Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 225 (2007) 2320–2332
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

An efficient, scalable numerical algorithm for the simulation of electrochemical systems on irregular domains

Matthew Buoni [a,*], Linda Petzold [b]

[a] *Mechanical Engineering Department, University of California, Santa Barbara, 6750 El Colegio Road, Apt. 420, Goleta, CA 93106, United States*

[b] *Mechanical Engineering Department and Computer Science Department, University of California, Santa Barbara, CA 93106, United States*

Received 27 July 2006; received in revised form 12 February 2007; accepted 22 March 2007
Available online 7 April 2007

Abstract

We present a projection method for the solution of the diffusive transport and reaction equations of electrochemical systems on irregular time-dependent domains. Specific applications include electrodeposition of copper in sub-micron trenches, as well as any other electrochemical system with an arbitrarily shaped bulk region of dilute electrolyte solution. Our method uses a finite volume spatial discretization that is second-order accurate throughout, including a nonuniform region used as a transition to the far-field chemical concentrations. Time integration is performed with a splitting technique that includes a projection step to solve for the electric potential. The resulting method is first-order accurate in time, and is observed to be stable for relatively large time steps. Furthermore, the algorithm complexity scales very respectably with grid refinement and is naturally parallelizable.

PACS: 82.47.a

Keywords: Electrochemical systems; Irregular domain; Splitting method; Projection method

1. Introduction

1.1. Overview

The purpose of this paper is to present a novel methodology for solving the governing equations of electrochemistry under conditions of dilute electrolyte solution. Such systems, with irregular and moving boundaries, are of interest in copper electrodeposition and play an important role in the fabrication of interconnects for the next generation of computer processors [1] (see Fig. 1).

Our interest is to simulate copper infill of sub-micron scale trenches. This problem is inherently multiscale because the chemical reactions at the copper surface represent a length scale of nanometers (surface

* Corresponding author.
E-mail address: mjbuoni@gmail.com (M. Buoni).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2007.03.025

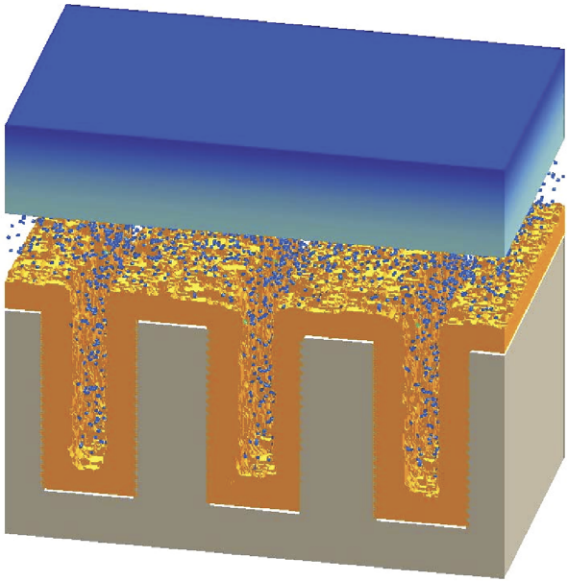

Fig. 1. Schematic of a multiscale simulation of the electrochemical process for manufacturing on-chip copper interconnects. The dots represent Cu^{2+} ions in solution, with the film on the surface being metallic copper.

roughness) and a timescale of nanoseconds to microseconds, while the diffusion and migration processes in the electrolyte solution occur at the micrometer to millimeter length scale and millisecond to seconds time scale [2]. A hybrid simulation methodology was proposed and implemented in [3] and was used to study trench infill. This approach consisted of two codes linked together: a finite difference code in the electrolyte region and a kinetic Monte-Carlo code at the copper surface. Subsequent refinements of this method have been made including the development of finite volume spatial discretization to address unphysical numerical errors (negative chemical concentrations) [6] and control systems analysis of code linkage to minimize instabilities and improve accuracy [4].

Despite the progress that has been made, these simulation methods still suffer from high computational cost. Two-dimensional simulations of modest resolution (100×100) take days to perform, scale poorly with grid refinement and are not readily parallelizable [3]. Specifically, the simulation in the electrolyte region has been a serious bottleneck. In this paper, we will present a numerical method which takes advantage of the structure of the problem to achieve a considerable gain in efficiency.

This paper is organized as follows. In Section 2, we describe the governing equations for the electrolyte region, and discuss the existing numerical approaches and their observed shortcomings. In Section 3, we derive our numerical method directly from the governing equations. This is done in two parts: first the spatial discretization is derived by integrating the governing equations over grid-sized cells; second the temporal discretization is derived by splitting the total time derivative into groups of physically related terms, and applying the Implicit Euler method to two terms and a projection method to the third term. Section 4 briefly addresses issues involved in the implementation. We assess the performance of our method in Section 5 by studying three sample problems. The order of accuracy is confirmed and a point is made about the spatial and temporal refinement required to achieve a given accuracy of the numerical solution. Also, we measure the computational complexity of our method for these three problems, and find that it scales very well as the grid is refined. We conclude the paper by summarizing our findings and highlighting areas of possible future work.

2. Governing equations

The governing equations are stiff nonlinear partial differential equations with algebraic constraints [10]. These equations describe the time evolution of the concentrations of each chemical species, c_k. They are derived by conservation of mass with chemical reactions, diffusion and migration due to electric fields,

$$\frac{\partial c_k}{\partial t} = R_k(\{c'_k\}) - \vec{\nabla}\cdot\vec{N}_k, \tag{1a}$$

$$\sum_k z_k c_k = 0, \tag{1b}$$

where R_k is the net rate of production of chemical species k due to chemical reactions, and is a function of all the other chemical species concentrations, $\{c'_k\}$. And $\vec{N}_k$ is the flux of chemical species k due to diffusion and migration.

The detailed form of R_k is given by considering N_{rxns} elementary reactions, where reaction j is given by

$$\sum_{k'} a^{\mathrm{LHS}}_{j,k'}[c_{k'}] \underset{k_j^B}{\overset{k_j^F}{\rightleftharpoons}} \sum_{k'} a^{\mathrm{RHS}}_{j,k'}[c_{k'}]. \tag{2}$$

Then the net rate of production of species k due to reaction j satisfies

$$R_k^{(j)} = (a^{\mathrm{LHS}}_{j,k} - a^{\mathrm{RHS}}_{j,k})\left(-k_j^F \prod_{k'} c_{k'}^{a^{\mathrm{LHS}}_{j,k'}} + k_j^B \prod_{k'} c_{k'}^{a^{\mathrm{RHS}}_{j,k'}}\right). \tag{3}$$

Summing over all chemical reactions, the total rate of production of chemical species k is

$$R_k(\{c_{k'}\}) = \sum_{j=1}^{N_{rxns}} R_k^{(j)}. \tag{4}$$

The detailed form of $\vec{N}_k$ is given by

$$\vec{N}_k = -D_k\vec{\nabla}c_k - z_k u_k F c_k \vec{\nabla}\Phi, \tag{5}$$

where Φ is the electric potential, D_k is the diffusion coefficient for species k, z_k is the charge of species k, u_k is the mobility constant for species k, and F is Faraday's constant. The algebraic constraint (Eq. (1b)) enforces zero net charge density for the electrolyte solution.

