Level-Set Method for the 2D Fisher–Stefan Model

Alexander K. Y. Tam[*1,2] **and Matthew J. Simpson**[2]

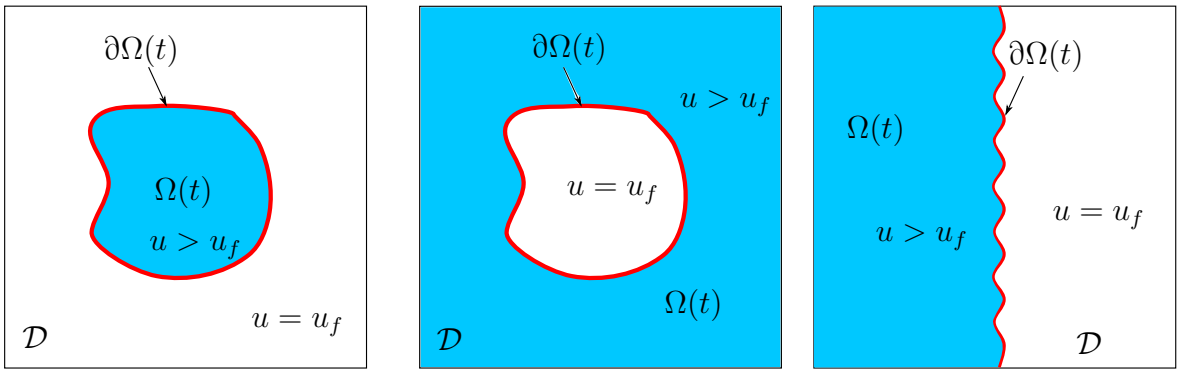
[1]UniSA STEM, The University of South Australia, Mawson Lakes SA 5095, Australia.
[2]School of Mathematical Sciences, Queensland University of Technology, Brisbane QLD 4000, Australia

June 5, 2022

In this document, we describe a level-set numerical method to solve the Fisher–Stefan model in two spatial dimensions. We have implemented the method in Julia, and code is publicly available on GitHub.

1 Fisher–Stefan Model and Level-Set Method

We implement the level-set method [1, 2] to compute numerical solutions to the two-dimensional Fisher–Stefan model. The Fisher–Stefan model is a moving-boundary problem that involves solving the Fisher–Kolmogorov–Petrovsky–Piskunov (Fisher–KPP) equation on a two-dimensional region $\Omega(t)$, subject to Dirichlet boundary conditions on the boundary $\partial\Omega(t)$. The boundary $\partial\Omega(t)$ corresponds to the interface between a population that diffuses and reacts, and an inactive background population of density u_f, where $0 \leq u_\mathrm{f} < 1$. This interface evolves with time according to a Stefan-like condition. The region $\Omega(t)$ is contained within a two-dimensional computational domain $\mathcal{D}$. Three example configurations of $\Omega(t)$, $\partial\Omega(t)$, and $\mathcal{D}$ are illustrated in Figure 1.1.



(a) Domain and interface for a survival/extinction problem.

(b) Domain and interface for a hole-closing problem.

(c) Domain and interface for a front propagation problem.

Figure 1.1: The two-dimensional computational domain for solving the Fisher–Stefan model using the level-set method. The density $u(\boldsymbol{x}, t)$ is defined in the region $\Omega(t)$ (shaded in blue), on which we solve the 2D Fisher–KPP equation. The boundary of this region, $\partial\Omega(t)$, evolves according to a Stefan-like condition.

*Contact: alex.tam@unisa.edu.au

We consider the multidimensional Fisher–Stefan model with surface-tension regularisation,

$$\frac{\partial u}{\partial t} = D\nabla^2 u + \lambda\,(1-u) \quad \text{on} \quad \Omega(t), \tag{1.1a}$$

$$u(\boldsymbol{x},t) = u_{\mathrm{f}} - \gamma K \quad \text{on} \quad \partial\Omega(t), \tag{1.1b}$$

$$V = -\kappa \nabla u \cdot \hat{\boldsymbol{n}} \quad \text{on} \quad \partial\Omega(t), \tag{1.1c}$$

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \quad \text{on} \quad \Omega(0), \tag{1.1d}$$

where $\boldsymbol{x}$ and t denote space and time, respectively. In (1.1), $u(\boldsymbol{x},t)$ represents population density, D is the diffusivity, λ is the reaction rate, $u_{\mathrm{f}} \in [0,1)$ is the background population density, γ is the surface-tension coefficient, K is the local boundary curvature, V is the normal speed of the boundary, κ is the inverse Stefan number, $\hat{\boldsymbol{n}}$ is the unit normal vector to the boundary, and $u_0(\boldsymbol{x})$ is a function specifying the initial condition.

The level-set method enables the numerical solution of moving-boundary problems on fixed computational domains. The method avoids the need to parameterise the domain $\Omega(t)$, which can take arbitrary, even non-simply-connected, shapes. The key feature of the level-set method is embedding the boundary position as the zero of a scalar *level-set function* $\phi(\boldsymbol{x},t)$. That is,

$$\partial\Omega(t) = \{\boldsymbol{x} \mid \phi(\boldsymbol{x},t) = 0\}, \tag{1.2}$$

where $\phi(\boldsymbol{x},t)$ is defined everywhere in the computational domain $\mathcal{D}$. The level-set function $\phi(\boldsymbol{x},t)$ describes the position of $\partial\Omega(t)$ implicitly, because the zero level-set of ϕ evolves according to the level-set equation

$$\frac{\partial \phi}{\partial t} + F\,|\nabla\phi| = 0, \tag{1.3}$$

where $F(\boldsymbol{x},t)$ is a scalar function defined on $\mathcal{D}$, such that it gives the normal speed V on $\partial\Omega(t)$, $F = V$. After introducing the level-set function, the numerical problem to solve becomes

$$\frac{\partial u}{\partial t} = D\nabla^2 u + \lambda\,(1-u) \quad \text{on} \quad \Omega(t), \tag{1.4a}$$

$$\frac{\partial \phi}{\partial t} + F\,|\nabla\phi| = 0 \quad \text{on} \quad \mathcal{D}, \tag{1.4b}$$

$$u(\boldsymbol{x},t) = u_{\mathrm{f}} - \gamma K \quad \text{on} \quad \partial\Omega(t), \tag{1.4c}$$

$$F = -\kappa \nabla u \cdot \hat{\boldsymbol{n}} \quad \text{on} \quad \partial\Omega(t), \tag{1.4d}$$

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \quad \text{on} \quad \Omega(0). \tag{1.4e}$$

The level-set function is usually chosen to be the signed-distance from the boundary [2], such that $\phi(\boldsymbol{x},t) < 0$ at all points $\boldsymbol{x} \in \Omega(t)$, $\phi(\boldsymbol{x},t) \geq 0$ at all points $\boldsymbol{x} \notin \Omega(t)$, and

$\phi(\boldsymbol{x}, t) = 0$ at all points $\boldsymbol{x} \in \partial\Omega(t)$. In this document, we consider (1.4) on a two-dimensional computational domain $\mathcal{D}$, subject to Dirichlet boundary conditions on the left and right boundaries, and periodic boundary conditions on the top and bottom boundaries. These boundary conditions are suitable for all three configurations in Figure 1.1.

