UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a master's thesis by

Paul Thomas Macklin

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

John Lowengrub

Name of Faculty Advisor

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

Numerical Simulation of Tumor Growth and Chemotherapy

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Paul Thomas Macklin

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

September 2003

©Paul Thomas Macklin 2003

Acknowledgements

I would like to thank the National Science Foundation, which funded this work through a Graduate Research Fellowship. The Institute for Mathematics and its Applications (IMA) contributed invaluable computer resources and support; furthermore, the IMA provided a positive, friendly work environment that made a great difference for me. I also want to thank my advisor, John Lowengrub, for his unsurpassed dedication to his students. Last in this list but first in my heart, I want to thank my wife, Angela, without whose support and enthusiasm this research would never have happened. Thanks for making this fun!

Abstract

We implement numerically a model for free boundary necrotic and non-necrotic tumor growth and chemotherapy, where the tumor-healthy tissue interface is a moving deformable boundary. In the process, we improve upon existing techniques and develop new finite difference and 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. Our new methods include a robust boundary condition-capturing Poisson solver, improved discretizations of the normal vector and curvature, a new technique for extending variables beyond the zero level set, and a new application of Gaussian filter technology ordinarily associated with image processing. We conduct some parameter studies on 2D necrotic and non-necrotic tumor growth with and without chemotherapy, and we conclude with a 2D simulation of tumor breakup while undergoing chemotherapy.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation for this Research

While the past few decades have seen considerable progress in cancer research and encouraging improvements in survival rates, cancer remains a timely problem. For instance, in 1980, malignant neoplasms (tumors) were the second-leading cause of death at 21% of all deaths in the United States; twenty years later, cancer remains the second-leading cause of death at an increased rate of 23% of all deaths [1].

In the meantime, the level of cooperation between medicine, the mathematics, and the physical sciences has increased. Interdisciplinary journals are thriving (e.g., Mathematical Biosciences and the Journal of Mathematical Biology), and techniques from once unrelated fields such as computational fluid mechanics are being applied to biophysical and biomedical problems. Computer technology is advancing rapidly even as it becomes less expensive, allowing for the implementation of increasingly sophisticated models at lower overall cost.

Given these trends, the time is ripe to develop and numerically implement mathematical models for tumor growth and chemotherapy, and such work has already begun. (See [5, 6, 17, 24, 10, 2] for some examples and further sources.) The successful development and deployment of such a model would have a great impact on cancer research and the development of treatments. Drug researchers could use inexpensive computer power to analyze potential drug treatments, thereby allowing more-focused, accelerated development of the most-promising drugs. Chemotherapists could use the computational models to optimize treatment regimens while reducing patients' drug exposure. Practitioners could use such a computational tool to improve their diagnosis of cancer and tailor available treatment options to individual patients.

1.2 Summary of New Techniques Developed

We began our numerical implementation with well-established finite difference and level set techniques from computational fluid mechanics. We did not use the boundary integral method previously implemented in [10, 11] because it is not equipped for the complex morphological changes observed in tumor masses. We chose finite difference techniques over finite element methods because they allowed for more rapid development.

In our first attempts at numerical implementation, we used essentially non-oscillatory (ENO) methods and total variatation diminishing Runge-Kutta methods, along with a boundary condition capturing Poisson solver as presented in [13]. We choose these methods for their robustness, their lack of explicit smoothing, and their potential for high accuracy on a Cartesian grid. However, these methods yielded sub-first-order overall convergence on the full moving boundary problem. We consequently turned our attention to the developement of a state-of-the-art method to obtain overall second-order accuracy in space and time. We also aimed to develop methods that could be readily extended to three dimensions.

To attain this order of accuracy, we first developed a new Poisson solver capable of capturing geometric (e.g., curvature) and other boundary data on complicated geometries. Our technique is an extension of the ghost fluid method introduced by Aslam, Fedkiw, Merriman, and Osher in [4] to higher accuracy with modifications and simplifications inspired by [13].

We then developed curvature and normal vector techniques that automatically detect discontinuities in the level set function and apply appropriate one-sided differences, allowing greater stability and accuracy during morphological changes. These techniques differ from the current approaches which typically use centered differences for the curvature without considering the geometry and use weighted averages for the normal vector. (See, for example, [9] and [26].)

When we used traditional PDE-based velocity extension techniques [23], we found that the performance was unstable and highly inaccurate because the tumor equations are solved on the interior of the moving boundary, and so the ENO (and weighted WENO) discretizations had insufficient data to construct the difference operators. We therefore developed a pre-extension routine that took full advantage of the geometric information afforded by the level set function to extend the components of the gradient (or any quantity) using polynomial extension that preserves information flow outward from the interface. We then have enough data to extend the velocity using the standard PDE technique in [23]. We apply Gaussian filter techniques ordinarily used in image processing to smooth the extended normal velocity, thereby preserving long-term stability without decreasing the overall order of accuracy.

2 Governing Equations

2.1 Formulation of the Model

We shall implement and study a model for solid and necrotic tumor growth given by Cristini, Lowengrub, and Nie in [10, 11], which is a new formulation of several classical models [5, 6, 17, 24]. In this model, we study a tumor occupying a volume $\Omega(t)$ with boundary $\partial\Omega$, which we shall denote by Σ. We describe the diffusion and consumption of nutrients (e.g., oxygen and glucose) with a single nutrient concentration σ which accounts for their net effect. If Γ is the rate at which nutrient concentration is added at any point and D is a diffusion constant, then the quasi-steady nutrient concentration is given by

$$0 = D\nabla^2\sigma + \Gamma. \tag{1}$$

The quasi-steady state assumption holds because nutrient diffusion occurs on a much faster time scale than tumor growth.

We assume the amount of nutrient concentration consumed by the tumor cells per time is $\lambda\sigma$, where λ is uniform. In the healthy tissue, nutrient is diffused but not consumed. Nutrient concentration is delivered by the blood vasculature at a linear rate $-\lambda_B(\sigma - \sigma_B)$, where σ_B is the (uniform) nutrient concentration in the blood; λ_B may vary with position.[1] Thus, we have

$$\Gamma = -\lambda_B\left(\sigma - \sigma_B\right) + \begin{cases} -\lambda\sigma & \text{if } \mathbf{x} \in \Omega \\ 0 & \text{else,} \end{cases} \tag{2}$$

and

$$0 = D\nabla^2\sigma - \lambda_B\left(\sigma - \sigma_B\right) + \begin{cases} -\lambda\sigma & \text{if } \mathbf{x} \in \Omega \\ 0 & \text{else.} \end{cases} \tag{3}$$

As nutrient is consumed throughout the volume Ω of the tumor, an intra-tumor nutrient concentration gradient develops and σ decreases with distance from Σ. If the concentration drops below a minimum value σ_N for tumor cell viability, the cells undergo necrosis, leading to the formation of a necrotic core [11]. We denote this necrotic core by $\Omega_N \subseteq \Omega$ and define $\Omega_P \subseteq \Omega$ to be the region in which the tumor cells are proliferating. Thus,

$$\Omega_N = \left\{\mathbf{x} : \sigma(\mathbf{x}) < \sigma_N\right\}. \tag{4}$$

We denote the interface between the proliferating and necrotic regions by Σ_N. Note that $\Omega = \Omega_P \cup \Sigma_N \cup \Omega_N$. Please see Figure 1.

[1]Note that if $\sigma > \sigma_B$, delivery occurs at a positive rate.

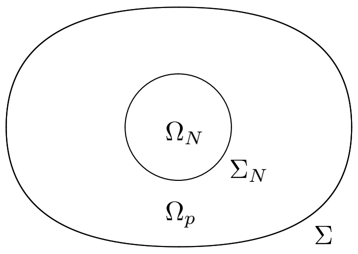

Figure 1: Tumor with a necrotic core.

We model the tumor as an incompressible fluid with a source due to cell proliferation and a sink due to degradation of the necrotic cells; the velocity field $\mathbf{u}$ satisfies

$$\nabla \cdot \mathbf{u} = \begin{cases} \lambda_P & \text{in } \Omega_P \\ -\lambda_N & \text{in } \Omega_N \\ 0 & \text{else,} \end{cases} \tag{5}$$

where λ_P is the net cell proliferation rate and λ_N is the rate of volume loss due to disintegration and enzymatic breakdown of cellular debris. We model λ_P by

$$\lambda_P = b\sigma - \lambda_A, \tag{6}$$

where b is a constant related to the cell mitosis rate and λ_A is the uniform apoptosis (cell death) rate. We note that the model sets a characteristic mitosis rate

$$\lambda_M = b\sigma^\infty, \tag{7}$$

where σ^∞ is a uniform nutrient concentration far from Σ.

We assume the velocity satisfies Darcy's law [17]:

$$\mathbf{u} = -\mu \nabla P, \tag{8}$$

where μ is the cell mobility and P is the pressure.

Far from Σ, we assume the nutrient concentration is a constant σ^∞. If Ω_D is the computational domain containing Ω and if $\partial\Omega_D$ is far from Σ, then we apply

$$\sigma\Big|_{\partial\Omega_D} = \sigma^\infty. \tag{9}$$

We model the cell-to-cell adhesive forces in Ω by a surface tension on the boundary Σ (a Laplace-Young boundary condition), and far from the interface, $P = P_\infty$. Thus,

$$[P]\Big|_\Sigma = \gamma\kappa \tag{10}$$

$$P\Big|_{\partial\Omega_D} = P_\infty, \tag{11}$$

where γ is constant and κ is the mean curvature.

By (5) and (8), the pressure satisfies the system

$$\begin{aligned} -\mu\nabla^2 P &= \begin{cases} b\sigma - \lambda_A & \text{in} \quad \Omega_P \\ -\lambda_N & \text{in} \quad \Omega_N \\ 0 & \quad\text{else,} \end{cases} \\ [P]\Big|_\Sigma &= \gamma\kappa, \\ P\Big|_{\partial\Sigma_D} &= P_\infty. \end{aligned} \tag{12}$$

We note that as $P \equiv P_\infty$ satisfies $\nabla^2 P = 0$, $P|_{\partial\Omega_D} = P_\infty$, we can replace the above by

$$\begin{aligned} -\mu\nabla^2 P &= \begin{cases} b\sigma - \lambda_A & \text{in} \quad \Omega_P \\ -\lambda_N & \text{in} \quad \Omega_N, \end{cases} \\ P\Big|_\Sigma &= P_\infty + \gamma\kappa, \\ P \equiv P_\infty &\quad \text{in} \quad \Omega_D - \overline{\Omega}. \end{aligned} \tag{13}$$

Lastly, the velocity on the interface is given by

$$V = \mathbf{u}\cdot\mathbf{n} = -\mu\nabla P\cdot\mathbf{n}. \tag{14}$$

Because the velocity depends on the gradient of P only, P_∞ does not affect the evolution of the interface. Therefore, we will typically set $P_\infty = 0$ for simplicity.

2.2 Nondimensionalization

We begin by noting that the pressure system (12) reveals an intrinsic length scale

$$L_D = \sqrt{\frac{D}{\overline{\lambda}_B + \lambda}}, \tag{15}$$

where $\overline{\lambda}_B$ is a characteristic value of λ_B. If $\overline{\lambda}_B = 0$, then L_D estimates the stable size of an avascular tumor [10]. We also have an intrinsic relaxation time scale λ_R^{-1} given by

$$\lambda_R = \frac{\mu\gamma}{L_D^3}. \tag{16}$$

Using these scales, we introduce the dimensionless variables

$$\widehat{\mathrm{x}} = \frac{\mathbf{x}}{L}, \quad \widehat{t} = \lambda_R t \tag{17}$$

and use $\widehat{}$ to denote differentiation in the dimensionless variables.

Next, we nondimensionalize the nutrient concentration by

$$c = \frac{\sigma}{\sigma^{\infty}}, \tag{18}$$

where c is the nondimensional concentration field, and

$$\begin{aligned} 0 &= \widehat{\nabla}^2 c + B\left(\widehat{\mathbf{x}}\right) - \begin{cases} \left(\frac{1+\frac{\lambda_B}{\lambda}}{1+\frac{\overline{\lambda}_B}{\lambda}}\right) c & \text{in } \Omega \\ \left(\frac{\frac{\lambda_B}{\lambda}}{1+\frac{\overline{\lambda}_B}{\lambda}}\right) c & \text{else,} \end{cases} \\ c\Big|_{\partial\Omega_D} &= 1, \end{aligned} \tag{19}$$

where B is the dimensionless function given by

$$B\left(\widehat{\mathbf{x}}\right) = \frac{\sigma_B}{\sigma^{\infty}} \frac{\lambda_B\left(\widehat{\mathbf{x}}\right)}{\overline{\lambda}_B + \lambda}. \tag{20}$$

We also introduce the nondimensional parameter

$$N = \frac{\sigma_N}{\sigma^{\infty}}, \tag{21}$$

with which we define Ω_N to be the region where $c < N$.

In the case where $\lambda_B \equiv \overline{\lambda}_B$ and $N = 0$, (20) simplifies to equation (55) in [10], the dimensionless concentration equation in Ω becomes

$$\begin{aligned} 0 &= \widehat{\nabla}^2 c + B - c \text{ in } \Omega \\ c\Big|_{\partial\Omega_D} &= 1, \end{aligned} \tag{22}$$

and we can obtain the formulation for the dimensionless concentration $\overline{\Gamma}$ given in [10] through the relation

$$c = \overline{\Gamma}(1 - B) + B. \tag{23}$$

We nondimensionalize the pressure by

$$P = \frac{\gamma}{L_D}\widehat{p}. \tag{24}$$

If we define the dimensionless parameters

$$\mathcal{A} = \frac{\lambda_A}{\lambda_M}, \quad \mathcal{G} = \frac{\lambda_M}{\lambda_R}, \text{ and } \quad G_N = \frac{\lambda_N}{\lambda_M}, \tag{25}$$

then the nondimensionalized pressure is given by

$$\widehat{\nabla}^2 \widehat{p} = \begin{cases} -\mathcal{G}\,(c - \mathcal{A}) & \text{in } \Omega_P \\ \mathcal{G} G_N & \text{in } \Omega_N, \end{cases} \tag{26}$$

$$\widehat{p} \equiv \frac{L_D P_\infty}{\gamma} \text{ in } \Omega_D - \overline{\Omega}, \tag{27}$$

and the boundary condition is given by

$$\widehat{p}\Big|_\Sigma = \frac{L_D P_\infty}{\gamma} + \widehat{\kappa}. \tag{28}$$

Notice that if $B \equiv 0$, then $\mathcal{A}$ and $\mathcal{G}$ coincide with A and G in [10]. If $\lambda_B \equiv \overline{\lambda}_B$, $N = 0$, $P_\infty = 0$, and if we define A and G as in [10], then we can obtain the nondimensional pressure $\overline{p}$ in [10] through the relation

$$\widehat{p} = \overline{p} + (1 - \overline{\Gamma})G + AG\frac{\widehat{\mathbf{x}} \cdot \widehat{\mathbf{x}}}{2d}. \tag{29}$$

Lastly, the nondimensional velocity is given by

$$\widehat{V} = -\widehat{\mathbf{n}} \cdot \widehat{\nabla}\widehat{p}. \tag{30}$$

In the remainder of the text, we shall drop the $\widehat{}$ notation for simplicity.