Substituting Eq. (5) into Eq. (1a) yields

$$\frac{\partial c_k}{\partial t} = R_k(\{c_{k'}\}) + D_k\nabla^2 c_k + (z_k u_k F)\vec{\nabla}\cdot(c_k\vec{\nabla}\Phi), \tag{6}$$

which, together with the electroneutrality constraint (Eq. (1b)), defines our system.

Convective transport may also be included by coupling these equations to the Navier–Stokes equations [9], but this is often neglected for systems with dimensions below 1 μm. For these systems, which will be our focus here, diffusion dominates because the Peclet number is small (see Fig. 2).

The boundary conditions are application dependent, so we focus now on our present application: electrodeposition. In electrodeposition, there is an *active* copper boundary where chemical reactions occur on the surface, creating a flux of each chemical species into the electrolyte solution,

$$-\vec{N}_k\cdot\hat{n} = J_k, \tag{7}$$

where $\hat{n}$ is the outward normal direction along the active boundary. J_k is calculated from a separate surface reaction model using the Kinetic Monte-Carlo (KMC) method, and is a function of the surface concentrations of the chemical species, $\{c_{k'}\}|_{\mathrm{surface}}$. For details of the KMC surface reaction model and its linkage with continuum simulations in the electrolyte region, see [3,4]. For our purposes, we regard J_k as a changing but known quantity at any given time.

At the upper boundary there is a far-field set of fixed values for each of the chemical species concentrations and the electric potential, represented by Dirichlet boundary conditions,

$$c_k = c_k^{FF}, \tag{8}$$

$$\Phi = \Phi^{FF}. \tag{9}$$

On the sides of the physical domain one can assume either zero flux or periodic boundary conditions, depending on the shape of the active copper boundary. In this paper we will assume without loss of generality a non-periodic trench-shaped active copper boundary with zero flux boundary condition on the sides,

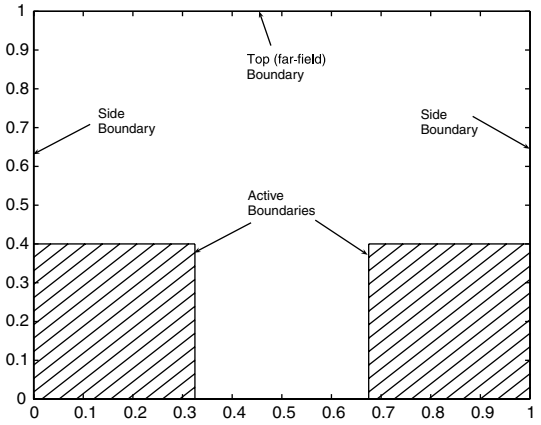

Fig. 2. Boundary conditions shown in diagram above.

$$\vec{N}_k \cdot \hat{n} = 0. \tag{10}$$

The active boundary moves due to the deposition of copper resulting from surface reactions, and is tracked implicitly by a level set method. In the level set method, a signed distance function, ϕ, is defined in the vicinity of the active boundary. The $\phi = 0$ contour implicitly defines the active boundary and is tracked by solving the advection equation

$$\frac{\partial \phi}{\partial t} = -\vec{v} \cdot \vec{\nabla}\phi = -v_n, \tag{11}$$

where

$$v_n = \frac{J_{\mathrm{Cu}}}{\rho_{\mathrm{Cu}}} \tag{12}$$

is the velocity of the active boundary, computed by the flux of copper divided by the density of copper. This velocity is directed normal to the interface and is extended along lines parallel to the $\vec{\nabla}\phi$ using the closest point fast marching method. Details on implementation of the level set and fast marching methods may be found elsewhere [13–16].

As mentioned above, attempts have been made to solve these equations by Drews, Li, Braatz, Alkire [5,6]. Their approach has been to use the method of lines (MOL) to transform the PDE system into an ODE system with algebraic constraints, i.e. differential algebraic equations (DAEs). The resulting index-2 DAE system was then solved using DASPK3 [7,8]. Although this strategy works, it was not very computationally efficient due to difficulties with finding an effective preconditioner, which causes the code to run slowly even for modest size spatial grids (100×100) and a few chemical species. Much effort has been put into trying to design better preconditioners for use with DASPK in order to improve efficiency, but without much success [6]. The problem becomes more severe as one refines the grid; code profiling reveals that the computation time scales as $(N_{eqns})^p$, where N_{eqns} is the total number of grid variables and $p \approx 2$, making highly resolved simulations computationally infeasible.

The goal of this work has been to develop a computational algorithm for the solution of Eqs. (1a) and (1b) on irregular domains with a moving active boundary that is efficient, scales well with grid refinement, and is easy to parallelize. In the remainder of this paper, we will describe our algorithm in detail, apply it to sample problems and measure its accuracy and efficiency, thus demonstrating that we have achieved this goal.

3. Numerical solution

The governing equations of Section 2 are solved numerically by discretizing the spatial and temporal domains. Details of the discretization are provided in [20]. Here, we briefly describe the important points.

3.1. Spatial discretization

The spatial domain is divided into cells of finite volume (FV). All spatial derivatives in Eq. (6) are computed with $O(\Delta x^2)$ accuracy, including boundary cells.

In our algorithm, we define two regions of cells: uniform and nonuniform. The uniform cells form the lower region of the computational domain, and include the trench and active boundary, since all such points require roughly the same resolution. The nonuniform cells above the trench serve as a transition zone from the trench region to the far-field. Since this transition length is often much greater than the trench length scale, a uniform grid here would add unnecessary computational expense. As a result, there are three distinct types of FV cells, as shown in Fig. 3: uniform region cells, boundary cells and nonuniform region cells.

