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

Linhai Qiu*, Wenlong Lu*, Ronald Fedkiw*

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

Abstract

We present a novel method for simulating compressible flow on a multitude of Cartesian grids that can rotate and translate. Following previous work, we split the time integration into an explicit step for advection followed by an implicit solve for the pressure. A second order accurate flux based scheme is devised to handle advection on each moving Cartesian grid using an effective characteristic velocity that accounts for the grid motion. In order to avoid the stringent time step restriction imposed by very fine grids, we propose strategies that allow for a fluid velocity CFL number larger than 1. The stringent time step restriction related to the sound speed is alleviated by formulating an implicit linear system in order to find a pressure consistent with the equation of state. This implicit linear system crosses overlapping Cartesian grid boundaries by utilizing local Voronoi meshes to connect the various degrees of freedom obtaining a symmetric positive-definite system. Since a straightforward application of this technique contains an inherent central differencing which can result in spurious oscillations, we introduce a new high order diffusion term similar in spirit to ENO-LLF but solved for implicitly in order to avoid any associated time step restrictions. The method is conservative on each grid, as well as globally conservative on the background grid that contains all other grids. Moreover, a conservative interpolation operator is devised for conservatively remapping values in order to keep them consistent across different overlapping grids. Additionally, the method is extended to handle two-way solid fluid coupling in a monolithic fashion including cases (in the appendix) where solids in close proximity do not properly allow for grid based degrees of freedom in between them.

1. Introduction

Numerical simulation of compressible flow is important in many areas of science and engineering such as aircraft design, the study of underbody blasts, etc. In many of these problems, adding adaptivity can greatly increase the efficiency. For example, when simulating highly nonlinear compressible flows with shocks, contacts, and rarefactions, allocating more degrees of freedom around shocks and contacts while allocating fewer in smooth regions of the flow can significantly reduce the computational time while achieving the same or greater accuracy. Moreover, in solid fluid coupling problems, it is desirable to have higher resolutions near the solid fluid interfaces in order to resolve both the boundary layers and the solid fluid coupling physics.

There are a wide variety of methods for adaptive discretization. Unstructured methods, such as tetrahedral meshes (see e.g. [36, 35, 49]), especially allow for flexible adaptivity. However, one needs to carefully monitor for the presence of poorly conditioned elements as they can often produce inaccurate simulation results. Conditioning of the elements can be controlled to some degree via dynamic remeshing such as that used in Arbitrary Lagrangian Eulerian (ALE) schemes, see e.g. [30, 17]. However, the computational cost of remeshing can be large, and the dissipation due to remapping can be significant. Moreover, it can be challenging to store and access unstructured data in a cache coherent fashion. In parallel computing environments, finding and maintaining a domain decomposition that evenly distributes the workload can be costly, which is exacerbated if frequent remeshing and repartitioning are necessary. Both the issues with

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

element conditioning and cache coherency can be addressed with the use of structured grids, which also lend themselves to simple and accurate discretizations. Octrees (see e.g. [44, 43, 11, 50, 56]) and adaptive mesh refinement (AMR) (see e.g. [8, 7, 41, 2, 67, 48, 12, 53, 60, 34]) are two examples of adding adaptivity to structured grids. Unfortunately, octrees and similar hierarchical structures suffer from issues similar to unstructured meshes such as cache incoherency and costly domain decomposition. AMR methods improve upon this allowing for multiple Cartesian grids to be patched upon one another, as long as the number of patches and repatching frequencies are small. However, that premise is often not true as typical AMR methods (starting from [7]) constrain grid patches to be axis-aligned greatly increasing the number of patches required exacerbating issues with cache coherency and domain decomposition. With so many small patches, AMR methods are more akin to unstructured grids with respect to parallelization and scalability. Allowing these AMR grid patches to rotate, as in the original AMR method [8] or Chimera grid methods (see e.g. [6, 66, 4, 5, 65, 29, 26, 27, 54, 55, 28]), allows one to resolve interesting features with far fewer degrees of freedom. In fact, resolving a curved interface using AMR is heuristically similar to using a SLIC method [51, 31], while using rotated Chimera grids is heuristically similar to a PLIC method [13, 72, 73]. In this fashion AMR methods give a piecewise constant geometric representation of an interface while Chimera grids give a piecewise linear representation. Moreover, unlike AMR methods which need to have the coarse grid lines match up with the fine grid lines along patch boundaries, Chimera grids do not have this requirement and can even move in order to for example follow the motion of moving solids, which is much more efficient for resolving the boundary layers as pointed out by [18]. In this paper, we follow the Chimera grid framework of [14] which uses a multitude of Cartesian grids that can each rotate and translate.

