A new ghost cell/level set method for moving boundary problems

Macklin, Paul; Lowengrub, John S.

Published in:
Journal of Scientific Computing

DOI:
10.1007/s10915-008-9190-z

Publication date:
2008

Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

A New Ghost Cell/Level Set Method for Moving Boundary Problems: Application to Tumor Growth

Paul Macklin · John S. Lowengrub

Received: September 14, 2007 / Accepted: date

Abstract In this paper, we present a ghost cell/level set method for the evolution of interfaces whose normal velocity depend upon the solutions of linear and nonlinear quasi-steady reaction-diffusion equations with curvature-dependent boundary conditions. Our technique includes a ghost cell method that accurately discretizes normal derivative jump boundary conditions without smearing jumps in the tangential derivative; a new iterative method for solving linear and nonlinear quasi-steady reaction-diffusion equations; an adaptive discretization to compute the curvature and normal vectors; and a new discrete approximation to the Heaviside function. We present numerical examples that demonstrate better than 1.5-order convergence for problems where traditional ghost cell methods either fail to converge or attain at best sub-linear accuracy. We apply our techniques to a model of tumor growth in complex, heterogeneous tissues that consists of a nonlinear nutrient equation and a pressure equation with geometry-dependent jump boundary conditions. We simulate the growth of *glioblastoma* (an aggressive brain tumor) into a large, 1 cm square of brain tissue that includes heterogeneous nutrient delivery and varied biomechanical characteristics (white matter, gray matter, cerebrospinal fluid, and bone), and we observe growth morphologies that are highly dependent upon the variations of the tissue characteristics–an effect observed in real tumor growth.

Paul Macklin
SHIS, 7000 Fannin, Suite 600, U. of Texas Health Science Center, Houston, TX 77030
www: `http://biomathematics.shis.uth.tmc.edu`

John Lowengrub
Dept. of Mathematics, 103 MSTB, University of California, Irvine, CA 92697
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: lowengrub@math.uci.edu
www: `http://math.uci.edu/~lowengrb`

Keywords ghost fluid method · ghost cell method · level set method · tumor growth · NAGSI · nonlinear elliptic equations · Heaviside function · heterogeneous media · heterogeneous tissue structure · adaptive normal vector calculation · normal derivative jump boundary condition · Poisson equation

Mathematics Subject Classification (2000) 35 · 65 · 92

1 Introduction

The algorithms we develop in this paper are motivated by our interest in modeling tumor growth in complex, heterogeneous tissues. Cancer is a fundamental scientific and societal problem, and in the past several decades, intensive research has been focused on understanding the complexity of cancer progression, developing new therapies, and formulating optimal treatment protocols. While much work has been done in the mathematical community on tumor modeling (e.g., see the reviews [4,9,8,11,48,47]), to date there has been little work in modeling tumor growth in realistic, heterogeneous tissues on large spatial scales. The methods we present in this paper will provide the foundation for a biologically-detailed millimeter-to-centimeter-scale model of tumor growth in heterogeneous tissues with realistic features (e.g., mechanically soft and hard regions, bone, and inhomogeneous nutrient delivery) [6,35,37]. However, the methods described in this paper have applications beyond the tumor growth context and can be applied to general systems of linear and nonlinear quasi-steady reaction-diffusion problems on moving, heterogeneous domains.

In previous work [36,38–40], we investigated simpler models of tumor growth using a level set/ghost fluid method that we developed in [36] and [38]; our technique tested second-order accurate when applied to interior problems, including the tumor growth model. (Hereafter, we refer to ghost fluid methods as ghost cell methods to emphasize that they have applications beyond fluid mechanics.) In [39], we improved the accuracy and robustness of level set-based curvature calculations in cases where two interfaces are in close contact, and we extended our approach to the two-sided problem in [40]. However, this work still smeared any jumps in the tangential derivative across the interface, assumed homogeneous tumor microenvironments (with piecewise constant biophysical parameters) and was not capable of simulating growth into complex tissue structures. We note that Zheng et al. [57] and Hogea et al. [28] have also used level set methods to study tumor growth and angiogenesis, but this work also assumed homogeneous tissues and used lower-order accurate level set methods. Frieboes et al. [19, 18] and Wise et al. [54] have begun studying 3D tumor growth using a diffuse interface approach, while others have begun studying the tumor problem using multiphase mixture models (e.g., see [5], [10], and [12]). Still others use discrete models, such as cellular automata and agents (e.g., see [1], [7], and [11] for some recent examples).

In this paper, we present a ghost cell/level set method for the evolution of interfaces whose normal velocity depend upon the solutions of linear and nonlin-

ear quasi-steady reaction-diffusion equations with curvature-dependent boundary conditions. We introduce a new normal derivative jump discretization for the ghost cell method that accurately discretizes the jump without numerically smearing any tangential derivative jump. In this approach, the normal jump is written as a combination of two grid-aligned jumps that are easier to compute. This addresses a longstanding problem with the ghost cell method, and in numerical testing, our extended ghost cell method achieves better than 1.5-order accuracy in cases where the traditional normal derivative jump stencil either fails to converge or attains sub-first-order accuracy, regardless of mesh refinement. We also present a new adaptive normal vector calculation that allows us to robustly calculate appropriate normal vectors even in the presence of multiple, non-convex regions; we use this new adaptive normal vector discretization in our improved ghost cell method.

To solve nonlinear quasi-steady reaction-diffusion equations on large domains, we developed a nonlinear adaptive Gauss-Seidel-type iterative method (NAGSI) that can solve both linear and nonlinear problems using a localized update on a regular Cartesian mesh, and is fully compatible with ghost cell extrapolations. NAGSI is an adaptive solution method that builds upon earlier Gauss-Seidel-type iterative (GSI) methods by using a dynamic selection criterion to focus computational effort without the need for a complex adaptive mesh. We find that NAGSI is second-order accurate when used to solve a variety of linear and nonlinear problems, and its adaptivity achieves between a 10% and 50% reduction in computational time when compared to identical GSI methods without our adaptivity.

We apply these techniques to nonlinear moving boundary problems, where the velocity of the boundary depends upon the gradients of linear and nonlinear quasi-steady reaction-diffusion equations. When testing on a modified Hele-Shaw flow problem, our overall method demonstrates second-order accuracy. In the Hele-Shaw type problem, we simulate a growing drop of incompressible fluid in a medium with heterogeneous permeability; the drop grows preferentially in the regions of highest permeability. We also apply the techniques developed in this paper to model the growth of *glioblastoma* (an aggressive brain tumor) in a large (1 cm $\times$ 1 cm), heterogeneous section of brain tissue, including white and gray matter with differing biomechanical properties, cerebrospinal fluid, and bone. The numerical advances presented in this paper enabled us to solve this complex problem in a short period of time (under 24 hours of computation) while observing new behavior, such as preferential growth of the tumor in regions of reduced biomechancial resistance.

The outline of this paper is as follows. In Section 2, we introduce the general system of quasi-steady, linear and nonlinear reaction-diffusion equations that we solve on moving domains. In Section 3, we discuss the level set method, present our techniques for robustly and accurately calculating geometric quantities (i.e., curvature and normal vectors), introduce the ghost cell method, present our new normal derivative jump discretization that preserves the tangential derivative jump, and introduce our nonlinear adaptive Gauss-Seidel-type iterative (NAGSI)

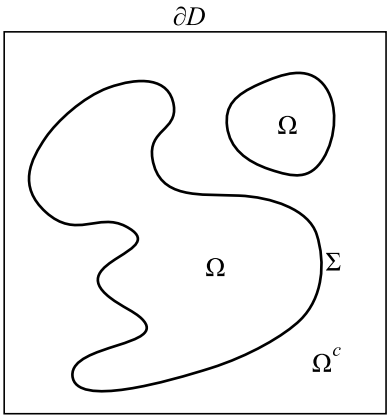

Fig. 1 Regions for the general nonlinear quasi-steady reaction-diffusion moving boundary system.

scheme for solving linear and nonlinear quasi-steady reaction-diffusion equations. We close Section 3 by combining these techniques to solve the general system presented in Section 2. In Section 4, we test the numerical convergence of our new ghost cell method using the new normal derivative jump discretization and the NAGSI solver, as well as our overall technique. In Section 5, we present examples derived from Hele-Shaw flow in a heterogeneous material and tumor growth in a complex, heterogeneous simulated tissue. We discuss our results and future work in Section 6.

2 The Equations for the Quasi-Steady Reaction-Diffusion System

We wish to solve systems of (potentially nonlinear) quasi-steady reaction-diffusion equations on a domain $\mathcal{D}$ that is divided into two subdomains $\Omega(t)$ and $\Omega^c(t)$ by a moving interface $\Sigma(t)$. See Figure 1. The interface $\Sigma(t)$ evolves with a velocity that depends upon the gradients of these solutions. That is, we solve for a system of functions $p_1, p_2, \ldots, p_k$ on $\mathcal{D}$ with that satisfy equations of the form

$$\begin{aligned} 0 = \nabla \cdot (D_i(\mathbf{x}, t, p_i) \nabla p_i) + f_{R,i}(\mathbf{x}, t, p_1, \cdots, p_i)\, p_i \\ + f_{S,i}(\mathbf{x}, t, p_1, \cdots, p_i) \end{aligned} \tag{1}$$

on $\mathcal{D} \backslash \Sigma$, coupled with jump boundary conditions

$$[p_i] = g_i \tag{2}$$

$$[D_i \nabla p_i \cdot \mathbf{n}] = h_i \tag{3}$$

on Σ and either Dirichlet, Neumann, or extrapolation (extrapolated from the interior of the domain) boundary conditions on $\partial\mathcal{D}$. Here, $\mathbf{n}$ is the outward unit

normal vector (pointing into Ω^c), and we define a jump in a quantity q at a point $\mathbf{x}_\Sigma \in \Sigma$ by

$$\begin{aligned}[q(\mathbf{x})] &= q_{in} - q_{out} \\ &= \lim_{\Omega \ni \mathbf{x} \to \mathbf{x}_\Sigma} q(\mathbf{x}) - \lim_{\Omega^c \ni \mathbf{x} \to \mathbf{x}_\Sigma} q(\mathbf{x}). \end{aligned} \tag{4}$$

In the case where $g_i = 0$ and $h_i = 0$, this reduces to a regular (linear or nonlinear) diffusion problem throughout the domain $\mathcal{D}$.

The interfacial outward normal velocity V is given by

$$V = \sum_{i=1}^{k} \alpha_i \nabla p_i \cdot \mathbf{n}. \tag{5}$$

3 Numerical Solution Techniques

Before discussing our solution technique for the overall system, we introduce the key methods that will be required. Our overall technique is centered around a level set/ghost cell method which we first developed for a tumor growth problem in [36], [38], [39], and [40]. For completeness, we shall describe the overall approach, with a focus on new improvements in the method.

3.1 Narrow Band/Local Level Set Method

Level set methods were first developed by Osher and Sethian in [45] and have been used to study the evolution of moving surfaces that experience frequent topology changes (e.g., merger of regions and fragmentation), particularly in the contexts of fluid mechanics and computer graphics. (See the books [49, 44] and references [45, 43, 50].) In the level set method, the location of a region Ω is captured implicitly by introducing an auxilliary signed distance function ϕ that satisfies

$$\begin{cases} \phi(\mathbf{x}) < 0 & \mathbf{x} \in \Omega \\ \phi(\mathbf{x}) = 0 & \mathbf{x} \in \Sigma = \partial\Omega \\ \phi(\mathbf{x}) > 0 & \mathbf{x} \notin \Omega \\ |\nabla \phi(\mathbf{x})| \equiv 1. \end{cases} \tag{6}$$

In the level set approach, instead of explicitly tracking the position of interface Σ and manually handling topology changes, the level set function is updated by solving a PDE, which automatically accounts for the interface motion and all topology changes. If V is the outward normal velocity of the interface, then we update the position of the interface implicitly via

$$\phi_t + \widetilde{V} |\nabla \phi| = 0, \tag{7}$$

where $\widetilde{V}$ is an extension of V off of the interface. The extension $\widetilde{V}$ is often obtained using a Hamilton-Jacobi PDE. (e.g., see [56] and [2].) The fast marching method

developed by Adalsteinsson and Sethian in [3] constructs an extension $\widetilde{V}$ while simultaneously reinitializing the level set function using an ordered sequence of discrete operations, but is only first-order accurate. In [38] we developed a bilinear extension technique that is both faster and more accurate than the traditional, PDE-based approach.

Solving (7) can introduce numerical error into the level set function that perturbs it away from being a distance function, even for special choices of $\widetilde{V}$ that are constant in the normal direction from the interface [2,49] and thereby preserve distance functions. This is compensated for by reinitializing the level set function at regular intervals by solving

$$\phi_\tau = \text{sign}(\phi^0)\left(1 - |\nabla\phi|\right) \tag{8}$$

to steady state [46,52]. Here, τ is pseudo-time, and ϕ^0 is the original level set function prior to the reinitialization.

We discretize the spatial operator $|\nabla\phi|$ in (7) and (8) using the fifth-order weighted essentially non-oscillatory (WENO) method [30,29], and we discretize pseudo-time in (8) using the third-order total variation-diminishing Runge-Kutta method (TVD-RK) from [25] and [26]. Due to the computational cost and the complexity of our tumor system, we currently discretize time in (7) using a forward Euler algorithm and a small step size. We discretize the sign function as in [52].

Lastly, because the primary purpose of a level set function is to track the position of the interface Σ over time, its accuracy is most important on and near the interface. To find the best compromise between accuracy and computational efficiency, we seek to update ϕ only as much as is necessary to accurately advect the interface. This can be done using the narrow band/local level set technique [41,46]. Given an initialized level set function ϕ, only the points that fall within a fixed distance of the interface are updated during level set operations (e.g., velocity extensions and level set reinitialization). In the level set context, the narrow band can be identified by

$$\{\mathbf{x} : |\phi(\mathbf{x})| \leq R\}, \tag{9}$$

where $R > 0$ is a fixed constant that is chosen to suit the problem. In our work with the tumor problem, we use $R = 20\Delta x$. In some cases, we shall use a *semi*band $\{\mathbf{x} : \phi(\mathbf{x}) \leq R\}$.

