

Cartesian Grid Embedded Boundary Finite Difference Methods for Elliptic and Parabolic
Partial Differential Equations on Irregular Domains

by

Hans Svend Johansen

B.S. (University of California, Berkeley) 1991

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering—Mechanical Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY of CALIFORNIA, BERKELEY

Committee in charge:

Professor Phillip Colella, Chair

Professor Dorian Liepmann

Professor John Strain

Fall 1997

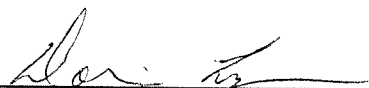
The dissertation of Hans Svend Johansen is approved:



12/8/97

Phillip Colella, Chair

Date



December 8, 1997

Dorian Liepmann

Date



97.12.08

John Strain

Date

The University of California, Berkeley

Fall 1997

© copyright 1997

by

Hans Svend Johansen

Abstract

Cartesian Grid Embedded Boundary Finite Difference Methods for Elliptic and Parabolic
Partial Differential Equations on Irregular Domains

by

Hans Svend Johansen

Doctor of Philosophy in Engineering–Mechanical Engineering

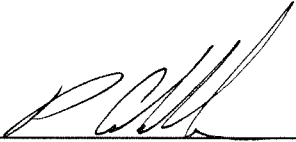
University of California, Berkeley

Professor Phillip Colella

This thesis presents a natural extension of the embedded boundary method, to include elliptic and parabolic partial differential equations defined on moving domains. Our motivation is to augment the algorithms available for hyperbolic problems, and provide the tools needed to approach a broader class of problems that include both heat transfer and phase change. This thesis attempts to overcome the major drawbacks of rectangular-grid finite-volume methods – lack of adaptivity and geometric flexibility – while retaining their simplicity and regular data structures.

To demonstrate that this is possible, three model problems with Dirichlet boundary conditions are discretized on time-dependent domains: the Poisson equation, the heat equation, and classical Stefan problem. In each case, a consistent finite-volume method is derived, by integrating the differential equation over the space-time volume, defined by the intersection of a piecewise-linear moving boundary and each cell of a Cartesian grid. The resulting surface integrals are discretized with single-point quadrature rules, that take advantage of geometric properties to cancel the leading-order error terms. The integrands

are then approximated to second-order, using finite-differences of the cell-centered solution, even when those cell-centers are outside the domain. Additional consideration is given to the conditioning of the resulting set of equations; gradients at the boundary are calculated using stencils that are well-separated from the partial cells. Although this yields a nonsymmetric linear system, we are able to combine point-relaxation with a multi-grid iteration strategy, to obtain residual reduction rates per iteration that are nearly independent of the grid spacing and the presence of arbitrarily small cells. This behavior persists, even when incorporated into an adaptive mesh hierarchy.



Phillip Colella, Chair

12/8/97

Date

Contents

Chapter List of Figures	iv
Chapter List of Tables	xii
Chapter 1 Introduction	1
1.1 Target Application: Bulk Crystal Growth	3
1.2 Motivation of the Numerical Investigation	6
1.3 Contributions to the Embedded Boundary Method	9
1.4 Literature Survey of Related Work	11
1.4.1 Finite Element Method	12
1.4.2 Finite Difference Methods	13
1.4.3 Front Capturing Methods	13
1.4.4 Boundary Integral Methods	14
1.4.5 Immersed Boundary Technique	15
1.4.6 Immersed Interface Method	15
Chapter 2 Poisson Equation	17
2.1 Introduction	17
2.2 One-Dimensional Algorithm	19
2.2.1 Discretization	19
2.2.2 Truncation Error Analysis	22
2.2.3 Discussion	25
2.3 Two-Dimensional Case	26
2.3.1 Discretization	27
2.3.2 Truncation Error Estimates	31
2.3.3 Neumann boundary conditions	33
2.4 A Multi-Grid Method for our Discretization	33
2.4.1 Point relaxation scheme	34
2.4.2 Coarse problem definition	34
2.5 Adaptive Mesh Refinement	35
2.5.1 Coarse-fine interface fluxes and interpolation stencils	36
2.5.2 Relationship to multi-grid iteration strategy	38
2.5.3 Refinement criterion	39

2.6	Software Implementation	39
2.6.1	Description of BoxLib	40
2.6.2	Description of AmrPoisson	41
2.6.3	Sparse data structures	41
2.6.4	Grid generation classes	42
2.6.5	Graphics libraries	42
2.7	Results	43
2.7.1	Problem 1: Comparison with an exact solution	45
2.7.2	Problem 2: Including variable coefficients	50
2.7.3	Problem 3: Grid Convergence Study	52
2.8	Conclusions	55
Chapter 3	Heat Equation	58
3.1	Introduction	58
3.2	Derivation in One Dimension	59
3.2.1	Finite volume derivation	60
3.2.2	Properties of update matrix	68
3.2.3	An implicit Runge-Kutta method	72
3.2.4	Hybrid method for heat equation	73
3.3	Results in One Dimension	76
3.3.1	Decay of a sinusoid	77
3.3.2	Decay of non-smooth initial conditions in 1D	79
3.4	Extension to Two Dimensions	81
3.4.1	Finite volume derivation in two dimensions	82
3.5	Multi-Grid Iteration Strategy	85
3.6	Software Implementation	86
3.7	Results in Two Dimensions	87
3.8	Conclusions	90
Chapter 4	Stefan Problem	93
4.1	Introduction	93
4.2	Approach in 1D	94
4.2.1	Domain Discretization and Nomenclature	95
4.2.2	Finite Volume Derivation with Moving Boundary	96
4.2.3	Derivation of the Finite Volume Method	99
4.2.4	Overall Time-Stepping Procedure	104
4.2.5	Discrete conservation	106
4.2.6	Summary of time-stepping algorithm	108
4.3	Results in One Dimension	109
4.3.1	Analytic Solution in One Dimension	109
4.3.2	Heat Equation with Moving Boundary	111
4.3.3	Stefan Problem	120
4.4	Front Tracking	126
4.4.1	Volume of Fluid Algorithm and Derivation	126
4.4.2	Accuracy Assessment	140

4.5	Discretization in Two Dimensions	140
4.5.1	Finite volume derivation in 2D	141
4.5.2	Summary of finite volume discretization	151
4.5.3	Overall time-stepping procedure	152
4.6	Modifications to Multi-grid iteration strategy	153
4.6.1	Coarsening strategy	154
4.7	Software Implementation	155
4.7.1	Description of <code>MovingVoF</code>	155
4.7.2	Description of <code>StefOp</code>	156
4.8	Results in Two Dimensions	156
4.8.1	Laplacian on a Changing Domain	157
4.8.2	Parabolic solution for the heat equation	158
4.8.3	Axisymmetric Analytic Solution with $St = 1$	160
4.8.4	Axisymmetric initial conditions	162
4.8.5	Star-shaped initial configuration	165
4.9	Conclusions	166
Chapter 5	Conclusions	169
5.1	Summary	169
5.2	Poisson Equation	170
5.3	Heat Equation	171
5.4	Stefan Problem	172
5.5	Future Research Directions	173
Chapter	Bibliography	175

List of Figures

1.1	Schematic of the Czochralski growth of a semiconductor crystal.	3
2.1	Diagram of the second-order stencil for the gradient at $x = \ell$. A quadratic polynomial is fitted to the two values of ϕ in neighboring cells, and the value at the interface; the value in the last cell is not used in the calculation. . . .	21
2.2	The effect of volume fraction Λ , on the two-norm condition number of the linear system, DL , where D is a diagonal matrix with ones on the diagonal, except for $D_{NN} = \Lambda$, for our system (dashed) and a piecewise-linear Galerkin discretization, with $N - 1$ variables, on the same grid (solid).	26
2.3	Diagram showing (a) the control volume formulation, which is based on the divergence of appropriately-centered fluxes, and (b) how a properly-centered normal derivative is found by interpolating between two neighboring values.	28
2.4	Diagram of the second-order stencil for the gradient normal to the interface, q^f . Two values are found on the neighboring grid lines, using quadratic interpolation. The gradient is then calculated by fitting a parabola to the interpolated values and the value at the interface.	30
2.5	Diagram of the coarsening strategy for the multi-grid method. The coarse grid preserves the apertures and volumes of the fine grid, but uses a coarser, piece-wise linear representation.	34
2.6	The stencil at the coarse-fine interface is represented. A value is found on a fine-level grid line, from quadratic interpolation on the coarse level. This value, along with two values on the fine grid, is used to calculate a gradient at the coarse-fine interface. The coarse-grid stencil can be shifted when necessary. Finally, the coarse-grid flux (shaded arrow at right) is defined as the sum of the corresponding fine-grid fluxes (white arrows).	36
2.7	Contour plot of the solution to Problem 1.	46
2.8	For Problem 1, we plot (a) the magnitude of the volume-weighted truncation error and (b) the partial volume, on Ω_P versus θ . Note that the former is bounded, even for arbitrarily small volumes.	48
2.9	Plot of the one-norm of the error in the discrete solution, ξ , and its two components: ξ_P , from the $O(\Delta x)$ truncation error on Ω_P ; and ξ_I , from the $O(\Delta x^2)$ truncation error on Ω_I . Note that ξ_P converges like $O(\Delta x^3)$, while ξ_I converges as $O(\Delta x^2)$	50

2.10	Plot of the ∞ -norm of the partial volume-weighted residual, versus multi-grid iteration m , for the single-grid (solid) and adaptive-grid (solid lines with plus signs) solutions of Problem 2.	53
2.11	Surface plot of the solution to Problem 3, for $N = 80$	54
2.12	We present two plots of the first quadrant of Problem 3, Case 1. (a) represents the valid regions of each level, and (b) gives grid block boundaries, and contours of the solution.	57
3.1	We plot the solution of (3.25) for $N = 20, 40, 80, 160, \dots$, for $\Lambda = 10^{-2}$. The error in the solution is $\xi = O(\Delta x^3)$, while the gradient at $x = \ell$ is $G^f \xi = O(\Delta x^2)$, and negative.	67
3.2	We plot the eigenvalues of $-L$ for $\Lambda \in [2^{-5}, 1]$, for $N = 100$ cells. There is a pair of complex conjugate eigenvalues for $\Lambda > 0.1$ (real and imaginary parts marked by stars and circles), for oscillatory modes corresponding to cell N . For $\Lambda \ll 1$, this eigenvalue grows as Λ^{-1} , while the eigenvector resembles a discrete delta function in cell N	69
3.3	Here we plot the eigenvectors of $-L$, for $N = 100$, corresponding to the left- and rightmost cells. The mode on the left is independent of Λ , but the eigenvectors on the right are complex and oscillatory (solid lines) for $\Lambda = O(1)$, although their absolute value is smooth (dotted line). For $\Lambda \ll 1$, these modes transform into a delta function in cell N (dashed line).	70
3.4	We plot the eigenvalues of the hybrid (solid line) discretization, for $\Delta t = \Delta x$, $\Lambda = 2^{-3}$, and $N = 100$ cells, and which uses the backward Euler discretization in the first and last cells. The spectrum is very similar to that of the Crank-Nicholson discretization (dashed line), except for two features. On the right, we see Crank-Nicholson's complex $\lambda \approx -1$ (circles on dashed line) have been replaced by a real $\lambda \approx -1$ (circle on solid line), and a real $\lambda \ll 1$ (same in inset). Similarly, the eigenvalue corresponding to the leftmost cell in the Crank-Nicholson discretization (star, right), has been replaced by another $\lambda \ll 1$ (star, inset).	74
3.5	We plot the eigenvectors for the first and last cells, for the hybrid and Crank-Nicholson discretizations corresponding to Figure 3.4. The undamped eigenvector of the Crank-Nicholson discretization (solid line on left) has been replaced by a heavily damped eigenvector (dashed-dotted) in the hybrid discretization. Crank-Nicholson's complex eigenvectors corresponding to cell N have been replaced with an undamped real mode (solid line on right), and a heavily damped mode (dashed).	75
3.6	We plot the truncation error for each of the methods using the exact solution (3.31). In the interior of the domain, the backward Euler method is first-order accurate (circles), while the Crank-Nicholson (stars) and Runge-Kutta (plus signs) discretizations are second order accurate. On the right, the partial-volume-weighted truncation error in the last cell, $\Lambda \tau_N$, is $O(\Delta x)$ and well-behaved for over a range of Λ and N (top line $\Lambda = 1$, lowest line $\Lambda = 2^{-6}$).	77

3.7	Error in the discrete solution. The left-side plot show the ∞ - and two-norms of ξ after one time step, for a wide range of N . For the backward Euler method (circles), $\xi = O(\Delta t^2)$, indicating a first-order accurate method. The Crank-Nicholson (stars) and RK (plus signs) discretizations are second-order accurate. On the right, we plot same error norms at $t = 0.25$, for these three methods and the hybrid discretization. The backward Euler method is first-order accurate, whereas the others are second-order accurate.	78
3.8	Discontinuous initial conditions. We plot the solution at $t = 0.01$, for the heat equation with $\varphi = 0$ at $x = 0, 1$, and initial conditions $\varphi(x, t = 0) = 1 - 2x - 1 $ (dotted V-shape). The discrete solutions for each of the four methods use $N = 100$, $\Delta t = 0.01$, and $\Lambda = 10^{-3}$. On the left, the exact solution (dotted curve) is smooth, whereas the Crank-Nicholson (solid) and hybrid (dashed) discretizations cause cusps at $x = \frac{1}{2}$. The CN discretization also suffers from $O(1)$ oscillations, due to the discontinuous initial conditions. On the right, the Runge-Kutta method causes small oscillations, while the backward Euler method (dashed curve) causes none at all.	80
3.9	Error at $t = 0.125$ for discontinuous initial conditions. The error norm $\ \xi\ _1$ for the backward Euler method (circles) indicates first-order accuracy. The hybrid method (crosses) is also first-order accurate, due to the persistence of the cusp at $x = \frac{1}{2}$ shown in Figure 3.8. The Runge-Kutta method (plus signs) is second-order accurate, as expected, while the $O(1)$ oscillations at $x = 0, 1$ in the Crank-Nicholson solution (stars) persist, preventing convergence. . .	81
3.10	Discrete solution for the test problem in two-dimensions, at $t = 0.125$, for $\Delta x = \frac{1}{80}$, with twenty contours between $\varphi = 0$ and $\varphi \approx 0.07$	87
3.11	Volume-weighted local truncation error for the test problem in two dimensions. The backward Euler method (circles) has $\ \Delta t \Lambda \tau\ _1 = O(\Delta t^2)$, as expected for a first-order method, while the hybrid (crosses), Crank-Nicholson (stars), and Runge-Kutta discretizations (plus signs) are second-order accurate. On the right, we plot the two components of the larger truncation error, $\Delta t \Lambda \tau_P = O(\Delta t^2)$, due to partial cells (dashed), and τ_I , due to the discretization in full cells, for the Crank-Nicholson method.	89
3.12	Error for two-dimension test problem. The error in the discrete solution after one time step is plotted versus the grid spacing. For the backward Euler method (circles), $\ \xi\ _1 = O(\Delta x^2)$, indicating first-order accuracy, whereas the other methods appear to be second-order accurate. On the right, we have plotted the two components of the error after one timestep, for the Crank-Nicholson discretization. We see that the error induced by the partial cells, ξ_P (dashed line), converges at the same rate as that from the interior of the domain, ξ_I	90

3.13	Error at $t = 0.125$. In the left-hand plot, we see that the ∞ -norm of the discrete error is $O(\Delta x)$ for the backward Euler method (circles), $O(\Delta x^2)$ for the Crank-Nicholson (stars) method, and slightly less than $O(\Delta x^2)$ for the Runge-Kutta (plus signs) discretization. The hybrid method also seems to be second-order accurate, although convergence is erratic for $\ \xi\ _\infty$. On the right, $\ \xi\ _2$ behaves more smoothly, as expected.	91
4.1	The front advances by at most Δx in on time step Δt . It either spends the entire time step in cell N (left), or it crosses between cells N and $N + 1$. . .	95
4.2	We plot the space-time surfaces corresponding to the integrals in (4.3). If we approximate the integrals using the midpoint rule (square boxes, left), we obtain a first-order accurate method. Alternatively, the time integrals may be evaluated at the same location in space, but at t^{n+1} (right), without losing any accuracy	98
4.3	When the front stays in cell N during the time step, we require an approximation to the gradient at \bar{s}_N . This is accomplished by fitting a quadratic polynomial to the two values ϕ_{N-2} and ϕ_{N-1} , and the value 0 at s^{n+1} . The slope of the polynomial is then evaluated at \bar{s}_N to approximate $\partial_x \phi$	102
4.4	When the front crosses between cells N and $N + 1$, the difference formula requires an approximation to the gradient at \bar{s}_N . This is accomplished by fitting a quadratic polynomial to the three values ϕ_{N-1} , ϕ_N , and ϕ_{N+1} , and evaluating its slope at \bar{s}_N	103
4.5	The exact solution to the Stefan problem in one dimension, for $St = 1$, is plotted at times $t = 0.1, 0.3, 0.6$. Because our discrete solution ϕ can extend beyond the front location, we extend the exact solution using (4.25), even when the result is less than zero.	112
4.6	We plot the error in the discrete solution (ξ on the left) for the backward Euler method applied to the heat equation, with $St = 1$, after $n = \frac{1}{2\Delta x}$ time steps. The difference between the exact front speed, and that calculated from $F^{f,n}$, is plotted versus time on the right. In both cases, the errors appear smooth and $O(\Delta x)$	113
4.7	We plot the error in the discrete solution for the hybrid method applied to the heat equation, with $St = 1$, after $n = \frac{1}{2\Delta x}$ time steps. Note the oscillations in the error for $x \gtrsim 0.37$, which we attribute to the hybrid method's inability to damp high-frequency components of the error, which are introduced at the front each time step.	114
4.8	We plot the error in the calculated front speed for the hybrid method applied to the heat equation with $St = 1$, for $n = \frac{1}{2\Delta x}$ time steps determined from (4.26). The error for each grid spacing is very oscillatory (left), but a linear least-squares fit to the data (right) appears to converge like $O(\Delta x^2)$	115
4.9	We plot the total conservation losses $\delta\phi$ versus time for the backward Euler (left) and hybrid methods (right) applied to the first test problem. Both $\delta\phi$ are $O(\Delta x^2)$, although the hybrid case appears to be more oscillatory. . . .	116

4.10	We plot the error in the discrete solution for the backward Euler method with the lagged conservation term, applied to the first test problem. Again, the error appears to be $O(\Delta x)$	116
4.11	We plot the difference between the calculated and exact front speeds, versus time, for the first test problem. Again, there is a oscillatory component to the error (left), but a least-squares fit (right) suggests that it is still a first-order accurate approximation.	117
4.12	The exact solution to the Stefan problem in one dimension, for $St = 100$, is plotted at times $t = 0.01, 0.03, 0.06$ (from left to right).	117
4.13	We plot the error in the discrete solution (left) and calculated front speed (right) for the second test problem, with $St = 100$. Both appear to be $O(\Delta x)$	118
4.14	For the hybrid method applied to the second test problem, the error in the discrete solution (left) appears to be $O(\Delta x^2)$. The error in the calculated front speed looks very much like Figure 4.8, and the linear least-squares fit suggests that it is also $O(\Delta x^2)$	118
4.15	We plot the total conservation losses $\delta\phi$ versus time, for the backward Euler (left) and hybrid (right) algorithms for the second test problem. Both $\delta\phi$ are $O(\Delta x^2)$, as in the first test problem.	119
4.16	When we reincorporate $\delta\phi$, the backward Euler method applied to the second test problem still produces a first-order accurate discrete solution (left). The error in the calculated front speed is again very noisy, like Figure 4.11, but the linear least-squares fit suggests that it is $O(\Delta x)$	119
4.17	The backward-Euler method for the heat equation and forward-Euler method for advancing the front have been applied to the third test problem, with $St = 1$. The error in the discrete solution is smooth, and appears to be first-order accurate.	121
4.18	We plot the error in the front position (left) and the calculated speed (right) versus time, for the backward-Euler method applied to the third test problem. In both cases, the error appears to be $O(\Delta x)$ and smooth.	122
4.19	We plot the error in the discrete solution (left), for the third test problem, using the hybrid method with Adams-Bashforth for advancing the front in time. This results in a second-order accurate front position (right) and solution.	122
4.20	We plot the difference between the calculated and exact front speeds, versus time, for the hybrid method applied to the first test problem. Again, there is a oscillatory component to the error (left), but a least-squares fit (right) suggests that it is $O(\Delta x^2)$	123
4.21	The backward-Euler method for the heat equation and forward-Euler method for advancing the front result in a first-order accurate solution for the the fourth test problem, with $St = 100$	124
4.22	We plot the error in the front position (left) and the calculated speed (right) versus time, for the backward-Euler method applied to the fourth test problem. In both cases, the error appears to be $O(\Delta x)$, but more oscillatory than the results in Figure 4.18, with $St = 1$	125

4.23	We plot the error in the discrete solution (left), for the fourth test problem, using the hybrid method, with Adams-Bashforth for advancing the front in time. The $O(\Delta x^2)$ rate of convergence of the discrete solution is not apparent except at the finest grid resolutions. The front position (right) is oscillatory, but second-order accurate.	125
4.24	A close-up of the error in the calculated front speed, plotted versus time, is presented for the hybrid method applied to the fourth test problem. Again, there is a oscillatory component to the error (left), but a least-squares fit (right) suggests that it is $O(\Delta x^2)$	126
4.25	To extend the front velocity, we $\mathbf{u}_{i,j}$ is calculated first in the cells that contain a portion of the front (filled circles, for which $P_{i,j} = 1$). These velocities are then extended the the remaining cells in $\text{ngh}(i, j)$ (marked with unfilled circles, where $P'_{i,j} = 1$ but $P_{i,j} = 0$), by calculating $\bar{\mathbf{u}}$ from (4.29).	129
4.26	The volume fraction flux across edge $(i + \frac{1}{2}, j)$ is computed by shifting the edge by a distance $-\Delta t u_{i+\frac{1}{2},j}$. $F_{i+\frac{1}{2},j}^\Lambda$ is found by calculating the area of this shifted region (between the dashed line and edge $(i + \frac{1}{2}, j)$), intersected with the region $\Lambda_{i,j}^n$ given by the linear front reconstruction.	132
4.27	There are three cases when specifying the trajectory of the front in space-time. For (a) and (b) when only one of Λ^n or Λ^{n+1} is between 0 and 1, the front trajectory is specified by that Λ , \mathbf{n} , and \dot{s} . In case (c) when both $0 < \Lambda^n, \Lambda^{n+1} < 1$, the volume fractions and \mathbf{n} are used to specify the front location.	133
4.28	For the case when $0 < \Lambda_{i,j}^n, \Lambda_{i,j}^{n+1} < 1$, we calculate the front speed by first reconstructing the front at t^n and t^{n+1} , so that it has normal $\mathbf{n}_{i,j}$, and matches the volume fractions at each time. The quantity $\Delta t \dot{s}_{i,j}$ is given by the distance between these parallel lines, which determines $\dot{s}_{i,j}$	134
4.29	Here we describe the surfaces of the space-time volume, $\Gamma(\mathbf{x}, t)$. The plane S^f represents the trajectory of the front in space-time through the cell $(i, j) \times [t^n, t^{n+1}]$. Where S^f the front location at each time is given by the intersection of S^f with the planes $t = t^n$ and $t = t^{n+1}$	136
4.30	A gradient in the normal direction is calculated at $\bar{\mathbf{x}}_{i,j}^f$, marked with an open circle, using a Taylor series expansion about the front midpoint, \mathbf{x}_m . Instead of performing this in the (x, y) coordinate system, we use the system defined by the cell's outward-pointing normal, and tangent, as one traverses the boundary in the counter-clockwise direction. We then calculate the location of $\bar{\mathbf{x}}_{i,j}^f$ relative to \mathbf{x}_m in the new system, (μ_n, μ_t)	146
4.31	When the front is still present in cell (i, j) , we calculate φ_n at $\bar{\mathbf{x}}_{i,j}^f$ starting with the approximation for φ_n at \mathbf{x}_m (left). We again use the same stencil for G^f , given in (2.17), to approximate both φ_n and φ_{nn} from the value at \mathbf{x}_m , and two values interpolated in the interior, ϕ_1^I and ϕ_2^I . For the remaining cross term, φ_{nt} , we use a six-point stencil that is shifted away from the boundary (right).	148

4.32	When the front has left cell (i, j) , we calculate an approximation to φ_n at $\bar{\mathbf{x}}_{i,j}^f$ using a six-point stencil, but no information about the location of the boundary at the new time.	149
4.33	The location of $\bar{\mathbf{x}}_{i,j}^f$ and $\bar{\mathbf{x}}_{i,j}$ on coarser grids (marked with circles) is determined by the $A_{i,j}^f$ - or $V_{i,j}$ -weighted average of values from the fine grids (marked with squares).	154
4.34	We plot the results of the first test problem, with $\varphi(\mathbf{x}) = \ln \frac{r}{R}$, versus the number of points in each grid direction, N . On the left, we see that the ∞ -norm (solid line) and one-norm (dash-dot line) of the truncation error $ \Lambda\tau $ in partial cells is $O(\Delta x)$, whereas it is $O(\Delta x^2)$ in the interior (dashed line). However, on the right, we see that the error induced by the truncation error in partial cells is $\xi = O(\Delta x^3)$, in all norms.	157
4.35	We plot the initial conditions for the second test problem, with exact solution $\varphi(r, t) = \frac{1}{4}(r_0^2 - r^2) + \beta t$	159
4.36	Because the exact solution to the second test problem is a low-order polynomial, the truncation error for our discretization is zero everywhere, except at the boundary, where it is $O(\Delta x)$ in the ∞ -norm (solid line). The corresponding error in the solution, $ \xi $, is $O(\Delta x^3)$, either after one (dashed line) or $O(N)$ time steps (dash-dot).	160
4.37	We plot the truncation error for the third test problem, where φ is given by the exact solution to the Stefan problem, (4.61). In partial cells, the ∞ - and one-norms of $ \Lambda\tau $ are $O(\Delta x)$ (solid, dashed lines with circles, respectively). Over the part of the domain away from the origin, the truncation error is $O(\Delta x^2)$ (solid, dashed lines).	161
4.38	We plot the discrete solution to the fourth test problem, with $N = 80$, after 40 time steps. The result is smooth, and has a mild change in curvature, indicating that the front moves more quickly than the rate of diffusion. . . .	162
4.39	We plot the error in the discrete solution, $ \xi $, for the fourth test problem. The backward Euler method is first-order accurate in both the ∞ - and one-norms (solid and dashed lines at top), whereas the hybrid method is second-order accurate (solid and dashed lines at bottom).	163
4.40	We apply the hybrid algorithm with $N = 80$ to the fourth test problem, and plot the solution error after 40 timesteps. Note that the hybrid method does not damp high-frequency components of the error caused from the initial discretization error at $r = 0.5$	164
4.41	We plot the initial solution value and boundary position for the last test problem, with $N = 80$	164
4.42	We plot the evolution of the front in time for the last test problem, with $N = 40$. For this coarse grid spacing, the front-tracking algorithm is still able to maintain a coherent interface at the fastest-moving part of the front. . . .	166

- 4.43 We plot the ∞ -, one-, and two-norms of the solution error $|\xi|$ (solid, dashed, and dash-dot lines) for the backward Euler (left) and hybrid (right) methods. At coarser grid resolutions, both appear to be $O(\Delta x^2)$; we attribute this to the convergence of the discretization of the initial conditions. The $O(\Delta x)$ errors of the backward Euler method become apparent at smaller grid spacings, whereas this is not yet the case for the hybrid algorithm. 167

List of Tables

2.1	The norms and convergence rates of the partial-volume-weighted truncation error are presented for Problem 1. The values in partial cells are first-order in the grid spacing, while values in the interior are second-order.	47
2.2	We present the two components of the solution error in Problem 1, and their corresponding convergence rates. The error due to truncation error in the partial cells converges at a substantially higher rate than that due to the interior truncation error.	49
2.3	Adaptive mesh refinement errors for the solution in Problem 1. Each case uses different levels of refinement. We also give the results for uniform grid spacing; arrows indicate which sets of values to compare.	51
2.4	We list errors and convergence rates for Problem 2. The results are very similar to those obtained in Problem 1.	52
2.5	We present results for Problem 3. The error between successive levels is approximately second-order in the grid spacing. In addition, the last column indicates that relatively few cells violate a discrete maximum principle. . . .	55
2.6	Adaptive mesh refinement errors for the solution in Problem 3. Errors are found by comparing the solution to values interpolated from the finest result. The last row contains results for uniform grid spacing; arrows indicate which set of values to compare.	56

Acknowledgements

This research has been funded by:

- “Adaptive Numerical Methods for Partial Differential Equations,” U.S. Department of Energy Mathematical, Computing, and Information Sciences Division, Grant DE-FG03-94ER25205.
- “Computational Fluid Dynamics and Combustion Dynamics,” U.S. Department of High-Performance Computing and Communications Grand Challenge Program, Grant DE-FG03-92ER25140.
- U.S. Air Force Office of Scientific Research, AASERT Grant No. F49620-93-1-0521.
- National Science Foundation Graduate Research Fellowship Program.

Chapter 1

Introduction

Thesis Overview

In this thesis, we have extended a recent numerical approach, the Cartesian grid embedded boundary method, to complete the set of basic tools needed to simulate materials processing and other industrial processes. The embedded boundary method was developed to increase the geometric flexibility and adaptivity of traditional finite difference discretizations, without giving up their computational efficiency. The approach has shown great promise in the aerodynamics community, by reducing the cost of grid generation and simulation of inviscid, compressible flows around complicated aircraft configurations. To augment the advances made in that sector, we have added the algorithms and analysis needed to simulate heat transfer and viscous fluid flow, on changing domains in two dimensions. The resulting collection of algorithms is ideally-suited for simulating a wide variety of free surface problems in industrial applications.

We will focus on one particular application which has benefited from numerical simulations: the growth of large crystals of semiconductor substrates. The Czochralski (CZ) method for growing such crystals involves drawing a seed crystal from a heated crucible, which contains the material's molten components. A crystal's electronic properties are de-

terminated by its composition and thermal history, so it is critical to control heat transfer and fluid motion inside the furnace. However, for most semiconductor compounds, the furnace is a high-temperature, caustic environment, which is difficult to monitor, leading to expensive experimental trial-and-error improvements in the process. This has led researchers to use numerical simulations, to isolate factors in the growth process, and improve the quality of the resulting crystal.

This thesis presents three algorithms which contribute to the Cartesian grid embedded boundary approach, and represent a significant step towards modelling the CZ process. First, we describe a second-order accurate, finite volume discretization for Poisson's equation with spatially-variable coefficients. Our approach calculates a solution centered on a rectangular grid which encompasses the given domain. The finite volume method is derived on each irregular cell, given by the intersection of the rectangular grid and the domain boundary. We use a careful truncation error analysis to derive a consistent finite-volume operator, centered on irregular cells. Boundary fluxes are discretized in a way that results in a well-conditioned linear system, which is amenable to multi-grid iterations. The discretization is also extended to include adaptive mesh refinement (AMR), which improves the accuracy of the solution by decreasing the mesh spacing locally. Second, this approach is extended to create a conservative, second-order accurate discretization for the heat equation. In this case, we use the multi-grid strategy in implicit, time-integration schemes, to avoid time step restrictions. Finally, we have developed a finite-volume discretization of the classical Stefan problem, the simplest model of crystal growth. By combining our strategy for the heat equation with a volume-of-fluid front tracking algorithm, we are able to evolve the solidification front. The combination of algorithms for these three model problems, along with previous embedded boundary implementations for fluid flow, helps complete a set of computational tools that can be extended to more complicated problems.

The remainder of this chapter describes the Czochralski method of crystal growth and its governing equations, and reviews previous numerical simulations using the embedded

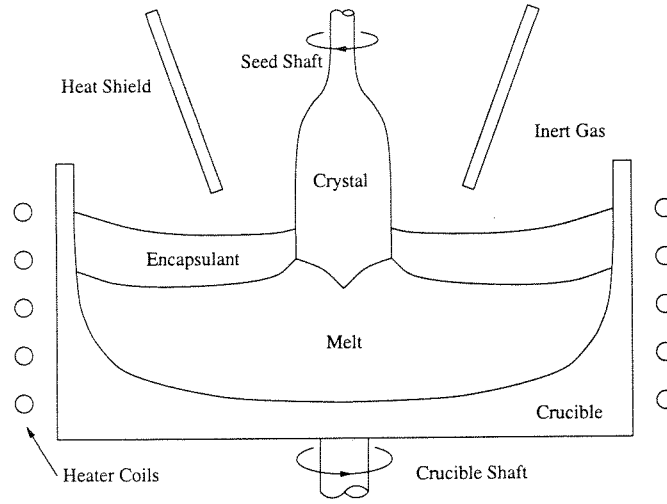


Figure 1.1: Schematic of the Czochralski growth of a semiconductor crystal.

boundary method. Then we introduce our three model problems, and note their contribution to the embedded boundary method. Finally, we offer a general overview of numerical methods, as they relate to crystal growth. The second, third, and fourth chapters present the finite difference algorithms for the Poisson equation, heat equation, and Stefan problem, respectively. In each of those chapters, the discretizations are derived first in one space dimension, then extended to two or more. This allows us to perform a detailed analysis to explain numerical results obtained in two dimensions. After a discussion of each software implementation, each chapter concludes with test problems and convergence studies. The last chapter is devoted to conclusions and suggestions for future work involving the embedded boundary method.

1.1 Target Application: Bulk Crystal Growth

The growth of large single crystals with precise material properties is an important step in the manufacture of electronics components, where it is the most popular method of producing semiconductor substrates used in integrated circuits and solid-state lasers. The

majority of such crystals are grown using the Czochralski (CZ) method and its variants. This approach is based on lowering a seed crystal into a molten mixture of the semiconductor components, and then slowly removing the seed while cooling it at the same time (Figure 1.1). There are a number of factors that can be controlled in this process [33]: the rotation of the seed crystal and melt crucible, the pull rate of the crystal, and the power output and layout of heaters in the furnace. The entire procedure is very difficult to monitor, because of the high-temperatures and caustic environment inside the reactor vessel. However, extensive trial-and-error adjustments to the CZ process have made it the most common means of producing large, high-quality silicon crystals.

Unfortunately, this method has not been as successful for most multiple-component semiconductors, such as gallium-arsenide or indium-phosphide. This is generally believed to be caused by two characteristics of the CZ growth process: large thermal gradients and melt segregation [33]. Thermal gradients in the crystal result in thermal stresses, which in turn adversely affect its microscopic and electronic properties. Large thermal gradients also give rise to buoyancy-driven convection in the melt, which can cause detrimental remelting of the crystal interface [33]. Segregation of the semiconductor components in the melt causes the resulting crystal to have non-uniform properties, and reduces the yield from the process. Similarly, rejection of the solute during solidification can cause the composition of the crystal to change over the length of a run. The need to control these effects has led to variations on the CZ process, include using liquid encapsulants for volatile compounds, pressurizing the reactor vessel, and applying external magnetic fields to suppress convection in the melt. Again, parameters in the growing process have been mostly determined by expensive experimentation, with the quality of the resulting crystal guiding changes in the process.

This has led some researchers to develop numerical methods for the simulation of CZ crystal growth ([70, 66, 71] for example). However, few of these simulations have included all of the aspects of the process. For instance, even though buoyancy-driven

convection is an important, three-dimensional phenomenon in CZ growth, most researchers have assumed the governing equations to be axisymmetric, to reduce computer time and memory usage [70]. It has also been found for some materials, such as semi-transparent oxide crystals used for solid-state lasers, internal radiation can greatly affect the interface shape [65, 12]. Because radiative heat transfer can be computationally expensive, it is often treated using simple “shape factors” between only fixed surfaces [70, 69]. Similarly, many simulations ignore the effect of composition and temperature on material properties, even though these are responsible for some important flow phenomena [25]. The interactions between these phenomena, in the presence of moving interfaces and time-dependent inputs, makes a complete simulation of a CZ furnace prohibitively expensive, at the present time.

Governing Equations

The governing equations for CZ crystal growth can be obtained from a control volume analysis of the conservation laws for mass, momentum, and energy [36], which states that the total change in each quantity consists of fluxes through the the control surface, and sources or sinks within the control volume. The diffusion of mass, momentum, and energy are usually assumed to be governed by Fick’s, Newton’s, and Fourier’s laws, respectively: the diffusive flux of each quantity through the control surface is proportional to its gradient. It has also been observed experimentally that the materials in the furnace are effectively incompressible, so that their bulk density and other properties depend on temperature and composition alone. In the limit of an infinitesimal control volume, these assumptions yield the following partial differential equations (PDE’s) for the system:

$$\partial_t \rho_i + \nabla \cdot (\rho_i \mathbf{u}) = \nabla \cdot \rho D_i \nabla \left(\frac{\rho_i}{\rho} \right) \quad (1.1)$$

$$\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = \nabla \cdot (\mu \nabla \mathbf{u}) - \nabla p + \mathbf{F} \quad (1.2)$$

$$\partial_t(\rho c_p T) + \nabla \cdot (\rho \mathbf{u} c_p T) = \nabla \cdot (\kappa \nabla T) + Q \quad (1.3)$$

$$\nabla \cdot \mathbf{u} = 0 . \quad (1.4)$$

The important variables in these equations are the density, ρ , velocity, \mathbf{u} , and temperature, T . Additional variables are state-dependent material properties, such as the thermal conductivity κ , or source terms, such as a body force induced by gravity, \mathbf{F} .

Equations (1.1)-(1.4) apply separately in each region of Figure 1.1, and require appropriate interface conditions to indicate how these regions interact. For instance, the velocities in the liquids are prescribed where they contact the crucible and crystal. Surface tension affects the interface shapes between the ambient gas, encapsulant, and melt. The excellent reference [33] details additional boundary conditions that are relevant to the CZ method. We must briefly note the most important interaction: the liberation of heat at the crystal-melt interface. When the transition from solid to liquid occurs at the melting point, T_m , conservation of energy requires:

$$\rho_s h_{sl} U_{int} = \mathbf{n} \cdot (\kappa_s \nabla T_s - \kappa_l \nabla T_l) , \quad (1.5)$$

where U_{int} is the relative interface speed between the solid crystal and the liquid melt, denoted by the subscripts s and l , respectively. This merely states that the energy liberated by converting the liquid to solid, with enthalpy of formation h_{sl} , is balanced by the heat flowing from the interface.

1.2 Motivation of the Numerical Investigation

Our goal is to develop conservative finite volume discretizations for simulating engineering problems with free surfaces. Finite volume methods are generally derived from conservation laws applied to a discrete control volume, instead of the infinitesimal one used to obtain the PDE's (1.1)-(1.4). Finite difference operators can then be used to approximate

fluxes across the control surface, so that the discrete evolution equation also satisfies the conservation law. Because of this close relationship to the governing equations, finite volume methods have been developed for a multitude of engineering applications.

The approach suffers from two major shortcomings, however: the limitations of using a global mesh spacing, and the inability to represent complicated geometries. For traditional finite volume methods, the only means of increasing numerical resolution is to decrease the global mesh spacing. For problems where only a small portion of the computational domain is under-resolved, using global refinement wastes computational resources, and generally results in a less-accurate solution. Adaptive mesh refinement (AMR) algorithms have been developed to address this issue. AMR decreases the grid spacing locally, either cell-by-cell, or in block-shaped regions, to improve the discrete solution's accuracy only where desired. This reduces the computer time and memory needed to attain a given level of accuracy.

The Cartesian grid embedded boundary approach has been introduced 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. At 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. Because this requires only local geometric information, such as cell volumes or intersections with grid lines, grid generation is much less expensive, and affects the discretization only locally. And although the method is known to lose accuracy at the domain boundaries, for a large number of problems, this does not significantly affect the resulting global accuracy of the solution.

Both AMR and the embedded boundary approach have been used in compressible fluid dynamics simulations. A conservative volume-of-fluid discretization was presented for hyperbolic conservation laws by Chern and Colella [18]. Their approach was a precursor of the embedded boundary method, in that it used a standard finite volume discretization,

except near jumps in the solution. There, a specialized algorithm was used to track discontinuities in the solution. This front-tracking approach was later extended to include AMR for inviscid compressible fluids [9]. Similar algorithms have also been developed for unsteady inviscid flows with solid-wall boundaries, in two and three space dimensions [49]. The combination of AMR and the embedded boundary method has been effective in steady aerodynamics simulations as well [2, 21]; the approach is able to obtain highly-refined solutions for inviscid flows, without the expensive grid generation procedures used with structured- or unstructured-grid flow solvers. The combination of AMR and the embedded boundary method has also been used for solving the steady, compressible Navier-Stokes equations [20].

Cartesian grid embedded boundary methods have also been developed for incompressible flows, using projection methods. Projection methods provide a means of enforcing divergence constraints like (1.4), and generally involve solving an elliptic equation with Neumann boundary conditions. This approach has been used in predictor-corrector methods for viscous, incompressible fluids [8], including flows with density jumps [48] and low-Mach number combustion [31], but only on rectangular domains. In [3], an embedded boundary projection method was presented for the incompressible Euler equations in two dimensions. Their approach used advection algorithms developed for compressible flow simulations [49], along with two discrete projection operators, derived from variational and conservative (MAC-based) discretizations. The algorithm's numerical results were globally second-order accurate, but attained slightly lower accuracy near the boundary. However, the flexibility of this approach has been demonstrated in simulations of industrial furnaces with many more complicated phenomena, like radiative heat transfer and turbulence models [50]. Although AMR has been used with projection methods [4, 46, 32], the two approaches have not yet been combined with the Cartesian grid embedded boundary method.

1.3 Contributions to the Embedded Boundary Method

Although the embedded boundary method has been used successfully for hyperbolic conservation laws, additional work must be done before the approach can be applied to more general processes, like the Czochralski method of bulk crystal growth. This is because the PDE's (1.1)-(1.3) are parabolic. Very few embedded boundary discretizations have been presented for this type of equation, and none has been thoroughly verified. One must also consider how to enforce divergence constraints, like (1.4); the projection method used in [3] is one means of doing this. That discretization of the Laplacian was conservative, but formally inconsistent, and only defined for Neumann-type boundary conditions. A final open question, is how to develop finite volume formulations in the presence of free boundaries, such as the solidification front in (1.5). Approaches have been presented for hyperbolic conservation laws, but not for elliptic and parabolic equations. These topics must be addressed before the embedded boundary method can be applied to CZ crystal growth.

The present work gives a systematic approach for developing conservative discretizations of elliptic and parabolic equations. Our theoretical framework draws on ideas from Young, et al. [68], in their treatment of steady transonic potential flow around complex bodies. They used a non-conservative formulation based on rectangular finite elements, where nodal values of the solution could be inside or outside the domain. However, the corresponding volume integrals were only over the regions of each cell, that were inside the physical domain. The discrete projection operators in [3] used similar ideas; their cell-centered discretization is the basis of our approach here. In particular, we treat the independent variable as a cell-centered quantity on a regular Cartesian grid. The discrete operator is then derived using a control volume analysis on the valid region of each cell. The resulting integrals are then evaluated, using quadrature formulas and finite difference approximations of the integrands, which determines the accuracy of the discrete operator.

This approach provides us with a framework for deriving finite volume methods for more general conservation laws.

This thesis uses the same procedure to obtain discretizations for three model problems, one in each chapter. The first model problem is a self-adjoint elliptic operator on fixed, irregular domains,

$$\nabla \cdot \beta \nabla \varphi = \rho , \quad (\text{Eq. I})$$

with Neumann- or Dirichlet-type boundary conditions, and $\beta(x, y) > 0$. This equation is an essential part of projection methods for incompressible flows with variable density. In Chapter 2, we derive a discretization for (Eq. I), based on the standard second-order accurate difference stencil in the domain interior, and a first-order discretization in the boundary cells. This larger error at the boundary is shown to induce an error that is third-order in the mesh spacing, due to the Dirichlet boundary condition and the smaller number of boundary cells. With this phenomena in mind, we can then explore the advantages of adaptive mesh refinement for elliptic equations. In addition, we demonstrate the use of multi-grid iterations to solve the resulting linear system. Developing a well-conditioned discretization for (Eq. I), along with a multi-grid strategy compatible with AMR, is the major contribution of Chapter 2.

In Chapter 3, the discretization of (Eq. I) is applied to the initial value problem for the heat equation with variable coefficients,

$$\varphi_t = \nabla \cdot \beta \nabla \varphi , \text{ with } \varphi(\mathbf{x}, t = 0) = f(\mathbf{x}) , \quad (\text{Eq. II})$$

on fixed, irregular domains. In this case, we also find that the larger truncation error near domain boundaries does not affect the time-accuracy of the solution. In one space dimension, we show this using an order-of-magnitude argument, while in two dimensions we rely on numerical test problems. Discussing the discretization in one dimension also allows

us to examine the eigensystem of the discrete update matrix of implicit time-integration schemes, such as the backward Euler, Crank-Nicholson, and Runge-Kutta methods. In two dimensions, these approaches take advantage of the same multi-grid iteration strategy, with minimal modification.

Our last model equation is the classical Stefan problem, which combines the heat equation with an evolution equation for the domain boundary:

$$u(\mathbf{x}, t) = -\text{St } \mathbf{n} \cdot \beta \nabla \varphi(\mathbf{x}, t), \text{ with } \varphi(\mathbf{x}, t) = 0, \quad (\text{Eq. III})$$

where u is the normal velocity at some point \mathbf{x} on the boundary, and St is the Stefan number. This is a simpler form of (1.5), so that (Eq. III) still represents an evolution equation for the domain based on conservation of energy. In Chapter 4, we derive first- and second-order accurate finite-volume discretizations for the heat equation on expanding domains. These are combined with a simple front tracking algorithm, to produce a viable embedded boundary method for the classical Stefan problem.

1.4 Literature Survey of Related Work

Because we are considering both a specific field of application, and the implementation of a general algorithm, it is difficult to provide a complete survey of past work. Instead, we include several major “threads” of research which pertain to crystal growth simulations or the model equations that we discuss in this thesis. In either case, we attempt to describe the benefits and difficulties of each method.

A great deal of information is contained in the recent *Handbook of Crystal Growth* [33], which is an excellent reference for both overviews and detailed descriptions of processes related to crystal growth. In particular, it surveys numerical simulation of unstable or dendritic solidification. This form of crystal growth occurs when a seed crystal is introduced into a supercooled liquid, causing rapid solidification and fingering. This is a difficult

process to model, because the solidification front moves with a curvature-dependent speed, and the resulting dendrites can go through topological changes, like breaking and merging. Even though dendritic solidification is not directly relevant to bulk crystal growth, it is a rigorous test of numerical methods that might be applied to both processes, so, in the spirit of completeness, we will include some of these methods here.

1.4.1 Finite Element Method

Adaptive solutions of elliptic and parabolic equations have long been dominated by the FEM ([28, 23, 34], in addition to many others). This approach is closely tied to the variational formulation of these problems, which results in desirable properties for the resulting linear systems. In addition, general software is readily available, making it relatively easy to include additional effects. Thus, it is no surprise that the FEM is the most popular choice for crystal growth simulations. One noteworthy approach is given in [26, 24], where a finite-element method with structured grids was used to compute axisymmetric flows, with complicated heat transfer mechanisms, for the CZ method with external magnetic fields. Their approach was able to recover a relationship between heater power and crystal growth rate that had been observed experimentally. A more advanced, three-dimensional simulation was performed in [65, 66], which used an unstructured grid method, which was implemented on a parallel architecture. Their simulations were able to identify important three-dimensional phenomena that affected heat and solute transport in the melt.

The two main factors to consider with the FEM are grid generation and performance of the resulting data structures. Generally, when applying the finite-element method to moving boundary problems, one must take great care that the generated grid is of good quality everywhere. An excellent example of this is given by the method for dendritic solidification in [55]; in this case, careful grid generation constitutes an entire subproblem of the numerical method. One must also consider that unstructured grids have overhead associated with their underlying data structures. The required connectivity information incurs

an extra expense, which becomes significant when iterative methods are used to solve the resulting linear equations.

1.4.2 Finite Difference Methods

Finite difference methods on structured grids have been the mainstay of the aerodynamics community for some time. The approach admits conservative and non-conservative discretizations, with varying order and complication. Structured grids also allow for the clustering of grid lines to improve numerical resolution; this is seen most often in simulating the very thin boundary layers surrounding aircraft. Because of the method's popularity, there is a great deal of infrastructure for generating high-quality, adaptive grids for such applications. However, this can become a very expensive computational task for time-dependent or highly intricate configurations in three-dimensions. Further information about structured grid methods for aerodynamics applications can be found in [63].

This approach has also been applied to industrial crystal growth applications; [69, 70] presents a discretization for the simulation of axisymmetric flows in the CZ method. In that reference, an implicit time integration of the governing equations is used to achieve steady state, and grid lines are clustered along the melt interfaces for greater resolution. As with the FEM, additional effects, such as radiative heat transfer and surface tension, have been successfully incorporated into the approach [69].

1.4.3 Front Capturing Methods

Another approach is based on capturing the interfaces between various regions, instead of explicitly tracking them. The capturing technique used the same governing equations everywhere, and representing the interface simply by changing the material properties. This allows the multiple regions to be treated in unison, and removes the need for generating time-dependent grids. One such approach is called the “enthalpy method” [60, 57], which treats the jump conditions at the melt interface using source terms.

Their approach allowed them to advance the temperature and concentration fields without explicitly knowing the location of the solidification front.

Similarly, “phase field” [64, 29] methods have been used to model the solidification front as a diffuse interface. The method is derived by considering an additional phase variable, whose value is one in the solid state, and zero in the liquid. An evolution equation for the phase variable is then derived from basic thermodynamic arguments, coupled to the heat equation. The approach may also be extended to mixtures of pure substances, by considering additional “order parameters” [64]. This method also has the advantage that the interface is no longer explicitly tracked. However, the results depend on a small interface thickness, which theoretically should be resolved in order to consider the solution converged. For finite difference calculations, resolving these internal structures requires daunting computational resources; employing a spectral method [58] or adaptive mesh refinement [6] would be appropriate for exploring this effect. At the present, the accuracy of under-resolved computations is still open to debate.

1.4.4 Boundary Integral Methods

Boundary integral methods use the known form of Green’s function for a given problem. This allows such methods to perform calculations only on the boundary of a domain, and not in the interior; however, only problems for which Green’s function is known may be treated this way.

One example of this approach was presented in [45], where fast multipole and boundary integral methods for Laplace’s equation, were used in conjunction with a finite-difference method for Poisson’s equation with discontinuous right-hand side [44]. Their method was second-order accurate, even in very complicated regions, and had near-optimal work estimates. A significant contribution to the boundary integral approach was recently presented in [30]. There, a similar integral equation approach was incorporated with spectral approximation on an adaptive quad-tree data structure. The resulting combination was

extremely well-suited for smooth right-hand sides with compact support.

Another example was given in [56], where a combined level set and boundary integral method was applied to dendritic solidification in two dimensions. The level set method [47] tracks an interface by treating it as the level set (contour) of a distance function. This makes it well-suited for unstable crystal growth; breaking and merging of dendrites is straightforward, as are interface curvature calculations. The velocity of the solidification front was calculated using an efficient moving boundary integral approach for the heat equation. Using the combination of techniques, it was possible to generate a number of convincing and realistic phenomena.

1.4.5 Immersed Boundary Technique

The “immersed boundary” method (in [51], for example) was introduced to enforce boundary conditions for incompressible flows on changing domains. The idea is based on using discrete delta functions to create smooth jumps in the solution across interfaces. These delta functions are then represented as source terms in a finite-difference calculation on a regular Cartesian mesh. This method is extremely flexible, and has been applied to a variety of problems in fluid dynamics [62, 17]. However, the method requires the discrete impulses to be carefully spaced along the interface, and in some situations it has been shown to lose accuracy [39]. This approach was combined with a careful particle-tracking method in [35], which had the ability to detect and account for topological changes in the interface. The algorithm was also able to incorporate jumps in material properties across the interface using discrete delta functions, but it also used explicit time integration of the heat equation, which severely limited the algorithm’s maximum allowable time step.