Adaptive discretizations readily introduce small cells which can impose prohibitive time step restrictions. This is further exacerbated when simulating compressible flow using explicit schemes, see e.g. [63, 64, 33], where the sound speed makes time step restriction even more stringent. Thus, we follow the semi-implicit formulation proposed by [38] that splits the time integration into an explicit step for advection followed by an implicit solve for the pressure, thus avoiding any time step restrictions related to the sound speeds. In addition, [38] allows one to formulate two-way solid fluid coupling in a monolithic fashion that is stable for arbitrarily large density-to-mass ratios without introducing any new time step restrictions as shown in [24, 23]. Moreover, it also allows one to monolithically couple compressible flow to incompressible flow as shown in [1] and to robustly treat compressible multiphase flow as shown in [32]. In order to solve for the pressure implicitly on Chimera grids, one needs to globally couple the degrees of freedom across different grids. We follow [14] constructing local Voronoi meshes along intergrid boundaries to obtain a contiguous Voronoi mesh resulting in a symmetric positive-definite linear system that can be solved using the preconditioned conjugate gradient method. In order to maintain global conservation on the background grid and local conservation on each Cartesian grid, we interpolate the resulting pressure from the Voronoi mesh to the individual Cartesian grids so that pressure fluxes can be computed on all Cartesian grid faces enabling conservative updates for momentum and energy. Additionally, we extend the solid fluid coupling method of [24, 23] to Chimera grids in Section 9 making use of similar implicit discretizations on the Voronoi mesh. Since the implicit discretization of the pressure terms contains inherent central differencing which may lead to spurious oscillations, we propose a novel method for adding high order diffusion similar in spirit to ENO-LLF. The high order diffusion terms are discretized implicitly again utilizing the Voronoi mesh in order to avoid any time step restrictions, and then interpolated back to the Cartesian grids to be applied in a conservative manner via fluxes.

Advection is performed on each Cartesian grid for each component of the conserved variables independently using a second order accurate flux based advection scheme which handles the advection and grid motion using an effective object space characteristic velocity and thus does not require any extra remapping that may lead to undesired artificial dissipation. In order to alleviate the stringent time step restriction imposed by the small cells due to the fluid velocity, we propose strategies that allow the fluid velocity CFL number to be set larger than 1. Stability is achieved in one of the two ways, either by subdividing the time step during advection before solving for the pressure using the original full time step or by utilizing an unconditionally stable conservative semi-Lagrangian scheme [39] on finer grids. Updating each grid independently in advection, as well as in applying the aforementioned pressure and high order diffusion fluxes, potentially

creates inconsistent values in the regions overlapped by multiple grids. In order to enforce consistency while maintaining conservation on each Cartesian grid, we treat values interpolated from the Voronoi mesh as target values and devise a conservative interpolation operator that conservatively remaps the values in overlapped regions to be consistent with the target values. This conservative interpolation operator bounds the remapped values between the original values and the target values guaranteeing convergence while minimizing the creation of extrema. Moreover, it maintains global conservation on the background grid and local conservation on each Cartesian grid.

We demonstrate convergence of the method with a number of examples including the Sod shock tube, wind tunnel with a step, etc. We also show the behavior of our method in various other scenarios such as shocks crossing grid boundaries and grids following shocks. We also demonstrate convergence for examples that two-way couple compressible flow with rigid and deformable solids where finer grids follow the motion of the solids. Furthermore, in the appendix, we briefly comment on our efforts to extend the method of [58] to handle compressible gap flow.

2. Equations

Consider the multidimensional Euler equations,

$$\begin{pmatrix} \rho \\ \rho\vec{u} \\ E \end{pmatrix}_t + \begin{pmatrix} \nabla \cdot \rho\vec{u} \\ \nabla \cdot (\rho\vec{u} \otimes \vec{u}) \\ \nabla \cdot (E\vec{u}) \end{pmatrix} + \begin{pmatrix} 0 \\ \nabla p \\ \nabla \cdot (p\vec{u}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{1}$$

with density ρ, velocity $\vec{u}$, total energy E, and pressure p. Here we split the flux terms into advection terms and pressure terms and follow the semi-implicit formulation of compressible flow proposed in [38].

