An adaptive discretization of incompressible flow using a multitude of moving Cartesian grids

R. Elliot English*, Linhai Qiu**, Yue Yu**, Ronald Fedkiw**

Stanford University, 353 Serra Mall Room 207, Stanford, CA 94305

Abstract

We present a novel method for discretizing a multitude of moving and overlapping Cartesian grids each with an independently chosen cell size to address adaptivity. 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. Likewise, an implicit temporal discretization is used for the parabolic terms, such as the heat equation and Navier-Stokes viscosity. The most intricate aspect of any such discretization is the method used in order to solve the elliptic equation for the Navier-Stokes pressure or that resulting from the temporal discretization of parabolic terms. We address this by first removing any degrees of freedom which duplicately cover spatial regions due to overlapping grids, and then providing a discretization for the remaining degrees of freedom adjacent to these regions. We observe that a robust second order accurate symmetric positive definite readily preconditioned discretization can be obtained by constructing a local Voronoi region on the fly for each degree of freedom in question in order to obtain both its stencil (logically connected neighbors) and stencil weights. We independently demonstrate each aspect of our approach on test problems in order to show efficacy and convergence before finally addressing a number of common test cases for incompressible flow with potentially moving solid bodies.

1. Introduction

Adaptive discretizations are important in many incompressible flow problems since it is often necessary to resolve details on multiple levels. In fluid structure interaction problems it is critical to resolve the turbulent flows in the wakes behind objects in order to accurately predict even large scale behaviors. In many problems accurately modeling far field boundary conditions is also important and necessitates a method that allows large regions of space to be modeled using a reduced number of degrees of freedom. 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. [37, 36, 52]), hexahedra and other irregularly shaped (see e.g. [21, 48, 13, 34]) and non-linear elements, as well as meshless methods which use disjoint particles such as Smoothed Particle Hydrodynamics (SPH) (see e.g. [25, 47, 71, 20, 18]) and the Moving-Particle Semi-Implicit method (see e.g. [38, 76]). 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. [73, 31]), 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. [50, 49, 45, 46, 39]). 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 [27]) and the Fluid-Implicit Particle method (FLIP) (see [10, 9]) which use Lagrangian particles for advection and a background grid to solve for pressure. Other authors have explored

*eenglish@gmail.com, Stanford University

**{lqiu,yuey,rfedkiw}@stanford.edu, Stanford University

Preprint submitted to Elsevier *November 2, 2012*

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. [44]). 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 spatially 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, the domain decomposition necessary to evenly distribute the computational work load and storage requirements can dwarf the mathematical issues.

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, 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 hand built curvilinear grids (see e.g. [17, 14, 75, 23]) where a regular Cartesian grid is deformed parametrically to conform closely to the solid interface. 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. [43, 42, 16, 53, 58]), and adaptive mesh refinement (AMR) (see e.g. [8, 7, 41, 2, 70, 51, 19, 55, 64, 33]). While allowing for similar levels of adaptivity as unstructured methods, octrees and similar hierarchical structures suffer from the same issues of cache coherency and domain decomposition, even if care is taken to maximize cache coherency and minimize indirections.

AMR methods allow multiple regular 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 and the benefits of those structures while providing for spatial adaptivity. In [8] these 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. [7]) 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). Chimera grid methods (see e.g. [6, 69, 4, 5, 68]) 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 represent a single domain without explicitly connecting the discretizations. Unfortunately these methods have been somewhat impeded by the lack of a simple and efficient numerical discretization for solving globally coupled equations. In particular, an accurate and monolithic symmetric positive definite discretization for solving the pressure Poisson equation has limited current methods.

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. 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 [12, 30] which can cause the solver to diverge. Certain authors (see e.g. [28, 29]) 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 preventing the use of faster classes of solvers. Coupling together grids has received significant attention within the literature (see e.g. [59, 60]). One popular approach to implicitly coupling grids together are cut cell methods (see e.g. [74]) 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. [35, 72]) 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. [15] 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 2.1. 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 4.1 where a Voronoi diagram (see e.g. [54]) 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 3.4. We then extend our discretization to solve heat equations on the cell centers of moving grids in Section 5. In order to solve for viscous forces on the staggered face velocity degrees of freedom we compute cell center velocities and then solve a heat 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 5.1.

Finally, we summarize our method for incompressible flow, including interaction with static and kinematic objects in Section 6. Numerical results are provided in Section 6.1 including a lid driven cavity example showing similar velocity profiles and vortex patterns as [24], 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 [32], 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.

2. Grids and Parallelization

Our Chimera grid simulation framework consists of a collection of grids that partition the simulated domain into regions of interest as shown in Figure 1. 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 velocities in world space to those in a grid's object space using the equations

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

$$\vec{v}_{\text{world}} = \mathbf{R}\vec{v}_{\text{object}} \tag{2}$$

where $\mathbf{R}$ is the orientation of the grid stored as a rotation matrix and $\vec{s}$ is the translation of the grid. 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 also allow grids' rigid frames to be pinned to the transformations of their respective rigid bodies. We also allow grids to 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. For scalar quantities, such as density, values stored at cell centers can be used directly since they are invariant to the orientation of the grid. For vector quantities, since we use a Marker-and-Cell (MAC) grid approach a scalar is stored at each face representing the component of the vector field in the direction normal to the face in world space. 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 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 this is not affected by translation).

2.1. Explicit Grid Coupling

While our method applies fully implicit coupling to compute the pressure, heat and viscosity terms across multiple grids, an explicit coupling scheme is suitable for operations that do not involve global communication/coupling. This sort of explicit coupling occurs most naturally near the boundary of the grid where the computational stencil that reaches across the boundary will require information from one of the other grids. Although one could simply interpolate the required value from the appropriate grid, this approach typically leads to issues with cache coherency, thus increasing computational cost.

Instead, 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, we can proceed to perform explicit operations on each grid independent of the other grids, which is typical of AMR approaches. We regard a cell/face as overlapped by a 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 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. Face centers also need to lie a distance of $\Delta x/2$ within the domain to be considered as overlapped, because the grids are generally not orthogonal and thus all dimension components of the grid that is interpolated from are required to compute a single scalar for each ghost face value. (Only in the special case of aligned grids, this $\Delta x/2$ restriction can be relaxed in the dimension parallel to the face normal.) When multiple grids overlap a given ghost cell, we

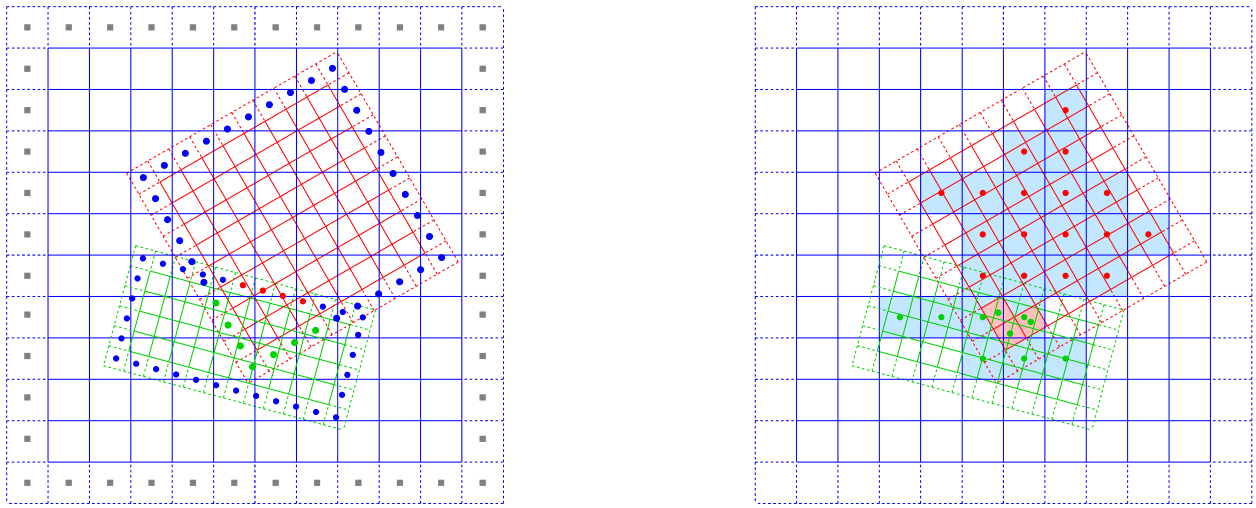


Figure 1: The explicit grid coupling between two partially overlapping grids (red and green) and a blue background grid. The green grid is finer than the red grid, and the blue grid is the coarsest. For clarity, each grid has only one layer of ghost cells in this example. (Left) Illustration of filling ghost cells. The dotted cells are the ghost cells of the grids. The color of each dot indicates which grid to get its real data from: most of the ghost cells of the red grid and the green grid are filled by interpolating real values from the blue background grid. However, where the red grid and the green grid overlap, the red grid's ghost cells are filled with real data from the green grid and vice versa. 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) Illustration of filling overlapped regions. The dotted cells are cells that are overlapped by other finer grids. The color of each dot indicates which grid the real data comes from, and the fill color of a cell helps clarify which grid the cell belongs to. Note that each blue shaded cell on the blue grid receives data from either the green or the red grid, preferably using the green grid whenever it is possible since it is finer. In addition, there are three red shaded cells that can be interpolated from the finer green grid. Note, as discussed in the text, updating these three red cells could affect the values the red grid gives to the blue grid, and therefore the order in which this is done matters.

always interpolate from the finest overlapping grid (see Figure 1 (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.

While the use of ghost cells and an appropriate time step restriction provides each grid with access to all the values they need for explicit operations, numerical tests showed that values on the interiors of overlapped grid regions could tend to gradually drift apart. For example, consider one fine grid in the interior of a coarse grid which obtains its ghost cells from the coarse grid. Subsequent calculations on the interior of that fine grid provide values that can drift away from those calculated on the underlying coarse grid. Moreover, the values calculated on the fine grid in our example 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 interior degree of freedom by a value interpolated from a finer grid that overlaps it (using the finest grid possible), as shown in Figure 1 (Right). Going back to our example, this two-way couples the grids with communication from fine to coarse by updating the coarse grid degrees of freedom that are overlapped by the fine grid. Note that this procedure can have a negative impact when using MPI, since the overlapped regions are not lower-dimensional, as are the ghost cell regions–therefore, its cost does not tend to zero upon grid refinement. However, assuming bounded grid motion, the cost of filling these overlapped regions can be optimized based on the observation that only a thin layer of cells near the boundary of the overlapped regions (see Figure 2) actually needs to be interpolated, and any numerical drift in the remaining interior cells will not affect real simulation data. This is because these cells will not be used as interpolation stencils for unoverlapped cells and will still be covered in the next time step due to the bounding of grid motion.

When filling overlapped cells, the order in which the grids are filled matters. As pointed out in Figure 1 (Right), the values the red grid sends to the blue grid can change if the three shaded red grid cells are first

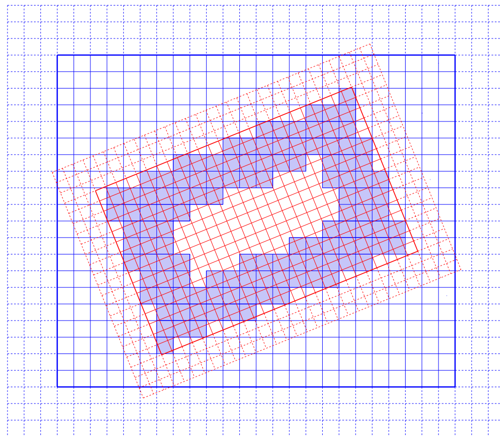


Figure 2: Illustration of the lower-dimensional layer of overlapped cells. A fine red grid is shown overlapping a coarse blue grid. In this example, we only fill the shaded blue grid cells with data from the red grid and otherwise ignore the interior hollowed out region of the blue grid. Based on the 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 the relevant data from the red to the blue grid.

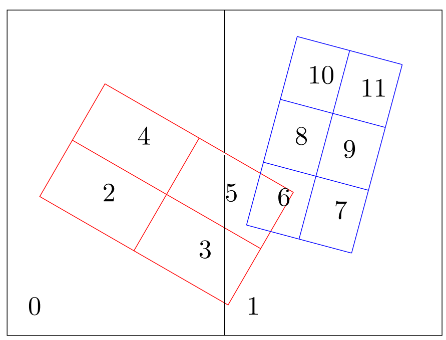


Figure 3: Grids are split into subgrids and assigned to multiple processors. Grids with the same color correspond to a single grid before splitting. The number on each subgrid indicates the index of the corresponding processor.

updated by the green grid. Therefore from the standpoint of accuracy, one should start by filling the second finest grid by the finest grid and proceed recursively to the coarser grids. In the serial implementation of our algorithm, the grids are sorted from the finest to the coarsest beforehand and subsequently filled in this order in-place. However, in a parallel implementation of our algorithm, one might want to use non-updated overlapping regions instead. That is, the non-updated red cells will be used to update the blue grid so that the process would not need to wait for their update from the green grid.

2.2. 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 each grid 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 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 3. Each of the 12 subgrids in Figure 3 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 3 will first sync up each of their associated subgrids with each other. After that the algorithm can proceed as usual ignoring the subgrids, 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.

3. 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 [66]. 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 2.1 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

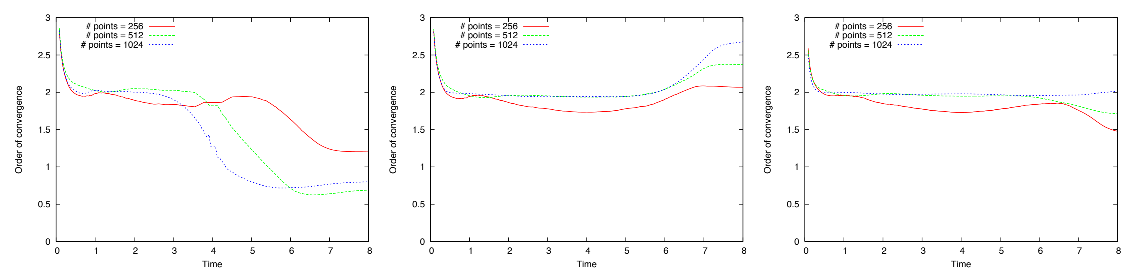


Figure 4: The order of accuracy of a time-varying single vortex test in two spatial dimensions with the SL-MacCormack method. (Left) uses velocities at time t^n for advection and a fixed CFL number. (Middle) uses velocities at time $t^{n+1/2}$ for advection and a fixed CFL number. (Right) uses velocities at time t^n for advection and fixes Δt to the value used on a 4096-point grid.

Before presenting our Chimera grid advection approach we discuss a few aspects of the SL-MacCormack method of [66], in order to clarify a few important details not discussed in the original work. In first order accurate semi-Lagrangian advection, characteristic paths are traced backwards to find locations from which to linearly interpolate new values. 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 value $\phi^n(\vec{x})$ forward in time to obtain $\hat{\phi}^{n+1}(\vec{x})$ and then advect this value backward in time to obtain $\hat{\phi}^n(\vec{x})$ and subsequently

$$E(\vec{x}) = (\hat{\phi}^n(\vec{x}) - \phi^n(\vec{x}))/2 \tag{3}$$

$$\phi^{n+1}(\vec{x}) = \hat{\phi}^{n+1}(\vec{x}) - e(\vec{x}) \tag{4}$$

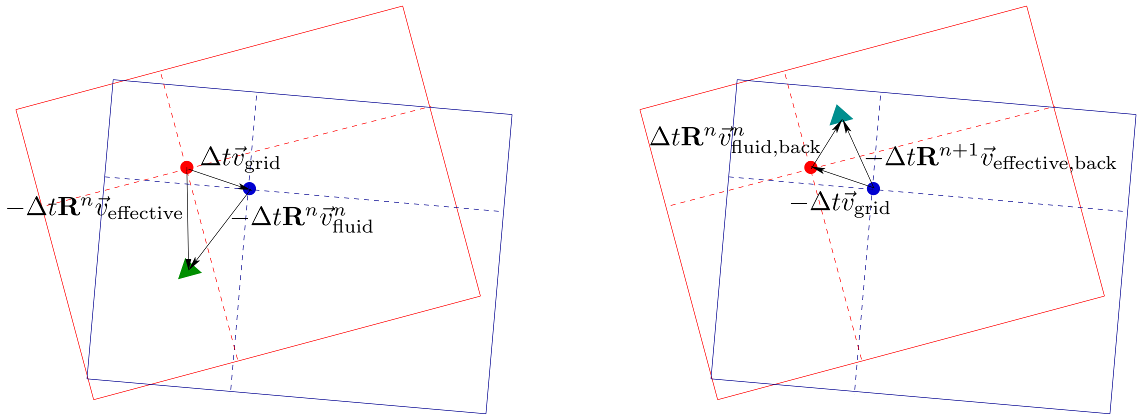


Figure 5: An illustration of our ALE advection in the world space for the point $\vec{x}_{\text{object}}$ that moves from $\vec{x}^n_{\text{world}}$ (shown as the red point) to $\vec{x}^{n+1}_{\text{world}}$ (shown as the blue point) according to the grid motion. (Left) Forward advection. (Right) Backward advection for SL-MacCormack.