1.4.6 Immersed Interface Method

An approach that is closely related to our own is the “immersed interface” method [39], which incorporates interface jump conditions and the given PDE in a local coordinate

system. The result is a finite difference stencil with first-order accurate truncation error, which generally yields second-order accuracy, globally. The immersed interface method has been successfully applied to a variety of flow problems with immersed boundaries [41], such as porous media problems, fluid-boundary interactions at low Reynolds numbers, and Hele-Shaw flow, which is similar to dendritic crystal growth. Time-step restrictions have been avoided by using implicit methods, and in some cases it has been found that the resulting linear system is ill-conditioned. This has recently been resolved with fast solution methods, such as the GMRES and multi-grid [1, 40, 67] algorithms. One drawback of the approach is that it is not conservative, due to the rotated difference stencil. In addition, by using the PDE to cancel error terms in the finite difference stencil, generating the stencil coefficients becomes problem-dependent, and more difficult to automate.

Chapter 2

Poisson Equation

2.1 Introduction

In this chapter we present a numerical method for solving the variable-coefficient Poisson equation,

$$\nabla \cdot \beta \nabla \varphi = \rho \text{ with } \beta = \beta(x, y) > 0 \quad (\text{Eq. I})$$

on a bounded two-dimensional region Ω , with either Dirichlet or Neumann-type boundary conditions on the domain boundary, $\partial\Omega$. Our approach uses a finite-volume discretization, which embeds the domain in a regular Cartesian grid with mesh spacing Δx . We treat the solution as cell-centered on a rectangular grid, even when the cell centers are outside the domain. We discretize (Eq. I) on each cell by applying the divergence theorem on the intersection of that cell with Ω . This leads to a conservative, finite-volume discretization on the cells that intersect $\partial\Omega$. Thus, the discretized operator is centered at the centroids of partially covered cells, in contrast to the solution values, which are centered on the rectangular grid. The fluxes at the cell edges are computed using second-order accurate differences of the cell-centered values of the solution. In cells away from the boundary, the algorithm

reduces to the standard five-point discretization for (Eq. I), with a truncation error that is $O(\Delta x^2)$. On the boundary, this discretization results in an $O(\Delta x)$ truncation error; however, this boundary truncation error induces a solution error that is $O(\Delta x^3)$, so that the overall solution is still $O(\Delta x^2)$. For each cell covered by a portion of $\partial\Omega$, along which Dirichlet boundary conditions are enforced, the flux through the boundary is calculated using only values from other cells. This leads to a linear system whose conditioning properties are uniform independent of the smallest partial cell volume, and are essentially the same as that of a problem without irregular boundaries having the the same rectangular mesh spacing. This allows us to use multi-grid iterations with a simple domain-decomposition point relaxation strategy. We have combined this with an adaptive mesh refinement (AMR) procedure, based on the block-structured approach of Berger and Oliger [10]. We show evidence that the algorithm is second-order accurate for various exact solutions, and compare the adaptive and non-adaptive calculations.

For the remainder of this chapter, we will give the details of the algorithm and its implementation. In Section 2, we describe the discretization in one dimension, and provide some analysis of the accuracy of the method, as well as the conditioning of the resulting linear system. We then describe the non-adaptive algorithm for two dimensions in Section 3. In Section 4, we discuss our multi-grid iterative method; Section 5 explains the modifications needed to include adaptive mesh refinement. Section 6 briefly describes the software implementation in C++ and FORTRAN. In the last section, we present numerical test cases and demonstrate the method's accuracy.

2.2 One-Dimensional Algorithm

2.2.1 Discretization

Consider the one-dimensional case of (Eq. I): the Poisson equation with Dirichlet boundary conditions,

$$\varphi_{xx} = \rho \text{ for } x \in [0, \ell], \text{ with } \varphi(0) = \Phi^0, \varphi(\ell) = \Phi^f. \quad (2.1)$$

We discretize the interval $[0, \ell]$ with N finite difference cells, by first choosing a volume fraction for the last cell, $\Lambda \in (0, 1]$, and then defining the grid spacing as

$$\Delta x = \frac{\ell}{N - 1 + \Lambda}.$$

Then the volume V_i of each cell is Δx , except for the rightmost cell N abutting $x = \ell$, for which $V_N = \Lambda \Delta x$. Cells which have $V_i = \Delta x$ are called “full” cells; cell N is a “partial” cell. Thus, the locations of cell edges are given by $x_{i+\frac{1}{2}} = i \Delta x$, $i = 0, \dots, N - 1$, but the right edge of cell N at $x = \ell$.

Our discretized solution is denoted by ϕ_i , $i = 1, \dots, N$, which is centered at the middle of the regular “Cartesian” cells of length Δx ,

$$\phi_i \approx \varphi(x_i), \text{ where } x_i = \left(i - \frac{1}{2}\right) \Delta x \text{ for } i = 1, \dots, N.$$

Note that ϕ_N is assumed to be a cell-centered quantity, even if the center of the Cartesian cell is outside the problem domain. In that case, we are assuming that the solution φ can be extended smoothly a small distance beyond $x = \ell$, and that the discrete solution approximates the value of this extended solution.

Our approach is then based on averaging (2.1) over each cell volume, to obtain

$$\frac{1}{V_i} \int_{V_i} \varphi_{xx} dx' = \frac{1}{V_i} \int_{V_i} \rho dx' . \quad (2.2)$$

Of course the right-hand side of (2.2) is just the average value of ρ over cell i . We then apply the divergence theorem to the integral of φ_{xx} ; for full cells, this reduces to

$$\frac{\varphi_x \left(x_{i+\frac{1}{2}} \right) - \varphi_x \left(x_{i-\frac{1}{2}} \right)}{\Delta x} = \frac{1}{V_i} \int_{V_i} \rho dx' ,$$

while for the partial cell, we have

$$\frac{\varphi_x(\ell) - \varphi_x \left(x_{N-\frac{1}{2}} \right)}{\Lambda \Delta x} = \frac{1}{V_i} \int_{V_i} \rho dx' .$$

We obtain a discretization of (2.1), by using second-order finite difference approximations for φ_x at cell edges and the integral of ρ over cell i .

Let us denote the finite difference approximations of φ_x using $G\phi$, with appropriate subscripts. On cell edges in the interior of the domain, we use centered differences to approximate φ_x :

$$G\phi_{i+\frac{1}{2}} = \frac{\phi_{i+1} - \phi_i}{\Delta x}, \quad i = 1, \dots, N-1$$

Note that this same gradient discretization is used on the interior edge of the partial cell, N , abutting $x = \ell$ (Figure 2.1). This expresses the idea that values of the solution are cell-centered, even if those centers are *outside* the domain. These centered difference approximations of φ_x are second-order accurate in the grid spacing. To approximate a gradient at $x = 0$, we fit a quadratic polynomial through the values Φ^0 , ϕ_1 , and ϕ_2 , and evaluate its

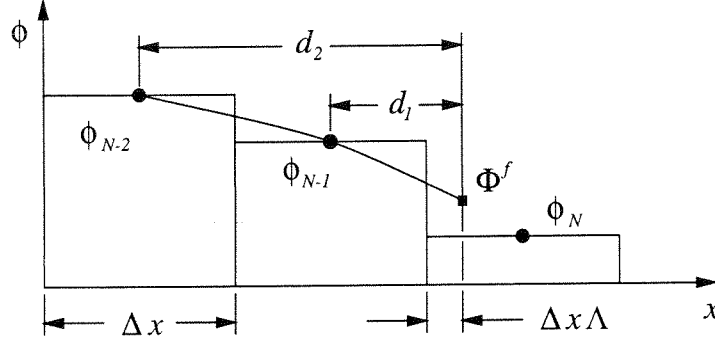


Figure 2.1: Diagram of the second-order stencil for the gradient at $x = \ell$. A quadratic polynomial is fitted to the two values of ϕ in neighboring cells, and the value at the interface; the value in the last cell is not used in the calculation.

slope at $x = 0$:

$$G\phi_{\frac{1}{2}} = \frac{1}{3\Delta x} (9\phi_1 - \phi_2 - 8\Phi^0)$$

This is also a second-order accurate finite difference discretization. Finally, to approximate the gradient at $x = \ell$, we apply a similar one-sided difference stencil, but using only values in neighboring cells. This specialized gradient discretization is denoted by the superscript f , and given by

$$G^f\phi = \frac{1}{d_2 - d_1} \left((\Phi^f - \phi_{N-1}) \frac{d_2}{d_1} - (\Phi^f - \phi_{N-2}) \frac{d_1}{d_2} \right). \quad (2.3)$$

This difference formula is depicted in Figure 2.1; one may verify that (2.3) is second-order accurate by examining a Taylor series of ϕ_{N-1} and ϕ_{N-2} about $x = \ell$.

The average of ρ in (2.2) can be approximated to second-order accuracy using the midpoint rule. In our discretization, the midpoint or centroid \bar{x}_i of any full cell is merely x_i ; for the partial cell, it is given by

$$\bar{x}_N = \frac{1}{2} \left(x_{N-\frac{1}{2}} + \ell \right).$$

We then use $\bar{\rho}_i = \rho(\bar{x}_i)$ to approximate the right-hand side of (2.2), thus completing the discretization of (2.1). To summarize, in full cells, the discrete equations are given by

$$L\phi_i = \frac{1}{\Delta x} \left(G\phi_{i+\frac{1}{2}} - G\phi_{i-\frac{1}{2}} \right) = \bar{\rho}_i . \quad (2.4)$$

Note that using centered differences for $G\phi$ reduces (2.4) to the standard three-point finite difference stencil for the Laplacian. We have merely derived it using a finite volume interpretation. In the partial cell abutting $x = \ell$, we have a slightly different form, however:

$$L\phi_N = \frac{1}{\Lambda\Delta x} \left(G^f\phi - G\phi_{N-\frac{1}{2}} \right) = \bar{\rho}_N . \quad (2.5)$$

2.2.2 Truncation Error Analysis

Let ϕ_i^e be the value of the exact solution at centers of Cartesian cells: $\phi_i^e = \varphi(x_i)$.

Then the truncation error τ is defined as

$$\tau_i = \bar{\rho}_i - L\phi_i^e$$

Note that τ , like ρ and $L\phi^e$, is centered on the irregular grid. The error $\xi = \phi - \phi^e$ satisfies the following system of equations:

$$L\xi = \tau, \quad \Phi^0 = \Phi^f = 0 \quad (2.6)$$

We have the following error estimates for τ :

$$\begin{aligned} \tau_1 &= C_1\Delta x \\ \tau_i &= C_i\Delta x^2, \text{ for } i = 2, \dots, N-1 \\ \tau_N &= C_N \frac{\Delta x}{\Lambda} . \end{aligned} \quad (2.7)$$

In the estimates (2.7), C_1, \dots, C_{N-1} are functions of Δx that are uniformly bounded in $\Delta x, i$, provided φ is smooth. C_N is a function of Δx and Λ that is uniformly bounded as both those quantities vary. At first glance this estimate of τ_N may seem singular as $\Lambda \rightarrow 0$, but the singularity is matched by that in the denominator of the expression defining the operator in (2.5). Ultimately, this leads to an estimate of $\xi = O(\Delta x^2)$, uniformly in Λ . We demonstrate this as follows.

We first use (2.4) and (2.5) to calculate $L\xi$, and multiply by the appropriate cell volumes. After summing, we obtain

$$\begin{aligned}
G\xi_{i+\frac{1}{2}} &= G^f\xi + \sum_{i < j < N} \Delta x \tau_j + \Lambda \Delta x \tau_N \\
&= G^f\xi + \Delta x^3 \sum_{i < j < N} C_j + C_N \Delta x^2 \quad \text{if } i > 0 \\
&= G^f\xi + \Delta x^3 \sum_{1 < j < N} C_j + C_N \Delta x^2 + C_1 \Delta x^2 \quad \text{if } i = 0 \\
&= G^f\xi + D_{i+\frac{1}{2}} \Delta x^2
\end{aligned} \tag{2.8}$$

where the D_i 's are uniformly bounded in $\Delta x, i$, and Λ . Given this expression for the $G\xi$, we can solve for the ξ 's:

$$\begin{aligned}
\xi_1 &= \frac{\Delta x}{8} \left(3 G\xi_{\frac{1}{2}} + G\xi_{\frac{3}{2}} \right) \\
&= \frac{\Delta x}{2} G^f\xi + E_1 \Delta x^3 \\
\xi_i &= \xi_1 + \Delta x \sum_{1 \leq j < i} G\xi_{j+\frac{1}{2}}, \quad i = 2, \dots, N \\
&= x_i G^f\xi + E_i \Delta x^2
\end{aligned} \tag{2.9}$$

Again, the E_i 's are uniformly bounded in $\Delta x, i$ and Λ . Combining (2.8), (2.9), and the

boundary condition (2.3), we obtain the following relation for $G^f \xi$:

$$\begin{aligned} G^f \xi &= \frac{1}{\Delta x} \left(-\frac{d_2}{d_1} \xi_{N-1} + \frac{d_1}{d_2} \xi_{N-2} \right) \\ &= \left(-\frac{d_2}{d_1} (\ell - d_1) + \frac{d_1}{d_2} (\ell - d_2) \right) G^f \xi + \left(\frac{d_1}{d_2} E_{N-2} - \frac{d_2}{d_1} E_{N-1} \right) \Delta x^2 \end{aligned} \quad (2.10)$$

Solving for $G^f \xi$, we finally obtain:

$$G^f \xi = \frac{d_1^2 E_{N-2} - d_2^2 E_{N-1}}{\ell(d_1 + d_2)} \Delta x \quad (2.11)$$

Thus, $G^f \xi$ is $O(\Delta x^2)$, uniformly in Λ . From this and the estimates (2.9) we obtain the result that ξ is $O(\Delta x^2)$ uniformly in Λ .

We can obtain more detailed information regarding the effect of the larger truncation error in the irregular cell. We define ξ_P to be the contribution to ξ from τ_N separately; this is obtained by solving

$$\begin{aligned} (L\xi_P)_i &= 0 \quad \text{if } i \neq N \\ (L\xi_P)_N &= \tau_N \quad \text{otherwise.} \end{aligned} \quad (2.12)$$

A simple calculation shows that the discrete solution is linear, with constant gradient,

$$(G\xi_P)_{i+\frac{1}{2}} = \frac{\tau_N \Lambda \Delta x^2}{\left(\frac{d_2}{d_1} - \frac{d_1}{d_2} \right) \ell} = O(\Delta x^3) \quad i = 0, \dots, N,$$

so that $\xi_P = O(\Delta x^3)$, uniformly in Λ . Thus we observe that the apparently singular contribution to the truncation error in the irregular cell does not lead to a singularity in the error estimate, due to the multiplication by the length of the cell in (2.8). In fact, ξ_P , the contribution to the error, is two orders smaller than τ_N , uniformly in Λ .

This fact can be understood from the point of view of potential theory. We can view the error equation (2.6) as being approximated by a continuous potential theory problem

for (2.1), in which the charge is piecewise constant in cells with values given by the τ_i 's. In that case, the contribution to the field ξ from τ_N , in the sense of (2.12) is given by a dipole located at $x = \ell$ of strength $\Lambda\Delta x \tau_N$. The reason this is a dipole, rather than a monopole with the same charge, is that the effect of the homogeneous Dirichlet boundary condition at ℓ on the field induced by τ_N can be represented by an image charge with the same total charge, but of opposite sign, located at a distance $\frac{1}{2}\Lambda\Delta x$ to the right of ℓ . The dipole results in an induced field which is one order smaller in Δx , giving $\xi_P \sim O(\Delta x^3)$. We have shown that the conclusion from this potential-theoretic model is rigorously correct in one dimension. For the extension of this algorithm to two space dimensions in the next section, we will use this idea to interpret the various contributions to the error observed numerically.

2.2.3 Discussion

First, we would like to note that incorporating Neumann boundary conditions into (2.1) is straightforward. Our discretization is derived by approximating gradients on the domain boundaries; this quantity is known a priori with Neumann boundary conditions. It is easily demonstrated that we obtain the same truncation error estimate for this problem. However, the Neumann boundary condition does not generate the dipole error field that was shown above, and the effect of the first-order truncation error at either end of the domain is $O(\Delta x^2)$. Note, however, that this result is still well-behaved for arbitrarily small Λ .

For the Dirichlet boundary condition, we wish to emphasize that it is essential that the gradient stencil at $x = \ell$ be well-separated from the boundary. The use of such a stencil leads to the uniform boundedness of the conditioning of the linear algebra as Λ approaches zero. This is definitely not the case with more conventional Galerkin approximations on this kind of irregular grid. In Figure 2.2 we plot the condition number of the partial-volume-weighted matrix versus Λ , with $N = 50$, along with that of a piecewise-linear Galerkin discretization, with $N - 1$ degrees of freedom and the same cell sizes. The poor

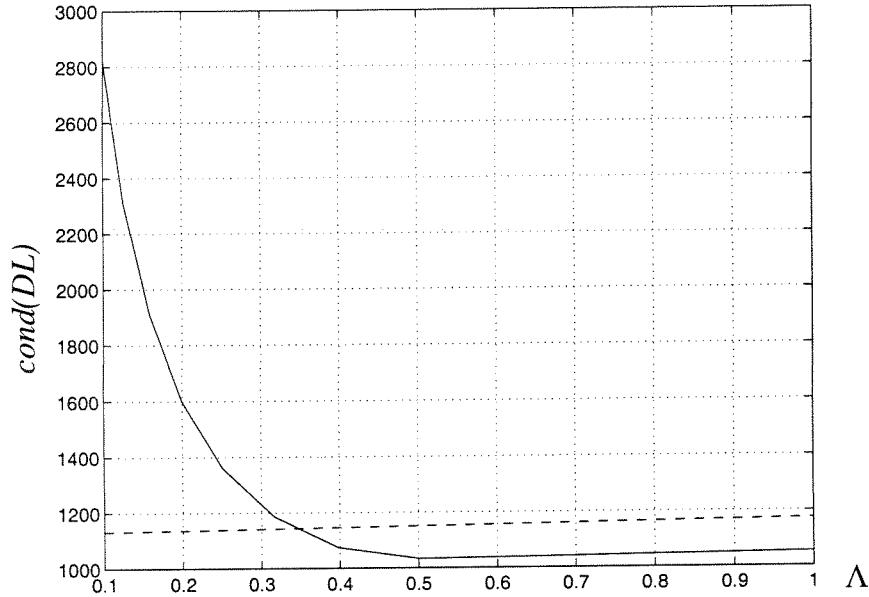


Figure 2.2: The effect of volume fraction Λ , on the two-norm condition number of the linear system, DL , where D is a diagonal matrix with ones on the diagonal, except for $D_{NN} = \Lambda$, for our system (dashed) and a piecewise-linear Galerkin discretization, with $N - 1$ variables, on the same grid (solid).

conditioning of the Galerkin approach is due to a refined approximation of the gradient at $x = \ell$, as $\Lambda \rightarrow 0$. We have effectively exchanged this dependence on Λ for one involving Δx , to avoid poor conditioning in the presence of arbitrarily small volume fractions. The price we pay is that the matrix is not symmetric due to the gradients calculated from quadratic polynomials. In addition, the matrix is no longer diagonally dominant, and thus may not satisfy a discrete maximum principle. The result is that for Laplace's equation, the discrete solution might attain its maximum value inside the domain boundary, unlike the analytic solution.

2.3 Two-Dimensional Case

The algorithm in the previous section extends naturally to more space dimensions, because it is based on a finite-volume formulation. The dependent variables ϕ are cell-

centered on a uniform rectangular grid: $\phi_{i,j} \approx \varphi((i - \frac{1}{2})\Delta x, (j - \frac{1}{2})\Delta y)$, where φ is a solution to (Eq. I). The operator is discretized by integrating (Eq. I) over the control volume of each cell; however, to calculate this integral we must first define how the domain boundary is represented. We use a piecewise-linear representation, which is linear in each cell and defined by the intersection of the domain boundary, or “front,” with the cell edges (Figure 2.3). The volume fraction and front normal are then determined from this representation. In each cell (i, j) , a simple relationship exists between the inward-facing normal \mathbf{n} , the area of the front A_f , and the areas of the cell edges, or “apertures”:

$$A_{i,j}^f \mathbf{n}_{i,j} = \left(A_{i+\frac{1}{2},j} - A_{i-\frac{1}{2},j} \right) \hat{i} + \left(A_{i,j+\frac{1}{2}} - A_{i,j-\frac{1}{2}} \right) \hat{j}. \quad (2.13)$$

For full cells, all aperture values are equal to the grid spacing, implying that A^f is zero, and $\Lambda = 1$. In partial cells, A^f is non-zero. We are ignoring the possibility of very narrow (with width less than Δx) “fingers” of the boundary that enter and exit through the same cell edge. A similar algorithm can be used for three dimensions, where the cell faces are defined analogously, and they in turn define the front normal and area. See [49] for a discussion of this kind of geometry discretization, and some of its limitations.

2.3.1 Discretization

The first step in the derivation is to integrate (Eq. I) over each cell’s control volume, and then invoke the divergence theorem. In order to best approximate the surface integral of the resulting fluxes, they are evaluated at the midpoint of each full or partial edge, as in Figure 2.3(a). After dividing by the cell volume, $V_{i,j}$, we obtain the difference approximation

$$\begin{aligned} L\phi_{i,j} &= \frac{1}{\Delta x \Delta y \Lambda_{i,j}} \left(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j} + F_{i,j+\frac{1}{2}} - F_{i,j-\frac{1}{2}} - F_{i,j}^f \right) \\ &= \bar{\rho}_{i,j}, \end{aligned} \quad (2.14)$$

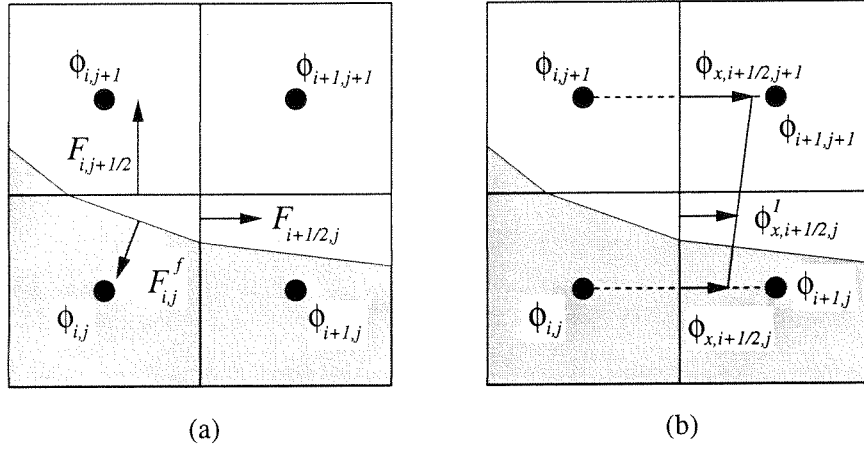


Figure 2.3: Diagram showing (a) the control volume formulation, which is based on the divergence of appropriately-centered fluxes, and (b) how a properly-centered normal derivative is found by interpolating between two neighboring values.

where we have introduced the volume fraction, $\Lambda_{i,j} \in (0, 1]$, with $V_{i,j} = \Delta x \Delta y \Lambda_{i,j}$. The flux through each surface of the control volume in Figure 2.3(a) is denoted by F . The right-hand side of (2.14) uses $\bar{\rho}_{i,j} = \rho(\bar{\mathbf{x}}_{i,j})$, the value of ρ evaluated at the centroid of the irregular cell, $V_{i,j}$. As in one dimension, this is a second-order accurate approximation of the average of ρ over $V_{i,j}$.

For full edges, the flux is found by first calculating a gradient of ϕ normal to the face, using central differencing of neighboring cell values. Note again that ϕ is treated as a cell-centered quantity, even when that center is beyond the domain boundary (as demonstrated in Figure 2.3). Finally, to calculate F , the gradient is multiplied by β , evaluated at the midpoint of the face, and the area of the face. For the full edge at $(i + \frac{1}{2}, j)$, this reduces to

$$F_{i+\frac{1}{2},j} = \Delta y \beta_{i+\frac{1}{2},j} \frac{(\phi_{i+1,j} - \phi_{i,j})}{\Delta x}. \quad (2.15)$$

For full cells, it is obvious that this reduces (2.14) to the standard five-point finite difference stencil for (Eq. I). In partial cells, there are partial apertures which are not equal to the

grid spacing, so that the front area A^f is non-zero. In that case, we must do some additional work to construct second-order accurate fluxes.

On a partial edge, the centering of the gradient and β should still be the midpoint of that edge. Therefore, to calculate the normal gradient, we have chosen to linearly interpolate between values at the midpoints of full edges. More specifically, in Figure 2.3(b), the partial edge $(i + \frac{1}{2}, j)$ has midpoint $\bar{\mathbf{x}}_{i+\frac{1}{2},j}$. If $\bar{y}_{i+\frac{1}{2},j} > y_{i,j}$, we use centered differences at neighboring edge $(i + \frac{1}{2}, j + 1)$ to obtain an interpolated gradient,

$$\bar{G}\phi_{i+\frac{1}{2},j} = (1 - \eta) \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x} + \eta \frac{\phi_{i+1,j+1} - \phi_{i,j+1}}{\Delta x},$$

where $\eta = \frac{1}{\Delta y} (\bar{y}_{i+\frac{1}{2},j} - y_{i,j})$. Had it been that $\bar{y}_{i+\frac{1}{2},j} < y_{i,j}$, we would have used a gradient from $(i + \frac{1}{2}, j - 1)$ to complete the interpolation:

$$\bar{G}\phi_{i+\frac{1}{2},j} = (1 + \eta) \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x} - \eta \frac{\phi_{i+1,j-1} - \phi_{i,j-1}}{\Delta x}.$$

The flux used in (2.14) is then given by

$$F_{i+\frac{1}{2},j} = \beta_{i+\frac{1}{2},j} A_{i+\frac{1}{2},j} \bar{G}\phi_{i+\frac{1}{2},j}, \quad (2.16)$$

where $\beta_{i+\frac{1}{2},j}$ represents $\beta(x, y)$, evaluated at $\bar{\mathbf{x}}_{i+\frac{1}{2},j}$. We can define similar formulas for $(i, j + \frac{1}{2})$ edges using values at their midpoints. In either case, this provides a second-order accurate approximation of the flux through the cell edge. Note that, for full edges $\bar{y}_{i+\frac{1}{2},j} = y_{i,j}$, and both formulas reduce to (2.15).

To obtain a consistent discretization of (2.14), F^f should also be based on quantities centered at the midpoint of the front. Because only the normal component of the gradient contributes to the resulting flux, we have chosen to calculate it using values along a line normal to the interface, and passing through its midpoint (see Figure 2.4). As in one dimension, we employ a three-point gradient stencil, using values from cells other than

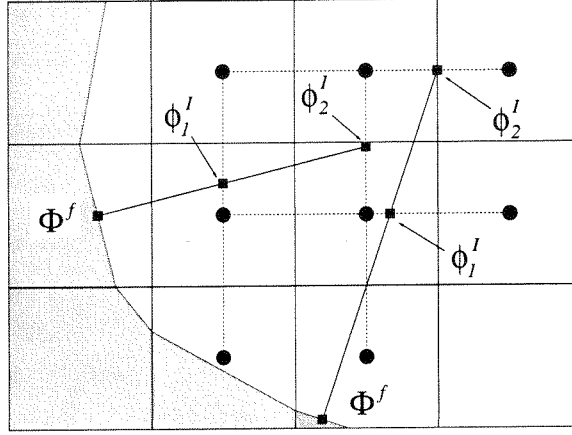


Figure 2.4: Diagram of the second-order stencil for the gradient normal to the interface, q^f . Two values are found on the neighboring grid lines, using quadratic interpolation. The gradient is then calculated by fitting a parabola to the interpolated values and the value at the interface.

the current cell. To do this, we select the first pair of grid lines that intersect with the line normal to the interface, but do not pass through the current cell. We then interpolate between values along each grid line (marked with circles in Figure 2.4), to the intersection points (marked with boxes in Figure 2.4). To obtain a second-order accurate gradient, we must use quadratic polynomial interpolation along grid lines, and then apply the gradient formula as in one dimension:

$$G^f \phi = \frac{1}{d_2 - d_1} \left(\frac{d_2}{d_1} (\Phi^f - \phi_1^I) - \frac{d_1}{d_2} (\Phi^f - \phi_2^I) \right) \quad (2.17)$$

Here we have used Φ^f for the value of ϕ on the front; this is given by the Dirichlet boundary condition at the front's midpoint. Interpolation along grid lines determines ϕ_1^I and ϕ_2^I , at the points distance d_1 and d_2 away from the interface. Finally, we can evaluate the interface flux in cell (i, j) ,

$$F_{i,j}^f = \beta_{i,j}^f A_{i,j}^f G^f \phi_{i,j} , \quad (2.18)$$

where β^f is the value of $\beta(x, y)$ at the midpoint of the front, $\bar{\mathbf{x}}_{i,j}^f$.

By constructing the gradients in this fashion, we impose one more constraint on the discretization of the domain: the interpolation stencil must not reach into cells with zero volume. For the quadratic gradient stencil, this may imply certain constraints on the discretization of the domain. However, the fact that a zero-volume cell is within two rows of another partial cell would indicate that the local boundary is substantially under-resolved. Such domains are more appropriately treated with adaptive mesh refinement, which is described in Section 5.

2.3.2 Truncation Error Estimates

Using arguments similar to the one dimensional case, we can compute the local truncation error. We assume that $\varphi = \varphi(x, y)$ is a smooth solution to (Eq. I), for the case that ρ , β , and $\partial\Omega$ are smooth. We further assume that φ can be extended smoothly to a slightly larger open set containing Ω . Then for Δx , Δy sufficiently small, we can define the truncation error $\tau_{i,j}$:

$$\begin{aligned}\tau_{i,j} &= \bar{\rho}_{i,j} - L\phi_{i,j}^e \\ \phi_{i,j}^e &= \varphi(\mathbf{x}_{i,j})\end{aligned}\tag{2.19}$$

Note that here, as in one dimension, the dependent variable ϕ^e is centered on the rectangular Cartesian grid, while the truncation error, like the operator and right-hand side, is centered at the centroid of the partial cells. In that case, we have the following estimates of the truncation error:

$$\begin{aligned}\tau_{i,j} &= C_{i,j}\Delta x^2 \quad \text{for full cells} \\ &= C_{i,j}\frac{\Delta x}{\Lambda} \quad \text{for partial cells} .\end{aligned}\tag{2.20}$$

Here $\Delta x = r\Delta y$ for some fixed $r > 0$, and the coefficients $C_{i,j}$ are bounded independent of Δx , Λ , and (i, j) . For interior cells, we obtain the standard centered-difference cancellation of error so that the local truncation error is $O(\Delta x^2)$. On the partial cells, that cancellation does not take place, so that the standard Taylor-expansion arguments, plus the fact that the truncation error in the flux calculations is $O(\Delta x^2)$, lead to the estimate given above.

Based on a similar potential-theoretic argument as discussed in the one-dimensional case, we expect that the estimates (2.19) are sufficient to guarantee second order accuracy of the the solution. Specifically, we consider the error equation

$$L\xi = \tau, \quad \xi = \phi - \phi^e \quad (2.21)$$

If we approximate this as a continuous potential theory problem with a piecewise constant charge $\tau_{i,j}$ on each cell, we expect the contribution of each cell to ξ to be proportional to the total charge on that cell. For a full cell (i, j) , the total charge is $\tau_{i,j} \times r\Delta x^2 = O(\Delta x^4)$. There are $O(\frac{1}{\Delta x^2})$ such cells, leading to a contribution of $O(\Delta x^2)$ to ξ . The total charge in an partial cell (i, j) is $\tau_{i,j} \times \Lambda r\Delta x^2 = O(\Delta x^3)$, uniformly in Λ . However, the contribution to ξ from that charge is a dipole field that is one order smaller in Δx , i.e. $O(\Delta x^4)$. This is because of the influence of the homogeneous Dirichlet boundary condition, whose effect on the field induced by the charge in the partial cell can be represented as an image charge of the same strength but of opposite sign located a distance $O(\Delta x)$ away from the partial cell just outside the boundary of the domain. The field strength at a distance d from a given pair of charges can be approximated by

$$\begin{aligned} \xi(d) &\approx \tau_{i,j} \Lambda_{i,j} r\Delta x^2 \left(\ln(d + \Delta x) - \ln(d - \Delta x) \right) \\ &\approx O(\Delta x^3) \times O(\Delta x). \end{aligned}$$

There are $O(\frac{1}{\Delta x})$ such cells, so that the contribution to the error is $O(\Delta x^3)$, uniformly with respect to the range of values taken on by the $\Lambda_{i,j}$'s. We will verify in detail this behavior

in our discussion of the results below.

2.3.3 Neumann boundary conditions

Finally, we wish to note that the finite volume formulation (2.14) allows us to impose Neumann boundary conditions for (Eq. I) with minimal difficulty. We merely substitute the value of the gradient at the front midpoint, instead of using the finite difference stencil (2.17). However, the solution error due to partial cells is now $O(\Delta x^2)$, instead of the $O(\Delta x^3)$ for Dirichlet boundary conditions. Again, the error is well-behaved as $\Lambda \rightarrow 0$.

2.4 A Multi-Grid Method for our Discretization

In order to efficiently find the solution to the linear system derived from (2.14), we use multi-grid iterations [14]. The multi-grid method is based on combining simple point-relaxation schemes and a hierarchy of coarser grids. After applying point relaxation on the finest grid, a correction term is found by representing the fine-grid residual on the next coarsest grid, and using point-relaxation there. This is applied recursively, down the hierarchy of grids, until the problem is coarsened enough to be solved directly. The correction terms are then interpolated back up the hierarchy, while applying point-relaxation at each level. In all, this is called a multi-grid “V-cycle”. Multi-grid methods have the dual benefit of low memory overhead and theoretically-optimal convergence rate. Generally, the method’s difficulties are in defining appropriate “coarsened” operators, along with restriction and interpolation functions; poor choices can result in significantly slower convergence. Because of the enormous interest in multi-grid methods, there is a theoretically-optimum choice of these operators for discretizations based on the variational form of (Eq. I) [7]. However, for non-variational finite volume discretizations, this is not as straight-forward, although some work has been done [13]. We have chosen a simple approach, which performs well for the test problems presented at the end of this chapter.

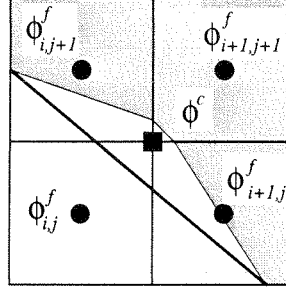


Figure 2.5: Diagram of the coarsening strategy for the multi-grid method. The coarse grid preserves the apertures and volumes of the fine grid, but uses a coarser, piece-wise linear representation.

2.4.1 Point relaxation scheme

The details of the multi-grid iteration scheme are straightforward, once the grid hierarchy is established. The point relaxation scheme that we use resembles a multiplicative Schwarz algorithm from domain decomposition [16]. On the partial cells, we perform one point-Jacobi iteration, while holding the values in the full cells fixed. For iteration m , this is expressed as

$$\phi_{i,j}^m = \phi_{i,j}^{m-1} - \frac{1}{\mu} \left(\bar{\rho}_{i,j} - L\phi_{i,j}^{m-1} \right), \quad (2.22)$$

where μ is the (i, j) entry on the diagonal of the matrix operator L . Note that even though $\frac{1}{\mu} \propto \Lambda_{i,j}$, this cancels with the operator's denominator. We then perform one sweep of Gauss-Seidel relaxation on the full cells, with either red or black ordering, while holding the partial-cell values fixed. The partial cells, along with the full cells used in the stencil for (2.14), define a region of overlap between the two domains. Although we can provide no convergence analysis for this approach ([16] might provide a good starting point), the convergence rates for the entire multi-grid procedure demonstrate its efficacy.

2.4.2 Coarse problem definition

The grid hierarchy is generated by coarsening the finest discretization of the bound-

ary, as follows. The coarse grid spacing in each direction is twice that of the fine grid, and a coarse grid's apertures and normals are defined exactly like those of a fine grid: intersection points of the domain boundary with coarse-cell edges define the apertures, which in turn define A^f and \mathbf{n} (Figure 2.5). However, a coarse cell's volume is defined as the sum of its corresponding four fine-cell volumes; this is required to maintain the flux-difference form of (2.14). The interface gradient stencil is then determined from this coarse interface representation. This definition of the geometry does have one drawback: it still requires that the interface not cross any *coarse* cell edge more than once. In addition, the limitations of the finite-difference stencil for q^f must be considered. On very coarse grids, these constraints can be violated; we use this criteria to determine the coarsest level in our multi-grid hierarchy.

Prolongation and Restriction operators

The residual is restricted to the coarser grid by volume-weighted averaging; the definition of the coarse volume then ensures that a constant $\bar{\rho}$ is coarsened properly. The finite-difference stencil for the gradient on coarser grids is found in the same manner as on the fine grid, but using the coarsened discretization. On the coarsest grid, we apply the point-relaxation procedure as many times as there are valid points. This is the simplest option, and requires no additional memory or data structures. The coarse correction is then treated as piecewise constant on all cells when interpolating back up the grid hierarchy. This is the least expensive approach, and point-relaxation quickly redistributes coarse corrections locally.

2.5 Adaptive Mesh Refinement

Oftentimes, the solution provided by a single, uniform discretization of the domain may not be accurate enough. Large gradients in the solution or variation in the domain

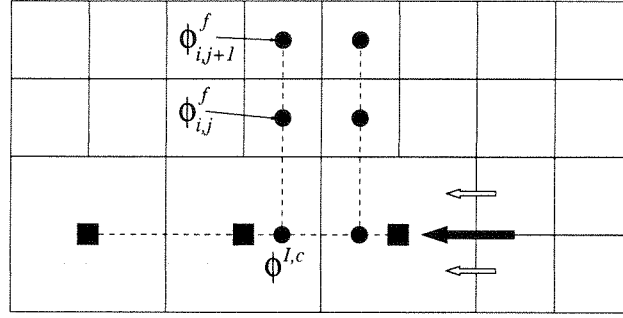


Figure 2.6: The stencil at the coarse-fine interface is represented. A value is found on a fine-level grid line, from quadratic interpolation on the coarse level. This value, along with two values on the fine grid, is used to calculate a gradient at the coarse-fine interface. The coarse-grid stencil can be shifted when necessary. Finally, the coarse-grid flux (shaded arrow at right) is defined as the sum of the corresponding fine-grid fluxes (white arrows).

boundary can require a finer grid spacing than is available with limited computer resources. An adaptive mesh hierarchy enables one to increase grid resolution where necessary; such an approach can greatly reduce the memory required to obtain a given level of accuracy. Our algorithm uses block-grid refinement, based on the work of Berger and Olinger [10]. This permits us to use simple computational data structures, instead of a linked, quad- or oct-tree object (as used in [30], for example). Our adaptive algorithm is based mostly on work and source code from Martin and Cartwright [42], for adaptive solution of Poisson's equation on rectangular domains. They used refinement factors of two and higher in their work; here we will use only factors of two, to simplify our discussion.

2.5.1 Coarse-fine interface fluxes and interpolation stencils

We first define each level of the grid hierarchy is defined as a union of rectangular blocks, all with the same grid spacing. After defining a base level, whose quantities are denoted by the superscript $l = 0$, successively finer levels $l = 1, 2, \dots$ are laid only on top of each coarser level. The valid region of each level is that which is not covered by any finer level. We also require that each level is properly nested on top of the next coarser one. This is best expressed by saying that boundaries may only occur between neighboring levels in

the hierarchy. In additions, we use grid blocks of at least eight cells; with the factor of two refinement, this guarantees our ability to calculate boundary conditions at the coarse-fine interface, described below.

In the valid region of each level, we use (2.14) to discretize (Eq. I). Again, the burden of consistency falls on the discretization of edge gradients, due to the finite volume approach. Interfaces between fine and coarse levels have fluxes that are defined by the sum of the more-accurate fine level fluxes (Figure 2.6). Specifically, the flux on level l is given by

$$F_{i+\frac{1}{2},j}^l \equiv F_{p+\frac{1}{2},q}^{l+1} + F_{p+\frac{1}{2},q+1}^{l+1} ,$$

where p and q are the appropriate indices on the finer level, $l+1$. These fine level fluxes are calculated just as before, except the edge gradients use one-sided difference stencils except for a value interpolated from nearby coarse-level cells. This interpolated value enforces the continuity of the solution between the coarse and fine levels.

As is implied in Figure 2.6, a quadratic polynomial is fitted to the values in three coarse-grid cells lying beside the fine grid. Then, a second quadratic polynomial is fitted to the values normal to the boundary, using two fine grid points and the value interpolated from coarse-grid cells. The gradient is then evaluated at the coarse-fine interface. Specifically, the edge gradient in Figure 2.6 would be calculated with

$$(G\phi)_{i,j-\frac{1}{2}}^{l+1} = \frac{\phi_{i,j}^{l+1} - \Phi^c}{3\Delta y} + \frac{\phi_{i,j+1}^{l+1} - \Phi^c}{5\Delta y} , \quad (2.23)$$

where Φ^c is the value interpolated from the coarse grid ϕ^l . This procedure results in second-order accurate fluxes at the coarse-fine interfaces, which in turn means the discretization of (Eq. I) has a first-order truncation error in cells along the coarse-fine interface. However, the coarse-fine interface is a one-dimension smaller set, so we expect the error in the solution to still be $O(\Delta x^2)$.

2.5.2 Relationship to multi-grid iteration strategy

A detailed description of this algorithm, and the multi-grid iteration scheme used to solve the resulting linear system, can be found in [42] or [46]. The changes required to extend this algorithm to our embedded-boundary method are straightforward. The point relaxation scheme on a level is that described in the previous section, suitably modified to account for the coarse-fine boundary conditions. The averaging and interpolation operators that transmit information between AMR levels are also the same described above. However, we must change the residual-correction structure of multi-grid in order to accomodate the coarse-fine interface conditions.

Define the multi-grid restriction or averaging operator to be \mathcal{A} , and the prolongation operator to be \mathcal{P} . The multi-grid algorithm can then be given by this series steps:

- Calculate the residual on the fine level, $R^l = \rho^l - L\phi^l$.
- Perform point relaxation on the correction equation, $L\psi^l = R^l$, with $\psi^{l-1} = 0$.
- Calculate a new residual, $R^l = \rho^l - L(\phi^l + \psi^l)$.
- Wherever level $l - 1$ is covered by level l , set to $R^{l-1} = \mathcal{A} R^l$.

This procedure is then repeated on level $l - 1$, and so on down the level hierarchy. At the coarsest level, the standard multi-grid procedure described above is used. We now have estimates for the correction ψ^l on all levels, so we traverse back up the hierarchy:

- Add $\mathcal{P} \psi^{l-1}$ into ψ^l .
- Find $R^l = \rho^l - L\psi^l$ on level l and beneath level $l + 1$, using ψ^{l-1} at the boundaries.
- Perform point relaxation on another correction equation, $L\chi^l = R^l$, with $\chi^{l-1} = 0$.
- Add χ^l into ψ^l .
- Finally, add ψ^l into ϕ^l .

This is continued for all levels, up to the top level. The whole process is repeated, until the maximum of the residual on all levels is less than some specified tolerance.

2.5.3 Refinement criterion

We use Richardson extrapolation of the error to determine what regions might benefit from refinement. Because of the form of the truncation error is known for this method, it is easily approximated from the discrete solution ϕ^l . We can approximate the truncation error on the next coarser level, τ^{l-1} , using the averaging operator, \mathcal{A} :

$$\tau^{l-1} \approx \mathcal{A}\rho^l - L\mathcal{A}\phi^l.$$

For full cells away from coarse-fine boundaries, this quantity is $O(\Delta x^2)$. When τ is greater than some specified tolerance, the four fine cells contributing to it are tagged for grid refinement. To simplify the implementation of the embedded boundary algorithm, we also refine all partial cells, along with a suitably sized buffer for the gradient stencils. In addition, we know that the truncation error at the coarse-fine interface is $O(\Delta x)$, so that values of τ along it are ignored. Finally, all tagged cells are grouped into block grids of some minimum size, using an algorithm developed by Berger and Rigoutsos [11]. These block grids are then used to define the next finest level in the hierarchy.

2.6 Software Implementation

The algorithm is implemented in a hybrid C++ and FORTRAN code, where complex organizational tasks are accomplished using a C++ class library, called `BoxLib` [53]. A discussion of the broader software issues that have been dealt with in `BoxLib` can be found in the paper by Crutchfield and Welcome [22]. In this section, we will briefly review `BoxLib` and our programming approach for the numerical algorithm.

2.6.1 Description of BoxLib

We have chosen to use the C++ class library **BoxLib** [53] to implement our algorithm. **BoxLib** was developed by researchers at the Center for Computational Science and Engineering at the Lawrence Berkeley National Laboratory. It is based on a dimension-independent approach, and was specifically designed for block-refined implementations of adaptive mesh refinement.

BoxLib is based on a global regular index space, which is used to define rectangular regions called **Box**'s. Thus, in two dimensions, a **Box** is defined by its lower-left and upper-right global indices, (i_1, j_1) and (i_2, j_2) . Contiguous **FORTRAN**-type arrays, called **Fab**'s, are then built on top of this, thereby allowing a single data structure to manage the array's bounds and number of components, as well as pointers to the underlying memory. The concepts of cell-centered, edge-centered, and node-centered **Fab**'s are easily obtained by shifting one or both of a **Box**'s indices by a half. This allows single-block calculations to be implemented in **FORTRAN**, which is highly optimized for such structures.

However, the greatest benefit of **BoxLib** is its library of tools for adaptive mesh refinement. In particular, unions of **Box**'s and **Fab**'s can be defined on the same index space, and managed as single units. Thus the concept of a level in the adaptive hierarchy is more easily defined. Thus, the point-relaxation scheme described in the previous sections is easily implemented as follows:

1. Define a union of cell-centered **Fab**'s that overlap by two indices. The overlap region defines the traditional "ghost" cells use in finite-difference calculations.
2. Copy data from each **Fab**'s neighbor into its ghost cells.
3. For **Fab**'s on the domain boundaries, fill ghost cells using the boundary conditions.
4. For **Fab**'s that abut the coarser level, use the interpolation scheme in the ghost cells.
5. Apply the point-relaxation algorithm to each **Fab** independently.

This algorithm’s difficulty is shown in steps 2, 3, and 4; they involve determining the neighbor bordering a given **Fab**. **BoxLib** includes lists that maintain this information, and uses the global index space to help coarse and fine **Fab**’s to interact efficiently.

Using **BoxLib**, traditional finite-difference algorithms can be more easily extended to a block-refined, adaptive framework. In addition, the resulting algorithm is implemented in C++, where more emphasis can be placed on object-oriented programming and reusability of the individual parts.

2.6.2 Description of **AmrPoisson**

The class structure we use was developed by Martin and Cartwright [42], to implement a multi-grid algorithm for solving Poisson’s equation. The top-level data structure, called **AMRPoisson**, contains the problem parameters and manages individual levels of refinement. Each of these **AMRLevel**’s contains the data and functions necessary to perform multi-grid iterations on its level. This includes classes for applying the discrete operator, filling in ghost-cell values, and performing point-relaxation sweeps.

2.6.3 Sparse data structures

We have defined two classes that maintain the additional information needed at the embedded boundary, and are contained within the **AMRLevel** class. On each **Fab** in a given refinement level, we define a **VoF** class, or volume-of-fluid representation of the embedded boundary. This includes the geometric data, such as the apertures, partial volume, and interface normal, in each cell. We also define the **EmbBndry** class, which contains a **VoF**, in addition to functions and data for applying the operator at the embedded boundary. Finally, a union of **EmbBndry**’s is defined, which is used to perform operations on given level’s union of **Fab**’s.

2.6.4 Grid generation classes

There are two aspects to grid generation that need to be considered in our approach. The first, is the ability to generate a group of block-refined grids just from a set of cells tagged for refinement. This is accomplished through the class `GridGenerator`, which is based on an algorithm by Berger and Rigoutsos [11], and implemented by [42]. Given a group of tagged cells, the class is designed to find a union of `Box`'s, each with some minimum length or width, which covers all the tagged cells. This union of boxes is then used by `AMRPoisson`, to add additional resolution using another level of refinement.

The second form of grid generation is finding the geometric information in a `VoF`, given some analytical description of the embedded boundary. We have therefore developed a `Curve` class, whose purpose is to generate the piecewise-linear description required by the algorithm. This is accomplished by discretizing the analytical formula into a polygonal curve, and using this representation to intersect cell edges and define the `VoF`. In this form, it is much easier to tranverse the polygon, and determine intersection with grid lines. Details of the grid generation algorithm will be given in the next section. Note also that the coarsening strategy of the multi-grid algorithm requires `VoF`'s on coarser levels of refinement to be derived from finer ones. To this end, the `Curve` class provides functions which properly coarsen a union of `VoF`'s.

2.6.5 Graphics libraries

Finally, we wish to give a brief description of the graphics libraries that we have used for representing our data structures. We have developed our visualization tools using `VIGL` [27], which is a set of graphics functions using `X` and `Tcl/Tk`, and was written by Allen Downey. Elements that can be drawn include graphics primitives, such as points, lines, and polygons, and more complicated structures such as `Fab`'s. A simple user interface provides advanced capabilities, such as zooming, inspection of a `Fab`'s arrays point-by-point or with

slices, and drawing contour or quiver plots. Finally, objects can be dynamically created and destroyed, as well as used to generate output in other formats. We have extended these libraries to include `BoxLib` primitives, such as `Box`'s and unions of `Fab`'s, in addition to graphical interpretations of more complicated objects, like a `VoF` or `EmbBndry`. In all, these routines have been an invaluable aid for interactive debugging, and for an informative presentation of results.

2.7 Results

We have chosen three simple problems to demonstrate the the single-grid and adaptive algorithms, and verify the method's characteristics as laid out in the previous sections. First, we will describe our method of grid generation and its influence on the geometric properties. Then we define the discrete norms that will be used to measure the solution's accuracy. We will then proceed to the description of the three test problems, and an interpretation of their results.

Grid Generation Algorithm

The boundary $\partial\Omega$ is discretized by first representing it as a parametrization, $r(\theta)$; we then choose points in this parametrization, which are no more than $\alpha\Delta x$ apart (the following calculations use $\alpha = 0.3$). By connecting these points, we form a piecewise-linear representation of the interface. The intersections of this representation, with the finest level's grid lines, define the apertures; these then define the area of the front from (2.13). We also include two modifications to this grid generation algorithm. The first is related to the grid requirements described above and in [49]; if the boundary representation enters and leaves a cell through the same edge, the intersections with that edge are ignored. In effect, this "clips" the boundary where it crosses the same cell edge twice. The second modification, which occurs rarely in practice, is to adjust the boundary to remove cells with

volume fractions $\Lambda < 10^{-6}$. This is done by shifting the boundary points to the nearest intersection of grid lines, thus making the cell volume zero. In this process, the boundary is moved by at most 0.1% of the *grid spacing*; this incurs a negligible error relative to that of the numerical algorithm.

Definition of Discrete Norms

The norms used in the following analysis warrant some additional explanation. We must first separate the computational domain into the adaptive levels, Ω^l , where l is the number of cells per unit length, in each grid direction. For example, Ω^{80} designates the set of all valid cells with a grid spacing of $\Delta x = \frac{1}{80}$, that are not covered by a finer level (*i.e.*, Figure 2.12(a)). The entire computational domain is then defined by $\Omega = \bigcup \Omega^l$. We note again that the domain boundary is represented only on the finest level, which can be divided into two sets of cells: Ω_I^l , which consists of full cells; and Ω_P^l , consisting of partial cells. The set of cells on the finest level is then just $\Omega^l = \Omega_I^l \cup \Omega_P^l$. Unless otherwise noted, cell counts for a level Ω^l refer to the number of uncovered full or partial cells in Ω^l ; this excludes both dummy values outside the domain, and values covered by finer levels.

We can now define a volume-weighted norm of a variable e on some set of cells, Ω_k :

$$\|e\|_p^{\Lambda, \Omega_k} = \left(\sum_{(i,j) \in \Omega_k} |e_{i,j}|^p \Lambda_{i,j} V^l \bigg/ \sum_{(i,j) \in \Omega_k} \Lambda_{i,j} V^l \right)^{\frac{1}{p}}, \quad (2.24)$$

where $V^l = \Delta x^2$ is the full cell volume on a given level. An unweighted norm, $\|\cdot\|_p^{\Omega_k}$, merely removes $\Lambda_{i,j}$ from equation (2.24). Any ∞ -norm, $\|\cdot\|_\infty^{\Omega_k}$ is just the maximum value over the cells in Ω_k . We can now define the rate of convergence between two norms, e_1 and e_2 , with two different grid spacings h_1 and h_2 , as

$$r = \log \left(\frac{e_1}{e_2} \right) \bigg/ \log \left(\frac{h_1}{h_2} \right).$$

Thus with $h_1 < h_2$, a rate of $r = 1$ for the two errors e_1 and e_2 indicates a first-order accurate method.

