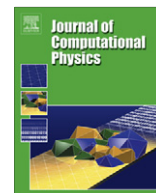




ELSEVIER

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

The moving boundary node method: A level set-based, finite volume algorithm with applications to cell motility

Charles W. Wolgemuth^{a,*}, Mark Zajac^b^a Department of Cell Biology and Center for Cell Analysis and Modeling, University of Connecticut Health Center, Farmington, CT 06030-3505, United States^b Department of Mathematics, University of Utah, Salt Lake City, UT 84112, United States

ARTICLE INFO

Article history:

Received 13 January 2010

Received in revised form 4 June 2010

Accepted 9 June 2010

Available online xxx

Keywords:

Diffusion

Moving boundaries

Numeric method

Level set

Cell motility

Finite volume

ABSTRACT

Eukaryotic cell crawling is a highly complex biophysical and biochemical process, where deformation and motion of a cell are driven by internal, biochemical regulation of a poro-elastic cytoskeleton. One challenge to built quantitative models that describe crawling cells is solving the reaction–diffusion–advection dynamics for the biochemical and cytoskeletal components of the cell inside its moving and deforming geometry. Here we develop an algorithm that uses the level set method to move the cell boundary and uses information stored in the distance map to construct a finite volume representation of the cell. Our method preserves Cartesian connectivity of nodes in the finite volume representation while resolving the distorted cell geometry. Derivatives approximated using a Taylor series expansion at finite volume interfaces lead to second-order accuracy even on highly distorted quadrilateral elements. A modified, Laplacian-based interpolation scheme is developed that conserves mass while interpolating values onto nodes that join the cell interior as the boundary moves. An implicit time stepping algorithm is used to maintain stability. We use the algorithm to simulate two simple models for cellular crawling. The first model uses depolymerization of the cytoskeleton to drive cell motility and suggests that the shape of a steady crawling cell is strongly dependent on the adhesion between the cell and the substrate. In the second model, we use a model for chemical signalling during chemotaxis to determine the shape of a crawling cell in a constant gradient and to show cellular response upon gradient reversal.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

Moving boundary problems are notoriously difficult. Even for linear, dynamical equations, the presence of a free or otherwise mobile boundary can add sufficient complexity to make analytic solutions onerous or impossible. Therefore, computational methods are often necessary. Even then, the ability to accurately track the boundary and solve the bulk physical or chemical equations numerically is challenging. A classical example of this is the Stefan problem that examines heat flux in a material that can undergo temperature-induced phase changes. This long-standing problem is still actively researched using both analytical and numerical techniques [1–5].

Modeling cell motility is even more difficult. Many eukaryotic cells can move when in contact with a surface, the extracellular matrix, or other cells. This process is known as cell crawling and plays a significant role in developmental biology, cancer metastasis, and wound healing. In most eukaryotic cells, the primary structural component is the cytoskeleton, which is composed of actin filaments and microtubules. Crawling motility typically relies on the actin network and has been shown

* Corresponding author.

E-mail addresses: cwolgemuth@uchc.edu (C.W. Wolgemuth), mzajac@alumni.nd.edu (M. Zajac).

to entail three major biophysical processes: protrusion of the actin network at the front of the cell, adhesion to the surface, and pulling forward of the rear of the cell [6,7]. A description of the process by which cells achieve this motility is as follows. Chemical signals from outside the cell activate membrane bound proteins, which act to activate or enhance local actin polymerization [8,9]. Actin polymerization pushes out the front of the cell, most likely driven by a process known as the polymerization ratchet [10]. Once polymerized, the biochemical processes that modify and/or depolymerize the actin network are quite complex [11]. Focal adhesions are highly-regulated assemblies of a large number of different molecules that form at the front of the cell and then mature as they move rearward with the actin network [12]. These adhesions provide anchorage of the cell to the external environment. The mechanism whereby cells pull up the rear is often attributed to an actomyosin contraction [13], but other groups have also suggested that depolymerization of the network may play a role [14,15]. Therefore, a complete model for cell motility needs to evolve a complex biochemical kinetic system that drives internal force production and leads to boundary displacement. As the cell membrane prevents many of the chemical species from leaving the cell, a method that can conserve the total mass of each species is preferred.

Many models for cell motility have focused on steady crawling, where the geometry of the cell is assumed to remain fixed as the cell crawls forward, thereby ignoring the moving boundary problem [15–18]. Recently, the level set method has been used in order to simulate the biochemical reactions inside a cell moving with prescribed boundary conditions [19], and others have used finite element methods to explore the mechanical deformations in a migrating cell [20–22] or the Pott’s model to simulate cell motility driven by chemical reactions [23]. Here we develop the Moving Boundary Node method: a general method for handling moving boundary problems with internal physical and/or chemical constituents that can be used to define the boundary velocity. In Section 2, we describe our method. In Section 3, we construct a number of test cases to determine the accuracy of the method. We show that the algorithm accurately solves reaction–diffusion–advection problems in 2D and also can be used for 2D surfaces embedded in 3D. We then apply this algorithm to simple models for cell crawling. As the algorithm presented here is not the only method that can handle free boundary problems, in Section 4 we compare the Moving Boundary Node Method to existing numerical methods. The final section is the conclusion, which summarizes our method with particular attention to biological applications.

2. Numerical method

The algorithm that we develop here is a composite algorithm that separates boundary motion from the solution of the internal chemistry/physics. We focus on problems in two dimensions, but the methods that we develop are extendable to three dimensions.

We use a standard implementation of the Level Set method to move the boundary [30]. This method defines a signed distance map on a Cartesian grid and evolves it in time. The zeros of this function provide sub-pixel resolution of the boundary. Using the information stored in the distance map, we construct a resolved finite volume discretization of the geometry that is second-order in space, while, topologically, grid nodes retain simple Cartesian connectivity. To evolve the system in time, we construct a discretization of the geometry at time t and at time $t + \Delta t$. Then, using a modified Crank–Nicolson-type scheme that takes into account the change in the volumes between the two time steps, we evolve the contents of each finite volume. A Laplacian-based interpolation scheme is constructed that sets values on new nodes and globally conserves mass. We also describe a method for defining internal flow inside the geometry, which provides an advection velocity that can be used to transport internal chemical components.

The basic flow of our algorithm is as follows:

- (1) Define initial conditions for the distance map and internal chemical/physical components.
- (2) Mask for nodes that are inside the current geometry and interpolate onto newly uncovered nodes, if needed.
- (3) Construct a finite volume discretization of the derivative terms for the current geometry.
- (4) Calculate the boundary velocity.
- (5) Displace the boundary using the Level Set algorithm.
- (6) Using the same masked nodes, compute a new finite volume discretization of the derivative terms for the new geometry.
- (7) Use a semi-implicit method to evolve the contents of each finite volume.
- (8) Repeat steps (2)–(7) for each time step.

For linear dynamic equations, steps (2)–(7) represent a full time step evolution of the dynamics. For nonlinear problems, we propose using a second-order Runge–Kutta method, based primarily on the midpoint rule, as is used in Immersed Boundary methods [24]. For these problems, each time step involves a preliminary substep to the half-time level followed by a final substep from time t to time $t + \Delta t$, and steps (2)–(7) represent the procedure that is used for each substep. The specific details for how these steps are handled are discussed in the following subsections.

2.1. Level set method

The Level Set method is a robust, general method for numerically tracking boundary movements [25–28]. Level set methods use a continuous function, $\psi \in \mathbb{R}^n$, to describe a boundary in \mathbb{R}^{n-1} . The common choice for ψ is the signed distance map,

which is the distance of any point in space to the nearest point on the boundary, where negative values correspond to points inside the boundary. The position of the boundary is defined by the zero level set:

$$\psi(\mathbf{r}) = 0, \quad (1)$$

and the normal direction to any contour of ψ is:

$$\hat{\mathbf{N}} = \frac{\nabla\psi}{|\nabla\psi|}. \quad (2)$$

Motion of the boundary evolves via an advection equation:

$$\frac{\partial\psi}{\partial t} = -\mathbf{V} \cdot \nabla\psi, \quad (3)$$

where \mathbf{V} is a velocity that is defined at all points in space. For many applications, \mathbf{V} is only known at the boundary, in which case an *extension velocity* with the correct limit as $\psi \rightarrow 0$ must be computed for all distant points.

2.1.1. Calculation of the extension velocity

We assume that the value of \mathbf{V} at the boundary is given by the internal biochemistry and physics and that it points along the normal direction, $\mathbf{V} = V\hat{\mathbf{N}}$. To extrapolate this velocity to each point inside the computational domain, we use a modified version of the extension velocity algorithm [29], which uses the orthogonality criterion:

$$\nabla V \cdot \hat{\mathbf{N}} = 0, \quad (4)$$

to define V .

Following [31], we solve Eq. (4) inside a band of points about the zero contour. The narrow band is defined by $|\psi| < 5h$, where h is the grid spacing. We developed an implicit solver to integrate (4) inside this band. We consider $\hat{\mathbf{N}} \cdot \nabla$ to be a matrix operator that can be broken up into two pieces, $N_x D_x + N_y D_y$, where N_x and N_y are the x and y components of $\hat{\mathbf{N}}$, and D_x and D_y are discrete versions of the x and y derivative operators, respectively.

To construct the matrix that corresponds to $N_x D_x \psi$, we first mask for all points in the domain that are inside the narrow band and have $\psi N_x < 0$. For all nodes inside this mask, if the node at (i, j) has neighbors $(i+1, j)$, $(i+2, j)$ and $(i+3, j)$ that are also inside the narrow band, then we construct the third-order accurate approximation to the purely upwind, first derivative:

$$(N_x D_x \psi)_{ij} = \frac{(N_x)_{ij}}{h} \left(-\frac{11}{6} \psi_{ij} + 3\psi_{i+1,j} - \frac{3}{2} \psi_{i+2,j} + \frac{1}{3} \psi_{i+3,j} \right). \quad (5)$$

If the neighbors at $(i+1, j)$ and $(i+2, j)$ are inside the narrow band, but $(i+3, j)$ is outside, then we use the second-order accurate, upwind derivative:

$$(N_x D_x \psi)_{ij} = \frac{(N_x)_{ij}}{h} \left(-\frac{3}{2} \psi_{ij} + 2\psi_{i+1,j} - \frac{1}{2} \psi_{i+2,j} \right), \quad (6)$$

and, if only $(i+1, j)$ is inside the narrow band, then we use the first-order accurate, upwind derivative:

$$(N_x D_x \psi)_{ij} = \frac{(N_x)_{ij}}{h} (-\psi_{ij} + \psi_{i+1,j}). \quad (7)$$

For all points in the narrow band that have $\psi N_x \geq 0$ and have neighbors $(i-1, j)$, $(i-2, j)$ and $(i-3, j)$ that are inside the narrow band:

$$(N_x D_x \psi)_{ij} = \frac{(N_x)_{ij}}{h} \left(\frac{11}{6} \psi_{ij} - 3\psi_{i-1,j} + \frac{3}{2} \psi_{i-2,j} - \frac{1}{3} \psi_{i-3,j} \right). \quad (8)$$

If the neighbors at $(i-1, j)$ and $(i-2, j)$ are inside the narrow band, but $(i-3, j)$ is outside, then:

$$(N_x D_x \psi)_{ij} = \frac{(N_x)_{ij}}{h} \left(\frac{3}{2} \psi_{ij} - 2\psi_{i-1,j} + \frac{1}{2} \psi_{i-2,j} \right), \quad (9)$$

and, if only $(i-1, j)$ is inside the narrow band, then:

$$(N_x D_x \psi)_{ij} = \frac{(N_x)_{ij}}{h} (\psi_{ij} - \psi_{i-1,j}). \quad (10)$$

A similar discretization handles the y components and is easily generalizable to three dimensions. For smooth geometries, this method can provide up to fourth-order accuracy (results not shown).

Standard methods such as Gauss–Jordan elimination or conjugate gradient methods can then be used to implicitly solve the linear system that corresponds to Eq. (4) inside the narrow band. For the simulations in this paper, we coded our routines in MATLAB and used the backslash function to invert this system of equations.

For points outside the narrow band we use the interpolation scheme described in [31] to define values of V that satisfy Eq. (4). This algorithm determines the value of $V(\mathbf{x}_{i,j})$ by interpolating the velocity field at the point $\mathbf{x}_{i,j} - (\psi \hat{\mathbf{N}})_{i,j}$; i.e., points outside the band can be determined completely in terms of points that fall inside the narrow band. In this paper, we use cubic interpolation.

2.1.2. Time evolution of the distance map

At the start of each time step, we test the quality of the distance map. We do this by calculating the magnitude of the gradient of ψ inside a band of width $6h$ about the zero contour. The derivatives are calculated using a WENO discretization [28,30]. If $|\nabla\psi - 1| > 0.01$, then we *reinitialize* the distance map using the method described in [30].

We then solve Eq. (3) on a periodic, Cartesian domain. Spatial discretization is handled using WENO derivatives, and a third-order TVD Runge–Kutta method is used for time stepping [30].

2.2. Finite volume discretization

In this section, we describe our method for constructing a finite volume discretization of the irregular geometry that is defined by the zero contour of a signed distance map. As mentioned previously, we work with a distance map that is defined on a $M \times N$ Cartesian grid. Our method distorts a Cartesian grid, while maintaining node connectivity, to yield a node-centered finite volume representation with boundary-fitted control volumes. We then construct operators that define the first derivatives and the generalized diffusion operator on this irregular geometry.