2.3 Special Case: Avascular Tumor Growth

In this paper, we shall focus on avascular (or pre-vascular) tumor growth In this case $\lambda_B \equiv 0$, $B \equiv 0$, $\mathcal{A} = A$, and $\mathcal{G} = G$ as in [10]. (We shall henceforth use A and G rather than $\mathcal{A}$ and $\mathcal{G}$, as they coincide in this case.)

As before, we denote the proliferating region of the tumor by Ω_P, the necrotic region by Ω_N, and the computational domain by Ω_D. By Σ we denote the boundary of the tumor, and Σ_N marks the tumor-necrotic interface as earlier. The tumor can pinch off and capture regions of the outer domain [10]; we denote the union of all such captured regions by Ω_C. Note that $\Omega = \Omega_P \cup \Omega_N \cup \Sigma_N$. The interface between the tumor cells and the outer region is Σ. Where necessary, we differentiate between Σ_i (the interface between the tumor and captured regions) and Σ_o (the interface between the tumor and the outer portion of the computational domain). See Figure 2.

For simplicity, we shall set $c = 1$ in the outer domain (including Ω_C) and on Σ; our resulting system is

$$\begin{cases} \nabla^2 c = c \text{ in } \Omega \\ c|_\Sigma = 1. \end{cases} \tag{31}$$

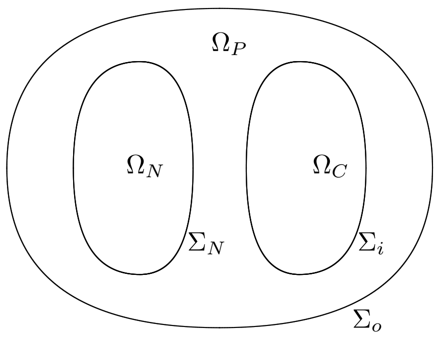

Figure 2: Tumor with a captured outer region and necrotic core.

The pressure is modeled as before: we set $p = P_\infty = 0$ in the outer domain and Ω_C, and we apply the surface tension boundary condition to both Σ_o and Σ_i:

$$\begin{cases} \nabla^2 p &= \begin{cases} -G(c-A) & \text{in } \Omega_P \\ GG_N & \text{in } \Omega_N \end{cases} \\ p\big|_\Sigma &= \kappa \\ p &= 0 \text{ in } \Omega_C. \end{cases} \tag{32}$$

As before, the velocity is given by

$$V\big|_\Sigma = -\mathbf{n} \cdot \nabla p. \tag{33}$$

Notice that (31) and (32) are defined on Ω. We shall refer to these as interior Poisson problems.

2.4 Modeling Chemotherapy

Recall that the parameter A is related to the relative strength of the apoptosis and mitosis rates of the tumor cells:

$$A = \frac{\lambda_A}{\lambda_M}.$$

Within the context of the model, cancer treatments can be considered as external processes that increase the apoptosis rate λ_A of the tumor cells. Consequently, we model an active therapy session by increasing the value of A.

By *chemotherapy session* or *treatment session*, we mean the administration of a chemical species (with uniform concentration throughout the domain) that temporarily increases the ratio of apoptosis to mitosis in the tumor cells. We model the presence of the chemical species as an elevated value of A above a baseline level A_0.

We characterize a chemotherapy session as a triple (t_1, t_2, A), where $A > A_0$ and $0 \leq t_1 < t_2$. We model the *treatment regimen* by

$$A(t) = \begin{cases} A_n & \text{if } t_n \leq t < t_{n+1} \text{ for some } n \\ A_0 & \text{else.} \end{cases} \tag{34}$$

In [11], it was shown numerically and verified experimentally that necrosis is the dominant form of tumor cell death in the avascular regime because necrosis and cell proliferation occur on a faster time scale than apoptosis. Therefore, we shall set N and G_N nonzero and A_0 small when modeling chemotherapy.

In this paper, A_n is constant for each n. In future work, we will model the dissipation of the chemical species over time (which would yield a decay in A_n throughout each treatment session). We will also model the development of *resistance*: the surviving tumor cells adapt to the chemical species over time, leading to an additional decay of A_n during each treatment session and reducing the effective initial level of each A_n.

3 Level Set Methods for Evolving Interfaces

The most traditional method used to track an evolving interface is to do so explicitly with a "marker" technique: the initial interface is specified, and markers are placed at predetermined intervals and sorted. As time progresses, each marker's position is updated according to the velocity field. While this method is straightforward to implement, there are several drawbacks to the approach.

As the interface evolves, the markers tend to lose their uniform spacing, leading some regions of the interface to be poorly resolved. This problem is solved by periodically redistributing the markers. Frequently, the interface's total arclength changes over time, which means that the interface will no longer be resolved at the original level. This can be overcome by adding and subtracting markers and reordering them when redistributing the markers.

A larger problem occurs when the morphology of the interface changes (e.g., the interface splits or reconnects). In its most basic form, the marker method fails in such a case because it assumes the interface is a simple closed curve. This problem can be solved by using a linked list data structure for the various interface segments, but such an approach can be difficult and costly to implement, particularly in three dimensions and in automating the detection of morphological changes in the interface.

In our particular application, we anticipate frequent morphological changes in the interface. For this reason, we use the level set method developed by Sethian and Osher and described in [26] to implicitly track the boundary.

3.1 Basic Properties

To implement the level set method, we introduce a level set function φ and require that it satisfy several properties:

1. If Σ denotes the interface, then

$$\Sigma = \{\mathbf{x} \in \mathbb{R}^n : \varphi(\mathbf{x}) = 0\}.$$

   The zero level set of φ coincides with the interface.

2. $\varphi(\mathbf{x}) < 0$ for points inside the tumor boundary and $\varphi(\mathbf{x}) > 0$ for points outside the boundary.

See Figure 3 for an illustration of this concept.

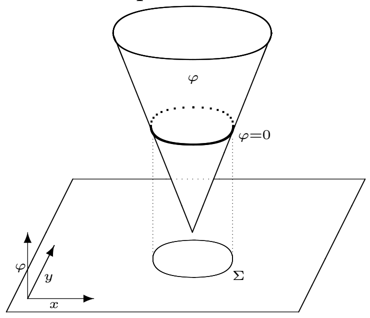

Figure 3: Concept of the level set function: representing a circle with a level set function.

In addition, we stipulate that φ is a signed distance function:

3. $|\varphi(\mathbf{x})| = d\,(\mathbf{x}, \Sigma)$.

An immediate advantage of this formulation is that geometric quantities are readily calculated based upon the level set function. We find the normal vector (oriented in the outer direction) by

$$\mathbf{n} = \frac{\nabla\varphi}{|\nabla\varphi|}.$$

Note that $|\nabla\varphi| = 1$ because φ is a signed distance function. However, we retain the denominator because numerical error may perturb $|\nabla\varphi|$ away from 1. The curvature is also readily calculated from the level set function. In two dimensions,

$$\begin{aligned} \kappa &= \nabla\cdot\mathbf{n} \\ &= \nabla\cdot\frac{\nabla\varphi}{|\nabla\varphi|} \\ &= \frac{\varphi_{xx}\varphi_y^2 - 2\varphi_y\varphi_x\varphi_{xy} + \varphi_{yy}\varphi_x^2}{\left(\varphi_x^2+\varphi_y^2\right)^{\frac{3}{2}}}, \end{aligned} \tag{35}$$

and in three dimensions, the mean curvature is

$$\begin{aligned} \kappa_M &= \nabla\cdot\mathbf{n} \\ &= \nabla\cdot\frac{\nabla\varphi}{|\nabla\varphi|} \\ &= \frac{\left\{\begin{array}{c} \left(\varphi_{yy}+\varphi_{zz}\right)\varphi_x^2 + \left(\varphi_{xx}+\varphi_{zz}\right)\varphi_y^2 + \left(\varphi_{xx}+\varphi_{yy}\right)\varphi_z^2 \\ -2\left[\varphi_x\varphi_y\varphi_{xy} + \varphi_x\varphi_z\varphi_{xz} + \varphi_y\varphi_z\varphi_{yz}\right] \end{array}\right\}}{\left(\varphi_x^2+\varphi_y^2+\varphi_z^2\right)^{\frac{3}{2}}}, \end{aligned} \tag{36}$$

and the Gaussian curvature is given by

$$\kappa_G = \frac{\left\{\begin{array}{c} \varphi_x^2\left(\varphi_{yy}\varphi_{zz}-\varphi_{yz}^2\right) + \varphi_y^2\left(\varphi_{xx}\varphi_{zz}-\varphi_{xz}^2\right) \\ +\varphi_z^2\left(\varphi_{xx}\varphi_{yy}-\varphi_{xy}^2\right) + 2\left[\varphi_x\varphi_y\left(\varphi_{xz}\varphi_{yz}-\varphi_{xy}\varphi_{zz}\right)\right] \\ +2\left[\varphi_y\varphi_z\left(\varphi_{xy}\varphi_{xz}-\varphi_{yz}\varphi_{xx}\right) + \varphi_x\varphi_z\left(\varphi_{xy}\varphi_{yz}-\varphi_{xz}\varphi_{yy}\right)\right] \end{array}\right\}}{\left(\varphi_x^2+\varphi_y^2+\varphi_z^2\right)^2}. \tag{37}$$

If $\mathbf{u}$ denotes the velocity field, we evolve the interface according to

$$\partial_t\varphi + \mathbf{u}\cdot\nabla\varphi = \partial_t\varphi + V(\mathbf{u})\,|\nabla\varphi| = 0,$$

where

$$V(\mathbf{u}) = \mathbf{u}\cdot\frac{\nabla\varphi}{|\nabla\varphi|} = \mathbf{u}\cdot\mathbf{n}$$

is the normal velocity of the interface. For the remaining discussion, we shall restrict our discussion to two dimensions; the three-dimensional case is completely analagous.

3.2 Initializing the Level Set Function

Suppose we are given a two-dimensional interface Σ. We describe here how to generate the signed distance function to Σ on a discrete lattice $\{\varphi_{i,j}\}$ to be used in numerical simulation. To initialize the lattice, we first set

$$\varphi_{i,j} = \begin{cases} -1 & \text{if } \left(x_i, y_j\right) \text{ is inside the interface} \\ 0 & \text{if } \left(x_i, y_j\right) \text{ is on the interface} \\ 1 & \text{if } \left(x_i, y_j\right) \text{ is outside the interface.} \end{cases}$$

Next, for each point (x_i, y_j), we search $\{\varphi_{i,j}\}$ for the closest point (x_ℓ, y_m) such that

$$\operatorname{sign}(\varphi_{i,j}) \neq \operatorname{sign}(\varphi_{\ell,m})$$

and reassign

$$\varphi_{i,j} = \operatorname{sign}(\varphi_{i,j}) \sqrt{(x_i - x_\ell)^2 + (y_j - y_m)^2}.$$

If the interface is already specified as a level set function, then we can skip the preceding steps.

We then solve the following partial differential equation to steady-state:

$$\varphi_\tau = \operatorname{sign}\left(\varphi^0\right)(1 - |\nabla\varphi|), \tag{38}$$

where φ_0 is the original level set function and τ is pseudo-time. Notice that when $\varphi_\tau = 0$, either $|\nabla\varphi| = 1$ or $\operatorname{sign}(\varphi^0) = 0$, so the steady-state solution is a distance function ($|\nabla\varphi| = 1$) with the same zero level set as φ_0 ($\operatorname{sign}\left(\varphi^0\right) = 0$). If we only require the level set function within a fixed distance R of the interface, then we solve (38) until $\tau = R$ [27]. Equation (38) is also used to reinitialize the level set function between iterations when evolving the entire pressure-concentration-interface system.

3.3 Narrow Band Level Set Technique

Our primary goal is to sucessfully calculate the location of the interface Σ over time. Therefore, we seek to update φ only as much as is necessary to accurately advect the interface. This can be done by only updating φ within a distance R of the interface. Given an initialized level set function φ, the points which fall within that distance are

$$\left\{\mathbf{x} : |\varphi(\mathbf{x})| \leq R + \epsilon\right\} \tag{39}$$

for some small ϵ. This set is referred to as a "narrow band" about φ, R is the width of the band, and the technique is known as the "narrow band" level set method. The value of R is determined by the numerical implementation. We shall revisit this point in a later section. Please see [26, 21] for further details on the method.

4 Numerical Solution: General Technique

We employ finite differences to solve the concentration-pressure-interface system. We begin by enclosing the interface within a larger, rectangular computational domain $[a,b] \times [c,d]$. (We will postpone our discussion of how large $[a,b] \times [c,d]$ must be for a later part of this paper.) Next, we discretize the computational domain using regular step sizes Δx and Δy to obtain $\{(x_i, y_j) : 1 \leq i \leq M, 1 \leq j \leq N\}$, where M and N denote the number of node points in the x- and y-directions, respectively. For convenience, we will generally take $\Delta x = \Delta y$. We then proceed according to the following procedure:

4.1 Initial Steps

1. Introduce and initialize a level set function φ to match the interface Σ as discussed earlier. Introduce all necessary parameters.

2. Determine whether there are sufficiently many node points between Σ and the computational boundary. If not, extend x and y and reinitialize φ.

4.2 Main Loop

Once we have initialized the code, we enter the main control loop of the algorithm. We advance in time and do the following for each fixed time step:

1. Check for proximity of the interface Σ to the computational boundary. If there is insufficient space between Σ and $\partial([a,b] \times [c,d])$, then extend x and y and reinitialize φ. Update A according to the chemotherapy choice.

2. Reinitialize the level set function φ, and calculate the normal vector $\mathbf{n}$ and the curvature κ where required.

3. Solve for the concentration c and pressure p (in that order).

4. Calculate the gradient ∇p inside and on Σ, and extend the components of ∇p beyond Σ.

5. Calculate the normal velocity V in a band about Σ according to (33). Extend the normal velocity V normally from the interface Σ, and filter high-frequency numerical noise from the extended speed.