To obtain the finite volume equations, we begin in the standard way by integrating the governing PDE, Eqs. (1a) and (1b), over a small cell in our domain. After applying the divergence theorem and approximating boundary integrals by products of average values (at boundary midpoints) multiplied by boundary length and area integrals by average values (at cell centroids) multiplied by area, we obtain an equation of the form:

$$V_{i,j}^{\mathrm{rel}} \frac{\partial \overline{(c_k)}_{i,j}}{\partial t} = (\mathrm{RHS})^{(rxns)} + (\mathrm{RHS})^{(\mathrm{diff})} + (\mathrm{RHS})^{(\mathrm{migration})} + (\mathrm{RHS})^{(\mathrm{boundary\ flux})}, \tag{13}$$

where $\overline{(c_k)}_{i,j}$ is the concentration of chemical species k at the centroid of cell (i,j).

$(\mathrm{RHS})^{(rxns)}$, $(\mathrm{RHS})^{(\mathrm{diff})}$, $(\mathrm{RHS})^{(\mathrm{migration})}$ and $(\mathrm{RHS})^{(\mathrm{boundary\ flux})}$ are the cell-integrated reaction, diffusion, migration and boundary flux terms, given by

$$(\mathrm{RHS})^{(rxns)} = V_{i,j}^{\mathrm{rel}} R_k(\{\overline{(c_{k'})}_{i,j}\}), \tag{14}$$

$$(\mathrm{RHS})^{(\mathrm{diff})} = \frac{D_k}{\Delta x_i}\left(\left.\frac{\partial c_k}{\partial x}\right|_{\mathrm{right}} \theta_{\mathrm{right}} - \left.\frac{\partial c_k}{\partial x}\right|_{\mathrm{left}} \theta_{\mathrm{left}}\right) + \frac{D_k}{\Delta y_j}\left(\left.\frac{\partial c_k}{\partial y}\right|_{\mathrm{up}} \theta_{\mathrm{up}} - \left.\frac{\partial c_k}{\partial y}\right|_{\mathrm{down}} \theta_{\mathrm{down}}\right), \tag{15}$$

$$\begin{aligned}(\mathrm{RHS})^{(\mathrm{migration})} = {} & \frac{(z_k u_k F)}{\Delta x_i}\left(\left(c_k \frac{\partial \Phi}{\partial x}\right)\Big|_{\mathrm{right}} \theta_{\mathrm{right}} - \left(c_k \frac{\partial \Phi}{\partial x}\right)\Big|_{\mathrm{left}} \theta_{\mathrm{left}}\right) \\ & + \frac{(z_k u_k F)}{\Delta y_j}\left(\left(c_k \frac{\partial \Phi}{\partial y}\right)\Big|_{\mathrm{up}} \theta_{\mathrm{up}} - \left(c_k \frac{\partial \Phi}{\partial y}\right)\Big|_{\mathrm{down}} \theta_{\mathrm{down}}\right),\end{aligned} \tag{16}$$

$$(\mathrm{RHS})^{(\mathrm{boundary\ flux})} = \frac{\Delta s_{i,j}}{\Delta x_i \Delta y_j}\left(D_k \left.\frac{\partial c_k}{\partial n}\right|_{\mathrm{active}} + (z_k u_k F)\left(c_k \frac{\partial \Phi}{\partial n}\right)\Big|_{\mathrm{active}}\right), \tag{17}$$

where $\Delta x_i \times \Delta y_j$ are the dimensions of the FV cell, $V_{i,j}^{\mathrm{rel}}$ is the volume fraction of the cell in solution, Δs is the active boundary length and $\theta_{\mathrm{up}}, \theta_{\mathrm{down}}, \theta_{\mathrm{left}}, \theta_{\mathrm{right}}$ are the fractions of the cell faces in solution (See Fig. 4).

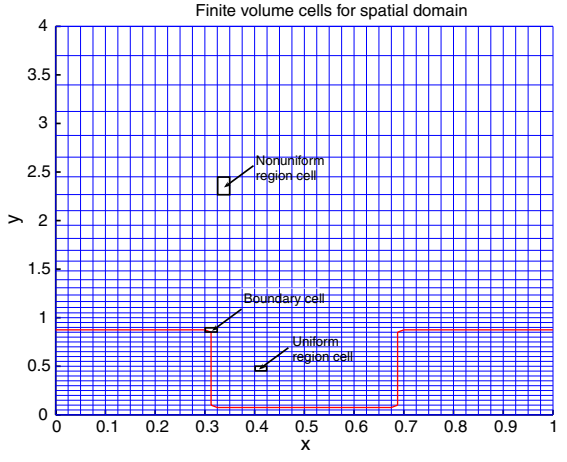

Fig. 3. Three types of finite volume cells shown in figure above.

Finally, we note that similar spatial discretizations have been used to solve the heat and Poisson equations on irregular domains with moving boundaries [11,12].

3.2. Temporal discretization

Temporal discretization is accomplished via a splitting technique that uses the Backward (implicit) Euler method combined with a projection step. We split the right-hand side of Eq. (13) into three sets of terms: (1) reaction terms, (2) diffusion terms (plus boundary flux terms), and (3) migration terms, as indicated by the superscript used. To advance the concentration fields, $(c_k)_{i,j}$, from time t_n to $t_{n+1} = t_n + \Delta t$, two intermediate values, $(c_k)_{i,j}^{(*,rxns)}$ and $(c_k)_{i,j}^{(*,\mathrm{diff})}$, are calculated. Schematically, we do the following:

$$(c_k)_{i,j}^{(n)} \overset{\text{reaction}}{\rightarrow} (c_k)_{i,j}^{(*,rxns)} \overset{\text{diffusion}}{\rightarrow} (c_k)_{i,j}^{(*,\mathrm{diff})} \overset{\text{projection}}{\rightarrow} \Phi_{i,j} \overset{\text{migration}}{\rightarrow} (c_k)_{i,j}^{(n+1)}. \tag{18}$$

By *projection*, what is meant is that $\Phi_{i,j}$ is computed such that after migration, the charge neutrality constraint is satisfied at every solution-containing cell center.