2.2.1. Grid distortion

The signed distance map describes the distance from any point in space to the nearest point on the boundary; i.e., the nearest point on the boundary from the grid position $\mathbf{x}_{i,j}$ can be defined as:

$$\mathbf{X}_b(\mathbf{x}_{i,j}) = \mathbf{x}_{i,j} - \psi \hat{\mathbf{N}}. \quad (11)$$

We consider the nodes of the Cartesian grid as identifying quadrilateral areas. A quadrilateral is considered to be inside the geometry if interpolation gives a value of ψ less than zero at the center of a quadrilateral (Fig. 1(a)). Boundary nodes are nodes with at least one neighboring grid point outside the boundary (open circles in Fig. 1(a)). Boundary quads are then quadrilaterals which have at least one boundary node as a corner (red x's in Fig. 1(a)). With this scheme, it is possible to get boundary nodes that are only connected to exterior or boundary nodes. These nodes represent unresolved features and are removed from the mask of interior points. The remaining boundary nodes are then displaced using Eq. (11), thereby constructing a discrete representation of the geometry (Fig. 1(b)).

A nice feature of this representation is that each node in the irregular geometry refers to a node located on the Cartesian grid. Therefore, the connectivity of the nodes is given. In addition, the resolved boundary removes the necessity of using ghost point methods, as has been used with other Level Set based algorithms [30]. However, moving the boundary nodes with Eq. (11) can lead to large deformations of the rectangular grid, which can produce inaccuracies when standard finite volume methods are employed [32]. To overcome this difficulty, we have developed a quad-based discretization that maintains second-order accuracy even when the quads become greatly deformed.

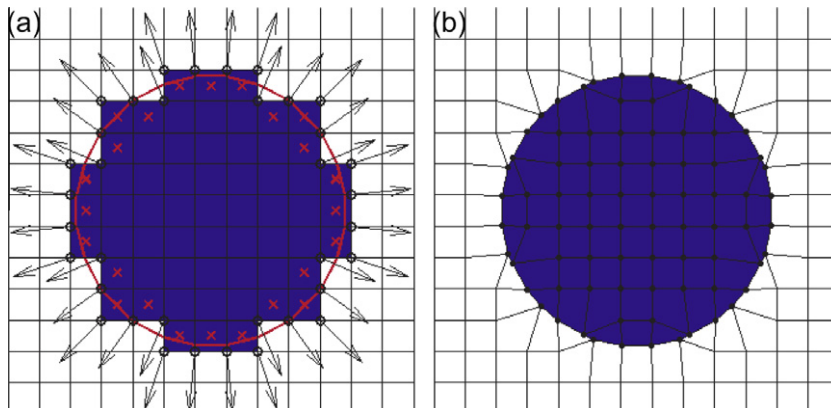


Fig. 1. Construction of the finite volume discretization. (a) Grid boxes with centers inside the boundary are identified; i.e., the value of ψ interpolated onto the center of the quads is less than zero (quads shaded in blue). The edge quads (red x's) and the nodes that lie near the boundary (open circles) are identified. These boundary nodes are displaced a distance $-\psi$ along the normal direction (arrows). (b) Displacement of grid nodes replaces grid boxes with quadrilaterals that fit the boundary. Blue shaded quads are inside the geometry and the nodes are shown as black dots. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

We consider the quad shown in Fig. 2(a) that is defined by nodes 1, 2, 3 and 4. The center of the quad is defined using the center of mass:

$$\mathbf{x}_5 = \frac{1}{4}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4). \quad (12)$$

Focusing on node 1, we define a *subcontrol volume* about this node, Δv_1 , using the node, \mathbf{x}_1 ; the point that bisects the line between nodes 1 and 2, $(\mathbf{x}_1 + \mathbf{x}_2)/2$; the quad center, \mathbf{x}_5 ; and the point that bisects the line between nodes 1 and 4, $(\mathbf{x}_1 + \mathbf{x}_4)/2$. By breaking this subcontrol volume into two triangles and using the standard formula for the area of a triangle, we find the area for this subcontrol volume:

$$\Delta v_1 = \frac{1}{4}|(\mathbf{x}_4 - \mathbf{x}_1) \times (\mathbf{x}_5 - \mathbf{x}_1)| + \frac{1}{4}|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_5 - \mathbf{x}_1)|. \quad (13)$$

Note that this subcontrol volume is not necessarily the total control volume for node 1, as node 1 may also be connected to other quadrilaterals that will contribute to the total control volume (Fig. 2(b)). We distinguish between the total control volume and the subcontrol volume about node 1 by ΔV_1 and Δv_1 , respectively.

2.2.2. Volume integration operators

In a finite volume discretization, it is necessary to compute the integrals of quantities, such as chemical concentrations, over the control volumes [32]. The standard way of handling this integration is to multiply the value of the concentration at the node by the control volume. For nodes that reside on or near the boundary, however, the node is not necessarily located at the center of its control volume, and therefore does not represent the average value of the concentration within the control volume. The extreme cases are the boundary nodes that reside on the edge of their control volume. Using the node values to represent the average value of the concentration inside the volume can introduce a small amount of error and can also lead to changes in the total mass during time stepping.

In place of the standard method, we choose to approximate the integral over each subcontrol volume. Summing the integrals over all the subcontrol volumes about each node then provides an approximation to the integral over the full control volume. Since each of the subcontrol volumes are centered about a non-grid point (see Fig. 2), we use bilinear interpolation to describe the concentration at the centers of the subcontrol volumes. In terms of the shaded subcontrol volume depicted in Fig. 2(b), we define the integral of a function f over this shaded region to be:

$$\int_{\Delta v_1} f dV \equiv \bar{f} \approx \frac{\Delta v_1}{16}(9f_1 + 3f_2 + f_3 + 3f_4), \quad (14)$$

where the bar represents that the quantity is the integral of the function over the subcontrol volume. Therefore, the integral of a function over the total control volume about the i th node is:

$$\int_{\Delta V_i} f dV \approx \sum_{\Delta v_i} \bar{f}. \quad (15)$$

The subscript Δv_i on the sum denotes that the sum is over the subcontrol volumes for the i th node. Eq. (14) couples nearby nodes, and therefore, the integral over the total control volume (Eq. (15)) also couples nearby nodes. We define a matrix $\hat{\mathcal{V}}$ that computes this integral over control volumes given the vector of values of a function f defined on the nodes:

$$\hat{\mathcal{V}}f_i \equiv \sum_{\Delta v_i} \bar{f}. \quad (16)$$

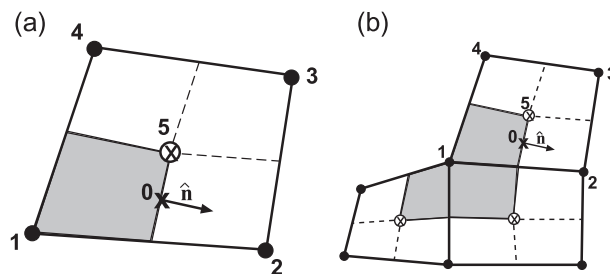


Fig. 2. (a) A distorted quadrilateral area defined by the nodes labeled 1–4. The center of the quad is denoted by \otimes and is labeled as location 5. We focus our attention on the node labeled 1. The subcontrol volume about this point, Δv_1 , is shaded in grey. The text describes computation of the flux through the eastern side of the area, which is centered about the point labeled 0. The normal vector along this side is $\hat{\mathbf{n}}$. (b) Schematic showing the full control volume about node 1, ΔV_1 (shaded in grey), which is subdivided into three subcontrol volumes. (Note that here node 1 is a boundary node. Interior nodes have four control volumes.) To compute the total number of particles entering the control volume of node 1 in a unit time, we integrate the flux over the boundary of ΔV_1 . Volume integration is achieved by integrating over each subcontrol volume and then summing.

2.2.3. First derivative operators

To compute the finite volume approximation to the first derivatives of a function f , we use the identity:

$$\int \nabla f dV = \oint f \hat{\mathbf{n}} dS, \quad (17)$$

where S is the surface enclosing the volume V and $\hat{\mathbf{n}}$ is the outward directed normal to the volume. This identity can be broken into components, leading to:

$$\begin{aligned} \frac{\partial f}{\partial x}(\mathbf{x}_{ij}) &= \frac{1}{\Delta V_{ij}} \oint f n_x dS, \\ \frac{\partial f}{\partial y}(\mathbf{x}_{ij}) &= \frac{1}{\Delta V_{ij}} \oint f n_y dS, \end{aligned} \quad (18)$$

where ΔV_{ij} is the total control volume about the node located at \mathbf{x}_{ij} .

To calculate the contribution to these derivatives at node 1 that comes from the control volume edge that separates the subcontrol volume about node 1 from that about node 2 in Fig. 2(a), we use bilinear interpolation to define the value of f at the quadrilateral center:

$$f_5 = \frac{1}{4}(f_1 + f_2 + f_3 + f_4). \quad (19)$$

Then, the value of f at the center of the control volume edge can be approximated as:

$$f_0 = \frac{3}{8}(f_1 + f_2) + \frac{1}{8}(f_3 + f_4), \quad (20)$$

and the contributions to the derivatives from this edge are:

$$\left. \frac{\partial f}{\partial x} \right|_{1,2} = \frac{f_0 n_x \ell_{1,2}}{\Delta V_1}, \quad (21)$$

$$\left. \frac{\partial f}{\partial y} \right|_{1,2} = \frac{f_0 n_y \ell_{1,2}}{\Delta V_1}, \quad (22)$$

where $\ell_{1,2} = |(\mathbf{x}_5 - \frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_2))|$ is the length of this edge of the control volume ΔV_1 . These contributions are equal to the contribution to the derivatives at node 2 along this same edge. All control volume edges that lie inside the quadrilaterals are defined in a similar fashion.

If a node in the quadrilateral is a boundary node, then we need to calculate the contributions from the boundary edges separately. Consider that the edge between nodes 1 and 4 is a boundary edge (as is shown in Fig. 2(b)). For this case, the contributions to the derivatives at node 1 from the control volume edge defined by the line between nodes 1 and 4 are:

$$\left. \frac{\partial f}{\partial x} \right|_{\text{boundary}} = \frac{-(\frac{3}{4}f_1 + \frac{1}{4}f_4)(\mathbf{x}_4 - \mathbf{x}_1) \cdot \hat{\mathbf{y}}}{\Delta V_1}, \quad (23)$$

$$\left. \frac{\partial f}{\partial y} \right|_{\text{boundary}} = \frac{(\frac{3}{4}f_1 + \frac{1}{4}f_4)(\mathbf{x}_4 - \mathbf{x}_1) \cdot \hat{\mathbf{x}}}{\Delta V_1}, \quad (24)$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are unit vectors in the x and y directions, respectively. We denote the matrix operators that correspond to these discretizations as $\widehat{\mathcal{D}}_x$ and $\widehat{\mathcal{D}}_y$, such that:

$$\widehat{\mathcal{D}}_x f \approx \sum_{\text{faces}} \frac{\partial f}{\partial x}, \quad (25)$$

$$\widehat{\mathcal{D}}_y f \approx \sum_{\text{faces}} \frac{\partial f}{\partial y}. \quad (26)$$

Here the sum is over all of the control volume faces, and the approximate equivalence reflects that these matrices give approximations to the derivatives.

2.2.4. First derivatives related to transport

For equations that have first-order spatial derivatives related to transport, such as would come from a continuity equation, i.e., terms of the form $\nabla \cdot (f\mathbf{v})$, we compute $\partial(fv_x)/\partial x$ and $\partial(fv_y)/\partial y$ using the same method that was described in the previous section for calculating the gradient. Specifically, we use that the contribution to the derivatives at node 1 from the control volume edge between nodes 1 and 2 in Fig. 2(a) is given by:

$$\left. \frac{\partial(f v_x)}{\partial x} \right|_{1,2} = \frac{(f v_x)_0 n_x \ell_{1,2}}{\Delta V_1}, \quad (27)$$

$$\left. \frac{\partial(f v_y)}{\partial y} \right|_{1,2} = \frac{(f v_y)_0 n_y \ell_{1,2}}{\Delta V_1}, \quad (28)$$

where

$$(f v_x)_0 = \frac{3}{8} (f_1 v_{x,1} + f_2 v_{x,2}) + \frac{1}{8} (f_3 v_{x,3} + f_4 v_{x,4}), \quad (29)$$

$$(f v_y)_0 = \frac{3}{8} (f_1 v_{y,1} + f_2 v_{y,2}) + \frac{1}{8} (f_3 v_{y,3} + f_4 v_{y,4}). \quad (30)$$

As before, the total contribution to these derivatives is computed by summing over all faces of the control volumes. We denote the matrices that correspond to the integrals over the control volume of Eqs. (27) and (28) as $\hat{A}_x(v_x)$ and $\hat{A}_y(v_y)$, which are shown here with explicit dependence on the components of the velocity.

2.2.5. Generalized diffusion operator

To construct a scheme that can handle general descriptions of biochemistry and physics inside moving geometries may require the solution to diffusion-like equations that have spatially-dependent diffusion coefficients. Furthermore, if the cytoskeleton behaves like an elastic solid then a physical description of the cytoskeleton may also require an effective diffusion coefficient that depends on direction. Indeed, recent experiments show that the elastic behavior of the cytoskeleton of crawling nematode sperm behaves anisotropically [15]. Therefore, we seek a discretization for terms of the form:

$$\int \left(\frac{\partial}{\partial x} \left(v_x \frac{\partial f}{\partial x} \right) + \frac{\partial}{\partial y} \left(v_y \frac{\partial f}{\partial y} \right) \right) dV = \oint \left(v_x \frac{\partial f}{\partial x} n_x + v_y \frac{\partial f}{\partial y} n_y \right) dS, \quad (31)$$

where v_x and v_y are spatially-dependent diffusion coefficients in the x and y directions, respectively.