6. Update φ according to the normal velocity V.

7. Repeat (1)-(6) for each step of the time discretization.

5 Discretization of the Operators

Having laid out the general numerical scheme, we now detail the discretization of the operators.

5.1 Notation

Given a sufficiently differentiable function $f(x, y, t)$ and a discretization $\{x_i, y_j\}$ of (x, y), we denote $f(x_i, y_j, t)$ by $f_{i,j}(t)$. If f is a function of fewer or more variables, we adjust the indexing accordingly. We define the following finite differential and finite difference operators for a function $f(x)$:

$$\begin{aligned}
\Delta^+ f &= f_{i+1} - f_i \\
\Delta^- f &= f_i - f_{i-1} \\
\Delta^0 f &= f_{i+1} - f_{i-1} \\
D^+ f &= \frac{\Delta^+ f}{\Delta x} = f'(x_i) + \mathcal{O}(\Delta x) \\
D^- f &= \frac{\Delta^- f}{\Delta x} = f'(x_i) + \mathcal{O}(\Delta x) \\
D^0 f &= \frac{\Delta^0 f}{2\Delta x} = f'(x_i) + \mathcal{O}\left(\Delta x^2\right). \qquad (40)
\end{aligned}$$

These are all obtained by expanding f in a Taylor series about x_i and evaluating at $x = x_{i\pm1}$.

When f is a function of multiple variables, we shall specify the intended finite difference (differential) with an appropriate variable subscript. (e.g., $D_x^+ f$.) If only one index varies, we drop the fixed indicies for simplicity of notation.

The operators can be applied right-to-left to obtain higher-order differences. For example,

$$\begin{aligned}
D_x^{++} f &= D_x^+\left(D_x^+ f\right) \\
&= D_x^+\left(\frac{f_{i+1} - f_i}{\Delta x}\right) \\
&= \frac{1}{\Delta x}\left(\frac{f_{i+2} - f_{i+1}}{\Delta x} - \frac{f_{i+1} - f_i}{\Delta x}\right) \\
&= \frac{f_{i+2} - 2f_{i+1} + f_i}{\Delta x^2} = f''(x_{i+1}) + \mathcal{O}\left(\Delta x^2\right).
\end{aligned}$$

5.2 Temporal Discretizations

For this problem, we will use total variation diminishing Runge-Kutta (TVD-RK) schemes to approximate the temporal derivatives. These schemes yield highly accurate solutions without introducing artificial oscillations and instabilities in the time stepping [16, 14, 15].

Suppose we are given a partial differential equation of the form

$$f_t = L(f, t),$$

where L is a spatial operator. Then the third-order TVD-RK scheme is given by [14, 15]

$$\begin{aligned}
f^{(0)} &= f^k \\
f^{(1)} &= f^{(0)} + \Delta t L\left(f^{(0)}, t^k\right) \\
f^{(2)} &= f^{(0)} + \frac{1}{4}\Delta t\Big[L\left(f^{(0)}, t^k\right) + L\left(f^{(1)}, t^k + \Delta t\right)\Big] \\
f^{(3)} &= f^{(0)} + \frac{1}{6}\Delta t\left[L\left(f^{(0)}, t^k\right) + L\left(f^{(1)}, t^k + \Delta t\right) + 4L\left(f^{(2)}, t^k + \frac{1}{2}\Delta t\right)\right] \\
f^{k+1} &= f^{(3)}.
\end{aligned} \tag{41}$$

In our problem, $L(\varphi, t)$ will be of the form

$$L(\varphi, t) = V\,|\nabla\varphi| \text{ or } L(\varphi, t) = \mathbf{V}\cdot\nabla\varphi. \tag{42}$$

In these cases we shall use the CFL condition

$$\Delta t \leq \frac{1}{2\max\left(|V|\right)}\Delta x$$

to choose Δt to ensure convergence and numerical stability.

5.3 Spatial Discretizations

5.3.1 WENO Discretizations

Chakravarthy, Engquist, Harten, Osher and Shu developed the now-classic essentially non-oscillatory (ENO) schemes to solve hyperbolic problems while attaining high-order accuracy, avoiding undue oscillations and instabilities near shocks, and maintaining sharper discontinuities where desired [19, 26]. Chan, Liu, and Osher built upon the ideas of the ENO schemes when they devised the first weighted ENO (WENO) schemes. Here, the core idea is to use a linear combination of the candidate stencils of the ENO schemes to obtain a higher-order method when the solution is smooth and to emulate the ENO schemes near shocks and discontinuities [19, 18]. This is acheived by calculating weights that converge to the ENO schemes near shocks and otherwise converge to the highest-order stencil available for a given number of candidate stencils. In particular, we are interested in the third-order and fifth-order WENO methods. We use the formulation found in [18], which we give below for reference.

Third-Order WENO Method: For ease of notation, let us define

$$\begin{aligned}(u)^+ &= \max(u,0)\\ (u)^- &= \min(u,0).\end{aligned}$$

To obtain the third-order WENO scheme, we first calculate

$$\begin{aligned}\varphi_x^- &= \frac{1}{2}\left(\frac{\Delta_x^+\varphi_{i-1}}{\Delta x}+\frac{\Delta_x^+\varphi_i}{\Delta x}\right)-\frac{\omega_{-,x}}{2}\left(\frac{\Delta_x^+\varphi_{i-2}}{\Delta x}-2\frac{\Delta_x^+\varphi_{i-1}}{\Delta x}+\frac{\Delta_x^+\varphi_i}{\Delta x}\right)\\ \varphi_x^+ &= \frac{1}{2}\left(\frac{\Delta_x^+\varphi_{i-1}}{\Delta x}+\frac{\Delta_x^+\varphi_i}{\Delta x}\right)-\frac{\omega_{+,x}}{2}\left(\frac{\Delta_x^+\varphi_{i+1}}{\Delta x}-2\frac{\Delta_x^+\varphi_i}{\Delta x}+\frac{\Delta_x^+\varphi_{i-1}}{\Delta x}\right),\end{aligned} \tag{43}$$

where

$$\begin{aligned}r_{-,x} &= \frac{\epsilon+\left(\Delta_x^-\Delta_x^+\varphi_{i-1}\right)^2}{\epsilon+\left(\Delta_x^-\Delta_x^+\varphi_i\right)^2}\\ \omega_{-,x} &= \frac{1}{1+2r_{-,x}^2}\\ r_{+,x} &= \frac{\epsilon+\left(\Delta_x^-\Delta_x^+\varphi_{i+1}\right)^2}{\epsilon+\left(\Delta_x^-\Delta_x^+\varphi_x\right)^2}\\ \omega_{+,x} &= \frac{1}{1+2r_{+,x}^2}.\end{aligned} \tag{44}$$

We calculate $\varphi_y^\pm$, $r_{\pm,y}$, and $\omega_{\pm,y}$ similarly. Then, we approximate $V\left|\nabla\varphi\right|$ by

$$\begin{aligned}\left|\nabla^+\varphi\right| &= \sqrt{\left(\left(\varphi_x^-\right)^+\right)^2+\left(\left(\varphi_x^+\right)^-\right)^2+\left(\left(\varphi_y^-\right)^+\right)^2+\left(\left(\varphi_y^+\right)^-\right)^2}\\ \left|\nabla^-\varphi\right| &= \sqrt{\left(\left(\varphi_x^-\right)^-\right)^2+\left(\left(\varphi_x^+\right)^+\right)^2+\left(\left(\varphi_y^-\right)^-\right)^2+\left(\left(\varphi_y^+\right)^+\right)^2}\\ V\left|\nabla\varphi\right| &\approx (V)^+\left|\nabla^+\varphi\right|+(V)^-\left|\nabla^-\varphi\right|.\end{aligned} \tag{45}$$

If we approximate $(u,v)\cdot\nabla\varphi$ instead, we get

$$\begin{aligned}(u,v)\cdot\nabla\varphi &\approx (u)^+\varphi_x^-+(u)^-\varphi_x^+\\ &\quad+(v)^+\varphi_y^-+(v)^-\varphi_y^+.\end{aligned} \tag{46}$$

Fifth-Order WENO Method: The fifth-order WENO method cannot be stated as simply as the third-order method. For a given point (x_i, y_j), we define

$$\begin{aligned}\varphi_{x,i}^- &= \frac{1}{12}\left(-\frac{\Delta^+\varphi_{i-2}}{\Delta x}+7\frac{\Delta_x^+\varphi_{i-1}}{\Delta x}+7\frac{\Delta_x^+\varphi_i}{\Delta x}-\frac{\Delta_x^+\varphi_{x+1}}{\Delta x}\right)\\ &\quad-\Phi^{\text{WENO}}\left(\frac{\Delta_x^-\Delta_x^+\varphi_{i-2}}{\Delta x},\frac{\Delta_x^-\Delta_x^+\varphi_{i-1}}{\Delta x},\frac{\Delta_x^-\Delta_x^+\varphi_i}{\Delta x},\frac{\Delta_x^-\Delta_x^+\varphi_{i+1}}{\Delta x}\right)\end{aligned} \tag{47}$$

and

$$\begin{aligned}\varphi_{x,i}^+ &= \frac{1}{12}\left(-\frac{\Delta^+\varphi_{i-2}}{\Delta x}+7\frac{\Delta_x^+\varphi_{i-1}}{\Delta x}+7\frac{\Delta_x^+\varphi_i}{\Delta x}-\frac{\Delta_x^+\varphi_{x+1}}{\Delta x}\right)\\ &\quad+\Phi^{\text{WENO}}\left(\frac{\Delta_x^-\Delta_x^+\varphi_{i+2}}{\Delta x},\frac{\Delta_x^-\Delta_x^+\varphi_{i+1}}{\Delta x},\frac{\Delta_x^-\Delta_x^+\varphi_i}{\Delta x},\frac{\Delta_x^-\Delta_x^+\varphi_{i-1}}{\Delta x}\right),\end{aligned} \tag{48}$$

where

$$\Phi^{\text{WENO}}(a,b,c,d) = \frac{1}{3}(a-2b+c)\omega_0 + \frac{1}{6}(b-2c+d)\left(\omega_2 - \frac{1}{2}\right). \tag{49}$$

In (49), we have

$$\begin{aligned}
IS_0 &= 13(a-b)^2 + 3(a-3b)^2,\\
IS_1 &= 13(b-c)^2 + 3(b+c)^2,\\
IS_2 &= 13(c-d)^2 + 3(3c-d)^2,\\
\alpha_0 &= \frac{1}{(\epsilon + IS_0)^2},\\
\alpha_1 &= \frac{6}{(\epsilon + IS_1)^2},\\
\alpha_2 &= \frac{3}{(\epsilon + IS_2)^2},\\
\omega_0 &= \frac{\alpha_0}{\alpha_0+\alpha_1+\alpha_2}, \quad \text{and}\\
\omega_2 &= \frac{\alpha_2}{\alpha_0+\alpha_1+\alpha_2}.
\end{aligned} \tag{50}$$

We calculate $\varphi^{\pm}_{y,j}$ similarly, and we approximate $V\,|\nabla\varphi|$ as in (45). The quantity $(u,v)\cdot\nabla\varphi$ is obtained as in (46). We shall use the fifth-order WENO method in all our discretizations of $|\nabla\varphi|$ and $\mathbf{u}\cdot\nabla\varphi$.

5.4 The Sign Function

A common discretization of the Heaviside and sign functions [22] is given by

$$\begin{aligned}
H_\delta(\varphi) &= \begin{cases} 0 & \text{if } \varphi < -\delta \\ \frac{1}{2}\left(1 + \frac{\varphi}{2\delta} + \frac{1}{\pi}\sin\left(\frac{\pi\varphi}{\delta}\right)\right) & \text{if } |\varphi| \le \delta \\ 1 & \text{if } \varphi > \delta, \end{cases}\\
\text{sign}_\delta(\varphi) &= 2\left(H_\delta(\varphi) - \frac{1}{2}\right),
\end{aligned} \tag{51}$$

where δ is a small positive number. In all our calculations, we use $\delta = \Delta x$.

5.5 Application to the Level Set Function

Recall that we advect the interface Σ by solving

$$\varphi_t + V\,|\nabla\varphi| = 0, \tag{52}$$

where V is the normal velocity defined in (33). In our numerical implementation, we discretize $V\,|\nabla\varphi|$ with the fifth-order WENO method. We approximate the temporal derivative with the third-order TVD-RK method, where

$$L(\varphi) = -V\left|\nabla\varphi\right|. \tag{53}$$

Note that we must solve for the concentration and pressure at each intermediate step of the TVD-RK discretization in order to obtain V.

We reinitialize the level set function by solving

$$\varphi_\tau - \operatorname{sign}(\varphi^0)\left(1 - \left|\nabla\varphi\right|\right) = 0. \tag{54}$$

We discretize $\operatorname{sign}(\varphi^0)\left|\nabla\varphi\right|$ with the fifth-order WENO method and approximate the pseudo-temporal derivative (in τ) by the third-order TVD-RK method, where

$$L(\varphi) = \operatorname{sign}(\varphi^0) - \operatorname{sign}(\varphi^0)\left|\nabla\varphi\right|. \tag{55}$$

6 Poisson Solver

6.1 Introduction

Our solution technique for the pressure and concentration is an extension of the ghost fluid method in [4] to higher-order accuracy with modifications and simplifications inspired by [13]. In this method, we solve the interior problem in a complex domain

$$\begin{cases} \nabla^2 u = f\left(u, \mathbf{x}\right) & \text{in } \Omega \\ u = g\left(\mathbf{x}\right) & \text{on } \Sigma, \end{cases} \tag{56}$$

by embedding the problem in a rectangular domain $[a,b] \times [c,d]$ and extending u as a constant γ into $\Omega_o = [a,b] \times [c,d] - \overline{\Omega}$, which we refer to as the outer domain. Thus, we solve the system

$$\begin{cases} \nabla^2 u = f\left(u, \mathbf{x}\right) & \text{in } \Omega \\ u = g\left(\mathbf{x}\right) & \text{on } \Sigma \\ u = \gamma & \text{in } [a,b] \times [c,d] - \overline{\Omega}. \end{cases} \tag{57}$$

We shall discretize (57) on the entire domain and include the rows with trivial solutions to preserve row (or column) ordering of the resulting coefficient matrix.

The solution u is assumed to be smooth within Ω up to Σ. For simplicity, we shall refer to Ω as the inner domain and Ω_o as the outer domain. 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.

