A SCALABLE AND ADAPTIVE DISCRETIZATION FOR FREE SURFACE INCOMPRESSIBLE FLOW SIMULATION USING OVERLAPPING CARTESIAN GRIDS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Robert Elliot English
May 2013

Abstract

This dissertation presents a novel method for discretizing the incompressible Navier-Stokes equations using moving and overlapping Cartesian grids. Spatial adaptivity is addressed by chosing a cell size independently for each grid.

Advection is handled with first and second order accurate semi-Lagrangian schemes in order to alleviate any time step restriction associated with small grid cell sizes. The primary focus of this thesis, however, is the Poisson discretization used in order to solve the elliptic equation for the Navier-Stokes pressure or that resulting from the temporal discretization of parabolic terms such as viscosity. A novel discretization is introduced where the coupling terms are defined implicitly through the use of a Voronoi diagram computed for a subset of the Cartesian grid cell centers. The resulting Poisson discretization is second order accurate and requires the solution of a symmetric positive definite linear system. Each aspect of the approach is demonstrated independently on test problems in order to show efficacy and convergence before finally addressing a number of common test cases for incompressible flow with stationary and moving solid bodies. In order to represent free surfaces the particle level set method is extended to overlapping grids. The method includes a new treatment for particles near grid boundaries with disparate cell sizes, and strategies to deal with locality in the implementation of the level set and fast marching algorithms.

The resulting method is highly scalable on distributed parallel architectures with minimal communication costs, since it utilizes uniform Cartesian grids that exploit cache coherent data structures and algorithms.

Acknowledgments

I would like to thank ...

Contents

Abstract **v**

Acknowledgments **vi**

1 Introduction **1**

2 Overlapping Grids **11**

2.1 Cartesian Grids 11

2.2 Explicit Grid Coupling 13

2.3 Parallelization 17

3 Generalized Semi-Lagrangian Advection **19**

3.1 Semi-Lagrangian MacCormack Advection 19

3.2 Chimera Advection Scheme 24

3.2.1 First Order Semi-Lagrangian Scheme 24

3.2.2 Semi-Lagrangian MacCormack Advection Scheme 27

3.3 Time Step Size and Ghost Cells 30

3.4 Numerical Results 33

4 SPD Poisson Equation Discretization **39**

4.1 Poisson Equation 39

4.2 Overlapping Grid Solvers 40

4.3 Voronoi Diagram Discretization 46

4.4 Numerical Results 51

4.5 Matrix Conditioning 55

5 Diffusion Discretization **56**

5.1 Scalar Diffusion Equation 56

5.1.1 Moving Grids 59

5.2 Navier-Stokes Viscosity 63

5.3 Monolithically Coupled Formulation 65

6 Incompressible Flow **67**

6.1 Fluid Integration Scheme 67

6.2 Numerical Results 70

6.2.1 Two-dimensional Couette flow 70

6.2.2 Two-dimensional lid driven cavity 71

6.2.3 Two-dimensional moving vortex 76

6.2.4 Two-dimensional flow past a stationary circular cylinder . . . 77

6.2.5 Two-dimensional flow past a rotating elliptic cylinder 84

6.2.6 Two-dimensional flow past multiple rotating elliptic cylinders 86

6.2.7 Three-dimensional smoke jet past rotating ellipsoid 92

7 Fluid Structure Interaction **96**

7.1 Ghost Cell Computation and Advection 96

7.2 Coupled Solver 97

7.2.1 Preconditioning 98

7.3 Coupled Integration Scheme 98

8 Free Surface Flow **101**

8.1 Level Set Method 101

8.1.1 Particle Level Set Method 105

8.1.2 Pressure solver 107

8.2 Rendering 109

8.3 Results 109

9 Conclusion and Future Work **114**

Bibliography **117**

List of Tables

3.1 The order of accuracy of our ALE semi-Lagrangian (SL) and SL-MacCormack (SL-MC) methods on the constant uniform velocity test. n is the number of points along each axis on each grid. 35

3.2 The order of accuracy of our ALE semi-Lagrangian (SL) and SL-MacCormack (SL-MC) methods on the constant single vortex test when the bump has been rotated for one cycle. n is the number of points along each axis on each grid. 36

4.1 Domains, cell sizes, positions ($\vec{s}$) and orientations (θ) of the four grids used in our Poisson equation tests in two spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid. Note that all cells on all grids are square, the 3rd grid is rectangular with more square grid cells in one direction, and the 4th grid is extra fine where the numerator is correctly listed as .15 in the table. See also Figure 4.6. 52

4.2 Convergence results for solving a Poisson equation with analytic solution $\phi(x, y) = \sin(\pi x)\sin(\pi y)$. 52

4.3 Convergence results for solving a Poisson equation with analytic solution $\phi(x, y) = e^x(x^2 sin(x) + y^2)$. 52

4.4 Domains, cell sizes, positions and orientations (angle θ, axis $\vec{a}$) of the four grids used in our Poisson tests in three spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid. . 54

4.5 Convergence results for solving a Poisson equation with analytic solution $\phi(x, y, z) = sin(\pi x)sin(\pi y)sin(\pi z)$. 54

4.6 Convergence results for solving a Poisson equation with analytic solution $\phi(x, y, z) = e^{-x^2-y^2-z^2}$ 54

4.7 Convergence results for solving a Poisson equation with analytic solution $\phi(x, y, z) = e^x + e^y + e^z$ 54

4.8 The number of iterations taken by CG and ICPCG in order to converge for successively finer resolutions in both two and three dimensions. . . 55

5.1 Domains, cell sizes, positions ($\vec{s}$) and orientations (θ) of the four grids used in our diffusion equation tests in two spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid. . . . 58

5.2 Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t}sin(\pi x)sin(\pi y)$ on two stationary grids using backward Euler time integration. 58

5.3 Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t}sin(\pi x)sin(\pi y)$ on two stationary grids using trapezoid rule time integration. 58

5.4 Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t}sin(\pi x)sin(\pi y)$ on one stationary grid and one moving grid using semi-Lagrangian remapping and backward euler time integration. 60

5.5 Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t}sin(\pi x)sin(\pi y)$ on one stationary grid and one moving grid using SL-MacCormack remapping and trapezoid rule time integration. 61

5.6 Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t}sin(\pi x)sin(\pi y)$ on a moving grid using semi-Lagrangian remapping and trapezoid rule time integration. 61

5.7 Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ, axis $\vec{a}$) of the two grids used in our diffusion equation tests in three spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid. 61

5.8 Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, z, t) = e^{-.03\pi^2 t} sin(\pi x) sin(\pi y) sin(\pi z)$ on one stationary grid and one moving grid using SL-MacCormack remapping and trapezoid rule time integration. 62

5.9 Convergence results for solving a separate diffusion equation in each direction with analytic solution $\vec{\phi}(x, y, t) = (e^{-.02\pi^2 t} sin(\pi x) sin(\pi y), e^{-.13\pi^2 t} sin(2\pi x) sin(3\pi y))$ on one stationary and one moving grid using SL-MacCormack remapping and trapezoid rule time integration. 64

5.10 Convergence results for solving a separate diffusion equation in each direction with analytic solution $\vec{\phi}(x, y, t) = (e^{-.03\pi^2 t} sin(\pi x) sin(\pi y) sin(\pi z), e^{-.12\pi^2 t} sin(2\pi x) sin(2\pi y)$ on one stationary grid and one moving grid using SL-MacCormack remapping and trapezoid rule time integration. 64

5.11 Convergence results for solving a diffusion equation in each direction coupled in a monolithic system with analytic solution $\vec{\phi}(x, y, t) = (e^{-.02\pi^2 t} sin(\pi x) sin(\pi y), e^{-.12\pi^2 t} sin(2\pi x) sin(3\pi y))$ on two stationary grids using backward euler time integration. 66

6.1 Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the two grids used in our Couette flow and lid driven cavity tests. n indicates the number of cells in each dimension on the coarsest grid. In the case of the lid driven cavity test, for Reynolds numbers 100 and 400, $n = 128$ was used and for Reynolds numbers 1000 and 5000, $n = 256$ was used. Additionally for the case of the lid driven cavity test, during parallel simulation each grid remained unsubdivided and was allocated a single MPI process since the number of cells on each grid was the identical. 71

6.2 Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the two grids used in our vortex flow past grid boundary tests. n indicates the number of cells in each dimension on the coarsest grid. During parallel simulation each grid remained unsubdivided and was allocated a single MPI process since the number of cells on each grid was the identical. 77

6.3 Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the three grids used in our flow past a circular cylinder tests. n indicates the number of cells along the y-axis on the coarsest grid. During parallel simulation the background grid was allocated 24 processors, the fine grid enclosing the cylinder was allocated 16 processors, and the grid capturing the wake was allocated 24 processors. 82

6.4 The coefficients of drag and lift (C_D & C_L) and Strouhal numbers for varying Reynolds numbers computed using our method with all grids stationary as listed in Table 6.3 and $n = 512$. Note the good agreement of all values with those produced and cited by [51]. 84

6.5 The coefficients of drag and lift (C_D & C_L) and Strouhal number for Reynolds number 100 computed using our method where the rotation of the grid enclosing the cylinder is specified as listed in Table 6.3 and $n = 512$. Note that the values are close to those for the stationary case as listed in Table 6.4. 84

6.6 The coefficients of drag and lift (C_D & C_L) and Strouhal numbers for Reynolds number 100 computed using our method with all grids stationary as listed in Table 6.3 and varying resolutions. The coefficients of drag and lift and Strouhal numbers clearly tend towards the values produced and cited in [51]. Additionally, the Strouhal numbers clearly demonstrate first order convergence as the grids are refined. While the exact convergence regimes of the coefficients of drag and lift are less clear, the difference between the values at successive resolutions is decreasing. 85

6.7 Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the three grids used in our flow past a rotating elliptic cylinder example. n indicates the number of cells in x-axis on the coarsest grid. For all tests in this example $n = 256$. Note that the second grid encloses the elliptic cylinder and rotates with the cylinder at an angular velocity of $\pi/4$. During parallel simulation each of the three grids remains unsubdivided and was allocated a single MPI process since the number of cells on each grid were similar enough to not warrant any subdivision. 86

6.8 Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the six grids used in our flow past three rotating elliptic cylinders simulation where $n = 256$. Grids 3, 4 and 5 each enclose and track one of the elliptical cylinders as shown in Figures 6.12 and 6.13. During parallel simulation, a single MPI process was allocated to each grid enclosing a cylinder. Grid 2 was allocated two MPI processes and placed over the three cylinders in order to capture the interactions between them. Grid 6 was added to in order to capture the wake and was allocated two MPI processes. The background grid 1 was only allocated a single MPI process due to its relative low resolution. 91

6.9 Axis lengths, positions ($\vec{s}$) and orientations (angle θ) of the three elliptic cylinders used in our flow past three rotating elliptic cylinders example. Note that the long axis of each cylinder lies along the x axis in object space. 91

6.10 Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ, axis $\vec{a}$) of the three grids used in our three-dimensional smoke jet past rotating ellipsoid example. In this example we use $n = 256$. During parallel simulation we allocate 8 MPI processes to grid 1, 6 MPI processes to grid 2 and 10 MPI processes to grid 3 in order to load balance between two dual 6-core computers. 93

8.1 Timing data (in seconds) for the island and propeller examples. The table includes timing only for the major steps of the algorithm. . . . 110

List of Figures

1.1 A circle is surrounded by a body fitted grid on top of a Cartesian background grid. 3

1.2 A circle is discretized using block structured approaches by placing a number of overlapping Cartesian grid patches. (Left) In AMR approaches patches are restricted to being axis-aligned, requiring more patches to efficiently and accurately represent the structure boundary. 29 patches are used in this case. (Right) In our Chimera grid approach patches are allowed to be arbitrarily rotated and translated, allowing non-axis-aligned features to be rasterized much more efficiently. Only 8 patches are required in this case. 4

2.1 We store scalar fields ϕ at cell centers and vector fields (u, v) as scalar components in the normal direction at each face. Note that face normal directions are tied to the grid orientation and that when reconstructing a world space vector it is necessary to rotate the object space vector into world space (i.e. $\vec{u}_{\text{world}} = \mathbf{R}\vec{u}_{\text{object}}$ where $\vec{u}_{\text{object}} = (u, v)$). 12

2.2 The explicit grid coupling between two partially overlapping grids (red and green) and a blue background grid. The green grid is finer than the red grid, and the blue grid is the coarsest. For clarity, each grid has only one layer of ghost cells in this example. (Left) Ghost cells are filled with values interpolated from the finest overlapping grid with a valid interpolation stencil. The dotted cells are the ghost cells of the grids. The color of each dot indicates which grid its value is interpolated from. The blue background grid's ghost cells (marked with gray squares) have no data to interpolate from, thus will be filled according to the boundary condition. (Right) Overlapped interior cells are filled with values interpolated from a finer grid. Note that we fill grids from fine to coarse and use the finest grid with a valid interpolation stencil when interpolating values. The dotted cells are overlapped by finer grids. The color of each dot indicates which grid its value is interpolated from, and the fill color of a cell helps clarify which grid the cell belongs to. 15

2.3 Illustration of the lower-dimensional layer of overlapped cells. A fine red grid is shown overlapping a coarse blue grid. In this example, we only fill the shaded blue grid cells with data from the red grid and otherwise ignore the hollowed out interior region of the blue grid. Based on the stencils used to update the blue grid as well as the subsequent motion of the red grid, one can specify how large of an interior region can be hollowed out to reduce the communication from the red grid to the blue grid down to a lower-dimensional set for the purpose of communication optimization while still providing all the relevant data from the red to the blue grid. 16

2.4 Grids are split into subgrids and assigned to multiple processors. Grids with the same color correspond to a single grid before splitting. The number on each subgrid indicates the index of the corresponding processor. 17

3.1 The order of accuracy of a time-varying single vortex test in two spatial dimensions with the SL-MacCormack method. (Top) uses velocities at time t^n for advection and a fixed CFL number. (Middle) uses velocities at time $t^{n+1/2}$ for advection and a fixed CFL number. (Bottom) uses velocities at time t^n for advection and fixes Δt to the value used on a 4096-point grid. 22

3.2 An illustration of our ALE advection in the world space for the point $\vec{x}_{\text{object}}$ that moves from $\vec{x}^n_{\text{world}}$ (shown as the red point) to $\vec{x}^{n+1}_{\text{world}}$ (shown as the blue point) according to the grid motion. (Top) Forward advection. (Left) Backward advection for SL-MacCormack. 25

3.3 Illustration of the relation between the time step size and ghost cells. (Left) Forward advection. (Right) Backward advection for SL-MacCormack. The inner dashed boxes are the ghost domains bounding the grid motion—that is, the blue solid box should be completely inside the inner dashed red box, and in SL-MacCormack advection the red solid box also needs to be inside the inner blue dashed box. The outer dashed boxes are the ghost domains bounding the potential lookup points containing the values to be advected. 30

3.4 Illustration of motion of the grids. The background grid has no rotation or translation, while the fine grid spins and translates along a straight line inside the domain of the background grid. 33

3.5 Snapshots of a simulation that advects a circular bump in a constant uniform velocity field. The circles shown in the figure are contours of the bump function. (Left) The snapshot at $t = 0$. (Middle) The snapshot at $t = 1$. (Right) The snapshot at $t = 3$. 34

3.6 The L^1 norm error of the numerical solutions as a function of time in the constant uniform velocity test. (Top) The results of our ALE semi-Lagrangian method. (Bottom) The results of our ALE SL-MacCormack method. 35

3.7 The L^1 norm errors of the numerical solutions as a function of time in the constant single vortex test. (Top) The results of our ALE semi-Lagrangian method. (Bottom) The results of our ALE SL-MacCormack method. 36

3.8 The L^1 norm error of the numerical solutions as a function of time in the time-varying single vortex test. (Top) The results of our ALE semi-Lagrangian method. (Bottom) The results of our ALE SL-MacCormack method. 37

3.9 The order of accuracy of the time-varying single vortex test.(Top) The order of accuracy based on L^1 norm of errors. (Bottom) The order of accuracy based on L^∞ norm of the errors. 38

4.1 For fully overlapping grids one must omit a number of grid cells in the interior in order to provide boundaries for the application of boundary conditions, in this case creating an interior boundary, $\partial\Omega_b$, on the blue grid to receive information from the red grid. For partially overlapping grids one may omit cells for the sake of efficiency, but it is not always necessary in order to create a boundary on which one can prescribe coupling as in the case for fully overlapping grids. The black boundaries of both grids, $\partial\Omega_b$ and $\partial\Omega_r$, are locations on which Neumann boundary conditions would be applied and the dotted cells of both grids are locations on which Dirichlet boundary conditions would be applied. . 40

4.2 In one dimension, two grids overlap by 3 cells with the cell centered solution variables ϕ_1^1, ϕ_2^1 and ϕ_3^1 on the grid 1 and ϕ_1^2, ϕ_2^2 and ϕ_3^2 on the grid 2. In this case ϕ_3^1 and ϕ_1^2 are ghost cells where Dirichlet boundary conditions would be applied. The bold face lines between ϕ_2^1 and ϕ_3^1, and ϕ_1^2 and ϕ_2^2 represent where Neumann boundary conditions would be applied. 42

4.3 In this two grid example, we cut overlapped cells on the blue coarse grid by removing parts of these cells which are covered by finer red grid cells. The blue dots are the locations of ϕ samples on the blue grid. The red dots are the locations of ϕ samples on the fine red grid. The dashed lines indicate the direction of the components of the gradient across faces along the intergrid boundaries. Generally the direction of these components are far from orthogonal to their respective faces. The components of the gradients incident to the blue ϕ sample contained within the red grid (in the blown up portion of the figure to the right) are even inverted. 45

4.4 The composite grid resulting from our proposed method for coupling multiple grids by generating a Voronoi mesh along their boundaries. The dots are the cell centers of boundary cells that are incident to voronoi faces - note every dot corresponds to a degree of freedom originally from either the red or blue grid as indicated by their color. . . 46

4.5 Our procedure for generating the Voronoi face between cell i and its first neighbor cell 1. (a) The remaining Cartesian cell geometry after cutting out overlapping cells on the grids is drawn in blue for the coarse grid and red for the fine grid. The black dots indicate non-removed cells along the intergrid boundary with only partial geometry remaining after the cell removing procedure. The black line indicates the initial unclipped face between cells i and 1 and the dotted line indicates the orthogonal line between those cells' centers. (b)-(g) The face between cells i and 1 is sequentially clipped by the plane (the dashed lines) between cell i and its neighbor cells 2 to 7. The black line shows resulting face between cells 1 and i after each clipping operation. (h) Shows the final face in context with the complete Voronoi diagram (shown in green). 48

4.6 The domains of the grids used in our Poisson equation tests in two spatial dimensions. 51

4.7 The domains of the grids used in our Poisson equation tests in three spatial dimensions. 53

5.1 The domains of the grids used in our diffusion equation tests in 2 spacial dimensions with the second grid shown at its time $t = 0$, $t = .5$ and $t = 1$ positions and orientations. 57

5.2 When interpolating between cell centered values on the Voronoi mesh (drawn as blue and red faces which correspond to the Cartesian faces taken from the coarse and fine grids respectively, and green faces which correspond to unstructured Voronoi faces), we use a hybrid interpolation scheme. The interpolation mesh is drawn in black overtop the Voronoi mesh. If an interpolation location lies within a square on the interpolation mesh, bilinear interpolation is applied. Otherwise, if an interpolation location lies within a triangle of the Delaunay mesh dual of the unstructured part of the Voronoi diagram, barycentric interpolation over that triangle is applied. Note that in the former case, if we were using the full Delaunay mesh dual of the Voronoi diagram these squares would be tesselated with triangles and only barycentric interpolation would be used. However, this would reduce accuracy. . . 59

5.3 The domains of the grids used in our diffusion tests in three spatial dimensions with the second grid shown at its time $t = 0$, $t = .5$ and $t = 1$ positions and orientations. 62

6.1 The results for our Couette flow example using two grids of varying resolutions and when using only the single coarser grid at the finest resolution. The analytic solution is also drawn. (Top) shows the pressure along the horizontal plane through the geometry center of the domain. (Bottom) shows the u velocities on the vertical plane through the geometric center of the domain. Note that the single grid simulation produces values nearly identical to that of the analytic solution while the two grid simulation includes a truncation error that vanishes under refinement demonstrating first order accuracy. Note that although this example is intended to examine the behavior of a steady state solution, since the second grid is moving a time varying error is introduced and the above plots are given for the solution at $t = 41.6667$ at which point the solution is well into its steady state regime. The pressure and velocity profiles at different times within this regime demonstrate similar convergence behavior. 72

6.2 Streamlines for the lid driven cavity example. Notice the tiny vortices at the bottom right corners of the Reynolds number 1000 and 5000 simulations and at the bottom left corner of the Reynolds number 5000 simulation. These vortices are the same as reported by [34]. . . 73

6.3 The u velocities on the vertical plane through the geometric center of cavity. The red lines correspond to our results and the blue '+' symbols correspond to the results from [34]. Note the good agreement between the results, particularly for smaller Reynolds numbers. 74

6.4 The v velocities on the horizontal line through the geometric center of cavity. The red lines correspond to our results and the blue '+' symbols correspond to the results from [34]. Note the good agreement between the results, particularly for smaller Reynolds numbers. 75

6.5 Vorticity isocontours for the vortex flow across grid boundary example at $t = .20833$. Note that the isocontours match at the grid boundaries and no artifacts are visible along or further away from the boundaries. 78

6.6 The errors and orders of accuracy of velocities for the vortex flow across grid boundary example, with a stationary fine grid. Figures (a) and (c) show that the error tends towards zero in both the L^1 and L^∞ norms implying self convergence, whereas Figures (b) and (d) show that the error seems to be improving towards first order accuracy as the grid is refined. 79

6.7 The errors and orders of accuracy of velocities for the vortex flow across grid boundary example, with a rotating fine grid. Figures (a) and (c) show that the error tends towards zero in both the L^1 and L^∞ norms implying self convergence, whereas Figures (b) and (d) show that the error seems to be improving towards first order accuracy as the grid is refined. 80

6.8 Vorticity isocontours for the flow past a circular cylinder example with Reynolds number 200. 82

6.9 Pressure contours for the flow past a circular cylinder example taken when the coefficient of lift was at its most negative value. Note that (d) agrees with the Reynolds number 200 pressure plots from [51] and that the Reynolds number 100 pressure plots for the (a) stationary grid and (b) rotating grid are nearly identical. 83

6.10 Vorticity isocontours for the flow past an elliptic cylinder example. Note that the elliptical cylinder and the grid attached to it are rotating with angular velocity $\pi/4$. 87

6.11 The error and order of accuracy of the velocities in the flow past rotating elliptic cylinder example. Figures (a) and (c) show that the error tends towards zero in both the L^1 and L^∞ norms implying self convergence. Figure (b) shows that the L^1 error tends towards first order accuracy and (d) shows that the L^∞ error tends towards half order accuracy under refinement. 88

6.12 Vorticity isocontours at time $t = 5.3333$ for the flow past multiple elliptic cylinders example. 89

6.13 Vorticity at time $t = 18.958$ for the flow past multiple elliptic cylinders example. 90

6.14 The passive scalar rendered as smoke for the three dimensional smoke jet past rotating ellipsoid example. 94

6.15 The passive scalar rendered as smoke at $t = 13.958$ for the three dimensional smoke jet past rotating ellipsoid example. 95

7.1 Resolving a three dimensional glider (yellow) with a number of Chimera grids gives it the ability to fly in a large domain. The solid line shows the parabolic path a ballistic rigid body would take if it did not fly. The blue glider is simulated with only the background grid and falls almost immediately. Meanwhile, the red and green gliders each use only two of the six grids attached to the yellow glider. Although the green glider crashes, for illustration purposes, we can carefully adjusted the position and orientation of the red glider's two grids allowing it to fly. Note that this hand-picked rasterization will remain constant throughout the entire simulation since the grids move with the glider enabling the user to handcraft efficient rasterizations that improve the physical realism. We emphasize to the reader that we did not specially tune the positions of the grids in the case of the yellow glider as it was well resolved by its finer grids. Slight perturbations to the the placement of its grids did not alter the glider's behavior. 100

8.1 A single water drop impacts a flat water surface generating several circular waves. The large domain allows these waves to propagate a long distance without reaching the domain boundaries. Note that the waves pass from one grid to the next without visible artifacts. The bottom right figure is the same as the bottom left except that we zoom the camera back even further in order to show the computational domain and the three grids used in the simulation. 102

8.2 Many water droplets in the shape of armadillos are dropped into still water. Each armadillo is enclosed by two moving grids: one tightly wrapped to resolve the armadillo as it falls and one slightly larger to catch the resulting splash. Our Chimera grid strategy allows detailed resolution of the small armadillo droplets even in this large domain. . 103

8.3 A free surface is shown on two grids as a black line with the filled cyan area indicating the fluid phase. When reintializing the level set after advection and extrapolating the velocities across the interface after the pressure solve and velocity update, values are computed for only a limited band of cells around the interface. The limits of these bands are drawn as dashed lines (with the color indicating the corresponding grid). Note that the band is proportionally smaller on the fine grid. (Left) When interpolating values for ghost and overlapped cells invalid values can be produced. Since the cell on the coarse grid indicated by the black dot if beyond the band around the interface on the fine grid an incorrect value is computed. An additional reinitilization step is used to compute new valid values. (Right) In order to compute a second order accurate pressure discretization on the free surface, the ϕ values for cells on either side of the interface are used to compute the dual cell volume fraction. Due to the unstructured connectivity between grids on the Vonoroi mesh, cells on coarse grids along the interface may be adjacent to cells on the fine grid outside of the reinitialization bandwidth as illustrated by the two cells with black dots at their cell centers. We instead compute an updated ϕ value for the cell on the fine grid by linearly extrapolating from the cell on the coarse grid. . . 104

8.4 A kinematically driven grid moves in and out of still water without disturbing the surface. Throughout the entire simulation the water velocity properly remains within rounding tolerance of zero. The left figure shows the initial state and the right figure shows the water surface unchanged after 600 frames. 106

8.5 A kinematically driven propeller spins with a number of attached Chimera grids. The grids move and rotate with the rigid body frame of the propeller. 108

8.6 Pouring water drives a water wheel via two-way solid-fluid coupling. The domain around the water wheel buckets is resolved by eight grids that rotate with the wheel in order to resolve the thin bucket walls. In addition, a single larger grid encases the entire wheel and also rotates in the same rigid body frame as the wheel. 110

8.7 The top two figures show a two-way coupled rigid body ship that moves under its own power due to the rotation of two articulated propellers. The lower left figure illustrates that removing the grids surrounding the propellers limits the realism of the simulation resulting in a ship that churns water but cannot propel itself forward. The lower right figure shows that removing the grids around the ship results in a loss of buoyancy and the ship falls through the large grid sinking to the bottom of the ocean. 111

8.8 Two ships in stormy seas near Longfellow island. We refine the domain near the ships by placing grids in their object spaces to add detail and allow them to propel themselves using their two-way solid-fluid coupled propellers. 112

8.9 Another view from the large scale island example in Figure 8.8. Each ship is driven by two propellers and uses the same attached grids as in the case of the single ship example in Figure 8.7. We initialize the fluid surface and velocity with a large number of rolling waves which slowly move across the domain. As a result of the large domain, the simulation does not require seeding additional waves using boundary conditions or external forces within the time scale of the simulation. The top two figures show the spray produced by the ship propellers when they are exposed on the surface of the water, noting that both figures are of the same frame. The lower two figures show the initial grid placement around the ships and island. 113

Chapter 1

Introduction

Adaptive discretizations are important in many incompressible flow problems since it is often necessary to resolve details on multiple length and time scales. In fluid structure interaction problems it is critical to resolve the effects of boundary layers and the turbulent flows in the wakes behind objects in order to accurately predict the effect of the fluid on the structure. Many of these problems also incorporate large domains away from the structure and region of interest, thus requiring the accurate modeling of far field boundary conditions. This necessitates a method that allows large regions of space to be modeled using a reduced number of degrees of freedom. Much of this work was inspired by the large scale ship modeling performed in [102, 15, 108]. This work is exemplar of the complicated problems we are interested in solving due to the free surface viscous incompressible flow being modeled in order to capture a wide range of details such as a ship's breaking bow waves and bubbles entrained in its wake.

There are a wide variety of methods for adaptively discretizing space. Unstructured methods include both mesh based methods which use topologically connected meshes constructed with tetrahedra (see e.g. [58, 57, 76]), hexahedra and other irregularly shaped (see e.g. [27, 72, 14, 53]) and non-linear elements, as well as meshless methods which use disjoint particles such as Smoothed Particle Hydrodynamics (SPH)

(see e.g. [35, 71, 101, 24, 22]) and the Moving-Particle Semi-Implicit method (see e.g. [59, 109]). Many of these methods are also mirrored in the computer graphics literature (see e.g. [32, 56] for tetrahedral based methods and [25, 79, 86] for particle based methods). While these methods allow for conceptually simple adaptivity they often produce inaccurate numerical derivatives due to poorly conditioned elements. Dynamic remeshing, such as that used in many Arbitrary Lagrangian Eulerian (ALE) schemes (see e.g. [106, 48]), can be used to control the conditioning of elements during simulation. However, in addition to the significant added computational cost and complexity, these methods tend to introduce significant numerical dissipation if values need to be remapped (see e.g. [74, 73, 69, 70, 60]). In order to avoid these issues, many methods combining unstructured and structured methods have been developed. The two most well known are the Particle-In-Cell method (PIC) (see [39]) and the Fluid-Implicit Particle method (FLIP) (see [11, 10]) which use Lagrangian particles for advection and a background grid to solve for pressure. Other authors have explored using structured and unstructured methods to model different flow regimes within the same simulation, such as by combining SPH and Cartesian grid solvers (see e.g. [68]). While there is a wealth of literature addressing the mathematical issues of unstructured methods with regards to remeshing and computing accurate high order numerical stencils, the computational complexity and cost of these methods can still weigh heavily against the level of adaptivity afforded. For example, due to the unstructured nature of these discretizations it is generally not possible to store data in a cache coherent memory layout. Instead, pointer structures, are often used to store values incurring a suprisingly large computational expense due to the high number of indirections during traversal and the resulting increase in cache misses. Furthermore, in parallel computing environments, finding and maintaining (due to remeshing) a domain decomposition that evenly distributes the computational work load and storage requirements can cost as much as or even more than the time integration procedure.

Alternatively, structured methods predictably place degrees of freedom allowing for accurate and simple finite difference schemes, light weight cache coherent data structures and straight forward domain decomposition. Despite their lack of adaptivity,

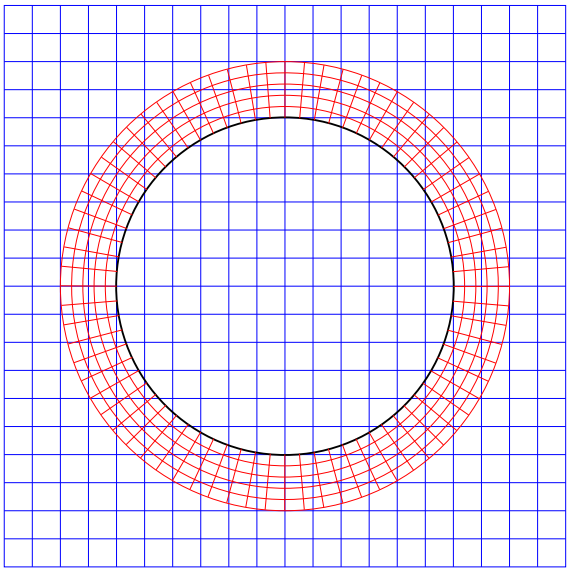


Figure 1.1: A circle is surrounded by a body fitted grid on top of a Cartesian background grid.

Cartesian grids have often outperformed adaptive methods even at high resolutions due to their simple and accurate numerical stencils as well as their regular layout of data in memory allowing for fast traversal. Often the most effective methods are structured methods tailored to specific problems, such as manually generated curvilinear grids (see e.g. [19, 15, 108, 33]) where where logically rectangular grids are parametrically deformed to conform closely to the solid interface as illustrated in Figure 1.1. Structured methods allow for many times the number of degrees of freedom to be used when compared to even the most efficient adaptive schemes, at the same computational cost. In order to exploit the efficiency of structured methods, authors have explored directly adding adaptivity to structured discretizations through methods such as octrees (see e.g. [67, 66, 17, 77, 85]), and adaptive mesh refinement (AMR) (see e.g. [9, 8, 65, 3, 100, 75, 23, 83, 92, 52]). The graphics community has also explored lattice based tetrahedral methods such as [78, 61, 18, 4]. While allowing for similar levels of adaptivity as unstructured methods, octrees and similar hierarchical structures suffer from the similar issues of cache coherency and domain decomposition, even if care is taken to maximize cache coherency and minimize indirections.

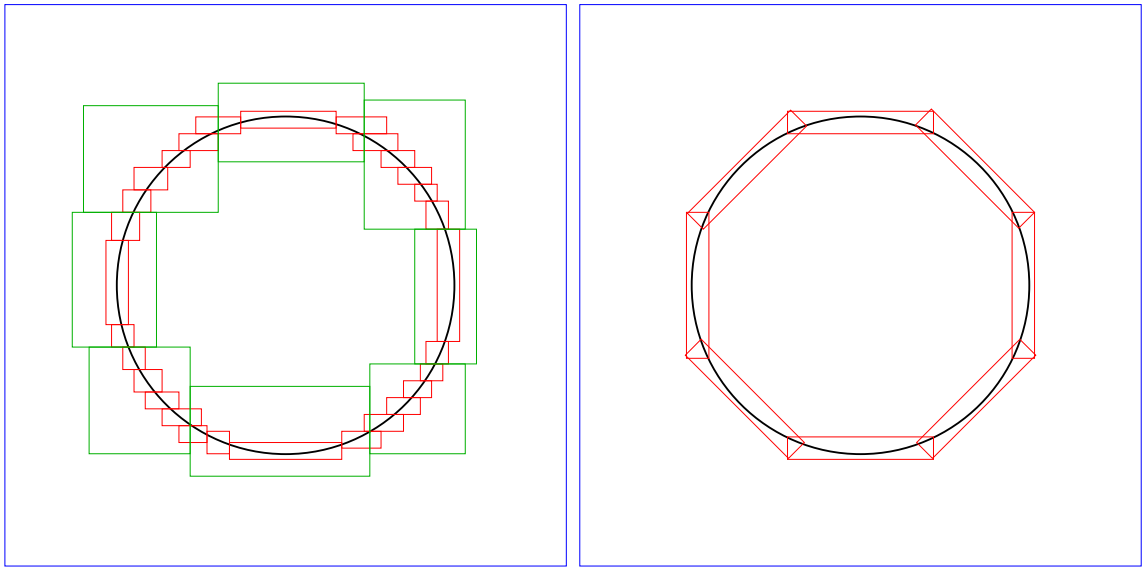


Figure 1.2: A circle is discretized using block structured approaches by placing a number of overlapping Cartesian grid patches. (Left) In AMR approaches patches are restricted to being axis-aligned, requiring more patches to efficiently and accurately represent the structure boundary. 29 patches are used in this case. (Right) In our Chimera grid approach patches are allowed to be arbitrarily rotated and translated, allowing non-axis-aligned features to be rasterized much more efficiently. Only 8 patches are required in this case.

Block structured AMR methods allow multiple Cartesian grids to be patched upon one another in order to allow higher resolution grids to represents parts of the domain with fine details, while the majority of the domain is covered by a single coarse grid. As a result AMR methods have been extremely successful due to their block Cartesian grid structure resulting in improved cache coherency and low cost decomposition, if the number of patches and repatching frequency are small enough, while providing for spatial adaptivity. In [9] grids were allowed to be both translated and rotated allowing accurate tracking of solid boundaries and flow features. However, in subsequent work (see e.g. [8]) grids were constrained to lie along the same axes and have coarse grid lines match up with fine grid lines along patch boundaries, thus simplifying the construction of computational stencils. In order to capture non-grid aligned features this then forces one to either sacrifice the efficiency of the method by requiring either the use of large fine grids to cover these features unnecessarily refining space far away from these features, or the creation of many small grids essentially rasterizing these features resulting in problems similar to those of hierarchical structured methods (e.g. octrees) as illustrated in Figure 1.2 (Right). Many modern AMR schemes are fully unstructured and allow for cell-wise refinement. In this case space-filling curves (see e.g. [2]) can be used to improve cache coherency and simplify domain decomposition. However, they still incur a significant computational overhead when compared to block structured approaches and can have high communication costs when repartitioning the domain. Chimera grid methods (see e.g. [7, 99, 5, 6, 98, 55]) also rely upon building adaptive discretizations by patching together independent grids. Unlike standard AMR, Chimera grids are more general allowing for many types of grids (such as Cartesian, curvilinear and deforming) to be rotated and moved while being used together to decompose a single domain without explicitly constructing an global mesh with regular connectivity as illustrated in Figure 1.2 (Right). This has allowed Chimera grid methods to be extremely effective in practical engineering applications. In fact, they were originally designed to simulate the compressible transonic flow regimes around the space shuttle using the Overflow solver [7]. Chimera grid methods have also been combined with AMR methods allowing for block structured adaptivity on each component grid [44, 45, 46]. Chimera style approaches have

also been explored in the graphics community (see e.g. [89, 96, 82, 26, 103, 21, 36]). However, each previous approach has been limited in either generality or scalability when compared to a full-blown Chimera grid scheme.

In Chimera grid schemes, the solution on different grids is coupled by exchanging boundary conditions. For explicit operations such as advection this is typically achieved by interpolating ghost cell values from overlapping grids before running the single grid code on each grid independently. Note that compared to certain unstructured and structured adaptive methods, this means that we perform some duplicate computation in regions overlapped by more than one grid. However, assuming a reasonable grading and placement of grids, for most operations the overall cost of changing the method to avoid computing the solution in overlapped cells would exceed that of simply computing the solution throughout the entire domain. This is exacerbated by the domain decomposition used for parallelization. Furthermore, by operating on the entire grid we can exploit the single grid implementations of certain operations without modification, significantly reducing the implementation effort. When implicitly solving for stiff terms such as viscous or pressure forces, this process is typically iterated in Gauss-Seidel fashion until converged in order to strongly couple the solution. This partitioned coupling often suffers from slow convergence and in certain cases the individual systems can even be singular as reported by [13, 47] which can cause the solver to diverge. Certain authors (see e.g. [40, 41]) have instead directly substituted the interpolation stencils used to fill ghost and overlapped cells directly into the system matrix in order to exploit more robust solvers. While this avoids many of the convergence issues of Gauss-Seidel approaches, the resulting system is asymmetric and can be expensive to solve robustly. Despite this, many efficient methods have developed to solve these system such as multigrid methods [43, 42] and approximate factored schemes [50]. Coupling together overlapping grids has received significant attention within the literature (see e.g. [87, 88]). One popular approach to implicitly coupling grids together are cut cell methods (see e.g. [107]) which alter the cells on each grid by removing the parts of these cells overlapped by finer grids. In order to build divergence operators these methods used a finite volume approach

to compute divergence. Computing the gradient at cell faces is considerably more problematic due to the creation of faces which are not orthogonal to the line between the pressure samples at incident cells. As a result, using the pressure difference between the samples located at incident cell centers can introduce errors which do not vanish as the grid is refined. Deferred correction methods (see e.g. [54, 105]) apply an iterative process in order to compute the correct pressure gradient at faces by reconstructing the full gradient at faces using the values computed in the previous time step or iteration. However, these methods do not guarantee the existence of a unique solution, and as a result can be extremely slow to converge to the correct solution if at all. [16] addressed this problem by including the stencil for reconstructing the full gradient in the system matrix, however this results in an asymmetric and possibly indefinite matrix with complicated stencils.

