Simulation Methods for Spatiotemporal Models of Biochemical Signaling Networks

Wanda Strychalski

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics.

Chapel Hill
2009

Approved by

Advisor: David Adalsteinsson
Advisor: Timothy C. Elston
Reader: M. Gregory Forest
Reader: Kenneth Jacobson
Reader: Laura A. Miller

©2009
Wanda Strychalski
ALL RIGHTS RESERVED

ABSTRACT

WANDA STRYCHALSKI: Simulation Methods for Spatiotemporal Models of Biochemical Signaling Networks

(Under the direction of David Adalsteinsson and Timothy C. Elston)

Cells use signaling networks consisting of multiple interacting proteins to respond to changes in their environment. In many situations, such as chemotaxis, spatial and temporal information must be transmitted through a signaling network. Recent computational studies have emphasized the importance of cellular geometry in signal transduction, but have been limited in their ability to accurately represent complex cell morphologies. We present a finite volume method that addresses this problem. Our method uses Cartesian cut cells in a differential algebraic formulation to handle the complex boundary dynamics encountered in biological systems. The method is second order in space and time. Several models of signaling systems are simulated in realistic cell morphologies obtained from live cell images. We then examine the effects of geometry on signal transduction.

External signals can trigger cells to polarize and move in a specific direction. During migration, spatially localized activity of proteins is maintained. To investigate the effects of morphological changes on intracellular signaling, we present a numerical scheme consisting of a cut cell finite volume spatial discretization coupled with level set methods to simulate the resulting advection-reaction-diffusion equation. We then show that shape deformations drive a Turing-type system into an unstable regime. The method is also applied to a model of a signaling network in a migrating fibroblast.

Determining the signaling mechanisms used by membrane proteins that interact with the cytoskeleton is important for understanding phenomena such as T-cell activation and viral infection. To investigate these interactions, recent experiments have tracked the movements of single lipids and glycosyl-phosphatidylinositol (GPI) anchored protein clusters tagged with 40 nm gold particles. These experiments reveal regions of transient confinement and transient anchorage of the particles. The distribution of transient anchorage release times exhibits a long tail. We developed a stochastic model of the system to explain the transient anchorage release times and the underlying biochemical reaction system.

ACKNOWLEDGMENTS

I sincerely thank David Adalsteinsson and Timothy Elston for their time, support, and enthusiasm in training me. I am often humbled by the challenges that inevitably arise in difficult problems, and Tim and David always have interesting and insightful ideas. I thank my committee members for their service. Greg Forest has been a father figure to me during my tenure at UNC. I appreciate his availability and advice whenever I needed guidance. I thank and recognize Ken Jacobson, Yun Chen, and John Fricks for their efforts in the transient anchorage project. I would like to acknowledge Meng Jin for help with modeling and for providing the substrate depletion model in Chapter 3.

My office and lab mates have been great colleagues and friends to me. Specifically, I would like to thank Jeanette Olli, Elizabeth Sell, Abby Todd, Andrea Richmond, Christina Hamlet, Marcelo Behar, Meng Jin, and Richard Allen. Other friends that have been very supportive are Amber Jackson, Kiran Khan, and Shilvi Shah. I thank my family for constantly supporting and encouraging me to achieve my goals.

Finally, I acknowledge my funding sources: NIH grants R01-GM079271 and R01-GM078994. I was also funded by the UNC Dissertation Completion Fellowship for the 2008-2009 academic year.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The cell is the fundamental unit of living matter. Even though the human body contains approximately 10^{13} different cell types [**4**], each cell contains the genetic information that encodes the structure of the entire organism. Each cell must perform basic tasks, such as growth, proliferation, procurement of nutrients, and response to environmental stimuli. These tasks are accomplished by protein molecules that undergo chemical reactions inside of the cell.

Changes in cellular behavior are triggered by environmental stimuli. For example, in animal cells, extracellular growth factors called mitogens are necessary for cell proliferation. Mitogens are an example of an extracellular signal. Conversion of a cellular signal into a response is called signal transduction. A signal transduction pathway is the cascade of biochemical reactions that leads to a cellular response. Defects in these pathways can result in diseases, such as cancer, diabetes, heart disease, and autoimmunity [**38, 39, 42**]. Therefore, understanding how intracellular signaling pathways function is not only a fundamental problem in cell biology, but also important for developing therapeutic strategies for treating disease.

The concentrations of proteins in these pathways change in response to transient signals as well as spatial localization. For example, an important messenger protein, cyclic adenosine monophosphate (cAMP), is found in many cell types and is involved in cellular processes such as regulation of the cell cycle, ion fluxes, and neurotransmission [**4, 78**]. The spatial distribution of cAMP is limited by phosphodiesterases [**78**]. Microscopy imaging clearly shows spatial localization of cAMP [**4**]. In this thesis, we present mathematical models for signaling networks with an emphasis on simulation methods for models of systems that vary in space and time.

1.1. Temporal signaling dynamics

When formulating a model of a signaling network, information about the biological system must be known. Wiring diagrams illustrating signaling networks typically contain many nodes and high connectivity. To further complicate matters, a cell uses redundant pathways to obtain tight regulation of intracellular systems. Information about the specific biological system being modeled is used to identify important network components and interactions. For example, in the 1980s key proteins that control events in the cell-cycle were identified through the study of temperature sensitive mutants of the yeast *Saccharomyces cerevisiae* [**4**]. Temporal oscillations in the concentration of cyclin and cyclin-dependent kinases (Cdks) regulate major events in the cycle. The dynamic behavior of these proteins can be captured by mathematical models (for example, [**17**]) that led to model predictions about specific control mechanisms. For the frog *Xenopus*, several cell-cycle model predictions from [**58**] were verified experimentally in [**73**].

Typically, biochemical species in a mathematical model are represented by concentration if there is a large number of molecules, and the role of noise in the system is negligible. Wiring diagrams are then translated into differential equations that describe the dynamics of chemical species over time. Many cellular processes are catalyzed by enzymes. Under certain assumptions, elementary reactions in these processes can be simplified into one equation described by nonlinear rate laws such as Michaelis-Menten [**39**]. The dynamical system describing a biological process is typically nonlinear and exhibits complex behaviors such as saturation, irreversible switches, toggle switches, oscillators, and adaptivity [**81**]. The system of differential equations can be integrated and analyzed with software packages such as `MATLAB`® or `SUNDIALS` [**28**]. Simulations results lead to model predictions and suggest experiments to further investigate the underlying biological process.

1.2. Spatiotemporal signaling dynamics

In many pathways, proper signal transduction requires that both the spatial and temporal dynamics of the system are tightly regulated [**34, 35, 57**]. For example, recent experiments have revealed spatial gradients of Rho family guanosine triphosphate (GTPase) protein activation in migrating cells [**36, 55, 62**]. Spatial localization of cAMP and calcium ions (Ca+) have also been visualized [**5, 13**]. Mathematical modeling is a tool to elucidate the control mechanisms in the regulation of spatiotemporal dynamics of signaling pathways. In [**25**] a spatiotemporal model was used to provide insight into the polarization profile of a protein involved in budding yeast cells. Several spatial models of protein networks involved in cell motility have been proposed [**18, 47**]. Polarization is important in determining the direction of migration and for gradient sensing. A simple model in [**52**] provides a mechanism for sustained polarity after a transient spatial signal. The model consists of a protein with an inactive and active form. Different diffusion rates and nonlinear positive feedback lead to signal amplification and the formation of a stable polarized concentration profile. The role of cell morphology and size on signaling proteins was investigated in [**50, 70**].

1.2.1. Computational methods for spatial systems. A complicating factor in simulating spatiotemporal models of cells is the addition of spatial terms, such as diffusion and advection. Typically, mathematical models that consider spatial effects are represented using partial differential equations (PDEs). However, recent work on incorporating diffusion with stochastic simulation methods should be noted [**19, 29**]. A numerical method for PDE models must be able to handle nonlinear reaction terms and nonlinear flux-based boundary conditions. Several spatial models of cell polarity make the simplifying assumption the cell is one-dimensional [**18, 52, 60**]. These studies provide useful information about generating and maintaining a polarized concentration profile, but ignore geometric effects. Other recent computational studies emphasize the importance of cellular geometry in signaling networks [**50, 57, 70**]. For computational simplicity, many of these investigations assume idealized two or three dimensional cell geometries [**25, 41, 50**], whereas others approximate irregularly shaped cells using a "staircase" representation of the cell membrane [**69**].

Cartesian grid-based methods have been used to obtain numerical solutions to PDEs with irregular domains and interfaces. The immersed boundary method was developed to solve fluid-structure interaction problems in biological fluids [**63**]. Discrete delta functions are used to describe the distribution of forces onto nearby grid points. One drawback to this approach is that the accuracy of the method is linked to the choice of approximate delta function. Typically, explicit time updates with severe time step restrictions are implemented to integrate immersed boundary discretizations, though recent work has been proposed to overcome this challenge [**53**]. In the immersed interface method, jump conditions across interfaces given in a local coordinate system are used to handle boundary conditions on irregular geometries [**43**]. This method has been successful in solving many interface problems such as Hele-Shaw flow, the Stefan problem, and the Navier-Stokes equations. Open problems for this method include conservative finite difference schemes for elliptic and parabolic problems. Another challenge arises when the jump conditions are nonlinear. Additionally, implementing a three dimensional immersed interface method for a moving boundary problem is difficult. Embedded boundary methods [**15, 16, 32, 45, 48, 71**] have been used to solve Poisson's equation [**32**] and the heat equation [**48, 71**] with homogeneous Dirichlet and Neumann boundary conditions as well as hyperbolic conservation laws [**16**]. The temporal update for the heat equation in [**48**] is an implicit Runge-Kutta method [**80**]. Modifications in the method are needed to handle nonlinear terms.

Both finite element and finite volume methods have been used to simulate spatial models of biochemical reaction networks [**50, 69, 70, 83**]. The most common finite volume algorithm to simulate models of reaction networks in two and three dimensions is the Virtual Cell algorithm [**69**]. Cellular geometries are represented by staircase curves (Fig. 1.1). The authors note that the approximation of fluxes across membranes leads to

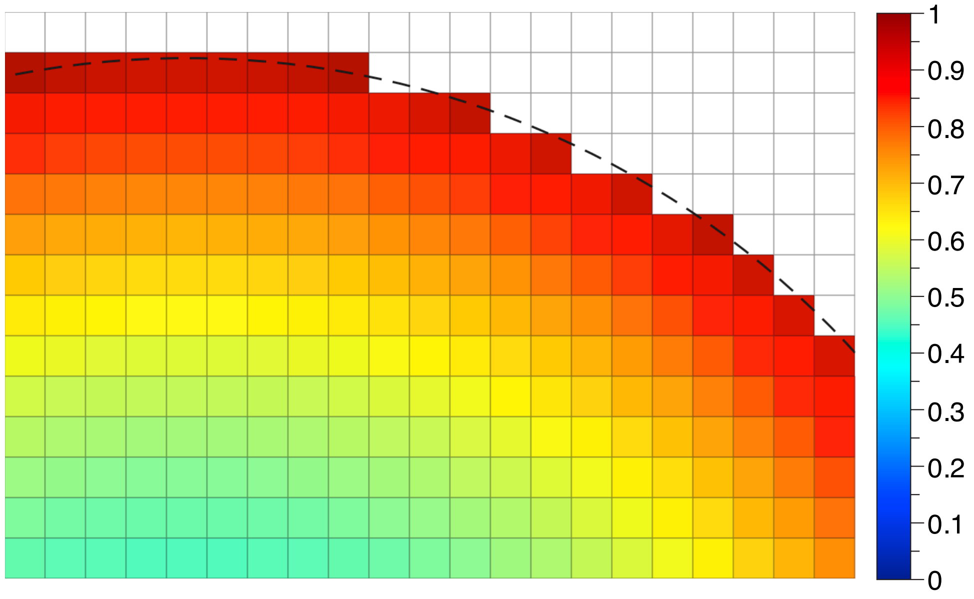


FIGURE 1.1. Boundary approximation taken from a Virtual Cell simulation. The dashed line indicates the geometry specified in the Virtual Cell program. The color ramp is the intensity of the grayscale component from a simulation.

a decrease in the spatial accuracy of the numerical method to first order. The temporal accuracy of algorithm in [**69**] is also limited to first order. For finite element methods, which typically require a triangulation of the computational domain, grid generation can be a challenge. This is particularly problematic if the boundaries of the computational domain are moving.

Few simulation methods exist for moving boundary problems coupled with intracellular dynamics. When solving moving boundary problems with finite element methods, mesh generation is a challenging problem. In [**66**] a physical model of a motile cell was simulated with finite element methods by triangulating every time step. The algorithm includes a step of interpolation to ensure boundary points are equally spaced. The simulations presented in [**66**] are feasible because a course grid with several hundred elements was used. The Cellular Potts Model (CPM) [**24**], which is a type of cellular automata, has been used to simulate a motile cell [**47**]. The dynamics in the CPM are based on minimizing a Hamiltonian with the Metropolis algorithm [**49**]. The CPM includes a system temperature that represents fluctuations in the system. Though this type of model can produce qualitatively realistic data, it is unclear what quantitative data can be obtained from simulations. Another way to simulate a motile cell is by a one-dimensional model in [**18**]. New methods are needed to simulate signaling networks and biophysical models of cellular systems in moving geometries.

1.3. Thesis overview

A finite volume method to simulate models of spatiotemporal reaction networks on arbitrary stationary geometries is presented in Chapter 2. The method accurately represents complex boundaries and utilizes a Cartesian grid. Our numerical scheme is based on a cut cell method that represents the cell boundary using a piecewise-linear approximation. The method presented here extends the results on embedded boundary methods [**15, 16, 32, 45, 48, 71**] to systems of nonlinear reaction diffusion equations with arbitrary boundary conditions. In contrast to previous embedded boundary methods, we also offer an alternative formulation to handle the temporal update. In our formulation, the boundary conditions form a system of nonlinear algebraic equations that can be solved with existing differential algebraic equation solvers. We provide a novel use of DASPK (Differential Algebraic Solver Pack) [**10**] as a time integrator for the finite volume method. The embedded boundary spatial discretization combined with the differential algebraic formulation allows us to achieve second order accuracy in space and time. Our method also provides an appropriate framework for addressing moving boundary problems using level set methods [**59, 72**].

The numerical method in Chapter 3 extends the finite volume method to moving boundaries. We propose a novel method for simulating models of biochemical reaction networks in moving cell morphologies. The method solves systems of advection-reaction-diffusion equations on an underlying Cartesian grid. The cell geometry is embedded with a signed distance function and updated with level set methods [**59, 72**]. The operators are separated into an advection term and a combined reaction-diffusion term. Reaction-diffusion terms are updated with an implicit differential-algebraic formulation from Chapter 2. Advection terms are treated with the same spatial discretization used in level set methods.

In Chapter 4, we present a mathematical model to describe the behavior of a type of protein diffusing on the cell membrane. The proteins, called glycosyl-phosphatidylinositol-anchored proteins, were cross-linked to a gold particle for tracking purposes. These particles exhibited periods of no visible displacement, called transient anchorage. Interestingly, the distribution of release times from an anchored state exhibited a longer tail than could be explained with a single exponential distribution [**14**]. To explain this behavior, we developed a stochastic model and compared our results to the proposed chemical linkage scheme to the cytoskeleton. An algorithm to simulate diffusion with transient anchorage was also developed.

Chapter 5 summarizes the work presented here. Future research directions are discussed.

CHAPTER 2

A Cut Cell Method for Simulating Spatial Models of Biochemical Reaction Networks in Arbitrary Geometries

In this chapter we present a method to simulate models of signaling networks in complex geometries. Our method includes a finite volume discretization based on a Cartesian grid in two dimensions. A differential algebraic formulation is used to handle the complex boundary dynamics and nonlinearities encountered in biological systems. The method is second order in space and time. Several models of signaling systems are simulated in realistic cell morphologies obtained from live cell images to demonstrate the method.

The models presented in this chapter and in Chapter 3 lack specific biologically relevant parameter values due to the difficulty in obtaining data for reaction rates and diffusion coefficients. Diffusion coefficients depend on cell type and location. Because of the large variation in parameter estimates, the models are simulated on a unit box with parameters chosen to highlight the influence of spatial terms on concentration profiles.

2.1. Mathematical formulation

Spatial models of biochemical reaction networks are typically represented using partial differential equations consisting of reaction and diffusion terms. Active transport, driven by molecular motors, also occurs within cells. This effect can be included in our numerical scheme by the use of advection terms and will be addressed in the next chapter. For simplicity we restrict ourselves to two spatial dimensions x and y. For a given chemical species, the reaction terms encompass processes such as activation, degradation, protein modifications and the formation of molecular complexes. These reactions typically include nonlinear terms, such as those arising from Michaelis-Menten kinetics. In a system consisting of n chemical species, the concentration of the i^{th} species c_i evolves in space and time according to the following equation:

$$\frac{\partial c_i}{\partial t} = -\nabla \cdot \mathbf{J} + f_i(\boldsymbol{c}), \tag{2.1}$$

where $\mathbf{J} = -D_i \nabla c_i$ is the flux density, D_i is the diffusion coefficient, and the function $f_i(\boldsymbol{c})$ models the reactions within the cell that affect c_i. The elements of the vector $\boldsymbol{c}$ are the concentrations of the n chemical species. Reactions also may occur on the cell membrane yielding nonlinear conditions on the boundary $\partial\Omega$,

$$-D\vec{n} \cdot \nabla c_i|_{\partial\Omega} + g(\boldsymbol{c})|_{\partial\Omega} = 0. \tag{2.2}$$

Eqs. (2.1) and (2.2) are solved subject to appropriate initial conditions $c_i(x, y, 0)$ for each species in the system.

2.2. Numerical methods

Our goal is to develop a simulation tool that can accurately and efficiently solve spatial models of signaling and regulatory pathways in realistic cellular geometries. We obtain the computational domain from live-cell images. The model equations are solved on a Cartesian grid by discretizing the Laplacian operator, which models molecular diffusion, using a finite volume method.

2.2.1. Computational domain.

2.2.1. Computational domain. Fig. 2.1 shows a grayscale image of a mouse fibroblast [**62**]. Because the original image is noisy, the image was smoothed by convolving it twice with the standard five point Gaussian smoothing filter. After smoothing, a suitable thresholding value was picked, and the front was computed by an iso-contour finder. A signed distance function is constructed with the smoothened boundary using fast marching methods [**46**]. The zero level set of the signed distance function yields piecewise linear segments used to define cut cells (Fig. 2.2).

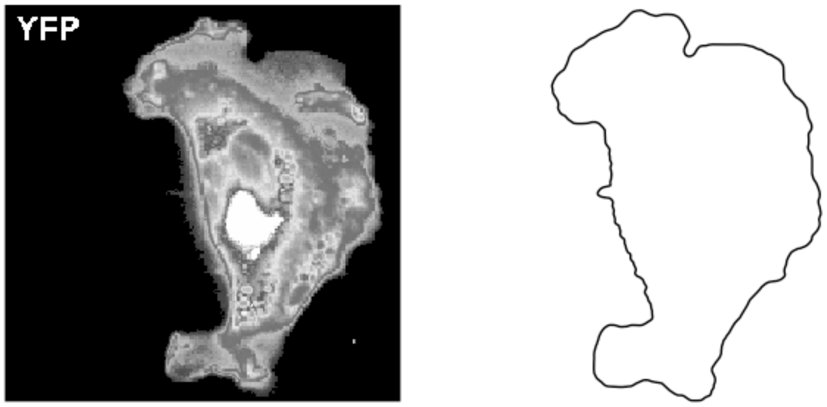


FIGURE 2.1. Grayscale image of a mouse fibroblast taken from supplemental data in [**62**] (left) and the smoothened boundary (right). Reprinted with permission from Macmillan Publishers Ltd: Nature [**62**], copyright ©2006.

2.2.2. Discretization of the spatial operator. We utilize a Cartesian grid-based, finite volume algorithm originally presented in [**32**] to discretize the diffusion operator arising from Eq. (2.1). Finite volume methods store the average value of the concentration over a computational grid cell at the location (i, j). That is,

$$\bar{c}_{i,j} = \frac{1}{V_{i,j}} \iint_{V_{i,j}} c(x, y) dV, \tag{2.3}$$

where $V_{i,j}$ is the volume of the (i, j) grid cell. Inserting Eq. (2.3) into Eq. (2.1) produces

$$\frac{\partial \bar{c}_{i,j}}{\partial t} - \overline{f(c)}_{i,j} = -\frac{1}{V_{i,j}} \iint_{V_{i,j}} \nabla \cdot \mathbf{J} dV. \tag{2.4}$$

The divergence theorem allows us to convert the above volume integral into a surface integral,

$$\frac{\partial \bar{c}_{i,j}}{\partial t} - \overline{f(c)}_{i,j} = -\frac{1}{V_{i,j}} \int_{\partial V_{i,j}} (\mathbf{J} \cdot \vec{n}) \, dS. \tag{2.5}$$

For interior grid cells, we have

$$\frac{\partial \bar{c}_{i,j}}{\partial t} - \overline{f(c)}_{i,j} = -\frac{1}{V_{i,j}}\left[\int_{y_{j-1/2}}^{y_{j+1/2}} \big(J_x(x_{i+1/2},y) - J_x(x_{i-1/2},y)\big)\,dy \right.$$
$$\left. + \int_{x_{i-1/2}}^{x_{i+1/2}} \big(J_y(x,y_{j+1/2}) - J_y(x,y_{j-1/2})\big)\,dx\right], \tag{2.6}$$

where $J_x = -D\frac{\partial c}{\partial x}$ and $J_y = -D\frac{\partial c}{\partial y}$. Approximation of the integrals in Eq. (2.6) with the midpoint rule yields

$$\frac{\partial c_{i,j}}{\partial t} - f(c_{i,j}) \approx -\frac{1}{V_{i,j}}\left[\Delta y\left(J_x(x_{i+1/2},y_j) - J_x(x_{i-1/2},y_j)\right)\right.$$
$$\left. + \Delta x\left(J_y(x_i,y_{j+1/2}) - J_y(x_i,y_{j-1/2})\right)\right]. \tag{2.7}$$

By approximating the gradient terms with centered differences, we arrive at the standard five point Laplacian. Therefore in computational grid cells with volume $V_{i,j} = 1$, the finite volume stencil is the same as the the five point Laplacian approximation.

The cut cell method generalizes as follows. The boundary of the computational domain is approximated as a piecewise linear segments (Fig. 2.2, dashed line), and grid cells that the boundary passes through are referred to as cut cells. To calculate the volume of a cut cell, we apply the divergence theorem with the vector field $\mathbf{F} = (x/2, y/2)$ (note that $\nabla \cdot \mathbf{F} = 1$),

$$V_{i,j} = \iint_{V_{i,j}} (\nabla \cdot \mathbf{F})\,dV = \int_{\partial V_{i,j}} \mathbf{F}\cdot \vec{n}\,dS, \tag{2.8}$$

where $\vec{n}$ is the unit normal vector to the surface. A cut cell can have up to five line segments where the above surface integral must be computed. The volume of a cut cell is computed by recasting the volume integral as a boundary integral,

$$V_{i,j} = \iint_{V_{i,j}} dV = \iint_{V_{i,j}} \nabla\cdot\left(\frac{x}{2},\frac{y}{2}\right) dV = \int_{\partial V_{i,j}} \left(\left(\frac{x}{2},\frac{y}{2}\right)\cdot\vec{n}\right) dS, \tag{2.9}$$

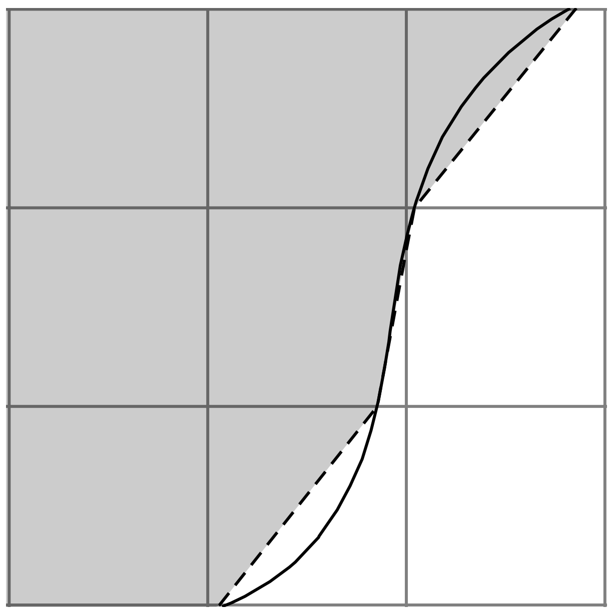


FIGURE 2.2. Computational boundary (dashed line) with an assumed higher order representation of the cell boundary drawn as a solid line.

where the integral on the right can be computed exactly for the polygon. Each segment is evaluated, then summed. The center of mass can also be computed using a boundary integral. For example,

$$\iint_{V_{ij}} x dV = \iint_{V_{ij}} \nabla \cdot \left(\frac{x^2}{2}, 0\right) dV = \int_{\partial V_{i,j}} \left(\left(\frac{x^2}{2}, 0\right) \cdot \vec{n}\right) dS. \tag{2.10}$$

The parameterization for a linear path that begins at (x_0, y_0) and ends at (x_1, y_1) is $\mathbf{r}(t) = (x_0 + t(x_1 - x_0), y_0 + t(y_1 - y_0))$ for $t \in [0, 1]$. The surface integral over one face is

$$\int_0^1 \mathbf{F}(\mathbf{r}(t)) \cdot \vec{n} |\mathbf{r}'(t)| dt. \tag{2.11}$$

A cut cell contains either a right triangle or a rectangle. In the case where a cut cell contains a triangle, the cell contains normal sides a and b. The outward normal to the hypotenuse is $\frac{1}{\sqrt{a^2+b^2}}(\pm a, \pm b)$. The sign on each component of the normal vector depends on the cut cell's configuration. For example, in Fig. 2.3 the x-component of the normal is positive, and the y-component is negative. In the case that a cut cell is rectangular, the normal is simply $(\pm 1, 0)$ or $(0, \pm 1)$. To approximate the average of the function $c(x, y)$ over a cut cell, $c(x, y)$ is evaluated at the cell's center of mass. The function value at the centroid is used to initialize at cut cell grid points as in [**32, 48**]. The x-component of the centroid is computed by evaluating the integral

$$\iint_{V_{i,j}} x dV, \tag{2.12}$$

which is calculated as a surface integral using the divergence theorem with the vector field $\mathbf{F} = (x^2/4, xy/2)$. The y-component of the centroid is computed in a similar fashion.

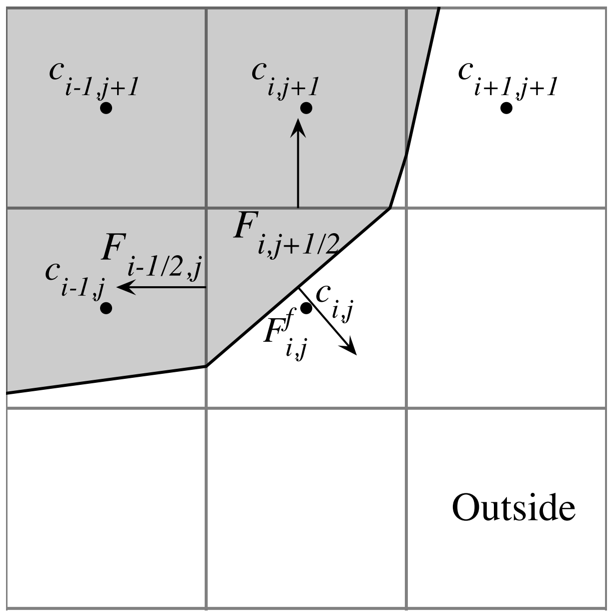


FIGURE 2.3. Diagram of fluxes for cut cells where shaded boxes indicate cells that are inside the boundary.

Next, we construct the integral on the right side of Eq. (2.5) for a cut cell. In general, there are up to five surface integrals to approximate. Let $a_{l,m} \in [0, 1]$ represent the fraction of each of the four cell edges covered by the cut cell and a_f be the length of the line segment representing the boundary. Then Eq. (2.7) becomes

$$\begin{aligned}\frac{\partial c_{i,j}}{\partial t} - f(c_{i,j}) \approx -\frac{1}{V_{i,j}}\Big[&\Delta y\left(a_{i+1/2,j}J_x(x_{c_{i+1/2}}, y_j) - a_{i-1/2,j}J_x(x_{c_{i-1/2}}, y_j)\right)\\ &+ \Delta x\left(a_{i,j+1/2}J_y(x_i, y_{c_{j+1/2}}) - a_{i,j-1/2}J_y(x_i, y_{c_{j-1/2}})\right)\\ &+ a^f J_f\Big]\,.\end{aligned} \tag{2.13}$$

The points $(x_{c_{i\pm1/2}}, y_j)$ and $(x_i, y_{c_{j\pm1/2}})$ are the coordinates corresponding to the midpoint of a cut face. Let $F_{i\pm1/2,j} = -a_{i\pm1/2,j}\Delta y J_x(x_{c_{i\pm1/2}}, y_j)$ and $F_{i,j\pm1/2} = -a_{i,j\pm1/2}\Delta x J_y(x_i, y_{c_{j\pm1/2}})$. With this notation, we rewrite the previous equation as

$$\frac{\partial c_{i,j}}{\partial t} - f(c_{i,j}) \approx \frac{1}{V_{i,j}}\left(F_{i+1/2,j} - F_{i-1/2,j} + F_{i,j+1/2} - F_{i,j-1/2} - F^f_{i,j}\right). \tag{2.14}$$

We refer to the F's as the surface fluxes (Fig. 2.3). On a full edge with $a_{l,m} = 1$ the surface flux is calculated with centered differences. For example, in Fig. 2.3, we have

$$F_{i-1/2,j+1} = D\Delta y\frac{c_{i,j+1} - c_{i-1,j+1}}{\Delta x}. \tag{2.15}$$

The flux gradient across a cut edge, e.g. $(x_{i-1/2}, y_j)$, is approximated by a linear interpolation of two gradients, which are computed by centered differences. A linear interpolation formula between two points y_1 and y_2 as a function of a parameter $\mu \in [0,1]$ is

$$y^I = (1-\mu)y_1 + \mu y_2. \tag{2.16}$$

In the case of a cut cell edge, $\mu = (1 + a_{l,m})/2$. For example, to construct $F_{i-1/2,j}$ in Fig. 2.4, the gradient at $(x_{i-1/2}, y_j)$ and $(x_{i-1/2}, y_{j+1})$ is used,

$$\begin{aligned}F_{i-1/2,j} =& D a_{i-1/2,j}\Delta y\left[\frac{(1 + a_{i-1/2,j})}{2}\frac{(c_{i,j} - c_{i-1,j})}{\Delta x} + \right.\\ &\left.\frac{(1 - a_{i-1/2,j})}{2}\frac{(c_{i,j+1} - c_{i-1,j+1})}{\Delta x}\right].\end{aligned} \tag{2.17}$$

To calculate the flux through a boundary, e.g. $F^f_{i,j}$, we compute the gradient along

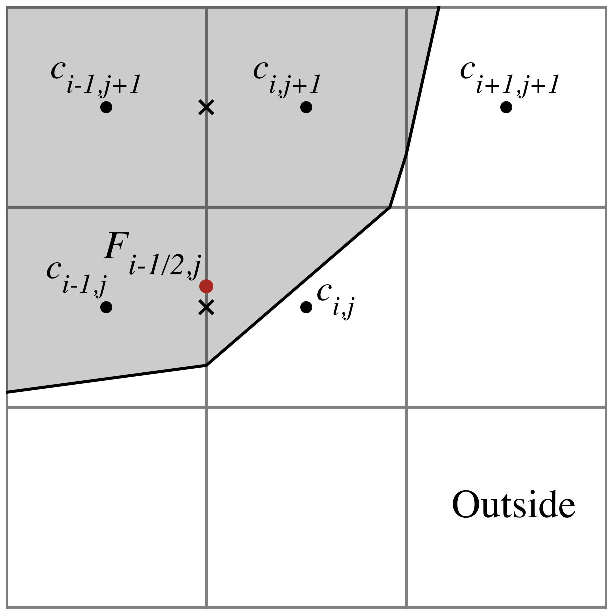


FIGURE 2.4. Gradient interpolation diagram for a partially cut face. The cross indicates points used to interpolate the gradient at the point labeled with a red circle.

a line normal to the boundary, centered at the boundary midpoint. To find function values on the normal line, we interpolate using three equally spaced cell centered points (Fig. 2.5). If the normal line is oriented with an angle of $\pi/4 < |\theta| < 3\pi/4$ relative to the horizontal grid lines, horizontal grid points are used to compute the values on the line. Otherwise vertical points are used. The two points computed along the normal line and the value on the boundary are then used to construct a quadratic polynomial. The concentration gradient is calculated by differentiating the quadratic polynomial and evaluating the result at the boundary point c^f,

$$G^f = \frac{1}{d_2 - d_1}\left[\frac{d_2}{d_1}(c^f - c_1^I) - \frac{d_1}{d_2}(c^f - c_2^I)\right], \tag{2.18}$$

where c_1^I and c_2^I are the interpolated values along the normal line and d_1 and d_2, respectively, are the distances of these two points from the boundary. The flux F_{ij}^f in Eq. (2.14) is calculated by multiplying G^f by the area of the cut cell edge a_f and the diffusion coefficient D. The discretization of the boundary condition Eq. (2.2) becomes the algebraic

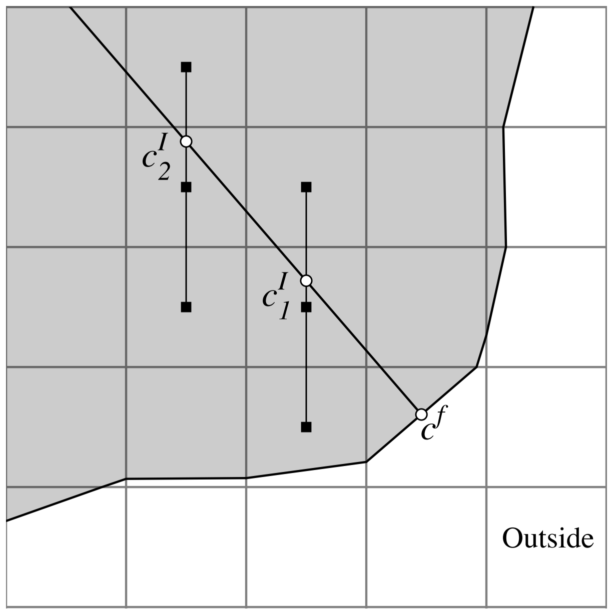


FIGURE 2.5. Gradient interpolation diagram to obtain the flux through a boundary. Circles indicate interpolated values that depend on the grid-based values.

equation

$$DG^f + g(\boldsymbol{c}^f) = 0. \tag{2.19}$$

Because all gradients are constructed with second order methods, the overall discretization scheme is second order in space. Further discussion on the accuracy of the spatial discretization scheme can be found in [**32**].