3.2 Calculating Geometric Quantities

One of the advantages of the level set method is that the level function encodes all the geometric information. In particular, the outward-facing normal vector $\mathbf{n}$ is given by

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}, \tag{10}$$

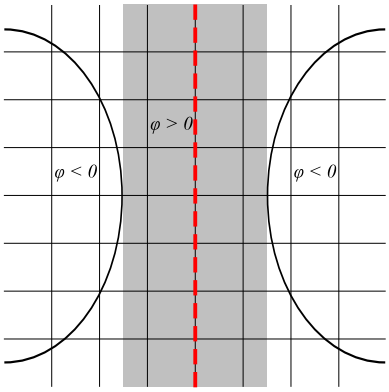

Fig. 2 Two interfaces in close contact: Points along the central dashed line are equidistant from both interfaces, resulting in discontinuities in the level set derivatives. The level set function ϕ tends to be an inaccurate approximation of a distance function and irregular in the adjacent gray areas.

and the mean curvature can be computed via

$$\kappa = \nabla \cdot \mathbf{n} = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right). \tag{11}$$

As we noted and used in [38] and [39], the level set function can also be used to estimate the closest point $\mathbf{x}_\Sigma = (x_\Sigma, y_\Sigma)$ on the interface to a given point $\mathbf{x} = (x, y)$ by

$$\mathbf{x}_\Sigma = \mathbf{x} - \phi(\mathbf{x})\mathbf{n}(\mathbf{x}). \tag{12}$$

These geometric quantities can readily be calculated at computational grid points using standard centered differences. If a geometric quantity (e.g., curvature) is desired at a non-grid point $\mathbf{x}'$, then we calculate the geometric quantity at nearby node points and interpolate to find the desired quantity at $\mathbf{x}'$ [36,38,39]. In our work, we have generally used bicubic interpolation (or cubic interpolation when $\mathbf{x}'$ lies on a grid edge but not on a computational grid point).

However, as we demonstrated in [36] and [38], the level set function can develop discontinuities in its derivatives in regions that are equidistant from multiple portions of the interface. Furthermore, advecting and reinitializing the level set function tends to introduce error into the regions near the singularities. This can lead to difficulty when computing normal vectors and curvature when two interfaces are in close contact, introducing inaccuracy into the geometric quantities. See Figure 2.

In [39], we introduced a new, geometry-aware curvature discretization to automatically detect and accurately deal with this scenario. To calculate the curvature κ at a point $\mathbf{x}_\Sigma = (x_\Sigma, y_\Sigma)$ on the interface, recall that we need to compute and

interpolate the curvature $\kappa_{i,j}$ at nearby computational node points. Using the level set quality function

$$Q(\mathbf{x}) = |1 - |\nabla \phi(\mathbf{x})|| \tag{13}$$

that we first defined in [36] and [38], we detected level set irregularity whenever $Q \geq \eta$ for some threshold $\eta > 0$. (In our work, we have generally used $\eta \sim 0.001$.) To calculate the curvature $\kappa_{i,j}$ at a computational node point (x_i, y_j), we evaluated Q at each of the nine grid points in $\left\{ \left(x_{i+k}, j_{j+\ell}\right) : -1 \leq k, \ell \leq 1 \right\}$. If $Q < \eta$ at each of these points, then the level set function was deemed sufficiently smooth to calculate the curvature $\kappa_{i,j}$ using the standard 9-point curvature stencil

$$\begin{cases} \phi_x \approx \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} & \phi_y \approx \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \\ \phi_{xx} \approx \frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{\Delta x^2} & \phi_{yy} \approx \frac{\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}}{\Delta y^2} \\ \phi_{xy} \approx \frac{\phi_{i+1,j+1} - \phi_{i-1,j+1} - \phi_{i+1,j-1} + \phi_{i-1,j-1}}{4\Delta x \Delta y} & \\ \kappa_{i,j} \approx \frac{\phi_{xx}\phi_y^2 - 2\phi_x \phi_y \phi_{xy} + \phi_{yy}\phi_x^2}{\left(\phi_x^2 + \phi_y^2\right)^{\frac{3}{2}}}, & \end{cases} \tag{14}$$

which is second order accurate where the level set function is smooth [36,38]. If we could calculate the curvature $\kappa_{i,j}$ at enough nearby points to compute a bicubic or bilinear interpolation at $\mathbf{x}_\Sigma$, then we used that interpolated curvature value.

If $Q > \eta$ at one of the points on the stencil, we constructed a positively-oriented local approximation of the interface $\gamma(s) = (x(s), y(s))$ by finding five points $\{\mathbf{x}_k = (x_k, y_k)\}_{k=-2}^{2}$ with $\mathbf{x}_0 = \mathbf{x}_\Sigma$ on the interface and calculating a quadratic least squares polynomial fit. After adjusting γ to ensure that $\gamma(0) = \mathbf{x}_\Sigma$, we then used γ to construct a local level set function that effectively removed the influence of the nearby irregularity, which we could then discretize using the standard 9-point stencil. In our numerical testing, this geometry-aware, adaptive curvature discretization was second order accurate, even during periods of topological change such as the merger of drops in modified Hele-Shaw flow [39]. In more recent testing, we have found that so long as γ is a least squares quadratic or cubic polynomial fit and not a direct interpolation of the five points $\mathbf{x}_k$ on the interface, it can be differentiated directly to compute the curvature.

We now extend this technique to calculate normal vectors. Suppose we desire the normal vector at a computational node point (x_i, y_j). If the level set is sufficiently smooth at the four points of $\left\{(x_{i-1}, y_j), (x_i, y_{j-1}), (x_{i+1}, y_j), (x_i, y_{j+1})\right\}$ (i.e., $Q < \eta$ at those points), then we use the standard normal vector discretization

$$\begin{cases} \phi_x \approx \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \\ \phi_y \approx \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \\ \mathbf{n} \approx \frac{1}{\sqrt{\phi_x^2 + \phi_y^2 + \epsilon}} \left(\phi_x, \phi_y\right), \end{cases} \tag{15}$$

where ϵ is a small positive number used to avoid division by zero; we use $\epsilon \sim 10^{-16}$ in our work.

Otherwise, we identify the closest point on the interface $\mathbf{x}_\Sigma$ by (12), construct the approximating curve $\gamma(s) = (x(s), y(s))$ through $\mathbf{x}_\Sigma$ as in the adaptive curvature algorithm, and directly differentiate the curve to determine the unit tangent and outward normal vectors:

$$\mathbf{s} = \frac{\gamma'(0)}{||\gamma'(0)||} = \frac{1}{\sqrt{(x'(0))^2 + (y'(0))^2}} \Big(x'(0)\ ,\ y'(0) \Big) \tag{16}$$

$$\mathbf{n} = \frac{1}{\sqrt{(x'(0))^2 + (y'(0))^2}} \Big(y'(0)\ ,\ -x'(0) \Big). \tag{17}$$

We have found that both quadratic and cubic least squares polynomial fits are sufficiently smooth for this direct differentiation of the normal and tangent vectors.

3.3 The Ghost Cell Method

We wish to solve quasi-steady reaction-diffusion problems of the form

$$\begin{cases} 0 = \nabla \cdot (D(\mathbf{x})\nabla p) + f_R(\mathbf{x})p + f_S(\mathbf{x}) & \mathbf{x} \in \Omega \cup \Omega^c \\ [p] = g & \mathbf{x} \in \Sigma \\ [D\nabla p \cdot \mathbf{n}] = h & \mathbf{x} \in \Sigma, \end{cases} \tag{18}$$

coupled with standard (e.g., Dirichlet, Neumann, or extrapolation) boundary conditions on $\partial\mathcal{D}$. (Note that D and p are scalars, i.e., $D = D_i$ and $p = p_i$ for some fixed $1 \le i \le k$ in the notation of (1).)

Standard finite differences cannot be applied across the interface due to the jump boundary conditions on Σ. The ghost cell method was developed to deal with this issue when solving elliptical problems by creating "ghost" computational points and using those ghost points in standard finite difference discretizations [24,17,34,23,22]. In [36] and [38], we extended the ghost cell method to attain second-order accuracy on interior problems (i.e., p is constant in Ω^c) with boundary conditions that depend upon the geometry (e.g., curvature) and without a jump condition on the normal derivative. A similar extension to the ghost cell method was presented in [21] to solve Laplace's equation without geometric boundary conditions and yielded fourth-order convergence on fixed domains and third-order convergence on moving boundaries. In [40], we extended our approach to solve systems like (18) in the case where $h = 0$ and D was constant in Ω and Ω^c (with different constants). We applied the method to model avascular tumor growth in [40] and verified second-order accuracy for our overall solutions, although our method numerically smeared jumps in the tangential derivative. In the following section, we present our new ghost cell scheme, which includes a new technique for discretizing the normal derivative jump condition without smearing the tangential derivative jump.

Alternative approaches for overcoming the tangential smearing problem include the immersed interface method (IIM) [32] and the matched interface boundary (MIB) method [59,58,55]. The IIM requires the use of local coordinates (based on the normal and tangential directions of the interface) to properly discretize the normal derivative jump. The MIB method is a high-order generalization of the IIM and the ghost cell method that constructs elaborate extensions of the solution to several ficticious points on both sides of the interface; the extensions are designed to explicitly satisfy $[p]$, $[\partial p/\partial \mathbf{s}]$, and $[D\partial p/\partial \mathbf{n}]$ simultaneously. The ghost fluid method has several advantages over these alternatives. Because the method is applied in a dimension-by-dimension manner, it is simple to implement and can be trivially extended to higher dimensions. Its accuracy can easily be improved by using higher-order extrapolations on each side of the interface. Like the MIB method, our new ghost cell method satisfies the normal derivative jump boundary condition without smearing the tangential derivative jump, but it retains the dimension-by-dimension aspect of the ghost cell method and does not require the explicit treatment of the tangential derivative jump. It is also much simpler to implement and can be incorporated into existing ghost cell frameworks.

3.3.1 Ghost Cell Extrapolations for the Diffusional Term

Suppose we wish to discretize the x-derivative of $\nabla\cdot(D(\mathbf{x})\nabla p)$ at a computational node point $\mathbf{x} = (x_i, y_j)$, and assume that D is C^1 with respect to $\mathbf{x}$ throughout Ω and Ω^c. (The diffusion constant may be discontinuous across the interface Σ; this case is treated below.) If (x_{i-1}, y_j), (x_i, y_j), and (x_{i+1}, y_j) are all in the same region, i.e.,

$$\phi_{i-1,j} \leq 0, \qquad \phi_{i,j} \leq 0, \text{ and} \qquad \phi_{i+1,j} \leq 0, \tag{19}$$

or

$$\phi_{i-1,j} > 0, \qquad \phi_{i,j} > 0, \text{ and} \qquad \phi_{i+1,j} > 0, \tag{20}$$

then we can use the standard second-order discretization

$$\begin{aligned} \partial_x \left(D(x)p_x\right) \approx \quad & \frac{1}{\Delta x^2}\left(D_x^- p_{i-1,j} - \left(D_x^- + D_x^+\right) p_{i,j} + D_x^+ p_{i+1,j}\right) \\ D_x^- &= D\left(x_i - \frac{1}{2}\Delta x, y_j\right) \\ D_x^+ &= D\left(x_i + \frac{1}{2}\Delta x, y_j\right). \end{aligned} \tag{21}$$

Suppose, however, that (x_i, y_j) and (x_{i+1}, y_j) are not in the same region. Assume without loss of generality that $(x_i, y_j) \in \Omega$ and $(x_{i+1}, y_j) \in \Omega^c$; the case where $(x_i, y_j) \in \Omega^c$ and $(x_{i+1}, y_j) \in \Omega$ is treated similarly. Then the interface Σ must separate (x_i, y_j) and (x_{i+1}, y_j) at some point

$$\mathbf{x}_\Sigma = (x_\Sigma, y_j) = (x_i + \theta\Delta x, y_j), \tag{22}$$

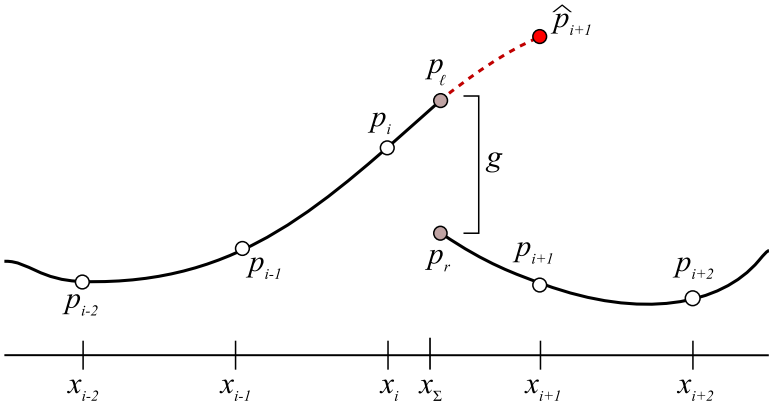

Fig. 3 A typical ghost cell extrapolation $\widehat{p}_{i+1}$. Particular care must be taken to satisfy $[p] = g$ and $[D\nabla p \cdot \mathbf{n}] = h$ without numerically smearing any tangential derivative jump $[\nabla p \cdot \mathbf{s}]$.

where $0 < \theta \leq 1$. In this scenario, we must modify our computational stencil by extending the solution in Ω to a "ghost cell point" $\widehat{p}_{i+1,j}$ in the other region; we replace $p_{i+1,j}$ in (21) with the extrapolation $\widehat{p}_{i+1,j}$. See Figure 3. If (x_{i-2}, y_j) and (x_{i-1}, y_j) are both in Ω, then we obtain $\widehat{p}_{i+1,j}$ as a quadratic extrapolation of p from $p_{i-2,j}$, $p_{i-1,j}$, and u_ℓ:

$$\widehat{p}_{i+1,j} = \frac{2(1-\theta)}{2+\theta} p_{i-2} - \frac{3(1-\theta)}{1+\theta} p_{i-1} + \frac{6}{(1+\theta)(2+\theta)} p_\ell. \tag{23}$$

If $(x_{i-1}, y_j) \in \Omega$ but $(x_{i-2}, y_j) \notin \Omega$, then we obtain $\widehat{p}_{i+1,j}$ by linear extrapolation from $p_{i-1,j}$ and p_ℓ:

$$\widehat{p}_{i+1,j} = \frac{-(1-\theta)}{1+\theta} p_{i-1,j} + \frac{2}{1+\theta} p_\ell. \tag{24}$$

If $(x_{i-1}, y_j) \notin \Omega$, then we use the constant extrapolation $\widehat{p}_{i+1,j} = p_\ell$. Note that in all cases, we require p_ℓ. In the ghost cell approach, p_ℓ is determined by the jump boundary conditions. We shall return to this point in the next section.

If D is continuous across the interface Σ, then D_x^+ may be used in the ghost cell approximation without modification. If D is discontinuous across Σ, then we replace it with an extension $\widehat{D}_x^+$ in a manner analogous to $\widehat{p}_{i+1,j}$. Suppose that D is defined at computational node points. If $\left(x_{i-2}, y_j\right)$ and $\left(x_{i-1}, y_j\right)$ are both in Ω, then we use quadratic extrapolation:

$$\widehat{D}_x^+ = \frac{3}{8} D_{i-2,j} - \frac{5}{4} D_{i-1,j} + \frac{15}{8} D_{i,j}. \tag{25}$$

If $\left(x_{i-1}, y_j\right) \in \Omega$ but $\left(x_{i-2}, y_j\right) \notin \Omega$, then we use linear extrapolation from $D_{i-1,j}$ and $D_{i,j}$:

$$\widehat{D}_x^+ = -\frac{1}{2} D_{i-1,j} + \frac{3}{2} D_{i,j}. \tag{26}$$

If $(x_{i-1}, y_j) \notin \Omega$, then we use constant extrapolation $\widehat{D}_x^+ = D_{i,j}$.