To discretize the righthand side of Eq. (31), we consider a Taylor series expansion of the scalar function f about the midpoint along the eastern face of Δv_1 (denoted by \mathbf{X} in Fig. 2):

$$f(\mathbf{x}) \approx f_0 + x \left. \frac{\partial f}{\partial x} \right|_0 + y \left. \frac{\partial f}{\partial y} \right|_0. \quad (32)$$

In this equation, there are three unknown constants: f_0 , $\left. \frac{\partial f}{\partial x} \right|_0$ and $\left. \frac{\partial f}{\partial y} \right|_0$. To evaluate these terms, we use the value of f at nodes 1, 2 and 5, with the value of f at node 5 given by Eq. (19). The approximation to the derivatives is then:

$$\left. \frac{\partial f}{\partial x} \right|_0 \approx \frac{1}{A} ((f_5 - f_1)(y_2 - y_1) - (f_2 - f_1)(y_5 - y_1)), \quad (33)$$

$$\left. \frac{\partial f}{\partial y} \right|_0 \approx \frac{1}{A} ((f_2 - f_1)(x_5 - x_1) - (f_5 - f_1)(x_2 - x_1)), \quad (34)$$

where

$$A = (x_5 - x_1)(y_2 - y_1) - (x_2 - x_1)(y_5 - y_1). \quad (35)$$

This approximation for the first derivatives (Eqs. (33) and (34)) is equivalent to a bilinear interpolation.

Using Eq. (31), we can define a general, finite volume diffusion operator:

$$\hat{\mathcal{L}}(v_x, v_y)f = \sum_{\text{faces}} \ell_{\text{face}} \left(v_x \left. \frac{\partial f}{\partial x} \right|_0 n_x + v_y \left. \frac{\partial f}{\partial y} \right|_0 n_y \right). \quad (36)$$

As our discretization of this second derivative operator (Eq. (36)) couples nodes to their nearest and next nearest neighbors through the Taylor series approximation of the derivatives on the control volume edge (Eqs. (33) and (34)), we are able to maintain second-order accuracy even on highly distorted grids.

2.3. Crank–Nicolson method on a moving boundary

A general reaction–diffusion–advection equation can be written as:

$$\frac{\partial C}{\partial t} = \nabla \cdot (v \nabla C) - \nabla \cdot (C \mathbf{v}) + \gamma(\mathbf{x})C + \mathcal{N}(C, \mathbf{x}, t), \quad (37)$$

where $\gamma(\mathbf{x})$ is a spatially-dependent function that does not depend on the concentration, C , and $\mathcal{N}(C, \mathbf{x}, t)$ is a nonlinear function of the concentration that can also depend on space and time. For example, $\gamma(\mathbf{x})$ might represent a spatially-varying decay rate, and \mathcal{N} could be a nonlinear reaction term, such as is used in logistic growth models. Integrating Eq. (37) over a small volume, we get:

$$\int \frac{\partial C}{\partial t} dV = \oint (v \nabla C - C \mathbf{v}) \cdot \hat{\mathbf{n}} dS + \int (\gamma(\mathbf{x}) + \mathcal{N}(C, \mathbf{x}, t)) dV. \quad (38)$$

The first integral on the righthand side of Eq. (38) can be discretized using the methods described in the previous sections, and the second integral can be treated by multiplying the terms evaluated at the nodes by the volume integration operator, $\hat{\mathcal{V}}$. To discretize the lefthand side of the equation we note that the volume changes in time with the motion of the nodes. In addition, the time derivative in Eq. (38) should be taken at a fixed location. Therefore, we begin by working in terms of the volume at the beginning of the time step. If the positions of the nodes at time t are given by \mathbf{x}_0 , then the positions at later times are $\mathbf{x} = \mathbf{x}_0 + \dot{\mathbf{x}}t$, where $\dot{\mathbf{x}}$ is the velocity of the nodes. Therefore, $\dot{\mathbf{x}}$ defines the manner in which the volume changes. Intuitively, the divergence of $\dot{\mathbf{x}}$ gives the flux of new volume into dV , which is swept out when the surface of dV moves at the given velocity. To make this more concrete, we consider how the product $C dV$ changes with time:

$$\frac{d}{dt}(C dV) = dV \frac{dC}{dt} + C \frac{d(dV)}{dt} = dV \left(\frac{\partial C}{\partial t} + \dot{\mathbf{x}} \cdot \nabla C \right) + C dV (\nabla \cdot \dot{\mathbf{x}}) = dV \left(\frac{\partial C}{\partial t} + \nabla \cdot (C \dot{\mathbf{x}}) \right), \quad (39)$$

where we have used the identity:

$$\frac{d}{dt} dV = (\nabla \cdot \dot{\mathbf{x}}) dV. \quad (40)$$

It is clear that the first term in the last line of Eq. (39) corresponds to the lefthand side of Eq. (38). The lefthand side of Eq. (39) can be discretized using the standard discretization of the time derivative and the volume integration operator:

$$\frac{d}{dt}(C dV) = \frac{\hat{\mathcal{V}}^{t+\Delta t} C^{t+\Delta t} - \hat{\mathcal{V}}^t C^t}{\Delta t}. \quad (41)$$

Since our method for creating the finite volume discretization only moves boundary nodes, over the course of one time step, the boundary nodes are displaced with the boundary velocity, and all other nodes remain fixed at their Cartesian locations. For this case, $\dot{\mathbf{x}}$ is zero for all internal nodes and equal to the boundary velocity at boundary nodes.

We use the time discretization given in Eq. (41) and average the values of the linear terms at time t and $t + \Delta t$ to construct a semi-implicit time evolution scheme. Using the relation for the time rate of change of the volume element, Eq. (40) leads to the discretization of Eq. (38):

$$\left(\hat{\mathcal{V}}^{t+\Delta t} - \frac{\Delta t}{2} \hat{\mathcal{M}}^{t+\Delta t} \right) C^{t+\Delta t} = \left(\hat{\mathcal{V}}^t + \frac{\Delta t}{2} \hat{\mathcal{M}}^t \right) C^t + \Delta t \mathcal{N}^t, \quad (42)$$

where the linear operator $\hat{\mathcal{M}}$ is defined as:

$$\hat{\mathcal{M}} = \hat{\mathcal{L}}(v_x, v_y) - \hat{\mathcal{A}}_x(v_x - \dot{x}) - \hat{\mathcal{A}}_y(v_y - \dot{y}) - \gamma(\mathbf{x}) \hat{\mathcal{V}}. \quad (43)$$

Notice that the advection operators depend on two different velocities, the physical velocity that transports the chemical, plus the boundary velocity, $\dot{\mathbf{x}} = \dot{x}\hat{\mathbf{x}} + \dot{y}\hat{\mathbf{y}}$. These velocities go into the advection matrices with opposite sign. It can be shown that this discretization (Eq. (42)) is equivalent to a finite volume method in space and time, similar to the construction that is described in [33].

Eq. (42) is the principle time evolution equation that we use. It represents a linear system of equations of the form $Ax = b$, where x denotes the unknown concentrations at time $t + \Delta t$. For all simulations described in this paper, we solve this system of equations by generating the large matrix A and then perform a direct solve on the system (our algorithms are coded in MATLAB and we use the backslash operator to find the solution). For larger systems of equations, a conjugate gradient method, such as gmres, can be used.

To get second-order accuracy for nonlinear terms, we suggest using a second-order Runge–Kutta scheme to evolve the equations through a full time step. For this procedure, each full time step is broken up into two substeps, a preliminary substep to the half-time level followed by a final substep from time t to time $t + \Delta t$. Eq. (42) is used to handle each substep. That is, we discretize the full time step as:

$$\left(\hat{\mathcal{V}}^{t+\frac{\Delta t}{2}} - \frac{\Delta t}{4} \hat{\mathcal{M}}^{t+\frac{\Delta t}{2}} \right) C^{t+\frac{\Delta t}{2}} = \left(\hat{\mathcal{V}}^t + \frac{\Delta t}{4} \hat{\mathcal{M}}^t \right) C^t + \frac{\Delta t}{2} \mathcal{N}^t, \quad (44)$$

$$\left(\hat{\mathcal{V}}^{t+\Delta t} - \frac{\Delta t}{2} \hat{\mathcal{M}}^{t+\Delta t} \right) C^{t+\Delta t} = \left(\hat{\mathcal{V}}^{t+\frac{\Delta t}{2}} + \frac{\Delta t}{2} \hat{\mathcal{M}}^{t+\frac{\Delta t}{2}} \right) C^{t+\frac{\Delta t}{2}} + \Delta t \mathcal{N}^{t+\frac{\Delta t}{2}}. \quad (45)$$

This procedure is used in Immersed Boundary methods and leads to second-order accuracy [24].

2.4. Mass conserving interpolation scheme

After each time step, we again consider which nodes are inside the boundary using the criterion described in Section 2.2.1. What this means is that between time steps we have two separate discretizations of the same geometry, one being the new nodes and the other being the old set of nodes that were selected using the distance map at the previous time step. With changes in geometry, two different scenarios can add new boundary or interior points that must be initialized by

interpolation. If the boundary moves such that a previously exterior node joins the boundary, then there is no information stored at this node at the beginning of the time step. The other case occurs when either a node that was on the boundary becomes an interior node, or an interior node becomes a boundary node. In these cases, displaced nodes must be updated to accurately reflect different conditions at the new location. For Neumann boundary conditions, newly displaced boundary or interior points must each be set correctly at the beginning of the time step. If Dirichlet boundary conditions are being used, then it is easy to define the value of new nodes on the boundary; however, new interior nodes still need to be properly initialized. We have developed an interpolation scheme that is second-order accurate and that conserves mass. A standard polynomial interpolation scheme is not sufficient to maintain mass conservation and requires choosing among different stencils for polynomial interpolation, depending on which neighboring points are available.

Instead of using polynomial interpolation, we modify a Laplacian-based interpolation scheme [34]. Given a defined boundary condition, the Laplacian acts to create the smoothest function (i.e., it minimizes the integral of the squared magnitude of the gradient) that satisfies those boundary conditions. Therefore, if one solves:

$$\nabla^2 C = 0, \quad (46)$$

on all nodes that need to be initialized, the result will smoothly interpolate the concentration field onto those nodes.

Direct application of this method does not guarantee that the integral of the concentration over the entire domain will remain constant. To enforce this condition, we propose an energy functional:

$$\mathcal{E} = \frac{1}{2} \int |\nabla C|^2 dV + \mathcal{A} \left(\int C dV_{\text{new}} - \int C dV_{\text{old}} \right), \quad (47)$$

where \mathcal{A} is a global Lagrange multiplier that enforces that the integrated concentration over the new set of nodes is equal to the integral over the old set of nodes. Minimization of this energy functional with respect to the concentration leads to the Euler–Lagrange equation:

$$\nabla^2 C = \mathcal{A}, \quad (48)$$

with the constraint that:

$$\int C dV_{\text{new}} = \int C dV_{\text{old}}. \quad (49)$$

We solve Eqs. (48) and (49) for the concentration of the new nodes that need initialized and for \mathcal{A} . The effect of the Lagrange multiplier is equivalent to an additional tiny source or sink at each grid node, which exactly compensates for any overall gain or loss of matter that might otherwise come from the interpolation. Note that though this scheme conserves the total mass of protein inside the cell, it does not constrain the enclosed volume of the cell, which is dependent on the boundary velocity.

We use the generalized diffusion operator defined in Section 2.2.5 to discretize the Laplacian in Eq. (48), and the nodes that did not change from one time step to the next are treated as Dirichlet points. It is important to solve this equation using the same boundary conditions as in the time evolution equations for the chemical species. For example, consider the case where a molecular species inside the cell can bind and unbind from the membrane (as is considered in the Balanced Inactivation model that we consider in Section 3.6). For this case, the boundary condition for the freely-diffusing species is that the total flux (the sum of the advective and diffusive fluxes) of the chemical species at the membrane is equal to the rate that the freely-diffusing species is produced at the membrane (Eq. (79)). Therefore, the interpolation routine should incorporate this boundary condition, in order to provide an accurate value for the concentration on newly uncovered points.

2.5. Calculation of fluid flows inside the moving geometry

The migration of crawling cells is driven by a menagerie of proteins. Some of these proteins are soluble and flow with the motion of the fluid inside the cell (the cytoplasm). Other proteins, such as actin and microtubules (which are both cytoskeletal proteins), form long polymers that don't necessarily move with the cytoplasm. Still other proteins, such as myosin, can bind to the polymeric proteins inside the cell. To simulate the dynamics of any of these proteins requires a description of the flow of the cytoplasm or the polymer. However, to date, consensus on what physics accurately describes these flows is lacking. The inside of a cell is a complex environment. As alluded to above, experiments show that the cytosolic flow inside a cell is not entrained with the cytoskeletal flows. For example, at the leading edge of a fish keratocyte, there is retrograde flow of the actin cytoskeleton, yet the cytosol flows forward [35]. Therefore, there are at least two interpenetrating “fluids” inside a motile cell. These two fluid interact through viscous drag. The cytoskeletal component protrudes at the leading edge and contracts in other regions. The fluid component flows in response to the pressure and the motions of the cytoskeleton.