Starting from Eq. (13), with the left-hand side discretized in time, the algorithm proceeds as follows:

(1) *Reaction terms*

$$V_{i,j}^{\mathrm{rel}} \frac{(\overline{(c_k)}_{i,j}^{(*,rxns)} - \overline{(c_k)}_{i,j}^{(n)})}{\Delta t} = (\mathrm{RHS})^{(*,rxns)}. \tag{19}$$

(2) *Diffusion terms* (plus boundary flux)

$$V_{i,j}^{\mathrm{rel}} \frac{(\overline{(c_k)}_{i,j}^{(*,\mathrm{diff})} - \overline{(c_k)}_{i,j}^{(*,rxns)})}{\Delta t} = (\mathrm{RHS})^{(*,\mathrm{diff})} + (\mathrm{RHS})^{(\text{boundary flux})}. \tag{20}$$

(3) *Projection step*

$$\sum_k z_k V_{i,j}^{\mathrm{rel}} \frac{(\overline{(c_k)}_{i,j}^{(n+1)} - \overline{(c_k)}_{i,j}^{(*,\mathrm{diff})})}{\Delta t} = \sum_k z_k (\mathrm{RHS})^{(\text{migration})}. \tag{21}$$

Eq. (21), together with the charge neutrality condition, $\sum_k z_k \overline{(c_k)}_{i,j}^{(n+1)} = 0$, leads to an implicit Poisson-like equation for the electric potential, $\Phi_{i,j}$ (contained in $(\mathrm{RHS})^{(\text{migration})}$):

$$\Delta t \sum_k z_k (\mathrm{RHS})^{(\text{migration})} = -V_{i,j}^{\mathrm{rel}} \sum_k z_k \overline{(c_k)}_{i,j}^{(*,\mathrm{diff})}. \tag{22}$$

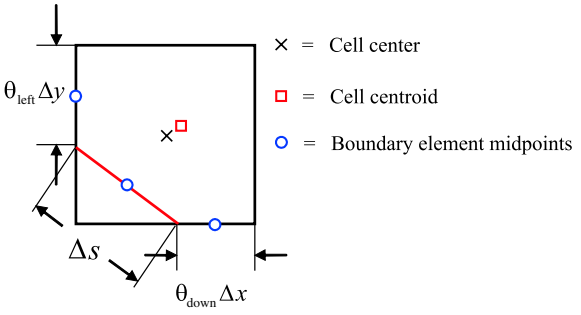

Fig. 4. Location of cell centroid and boundary element midpoints for a boundary cell.

(4) *Migration terms (using $\Phi_{i,j}$ obtained in step 3)*

$$V_{i,j}^{\mathrm{rel}} \frac{\left(\overline{(c_k)}_{i,j}^{(n+1)} - \overline{(c_k)}_{i,j}^{(*,\mathrm{diff})}\right)}{\Delta t} = (\mathrm{RHS})^{(\mathrm{migration})}. \tag{23}$$

The resulting concentrations, $(c_k)_{i,j}^{(n+1)}$, are $O(\Delta t)$ accurate and satisfy the charge neutrality condition, $\sum_k z_k (c_k)_{i,j}^{(n+1)} = 0$, to machine precision. To see that this method is convergent with $O(\Delta t)$ accuracy, we need to verify that the discretization is $O(\Delta t)$-consistent and 0-stable [7]. We begin by writing the FV equations as a DAE system,

$$\frac{\mathrm{d}c}{\mathrm{d}t} = R(c) + D(c) + G^{\mathrm{T}}(c)\lambda, \tag{24a}$$

$$Ac = 0. \tag{24b}$$

Using the algebraic constraint (Eq. (24b)), we solve for λ to obtain the underlying ODE system,

$$\frac{\mathrm{d}c}{\mathrm{d}t} = R(c) + D(c) - G^{\mathrm{T}}(c)(AG^{\mathrm{T}}(c))^{-1}A(R(c) + D(c)). \tag{25}$$

Now, consider our numerical method. We have

$$\frac{c^{(*)} - c^{(n)}}{\Delta t} = R(c^{(*)}), \tag{26}$$

$$\frac{c^{(**)} - c^{(*)}}{\Delta t} = D(c^{(**)}), \tag{27}$$

$$\frac{c^{(n+1)} - c^{(**)}}{\Delta t} = G^{\mathrm{T}}(c^{(**)})\lambda, \tag{28}$$

where λ is computed by pre-multiplying Eq. (28) by A and solving for λ to obtain

$$\lambda = -\frac{(AG^{\mathrm{T}}(c^{(**)}))^{-1}Ac^{(**)}}{\Delta t}, \tag{29}$$

which may be expressed as (using Eqs. (26) and (27) and $Ac^{(n)} = 0$)

$$\lambda = -(AG^{\mathrm{T}}(c^{(**)}))^{-1}A(R(c^{(*)}) + D(c^{(**)})). \tag{30}$$

Finally, substituting Eq. (30) into Eq. (28) and summing Eqs. (26)–(28) gives

$$\frac{c^{(n+1)} - c^{(n)}}{\Delta t} = R\left(c^{(*)}\right) + D\left(c^{(**)}\right) - G^{\mathrm{T}}\left(c^{(**)}\right)\left(AG^{\mathrm{T}}\left(c^{(**)}\right)\right)^{-1}A\left(R\left(c^{(*)}\right) + D\left(c^{(**)}\right)\right). \tag{31}$$

Since $c^{(*)} = c^{(n)} + O(\Delta t)$ and $c^{(**)} = c^{(n)} + O(\Delta t)$, Eq. (31) may be expressed as

$$\frac{c^{(n+1)} - c^{(n)}}{\Delta t} = R(c^{(n)}) + D(c^{(n)}) - G^{\mathrm{T}}(c^{(n)})(AG^{\mathrm{T}}(c^{(n)}))^{-1}A(R(c^{(n)}) + D(c^{(n)})) + O(\Delta t). \tag{32}$$

Comparing Eqs. (32) and (25), we see that our method is $O(\Delta t)$-consistent. To see 0-stability, start with Eq. (32) and simply refer to the proof for the forward Euler method as given in [7].

4. Notes on implementation

In this section, we highlight some of the properties of the equations to be solved in our numerical method and present our implementation strategies. Consider each of the four steps of the time-splitting algorithm, in turn.

(1) *Reaction terms*

For the reaction terms we obtain N_{cells} independent nonlinear systems of N_{species} equations each for $(c_k)_{i,j}^{(*,rxns)}$. These systems are solved by Newton's method with an LU-decomposition of the Jacobian

matrix (computed analytically), which is saved from adjacent FV cells and is updated only when the method fails to converge after a predefined number of iterations. Initial guesses for the solutions to these systems are taken to be the current time step solution at an already computed adjacent FV cell, if one is available, or the solution at the previous time step for the current FV cell.