2.7.1 Problem 1: Comparison with an exact solution

For the first two problems, the domain interior is defined by

$$\Omega = \{(r, \theta) : r \leq 0.30 + 0.15 \cos 6\theta\}.$$

The boundary is sufficiently complicated to test the algorithm, without compromising the requirements of the finite-difference stencil mentioned in Section 3.

We first set $\beta = 1$, to demonstrate several results for Poisson's equation. The values of the right-hand side are given by the exact Laplacian of the solution,

$$\nabla \cdot \nabla \varphi = 7r^2 \cos 3\theta,$$

evaluated at the centroid of each finite-difference cell. This is done to best represent the average value in (2.14). Note that fourth-order derivatives of φ are discontinuous at the origin, and higher-order derivatives are singular. Dirichlet boundary conditions on $\partial\Omega$ are specified by the exact solution:

$$\varphi(r, \theta) = r^4 \cos 3\theta,$$

which has a maximum value of about ± 0.041 at $r = 0.45$ on $\partial\Omega$. The exact solution φ enters into the discretization by taking its value at the midpoint of the front in each cell, and using this as the value of Φ^f in (2.17). A plot of the discrete solution is given in Figure 2.7, with 40 evenly-spaced contours between $\varphi = \pm 0.0412$.

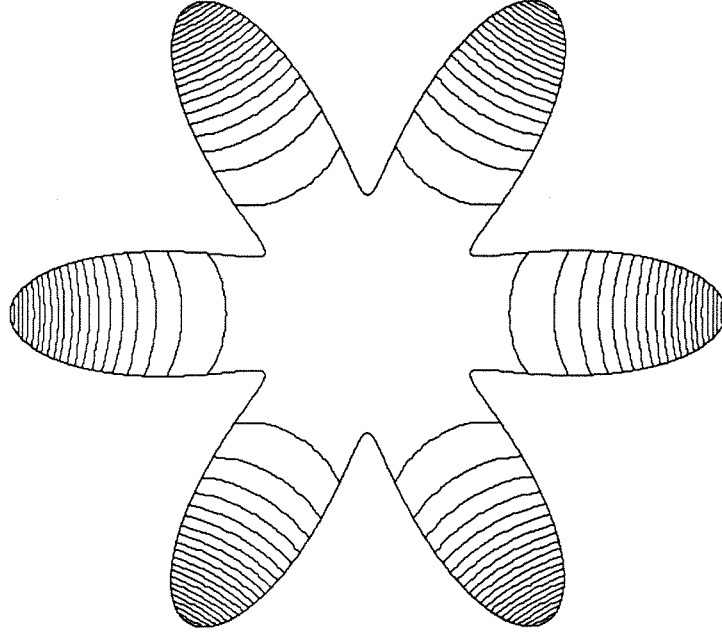


Figure 2.7: Contour plot of the solution to Problem 1.

Truncation Error

We will first analyze the algorithm with uniform grid spacing over the domain. We compute the truncation error τ , defined in (2.19), for this solution. In Table 2.1, we can still see that $\Lambda\tau$ is $O(\Delta x)$ on Ω_P , consistent with the error estimate (2.20). In the same table we see that the interior truncation error (τ on Ω_I), which is due to the standard five-point difference scheme, is $O(\Delta x^2)$. Because the domain is star-shaped, we can use θ as an independent variable as we walk along the interface. In Figure 2.8(a), we plot $\Lambda\tau$ versus the angle θ , in Ω_P only (partial cells); obviously, it is certainly not a smooth function, and contains a substantial high-wavenumber component. This is due to the error being dependent on many non-smooth factors, such as the apertures and distance to interpolation lines. In Figure 2.8(b), we plot the volume fraction Λ as a function of θ . We see that there is no “blow up” in $\Lambda\tau$ for small Λ .

N	$\ \Lambda\tau\ _{\infty}^{\Omega_P}$	r	$\ \Lambda\tau\ _1^{\Omega_P}$	r	N^P	$\ \tau\ _{\infty}^{\Omega_I}$	r	N^I
40	$1.20 \cdot 10^{-1}$		$2.63 \cdot 10^{-2}$		208	$1.66 \cdot 10^{-3}$		400
80	$7.71 \cdot 10^{-2}$	0.64	$1.45 \cdot 10^{-2}$	0.86	420	$4.15 \cdot 10^{-4}$	2.0	1824
160	$4.20 \cdot 10^{-2}$	0.88	$7.35 \cdot 10^{-3}$	0.98	856	$1.04 \cdot 10^{-4}$	2.0	7712
320	$2.18 \cdot 10^{-2}$	0.95	$3.73 \cdot 10^{-3}$	0.98	1716	$2.59 \cdot 10^{-5}$	2.0	31716
640	$1.11 \cdot 10^{-2}$	0.98	$1.89 \cdot 10^{-3}$	0.98	3416	$6.49 \cdot 10^{-6}$	2.0	128604

Table 2.1: The norms and convergence rates of the partial-volume-weighted truncation error are presented for Problem 1. The values in partial cells are first-order in the grid spacing, while values in the interior are second-order.

Error induced by the truncation error

We can also measure the error in the discrete solution, ξ , defined in equation (2.21). However, to elucidate the resulting behavior, we have also computed solutions to two subproblems. Specifically, we solve

$$\begin{aligned} L\xi_P &= \tau_P \\ L\xi_I &= \tau_I \end{aligned} \tag{2.25}$$

where τ_P , τ_I are, respectively, equal to the truncation error on the partial and full cells, and zero elsewhere. In that case, $\xi = \xi_P + \xi_I$, and ξ_P , and ξ_I represent the contributions of the error from the interior and the irregular boundary, respectively. In Table 2.2 we see that ξ_P converges at a rate $r \approx 3$ in the ∞ -norm. Our explanation of this behavior is the potential-theoretic model for the error on the partial cells described at the end of Section 2. The partial cells induce a dipole distribution on the boundary, due to the homogeneous Dirichlet boundary condition for the error equation. The field induced by this dipole distribution is $O(\Delta x^3)$, uniformly in the range of values taken on by the Λ 's. However, ξ_I is strictly second-order accurate. Table 2.2 demonstrates that the overall solution error converges at a rate of $r \approx 3$ for coarser grids, and is then only second-order accurate for finer grids.

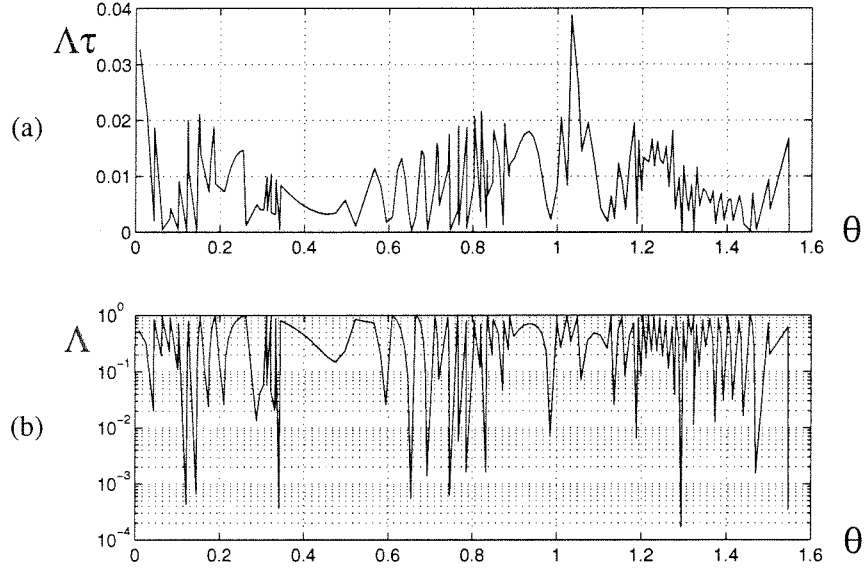


Figure 2.8: For Problem 1, we plot (a) the magnitude of the volume-weighted truncation error and (b) the partial volume, on Ω_P versus θ . Note that the former is bounded, even for arbitrarily small volumes.

Figure 2.9 demonstrates this for the one-norm. Essentially, the error on coarser grids is dominated by the effect of the truncation error in partial cells; for finer grids, the effect of the interior truncation error begins to dominate. Both ξ_P and ξ_I , converge to zero at the stated asymptotic rates; however, their sum does not settle down to its asymptotic rate until $\xi_I \gg \xi_P$. This leads to some anomalous behavior in the convergence rate for ξ . For example, the rate of convergence for $\|\xi\|_1^\Omega$ appears to be less than second order, even though both summands are converging at rates greater than or equal to second order. The reason for this is easily seen in Figure 2.9. At the grid spacing where $\|\xi_P\|$ and $\|\xi_I\|$ are comparable, there is partial cancellation between the two components of the error. At the finer grid spacings, as that cancellation diminishes because of the more rapid convergence of ξ_P , the convergence rate of ξ decreases slightly as it asymptotes to ξ_I .

N	$\ \xi_P\ _\infty^\Omega$	$\ \xi\ _\infty^\Omega$	r	$\ \xi_P\ _1^\Omega$	r	$\ \xi_I\ _1^\Omega$	r	$\ \xi\ _1^\Omega$	r
40	$5.89 \cdot 10^{-5}$	$5.85 \cdot 10^{-5}$		$9.33 \cdot 10^{-6}$		$1.38 \cdot 10^{-6}$		$8.34 \cdot 10^{-6}$	
80	$7.35 \cdot 10^{-6}$	$7.36 \cdot 10^{-6}$	3.0	$1.33 \cdot 10^{-6}$	2.8	$3.97 \cdot 10^{-7}$	1.8	$1.02 \cdot 10^{-6}$	3.0
160	$1.17 \cdot 10^{-6}$	$1.17 \cdot 10^{-6}$	2.6	$1.76 \cdot 10^{-7}$	2.9	$1.05 \cdot 10^{-7}$	1.9	$1.07 \cdot 10^{-7}$	3.2
320	$1.67 \cdot 10^{-7}$	$1.68 \cdot 10^{-7}$	2.8	$2.27 \cdot 10^{-8}$	3.0	$2.70 \cdot 10^{-8}$	2.0	$1.80 \cdot 10^{-8}$	2.6
640	$2.26 \cdot 10^{-8}$	$2.27 \cdot 10^{-8}$	2.9	$2.86 \cdot 10^{-9}$	3.0	$6.84 \cdot 10^{-9}$	2.0	$5.02 \cdot 10^{-9}$	1.8

Table 2.2: We present the two components of the solution error in Problem 1, and their corresponding convergence rates. The error due to truncation error in the partial cells converges at a substantially higher rate than that due to the interior truncation error.

Benefits of AMR

With this in mind, we can also demonstrate some benefits of adaptive mesh refinement for this problem, even though the right-hand side is evenly distributed over the whole domain. Table 2.3 shows three cases using adaptive mesh refinement:

- Case 1: Two levels of refinement, with coarsest level Ω^{80} .
- Case 2: Two levels of refinement, with coarsest level Ω^{160} .
- Case 3: Three levels of refinement, with coarsest level Ω^{160} .

In each case, we refine only the boundary region, subject to the constraint that each grid block has at least eight points in each direction (similar to Figure 2.12(b)). By refining around the boundary, we can reduce the impact of the larger truncation error there. For example, the error in the adaptive solution for Case 1 and Case 2, is roughly that of the finest grid, yet both require fewer points than a calculation with uniform grid spacing. However, the effect of the interior truncation error is seen again in Case 3; the algorithm is not able to improve the solution significantly without global refinement, since the truncation error in the interior is evenly distributed for this problem.

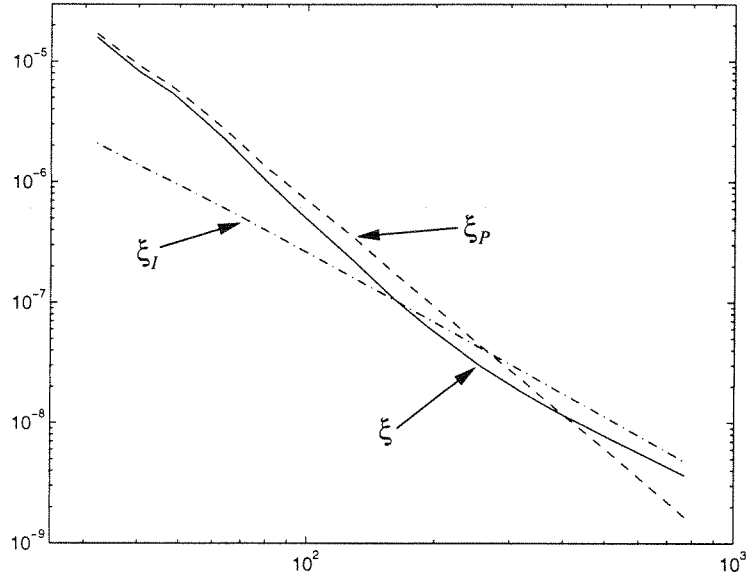


Figure 2.9: Plot of the one-norm of the error in the discrete solution, ξ , and its two components: ξ_P , from the $O(\Delta x)$ truncation error on Ω_P ; and ξ_I , from the $O(\Delta x^2)$ truncation error on Ω_I . Note that ξ_P converges like $O(\Delta x^3)$, while ξ_I converges as $O(\Delta x^2)$.

2.7.2 Problem 2: Including variable coefficients

Here we include variation in the coefficient β ,

$$\beta(r, \theta) = 1 - r^2,$$

which is evaluated at the midpoint of the actual edges in the finite-difference cell of Figure 2.3. The right-hand side is then given by

$$\nabla \cdot \beta \nabla \varphi = (7r^2 - 15r^4) \cos 3\theta,$$

so that the exact solution is the same as in the first problem. Again, the solution is evaluated at cell-centers when calculating the truncation error, and the right-hand side is evaluated at cell centroids. We can see from Table 2.4 that the non-adaptive cases have results similar

	Level	Ω^{80}	Ω^{160}	Ω^{320}	Ω^{640}	Overall
Case 1	N in Ω^l	896	4984			5880
	$\ \xi\ _{\infty}^{\Omega^l}$	$1.05 \cdot 10^{-6}$	$1.36 \cdot 10^{-6}$			$1.36 \cdot 10^{-6}$
	$\ \xi\ _1^{\Omega^l}$	$2.87 \cdot 10^{-7}$	$2.24 \cdot 10^{-7}$			$2.51 \cdot 10^{-7}$
Case 2	N in Ω^l		5568	11160		16728
	$\ \xi\ _{\infty}^{\Omega^l}$		$2.36 \cdot 10^{-7}$	$2.58 \cdot 10^{-7}$		$2.58 \cdot 10^{-7}$
	$\ \xi\ _1^{\Omega^l}$		$9.39 \cdot 10^{-8}$	$4.49 \cdot 10^{-8}$		$7.75 \cdot 10^{-8}$
Case 3	N in Ω^l		4480	9664	21684	35828
	$\ \xi\ _{\infty}^{\Omega^l}$		$2.79 \cdot 10^{-7}$	$3.20 \cdot 10^{-7}$	$1.50 \cdot 10^{-7}$	$3.20 \cdot 10^{-7}$
	$\ \xi\ _1^{\Omega^l}$		$1.02 \cdot 10^{-7}$	$8.40 \cdot 10^{-8}$	$2.35 \cdot 10^{-8}$	$8.40 \cdot 10^{-8}$
Uniform	$N \in \Omega^l$	2244	8568	33432	132020	\uparrow
	$\ \xi\ _{\infty}^{\Omega^l}$	$7.36 \cdot 10^{-6}$	$1.17 \cdot 10^{-6}$	$1.68 \cdot 10^{-7}$	$2.27 \cdot 10^{-8}$	\Leftarrow
	$\ \xi\ _1^{\Omega^l}$	$1.02 \cdot 10^{-6}$	$1.07 \cdot 10^{-7}$	$1.80 \cdot 10^{-8}$	$5.02 \cdot 10^{-9}$	

Table 2.3: Adaptive mesh refinement errors for the solution in Problem 1. Each case uses different levels of refinement. We also give the results for uniform grid spacing; arrows indicate which sets of values to compare.

to those of the constant coefficient problem.

Multi-grid convergence

We can also analyze the effectiveness of the multi-grid algorithm for this problem. Each multi-grid iteration applies the point-relaxation scheme four times (*i.e.*, four full sweeps of Gauss-Seidel relaxation), before and after the coarse-grid correction is applied. Figure 2.10 plots the norm of the residual,

$$\|\Lambda(L\phi^m - \bar{\rho})\|_{\infty}^{\Omega},$$

versus iteration number, m , for all the calculations on a uniform grid. The solution ϕ is initialized to zero, so that the initial residual grows as $O(\Delta x^{-2})$, due to the inhomogeneous boundary conditions. Our multi-grid algorithm reduces the residual by about an order of

N	$\ \Lambda\tau\ _\infty^\Omega$	r	$\ \xi\ _\infty^\Omega$	r	$\ \xi\ _1^\Omega$	r
40	$9.60 \cdot 10^{-2}$		$5.86 \cdot 10^{-5}$		$8.16 \cdot 10^{-6}$	
80	$6.17 \cdot 10^{-2}$	0.64	$7.30 \cdot 10^{-6}$	3.0	$9.57 \cdot 10^{-7}$	3.1
160	$3.36 \cdot 10^{-2}$	0.88	$1.17 \cdot 10^{-6}$	2.6	$9.58 \cdot 10^{-8}$	3.3
320	$1.75 \cdot 10^{-2}$	0.94	$1.68 \cdot 10^{-7}$	2.8	$2.00 \cdot 10^{-8}$	2.3
640	$8.84 \cdot 10^{-3}$	0.99	$2.28 \cdot 10^{-8}$	2.9	$6.00 \cdot 10^{-9}$	1.7

Table 2.4: We list errors and convergence rates for Problem 2. The results are very similar to those obtained in Problem 1.

magnitude per iteration, even as its norm approaches the cutoff of 10^{-11} . There is a slight decrease in performance as the grid spacing is reduced, so that the reduction rates are around 8.5 for the finest grid. The adaptive cases are shown in Figure 2.10, also. Even with the coarse-fine interface relations, we are able to obtain nearly an order of magnitude reduction in the residual per iteration, despite the unsophisticated interpolation operator.

2.7.3 Problem 3: Grid Convergence Study

We also wish to test the algorithm for problems without analytic solutions, to demonstrate that values centered outside the domain do not cause problems. We solve Laplace's equation on $\Omega = \Upsilon_1 \cap \Upsilon_2$, where

$$\Upsilon_1 = \{(r, \theta) : r \geq 0.25 + 0.05 \cos 6\theta\}$$

and Υ_2 is the unit square centered at the origin. The Dirichlet boundary conditions are $\phi = 1$ on $\partial\Upsilon_1$, and $\phi = 0$ on $\partial\Upsilon_2$.

Discrete maximum principle

A plot of the solution is given in Figure 2.11; we can see that it extends smoothly outside of $\partial\Upsilon_1$, and overshoots the boundary condition as a result. For equal grid spacing

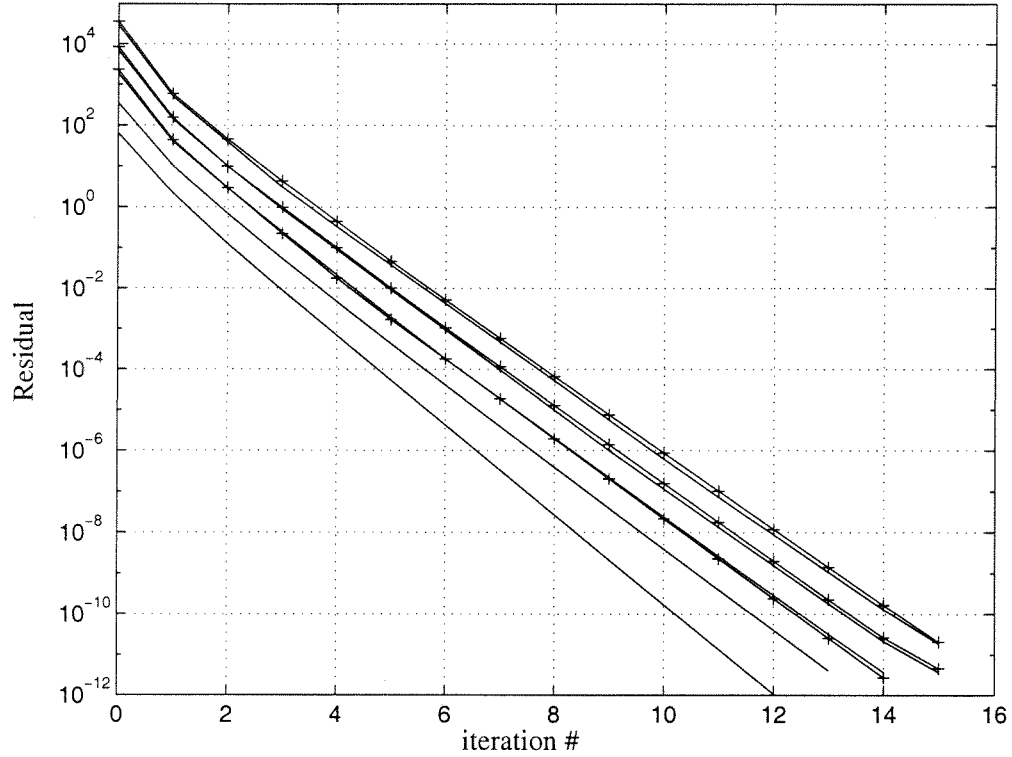


Figure 2.10: Plot of the ∞ -norm of the partial volume-weighted residual, versus multi-grid iteration m , for the single-grid (solid) and adaptive-grid (solid lines with plus signs) solutions of Problem 2.

in both directions, the maximum principle for Laplace's equation dictates that there should be no overshoot for values of $\Lambda < \frac{1}{2}$, while values in cells with $\Lambda > \frac{1}{2}$ should be greater than one. Table 2.5 shows the number of cells, k , in Ω_P that violate this criterion; we see that it is variable, but small, with respect to the total number of points in Ω_P . For the finest grid, $N = 640$ and $k = 0$. We assume that this violation occurs when a partial volume, $\Lambda \approx \frac{1}{2}$, is in a region with significantly under-resolved gradients.

Comparison of different discrete solutions

To evaluate the convergence of the algorithm for the single grid case, we compare two solutions, with a factor of two difference in grid spacing. To do this comparison, we must

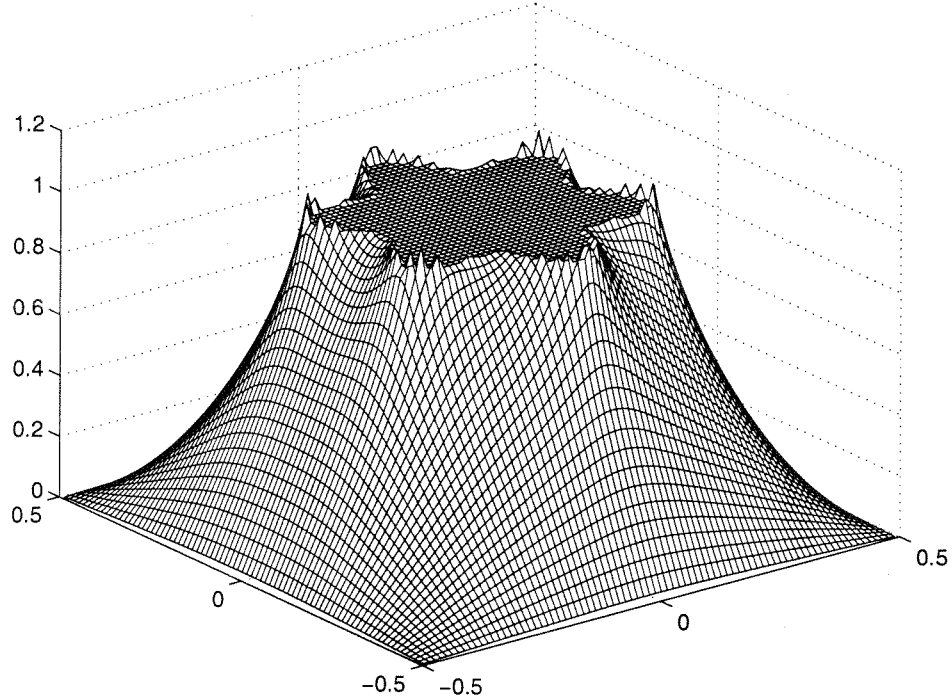


Figure 2.11: Surface plot of the solution to Problem 3, for $N = 80$.

interpolate the solution on the fine level, to the coarse grid cell centers. This is done with bilinear interpolation, denoted with B , between the four fine-level solution values closest to the coarse grid cell center, as in Figure 2.5. We can then define the error as the difference between the two results:

$$\xi^l = B^{l+1}_l \phi^{l+1} - \phi^l \text{ on } \Omega^l_I .$$

We do this only for interior cells on the coarse grid, because the values needed on the fine level, in order to interpolate to the center of a partial cell on the coarse-grid, are not necessarily available. Table 2.5 contains the convergence rates for this error, which are roughly second-order in both norms.

N	$\ \xi\ _{\infty}^{\Omega_I}$	r	$\ \xi\ _1^{\Omega_I}$	r	k/N^P
40	$1.27 \cdot 10^{-2}$		$2.82 \cdot 10^{-3}$		8/104
80	$4.32 \cdot 10^{-3}$	1.6	$6.28 \cdot 10^{-4}$	2.2	0/208
160	$1.27 \cdot 10^{-3}$	1.8	$1.50 \cdot 10^{-4}$	2.1	4/408
320	$2.45 \cdot 10^{-4}$	2.4	$3.00 \cdot 10^{-5}$	2.3	0/820

Table 2.5: We present results for Problem 3. The error between successive levels is approximately second-order in the grid spacing. In addition, the last column indicates that relatively few cells violate a discrete maximum principle.

Efficacy of AMR

To demonstrate the AMR algorithm, we compare the solution on each level, to the solution with the finest uniform grid spacing, i. e., we replace ϕ^{l+1} above with ϕ^{640} , where Ω^{640} is the finest grid level in this case. Although the resulting errors are not appropriate for calculating convergence rates, they are accurate up to the error on the finest grid. In Table 2.6, we see that the AMR algorithm is able to improve the accuracy of the method substantially, by merely refining around the boundaries. In Case 1, with two levels, we are able to obtain results with the accuracy of the finest grid, with only 35% of the points; in Case 2, with three levels, this holds true with 17%. Figure 2.12(a) demonstrates the valid regions on which norms are computed, for for Case 1; Figure 2.12(b) plots the solution and block grid structure.

2.8 Conclusions

The algorithm described in this chapter have several important characteristics. The finite-volume formulation uses second-order accurate gradients for calculating surface fluxes. These gradients are calculated from cell-centered quantities, even when those centers are outside the domain. The truncation error for the resulting discretization for (Eq. I) is first-order in the mesh spacing only along the domain boundary, and second-order in the

	Level	Ω^{80}	Ω^{160}	Ω^{320}	Overall
Case 1	N in Ω_I^l	4288	3096		7384
	$\ \xi\ _\infty^{\Omega_I^l}$	$6.64 \cdot 10^{-4}$	$1.49 \cdot 10^{-3}$		$1.49 \cdot 10^{-3}$
	$\ \xi\ _1^{\Omega_I^l}$	$1.48 \cdot 10^{-4}$	$3.71 \cdot 10^{-4}$		$1.82 \cdot 10^{-4}$
Case 2	N in Ω_I^l	3792	3664	6148	13604
	$\ \xi\ _\infty^{\Omega_I^l}$	$1.58 \cdot 10^{-4}$	$1.43 \cdot 10^{-4}$	$2.46 \cdot 10^{-4}$	$2.46 \cdot 10^{-4}$
	$\ \xi\ _1^{\Omega_I^l}$	$2.83 \cdot 10^{-5}$	$6.42 \cdot 10^{-5}$	$7.45 \cdot 10^{-5}$	$3.83 \cdot 10^{-5}$
Uniform	N in Ω_I^l	5016	20248	81476	\uparrow \leftarrow
	$\ \xi\ _\infty^{\Omega_I^l}$	$5.49 \cdot 10^{-3}$	$1.49 \cdot 10^{-3}$	$2.45 \cdot 10^{-4}$	
	$\ \xi\ _1^{\Omega_I^l}$	$8.06 \cdot 10^{-4}$	$1.80 \cdot 10^{-4}$	$3.00 \cdot 10^{-5}$	

Table 2.6: Adaptive mesh refinement errors for the solution in Problem 3. Errors are found by comparing the solution to values interpolated from the finest result. The last row contains results for uniform grid spacing; arrows indicate which set of values to compare.

interior. In our three test problems, the solution is found to be second-order accurate, on domains with significant curvature and variation. Convergence rates of $O(\Delta x^3)$ were also observed for larger Δx , which are attributed to the truncation error at the boundary using a potential-theoretic argument. The Dirichlet boundary condition regards the partial-volume weighted truncation error as a dipole on the domain boundary. Because each dipole has an $O(\Delta x^4)$ field, the $O(N)$ boundary points cause only an $O(\Delta x^3)$ error in the solution.

Our analysis in one dimension demonstrates that our discretization is well-conditioned, even in the presence of arbitrarily small or thin cells. In addition, the multi-grid algorithm uses only a simple point-relaxation scheme, with volume-weighted restriction and piecewise constant prolongation operators, and we obtain nearly the same multi-grid reduction rates for the residual, regardless of grid size or quality. This suggests that we retain a well-conditioned system in more than one dimension.

We also demonstrate that our method is amenable to the introduction of adaptive mesh refinement to improve the accuracy locally. This refines the cells containing portions of the domain boundary, and simultaneously refines the geometry description, while reducing

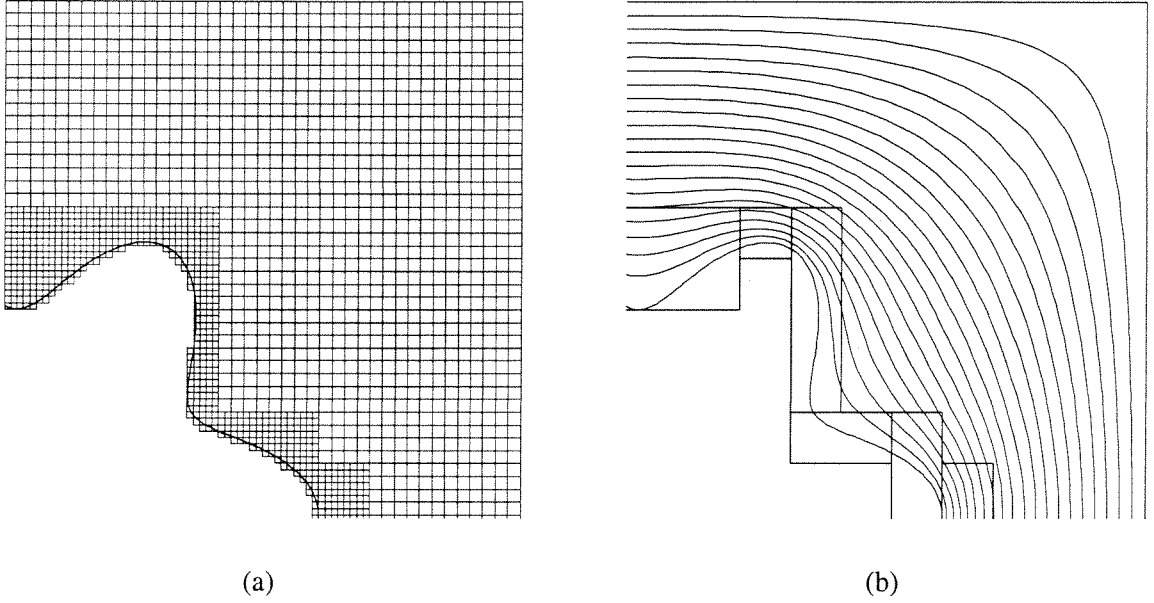


Figure 2.12: We present two plots of the first quadrant of Problem 3, Case 1. (a) represents the valid regions of each level, and (b) gives grid block boundaries, and contours of the solution.

the effect of the larger truncation error. The multi-grid framework attains reduction rates for the adaptive grid hierarchy that are comparable to those obtained on uniform grids.

Chapter 3

Heat Equation

3.1 Introduction

In this chapter we extend our discretization of (Eq. I) to a second model problem, the heat equation with spatially-varying thermal conductivity, on a fixed domain:

$$\varphi_t = \nabla \cdot \beta \nabla \varphi \text{ on } \Omega, \text{ with } \varphi(\mathbf{x}, 0) = f(\mathbf{x}) . \quad (\text{Eq. II})$$

As with (Eq. I), we impose either Dirichlet- or Neumann-type boundary conditions on $\partial\Omega$, with $\beta(x, y) > 0$ a given quantity. Our approach is again based on a finite volume formulation, which is derived by integrating (Eq. II) over each irregular cell, and for a single time step, Δt , from time t^n to t^{n+1} . The resulting integrals are then evaluated using appropriate quadrature rules. The spatial discretization is the same as in the previous chapter: volume integrals are centered at the irregular cell centroids, while surface integrals use the midpoint rule. The temporal discretization for (Eq. II) is determined by choosing quadrature formulas for the time integrals. Evaluating the integrand at t^n results in an explicit, finite-volume approach, for which the maximum stable time step is severely limited, because of the arbitrarily-small cell volumes. This leads us to consider implicit discretizations of (Eq.

II), such as the backward-Euler, Crank-Nicholson, and implicit Runge-Kutta (RK) methods. In addition, we consider another implicit, but non-conservative, approach, and provide another means of maintaining discrete conservation.

The algorithms for (Eq. II) are presented as in the last chapter. In Section 2, we first derive the finite volume approach in one dimension, using different temporal discretizations; this allows us to determine the accuracy and stability characteristics of each approach. In Section 3, we extend the approaches to two dimensions. Sections 4 and 5 describe the modifications to the multi-grid algorithm, and resulting software implementation, respectively. Finally, in Section 6 we present numerical results to support our arguments.

3.2 Derivation in One Dimension

We will first develop our discretization of the heat equation in one spatial dimension,

$$\varphi_t = \beta \varphi_{xx} \text{ for } x \in [0, \ell] \text{ and } t \in [0, \infty), \quad (3.1)$$

where $\beta > 0$ is a constant. This will allow us to provide an analysis similar to that of the Poisson equation in the previous chapter. To complete the problem we specify the initial conditions, and Dirichlet boundary conditions at each endpoint:

$$\varphi(x, t = 0) = \varphi_0(x), \text{ with } \varphi(x = 0, t) = \Phi^0(t) \text{ and } \varphi(x = \ell, t) = \Phi^f(t). \quad (3.2)$$

We then use the same spatial discretization that was used for the Poisson equation in one dimension, namely $N - 1$ cells of cell Δx , and the last cell N is bordered by the domain boundary at $x = \ell$, and has volume fraction $\Lambda \in (0, 1]$. The independent variable ϕ is still cell-centered, at points $x_i = (i - \frac{1}{2})\Delta x$, while cell edges are denoted by $x_{i+\frac{1}{2}}$ for $i = 0, \dots, N - 1$.

3.2.1 Finite volume derivation

To discretize (3.1), we first integrate it over the control volume $x \in [x_L, x_R]$, and the time interval $t \in [t^n, t^{n+1}]$, and apply the divergence theorem to obtain:

$$\begin{aligned} \int_{x_L}^{x_R} \varphi(x', t^{n+1}) dx' - \int_{x_L}^{x_R} \varphi(x', t^n) dx' \\ = \beta \int_{t^n}^{t^{n+1}} \varphi_x(x_R, t') dt' - \beta \int_{t^n}^{t^{n+1}} \varphi_x(x_L, t') dt'. \end{aligned} \quad (3.3)$$

This form of (3.1) merely states that the total change of φ in the control volume, is balanced by its diffusive fluxes through the cell edges. We will now apply this to the control volume defined by each cell i .

Approximation of spatial integrals

The spatial integrals in (3.3) will be approximated differently for full and partial cells. For full cells, we expand the integral about the cell midpoint, x_i , using the Taylor series for φ evaluated at time t^n :

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \varphi(x', t^n) dx' = \Delta x \varphi(x_i, t^n) + C_1 \Delta x^3 \varphi_{xx}(x_i, t^n) + O(\Delta x^5). \quad (3.4)$$

Note that we have kept an extra term here, in order to obtain more specific results for the truncation error later. For partial cells, we will use a lower order approximation, based on values at the cell center,

$$\int_{x_{N-\frac{1}{2}}}^{\ell} \varphi(x', t^n) dx' = \Lambda \Delta x (\varphi(x_N, t^n) + C_2 \Delta x \varphi_x(x_N, t^n) + O(\Delta x^2)), \quad (3.5)$$

where the constant C_2 is uniformly bounded in Λ .

Approximation of time integrals

Similarly, the integral of the heat flux across any cell edge can be approximated using values of φ_x at t^n , t^{n+1} , or an average of the two. These in turn lead to the familiar forward Euler, backward Euler, and Crank-Nicholson discretizations of the heat equation. More specifically, given the time step $\Delta t = t^{n+1} - t^n$, using a value at t^n implies

$$\int_{t^n}^{t^{n+1}} \varphi_x \left(x_{i+\frac{1}{2}}, t' \right) dt' = \Delta t \varphi_x \left(x_{i+\frac{1}{2}}, t^n \right) + C_3 \Delta t^2 \varphi_{xt} \left(x_{i+\frac{1}{2}}, t^n \right) + O(\Delta t^3), \quad (3.6)$$

while a value at t^{n+1} gives

$$\int_{t^n}^{t^{n+1}} \varphi_x \left(x_{i+\frac{1}{2}}, t' \right) dt' = \Delta t \varphi_x \left(x_{i+\frac{1}{2}}, t^{n+1} \right) + C_4 \Delta t^2 \varphi_{xt} \left(x_{i+\frac{1}{2}}, t^{n+1} \right) + O(\Delta t^3), \quad (3.7)$$

Finally, averaging φ_x at the new and old times produces the result

$$\begin{aligned} \int_{t^n}^{t^{n+1}} \varphi_x \left(x_{i+\frac{1}{2}}, t' \right) dt' &= \frac{\Delta t}{2} \left(\varphi_x \left(x_{i+\frac{1}{2}}, t^n \right) + \varphi_x \left(x_{i+\frac{1}{2}}, t^{n+1} \right) \right) \\ &\quad + C_5 \Delta t^3 \varphi_{xtt} \left(x_{i+\frac{1}{2}}, t^{n+\frac{1}{2}} \right) + O(\Delta t^5). \end{aligned} \quad (3.8)$$

Note that the extra term in these expansions will be used below, to help determine each algorithm's truncation error.

Finite volume derivation of finite difference schemes

Given these three different representations for the flux integral, we can now choose discretizations for (3.3) that correspond to the three classical finite difference methods mentioned above. We demonstrate this first for full cells, and then extend the result for partial cells, later. The forward Euler method is derived by combining (3.4) with (3.6) with

(3.3) to obtain

$$\begin{aligned}
\Delta x (\varphi(x_i, t^{n+1}) - \varphi(x_i, t^n)) &= \beta \Delta t \left(\varphi_x \left(x_{i+\frac{1}{2}}, t^n \right) - \varphi_x \left(x_{i-\frac{1}{2}}, t^n \right) \right) \\
&\quad - C_1 \Delta x^3 (\varphi_{xx}(x_i, t^{n+1}) - \varphi_{xx}(x_i, t^n)) \\
&\quad + C_3 \Delta t^2 \left(\varphi_{xt} \left(x_{i+\frac{1}{2}}, t^n \right) - \varphi_{xt} \left(x_{i-\frac{1}{2}}, t^n \right) \right) \\
&\quad + O(\Delta x^5) + O(\Delta t^3 \Delta x).
\end{aligned} \tag{3.9}$$

The first line of (3.9) represents the forward Euler discretization of the heat equation. The second and third lines represent the quadrature error for the various integrals in (3.3). We note that the term in parentheses on the second line is a simple difference in time of φ_{xx} , which is $O(\Delta t)$. Similarly, the term in parentheses on the third line is a difference in space of φ_{xt} , and is $O(\Delta x)$. The $O(\Delta x^5)$ and $O(\Delta t^3 \Delta x)$ terms on the last line also come from similar cancellations. We can therefore conclude that the quadrature formulas for (3.9) lead to an $O(\Delta t \Delta x (\Delta x^2 + \Delta t))$ truncation error in full cells.

This derivation of quadrature errors was necessary, in order to determine the accuracy of the resulting finite difference algorithm. To continue, we use a centered difference approximation of φ_x at edges,

$$G\phi_{i+\frac{1}{2}}^n = \frac{\phi_{i+1}^n - \phi_i^n}{\Delta x} \quad \text{for } i = 1, \dots, N-1. \tag{3.10}$$

If we then discretize φ using its values at cell centers, and at t^n and t^{n+1} , $\phi_i^{e,n+1} = \varphi(x_i, t^{n+1})$ and $\phi_i^{e,n} = \varphi(x_i, t^n)$, then the truncation error for (3.10) is

$$G\phi_{i+\frac{1}{2}}^{e,n} = \frac{\phi_{i+1}^{e,n} - \phi_i^{e,n}}{\Delta x} \tag{3.11}$$

$$= \varphi_x \left(x_{i+\frac{1}{2}}, t^n \right) + C_6 \Delta x^2 \varphi_{xxx} \left(x_{i+\frac{1}{2}}, t^n \right) + O(\Delta x^4). \tag{3.12}$$

When we substitute (3.4) and (3.10) into (3.9), and divide by $\Delta t \Delta x$, we obtain the forward

Euler method

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \beta \frac{G\phi_{i+\frac{1}{2}}^n - G\phi_{i-\frac{1}{2}}^n}{\Delta x} . \quad (3.13)$$

After replacing ϕ with ϕ^e , the difference of terms involving φ_{xxx} in (3.12) becomes one higher order, $O(\Delta x^3)$, so that the method's truncation error becomes

$$\frac{\phi_i^{e,n+1} - \phi_i^{e,n}}{\Delta t} = \beta \frac{G\phi_{i+\frac{1}{2}}^{e,n} - G\phi_{i-\frac{1}{2}}^{e,n}}{\Delta x} + O(\Delta x^2 + \Delta t) . \quad (3.14)$$

This is the expected result; the forward Euler algorithm is known to be accurate to second-order in space and first-order in time. However, we have arrived at this by a route that is more complicated than a simple Taylor expansion of (3.1) about (x_i, t^n) . By using quadrature rules to approximate the integrals in (3.3), we have been able to demonstrate the cancellations necessary to obtain the truncation error estimate (3.14). Of course, using (3.7) instead of (3.6) to approximate integrals of fluxes leads to the classical backward Euler method for the heat equation, while (3.8) leads to the Crank-Nicholson method, which result in first- and second-order accurate methods, respectively.

As a final detail, we define the flux at the left boundary using the formula from the previous chapter, centered at the correct time,

$$G\phi_{\frac{1}{2}}^n = \frac{1}{3\Delta x} (9\phi_1^n - \phi_2^n - 8\Phi^0(t^n)) .$$

This reduces the spatial accuracy of (3.14) to first order in the left-most cell.

Finite volume derivation for partial cells

We will now derive the forward Euler method for the partial cells, based on substituting (3.5) and (3.6) into (3.3). This produces

$$\begin{aligned}
\Lambda \Delta x (\varphi(x_N, t^{n+1}) - \varphi(x_N, t^n)) &= \beta \Delta t \left(\varphi_x(\ell, t^n) - \varphi_x(x_{N-\frac{1}{2}}, t^n) \right) \\
&\quad - C_2 \Lambda \Delta x^2 (\varphi_x(x_N, t^{n+1}) - \varphi_x(x_N, t^n)) \\
&\quad + C_4 \Delta t^2 \left(\varphi_{xt}(\ell, t^n) - \varphi_{xt}(x_{N-\frac{1}{2}}, t^n) \right) \\
&\quad + O(\Lambda \Delta x^3) + O(\Delta t^3 \Delta x).
\end{aligned} \tag{3.15}$$

Note that we obtain another cancellation involving the difference of spatial integrals in time; the difference of φ_x terms on the second line becomes $O(\Delta t)$. We also obtain the same cancellation in the time integrals as before, so that the difference of terms in φ_{xt} is $O(\ell - x_{N-\frac{1}{2}})$, which is $O(\Lambda \Delta x)$.

Again, we substitute finite differences of ϕ for terms in φ and φ_x , to find the truncation error in partial cells. The last thing we need is an approximation of φ_x at the right boundary, $x = \ell$. We will use the same discretization that was introduced in the previous chapter, but with the proper time centering:

$$G^f \phi^n = \frac{1}{d_2 - d_1} \left(\left(\Phi^f(t^n) - \phi_{N-1}^n \right) \frac{d_2}{d_1} - \left(\Phi^f(t^n) - \phi_{N-2}^n \right) \frac{d_1}{d_2} \right), \tag{3.16}$$

where d_1 and d_2 are the distances from the front to cell centers x_{i-1} and x_{i-2} , respectively. When we substitute the discretized exact solution ϕ^e into this expression we obtain the local truncation error

$$\begin{aligned}
G^f \phi^n &= \frac{1}{d_2 - d_1} \left(\left(\Phi^f(t^n) - \phi_{N-1}^n \right) \frac{d_2}{d_1} - \left(\Phi^f(t^n) - \phi_{N-2}^n \right) \frac{d_1}{d_2} \right) \\
&= \varphi_x(\ell, t^n) + C_7 \Delta x^2 \varphi_{xxx}(\ell, t^n) + O(\Delta x^3),
\end{aligned} \tag{3.17}$$

where C_7 is bounded uniformly in Λ and Δx . Substituting (3.5), the approximation of the volume integral in cell N , and (3.16) into (3.15), and dividing by $\Lambda \Delta t \Delta x$, leads to

$$\frac{\phi_N^{n+1} - \phi_N^n}{\Delta t} = \beta \frac{G^f \phi^n - G \phi_{N-\frac{1}{2}}^n}{\Lambda \Delta x} . \quad (3.18)$$

Setting $\phi = \phi^e$ leads to an expression for the local truncation error,

$$\frac{\phi_N^{e,n+1} - \phi_N^{e,n}}{\Delta t} = \beta \frac{G^f \phi^{e,n} - G \phi_{N-\frac{1}{2}}^{e,n}}{\Lambda \Delta x} + O\left(\frac{\Delta x}{\Lambda} + \Delta x + \Delta t\right) . \quad (3.19)$$

This is similar to the result from the last chapter, where our spatial discretization led to a truncation error in the last cell that was singular for $\Lambda \rightarrow 0$, but was well behaved when multiplied by the partial volume, Λ .

We now use the notation for the discretized self-adjoint elliptic operator, L , from Chapter 2, so that the forward Euler method may be written concisely as

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = L \phi_i^n . \quad (3.20)$$

The backward Euler method is given by

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = L \phi_i^{n+1} , \quad (3.21)$$

so that it has the same truncation error as the forward Euler method. Finally, the Crank-Nicholson method is given by

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{2} \left(L \phi_i^{n+1} + L \phi_i^n \right) , \quad (3.22)$$

and its truncation error can be derived from (3.4), (3.5) and (3.8), along with the results of Chapter 2. For interior cells, $i = 2, \dots, N-1$, and the truncation error is $O(\Delta t^2 + \Delta x^2)$,

while for the leftmost ($i = 1$) cell it is $O(\Delta t^2 + \Delta x)$. Finally, for $i = N$, the truncation error is $O(\Delta t^2 + \Delta x + \frac{\Delta x}{\Lambda})$. Note that in order for the Crank-Nicholson method to be second-order accurate in the interior cells, we must have $\beta \Delta t \propto \Delta x$ in the limit of $\Delta t, \Delta x \rightarrow 0$.

Truncation Error Analysis

We can analyze the effect of the truncation error for these methods, in a fashion similar to that done for the Poisson equation in one dimension, using the modified equation for the method. We again evaluate the truncation error τ for the Crank-Nicholson scheme by applying (3.22) ϕ^e ,

$$\tau_i^n = \frac{\phi_i^{e,n+1} - \phi_i^{e,n}}{\Delta t} - \frac{1}{2} \left(L\phi_i^{e,n} + L\phi_i^{e,n+1} \right) .$$

We have shown that τ is $O(\Delta t^2 + \Delta x^2)$ in the interior, and $O(\Delta t^2 + \frac{\Delta x}{\Lambda})$ in cell N .

If we set $\xi = \phi - \phi^e$ we obtain an evolution equation for the error,

$$\frac{\xi_i^{n+1} - \xi_i^n}{\Delta t} = \frac{1}{2} (L\xi_i^n + L\xi_i^{n+1}) - \tau_i^n , \quad (3.23)$$

where ξ satisfies homogeneous Dirichlet boundary conditions. For the first time step, $\xi^0 = 0$, assuming that ϕ is initialized with ϕ^e . For the first time step, this implies

$$\xi_i^1 - \frac{\Delta t}{2} L\xi_i^1 = \frac{\Delta t}{2} \tau_i^0 . \quad (3.24)$$

As in the previous chapter, we can use this system of equations to obtain an order of magnitude approximation for the error, ξ , in terms of Δx and Δt .

We will not try to provide a rigorous bound on ξ for (3.24), as was done in the last chapter. Instead, we will explore how the truncation error in the last cell, τ_N , affects ξ after one time step. We will consider the particular case of $\beta \Delta t \propto \Delta x$, as $\Lambda, \Delta t, \Delta x \rightarrow 0$. As mentioned above, this represents the time step constraint for the stability of most explicit

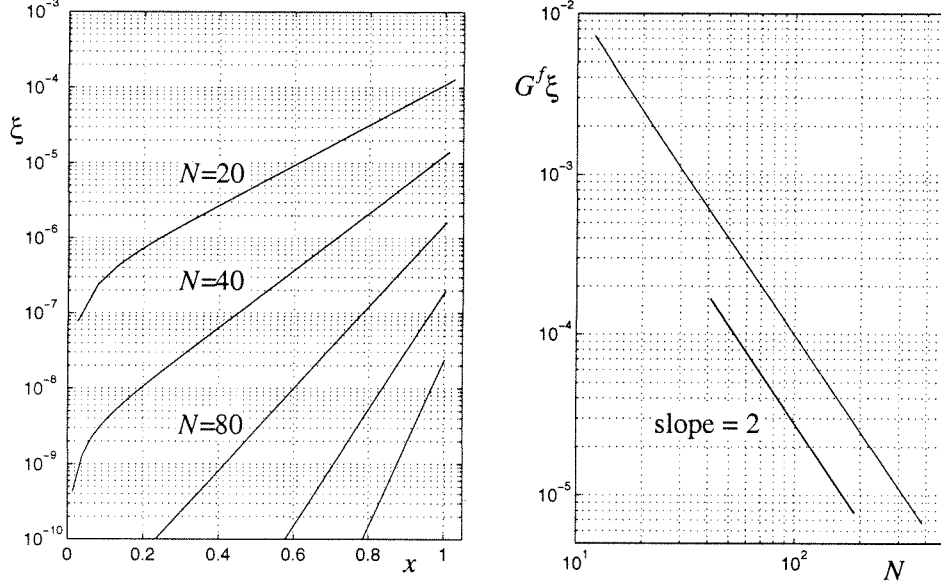


Figure 3.1: We plot the solution of (3.25) for $N = 20, 40, 80, 160, \dots$, for $\Lambda = 10^{-2}$. The error in the solution is $\xi = O(\Delta x^3)$, while the gradient at $x = \ell$ is $G^f \xi = O(\Delta x^2)$, and negative.

advection methods, as well as the proper ratio for obtaining second-order accuracy from the Crank-Nicholson discretization. If we assume that $\beta = 1$, $\Delta t = 2\Delta x$, and set $\tau_i = 0$, except for $\tau_N = \frac{\Delta x}{\Lambda}$, then (3.24) reduces to

$$\begin{aligned} \xi_i - \Delta x L \xi_i &= 0 \quad \text{for } i \neq N, \\ \xi_N - \Delta x L \xi_N &= \frac{\Delta x^2}{\Lambda}. \end{aligned} \quad (3.25)$$

It is possible to write the exact discrete solution to this equation (see [59]) on $x \in [0, 1]$. However, it is easier to determine the important characteristics of ξ from Figure 3.1, for $\Delta x = N^{-1}$ and $\Lambda = 10^{-2}$. The maximum value of ξ behaves like $O(\Delta x^3)$, and quickly decays to zero away from $x = 1$. The corresponding $G^f \xi$, also in Figure 3.1, are $O(\Delta x^2)$ in magnitude, and negative; this is because ξ is steepening with N , which is at odds with the interpolating polynomial used in computing $G^f \xi$. These trends persist for $\Lambda \rightarrow 0$, and with

other $\Delta t \propto \Delta x$. We conclude that the discretization is well-behaved as $\Lambda \rightarrow 0$, and expect that the truncation error in the partial cell will induce an error in the solution of $O(\Delta x^3)$.