We first update the advection terms of the fluxes to obtain intermediate values ρ^*, $(\rho\vec{u})^*$, and E^*. Since pressure does not affect the continuity equation, $\rho^{n+1} = \rho^*$. Treating pressure implicitly yields

$$(\rho\vec{u})^{n+1} = (\rho\vec{u})^* - \Delta t \nabla p^{n+1}, \tag{2}$$

$$E^{n+1} = E^* - \Delta t \nabla \cdot (p\vec{u})^{n+1}. \tag{3}$$

In order to solve for p^{n+1}, we discretize the pressure evolution equation [19],

$$p_t + \vec{u} \cdot \nabla p = -\rho c^2 \nabla \cdot \vec{u}, \tag{4}$$

fixing $\nabla \cdot \vec{u}$ to time t^{n+1} and ρc^2 to time t^n. Denoting an advected pressure as $p^a = p^n - \Delta t \vec{u} \cdot \nabla p$, Equation 4 can be discretized as

$$p^{n+1} = p_a - \Delta t \rho^n (c^n)^2 \nabla \cdot \vec{u}^{n+1}. \tag{5}$$

Dividing Equation 2 by ρ^{n+1} and taking the divergence of both sides yields

$$\nabla \cdot \vec{u}^{n+1} = \nabla \cdot \vec{u}^* - \Delta t \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right), \tag{6}$$

which can be substituted into Equation 5 and subsequently rearranged to obtain

$$p^{n+1} - \rho^n (c^n)^2 \Delta t^2 \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right) = p^a - \rho^n (c^n)^2 \Delta t \nabla \cdot \vec{u}^\star. \tag{7}$$

Following [23], we compute the advected pressure by using the equation of state directly on the advected flow field, i.e. $p^a = p(\rho^*, e^*)$, where e^* is the advected internal energy defined via $E^* = \rho^* e^* + \frac{||(\rho\vec{u})^*||^2}{2\rho^*}$. We use the ideal gas equation of state, so $p^a = (\gamma - 1)\rho^* e^*$ where $\gamma = 1.4$.

We note that computing the advected pressure p^a from ρ^* and e^* introduces an $O(\Delta t)$ error, because the advected pressure $p^a = p^n - \Delta t \vec{u} \cdot \nabla p$ does not include expansion due to $\nabla \cdot \vec{u}$, whereas ρ^*, $(\rho\vec{u})^*$, and

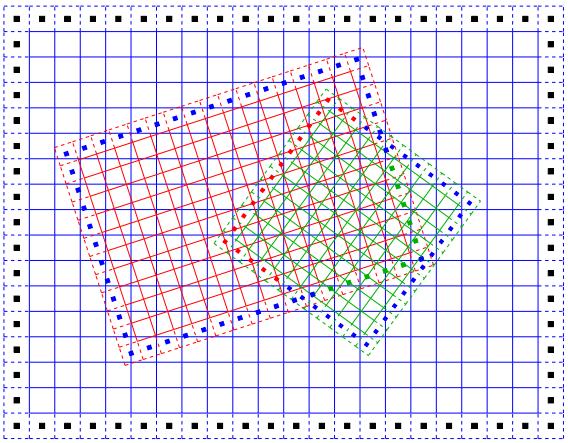


Figure 1: Illustration of filling ghost cells, marked with dots, surrounding each grid. For clarity, each grid is depicted with only one layer of ghost cells. The background grid's ghost cells are filled based on the boundary conditions for the computational domain. The red and green grids' ghost cells are filled with values interpolated from the finest overlapping grid with a valid interpolation stencil. The color of each dot indicates which grid its value is interpolated from.

E^* include this expansion. However, computing p^a from ρ^* and e^* not only leads to convergent results, but moreover seems to reduce spurious oscillations near the shock front as compared to using p^a computed from a Hamilton-Jacobi advection step. In fact, adding more expansion of the form $-\Delta t p^n \nabla \cdot \vec{u}^n$ to the calculation of p^a seems to reduce spurious oscillations even further. Similar results are obtained adding $-\Delta t \rho^n \nabla \cdot \vec{u}^n$, $-\Delta t (\rho\vec{u})^n \nabla \cdot \vec{u}^n$, and $-\Delta t E^n \nabla \cdot \vec{u}^n$ to the computation of ρ^*, $(\rho\vec{u})^*$, and E^*, respectively. The spurious oscillations may be related to the odd-even decoupling inherent in the central differencing applied during the implicit pressure update, and the additional expansion terms seem to alleviate such decoupling. We will see in Section 6 that adding high order diffusion also helps to alleviate these oscillations.