2 Numerical Algorithm

We solve (1.4) on the two-dimensional computational domain $\mathcal{D} = [0, L_x] \times [0, L_y]$, where L_x and L_y are the widths in the x and y-directions respectively. We compute the solution on an equispaced discrete grid (x_i, y_j). The discrete grid points (or *nodes*) are given by $x_i = i\Delta x$, and $y_j = j\Delta y$, for $i = 0, \dots N_x$, and $j = 0, \dots N_y$, where N_x and N_y are the number of grid intervals in the x and y-directions respectively, and Δx and Δy are the (constant) grid spacing in the respective direction. For the temporal domain, we define $t_k = k\Delta t$, where $k = 0, \dots N_t$, and Δt is constant. The symbols $u^k_{i,j} = u(x_i, y_j, t_k)$, $\phi^k_{i,j} = \phi(x_i, y_j, t_k)$, and $F_{i,j} = F(x_i, y_j)$, then denote the value of each quantity at discrete points in time and space.

The procedure to solve the Fisher–Stefan model using the level-set method consists of the following steps:

1. Initialise the problem by defining the model parameters D, λ, γ, and κ, and initial conditions $u_0(\boldsymbol{x})$ and $\phi_0(\boldsymbol{x})$.

2. Advance in time to solve for $u^{k+1}_{i,j}$, $\phi^{k+1}_{i,j}$, and $F^{k+1}_{i,j}$, for $i = 0, \dots N_x$, $j = 0, \dots N_y$, and $k = 0, \dots, N_t - 1$. At each time step, perform the following:

   (a) Identify the region $\Omega(t)$, and the interface position, $\partial\Omega(t)$.

   (b) Solve the 2D Fisher–KPP equation (1.4a) on $\Omega(t)$ to obtain $u^{k+1}_{i,j}$, subject to the boundary condition (1.4c).

   (c) Compute the interface velocity, V, using the Stefan condition (1.4d). Subsequently, extend the velocity field to obtain a speed function $F(\boldsymbol{x}, t)$ defined for all $\boldsymbol{x} \in \mathcal{D}$.

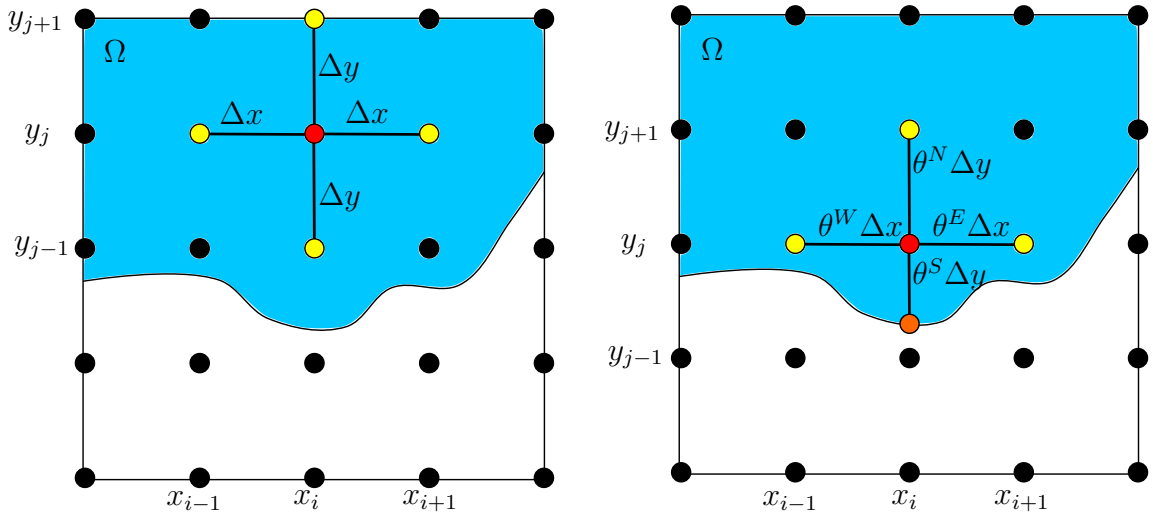
   (d) Solve the level-set equation (1.4b) to obtain $\phi^{k+1}_{i,j}$, which updates the interface position.

   (e) (Optional) Reinitialise the level-set function to a signed-distance function in an appropriate neighbourhood of the interface.

We implement this procedure using code developed in Julia. A detailed description of the methods used for steps 2.(a)–(e) is provided in subsequent subsections.

2.1 Identifying the Domain and Interface

After specifying the parameters and initial conditions, the first step is to identify the domain $\Omega(t)$ and interface $\partial\Omega(t)$. To locate $\Omega(t)$, we classify all interior grid points (x_i, y_j), for $i = 1, \dots N_x - 1$, and $j = 0, \dots, N_y$, as either inside or outside $\Omega(t)$, such that $(x_i, y_j) \in \Omega(t)$ if $\phi(x_i, y_j, t) < 0$, and $(x_i, y_j) \notin \Omega(t)$. To locate the interface $\partial\Omega(t)$, we identify all grid points (x_i, y_j) such that ϕ at any adjacent grid point has the opposite sign. We refer to these as *irregular* grid points (and points where ϕ has the same sign at all adjacent nodes as *regular* grid points).



(a) Regular grid point.

(b) Irregular grid point.

Figure 2.1: Regular and irregular grid points in the computational domain. Black dots represent grid points, the red dot is the point (x_i, y_j), and yellow dots are nodes adjacent to (x_i, y_j) that are in Ω. For the irregular point in (b), the orange dot represents a ghost node on the interface $\partial\Omega$, in the southern direction from (x_i, y_j).

Figure 2.1 illustrates the difference between regular and irregular grid points. In Figure 2.1a, the red node is a regular grid point, because $\phi_{i,j} < 0$, and $\phi < 0$ at all adjacent (yellow) nodes. However, in Figure 2.1b the red node is irregular, because its southern neighbouring node (x_i, y_{j-1}) is outside $\Omega(t)$. To distinguish between regular and irregular grid points, we define $\theta^W_{i,j}$, $\theta^E_{i,j}$, $\theta^S_{i,j}$, and $\theta^N_{i,j}$ to be the distance (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 $j \neq 0$,

$$\theta^S_{i,j} = \begin{cases} 1 & \text{if} \quad \operatorname{sgn}(\phi_{i,j}) = \operatorname{sgn}(\phi_{i,j-1}) \\ \dfrac{\phi_{i,j}}{\phi_{i,j} - \phi_{i,j-1}} & \text{otherwise,} \end{cases} \tag{2.1}$$

where we apply linear interpolation at irregular grid points to find the zero level-set ($\phi = 0$) if the interface passes between (x_i, y_j) and its southern neighbour. If so, the interface ghost

node (orange dot in Figure 2.1b) (zero level-set) position is (x_i, y^G), where $y^G = y_j - \theta^S_{i,j}\Delta y$. If $j = 0$, then we modify (2.1) to account for the periodic boundary, and have

$$\theta^S_{i,0} = \begin{cases} 1 & \text{if} \quad \operatorname{sgn}(\phi_{i,0}) = \operatorname{sgn}(\phi_{i,N_y-1}) \\ \dfrac{\phi_{i,0}}{\phi_{i,0} - \phi_{i,N_y-1}} & \text{otherwise.} \end{cases} \tag{2.2}$$

Similar interpolation formulae can be developed for $\theta^W_{i,j}$, $\theta^E_{i,j}$, and $\theta^N_{i,j}$ as necessary, and we store their values at each interior grid point in $\mathcal{D}$. These enable development of finite-difference methods for the Fisher–KPP equation and extension velocity field that respect the shape of Ω and the interface location. Throughout the development of our numerical methods, we assume that the interface exists at a maximum of one location between adjacent grid points. That is, there is a maximum of one position between adjacent grid points where $\phi = 0$. If Ω and the grid spacing are such that there are multiple interface locations, we refine the grid until Δx and Δy are sufficiently small for only one to remain.

