Solving Partial Differential Equations on Irregular Domains with Moving Interfaces, with Applications to Superconformal Electrodeposition in Semiconductor Manufacturing

J. A. Sethian and Ying Shan
Department of Mathematics
University of California, Berkeley
Berkeley, California 94720 *

March 4, 2008

Abstract

We present a numerical algorithm for solving partial differential equations on irregular domains with moving interfaces. Instead of the typical approach of solving in a larger rectangular domain, our approach performs most calculations only in the desired domain. To do so efficiently, we have developed a one-sided multigrid method to solve the corresponding large sparse linear systems.

Our focus is on the simulation of the electrodeposition process in semiconductor manufacturing in both two and three dimensions. Our goal is to track the position of the interface between the metal and the electrolyte as the features are filled and to determine which initial configurations and physical parameters lead to superfilling.

We begin by motivating the set of equations which model the electrodeposition process. Building on existing models for superconformal electrodeposition, we develop a model which naturally arises from a conservation law form of surface additive evolution. We then introduce several numerical algorithms, including a conservative material transport level set method and our multigrid method for one-sided diffusion equations. We then analyze the accuracy of our numerical methods. Finally, we compare our result with experiment over a wide range of physical parameters.

1 Introduction

In this article, we will design a numerical algorithm to solve partial differential equations on irregular domains with moving interfaces. This approach is considerably faster than existing ones: most of the calculations are performed only in the desired domain instead of in an extended rectangular domain, aided by the use of a one-sided version of the multigrid method to solve the corresponding large sparse linear systems. Our method has been tested to give accurate numerical solutions for problems defined on domains with convoluted geometries, including thin fingers and sharp corners.

Our focus application is the simulation of the electrodeposition process. Electroplating (see [6]) is deposition process that permits filling of high-aspect ratio features without seams or voids through the process of superconformal deposition, also called superfilling. Our goal is to track the position of the interface between the metal and the electrolyte as features are filled in order to determine numerically what initial configurations lead to superfilling.

Building on existing models for superconformal electrodeposition, we develop a model which naturally arises from a conservation law form of surface additive evolution. This model allows us to perform a careful analysis of how superfilling depends on the choice of physical parameters, with close comparison to experiment.

*This work was supported in part by the Applied Mathematical Science subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract Number DE-AC03-76SF00098, and by the Computational Mathematics Program of the National Science Foundation

This work was supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11232 through the Lawrence Berkeley National Laboratory, Berkeley, CA.

In order to successfully compute the solution to this model, several new computational techniques are developed in this paper. These include:

1. A new conservative material transport level set method in two and three dimensions for interfaces that carry scalar fields as they evolve.
2. An immersed interface type method for building one-sided difference operators for complex interfaces with thin arms and fingers.
3. A multigrid method in two and three dimensions for solving one-sided diffusion equations with irregular moving interface.

The outline of this paper is as follows. Section 2 briefly describes some existing work on electrodeposition, and then describes a general overview of some related numerical methods. Section 3 describes the underlying physics of the electrodeposition process and also the determination of some of the parameters used in our model. The basic equations that need to be solved for modeling the deposition process are derived based on previously existing models. However modifications are made so that the model is physically more reasonable.

Section 4 presents the finite difference methods for level set equations, conservation laws, and diffusion equations. In this section, most of the discretizations are done in two space dimensions. We discuss in Section 5 how to solve large linear systems in an efficient way and details of the one-sided multigrid method. The last section is devoted to numerical results, conclusions, extension to the three-dimensional case, and suggestions for future work using the methods described in this article.

2 Background Material

2.1 Target Application: Electrodeposition

Electrodeposited copper can be used as the material for on-chip trenches and vias. The process of copper electrodeposition (see Figure 1) depends on the use of additives that affect the local deposition rate and this leads to superconformal filling of trenches.

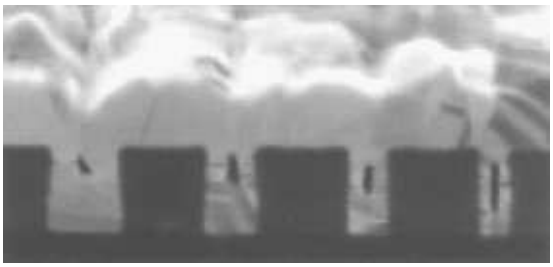

Figure 1: An image of copper deposited from electrolyte. Voids are apparent in the trenches. The picture is taken from [33].

Early modeling studies focused on leveling theory [48], in which the growth rate is dependent on the accumulation of inhibiting species onto the metal surface. Such leveling methods are not very successful in explaining the superfilling phenomena.

Subsequently, curvature-enhanced accelerator coverage (CEAC) has been proposed as the mechanism behind this process. According to the CEAC mechanism, deposition on a non-planar surface is accompanied by changes in the local surface area which affect the local adsorbate surface coverage. The coverage increases on concave segments and decreases on convex segments. This leads to bottom-up filling of features since the deposition rate is proportional to the catalyst coverage. Let κ be the local curvature. In [49],

$$\frac{\partial \theta}{\partial t} = \kappa v \theta + \text{source term} \tag{1}$$

is taken as the equation satisfied by the accelerator coverage, while in [50],

$$\frac{\partial \theta}{\partial t} + v\nabla \cdot (\boldsymbol{n}\theta) = \text{source term} \tag{2}$$

is solved instead, which implicitly depends on the curvature since $\kappa \propto \nabla \cdot \boldsymbol{n}$.

However, the numerical results in [49] and [50] somewhat demonstrate the superfilling phenomena, they both fail to accurately predict the experimental results given in [34]. The idea of some sort of curvature dependence in the CEAC mechanism is both appealing and natural, however the exact relationship between the rate of deposition and the curvature of the interface is not clear. The approach in this paper is to build on geometric arguments: we devise a physically reasonable equation for the accelerator coverage in the form of a conservation law, which depends on the curvature in a rather implicit way. The equations are solved using fast and accurate schemes. A comparison between the numerical solutions in section 7 and the experimental results in [34] shows that among all available methods, this leads to the most accurate prediction to the experimental results.

2.2 Literature Survey of Related Work

2.2.1 The Physical Problem

A great deal of information on modeling electrodeposition can be found in [22], [23], [34] and [50], which are excellent references for both overviews and detailed descriptions of the processes related to electrodeposition. In particular, [50] surveys the CEAC mechanism and the corresponding numerical simulation of the process. The simulations in [34] and [50] do not conform very well with the results from real experiments as shown in [34], in part because CEAC may not be the correct way to model the process, in part because of numerical problems that may arise either when sharp corners form or when the side walls of the trench are come close.

2.2.2 Survey of Numerical Algorithms

From an algorithmic standpoint, a successful numerical simulation of electrodeposition requires numerical techniques to track moving interfaces, as well as schemes to solve partial differential equations on regions bounded by moving boundaries. Many different approaches have been developed to tackle these issues. These include,

1. **Structured mesh finite volume methods [25] for solving problems on irregular domains with moving interfaces.** They are derived from conservation laws applied to a discrete control volume. Finite difference operators can then be used to approximate fluxes across the control surface in such a way that the discrete evolution equation is also in the form of a conservation law. The disadvantage of this approach is that it is incapable of representing complicated geometries and thus may be inaccurate in the presence of close walls or sharp corners.

2. **Cartesian grid embedded boundary methods introduced in [21] and [32] to increase the geometric flexibility of finite volume methods.** Away from the boundaries of the computational domain, this approach uses traditional finite difference discretizations on a regular Cartesian grid. On the domain boundary, the local geometry is incorporated by intersecting the domain with each grid cell. The operator is then approximated on each irregular cell using a finite volume discretization. This method may lose accuracy at the domain boundaries.

3. **Finite difference methods on structured grids (see [44] and [46]).** These methods are very popular, and a great deal of work has been devoted to adaptive grid finite difference methods (see, for example, [36]) to deal with boundary conditions defined along irregular interfaces. However, this approach can be very expensive for time-dependent problems.

4. **Level set methods introduced by Osher and Sethian in [35], which are numerical techniques designed to track the evolution of interfaces.** These methods track the moving boundary by embedding the interface into higher dimensions, and rely in part on the theory of curve and surface evolution given in [39] and [40] and on the link between front propagation and hyperbolic conservation laws discussed in [41]. The key idea is to recast interface motion as a time-dependent Eulerian initial value partial differential equation.

5. **A class of numerical methods described in [18] and [24] for nonlinear systems of conservation laws.** They are designed to solve conservation laws with some desired properties (such as the choosing a solution satisfying the entropy condition).

6. **Immersed interface methods designed by LeVeque and Li in [26] and [27].** These are numerical methods which incorporate interface jump conditions and the given partial differential equation in a local coordinate system. Time-step restrictions can be avoided by using implicit schemes, and the resulting large linear systems can be solved with fast linear solvers such as GMRES [38] and the multigrid methods [9]. One drawback of this approach is that it is not conservative due to the rotated local coordinate system. In addition, by using the partial differential equation to cancel error terms in the finite difference stencil, generating stencil coefficients becomes problem-dependent and thus more difficult to automate. A complete reference in this topic is [29].

7. **Ghost fluid methods [16] for multimaterial interfaces problems.** These methods track multimaterial interfaces with level set functions, and use ghost cells to keep the density profile from smearing out while still keeping the scheme robust and easy to program with simple extensions to multidimensions and multilevel time integration.

8. **A methodology to model arbitrary holes and material interfaces (inclusions) without meshing the internal boundaries [45].** This numerical method couples the level set method to the extended finite-element method (X-FEM), and the finite-element approximation is enriched by additional functions through the notion of partition of unity.

9. **An approach for solving Poisson equations on irregular domains in [19], [20] and [30].** This augmented approach tracks moving boundaries with level set methods and solves Poisson equations using the fast Poisson solver based on fast Fourier transform.

We will use a combination of some of the numerical methods mentioned above, including immersed interface methods for discretizing the diffusion equations, adaptive finite difference methods for solving the discretized problems, followed by multigrid methods for solving the large linear systems resulting from our implicit schemes, level set methods for tracking the interface and making continuous extensions, and conservative schemes for solving equations in the form of conservation laws. The combination leads to a direct method and is simple to implement.

3 Model Specification

Modeling of copper deposition requires the simultaneous tracking of the copper/electrolyte interface location, the surface coverage of the additives, and the concentration profiles of different components in the electrolyte. We will use a level set method along with a velocity extension methodology to track the evolution of the interface. The evolution of the accelerator coverage is determined by an equation in the form of a conservation law with source terms which accounts for the change of the interface shape, influx from the electrolyte and consumption into the metal. Concentrations within the electrolyte satisfy diffusion equations.

Assume we have the configuration as shown in Figure 2 for a trench with width w and height h. Experiments (see [34]) show that the relationship between the velocity of normal propagation of the interface and the accelerator coverage can be expressed as:

$$\boldsymbol{v} = \frac{i(\theta)\Omega \boldsymbol{n}}{2F}, \tag{3}$$

where $i(\theta)$ is the current density which is a function of θ, $\boldsymbol{n}$ is the unit normal of the interface pointing into the electrolyte, Ω is the atomic volume of copper and F is Faraday's constant. The "2" in the denominator in Eqn. (3) comes from the cupric ion charge. The current density i is given by the Butler-Volmer equation [50], namely

$$i = i(\eta) = i_0 \frac{C_c^i}{C_c^\infty} \exp\left(-\frac{\alpha F}{RT}\eta\right), \tag{4}$$

where i_0 is the exchange current density, C_c^i and C_c^∞ are the concentrations of copper along the interface and in the far field respectively, α is the transfer coefficient determined by experiments, R is the gas constant, T is the

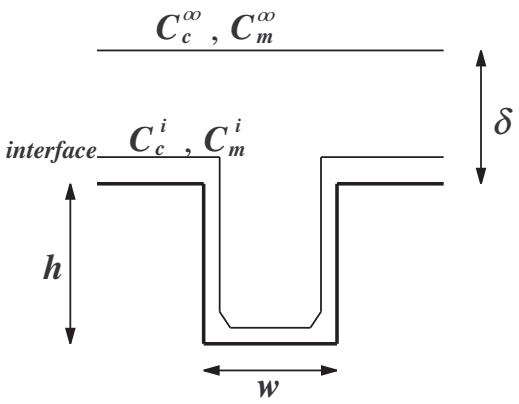

Figure 2: A silicon trench is immersed in copper-contained electrolyte, with an initial thin layer of solid copper deposited on the trench. C_c^{∞}, C_m^{∞} are constant concentrations in the far field, and C_c^i, C_m^i are concentrations along the copper/electrolyte interface, where c is for copper and m is for accelerator

temperature and η is the over-potential. Dependence of the current density on the accelerator coverage θ, that is, the $i-\theta$ relationship, is empirically established on flat surfaces to be of the form

$$i_0(\theta) = b_0 + b_1\theta, \tag{5}$$

and

$$\alpha(\theta) = m_0 + m_1\theta, \tag{6}$$

where b_0, b_1, m_0 and m_1 are constants. A combination of Eqns. (3), (4), (5) and (6) gives the final expression for the normal speed of propagation:

$$v = (b_0 + b_1\theta)\cdot\frac{C_c^i}{C_c^{\infty}}\cdot\exp\left(-\frac{(m_0+m_1\theta)F}{RT}\eta\right)\frac{\Omega}{2F}. \tag{7}$$

The concentrations of copper and accelerator in the electrolyte are governed by diffusion equations of the form

$$\frac{\partial C_\xi}{\partial t} = D_\xi \Delta C_\xi, \qquad C_\xi = C_\xi^{\infty} \text{ in the far field,} \tag{8}$$

where D_ξ is the diffusion coefficient and the subscript ξ is given by

$$\xi = \begin{cases} m & \text{for accelerator} \\ c & \text{for copper.} \end{cases}$$

The flux loss from the electrolyte at the interface defines another set of boundary conditions for Eqn. (8)

$$-D_\xi\frac{\partial C_\xi}{\partial n} = \begin{cases} -k(1-\theta)C_m^i & \text{for accelerator} \\ -v(V_c - C_c^i) & \text{for copper,} \end{cases} \tag{9}$$

where V_c is the molar volume of solid copper, and k is the jump potential that varies with the over-potential η

$$k(\eta) = k_0 - k_3\eta^3. \tag{10}$$

The rate of change of accelerator coverage θ is partly due to adsorption and consumption. The conservation of the accelerator in addition to adsorption from the electrolyte and consumption by the deposited copper gives

$$\frac{d\theta}{dt} + \nabla\cdot(\boldsymbol{v}\theta) = J_a - J_d, \tag{11}$$

where the left hand side describes the conservation of the quantity θ, and J_a and J_d are the fluxes due to adsorption and consumption given by

$$J_a = k_a(1-\theta)C_m^i, \qquad J_d = k_d\theta^q, \quad q = m\eta + b, \tag{12}$$

where the rate constants k_a and k_d are potential-dependent and again determined by experiments [34] to be

$$k_a = k_0 \exp\left(-\frac{\alpha_a F}{RT}\eta\right), \tag{13}$$

and

$$k_d = B_d + \frac{A}{\exp(B_a(\eta + V_d)) + \exp(B_b(\eta + V_d))}, \tag{14}$$

where A, B_a, B_b, B_d and V_d are constant coefficients computed by fitting experimental results.

If we look back at Eqns. (1), (2) and (11) for the evolution of the accelerator coverage θ, their differences lie in the curvature-dependent term. We can rewrite the term $v\nabla\cdot(\boldsymbol{n}\theta)$ as

$$v\nabla\cdot(\boldsymbol{n}\theta) = v\theta\nabla\cdot\boldsymbol{n} + v\nabla\theta\cdot\boldsymbol{n} = v\theta\kappa + \nabla\theta\cdot\boldsymbol{v},$$

from which we can see that the curvature κ does indeed appear in all three equations, although we do not need to compute κ explicitly in Eqn. (11) to solve this equation.