If we are interested in solving for the motions and reactions of soluble proteins inside the cell, then we need an accurate model for the cytosol. However, this requires an accurate model for the physics of the cytoskeleton. In the absence of a complete understanding of the physics of this complicated system, it is still possible to make reasonable guesses as to the flow of the cytosol. Here we describe one such choice. It should be noted at the outset, though, that this choice is not required to use the algorithm that is described in the preceding sections; the algorithm can handle arbitrarily defined advective velocities.

If we consider the cell membrane to be impermeable, then the motion of the boundary defines the cytosolic velocity at the cell membrane. A common assumption that is used in the literature is to model the cytosolic flow using Darcy's law, since the cytosol flows through the porous cytoskeleton. Darcy's law sets the velocity of the fluid to be proportional to the gradient of the pressure. If one then assumes incompressibility of the fluid, the equation that governs the flow is given by the Laplacian of the pressure being equal to zero, with boundary condition that the normal component of the gradient of the pressure is proportional to normal component of the boundary velocity. There is a problem here, though. Because the cytoskeleton can contract and expand, the cytosol does not behave like a single-phase incompressible fluid (i.e., the divergence of the cytosolic velocity need not equal zero). Indeed, direct application of Darcy's law will lead to singularities in the fluid flow if there is even an extremely small change in the area of the cell. Since a cell is really a three-dimensional object, there can be source flows that provide the additional fluid to allow the cell to change two-dimensional area. We could model this using a source term that is proportional to the pressure, which would lead to a modified Darcy's law where the divergence of the velocity is proportional to the pressure. The constant of proportionality, though, is unknown. If we treat this constant as being infinite (i.e., that pressure gradients are instantly equilibrated by fluid flow), then the Laplacian of the velocity is equal to zero:

$$\nabla^2 \mathbf{v}_f = 0, \quad (50)$$

where \mathbf{v}_f is the fluid velocity. This equation is also equivalent to assuming that the cytosolic velocity is the smoothest velocity field that satisfies the boundary conditions.

Unlike Darcy's law, which only requires a boundary condition on the normal component of the velocity, Eq. (50) requires a boundary condition on both components of the velocity. In general, boundary motion results from steady translation of the cell and deformations of the cell shape that do not move the center of mass. In some cases, though, only the normal component of the boundary velocity is given, and this normal component subsumes both the steady translation and the deformation. We propose a method for recovering the distinct translational and deformation velocities, with deformation velocity $V_d \hat{\mathbf{n}}$ along the normal direction at each boundary node, and constant translational velocity, \mathbf{v}_s , common to all boundary points. We begin by decomposing the boundary velocity into its components along the tangential and normal direction, $\mathbf{v}_f|_b = V_t \hat{\mathbf{t}} + V_n \hat{\mathbf{n}}$, where $\hat{\mathbf{t}}$ is the tangent vector of the boundary, and we assume that V_n is given. Therefore, we have that:

$$\mathbf{v}_f|_b = V_t \hat{\mathbf{t}} + V_n \hat{\mathbf{n}} = \mathbf{v}_s + V_d \hat{\mathbf{n}}. \quad (51)$$

From the tangential component of this equation we get that:

$$\mathbf{v}_s \cdot \hat{\mathbf{t}} = V_t. \quad (52)$$

Since the deformation velocity does not move the center of mass, the integral of $V_d \hat{\mathbf{n}}$ over the entire boundary must be equal to zero, which leads to the following equation for the \mathbf{v}_s :

$$\mathbf{v}_s \cdot \oint_{\delta\Omega} (\hat{\mathbf{I}} - \hat{\mathbf{t}}\hat{\mathbf{t}}) ds = \oint_{\delta\Omega} V_n \hat{\mathbf{n}} ds, \quad (53)$$

where $\hat{\mathbf{I}}$ is the identity matrix, and $\delta\Omega$ is the boundary of the cell, which is parameterized by s . In other words, the steady translational velocity \mathbf{v}_s is uniquely determined if the normal component of the boundary is given. The velocity of the fluid at the boundary is then:

$$\mathbf{v}_f|_b = \mathbf{v}_s + (V_n - (\mathbf{v}_s \cdot \hat{\mathbf{n}})) \hat{\mathbf{n}}, \quad (54)$$

which is used as the boundary condition in Eq. (50).

3. Results

3.1. Time independent solution on a stationary boundary

As a first test of our method, we explore the accuracy of our spatial discretization using a simple, time independent, reaction–diffusion problem on a circular domain. We look at the test case of a single, diffusing chemical species that degrades via a first order kinetic process:

$$D\nabla^2 C - \gamma C = 0, \quad (55)$$

where D is the diffusion coefficient and γ is the decay rate. For simplicity, we set $D = \gamma = 1$ and solve the equations on a unit circle with the flux boundary condition:

$$\nabla C \cdot \hat{\mathbf{n}}|_{\text{boundary}} = -1. \quad (56)$$

Eq. (55) can be solved in cylindrical coordinates and the solution only depends on the radial direction, r . The analytic solution to this equation is:

$$C(r) = \frac{I_0(r)}{I_1(1)}, \quad (57)$$

where I_n is the n th modified Bessel function of the first kind. Our simulations are in good agreement with the analytic solution (Fig. 3). We varied the grid spacing, h , using grids that were 30×30 , 50×50 , 70×70 , 100×100 , 150×150 and 200×200 and computed the root mean squared error, L_2 , (Fig. 3(c)) and the maximum error, L_∞ , (Fig. 3(d)), as a function of h . Both errors converge with second-order accuracy. The fact that L_∞ converges with second-order accuracy is important, as our algorithm for constructing the finite volume representation only distorts control volumes near the boundary. Therefore, it is possible that the L_2 error would converge with roughly second-order accuracy even if our finite volume discretization was not second-order accurate, since the number of distorted volumes (i.e., nodes where our discretization method are different than a standard finite difference scheme) scales like the perimeter of the geometry, and the number of undistorted volumes scales with the area. If there were large errors coming from boundary elements, then this should be apparent in L_∞ . To further confirm this accuracy, a more stringent test of the spatial discretization is given in the next section.

3.2. Diffusion on a two-dimensional surface embedded in three dimensions

As mentioned in the previous section, in two dimensions, only control volumes that are near the boundary get distorted. To test our discretization on a fully distorted grid, we consider the problem of diffusion on the surface of a sphere. We begin with the distance map for a unit sphere, $\psi = \sqrt{x^2 + y^2 + z^2} - 1$, and create a distorted grid using the procedure described in Section 2.2.1 generalized to three dimensions (Fig. 4(a) and (b)). The boundary nodes in this distorted grid represent a closed, two-dimensional surface. We consider the problem of diffusion on this surface:

$$\frac{\partial C}{\partial t} = \nu \nabla^2 C, \quad (58)$$

with initial condition:

$$C(\theta, \phi, t = 0) = 2 + 0.5 \cos \phi \sin \theta + \cos 2\phi \sin^2 \theta + \cos 3\phi \sin^3 \theta. \quad (59)$$

Here, θ is the angle with respect to the z -axis and ϕ is the azimuthal angle. Eq. (58) can be discretized using the same procedure that was described for planar two-dimensional geometries. The analytic solution to Eq. (58) for this initial condition is:

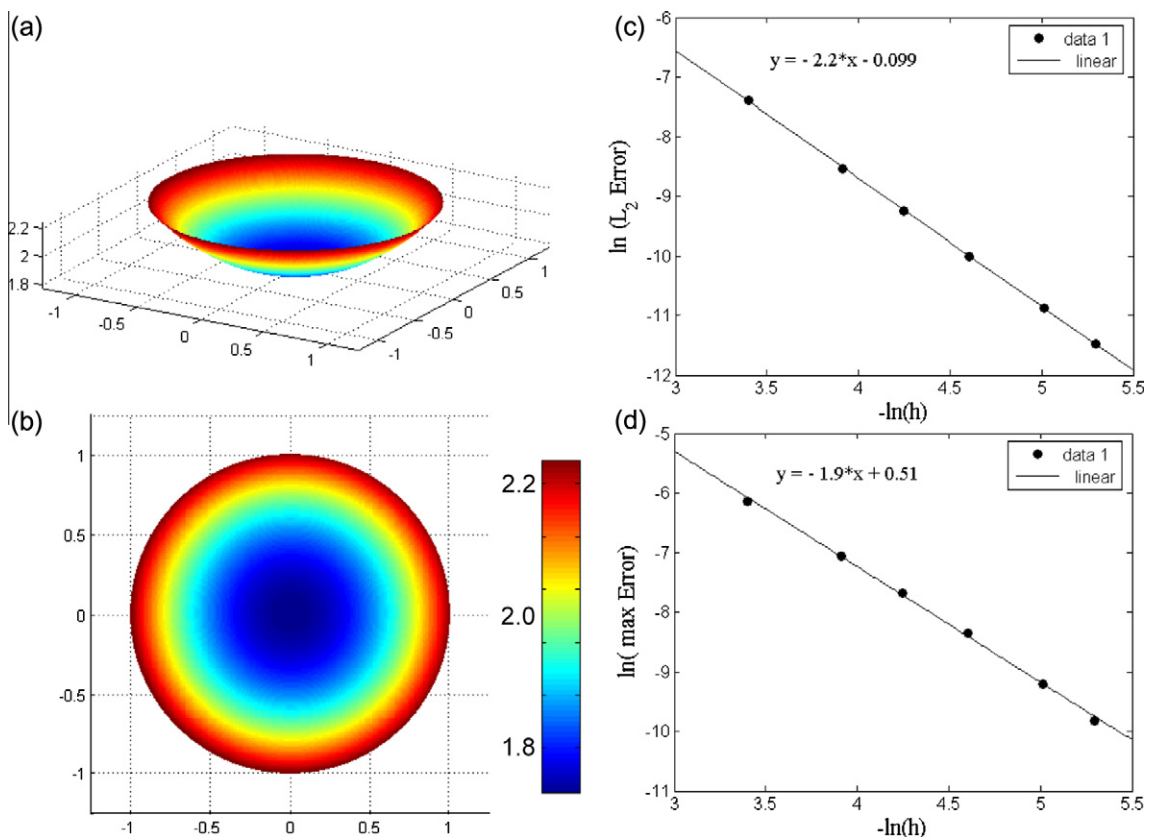


Fig. 3. Simulation result and error analysis for the two-dimensional steady state problem. (a) and (b) Surface plots of the solution to the diffusion/decay problem with Neumann boundary conditions. (c) L_2 and (d) L_∞ errors as a function of grid spacing, showing that the spatial discretization is second order.

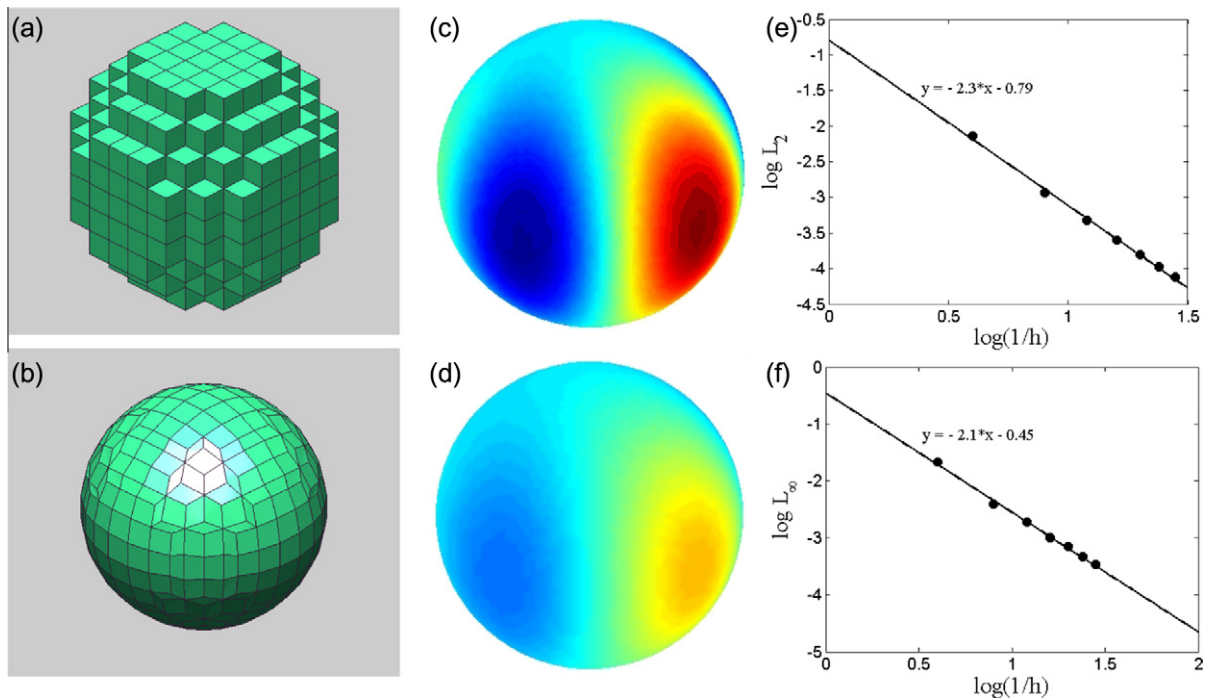


Fig. 4. Grid distortion, simulation and error analysis for the diffusion on the surface of a sphere. (a) We begin with a staircased, spherical geometry. (b) Using the distance map for a sphere, nodes on the surface of the staircase are moved onto the surface of the sphere to form a finite volume representation of the surface. We test the discretization using a diffusion problem with a spatially-dependent concentration field, as described in the text. (c) Shows the initial concentration field and (d) shows the concentration field at time $t = 1$. Colormap in (c) and (d) goes from 0.6 (blue) to 4.3 (red). (e) L_2 and (f) L_∞ errors as a function of grid spacing, showing that the spatial discretization is second order. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$C(\theta, \phi, t) = 2 + 0.5 \cos \phi \sin \theta e^{-2t} + \cos 2\phi \sin^2 \theta e^{-6t} + \cos 3\phi \sin^3 \theta e^{-12t}. \quad (60)$$

We tested the convergence of the solution as a function of grid spacing using seven different grids with dimensions of $10 \times 10 \times 10$ up to $70 \times 70 \times 70$ and a time step of 10^{-4} . We compared our simulation results to the analytic solution at time $t = 1$ (Fig. 4(c)–(f)). Computing the L_2 and L_∞ errors, we find that the method is second-order with respect to the grid spacing (Fig. 4(e) and (f)). This same problem was explored in [36], which uses an algorithm that maps a triangulated grid into the tangent plane to handle diffusion on an embedded surface. Our algorithm slightly out performs this algorithm in terms of accuracy and rate of convergence.