2.2.3. Time discretization.

Spatial discretizations of Eqs. (2.1) and (2.2) are treated as a differential-algebraic system of nonlinear equations (DAE). The general form for a differential-algebraic system is

$$F(t, \boldsymbol{C}, \boldsymbol{C}') = \boldsymbol{0}, \tag{2.20}$$

where $\boldsymbol{C}$ is an $(N_g + N_b) \times 1$ vector. The first N_g entries are associated with Cartesian grid based values in the differential-algebraic system from the discretization of Eq. (2.1) for the chemical species concentrations. These entries have an explicit time derivative term. The N_b remaining entries arise from discretizing the boundary conditions given in Eq. (2.2) that form algebraic constraints. As noted in [**6**], reformulating algebraic constraints in a nonlinear model as a system of ordinary differential equations may be time consuming or impossible. DAEs formed by reaction-diffusion equations described in section 2 are semi-explicit, index-1 systems of the form

$$\begin{aligned} \boldsymbol{C}_1' &= F_1(\boldsymbol{C}_1, \boldsymbol{C}_2, t) \\ \mathbf{0} &= F_2(\boldsymbol{C}_1, \boldsymbol{C}_2, t). \end{aligned} \tag{2.21}$$

$\boldsymbol{C}_1$ represents the first N_g variables and $\boldsymbol{C}_2$ represents the remaining N_b variables. Eq. (2.21) is an index-1 system if and only if $\partial F_2/\partial \boldsymbol{C}_2$ is nonsingular [**6**]. Ordinary differential equations are index-0.

We use the DASPK solver described in [**10**] as a time integrator for our differential algebraic system. In DAPSK, backward differentiation formulas (BDF) discretize the time derivative in Eq. (2.20). A basic implicit method with a backward Euler time discretization of Eq. (2.20) is given by,

$$F\left(t^{n+1}, \boldsymbol{C}^{n+1}, \frac{\boldsymbol{C}^{n+1} - \boldsymbol{C}^{n}}{\Delta t}\right) = \mathbf{0}, \tag{2.22}$$

where n is defined such that $t^n = n\Delta t$. Newton's method can be used to solve the resulting nonlinear equations for $\boldsymbol{C}^{n+1}$,

$$\boldsymbol{C}^{n+1}_{m+1} = \boldsymbol{C}^{n+1}_{m} - \left(\frac{\partial F}{\partial \boldsymbol{C}} + \frac{1}{\Delta t}\frac{\partial F}{\partial \boldsymbol{C}'}\right)\Big|^{-1}_{\boldsymbol{C}^{n+1}_{m}} F\left(t^{n+1}, \boldsymbol{C}^{n+1}_{m}, \frac{\boldsymbol{C}^{n+1}_{m} - \boldsymbol{C}^{n}}{\Delta t}\right), \tag{2.23}$$

where m is the index of the Newton iteration. In order to achieve higher order temporal accuracy, a higher order interpolating polynomial is used to approximate the time derivative.

In a k-step BDF, the time derivative is replaced by the derivative of an interpolating polynomial at $k+1$ times $t^{n+1}, t^n \ldots, t^{n+1-k}$ evaluated at t^{n+1}. If we approximate the derivative using a k^{th} order stencil using k known values and the implicit value C^{n+1} we get

$$\boldsymbol{C}'^{n+1} \approx \frac{1}{\Delta t}\left(\alpha_0 \boldsymbol{C}^{n+1} + \sum_{i=1}^{k} \alpha_i \boldsymbol{C}^{n+1-i}\right). \tag{2.24}$$

The coefficients of the BDF are given by α_i's. In DAPSK, these values are coefficients of the Newton divided difference interpolating polynomial [**6**]. The default order of the BDF method in DASPK is five. The new implicit equation to be solved at each time step is

$$F\left(t^{n+1}, \boldsymbol{C}^{n+1}, \frac{1}{\Delta t}\left(\alpha_0 \boldsymbol{C}^{n+1} + \sum_{i=1}^{k} \alpha_i \boldsymbol{C}^{n+1-i}\right)\right) = \mathbf{0}. \tag{2.25}$$

Eq. (2.25) can be written as

$$F(t^{n+1}, \boldsymbol{C}^{n+1}, \frac{\alpha_0}{\Delta t}\boldsymbol{C}^{n+1} + \mathbf{v}) = \mathbf{0}, \tag{2.26}$$

where $\mathbf{v}$ is a vector that depends on previously computed time values. Details of choosing stepsize, starting selection and variable order strategies are found in [**6**]. The nonlinear system is solved with a modified Newton's method, given by

$$\boldsymbol{C}^{n+1}_{m+1} = \boldsymbol{C}^{n}_{m} - \zeta\left(\frac{\partial F}{\partial \boldsymbol{C}} + \frac{\alpha_0}{\Delta t}\frac{\partial F}{\partial \boldsymbol{C}'}\right)\Bigg|^{-1}_{\boldsymbol{C}^{n+1}_{m}} F(t^{n+1}, \boldsymbol{C}^{n+1}_{m}, \frac{\alpha_0}{\Delta t}\boldsymbol{C}^{n+1}_{m} + \mathbf{v}), \tag{2.27}$$

where ζ is a constant chosen to speed up convergence and m is the iteration index. Each step of the Newton iteration requires inverting the matrix

$$A = \frac{\partial F}{\partial \boldsymbol{C}} + \frac{\alpha_0}{\Delta t}\frac{\partial F}{\partial \boldsymbol{C}'}. \tag{2.28}$$

We store this matrix in sparse triple format, and use routines from SPARSKIT [**67**] to solve the linear system iteratively. The Generalized Minimal Residual (GMRES) method [**68**] with an incomplete LU (ILU) preconditioner is used to solve the linear system.

2.3. Software implementation

The software is implemented in the `C, C++` and `Fortran` programming languages. `Triangle` files from [**74**] were written in `C`, and a `C++` wrapper class was developed to create a triangular grid object. DASPK source code consisted of a `Fortran` file, which we compiled into a library. The functions needed to use DASPK were declared gobally and compiled using a `Fortran` to `C` library. The rest of the software is written in `C++` with `Xcode`® developing software. Data structures from `DTSource` [**1**] are used for data storage and access. `DataTank` is the software harness that manages communication between user input and output as well as data visualization. The input of the program includes a 2D boundary (`DTPath2D`), edge-centered grid (`DTMesh2DGrid`), diffusion coefficients (`DTDoubleArray`), reaction coefficients (`DTDoubleArray`), initial conditions for each species (`DTFunction2D` or `DTMesh2D`), time step, end time for the simulation, and the frequency of saving the output from the program.

The class `CutCellCompartment` manages spatial information associated with a compartment. The constructor is initialized with a boundary (`DTPath2D`), edge-centered grid (`DTMesh2DGrid`), and signed distance function (`DTMutableDoubleMesh2D`). The class stores the spatial location of the row-stacked unknowns. A boundary list stores the unknowns that are cut cells. Back pointer arrays are stored in the class to link the boundary unknowns to (i, j) grid locations. This information is necessary to build the Jacobian matrix for the solver. The class stores boundary points, the zero level set that defines cut cells, and centroids. The finite volume stencil requires the volume of each computational grid cell, the area of each cut face, interpolation points, and the centroid of the piecewise linear approximation of the boundary. This data are also stored in the `CutCellCompartment` class. A triangulated grid is a member of this class.

The class `CombinedSolver` builds sparse matrices for DASPK. The class contains a list of `CutCellCompartment` objects and an integer array with the number of unknowns in each compartment. Two sparse matrices are stored in this class. The first is the Jacobian that includes diffusion coefficients, reaction coefficients, a scaling coefficient from DASPK, and and a stacked array with the current state of the system. The second matrix computes the algebraic portion of the system. Sparse-matrix multiplication is used to apply this part of the operator to the system in the residual function in DASPK.

The `State` class manages and organizes the raw data. DASPK uses pointers to arrays for input and output. A `State` object stores the state of the system in an array at the current simulation time. If a mesh version of the data is requested for output, the `State` returns the object. This class interfaces with DASPK to update the system. `State` is initialized with a list of `CutCellCompartments`, number of species in each compartment, diffusion coefficients, and reaction coefficients. The `State` contains a global `CombinedSolver` object that is initialized in the constructor. Parameters for DASPK are class members and initialized in the constructor.

DASPK requires the user to implement three functions: the residual, Jacobian, and an implementation of the preconditioner. Because these functions are defined in `ddaspk.f` from DASPK, they are not a part of a class. We include their implementation in `State`. The most computationally expensive part of the code is the residual function. We implemented a fast Laplacian operator by using offsets created and stored in `CutCellCompartment`. The boundary operator is applied by a sparse matrix multiply computed in `CombinedSolver`. In the preconditioner function, the incomplete LU factorization is performed with SPARSEKIT routines [**67**]. A sparse matrix structure, `SPTriple` was created to take a sparse coordinate format of the Jacobian matrix, convert it to compressed sparse row format, and create the ILU factorization. The `SPTriple` instance in `State` must be a global variable because the Jacobian function and the preconditioner need access to the factorization.

A summary of class interactions in shown in Fig. 2.6. In the main routine, a list of `CutCellCompartments` is created. The number of species in each compartment is specified. A `State` object is created with this information and other user input from `DataTank`. In the main routine, a loop over time contains a function call to the `State` object to update itself, and output is saved.

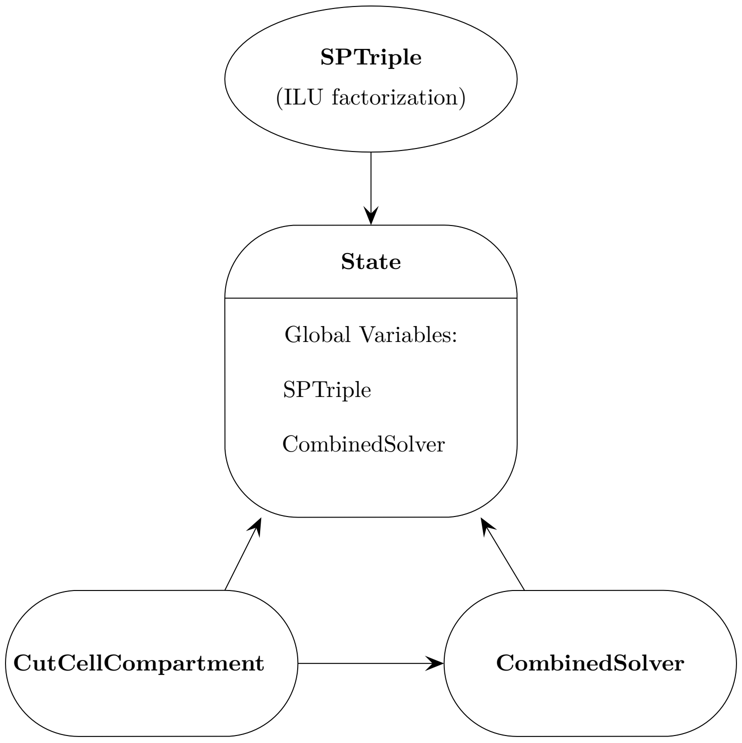


FIGURE 2.6. Schematic of class interactions. A rounded box indicates a class. An ellipse indicates a structure. An arrow pointing from one class or structure to another means the class with the arrow pointing to it contains an object or structure of the other one's type.

2.4. Convergence tests

In order to demonstrate the accuracy of our method, convergence is tested by comparing against an exact solution on a circular domain containing all types of cut cells. The exact solution to the diffusion equation with a zero Dirichlet boundary condition can be found in terms of Bessel functions. Let λ denote the first root of the Bessel function $J_0(x)$, and r be the radius of the circle. Then the expression:

$$f(x,y,t) = \exp\left(-D\left(\frac{\lambda}{r}\right)^2 t\right) J_0\left(\lambda\frac{\sqrt{(x-0.5)^2+(y-0.5)^2}}{r}\right) \tag{2.29}$$

is an exact solution to the diffusion equation. Fig. 2.7(a) shows the initial condition and Fig. 2.7(b) shows the computed solution at $t = 0.2$ using the constants $D = 0.05$ and

$r = 0.3$. For visualization purposes, the computational domain and boundary points are triangulated with Triangle [**74**]. For this example, the error is computed as the difference between computed solution values on a triangular grid subtracted from the exact solution. The grids for both two dimensional triangular meshes were the same. For purposes of generating the following convergence data, the spatial steps Δx and Δy are equal and set to $1/N$, where N is the grid size. The time step Δt is set to $\Delta x/4$ (i.e. it is refined with the spatial step size). Because DASPK uses variable time steps, the output at the time step requested may be interpolated as described in [**6**]. A time series of the truncation error in the infinity norm over time is shown in Fig. 2.8. Table 2.1 lists the truncation error at the simulation time $t = 0.4$. The convergence rate r is calculated as

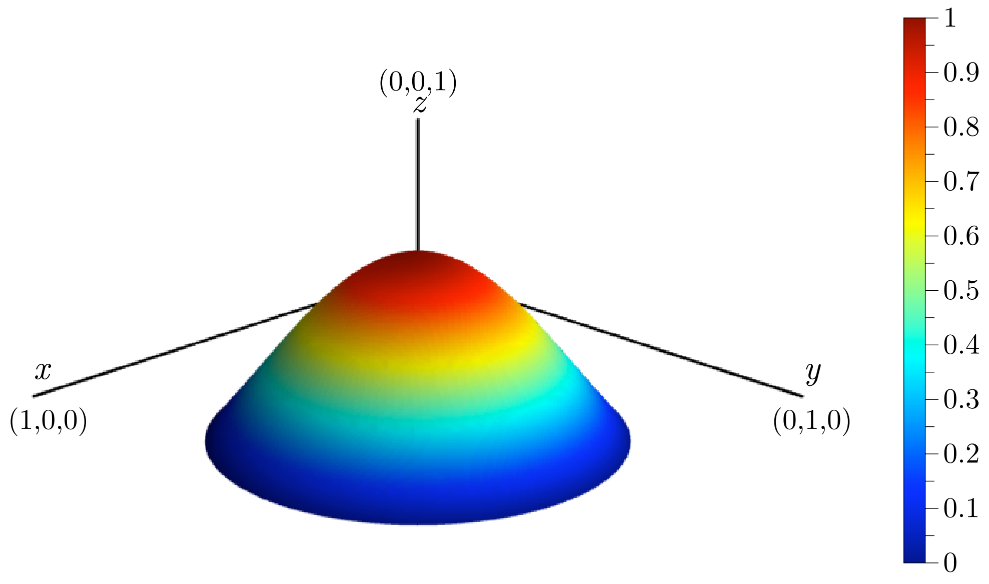
$$r = \log\left(\frac{e_1}{e_2}\right) / \log\left(\frac{\Delta x_1}{\Delta x_2}\right), \tag{2.30}$$

where e_1 and e_2 are errors computed in norms with grid spacing Δx_1 and Δx_2. A log-log plot of truncation error as a function of the spatial step is shown in Figure 2.9. The error was calculated with the computed and exact solutions at the time value of $t = 0.4$. The results of this analysis demonstrate global second order accuracy of the numerical method.

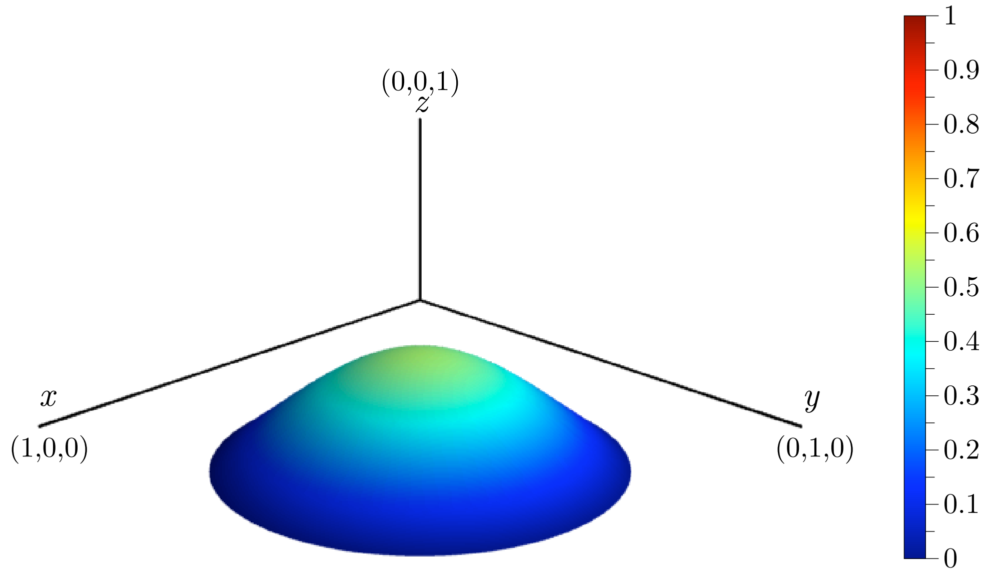
Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.00e-03	2.95e-04	–	2.61e-04	–	5.46e-04	–
100 × 100	2.50e-03	4.94e-05	2.58	4.32e-05	2.59	9.28e-05	2.56
200 × 200	1.25e-03	1.05e-05	2.24	9.20e-06	2.23	2.09e-05	2.15
400 × 400	6.25e-04	2.42e-06	2.11	2.13e-06	2.11	5.42e-06	1.95

TABLE 2.1. The norms and convergence rates for the diffusion equation at the time value of 0.4

Next we tested a nonlinear system with the method. The model system consists of a protein C with two distinct chemical states: active and inactive. The reactions that convert the protein between the two states are assumed to follow Michaelis-Menten kinetics, which describes the kinetics of many enzymatic reactions including phosphorylation and dephosphorylation events [**35**]. The protein C is deactivated in the interior of the



(a) Initial condition for the diffusion equation.



(b) The computed solution to the diffusion equation at $t = 0.2$.

FIGURE 2.7. Computed solution to the diffusion equation at two time values. The x, y, and z axes are bounded by the unit cube.

computational domain according to the following equations:

$$\begin{aligned} \frac{\partial C_i}{\partial t} &= D\Delta C_i + \frac{k_2 C_a}{K_{m2} + C_a} \\ \frac{\partial C_a}{\partial t} &= D\Delta C_a - \frac{k_2 C_a}{k_{m2} + C_a}, \end{aligned} \tag{2.31}$$

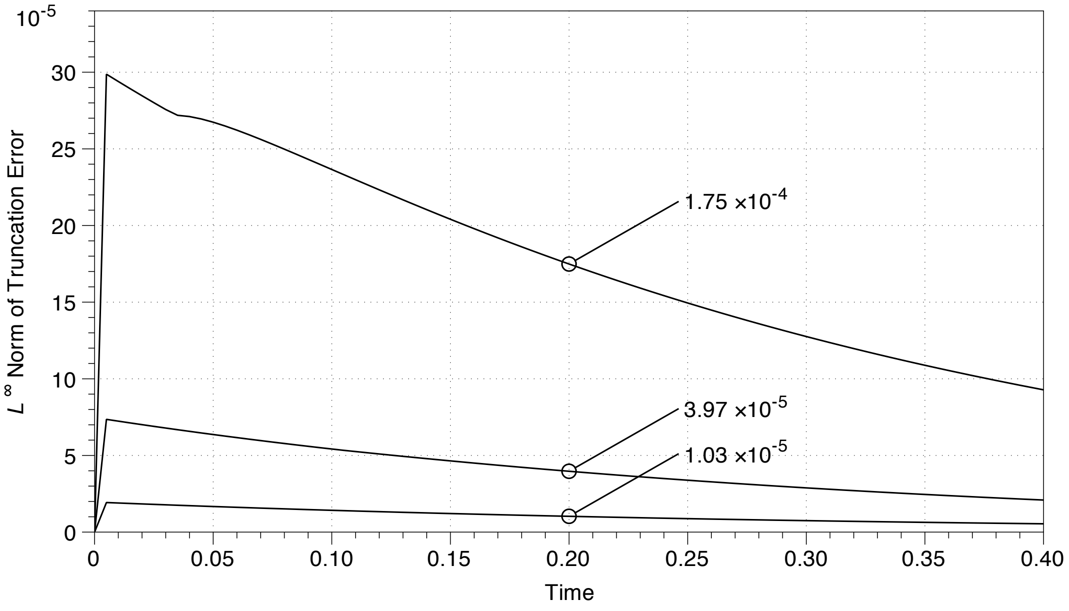


FIGURE 2.8. Truncation error computed in the infinity norm for the diffusion equation. The grid size N is 100 (top), 200 (middle), and 400 (bottom). The time step at each refinement was set to $1/(4N)$, where N is the grid size.

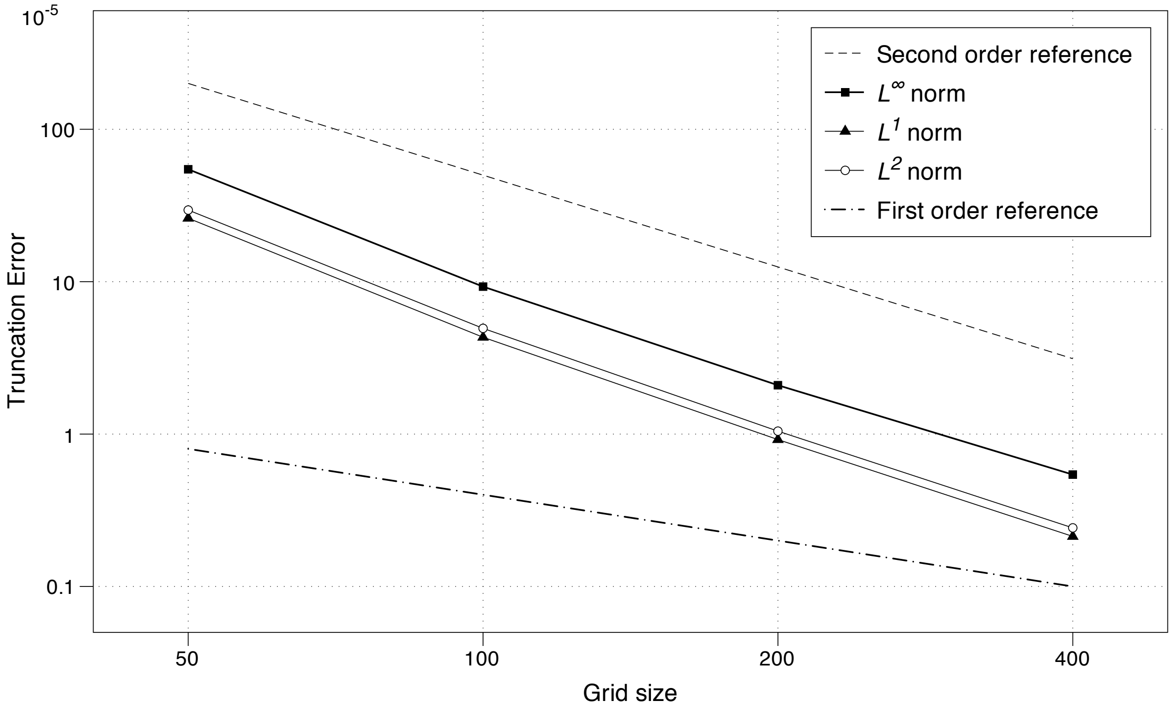


FIGURE 2.9. Truncation error for the diffusion equation at the time value of 0.4. The convergence data are the same as given in Table 2.1.

where C_i and C_a are the concentrations of inactive and active protein, respectively, k_2 is the maximum deactivation rate, and K_{m2} is the Michaelis constant. Activation occurs on the boundary, $\partial\Omega$, according to the following boundary conditions:

$$\begin{aligned} -D\vec{n}\cdot\nabla C_i|_{\partial\Omega} &= \left.\frac{k_1 S C_i}{K_{m1}+C_i}\right|_{\partial\Omega} \\ -D\vec{n}\cdot\nabla C_a|_{\partial\Omega} &= -\left.\frac{k_1 S C_i}{K_{m1}+C_i}\right|_{\partial\Omega}, \end{aligned} \tag{2.32}$$

where k_1 is the maximum activation rate and K_{m1} is the Michaelis constant. The equations are solved in the domain

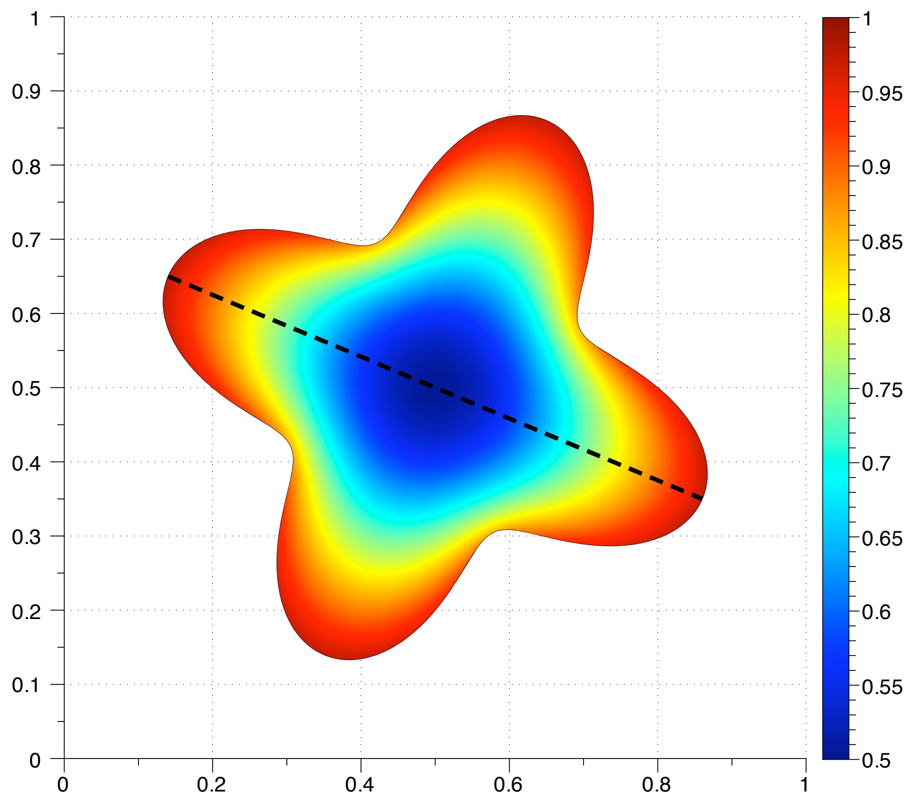
$$\Omega(r,\theta) = r \leq 0.3 - 0.09\sin(4\theta). \tag{2.33}$$

In our simulation, Ω is shifted to the center of the unit box. The initial concentration of inactive protein is assumed to be constant and equal to 1. There is initially no active protein. Fig. 2.10(a) shows a plot of the active concentration at $t = 0.25$. A cross section of the two dimensional geometry at several time values is shown in Fig. 2.10(b). Table 2.2 lists the constants used in the simulation. The constants were arbitrarily chosen to generate a gradient. In [**9**] the reaction rate for a first order phosphatase (dephosphorylation) reaction is between 0.1 and 100 s^{-1}. Michaelis-Menten constants were estimated from 0.1 to 20 μm. Times for execution on a Mac Pro desktop computer with dual-core 2.66 GHz Intel Xeon processors for different grid sized are listed in Table 2.3.

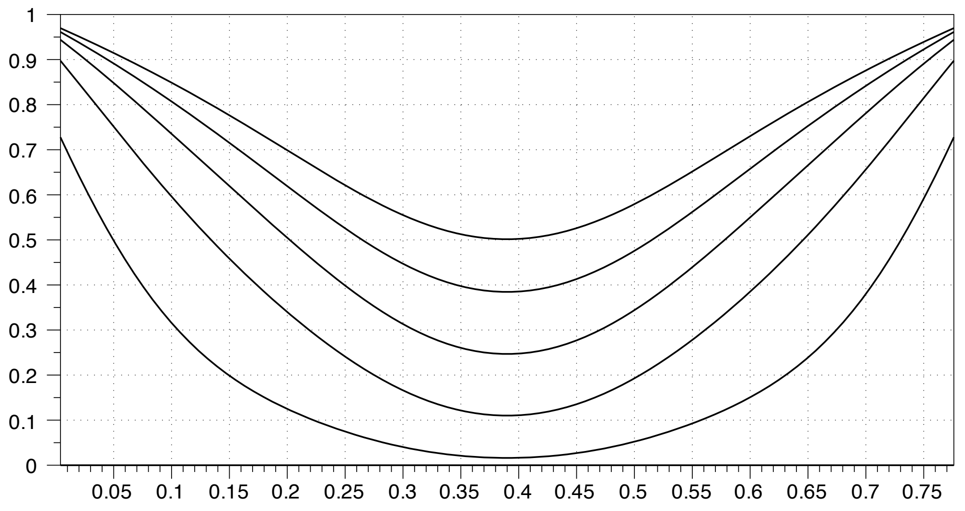
Constant	**Value**	**Constant**	**Value**
D	0.1	k_2	1.0
S	1.0	K_{m1}	0.2
k_1	1.0	k_{m2}	0.2

TABLE 2.2. Constants used in the simulation of the two species model.

We compute the error as the difference between successive grid refinements as follows. The truncation error function $E(x, y, t)$ is defined on interior values of the course grid.



(a) Concentration of the active species at t=0.25.



(b) Active species concentration at evenly spaced time values for $t \in [0, 0.25]$.

FIGURE 2.10. Concentration of the active species C_a. The mesh is sliced along the dashed line in the top figure.

Grid size	**Time step**	**Execution Time (seconds)**
50 × 50	5.000e-03	1.76
100 × 100	2.500e-03	5.81
200 × 200	1.250e-03	32.15
400 × 400	6.250e-04	203.95
800 × 800	3.125e-04	1202.18

TABLE 2.3. Execution times for the two species model. The end time of the simulation was $t = 0.5$.

Computed solution values located in course grid cut cells are excluded from the domain. This includes some values located in interior points for the more refined grid (Fig. 2.11). The truncation error function is defined as

$$E(x, y, t) = c_{\Delta x}(x, y, t) - c_{\Delta x/2}(x, y, t). \tag{2.34}$$

The course grid values are located in the center of a box defined by four refined grid values. Four refined grid values are averaged and subtracted from one course value. Because the time integration is handled implicitly, a different convergence rate of the truncation error in cut cells and boundary values would affect the convergence rate of the truncation error for interior cells. Therefore by computing the error with interior cells, we are still able to draw conclusions about the order of the method.

Table 2.4 lists convergence data for the two species system given by Eqs. (2.31) and (2.32). The data used for calculating the error was taken from computed solutions at the simulation time value of $t = 0.5$. Note that the norms of truncation errors for both C_i and C_a are the same. The system is mass conservative, and the computed solution is also conservative to machine precision. Therefore we only show convergence figures for species C_i. The truncation error for species C_i in the infinity norm as a function of time is listed in Fig. 2.12. A log-log plot of the truncation error as a function of the grid size is listed in Fig. 2.13. From this analysis, we conclude second order accuracy.