We propose a Chimera grid method which combines multiple moving and arbitrarily oriented Cartesian grids into a single computational domain. These grids are allowed to move both kinematically and as dynamic elements driven by the flow or attached to immersed rigid bodies. In order to couple together the solution at grid boundaries and in overlapped regions we compute values for ghost cells and cells in overlapped regions by interpolating values from finer overlapping grids as described in Section 2.2. In Section 3.2 we introduce both first and second order accurate ALE advection methods built using the semi-Lagrangian tracing of rays backwards and forwards in time in order to both avoid requiring an expensive time step stability restriction and to remap values in a single step without introducing additional numerical dissipation. In order to efficiently advect values and exploit Cartesian grid data structures, our ALE advection scheme first constructs a velocity field on each grid taking into account the grid's motion and then applies the single grid advection code. Since this process relies on interpolating values from the local grid at locations outside of each grid's time t^n and time t^{n+1} domains we use a fixed number of ghost cells. This imposes a loose time step restriction based both upon the fluid velocity and grid motion which we discuss in Section 3.3.

In order to implicitly solve for pressure and viscous forces we introduce a new spatial

discretization of the Laplacian operator in Section 4.3 where a Voronoi diagram (see e.g. [97, 12, 81]) is used to determine the connectivity of the pressure samples and the corresponding face areas used in the stencils along intergrid boundaries. By using a Voronoi diagram to define the cell geometry we are guaranteed orthogonal centered finite differences at faces, allowing us to exactly satisfy hydrostatic problems and compute pressure values to second order accuracy. We build the Voronoi diagram by considering pairs of cells along the intergrid boundary and computing the geometry of the face incident to each of these pairs by clipping a candidate plane by the candidate planes formed between nearby cell pairs. By directly computing the Voronoi diagram we avoid the issues of robustness and efficiency associated with methods which compute the Delaunay triangulation before computing the dual Voronoi diagram. We note that while our method does not produce the complete connectivity information, it is only necessary to compute the face areas and corresponding cell adjacency information in order to define the stencils in our discretization. As a result our method is robust to perturbations in the positions of degrees of freedom. Furthermore by computing a continuous discretization as opposed to the overlapped discretizations and coupling methods of previous Chimera grid schemes, the resulting linear system is symmetric positive definite allowing us to apply efficient linear solvers such as preconditioned conjugate gradient. We apply our discretization to several Poisson equation examples in Section 4.4. We then extend our discretization to solve diffusion equations on the cell centers of moving grids in Section 5.1. In order to solve for viscous forces on the staggered face velocity degrees of freedom we compute cell center velocities and then solve a diffusion equation independently in each direction using the cell center formulation before averaging the differences back to the original face degrees of freedom in order to update the original staggered degrees of freedom in Section 5.2. In both the viscosity and pressure solves we address object handling by using an immersed boundary approach by specifying the velocity at overlapped cells and faces.

We summarize our method for single-phase incompressible flow, including interaction with static and kinematic objects in Chapter 6.1. Numerical results are provided in

Section 6.2 including a Couette flow example demonstrating converge towards the analytic solution, a lid driven cavity example showing similar velocity profiles and vortex patterns as [34], an example with a two-dimensional vortex flowing from a fine grid to a coarse grid demonstrating self convergence, a two-dimensional flow past a stationary circular cylinder example showing the correct drag coefficients, lift coefficients and Strouhal numbers as compared to those published and cited in [51], and a flow past a rotating elliptic cylinder example showing self convergence for the case with a rotating solid and attached grid. We also include a more complicated example in two dimensions with three rotating elliptic cylinders, and one three-dimensional example with a rotating ellipsoid in order to emphasize the simplicity and feasibility of our approach.

In Chapter 7 we extend our pressure solver to support monolithic dynamic bodies and two-way fluid structure interaction problems. Since the exact motion of dynamics bodies is not known when computing the time step length at the beginning of each step, we approximate the velocity of each body when computing the time step and dynamically adjust the number of ghost cells during advection as discussed in Section 7.1. In order to allow for two way interaction between structures and the fluid we take the same approach as [90] by modifying our immersed boundary approach to include both velocity compatibility constraints at fluid faces along the interface, and Lagrange multipliers terms in the momentum update equations to enforce those constraints and ensure momentum conservation. We demonstrate the results by encompassing a glider with a number of grids in an extended background grid and examine its glide path under several grid configurations in Figure 7.1.

While level sets have been successfully used to track free surfaces on a number of adaptive discretizations (see e.g. [100, 66, 102, 4]), special care is necessary in order to avoid artifacts due to the sensitive nature of free surfaces and inconsistent level set and velocity representations in overlapping regions. Sections 8.1 and 8.1.1 describe our approach for adapting the particle level set approach of [29] to overlapping grids. On each grid we store both a level set and a collection of particles. The level set is

advected using our first order accurate ALE semi-Lagrangian approach and reintialized using the fast marching method independently on each grid after filling the ghost cells. Like other level set approaches, we reinitialize the level set and extrapolate the velocity across the free surface only in a limited number of layers of cells along the free surface interface due to the use of a limited reinitialization bandwidth. We address these issues by taking several additional reinitialization and extrapolation steps when necessary. In order to render the free surface we use a ray tracing approach in which the intersection with the free surface is found using a root finding procedure. In order to combine the overlapping level sets into a seamless surface, at each sample location we blend together the level set values from each overlapping grid into a continuous function. The root finding procedure is then directly applied to this composite function creating a smooth surface. We demonstrate the final results of this dissertation, including a number of two-way coupled free surface examples and timing information, in Section 8.3.

Chimera grids are extremely flexible and provide a convenient way to track flow features such as vortices and thin splashes as well as solid objects. We generally use a large background grid as can be seen in the bottom right of Figure 8.1. This gives a large simulation domain allowing waves to travel a long distance without hitting or reflecting off of the domain boundaries. In addition, we place smaller and finer grids near interesting features such as the drops and splashes in Figures 8.1 and 8.2. Note that in Figure 8.2 the grids follow the armadillo drops to track and preserve their shape and detail. Chimera grids are also used to follow and track rigid objects as can be seen in Figure 8.5. The efficacy of our approach is further illustrated in Figures 8.6, 7.1, and 8.7, which show the resolution of thin features such as the buckets of a water wheel, the wings and tail of a glider, and the propellers on a ship.

Chapter 2

Overlapping Grids

2.1 Cartesian Grids

Our Chimera grid simulation framework consists of a collection of grids that partition the simulated domain into regions of interest as shown in Figure 2.2. In this paper we consider only Cartesian grids undergoing rigid motion as described by a rigid frame consisting of a translation and a rotation. We represent a rigid transformation as the combination of a rotation using a rotation matrix, $\mathbf{R}$, and a translation vector, $\vec{s}$. Using these representations we relate locations and vectors in world space to those in a grid's object space using the equations

$$\vec{x}_{\text{world}} = \mathbf{R}\vec{x}_{\text{object}} + \vec{s} \tag{2.1}$$

$$\vec{u}_{\text{world}} = \mathbf{R}\vec{u}_{\text{object}} \tag{2.2}$$

Similar to other rigid body dynamics implementations we internally store the orientation of each grid as a unit quaternion. However, for the purposes of this exposition it is simpler to work with rotation matrices. In order for fine resolution grids to exactly follow the motion of objects in the fluid flow, we allow grids' rigid frames to be pinned to the transformations of their respective rigid bodies. We also allow grids to

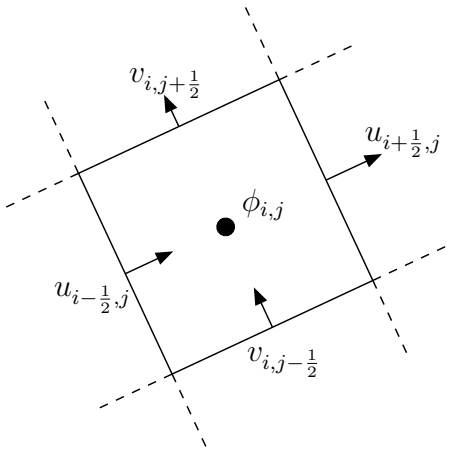


Figure 2.1: We store scalar fields ϕ at cell centers and vector fields (u, v) as scalar components in the normal direction at each face. Note that face normal directions are tied to the grid orientation and that when reconstructing a world space vector it is necessary to rotate the object space vector into world space (i.e. $\vec{u}_{\text{world}} = \mathbf{R}\vec{u}_{\text{object}}$ where $\vec{u}_{\text{object}} = (u, v)$).

be kinematically driven to follow flow features depending upon the problem.

In order to exploit existing single-grid code, our framework stores the original single-grid data structures for each grid in its own container. For each grid, these include a set of arrays storing values lying at cell or face centers, including quantities such as density, pressure and velocity. Also included in this container is the structural information for the associated grid, including the Cartesian grid parameters, (the grid's domain and cell counts in each dimension) as well as the grid's rigid frame. We represent field quantities in object space using a Marker-and-Cell (MAC) representation [38] as illustrated in Figure 2.1. Scalar quantities are represented as cell center samples and vector quantities are represented as a staggered arrangement of scalar samples stored at face centers with each sample representing the component of the vector field in the world space normal direction of the corresponding face. While scalar field samples can be used directly since they are invariant to the orientation of the grid, vector quantities must be rotated into world space. Thus, In order to

construct a full velocity vector in world space, we apply the regular interpolation scheme in the grid's object space and then transform the vector to world space by using Equation 2.2 to rotate the vector. This means that if a grid which was originally aligned with the world space axes is rotated by 90 degrees, all the x-velocity MAC grid faces would have to be switched to the y-velocity MAC grid faces–more generally, one needs to be careful to account for the fact that the components of velocity stored on faces change as the grid rotates (but are not affected by translation).

We note that while in certain parts of the algorithm in Sections 5 and 6 we exploit cell centered velocities to simplify interpolation and to allow us to decouple the velocity components during the viscous solve, the persistent velocity representation is staggered. A staggered arrangement is necessary due to the centered differencing in the gradient, divergence and Laplacian operators used in the Poisson equation discretization. It is well known that neighboring pressures and velocities become decoupled if a collocated velocity representation is used in combination with centered differencing. This decoupling can lead to instability and is avoided by using a MAC representation.

2.2 Explicit Grid Coupling

While our method applies fully implicit coupling to compute the pressure and viscosity terms across multiple grids, an explicit coupling scheme is suitable for operations that do not involve global communication/coupling such as our semi-Lagrangian advection scheme (noting that one could alternatively use an implicit scheme for advection in which case a more strongly coupled approach would be necessary or that one could solve the diffusion equation explicitly in which case an explicit coupling approach would be sufficient). Grid coupling most naturally occurs near the exterior boundary of a grid where computational stencils reach across the boundary and require information available on one of the other grids. Although one could simply interpolate the required values from the appropriate grid when evaluating numerical stencils, this approach typically leads to increased code complexity and issues with cache coherency,

thus increasing computational cost.

Instead, in our method each grid is allocated a band of ghost cells surrounding its domain. By filling these ghost cells with valid data from other grids that overlap these cells as shown in Figure 2.2 (Left), we can proceed to perform operations on each grid independent of the other grids, which is typical of block structured AMR approaches. We regard a cell/face as overlapped by another grid if and only if it lies inside the grid's interpolation domain, i.e. a valid interpolation stencil exists in the grid domain. For example, in the case of linear interpolation, a cell or face center has to be at least a distance of $\Delta x/2$ inside the grid domain boundaries to be considered as overlapped by that grid–this means that it is inside the rectangle created by the four (in two spatial dimensions, or eight in three spatial dimensions) neighboring cell centers on the grid that we are interpolating from (although we have not implemented it, in the special case of aligned grids, this $\Delta x/2$ restriction can be relaxed for face centers in the dimension parallel to the face normal). When multiple grids overlap a given ghost cell, we always interpolate from the finest overlapping grid (see Figure 2.2 (Left)). The number of ghost cells for each grid is determined by considering both the stencils used by the operators being applied to each grid, and the relative motions of the grids. The method for deciding the number of ghost cells is detailed in Section 3.3.

While the use of ghost cells and an appropriate time step restriction provides each grid with access to all the values they need to evaluate numerical stencils, tests showed that values on the interiors of overlapped grid regions could tend to gradually drift apart. Moreover, in the case of a fine grid completely contained within a coarse grid, the values calculated on the overlapping fine grid would never feed back into the coarse grid, unless the fine grid is moved to overlap the ghost cells of the coarse grid. We resolve this issue by replacing the value of every overlapped coarse cell by a value interpolated from a finer grid that overlaps it (using the finest grid possible), as shown in Figure 2.2 (Right). Note that this can incur a large communication cost when using MPI which can be resolved without affecting the solution by only filling a band of these cells near non-overlapped cells (see Figure 2.3) as is done in typical Chimera grid methods. When filling overlapped cells, the order in which the grids are

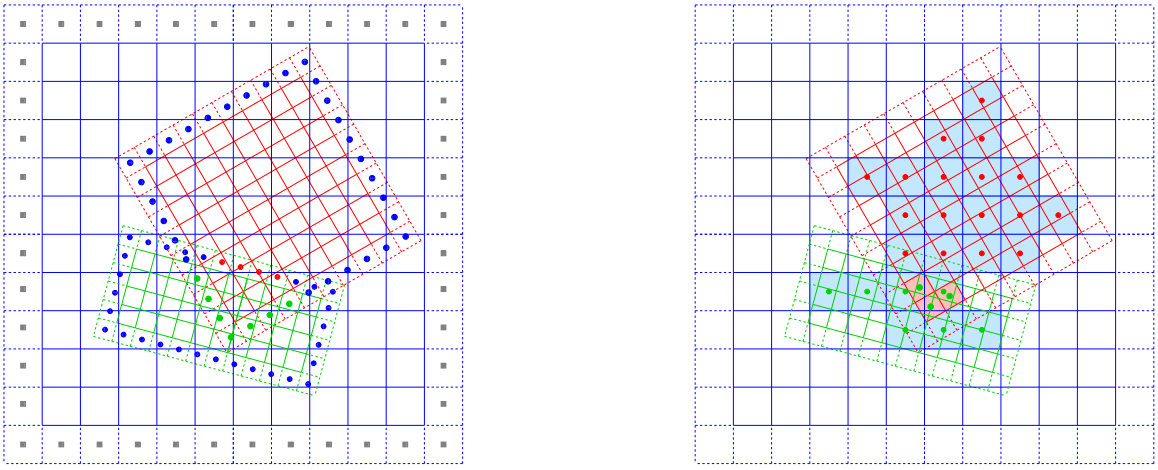


Figure 2.2: The explicit grid coupling between two partially overlapping grids (red and green) and a blue background grid. The green grid is finer than the red grid, and the blue grid is the coarsest. For clarity, each grid has only one layer of ghost cells in this example. (Left) Ghost cells are filled with values interpolated from the finest overlapping grid with a valid interpolation stencil. The dotted cells are the ghost cells of the grids. The color of each dot indicates which grid its value is interpolated from. The blue background grid's ghost cells (marked with gray squares) have no data to interpolate from, thus will be filled according to the boundary condition. (Right) Overlapped interior cells are filled with values interpolated from a finer grid. Note that we fill grids from fine to coarse and use the finest grid with a valid interpolation stencil when interpolating values. The dotted cells are overlapped by finer grids. The color of each dot indicates which grid its value is interpolated from, and the fill color of a cell helps clarify which grid the cell belongs to.

filled affects the final outcome since we use the most recent values when interpolating. Hence, we fill grids from fine to coarse as illustrated in Figure 2.2 (Right).

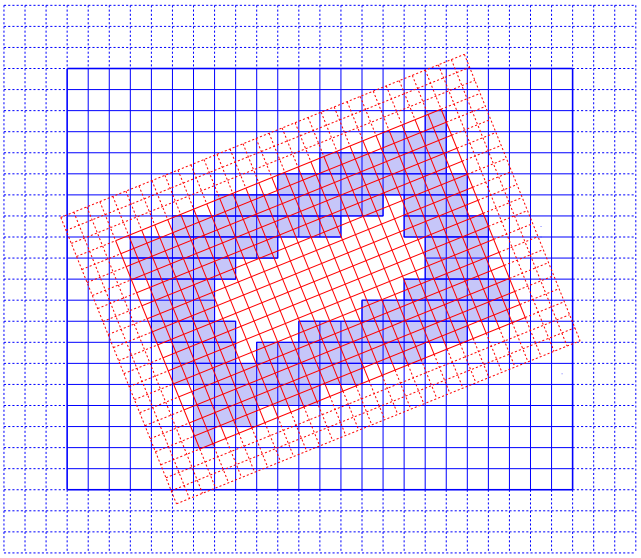


Figure 2.3: Illustration of the lower-dimensional layer of overlapped cells. A fine red grid is shown overlapping a coarse blue grid. In this example, we only fill the shaded blue grid cells with data from the red grid and otherwise ignore the hollowed out interior region of the blue grid. Based on the stencils used to update the blue grid as well as the subsequent motion of the red grid, one can specify how large of an interior region can be hollowed out to reduce the communication from the red grid to the blue grid down to a lower-dimensional set for the purpose of communication optimization while still providing all the relevant data from the red to the blue grid.

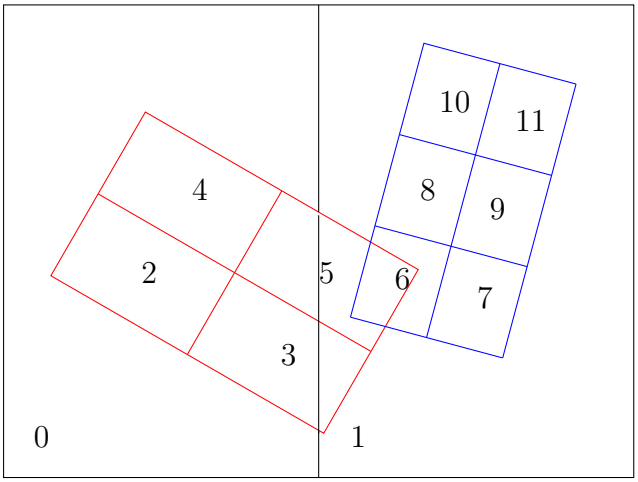


Figure 2.4: Grids are split into subgrids and assigned to multiple processors. Grids with the same color correspond to a single grid before splitting. The number on each subgrid indicates the index of the corresponding processor.

2.3 Parallelization

Once the ghost cells on each grid have been filled, the explicit operation can be performed on each grid independent of other grids. This makes it feasible to parallelize our algorithm using MPI by assigning each grid and its associated data container to a distinct processor, where the communication between processors is only necessary when filling ghost cells and filling overlapped regions. While data is distributed and consequently each grid's associated data container is only visible to one processor, the structural information (rigid body frames, domain sizes and cell sizes) for all the grids is stored identically in every process. This redundantly stored information is very lightweight and adds only a negligible increase in memory. However, it conveniently informs every process of the entire domain decomposition so that each processor can readily decide which other processes it sends to and receives from.

In order to make full use of computational resources, our software takes an approach similar to that used in [46] and implements a procedure to split grids in order to balance the number of spatial degrees of freedom in each process. One example of how grids are split is given in Figure 2.4. Each of the 12 subgrids in Figure 2.4 behaves

the same as a standalone grid except for the following differences. The ghost cells along splitting boundaries between subgrids are filled *first* using simple injection as opposed to interpolation, noting that simple injection (as opposed to interpolation) is sufficient since the ghost cells are collocated with real cells from the adjacent subgrids. This means that each of the black, red and blue composite grids in Figure 2.4 will first sync up among its own subgrids. After that the algorithm can proceed as if no grid splitting took place, instead only dealing with the composite black, red and blue grids. One caveat is that when interpolating from a grid that is divided into subgrids, a cell that lies near the subgrid boundaries could interpolate from any of the adjacent subgrids–however one obtains the same answer regardless of which subgrid is used.

Chapter 3

Generalized Semi-Lagrangian Advection

We consider both first order accurate semi-Lagrangian advection as well as second order accurate semi-Lagrangian style MacCormack (SL-MacCormack) advection as introduced in [94]. We have chosen to apply these method-of-characteristics type approaches to exploit their unconditional stability in order to avoid the strict time step restrictions which can be imposed by very small cells on fine grids. The coupling between grids has been addressed in Section 2.2 allowing advection to be performed independently on each grid. In order to account for the effect of each grid's motion while exploiting existing single-grid implementations, we have implemented a wrapper function which transforms the velocities, used to advect values, from world space to each grid's object space. This is detailed in Section 3.2.

3.1 Semi-Lagrangian MacCormack Advection

Before presenting our Chimera grid advection approach we discuss a few aspects of the SL-MacCormack method of [94], in order to clarify a few important details not

discussed in the original work. We consider the advection equation

$$\frac{\partial \phi}{\partial t} + \vec{u} \cdot \nabla \phi = 0 \tag{3.1}$$

where ϕ is a scalar quantity and $\vec{u}$ is a divergence free velocity field. In first order accurate semi-Lagrangian advection, characteristic paths are traced backwards in time to find locations from which to interpolate new values. Equation 3.1 is discretized using this approach as follows:

$$\phi^{n+1}(\vec{x}) = \phi^n(\vec{x} - \Delta t \vec{u}^n(\vec{x})) \tag{3.2}$$

where ϕ^{n+1} are the updated time t^{n+1} values, ϕ^n are the time t^n values, and $\vec{u}$ is the velocity field.

The SL-MacCormack method is then built using this first order accurate scheme to advect values forward and backward in time during each time step in order to estimate the advection error. It is assumed that in both of these steps, approximately the same error is added to the resulting values. Thus, we advect the time t^n values $\phi^n(\vec{x})$ forward in time to obtain the temporary forward advected values as follows:

$$\hat{\phi}^{n+1}(\vec{x}) = \phi^n(\vec{x} - \Delta t \vec{u}^n(\vec{x})) \tag{3.3}$$

These values are then advected backward in time as follows:

$$\hat{\phi}^n(\vec{x}) = \hat{\phi}^{n+1}(\vec{x} + \Delta t \vec{u}^n(\vec{x})) \tag{3.4}$$

Subsequently, the advection error is approximated and the final solution is computed as follows:

$$E(\vec{x}) = (\hat{\phi}^n(\vec{x}) - \phi^n(\vec{x}))/2 \tag{3.5}$$

$$\phi^{n+1}(\vec{x}) = \hat{\phi}^{n+1}(\vec{x}) - E(\vec{x}) \tag{3.6}$$

where $E(\vec{x})$ is the error and $\phi^{n+1}(\vec{x})$ is the final solution.

Whereas [94] only considered time-constant velocity fields in their precise analysis of the method, we consider time-varying velocity fields. First consider an initial circular bump function

$$\phi(\vec{x}) = \begin{cases} e^{\left(2 - \frac{2}{1-(\|\vec{x}-\vec{x}_c\|/r)^2}\right)} & \text{if } \|\vec{x}-\vec{x}_c\| < r \\ 0 & \text{if } \|\vec{x}-\vec{x}_c\| \geq r \end{cases} \tag{3.7}$$

where $\vec{x}_c = (0, .5)$ is the center of the bump and $r = .45$ is the radius of the bump. Then a single time-varying vortex velocity field defined by the stream function

$$\Psi(x, y) = \frac{2}{\pi} \sin^2\left(\pi \frac{x-1}{2}\right) \sin^2\left(\pi \frac{y-1}{2}\right), \quad (x, y) \in [-1, 1] \times [-1, 1] \tag{3.8}$$

is time-modulated by $\cos(\pi t/8)$ so that analytically the bump function will be twisted to its maximum extent at $t = 4$ and returned to its exact initial value at $t = 8$. The order of accuracy is computed using three successive grids and plotted against time in Figure 3.1 (Top) for Equations 3.3-3.6. Note that in the second half of the graph the solution drops to first order accuracy and worsens under refinement. Instead, second order accuracy is obtained by modifying Equations 3.3 and 3.4 to use velocities at time $t^{n+1/2}$ as shown in Figure 3.1 (Middle) but not by using velocities at time t^n (which was not pointed out in [94]). To clarify that this is a temporal error, Figure 3.1 (Bottom) uses a fixed Δt for all grids emphasizing that time t^n velocities still lead to second order accuracy in space. In practical simulations, although the velocity field at time $t^{n+1/2}$ is unknown when advecting from time t^n to time t^{n+1}, the spatial errors are often larger than the temporal errors, which implies that using time t^n velocities will often be satisfactory in practice.

We now perform a one-dimensional accuracy analysis to show that using time $t^{n+1/2}$ velocities achieves second order accuracy. Consider the one-dimensional version of Equation 3.1 as follows:

$$\frac{\partial \phi}{\partial t} + u(x, t) \frac{\partial \phi}{\partial x} = 0 \tag{3.9}$$

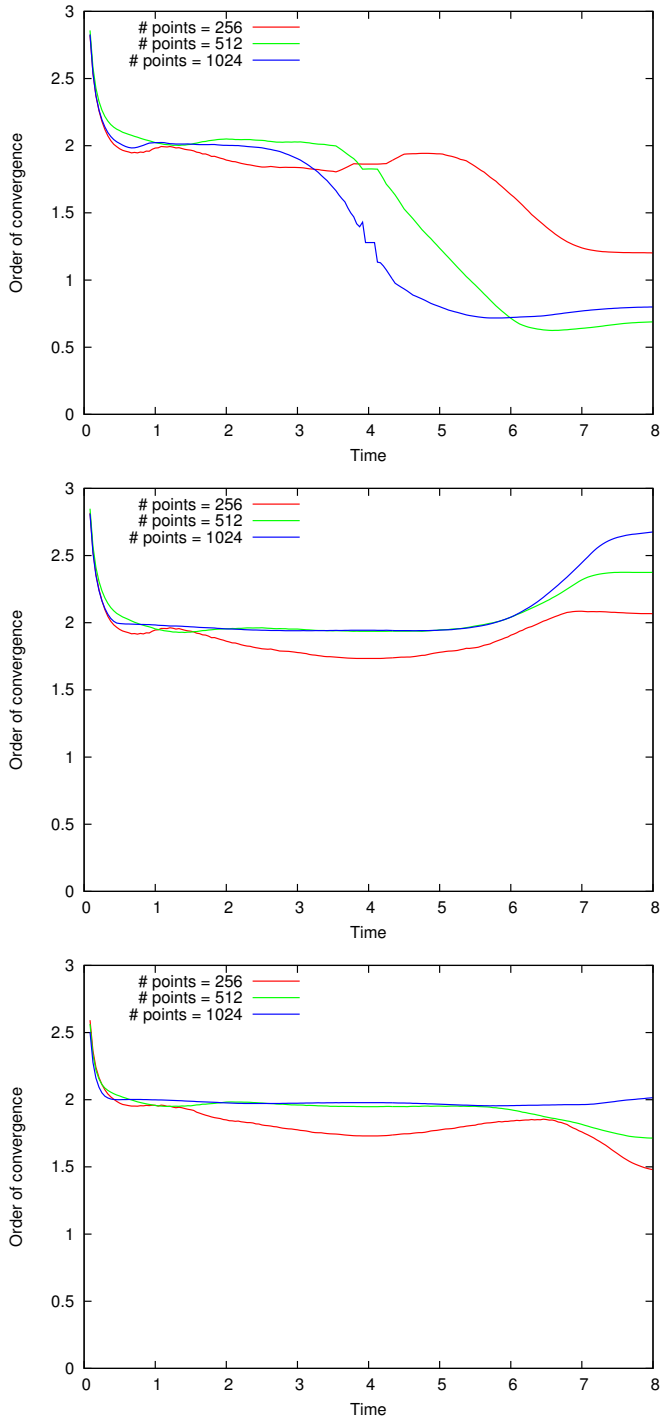


Figure 3.1: The order of accuracy of a time-varying single vortex test in two spatial dimensions with the SL-MacCormack method. (Top) uses velocities at time t^n for advection and a fixed CFL number. (Middle) uses velocities at time $t^{n+1/2}$ for advection and a fixed CFL number. (Bottom) uses velocities at time t^n for advection and fixes Δt to the value used on a 4096-point grid.

Assuming a positive velocity and a CFL less than one, we discretize Equation 3.9 using forward Euler in time and upwinding in space to get the forward advection step as follows:

$$\hat{\phi}_i^{n+1} = \phi_i^n + \frac{\Delta t}{\Delta x} u_i^{n+1/2}(\phi_{i-1}^n - \phi_i^n)$$

Similarly, the backward advection step can be discretized as

$$\hat{\phi}_i^n = \hat{\phi}_i^{n+1} + \frac{\Delta t}{\Delta x} u_i^{n+1/2}(\hat{\phi}_{i+1}^{n+1} - \hat{\phi}_i^{n+1})$$

Then the final solution of the SL-MacCormack advection is

$$\begin{aligned}\phi_i^{n+1} &= \hat{\phi}_i^{n+1} - (\hat{\phi}_i^n - \phi_i^n)/2 \\ &= \phi_i^n - \frac{\Delta t}{\Delta x} u_i^{n+1/2}(\frac{\phi_{i+1}^n - \phi_{i-1}^n}{2}) + \frac{\Delta t^2}{2\Delta x^2}((u_i^{n+1/2})^2(\phi_{i-1}^n - \phi_i^n) + u_i^{n+1/2} u_{i+1}^{n+1/2}(\phi_{i+1}^n - \phi_i^n))\end{aligned} \tag{3.10}$$

In order to eliminate the reference to $u_{i+1}^{n+1/2}$ we substitute $u_{i+1}^{n+1/2} = u_i^{n+1/2} + \Delta x \frac{\partial u_i^{n+1/2}}{\partial x} + O(\Delta x^2)$ into Equation 3.10 and after simplification we get

$$\begin{aligned}\phi_i^{n+1} =& \phi_i^n - \frac{\Delta t}{\Delta x} u_i^{n+1/2}(\frac{\phi_{i+1}^n - \phi_{i-1}^n}{2}) + \frac{\Delta t^2}{2\Delta x^2}((u_i^{n+1/2})^2(\phi_{i+1}^n - 2\phi_i^n + \phi_{i-1}^n) \\ &+ u_i^{n+1/2}(\Delta x \frac{\partial u_i^{n+1/2}}{\partial x} + O(\Delta x^2))(\phi_{i+1}^n - \phi_i^n))\end{aligned} \tag{3.11}$$

We further simplify Equation 3.11 by substituting $\frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\Delta x} = \frac{\partial \phi_i^n}{\partial x} + O(\Delta x^2)$, $\frac{\phi_{i+1}^n - 2\phi_i^n + \phi_{i-1}^n}{\Delta x^2} = \frac{\partial^2 \phi_i^n}{\partial^2 x} + O(\Delta x^2)$, and $\frac{\phi_{i+1}^n - \phi_i^n}{\Delta x} = \frac{\partial \phi_i^n}{\partial x} + O(\Delta x)$ which results in the final expression for the MacCormack update as follows:

$$\begin{aligned}\phi_i^{n+1} =& \phi_i^n - \Delta t u_i^{n+1/2} \frac{\partial \phi_i^n}{\partial x} + \frac{\Delta t^2}{2}((u_i^{n+1/2})^2 \frac{\partial^2 \phi_i^n}{\partial x^2} + u_i^{n+1/2} \frac{\partial u_i^{n+1/2}}{\partial x} \frac{\partial \phi_i^n}{\partial x}) \\ &+ O(\Delta t^2 \Delta x + \Delta t \Delta x^2 + \Delta t^2 \Delta x^2)\end{aligned} \tag{3.12}$$

In order to show second order accuracy we next compute a Taylor expansion for ϕ_i^{n+1}

in time with respect to ϕ_i^n and then use Equation 3.9 and the chain rule in order to exchange temporal and spatial derivatives as follows:

$$\begin{aligned}\phi_i^{n+1} =& \phi_i^n + \Delta t\frac{\partial \phi_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \phi_i^n}{\partial t^2} + O(\Delta t^3) \\ =& \phi_i^n - \Delta t u_i^n \frac{\partial \phi_i^n}{\partial x} + \frac{\Delta t^2}{2}(-\frac{\partial u_i^n}{\partial t}\frac{\partial \phi_i^n}{\partial x} + (u_i^n)^2\frac{\partial^2 \phi_i^n}{\partial x^2} + u_i^n\frac{\partial u_i^n}{\partial x}\frac{\partial \phi_i^n}{\partial x}) + O(\Delta t^3)\end{aligned} \tag{3.13}$$

Finally we substitute the Taylor expansions $u_i^{n+1/2} = u_i^n + \frac{\Delta t}{2}\frac{\partial u^n}{\partial t} + O(\Delta t^2)$ and $\frac{\partial u_i^n}{\partial x} = \frac{\partial u_i^{n+1/2}}{\partial x} + O(\Delta t)$ into Equation 3.13 as follows

$$\begin{aligned}\phi_i^{n+1} =& \phi_i^n - \Delta t(u_i^{n+1/2} - \frac{\Delta t}{2}\frac{\partial u^n}{\partial t})\frac{\partial \phi_i^n}{\partial x} \\ &+ \frac{\Delta t^2}{2}(-\frac{\partial u_i^n}{\partial t}\frac{\partial \phi_i^n}{\partial x} + (u_i^{n+1/2})^2\frac{\partial^2 \phi_i^n}{\partial x^2} + u_i^{n+1/2}\frac{\partial u_i^{n+1/2}}{\partial x}\frac{\partial \phi_i^n}{\partial x}) + O(\Delta t^3) \\ =& \phi_i^n - \Delta t u_i^{n+1/2}\frac{\partial \phi_i^n}{\partial x} + \frac{\Delta t^2}{2}((u_i^{n+1/2})^2\frac{\partial^2 \phi_i^n}{\partial x^2} + u_i^{n+1/2}\frac{\partial u_i^{n+1/2}}{\partial x}\frac{\partial \phi_i^n}{\partial x}) + O(\Delta t^3)\end{aligned} \tag{3.14}$$

By comparing Equation 3.12 and Equation 3.14, it can be seen that the difference between the numerical solution and the exact solution is on the order of $O(\Delta t^3 + \Delta t^2\Delta x + \Delta t\Delta x^2)$. Therefore, the numerical solution obtained by using time $t^{n+1/2}$ velocities in SL-MacCormack advection is second order accurate. Through similar derivations, it is easy to show that either using time t^n or time t^{n+1} velocities, the dominant error term is on the order of $O(\Delta t^2)$, leading to first order accuracy in time.

3.2 Chimera Advection Scheme

3.2.1 First Order Semi-Lagrangian Scheme

After filling ghost cells and overlapping regions as discussed in Section 2.2, we update each interior point $\vec{x}_{\text{object}}$ of each grid as follows. The standard first order accurate

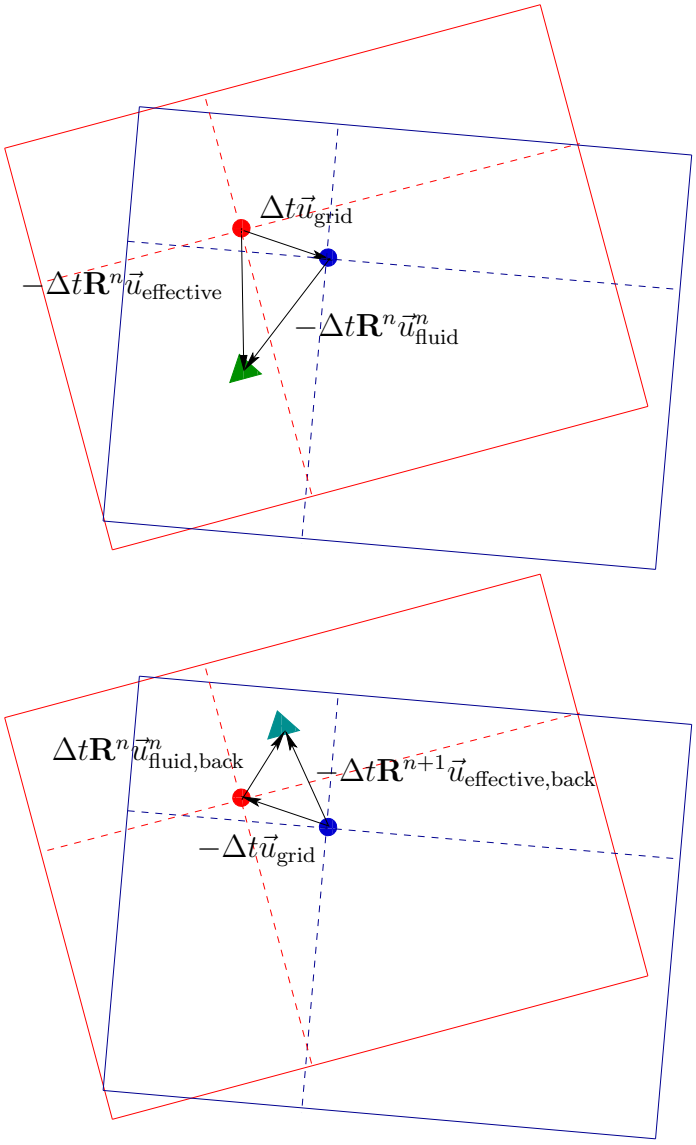


Figure 3.2: An illustration of our ALE advection in the world space for the point $\vec{x}_{\text{object}}$ that moves from $\vec{x}^n_{\text{world}}$ (shown as the red point) to $\vec{x}^{n+1}_{\text{world}}$ (shown as the blue point) according to the grid motion. (Top) Forward advection. (Left) Backward advection for SL-MacCormack.

semi-Lagrangian update for a world space location $\vec{x}_{\text{world}}$ (the blue dot in Figure 3.2 (Top)) on a stationary grid traces backwards along the characteristic ray defined by the time t^n velocity interpolated at $\vec{x}_{\text{world}}$ and interpolates an updated value at the lookup location (the green triangle in Figure 3.2 (Top)) as follows:

$$\phi^{n+1}_{\text{world}}(\vec{x}_{\text{world}}) = \phi^{n}_{\text{world}}(\vec{x}_{\text{world}} - \Delta t \vec{u}^{n}_{\text{world}}(\vec{x}_{\text{world}})) \tag{3.15}$$

where $\phi^{n+1}_{\text{world}}$, ϕ^{n}_{world} and $\vec{u}^{n}_{\text{world}}$ are parameterized by world space locations. In order to allow for grid motion we replace the world space values with the appropriate object space values transformed into world space. Since the grid is moving, we need to first determine where the point $\vec{x}_{\text{object}}$ will be at time t^{n+1}. This corresponds to $\vec{x}^{n+1}_{\text{world}}$ as follows:

$$\vec{x}^{n+1}_{\text{world}} = \mathbf{R}^{n+1}\vec{x}_{\text{object}} + \vec{s}^{n+1}$$

where $\mathbf{R}^{n+1}$ and $\vec{s}^{n+1}$ are the grid's time t^{n+1} rotation and translation respectively. Since the velocity field $\vec{u}^n$ is stored in time t^n object space we translate $\vec{x}^{n+1}_{\text{world}}$ into time t^n object space as $\vec{x}_{\text{new}} = (\mathbf{R}^n)^{-1}(\vec{x}^{n+1}_{\text{world}} - \vec{s}^n)$. Unlike standard semi-Lagrangian advection, $\vec{x}_{\text{new}}$ is typically not a grid point and therefore one must interpolate a velocity $\vec{u}^n_{\text{fluid}} = \vec{u}^n(\vec{x}_{\text{new}})$. We then rotate this velocity back into world space giving the characteristic velocity as follows:

$$\vec{u}^n_{\text{world}} = \mathbf{R}^n\vec{u}^n_{\text{fluid}}$$

Since ϕ^n is stored in object space, after computing the world space lookup location (the green triangle in Figure 3.2 (Top)) of the characteristic ray we transform this location into time t^n object space and interpolate the final value. Thus we arrive at the updated semi-Lagrangian advection formula for moving grids as follows:

$$\begin{aligned}\phi^{n+1}(\vec{x}_{\text{object}}) &= \phi^n((\mathbf{R}^n)^{-1}(\vec{x}^{n+1}_{\text{world}} - \Delta t\vec{u}_{\text{world}} - \vec{s}^n)) \\ &= \phi^n(\vec{x}_{\text{object}} - \Delta t(\vec{u}^n_{\text{fluid}} - (\mathbf{R}^n)^{-1}\vec{u}_{\text{grid}}))\end{aligned} \tag{3.16}$$

where $\phi^{n+1}_{\text{world}}(\vec{x}^{n+1}_{\text{world}})$ is equivalently replaced by $\phi^{n+1}(\vec{x}_{\text{object}})$. Additionally, the effective velocity of the grid location being updated is defined as $\vec{u}_{\text{grid}} = (\vec{x}^{n+1}_{\text{world}} - \vec{x}^{n}_{\text{world}})/\Delta t$ where $x^{n}_{\text{world}} = \mathbf{R}^n \vec{x}_{\text{object}} + \vec{s}^n$. Since we want to apply the single grid advection code, we simplify this formula as follows:

$$\phi^{n+1}(\vec{x}_{\text{object}}) = \phi^{n}(\vec{x}_{\text{object}} - \Delta t \vec{u}^{n}_{\text{effective}}) \tag{3.17}$$

where $\vec{u}^{n}_{\text{effective}} = \vec{u}^{n}_{\text{fluid}} - (\mathbf{R}^n)^{-1}\vec{u}_{\text{grid}}$. Note that Equation 3.17 traces the same world space characteristics as Equation 3.15 as shown in Figure 3.2 (Top).