2.2 2D Fisher–KPP Equation

We solve the 2D Fisher–KPP equation using the explicit finite-difference method of lines. At each interior grid point in $\mathcal{D}$, we discretise the right-hand side of (1.4a) using second-order accurate finite-difference approximations for the derivatives. This yields a semi-discrete system of ordinary differential equations (ODEs) of the form

$$\frac{\mathrm{d}u_{i,j}}{\mathrm{d}t} = \begin{cases} D\left(\nabla^2_d u_{i,j}\right) + \lambda u_{i,j}\left(1 - u_{i,j}\right) & \text{if} \quad \phi(x_i, y_j) < 0 \\ 0 & \text{if} \quad \phi(x_i, y_j) \geq 0, \end{cases} \tag{2.3}$$

for $i = 0, \ldots, N_x$, and $j = 0, \ldots, N_y$. We apply Dirichlet boundary conditions at nodes that lie on the left and right boundaries of the computational domain. For survival/extinction problems (Figure 1.1a), we use the Dirichlet conditions $u_{0,j}(t) = u_{N_x,j}(t) = u_\mathrm{f}$, for hole-closing problems (Figure 1.1b) we set $u_{0,j}(t) = u_{N_x,j}(t) = 1$, and for front propagation problems (Figure 1.1c) we set $u_{0,j}(t) = 1$ and $u_{N_x,j}(t) = u_\mathrm{f}$. The term $\nabla^2_d u_{i,j}$ denotes the second-order central finite-difference approximation for the Laplacian, $\nabla^2 u = u_{xx} + u_{yy}$.

The finite-difference discretisation of $\nabla^2 u$ differs depending on whether (x_i, y_j) is regular or irregular. At regular grid points, we apply the standard five-point central finite-difference stencil,

$$\nabla^2_d u_{i,j} = \frac{u^E_{i,j} - 2u_{i,j} + u^W_{i,j}}{(\Delta x)^2} + \frac{u^N_{i,j} - 2u_{i,j} + u^S_{i,j}}{(\Delta y)^2}. \tag{2.4}$$

In (2.4), we have introduced the superscript notation $u^A_{i,j}$, $A \in \{W, E, S, N\}$, to denote the value of u at the western, eastern, southern, and northern neighbours to (x_i, y_j) respectively.

However, we cannot use (2.4) at irregular grid points, because one or more of the grid points on the standard stencil lies outside Ω. Instead, we introduce ghost nodes on the interface (for example the orange dot in the southern direction in Figure 2.1b) to replace nodes outside Ω that would otherwise be in the standard stencil. The density at these ghost nodes adheres to the interface condition (1.4c), $u = u_{\mathrm{f}} - \gamma K$. With our definitions of $\theta^A_{i,j}$, $A \in \{W, E, S, N\}$ above, this gives rise to

$$\begin{aligned}\nabla_d^2 u_{i,j} = &\frac{2}{(\Delta x)^2}\left[\frac{u^E_{i,j}}{\theta^E_{i,j}(\theta^W_{i,j}+\theta^E_{i,j})} - \frac{u_{i,j}}{\theta^W_{i,j}\theta^E_{i,j}} + \frac{u^W_{i,j}}{\theta^W_{i,j}(\theta^W_{i,j}+\theta^E_{i,j})}\right] \\ &+ \frac{2}{(\Delta y)^2}\left[\frac{u^N_{i,j}}{\theta^N_{i,j}(\theta^S_{i,j}+\theta^N_{i,j})} - \frac{u_{i,j}}{\theta^S_{i,j}\theta^N_{i,j}} + \frac{u^S_{i,j}}{\theta^S_{i,j}(\theta^S_{i,j}+\theta^N_{i,j})}\right].\end{aligned} \tag{2.5}$$

If $\theta^A_{i,j} = 1$ for all $A \in \{N, E, S, W\}$ as for regular grid points, then the formula for the irregular stencil (2.5) is the same as the regular stencil formula (2.4). The approximation (2.5) can involve division by small quantities if any $\theta^A_{i,j}$ ($A \in \{N, E, S, W\}$) is small. If so, small time-steps would be required in the method of lines to retain numerical stability. Following Morrow et al. [3], we circumvent this difficulty by setting $u_{i,j} = u_{\mathrm{f}} - \gamma K$ if any $\theta^A_{i,j} < \theta_{\partial\Omega}$, for some threshold $\theta_{\partial\Omega}$. In all computed solutions, we use $\theta_{\partial\Omega} = 0.01$.

The values $u^W_{i,j}$, $u^E_{i,j}$, $u^S_{i,j}$, and $u^N_{i,j}$ in (2.4) and (2.5) depend on whether the neighbouring nodes lie inside $\Omega(t)$, and whether the neighbouring nodes involve crossing the periodic boundaries. For example, we have

$$u^W_{i,j} = \begin{cases} u_{i-1,j} & \text{if} \quad \phi_{i-1,j} < 0 \\ u_{\mathrm{f}} - \gamma K^{i,j}_{i-1,j} & \text{if} \quad \phi_{i-1,j} \geq 0, \end{cases} \tag{2.6}$$

and

$$u^S_{i,j} = \begin{cases} u_{i,j-1} & \text{if} \quad \phi_{i,j-1} < 0, j \neq 0 \\ u_{\mathrm{f}} - \gamma K^{i,j}_{i,j-1} & \text{if} \quad \phi_{i,j-1} \geq 0, j \neq 0, \end{cases} \quad u^S_{i,0} = \begin{cases} u_{i,N_y-1} & \text{if} \quad \phi_{i,N_y-1} < 0 \\ u_{\mathrm{f}} - \gamma K^{i,0}_{i,N_y-1} & \text{if} \quad \phi_{i,N_y-1} \geq 0, \end{cases} \tag{2.7}$$

and similar for $u^E_{i,j}$ and $u^N_{i,j}$. In (2.6) and (2.7), we have introduced new notation for the interface curvature, K, at points on the interface. For example, the symbol $K^{i,j}_{i-1,j}$ in (2.6) refers to the curvature at the ghost node on the interface (such as the orange point in Figure 2.1b) that lies between the grid points (x_i, y_{j-1}) and (x_i, y_j). We outline the method to compute these local curvature values in §2.2.1.

After constructing the system of ODEs (2.3) at all interior grid points in the computational domain, we integrate in time using the fifth-order Tsitouras Runge–Kutta method [4]. The calculation is performed using the in-built ODE solver in `DifferentialEquations.jl` [5]. At each time step in our algorithm, we solve to advance in time by an increment of Δt. Depending on the numerical stability of (2.3), the solver might take multiple intermediate

time steps. We retain this adaptive time-stepping for accuracy, and apply the absolute tolerance of 1×10^{-3} and relative tolerance of 1×10^{-6}.

2.2.1 Local Interface Curvature

To implement the finite-difference approximation for irregular grid points (2.5), we require a method to compute the local curvature, K, of the interface. In terms of the level-set function, ϕ, the local curvature is

$$K = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_y\phi_x\phi_{xy} + \phi_{yy}\phi_x^2}{\left(\phi_x^2 + \phi_y^2\right)^{3/2}}, \tag{2.8}$$

where subscripts denote partial differentiation. Consider an irregular grid point with an interface ghost node that lies southward, as per Figure 2.1b. Using (2.8), we approximate the curvature at that ghost node using

$$K_{i,j-1}^{i,j} = \left[\frac{\phi_{xx}\phi_y^2 - 2\phi_y\phi_x\phi_{xy} + \phi_{yy}\phi_x^2}{\left(\phi_x^2 + \phi_y^2\right)^{3/2}} \right]_{i,j-1}^{i,j}, \tag{2.9}$$

and similar formulae for $K_{i,j}^{i-1,j}$, $K_{i,j}^{i,j+1}$, and $K_{i,j}^{i,j-1}$ where applicable. In this Appendix, we describe the procedure for an interface ghost node to the south of a reference grid node. Analogous formulae can be developed if the interface lies to the north, west, or east of the reference node.