The case where we require a ghost cell extrapolation $\widehat{p}_{i-1,j}$ is completely analogous, and the y-derivative is discretized in exactly the same manner. To apply the method to a 3-D problem, for instance, we need only repeat the process separately for the z-derivative term $\partial_z \left(D(\mathbf{x}) p_z \right)$.

3.3.2 Determining p_ℓ from the Jump Boundary Conditions

As we have seen, the ghost cell extrapolations require p_ℓ. At the same time, the jump boundary conditions have not yet been applied to the scheme. By introducing the jump boundary conditions into the ghost cell extrapolation, we can simultaneously satisfy the jump conditions while eliminating p_ℓ from the extrapolation, instead expressing the discretization solely in terms of computational node points.

First, we discretize the jump boundary condition by

$$p_\ell - p_r = -\text{sign}(\phi_{i,j}) g(\mathbf{x}_\Sigma). \tag{27}$$

The $-\text{sign}(\phi_{i,j})$ term ensures that the jump condition has been applied in the proper direction from region Ω to region Ω^c.

This introduces an additional, non-grid point p_r into our calculation. However, by considering the normal derivative jump condition, we shall have two equations for p_ℓ and p_r, allowing us to completely eliminate them from the extrapolation. The proper discretization of the normal derivative jump $[D \nabla p \cdot \mathbf{n}]$ across the interface has been an open problem since the introduction of the ghost cell method for the Poisson problem [17,34,23]. Suppose that we wish to discretize $[D \nabla p \cdot \mathbf{n}]$ at the point $\mathbf{x}_\Sigma = (x_\Sigma, y_j) = (x_i + \theta \Delta x, y_j)$ from the preceding discussion. In [34], the normal derivative jump was discretized as

$$\begin{aligned} [D \nabla p \cdot \mathbf{n}] &= \left(D \frac{\partial p}{\partial \mathbf{n}} \right)_\ell - \left(D \frac{\partial p}{\partial \mathbf{n}} \right)_r \\ &\approx D_\ell \frac{p_\ell - p_{i,j}}{\theta \Delta x} - D_r \frac{p_{i+1,j} - p_r}{(1-\theta)\Delta x}, \end{aligned} \tag{28}$$

where

$$\begin{aligned} D_r = \lim_{\zeta \downarrow x + \theta \Delta x} D\left(\zeta, y_j\right), \qquad D_\ell = \lim_{\zeta \uparrow x + \theta \Delta x} D\left(\zeta, y_j\right), \\ p_r = \lim_{\zeta \downarrow x + \theta \Delta x} p\left(\zeta, y_j\right), \qquad \text{and} \qquad p_\ell = \lim_{\zeta \uparrow x + \theta \Delta x} p\left(\zeta, y_j\right). \end{aligned} \tag{29}$$

Notice that this is equivalent to assuming that the normal vector cuts the computational mesh at a right angle (i.e., $\mathbf{n} = (1, 0)$), potentially leading to numerical smearing of any jump in the tangential derivative. Furthermore, if $\theta \sim 0$ or $\theta \sim 1$, then the discretization can be unstable due to the uneven spacing of the stencil points. See the left frame in Figure 4. In numerical testing in [34], this stencil was less than first-order accurate due to the low order of

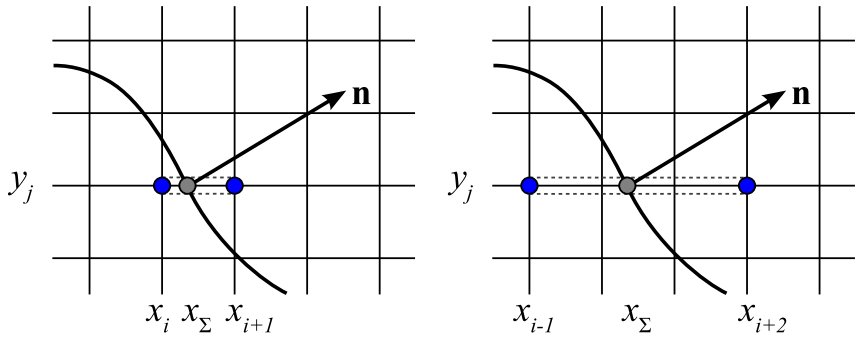

Fig. 4 Left: The traditional, unstable stencil for $[D\nabla p \cdot \mathbf{n}]$ from [34]. **Right:** Our stablized extension of this stencil from [40].

the discretization (first-order left and right differences of the derivative), coupled with the numerical smearing of the tangential derivatives.

In [40], we introduced a new discretization for the normal derivative jump in the case where $[D\nabla p \cdot \mathbf{n}] = 0$:

$$[D\nabla p \cdot \mathbf{n}] \approx D_\ell \frac{p_\ell - p_{i-1,j}}{(1+\theta)\Delta x} - D_r \frac{p_{i+2,j} - p_r}{(2-\theta)\Delta x}. \tag{30}$$

This discretization guarantees that all three stencil points (x_{i-1}, y_j), (x_Σ, y_j), and (x_{i+1}, y_j) are at least Δx apart and consequently solves the stability problem. However, the discretization still numerically smears any jump in the tangential derivative because it approximates the normal vector as $\mathbf{n} = (1, 0)$. See the right frame in Figure 4.

We now introduce a new normal derivative jump discretization that eliminates the numerical smearing of the jump in the tangential derivative. Let $\mathbf{n} = (n_x, n_y)$, and suppose that $n_x \cdot n_y \geq 0$, i.e., the normal vector faces up and right or down and left. Assuming no additional nearby interfaces (so that all right and left points are contained within the same regions), we begin by defining

$$\begin{array}{ll} \mathbf{u}_r = (x_{i+2} - x_\Sigma, 0) & = ((2-\theta)\Delta x, 0)\,, \\ \mathbf{v}_r = \big(x_{i+1} - x_\Sigma, y_{j+1} - y_j\big) & = ((1-\theta)\Delta x, \Delta y)\,, \\ \mathbf{u}_\ell = (x_{i-1} - x_\Sigma, 0) & = (-(1+\theta)\Delta x, 0)\,, \text{ and} \\ \mathbf{v}_\ell = \big(x_i - x_\Sigma, y_{j-1} - y_j\big) & = (-\theta\Delta x, -\Delta y)\,. \end{array} \tag{31}$$

See the left frame in Figure 5.

Because $\mathbf{u}_r$ and $\mathbf{v}_r$ are linearly independent, they form a basis for $\mathbb{R}^2$, and we can write

$$\mathbf{n} = a_r \mathbf{u}_r + b_r \mathbf{v}_r, \tag{32}$$

where a_r and b_r are obtained by solving the linear system

$$\begin{array}{l} (\mathbf{u}_r, \mathbf{u}_r)\, a_r + (\mathbf{v}_r, \mathbf{u}_r)\, b_r = (\mathbf{n}, \mathbf{u}_r) \\ (\mathbf{u}_r, \mathbf{v}_r)\, a_r + (\mathbf{v}_r, \mathbf{v}_r)\, b_r = (\mathbf{n}, \mathbf{v}_r)\,. \end{array} \tag{33}$$

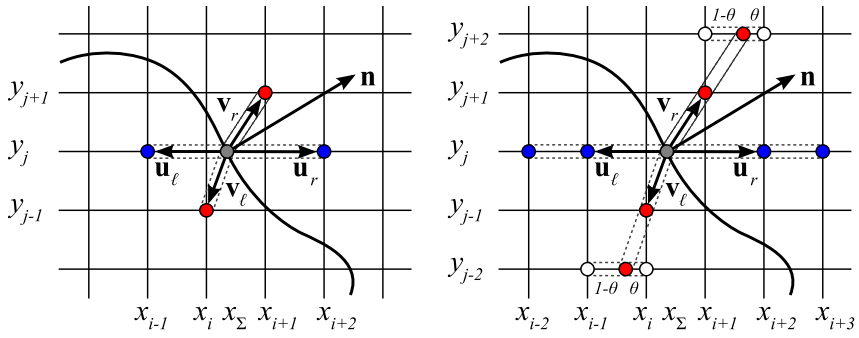

Fig. 5 Our new first-order (**left**) and higher-order (**right**) computational stencils for the normal derivative jump $[D\nabla p \cdot \mathbf{n}]$.

Using this, we can express $(\partial p/\partial \mathbf{n})_r$ as a combination of grid ($\mathbf{u}_r$) and off-grid ($\mathbf{v}_r$) directional derivatives:

$$\begin{aligned}\left(\frac{\partial p}{\partial \mathbf{n}}\right)_r &= (\nabla p)_r \cdot \mathbf{n} \\ &= a_r (\nabla p)_r \cdot \mathbf{u}_r + b_r (\nabla p)_r \cdot \mathbf{v}_r \\ &= a_r ||\mathbf{u}_r|| (\nabla p)_r \cdot \frac{\mathbf{u}_r}{||\mathbf{u}_r||} + b_r ||\mathbf{v}_r|| (\nabla p)_r \cdot \frac{\mathbf{v}_r}{||\mathbf{v}_r||} \\ &= a_r ||\mathbf{u}_r|| \left(\frac{\partial p}{\partial (\mathbf{u}_r/||\mathbf{u}_r||)}\right)_r + b_r ||\mathbf{v}_r|| \left(\frac{\partial p}{\partial (\mathbf{v}_r/||\mathbf{v}_r||)}\right)_r, \end{aligned} \tag{34}$$

where

$$||\mathbf{u}_r|| = (2-\theta)\Delta x, \qquad ||\mathbf{v}_r|| = \sqrt{(1-\theta)^2 \Delta x^2 + \Delta y^2}. \tag{35}$$

Using one-sided, first-order differences, we obtain the approximation

$$\begin{aligned}\left(\frac{\partial p}{\partial \mathbf{n}}\right)_r &\approx a_r ||\mathbf{u}_r|| \frac{(p_{i+2,j} - p_r)}{||\mathbf{u}_r||} + b_r ||\mathbf{v}_r|| \frac{(p_{i+1,j+1} - p_r)}{||\mathbf{v}_r||} \\ &= a_r (p_{i+2,j} - p_r) + b_r (p_{i+1,j+1} - p_r). \end{aligned} \tag{36}$$

Similarly, $\mathbf{u}_\ell$ and $\mathbf{v}_\ell$ form a basis for $\mathbb{R}^2$, and we can express

$$\mathbf{n} = a_\ell \mathbf{u}_\ell + b_\ell \mathbf{v}_\ell, \tag{37}$$

rewrite the normal derivative as

$$\left(\frac{\partial p}{\partial \mathbf{n}}\right)_\ell = a_\ell ||\mathbf{u}_\ell|| \left(\frac{\partial p}{\partial (\mathbf{u}_\ell/||\mathbf{u}_\ell||)}\right)_\ell + b_\ell ||\mathbf{u}_\ell|| \left(\frac{\partial p}{\partial (\mathbf{v}_\ell/||\mathbf{v}_\ell||)}\right)_\ell, \tag{38}$$

and approximate it (to first-order) as

$$\left(\frac{\partial p}{\partial \mathbf{n}}\right)_\ell \approx a_\ell (p_{i-1,j} - p_\ell) + b_\ell (p_{i,j-1} - p_\ell). \tag{39}$$

By combining (36) and (39), we obtain a new discretization of $[D\nabla p \cdot \mathbf{n}] = h$:

$$\begin{aligned}[D\nabla p \cdot \mathbf{n}] \approx \quad & D_\ell\Big(a_\ell\left(p_{i-1,j} - p_\ell\right) + b_\ell\left(p_{i,j-1} - p_\ell\right)\Big) \\ & -D_r\Big(a_r\left(p_{i+2,j} - p_r\right) + b_r\left(p_{i+1,j+1} - p_r\right)\Big) \\ = \ & -\text{sign}(\phi_{i,j})h(\mathbf{x}_\Sigma). \end{aligned} \tag{40}$$

When we combine this with the jump condition in (27), p_ℓ and p_r are completely determined. In practice, we implement this new discretization in the following way:

1. We construct 2×2 linear systems for a_r and b_r and and a_ℓ and b_ℓ and solve them exactly using the standard inversion for 2×2 systems.
2. We construct a 2×2 linear system for p_r and p_ℓ using the jump condition for $p_\ell - p_r$ and the normal derivative jump discretization in (40). We solve this linear system for p_ℓ and p_r using the standard 2×2 inversion and use p_ℓ in the ghost cell discretization.

The case where $n_x \cdot n_y < 0$ is similar: we replace $p_{i+1,j+1}$ by $p_{i+1,j-1}$ in $\mathbf{v}_r$, and we replace $p_{i,j-1}$ by $p_{i,j+1}$ in $\mathbf{v}_r$. All other calculations are identical. The case of discretizing $[D\nabla p \cdot \mathbf{n}]$ at a point $(x_i - \theta\Delta x, y_j)$ is completely analogous, and the discretization in the y-direction is identical. In the 3-D case, we would proceed similarly by defining third basis vectors such as

$$\begin{aligned} \mathbf{w}_r &= (x_{i+1} - x_\Sigma, 0, z_{k+1} - z_k) = \Big((1-\theta)\Delta x, 0, \Delta z\Big), \\ \mathbf{w}_\ell &= (x_i - x_\Sigma, 0, z_{k-1} - z_k) \quad = \Big(-\theta\Delta x, 0, -\Delta z\Big). \end{aligned} \tag{41}$$

This method has several advantages. In the spirit of the ghost cell method, the new stencil can be expressed entirely in terms of computational grid points and the jump interface location. Furthermore, the discretization can be applied in a dimension-by-dimension manner, just as in existing ghost cell methods. The implementation is simple and can be readily substituted into existing code in place of previous normal discretizations. Because the stencil discretizes $D\partial p/\partial \mathbf{n}$ in the true direction of the normal vector, it should result in more accurate numerical solutions with less numerical smearing of the tangential derivative jump. Whereas the ghost cell method in [34] was constructed with restrictions to guarantee a diagonally-dominant matrix system that must necessarily converge, we have no rigorous proof of convergence for our method. However, we find that in practice that our method does converge, even for difficult interfacial morphologies. (Indeed, we shall verify that our method is first-order accurate, even in situations where the discretization by [34] fails to converge; the discretization we present in the next section attains above-1.5-order accuracy.) Lastly, we note that if $\mathbf{n} = (1, 0)$ or $\mathbf{n} = (-1, 0)$, then the normal derivative jump discretization in (40) simplifies to (30). This further illustrates the point that earlier discretizations of the normal derivative jump were equivalent to assuming that the interface cuts the computational grid at a right angle.