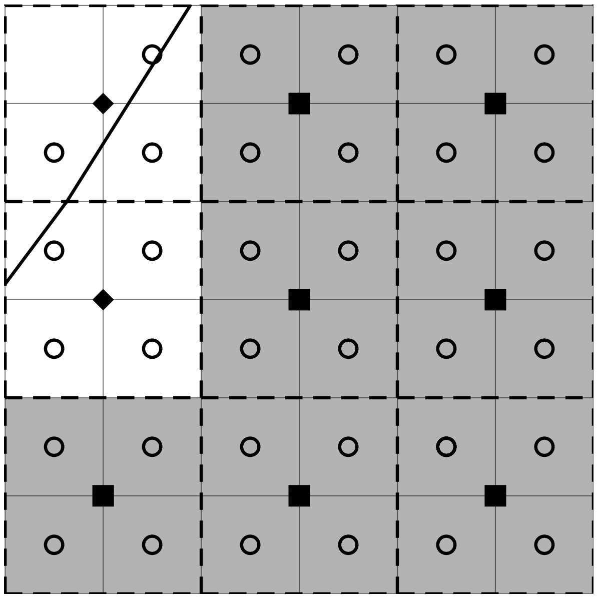


Figure 2.11. Interior grid cells on the course grid (dashed lines) are shaded. Square and diamond filled points indicate locations of cell centered values on the course grid. Values associated with diamond grid points represent cut cells for the courser grid. Course and refined values in these cut cells are not used in the averaging scheme. The refined grid is indicated by solid lines. Circles mark the cell centers of the refined grid cells. Four refined point values are averaged and compared to the the square point on the course grid.

Species C_i							
Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.000e-03	–	–	–	–	–	–
100 × 100	2.500e-03	7.59e-04	–	1.67e-04	–	1.49e-03	–
200 × 200	1.250e-03	1.92e-04	1.98	4.44e-05	1.91	5.01e-04	1.57
400 × 400	6.250e-04	4.57e-05	2.07	1.08e-05	2.04	1.25e-04	2.00
800 × 800	3.125e-04	1.09e-05	2.07	2.61e-06	2.05	3.12e-05	2.00
Species C_a							
Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.00e-03	–	–	–	–	–	–
100 × 100	2.500e-03	7.59e-04	–	1.67e-04	–	1.49e-03	–
200 × 200	1.250e-03	1.92e-04	1.98	4.44e-05	1.91	5.01e-04	1.57
400 × 400	6.250e-04	4.57e-05	2.07	1.08e-05	2.04	12.5e-04	2.00
800 × 800	3.125e-04	1.09e-05	2.07	2.61e-06	2.05	3.12e-05	2.00

Table 2.4. The norms and convergence rates for the two species model at the time value of 0.5.

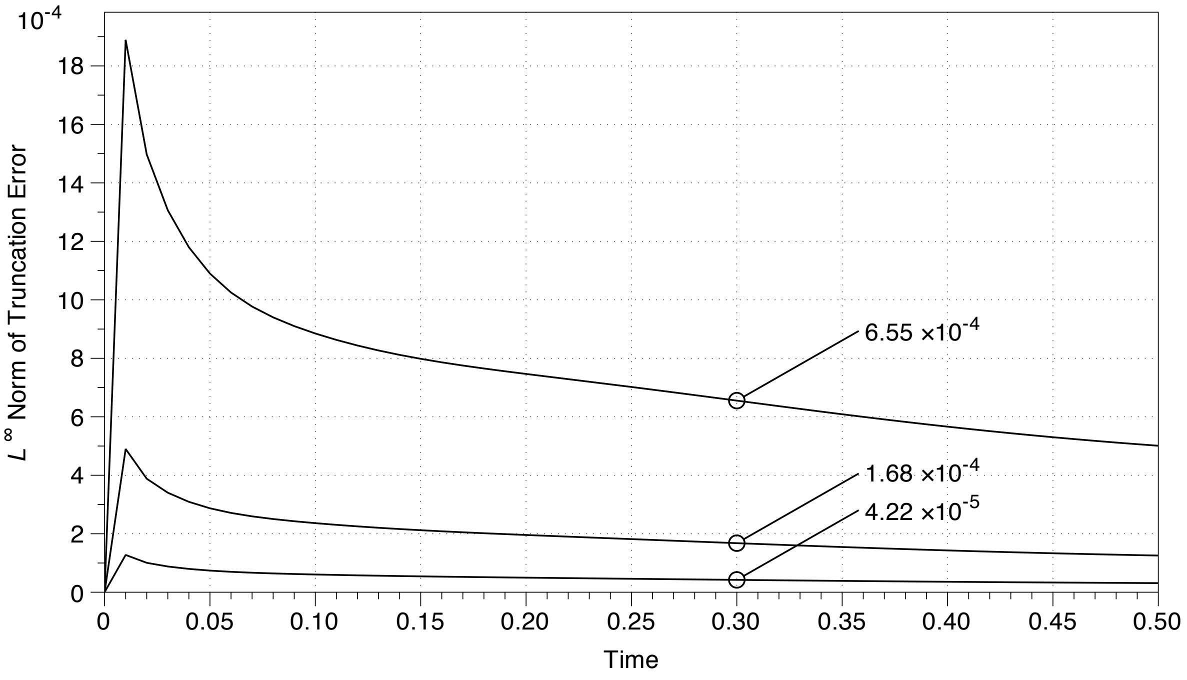


FIGURE 2.12. Truncation error for species C_i as computed in the infinity norm for the reaction-diffusion equation. The error for the top plot was computed by subtracting the solution at $N = 200$ from $N = 100$ as described in the text. The middle plot was calculated with grid sizes $N = 200$ and $N = 400$. The bottom was calculated with grid sizes $N = 400$ and $N = 800$.

2.5. A two compartment model

In this model, we have two compartments: the cytoplasm and nucleus. The cellular geometry was taken from a yeast cell undergoing chemotrophic growth in the direction of a pheromone gradient [**56**]. Proteins involved in the pheromone response pathway are known to localize on the plasma membrane, the nucleus, and in the cytosol [**20**]. The nucleus is modeled as a circle located toward the front of the cell. Because yeast cells are three dimensional, we model the top view of the cell as in [**18**], where membrane-bound species are located in the interior of the computational domain but are assumed to diffuse slower than cytosolic forms.

The model consists of two species, A and C, with inactive and active forms. Protein C is allowed to enter and exit the nucleus, whereas protein A is restricted to the cytoplasm (Fig. 2.14(a)). Initially both A and C are in their inactive forms. At the beginning of

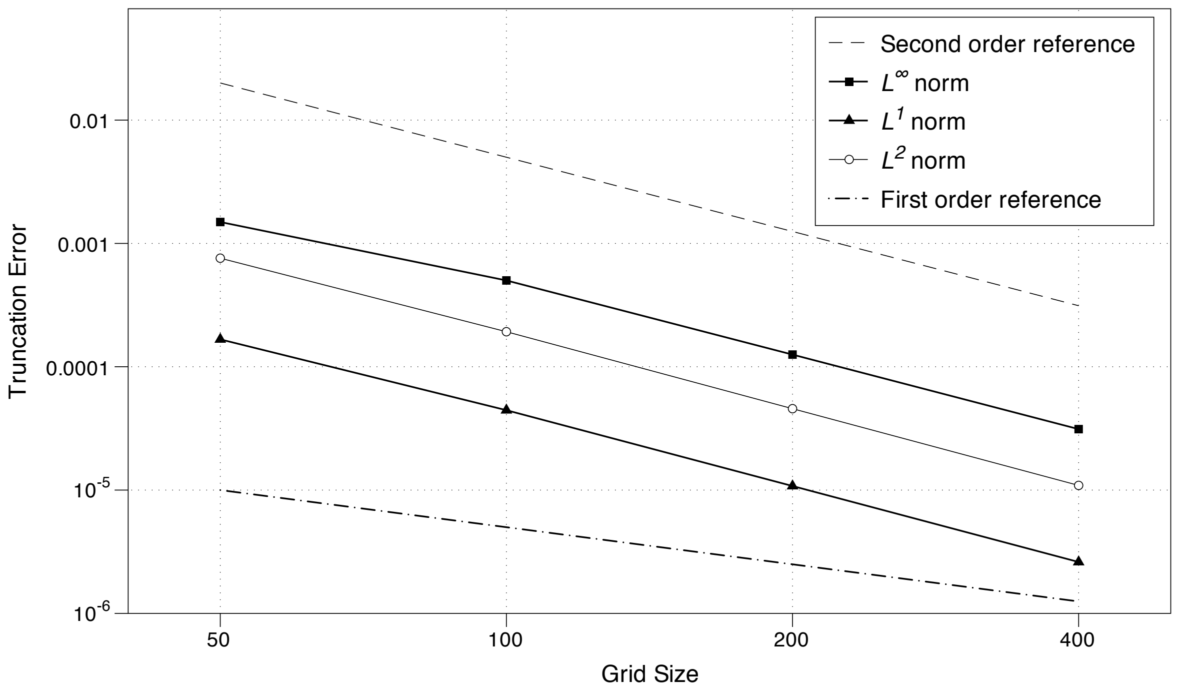


FIGURE 2.13. Truncation error for C_i at the time value of 0.5. The convergence data are the same as given in Table 2.4.

the simulation, the reaction rate for the activation of A, k_0, is instantaneously increased from 0 to 1. This is meant to model the cell receiving an external signal. Once A is activated it is assumed to interact with the cell membrane, causing a reduction in the protein's diffusion coefficient [**82**]. The active form of A can then activate protein C. The active form of C is only deactivated within the nucleus. This simple model captures some of the signaling events that occur during the pheromone response of yeast [**75**]. If we denote the concentration of a chemical species with brackets, the equations for the cytoplasmic species are:

$$\begin{aligned}
\frac{\partial[A_c]}{\partial t} &= D_1\Delta[A_c] - k_0[A_c] \\
\frac{\partial[A_c^*]}{\partial t} &= D_2\Delta[A_c^*] + k_0[A_c] \\
\frac{\partial[C_c]}{\partial t} &= D_1\Delta[C_c] - k_1[A_c^*][C_c] \\
\frac{\partial[C_c^*]}{\partial t} &= D_1\Delta[C_c^*] + k_1[A_c^*][C_c],
\end{aligned} \tag{2.35}$$

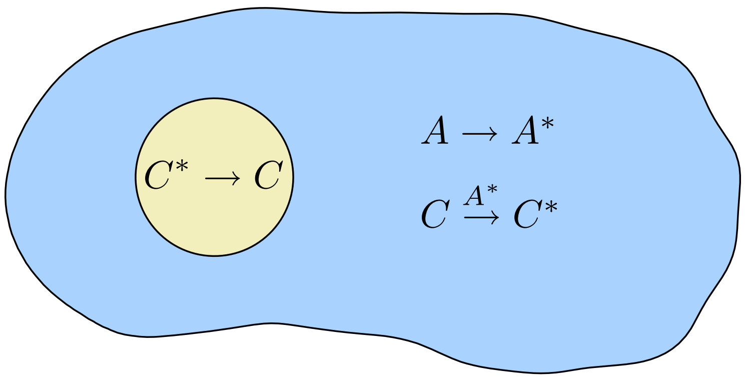
where the asterisks denote the active form of the protein, D_1 is the diffusion coefficient in the cytoplasm, D_2 is diffusion coefficient in the membrane, and the k's represent the reaction rates. Subscripts indicate cytosolic and nuclear species. The boundary conditions at the cell membrane $\partial\Omega_1$ are no flux for all chemical species. The nuclear boundary conditions for species A are also no flux, whereas species C are allowed to move through the nuclear membrane $\partial\Omega_2$ and satisfy the conditions

$$\begin{aligned}
-D_1(\vec{n}\cdot\nabla[C_c])|_{\partial\Omega_2} &= -k_2([C_n]-[C_c])|_{\partial\Omega_2}\\
-D_1(\vec{n}\cdot\nabla[C_c^*])|_{\partial\Omega_2} &= -k_2([C_n^*]-[C_c^*])|_{\partial\Omega_2}\\
-D_1(\vec{n}\cdot\nabla[C_n])|_{\partial\Omega_2} &= k_2([C_n]-[C_c])|_{\partial\Omega_2}\\
-D_1(\vec{n}\cdot\nabla[C_n^*]|_{\partial\Omega_2} &= k_2([C_n^*]-[C_c^*])|_{\partial\Omega_2}.
\end{aligned}$$

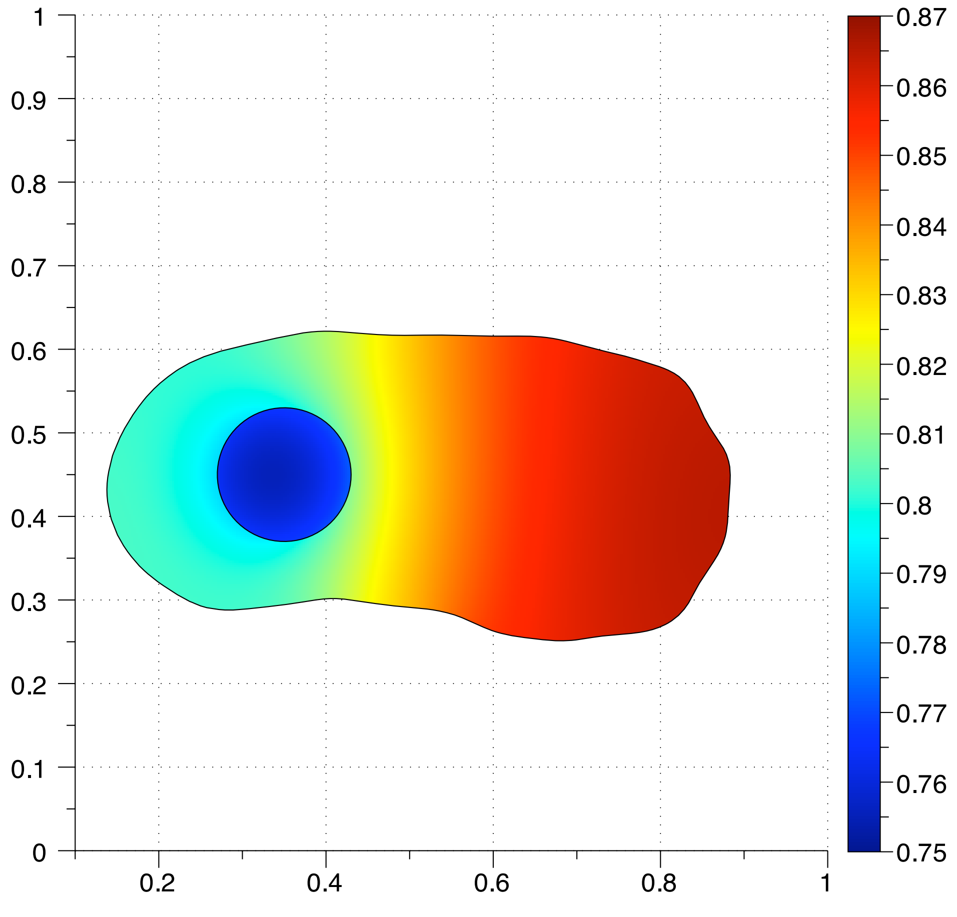
Nuclear C^* is deactivated according to the equations

$$\begin{aligned}
\frac{\partial[C_n]}{\partial t} &= D_1\Delta[C_n] + k_3[C_n^*]\\
\frac{\partial[C_n^*]}{\partial t} &= D_1\Delta[C_n^*] - k_3[C_n^*].
\end{aligned} \tag{2.36}$$

The steady state spatial distribution of active C is illustrated in Fig. 2.14(b). All reaction constants were arbitrarily chosen to be one, $D_1 = 0.1$, $D_2 = 0.01$, and $\Delta x = 1/200$. In our simulation, the cell is bounded by the unit box. However, budding yeast have a diameter of about 10 μm [**4**]. Diffusion coefficients range from 0.0025 to 0.1 $\mu\text{m}^2/s$ [**82**]. The initial values were zero except for $[A_c](x, y, 0) = [C_c](x, y, 0) = 1$. The execution time of the simulation to run from $t = 0$ until $t = 20$ was 150 seconds on a Mac Pro desktop computer with dual-core 2.66 GHz Intel Xeon processors. To verify that the system is close a steady state solution at $t = 20$, we subtracted the solution of active C in the cytoplasm $[C_c^*]$ for all times from the assumed steady state solution at the time value of $t = 20$. If the system exponentially converges to the computed solution at $t = 20$, we assume this time value is close to steady state. Fig. 2.15 shows the infinity norm of the difference between the computed solution and the assumed steady-state solution sampled over time. Based on this data, the system is close to steady state.



(a) Reactions and species in the two compartment model.



(b) Steady state concentration values for active C species in the cytoplasm and nucleus.

FIGURE 2.14. Two compartment model

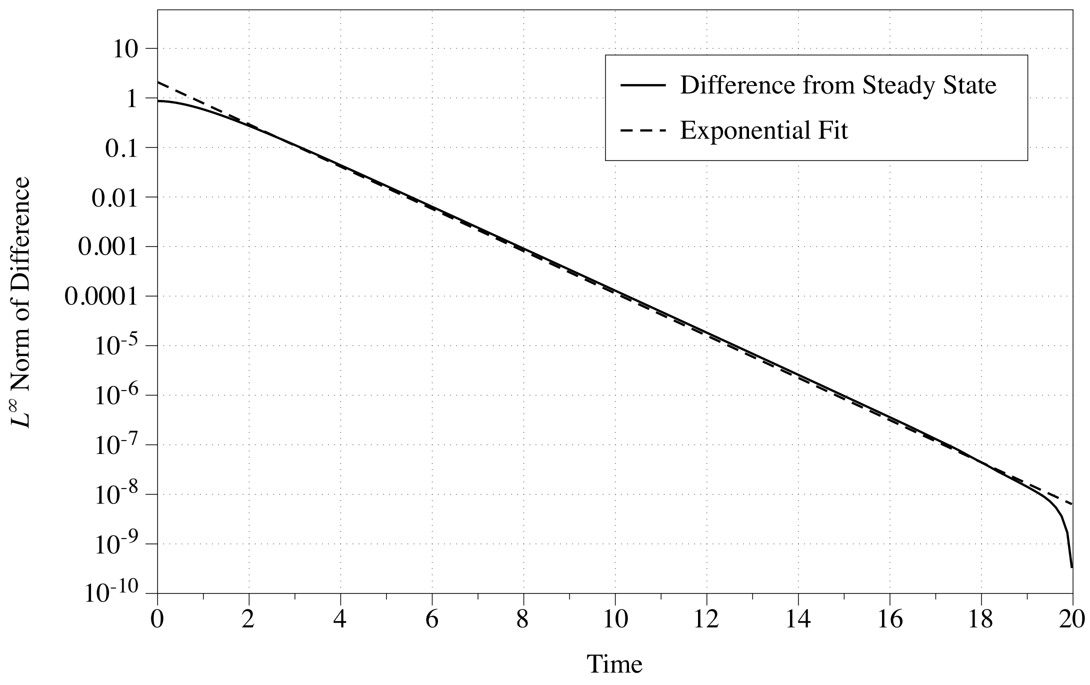


FIGURE 2.15. The solid line indicated the norm of difference of the computed solution at the assumed steady state value at $t = 20$ from computed solution over time. The dashed line is the exponential fit. The scale for the y axis is logarithmic.

The model simulation suggests a spatial activation gradient can be generated by the position of the nucleus. The inactivation of C in the nucleus leads to a higher concentration of active protein in the rear of the cell in spite of a uniform spatial signal from active A.

2.6. Rho family GTPase model

The Rho family of GTPases regulates many cellular functions, including polarization and motility. We created a model with three key members of this family, $Cdc42$, *Rac*, and *Rho* (Fig. 2.6). The interactions are based on [**12**]. A more complicated model involving these proteins in one dimension can be found in [**18**]. As in the previous example, we assume a top view of a three dimensional cell with membrane bound active forms and cytosolic inactive forms of the three proteins. The model has a total of six species. The mouse embryonic fibroblast (MEF) cell boundary $\partial\Omega$ is taken from supplemental material from [**62**].

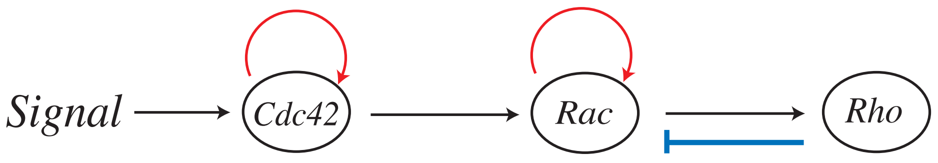


FIGURE 2.16. Schematic of interactions for the Rho GTPase model.

In our model, a uniform extracellular signal triggers the activation of $Cdc42$ protein on the cell edge,

$$\begin{aligned} -D\vec{n}\cdot\nabla\left[Cdc42_i\right]\big|_{\partial\Omega} &= \frac{k_1 S\left[Cdc42_i\right]}{K_{m1}+\left[Cdc42_i\right]}\bigg|_{\partial\Omega} \\ -D\vec{n}\cdot\nabla\left[Cdc42_a\right]\big|_{\partial\Omega} &= -\frac{k_1 S\left[Cdc42_i\right]}{k_{m2}+\left[Cdc42_i\right]}\bigg|_{\partial\Omega}. \end{aligned} \tag{2.37}$$

In the cell interior, active $Cdc42$ is inactivated. A positive feedback loop increases the activation of $Cdc42$,

$$\begin{aligned} \frac{\partial\left[Cdc42_i\right]}{\partial t} &= D\Delta\left[Cdc42_i\right] + \frac{k_2\left[Cdc42_a\right]}{K_{m3}+\left[Cdc42_a\right]} - \frac{k_3\left[Cdc42_a\right]\left[Cdc42_i\right]}{K_{m4}+\left[Cdc42_i\right]} \\ \frac{\partial\left[Cdc42_a\right]}{\partial t} &= D\Delta\left[Cdc42_a\right] - \frac{k_2\left[Cdc42_a\right]}{k_{m5}+\left[Cdc42_a\right]} + \frac{k_3\left[Cdc42_a\right]\left[Cdc42_i\right]}{k_{m6}+\left[Cdc42_i\right]}. \end{aligned} \tag{2.38}$$

Rac is activated by $Cdc42$, and a positive feedback loop increases the concentration of active Rac. Active Rho increases the deactivation of Rac in the cytosol,

$$\begin{aligned} \frac{\partial\left[Rac_i\right]}{\partial t} &= D\Delta\left[Rac_i\right] + \frac{\left(k_4\left[Rho_a\right]+k_5\right)\left[Rac_a\right]}{K_{m7}+\left[Rac_a\right]} \\ &\quad - \frac{\left(k_6\left[Cdc42_a\right]+k_7\left[Rac_a\right]\right)\left[Rac_i\right]}{K_{m8}+\left[Rac_i\right]} \\ \frac{\partial\left[Rac_a\right]}{\partial t} &= D\Delta\left[Rac_a\right] - \frac{\left(k_4\left[Rho_a\right]+k_5\right)\left[Rac_a\right]}{k_{m9}+\left[Rac_a\right]} \\ &\quad + \frac{\left(k_6\left[Cdc42_a\right]+k_7\left[Rac_a\right]\right)\left[Rac_i\right]}{k_{m10}+\left[Rac_i\right]}. \end{aligned} \tag{2.39}$$

Rho is activated by the active form of *Rac* and deactivated in the interior,

$$\begin{aligned}\frac{\partial\,[Rho_i]}{\partial t} &= D\Delta[Rho_i] + \frac{k_8\,[Rho_a]}{K_{m11}+[Rho_a]} - \frac{k_9\,[Rac_a]\,[Rho_i]}{K_{m12}+[Rho_i]}\\ \frac{\partial\,[Rho_a]}{\partial t} &= D\Delta\,[Rho_a] - \frac{k_8\,[Rho_a]}{k_{m13}+[Rho_a]} + \frac{k_9\,[Rac_a]\,[Rho_i]}{k_{m14}+[Rho_i]}.\end{aligned} \tag{2.40}$$

The boundary conditions for *Rac* and *Rho* species are no flux. The steady state distribution is displayed in Fig. 2.6. To achieve these results, a step size $\Delta x = 1/200$ and a diffusion coefficient $D = 0.1$ were used. Again, our simulation is on a unit box. An estimate of the MEF cell length is 80 μm [**62**]. Estimates for diffusion coefficients in [**9**] are 1 to 10 $\mu m^2/s$ for general eukaryotic circular cells with radius 10 μm. Our diffusion coefficient was chosen to highlight the gradient. The reaction constants from the simulation were arbitrarily chosen and are listed in Table 2.5. The initial concentration of inactive chemical species was set to one and zero for active species. The execution time was 217 seconds for 1600 time steps on a Mac Pro desktop computer with dual-core 2.66 GHz Intel Xeon processors.

In this model, a gradient is formed by protein activation on the cell edge. The gradient is propagated to the downstream signaling components *Rac* and *Rho*. Fig. 2.6 shows that filopodia and thin protrusions have higher activation levels due the increased ratio of cell membrane to cell volume in these regions [**50**].

Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
S	1.0	k_8	3.0	K_{m3}	0.2
k_1	5.0	k_9	5.0	K_{m4}	0.2
k_2	3.0	K_{m1}	0.2	k_{m5}	0.2
k_3	1.0	k_{m2}	0.2	k_{m6}	0.2
k_4	3.0	K_{m7}	0.2	K_{m11}	0.2
k_5	1.0	K_{m8}	0.2	K_{m12}	0.2
k_6	5.0	k_{m9}	0.2	k_{m13}	0.2
k_7	1.0	k_{m10}	0.2	k_{14}	0.2

Table 2.5. Reaction constants used in the simulation of the Rho GTPase model.

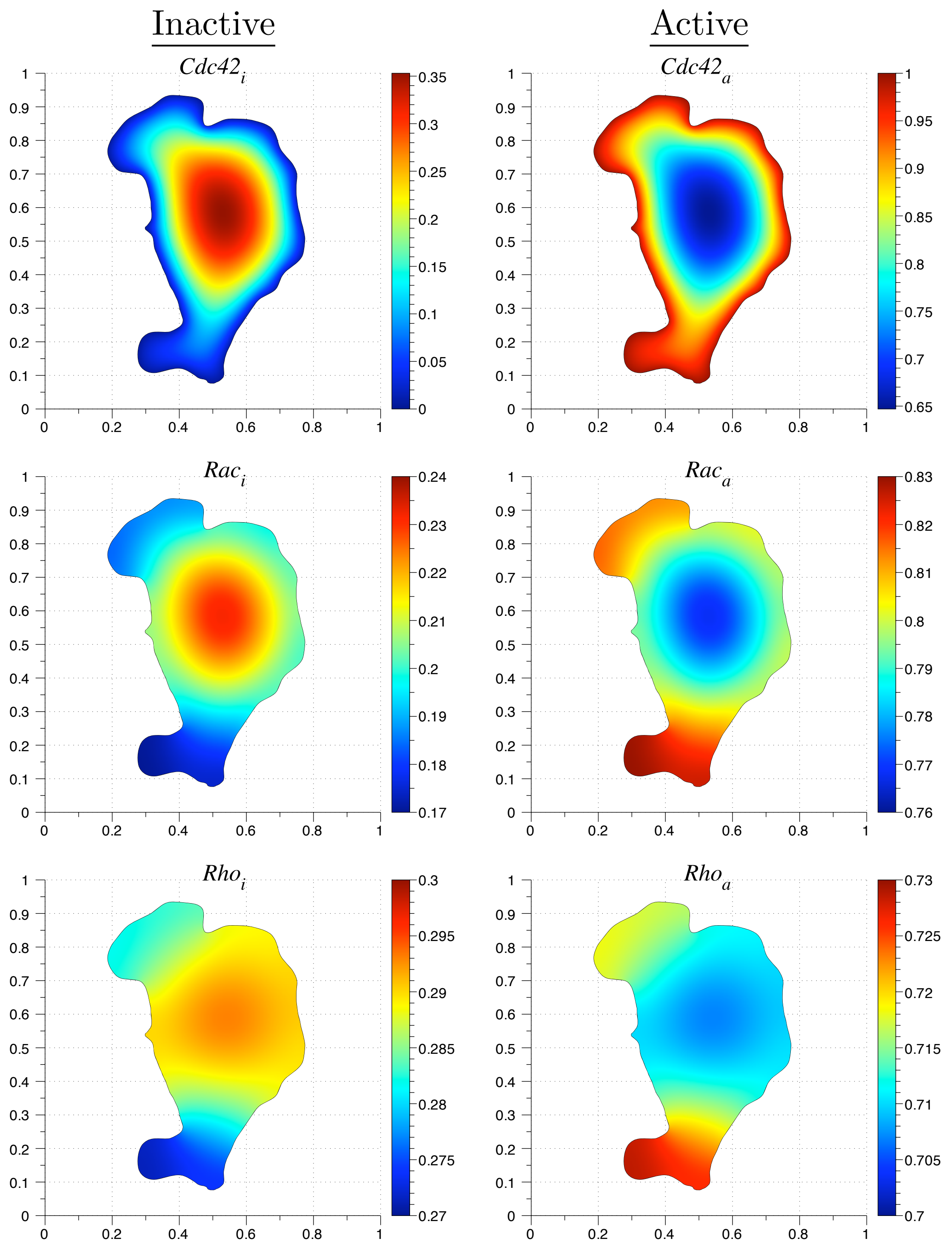


FIGURE 2.17. Steady state distribution of protein concentration amounts in a fibroblast. The boundary was taken from a live cell image [**62**].

2.7. Conclusions

We have developed an accurate and efficient cut cell method for simulating spatial models of signaling pathways in realistic cellular geometries. We demonstrated our method using models that consist of multiple species interacting in multiple compartments. The examples were chosen to illustrate the numerical methods and therefore lack many details found in real biological signaling systems. In particular, feedback and feed forward control mechanisms that regulate pathway activity were not considered in detail. Our numerical methods provide important tools for investigating such regulatory mechanisms in realistic cell geometries and, therefore, should provide important insights into the ways signaling networks process and transmit information.

Our algorithm extends previous work on embedded boundary methods [**16, 32, 48, 71**]. These methods have been implemented in two and three dimension for Poisson's equation, the heat equation, and hyperbolic conservation laws. Our formulation extends these methods to systems of reaction-diffusion equations with nonlinear reactions in the interior as well as nonlinear reactions affecting boundary values. The boundary conditions treated in previous work [**32, 48, 71**] have been homogenous Dirichlet and Neumann, which is not sufficient for many models of signaling pathways [**50**]. In [**48**] a second order implicit method was used to update the heat equation in time [**80**]. In our method, we use an implicit nonlinear solver to handle nonlinear reactions occurring in the interior. An advantage of the differential-algebraic formulation is the ability to treat the boundary conditions as algebraic constraints. This allows us to handle reactions that take place on the physical boundary of the reaction-diffusion equation.

One limitation of the finite volume discretization arises from the interpolation method to obtain the normal derivative to the surface as shown in Fig. 2.5. Cut cells must not have a zero volume cell within two rows or columns. For biological cells with long, thin, or irregularly-shaped components such as neurons, mesh adaptive refinement may be needed to resolve the cellular geometry.

The underlying Cartesian-grid based finite volume discretization allows us to use advection schemes originally developed for hyperbolic conservation laws to simulate active transport or motility. In the next chapter, we show how level set methods [**59, 72**] can be combined with biochemical reaction networks to investigate the effect of moving boundaries on cell signaling.

CHAPTER 3

Simulating Models of Biochemical Signaling Networks in Complex Moving Geometries

The algorithm presented in the previous chapter is based on an underlying Cartesian grid. Hence, modifications can be made to extend the fixed boundary algorithm to a moving boundary framework. Applications of this algorithm include simulating biochemical reaction networks in motile and growing cells. Chemotrophic growth of yeast is associated with spatial gradients of proteins [**56**]. A model of chemotrophic growth of yeast has been proposed in [**31**]. The authors solve reaction-diffusion equations to steady state every time the boundary changes. This assumption would not hold in fast migrating cells, such as keratocytes [**37**]. Spatiotemporal dynamics in moving and deforming cells have been shown in [**36, 55, 62**]. Simulations of these systems can reveal the role of cell morphology on spatial distribution of signaling proteins as cells move and deform.

3.1. Mathematical formulation

We follow the same formulation as Chapter 2, where spatial models of biochemical reaction networks are represented using partial differential equations. In this chapter, we consider an advection term in addition to reaction and diffusion terms. For simplicity we restrict ourselves to two spatial dimensions x and y. For a given chemical species, the advection term represents active transport. In a system consisting of n chemical species, the concentration of the i^{th} species c_i evolves in space and time according to the following equation:

$$\frac{\partial c_i}{\partial t} + \vec{U} \cdot \nabla c_i = -\nabla \cdot \mathbf{J} + f_i(\boldsymbol{c}). \tag{3.1}$$

The flux density is given by $\mathbf{J} = -D_i \nabla c_i$, where D_i is the diffusion coefficient. The function $f_i(\boldsymbol{c})$ models reactions within the cell that affect c_i. The cell and biochemical species inside it propagate according to the velocity field $\vec{U}$. The elements of the vector $\boldsymbol{c}$ are the concentrations of the n chemical species. Reactions also may occur on the cell membrane yielding nonlinear conditions on the time-dependent boundary $\partial\Omega(t)$,

$$-D\vec{n} \cdot \nabla c_i \,|_{\partial\Omega(t)} + g(\boldsymbol{c})|_{\partial\Omega(t)} = 0. \tag{3.2}$$

Eqs. (2.1) and (2.2) are solved subject to appropriate initial conditions $c_i(x, y, 0)$ for each species in the system.