Implementing (2.9) requires finite-difference approximations for the first and second derivatives of the level-set function, ϕ, at the interface ghost node. To compute ϕ_x, we first interpolate linearly to approximate ϕ one grid point directly to the left and right of the interface ghost node, using

$$\phi_{i-1,j-1}^{i-1,j} = \theta_{i,j}^S \phi_{i-1,j-1} + \left(1 - \theta_{i,j}^S\right) \phi_{i-1,j}, \tag{2.10a}$$

$$\phi_{i+1,j-1}^{i+1,j} = \theta_{i,j}^S \phi_{i+1,j-1} + \left(1 - \theta_{i,j}^S\right) \phi_{i+1,j}. \tag{2.10b}$$

Like the subscript and superscript notation for curvature K, the symbol $\phi_{i-1,j-1}^{i-1,j}$ is the estimate of ϕ at a relevant ghost node between grid points (x_{i-1}, y_{j-1}), and (x_{i-1}, y_j). We then use the standard central difference to obtain

$$[\phi_x]_{i,j-1}^{i,j} = \frac{\phi_{i+1,j-1}^{i+1,j} - \phi_{i-1,j-1}^{i-1,j}}{2\Delta x}. \tag{2.11}$$

In our solutions, we do not allow the interface to approach the left and right boundaries, such that boundary schemes are not needed when finding ϕ_x. To approximate ϕ_y, we first

apply central difference to obtain estimates at grid points,

$$[\phi_y]_{i,j} = \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y}, \tag{2.12}$$

for $j = 1, \ldots, N_y - 1$. If necessary, we alternatively use

$$[\phi_y]_{i,0} = [\phi_y]_{i,N_y} = \frac{\phi_{i,1} - \phi_{i,N_y-1}}{2\Delta y} \tag{2.13}$$

to apply the periodic boundary conditions. We then interpolate to estimate ϕ_y at the ghost node. This gives

$$[\phi_y]^{i,j}_{i,j-1} = \theta^S_{i,j} \, [\phi_y]_{i,j-1} + \left(1 - \theta^S_{i,j}\right) [\phi_y]_{i,j} \, . \tag{2.14}$$

The next step is to approximate second derivatives ϕ_{xx} and ϕ_{yy}, for which we also use second-order central differences. We have

$$[\phi_{xx}]^{i,j}_{i,j-1} = \frac{\phi^{i+1,j}_{i+1,j-1} - 2\phi^{i,j}_{i,j-1} + \phi^{i-1,j}_{i-1,j-1}}{(\Delta x)^2}, \tag{2.15}$$

where $\phi^{i,j}_{i,j-1} = 0$ at the interface ghost node, and $\phi^{i-1,j}_{i-1,j-1}$ and $\phi^{i+1,j}_{i+1,j-1}$ are given by (2.10). To approximate ϕ_{yy}, we apply

$$[\phi_{yy}]_{i,j} = \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{(\Delta y)^2}, \tag{2.16}$$

for $j = 1, \ldots, N_y - 1$. If necessary, we alternatively use

$$[\phi_{yy}]_{i,0} = \frac{\phi_{i,1} - 2\phi_{i,0} + \phi_{i,N_y-1}}{(\Delta y)^2} \tag{2.17}$$

to apply the periodic boundary conditions. We then interpolate to obtain

$$[\phi_{yy}]^{i,j}_{i,j-1} = \theta^S_{i,j} \, [\phi_{yy}]_{i-1,j-1} + \left(1 - \theta^S_{i,j}\right) [\phi_{yy}]_{i-1,j} \, , \tag{2.18}$$

as per the method to find ϕ_y.

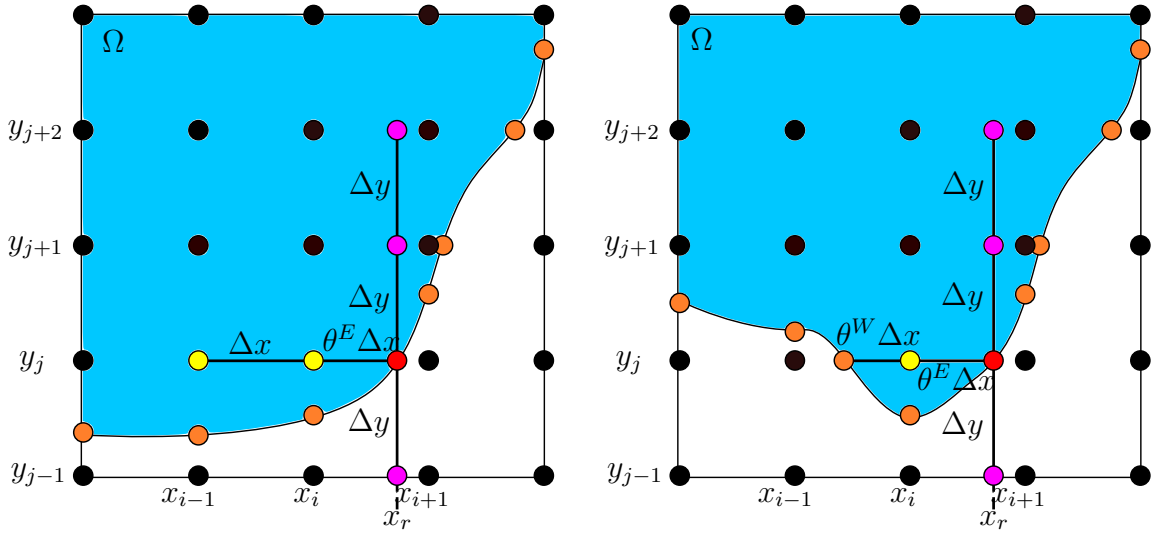
Lastly, we require a discretisation for ϕ_{xy}. We approximate ϕ_y at the four grid points $(x_{i-1,j})$, $(x_{i-1,j-1})$, $(x_{i+1,j})$, and $(x_{i+1,j-1})$, using second-order central differences (2.12), or a second-order boundary scheme such as (2.13) if applicable. We then interpolate linearly to approximate one grid point directly to the left and right of the interface ghost node (*i.e.* $[\phi_y]^{i-1,j}_{i-1,j-1}$ and $[\phi_y]^{i+1,j}_{i+1,j-1}$), using (2.14). We then apply the standard central difference in the x-direction to obtain

$$[\phi_{xy}]^{i,j}_{i,j-1} = \frac{[\phi_y]^{i+1,j}_{i+1,j-1} - [\phi_y]^{i-1,j}_{i-1,j-1}}{2\Delta x}. \tag{2.19}$$

The formula (2.19) then provides an estimate for ϕ_{xy} at the interface ghost node that lies south of the reference node. We develop similar formulae to (2.19) for interface ghost nodes that lie in the northern, eastern, and western directions from the reference node. The formulae for east and west interface ghost nodes involve first approximating ϕ_x analogously to (2.14), and then applying central difference in the y-direction to obtain ϕ_{xy}.

