Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 203 (2005) 191–220
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth

Paul Macklin, John Lowengrub *

Department of Mathematics, 103 MSTB, University of California, Irvine, CA 92697, USA

Received 28 January 2004; received in revised form 2 August 2004; accepted 16 August 2004
Available online 29 September 2004

Abstract

We develop an algorithm for the evolution of interfaces whose normal velocity is given by the normal derivative of a solution to an interior Poisson equation with curvature-dependent boundary conditions. We improve upon existing techniques and develop new finite difference, ghost fluid/level set methods to attain full second-order accuracy for the first time in the context of a fully coupled, nonlinear moving boundary problem with geometric boundary conditions (curvature). The algorithm is capable of describing complex morphologies, including pinchoff and merger of interfaces. Our new methods include a robust, high-order boundary condition-capturing Poisson solver tailored to the interior problem, improved discretizations of the normal vector and curvature, a new technique for extending variables beyond the zero level set, a new orthogonal velocity extension technique that is both faster and more accurate than traditional PDE-based approaches, and a new application of Gaussian filter technology ordinarily associated with image processing. While our discussion focuses on two-dimensional problems, the techniques presented can be readily extended to three dimensions. We apply our techniques to a model for tumor growth and present several 2D simulations. Our algorithm is validated by comparison to an exact solution, by resolution studies, and by comparison to the results of a spectrally accurate method boundary integral method (BIM). We go beyond morphologies that can be described by the BIM and present accurate simulations of complex, evolving tumor morphologies that demonstrate the repeated encapsulation of healthy tissue in the primary tumor domain – an effect seen in the growth of real tumors.

Keywords: Moving boundary problems; Level set method; Tumor growth; Interior Poisson problem; Second-order accuracy; Finite differences; Ghost fluid method; Curvature discretization; Velocity extension; Gaussian filter

* Corresponding author. Tel.: +1 9498242655; fax: +1 9498247993.
E-mail addresses: pmacklin@math.uci.edu (P. Macklin), lowengrb@math.uci.edu (J. Lowengrub).
URLs: www.math.uci.edu/~pmacklin, www.math.uci.edu/~lowengrb.

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2004.08.010

1. Introduction

The algorithms developed herein are motivated by our interest in modeling tumor growth and the morphological response of tumors to environmental stimuli and tissue inhomogeneity. Tumor growth is a fundamental scientific and societal problem. While much work has been done in the mathematics community on tumor modeling (e.g., see the recent review [32]), the state-of-the-art in modeling and numerical simulation lags behind the current understanding of the biophysical processes. The work presented in this paper is a step towards closing this gap and can be viewed as a building block towards a sophisticated virtual cancer simulator. In addition, the methods described in this paper have application beyond the tumor growth context and can be applied to general systems of coupled interior Poisson problems on a moving domain with geometric boundary conditions.

The tumor model we consider here was previously investigated by Cristini et al. [7]. This model is a reformulation of several classical models by Adam and co-workers [2–5,7,27]. A continuum-level description of tumor growth is used, and a sharp interface separates the tumor and healthy tissue. The tumor tissue is modeled as an incompressible fluid, and tissue elasticity is neglected. Cell-to-cell adhesive forces are modeled by a surface tension at the tumor-healthy tissue interface. The cell velocity is determined by Darcy's law, and growth occurs due to pressure gradients induced by mitosis (cell proliferation). A single nutrient (e.g., oxygen or glucose) is required for cell viability and mitosis. The nutrient diffuses through the tissue and is consumed by the tumor cells. This can limit the overall growth through the formation of a necrotic core (region of dead cells). Tumor cells die when the nutrient level drops below a critical level necessary for cell viability. This model is appropriate for characterizing solid tumors of sufficient size growing into soft tissue such as the brain. We note that discrete models such as cellular automata have been used to simulate tumor growth and are particularly applicable when the tumor boundary is fractal-like or diffuse [20].

Currently, we do not model a number of important biophysical processes, including angiogenesis (the formation of new blood vessels), genetic mutations, different cell species, and more realistic tissue responses (e.g., viscoelastic). These effects can be included in our framework. In fact, there is very recent work by Zheng et al. [38] that uses an adaptive level set method to simulate tumor necrosis, angiogenesis, and tissue invasion. Genetic effects can be incorporated by including different cell species and by varying the biophysical parameters via a stochastic model. The different cell species can be included in this framework by introducing an interface for each species, which is straightforward in the level set approach taken in this paper.

In [7], Cristini et al. presented the first nonlinear simulations of this continuum model of tumor growth using a spectrally accurate boundary integral method. However, the boundary integral method does not allow for inhomogeneous microphysical parameters and is not well-suited to the complex morphological changes inherent in tumor evolution, including the pinchoff and coalescence of tumor tissue and the development and evolution of a necrotic core. The tumor model is a special case of a classical system of coupled interior Poisson and Poisson-like problems on a moving domain with geometric boundary conditions. The velocity of the domain boundary is determined from the normal derivatives of the solutions to the Poisson equations. Therefore, we sought to use a robust, second-order accurate finite difference ghost fluid/level set method. The methods described in this paper are formulated for the classical system and thus can be applied beyond the tumor growth context.

Ghost fluid and level set methods have been applied with great success in a wide variety of physical applications (e.g. see the texts [34,29]). We first applied standard level set [34,28,29], WENO [19,18], total variation diminishing Runge–Kutta [17,16], and ghost fluid methods [14,9,21,12,6], as well as a standard PDE-based velocity extension [37] to the tumor growth model. However, because the full moving boundary problem is sensitive to variations in the curvature, the speed can become noisy when even small perturbations in the level set function are present. For these equations, the dependence of the normal velocity upon the derivative of the curvature requires a severe third-order CFL time step restriction [7]. Furthermore, we

still obtained merely first-order to 1.6-order convergence, and the standard discretizations for the normal vector and curvature were highly inaccurate near merging interfaces.

To obtain full second-order accuracy in space and time, we develop a new Poisson solver capable of capturing geometric boundary conditions on a complicated interface. We develop geometry-aware discretizations of the normal vectors and curvature that automatically detect and cope with level set irregularity, particularly during morphological changes. We also develop new gradient and velocity extension techniques that take full advantage of the geometric information embedded in the level set function to obtain greater accuracy and faster computational speed than techniques currently in use. As a way to remove the high-order time step constraint, Gaussian filtering is applied to remove small, high frequency perturbations before they pollute the numerical solution. This is computationally inexpensive, does not degrade the accuracy of the numerical solution, and allows a first-order time step restriction.

Our Poisson solver, an extension of the ghost fluid method found in [21,12,13], retains all the qualities developed therein. In particular, we avoid the complication of solving on an irregular grid by solving on a simpler rectangular grid. The solution satisfies the boundary condition at the precise location of the interface, rather than at nearby nodes. The method is robust and allows a straightforward, dimension-by-dimension implementation, although a small consideration needs to be made for the interaction of spatial dimensions in one case. We present numerical evidence that strongly suggests our algorithm yields second-order accuracy, even when applied to the full moving boundary problem with geometric boundary conditions (e.g., curvature). We note that Gibou and Fedkiw recently proposed an extension of the ghost fluid method in [11] that attains fourth-order accuracy on a fixed domain and third-order accuracy for the Stefan problem (on a moving boundary) without curvature-dependent boundary conditions (i.e., zero surface tension).

The outline of this paper is as follows. In Section 2, we formulate the classical system of interior Poisson-like problems on a moving boundary; our tumor growth model is a special case. Section 3 provides an outline of our general method. In Section 4, we describe our new interior Poisson solver, our gradient discretization and new gradient extension, our new velocity extension, our modifed normal vector and curvature discretizations, and our new application of Gaussian filter technology. In Section 5, we verify the second-order convergence with geometric boundary conditions, compare our results to spectrally accurate results in [7], and investigate the effects of the velocity filtering, the new velocity extension technique, and our modifications to the normal vector and curvature algorithms. Lastly, in Section 6, we give numerical evidence for second-order convergence in the presence of necrosis and present several simulations of tumor growth that showcase the robustness of the algorithm through complex morphological changes. This work is a continuation of the techniques developed in Paul Macklin's M.S. thesis [22].

2. The equations for the interior problem

2.1. Interior equations

We wish to solve a system of Poisson-like problems in a moving domain $\Omega(t)$ whose boundary $\Sigma(t)$ 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 $\Omega \cup \Sigma$ that satisfy

$$\begin{cases} \nabla^2 p_i = f_i(p_1, p_2, \ldots, p_{i-1}, p_i, \mathbf{x}, t) & \text{in } \Omega \\ p_i = g_i(\kappa, \mathbf{x}, t) & \text{on } \Sigma \end{cases} \quad 1 \leqslant i \leqslant k \tag{1}$$

and determine the outward normal velocity of the interface by

$$V|_{\Sigma} = \sum_{i=1}^{k} \alpha_i (\nabla p_i \cdot \mathbf{n})|_{\Sigma}, \tag{2}$$

where $\mathbf{n}$ is the unit normal vector on Σ oriented outward from Ω, and each ∇p_i is a one-sided "interior" gradient at Σ based on values on Σ and in Ω. In our formulation, each p_i depends upon $p_{i-1}, p_{i-2}, \ldots$, allowing for a partial decoupling of the system, but this restriction could be removed.

As in [34,28–30], we capture the boundary Σ implicitly by introducing a level set function φ defined on a rectangular domain $\mathscr{D} \supset (\Omega \cup \Sigma)$ such that

$$\varphi(\mathbf{x}) \begin{cases} < 0 & \text{if } \mathbf{x} \in \Omega, \\ = 0 & \text{if } \mathbf{x} \in \Sigma, \\ > 0 & \text{else.} \end{cases} \tag{3}$$

In this framework, we call Ω the interior region, $\Omega_o = \mathscr{D} \setminus (\Sigma \cup \Omega)$ the exterior region, and Σ the interface between the regions. (Let us denote by $A \setminus B$ the set subtraction B from A.) To update the interface position Σ in time, we solve the additional Hamilton–Jacobi equation

$$\varphi_t + \widetilde{V} |\nabla \varphi| = 0 \tag{4}$$

throughout $\mathscr{D}$ [34,29,30]. Here, $\widetilde{V}$ is an extension of V beyond the interface. We further stipulate that φ is a signed distance function: $|\varphi(\mathbf{x})| = d(\mathbf{x},\Sigma)$. We ensure this property by reinitializing φ at every time step [34,29,35]. From the level set function, we can readily compute geometric quantities:

$$\mathbf{n} = \frac{\nabla \varphi}{|\nabla \varphi|} \tag{5}$$

and

$$\kappa = \nabla \cdot \left(\frac{\nabla \varphi}{|\nabla \varphi|} \right). \tag{6}$$

2.2. Application: tumor growth

We will apply the techniques developed in this paper to a model of tumor growth, which is a reformulation of several classical models found in [2–5,7,27]. Let Ω denote a two-dimensional tumor mass, let Σ be its boundary, let Ω_N denote the necrotic core of Ω (note that $\Omega_N \subset \Omega$), and let us denote the boundary of Ω_N by Σ_N. As stated earlier, we enclose $\overline{\Omega} = \Omega \cup \Sigma$ in a larger rectangular domain $\mathscr{D}$, and we define $\Omega_o = \mathscr{D} \setminus \overline{\Omega}$.

Let c and p denote a nondimensionalized concentration and pressure, respectively (see [3,7,22] for the model and nondimensionalization). The dimensionless parameters include G which is related to the rate of mitosis (cell proliferation), and G_N measures the rate of volume loss due to necrosis (cell degradation) relative to the rate of mitosis. In addition, the parameter A measures the rate of apoptosis ("pre-programmed" cell death), and N is the value of c necessary for cell viability. Note that the necrotic core Ω_N is the region where $c < N$.

From [7], the concentration satisfies

$$\begin{cases} \nabla^2 c = c & \text{in } \Omega, \\ c|_{\Sigma} = 1, & \\ c = 1 & \text{outside } \Sigma \end{cases} \tag{7}$$

and the pressure satisfies

$$\begin{cases} \nabla^2 p = \begin{cases} -G(c-A) & \text{in } \Omega \text{ if } c \geqslant N, \\ GG_{\mathrm{N}} & \text{in } \Omega \text{ if } c < N, \end{cases} \\ p|_{\Sigma} = \kappa \\ p = 0 \quad \text{outside } \Sigma. \end{cases} \tag{8}$$

Note that the concentration is determined solely by the position of the interface Σ and can be solved independently of the pressure. This allows the necrotic core to be determined prior to the pressure solve by the region where $c - N$ is negative.

The outward normal velocity is given by Darcy's law

$$V|_{\Sigma} = -\mathbf{n} \cdot \nabla p, \tag{9}$$

where ∇p is the interior pressure gradient in the region $\overline{\Omega}$.