Discrete Conservation

One of the desirable properties of a finite volume method is that it satisfies a discrete version of (3.3). Because the underlying principles of diffusion problems involve conservation of some physical quantity (such as mass, momentum, or energy), it can be considered desirable to have a discrete algorithm which mimics such behavior. Our finite volume formulation implies discrete conservation of energy; for example, the Crank-Nicholson scheme satisfies

$$\begin{aligned} \Delta x \sum_{i \neq N} \phi_i^n + \Lambda \Delta x \phi_N^n - \Delta x \sum_{i \neq N} \phi_i^0 - \Lambda \Delta x \phi_N^0 \\ = \frac{\beta}{2} \sum_{k=0}^{n-1} \Delta t_k \left(G^f \phi^{k+1} + G^f \phi^k \right) - \frac{\beta}{2} \sum_{k=0}^{n-1} \Delta t_k \left(G \phi_{\frac{1}{2}}^{k+1} + G \phi_{\frac{1}{2}}^k \right). \end{aligned} \quad (3.26)$$

This was derived by multiplying equation (3.22) by V_i , and summing over all cells, for n time steps. The result is obtained by noting that fluxes on interior cells cancel when summed over space, as do the intermediate spatial integrals, when summed in time.

3.2.2 Properties of update matrix

In this section we examine the eigensystems of the update operators in (3.20), (3.21), and (3.22). To accomplish this, we will use the matrix decomposition routines supplied with the MATLAB software suite [43]. This program is well-known and reliable, and we trust that its results are sufficiently accurate for the purpose of this discussion.

All the time-integration schemes use our discrete Laplacian operator, L , so it is a logical place to begin our analysis. In Figure 3.2, we plot the eigenvalues of $-L$, sorted by magnitude, for $N = 100$ and a range of $\Lambda \in [2^{-5}, 1]$. The spectrum is similar to that

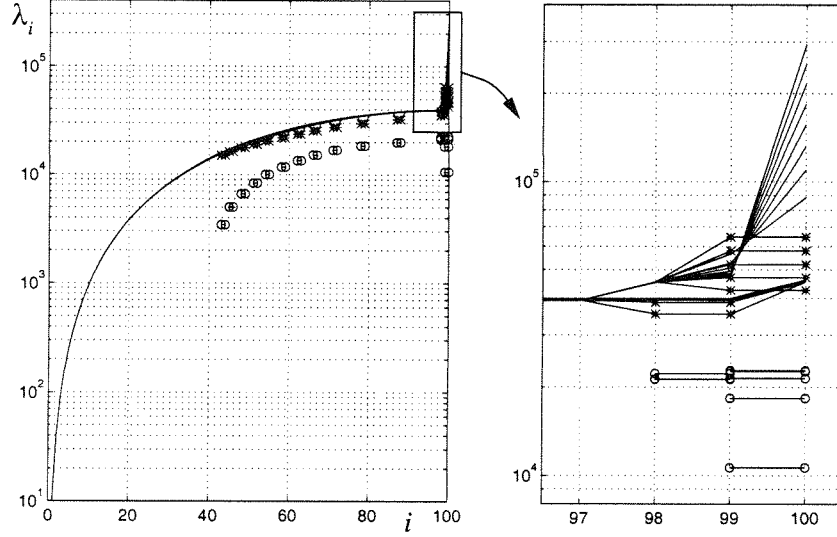


Figure 3.2: We plot the eigenvalues of $-L$ for $\Lambda \in [2^{-5}, 1]$, for $N = 100$ cells. There is a pair of complex conjugate eigenvalues for $\Lambda > 0.1$ (real and imaginary parts marked by stars and circles), for oscillatory modes corresponding to cell N . For $\Lambda \ll 1$, this eigenvalue grows as Λ^{-1} , while the eigenvector resembles a discrete delta function in cell N .

of the three-point difference stencil for the negative Laplacian [59], with eigenvalues in the range $(0, 4N^2)$, except for two features. First, we note the presence of a single pair of complex conjugate eigenvalues when $\Lambda \gtrsim 0.1$, which are marked by stars in Figure 3.2. The corresponding eigenvector's real and imaginary parts are plotted in Figure 3.3, along with its absolute value, for $\Lambda = 0.8$. We see that the eigenvectors correspond to an oscillatory mode for the last cell N , whose absolute value is smooth, but does not satisfy the boundary condition. As $\Lambda \rightarrow 0$, this mode becomes more like a discrete delta function in cell N ; the inset in Figure 3.2 shows the eigenvalue behaves like $\Lambda^{-1}N^2$. This is the expected result; in Figure 2.2 Chapter 2, we found that the partial-volume-weighted operator's condition number was well-behaved as $\Lambda \rightarrow 0$. A final anomaly in Figure 3.2 is the eigenvalue slightly larger than $4N^2$, and independent of Λ . In Figure 3.3, we see that the corresponding oscillatory eigenvector is due to the discretization in the leftmost cell. Because these specific

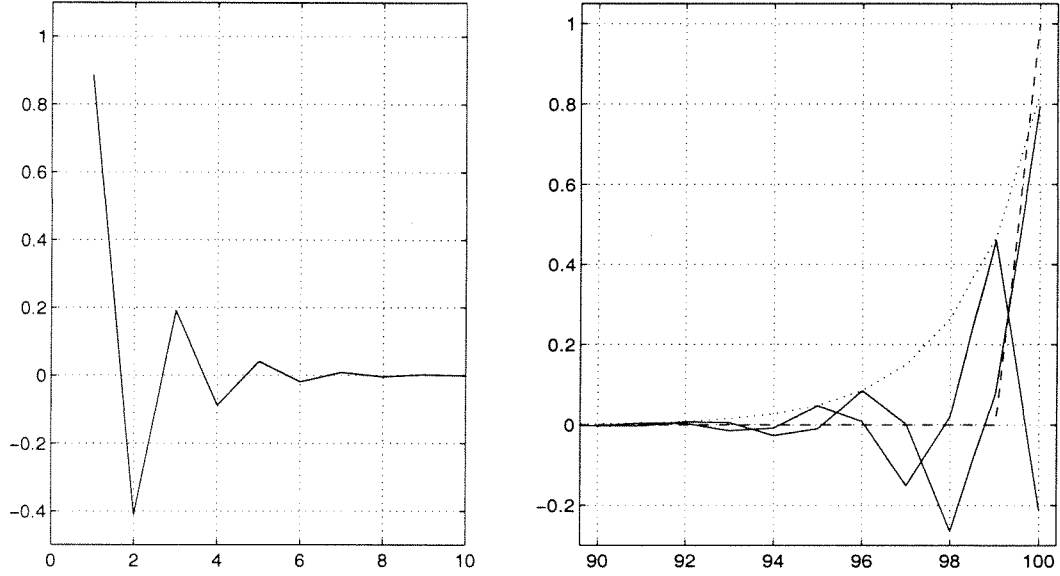


Figure 3.3: Here we plot the eigenvectors of $-L$, for $N = 100$, corresponding to the left- and rightmost cells. The mode on the left is independent of Λ , but the eigenvectors on the right are complex and oscillatory (solid lines) for $\Lambda = O(1)$, although their absolute value is smooth (dotted line). For $\Lambda \ll 1$, these modes transform into a delta function in cell N (dashed line).

oscillatory eigenmodes are not present in standard discretizations of (Eq. I) [59], we can conclude that they are due to the construction of gradients using quadratic polynomials in the first and last cells.

Having explored the spectrum of $-L$, we can now examine the properties of each time-integration scheme, and examine the effect of varying Δt , Δx , and Λ . The first case will be the forward Euler method, for which we have

$$\phi_i^{n+1} = \phi_i^n + \Delta t L \phi_i^n . \quad (3.20)$$

Thus, the amplification factor ω for an eigenvector of L (not $-L$, as in Figure 3.2), with eigenvalue λ , is

$$\omega = 1 + \Delta t \lambda .$$

For the scheme (3.20) to be stable, we must have $|\omega| \leq 1$. Given the eigenvalue distribution in Figure 3.2, this requires that the time step satisfy

$$\beta \Delta t \leq \min \left(\Lambda, \frac{1}{2} \right) \Delta x^2 . \quad (3.27)$$

As $\Lambda \rightarrow 0$, this is very restrictive. This is a well-known property of explicit, conservative finite volume methods, such as those used with hyperbolic conservation laws [49].

This time step limitation can be avoided by employing an implicit time-integration scheme. For example, the amplification factor for the backward Euler method (3.21) is

$$\omega = \frac{1}{1 - \Delta t \lambda} ,$$

while that of the Crank-Nicholson method (3.22) is

$$\omega = \frac{1 + \frac{\Delta t \lambda}{2}}{1 - \frac{\Delta t \lambda}{2}} .$$

The eigenvalues of L all have negative real parts, so that in these equations $|\omega| < 1$, and both methods are unconditionally stable, as expected. Another significant feature is that for $\lambda \ll -1$, the amplification factor for the backward Euler method goes to zero. This is because the backward Euler method is \mathcal{L}_0 -stable [37], causing the larger eigenvalues to be dissipated more quickly. This is not the case for the Crank-Nicholson discretization, because $\omega \approx -1$ for large negative λ , so that the corresponding eigenvector changes sign every time step, with minimal damping. In Figure 3.2, $|\lambda| \gg 1$ for a majority of the eigenvalues, especially in the case $\Lambda \rightarrow 0$, and we expect nearly all the eigenvectors to have this behavior for the Crank-Nicholson discretization.

3.2.3 An implicit Runge-Kutta method

The oscillatory nature of the Crank-Nicholson algorithm can cause severe problems in some instances. We mentioned earlier that our desire was to combine our method with high-resolution advection algorithms, as done in [3]. These algorithms are designed for stability in the presence of discontinuities in the advected quantity, by using slope-limiters and upwinding, for example [38]. For embedded boundary, the restrictive time step constraint (3.27) can be avoided by redistributing the conserved quantity [49], or by giving up conservation. Regardless, this lowers accuracy near the embedded boundary [20, 2], or in the presence of significant high-frequency component [54]. We expect that these algorithms will interact poorly with the Crank-Nicholson method, especially near cells with small partial volumes.

This has motivated us to explore implicit Runge-Kutta (RK) methods. RK discretizations have been studied extensively, and are a popular method of integrating initial value problems for ordinary differential equations (ODE's) [37]. Exact solutions to ODE's are often expressed as exponentials of functions, and RK methods can be viewed as rational approximations to such exponentials. The approach is applied to PDE's using the *method of lines*, in which the spatial discretization is fixed, and the unknowns are advanced in time as a system of ODE's. From this viewpoint, the three temporal discretizations presented above are RK methods, with rational approximations given by the amplification factor, ω [37]. A second-order accurate family of RK methods was presented in [61], for which the rational approximation is

$$\begin{aligned}\omega &= \frac{1 + (1 - \alpha)(\Delta t \lambda)}{1 - \alpha(\Delta t \lambda) + (\alpha - \frac{1}{2})(\Delta t \lambda)^2} \\ &= e^{\Delta t \lambda} + O(\Delta t^2 \lambda^2) .\end{aligned}$$

In [61], it was shown that for $\alpha > \frac{1}{2}$, this approximation is \mathcal{L}_0 -stable, and for $\alpha = \frac{1}{2}$ it

equation reduces to the Crank-Nicholson discretization. They also note that for $\alpha < 2 - \sqrt{2}$ or $\alpha > 2 + \sqrt{2}$, it is possible to factor the denominator using real arithmetic only, so that the discretization can be written as

$$\begin{aligned}\hat{\phi}_i &= \phi_i^n + (1 - \alpha)\Delta t L\phi_i^n + \alpha_+ \Delta t L\hat{\phi}_i \\ \phi_i^{n+1} &= \hat{\phi}_i + \alpha_- \Delta t L\phi_i^{n+1} ,\end{aligned}\tag{3.28}$$

where $\alpha_{\pm} = (2\alpha - 1)/(\alpha \pm (\alpha^2 - 4\alpha + 2)^{\frac{1}{2}})$, so that $\alpha_+ + \alpha_- = \alpha$. In two dimensions, this allows us to solve (3.28) using our standard multi-grid method, instead of having to invert a linear sum of biharmonic and Laplacian operators. So although we now have to solve two linear systems per time step, we will see below that the results are much less oscillatory than the Crank-Nicholson discretization, but achieve the same level of accuracy.

3.2.4 Hybrid method for heat equation

Another approach is to give up conservation for the cells abutting the domain boundaries, in favor of removing the oscillatory eigenmodes associated with them. One way to accomplish this is to use the \mathcal{L}_0 -stable backward Euler method near the boundary,

$$\phi_N^{n+1} - \Delta t L\phi_N^{n+1} = \phi_N^n ,$$

and the more accurate Crank-Nicholson method in the interior

$$\phi_i^{n+1} - \frac{\Delta t}{2} L\phi_i^{n+1} = \phi_i^n + \frac{\Delta t}{2} L\phi_i^n .$$

This is motivated by similar approaches that have been used for hyperbolic conservation laws [49]. In that case, the finite-volume discretization is abandoned in small irregular cells, in favor of better stability properties for the overall algorithm. Below, we will suggest how discrete conservation can be maintained without destroying the accuracy of the hybrid

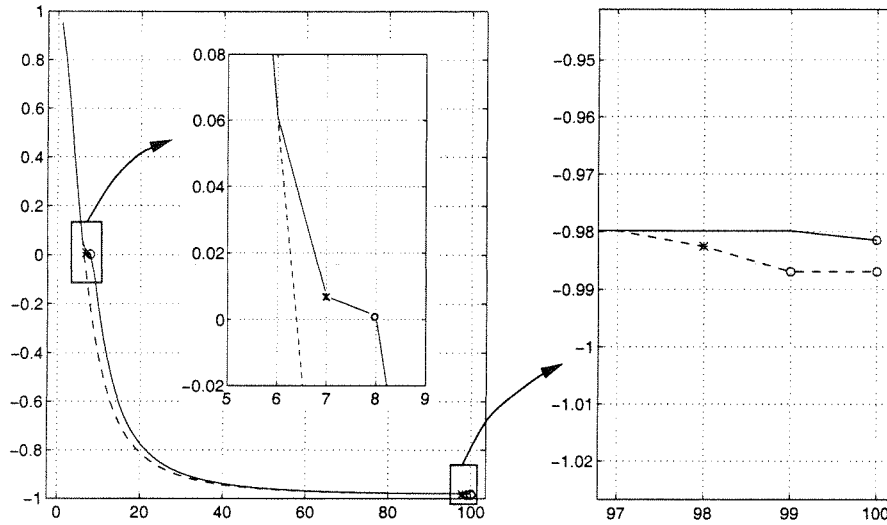


Figure 3.4: We plot the eigenvalues of the hybrid (solid line) discretization, for $\Delta t = \Delta x$, $\Lambda = 2^{-3}$, and $N = 100$ cells, and which uses the backward Euler discretization in the first and last cells. The spectrum is very similar to that of the Crank-Nicholson discretization (dashed line), except for two features. On the right, we see Crank-Nicholson's complex $\lambda \approx -1$ (circles on dashed line) have been replaced by a real $\lambda \approx -1$ (circle on solid line), and a real $\lambda \ll 1$ (same in inset). Similarly, the eigenvalue corresponding to the leftmost cell in the Crank-Nicholson discretization (star, right), has been replaced by another $\lambda \ll 1$ (star, inset).

approach.

Using the same notation as the previous section, we can look at the eigensystem of the update matrix defined by (3.22) for cells $i \neq N$, and (3.21) in cell N . Because this method is not a linear function of L , the spectrum in Figure 3.2 is no longer applicable. In Figure 3.4 we plot the eigenvalues of the update matrix, for $N = 100$, $\Delta t = \Delta x$, and $\Lambda = 2^{-3}$. We have used three different hybrid operators, which use the backward Euler method in cell $i = N$, cells $i = \{1, N\}$, and cells $i = \{1, N - 1, N\}$, respectively. In all cases, the magnitude of the amplification factor $|\omega|$ is less than one. This was also verified for $\Delta t \in [0.1\Delta x^2, 10\Delta x]$ and $\Lambda \in [0, 1]$, so that the schemes seem to be stable in this range of values.

In comparing the eigenvectors of the Crank-Nicholson and hybrid operators, each eigenvector is scaled to have unit magnitude in the Euclidean norm. An eigenvector v_i is then

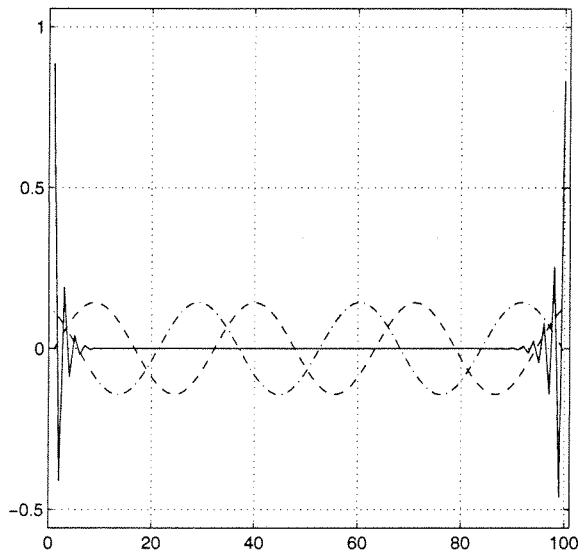


Figure 3.5: We plot the eigenvectors for the first and last cells, for the hybrid and Crank-Nicholson discretizations corresponding to Figure 3.4. The undamped eigenvector of the Crank-Nicholson discretization (solid line on left) has been replaced by a heavily damped eigenvector (dashed-dotted) in the hybrid discretization. Crank-Nicholson's complex eigenvectors corresponding to cell N have been replaced with an undamped real mode (solid line on right), and a heavily damped mode (dashed).

associated with cell i if its absolute value in that cell is the largest, for all the eigenvectors. For $\Lambda \gtrsim 0.1$, we see in Figure 3.5 that the complex eigenvectors have disappeared, but have been replaced by a single, undamped eigenmode in the right-most cell, much like that in Figure 3.3 for the left-most cell. The second-largest eigenvector in that cell corresponds to low-frequency, damped mode (Figure 3.3, left inset). For the second hybrid method, the oscillatory mode is removed, while for the last method, we see a return of the mode similar to a discrete delta function in cell N , pictured in Figure 3.5. However, the corresponding eigenvalue in Figure 3.4 is now much smaller, suggesting that it will be severely damped in the time-integration scheme. We must conclude that the hybrid approach is able to damp the large-eigenvalue-modes that persisted in the Crank-Nicholson method, in this range of parameters.

Calculation of conservative losses

However, if we still desire a conservative discretization, we must now account for the flux mismatch between cells that use the Crank-Nicholson and backward Euler methods. This lack of conservation $\delta\phi$ at edge $i+\frac{1}{2}$ is found by subtracting the two different fluxes in (3.7) and (3.8):

$$\delta\phi_i^{n+1} = \frac{\beta\Delta t}{2\Delta x} \left(G\phi_{i-\frac{1}{2}}^n - G\phi_{i-\frac{1}{2}}^{n+1} \right). \quad (3.29)$$

Note that $\delta\phi$ is $O(\Delta t)$ for $\Delta t \propto \Delta x$, and thus one order larger than the $O(\Delta t^2)$ local truncation error for the backward Euler method. We have chosen to associate $\delta\phi$ with cell i , so that it will be damped by the hybrid method when it is re-incorporated. If the transition occurs in cell N , for example, then $\delta\phi$ included in the next time step,

$$\phi_N^{n+1} = \phi_N^n + \Delta t L \phi_N^{n+1} + \Lambda^{-1} \delta\phi_N^n, \quad (3.30)$$

so that conservation is lagged until the following time step. We can see how this affects the solution by noting that (3.30) is the same as (3.25), except that $\delta\phi^n = O(\Delta t)$ is one order lower in Δt . However, our numerical results below will show that the solution error after $O(N)$ steps is still $O(\Delta t^2)$.

3.3 Results in One Dimension

We may now apply the numerical algorithm to a series of test problems, to verify the analysis in the previous sections. In this discussion, we will use the same definitions that were given in Chapter 2. Discrete norms are given by $\|\cdot\|$, with subscripts ∞ , 1, and 2 denoting max-, 1-, and 2-norms, respectively. We will again use Ω_P to represent all cells that contain a portion of the irregular boundary; all other valid cells make up the interior, Ω_I , so that the computational domain is given by $\Omega = \Omega_P \cup \Omega_I$.

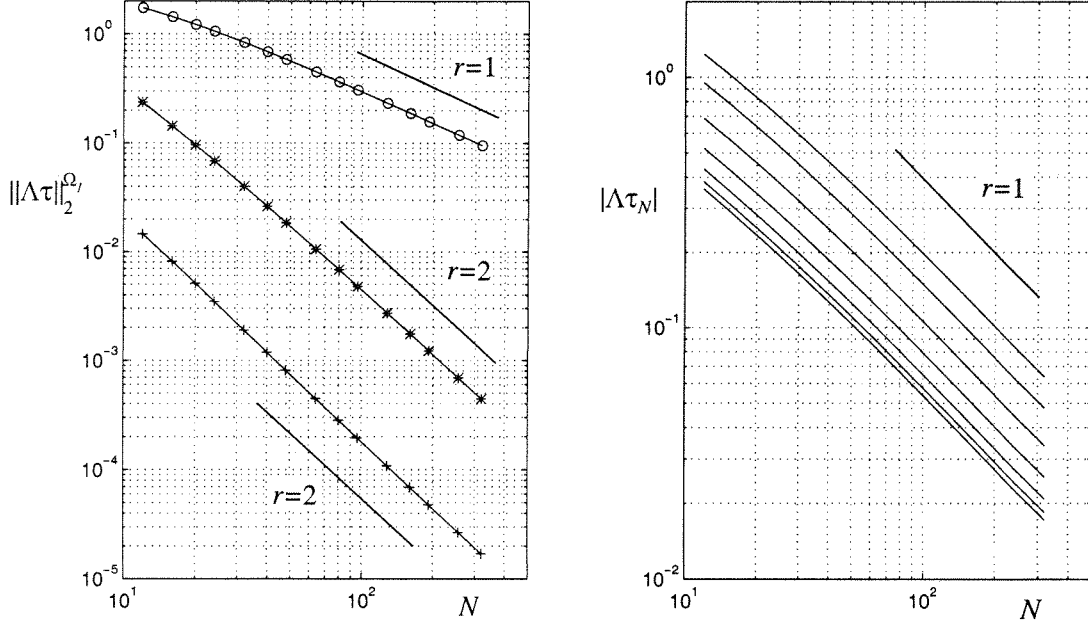


Figure 3.6: We plot the truncation error for each of the methods using the exact solution (3.31). In the interior of the domain, the backward Euler method is first-order accurate (circles), while the Crank-Nicholson (stars) and Runge-Kutta (plus signs) discretizations are second order accurate. On the right, the partial-volume-weighted truncation error in the last cell, $\Lambda\tau_N$, is $O(\Delta x)$ and well-behaved for over a range of Λ and N (top line $\Lambda = 1$, lowest line $\Lambda = 2^{-6}$).

3.3.1 Decay of a sinusoid

For β constant, the exact solutions to (Eq. II) on simple domains can be found using the “separation of variables” technique. In particular, consider (3.1) with homogeneous Dirichlet boundary conditions and $\beta = 1$, on the interval $x \in [0, 1]$, for which

$$\varphi(x, t) = e^{-\pi^2 \beta t} \sin \pi x \quad (3.31)$$

is an exact solution, for $t \geq 0$. This can be used to verify the truncation error arguments of Section 2, and determine the effect of the larger boundary errors on the discrete solution.

We will first verify the estimates (3.20)-(3.22) for the truncation error after one

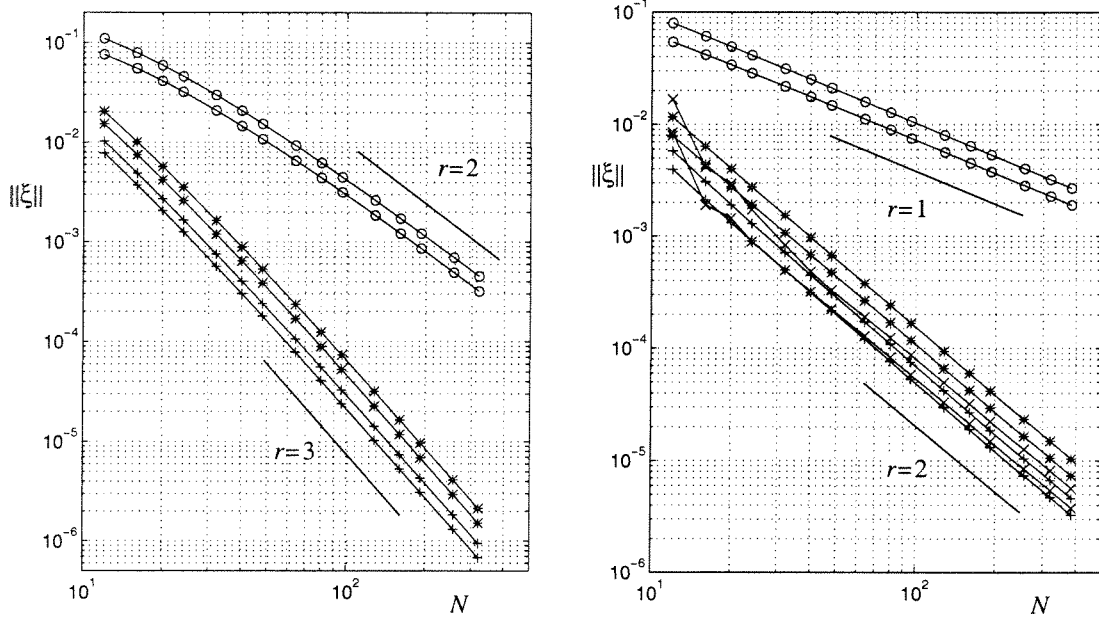


Figure 3.7: Error in the discrete solution. The left-side plot show the ∞ - and two-norms of ξ after one time step, for a wide range of N . For the backward Euler method (circles), $\xi = O(\Delta t^2)$, indicating a first-order accurate method. The Crank-Nicholson (stars) and RK (plus signs) discretizations are second-order accurate. On the right, we plot same error norms at $t = 0.25$, for these three methods and the hybrid discretization. The backward Euler method is first-order accurate, whereas the others are second-order accurate.

timestep, from $t = 0$. Figure 3.6 demonstrates that for $\Delta t = \Delta x$, and $\Lambda = 10^{-2}$, the two-norm of the truncation error in the interior of the domain is $O(\Delta t)$ for the backward Euler method, and $O(\Delta t^2)$ for the Crank-Nicholson and Runge-Kutta discretizations. There is no difference between the hybrid and Crank-Nicholson methods in the interior of the domain. Figure 3.6 also shows that, in the last cell, the partial-volume-weighted truncation error, $\Lambda \tau_N$, is bounded as $\Lambda \rightarrow 0$, and $O(\Delta t)$ for all methods.

This larger truncation error at the domain boundaries does not affect the convergence rate of the solution. Figure 3.7 shows the error in the discrete solution $\xi = \phi^e - \phi$, for a range of N . This is calculated for each of the methods by advancing the solution one time step, from $t = 0$ to $t = \Delta t$. As expected, ξ asymptotes to $O(\Delta t^2)$ for the backward Euler

discretization, in both the ∞ - and two-norms. The Crank-Nicholson and Runge-Kutta algorithms are one-order higher, $\|\xi\| = O(\Delta t^3)$. Note that the larger error from the endpoints does not corrupt the convergence rate of the solution in each case. The results from the hybrid method are not distinguishable from the Crank-Nicholson method for the first time step. Also in Figure 3.7, the ∞ - and 2-norms of the solution error are plotted at time $t = 0.25$, when the solution has a maximum value of $\varphi \approx 8.5 \cdot 10^{-2}$. The backward Euler method is first-order accurate for this case, while the Crank-Nicholson and Runge-Kutta discretization are second-order accurate. These results suggest that the larger truncation error at the boundary does not affect the accuracy of the solution. The hybrid method, with lagged conservation term $\delta\phi$, also appears to be second-order accurate, although convergence is more erratic for coarser grids. We have found that it is important to re-introduce $\delta\phi$ into a cell using the backward Euler discretization:

$$\phi_N^{n+1} - \Delta t L\phi_N^{n+1} = \phi_N^n + \delta\phi_N^n .$$

When $\delta\phi_N^n$ is put into a cell using the Crank-Nicholson discretization, it was found that the hybrid method is unstable in the range of parameters used here.

3.3.2 Decay of non-smooth initial conditions in 1D

We will now demonstrate how each method attenuates high-frequency components, by using an initial profile that is not C^0 , such as $\varphi(x, 0) = 1 - |2x - 1|$, for $x \in [0, 1]$. Again, we have a Fourier series representation of the solution at time t :

$$\varphi(x, t) = \frac{8}{\pi^2} \sum_{\text{odd } k > 0}^{\infty} \left(\frac{\pi}{2}k - \sin \frac{\pi}{2}k \right) \frac{e^{-\pi^2 k^2 \beta t}}{k^2} \sin \pi k x$$

Figure 3.8 shows the four different discrete solutions, with $N = 100$ and $\Lambda = 10^{-3}$, along with the initial conditions at $t = 0$, and the exact solution after one time step, $t = 0.01$. We

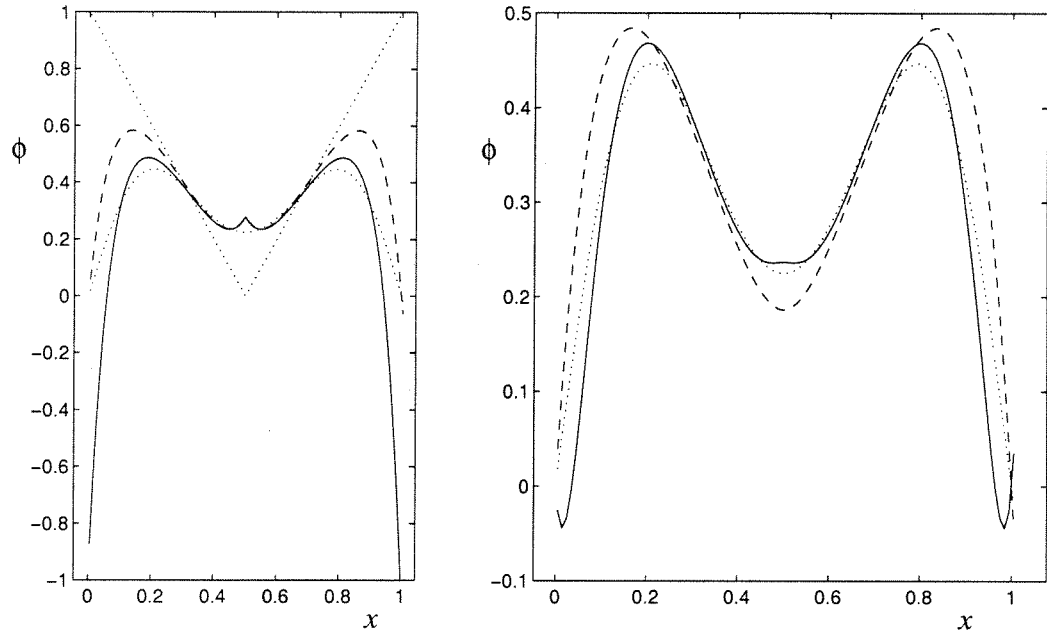


Figure 3.8: Discontinuous initial conditions. We plot the solution at $t = 0.01$, for the heat equation with $\varphi = 0$ at $x = 0, 1$, and initial conditions $\varphi(x, t = 0) = 1 - |2x - 1|$ (dotted V-shape). The discrete solutions for each of the four methods use $N = 100$, $\Delta t = 0.01$, and $\Lambda = 10^{-3}$. On the left, the exact solution (dotted curve) is smooth, whereas the Crank-Nicholson (solid) and hybrid (dashed) discretizations cause cusps at $x = \frac{1}{2}$. The CN discretization also suffers from $O(1)$ oscillations, due to the discontinuous initial conditions. On the right, the Runge-Kutta method causes small oscillations, while the backward Euler method (dashed curve) causes none at all.

see that the Crank-Nicholson and hybrid discretizations suffer from a lack of attenuation of the small-wavelength components in the solution; both algorithms have a cusp at $x = \frac{1}{2}$. The Crank-Nicholson discretization also has large oscillations at $x = 0$ and $x = 1$, whereas this is not as evident for the hybrid algorithm. In contrast to these two approaches, the \mathcal{L}_0 -stable backward-Euler method has no such difficulties. The Runge-Kutta discretization has small oscillations at $x = 0, 1$. At $t = 0.125$ the exact solution has a maximum value of $\varphi \approx 0.135$ and contains no high-frequency components. Figure 3.9 shows that the backward Euler method is first-order accurate in this case. This is true of the hybrid method as well, because the cusp at $x = \frac{1}{2}$ persists, even after $O(N)$ time steps. In addition to this effect, the Crank-Nicholson discretization contains $O(1)$ errors at $x = 0, 1$, and does not converge with grid refinement. These two errors, the cusp at $x = \frac{1}{2}$ and jumps at $x = 0, 1$

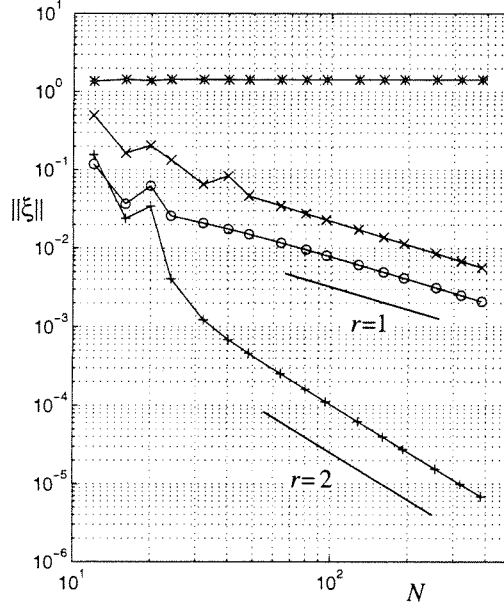


Figure 3.9: Error at $t = 0.125$ for discontinuous initial conditions. The error norm $\|\xi\|_1$ for the backward Euler method (circles) indicates first-order accuracy. The hybrid method (crosses) is also first-order accurate, due to the persistence of the cusp at $x = \frac{1}{2}$ shown in Figure 3.8. The Runge-Kutta method (plus signs) is second-order accurate, as expected, while the $O(1)$ oscillations at $x = 0, 1$ in the Crank-Nicholson solution (stars) persist, preventing convergence.

are oscillatory because they are associated with modes whose eigenvalues are $\lambda \approx -1$, and thus change their every time step, without decaying. In contrast, the Runge-Kutta method is second-order accurate, even for these discontinuous initial conditions.

3.4 Extension to Two Dimensions

The derivation in the previous section extends naturally to two or more dimensions, given an appropriate discretization of the domain. As in Chapter 2, we will use a cell-centered Cartesian grid, with the boundary represented as piecewise-linear, with one segment in each cell (i, j) . The same relationship (2.13) holds between the apertures A , normal \mathbf{n} , and area A^f of the front in each cell.

3.4.1 Finite volume derivation in two dimensions

We start by integrating (Eq. II) over the control volume defined by the irregular cell $V_{i,j}$, and from t^n to t^{n+1} , to obtain its surface integral representation:

$$\int_{V_{i,j}} \varphi(\mathbf{x}', t^{n+1}) d\mathbf{x}' - \int_{V_{i,j}} \varphi(\mathbf{x}', t^n) d\mathbf{x}' = \int_{t^n}^{t^{n+1}} \int_{\partial V_{i,j}} \beta \nabla \varphi \cdot \mathbf{n} dA_{i,j} dt'. \quad (3.32)$$

Time integrals of fluxes

In the one-dimensional algorithm, we were able to show that the algorithms were consistent due to specific cancellations in the quadrature errors for the flux integrals. We will briefly demonstrate this point in two dimensions, by using the truncation error estimates for $L\phi$ from Chapter 2. First, note that the right-hand side in (3.32) is the Laplacian of φ evaluated at time t , and integrated over cell (i, j) to obtain the flux difference form. The discretization we used before is still applicable, when evaluated at the proper time:

$$\int_{\partial V_{i,j}} \beta \nabla \varphi \cdot \mathbf{n} dA_{i,j} = V_{i,j} \left(L\phi_{i,j}^e + C_{i,j} \frac{\Delta x}{\Lambda_{i,j}} \right), \quad (3.33)$$

where we have again evaluated φ at cell centers $\mathbf{x}_{i,j}$ to obtain $\phi_{i,j}^e$. In addition, $C_{i,j}$ is uniformly bounded in Λ and Δx , as before, for sufficiently smooth φ . We have also set $\Delta y = \Delta x$ to simplify our derivation, although the results still apply for $\Delta y \propto \Delta x$, as in Chapter 2.

The approximations used for time integrals in one dimension are directly applicable here, because the constant $C_{i,j}$ is a smooth function of time. This is because the truncation error for $L\phi^e$ is only a function of the cell geometry and the spatial derivatives of φ . The cell geometry does not change with time, so we may freely take the time derivative of $C_{i,j}$ if φ is smooth enough. We can then provide error estimates for the forward Euler discretization

of the right-hand side of (3.32),

$$\int_{t^n}^{t^{n+1}} \int_{\partial V_{i,j}} \beta \nabla \varphi \cdot \mathbf{n} dA_{i,j} dt' = V_{i,j} \Delta t \left(L\phi_{i,j}^{e,n} + \frac{\Delta t \Delta x}{\Lambda_{i,j}} \partial_t C_{i,j}(t^n) \right), \quad (3.34)$$

where the error term is $V_{i,j} \Delta t \times O(\frac{\Delta x}{\Lambda_{i,j}})$. Of course, we obtain a similar result for the backward Euler discretization. For the Crank-Nicholson discretization, the trapezoidal rule for the time integral yields the same higher-order cancellation as was seen in one dimension,

$$\begin{aligned} \int_{t^n}^{t^{n+1}} \int_{\partial V_{i,j}} \beta \nabla \varphi \cdot \mathbf{n} dA_{i,j} dt' = \\ V_{i,j} \Delta t \left(\frac{1}{2} \left(L\phi_{i,j}^{e,n+1} + L\phi_{i,j}^{e,n} \right) + \frac{\Delta t^2 \Delta x}{\Lambda_{i,j}} \partial_{tt} C_{i,j}(t^{n+\frac{1}{2}}) \right). \end{aligned} \quad (3.35)$$

Taylor-series expansions of volume integrals

In (3.32), the integrals of φ over the cell volume can be approximated using the cell-center value; this is second-order accurate in full cells. In partial cells, we can obtain first-order accuracy, by expanding the integral about the cell center, $x_{i,j}$,

$$\int_{V_{i,j}} \varphi(\mathbf{x}', t^n) d\mathbf{x}' = \varphi(\mathbf{x}_{i,j}, t^n) \int_{V_{i,j}} d\mathbf{x}' + \nabla \varphi(\mathbf{x}_{i,j}, t^n) \cdot \int_{V_{i,j}} (\mathbf{x}' - \mathbf{x}_{i,j}) d\mathbf{x}' + O(\Delta x^4). \quad (3.36)$$

Note that the first integral on the right side is just the volume $V_{i,j}$. The second integral is just the volume times the vector from the cell centroid, $\mathbf{x}_{i,j}^c$, to the cell center. Thus, the expression reduces to

$$\int_{V_{i,j}} \varphi(\mathbf{x}', t^n) d\mathbf{x}' = V_{i,j} \varphi(\mathbf{x}_{i,j}, t^n) + V_{i,j} (\mathbf{x}_{i,j}^c - \mathbf{x}_{i,j}) \cdot \nabla \varphi(\mathbf{x}_{i,j}, t^n) + O(\Delta x^4). \quad (3.37)$$

Thus, when we difference the two integrals over the cell in (3.32), the difference of terms involving $\nabla \varphi$ is $O(\Delta t)$. Since $\mathbf{x}_{i,j}^c - \mathbf{x}_{i,j} = O(\Delta x)$, the error term becomes $V_{i,j} \times O(\Delta x \Delta t)$.

Truncation error for the methods

With these estimates in mind, we can now write the truncation error for the each time discretization, starting with the forward Euler method. If we use $\phi_{i,j}^n$ in place of $\varphi(\mathbf{x}_{i,j}, t^n)$, substitute (3.34) and (3.37) into (3.32), and finally divide by $V_{i,j}\Delta t$, we obtain

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} = L\phi_{i,j}^n . \quad (3.38)$$

Of course, for full cells, this formula reduces to the standard, forward Euler discretization in two dimensions. The truncation error for (3.38) is found by substituting $\phi_{i,j}^{e,n} = \varphi(\mathbf{x}_{i,j}, t^n)$ for $\phi_{i,j}^n$, and using the quadrature error estimates (3.34) and (3.37), which leads to

$$\frac{\phi_{i,j}^{e,n+1} - \phi_{i,j}^{e,n}}{\Delta t} = L\phi_{i,j}^{e,n} + O\left(\Delta t + \frac{\Delta x}{\Lambda_{i,j}}\right) , \quad (3.39)$$

in partial cells. Because the the truncation error for $L\phi$ is $O(\Delta x^2)$ for interior cells, the local truncation error is $O(\Delta t + \Delta x^2)$ there. The same results hold for the backward Euler method,

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} = L\phi_{i,j}^{n+1} . \quad (3.40)$$

Finally, using (3.35) in partial cells defines the Crank-Nicholson discretization,

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} = \frac{1}{2} \left(L\phi_{i,j}^{n+1} + L\phi_{i,j}^n \right) , \quad (3.41)$$

with truncation error $O(\Delta t^2 + \frac{\Delta x}{\Lambda_{i,j}})$. For interior cells, the method is second-order accurate in time and space, as expected. As in one dimension, the hybrid method is defined by using (3.40) in partial cells, and (3.41) in full cells.

Discrete conservation

The three methods, (3.38), (3.38), and (3.38), still satisfy a discrete conservation principle, as in one dimension. This is easily seen using results from Chapter 2; if we multiply (2.14) for $L\phi$ by the cell volume, $V_{i,j} = \Lambda_{i,j}\Delta x\Delta y$, and sum over any subset of cells $D \in \Omega$, the fluxes on interior edges cancel out in this sum. This leaves only the fluxes on the domain boundary, $G^f\phi$, and those between cells inside and outside D . With this result in mind, we can multiply (3.40) by $\Delta t V_{i,j}$, and sum over cells $(i,j) \in D$, and all time steps from $t = 0$ to t^n . This leads to the following discrete conservation equation:

$$\sum_{(i,j) \in D} V_{i,j}(\phi_{i,j}^n - \phi_{i,j}^0) = \Delta t \sum_{k=1}^n \left[\sum_{\text{edges} \in \partial D} \mathbf{n} \cdot G\phi^k \Big|_{\text{edge}} - \sum_{(i,j) \in D} G^f \phi_{i,j}^k \right]. \quad (3.42)$$

Similar equations for the forward Euler and Crank-Nicholson methods, which can be derived by starting with (3.38) or (3.41), and evaluating the same sums.

3.5 Multi-Grid Iteration Strategy

The non-adaptive multi-grid approach from Chapter 2 is easily extended to include systems of equations like (3.40). These implicit discretizations require the solution to a linear system of the form:

$$\phi_{i,j} - \zeta L\phi_{i,j} = \rho_{i,j}. \quad (3.43)$$

The only modification needed to apply the multi-grid strategy to (3.43), is to replace the point-relaxation procedure (2.22) with:

$$\phi_{i,j}^m = \phi_{i,j}^{m-1} + \frac{1}{\mu} \left(\rho_{i,j} - \phi_{i,j}^{m-1} + \zeta L\phi_{i,j}^{m-1} \right), \quad (3.44)$$

where the parameter μ is now the diagonal entry of the matrix operator $(I - \zeta L)$. Note that the superscript m refers to the point-relaxation iteration number, and is not related to the time centering, n . No other modifications to the multi-grid procedure are necessary.

3.6 Software Implementation

As in Chapter 2, our algorithm for (Eq. II) is implemented using a combination of the C++ and FORTRAN programming languages. The major difference is that the current implementation is non-adaptive, and is no longer based on the modules developed by Martin and Cartwright [42]. This allows us to use the simpler, single-block multi-grid algorithm, without needing the multiple-block data structures in `BoxLib`. This has been implemented in a `MultiGrid` class, which uses the algorithm presented in Section 2.4.

We have also used the object-oriented features in C++, such as derivation and polymorphism. First, note that in the two different multi-grid algorithms for (Eq. I) and (Eq. II), only the linear operator and point-relaxation schemes changed. For the general case where this is true, we have developed a common base class, `Operator`, which implements standard finite-difference operators on rectangular domains, using *virtual* member functions. The `MultiGrid` class then contains an array of *pointers* to `Operator` objects, one for each grid in the hierarchy. Thus, the basic `MultiGrid` class can be used to invert linear systems corresponding to standard discretizations on rectangular domains.

The benefit of this approach is that embedded boundary classes can be derived from `Operator`. Such derived classes have direct access to `Operator`'s member functions, can use them for applying finite-difference operators on single-block data, and avoid multiple implementations of these functions. For problems like (3.43), we have created a derived class, `EBOper`, which implements point-relaxation and other operators for the embedded boundary discretization. Information about the domain geometry and finite-difference stencil is contained in the member classes `VoF` and `EmbBndry`, respectively, which were described

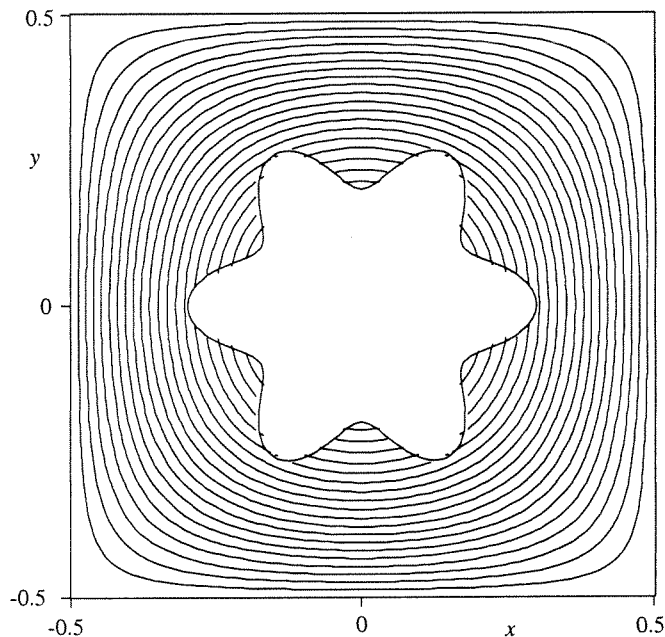


Figure 3.10: Discrete solution for the test problem in two-dimensions, at $t = 0.125$, for $\Delta x = \frac{1}{80}$, with twenty contours between $\varphi = 0$ and $\varphi \approx 0.07$.

in Chapter 2. Another benefit of the derivation mechanism in C++, is that virtual member functions in `Operator` can be redefined by `EB0per`. This results in polymorphism: a dereferenced pointer to an `EB0per` is treated as an `Operator`, but calls to virtual functions in `Operator` use the redefined versions in `EB0per`. This allows us to specify a fixed programming interface between `Operator` and `MultiGrid`, thereby isolating the `MultiGrid` class from changes in the finite-difference discretization.

3.7 Results in Two Dimensions

In two dimensions, we will again use an exact solution to verify the error estimates given above. Consider the domain $\Omega = \Upsilon_1 \cap \Upsilon_2$ used in Problem 3 of the last chapter

(Figure 3.10), where

$$\Upsilon_1 = \{(r, \theta) : r \geq 0.25 + 0.05 \cos 6\theta\}$$

and Υ_2 is the unit square, centered at the origin, $\Upsilon_2 = \{(x, y) : |x| \leq \frac{1}{2}, |y| \leq \frac{1}{2}\}$. We will use the following exact solution of (Eq. II) with $\beta = 1$,

$$\varphi(x, y, t) = e^{-2\pi^2 t} \cos \pi x \cos \pi y, \quad (3.45)$$

which satisfies the boundary conditions $\varphi = 0$ on $\partial\Upsilon_2$. On the interface $\partial\Upsilon_1$, we enforce inhomogeneous Dirichlet boundary conditions given by (3.45), evaluated at the midpoint of the front in each cell. The discrete solution is initialized using (3.45), evaluated at $t = 0$, at cell-centers on grid with spacing Δx .

As in one dimension, we set $\Delta t = \Delta x$, and use the exact solution at $t = \Delta t$ to evaluate the accuracy of each of the methods. Our first result is the $\|\cdot\|_1$ norm of the volume-fraction-weighted local truncation error, $\Delta t \Lambda \tau$, plotted in Figure 3.11. We see that the backward Euler method is first-order accurate, with local truncation error $O(\Delta t^2)$, while the hybrid, Crank-Nicholson, Runge-Kutta discretizations are second-order accurate. Also in Figure 3.10, we plot the partial- and full-cell components of the local truncation error for the Crank-Nicholson discretization. We see that the full cells have $\Delta t \Lambda \tau_I = O(\Delta t^3)$, as expected for a second-order accurate method, while the partial cell values are one order lower, or $\Delta t \Lambda \tau_P = O(\Delta t^2)$. However, in Figure 3.12, we see that the error after one time-step also behaves as in one dimension, with $\xi = O(\Delta t^2)$ for the backward Euler method, and $\xi = O(\Delta t^3)$ for the other discretizations, although the Runge-Kutta method seems to be approaching this limit only at the finest resolution. Also in Figure 3.12, we have plotted the two components of the error for the Crank-Nicholson discretization, ξ_P due to partial cells, and ξ_I from full cells, both of which are $O(\Delta t^3)$. We conclude that the larger errors near boundaries do not deteriorate the overall accuracy of each approach after one time

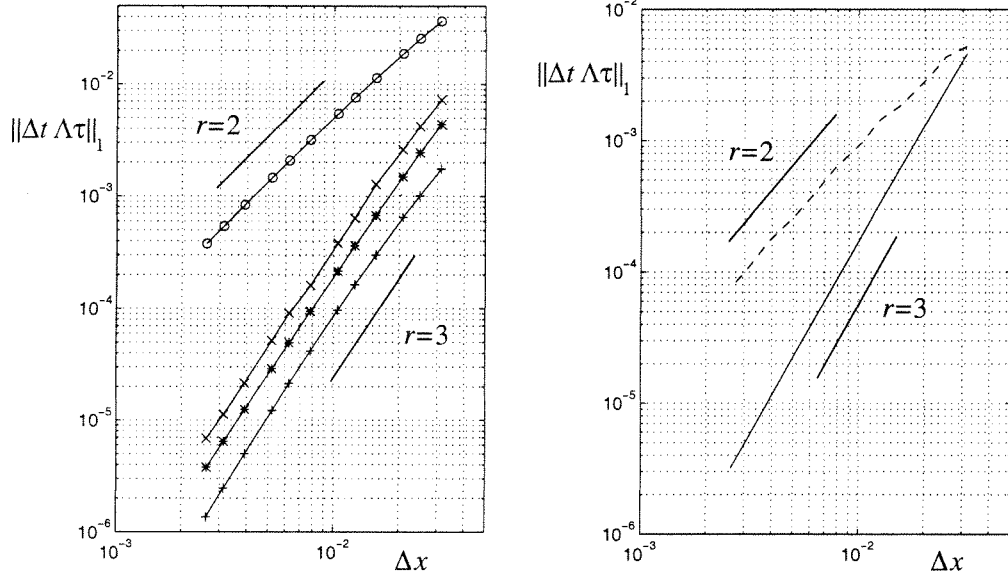


Figure 3.11: Volume-weighted local truncation error for the test problem in two dimensions. The backward Euler method (circles) has $\|\Delta t \Lambda \tau\|_1 = O(\Delta t^2)$, as expected for a first-order method, while the hybrid (crosses), Crank-Nicholson (stars), and Runge-Kutta discretizations (plus signs) are second-order accurate. On the right, we plot the two components of the larger truncation error, $\Delta t \Lambda \tau_P = O(\Delta t^2)$, due to partial cells (dashed), and τ_I , due to the discretization in full cells, for the Crank-Nicholson method.

step.