2.3 Constructing the Extension Velocity Field

We apply the Stefan condition at all ghost nodes on the interface (orange nodes in Figure 2.1 and subsequent). To apply (1.4d), we implement second-order accurate finite-differences for u_x, u_y, ϕ_x, and ϕ_y. The finite-difference scheme depends on the location of the interface and the adjacent nodes. Without loss of generality, here we describe the procedure used if the ghost node lies in the x-direction between two grid points, and if the region $\Omega(t)$ lies directly west and north of the ghost node (see Figure 2.2). Although this is not the only possible configuration, where necessary different approximations can be developed similarly. For example, if Ω lies in the eastern rather than western direction, we apply forward differencing to compute u_x instead of the backward differences developed here.



(a) Regular. (b) Irregular.

Figure 2.2: Finite-difference stencils used to compute the interface normal speed for grid points with $\theta^E_{i,j} > \theta_{\partial\Omega}$. Black dots denote nodes of the computational grid. The red dots denote the ghost node at which we compute V using the Stefan condition, and orange dots denote other ghost nodes on the interface. Yellow nodes denote grid points used in the stencils for $[u_x]^{i+1,j}_{i,j}$, and pink dots denote ghost nodes used in the stencils for $[u_x]^{i+1,j}_{i,j}$.

To approximate $[u_x]^{i+1,j}_{i,j}$ at the interface ghost node between (x_i, y_j) and (x_{i+1}, y_j) (red dot in Figure 2.2), we apply second-order accurate finite-difference approximations. If the grid point to the left of the ghost node is in Ω, as in Figure 2.2, the finite-difference

formula is

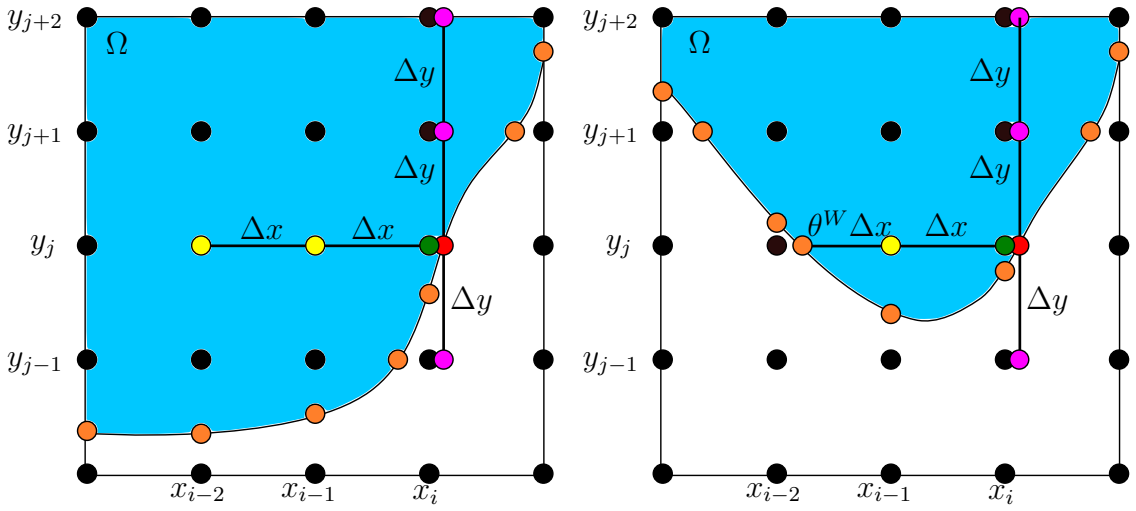
$$[u_x]_{i,j}^{i+1,j} = \frac{1}{\Delta x}\left[\frac{(2\theta_{i,j}^E + \theta_{i,j}^W)(u_\mathrm{f} - \gamma K_{i,j}^{i+1,j})}{\theta_{i,j}^E(\theta_{i,j}^W + \theta_{i,j}^E)} - \frac{(\theta_{i,j}^W + \theta_{i,j}^E)u_{i,j}}{\theta_{i,j}^W\theta_{i,j}^E} + \frac{\theta_{i,j}^E u_{i,j}^W}{\theta_{i,j}^W(\theta_{i,j}^W + \theta_{i,j}^E)}\right]. \quad (2.20)$$

The value $u_{i,j}^W$ depends on whether the node (x_{i-1}, y_j) is in Ω, and is given by (2.6). If $\theta_{i,j}^W = \theta_{i,j}^E = 1$, then (2.20) becomes the standard second-order one-sided difference for $[u_x]_{i,j}^{i+1,j}$.

Like the non-standard difference formulae (2.5) used when solving the Fisher–KPP equation, the scheme (2.20) can encounter division by small quantities if $\theta_{i,j}^W \ll 1$ and/or $\theta_{i,j}^E \ll 1$. Thus, if $\theta_{i,j}^E < \theta_{\partial\Omega}$, we instead assume that the interface lies exactly on (x_i, y_j). Provided $(x_{i-1}, y_j) \in \Omega$ and $(x_{i-2}, y_j) \in \Omega$, we apply the standard formula

$$[u_x]_{i,j}^{i+1,j} = \frac{3(u_\mathrm{f} - \gamma K_{i,j}^{i+1,j}) - 4u_{i-1,j} + u_{i-2,j}}{2\Delta x}. \quad (2.21)$$

This scenario is illustrated in Figure 2.3a. If $(x_{i-1}, y_j) \notin \Omega$, we set $[u_x]_{i,j}^{i+1,j} = 0$. If $(x_{i-1}, y_j) \in \Omega$ but $(x_{i-2}, y_j) \notin \Omega$ (as in Figure 2.3b), we adjust (2.21) to obtain an appropriate non-standard three-point stencil. The formula is given by (2.20) evaluated at the point (x_{i-1}, y_j), with $\theta_{i-1,j}^E = 1$, and $u_{i-1,j}^W = u_\mathrm{f}$. If $\theta_{i-1,j}^W < \theta_{\partial\Omega}$, we instead set $[u_x]_{i,j}^{i+1,j} = 0$.



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

Figure 2.3: Finite-difference stencils used to compute the interface normal speed for grid points with $\theta_{i,j}^E < \theta_{\partial\Omega}$. Orange and red nodes denote ghost nodes on the interface, at which we compute the speed using the Stefan condition. Yellow nodes denote grid points used in the stencils for $[u_x]_{i,j}^{i+1,j}$, and pink dots denote ghost nodes used in the stencils for $[u_y]_{i,j}^{i+1,j}$. The green dot indicates the grid point close to the interface, where we assume $u_{i,j} = u_\mathrm{f} - \gamma K_{i,j}$. Black dots denote nodes of the computational grid.

