\
& + \frac{\Delta}{16}\left[\left(\varphi^k_{i,j}\right)_y - \left(\varphi^k_{i,j+1}\right)_y + \left(\varphi^k_{i+1,j}\right)_y - \left(\varphi^k_{i+1,j+1}\right)_y\right] \\
& - \frac{\Delta t}{2}\left[H\left(\frac{\varphi^{k+1/2}_{i+1,j} - \varphi^{k+1/2}_{i,j}}{\Delta}, \frac{\varphi^{k+1/2}_{i+1,j+1} - \varphi^{k+1/2}_{i+1,j}}{\Delta}\right)\right. \\
& \left. + H\left(\frac{\varphi^{k+1/2}_{i+1,j+1} - \varphi^{k+1/2}_{i,j+1}}{\Delta}, \frac{\varphi^{k+1/2}_{i,j+1} - \varphi^{k+1/2}_{i,j}}{\Delta}\right)\right],
\end{aligned} \tag{32}$$

where $(x_{i+1/2}, y_{j+1/2}) = (x_i + \Delta/2, y_j + \Delta/2)$ for $i = 0, \ldots N_x - 1$ and $j = 0, \ldots N_y - 1$ defines the staggered grid. Computing 32 involves evaluations of H at half-time-points

$t_{k+1/2} = t_k + \Delta t/2$, and discrete derivatives of φ at non-staggered grid points (denoted by the subscripts x and y). The missing mid-values in time are obtained by Taylor expansion about t_k, yielding

$$\varphi_{i,j}^{k+1/2} = \varphi_{i,j}^{k} - \frac{\Delta t}{2} H\left[\left(\varphi_{i,j}^{k}\right)_x, \left(\varphi_{i,j}^{k}\right)_y\right]. \tag{33}$$

The derivatives are calculated employing an appropriate flux limiter to guarantee the non-oscillatory nature of the scheme, as proposed by [8]. Specifically, we adopt the min-mod limiter [8], such that

$$\left(\varphi_{i,j}^{k}\right)_x = MM\left(\Upsilon\frac{\varphi_{i+1,j}^{k} - \varphi_{i,j}^{k}}{\Delta}, \frac{\varphi_{i+1,j}^{k} - \varphi_{i-1,j}^{k}}{2\Delta}, \Upsilon\frac{\varphi_{i,j}^{k} - \varphi_{i-1,j}^{k}}{\Delta}\right) \tag{34a}$$

$$\left(\varphi_{i,j}^{k}\right)_y = MM\left(\Upsilon\frac{\varphi_{i,j+1}^{k} - \varphi_{i,j}^{k}}{\Delta}, \frac{\varphi_{i,j+1}^{k} - \varphi_{i,j-1}^{k}}{2\Delta}, \Upsilon\frac{\varphi_{i,j}^{k} - \varphi_{i,j-1}^{k}}{\Delta}\right), \tag{34b}$$

for some $\Upsilon \in [1, 2]$. In our scheme, we use $\Upsilon = 1.99$ [11], indicating a preference for central differencing where possible. The symbol MM denotes the min-mod function, which applied to a vector v_i is

$$MM(v_i) = \begin{cases} \min_i \{v_i\} & \text{if} \quad v_i > 0 \quad \forall i \\ \max_i \{v_i\} & \text{if} \quad v_i < 0 \quad \forall i \\ 0 & \text{otherwise.} \end{cases} \tag{35}$$

After computing the staggered solution using 32, the conversion to a non-staggered scheme can be achieved through averaging, as outlined in [9],

$$\begin{aligned} \varphi_{i,j}^{k+1} = {} & \frac{1}{4}\left(\varphi_{i+1/2,j+1/2}^{k+1} + \varphi_{i-1/2,j+1/2}^{k+1} + \varphi_{i-1/2,j-1/2}^{k+1} + \varphi_{i+1/2,j-1/2}^{k+1}\right) \\ & + \frac{\Delta}{16}\left[\left(\varphi_{i-1/2,j-1/2}^{k+1}\right)_x - \left(\varphi_{i+1/2,j-1/2}^{k+1}\right)_x + \left(\varphi_{i-1/2,j+1/2}^{k+1}\right)_x - \left(\varphi_{i+1/2,j+1/2}^{k+1}\right)_x\right] \\ & + \frac{\Delta}{16}\left[\left(\varphi_{i-1/2,j-1/2}^{k+1}\right)_y - \left(\varphi_{i-1/2,j+1/2}^{k+1}\right)_y + \left(\varphi_{i+1/2,j-1/2}^{k+1}\right)_y - \left(\varphi_{i+1/2,j+1/2}^{k+1}\right)_y\right], \end{aligned} \tag{36}$$