3. Numerical solution: general technique

We begin by enclosing the interface within a larger, rectangular computational domain $\mathscr{D} = [a,b] \times [c,d]$. (We will postpone our discussion of how large $[a,b]\times[c,d]$ is for a later part of this paper.) We then proceed via:

(1) Initialize a level set function φ to represent the interface Σ while ensuring that there are sufficiently many computational node points between Σ and the computational boundary.
(2) Check for proximity of the interface Σ to the computational boundary. If there is insufficient space between Σ and $\partial\mathscr{D}$, then extend x, y, and φ. Reinitialize φ.
(3) Calculate the normal vector $\mathbf{n}$ and the curvature κ where required.
(4) Solve the Poisson problems for $p_1,\ldots,p_k$.
(5) Calculate the gradients ∇p_i inside and on Σ, and extend the components of the gradients beyond Σ into Ω_{o}.
(6) Calculate the normal velocity V in a band about Σ according to (2). Extend the normal velocity orthogonally from the interface Σ, and filter high-frequency numerical noise from the extended speed.
(7) Update φ according to (4).
(8) Repeat (2)–(7) for each step of the time discretization.

4. Discretizations

4.1. Interior Poisson solver

Our solution technique for the Poisson problem was first developed in [22] and is an extension of the ghost fluid methods in [14,9,21] to higher-order accuracy. In this method, we solve a general interior problem for u (which can be either the nutrient concentration c or the pressure p) in a complex domain

$$\begin{cases} \nabla^2 u = f(u,\mathbf{x}) & \text{in } \Omega, \\ u = g(\kappa,\mathbf{x}) & \text{on } \Sigma, \end{cases} \tag{10}$$

by embedding the problem in the rectangular domain $\mathscr{D}$ and extending u as a constant γ into Ω_{o}. Thus, we solve the system

$$\begin{cases} \nabla^2 u = f(u,\mathbf{x}) & \text{in } \Omega, \\ u = g(\kappa,\mathbf{x}) & \text{on } \Sigma, \\ u = \gamma & \text{in } \Omega_o. \end{cases} \tag{11}$$

The solution u can be assumed to be smooth within Ω up to Σ. We assume that Σ is defined by means of a level set function φ as described before. Also, we assume that g is a function that can be evaluated at all node points near Σ.

Note that the equations governing the tumor nutrient concentration (7) and the pressure (11) can be written in the above form, where

$$u = c, \quad f(c,\mathbf{x}) = c, \quad g(\kappa,\mathbf{x}) = 1, \quad \gamma = 1 \tag{12}$$

and

$$u = p, \quad f(p,\mathbf{x}) = \begin{cases} -G(c-A) & \text{in } \Omega_P, \\ GG_N & \text{in } \Omega_N, \end{cases} \quad g(\kappa,\mathbf{x}) = \kappa, \ \ \gamma = 0, \tag{13}$$

respectively. Note that in the presence of necrosis, the pressure has a discontinuous second derivative at the boundary of the necrotic core.

The central idea of the discretization technique [9,21] begins with the standard centered difference for u_{xx}: if $[x_{i-1}, x_{i+1}]$ lies entirely within Ω, then

$$u_{xx} = \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} + \mathcal{O}(\Delta x^2). \tag{14}$$

However, if the interface intersects $[x_{i-1}, x_{i+1}]$, then u is potentially discontinuous, and the finite difference approximation in (14) is inaccurate. Supposing that the interface occurs between x_i and x_{i+1}, u_{i+1} is replaced in (14) with $\hat{u}_{i+1}$, a smooth extension of u from the inner domain to x_{i+1} (see Fig. 1).

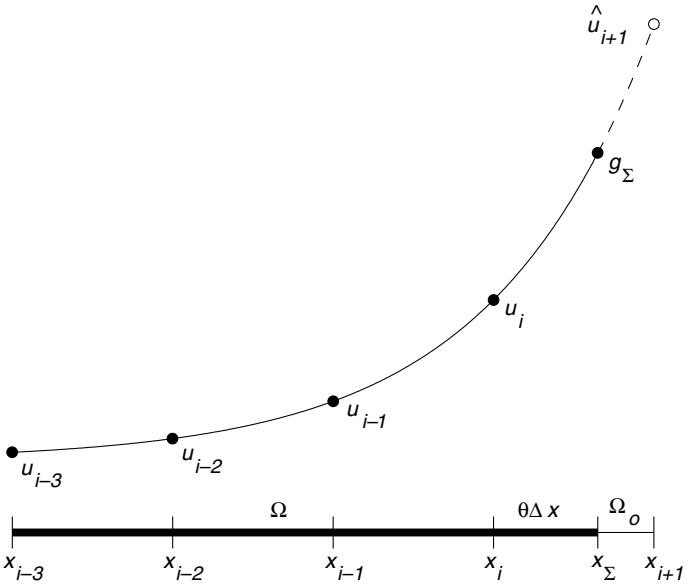

Fig. 1. Ghost fluid method: extrapolation to $\hat{u}_{i+1}$.

As stated above, the pressure has a discontinuous second derivative across the boundary of the necrotic core. If left untreated (as is done here) this limits the overall accuracy of the scheme to second order. A higher-order accurate treatment can be achieved by applying a ghost fluid discretization at the boundary of the necrotic region as well as at the interface Σ.

In our approach, we make three principal approximations. We estimate the location of the interface between x_i and x_{i+1} by linear interpolation of φ; this is known as *subcell resolution* [21,12,13]. We approximate the value of the boundary condition at the interface by cubic interpolation of g near Σ. Lastly, we extrapolate $\hat{u}_{i+1}$ from the boundary condition and multiple points within Ω using linear, quadratic, or cubic extrapolation.

4.1.1. Classification of node points

For points contained in Ω, the solver proceeds by constructing an approximation to $\nabla^2 u$ at each point (x_i,y_j) while considering which points of the 5-point stencil

$$\{(x_i, y_j), (x_i, y_{j\pm1}), (x_{i\pm1}, y_j)\}$$

are contained in Ω, in Ω_o, and on Σ. The level set formulation of the interface makes this classification a straightforward matter:

$$x_{i,j} \in \begin{cases} \Omega & \text{if } \varphi_{i,j} < -\epsilon, \\ \Sigma & \text{if } |\varphi_{i,j}| \leqslant \epsilon, \\ \Omega_o & \text{if } \varphi_{i,j} > \epsilon, \end{cases} \tag{15}$$

where ϵ is introduced to account for finite machine precision. We take $\epsilon = 2\epsilon_{\text{mach}}$, where

$$\epsilon_{\text{mach}} = \max\{\epsilon > 0 : 1.0 + \epsilon = 1.0\} \tag{16}$$

in machine floating-point arithmetic. Note that because computer hardware can only represent finitely many floating-point numbers, this set has a unique, nonzero maximum. On most modern, 32-bit machines, this number is typically $2^{-53} \approx 1.11\text{ e}{-16}$.

4.1.2. Discretizing the equation

We discretize (11) on the full rectangular domain although we are solving the interior problem. The rows corresponding to the trivially solvable discretizations are included in the coefficient matrix because this preserves the row (or column) ordering of the coefficient matrix, yielding a banded matrix that can be stored efficiently in memory. We proceed by discretizing (11) at each node x_i according to the classification of x_i and its neighbors by (15).

The discretization on Σ and in Ω_o is trivial:

(1) *Case:* $x_i \in \Omega_o$

By (11), $u_i = \gamma$. To improve the conditioning number of the coefficient matrix, we shall use

$$\frac{-1}{\Delta x^2} u_i = \frac{-1}{\Delta x^2} \gamma. \tag{17}$$

(2) *Case:* $x_i \in \Sigma$

In this case, $u_i = g(\kappa_i, x_i)$. Again, we set

$$\frac{-1}{\Delta x^2} u_i = \frac{-1}{\Delta x^2} g(\kappa_i, x_i). \tag{18}$$

to improve the conditioning number of the coefficient matrix.

When considering points in Ω, we must approximate $\nabla^2 u$. Let us first consider the discretization of u_{xx}. We shall then approximate $\nabla^2 u$ dimension-by-dimension, as the discretization of u_{yy} is identical except in

one case where the two-dimensionality is important. We proceed by classifying the node points $\{(x_{i\pm1}),x_i\}$. Consider the following cases:

(3) *Case:* $x_i \in \Omega$

(a) *Case:* $x_{i-1} \in \Omega$ and $x_{i+1} \in \Omega$

In this case, the entire stencil is contained in the inner region, so we can use the standard second-order approximation to $u_{xx} = f$:

$$\frac{1}{\Delta x^2}(u_{i-1} - 2u_i + u_{i+1}) = f(u_i, x_i). \tag{19}$$

(b) *Case:* $x_{i-1} \in \Omega$ and $x_{i+1} \in (\Sigma \cup \Omega_o)$

In this case, the interface is located between x_i and x_{i+1} on the right-hand side of the stencil. Let us denote this location by x_Σ. We denote

$$x_\Sigma = x_i + \theta\Delta x, \quad 0 < \theta \leqslant 1, \tag{20}$$

where θ is determined by interpolating the level set function φ. This provides us with the subcell resolution introduced earlier. Notice that if $\theta \to 0$, then $x_i \in \Sigma$, and we are in case 2.

Next, let us define $\mathscr{S} = \{x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$. We evaluate g at the points in $\mathscr{S}$ and the corresponding $\{\kappa_{i-1},\kappa_i,\kappa_{i+1},\kappa_{i+2}\}$, apply cubic interpolation, and evaluate the interpolation at x_Σ. Let us denote the value of the interpolation by g_Σ.

We extend u from the interior region to x_{i+1} and obtain a "ghost value" $\hat{u}_{i+1}$. We determine $\hat{u}_{i+1}$ by extrapolating from the neighboring values of u contained in Ω, solving algebraically for $\hat{u}_{i+1}$, and substituting the expression for $\hat{u}_{i+1}$ in

$$\frac{1}{\Delta x^2}(u_{i-1} - 2u_i + \hat{u}_{i+1}) = f(u_i, x_i). \tag{21}$$

For completeness, we give linear, quadratic, and cubic extrapolations in Appendix A. Similar extrapolations are also given in [11]. We note that because all the points in our extrapolations are at least Δx apart, the case, where $\theta \to 0$, if it should occur, poses no difficulty for our discretization.

(c) *Case:* $x_{i-1} \in (\Sigma \cup \Omega_o)$ and $x_{i+1} \in \Omega$

In this case, the interface is located between x_{i-1} and x_i. The discretization is completely analogous to that in the previous case.

(d) *Case:* $x_{i-1} \in (\Sigma \cup \Omega_o)$ and $x_{i+1} \in (\Sigma \cup \Omega_o)$

In this case, the interface intersects the stencil not once but twice; this requires more careful consideration as a series of subcases:

(i) *Subcase:* $x_{i-1} \in \Sigma$ and $x_{i+1} \in \Sigma$

We can construct an approximation to u_{xx} by

$$\frac{1}{\Delta x^2}(g(\kappa_{i-1}, x_{i-1}) - 2u_i + g(\kappa_{i+1}, x_{i+1})). \tag{22}$$

(ii) *Subcase:* $x_{i-1} \in \Sigma$ and $x_{i+1} \in \Omega_o$

In this subcase, we can proceed as in Case 3b with two minor modifications: we replace u_{i-1} by $g(\kappa_{i-1},x_{i-1})$ in the extrapolation for $\hat{u}_{i+1}$, and the extrapolation must be linear. (We do not allow for extrapolations using both u_i and g_Σ because such extrapolations become unstable as $\theta \to 0$.)

(iii) *Subcase:* $x_{i-1} \in \Omega_o$ and $x_{i+1} \in \Sigma$

This case is completely analogous to the previous subcase.

(iv) *Subcase:* $x_{i-1} \in \Omega_o$ and $x_{i+1} \in \Omega_o$

The interface occurs on both the right- and left-hand sides of the stencil, and there is insufficient data to extrapolate both $\hat{u}_{i-1}$ and $\hat{u}_{i+1}$. (We avoid extrapolations using both u_i and $u(x_i + \theta\Delta x)$, as these become unstable as $\theta \to 0$. Similarly, we avoid extrapolations using u_i and $u(x_{i-1} + \theta\,\Delta x)$, as these become unstable as $\theta \to 1$.) In this case, we take $u_{xx} = 0$ and consider the y-direction. If the same occurs so that we take $u_{yy} = 0$, we say that the discretization fails to resolve Σ around (x_i,y_j) and take the point to fall in Ω_o. Hence, we set

$$\frac{-1}{\Delta x^2}u_i = \frac{-1}{\Delta x^2}\gamma. \tag{23}$$

Notice that we cannot make such a distinction without considering the two-dimensionality of the problem (see Fig. 2). We note that in [13,11] constant extrapolations are also allowed and so in those works it was unnecessary to consider the two-dimensionality.

When we use this technique with cubic extrapolation, which shall denote it by Poisson3. Likewise, quadratic and linear extrapolation for $\hat{u}$ are Poisson2 and Poisson1, respectively. When using Poisson3, if there are not sufficiently many interior node points for cubic extrapolation to $\hat{u}$, we use Poisson2 or Poisson1 in that instance. The same applies to Poisson2. In our work, we solved the resulting linear systems with the stabilized biconjugate gradient method (BiCG-Stab(2)) [8] with a compact banded matrix storage scheme [31].