After approximating u_x, the next step is to obtain the derivatives of the level-set

function, ϕ_x and ϕ_y. The methods used are as described in §2.2.1, and we repeat them here for an east-located ghost node for completeness. To compute $[\phi_x]_{i,j}^{i+1,j}$, we first apply second-order central finite-differences at the two grid points adjacent to the interface ghost node. This yields

$$[\phi_x]_{i,j} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}, \tag{2.22a}$$

$$[\phi_x]_{i+1,j} = \frac{\phi_{i+2,j} - \phi_{i,j}}{2\Delta x}. \tag{2.22b}$$

In practice, we prevent the interface from approaching the left and right boundaries of $\mathcal{D}$, avoiding the need for boundary schemes for (2.22). To approximate ϕ_x on the interface ghost node, we then interpolate linearly to obtain

$$[\phi_x]_{i,j}^{i+1,j} = \theta_{i,j}^E \, [\phi_x]_{i+1,j} + \left(1 - \theta_{i,j}^E\right) [\phi_x]_{i,j} \, . \tag{2.23}$$

To approximate ϕ_y at the interface ghost node, we define two additional ghost nodes directly above and below the interface ghost node (pink dots in Figures 2.2 and 2.3). We then use linear interpolation to approximate ϕ at these nodes, which gives

$$\phi_{i,j+1}^{i+1,j+1} = \theta_{i,j}^E \phi_{i+1,j+1} + \left(1 - \theta_{i,j}^E\right) \phi_{i,j+1}, \tag{2.24a}$$

$$\phi_{i,j-1}^{i+1,j-1} = \theta_{i,j}^E \phi_{i+1,j-1} + \left(1 - \theta_{i,j}^E\right) \phi_{i,j-1}. \tag{2.24b}$$

The first derivative approximation is then obtained using the central scheme

$$[\phi_y]_{i,j}^{i+1,j} = \frac{\phi_{i,j+1}^{i+1,j+1} - \phi_{i,j-1}^{i+1,j-1}}{2\Delta y}. \tag{2.25}$$

Finally, we also need $[u_y]_{i,j}^{i+1,j}$ to apply the Stefan condition (1.4d). To apply a one-sided difference, we define two additional ghost nodes directly above the ghost node on the interface (see Figures 2.2 and 2.3), such that

$$u_{i,j}^{i+1,j} = u_{\mathrm{f}} - \gamma K_{i,j}^{i+1,j}, \tag{2.26a}$$

$$u_{i,j+1}^{i+1,j+1} = \theta_{i,j}^E u_{i+1,j+1} + (1 - \theta_{i,j}^E) u_{i,j+1}, \tag{2.26b}$$

$$u_{i,j+2}^{i+1,j+2} = \theta_{i,j}^E u_{i+1,j+2} + (1 - \theta_{i,j}^E) u_{i,j+2}. \tag{2.26c}$$

The second-order one-sided finite-difference approximation is then

$$u_y = \frac{-3u_{i,j}^{i+1,j} + 4u_{i,j+1}^{i+1,j+1} - u_{i,j+2}^{i+1,j+2}}{2\Delta y}. \tag{2.27}$$

The approximation (2.27) assumes that both ghost nodes (x_r, y_{j+1}) and (x_r, y_{j+2}) lie within Ω, that is $\phi_{i,j+1}^{i+1,j+1} < 0$ and $\phi_{i,j+2}^{i+1,j+2} < 0$. The value of $\phi_{i,j+1}^{i+1,j+1}$ can be checked using (2.24a),

and the value of $\phi_{i,j+2}^{i+1,j+2}$ can be estimated using similar linear interpolation,

$$\phi_{i,j+2}^{i+1,j+2} = \theta_{i,j}^{E}\phi_{i+1,j+2} + \left(1 - \theta_{i,j}^{E}\right)\phi_{i,j+2}. \tag{2.28}$$

If both $\phi_{i,j+1}^{i+1,j+1} < 0$ and $\phi_{i,j+2}^{i+1,j+2} < 0$, we use (2.27). If $\phi_{i,j+1}^{i+1,j+1} < 0$ but $\phi_{i,j+2}^{i+1,j+2} \geq 0$, we introduce another ghost node (see Figure 2.4) and define

$$\theta^{G} = \frac{\phi_{i,j+1}^{i+1,j+1}}{\phi_{i,j+1}^{i+1,j+1} - \phi_{i,j+2}^{i+1,j+2}} \tag{2.29}$$

to be the distance from the ghost node at $(x_i + \theta_{i,j}^{E}\Delta x, y_{j+1})$ to the interface, scaled by Δy. For $\theta^{G} \geq \theta_{\partial\Omega}$, we use the approximation

$$[u_y]_{i,j}^{i+1,j} = \frac{1}{\Delta y}\left[-\frac{(2+\theta^{G})(u_{\mathrm{f}} - \gamma K^{G})}{1+\theta^{G}} + \frac{(1+\theta^{G})u_{i,j+1}^{i+1,j+1}}{\theta^{G}} - \frac{u_{\mathrm{f}} - \gamma K_{i,j}^{i+1,j}}{\theta^{G}(1+\theta^{G})}\right], \tag{2.30}$$

where K^{G} represents the interface curvature at the new ghost node. We obtain the value for K^{G} using two-dimensional linear interpolation formulae that involve the four surrounding grid points, (x_i, y_{j+1}), (x_{i+1}, y_{j+1}), (x_i, y_{j+2}), and (x_{i+1}, y_{j+2}). To prevent division by small quantities in (2.30), we set $[u_y]_{i,j}^{i+1,j} = 0$ if $\theta^{G} < \theta_{\partial\Omega}$. Formulae analogous to (2.30) can be developed if Ω is below the interface, that is $\phi_{i,j+1}^{i+1,j+1} \geq 0$, but $\phi_{i,j-1}^{i+1,j-1} < 0$. If both $\phi_{i,j+1}^{i+1,j+1} \geq 0$ and $\phi_{i,j-1}^{i+1,j-1} \geq 0$, we set $[u_y]_{i,j}^{i+1,j} = 0$. Once numerical approximations to u_x, u_y, ϕ_x, and ϕ_y are found at the interface, we apply the Stefan condition to obtain

$$V_{i,j}^{i+1,j} = -\kappa\left[u_x\phi_x + u_y\phi_y\right]_{i,j}^{i+1,j}, \tag{2.31}$$

and store the normal speed at each interface ghost node.

To summarise, the procedure for computing the velocity at a ghost node on the interface is:

1. Determine whether the ghost node lies in the x or y-direction of two grid points, and determine which point (left or right/bottom or top) lies in $\Omega(t)$.

2. Compute the local curvature, K, at each point along the interface $\partial\Omega(t)$, using the methods described in Section 2.2.1.

3. Use a second-order accurate one-sided finite-difference formula to compute the first derivative of u in the direction of the interface ghost point. This involves applying either (2.21) or (2.20), depending on the interface position.

4. Approximate ϕ_x and ϕ_y at the interface ghost node using central differencing and linear interpolation, (2.23) and (2.25).

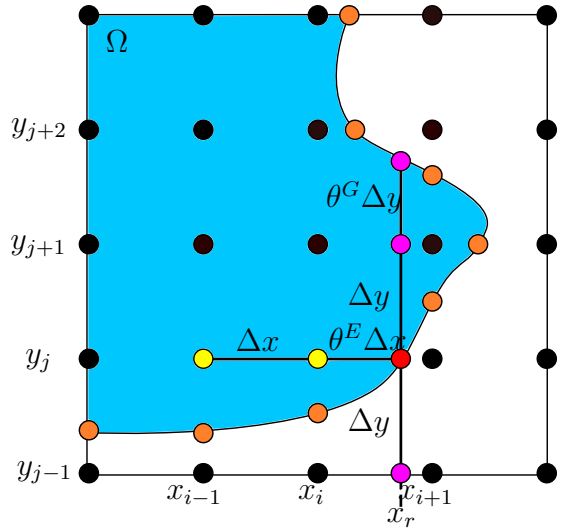


Figure 2.4: Finite-difference stencil used to compute $[u_y]_{i,j}^{i+1,j}$, where $\phi_{i,j+1}^{i+1,j+1} < 0$ but $\phi_{i,j+2}^{i+1,j+2} \geq 0$. The derivative is computed at the red ghost node, and pink nodes indicate the stencils used to compute $[u_y]_{i,j}^{i+1,j}$ and $[\phi_y]_{i,j}^{i+1,j}$. Black and yellow dots denote nodes in the computational grid, and orange dots are other ghost nodes on the interface.

5. Compute the first derivative of u along the non-grid direction using a second-order accurate one-sided difference. Depending on the interface location, this involves applying either (2.27) or (2.30).