3.3.3 Higher-Order Approximations of $[D\nabla p \cdot \mathbf{n}]$

By using additional stencil points (with the same definitions of $\mathbf{u}_\ell$, $\mathbf{u}_r$, $\mathbf{v}_\ell$, and $\mathbf{v}_r$), we can improve the accuracy of the normal derivative jump discretization. If (x_{i+1}, y_j), (x_{i+2}, y_j), and (x_{i+3}, y_j) are all in the same region, we can replace the grid-aligned partial derivative in (34) with the second-order approximation

$$\left(\frac{\partial p}{\partial \left(\mathbf{u}_r / \left|\left|\mathbf{u}_r\right|\right|\right)}\right)_r \approx \frac{-(5+2\theta)}{(2-\theta)(3-\theta)\Delta x} p_r + \frac{(3-\theta)}{(2-\theta)\Delta x} p_{i+2,j} - \frac{(2-\theta)}{(3-\theta)\Delta x} p_{i+3,j}. \tag{42}$$

See the right frame of Figure 5.

Similarly, if (x_i, y_j), (x_{i-1}, y_j), and (x_{i-2}, y_j) are all in the same region, then we can replace the grid-aligned partial derivative in (38) with the second-order approximation

$$\left(\frac{\partial p}{\partial \left(\mathbf{u}_\ell / \left|\left|\mathbf{u}_\ell\right|\right|\right)}\right)_\ell \approx -\frac{(1+\theta)}{(2+\theta)\Delta x} p_{i-2,j} + \frac{(2+\theta)}{(1+\theta)\Delta x} p_{i-1,j} - \frac{(3+2\theta)}{(1+\theta)(2+\theta)\Delta x} p_\ell. \tag{43}$$

Because $\mathbf{v}_\ell$ and $\mathbf{v}_r$ generally do not intersect the computational mesh at grid points, interpolation of nearby grid points is required to improve the accuracy of the off-grid directional derivatives in (34) and (38).

If (x_{i+1}, y_j), (x_{i+1}, y_{j+1}), (x_{i+1}, y_{j+2}), and (x_{i+2}, y_{j+2}) are all in the same region, then we can replace the off-grid directional derivative in (34) by the higher-order approximation

$$\left(\frac{\partial p}{\partial \left(\mathbf{v}_r / \left|\left|\mathbf{v}_r\right|\right|\right)}\right)_r \approx -\frac{3}{2\left|\left|\mathbf{v}_r\right|\right|} p_r + \frac{2}{\left|\left|\mathbf{v}_r\right|\right|} p_{i+1,j+1} - \frac{1}{2\left|\left|\mathbf{v}_r\right|\right|}\Big[\theta p_{i+1,j+2} + (1-\theta) p_{i+2,j+2}\Big], \tag{44}$$

where the bracketed term is a linear interpolation to obtain a point along the path of $\mathbf{v}_r$. See the right frame of Figure 5.

Similarly, if (x_i, y_j), (x_i, y_{j-1}), (x_i, y_{j-2}), and (x_{i-1}, y_{j-2}) are all in the same region, then we can improve our approximation of the off-grid derivative in (38) with

$$\left(\frac{\partial p}{\partial \left(\mathbf{v}_\ell / \left|\left|\mathbf{v}_\ell\right|\right|\right)}\right)_\ell \approx -\frac{3}{2\left|\left|\mathbf{v}_\ell\right|\right|} p_\ell + \frac{2}{\left|\left|\mathbf{v}_\ell\right|\right|} p_{i,j-1} - \frac{1}{2\left|\left|\mathbf{v}_\ell\right|\right|}\Big[(1-\theta) p_{i,j-2} + \theta p_{i-1,j-2}\Big]. \tag{45}$$

Putting all this together, the higher-order discretization of $[D\nabla p \cdot \mathbf{n}] = h$ is given by

$$\begin{aligned}
[D\nabla p \cdot \mathbf{n}] \approx \quad & D_\ell a_\ell \left(-\frac{(1+\theta)^2}{(2+\theta)} p_{i-2,j} + (2+\theta) p_{i-1,j} - \frac{(3+2\theta)}{(2+\theta)} p_\ell \right) \\
& + D_\ell b_\ell \left(-\frac{3}{2} p_\ell + 2p_{i,j-1} - \frac{1}{2} \left[(1-\theta) p_{i,j-2} + \theta p_{i-1,j-2} \right] \right) \\
& - D_r a_r \left(\frac{-(5+2\theta)}{(3-\theta)} p_r + (3-\theta) p_{i+2,j} - \frac{(2-\theta)^2}{(3-\theta)} p_{i+3,j} \right) \\
& - D_r b_r \left(-\frac{3}{2} p_r + 2p_{i+1,j+1} - \frac{1}{2} \left[\theta p_{i+,j+2} + (1-\theta) p_{i+2,j+2} \right] \right) \\
= \; & -\text{sign}(\phi_{i,j}) h\left(\mathbf{x}_\Sigma\right). \qquad (46)
\end{aligned}$$

In principle, all these partial differences can be made more accurate still by using additional node points. In particular, one could use quadratic or cubic interpolations in the off-grid directional derivative differences.

3.4 NAGSI: a Nonlinear Adaptive Gauss-Seidel type Iterative method for solving Nonlinear Quasi-Steady Reaction-Diffusion Equations

We now develop a fully-nonlinear, adaptive method to solve the nonlinear reaction-diffusion equation

$$0 = \nabla \cdot \left(D\left(\mathbf{x}, p\right) \nabla p \right) + f_R\left(\mathbf{x}, p\right) p + f_S\left(\mathbf{x}, p\right), \qquad \mathbf{x} \in \mathcal{D}, \qquad (47)$$

where D and p are scalars (i.e., $D = D_i$ and $p = p_i$ for some fixed $1 \le i \le k$ in the notation of (1)), $\mathcal{D}$ is a rectangular domain in $\mathbb{R}^n$, and standard boundary conditions (e.g., Neumann, Dirichlet, or extrapolation) have been assigned. Here, f_R and f_S are the reaction and source terms, respectively. Without loss of generality, we assume that $f_R \le 0$; if this condition is not satisfied, then the positive part of $f_R\, p$ can be rewritten as part of the source term f_S. The diffusivity D is assumed to be smooth and strictly positive. For simplicity, we shall consider the 2-D case where $\mathcal{D} = [a,b] \times [c,d]$; the n-D case is completely analogous.

In Section 3.4.1, we give a semi-implicit formulation of our method which allows for solving (47) using a GSI method, similar to pseudo-time methods discussed in [14,33,42] and GSI method literature. (e.g., see [27], [13], and [53].) Using this formulation, we present a new form of adaptivity in Section 3.4.2 that can be applied to regular Cartesian meshes.

3.4.1 A Semi-Implicit Formulation of GSI Methods

To solve (47), we solve the related equation

$$\frac{\partial p}{\partial \tau} = \nabla \cdot \left(D\left(\mathbf{x}, p\right) \nabla p \right) + f_R\left(\mathbf{x}, p\right) p + f_S\left(\mathbf{x}, p\right) \qquad (48)$$

to steady state, where τ is pseudo-time. For numerical stability, we begin by implicitly discretizing pseudo-time with a backwards Euler difference, lagging the dependence of D on p, and applying the standard second-order centered difference to the diffusional term:

$$\begin{aligned}
\frac{p_{i,j}^{n+1}-p_{i,j}^{n}}{\Delta\tau} = & \frac{1}{\Delta x^2}\left(D_x^- p_{i-1,j}^{n+1} - \left(D_x^- + D_x^+\right) p_{i,j}^{n+1} + D_x^+ p_{i+1,j}^{n+1}\right) \\
& + \frac{1}{\Delta y^2}\left(D_y^- p_{i,j-1}^{n+1} - \left(D_y^- + D_y^+\right) p_{i,j}^{n+1} + D_y^+ p_{i,j+1}^{n+1}\right) \\
& + f_R\left(x_i, p_i^n\right) p_i^{n+1} + f_S\left(x_i, p_i^n\right) + \mathcal{O}\left(\Delta\tau + \Delta x^2 + \Delta y^2\right), \\
D_x^- = & D\left(x_i - \frac{1}{2}\Delta x, y_j, \frac{1}{2}\left(p_{i-1,j}^n + p_{i,j}^n\right)\right), \\
D_x^+ = & D\left(x_i + \frac{1}{2}\Delta x, y_j \frac{1}{2}\left(p_{i,j}^n + p_{i+1,j}^n\right)\right), \\
D_y^- = & D\left(x_i, y_j - \frac{1}{2}\Delta y, \frac{1}{2}\left(p_{i,j-1}^n + p_{i,j}^n\right)\right), \\
D_y^+ = & D\left(x_i, y_j + \frac{1}{2}\Delta y, \frac{1}{2}\left(p_{i,j}^n + p_{i,j+1}^n\right)\right).
\end{aligned} \tag{49}$$

Here, $p_{i,j}^n = p\left(x_i, y_j, \tau_n\right)$, $x_i = a + i\Delta x$, $y_j = c + j\Delta y$, $\tau_n = n\Delta\tau$, and Δx, Δy and $\Delta\tau$ are spatial and pseudo-temporal discretization step sizes, respectively.

This system has the form

$$A\left(\mathbf{x}, \mathbf{p}^n\right)\mathbf{p}^{n+1} = \mathbf{b}\left(\mathbf{x}, \mathbf{p}^n\right) \tag{50}$$

which can be solved to steady state by constructing the operator $A\left(\mathbf{x}, \mathbf{p}^n\right)$ and right-hand side $\mathbf{b}\left(\mathbf{x}, \mathbf{p}^n\right)$ and solving the linear system in (50) with an iterative method (e.g., BiCG-Stab(ℓ) from [51]) at every pseudo-time step until the system reaches steady state. However, constructing these operators may be computationally expensive, making the method disadvantageous. Furthermore, after some initial large change throughout the computational domain, the solution may only be rapidly changing on a small subset of the domain. In such a case, the majority of the computational cost of constructing and solving a new linear system at every pseudo-timestep will be unnecessary.

By changing the points used in the discretization of $\nabla \cdot (D\nabla p)$, we can make the scheme semi-implicit. Assuming that we sweep through grid points with increasing i and j:

$$\begin{aligned}
\frac{p_{i,j}^{n+1}-p_{i,j}^{n}}{\Delta\tau} = & \frac{1}{\Delta x^2}\left(D_x^- p_{i-1,j}^{n+1} - \left(D_x^- + D_x^+\right) p_{i,j}^{n+1} + D_x^+ p_{i+1,j}^{n}\right) \\
& + \frac{1}{\Delta y^2}\left(D_y^- p_{i,j-1}^{n+1} - \left(D_y^- + D_y^+\right) p_{i,j}^{n+1} + D_y^+ p_{i,j+1}^{n}\right) \\
& + f_R\left(x_i, p_i^n\right) p_i^{n+1} + f_S\left(x_i, p_i^n\right) + \mathcal{O}\left(\Delta\tau + \Delta x^2 + \Delta y^2\right),
\end{aligned}$$

$$
\begin{aligned}
D_x^- &= D\left(x_i - \frac{1}{2}\Delta x, y_j, \frac{1}{2}\left(p_{i-1,j}^{n+1} + p_{i,j}^n\right)\right), \\
D_x^+ &= D\left(x_i + \frac{1}{2}\Delta x, y_j \frac{1}{2}\left(p_{i,j}^n + p_{i+1,j}^n\right)\right), \\
D_y^- &= D\left(x_i, y_j - \frac{1}{2}\Delta y, \frac{1}{2}\left(p_{i,j-1}^{n+1} + p_{i,j}^n\right)\right), \\
D_y^+ &= D\left(x_i, y_j + \frac{1}{2}\Delta y, \frac{1}{2}\left(p_{i,j}^n + p_{i,j+1}^n\right)\right),
\end{aligned} \tag{51}
$$

which we can solve algebraically for $p_{i,j}^{n+1}$ based upon known quantities:

$$
p_{i,j}^{n+1} = \frac{\frac{p_{i,j}^n}{\Delta\tau} + \frac{D_x^+ p_{i+1,j}^n + D_x^- p_{i-1,j}^{n+1}}{\Delta x^2} + \frac{D_y^+ p_{i,j+1}^n + D_y^- p_{i,j-1}^{n+1}}{\Delta y^2} + f_S\left(x_i, y_j, p_{i,j}^n\right)}{\frac{1}{\Delta\tau} + \frac{D_x^- + D_x^+}{\Delta x^2} + \frac{D_y^- + D_y^+}{\Delta y^2} - f_R\left(x_i, y_j, p_{i,j}^n\right)} \tag{52}
$$

Using this, we now introduce a 2-D GSI scheme:

Step 1: Discretize the domain $[a, b] \times [c, d]$ by

$$
a = x_0, \cdots, x_i = a + i\Delta x, \cdots, x_m = a + m\Delta x = b \tag{53}
$$

$$
c = y_0, \cdots, y_j = c + j\Delta y, \cdots, y_n = c + n\Delta n = d. \tag{54}
$$

Step 2: Store an initial guess in the 2-D array $\mathbf{p} = \left(p_{i,j}\right)$.

Step 3: Update the boundary points $\left\{p_{0,j}, p_{m,j}\right\}_{j=0}^n$ and $\left\{p_{i,0}, p_{i,n}\right\}_{i=0}^m$ according to the boundary conditions.

Step 4: Choose a tolerance ϵ, and set $r > \epsilon$.

Step 5: While $r > \epsilon$:

Part a: For $1 \leq i < m$ and $1 \leq j < n$:

i: Set $r = 0$.

ii: Calculate $p_{i,j}^{n+1}$ based upon Equation 52.

iii: If $r_{i,j} = \left|p_{i,j}^{n+1} - p_{i,j}^n\right| > r$, set $r = r_{i,j}$.

iv: Overwrite $p_{i,j}$ with the newly calculated $p_{i,j}^{n+1}$.

The resulting scheme, which overwrites previous values $p_{i,j}^n$ with updated values $p_{i,j}^{n+1}$ while sweeping through the domain, is a nonlinear Gauss-Seidel-like iterative (GSI) method. To prevent biases (e.g., asymmetry) from the update order, we alternate sweeping directions: up and right (increasing i and j), then up and left (decreasing i, increasing j), then down and left (decreasing i and j), and then down and right (increasing i and decreasing j). By overwriting the current values in the $\left\{p_{i,j}\right\}$ data structure with newly-calculated values, the proper indexing of the pseudo-time index in the discretization of $\nabla \cdot (D\nabla p)$ is automatic.