3. Chimera Grid Framework

Following previous work [14], our Chimera grid framework consists of a collection of overlapping Cartesian grids undergoing rigid motion. We describe a rigid transformation as the combination of a translation vector, $\vec{s}$, and a rotation matrix, $\mathbf{R}$. The relationship between positions and velocities in world space and a grid's object space is then given by

$$\vec{x}_{\text{world}} = \mathbf{R}\vec{x}_{\text{object}} + \vec{s} \tag{8}$$

$$\vec{u}_{\text{world}} = \mathbf{R}\vec{u}_{\text{object}} \tag{9}$$

Each Cartesian grid stores compressible flow variables ρ, $\rho\vec{u}$, and E at grid cell centers in standard fashion. Note that we store $\rho\vec{u}_{\text{world}}$ at each grid cell center as opposed to $\rho\vec{u}_{\text{object}}$. Although the representations are interchangeable via Equation 9, we have found storing $\rho\vec{u}_{\text{world}}$ more convenient for various purposes.

As discussed in [14], various explicit discretizations, such as a semi-Lagrangian advection scheme, only require filling a band of ghost cells surrounding the domain of each grid before performing operations on each grid independently. See Figure 1. For the background grid, we fill its ghost cells based on the boundary conditions for the computional domain. For other grids, we fill a ghost cell by interpolating its value from the finest grid that overlaps that cell enough to have a valid interpolation stencil.

Updating each grid independently creates potentially inconsistent values in regions overlapped by multiple grids, and these values may tend to drift even further apart over time. This can be rectified using a strategy similar to that for filling ghost cells. That is, one can simply replace the value on a coarser grid with an interpolated value from the finest grid that has a valid interpolation stencil for that point. See Figure 2.

Note that this has to be done in a hierarchical fashion, since a cell on one grid may have its value replaced by a finer grid while at the same time be used for interpolation on a coarser grid.

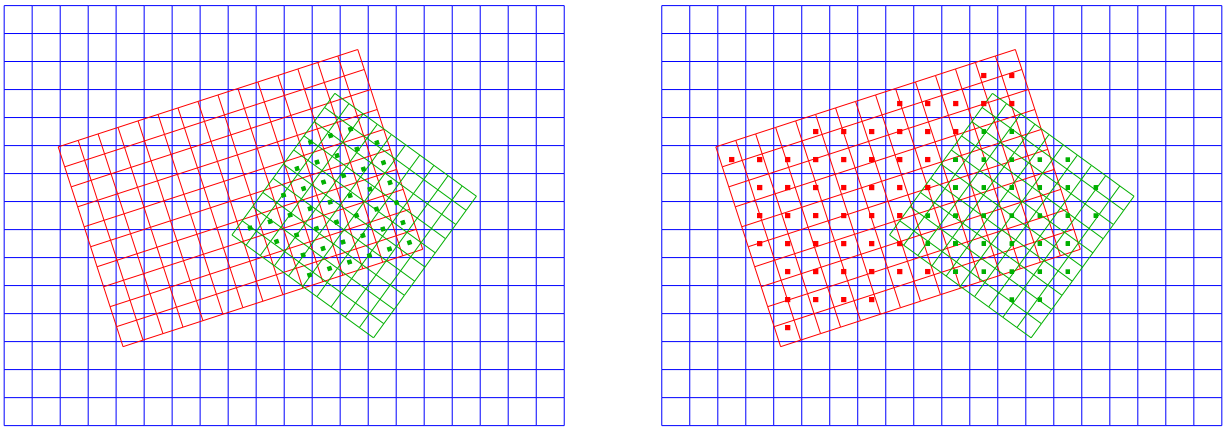


Figure 2: (Left) The green dots mark the centers of the red grid cells that are overlapped by the finer green grid, and so these cells can be filled with values interpolated from the green grid. (Right) The red and green dots mark the centers of the background grid cells that are overlapped by the finer red and green grids. The cells marked with green dots can be filled with values interpolated from the green grid, while the cells marked with red dots can be filled with values interpolated from the red grid.