Figure 3.10 shows the domain and discrete solution at $t = 0.125$, for grid spacing $\Delta x = \frac{1}{80}$ and $\Delta t = \Delta x$. The exact solution (3.45) at this time has maximum value of $\varphi \approx 7 \cdot 10^{-2}$ on Υ . The solution errors for each of the methods are given in Figure 3.13. The left plot shows that $\|\xi\|_\infty$ behaves as expected for the backward Euler and Crank-Nicholson discretizations. The hybrid method with lagged conservation also seems to be roughly second-order accurate, with variations due to erratic convergence of partial cells in the ∞ -norm. The Runge-Kutta method seems to be slightly less than second-order accurate. The right-side graph in Figure 3.13 shows that $\|\xi\|_2$ behaves more as expected.

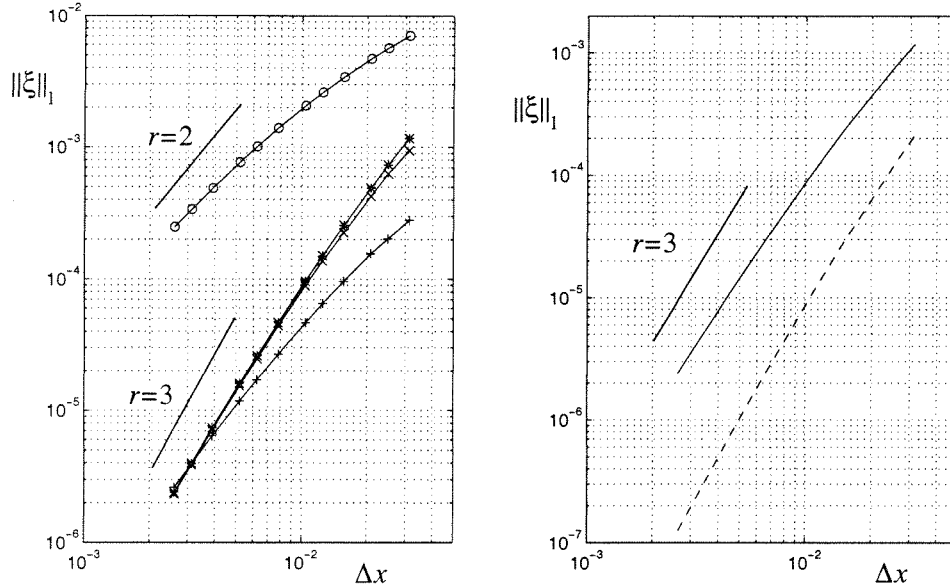


Figure 3.12: Error for two-dimension test problem. The error in the discrete solution after one time step is plotted versus the grid spacing. For the backward Euler method (circles), $\|\xi\|_1 = O(\Delta x^2)$, indicating first-order accuracy, whereas the other methods appear to be second-order accurate. On the right, we have plotted the two components of the error after one timestep, for the Crank-Nicholson discretization. We see that the error induced by the partial cells, ξ_P (dashed line), converges at the same rate as that from the interior of the domain, ξ_I .

3.8 Conclusions

In this chapter we have extended the finite-volume discretization of (Eq. I) presented in Chapter 1 to the heat equation with variable coefficients (Eq. II). We derive a consistent method from a discrete control-volume analysis, which used quadrature rules for volume and surface integrals, combined with finite-difference approximations for the integrands. Our first decision is to discard the forward Euler method, because the maximum stable time step is severely limited in the presence of small cells. This leads us to consider three implicit, conservative time discretizations: the backward Euler and Crank-Nicholson methods, and a second-order accurate Runge-Kutta method. In addition, we consider a non-conservative hybrid method, which uses a backward Euler discretization in cells containing a portion of the domain boundary, and the Crank-Nicholson method elsewhere.

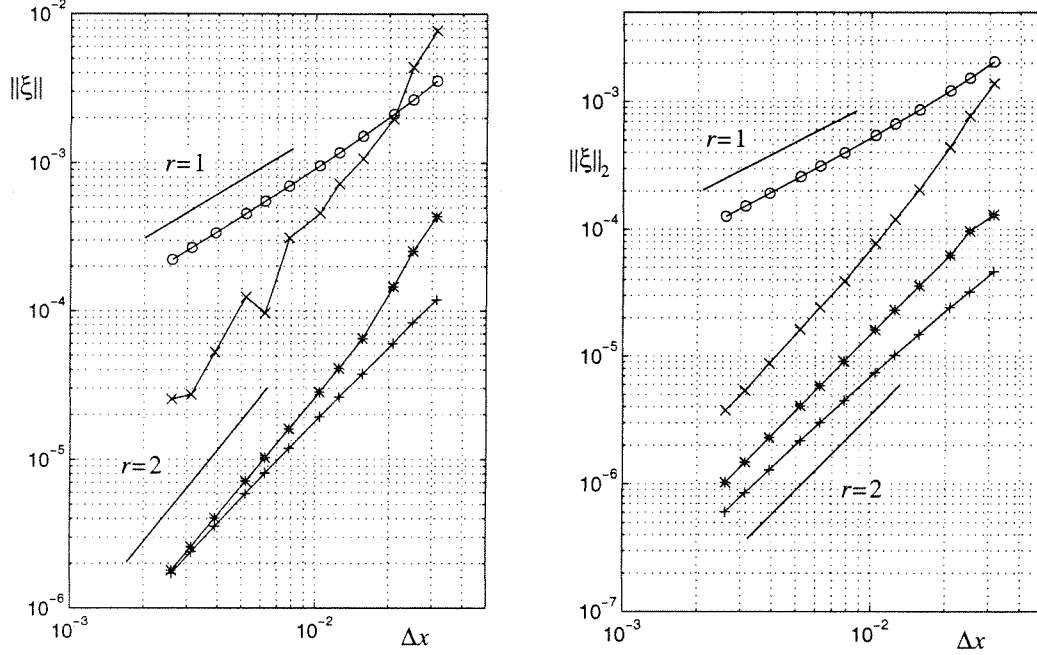


Figure 3.13: Error at $t = 0.125$. In the left-hand plot, we see that the ∞ -norm of the discrete error is $O(\Delta x)$ for the backward Euler method (circles), $O(\Delta x^2)$ for the Crank-Nicholson (stars) method, and slightly less than $O(\Delta x^2)$ for the Runge-Kutta (plus signs) discretization. The hybrid method also seems to be second-order accurate, although convergence is erratic for $\|\xi\|_\infty$. On the right, $\|\xi\|_2$ behaves more smoothly, as expected.

Discrete conservation is maintained by introducing a lagged source term, to account for the mismatch between the fluxes of the two different methods.

In all cases, the truncation error in partial cells is found to be $O(\Delta x \Lambda^{-1})$, for $\Delta t \propto \Delta x$. We are able to show that this truncation error at the boundary produces an $O(\Delta x^3)$ error in the solution after one timestep. Numerical results in one and two dimensions demonstrate this fact, and confirm that the larger truncation error in partial cells does not deteriorate the overall accuracy of each of the methods. Thus, for smooth initial, the backward Euler method is globally first-order accurate, whereas the other methods are globally second-order accurate. The hybrid method with lagged conservation is second-order accurate, when the source term is incorporated into a cell using the backward Euler method. However, if that cell uses a Crank-Nicholson discretization, the algorithm is found

to be unstable in the given parameter ranges.

In one dimension, analysing the eigensystem of $L\phi$ also allows us to determine the damping properties of each of the conservative methods. We demonstrate that the maximum eigenvalue of L grows like $O(N^2\Lambda^{-1})$, which is unbounded for arbitrarily small cell volumes. For the \mathcal{L}_0 -stable backward Euler and Runge-Kutta discretizations, the corresponding eigenmodes are heavily damped in time. For the Crank-Nicholson discretization, these modes change sign every time-step, and were practically undamped. The hybrid method is shown to be effective at damping eigenmodes associated with the partial cells, but not other cells. Finally, we introduce a numerical test, which uses an initial condition that was discontinuous at each end of the interval, and whose gradient was discontinuous at the interval's midpoint. The \mathcal{L}_0 backward-Euler and Runge-Kutta methods are still first- and second-order accurate, respectively, whereas the Crank-Nicholson method does not converge due to the undamped $O(1)$ oscillations at each endpoint. The hybrid method was only first-order accurate, due to an $O(\Delta x)$ oscillation at the interval midpoint. This is not surprising, because the two methods differ only at the domain boundaries.

We also demonstrate that each discretization is easily extended to two dimensions, without modifying the approach used for (Eq. I). Each approach relies on a cancellation of quadrature errors in the integral of φ over irregular cells; these terms can then be approximated with cell-centered values, instead of using the $O(\Delta x^2)$ midpoint rule, without loss of accuracy. This allows to make minimal changes to the multi-grid algorithm, to treat each of the four linear systems. For smooth initial conditions, numerical experiments show that the truncation error is again $O(\Delta x \Lambda^{-1})$ in partial cells, and induces only an $O(\Delta t^3)$ error in the solution each timestep. In each case, the solution has a time-accuracy corresponding to the method used in the interior of the domain.

Chapter 4

Stefan Problem

4.1 Introduction

In this chapter we extend our discretization of the heat equation (Eq. II) on fixed domains, to include changing domains $\Omega(t)$. As described in Chapter 1, the classical Stefan problem represents the simplest set of governing equations for solidification phenomena, and specifies Dirichlet boundary conditions, $\varphi(\mathbf{x}, t) = 0$, on the domain boundary, $\partial\Omega(t)$. In addition, the boundary moves normal to itself, with speed

$$u(\mathbf{x}, t) = -St \, \mathbf{n} \cdot \beta \nabla \varphi(\mathbf{x}, t). \quad (\text{Eq. III})$$

This represents conservation of energy, in that φ diffusing through the boundary causes the front to traverse a specified volume. Therefore, the Stefan number St can be interpreted as a parameter representing the balance between the front speed and the rate of diffusion. For most semiconductor applications, $St \lesssim 1$, and diffusion time scales are shorter than those of the front motion. For liquid metals, typical applications have $St \gg 1$, so that conduction dominates. Our discretized equations will again be a finite volume method, based on integrating (Eq. II) over the part of each cell inside Ω , for one time step. However,

this now requires that we represent $\partial\Omega$ in space and time, and compute surface integrals on the moving front. Because (Eq. III) requires us to keep track of the volume moving through $\partial\Omega$, we employ an explicit volume-of-fluid front-tracking algorithm to provide a discretization of the moving control volume, and advance the boundary in time. We can then define an implicit finite-volume discretization using quadrature formulas for the surface and volume integrals, along with finite difference approximations of $\nabla\varphi$. The constraint represented by (Eq. III) is enforced using a lagged conservation term, as with the hybrid method in Chapter 3. Although this avoids coupling (Eq. III) with finding φ^{n+1} in the heat equation, it introduces stability problems for the both discretizations, so that we are unable to develop a conservative discretization for the Stefan problem. The non-conservative discretizations are stable, however, and results are presented for both discretizations in one and two space dimensions.

We present a simplified derivation in one space dimension next, in Section 2. This consists of both a discretization for the heat equation with moving boundaries, and a means of enforcing (Eq. III). In Section 3 we evaluate numerical results from the one-dimensional algorithm, in order to guide the derivation in higher dimensions. Section 4 describes a volume-of-fluid front-tracking algorithm, which is needed to extend the algorithm to two dimensions. The finite-volume discretization is derived for that case in Section 5. In the last section, we evaluate the algorithm's numerical results for test problems in two dimensions.

4.2 Approach in 1D

We first consider the heat equation in one dimension,

$$\varphi_t = \beta \varphi_{xx} \text{ for } x \in [0, s(t)], \text{ with } \varphi(s(t)) = 0, \quad (4.1)$$

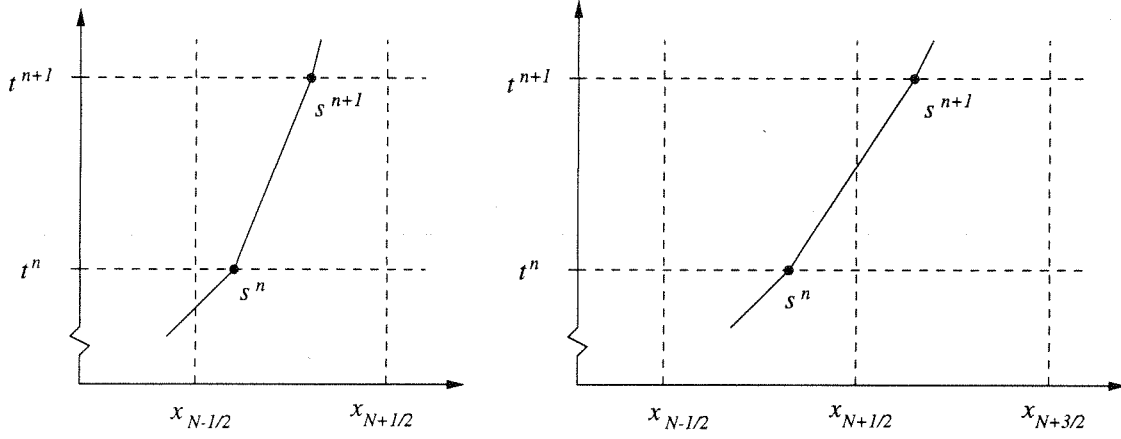


Figure 4.1: The front advances by at most Δx in on time step Δt . It either spends the entire time step in cell N (left), or it crosses between cells N and $N + 1$.

with a moving boundary, $s(t)$, and constant $\beta > 0$. In this case, (Eq. III) for the speed of the boundary reduces to

$$u(t) = -\text{St} \beta \varphi_x(s(t), t), \quad (4.2)$$

so that the only free parameter in the problem is the Stefan number, St . In addition, we will assume that the front only advances in time, so that $u = \dot{s} > 0$.

4.2.1 Domain Discretization and Nomenclature

The control volume formulation first requires that a representation of the front's trajectory, $s(t)$. To this end, we first assume that the front's speed u is constant from time t^n to time t^{n+1} . This implies that

$$\Delta s \equiv s^{n+1} - s^n = \Delta t u^n,$$

where s^n and s^{n+1} are the front's locations at each time. This is represented in the space-time diagram in Figure 4.1. Note that N , the right-most cell inside the domain, and Λ are

now functions of the front position, s , and therefore functions of time. We use the notation that cell N is the cell containing the front at t^n ,

$$N = 1 + \left\lfloor \frac{s^n}{\Delta x} \right\rfloor ,$$

where $\lfloor \cdot \rfloor$ represents rounding down to the next whole number. In discussing the algorithm between t^n and t^{n+1} , we drop the superscripts from u , N , and Δs to simplify notation. The volume fraction Λ^n at time t^n is given by

$$\Lambda^n = \frac{s^n}{\Delta x} - \left\lfloor \frac{s^n}{\Delta x} \right\rfloor ,$$

with an analogous formula for Λ^{n+1} given by substituting s^{n+1} in place of s^n .

Note that if $s^{n+1} \leq x_{N+\frac{1}{2}}$, then the front stays in cell N during the time step; otherwise it crosses from cell N to $N+1$. We make the restriction that $\Delta s \leq \Delta x$; the front moves no more than one cell width during a time step. Let us introduce the notation that δs_i is the distance the front covered in cell i ,

$$\delta s_N = \min \left(x_{N+\frac{1}{2}} - s^n, \Delta s \right) \quad \delta s_{N+1} = \max \left(s^{n+1} - x_{N+\frac{1}{2}}, 0 \right) .$$

Similarly, we split up Δt into the amount of time the front spends in each cell, so that $\delta t_N = \delta s_N / u$ and $\delta t_{N+1} = \delta s_{N+1} / u$. With these definitions we have

$$\delta s_N + \delta s_{N+1} = \Delta s \quad \text{and} \quad \delta t_N + \delta t_{N+1} = \Delta t .$$

4.2.2 Finite Volume Derivation with Moving Boundary

First, we will focus on the discretization in cell N , in order to demonstrate the general finite volume approach. Note that in cells $i = 1, \dots, N-1$ the algorithm is the same as was used in the previous chapter for the heat equation, because they do not interact

directly with the front. Again referring to the diagram in Figure 4.1, we can formulate a finite-volume algorithm by applying the divergence theorem to the space-time control volume in cell N ,

$$\begin{aligned} \int_{t^n}^{t^{n+1}} \int_{x_{N-\frac{1}{2}}}^{s(t')} (\varphi_t - \beta \varphi_{xx}) dx' dt' &= \int_{t^n}^{t^{n+1}} \beta \varphi_x(x_{N-\frac{1}{2}}, t') dt' \\ &+ \int_{x_{N-\frac{1}{2}}}^{s^{n+1}} \varphi(x', t^{n+1}) dx' - \int_{x_{N-\frac{1}{2}}}^{s^n} \varphi(x', t^n) dx' \\ &- \int_{t^n}^{t^{n+1}} u \varphi(s(t'), t') dt' - \int_{t^n}^{t^{n+1}} \beta \varphi_x(s(t'), t') dt' \end{aligned} \quad (4.3)$$

Note that this applies to the case where the front does not leave cell N during the time step (Figure 4.2), that is $x_{N-\frac{1}{2}} \leq s^n, s^{n+1} \leq x_{N+\frac{1}{2}}$; the other case will be dealt with in more detail later. In (4.3), the right-hand side of the first line represents the integral of the fluxes through the left edge of cell N , like the same term in the heat equation. The second line of (4.3) represents the change in the total integral of φ in cell N between times t^{n+1} and t^n . The last line represents the effect of the boundary motion on the total integral of φ . The first integral is the change in φ due to the front overtaking regions where $\varphi(s(t), t)$, whereas the second integral is the total flux through the moving interface over the time step.

If we integrate the constraint (4.2) from t^n to t^{n+1} , we obtain

$$\Delta s = -\text{St} \int_{t^n}^{t^{n+1}} \beta \varphi_x(s(t'), t') dt' . \quad (4.4)$$

Note that this is proportional to the last term in (4.3). Thus, the finite volume interpretation of the Stefan problem requires that the time-integral of the flux through the front is exactly proportional to the total distance covered by the front. This represents a conservation of φ , in that the volume converted by the front is driven by the diffusion of φ (given by $\beta \varphi_x$) from the interior. For the remainder of this section, it will be assumed that $\beta = 1$, to simplify the derivation and discussion.

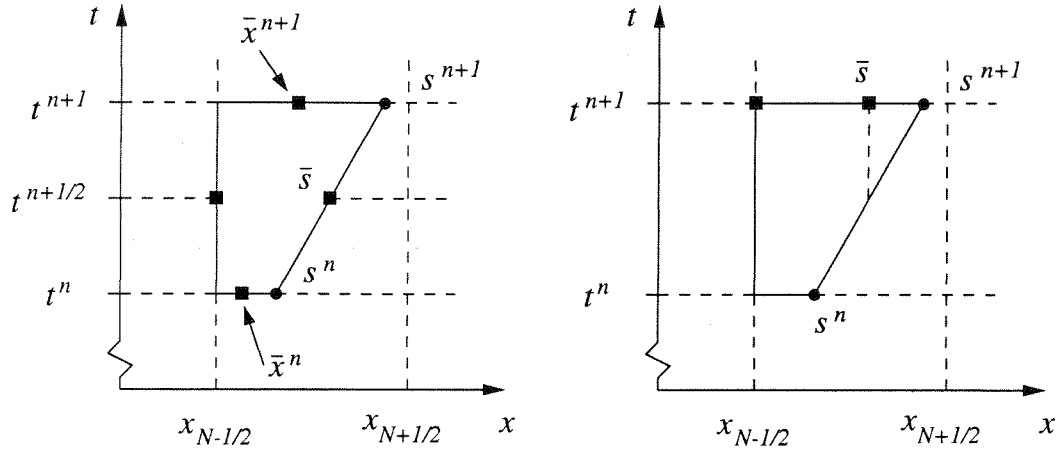


Figure 4.2: We plot the space-time surfaces corresponding to the integrals in (4.3). If we approximate the integrals using the midpoint rule (square boxes, left), we obtain a first-order accurate method. Alternatively, the time integrals may be evaluated at the same location in space, but at t^{n+1} (right), without losing any accuracy.

Cancellation of Quadrature Errors

We now wish to show a more general characteristic of finite volume methods of the form (4.3). In Chapter 3, we obtained a cancellation of quadrature errors in the finite-volume formulation for (Eq. II) with fixed boundaries. This allowed us to use only first-order accurate quadrature rules, and still retain a consistent discretization. We will now demonstrate that this behavior extends to the case with moving boundaries.

Let us expand the integrals of φ in (4.3) in a Taylor series about some x^* . After some simplification, we get

$$\begin{aligned}
 \int_{t^n}^{t^{n+1}} \int_{x_{N-1/2}}^{s(t')} \varphi_t dx' dt' &= \Lambda^{n+1} \Delta x \varphi(x^*, t^{n+1}) - \Lambda^n \Delta x \varphi(x^*, t^n) - \Delta s \varphi(x^*, t^{n+\frac{1}{2}}) \\
 &\quad + \Lambda^{n+1} \Delta x (\bar{x}^{n+1} - x^*) \varphi_x(x^*, t^{n+1}) - \Lambda^n \Delta x (\bar{x}^n - x^*) \varphi_x(x^*, t^n) \\
 &\quad - \Delta s (\bar{s} - x^*) \varphi_x(x^*, t^{n+\frac{1}{2}}) + O(\Delta x^3).
 \end{aligned}$$

Here we have treated u as a constant between t^n and t^{n+1} , and represented the irregular

cell N centers as \bar{x} , and the front location at $t^{n+\frac{1}{2}}$ by $\bar{s} = \frac{1}{2}(s^n + s^{n+1})$. We can also expand the last two lines of terms in $\varphi_x(x^*, t)$, about some time t^* , at which point they reduce to

$$\begin{aligned} & - (\Lambda^{n+1} \Delta x - \Lambda^n \Delta x - \Delta s) x^* \varphi_x(x^*, t^*) \\ & + (\Lambda^{n+1} \Delta x \bar{x}^{n+1} - \Lambda^n \Delta x \bar{x}^n - \Delta s \bar{s}) \varphi_x(x^*, t^*) + O(\Delta x^2 \Delta t) . \end{aligned}$$

The first term in parentheses is zero because $\Delta s = (\Lambda^{n+1} - \Lambda^n) \Delta x$. Similarly, by using the definitions of \bar{x} , which in this case are

$$\bar{x}^{n+1} = \frac{1}{2} (x_{N-\frac{1}{2}} + s^{n+1}) \quad \text{and} \quad \bar{x}^n = \frac{1}{2} (x_{N-\frac{1}{2}} + s^n) ,$$

we see that the second term in parentheses is zero, also. With these cancellations, the remaining error term is $O(\Delta x^2 \Delta t)$, implying the algorithm is first order accurate. Thus, by evaluating the integrals of φ at x^* and the proper time given by the midpoint rule, we still obtain a consistent method.

We can show a similar result for the integrals in (4.3) involving φ_x , but with the roles of x and t reversed. Here φ_x is evaluated at some arbitrary t^* , but at the location in space given by the midpoint rule (Figure 4.2),

$$\int_{t^n}^{t^{n+1}} \int_{x_{N-\frac{1}{2}}}^{s(t')} \varphi_{xx} dx' dt' = \Delta t \left(\varphi_x(\bar{s}, t^*) - \varphi_x(x_{N-\frac{1}{2}}, t^*) \right) + O(\Delta x \Delta t^2) \quad (4.5)$$

Thus, we have a first-order accurate discretization of φ_{xx} in (4.1), provided the finite difference stencils for φ_x are second-order accurate.

4.2.3 Derivation of the Finite Volume Method

We may now return to the problem at hand, given a bit more confidence about our ability to discretize (4.1). We will use a method similar to what those presented in

Chapter 3 for the heat equation. This means either a backward Euler or hybrid method in any cell the front intersects during the time step. Because the algorithm for full cells was described in the previous chapter, we will not treat it with any more detail here. However we will provide a detailed description of the time-stepping procedure for cells that intersect the front.

Approximation of φ integrals

At this point, we will make use of the boundary condition $\varphi(s(t), t) = 0$, in (4.1). This is substituted directly into the integral of $u\varphi$ in (4.3), making that term zero. The spatial integrals that are left are then approximated using the midpoint rule. For example, at t^n , this means

$$\int_{x_{N-\frac{1}{2}}}^{s^n} \varphi(x', t^n) dx' \approx \Delta x \varphi(\bar{x}^n, t^n) + C_1 \Delta x^3 \varphi_{xx}(\bar{x}^n, t^n) + O(\Delta x^5). \quad (4.6)$$

Note that \bar{x}^n depends on the front location s^n . For instance, if s^{n+1} is not in cell N , then we use $\bar{x}^{n+1} = x_N$, which is the Cartesian cell center. Now we must determine an approximation to $\varphi(\bar{x}^n, t^n)$.

However, it was shown above that we could have obtained a consistent discretization even if the terms were all evaluated at any point x^* , like x_i . This was the approach taken in Chapter 3, for fixed domains. In this case, we would not be able to use the fact that $\varphi(s(t), t) = 0$, to remove the $u\varphi$ term in (4.1). We would have to approximate $\varphi(x^*, t)$ using an interpolation in time, in conjunction with an extrapolation in space, if $x^* > s$. This could have consequences for the stability of the algorithm, and so we prefer to approximate the spatial integrals of φ with values at \bar{x} (Figure 4.2).

For cell N , we can linearly interpolate neighboring cell-centered values of ϕ to \bar{x} ,

by defining an averaging operator,

$$\text{avg}(\bar{x}_N^n, \phi^n) = \frac{(\bar{x}_N^n - x_{N-1})}{\Delta x} \phi_N^n + \frac{(\bar{x}_N^n - x_N)}{\Delta x} \phi_{N-1}^n. \quad (4.7)$$

Using linear interpolation provides second-order accuracy to the value of $\varphi(\bar{x}^n, t^n)$. By substituting this into (4.6) we obtain

$$\int_{x_{N-\frac{1}{2}}}^{s^n} \varphi(x', t^n) dx' \approx \Lambda^n \Delta x \text{avg}(\bar{x}_N^n, \phi^n) + O(\Delta x^3).$$

Of course, an analogous approach is used to approximate the spatial integrals at t^{n+1} using the averaging operator at the new time, $\text{avg}(\bar{x}^{n+1}, \phi^{n+1})$.

Note that if the front has left cell N at t^{n+1} , x_N is the cell centroid at that time, and no interpolation is necessary. In that case, the front has entered cell $N+1$ during the time step, and we use the interpolation stencil at t^{n+1} there. We assume the integral of φ^n is zero in that cell, and no interpolation is necessary.

Integrals of spatial fluxes

At this point we must separate the problem into two distinct cases. The first is when the front stays in cell N for the duration of the time step, that is $x_{N-\frac{1}{2}} \leq s^n, s^{n+1} \leq x_{N+\frac{1}{2}}$. The second case is when the front leaves cell N and enters into cell $N+1$ during the time step, or $s^n \leq x_{N+\frac{1}{2}} \leq s^{n+1}$. We will now treat each case separately.

Front remains in cell N . If the front stays in cell N , we use the quadrature error cancellation presented above to formulate a backward Euler discretization. Instead of evaluating φ_x at the time indicated by the midpoint rule, we will use $t^* = t^{n+1}$, and use the formula (4.5)

$$\int_{t^n}^{t^{n+1}} \varphi_x(s(t'), t') dt' - \int_{t^n}^{t^{n+1}} \varphi_x(x_{N-\frac{1}{2}}, t') dt' \approx \Delta t \left(F_{\bar{s}_N}^{f, n+1} - F_{N-\frac{1}{2}}^{n+1} \right). \quad (4.8)$$

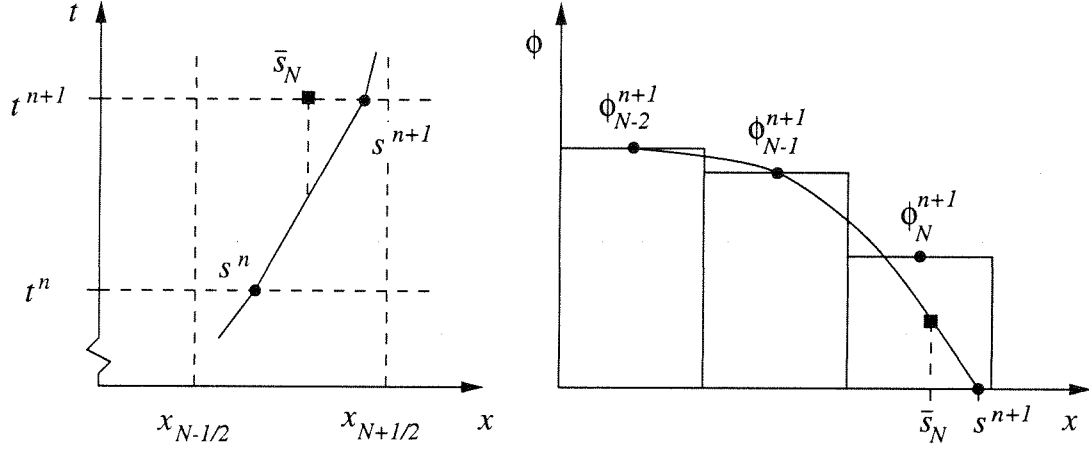


Figure 4.3: When the front stays in cell N during the time step, we require an approximation to the gradient at \bar{s}_N . This is accomplished by fitting a quadratic polynomial to the two values ϕ_{N-2} and ϕ_{N-1} , and the value 0 at s^{n+1} . The slope of the polynomial is then evaluated at \bar{s}_N to approximate $\partial_x \phi$.

The flux $F_{N-\frac{1}{2}}^{n+1}$ represents centered differences of ϕ^{n+1} as in Chapter 3. We have also introduced a new flux approximation, $F_{\bar{s}}^{f,n+1}$, which uses values ϕ^{n+1} , but is calculated at \bar{s}_N instead of at the actual front location, s^{n+1} . We can extend the formula for $F^{f,n+1}$, from the previous chapter, by evaluating the quadratic interpolant at this new location to obtain

$$F_{\bar{s}_N}^{f,n+1} = \frac{1}{d_2 - d_1} \left(\phi_{N-2}^{n+1} \frac{d_1}{d_2} - \phi_{N-1}^{n+1} \frac{d_2}{d_1} \right) + \frac{2(\bar{s}_N - s^{n+1})}{d_2 - d_1} \left(\phi_{N-2}^{n+1} \frac{1}{d_2} - \phi_{N-1}^{n+1} \frac{1}{d_1} \right), \quad (4.9)$$

where d_1 and d_2 are the distances from s^{n+1} to cell centers x_{i-1} and x_{i-2} respectively. We have also used the Dirichlet boundary condition $\Phi^f(t^{n+1}) = 0$ to simplify the formula. A graphical depiction of it is given in Figure 4.3, for case we are studying now. Note that when the front is not moving, $\bar{s}_N = s^{n+1}$ and the second term vanishes, and the formula reduces to $F^{f,n+1}$.

Front crosses cell edge. When the front crosses from cell N to cell $N+1$ during the time step, we break up Δs and Δt into parts corresponding to each cell, using the notation introduced earlier. See Figure 4.4 for a diagram corresponding to this case. For cell $N+1$,

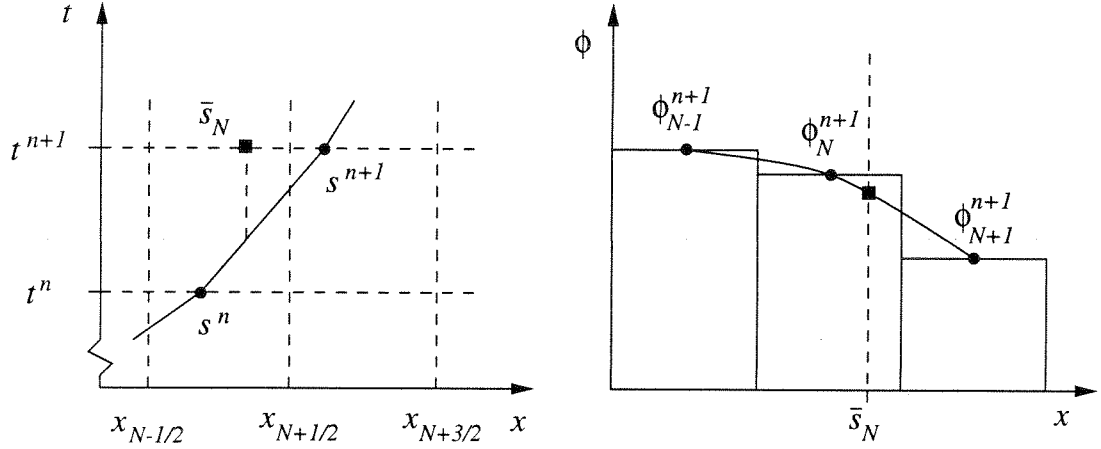


Figure 4.4: When the front crosses between cells N and $N + 1$, the difference formula requires an approximation to the gradient at \bar{s}_N . This is accomplished by fitting a quadratic polynomial to the three values ϕ_{N-1} , ϕ_N , and ϕ_{N+1} , and evaluating its slope at \bar{s}_N .

the time integrals are approximated by

$$\int_{t_c}^{t^{n+1}} \varphi_x(s(t'), t') - \varphi_x(x_{N+\frac{1}{2}}, t') dt' \approx \delta t_{N+1} \left(F_{\bar{s}_{N+1}}^{f, n+1} - F_{N+\frac{1}{2}}^{n+1} \right). \quad (4.10)$$

Here $\bar{s}_{N+1} = \bar{x}_{N+1}$, because the entire irregular cell $N + 1$ was traversed by the front.

The interface flux for cell N is computed differently, when the front leaves during the time step. In order to avoid tracking the front's properties outside of cell N , we have chosen to interpolate a flux to the point \bar{s}_N , and call the result $F_{\bar{s}_N}^{I, n+1}$. This is done by differencing cells $N - 1$, N , and $N + 1$ to obtain an approximation of the gradient at \bar{s}_N and t^{n+1} ,

$$F_{\bar{s}_N}^{I, N+1} = \frac{1}{\Delta x} \left(\left(\gamma - \frac{1}{2} \right) \phi_{N-1}^{n+1} - 2\gamma \phi_N^{n+1} + \left(\gamma + \frac{1}{2} \right) \phi_{N+1}^{n+1} \right), \quad (4.11)$$

where $\gamma = (\bar{s}_N - x_N) / \Delta x$. Note that this is a second-order accurate difference stencil.

The other time integrals in this case are computed using centered differences at

t^{n+1} , as before. In all, this results in the difference of fluxes given by

$$\begin{aligned} \int_{t_c}^{t^{n+1}} \varphi_x \left(x_{N+\frac{1}{2}}, t' \right) dt' + \int_{t^n}^{t_c} \varphi_x(s(t'), t') dt' - \int_{t^n}^{t^{n+1}} \varphi_x \left(x_{N-\frac{1}{2}}, t' \right) dt' \\ \approx (\Delta t - \delta t_N) F_{N+\frac{1}{2}}^{n+1} + \delta t_N F_{\bar{s}_N}^{I, N+1} - \Delta t F_{N-\frac{1}{2}}^{n+1}. \end{aligned} \quad (4.12)$$

This concludes the enumeration of time integral discretizations.

4.2.4 Overall Time-Stepping Procedure

We will now summarize the procedure for advancing the front position s and solution ϕ to time t^{n+1} , given their values at t^n .

Motion of the front

First, the front is moved by approximating its speed at t^n using (4.2), and then using this to find the new location of the front. Specifically, we use

$$\varphi_x(s^n, t^n) \approx F^{f, n},$$

where $F^{f, n}$ is the flux at t^n , given by (3.16). The front is moved by applying

$$s^{n+1} = s^n - \text{St } \Delta t F^{f, n}. \quad (4.13)$$

Here Δt is chosen so that $\Delta s = \text{Cr } \Delta x$, where $\text{Cr} \leq 1$ is the Courant number. This is not a requirement for the stability of the algorithm, but only a constraint that the front may move at most one cell length per time step.

Note that (4.13) is an explicit first-order method for moving the interface, and requires only that $F^{f, n}$ be a first-order accurate estimate of the front speed. The result is that an $O(\Delta t^2)$ error is introduced in the front location, each time step. If it is desired to

have a second-order accurate discretization of the of the front motion, we may employ the explicit Adams-Bashforth method [37], which uses velocities at both t^{n-1} and t^n :

$$s^{n+1} = s^n - \text{St} \frac{\Delta t}{2} \left(3F^{f,n} - F^{f,n-1} \right). \quad (4.14)$$

In order for this method to be second-order accurate, Δt must be constant. In addition, $F^{f,n}$ must now be second-order accurate, so that the error in the front location, induced by applying (4.14), is $O(\Delta t^3)$ every time step. Finally, because the value of $F^{f,n-1}$ is not available initially, we must use (4.13) for the first time step. Although this causes an $O(\Delta x^2)$ error in the front position, this is of the same order as that caused by (4.14) after $O(\Delta x^{-1})$ time steps. Thus, the algorithm should still produce a second-order accurate front position.

Advancing the heat equation

After moving the front, we have all the geometric properties necessary to approximate the necessary integrals. We will now go through each of the cases again and write down the finite difference formulas for ϕ^{n+1} , for the two cases. Note that in cells that do not contain the front during the time step, we have the simple update formula

$$\phi_i^{n+1} - \Delta t (L\phi^{n+1})_i = \phi_i^n, \quad (4.15)$$

for the backward Euler algorithm, and

$$\phi_i^{n+1} - \frac{\Delta t}{2} (L\phi^{n+1})_i = \phi_i^n + \frac{\Delta t}{2} (L\phi^n)_i \quad (4.16)$$

for the hybrid algorithm. These are the same methods from Chapter 3.

Front remains in cell N . In this case, we combine all the approximations of integrals into a single update formula in cell N ,

$$\Lambda^{n+1} \Delta x \text{avg}(\phi_N^{n+1}, \phi_{N-1}^{n+1}) - \Lambda^n \Delta x \text{avg}(\phi_N^n, \phi_{N-1}^n) = \Delta t \left(F_{\bar{s}_N}^{f,n+1} - F_{N-\frac{1}{2}}^{n+1} \right). \quad (4.17)$$

This comes from substituting (4.7) and (4.8) into (4.3), and represents a first-order accurate discretization of the heat equation with a moving boundary, in cell N .

Front crosses cell edge. In this case, we require a discretization of (4.3) in both cell N and $N+1$. The discretization in cell $N+1$ is derived by combining (4.7) and (4.10) to get

$$\Lambda^{n+1} \Delta x \text{avg}(\phi_{N+1}^{n+1}, \phi_N^{n+1}) = \delta t_{N+1} \left(F_{\bar{s}_{N+1}}^{f,n+1} - F_{N+\frac{1}{2}}^{n+1} \right). \quad (4.18)$$

In cell N , we use the interpolated gradient in (4.12), given by (4.11), to obtain

$$\Delta x \phi_N^{n+1} - \Lambda^n \Delta x \text{avg}(\phi_N^n, \phi_{N-1}^n) = (\Delta t - \delta t_N) F_{N+\frac{1}{2}}^{n+1} + \delta t_N F_{\bar{s}_N}^{f,n+1} - \Delta t F_{N-\frac{1}{2}}^{n+1}. \quad (4.19)$$

Again, each of these formulas is first-order accurate in space and time.

4.2.5 Discrete conservation

The time integral of $\varphi_x(s(t), t)$ is proportional to the distance moved by the front during the time step; this is expressed in (4.4). We could make use of this fact, and enforce an inhomogeneous Neumann boundary condition when updating ϕ^{n+1} , thus producing a single step, conservative algorithm. If we were to do this, however, the update equations for ϕ would not explicitly use the boundary condition $\varphi(s(t), t) = 0$. This fact would only appear in the conservative flux representation of (4.3). Thus any error in the solution would allow the value of $\varphi(s(t), t)$ to drift away from 0 after several time steps. In addition, we would not have the effect of the Dirichlet boundary condition, which reduces the effect of

the truncation error by one order in Δx . So we have chosen to use the Dirichlet boundary condition in the discretization, and account for the energy balance by some other means.

Sub-iteration strategy

A first option that is used frequently in the literature, is to try to match the velocity of the front with the discrete flux calculated at the interface. This essentially means performing a sub-iteration between (4.1) and (4.2), to couple the two equations. One problem with this approach is that it involves advancing the heat equation repeatedly, until the (4.4) is satisfied to some desired tolerance. For a conservative method, that tolerance could be nine or ten *orders* of magnitude smaller than the temperature field, thus requiring a significant number of iterations, each involving advancing the heat equation. In two or more dimensions this is a daunting amount of work, so this approach would be more appropriate for a non-conservative or explicit method, and less so for our implicit method.

Lagged conservation term

Instead we choose to calculate the energy mismatch due to (4.4) not being satisfied by our discrete equations. This can be quantified as

$$\delta\phi_N = -\text{St}^{-1} \Delta s - \Delta t F_{\bar{s}_N}^{f,n+1}, \quad (4.20)$$

if the front does not exit cell N , and

$$\begin{aligned} \delta\phi_{N+1} &= -\text{St}^{-1} \delta s_{N+1} - \delta t_{N+1} F_{\bar{s}_{N+1}}^{f,n+1} \\ \delta\phi_N &= -\text{St}^{-1} \delta s_N - \delta t_N F_{\bar{s}_N}^{I,n+1}, \end{aligned}$$

if the front crosses the edge of cell N . Because the front is moved explicitly using (4.13), and the fluxes are first order, we expect this correction to be $O(\Delta t^2)$.

This loss of $\delta\phi$ can be reincorporated on the next time step, to maintain total discrete conservation of ϕ . For example, during that last time step, from t^{n-1} to t^n , a $\delta\phi^{n-1}$ was calculated for the current cell N . This is then added back into the evolution equations from t^n to t^{n+1} :

$$\Lambda^{n+1}\Delta x \text{ avg } (\phi_N^{n+1}, \phi_{N-1}^{n+1}) - \Lambda^n\Delta x \text{ avg } (\phi_N^n, \phi_{N-1}^n) = \Delta t \left(F_{\bar{s}_N}^{f,n+1} - F_{N-\frac{1}{2}}^{n+1} \right) + \delta\phi_N^{n-1}, \quad (4.21)$$

for instance, where the front does not exits cell N during the time step. Analogous expressions for the other cases are derived simply by adding $\delta\phi^{n-1}$ into the right-hand side of the evolution equations.

Flux mismatch for the hybrid method

The hybrid discretization has an additional difficulty due to the mismatch of fluxes through edge $N - \frac{1}{2}$ in equations (4.16) and (4.17) or (4.18):

$$\delta\phi_N = -\text{St}^{-1}\delta s_N - \delta t_N F_{\bar{s}_N}^{I,n+1} + \frac{\Delta t}{2} \left(F_{N-\frac{1}{2}}^{n+1} - F_{N-\frac{1}{2}}^n \right), \quad (4.22)$$

if the front crosses the edge of cell N , or

$$\delta\phi_N = -\text{St}^{-1}\Delta s - \Delta t F_{\bar{s}_N}^{f,n+1} + \frac{\Delta t}{2} \left(F_{N-\frac{1}{2}}^{n+1} - F_{N-\frac{1}{2}}^n \right), \quad (4.23)$$

if it does not cross.

4.2.6 Summary of time-stepping algorithm

To summarize, the solution is advanced in time by performing each of these steps in succession:

- Explicitly move the front, using the temperature gradient at t^n in (4.13).

- Solve for ϕ^{n+1} using the algorithm for the heat equation with moving boundary, and reincorporating the loss term $\delta\phi^{n-1}$ using (4.21).
- Calculate the loss $\delta\phi^n$ for this timestep, using (4.20).

At this point, we have updated ϕ^{n+1} and s^{n+1} , and repeat each part for the next time step. There are no specific start-up procedures, other than setting ϕ and s to their given initial values, and the conservation term $\delta\phi$ equal to zero.

4.3 Results in One Dimension

We will now evaluate the algorithm described above in a series of numerical test problems. In order to separate out the different contributions to the discrete error, we will calculate numerical solutions to for an intermediate problem, the heat equation on a changing domain (4.1), without the constraint (4.2) for the Stefan problem. The boundary motion will be specified using an analytic solution to the Stefan problem in one dimension, so that our results for this simpler problem can help us evaluate the results we obtain for the entire algorithm. The second test will be the complete Stefan problem, using both a conservative and non-conservative algorithms.

4.3.1 Analytic Solution in One Dimension

Carslaw and Jaeger [15] provide an analytic solution for the Stefan problem in one dimension. Consider the initial value problem for the heat equation (4.1):

$$\varphi(x, 0) = 0 \text{ for } x > 0, \text{ and } \varphi(0, t) = 1, \varphi(s(t), t) = 0, \quad (4.24)$$

where we specify $s(t) = 2\kappa\sqrt{\beta t}$, with κ a constant parameter. We have also set β equal to a constant in (4.1), so as to obtain the following analytic solution for (4.24):

$$\varphi(x, t) = 1 - \operatorname{erf}\left(\frac{x}{2\sqrt{\beta t}}\right) / \operatorname{erf}(\kappa) \quad \text{for } x \in (0, s(t)). \quad (4.25)$$

The normalized error function,

$$\operatorname{erf}(t) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-r^2} dr,$$

is easily calculated. At any time $t > 0$, φ is C^∞ on the interval $x \in (0, s(t))$, due to the smoothness of the error function.

We may also calculate $\varphi_x(s(t), t)$ from (4.25),

$$\varphi_x(s(t), t) = -\frac{1}{\sqrt{\pi\beta t}} \frac{e^{-\kappa^2}}{\operatorname{erf} \kappa}.$$

With the definition of $s(t)$ given above, we have an expression for the front speed, $u(t) = \dot{s}(t) = \kappa\sqrt{\frac{\beta}{t}}$. After applying the constraint (4.1) on $\dot{s}(t)$, we obtain the following non-linear algebraic equation for κ :

$$\operatorname{St} e^{-\kappa^2} = \sqrt{\pi} \kappa \operatorname{erf} \kappa,$$

where St is the Stefan number. For $\operatorname{St} \ll 1$, time scales for diffusion are smaller than those for the front motion, which produces an almost linear solution between $x = 0$ and $x = s(t)$. When $\operatorname{St} \gg 1$, the front moves more quickly, and slower diffusion causes the solution to have larger second derivatives.

4.3.2 Heat Equation with Moving Boundary

We will now use this analytic solution to the Stefan problem to test the finite volume algorithm for the heat equation with a specified boundary motion, $s(t) = 2\kappa\sqrt{\beta t}$. We will discretize $s(t)$ in a way that simulates the behavior of the advection algorithm: choose some $\text{Cr} \leq 1$, and solve the following equation for the Δt that produces the desired front motion:

$$\text{Cr} \Delta x = u \Delta t \quad \Rightarrow \quad \Delta t = \frac{1}{4\kappa^2\beta} [(s^{n+1})^2 - (s^n)^2]. \quad (4.26)$$

The speed of the front during the time step is determined from $u = \frac{\Delta s}{\Delta t}$. To exercise all the dependencies of the discretization on Λ^n , we will set $\text{Cr} = \sqrt{\frac{1}{2}}$, an irrational number. When Cr is a rational number, Λ^n cycles through a list of values that depends on the common factors of the numerator and denominator of Cr . In that case, we could find that $\Lambda^n \ll 1$ or $\Lambda^n \approx 1$ never occur, and the algorithm would not be thoroughly tested. With Cr irrational, this is almost guaranteed to occur, given a large number of time steps.

Problem 1: Heat equation with $\text{St} = 1$

To perform the calculation, we choose an initial time $t^0 = 0.1$, and set $\phi(x_i, t^0) = \varphi(x_i, t^0)$, and $s^0 = s(t^0)$, where $s(t)$ is of the form given above. The discrete solution is advanced using (4.26) with $\text{Cr} = \sqrt{\frac{1}{2}}$, for a total of $n = (2\Delta x)^{-1}$ time steps. We have set $\beta = 1$ and $\text{St} = 1$ in the following calculations.

The exact solution is plotted in Figure 4.5 at times $t = 0.1, 0.3, 0.6$, and we see that it varies smoothly between $\varphi = 1$ at $x = 0$, to $\varphi = 0$ at $x = s(t)$. In each of the following cases, the discrete solution at cell centers is compared to the exact solution at the same final time, using (4.25). Because it is possible that $x_i > s(t^n)$, the exact solution (4.25) must be evaluated outside of the domain for some cases, and we assume that this is equivalent to smoothly extending the solution past the domain boundary.

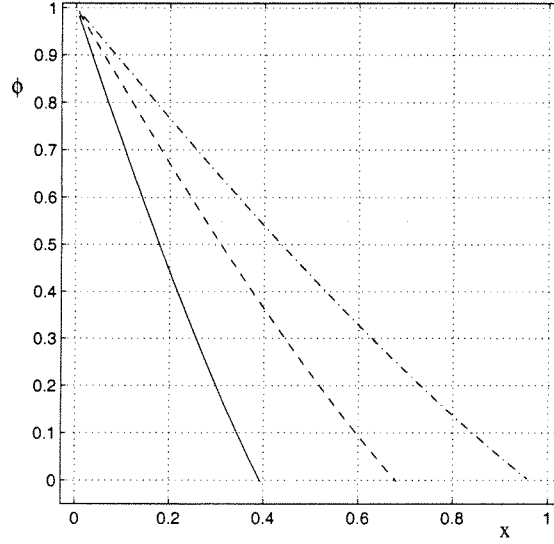


Figure 4.5: The exact solution to the Stefan problem in one dimension, for $St = 1$, is plotted at times $t = 0.1, 0.3, 0.6$. Because our discrete solution ϕ can extend beyond the front location, we extend the exact solution using (4.25), even when the result is less than zero.

In Figure 4.6 we show the error in the discrete solution, $\xi = \phi_i^n - \varphi(x_i, t^n)$, for the non-conservative backward Euler method. The results are presented for six different $\Delta x = 0.1 \cdot 2^{-i}, i \in \{2, 3, 4, 5, 6, 7\}$, while the final time and front position is the same for all i . The solution error is smooth, and reduces by a factor of two with each refinement of Δx , suggesting that the algorithm is first-order accurate. Also in Figure 4.6 is the difference between the calculated front speed, using (4.13), and the front speed $u^e(t) = \dot{s}(t)$ from the exact solution (which is $u^e(t) \approx 1 \sim 2$ for $t = 0.1 \sim 0.4$). This also appears to be first-order accurate, as expected.

In Figure 4.7 we plot the error in the discrete solution, for the hybrid method, with the same values of Δx . The error is relatively smooth for $x \lesssim 0.37$, but increasingly more oscillatory as x approaches the front location. Because the initial location of the front is $s(0.1) \approx 0.37$, we conclude that this is due to the error introduced at the front every time step. Because the hybrid method does not dampen high-frequency components of the error, it persists, even after $O(N)$ time steps. It is important to note, however, that the

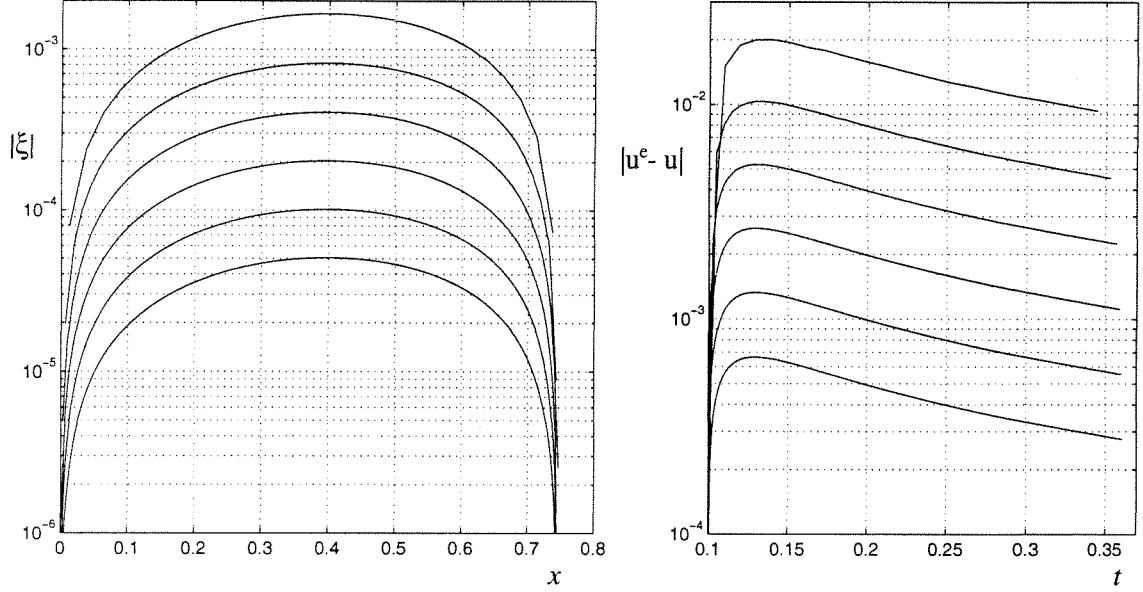


Figure 4.6: We plot the error in the discrete solution (ξ on the left) for the backward Euler method applied to the heat equation, with $St = 1$, after $n = \frac{1}{2\Delta x}$ time steps. The difference between the exact front speed, and that calculated from $F^{f,n}$, is plotted versus time on the right. In both cases, the errors appear smooth and $O(\Delta x)$.

error appears to converge like $O(\Delta x^2)$, so that the method is second order accurate. This confirms that the $O(\Delta x)$ truncation error in the cells neighboring the front do not adversely affect the overall accuracy of the discrete solution.