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

Whereas [66] only considered time-constant velocity fields in their precise analysis of the method, we here consider time-varying velocity fields. First consider an initial circular bump function

$$\phi(\vec{x}) = \begin{cases} e^{\left(2-\frac{2}{1-(\|\vec{x}-\vec{x}_c\|/r)^2}\right)} & \text{if } \|\vec{x}-\vec{x}_c\| < r \\ 0 & \text{if } \|\vec{x}-\vec{x}_c\| \geq r \end{cases} \tag{5}$$

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

$$\Psi(x,y) = \frac{2}{\pi}\sin^2\left(\pi\frac{x-1}{2}\right)\sin^2\left(\pi\frac{y-1}{2}\right), \quad (x,y) \in [-1,1]\times[-1,1] \tag{6}$$

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 4. Note that second order accuracy is obtained by using velocities at time $t^{n+1/2}$, but not by using velocities at time t^n (which was not pointed out in [66]). To clarify that this is a temporal error, Figure 4 (Right) 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.

3.2. Chimera Advection Scheme

We use a standard MAC grid to store velocity fields, storing what corresponds to world space velocities on each grid when those grids are placed in world space. In other words, if a particular grid is rotated and translated into world space, and one interpolates a velocity at a given point within that grid, one obtains a world space velocity for that point. Thus, when the grid moves in world space, updating its rotation and translation from object to world space, all the component values of the velocity field defined in that grid are no longer valid. To determine the velocity at a point $x^n_{\text{world}} = \mathbf{R}^n\vec{x}_{\text{object}} + \vec{s}^n$ in world space, one needs to determine the appropriate point in object space $\vec{x}^n_{\text{object}} = (\mathbf{R}^n)^{-1}(\vec{x}^n_{\text{world}} - \vec{s}^n)$ to interpolate the object space velocity $\vec{v}^n_{\text{object}} = \vec{v}^n_{\text{object}}(\vec{x}_{\text{object}})$. This gives scalar values of a velocity field $\vec{v}^n_{\text{object}} = (u^n, v^n)^T$ corresponding to the components of axial directions of the grid in object space, $\vec{e}_1$ and $\vec{e}_2$, which when rotated to world space become $\vec{n}_1 = \mathbf{R}^n\vec{e}_1$ and $\vec{n}_2 = \mathbf{R}^n\vec{e}_2$, therefore making the world space velocity $\vec{v}^n_{\text{world}} = u^n\vec{n}_1 + v^n\vec{n}_2$. We write this more concisely as $\vec{v}^n_{\text{world}} = \mathbf{R}^n(u^n, v^n)^T = \mathbf{R}^n\vec{v}^n_{\text{object}}$.

After filling ghost cells and overlapping regions as discussed in Section 2, we update each interior point $\vec{x}_{\text{object}}$ of each grid as follows. 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 $\vec{x}^{n+1}_{\text{world}}$ in world space and therefore $\vec{x}_{\text{new}}(\vec{x}_{\text{object}}) = (\mathbf{R}^n)^{-1}(\vec{x}^{n+1}_{\text{world}} - \vec{s}^n)$ on our local grid. Note that unlike standard semi-Lagrangian advection, this arrival point $\vec{x}_{\text{new}}$ is typically not a grid point and therefore one must interpolate a velocity $\vec{v}^n_{\text{fluid}} = \vec{v}^n_{\text{object}}(\vec{x}_{\text{new}})$ to that point before a characteristic ray can be traced backwards in time. Note that $-\Delta t\mathbf{R}^n v^n_{\text{fluid}}$ is the vector tracing semi-Lagrangian characteristics backward from $\vec{x}^{n+1}_{\text{world}}$ to a point in world space (the green triangle in Figure 5) which contains the world space data that we will use to update the value ϕ at $\vec{x}_{\text{object}}$. Alternatively, this process can be summarized by compositing the vector $(\vec{x}^{n+1}_{\text{world}} - \vec{x}^n_{\text{world}})$ together with the vector $-\Delta t\mathbf{R}^n v^n_{\text{fluid}}$ to get an effective displacement $-\Delta t\mathbf{R}^n\vec{v}_{\text{effective}} = \Delta t\vec{v}_{\text{grid}} - \Delta t\mathbf{R}^n\vec{v}^n_{\text{fluid}}$, where $\vec{v}_{\text{grid}} = (\vec{x}^{n+1}_{\text{world}} - \vec{x}^n_{\text{world}})/\Delta t$, see Figure 5 (Left). This means in order to use the single grid advection code, one uses the effective velocity $\vec{v}_{\text{effective}} = -(\mathbf{R}^n)^{-1}\vec{v}_{\text{grid}} + \vec{v}^n_{\text{fluid}}$.

When updating velocities themselves, the scheme is a bit more complicated because of the way velocities are stored and the fact that the normal directions of faces $\vec{n}_1$ and $\vec{n}_2$ will change as grids rotate. To deal with this, we update a full velocity vector on each MAC grid face by applying semi-Lagrangian advection to both of the coordinate directions at this face location. This would result in the correct velocity vector on each face if the new grid orientation $\mathbf{R}^{n+1}$ were the same as the old orientation $\mathbf{R}^n$. However, since the orientation can change, we premultiply this 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. Finally, this velocity vector in object space is dot-producted with the coordinate directions of the grid $\vec{e}_1$ and $\vec{e}_2$ to store the appropriate components—that is, we store either the first or the second component of the velocity vector after premultiplying the updated velocity vector by $(\mathbf{R}^{n+1})^{-1}\mathbf{R}^n$.

The first step of SL-MacCormack advection is identical to the first order accurate semi-Lagrangian advection described above. In the second step, we fill the ghost cells of the forward advection results $\hat{\phi}^{n+1}$ and then advect these values backwards with the grid motion and fluid velocity reversed. This time $\vec{x}_{\text{object}}$ travels from $\vec{x}^{n+1}_{\text{world}}$ to $\vec{x}^n_{\text{world}}$. Conveniently, the fluid velocity at time t^n is already defined at the destination point (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. We evaluate this velocity as $\vec{v}^n_{\text{fluid,back}} = \vec{v}^n(\vec{x}_{\text{object}})$ using the standard MAC grid averaging operators one would use for the standard single-grid version of the scheme. Similar to semi-Lagrangian advection the world space location for interpolation is now $\vec{x}^n_{\text{world}} + \Delta t\mathbf{R}^n\vec{v}^n_{\text{fluid,back}}$, see Figure 5 (Right). (Note that if we were using the fluid velocity at time t^{n+1} instead of time t^n, the interpolation point would be $\vec{x}^n_{\text{world}} + \Delta t\mathbf{R}^{n+1}\vec{v}^{n+1}_{\text{fluid,back}}$. Note too that $\vec{v}^{n+1}_{\text{fluid,back}}$ 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.) Finally, we can obtain the effective velocity as $\vec{v}_{\text{effective,back}} = (\mathbf{R}^{n+1})^{-1}\vec{v}_{\text{grid}} - (\mathbf{R}^{n+1})^{-1}(\mathbf{R}^n)\vec{v}^n_{\text{fluid,back}}$. See Figure 5 (Right).

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 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.

$$\phi^{n+1}(\vec{x}_{\text{object}}) = \hat{\phi}^{n+1}(\vec{x}_{\text{object}}) - E(\vec{x}_{\text{new}}) \tag{7}$$

When updating a velocity field, the backward advection step advects a full velocity vector for each face, premultiplies 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

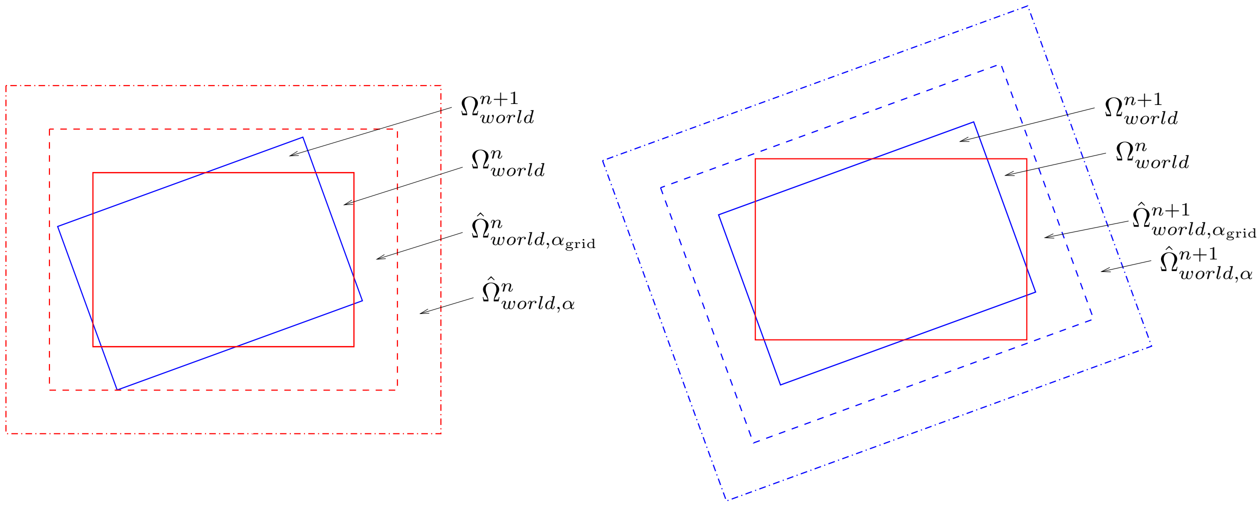


Figure 6: Illustration of the relation between the time step size and ghost cells. (Left) Forward advection. (Right) Backward advection for SL-MacCormack. The inner dashed boxes are the ghost domains bounding the grid motion—that is, the blue solid box should be completely inside the inner dashed red 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.

the scalar error correction for Equation 7, one premultiplies by $(\mathbf{R}^{n+1})^{-1}\mathbf{R}^n$ and takes the appropriate component corresponding to the face direction being considered.

3.3. Time Step Size and Ghost Cells

For a regular 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 $\vec{v}^n_{\text{fluid}}$ 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 6 (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 communication overheads 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 the grid to discontinuously change in position, because in that case a grid could move by a distance larger than the ghost region even if Δt is arbitrarily small. We do allow for a discontinuous velocity. Although a smaller Δt does not necessarily lead to less grid motion, a continuous position guarantees that the grid displacement 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 $\vec{x}_{\text{lookup}} = \vec{x}_{\text{object}} - \Delta t \vec{v}_{\text{effective}}$. 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 $\vec{v}^n_{\text{fluid}}$ 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

$$\Delta t_{\text{fluid}} \leq \alpha_{\text{fluid}} \frac{\min(\Delta x, \Delta y)}{\max|\vec{v}^n_{\text{fluid}}|} \tag{8}$$

for every point $\vec{x}_{\text{new}}$ in $\hat{\Omega}^n_{\text{world},\alpha_{\text{grid}}}$. As we use bilinear interpolation to compute $\vec{v}^n_{\text{fluid}}$ at these points, we can more conveniently apply Equation 8 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 6 Right). While values of $\vec{v}^n_{\text{fluid,back}}$ 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 $\vec{x}_{\text{lookup,back}} = \vec{x}_{\text{object}} - \Delta t \vec{v}_{\text{effective,back}}$, lies inside $\hat{\Omega}^{n+1}_{\text{world},\alpha}$. Since we are using the time t^n velocities $\vec{v}^n_{\text{fluid,back}}$ 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 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 from top left to bottom right, see Figure 7. 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 5 with different initial positions and radii specified below.

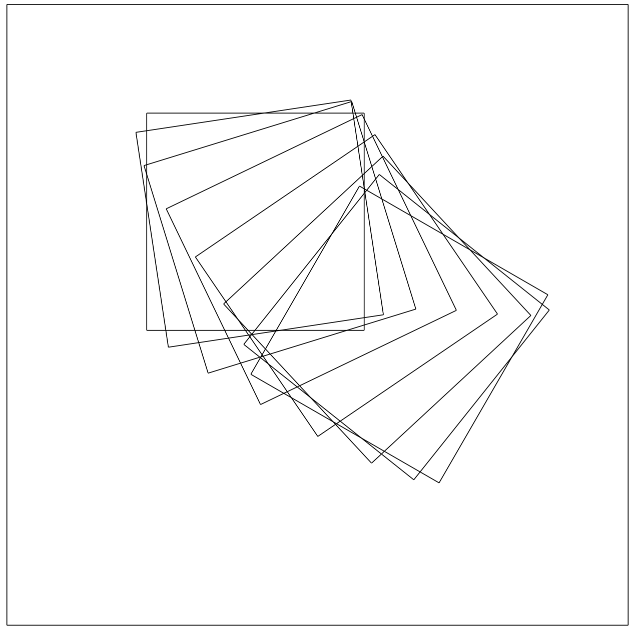


Figure 7: Illustration of motion of the grids. The background grid has no rotation or translation, while the fine grid spins and translates along a straight line inside the domain of the background grid.

The first test advects the density field through a constant uniform velocity field. Snapshots of the simulation are given in Figure 8. The initial position of the bump is $(-.75, .4)$ and the radius is $r = .2$. The uniform velocity field is $\vec{v} = (\sqrt{3}/4, -1/4)$. The results in Figure 9 show that both the ALE semi-Lagrangian and ALE SL-MacCormack methods converge to the analytic solution. Note that in Figure 9, 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 1.

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 $\vec{v}(x, y) = \frac{12\pi}{25}(-y, x)$. The L^1 norm errors of different resolution simulations are plotted in Figure 10, and the orders of accurary are calculated when the field is rotated exactly one cycle, see Table 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 4096-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 and the two-moving-grid simulations converge to the same solution. Figure 12 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 11 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.

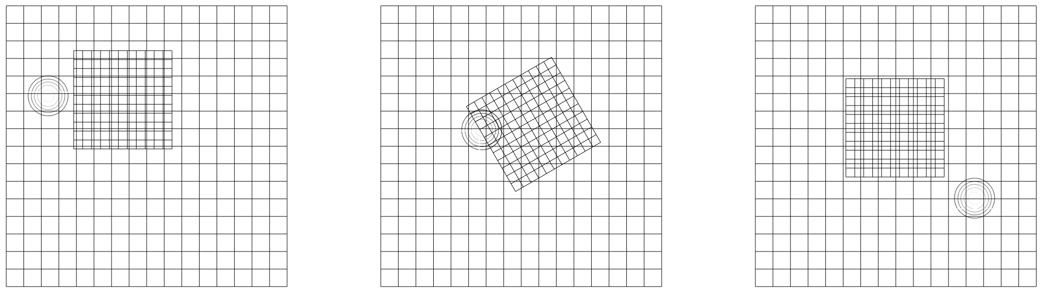


Figure 8: Snapshots of a simulation that advects a circular bump in a constant uniform velocity field. The circles shown in the figure are contours of the bump function. (Left) The snapshot at $t = 0$. (Middle) The snapshot at $t = 1$. (Right) The snapshot at $t = 3$.

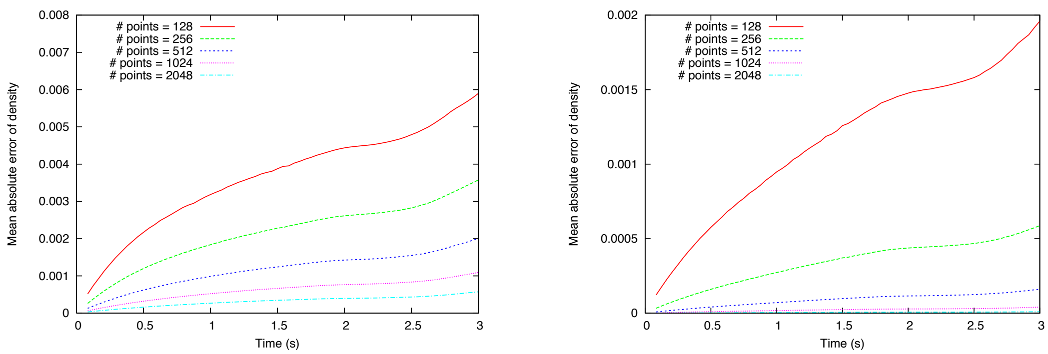


Figure 9: The L^1 norm error of the numerical solutions as the function of time in the constant uniform velocity test. (Left) The results of the ALE semi-Lagrangian method. (Right) The results of ALE SL-MacCormack method.