There are numerous advantages to this technique. First and foremost, there is no need to invert a large linear system at every iteration. Second, because the scheme makes use of updated information while sweeping through the domain

(unlike Jacobi-like iterations), the effects of linearizing the equation with respect to pseudo-time are reduced. Because (52) updates the solution based upon *local* operations (i.e., it only requires information on nearby computational nodes), it is parallelizable and simple to implement, particularly on shared-memory architectures. Lastly, we note that this technique is fully compatible with the ghost cell method of enforcing jump boundary conditions on irregular domains by replacing the appropriate point in the stencil with a ghost cell extrapolation. (e.g., replace $p^n_{i+1,j}$ with $\widehat{p}^n_{i+1.j}$.)

We note that Gauss-Seidel-like iterative methods have been used to solve nonlinear problems in the past, generally in the context of nonlinear optimization. (e.g., see [27], [13].) The idea of adapting linear iterative methods to solving nonlinear problems is not new. (e.g., see [53].) However, most of those techniques use complicated, block structures, which in themselves require iterative solutions and are not as well-suited to adaptivity.

3.4.2 Adaptivity

The local nature of our GSI technique allows for a new approach to adaptivity using a regular Cartesian mesh. On any sweep through the solution domain, as the solution converges, the numerical solution tends to change most on a small fraction of the computational nodes. Therefore, we can select computational nodes where the numerical solution is changing most rapidly and use (52) to update only those nodes. The modified, nonlinear adaptive GSI technique (NAGSI) is as follows:

Step 1: Discretize the domain $[a,b]\times[c,d]$ by

$$a = x_0, \cdots, x_i = a + i\Delta x, \cdots, x_m = a + m\Delta x = b \tag{55}$$

$$c = y_0, \cdots, y_j = c + j\Delta y, \cdots, y_n = c + n\Delta n = d. \tag{56}$$

Step 2: Store an initial guess in the 2-D array $\mathbf{p} = \left(p_{i,j}\right)$.

Step 3: Update the boundary points $\left\{p_{0,j}, p_{m,j}\right\}_{j=0}^{n}$ and $\left\{p_{i,0}, p_{i,n}\right\}_{i=0}^{m}$ according to the boundary conditions.

Step 4: Choose a tolerance ϵ, and set $r > \epsilon$.

Step 5: While $r > \epsilon$:

- **Part a:** For $1 \le i < m$ and $1 \le j < n$:
  - **i:** Set $r = 0$.
  - **ii:** Calculate $p^{n+1}_{i,j}$ based upon (52).
  - **iii:** If $r_{i,j} = \left|p^{n+1}_{i,j} - p^n_{i,j}\right| > r$, set $r = r_{i,j}$.
  - **iv:** Overwrite $p_{i,j}$ with the newly calculated $p^{n+1}_{i,j}$.
- **Part b:** Set a threshold $\eta < r$. In our work, we have used $\eta = \frac{1}{4}r$.
- **Part c:** Sweep through the domain again (with a different sweep direction), this time creating a list $\mathcal{L} = \{(i_k, j_k)\}_{k=1}^{N}$ of nodes where $r_{i,j} > \eta$.
- **Part d:** Repeat M times:

i: For $1 \leq k \leq N$, update p_{i_k,j_k} according to Equation 52.

The scheme is illustrated schematically in Figure 6.

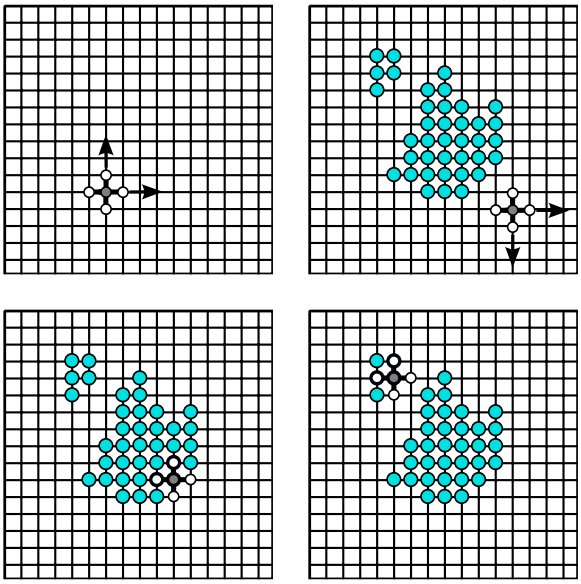

Fig. 6 Overview of the NAGSI adaptivity: **upper left:** We sweep through the entire domain and update all points. We then set a threshold η not exceeding the residual. **upper right:** We sweep through the entire domain again (using a different sweep pattern) and flag all points where the change exceeded the threshold η. **lower left and right:** We sweep through and update the flagged points only, for one or more times.

In our work, we use $M = 2$; in testing, we have found that additional iterations through the selected nodes $\mathcal{L}$ resulted in little change in the approximate solution. This is because $\mathcal{L}$ behaves as a small, irregular subdomain of $\mathcal{D}$ with fixed boundary conditions (i.e., the remaining computational nodes), and so an elliptical equation can approach a steady state on the subdomain quickly. However, the optimal choice of M may depend upon the problem under study and could potentially be dynamically chosen; we are currently investigating these approaches.

While the method does need to sweep through the entire computational domain for some of the iterations, its adaptivity is based upon the same philosophy of traditional adaptive mesh techniques of focusing computational effort adaptively where it is most needed, leading to accelerated convergence when compared to non-adaptive, fixed grid methods. In testing on level set problems with large solution gradients, we have found that the adaptivity decreases the computational time of NAGSI by 10% to 50% (results not shown). Furthermore, the strategy

for choosing the list $\mathcal{L}$ of flagged points can be tailored to the computational problem. (e.g., in level set methods, one might choose $\mathcal{L}$ to include a narrow band about the zero level set.) Because the method requires little extra effort to implement, it can conceivably be used to improve performance in existing computational frameworks with a minimum of reprogramming. We note that NAGSI's convergence could be accelerated by combining it with multigrid techniques (e.g., such as those used for iterative techniques in computational fluid dynamics problems by [14], [33], and [42]) by implementing NAGSI at each level of the multigrid technique. Lastly, we note that while we have focused our adaptivity approach on improving performance on fixed, regular grids, we believe it is possible to apply a similar philosophy to irregular grids by choosing local discretizations that can be applied to selected mesh points for improved performance.

3.5 Solving the Overall System

We solve the overall model from Section 2 using a level set/ghost cell method that uses the methods we just introduced. At every fixed time t, our method consists of the following steps:

Step 1: Maintain ϕ as a distance function, and pre-compute any required geometrical quantities, such as normal vectors and curvature. Use the geometry-aware discretization discussed in Section 3.2.

Step 2: Solve the quasi-steady reaction-diffusion problems for each p_i using NAGSI. (See Section 3.4.) In the event that jump boundary conditions $[p_i]$ and $[D_i \nabla p_i \cdot \mathbf{n}]$ are prescribed, use the ghost cell extrapolations described in Section 3.3.

Step 3: Calculate and extend the normal velocity V:

- **Part a:** Calculate ∇p_i for each i in the narrow band $\{\mathbf{x} : |\phi(\mathbf{x})| \leq R\}$ using second-order centered differences if all points in the stencil are in the same region (all inside Ω or all outside Ω). Otherwise, use high-order, one-sided Taylor expansions. In our work, we use five-point stencils when possible, and degrade to lower-order one-sided stencils as necessary.
- **Part b:** Use our bilinear velocity extension from [38] to create a normal extension $\widetilde{V}$ of $V = \sum \alpha_i \nabla p_i \cdot \mathbf{n}$.
- **Part c:** Problems with geometric boundary conditions (e.g., those that depend upon the curvature κ) require that $\Delta t \sim \Delta x^3$ to maintain stability. To avoid this prohibitive time step restriction, apply the Gaussian filtering technique we developed in [36] and [38] to the extended velocity $\widetilde{V}$, which removes high-frequency noise from the velocity. Note that if the jump boundary conditions of the p_i do not require the curvature, then this filtering may not be required.
- **Part d:** Use our bilinear velocity extension from [38] to extend the filtered velocity.
- **Part e:** Calculate the CFL condition $\Delta t = \frac{\Delta x}{2\max(\widetilde{V})}$.

Traditional Stencil	
Δx	ℓ_∞ error
0.16	0.168897
0.08	0.170776
0.04	0.168131
order	0.00751

New Stencil (1st order)	
Δx	ℓ_∞ error
0.16	0.00295862
0.08	5.11947e-4
0.04	9.22828e-5
order	2.50

New Stencil (Higher order)	
Δx	ℓ_∞ error
0.16	0.00331730
0.08	5.73877e-4
0.04	8.18444e-5
order	2.67

Table 1 Comparison of the $[D\nabla p \cdot \mathbf{n}]$ stencil for the traditional (**left**), first-order new (**middle**), and higher-order new (**right**) methods on Example 1.

Step 4: Construct $\widetilde{V}\,|\nabla\phi|$ using fifth-order WENO. Use this to advect the interface Σ.

Step 5: (*Optional*) If using a higher-order Runge-Kutta approximation, repeat steps (2)-(4) for each part of the Runge-Kutta scheme.

4 Convergence of the Numerical Techniques

We now present convergence results for the newly developed numerical techniques and the overall scheme. For a given norm $||\cdot||$, we define two orders of convergence. If p_h is a numerical solution computed on a computational grid with mesh length h, and if the exact solution p is known, then we define

$$\text{order of convergence} = \frac{\log\left(\frac{||p_{h_1}-p||}{||p_{h_2}-p||}\right)}{\log\left(\frac{h_1}{h_2}\right)} \tag{57}$$

where the norm is computed at the computational node points, and $h_2 < h_1$ are two different mesh lengths.

If the exact solution is unknown, then we solve on meshes with mesh lengths $h_1 = h$, $h_2 = \frac{1}{2}h$, and $h_3 = \frac{1}{4}h$, and we compute the order of convergence via

$$\text{order of convergence} = \frac{\log\left(\frac{||p_h-p_{h_2}||}{||p_{h_2}-p_{h_3}||}\right)}{\log(2)}, \tag{58}$$

where each norm is computed on the common grid points. In our work, we use the discrete maximum ℓ_∞ norm for all convergence testing.

4.1 Convergence of the Ghost Cell Method with the New $[D\nabla p \cdot \mathbf{n}]$ stencils and NAGSI

Example 1: An Example with Large Tangential and Normal Jumps:

To test the convergence and impact of our new normal derivative jump discretization, we studied the following problem

$$\nabla^2 p = 0 \qquad \text{if } |\mathbf{x}| < 1 \tag{59}$$
$$\nabla^2 p = 0 \qquad \text{if } |\mathbf{x}| > 1 \tag{60}$$
$$[p] = x + y \qquad \text{if } |\mathbf{x}| = 1 \tag{61}$$
$$[\nabla p \cdot \mathbf{n}] = x + y \qquad \text{if } |\mathbf{x}| = 1 \tag{62}$$
$$u = 0 \qquad \text{if } \mathbf{x} \in \partial\left([-2,2] \times [-2,2]\right), \tag{63}$$

whose solution is

$$p(x,y) = \begin{cases} x + y & \text{if } |\mathbf{x}| \leq 1 \\ 0 & \text{else.} \end{cases} \tag{64}$$

Note that the solution in (64) has a nonzero jump in tangential derivative:

$$[\nabla p \cdot \mathbf{s}] = (1,1) \cdot (y, -x) = y - x, \tag{65}$$

where $\mathbf{s}$ is the positively-oriented tangent vector along boundary of the circle.

We solved this system with mesh sizes $\Delta x \in \{0.16, 0.08, 0.04\}$, using our new NAGSI solver along with the ghost cell extrapolations described above, using both the traditional, grid-aligned normal derivative stencil from [34] and the new normal derivative jump stencils. The traditional stencil from [34] (as stabilized in [40]) failed to converge (left part of Table 1), with visible distortions in the solution (blue squares in Figure 7.) In contrast, when we recomputed the solution using our new normal derivative jump stencils, we obtained 2.50- and 2.67-order convergence for our first-order and higher-order stencils, respectively (middle and right parts of Table 1, respectively). Thus, we see that preserving the accuracy in the tangential derivative jump can have a substantial impact on the overall accuracy of the solution and is necessary for convergence in this example.

Example 2: An Example with a Large Tangential Jump and no Normal Jump:

We solved the same system as in the previous example, but with

$$[\nabla p \cdot \mathbf{n}] = 0 \quad \text{if } |\mathbf{x}| = 1. \tag{66}$$

In this example, the traditional stencil converged, but only to order 0.319 (top part of Table 2). In contrast, our first-order method attained 1.11-order accuracy (lower left part of Table 2), and our higher-order method achieved 1.53-order convergence (lower right part of Table 2).

Example 3: An Example with Large Normal Jumps and Zero Tangential Jumps:

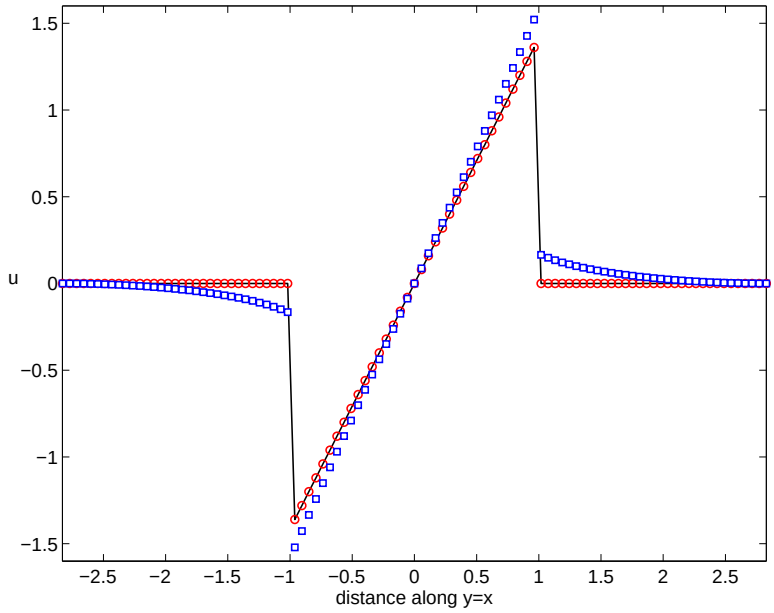

Fig. 7 Comparison of solutions along the line $y = x$ using the traditional (blue squares) and new (red circles) $[D\nabla p \cdot \mathbf{n}]$ stencils using $\Delta x = 0.04$. The exact solution is given by the solid black line.

difference	ℓ_∞ norm
$p_{0.16} - p_{0.08}$	0.0282768
$p_{0.08} - p_{0.04}$	0.0226730
order	0.319

difference	ℓ_∞ norm
$p_{0.16} - p_{0.08}$	0.0201711
$p_{0.08} - p_{0.04}$	0.00936549
order	1.11

difference	ℓ_∞ norm
$p_{0.16} - p_{0.08}$	0.00655476
$p_{0.08} - p_{0.04}$	0.00226471
order	1.53

Table 2 Comparison of the $[D\nabla p \cdot \mathbf{n}]$ stencil for the traditional (**top**) and new first-order (**bottom left**) and higher-order (**bottom right**) methods on Example 2.