3.3. Advection–diffusion inside a steadily-moving geometry

In this section we show that our method is second-order accurate, conserves the total mass of chemical components, and that handling fluid flows inside the cell using our method (Section 2.5) provides an accurate means for moving chemical components with the geometry. We consider the case of diffusion inside an impermeable boundary. Because the boundary is impenetrable, the fluid inside the cell must get carried along as the boundary moves. Therefore, diffusion of a chemical species inside this steadily-translating cell should behave as if the cell were not moving.

We study the test problem of a steadily-translating circular geometry of unit radius that moves with a velocity $\mathbf{v}_0 = V_0 \hat{y}$. A freely-diffusing chemical species is released at time $t = 0$, with an initial profile inside the geometry:

$$C(t = 0) = 2 + J_0(mr), \quad (61)$$

where J_0 is the 0th order Bessel function, $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$, and m^2 is the first root of the Bessel function. Here, x_0 and y_0 are the coordinates of the center of the circle at time $t = 0$. The chemical species gets carried along by a fluid velocity inside the cell, \mathbf{v}_f , which is computed using as described in Section 2.5. Therefore, we consider the problem:

$$\frac{\partial C}{\partial t} = D \nabla^2 C - \mathbf{v}_f \cdot \nabla C, \quad (62)$$

on a steadily-translating circular geometry with zero flux boundary conditions and initial condition given by Eq. (61). The analytical solution to this equation at time t is:

$$C(t) = 2 + J_0(mr)e^{-m^2t}. \quad (63)$$

For this problem we can define a Peclet number as the magnitude of the velocity times the radius of the circle divided by the diffusion coefficient. Since we ran simulations with unit radius and diffusion coefficient, this Peclet number is equal to the magnitude of the velocity. We found that when the velocity was large, that the error from the spatial discretization dominated the total error; i.e., for time steps smaller than the maximum time step that is set by the CFL condition [27,28], changing the time step only weakly affects the error. In order to determine the temporal accuracy of the simulation, we varied the time step and grid spacing simultaneously, using a fixed ratio between the two. The initial condition and the solution at $t = 0.1$ are shown in Fig. 5(a) and (b). For this simulation, the diffusion coefficient was $D = 1$ and the geometry moved at a steady velocity $\mathbf{v}_0 = 5\hat{\mathbf{y}}$. Using this velocity and grids of size 33×33 , 45×45 , 57×57 , 69×69 , 83×83 , 95×95 , 107×107 and 119×119 , we set the time step to be equal $0.125h$ and find that this method converges to the analytic solution with second-order accuracy when we vary the time step and grid spacing simultaneously (Fig. 5(c)). We also ran a number of simulations using velocities between 0 and $8\hat{\mathbf{y}}$, and varied the time step and grid spacing and obtained the same convergence.

To determine the precision of the algorithm and interpolation scheme at conserving mass, we calculated the integral of the concentration over the entire area of the cell as a function of time. We find that the total chemical mass inside the cell is conserved to roughly machine accuracy; the total change in mass over a simulation was $\lesssim 10^{-13}$.

3.4. Reaction–diffusion with a physics-driven boundary

The previous examples have considered test cases with a steadily-translating geometry. Many problems of interest, however, have free boundaries where the internal physics controls the displacement of the boundary. To test the accuracy of our method for a case where the boundary is driven by the internal physics, we consider a simple reaction–diffusion equation where the chemical species degrades via a first order kinetic process with rate constant γ :

$$\frac{\partial C}{\partial t} = \nabla^2 C - \gamma C. \quad (64)$$

If the chemical species represents the cytoskeleton, then degradation of the concentration could cause the cell to shrink. Therefore, we use a physics-driven boundary velocity in the normal direction given by:

$$V_b = -\nabla C \cdot \hat{\mathbf{n}}. \quad (65)$$

We consider an initially circular geometry with radius 1 and a cylindrically-symmetric initial condition:

$$C(t = 0) = I_0(r). \quad (66)$$

We used grids that were 33×33 , 41×41 , 61×61 , 81×81 , 101×101 and 121×121 and a time step of 5×10^{-3} and compared the results of our simulations to the solution generated by a highly-resolved one-dimensional simulation evaluated at time $t = 0.5$. Our algorithm maintains radial symmetry and converges to the highly-resolved solution (Fig. 6); however, the convergence is only first order in space (Fig. 6(c)). The reason that we only get first order convergence is that our discretization of the gradient (Section 2.2.3), is only first order for the boundary nodes. A second-order approximation to the derivatives at the boundary nodes can be created by interpolating onto two points that lie backward along the normal direction from each boundary node and then using a second-order finite difference approximation to the derivative (similar to the method described in [45]). For a boundary point at location \mathbf{x}_b , we interpolate the value of the concentration at points

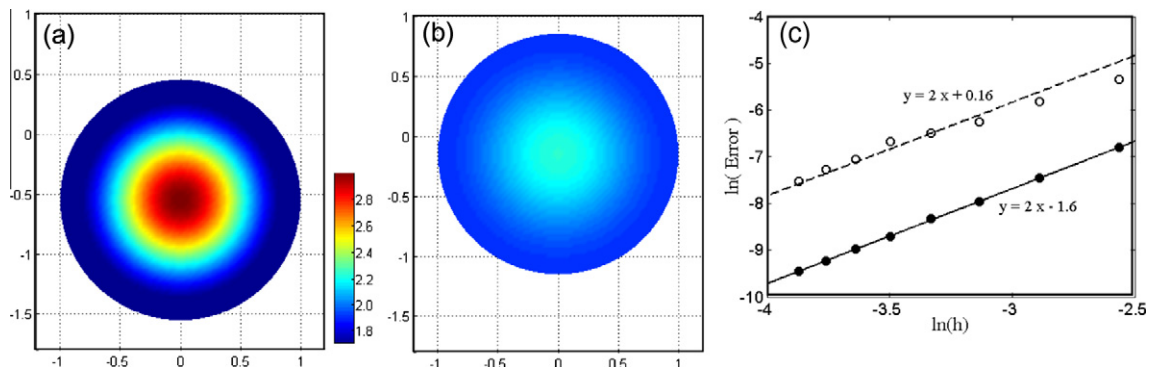


Fig. 5. Diffusion of a chemical species inside a moving cell with impermeable boundaries. The cell moves with velocity $\mathbf{v} = 5\hat{\mathbf{y}}$. (a) Concentration profile at time $t = 0$. (b) Concentration profile at time $t = 0.1$. (c) Convergence of the numerical solution to the analytical solution when the time step and grid size are changed together, with a constant ratio between them. The L_2 -norm (solid circles) and the maximum errors (open circles) are shown as a function of grid spacing. Lines show linear fits to the data, corresponding to second order convergence.

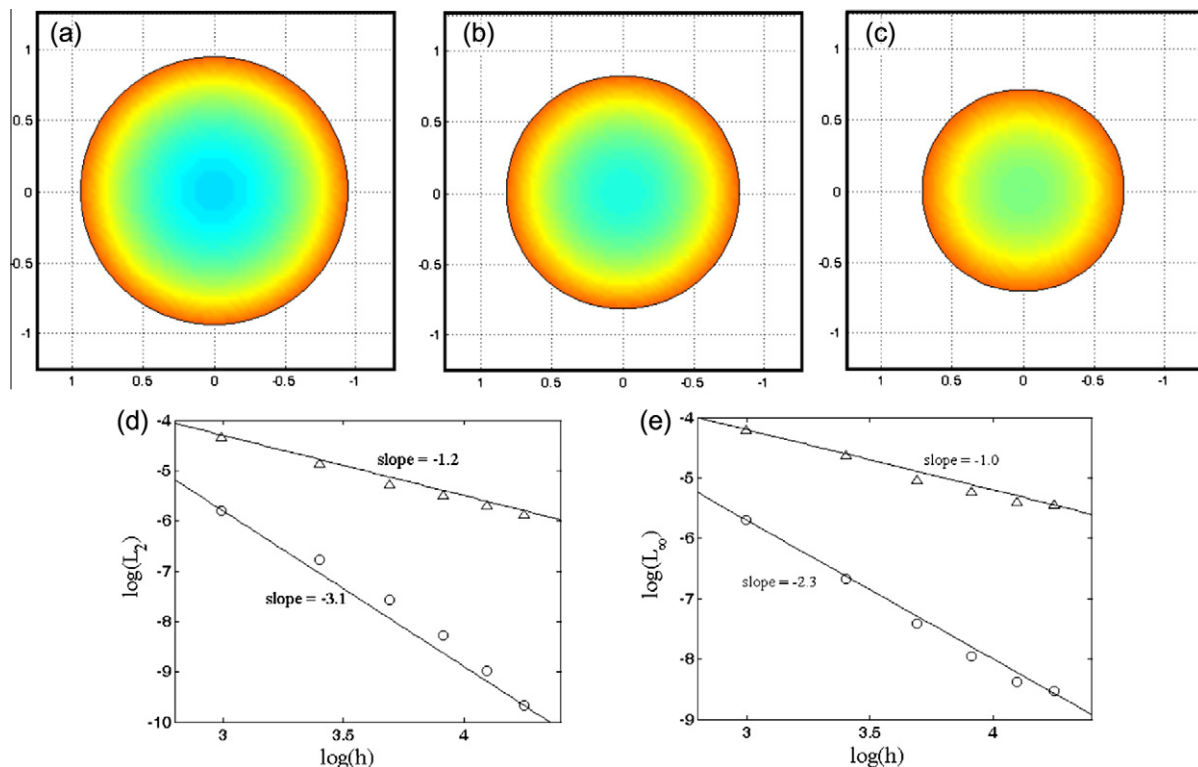


Fig. 6. Reaction–diffusion of a chemical species with a physics-driven boundary. Decay of a chemical species causes the domain to shrink. The boundary velocity is proportional to the normal component of the gradient of the chemical species at the boundary. Solution at time $t = 0.1$ (a), $t = 0.3$ (b) and $t = 0.5$ (c). Colorscale varies between 0.8 (blue) and 1.4 (red). The L_2 (d) and L_∞ (e) errors as a function of grid spacing. Using the discretization of the derivative given in Section 2.2.3 leads to first order convergence (triangles), whereas a second order finite difference approximation of the normal component of the gradient gives second order convergence (circles). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$\mathbf{x}_1 = \mathbf{x}_b - 3\Delta\mathbf{x}\mathbf{n}$ and $\mathbf{x}_2 = \mathbf{x}_b - 4\Delta\mathbf{x}\mathbf{n}$, where \mathbf{n} is the normal vector at \mathbf{x}_b . The normal component of the gradient of the concentration is then given by:

$$\frac{\partial C}{\partial n}\bigg|_b = \frac{7C(\mathbf{x}_b) - 16C(\mathbf{x}_1) + 9C(\mathbf{x}_2)}{12\Delta x}. \quad (67)$$

Using this method to calculate the normal component of ∇C , we find better accuracy and roughly third-order convergence to the highly resolved solution (Fig. 6(d)).

3.5. Cell crawling powered by depolymerization

As mentioned in the Introduction, eukaryotic cell crawling is a complex process where biochemical reactions inside the cell are coupled to the biophysical mechanisms that move and deform the cell. Therefore, to construct realistic models for cell motility requires simulating reaction–diffusion equations inside a moving boundary. In this section, we develop a simple model for cell motility and show that our moving boundary algorithm provides a useful numerical scheme for addressing these kinds of problems.

Some of the simplest crawling cells are the spermatozoa from nematodes [46]. The sole function of these cells is the transport of genetic material. Indeed, the crawling sperm cell ejects many cellular components prior to activating. Therefore, these cells may represent a fundamental, stripped-down crawling motility apparatus. As in other crawling cells, motility is driven by a meshwork of polymer filaments. In place of actin filament, however, nematode sperm use a cytoskeleton composed of Major Sperm Protein (MSP). Polymerization of MSP at the leading edge of the cell pushes the front edge of the cell forward. The mechanism that the cell uses to adhere to the substrate is unknown, but recent experiments show that the adhesion at the front of the cell is stronger than at the rear [15]. In vitro experiments using cell-free extracts have suggested that depolymerization of the MSP network is responsible for producing the force that hauls the rear of the cell forward [14], and biophysical models for depolymerization–driven contraction of the MSP cytoskeleton agree well with experimental data [15,47]. In addition, recent experiments using Scanning Force Microscopy to measure forces produced by the actin cytoskeleton during the crawling of fish keratocytes also suggest that depolymerization may play a role in generating cytoskeletal flows in

these cells [48]. Therefore, it is interesting to explore what shapes are produced by a depolymerization-driven model of cell motility.