4. Poisson Equation

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

$$\nabla \cdot \beta(\vec{x})\nabla\phi(\vec{x}) = f(\vec{x}), \quad \vec{x} \in \Omega \tag{9}$$

$$\phi(\vec{x}) = g(\vec{x}), \quad \vec{x} \in \partial\Omega_D \tag{10}$$

$$\vec{n}(\vec{x}) \cdot \nabla\phi(\vec{x}) = h(\vec{x}), \quad \vec{x} \in \partial\Omega_N \tag{11}$$

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.

In Chimera grid methods, 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 13. Although Figure 13 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

	SL (time=3.0)				SL-MC (time=3.0)			
Number of Points	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 1: The order of accuracy of ALE semi-Lagrangian (SL) and SL-MacCormack (SL-MC) methods of the constant uniform velocity test.

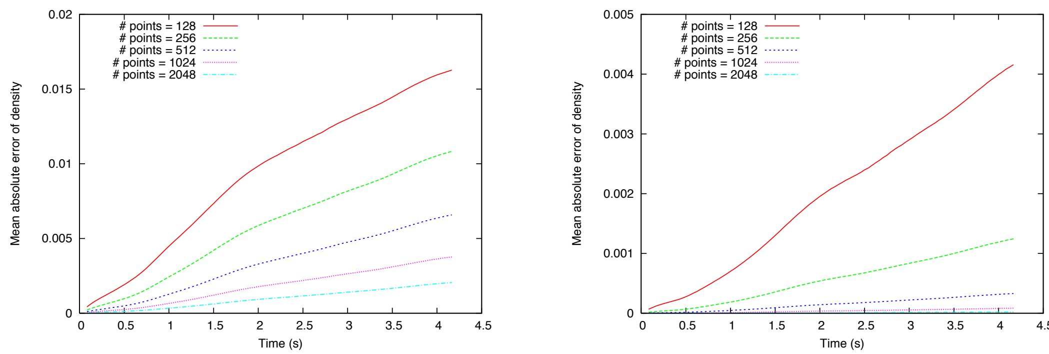


Figure 10: The L^1 norm errors of the numerical solutions as the function of time in the constant single vortex test. (Left) The results of ALE semi-Lagrangian method. (Right) The results of ALE SL-MacCormack method.

also important to minimize this overlap in order to prevent the solutions within these overlapping regions from drifting apart. See [17] for further discussion of these aspects of grid generation.

In order to enforce Dirichlet and 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 9-11 for two overlapping grids:

$$\mathbf{D}_1 \left(\mathbf{G}_1\boldsymbol{\phi}_1 + \mathbf{G}_{1g}\mathbf{J}_{1,2}\boldsymbol{\phi}_2 + \mathbf{H}_{1,2}\mathbf{G}_2\boldsymbol{\phi}_2\right) = \mathbf{f}_1 - \mathbf{D}_1\left(\mathbf{G}_{1d}\boldsymbol{\phi}_{1d} + \frac{\partial \boldsymbol{\phi}_{1n}}{\partial \vec{n}}\right) \quad (12)$$

$$\mathbf{D}_2 \left(\mathbf{G}_2\boldsymbol{\phi}_2 + \mathbf{G}_{2g}\mathbf{J}_{2,1}\boldsymbol{\phi}_1 + \mathbf{H}_{2,1}\mathbf{G}_1\boldsymbol{\phi}_1\right) = \mathbf{f}_2 - \mathbf{D}_2\left(\mathbf{G}_{2d}\boldsymbol{\phi}_{2d} + \frac{\partial \boldsymbol{\phi}_{2n}}{\partial \vec{n}}\right) \quad (13)$$

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 9, $\frac{\partial \phi_{1n}}{\partial \vec{n}}$ and $\frac{\partial \phi_{2n}}{\partial \vec{n}}$ are the Neumann boundary conditions on the computational domain boundary at the appropriate locations on grids 1 and 2 respectively as specified in Equation 11, and $\mathbf{D}_1$ 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 10. ϕ_{1d} and ϕ_{2d} are the Dirichlet conditions on the computational domain boundary on the appropriate locations of grids 1 and 2 respectively, as specified in Equation 10. $\mathbf{J}_{1,2}$ 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

	SL				SL-MC			
Number of Points	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 2: The order of accuracy of ALE semi-Lagrangian (SL) and SL-MacCormack (SL-MC) methods of the constant single vortex test when the bump has been rotated for one cycle.

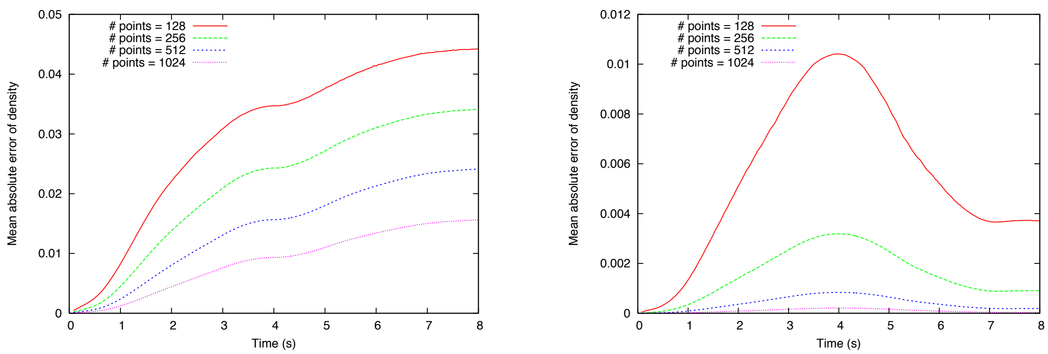


Figure 11: The L^1 norm error of the numerical solutions as the function of time in the time-varying single vortex test. (Left) The results of ALE semi-Lagrangian method. (Right) The results of ALE SL-MacCormack method.

1 to ghost faces on grid 2. We note that if one did not allow for a large enough overlap between the grids, additional terms would appear in Equations 12 and 13 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 [17] in which they state one would perform an additional implicit solve to compute the true interpolation weights as a preprocessing step, though they also seem to recommend supporting a large enough overlap such that this is not necessary.

When enforcing compatibility between the solutions on different grids using Neumann boundary conditions the system can be singular as illustrated in Figure 14. If we assume $\Delta x = 1$ then 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_2^1 are $\frac{\partial^2 \phi_2^1}{\partial x^2} = \phi_3^1 - 2\phi_2^1 + \phi_1^1$ and $\frac{\partial^2 \phi_2^2}{\partial x^2} = \phi_3^2 - 2\phi_2^2 + \phi_1^2$, respectively. Enforcing a Neumann boundary condition of $\frac{\partial \phi_{2.5}^1}{\partial x} = \phi_3^2 - \phi_2^2$ on grid 1, gives the modified Laplacian $\frac{\partial^2 \phi_2^1}{\partial x^2} = \phi_3^2 - \phi_2^2 - \phi_2^1 + \phi_1^1$. Enforcing a Neumann boundary condition of $\frac{\partial \phi_{1.5}^2}{\partial x} = \phi_2^1 - \phi_1^1$ on grid 2, gives the modified Laplacian $\frac{\partial^2 \phi_2^2}{\partial x^2} = \phi_3^2 - \phi_2^2 - \phi_2^1 + \phi_1^1$. 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.

In general Equations 12 and 13 do not yield a symmetric system. This asymmetry is fundamentally 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

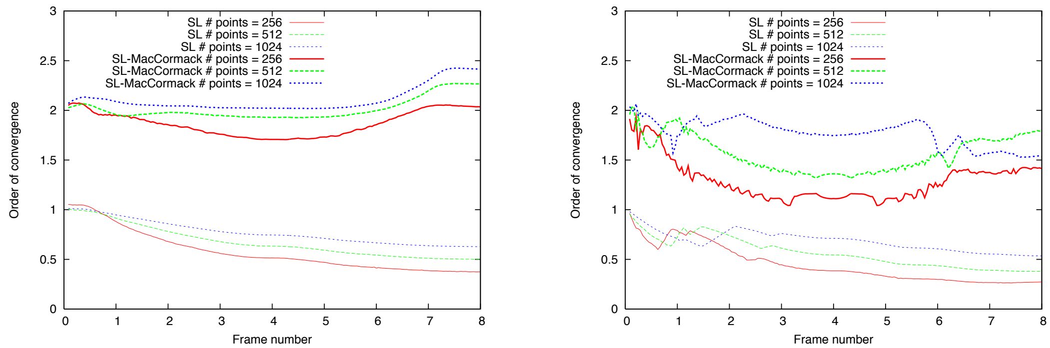


Figure 12: The order of accuracy of the time-varying single vortex test.(Left) The order of accuracy based on L^1 norm of errors. (Right) The order of accuracy based on L^∞ norm of the errors.

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. [63, 62]. One could follow this strategy by modifying Equations 12 and 13 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 12 and 13 is to use a block Gauss-Seidel outer iteration. 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. 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. At best these methods converge slowly. Other common approaches include multigrid solvers, monolithic iterative solvers such as GMRES, and direct methods. See [30, 56, 28, 29] for details on various approaches.

In an attempt to build a symmetric coupling between the grids, we first consider a cut cell approach as shown in Figure 15. 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 15 (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 [3] and [57]). 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 refinement. This is particularly evident at sharp corners along the intergrid boundary. Authors such as [72] 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.

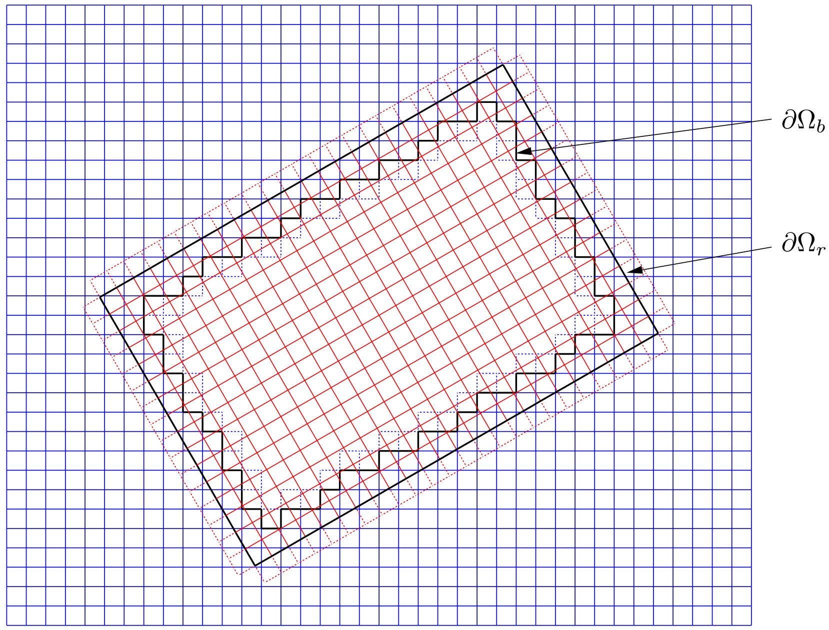


Figure 13: For fully overlapping grids one must omit a number of grid cells in the interior in order to provide boundaries for the application of boundary conditions, in this case creating an interior boundary, $\partial\Omega_b$, on the blue grid to receive information from the red grid. For partially 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.

4.1. Voronoi Diagram Discretization

Inspired by [67] and [11], we take the approach of using a Voronoi diagram to compute the coupling terms in Equations 12 and 13 as illustrated in Figure 16. 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 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 fast and stable solvers such as preconditioned conjugate gradient. However, note that this lower dimensional meshing must occur every time step if the grids are moving.

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. 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 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 noting that for all the interior

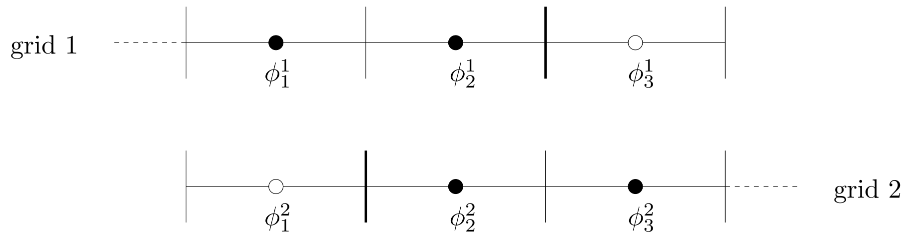


Figure 14: In one dimension, two grids overlap by 3 cells with the cell centered solution variables ϕ_1^1, ϕ_2^1 and ϕ_3^1 on the grid 1 and ϕ_1^2, ϕ_2^2 and ϕ_3^2 on the grid 2. In this case ϕ_3^1 and ϕ_1^2 are ghost cells where Dirichlet boundary conditions would be applied. The bold face lines between ϕ_2^1 and ϕ_3^1, and ϕ_1^2 and ϕ_2^2 represent where Neumann boundary conditions would be applied.

cells of each grid the Cartesian mesh already is the 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 using 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 that while building a Voronoi mesh can sometimes require sensitive calculations, this is only for arbitrary point sets and is not a concern for the highly 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.

We use a finite volume approach to compute the volume weighted divergence by defining it for each non-removed cell as the sum of the area-weighted normal components of the vector field at each incident face. The gradient is then computed at each face as the difference in the ϕ samples at its incident cell centers divided by the distance between the cell centers. Since we can view our Voronoi discretization as continuous we can formulate our final system as a single set of equations, which when using our definitions for the gradient and divergence operators, take the form of the following symmetric positive definite system with orthogonal gradient components computed at each face:

$$-\mathbf{V}_c\mathbf{D}\mathbf{G}\boldsymbol{\phi} = -\mathbf{V}_c\mathbf{f} + \mathbf{V}_c\mathbf{D}\left(\mathbf{G}_d\boldsymbol{\phi}_d + \frac{\partial\boldsymbol{\phi}_n}{\partial\vec{n}}\right) \tag{14}$$

where $\boldsymbol{\phi}$ contains the discrete values of ϕ at non-removed cells over the entire Chimera grid. $\mathbf{V}_c$ is a matrix with the Voronoi cell volumes as entries along the diagonal. Note that Equation 14 is the volume weighted form of Equation 9 in order to maintain symmetry. In order to solve Equation 14 we use incomplete Cholesky preconditioned conjugate gradient.

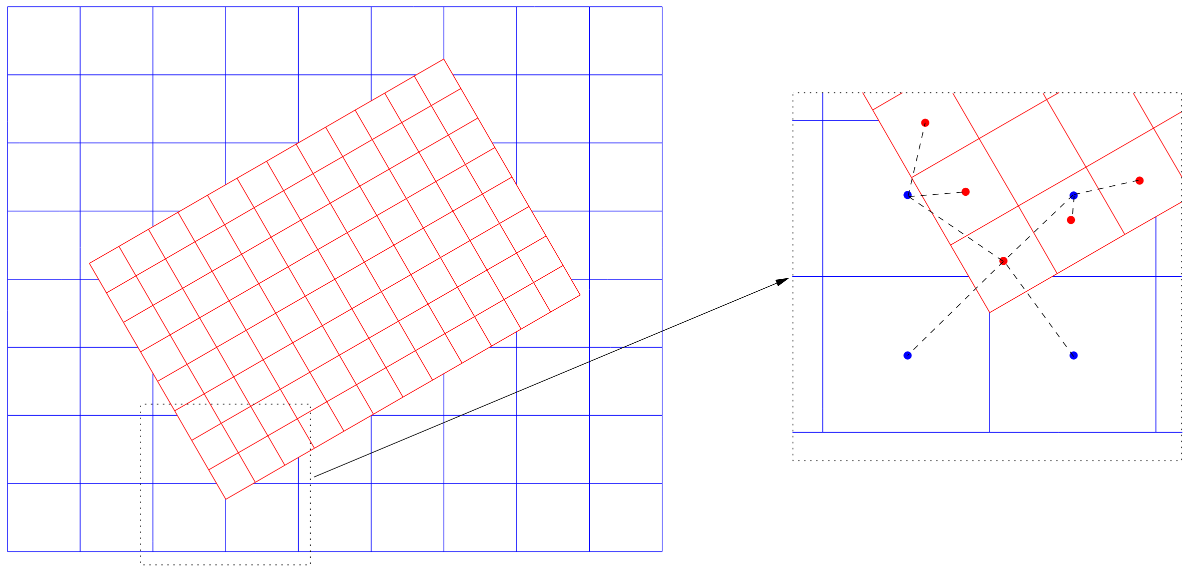


Figure 15: In this two grid example, we cut overlapped cells on the blue coarse grid by removing parts of these cells which are covered by finer red grid cells. The blue dots are the locations of ϕ samples on the blue grid. The red dots are the locations of ϕ samples on the fine red grid. The dashed 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.