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

$$u_{xx} = \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} + \mathcal{O}\left(\Delta x^2\right). \tag{58}$$

However, if the interface intersects $[x_{i-1}, x_{i+1}]$, then u is potentially discontinuous and the finite difference in (58) is invalid in its current form. Supposing that the interface occurs between x_i and x_{i+1}, we seek to replace u_{i+1} in (58) with $\widehat{u}_{i+1}$, a smooth extension of u in the inner domain to x_{i+1}.

In the ghost fluid method, this is done by extrapolating $\widehat{u}_{i+1}$ and any coefficient data from the values at x_{i-1} and x_i and relationships between the inner and outer domains (via equations of state and boundary conditions). These extrapolated values are then explicitly stored as "ghost fluid" points. In [13], Fedkiw et al. avoided this explicit storage by relating u_i and u_{i+1} solely through jump boundary conditions. After this relationship was established, the terms in the discretization were rearranged, leaving the standard centered difference stencil on the left-hand side and all other terms in the right-hand side. In essence, the method works by introducing an additional source / sink function to the equation that acts similarly to a delta function in satisfying the jump boundary conditions.

Our technique combines and extends these two approaches. We obtain $\widehat{u}_{i+1}$ by cubic interpolation of u_{i-3}, u_{i-2}, u_{i-1}, and the boundary data, obtaining both a new computational stencil and right-hand side. We choose the boundary value for the interpolation by precisely locating the interface between x_i and x_{i+1} (by cubic interpolation of φ), interpolating the boundary data function, and evaluating that interpolation at the interface. When u_{i-3} or u_{i-2} is not in the inner domain, we modify our approach to use lower-order interpolation.

Our approach attains many of the desirable features in [13]. The resulting solution automatically satisfies the boundary condition at the estimated location of the interface, rather than on nearby computational nodes; this is known as *subcell resolution.* The method is robust and straightforward to implement, even for complex geometries. We can avoid the complications of solving a problem on an irregular grid by rather solving a diffusion problem on the entire rectangular grid. As in [13], our method can be applied dimension-by-dimension, although a small consideration needs to be made for the interaction of the spatial dimensions in one case.

Our approach differs from the boundary condition capturing (BCC) method in [13] in several respects. The BCC method was designed to solve a variable-coefficient Poisson problem on a rectangular domain with internal jump conditions in the solution and normal derivative posed on a complex internal boundary. The method determines the location of the interface and approximates the boundary data with linear interpolation, although it could be extended to higher-order interpolation; in [7] and [8], it is stated that higher-order interpolation is unnecessary for second-order convergence when the position of the interface is known precisely. While the BCC method could be modified and applied to our problem, we found it yielded sub-first-order convergence when applied to the full moving boundary problem with geometric boundary conditions. In [7], it was also stated that the BCC method only gives first-order accuracy of the interface when applied to a moving boundary problem.

Our method was specifically designed to solve the interior Poisson problem with Dirichlet conditions on a complex domain. Our technique uses high-order (quadratic and cubic) interpolation of the level set function and the boundary data for higher-accuracy subcell resolution and can be readily extended to higher accuracy. Because the BCC method was designed for problems with jump boundary conditions, it classifies points as belonging to either the inner or outer domain. Our technique, designed for the Dirichlet problem, classifies points as lying in the inner or outer domain or on the boundary, allowing us to use precisely-interpolated, known values of the boundary data in place of u in the stencil to improve accuracy. Lastly, our technique attains second-order accuracy in a wide variety of cases, including those with geometric boundary conditions (e.g., curvature). We note that while our method was designed for the Poisson problem $\nabla \cdot (\beta \nabla u)$ with $\beta \equiv 1$, it could be extended to the variable β case.

6.2 Classification of Node Points

The Poisson solver solves (56) by solving (57) on the full rectangular domain. 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 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:

$$\varphi_{i,j} \begin{cases} < 0 & \text{if } x_{i,j} \in \Omega \\ = 0 & \text{if } x_{i,j} \in \Sigma \\ > 0 & \text{if } x_{i,j} \in \Omega_o. \end{cases}$$

However, the classification must be modified to take machine floating-point arithmetic into account. We define *machine zero* ϵ_{mach} by

$$\epsilon_{\text{mach}} = \max\{\epsilon > 0 : 1.0 + \epsilon = 1.0\} \tag{59}$$

in machine floating-point arithmetic. Note that because computer hardware can only represent finitely-many floating-point numbers, this set has a unique maximum. On most modern, 32-bit machines, this number is typically $2^{-53} \approx 1.11\,\text{e-}16$. Any element of $[-\epsilon_{\text{mach}}, \epsilon_{\text{mach}}]$ should be interpreted as zero, and conversely, zero could be represented as any floating-point number in that range. Thus, in machine floating-point arithmetic, "zero" should be considered as an equivalence class of the floating-point numbers in the range $[-\epsilon_{\text{mach}}, \epsilon_{\text{mach}}]$.

Accordingly, we modify our classification to

$$\varphi_{i,j} \begin{cases} < -\epsilon & \text{if } x_{i,j} \in \Omega \\ \in [-\epsilon, \epsilon] & \text{if } x_{i,j} \in \Sigma \\ > \epsilon & \text{if } x_{i,j} \in \Omega_o, \end{cases} \tag{60}$$

where $\epsilon \geq \epsilon_{\text{mach}}$. In our calculations, we generally use $\epsilon = 2\epsilon_{\text{mach}}$.

6.3 Discretizing the Equation

We discretize (57) 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 cofficient matrix, which makes it a banded matrix that can be stored more efficiently in memory.

The discretization on Σ and in Ω_o is trivial:

1. **Case: $\varphi_i > \epsilon$:**

   By (60), $x_i \in \Omega_0$, so we set $u_i = \gamma$. To improve the conditioning number of the coefficient matrix, we shall use

$$\frac{-1}{\Delta x^2} u_i = \frac{-1}{\Delta x^2}\gamma. \tag{61}$$

2. **Case: $-\epsilon \leq \varphi_i \leq \epsilon$:**

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

   $$\frac{-1}{\Delta x^2} u_i = \frac{-1}{\Delta x^2} g(x_i). \tag{62}$$

   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: $\varphi_i < -\epsilon$:**

   As $x_i \in \Omega$, we proceed to build the stencil to approximate u_{xx}. The construction depends upon whether the neighboring node points are also in the inner domain.

   (a) **Case: $\varphi_{i-1} < -\epsilon$ and $\varphi_{i+1} < -\epsilon$:**

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

   $$\frac{1}{\Delta x^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) = f(u_i, x_i). \tag{63}$$

   (b) **Case: $\varphi_{i-1} < -\epsilon$ and $\varphi_{i+1} \geq -\epsilon$:**

   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

   $$x_\Sigma = x_i + \theta\,\Delta x, \qquad 0 < \theta \leq 1, \tag{64}$$

   where θ is determined by interpolating the level set function φ. The location of x_Σ is critical to the accuracy of the method, so we construct a cubic interpolation of φ through the points in the set $\mathcal{S} = \{x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$ and find the root between x_i and x_{i+1}. (If there are multiple roots in that interval, they are too close together to be resolved by the method. In this case, we use the root closest to that given by linear interpolation.) This provides us with the subcell resolution introduced earlier.

   Next, we evaluate g at the points in $\mathcal{S}$, apply cubic interpolation to them, and evaluate the interpolation at x_Σ. Let us denote the value of the interpolation by g_Σ.

   Our solution technique involves extending u from the inner domain to x_{i+1} and obtaining a "ghost value" $\widehat{u}_{i+1}$. We determine $\widehat{u}_{i+1}$ by interpolating the neighboring values of

u contained in Ω. To maintain the second-order accuracy of the scheme, it is preferable that we use third-order (cubic) interpolation of u to determine u_Σ.

If $x_{i-3}, x_{i-2} \in \Omega$ as described in (60), then we use cubic interpolation with u_{i-3}, u_{i-2}, u_{i-1}, and u_Σ, where

$$u_\Sigma = u(x_\Sigma) = g(x_\Sigma) = g_\Sigma. \tag{65}$$

Having estimated u_Σ and $\widehat{u}_{i+1}$, we then approximate $u_{xx} = f$ by

$$\frac{1}{\Delta x^2}\left(u_{i-1} - 2u_i + \widehat{u}_{i+1}\right) = f(u_i, x_i). \tag{66}$$

If we use cubic interpolation of u, for instance, then $\widehat{u}_{i+1}$ is given by

$$\begin{aligned} \widehat{u}_{i+1} &= -3\left(\frac{1-\theta}{3+\theta}\right) u_{i-3} + 8\left(\frac{1-\theta}{2+\theta}\right) u_{i-2} \\ &\quad -6\left(\frac{1-\theta}{1+\theta}\right) u_{i-1} + \frac{24}{(1+\theta)(2+\theta)(3+\theta)}\, g_\Sigma, \end{aligned} \tag{67}$$

and the stencil becomes

$$\begin{aligned} \frac{1}{\Delta x^2}\Bigg\{ &-3\left(\frac{1-\theta}{3+\theta}\right) u_{i-3} + 8\left(\frac{1-\theta}{2+\theta}\right) u_{i-2} \\ &+ \left[1 - 6\left(\frac{1-\theta}{1+\theta}\right)\right] u_{i-1} - 2u_i \\ &+ \frac{24}{(1+\theta)(2+\theta)(3+\theta)}\, g_\Sigma \Bigg\} = f(x_i). \end{aligned} \tag{68}$$

(c) **Case: $\varphi_{i-1} \geq -\epsilon$ and $\varphi_{i+1} < -\epsilon$:**

In this case, the interface is located between x_{i-1} and x_i . Let us denote this location by x_Σ. We proceed as in the previous case, first defining $0 \leq \theta < 1$ by

$$x_\Sigma = x_{i-1} + \theta \Delta x, \tag{69}$$

and then extending the inner domain by interpolation to x_{i-1} to define $\widehat{u}_{i-1}$. If $x_{i+2}, x_{i+3} \in \Omega$, then we can use cubic interpolation as before, obtaining the discretization

$$\begin{aligned} \frac{1}{\Delta x^2}\Bigg\{ &3\left(\frac{\theta}{\theta-4}\right) u_{i+3} + 8\left(\frac{\theta}{3-\theta}\right) u_{i+2} \\ &+ \left[1 + 6\left(\frac{\theta}{\theta-2}\right)\right] u_{i+1} - 2u_i \\ &- \frac{24}{(\theta-2)(\theta-3)(\theta-4)}\, g_\Sigma \Bigg\} = f(x_i). \end{aligned} \tag{70}$$

(d) **Case: $\varphi_{i-1} \geq -\epsilon$ and $\varphi_{i+1} \geq -\epsilon$:**

This case requires more careful consideration as a series of subcases:

i. **Case: $|\varphi_{i-1}| \le \epsilon$ and $|\varphi_{i+1}| \le \epsilon$:**

In this case, $x_{i\pm1} \in \Sigma$, so we can construct an approximation to $\nabla^2 u = f$ by

$$\frac{1}{\Delta x^2}\Big(g(x_{i-1}) - 2u_i + g(x_{i+1})\Big) = f(u_i, x_i). \tag{71}$$

ii. **Case: $|\varphi_{i-1}| \le \epsilon$ and $\varphi_{i+1} > \epsilon$:**

In this case, $x_{i-1} \in \Sigma$ and the interface also occurs between x_i and x_{i+1} at

$$x_\Sigma = x_i + \theta \Delta x, \tag{72}$$

where θ is determined by interpolation of φ.

As with the case where $x_{i-1} \in \Omega$, we construct a stencil for $\nabla^2 u$ as

$$\begin{aligned} \nabla^2 u &\approx \frac{1}{\Delta x^2}\Big(u_{i-1} - 2u_i + \widehat{u}_{i+1}\Big) \\ &= \frac{1}{\Delta x^2}\Big(g(x_{i-1}) - 2u_i + \widehat{u}_{i+1}\Big), \end{aligned} \tag{73}$$

as $x_{i-1} \in \Sigma$. To determine $\widehat{u}_{i+1}$, we fit a linear curve through $u_\Sigma = g_\Sigma$ with slope determined by u_i and $u_{i-1} = g_{i-1}$:

$$\begin{aligned} \widehat{u}_{i+1} &= (1-\theta)\big(u_i - g_{i-1}\big) + u_\Sigma \\ &= (1-\theta)\big(u_i - g_{i-1}\big) + g_\Sigma, \end{aligned} \tag{74}$$

where as before $g_\Sigma = g(x_\Sigma)$. This leads to a stencil

$$\frac{1}{\Delta x^2}\Big(\theta g_{i-1} - (1+\theta)u_i + g_\Sigma\Big) = f(u_i, x_i). \tag{75}$$

iii. **Case: $\varphi_{i-1} > \epsilon$ and $|\varphi_{i+1}| \le \epsilon$:**

In this case, $x_{i+1} \in \Sigma$ and the interface also occurs between x_{i-1} and x_i at

$$x_\Sigma = x_{i-1} + \theta \Delta x, \tag{76}$$

where θ is determined by interpolation of φ.

As in the previous case, we construct a linear curve through $u_\Sigma = g_\Sigma$ with slope determined by u_i and $u_{i+1} = g_{i+1}$ to find $\widehat{u}_{i-1}$, obtaining the discretization

$$\frac{1}{\Delta x^2}\Big(g_\Sigma + (\theta - 2)u_i + (1-\theta)g_{i+1}\Big) = f(x_i, u_i). \tag{77}$$

iv. **Case: $\varphi_{i-1} > \epsilon$ and $\varphi_{i+1} > \epsilon$:**

The interface occurs on both the right- and left-hand sides of the stencil, and there is insufficient data to interpolate both $\widehat{u}_{i-1}$ and $\widehat{u}_{i+1}$. In this case, we take $u_{xx} = 0$ and consider the y-direction. If the same occurs and we take $u_{yy} = 0$, then 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

$$\frac{-1}{\Delta x^2} u_i = \frac{-1}{\Delta x^2} \gamma. \tag{78}$$

Notice that we cannot make such a distinction without considering the two-dimensionality of the problem. (See Figure 4.)

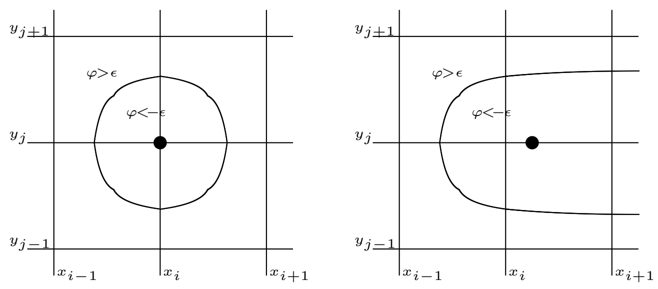

Figure 4: Impact of Two-Dimensionality on the Poisson Solver: In the left figure, the interface Σ is unresolved near (x_i, y_j). In the right figure, $u_{yy} = 0$ but the interface is still resolved.