(2) *Diffusion terms*
For the diffusion terms we obtain N_{species} independent linear systems of N_{cells} equations each for $(c_k)_{i,j}^{(*,\text{diff})}$. The matrices corresponding to these linear systems are symmetric for the uniform cell equations, slightly asymmetric for the nonuniform cell equations (as long as the ratio of adjacent cell sizes is close to one) and completely asymmetric for boundary cell equations. These systems are easily and efficiently solved by a general preconditioned iterative linear solver. We use BICGSTAB with ILU preconditioning, as implemented in SPARSEKIT2 [18]. Maximum efficiency is found empirically by tuning the ILU parameters, drop tolerance and fill level to 10^{-4} and $+15$, respectively.

(3) *Projection step*
The projection step requires the solution of a single linear system of N_{cells} equations for $\Phi_{i,j}$, with matrix symmetry similar in form to the matrix involving the diffusion terms.

(4) *Migration terms*
The migration terms require the electric potential, $\Phi_{i,j}$, computed in the projection step. Once obtained, the equations for the new chemical concentrations, $(c_k)_{i,j}^{(n+1)}$, are fully explicit everywhere except along the active boundary. There, we get N_{species} small independent linear systems of N_{boundary} equations each, one for each chemical species, k.

We note that steps 1, 2 and 4 are easily parallelizable, owing to the independence of the equation systems that are solved. We plan to explore this in a future paper.

5. Numerical results

5.1. Efficiency results

To test the efficiency of our algorithm, we measure CPU time over 1000 time steps for different sized grids, ranging from 20×20 to 640×640, with $\Delta t = 0.0001$ for each grid. In Fig. 5, we plot average time for one time step versus number of grid variables. The CPU time vs. problem size appears to obey a power law, which we compute as

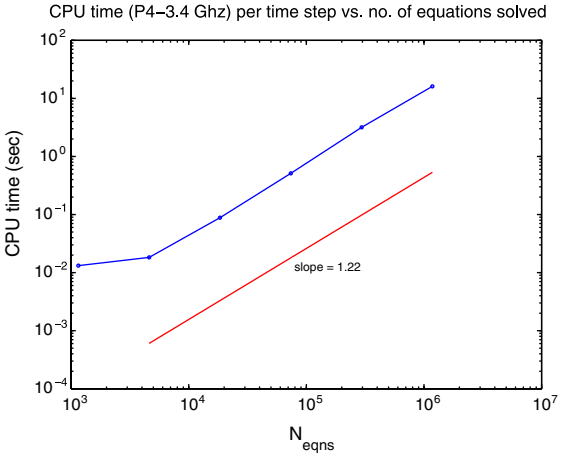

Fig. 5. The plot above shows the computationally efficiency of our algorithm. CPU time on a P4-3.4 GHz machine is given for one time step vs. total number of equations solved, corresponding to trench-shaped grids ranging from 20×20 to 640×640. The result is a power law of 1.22, where 1.0 is optimal.

$$p = \frac{\log\left(\frac{T_{640}}{T_{40}}\right)}{\log\left(\frac{640^2}{40^2}\right)}, \tag{33}$$

where T_{40} and T_{640} are the CPU times for the 40×40 and 640×640 grids, respectively. From our measurements, we computed a value of $p = 1.22$. Note that the parameter values used for this test were the same as those given in Section 5.2.

5.2. Convergence results

We perform three sets of tests to validate the accuracy of our method. In all tests, we use three chemical species with diffusion coefficients $D_1 = 1.0$, $D_2 = 10.0$, $D_3 = 100.0$, mobility constants $u_1 = 1.0$, $u_2 = 10.0$, $u_3 = 100.0$, and charge $z_1 = 1.0$, $z_2 = -2.0$, $z_3 = -1.0$. Also, in all tests, we set the top Dirichlet boundaries to $c_1^{FF} = 3.0, c_2^{FF} = 1.0, c_3^{FF} = 1.0$ and let the initial species concentrations be given by a narrowly peaked two-dimensional Gaussian distribution,

$$c_k^0(x,y) = \left(5\mathrm{e}^{\frac{((x-0.5)^2+(y-0.5)^2)}{0.1^2}} + 1\right) c_k^{FF}. \tag{34}$$