4.2. Numerical Results

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.

	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 3: 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 17.

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 3 and shown in Figure 17. Table 4 gives the errors and orders of accuracy for $\phi(x,y)=sin(\pi x)sin(\pi y)$. Table 5 gives the errors and orders of accuracy for $\phi(x,y)=e^{-x^2-y^2}$. Table 6 gives the errors and orders of accuracy for $\phi(x,y)=e^x+e^y$. Table 7 gives the errors and orders of accuracy for $\phi(x,y)=e^x(x^2sin(y)+y^2)$.

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 8 and shown in Figure 18. Table 9 gives the errors and orders of accuracy for $\phi(x,y,z)=sin(\pi x)sin(\pi y)sin(\pi z)$. Table 10 gives the errors and orders of accuracy for $\phi(x,y)=e^{-x^2-y^2-z^2}$. Table 11 gives the errors and orders of accuracy for $\phi(x,y,z)=e^x+e^y+e^z$.

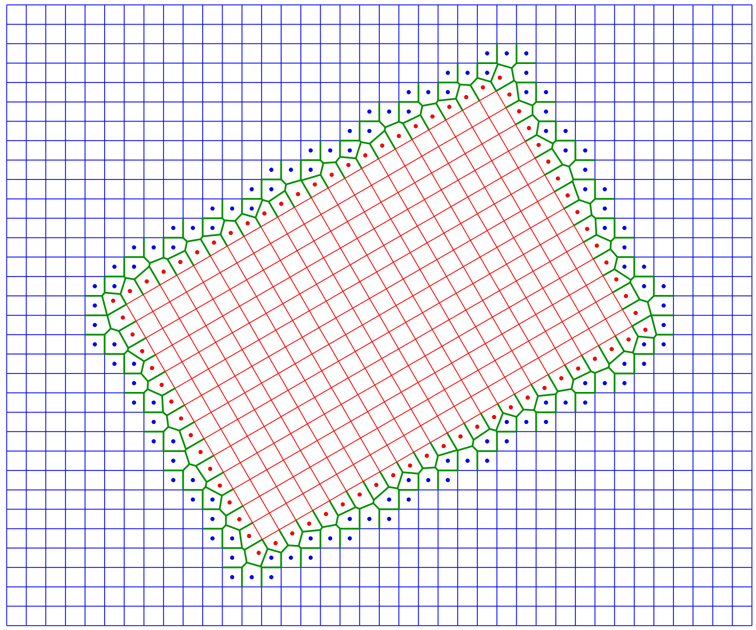


Figure 16: The composite grid resulting from our proposed method for coupling multiple grids by generating a Voronoi mesh along their boundaries. The dots are the cell centers of boundary cells that are incident to voronoi faces - note every dot corresponds to a degree of freedom originally from either the red or blue grid as indicated by their color.

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

4.2.1. 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 for the solution to be within 2, 4 and 8 significant digits of the exact solution to this system. When comparing the current solution to the exact solution we use the convergence tolerance $.5 \times 10^{\lfloor log_{10}(max|\phi|)\rfloor - n_s + 1}$ where n_s is the desired number of significant figures, $max|\phi|$ is the maximum absolute value of the exact solution, and $\lfloor \cdot \rfloor$ is the floor operator. 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 several of our examples are shown in Table 12 and Table 13 in two and three spatial dimensions, respectively. 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.

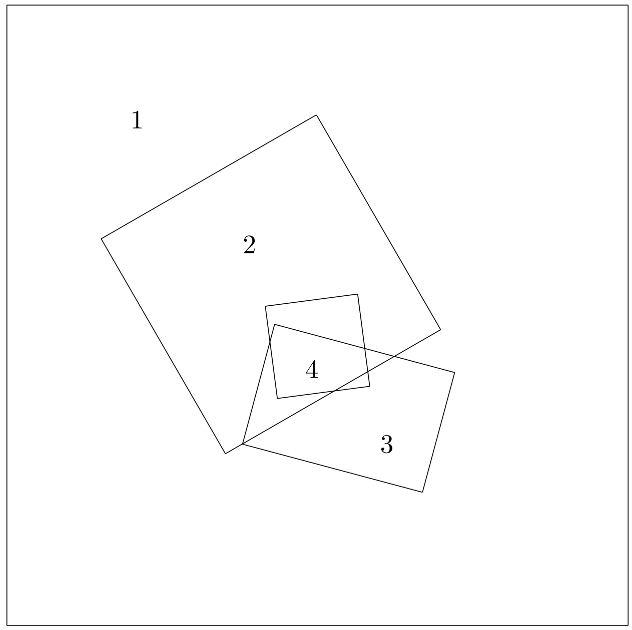


Figure 17: The domains of the grids used in our Poisson equation tests in two spatial dimensions.

n	L^1 Error	Order	L^∞ Error	Order
32	7.12×10^{-4}	–	1.65×10^{-3}	–
64	1.73×10^{-4}	2.05	3.99×10^{-4}	2.05
128	4.25×10^{-5}	2.02	1.03×10^{-4}	1.95
256	1.06×10^{-5}	2.01	2.63×10^{-5}	1.97
512	2.62×10^{-6}	2.01	6.61×10^{-6}	1.99

Table 5: Convergence results for solving a Poisson equation with analytic solution $\phi(x, y) = e^{-x^2-y^2}$.

5. Heat Equation

We next modify our Poisson solver to solve the following constant coefficient heat equation:

$$\frac{\partial \phi}{\partial t} = \nabla \cdot \beta(\vec{x}) \nabla \phi(\vec{x}), \qquad \vec{x} \in \Omega \tag{15}$$

$$\phi(\vec{x}) = g(\vec{x}, t), \qquad \vec{x} \in \partial\Omega_D \tag{16}$$

$$\vec{n}(\vec{x}) \cdot \nabla \phi(\vec{x}) = h(\vec{x}, t), \qquad \vec{x} \in \partial\Omega_N \tag{17}$$

where β is the diffusion coefficient. We first solve Equations 15-17 for values of ϕ located at cell centers by modifying Equation 14 from Section 4.1 to implement a backward Euler time integration scheme by solving the following symmetric positive definite system:

$$\left(\mathbf{V}_c - \Delta t \beta \mathbf{V}_c \mathbf{D} \mathbf{G}\right) \boldsymbol{\phi}^{n+1} = \mathbf{V}_c \boldsymbol{\phi}^n + \Delta t \beta \mathbf{V}_c \mathbf{D} \left(\mathbf{G}_d \boldsymbol{\phi}_d^{n+1} + \frac{\partial \boldsymbol{\phi}_n^{n+1}}{\partial \vec{n}} \right) \tag{18}$$

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 18 is the volume weighted

n	L^1 Error	Order	L^∞ Error	Order
32	1.57×10^{-4}	–	7.19×10^{-4}	–
64	3.77×10^{-5}	2.06	1.84×10^{-4}	1.96
128	9.23×10^{-6}	2.03	3.91×10^{-5}	2.23
256	2.28×10^{-6}	2.02	1.05×10^{-5}	1.90
512	5.17×10^{-7}	2.14	2.81×10^{-6}	1.90

Table 6: Convergence results for solving a Poisson equation with analytic solution $\phi(x, y) = e^x + e^y$.

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

	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 8: 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.

form of Equation 15 in order to maintain symmetry. For second order accuracy we use a trapezoid rule time integration scheme by solving the following equation:

$$\left(\mathbf{V}_c - \frac{\Delta t}{2}\beta\mathbf{V}_c\mathbf{D}\mathbf{G}\right)\boldsymbol{\phi}^{n+1} = \left(\mathbf{V}_c + \frac{\Delta t}{2}\beta\mathbf{V}_c\mathbf{D}\mathbf{G}\right)\boldsymbol{\phi}^{n} + \Delta t\beta\mathbf{V}_c\mathbf{D}\left(\mathbf{G}_d\frac{\boldsymbol{\phi}_d^{n+1} + \boldsymbol{\phi}_d^{n}}{2} + \frac{\frac{\partial\boldsymbol{\phi}_n^{n+1}}{\partial\vec{n}} + \frac{\partial\boldsymbol{\phi}_n^{n}}{\partial\vec{n}}}{2}\right) \quad (19)$$

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 14 and shown in Figure 19. In all of our heat equation tests $\beta = .01$. 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 15 and 16 show the results for backward euler and trapezoid rule time integration on a stationary grid.

In order to treat moving grids we remap ϕ values at the beginning each of step using the semi-Lagrangian advection scheme 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 scheme we first need to define ϕ at every location on every grid. 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. 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

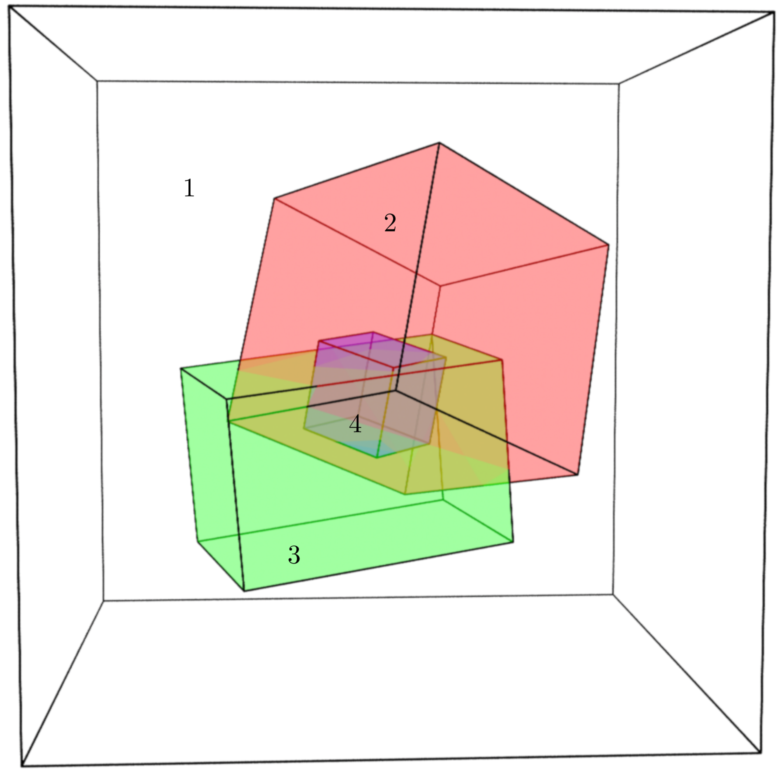


Figure 18: The domains of the grids used in our Poisson equation tests in three spatial dimensions.

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. We also note that the semi-Lagrangian Advection scheme causes the second order accurate version of the method in Equation 19 to degenerate to first order, but this can be alleviated by using the 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 17 and 18 show the results for backward euler and trapezoid rule time integration using semi-Lagrangian and SL-MacCormack remapping respectively. Finally, table 19 shows that using semi-Lagrangian remapping degenerates the trapezoid rule time integration scheme towards first order as compared to when using SL-MacCormack remapping.

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 20 and shown in Figure 20. 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 21 gives the errors and orders of accuracy when using trapezoid rule time integration and SL-MacCormack remapping.

5.1. Navier-Stokes Viscosity

For spatially constant viscosity, the Navier-Stokes equations dictate solving a heat equation independently in each orthogonal direction. 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

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 9: 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 10: Convergence results for solving a Poisson equation with analytic solution $\phi(x, y, z) = e^{-x^2-y^2-z^2}$.

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 heat 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 once again consider the moving grids in Table 14, Figure 19, Table 20 and Figure 20. 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 22 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 $\vec{\phi}(x, y, z, t) = (e^{-.03\pi^2 t} sin(\pi x)sin(\pi y)sin(\pi z), e^{-.12\pi^2 t} sin(2\pi x)sin(2\pi y)sin(2\pi z), e^{-.14\pi^2 t} sin(\pi x)sin(2\pi y)sin(3\pi z))$. Table 23 gives the errors and orders of accuracy when using trapezoid time integration and SL-MacCormack remapping.

5.2. Spatially Varying Navier-Stokes Viscosity

In the case of spatially varying viscosity, the diffusion equations are coupled across spatial dimensions making the discretization more complex. In fact, it can be quite difficult to implement a fully implicit monolithic method since one replaces the individual heat equations with a fully coupled one which stacks and mixes the velocities from all spatial directions. One approach to simplify this was considered in [61] where the coupling terms were treated explicitly in order to separate the solve into three separate heat equations. Although we do not consider spatially varying viscosity in this paper, we briefly consider a fully coupled solve along the lines of [65] 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).

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

	$\phi(x, y) = \sin(\pi x)\sin(\pi y)$		$\phi(x, y) = e^{-x^2-y^2}$	
Significant Figures	CG	ICPCG	CG	ICPCG
2	6656	372	8063	410
4	19016	873	19798	1078
8	43748	1869	43347	2297

Table 12: The number of iterations taken by CG and ICPCG in order to converge to the specified number of significant figures for two Poisson equation tests in two spatial dimensions. In these tests we used $n = 128$ so that ϕ was solved for on 1.5×10^5 cells.

	$\phi(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$		$\phi(x, y, z) = e^{-x^2-y^2-z^2}$	
Significant Figures	CG	ICPCG	CG	ICPCG
2	1606	158	3596	132
4	6769	345	8477	310
8	17056	717	18199	687

Table 13: The number of iterations taken by CG and ICPCG in order to converge to the specified number of significant figures for two Poisson equation tests in three spatial dimensions. In these tests we used $n = 128$ so that ϕ was solved for on 8.6×10^6 cells.

Since the following exercise is only done for the purpose of illustration and we do not consider the fully coupled case in the rest of the paper, we make several simplifications. We first consider solving on the Voronoi mesh only, and do not map back and forth from the Cartesian grids. As a consequence we only consider the case where the grids are stationary. We also only consider applying Dirichlet boundary conditions on the exterior of the domain in order to simplify the mapping of these 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 Cartesian grid direction 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 $\mathbf{W}$:

$$\left(\mathbf{V}_f - \Delta t\beta\mathbf{W}^T\begin{pmatrix}\mathbf{V}_c\mathbf{DG} & \mathbf{0}\\ \mathbf{0} & \mathbf{V}_c\mathbf{DG}\end{pmatrix}\mathbf{W}\right)\boldsymbol{\phi}_f^{n+1} = \mathbf{V}_f\boldsymbol{\phi}_f^n$$
$$+\Delta t\beta\mathbf{W}^T\begin{pmatrix}\mathbf{V}_c\mathbf{DG} & \mathbf{0}\\ \mathbf{0} & \mathbf{V}_c\mathbf{DG}\end{pmatrix}\mathbf{W}_d\boldsymbol{\phi}_{f,d}^{n+1} + \Delta t\beta\mathbf{W}^T\begin{pmatrix}\mathbf{V}_c\mathbf{DG}_d\boldsymbol{\phi}_{x,d}^{n+1}\\ \mathbf{V}_c\mathbf{DG}_d\boldsymbol{\phi}_{y,d}^{n+1}\end{pmatrix} \tag{20}$$

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

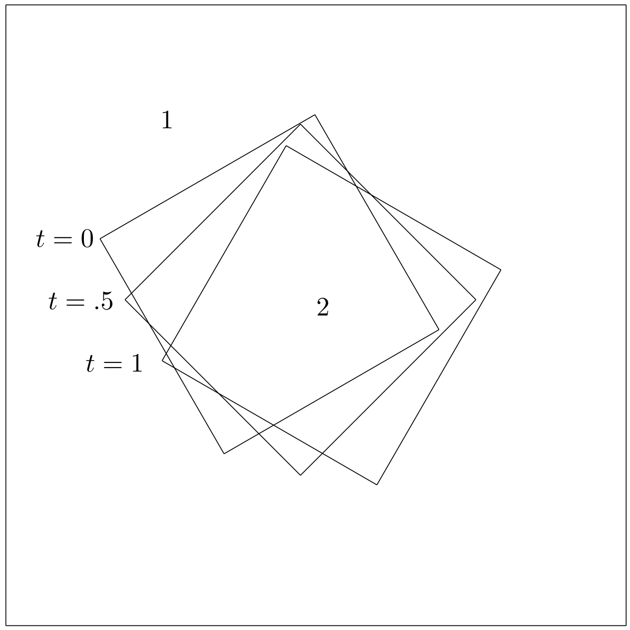


Figure 19: The domains of the grids used in our heat equation tests in 2 spacial dimensions with the second grid shown at its time $t = 0$, $t = .5$ and $t = 1$ positions and orientations.

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

Table 24 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 20, 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.