4.2. Gradients

The normal velocity in (2) requires ∇p_i on Σ for each i. For our method, we must also calculate the gradients in Ω. Let u be a function whose gradient we wish to calculate, and consider u_x. For interior points $x_i \in \overline{\Omega}$ where $\varphi \leqslant \epsilon$, we use the five-point stencil

$$u_x(x_i) = \frac{1}{12\Delta x}(u_{i-2} - 8u_{i-1} + 8u_{i+1} - u_{i+2}) + \mathcal{O}(\Delta x^4), \tag{24}$$

when $\{x_i, x_{i\pm1}, x_{i\pm2}\} \subset \overline{\Omega} = \Omega \cup \Sigma$; when only $\{x_i, x_{i\pm1}\} \subset \overline{\Omega}$, we use the standard second-order centered difference. If one of $x_{i\pm1} \in \Omega_o$, we construct a polynomial interpolation of u in $\overline{\Omega}$ using two-to-four nearby points in Ω, differentiate the interpolation, and evaluate at x_i. In this way, we can calculate u_x to second order or better accuracy at all points in Ω and on Σ. We obtain the partial derivative u_y similarly.

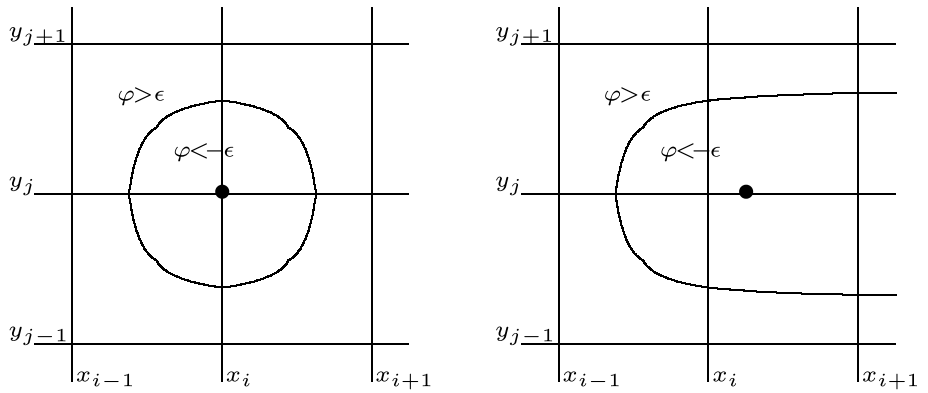

Fig. 2. Impact of two-dimensionality on the Poisson solver. In the left figure, the interface Σ is unresolved near (x_i,y_j). In the right figure, $u_{yy} = 0$ but the interface is still resolved.

4.3. Extensions

Since we solve the advection Eq. (4), we require an extended normal velocity in a band surrounding Σ. Because the Poisson solutions p_i only have meaningful gradients (in the context of the interior problem) on $\overline{\Omega}$, the first step of our extension procedure is to extend the individual components of the gradients ∇p_i beyond Σ into Ω_o. Once this is done, we can evaluate (2) at any point near Σ.

To help maintain the accuracy of the level set φ, we then use a velocity extension technique that satisfies the orthogonality criterion $\nabla V \cdot \mathbf{n} = 0$, which helps preserve the spacing of the level set contours. We note that our orthogonal extension is a new technique based upon bilinear interpolation.

Once we have orthogonally extended the velocity, we apply a Gaussian filter in a narrow band about the interface; this removes high frequency noise from the speed function that would otherwise perturb the interface and destabilize the calculation (see Section 4.4). Because the filter only smooths the speed closest to the interface, we must extend the smoothed velocity one final time. We found this approach works best among the various combinations of extension and filtering available.

4.3.1. Gradient extension

As the gradient algorithm only defines the gradients where $\varphi \leqslant \epsilon$, we must extend to a band of nodes where $\varphi > \epsilon$. For stability, our technique must preserve information flow in an outward direction from the interface. The method we describe can be used to extend any scalar function f defined on Σ and in Ω, and we apply it to the components of the ∇p_i individually.

We extend f to a point $\mathbf{x} \in \Omega_o$ by one-dimensional, grid-aligned extrapolation from points where f has either been previously extended or was originally defined (e.g., in Ω). We choose the points used in the extrapolation according to whether the normal vector $\mathbf{n} = (n_1, n_2)$ at $\mathbf{x}$ is mostly horizontal ($|n_1| - |n_2| > \epsilon$ as at point **a** in Fig. 3), mostly vertical ($|n_2| - |n_1| > \epsilon$ as at point **c** in Fig. 3), or mostly diagonal ($\|n_1| - |n_2\| \leqslant \epsilon$ as at point **b** in Fig. 3). This allows the use of high-order extrapolation without the complexity of multidimensional extrapolation; in our work, we used cubic extrapolation (see Fig. 3).

To preserve information flow in the outward direction from the interface, we tag the points requiring extension (larger circles in Fig. 3), and among those points, we choose the point closest to Σ which has not yet been updated (open circles); notice that by the level set formulation, this point can be determined by choosing the remaining point with the smallest positive value of φ.

In our simulations, we applied this technique to each component of the ∇p_i within a band of width $5\Delta x$.

4.3.2. Identifying the closest point on the interface

Ordinarily, it can be an expensive operation to determine the closest point $\mathbf{x}_1$ on the interface Σ to a given point $\mathbf{x}_0$ [1,34,29]. However, we can use the information afforded by the level set function φ to make this a simple, efficient operation; no search is required. At any point $\mathbf{x}_0$, the outward normal vector $\mathbf{n}(\mathbf{x}_0)$ points away from the interface, and $|\varphi(\mathbf{x}_0)|$ gives the distance to the interface. Therefore, the vector

$$\mathbf{W}(\mathbf{x_0}) = -|\varphi(\mathbf{x_0})|\mathbf{n}(\mathbf{x_0}) = -\varphi\mathbf{n} \tag{25}$$

points towards the closest point on Σ and has length equal to the distance from Σ. The point

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{W} \tag{26}$$

explicitly gives the closest point to $\mathbf{x}_0$ on Σ to second order (see Fig. 4).

4.3.3. Orthogonal velocity extension

Once we have a velocity defined in a band about the interface Σ, we apply an extension routine to ensure that $\nabla \widetilde{V} \cdot \mathbf{n} = 0$. We developed an extension based on bilinear interpolation of the velocity near the interface that proved to be more accurate and less computationally expensive than PDE-based techniques (e.g.,

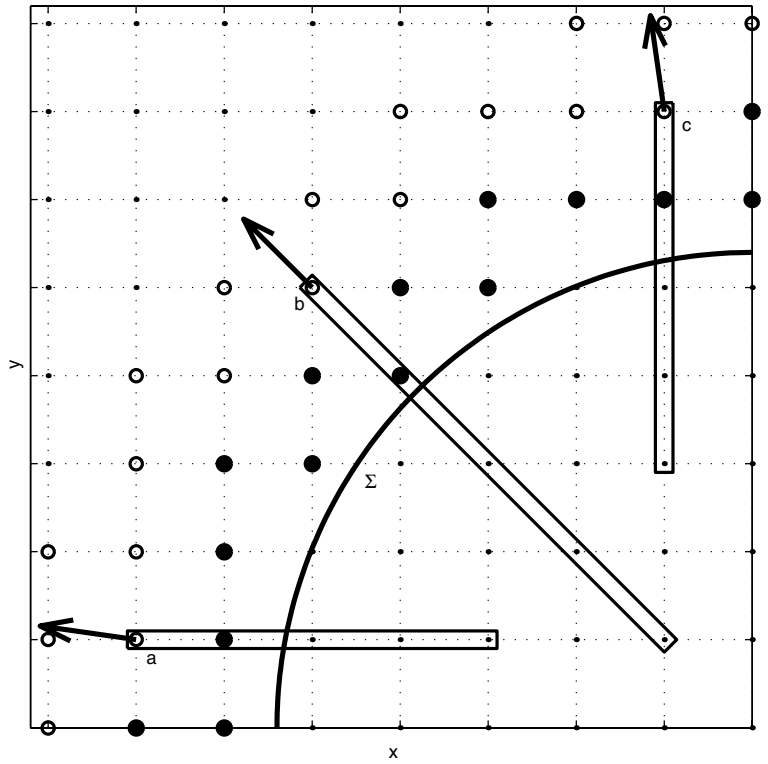

Fig. 3. Gradient extension. We extend a scalar function beyond $\Omega \cup \Sigma$ by one-dimensional, grid-aligned extrapolation. The points used in the extrapolation are chosen according to the direction of the normal vector. We preserve outward information flow by choosing the next point for extension according to the value of the level set function at the remaining points (open circles).

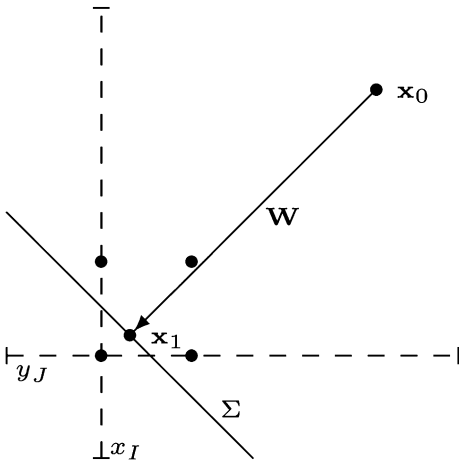

Fig. 4. Finding the closest point on the interface. $\mathbf{W} = -\varphi(\mathbf{x}_0)\,\mathbf{n}(\mathbf{x}_0)$.

that given in [37]). The vector **W** defined in (25) suggests the new extension technique: if we wish to extend V to $\mathbf{x}_0$, we define **W** as in (25) and

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{W} \tag{27}$$

to be the closest point to $\mathbf{x}_0$ on the interface. Next, we locate (x_I, y_J) such that $\mathbf{x}_1$ is contained in the box $[x_I, x_{I+1}] \times [y_J, y_{J+1}]$ and calculate $V(\mathbf{x}_1)$ with bilinear interpolation of V at the corners of the box (see Fig. 4). Lastly, we define $\widetilde{V}(\mathbf{x_0}) = V(\mathbf{x_1})$. Notice that as $\widetilde{V}$ is constant along **W**, which is parallel to $-\mathbf{n}$ at all extended points,

$$\frac{\partial \widetilde{V}}{\partial \mathbf{n}} \equiv 0. \tag{28}$$

This approach differs from the discrete, fast marching velocity extension given in [1] in several ways. First, the fast marching extension technique extends the velocity outward from the interface while simultaneously updating the level set function; ours uses an already-updated level set function to aid in the extension process. We use the level set function to readily locate the closest position on the interface, while the fast marching technique depends on explicitly reconstructing the zero level set as piecewise linear curves and considering multiple cases. Also, while the fast marching method depends upon solving a discretized PDE at every point of extension, ours depends upon a simpler interpolation of previously known values in a way similar to [26].

4.4. Velocity filtering

Because the physical problem is sensitive to variations in the curvature, the speed can become noisy when even small perturbations in the level set function are present. In addition, grid effects such as mesh-induced anisotropies can act as sources of numerical perturbations. In the boundary integral context [7], it was shown that for these equations, the normal velocity depends upon the derivative of the curvature (i.e., $V \sim \mathscr{H}(\kappa_s)$, where $\mathscr{H}$ is the Hilbert transform and s is arclength). It can be shown that such velocity fields damp high frequency perturbations δ_k at the rate $-|k|^3$, where k is the wave number. Thus, perturbations evolve according to $\delta_k \sim \mathrm{e}^{-|k|^3 t}$ at large k. From this consideration, the high frequency perturbations in the speed and interface position should be damped away, provided a CFL restriction is satisfied. The CFL restriction for an explicit boundary integral method is $\Delta t \sim \Delta s^3$. An analysis of our ghost fluid/level set method reveals that this time step restriction also applies, with Δs replaced by Δx. This severe time step restriction can be overcome in a number of ways. For example, in [7], a non-stiff, time integration scheme using a discretization in which the leading order term (term with the largest number of spatial derivatives) is integrated explicitly. This effectively removes the third-order constraint, leaving only a standard first-order CFL time step restriction. The application of such an implicit time integration scheme in the level set context is not straightforward. We are currently working to develop such a scheme.

Another way to remove the high-order time step constraint is to use numerical diffusion to remove small, high frequency perturbations before they pollute the numerical solution. This has the advantage that it is computationally inexpensive, and if done carefully, it does not degrade the accuracy of the numerical solution. We find that adapting a Gaussian filter from image processing applications [15] to smooth the normal velocity within a prescribed band about the interface provides an efficient means of controlling the noise without affecting the accuracy. In addition, the Gaussian filtering removes grid anisotropies as a side benefit.

In one spatial dimension, a Gaussian filter is applied to a function f by

$$\hat{f}_I = \frac{1}{\sigma\sqrt{2\pi}} \sum_i f_{I-i} \exp\left(-\frac{(\mathrm{i}\Delta x)^2}{2\sigma^2}\right)\Delta x, \tag{29}$$

where σ is the standard deviation of the filter. Typically, $\sigma = M\Delta x$ for some integer M. For $|i\Delta x| \geqslant 3\sigma$, the exponential function in the convolution has a very small value (less than approximately 0.0111); consequently, we can truncate the sum above to

$$\hat{f}_I = \frac{1}{S}\frac{1}{M\sqrt{2\pi}}\sum_{i=-3M}^{3M} f_{I-i}\exp\left(-\frac{1}{2}\left(\frac{i}{M}\right)^2\right), \tag{30}$$

where S is the value of the sum for $f \equiv 1$.