In [15], a model for depolymerization of MSP was used to analyze experiments that related the size and shape of crawling cells to the translational velocity. Results suggested that the MSP filaments behave like a depolymerizing, anisotropic, linearly elastic solid. In that model, effects of the cytoskeleton and the cytosolic fluid were taken into account. To test our algorithm with the simplest cell crawling mechanism, we ignore the cytosolic effects. We work in terms of the MSP volume fraction, ϕ . If adhesion between the substrate and the cytoskeleton acts as a resistive drag, then the polymer velocity is proportional to the divergence of a stress, $\zeta \mathbf{v}_p = \nabla \cdot \boldsymbol{\sigma}$, where $\zeta(\mathbf{x})$ is a drag coefficient that can depend on the spatial coordinates. The stress is assumed to be linear in ϕ , diagonal and anisotropic [15]:

$$\boldsymbol{\sigma} = \sigma_0(\phi_0 - \phi) \begin{pmatrix} 1 & 0 \\ 0 & \alpha^{-1} \end{pmatrix}, \quad (68)$$

where σ_0 is the hydrostatic compressibility, ϕ_0 is the equilibrium volume fraction, and α is the anisotropy. The dynamics for ϕ are given by the continuity equation augmented by depolymerization at rate γ . Linearization of this equation about $\phi \approx \phi_0$ leads to:

$$\frac{\partial \phi}{\partial t} = \sigma_0 \phi_0 \left(\frac{\partial}{\partial x} \left(\frac{1}{\zeta} \frac{\partial \phi}{\partial x} \right) + \frac{1}{\alpha} \frac{\partial}{\partial y} \left(\frac{1}{\zeta} \frac{\partial \phi}{\partial y} \right) \right) - \gamma \phi. \quad (69)$$

Experimental evidence suggests that there is a sharp transition between high and low drag between the cytoskeleton and the substrate. To model this we use a drag coefficient given by:

$$\zeta(\mathbf{x}) = C_1 + \frac{C_2}{2} \left(1 + \tanh \left(\frac{R - \Delta_1}{\lambda_1} \right) \right), \quad (70)$$

where C_1 and C_2 are constants. For our simulations, we use $C_1 = 5$ and $C_2 = 25$. Here, R is the distance from the center of an ellipse:

$$R = \sqrt{\beta(x - x_0)^2 + (y - y_0)^2}, \quad (71)$$

with (x_0, y_0) the position of the rear of the cell, and β defines the eccentricity of the ellipse. Δ_1 defines the distance from the rear of the cell to the roll over from high to low drag, and λ_1 is the width of this rollover region. For all of our simulations, $\lambda_1 = 0.2$.

The surface tensions in the membrane of crawling cells has been estimated previously [15,21]. The values obtained suggest that the surface tension is small compared to the stresses generated in the cytoskeleton. We assume, though, that the cell has a preferred area (or volume in 3D), A_0 . Deviations from this area induce stress in the polymer. Therefore, Laplace's law at the boundary is:

$$\boldsymbol{\sigma} \cdot \hat{\mathbf{n}} - \frac{\Gamma(A - A_0)}{A_0} = 0, \quad (72)$$

where Γ is a constant. This boundary condition defines ϕ at the boundary.

The motion of the boundary is driven by two effects, polymerization of the MSP and motion of the existing polymer. We assume that the polymerization of the MSP at the boundary pushes out in the direction normal to the boundary and has a spatially-dependent magnitude, V_a . Therefore, the normal component of the velocity at the boundary of the cell is:

$$\mathbf{v}_b \cdot \hat{\mathbf{n}} = V_a + \mathbf{v}_p \cdot \hat{\mathbf{n}}. \quad (73)$$

We assume that V_a is positive at the front of the cell and zero near the rear:

$$V_a = C_3 \left(1 + \tanh \left(\frac{R - \Delta_2}{\lambda_2} \right) \right). \quad (74)$$

For our simulations, we used that $C_3 = 0.1$, $\Delta_2 = 1.6$ and $\lambda_2 = 0.5$.

We solved Eq. (69) on a periodic, 50×50 grid of size 2×2 with a time step $\Delta t = 0.05$. We used an initial condition of a unit circle with $\phi = \phi_0$ everywhere. The location of the center of the ellipse given by R is tracked by moving the point (x_0, y_0) with the average velocity of the cell. To see how the placement of the rollover in the substrate adhesion affects the cell shape during crawling, we ran simulations with $\Delta_1 = 1.2$ and 0.7 . The former case places the rollover in drag past the center of the circle. We find that with these conditions the dynamics quickly produce a shape that is slightly pointed near the rear and is convex everywhere along the boundary (Fig. 7). The shape steadily translates and its globally convex boundary make it crudely similar to the shape of a crawling nematode sperm cell.

For the case when the drag rollover is closer to the rear of the cell, $\Delta_1 = 0.7$, we also find that the cell quickly reaches a steadily-translating shape. In this simulation, because we track the position of the rear of the cell (x_0, y_0) , using the average velocity, the point (x_0, y_0) detaches from the true back of the cell. This acts to make the rollover in the drag even closer to the

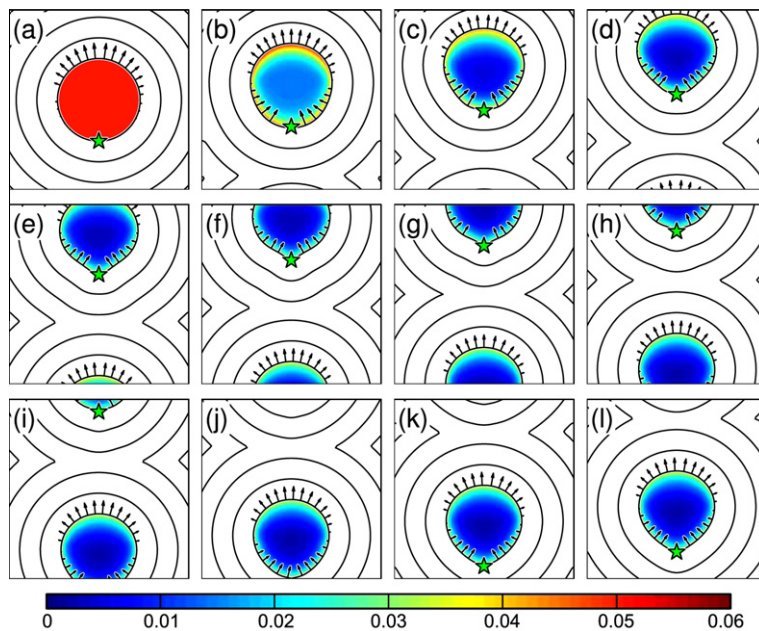


Fig. 7. Simulation of a crawling cell driven by depolymerization. Panels (a)–(l) are evenly spaced time intervals of a simulated crawling cell. The colormap shows the volume fraction. Contours of the distance map and the boundary velocity (arrows) are also shown. The star shows the position of the rear of the cell. This simulation uses $\Delta_1 = 1.2$. Note that the rear of the cell is convex.

back of the cell. Interestingly, the rear of the cell becomes concave, and the overall morphology resembles a half-moon. This shape is characteristic of crawling keratocytes [22,49,50]. Even though actin depolymerization has not been suggested as a force-producing mechanism in these cells, any bulk contraction may behave in a similar fashion [51]. In keratocytes, myosin II contraction at the rear of the cell may act to reduce the effective drag close to the rear of the cell [52]; therefore, this localization of lower drag may be responsible for the concavity of the back of the cell (see Fig. 8).

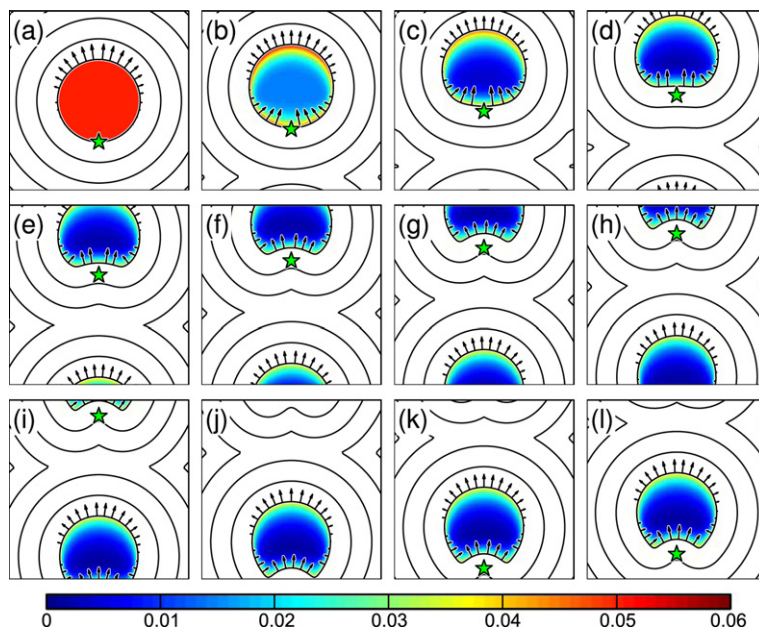


Fig. 8. Simulation of a depolymerization-driven crawling cell with drag rollover closer to the rear of the cell. Panels (a)–(l) are evenly spaced time intervals. The colormap shows the volume fraction. Contours of the distance map and the boundary velocity (arrows) are also shown. The star shows the position of the rear of the cell. This simulation uses $\Delta_1 = 0.7$. The cell becomes “half-moon shaped” and crawls at a steady rate.

3.6. Cellular chemotaxis oriented by balanced inactivation

Many crawling cells are chemotactic. Cells sense their environment and move toward or away from chemical signals. As mentioned in the Introduction, receptors in the cell membrane activate membrane bound proteins, which begin a signalling cascade that locally triggers or enhances actin polymerization. The activation of membrane bound proteins has been shown to amplify the external signals, which allows cells to measure very small chemical gradients. For example, neutrophils and *Dictyostelium discoideum* cells can measure a concentration difference of roughly 1% across the length of the cell [53,54]. One strength of our algorithm is that at the boundary nodes lie explicitly on the boundary. Therefore, it is easy to couple chemical reactions between membrane bound species with cytosolic factors, which is not true for algorithms that use cut cells and ghost points to represent the sharp interface [19].

To illustrate the applicability of our algorithm to these types of problems, we consider a simple chemical system that has been proposed as a mechanism for directional sensing during eukaryotic chemotaxis, which is known as a balanced inactivation model [55]. A strength of this model is that it captures a number of features of directional sensing, such as adaptation and the response of a cell in a high background of signal, which are not captured in other chemotaxis signalling models.

The balanced inactivation model proposes that chemical signal outside of the cell, denoted by S , triggers the cleavage of a membrane bound protein into a component that remains on the membrane, A , and a component that is released into the cytoplasm, B . The cytosolic component can diffuse with diffusion constant D and rebind to the membrane with rate k_b , thereby becoming the membrane bound species B_m . Diffusion of the membrane bound species is assumed to be negligible. On the membrane, species A and B_m spontaneously degrade with rates k_{-a} and k_{-b} , respectively. In addition, species B_m can bind to A with a rate constant k_i . This reaction inactivates species A . Therefore, the time evolution of the chemical species are given by:

$$\begin{aligned}\frac{\partial A}{\partial t} &= k_a S - k_{-a} A - k_i A B_m \quad (\text{on the membrane}), \\ \frac{\partial B_m}{\partial t} &= k_b B - k_{-b} B_m - k_i A B_m \quad (\text{on the membrane}), \\ \frac{\partial B}{\partial t} &= D \nabla^2 B \quad (\text{in the cytosol}),\end{aligned}\tag{75}$$

where k_a is the rate constant for cleaving the membrane bound component. To account for production and binding of B to the membrane, the boundary condition for the final equation is:

$$D \nabla B \cdot \mathbf{n} = k_i S - k_b B.\tag{76}$$

This model was solved previously on a stationary circular boundary [55]. If the background signal concentration is uniform, $S = S_0$, then it can be shown analytically that the steady state values of the chemical species are:

$$\begin{aligned}B_0 &= \frac{k_a S_0}{k_b}, \\ B_{m,0} &= \frac{k_b B_0}{k_i A_0 + k_{-b}}, \\ A_0 &= \frac{-k_{-a} k_{-b} + \sqrt{(k_{-a} k_{-b})^2 + 4 k_a k_i k_{-a} k_{-b} S_0}}{2 k_i k_{-a}}.\end{aligned}\tag{77}$$

Here, we set out to determine the equilibrium shape that is generated by this model and the dynamic response of the cell morphology upon a rapid change in the external signal gradient. We assume that the concentration of A determines the polymerization rate of actin at the membrane. If actin polymerization is predominantly normal to the boundary, then the polymerization of actin will push the boundary in the normal direction. In addition, we define a preferred contact area between the cell and the substrate, a_0 , since cells are observed to maintain a fairly constant area (i.e., a_0 is the 2D area of the cell). Therefore, we set the boundary velocity of the cell to be:

$$\mathbf{v}_b = \left(\frac{V_0 A}{A_0} + \frac{\beta(a_0 - a)}{a_0} \right) \mathbf{n},\tag{78}$$

where V_0 and β are constants and a is the total contact area of the cell and the substrate. Motion of the cell membrane generates a flow of the cytosolic fluid, \mathbf{v}_f , which transports any cytosolic proteins. We model this flow using the methods described in Section 2.5. With the addition of this cytosolic velocity, the time evolution equation for B becomes:

$$\frac{\partial B}{\partial t} = -\nabla \cdot (B \mathbf{v}_f - D \nabla B),\tag{79}$$

with boundary condition:

$$(D \nabla B - B \mathbf{v}_f) \cdot \mathbf{n} = k_a S - k_b B.\tag{80}$$

We solved the Balanced Inactivation equations on an initially circular domain of diameter $5\ \mu\text{m}$ in a linear signal gradient given by:

$$S = S_0(1 + 0.025y/R), \quad (81)$$

with $S_0 = 10$ and R being the radius of the cell. We used $V_0 = 0.0625\ \mu\text{m/s}$ and $\alpha = 25\ \mu\text{m/s}$. The other parameters of the model are taken directly from the original paper [55]; i.e., $k_a = 1\ \text{s}^{-1}$, $k_i = 1000\ \mu\text{m}\ (\text{s molecule})^{-1}$, $k_b = 3\ \mu\text{m/s}$, $k_{-a} = 0.2\ \text{s}^{-1}$, $k_{-b} = 0.2\ \text{s}^{-1}$ and $D = 10\ \mu\text{m}^2/\text{s}$.