To further investigate the accuracy of our new normal jump stencil, we studied the following problem:

$$\nabla^2 p = -4 \qquad \text{if } |\mathbf{x}| < 1 \tag{67}$$

$$\nabla^2 p = 0 \qquad \text{if } |\mathbf{x}| > 1 \tag{68}$$

$$[p] = 0 \qquad \text{if } |\mathbf{x}| = 1 \tag{69}$$

$$[\nabla p \cdot \mathbf{n}] = -2 \qquad \text{if } |\mathbf{x}| = 1 \tag{70}$$

Traditional Stencil	
Δx	ℓ_∞ error
0.16	0.647074
0.08	0.540950
0.04	0.458996
order	0.248

New (1st Order) Stencil	
Δx	ℓ_∞ error
0.16	0.244280
0.08	0.101704
0.04	0.0492541
order	1.16

New (Higher Order) Stencil	
Δx	ℓ_∞ error
0.16	0.0401866
0.08	0.00830349
0.04	0.00312614
order	1.84

Table 3 Comparison of the $[D\nabla p \cdot \mathbf{n}]$ stencil for the traditional (**left**) and new first-order (**middle**) and higher-order (**right**) methods on Example 3.

$$p = 0 \qquad \text{if } \mathbf{x} \in \partial\left([-2,2] \times [-2,2]\right), \tag{71}$$

whose solution is

$$p(x,y) = \begin{cases} 1 - x^2 - y^2 & \text{if } |\mathbf{x}| \leq 1 \\ 0 & \text{else.} \end{cases} \tag{72}$$

Due to the jump boundary condition, this problem is sensitive to the discretization of the discontinuous source term: error in the numerical integral of the source term inside $|\mathbf{x}| \leq 1$ will vertically shift the quadratic (interior) part of the solution, and due to the coupling of the interior and exterior normal derivatives, error in the source term will lead to error in the exterior region as well. We treat the discontinuous source term by solving

$$0 = \nabla^2 p + 4\, H(-\phi), \qquad \mathbf{x} \in [-2,2] \times [-2,2], \tag{73}$$

and we discretize H with a numerical Heaviside function $\widetilde{H}$. See Appendix A for further discussion on the choice of $\widetilde{H}$.

Notice that this solution has no jump in the tangential derivative. Nonetheless, the traditional normal jump stencil (again, as stabilized in [40]) yields sub-first-order convergence (left part of Table 3), whereas our new stencil is first-order accurate (middle part of Table 3), and the higher-order method is 2.65-order accurate. Thus, we can see that the failure of the traditional $[D\nabla p \cdot \mathbf{n}]$ stencil to properly separate the normal and tangential jumps degrades the accuracy of the entire solution, even in the absence of a tangential derivative jump.

4.2 Convergence of the Overall Method

We examined the convergence of the overall method by studying a drop moving under Hele-Shaw-type flow in an heterogeneous medium. Let Σ be the boundary of a circle of radius 1 centered at $\left(5\sqrt{2}, 5\sqrt{2}\right)$.

We represent Σ as the zero contour of a level set function ϕ on the computational domain $\mathcal{D} = [0,10] \times [0,10]$ containing both $\Omega = \{\mathbf{x} \in \mathcal{D} : \phi(\mathbf{x}) < 0\}$ and its complement $\Omega^c = \{\mathbf{x} \in \mathcal{D} : \phi(\mathbf{x}) > 0\}$.

The drop has normal velocity

$$V = -\mu \nabla P \cdot \mathbf{n} \quad \text{if } \mathbf{x} \in \Sigma, \tag{74}$$

which we implement in the level set context as

$$\phi_t + \widetilde{V} \left| \nabla \phi \right| = 0, \tag{75}$$

where $\widetilde{V}$ is the normal extension of the velocity off of Σ that we described in [38].

The pressure P satisfies

$$0 = \nabla \cdot (\mu \nabla P) + H(-\phi) \quad \text{if } \mathbf{x} \in \Omega \cup \Omega^c \tag{76}$$

$$[P] = \kappa \quad \text{if } \mathbf{x} \in \Sigma \tag{77}$$

$$[\mu \nabla P \cdot \mathbf{n}] = 0 \quad \text{if } \mathbf{x} \in \Sigma \tag{78}$$

$$\tag{79}$$

with

$$\mu = \eta + (1-\eta) e^{-1.5\left(\sqrt{x^2+y^2}-5\sqrt{2}\right)^8} \quad \text{if } \mathbf{x} \in \mathcal{D} \tag{80}$$

$$\eta = 0.0001, \tag{81}$$

and with boundary conditions

$$P \equiv 0 \quad \text{on } \partial D. \tag{82}$$

See Figure 8 for a plot of the permeability μ. Note that this models the growth of a drop of incompressible fluid in a heterogeneous domain, where fluid is added at a constant rate throughout the drop domain via the Heaviside source term $H(-\phi)$.

We chose three spatial resolutions: $\Delta x = \Delta y = 0.20$ (low resolution), $\Delta x = \Delta y = 0.10$ (medium resolution), and $\Delta x = \Delta y = 0.05$ (high resolution). We used the discrete numerical Heaviside function that we describe in Appendix A. We used a simple, first-order forward Euler time discretization with CFL condition

$$\Delta t \leq \max\left(\frac{\Delta x}{2 \max |V|}, 0.05 \right). \tag{83}$$

and fifth-order WENO for $V \left| \nabla \phi \right|$. As in [36,38], we used Gaussian filtering to attain a first-order CFL stability condition while maintaining accuracy. In these simulations, we used a smoothing parameter (standard deviation) of $\sigma = 2\Delta x = 0.40$ for low resolution, $\sigma = 3\Delta x = 0.30$ for medium resolution, and $\sigma = 4\Delta x = 0.15$ for high resolution.

We calculated the order order of convergence (of the level set function ϕ) at times $t \in \{0.10, 0.20, 0.30, 0.40, 0.50\}$ according to (58). Due to the small time step, the time error is small and is consequently dominated by the spatial error. The average order of convergence was 1.86. See Table 4. We shall discuss the behavior of the solution in Section 5.1.

t	$\|\|\phi_{0.2} - \phi_{0.1}\|\|_\infty$	$\|\|\phi_{0.1} - \phi_{0.05}\|\|_\infty$	order
0.1	0.0504799	0.0184189	1.45
0.2	0.0946371	0.0179485	2.40
0.3	0.0665503	0.0396410	0.747
0.4	0.130442	0.0342100	1.93
0.5	0.231275	0.0335707	2.78
average order of convergence			1.86

Table 4 Convergence of the Overall Method

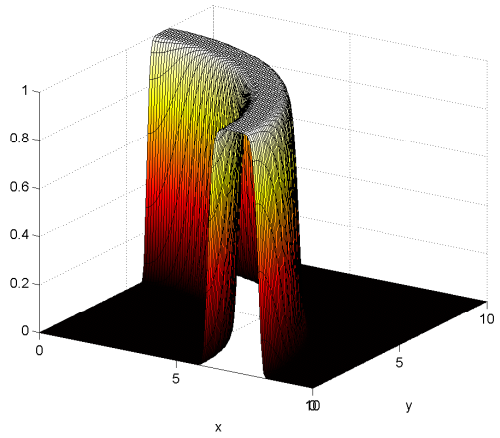

Fig. 8 Permeability μ for the convergence example.

5 Numerical Examples

5.1 Hele-Shaw-type Flow in Heterogeneous Media

We solved the Hele-Shaw-type problem from the overall convergence study with $\Delta x = \Delta y = 0.10$, $\mathcal{D} = [0, 10] \times [0, 10]$, and the permeability μ shown in Figure 8 until $t = 1.8$. The solution is plotted in 0.20 time increments in Figure 9. As volume is added to the drop Ω (via the Heaviside source term in the pressure equation), pressure builds inside the drop that pushes the boundary Σ outward and causes the region Ω to grow. Because the permeability μ varies between 0.0001 and 1 throughout the domain, the drop grows preferentially inside the region where $\mu \sim 1$. See Figure 9. Because volume is added at a constant rate throughout the drop, the rate of growth is proportional to the volume of the drop; this can be observed as an increasing distance between solution curves.

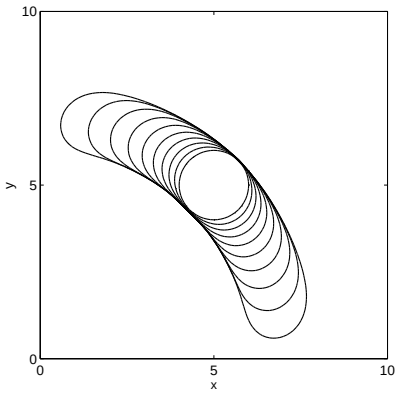

Fig. 9 Outward growth of the medium-resolution ($\Delta x = \Delta y = 0.10$) solution in $t = 0.20$ increments for the convergence example.

5.2 Tumor Growth in Heterogeneous Tissues

We now demonstrate our technique by applying it to a nonlinear tumor growth problem. Let ϕ be a level set function whose zero contour denotes the boundary Σ of an avascular tumor $\Omega = \Omega(\mathbf{x}, t)$ growing into a surrounding, non-cancerous tissue $\Omega^c = \Omega^c(\mathbf{x}, t)$. This models the early stage of *in vivo* growth before the onset of angiogenesis. We take our computational domain $\mathcal{D}$ to be a rectangular region that fully contains Ω and Ω^c; note that $\mathcal{D} = \Omega \cup \Omega^c$.

Let c denote the non-dimensionalized nutrient concentration within the computational domain, scaled by the far-field nutrient value in well-vascularized, non-pathological tissue. We scale space by the oxygen diffusional length scale $L \approx 200\ \mu$m. Outside the tumor, the blood vasculature (with density B) delivers nutrient, which diffuses into the tumor and is consumed by proliferating cells. As the tumor grows, less nutrient reaches the interior, until it drops to a level c_H where tumor cells become hypoxic. The hypoxic tumor cells become quiescent and consume less nutrient. If the tumor continues to grow and the interior nutrient level drops further below a critical threshold c_N, the tumor cells begin to die (necrose). When cells necrose, they release their cellular contents, which are both oxygen-reactive (e.g., see [31,20]) and growth-inhibiting. These processes can be modeled as

$$0 = \nabla \cdot (D \nabla c) - \lambda^c(\mathbf{x}, c)c + \lambda^c_{bulk}(1 - c)B(\mathbf{x})H(\phi) \quad \text{if } \mathbf{x} \in \Omega \tag{84}$$

$$\nabla c \cdot \mathbf{n} = 0 \quad \text{if } \mathbf{x} \in \partial\mathcal{D}, \tag{85}$$

where B is the pre-existing blood vessel density, λ^c_{bulk} is the nutrient delivery rate in the pre-existing blood vasculature, D is the nutrient diffusivity, and $\lambda^c(\mathbf{x}, c)$ is chosen to combine the rates of nutrient uptake (in the viable and hypoxic

portion of the tumor) and decay (in the necrotic portion of the tumor); we assume no nutrient uptake in the non-cancerous tissue Ω^c. Normalized by the nutrient uptake rate in the viable region, the uptake and decay function is modeled by

$$\lambda^c(\mathbf{x}, c) = \begin{cases} 0 & \text{if } \mathbf{x} \in \Omega^c \\ 1 & \text{if } \mathbf{x} \in \Omega \text{ and } 1 \geq c > c_H \\ q(c) & \text{if } \mathbf{x} \in \Omega \text{ and } c_H \geq c > c_N \\ \lambda_N^c & \text{if } \mathbf{x} \in \Omega \text{ and } c_N \geq c. \end{cases} \tag{86}$$

Here, λ_N^c is the rate of nutrient decay in the necrotic core, and $q(c)$ is a polynomial that smoothly connects regions and is chosen to satisfy

$$\begin{aligned} q(c_H) &= 1 \\ q'(c_H) &= 0 \\ q\left(\frac{c_H + c_N}{2}\right) &= \lambda_H^c \\ q(c_N) &= \lambda_N^c \end{aligned} \tag{87}$$

$$q'(c_N) = 0, \tag{88}$$

where λ_H^c is the rate of nutrient uptake by hypoxic cells. In our numerical example, we shall take $c_H = 0.3$, $c_N = 0.2$, $\lambda_H^c = 0.5$, and $\lambda_N^c = 0.25$. In this case,

$$q(c) = -20000c^4 + 18500c^3 - 6275c^2 + 930c - 50.75. \tag{89}$$

Notice that this makes our nutrient equation nonlinear.

We model the tumor as an incompressible fluid growing in a porous medium, and so the local rate of change in tumor volume is given by $\nabla \cdot \mathbf{u}$, where $\mathbf{u}$ is the cellular velocity field. Proliferating tumor cells in the viable rim of the tumor generate an internal biomechanical pressure p that increases the tumor volume (at a rate proportional to the nutrient level c) and pushes the tumor boundary outward with normal velocity V via Darcy's law ($\mathbf{u} = -\mu \nabla p$). The enzymatic breakdown of necrotic tumor tissue is modeled by a local decrease in the pressure that reduces volume (at a constant rate G_N) and slows growth. Cell-to-cell adhesion is modeled as a surface tension (curvature) boundary condition on Σ. The non-cancerous tissue in Ω^c is also assumed to be affected by the tumor-generated pressure, but the cells in Ω^c do not proliferate. We model the pressure by

$$\nabla \cdot \mathbf{u} = -\nabla \cdot (\mu \nabla p) = \begin{cases} 0 & \text{if } \mathbf{x} \in \Omega^c \\ c & \text{if } \mathbf{x} \in \Omega \text{ and } c > c_H \\ 0 & \text{if } \mathbf{x} \in \Omega \text{ and } c_H \geq c > c_N \\ -G_N & \text{if } \mathbf{x} \in \Omega \text{ and } c_N \geq c \end{cases} \tag{90}$$

with boundary conditions

$$[p] = \frac{1}{G}\kappa \quad \text{if } \mathbf{x} \in \Sigma \tag{91}$$

$$[\mu \nabla p \cdot \mathbf{n}] = 0 \quad \text{if } \mathbf{x} \in \Sigma. \tag{92}$$

We model the normal velocity of the tumor boundary by Darcy's law:

$$V = -\mu \nabla p \cdot \mathbf{n} \quad \text{if } \phi(\mathbf{x}) = 0. \tag{93}$$

We choose boundary conditions on p along $\partial\mathcal{D}$ to suit the problem under study. Here, G that characterizes the tumor aggressiveness (the rate of proliferation compared to the cell-cell adhesion time scale), G_N is a parameter that governs the rate of tumor cell breakdown in necrotic regions, and μ is the cellular mobility. This tumor growth model is an extension of current models given in [15], [36], [38], [57], [39] and [40], and will be further extended and investigated in future work [35,6].