To smooth a two-dimensional data array, we use (30) first in the x-direction, and then again in the y-direction. In our calculations, we found that the necessary value of σ depends upon the spatial resolution but decreases with refinement. Because the filter requires that f be defined within a distance of 3σ, we only apply the filter to a narrow band around Σ. In our simulations, we used a narrow band of width $3\Delta x$. As will be shown later, this technique yields accurate results using only a first-order CFL time step restriction.

4.5. Level set reinitialization and advection

As in [35], we reinitialize φ to be a signed distance function by solving

$$\varphi_\tau - \operatorname{sign}(\varphi^0)(1 - |\nabla\varphi|) = 0, \tag{31}$$

where φ^0 is the level set function prior to reinitialization and τ is pseudo-time. We discretize the temporal derivative with the third-order total variation diminishing Runge–Kutta method (TVD-RK), and we approximatesign(φ^0)$|\nabla\varphi|$ with either the third-order or the fifth-order WENO scheme [19,18]. We discretize the sign function according to

$$\operatorname{sign}_\delta(\varphi) = 2\left(H_\delta(\varphi) - \frac{1}{2}\right), \tag{32}$$

where

$$H_\delta(\varphi) = \begin{cases} 0 & \text{if } \varphi < -\delta, \\ \frac{1}{2}\left(1 + \frac{\varphi}{2\delta} + \frac{1}{\pi}\sin\left(\frac{\pi\varphi}{\delta}\right)\right) & \text{if } |\varphi| \leqslant \delta, \\ 1 & \text{if } \varphi > \delta \end{cases} \tag{33}$$

and δ is a small number [36]. In our calculations, we took $\delta = \Delta x$.

In our numerical implementation of the level set Eq. (4), we discretize $V|\nabla\varphi|$ with the third-order or fifth-order WENO method. We approximate the temporal derivative with the third-order total variation diminishing Runge–Kuta (TVD-RK) method [17,16], and we use the CFL condition

$$\Delta t \leqslant \frac{\Delta x}{4\max|V|}. \tag{34}$$

4.6. Normal vectors and curvature

The standard, second-order discretization of the normal vector $\mathbf{n}$ uses centered differences for φ_x and φ_y and normalizes the result. For curvature, the standard second-order method is to calculate each partial derivative in

$$\kappa = \nabla\cdot\frac{\nabla\varphi}{|\nabla\varphi|} = \frac{\varphi_{xx}\varphi_y^2 - 2\varphi_x\varphi_y\varphi_{xy} + \varphi_{yy}\varphi_x^2}{(\varphi_x^2 + \varphi_y^2)^{\frac{3}{2}}} \tag{35}$$

using second order, centered differences. Note that this uses a 9-point stencil, and we discretize φ_{xy} as in [6] by

$$\varphi_{xy}(x_i, y_j) \approx \frac{1}{4\Delta x \Delta y}(\varphi_{i+1,j+1} - \varphi_{i-1,j+1} - \varphi_{i+1,j-1} + \varphi_{i-1,j-1}). \tag{36}$$

However, there are cases where these normal vector and curvature discretizations are inaccurate. If two interfaces approach one another, a "ridge" forms between them where the derivatives of φ are discontinuous. Discretization across this ridge will cause large errors in the normal vector and curvature for an exact (i.e., unperturbed, error-free) level set (see Fig. 5).

Because such ridges tend to introduce error into the surrounding level set function during reinitialization and advection level set operations, the standard discretizations of curvature and the normal vectors are also erratic in the nodes near the ridge. Because our extension techniques require that the normal vectors point away from the interface, and as the boundary conditions of the Poisson problems depend on the curvature, it is critical that we develop a technique to detect these situations and discretize accordingly. In our approach, we first detect these "ridges" and any other irregularities in the level set function, create a field of direction vectors near the ridges to assist in determining one-sided discretizations, and finally discretize the normal vectors and curvature.

4.6.1. Detecting "Ridges" in the level set

Recall that for a signed distance function φ, $|\nabla\varphi| \approx 1$. Thus, a simple technique to detect the points on and near a ridge is to compute $\mathbf{v} = \nabla\varphi$ using centered second-order differences, and then to define a "normal quality function" $Q(\mathbf{v})$ according to

$$Q(\mathbf{v}) = |1 - |\mathbf{v}||. \tag{37}$$

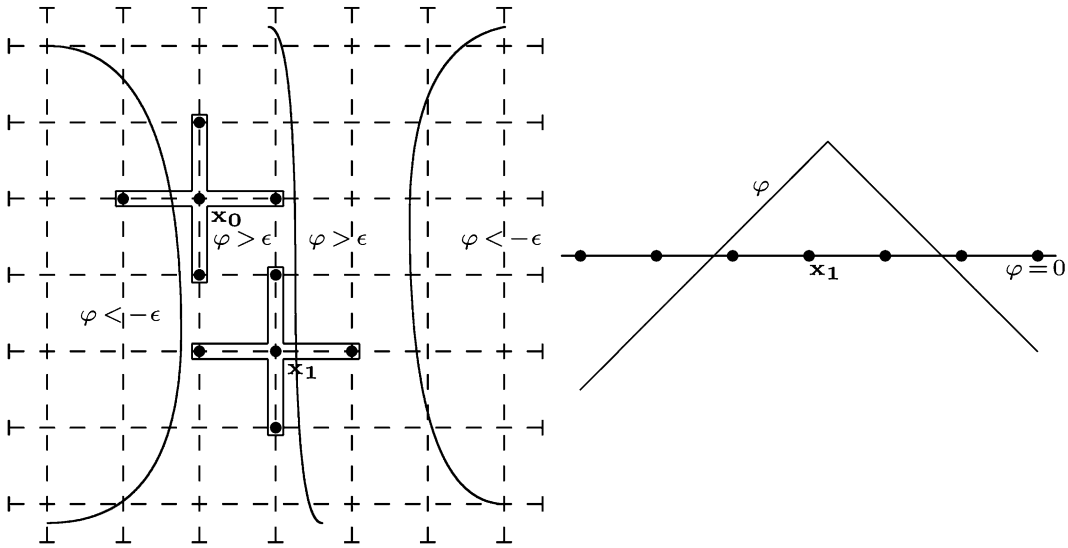

Fig. 5. Effect of level set irregularity on κ and $\mathbf{n}$. In the left figure, two interfaces are close together. The middle curve shows the points equidistant from both interfaces, and the level set function is irregular along this curve. The standard techniques for calculating κ and $\mathbf{n}$ work well at $\mathbf{x}_0$ (where the derivatives of φ are continuous), whereas they break down numerically at $\mathbf{x}_1$. The right figure shows a cross-section through $\mathbf{x}_1$ of the level set function; the "peak" in the middle is equidistant from the two interfaces and a point of irregularity in φ.

We set $Q_{i,j} = Q(\nabla\varphi(x_i,y_j))$. If $Q_{i,j} \geqslant \eta$ for some fixed $0 < \eta < 1$, then the point (x_i,y_j) is on or near a ridge. In our testing, we found that $\eta = .1$ reliably detects the points on and near the ridges with few false positives.

4.6.2. Creating a direction vector field

We next introduce a direction field $\mathbf{D}(x,y)$ to assist in determining whether two neighboring points are on the same side of a ridge. We require that $\mathbf{D}(x_i,y_j)$ points away from a ridge if $Q \geqslant \eta$ at (x_i,y_j) or any one of its eight neighbors in the Cartesian grid, in which case $\mathbf{D}(x_i,y_j)$ points towards one of the eight neighboring points. Otherwise, $\mathbf{D} = 0$. If $\mathbf{D} \neq 0$, we take it to be among the set

$$\mathscr{V} = \{(0,-1),(0,1),(-1,0),(-1,-1),(-1,1),(1,0),(1,-1),(1,1)\}. \tag{38}$$

For a function f at (x_i,y_j), we define the **D**-difference of ∇f component-wise by:

$$\partial_x f = \begin{cases} \frac{f_{i,j}-f_{i-1,j}}{\Delta x} & \text{if } D_x = -1, \\ \frac{f_{i+1,j}-f_{i,j}}{\Delta x} & \text{if } D_x = 1, \\ \frac{f_{i+1,j}-f_{i-1,j}}{2\Delta x} & \text{if } D_x = 0, \end{cases} \tag{39}$$

where $(D_x,D_y) = \mathbf{D}(x_i,y_j)$; $\partial_y f$ is defined similarly.

We determine **D** component-wise according to the value of Q at (x_i,y_j) and its eight neighbors. For the x-direction,

$$D_x = \begin{cases} -1 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i+1,j} \geqslant \eta, \\ 1 & \text{if } Q_{i-1,j} \geqslant \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i,j} < \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} \geqslant \eta \text{ and } Q_{i,j} \geqslant \eta \text{ and } Q_{i+1,j} \geqslant \eta, \\ \text{undetermined} & \text{otherwise.} \end{cases} \tag{40}$$

We determine D_y similarly. We shall denote $\mathbf{D}(x_i,y_j)$ by $\mathbf{D}_{i,j}$. If D_x or D_y is undetermined, then we set $\mathbf{D}'$ equal to the element in $\mathscr{V}$ most parallel to $\nabla\varphi$ (where we again use centered differences). We set $\boldsymbol{D}^1$ and $\mathbf{D}^2$ to be perpendicular to $\mathbf{D}'$, and we define $\mathbf{v}^1$ and $\mathbf{v}^2$ to be the $\mathbf{D}^1$- and $\mathbf{D}^2$-differences of $\nabla\varphi$, respectively. If $Q(\boldsymbol{v}^1) < Q(\mathbf{v}^2) + \mu$, then we choose $\mathbf{D}_{i,j} = \boldsymbol{D}^1$; otherwise, $\mathbf{D}_{i,j} = \mathbf{D}^2$. It is desirable to choose $\mu \neq 0$ to give the direction field **D** a small bias towards one side for points resting exactly on a ridge, as $Q(\mathbf{v}^1) \approx Q(\mathbf{v}^2)$ in such cases. In our testing, we found $\mu = \frac{1}{8}\eta$ works well.

Using these direction vectors, we can readily determine if two adjacent points are on the same side of a ridge or other level set irregularity. Consider, for example, $\mathbf{D}_{i-1,j}$ and $\mathbf{D}_{i,j}$. If the dot product $\mathbf{D}_{i-1,j} \cdot \mathbf{D}_{i,j} > 0$ or $\boldsymbol{D}_{i-1,j} = 0$, then we say that (x_{i-1},y_j) and (x_i,y_j) are on the same side of any and all ridges and level set irregularities.

4.6.3. Discretizing the normal vector

To discretize **n** at (x_i,y_j), we discretize the x-component of **n** as in Table 1. If we use a left-based stencil, we use a stencil based of φ on

$$\Big\{(x_{i-4},y_j),(x_{i-3},y_j),(x_{i-2},y_j),(x_{i-1},y_j),(x_i,y_j)\Big\}. \tag{41}$$

Let K be the number of points adjacent to and left of (x_i,y_j) for which all the points are on the same side of all the ridges; that is, let

$$K = \max\{k : \mathbf{D}_{i-\ell,j} \cdot \mathbf{D}_{i,j} > 0 \text{ or } |\mathbf{D}_{i-\ell,j}| = 0 \text{ for all } 1 \leqslant \ell \leqslant k\}. \tag{42}$$

Table 1
Discretization of the x-component of $\mathbf{n}$ based on the direction vectors

Case	Discretization of x-component of $\mathbf{n}$
$(\mathbf{D}_{i-1,j}\cdot\mathbf{D}_{i,j}>0$ or $\lvert\mathbf{D}_{i-1,j}\rvert=0)$ and $\mathbf{D}_{i+1,j}\cdot\mathbf{D}_{i,j}\leqslant 0$	Use a left-based stencil for φ_x
$\mathbf{D}_{i-1,j}\cdot\mathbf{D}_{i,j}\leqslant 0$ and $(\mathbf{D}_{i+1,j}\cdot\mathbf{D}_{i,j}>0$or $\lvert\mathbf{D}_{i+1,j}\rvert=0)$	Use a right-based stencil for φ_x
$(\mathbf{D}_{i-1,j}\cdot\mathbf{D}_{i,j}>0$ or $\lvert\mathbf{D}_{i-1,j}\rvert=0)$ and $(\mathbf{D}_{i+1,j}\cdot\mathbf{D}_{i,j}>0$ or $\lvert\mathbf{D}_{i+1,j}\rvert=0)$	Use centered difference for φ_x
$\mathbf{D}_{i-1,j}\cdot\mathbf{D}_{i,j}\leqslant 0$ and $\mathbf{D}_{i+1,j}\cdot\mathbf{D}_{i,j}\leqslant 0$	Use centered difference for φ_x

If $K>5$, set $K=5$. Then for the left stencil, we approximate φ_x using the K-point difference of φ at $\{(x_{i-\ell},y_j)\}_{\ell=0}^{K}$. A right-based stencil for φ_x can be defined similarly, and the y-component is discretized analogously. The resulting vector is then normalized.

4.6.4. Discretizing the curvature

Recall that the standard curvature discretization involves the nine points in the square $[x_{i-1},x_{i+1}]\times[y_{j-1},y_{j+1}]$. If each $Q_{k,\ell}<\eta$ for $i-1\leqslant k\leqslant i+1$ and $j-1\leqslant\ell\leqslant j+1$, we use the standard curvature discretization. If any $Q\geqslant\eta$ for one of these nine points but $Q<\eta$ on

$$\mathscr{P}=(x_{i\pm1},y_j)\cup(x_i,y_{j\pm1})\cup(x_i,y_j), \tag{43}$$

then we use the alternate discretization of κ via

$$\kappa=\nabla\cdot\mathbf{n}, \tag{44}$$

where we use second-order centered differences for $\partial_x n_x$ and $\partial_y n_y$.