6. Incompressible Flow

We consider the incompressible Navier-Stokes equations as follows:

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\,\vec{u}\right) = -\nabla p + \mu\nabla^2\vec{u} \tag{21}$$

$$\nabla\cdot\vec{u} = 0 \tag{22}$$

where ρ is the density, $\vec{u}$ is the velocity, p is the pressure and μ is the viscosity. We split solution of Equations 21 and 22 into three steps. First we apply the SL-MacCormack advection scheme from Section 3.2 to obtain an intermediate velocity field, which is then augmented by the viscous terms using a second order accurate

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 15: Convergence results for solving a heat 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 16: Convergence results for solving a heat 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.

time discretization (Equation 19) independently in each Cartesian direction as described in Section 5.1. In order to compute the divergence of the post viscosity velocities for the right hand side of the pressure Poisson equation we interpolate a velocity from the Cartesian grids to 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 14:

$$-\mathbf{V}_c \mathbf{D} \mathbf{G} \hat{\mathbf{p}} = -\mathbf{V}_c \mathbf{D} \mathbf{u}^* + \mathbf{V}_c \mathbf{D} \left(\mathbf{G}_d \hat{\mathbf{p}}_d - \mathbf{u}_n^{n+1} \right) \tag{23}$$

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 Neumann boundary conditions. In order to update the velocities after solving Equation 23 we first update the velocities at all faces on the Voronoi mesh using pressure gradients computed by differencing the pressure samples at incident non-removed cells. 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 5.1. 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, 22]. 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. In the pressure solve we similarly set velocity Neumann boundary conditions at faces on the Voronoi mesh whose centers lie inside an object.

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 17: Convergence results for solving a heat 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 18: Convergence results for solving a heat 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.

6.1. 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 2.2. 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 2.2. 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.

When specifying boundary conditions, we define inflow boundary condition by specifying the velocity using Neumann boundary conditions in the pressure solve and Dirichlet boundary conditions for all components in the viscosity solve. Non-slip boundary conditions at structure interfaces are also handled in the exact same manner as further discussed in Section 6. Outflow boundary conditions are specified by setting zero pressure Dirichlet boundary conditions in the pressure solve and zero Neumann boundary conditions for all velocity components in the viscosity solve in order to prevent momentum from being exchanged across the boundary. Slip boundary condition are specified by setting Neumann boundary conditions in the pressure solve, and setting zero Neumann boundary conditions in the tangential components and Dirichlet boundary conditions in the normal component in the viscosity solve.

6.1.1. Two-dimensional lid driven cavity

We first consider a two-dimensional lid driven cavity and compare our results to those of [24]. 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 25 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 21 for Reynolds numbers 100, 400, 1000

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 19: Convergence results for solving a heat 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.

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

and 5000. Our method produces the same vortices as observed by [24]. The u direction velocities on the x medial plane and v direction velocities on the y medial plane are shown in Figures 22 and 23 respectively. Our results are very close to those from [24], 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.

6.1.2. 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 26, 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:

$$\vec{u}^0(\vec{x}) = (1, 0)^T + \frac{(x_c - y, x - y_c)^T}{|\vec{x} - \vec{x}_c|} \begin{cases} e^{\left(\frac{-.25}{|\vec{x}-\vec{x}_c|/r - |\vec{x}-\vec{x}_c|^2/r^2}\right)} & \text{if } \|\vec{x} - \vec{x}_c\| < r \\ 0 & \text{if } \|\vec{x} - \vec{x}_c\| \geq r \end{cases} \tag{24}$$

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 24 at time $t = .20833$. Note that no artifacts are observed at the grid boundaries when the fine grid is stationary or rotating. Figures 25 and 26 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 towards to zero implying self convergence and that the orders of accuracy tend towards second order as the grids are refined.

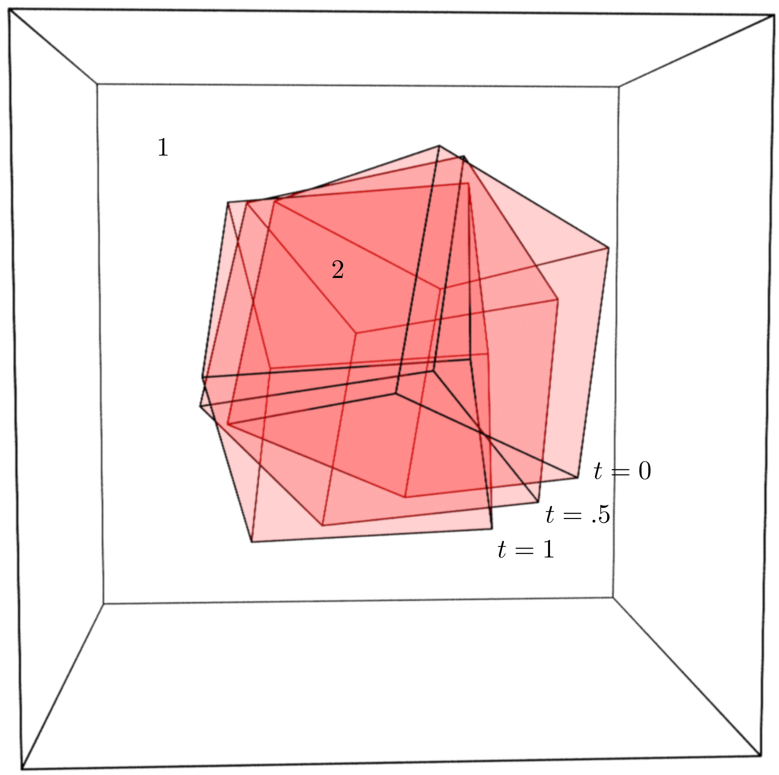


Figure 20: The domains of the grids used in our heat tests in three spatial dimensions with the second grid shown at its time $t = 0$, $t = .5$ and $t = 1$ positions and orientations.

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 21: Convergence results for solving a heat 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.

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 22: Convergence results for solving a separate heat 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

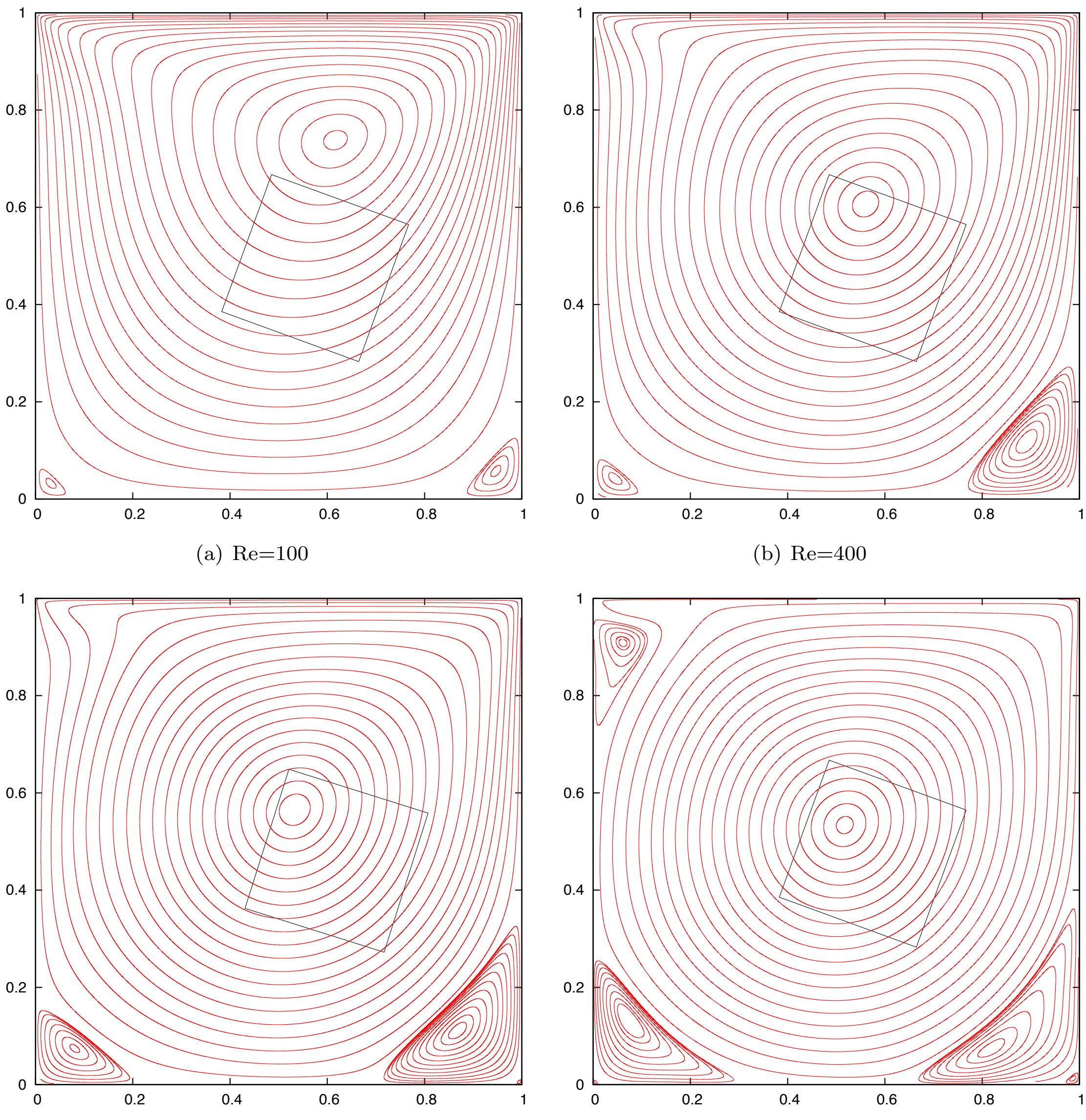
Table 23: Convergence results for solving a separate heat equation in each direction with analytic solution $\vec{\phi}(x, y, t) = (e^{-.03\pi^2 t} sin(\pi x) sin(\pi y) sin(\pi z), e^{-.12\pi^2 t} sin(2\pi x) sin(2\pi y) sin(2\pi z), e^{-.14\pi^2 t} sin(\pi x) sin(2\pi y) sin(3\pi z))$ on one stationary grid and one moving grid using SL-MacCormack remapping and trapezoid rule time integration.

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 24: Convergence results for solving a heat equation in each direction coupled in a monolithic system 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 two stationary grids using backward euler time integration.

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

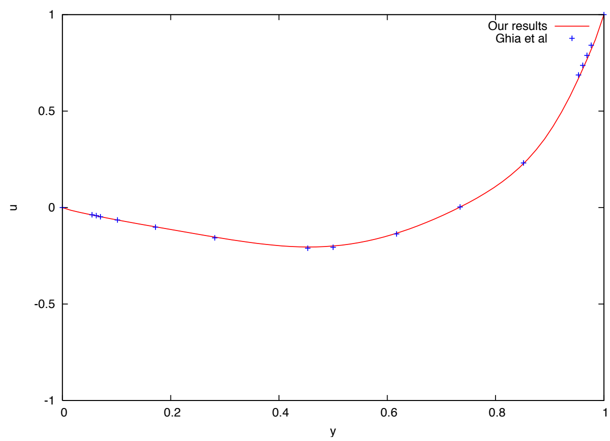
Table 25: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the two grids used in our lid driven cavity tests. n indicates the number of cells in each dimension on the coarsest grid. For Reynolds numbers 100 and 400, $n = 128$ was used. For Reynolds numbers 1000 and 5000, $n = 256$ was used. 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.



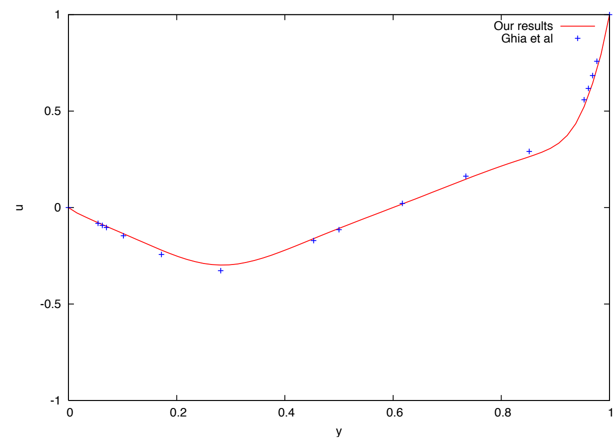
(a) Re=100 (b) Re=400

(c) Re=1000 (d) Re=5000

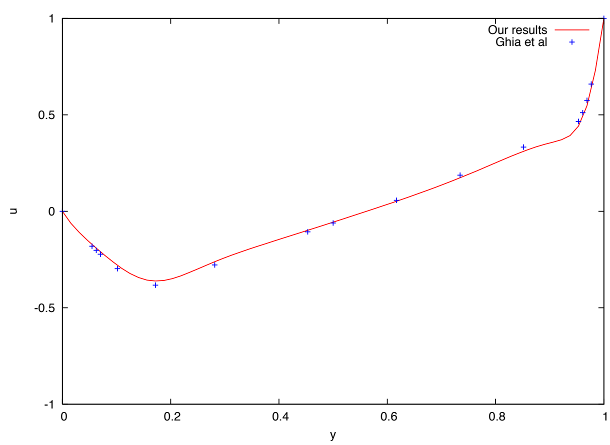
Figure 21: Streamlines for the lid driven cavity example. Notice the tiny vortices at the bottom right corners of the Reynolds number 1000 and 5000 simulations and at the bottom left corner of the Reynolds number 5000 simulation. These vortices are the same as reported by [24].



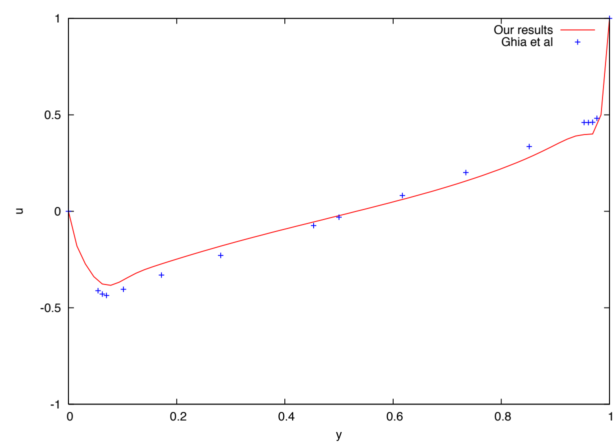
(a) Re=100



(b) Re=400

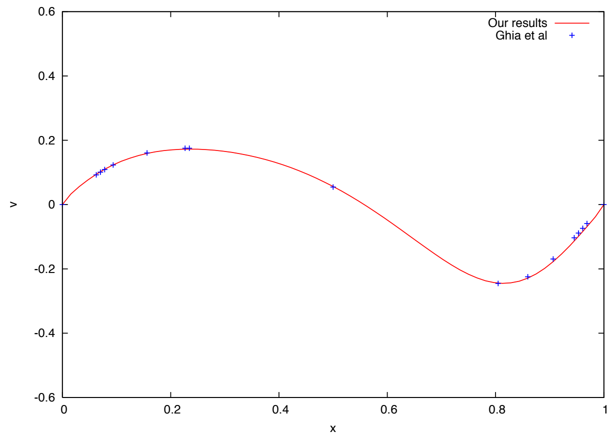


(c) Re=1000

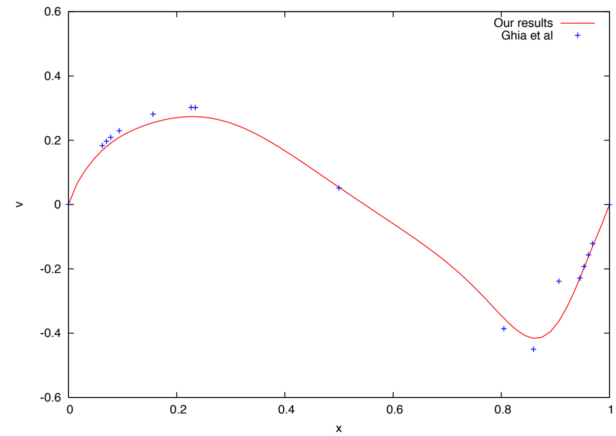


(d) Re=5000

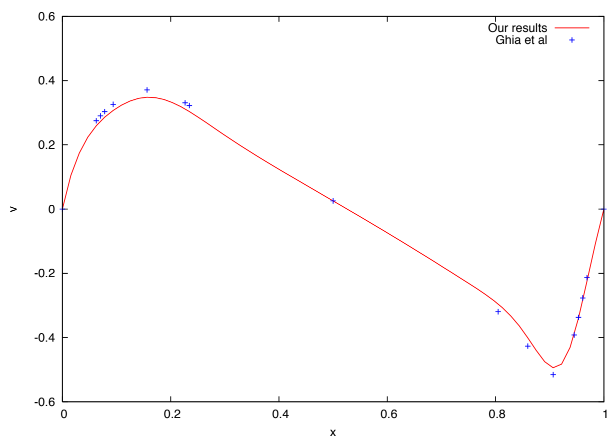
Figure 22: The u velocities on the vertical plane through the geometric center of cavity. The red lines correspond to our results and the blue '+' symbols correspond to the results from [24]. Note the good agreement between the results, particularly for smaller Reynolds numbers.