where again, we compute the relevant derivatives using the min-mod limiter 34. Combining 32 with 36, we define a scheme to determine $\varphi_{i,j}^{k+1}$ given $\varphi_{i,j}^{k}$. Although the level-set equation 4 technically requires no boundary conditions, when solving 31, we impose Dirichlet boundary conditions on the left and right boundaries $\varphi_{0,j}^{k+1} = \varphi_{0,j}^{k}$, $\varphi_{N_x,j}^{k+1} = \varphi_{N_x,j}^{k}$, for all k. This is appropriate, since we assume the interface is distant from the left and right boundaries of $\mathcal{D}$. If needed, periodic conditions are applied in the y-direction at upper and lower boundaries. These periodic conditions are pertinent for all central difference formulas involving φ_y in 32, 33, 34 and 36. Their implementation closely follows the methods outlined previously.

2.5 Level-Set Function Reinitialisation

Employing orthogonal extrapolation to acquire the extension velocity field fails to uphold the signed-distance property of the level-set function, φ [2]. Hence, we introduce

a method to modify the level-set function, restoring the signed-distance property. We achieve this by solving the reinitialisation equation [1], which is the PDE

$$\frac{\partial \varphi}{\partial \Upsilon} + S(\varphi)\,(|\nabla\varphi|-1) = 0, \tag{37}$$

where $S(\varphi)$ is a modified sign function defined as

$$S(\varphi) = \frac{\varphi}{\sqrt{|\nabla\varphi|^2+\Delta^2}}. \tag{38}$$

Similar to the level-set equation 4, the reinitialisation equation 37 is a Hamilton–Jacobi equation. It takes the form $\varphi_\Upsilon + H(\varphi, \nabla\varphi) = 0$, with $H = S(\varphi)(|\nabla\varphi|-1)$. Consequently, we adapt the Lin–Tadmor method 32, 33, 34, and 36, originally employed for solving the level-set equation, to address 37. In instances requiring the assessment of H, we use $\varphi^k_{i,j}$ instead of values at the half-time steps. In our solutions, we reinitialize the signed-distance function at every time-step, conducting 20 Υ-steps, with $\Delta\Upsilon = \Delta/5$.

References

[1] S. Osher, *Level Set Methods and Dynamic Implicit Surfaces*, 1st ed., ser. Applied Mathematical Sciences, 153. New York, NY: Springer New York, 2003.

[2] J. A. Sethian, *Level set methods and fast marching methods : evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, 2nd ed., ser. Cambridge monographs on applied and computational mathematics ; 3. Cambridge, U.K. ;: Cambridge University Press, 1999.

[3] A. K. Tam and M. J. Simpson, "The effect of geometry on survival and extinction in a moving-boundary problem motivated by the fisher–kpp equation," *Physica. D*, vol. 438, p. 133305, 2022.

[4] L. C. MORROW, T. J. MORONEY, M. C. DALLASTON, and S. W. MCCUE, "A review of one-phase hele-shaw flows and a level-set method for nonstandard configurations," *The ANZIAM journal*, vol. 63, no. 3, pp. 269–307, 2021.

[5] C. Tsitouras, "Runge–kutta pairs of order 5(4) satisfying only the first column simplifying assumption," *Computers and mathematics with applications (1987)*, vol. 62, no. 2, pp. 770–775, 2011.

[6] C. Rackauckas and Q. Nie, "Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia," *Journal of open research software*, vol. 5, no. 1, 2017.

[7] T. D. Aslam, "A partial differential equation approach to multidimensional extrapolation," *Journal of computational physics*, vol. 193, no. 1, pp. 349–355, 2004.

[8] C.-T. LIN and E. TADMOR, "High-resolution nonoscillatory central schemes for hamilton-jacobi equations," *SIAM journal on scientific computing*, vol. 21, no. 6, pp. 2163–2186, 2000.

[9] G.-S. Jiang, D. Levy, C.-T. Lin, S. Osher, and E. Tadmor, "High-resolution nonoscillatory central schemes with nonstaggered grids for hyperbolic conservation laws," *SIAM journal on numerical analysis*, vol. 35, no. 6, pp. 2147–2168, 1998.

[10] H. Nessyahu and E. Tadmor, "Non-oscillatory central differencing for hyperbolic conservation laws," *Journal of computational physics*, vol. 87, no. 2, pp. 408–463, 1990.

[11] M. J. Simpson, K. A. Landman, and T. Clement, "Assessment of a non-traditional operator split algorithm for simulation of reactive transport," *Mathematics and computers in simulation*, vol. 70, no. 1, pp. 44–60, 2005.