Our first scenario simulates a cell in a fixed, constant signal gradient. We solve the equations on a $10\ \mu\text{m} \times 10\ \mu\text{m}$ (101×101 nodes) grid, using a fixed time step of $0.005\ \text{s}$. The only nonlinearities in this system come in the reaction rates of the membrane bound components. Therefore, we used a second-order Runge–Kutta algorithm to integrate the membrane bound components (first two equations in (75)) and used the fully implicit version of our algorithm, described in Section 2, to evolve the cytosolic concentration B . As our simulations are done on a periodic domain, we adjust the gradient as the cell crawls in order to maintain the correct signal concentration on the periodic domain. We find that the initially circular shape stretches out as the cell begins to crawl up the gradient (Fig. 9(a)–(d)). The cell takes on a tear drop shape, and the rear pulls forward slightly before the cell reaches a steady translating shape (Fig. 9(e)–(h)). The black arrows in Fig. 9 show the boundary velocity, which is linearly related to concentration of species A . The cell quickly polarizes with a high concentration of A developing from the slightly higher concentration of signal at the top portion of the cell. It is interesting to note that the polarization and magnitude of the concentration of A reaches a steady state, even though the absolute concentration of the external signal increases as the cell crawls. This is a realistic feature of the Balanced Inactivation model that is not captured by lateral excitation/global inhibition models. The model produces shapes that are extended in the direction of motion; however, they are not overly reminiscent of the actual crawling of *D. discoideum* cells, which have a more uniform width. Clearly, a more accurate model for the internal dynamics of the cytoskeleton is needed to capture the actual morphology of *Dictyostelium*.

To explore how the shape of a cell driven by the Balanced Inactivation model responds to a time-dependent gradient, we model an initially circular cell that is in an initially constant gradient of external signal. Between times 15 and 19 s, the gradient is switched to point in the opposite direction (Fig. 10). Within less than 10 s of reversing the gradient, the cell begins to crawl in the new direction. Interestingly, the final morphology is half-moon shaped and reminiscent of a keratocyte, as opposed to the tear drop-shaped morphology that is observed during the crawling stages in a constant gradient (compare Figs. 9 and 10).

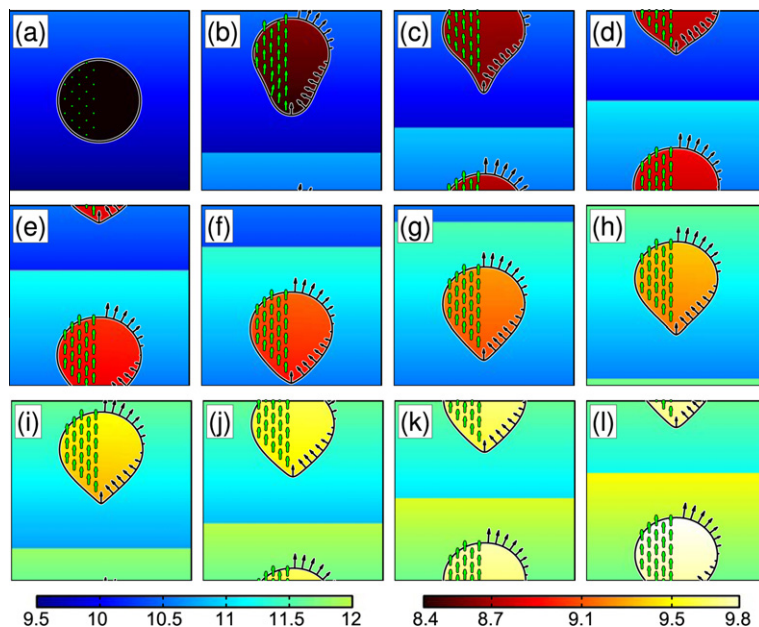


Fig. 9. Simulation of a crawling cell driven by the Balanced Inactivation model in a linear external chemical gradient. Panels correspond to evenly spaced intervals of time with 9 s between frames. (a)–(l) The cell begins with a circular shape in a gradient that increases toward the top of the page. The cell stretches and eventually becomes tear drop shaped. The rear of the cell pulls up slightly as the cell reaches a steady crawling shape. The internal colormap shows the concentration of the diffusing cytosolic protein B . The black arrows show the boundary velocity (proportional to the concentration to the concentration of species A as in Eq. (75)), and the green arrows show the internal cytosolic velocity. The color bars give concentrations in molecules per square micron, with external concentration on the left and internal concentration on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

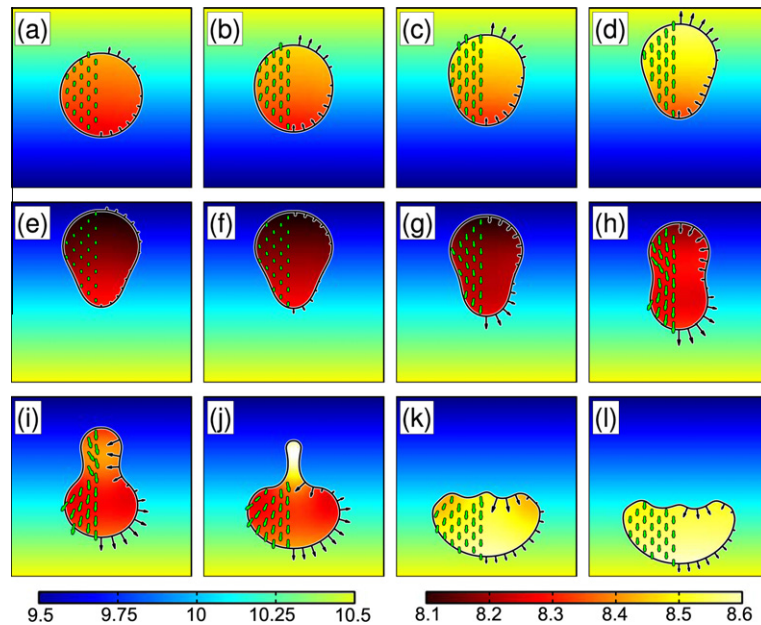


Fig. 10. Simulation of a crawling cell driven by the Balanced Inactivation model in a linear external chemical gradient that switches direction. Panels correspond to evenly spaced intervals of time with 3.75 s between frames. (a)–(d) The cell begins with a circular shape in a gradient that increases toward the top of the page. (e) Between 15–19 s, the direction of the gradient is switched to point in the opposite direction. Panels (f)–(l) show the response of the cell to the change in the external gradient. The background coloring shows the external chemical gradient. The internal colormap shows the concentration of the diffusing cytosolic protein B . The black arrows show the boundary velocity, and the green arrows show the internal cytosolic velocity. The color bars give concentrations in molecules per square micron, with external concentration on the left and internal concentration on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

These simulations show that the Moving Boundary Node Method can handle simulations where reactions between membrane bound and cytosolic proteins drive cell motility. Therefore, this method is well-suited for handling many problems that will arise during eukaryotic cell migration.

4. Comparison to other methods

Our algorithm is not the only numerical method for solving PDEs inside moving boundaries. In this section, we compare our method to some of the existing methods that can also be used to solve these types of problems.

For any moving boundary problem, a method must be chosen for defining the location of the boundary as a function of time. The two main methods for defining the position of the boundary are material point methods, such as the immersed boundary method [24], and level set methods [27,28]. A material point method defines the position of fixed points on a boundary and then evolves the positions of these points using the local velocity of the boundary. A major advantage of this method is that the new positions of the material points are easily calculated from the given velocity. However, care must be taken to prevent a chain of material points from crossing itself, to form loops. Typically, spring-like interactions are used to maintain a fixed connectivity between the material points. Two further drawbacks material point methods are that they do not handle merging or splitting of boundaries (such as might occur during cell fusion or division) and geometric properties of the boundary that are computed numerically, such as the normal direction and curvature, are prone to numerical errors that can lead to instabilities.

The level set method is an attractive alternative method that is widely used. The distance map representation of the boundary allows accurate calculation of geometric properties of the boundary and the implicit description of the boundary easily handles merging or splitting of boundaries. The primary drawback to the level set method is that reinitialization and high order methods (such as WENO) need to be employed to preserve the quality of the distance map [28], and time stepping is more computationally intensive than material point methods, since the boundary is represented by a function that lives in a higher-dimensional space. For problems in cell motility, the geometric properties of the boundary are important, and there are often cases where maintaining the connectivity of material points might be difficult, such as in the retraction of a long, thin filipodium. Therefore, we opted to use the level set method to track the boundary of moving cells in our algorithm. Other groups have also begun to use the level set method to track the boundary of motile cells obtained from experimental images [37], to compute membrane diffusion [38], to model tumor growth [31], and to model chemotaxis in crawling cells [39]. Building on the strengths of the level set method, we then use the geometric information stored in the distance map to

construct the finite volume representation of the computational domain; i.e., no additional calculation are required to determine the boundary nodes that are used in our method.

Level set methods have been used to solve bulk physical equations inside moving boundaries without displacing grid nodes. These methods typically rely on using ghost nodes that lie outside the boundary. For example, Fedkiw et al. developed a ghost cell method for multimaterial flows [30] and similar methods have been applied to solve tumor growth models [40,31]. These methods do not provide nodes that lie on the boundary and mass conservation can be difficult to maintain. Because we use a finite volume discretization, mass conservation is natural. Unstructured grids [21] or body-fitted coordinates [41] could provide an equivalent finite volume representation but involve computationally intensive node redistribution after each boundary displacement. Instead, our method uses the distance map to reposition boundary nodes in a single step for each update leaving interior nodes on a fixed Cartesian grid.

A cut cell method [42] could also be used in place of our finite volume discretization. In these methods, the Cartesian grid is not modified; however, the contributions from grid cells that contain the boundary are approximated using the apertures that are computed from the boundary positions. The computational cost for computing these contributions is roughly comparable to the cost of computing the contributions from our distorted grids, and our method transparently avoids tiny control volume, which must be detected and eliminated in the cut cell method. In addition, cut cell methods do not provide nodes that lie on the boundary, which is also a drawback of adaptive mesh refinement [43]. During cell motility, membrane bound proteins often provide biochemical activation or inhibition of intracellular processes. Since the moving boundary node method provides explicit nodes on the boundary that are directly attached to interior nodes, it is straightforward to calculate membrane diffusion and biochemical reaction kinetics for chemical species on the membrane interacting with cytosolic proteins, which is not the case for cut cell methods. Indeed, our method also provides a method for solving for membrane diffusion that does not require spreading the membrane onto a higher-dimensional grid, as has been used recently [38,44].

A number of published alternatives combine finite volume and implicit surface methods, without moving nodes onto the boundary. Conveniences of these include the following: the potential for systematic extension to accuracy beyond second order [57]; accurate discretization of normal derivative jump boundary conditions [58]; direct application to medical imaging without complicated difference approximations for metric terms in surface derivatives [38]; and variable time steps for internal and surface nonlinear reaction terms [60]. However, the current implementations of these methods also come with a number of drawbacks, such as restriction to immobile surfaces [57,38,60] or, with loss of conservation form, rigidly translating surfaces [59]; diffusion restricted to either the surface [38] or the interior [57–60]; presumption of a quasi-steady state for interior reactions and diffusion [58]. Specialization makes alternatives to the Moving Boundary Node Method more advantageous for some problems but less advantageous for others. Straightforward hybridization has been proposed, almost universally, by the authors of each method, yet, at present, no single alternative can handle all the biologically relevant scenarios covered by the Moving Boundary Node Method (Section 3).

5. Conclusions

Here we have presented a finite volume-based algorithm for handling the dynamics of reaction–diffusion–advection equations inside a moving boundary. The boundary motion is tracked using standard level set method techniques, which provide a robust and accurate method for following physics-driven boundary displacements. In addition, the distance map provides the proper information to construct a discretization of irregular geometries using the nodes of a Cartesian grid; however, the procedure can lead to highly distorted quadrilateral areas near the boundary of the geometry. Therefore, to maintain second-order accuracy in our spatial discretization, we consider the Taylor series expansion about the centers of the surface areas that enclose the finite volumes in our discretization. The Taylor series expansion couples neighboring points as well as nearest neighbors.