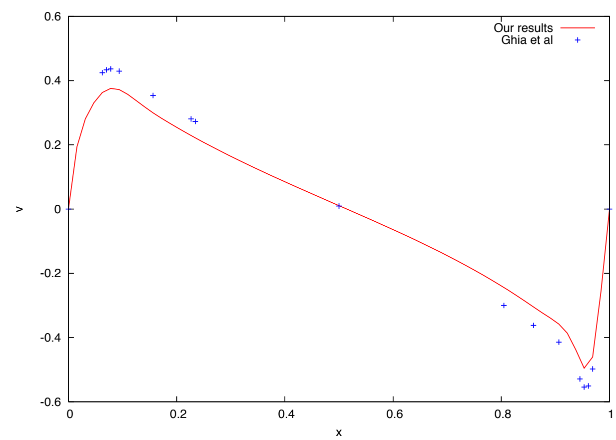
(a) Re=100



(b) Re=400



(c) Re=1000



(d) Re=5000

Figure 23: 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 [24]. Note the good agreement between the results, particularly for smaller Reynolds numbers.

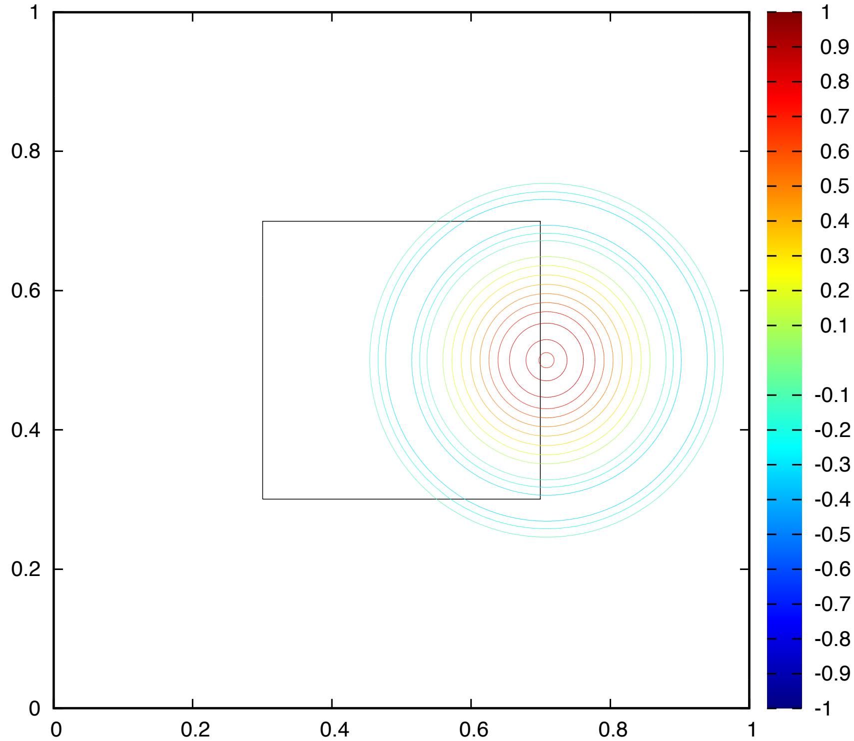
	Object space domain	Δx	$\vec{s}$	θ
1	$[0,1] \times [0,1]$	$1/n$	$(0,0)$	0
2	$[-.2,.2] \times [-.2,.2]$	$0.4/n$	$(.5,.5)$	$-t\pi/3$

Table 26: 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.

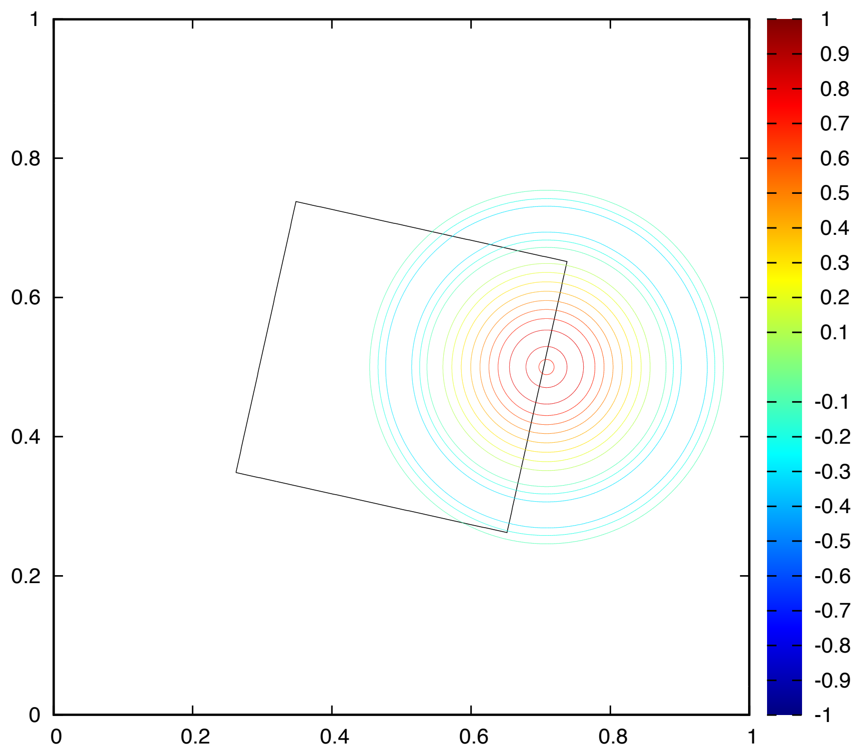
6.1.3. Two-dimensional flow past a stationary circular cylinder

We consider the two dimensional stationary circular cylinder example from [32] 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 27 and as illustrated in Figure 27, 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 is located at $(9.6, 12.8)$. 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 28 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 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 [32]. In particularly the Strouhal numbers are all in very close agreement with both the numerical and experimental results given in [32]. Similarly Table 29 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 [32] 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 28 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 [32]. 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.

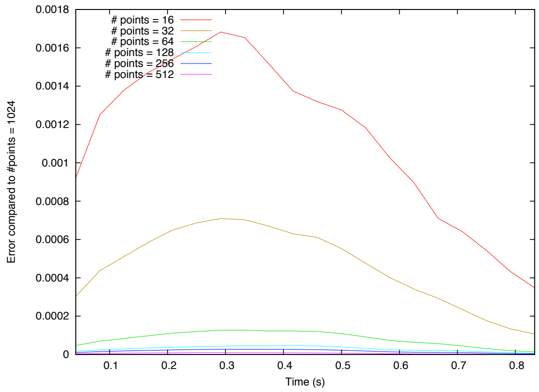


(a) Non-moving grid

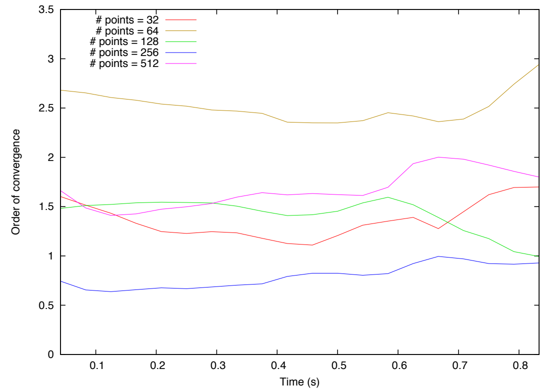


(b) Moving grid

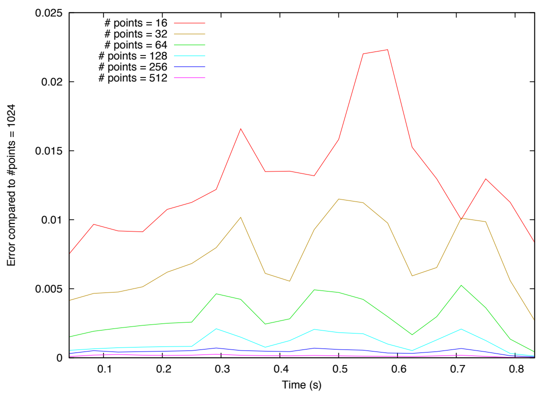
Figure 24: Vorticity isocontours for the vortex flow across grid boundary example at t = .20833. Note that the isocontours match at the grid boundaries and no artifacts are visible along or further away from the boundaries.



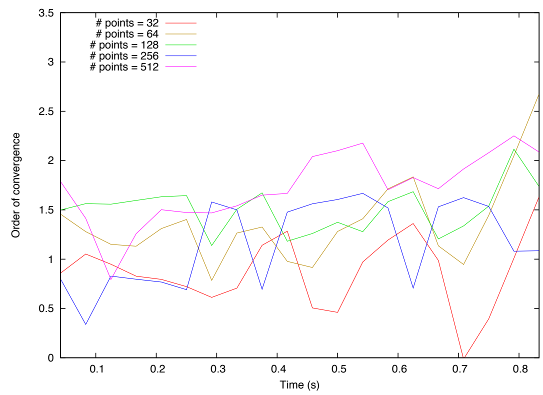
(a) Error in L^1 norm



(b) Order in L^1 norm

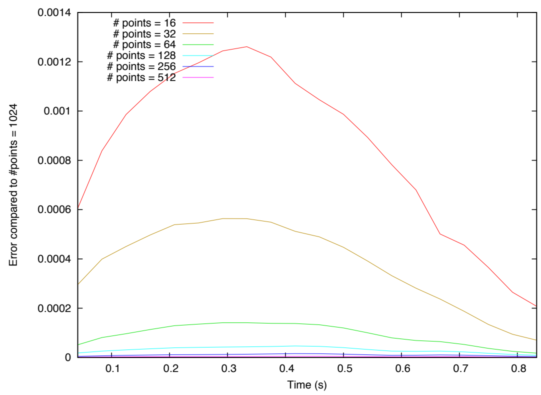


(c) Error in L^∞ norm

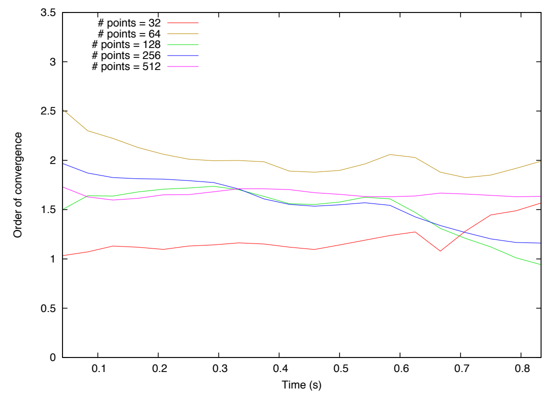


(d) Order in L^∞ norm

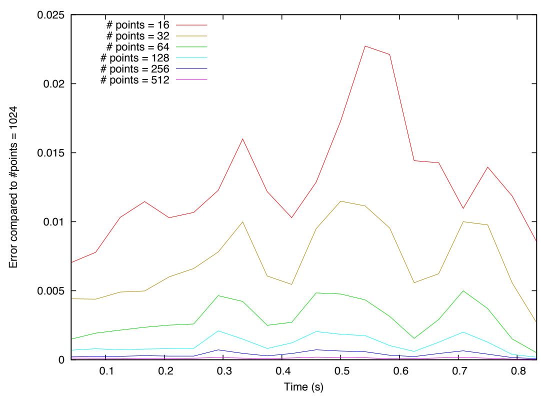
Figure 25: The errors and orders of accuracy of velocities for the vortex flow across grid boundary example, with a stationary fine grid. Figures (a) and (c) show that the error tends towards zero in both the L^1 and L^∞ norms implying self convergence, whereas Figures (b) and (d) show that the error seems to be improving towards second order accuracy as the grid is refined.



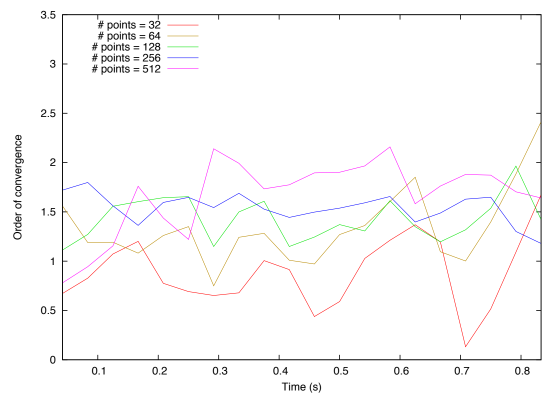
(a) Error in L^1 norm



(b) Order in L^1 norm



(c) Error in L^∞ norm



(d) Order in L^∞ norm

Figure 26: 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 second order accuracy as the grid is refined.