3.2. Numerical methods

In this section, we detail the methods used to simulate models of signaling and regulatory pathways in realistic cellular geometries that are moving. This involves obtaining the domain from live-cell images and embedding the boundary in a signed distance function. We describe level set methods and the finite volume method used in the complete time update, which involves operator splitting.

3.2.1. Computational domain and level set methods.

The computational domain can be specified explicitly or obtained from live cell images. In Chapter 2 cell images are smoothed with a Gaussian filter, and the boundary is obtained by thresholding the smoothened image. In either case, a signed distance function is constructed such that the zero level set is the boundary of the cell. The signed distance function denoted by $\phi(x, y, 0)$ is the initial value for equation of motion in level set methods. Note that $\phi(x, y, t)$ is defined on the Cartesian grid $(x_0 + i\Delta x, y_0 + j\Delta y)$ for $0 \leq i \leq N$ and $0 \leq j \leq M$. The stepsizes are defined as $\Delta x = 1/N$ and $\Delta y = 1/M$.

Level set methods are algorithms for tracking the time evolution of boundaries and interfaces. The evolution of the signed distance ϕ is given by the level set equation originally presented in [**59**],

$$\phi_t + F|\nabla\phi| + \vec{U}\cdot\nabla\phi = 0, \tag{3.3}$$

where F is the speed in the normal direction to the boundary and $\vec{U}$ is the advection velocity. In order to numerically solve the partial differential equation in Eq. (3.3), spatial and temporal operators must be discretized. The boundary that we wish to capture may not be differentiable (e.g. sharp corners). Schemes developed to numerically solve hyperbolic conservation laws can be applied to Eq. (3.3). We implemented a second order upwinding scheme to discretize the advection operators in (3.3). Specifically, we used an essentially non-oscillatory (ENO) scheme to approximate spatial derivatives [**26**]. Finite difference schemes can also be found in [**72**].

A second order Runge Kutta method is used to update Eq. (3.3) in time. Following [**3**] Eq. (3.3) is written as

$$\phi_t = L(\vec{U}, F, \phi).$$

Then the time update is

$$\begin{aligned} T_1 &= L(\vec{U}(t^n), F(t^n), \phi)\Delta t \\ \phi &= \phi + T_1 \\ T_2 &= L(\vec{U}(t^{n+1}), F(t^{n+1}), \phi)\Delta t \\ \phi &= \phi + (T_2 - T_1)/2. \end{aligned} \tag{3.4}$$

The time step t^n is defined as $t_0 + n\Delta t$. In section 3.2.3, we describe the role of the level set update in the complete update for state of the advection-reaction-diffusion system. A more detailed discussion of level set methods is provided in Appendix A.

3.2.2. Finite volume method.

The finite volume method in Chapter 2 is used to discretize reaction and diffusion terms in Eq. (3.1) with boundary equations Eq. (3.2).

Recall the concentration over a control volume V_{ij} is stored at c_{ij},

$$\bar{c}_{i,j} = \frac{1}{V_{i,j}} \iint_{V_{i,j}} c(x,y)dV. \tag{3.5}$$

The grid where concentration values are located is different from the grid where signed distance values are stored. In the finite volume method, the grid points are the cell centered values $(x_0 + (i+1/2)\Delta x, y_0 + (j+1/2)\Delta y)$ for $0 \le i \le N$ and $0 \le j \le M$. In the description of the method in Chapter 2, we referred to (i,j) grid points as cell-centered. The signed distance function is defined on edge-centered grid points because the zero level set defines control volumes. Fig. 3.1 illustrates the computational points in the finite volume method. Boundary values are located at the midpoint of the piecewise linear boundary approximation.

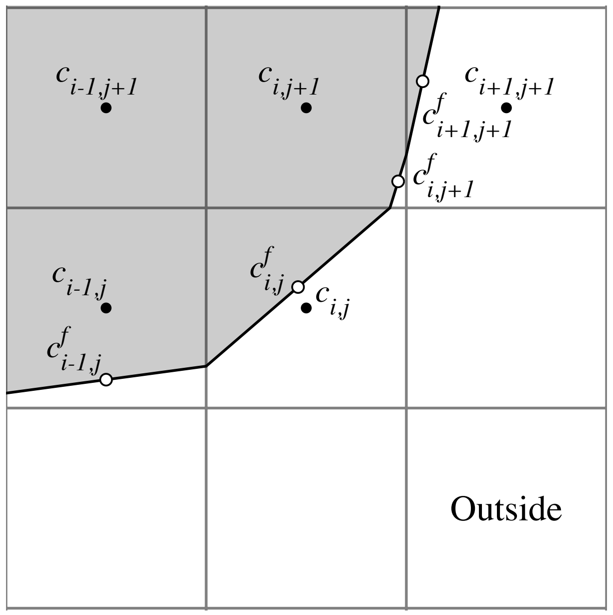


FIGURE 3.1. Stencil points for the finite volume method. The solid line indicates a piecewise linear approximation to the zero level set of the signed distance function. The black points labeled $c_{i,j}$ indicate values that lie on a Cartesian grid. White points labeled $c^f_{i,j}$ indicate values associated with the boundary.

Inserting the Eq. (3.5) into reaction and diffusion terms in Eq. (3.1) produces

$$\frac{\partial \bar{c}_{i,j}}{\partial t} - \overline{f(c)}_{i,j} = -\frac{1}{V_{i,j}} \iint_{V_{i,j}} \nabla \cdot \mathbf{J} dV = -\frac{1}{V_{i,j}} \int_{\partial V_{i,j}} (\mathbf{J} \cdot \vec{n})\, dS, \tag{3.6}$$

by the divergence theorem. Approximations of the surface integrals give a discretization for the reaction-diffusion equation. An approximation to the normal component of the diffusive flux is used to discretize the boundary conditions in Eq. (3.2), forming algebraic constraints. The discretizations of Eqs. (3.1) and (3.2) are treated as a differential-algebraic system of nonlinear equations of the form (2.21). DASPK [**10**] is used to solve the Newton iteration (2.27).

3.2.3. Overview of complete temporal update. The complete numerical scheme involves splitting the advection operator from combined reaction-diffusion operator. We implemented the second order Strang operator splitting method. Fig. 3.2 illustrates the separation of the operators as the system is updated in one time step from t^n to t^{n+1}. The reaction-diffusion operator with the appropriate boundary conditions is solved using the cut cell method (Chapter 2) on the boundary $\Omega(t^n)$ for a half time step t^n to $t^{n+1/2}$. Chemical species and the boundary are advected for a full time step t^n to t^{n+1}. The reaction-diffusion solver is reinitialized. The reaction-diffusion equation is then solved on the domain $\Omega(t^{n+1})$ from $t^{n+1/2}$ until t^n. This concludes the update. In the cut cell method, there are three types of points: interior, cut cells and boundary points (Fig. 3.1). In the remainder of this section, we proceed to discuss the details involved in the process of updating and reinitializing these values during a time step.

3.2.4. Interior advection scheme. To approximate the gradient operator with second order upwinding, ghost cells are needed. Minimally two cells in the north, south, east, and west direction are required to create a valid advection update for interior cells. The values in cut cells are recomputed because they do not correspond to values at grid points. The extension values are calculated with bicubic interpolation that is exact for cubic polynomials in $p(x)$, $q(y)$ and $p(x) \cdot q(y)$. The values for the interpolation are located along grid lines. If more than one choice is available, interpolation is performed along each line and the resulting value is the average of these interpolations (Fig. 3.3.) The scheme shown in Fig. 3.3 requires the grid cell containing the interpolation value to lie horizontally or vertically (not diagonally) of an interior grid cell. The cell directly

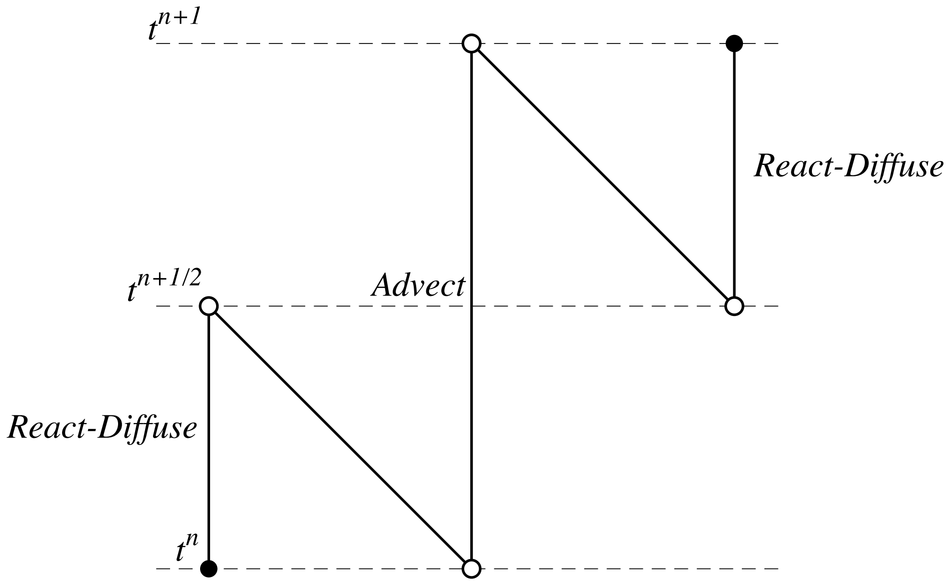


FIGURE 3.2. Strang splitting scheme.

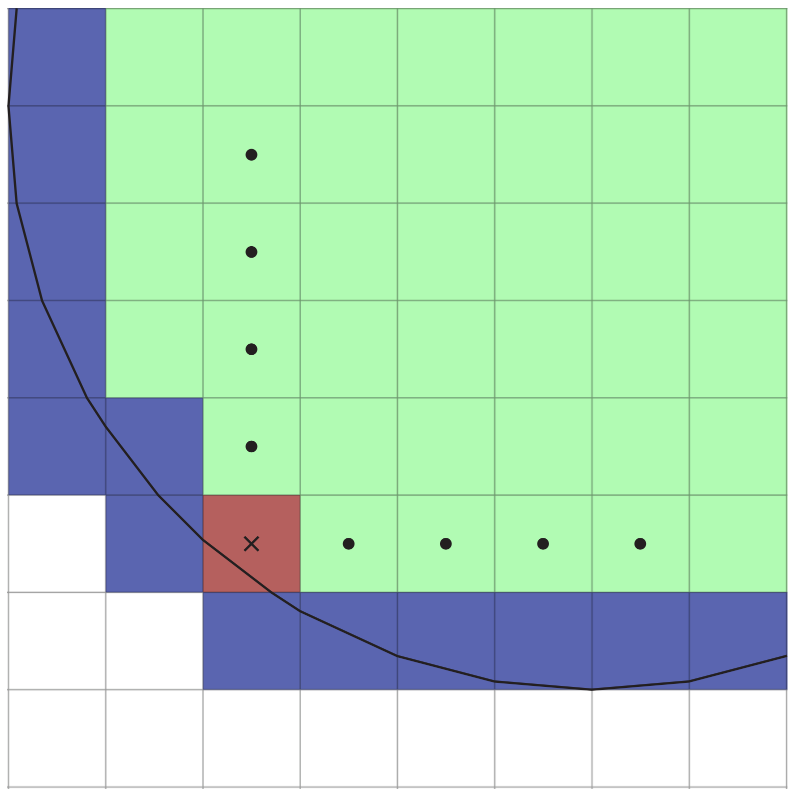


FIGURE 3.3. Interpolation diagram. Interior values are colored green. Cut cell values are colored blue. Circles mark the interior points used to interpolate at the point marked with a cross.

left of the red grid cell does not satisfy this requirement. This type of cell is marked for the second round of interpolation. Later rounds of interpolation use values from earlier rounds. The interpolation continues in bands until sufficient coverage for a second order ENO update. Fig. 3.4 shows an extension computed by interpolation the function,

$$f(x,y) = \cos\left((2\pi(x-0.5)\right)\cos\left(2\pi(y-0.5)\right)$$

from interior points with a grid size of $\Delta x = 1/30$. The extension is larger than two bands in certain places to ensure a valid extension when the front moves. Table 3.1 lists

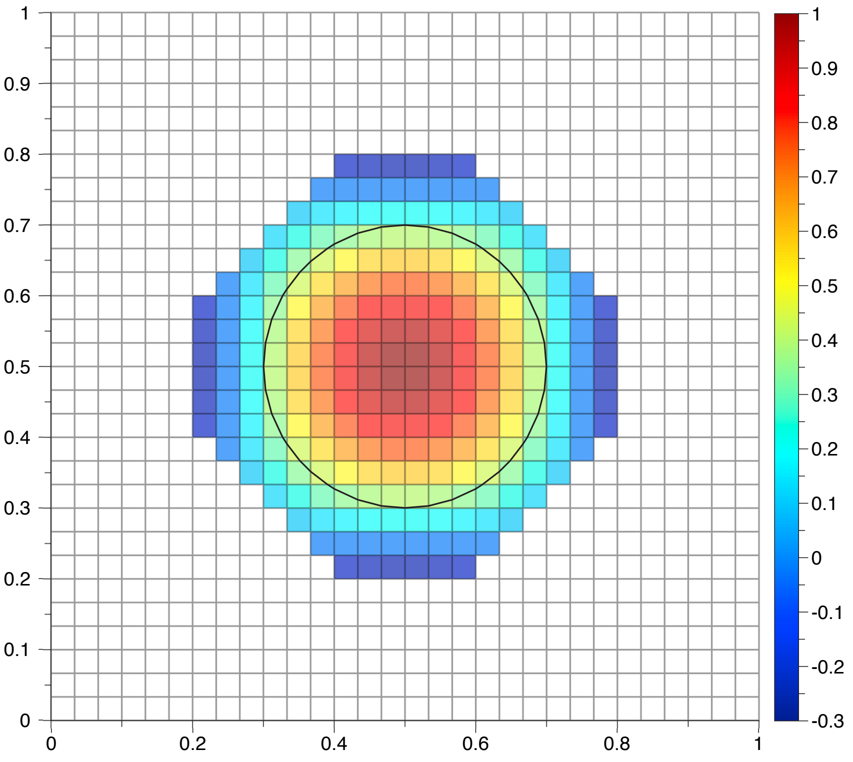


FIGURE 3.4. Example of an interior extension.

the error at different grid sizes for the same function. The extrapolation scheme requires four interior points in the $\vec{n} = (\pm 1, 0)$ or $(0, \pm 1)$ directions. Fig. 3.5 shows a case where the extension fails.

After a time step, the cut cells must be reinitialized for the reaction-diffusion solver. In the stationary boundary solver, we initialized the cut cells with the value of the chemical

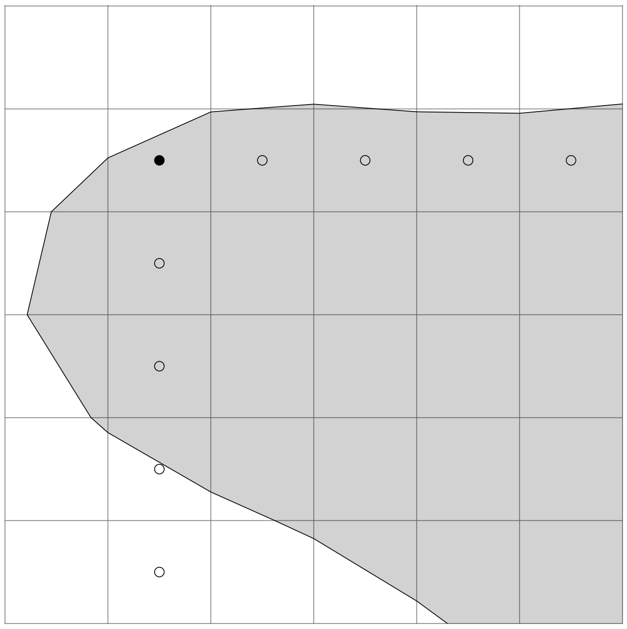


FIGURE 3.5. Example of an under-resolved geometry at the black grid value.

concentration function at the centroid. In the moving boundary case, we use the same bicubic interpolation scheme to interpolate the value at the centroid point.

3.2.5. Convergence tests for the advection operator. We test the discretized advection scheme at values associated with grid based points. A scalar function is evaluated at cell center values on the interior, and centroid values at cut cells. The signed distance is located at cell edge values. Therefore, if quantities such as normals and curvature are needed at cell-centered values, the value at cell centers is interpolated from edge values. The exact solution is a function $f(x, y, t)$ evaluated at cell center points $((x_0 + (i + 1/2)\Delta x, y_0 + (j + 1/2)\Delta y)$. Our first test is to advect a quadratic polynomial in x and y with constant velocity $\vec{U} = (u, v)$. The exact solution is given by

Grid size	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
25 × 25	4.27621e-02	–	8.35089e-03	–	1.28952e-01	–
50 × 50	1.48071e-03	4.85	1.53104e-04	5.77	4.68203e-03	4.78
100 × 100	5.97053e-05	4.63	3.61225e-06	5.41	2.67768e-04	4.13
200 × 200	2.61912e-06	4.51	1.05039e-07	5.10	1.71317e-05	3.97

TABLE 3.1. Extension errors for extrapolating the function $f(x, y) = \cos((2\pi(x - 0.5)) \cos(2\pi(y - 0.5))$. The convergence rate r is computed by Eq. (2.30).

$p(x,y,t) = p(x-ut, y-vt)$. The computed solution is exact up to machine precision, as expected. Next we consider initialize with a Gaussian function,

$$f(x,y) = \exp\left(-20\left((x-0.5)^2 + (y-0.5)\right)^2\right).$$

Again, we consider scalar advection with constant velocity $\vec{U} = (0.1, 0.1)$. Figure 3.6 shows error over time computed in the L^1 norm. The scaling of the error appears to be

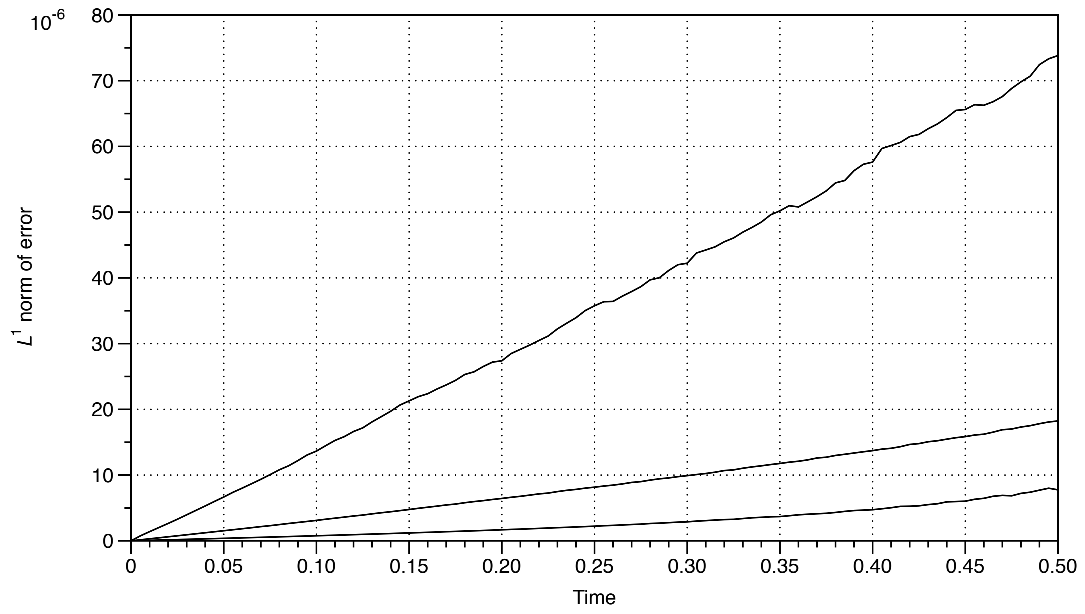


FIGURE 3.6. Time series of the truncation error for an advecting scalar function defined on a mesh with a cut cell discretization. The square grid sizes were 100 (top), 200 (middle), and 400 (bottom). The time step was set to one-fourth of the spatial step.

consistent with second order convergence. The error in the infinity and L^2 norm indicates an instability occurs in a finite amount of time (data not shown). To isolate the error, we advect the scalar function,

$$S(x,y) = \sin(50y)$$

with constant velocity $\vec{U} = (0.5, 0)$. The domain used was a circle with radius 0.2 centered at $(0.5, 0.5)$. The instability is apparent in Fig. 3.7. This problem will be addressed in future work. For our purposes, diffusion will damp high frequency modes that cause the instability.

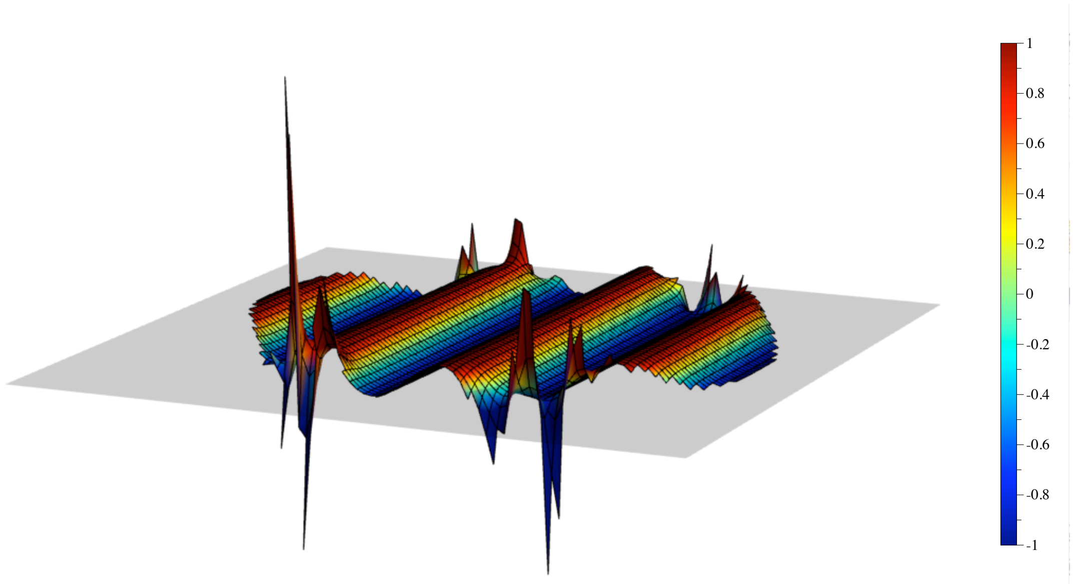


FIGURE 3.7. The growth of an instability at the simulation time value of $t = 0.3$. The grid size used was 150 with a time step equal to one-fourth of the spatial step. The scalar extension from the advection update is shown. The shaded plane is $z = 0$.

3.2.6. Boundary advection scheme. The advection scheme requires concentration values to be located on a Cartesian grid. Information from values on the boundary is lost after the advective step of the operator splitting method. One possible solution is to employ the same bicubic interpolation scheme to extrapolate boundary values from interior values. The new boundary values can be input into the reaction-diffusion solver. Rather than bootstrap these values, we advect the boundary values separately for the purpose of reinitializing the reaction-diffusion solver.

We accomplish this by implementing a method for advecting a scalar on a propagating front originally presented in [**3**]. We summarize as follows. A scalar function $S(x, y, t)$ on a boundary that advects with a specified velocity field $\vec{U}$ and speed function F satisfies the equation,

$$\begin{aligned} \frac{\partial S}{\partial t} &= -\vec{U} \cdot \nabla S - \left(n_y^2 u_x - n_x n_y \left(u_y + u_x\right) + n_x^2 v_y\right) S \\ &\quad -F\left(\vec{n} \cdot \nabla S + \kappa S\right), \end{aligned} \tag{3.7}$$

where $\vec{n} = (n_x, n_y)$ is the normal to the surface at the point (x, y) and κ is the curvature operator. The notation u_x denotes the partial derivative with respect to x of the first component of the velocity vector. The extra second term on the right hand side and curvature terms come from simplifying the conservative advection equation. The derivation can be found in [**3**]. The scalar S is extended off the interface so that finite differences can be used to discretize the various operators. The extension is based on methods for constructing extension velocities presented in [**2**]. A temporary signed distance function ϕ^{temp} is constructed near the zero level set. This function satisfies the equation

$$\nabla\phi^{\text{temp}} \cdot \nabla S_{\text{ext}} = 0. \tag{3.8}$$

The above construction ensures that the level set function ϕ remains the signed distance function when the function propagates with an extension velocity over time. As ϕ^{temp} is constructed, Eq. (3.8) is solved. The numerical solution is computed similar to solving the Eikonal equation Eq. (A.22) with fast marching methods.

3.2.7. Convergence tests for the boundary advection scheme. To test our boundary advection scheme, we numerically solve Eq. (3.7) with a $F = 0.1$, zero velocity field and an initial condition $S(x, y, 0) = 1$. $S(x, y, t)$ is defined on a circular boundary centered at $(0.5, 0.5)$ with radius $r = 0.2$. We initialize ϕ with the exact signed distance function described in Appendix A. The exact solution to this equation is the scalar given by

$$S(x, y, t) = \frac{r}{r + 0.1t}. \tag{3.9}$$

The curvature operator is approximated by centered finite differences as follows,

$$\kappa = \frac{\left(D^{0y}\phi\right)^2 \left(D^{+x-x}\phi\right) - 2\left(D^{0x}\phi\right)\left(D^{0y}\phi\right)\left(D^{0y}D^{0x}\phi\right) + \left(D^{0x}\phi\right)^2 \left(D^{+y-y}\phi\right)}{\left(\left(D^{0x}\phi\right)^2 + \left(D^{0y}\phi\right)^2\right)^{3/2}}. \tag{3.10}$$

Finite difference operators are defined in Appendix A. The error over time is shown in Figure 3.8 and is consistent with second order convergence.

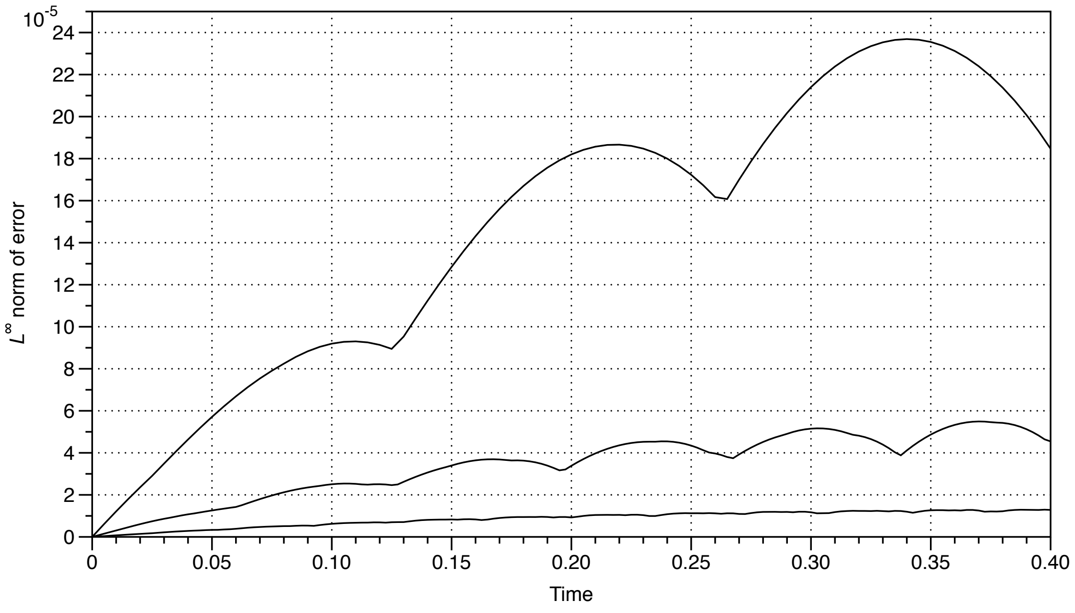


FIGURE 3.8. Time series of the truncation error for a scalar function defined on a circle expanding in the normal direction with a speed function $F = 0.1$. The square grid sizes were 50 (top),100 (middle), and 200 (bottom). The time step was set to one-fourth of the spatial step.

Next we numerically solve Eq. (3.7) with a constant velocity field $\vec{U} = (0.1, 0.1)$, zero speed function and an initial condition $S(x, y, 0) = x$. Again, the boundary that $S(x, y, t)$ is defined on is a circle centered at $(0.5, 0.5)$ with radius $r = 0.2$. The exact solution to this equation is the scalar given by

$$S(x, y, t) = x - 0.1t. \tag{3.11}$$

Figure 3.9 shows the error in the L^2 norm averaged over time. The convergence is consistent with a first order scheme. The extension method is based on fast marching methods, which are first order accurate. We are currently exploring higher-order efficient boundary extensions methods.

3.3. Software implementation

The software organization and data input/output is the same as in Chapter 2. The `State` class is modified to include routines for advection. The `State` class here also

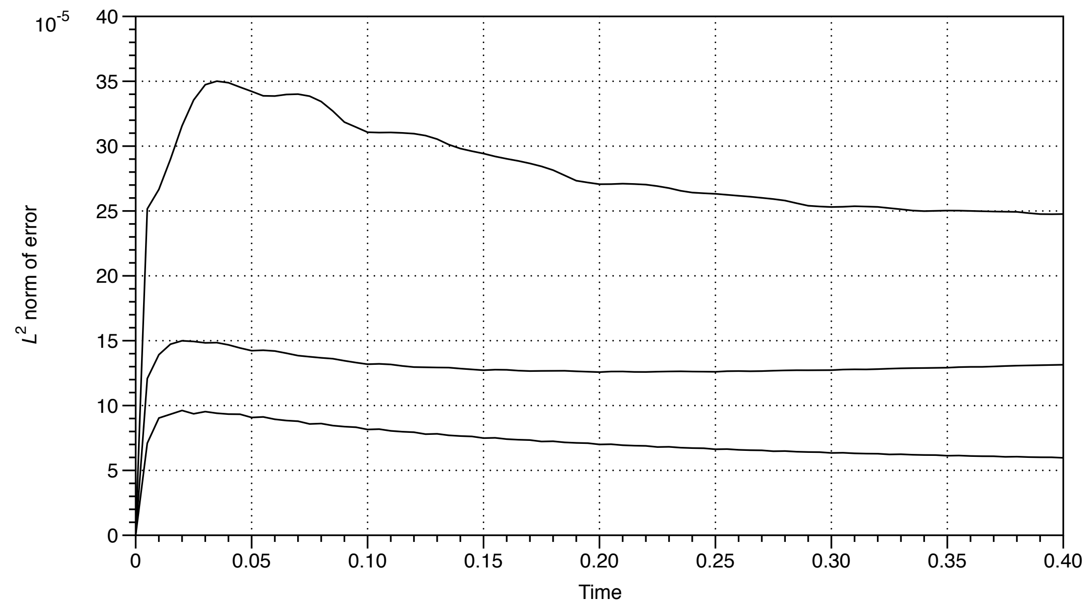


FIGURE 3.9. Time series for the truncation error for scalar function on a boundary advecting with constant velocity. The square grid sizes were 100 (top), 200 (middle), and 400 (bottom). The time step was set to one-fourth of the spatial step.

contains variables for velocity and a speed function. The signed distance function is a variable in the `State`. The `advect` routine updates the signed distance function with the time update in Eq. (3.4) and upwinding scheme [**26**]. Quantities computed with the signed distance that are used for advecting concentration variables, such a normals and curvature, are computed and stored. Compartment information from the previous and current time is passed into another function that advects one species. In this function, a mesh representation of the data is obtained from the array that stores all of the chemical species information. The interior and boundary extension are constructed. The level set update is applied. Centroid values at cut cells are interpolated. Routines from `DTExtensions`, software to interface level set methods with `DTSource`, calculate the boundary extension and second order spatial stencils. Additional routines were written for interpolation, extension, gradient, and curvature calculations. The new data are put into an array that stores the updated state of the system. After every chemical species is updated, the `CombinedSolver` class member of `State` is updated. The DASPK solver is reset for the next reaction-diffusion step. DASPK must compute consistent initial conditions for the DAE system every time the solver is reset after a step of advection.

3.4. Convergence tests for the hybrid algorithm

To test the numerical method, the computed solution to the advection-reaction-diffusion equation is compared to the fixed boundary solution computed by the cut cell algorithm in Chapter 2. Given a specified constant velocity field $\vec{U} = (u, v)$ and initial condition $f(x, y, 0)$, the advection-reaction-diffusion equation is numerically solved until t^{end}. To compare to the fixed boundary solution, the reaction-diffusion solver is initialized with the value $f(x - ut^{\text{end}}, y - vt^{\text{end}})$ and the exact boundary $\Gamma(t^{\text{end}})$. The solution on a fixed domain is solved from $t = 0$ until t^{end} for comparison to the advection-reaction-diffusion solution. In the following tests, the ratio $\Delta x/\Delta t = 4$ is fixed. During a grid refinement, the time step is also reduced by the same factor as the spatial step.