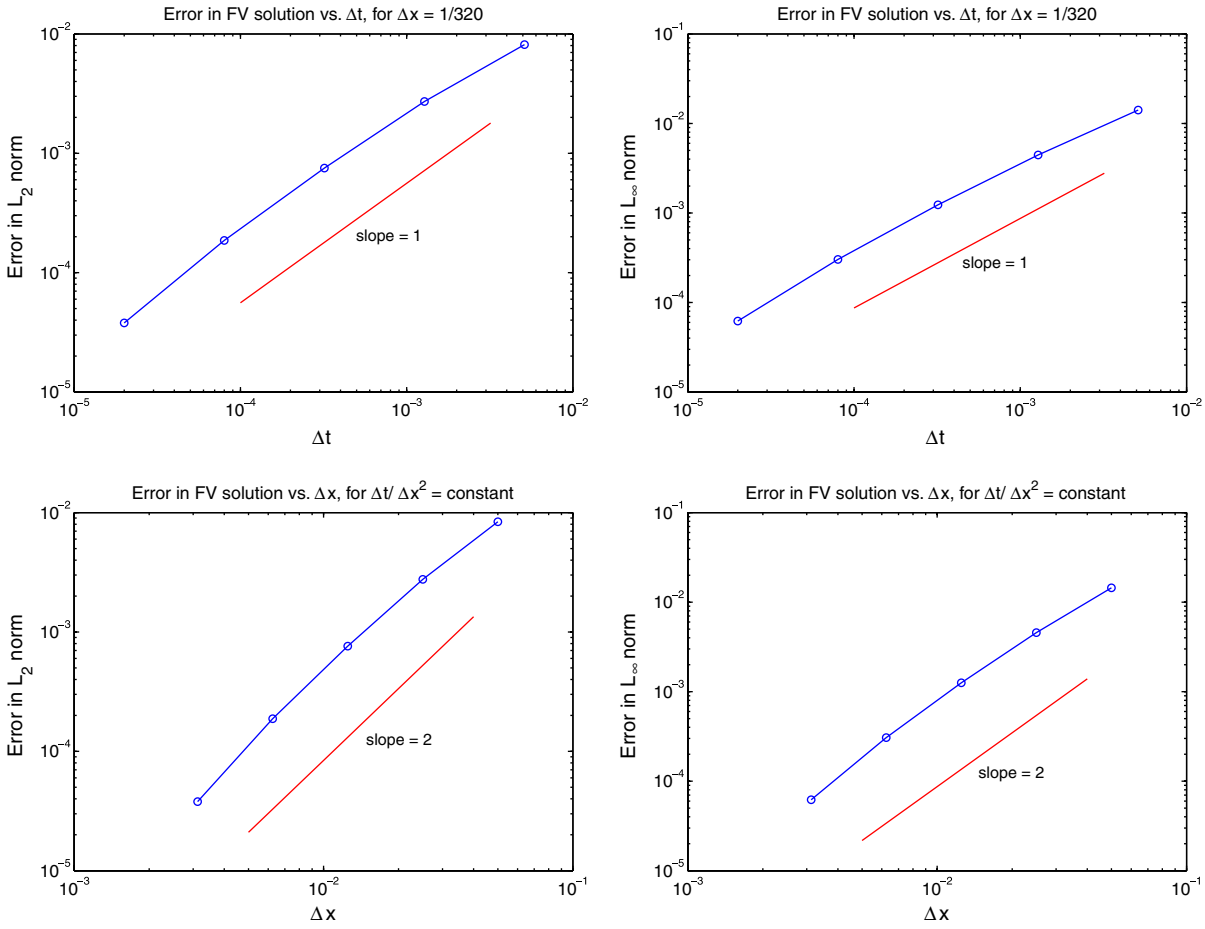

Fig. 6. Convergence (L_2-norm – left, L_∞-norm – right) of numerical method in time (top) and space (bottom) for a rectangular domain with no chemical reactions and active boundary influx, $J_k = 0$. The straight lines with slopes 1 and 2 indicate $O(\Delta t)$ and $O(\Delta x^2)$ accuracy, respectively.

The time scale, τ, for the evolution of the chemical concentration fields is determined by the largest diffusion coefficient, $\tau = \sqrt{\frac{L}{\max D_k}}$, where L is the length scale of the physical domain. Thus, we get $\tau = \sqrt{\frac{1}{100.0}} = 0.1$. Note that these parameters have been nondimensionalized and represent a range of physical values that might be used in a typical simulation. For example, diffusion coefficients for various ions in aqueous solution (Cu^{2+}, H^+, etc.) typically range from 10^{-10} to 10^{-8} m^2/s [3]. Length scales are of order 10^{-6} m and time scales vary from 10^{-3} to 10^{2} s [2]. The reason for using an initial Gaussian distribution, however, is for generating non-trivial numerical solutions for testing our algorithm.

We compute the numerical solutions from $t_0 = 0$ to $t_1 = \tau = 0.1$ for grids ranging from 20×20, 40×40, 80×80, 160×160, 320×320 to 640×640. For each grid, the time step Δt ranges from 0.00512 to 0.000005, being decreased by a factor of 4 successively. The numerical solution on the 640×640 grid with $\Delta t = 0.000005$ is taken to approximate the exact solution for purposes of error calculation.

For each of our numerical solutions, we calculate the relative error in both the L_2 and L_∞ norms as follows. First, the most refined numerical solution ($\Delta x = 1/640$, $\Delta t = 0.000005$) is averaged onto the coarser grids, giving an approximation to the exact solution on these grids, i.e.a $(c_k)_{i,j}^{(\text{exact})}$. Then the errors are computed as

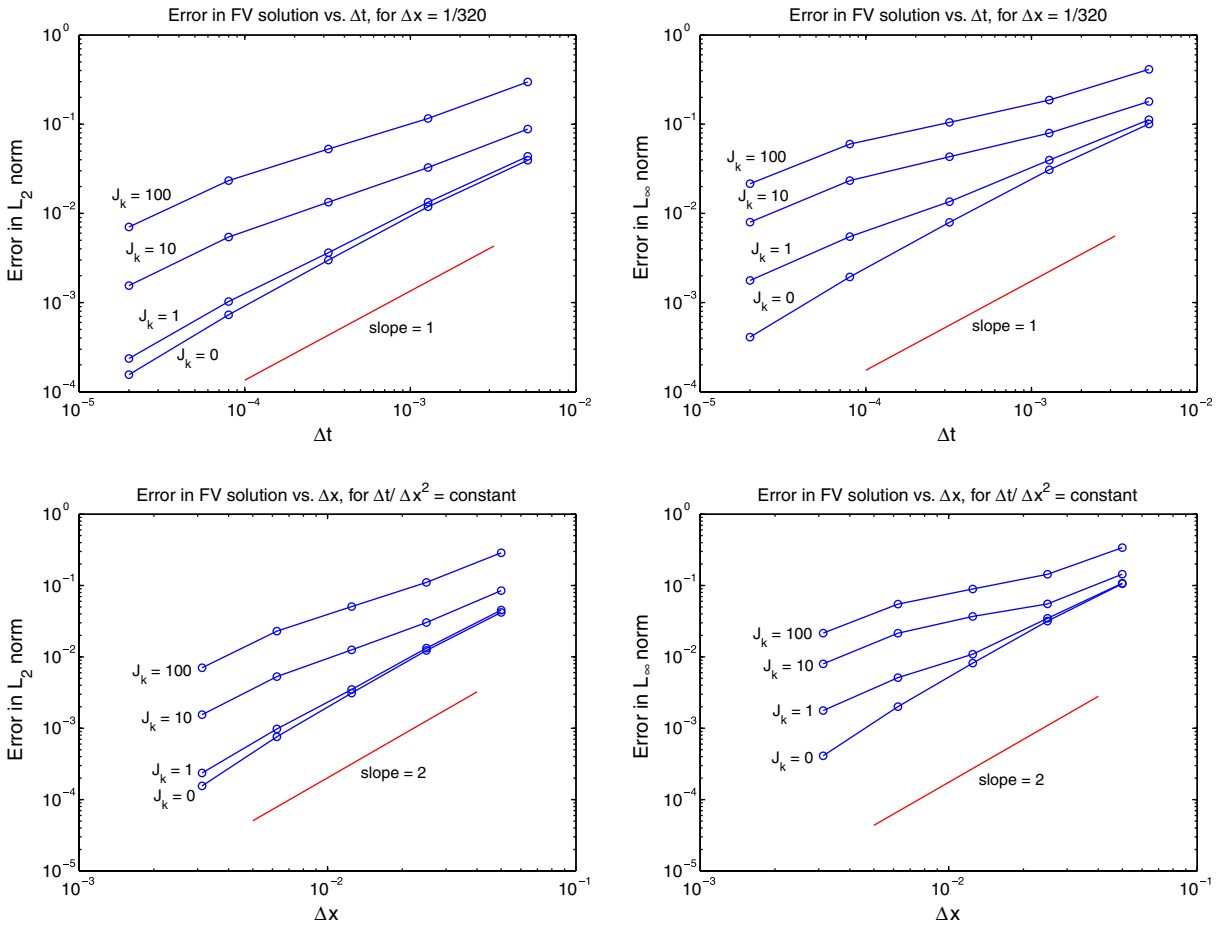

Fig. 7. Convergence (L_2-norm – left, L_∞-norm – right) of numerical method in time (top) and space (bottom) for a trench-shaped domain with one chemical reaction and active boundary influx, $J_k = 0, 1, 10, 100$. The straight lines with slopes 1 and 2 indicate $O(\Delta t)$ and $O(\Delta x^2)$ accuracy, respectively.