Implicit discretizations require global coupling of degrees of freedom from different grids. This is accomplished using a Voronoi mesh as shown in Figure 3. Note that all the cell centers on the Voronoi mesh originate from one of the Cartesian grids as indicated by the color of each dot. Before constructing the Voronoi mesh, we first remove enough cell centers to prevent the remaining cell centers from overlapping or being overlapped by another grid. In order to avoid removing too many cells creating too large a gap between the degrees of freedom from different grids, we remove any coarse cell which contains the cell center of a finer cell as well as any coarse cell whose cell center is contained within a finer cell. Recursively, we only check a coarse cell against a finer cell after we first check whether the finer cell itself is removed or not. After selecting the degrees of freedom that will remain, the Voronoi mesh is constructed as in [14]. Note that Cartesian grids are already Voronoi meshes for their cell center degrees of freedom, and so we only need to compute Voronoi faces for cells along intergrid boundaries making the method quite efficient.

After carrying out computations on the Voronoi mesh, we need to update the values on the original Cartesian grids. As the Voronoi mesh is contiguous throughout the whole computational domain, it provides an interpolation stencil for every location in the domain. Standard multilinear interpolation can be used to interpolate to points that are interior to the regular Cartesian grid subset of the Voronoi mesh. Otherwise, we use the regular Delaunay mesh dual of the Voronoi mesh along with barycentric interpolation. See Figure 4. Vector value interpolation requires extra consideration when scalar components of the vectors are stored on face centers of the Voronoi mesh, e.g. fluxes. Thus, we first construct a full vector at each cell center of the Voronoi mesh using a least squares fit of the components at incident Voronoi faces. This is accomplished in [14] by approximating the vector field as a linear function in each cell obtaining a system that is often underdetermined and solved using regularized least squares [68, 22]. [15] found this approach sometimes highly inaccurate, and instead used a constant vector field in each cell obtaining an overconstrained system that can be solved with standard least squares. We use the method of [15] throughout. The resulting Voronoi cell center vector values are then interpolated to removed Cartesian grid cells as well as one layer of ghost cells in standard fashion, and then used to compute values on removed Cartesian grid faces via simple averaging.

The Voronoi mesh can also be used to fill the overlapped degrees of freedom depicted in Figure 2 simply by removing redundant degrees of freedom and constructing the Voronoi mesh depicted in Figure 3 before subsequently interpolating from the Voronoi mesh to the overlapped degrees of freedom as in Figure 4.

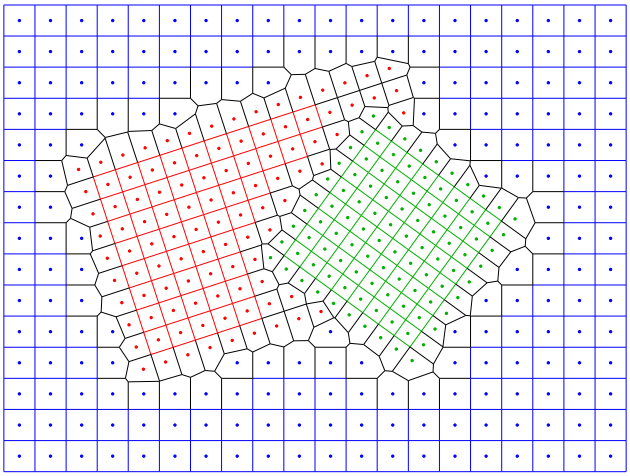


Figure 3: The Voronoi mesh used for implicitly coupling degrees of freedom across multiple grids. The black lines are the constructed Voronoi faces, while the green, red, and blue lines are the remaining original Cartesian grid faces. The dots represent cell centers from the original three grids and are color-coded to indicate which grid they came from.

This process ends up defining a slightly different set of cells as redundant overlapped degrees of freedom as compared to Figure 2, but is probably more contiguous. In fact, [15] found this method more accurate for treating the level set function, and we prefer this method for filling overlapped cells. This method can also be used for filling ghost cells in Figure 1.