We also note that the term $v\nabla\cdot(\boldsymbol{n}\theta)$ can be written as

$$v\nabla\cdot(\boldsymbol{n}\theta) = \nabla\cdot(v\boldsymbol{n}\theta) - \theta\boldsymbol{n}\cdot\nabla v = \nabla\cdot(\boldsymbol{v}\theta) - \theta\frac{\partial v}{\partial n},$$

where $\frac{\partial v}{\partial n}$ vanishes if the velocity v is extended in a way such that v is constant along the normal direction, which means that $\frac{\partial v}{\partial n} = 0$. Other extension choices would not make this term vanish, however we note that regardless of the choice of extension, numerical round-off errors will still lead to a non-zero component.

A summary of constant parameters used in the numerical simulation is given in Table 1. The parameters are empirically determined by performing experiments on planar surfaces and fitting the results [34].

Given the equations and parameters, our goal is to solve them one at a time in each of the following sections, starting from the level set equation for the moving interface discussed in the next section.

4 Numerical Methods for Solving the Differential Equations

4.1 Level Set Methods

To predict whether a void appears during the deposition process, it is necessary to track the interface between deposited copper and the electrolyte. Given the initial position of this interface and the speed of propagation at each point along the interface, we track the evolving interface using a level set method.

Level set methods, introduced in Osher and Sethian [35], are numerical methods for tracking moving interfaces: they rely in part on the theory of curve and surface evolution given in [39] and [40] and on the link between front propagation and hyperbolic conservation laws discussed in [41]. These techniques recast interface motion as a time-dependent Eulerian initial value partial differential equation.

The equation of motion for the evolving level set function ϕ is given by

$$\phi_t + F|\nabla\phi| + \boldsymbol{u}\cdot\nabla\phi = 0 \tag{15}$$

$$\text{given } \ \phi(x, t=0) = \pm d, \tag{16}$$

where F is the speed of propagation in the normal direction, $\boldsymbol{u}$ is the advection velocity, and $\pm d$ is the signed distance from a given point x to the initial interface. For a general introduction and overview, see Sethian [43].

Level set methods have been extended to solve material transport problems by Adalsteinsson and Sethian, see [3], see also [52]. In this paper, we shall also develop such methods but follow a different approach, leading to a conservative numerical scheme for the key variables.

Table 1: A list of some of the parameters

parameter	value	unit
b_0	0.69	A/m^2
b_1	6.4	A/m^2
m_0	0.447	–
m_1	0.299	–
Ω	7.1e−6	m^3/mol
D_c	4e−10	m^2/s
D_m	4e−10	m^2/s
V_c	14100	mol/m^3
m	4	V^{-1}
b	2.65	–
Γ_0	6.35e−6	mol/m^2
F	96485	C/mol
R	8.314	$J/K \cdot mol$
T	298	K
h	9.2e−7	m
w	5e−7	m
δ	1e−6	m
η	−0.25	V

4.2 Material Transport and Conservation Laws

The electrodeposition process was originally modeled using the leveling theory, which failed to explain the superfilling phenomenon ([33] and [48]). Later, the CEAC mechanism was proposed, and has been shown to be able to model the superconformal film growth better than the leveling theory. Superfilling is caused by the fact that the growth rate of copper is proportional to the accelerator coverage, while the rate of buildup of the accelerator scales with the local curvature κ. We will show that this can be explained more precisely by examining the role of conservation laws.

Given a short segment of the interface, the total amount of a scalar (or the integral of the scalar) is conserved when the interface moves. Consider a segment at the concave part of the trench. When the interface moves into the electrolyte, the length of this tiny segment decreases. For the integral of the scalar to be conserved, its point-wise value has to increase. The opposite is true for the convex case where the length of a segment increases as the interface moves into the electrolyte.

More specifically, we consider a closed curve moving in the xy-plane with a scalar quantity $G(\boldsymbol{u})$ defined along this curve, as shown in Figure 3. The interface is advected under the velocity field $\boldsymbol{u} = (u, v)$, which can be defined either along the front only or in the whole domain.

4.2.1 Derivation of Interface Material Transport in Conservation Form

We assume that the advection velocity depends on time and position for simplicity of the equations in our proof. With $L = \sqrt{\phi_x^2 + \phi_y^2}$, we have the following lemma.

Lemma 1.

$$(LG)_t + (uLG)_x + (vLG)_y = 0. \tag{17}$$

Proof. Consider a parameterized curve $\Gamma(s) = (\alpha(s), \beta(s))$ where $s \in [0, 1]$ and $\Gamma(0) = \Gamma(1)$ at time t_0. Let the curve

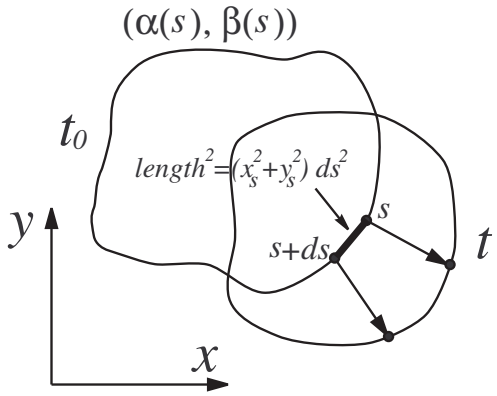

Figure 3: Moving interface in 2D from time t_0 to time t. The closed curve is parameterized as $(x, y) = (\alpha(s), \beta(s))$

propagate under the speed functions (u, v) for a small time t. Then points along the interface satisfy:

$$\begin{aligned} x(s,t) &= \alpha(s) + \int_{t_0}^{t} u(x(s,\tau), y(s,\tau), \tau) d\tau, \\ y(s,t) &= \beta(s) + \int_{t_0}^{t} v(x(s,\tau), y(s,\tau), \tau) d\tau. \end{aligned}$$

Taking the derivative of (x, y) with respect to s and omitting (s, t) and (s, τ), we have

$$\begin{aligned} x_s &= \alpha_s + \int_{t_0}^{t} (u_x \cdot x_s + u_y \cdot y_s) d\tau, \\ y_s &= \beta_s + \int_{t_0}^{t} (v_x \cdot x_s + v_y \cdot y_s) d\tau. \end{aligned}$$

Define $C(t) = x_s(t)^2 + y_s(t)^2$, which measures the length of the tangent vector along the interface at time t. Then

$$\begin{aligned} C(t)^2 = x_s^2 + y_s^2 &= \alpha_s^2 + \beta_s^2 + 2\left(\alpha_s \int_{t_0}^{t} (u_x x_s + u_y y_s) d\tau + \beta_s \int_{t_0}^{t} (v_x x_s + v_y y_s) d\tau\right) \\ &+ \left(\int_{t_0}^{t} (u_x x_s + u_y y_s) d\tau\right)^2 + \left(\int_{t_0}^{t} (v_x x_s + v_y y_s) d\tau\right)^2. \end{aligned}$$

To get the scaling of the length, we take the time derivative of $C(t)^2$. Noting that $x_s(s, t_0) = \alpha_s(s), y_s(s, t_0) = \beta_s(s)$, the differentiation yields

$$\begin{aligned} \frac{d}{dt}(C(t))^2 &= 2x_s(t_0)(u_x x_s(t) + u_y y_s(t)) + 2y_s(t_0)(v_x x_s(t) + v_y y_s(t)) \\ &+ 2\left(\int_{t_0}^{t} (u_x x_s + u_y y_s) d\tau\right)(u_x x_s(t) + u_y y_s(t)) \\ &+ 2\left(\int_{t_0}^{t} (v_x x_s + v_y y_s) d\tau\right)(v_x x_s(t) + v_y y_s(t)). \end{aligned}$$

Taking the limit as $t \to t_0$, we have

$$C(t_0)\frac{dC(t_0)}{dt} = x_s(t_0)(u_x x_s(t_0) + u_y y_s(t_0)) + y_s(t_0)(v_x x_s(t_0) + v_y y_s(t_0)).$$

The conservation of the quantity CG implies that

$$C(t,s)G(t,x_0+u(t-t_0),y_0+v(t-t_0)) = C(t_0,s)G(t_0,x_0,y_0).$$

Thus, we have

$$\frac{d}{dt}\left(\frac{G(t,x_0+u(t-t_0),y_0+v(t-t_0))}{G(t_0,x_0,y_0)}\right) = \frac{d}{dt}\left(\frac{C(t_0,s)}{C(t,s)}\right) = -\frac{C(t_0,s)\frac{dC(t,s)}{dt}}{C(t,s)^2}.$$

If we evaluate everything at time t_0, since $(n_x,n_y) = \frac{(y_s,-x_s)}{\sqrt{x_s^2+y_s^2}}$, we then have

$$\begin{aligned}(G_t+uG_x+vG_y)|_{(t_0,x_0,y_0)} &= G(t_0,x_0,y_0)\left(-\frac{C(t_0,s)\frac{dC(t_0,s)}{dt}}{C(t_0,s)^2}\right)\\ &= -G(t_0,x_0,y_0)\frac{u_x x_s^2+u_y x_s y_s+v_x x_s y_s+v_y y_s^2}{x_s^2+y_s^2}\\ &= -G(t_0,x_0,y_0)(u_x n_y^2-(u_y+v_x)n_x n_y+v_y n_x^2).\end{aligned}$$

t_0, x_0 and y_0 are arbitrary. Thus

$$G_t(x,y,t) = -(u,v)\cdot\nabla G - (n_y^2 u_x - n_x n_y(u_y+v_x)+n_x^2 v_y)G,$$

where the first term comes from advection, and the second from local compression/expansion.

Using the fact that $L=\sqrt{\phi_x^2+\phi_y^2}$, Eqn. (17) is equivalent to its expansion

$$\begin{aligned}LG_t &- G\frac{u\phi_x\phi_{xx}+(u\phi_y+v\phi_x)\phi_{xy}+v\phi_y\phi_{yy}}{L} - G\frac{u_x\phi_x^2+(v_x+u_y)\phi_y\phi_x+v_y\phi_y^2}{L}\\ &+ u_x LG + u\frac{\phi_x\phi_{xx}+\phi_y\phi_{xy}}{L}G + uLG_x\\ &+ v_x LG + v\frac{\phi_y\phi_{yy}+\phi_x\phi_{xy}}{L}G + vLG_y = 0,\end{aligned}$$

which simplifies to

$$G_t - G\frac{u_x\phi_x^2+(v_x+u_y)\phi_y\phi_x+v_y\phi_y^2}{L^2} + u_x G + uG_x + v_y G + vG_y = 0.$$

Since $(n_x,n_y) = \frac{(\phi_x,\phi_y)}{L}$, we can prove Eqn. (17) by substituting n_x and n_y.

□

Assume that in addition to advection, the curve propagates with normal speed F. The propagation can be thought of as advection under velocity field (Fn_x, Fn_y). Thus, we only need to replace (u,v) in Eqn. (17) with

$$u = u_{adv} + Fn_x, \quad v = v_{adv} + Fn_y.$$

Eqn. (17) can be generalized to the n-dimensional case in the following form:

$$(LG)_t + \nabla\cdot(LG\,\boldsymbol{u}) = 0, \tag{18}$$

where $\boldsymbol{u} = (u^1,...,u^n), u^i = u^i_{adv} + Fn^i$ for $1\le i\le n$.

Maintaining the signed distance function implies that we always have that $L=|\nabla\phi|$ is approximately equal to 1, and Eqn. (18) simplifies to

$$G_t + \nabla\cdot(G\,\boldsymbol{u}) = 0,$$

4.2.2 Conservation Laws and Numerical Scheme

Eqn. (17) is in the form of a hyperbolic conservation law: a simple numerical scheme that obeys conservation form is given by the Lax-Friedrichs method

$$u_j^{n+1} = \frac{u_{j-1}^n + u_{j+1}^n}{2} - \lambda \frac{f(u_{j+1}^n) - f(u_{j-1}^n)}{2} \tag{19}$$

where $\lambda = \frac{dt}{dx}$.

To produce a scheme that is second-order in space and time, we start with a general form

$$H_{ij}^{n+1} - H_{ij}^n = -\lambda_x(U(i,i+1) - U(i-1,i)) - \lambda_y(V(j,j+1) - V(j-1,j)), \tag{20}$$

where $H = LG$, $\lambda_x = \frac{dt}{dx}$, $\lambda_y = \frac{dt}{dy}$. A Taylor's expansion of $H(t + \Delta t)$ gives

$$\begin{aligned} U(i,i+1) &= \frac{u_{ij}H_{ij} + u_{i+1,j}H_{i+1,j}}{2} \\ &+ dt\frac{u'_{ij}H_{ij} + u'_{i+1,j}H_{i+1,j}}{4} \\ &- \lambda_x\frac{(u_{ij} + u_{i+1,j})(u_{i+1,j}H_{i+1,j} - u_{ij}H_{ij})}{4} \\ &- \frac{\lambda_y}{8}[u_{ij}(v_{i,j+1}H_{i,j+1} - v_{i,j-1}H_{i,j-1}) \\ &\quad + u_{i+1,j}(v_{i+1,j+1}H_{i+1,j+1} - v_{i+1,j-1}H_{i+1,j-1})], \end{aligned}$$

where $(\cdot)'$ is the time derivative, and $U(i-1,i), V(j,j+1), V(j-1,j)$ are defined by analogous formulas.

A first-order approximation of u' is enough for the whole scheme to be second-order, and we take $\frac{u^{n-1}-u^{n-2}}{dt}$ to approximate u'. We do not use $\frac{u^n-u^{n-1}}{dt}$, because our speed functions may depend on G, which means G^n is sometimes needed to compute u^n, and we would prefer to avoid an implicit formulation.

During the first step, since u' is not available, we simply take $u' = 0$. The first step itself is then only first-order (local truncation error is second-order). But this does not affect the second-order accuracy of the entire procedure, as this happens only once.

4.2.3 Other Implementation Issues

We use the narrow band level set method, as first introduced in [1].

With the source term included in Eqn. (11), we apply an operator splitting technique for each time step [18]. Let $A_t \circ \theta_0$ be the solution to

$$\theta_t + (u\theta)_x + (v\theta)_y = 0, \qquad \theta(t=0) = \theta_0,$$

and $S_t \circ \theta_0$ be the solution to

$$\theta_t = J_a(\theta) - J_d(\theta), \qquad \theta(t=0) = \theta_0,$$

at time t, then we can update θ by

$$\theta^{n+1} = S_{\Delta t} \circ (A_{\Delta t} \circ \theta^n),$$

where $A_{\Delta t}$ is computed using the conservative scheme explained above and $S_{\Delta t}$ is simply defined as

$$S_{\Delta t} \circ \theta = \theta + \Delta t(J_a(\theta) - J_d(\theta)).$$

4.2.4 Numerical Tests

In this section, we test the numerical schemes to demonstrate convergence and second-order accuracy. To do so, we solve a problem with known exact solution. Consider a test case (given in [3]) in which we have an ellipse with the

origin as its center and major axes 0.3 and 0.4. The ellipse is rotating around its center; equivalently, for each point (x, y) on the interface, it moves with speed $\boldsymbol{u} = (-y, x)$. The scalar value G is chosen to be

$$G_0(x, y) = x^2 + y^2 - \frac{xy}{\sqrt{x^2 + y^2}} + 0.1. \tag{21}$$

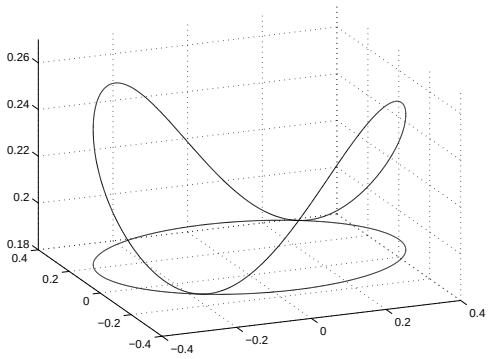

Figure 4: The interface and the exact solution

We test our numerical algorithm with four different mesh sizes: $n = 122, n = 182, n = 242$ and $n = 362$. The errors are computed by subtracting the numerical solution interpolated from the grid points to the interface and the exact solution on 1000 evenly distributed points. We compute both the 2-norm and the ∞-norm errors (see Figure 5 and Figure 6). If we use straight lines to interpolate the two sets of points, the slopes of the lines are -2.13 and -2.16 respectively, which means that the order of accuracy is approximately 2.