If $Q\geqslant\eta$ on any point in $\mathscr{P}$, we have found that no one-sided difference can stably calculate κ. In such a case, we apply an extension of the previously defined κ values in a manner similar to the components of the pressure gradient as described in Section 4.2. The only difference is that rather than fitting a higher-order (up to cubic) polynomial through the interpolated points and extrapolating, we fit a least-squares line through those data points. In our testing, we found that this gives much more stable results. Because the algorithm is one-sided, we apply it twice: once for the undefined curvature values outside Σ, and once for the undefined curvature values inside Σ.

4.7. The narrow band/local level set technique and the size of the computational domain

Following [25,30], we update φ within a distance R of the interface. Given an initialized level set function φ, the points which fall within that distance are

$$\{\mathbf{x}:|\varphi(\mathbf{x})|\leqslant R+\epsilon\} \tag{45}$$

for some small ϵ. This set is referred to as a "narrow band" about Σ, R is the width of the band, and the technique is known as the "narrow band" (or "local") level set method. The value of R is determined by the numerical implementation; we begin this determination by considering the smoothed normal velocity.

We require a smoothed normal velocity within three nodes of the interface. Thus, $R\geqslant 3\Delta x$. If σ is the standard deviation of the Gaussian filter, then the outermost of these smoothed points requires that $\widetilde{V}$ be defined within a rectangle that extends 3σ in all four mesh directions. The farthest node point within this rectangle is at a distance of $3\sqrt{2}\sigma$, so $R\geqslant 3\Delta x+3\sigma\sqrt{2}$.

To extend the velocity to the outermost of these points, we require a valid normal vector. As we obtain the normal vector with centered difference of φ, this requires one-to-two additional node points. Thus, $R\geqslant 3\Delta x+3\sqrt{2}\sigma+2\Delta x$. Lastly, we often multiply R by a safety factor because the interface tends to

change position between intermediate steps of the TVD-RK method. In our calculations, we chose a safety factor of 1.25. Thus, our band size is

$$R = 1.25(5\Delta x + 3\sqrt{2}\sigma) = R = 1.25(5 + 3M\sqrt{2})\Delta x, \tag{46}$$

where we have used $\sigma = M\Delta x$.

The size of the band for the narrow band level set method determines the size of the computational domain $\mathscr{D}$: it must be large enough to contain the contour $\{\mathbf{x}:\varphi(\mathbf{x}) \leqslant R + \epsilon\}$. We typically allow three-to-four additional nodes of buffer between the edge of the computational domain and this contour. Thus, whenever

$$|\varphi(\mathbf{x})| > R + 3\Delta x, \tag{47}$$

for any $\mathbf{x} \in \partial\mathscr{D}$, we must extend the computational domain. We therefore modify the width R of the narrow band to include this distance:

$$R = 1.25(5\Delta x + 3\sqrt{2}\sigma + 3\Delta x) = R = 1.25(8 + 3M\sqrt{2})\Delta x. \tag{48}$$

5. Convergence and testing results

5.1. Convergence of the full method: exact circular solution

We tested our algorithm on the full system (7)–(9) with $A = 0.5$, $G = 20$ and $N = 0$ (i.e., no necrosis). Note that G_N is not used in the absence of necrosis. The initial interface Σ is a circle of radius 2.0 centered at the origin. According to [7], if $R(t)$ denotes the radius of Σ at time t, the exact solution of this problem is given by solving

$$R\prime(t) = -AG\frac{R}{2} + G\frac{I_1(R)}{I_0(R)}, \quad R(0) = 2. \tag{49}$$

The level set function φ at time t is given by

$$\varphi(r,t) = r(x,y) - R(t), \tag{50}$$

where $r(x,y) = \sqrt{x^2 + y^2}$. To test the convergence, we measured the maximum absolute error at the common mesh points within the band $\mathscr{B} = \{(x,y) : |r(x,y) - R(t)| < 0.5\}$

$$\ell_\infty^{\text{band}}(\Delta x) = \max\{|\varphi_{\Delta x}(x_i,y_j,t) - \varphi_{\text{actual}}(x_i,y_j,t)| \,:\, (x_i,y_j) \in \mathscr{B}\} \tag{51}$$

at $t = 0.05$ to $t = 0.25$ in 0.05 increments. We tested with $\Delta x = \Delta y = 0.16$, $\Delta x = \Delta y = .08$, and $\Delta x = \Delta y = 0.04$. In all these calculations, we used linear interpolation of φ for the subcell resolution in the various Poisson solvers. We calculated curvature within a band of width 3 Δx of Σ, we set $\eta = 0.1$ for the normal vector and curvature algorithms, and we chose σ large enough to maintain the expected circular symmetry and convergence at every time step. The values of σ used are given in Table 2. Notice that σ decreases with refinement of the computational mesh, so there is no lower limit on the feature size that can be

Table 2
Filtering parameter σ used for each spatial resolution

Δx	σ
0.16	$2\Delta x = 0.32$
0.08	$3\Delta x = 0.24$
0.04	$4\Delta x = 0.16$

resolved by mesh refinement. All calculations used our new bilinear velocity extension technique. We define the overall convergence rate at a given time t to be

$$\text{convergence rate} = \frac{\log\left(\frac{\ell_\infty^{\text{band}}(\Delta x=.16)}{\ell_\infty^{\text{band}}(\Delta x=.04)}\right)}{\log 4}. \tag{52}$$

We tested all combinations of WENO and Poisson orders and determined that WENO5 with Poisson2 is the best combination of algorithms to yield full second-order accuracy. The convergence results for WENO5-Poisson2 are given in Table 3. Furthermore, we found that using Poisson1 yielded 1.6-order convergence with significantly larger errors. See Table 4 for a characteristic example with WENO5-Poisson1 and bilinear velocity extension. Similar results have been obtained for simulations including necrotic effects (see Section 6.1).

5.2. Convergence of the full method for complex morphology and comparison to boundary integral results

Consider the problem (7)–(9) with $A = 0.5$, $G = 20$ and $N = 0$ (i.e., no necrosis). We solve with $\Delta x = \Delta y = 0.08$, $\eta = 0.1$, WENO5, and Poisson2. The initial shape is given by

$$\Sigma(s) = (2 + 0.2\cos(2s), 2 + 0.2\sin(2s)), \quad 0 \leqslant s \leqslant 2\pi. \tag{53}$$

Table 3
Full convergence results for WENO5, Poisson2, bilinear velocity extension

Time	$\Delta x = .16$	$\Delta x = 0.08$	$\Delta x = 0.04$	Order
0.05	0.005691	0.001447	3.634e−4	1.98
0.10	0.01111	0.002765	7.424e−4	1.95
0.15	0.01572	0.003869	9.839e−4	2.00
0.20	0.01963	0.004942	0.001259	1.98
0.25	0.02405	0.005774	0.001467	2.02

Table 4
Full convergence results for WENO5, Poisson1, bilinear velocity extension

Time	$\Delta x = .16$	$\Delta x = .08$	$\Delta x = .04$	Order
0.05	0.01520	0.004157	0.001535	1.65
0.10	0.02650	0.008049	0.002896	1.60
0.15	0.03739	0.01111	0.004055	1.60
0.20	0.04663	0.01372	0.005074	1.60
0.25	0.05528	0.01733	0.006007	1.60

Table 5
Full convergence results for non-necrotic, complex morphology

Time	$\ell_\infty^{\text{band}}(\Delta x, 2\Delta x)$	$\ell_\infty^{\text{band}}(\Delta x, \frac{1}{2}\Delta x)$	Order
0.05	0.005650	0.001333	2.08
0.10	0.009698	0.002453	1.98
0.15	0.01339	0.003473	1.97
0.20	0.01607	0.004014	2.00
0.25	0.1850	0.004878	1.92
0.50	0.03282	0.007068	2.22
0.70	0.03417	0.009704	1.82

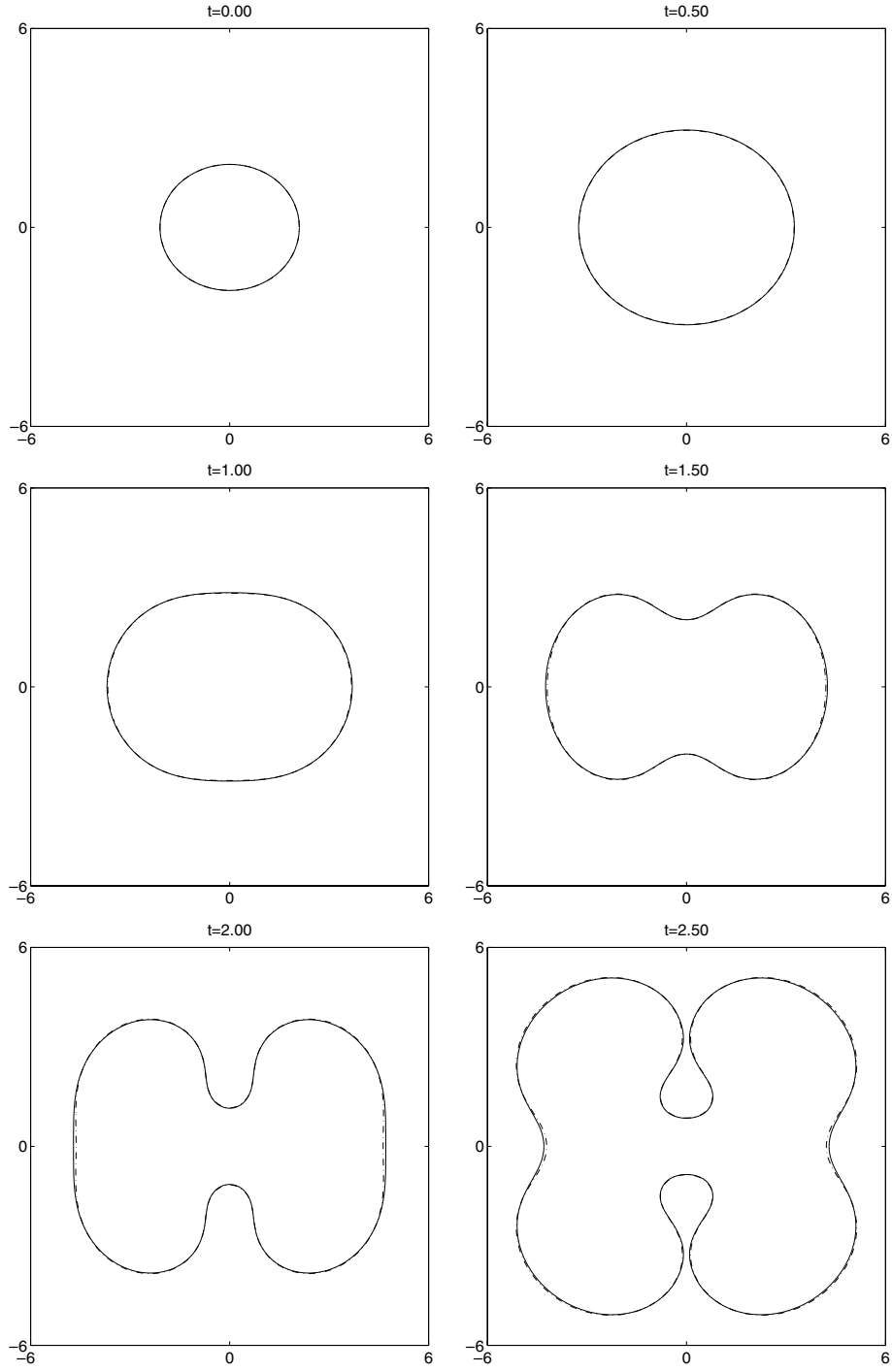

Fig. 6. Comparison of computed solutions. We compare the WENO5-Poisson2 (dashed curves) and boundary integral (solid curves) solutions from $t = 0.0$ to $t = 2.50$ in 0.5 increments.

There is no analytical solution for this case. Thus, to test convergence, we compare the solutions at different resolutions. This case has been investigated previously using boundary integral methods in [7]; we also compare our results to these.

In Table 5, the differences between the solutions at different mesh refinements in maximum norm and the associated orders of convergence are shown at various times. The differences are defined by

$$\ell_\infty^{\text{band}}(\Delta x_1, \Delta x_2) = \max\{|\varphi_{\Delta x_1}(x_i, y_j, t) - \varphi_{\Delta x_2}(x_i, y_j, t)| : (x_i, y_j) \in \mathscr{B}\}, \tag{54}$$

where the band $\mathscr{B}$ consists of the set of common mesh points within a distance of 0.5 of the interface. The order of convergence is given by

$$\text{convergence rate} = \frac{1}{\log 2} \log\left(\frac{\ell_\infty^{\text{band}} \Delta x, 2\Delta x}{\ell_\infty^{\text{band}} \Delta x, \frac{1}{2}\Delta x}\right). \tag{55}$$

The results in Table 5 clearly demonstrate that the overall solution is second-order accurate.