We simulated this tumor system on a computational domain $\mathcal{D} = [0, 50] \times [0, 50]$ with $\Delta x = \Delta y = 0.10$, with tissue and tumor properties chosen to model the evolution of *glioblastoma* in brain tissue. Because the oxygen diffusional length scale is approximately 200 μm, this corresponds to an approximately 1 cm square of simulated brain tissue. We set $G = 20$, $G_N = 1$, $\lambda^c_{bulk} = 1$, $c_H = 0.3$, $c_N = 0.2$, and used $q(c)$ as given in (89). We model growth in a complex, heterogeneous brain tissue as shown in the first frame in Figure 10. In the white region, $\mu = 0.0001$, $D = 0.0001$, and $B = 0$, which models a rigid material such as the skull. In the black regions, $\mu = 10$, $D = 1$, and $B = 0$, which models an incompresible fluid (cerebrospinal fluid). The light and dark gray regions model tissues of differing biomechanical properties (white and gray matter). In the light gray regions, $\mu = 1.5$, $D = 1$, and $B = 1$; in the dark gray regions, $\mu = 0.5$, $D = 1$, and $B = 1$. The red region (color images are available online) denotes the initial shape and position of the simulated tumor. We smoothed μ, B, and D using a Gaussian filter with standard deviation $\sigma = 3\Delta x = 0.3$ to satisfy smoothness requirements of the reaction-diffusion equations. We used (linear) extrapolation boundary conditions on the pressure along $x = 0$, $y = 0$, and $y = 50$ to simulated growth into a larger, unshown tissue, and we set $p = 0$ along the rigid boundary at $x = 50$.

We simulated from $t = 0$ to $t = 60$. Using a 3.3 GHz Pentium 4 workstation and a C++ implementation, the 501×501 simulation required under 24 hours to compute. Because our (mitosis) time scale ranges from approximately 18 to 36 hours for this problem, this corresponds to 45 to 90 days of growth. We plot our solution in $t = 5.0$ (approximately 5 days) increments in Figures 10 and 11. In those plots, red regions correspond to viable tumor tissue (where $c > c_H$), blue regions denote hypoxic tumor tissue ($c_H \geq c > c_N$), and brown regions denote necrotic tumor cells ($c_N \geq c$). (Please see the online version of the article for color figures.) In this simulation, the tumor grows rapidly until the nutrient level drops below $c_H = 0.30$ (see $t = 5.0$), at which time a large portion of the tumor becomes hypoxic. The tumor continues to grow at a slower rate until the interior of the tumor becomes necrotic. (See $t = 10.0$.) This causes non-uniform volume loss within the tumor and contributes to morphological instability. We note that because the biomechanical responsiveness is continuous across the tumor boundary and the microenvironment has a moderate nutrient gradient, this simulation

corresponds to the border between the invasive, fingering growth regime and the invasive, fragmenting growth regime that we investigated in [40].

However, additional effects can be seen that were not observed in the aforementioned study. As the tumor grows out of the biomechanically permissive tissue (light gray; $\mu = 1.5$) and into the biomechanically resistant tissue (dark gray; $\mu = 0.5$), its rate of invasion into the tissue slows. (See $t = 15$ to 25.0.) This results in preferential growth into the permissive (light gray) material, a trend which can be clearly seen from $t = 30.0$ onward. When the tumor grows through the resistive tissue (dark gray) and reaches the fluid (black), the tumor experiences a sudden drop in biomechanical resistance to growth. As a result, the tumor grows rapidly and preferentially in the 1/2 mm fluid structures that separate the tissue. Such growth patterns are not observed when simulating homogeneous tissues.

Other observed differences are due to our new treatment of hypoxic (quiescent) tumor cells. Certain regions that we had previously classified as necrotic (in [36,38–40]) are now treated as quiescent. As a result, tumor volume loss is reduced, and in particular, this may result in large hypoxic regions that have little or no viable rim. Had these regions been treated as necrotic, the invasive fingers would have been thinner, and the tumor may have fragmented. Therefore, the separate treatment of the hypoxic regions can have a significant impact on the details of the invasive morphology of the tumor. We shall investigate this effect in greater detail in future work.

6 Conclusions and Future Work

In this paper, we built upon our earlier work from [36], [38], [39] and [40] to develop an accurate ghost cell/level set technique for evolving interfaces whose normal velocity is given by the normal derivatives of solutions to linear and nonlinear quasi-steady reaction-diffusion equations with curvature-dependent boundary conditions. The technique is capable of describing complex morphologies evolving in heterogeneous domains. The algorithm involved several new developments, including a new ghost cell technique for accurately discretizing jumps in the normal derivative without smearing jumps in the tangential derivative, a new adaptive solver for linear and nonlinear quasi-steady reaction-diffusion problems (NAGSI), an adaptive normal vector discretization for interfaces in close contact, and an accurate discrete approximation to the Heaviside function.

We demonstrated the accuracy, efficiency, and capabilities of the method on a variety examples. For instance, we considered a model of solid tumor growth consisting of a fully nonlinear reaction-diffusion equation for the nutrient and a pressure equation that includes geometric boundary conditions. We solved the tumor system in a heterogeneous environment including complex structures (white matter, gray matter, cerebrospinal fluid, and bone), much like human brain tissue. We observed growth morphologies that were highly dependent upon the variations in the cellular mobility and the nutrient delivery–an effect observed in

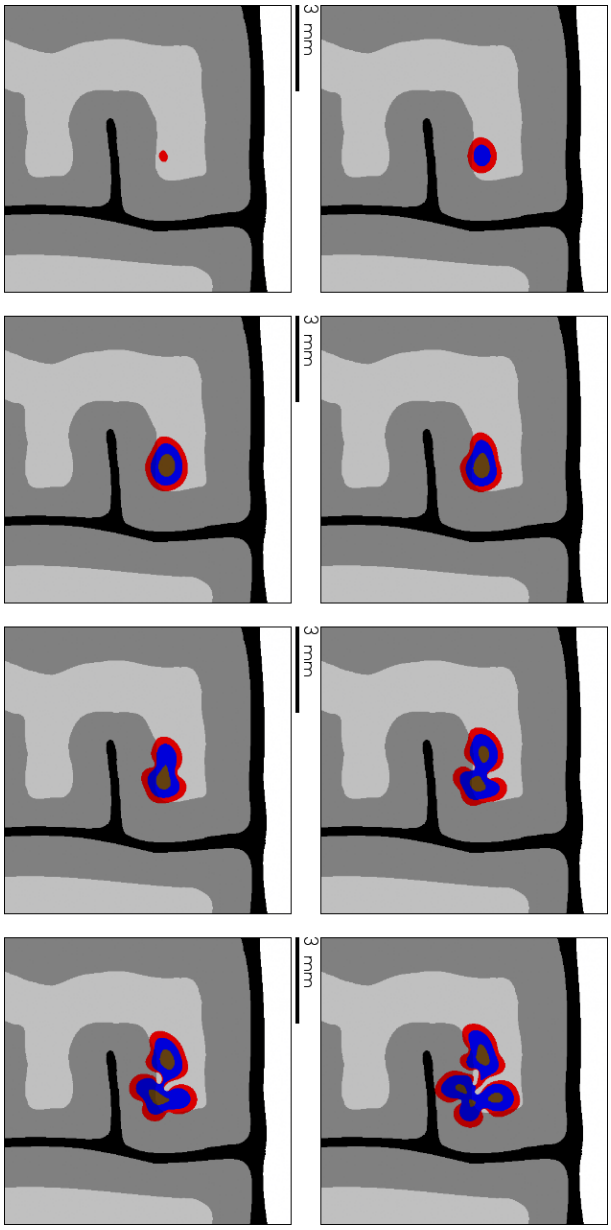

Fig. 10 Long-time tumor simulation from t = 0.0 days (top left) to t = 35.0 days (bottom right) in 5 day increments. The color version of this figure is available online.

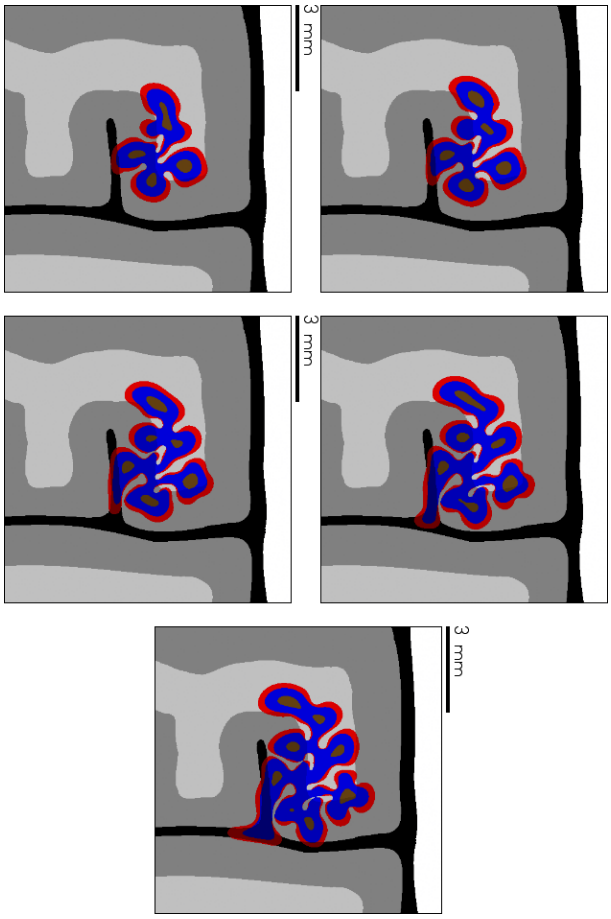

Fig. 11 Long-time tumor simulation (continued) from t = 40.0 days (top left) to t = 60.0 days (bottom) in 5 day increments. The color version of this figure is available online.

real tumor growth. The accuracy of the algorithm is a key step in the development of a new generation of predictive tumor growth models that can eventually lead to clinical applications. In future work, we will conduct a more thorough study of tumor growth in inhomogeneous tissue, investigate models of tumor-microenvironment coupling that include active remodeling of the tissue by the tumor [35], and study the effects of coupling tumor growth to complex models of angiogenesis with Alexander Anderson, Mark Chaplain, Steven McDougall, and Vittorio Cristini.

Acknowledgements

We thank the National Science Foundation Division of Mathematical Sciences and the UCI Department of Mathematics for their financial support. John Lowengrub is partially supported by the NIH grant P50GM76516 for a National Center of Excellence in Systems Biology at UCI. Paul Macklin was partially supported by a U.S. Department of Education GAANN (Graduate Assistance in Areas of National Need) fellowship.

We thank Alexander "Sandy" Anderson and Mark Chaplain at the University of Dundee, Steven McDougall at Heriot-Watt University, and Vittorio Cristini at the University of Texas Health Science Center in Houston for valuable discussions. We thank the University of Dundee and Heriot-Watt University for their gracious hospitality during numerous visits in 2006 and 2007.

A A Simple Numerical Heaviside Function

Given a discretized domain with grid points $\{x_i\}_{i=0}^m \times \{y_j\}_{j=0}^n$ and a function f defined on those node points, we wish to define a numerical Heaviside function $\widetilde{H}(f) = H_{i,j}(f)$ on the computational node points which approximates the true Heaviside function

$$H(f) = \begin{cases} 0 & \text{if } f < 0 \\ 1 & \text{if } f \geq 0 \end{cases} \tag{94}$$

and such that

$$H_{i,j}(f)\Delta x \Delta y \approx \int_{x_i - \frac{1}{2}\Delta x}^{x_i + \frac{1}{2}\Delta x} \int_{y_j - \frac{1}{2}\Delta y}^{y_j + \frac{1}{2}\Delta y} H\left(f(s,t)\right)\, ds\, dt, \tag{95}$$

i.e., the numerical Heaviside function approximates the percentage of the computational node centered at (x_i, y_j) that is occupied by the region $\{\mathbf{x} : f(\mathbf{x}) > 0\}$. We accomplish this by examining the value of f at (i, j) and the eight surrounding computational nodes:

$$H_{i,j}(f) = \sum_{s=i-1}^{i+1} \sum_{t=j-1}^{j+1} w_{|s-i|,|t-j|} H(f_{i,j}), \tag{96}$$

where the weights are given by the relative areas of the sub-grid sections in Figure 12, and can be written as

$$w_{m,n} = \frac{2^{2-m-n}}{16} = \begin{cases} \frac{1}{16} & \text{if } m = 1 \text{ and } n = 1 \\ \frac{1}{8} & \text{if } m = 1 \text{ and } n = 0 \\ \frac{1}{8} & \text{if } m = 0 \text{ and } n = 1 \\ \frac{1}{4} & \text{if } m = 0 \text{ and } n = 0. \end{cases} \tag{97}$$

See Figure 12. For the function in the figure, $H_{i,j}(f) = \frac{11}{16}$.

To study the accuracy of this numerical Heaviside function, we calculated the order of the convergence for third ghost cell method example in Section 4.1 using our discrete

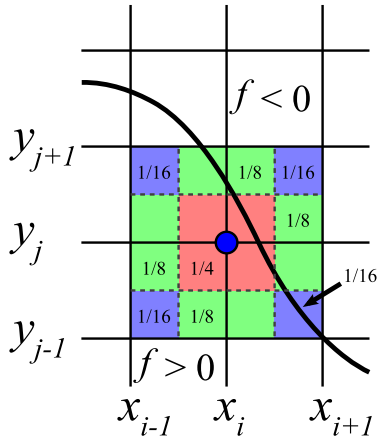

Fig. 12 The weights w used for computing our numerical Heaviside function at a node (i, j).

Heaviside function. For comparison, we also considered the Heaviside function from Sussman and Fatemi [52]

$$\widetilde{H}_\delta(f) = \begin{cases} 0 & \text{if } \ f < -\delta \\ \frac{1}{2}\left[1 + \frac{f}{\delta} + \frac{1}{\pi}\sin\left(\frac{\pi f}{\delta}\right)\right] & \text{if } \ |f| \le \delta \\ 1 & \text{if } \ f > \delta \end{cases} \tag{98}$$

with $\delta = \Delta x$, and the Heaviside function function in recent work by Engquist et al. [16]:

$$\widetilde{H}_\delta(f) = \begin{cases} 0 & \text{if } \ f \le -\delta \\ \frac{1}{2}\left(1 + \frac{f}{\delta}\right) & \text{if } \ |f| < \delta \\ 1 & \text{if } \ f \ge \delta, \end{cases} \tag{99}$$

where we used $\delta = \Delta x$.