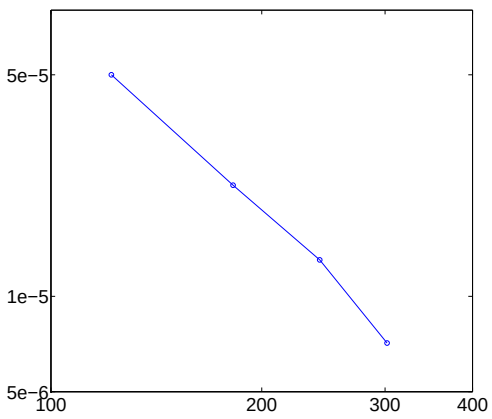

Figure 5: 2-norm error vs. mesh size

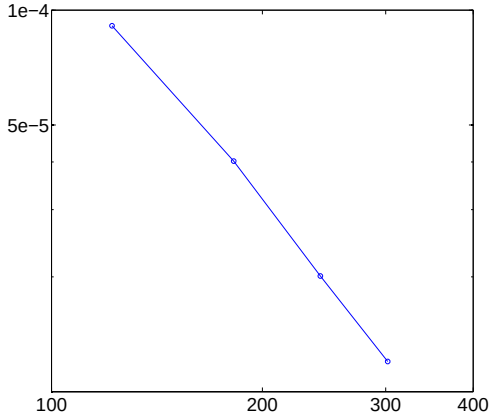

Figure 6: ∞-norm error vs. mesh size

4.3 Diffusion Equation on an Irregular Domain with Moving Interface

In this section, we consider the diffusion equation

$$u_t = \beta \Delta u \tag{22}$$

in a two-dimensional region $\Omega(t)^+$, with mixed boundary condition

$$\beta \frac{\partial u}{\partial n} + \alpha u = g(\boldsymbol{x}, t) \qquad \text{on} \qquad \Gamma = \partial\Omega(t)^+. \tag{23}$$

Cartesian grid finite difference methods are problematic for handling such boundary conditions on irregular, moving interfaces. The difficulty is that first-order one-sided difference approximations to the normal derivatives close to the interface, combined with a standard five-point stencil scheme for the Laplacian at regular interior points, are only first-order accurate, while second-order approximations to the derivative yield desired accuracy at the cost of using either a wide difference stencil or grid points from the other side. Another problem which results from the moving interface is that an outward motion from $\Omega(t)^+$ to $\Omega(t+dt)^+$, where $\Omega(t)^+ \subset \Omega(t+dt)^+$, requires the estimation of values of $u(t)$ defined on $\Omega(t+dt)^+ - \Omega(t)^+$ to update from $u(t)$ to $u(t+dt)$.

4.3.1 Immersed Interface Methods

We will embed the irregular domain in a larger rectangular domain, with the partial differential equations extended to the rectangular domain correspondingly by introducing jump conditions across the interface, and then apply the immersed interface method to the rectangular domain. From now on, we will use u to denote both the numerical solution and the analytical solution for simplicity.

To embed Eqns. (22) and (23) into the larger rectangular domain, we would like to impose the jump conditions

$$[u] \equiv u^+ - u^- = 0 \tag{24}$$

and

$$[\Delta u] \equiv (\Delta u)^+ - (\Delta u)^- = 0 \tag{25}$$

so that Eqn. (22) holds in the domain $\Omega = \Omega^+ \cup \Omega^-$. The boundary condition given by Eqn. (23) can be rewritten as

$$\beta \frac{\partial u^+}{\partial n} + \alpha u^+ = g. \tag{26}$$

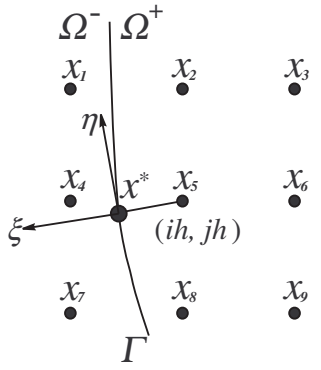

Figure 7: x_5 is an irregular point with one of its neighbors on the other side of the interface. x^* is its closest point on the interface Γ

Consider the grid point x_5 as shown in Figure 7. We label points (like x_5) with neighbors on the other side of the interface as "irregular points". Let x^* be the closest point on Γ to x_5. Applying the immersed interface method (see [26] and [27]) to Eqn. (22) with jump conditions given in Eqns. (24), (25) and (26) leads to a local linear system for x_5:

$$\sum_{k \in K^+} \left(1 - \frac{\alpha}{\beta}\xi_k - \left(\frac{\alpha}{\beta}\right)' \xi_k \eta_k + \frac{\chi''}{2}\frac{\alpha}{\beta}(\eta_k^2 - \xi_k^2) \right) \gamma_k + \sum_{k \in K^-} \gamma_k = 0, \tag{27}$$

$$\sum_{k\in K^+} \frac{\chi''}{2}(\eta_k^2 - \xi_k^2)\gamma_k + \sum_{k\in K^-} \xi_k\gamma_k = 0, \tag{28}$$

$$\sum_{k\in K^+} \left(\eta_k + \left(\chi'' - \frac{\alpha}{\beta}\right)\xi_k\eta_k\right)\gamma_k + \sum_{k\in K^-} \eta_k\gamma_k = 0, \tag{29}$$

$$\sum_{k\in K^+} \xi_k^2\gamma_k + \sum_{k\in K^-} \xi_k^2\gamma_k = 2, \tag{30}$$

$$\sum_{k\in K^-} \xi_k\eta_k\gamma_k = 0, \tag{31}$$

$$\sum_{k\in K^+} \eta_k^2\gamma_k + \sum_{k\in K^-} \eta_k^2\gamma_k = 2, \tag{32}$$

where $\{\gamma_k\}$ is the set of unknown coefficients, K^+ and K^- are a partition of the nine-point stencil which are the indices of those points that are in $\Omega^+\cup\Gamma$ and Ω^- respectively, (ξ_k, η_k) are the coordinates for x_k in the local coordinate system, $\xi = \chi(\eta)$ is the local representation of the interface, and χ'' is the curvature evaluated at x^*. Then the Laplacian is approximated as

$$\Delta u \approx \sum_{k\in K^+} \gamma_k u(x_k) + \sum_{k\in K^-} \gamma_k u(x_k) - C,$$

where

$$C = \sum_{k\in K^+} \gamma_k \left\{ \frac{g}{\beta}\left[\xi_k - \frac{1}{2}\chi''(\eta_k^2 - \xi_k^2)\right] + \left(\frac{g}{\beta}\right)' \xi_k\eta_k \right\}.$$

4.3.2 Stencil Reduction

The next goal to determine which six points one should choose out of the nine-point stencil. As a preliminary, if we look at Eqns. (27) - (32) carefully, we find out that if we multiply Eqn. (30) by $\frac{1}{2}\chi''$ and Eqn. (32) by $-\frac{1}{2}\chi''$ and add them to Eqn. (28), we have

$$\sum_{k\in K^-} \left\{ \xi_k + \frac{1}{2}\chi''(\eta_k^2 - \xi_k^2) \right\} \gamma_k = 0. \tag{33}$$

This can be used to replace Eqn. (28). We observe that both Eqn. (31) and Eqn. (33) have nonzero terms only for grid points in Ω^-. Therefore, if there are exactly two exterior points in the six-point stencil we choose, the coefficients of these two grid points are zero, and we only need solve the four equations (27), (29), (30) and (32) for the coefficients for the four chosen interior points.

This observation not only reduces a 6×6 linear system to a 4×4 system, it saves a considerable amount of computation time, assuming that we are able to choose a stencil with exactly four interior grid points and two exterior points for each interior irregular point. In such cases, for each irregular point, we have a 4×4 linear system only involving interior points. We also know that the standard five-point stencil for any regular points consists of interior points only. Based on these two observations, and under the assumption of the existence of four interior points and two exterior points, what happens to the exterior points is irrelevant. The interior part can be solved independent of the exterior, though the values of the exterior points depend on the interior. Consider a unit circle embedded in a square region with side length 2. Then with an $N \times N$ discretization, the assembled linear system we need to solve has size about $\frac{\pi}{16}N^2$ instead of N^2, which is a significant improvement.

4.3.3 Stencil Selection for High-Curvature Interfaces

The above discussion is based on the assumption that for each interior irregular point, there exists a six-point stencil with four interior points and two exterior points. Unfortunately, this is not always true. For example, situations shown in Figure 8 and Figure 9 are possible. Most significantly, this happens in our electrodeposition problem when sharp corners arise at the bottom of the trench.

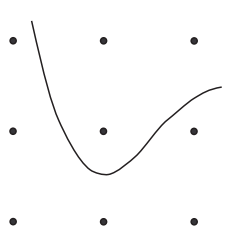

Figure 8: An irregular point with three interior neighbors

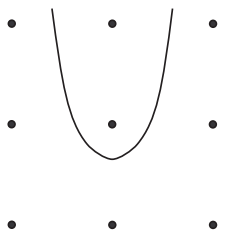

Figure 9: An irregular point with two interior neighbors

We do not want to add exterior grid points into the set of interior points and solve for them. If we add one exterior point, we also need to add its neighbors due to the dependency. Similarly the neighbors' neighbors should be added as well. If we choose this approach, we are soon required to solve the full linear system defined in the whole domain $\Omega = \Omega^+ \cup \Omega^-$ instead of the reduced system in Ω^+.

Instead, if we reexamine the procedure by which we deduce the six equations for the coefficients $\{\gamma_k\}$, we will see that nowhere do we assume that x_k's need be grid points. With this observation, if an interior irregular point has (including itself) less than four interior neighbors, we can just pick some arbitrary points which lie in Ω^+ so that we have four interior points in total, and we are then able to compute the four equations they satisfy. The complication with this approach is that we are in fact introducing new points into the existing system, and they require corresponding equations.

These equations can be determined if the additional points are carefully chosen. Consider an interior irregular point that has only three interior neighbors including itself, as shown in Figure 10. We can find its neighbor closest to the interface among all the exterior neighbors, then attach to this exterior point the information of its closest point to the interface, and use the local coordinates of this closest point to compute the coefficients for the 4×4 or 6×6 linear system. This new point is counted as an interior point and all the information we need about it are its local coordinates.

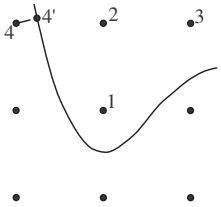

Figure 10: Replacing an exterior neighbor x_4 with the closest point x_4'

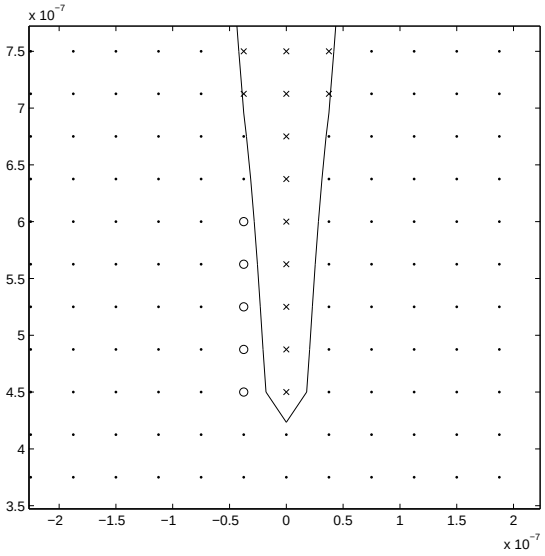

Figure 11: Adding exterior points into the unknown set

In some cases, the corresponding exterior point already has four interior neighbors from which we can compute the four equations. If this is not true, we perform the above procedure once more to this new point. The worst case

situation occurs when there is a thin tube with width less than one grid cell. Unlike the previous idea, this procedure of adding new points will not keep going forever even in this worst case. It terminates with at most one layer of new grid points as shown in Figure 11, where $\times$ are interior points, $\circ$ are exterior points that have been added into the unknown set, and $\bullet$ are exterior points that will not be used to solve for the linear system. We can see that each interior and newly added points has (including itself) at least four neighbors which are interior points or newly added points.

To implement this algorithm, we first loop through all the interior points. For each irregular interior point with less than four interior neighbors, we add some exterior points into the set and treat them as interior points. For any of these points that has less than four interior neighbors, we add more points into the set recursively. After the loop, each interior irregular point should have at least four interior neighbors. Therefore, we can compute the 4×4 matrix for each irregular point, solve for the coefficients, and assemble the big matrix.

4.3.4 Linear Programming and Least Squares in Stencil Selection

In contrast to what was discussed in the last section, sometimes an irregular point can have more than four interior neighbors. In this case, the simplest thing we can do is to choose four points randomly. The question is: are there better ways to do it?

Different stencil leads to different assembled matrices, some of which are much more ill-conditioned than others. Our goal is to make the big matrix good-conditioned, or more advantageously, a diagonally dominant matrix, so that an iterative solver used to solve the linear system converges fast.

One way to select such a stencil is via an associated linear programming problem. Assume that $\{\gamma_k\}$ is the set of coefficients, and that $k = 1$ corresponds to the center of the nine-point stencil. We then want to solve:

$$\begin{aligned} \max \quad & |\gamma_1| - \sum_{k \neq 1} |\gamma_k| \\ \text{s.t.} \quad & \sum_j c_{ij}\gamma_j = b_i, \quad 1 \leq i \leq 4. \end{aligned} \tag{34}$$

For a regular point, the coefficients are $\gamma_1 = -4$ and $\gamma_k = 1$ for $k \neq 1$. Therefore we would like to impose the conditions $\gamma_1 \leq 0$ and $\gamma_k \geq 0, k \neq 1$, and the optimization problem (34) simplifies to

$$\begin{aligned} \max \quad & -\gamma_1 - \sum_{k \neq 1} \gamma_k \\ \text{s.t.} \quad & \gamma_1 \leq 0, \\ & \gamma_k \geq 0, \quad k \neq 1 \\ & \sum_j c_{ij}\gamma_j = b_i, \quad 1 \leq i \leq 4. \end{aligned} \tag{35}$$

The objective function tries to make the matrix as close as possible to be diagonally dominant, and the equality constraints are the four equations satisfied by the coefficients $\{\gamma_k\}$. This approach was also discussed in [28] and [51].