When updating the velocities themselves, the scheme is a bit more complicated due to the rotation of grids' basis vectors. We approach this by first computing a full velocity vector on each MAC grid face by applying semi-Lagrangian advection to both of the coordinate directions at this face location and premultiplying this advected velocity vector by $(\mathbf{R}^{n+1})^{-1}\mathbf{R}^n$ to get the correct scalar components for the time t^{n+1} grid coordinate directions as follows:

$$\vec{u}_{\text{full}} = (\mathbf{R}^{n+1})^{-1}\mathbf{R}^n \vec{u}^n(\vec{x}_{\text{object}} - \Delta t \vec{u}^{n}_{\text{effective}}) \tag{3.18}$$

We then compute the final component value as $u^{n+1}(\vec{x}_{\text{object}}) = u_{\text{full}}$ or $v^{n+1}(\vec{x}_{\text{object}}) = v_{\text{full}}$ depending upon the component of the velocity field stored at the face at $\vec{x}_{\text{object}}$.

3.2.2 Semi-Lagrangian MacCormack Advection Scheme

The first step of SL-MacCormack advection is identical to the first order accurate semi-Lagrangian advection and results in the scalar field $\hat{\phi}^{n+1}$. In the second step, we fill the ghost cells of $\hat{\phi}^{n+1}$ and then advect these values backwards in time with the grid motion and fluid velocity reversed. The standard backward advection step in MacCormack advection for a stationary grid using world space values is as follows:

$$\hat{\phi}^{n}_{\text{world}}(\vec{x}_{\text{world}}) = \hat{\phi}^{n+1}_{\text{world}}(\vec{x}_{\text{world}} + \Delta t \vec{u}^{n}_{\text{world}}(\vec{x}_{\text{world}})) \tag{3.19}$$

Following a similar derivation as that used to derive Equation 3.16, by substituting object space values transformed to world space into Equation 3.19, we arrive at the backward advection step for moving grids as follows:

$$\begin{aligned}\hat{\phi}^n(\vec{x}_{\text{object}}) &= \hat{\phi}^{n+1}((\mathbf{R}^{n+1})^{-1}(\vec{x}^n_{\text{world}} + \Delta t \mathbf{R}^n \vec{u}_{\text{fluid,back}} - \vec{s}^n)) \\ &= \hat{\phi}^{n+1}(\vec{x}_{\text{object}} + \Delta t(\mathbf{R}^{n+1})^{-1}(\mathbf{R}^n \vec{u}_{\text{fluid,back}} - \vec{u}_{\text{grid}})) \end{aligned} \tag{3.20}$$

where $\vec{u}_{\text{fluid,back}} = \vec{u}^n(\vec{x}_{\text{object}})$. Note that conveniently, the fluid velocity at time t^n is already defined at the destination (which is a grid point) by standard averaging and does not need to be interpolated as it does in the first semi-Lagrangian advection step. (Note that if we were using the fluid velocity at time t^{n+1} that $\vec{u}_{\text{fluid,back}}$ would not be defined at the point $\vec{x}^n_{\text{world}}$ and interpolation will be required similar to the treatment of $\vec{x}_{\text{new}}$ in the semi-Lagrangian case.) Once again since we want to apply the single grid advection code, we simplify Equation 3.20 as follows:

$$\hat{\phi}^n(\vec{x}_{\text{object}}) = \hat{\phi}^{n+1}(\vec{x}^n_{\text{object}} + \Delta t \vec{u}_{\text{effective,back}}) \tag{3.21}$$

where $\vec{u}_{\text{effective,back}} = (\mathbf{R}^{n+1})^{-1}(\mathbf{R}^n \vec{u}_{\text{fluid,back}} - \vec{u}_{\text{grid}})$ is the effective velocity. Note that Equation 3.21 traces the same characteristics as Equation 3.19 as illustrated in Figure 3.2 (Bottom).

Once both the forward and backward advection steps have been performed, we again fill the ghost cells of the backward advection results $\hat{\phi}^n$ and then use Equation 3.5 to compute the error estimate

$$E(\vec{x}_{\text{object}}) = (\hat{\phi}^n(\vec{x}_{\text{object}}) - \phi^n(\vec{x}_{\text{object}}))/2$$

at each grid point in the usual manner. However, note that the values of $E(\vec{x}_{\text{object}})$ correspond to time t^n world space locations of the grid points $\vec{x}_{\text{object}}$, and the results from the first semi-Lagrangian step $\hat{\phi}^{n+1}(\vec{x}_{\text{object}})$ correspond to time t^{n+1} world space locations of $\vec{x}_{\text{object}}$. Thus, in order to compute the correct error correction at $\vec{x}^{n+1}_{\text{world}}$,

one needs to interpolate the error E at the location $\vec{x}_{\text{new}}$, i.e.

$$\phi^{n+1}(\vec{x}_{\text{object}}) = \hat{\phi}^{n+1}(\vec{x}_{\text{object}}) - E(\vec{x}_{\text{new}}) \tag{3.22}$$

When updating a velocity field, the backward advection step advects a full velocity vector for each face, premultiplies the resulting full velocity vector by $(\mathbf{R}^n)^{-1}\mathbf{R}^{n+1}$, and then takes the appropriate component depending on the face being considered– just as it was done in the forward advection step. The error is then computed for each face in their time t^n world space locations, obtaining two different error fields, E_u and E_v. The vector error at $\vec{x}_{\text{new}}$ is then calculated as $\vec{E}(\vec{x}_{\text{new}}) = (E_u(\vec{x}_{\text{new}}), E_v(\vec{x}_{\text{new}}))^T$ via interpolation. Finally, in order to obtain the scalar error correction for Equation 3.22, one premultiplies $\vec{E}(\vec{x}_{\text{new}})$ by $(\mathbf{R}^{n+1})^{-1}\mathbf{R}^n$ and takes the appropriate component corresponding to the face direction being considered.

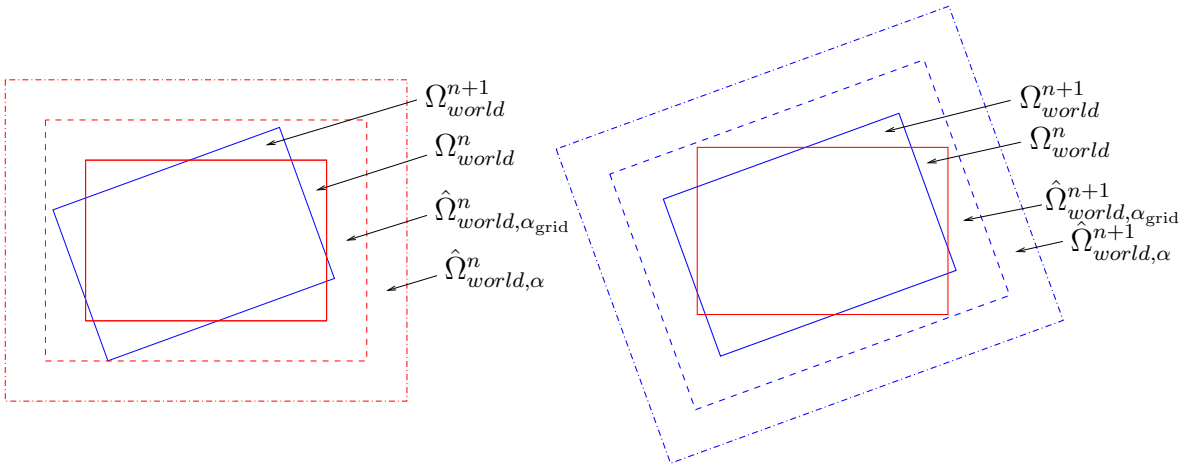


Figure 3.3: Illustration of the relation between the time step size and ghost cells. (Left) Forward advection. (Right) Backward advection for SL-MacCormack. The inner dashed boxes are the ghost domains bounding the grid motion—that is, the blue solid box should be completely inside the inner dashed red box, and in SL-MacCormack advection the red solid box also needs to be inside the inner blue dashed box. The outer dashed boxes are the ghost domains bounding the potential lookup points containing the values to be advected.

3.3 Time Step Size and Ghost Cells

For a Cartesian grid we define the object space domain as $\Omega_{\text{object}} = [a, b] \times [c, d]$ and the larger domain which contains the ghost cells as $\Omega_{\text{object},\alpha_{\text{grid}}} = [a - \alpha_{\text{grid}}\Delta x, b + \alpha_{\text{grid}}\Delta x] \times [c - \alpha_{\text{grid}}\Delta y, d + \alpha_{\text{grid}}\Delta y]$ where α_{grid} is the number of ghost cells. We then define the time t^n and t^{n+1} world space domains for Ω_{object} and $\Omega_{\text{object},\alpha_{\text{grid}}}$ as Ω^n_{world}, $\Omega^n_{\text{world},\alpha_{\text{grid}}}$, $\Omega^{n+1}_{\text{world}}$ and $\Omega^{n+1}_{\text{world},\alpha_{\text{grid}}}$. When carrying out the semi-Lagrangian advection as discussed in Section 3.2 one needs to interpolate $\vec{u}^n_{\text{fluid}}$ at every $\vec{x}_{\text{new}}$ location, which means that $\Omega^{n+1}_{\text{world}} \subseteq \Omega^n_{\text{world},\alpha_{\text{grid}}}$ so that a valid time t^n velocity can be interpolated from the $\Omega^n_{\text{world},\alpha_{\text{grid}}}$ grid for every degree of freedom in $\Omega^{n+1}_{\text{world}}$ that needs to be advected. In fact, to obtain a valid interpolation stencil on a staggered MAC grid one actually needs to shrink $\Omega_{\text{object},\alpha_{\text{grid}}}$ by half a grid cell in every dimension obtaining $\hat{\Omega}_{\text{object},\alpha_{\text{grid}}}$, and enforce that $\Omega^{n+1}_{\text{world}} \subseteq \hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$, see Figure 3.3 (Left).

Given a prescribed grid motion and a time step Δt, one could calculate the number of

ghost cells α_{grid} required to enforce this subset condition. However, this requires either reallocating ghost cells each time step which leads to cache coherency issues and high communication costs in MPI, or preallocating a sufficiently large number of ghost cells and using a subset of them which also poses issues due to inordinate memory allocation. Therefore, we instead fix the number of ghost cells and limit the time step Δt. Note that this strategy rules out the ability of grids to discontinuously change in position by an arbitrarily large distance, because in that case a grid could move by a distance larger than the ghost region even if Δt is arbitrarily small. Therefore we require grids to have a bounded velocity. We note that in this case while a smaller Δt still does not necessarily lead to less grid motion (e.g. a grid in one-dimension with position $x(t) = sin(t)$ has a larger displacement at $t = \pi/2$ than $t = \pi$), that a bounded velocity does guarantee that a grids displacement over a time step tends to zero as Δt does–guaranteeing that a Δt always exists such that $\Omega^{n+1}_{\text{world}} \subseteq \Omega^{n}_{\text{world},\alpha_{\text{grid}}}$.

It is inexpensive to check whether a rectangular domain lies within another rectangular domain, since it is only necessary to check if the four corners of the first domain lie within the second domain. This is a very light $O(1)$ computation compared to the $O(n^2)$ number of grid points on which advection is performed. Therefore, in order to maximize the allowable time step, Δt should be chosen as large as possible, implying that at least one of the four corners of $\Omega^{n+1}_{\text{world}}$ would lie exactly on the boundary of $\hat{\Omega}^{n}_{\text{world},\alpha_{\text{grid}}}$. This minimizes the total computation time by minimizing the number of time steps taken. Although various strategies exist to linearize and approximate Δt, a simple bisection procedure is also sufficient. This process is carried out for each grid and the minimum Δt over all grids taken as Δt_{grid}. Note that because grids can move further in shorter time steps, using Δt_{grid} for every grid could result in one of the grids moving outside of its respective $\hat{\Omega}^{n}_{\text{world},\alpha_{\text{grid}}}$ domain. Therefore, this condition needs to be checked at Δt_{grid} for all grids, and if invalid for any grid one can recompute the bisection for that grid in the interval $(0, \Delta t_{\text{grid}}]$ and then clamp all grids to this new value, repeating the process—which is guaranteed to converge as stated above.

Next, for each degree of freedom and corresponding location $\vec{x}_{\text{new}}$, one traces back along the fluid characteristic to interpolate a value at a point in object space $\vec{x}_{\text{lookup}} =$

$\vec{x}_{\text{object}} - \Delta t \vec{u}_{\text{effective}}$. We therefore use the final number of ghost cells $\alpha = \alpha_{\text{grid}} + \alpha_{\text{fluid}}$, and its corresponding domain, $\hat{\Omega}_{\text{object},\alpha}$, which is reduced by half a grid cell in every spatial dimension. The extra α_{fluid} ghost cells allow for tracing the fluid velocity characteristic backwards to find time t^n values of the advected quantity which lie outside of $\hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$. Note that values of $\vec{u}^n_{\text{fluid}}$ will be interpolated at the degrees of freedom inside $\Omega^{n+1}_{\text{world}}$ which are contained within $\hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$. Therefore for every fluid velocity in $\hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$ (which includes ghost cell velocities) we need to ensure that the time step is small enough such that the world space position of $\vec{x}_{\text{lookup}}$ does not lie outside $\hat{\Omega}^n_{\text{world},\alpha}$. We satisfy this with the following CFL condition

$$\Delta t_{\text{fluid}} \leq \alpha_{\text{fluid}} \frac{\min(\Delta x, \Delta y)}{\max|\vec{u}^n_{\text{fluid}}|} \tag{3.23}$$

for every point $\vec{x}_{\text{new}}$ in $\hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$. As we use bilinear interpolation to compute $\vec{u}^n_{\text{fluid}}$ at these points, we can more conveniently apply Equation 3.23 instead for every grid point of $\hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$.

Although we have allowed for two regions of ghost cells defined by α_{grid} and α_{fluid} to account for both the motion of the grid and fluid separately, it is not sufficient to take the minimum of Δt_{grid} and Δt_{fluid}, because as mentioned above, shrinking Δt may lead to larger grid motion. Therefore we first determine Δt_{fluid} and then compute a valid Δt_{grid} within the interval $(0, \Delta t_{\text{fluid}}]$.

In the case of SL-MacCormack advection, the forward advection step proceeds as in the first order accurate semi-Lagrangian case, and then for backward advection, we define the ghost cell domains $\hat{\Omega}^{n+1}_{\text{world},\alpha_{\text{grid}}}$ and $\hat{\Omega}^{n+1}_{\text{world},\alpha}$ similar to as was done for forward advection (see Figure 3.3 Right). While values of $\vec{u}^n_{\text{fluid,back}}$ do not need to be interpolated as discussed in Section 3.2, it is necessary to ensure that the destination of the characteristic path, the world space position of $\vec{x}_{\text{lookup,back}} = \vec{x}_{\text{object}} - \Delta t \vec{u}_{\text{effective,back}}$, lies inside $\hat{\Omega}^{n+1}_{\text{world},\alpha}$. Since we are using the time t^n velocities $\vec{u}^n_{\text{fluid,back}}$ for backward advection, Δt_{fluid} is sufficient to guarantee this as long as Ω^n_{world} lies within $\hat{\Omega}^{n+1}_{\text{world},\alpha_{\text{grid}}}$. It turns out that the Δt_{grid} which guarantees that $\Omega^{n+1}_{\text{world}} \subseteq \hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$ does not also necessarily guarantee that $\Omega^n_{\text{world}} \subseteq \hat{\Omega}^{n+1}_{\text{world},\alpha_{\text{grid}}}$. Therefore after determining

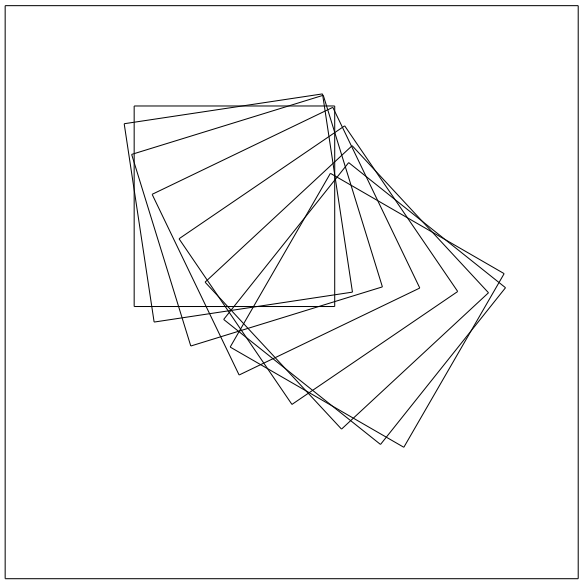


Figure 3.4: Illustration of motion of the grids. The background grid has no rotation or translation, while the fine grid spins and translates along a straight line inside the domain of the background grid.

Δt_{fluid}, we use our search algorithm to find a Δt_{grid} in the interval $(0, \Delta t_{\text{fluid}}]$ which guarantees both $\Omega^{n+1}_{\text{world}} \subseteq \hat{\Omega}^{n}_{\text{world},\alpha_{\text{grid}}}$ and $\Omega^{n}_{\text{world}} \subseteq \hat{\Omega}^{n+1}_{\text{world},\alpha_{\text{grid}}}$ for all grids (before taking a time step).

3.4 Numerical Results

In order to examine the convergence of our Chimera advection schemes we have implemented three convergence tests which consider the same grid configuration applied to three different velocity fields. The domain consists of a coarse background grid that has no rotation or translation, and a fine grid which is rotating and translating inside the coarse grid's domain. The world space domain of the coarse grid is $[-1, 1] \times [-1, 1]$, while the fine grid's object space domain is $[-.25, .45] \times [-.25, .45]$. The fine grid is kinematically driven with the position $\vec{s}(t) = (-.3, .2)\cos(\frac{t}{2\pi})$, and orientation $\theta(t) = \frac{t}{6\pi}$–that is, the fine grid spins and translates along a straight line

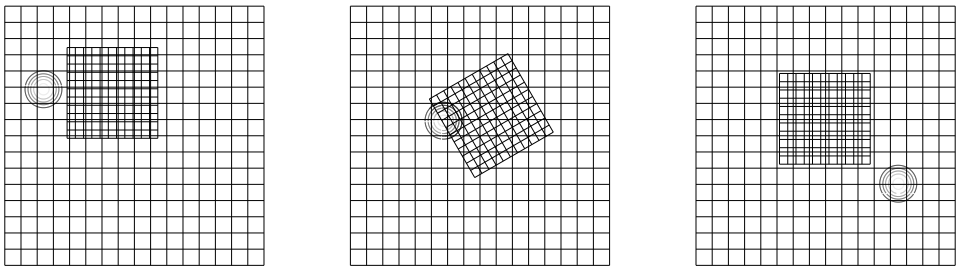


Figure 3.5: Snapshots of a simulation that advects a circular bump in a constant uniform velocity field. The circles shown in the figure are contours of the bump function. (Left) The snapshot at $t = 0$. (Middle) The snapshot at $t = 1$. (Right) The snapshot at $t = 3$.

from top left to bottom right, see Figure 3.4. The cell size of the fine grid is half that of the coarse grid, doubling the resolution of the area covered by the fine grid. In each of the tests we set $\alpha_{\text{grid}} = 2$ and $\alpha_{\text{fluid}} = 1$, and the number of ghost cells equal to 3. The initial density field in each test is the bump function defined in Equation 3.7 with different initial positions and radii specified below.

The first test advects the density field through a constant uniform velocity field. Snapshots of the simulation are given in Figure 3.5. The initial position of the bump is $(-.75, .4)$ and the radius is $r = .2$. The uniform velocity field is $\vec{u} = (\sqrt{3}/4, -1/4)$. The results in Figure 3.6 show that both the ALE semi-Lagrangian and ALE SL-MacCormack methods converge to the analytic solution. Note that in Figure 3.6, the plateau regions where the errors grow more slowly correspond to the times when the density field is primarily in the fine grid. The orders of accuracy for both methods are shown in Table 3.1.

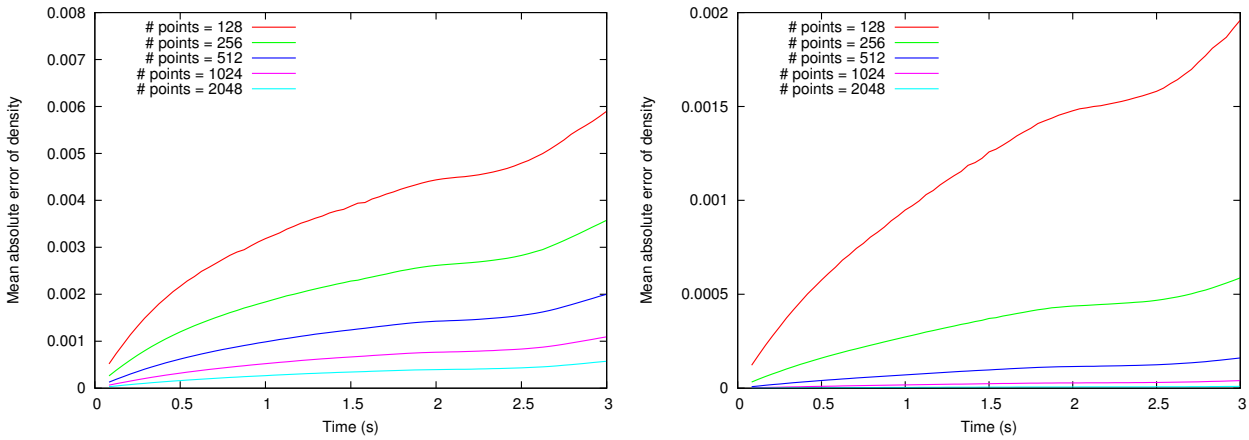


Figure 3.6: The L^1 norm error of the numerical solutions as a function of time in the constant uniform velocity test. (Top) The results of our ALE semi-Lagrangian method. (Bottom) The results of our ALE SL-MacCormack method.

	SL (time=3.0)				SL-MC (time=3.0)			
n	L^1 Error	Order	L^∞ Error	Order	L^1 Error	Order	L^∞ Error	Order
128	5.90×10^{-3}	–	4.26×10^{-1}	–	1.96×10^{-3}	–	1.22×10^{-1}	–
256	3.58×10^{-3}	0.72	2.50×10^{-1}	0.77	5.88×10^{-4}	1.74	4.53×10^{-2}	1.43
512	2.00×10^{-3}	0.84	1.32×10^{-1}	0.92	1.61×10^{-4}	1.87	1.88×10^{-2}	1.27
1024	1.09×10^{-3}	0.87	6.90×10^{-2}	0.94	4.04×10^{-5}	1.99	6.45×10^{-3}	1.54
2048	5.74×10^{-4}	0.93	4.07×10^{-2}	0.76	9.72×10^{-6}	2.05	1.86×10^{-3}	1.79

Table 3.1: The order of accuracy of our ALE semi-Lagrangian (SL) and SL-MacCormack (SL-MC) methods on the constant uniform velocity test. n is the number of points along each axis on each grid.

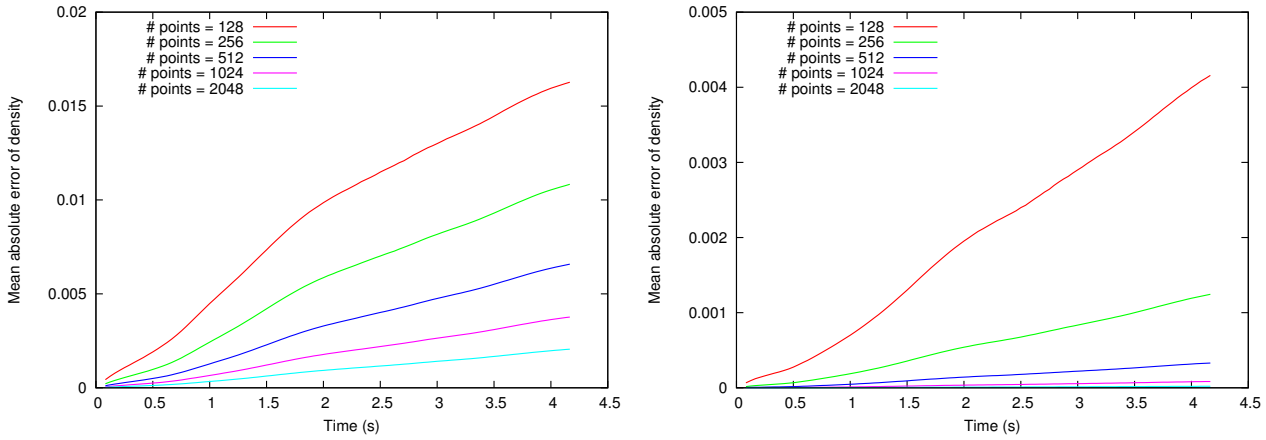


Figure 3.7: The L^1 norm errors of the numerical solutions as a function of time in the constant single vortex test. (Top) The results of our ALE semi-Lagrangian method. (Bottom) The results of our ALE SL-MacCormack method.

	SL				SL-MC			
n	L^1 Error	Order	L^∞ Error	Order	L^1 Error	Order	L^∞ Error	Order
128	1.63×10^{-2}	–	5.81×10^{-1}	–	4.16×10^{-3}	–	1.40×10^{-1}	–
256	1.08×10^{-2}	0.59	3.91×10^{-1}	0.57	1.24×10^{-3}	1.74	5.58×10^{-2}	1.32
512	6.58×10^{-3}	0.72	2.29×10^{-1}	0.77	3.30×10^{-4}	1.92	1.86×10^{-2}	1.59
1024	3.77×10^{-3}	0.80	1.24×10^{-1}	0.89	8.39×10^{-5}	1.98	5.02×10^{-3}	1.89
2048	2.06×10^{-3}	0.87	6.40×10^{-2}	0.95	2.04×10^{-5}	2.04	1.42×10^{-3}	1.82

Table 3.2: The order of accuracy of our ALE semi-Lagrangian (SL) and SL-MacCormack (SL-MC) methods on the constant single vortex test when the bump has been rotated for one cycle. n is the number of points along each axis on each grid.

The second test advects the density field through a constant single vortex velocity field. The initial position of the bump is $(0, .5)$ and the radius is $r = .3$. The velocity field function is given by $\vec{u}(x, y) = \frac{12\pi}{25}(-y, x)$. The L^1 norm errors of different resolution simulations are plotted in Figure 3.7, and the orders of accurary are calculated when the field is rotated exactly one cycle, see Table 3.2.

The third test we ran was the time-varying single vortex example as used in the single grid test from Section 3.1. To calculate the errors and the orders of accuracy, the results from the 16384-point-resolution simulation performed on a single grid are used in this test as the ground truth, allowing us to show that the single grid simulations

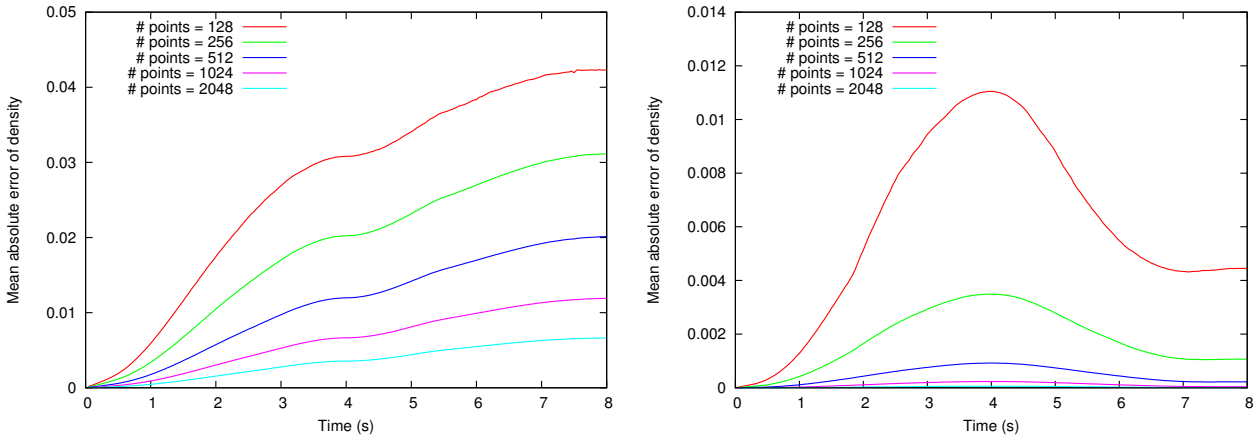


Figure 3.8: The L^1 norm error of the numerical solutions as a function of time in the time-varying single vortex test. (Top) The results of our ALE semi-Lagrangian method. (Bottom) The results of our ALE SL-MacCormack method.

and the two-moving-grid simulations converge to the same solution. Figure 3.9 shows that the ALE SL-MacCormack method achieves second order accuracy using time $t^{n+1/2}$ velocities for advection, while the ALE semi-Lagrangian method is near first order accurate. The L_1 norm error plot in Figure 3.8 shows that in tests using the ALE SL-MacCormack method the errors start to decrease at $t = 4$. This is because the errors of each time step in numerical simulations are signed errors, which may either cancel or accumulate when summed up in time. In this test, the velocity field is antisymmetric with respect with $t = 4$, making some error terms in SL-MacCormack advection also antisymmetric and being able to cancel. However, the errors of tests using the semi-Lagrangian method continues to increase in time after $t = 4$ because the semi-Lagrangian advection operator is not symmetric either in space or in time.

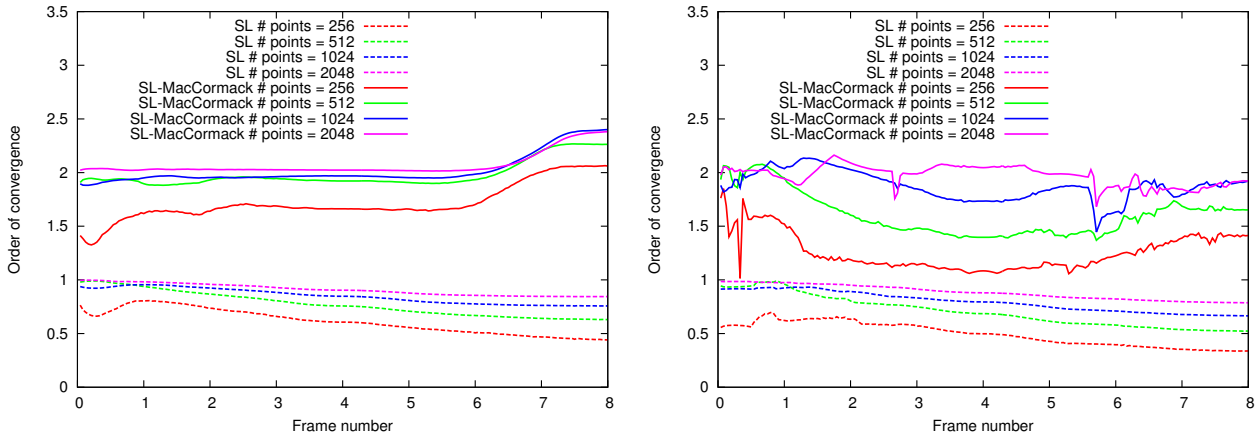


Figure 3.9: The order of accuracy of the time-varying single vortex test.(Top) The order of accuracy based on L^1 norm of errors. (Bottom) The order of accuracy based on L^∞ norm of the errors.

Chapter 4

SPD Poisson Equation Discretization

4.1 Poisson Equation

With an eye towards incompressible flow presented in section 6, we first consider the Poisson equation:

$$\nabla \cdot \beta(\vec{x}) \nabla \phi(\vec{x}) = f(\vec{x}), \quad \vec{x} \in \Omega \tag{4.1}$$

$$\phi(\vec{x}) = g(\vec{x}), \quad \vec{x} \in \partial\Omega_D \tag{4.2}$$

$$\vec{n}(\vec{x}) \cdot \nabla \phi(\vec{x}) = h(\vec{x}), \quad \vec{x} \in \partial\Omega_N \tag{4.3}$$

where $\vec{n}$ is the outward pointing normal to the boundary, Ω is the computational domain, and $\partial\Omega_D$ and $\partial\Omega_N$ are the portions of the boundary on which Dirichlet and Neumann boundary conditions are enforced, respectively. For simplicity of presentation, we take β equal to one noting that nothing about our method prevents it from being straightforward to extend to a variable β.

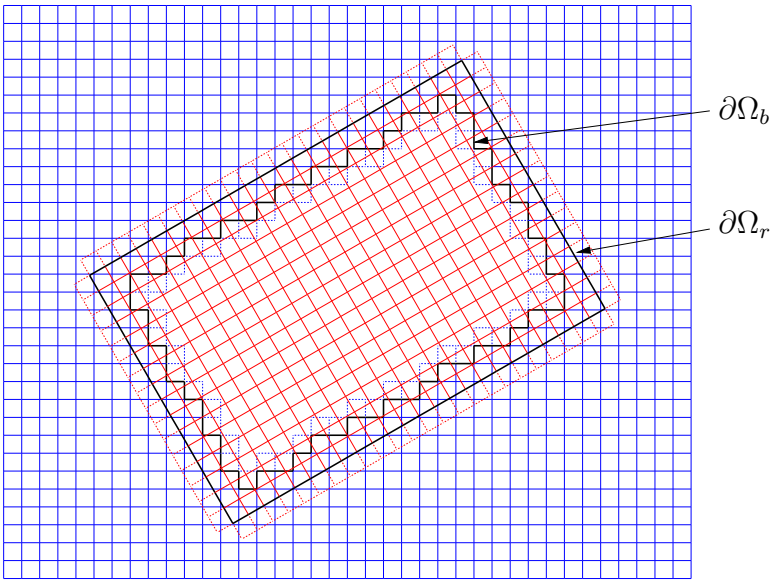


Figure 4.1: For fully overlapping grids one must omit a number of grid cells in the interior in order to provide boundaries for the application of boundary conditions, in this case creating an interior boundary, $\partial\Omega_b$, on the blue grid to receive information from the red grid. For partially overlapping grids one may omit cells for the sake of efficiency, but it is not always necessary in order to create a boundary on which one can prescribe coupling as in the case for fully overlapping grids. The black boundaries of both grids, $\partial\Omega_b$ and $\partial\Omega_r$, are locations on which Neumann boundary conditions would be applied and the dotted cells of both grids are locations on which Dirichlet boundary conditions would be applied.

4.2 Overlapping Grid Solvers

We begin with an overview of the general approach used for solving Poisson equations on overlapping grids. Both for the sake of computational efficiency and to facilitate the application of boundary conditions, one typically removes a number of cells in the overlapping region betweens grids as shown in Figure 4.1. Although Figure 4.1 shows one grid completely enclosed within another, we note that cells are still removed when grids are only partially overlapped. In that case it is primarily performed for efficiency. When deciding which cells to remove, it is necessary to allow for a large enough overlap such that valid interpolation stencils exist for nodes on which the boundary conditions are specified. However, it is also important to minimize

this overlap in order to prevent the solutions within these overlapping regions from drifting apart. See [19] for further discussion of these aspects of grid generation.

In order to enforce either Dirichlet or Neumann boundary conditions along intergrid boundaries, operators which compute values for ghost nodes by interpolating from non-ghost nodes on other overlapping grids are substituted into the equations. With these substitutions we arrive at the following discretized versions of Equations 4.1-4.3 for two overlapping grids:

$$\mathbf{D}_1 \left(\mathbf{G}_1\boldsymbol{\phi}_1 + \mathbf{G}_{1g}\mathbf{J}_{1,2}\boldsymbol{\phi}_2 + \mathbf{H}_{1,2}\mathbf{G}_2\boldsymbol{\phi}_2\right) = \mathbf{f}_1 - \mathbf{D}_1 \left(\mathbf{G}_{1d}\boldsymbol{\phi}_{1d} + \frac{\partial \boldsymbol{\phi}_{1n}}{\partial \vec{n}}\right) \tag{4.4}$$

$$\mathbf{D}_2 \left(\mathbf{G}_2\boldsymbol{\phi}_2 + \mathbf{G}_{2g}\mathbf{J}_{2,1}\boldsymbol{\phi}_1 + \mathbf{H}_{2,1}\mathbf{G}_1\boldsymbol{\phi}_1\right) = \mathbf{f}_2 - \mathbf{D}_2 \left(\mathbf{G}_{2d}\boldsymbol{\phi}_{2d} + \frac{\partial \boldsymbol{\phi}_{2n}}{\partial \vec{n}}\right) \tag{4.5}$$

where ϕ_1 and ϕ_2 are the discrete values of ϕ located at non-ghost cells on grids 1 and 2 respectively, $\mathbf{f}_1$ and $\mathbf{f}_2$ are the discrete values of the right hand side of Equation 4.1, $\frac{\partial \phi_{1n}}{\partial \vec{n}}$ and $\frac{\partial \phi_{2n}}{\partial \vec{n}}$ are the Neumann boundary conditions on grids 1 and 2 respectively as specified in Equation 4.3, and ϕ_{1d} and ϕ_{2d} are the Dirichlet conditions on grids 1 and 2 respectively as specified in Equation 4.2. $\mathbf{D}_1$ and $\mathbf{D}_2$ are the divergence operators, $\mathbf{G}_1$ and $\mathbf{G}_2$ are the terms from the gradient operators corresponding to non-ghost cells, $\mathbf{G}_{1g}$ and $\mathbf{G}_{2g}$ are the terms from the gradient operator corresponding to ghost cells, and $\mathbf{G}_{1d}$ and $\mathbf{G}_{2d}$ are the terms from the gradient operator corresponding to the Dirichlet boundary conditions on the computational domain as specified in Equation 4.2. $\mathbf{J}_{1,2}$ interpolates values of ϕ from non-ghost cells on grid 2 to ghost cells on grid 1, and similarly $\mathbf{J}_{2,1}$ interpolates values of ϕ from non-ghost cells on grid 1 to ghost cells on grid 2. $\mathbf{H}_{1,2}$ interpolates discretized values of the gradient of ϕ_2 from non-ghost faces on grid 2 to ghost faces on grid 1, and similarly $\mathbf{H}_{2,1}$ interpolates discretized values of the gradient of ϕ_1 from non-ghost faces on grid 1 to ghost faces on grid 2.

In general $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_{1d}$ and $\mathbf{G}_{2d}$ are defined using the same stencils as would be used for single grids (e.g. using the composite finite difference discretization and deformation Jacobian in the case of curvilinear grids). The interpolation operators $\mathbf{J}_{1,2}$, $\mathbf{J}_{2,1}$, $\mathbf{H}_{1,2}$ and $\mathbf{H}_{2,1}$ are dependent upon the desired order of the scheme (e.g.