First we test our method by solving an advection-diffusion equation propagating with constant velocity. In our first example, the initial condition is a Gaussian function

$$f(x, y) = \exp\left(-100\left((x - 0.5)^2 + (y - 0.5)^2\right)\right).$$

The boundary condition is no flux on the boundary $\Gamma(t)$, which is given by the zero level set of the signed distance function ϕ. A time series from the computed solution to the advection-diffusion equation is shown in Figure 3.10. The truncation error in several norms and convergence rates are listed in Table 3.2. The norms are defined as in [**32**],

$$||e||_p = \left(\sum_{(i,j)\in\Omega} |e_{i,j}|^p V_{i,j} \Big/ \sum_{(i,j)\in\Omega} V_{i,j}\right)^{1/p}, \tag{3.12}$$

and the infinity norm is defined as the maximum value over the domain. The convergence rate r is given by Eq. (2.30). The data presented in a log-log graph in Fig. 3.11 suggest the method is second order accurate.

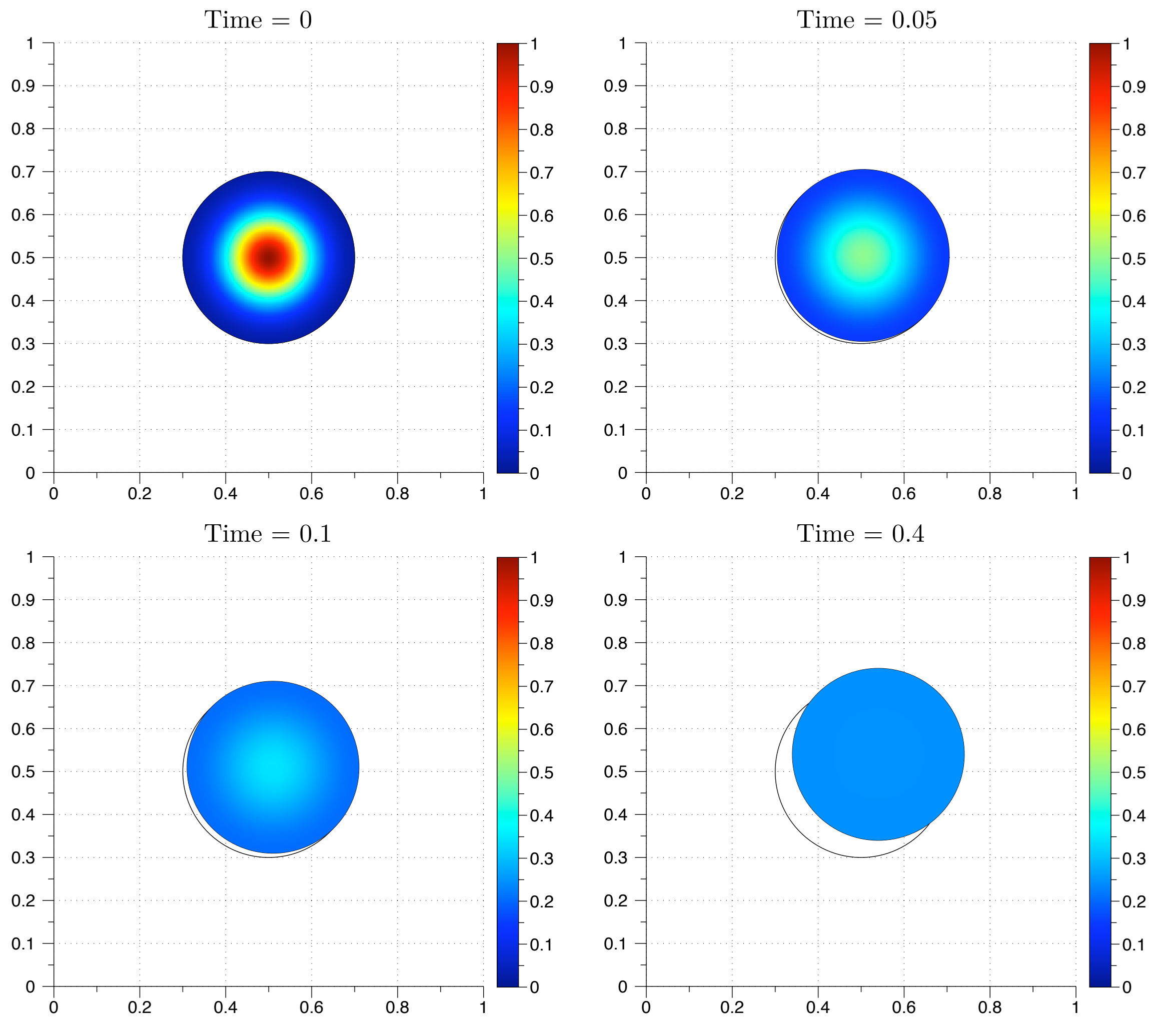


FIGURE 3.10. Time values from the computed solution of an advection-diffusion equation solved on a circle with a Gaussian initial condition. The diffusion coefficient for this simulation was 0.05. The grid size is 200 × 200. The initial boundary is the circle indicated by a solid black line.

Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.00e-03	1.98e-03	–	1.62e-03	–	2.97e-03	–
100 × 100	2.50e-03	2.42e-04	3.03	2.36e-04	2.78	3.27-e04	3.18
200 × 200	1.25e-03	4.83e-05	2.33	4.70e-05	2.33	6.76e-05	2.28
400 × 400	6.25e-04	1.05e-05	2.20	1.02e-05	2.20	1.50e-05	2.17

TABLE 3.2. The norms and convergence rates for the advection-diffusion equation at the time value of 0.3.

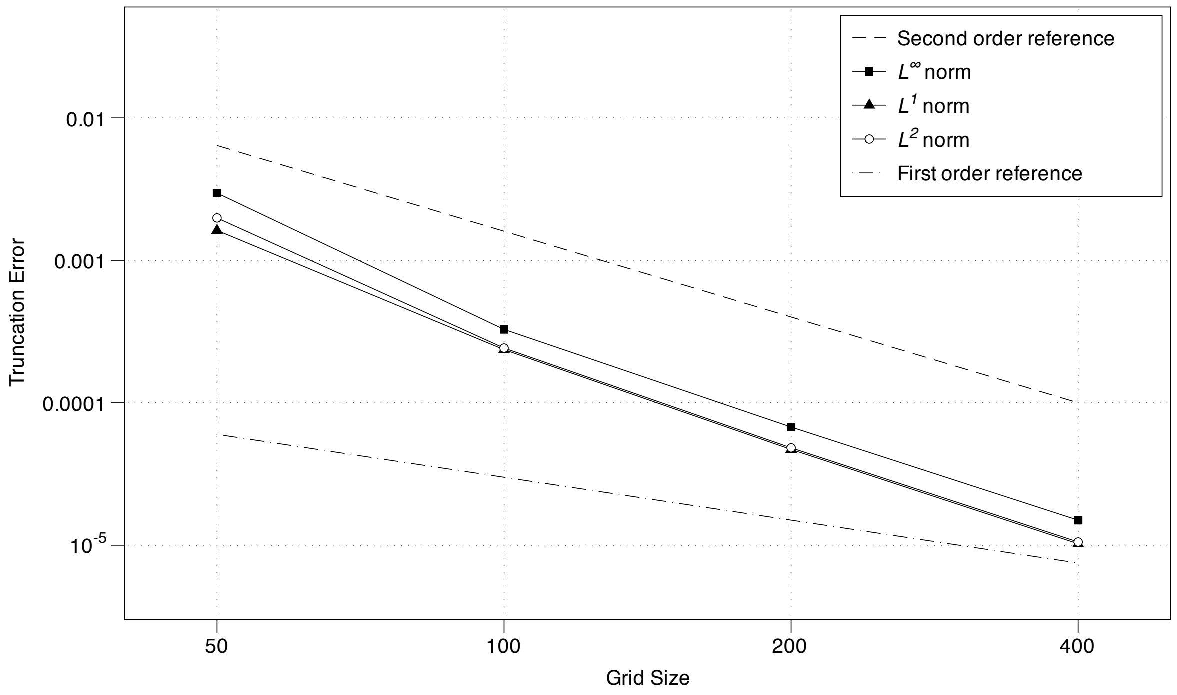


FIGURE 3.11. Truncation error for the advection-diffusion equation at the time value of 0.3. The convergence data are the same as given in Table 3.2.

For our next example, we consider a two species system of advection-reaction-diffusion equations. The reaction-diffusion terms in the system are given in Eqs. (2.31) and (2.32). The equations model phosphorylation-dephosphorylation of a protein with Michaelis-Menten kinetics. The initial domain is

$$\Gamma(r, \theta, 0) = r \leq 0.2964\,(1 - 0.3\sin(4\theta))\,. \tag{3.13}$$

The values for constants used in the simulation are listed in Table 2.2. Figure 3.12 shows a time series from the model simulation. Chemical species are advected with the constant velocity field $\vec{U} = (0.1, 0.1)$. The convergence data was generated as in the previous example. The moving boundary solution was subtracted from the fixed boundary solution to the reaction-diffusion equation. The convergence data are listed in Table 3.3. A log-log plot of the truncation error is given in Fig. 3.13. Power function fits to the convergence data in various norms are listed in Table 3.4. The data suggest

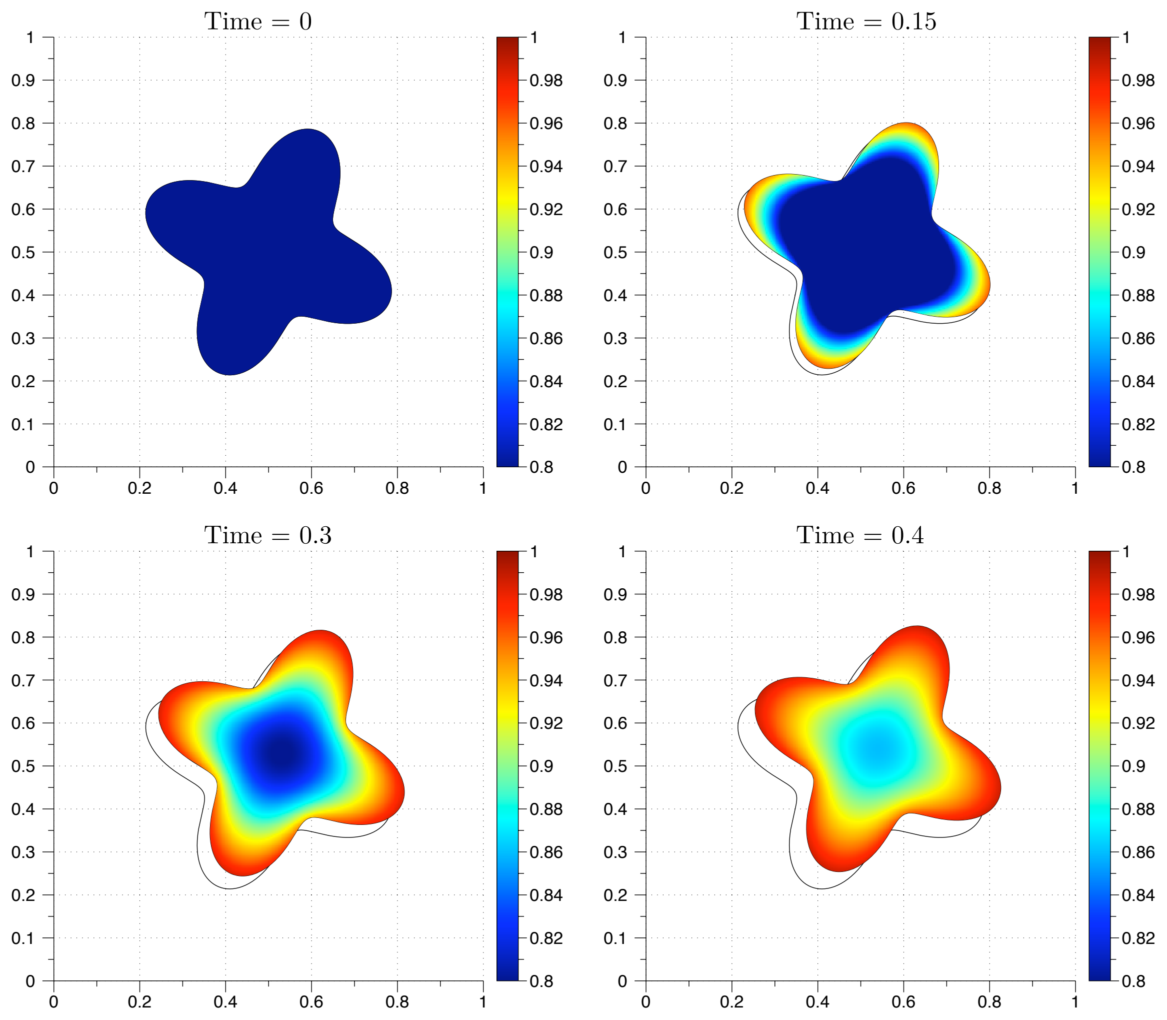


FIGURE 3.12. Time values from the computed solution of an advection-reaction-diffusion equation solved on a moving boundary. The grid size is 200 × 200. The initial boundary is indicated by a solid black line.

the order of the method is reduced to first order. There are several sources of error to consider that were not encountered with the previous example. First, the boundary extension in the method uses first order interpolation. In [**3**] second order accuracy is shown. However, the boundary extension in our method is recalculated after every time step, which is not the case in [**3**]. Also, values on the boundary lie on a piecewise linear approximation to the true boundary. Boundary values in [**3**] lie on the boundary given by the zero level set. Another source of error is the centroid approximation. In our method, bicubic interpolation gives the values to initialize cut cells for the reaction-diffusion solver.

Species C_i							
Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.000e-03	7.54e-03	–	6.95e-03	–	1.10e-02	–
100 × 100	2.500e-03	2.68e-03	1.49	2.50e-03	1.48	3.96e-03	1.47
200 × 200	1.250e-03	1.12e-03	1.25	1.05e-03	1.25	1.69e-03	1.23
400 × 400	6.250e-04	4.98e-04	1.17	4.67e-04	1.17	7.55e-04	1.16
Species C_a							
Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.000e-03	7.56e-03	–	6.95e-03	–	1.10e-02	–
100 × 100	2.500e-03	2.74e-03	1.47	2.50e-03	1.48	3.96e-03	1.47
200 × 200	1.250e-03	1.12e-03	1.28	1.05e-03	1.25	1.69e-03	1.23
400 × 400	6.250e-04	5.04e-04	1.16	4.67e3-04	1.17	7.55e-04	1.16

TABLE 3.3. The norms and convergence rates for the two species model advecting with constant velocity at the time value of 0.4.

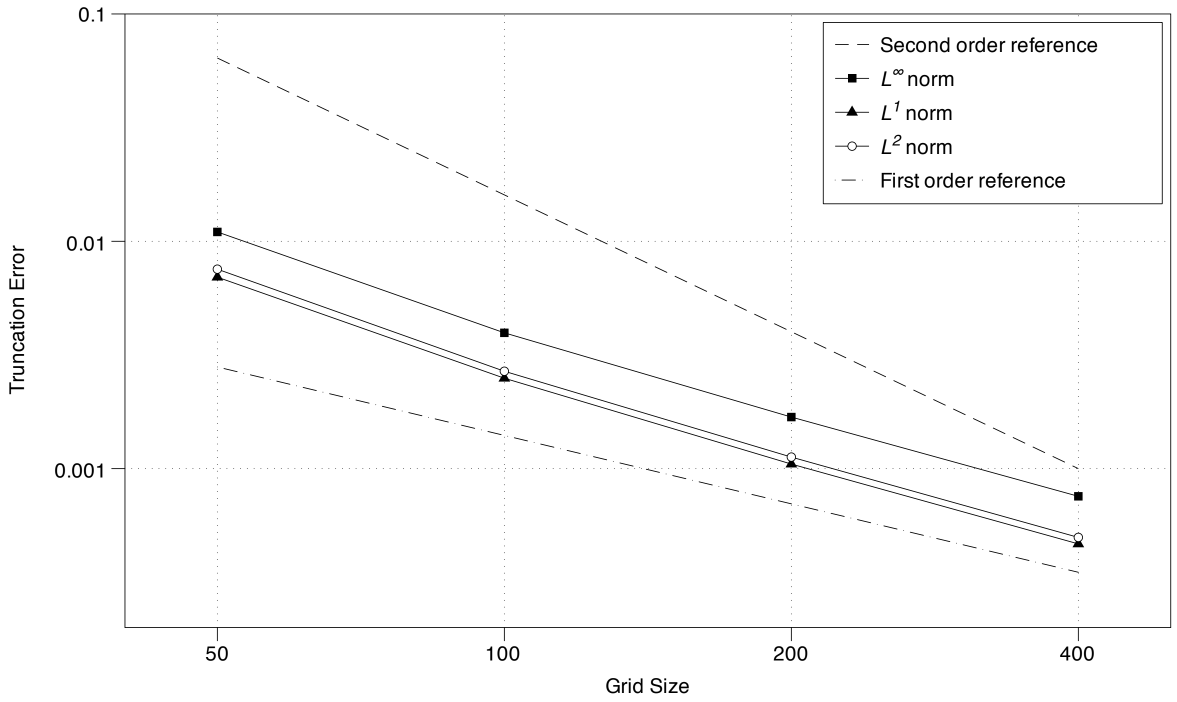


FIGURE 3.13. Truncation error for the advection-reaction-diffusion equation at the time value of 0.4. The convergence data are the same as given in Table 3.3. Power fits for the truncation errors are given in Table 3.4.

Initialization with centroid values is an approximation to the integral of the function over the control volume. Higher order embedded boundary methods are suggested in [**45**]. Higher order flux construction and cut cell initialization may be needed to increase the convergence rate.

Species C_i Truncation Error	
Norm	**Power Fit**
L^2	$y = 1.15x^{-1.30}$
L^1	$y = 1.04x^{-1.29}$
L^∞	$y = 1.56x^{-1.28}$

TABLE 3.4. Power fit of the truncation error from the two species model at the time value of 0.4.

Next we consider the integral of both chemical species over time. In Chapter 2, the integral is conserved to machine precision for the system. This is not the case for the moving boundary problem. Fig. 3.14 shows the integral sampled over time subtracted from the integral of the initial condition. The change in area converges to the initial value with second order accuracy.

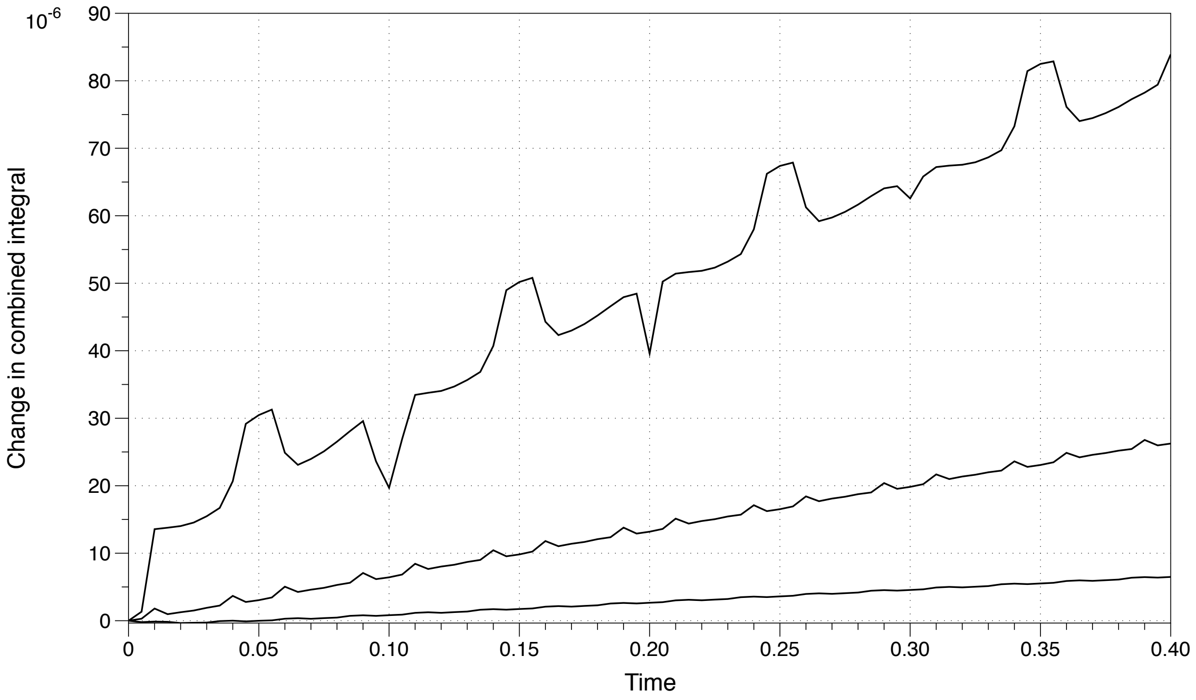


FIGURE 3.14. Numerical integral of total concentration subtracted from the initial value. The top plot is for a grid size of 100×100 followed by 200×200 and 400×400 on the bottom.

To demonstrate the method on a boundary propagating with non-constant velocity, we simulated the two species model (Eqs. (2.31) and (2.32)) propagating with the velocity field

$$\vec{U} = 0.4 \begin{pmatrix} x - 0.5 \\ 0.5 - y \end{pmatrix}. \tag{3.14}$$

Fig. 3.15 shows the computed solution at several time values. For this example, the error is defined as the difference between successive grid refinements as in Chapter 2. The truncation error function $E(x, y, t)$ is defined on interior values on the course grid. Computed solution values located in course grid cut cells are excluded from the domain. This includes some values located in interior points for the more refined grid. The truncation error function is defined as

$$E(x, y, t) = c_{\Delta x}(x, y, t) - c_{\Delta x/2}(x, y, t). \tag{3.15}$$

The course grid values are located in the center of a box defined by four refined grid values. Four refined grid values are averaged and subtracted from one course value. Table 3.5 lists convergence data for this system, which is similar to the constant velocity convergence data for the same system. We conclude that complicated boundary dynamics lower the convergence rate of the method to first order.

Species C_i							
Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.000e-03	–	–	–	–	–	–
100 × 100	2.500e-03	2.79e-03	–	3.05e-04	–	3.51e-03	–
200 × 200	1.250e-03	9.53e-04	1.55	1.10e-04	1.47	1.28e-03	1.45
400 × 400	6.250e-04	4.21e-04	1.18	5.00e-05	1.14	5.65e-04	1.18
Species C_a							
Grid size	**Time step**	L^2 **norm**	r	L^1 **norm**	r	L^∞ **norm**	r
50 × 50	5.000e-03	–	–	–	–	–	–
100 × 100	2.500e-03	2.79e-03	–	3.05e-04	–	3.51e-03	–
200 × 200	1.250e-03	9.53e-04	1.55	1.10e-04	1.47	1.28e-03	1.45
400 × 400	6.250e-04	4.21e-04	1.18	5.00e-05	1.14	5.65e-04	1.18

TABLE 3.5. The norms and convergence rates for the two species model advecting with non-constant velocity at the time value of 0.4.

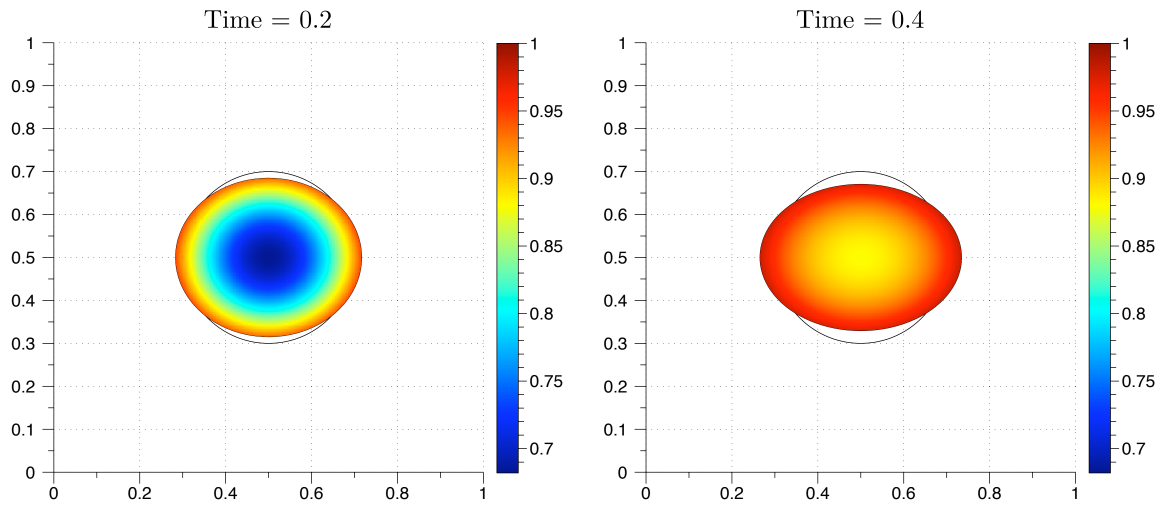


FIGURE 3.15. Time values from the computed solution of an advection-reaction-diffusion equation solved on a moving boundary advecting with non-constant velocity. The initial boundary is a circle indicated by a solid black line. The grid size is 200 × 200.

3.5. Substrate depletion model

Generating robust spatial patterns is important for many problems in developmental biology, and occurs in many biological systems. Pattern formation can be established through local activation and global inhibition of reacting chemical species. Turing originally proposed the idea that symmetry breaking can occur via a diffusion driven instability, leading to spatial patterns of interacting chemical species [**79**]. Turing instabilities have been proposed as a mechanism to explain pattern formation in many systems including morphogenesis in *Drosophila* [**11**], animal coat markings [**54**], and the rapid polarization of proteins involved in the yeast mating response pathway [**25**]. It is well known that domain size is a bifurcation parameter in Turing systems [**7**]. Geometry also contributes to pattern formation [**11, 54**]. In this section we explore the effects of a deforming geometry on pattern formation for a model system.

In the model system [**31**], a Turing instability is caused by a slow diffusing protein locally activated through positive feedback. This protein depletes the fast diffusing substrate [**22**]. We model a signaling protein that exists in two forms: a membrane bound active guanosine triphosphate (GTP)-bound form and a cytosolic, inactive GDP-bound form. Let $[R_i]$ and $[R_a]$ denote the inactive and active concentrations. The active form for this type of protein is known to diffusion slower than the cytosolic inactive form [**82**]. We assume a top down view of a three dimensional cell as in [**18, 50**] where membrane bound GTP-forms are modeled with smaller diffusion coefficient.

The substrate depletion model is given by the following equations,

$$\frac{\partial [R_i]}{\partial t} + \vec{U} \cdot \nabla [R_i] = D_1 \Delta [R_i] + \gamma \Big(- d_0 [R_i] - \frac{\left(k_2 + k_{2p} [R_a]^2\right) [R_i]}{k_{m1} + [R_i]} + k_{2r} [R_a] + k_0 \Big) \tag{3.16}$$

$$\frac{\partial [R_a]}{\partial t} + \vec{U} \cdot \nabla [R_a] = D_2 \Delta [R_a] + \gamma \Big(- d_0 [R_a] + \frac{\left(k_2 + k_{2p} [R_a]^2\right) [R_i]}{k_{m1} + [R_i]} - k_{2r} [R_a] \Big). \tag{3.17}$$

The boundary conditions are no flux for both chemical species.

First, we allow the system to reach steady state on an initial geometry, then advect the boundary and chemical species with a nonzero velocity field. For the first 200 time units, the simulation is run with the velocity $\vec{U} = \mathbf{0}$. The initial domain is a circle centered at the point $(0.5, 0.5)$ with radius $r = 0.2$. The initial conditions are

$$\begin{aligned} [R_i](x, y, 0) &= 2 + 0.2 \Big(\exp\left(-50\left((x-0.7)^2 + (y-0.5)^2\right)\right) \\ [R_a](x, y, 0) &= 0. \end{aligned} \tag{3.18}$$

The Gaussian terms in the initial concentration of $[R_i]$ represent small perturbations to the spatially homogenous initial condition. Table 3.6 lists the values of constants used in the model simulation. The steady state solution to the substrate depletion model on a circlular domain is spatially homogeneous in spite of the initial perturbation, and is

Constant	**Value**	**Constant**	**Value**
D_1	5.00	k_{2p}	7.00
D_2	0.05	k_{2r}	4.00
d_0	0.03	k_{m1}	1.65
k_0	0.10	γ	1.2
k_2	3.00		

TABLE 3.6. Values of constants in the first substrate depletion model simulation.

given by

$$\begin{aligned} [R_i](x,y,200) &= 0.376138 \\ [R_a](x,y,200) &= 2.950538. \end{aligned} \tag{3.19}$$

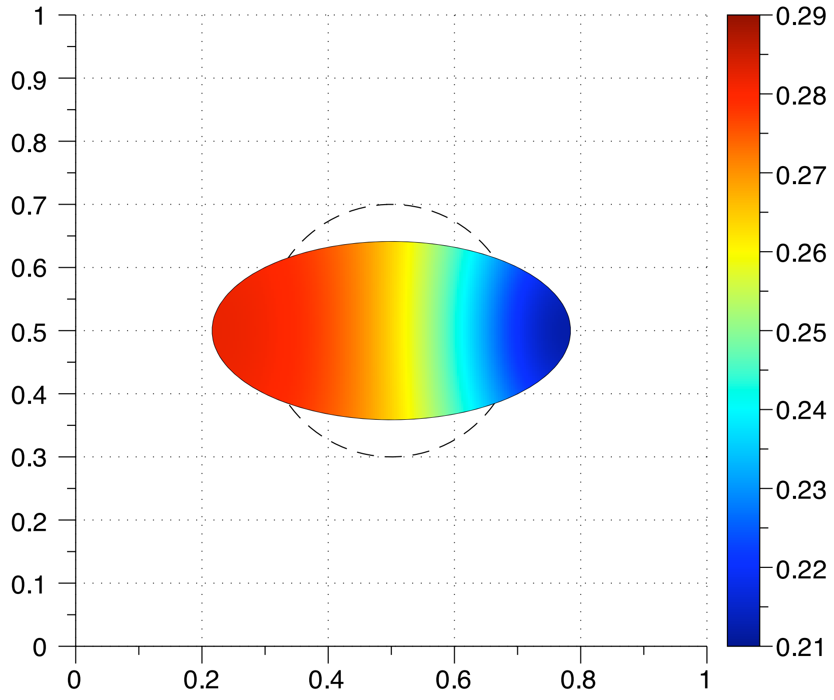
The steady state solution is robust to large spatial perturbations. We conclude that no diffusion driven Turing instability forms for the circular geometry with the specified parameters. To investigate the effects of geometry on the stability of Eqs. (3.16) and (3.17), we simulated the equations with the velocity field

$$\vec{U} = 0.01\,(x-0.5, 0.5-y)\,, \tag{3.20}$$

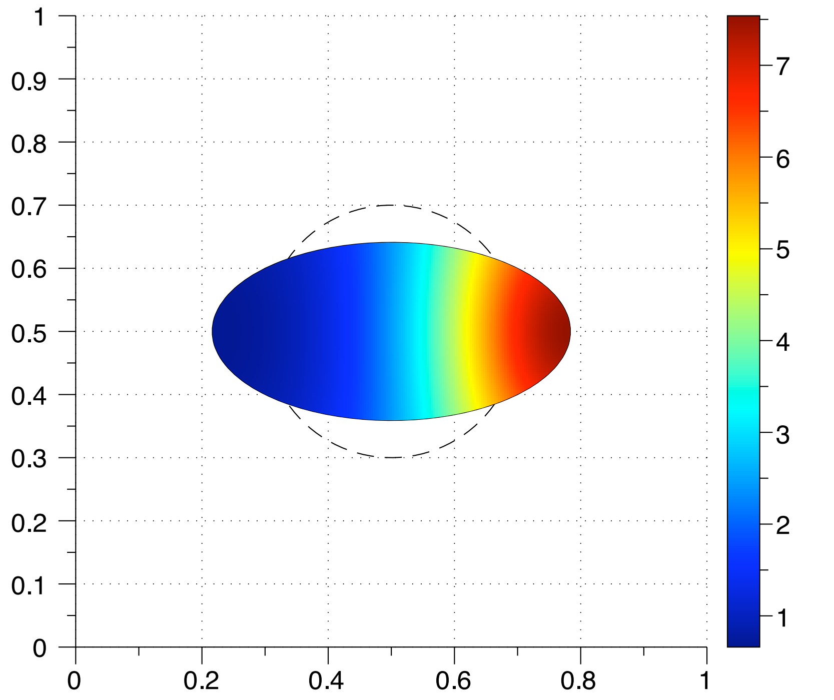
with initial concentrations and cellular geometry from the steady state solution Eq. (3.19). As the boundary deforms, an instability forms in an area of high curvature (Figs. 3.16(a) and 3.16(b)). Fig. 3.17 shows the active concentration through the center of the horizontal domain at time values near the formation of the instability. The area of the domain is remains unchanged, but the arc length increases. The geometric change drives the system into forming a regime where the spatially homogeneous steady state is unstable.

Next, we consider the case with two initial perturbations,

$$\begin{aligned} [R_i](x,y,0) &= 2 + 0.2\Big(\exp\left(-500\left((x-0.35)^2+(y-0.5)^2\right)\right) \\ &\qquad + \exp\left(-500\left((x-0.65))^2+(y-0.5)^2\right)\right)\Big) \\ [R_a](x,y,0) &= 0. \end{aligned} \tag{3.21}$$



(a) Concentration of inactive R.



(b) Concentration of active R.

FIGURE 3.16. Concentration of R at $t = 35$. The initial geometry is indicated by a dashed line. The grid size used was 100×100.