Figure 4.8 shows, however, that the error in the front speed is extremely noisy, albeit relatively small. We attribute this to the combination of a high-frequency component in the solution error, and the polynomial interpolation used to approximate F^f . By calculating the linear least-squares fit to the logarithmic data, we can that u is behaving roughly like $O(\Delta x^2)$, indicating that the gradient calculation is also second-order accurate (Figure 4.8).

In this section we are approximating the heat equation with a moving boundary, and thus do not have to satisfy the constraint given by (4.4). However, our failure to satisfy (4.4) can be still be calculated from $\delta\phi$ in (4.20), which is plotted in Figure 4.9. As stated above, $\delta\phi$ behaves like $O(\Delta x^2)$, for both the backward Euler (left) and hybrid (right)

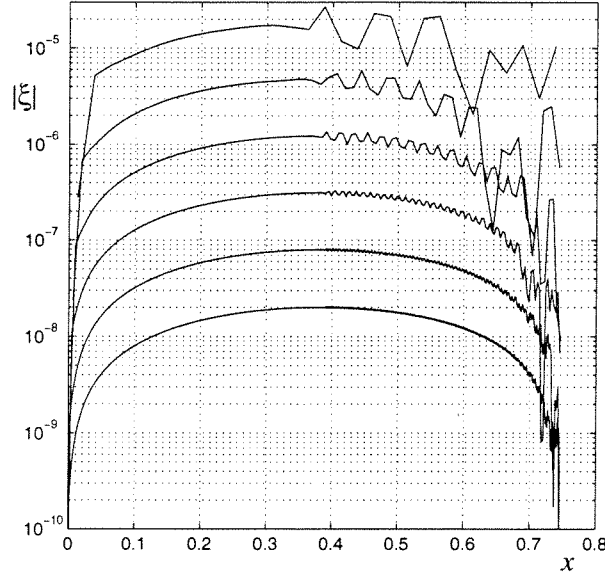


Figure 4.7: We plot the error in the discrete solution for the hybrid method applied to the heat equation, with $St = 1$, after $n = \frac{1}{2\Delta x}$ time steps. Note the oscillations in the error for $x \gtrsim 0.37$, which we attribute to the hybrid method's inability to damp high-frequency components of the error, which are introduced at the front each time step.

methods.

Finally, we can consider what happens when the front motion is specified, but $\delta\phi$ is reincorporated using (4.21). This resembles the complete algorithm for the Stefan problem, while avoiding any feedback between the algorithms for moving the front and advancing the heat equation. Figure 4.10 shows the error in the solution, which is smooth, and indicates that the algorithm is still first-order accurate. Figure 4.11 shows that the error in the calculated front speed is as noisy as that of the non-conservative hybrid method. In addition, the linear least-squares fit suggests that the calculated speed is first-order accurate, as expected.

When the hybrid algorithm, with the lagged conservation term $\delta\phi$, is applied to this same test problem, it was found to be unstable. We again attribute this to a feedback mechanism between the calculation of $\delta\phi$, which uses quadratic polynomial interpolation to calculate F^f , and the lack of damping provided by the hybrid operator, for high-frequency

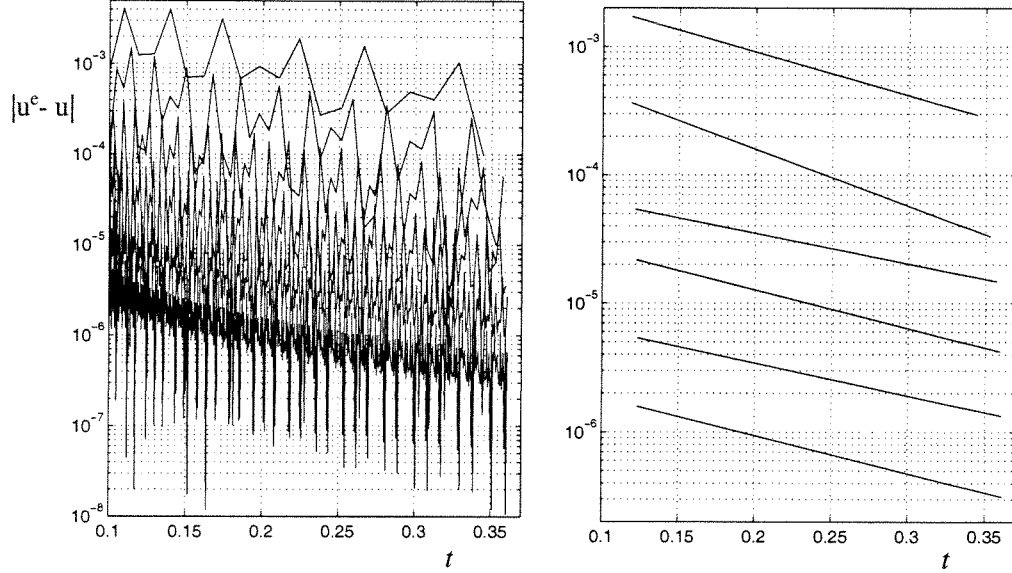


Figure 4.8: We plot the error in the calculated front speed for the hybrid method applied to the heat equation with $St = 1$, for $n = \frac{1}{2\Delta x}$ time steps determined from (4.26). The error for each grid spacing is very oscillatory (left), but a linear least-squares fit to the data (right) appears to converge like $O(\Delta x^2)$.

components of the solution.

Problem 2: Heat equation with $St = 100$

For $St \gg 1$, the front moves more quickly than the diffusion of ϕ , so that the exact solution's second derivatives in space are larger than the $St = 1$ case. The exact solution for $St = 100$ is plotted in Figure 4.12 at times $t = 0.01, 0.03, 0.06$. In the following numerical experiments, the Stefan number is the only parameter that differs from the previous section.

In Figure 4.13 we show the error in the discrete solution, for the backward Euler method; it appears to be first-order accurate for this case. The error in the calculated front speed is also plotted in Figure 4.13, where we see that it is noisier than the results for $St = 1$, but still $O(\Delta x)$. Figure 4.14 presents the solution error for the hybrid method, which again appears to be second-order accurate, although a bit less noisy. The velocity error closely resembles the very noisy results for $St = 1$, so that we only present the least-squares, linear fit to the data, which appears to be approximately second-order accurate, as well.

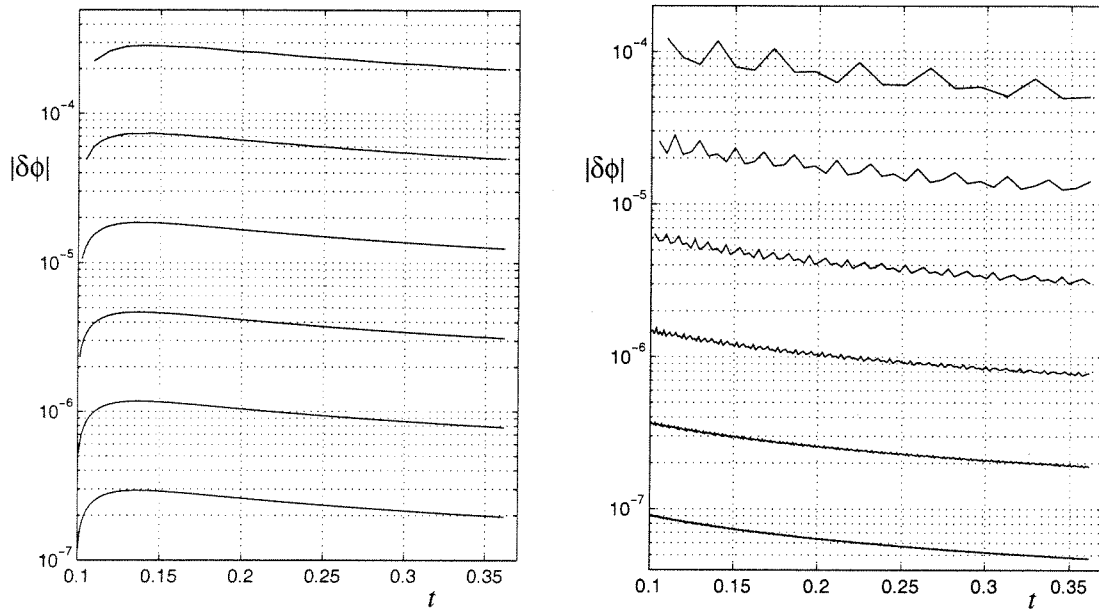


Figure 4.9: We plot the total conservation losses $\delta\phi$ versus time for the backward Euler (left) and hybrid methods (right) applied to the first test problem. Both $\delta\phi$ are $O(\Delta x^2)$, although the hybrid case appears to be more oscillatory.

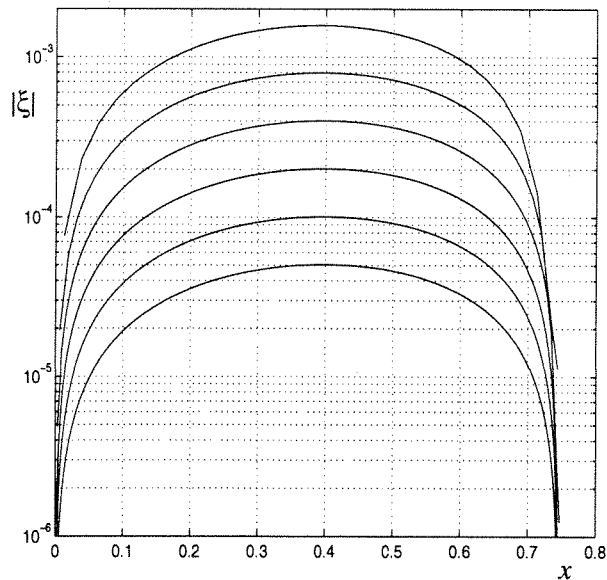


Figure 4.10: We plot the error in the discrete solution for the backward Euler method with the lagged conservation term, applied to the first test problem. Again, the error appears to be $O(\Delta x)$.

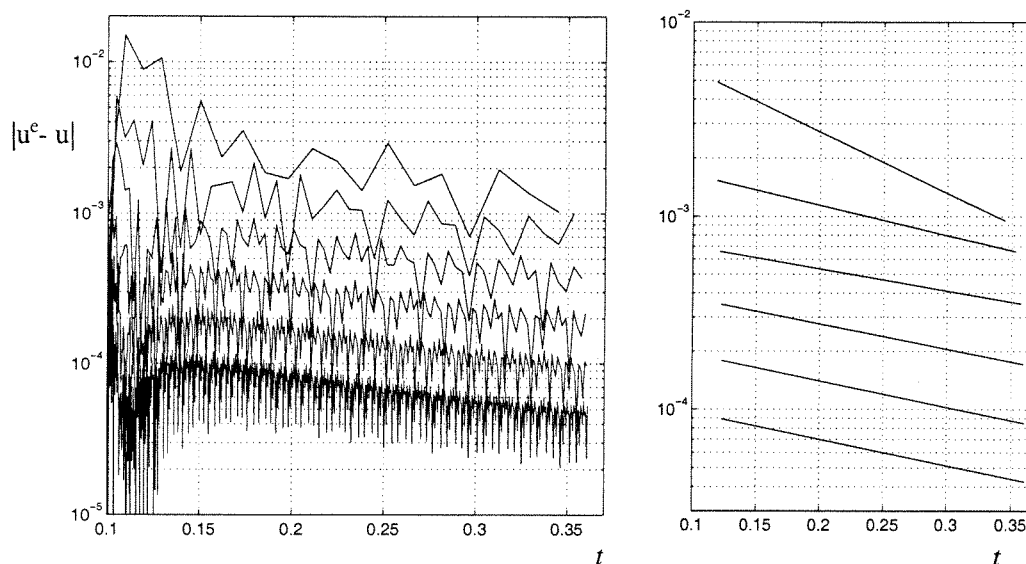


Figure 4.11: We plot the difference between the calculated and exact front speeds, versus time, for the first test problem. Again, there is a oscillatory component to the error (left), but a least-squares fit (right) suggests that it is still a first-order accurate approximation.

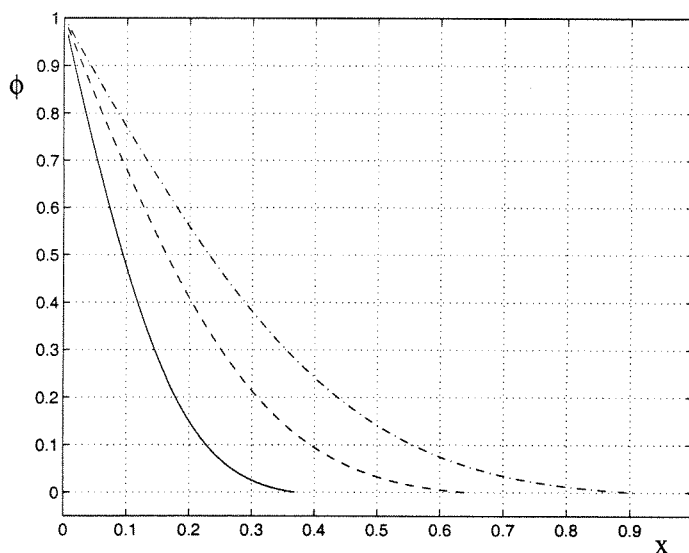


Figure 4.12: The exact solution to the Stefan problem in one dimension, for $St = 100$, is plotted at times $t = 0.01, 0.03, 0.06$ (from left to right).

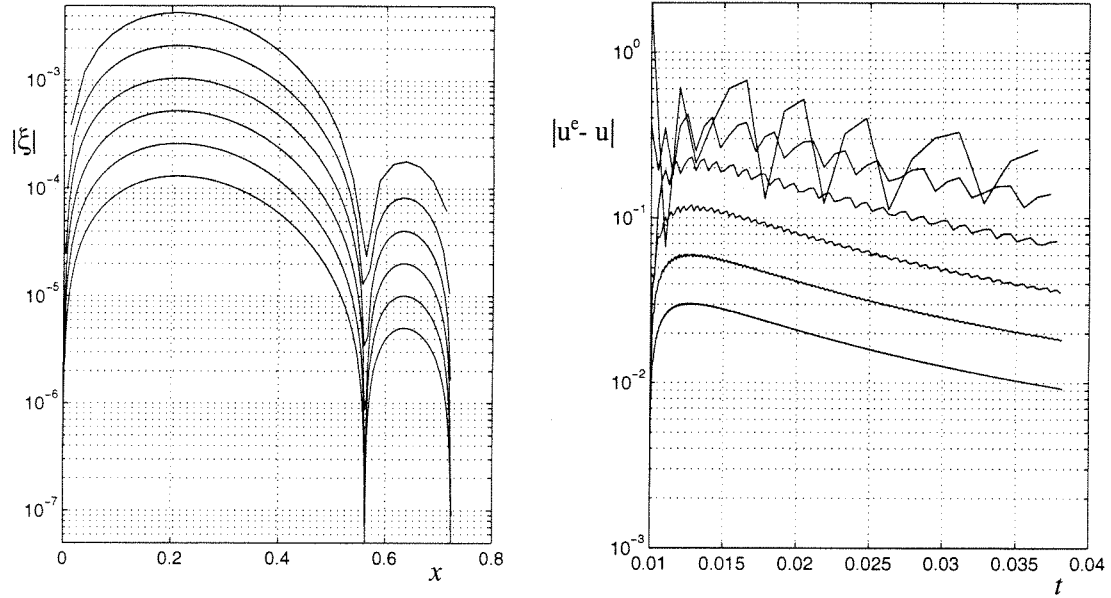


Figure 4.13: We plot the error in the discrete solution (left) and calculated front speed (right) for the second test problem, with $St = 100$. Both appear to be $O(\Delta x)$.

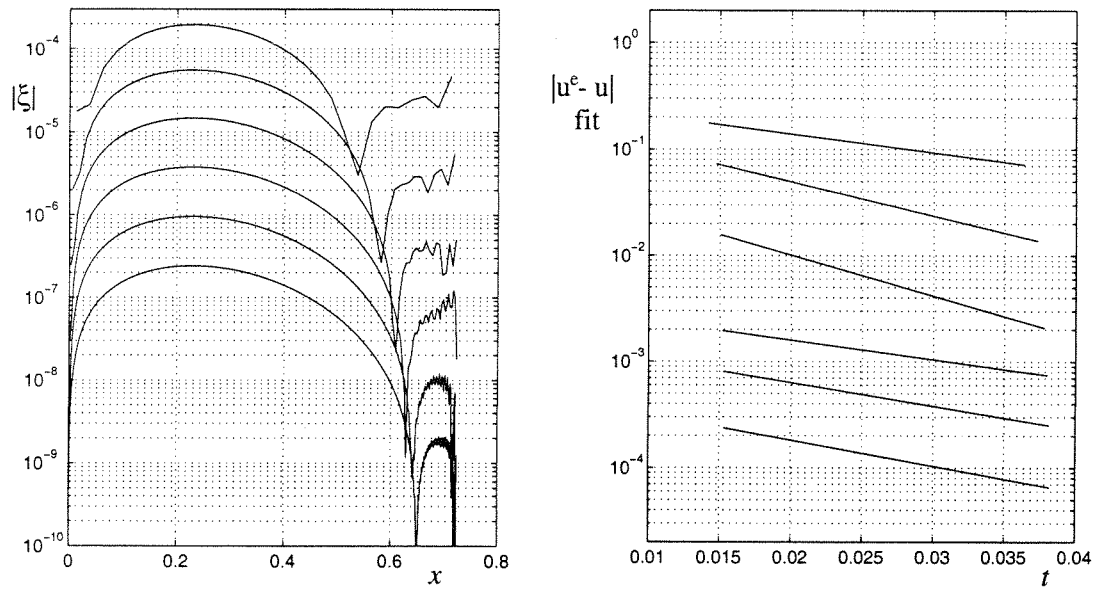


Figure 4.14: For the hybrid method applied to the second test problem, the error in the discrete solution (left) appears to be $O(\Delta x^2)$. The error in the calculated front speed looks very much like Figure 4.8, and the linear least-squares fit suggests that it is also $O(\Delta x^2)$.

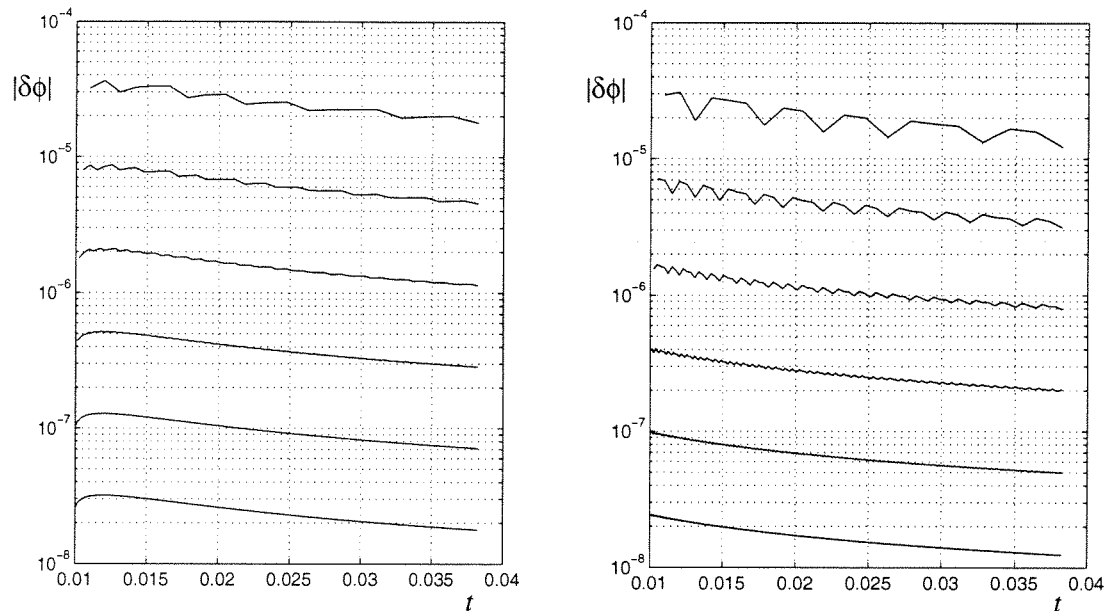


Figure 4.15: We plot the total conservation losses $\delta\phi$ versus time, for the backward Euler (left) and hybrid (right) algorithms for the second test problem. Both $\delta\phi$ are $O(\Delta x^2)$, as in the first test problem.

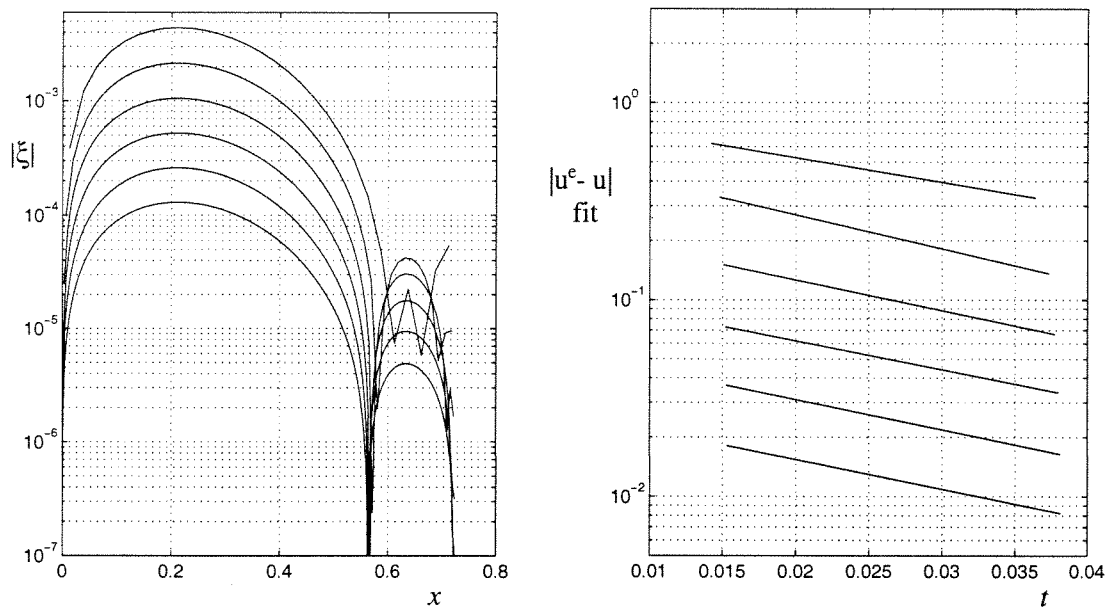


Figure 4.16: When we reincorporate $\delta\phi$, the backward Euler method applied to the second test problem still produces a first-order accurate discrete solution (left). The error in the calculated front speed is again very noisy, like Figure 4.11, but the linear least-squares fit suggests that it is $O(\Delta x)$.

The deviation from conservation for this test problem is given in Figure 4.15, which shows that it is again $O(\Delta x^2)$ for both methods. Reincorporating $\delta\phi$ into the backward Euler method is also stable for the $St = 100$ case. Figure 4.16 shows the error in the solution, as well as the least-squares, linear fit to the velocity error. Again, we find that the calculated velocity is first-order accurate. However, when $\delta\phi$ is reincorporated with the hybrid algorithm, it is again found to be unstable.

Conclusions for the heat equation

Given these results, we may conclude that our discretization for the heat equation with moving boundaries performs as expected, when we are not concerned with the constraint (4.4). The backward Euler method produces a first-order accurate solution, and an approximation of the front speed that is $O(\Delta x)$ as well. Note that the backward Euler method is still a conservative discretization of the heat equation with a moving boundary. When the conservative losses $\delta\phi$ are reincorporated with (4.21), the accuracy of the method remains the same, although the error in the calculated speed has a significant oscillatory component.

The hybrid method, which is not conservative due to the mismatch in fluxes at $x_{N-\frac{1}{2}}$, does produce both a second-order accurate solution and approximation to the front speed, as we concluded from the derivation. This is particularly significant, because the algorithm is identical to the backward Euler method in the cells that contain a portion of the front. These two test problems for the heat equation have demonstrated that the lower accuracy near the moving boundary, does not inhibit the overall accuracy of the solution.

4.3.3 Stefan Problem

For the Stefan problem, the algorithm is exactly the same, except that we now use an explicit algorithm for advancing the front. Again, we perform calculations for both $St = 1$ and $St = 100$, using either the backward-Euler method, given by (4.15), or the

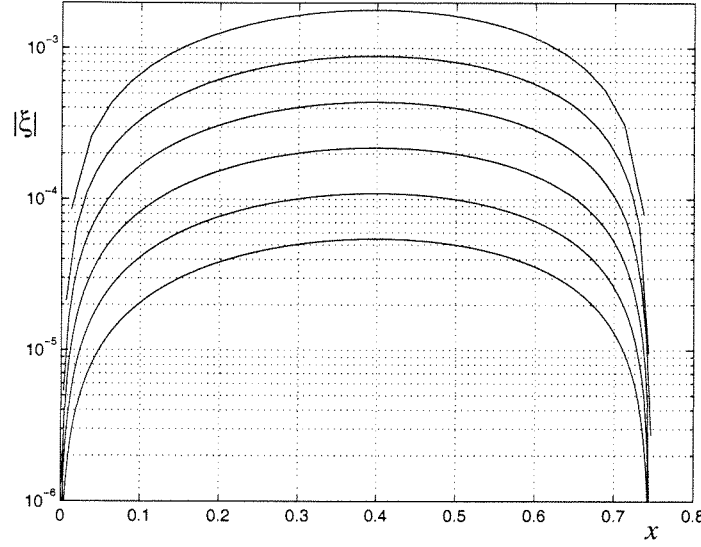


Figure 4.17: The backward-Euler method for the heat equation and forward-Euler method for advancing the front have been applied to the third test problem, with $St = 1$. The error in the discrete solution is smooth, and appears to be first-order accurate.

hybrid method (4.16). We have demonstrated that the backward-Euler method is first-order accurate for the heat equation, so we pair it with the explicit Euler method (4.13) with $Cr = \sqrt{\frac{1}{2}}$, to approximate the front motion. For the hybrid algorithm, we were able to obtain a second-order accurate approximation of the front speed, which suggests using the Adams-Bashforth method (4.14) to update the front position. However, in that case, Δt must be held fixed for the duration of the calculation. For that case, Δt is calculated from $Cr = \sqrt{\frac{1}{2}}$ and the initial calculated front speed. However, both approaches take a total of $n = (2\Delta x)^{-1}$ time steps.

Problem 3: Stefan problem with $St = 1$

Figure 4.17 shows the results for the backward Euler method for the six different $\Delta x = 0.1 \cdot 2^{-i}, i \in \{2, 3, 4, 5, 6, 7\}$, with initial time $t = 0.1$. The error in the discrete solution appears to be $O(\Delta x)$, as expected. Figure 4.18 plots the error in the position of the front $|s(t^n) - s^n|$, as well as the error in the calculated speed of the front. Both of these appear

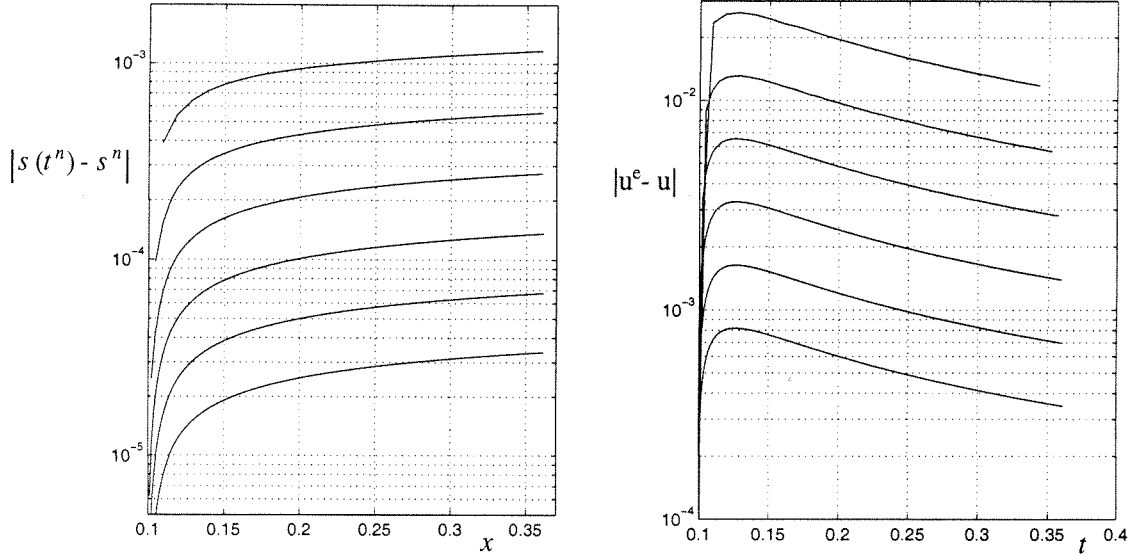


Figure 4.18: We plot the error in the front position (left) and the calculated speed (right) versus time, for the backward-Euler method applied to the third test problem. In both cases, the error appears to be $O(\Delta x)$ and smooth.

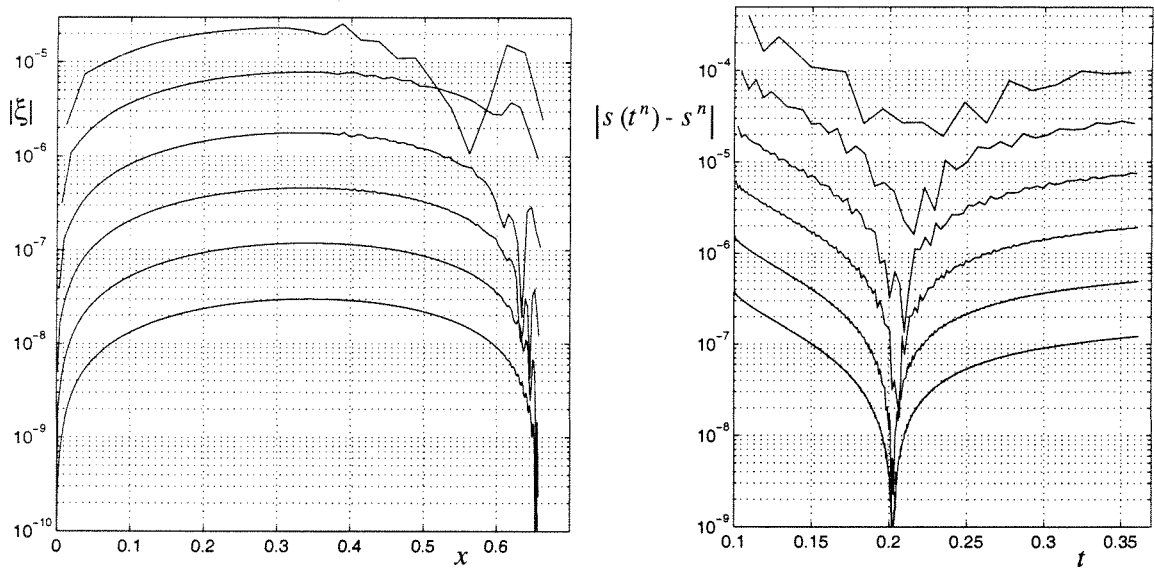


Figure 4.19: We plot the error in the discrete solution (left), for the third test problem, using the hybrid method with Adams-Bashforth for advancing the front in time. This results in a second-order accurate front position (right) and solution.

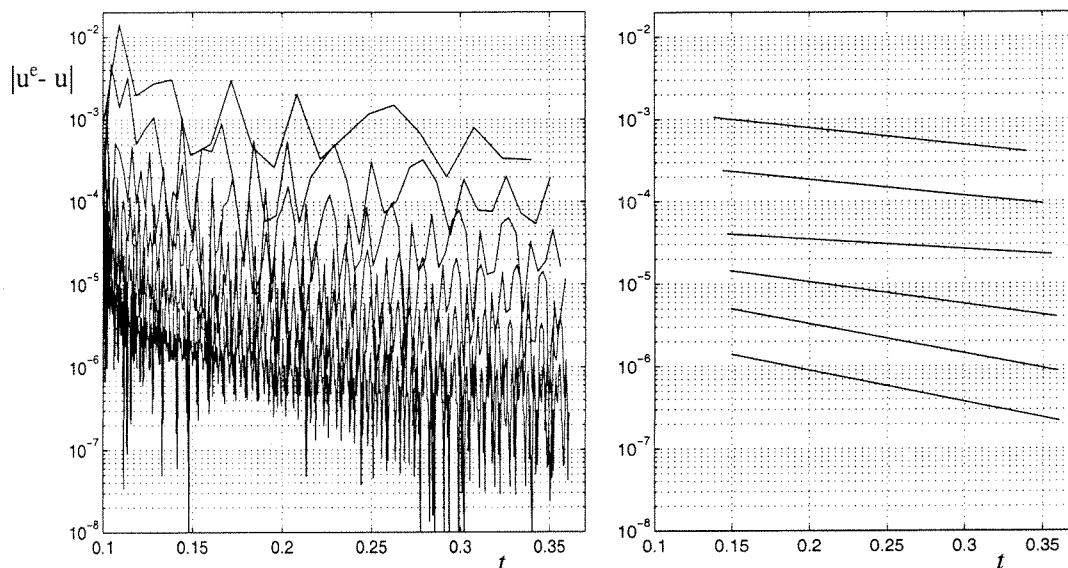


Figure 4.20: We plot the difference between the calculated and exact front speeds, versus time, for the hybrid method applied to the first test problem. Again, there is a oscillatory component to the error (left), but a least-squares fit (right) suggests that it is $O(\Delta x^2)$.

to converge smoothly, and indicate that the algorithm is first-order accurate overall.

From Figure 4.19, we may deduce that the hybrid algorithm is able to produce second-order accurate (but mildly noisy) results for both ϕ and the front position. In Figure 4.20 we see that the calculated front speed continues to be very noisy, although the least-square linear fit suggests that it is still $O(\Delta x^2)$. Indeed, if the front speed were not second-order accurate, we would see the accuracy of the calculated front position deteriorate, as well.

The calculations attempting to maintain conservation with the lagged source term $\delta\phi$ were unstable, for both the backward-Euler and hybrid methods. Because this was not the case for the backward-Euler method, when the boundary motion was specified, we must conclude that this is a feedback mechanism between reincorporating $\delta\phi$, which is $O(\Delta t^2)$, and the front speed calculation, which uses the quadratic polynomial interpolation in evaluating F^f . It is apparent that the backward Euler method is unable to damp the erratic source term enough to maintain the stability of the algorithm.

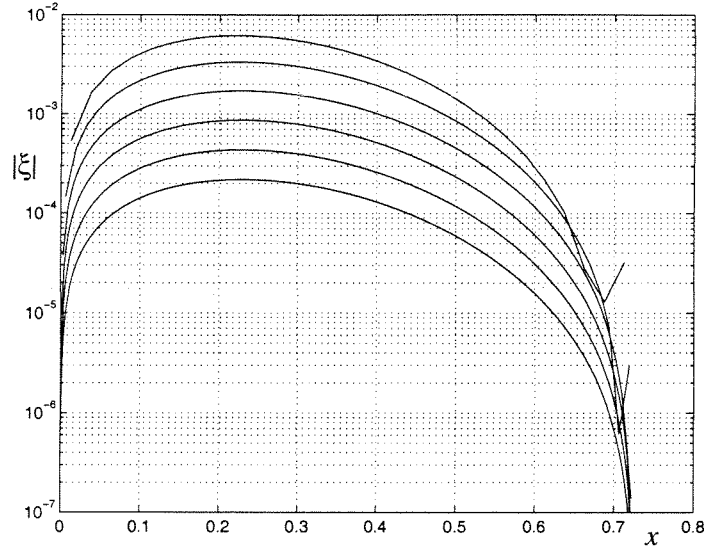


Figure 4.21: The backward-Euler method for the heat equation and forward-Euler method for advancing the front result in a first-order accurate solution for the the fourth test problem, with $St = 100$.

Problem 4: Stefan problem with $St = 100$

For $St = 100$, the calculations required a slightly finer grid resolution to discern the accuracy of each algorithm; we have chosen six different $\Delta x = 0.1 \cdot 2^{-i}$, $i \in \{3, 4, 5, 6, 7, 8\}$ for this case. Figure 4.21 shows the results from applying the backward-Euler method, which show that the error behaves like $O(\Delta x)$, as expected. Similar results are obtained for the front position and calculated speed (Figure 4.22, although they are considerably more noisy than the corresponding results for $St = 1$).

The hybrid algorithm, which uses an Adams-Bashforth discretization of the front motion, continues to produce second-order accurate results for the solution and front position (Figure 4.23). The error in the front speed continues to be noisy but second-order accurate, as in Figure 4.20. Figure 4.24 shows the final steps for each run, from which the noisy component of the error is evident.

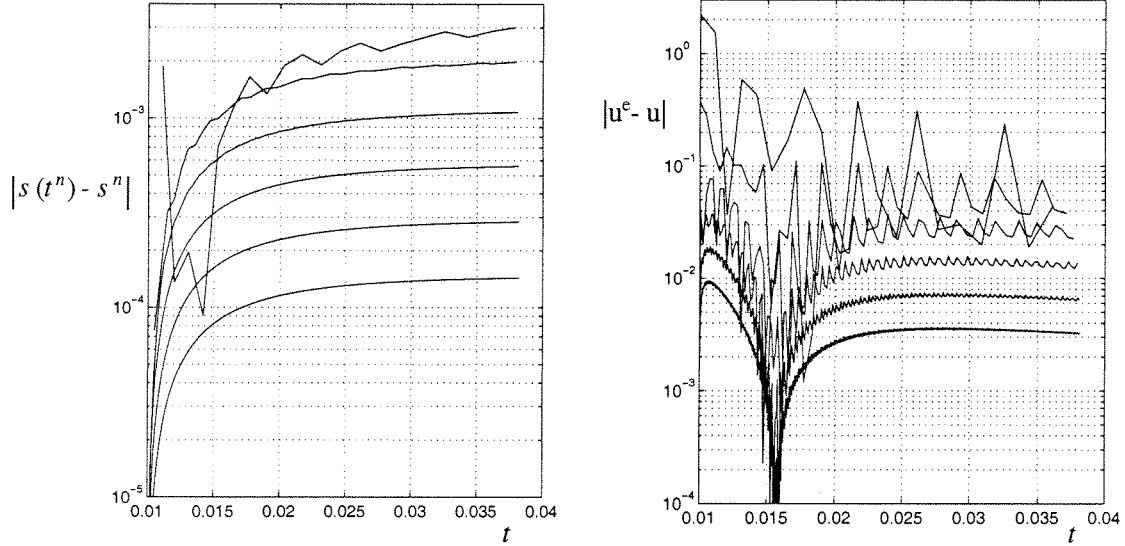


Figure 4.22: We plot the error in the front position (left) and the calculated speed (right) versus time, for the backward-Euler method applied to the fourth test problem. In both cases, the error appears to be $O(\Delta x)$, but more oscillatory than the results in Figure 4.18, with $St = 1$.

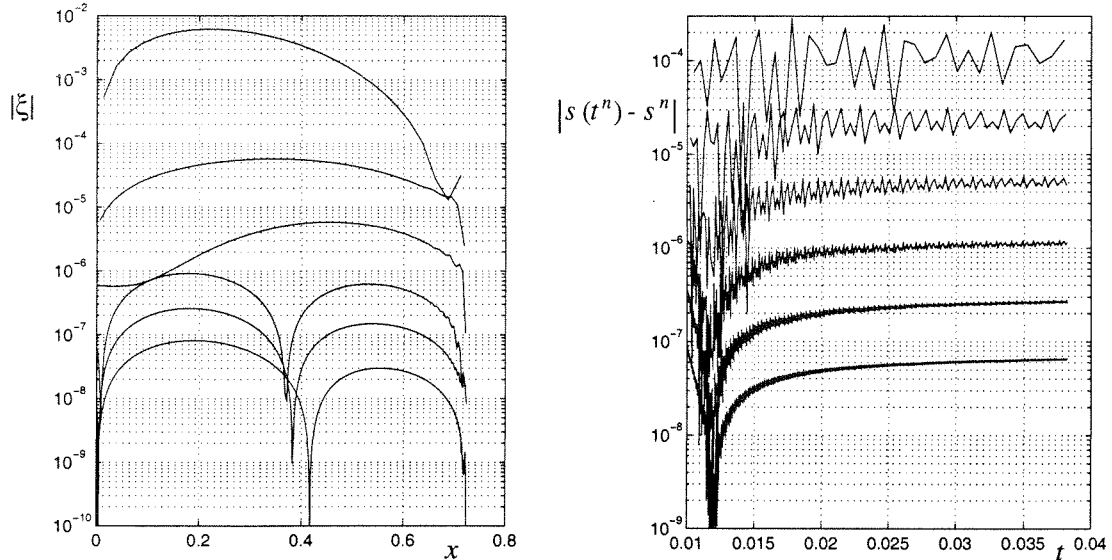


Figure 4.23: We plot the error in the discrete solution (left), for the fourth test problem, using the hybrid method, with Adams-Bashforth for advancing the front in time. The $O(\Delta x^2)$ rate of convergence of the discrete solution is not apparent except at the finest grid resolutions. The front position (right) is oscillatory, but second-order accurate.

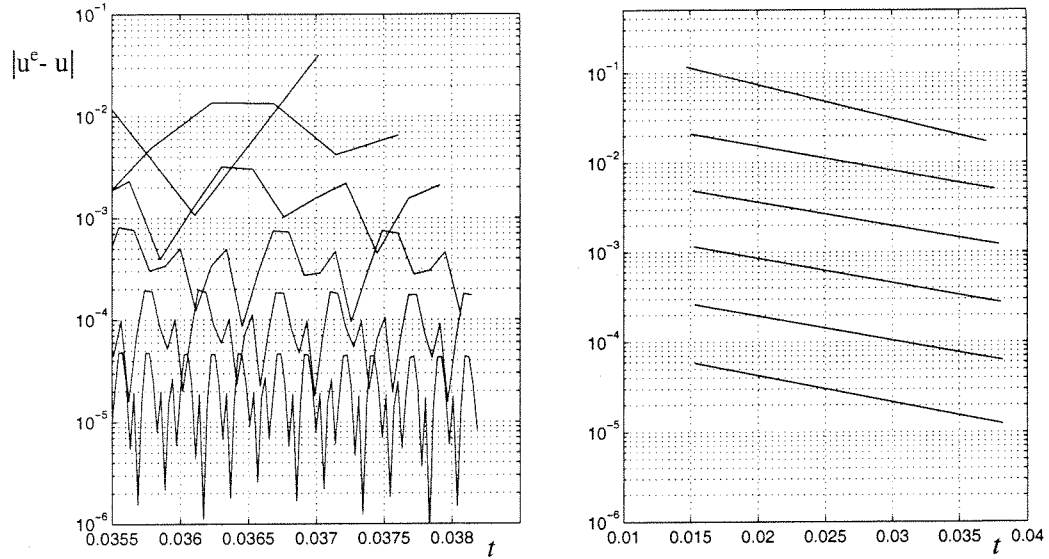


Figure 4.24: A close-up of the error in the calculated front speed, plotted versus time, is presented for the hybrid method applied to the fourth test problem. Again, there is an oscillatory component to the error (left), but a least-squares fit (right) suggests that it is $O(\Delta x^2)$.

4.4 Front Tracking

Our means of tracking the melting front is based on work done by three sets of authors, employing volume of fluid (“VoF”) methods, which use a scalar volume fraction to track the front location. The first set of authors, Chern and Colella [18], developed an algorithm for tracking shocks in the unsteady Euler equations for compressible fluid flow. This was extended by a Bell, et al. [9], who used a volume-fraction representation, instead of the polygonal curve used in [18]. Finally, a more abstract method was introduced by Pilliod and Puckett [52], where a specific interface reconstruction algorithm was combined with standard advection algorithms for the volume fraction.

4.4.1 Volume of Fluid Algorithm and Derivation

The volume fraction field, $\Lambda_{i,j}$, is first initialized to represent the front location at $t = 0$. This is accomplished with the grid generation algorithm from Chapter 2: the front is treated as piecewise-linear in each cell (i, j) . Intersections with grid lines determine the

linear front representation needed to calculate $\Lambda_{i,j}$. The volume fraction in other cells is defined as before: $\Lambda_{i,j} = 0$ if the cell is outside the domain, while $\Lambda_{i,j} = 1$ if it is inside.

Formally, the motion of the volume fraction is assumed to be governed by an advection equation,

$$\Lambda_t + \mathbf{u} \cdot \nabla \Lambda = 0 ,$$

or in another form,

$$\Lambda_t + \nabla \cdot (\mathbf{u} \Lambda) = \Lambda \nabla \cdot \mathbf{u} , \tag{4.27}$$

where \mathbf{u} is the front velocity. The volume-of-fluid algorithm can be thought of as a consistent discretization for (4.27), in which Λ is piecewise constant. In this discretization, we first reconstruct a linear representation of the front in each cell (i, j) , based on an approximate least-squares fit of a linear front to the volume fraction field, $\Lambda_{i,j}$. We then approximate the front velocity using a discrete version of (Eq. III), and extend the result to cell edges participating in the advection algorithm. The evolution of Λ is then accomplished with an operator-split, flux-based discretization of (4.27), whose fluxes depend specifically on a piecewise linear representation of the front. Finally, the front velocity and updated volume fraction are used to find the front's trajectory within each cell, and derive the areas and centroids of the space-time surfaces required for the finite volume discretization.

Approximate least-squares front reconstruction

This part of the algorithm is due to Pilliod and Puckett [52]. Given the volume fraction field, $\Lambda_{i,j}$, they were able to exactly reconstruct a linear interface, using only $\Lambda_{i,j}$ in cells in the 3×3 block centered on cell (i, j) (which we denote by $\text{ngh}(i, j)$). Their method was demonstrated to be second-order accurate for several test problems, when used in conjunction with an operator-split advection algorithm [52]. We have chosen to use it

here, due to its simplicity and relatively good accuracy.

Consider the volume fractions in $\text{ngh}(i, j)$, and the sums of the volume fraction in each vertical column in $\text{ngh}(i, j)$, σ_{i-1} , σ_i , and σ_{i+1} . For example,

$$\sigma_{i-1} = \sum_{k=j-1}^{j+1} \Lambda_{i-1,k}.$$

It was shown in [52], that if the interface $\partial\Omega$ is linear, with slope $m \in [0, 1]$, and cell (i, j) contains a portion of the interface, then simple finite differences of the σ 's will yield the correct slope, m . Specifically,

$$m = \frac{1}{2} (\sigma_{i+1} - \sigma_{i-1}),$$

if $\partial\Omega$ intersects the left and right sides of the box containing $\text{ngh}(i, j)$. Similarly,

$$m = \sigma_{i+1} - \sigma_i \text{ and } m = \sigma_i - \sigma_{i-1},$$

if $\partial\Omega$ intersects the left and top, or bottom and right, sides of the box, respectively. Thus, one of these difference stencils will exactly determine the slope of a linear front. If $m \in (1, \infty)$, switching the roles of i and j , and performing these same differences, will produce the correct result. Similarly, if $m < 0$, then flipping the volume fraction array upside-down and going through the procedure, yields the correct slope.

For a curved front, it is not obvious which of these difference approximations to use. The solution suggested in [52], is to calculate each of the six differences (forward, backward, or centered differences of column- and row-sums), and find which one best matches the volume fraction array. Specifically, for each of the six slopes, we calculate what the 3×3 volume fraction array would be, if the interface were linear, with the given slope, and matched $\Lambda_{i,j}$ exactly. This volume fraction array, which we will call $\Lambda_{k,l}^m$ for $(k, l) \in \text{ngh}(i, j)$, is easily calculated from the slope and volume fraction $\Lambda_{i,j}$ (the appendix in [18] provides a

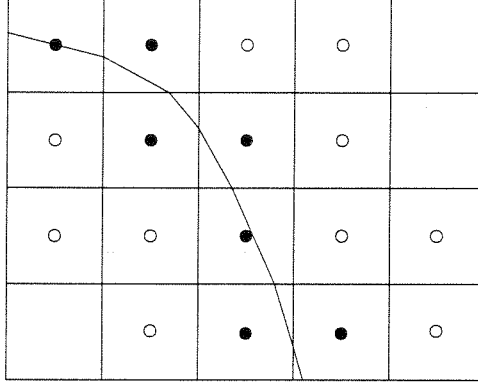


Figure 4.25: To extend the front velocity, we $u_{i,j}$ is calculated first in the cells that contain a portion of the front (filled circles, for which $P_{i,j} = 1$). These velocities are then extended to the remaining cells in $\text{ngh}(i, j)$ (marked with unfilled circles, where $P'_{i,j} = 1$ but $P_{i,j} = 0$), by calculating \bar{u} from (4.29).

simple explanation). We can then define a norm of the error between Λ^m and Λ using the equation

$$\|\Lambda^m - \Lambda\| = \sum_{l=j-1}^{j+1} \sum_{k=i-1}^{i+1} (\Lambda_{k,l}^m - \Lambda_{k,l})^2 .$$

Of the six slopes, we choose the one that generates a Λ^m with smallest error norm, $\|\Lambda^m - \Lambda\|$. The interface in cell (i, j) is then represented as the linear profile that produced the smallest error. Note that the intersection of this profile with cell (i, j) has the correct volume fraction $\Lambda_{i,j}$. It was shown in [52] that this approach leads to a discontinuous, but second-order accurate, linear reconstruction of the front in each cell.

Calculation of front velocities

To update Λ we first calculate the front velocity using (Eq. III), which gives a relationship between the front speed and $\mathbf{n} \cdot \nabla \varphi$. We can approximate the normal gradient at the front using $G_{i,j}^{f,n}$, given by (2.17), and extend the resulting velocities to grid edges, for use in the advection algorithm for $\Lambda_{i,j}$.

This is done in the following manner, which is similar to that used in [9]. We first introduce a marker, $P_{i,j}$, associated with each cell. If that cell contains a portion of the

front, and thus an estimate of the normal gradient $G_{i,j}^{f,n}$, we set $P_{i,j} = 1$, and $P_{i,j} = 0$, otherwise. We construct a vector velocity in that cell simply by combining the front normal and speed given by approximating (Eq. III) at time t^n ,

$$\mathbf{u}_{i,j} = -\text{St } \mathbf{n}_{i,j}^f F_{i,j}^{f,n}. \quad (4.28)$$

Each cell in $\text{ngh}(i,j)$ will need some approximation of the velocity vector for the advection algorithm. We can now extend the front velocity to neighboring cells using an average value,

$$\bar{\mathbf{u}}_{i,j} = \frac{\sum_k P_k \mathbf{u}_k}{\sum_k P_k}, \quad (4.29)$$

where the sums are over cells $k \in \text{ngh}(i,j)$. We now use these quantities to calculate edge velocities that will participate in the advection algorithm. If we have two values of $\mathbf{u}_{i,j}$ in the cells neighboring a given edge, then average them; with only one value of $\mathbf{u}_{i,j}$, that value is used at the edge. If neither cell has a value for $\mathbf{u}_{i,j}$, we then repeat the procedure with $\bar{\mathbf{u}}_{i,j}$. For example, if edge $(i + \frac{1}{2}, j)$ one or two neighbors with $P_{i,j} = 1$, we would use the formula,

$$u_{i+\frac{1}{2},j} = \frac{P_{i+1,j} u_{i+1,j} + P_{i,j} u_{i,j}}{P_{i+1,j} + P_{i,j}}. \quad (4.30)$$

If the edge has only neighbors with $P_{i,j} = 0$, we would define a new marker field with $P'_{i,j}$, such that $P'_{i,j} = 1$ if a front cell is in $\text{ngh}(i,j)$, otherwise $P'_{i,j} = 0$. We would then use the a similar formula to extend the $\bar{\mathbf{u}}_{i,j}$ to these cell edges,

$$u_{i+\frac{1}{2},j} = \frac{P'_{i+1,j} \bar{u}_{i+1,j} + P'_{i,j} \bar{u}_{i,j}}{P'_{i+1,j} + P'_{i,j}}. \quad (4.31)$$

This guarantees that all the edges participating in the advection algorithm have a defined velocity. We have attempted to depict this extension algorithm in Figure 4.25. Of course,

domain boundaries are treated differently, depending on the type of boundary condition enforced there. Dirichlet-type boundary conditions do not change the extension algorithm, while Neumann boundary conditions set the edge velocity there to zero.