For the problems that we considered here, the most time consuming aspect of our algorithm was the reinitialization of the distance map. However, we did find that for large grids, the computational cost of constructing the matrices for the spatial discretization of the derivatives became the time limiting step. As we have not worked to optimize our algorithms, we have not systematically studied the optimal computational efficiency of the Moving Boundary Node Method.

One goal of this work was to construct a moving boundary algorithm that would maintain the total mass in a conservative system. Two aspects of our method are required to achieve this. First, because edge nodes are not located at the centers of their control volumes, time stepping can cause loss of mass at the geometry boundary. Therefore, the integral over a control volume that is associated with a node needs to be computed using subcontrol volume averages that are computed at the centers of the subcontrol volumes. Second, as our method gains and loses nodes as the boundary moves, an interpolation scheme is needed that preserves the total mass. We use a Laplacian integration scheme that is modified by a Lagrange multiplier to conserve the total amount of material inside the cell. The combination of these two procedures, conserves the total chemical mass inside the simulated geometry to roughly machine accuracy.

One field where we expect this method to be especially useful is in modeling eukaryotic cell motility. In these cells, complex biochemical reactions are coupled to the biophysical mechanisms that drive cell motility. Changes to the cellular biochemistry have been shown to lead to morphological changes in the crawling cell [56]. Feedback between the cell shape and the biochemistry are expected to be complex and will require good numerical methods for handling the moving boundary

problem. In addition, conservation of the internal chemical components over the course of a simulation is imperative. Here we have shown that this algorithm can simulate the physics-driven motions of a simple model for cell motility. We show that the profile for the adhesion between the cell and the substrate may play a major role in determining the shape of a crawling cell. In addition, we have shown that the Balanced Inactivation model proposed in Levine et al. can produce steady cell crawling in a fixed external chemical gradient.

The example simulations that we have done consider cellular motion driven by the presence of chemical species inside the cell. For simplicity, we have assumed that the concentration of the protein species at the boundary defines the normal component of the velocity. This assumption is not a requirement of our algorithm. Our algorithm can also handle cases where the protein species sets the tangential component of the velocity. For these cases, the tangential component of the velocity will not affect the displacement of the boundary nor the displacement of the boundary nodes. However, the protein species that are defined at or on the boundary will be advected by any tangential flows. These tangential velocities need only to be included in the advection matrices.

References

- [1] S.C. Gupta, *The Classical Stefan Problem: Basic Concepts, Modeling and Analysis*, Elsevier Science, Amsterdam, 2003.
- [2] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving Stefan problems, *J. Comput. Phys.* 135 (1997) 8–29.
- [3] J.A. Sethian, J.D. Strain, Crystal growth and dendritic solidification, *J. Comput. Phys.* 98 (1992) 231–253.
- [4] F. Gibou, R. Fedkiw, A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem, *J. Comput. Phys.* 202 (2005) 577–601.
- [5] A. Karma, W.-J. Rappel, Quantitative phase-field modeling of dendritic growth in two and three dimensions, *Phys. Rev. E* 57 (1997) 4323–4349.
- [6] D.A. Lauffenburger, A.F. Horwitz, Cell migration: a physically integrated molecular process, *Cell* 84 (1996) 359–369.
- [7] T.J. Mitchison, L.P. Cramer, Actin-based cell motility and cell locomotion, *Cell* 84 (1996) 371–379.
- [8] R. Rohatgi, P. Nollau, H.-Y.H. Ho, M.W. Kirschner, B.J. Mayer, Nck and phosphatidylinositol 4,5-bisphosphate synergistically activate actin polymerization through the N-WASP-Arp2/3 pathway, *J. Biol. Chem.* 276 (2001) 26448–26452.
- [9] C. Le Clainche, D. Pantaloni, M.-F. Carrier, ATP hydrolysis on actin-related protein 2/3 complex causes debranching of dendritic actin arrays, *Proc. Nat. Acad. Sci. USA* 100 (2003) 6337–6342.
- [10] A. Mogilner, G. Oster, Force generation by actin polymerization II: the elastic ratchet and tethered filaments, *Biophys. J.* 84 (2003) 1591–1605.
- [11] T.D. Pollard, L. Blanchoin, R.D. Mullins, Molecular mechanisms controlling actin filament dynamics in nonmuscle cells, *Annu. Rev. Biophys. Biomol. Struct.* 29 (2000) 545–576.
- [12] M. Vicente-Manzanares, J. Zareno, L. Whitmore, C.K. Choi, A.F. Horwitz, Regulation of protrusion, adhesion dynamics, and polarity by myosins IIA and IIB in migrating cells, *J. Cell Biol.* 176 (2007) 573–580.
- [13] H.E. Huxley, Muscular contraction and cell motility, *Nature* 243 (1973) 445–449.
- [14] L. Miao, O. Vanderlinde, M. Stewart, T.M. Roberts, Retraction in amoeboid cell motility powered by cytoskeletal dynamics, *Science* 302 (2003) 1405–1407.
- [15] M. Zajac, B. Dacanay, W.A. Mohler, C.W. Wolgemuth, Depolymerization-driven flow in nematode spermatozoa relates crawling speed to size and shape, *Biophys. J.* 94 (2008) 3810–3823.
- [16] A. Mogilner, L. Edelstein-Keshet, Regulation of actin dynamics in rapidly moving cells: a quantitative analysis, *Biophys. J.* 83 (2002) 1237–1258.
- [17] A.T. Dawes, 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) 744–768.
- [18] J.-F. Joanny, F. Julicher, J. Prost, Motion of an adhesive gel in a swelling gradient: a mechanism for cell locomotion, *Phys. Rev. Lett.* 90 (2003) 168102.
- [19] W. Strychalski, *Simulation Methods for Spatiotemporal Models of Biochemical Signaling Networks*, Ph.D. Thesis, University of North Carolina, Chapel Hill, 2009.
- [20] D. Bottino, A. Mogilner, T. Roberts, M. Stewart, G. Oster, How nematode sperm crawl, *J. Cell Sci.* 115 (2002) 367–384.
- [21] M. Herant, W.A. Marganski, M. Dembo, The mechanics of neutrophils: synthetic modeling of three experiments, *Biophys. J.* 84 (2003) 3389–3413.
- [22] B. Rubinstein, K. Jacobson, A. Mogilner, Multiscale two-dimensional modeling of a motile simple-shaped cell, *Multiscale Model. Simul.* 3 (2005) 413–439.
- [23] A.F.M. Maree, A. Jilkine, A. Dawes, V.A. Grieneisen, L. Edelstein-Keshet, Polarization and movement of keratocytes: a multiscale modeling approach, *Bull. Math. Biol.* 68 (2006) 1169–1211.
- [24] C.S. Peskin, The immersed boundary method, *Acta Numerica* 11 (2002) 479–517.
- [25] S. Osher, J. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* 79 (1988) 12–49.
- [26] J.A. Sethian, Tracking interfaces with level sets, *Am. Sci.* 85 (1997).
- [27] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science*, Cambridge University Press, New York, NY, 1999.
- [28] S. Osher, R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, New York, NY, 2000.
- [29] D. Adalsteinsson, J.A. Sethian, The fast construction of extension velocities in level set methods, *J. Comput. Phys.* 148 (1999) 2–22.
- [30] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), *J. Comput. Phys.* 152 (1999) 457–492.
- [31] P. Macklin, J. Lowengrub, Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth, *J. Comput. Phys.* 203 (2005) 191–220.
- [32] T.J. Chung, *Computational Fluid Dynamics*, Cambridge University Press, Cambridge, UK, 2002.
- [33] P.J. Zwart, G.D. Raithby, ad M.J. Raw, The integrated space-time finite volume method and its application to moving boundary problems, *J. Comput. Phys.* 154 (1999) 497–519.
- [34] T. Oosterom, A. Oosterom, G. Huiskamp, Interpolation on a triangulated 3D surface, *J. Comput. Phys.* 80 (1989) 331–343.
- [35] K. Keren, P.T. Yam, A. Kinkhabwala, A. Mogilner, J.A. Theriot, Intracellular fluid flow in rapidly moving cells, *Nat. Cell Biol.* 11 (2009) 1219–1224.
- [36] I.L. Novak, F. Gao, Y. Choi, D. Resasco, J.C. Schaff, B.M. Slepchenko, Diffusion on a curved surface coupled to diffusion in the volume: application to cell biology, *J. Comput. Phys.* 226 (2007) 1271–1290.
- [37] M. Machacek, G. Danuser, Morphodynamic profiling of protrusion phenotypes, *Biophys. J.* 90 (2006) 1439–1452.
- [38] P. Schwartz, D. Adalsteinsson, P. Colella, A.P. Arkin, M. Osum, Numerical computation of diffusion on a surface, *Proc. Nat. Acad. Sci. USA* 102 (2005) 11151–11156.
- [39] L. Yang, J.C. Effler, B.L. Kutscher, S.E. Sullivan, D.N. Robinson, P.A. Iglesias, Modeling cellular deformations using the level set formalism, *BMC Syst. Biol.* 2 (2008) 95.
- [40] S. Astanin, L. Preziosi, Mathematical modeling of the Warburg effect in tumour cords, *J. Theor. Biol.* 258 (2009) 578–590.
- [41] J. Thompson, *Numerical Grid Generation: Foundations and Applications*, Elsevier, New York, NY, 1985.

- [42] H. Ji, F.-S. Lien, E. Yee, An efficient second-order accurate cut-cell method for solving the variable coefficient Poisson equation with jump conditions on irregular domains, *Int. J. Numer. Meth. Fluids* 52 (2006) 723–748.
- [43] H.Z. Tang, T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, *SIAM J. Numer. Anal.* 41 (2003) 487–515.
- [44] D. Adalsteinsson, J.A. Sethian, Transport and diffusion of material quantities on propagating interfaces via level set methods, *J. Comput. Phys.* 185 (2003) 271–288.
- [45] H. Johansen, P. Colella, A cartesian grid embedded boundary method for Poisson's equation on irregular domains, *J. Comput. Phys.* 147 (1998) 60–85.
- [46] T.M. Roberts, M. Stewart, Nematode sperm amoeboid movement without actin, *Trends Cell Biol.* 7 (1997) 368–373.
- [47] C.W. Wolgemuth, L. Miao, O. Vanderlinde, T. Roberts, G. Oster, MSP dynamics drives nematode sperm locomotion, *Biophys. J.* 88 (2005) 2462–2471.
- [48] M. Goegler, J.A. Käs, A. Ehrlicher, D. Kock, C.W. Wolgemuth, C. Brunner, The origin and spatial distribution of forces in motile cells, *PLoS Biology*, submitted for publication.
- [49] H.P. Grimm, A. Verkhovskiy, A. Mogilner, J.-J. Meister, Analysis of actin dynamics at the leading edge of crawling cells: implications for the shape of keratocyte lamellipodia, *Eur. Biophys. J.* 32 (2003) 563–577.
- [50] K. Keren, Z. Pincus, G.M. Allen, E.L. Barnhart, G. Marriotti, A. Mogilner, J.A. Theriot, Mechanism of shape-determination in motile cells, *Nature* 453 (2008) 475–480.
- [51] C.W. Wolgemuth, Lamellipodial contractions during crawling and spreading, *Biophys. J.* 89 (2005) 1643–1649.
- [52] T.M. Svitkina, A.B. Verkhovskiy, K.M. McQuade, G.G. Borisy, Analysis of actin–myosin II system in fish epidermal keratocytes: mechanism of cell body translocation, *J. Cell Biol.* 139 (1997) 397–415.
- [53] S.H. Zigmond, Ability of polymorphonuclear leukocytes to orient in gradients of chemotactic factors, *J. Cell Biol.* 75 (1977) 606–616.
- [54] L. Song, S.M. Nadkarni, H.U. Bödeker, C. Beta, A. Bae, C. Franck, W.-J. Rappel, W.F. Loomis, E. Bodenschatz, *Dictyostelium discoideum* chemotaxis: threshold for directed motion, *Eur. J. Cell Biol.* 85 (2006) 981–989.
- [55] H. Levine, D.A. Kessler, W.-J. Rappel, Directional sensing in eukaryotic chemotaxis: a balanced inactivation model, *Proc. Nat. Acad. Sci. USA* 103 (2006) 9761–9766.
- [56] C.I. Lacayo, Z. Pincus, M.M. VanDuijn, C.A. Wilson, D.A. Fletcher, F.B. Gertler, A. Mogilner, J.A. Theriot, Emergence of large-scale cell morphology and movement from local actin filament growth dynamics, *Plos Biol.* 5 (2007) e233.
- [57] T.J. Ligocki, P.O. Schwartz, J. Percelay, P. Colella, Embedded boundary grid generation using the divergence theorem, implicit functions, and constructive solid geometry, *J. Phys. Conf. Ser.* 125 (2008) 012080.
- [58] P. Macklin, J. Lowengrub, A new ghost cell level set method for moving boundary problems: application to tumor growth, *J. Sci. Comput.* 35 (2008) 266–299.
- [59] P. McCorquodale, P. Colella, H. Johansen, A Cartesian grid embedded boundary method for the heat equation on irregular domains, *J. Comput. Phys.* 173 (2001) 620–635.
- [60] W. Strychalski, D. Adalsteinsson, T.C. Elston, A cut cell method for simulating spatial models of biochemical reaction networks in arbitrary geometries, *Commun. Appl. Math. Comput. Sci.* 5 (2010) 31–53.