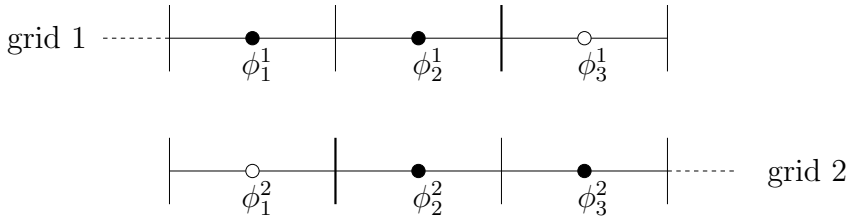


Figure 4.2: In one dimension, two grids overlap by 3 cells with the cell centered solution variables ϕ_1^1, ϕ_2^1 and ϕ_3^1 on the grid 1 and ϕ_1^2, ϕ_2^2 and ϕ_3^2 on the grid 2. In this case ϕ_3^1 and ϕ_1^2 are ghost cells where Dirichlet boundary conditions would be applied. The bold face lines between ϕ_2^1 and ϕ_3^1, and ϕ_1^2 and ϕ_2^2 represent where Neumann boundary conditions would be applied.

multilinear operators for second order accuracy and tensor product Lagrange interpolants more generally). We note that if one did not allow for a large enough overlap between the grids, additional terms would appear in Equations 4.4 and 4.5 where values from one grid would be interpolated to another grid as boundary conditions, which would then in turn be interpolated back to the original grid as additional terms in the former grid's own boundary conditions. This circular dependency is further discussed in [19] in which they state one could either perform an additional implicit solve to compute the true interpolation weights as a preprocessing step or include the additional implicit interpolation equations into the full system. While it is possible to support a large enough overlap such that this is not necessary [19] suggests that in the case of coarse grids this implicit interpolation is advantageous by allowing a minimal overlap.

While most methods use only Dirichlet conditions to couple grids (i.e. $\mathbf{H}_{1,2} = \mathbf{0}$ and $\mathbf{H}_{2,1} = \mathbf{0}$), if instead one chooses to enforce compatibility between the solutions on different grids using Neumann boundary conditions along intergrid boundaries, the system can be singular. To illustrate this we consider the example of two grids as shown in Figure 4.2. The Laplacians (before cutting out the cells containing ϕ_3^1 and ϕ_1^2) for the cells containing samples ϕ_2^1 and ϕ_2^2 are $\frac{\partial^2 \phi_2^1}{\partial x^2} = (\phi_3^1 - 2\phi_2^1 + \phi_1^1)/\Delta x^2$ and $\frac{\partial^2 \phi_2^2}{\partial x^2} = (\phi_3^2 - 2\phi_2^2 + \phi_1^2)/\Delta x^2$ respectively. Enforcing a Neumann boundary condition of $\frac{\partial \phi_{2.5}^1}{\partial x} = (\phi_3^2 - \phi_2^2)/\Delta x$ on grid 1, gives the modified Laplacian $\frac{\partial^2 \phi_2^1}{\partial x^2} = (\phi_3^2 - \phi_2^2 - \phi_2^1 + \phi_1^1)/\Delta x^2$. Enforcing a Neumann boundary condition of $\frac{\partial \phi_{1.5}^2}{\partial x} = (\phi_2^1 - \phi_1^1)/\Delta x$ on grid

2, gives the modified Laplacian $\frac{\partial^2 \phi_2^2}{\partial x^2} = (\phi_3^2 - \phi_2^2 - \phi_2^1 + \phi_1^1)/\Delta x^2$. Thus, $\frac{\partial^2 \phi_2^1}{\partial x^2} = \frac{\partial^2 \phi_2^2}{\partial x^2}$ and consequently the resulting coupled system is singular. Note that while more complex multidimensional cases will not always be exactly singular due to approximations when interpolating, the system still asymptotes towards singularity as the grid is refined and thus will still be poorly conditioned. Although the rank 1 nullspace admitted by solid wall boundary conditions in standard incompressible flow has long been addressed by projecting out the nullspace, the complexity of the nullspace resulting from the interpolation makes it difficult to compute. This singularity is also avoided by using mixed boundary conditions where the boundary condition at one or more locations is replaced by a Dirichlet coupling condition.

In general Equations 4.4 and 4.5 do not yield a symmetric system due to the fact that the intergrid boundary conditions applied to the first grid are not coincident with those applied to the second grid. Similar issues arise in fluid-structure interaction problems where the fluid-structure boundaries are non-conforming. One common approach to solving these problems is to couple the solid and fluid velocities together at fluid faces by using a Lagrange multiplier to conservatively force continuity between the fluid and structures velocities across the boundary, see e.g. [91, 90]. One could follow this strategy by modifying Equations 4.4 and 4.5 to include similar forcing terms in order to enforce matching values of $\nabla\phi$ at boundaries. However, while this will produce a symmetric system, the same nullspace issues discussed above for the case of using Neumann boundary conditions will apply here.

A common approach to solving Equations 4.4 and 4.5 is to use a block Gauss-Seidel outer iteration such as follows:

$$\mathbf{D}_1\mathbf{G}_1\boldsymbol{\phi}_1^{k+1} = \mathbf{f}_1 - \mathbf{D}_1\left(\mathbf{G}_{1g}\mathbf{J}_{1,2}\boldsymbol{\phi}_2^k + \mathbf{H}_{1,2}\mathbf{G}_2\boldsymbol{\phi}_2^k + \mathbf{G}_{1d}\boldsymbol{\phi}_{1d} + \frac{\partial \boldsymbol{\phi}_{1n}}{\partial \vec{n}}\right) \tag{4.6}$$

$$\mathbf{D}_2\mathbf{G}_2\boldsymbol{\phi}_2^{k+1} = \mathbf{f}_2 - \mathbf{D}_2\left(\mathbf{G}_{2g}\mathbf{J}_{2,1}\boldsymbol{\phi}_1^{k+1} + \mathbf{H}_{2,1}\mathbf{G}_1\boldsymbol{\phi}_1^{k+1} + \mathbf{G}_{2d}\boldsymbol{\phi}_{2d} + \frac{\partial \boldsymbol{\phi}_{2n}}{\partial \vec{n}}\right) \tag{4.7}$$

where the superscript on ϕ indicates the Gauss-Seidel iterate. This scheme allows the diagonal block associated with each grid to be solved independently of the other

grids with the single grid solver of the implementer's choice (i.e. Equations 4.6 and 4.7 would be solved in an alternating fashion until a convergence criterion was reached). However, this scheme, also known as a Schwartz alternating method or Partitioned method, suffers from significant convergence issues. While for some cases there exists proofs showing that the method allows the overall scheme to converge to the solution of the original problem, the method can often diverge. In fact, when enforcing compatibility between the solution on different grids using Neumann boundary conditions, in addition to the diagonal blocks themselves being singular, the right hand side can even be incompatible.

Some of the more successful approaches have directly solved Equations 4.4 and 4.5 monolithically for $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$. Multigrid methods have been shown as extremely effective (see e.g. [43, 42]) by exploiting the fact that the component grids are themselves logically rectangular. This allows for straighforward and accurate coarsening strategies by simultaneously coarsening each grid by the same factor and then computing the corresponding monolithic coupled system for the new coarse discretizations. Krylov methods (see e.g. [50]) have also been successful using either biCG-stab or GMRES with an incomplete LU preconditioner.

In an attempt to build a symmetric coupling between the grids, we first consider a cut cell approach as shown in Figure 4.3. The degrees of freedom remain at the cell centers of both grids and each degree of freedom corresponds to either a full or partially cut cell. A finite volume approach can be used to compute the volume weighted discrete divergence for each cell by computing the net flux across all incident faces. However, computing the gradient is less obvious. In order to produce a symmetric Laplacian, the gradient operator for each face must include terms only for incident cells as illustrated in Figure 4.3 (Right). Unfortunately, regardless of exactly how weights are chosen in these stencils, for most grid configurations, discretization errors appear which do not vanish in the L^∞ norm under grid refinement (see, for example [4] and [84]). This is due to the fact that the component of the gradient computed at each face is not orthogonal to the face and does not tend towards orthogonality upon

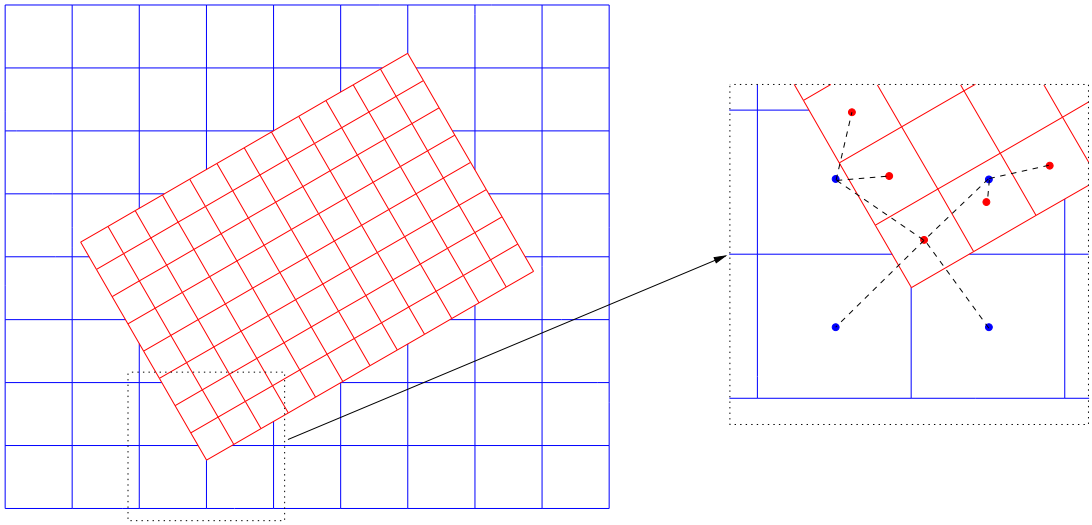


Figure 4.3: In this two grid example, we cut overlapped cells on the blue coarse grid by removing parts of these cells which are covered by finer red grid cells. The blue dots are the locations of ϕ samples on the blue grid. The red dots are the locations of ϕ samples on the fine red grid. The dashed lines indicate the direction of the components of the gradient across faces along the intergrid boundaries. Generally the direction of these components are far from orthogonal to their respective faces. The components of the gradients incident to the blue ϕ sample contained within the red grid (in the blown up portion of the figure to the right) are even inverted.

refinement. This is particularly evident at sharp corners along the intergrid boundary. Authors such as [105] have approached this issue by using deferred correction methods which attempt to iteratively improve the error by adding the difference in the gradient components using a gradient computed in the previous iteration. While this can produce accurate results for cases where the angle between the gradient and face normal is small enough, convergence issues can persist in more skewed cases. Furthermore, the additional cost of requiring multiple outer iterations of the entire system makes the method computationally infeasible. The main idea behind our coupling is that rather than trying to change effective ϕ sample locations by constructing complicated gradient stencils, one could instead modify the cell geometry while fixing ϕ sample locations in order to produce accurate centered difference ϕ derivatives along grid boundaries.

4.3 Voronoi Diagram Discretization

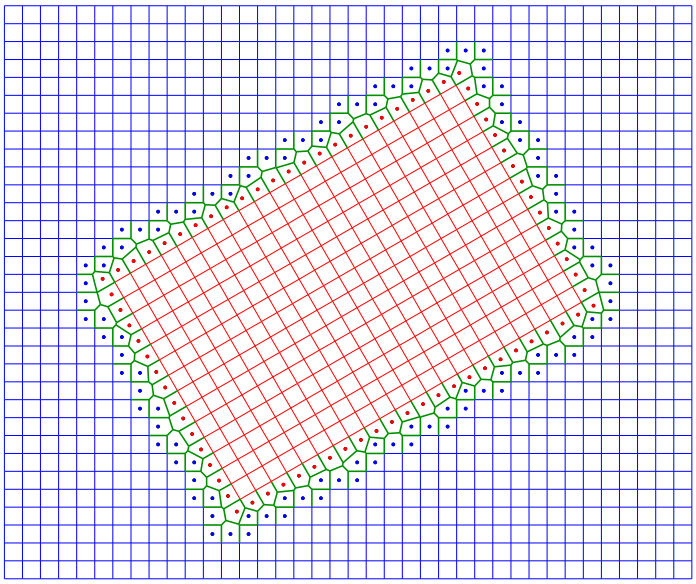


Figure 4.4: The composite grid resulting from our proposed method for coupling multiple grids by generating a Voronoi mesh along their boundaries. The dots are the cell centers of boundary cells that are incident to voronoi faces - note every dot corresponds to a degree of freedom originally from either the red or blue grid as indicated by their color.

Inspired by [97] and [12], we take the approach of using a Voronoi diagram to compute the coupling terms between the discretizations on each grid, as illustrated in Figure 4.4. By definition each face in a Voronoi diagram is both orthogonal to and bisects the line segment between the centers of the cells incident to the face. We note that because Cartesian grids are already Voronoi diagrams it is only necessary to mesh along intergrid boundaries. While this approach deviates from traditional Chimera grid schemes which do not apply any meshing in order to couple together overlapping grids, we stress that typically one is already spending considerable effort constructing and maintaining body-fitted grids with curvilinear coordinates, so it is reasonble to perform a small amount of additional meshing on a lower-dimensional manifold in order to produce a well conditioned symmetric positive definite system which allows for the use of simple and stable solvers such as preconditioned conjugate gradient.

We do note that this lower-dimensional meshing must occur every time step if the grids are moving. However, if one is using more complicated curvilinear grids, one must also compute the inverse mappings for interpolation locations each time step.

Since we do not exchange information between grids through the use of overlapping regions, we do not require our grids to overlap by a certain number of cells. Instead, we cut out enough cells in order to explicitly prevent the remaining parts of each grid from overlapping as illustrated in Figure 4.5 (a). It is also important to not remove too many cells and create large gaps between the grids which can also introduce significant numerical error in the resulting discretization. We proceed by removing any coarse cells which contain the cell center of a finer cell, as well any coarse cells whose cell center is contained within a finer cell. It is important to emphasize that when checking a coarse cell against another finer cell, we first check whether the finer cell itself is not cut out by an even finer cell. Although this fine to coarse strategy favors finer grid cells, any other reasonable strategy could be used with proper modifications. For example, in the case when two solid bodies (each with their own grid) are in close proximity, it may be desirable to prefer cells based on their distance from their respective bodies.

We directly compute the Voronoi diagram for all remaining cell centers as shown in Figure 4.5, noting that for all the interior cells of each grid the Cartesian geometry already is a Voronoi diagram - and thus we only need to compute the geometry for boundary cells. For each boundary cell we first find all nearby cell centers within some prescribed distance τ. Then considering each nearby cell center one at a time we construct a candidate plane for the polygonal face between these two cell centers, equidistant between these two cell centers. For every other neighboring cell center within the distance τ, this candidate plane is clipped to a smaller polygonal area by the candidate planes formed between the original cell center and each of the other nearby cell centers. Note that the final plane could be an empty set in which case the two cells values do not interact and are not directly coupled in the discretization. Note also that while building a Voronoi mesh can sometimes require sensitive calculations, this sensitivity only applies to arbitrary point sets and is not a concern for the highly

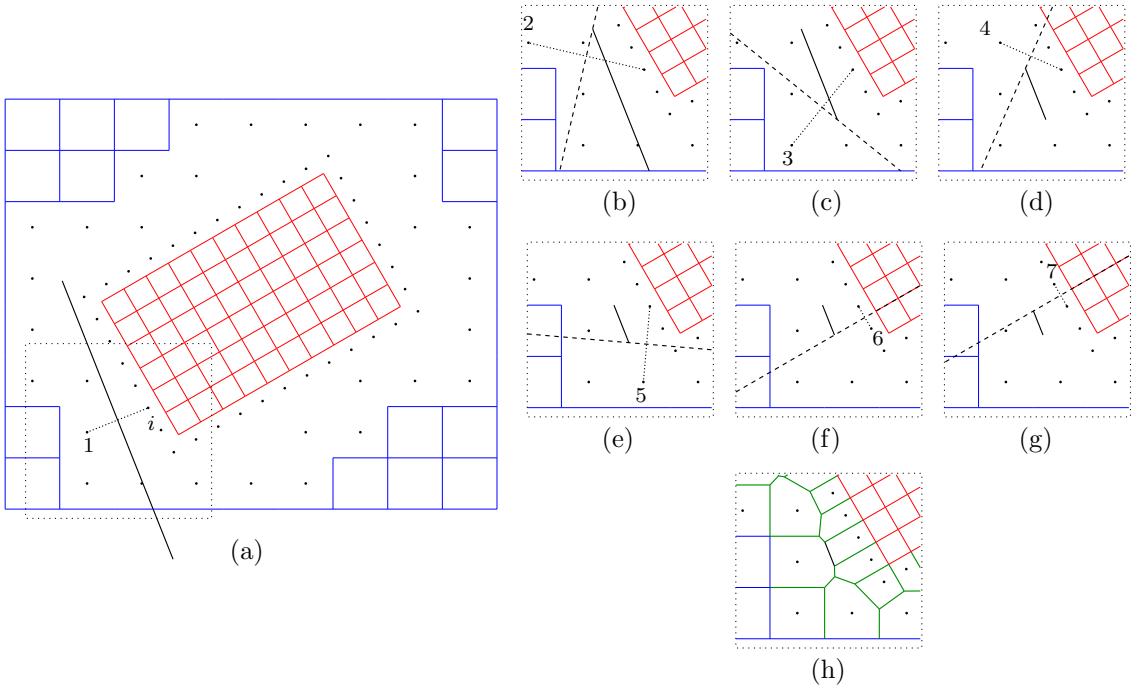


Figure 4.5: Our procedure for generating the Voronoi face between cell i and its first neighbor cell 1. (a) The remaining Cartesian cell geometry after cutting out overlapping cells on the grids is drawn in blue for the coarse grid and red for the fine grid. The black dots indicate non-removed cells along the intergrid boundary with only partial geometry remaining after the cell removing procedure. The black line indicates the initial unclipped face between cells i and 1 and the dotted line indicates the orthogonal line between those cells' centers. (b)-(g) The face between cells i and 1 is sequentially clipped by the plane (the dashed lines) between cell i and its neighbor cells 2 to 7. The black line shows resulting face between cells 1 and i after each clipping operation. (h) Shows the final face in context with the complete Voronoi diagram (shown in green).

structured samples in our application.

For the sake of exposition, we assume that our grids have equal edge length cube cells. Then, given our prescribed algorithm for deleting cell centers, the maximum distance to a remaining cell center from a point randomly chosen within a certain grid is $|\Delta\vec{x}|$. To see this, first note that if the randomly chosen point lies within a cell that is not deleted, it is trivially true. Otherwise, if that cell was deleted, it either contains a cell center from a finer grid and again it is trivially true, or a finer cell contains the cell center of the original grid and it is again true. In this case, the distance from the randomly chosen point to the deleted cell's center is at most $|\Delta\vec{x}|/2$ and the distance from the that cell's center to the finer grid cell center is at most $|\Delta\vec{x}|/2$, thus proving the assertion. As a result, for any cell center more than a distance of $|\Delta\vec{x}|$ within a grid's domain, each point within the corresponding Voronoi cell must be no more than $|\Delta\vec{x}|$ from the cell center. Furthermore, each face incident to the cell must be no further than $|\Delta\vec{x}|$ from the cell center. Since faces are equidistant to their incident cell centers, the original cell center must be less than a distance of $2|\Delta\vec{x}|$ away from another cell's center in order for the two cells to share a face. Thus in order to compute τ for a given cell, we find the finest grid whose domain, shrunken by the grid's corresponding $|\Delta\vec{x}|$, contains the cell center and set $\tau = 2|\Delta\vec{x}|$. If no grid is found we use $|\Delta\vec{x}|$ from the coarsest grid and include ghost cells which lie outside the computational domain when creating and clipping Voronoi faces.

Once we have computed the Voronoi mesh geometry we proceed to discretize Equations 4.1-4.3 using a finite volume approach as follows. We begin by integrating Equation 4.1 over the control volume, for a cell i as follows:

$$\int_{\Omega_i} \nabla \cdot \nabla\phi(\vec{x})d\vec{x} = \int_{\Omega_i} f(\vec{x})d\vec{x}$$

where Ω_i is the control volume of cell i. We then invoke the divergence theorem to change the volume integral into a surface integral as follows:

$$\int_{\partial\Omega_i} \vec{n}(\vec{x}) \cdot \nabla\phi(\vec{x})d\vec{x} = \int_{\Omega_i} f(\vec{x})d\vec{x}$$

where $\partial\Omega_i$ is the surface of the control volume of cell i, and $\vec{n}$ is the outward pointing normal on the surface of cell i. We subsequently discretize these equations by approximating the integral on the left hand side by summing over the area weighted normal derivatives at face centers and by approximating the right hand side integral as the cell volume times $f(\vec{x})$ evaluated at the cell center as follows.

$$\sum_{j \in N_i} A_{i,j}(\vec{n}_{i,j} \cdot \nabla\phi_{i,j}) = V_i f_i \tag{4.8}$$

where N_i contains the indices of the cells adjacent to cell i, $A_{i,j}$ is the area of the face between cells i and j and V_i is the volume of cell i. $\vec{n}_{i,j} = (\vec{x}_j - \vec{x}_i)/|\vec{x}_j - \vec{x}_i|$ is the normal pointing from cell i to cell j where $\vec{x}_i$ and $\vec{x}_j$ are the centers of cells i and j respectively. $\nabla\phi_{i,j}$ is the gradient of ϕ at the center of the face between cells i and j, and $f_i = f(\vec{x}_i)$ is the value of the right hand side of Equation 4.1 at the center of cell i. We then discretize the normal derivatives of ϕ using second order accurate centered finite differencing as follows:

$$\vec{n}_{i,j} \cdot \nabla\phi_{i,j} = (\phi_j - \phi_i)/|\vec{x}_j - \vec{x}_i| \tag{4.9}$$

where ϕ_i and ϕ_j are the values of ϕ located at the centers of cells i and j respectively. Since we can view our Voronoi discretization as a global mesh with regular connectivity we can formulate our final system as a single set of equations over the entire mesh using our definitions for the gradient and divergence operators in Equations 4.8 and 4.9. After taking into account the boundary conditions and rearranging terms, we arrive at the following symmetric positive definite system with orthogonal gradient components computed at each face:

$$-\mathbf{V}_c\mathbf{D}\mathbf{G}\boldsymbol{\phi} = -\mathbf{V}_c\mathbf{f} + \mathbf{V}_c\mathbf{D}\left(\mathbf{G}_d\boldsymbol{\phi}_d + \frac{\partial\boldsymbol{\phi}_n}{\partial\vec{n}}\right) \tag{4.10}$$

where $\boldsymbol{\phi}$ contains the discrete values of ϕ at non-removed cells over the entire Chimera grid, $\mathbf{D}$ is the discrete divergence and $\mathbf{G}$ is the discrete gradient. $\mathbf{V}_c$ is a matrix with the Voronoi cell volumes as entries along the diagonal. Note that Equation 4.10 is the

volume weighted form of Equation 4.1 in order to maintain symmetry. In order to solve Equation 4.10 we use incomplete Cholesky preconditioned conjugate gradient.

4.4 Numerical Results

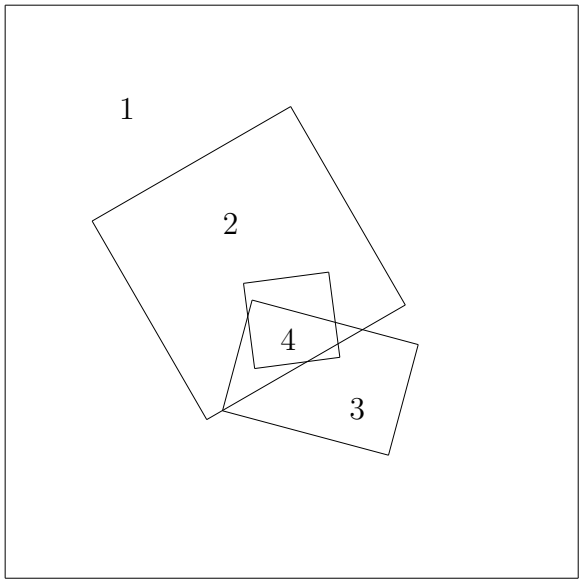


Figure 4.6: The domains of the grids used in our Poisson equation tests in two spatial dimensions.

In order to examine the convergence of our spatial discretization we have implemented several convergence tests in both two and three spatial dimensions. In all tests we used Dirichlet boundary conditions set along the exterior computational boundary.

In two spatial dimensions we use the domain $\Omega = [-1, 1] \times [-1, 1]$ for all tests which is discretized by four overlapping grids as listed in Table 4.1 and shown in Figure 4.6. Table 4.2 gives the errors and orders of accuracy for $\phi(x, y) = sin(\pi x)sin(\pi y)$. Table 4.3 gives the errors and orders of accuracy for $\phi(x, y) = e^x(x^2 sin(y) + y^2)$.

	Object space domain	Δx	$\vec{s}$	θ
1	$[-1,1] \times [-1,1]$	$2/n$	$(0,0)$	0
2	$[-.4,.4] \times [-.4,.4]$	$.8/n$	$(-.15,.1)$	$\pi/6$
3	$[-.3,.3] \times [-.2,.2]$	$.4/n$	$(.1,-.3)$	$-\pi/12$
4	$[-.15,.15] \times [-.15,.15]$	$.15/n$	$(0,-.1)$	$\pi/24$

Table 4.1: Domains, cell sizes, positions ($\vec{s}$) and orientations (θ) of the four grids used in our Poisson equation tests in two spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid. Note that all cells on all grids are square, the 3rd grid is rectangular with more square grid cells in one direction, and the 4th grid is extra fine where the numerator is correctly listed as .15 in the table. See also Figure 4.6.

n	L^1 Error	Order	L^∞ Error	Order
32	1.88×10^{-3}	–	6.25×10^{-3}	–
64	4.38×10^{-4}	2.10	1.72×10^{-3}	1.93
128	1.08×10^{-4}	2.02	4.33×10^{-4}	1.99
256	2.68×10^{-5}	2.01	1.16×10^{-4}	1.90
512	6.67×10^{-6}	2.01	2.73×10^{-5}	2.09

Table 4.2: Convergence results for solving a Poisson equation with analytic solution $\phi(x,y) = \sin(\pi x)\sin(\pi y)$.

n	L^1 Error	Order	L^∞ Error	Order
32	2.82×10^{-4}	–	1.41×10^{-3}	–
64	6.75×10^{-5}	2.06	3.69×10^{-4}	1.94
128	1.66×10^{-5}	2.03	9.87×10^{-5}	1.90
256	4.10×10^{-6}	2.01	2.55×10^{-5}	1.95
512	1.02×10^{-6}	2.01	6.62×10^{-6}	1.95

Table 4.3: Convergence results for solving a Poisson equation with analytic solution $\phi(x,y) = e^x(x^2 sin(x) + y^2)$.

In three spatial dimensions we use the domain $\Omega = [-1,1] \times [-1,1] \times [-1,1]$ for all tests which is discretized by four overlapping grids as listed in Table 4.4 and shown in Figure 4.7. Table 4.5 gives the errors and orders of accuracy for $\phi(x,y,z) = sin(\pi x)sin(\pi y)sin(\pi z)$. Table 4.6 gives the errors and orders of accuracy for $\phi(x,y) = e^{-x^2-y^2-z^2}$. Table 4.7 gives the errors and orders of accuracy for $\phi(x,y,z) = e^x+e^y+e^z$.

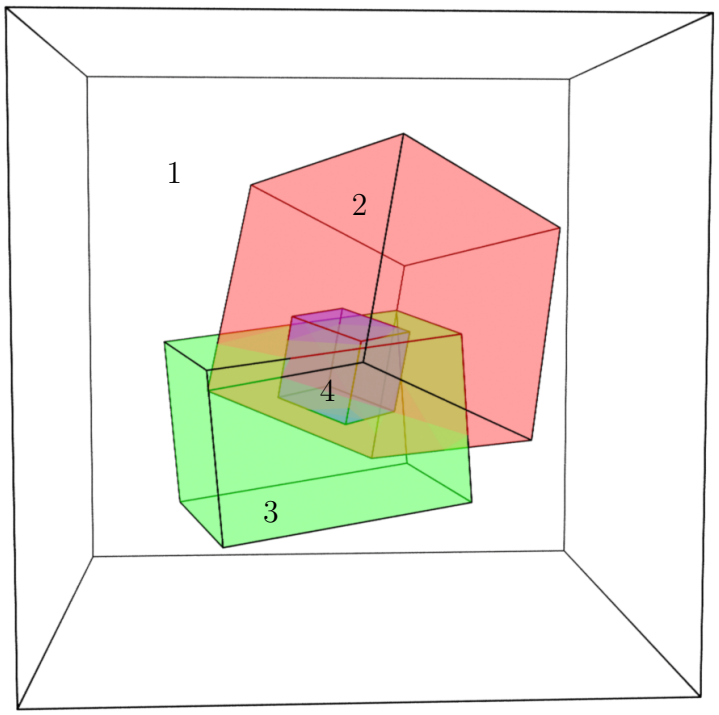


Figure 4.7: The domains of the grids used in our Poisson equation tests in three spatial dimensions.

	Object space domain	Δx	$\vec{s}$	$\theta, \vec{a}$
1	$[-1,1] \times [-1,1] \times [-1,1]$	$2/n$	$(0,0,0)$	$0, (0,0,0)$
2	$[-.4,.4] \times [-.4,.4] \times [-.4,.4]$	$0.8/n$	$(-.15,.1,0)$	$\pi/4, (1/\sqrt{6}, 2/\sqrt{6}, 1/\sqrt{6})$
3	$[-.45,.45] \times [-.3,.3] \times [-.3,.3]$	$0.6/n$	$(.1,-.3,0)$	$\pi/10, (-1/\sqrt{11}, 3/\sqrt{11}, -1/\sqrt{11})$
4	$[-.15,.15] \times [-.15,.15] \times [-.15,.15]$	$0.3/n$	$(0,-.1,0)$	$\pi/2, (4/\sqrt{21}, -1/\sqrt{21}, 2/\sqrt{21})$

Table 4.4: Domains, cell sizes, positions and orientations (angle θ, axis $\vec{a}$) of the four grids used in our Poisson tests in three spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid.

n	L^1 Error	Order	L^∞ Error	Order
16	4.36×10^{-3}	–	2.14×10^{-2}	–
32	1.00×10^{-3}	2.12	6.21×10^{-3}	1.78
64	2.39×10^{-4}	2.07	1.70×10^{-3}	1.87
128	5.82×10^{-5}	2.04	5.05×10^{-4}	1.75
256	1.44×10^{-5}	2.02	1.44×10^{-4}	1.81

Table 4.5: Convergence results for solving a Poisson equation with analytic solution $\phi(x,y,z) = sin(\pi x)sin(\pi y)sin(\pi z)$.

n	L^1 Error	Order	L^∞ Error	Order
16	2.10×10^{-3}	–	7.45×10^{-3}	–
32	4.94×10^{-4}	2.09	1.68×10^{-3}	2.15
64	1.20×10^{-4}	2.04	4.26×10^{-4}	1.98
128	2.95×10^{-5}	2.02	1.09×10^{-4}	1.97
256	7.33×10^{-6}	2.01	2.86×10^{-5}	1.93

Table 4.6: Convergence results for solving a Poisson equation with analytic solution $\phi(x,y,z) = e^{-x^2-y^2-z^2}$.

n	L^1 Error	Order	L^∞ Error	Order
16	5.70×10^{-4}	–	2.46×10^{-3}	–
32	1.27×10^{-4}	2.16	6.37×10^{-4}	1.95
64	3.00×10^{-5}	2.09	1.70×10^{-4}	1.91
128	7.28×10^{-6}	2.04	4.38×10^{-5}	1.95
256	1.79×10^{-6}	2.02	1.13×10^{-5}	1.96

Table 4.7: Convergence results for solving a Poisson equation with analytic solution $\phi(x,y,z) = e^x + e^y + e^z$.

4.5 Matrix Conditioning

In order to examine the conditioning of the matrix produced by this spatial discretization, we have compared the number of conjugate gradient (CG) and incomplete Cholesky preconditioned conjugate gradient (ICPCG) iterations required to satisfy $|\mathbf{V}_c^{-1}\mathbf{r}|_\infty < 10^{-8}$ where $\mathbf{r}$ is the current residual of Equation 4.10. Note that we multiply the residual by $\mathbf{V}_c^{-1}$ since in this test we are interested in the residual of the unweighted discretized Poisson equation. We note that each iteration of ICPCG takes roughly 2.5 times longer than an iteration of CG due to the backwards and forwards substitutions performed when applying the preconditioner. The iteration counts for both two- and three-dimensional examples are shown in Table 4.8. Notice that the preconditioner works, significantly reducing the iteration counts, and that even with the added cost of applying the preconditioner there is a large computational saving.

	$\phi(x,y) = \sin(\pi x)\sin(\pi y)$		$\phi(x,y,z) = sin(\pi x)sin(\pi y)sin(\pi z)$	
n	CG	ICPCG	CG	ICPCG
16	511	37	235	33
32	1955	61	702	53
64	6323	110	1833	99
128	22715	328	6702	225
256	96511	795	24677	492
512	385218	4028	-	-

Table 4.8: The number of iterations taken by CG and ICPCG in order to converge for successively finer resolutions in both two and three dimensions.

Chapter 5

Diffusion Discretization

5.1 Scalar Diffusion Equation

We next modify our Poisson solver to solve the following diffusion equation:

$$\frac{\partial \phi}{\partial t} = \nabla \cdot \beta(\vec{x}) \nabla \phi(\vec{x}), \qquad \vec{x} \in \Omega \tag{5.1}$$

$$\phi(\vec{x}) = g(\vec{x}, t), \qquad \vec{x} \in \partial\Omega_D \tag{5.2}$$

$$\vec{n}(\vec{x}) \cdot \nabla \phi(\vec{x}) = h(\vec{x}, t), \qquad \vec{x} \in \partial\Omega_N \tag{5.3}$$

where β is the diffusion coefficient. We first solve Equations 5.1-5.3 for values of ϕ located at cell centers by modifying Equation 4.10 from Section 4.3 to implement a backward Euler time integration scheme by solving the following symmetric positive definite system:

$$\left(\mathbf{V}_c - \Delta t \beta \mathbf{V}_c \mathbf{D} \mathbf{G}\right) \boldsymbol{\phi}^{n+1} = \mathbf{V}_c \boldsymbol{\phi}^n + \Delta t \beta \mathbf{V}_c \mathbf{D} \left(\mathbf{G}_d \boldsymbol{\phi}_d^{n+1} + \frac{\partial \boldsymbol{\phi}_n^{n+1}}{\partial \vec{n}} \right) \tag{5.4}$$

where ϕ^n and ϕ^{n+1} are the discrete ϕ values at non-removed cells at times t^n and t^{n+1} respectively. We assume β to be spatially constant for the sake of exposition. Note that Equation 5.4 is the volume weighted form of Equation 5.1 in order to maintain

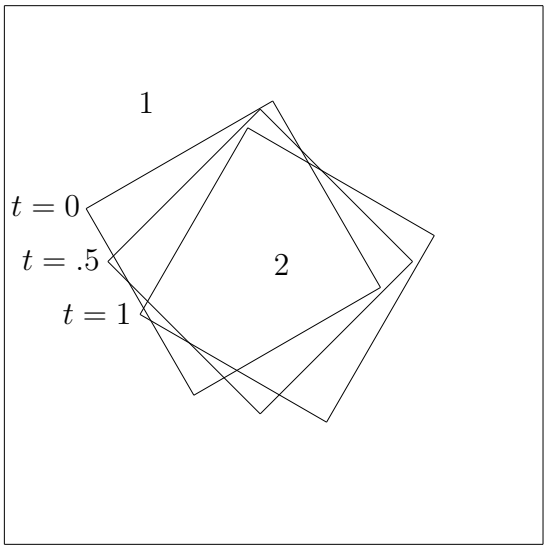


Figure 5.1: The domains of the grids used in our diffusion equation tests in 2 spacial dimensions with the second grid shown at its time $t = 0$, $t = .5$ and $t = 1$ positions and orientations.

symmetry. For second order accuracy we use a trapezoid rule time integration scheme by solving the following equation:

$$\left(\mathbf{V}_c - \frac{\Delta t}{2}\beta\mathbf{V}_c\mathbf{DG}\right)\boldsymbol{\phi}^{n+1} = \left(\mathbf{V}_c + \frac{\Delta t}{2}\beta\mathbf{V}_c\mathbf{DG}\right)\boldsymbol{\phi}^{n} + \Delta t\beta\mathbf{V}_c\mathbf{D}\left(\mathbf{G}_d\frac{\boldsymbol{\phi}_d^{n+1}+\boldsymbol{\phi}_d^{n}}{2} + \frac{\frac{\partial\boldsymbol{\phi}_n^{n+1}}{\partial\vec{n}} + \frac{\partial\boldsymbol{\phi}_n^{n}}{\partial\vec{n}}}{2}\right) \tag{5.5}$$

In two spatial dimensions we use the domain $\Omega = [-1, 1] \times [-1, 1]$ which is discretized by two overlapping grids as listed in Table 5.1 and shown in Figure 5.1. In all of our diffusion equation tests we set $\beta = .01$ and specify Dirichlet boundary conditions along the exterior computational boundary. In each test we integrate the solution from time $t = 0$ to $t = 1$ using the time step $\Delta t \approx 1/n$ and compute the errors at time $t = 1$. Note that for now we only consider case where the second grid is stationary and remains at its initial time $t = 0$ location. We first consider the exact solution as given by $\phi(x, y, t) = e^{-.02\pi^2 t}sin(\pi x)sin(\pi y)$. Tables 5.2 and 5.3 show the results for backward euler and trapezoid rule time integration on a stationary grid.

	Object space domain	Δx	$\vec{s}$	θ
1	$[-1,1] \times [-1,1]$	$2/n$	$(0,0)$	0
2	$[-.4,.4] \times [-.4,.4]$	$.8/n$	$(-.15,.1) + t(.2,-.1)$	$(1+t)\pi/6$

Table 5.1: Domains, cell sizes, positions ($\vec{s}$) and orientations (θ) of the four grids used in our diffusion equation tests in two spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid.

n	L^1 Error	Order	L^∞ Error	Order
32	4.34×10^{-4}	–	2.41×10^{-3}	–
64	1.56×10^{-4}	1.48	6.74×10^{-4}	1.84
128	6.35×10^{-5}	1.29	1.59×10^{-4}	2.09
256	2.79×10^{-5}	1.18	6.99×10^{-5}	1.18
512	1.32×10^{-5}	1.08	3.30×10^{-5}	1.08
1024	6.46×10^{-6}	1.03	1.61×10^{-5}	1.04

Table 5.2: Convergence results for solving a diffusion equation with analytic solution $\phi(x,y,t) = e^{-.02\pi^2 t} sin(\pi x) sin(\pi y)$ on two stationary grids using backward Euler time integration.

n	L^1 Error	Order	L^∞ Error	Order
32	2.86×10^{-4}	–	2.39×10^{-3}	–
64	7.05×10^{-5}	2.02	7.35×10^{-4}	1.70
128	1.72×10^{-5}	2.04	1.82×10^{-4}	2.02
256	4.26×10^{-6}	2.01	4.57×10^{-5}	1.99
512	1.06×10^{-6}	2.01	1.17×10^{-5}	1.97
1024	2.65×10^{-7}	2.00	2.90×10^{-6}	2.01

Table 5.3: Convergence results for solving a diffusion equation with analytic solution $\phi(x,y,t) = e^{-.02\pi^2 t} sin(\pi x) sin(\pi y)$ on two stationary grids using trapezoid rule time integration.