$$E_{L_2} = \frac{\left(\frac{\sum_{k=1}^{N_{\text{species}}}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}((c_k)_{i,j}-(c_k)_{i,j}^{(\text{exact})})^2}{N_{\text{species}}N_xN_y}\right)^{1/2}}{\left(\frac{\sum_{k=1}^{N_{\text{species}}}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}((c_k)_{i,j}^{(\text{exact})})^2}{N_{\text{species}}N_xN_y}\right)^{1/2}}, \tag{35}$$

$$E_{L_\infty} = \frac{\max_{k,i,j}|(c_k)_{i,j} - (c_k)_{i,j}^{(\text{exact})}|}{\max_{k,i,j}|(c_k)_{i,j}^{(\text{exact})}|}. \tag{36}$$

In our first test, we use a rectangular grid with no chemical reactions and no influx along the active boundary, $J_k = 0$. This test is chosen as a preliminary validation of the accuracy of our spatial discretization, projection scheme and linear solvers. Our results are plotted in Fig. 6. We find that the method is indeed $O(\Delta t, \Delta x^2)$ accurate, as expected. Also, the charge neutrality condition, Eq. (1b), is satisfied to the precision set for our linear solvers.

In our second test, we use a trench-shaped grid with one chemical reaction, $[1] + [2] \underset{30.0}{\overset{10.0}{\rightleftharpoons}} [3]$. The active boundary influx, J_k, is set to the same value for each chemical species and is increased during a series of four subtests: $J_k = 0.0$, 1.0, 10.0, 100.0. The purpose of these subtests is to understand how large values of J_k can degrade the accuracy of our numerical solutions. Our results are plotted in Fig. 7. Notice that for $J_k \gg 1.0$, our numerical solutions lose accuracy and converge slower than expected.

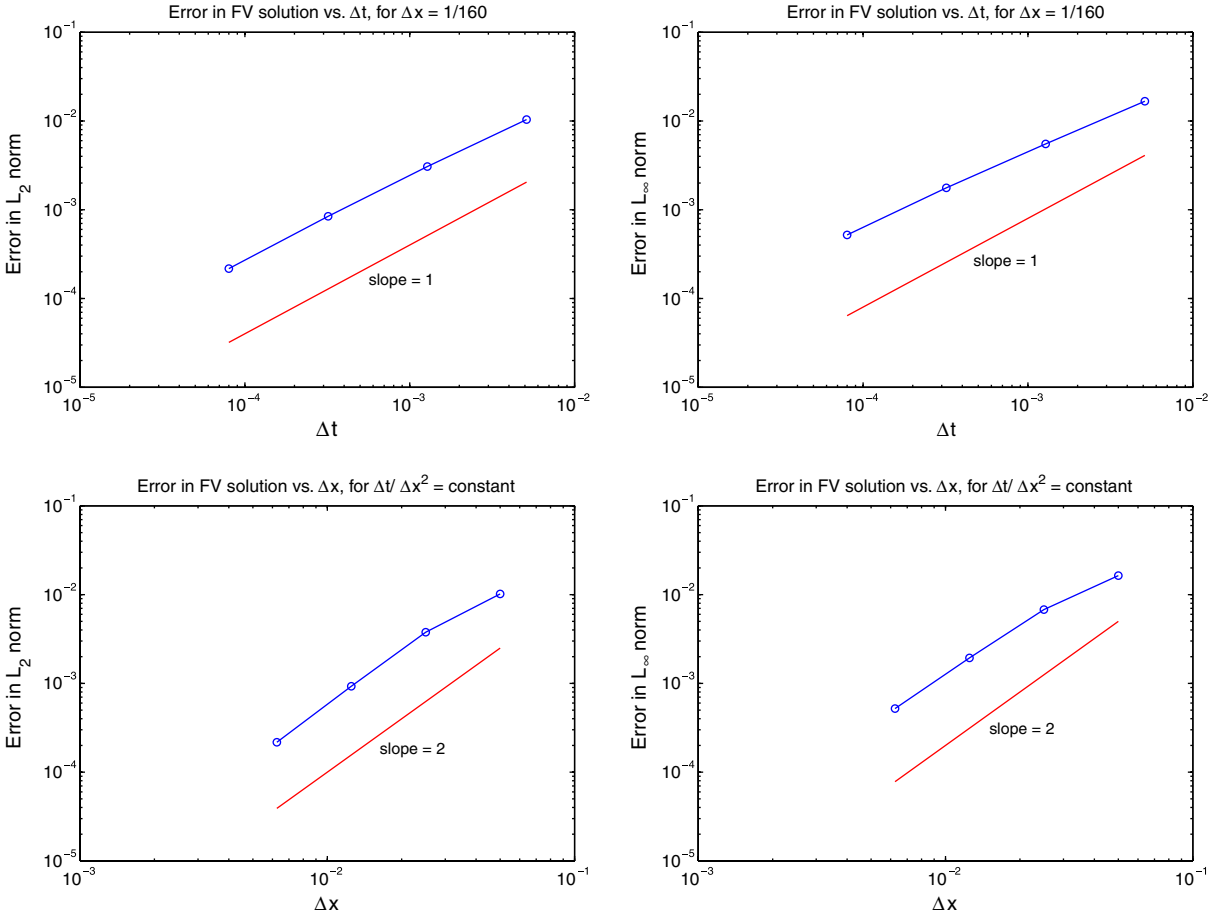

Fig. 8. Convergence (L_2-norm – left, L_∞-norm – right) of numerical method in time (top) and space (bottom) for a moving boundary domain, where the boundary is advected via the level set method. The straight lines with slopes 1 and 2 indicate $O(\Delta t)$ and $O(\Delta x^2)$ accuracy, respectively.