6. Once all derivatives of u and ϕ are computed, store the ghost node position and velocity $V = -\kappa(u_x\phi_x + u_y\phi_y)$.

2.3.1 Velocity Extrapolation Away From Interface

To obtain an extension velocity field, $F(x, y, t)$, defined throughout $\mathcal{D}$, we extrapolate V away from the interface in the orthogonal direction. We achieve this by solving the PDE [6]

$$\frac{\partial F}{\partial \tau} + \nabla F \cdot \hat{\boldsymbol{n}} = 0, \tag{2.32}$$

subject to the boundary condition $F = V$ on $\partial\Omega$, where $\hat{\boldsymbol{n}}$ is the unit outward normal to the interface. To obtain F, we first solve (2.32) with the initial condition $F(\boldsymbol{x}, 0) = 0$ everywhere except $\partial\Omega$. This extrapolates the velocity in the outward normal direction. We then use the solution to (2.32) as the initial condition and solve

$$\frac{\partial F}{\partial \tau} + \nabla F \cdot (-\hat{\boldsymbol{n}}) = 0, \tag{2.33}$$

to extrapolate in the inwards normal direction. To solve (2.32) and (2.33) on $\mathcal{D}$, the unit outward normal vector on the interface can be extended to the entire domain via the level-set function, $\hat{\boldsymbol{n}} = \nabla\phi/|\nabla\phi|$. Thus, if $\nabla\phi$ is known we can rewrite both (2.32)

and (2.33) as the linear advection equation

$$\frac{\partial F}{\partial \tau} + a\frac{\partial F}{\partial x} + b\frac{\partial F}{\partial y} = 0. \tag{2.34}$$

To solve (2.32), we use $a = \phi_x/|\nabla\phi|$ and $b = \phi_y/|\nabla\phi|$. To solve (2.33) and extrapolate in the inwards normal direction, we use $a = -\phi_x/|\nabla\phi|$ and $b = -\phi_y/|\nabla\phi|$.

We solve (2.34) using the method of lines, where we approximate derivatives of x and y using first-order upwind finite-differences. At all interior grid points, we approximate the derivatives of ϕ using the central schemes

$$[\phi_x]_{i,j} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}, \quad [\phi_y]_{i,j} = \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y}. \tag{2.35}$$

We assume $\partial\Omega$ is far enough from the left and right boundaries to prevent the need for boundary schemes for $[\phi_x]_{i,j}$. At the periodic boundaries $j = 0, N_y$, we apply

$$[\phi_y]_{i,0} = [\phi_y]_{i,N_y} = \frac{\phi_{i,1} - \phi_{i,N_y-1}}{2\Delta y}. \tag{2.36}$$

Equations (2.35) and (2.36) are sufficient to determine the advection coefficients a and b in (2.34). Then, to apply the method of lines we use the semi-discrete form

$$\frac{\mathrm{d}\phi_{i,j}}{\mathrm{d}\tau} = a\left[\nabla_x F\right]_{i,j} + b\left[\nabla_y F\right]_{i,j}, \tag{2.37}$$

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

$$[\nabla_x F]_{i,j} = \begin{cases} (F_{i+1,j} - F_{i,j})/\Delta x & \text{if} \quad a < 0 \\ (F_{i,j} - F_{i-1,j})/\Delta x & \text{if} \quad a \geq 0. \end{cases} \tag{2.38}$$

At irregular grid points, if the sign of a or b requires differencing in a direction that crosses the interface, we construct these derivatives using the computed speed on the interface ghost node. For example, if the interface lies east of the grid point (x_i, y_j) and $a < 0$, then we use

$$[\nabla_x F]_{i,j} = \frac{V_{i,j}^{i+1,j} - F_{i,j}}{\theta_{i,j}^E \Delta x}. \tag{2.39}$$

If $\theta_{i,j}^E < \theta_{\partial\Omega}$, then we instead set $F_{i,j} = V_{i,j}^{i+1,j}$, and $[\nabla_x F]_{i,j} = [\nabla_y F]_{i,j} = 0$. Again, this is to prevent division by small quantities in (2.39). Other cases at irregular grid points are handled similarly. When solving (2.32) and (2.33), assume that the interface $\partial\Omega$ remains far 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 (2.38) and (2.39), but for $[\nabla_y F]_{i,j}$). The implementation of periodic boundary conditions follows similar methods to those outlined previously.

We integrate in the 'pseudo-time' τ using `DifferentialEquations.jl`, again using the fifth-order Tsitouras Runge–Kutta method with the absolute tolerance of 1×10^{-3} and relative tolerance of 1×10^{-6}. For both equations (2.32) and (2.33), we compute solutions until $\tau = 20\Delta\tau$, where $\Delta\tau = 0.5(\Delta x/5 + \Delta y/5)$. This ensures that the velocity field $F(\boldsymbol{x}, t)$ is accurate in a neighbourhood of the interface. In turn, this ensures that 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 (1.4b) is a multidimensional Hamilton–Jacobi PDE of the form

$$\frac{\partial \phi}{\partial t} + H\left(\nabla \phi\right) = 0, \tag{2.40}$$

where $H = F|\nabla\phi|$ is the Hamiltonian. These equations are similar to hyperbolic conservation laws, and require similar numerical methods. We implement the second-order scheme described by Lin and Tadmor [7], and convert this to a non-staggered scheme using the averaging method of Jiang et al. [8]. The Lin–Tadmor method is based on the Nessyahu–Tadmor scheme for hyperbolic PDEs [9]. The method is a Godunov-type central scheme that does not require the approximate solution of a Riemann problem, and does not require upwind differencing.

The numerical scheme described by Lin and Tadmor [7] is a fully-discrete method that solves (2.40) for ϕ^{k+1} on a staggered grid. The method is (where subscripts outside the brackets now denote partial differentiation)

$$\begin{aligned}
\phi^{k+1}_{i+1/2,j+1/2} = {} & \frac{1}{4}\left(\phi^k_{i,j} + \phi^k_{i+1,j} + \phi^k_{i,j+1} + \phi^k_{i+1,j+1}\right) \\
& + \frac{\Delta x}{16}\left[\left(\phi^k_{i,j}\right)_x - \left(\phi^k_{i+1,j}\right)_x + \left(\phi^k_{i,j+1}\right)_x - \left(\phi^k_{i+1,j+1}\right)_x\right] \\
& + \frac{\Delta y}{16}\left[\left(\phi^k_{i,j}\right)_y - \left(\phi^k_{i,j+1}\right)_y + \left(\phi^k_{i+1,j}\right)_y - \left(\phi^k_{i+1,j+1}\right)_y\right] \\
& - \frac{\Delta t}{2}\left[H\left(\frac{\phi^{k+1/2}_{i+1,j} - \phi^{k+1/2}_{i,j}}{\Delta x}, \frac{\phi^{k+1/2}_{i+1,j+1} - \phi^{k+1/2}_{i+1,j}}{\Delta y}\right)\right. \\
& \left. + H\left(\frac{\phi^{k+1/2}_{i+1,j+1} - \phi^{k+1/2}_{i,j+1}}{\Delta x}, \frac{\phi^{k+1/2}_{i,j+1} - \phi^{k+1/2}_{i,j}}{\Delta y}\right)\right],
\end{aligned} \tag{2.41}$$

where $(x_{i+1/2}, y_{j+1/2}) = (x_i + \Delta x/2, y_j + \Delta y/2)$ for $i = 0, \ldots N_x - 1$ and $j = 0, \ldots N_y - 1$ defines the staggered grid. Computing (2.41) 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

$$\phi_{i,j}^{k+1/2} = \phi_{i,j}^{k} - \frac{\Delta t}{2} H\left[\left(\phi_{i,j}^{k}\right)_x, \left(\phi_{i,j}^{k}\right)_y\right]. \tag{2.42}$$