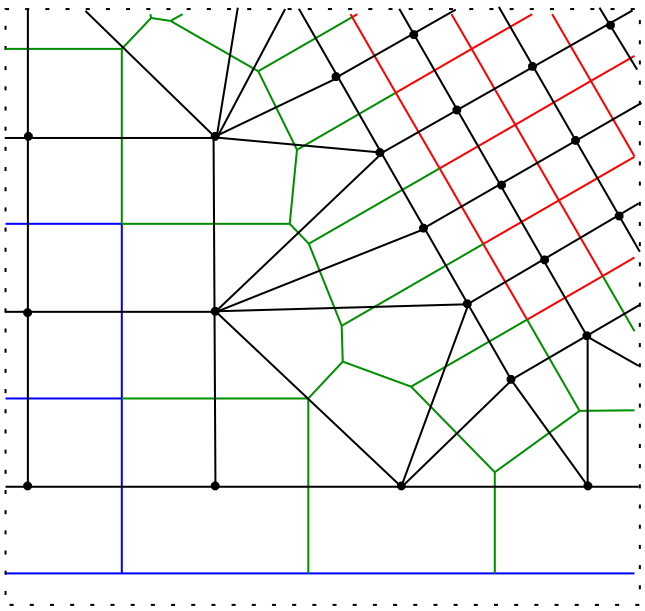


Figure 5.2: When interpolating between cell centered values on the Voronoi mesh (drawn as blue and red faces which correspond to the Cartesian faces taken from the coarse and fine grids respectively, and green faces which correspond to unstructured Voronoi faces), we use a hybrid interpolation scheme. The interpolation mesh is drawn in black overtop the Voronoi mesh. If an interpolation location lies within a square on the interpolation mesh, bilinear interpolation is applied. Otherwise, if an interpolation location lies within a triangle of the Delaunay mesh dual of the unstructured part of the Voronoi diagram, barycentric interpolation over that triangle is applied. Note that in the former case, if we were using the full Delaunay mesh dual of the Voronoi diagram these squares would be tesselated with triangles and only barycentric interpolation would be used. However, this would reduce accuracy.

5.1.1 Moving Grids

In order to treat moving grids we remap ϕ values at the beginning each of step using the semi-Lagrangian advection schemes from Section 3.2 applied with a zero velocity field to calculate time t^n values of ϕ on the grids in their time t^{n+1} locations. In order to apply the semi-Lagrangian schemes we first need to define ϕ at every location on every grid in order to compute updated values for removed cells not included in the implicit solve. This requires interpolating from the Voronoi degrees of freedom back to the Cartesian grid degrees of freedom that were removed when

constructing the Voronoi mesh. We accomplish this by interpolating over a modified version of the Delaunay mesh dual of the Voronoi mesh as illustrated in Figure 5.2. If a removed degree of freedom lies within the support of non-removed degrees of freedom from one of the Cartesian grids, we simply use multilinear interpolation. Otherwise, the interpolation is slightly more intricate and needs to be accomplished using the aggregate Voronoi mesh, in which case we interpolate values of ϕ by using barycentric coordinates to interpolate across the tetrahedra belonging to the Delaunay mesh dual of the Voronoi diagram. We note that since we do not have the exact connectivity of our Voronoi mesh, we allow overlapping tetrahedra in (near) degenerate cases in order to guarantee that valid interpolation stencils exist for all interpolation locations.

When using the first order accurate semi-Lagrangian Advection scheme to remap values we found that the overall scheme degenerates to first order even when using the second order accurate version of the method in Equation 5.5. We found that this can be alleviated by using the second order accurate SL-MacCormack advection scheme from Section 3.2 to remap time t^n values of ϕ.

We now consider the case where the second grid is allowed to move. Using the same analytic function as in the stationary case, Tables 5.4 and 5.5 show the results for backward euler and trapezoid rule time integration using semi-Lagrangian and SL-MacCormack remapping respectively. Finally, table 5.6 shows that using semi-Lagrangian remapping degenerates the trapezoid rule time integration scheme towards first order as compared to when using SL-MacCormack remapping.

n	L^1 Error	Order	L^∞ Error	Order
32	1.82×10^{-3}	–	1.89×10^{-2}	–
64	7.13×10^{-4}	1.35	8.12×10^{-3}	1.22
128	3.06×10^{-4}	1.22	3.71×10^{-3}	1.13
256	1.42×10^{-4}	1.11	1.83×10^{-3}	1.02
512	6.82×10^{-5}	1.06	9.02×10^{-4}	1.02
1024	3.33×10^{-5}	1.03	4.47×10^{-4}	1.01

Table 5.4: Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t} sin(\pi x) sin(\pi y)$ on one stationary grid and one moving grid using semi-Lagrangian remapping and backward euler time integration.

n	L^1 Error	Order	L^∞ Error	Order
32	5.68×10^{-4}	–	1.04×10^{-2}	–
64	1.34×10^{-4}	2.08	2.33×10^{-3}	2.16
128	3.29×10^{-5}	2.03	5.75×10^{-4}	2.02
256	8.40×10^{-6}	1.97	1.41×10^{-4}	2.02
512	2.09×10^{-6}	2.01	3.44×10^{-5}	2.04
1024	5.26×10^{-7}	1.99	8.77×10^{-6}	1.97

Table 5.5: Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t} sin(\pi x) sin(\pi y)$ on one stationary grid and one moving grid using SL-MacCormack remapping and trapezoid rule time integration.

n	L^1 Error	Order	L^∞ Error	Order
32	1.77×10^{-3}	–	1.90×10^{-2}	–
64	6.86×10^{-4}	1.37	8.22×10^{-3}	1.21
128	2.93×10^{-4}	1.23	3.76×10^{-3}	1.13
256	1.36×10^{-4}	1.11	1.85×10^{-3}	1.02
512	6.51×10^{-5}	1.06	9.14×10^{-4}	1.02
1024	3.18×10^{-5}	1.03	4.53×10^{-4}	1.01

Table 5.6: Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, t) = e^{-.02\pi^2 t} sin(\pi x) sin(\pi y)$ on a moving grid using semi-Lagrangian remapping and trapezoid rule time integration.

In three spatial dimensions we use the domain $\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$ which is discretized by two overlapping grids as listed in Table 5.7 and shown in Figure 5.3. We consider solving for ϕ values at cell centers for the analytic function $\phi(x, y, z, t) = e^{-.03\pi^2 t} sin(\pi x) sin(\pi y) sin(\pi z)$. Table 5.8 gives the errors and orders of accuracy when using trapezoid rule time integration and SL-MacCormack remapping.

	Object space domain	Δx	$\vec{s}$	$\theta, \vec{a}$
1	$[-1, 1] \times [-1, 1] \times [-1, 1]$	$2/n$	$(0, 0, 0)$	$0, (0, 0, 0)$
2	$[-.4, .4] \times [-.4, .4] \times [-.4, .4]$	$0.8/n$	$(-.15, .1, 0) + t(.2, -.1, .05)$	$(1 + t)\pi/4, (1/\sqrt{6}, 2/\sqrt{6}, 1/\sqrt{6})$

Table 5.7: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ, axis $\vec{a}$) of the two grids used in our diffusion equation tests in three spatial dimensions. n indicates the number of cells in each dimension on the coarsest grid.

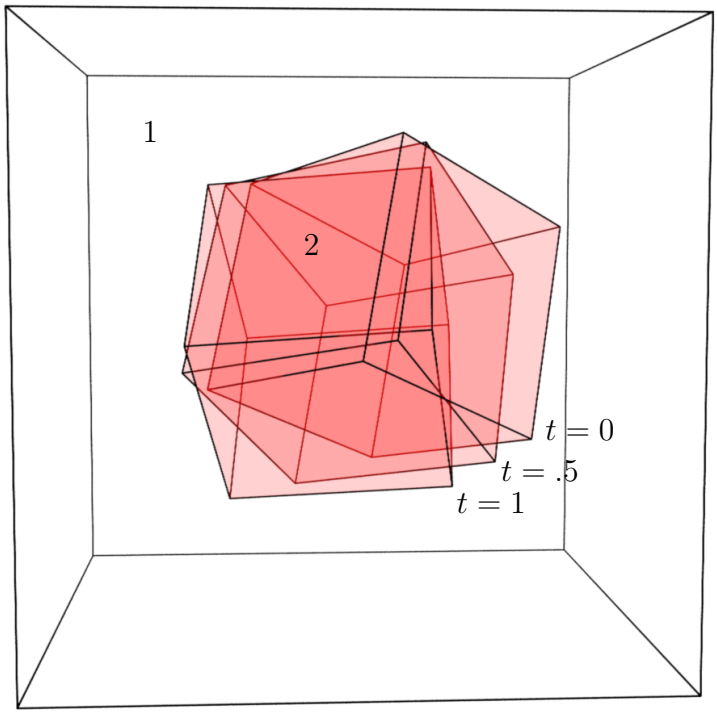


Figure 5.3: The domains of the grids used in our diffusion tests in three spatial dimensions with the second grid shown at its time $t = 0$, $t = .5$ and $t = 1$ positions and orientations.

n	L^1 Error	Order	L^∞ Error	Order
32	3.09×10^{-4}	–	9.95×10^{-3}	–
64	7.54×10^{-5}	2.04	2.64×10^{-3}	1.91
128	1.86×10^{-5}	2.02	6.64×10^{-4}	1.99
256	4.61×10^{-6}	2.01	1.77×10^{-4}	1.90

Table 5.8: Convergence results for solving a diffusion equation with analytic solution $\phi(x, y, z, t) = e^{-.03\pi^2 t} sin(\pi x) sin(\pi y) sin(\pi z)$ on one stationary grid and one moving grid using SL-MacCormack remapping and trapezoid rule time integration.

5.2 Navier-Stokes Viscosity

For spatially constant viscosity, the viscous terms in the Navier-Stokes equations can be treated in an implicit mannger by independently solving a scalar diffusion-like operator for each of the components of the velocity. When using standard MAC grids, this becomes problematic when one grid is rotated with respect to another since the cleanly separated MAC grid degrees of freedom on one grid are mixed when considered using the coordinate system on the other grid. Thus we compute a world space velocity vector at each cell center by averaging the samples stored at incident faces and then rotating the resulting vector into world space. We then apply the cell based diffusion equation separately and independently in each component direction, i.e. for each component of the cell centered vector field. Notably, this does not require constructing an additional mesh, as we can reuse the one that will be used for the pressure Poisson solve. After applying the viscous update to the cell center velocity components, we could interpolate these back to the grid degrees of freedom but this increases numerical dissipation. Instead, one could interpolate the time t^{n+1} values back to the original Cartesian grid cell center degrees of freedom, compute differences with the time t^n values, and then map these differences back to the face degrees of freedom–however, this also seemed to lower the order of accuracy. Therefore, we compute differences between the time t^n and time t^{n+1} values directly on the Voronoi cell center degrees of freedom and then interpolate these differences back to the removed cells and a one layer thick band of ghost cells on each grid before distributing these differences back to the faces. In order to prevent numerical drift in overlapped regions, faces incident only to removed cells are updated by averaging the interpolated time t^{n+1} values at incident cell centers. We note that applying slip boundary conditions to a rotated grid would cause the velocity components to no longer be cleanly separated, however, we do not consider this case.

We once again consider the moving grids in Table 5.1, Figure 5.1, Table 5.7 and Figure 5.3. In two spatial dimensions we consider the analytic vector valued function $\vec{\phi}(x, y, t) = (e^{-.02\pi^2 t} sin(\pi x), sin(\pi y), e^{-.13\pi^2 t} sin(2\pi x) sin(3\pi y))$. Table 5.9 gives

the errors and orders of accuracy when using trapezoid time integration and SL-MacCormack remapping. In three spatial dimensions we consider the analytic vector valued function

$$\vec{\phi}(x,y,z,t) = \begin{pmatrix} e^{-.03\pi^2 t} sin(\pi x) sin(\pi y) sin(\pi z) \\ e^{-.12\pi^2 t} sin(2\pi x) sin(2\pi y) sin(2\pi z) \\ e^{-.14\pi^2 t} sin(\pi x) sin(2\pi y) sin(3\pi z) \end{pmatrix} \tag{5.6}$$

Table 5.10 gives the errors and orders of accuracy when using trapezoid time integration and SL-MacCormack remapping.

n	L^1 Error	Order	L^∞ Error	Order
32	7.47×10^{-3}	–	4.41×10^{-2}	–
64	1.82×10^{-3}	2.04	1.01×10^{-2}	2.13
128	4.50×10^{-4}	2.02	2.51×10^{-3}	2.01
256	1.12×10^{-4}	2.01	6.26×10^{-4}	2.00
512	2.78×10^{-5}	2.00	1.56×10^{-4}	2.00
1024	6.94×10^{-6}	2.00	3.90×10^{-5}	2.00

Table 5.9: Convergence results for solving a separate diffusion equation in each direction with analytic solution $\vec{\phi}(x,y,t) = (e^{-.02\pi^2 t} sin(\pi x) sin(\pi y), e^{-.13\pi^2 t} sin(2\pi x) sin(3\pi y))$ on one stationary and one moving grid using SL-MacCormack remapping and trapezoid rule time integration.

n	L^1 Error	Order	L^∞ Error	Order
32	4.99×10^{-3}	–	3.98×10^{-2}	–
64	1.23×10^{-3}	2.03	1.01×10^{-2}	1.98
128	3.04×10^{-4}	2.01	2.49×10^{-3}	2.02
256	7.55×10^{-5}	2.01	5.97×10^{-4}	2.06

Table 5.10: Convergence results for solving a separate diffusion equation in each direction with analytic solution $\vec{\phi}(x,y,t) = (e^{-.03\pi^2 t} sin(\pi x) sin(\pi y) sin(\pi z), e^{-.12\pi^2 t} sin(2\pi x) sin(2\pi y) sin(2\pi z), e^{-.14\pi^2 t} sin(\pi x) sin(2\pi y) sin(3\pi z))$ on one stationary grid and one moving grid using SL-MacCormack remapping and trapezoid rule time integration.

5.3 Monolithically Coupled Formulation

There are several cases where the component-wise solution approach from Section 5.2 poses issues. For example, when applying slip boundary conditions the components of the velocity are coupled together along the boundaries or in the case of spatially varying viscosity where the diffusion equations are coupled across spatial dimensions. One approach to simplify this was considered in [89] where the coupling terms were treated explicitly in order to separate the solve into three separate diffusion equations. Although we do not consider spatially varying viscosity in this paper or non-axis aligned slip boundary conditions, we briefly consider a fully coupled solve along the lines of [93] which does not require interpolating back and forth (or in our case interpolating in one direction and mapping the differences back in the other direction).

Since the following exercise is only done for the purpose of illustration we consider solving on a fixed Voronoi mesh only and do not map back and forth from the Cartesian grids. In addition we only consider Dirichlet boundary conditions. With these simplifications our approach to a coupled solver is as follows. We start with component values of $\vec{\phi}$ on each Voronoi face and stack all the internal faces into a single vector $\boldsymbol{\phi}_f$, and all the boundary faces with Dirichlet boundary conditions into $\boldsymbol{\phi}_{f,d}$. For each component of the velocity in world space we use unweighted least squares to interpolate from incident Voronoi faces to cell centers, i.e. $\boldsymbol{\phi}_x = \mathbf{W}_x\boldsymbol{\phi}_f + \mathbf{W}_{x,d}\boldsymbol{\phi}_{f,d}$ and $\boldsymbol{\phi}_y = \mathbf{W}_y\boldsymbol{\phi}_f + \mathbf{W}_{y,d}\boldsymbol{\phi}_{f,d}$. In order to compute the gradient at faces along the domain boundary, it is necessary to have $\vec{\phi}$ values at ghost cells across faces with Dirichlet boundary conditions, which we denote as $\boldsymbol{\phi}_{x,d}$ and $\boldsymbol{\phi}_{y,d}$. Using $\boldsymbol{\phi}_x$, $\boldsymbol{\phi}_y$, $\boldsymbol{\phi}_{x,d}$, and $\boldsymbol{\phi}_{y,d}$ we can discretize the viscous forces at cell centers and then conservatively distribute these forces back to the Voronoi faces by multiplying by $\mathbf{W}_x^T$ and $\mathbf{W}_y^T$. Using backward Euler time integration to integrate these forces we arrive at the following symmetric positive definite system for fully coupled face unknowns where we

have stacked $\mathbf{W}_x$ and $\mathbf{W}_y$ into $\mathbf{W}$:

$$\left(\mathbf{V}_f - \Delta t\beta\mathbf{W}^T\begin{pmatrix}\mathbf{V}_c\mathbf{DG} & \mathbf{0}\\ \mathbf{0} & \mathbf{V}_c\mathbf{DG}\end{pmatrix}\mathbf{W}\right)\boldsymbol{\phi}_f^{n+1} = \mathbf{V}_f\boldsymbol{\phi}_f^n + \Delta t\beta\mathbf{W}^T\begin{pmatrix}\mathbf{V}_c\mathbf{DG} & \mathbf{0}\\ \mathbf{0} & \mathbf{V}_c\mathbf{DG}\end{pmatrix}\mathbf{W}_d\boldsymbol{\phi}_{f,d}^{n+1} + \Delta t\beta\mathbf{W}^T\begin{pmatrix}\mathbf{V}_c\mathbf{DG}_d\boldsymbol{\phi}_{x,d}^{n+1}\\ \mathbf{V}_c\mathbf{DG}_d\boldsymbol{\phi}_{y,d}^{n+1}\end{pmatrix} \quad (5.7)$$

where $\mathbf{V}_f$ is a diagonal matrix of face dual cell volumes.

Table 5.11 gives the errors and orders of accuracy when applying this scheme with the analytic function $\vec{\phi}(x,y,t) = (e^{-.02\pi^2 t}sin(\pi x)sin(\pi y), e^{-.13\pi^2 t}sin(2\pi x)sin(3\pi y))$. In tests we found that solving Equation 5.7, over the previous method of solving for the updated values independently in each direction, did not reduce numerical dissipation. We did not experiment with trapezoid rule and other ways of raising the order of accuracy, or explore ways for handling moving grids by mapping back and forth with the original MAC grid degrees of freedom because we were not able to devise a workable preconditioner for this approach. We found that our incomplete Cholesky preconditioner actually led to more instead of less iterations.

n	L^1 Error	Order	L^∞ Error	Order
32	1.42×10^{-2}	–	2.08×10^{-1}	–
64	6.52×10^{-3}	1.12	1.06×10^{-1}	0.96
128	3.33×10^{-3}	0.97	8.33×10^{-2}	0.35
256	1.88×10^{-3}	0.83	5.51×10^{-2}	0.60

Table 5.11: Convergence results for solving a diffusion equation in each direction coupled in a monolithic system with analytic solution $\vec{\phi}(x,y,t) = (e^{-.02\pi^2 t}sin(\pi x)sin(\pi y), e^{-.12\pi^2 t}sin(2\pi x)sin(3\pi y))$ on two stationary grids using backward euler time integration.

Chapter 6

Incompressible Flow

6.1 Fluid Integration Scheme

We consider the incompressible Navier-Stokes equations as follows:

$$\rho\left(\frac{\partial\vec{u}}{\partial t}+\left(\vec{u}\cdot\nabla\right)\vec{u}\right)=-\nabla p+\mu\nabla^2\vec{u} \tag{6.1}$$

$$\nabla\cdot\vec{u}=0 \tag{6.2}$$

where ρ is the density, $\vec{u}$ is the velocity, p is the pressure and μ is the viscosity. In order to solve Equations 6.1 and 6.2 we use a splitting method (see [20]). The first step addresses the advective component as follows:

$$\hat{\vec{u}}=\vec{u}^n-(\vec{u}\cdot\nabla)\vec{u} \tag{6.3}$$

We solve Equation 6.3 using the second order accurate SL-MacCormack advection scheme from Section 3.2 to obtain an intermediate velocity field $\hat{\vec{u}}$. We address boundary conditions at the computational boundary by using constant extrapolation to fill ghost cells. After applying these boundary conditions, we proceed to advect values to every face, obtaining a valid velocity field everywhere. We then continue to add

viscous forces by solving the following equation:

$$\vec{u}^* = \hat{\vec{u}} + \frac{\Delta t}{\rho}\mu\nabla^2\vec{u}^* \tag{6.4}$$

We solve Equation 6.4 using a second order accurate time discretization (Equation 5.5) independently in each world space component direction as described in Section 5.2. In this step we handle inflow boundary conditions by specifying Dirichlet boundary conditions in all components of the velocity. Outflow boundary conditions are handled by specifying zero Neumann boundary conditions in all components of the velocity to prevent momentum from being exchanged across the boundary. Slip boundary conditions are handled by specifying Dirichlet boundary conditions in the normal component and zero Neuman boundary conditions in the tangential components (noting that in this case we assume the domain boundaries lie along axis aligned planes in order to simplify applying slip boundary conditions in the viscous solve as discussed in Section 5.3). After adding viscous forces we proceed to solve for pressure as follows:

$$\nabla^2 p = \frac{\rho}{\Delta t}\nabla\cdot\vec{u}^* \tag{6.5}$$

Since we solve Equation 6.5 on the Voronoi mesh it is first necessary to compute values of $\vec{u}^*$ for each face on the Voronoi mesh before we can compute the divergence of of the post viscosity velocities. Thus, we interpolate a velocity from the Cartesian grids for each face of the Voronoi mesh retaining only the component normal to that face. Note that most Voronoi faces coincide with Cartesian grid faces and thus interpolation is not required for these faces. Then denoting the vector of all these Voronoi face velocities as $\mathbf{u}^*$ we solve the following analog of Equation 4.10:

$$-\mathbf{V}_c\mathbf{D}\mathbf{G}\hat{\mathbf{p}} = -\mathbf{V}_c\mathbf{D}\mathbf{u}^* + \mathbf{V}_c\mathbf{D}\left(\mathbf{G}_d\hat{\mathbf{p}}_d - \mathbf{u}_n^{n+1}\right) \tag{6.6}$$

where $\hat{\mathbf{p}}$ are the pressures scaled by $\Delta t/\rho$, $\hat{\mathbf{p}}_d$ represents Dirichlet boundary conditions and $\mathbf{u}_n^{n+1}$ represents the Neumann boundary conditions implied by faces with fixed velocities. We handle inflow and slip boundary conditions by setting values of $\mathbf{u}_n^{n+1}$

at faces along the boundary. Outflow boundary conditions are handled by specifying zero Dirichlet boundary conditions. After solving for pressure we update the velocities as follows:

$$\vec{u}^{n+1} = \vec{u}^{*} - \frac{\Delta t}{\rho}\nabla p \tag{6.7}$$

Equation 6.7 is used to update the velocities at all faces on the Voronoi mesh using pressure gradients computed by differencing the pressure samples at incident non-removed cells as in Equation 4.9. In order to update Cartesian grid faces not coincident to faces on the Voronoi mesh we use an approach similar to that applied at the end of the viscous step in Section 5.2. First we compute a velocity vector at each non-removed cell center by using regularized linear least squares fit of the velocity components at incident faces on the Voronoi mesh. We then interpolate a full velocity vector at removed and ghost cell centers before computing the updated velocity components at removed Cartesian grid faces by averaging the velocities at incident Cartesian grid cell centers.

We address our object handling as follows. When advecting velocities we set velocity Dirichlet boundary conditions at faces whose centers lie inside objects. We then advect every face obtaining a valid velocity field everywhere after advection. In certain examples we also solve an advection equation for a passive scalar ϕ at cell centers for visualization purposes. In this case we handle objects by first creating a levelset for each object and then extrapolating ϕ in the normal direction using an $O(n\log n)$ fast marching type method as described in [1, 30]. In both the viscous and pressure steps we use a simple immersed boundary type approach and note that a higher order object handling approach would be straightforward to apply. In the viscous step, since the velocity is known inside objects, we simply specify Dirichlet boundary conditions at the non-removed cell centers inside objects using the pointwise object velocity. In the pressure solve we similarly set values of $\mathbf{u}_n^{n+1}$ at faces on the Voronoi mesh whose centers lie inside an object.

6.2 Numerical Results

We use the ghost cell parameters $\alpha_{\text{grid}} = 2$ and $\alpha_{\text{fluid}} = 1$ as described in Section 3.3. Due to the way the grids are chosen and the fact that they are all Cartesian we have considerable flexibility when deciding how to decompose the domain when allocating MPI processes as discussed in Section 2.3. In most of the simpler examples we allocate a separate MPI process per logical grid. For larger examples we split each logical grid into several subgrids each having their own MPI process in order to balance computational and memory loads among the computational nodes as described in Section 2.3. In the following subsections we list explicitly how the grids are subdivided and MPI processes are allocated in the captions of the corresponding grid configuration tables.

6.2.1 Two-dimensional Couette flow

We first consider a two-dimensional Couette flow where fluid flows horizontally between two walls. The bottom wall is stationary while the top moves from left to right with a horizontal velocity $u_0 = 1$. In this test we use the domain $[-1, 1] \times [-1, 1]$ and let $\rho = 1$, $\mu = .01$ and target pressure gradient $\frac{\partial p}{\partial x} = -.15$. With these conditions, the analytic solution is given as $\vec{u}(x, y, t) = (u_0 y + \frac{1}{2\mu}\frac{\partial p}{\partial x}(y^2 - y), 0)$. In this test we use analytic velocity boundary conditions on the top, left and bottom sides of the domain, and zero pressure outflow boundary conditions on the right side of the domain. We discretize the domain with a large stationary grid and insert a second finer moving grid as listed in Table 6.1. The second grid is not intended to add any additional detail or accuracy to the flow, but rather in this case we are demonstrating that it does not adversely affect the flow field. The u direction velocities on the x medial plane and pressures on the y medial plane are shown in Figure 6.1. Note that for both velocity and pressure the results convergence towards the analytic solution as the grid is refined and that the convergence rate corresponds to first order accuracy. Note that the sharp changes in the pressure visible in Figure 6.1 (Top) are at grid

boundaries and vary over time due to error incurred in the velocity field as a result of the remapping as the second grid moves. However, first order accurate convergence was found at each time. In addition, the results for the same test using only the background grid are shown producing nearly the analytic solution.

	Object space domain	Δx	$\vec{s}$	θ
1	$[0,1] \times [0,1]$	$1/n$	$(0,0)$	0
2	$[-.15,.15] \times [-.15,.15]$	$0.3/n$	$(.5,.5) + \cos(.5\pi t)(-.15,.05)$	$t\pi/6$

Table 6.1: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the two grids used in our Couette flow and lid driven cavity tests. n indicates the number of cells in each dimension on the coarsest grid. In the case of the lid driven cavity test, for Reynolds numbers 100 and 400, $n = 128$ was used and for Reynolds numbers 1000 and 5000, $n = 256$ was used. Additionally for the case of the lid driven cavity test, during parallel simulation each grid remained unsubdivided and was allocated a single MPI process since the number of cells on each grid was the identical.

6.2.2 Two-dimensional lid driven cavity

We consider a two-dimensional lid driven cavity and compare our results to those of [34]. In this test we use the domain $[-1,1] \times [-1,1]$ with zero normal and tangential velocity boundary conditions on each side except for the top of the domain along which we specify a tangential velocity of 1. We discretize the domain with a large stationary grid and insert a second finer moving grid as listed in Table 6.1 with the MPI subdivision parameters in the table's caption. The second grid is not intended to add any additional detail or accuracy to the flow, but rather in this case we are demonstrating that it does not adversely affect the flow field. The grid configurations and streamlines are shown in Figure 6.2 for Reynolds numbers 100, 400, 1000 and 5000. Our method produces the same vortices as observed by [34]. The u direction velocities on the x medial plane and v direction velocities on the y medial plane are shown in Figures 6.3 and 6.4 respectively. Our results are very close to those from [34], particularly for smaller Reynolds numbers. We also compared our results to values computed using only a single grid and found that they were nearly identical.

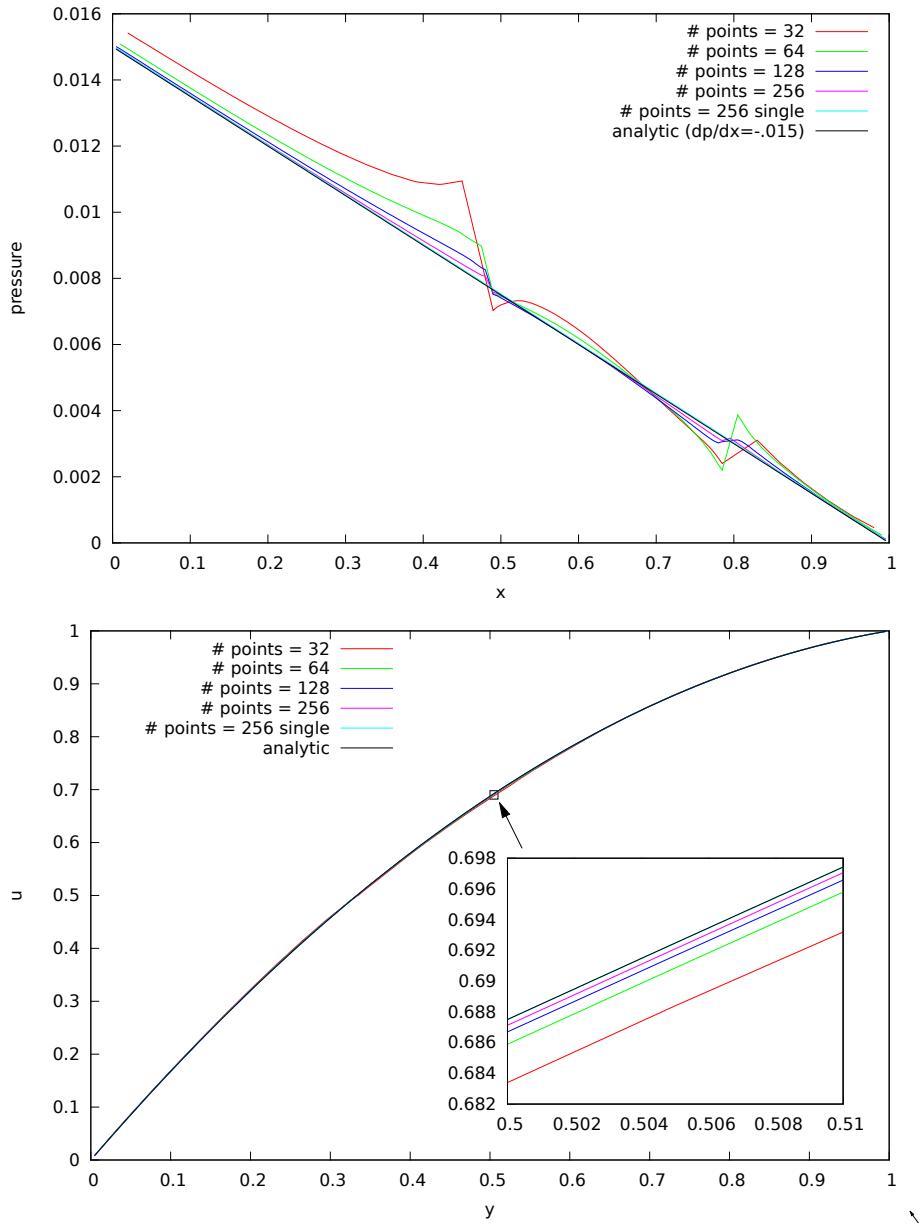
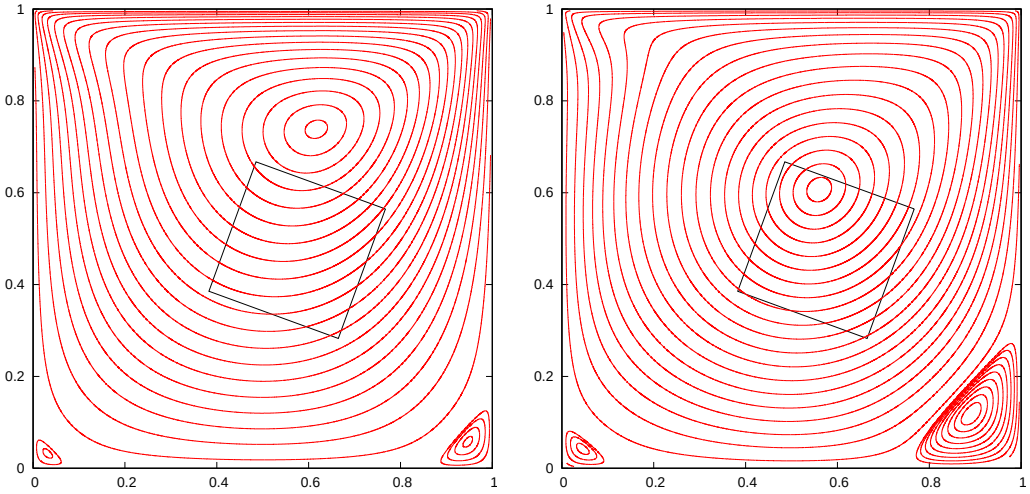


Figure 6.1: The results for our Couette flow example using two grids of varying resolutions and when using only the single coarser grid at the finest resolution. The analytic solution is also drawn. (Top) shows the pressure along the horizontal plane through the geometry center of the domain. (Bottom) shows the u velocities on the vertical plane through the geometric center of the domain. Note that the single grid simulation produces values nearly identical to that of the analytic solution while the two grid simulation includes a truncation error that vanishes under refinement demonstrating first order accuracy. Note that although this example is intended to examine the behavior of a steady state solution, since the second grid is moving a time varying error is introduced and the above plots are given for the solution at $t = 41.6667$ at which point the solution is well into its steady state regime. The pressure and velocity profiles at different times within this regime demonstrate similar convergence behavior.



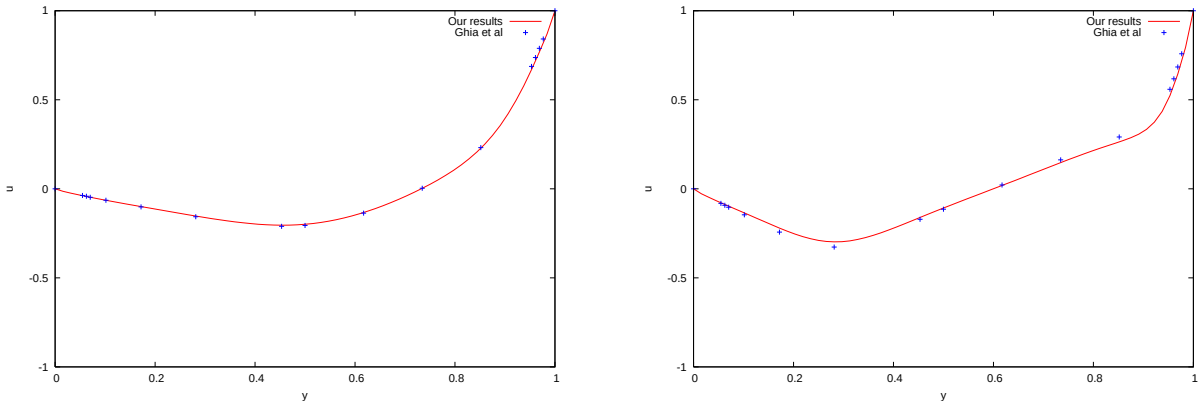
(a) Re=100

(b) Re=400

(c) Re=1000

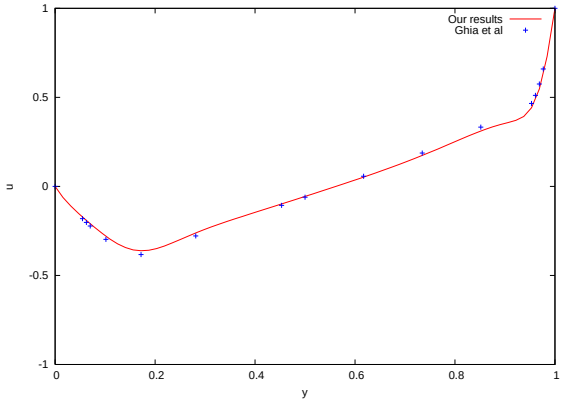
(d) Re=5000

Figure 6.2: Streamlines for the lid driven cavity example. Notice the tiny vortices at the bottom right corners of the Reynolds number 1000 and 5000 simulations and at the bottom left corner of the Reynolds number 5000 simulation. These vortices are the same as reported by [34].

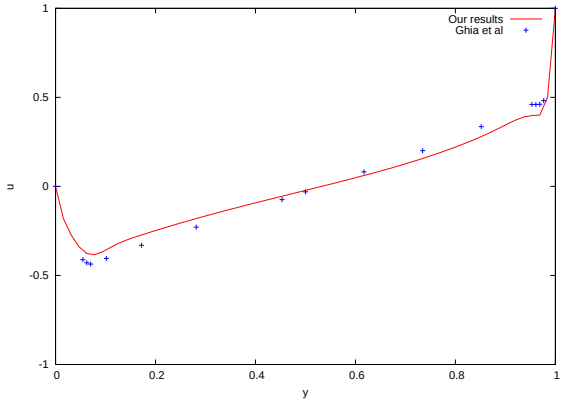


(a) Re=100

(b) Re=400

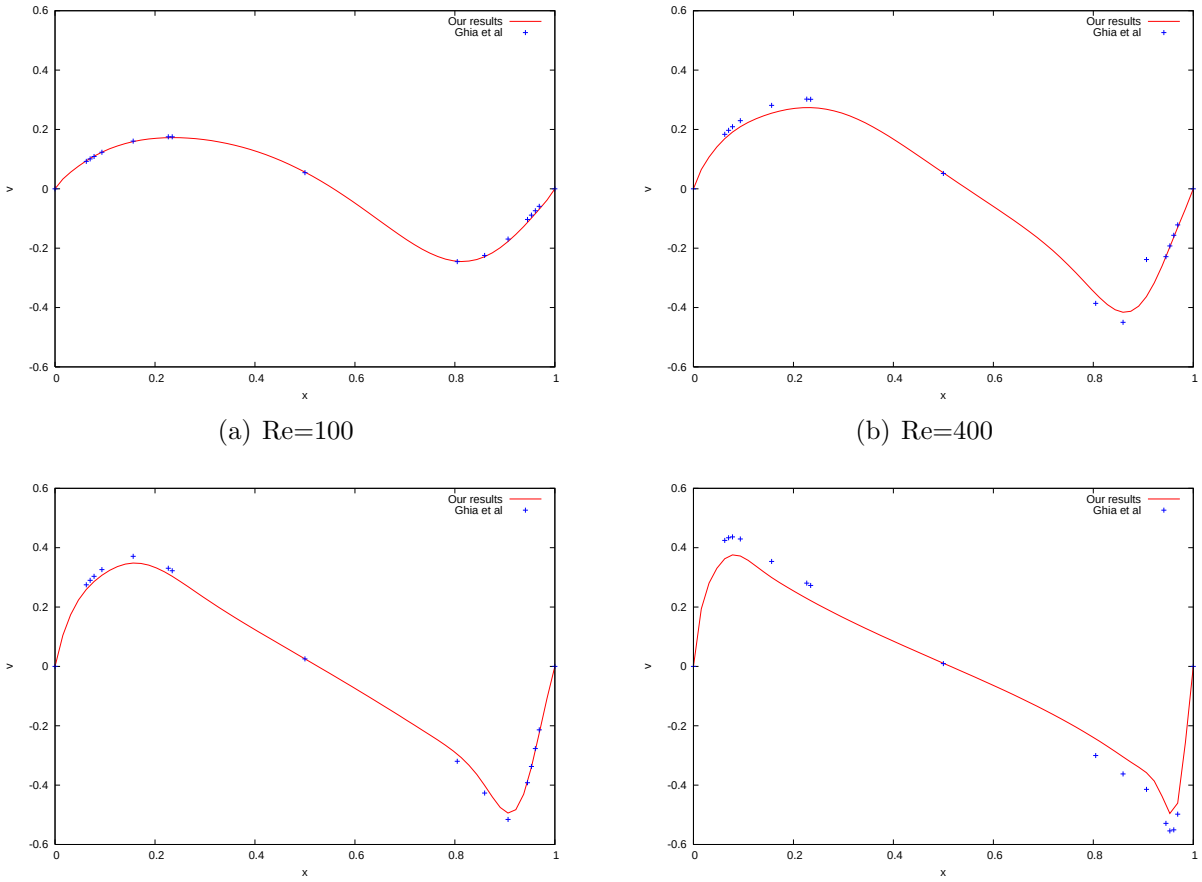


(c) Re=1000



(d) Re=5000

Figure 6.3: The u velocities on the vertical plane through the geometric center of cavity. The red lines correspond to our results and the blue '+' symbols correspond to the results from [34]. Note the good agreement between the results, particularly for smaller Reynolds numbers.