The optimization problem (35) does not always have a solution. If the solution does not exist (sometimes the inequality and equality constraints cannot be satisfied all at the same time), we switch to solving the least square problem given by

$$\begin{aligned} \min \quad & \sum_j \gamma_j^2 \\ \text{s.t.} \quad & \sum_j c_{ij}\gamma_j = b_i, \quad 1 \leq i \leq 4. \end{aligned} \tag{36}$$

The advantage of solving the above additional optimization problem is that it makes solving the whole linear system with an iterative solver easier. For example, if we use Gauss-Seidel as the iterative solver to solve the linear system $Ax = b$, the convergence rate is related to the magnitude of

$$||M|| = ||(D + L)^{-1}U||,$$

where D, L and U represent the diagonal, strictly lower triangular, and strictly upper triangular parts of the coefficient matrix A. The 2-norm of M is usually less than 1 if stencils are chosen based on the solutions of the above optimization problems, while $||M||_2$ can be as large as 100 if stencils are chosen randomly, making it almost impossible for the iteration to converge.

4.3.5 Second-Order Extension of the Solution

For our problem, the accelerator coverage θ is updated in a narrow band around the moving interface according to Eqn. (11), where the source term $J_a(\theta)$ depends on C_m. Thus we need the concentration values in this narrow band, instead of just on one side Ω^+. Moreover, for other problems where the interface moves outwards, it is possible that $\Omega^+(t+dt) \not\subset \Omega^+(t)$, that is, there exist points in $\Omega^+(t+dt)$ that are not in $\Omega^+(t)$. Therefore, we need to extend the concentration values from Ω^+ to Ω^- at each time step. A C^2-extension is desired because the concentrations satisfy the diffusion equation with second-order derivative terms (in fact, usually an arbitrary order extension can be made using the same mechanism to be explained in this section). A second-order extension usually suffices; similiar to polynomial interpolation and extrapolation, extrapolated functions tend to be oscillating if higher order polynomials are used.

The velocity extension mechanism associated with the level set method is explained in [2]. A C^2-extension is quite similar. Assume that we would like to extend u to the other side. First we make a continuous or C^0-extension for the second-order normal derivative u_{nn}, followed by a C^1-extension for the first-order normal derivative u_n using the values of u_{nn}, and then eventually a C^2-extension for u using the values of u_n.

To be more specific, we first compute the value of u_n as

$$u_n = \nabla u \cdot \boldsymbol{n} = \nabla u \cdot \nabla \phi = u_x \phi_x + u_y \phi_y,$$

since the signed distance ϕ is extended such that $|\nabla \phi| = 1$. The computation is performed on each point in Ω^+ with all four neighbors in Ω^+, since these four neighbors are needed to approximate u_x and u_y using central differences. Similarly, we compute u_{nn} as

$$u_{nn} = \nabla u_n \cdot \boldsymbol{n} = (u_n)_x \phi_x + (u_n)_y \phi_y,$$

and we do this for points where the values of u_n have been computed already as above at all four neighbors. With such values of u_{nn} and u_n available, we extend u_{nn} in the same manner as the way in which we extend the velocity function according to $\nabla u_{nn} \cdot \nabla \phi = 0$

$$(u_{nn})_{ij} = \frac{(u_{nn})_{i-1,j} \cdot \frac{\phi_{ij}-\phi_{i-1,j}}{dx^2} + (u_{nn})_{i,j-1} \cdot \frac{\phi_{ij}-\phi_{i,j-1}}{dy^2}}{\frac{\phi_{ij}-\phi_{i-1,j}}{dx^2} + \frac{\phi_{ij}-\phi_{i,j-1}}{dy^2}},$$

assuming that we are updating (i, j) from $(i-1, j)$ and $(i, j-1)$. For u_n, we use a slightly modified equation

$$\nabla u_n \cdot \nabla \phi = u_{nn}$$

which can be verified to be a C^1-extension. The corresponding difference equation is

$$(u_n)_{ij} = \frac{u_{nn} + (u_n)_{i-1,j} \cdot \frac{\phi_{ij}-\phi_{i-1,j}}{dx^2} + (u_n)_{i,j-1} \cdot \frac{\phi_{ij}-\phi_{i,j-1}}{dy^2}}{\frac{\phi_{ij}-\phi_{i-1,j}}{dx^2} + \frac{\phi_{ij}-\phi_{i,j-1}}{dy^2}}.$$

Similarly we can build a C^2-extension for u:

$$u_{ij} = \frac{u_n + u_{i-1,j} \cdot \frac{\phi_{ij}-\phi_{i-1,j}}{dx^2} + u_{i,j-1} \cdot \frac{\phi_{ij}-\phi_{i,j-1}}{dy^2}}{\frac{\phi_{ij}-\phi_{i-1,j}}{dx^2} + \frac{\phi_{ij}-\phi_{i,j-1}}{dy^2}}.$$

We note that if we start from a continuous extension of $\frac{\partial^k u}{\partial n^k}$, we can build a C^k-extension.

The computations of u_n and u_{nn} do not need to be performed at every point: they are needed only within a very thin narrow band. For example, consider the interface in Figure 12. We only need to compute u_n at points marked as $\times$ which makes a band of bandwidth around 3 and then compute u_{nn} at points marked as $\circ$ which makes a band of bandwidth about 1, since that is all we need to extend u to the other side of the interface.

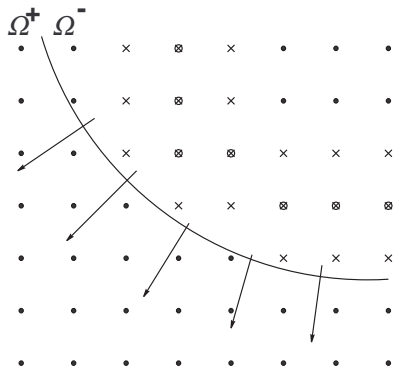

Figure 12: The grid points that are involved in making a second-order extension

4.3.6 Numerical Tests

In this section, we test our proposed methods by solving several problems numerically. For all the problems in this section, we use the simplest form of the diffusion equation with the same exact solution, but defined on different and possibly moving regions.

If we let

$$u(r,t) = \exp(-t) \cdot \sum_{k=0}^{\infty} (-1)^k \frac{r^{2k}}{4^k (k!)^2},$$

where $r = r(x,y) = \sqrt{x^2+y^2}$, then $u(x,y,t)$ is an exact solution to the diffusion equation $u_t = \Delta u$. For the mixed boundary conditions, we can take an arbitrary function α and let g be defined as $g = \frac{\partial u}{\partial n} + \alpha u$ where $\boldsymbol{n}$ is the unit normal pointing inwards, then

$$\frac{\partial u}{\partial n} + \alpha u = g$$

can be taken as the boundary condition. For example, we set $\alpha = r^2$, and

$$\begin{aligned} g &= \frac{\partial u}{\partial n} + \alpha u = -\frac{\partial u}{\partial r} + r^2 u \\ &= \exp(-t) \left(-\sum_{k=0}^{\infty} (-1)^{k+1} \frac{r^{2k+1}(k+1)}{4^k \cdot 2 \cdot ((k+1)!)^2} + \sum_{k=0}^{\infty} (-1)^k \frac{r^{2k+2}}{4^k (k!)^2} \right). \end{aligned}$$

Solving the Diffusion Equation on a Nice, Fixed Domain First we solve the diffusion equation on the unit disk with the above boundary condition defined along the unit circle. The immersed interface method discussed in Section 4.3.1 with Taylor expansion and appropriate stencil selection is used to solve the diffusion equation. The numerical solution is shown in Figure 13.

Solving the Diffusion Equation on a Moving Domain with Large Curvature The second test is performed on an origin-centered star-shaped region, which rotates around the origin. The star shape has very sharp corners at some points, thus we may not be able to find four neighbor grid points on the same side of the interface for each point. We then adopt the trick mentioned in Section 4.3.3 to deal with the problem. We still have the same solution as before except that this time it is defined on a different region as in Figure 15.

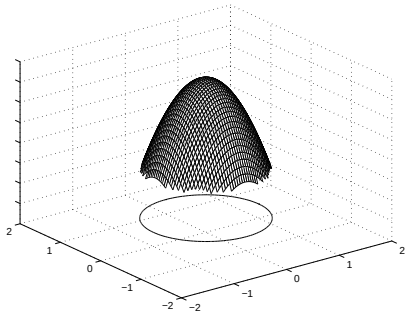

Figure 13: Solution on the unit disk with fixed interface

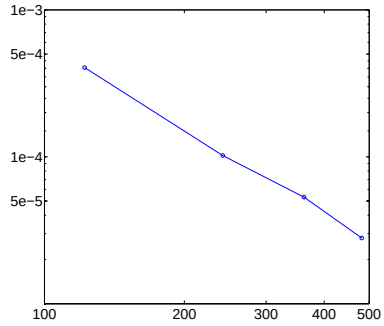

Figure 14: Maximum error on the unit disk with fixed interface

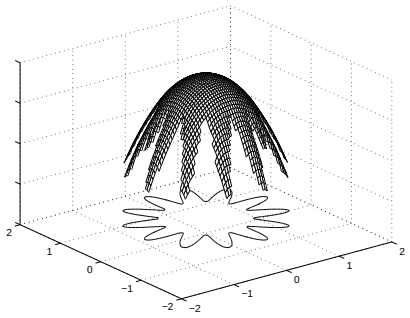

Figure 15: Solution on the rotating star shape

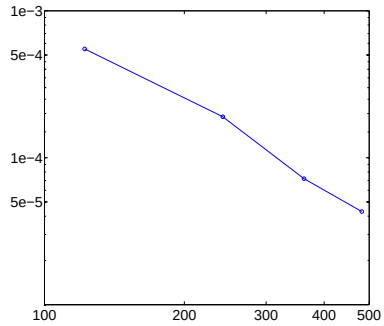

Figure 16: Maximum error on the rotating star shape

Solving the Diffusion Equation on an Expanding Domain Lastly, we test on an expanding star shape. Since $\Omega(t) \subset \Omega(t + dt)$ for this test problem, we can verify correctness of our method for extension in Section 4.3.5 by comparing the numerical solution with the exact solution. The numerical solution is shown in Figure 17.

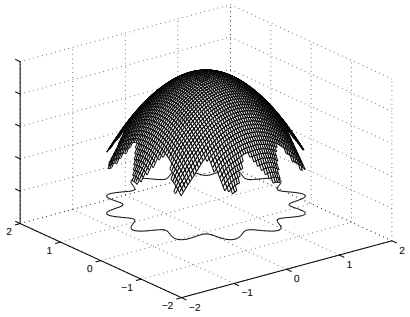

Figure 17: Solution on the expanding star shape

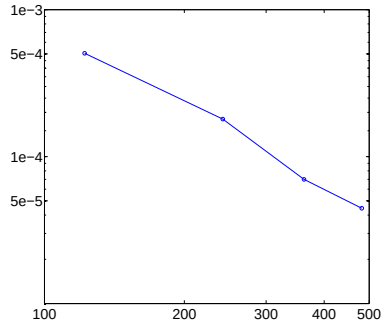

Figure 18: Maximum error on the expanding star shape

For each of the three test cases, we compute the maximum norm errors in the desired domain for four different mesh sizes and make a plot of error versus mesh size. The plots are shown in Figures 14, 16 and 18. The slopes of the

lines for the three test cases are approximately -1.99, -1.85, and -1.86 respectively, which verify the second-order accuracy of our numerical scheme.

5 Solving Large Linear Systems

Once we have a discrete approximation to the Laplacian operator, the differential equation (22) can be solved by either an explicit scheme or an implicit scheme:

$$\frac{u^{n+1}-u^n}{dt} = \beta\Delta^n u, \qquad \frac{u^{n+1}-u^n}{dt} = \frac{\beta}{2}(\Delta^n u + \Delta^{n+1}u), \qquad \text{or} \qquad \frac{u^{n+1}-u^n}{dt} = \beta\Delta^{n+1}u, \tag{37}$$

where Δ^n and Δ^{n+1} are the approximations to the Laplacian computed from section 4.3 at time t^n and t^{n+1} respectively.

If an explicit scheme is used to solve the diffusion equation, the time step size Δt and spatial step size Δx are required to satisfy the CFL condition $\beta\frac{\Delta t}{\Delta x^2} \leqslant C$ for some constant C of magnitude 1 for the method to be stable. For our problem, we have $\Delta x \sim 10^{-8}$ and $\beta \sim 10^{-10}$. Thus we need to have $\Delta t \lesssim 10^{-6}$ for stability. A physical deposition takes about 100s to complete, which means that for our numerical calculation with time step $\Delta t \lesssim 10^{-6}$, 10^8 steps will be needed for the whole process. Assuming that computing one time step takes 0.01s, the whole process will take a total of 10^6s, which is less than desirable.

However, if an implicit scheme is used, the numerical method for the diffusion equation is unconditionally stable, and we can take Δt as large as we want, as long as the desired accuracy is achieved and the numerical methods for other equations (especially the level set equation) are stable. Solving a differential equation with an implicit scheme involves solving a large sparse linear system. For this deposition problem, the size of the linear system is about 10000 – 15000 if we use a mesh of size 100×200, and the coefficient matrix is not symmetric. We need to solve two such linear systems at each time step, one for the concentration of the copper and one for the accelerator.

The second-order Crank-Nicolson method sounds appealing, but it causes oscillations in the numerical solutions for our problem unless the time step is as small as Δx^2. Moreover, even if we use a second-order accurate method to solve diffusion equation, the overall method is at most first-order accurate: for each time step, the differential equations (level set equations, conservation laws, and diffusion equations) are solved sequentially without the use of any time splitting technique to make it second-order.

We now consider both direct and iterative methods. For direct methods, one typically has two phases: symbolic determination of the nonzero structure of the factors, and numeric factorization and solutions. If the linear systems for different time steps have the same nonzero structure, then phase one needs to be performed only once. This leads to a very efficient scheme for solving the linear systems.

In our problem, however, the boundary of the interface is moving. It crosses some grid points at each time step, leading to a change in the structure of the coefficient matrix at such points. Moreover, we may want to write out the linear system only on one side of the interface instead of the whole rectangular domain for efficiency. In this case even the dimension of the linear system changes since the total number of points changes.

An iterative method starts with an initial guess, and its performance is closely related to the accuracy of the initial guess and the convergence rate of the coefficient matrix. The concentration values at consecutive time steps do not differ much, thus the solution from the last time step would be a good starting point. We can even make an initial guess by extrapolating from the last two steps.

This section gives an efficient iterative multigrid method to solve a large linear system resulting from the finite difference discretization of a differential equation on a rectangular mesh.

5.1 The Basics of the Multigrid Method

The four basic components of multigrid methods are the smoothing, interpolation, and restriction operators, and the definitions of coarse grid problems (see [9] and [31]). We define these operators as follows:

Smoothing Operator S^h We use a variation of the red-black Gauss-Seidel iteration as the smoothing operator. As shown in Figure 19, where we first update the points at the corner ($\bullet$) which are also the coarse grid points, followed

by the points in the center ($\circ$), and then points on the vertical edge ($\triangle$) and finally points on the horizontal edge ($\times$). Assume that the finite difference scheme is based on the standard nine-point stencil. Since for each of the four sets, the update of one point is completely independent of the values of the points in the same set, this ordering has a clear advantage in terms of parallel computation. Let the smoothing operator with mesh size h be S^h.