The derivatives are computed using a suitable flux-limiter to ensure that the scheme remains non-oscillatory [7]. We use the min-mod limiter [7], such that

$$\left(\phi_{i,j}^{k}\right)_x = MM\left(\theta\frac{\phi_{i+1,j}^{k} - \phi_{i,j}^{k}}{\Delta x}, \frac{\phi_{i+1,j}^{k} - \phi_{i-1,j}^{k}}{2\Delta x}, \theta\frac{\phi_{i,j}^{k} - \phi_{i-1,j}^{k}}{\Delta x}\right) \tag{2.43a}$$

$$\left(\phi_{i,j}^{k}\right)_y = MM\left(\theta\frac{\phi_{i,j+1}^{k} - \phi_{i,j}^{k}}{\Delta y}, \frac{\phi_{i,j+1}^{k} - \phi_{i,j-1}^{k}}{2\Delta y}, \theta\frac{\phi_{i,j}^{k} - \phi_{i,j-1}^{k}}{\Delta y}\right), \tag{2.43b}$$

for some $\theta \in [1, 2]$. In our scheme, we use $\theta = 1.99$ [10], indicating a preference for central differencing where possible. The symbol MM denotes the min-mod function, which applied to a vector v_i is

$$MM(v_i) = \begin{cases} \min_i \{v_i\} & \text{if} \quad v_i > 0 \quad \forall i \\ \max_i \{v_i\} & \text{if} \quad v_i < 0 \quad \forall i \\ 0 & \text{otherwise.} \end{cases} \tag{2.44}$$

Once the staggered solution is computed using (2.41), it can be converted to a non-staggered scheme by averaging following Jiang et al. [8],

$$\begin{aligned} \phi_{i,j}^{k+1} = {} & \frac{1}{4}\left(\phi_{i+1/2,j+1/2}^{k+1} + \phi_{i-1/2,j+1/2}^{k+1} + \phi_{i-1/2,j-1/2}^{k+1} + \phi_{i+1/2,j-1/2}^{k+1}\right) \\ & + \frac{\Delta x}{16}\left[\left(\phi_{i-1/2,j-1/2}^{k+1}\right)_x - \left(\phi_{i+1/2,j-1/2}^{k+1}\right)_x + \left(\phi_{i-1/2,j+1/2}^{k+1}\right)_x - \left(\phi_{i+1/2,j+1/2}^{k+1}\right)_x\right] \\ & + \frac{\Delta y}{16}\left[\left(\phi_{i-1/2,j-1/2}^{k+1}\right)_y - \left(\phi_{i-1/2,j+1/2}^{k+1}\right)_y + \left(\phi_{i+1/2,j-1/2}^{k+1}\right)_y - \left(\phi_{i+1/2,j+1/2}^{k+1}\right)_y\right], \end{aligned} \tag{2.45}$$

where again the appropriate derivatives are computed using the min-mod limiter (2.43). Together with (2.41), (2.45) describes a scheme to solve for $\phi_{i,j}^{k+1}$ given $\phi_{i,j}^{k}$. When solving (2.40), we apply Dirichlet boundary conditions on the left and right boundaries $\phi_{0,j}^{k+1} = \phi_{0,j}^{k}$, $\phi_{N_x,j}^{k+1} = \phi_{N_x,j}^{k}$, for all k. This is appropriate, because we assume the interface is far from the left and right boundaries of $\mathcal{D}$. If necessary, we apply periodic conditions in the y-direction at upper and lower boundaries. Periodic conditions apply for all central difference formula for ϕ_y in (2.41)–(2.43) and (2.45). Their implementation follows similar methods to those outlined previously.

2.5 Level-Set Function Reinitialisation

Using orthogonal extrapolation to obtain the extension velocity field does not maintain the signed-distance property of the level-set function, ϕ [2]. Therefore, we develop a method

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

$$\frac{\partial \phi}{\partial \tau} + S(\phi)\left(|\nabla \phi| - 1\right) = 0, \tag{2.46}$$

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

$$S(\phi) = \frac{\phi}{\sqrt{|\nabla \phi|^2 + (\Delta x)^2}}. \tag{2.47}$$

Like the level-set equation (1.4b), the reinitialisation equation (2.46) is a Hamilton–Jacobi equation. It has the form $\phi_\tau + H(\phi, \nabla\phi) = 0$, with $H = S(\phi)(|\nabla\phi| - 1)$. Thus, we adapt the Lin–Tadmor method (2.41)–(2.43) and (2.45) used to solve the level-set equation to solve (2.46). Where required in the evaluation of H, we use $\phi^k_{i,j}$ instead of values at the half-time steps. In our solutions, we reinitialise the signed-distance function every time-step, and apply 20 τ-steps, with $\Delta\tau = 0.5(\Delta x/5 + \Delta y/5)$.

References

[1] S. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, ed. by S. S. Antman, J. E. Marsden, and L. Sirovich, Applied Mathematical Sciences, New York: Springer-Verlag, 2003, ISBN: 978-0-387-22746-7.

[2] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, ed. by P. G. Ciarlet, A. Iserles, R. V. Kohn, and M. H. Wright, Second, Cambridge Monographs on Applied and Computational Mathematics, 32 Avenue of the Americas, New York, NY 10013-2473, USA: Cambridge University Press, 1999, ISBN: 978-0-521-64557-7.

[3] 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", *ANZIAM Journal* 63 (2021), pp. 269–307, ISSN: 1446-1811, 1446-8735, DOI: 10.1017/S144618112100033X.

[4] C. Tsitouras, "Runge–Kutta Pairs of Order 5(4) Satisfying Only the First Column Simplifying Assumption", *Computers & Mathematics with Applications* 62 (2011), pp. 770–775, ISSN: 0898-1221, DOI: 10.1016/j.camwa.2011.06.002.

[5] C. V. Rackauckas and Q. Nie, "DifferentialEquations.Jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia", *Journal of Open Research Software* 5, 15 (1 2017), ISSN: 2049-9647, DOI: 10.5334/jors.151.

[6] T. D. Aslam, "A Partial Differential Equation Approach to Multidimensional Extrapolation", *Journal of Computational Physics* 193 (2004), pp. 349–355, ISSN: 0021-9991, DOI: 10.1016/j.jcp.2003.08.001.

[7] C. T. Lin and E. Tadmor, "High-Resolution Nonoscillatory Central Schemes for Hamilton–Jacobi Equations", *SIAM Journal on Scientific Computing* 21 (2000), pp. 2163–2186, ISSN: 1064-8275, DOI: 10.1137/S1064827598344856.

[8] 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* 35 (1998), pp. 2147–2168, ISSN: 0036-1429, DOI: 10.1137/S0036142997317560.

[9] H. Nessyahu and E. Tadmor, "Non-Oscillatory Central Differencing for Hyperbolic Conservation Laws", *Journal of Computational Physics* 87 (1990), pp. 408–463, ISSN: 0021-9991, DOI: 10.1016/0021-9991(90)90260-8.

[10] M. J. Simpson, K. A. Landman, and T. P. Clement, "Assessment of a Non-Traditional Operator Split Algorithm for Simulation of Reactive Transport", *Mathematics and Computers in Simulation* 70 (2005), pp. 44–60, ISSN: 0378-4754, DOI: 10.1016/j.matcom.2005.03.019.