The results, given in Table 5, demonstrate approximately first-order convergence of the problem using the Sussman and Fatemi Heaviside function (left part of Table 5), slightly better than first-order convergence for the Engquist et al. Heaviside function (middle part of Table 5), and near-second-order convergence for our discrete Heaviside function (right part of Table 5). The key to this problem is the accurate approximation of the source term without smearing the integral of the source outside of the circle $|\mathbf{x}| \le 1$. All three methods approximate the area of the circle to second-order accuracy (not shown), but only the discrete Heaviside approximation was designed to accurately approximate the area of the region locally as well as globally.

References

1. Abbott, R.G., Forrest, S., Pienta, K.J.: Simulating the Hallmarks of Cancer. Artif. Life **12**(4), 617–34 (2006). DOI 10.1162/artl.2006.12.4.617

Sussman and Fatemi [52]	
Δx	ℓ_∞ error
0.16	0.0269720
0.08	0.00776372
0.04	0.00671369
order	1.00

Engquist et al. [16]	
Δx	ℓ_∞ error
0.16	0.0307405
0.08	0.00843434
0.04	0.00598091
order	1.18

discrete Heaviside function	
Δx	ℓ_∞ error
0.16	0.0401866
0.08	0.00830349
0.04	0.00312614
order	1.84

Table 5 Convergence of a ghost cell problem using a Heaviside function, computed with the continuous approximation by Sussman and Fatemi in [52] (left), the continuous approxmation by Engquist et al. in [16] (middle), and our discrete Heaviside approximation (right).

2. Adalsteinsson, D., Sethian, J.A.: The Fast Construction of Extension Velocities in Level Set Methods. J. Comput. Phys. **148**(1), 2–22 (1999). DOI 10.1006/jcph.1998.6090
3. Adalsteinsson, D., Sethian, J.A.: The Fast Construction of Extension Velocities in Level Set Methods. J. Comput. Phys. **148**(1), 2–22 (1999). DOI 10.1006/jcph.1998.6090
4. Adam, J.: General aspects of modeling tumor growth and the immune response. In: J. Adam, N. Bellomo (eds.) A survey of models on tumor immune systems dynamics, pp. 15–87. Birkhauser, Boston, MA (1996)
5. Ambrosi, D., Preziosi, L.: On the closure of mass balance models for tumor growth. Math. Mod. Meth. Appl. Sci. **12**(5), 737–754 (2002). DOI 10.1142/S0218202502001878
6. Anderson, A.R.A., Chaplain, M.A.J., Lowengrub, J.S., Macklin, P., McDougall, S.: Nonlinear Simulation of Tumour Invasion and Angiogenesis. Bull. Math. Biol. (2007). (in preparation)
7. Anderson, A.R.A., Weaver, A.M., Cummings, P.T., Quaranta, V.: Tumor Morphology and Phenotypic Evolution Driven by Selective Pressure from the Microenvironment. Cell **127**(5), 905–915 (2006). DOI 10.1016/j.cell.2006.09.042
8. Araujo, R.P., McElwain, D.L.S.: A history of the study of solid tumor growth: The contribution of mathematical modeling. Bull. Math. Biol. **66**(5), 1039–1091 (2004). DOI 10.1016/j.bulm.2003.11.002
9. Bellomo, N., de Angelis, E., Preziosi, L.: Multiscale modelling and mathematical problems related to tumor evolution and medical therapy. J. Theor. Med. **5**(2), 111–136 (2003). DOI 10.1080/1027336042000288633
10. Byrne, H., Preziosi, L.: Modelling solid tumour growth using the theory of mixtures. Math. Med. Biol. **20**(4), 341–366 (2003). DOI 10.1093/imammb/20.4.341
11. Byrne, H.M., Alarcón, T., Owen, M.R., Webb, S.D., Maini, P.K.: Modeling aspects of cancer dynamics: A review. Phil. Trans. R. Soc. A **364**(1843), 1563–1578 (2006). DOI 10.1098/rsta.2006.1786
12. Chaplain, M.A.J., Graziano, L., Preziosi, L.: Mathematical modelling of the loss of tissue compression responsiveness and its role in solid tumour development. Math. Med. Biol. **23**(3), 192–229 (2006). DOI 10.1093/imammb/dql009
13. Chatterjee, C., Chong, E.K.P.: Efficient Algorithms for Finding the Centers of Conics and Quadrics in Noisy Data. Patt. Recog. **30**(5), 673–684 (1997). DOI 10.1016/S0031-3203(96)00122-7
14. Cowles, G., Martinelli, L.: Control theory based shape design for the incompressible navier-stokes equations. Int. J. Comput. Fluid Dyn. **17**(6), 499–514 (2003). DOI 10.1080/10618560310001614773
15. Cristini, V., Lowengrub, J.S., Nie, Q.: Nonlinear simulation of tumor growth. J. Math. Biol. **46**, 191–224 (2003). DOI 10.1007/s00285-002-0174-6

16. Engquist, B., Tornberg, A.K., Tsai, R.: Discretization of Dirac delta functions in level set methods. J. Comput. Phys. **207**(1), 28–51 (2005). DOI 10.1016/j.jcp.2004.09.018
17. Fedkiw, R.P., Aslam, T., Merriman, B., Osher, S.: A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method). J. Comput. Phys. **152**(2), 457–492 (1999). DOI 10.1006/jcph.1999.6236
18. Frieboes, H.B., Lowengrub, J.S., Wise, S., Zheng, X., Macklin, P., Bearer, E.L., Cristini, V.: Computer simulation of glioma growth and morphology. NeuroImage **37**(S1), S59–S70 (2007). DOI 10.1016/j.neuroimage.2007.03.008
19. Frieboes, H.B., Wise, S.M., Lowengrub, J.S., Cristini, V.: Three-dimensional Diffuse-Interface Simulation of Multispecies Tumor Growth-II: Investigation of Tumor Invasion. Bull. Math. Biol. (2006). (in review)
20. Galaris, D., Barbouti, A., Korantzopoulos, P.: Oxidative Stress in Hepatic Ischemia-Reperfusion Injury: The Role of Antioxidants and Iron Chelating Compounds. Current Pharma. Design **12**(23), 2875–2890 (2006). DOI 10.2174/138161206777947614
21. Gibou, F., Fedkiw, R.: A Fourth Order Accurate Discretization for the Laplace and Heat Equations on Arbitrary Domains, with Applications to the Stefan Problem. J. Comput. Phys. **202**(2), 577–601 (2005). DOI 10.1016/j.jcp.2004.07.018
22. Gibou, F., Fedkiw, R., Caflisch, R., Osher, S.: A Level Set Approach for the Numerical Simulation of Dendritic Growth. J. Sci. Comput. **19**(1-3), 183–199 (2003). DOI 10.1023/A:1025399807998
23. Gibou, F., Fedkiw, R., Cheng, L.T., Kang, M.: A Second Order Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. J. Comput. Phys. **176**(1), 205–227 (2002). DOI 10.1006/jcph.2001.6977
24. Glimm, J., Marchesin, D., McBryan, O.: A Numerical-Method for 2 Phase Flow with an Unstable Interface. J. Comput. Phys. **39**, 179–200 (1981). DOI 10.1016/0021-9991(81)90144-3
25. Gottlieb, S., Shu, C.W.: Total Variation Diminishing Runge-Kutta Schemes. Math. Comp. **67**(221), 73–85 (1997). DOI 10.1090/S0025-5718-98-00913-2
26. Gottlieb, S., Shu, C.W., Tadmor, E.: Strong Stability-Preserving High-Order Time Discretization Methods. SIAM Review **43**(1), 89–112 (2001). DOI 10.1137/S003614450036757X
27. Hallett, A.H., Ma, Y., Yin, Y.P.: Hybrid algorithms with automatic switching for solving nonlinear equation systems. J. Econ. Dynam. Control **20**(6), 1051–1071 (1996). DOI 10.1016/0165-1889(95)00889-6
28. Hogea, C.S., Murray, B.T., Sethian, J.A.: Simulating complex tumor dynamics from avascular to vascular growth using a general level-set method. J. Math. Biol. **53**(1), 86–134 (2006). DOI 10.1007/s00285-006-0378-2
29. Jiang, G.S., Peng, D.: Weighted ENO Schemes for Hamilton-Jacobi Equations. SIAM J. Sci. Comput. **21**(6), 2126–2143 (2000). DOI 10.1137/S106482759732455X
30. Jiang, G.S., Shu, C.W.: Efficient Implementation of Weighted ENO Schemes. J. Comput. Phys. **126**(2), 202–228 (1996). DOI 10.1006/jcph.1996.0130
31. Kloner, R.A., Jennings, R.B.: Consequences of brief ischemia: stunning, preconditioning, and their clinical implications: part 1. Circulation **104**(24), 2981–2989 (2001)
32. LeVeque, R.J., Li, Z.: The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources. SIAM J. Numer. Anal. **31**(4), 1019–1044 (1994). DOI 10.1137/0731054
33. Lin, P.T., Baker, T.J., Martinelli, L., Jameson, A.: Two-dimensional implicit time-dependent calculations on adaptive unstructured meshes with time evolving boundaries. Int. J. Numer. Meth. Fluids **50**(2), 199–218 (2006). DOI 10.1002/fld.1050
34. Liu, X.D., Fedkiw, R., Kang, M.: A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains. J. Comput. Phys. **160**(1), 151–178 (2000). DOI 10.1006/jcph.2000.6444
35. Lowengrub, J.S., Macklin, P.: A Centimeter-Scale Nonlinear Model of Tumor Growth in Complex, Heterogeneous Tissues. J. Math. Biol. (2007). (in preparation)

36. Macklin, P.: Numerical Simulation of Tumor Growth and Chemotherapy. M.S. thesis, University of Minnesota School of Mathematics (2003)
37. Macklin, P.: Toward Computational Oncology: Nonlinear Simulation of Centimeter-Scale Tumor Growth in Complex, Heterogeneous Tissues. Ph.D. dissertation, University of California, Irvine Department of Mathematics (2007)
38. Macklin, P., Lowengrub, J.S.: Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth. J. Comput. Phys. **203**(1), 191–220 (2005). DOI 10.1016/j.jcp.2004.08.010
39. Macklin, P., Lowengrub, J.S.: An improved geometry-aware curvature discretization for level set methods: application to tumor growth. J. Comput. Phys. **215**(2), 392–401 (2006). DOI 10.1016/j.jcp.2005.11.016
40. Macklin, P., Lowengrub, J.S.: Nonlinear simulation of the effect of microenvironment on tumor growth. J. Theor. Biol. **245**(4), 677–704 (2007). DOI 10.1016/j.jtbi.2006.12.004
41. Malladi, R., Sethian, J.A., Vemuri, B.C.: A fast level set based algorithm for topology-independent shape modeling. J. Math. Imaging Vision **6**(2-3), 269–289 (1996). DOI 10.1007/BF00119843
42. McMullen, M.S., Jameson, A.: The computational efficiency of non-linear frequency domain methods. J. Comput. Phys. **212**(2), 637–661 (2006). DOI 10.1016/j.jcp.2005.07.021
43. Osher, S., Fedkiw, R.: Level Set Methods: An Overview and Some Recent Results. J. Comput. Phys. **169**(2), 463–502 (2001). DOI 10.1006/jcph.2000.6636
44. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer, New York, NY (2002)
45. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J. Comput. Phys. **79**(1), 12–49 (1988). DOI 10.1016/0021-9991(88)90002-2
46. Peng, D., Merriman, B., Osher, S., Zhao, H., Kang, M.: A PDE-Based fast local level set method. J. Comput. Phys. **155**(2), 410–438 (1999). DOI 10.1006/jcph.1999.6345
47. Quaranta, V., Weaver, A.M., Cummings, P.T., Anderson, A.R.A.: Mathematical Modeling of Cancer: The future of prognosis and treatment. Clinica Chimica Acta **357**(2), 173–9 (2005). DOI 10.1016/j.cccn.2005.03.023
48. Sanga, S., Sinek, J.P., Frieboes, H.B., Fruehauf, J.P., Cristini, V.: Mathematical modeling of cancer progression and response to chemotherapy. Expert. Rev. Anticancer Ther. **6**(10), 1361–76 (2006). DOI 10.1586/14737140.6.10.1361
49. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Cambridge University Press, New York, NY (1999)
50. Sethian, J.A., Smereka, P.: Level set methods for fluid interfaces. Ann. Rev. of Fluid Mech. **35**(1), 341–372 (2003). DOI 10.1146/annurev.fluid.35.101101.161105
51. Sleijpen, G.L.G., van der Vorst, H.A., Fokkema, D.R.: bicgstab(ℓ) and other hybrid Bi-CG methods. Numer. Algorithms **1**(7), 75109 (1994). DOI 10.1007/BF02141261
52. Sussman, M., Fatemi, E.: An Efficient, Interface Preserving Level Set Re-Distancing Algorithm and its Application to Interfacial Incompressible Fluid Flow. SIAM J. Sci. Comput. **20**(4), 1165–1191 (1999). DOI 10.1137/S1064827596298245
53. Vrahatis, M.N., Magoulas, G.D., Plagianakos, V.P.: From linear to nonlinear iterative methods. Appl. Num. Math. **45**(1), 59–77 (2003). DOI 10.1016/S0168-9274(02)00235-0
54. Wise, S.M., Lowengrub, J.S., Frieboes, H.B., Cristini, V.: Three-dimensional Diffuse-Interface Simulation of Multispecies Tumor Growth-I: Numerical Method. Bull. Math. Biol. (2006). (in review)
55. Yu, S., Zhou, Y., Wei, G.W.: Matched interface and boundary (MIB) method for elliptic problems with sharp-edged interfaces. J. Comput. Phys. (2007). DOI 10.1016/j.jcp.2006.10.030. In press
56. Zhao, H.K., Chan, T., Merriman, B., Osher, S.: A variational level set approach to multiphase motion. J. Comput. Phys. **127**(1), 179–195 (1996). DOI 10.1006/jcph.1996.0167

57. Zheng, X., Wise, S.M., Cristini, V.: Nonlinear simulation of tumor necrosis, neovascularization and tissue invasion via an adaptive finite-element/level set method. Bull. Math. Biol. **67**(2), 211–259 (2005). DOI 10.1016/j.bulm.2004.08.001
58. Zhou, Y.C., Wei, G.W.: On the ficticious-domain and interpolation formulations of the matched interface and boundary (MIB) method. J. Comput. Phys. **219**(1), 228–246 (2006). DOI 10.1016/j.jcp.2006.03.027
59. Zhou, Y.C., Zhao, S., Feig, M., Wei, G.W.: High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. J. Comput. Phys. **213**(1), 1–30 (2006). DOI 10.1016/j.jcp.2005.07.022