6.4 Application to the Concentration and Pressure Equations

Using the method described in the previous section, we discretize the concentration equation for c in (31) by using

$$f(c, \mathbf{x}) = c, \quad g(c, \mathbf{x}) = 1, \quad \gamma = 1 \tag{79}$$

in (57).

Similarly, we discretize the pressure equation for p in (32) with

$$f(p, \mathbf{x}) = \begin{cases} -G\,(c - A) & \text{in } \Omega_P \\ G\,G_N & \text{in } \Omega_N \end{cases}, \quad g(p, \mathbf{x}) = \kappa, \quad \gamma = 0 \tag{80}$$

in (57).

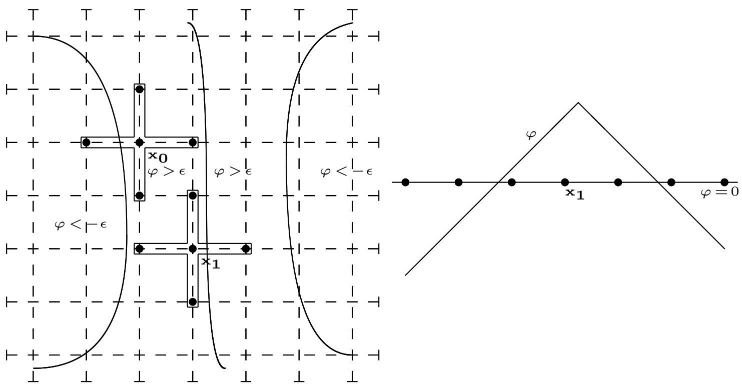

Figure 5: Effect of discontinuity on κ and $\mathbf{n}$: There are two interfaces close together, as shown in the left figure. The middle curve shows the points equidistant from both interfaces. The right figure shows a cross-section through $\mathbf{x_1}$ of the level set function; the "peak" in the middle is equidistant from the two interfaces and a point of discontinuity in φ_x and φ_y. The standard techniques work well at $\mathbf{x_0}$ (where φ_x and φ_y are continuous), whereas they break down numerically at $\mathbf{x_1}$.

7 Other Discretizations

7.1 Normal Vector

Recall that for our level set formulation, $\mathbf{n} = \nabla\varphi / |\nabla\varphi|$ and $|\nabla\varphi| = 1$. However, this formulation breaks down numerically for points nearly equidistant from two interfaces. Two features characterize this breakdown: one or both partial derivatives of φ demonstrates a sign change across the standard 5-point stencil, and the norm of the gradient calculated by the standard centered differences is not equal to 1. See Figure 5. Therefore, we define a one-sided difference for this situation:

1. Calculate φ_x and φ_y using the standard centered differences. Define $d = \sqrt{\varphi_x^2 + \varphi_y^2}$. If $|d-1| \leq \eta$, define $\mathbf{n} = \frac{1}{d}(\varphi_x, \varphi_y)$, where η is some small positive tolerance. In our simulations, we chose $\eta = .25$ because it worked well to detect all the discontinuities in φ_x and φ_y of interest.

2. Otherwise, first consider φ_x;compute $\Delta_x^-\varphi$ and $\Delta_x^+\varphi$. If $(\Delta_x^-\varphi)(\Delta_x^+\varphi) > \epsilon$, keep φ_x as defined in the previous step, as there is no sign change in φ_x. Otherwise, take

$$\varphi_x = \frac{1}{\Delta x}\begin{cases} \Delta_x^-\varphi & \text{if } |\varphi_{i-1}| < |\varphi_{i+1}| \\ \\ \Delta_x^+\varphi & \text{else.} \end{cases}$$

That is, we use the one-sided difference involving the points closest to the interface. Define φ_y similarly, recalculate d, and define $\mathbf{n} = \frac{1}{d}\left(\varphi_x, \varphi_y\right)$.

7.2 Curvature

When calculating the curvature κ, we use the standard centered differences for each of the partial derivatives in (35) except when the level set function has discontinuous partial derivatives within the 5-point stencil. In that case, we modify our technique to use one-sided differences of $\nabla \cdot \mathbf{n}$:

1. If the test $|d-1| \leq \eta$ was satisfied in the first step of the normal vector calculation, we use the standard centered differences for all the partial derivatives in κ.

2. Otherwise, we take $\kappa = \nabla \cdot \mathbf{n}$. We calculate the partial derivatives of the components of $\mathbf{n}$ using the same one-sided technique introduced in the normal vector calculation.

7.3 Gradients

The normal velocity in (33) requires ∇p on Σ. For our method, we must also calculate the gradients in Ω. Let u be a function whose gradient we wish to calculate. For interior points, we use the five-point stencil

$$u_x(x_i) = \frac{1}{12\Delta x}\Big(u_{i-2} - 8u_{i-1} + 8u_{i+1} - u_{i+2}\Big) + \mathcal{O}\left(\Delta x^4\right) \tag{81}$$

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 $x_{i\pm1} \in \Omega_0$, 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.

8 Extensions

In the level set method, we advect the interface Σ implicitly by advecting the level set function φ. Consequently, we must extend the normal velocity off the interface to the node points in a band surrounding Σ. By an extension velocity, we mean a function $\widetilde{V}$ defined on a larger domain than that of V such that $\widetilde{V}\big|_{\Sigma} \equiv V$. Beyond this requirement, there is considerable freedom in choosing $\widetilde{V}$. Before launching into the details of our technique, we outline our overall strategy and the motivation for it.

Evaluating the velocity given in (33) both on and off Σ would give a natural extension velocity. However, as the pressure gradient (in the context of the interior Poisson problem) is only defined in Ω and on Σ, this approach is not initially possible; consequently, our first step is to extend the components of the gradient outside Σ. Once the gradient has been extended, we can evaluate the velocity as in (33) wherever we wish.

However, this is not the most ideal extension. Because the velocity varies in the normal direction, it does not maintain the spacing between the level sets of φ, which adversely affects the accuracy of φ. Therefore, we apply a PDE-based extension technique that enforces $\nabla V \cdot \mathbf{n} = 0$ in a band around Σ [23]. This helps ensure the accuracy of the level set function, which in turn improves the accuracy of the subcell resolution in the Poisson solver, the normal vectors, and the curvature in the intermediate steps of the TVD-RK method.

One may be tempted to skip the gradient extension and simply define the velocity on Σ and in Ω as in (33) and then extend with the PDE method. However, this gives innacurate and unstable results, as there is not enough data about the interface to successfully define the spatial derivatives.

Once we have extended the velocity, we apply a Gaussian filter in a narrow band about the interface. This allows us to remove high-frequency noise from the speed function that would otherwise perturb the interface. Because this only smooths the data in a very thin band about the interface, we must extend the smoothed velocity with the PDE technique one final time. We found this technique to work best among the numerous combinations of extension and smoothing available.

8.1 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 from a given point $\mathbf{x_0}$ [3, 26]. 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 $\mathbf{n}\left(\mathbf{x_0}\right)$ points away from the interface, and $\left|\varphi\left(\mathbf{x_0}\right)\right|$ gives the distance to the interface. Therefore, the vector

$$\mathbf{W}\left(\mathbf{x_0}\right) = -\left|\varphi\left(\mathbf{x_0}\right)\right| \mathbf{n}\left(\mathbf{x_0}\right) = -\varphi\mathbf{n} \tag{82}$$

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

$$\mathbf{x_1} = \mathbf{x_0} + \mathbf{W} \tag{83}$$

explicitly gives the closest point to $\mathbf{x_0}$ on Σ. See Figure 6.

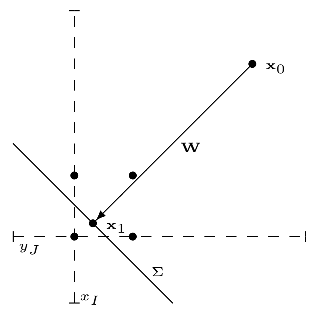

Figure 6: Finding the closest point on the interface.

8.2 Gradient Extension

Because the gradient algorithm only defines the gradient where $\varphi \leq \epsilon$, we must extend it 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 ∇p individually.

Our approach for extending the function f to a point $\mathbf{x}$ is to interpolate f in a direction aligned with the computational mesh. This allows the use of a high-order, one-dimensional interpolation, which attains high accuracy without the complexity of multidimensional interpolation. We choose the direction of the interpolation according to the vector $\mathbf{W}$ defined in the previous section.

We preserve information flow in the outward direction from the interface by successively updating the points closest to the interface. We keep track of which points we need to update (with extended values of f) by introducing a temporary lattice U and a counter C of the number of points remaining that require an update. As we extend f to points specified by U, we update U and the counter C. We detail our approach to implement the scheme outside of Σ within a band of width R below:

1. Create a temporary array $U[i, j]$ of zeros, and set $U_{i,j} = 1$ if f has not been defined at (x_i, y_j) and $\varphi(x_i, y_j) \leq R + \epsilon$. Set the counter C equal to the total number of points flagged for extension, indicated by $U_{i,j} > 0$. This gives us a simple means of keeping track of which points and how many require extension.

2. While $C > 0$, continue with the following steps:

3. Select among the remaining points indicated by U the point closest to Σ by choosing (I, J) such that
$$\varphi_{I,J} = \min\left\{\varphi_{i,j} : U_{i,j} > 0.0\right\}.$$

4. Define $\mathbf{W} = (W_x, W_y)$ according to (82).

5. We define integers m_x, m_y that express the direction of $\mathbf{W}$ in terms of the indices of f. Define m_x according to
$$m_x = \begin{cases} -1 & \text{if } W_x < -\epsilon \\ 0 & \text{if } |W_x| \leq \epsilon \\ 1 & \text{if } W_x > \epsilon. \end{cases}$$
Define m_y similarly. For example, if $\mathbf{W} = (3, -2)$, then $(m_x, m_y) = (1, -1)$.

   These integers, as modified in the following step, will allow us to express the points in the interpolation as a single case, rather than several.

   Note that if $m_x = m_y = 0$, then f is already updated at (x_i, y_j), as $\mathbf{W} = \mathbf{0}$ if and only if $\varphi(x_i, y_j) = 0$. Accordingly, set $U_{I,J} = 0$, reduce C by 1, and return to step 2 in such a case. Otherwise, continue to step 6.

6. Choose the direction of the interpolation according to whether $\mathbf{W}$ is more horizontal than vertical, more vertical than horizontal, or nearly diagonal. We do this by setting $m_y = 0$ if $|W_x| > |W_y| + \epsilon$ (the direction towards the interface is mostly horizontal), by setting $m_x = 0$ if $|W_y| > |W_x| + \epsilon$ (the direction towards the interface is mostly vertical), or by leaving m_x and m_y as they are if $\Big||W_x| - |W_y|\Big| \leq \epsilon$ (the direction towards the interface is nearly diagonal).

7. Now that m_x and m_y are defined, we can express the points along the direction of interpolation with a single notation:
$$\begin{aligned} \mathcal{S} &= \Big\{\left(x_{I+m_x}, y_{J+m_y}\right), \left(x_{I+2\,m_x}, y_{J+2\,m_y}\right), \\ &\qquad \left(x_{I+3\,m_x}, y_{J+3\,m_y}\right), \left(x_{I+4\,m_x}, y_{J+4\,m_y}\right)\Big\} \\ &= \Big\{\left(x_{I+k\,m_x}, y_{J+k\,m_y}\right)\Big\}_{k=1}^{4}. \end{aligned} \tag{84}$$

   We choose which points in the set $\mathcal{S}$ to use for interpolation (and hence the order of the interpolation) so as to preserve information flow outward from the interface. Because we have ordered our scheme to extend f in order of increasing φ, we can assume that points where the level set function has lower values are already updated. Hence, we can attain the desired information flow property by only using $f_{I+k\,m_x,J+k\,m_y}$ in the interpolation if

$$\varphi_{I+k\,m_x,J+k\,m_y} < \varphi_{I+(k-1)\,m_x,J+(k-1)\,m_y},$$

that is, if

$$\left(x_{I+k\,m_x}, y_{J+k\,m_y}\right)$$

is closer to the interior of Ω than

$$\left(x_{I+(k-1)\,m_x}, y_{J+(k-1)\,m_y}\right).$$

Evaluate the interpolation at (x_I, y_J), set $U_{I,J} = 0$, reduce the counter C by 1, and return to step 2.

In our simulations, we apply this technique to each component of ∇p within a band of width $R = 5\Delta x$.

8.3 Velocity Extension

Once we have a velocity defined in a band about the interface Σ, we apply an extension routine to ensure that $\nabla\widetilde{V}\cdot\mathbf{n} = 0$. By [26, 28], we can attain this within a band of width R by evolving

$$\widetilde{V}_\tau + \text{sign}(\varphi)\mathbf{n}\cdot\nabla\widetilde{V} = 0, \tag{85}$$

where τ is pseudo-time, $\text{sign}(\varphi)\mathbf{n}\cdot\nabla\widetilde{V}$ is discretized according to the third-order or fifth-order WENO scheme, and $\mathbf{n}$ is discretized as above.

As was the case with the level set initialization, the orthogonality property $\nabla\widetilde{V}\cdot\mathbf{n} = 0$ becomes true first near the zero level set and advects outward, so if we only require $\widetilde{V}$ within a distance R of Σ, we evolve (85) until $\tau = R$.

8.4 An Alternative Velocity Extension Technique

Although we do not use the following method, we note that the vector $\mathbf{W}$ defined in (82) suggests an alternative extension technique. In that technique, if we wish to extend V to $\mathbf{x_0}$, we define $\mathbf{W}$ as in (82) and

$$\mathbf{x_1} = \mathbf{x_0} + \mathbf{W} \tag{86}$$

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

$$\frac{\partial \widetilde{V}}{\partial \mathbf{n}} \equiv 0,$$

that is, $\nabla \widetilde{V} \cdot \mathbf{n} = 0$.

This approach differs from the discrete, fast marching velocity extension given in [3] in several ways. First, the fast marching extension technique as presented in [3] 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 can make use of the level set function to readily locate the closest position on the interface and direction of interpolation, 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 [20].

We found this alternative technique to be second-order, but we chose not to use it because it was not always stable.