Next, we compare our simulation to the boundary integral result from [7]. In [7], it was shown that this tumor undergoes a morphological instability, and the evolving interface was accurately simulated using a spectrally accurate boundary integral method.

In Fig. 6, we compare our results (dashed curves) to the spectrally accurate results (solid curve) from [7]. This is an especially difficult test due to the morphological instability which makes the solution very sensitive to numerical errors. There is excellent agreement between the results. Shortly after the final time ($t = 2.531$) shown in Fig. 6, the boundary integral method breaks down as the tumor boundary self-intersects, resulting in the capture of healthy tissue within the tumor domain.

In Fig. 7, we continue our solution. As the tumor grows, healthy tissue is captured by the tumor multiple times as morphological stability occurs. It is well-known that healthy tissue often mixes with tumor tissue, especially near the tumor/healthy tissue boundary [24]. The collapse of tumors encapsulating healthy tissue has also been observed (S. Ramakrishnan, private communication [33]). In our simple model, these features are reflected through the multiple tumor boundary reconnections.

5.3. Impact of speed filtering

We now demonstrate the necessity of speed filtering in maintaining a first-order CFL time step restriction. We solve the same problem as in Section 5.2 with $\Delta x = \Delta y = 0.08$, $\eta = 0.1$, WENO5, Poisson2, and no speed filtering.

Without speed filtering, significant perturbations in the interface appear as early as $t = 0.02$, leading to large oscillations in the curvature because the coefficient of the curvature is order one in comparison to the other microphysical parameters. The large variations in curvature disturb the pressure solution and its gradient near the interface, which creates further feedback to disturb the interface. In Fig. 8, we see that these disturbances quickly grow to destabilize the entire simulation (solid curve). We show the same calculation with speed filtering for comparison (dashed curve).

5.4. Impact of the new velocity extension technique

We tested the impact of the new bilinear velocity extension technique by solving the same problem as in Section 5.2 with either the bilinear velocity extension or the traditional PDE-based velocity extension, where one solves the PDE

$$\widetilde{V}_\tau + \text{sign}(\varphi)\mathbf{n} \cdot \nabla \widetilde{V} = 0 \tag{56}$$

to steady state. Here, $\widetilde{V}(\tau = 0)$ equals the unextended velocity [37].

The results, shown in Fig. 9 at $t = 2.50$, demonstrate that the bilinear velocity extension (dashed) gives results superior to those obtained with the PDE-based velocity extension (dash-dotted) when compared to the boundary integral results (solid) from [7].

5.5. Impact of the curvature and normal vector modifications

To study the impact of our modifications to the curvature and normal vector, we solved the system (7)–(9) again with the same setup as in Section 5.2. In Fig. 10, we show the position of the interface at $t = 2.5$, 2.75, and 2.77 with both the standard (top) and modified (bottom) curvature and normal vector algorithms. Because our speed extension requires normal vectors that point away from the interface, we used the standard PDE-based extension (see [37]) for simulations with the standard curvature and normal vector routines. In the results that use the standard discretizations, an artificial "repulsive" effect can be seen that prevents approaching interfaces from merging until much later times when numerical error finally causes them to merge. In Fig. 11, we plot the contours of the curvatures at $t = 2.5$ around the merging interfaces for both algorithms. As we can see, the modified curvature (right) is smooth, whereas the standard curvature (left) is oscillatory.

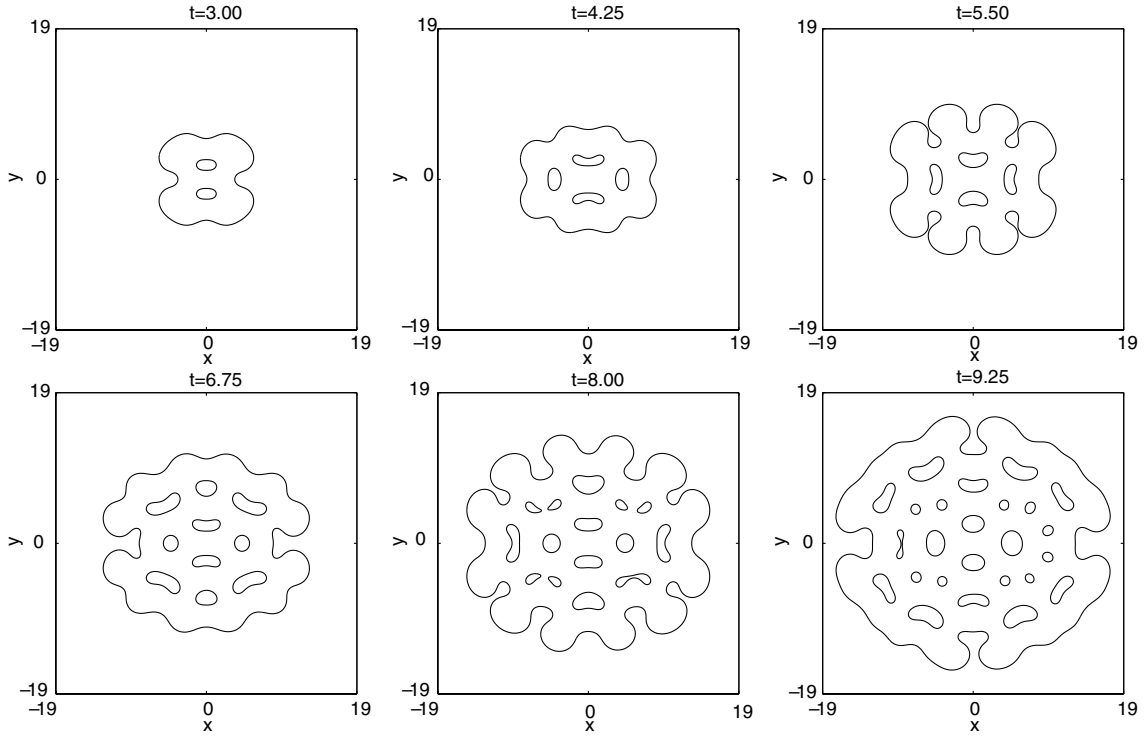

Fig. 7. Comparison of computed solutions. We continue our WENO5-Poisson2 solution to additional times which the boundary integral method cannot compute. Note that the spatial scale is different than in Fig. 6.

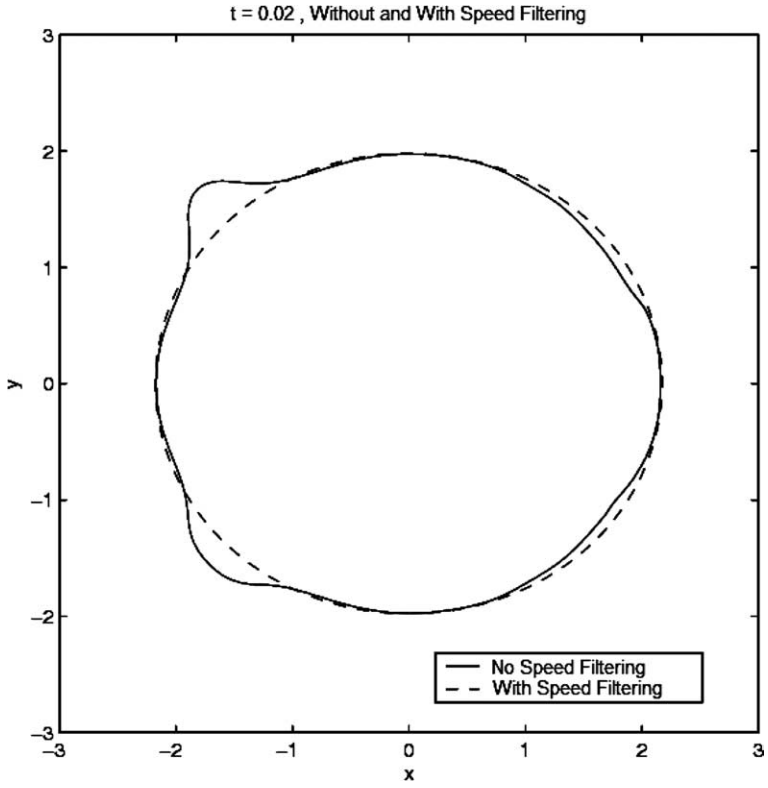

Fig. 8. Effect of filtering on overall stability and accuracy. Initially small perturbations have grown to grossly distort the shape of the interface by $t = 0.02$. The dashed curve shows the solution at the same time with speed filtering.

6. Numerical examples with necrotic effects

6.1. An example with symmetric initial data

To demonstrate the robustness of our technique, we next solve the system with necrosis. The initial interface is given by (53) as before. Here, $A = 0.0$ (i.e., no apoptosis), $G = 20.0$, $N = 0.35$, and $G_N = 1.0$. Again, we use $\Delta x = \Delta y = 0.08$, $\sigma = 3\Delta x = 0.24$, and $\eta = 0.1$. We use Poisson2 and WENO5 with our bilinear velocity extension method.

We begin by demonstrating the convergence of the overall numerical solution. As in Section 5.2, we determine the order by considering ratios of the differences in the numerical solutions at three spatial resolutions ($2\Delta x$,Δx, and $\frac{1}{2}\Delta x$). The results, given in Table 6, clearly demonstrate the second-order accuracy of the overall method, indicating that necrosis (and the associated discontinuities in the second derivatives of the pressure) does not affect the result.

In Fig. 12, the morphologies of the growing tumor are shown. In this figure, we see that as the tumor grows, healthy tissue is captured as morphological instability occurs, just as in the non-necrotic case (see Figs. 6 and 7). This capture of healthy tissue is repeated, leading to a complex, lattice-like structure. A necrotic core (indicated as a black region) first develops in the center of the tumor, splits, and also changes morphology multiple times.

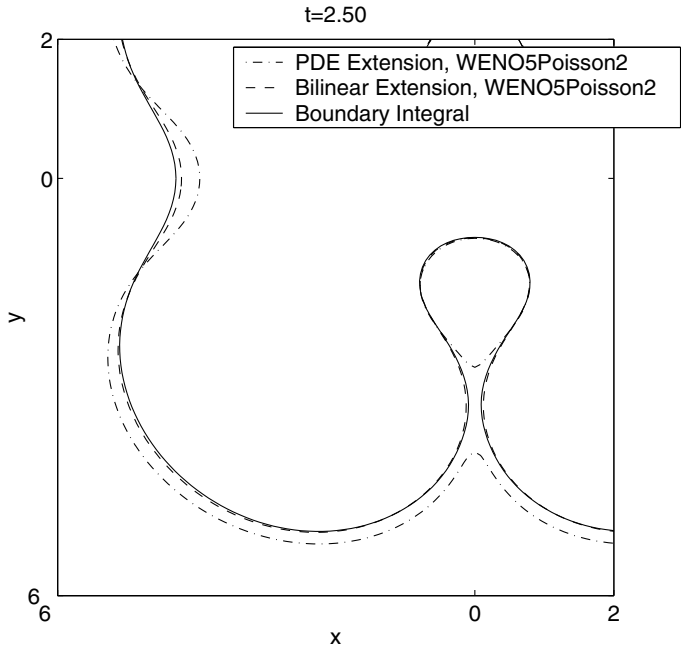

Fig. 9. Comparison of the velocity extension techniques at $t = 2.50$ with Poisson2 and WENO5. The boundary integral solution is given by the solid curve.

In our implementation, $c = 1$ on the boundary of the captured regions. This mimics the vascularization of the tumor from a third spatial dimension, thereby providing an source of nutrient internal to the tumor. In addition, the velocity along the boundaries of the captured regions is determined from Darcy's law, where the pressure gradient is taken from the tumor side of the interface. This allows the captured regions to grow or shrink depending upon the local tumor pressure gradient. For example, if the normal velocity is negative, this mimics the pulling of healthy tissue from the third spatial dimension into the tumor interior. This expansion could also be interpreted as mimicking the compressibility of the healthy tissue. (In the future, we will modify these internal boundary conditions to be more realistic and allow nutrient to diffuse into the captured healthy tissue instead, and we will and prescribe limits on the volume change of the captured regions.).

Observe in Fig. 12 that the necrotic core is roughly equidistant to the tumor/healthy tissue interfaces; the distance is the (diffusion) length that the nutrient molecules diffuse before they are consumed by cells. Beyond this diffusion distance, the levels of nutrient are too low for cells to be viable.

Note that it would be very difficult to perform such a simulation with the boundary integral techniques used in [7] due to the frequent morphological changes in both the tumor boundary and the necrotic core.