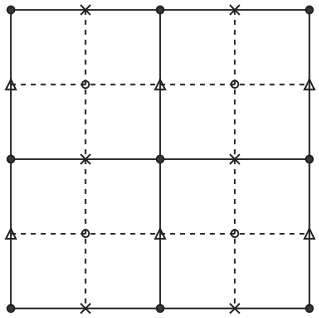

Figure 19: Variation of red-black Gauss-Seidel

Interpolation Operator I_{2h}^h To interpolate from coarse grid values to fine mesh, we can use a linear interpolation. In the one-dimensional case we have

$$\begin{aligned} u_{2i}^h &= (I_{2h}^h u^{2h})_{2i} = u_i^{2h}, \\ u_{2i+1}^h &= (I_{2h}^h u^{2h})_{2i+1} = \frac{1}{2}(u_i^{2h} + u_{i+1}^{2h}), \end{aligned}$$

and for two-dimensional problems,

$$\begin{aligned} u_{2i,2j}^h &= (I_{2h}^h u^{2h})_{2i,2j} = u_{ij}^{2h}, \\ u_{2i+1,2j}^h &= (I_{2h}^h u^{2h})_{2i+1,2j} = \frac{1}{2}(u_{ij}^{2h} + u_{i+1,j}^{2h}), \\ u_{2i,2j+1}^h &= (I_{2h}^h u^{2h})_{2i,2j+1} = \frac{1}{2}(u_{ij}^{2h} + u_{i,j+1}^{2h}), \\ u_{2i+1,2j+1}^h &= (I_{2h}^h u^{2h})_{2i+1,2j+1} = \frac{1}{4}(u_{ij}^{2h} + u_{i+1,j}^{2h} + u_{i,j+1}^{2h} + u_{i+1,j+1}^{2h}). \end{aligned}$$

Restriction Operator R_h^{2h} Two straightforward ways to project the solution from the fine mesh (with mesh size h) to the coarse mesh (with mesh size $2h$) are to use a weighted average of the values at neighboring fine mesh points, or to simply use the value at the same grid point. However, for a coarse grid point next to the interface, not all its neighboring fine mesh points are easily available. Therefore we adopt the second choice.

Coarse Grid Problem Still yet to be determined is the coarse grid problem. We are given only the information to solve for the finest grid in the form of $A^h u^h = f^h$ where A^h and f^h are known, and u^h is to be solved. Similarly the coarse grid problem can be written as $A^{2h} u^{2h} = f^{2h}$, and one commonly used way to define the problem is

$$A^{2h} = R_h^{2h} A^h I_{2h}^h, \qquad \text{and} \qquad f^{2h} = R_h^{2h} f^h,$$

where the first equation is called the Galerkin condition.

With the smoothing, restriction, interpolation and coarse-grid operators defined as above, a family of multigrid cycling schemes called the μ-cycle method is given in [9]. We usually set either $\mu = 1$ (called the V-cycle multigrid) or $\mu = 2$ (called the W-cycle multigrid), and the numbers of smoothing steps ν_1 and ν_2 seldom exceed 3. To apply this multigrid algorithm to our deposition problem, three things should be noted.

First, we have a constant Dirichlet boundary condition at the top of the whole rectangular region, and the concentration values at points that are far away from the interface are not likely to change much across different time steps, compared with points close to the interface. To save the workload further, we may not have to perform the smoothing step for each point in the rectangular domain or Ω^+. Instead, consider a total of three levels. We can carry out the procedure on the finest level mesh for points in a band around the interface, and for the second level, we do it on a thicker band which contains the band for the first level. We only update all the points for the coarsest level, as shown in Figure 21 for the interface given in Figure 20.

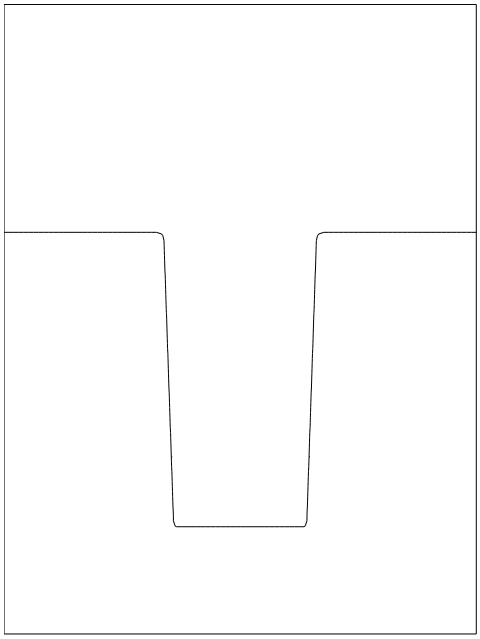

Figure 20: Some arbitrary interface position

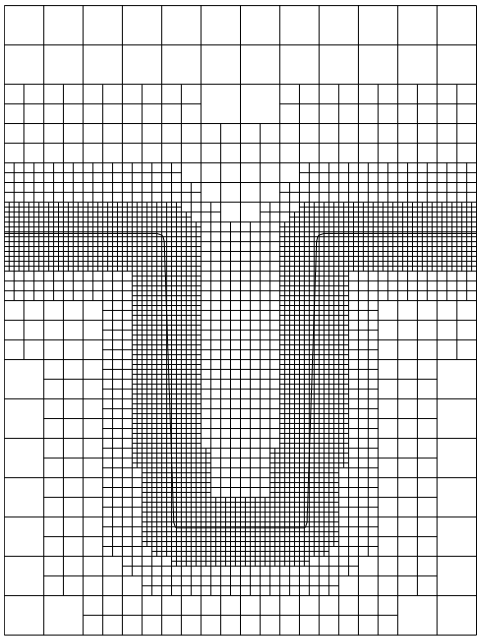

Figure 21: Adaptive mesh corresponding to a given interface

Second, the unknown variables are the concentrations of copper and the accelerator, which are defined only on one side of the interface Ω^+, an irregular domain. Thus we may not know how to write the equations for points in Ω^- that are on the other side, especially for points near the interface. Even if the equations are available to us, we would prefer to solve only in Ω^+ as mentioned before for efficiency.

Finally, the interpolation operator defined above is based on an underlying assumption that the solution (or the error) is smooth, which is seldom true for any interface problem. We will use instead an "operator-induced" interpolator which is explained in detail in later sections.

5.2 Adaptive One-sided Multigrid

The basic idea of the adaptive multigrid method is to use a finer mesh where more accurate solutions are desired. For most interface problems, due to mixed boundary conditions on the interface or jump conditions across the interface, we seek more accurate solutions near the interface where accuracy is required and where more significant changes in the solutions of consecutive time steps will occur, compared with regions further from the interface. This reduces the workload considerably since we only need to do smoothing, a major computation part in the whole multigrid algorithm, at the finest level in a thin narrow band around the interface, rather than throughout the whole domain.

The way we use to generate adaptive mesh is to first assign value to a variable bw, which is the bandwidth, and then loop through the coarsest level to the second-finest level. Assume that the current level has mesh size h. For each regular cell in this level, if the distance between all the four corners (eight corners for three-dimensional problem) and the interface d_i satisfies $d_i < h \cdot bw$, we split the cell into four smaller cells, which are taken as regular cells for

the next finer level. For example, given the interface in Figure 20 with $bw = 3$ and four levels in total, the adaptive mesh looks like Figure 21.

To switch from the original non-adaptive multigrid to the adaptive multigrid scheme, everything is the same, except that we cannot do the same thing for the interpolation operator for points which lie between different levels of meshes, for example, P in Figure 22. Here, we need an alternative approach to transfer between coarse and fine meshes for such points.

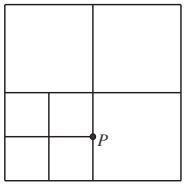

Figure 22: A point with two different levels of neighbors

The rest of the section explains how we map from fine mesh to coarse mesh and vice versa in addition to the smoothing operator. We need to pay attention to the domains and ranges of the mappings, especially across the boundary between coarse and fine mesh. The details of the algorithm are given at the end of this section.

Now consider two levels for simplicity. Similar to the Fast Adaptive Composite Grid Method (FAC) [31], we divide the points into four groups and add a set of imaginary points as in Figure 23. The fine mesh points with all neighbors being fine mesh points are the first group ($\times$); the remaining fine mesh points are the second group ($\bullet$); we add two more layers of imaginary fine mesh points with the first layer being the third group ($\triangle$) and the second layer being the fourth group ($\circ$); the remaining coarse mesh points are the last group ($\star$). In groups 1, 2 and 4, some points can be either fine mesh points or coarse mesh points. We use upper indices to differentiate: $\circ^h$ is for fine mesh points, and $\circ^{2h}$ is for coarse mesh points.

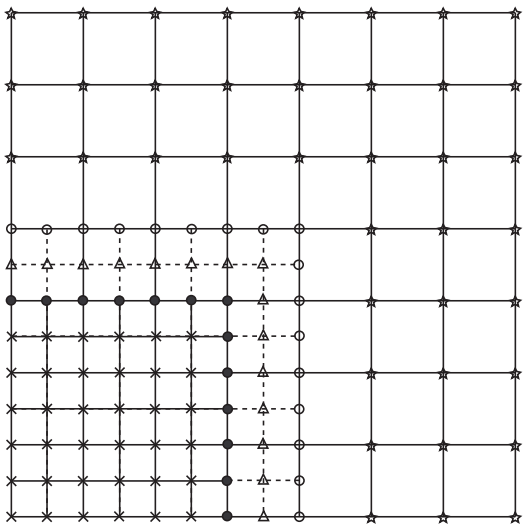

Figure 23: Grid points divided into several groups

Now consider the points in $\circ^h$. We cannot perform the smoothing step on them as mentioned above since some of their neighbors do not exist. Therefore, the original multigrid algorithm needs to be revised. We store the fine mesh solution values u^h for all the grid points that are in $\times^h \cup \bullet^h \cup \triangle^h \cup \circ^h$ and the coarse mesh solution values u^{2h} for points in $\times^h \cup \bullet^h \cup \circ^h \cup \star^h$. For the fine mesh, the smoothing step is only performed on points in $\times^h$, and solutions at the remaining mesh grid points, i.e., those in $\bullet^h \cup \triangle^h \cup \circ^h$, are interpolated from coarse grid solutions.

The domain and range of the restriction and interpolation operators are specified as:

$$R_h^{2h} : \triangle^h \cup \bullet^h \cup \times^h \to \bullet^{2h} \cup \times^{2h}, \qquad \text{and} \qquad I_{2h}^h : \times^{2h} \cup \bullet^{2h} \cup \circ^{2h} \to \times^h \cup \bullet^h \cup \triangle^h \cup \circ^h,$$

and the coefficient operator for level h is given by

$$A^h : \times^h \cup \bullet^h \to \times^h.$$

Written in matrix form, A^h has dimension $(N^h + m^h) \times (N^h)$, where N^h and m^h are the number of points classified as $\times^h$ and $\bullet^h$ respectively. A^h is not a square matrix anymore, because we must use points in $\bullet^h$ in the smoothing step to update a point in $\times^h$ with one of its neighbors in $\bullet^h$. Correspondingly, f^h is a vector of size $(N^h + 2)$. To compute the entries of A^{2h}, we still use the Galerkin condition $A^{2h} = R_h^{2h} A^h I_{2h}^h$ for points in the domain of I_{2h}^h, and we will compute the entries of A^{2h} elsewhere from the discretization of the partial differential equation since no information about A^h is available there. The solutions u^h at all levels need to be recorded, and the same is true for f^h. We use another vector v^h to record the solution at the finest level and the residual at all other levels.

With all of the above, the multigrid scheme is modified as follows:

0. Compute the operators R_h^{2h}, I_{2h}^h and A^h for each level in matrix form as above. Find an initial guess u^h and let $f^h \leftarrow f^h - A^h u^h$ for each level.

$v^h \leftarrow AM^h(v^h, f^h)$:

1. Smoothing: repeat $v^h \leftarrow S^h(A^h, f^h, v^h)$ ν_1 times for points in $\times^h$.
2. Compute the residual $r^h \leftarrow f^h - A^h v^h$ in $\times^h \cup \bullet^h \cup \triangle^h$;
   transfer it to the coarse mesh $f^{2h} \leftarrow R_h^{2h} r^h$ in $\times^{2h} \cup \bullet^{2h}$.
3. If $2h$ is the coarsest level, direct solve $v^{2h} \leftarrow DS(A^{2h}, f^{2h})$, else $v^{2h} \leftarrow 0, v^{2h} \leftarrow AM^{2h}(v^{2h}, f^{2h})$.
4. $f^{2h} \leftarrow f^{2h} - A^{2h} v^{2h}$ in $\star^{2h} \cup \circ^{2h}$.
5. Update coarse approximation for all coarse points: $u^{2h} \leftarrow u^{2h} + v^{2h}$.
6. Interpolate correction and update approximation for all fine points: $v^h \leftarrow v^h + I_{2h}^h v^{2h}$.

Finally, if we are close to the interface, we discretize the partial differential equation, choose the stencil and add new points following the procedure described in Section 4.3: this multigrid scheme can be easily turned into a one-sided scheme.

5.3 Operator-induced Interpolation Operator

A problem with the multigrid method is that it has to be specifically designed, or at least coded for each problem due to different grid configurations, especially for interface problems with various irregular boundaries, since the interpolation operator cannot be defined in the same way as before near the interface. In this section we discuss a particular choice of interpolation operators based on the fine grid operator. This is called operator-induced coarsening ([5] and [12]).

This problem does not exist for the smoothing and restriction operators. The smoothing step is completely determined by the coefficient matrix and the right hand side. Once they are computed, we can perform the smoothing step regardless of the shape of the domain and the grid structure. For the restriction part, we simply use the fine grid value for the coarse value at the same grid points.

Let $A^h u^h = f^h$ be the linear system for the fine mesh. For each grid point, we look at its nine-point stencil as shown in Figure 24. The equation for the unknown centered at x_5 is:

$$a_1 u_1 + a_2 u_2 + a_3 u_3 + a_4 u_4 + a_5 u_5 + a_6 u_6 + a_7 u_7 + a_8 u_8 + a_9 u_9 = f_5.$$

Our adaptive multigrid scheme is applied to the error (or the residual) which satisfies a similar equation:

$$a_1 e_1 + a_2 e_2 + a_3 e_3 + a_4 e_4 + a_5 e_5 + a_6 e_6 + a_7 e_7 + a_8 e_8 + a_9 e_9 = r_5 \approx 0. \tag{38}$$

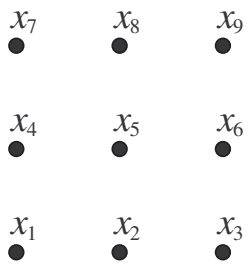

Figure 24: The nine-point stencil

Given the above equation and the values of the error e^{2h} at coarse mesh points, we need to interpolate $e^h = I_{2h}^h e^{2h}$ at each fine grid point.

First we consider the case where x_5 is a regular point, that is, all the nine points are in Ω^+. Assume that x_5 is also a coarse grid point, then we can simply take the coarse grid error at this point to be the "interpolated" fine value.

Now we consider the case where x_5 is on a vertical cell edge, that is, x_2 and x_8 are the coarse grid points. We want to express the error as

$$e_5 = c_2 e_2 + c_8 e_8,$$

where c_2 and c_8 are coefficients yet to be determined. Since only e_2, e_5 and e_8 appear in this equation, we use Taylor expansion for all the nine points and express them in terms of these three:

$$e_1 \approx e_2, \qquad e_3 \approx e_2, \qquad e_4 \approx e_5, \qquad e_6 \approx e_5, \qquad e_9 \approx e_8.$$

Substituting into Eqn.(38) leads to

$$c_2 = -\frac{a_1 + a_2 + a_3}{a_4 + a_5 + a_6} \qquad \text{and} \qquad c_8 = -\frac{a_7 + a_8 + a_9}{a_4 + a_5 + a_6}.$$

Similarly, if x_5 is on a horizontal cell edge with coarse grid points x_4 and x_6, we can interpolate the error on the fine mesh as

$$e_5 = c_4 e_4 + c_6 e_6,$$

where

$$c_4 = -\frac{a_1 + a_4 + a_7}{a_2 + a_5 + a_8} \qquad \text{and} \qquad c_6 = -\frac{a_3 + a_6 + a_9}{a_2 + a_5 + a_8}.$$

If x_5 is the center of a cell, we first compute all the other eight error values and then substitute them into Eqn.(38) to compute e_5.