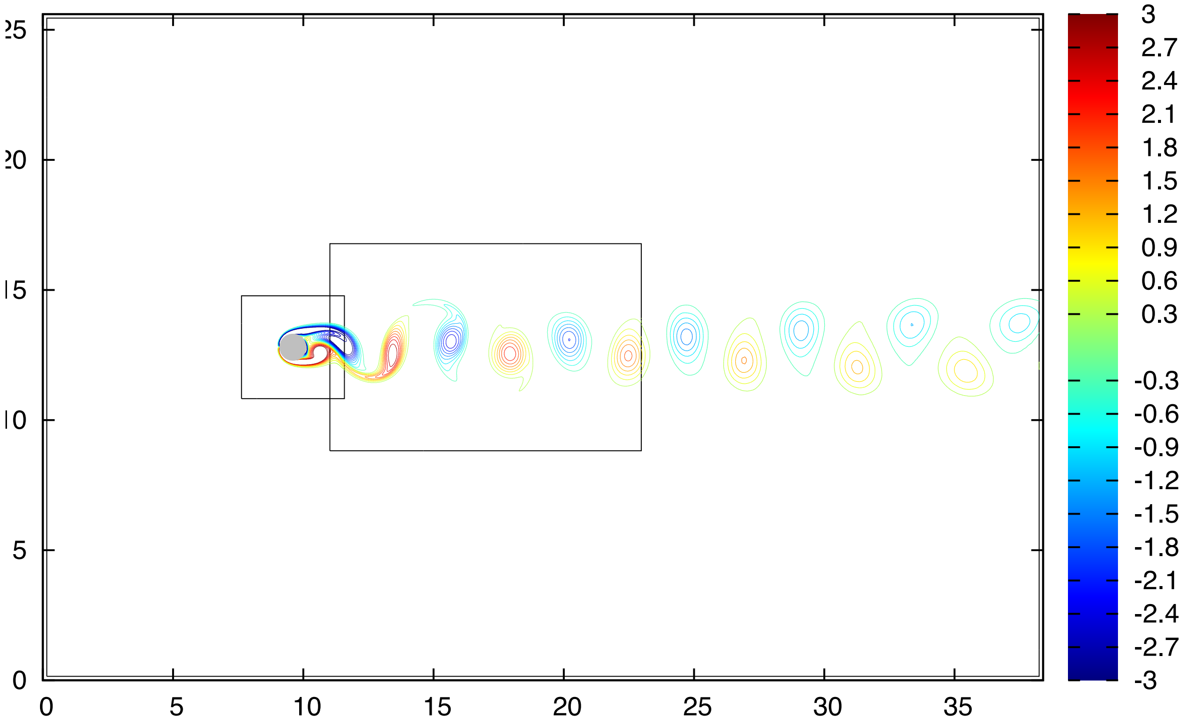


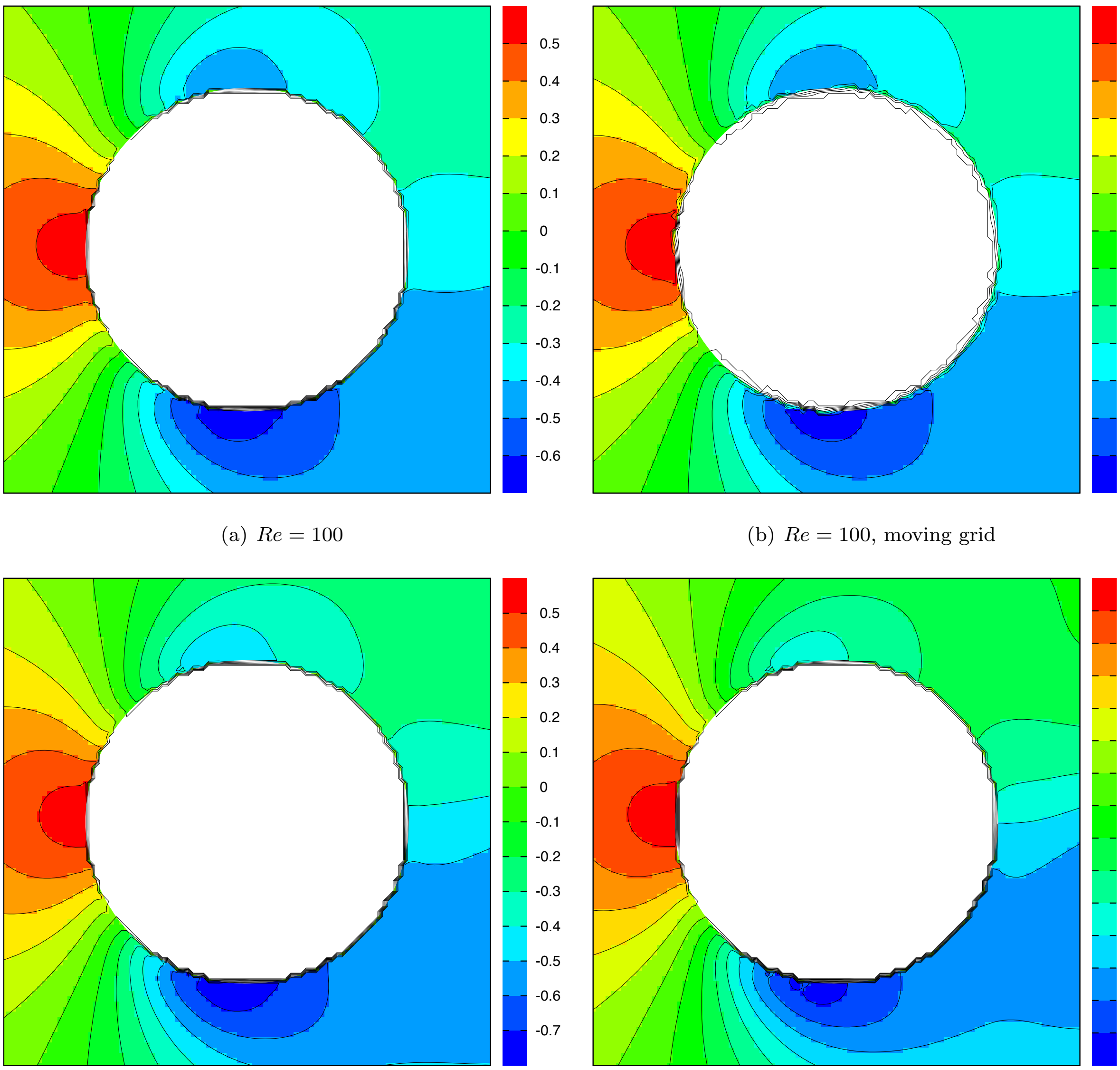
Figure 27: Vorticity isocontours for the flow past a circular cylinder example with Reynolds number 200.

	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	$[11, 23] \times [8.8, 16.8]$	$4/n$	$(0, 0)$	0

Table 27: 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. In all tests in this example we used $n = 256$. During parallel simulation each of the three grids remained unsubdivided and was allocated a single MPI process since the number of cells on each grid were close enough to not warrant any subdivision.

6.1.4. 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 [32]. We discretize the domain $[0, 25.6] \times [0, 24]$ with three grids as listed in Table 30 and as shown in Figure 29(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 $(9.6, 12)$ with its long axis along the x axis in object space. The cylinder rotates with angular velocity $\pi/4$ which is matched by the enclosing fine grid. For Reynolds number 200, Figure 30 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



(a) $Re = 100$

(b) $Re = 100$, moving grid

(c) $Re = 150$

(d) $Re = 200$

Figure 28: 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 [32] 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	$1.3433 \pm .0106$	$\pm .3291$	.1675
150	$1.3310 \pm .0261$	$\pm .5098$	.1852
200	$1.3459 \pm .0447$	$\pm .6631$	.1964

Table 28: 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 27. Note the good agreement of all values with those produced and cited by [32].

Reynolds Number	C_D	C_L	St
100	$1.3466 \pm .0152$	$\pm .3404$	.1675

Table 29: 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 27. Note that the values are close to those for the stationary case as listed in Table 28.

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 6.1.2 in order to examine the behavior of the errors when they are not dominated by those in the structure boundary layer. The vorticity contours are plotted in Figure 29 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 30: 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.

6.1.5. Two-dimensional flow past multiple rotating elliptic cylinders

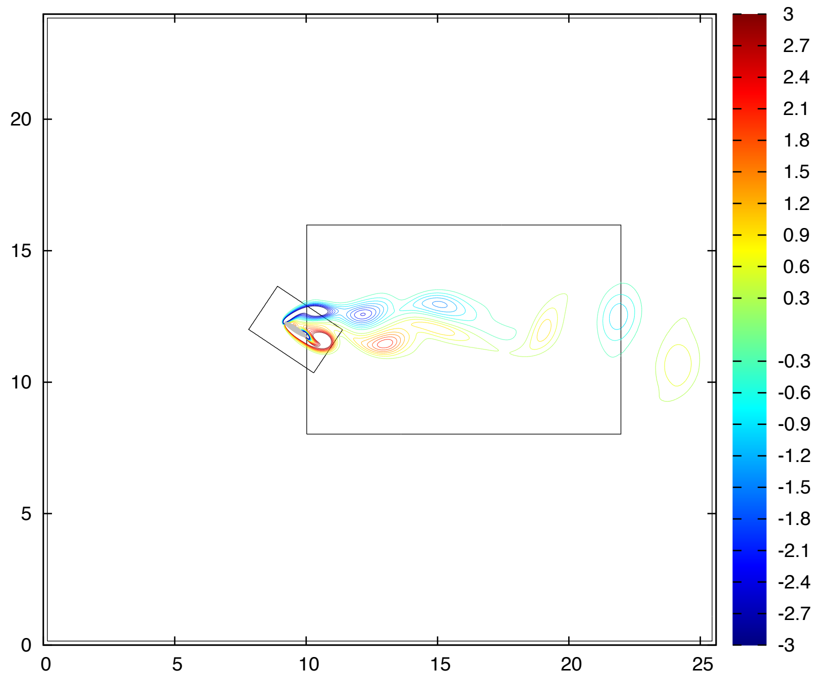
In our final two-dimensional example we consider three rotating elliptic cylinders in order to demonstrate our method on a more complex example. We discretize the domain $[0, 9] \times [0, 6]$ with six grids as listed in Table 31 and shown in Figure 31(a), where grids 3, 4 and 5 each enclose and move with an elliptic cylinder as listed in Table 32. Note that unlike previous examples the grids in this case were subdivided for parallel computation also as described in the caption of Table 31. 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 31 and 32 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.

	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

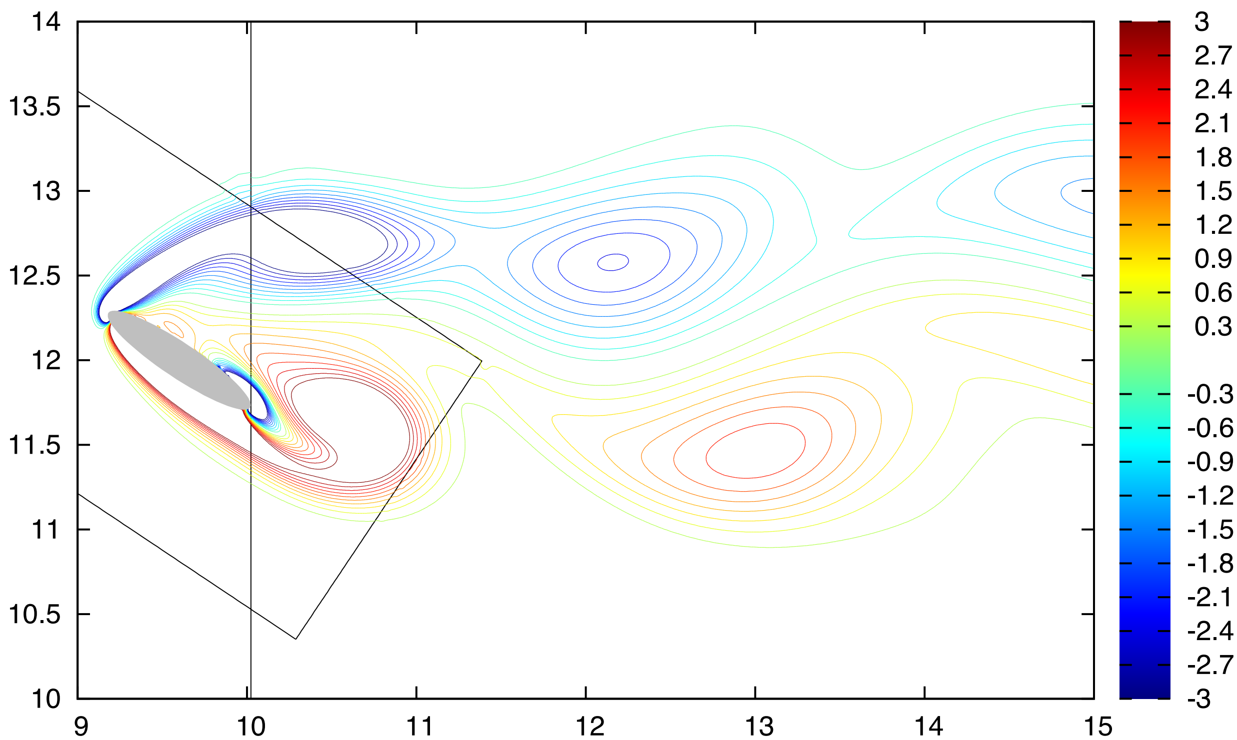
Table 31: Domains, cell sizes, positions ($\vec{s}$) and orientations (angle θ) of the six grids used in our flow past three rotating elliptic cylinders simulation where $n = 256$. Grids 3, 4 and 5 each enclose and track one of the elliptical cylinders as shown in Figures 31 and 32. During parallel simulation, 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)	$\vec{s}$	θ
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 32: 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.

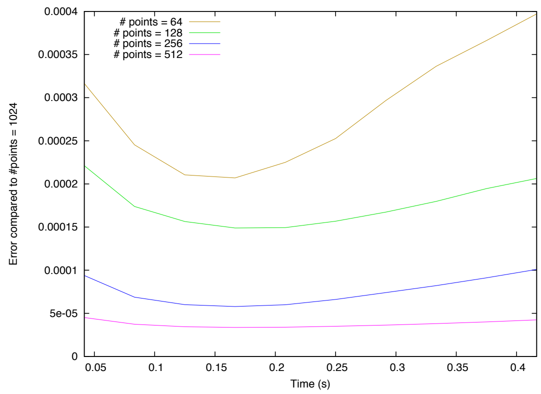


(a) Vorticity, entire domain

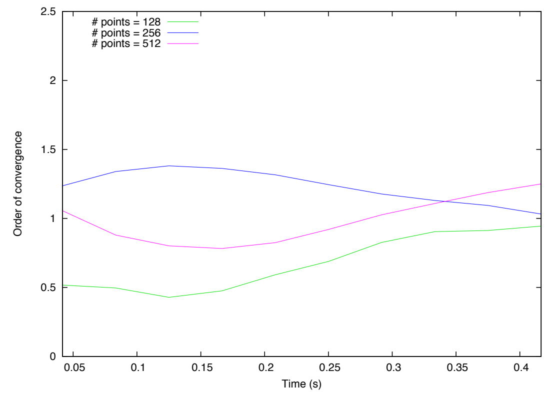


(b) Vorticity, close-up

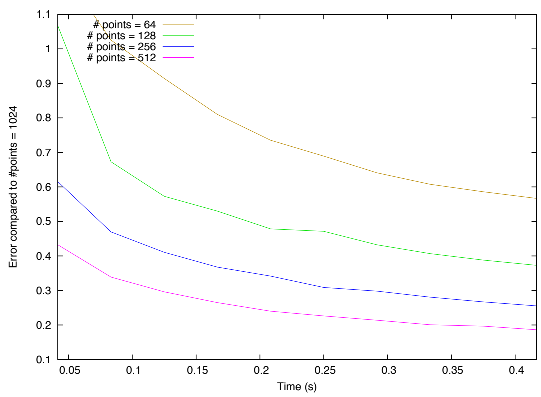
Figure 29: Vorticity isocontours for the flow past an elliptic cylinder example. Note that the elliptical cylinder and the grid attached to it are rotating with angular velocity $\pi/4$.



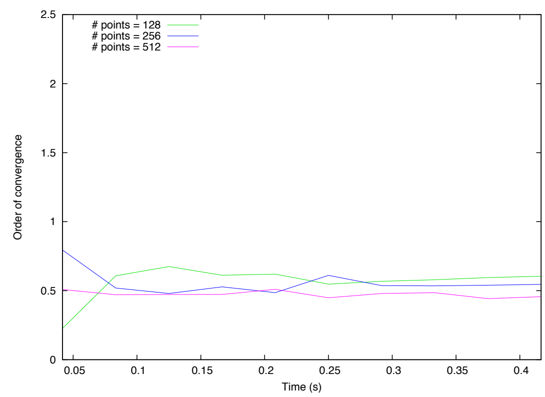
(a) Error in L^1 norm



(b) Order in L^1 norm

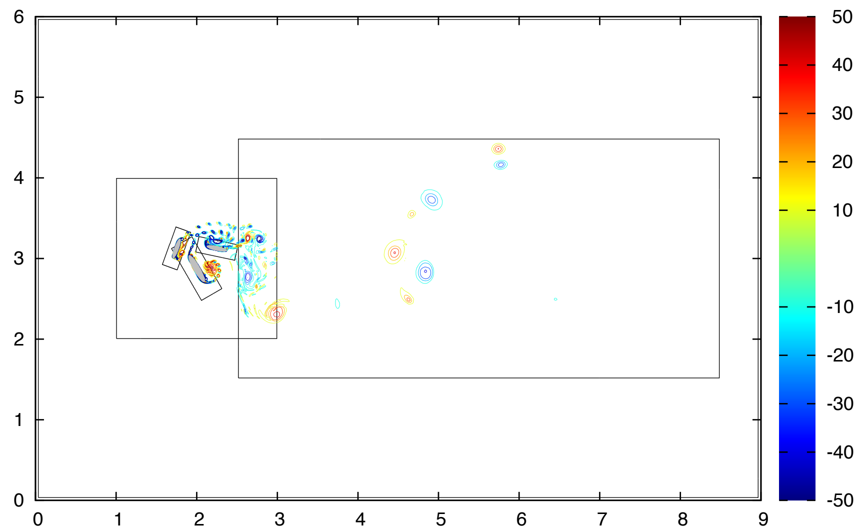


(c) Error in L^∞ norm

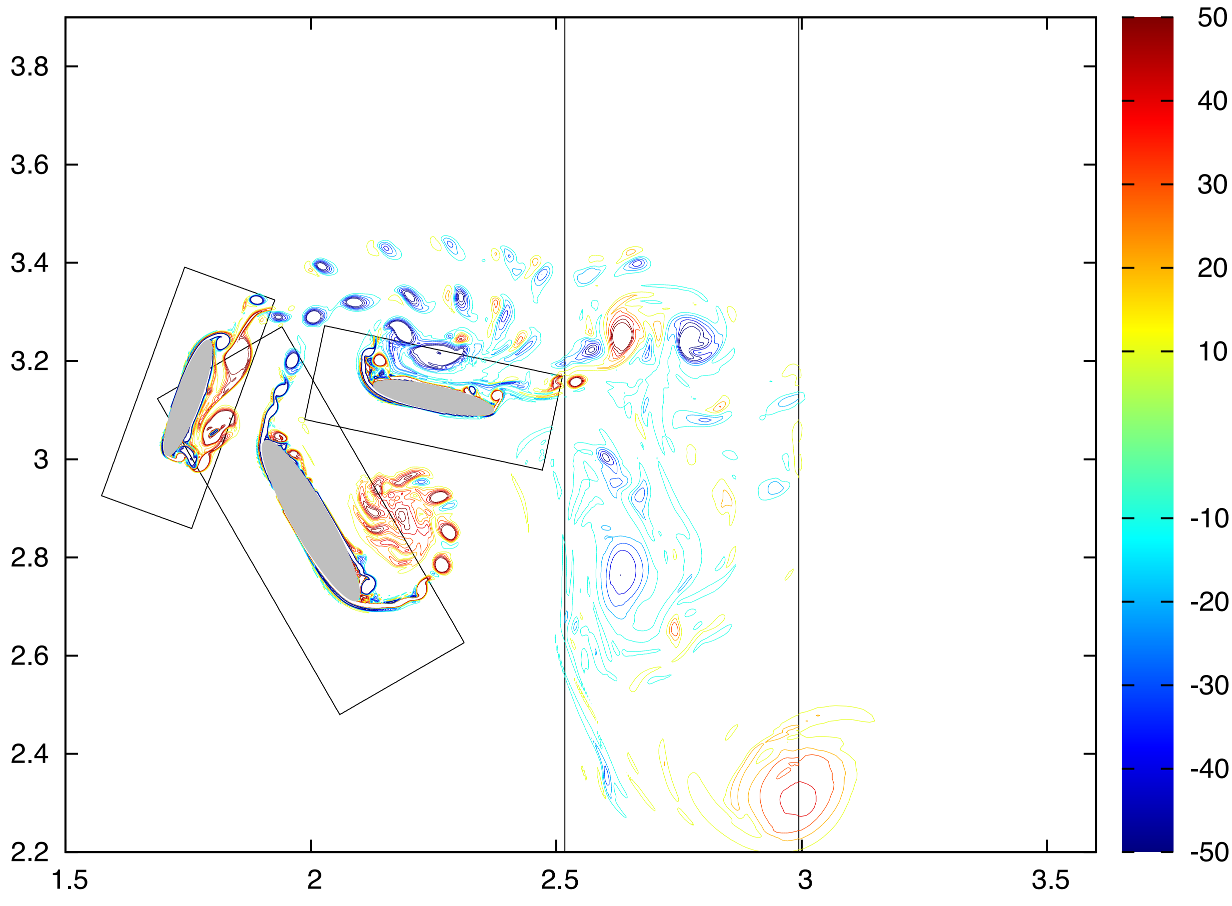


(d) Order in L^∞ norm

Figure 30: 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.