Operator-split advection for volume fraction.

This approach also comes from [9]. We discretize (4.27) using an operator-split method, where the x - and y -components of the equation are integrated separately, that is

$$\Lambda_t + \partial_x(u\Lambda) = \Lambda \partial_x u$$

$$\Lambda_t + \partial_y(v\Lambda) = \Lambda \partial_y v$$

are integrated one at a time. Each equation is advanced with a finite volume discretization, using a control volume around the entire Cartesian cell (i, j) . For the x equation (“sweep”), this yields

$$\frac{\Lambda_{i,j}^{n+\frac{1}{2}} - \Lambda_{i,j}^n}{\Delta t/2} + \frac{F_{i+\frac{1}{2},j}^{\Lambda,n} - F_{i-\frac{1}{2},j}^{\Lambda,n}}{\Delta x \Delta y} = \Lambda_{i,j}^{n+\frac{1}{2}} \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x}, \quad (4.32)$$

where F^Λ approximates the flux of Λ through edge $(i + \frac{1}{2}, j)$, which will be explained below. Note that we have chosen to center the right-hand side at $t^{n+\frac{1}{2}}$. This implicit differencing guarantees that $\Lambda \in [0, 1]$, subject to the stability constraint

$$\text{Cr} = \max \left(\frac{|u|\Delta t}{\Delta x}, \frac{|v|\Delta t}{\Delta y} \right) \leq \frac{1}{2}, \quad (4.33)$$

where Cr is the Courant number in the current context. This can be interpreted as saying that the front may move no more than a half cell width in either direction, during any time

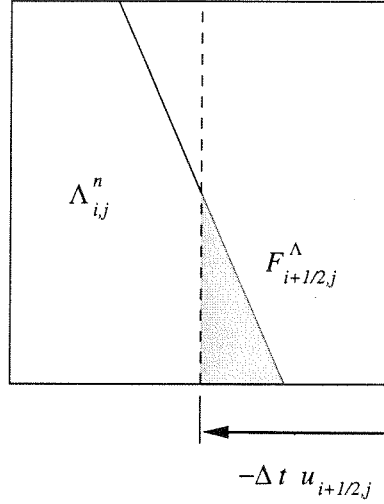


Figure 4.26: The volume fraction flux across edge $(i + \frac{1}{2}, j)$ is computed by shifting the edge by a distance $-\Delta t u_{i+\frac{1}{2},j}$. $F_{i+\frac{1}{2},j}^\Lambda$ is found by calculating the area of this shifted region (between the dashed line and edge $(i + \frac{1}{2}, j)$), intersected with the region $\Lambda_{i,j}^n$ given by the linear front reconstruction.

step. The y sweep is written in a similar manner:

$$\frac{\Lambda_{i,j}^{n+1} - \Lambda_{i,j}^{n+\frac{1}{2}}}{\Delta t/2} + \frac{F_{i,j+\frac{1}{2}}^{\Lambda,n+\frac{1}{2}} - F_{i,j-\frac{1}{2}}^{\Lambda,n+\frac{1}{2}}}{\Delta x \Delta y} = \Lambda_{i,j}^{n+1} \frac{u_{i,j+\frac{1}{2}} - u_{i,j-\frac{1}{2}}}{\Delta x}. \quad (4.34)$$

In order to obtain second-order accuracy in space with this approach, the x sweep and y sweep must be performed in an x - y , y - x pattern [59]. After each sweep the interface is reconstructed using the approximate least-squares algorithm, given above.

Calculation of edge fluxes.

We take a geometric approach to calculating the fluxes of Λ through each cell edge, $F_{i+\frac{1}{2},j}^\Lambda$, starting from a linear representation of the front in each cell, using the approximate reconstruction procedure. At each cell edge where F^Λ is required in the x -sweep, we first determine the “upwind” direction based on the sign of the normal component of \mathbf{u} on that edge. For instance, at the $(i + \frac{1}{2}, j)$ edge, we use the sign of $u_{i+\frac{1}{2},j}$ to choose which cell (i, j) or $(i + 1, j)$ the flux will come from. If $u_{i+\frac{1}{2},j} > 0$, the flux for edge $(i + \frac{1}{2}, j)$

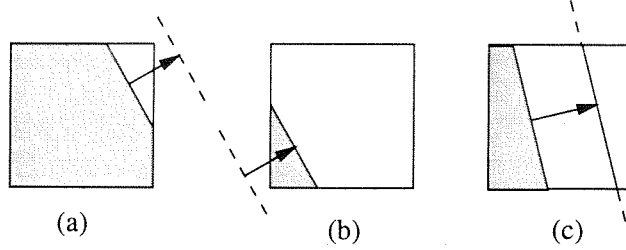


Figure 4.27: There are three cases when specifying the trajectory of the front in space-time. For (a) and (b) when only one of Λ^n or Λ^{n+1} is between 0 and 1, the front trajectory is specified by that Λ , \mathbf{n} , and \dot{s} . In case (c) when both $0 < \Lambda^n, \Lambda^{n+1} < 1$, the volume fractions and \mathbf{n} are used to specify the front location.

comes from cell (i, j) , and we define $F_{i+\frac{1}{2},j}^\Lambda$ to be the area of the intersection between $[x_{i+\frac{1}{2},j} - u_{i+\frac{1}{2},j} \Delta t, x_{i+\frac{1}{2},j}] \times [y_{i,j-\frac{1}{2}}, y_{i,j+\frac{1}{2}}]$ and the geometric description of Ω in cell (i, j) . This is depicted in Figure 4.26. If $u_{i+\frac{1}{2},j} < 0$, then $F_{i+\frac{1}{2},j}^\Lambda$ is computed from the area of the intersection of Ω in cell $(i+1, j)$, and $[x_{i+\frac{1}{2},j}, x_{i+\frac{1}{2},j} - u_{i+\frac{1}{2},j} \Delta t] \times [y_{i,j-\frac{1}{2}}, y_{i,j+\frac{1}{2}}]$. For the y -sweep, this procedure is applied to the $(i, j + \frac{1}{2})$ and $(i, j - \frac{1}{2})$ edges of each cell, in a similar fashion. Note again that the geometric representation of the interface is reconstructed after each x - or y -sweep.

Approximation of space-time areas and centroids

Given the volume fraction fields at the old and new times, $\Lambda_{i,j}^n$ and $\Lambda_{i,j}^{n+1}$, we can reconstruct the front's approximate trajectory in time in each cell (i, j) . This is done using an approach given in [18] and [9]. We have already assumed $\Lambda_{i,j}^{n+1} \geq \Lambda_{i,j}^n$, although the algorithm can be easily extended to include the other case [18]. Because the front is not allowed to move more than $\frac{\Delta x}{2}$ during a time step, we are guaranteed that cells with $\Lambda = 0$ or $\Lambda = 1$ at both times do not contain a portion of the front during that time step.

Since we are only dealing with cell (i, j) at this point, we will drop the subscripts for this part of the discussion. We first assume that the normal to the front, \mathbf{n} , is constant in time, and given by the value at t^n using the procedure described above. We define an approximate speed, \dot{s} , in one of three ways, each of which is depicted in Figure 4.27. First, if $\Lambda^{n+1} = 1$, then the front has left cell (i, j) during the time step. In this case, we set

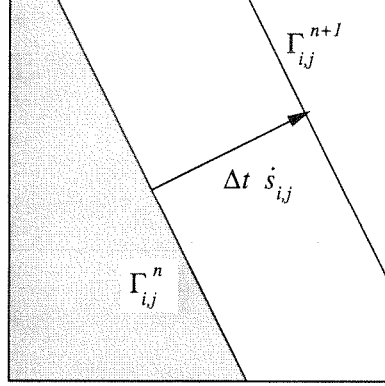


Figure 4.28: For the case when $0 < \Lambda_{i,j}^n, \Lambda_{i,j}^{n+1} < 1$, we calculate the front speed by first reconstructing the front at t^n and t^{n+1} , so that it has normal $\mathbf{n}_{i,j}$, and matches the volume fractions at each time. The quantity $\Delta t \dot{s}_{i,j}$ is given by the distance between these parallel lines, which determines $\dot{s}_{i,j}$

$\dot{s} = |\mathbf{u}|$, where \mathbf{u} is given by (4.28). However, because Λ^{n+1} is determined from an inexact advection algorithm, \dot{s} might not be large enough to guarantee that $\Lambda^{n+1} = 1$. In that case, we set \dot{s} to the minimum value that guarantees $\Lambda^{n+1} = 1$. Similarly, if $\Lambda^n = 0$, then we set $\dot{s} = |\bar{\mathbf{u}}|$, where $\bar{\mathbf{u}}$ is the extension of the front velocities. Again, if \dot{s} is not large enough, we use the minimum value that guarantees $\Lambda^n = 0$. Otherwise, $0 < \Lambda^n, \Lambda^{n+1} < 1$, and we must match the front speed with its old and new positions, determined by Λ^n, Λ^{n+1} , and \mathbf{n} . Specifically, at t^n and t^{n+1} , the normal \mathbf{n} is used to generate a linear representation of the front in each cell, whose areas at time t^n and t^{n+1} exactly match the volume fractions Λ^n and Λ^{n+1} respectively. Again, this is done using the formulae given in the first section and in [18]. The distance between the two parallel, linear representations at t^n and t^{n+1} approximates $\Delta t \dot{s}$, which in turn yields \dot{s} . Figure 4.28 depicts this reconstruction of the front speed from the Λ 's and \mathbf{n} .

Now that we have an approximation of \dot{s} , we can describe the space-time evolution of Ω in cell (i, j) with

$$\Gamma(\mathbf{x}, t) = \{ \mathbf{x} \in \text{cell}(i, j), t \in [t^n, t^{n+1}] : \mathbf{n} \cdot \mathbf{x} - \dot{s} t + C \leq 0 \} ,$$

where the constant C is chosen to match Λ^n and Λ^{n+1} . This polyhedron $\Gamma(\mathbf{x}, t)$ is depicted

in Figure 4.29. Note that it has at most seven faces, six of which coincide with the faces of the box defined by cell (i, j) in the time interval $[t^n, t^{n+1}]$, and the last being the trajectory of the front in space-time.

We will now define each of the space-time surfaces surrounding the volume $\Gamma(\mathbf{x}, t)$. The top surface is the plane $\Gamma(\mathbf{x}, t^{n+1})$, and defines the interior of the cell (i, j) at time t^{n+1} . This planar region is denoted by $V_{i,j}^{n+1}$, and has area $\Delta x \Delta y \Lambda_{i,j}^{n+1}$. Similarly, the bottom surface $\Gamma(\mathbf{x}, t^n)$ corresponds to $V_{i,j}^n$, with area $\Delta x \Delta y \Lambda_{i,j}^n$. These two constraints are used to determine the constant C in the equation for $\Gamma(\mathbf{x}, t)$ above; the construction of \dot{s} , above, guarantees that V^n and V^{n+1} match Λ^n and Λ^{n+1} , respectively. The vertical faces of $\Gamma(\mathbf{x}, t)$ correspond to an edge of cell (i, j) in space-time. For instance, we define the surface at $(i + \frac{1}{2}, j)$ with

$$S_{i+\frac{1}{2},j} = \Gamma\left(x_{i+\frac{1}{2},j}, y, t\right) .$$

The remaining edges surrounding cell (i, j) lead to similar definitions of the surfaces $S_{i-\frac{1}{2},j}$, $S_{i,j+\frac{1}{2}}$, and $S_{i,j-\frac{1}{2}}$. We denote the final boundary of $\Gamma(\mathbf{x}, t)$ with S^f , as it refers to the surface defined by the front's trajectory in space-time. Because the speed and normal are constant, S^f is simply the plane in space-time defined by

$$S^f = \left\{ \mathbf{x} \in \text{cell}(i, j), t \in [t^n, t^{n+1}] : \mathbf{n} \cdot \mathbf{x} - \dot{s} t + C = 0 \right\} .$$

In addition, the space-time normal to S^f is also constant in time, and given by

$$\vec{\nu}^f = \frac{(n_x, n_y, -\dot{s})}{\sqrt{1 + \dot{s}^2}} ,$$

where the denominator makes $\vec{\nu}^f$ a unit normal vector.

The surfaces of the space-time volume $\Gamma(\mathbf{x}, t)$ will now be used to determine time-averaged geometric properties within cell (i, j) . To simplify our notation, we will use the

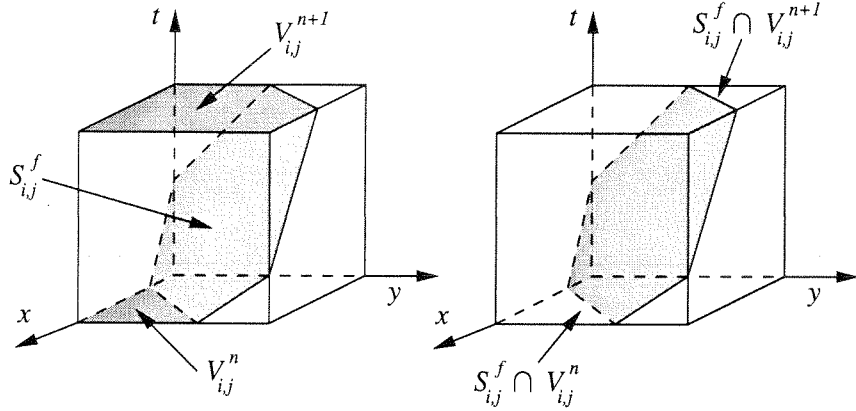


Figure 4.29: Here we describe the surfaces of the space-time volume, $\Gamma(\mathbf{x}, t)$. The plane S^f represents the trajectory of the front in space-time through the cell $(i, j) \times [t^n, t^{n+1}]$. Where S^f the front location at each time is given by the intersection of S^f with the planes $t = t^n$ and $t = t^{n+1}$.

variables defining planar regions, such as S^f and V^n , to also represent the areas of those regions, wherever it is unambiguous. By applying the divergence theorem to a constant vector \mathbf{c} , on the volume defined by $\Gamma(\mathbf{x}, t)$ we obtain

$$\iiint_{\Gamma} \nabla \cdot \mathbf{c} \, dV = \iint_{\partial \Gamma} \mathbf{c} \cdot \vec{\nu} \, dS, \quad (4.35)$$

where $\vec{\nu}$ represents the unit outward normal for each surface of $\partial \Gamma$. By considering each component of the vector, \mathbf{c} , we can determine relationships between the areas of each of these surfaces. For instance, setting \mathbf{c} equal to the unit vectors in x , y , and t , respectively, yields

$$\begin{aligned} S_{i+\frac{1}{2},j} - S_{i-\frac{1}{2},j} &= -\frac{n_x}{\sqrt{1+\dot{s}^2}} S^f \\ S_{i,j+\frac{1}{2}} - S_{i,j-\frac{1}{2}} &= -\frac{n_y}{\sqrt{1+\dot{s}^2}} S^f \\ V_{i,j}^{n+1} - V_{i,j}^n &= \frac{\dot{s}}{\sqrt{1+\dot{s}^2}} S^f. \end{aligned} \quad (4.36)$$

We are mainly interested in the role of these surface areas in our control-volume discretization. In two dimensions, we need the time integrals of edge lengths, over the interval

$[t^n, t^{n+1}]$, which we will call A . On the cell edges, $(i + \frac{1}{2}, j)$, etc., these are just the space-time areas, so $A^{i+\frac{1}{2},j} = S_{i+\frac{1}{2},j}$. We obtain the necessary result at the front by projecting S^f onto the normal direction in space, $(n_x, n_y, 0)$, to yield

$$\begin{aligned} A^f &= (n_x, n_y, 0) \cdot \vec{\nu}^f S^f \\ &= \frac{1}{\sqrt{1 + \dot{s}^2}} S^f . \end{aligned}$$

The relationship (4.36) can then be written as

$$\begin{aligned} A^{i-\frac{1}{2},j} - A^{i+\frac{1}{2},j} &= n_x A^f \\ A^{i,j-\frac{1}{2}} - A^{i,j+\frac{1}{2}} &= n_y A^f \\ V_{i,j}^{n+1} - V_{i,j}^n &= \dot{s} A^f . \end{aligned} \tag{4.37}$$

This set of equations will be necessary to obtain quadrature cancellations in the next section.

Our final derivation is for the space-time centroid of each of the surfaces; this is needed to apply the midpoint rule to flux integrals in two dimensions. In order to obtain an expression such as (4.37), we consider (4.35) with vector $\mathbf{c} = (0, x, 0)$, for example. This leads to

$$\begin{aligned} \iint_{\partial\Gamma} \mathbf{c} \cdot \vec{\nu} dS &= \iint_{S_{i,j+\frac{1}{2}}} x dS - \iint_{S_{i,j-\frac{1}{2}}} x dS + \nu_y^f \iint_{S^f} x dS \\ &= \bar{x}^{i,j+\frac{1}{2}} A^{i,j+\frac{1}{2}} - \bar{x}^{i,j-\frac{1}{2}} A^{i,j-\frac{1}{2}} + \nu_y^f \bar{x}^f A^f \\ &= 0 , \end{aligned}$$

where \bar{x} indicates the x coordinate of the centroid for each face. We obtain a similar result with $\mathbf{c} = (y, 0, 0)$. These centroid are also related to the spatial centroid of $V_{i,j}$ at t^n and

t^{n+1} ; with $\mathbf{c} = (0, 0, x)$ we obtain

$$\bar{x}_{i,j}^{n+1} V_{i,j}^{n+1} - \bar{x}_{i,j}^n V_{i,j}^n - \bar{x}_{i,j}^f \dot{s} A^f = 0 .$$

Another equation can be derived when $\mathbf{c} = (0, 0, y)$. Finally, the centroid of each face in time is given by $\mathbf{c} = (t, 0, 0)$,

$$\bar{t}^{i+\frac{1}{2},j} A^{i+\frac{1}{2},j} - \bar{t}^{i-\frac{1}{2},j} A^{i-\frac{1}{2},j} + \bar{t}_{i,j}^f n_x^f A^f = 0 .$$

In summary we have the following six relationships between the space-time centroids on each surface of $\Gamma(\mathbf{x}, t)$:

$$\bar{x}_{i,j}^f \dot{s} A^f = \bar{x}_{i,j}^{n+1} V_{i,j}^{n+1} - \bar{x}_{i,j}^n V_{i,j}^n \quad (4.38)$$

$$n_y \bar{x}^f A^f = \bar{x}^{i,j-\frac{1}{2}} A^{i,j-\frac{1}{2}} - \bar{x}^{i,j+\frac{1}{2}} A^{i,j+\frac{1}{2}} \quad (4.39)$$

$$n_x \bar{y}^f A^f = \bar{y}^{i-\frac{1}{2},j} A^{i-\frac{1}{2},j} - \bar{y}^{i+\frac{1}{2},j} A^{i+\frac{1}{2},j} \quad (4.40)$$

$$n_y \bar{t}^f A^f = \bar{t}^{i,j-\frac{1}{2}} A^{i,j-\frac{1}{2}} - \bar{t}^{i,j+\frac{1}{2}} A^{i,j+\frac{1}{2}} \quad (4.41)$$

$$n_x \bar{t}^f A^f = \bar{t}^{i-\frac{1}{2},j} A^{i-\frac{1}{2},j} - \bar{t}^{i+\frac{1}{2},j} A^{i+\frac{1}{2},j} . \quad (4.42)$$

Averaging of space-time areas and centroids

We have developed these approximations using only $\dot{s}_{i,j}$ and $\Lambda_{i,j}$. However, edge-centered quantities, such as $A_{i,j}^{i+\frac{1}{2},j}$, have also been determined in the neighboring cell $(i+1, j)$. Thus we have a mismatch of these quantities between cells; because the apertures have an essential role in our conservative discretizations, this is clearly unacceptable. To remedy this, we average the two quantities derived from data in different cells, wherever possible. For instance, if both cells (i, j) and $(i+1, j)$ generated approximations to $A^{i+\frac{1}{2},j}$,

we define the average value

$$A_{i+\frac{1}{2},j} = \frac{1}{2} \left(A_{i,j}^{i+\frac{1}{2},j} + A_{i+1,j}^{i+\frac{1}{2},j} \right).$$

For centroid values, this is done with area-weighted averages,

$$\bar{t}_{i+\frac{1}{2},j} = \frac{\bar{t}_{i,j}^{i+\frac{1}{2},j} A_{i,j}^{i+\frac{1}{2},j} + \bar{t}_{i+1,j}^{i+\frac{1}{2},j} A_{i+1,j}^{i+\frac{1}{2},j}}{A_{i,j}^{i+\frac{1}{2},j} + A_{i+1,j}^{i+\frac{1}{2},j}},$$

with similar definitions for $\bar{\mathbf{x}}$ on edges.

If only one of the two cells bordering an edge has an approximation of the moving front, then we ignore the moving front data there. For example, if there is a mismatch between the linear reconstruction of the interface in cell (i, j) , and the neighboring cell with $\Lambda_{i+1,j} = 0$, we assume that $A_{i+\frac{1}{2},j} = 0$, and no flux passes through that edge. Similarly, if cell $(i-1, j)$ has $\Lambda_{i-1,j} = 1$, then we set $A_{i+\frac{1}{2},j} = \Delta y$, and treat that edge as a full edge. This is done to prevent any “contact resistance,” represented by partial apertures, between the interior of the domain and cells that contain a portion of the front.

In the next section, we will show that the relations

$$A^{i-\frac{1}{2},j} - A^{i+\frac{1}{2},j} = n_x A^f \tag{4.43}$$

$$A^{i,j-\frac{1}{2}} - A^{i,j+\frac{1}{2}} = n_y A^f$$

are critical in the derivation of our finite-volume discretization for (Eq. II). However, we expect that the averaged edge values of A could satisfy (4.43) to first-order, at best. To rectify this, we recalculate $\mathbf{n}_{i,j}$ and $A_{i,j}^f$ using (4.43) and the four averaged edge apertures, so that the equations are satisfied exactly. Other quantities, such as the centroids \bar{t} and $\bar{\mathbf{x}}$ and front speed \dot{s} , do not play as significant a role in the finite-volume derivation, and are not recalculated.

4.4.2 Accuracy Assessment

The first thing to note about this front tracking algorithm, is its ability to exactly track a planar front moving with constant velocity. We see this through the following line of reasoning. Regardless of the orientation of the line representing $\partial\Gamma$, the approximate least-squares algorithm is able to reconstruct it exactly, given only the correct volume fraction array. The operator-split advection algorithm will calculate the correct volume fraction fluxes, F^Λ , given the exact normal and constant speed. Finally, the space-time area and centroid calculations are exact; the fundamental assumption in calculating them is that the front is a plane in space-time, which is certainly the case here. Thus, no averaging or truncating of edge quantities is necessary. Every aspect of the algorithm is exact, so that the planar front with constant velocity is reproduced with only rounding errors.

In [52], it was shown for a variety of simple test problems, based on translating and rotating simple geometric shapes with constant velocity, that the combination of the approximate least-squares reconstruction and operator-split advection scheme is second-order accurate in space. With our simple algorithm for extending the velocity field, we expect only first-order accuracy for Λ . In addition, the averaged edge-centered quantities no longer satisfy (4.39) – (4.42), but we expect the approximation to be first-order accurate.

4.5 Discretization in Two Dimensions

Because the method in one dimension is based on a finite-volume-type discretization, it is easily extended to two or more dimensions. We will assume that we have the geometric properties of the moving front, such as volumes, time apertures, and time-centroids. For the purpose of the discussion in this section, we will use A_k and A_k^f to designate the *average* lengths of each of the sides of $V_{i,j}(t)$, from t^n to t^{n+1} . Thus, the A_k here are merely the time-apertures given in the last section, divided by Δt .

We then integrate (Eq. II) over this space-time control volume, and use carefully-

chosen quadrature rules for the resulting integrals, to obtain a consistent discretization. The spatial integrals are approximated using values at the cell centroids at the old and new times, whereas the time integrals of edge or interface fluxes are calculated using finite-difference stencils for the gradients. In view of the instability of the conservative algorithm in one dimension, we will focus on assessing the accuracy of the backward Euler and hybrid methods in two dimensions, without maintaining conservation with the lagged source term.

4.5.1 Finite volume derivation in 2D

We start from (Eq. II) in two dimensions, and integrate it from t^n to t^{n+1} over cell (i, j) . For cells that do not interact with the front during the time step, φ is governed by the heat equation, and the derivation given in Chapter 3 applies. In cells that contain the front, we will again consider the two cases: one where the front intersects the given cell (i, j) at both the beginning and end of the time step ($0 < \Lambda^n, \Lambda^{n+1} < 1$), and the other where it does not.

In order to assess the truncation error of the discretization, we first start by integrating (Eq. II) over the *exact* control volume, whose quantities we designate with a tilde ($\tilde{\cdot}$), and corresponds to the motion of the true front through cell (i, j) :

$$\begin{aligned} \int_{\tilde{V}_{i,j}^{n+1}} \varphi(\mathbf{x}', t^{n+1}) d\mathbf{x}' - \int_{\tilde{V}_{i,j}^n} \varphi(\mathbf{x}', t^n) d\mathbf{x}' \\ - \int_{t^n}^{t^{n+1}} \int_{\tilde{A}_{i,j}^f(t')} \varphi(\mathbf{x}', t') \mathbf{u} \cdot \mathbf{n}^f dx' dt' = \int_{t^n}^{t^{n+1}} \int_{\partial V_{i,j}(t')} \beta \nabla \varphi \cdot \mathbf{n} d\tilde{A} dt' . \end{aligned}$$

First, we note that the integral over \tilde{A}^f is zero, because of the Dirichlet boundary condition for φ . This can be simplified further by writing the last integral as a sum over all the sides of $\partial V_{i,j}(t')$:

$$\int_{\tilde{V}_{i,j}^{n+1}} \varphi(\mathbf{x}', t^{n+1}) d\mathbf{x}' - \int_{\tilde{V}_{i,j}^n} \varphi(\mathbf{x}', t^n) d\mathbf{x}' = \sum_k \int_{t^n}^{t^{n+1}} \int_{A_k} \beta \nabla \varphi \cdot \mathbf{n}_k d\tilde{A}_k dt' .$$

Here, \tilde{A}_k represents each of the five possible surfaces of $\tilde{V}_{i,j}$: \tilde{A} with half indices are for cell edges, while the front is designated by $\tilde{A}_{i,j}^f$.

Similarly, by integrating (Eq. III) in time over \tilde{A}^f , we also obtain an expression similar to (4.4),

$$\tilde{V}_{i,j}^{n+1} - \tilde{V}_{i,j}^n = -\text{St} \int_{t^n}^{t^{n+1}} \int_{\tilde{A}_{i,j}^f} \beta \nabla \varphi \cdot \mathbf{n}^f d\tilde{A}^f dt' . \quad (4.44)$$

Again, this represents conservation of energy, because the melting volume absorbs heat from the interior of the domain.

Cancellation of quadrature errors

In cells that contain a portion of the front during the timestep, we use the geometric properties of the approximation to the space-time volume from the previous section. To reiterate, we have a relationship between the time apertures and front normal, given by (4.37), which we will use to cancel the quadrature errors for the time integrals, to one higher order than expected.

Quadrature error for spatial integrals. We will again use the midpoint rule for the integrals over $\tilde{V}_{i,j}$ in (4.5.1). Given the intersections of the front with grid lines at time t^n , we have a second-order accurate approximation of $V_{i,j}^n$, for the purpose of calculating the truncation error. This implies that

$$\int_{\tilde{V}_{i,j}^n} \varphi(\mathbf{x}', t^n) d\mathbf{x}' = V_{i,j}^n \varphi(\bar{\mathbf{x}}_{i,j}^n, t^n) + O(\Delta x^4) , \quad (4.45)$$

where $\bar{\mathbf{x}}_{i,j}^n$ is the location of the centroid of cell (i, j) at time t^n . We have also assumed that $\Delta x = r\Delta y$ for some constant r . A similar result holds at t^{n+1} .

Quadrature errors for time integrals. The time-integral of fluxes in the right-hand side of (4.5.1) is approximated to second-order accuracy using the space-time centroids on each of the faces of $\tilde{V}_{i,j}$:

$$\sum_k \int_{t^n}^{t^{n+1}} \int_{\tilde{A}_k} \nabla \varphi \cdot \mathbf{n}_k d\tilde{A}_k dt' = \Delta t \sum_k \tilde{A}_k \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, \bar{t}_k) + O(\Delta x \Delta t^3) .$$

The first step in our derivation is to evaluate the fluxes at the correct location in space, but at the new time t^{n+1} . By performing a Taylor expansion of $\nabla \varphi$ about t^{n+1} , the sum on the right side becomes

$$\begin{aligned} \Delta t \sum_k \tilde{A}_k \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, \bar{t}_k) &= \Delta t \sum_k \tilde{A}_k \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, t^{n+1}) \\ &\quad + \Delta t \sum_k \tilde{A}_k (\bar{t}_k - t^{n+1}) \mathbf{n}_k \cdot \nabla \varphi_t(\bar{\mathbf{x}}_k, t^{n+1}) + O(\Delta x \Delta t^3) . \end{aligned}$$

Note that using the proper time centroid \bar{t}_k on each surface would remove the first error term. However, we can approximate the error term by shifting the quadrature points in space to $\mathbf{x}_{i,j}$,

$$\nabla \varphi_t(\bar{\mathbf{x}}_k, t^{n+1}) = \nabla \varphi_t(\mathbf{x}_{i,j}, t^{n+1}) + O(\Delta x) ,$$

which we will call \mathbf{c} from here on. The quadrature error then becomes

$$\Delta t \sum_k \tilde{A}_k (\bar{t}_k - t^{n+1}) \mathbf{n}_k \cdot \nabla \varphi_t(\bar{\mathbf{x}}_k, t^{n+1}) = \Delta t \sum_k \tilde{A}_k (\bar{t}_k - t^{n+1}) \mathbf{n}_k \cdot \mathbf{c} + O(\Delta x^2 \Delta t^2) .$$

Now we note that the sum on the right-hand side is a second-order accurate approximation of the divergence theorem, applied to $\mathbf{c}(t - t^{n+1})$:

$$\Delta t \sum_k \tilde{A}_k (\bar{t}_k - t^{n+1}) \mathbf{n}_k \cdot \mathbf{c} = \int_{t^n}^{t^{n+1}} \int_{\tilde{V}} \nabla \cdot (\mathbf{c}(t - t^{n+1})) d\tilde{V} dt' + O(\Delta x^2 \Delta t^2).$$

Since the spatial divergence of a function of time is zero, the error term is $O(\Delta x^2 \Delta t^2)$. Thus, shifting the time-centering of $\nabla \varphi$ to t^{n+1} does not change the truncation error of this sum.

However, we would like to use the A_k calculated from the front advection algorithm, which are first-order accurate. By substituting A_k for \tilde{A}_k we obtain

$$\begin{aligned} \sum_k A_k \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, t^{n+1}) &= \sum_k \tilde{A}_k \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, t^{n+1}) \\ &\quad + \sum_k (A_k - \tilde{A}_k) \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, t^{n+1}). \end{aligned}$$

The gradient in this last sum can be expanded around $\mathbf{x}_{i,j}$, so that the error term becomes

$$\sum_k (A_k - \tilde{A}_k) \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, t^{n+1}) = \nabla \varphi(\mathbf{x}_{i,j}, t^{n+1}) \cdot \sum_k \mathbf{n}_k (A_k - \tilde{A}_k) + O(\Delta x^3 \Delta t^2),$$

because the quantity $A_k - \tilde{A}_k$ is $O(\Delta x^2 \Delta t)$ for first-order accurate A_k . Finally, we note that the sum on the right-hand side is exactly zero: for \tilde{A}_k this is always true, but A_k has been constructed to be zero as well. Thus, the error term is zero, and we have shown that

$$\sum_k \int_{t^n}^{t^{n+1}} \int_{\tilde{A}_k} \nabla \varphi \cdot \mathbf{n}_k d\tilde{A}_k dt' = \Delta t \sum_k A_k \mathbf{n}_k \cdot \nabla \varphi(\bar{\mathbf{x}}_k, t^{n+1}) + O(\Delta x \Delta t^3). \quad (4.46)$$

Dividing this error by $\Delta x \Delta t^2$ yields $O(\Delta t)$, and we can conclude that evaluating fluxes at the spatial location of the space-time centroids, but t^{n+1} , still yields a first-order accurate discretization.

Approximation of volume integrals

The volume integral in (4.45) is best approximated by φ at the centroid of the cell, $\bar{\mathbf{x}}_{i,j}$, and not the cell center $\mathbf{x}_{i,j}$. Therefore, we will use our cell-centered values of ϕ to estimate a value at the centroid location, as in one dimension. Let us define a bilinear interpolation operator, M , which depends on the location of the centroid $\bar{\mathbf{x}}_{i,j}$. For example, if $\bar{x}_{i,j} < x_{i,j}$, and $\bar{y}_{i,j} < y_{i,j}$, this would imply that the four-point stencil should be

$$\begin{aligned} (M\phi)_{i,j}^n &= (1 + \delta_1)(1 + \delta_2)\phi_{i,j}^n - \delta_1(1 + \delta_2)\phi_{i-1,j}^n \\ &\quad - \delta_2(1 + \delta_1)\phi_{i,j-1}^n - \delta_1\delta_2\phi_{i-1,j-1}^n, \end{aligned} \quad (4.47)$$

where $(\Delta x \delta_1, \Delta y \delta_2) = \bar{\mathbf{x}}_{i,j} - \mathbf{x}_{i,j}$. Similar formulas are easily defined for the cases that $\bar{\mathbf{x}}_{i,j}$ is in any of the other quadrants of cell (i, j) . Regardless, this is a spatially second-order accurate interpolation of ϕ at $\bar{\mathbf{x}}_{i,j}$.

Calculation of second-order accurate gradients

We now need to approximate the normal gradients in (4.46) at the centroids $\bar{\mathbf{x}}_k$, to second-order accuracy. Our approach will use ideas from both Chapter 2, where gradients were interpolated along edges to achieve the proper centering, and from the discretization in one dimension. There are three cases, each needing separate consideration.

Gradients on cell edges. For each cell edge, gradients are evaluated at the spatial location of the edge's space-time centroid. This requires an interpolated gradient as was used in Chapter 2; for instance, at $(i + \frac{1}{2}, j)$ we would use

$$G_{i+\frac{1}{2},j}^{I,n+1} = (1 - \eta) \frac{\phi_{i+1,j}^{n+1} - \phi_{i,j}^{n+1}}{\Delta x} + \eta \frac{\phi_{i+1,j+1}^{n+1} - \phi_{i,j+1}^{n+1}}{\Delta x}, \quad (4.48)$$

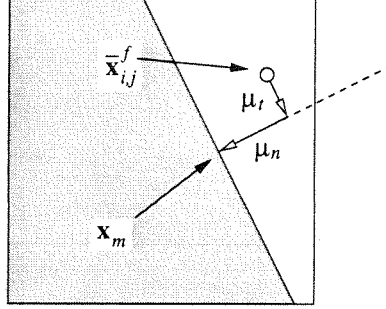


Figure 4.30: A gradient in the normal direction is calculated at $\bar{\mathbf{x}}_{i,j}^f$, marked with an open circle, using a Taylor series expansion about the front midpoint, \mathbf{x}_m . Instead of performing this in the (x, y) coordinate system, we use the system defined by the cell's outward-pointing normal, and tangent, as one traverses the boundary in the counter-clockwise direction. We then calculate the location of $\bar{\mathbf{x}}_{i,j}^f$ relative to \mathbf{x}_m in the new system, (μ_n, μ_t) .

for the case that $\bar{y}_{i+\frac{1}{2},j} > y_{i,j}$, with

$$\eta = \frac{\bar{y}_{i+\frac{1}{2},j} - y_{i,j}}{\Delta y}.$$

If $\bar{y}_{i+\frac{1}{2},j} < y_{i,j}$, of course, we use a gradient from $(i + \frac{1}{2}, j - 1)$ to complete the interpolation:

$$G_{i+\frac{1}{2},j}^{I,n+1} = (1 + \eta) \frac{\phi_{i+1,j}^{n+1} - \phi_{i,j}^{n+1}}{\Delta x} - \eta \frac{\phi_{i+1,j-1}^{n+1} - \phi_{i,j-1}^{n+1}}{\Delta x}.$$

We can define similar formulas for y edges using the spatial location of their space-time centroids. In either case, the formulas are spatially second-order accurate approximations of the normal gradient.

Gradients when front is present. Referring back to (4.9), for the algorithm in one dimension, we used a quadratic polynomial in space to estimate a gradient at the front centroid, \bar{s}_N . We will take the same approach here, but in two dimensions the algorithm is a bit more complicated; Figure 4.30 demonstrates why this is the case. In this discussion, we will only be concerned with the normal $\mathbf{n}_{i,j}^f$, so we will drop both the subscript (i, j) and superscript f .

First, we must describe the geometric construction of the second-order accurate gradient at the front. In Chapter 2, we were concerned with approximating a gradient at the midpoint of the interface, $G_{i,j}^f$, using a line normal to the interface extending from the midpoint. Now, we wish to approximate the normal gradient at the point $\bar{\mathbf{x}}_{i,j}^f$, using a Taylor series expansion from the midpoint, \mathbf{x}_m . Instead of performing this in (x, y) coordinates, we will use (n, t) , the normal and tangential directions:

$$\varphi_n|_{\bar{\mathbf{x}}^f} = \varphi_n|_{\mathbf{x}_m} + \mathbf{n} \cdot (\bar{\mathbf{x}}^f - \mathbf{x}_m) \varphi_{nn}|_{\mathbf{x}_m} + \mathbf{t} \cdot (\bar{\mathbf{x}}^f - \mathbf{x}_m) \varphi_{nt}|_{\mathbf{x}_m} + O(\Delta x^2). \quad (4.49)$$

This suggests how we can modify the formula for G^f used in the previous chapters, to calculate a flux at $\bar{\mathbf{x}}^f$. The interpolation line is defined by the line in the \mathbf{n}^f direction, intersecting the front in cell (i, j) at \mathbf{x}_m (Figure 4.30).

The next term in (4.30) involves the second derivative in the normal direction, φ_{nn} . The quadratic interpolant between Φ^f , ϕ_1^I , and ϕ_2^I , can be used to provide a first-order accurate approximation of φ_{nn} at \mathbf{x}_m (Figure (4.31)):

$$\phi_{nn} = \frac{2}{d_2 - d_1} \left(\left(\phi_2^{n+1} - \Phi^f \right) \frac{1}{d_2} - \left(\phi_1^{n+1} - \Phi^f \right) \frac{1}{d_1} \right). \quad (4.50)$$

In one dimension, we needed the distance $\bar{s}_N - s^{n+1}$ in (4.9); in two dimensions the corresponding distance is $\mu_n = \mathbf{n} \cdot (\bar{\mathbf{x}}^f - \mathbf{x}_m)$, from the projection of $\bar{\mathbf{x}}^f$ onto the interpolation line, to the midpoint of the front at t^{n+1} .

The next term in (4.49) involves the cross derivative φ_{nt} , and the distance from $\bar{\mathbf{x}}_{i,j}^f$ to the interpolation line, $\mu_t = \mathbf{t} \cdot (\bar{\mathbf{x}}^f - \mathbf{x}_m)$. In order to calculate φ_{nt} , we apply the chain rule to $\varphi_n = \mathbf{n} \cdot \nabla \varphi$, to obtain

$$\varphi_{nt} = -n_x n_y (\varphi_{xx} - \varphi_{yy}) + (n_x^2 - n_y^2) \varphi_{xy}. \quad (4.51)$$

Figure 4.31 demonstrates the finite-difference stencil that are used to approximate these

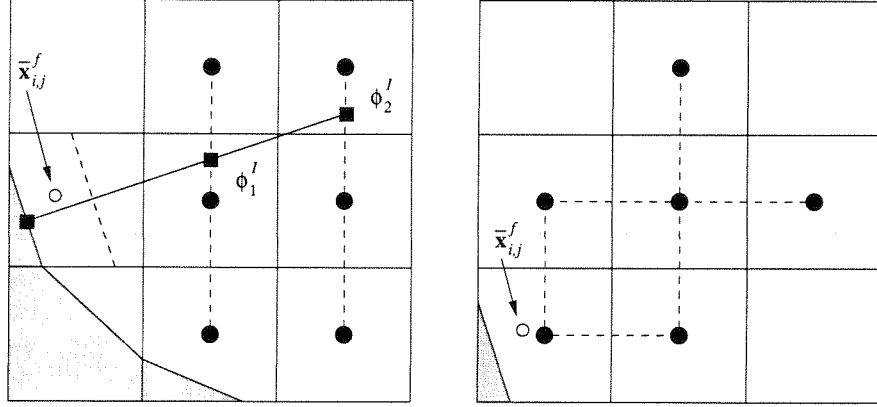


Figure 4.31: When the front is still present in cell (i, j) , we calculate φ_n at $\bar{\mathbf{x}}_{i,j}^f$ starting with the approximation for φ_n at \mathbf{x}_m (left). We again use the same stencil for G^f , given in (2.17), to approximate both φ_n and φ_{nn} from the value at \mathbf{x}_m , and two values interpolated in the interior, ϕ_1^I and ϕ_2^I . For the remaining cross term, φ_{nt} , we use a six-point stencil that is shifted away from the boundary (right).

second derivatives of φ . To guarantee that we do not reach past the domain boundary, we have shifted the stencil into the interior, away from the interface. For instance, if \mathbf{n} is pointing into the third quadrant, we use:

$$\begin{aligned}\phi_{xx} &= \frac{\phi_{i+2,j+1} - 2\phi_{i+1,j+1} + \phi_{i,j+1}}{\Delta x^2} \\ \phi_{yy} &= \frac{\phi_{i+1,j+2} - 2\phi_{i+1,j+1} + \phi_{i+1,j}}{\Delta y^2} \\ \phi_{xy} &= \frac{\phi_{i+1,j+1} - \phi_{i+1,j} - \phi_{i,j+1} + \phi_{i,j}}{\Delta x \Delta y},\end{aligned}$$

with similar formulas for \mathbf{n} pointing in other directions. To complete the first-order approximation of φ_{nt} , we substitute these difference formulae and \mathbf{n} into (4.51), and call the result ϕ_{nt} .

Finally, the gradient at $\bar{\mathbf{x}}_{i,j}^f$ is approximated using

$$G_{i,j}^{f,n+1} = \frac{1}{d_2 - d_1} \left(\phi_2^{n+1} \frac{d_1}{d_2} - \phi_1^{n+1} \frac{d_2}{d_1} \right) + \mu_n \phi_{nn} + \mu_t \phi_{nt}. \quad (4.52)$$

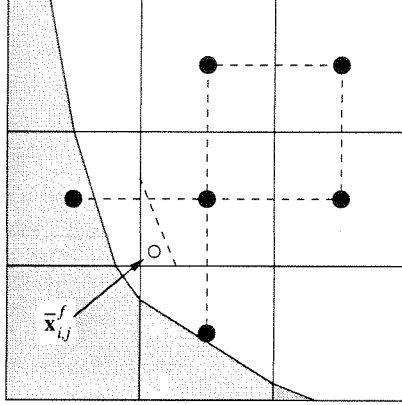


Figure 4.32: When the front has left cell (i, j) , we calculate an approximation to φ_n at $\bar{\mathbf{x}}_{i,j}^f$ using a six-point stencil, but no information about the location of the boundary at the new time.

Gradients when front has exited the cell. Again, we will use the algorithm in one dimension as a starting point for an appropriate discretization for the gradient at the front centroid, $\bar{\mathbf{x}}_{i,j}^f$. In (4.11), we fit a quadratic polynomial to three points, and then evaluated its slope at \bar{s}_N to approximate a gradient. Note that this did not use any information about the location of the front, or the Dirichlet boundary condition. In two dimensions, a second-order accurate gradient will require six cell-centered values. Note however, that $\Lambda_{i,j}^{n+1} = 1$, so that we can use any values in $\text{ngh}(i, j)$ without concern that the resulting finite difference stencil will reach outside the domain.

Figure 4.32 demonstrates the scenario that we are considering. A Taylor series expansion about $\mathbf{x}_{i,j}$ yields the following terms:

$$\varphi_x(\bar{\mathbf{x}}_{i,j}) = \varphi_x(\mathbf{x}_{i,j}) + (\bar{x}_{i,j} - x_{i,j})\varphi_{xx}(\mathbf{x}_{i,j}) + (\bar{y}_{i,j} - y_{i,j})\varphi_{xy}(\mathbf{x}_{i,j}) + O(\Delta x^2) \quad (4.53)$$

$$\varphi_y(\bar{\mathbf{x}}_{i,j}) = \varphi_y(\mathbf{x}_{i,j}) + (\bar{x}_{i,j} - x_{i,j})\varphi_{xy}(\mathbf{x}_{i,j}) + (\bar{y}_{i,j} - y_{i,j})\varphi_{yy}(\mathbf{x}_{i,j}) + O(\Delta x^2) .$$

We use the obvious finite difference stencils to approximate these derivatives at $\mathbf{x}_{i,j}$:

$$\begin{aligned}\phi_x &= \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \\ \phi_y &= \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \\ \phi_{xx} &= \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} \\ \phi_{yy} &= \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2} .\end{aligned}$$

For the ϕ_{xy} term, we use a four-point stencil reaching into the interior, away from the interface. For instance, if \mathbf{n} is pointing into the third quadrant (Figure 4.32), we use

$$\phi_{xy}(\mathbf{x}_{i,j}) = \frac{\phi_{i+1,j+1} - \phi_{i+1,j} - \phi_{i,j+1} + \phi_{i,j}}{\Delta x \Delta y} ,$$

with similar formulas for $\mathbf{n}_{i,j}^f$ pointing in other directions. Finally, all of the finite difference approximations are applied to ϕ^{n+1} , and substituted into the Taylor series expansion (4.53), to obtain a stencil for $G_{i,j}^{I,n+1}$. For the case where \mathbf{n} is pointing into the third quadrant, we have

$$G_{i,j}^I = n^x (\phi_x + \gamma_1 \phi_{xx} + \gamma_2 \phi_{xy}) + n^y (\phi_y + \gamma_2 \phi_{yy} + \gamma_1 \phi_{xy}) \quad , \quad (4.54)$$

where $(\Delta x \gamma_1, \Delta y \gamma_2) = \bar{\mathbf{x}}_{i,j} - \mathbf{x}_{i,j}$. For $\mathbf{n}_{i,j}^f$ in other quadrants, the appropriate finite difference stencil for ϕ_{xy} is substituted in (4.54), and the other terms remain the same. Note that we have chosen to use the superscript I for both interpolated gradients in (4.48) and (4.54); the subscripts of a cell edge $(i + \frac{1}{2}, j)$ or cell interior (i, j) differentiate the two formulas.

4.5.2 Summary of finite volume discretization

At this point, we have defined all of the operators necessary to complete the finite-volume discretization of (Eq. III) and (4.28). Again, we separate our discussions for cells in the interior, and cells that interact with the front.

Interior of the domain

The only choice that need be made in the interior of the domain, away from the front, is whether to use a backward Euler or a Crank-Nicholson discretization, corresponding to our “hybrid” algorithm, in time. Thus, when a cell does not contain a portion of the front, the update equation for ϕ^{n+1} is given by one of these formulas:

$$\phi_{i,j}^{n+1} - \Delta t (L\phi^{n+1})_{i,j} = \phi_{i,j}^n \quad (4.55)$$

$$\phi_{i,j}^{n+1} - \frac{\Delta t}{2} (L\phi^{n+1})_{i,j} = \phi_{i,j}^n + \frac{\Delta t}{2} (L\phi^n)_{i,j} . \quad (4.56)$$

These are the same discretizations of the heat equation that were used in Chapter 3, for the interior of the domain. We have demonstrated that these formulas are first- and second-order in time, respectively.

Cells that interact with the front

In this case, the front was in cell (i, j) at some point during the time step. This implies that $0 < \Lambda_{i,j}^n < 1$ and/or $0 < \Lambda_{i,j}^{n+1} < 1$ is true. Therefore, we will need to approximate each of the integrals in (4.5.1), using the finite-difference stencils given above. First, the volume integrals are approximated using the bilinear interpolation operator, M , defined in (4.47). Second, the edges of the cell each have space-time areas and centroids, which were derived from the front in the previous section, and resulted in equations (4.37), (4.39), and (4.40). For these edges, we must use the interpolated edge gradient, defined in

(4.48) at $(i + \frac{1}{2}, j)$ for instance, to approximate the flux in (4.5.1):

$$F_{i+\frac{1}{2},j}^{n+1} = \beta_{i+\frac{1}{2},j} A_{i+\frac{1}{2},j} G_{i+\frac{1}{2},j}^{I,n+1} .$$

Finally, the choice of interface flux depends on whether a portion of the still exists in cell (i, j) at the new time or not. If the front cuts cell (i, j) at t^{n+1} , we use the gradient approximation $G_{i,j}^{f,n+1}$, given in (4.52), and define the interface flux as

$$F_{i,j}^{f,n+1} = \beta_{i,j}^f A_{i,j}^f G_{i,j}^{f,n+1} .$$

Otherwise, the front has left cell (i, j) and we use the interpolated gradient $G_{i,j}^{I,n+1}$, given in (4.54),

$$F_{i,j}^{f,n+1} = \beta_{i,j}^f A_{i,j}^f G_{i,j}^{I,n+1} .$$

With these definitions, the finite volume scheme in cells interacting with the front is

$$V_{i,j}^{n+1}(M\phi)_{i,j}^{n+1} - V_{i,j}^n(M\phi)_{i,j}^n = \Delta t \sum_k n_k F_k^{n+1} , \quad (4.57)$$

where F_k^{n+1} represents the flux through each of the five surfaces of cell (i, j) , centered at t^{n+1} : edge fluxes $F_{i+\frac{1}{2},j}$, $F_{i-\frac{1}{2},j}$, $F_{i,j+\frac{1}{2}}$, or $F_{i,j-\frac{1}{2}}$; and the flux through the front, $F_{i,j}^f$. The variable n_k carries the proper sign for F to be the flux out of the control volume. Because the approximation of each term in (4.57) is second-order accurate, we obtain the cancellation of quadrature errors described earlier, resulting in a first-order accurate approximation of (Eq. II) in cells that contain a portion of the moving boundary.

4.5.3 Overall time-stepping procedure

We can now summarize the algorithm that is be used to advance (Eq. III) in time:

1. Calculate the front speed using (4.28), and extend the speed to neighboring cell edges using the algorithm described in the front-tracking section.
2. Move the front, using the operator-split advection scheme to update Λ , and the approximate least-squares reconstruction algorithm.
3. Calculate the space-time geometry, based on a piecewise-linear front moving with a constant velocity.
4. Recalculate the front normal and aperture to satisfy (4.37).
5. Construct the coefficients of the discretization (4.57), in cells that interact at the front.
6. Coarsen the geometry to each grid in the multi-grid hierarchy.
7. Use the multi-grid iteration strategy to calculate ϕ^{n+1} .