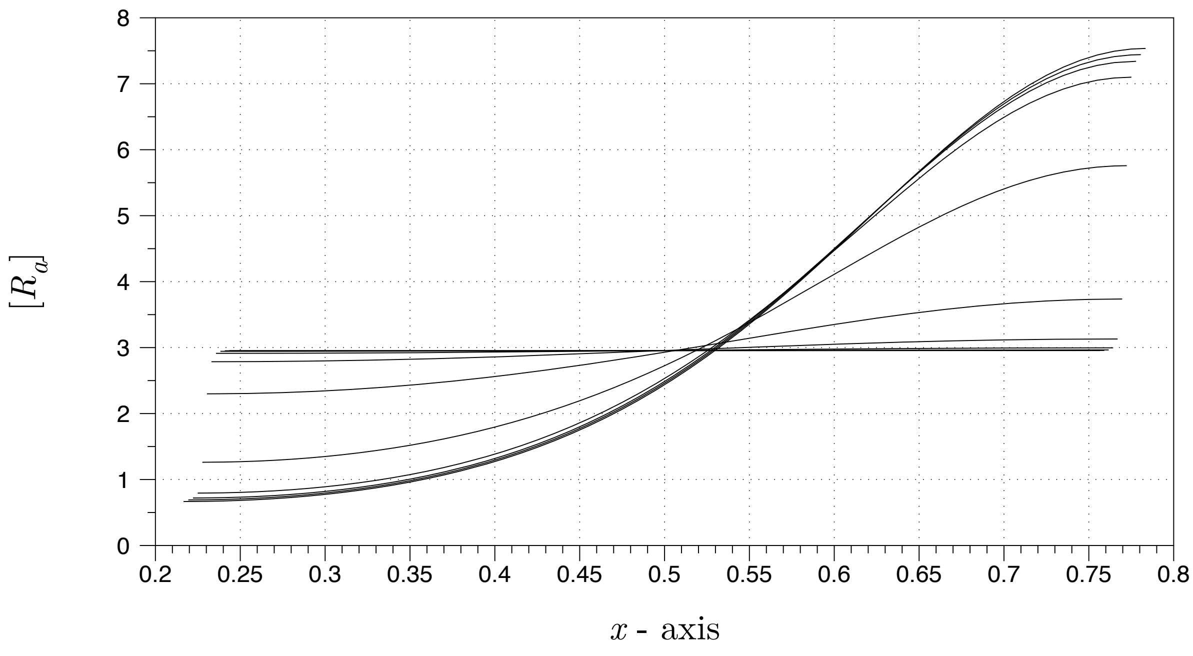


FIGURE 3.17. A slice through the mesh of active R at line $y = 0.5$ for $t \in [25, 35]$.

When the system is simulated with parameter values in Table 3.7, a transient perturbation forms on the right side of the domain before the instability forms on the left side (Fig. 3.18). We enforce a boundary deformation for $t \in [13.5, 14]$ given by the velocity

Constant	**Value**	**Constant**	**Value**
D_1	5.00	k_{2p}	7.00
D_2	0.05	k_{2r}	4.00
d_0	0.02	k_{m1}	1.88
k_0	0.10	γ	17
k_2	3.00		

TABLE 3.7. Values of constants in the second substrate depletion model simulation.

field,

$$\vec{U} = 0.7\,(x - 0.5, 0.5 - y)\,. \tag{3.22}$$

After $t = 14$, the simulation runs to a steady state profile. Fig. 3.19 shows the perturbation on the right is enforced by the boundary deformation. The results are independent of deformation speed.

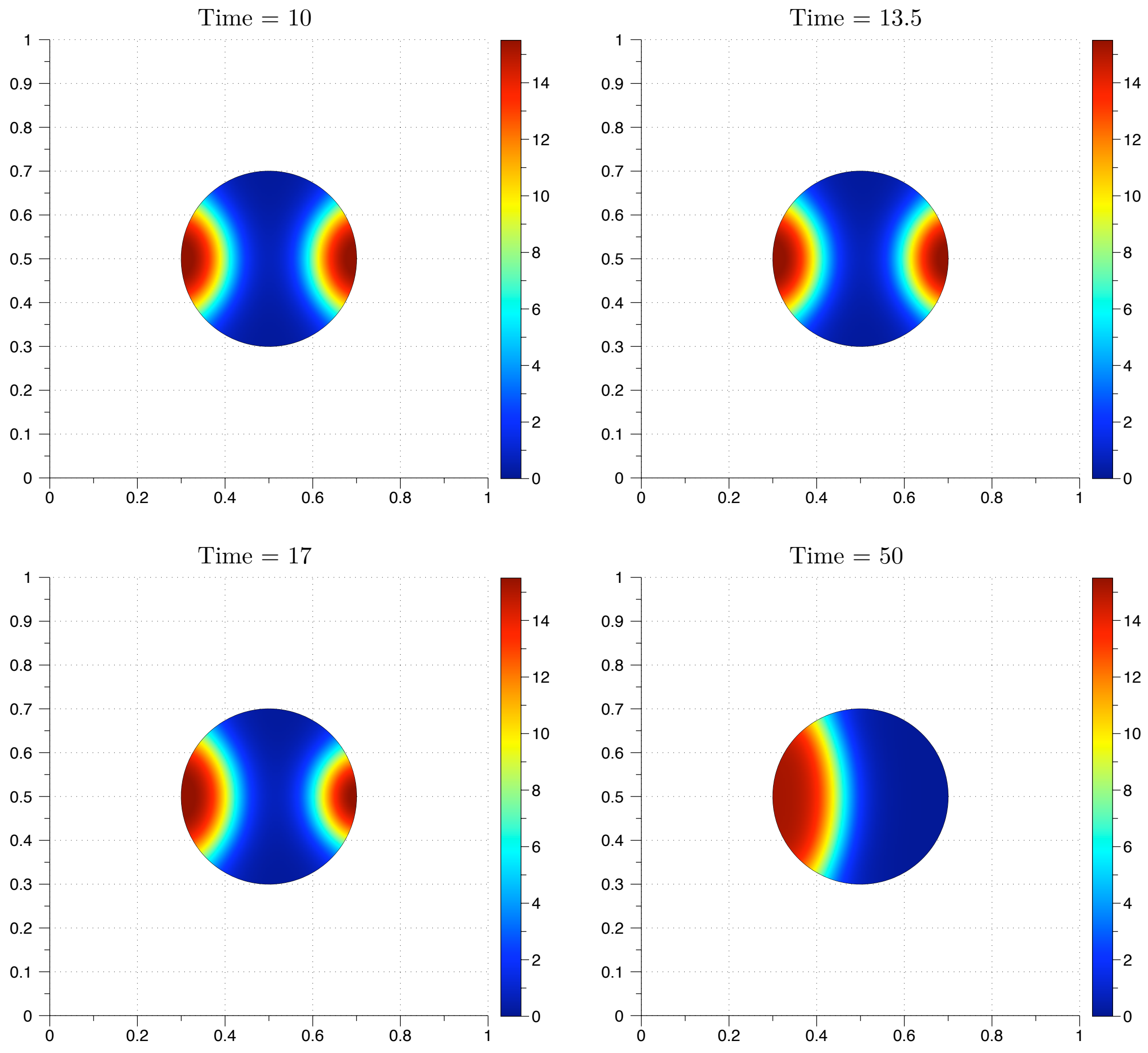


FIGURE 3.18. Time values from $[R_a]$ in the substrate depletion model with two initial perturbations and $\vec{U} = \mathbf{0}$. The grid size used was 150×150.

3.6. Rho family GTPase model

In this section, we model a crawling fibroblast to investigate the role of cell morphology and speed of migration on signaling proteins. We use the same model from Chapter 2 with six species: inactive and active forms of $Cdc42$, *Rac*, and *Rho*. $Cdc42$ is the master regulator of cell polarity that triggers downstream effectors leading to the polarization of *Rac* and *Rho*. The active form of these proteins leads to actin polymerization and cytoskeletal reorganization. *Rac* generates protrusive forces, while $Cdc42$ generates

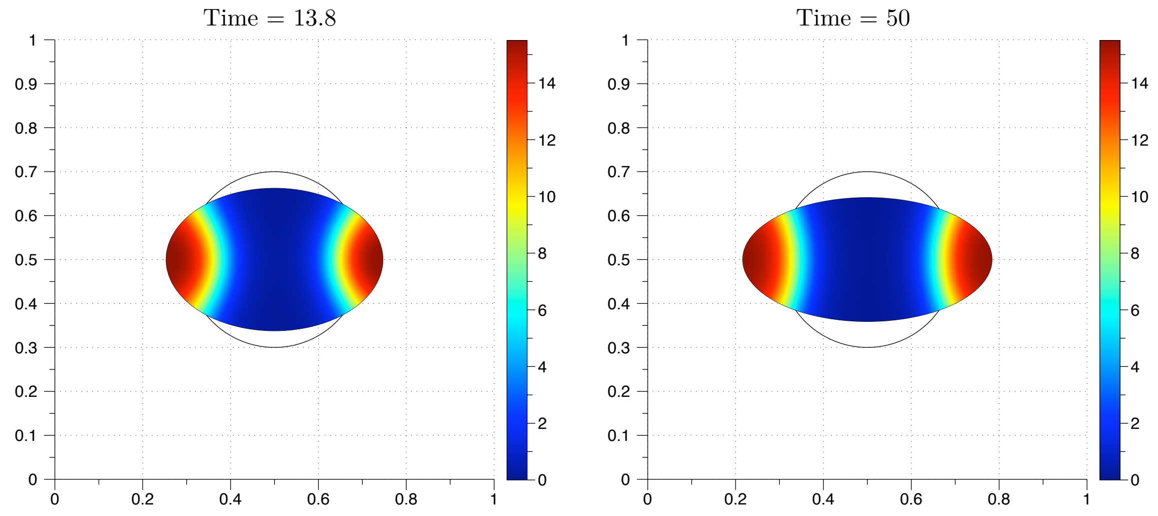


FIGURE 3.19. Time values from $[R_a]$ in the substrate depletion model with two initial perturbations. The boundary deforms from $t = 13.5$ to 14. Earlier time values are the same as in Fig. 3.18. The initial geometry is indicated with a solid line. The grid size used was 150×150.

filopodia. *Rho* activity is associated with the formation of stress fibers, focal adhesions, and myosin-based contractility [**4, 12**].

For simplicity, we concentrate on the basic hypothesized biochemical interactions proposed in [**12**]. The model equations are given by Eqs. (2.37) - (2.40). The schematic of interactions is illustrated in Fig. 2.6. In the model, an extracellular signal such as platelet-derived growth factor (PDGF) leads to activation of $Cdc42$ on the cell edge [**30**]. The active form of $Cdc42$ activates *Rac*. In turn, *Rac* activates *Rho*. Active *Rho* deactivates *Rac*. The active forms of *Rac*, *Rho*, and $Cdc42$ are deactivated in in the cell interior. We impose a velocity field that models active transport by molecular motors inside the cell. The cell migrates with a velocity field the stretches the cell in the y direction, and compresses and translates the cell in the positive x direction,

$$\vec{U} = c\Big(0.7\,(0.5 - x) + 0.5, 0.7\,(y - 0.5)\Big), \tag{3.23}$$

where c is a scaling constant. The velocity field is consistent with the movies of migrating fibroblasts from [**62**]. In the simulation, the initial conditions are 0 for the active species $Cdc42_a$, Rac_a, and Rho_a. The initial condition for the inactive species $Cdc42_i$, Rac_i, and

Rho_i is 1. We simulate polarization in response to an extracellular signal before migration by simulating the reaction-diffusion equation until a stable gradient forms (Fig. 3.20). Then we simulate migration by including an advective term with the velocity field from Eq. (3.23). The constants used in the simulation are listed in Table 3.8. The diffusion coefficient used for inactive species was 0.1. The diffusion coefficient for the active species was 0.05. The initial geometry is based on live cell images of a migrating fibroblast from [**62**]. Figures 3.21 - 3.23 show the concentration profiles after advecting for 150 time steps. The reaction-diffusion equation was solved from simulation time $t = 0$ until $t = 3.25$. The advection term was added with the constant c equal to 2. The advection-reaction-diffusion equation was simulated from $t = 3.25$ until $t = 3.5$. In general, the gradient in the concentration profiles obtained by solving the advection-reaction-diffusion equation is steeper than the steady state concentration profiles. The maximum values of the active forms of *Rac* and *Rho* are located where the initial protrusion was located whereas the maximum is located at the bottom of the cell in the steady state concentration profiles (Figs. 3.22 and 3.23). This suggests that migration speed can play a role in enhancing a transient signal from a thin protrusion with high phosphorylation levels.

Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
S	1.0	k_8	3.0	K_{m3}	0.2
k_1	5.0	k_9	5.0	K_{m4}	0.2
k_2	3.0	K_{m1}	0.2	k_{m5}	0.2
k_3	1.0	k_{m2}	0.2	k_{m6}	0.2
k_4	3.0	K_{m7}	0.2	K_{m11}	0.2
k_5	3.0	K_{m8}	0.2	K_{m12}	0.2
k_6	5.0	k_{m9}	0.2	k_{m13}	0.2
k_7	1.0	k_{m10}	0.2	k_{14}	0.2

TABLE 3.8. Reaction constants used in the simulation of the Rho GTPase model.

3.7. Conclusions

We have presented a novel numerical method to simulate systems of advection-reaction-diffusion equations in complex time-dependent geometries. The examples presented in this chapter have used divergence-free velocity fields, but the method is not

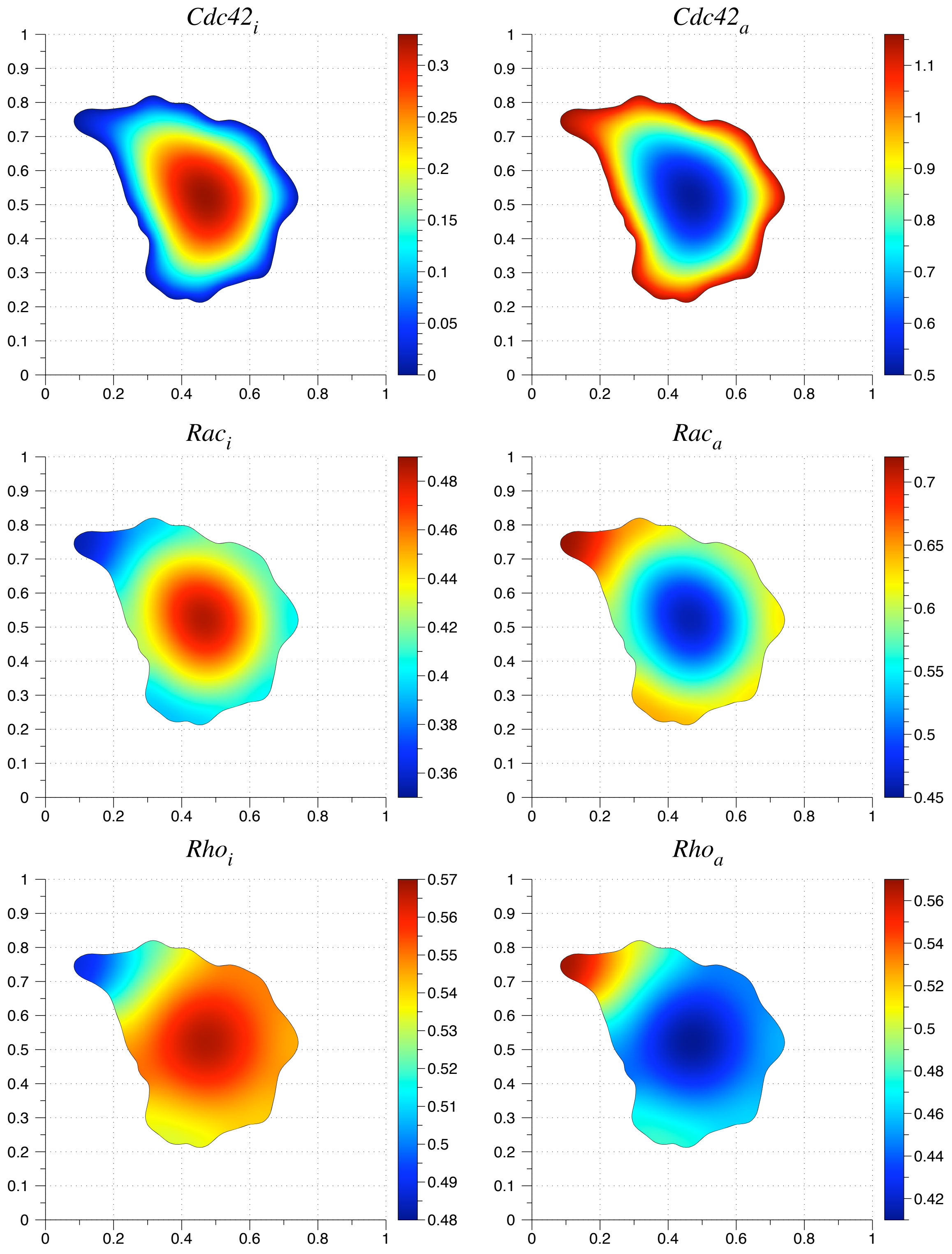


FIGURE 3.20. Initial condition for the advection-reaction-diffusion equation. The grid size used was 150. The time step was set to one-fourth of the spatial step size.

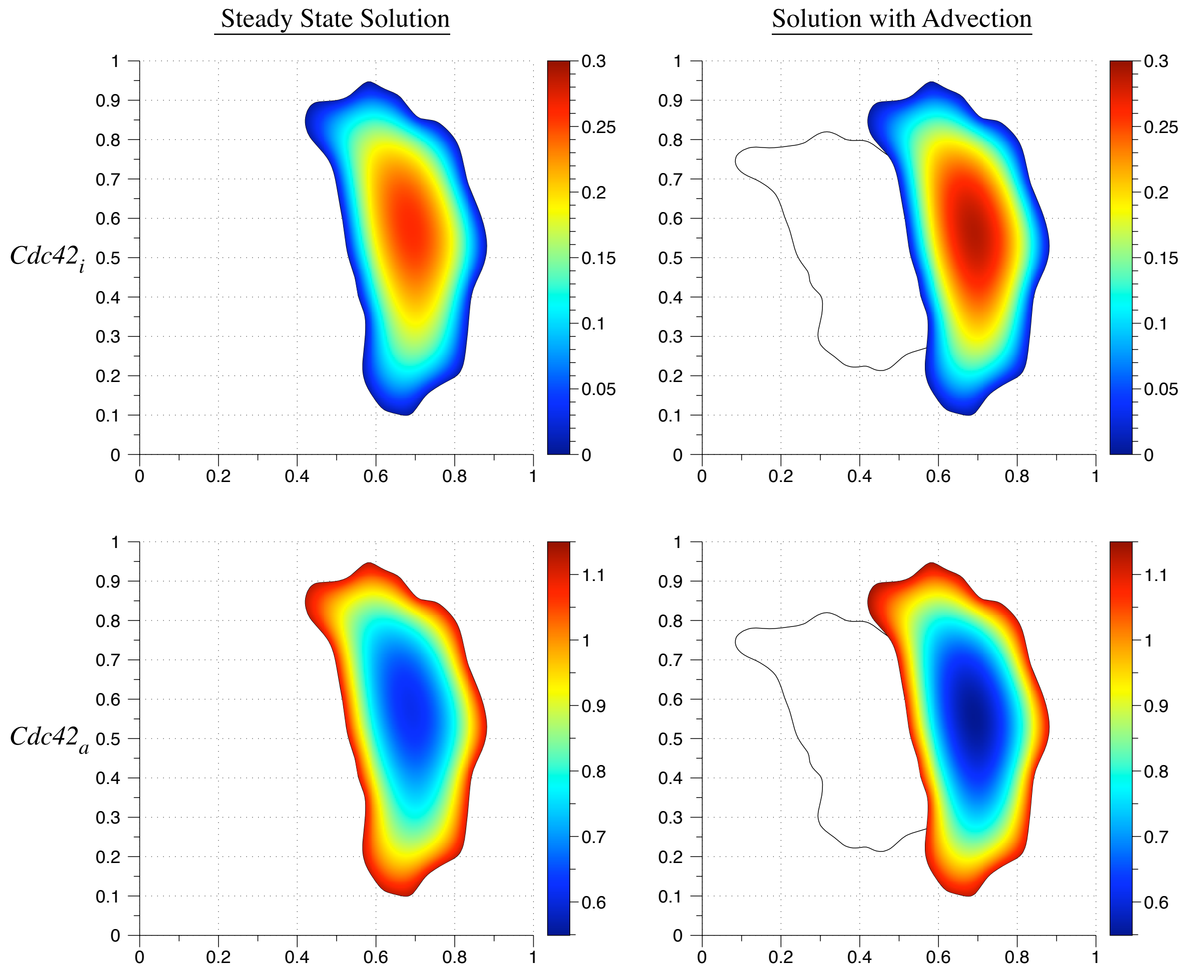


FIGURE 3.21. A comparison of the steady state $Cdc42$ concentration values with the computed solution to the advection-reaction-diffusion equation. The solid line indicates the initial cell geometry.

limited to a specific type of velocity field. Although the method is only approximately numerically conservative, it matches the conservation properties of level set methods which scale with second order accuracy. Because the methods presented here are based on a Cartesian grid, they can be extended to three-dimensions. The numerical accuracy of the method is overall first order. For some problems, such as the advection-diffusion equation in section 3.4, second order accuracy can be achieved. We discuss ideas for improving the accuracy in Chapter 5.

In the examples presented here, our main goal was to observe morphological effects on biochemical protein concentrations. Cell motility is a complex biophysical process. A

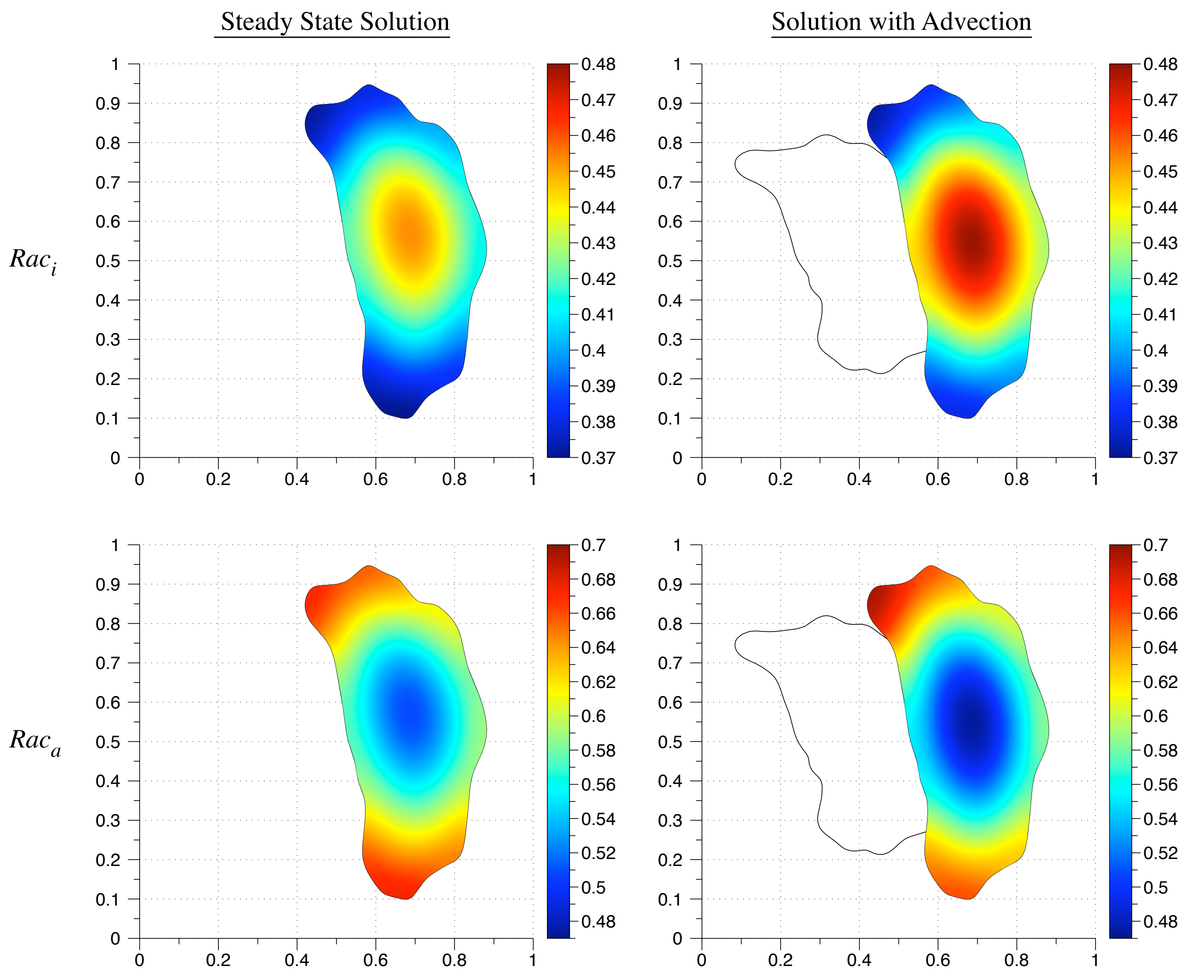


FIGURE 3.22. A comparison of the steady state *Rac* concentration values with the computed solution to the advection-reaction-diffusion equation. The solid line indicates the initial cell geometry.

realistic mathematical model must take into account the mechanochemical events leading to cell protrusion, extension, and retraction. In the future, the methods presented here will be coupled with a mechanical model driving the boundary deformation.

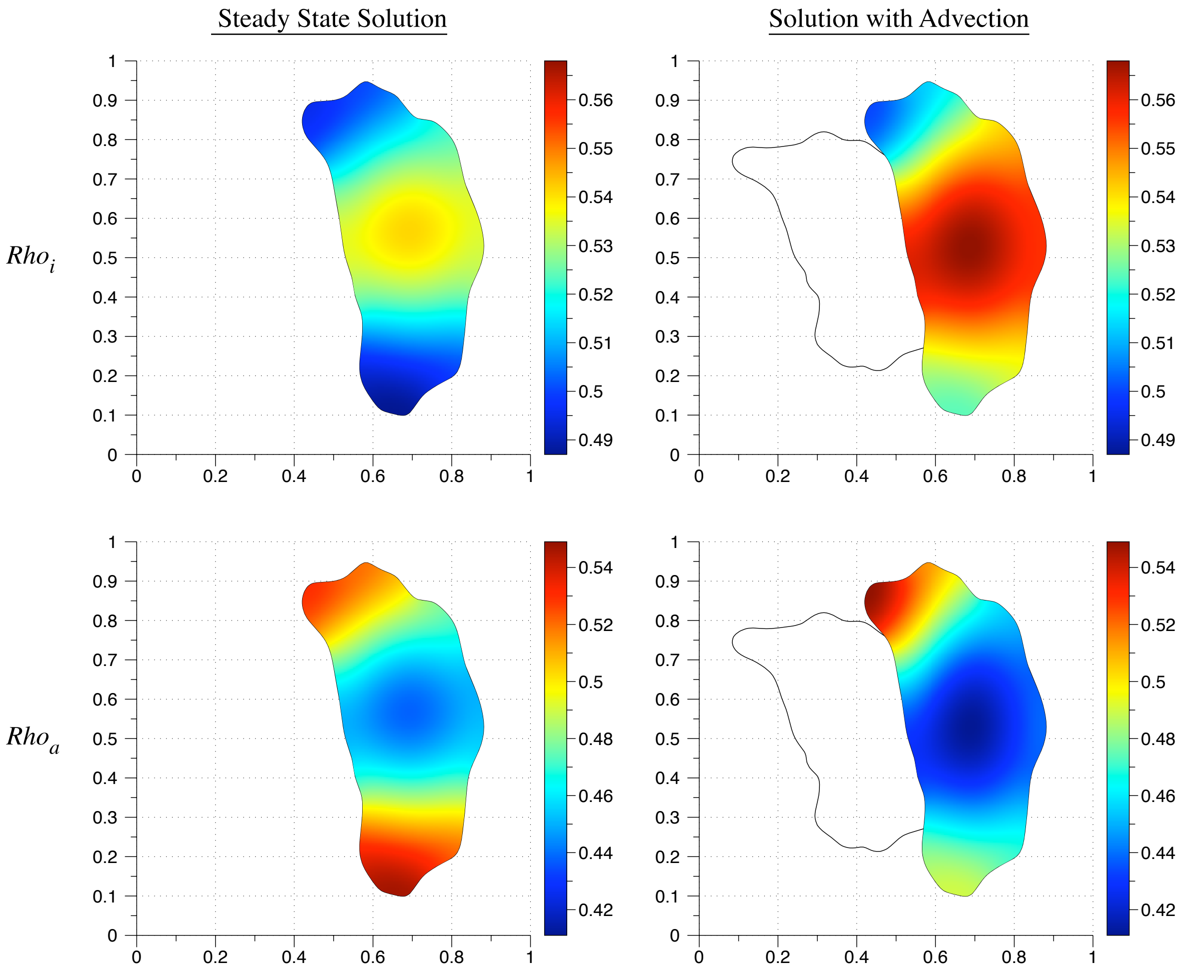


FIGURE 3.23. A comparison of the steady state *Rho* concentration values with the computed solution to the advection-reaction-diffusion equation. The solid line indicates the initial cell geometry.

CHAPTER 4

Modeling Transient Anchorage of Membrane Proteins

Elucidation of signal transduction from membrane proteins into the cell interior is an important question in cell biology. Signaling molecules bind to receptors on the target cell's surface and initiate a signaling cascade. Glycosyl-phosphatidylinositol (GPI) anchored proteins (GPIAPs) are membrane proteins located in many different cell types and tissues. These proteins are important because they have diverse functions in eukaryotic cells, such as signal transduction, prion disease pathogenesis, and immune response [**61**]. In a study of the disease paroxysmal nocturnal hemoglobinuria, deleting the GPI anchor on the gene *PIG-A* was lethal in mice [**33**].

GPI anchored proteins are hypothesized to associate with lipid raft domains (cholesterol-dependent nanodomains) [**64**]. These domains accommodate certain membrane proteins better than other areas of the plasma membrane [**4**]. The domains are thought to provide locations for signaling molecules to organize and activate other proteins in a signaling network. For example, a hypothesis from [**14**] is that clusters of GPIAPs induce the formation of cholesterol-dependent nanodomains. Activated Src family kinases (SFKs) lead to linkage of a transmembrane protein to the cytoskeleton.

In [**14**] experiments were performed on GPI anchored proteins to determine how these proteins associate with the cytoskeleton. In these experiments, single particle tracking was used to study the movements of single lipids and GPI anchored protein clusters tagged with 40 nm gold particles (Fig. 1(a)). Regions of transient confinement and transient anchorage of the particles were revealed (Fig. 1(b)). The distribution of transient anchorage release times was discovered to have a long tail. We developed a stochastic model of the system to explain the transient anchorage release times and the underlying biochemical reaction system.

4.1. Experimental data

Coordinate data were obtained from tracking the gold particles cross-linked with GPI anchored proteins. Single particles diffused on the cell membrane and transient anchorage events where particles exhibited no detectable displacement were observed. Experimental error in [**14**] was determined to be a distance of 25 nm. The criteria for a transient anchorage event was particle displacement was less than 25 nm for more than 132 ms. Figure 4.2 shows a histogram of release times from an anchored state.