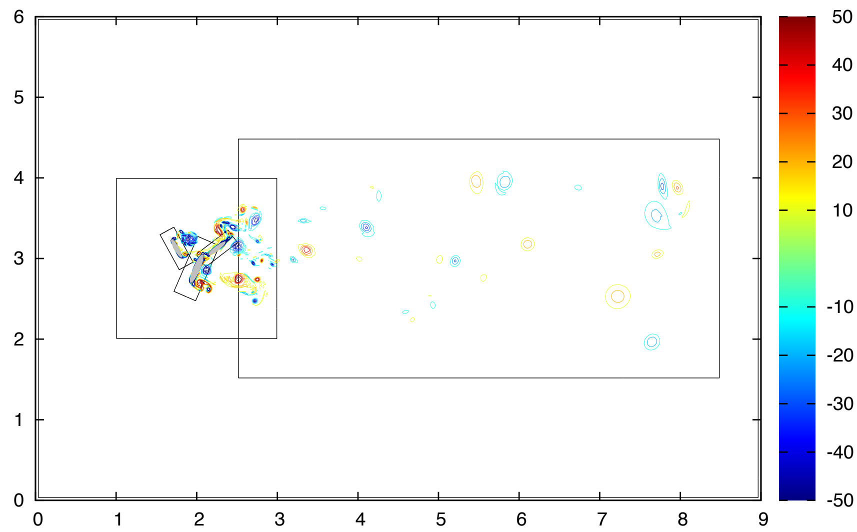


(a) Vorticity, entire domain

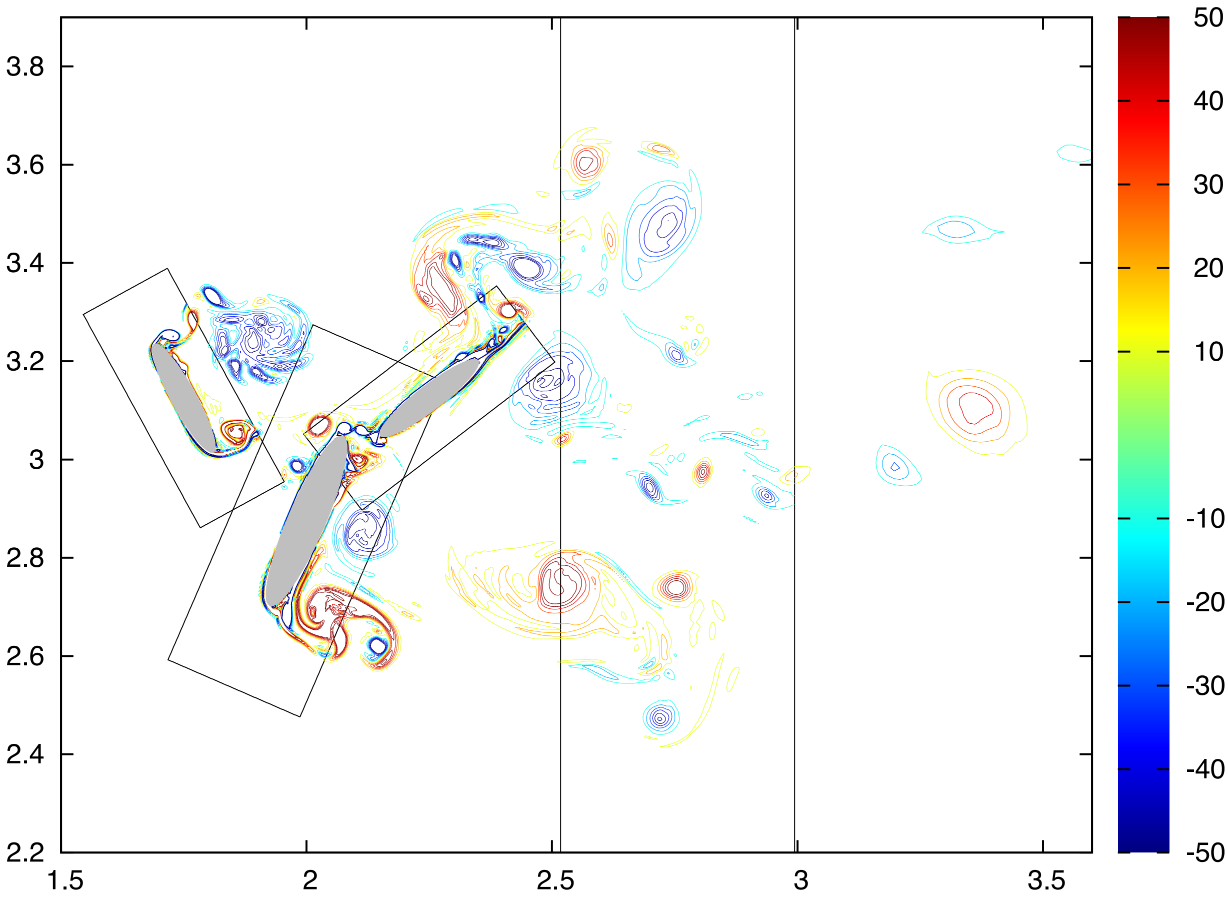


(b) Vorticity, close-up

Figure 31: Vorticity isocontours at time $t = 5.3333$ for the flow past multiple elliptic cylinders example.



(a) Vorticity, entire domain



(b) Vorticity, close-up

Figure 32: Vorticity at time $t = 18.958$ for the flow past multiple elliptic cylinders example.

6.1.6. 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 33 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 $(1, 3, 3)$ and the orientation is specified by a rotation of $t\sqrt{2}\pi/6$ radians about the axis $(1/\sqrt{2}, 0, 1/\sqrt{2})$. 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:

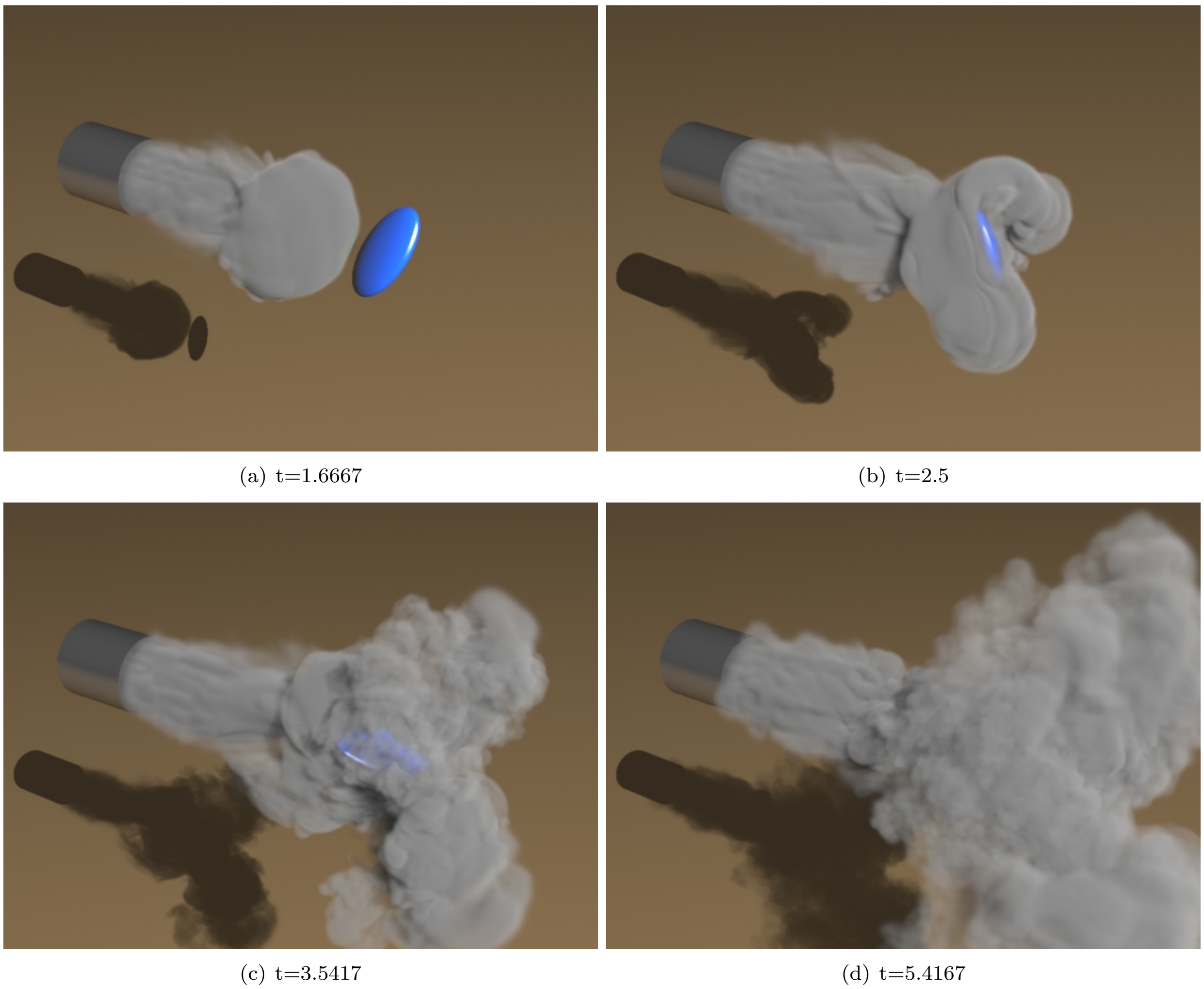
$$u(0, y, z) = \begin{cases} 1 & \text{if } |(y, z) - (3, 3)| \leq .09 \\ 1 - (|(y, z) - (3, 3)| - .09)/.03)) & \text{if } .09 < |(y, z) - (3, 3)| < .12 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

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 25 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 [66]. In Figure 33 we show the results for various times near the beginning of the simulation. Figure 34 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 33: 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 33: The passive scalar rendered as smoke for the three dimensional smoke jet past rotating ellipsoid example.

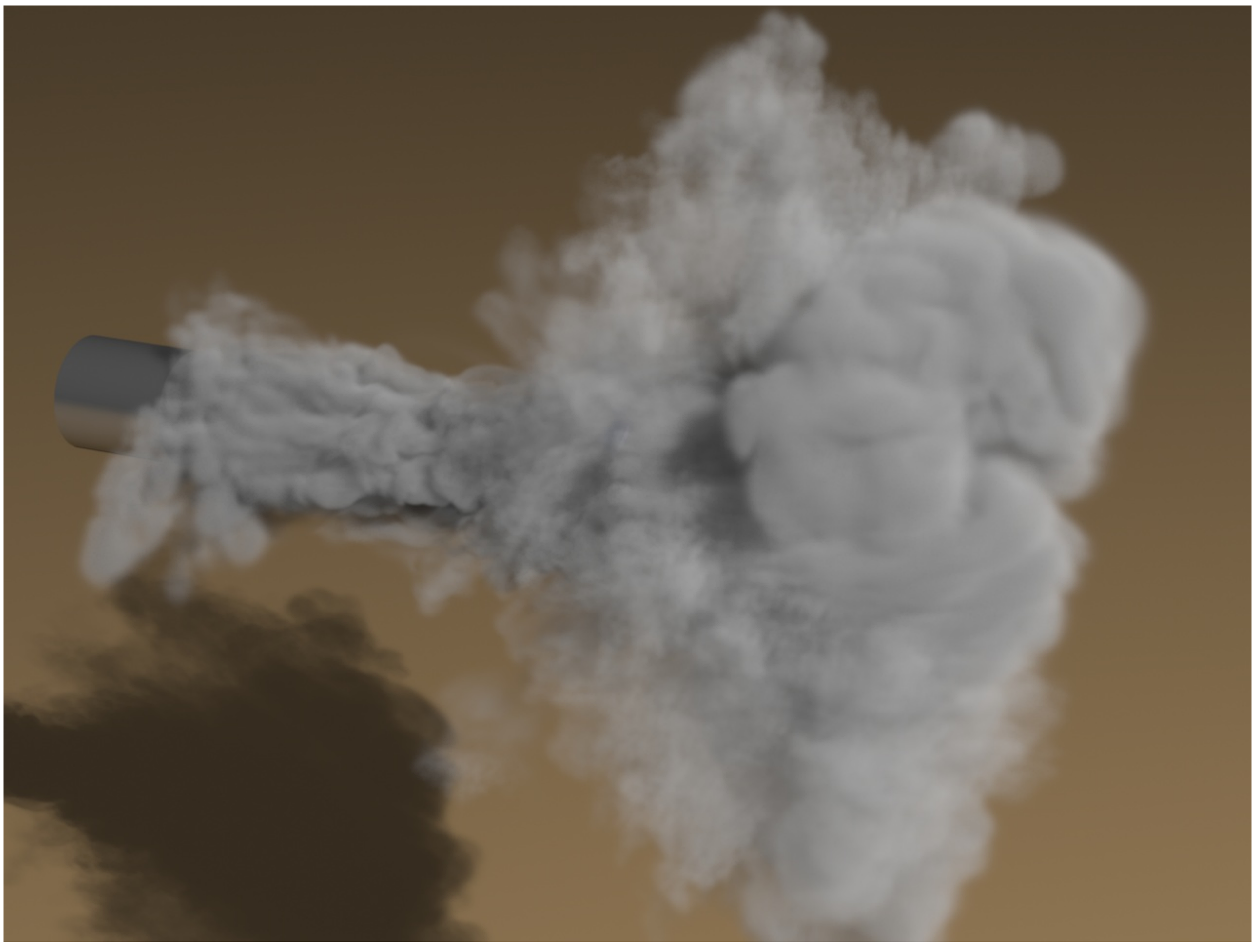


Figure 34: The passive scalar rendered as smoke at $t = 9.8333$ for the three dimensional smoke jet past rotating ellipsoid example.

7. Conclusion

We have introduced a new adaptive scheme for simulating the incompressible Navier-Stokes equations using a Chimera grid approach 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 heat equations on cells 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.

There are numerous avenues for future research. While we briefly addressed the problem of finding a more accurate and faster method for solving for viscous forces directly on the original face degrees of freedom, we believe a second order accurate treatment is important in order to allow for efficient solutions in the variable viscosity case. We also 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. This mapping also did not guarantee zero divergence at Cartesian cells with interpolated faces. Conservative advection (see e.g. [40, 26]) also poses interesting issues since one would need to account for the duplication of values in overlapped regions in order to guarantee conservation.

8. Acknowledgements

Research supported in part by ONR N00014-09-1-0101, ONR N-00014-11-1-0027, ONR N00014-11-1-0707, ARL AHPCRC W911NF-07-0027, and the Intel Science and Technology Center for Visual Computing. Computing resources were provided in part by ONR N00014-05-1-0479. R.E.E. was supported in part by a Stanford Graduate Fellowship and an NSERC Postgraduate Scholarship. Y.Y. was supported in part by a Stanford Graduate Fellowship.

9. 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] 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.

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

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] T. Brochu, C. Batty, and R. Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.*, 2010.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

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

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

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

[23] 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.

[24] 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.

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

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

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

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

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

[36] 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.

[37] 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.

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

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

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

[41] 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.

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

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] C. Y. Perng and R. L. Street. A coupled multigrid-domain-splitting technique for simulating incompressible flows in geometrically complex domains. *Int. J. Num. Meth. Fluids*, 13(3):269–286, 1991.

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

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

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

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

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

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

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

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

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

[75] 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.

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