9 Other Optimizations, Improvements, and Considerations

9.1 Gaussian Smoothing

Because the normal velocity is based (in part) upon the extended gradients of a second-order accurate pressure solution, the speed can develop numerical noise when even small perturbations in the level set function φ are present. Consequently, some smoothing to rid the data of this noise seems to be necessary. We found that it was not necessary to smooth any other quantities. As in image processing, we smooth the normal velocity by applying a Gaussian filter. The Gaussian filter is well-suited to this application because:

1. its kernel is separable, meaning that the data can be smoothed one dimension at a time for greater efficiency; and

2. its frequency response is greatest at high spatial frequencies and is monotone decreasing for lower spatial frequencies, meaning that it smooths high-frequency noise while preserving

lower-frequency details, and its response is predictable.

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

$$\widehat{f}_I = \frac{1}{\sigma\sqrt{2\pi}} \sum_i f_{I-i} \exp\left(-\frac{(i\,\Delta x)^2}{2\sigma^2}\right) \Delta x, \tag{87}$$

where σ is the standard deviation of the filter. Typically, $\sigma = N\Delta x$ for some integer N. For $|i\Delta x| \geq 3N\Delta x$, the exponential function in the convolution has a very small value (less than approximately .0111); consequently, we can truncate the sum above to

$$\widehat{f}_I = \frac{1}{A}\frac{1}{N\sqrt{2\pi}} \sum_{i=-3N}^{3N} f_{I-i} \exp\left(-\frac{1}{2}\left(\frac{i}{N}\right)^2\right), \tag{88}$$

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

To smooth a two-dimensional data-set, we use (88) first in the x-direction, and then again in the y-direction. In our calculations, we use $\sigma = 2\Delta x$. 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 use a narrow band of width $3\Delta x$.

9.2 Width of the Band for the Narrow Band Technique and Size of the Computational Domain

Now that we have outlined our numerical techniques, we revisit the narrow band method and determine the width of the band. We start by considering the smoothed normal velocity.

We require a smoothed normal velocity within three nodes of the interface. Thus, $R \geq 3\Delta x$. If σ is the standard deviation of the Gaussian filter for the smoothing, 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 $R \geq 3\Delta x + 3\sigma\sqrt{2}$.

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, $R \geq 3\Delta x + 3\sqrt{2}\sigma + 2\Delta x$. 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

$$R = 1.25\Big(5\Delta x + 3\sqrt{2}\sigma\Big) = R = 1.25\Big(5\Delta x + 6\sqrt{2}\Big)\Delta x, \tag{89}$$

where we have used $\sigma = 2\Delta x$.

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

$$|\varphi(\mathbf{x})| > R + 3\Delta x, \tag{90}$$

for any $\mathbf{x} \in \partial\left([a, b] \times [c, d]\right)$, we must extend the computational domain. We therefore modify the width R of the narrow band to include this distance (so that we can accurately gauge proximity to the computational boundary in (90)):

$$R = 1.25\Big(5\Delta x + 3\sqrt{2}\sigma + 3\Delta x\Big) = R = 1.25\Big(8\Delta x + 6\sqrt{2}\Big)\Delta x, \tag{91}$$

where we have used $\sigma = 2\Delta x$.

9.3 Storage of Large Banded Matrices

The Poisson solver requires that we store relatively large, sparse banded coefficient matrices. This can be accomplished in a compact, memory-saving manner as presented in [25].

Suppose for an $n \times n$ matrix M that $m_{i,j} = 0$ for all $j > i + u$ and all $i > j + \ell$. Then M has lower bandwidth (at most) ℓ and upper bandwidth (at most) u.[2] We store the nonzero entries of M as an $n \times (\ell + u + 1)$ matrix E as follows:

$$E_{n,k} = m_{n,\ell+1+(k-n)}. \tag{92}$$

For example, the matrix

$$M = \begin{bmatrix} 3 & 1 & & & & & \\ 4 & 1 & 5 & & & & \\ 9 & 2 & 6 & 5 & & & \\ & 3 & 5 & 8 & 9 & & \\ & & 7 & 9 & 3 & 2 & \\ & & & 3 & 8 & 4 & 6 \\ & & & & 2 & 4 & 4 \end{bmatrix}$$

[2]In our problem, if we solve the Poisson system (57) on an $N \times N$ discretization with a row or column ordering, then the upper and lower bandwidth are $3(N-2)$.

is stored as

$$E = \begin{bmatrix} \square & \square & 3 & 1 \\ \square & 4 & 1 & 5 \\ 9 & 2 & 6 & 5 \\ 3 & 5 & 8 & 9 \\ 7 & 9 & 3 & 2 \\ 3 & 8 & 4 & 6 \\ 2 & 4 & 4 & \square \end{bmatrix}.$$

The elements indicated by $\square$ have no data associated with M; they could be used to store the upper and lower bandwidths of M, for example. In our implementation, we opt instead to include E as a data member of a C++ banded matrix class which also stores information on the size and bandwidths of the original matrix as well as useful member functions. (e.g., efficient matrix-vector multiplication, as presented in the next section, linear solvers, etc.)

Because these matrices are often non-symmetric but positive-definite, we use an optimized biconjugate gradient method to solve the linear systems resulting from the concentration and pressure solvers. Under certain conditions related to pivoting, the biconjugate gradient method fails when a coefficient experiences numerical blowup [12]. In such cases, we use a Gaussian elimination method with partial pivoting that has been optimized for our banded matrix storage scheme; the banded Gaussian elimination details can be found in [25]. In future work, we will further address this problem by modifying the Bi-Lanczos formulation of the LU decomposition that is the basis of the method or by using the QMR approach [12].

9.4 Optimized Banded Matrix-Vector Multiplication

Because the matrix-vector product is central to the biconjugate gradient method, we developed an algorithm to optimize that product. Suppose an $n \times n$ matrix M is banded with upper bandwidth u and lower bandwidth ℓ, and suppose we want the k-th entry of $\mathbf{w} = M\mathbf{v}$, where $\mathbf{v}$ is $n \times 1$. Then

$$w_k = \sum_{j=k-\ell}^{k+u} M_{k,j} v_j.$$

Suppose, however, that only several of the entries $M_{k,j}$ in the sum are nonzero. Then many of the operations are wasted, and in machine-precision arithmetic, the zero terms in the sum will contribute additional error to the final result. We therefore seek to minimize the wasted computation, which will also increase the accuracy of the result.

We attain this optimization in several steps. For the matrix M, we store two additional arrays B and L of length $\ell + u + 1$, which we initialize to all zeros. As we construct the matrix M, whenever we make a non-zero entry in a particular band in the storage matrix for M, we set the corresponding value in B equal to 1. In this way, we obtain a representation of which bands contain nonzero elements in the matrix.

Once we have completely constructed the matrix M, we compile a list of nonzero bands and keep a count N of the number of nonzero bands. We do this by filling L with the column numbers of B which correspond to nonzero bands in ascending order. For example, for the matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & & & \\ 0 & 1 & 0 & 0 & 0 & 1 & & \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \\ & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ & & 0 & 0 & 1 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 1 & 0 \\ & & & & & 1 & 0 & 1 \end{bmatrix},$$

the lower bandwidth is 2, the upper bandwidth is 4,

$$B = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix},$$

$N = 4$, and

$$L = \begin{bmatrix} 1 & 3 & 4 & 7 & 0 & 0 & 0 \end{bmatrix}.$$

Then for each entry in the matrix-vector product, we truncate the sum to the indicies designated by the first N members of L. This technique only requires the storage of two additional one-dimensional arrays that are small relative to the size of the original matrix M. The setup adds little computational cost because B is constructed as the matrix M and its storage representation are created, and L is constructed only one time.

10 Convergence and Validation Results

We now give convergence results and validate the scheme against known solutions. We shall measure the error in an approximation u_{approx} of u by

$$\max |u_{approx}(\mathbf{x}_i) - u(\mathbf{x}_i)|, \tag{93}$$

where $\mathbf{x}_i$ ranges over all the nodes of a fixed region.

We define the order of convergence by

$$\text{order } = \frac{\log\left(\frac{\text{error}_1}{\text{error}_2}\right)}{\log\left(\frac{\text{mesh size 1}}{\text{mesh size 2}}\right)}. \tag{94}$$

10.1 Level Set Reinitialization and Curvature

We reintialize the level set function for a circle of radius 2, whose exact level set representation is given by

$$\varphi(x, y) = \sqrt{x^2 + y^2} - 2, \tag{95}$$

using both the third-order and fifth-order WENO schemes. Our calculations here and throughout shall use the third-order TVD-RK time discretization. Because there is a singularity in φ at $\mathbf{x} = 0$, we shall measure the maximum absolute error in the band $1.5 < r < 2.5$, where $r = \sqrt{x^2 + y^2}$. Both discretizations converge as expected. See Table 1.

mesh	WENO3	WENO5
51×51	7.97 e-4	5.72 e-6
101×101	7.14 e-5	1.72 e-7
201×201	7.31 e-6	9.66 e-9
overall order	3.38	4.60

Table 1: Absolute error in φ after reinitialization.

The accuracy and stability of the level set function are essential for properly calculating variables that are sensitive to numerical noise in the level set function. We illustrate this by computing the curvature in the band $1.5 < r < 2.5$, where the exact solution should be $\frac{1}{r}$; the results are shown in Table 2. Both reinitialization techniques yield curvatures that converge just under second-order, but the fifth-order WENO scheme comes closer to the convergence attained by using the exact level set function, and the error is a full order of magnitude less. As we shall see in the next section, the magnitude and order of the error in the level set function and curvature greatly impact on the convergence of the pressure solution.

mesh	exact φ	WENO3	WENO5
51×51	.00155	.0154	.00145
101×101	4.55 e-4	.00493	4.51 e-4
201×201	1.14 e-4	.00127	1.14 e-4
overall order	1.88	1.80	1.83

Table 2: Absolute error in κ after level set reinitialization.

10.2 Poisson Solver

Example 1: We examine the sample concentration problem

$$\begin{cases} \nabla^2 u = u & \mathbf{x} \in \Omega \\ u = 1 & \mathbf{x} \in \Sigma \\ u = 1 & \mathbf{x} \in \Omega_0, \end{cases} \tag{96}$$

where Σ is a circle of radius 2 in two dimensions. The exact solution is

$$u = \begin{cases} \frac{I_0\left(\sqrt{x^2+y^2}\right)}{I_0(2)} & \mathbf{x} \in \Omega \\ 1 & \mathbf{x} \in \Sigma \\ 1 & \mathbf{x} \in \Omega_0. \end{cases} \tag{97}$$

We solve on the computational domain $[-4,4] \times [-4,4]$ with the exact level set function in (95) and with the same level set function that has been reinitialized with the third- and fifth-order WENO schemes. The numerical error (measured as the maximum absolute error over the computational mesh) is given in Table 3.

We see that the Poisson solver converges to approximately second-order with all three meth-

mesh	exact φ	WENO3	WENO5
51×51	4.63 e-4	5.30 e-4	4.64 e-4
101×101	1.21 e-4	1.25 e-4	1.21 e-4
201×201	3.05 e-5	3.08 e-4	3.05 e-5
overall order	1.96	2.06	1.96

Table 3: Absolute error in u for the Poisson solver in Example 1.

ods, although the fifth-order WENO scheme yields smaller errors for coarser meshes.

Example 2: We examine the sample pressure problem

$$\begin{cases} \nabla^2 u = 0 & \mathbf{x} \in \Omega \\ u = \kappa - \frac{5}{2}\left|\mathbf{x}\right|^2 & \mathbf{x} \in \Sigma \\ u = 0 & \mathbf{x} \in \Omega_0, \end{cases} \tag{98}$$

where Σ is a circle of radius 2 in two dimensions. The exact solution is

$$u = \begin{cases} -9.5 & \mathbf{x} \in \Omega \\ -9.5 & \mathbf{x} \in \Sigma \\ 0 & \mathbf{x} \in \Omega_0. \end{cases} \tag{99}$$

As before, we solve on the computational domain $[-4, 4] \times [-4, 4]$ and with the exact and reinitialized level set function in (95). The numerical error (measured as the maximum absolute error over the computational mesh) is given in Table 4.

mesh	exact φ	WENO3	WENO5
51×51	7.05 e-4	.00575	6.60 e-4
101×101	2.00 e-4	.00493	2.00 e-4
201×201	5.00 e-5	.00127	5.00 e-5
overall order	1.91	1.09	1.86

Table 4: Absolute error in u for the Poisson solver in Example 2.

We see that the Poisson solver converges to approximately second-order with an exact level set function; this is the order of the method. However, when we use the third- and fifth-order WENO schemes for the level set reinitialization, we see the importance of having an accurate level set function; switching the WENO method from third- to fifth-order improves the convergence of the Poisson solver from first-order to second-order in this example. The main difference between this and the previous example is that the boundary condition is based on curvature, which depends on two derivatives of φ. As such, the solution is very sensitive to error in the level set function; error in φ has the greatest impact on the interpolation of the boundary condition.

10.3 Gradients

Example 1: Let us return to the sample concentration problem. Inside and up to Σ, we have

$$\nabla u = \frac{I_1(r)}{I_0(2)} \frac{\mathbf{x}}{r}. \tag{100}$$

Let us examine the error in u_x for the same mesh sizes as before. We test where $1.5 < r < 2.0$. The results are given in Table 5. The errors for the exact level set function φ demonstrate the

mesh	exact φ	WENO3	WENO5
51×51	.00155	.00503	.0049
101×101	7.54 e-4	8.6 e-4	7.54 e-4
201×201	1.24 e-4	1.37 e-4	1.24 e-4
overall order	1.82	2.60	2.65

Table 5: Absolute error in u_x for the Poisson solver in Example 1.

method to converge to just under second order. The errors calculated for the reinitialized level set functions show higher orders of convergence because the error at low resolutions is so large. We can see in these examples, however, that the fifth-order WENO method for the reinitialization gives the results closest to those corresponding to an exact level set function.

Example 2: Let us now turn to the sample pressure problem. Inside and up to Σ, we have $\nabla u = 0$. Note that this example is relevant to the normal velocity calculation. We examine the error in u_x for the same mesh sizes and node points as before. The results are given in Table 6. Again, the fifth-order WENO method gives much better results than the third-order WENO method; the level set that was reinitialized with the third-order WENO method converged under first order, whereas the fifth-order method yielded second-order gradients.

mesh	exact φ	WENO3	WENO5
51×51	.00489	.241	.00393
101×101	4.04 e-4	.156	9.53 e-4
201×201	9.57 e-5	.0875	2.22 e-4
overall order	2.84	.73	2.07

Table 6: Absolute error in u_x for the Poisson solver in Example 2.