4.6 Modifications to Multi-grid iteration strategy

Again, we use multi-grid iterations to solve for $\phi_{i,j}^{n+1}$ in (2.14). Rearranging the terms in (2.14) yields the following linear system:

$$V_{i,j}^{n+1}(M\phi)_{i,j}^{n+1} - \Delta t \sum_k n_k F_k^{n+1} = V_{i,j}^n(M\phi)_{i,j}^n .$$

We will use H to designate the linear operator applied to $\phi_{i,j}^{n+1}$ in this equation, we write the point-relaxation strategy as before,

$$\phi_{i,j}^m = \phi_{i,j}^{m-1} + \frac{1}{\mu} \left(\rho_{i,j} - H\phi_{i,j}^{m-1} \right) . \quad (4.58)$$

Here μ is the diagonal entry of H , and m refers to the point-relaxation iteration number. Because H is indefinite, there is not guarantee that this strategy will converge; however,

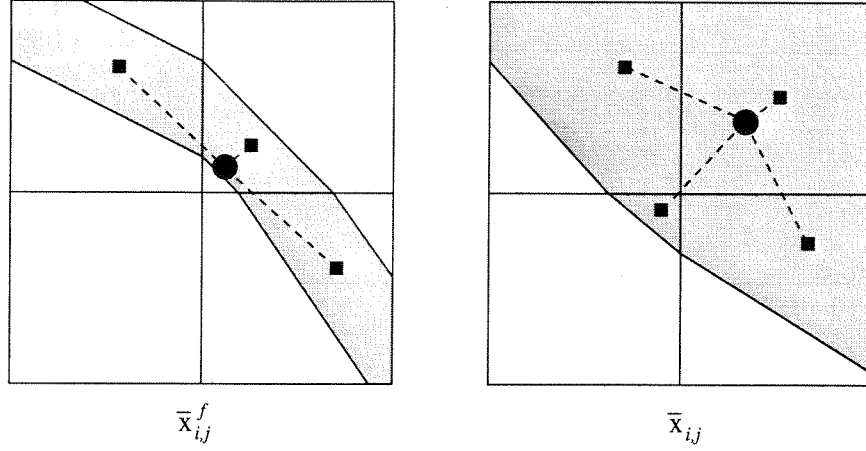


Figure 4.33: The location of $\bar{x}_{i,j}^f$ and $\bar{x}_{i,j}$ on coarser grids (marked with circles) is determined by the $A_{i,j}^f$ - or $V_{i,j}$ -weighted average of values from the fine grids (marked with squares).

the numerical results below will demonstrate that the approach is as successful here, as it was being applied to (Eq. I) and (Eq. II).

4.6.1 Coarsening strategy

The multi-grid approach for the Stefan problem incorporates the motion of the boundary, by deriving the coarse-grid stencils from the geometric properties on the finer grids. This is done by coarsening the space-time apertures and centroids, so that the finite-difference stencil in (2.14) can be defined on the coarse grid as well. First, the $A_{i+\frac{1}{2},j}$, $A_{i,j+\frac{1}{2}}$ and $V_{i,j}$ are coarsened as in the previous chapters: the coarse-grid quantities are defined as the sum of the corresponding fine-grid values. Next, the coarse-grid front apertures are defined as the sum of over the fine A^f . Given these coarse A_k and $V_{i,j}$, the coarse-grid normal is defined using (4.37), and solving for \mathbf{n} :

$$A^f \mathbf{n} = \left(A_{i+\frac{1}{2},j} - A_{i-\frac{1}{2},j}, A_{i,j+\frac{1}{2}} - A_{i,j-\frac{1}{2}} \right).$$

The remaining centroid-type quantities are defined in a way that preserves the properties of a linear front, on the coarse grid. For example, Figure 4.33 shows how the

front centroids $\bar{\mathbf{x}}_{i,j}^f$ are defined: the centroid of the combined fine-grid front areas, A^f , defines the coarsened front centroid. Similarly, the coarse cell centroids $\bar{\mathbf{x}}_{i,j}$ are weighted with $V_{i,j}$, although in this case we have to include the centroids of the full fine grid cells as well (Figure 4.33). Finally, the edge centroids $\bar{\mathbf{x}}_{i+\frac{1}{2},j}$ are coarsened using weighting based on the fine $A_{i+\frac{1}{2},j}$, again accounting for full edges on the fine grid. These averaging procedures guarantee that the space-time properties are reconstructed exactly, for a linear front moving with constant speed.

4.7 Software Implementation

Our discretization has been implemented using the C++ classes from the previous two chapters, with only two additional classes. As for the heat equation, the additional work is minimized due to the object-oriented programming methodology we have employed, using inheritance and polymorphism.

4.7.1 Description of MovingVoF

The first class, `MovingVoF`, organizes all the data needed to advance the front, and calculate the space-time characteristics of the front motion. It contains two `VoF` objects, which are used to describe the front location at t^n and t^{n+1} . A `MovingVoF` includes functions for implementing the advection algorithm described above, including extending velocities to cell edges, calculating edge fluxes of F^Λ , and reconstructing the front using the least-squares linear fit algorithm. In addition, quantities like A^f , $\bar{\mathbf{x}}$, etc., are maintained in one-dimensional arrays, which are resized every time step to match the number of partial cells.

4.7.2 Description of `StefOp`

The discretization (2.14) is implemented in the `StefOp` class, which contains all the information required for the finite difference stencils, like in Figures 4.31 and 4.32. This class is initialized using a `MovingVof`, and contains the functions needed to coarsen the space-time geometry for the multi-grid algorithm described in the previous section. The `StefOp` class is also derived from the `Operator` class for square domains. As in the previous chapter, this allows us to extend the algorithm for the partial cells only, and use existing functions for performing operations on the remainder of the domain. This is transparent to the `MultiGrid` class, so that the same structure is used to apply the prolongation and restriction operators, without any changes.

4.8 Results in Two Dimensions

In this section, we compute solutions for several test problems, to confirm the truncation error arguments made above, and estimate the influence of the error along the moving boundary on the solution. Each test problem evaluates a different aspect of the algorithm. First, we compute a solution to Laplace's equation on a changing domain, to check the accuracy of the fluxes in boundary cells. Next, a simple solution to the heat equation is used to assess the discretization of the time derivatives. An exact solution to the Stefan problem in two dimensions is then used to verify the truncation error at the domain boundary. Finally, we perform several convergence studies to assess the accuracy of the method for more complicated initial conditions and boundary motion.

In these calculations we set the number of points in each direction to be $N = k2^i$, where $k \in \{4, 5, 6\}$ and $i \in \{3, 4, 5, 6\}$, for a total of twelve different grid sizes, ranging from 32×32 to 384×384 . The grid spacing in both directions is determined from $\Delta x = \frac{1}{N}$, which in turn specifies the timestep $\Delta t = \text{Cr} \frac{\Delta x}{|u|}$, where $|u|$ is the front speed and Cr is the Courant number. Note that this implies $\Delta t \propto \Delta x$, so that we may use the expressions

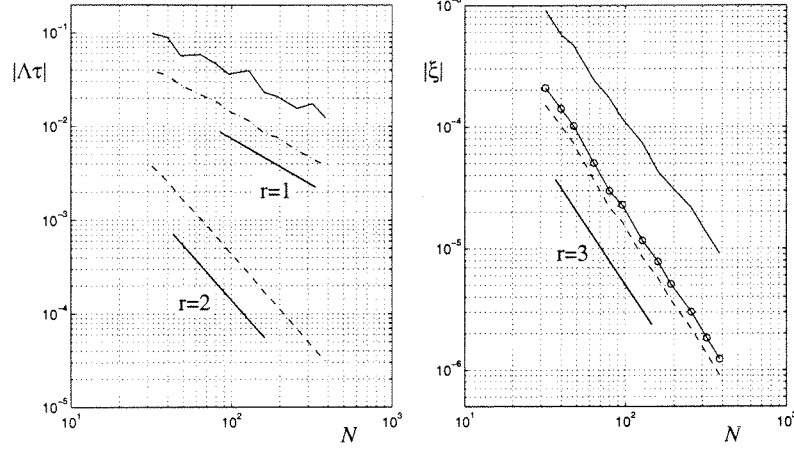


Figure 4.34: We plot the results of the first test problem, with $\varphi(\mathbf{x}) = \ln \frac{r}{R}$, versus the number of points in each grid direction, N . On the left, we see that the ∞ -norm (solid line) and one-norm (dash-dot line) of the truncation error $|\Lambda\tau|$ in partial cells is $O(\Delta x)$, whereas it is $O(\Delta x^2)$ in the interior (dashed line). However, on the right, we see that the error induced by the truncation error in partial cells is $\xi = O(\Delta x^3)$, in all norms.

$O(\Delta x)$ and $O(\Delta t)$ interchangeably.

For problems where the domain boundary is specified, we use the grid generation algorithm from Chapter 2, which finds the intersections of the domain boundary with cell edges, and then connects them to create a piecewise-linear representation. However, because the volume fraction always multiplies the the truncation error at the moving boundary, we will use $\Lambda\tau$ throughout this section. In addition, the other norms defined in Chapter 2 are no longer calculated with volume-fraction-weighting, so each part is equally weighted.

4.8.1 Laplacian on a Changing Domain

Our first test problem verifies the truncation error estimates, for the discretization of the Laplacian operator on a changing domain. We specify the exact solution $\varphi(r, t) = \ln \frac{r}{R}$, with $R = 0.5$, which is axisymmetric and harmonic. We compute the discrete solution

from $t = 0$, on the domain

$$\Upsilon = \{r : r > R + t\} \cap \{(x, y) : 0 < x < 1, 0 < y < 1\}.$$

Thus, φ is constant in time, and the front moves outward with speed $|u| = 1$. Dirichlet boundary conditions are given for $\varphi(r, t)$ along the boundaries at $r = R + t$, $x = 1$, and $y = 1$, and we specify homogeneous Neumann boundary conditions at $x = 0$ and $y = 0$.

The truncation error for the first time step is given in Figure 4.34. In partial cells, we observe that $|\Lambda\tau|$ is approximately $O(\Delta x)$ in both the ∞ - and one-norms; in the interior of the domain, the discrete Laplacian is second-order accurate, as expected. We may also calculate the error in the solution, $|\xi|$, induced by $|\Lambda\tau|$ in partial cells, as was done in Chapter 2. In Figure 4.34, we see that this is $O(\Delta x^3)$, in all norms. This result is the same at that obtained for fixed domains, which we attribute to the Dirichlet boundary condition, which causes each boundary cell to contribute only an $O(\Delta x^4)$ dipole field to the error, so that $O(N)$ boundary cells cause only an $O(\Delta x^3)$ error in the solution. We conclude that moving the charges a distance $O(\Delta x)$ away from the boundary does not destroy this effect.

4.8.2 Parabolic solution for the heat equation

We now construct an analytic solution to the heat equation with a moving boundary, which satisfies the homogeneous Dirichlet boundary condition at the front. Consider an axisymmetric $\varphi(r, t)$ given by

$$\varphi(r, t) = \frac{1}{4} (r^2 - r_0^2) + \beta t, \quad (4.59)$$

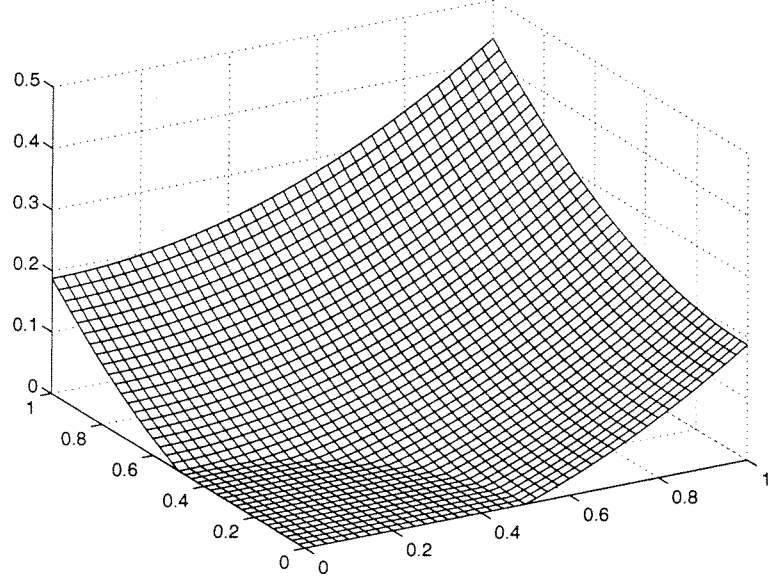


Figure 4.35: We plot the initial conditions for the second test problem, with exact solution $\varphi(r, t) = \frac{1}{4}(r_0^2 - r^2) + \beta t$.

for some constant r_0 . This satisfies (Eq. II) with constant β on $r > s(t)$, and has $\varphi(s(t), t) = 0$ for

$$s(t) = \sqrt{r_0^2 - 4\beta t} ,$$

so that the domain boundary is circular, and shrinking towards the origin.

We compute the solution on the region Υ in the first quadrant,

$$\Upsilon(t) = \{r : r > s(t)\} \cap \{(x, y) : 0 < x < 1, 0 < y < 1\} .$$

A plot of $\varphi(r, 0)$ on $\Upsilon(0)$ is given in Figure 4.35. We specify Dirichlet boundary conditions, using (4.59) with $r_0 = \frac{1}{2}$, on the boundaries at $x = 1$, $y = 1$, and $r = s(t)$, where $\varphi = 0$. Again, we specify homogeneous Neumann boundary conditions at $x = 0$ and $y = 0$. Finally, the discrete solution is initialized from (4.59) as well, evaluated at cell-centers, $t = 0$. The timestep is chosen so that $|s(t^n + \Delta t) - s(t^n)| = \text{Cr} \Delta x$, where we choose $\text{Cr} = \sqrt{\frac{1}{2}}$.

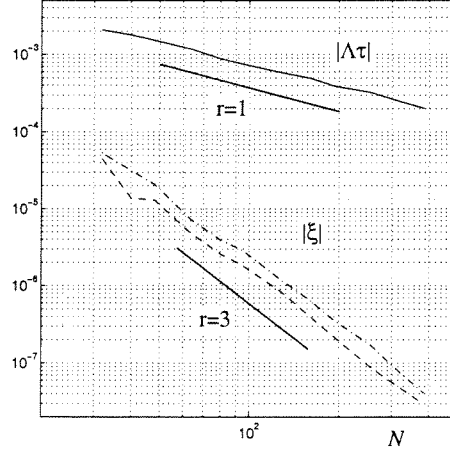


Figure 4.36: Because the exact solution to the second test problem is a low-order polynomial, the truncation error for our discretization is zero everywhere, except at the boundary, where it is $O(\Delta x)$ in the ∞ -norm (solid line). The corresponding error in the solution, $|\xi|$, is $O(\Delta x^3)$, either after one (dashed line) or $O(N)$ time steps (dash-dot).

This $\varphi(r, t)$ does not satisfy (Eq. III), but (4.36) is a low-order polynomial in r and t . This means that the backward Euler and hybrid discretizations will have *zero* truncation error in full cells; Figure 4.36 shows that the truncation error in partial cells is still $O(\Delta x)$. Because this is the only contributor to the error in ϕ , we find again that ξ is $O(\Delta x^3)$. However, Figure 4.36 demonstrates that even after $O(N)$ timesteps, $|\xi|$ is still $O(\Delta x^3)$. We assume that this is because errors introduced in previous time steps decay more rapidly than they accumulate.

4.8.3 Axisymmetric Analytic Solution with $St = 1$

Again, we refer to [15] for an axisymmetric exact solution to the Stefan problem (Eq. III), with β constant (we will use $\beta = 1$ here). Consider the initial value problem for the heat equation (Eq. II), with a constant source term, Q , centered at the origin, and

$$\varphi(r, 0) = 0 \text{ for } r > 0, \text{ and } \varphi(s(t), t) = 0. \quad (4.60)$$

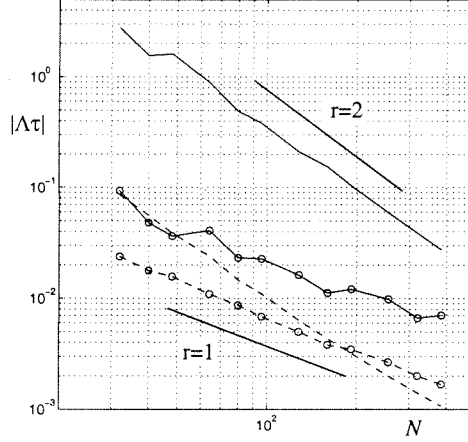


Figure 4.37: We plot the truncation error for the third test problem, where φ is given by the exact solution to the Stefan problem, (4.61). In partial cells, the ∞ - and one-norms of $|\Lambda\tau|$ are $O(\Delta x)$ (solid, dashed lines with circles, respectively). Over the part of the domain away from the origin, the truncation error is $O(\Delta x^2)$ (solid, dashed lines).

As in one dimension, we specify the location of the front as $s(t) = 2\kappa\sqrt{\beta t}$, with $\kappa \approx 0.27$.

We may then derive an exact solution for [15],

$$\varphi(r, t) = \frac{Q}{4\pi\beta} \left[\text{Ei} \left(\frac{r^2}{4\beta t} \right) - \text{Ei}(\kappa^2) \right] \quad \text{for } r \in (0, s(t)), \quad (4.61)$$

where $\text{Ei}(x)$ is the exponential integral,

$$\text{Ei}(x) = \int_x^\infty \frac{e^{-t}}{t} dt. \quad (4.62)$$

The choice of κ is necessary for (4.61) to satisfy (Eq. III), for $\text{St} = 1$. Note that φ is unbounded as $r \rightarrow 0$, because $\text{Ei}(x)$ has a logarithmic singularity at the origin. This prevents us from computing an accurate finite-difference representation of φ , but we can still use it to calculate the discretization's truncation error.

For this calculation, Δt is given by the exact front speed $|u| = \dot{s}$ and $\text{Cr} = \frac{1}{2}$, with an initial time $t = 1.0$, for which $s \approx 0.54$. For interior cells, we calculate $|\Lambda\tau|$ only for $r \geq 0.1$, to avoid the influence of the singularity. Figure 4.37 plots the ∞ - and one-norms

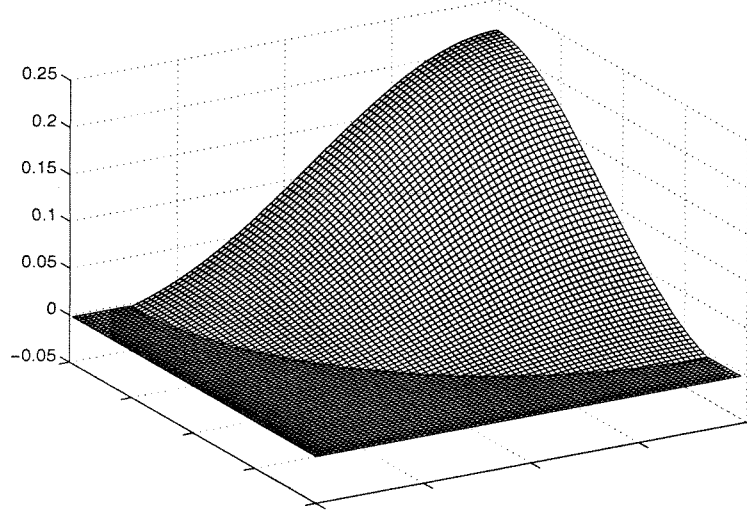


Figure 4.38: We plot the discrete solution to the fourth test problem, with $N = 80$, after 40 time steps. The result is smooth, and has a mild change in curvature, indicating that the front moves more quickly than the rate of diffusion.

of $|\Delta\tau|$ for the hybrid method. We see that the truncation error in the interior is $O(\Delta x^2)$; the large difference between the two norms is due to the influence of the singularity. For partial cells, the truncation error is $O(\Delta x)$ in both norms, indicating that we have correctly approximated the spatial and temporal derivatives next to the moving boundary.

4.8.4 Axisymmetric initial conditions

Next, we wish to evaluate the accuracy of the discretization for the heat equation after $O(N)$ timesteps. We specify the initial conditions

$$\varphi(r, t = 0) = 4(r_0^2 - r^2),$$

which has a maximum of $\varphi = 1$ for $r_0 = \frac{1}{2}$. A solution is computed on the quarter-circle $\Upsilon = \{(x, y) : x, y > 0 \text{ and } r < s(t) = r_0 + t, \text{ so that the speed of the boundary is } |u| = 1\}$. The timestep Δt is given by $\Delta t = Cr \Delta x$, where $Cr = \sqrt{\frac{1}{2}}$, and the solution is advanced for

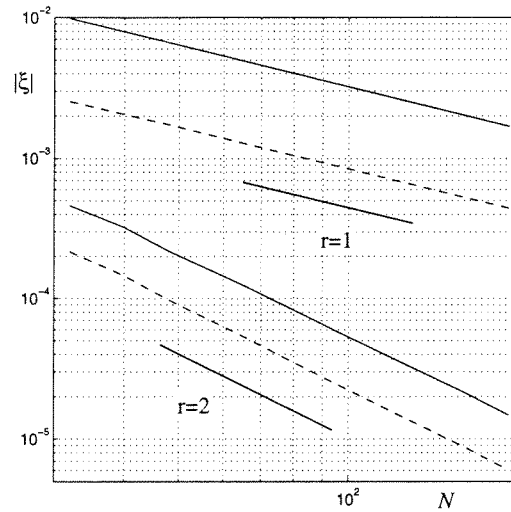


Figure 4.39: We plot the error in the discrete solution, $|\xi|$, for the fourth test problem. The backward Euler method is first-order accurate in both the ∞ - and one-norms (solid and dashed lines at top), whereas the hybrid method is second-order accurate (solid and dashed lines at bottom).

a total of $\frac{N}{2}$ timesteps.

Figure 4.38 shows the discrete solution at the final time for the backward Euler method, with $N = 80$ points in each direction; the change in the surface curvature suggests that the diffusion timescales are longer than those of the boundary motion. Although there is no simple exact solution for this problem, we can compare solutions for grid spacings that differ by a factor of two, as described in Chapter 2. Figure 4.39 plots the ∞ - and one-norms of $|\xi|$ for both the backward Euler and hybrid discretizations, which are $O(\Delta x)$ and $O(\Delta x^2)$, respectively. Figure 4.40 shows the error for the hybrid method, with $N = 80$; a high-frequency noise at $r \approx 0.5$ is immediately obvious. We attribute this to the hybrid method's inability to damp high-frequency components of the solution, so that the large initial errors (at $r = 0.5$) persist, even after $O(N)$ timesteps.

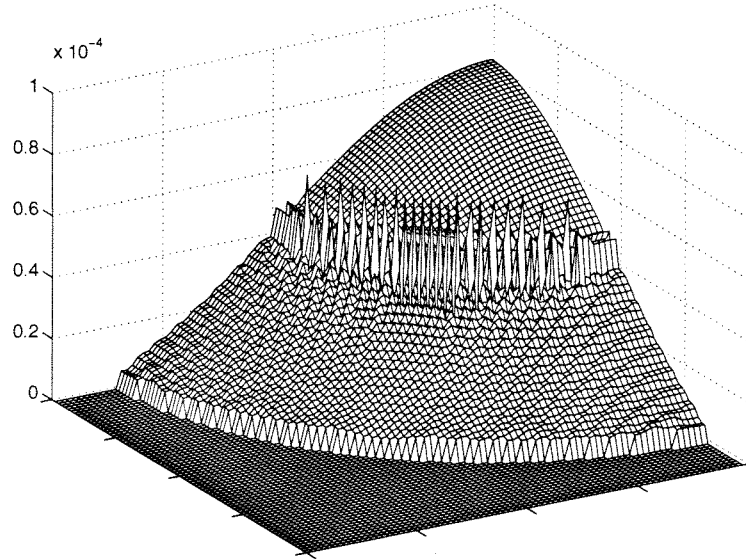


Figure 4.40: We apply the hybrid algorithm with $N = 80$ to the fourth test problem, and plot the solution error after 40 timesteps. Note that the hybrid method does not damp high-frequency components of the error caused from the initial discretization error at $\tau = 0.5$.

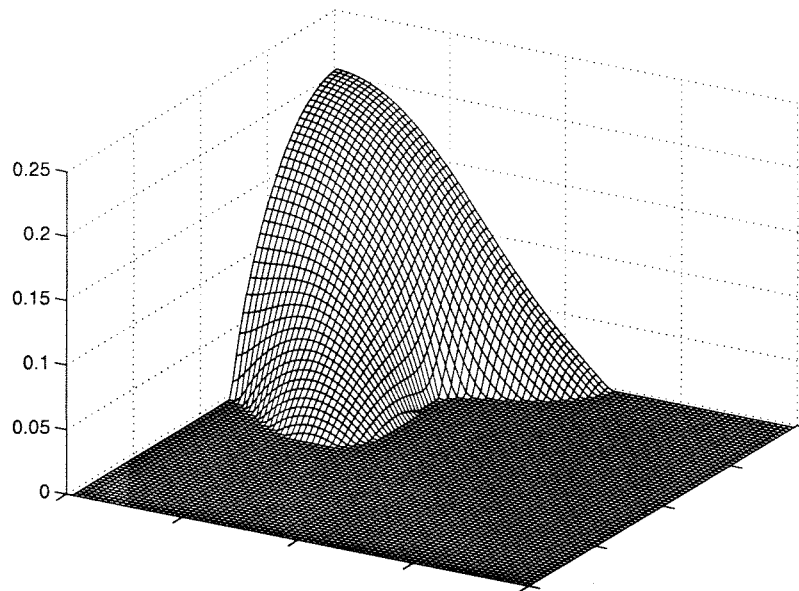


Figure 4.41: We plot the initial solution value and boundary position for the last test problem, with $N = 80$.

4.8.5 Star-shaped initial configuration

Now that the discretization for the heat equation has been tested, we can evaluate the combined front-tracking and heat equation algorithms, by applying them to the Stefan problem, with $St = 100$. We solve the following Poisson problem to specify a smooth initial value for φ :

$$\Delta\varphi(\mathbf{x}, t = 0) = 4 \text{ in } \Upsilon, \text{ with } \varphi = 0 \text{ on } \partial\Upsilon_1,$$

where $\Upsilon = \Upsilon_1 \cap \Upsilon_2$,

$$\Upsilon_1 = \{(r, \theta) : r \leq 0.25 + 0.05 \cos 6\theta\},$$

and $\Upsilon_2 = \{(x, y) : x > 0, y > 0\}$. The method described in Chapter 2 is used to discretize the initial conditions and calculate the $\varphi(\mathbf{x}, 0)$ for the initial boundary location (Figure 4.41). Note that the front is perpendicular to the boundaries at $x = 0$ and $y = 0$, where we specify homogeneous Neumann boundary conditions. The timestep is calculated from the discrete gradient of the solution, as specified by (4.28),

$$\mathbf{u}_{i,j} = -St \mathbf{n}_{i,j}^f F_{i,j}^{f,n},$$

and the maximum value of $Cr = \frac{1}{2}$, for the operator-split advection algorithm. In each case, a total of $\frac{N}{4}$ steps are taken.

Figure 4.42 plots the location of the front at each time, for the backward Euler discretization with $N = 40$. As expected, the parts of the front where the gradient is largest move more quickly, which makes the front more circular in time. At these parts of the boundary, the front-tracking algorithm keeps the interface relatively coherent, even at this coarse resolution. To assess the accuracy of the solution, we again compare results corresponding to different grid spacings: Figure 4.43 plots the the ∞ -, one-, and two-

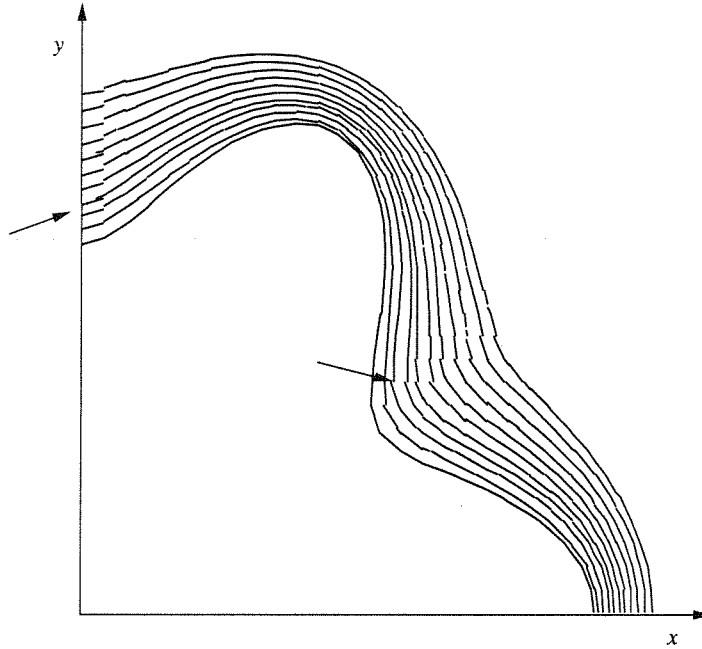


Figure 4.42: We plot the evolution of the front in time for the last test problem, with $N = 40$. For this coarse grid spacing, the front-tracking algorithm is still able to maintain a coherent interface at the fastest-moving part of the front.

norms of ξ , for both the backward Euler and hybrid methods. Both sets of results indicate convergence rates between $O(\Delta x)$ and $O(\Delta x^2)$. Because the initial conditions are computed using our discretization for the Poisson equation, we expect them to be $O(\Delta x^2)$ accurate; this is the behavior seen in Figure 4.43 for small N . For larger N , the backward Euler method seems to be asymptoting to $O(\Delta x)$, whereas the result for the hybrid method appears to be $O(\Delta x^2)$. Again, we attribute this behavior to convergence of the initial conditions, and not necessarily the accuracy of the algorithm.

4.9 Conclusions

In this chapter, we extend the approach for (Eq. II) to include domains that expand in time. A finite-volume discretization is presented, in which quadrature rules are applied at the surfaces of the space-time control volume defined by the expanding boundary.

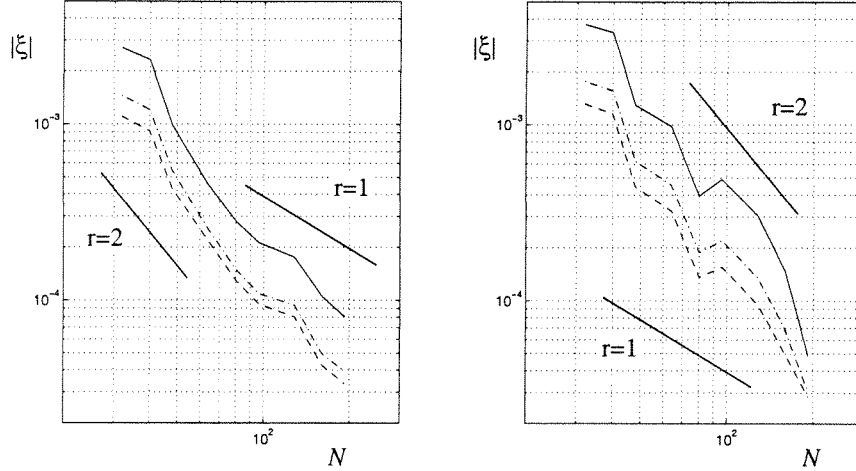


Figure 4.43: We plot the ∞ -, one-, and two-norms of the solution error $|\xi|$ (solid, dashed, and dash-dot lines) for the backward Euler (left) and hybrid (right) methods. At coarser grid resolutions, both appear to be $O(\Delta x^2)$; we attribute this to the convergence of the discretization of the initial conditions. The $O(\Delta x)$ errors of the backward Euler method become apparent at smaller grid spacings, whereas this is not yet the case for the hybrid algorithm.

Again, by carefully choosing the location of the quadrature points, we obtain a consistent discretization, using the geometric properties of the control volume. In conjunction with an explicit discretization of (Eq. III) to advance the domain boundary in time, we are also able to apply the method to the classical Stefan problem.

In one space dimension, we develop both backward Euler and hybrid discretizations for the heat equation on expanding domains. Both approaches apply careful quadrature rules to the surfaces of the space-time region traversed by the front, to obtain first-order accurate discretizations in cells next to the moving front. By comparing the results to an analytic solution to the Stefan problem, we find that the truncation error does not adversely affect the overall accuracy of the solution: the backward Euler method is still $O(\Delta x)$ accurate, while the hybrid approach is $O(\Delta x^2)$ accurate. However, lagged enforcement of the conservation equation (Eq. III) causes instability in both cases, so that we have no conservative algorithm for the Stefan problem in one dimension.

In two space dimensions, the space-time trajectory of the boundary requires us to use algorithms from the front-tracking literature. The space-time geometry is represented as a linear front with constant velocity, providing a first-order accurate description of the boundary motion. We again show that carefully applying quadrature rules at the surfaces of the control volume in space-time, leads to a consistent discretization with even this simple geometric description. Several test problems verify the backward Euler and hybrid approaches for the heat equation on an expanding domain. Incorporating an explicit advection algorithm for the front motion, provides a first-order accurate, but non-conservative, discretization for the classical Stefan problem.

Chapter 5

Conclusions

5.1 Summary

This thesis presents a natural extension of the embedded boundary method, to include elliptic and parabolic partial differential equations defined on moving domains. Our motivation is to augment the algorithms available for hyperbolic problems, and provide the tools needed to approach a broader class of problems that include both heat transfer and phase change. This thesis attempts to overcome the major drawbacks of rectangular-grid finite-volume methods – lack of adaptivity and geometric flexibility – while retaining their simplicity and regular data structures.

To demonstrate that this is possible, three model problems with Dirichlet boundary conditions are discretized on time-dependent domains: the Poisson equation, the heat equation, and classical Stefan problem. In each case, a consistent finite-volume method is derived, by integrating the differential equation over the space-time volume, defined by the intersection of a piecewise-linear moving boundary and each cell of a Cartesian grid. The resulting surface integrals are discretized with single-point quadrature rules, that take advantage of geometric properties to cancel the leading-order error terms. The integrands are then approximated to second-order, using finite-differences of the cell-centered solution,

even when those cell-centers are outside the domain. Additional consideration is given to the conditioning of the resulting set of equations; gradients at the boundary are calculated using stencils that are well-separated from the partial cells. Although this yields a nonsymmetric linear system, we are able to combine point-relaxation with a multi-grid iteration strategy, to obtain residual reduction rates per iteration that are nearly independent of the grid spacing and the presence of arbitrarily small cells. This behaviour persists, even when incorporated into an adaptive mesh hierarchy.

5.2 Poisson Equation

Chapter 2 presents a finite-volume discretization for (Eq. I), along with a multi-grid algorithm for solving the resulting linear system, and an adaptive mesh algorithm for improving the accuracy of the solution.

In deriving the finite-volume algorithm, the integrals of fluxes on each cell edge are approximated using the midpoint rule, along with second-order accurate finite-difference approximations for the solution gradient. The discretization's truncation error is $O(\Delta x \Lambda^{-1})$ along the domain boundary, where Λ is the volume fraction in that cell, and $O(\Delta x^2)$ in the interior. In three test problems, the approach yields second-order accurate solutions, on domains with significant curvature and variation. We also observe that the truncation error on the boundary causes only an $O(\Delta x^3)$ error in the solution, because of the Dirichlet boundary condition. Each boundary cell contributes a dipole field of strength $O(\Delta x^4)$, so that the $O(N)$ boundary points cause only an $O(\Delta x^3)$ error in the solution.

A significant feature of the discretization, is that gradients on the domain boundary are derived using finite-difference stencils which are well-separated from the domain boundary. In one space dimension, we show that the resulting discrete operator L is non-symmetric, but has a condition number that is unaffected by arbitrarily-small cells, and comparable to standard finite-difference discretizations on regular grids. A similar ap-

proach is employed in two dimensions, so that we may apply multi-grid iterations with only a simple point-relaxation scheme, with volume-weighted restriction and piecewise constant prolongation operators. This multi-grid algorithm obtains nearly the same multi-grid reduction rates for the residual, almost independent of Δx or grid quality. This suggests that we retain a well-conditioned system in more than one dimension, as well.

We also demonstrate that our method can be combined with adaptive mesh refinement to improve the accuracy locally. By refining the cells containing portions of the domain boundary, we simultaneously refine the geometry description, while reducing the effect of the larger truncation error. The multi-grid framework attains residual reduction rates for the adaptive grid hierarchy that are comparable to those observed on uniform grids.

5.3 Heat Equation

In Chapter 3, we extend the finite-volume discretization of (Eq. I) to the heat equation with variable coefficients (Eq. II). Four implicit discretizations are derived to address different accuracy and dissipation requirements when $\Delta t \propto \Delta x$. Each approach relies on a cancellation of quadrature errors in the integral of φ over irregular cells; these terms can then be approximated with cell-centered values, instead of using the $O(\Delta x^2)$ midpoint rule, without loss of accuracy.

By examining the eigensystem of $L\phi$ in one dimension, we are able to evaluate different time discretizations for (Eq. II). First, we demonstrate that the maximum eigenvalue of L is $O(N^2 \Lambda^{-1})$, where N is the number degrees of freedom. This suggests that explicit methods using L are unstable for arbitrarily small Λ ; instead, we introduce implicit discretizations based on the backward Euler and Crank-Nicholson methods. The Crank-Nicholson method is second-order accurate for smooth initial conditions, but does not damp the modes corresponding to the largest eigenvalues of L . The backward Euler method pro-

vides sufficient damping for these modes, but is only first-order accurate. We also present a Runge-Kutta algorithm, which provides moderate damping, and second-order accuracy, but requires that two linear systems be inverted. Finally, we develop a non-conservative, hybrid algorithm, which applies the backward Euler method in regions where dissipation is required, and the Crank-Nicholson algorithm elsewhere. In all cases, the truncation error in the partial cells is $O(\Delta x \Lambda^{-1})$, but only introduces an $O(\Delta x^3)$ error in the solution each timestep, regardless of the method. Again, this is attributed to the Dirichlet boundary condition, and the resulting dipole field.

Each of the discretizations extends easily to two dimensions, without modifying the approach used for (Eq. I). Only minimal changes are required to the multi-grid algorithm, to include each of the four linear systems. For smooth initial conditions, numerical experiments show that the truncation error is again $O(\Delta x \Lambda^{-1})$ in partial cells, and induces only an $O(\Delta x^3)$ error in the solution each timestep. In each case, the solution has a time-accuracy corresponding to the method used in the interior of the domain.

5.4 Stefan Problem

In Chapter 4, we extend the approach for (Eq. II) to include domains that expand in time. A finite-volume discretization is presented, in which quadrature rules are applied at the surfaces of the space-time control volume defined by the expanding boundary. Again, by carefully choosing the location of the quadrature points, we obtain a consistent discretization, using the geometric properties of the control volume. In conjunction with an explicit discretization of (Eq. III) to advance the domain boundary in time, we are also able to apply the method to the classical Stefan problem.

In one space dimension, we develop both backward Euler and hybrid discretizations for the heat equation on expanding domains. Both approaches apply careful quadrature rules to the surfaces of the space-time region traversed by the front, to obtain first-order

accurate discretizations in cells next to the moving front. By comparing the results to an analytic solution to the Stefan problem, we find that the truncation error does not adversely affect the overall accuracy of the solution: the backward Euler method is still $O(\Delta x)$ accurate, while the hybrid approach is $O(\Delta x^2)$ accurate. However, lagged enforcement of the conservation equation (Eq. III) causes instability in both cases, so that we have no conservative algorithm for the Stefan problem in one dimension.

In two space dimensions, the space-time trajectory of the boundary requires us to use algorithms from the front-tracking literature. The space-time geometry is represented as a linear front with constant velocity, providing a first-order accurate description of the boundary motion. We again show that carefully applying quadrature rules at the surfaces of the control volume in space-time, leads to a consistent discretization with even this simple geometric description. Several test problems verify the backward Euler and hybrid approaches for the heat equation on an expanding domain. Incorporating an explicit advection algorithm for the front motion, provides a first-order accurate, but non-conservative, discretization for the classical Stefan problem.

5.5 Future Research Directions

The results of this thesis suggest a number of future research topics. First, the framework we have presented for developing finite-volume algorithms could be extended to other conservative discretizations, for problems with irregular boundaries. For example, a similar approach could be applied to hyperbolic problems, modifying the approach discussed in [49], to obtain a method that is formally consistent at the boundary. It is unknown, however, what the tradeoffs are between improved accuracy and robustness in the presence of shocks and other discontinuities in such systems. Further extension to include incompressible fluid flows, like in [3] and [5], would allow us to simulate buoyancy-driven flows, with advected scalars. Finally, our discretization for the classical Stefan problem

could be added, to simulate complicated industrial processes, with the combined effects of fluid flow, heat transfer, and phase change across internal boundaries.

A number of issues have to be addressed before this can be accomplished. First, the current implementation is well-suited to external boundaries, but internal interfaces will require more sophisticated data structures, to allow jumps in the material properties and thin “fingers” within the multi-grid hierarchy. Another open question is how to develop volume of fluid representations in three dimensions, with more accurate boundary representations, as in [2]. For problems where body-fitted grids are preferable for resolving boundary layers, the algorithm could be extended to an overset grid approach [19], where the irregular boundary is given by intersections between moving grids. Another issue is conservation of energy; a lagged source term introduced instabilities for our discretization of the heat equation. Although more complicated redistribution algorithms have been introduced for hyperbolic problems [18, 49], it is not known how to do this for parabolic problems. Finally, extending adaptive mesh refinement to the heat equation, will present additional problems with time centering [5], and with moving boundaries crossing levels of refinement [9, 49].

Bibliography

- [1] L. Adams. A multigrid algorithm for immersed interface problems. In *Proceedings, 7th Copper Mountain Multigrid Conference, Copper Mountain, CO*, 1995. See <http://na.cs.yale.edu/mgnet/www/mgnet-ccmm95.html>.
- [2] M. J. Aftosmis, J. E. Melton, and M. J. Berger. Adaptation and surface modelling for Cartesian mesh methods. In *AIAA 12th Computational Fluid Dynamics Conference, San Diego, CA*, 1995. AIAA-95-1725-CP.
- [3] A. Almgren, J. B. Bell, P. Colella, and T. Marthaler. A Cartesian grid projection method for the incompressible Euler equations in complex geometries. *SIAM J. Sci. Comput.* (to appear).
- [4] A. S. Almgren, J. B. Bell, P. Colella, , and L. H. Howell. An adaptive projection method for the incompressible Euler equations. In *AIAA 11th Computational Fluid Dynamics Conference, Orlando, FL, July 6-9, 1993*.
- [5] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, , and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Sci. Comput.* (submitted).
- [6] R. Almgren and A. S. Almgren. Phase field instabilities and adaptive mesh refinement. In *Proceedings of Modern Methods for Modeling Microstructure in Materials, TMS/SIAM, Cleveland, OH, October, 1995*.

- [7] R. E. Bank. *PLTMG, a Software Package for Solving Elliptic Partial Differential Equations*. SIAM, Philadelphia, PA, 1994.
- [8] J. B. Bell, P. Colella, and L. Howell. An efficient second-order projection method for viscous incompressible flow. In *Proceedings, AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 24-27, 1991*.
- [9] J. B. Bell, P. Colella, and M. Welcome. Conservative front-tracking for inviscid compressible flow. In *Proceedings, AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 24-27, 1991*.
- [10] M. J. Berger and J. Olinger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484-512, 1984.
- [11] M. J. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Trans. on Sys. Man and Cyber.*, 21:1278-1286, 1991.
- [12] D. E. Bornside, T. A. Kinney, R. A. Brown, and G. Kim. Finite element Newton method for the analysis of czochralski crystal growth with diffuse-grey radiative heat transfer. *Int. J. Num. Meth. Engin.*, 30:133-54, 1990.
- [13] J. H. Bramble, R. E. Ewing, J. E. Pasciak, and J. Shen. The analysis of multigrid algorithms for cell-centered finite difference methods. *Adv. Comput. Math.*, 5:15-29, 1996.
- [14] W. Briggs. *A Multigrid Tutorial*. SIAM, Philadelphia, PA, 1987.
- [15] H. S. Carslaw and J. C. Jaeger. *Conduction of Heat in Solids*. Clarendon Press, New York, 1986.
- [16] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 41-143, 1994.

- [17] T. Cheng and C. S. Peskin. Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods. *SIAM J. Sci. Stat. Comp.*, 13:1361–76, 1992.
- [18] I. Chern and P. Colella. A conservative front tracking method for hyperbolic conservation laws. Technical Report UCRL JC-97200, Lawrence Livermore National Laboratory, July 1987. (unpublished).
- [19] G. Chesshire and W. D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *J. Comput. Phys.*, 90:1–64, 1990.
- [20] W. J. Coirier. *An Adaptively-Refined, Cartesian, Cell-Based Approach for the Euler and Navier-Stokes Equations*. PhD thesis, University of Michigan, Ann Arbor, Mich., 1994.
- [21] W. J. Coirier. *Automated Three-Dimensional Cartesian Grid Generation and Euler Flow Solutions for Arbitrary Geometries*. PhD thesis, University of California, Davis, CA, 1996.
- [22] W. Y. Crutchfield and M. L. Welcome. Object-oriented implementations of adaptive mesh refinement algorithms. *Scientific Programming*, 2:145–156, 1993.
- [23] L. Demkowicz, J. T. Oden, W. Rachowicz, , and O. Hardy. Toward a universal h-p adaptive finite element strategy. *Comp. Meth. in Appl. Mech. and Eng.*, 77:79, 1989.
- [24] N. Van den Bogaert and F. Dupret. Dynamic global simulation of the Czochralski process. 1. principles of the method. *J. Crystal Growth*, 171:65–76, 1997.
- [25] J. J. Derby. An overview of convection during the growth of single crystals from the melt. Technical Report 92-074, Army High-Performance Computing Research Center, 1992.

- [26] E. DornBerger, W. von Ammon, N. Van den Bogaert, and F. Dupret. Transient computer simulation of a CZ crystal growth process. In *Proceedings, Eleventh International Conference on Crystal Growth, The Hague, Netherlands, June 1995.*, pages 18–23, 1995.
- [27] A. B. Downey. *VIGL 3.0: visualization graphics library*. Technical Report UCB/CSD 95-000, Computer Science Division, University of California, Berkeley, August 1995. Code and documentation available at <http://http.cs.berkeley.edu/~downey/vigl/>.
- [28] I. Babuska *et al.*, editor. *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*. Springer-Verlag, New York, 1995.
- [29] M. Fabbri and V. R. Voller. The phase-field method in the sharp-interface limit: a comparison between model potentials. *J. Comput. Phys.*, 130:256–65, 1997.
- [30] L. Greengard and J. Lee. A direct Poisson solver of arbitrary order accuracy. *J. Comput. Phys.*, 125:415–424, 1996.
- [31] J. A. Hilditch. *A Projection Method for Low Mach Number Reacting Flow in the Fast Chemistry Limit*. PhD thesis, University of California, Berkeley, CA, 1997.
- [32] L. H. Howell and J. B. Bell. An adaptive-mesh projection method for viscous incompressible flow. *SIAM J. Sci. Comp.*, 18:996–1013, 1997.
- [33] D. T. J. Hurle, editor. *Handbook of Crystal Growth*. North-Holland, Amsterdam; New York, 1993–1994.
- [34] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, New York, 1987.
- [35] D. Juric and G. Tryggvason. A front-tracking method for dendritic solidification. *J. Comput. Phys.*, 123:127–148, 1996.

- [36] P. K. Kundu. *Fluid Mechanics*. Academic Press, San Diego, 1990.
- [37] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems*. Wiley, New York, 1991.
- [38] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhauser Verlag, Basel; Boston, 1992.
- [39] R. J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. of Numer. Anal.*, 31:1019–1044, 1994.
- [40] Z. Li. A fast iterative method for elliptic interface problems. *SIAM J. of Numer. Anal.* (to appear).
- [41] Z. Li. *The Immersed Interface Method – A Numerical Approach for Partial Differential Equations with Interfaces*. PhD thesis, University of Washington, Seattle, WA, 1994.
- [42] D. F. Martin and K. L. Cartwright. Solving Poisson’s equation using adaptive mesh refinement. Technical Report UCB/ERL M96/66, Electronics Research Laboratory Memorandum, University of California, Berkeley, October 1996. Code and documentation available at http://cfd.me.berkeley.edu/~colella_lab/.
- [43] MathWorks. *MATLAB, High-Performance Numeric Computations and Visualization Software*. Natick, Mass., 1993.
- [44] A. Mayo. The rapid evaluation of volume integrals of potential theory on general regions. *J. Comput. Phys.*, 100:236–245, 1992.
- [45] A. McKenney, L. Greengard, and A. Mayo. A fast Poisson solver for complex geometries. *J. Comput. Phys.*, 118:348–355, 1995.
- [46] M. Minion. A projection method for locally refined grids. *J. Comput. Phys.*, 127:158–178, 1996.

- [47] S. Osher and J. A. Sethian. Front propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.
- [48] R. B. Pember, A. Almgren, J. B. Bell, D. Marcus, , and W. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *J. Comput. Phys.*, 130:269–282, 1997.
- [49] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. Welcome. An adaptive Cartesian grid method for unsteady compressible flow in irregular regions. *J. Comput. Phys.*, 120:278–304, 1995.
- [50] R.B. Pember, A.S. Almgren, W.Y. Crutchfield, L.H. Howell, J.B. Bell, P. Colella, and V.E. Beckner. An embedded boundary method for the modeling of unsteady combustion in an industrial gas-fired furnace. In *1995 Fall Meeting of the Western States Section of the Combustion Institute, Stanford University, October 30–31, 1995*, 1995. See <http://www.nersc.gov/research/CCSE/publications/pub.html>.
- [51] C. S. Peskin and B. F. Printz. Improved volume conservation in the computation of flows with immersed elastic boundaries. *J. Comput. Phys.*, 105:33–46, 1993.
- [52] J. E. Pilliod and E. G. Puckett. Second-order volume-of-fluid algorithms for tracking material interfaces. *J. Comput. Phys.* (in preparation).
- [53] C. Rendleman. *BoxLib Tutorial*, 1993. Source code and documentation available at <http://www.nersc.gov/research/CCSE/software/boxlib.tutorial/boxlib.html>.
- [54] W. J. Rider. Approximate projection methods for incompressible flow: Implementation, variants, and robustness. Technical report, Computing, Information, and Communications Division, Los Alamos National Laboratory, NM, August 1994.
- [55] A. Schmidt. Computation of three dimensional dendrites with finite elements. *J. Comput. Phys.*, 125:293–312, 1996.

- [56] J. A. Sethian and J. Strain. Crystal growth and dendritic solidification. *J. Comput. Phys.*, 98:231–53, 1992.
- [57] W. Shyy and M. M. Rao. Calculation of meniscus shapes and transport processes in float zone. *Int. J. Heat and Mass Trans.*, 38:2281–2295, 1995.
- [58] J. Strain. Spectral methods for nonlinear parabolic systems. *J. Comput. Phys.*, 122:1–12, 1995.
- [59] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Wadsworth and Brooks/Cole, Pacific Grove, CA, 1989.
- [60] C.R. Swaminathan and V.R. Voller. Towards a general numerical scheme for solidification systems. *Int. J. Heat and Mass Trans.*, 40:2859–68, 1997.
- [61] E. H. Twizel, 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.*, 5:333–352, 1996.
- [62] S. O. Unverdi and G. Tryggvason. Computations of multi-fluid flows. *Physica D*, 60:70–83, 1992.
- [63] V. N. Vatsa, M. D. Sanetrik, and E. B. Parlette. Block-structured grids for complex aerodynamic configurations: Current status. In *Proceedings, Grid Generation Conference, NASA Lewis Research Center*, May 1995.
- [64] J. A. Warren and W. J. Boettinger. Prediction of dendritic growth and microsegregation patterns in a binary alloy using the phase-field method. *Acta Metallurgica et Materialia*, 43:689–703, 1995.
- [65] Q. Xiao and J. J. Derby. Heat transfer and interface inversion during the Czochralski growth of YAG and GGG. *J. Crystal Growth*, 139:147–57, 1994.

- [66] Q. Xiao and J. J. Derby. Three-dimensional melt flows in Czochralski oxide growth: high-resolution, massively parallel, finite element computations. *J. Crystal Growth*, 152:169–81, 1995.
- [67] Z. Yang. *A Cartesian Grid method for Elliptic Boundary Value problems in Irregular Regions*. PhD thesis, University of Washington, Seattle, WA, 1996.
- [68] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, and J. E. Bussoletti. An locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics. *J. Comput. Phys.*, 62:1–66, 1991.
- [69] H. Zhang and V. Prasad. A multizone adaptive process model for low and high pressure crystal growth. *J. Crystal Growth*, 155:47–65, 1995.
- [70] H. Zhang, V. Prasad, and D. F. Bliss. Modeling of high pressure, liquid-encapsulated Czochralski growth of InP crystals. *J. Crystal Growth*, 169:250–60, 1996.
- [71] W. Zhou, D. E. Bornside, and R. A. Brown. Dynamic simulation of Czochralski crystal growth using an integrated thermal-capillary model. *J. Crystal Growth*, 137:26–31, 1994.