4.2. Mathematical framework

In [**14**] the shape of the distribution of transient anchorage release times was used to show different cross-linking schemes generated similar results. Our goal is to formulate a model for the distribution consistent with observed data and to analyze the implications. To model the longer tail of the release time distribution, we assume the probability density function of release times has the form

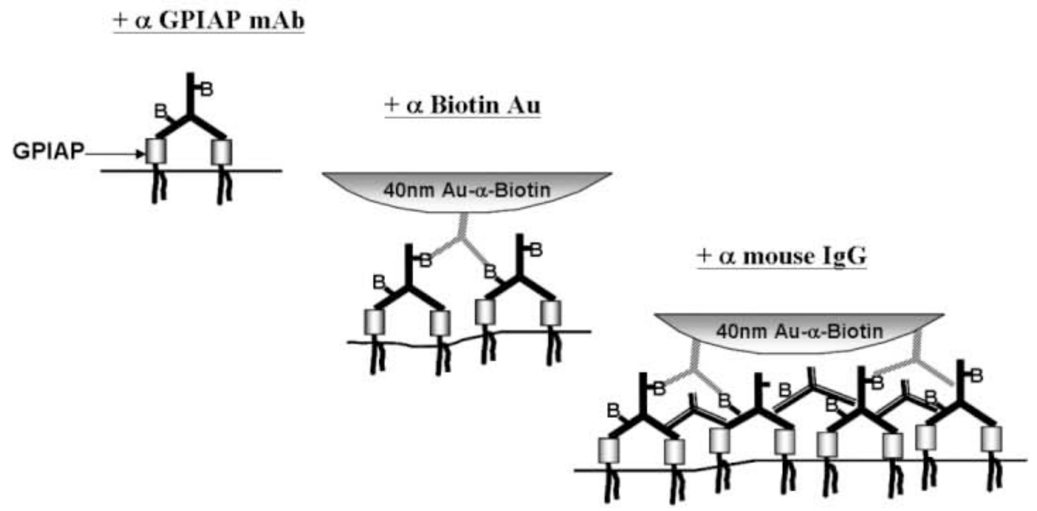
$$p(t) = ae^{c_1 t} + be^{c_2 t} \tag{4.1}$$

A histogram of the experimental data with a nonlinear least squares fit of the data is shown in Fig. 4.3. Single exponential and biexponential functions were used to fit the data. The single exponential function for the fit was

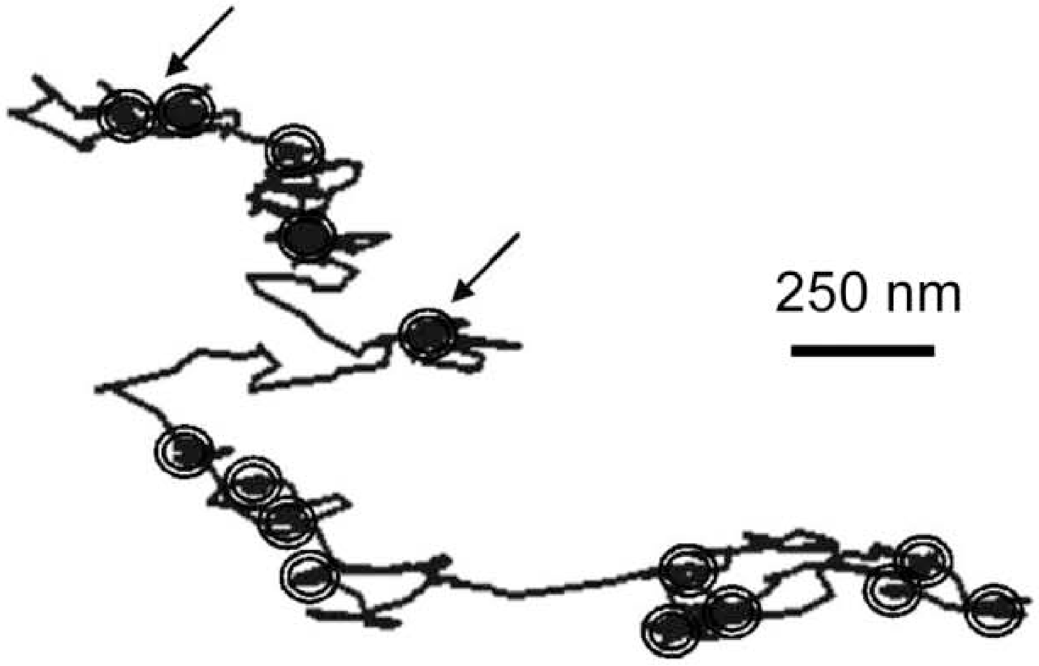
$$f(t) = \log\left(ae^{c_1 t}\right), \tag{4.2}$$

with parameter values $a = 4.641$ and $c_1 = -0.417$. The biexponential function of the form

$$f(t) = \log\left(ae^{c_1 t} + be^{c_2 t}\right) \tag{4.3}$$



(a) After incubating cells with biotinylated mouse primary antibodies recognizing specific GPI-APs (top), antibiotin gold particles are added on the cell membrane to form bonds with the primary antibodies (middle). Finally, tertiary polyclonal antibodies that bind to mouse IgG (immunoglobulin G) antibodies are added to further cross-link the GPIAPs (bottom)



(b) During single particle tracking, short periods with zero displacement were observed (arrows indicate representative transient anchorage events)

FIGURE 4.1. Maximal cross-linking scheme that produces transient anchorage. Reprinted with permission from the authors: Y. Chen, W. R. Thelin, B. Yang, S. L. Milgram, and K. Jacobson, and published by Rockefeller University Press (RUP).

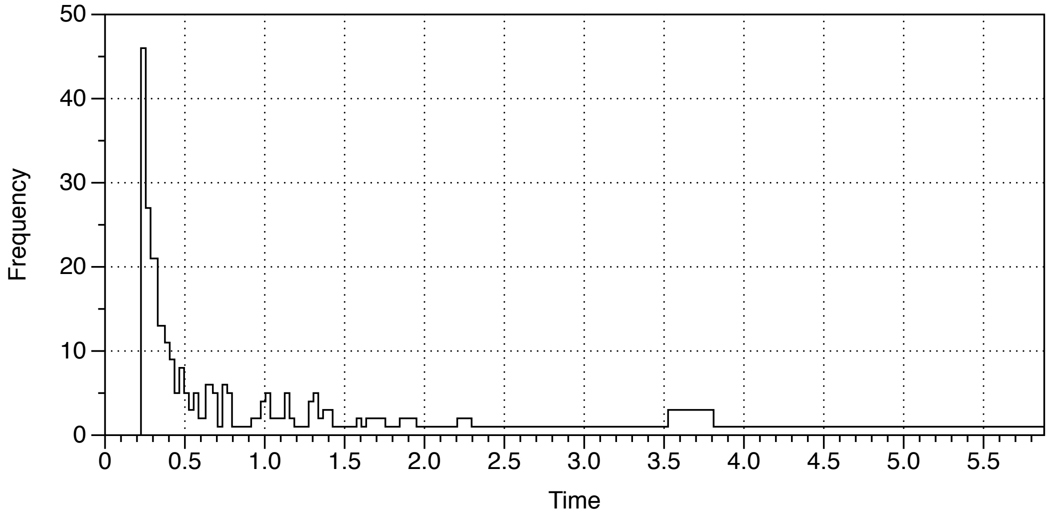


FIGURE 4.2. The histogram of release times from transient anchorage events taken from experimental data [**14**].

yielded the parameter values $a = 2.340$, $b = 295.795$, $c_1 = -0.194$, and $c_2 = -9.009$. The biexponential function provides a better fit of the experimental data.

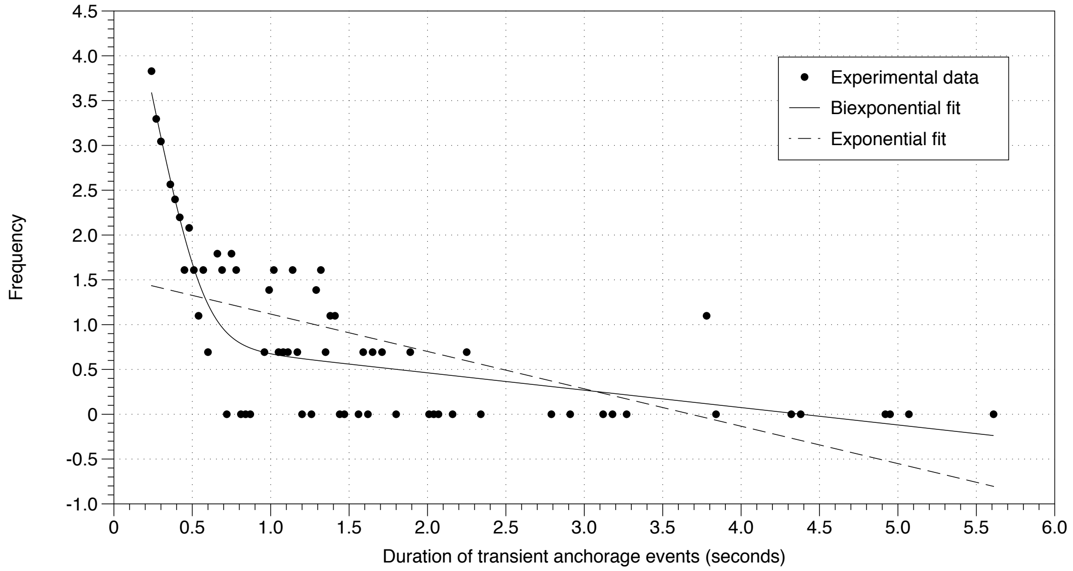


FIGURE 4.3. The histogram of transient anchorage events exhibits a long tail. A biexponential function provides a better fit for the data than a single exponential. The scale for the y axis is logarithmic.

A stochastic model for release times consistent with the proposed exponential form in Eq. (4.1) is given by

$$
\begin{aligned}
A \quad &\underset{b_1}{\overset{k_{on}}{\rightleftharpoons}} \quad B \\
B \quad &\underset{b_2}{\overset{f_1}{\rightleftharpoons}} \quad C,
\end{aligned}
\tag{4.4}
$$

where A is a state of free diffusion of the cross-linked GPIAP cluster on the cell membrane. B and C are states of transient anchorage. The extra state C allows for a time delay before exiting transient anchorage.

The parameters k_{on} and b_1 can be estimated from single particle trajectory data using a hidden Markov model [**21**] (HMM). An HMM is a stochastic dynamical system model with partially observed states. Unlike state B, state C cannot be observed from time series. Therefore, we are unable estimate the transition probabilities f_1 and b_2, and consider a basic stochastic model of transient anchorage consisting of two states. The first state is free diffusion, and the second is diffusion constrained by a tether. A continuous model of diffusion can be described by the Wiener process,

$$
dY^{(1)}(t) = \sigma dB(t), \tag{4.5}
$$

where $B(t)$ is standard Brownian motion with variance σ. A continuous model of transient anchorage is modeled with diffusion constrained by a tether is given by,

$$
dY^{(2)}(t) = -\kappa \left(Y^{(2)}(t) - Y(\tau) \right) dt + \sigma dB(t), \tag{4.6}
$$

where τ is the time of the last anchorage. The discrete version of this process is Gaussian, independent random variables with variance $\frac{\sigma^2}{2\kappa}(1 - e^{-2\kappa \Delta t})$. An approximate discrete version of the two state model is given by,

$$
Y_n = I_{\{Z_n=1\}} Y_{n-1} + I_{\{Z_n=2\}} Y_\tau + \sigma(Z_n) X_n, \tag{4.7}
$$

where τ is the index where a switch occurs, and $I_{Z_n=i}$ is one if $Z_n = 1$ is true and zero otherwise. The random variable X_n is a sequence of zero mean, unit variance normal random variable. The variance $\sigma(Z_n)$ depends on the state of binding Z_n. To remove the dependence on Y_τ, the time series is differenced,

$$Y_n = \sigma(Z_n)\left(X_n - I_{\{Z_n=2\}}X_{n-1}\right). \tag{4.8}$$

The switching state space model can be written as

$$Y_n = A_{Z_n}X_n \tag{4.9}$$

and

$$\begin{pmatrix} X_n \\ X_{n-1} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} X_{n-1} \\ X_{n-2} \end{pmatrix} + \begin{pmatrix} W_n \\ 0 \end{pmatrix}, \tag{4.10}$$

or in matrix notation,

$$\vec{X}_n = \Phi \vec{X}_{n-1} + \epsilon_n. \tag{4.11}$$

Z_n is a discrete space Markov chain with probability transition matrix K. W_n are independent, identically distributed random variables with mean zero. The vectors A_i are given by

$$A_1 = (\sigma(1), 0) \qquad A_2 = (\sigma(2), -\sigma(2)). \tag{4.12}$$

4.2.1. Filtering. We use filtering to find the probability of being in a certain state. Given an observable state V_n, a filter will allow us to estimate the probability density function of the unobserved state U_n, given previous observations. Because X_n is a Markov process, we know the density $f(u_n|u_{n-1})$. We also know the distribution $f(v_n|u_n)$. In our model (4.9), the conditional probability $f(\vec{x}_n|\vec{x}_{n-1})$ is normal with mean $(0, x_{n-1})$ and zero covariance matrix,

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{4.13}$$

The probability $f(\vec{y_n}|\vec{x_n}) = A_{Z_n} X_n$.

For a general hidden Markov model, the goal of the filter is to find $f(u_n|v_1, \ldots, v_n)$, denoted $f(u_n|v_{1:n})$. The mean of this density gives an estimate for the state the particle is in at t_n. The filter involves a prediction step and an update. The prediction step is

$$f(u_n|v_{1:n-1}) = \int f(u_n|u_{n-1}) f(u_{n-1}|v_{1:n-1}) du_{n-1}. \tag{4.14}$$

The first term in the integral is known from the Markov property. The second term is the filter from the previous time step. The update step is Bayes' rule,

$$f(u_n|v_{1:n}) = \frac{f(v_n|u_n) f(u_n|v_{1:n-1})}{\int f(v_n|u_n) f(u_n|v_{1:n-1}) du_n}. \tag{4.15}$$

The integrals in the prediction and update steps may not have closed forms. For the case where U_n and V_n are normal random variables with linear dependencies, the integrals have closed forms and the filter is called a Kalman filter. The Kalman filter can be described by a mean vector and covariance matrix. A brief summary is provided in Appendix B. For our switching model, a Kalman filter must be modified to accommodate the dependence of the switching state space model on the Markov chain Z_n. The filter is a mixture of Gaussian distributions and depends on the entire path up to the current time. Modifications currently being developed to address this problem are discussed in Chapter 5.

4.2.2. Parameter estimation. We would like to infer parameters from the distributions of the hidden processes Z_n and X_n, given the observed random variables Y_n. Specifically, we want estimates for the transition matrix K and values for σ. Maximum likelihood can be used to obtain estimates for parameters of the HMM, denoted by Θ. The likelihood function is computed from the joint probability density of observed trajectories and has the form,

$$
\begin{aligned}
L(\Theta) = f(Y_1, \ldots, Y_n; \Theta) &= \prod_{i=1}^{n} f(Y_i | Y_1, \ldots, Y_{i-1}) \\
&= \prod_{i=1}^{n} \int f(Y_i | x_i) f(x_i | Y_{1:i-1}) dx_i. \qquad (4.16)
\end{aligned}
$$

This expression is calculated in the second step of the filter (Eq. (4.15)). By calculating the filter for different values of Θ, the likelihood function can be numerically maximized. This form of the likelihood is called the error prediction decomposition approach. Discussion on the use of the Kalman filter in maximum likelihood calculations is found in [**8, 27**].

4.3. Algorithm for simulating transient anchorage

We formulated an algorithm to simulate transient anchorage for testing the time series analysis. Simulated time series data can be compared to likelihood estimates to determine confidence intervals for parameters. Diffusion in two dimensions with diffusion coefficients $\mathbf{D} = (D_1, D_2)$ can be modeled by the system of stochastic differential equations (SDEs),

$$\mathbf{X}_t = (2\mathbf{D})^{1/2} \, d\mathbf{W}_t, \qquad (4.17)$$

where $\mathbf{W}_t$ is a Wiener process. A Wiener process can be thought of as the limiting process of a random walk as the step size tends to zero. Each entry in the vector $d\mathbf{W}_t$ is an independent, identically distributed normal random variable with zero mean and variance Δt. Transient anchorage is simulated as diffusion with a tight spring, modeled by an Ornstein-Uhlenbeck process

$$\mathbf{X}_t = \kappa \Delta t \, (\mathbf{X}_0 - \mathbf{X}_t) + (2\mathbf{D})^{1/2} \, d\mathbf{W}_t, \qquad (4.18)$$

where κ is a spring constant. The transition probability into a state of transient anchorage occurs with probability $k_{\text{on}}\Delta t$. In experiments, a transiently anchored particle undergoes no visible displacement. To model this effect, the spring constant κ is very large and the stochastic differential equations in Eq. (4.18) are stiff. Therefore, an implicit Euler scheme is used to update the system. The release times are generated by simulating the reaction system in Eq. (4.4) with a stochastic simulation algorithm [**23**] (details are located in Appendix C). Parameters b_1, f_1, and b_2 are chosen such that the distribution of release times (i.e. entering state A) has the same shape and tail as the experimental data.

Reasonable estimates for diffusion coefficients are obtained by analyzing time series data. The sample variance of step sizes in the x and y direction are calculated. The diffusion coefficient is estimated by the variance divided by 2 times the step size Δt.

4.4. Results

We begin with the two state model given by free diffusion and an diffusion constrained by a tether (Eqns. (4.5) and (4.6)) in one spatial dimension. Given specified parameters D, Δt, k_{on}, and b_1, we investigate the accuracy of the filter predictions as the spring constant κ varies. If the filter probability $p(X_n|Y_n) > 0.5$, we assume the particle is bound. Estimates for diffusion coefficients from experimental data range from $30,000$ to 80,000 nm^2/s. We use $D = 50,000$. k_{on} is set to 0.02 and b_1 is 0.04. The time step $\Delta t = 0.034$ was taken from the experimental data [**14**]. Figs. 4.4 and 4.5 show time series for two different values of the spring constant κ and the estimate for the switch based on the filter. Table 4.4 lists the percentage of correct guesses for the filter as a function of κ. The analysis indicates a large spring constant is necessary for accurate filter predictions as well as qualitatively matching experimental obersevations from [**14**].

Next, we illustrate the transient anchorage simulation method from section 4.3. A stochastic simulation algorithm to simulate Eqs. (4.4) was used to pre-compute the

κ	Percentage
10	36.6
20	48.7
30	74.9
40	84.3
60	90.7
80	91.3
100	93.0

TABLE 4.1. Percentage of correct guesses for the filter as a function of κ.

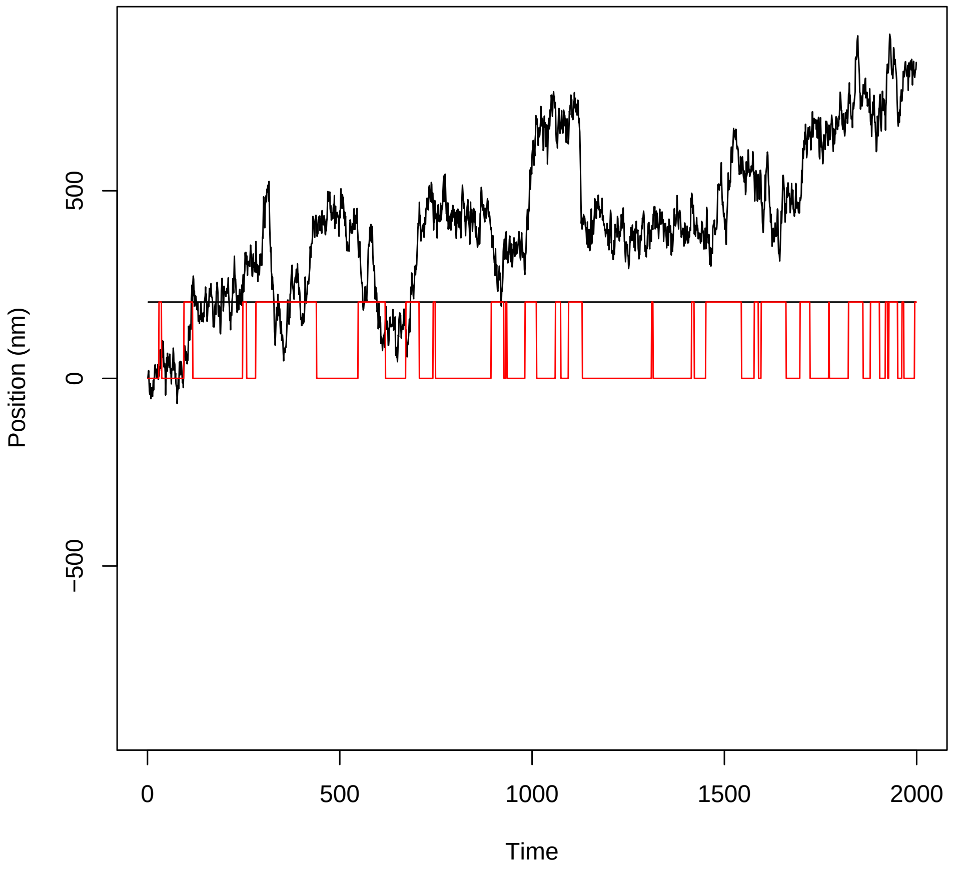


FIGURE 4.4. Simulated path trajectory for $\kappa = 10$ (black) and switch based on the filter (red).

release times. The initial conditions were $A = 0$, $B = 1$, and $C = 0$. When the particle entered state A, the time was saved, and the simulation re-initialized. The constants

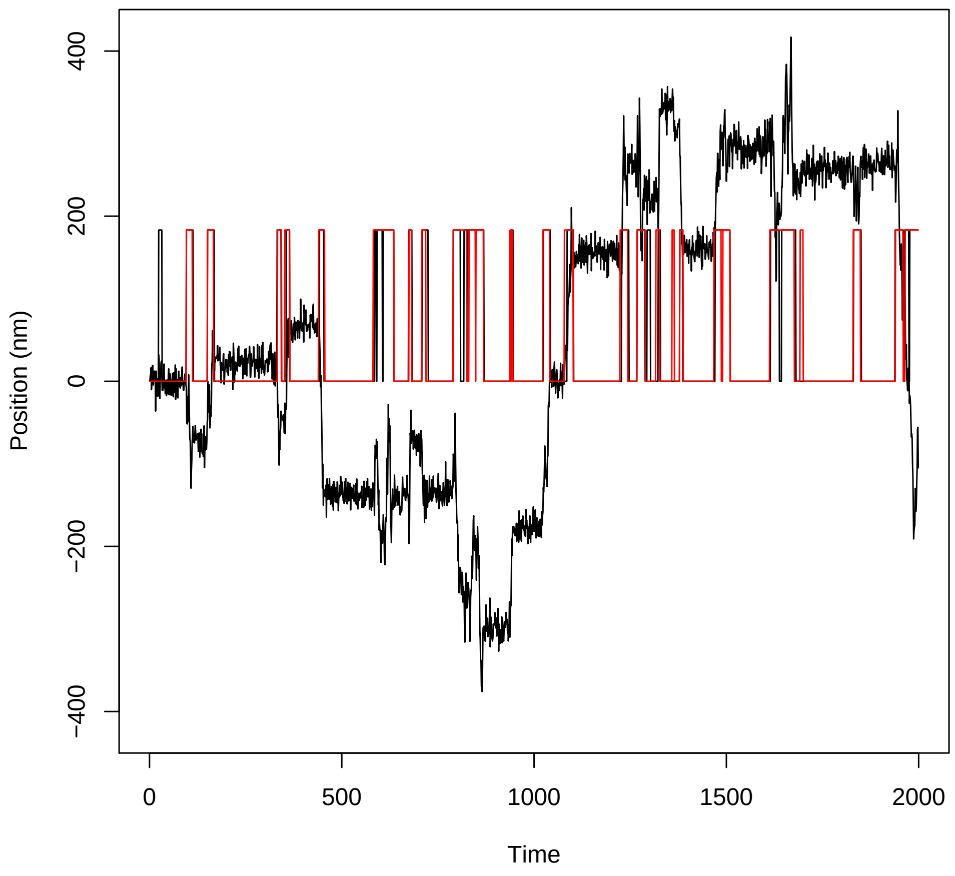


FIGURE 4.5. Simulated path trajectory for $\kappa = 70$ (black) and switch based on the filter (red).

$f_1 = 15$, $f_2 = 1$, and $b_1 = 1.2$ were chosen to capture the biexponential form and long tail of the experimental data from [**14**]. The constant k_{on} was arbitrarily chosen to be 0.01. The spring constant κ was set to 2200. The time step Δt was set to 0.001 to obtain numerical accuracy of the Euler method. However, trajectories can be sampled to match experimental ones. The diffusion coefficient D matched experimental data scaled for the smaller time step, and was set to 45 nm^2/s. A simulation of 1920 time steps with these values is shown in Figure 4.6.

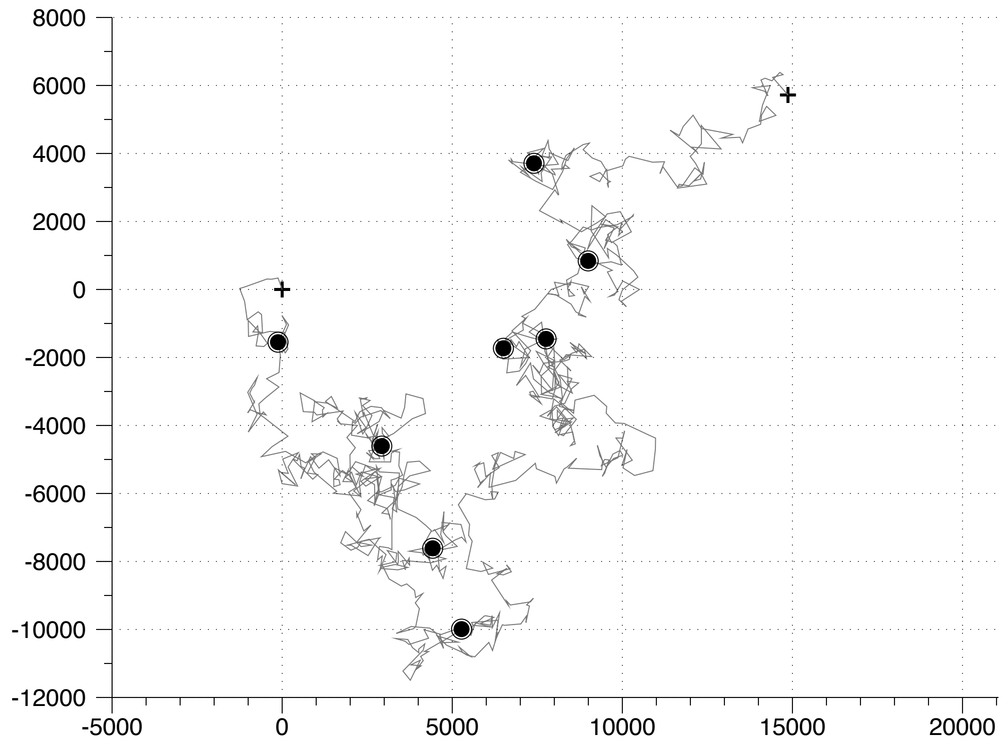


FIGURE 4.6. Simulation of transient anchorage. The black circles indicate locations of transient anchorage events. Plus marks indicate start and end locations. The x and y axis units are nm.

4.5. Conclusions

The cause of transient anchorage was hypothesized to be cross-linking of GPIAPs leading to the formation of cholesterol-dependent nanodomains on the inner and outer leaflets of the cell membrane [**14**]. The nanodomains include transmembrane proteins that facilitate signal transduction. An activated Src family kinase (SFK) can enter the domain, phosphorylate a transmembrane protein, leading to attachment to the actin cytoskeleton though adaptor proteins. Our model for the release times suggests that one linkage step is not enough to generate the long transient anchorage release time events. At least one other linker protein is needed to account for the long duration of some events.

Multi-state models have been used to explain other biological phenomena. Pauses in mechanical stepping of molecular motors are called dwell events. Multi-state models for the kinetic scheme of molecular motors have been shown to reproduce dwell-time distributions [**44**]. After the parameter estimation code is benchmarked (see Chapter 5), the method could be applied to other multi-state stochastic processes.

CHAPTER 5

Future Work

The algorithms for models of spatiotemporal signaling networks in Chapters 2 and 3 are limited to two dimensions. Three dimensional effects are important in many cell types, such as yeast cells. A three dimensional version of the algorithm in Chapter 2 would need a more computationally efficient method to invert the Jacobian matrix in Eq. (2.28). A multigrid method or preconditioner for the GMRES iteration are some options. Recent work suggests that higher order methods can be developed for embedded boundary methods through implicit function representation [**45**]. Taylor series approximations of boundary fluxes can be computed from moments. These ideas could be incorporated to increase the accuracy of the method in Chapter 2. Another possible improvement is to couple membrane diffusion to reactions on the boundary and interior using the scheme from [**3**].

There are several improvements for the moving boundary algorithm in Chapter 3. The first problem is the instability in the advection code. In [**16, 51**], a Godunov method is presented for embedded boundary methods that is between first and second order accurate. A redistribution scheme developed for shock tracking is needed to handle cells with arbitrarily small volumes, and an error term is added to make the solution conservative. The ideas from this method could be used to improve the accuracy of the advection-reaction-diffusion solver. Moment calculations described previously could also be used to increase the accuracy of the method.

A limitation of the method in Chapter 3 comes from the bicubic interpolation stencil. Adaptive mesh refinement on sections of the domain with thin protrusions would decrease computational resources necessary for model simulations.

The mathematical models presented in Chapters 2 and 3 lack realistic biological details due to the difficulty in obtaining data for parameter values, such as reaction rates and diffusion coefficients. The models in Chapters 2 and 3 were simulated with parameters that highlighted the influence of spatial terms on concentration profiles. In the future, the numerical methods presented here can be used in realistic spatial models to elucidate control mechanisms of cellular processes.

A stochastic model for the transient anchorage of membrane proteins was presented in Chapter 4. Software is currently being developed to detect the transitional probabilities, diffusion coefficients, and probability densities for Markov chain Z_n using sample trajectories obtained from the stochastic model. A challenge is computing conditional densities needed for the filter. Some calculations involve averaging over all possible paths of the hidden Markov chain. Given two states in the Markov chain, the number of paths to sum for each data point would be 2^n, where n is the number of points. The method currently being explored to simplify the calculation involves a normal approximation to the density of the hidden system given past observations. [**76**]. After the algorithm has been benchmarked, we will acquire parameter estimates from experimental trajectories.

APPENDIX A

Level Set Methods

Level set methods are algorithms for tracking the evolution of boundaries and interfaces over time. A boundary is represented as the zero level set of a higher dimensional signed distance function $\phi(x, y, t)$. This function gives the distance from grid point to the boundary, with a negative sign if it is enclosed by the boundary. Otherwise, the distance is assigned a positive sign (Fig. A.1). If a particle on the boundary is represented by the

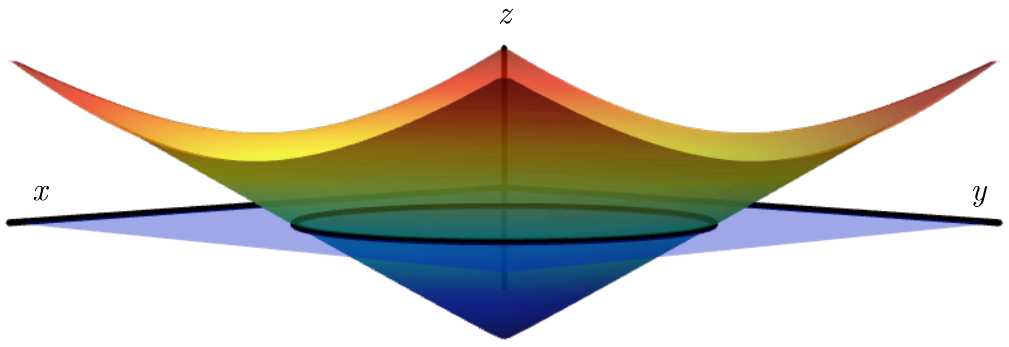


FIGURE A.1. An example of a signed distance function for a circle. The plane $z = 0$ is indicated in purple.

point $(x(t), y(t))$, then $\phi(x(t), y(t), t) = 0$. The chain rule gives us

$$\phi_t + \nabla\phi(x(t), y(t), t) \cdot (x'(t), y'(t)) = 0. \tag{A.1}$$

Speed in the normal direction is given by

$$F(x, y, t) = \vec{n} \cdot (x'(t), y'(t))\,, \tag{A.2}$$

where $\vec{n} = \nabla\phi|\nabla\phi|$. Then the evolution of ϕ is given by the level set equation originally presented in [**59**],

$$\phi_t + F|\nabla\phi| = 0, \tag{A.3}$$

subject to the initial condition $\phi(x, y, 0)$. Suppose we have a velocity field $\vec{U}$ that is a function of only position and time and passively advects the front. Then ϕ satisfies

$$\phi_t + F|\nabla\phi| + \vec{U} \cdot \nabla\phi = 0. \tag{A.4}$$

A.1. Operator discretization

In order to numerically solve the partial differential equation in Eq. (A.4), spatial and temporal operators must be discretized. The boundary that we wish to capture may not be differentiable and have sharp corners. Schemes developed for hyperbolic conservation laws to capture the evolution of the slope over time are used to obtain the weak solution to Eq. (A.4). The following schemes also converge to the unique viscosity solution. Details and further discussion of the methods presented here can be found in [**72**].