If x_5 is an irregular point, it may have some coarse grid neighbors on the other side of the interface, and they cannot be used to interpolate for the value of e_5. Since we have the boundary condition as given in (23), we can simply assume that the error satisfies the homogeneous version of the condition

$$\beta \frac{\partial e}{\partial n} + \alpha e = 0 \qquad \text{on} \qquad \Gamma = \partial\Omega(t)^+,$$

or even simpler, that the discrete error on the other side of the interface is 0. Then we slightly modify the interpolation operator for regular points, with all the coefficients corresponding to points in Ω^- set to 0, to get the interpolation operator for irregular points. For example, in Figure 25's case, we have

$$e_5 = c_2 e_2 + c_8 e_8,$$

where

$$c_2 = -\frac{a_2 + a_3}{a_5 + a_6} \qquad \text{and} \qquad c_8 = -\frac{a_8 + a_9}{a_5 + a_6}.$$

a_1, a_4 and a_7 are gone since they correspond to points in Ω^-.

This way of treating irregular points works for our problem, where the boundary condition is in the form of (23). For problems with jump conditions, this are somewhat different (see [4]).

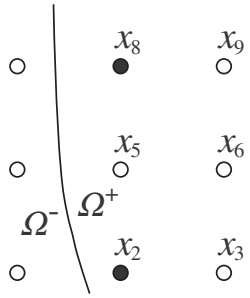

Figure 25: Interpolation from coarse grid points (x_2 and x_8) to fine grid point (x_5)

5.4 Convergence

The underlying steps we use to update the solution in the multigrid method is the Gauss-Seidel iteration. A necessary and sufficient condition for Gauss-Seidel iteration to converge for a specific class of matrices is given in [15]: if the matrix has a sign structure such that in each line, the sign of the diagonal element is opposite to the sign of all other elements, then a criterion called the generalized line criterion (GLC) is necessary and sufficient for Gauss-Seidel iteration to converge. We can verify numerically that for some steps, especially when the curvature gets very large at some point, the GLC is not satisfied. In that case, we switch to a direct solver (SuperLU [13], for example). We start with the iterative method and record the relative magnitude of the update for the solution. If it exceeds a given constant, we switch to the direct method.

6 Extension to Three-Dimensional Case

Most of the above discussion can be easily extended to three dimensions. However, certain aspects need to be treated carefully. First we need to to find a new local orthogonal coordinate system (ξ, η_1, η_2) centered at a given point x^* on the interface.

The second-order structure of a surface is characterized by a quadratic patch that shares first- and second-order contact with the surface at a point. The principal directions of the surface are those associated with the quadratic approximation, and the principal curvatures κ_1, κ_2 are the curvatures in those directions. The principal directions satisfy the properties that they are orthogonal to each other, and that they lie in the tangent plane of the surface at the point. Therefore, (ξ, η_1, η_2), where ξ is the normal direction and η_1, η_2 are the principal directions, is an orthogonal system and we can take it as our local coordinate system.

Assume that the surface is locally expressed as $\xi = \chi(\eta_1, \eta_2)$. Denote $\frac{\partial \chi}{\partial \eta_j} = \chi_j$ and $\frac{\partial^2 \chi}{\partial \eta_j \partial \eta_k} = \chi_{jk}$ for $j, k = 1, 2$. Then $\chi(0,0) = \chi_1(0,0) = \chi_2(0,0) = \chi_{12}(0,0) = 0$, and $\chi_{11}(0,0)$ and $\chi_{22}(0,0)$ are the principal curvatures due to our choice of the local coordinate system.

Similar to the two-dimensional case, we have the following ten relationships:

$$u^+ = u^-, \tag{39}$$

$$u^+_\xi \chi_j + u^+_{\eta_j} = u^-_\xi \chi_j + u^-_{\eta_j}, j = 1, 2, \tag{40}$$

$$\begin{aligned} &u^+_{\xi\xi}\chi_j\chi_k + u^+_{\xi\eta_k}\chi_j + u^+_\xi \chi_{jk} + u^+_{\xi\eta_j}\chi_k + u^+_{\eta_1\eta_2} \\ &\qquad = u^-_{\xi\xi}\chi_j\chi_k + u^-_{\xi\eta_k}\chi_j + u^-_\xi \chi_{jk} + u^-_{\xi\eta_j}\chi_k + u^-_{\eta_1\eta_2}, \\ &\qquad\qquad (j,k) = (1,1), (1,2), (2,2), \end{aligned} \tag{41}$$

$$u^+_{\xi\xi} + u^+_{\eta_1\eta_1} + u^+_{\eta_2\eta_2} = u^-_{\xi\xi} + u^-_{\eta_1\eta_1} + u^-_{\eta_2\eta_2}, \tag{42}$$

$$u^+_\xi - u^+_{\eta_1}\chi_1 - u^+_{\eta_2}\chi_2 + \frac{\alpha}{\beta}u^+ = \frac{(1+\chi_1^2+\chi_2^2)^{1/2} g}{\beta}, \tag{43}$$

$$u^+_{\xi\xi}\chi_j + u^+_{\xi\eta_j} - u^+_{\xi\eta_1}\chi_1\chi_j - u^+_{\eta_1\eta_j}\chi_1 - u^+_{\eta_1}\chi_{1j} - u^+_{\xi\eta_2}\chi_2\chi_j$$

$$-u^+_{\eta_2\eta_j}\chi_2 - u^+_{\eta_2}\chi_{2j} + \left(\frac{\alpha}{\beta}\right)_j u^+ + \frac{\alpha}{\beta}(u^+_\xi\chi_j + u^+_{\eta_j})$$
$$= \frac{\partial}{\partial\eta_j}\left\{(1+\chi_1^2+\chi_2^2)^{1/2}\frac{g}{\beta}\right\}, j = 1, 2. \tag{44}$$

Eqns. (40) - (43) are derived from continuity across the interface of the quantities u, $u_\xi\chi_j + u_{\eta_j}$, $u_{\xi\xi}\chi_j\chi_k + u_{\xi\eta_k}\chi_j + u_\xi\chi_{jk} + u_{\xi\eta_j}\chi_k + u_{\eta_j\eta_k}$ and $u_{\xi\xi} + u_{\eta_1\eta_1} + u_{\eta_2\eta_2}$, and the rest are derived from the boundary condition.

These equations are evaluated at $\eta_1 = \eta_2 = 0$. Since we have chosen a local coordinate system such that $\chi(0,0) = \chi_1(0,0) = \chi_2(0,0) = \chi_{12}(0,0) = 0$, they can be simplified to the matrix form

$$A^+U^+ = A^-U^- + G, \tag{45}$$

where

$$A^+ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \chi_{11} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \chi_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \frac{\alpha}{\beta} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \left(\frac{\alpha}{\beta}\right)_1 & 0 & \frac{\alpha}{\beta} - \chi_{11} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \left(\frac{\alpha}{\beta}\right)_2 & 0 & 0 & \frac{\alpha}{\beta} - \chi_{22} & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$A^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \chi_{11} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \chi_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$U = (u, u_\xi, u_{\eta_1}, u_{\eta_2}, u_{\xi\xi}, u_{\xi\eta_1}, u_{\eta_1\eta_1}, u_{\xi\eta_2}, u_{\eta_2\eta_2}, u_{\eta_1\eta_2})^T$$

and

$$G = \left(0, 0, 0, 0, 0, 0, 0, \frac{g}{\beta}, \left(\frac{g}{\beta}\right)_1, \left(\frac{g}{\beta}\right)_2\right)^T.$$

To explicitly find out the values of the local coordinate system, note that the second-order structure of the surface can be computed from the first- and second-order structure of the level set function ϕ. All of the shape information is contained in the field of normals given by $\boldsymbol{n}(\boldsymbol{x}) = (\phi_x, \phi_y, \phi_z)/|\nabla\phi|$.

To compute the principal curvatures, we compute the mean curvature and Gaussian curvature first. The mean curvature is

$$\begin{aligned} H &= \frac{\kappa_1 + \kappa_2}{2} \\ &= \frac{\phi_{xx}(\phi_y^2+\phi_z^2) + \phi_{yy}(\phi_x^2+\phi_z^2) + \phi_{zz}(\phi_x^2+\phi_y^2) - 2(\phi_x\phi_y\phi_{xy} + \phi_y\phi_z\phi_{yz} + \phi_x\phi_z\phi_{xz})}{2(\phi_x^2+\phi_y^2+\phi_z^2)^{2/3}}, \end{aligned}$$

and the Gaussian curvature is given by

$$\begin{aligned} K &= \kappa_1 \cdot \kappa_2 \\ &= \frac{\phi_x^2(\phi_{yy}\phi_{zz} - \phi_{yz}^2) + \phi_y^2(\phi_{xx}\phi_{zz} - \phi_{xz}^2) + \phi_z^2(\phi_{xx}\phi_{yy} - \phi_{xy}^2)}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^2} + \\ &\quad \frac{2[\phi_x\phi_y(\phi_{xz}\phi_{yz} - \phi_{xy}\phi_{zz}) + \phi_y\phi_z(\phi_{xy}\phi_{xz} - \phi_{yz}\phi_{xx}) + \phi_x\phi_z(\phi_{xy}\phi_{yz} - \phi_{xz}\phi_{yy})]}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^2}, \end{aligned}$$

The principal curvatures $\kappa_i, i = 1, 2$ can be computed as

$$\kappa_1 = H + \sqrt{H^2 - G}, \qquad \kappa_2 = H - \sqrt{H^2 - G}.$$

The 3×3 matrix of derivatives of $\boldsymbol{n}(\boldsymbol{x})$ is

$$N = -[\boldsymbol{n}_x\ \boldsymbol{n}_y\ \boldsymbol{n}_z].$$

The projection of N onto the tangent plane of the surface gives the shape matrix S. Let P be the normal projection operator, then

$$P = \boldsymbol{n} \otimes \boldsymbol{n} = \frac{1}{|\nabla\phi|^2} \begin{bmatrix} \phi_x^2 & \phi_x\phi_y & \phi_x\phi_z \\ \phi_x\phi_y & \phi_y^2 & \phi_y\phi_z \\ \phi_x\phi_z & \phi_y\phi_z & \phi_z^2 \end{bmatrix}.$$

The tangential projection operator is $T = I - P$, and the shape matrix is given by

$$S = NT.$$

The shape matrix S is singular and it has three real eigenvalues with three corresponding orthogonal eigenvectors. The eigenvector corresponding to the zero eigenvalue is the normal direction. The two eigenvectors η_1 and η_2 corresponding to the nonzero eigenvalues are the principal directions.

A very complicated calculation gives for $j = 1, 2$:

$$\begin{aligned} \eta_{j,1} &= \phi_x\phi_y^2\phi_{yz} - \phi_x\phi_y\phi_z\phi_{yy} + \phi_x\phi_y\phi_z\phi_{zz} - \phi_x\phi_z^2\phi_{yz} - \phi_y^3\phi_{xz} + \phi_y^2\phi_z\phi_{xy} - \phi_y\phi_z^2\phi_{xz} + \phi_z^3\phi_{xy} \\ \eta_{j,2} &= -\phi_x^2\phi_y\phi_{yz} - \phi_x^2\phi_z\phi_{zz} + \phi_x\phi_y^2\phi_{xz} + \phi_x\phi_y\phi_z\phi_{xy} + 2\phi_x\phi_z^2\phi_{xz} - \phi_y^2\phi_z\phi_{xx} - \phi_z^3\phi_x x \\ &\quad + \frac{w_j}{2v_3}(\phi_x^2\phi_z + \phi_y^2\phi_z + \phi_z^3) \\ \eta_{j,3} &= \phi_x^2\phi_z\phi_{yz} + \phi_x^2\phi_y\phi_{yy} - \phi_x\phi_z^2\phi_{xy} - \phi_x\phi_y\phi_z\phi_{xz} - 2\phi_x\phi_y^2\phi_{xy} + \phi_y\phi_z^2\phi_{xx} + \phi_y^3\phi_{xx} \\ &\quad - \frac{w_j}{2v_3}(\phi_x\phi_x\phi_y + \phi_y^3 + \phi_y\phi_z^2), \end{aligned}$$

where

$$\begin{aligned} v_1 &= [\phi_x^4(\phi_{yy} - \phi_{zz})^2 + \phi_y^4(\phi_{zz} - \phi_{xx})^2 + \phi_z^4(\phi_{xx} - \phi_{yy})^2 + \\ &\quad \phi_x^2(\phi_{yy}\phi_z - 2\phi_y\phi_{yz})^2 + \phi_x^2(\phi_{zz}\phi_y - 2\phi_z\phi_{yz})^2 + \\ &\quad \phi_y^2(\phi_{zz}\phi_x - 2\phi_z\phi_{xz})^2 + \phi_y^2(\phi_{xx}\phi_z - 2\phi_x\phi_{xz})^2 + \\ &\quad \phi_z^2(\phi_{xx}\phi_y - 2\phi_x\phi_{xy})^2 + \phi_z^2(\phi_{yy}\phi_x - 2\phi_y\phi_{xy})^2 + \\ &\quad 4\phi_y^2\phi_z^2\phi_{yz}^2 + 4\phi_x^2\phi_y^2\phi_{xy}^2 + 4\phi_x^2\phi_z^2\phi_{xz}^2 + \\ &\quad 8\phi_x\phi_y^2\phi_z\phi_{yy}\phi_{xz} + 8\phi_x\phi_y\phi_z^2\phi_{zz}\phi_{xy} + 8\phi_x^2\phi_y\phi_z\phi_{xx}\phi_{yz} + \\ &\quad 8\phi_y^3\phi_{xz}(-\phi_z\phi_{xy} - \phi_x\phi_{yz}) + 8\phi_z^3\phi_{xy}(-\phi_x\phi_{yz} - \phi_y\phi_{xz}) + 8\phi_x^3\phi_{yz}(-\phi_y\phi_{xz} - \phi_z\phi_{xy}) + \\ &\quad 4\phi_z^4\phi_{xy}^2 + 4\phi_y^4\phi_{xz}^2 + 4\phi_x^4\phi_{yz}^2 + 2\phi_{xx}\phi_{yy}(\phi_x^2\phi_y^2 - \phi_y^2\phi_z^2 - \phi_x^2\phi_z^2) + \\ &\quad 2\phi_{xx}\phi_{zz}(\phi_x^2\phi_z^2 - \phi_x^2\phi_y^2 - \phi_y^2\phi_z^2) + 2\phi_{yy}\phi_{zz}(\phi_y^2\phi_z^2 - \phi_x^2\phi_z^2 - \phi_x^2\phi_y^2) + \\ &\quad 4\phi_x^3(\phi_y\phi_{zz}\phi_{xy} + \phi_z\phi_{yy}\phi_{xz} - \phi_y\phi_{yy}\phi_{xy} - \phi_z\phi_{zz}\phi_{xz}) + \end{aligned}$$

$$
\begin{aligned}
& 4\phi_y^3(\phi_z\phi_{xx}\phi_{yz} + \phi_x\phi_{zz}\phi_{xy} - \phi_z\phi_{zz}\phi_{yz} - \phi_x\phi_{xx}\phi_{xy}) + \\
& 4\phi_z^3(\phi_x\phi_{yy}\phi_{xz} + \phi_y\phi_{xx}\phi_{yz} - \phi_x\phi_{xx}\phi_{xz} - \phi_y\phi_{yy}\phi_{yz})]^{1/2}, \\
v_2 &= \phi_{xx}(\phi_y^2+\phi_z^2) + \phi_{yy}(\phi_x^2+\phi_z^2) + \phi_{zz}(\phi_x^2+\phi_y^2) - 2(\phi_x\phi_y\phi_{xy} + \phi_y\phi_z\phi_{yz} + \phi_x\phi_z\phi_{xz}), \\
v_3 &= \phi_x^2+\phi_y^2+\phi_z^2, \\
w_{1,2} &= v_2 \pm v_1.
\end{aligned}
$$

These three eigenvectors are normalized to give the local coordinate system.