6.2. An example with asymmetric initial data

We give one final example with necrotic effects. We consider an asymmetric initial interface. We again solve (7)–(9) with the initial interface as given in Fig. 13. Here, $A = 0.5$, $G = 20.0$, $N = 0.5$, and $G_N = 1.0$. As before, we use $\Delta x = \Delta y = 0.08$, $\sigma = 3\Delta x = 0.24$, and $\eta = 0.1$. We use Poisson2 and WENO5 with the bilinear velocity extension. We plot our solution in 1.00 time-unit increments in Fig. 13. An evolution analogous to that seen in Figs. 6, 7 and 12 is observed. Because the necrotic parameter is larger in this simulation, the

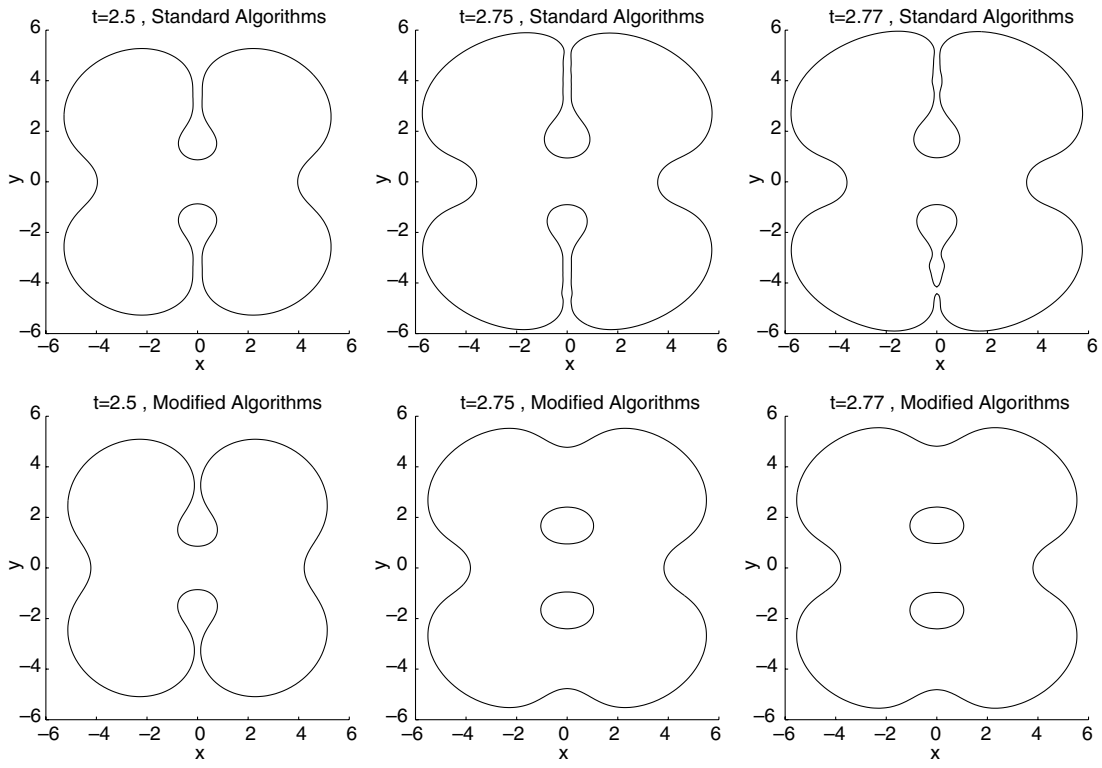

Fig. 10. Effect of the curvature and normal vector modifications on a tumor growth simulation. The plots show the solution to the problem in Section 5.3 at $t = 2.5$, $t = 2.75$, and $t = 2.77$. The top row shows the calculation using standard centered differences for κ and $\mathbf{n}$; the bottom row shows the same calculation with our modified algorithms.

lattice-like structure observed in Figs. 7 and 12 is more pronounced. Notice that despite the initial asymmetry, the lattice structure attains a level of regularity in its pattern. In addition, growth occurs through a "bump-by-bump" mechanism, where the tumor expands by the growth of small bumps that invade the neighboring region. This growth mechanism has been recently observed by Frieboes et al. [10] in experiments on tumor spheroids (see Fig. 14).

7. Conclusions and future work

In this paper, we developed a second-order accurate ghost fluid/level set algorithm for the evolution of interfaces whose normal velocity is given by the normal derivatives of solutions to interior Poisson equations with curvature-dependent boundary conditions. The algorithm is capable of describing complex morphologies including pinchoff and merger of interfaces. In particular, we developed a new Poisson solver capable of capturing geometric boundary conditions on a complicated interface. We developed geometry-aware discretizations of the normal vectors and curvature that automatically detect and cope with level set irregularity, particularly during morphological changes. We also developed new gradient and velocity extension techniques that take full advantage of the geometric information embedded in the level set function to obtain greater accuracy and faster computational speed than techniques currently in use. To main-

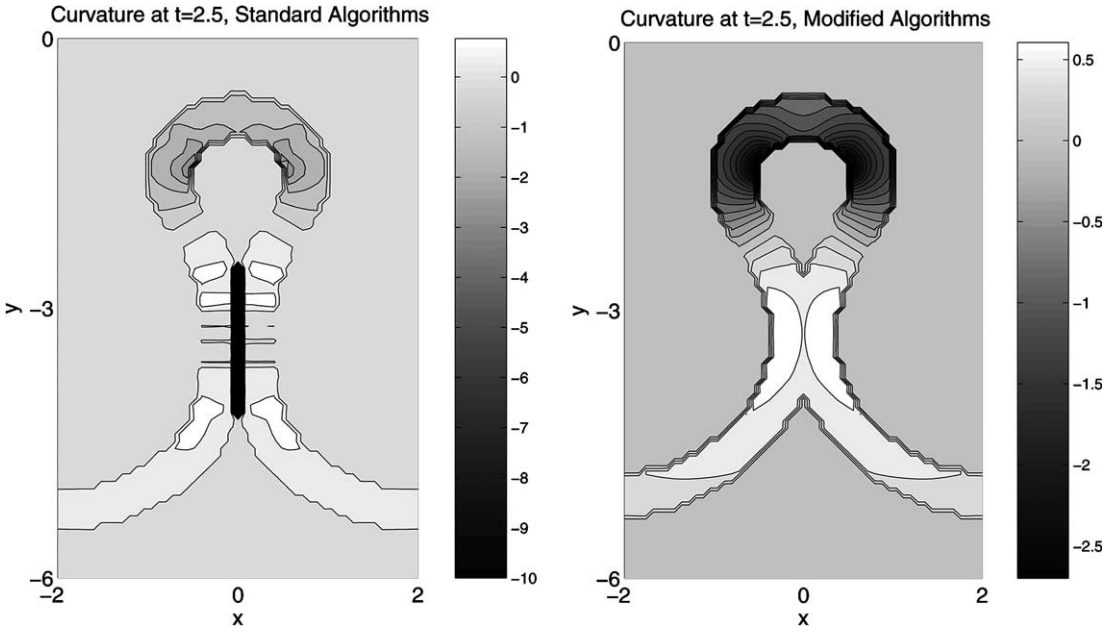

Fig. 11. Effect of the curvature and normal vector modifications on a tumor growth simulation. The left plot shows the curvature at $t = 2.5$ using the standard algorithms (the black regions are where $\kappa \sim -1e3$; the right plot shows the curvature using our modified algorithms at the same time).

Table 6
Full convergence results for necrotic, complex morphology

Time	$\ell_\infty^{\text{band}}(\Delta x, 2\Delta x)$	$\ell_\infty^{\text{band}}(\Delta x, \frac{1}{2}\Delta x)$	Order
0.05	0.004290	0.001153	1.89
0.10	0.006469	0.001612	2.00
0.15	0.006852	0.001820	1.91
0.20	0.009999	0.002754	1.86
0.25	0.01360	0.003528	1.95
0.30	0.01804	0.004908	1.88

tain stability, we applied Gaussian filter techniques often used in image processing to smooth the extended velocity while preserving the overall accuracy.

We validated the algorithm by simulating a model for tumor growth and comparing the numerical results to exact solutions and to spectrally accurate boundary integral results. We provided numerical evidence that our algorithm (i.e., WENO5 for the level set equation, Poisson2 for the interior Poisson equations, linear interpolation to determine the interface position, cubic interpolation for the curvature at the interface, bilinear velocity extension off the interface and Gaussian filtering for the normal velocity) indeed achieves full second-order accuracy, even when the coefficient of the curvature is order one (with respect to the other microphysical parameters). This is the first such demonstration we are aware of in the context of a fully coupled, nonlinear moving boundary problem with geometric boundary conditions (curvature).

We also went beyond the morphologies that can be described by the boundary integral method and presented accurate simulations of complex, evolving tumor morphologies that demonstrate the repeated

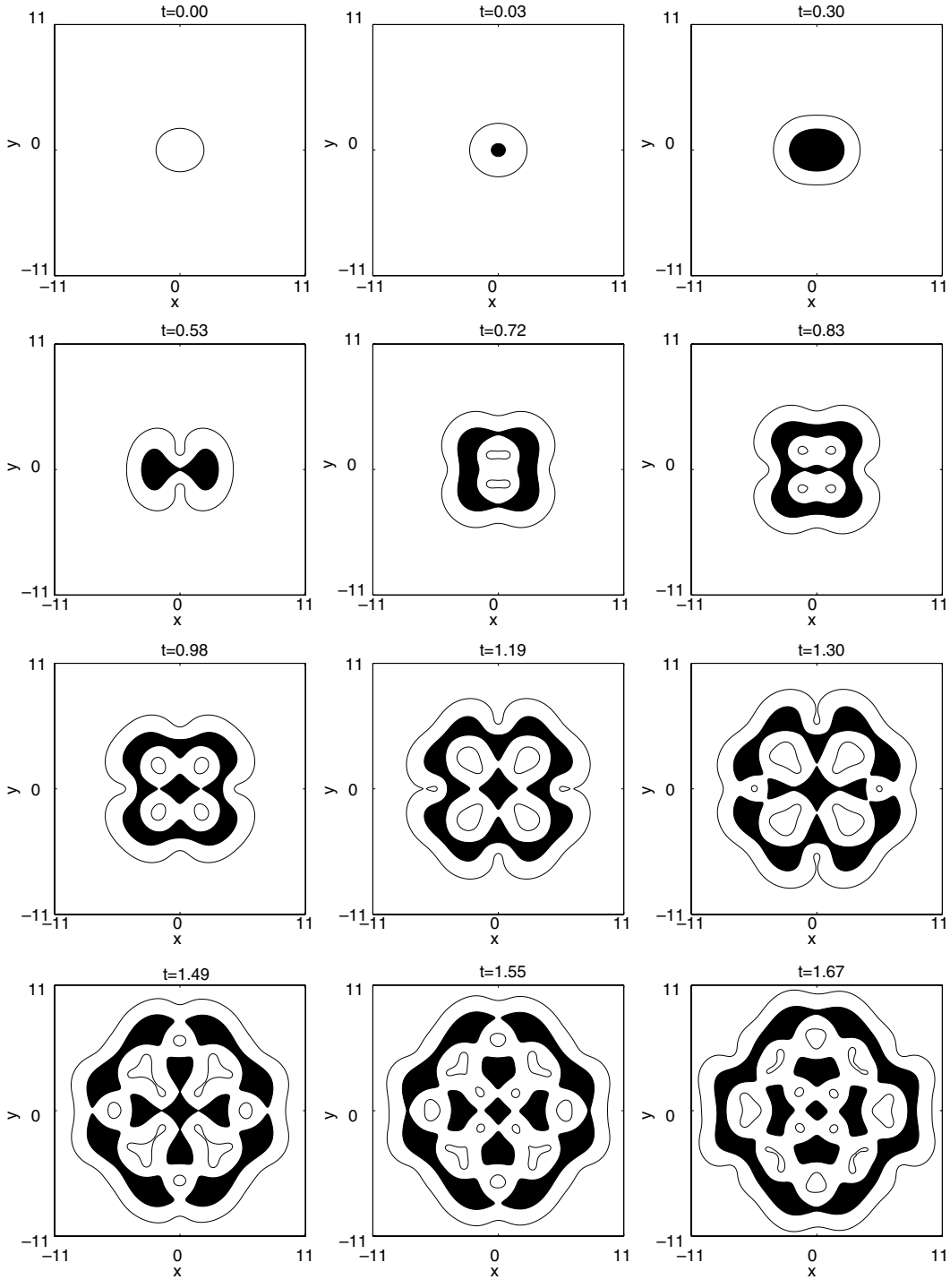

Fig. 12. A simulation including necrotic effects. The necrotic regions are shown in black.