Let us consider a scheme first order in space and time with one spatial dimension. If the speed function $F = 1$, then the update for ϕ at the i^{th} spatial point and $n+1^{\text{th}}$ time step is

$$\phi_i^{n+1} = \phi_i^n - \Delta t \left(\max(D_i^{-x}, 0)^2 + \min(D_i^{+x}, 0^2)\right)^{1/2}. \tag{A.5}$$

The operators $D_i^{\pm x}$ are discretizations of the $\partial\phi/\partial x$ given by

$$D^{+x}\phi(x,t) \equiv \frac{\phi(x+\Delta x, t) - \phi(x,t)}{\Delta x} \tag{A.6}$$

$$D^{-x}\phi(x,t) \equiv \frac{\phi(x,t) - \phi(x-\Delta x, t)}{\Delta x}. \tag{A.7}$$

The gradient approximations in the above equations is an example of upwinding. Consider the one dimensional advection equation with constant positive speed c. The domain of dependence of the point (x,t) is $(x-ct)$ [**40**]. Therefore, we must use computational points behind the i^{th} point to calculate the gradient. If c is negative, we use points in a direction ahead of the i^{th} point. In two-dimensions with a non-constant speed function F, an upwinding update is

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} - \Delta t \left(\max(F_{ij},0)\nabla^{+} + \min(F_{ij},0)\nabla^{-}\right), \tag{A.8}$$

where

$$\begin{aligned} \nabla^{+} &= \left(\max(D_{ij}^{-x},0)^2 + \min(D_{ij}^{+x},0)^2 + \max(D_{ij}^{-y},0)^2 + \min(D_{ij}^{+y},0)^2\right)^{1/2} \\ \nabla^{-} &= \left(\max(D_{ij}^{+x},0)^2 + \min(D_{ij}^{-x},0)^2 + \max(D_{ij}^{+y},0)^2 + \min(D_{ij}^{-y},0)^2\right)^{1/2}. \end{aligned} \tag{A.9}$$

The stencil for Eq. (A.4) is given by

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} - \Delta t \begin{pmatrix} \max(F_{ij},0)\nabla^{+} & + & \min(F_{ij},0)\nabla^{-} & + \\ \max(u_{ij}^{n},0)D_{ij}^{-x} & + & \min(u_{ij}^{n},0)D_{ij}^{+x} & + \\ \max(v_{ij}^{n},0)D_{ij}^{-y} & + & \min(v_{ij}^{n},0)D_{ij}^{+y} & \end{pmatrix}. \tag{A.10}$$

To obtain a second order spatial update, we use a higher order approximation of the derivative given by an ENO scheme from [**26**]. Then we have the following updates for ∇^{+} and ∇^{-}:

$$\nabla^{+} = \left(\max(A,0)^2 + \min(B,0)^2 + \max(C,0)^2 + \min(D,0)^2\right)^{1/2} \tag{A.11}$$

$$\nabla^{-} = \left(\max(B,0)^2 + \min(A,0)^2 + \max(D,0)^2 + \min(C,0)^2\right)^{1/2}, \tag{A.12}$$

where

$$\begin{aligned} A &= D_{ij}^{-x} + \tfrac{\Delta x}{2} m(D_{ij}^{-x-x}, D_{ij}^{+x-x}) \\ B &= D_{ij}^{+x} - \tfrac{\Delta x}{2} m(D_{ij}^{+x+x}, D_{ij}^{+x-x}) \\ C &= D_{ij}^{-y} + \tfrac{\Delta y}{2} m(D_{ij}^{-y-y}, D_{ij}^{+y-y}) \\ D &= D_{ij}^{+y} - \tfrac{\Delta y}{2} m(D_{ij}^{+y+y}, D_{ij}^{+y-y}), \end{aligned} \tag{A.13}$$

and m is the switch function

$$m = \begin{cases} x & \text{if } |x| \le |y| \text{ and } \quad xy \ge 0 \\ y & \text{if } |x| > |y| \text{ and } \quad xy \ge 0 \\ 0 & \qquad\qquad\qquad\quad xy < 0. \end{cases} \tag{A.14}$$

$D^{+x+x}\phi(x,t)$, $D^{+x-x}\phi(x,t)$, and $D^{-x-x}\phi(x,t)$ are discretizations for $\partial^2\phi/\partial^2 x$ given by

$$D^{+x+x}\phi(x,t) \equiv \frac{\phi(x+2\Delta x,t) - 2\phi(x+\Delta x,t) + \phi(x,t)}{\Delta x^2} \tag{A.15}$$

$$D^{-x-x}\phi(x,t) \equiv \frac{\phi(x-2\Delta x,t) - 2\phi(x-\Delta x,t) + \phi(x,t)}{\Delta x^2} \tag{A.16}$$

$$D^{+x-x}\phi(x,t) \equiv \frac{\phi(x+\Delta x,t) - 2\phi(x,t) + \phi(x-\Delta x,t)}{\Delta x^2}. \tag{A.17}$$

To make (A.10) second order in space, we replace D^{-x}_{ij} with A, D^{+x}_{ij} with B, etc.

To make the temporal update second order, we use a second order Runge-Kutta method (Heun's method). For an ordinary differential equation $y' = f(t, y(t))$ with initial condition $y(t_0) = y_0$, the update is given by

$$\begin{aligned} y^{*(n+1)} &= y^n + \Delta t f(t^n, y^n) \\ y^{n+1} &= y^n + \frac{\Delta t}{2}\left(f(t^n, y^n) + f(t^{n+1}, y^{*(n+1)})\right), \end{aligned} \tag{A.18}$$

where n represented the $y(t)$ at time t^n (or $n\Delta t$) and $y^{*(n+1)}$ is an intermediate time value. In our level set equation (A.4), the first step of the update is given by

$$\phi^{(n+1)*}_{ij} = \phi^n_{ij} - \Delta t \begin{pmatrix} \max(F^n_{ij},0)\nabla^{n+} & + & \min(F^n_{ij},0)\nabla^{n-} & + \\ \max(u^n_{ij},0)A^n & + & \min(u^n_{ij},0)B^n & + \\ \max(v^n_{ij},0)C^n & + & \min(v^n_{ij},0)D^n & \end{pmatrix}, \tag{A.19}$$

to obtain the intermediate time value. Then the second order temporal update is

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} - \Delta t/2 \begin{pmatrix} \max(F_{ij}^n,0)\nabla^{n+} & + & \min(F_{ij}^n,0)\nabla^{n-} & + \\ \max(u_{ij}^n,0)A^n & + & \min(u_{ij}^n,0)B^n & + \\ \max(v_{ij}^n,0)C^n & + & \min(v_{ij}^n,0)D^n & + \\ \max(F_{ij}^{n*},0)\nabla^{n*+} & + & \min(F_{ij}^{n*},0)\nabla^{n*-} & + \\ \max(u_{ij}^{n*},0)A^{n*} & + & \min(u_{ij}^{n*},0)B^{n*} & + \\ \max(v_{ij}^{n*},0)C^{n*} & + & \min(v_{ij}^{n*},0)D^{n*} & \end{pmatrix}, \tag{A.20}$$

where the $n*$ superscript indicates the quantities F_{ij}, ∇^+, ∇^-, A, B, C, and D are updated using the intermediate time value $\phi^{(n+1)*}$.

To ensure numerical stability of these methods, we require that the boundary cross no more than one grid cell during each time step. i.e.

$$\max_{\Omega}(F,u,v)\,\Delta t \le \Delta x. \tag{A.21}$$

A.2. Initialization

The partial differential equation Eq. (A.4) is an initial value problem. For certain domains, initialization of the signed distance function can be easily calculated. For example, the signed distance function $\phi(x,y,0)$ from a circle of radius r centered at (x_0,y_0) is computed as follows. The distance in absolute coordinates from the point (i,j) to the radius $(x_0/\Delta x, y_0/\Delta y)$ is subtracted from the scaled radius $r/\sqrt{\Delta x^2+\Delta y^2}$. The values inside of the circle are assigned a negative value.

For a general domain, fast marching methods can be used to initialize $\phi(x,y,t)$ [**46**]. These are methods to numerically solve the Eikonal equation

$$|\nabla T|F = 1, \tag{A.22}$$

where T is the arrival time of a boundary as it propagates with speed F. If a front moves with speed $F = 1$, then the arrival time gives us the signed distance to initialize the initial boundary problem. We provide a brief summary of the method. For details, see [**72**].

The first step in a fast marching method is to identify the points that are closest to the front (i.e. the shortest arrival time), compute the distance, then march outwards from smallest to largest value in a downwind direction. Values located at computational grid points are put into three categories: `Accepted`, `Close`, and `Far`. In our implementation, the initial distance values are found by iterating around the boundary points and finding the closest distance from each nearest grid point to the point that lies on the linear segment connecting two boundary points (x_1, y_1) and (x_2, y_2),

$$\mathbf{r}(t) = (1-t)\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + t\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}. \tag{A.23}$$

These initial values are moved into the `Accepted` category. Marching outwards, the next set of points updated are those that lie one grid cell away, and are marked as `Close`. Other points are marked as `Far`. The `Close` values are found by solving the discretized equation Eq. (A.22)

$$\begin{pmatrix} \max(D^{-x}_{i,j}T, -D^{+x}_{i,j}T, 0)^2 \quad + \\ \max(D^{-y}_{i,j}T, -D^{+y}_{i,j}T, 0)^2 \end{pmatrix}^{1/2} = 1, \tag{A.24}$$

which is a quadratic equation. The discretization in the above equation is an upwinding discretization from [**65**]. The value labeled `Trial` is the smallest of these values. The neighbors of this point are put into the `Close` category. The `Trial` value is then accepted and the algorithm marches onward. By using a heap sort with backpointers, the method is $\mathcal{O}(N \log N)$.

A.3. Boundary conditions

The boundaries of computational grid should be far enough away from the boundary given by the zero level set of $\phi(x, y, t)$. We use mirroring boundary conditions. Periodic boundary conditions can also be used.

A.4. Convergence test

The initial front is a circle of radius 0.2 centered at the point $(0.5, 0.5)$. The computational grid is a box with a lower point at the origin and an upper point $(1, 1)$. In our first example, the front is advected with the constant velocity field $\vec{U} = (0.1, 0.1)$. The exact solution is a circle with radius 0.2 centered at the point $(0.5+0.1t, 0.5+0.1t)$. The advection update is the second order method given in Eqs. (A.19) and (A.20). For this example, the speed function F is zero. The signed distance function is initialized by finding the minimum distance from a piecewise linear representation of the boundary to each grid point. The error is computed as the norm of the distance from the zero level set of $\phi(x, y, t)$ to the exact solution. Figure A.2 shows second order convergence in averaged L^2 norm. The error in other norms appears similar, and has the same scaling.

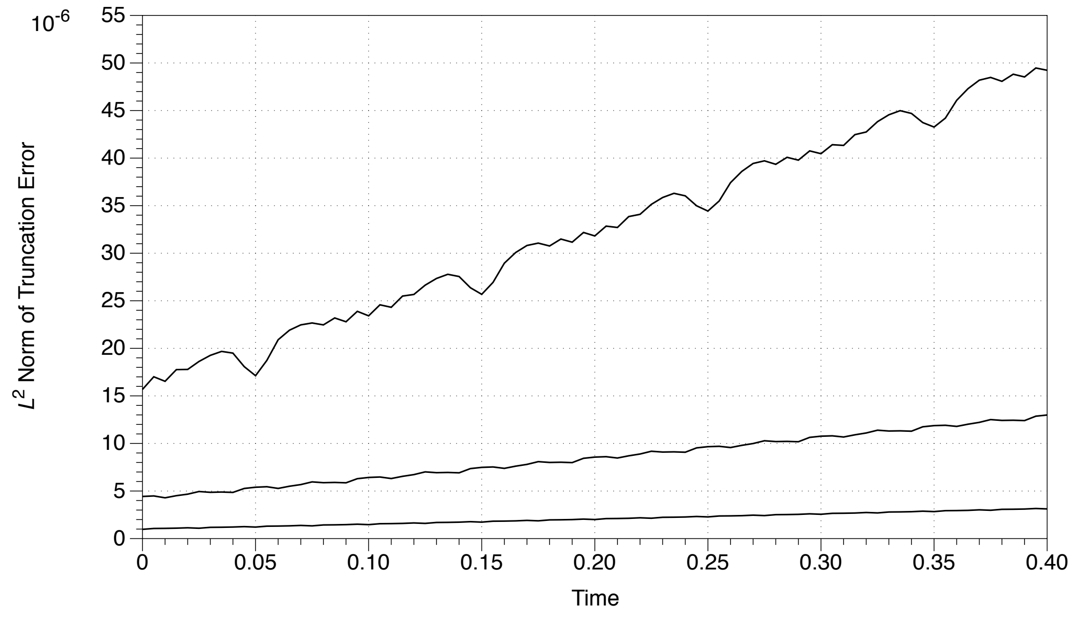


FIGURE A.2. The truncation error for a circle propagating with constant velocity. The top plot is a grid size of 100×100. The middle plot is 200×200, and the bottom is a 400×400 grid. The time step Δt was set to $1/4N$, where N is the grid size.

APPENDIX B

The Kalman Filter

The Kalman filter estimates of the current state of a stochastic dynamical system X_n given a set of noisy observations Y_n. The stochastic model is assumed to be a linear system model. The observations are assumed to depend linearly on the state of the system. The system equation is given by

$$X_n = \Phi X_{n-1} + \epsilon_n, \tag{B.1}$$

where $\epsilon_n \sim \mathcal{N}(0, Q)$. The observation equation is

$$Y_n = A_n X_{n-1} + \xi_n, \tag{B.2}$$

where $\xi_n \sim \mathcal{N}(0, R)$. In our model, we assume no observation error and set $R = 0$.

Calculations in the filter rely on theory from multivariate statistical analysis. A theorem gives the distribution of a portion of a Gaussian random vector conditioned on the other portion. Specifically, given a normal random vector A,

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right), \tag{B.3}$$

the distribution of A_1 given $A_2 = a$ is

$$\mathcal{N}\left(\mu_1 + \sigma_{12}\Sigma_{22}^{-1}(a - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right). \tag{B.4}$$

The notation $\mathcal{N}(\mu, \Sigma)$ indicates a multivariate normal distribution with mean vector μ and covariance matrix Σ.

The goal of the Kalman filter is to find the distribution of $X_n|Y_1,\ldots,Y_n$, denoted $X_{n|n}$, given a new observation Y_n and the distribution $X_{n-1|n-1}$. The conditional mean and covariance of $X_{n-1|n-1}$ are denoted by $\hat{X}_{n-1}$ and P_{n-1}^{n-1}. From Eq. (B.1), we have

$$X_{n|n-1} = \mathcal{N}\left(\Phi\hat{X}_{n-1}, \Phi P_{n-1}^{n-1}\Phi^t + Q\right). \tag{B.5}$$

For simplicity, we write the covariance matrix $P_n^{n-1} = \Phi P_{n-1}^{n-1}\Phi^t + Q$. From the theorem and conditioning on the first $n-1$ observations $Y_1,\ldots,Y_{n-1}$, we have

$$\begin{pmatrix} Y_{n|n-1} \\ X_{n|n-1} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} A_n\Phi\hat{X}_{n-1} \\ \Phi\hat{X}_{n-1} \end{pmatrix}, \begin{pmatrix} R + A_nP_n^{n-1}A_n^t & A_nP_n^{n-1} \\ P_n^{n-1}A_n^t & P_n^{n-1} \end{pmatrix}\right). \tag{B.6}$$

The theorem also gives us the mean of $X_{n|n}$,

$$\hat{X}_n = \Phi\hat{X}_{n-1} + P_n^{n-1}A_n^t\left(R + A_nP_n^{n-1}A_n^t\right)^{-1}\left(Y_n - A_n\Phi\hat{X}_{n-1}\right) \tag{B.7}$$

and covariance

$$P_n = P_n^{n-1} - P_n^{n-1}A_n^t\left(R + A_nP_n^{n-1}A_n^t\right)^{-1}A_nP_n^{n-1}. \tag{B.8}$$

Given a new observation at time n and the filter at $n-1$, we have the necessary recursion to obtain the value of the filter at time n. The conditional probability $Y_n|Y_{1,\ldots,n-1}$ is used in the error-prediction decomposition approach to calculating the maximum likelihood function. The distribution is given by the first entry of the vector in Eq. (B.6),

$$Y_{n|n-1} \sim \mathcal{N}\left(A_n\Phi\hat{X}_{n-1}, R + A_nP_n^{n-1}A_n^t\right). \tag{B.9}$$

Further discussion on the Kalman filter can be found in [**27, 77**].

APPENDIX C

Exact Stochastic Simulation Algorithm

For some cellular reaction systems, stochastic effects are important. For example, in gene transcription, there are usually two copies of each gene and small amounts of messenger RNA. In this appendix, we summarize Gillespie's method from [**23**] to simulate the chemical master equation. The master equation gives the probability that at a given time there will be certain number of molecules for a chemical species. The method is exact in the sense that trajectories generated by the methods that are statistically equivalent to those that result from solving the master equation.

C.1. Background

The reaction probability density function $P(\tau, \mu)$ is defined to be the probability that given the current state of a system with n reactants X_i at time t, the next reaction will occur in the time interval $(t+\tau, t+\tau+d\tau)$, and will be the reaction R_μ, given M reactions. Reaction rates are given by a_μ's. The probability that the μ^{th} reaction will occur in a time interval $d\tau$ given the current state of the system is $a_\mu d\tau$. $P(\tau, \mu)$ is defined to be the probability that no reaction will occur in the interval $(t, t+\tau)$, denoted $P_0(\tau)$, times the probability that the reaction R_μ will occur in the time interval $(t+\tau, t+\tau+d\tau)$,

$$P(\tau, \mu)d\tau = P_0(\tau)a_\mu d\tau. \tag{C.1}$$

The probability that no reaction will occur in time $d\tau$ is $1 - \sum_{i=1}^{M} a_i d\tau$. Then we have

$$P_0(\tau + d\tau) = P_0(\tau)\left(1 - \sum_{i=1}^{M} a_i d\tau\right), \tag{C.2}$$

or

$$P_0(\tau) = \exp\left(-\sum_{i=1}^{M} a_i\tau\right). \tag{C.3}$$

The reaction probability density function is given by

$$P(\tau,\mu) = \begin{cases} a_\mu \exp(-a_0\tau) & \text{for } 0 \le \tau < \infty \text{ and } \mu = 1,\dots,M \\ 0 & \text{otherwise,} \end{cases} \tag{C.4}$$

where $a_0 = \sum_{i=0}^{M} a_i$.

C.2. The Algorithm

(1) Initialize the system.
(2) Loop over time until $t = t_{\text{end}}$
    (a) Calculate the propensity functions, a_i.
    (b) Generate two random numbers r_1 and r_2.
    (c) Set $\tau = 1/a_0 \log(1/r_1)$.
    (d) Set μ to be the integer that satisfies $\sum_{i=1}^{\mu-1} a_i < r_2 a_0 \le \sum_{i=1}^{\mu} a_i$.
    (e) Set $t = t + \tau$ and the state $\vec{X} = \vec{X} + \Delta_\mu$, where Δ_μ is the μ^{th} $N \times 1$ column vector from the stoichiometric matrix Δ.
    (f) Return to step (a).

BIBLIOGRAPHY

1. D. Adalsteinsson, *DTSource*, `http://www.visualdatatools.com/DTSource.html`.

2. D. Adalsteinsson and J. A. Sethian, *The fast construction of extension velocities in level set methods*, J. Comput. Phys. **148** (1999), no. 1, 2–22.

3. D. Adalsteinsson and J. A. Sethian, *Transport and diffusion of material quantities on propagating interfaces via level set methods*, J. Comput. Phys. **185** (2003), no. 1, 271–288.

4. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular biology of the cell*, 4th ed., Garland Science, New York, 2002.

5. B. J. Bacskai, B. Hochner, M. Mahaut-Smith, S. R. Adams, B. K. Kaang, E. R. Kandel, and R. Y. Tsien, *Spatially resolved dynamics of cAMP and protein kinase A subunits in Aplysia sensory neurons*, Science **260** (1993), no. 5105, 222–226.

6. K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical solution of initial-value problems in differential-algebraic equations*, vol. 14, Society for Industrial and Applied Mathematics, Philadelphia, 1996.

7. N. F Britton, *Essential mathematical biology*, Springer, London, 2003.

8. P. J. Brockwell and R. A. Davis, *Time series: theory and methods*, 2nd ed., Springer-Verlag, New York, 1991.

9. G. C. Brown and B. N. Kholodenko, *Spatial gradients of cellular phospho-proteins*, FEBS Lett. **457** (1999), no. 3, 452–454.

10. P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, *Using Krylov methods in the solution of large-scale differential-algebraic systems*, SIAM J. Sci. Comput. **15** (1994), no. 6, 1467–1488.

11. B. Bunow, J. P. Kernevez, G. Joly, and D. Thomas, *Pattern formation by reaction-diffusion instabilities: application to morphogenesis in Drosophila*, J. Theor. Biol. **84** (1980), no. 4, 629–649.

12. K. Burridge and K. Wennerberg, *Rho and Rac take center stage*, Cell **116** (2004), no. 2, 167–179.

13. J. M. Cancela, F. Van Coppenolle, A. Galione, A. V. Tepikin, and O. H. Petersen, *Transformation of local Ca2+ spikes to global Ca2+ transients: the combinatorial roles of multiple Ca2+ releasing messengers*, EMBO J. **21** (2002), no. 5, 909–919.

14. Y. Chen, W. R. Thelin, B. Yang, S. L. Milgram, and K. Jacobson, *Transient anchorage of cross-linked glycosyl-phosphatidylinositol-anchored proteins depends on cholesterol, Src family kinases, caveolin, and phosphoinositides*, J. Cell Biol. **175** (2006), no. 1, 169–178.

15. P. Colella, D. Graves, T. Ligocki, D. Trebotich, and B. V. Straalen, *Embedded boundary algorithms and software for partial differential equations*, J. Phys. Conf. Ser. **125** (2008), 012084 (8pp).

16. P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, *A Cartesian grid embedded boundary method for hyperbolic conservation laws*, J. Comput. Phys. **211** (2006), no. 1, 347–366.

17. A. Csikász-Nagy, D. Battogtokh, K. C. Chen, B. Novák, and J. J. Tyson, *Analysis of a generic model of eukaryotic cell-cycle regulation*, Biophys. J. **90** (2006), no. 12, 4361–4379.

18. A. T. Dawes and L. Edelstein-Keshet, *Phosphoinositides and Rho proteins spatially regulate actin polymerization to initiate and maintain directed movement in a one-dimensional model of a motile cell*, Biophys. J. **92** (2007), no. 3, 744–768.

19. M. Dobrzynski, J. V. Rodriguez, J. A. Kaandorp, and J. G. Blom, *Computational methods for diffusion-influenced biochemical reactions*, Bioinformatics **23** (2007), no. 15, 1969–1977.

20. H. G. Dohlman, *G proteins and pheromone signaling*, Annu. Rev. Physiol. **64** (2002), 129–152.

21. R. J. Elliott, L. Aggoun, and J. B. Moore, *Hidden Markov models: estimation and control*, vol. 29, Springer-Verlag, New York, 1995.

22. A Gierer and H. Meinhardt, *Theory of biological pattern formation*, Kybernetik **12** (1972), no. 1, 30–39.

23. D.T. Gillespie, *Exact stochastic simulation of coupled chemical-reactions*, J. Phys. Chem. **81** (1977), no. 25, 2340–2361.

24. J. A. Glazier and F. Graner, *Simulation of the differential adhesion driven rearrangement of biological cells*, Phys. Rev. E. **47** (1993), no. 3, 2128–2154.

25. A. B. Goryachev and A. V. Pokhilko, *Dynamics of Cdc42 network embodies a Turing-type mechanism of yeast cell polarity*, FEBS Lett. **582** (2008), no. 10, 1437–1443.

26. A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes, 111*, J. Comput. Phys. **71** (1987), no. 2, 231–303.

27. A. C. Harvey, *Forecasting, structural time series models, and the Kalman filter*, Cambridge University Press, Cambridge, 1990.

28. A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, and C.S. Woodward, *SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers*, ACM T. Math. Software **31** (2005), no. 3, 363–396.

29. S.A. Isaacson and C.S. Peskin, *Incorporating diffusion in complex geometries into stochastic chemical kinetics simulations*, SIAM J. Sci. Comput. **28** (2006), no. 1, 47–74.

30. C. Jimenez, R. A. Portela, M. Mellado, J. M. Rodriguez-Frade, J. Collard, A. Serrano, C. Martinez-A, J. Avila, and A. C. Carrera, *Role of the PI3K regulatory subunit in the control of actin organization and cell migration*, J. Cell Biol. **151** (2000), no. 2, 249–262.

31. M. Jin, M. Behar, S. Nayak, W. Mather, J. Hasty, B. Errede, H.G. Dohlman, and T.C. Elston, *Computational modeling and experimental analysis reveal Bar1's role in yeast chemotrophic growth*, Preprint (2009).

32. H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys. **147** (1998), no. 1, 60–85.

33. K. Kawagoe, D. Kitamura, M. Okabe, I. Taniuchi, M. Ikawa, T. Watanabe, T. Kinoshita, and J. Takeda, *Glycosylphosphatidylinositol-anchor-deficient mice: implications for clonal dominance of mutant cells in paroxysmal nocturnal hemoglobinuria*, Blood **87** (1996), no. 9, 3600–3606.

34. B. N. Kholodenko, *Cell-signalling dynamics in time and space*, Nat. Rev. Mol. Cell Biol. **7** (2006), no. 3, 165–176.

35. B. N. Kholodenko, G. C. Brown, and J. B. Hoek, *Diffusion control of protein phosphorylation in signal transduction pathways*, Biochem. J. **350 Pt 3** (2000), 901–907.

36. V. S. Kraynov, C. Chamberlain, G. M. Bokoch, M. A. Schwartz, S. Slabaugh, and K. M. Hahn, *Localized Rac activation dynamics visualized in living cells*, Science **290** (2000), no. 5490, 333–337.

37. V. M Laurent, S. Kasas, A. Yersin, T. E. Schäffer, S. Catsicas, G. Dietler, A. B. Verkhovsky, and J. Meister, *Gradient of rigidity in the lamellipodia of migrating cells revealed by atomic force microscopy*, Biophys. J. **89** (2005), no. 1, 667–675.

38. C. Lawson and S. Wolf, *ICAM-1 signaling in endothelial cells*, Pharmacol. Rep. **61** (2009), no. 1, 22–32.

39. A. L. Lehninger, D. L. Nelson, and M. M. Cox, *Lehninger principles of biochemistry*, 4th ed., W.H. Freeman, New York, 2005.

40. R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2007.

41. H. Levine, D. A. Kessler, and W. Rappel, *Directional sensing in eukaryotic chemotaxis: a balanced inactivation model*, P. Natl. Acad. Sci. USA **103** (2006), no. 26, 9761–9766.

42. M. Li, Y. Zhou, G. Feng, and S. B. Su, *The critical role of toll-like receptor signaling pathways in the induction and progression of autoimmune diseases*, Curr. Mol. Med. **9** (2009), no. 3, 365–374.

43. Z. Li and K. Ito, *The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains*, Society for Industrial and Applied Mathematics, Philadelphia, 2006.

44. J. Liao, J. A. Spudich, D. Parker, and S. L. Delp, *Extending the absorbing boundary method to fit dwell-time distributions of molecular motors with complex kinetic pathways*, P. Natl. Acad. Sci. USA **104** (2007), no. 9, 3171–3176.

45. T. J. Ligocki, P. O. Schwartz, J. Percelay, and P. Colella, *Embedded boundary grid generation using the divergence theorem, implicit functions, and constructive solid geometry*, J. Phys. Conf. Ser. **125** (2008), 012080 (5pp).

46. R. Malladi, J. A. Sethian, and B. C. Vemuri, *A fast level set based algorithm for topology independent shape modeling*, J. Math. Imaging Vis. **6** (1996), 269–290.

47. A. F. M. Marée, A. Jilkine, A. Dawes, V. A. Grieneisen, and L. Edelstein-Keshet, *Polarization and movement of keratocytes: a multiscale modelling approach.*, Bull. Math. Biol. **68** (2006), no. 5, 1169–1211.

48. P. McCorquodale, P. Colella, and H. Johansen, *A Cartesian grid embedded boundary method for the heat equation on irregular domains*, J. Comput. Phys. **173** (2001), no. 2, 620–635.

49. N. Metropolis, A.E. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, J. Chem. Phys. **21** (1953), 1087–1092.

50. J. Meyers, J. Craig, and D. J. Odde, *Potential for control of signaling pathways via cell size and shape.*, Curr. Biol. **16** (2006), no. 17, 1685–1693.

51. D. Modiano and P. Colella, *A higher-order embedded boundary method for time-dependent simulation of hyperbolic conservation law*, ASME 2000 Fluids Engineering Division Summer Meeting (2000), 1–17.

52. Y. Mori, A. Jilkine, and L. Edelstein-Keshet, *Wave-pinning and cell polarity from a bistable reaction-diffusion system.*, Biophys. J. **94** (2008), no. 9, 3684–3697.

53. Y. Mori and C. S. Peskin, *Implicit second-order immersed boundary methods with boundary mass*, 2008, Comput. Method. Appl. M., pp. 2049–2067.

54. J. D. Murray, *Mathematical biology II: Spatial models and biochemical applications*, 3rd ed., vol. 2, Springer, New York, 2003.

55. P. Nalbant, L. Hodgson, V. Kraynov, A. Toutchkine, and K. M. Hahn, *Activation of endogenous Cdc42 visualized in living cells*, Science **305** (2004), no. 5690, 1615–1619.

56. Sujata Nayak, *Bem1 in yeast*, `http://biodynamics.ucsd.edu/sujata/wmather`.

57. S. R. Neves, P. Tsokas, A. Sarkar, E. A. Grace, P. Rangamani, S. M. Taubenfeld, C. M. Alberini, J. C. Schaff, R. D. Blitzer, I. I. Moraru, and R. Iyengar, *Cell shape and negative links in regulatory motifs together control spatial information flow in signaling networks*, Cell **133** (2008), no. 4, 666–680.

58. B. Novák and J. J. Tyson, *Numerical analysis of a comprehensive model of M-phase control in Xenopus oocyte extracts and intact embryos*, J. Cell Sci. **106 (Pt 4)** (1993), 1153–68.

59. S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys. **79** (1988), no. 1, 12–49.

60. M. Otsuji, S. Ishihara, C. Co, K. Kaibuchi, A. Mochizuki, and S. Kuroda, *A mass conserved reaction-diffusion system captures properties of cell polarity*, PLoS Comput. Biol. **3** (2007), no. 6, e108.

61. M. G. Paulick and C. R. Bertozzi, *The glycosylphosphatidylinositol anchor: a complex membrane-anchoring structure for proteins*, Biochemistry **47** (2008), no. 27, 6991–7000.

62. O. Pertz, L. Hodgson, R. L. Klemke, and K. M. Hahn, *Spatiotemporal dynamics of RhoA activity in migrating cells*, Nature **440** (2006), no. 7087, 1069–1072.

63. C. S. Peskin, *The immersed boundary method*, Acta Numer. **11** (2002), 1–39.

64. L. Rajendran and K. Simons, *Lipid rafts and membrane dynamics*, J. Cell Sci. **118** (2005), no. Pt 6, 1099–102.

65. E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal. **29** (1992), no. 3, 867–884.

66. B. Rubinstein, K. Jacobson, and A. Mogilner, *Multiscale two-dimensional modeling of a motile simple-shaped cell*, Multiscale Model. Simul. **3** (2005), no. 2, 413–439.

67. Y. Saad, *SPARSKIT: A basic tool-kit for sparse matrix computations (version 2)*, `http://www-users.cs.umn.edu/~saad/software/SPARSKIT/sparskit.html`.

68. Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **7** (1986), no. 3, 856–869.

69. J. C. Schaff, B. M. Slepchenko, Y. S. Choi, J. Wagner, D. Resasco, and L. M. Loew, *Analysis of nonlinear dynamics on arbitrary geometries with the Virtual Cell*, Chaos **11** (2001), no. 1, 115–131.

70. I. C. Schneider, E. M. Parrish, and J. M. Haugh, *Spatial analysis of 3' phosphoinositide signaling in living fibroblasts, III: influence of cell morphology and morphological polarity*, Biophys. J. **89** (2005), no. 2, 1420–1430.

71. P. Schwartz, M. Barad, P. Colella, and T. Ligocki, *A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions*, J. Comput. Phys. **211** (2006), no. 2, 531–550.

72. J. A. Sethian, *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, 2nd ed., Cambridge University Press, June 1999.

73. J. Sha, W.and Moore, K. Chen, A. D. Lassaletta, C. Yi, J. J. Tyson, and J. C. Sible, *Hysteresis drives cell-cycle transitions in Xenopus laevis egg extracts*, P. Natl. Acad. Sci. USA **100** (2003), no. 3, 975–980.

74. J. R. Shewchuk, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, Applied Computational Geometry: Towards Geometric Engineering (M. C. Lin and D. Manocha, eds.), Lect. Notes. Comput. Sc., vol. 1148, Springer-Verlag, May 1996, From the First ACM Workshop on Applied Computational Geometry, pp. 203–222.

75. Y. Shimada, M. P. Gulli, and M. Peter, *Nuclear sequestration of the exchange factor Cdc24 by Far1 regulates cell polarity during yeast mating*, Nat. Cell Biol. **2** (2000), no. 2, 117–124.

76. R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications*, Springer, New York, 2000.

77. R.L. Smith, *Time series*, course notes, Department of Statistics, University of North Carolina, Chapel Hill, NC (1999).

78. K. Taskén and E. M. Aandahl, *Localized effects of cAMP mediated by distinct routes of protein kinase A*, Physiol. Rev. **84** (2004), no. 1, 137–167.

79. A. Turing, *The chemical basis of morphogenesis*, Philos. T. R. Soc. **237** (1952), 37–72.

80. E. H. Twizell, A. B. Gumel, and M. A. Arigu, *Second-order, L_0-stable methods for the heat equation with time-dependent boundary conditions*, Adv. Comput. Math. **6** (1996), no. 1, 333–352.

81. J. J. Tyson, K. C. Chen, and B. Novak, *Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell*, Curr. Opin. Cell Biol. **15** (2003), no. 2, 221–231.

82. J. Valdez-Taubas and H. R. B. Pelham, *Slow diffusion of proteins in the yeast plasma membrane allows polarity to be maintained by endocytic cycling*, Curr. Biol. **13** (2003), no. 18, 1636–1640.

83. O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The finite element method: its basis and fundamentals*, 6th ed., Elsevier/Butterworth-Heinemann, Oxford, 2005.