With the above choice of coordinates, let

$$
Z_k = \left(1, \xi, \eta_1, \eta_2, \frac{1}{2}\xi^2, \xi\eta_1, \frac{1}{2}\eta_1^2, \xi\eta_2, \frac{1}{2}\eta_2^2, \eta_1\eta_2\right)_k^T,
$$

then

$$
\begin{aligned}
C &= \sum_{k\in K^+} \gamma_k Z_k^T (A^+)^{-1} G \\
&= \sum_{k\in K^+} \gamma_k \left\{ \frac{g}{\beta}\xi + \frac{1}{2}\frac{g}{\beta}(\chi_{11}+\chi_{22})\xi^2 + \left(\frac{g}{\beta}\right)_1 \xi\eta_1 - \frac{\chi_{11}}{2}\frac{g}{\beta}\eta_1^2 + \left(\frac{g}{\beta}\right)_2 \xi\eta_2 - \frac{\chi_{22}}{2}\frac{g}{\beta}\eta_2^2 \right\}_k .
\end{aligned}
$$

Finally, with

$$
\begin{aligned}
W &= (0,0,0,0,1,0,1,0,1,0)^T, \\
W &= \sum_{k\in K^+} \gamma_k \left[(A^+)^{-1}A^-\right]^T Z_k + \sum_{k\in K^-} \gamma_k Z_k
\end{aligned} \tag{46}
$$

is our system of equations for determining the coefficients γ_k. Like what we do in two dimensions, we can apply a linear transformation to the set of equations and get rid of the terms with $\gamma_k, k \in K^-$. We premultiply Eqn. (46) with the matrix

$$
T = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & \chi_{11}+\chi_{22} & 0 & -\chi_{11} & 0 & -\chi_{22} & 0
\end{bmatrix},
$$

and assume that we are able to find ten neighbors out of the 27 neighbors, among which exactly three of them are exterior points. Now we are left with a 7×7 system to solve:

$$
\begin{aligned}
\textstyle\sum_{k\in K^+} \quad & \gamma_k \left\{ 1 - \frac{\alpha}{\beta}\xi - \frac{1}{2}\frac{\alpha}{\beta}(\chi_{11}+\chi_{22})\xi^2 - \left(\frac{\alpha}{\beta}\right)_1 \xi\eta_1 + \frac{1}{2}\frac{\alpha}{\beta}\chi_{11}\eta_1^2 \right. \\
& \left. - \left(\frac{\alpha}{\beta}\right)_2 \xi\eta_2 + \frac{1}{2}\frac{\alpha}{\beta}\chi_{22}\eta_2^2 \right\} = 0, \\
\textstyle\sum_{k\in K^+} \quad & \gamma_k \left\{ \eta_1 + \xi\eta_1 \left(\chi_{11} - \frac{\alpha}{\beta}\right) \right\} = 0, \\
\textstyle\sum_{k\in K^+} \quad & \gamma_k \left\{ \eta_2 + \xi\eta_2 \left(\chi_{22} - \frac{\alpha}{\beta}\right) \right\} = 0, \\
\textstyle\sum_{k\in K^+} \quad & \gamma_k (\xi^2)_k = 2, \\
\textstyle\sum_{k\in K^+} \quad & \gamma_k (\eta_1^2)_k = 2,
\end{aligned}
$$

$$\sum_{k\in K^+} \quad \gamma_k(\eta_2^2)_k = 2,$$
$$\sum_{k\in K^+} \quad \gamma_k(\eta_1\eta_2)_k = 0.$$

We can do everything else in exactly the same way as what we do in two dimensions.

7 Numerical Results

With the numerical methods discussed in previous sections, we are able to solve each of the equations involved in the modeling of the electrodeposition process. In this section, we will first talk about some accuracy issues associated with solving our differential equations sequentially, followed by some numerical results. The last part gives some related future research directions.

7.1 Time-Dependent Boundary Conditions and Time Splitting

Most of the numerical methods under discussion are second-order accurate for simple test problems. In general, however, this may not be true for more complicated problems like the ones given in Section 3.

One difficulty lies in the fact that these differential equations usually depend on each other in complex ways. For example, the speed of propagation of the interface depends on both the accelerator coverage θ and the concentration of copper C_c as shown in Eqn. (7), while Eqns. (11) and (12) for θ depend on the speed and the concentration of the accelerator C_m.

To illustrate, we consider the step from time t to time $t + \Delta t$. Regardless of which equation we solve first, we will need the values of some of the other variables at time $t + \Delta t$ to make it second-order in space and time, and these are not yet available.

Another difficulty is that the interface is moving, and hence carrying boundary conditions. As an example, consider the diffusion equation. The mixed boundary conditions hold along the interface, which moves from time t to time $t+\Delta t$. Thus we need to do something special to make it second-order accurate in time. One way to solve this problem is to use the method of Twizell, Gumel, and Arigu (see [47]) for time-discretization, which solves the moving boundary problem by solving a sequence of fixed boundary problems.

To illustrate, consider the diffusion equation

$$\begin{aligned} u_t &= \Delta u \qquad \text{in} \quad \Omega(t) \\ \frac{\partial u}{\partial n} + \alpha(t)u &= g(t) \qquad \text{on} \quad \partial\Omega(t). \end{aligned}$$

Let the time-dependent operator $L^h(t)$ be such that $L^h(t)u$ is the discretization of the term Δu discretized as in section 4.3. The TGA method splits the time step Δt as

$$\Delta t = \mu_1 + \mu_2 + \mu_3.$$

For this numerical scheme to be second-order and L_0-stable (see [47]), a value a is picked such that $a \in [\frac{1}{2}, 2 - \sqrt{2}]$ and we set

$$\mu_1 = \frac{a - \sqrt{a^2 - 4a + 2}}{2}\Delta t, \quad \mu_2 = \frac{a + \sqrt{a^2 - 4a + 2}}{2}\Delta t, \quad \mu_3 = (1 - a)\Delta t.$$

The numerical solution u^n is then updated as

$$u^{n+1} = \left(I - \mu_1 L^h(t_{new})\right)^{-1} \left(I - \mu_2 L^h(t_{int})\right)^{-1} \left(I + \mu_3 L^h(t_{old})\right) u^n, \tag{47}$$

where

$$t_{old} = t^n, \quad t_{new} = t^n + \Delta t = t^{n+1}, \quad t_{int} = t^{n+1} - \mu_1 = t^n + \mu_2 + \mu_3.$$

Specifically, the discretization (47) is done on $\Omega(t_{new})$ at each step. The boundary conditions on the fixed boundary $\partial\Omega(t_{new})$ are computed by interpolating values from the moving interface at time t_{old}. We use the second-order

extension from Section 4.3.5 for this purpose, and this leads to an overall second-order accurate method for diffusion equations defined on a region with moving interface.

For our deposition problem, it is not really necessary to use a second-order scheme. A first-order scheme is enough for us to capture the superfilling phenomena. Thus in our implementation, for each sub-problem, we solve it using a second-order scheme. But we will not deal with the first difficulty mentioned above in any special way, leading to an overall first-order scheme. At each time step t^n, we update the variables sequentially. If some other variables are needed during the updating process, we will use their values at t^{n+1} only if they are available. Otherwise, we will use the values evaluated at time t^n.

7.2 Numerical Results in Two Dimensions

First we test our algorithm on a trench with different values of initial coverage and accelerator concentration. The trench has size $0.5\mu m \times 1.1\mu m$ with a width-depth ratio of about $1:2$. Experiments show that derivatization for 30 seconds in the electrolyte with 0.5, 5, 50, 500 and 1000 μ mol/L accelerator yields initial fractional catalyst coverage of approximately 0.00054, 0.0054, 0.054, 0.44 and 0.88 respectively (see [34]). These are taken as the initial conditions for our initial-boundary value problem.

We use a mesh with mesh size $h = 1.5 \cdot 10^{-8}$, and dt is computed based on the value of h and normal speed of propagation of the interface v so that the CFL stability condition is satisfied.

Experimental results under the same configurations are shown in Figure 26. Our numerical results shown in Figure 27 conform with the experimental results very well. We see that with an initial catalyst coverage of 0.00054, the deposition is predicted to be conformal because the geometric leveling is associated with the sloping sidewalls. In real experiments, the trench to be filled may not be smooth. We add some randomness to the initial shape and can observe the formation of void in this case as shown in Figure 28. Increasing the initial catalyst coverage an order of magnitude results in superfilling behavior: enrichment of the catalyst begins on the bottom of the corners, leading to significant acceleration of the copper deposition rate. Increasing the initial catalyst coverage even more to 0.054 results in near optimal superfilling behavior, which leads to the change in convexity on the bottom. Increasing the initial catalyst coverage to 0.44 or 0.88 is predicted to result in failure to superfill the trench. Catalyst enrichment on the advancing concave corners approaches unity, and a V-notch geometry is established. The CEAC mechanism fails to fill the feature because the coverage on the bottom is not much more than coverage on the sidewalls due to the near-saturation coverage on the concave corners.

If we measure the CPU time it takes for each time step, we can see that the multigrid approach to solving linear systems is noticeably faster than using a direct solver. Since for each time step, all the equations are solved using explicit methods except for the diffusion equation, we will just record the CPU time for the whole procedure instead of the time for solving the linear system only.

On a 3.4GHz CPU machine, when the finest grid is of size 96×160 with a total number of 4 levels, the multigrid method (if it converges) for solving the linear system takes only less than 0.3s. In contrast, the SuperLU method takes about 1.5s. While the difference may not seem to be big in this case, consider the situation where the accuracy requires us to use finer mesh, say 192×320 with 5 levels. Now multigrid takes about 1.2s to solve the linear system and SuperLU takes about 20s. Thus, the time it takes for the multigrid method to solve such a linear system increase much slower than a direct method like SuperLU as the mesh gets finer. Moreover it also makes efficient use of memory, since we use the finest mesh only for the part of the domain that is close to the interface.

To estimate the order of accuracy of the overall scheme, we compute the numerical solutions for several different mesh sizes, which correspond to different maximum number of levels. As we do not know the exact solution to this problem, we compare all the numerical solutions to the one with a maximum number of 6 levels (with mesh size 385×641) and the result is shown in Figure 29. For error, we use the L-infinity error in the concentration. The absolute value of the slope, which is an approximation to the order of accuray of the overall method, is 0.61.

7.3 Numerical Results in Three Dimensions

We test the algorithm on a three-dimensional trench with the size $0.5\mu m \times 0.5\mu m \times 1.1\mu m$ for two different cases with 5μ mol/L and 500μ mol/L accelerator in the electrolyte respectively.

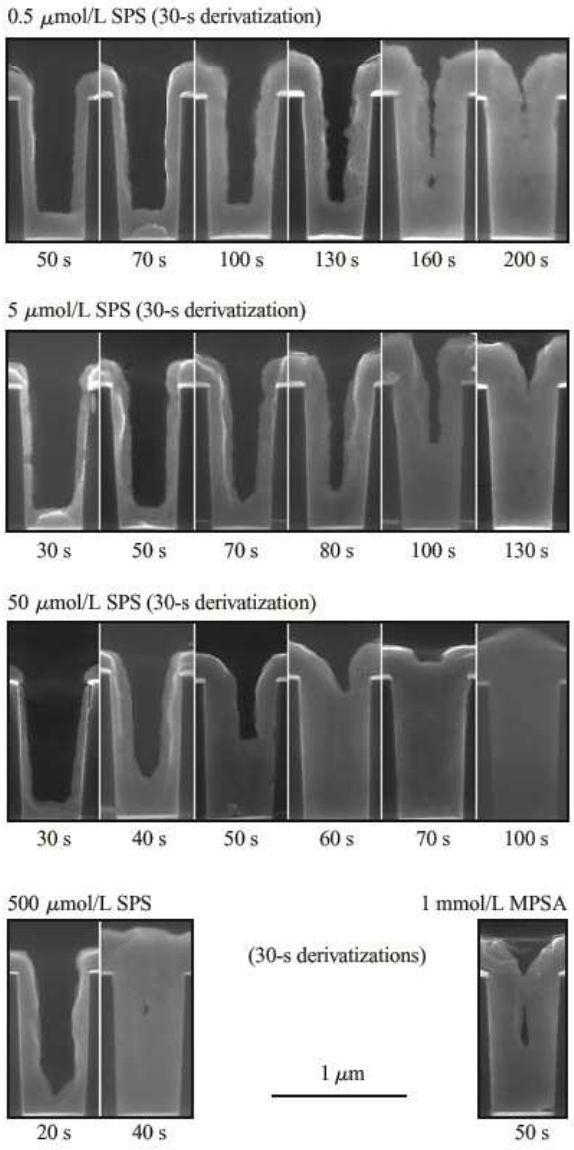

Figure 26: Experimental results for different initial catalyst coverages. The picture is taken from [34]

The numerical results are shown in Figure 30 and Figure 31. We can see from Figure 27 and Figure 30 that in two dimensions, we predict the phenomena of superfilling, while in the corresponding three dimensional case, we get a void.

One possible explanation for this void is an analogy with a collapsing dumbbell evolving under mean curvature flow: it is well-known that this can split into multiple objects (see [10]). Though the speed of propagation for the interface does not depend on the mean curvature explicitly, it is a function of the additive coverage θ, which evolves under a conservation law. The key issue is that this conservation law involves a mean curvature term if we rewrite it by expanding out each term.