(a) Re=100

(b) Re=400

(c) Re=1000

(d) Re=5000

Figure 6.4: The v velocities on the horizontal line through the geometric center of cavity. The red lines correspond to our results and the blue '+' symbols correspond to the results from [34]. Note the good agreement between the results, particularly for smaller Reynolds numbers.

6.2.3 Two-dimensional moving vortex

In order to demonstrate the ability of our method to smoothly transition flow features across grid boundaries, we consider a vortex in a flow channel being transported from one grid to another stationary larger grid. We discretize the domain $[0,1] \times [0,1]$ with two grids as listed in Table 6.2, noting that we consider both the case where the second grid remains stationary at its $t = 0$ orientation and the case where the second grid rotates. We specify inflow boundary conditions at the left side of the domain, slip boundary conditions at the top and bottom walls and outflow boundary conditions at the right side of the domain. We give the initial time $t = 0$ velocity field as follows:

$$\vec{u}^0(\vec{x}) = (1,0)^T + \frac{(x_c - y, x - y_c)^T}{|\vec{x} - \vec{x}_c|} \begin{cases} e^{\left(\frac{-.25}{|\vec{x}-\vec{x}_c|/r - |\vec{x}-\vec{x}_c|^2/r^2}\right)} & \text{if } \|\vec{x} - \vec{x}_c\| < r \\ 0 & \text{if } \|\vec{x} - \vec{x}_c\| \geq r \end{cases} \tag{6.8}$$

where $\vec{x} = (x, y)^T$, $\vec{x}_c = (x_c, y_c)^T$ is the center of the vortex, and $r = .25$ is the diameter of the vortex. We plot the vorticity as the vortex is moving from one grid to the other in Figure 6.5 at time $t = .20833$. Note that no artifacts are observed at the grid boundaries when the fine grid is stationary or rotating. Figures 6.6 and 6.7 give the errors, computed using an $n = 1024$ simulation as a baseline, and orders of accuracy for the stationary and rotating grid cases respectively. We note that both the L^1 and L^∞ errors tend to zero implying self convergence and that the orders of accuracy tend towards first order as the grids are refined.

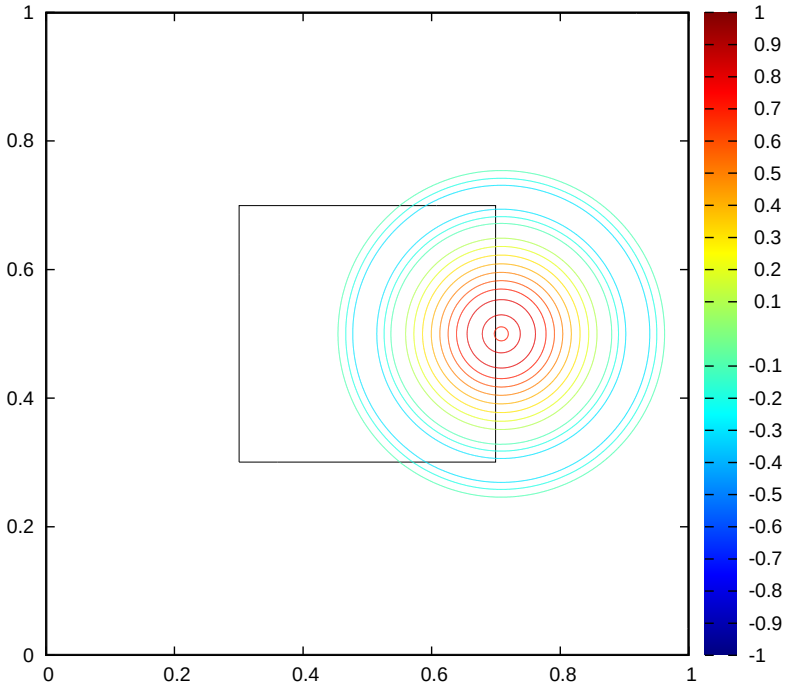
	Object space domain	Δx	$\vec{s}$	θ
1	$[0,1] \times [0,1]$	$1/n$	$(0,0)$	0
2	$[-.2,.2] \times [-.2,.2]$	$0.4/n$	$(.5,.5)$	$-t\pi/3$

Table 6.2: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the two grids used in our vortex flow past grid boundary tests. n indicates the number of cells in each dimension on the coarsest grid. During parallel simulation each grid remained unsubdivided and was allocated a single MPI process since the number of cells on each grid was the identical.

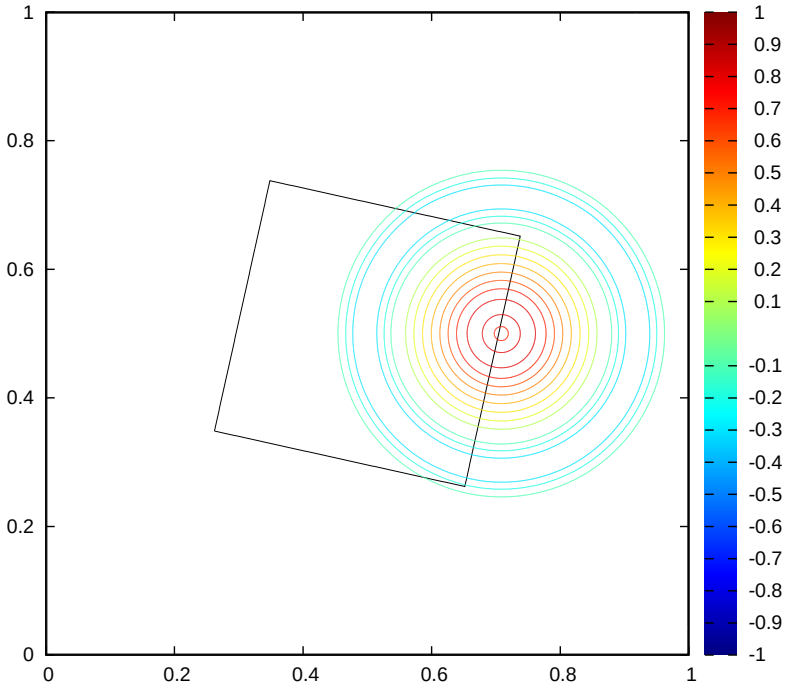
6.2.4 Two-dimensional flow past a stationary circular cylinder

We consider the two dimensional stationary circular cylinder example from [51] in which a two dimensional cylinder is placed within a flow field with approximate far field boundary conditions in order to examine the resulting vortex shedding patterns. We discretize the domain $[0, 38.4] \times [0, 25.6]$ using three grids as listed in Table 6.3 and as illustrated in Figure 6.8, noting that we consider both the cases where the grid containing the cylinder remains stationary at its time $t = 0$ orientation and when it is allowed to rotate while the cylinder remains stationary. Note that our Chimera grid approach allows us to discretize this large domain efficiently by using a coarse grid covering the entire domain in order to approximate far field boundary conditions such that vortices travel a long way before interacting with the domain walls. The boundary conditions are specified as follows: the left of the domain has inflow boundary conditions with a velocity of 1, the right of the domain has outflow boundary conditions, and the top and bottom walls are specified with slip boundary conditions. The cylinder has diameter 1 and its center is located at $\vec{s}_{\text{object}} = (9.6, 12.8)$. We use a characteristic length of 1 and a free stream velocity of 1 when computing the viscosity from the Reynolds number. We calculate the coefficient of drag C_D as two times the net force on the cylinder in the x direction and the coefficient of lift C_L as two times the net force on the cylinder in the y direction.

In Table 6.4 we give the average values and ranges of C_D, the ranges of C_L, and Strouhal numbers produced by simulations with Reynolds numbers 100, 150 and 200

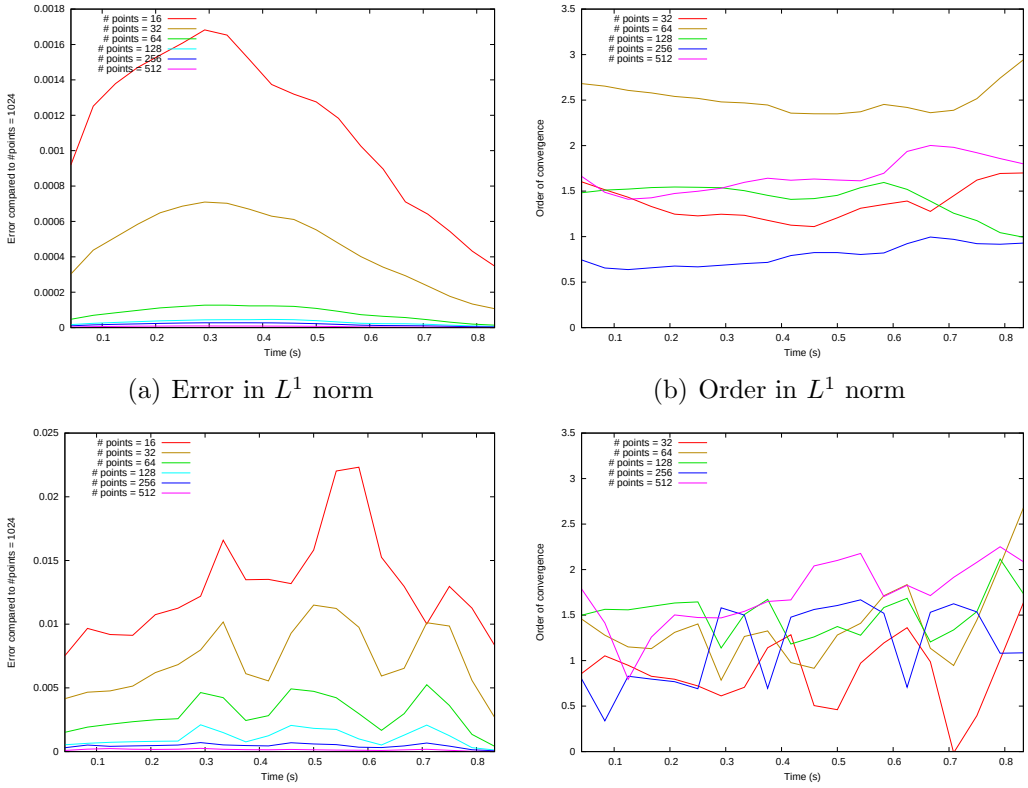


(a) Non-moving grid



(b) Moving grid

Figure 6.5: Vorticity isocontours for the vortex flow across grid boundary example at $t = .20833$. Note that the isocontours match at the grid boundaries and no artifacts are visible along or further away from the boundaries.



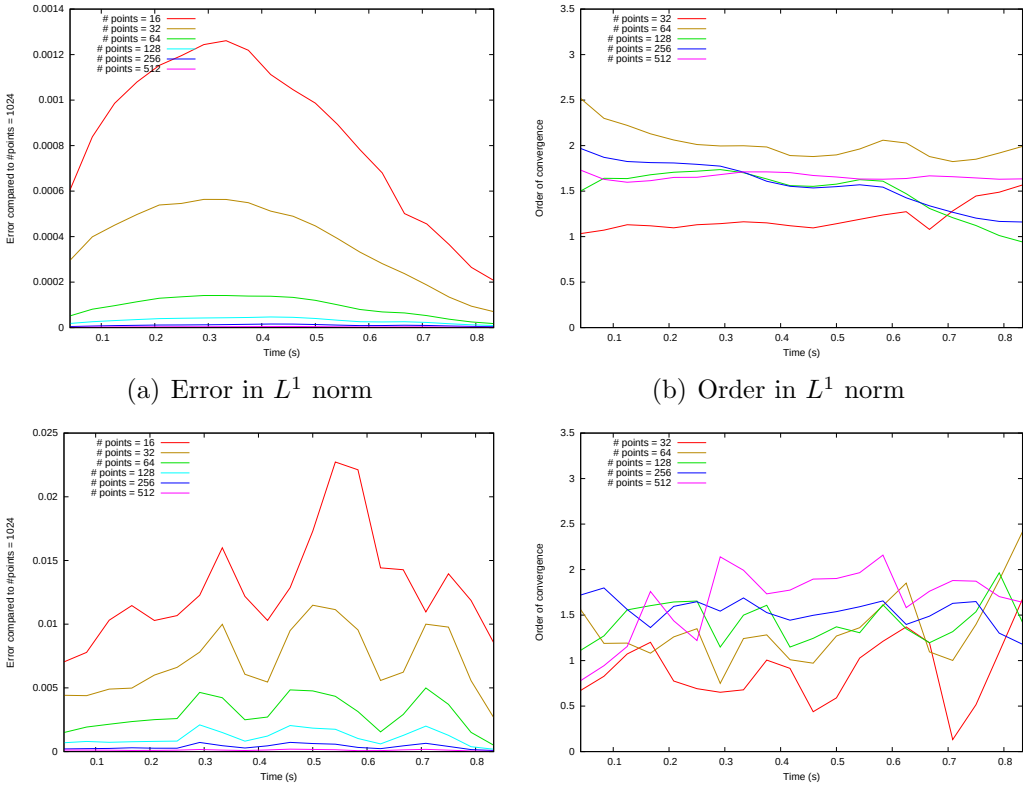
(a) Error in L^1 norm

(b) Order in L^1 norm

(c) Error in L^∞ norm

(d) Order in L^∞ norm

Figure 6.6: The errors and orders of accuracy of velocities for the vortex flow across grid boundary example, with a stationary fine grid. Figures (a) and (c) show that the error tends towards zero in both the L^1 and L^∞ norms implying self convergence, whereas Figures (b) and (d) show that the error seems to be improving towards first order accuracy as the grid is refined.



(a) Error in L^1 norm

(b) Order in L^1 norm

(c) Error in L^∞ norm

(d) Order in L^∞ norm

Figure 6.7: The errors and orders of accuracy of velocities for the vortex flow across grid boundary example, with a rotating fine grid. Figures (a) and (c) show that the error tends towards zero in both the L^1 and L^∞ norms implying self convergence, whereas Figures (b) and (d) show that the error seems to be improving towards first order accuracy as the grid is refined.

using our method with the fine grid enclosing the cylinder held stationary at its time $t = 0$ orientation. We note that the values produced by our method clearly lie within or are very close to the ranges of values produced and cited by [51]. In particularly the Strouhal numbers are all in very close agreement with both the numerical and experimental results given in [51]. For Reynolds number 100, Figure 6.6 gives the average values and ranges of C_D, the range of C_L, and Strouhal numbers as the grids are refined.

Table 6.5 gives the average value and range of C_D, the range of C_L, and Strouhal number produced by a simulation with Reynolds number 100 using our method with the fine grid enclosing the cylinder undergoing a specified rotation. We note that the range for the coefficient of drag is slightly larger, potentially induced by the motion of the grid enclosing the cylinder. Some artifacts of this type are to be expected before the method has exactly converged. However, we found that under refinement the range tended towards the values found in [51] and that produced by the simulation with the stationary grid. The other values are nearly identical to those from the stationary case including the Strouhal number indicating that the motion of the grid did not change the rate at which vortices were shed even though the grid rotated at a different frequency.

In Figure 6.9 we plot the pressures when C_L is at its negative extrema for all of the tests. The plot for the Reynolds number 200 case shows results comparable to the pressure plots from [51]. The pressure plots for the Reynolds number 100 case on the stationary and rotating grids are also nearly identical further confirming that the motion of the grid did not adversely affect the solution. We would also like to emphasize to the reader that since the pressures produced in an incompressible flow solver are equivalent to Lagrange multipliers enforcing the divergence free constraint, that the forces are highly susceptible to oscillations even on high refined grids. See [63, 95, 62] for further discussion.

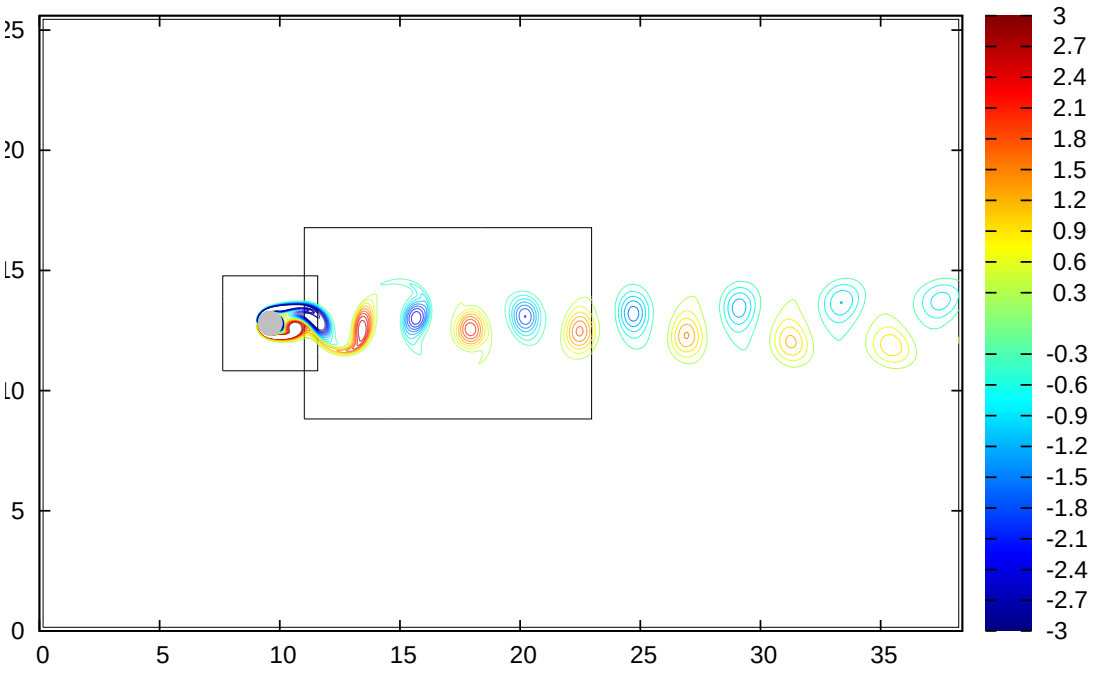
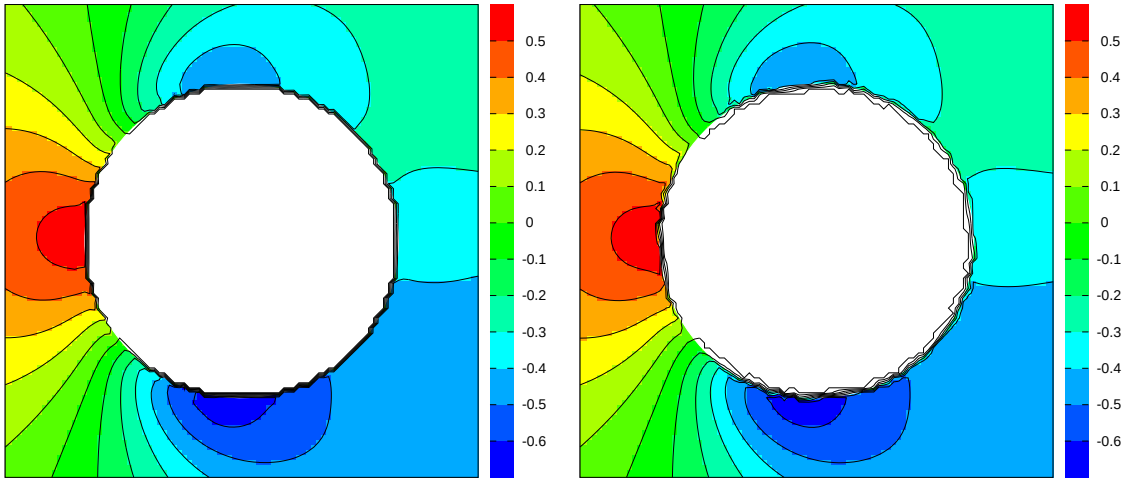


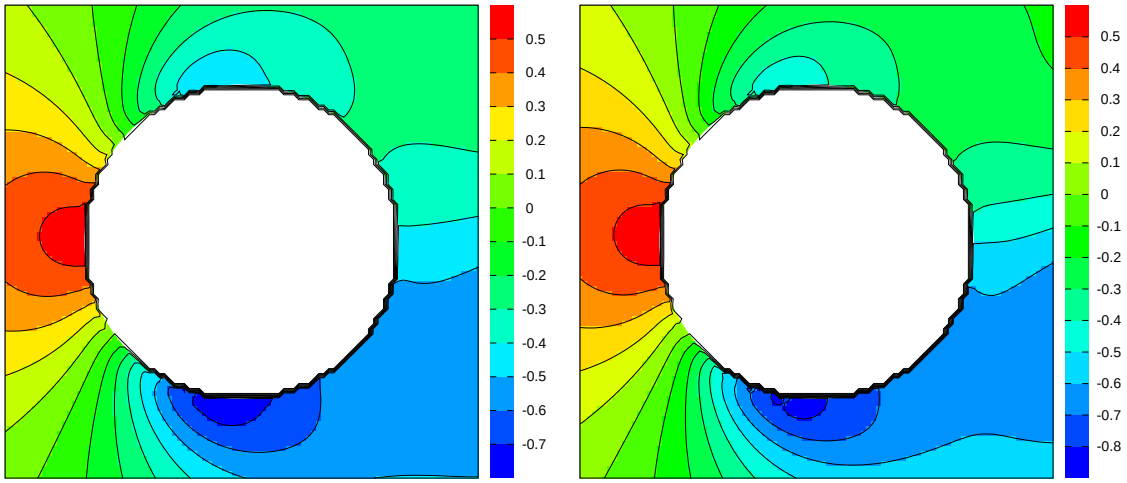
Figure 6.8: Vorticity isocontours for the flow past a circular cylinder example with Reynolds number 200.

	Object space domain	Δx	$\vec{s}$	θ
1	$[0, 38.4] \times [0, 25.6]$	$25.6/n$	$(0, 0)$	0
2	$[-2, 2] \times [-2, 2]$	$4/n$	$(9.6, 12.8)$	$t\pi/4$
3	$[11, 23] \times [8.8, 16.8]$	$8/n$	$(0, 0)$	0

Table 6.3: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the three grids used in our flow past a circular cylinder tests. n indicates the number of cells along the y-axis on the coarsest grid. During parallel simulation the background grid was allocated 24 processors, the fine grid enclosing the cylinder was allocated 16 processors, and the grid capturing the wake was allocated 24 processors.



(a) $Re = 100$ (b) $Re = 100$, moving grid



(c) $Re = 150$ (d) $Re = 200$

Figure 6.9: Pressure contours for the flow past a circular cylinder example taken when the coefficient of lift was at its most negative value. Note that (d) agrees with the Reynolds number 200 pressure plots from [51] and that the Reynolds number 100 pressure plots for the (a) stationary grid and (b) rotating grid are nearly identical.

Reynolds Number	C_D	C_L	St
100	$1.3754 \pm .0094$	$\pm.3347$	.1685
150	$1.3551 \pm .0260$	$\pm.5219$	.1868
200	$1.3638 \pm .0446$	$\pm.6822$	.1982

Table 6.4: The coefficients of drag and lift (C_D & C_L) and Strouhal numbers for varying Reynolds numbers computed using our method with all grids stationary as listed in Table 6.3 and $n = 512$. Note the good agreement of all values with those produced and cited by [51].

Reynolds Number	C_D	C_L	St
100	$1.3755 \pm .0108$	$\pm.3387$	.1684

Table 6.5: The coefficients of drag and lift (C_D & C_L) and Strouhal number for Reynolds number 100 computed using our method where the rotation of the grid enclosing the cylinder is specified as listed in Table 6.3 and $n = 512$. Note that the values are close to those for the stationary case as listed in Table 6.4.

6.2.5 Two-dimensional flow past a rotating elliptic cylinder

In order to examine the case where the structure is moving we consider the case of a two-dimensional rotating elliptic cylinder similar to the stationary elliptic cylinder example from [51]. We discretize the domain $[0, 25.6] \times [0, 24]$ with three grids as listed in Table 6.7 and as shown in Figure 6.10(a), and use the same boundary conditions as in case of the circular cylinder. We place an elliptic cylinder with a long axis length of 1 (also the characteristic length) and aspect ratio of .2 at $\vec{s}_{\text{solid}} = (9.6, 12)$ with its long axis along the x axis in object space. The cylinder rotates with angular velocity $\theta_{\text{solid}} = \pi/4$ which is matched by the enclosing fine grid. For Reynolds number 200, Figure 6.11 gives the errors and orders of accuracy for our method computed by comparing against a baseline simulation run at $n = 1024$. Note that both the L^1 and L^∞ errors tend towards zero implying self convergence. The orders of accuracy tend towards first order for the L^1 error and half order for the L^∞ error. We note that that errors are dominated by the errors at the cylinder's boundary and could be reduced by substituting a more accurate fluid-structure coupling scheme without changing the way the intergrid boundaries are handled. See the example in Section 6.2.3 in order to examine the behavior of the errors when they are not dominated by

n	C_D	C_L	St
64	$1.3865 \pm .0138$	$\pm.2871$	.1580
128	$1.3653 \pm .0083$	$\pm.3083$	.1643
256	$1.3716 \pm .0092$	$\pm.3295$	.1674
512	$1.3754 \pm .0094$	$\pm.3347$	.1685
1024	$1.3770 \pm .0097$	$\pm.3391$	.1691

Table 6.6: The coefficients of drag and lift (C_D & C_L) and Strouhal numbers for Reynolds number 100 computed using our method with all grids stationary as listed in Table 6.3 and varying resolutions. The coefficients of drag and lift and Strouhal numbers clearly tend towards the values produced and cited in [51]. Additionally, the Strouhal numbers clearly demonstrate first order convergence as the grids are refined. While the exact convergence regimes of the coefficients of drag and lift are less clear, the difference between the values at successive resolutions is decreasing.

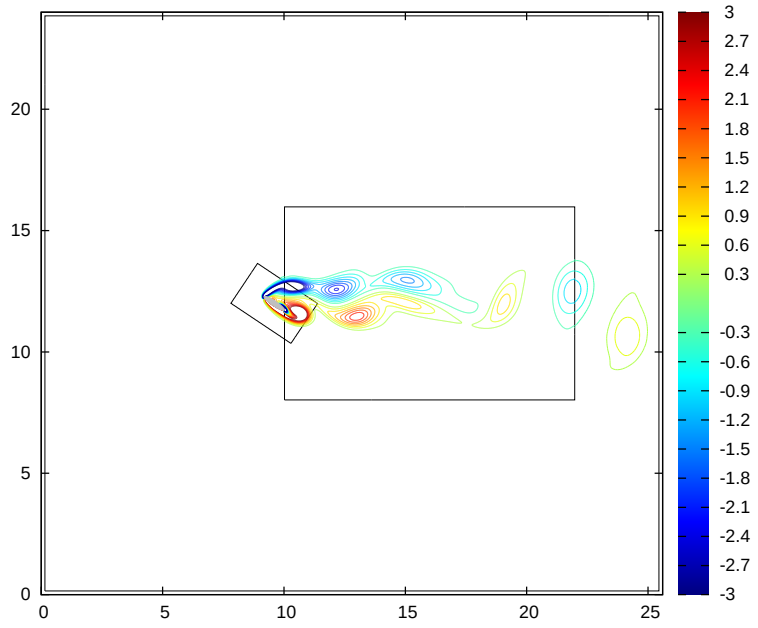
those in the structure boundary layer. The vorticity contours are plotted in Figure 6.10 and show good agreement at grid boundaries.

	Object space domain	Δx	$\vec{s}$	θ
1	$[0, 25.6] \times [0, 24]$	$25.6/n$	$(0, 0)$	0
2	$[-1, 1] \times [-1.5, 1.5]$	$2/n$	$(9.6, 12)$	$t\pi/4$
3	$[10, 22] \times [8, 16]$	$4/n$	$(0, 0)$	0

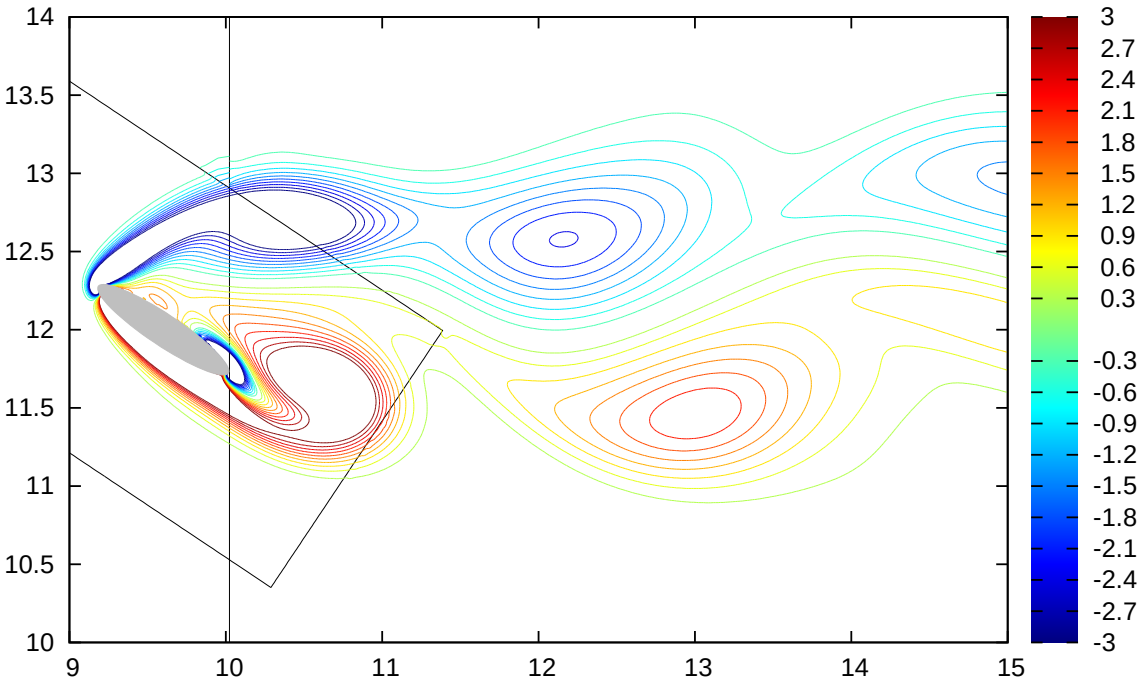
Table 6.7: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the three grids used in our flow past a rotating elliptic cylinder example. n indicates the number of cells in x-axis on the coarsest grid. For all tests in this example $n = 256$. Note that the second grid encloses the elliptic cylinder and rotates with the cylinder at an angular velocity of $\pi/4$. During parallel simulation each of the three grids remains unsubdivided and was allocated a single MPI process since the number of cells on each grid were similar enough to not warrant any subdivision.

6.2.6 Two-dimensional flow past multiple rotating elliptic cylinders

In our final two-dimensional example we consider three rotating elliptic cylinders in order to demonstrate our method on a more complex example. We discretize the domain $[0, 9] \times [0, 6]$ with six grids as listed in Table 6.8 and shown in Figure 6.12(a), where grids 3, 4 and 5 each enclose and move with an elliptic cylinder as listed in Table 6.9. Note that unlike previous examples the grids in this case were subdivided for parallel computation also as described in the caption of Table 6.8. We use the same boundary conditions as used in the case of the stationary circular cylinder. In order to maximize the number of details produced we used zero viscosity. Figures 6.12 and 6.13 show the vorticity at times $t = 5.3333$ and $t = 18.959$. Notice the highly detailed vortices coming off the tips of the elliptic cylinders and that they smoothly transfer onto the coarse grids.

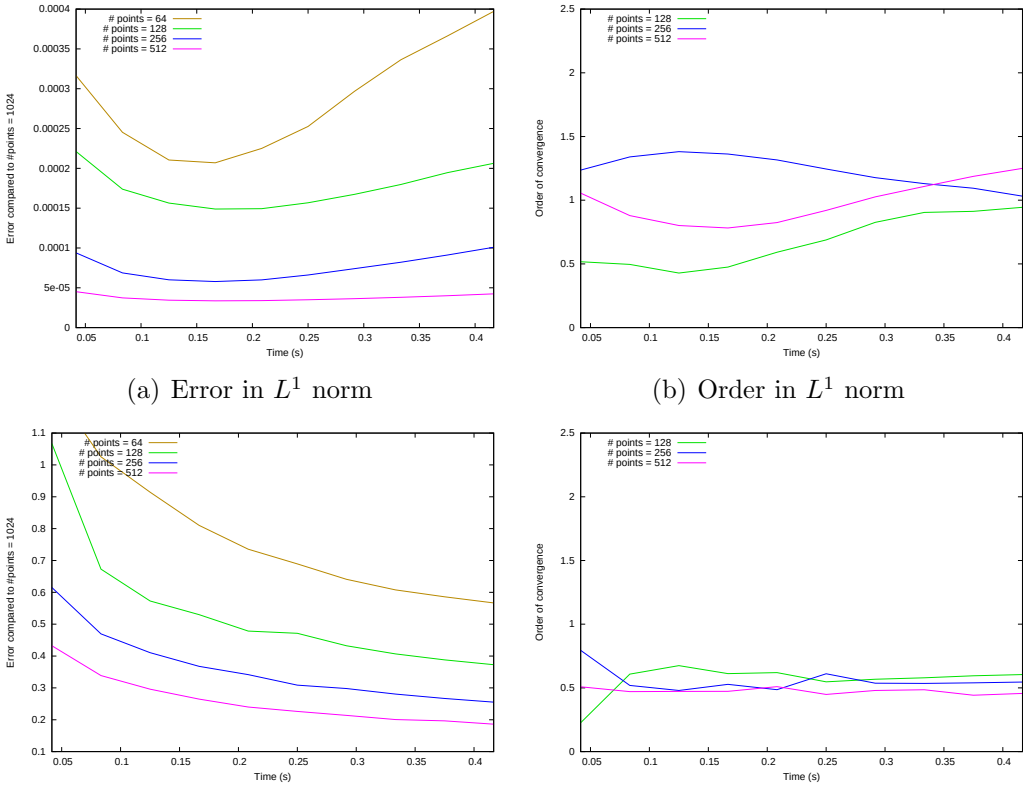


(a) Vorticity, entire domain



(b) Vorticity, close-up

Figure 6.10: Vorticity isocontours for the flow past an elliptic cylinder example. Note that the elliptical cylinder and the grid attached to it are rotating with angular velocity $\pi/4$.



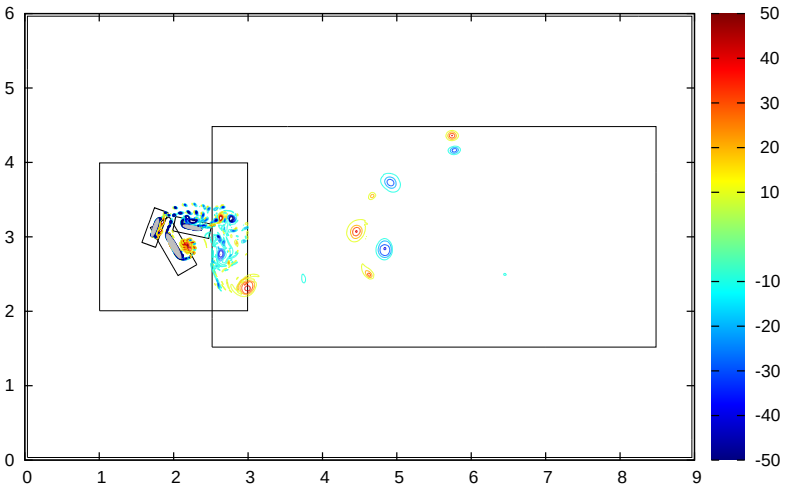
(a) Error in L^1 norm

(b) Order in L^1 norm

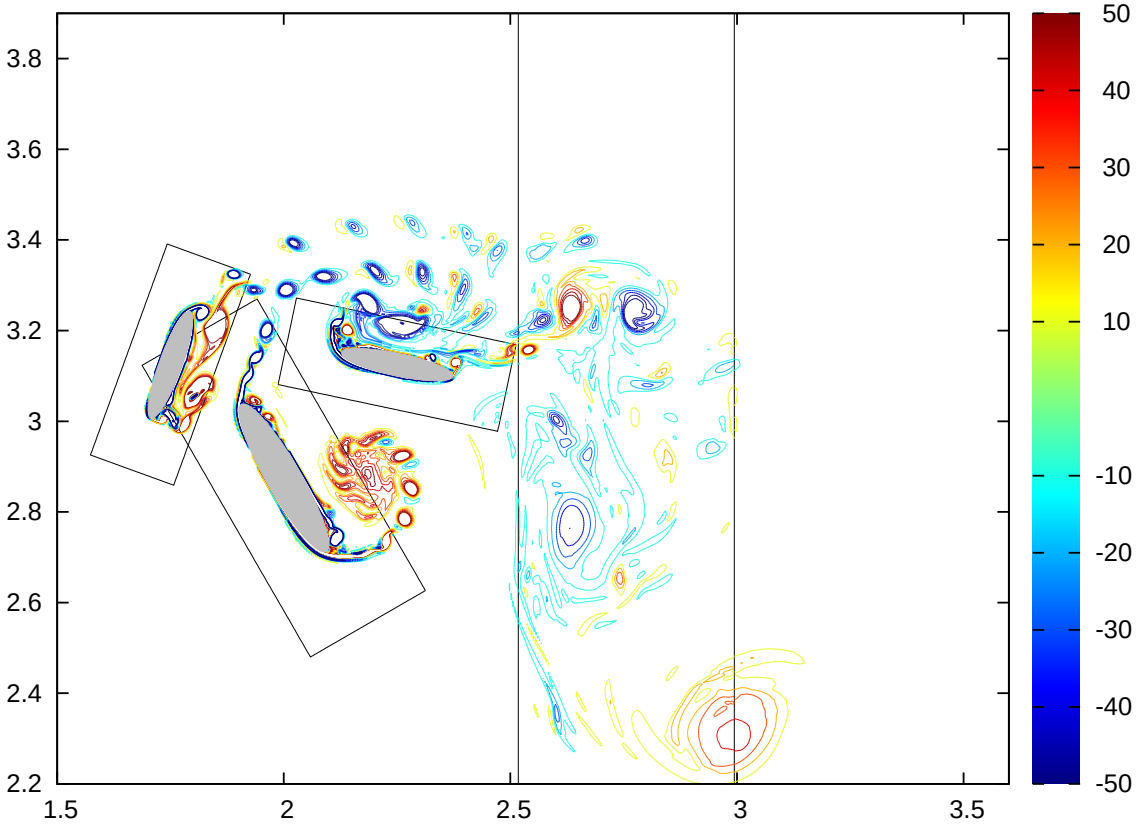
(c) Error in L^∞ norm

(d) Order in L^∞ norm

Figure 6.11: The error and order of accuracy of the velocities in the flow past rotating elliptic cylinder example. Figures (a) and (c) show that the error tends towards zero in both the L^1 and L^∞ norms implying self convergence. Figure (b) shows that the L^1 error tends towards first order accuracy and (d) shows that the L^∞ error tends towards half order accuracy under refinement.