The conclusions we draw from this second test are as follows. First, our method retains $O(\Delta t, \Delta x^2)$ accuracy in the presence of chemical reactions and a trench-shaped grid. For moderate values of J_k ($\approx$1.0–10.0), our method approaches $O(\Delta t, \Delta x^2)$ accuracy as we refine Δt and Δx to the smallest tested values. Slower convergence is observed for larger Δt and Δx and is more pronounced in the L_∞-norm. As J_k is increased further (to 100.0), the loss of accuracy becomes more severe. An explanation for this is that the exact solution exhibits a thin boundary layer near the active boundary that becomes steeper as J_k is increased. This boundary layer is difficult to resolve even on the most refined uniform grids, thus degrading the overall accuracy. The solution to this problem is simple in principle: use a nonuniform grid near the active boundary. And since the active boundary moves, the grid would have to be refined adaptively. This is an area of possible future work.

Our third test is designed to verify the accuracy of our method with a moving boundary. For this, we couple our numerical method to a simple surface reaction model using the level set method, as described in [6]. In our implementation of the level set method, we use a closest point algorithm to prevent our solutions from degrading to $O(\Delta x)$ accuracy near the moving boundary [17]. Our results are plotted in Fig. 8. Since we moved our boundary slowly (a distance of approximately 0.2 cells per timestep for the coarsest grids) and chose parameter values yielding relatively small boundary flux ($J_k \leqslant 1$), we observe $O(\Delta t, \Delta x^2)$ accuracy, as expected.

Overall, our algorithm exhibits small relative errors for moderately refined grids and time steps in most cases. For example, we often would like to obtain a numerical solution with less than 1% relative error. From Fig. 7, we see that this is achieved with an 80×80 grid and $\Delta t = 0.00032$, which means $\approx$300 time steps to integrate from $t_0 = 0.0$ to $t_1 = 0.1$ for $J_k = 0.0$, 1.0. The same 80×80 grid will work with $\Delta t = 0.00008$ for $J_k = 10.0$ and $\Delta t = 0.00002$ for $J_k = 100.0$. Notice from the plots that for practical values of Δt and Δx, accuracy is improved most by refining Δt rather than Δx.

6. Conclusions and future work

The algorithm described here provides a general numerical strategy for solving Eqs. (1a) and (1b) on irregular domains with moving boundaries. We split the right-hand side of Eq. (1a) into three groups of physically related terms: reaction, diffusion and migration. We then integrate the chemical concentration fields corresponding to each set of terms in turn. Similar splitting techniques are commonly used to solve systems with reaction and diffusion only and have been extended to higher order accuracy [19]. However, splitting methods have not been combined with migration and the charge neutrality constraint (Eq. (1b)) to our knowledge. The advantage of our method over others is in its efficiency, scalability, and ease of parallelization. It also appears to be very stable, which is the result of using the fully Implicit Euler method to integrate the potentially stiff reaction and diffusion terms. These properties will prove most useful when extending the algorithm to three dimensions. Since these equations are common to most electrochemical systems, we believe our method will be useful in many applications.

References

[1] R.D. Braatz, R.C. Alkire, E. Seebauer, E. Rusli, R. Gunawan, T.O. Drews, X. Li, Y. He, Perspectives on the design and control of multiscale systems, Journal of Process Control 16 (2006) 193–204.
[2] T.O. Drews, J.C. Ganley, R.C. Alkire, Evolution of surface roughness during copper electrodeposition in the presence of additives: comparison of experiments and Monte-Carlo simulations, J. Electrochem. Soc. 150 (2003) C325–C334.
[3] Timothy O. Drews, Eric G. Webb, David L. Ma, Jay Alameda, Richard D. Braatz, Richard C. Alkire, Coupled mesoscale-continuum simulations of copper electrodeposition in a trench, AIChE J. 50 (1) (2004) 226–240.
[4] E. Rusli, T.O. Drews, R.D. Braatz, Control systems analysis of a multiscale simulation code for copper electrodeposition, in: Proceeding of the American Control Conference, IEEE Press, Piscataway, NJ, USA, 2004, pp. 4243–4248.
[5] Timothy Drews, Multiscale Simulations of Nanofabricated Structures: Application to Copper Electrodeposition for Electronic Devices, Ph.D. Thesis, U. Illinois Urbana – Champaign, 2004.
[6] Xiaohai Li, Simulation of Electrochemical Surface Roughness Evolution in Moving Boundary Systems, M.S. Thesis, U. Illinois Urbana – Champaign, 2004.
[7] U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential Algebraic Equations, SIAM Press, Philadelphia, PA, USA, 1998.
[8] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial-value Problems in Differential Algebraic Equations, Elsevier Science Publishing Co., Inc., New York, NY, USA, 1989.

[9] M. Georgiadou, D. Veyret, R.L. Sani, R.C. Alkire, Simulation of shape evolution during electrodeposition of copper in the presence of additive, Journal of the Electrochemical Society 148 (2001) C54–C58.
[10] John Newman, Karen E. Thomas-Alyea, Electrochemical Systems, John Wiley and Sons, Inc., Hoboken, NJ, 2004.
[11] Peter McCorquodale, Phillip Colella, Hans Johanseny, A Cartesian grid embedded boundary method for the heat equation on irregular domains, Journal of Computational Physics 173 (2001) 620–635.
[12] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, Journal of Computational Physics 147 (1998) 60–85.
[13] J.A. Sethian, Level Set Methods, Cambridge University Press, New York, 1996.
[14] J.A. Sethian, Level Set methods and Fast Marching Methods, Cambridge University Press, New York, 1999.
[15] J.A. Sethian, D. Adalsteinsson, An overview of level set methods for etching, deposition, and lithography development, IEEE Transactions on Semiconductor Manufacturing 10 (1) (1997) 167–184.
[16] S. Osher, R.P. Fedkiw, Level set methods: an overview and some recent results, Journal of Computational Physics 169 (2) (2001) 463–502.
[17] S. Mauch, A Fast Algorithm for Computing the Closest Point and Distance Transform, Technical Report Caltech, 2000.
[18] Y. Saad, SPARSKIT: A basic tool-kit for sparse matrix computations, 2005. Available from: <http://www-users.cs.umn.edu/saad/software/SPARSKIT/sparskit.html>.
[19] Stephane Descombes, Convergence of a splitting method of high order for reaction–diffusion systems, Mathematics of Computation 70 (236) (2000) 1481–1501.
[20] M. Buoni, L. Petzold, An efficient, scalable numerical algorithm for the simulation of electrochemical systems on irregular domains: full technical details, 2006. Available from: <http://www.engineering.ucsb.edu/%7Ecse/>.