10.4 Speed Extension

We now test the speed extension. As before, we test on the computational domain $[-4, 4] \times [-4, 4]$ with 51×51, 101×101, and 201×201 mesh sizes. We shall use the exact level set function φ and the reinitialized level set functions using the third- and fifth-order WENO methods.

mesh	exact φ	WENO3	WENO5
51×51	.00119	6.24 e-4	.00119
101×101	2.91 e-4	2.52 e-4	2.91 e-4
201×201	7.26 e-5	7.03 e-5	7.26 e-5
overall order	2.02	1.57	2.02

Table 7: Absolute error in velocity extension without Gaussian smoothing.

mesh	exact φ	WENO3	WENO5
51×51	.00104	5.57 e-4	.00103
101×101	2.15 e-4	1.86 e-4	2.15 e-4
201×201	7.12 e-5	6.91 e-5	7.12 e-12
overall order	1.93	1.51	1.93

Table 8: Absolute error in velocity extension with Gaussian smoothing.

We test the extension of a sample velocity defined by

$$V = -\mathbf{n} \cdot \mathbf{x}$$

Hence, the extended speed should be $V \equiv -r$ with $V\big|_\Sigma \equiv -2$. We shall extend the components of $\mathbf{x}$ to a distance of $5\Delta x$, follow with the PDE-based normal velocity extension, and check the absolute error on $1.5 < r < 2.5$. The results without Gaussian smoothing are given in Table 7. The results with Gaussian smoothing are given in Table 8. The velocity extension technique is second-order both with and without Gaussian smoothing when used with the fifth-order WENO method, whereas the third-order WENO method gives sub-second-order convergence. The Gaussian smoothing reduces the magnitude of the error somewhat while having little impact on the order of convergence. Its larger impact is in eliminating spurious variations in the velocity that can lead to undue perturbations in the level set function. If such variations are not reduced, a feedback cycle ensues: the spurious variations in the velocity perturb the level set function, causing large pointwise errors in the curvature and consequently in the pressure gradient and velocity. The large errors in the velocity further perturb the level set function. This also creates a prohibitive time-step restriction by the CFL condition and reduces the computational speed of a simulation.

10.5 Convergence of Overall Scheme

We now test the convergence of the overall scheme. We start with an unperturbed circle of radius 2.0 and advect it to $t = 0.25$ according to (32)-(33) with $A = .5$, $G = 20$, $G_N = 1$, $N = 0$ (no necrosis), and no chemotherapy. According to [10], the exact solution in this case is given by

$$R'(t) = -AG\frac{R}{d} + G\frac{I_1(R)}{I_0(R)}. \tag{101}$$

Solving (101) to $t = .25$, we find that $R_{25} = R(.25) \approx 2.74749026506514$. Thus, at $t = .25$, the exact level set function should be given by

$$\varphi(x, y) = \sqrt{x^2 + y^2} - R_{25}. \tag{102}$$

We use the fifth-order WENO method and third-order TVD-Runge Kutta, and we test the maximum absolute error in φ within the band $1.5 < r < 2.5$. The results given in Table 9 indicate second-order convergence for the overall scheme.

mesh	Error for WENO5
51×51	.0238
101×101	.00674
201×201	.00136
overall order	2.06

Table 9: Absolute error in φ at $t = 0.25$.

10.6 Comparison with Known Results

We now test the scheme against the results given by a spectrally accurate boundary integral method in [10]. We use $A = 0.5$, $G = 20$, $N = 0$. The initial interface is a circle of radius 2 perturbed as

$$\Sigma(s) = (2 + .2\cos(2s), 2 + .2\sin(2s)), \qquad 0 \le s \le 2\pi. \tag{103}$$

We use $\Delta x = \Delta y = .08$ with fifth-order WENO and third-order TVD-Runge Kutta. We plot the two solutions at $t = 1.0$, $t = 1.92$, and $t = 2.5$ in Figure 7; we also plot our solution at $t = 3.0$, $t = 3.9$, and $t = 5$ to demonstrate that our method continues beyond morphological changes such as the capture of healthy tissue. The dashed solution is that given by the boundary integral technique in [10], and the solid is that given by the finite difference techniques presented here. As we can see, there is good agreement between the two techniques. A full animation of this simulation will be available at `http://www.ima.umn.edu/~macklin/Tumor.html`.

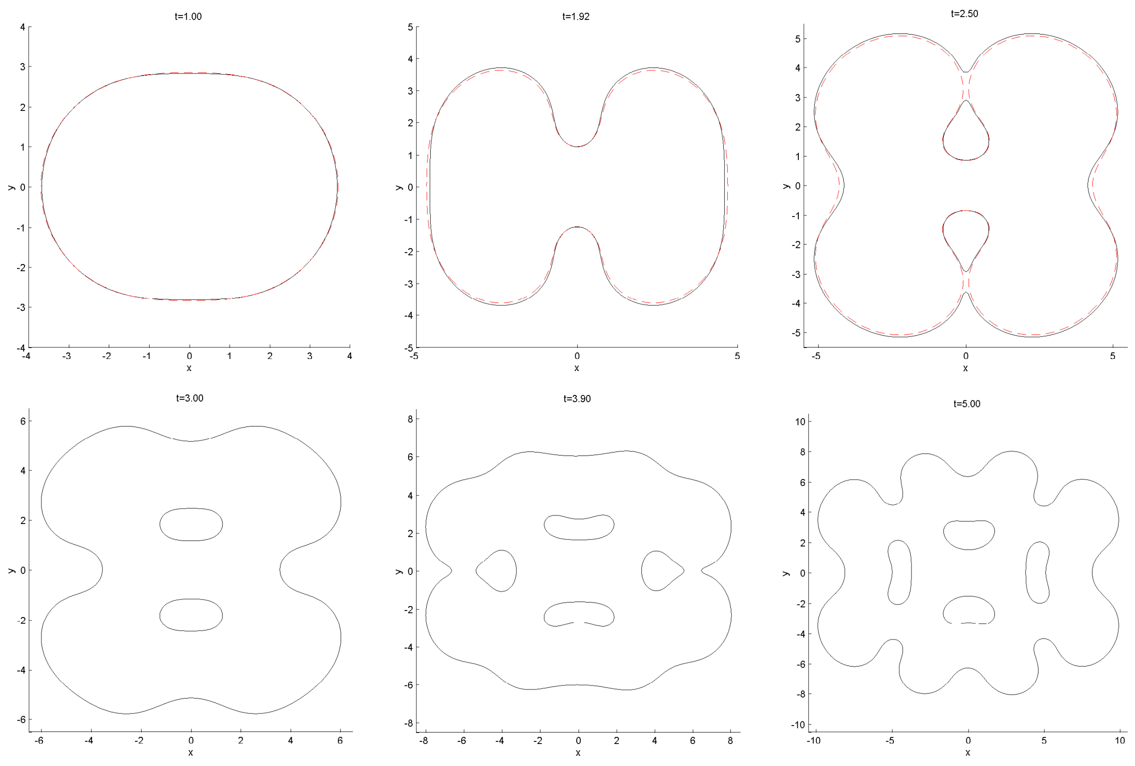

Figure 7: Comparison of computed solutions: We compare our solutions at $t = 1.0$, $t = 1.92$, and 2.50 to results from a spectrally-accurate boundary integral method in [10], indicated by the dashed curves. We also plot our solution at $t = 3.0$, $t = 3.9$, and $t = 5.0$; because the boundary integral method cannot continue beyond pinchoff, there are no results for comparison at these times.

11 Preliminary Results and Discussion

11.1 A Parameter Study on G_N

In this study, we hold $A = 0$, $G = 20$, and $N = .35$. We vary the value of G_N with values

$$G_N = .1, \quad G_N = 1, \quad G_N = 10, \tag{104}$$

and we conduct one baseline simulation with no necrosis ($N = 0$).

All simulations begin with the curve given in (103). The simulation for $N = 0$ was done with $\Delta x = \Delta y = .16$. The simulations with $G_N = .1$, $G_N = 1$, and $G_N = 10$ used $\Delta x = \Delta y = .1067$.

As was analyzed in [10], with $N = 0$, $A = 0$, $G = 20$, we are in the moderate vascularization parameter regime ($A \leq 0$ and $G > 0$). In this regime, tumor growth is rapid and unbounded, and perturbations decay to zero for two-dimensional growth [10]. We observe this in our simulations as well. For small G_N ($G_N = 0.1$), we observe that necrosis has a stabilizing effect on growth: the tumor volume is smaller than for a nonnecrotic tumor at any given time, and perturbations show no appreciable growth. However, for larger values of G_N (1.0 and 10.0), we see that while the necrosis further shrinks the overall size of the tumor, the morphology has been greatly destabilized, and perturbations grow rapidly. This can been seen in a plot of the four simulations at $t = 0.5$ in Figure 8. We shall use the view $[-12, 12] \times [-12, 12]$ in all our plots for better comparison of relative sizes. The dark regions denote Ω_N.

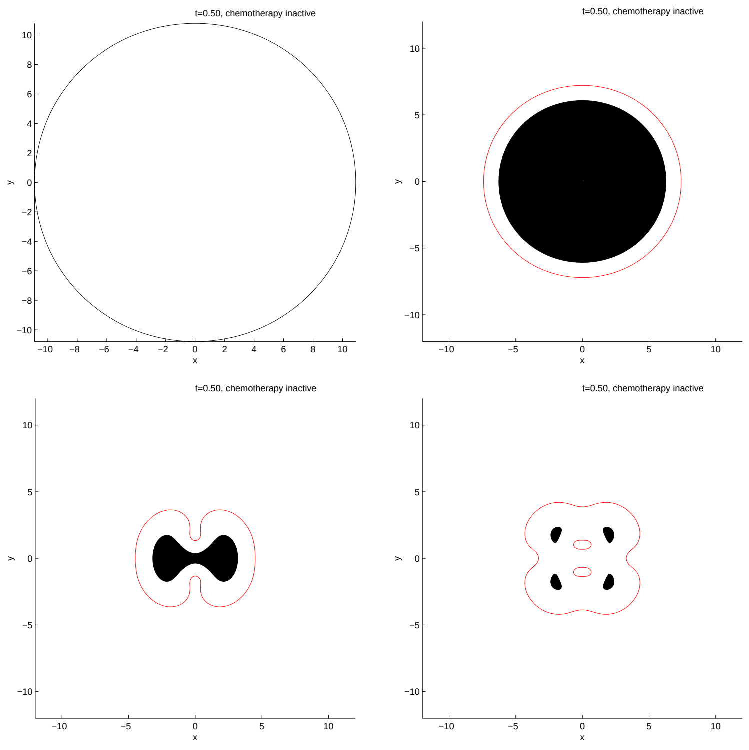

Figure 8: A Parameter Study on G_N: We plot (from left to right, starting at the top) for $N = 0$ (no necrosis), $G_N = .1$, $G_N = 1$, and $G_N = 10$ at $t = 0.5$.

11.2 A Parameter Study on G_N with Chemotherapy

We now conduct the same parameter study while applying a chemotherapy regimen. We alternate A between 0 and .5 in .15 increments, starting with $A = 0$ at $t = 0$.

We note when $A = .5$, the tumor is in the low vascularization parameter regime in [10]. In this case, perturbations often tend to grow with time (the behavior depends upon the mode of the perturbation and the radius of the underlying unperturbed tumor) [10]. In our simulation with no necrosis, the tumor demonstrates shape instabilities that grow during the chemotherapy sessions and shrink between those sessions. The tumor demonstrates significant shape instability by the end of the fourth chemotherapy session ($t = 1.19$); by the seventh session ($t = 2.04$), the shape instabilities are quite complex. See Figure 9.

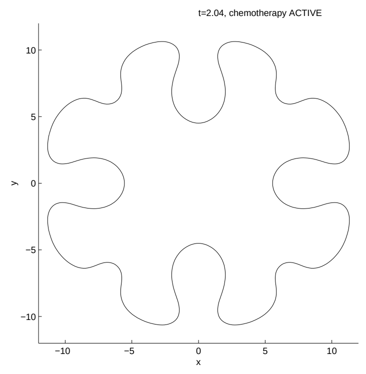

Figure 9: Complex shape instability for a non-necrotic tumor undergoing chemotherapy at $t = 2.04$.

With necrosis and a small value for G_N ($G_N = .1$), the shape instabilities are greatly reduced, as can be seen at the end of the second and fourth chemotherapy sessions. With larger values for G_N, however, the development of these shape instabilities is accelerated during the chemotherapy sessions. See Figure 10, where we plot the four simulations at the end of the second chemotherapy session, just before the third session, and at the end of the fourth session. These results demonstrate that the outcome of a given chemotherapy regimen is greatly dependent upon the physical characteristics of the tumor to which it is applied; a chemotherapy session that is effective against one tumor may cause undesirable behavior when applied to another tumor (e.g., invasive fingering, the capture of healthy tissue, or tumor breakup). This suggests that for a given tumor, great care must be taken in selecting a chemotherapy regimen, and some optimization may be possible. We shall give an example of a chemotherapy regimen that leads to tumor breakup.

11.3 Breakup of Tumor Undergoing Chemotherapy

We give an example of a chemotherapy regimen that causes a tumor to break up into multiple, smaller tumors. We use $\Delta x = \Delta y = .16$ with $A_0 = 0$, $G = 20$, no necrosis, and with the initial curve given in (103). Our chemotherapy regimen is given in Figure 11.

In our simulation, the tumor begins to break up during the last chemotherapy session. The first breakup occurs at $t = 3.35$, and five additional breakups occur at 3.4, 3.43 (two breakups), 3.44, and 3.5. After the chemotherapy session is over, each separate tumor mass begins to grow. We give some plots from the evolution of this tumor in Figure 12.

12 Future Work

In future work, we will improve our model for tumor growth. In captured healthy regions Ω_C, we currently set $c = 1$ and $p = 0$ and $p = \kappa$ on Σ_i. This implies an external source of nutrient supply for the captured regions (e.g., such as being fed from above). In an improved model, nutrient reaches Ω_C solely by diffusion through Ω. In such a model, we have

$$\nabla^2 c = 0 \text{ in } \Omega_C \tag{105}$$

with c continuous and smooth across Σ_i.

Similarly, our current implementation does not enforce $\nabla \cdot \mathbf{u} = \nabla^2 p = 0$ in Ω_C. We could improve the model by setting $\nabla^2 p = 0$ in Ω_C and applying a jump boundary condition to the interior boundary:

$$[p]\Big|_{\Sigma_i} = 0. \tag{106}$$

We could also improve our model for chemotherapy. During each chemotherapy session, the level of A remains constant. Thus, the administered treatment is just as effective in increasing the death rate of the tumor cells at the beginning of the session as at the end. However, tumors often develop a resistance to treatment, so A should decay over time as the treatment loses its full effectiveness. Lastly, we plan to include new physiological effects in the model, such as angiogenesis.