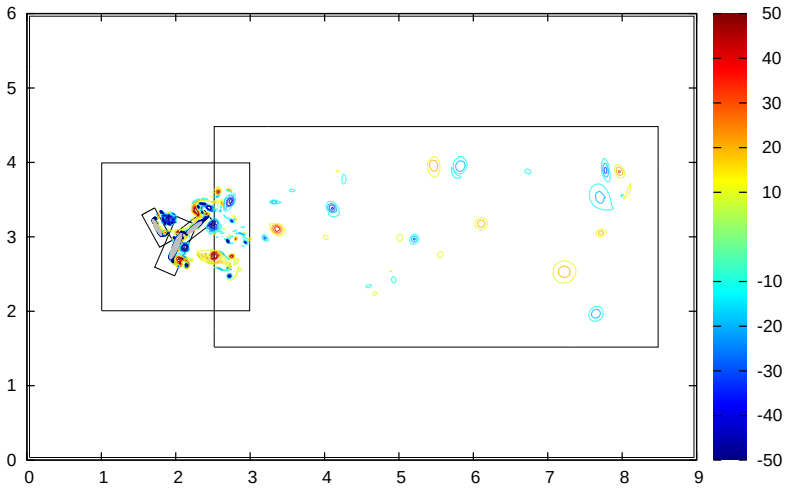


(a) Vorticity, entire domain

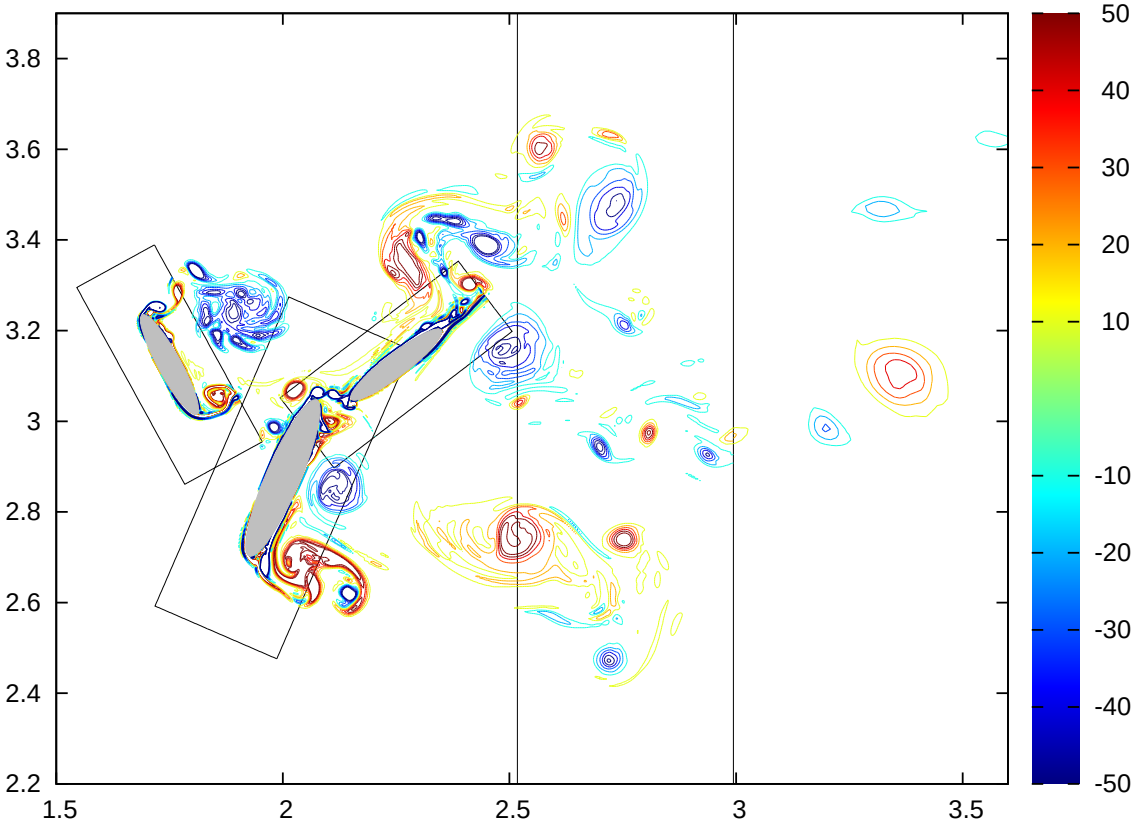


(b) Vorticity, close-up

Figure 6.12: Vorticity isocontours at time $t = 5.3333$ for the flow past multiple elliptic cylinders example.



(a) Vorticity, entire domain



(b) Vorticity, close-up

Figure 6.13: Vorticity at time $t = 18.958$ for the flow past multiple elliptic cylinders example.

	Object space domain	Δx	$\vec{s}$	θ
1	$[0,9] \times [0,6]$	$6/n$	$(0,0)$	0
2	$[-1,1] \times [-1,1]$	$1/n$	$(2,3)$	0
3	$[-0.1,0.1] \times [-0.25,0.25]$	$0.4/n$	$(1.75,3.125)$	$t\pi/6$
4	$[-0.375,0.375] \times [-0.15,0.15]$	$0.6/n$	$(2,2.875)$	$t\pi/8$
5	$[-0.25,0.25] \times [-0.1,0.1]$	$0.4/n$	$(2.25,3.125)$	$-t\pi/5$
6	$[2.5,8.5] \times [1.5,4.5]$	$3/n$	$(0,0)$	0

Table 6.8: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the six grids used in our flow past three rotating elliptic cylinders simulation where $n = 256$. Grids 3, 4 and 5 each enclose and track one of the elliptical cylinders as shown in Figures 6.12 and 6.13. During parallel simulation, a single MPI process was allocated to each grid enclosing a cylinder. Grid 2 was allocated two MPI processes and placed over the three cylinders in order to capture the interactions between them. Grid 6 was added to in order to capture the wake and was allocated two MPI processes. The background grid 1 was only allocated a single MPI process due to its relative low resolution.

	(Long axis length, Short axis length)	$\vec{s}_{\text{solid}}$	θ_{solid}
1	$(0.125, 0.025)$	$(1.75, 3.125)$	$\pi/2 + t\pi/6$
2	$(0.1875, 0.0375)$	$(2, 2.875)$	$t\pi/8$
3	$(0.125, 0.025)$	$(2.25, 3.125)$	$-t\pi/5$

Table 6.9: Axis lengths, positions ($\vec{s}$) and orientations (angle θ) of the three elliptic cylinders used in our flow past three rotating elliptic cylinders example. Note that the long axis of each cylinder lies along the x axis in object space.

6.2.7 Three-dimensional smoke jet past rotating ellipsoid

In order to demonstrate that our method extends trivially to three dimensions we consider a smoke jet impacting and dispersing around a rotating ellipsoid. We discretize the domain $[0,9] \times [0,6] \times [0,6]$ using three grids as listed in Table 6.10 where one grid encloses and moves with the rotating ellipsoid. The axis lengths of the ellipsoid are .25, .042 and .125 which correspond to the x, y and z axes in object space respectively. The location of the ellipsoid is $\vec{s}_{\text{solid}} = (1,3,3)$ and the orientation is specified by a rotation of $\theta_{\text{solid}} = t\sqrt{2}\pi/6$ radians about the axis $\vec{a}_{\text{solid}} = (1/\sqrt{2}, 0, 1/\sqrt{2})$. We specify inflow boundary conditions along the $x = 0$ side of the domain, where the tangential components of the velocity are zero and the normal component is specified as follows:

$$u(0,y,z) = \begin{cases} 1 & \text{if } |(y,z)-(3,3)| \le .09 \\ 1 - (|(y,z)-(3,3)| - .09)/.03)) & \text{if } .09 < |(y,z)-(3,3)| < .12 \\ 0 & \text{otherwise} \end{cases} \quad (6.9)$$

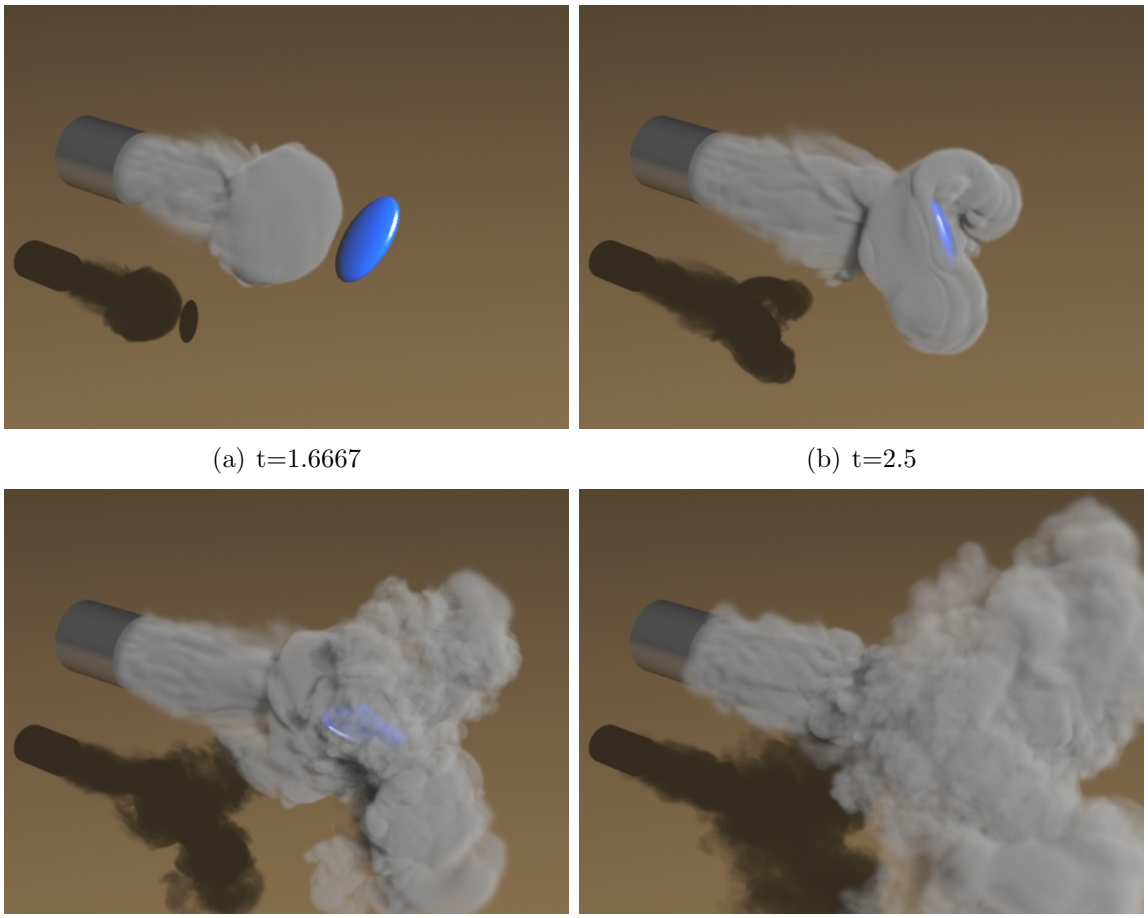
On the $y = 0$, $y = 6$, $z = 0$ and $z = 6$ sides of the domain we specify zero velocity for all components and on the $x = 9$ side of the domain we use outflow boundary conditions. We chose to use the velocity field defined in Equation 6.9 in order to generate a smooth velocity field since the background grid cells were too large to accurately resolve a circular source if the velocity field was discontinuous at the edges of the source.

In order to visualize the flow we passively advect a scalar field. The scalar field is controlled by specifying a single layer of cells along the $x = 0$ side of the domain using the same function as used for the inflow velocity, i.e. $\phi(0,y,z) = u(0,y,z)$. In the remainder of the domain, the value of the passive scalar is initially set to zero. For passive scalar advection, the ghost cells on the computational boundary of the domain are filled using constant extrapolation. In order to advect the scalar field we use SL-MacCormack advection, and note that we revert to first order accuracy when a local extrema is created as described in [94]. In Figure 6.14 we show the results

for various times near the beginning of the simulation. Figure 6.15 shows the results after the smoke has been allowed to propagate further into the domain. Note the sharp details near the object and the smooth transition between grids.

	Object space domain	Δx	$\vec{s}$	$\theta, \vec{a}$
1	$[0,9] \times [0,6] \times [0,6]$	$6/n$	$(0,0,0)$	$0, (0,0,0)$
2	$[-.5,.5] \times [-.5,.5] \times [-.5,.5]$	$1/n$	$(1,3,3)$	$t\sqrt{2}\pi/6, (1/\sqrt{2}, 0, 1/\sqrt{2})$
3	$[.7,6.7] \times [1.5,4.5] \times [1.5,4.5]$	$3/n$	$(0,0,0)$	$0, (0,0,0)$

Table 6.10: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ, axis $\vec{a}$) of the three grids used in our three-dimensional smoke jet past rotating ellipsoid example. In this example we use $n = 256$. During parallel simulation we allocate 8 MPI processes to grid 1, 6 MPI processes to grid 2 and 10 MPI processes to grid 3 in order to load balance between two dual 6-core computers.



(a) t=1.6667 (b) t=2.5

(c) t=3.5417 (d) t=5.4167

Figure 6.14: The passive scalar rendered as smoke for the three dimensional smoke jet past rotating ellipsoid example.

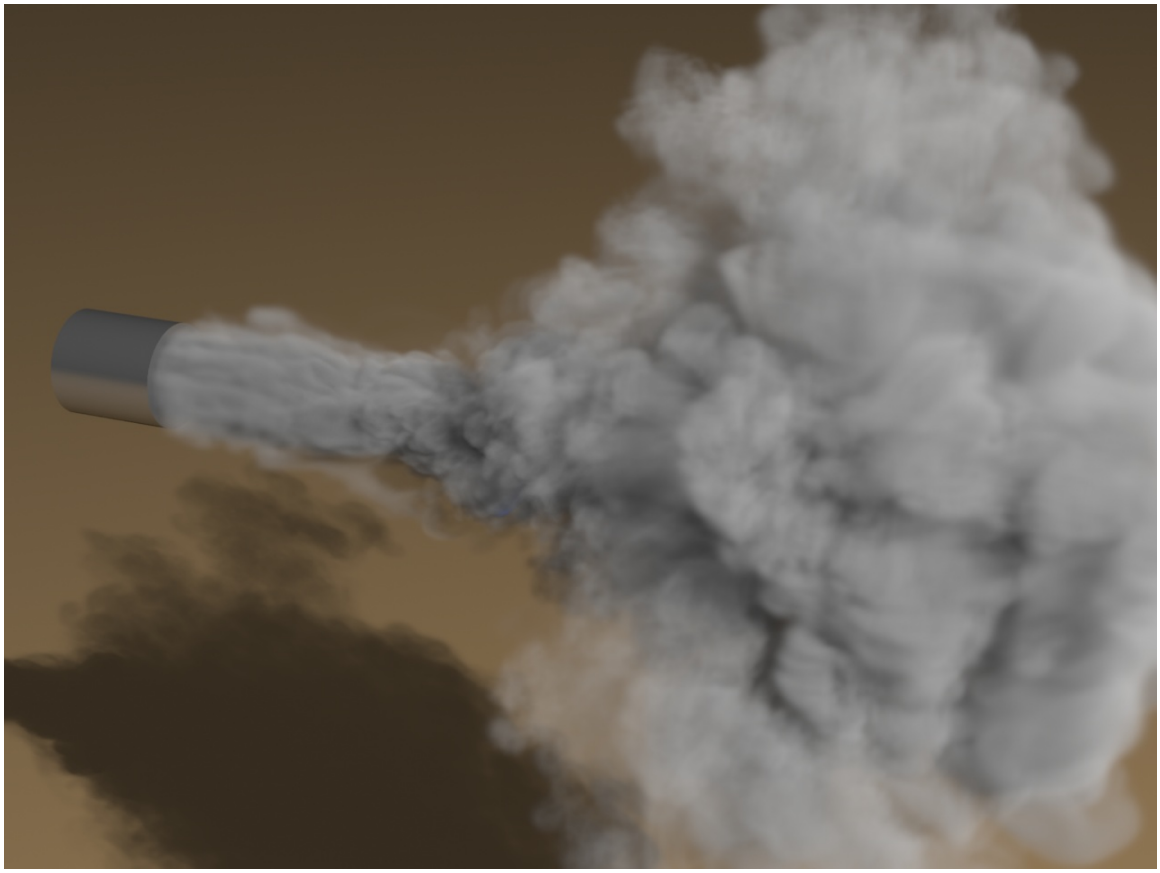


Figure 6.15: The passive scalar rendered as smoke at $t = 13.958$ for the three dimensional smoke jet past rotating ellipsoid example.

Chapter 7

Fluid Structure Interaction

7.1 Ghost Cell Computation and Advection

As shown in Figures 8.7, 8.8, 8.6, 8.5 – 8.9 we often attach grids to solid objects. Since each grid simply moves with its object preserving the relative placement of the fluid degrees of freedom with respect to the object interface, the object only needs to be rasterized once at the beginning of a simulation. This allows for a greater level of user control in crafting the initial rasterization (see the note about the red plane in Figure 7.1), especially as far as rasterized normals, watertightness, etc. are concerned. In order to compute the time step, we assume that objects move with constant linear and angular velocity in order to approximate the time t^{n+1} locations of the grids bound to these objects. However, the actual displacement during a time step will be affected by acceleration under gravity and buoyancy, interaction with other objects through contact, collision, and articulation, etc. Thus, the actual motion of the grid may exceed that predicted by the time step computation at the beginning of the time step. We handle this by dynamically filling additional ghost cell layers before advection and found the added cost to be small. The remainder of the advection step proceeds as in the kinemtic case described in Section 6.1

7.2 Coupled Solver

In the case of kinematic objects we apply a Neumann boundary condition as described in Section 6.1 in the pressure solve. In the two-way coupled case, we follow the approach of [90] by using a constraint equation and Lagrange multiplier force for each occluded face of the Voronoi mesh in order to enforce equality between fluid and solid velocities. The constraint is given as follows:

$$\mathbf{W}\mathbf{u}^{n+1} - \mathbf{J}\mathbf{v}^{n+1} = 0 \tag{7.1}$$

where $\mathbf{W}$ and $\mathbf{J}$ map from the fluid and solid velocities, respectively, to the appropriate components of the velocity field at constrained faces. The updated fluid and solid momentum update equations are given as follows:

$$\mathbf{u}^{n+1} = \mathbf{u}^* + \beta^{-1}\bar{\mathbf{D}}^T\hat{\mathbf{p}} + \beta^{-1}\mathbf{W}^T\lambda \tag{7.2}$$

$$\mathbf{v}^{n+1} = \mathbf{v}^* - \mathbf{M}^{-1}\mathbf{J}^T\lambda \tag{7.3}$$

where v^* are the explicit force integrated solid velocities, u^* are the post advection and explicit force integrated fluid velocities, and v^{n+1} and u^{n+1} are the time t^{n+1} solid and fluid velocities, respectively. $\hat{\mathbf{p}}$ are the pressures scaled by Δt, $\bar{\mathbf{D}} = \mathbf{V}_c\mathbf{D}$ is the volume weighted divergence, $\mathbf{M}$ is the solid mass matrix, and λ are the coupling force Lagrange multipliers. Note that $\frac{1}{\rho}\mathbf{G}$ has been equivalently replaced by $-\beta^{-1}\bar{\mathbf{D}}^T$ where β are the Voronoi mesh dual cell fluid masses. The dual cell fluid mass is defined for each face as the face's dual cell fluid volume times the fluid density. Further note that Equation 7.2 is the update equation for the faces on the Voronoi mesh and that non-coincident Cartesian faces are updated as described in Section 6.1. In order to solve for $\hat{\mathbf{p}}$ and λ, Equations 7.2 and 7.3 are substituted into the divergence constraint $\bar{\mathbf{D}}(\mathbf{u}^{n+1} + \mathbf{u}_n^{n+1}) = 0$ and Equation 7.1, giving the coupled system as follows:

$$\begin{pmatrix} \bar{\mathbf{D}}\beta^{-1}\bar{\mathbf{D}}^T & \bar{\mathbf{D}}^{-1}\beta^{-1}\mathbf{W}^T \\ \mathbf{W}\beta^{-1}\bar{\mathbf{D}}^T & \mathbf{W}\beta^{-1}\mathbf{W}^T + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \lambda \end{pmatrix} = \begin{pmatrix} -\bar{\mathbf{D}}(\mathbf{u}^* + \mathbf{u}_n^{n+1} + \beta^{-1}\bar{\mathbf{D}}_d^T\hat{\mathbf{p}}_d) \\ \mathbf{J}\mathbf{v}^* - \mathbf{W}(\mathbf{u}^* + \mathbf{u}_n^{n+1}) \end{pmatrix} \tag{7.4}$$

where $\bar{\mathbf{D}}_d$ includes the terms of the volume weighted divergence corresponding to faces between cells with solved pressure variables and cells with Dirichlet boundary conditions.

7.2.1 Preconditioning

We applied a block diagonal preconditioner in which we partition the variables in two sets corresponding to the pressures, and solid-fluid coupling/articulation lagrange multipliers. The diagonal block corresponding to the pressures $\bar{\mathbf{D}}\beta^{-1}\bar{\mathbf{D}}^T$ was preconditioned using an incomplete Cholesky (0-fill) factorization and the block corresponding to the lagrange multipliers, $\mathbf{W}\beta^{-1}\mathbf{W}^T + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ was diagonally preconditioned. When computing in parallel, the incomplete Cholesky preconditioner was applied independently on each computational node to the diagonal block corresponding to the local pressures only. While this could reduce the effectiveness of the preconditioner, for the problems we ran, which used up to 192 processors, we did not experience any significant drop in the effectiveness of the preconditioner. However, during tests we observed that without the diagonal preconditioning of the lagrange multiplier block, the residual could remain large for the equations corresponding to the coupling constraints, even though the incompressibility of the fluid was adequately enforced. In fact, the method would sometimes not even converge, resulting in spurious behavior including objects incorrectly sinking. However, This problem is not a result of our pressure discretization in particular and occurs with many monolithic solid-fluid coupling methods as the resolution is increased. On the other hand, we experienced no convergence issues or spurious behavior after applying the diagonal preconditioner.

7.3 Coupled Integration Scheme

We apply a slightly modified version of the time integration scheme from [91] where we replace the backward euler integrator in the position integration step with a forward

euler integrator in order to eliminate the additional coupled solve. The resulting scheme proceeds as follows:

1. Explicitly integrate the solid positions using x^n and v^n to produce x^{n+1}.
2. Advect the fluid velocities from u^n to $\hat{u}$.
3. Integrate the solid velocities with collision forces, gravity, and contact forces from $\hat{v}$ to v^*.
4. Integrate the fluid velocities with gravity from $\hat{u}$ to u^*.
5. Solve the coupled system in Equation 7.4 for $\hat{\mathbf{p}}$ and λ.
6. Update the solid velocities using Equation 7.3 to compute v^{n+1}.
7. Update the Voronoi mesh fluid velocities using Equation 7.2 and then update the remaining Cartesian grid velocities as described in Section 6.1 to produce u^{n+1}.

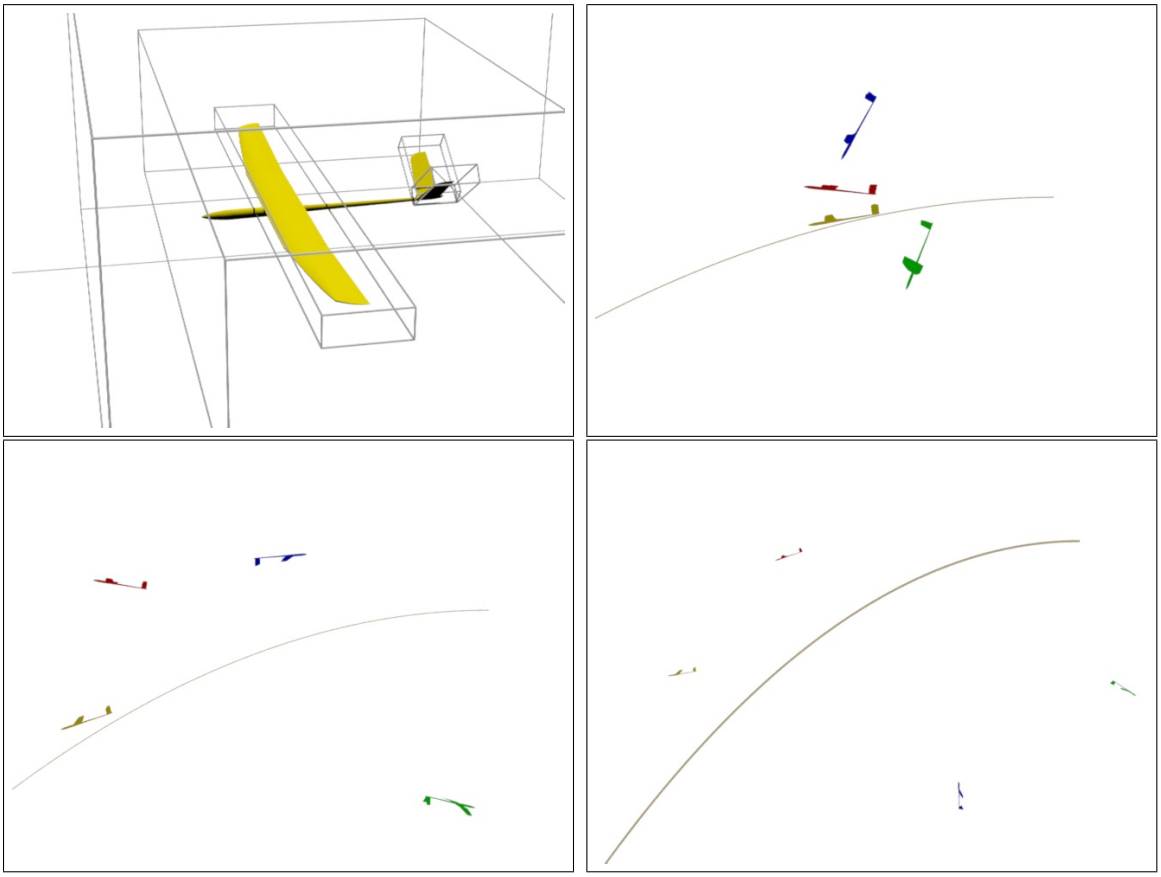


Figure 7.1: Resolving a three dimensional glider (yellow) with a number of Chimera grids gives it the ability to fly in a large domain. The solid line shows the parabolic path a ballistic rigid body would take if it did not fly. The blue glider is simulated with only the background grid and falls almost immediately. Meanwhile, the red and green gliders each use only two of the six grids attached to the yellow glider. Although the green glider crashes, for illustration purposes, we can carefully adjusted the position and orientation of the red glider's two grids allowing it to fly. Note that this hand-picked rasterization will remain constant throughout the entire simulation since the grids move with the glider enabling the user to handcraft efficient rasterizations that improve the physical realism. We emphasize to the reader that we did not specially tune the positions of the grids in the case of the yellow glider as it was well resolved by its finer grids. Slight perturbations to the the placement of its grids did not alter the glider's behavior.

Chapter 8

Free Surface Flow

8.1 Level Set Method

In order to represent and evolve the free surface on overlapping grids we extend the particle level set approach of [29]. On each grid we store the level set ϕ values at cell centers as in the single grid case. At the beginning of each time step we fill the the ϕ ghost cells using the scheme from Section 2.2, and then apply the first order accurate ALE semi-Lagrangian advection scheme from Section 3.2.1. After advection, we again fill the ghost cells before reinitializing the interface independently on each grid using the fast marching method and higher order interface initialization as described in [66]. Note that we fill enough ghost cell layers to cover the bandwidth used by the fast marching method. This allows us to march over both the interior and ghost cells in a single process and achieve the correct result on the interior of the domain. When using the fast marching method, which marches only in a limited band near the interface, the stopping distance on the fine grid can be too small to guarantee a valid interpolation stencil for filling overlapped cells as illustrated in Figure 8.3 (Left). However, this only occurs when the grading between grids is large and can be avoided by increasing the stopping distance on the fine grid so that at least cells next to the interface have correct values. In order to prevent the level set representations from

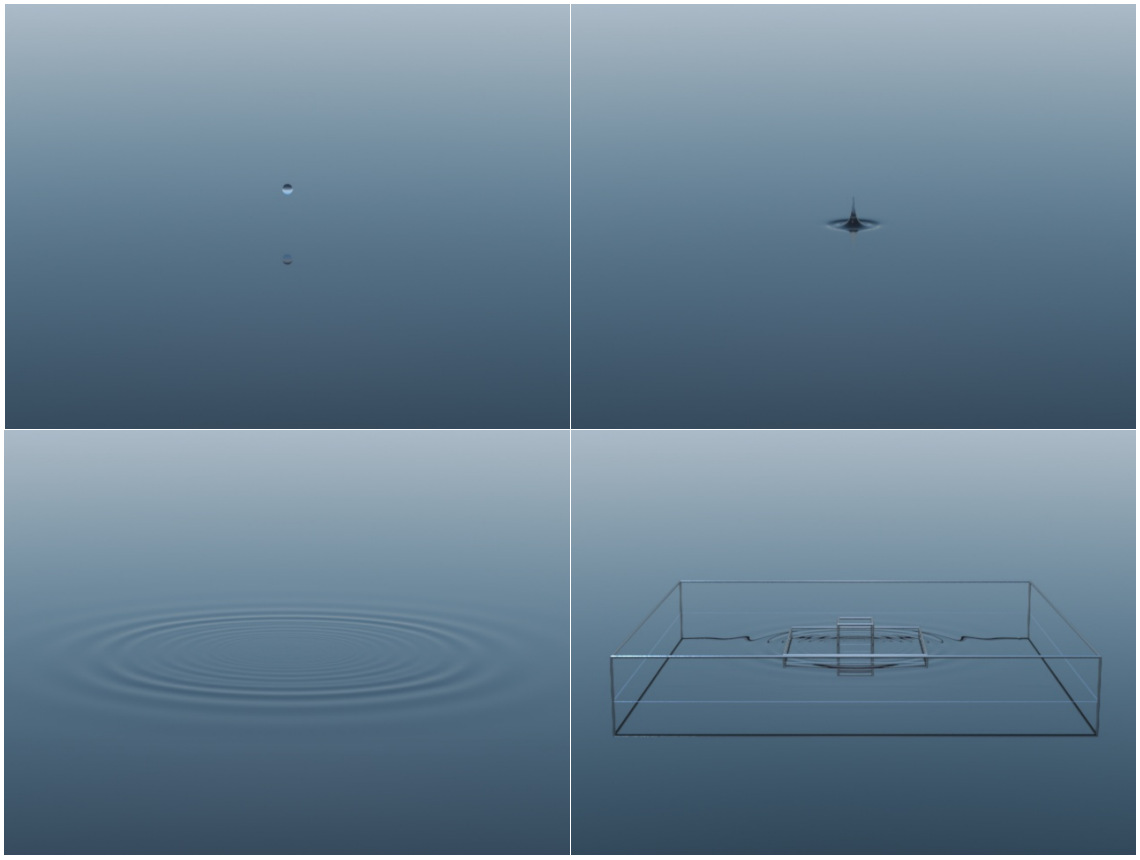


Figure 8.1: A single water drop impacts a flat water surface generating several circular waves. The large domain allows these waves to propagate a long distance without reaching the domain boundaries. Note that the waves pass from one grid to the next without visible artifacts. The bottom right figure is the same as the bottom left except that we zoom the camera back even further in order to show the computational domain and the three grids used in the simulation.

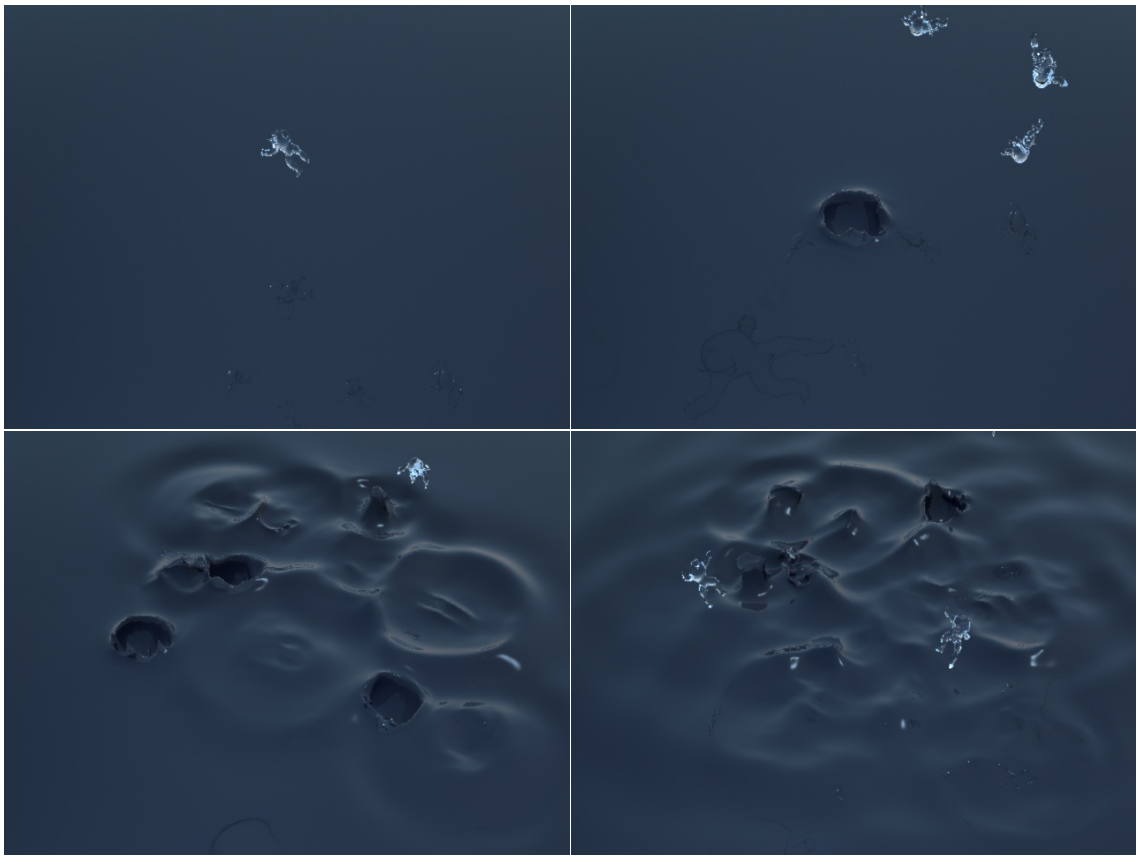


Figure 8.2: Many water droplets in the shape of armadillos are dropped into still water. Each armadillo is enclosed by two moving grids: one tightly wrapped to resolve the armadillo as it falls and one slightly larger to catch the resulting splash. Our Chimera grid strategy allows detailed resolution of the small armadillo droplets even in this large domain.

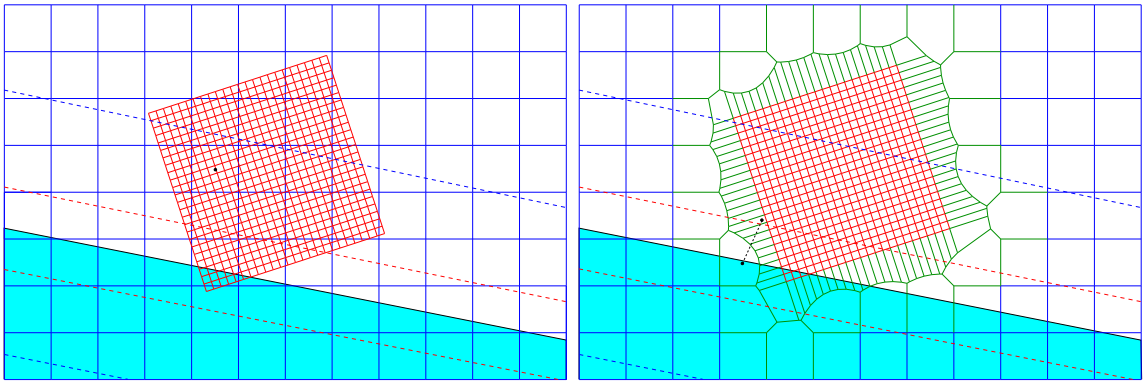


Figure 8.3: A free surface is shown on two grids as a black line with the filled cyan area indicating the fluid phase. When reintializing the level set after advection and extrapolating the velocities across the interface after the pressure solve and velocity update, values are computed for only a limited band of cells around the interface. The limits of these bands are drawn as dashed lines (with the color indicating the corresponding grid). Note that the band is proportionally smaller on the fine grid. (Left) When interpolating values for ghost and overlapped cells invalid values can be produced. Since the cell on the coarse grid indicated by the black dot if beyond the band around the interface on the fine grid an incorrect value is computed. An additional reinitilization step is used to compute new valid values. (Right) In order to compute a second order accurate pressure discretization on the free surface, the ϕ values for cells on either side of the interface are used to compute the dual cell volume fraction. Due to the unstructured connectivity between grids on the Vonoroi mesh, cells on coarse grids along the interface may be adjacent to cells on the fine grid outside of the reinitialization bandwidth as illustrated by the two cells with black dots at their cell centers. We instead compute an updated ϕ value for the cell on the fine grid by linearly extrapolating from the cell on the coarse grid.

drifting apart in overlapping regions on different grids, we inject the ϕ values in any overlapped cells as described in Section 2.2. After injecting to these cells we need to perform a second reinitialization step in order to guarantee valid values through to the stopping distance. This time, after filling the ϕ values in ghost cells, we reinitialize the level set values in the overlapped cells only in order to minimize dissipation.

In order to apply a second order accurate free surface boundary condition in the solver, a valid level set value is required at each cell along the free surface. Due to the limited bandwidth used in the fast marching method and the unstructured

connectivity between grids on the Voronoi mesh used in the pressure solve, certain coarse grid cells may neighbor fine grid cells which do not have valid ϕ values as illustrated in Figure 8.3 (Right). For each of these cells, we extrapolate a ϕ value from a neighboring cell on the Voronoi mesh by computing $\nabla\phi$ at this neighboring cell using values from this cell's intrinsic Cartesian grid. The neighboring cell must both have a valid value of ϕ and a valid value of $\nabla\phi$, and if more than one neighbor has valid information we use the neighbor with the smallest value of $|\phi|/\Delta x$ representing the value closest to the interface with the most accurate information. After computing these updated ϕ values, we interpolate values for any Cartesian cell centers that were removed during the Voronoi mesh construction as well as a one cell thick layer of ghost cells. These cells are filled using barycentric interpolation on the Delaunay triangle mesh dual of the Voronoi diagram. Without this interpolation step to enforce consistency between the Voronoi mesh and the Cartesian grid level set representations, water could become stuck along grid boundaries on the coarse grid since the fluid in these cells does not directly influence the pressure solution. We perform a final reinitialization step in these interpolated cells in order to guarantee that they are valid through to the stopping distance.