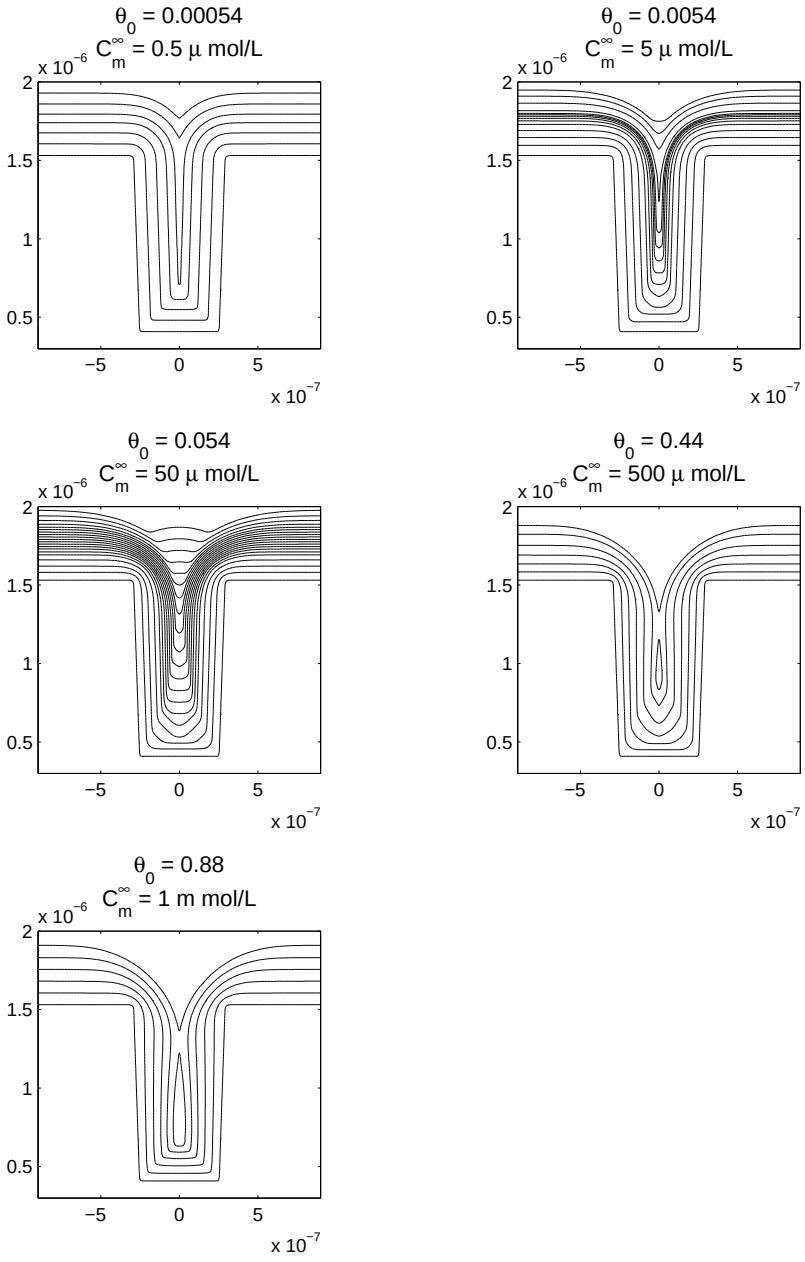

Figure 27: Numerical results for different initial catalyst coverages

7.4 Future Research Directions

The results of this article suggest some future research topics. First, the framework we have presented here can be extended to a wide range of physical applications involving moving boundaries coupled with partial differential

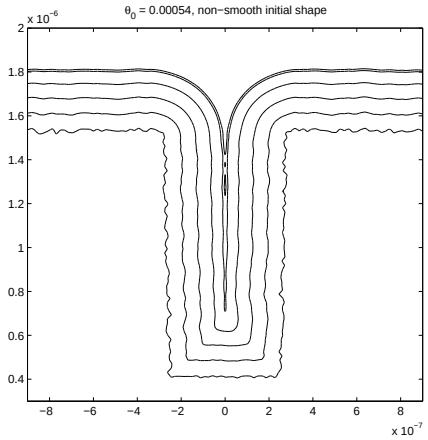

Figure 28: Formation of void in a non-smooth configuration

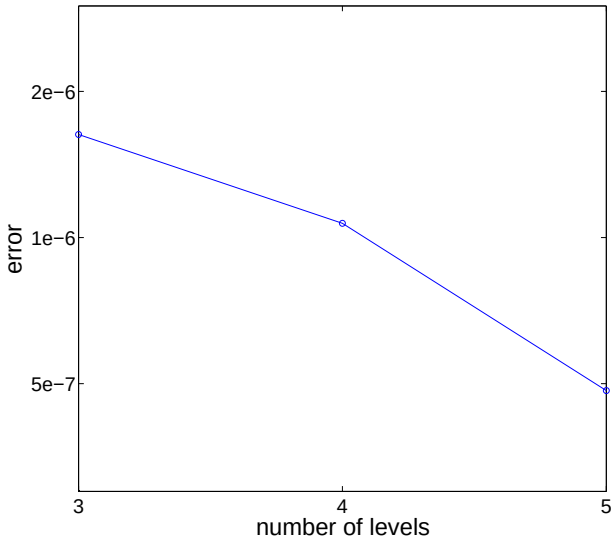

Figure 29: Errors of solutions with different mesh sizes: L-infinity error on the concentration

equations, for example, the hyperbolic conservation laws and the diffusion equation in particular.

An open question is how to solve the linear systems corresponding to the discretizations of diffusion equations, when the interface has very large local curvatures, or spikes. We may rewrite local equations, choose appropriate stencils, use non-rectangular grid cells or there may exist other ways for the multigrid algorithm to converge for problems with bad geometries.

To model the process more realistically, we can add some noise terms which have physical meanings instead of just by adding random numbers to the initial configuration, as what we did in our of our tests (Figure 28). This may give a better prediction to what initial configurations lead to superfilling.

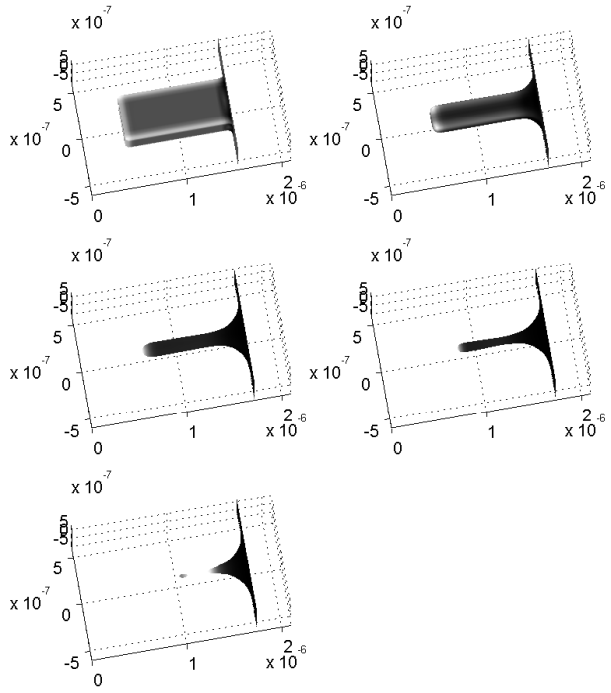

Figure 30: Numerical results for 5μ mol/L accelerator and thus 0.0054 initial fractional coverage at $t = 32s, 50s, 58s, 62s$ and $68s$

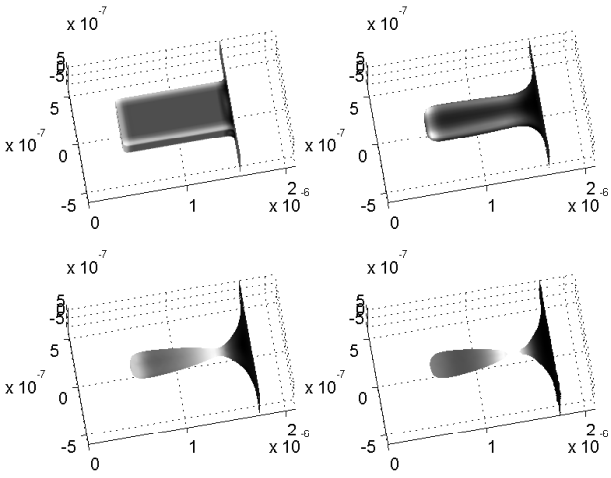

Figure 31: Numerical results for 500μ mol/L accelerator and thus 0.44 initial fractional coverage at $t = 1s, 3s, 6s$ and $9s$

Acknowledgement

We would like to thank Daniel Josell for help with modeling the electrodeposition process, Beresford Parlett for discussions concerning the linear system solver, and Jon Wilkening for discussions on the linear programming approach.

References

[1] D. Adalsteinsson, and J. A. Sethian, *A Fast Level Set Method for Propagating Interfaces*, J. Comp. Phys., 118, 2, pp. 269 – 277, 1995.

[2] D. Adalsteinsson, and J. A. Sethian, *The Fast Construction of Extension Velocities in Level Set Methods*, J. Comp. Phys., 148, 1, pp. 2 – 22, 1999.

[3] D. Adalsteinsson, and J. A. Sethian, *Transport and Diffusion of Material Quantities on Propagating Interfaces via Level Set Methods*, J. Comp. Phys., 185, 1, pp. 271 – 288, 2002.

[4] Loyce Adams, and Zhilin Li, *The Immersed Interface/Multigrid Methods for Interface Problems*, SIAM J. Sci. Comput., 24, 2, pp. 463 – 479, 2002.

[5] R. E. Alcouffe, A. Brandt, J. E. Dendy Jr., and J. W. Painter, *The Multi-grid Method for the Diffusion Equation with Strongly Discontinuous Coefficients*, SIAM J. Sci. Comput., 2, 4, pp. 430 – 454, 1981.

[6] P. C. Andricacos, C. Uzoh, J. O. Dukovic, J. Horkans, and H. Deligianni, *Damascene Copper Electroplating for Chip Interconnections*, IBM J. Res. Dev., 42, 5, pp. 567 – 574, 1998.

[7] Tariq D. Aslam, *A Partial Differential Equation Approach to Multidimensional Extrapolation* J. Comp. Phys., 193, 1, pp. 349 – 355, 2004.

[8] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1994.

[9] William L. Briggs, Van Emden Henson, and Steve F. McCormick, *A Multigrid Tutorial*, SIAM, 2000.

[10] David L. Chopp, and James A. Sethian, *Flow under Curvature: Singularity Formation, Minimal Surfaces, and Geodesics*, Exp. Math., 2, 4, pp. 235 – 255, 1993.

[11] T. A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM, 2006.

[12] J. E. Dendy Jr., *Black Box Multigrid*, J. Comput. Phys., 48, 3, pp. 366 – 386, 1982.

[13] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu, *A Supernodal Approach to Sparse Partial Pivoting*, SIAM J. Matrix Anal. Appl., 20, 3, pp. 720 – 755, 1999.

[14] L. C. Evans, *Partial Differential Equations*, AMS, 2002.

[15] M. V. P. Garcia, C. Humes Jr., and J. M. Stern, *Generalized Line Criterion for Gauss-Seidel Method*, Mat. Apl. Comput., 22, 1, pp. 91 – 97, 2003.

[16] Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, Stanley Osher, *A Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method)*, J. Comput. Phys., 152, 2, pp. 457 – 492, 1999.

[17] G. H. Golub, and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1989.

[18] H. Holden, and N. H. Risebro, *Front Tracking for Hyperbolic Conservation Laws*, Springer, 2000.

[19] T. Hou, Z. Li, S. Osher, and H.-K. Zhao, *A Hybrid Method for Moving Interface Problems with Applications to the Hele-Shaw Flow*, J. Comp. Phys., 134, 2, pp. 236 – 252, 1997.

[20] J. K. Hunter, Z. Li, and H. K. Zhao, *Reactive Autophobic Spreading of Drops*, J. Comp. Phys., 183, 2, pp. 335 – 366.

[21] Hans Johansen, and Phillip Colella, *A Cartesian Grid Embedded Boundary Method for Poisson's Equation on Irregular Domains*, J. Comp. Phys., 147, 1, pp. 60 – 85, 1998.

[22] D. Josell, D. Wheeler, W. H. Huber, and T. P. Moffat, *Superconformal Electrodeposition in Submicron Features*, Phys. Rev. Lett., 87, 1, 016102, 2001.

[23] D. Josell, D. Wheeler, W. H. Huber, J. E. Bonevich, and T. P. Moffat, *A Simple Equation for Predicting Superconformal Electrodeposition in Submicronmeter Trenches*, J. ElecChem. Soc., 148, 12, C767 – C773, 2001.

[24] Randall J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, 1990.

[25] Randall J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, 2002.

[26] Randall J. LeVeque, and Zhilin Li, *The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources*, SIAM J. Numer. Anal., 31, 4, pp. 1019 – 1044, 1994.

[27] Zhilin Li, *The Immersed Interface Method: A Numerical Approach for Partial Differential Equations with Interfaces*, PhD Dissertation, University of Washington, 1995.

[28] Zhilin Li, and Kazufumi Ito, *Maximum Principal Preserving Schemes for Interface Problems with Discontinuous Coefficients*, SIAM J. Sci. Comput., 23, 1, pp. 339 – 361.

[29] Zhilin Li, and Kazufumi Ito, *The Immersed Interface Method – Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*, SIAM Frontiers in Applied mathematics, 33, 2006.

[30] Z. Li, H. Zhao, and H. Gao, *A Numerical Study of Electro-migration Voiding by Evolving Level Set Functions on a Fixed Cartesian Grid*, J. Comp. Phys. 152, 1, pp. 281 – 304, 1999.

[31] Stephen F. McCormick, *Multilevel Adaptive Methods for PDEs*, SIAM, 2002.

[32] P. McCorquodale, P. Colella, and H. Johansen, *A Cartesian Grid Embedded Boundary Method for the Heat Equation on Irregular Domains*, J. Comp. Phys., 173, 2, pp. 620 – 635, 2001.

[33] T. P. Moffat, J. E. Bonevich, W. H. Huber, A. Stanishevsky, D. R. Kelly, G. R. Stafford, and D. Josell, *Superconformal Electrodeposition of Copper in 500–90 nm Features*, J. Electrochem. Soc., 147, 12, pp. 4524 – 4535, 2000.

[34] T. P. Moffat, D. Wheeler, M. D. Edelstein, and D. Josell, *Superconformal Film Growth: Mechanism and Quantification*, IBM J. Res. Dev., 49, 1, pp. 19 – 36, 2005.

[35] S. Osher, and J. A. Sethian, *Fronts Propagating with Curvature–Dependent Speed: Algorithms Based on Hamilton–Jacobi Formulations*, J. Comp. Phys., 79, 1, pp. 12 – 49, 1988.

[36] Ulrich Rüde, *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, SIAM, 1993.

[37] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[38] Y. Saad, and M. H. Schultz, *GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput., 7, 3, pp. 856 – 869, 1986.

[39] J. A. Sethian, *An Analysis of Flame Propagation*, PhD Dissertation, University of California, Berkeley, 1982.

[40] J. A. Sethian, *Curvature and the Evolution of Fronts*, Comm. in Math. Phys., 101, pp. 487 – 499, 1985.

[41] J. A. Sethian, *Numerical Methods for Propagating Fronts*, Variational Methods for Free Surface Interfaces, Proceedings of the Sept, 1985 Vallambrosa Conference, Eds. P. Concus and R. Finn, Springer-Verlag, NY, 1987.

[42] J. A. Sethian, *A Fast Marching Level Set Method for Monotonically Advancing Fronts*, Proc. Nat. Acad. Sci., 93, 4, pp. 1591 – 1595, 1996.

[43] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences*, Cambridge University Press, 1999.

[44] John Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, SIAM, 2004.

[45] N. Sukumar, D. L. Chopp, N. Moës, and T. Belytschkoc *Modeling holes and inclusions by level sets in the extended finite-element method*, Comput. Methods Appl. Mech. Engrg., 190, pp. 6183 – 6200, 2001.

[46] J. W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*, Springer, 2004.

[47] E. H. Twizell, A. B. Gumel, and M. A. Arigu, *Second-order, L_0-stable Methods for the Heat Equation with Time-dependent Boundary Conditions*, Adv. Comput. Math., 6, pp. 333 – 352, 1996.

[48] Alan C. West, *Theory of Filling of High-Aspect Ratio Trenches and Vias in Presence of Additives*, J. Electrochem. Soc., 147, 1, pp. 227 – 232, 2000.

[49] D. Wheeler, D. Josell, and T. P. Moffat, *Modeling Superconformal Electrodeposition in Trenches*, The Eighth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, 2002.

[50] D. Wheeler, D. Josell, and T. P. Moffat, *Modeling Superconformal Electrodeposition Using the Level Set Method*, J. Electrochem. Soc., 150, 5, C302, 2003.

[51] Jon Wilkening, *Multigrid on an Irregular Domain*, unpublished, 1999.

[52] J.-J. Xu, and H.-K. Zhao, *An Eulerian Formulation for Solving Partial Differential Equations along a Moving Interface*, J. Sci. Comp., 19, pp. 573 – 594, 2003.