In future work we will also strive to improve efficiency and extend capabilities. We shall optimize the Poisson solver and reduce memory use. We will also extend the methods to three dimensions and incorporate adaptive Cartesian meshes for improved resolution at lower computational cost.

References

[1] Table 32 in *Health, United States, 2002,* published by National Center for Health Statistics / Center for Disease Control (2002).

[2] *On growth and Form: Spatio-temporal pattern formation in Biology,* Ed. M.A.J. Chaplain, G.D. Singh, J.C. MacLachlan, Wiley Series in Mathematical and Computational Biology, New York, NY (1999).

[3] D. Adalsteinsson and J.A. Sethian, *The Fast Construction of Extension Velocities in Level Set Methods,* Journal of Computational Physics, Vol. 148, Issue 1 (1999), pp. 2-22.

[4] T. Aslam, R.P. Fedkiw, B. Merriman, and S. Osher, *A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method),* Journal of Computational Phyics, Vol. 152, Issue 2 (1999), pp. 457-492.

[5] H.M. Byrne, *In: A Survey of Models on Tumor Immune Systems Dynamics,* ed. J. Adam and N. Bellomo, Birkhauser, Boston (1996), pp. 15-87.

[6] H.M. Byrne and M.A.J. Chaplain, *Modelling the Role of Cell-Cell Adhesion in the Growth and Development of Carcinomas,* Mathematical and Computer Modelling, Vol. 24 (1996), pp. 1-17.

[7] R. Caflisch, R. Fedkiw, F. Gibou, and S. Osher, *A Level Set Approach for the Numerical Simulation of Dendritic Growth,* preprint (2002).

[8] L.T. Cheng, R. Fedkiw, F. Gibou, and M. Kang, *A Second Order Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains,* Journal of Computational Physics, Vol. 176, Issue 1 (2002), pp. 205ff.

[9] S. Chen, B. Merriman, S. Osher, and P. Smereka, *A Simple Level Set Method for Solving Stefan Problems,* Journal of Computational Physics, Vol. 135, Issue 1 (1997), pp. 8-29.

[10] V. Cristini, J.S. Lowengrub, and Q. Nie, *Nonlinear Simulation of Tumor Growth,* Journal of Mathematical Biology, Vol. 46, No. 3 (2003), pp. 191-224.

[11] V. Cristini, J.S. Lowengrub, and Q. Nie, *Tumor Growth in Silico and its Experimental Validation,* preprint (2003).

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

[13] R.P. Fedkiw, X.D. Liu, and M. Kang, *A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains,* Journal of Computatoinal Fluid Mechanics, Vol. 160, Issue 1 (2000), pp. 151-178.

[14] L.L. Feng, C.W. Shu, M. Zhang, *A Hybrid Cosmological Hydrodynamic/N-Body Code Based on the Weighted Essentially Non-Oscillatory Scheme,* Proceeding of 5th Sino-Germany Workshop on Cosmology and Galaxy Formation (2002)

[15] S. Gottlieb and C.W. Shu, *Total Variation Diminishing Runge-Kutta Schemes,* Mathematics of Computation, 67 (221), pp. 73-85

[16] S. Gottlieb, C.W. Shu, and E. Tadmor, *Strong Stability-Preserving High-Order Time Discretization Methods,* SIAM Review, Vol. 43, No. 1 (2001), pp. 89-112

[17] H.P. Greenspan, *On the Growth and Stability of Cell Cultures and Solid Tumors,* Journal of Theoretical Biology, Vol. 56 (1976), pp. 229-242.

[18] G.S. Jiang and D. Peng, *Weighted ENO Schemes for Multi-Dimensional Hamilton-Jacobi Equations,* SIAM Journal of Scientific Computation, Vol. 21, No. 6 (2000), pp. 2126-2143

[19] G.S. Jiang and C.W. Shu, *Efficient Implementation of Weighted ENO Schemes,* Journal of Computational Physics, Vol. 126, No. 2 (1996), pp. 202-228

[20] R. Malladi, J.A. Sethian, and B.C. Vemuri, *Shape Modeling with Front Propagation: A Level Set Approach,* IEEE Trans. Pattern Anal. Mach. Intell., Vol. 17, No. 2 (1995).

[21] D. Peng, B. Merriman, S. Osher, H.K. Zhao, and M. Kang, *A PDE-Based fast local level set method,* Journal of Computational Physics, Vol. 155 (1999), pp. 410ff.

[22] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, *An Improved Level Set Method for Incompressible Two-Phase Flows,* Computers and Fluids, Vol. 27, No. 5-6 (1998), pp. 663-680.

[23] H. Zhao, T. Chan, B. Merriman, and S. Osher, *A variational level set approach to multiphase motion,* Journal of Computational Physics, Vol. 127 (1996), pp. 179ff.

[24] D.L.S. McElwain and L.E. Morris, *Apoptosis as a Volume Loss Mechanism in Mathematical Models of Solid Tumor Growth,* Mathematical Biosciences, Vol. 39 (1978), pp. 147-157.

[25] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing,* 2nd Ed., Cambridge University Press, 1992. ISBN 0-521-43108-5

[26] J.A. Sethian, *Level Set Methods and Fast Marching Methods,* Cambridge University Press, New York, NY (1999). ISBN 0-521-64557-3

[27] Sussman, Mark and E. Fatemi, *An Efficient, Interface Preserving Level Set Re-Distancing Algorithm and its Application to Interfacial Incompressible Fluid Flow,* SIAM Journal on Scientific Computing, Vol. 20, No. 4 (1999), pp. 1165-1191.

[28] H. Zhao, T. Chen, B. Merriman, and S. Osher, *A Variational Level Set Approach to Multiphase Motion,* Journal of Computational Physics, Vol. 127, No. 1 (1996), pp. 179ff.

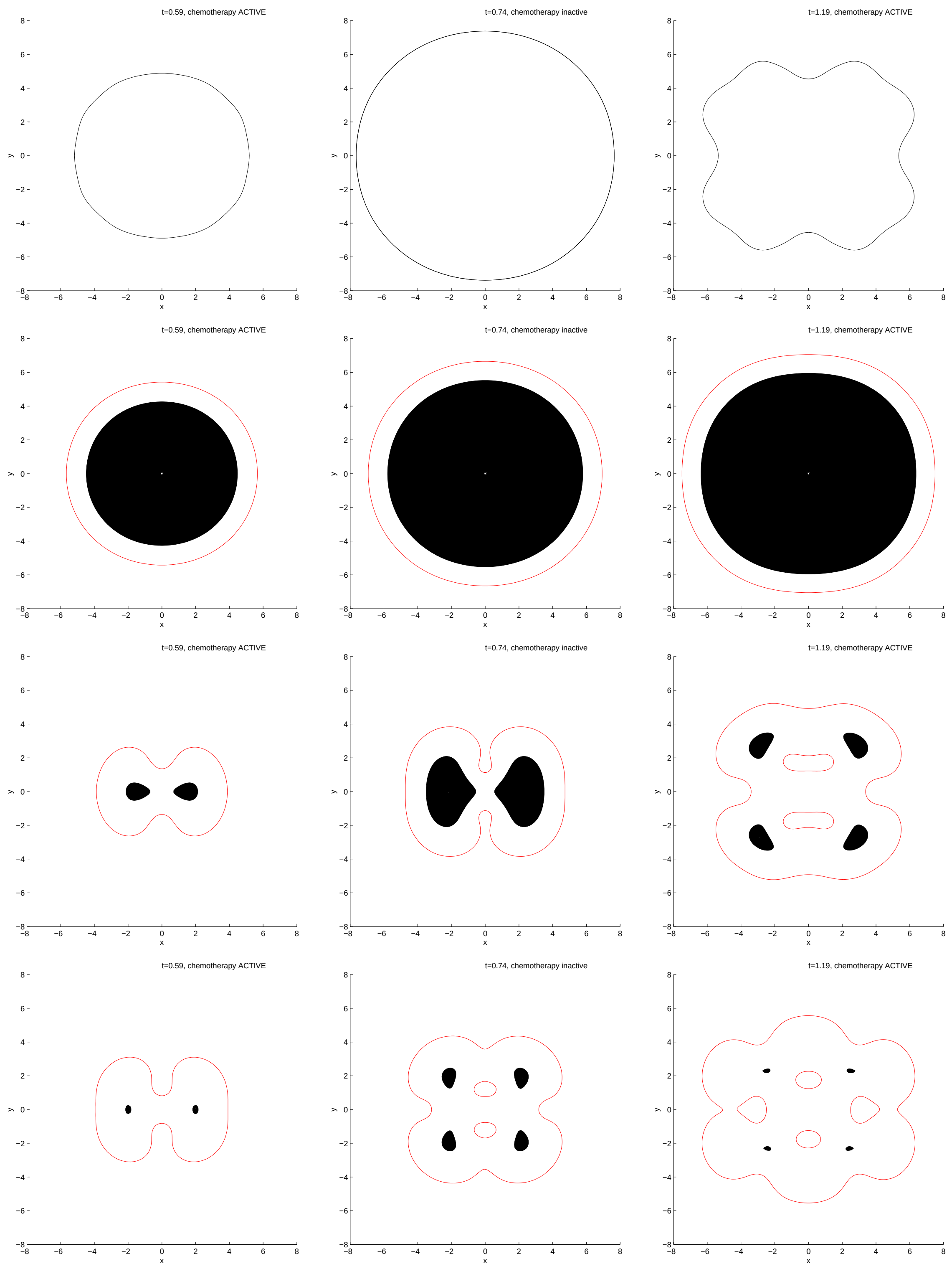

Figure 10: A Parameter Study on G_N with Chemotherapy: We plot for $N = 0$ (top row), $G_N = .1$ (second row), $G_N = 1$ (third row), and $G_N = 10$ (fourth row) at $t = 0.59$, $t = 0.74$, and $t = 1.19$ (from left to right).

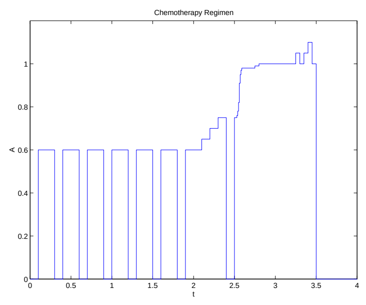

Figure 11: Chemotherapy regimen

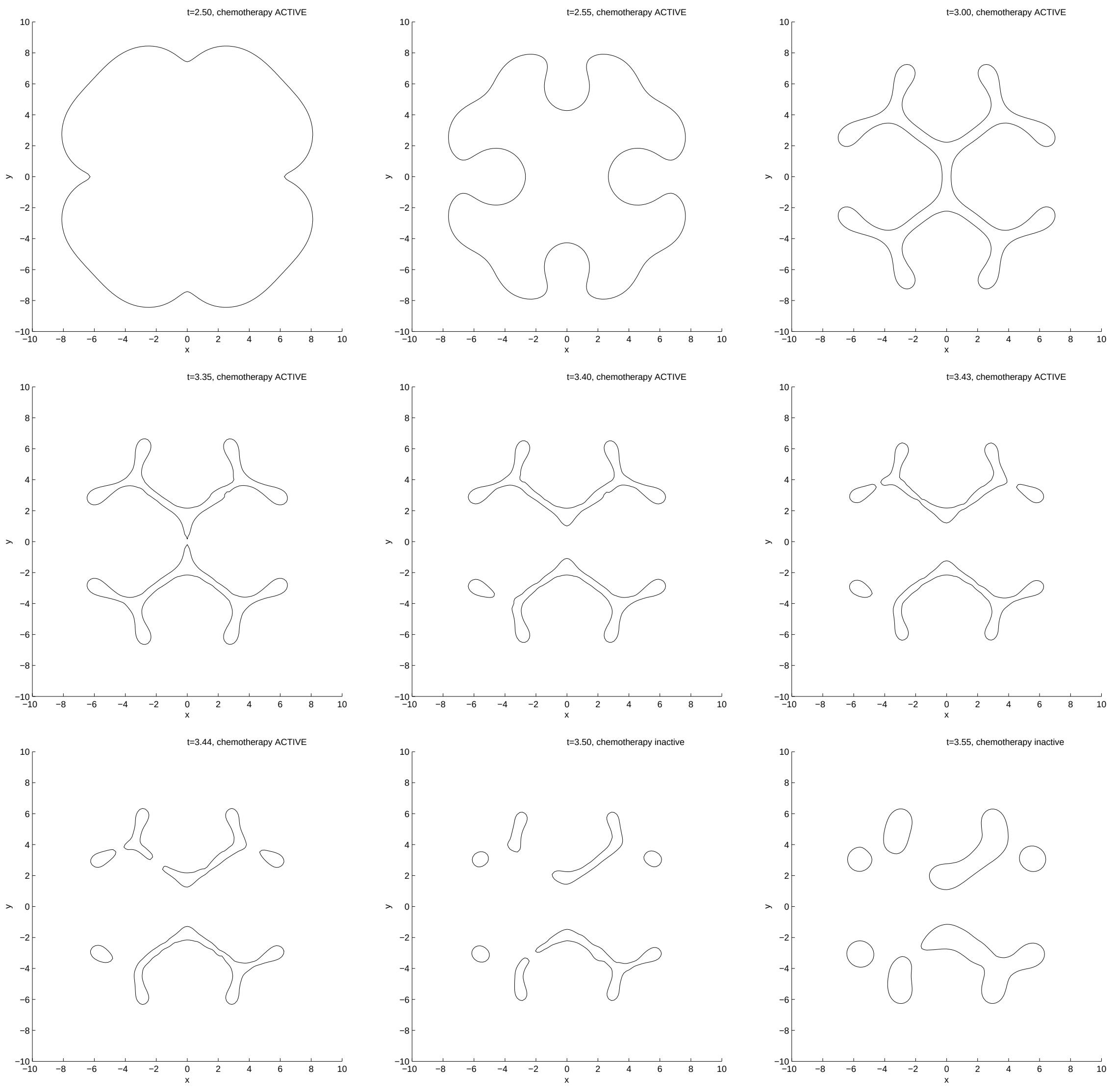

Figure 12: Breakup of a Tumor undergoing Chemotherapy: We plot (left to right from the top) at $t = 2.5$, $t = 2.55$, $t = 3.0$, $t = 3.35$, $t = 3.4$, $t = 3.43$, $t = 3.44$, $t = 3.5$, and $t = 3.55$.