8.1.1 Particle Level Set Method

We store a number of layers of particles along the free surface as is typical in the single grid case. After filling the ghost cells we seed particles in each ghost and overlapped cell using the updated ϕ values. While one could instead exchange particles in this step, seeding new particles in ghost cells each time step prevents large particles originating from a coarse grid from adversely effecting the level set function on a fine grid. Following seeding, the majority of the particle advection and level set correction steps remain the same as those in the single grid case. We advect the particles using a second order accurate Runge-Kutta scheme. Subsequently, the particles are used to correct the ϕ values both after advection and after the first reinitialization step only. Recall that the first reinitialization modifies the entire grid while the second

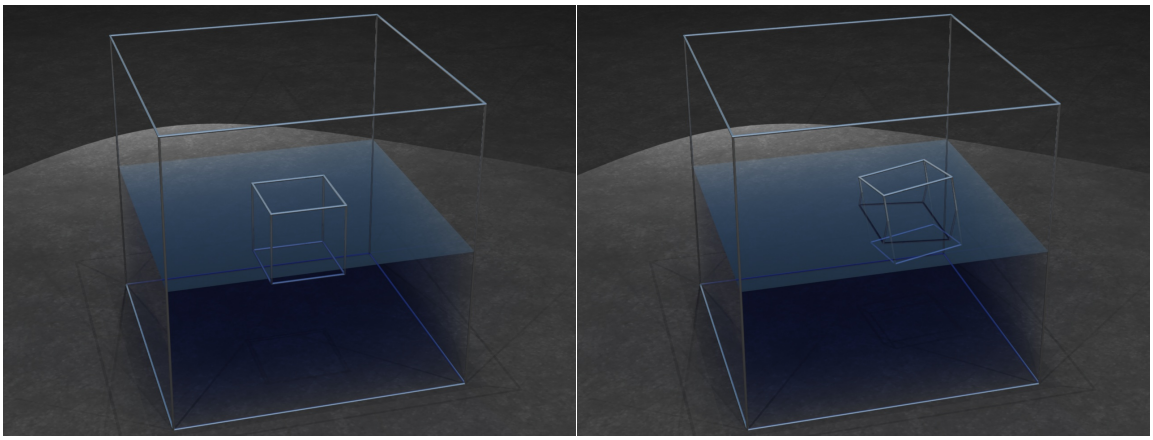


Figure 8.4: A kinematically driven grid moves in and out of still water without disturbing the surface. Throughout the entire simulation the water velocity properly remains within rounding tolerance of zero. The left figure shows the initial state and the right figure shows the water surface unchanged after 600 frames.

reinitialization only modifies overlapped regions.

At the end of each time step we flag any particles that have crossed the interface by more than their radius (or a small threshold) and designate these as "removed particles". This operation is performed in both the ghost and interior regions of each grid. We discard any removed particles that are within the interpolation stencil of an overlapped interior cell. These particles in overlapped regions represent underresolved features on the coarse grid which are simulated more accurately on the fine grid. We keep the removed particles generated in the ghost regions of each grid as they allow us to track the mass loss which can occur at grid boundaries. This approach allows a thin feature on a fine grid to move onto a coarse grid and be tracked as ballistic or passively advected particles for subsequent rendering, instead of simply having the feature disappear at a grid boundary. Finally, before ending the time step, we delete all the remaining non-removed particles in the ghost regions of each grid.

8.1.2 Pressure solver

Water surfaces are particularly sensitive to the approximations made during discretization, and we found that it was essential to solve the hydrostatic case exactly as illustrated in Figure 8.4 in order to prevent spurious vortices from forming at grid boundaries. This was in fact the motivation for our Voronoi diagram pressure solver.

For our single-phase solver in Section 6.1, the velocity for each Voronoi mesh face not coincident with a Cartesian grid face was interpolated from the finest available Cartesian grid. In certain cases, we found that these interpolated velocities caused noticeable artifacts at grid boundaries in free surface simulation. This was particularly evident when the velocity for a Voronoi face between to two fine grid cells was interpolated from a much coarser grid resulting in large error in the velocity. In order to avoid this problem, we first compute the cell center velocity at each adjacent cell centers by averaging on their respective Cartesian grids before taking an average of these cell center velocities to compute the Voronoi mesh face velocity. We further modify this in the case of Voronoi faces along the free surface as follows. Similar to the issues described for ϕ values in Section 8.1 along boundaries between coarse and fine grids, certain cells may not have valid velocities since they are beyond the free surface velocity extrapolation bandwidth. Hence, for these faces we take the velocity from the incident water cell rather than averaging from both cells. We found that these modifications produce a much more accurate velocity and do not cause any noticeable artifacts along grid boundaries.

In order to handle free surfaces in the pressure solver, we set a constant Dirichlet pressure boundary condition in air cells and use a second order accurate cut-cell approach [31, 4] at each face between an air cell and a water cell. This is straightforward to apply recalling that we ensured a valid ϕ value in each cell through extrapolation at the end of Section 8.1.

After computing the pressure, we update both the Voronoi and Cartesian grid face velocities by modifying the approach described in Section 6.1 to compute correct values for Cartesian faces near the interface that are not coincident with a Voronoi

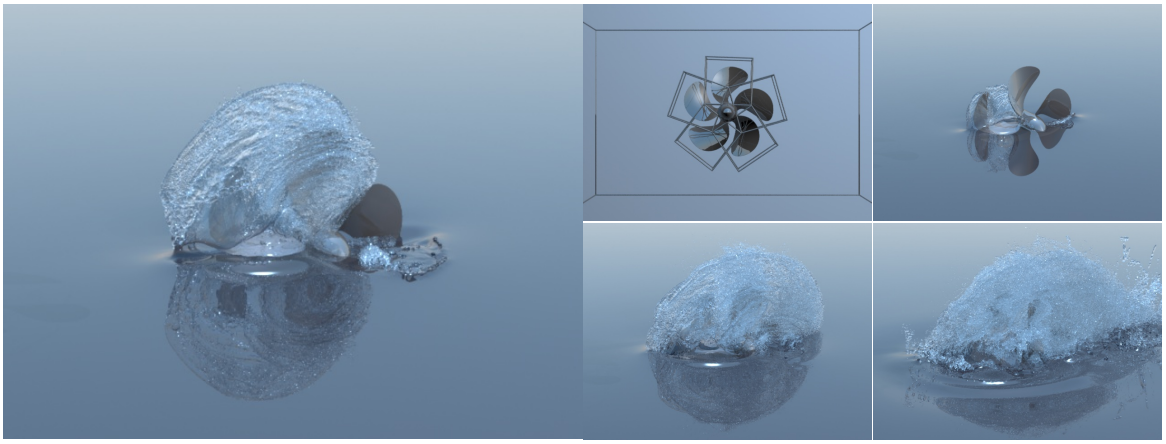


Figure 8.5: A kinematically driven propeller spins with a number of attached Chimera grids. The grids move and rotate with the rigid body frame of the propeller.

face. After computing the cell center velocities, we extrapolate these velocities across the level set water interface on the Voronoi mesh in order to guarantee a valid interpolation stencil for removed water cells. This extrapolation is accomplished using two Jacobi style iterations across the entire mesh where the velocity is computed by averaging the velocities from all adjacent cells with valid velocities. When computing the final Cartesian grid face velocities we average from the neighboring Cartesian grid cell centers using only a one sided average in the case that only one of its neighbors is a water cell (with the other being air).

After updating each water face, we fill the ghost cells and extrapolate the Cartesian grid velocities into the air using the fast velocity extrapolation of [30] independently on each grid. Note that we may not be able to interpolate a valid velocity for every ghost face depending upon the discretization of the overlapping grid. We handle this case by updating the invalid ghost cell faces in the extrapolation procedure as if they were air faces.

8.2 Rendering

In order to render the free surface without visible cracks at grid boundaries we modify the raycasting approach used in [30] to intersect a composite signed distance function. For each sample location the composite signed distance function is computed by considering each overlapping grid in order from coarsest to finest and blending the ϕ value from the current grid with the ϕ value computed from blending the coarser grids. The blending fraction is found by scaling and clamping the the signed distance function to the grid boundary so that the fraction is 0 at the edge of the grid's interpolation domain and 1 several cells within the grid. While the resulting composite ϕ function is no longer a strict signed distance function, the implicit surface defined at $\phi(\vec{x}) = 0$ is smooth and by blending and normalizing the normals in a second step using the same procedure, rendering artifacts are avoided. Note that the same blending procedure is used to define a continuous density function when rendering the smoke in Figures 6.14 and 6.15.

In Figure 8.5 we additionally render the removed negative particles as transparent spray by directly ray tracing the particles as metaballs which are blended with the level set. In Figures 8.8 and 8.9 we render the removed negative particles as a density field defined as the sum of kernel functions centered at each particle. Additionally, a Phillips spectrum was used to bump map the water to give it added detail as described in [104].

8.3 Results

We provide relative timing data for the propeller and island examples in Table 8.1. Note that while the overhead of the added second grid is significant, it does not dominate the simulation time and improves under refinement. Further optimization of the implementation could easily reduce this overhead. In the example shown in Figures 8.8 and 8.9, 17 grids were used with a total of 6 million cells. By extending

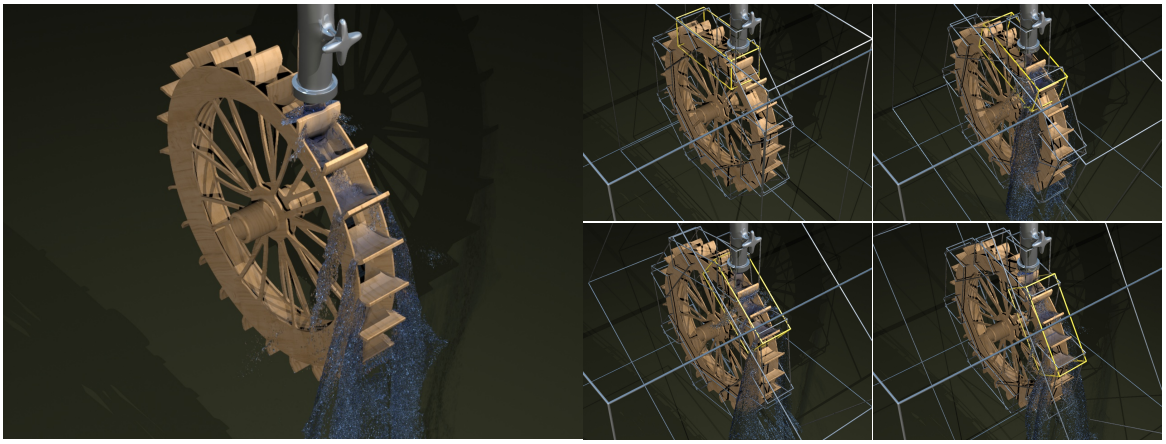


Figure 8.6: Pouring water drives a water wheel via two-way solid-fluid coupling. The domain around the water wheel buckets is resolved by eight grids that rotate with the wheel in order to resolve the thin bucket walls. In addition, a single larger grid encases the entire wheel and also rotates in the same rigid body frame as the wheel.

	Entire time step	Filling ghost and overlapped cells	Reseed particles	Advection	Meshing	Projection	Extrapolation	Delete particles
island, 1 proc	914	23	41	157	318	310	12.3	51
island, 39 procs	85	3.3	5.2	17	24	29	2.2	4.2
propeller, 1 proc	299	7.2	8.5	50	114	109	5.2	2.7
propeller, 23 procs	47	1.8	.87	8.7	17	17	1.5	.19

Table 8.1: Timing data (in seconds) for the island and propeller examples. The table includes timing only for the major steps of the algorithm.

the finest cell's Δx to the background grid, the effective resolution of the simulation was $36500 \times 3650 \times 36500$. While the effective resolution could be made arbitrarily large by decreasing the size of the smallest cell, it does give a sense of the disparate scales being resolved.

In our tests we found that we were computationally bound and that communication costs were relatively minimal compared the other parts of the code. However, as we scale up to more processors the preconditioned conjugate gradient method used to solve the two-way solid-fluid coupled Poisson equation system will become the bottleneck especially due to the frequent communication costs.

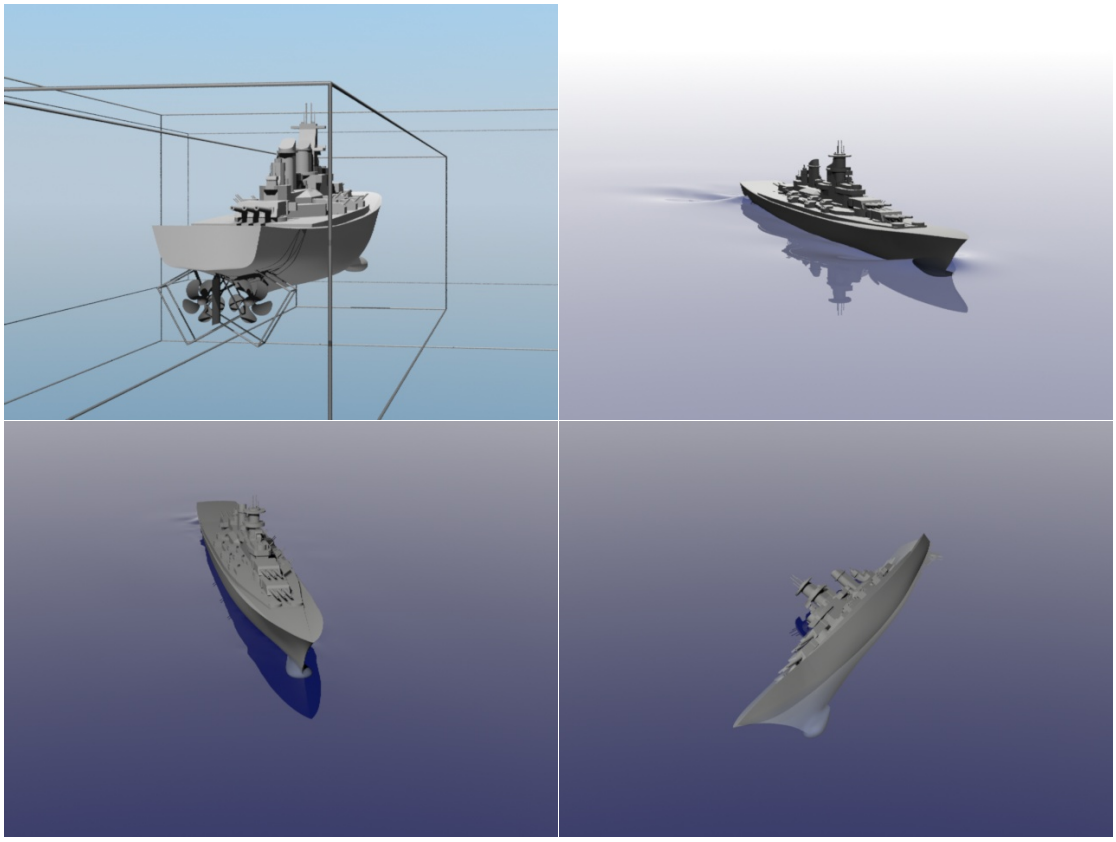


Figure 8.7: The top two figures show a two-way coupled rigid body ship that moves under its own power due to the rotation of two articulated propellers. The lower left figure illustrates that removing the grids surrounding the propellers limits the realism of the simulation resulting in a ship that churns water but cannot propel itself forward. The lower right figure shows that removing the grids around the ship results in a loss of buoyancy and the ship falls through the large grid sinking to the bottom of the ocean.

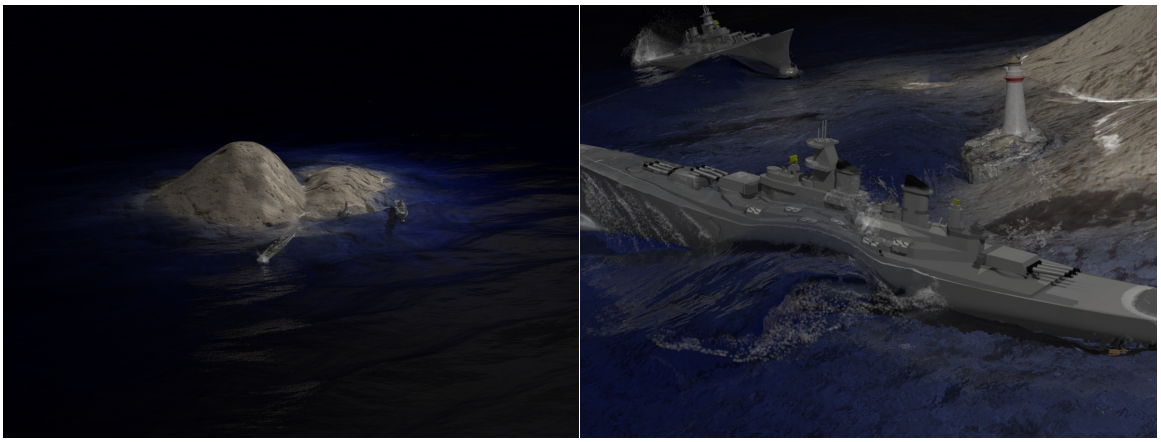


Figure 8.8: Two ships in stormy seas near Longfellow island. We refine the domain near the ships by placing grids in their object spaces to add detail and allow them to propel themselves using their two-way solid-fluid coupled propellers.

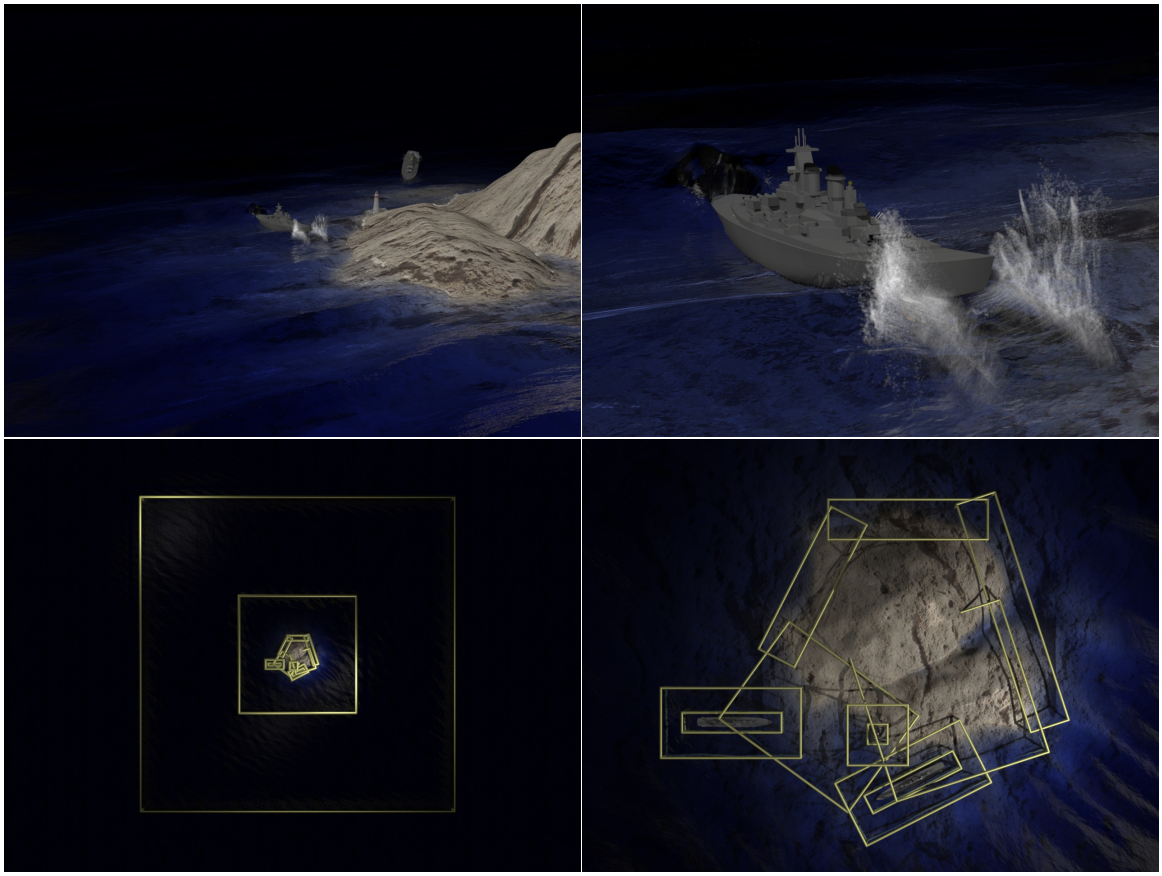


Figure 8.9: Another view from the large scale island example in Figure 8.8. Each ship is driven by two propellers and uses the same attached grids as in the case of the single ship example in Figure 8.7. We initialize the fluid surface and velocity with a large number of rolling waves which slowly move across the domain. As a result of the large domain, the simulation does not require seeding additional waves using boundary conditions or external forces within the time scale of the simulation. The top two figures show the spray produced by the ship propellers when they are exposed on the surface of the water, noting that both figures are of the same frame. The lower two figures show the initial grid placement around the ships and island.

Chapter 9

Conclusion and Future Work

This dissertation has presented a novel adaptive scheme for simulating the incompressible Navier-Stokes equations allowing multiple overlapping and arbitrarily translated and oriented Cartesian grids to be used as a composite domain. We have developed both first and second order accurate ALE semi-Lagrangian advection schemes allowing each grid to be advected independently by exchanging boundary conditions in ghost cells. We have developed a monolithic second order Poisson equation solver using a Voronoi diagram spatial discretization in order to combine the grids into a single continuous symmetric positive definite discretization. In order to compute the Voronoi diagram we have developed a simple and robust meshing scheme which scales well in parallel implementations by requiring that the geometry only be computed at intergrid boundaries. By utilizing a Voronoi diagram, our discretization uses second order accurate centered pressure differences which are orthogonal to their corresponding faces allowing hydrostatic cases to be solved exactly. We have also extended the Poisson solver to solve diffusion equations on cell centers directly and to solve for viscous forces on staggered velocity fields. By exploiting a Chimera grid approach we have preserved the accurate finite differences, lightweight cache coherent memory layouts and straightforward domain decomposition aspects of Cartesian grids. Unlike AMR approaches which are generally limited to axis aligned grids only, we are able to

efficiently represent non-grid aligned features. In some ways this is analogous to the second order accurate piecewise-linear interface calculation (PLIC) scheme of volume of fluid (VOF) methods as opposed to the first order simple line interface calculation (SLIC) scheme (see [80, 49]).

We have also proposed a set of modifications for allowing using the particle level set method on overlapping grids for free surface problems. We included a number of examples which demonstrated the ability of the method to produce accurate physical behavior in two-way solid-fluid coupling problems by locally refining space near solid objects.

There are numerous avenues for future research. We note that our pressure projection introduced some artifacts in the vorticity along intergrid boundaries when the solution on the Voronoi discretization is mapped back to the Cartesian grids. For graphics applications, this could limit the use of vorticity confinement, although the velocity field is visibly smooth and first order convergent. This mapping between the Voronoi mesh and Cartesian grids also did not guarantee zero divergence at Cartesian cells with interpolated faces. Examining this property more closely could certainly lead to a more accurate mapping that addresses both issues. Certain free surface flow examples also contained visual artifacts, particularly as waves travelled along grid boundaries in which case waves were clearly more resolved on the fine grid causing a visually noticeable grid seam when rendering with a highly specular shader. Conservative advection (see e.g. [64, 37]) also poses interesting issues since one would need to account for the duplication of values in overlapped regions in order to guarantee conservation. As we continue to scale our solver to large systems we expect the linear system to become the limiting factor particularly as the incomplete Cholesky preconditioner becomes less effective since we compute it only on the diagonal block local to the computational node when computing in parallel. Multigrid preconditioners hold significant promise and we believe that our SPD formulation could greatly ease their implementation for more complicated geometries.

Overall, although our approach loses some flexibility while compared to fully unstructured methods, it strikes an optimal balance between structure and adaptivity, allowing large scale simulations with features on many scales to be concurrently simulated while utilizing modern computer hardware and distributed computing resources. Figures 8.8 and 8.9 demonstrate the potential of the method.

Bibliography

[1] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, 148:2–22, 1999.

[2] M.J. Aftosmis, M.J. Berger, and S.M. Murman. Applications of space-filling curves to cartesian methods for cfd. *AIAA Paper*, 1232:2003–1237, 2004.

[3] A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome. A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.

[4] C. Batty, S. Xenos, and B. Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Proceedings of Eurographics*, 2010.

[5] J. A. Benek, P. G. Buning, and J. L. Steger. A 3-d chimera grid embedding technique. In *7th Computational Fluid Dynamics Conference*, pages 322–331. AIAA, 1985.

[6] J. A. Benek, T. L. Donegan, and N. E. Suhs. Extended chimera grid embedding scheme with applications to viscous flows. In *8th Computational Fluid Dynamics Conference*, pages 283–391. AIAA, 1987.

[7] J. A. Benek, J. L. Steger, and F. C. Dougherty. A flexible grid embedding technique with applications to the euler equations. In *6th Computational Fluid Dynamics Conference*, pages 373–382. AIAA, 1983.

[8] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[9] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[10] J. U. Brackbill, D. B. Kothe, and H. M. Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48:25–38, 1988.

[11] J. U. Brackbill and H. M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.*, 65:314–343, 1986.

[12] T. Brochu, C. Batty, and R. Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 47:1–47:9, 2010.

[13] T. M. Burton and J. K. Eaton. Analysis of a fractional-step method on overset grids. *J. Comput. Phys.*, 177(2):336–364, 2002.

[14] G. Carré, S. D. Pino, B. Després, and E. Labourasse. A cell-centered lagrangian hydrodynamics scheme on general unstructured meshes in arbitrary dimension. *J. Comput. Phys.*, 228(14):5160–5183, 2009.

[15] P. M. Carrica, R. V. Wilson, R. W. Noack, and F. Stern. Ship motions using single-phase level set with dynamic overset grids. *Computers and Fluids*, 36(9):1415–1433, 2007.

[16] L. Chang and G. Yuan. An efficient and accurate reconstruction algorithm for the formulation of cell-centered diffusion schemes. *J. Comput. Phys.*, 231(20):6935–6952, 2012.

[17] H. Chen, C. Min, and F. Gibou. A supra-convergent finite difference scheme for the poisson and heat equations on irregular domains and non-graded adaptive

cartesian grids. *J. Sci. Comput.*, 31(1):19–60, 2007.

[18] Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 219–228, 2007.

[19] G. Chesshire and W. D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *J. Comput. Phys.*, 90(1):1–64, 1990.

[20] A. Chorin. A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.*, 2:12–26, 1967.

[21] Jonathan M. Cohen, Sarah Tariq, and Simon Green. Interactive fluid-particle simulation using translating eulerian grids. In *Proc. of the 2010 ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, pages 15–22, 2010.

[22] F. Colin, R. Egli, and F. Lin. Computing a null divergence velocity field using smoothed particle hydrodynamics. *J. Comput. Phys.*, 217:680–692, 2006.

[23] S. L. Cornford, D. F. Martin, D. T. Graves, D. F. Ranken, A. M. Le Brocq, R. M. Gladstone, A. J. Payne, E. G. Ng, and W. H. Lipscomb. Adaptive mesh, finite volume modeling of marine ice sheets. *J. Comput. Phys.*, 2012. In Press.

[24] S. Cummins and M. Rudman. An SPH projection method. *J. Comput. Phys.*, 152(2):584–607, 1999.

[25] M. Desbrun and M.-P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In R. Boulic and G. Hegron, editors, *Comput. Anim. and Sim. '96 (Proc. of EG Wrkshp. on Anim. and Sim.)*, pages 61–76. Springer-Verlag, Aug 1996.

[26] Yoshinori Dobashi, Yasuhiro Matsuda, Tsuyoshi Yamamoto, and Tomoyuki Nishita. A fast simulation method using overlapping grids for interactions between smoke and rigid objects. *Comput. Graph. Forum*, 27(2):477–486, 2008.

[27] J. K. Dukowicz, M. C. Cline, and F. L. Addessio. A general topology godunov method. *J. Comput. Phys.*, 82(1):29–63, 1989.

[28] R.E. English, L. Qiu, Y. Yu, and R. Fedkiw. An adaptive discretization of incompressible flow using a multitude of moving cartesian grids. *(submitted)*, 2012.

[29] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183:83–116, 2002.

[30] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):736–744, 2002.

[31] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM2003–45144. ASME*, 2003.

[32] B. Feldman, J. O'Brien, B. Klingner, and T. Goktekin. Fluids in deforming meshes. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 255–259, 2005.

[33] L. Ge and F. Sotiropoulos. A numerical method for solving the 3d unsteady incompressible navier-stokes equations in curvilinear domains with complex immersed boundaries. *J. Comput. Phys.*, 225(2):1782–1809, 2007.

[34] U. Ghia, K. Ghia, and C. Shin. High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comput. Phys.*, 48:387–411, 1982.

[35] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics-theory and application to nonspherical stars. *Mon. Not. R. Astron. Soc.*, 181:375, 1977.

[36] Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey,

and Ming Lin. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph.*, 31(6):148:1–148:9, 2012.

[37] J. Grétarsson and R. Fedkiw. Fully conservative, robust treatment of thin shell fluid-structure interactions in compressible flows. *(submitted)*, 2012.

[38] F. Harlow and J. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Phys. Fluids*, 8:2182–2189, 1965.

[39] F. H. Harlow. The particle-in-cell computing method for fluid dynamics. *Methods in Computational Physics*, 3:319, 1963.

[40] W. D. Henshaw. A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids. *J. Comput. Phys.*, 113(1):13–25, 1994.

[41] W. D. Henshaw, H.-O. Kreiss, and L. G. M. Reyna. A fourth-order-accurate difference approximation for the incompressible navier-stokes equations. *Computers and Fluids*, 23(4):575–593, 1994.

[42] William D. Henshaw. On multigrid for overlapping grids. *SIAM Journal on Scientific Computing*, 26(5):1547–1572, 2005.

[43] William D. Henshaw and G. S. Chesshire. Multigrid on composite meshes. *SIAM Journal on Scientific and Statistical Computing*, 8(6):914–923, 1987.

[44] William D. Henshaw and Donald W. Schwendeman. An adaptive numerical scheme for high-speed reactive flow on overlapping grids. *J. Comput. Phys.*, 191:420–447, 2003.

[45] William D. Henshaw and Donald W. Schwendeman. Moving overlapping grids with adaptive mesh refinement for high-speed reactive and non-reactive flow. *J. Comput. Phys.*, 216(2):744–779, 2006.

[46] William D. Henshaw and Donald W. Schwendeman. Parallel computation of three-dimensional flows using overlapping grids with adaptive mesh refinement. *J. Comput. Phys.*, 227(16):7469–7502, 2008.

[47] M. Hinatsu and J. H. Ferziger. Numerical computation of unsteady incompressible flow in complex geometry using a composite multigrid technique. *Int. J. Num. Meth. Fluids*, 13(8):971–997, 1991.

[48] C. Hirt, A. Amsden, and J. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J. Comput. Phys.*, 135:227–253, 1974.

[49] C. Hirt and B. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[50] D. C. Jespersen, T. H. Pulliam, and P. G. Buning. Recent enhancements to OVERFLOW. paper 97-0644, AIAA, 1997.

[51] C. Ji, A. Munjiza, and J. J. R. Williams. A novel iterative direct-forcing immersed boundary method and its finite volume applications. *J. Comput. Phys.*, 231(4):1797–1821, 2012.

[52] S. Y. Kadioglu and M. Sussman. Adaptive solution techniques for simulating underwater explosions and implosions. *J. Comput. Phys.*, 227:2083–2104, 2008.

[53] Y. Kallinderis and H. T. Ahn. Incompressible navier-stokes method with general hybrid meshes. *J. Comput. Phys.*, 210(1):75–108, 2005.

[54] P. K. Khosla and S. G. Rubin. A diagonally dominant second-order accurate implicit scheme. *Computers and Fluids*, 2(2):207–209, 1974.

[55] C Kiris, D Kwak, S Rogers, and I-D Chang. Computational approach for probing the flow through artificial heart devices. *Journal of biomechanical engineering*, 119(4):452–460, 1997.

[56] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 25(3):820–825, 2006.

[57] B. Koobus and C. Farhat. On the implicit time integration of semi-discrete viscous fluxes on unstructured dynamic meshes. *Int. J. Num. Meth. Fluids*, 29(8):975–996, 1999.

[58] B. Koobus and C. Farhat. Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. *Comp. Meth. Appl. Mech. Eng.*, 170(1):103–129, 1999.

[59] S. Koshizuka, H. Tamako, and Y. Oka. A particle method for incompressible viscous flows with fluid fragmentation. *Comput. Fluid Dyn. J*, 1995.

[60] M. Kucharik, M. Shashkov, and B. Wendroff. An efficient linearity-and-bound-preserving remapping method. *J. Comput. Phys.*, 188(2):462–471, 2003.

[61] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph.*, 26(3), July 2007.

[62] Jinmo Lee and Donghyun You. An implicit ghost-cell immersed boundary method for simulations of moving body problems with control of spurious force oscillations. *Journal of Computational Physics*, 233(0):295 – 314, 2013.

[63] Jongho Lee, Jungwoo Kim, Haecheon Choi, and Kyung-Soo Yang. Sources of spurious force oscillations from an immersed boundary method for moving-body problems. *Journal of Computational Physics*, 230(7):2677 – 2695, 2011.

[64] M. Lentine, J.T. Grétarsson, and R. Fedkiw. An unconditionally stable fully conservative semi-lagrangian method. *J. Comput. Phys.*, 230:2857–2879, 2011.

[65] T. Linde and P. L. Roe. An adaptive cartesian mesh algorithm for the euler equations in arbitrary geometries. In *9th Computational Fluid Dynamics*

Conference, pages 1–7. AIAA, 1989.

[66] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 35:995–1010, 2006.

[67] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:457–462, 2004.

[68] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE TVCG*, 14(4):797–804, 2008.

[69] R. Loubčre and M. J. Shashkov. A subcell remapping method on staggered polygonal grids for arbitrary-Lagrangian-Eulerian methods. *J. Comput. Phys.*, 209(1):105–138, 2005.

[70] R. Loubčre, M. Staley, and B. Wendroff. The repair paradigm: new algorithms and applications to compressible flow. *J. Comput. Phys.*, 211(2):385–404, 2006.

[71] L. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical J.*, 82:1013–1024, 1977.

[72] P.-H. Maire and B. Nkonga. Multi-scale godunov-type method for cell-centered discrete lagrangian hydrodynamics. *J. Comput. Phys.*, 228(3):799–821, 2009.

[73] L. G. Margolin and M. Shashkov. Remapping, recovery and repair on a staggered grid. *Comp. Meth. Appl. Mech. Eng.*, 193(39-41):4139–4155, 2004.

[74] L. G. Margolin and Mikhail Shashkov. Second-order sign-preserving conservative interpolation (remapping) on general grids. *J. Comput. Phys.*, 184(1):266–298, 2003.

[75] D. F. Martin, P. Colella, and D. Graves. A cell-centered adaptive projection method for the incompressible navier-stokes equations in three dimensions. *J. Comput. Phys.*, 227(3):1863–1886, 2008.

[76] D. J. Mavriplis and Z. Yang. Construction of the discrete geometric conservation law for high-order time-accurate simulations on dynamic meshes. *J. Comput. Phys.*, 213(2):557–573, 2006.

[77] C. Min and F. Gibou. A second order accurate projection method for the incompressible Navier-Stokes equation on non-graded adaptive grids. *J. Comput. Phys.*, 219:912–929, 2006.

[78] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Meshing Roundtable*, pages 103–114, 2003.

[79] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 154–159, 2003.

[80] W. Noh and P. Woodward. SLIC (simple line interface calculation). In *5th International Conference on Numerical Methods in Fluid Dynamics*, pages 330–340, 1976.

[81] S. E. Norris, C. J. Were, P. J. Richards, and G. D. Mallinson. A voronoi-based ale solver for the calculation of incompressible flow on deforming unstructured meshes. *Int. J. Num. Meth. Fluids*, 65(10):1160–1179, 2011.

[82] Sanjit Patel, Anson Chu, Jonathan Cohen, and Frederic Pighin. Fluid simulation via disjoint translating grids. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[83] G. S. H. Pau, J. B. Bell, A. S. Almgren, K. M. Fagnan, and M. J. Lijewski. An adaptive mesh refinement algorithm for compressible two-phase flow in porous media. *Computational Geosciences*, 16:577–592, 2012.

[84] J. B. Perot and V. Subramanian. Discrete calculus methods for diffusion. *J. Comput. Phys.*, 224(1):59–81, May 2007.

[85] S. Popinet. Gerris: A tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Comput. Phys.*, 190:572–600, 2003.

[86] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker. Particle–based simulation of fluids. In *Comp. Graph. Forum (Eurographics Proc.)*, volume 22, pages 401–410, 2003.

[87] M. M. Rai. A conservative treatment of zonal boundaries for euler equation calculations. *J. Comput. Phys.*, 62(2):472–503, 1986.

[88] M. M. Rai. A relaxation approach to patched-grid calculations with the euler equations. *J. Comput. Phys.*, 66(1):99–131, 1986.

[89] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 193–202, 2004.

[90] A. Robinson-Mosher, C. Schroeder, and R. Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. *J. Comput. Phys.*, 230:1547–66, 2011.

[91] A. Robinson-Mosher, T. Shinar, J. T. Grétarsson, J. Su, and R. Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. on Graphics*, 27(3):46:1–46:9, August 2008.

[92] A. M. Roma, C. S. Peskin, and M. J. Berger. An adaptive version of the immersed boundary method. *J. Comput. Phys.*, 153(2):509–534, 1999.

[93] C. Schroeder, W. Zheng, and R. Fedkiw. Implicit surface tension formulation with a lagrangian surface mesh on an eulerian simulation grid. *J. Comput. Phys.*, 231:2092–2115, 2012.

[94] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. An Unconditionally Stable MacCormack Method. *J. Sci. Comp.*, 35(2):350–371, 2008.

[95] Jung Hee Seo and Rajat Mittal. A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. *Journal of Computational Physics*, 230(19):7347 – 7363, 2011.

[96] Maurya Shah, Jonathan M. Cohen, Sanjit Patel, Penne Lee, and Frédéric Pighin. Extended galilean invariance for adaptive fluid simulation. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 213–221, 2004.

[97] F. Sin, A. W. Bargteil, and J. K. Hodgins. A point-based method for animating incompressible flow. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, SCA '09, pages 247–255, New York, NY, USA, 2009. ACM.

[98] J. L. Steger and J. A. Benek. On the use of composite grid schemes in computational aerodynamics. *Comp. Meth. Appl. Mech. Eng.*, 64(1-3):301–320, 1987.

[99] J. L. Steger, F. C. Dougherty, and J. A. Benek. *Advances in Grid Generation*, volume 5, chapter A Chimera Grid Scheme. ASME, New York, 1985.

[100] M. Sussman, A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome. An adaptive level set approach for incompressible two-phase flows. *J. Comput. Phys.*, 148:81–124, 1999.

[101] K. Szewc, J. Pozorski, and J.-P. Minier. Analysis of the incompressibility constraint in the smoothed particle hydrodynamics method. *Int. J. Num. Meth. Eng.*, 92(4):343–369, 2012.

[102] Yusuke Tahara, Robert V. Wilson, Pablo M. Carrica, and Frederick Stern. Rans simulation of a container ship using a single-phase level-set method with overset grids and the prognosis for extension to a self-propulsion simulator. *Journal of Marine Science and Technology*, 11(4):209–228, 2006.

[103] Jie Tan, Xubo Yang, Xin Zhao, and Zhanxin Yang. Fluid animation with multi-layer grids. In *ACM SIGGRAPH/Eurographics Symposium of Computer*

Animation 2008 Posters, 2008.

[104] J. Tessendorf. Simulating ocean water. In *SIGGRAPH 2002 Course Notes #9 (Simulating Nature: Realistic and Interactive Techniques)*. ACM Press, 2002.

[105] P. Traoré, Y. M. Ahipo, and C. Louste. A robust and efficient finite volume scheme for the discretization of diffusive flux on extremely skewed meshes in complex geometries. *J. Comput. Phys.*, 228(14):5148–5159, 2009.

[106] J. G. Trulio. Theory and structure of the afton codes. Technical report, Air Force Weapons Laboratory, June 1966.

[107] Z. J. Wang. A fully conservative interface algorithm for overlapped grids. *J. Comput. Phys.*, 122(1):96–106, 1995.

[108] R. V. Wilson, P. M. Carrica, and F. Stern. Simulation of ship breaking bow waves and induced vortices and scars. *Int. J. Num. Meth. Fluids*, 54(4):419–451, 2007.

[109] H. Yoon, S. Koshizuka, and Y. Oka. A particle-gridless hybrid method for incompressible flows. *Int. J. Num. Meth. Fluids*, 30:407–424, 1999.