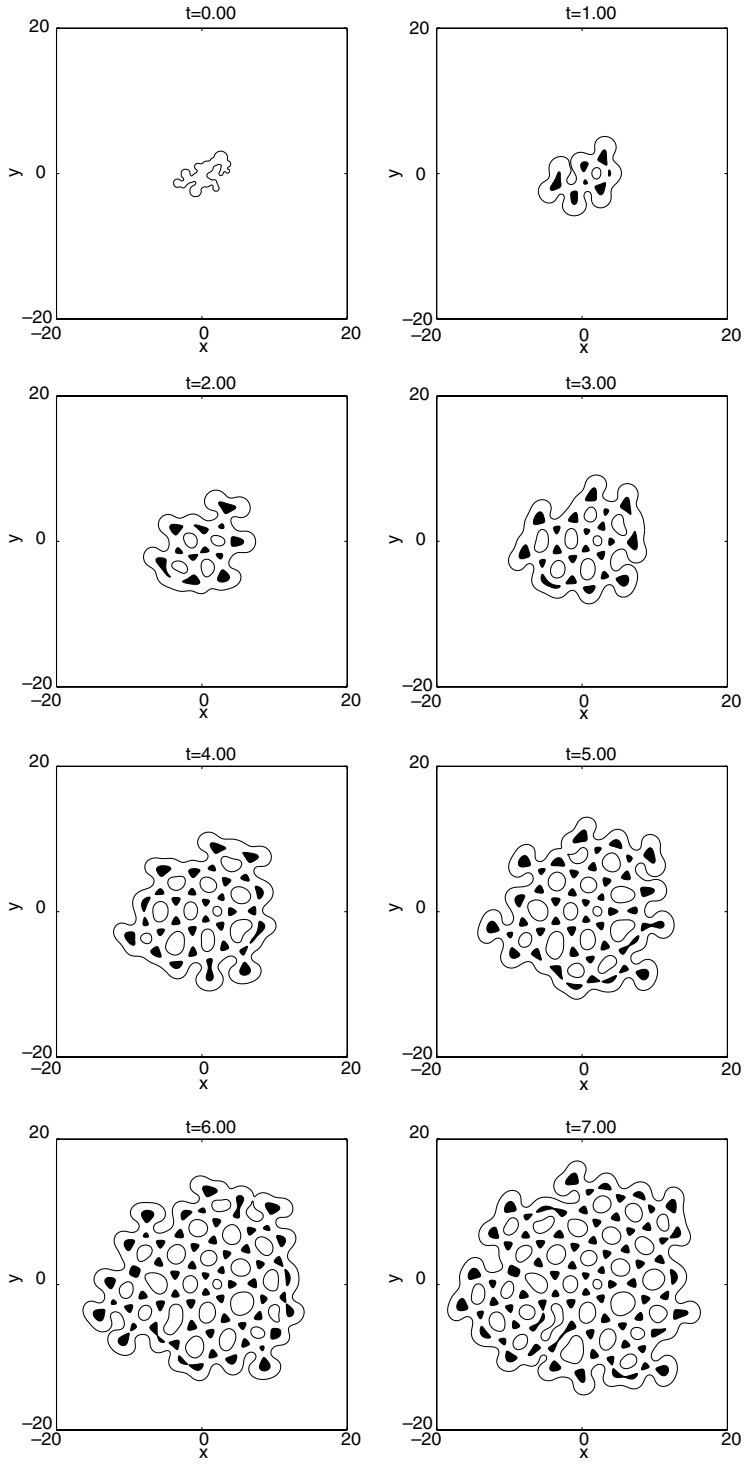

Fig. 13. A simulation including necrotic effects with asymmetric initial data. The necrotic regions are shown in black.

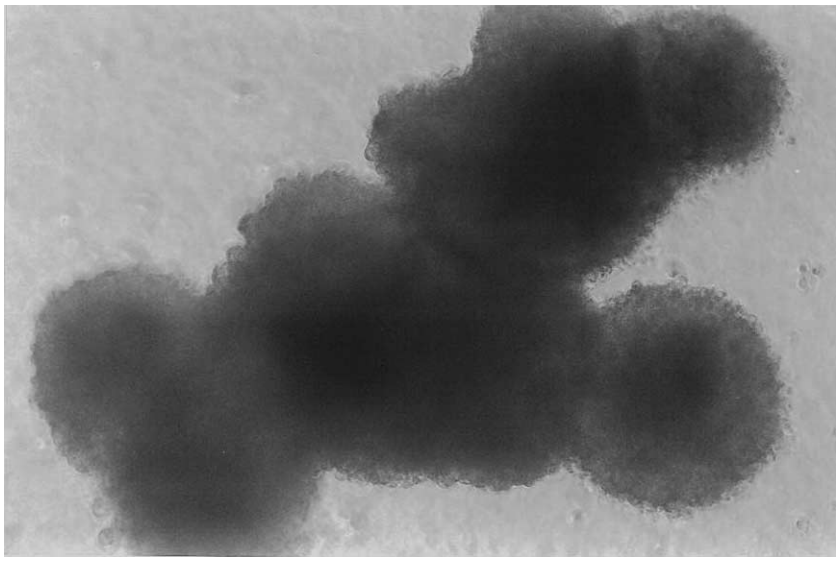

Fig. 14. In vitro glioblastoma from Frieboes et al. [10] growing by the "bump-by-bump" mechanism.

encapsulation of healthy tissue in the primary tumor domain– an effect seen in the growth of real tumors. In future work, we plan to continue developing and simplifying our new normal vector, curvature, and gradient extension routines. In addition, we will also develop implicit time integration schemes as an alternative means to remove the stiffness. We will apply and enhance the numerical techniques we have developed to study the biophysics of tumor growth. In a future work, we will consider more realistic microphysical parameters (i.e., A, G and G_N) by allowing spatial and temporal variability in order to study the morphological response of the tumors to chemotherapy, tissue inhomogeneity, and genetic mutations with the basic techniques developed here [23].

Acknowledgements

The authors gratefully thank Vittorio Cristini for discussions concerning this work. The authors also thank the Network and Academic Computing Services (NACS) at the University of California at Irvine (UCI) and the Minnesota Supercomputer Institute for generous computing resources. P. Macklin acknowledges support from the the National Science Foundation (Graduate Research Fellowship) as well as the Department of Mathematics and the Department of Biomedical Engineering at UCI. J. Lowengrub thanks the National Science Foundation (mathematics division) for partial support. We gratefully thank the mathematics and biomedical engineering departments for the use of computational resources funded by a National Science Foundation SCREMS grant and the Whitaker Foundation.

Appendix A. Extrapolations for the Poisson solver

In Section 4.1, if Σ intersected $[x_{i-1},x_{i+1}]$, we discretized u_{xx} at x_i by replacing u_{i+1} (or u_{i-1}) by an extrapolated value $\hat{u}_{i+1}$ (or $\hat{u}_{i-1}$). For completeness, we give the extrapolations for these cases here.

If $x_i < x_\Sigma \leqslant x_{i+1}$, then we extrapolate from interior and boundary data to replace $\hat{u}_{i+1}$ in the discretization of u_{xx} at x_i. Some possible discretizations include:

(1) *Cubic extrapolation:* If $\{x_{i-3}, x_{i-2}, x_{i-1}\} \subset \Omega$ as described in (15), then we extrapolate $\hat{u}_{i+1}$ from u_{i-3}, u_{i-2}, u_{i-1}, and u_Σ, where

$$u_\Sigma = u(x_\Sigma) = g(x_\Sigma) = g_\Sigma. \tag{A.1}$$

(2) *Quadratic extrapolation:* If $\{x_{i-2}, x_{i-1}\} \subset \Omega$, then we define $u_\Sigma = g_\Sigma$ as before, and we extrapolate $\hat{u}_{i+1}$ from u_{i-2}, u_{i-1}, and u_Σ.

(3) *Linear extrapolation:* If $x_{i-1} \in \Omega$, then we define $u_\Sigma = g_\Sigma$ as before, and we define a linear extrapolation via

$$\hat{u}_{i+1} = (1-\theta)(u_i - u_{i-1}) + g_\Sigma. \tag{A.2}$$

If $x_{i-1} \leqslant x_\Sigma < x_i$, then we extrapolate from interior and boundary data to replace $\hat{u}_{i-1}$ in the discretization of u_{xx} at x_i. Some possible discretizations include:

(1) *Cubic extrapolation:* If $\{x_{i+3}, x_{i+2}, x_{i+1}\} \subset \Omega$, then we use cubic extrapolation from u_{i+3}, u_{i+2}, u_{i+1}, and u_Σ, where

$$u_\Sigma = u(x_\Sigma) = g(x_\Sigma) = g_\Sigma. \tag{A.3}$$

(2) *Quadratic extrapolation:* If $\{x_{i+2}, x_{i+1}\} \subset \Omega$, then we define $u_\Sigma = g_\Sigma$ as before, and we extrapolate $\hat{u}_{i-1}$ from u_Σ, u_{i+1}, and u_{i+2}.

(3) *Linear extrapolation:* If $x_{i+1} \in \Omega$, then we define $u_\Sigma = g_\Sigma$ as before, and our linear extrapolation of $\hat{u}_{i-1}$ is

$$\hat{u}_{i-1} = g_\Sigma - \theta(u_{i+1} - u_i). \tag{A.4}$$

References

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

[2] J. Adam, General aspects of modeling tumor growth and immune response, in: J. Adam, N. Bellomo (Eds.), A Survey of Models on Tumor Immune Systems Dynamics, Birkhauser, Boston, 1996, pp. 15–87.

[3] H.M. Byrne, M.A.J. Chaplain, Growth of necrotic tumors in the presence and absence of inhibitors, Math. Biosci. 135 (1996) 187–216.

[4] H.M. Byrne, M.A.J. Chaplain, Modelling the role of cell–cell adhesion in the growth and development of carcinomas, Math. Comput. Model. 24 (1996) 1–17.

[5] M.A.J. Chaplain, G.D. Singh, J.C. MacLachlan (Eds.) On Growth and Form: Spatio-temporal Pattern Formation in Biology, Wiley Series in Mathematical and Computational Biology, New York, NY, 1999.

[6] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving Stefan problems, J. Comput. Phys. 135 (1) (1997) 8–29.

[7] V. Cristini, J.S. Lowengrub, Q. Nie, Nonlinear simulation of tumor growth, J. Math. Biol. 46 (3) (2003) 191–224.

[8] J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst, Numerical Linear Algebra for High-Performance Computers, Philadelphia, PA (1998), ISBN 0-89871-428-1.

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

[10] H. Frieboes, C.-H. Sun, B. Tromberg, V. Cristini, Diffusional instability as a mechanism for glioblastoma growth and invasion, 2004 (in preparation).

[11] F. Gibou, R. Fedkiw, A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem, J. Comput. Phys. 202 (2) (2005), in press.
[12] F. Gibou, R. Fedkiw, R. Caflisch, S. Osher, A level set approach for the numerical simulation of dendritic growth, J. Sci. Comput. 19 (2003) 183–199.
[13] F. Gibou, R. Fedkiw, L.T. Cheng, M. Kang, A second order accurate symmetric discretization of the Poisson equation on irregular domains, J. Comput. Phys. 176 (1) (2002) 205–227.
[14] J. Glimm, D. Marchesin, O. McBryan, A numerical-method for 2 phase flow with an unstable interface, J. Comp. Phys. 39 (1981) 179–200.
[15] R. Gonzalez, R. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1992, ISBN 0-201-18075-8.
[16] S. Gottlieb, C.W. Shu, Total variation diminishing Runge–Kutta schemes, Math. Comp. 67 (1998) 73–85.
[17] S. Gottlieb, C.W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Rev. 43 (1) (2001) 89–112.
[18] G.S. Jiang, D. Peng, Weighted ENO schemes for multi-dimensional Hamilton–Jacobi equations, SIAM J. Sci. Comput. 21 (6) (2000) 2126–2143.
[19] G.S. Jiang, C.W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (2) (1996) 202–228.
[20] A.A. Kansal, S. Torquato, G.R. Harsh IV, E.A. Chiocca, T.S. Deisboeck, Simulated brain tumor growth dynamics using 3-D cellular automaton, J. Theor. Biol. 203 (4) (2000) 367–382.
[21] X.D. Liu, R. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, J. Comput. Phys. 160 (1) (2000) 151–178.
[22] P. Macklin, Numerical simulation of tumor growth and chemotherapy, M.S. Thesis, University of Minnesota School of Mathematics, September 2003.
[23] P. Macklin, J. Lowengrub, Nonlinear simulations of tumor response to chemotherapy, 2004 (in preparation).
[24] E.A. Maher, F.B. Furnari, R.M. Bachoo, D.H. Rowitch, D.N. Louis, W.K. Cavenee, R.A. DelPinho, Malignant glioma: genetics and biology of a grave matter, Genes Dev. 15 (2001) 1311–1333.
[25] R. Malladi, J.A. Sethian, B.C. Vemuri, A fast level set based algorithm for topology-independent shape modeling, J. Math. Imaging Vision 6 (1996) 269–289.
[26] R. Malladi, J.A. Sethian, B.C. Vemuri, Shape modeling with front propagation: a level set approach, IEEE Trans. Pattern Anal. Mach. Intell. 17 (2) (1995).
[27] D.L.S. McElwain, L.E. Morris, Apoptosis as a volume loss mechanism in mathematical models of solid tumor growth, Math. Biosci. 39 (1978) 147–157.
[28] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[29] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, New York, NY, 2002, ISBN 0-387-95482-1.
[30] D. Peng, B. Merriman, S. Osher, H.K. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (1999) 410ff.
[31] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, 1992, ISBN 0-521-43108-5.
[32] L. Preziosi, Cancer Modeling and Simulation, Boca Raton, LA , 2003, ISBN 1-58488-361-8.
[33] S. Ramakrishnan, Department of Pharmacology, University of Minnesota, Personal Communication.
[34] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, New York, NY, 1999, ISBN 0-521-64557-3.
[35] M. Sussman, E. Fatemi, An efficient, interface preserving level set re-distancing algorithm and its application to interfacial incompressible fluid flow, SIAM J. Sci. Comput. 20 (4) (1999) 1165–1191.
[36] M. Sussman, E. Fatemi, P. Smereka, S. Osher, An improved level set method for incompressible two-phase flows, Comput. Fluids 27 (5–6) (1998) 663–680.
[37] H. Zhao, T. Chan, B. Merriman, S. Osher, A variational level set approach to multiphase motion, J. Comput. Phys. 127 (1996) 179ff.
[38] X. Zheng, S.M. Wise, V. Cristini, Nonlinear simulation of tumor necrosis, neo-vascularization and tissue invation via an adaptive finite-element/level-set method, B. Math. Biol. (in press).