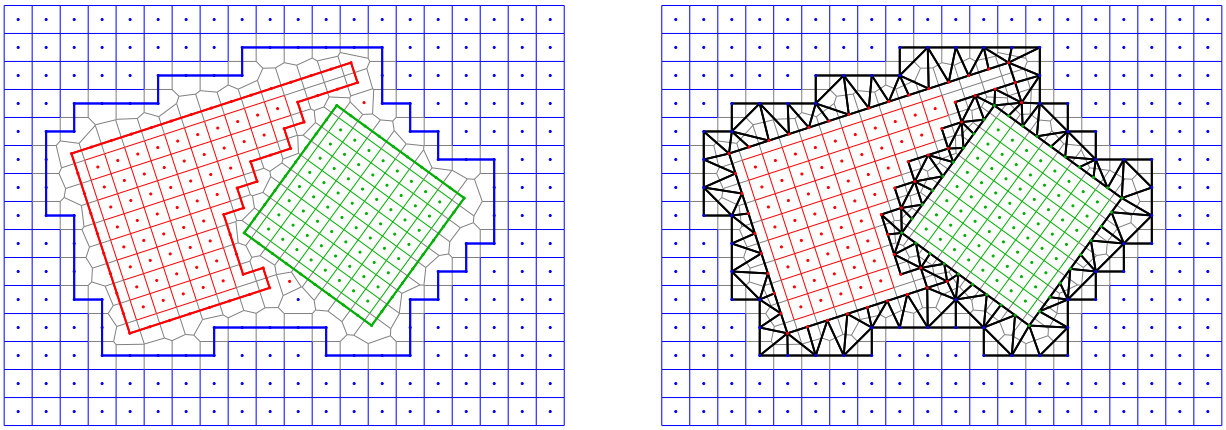


Figure 4: (Left) If a point lies outside the region enclosed by the thickened blue lines, we interpolate to this point from the background grid using standard multilinear interpolation. If a point lies inside the region enclosed by the thickened green/red lines, we interpolate from the green/red grid using standard multilinear interpolation. (Right) No valid multilinear interpolation stencils can be found for points outside the thickened green/red lines but inside the thickened blue lines. Points in this region lie in one of the Delaunay triangles (tetrahedra in three dimensions), and so we interpolate using standard barycentric interpolation.

4. Advection Strategies

Consider the discretization of the advection terms in Equation 1 for obtaining the intermediate values ρ^*, $(\rho\vec{u})^*$, and E^*. Since all the characteristic velocities of the advection terms are identically the fluid velocity $\vec{u}$, we advect each component of the state variables independently without transforming into characteristic

variables. As is discussed in Section 3, advection is performed on each Cartesian grid independently after filling a band of ghost cells. We set a single global CFL number which is used to compute a local time step on each grid, and the minimum of these is taken as the global time step Δt. For efficiency, the CFL number is typically chosen larger than 1, so that we are not limited by the small cell sizes on the finest grids. Since Δt may violate stability constraints, especially on finer grids, we subdivide Δt into a series of smaller time steps $(\Delta t_1, \Delta t_2, ..., \Delta t_m)$ for the purpose of advection, and subsequently solve for the pressure using the original full time step Δt. Using a CFL number of .9, we compute the allowable time step Δt_1 such that advection is stable on all grids, and then update all the grids' positions and orientations to time $t + \Delta t_1$ while also solving for advection. This process is iteratively repeated until the last time step Δt_m has an effective CFL number less than or equal to .9. Then, the more expensive pressure solve is done once using the full time step Δt. Note that one could have used larger time steps on the coarser slower moving grids or those with smaller effective fluid velocities utilizing an asynchronous time integration approach. However, this complicates load balancing for parallel processing, and thus it does not necessarily lead to performance improvements that warrant the complexity of the approach. We note that one can alternatively alleviate time step restrictions on finer grids or those with larger effective fluid velocities by using an unconditionally stable conservative semi-Lagrangian scheme [39].

4.1. Flux based advection

Consider the conservation law for updating a single component of Equation 1 according to the advection equation,

$$\phi_t + \nabla \cdot (\phi \vec{u}) = 0. \tag{10}$$

We discretize Equation 10 using third order accurate TVD Runge-Kutta in time and ENO-LLF in space [64], modified to account for the grid motion. For the sake of exposition, we consider two spatial dimensions. In the first Euler step of third order accurate TVD Runge-Kutta, we update the value of ϕ on every grid from time t^n to time t^{n+1} using the standard flux based formula,

$$\phi_{i,j}^{n+1} = \phi_{i,j}^n - \Delta t \left(\frac{F_{i+\frac{1}{2},j}^n - F_{i-\frac{1}{2},j}^n}{\Delta x} + \frac{G_{i,j+\frac{1}{2}}^n - G_{i,j-\frac{1}{2}}^n}{\Delta y} \right). \tag{11}$$

To compute the x-direction numerical fluxes for a static grid, the first divided differences at grid face $(i+\frac{1}{2}, j)$ are computed as

$$D_{k,j}^1 \mathcal{F}^{\pm} = \phi_{k,j}^n (\mathbf{R}^{-1} \vec{u}_{k,j}^n)_x \pm \alpha_{i+\frac{1}{2},j}^n \phi_{k,j}^n, \tag{12}$$

where k is set to i for $\mathcal{F}^+$ and $i+1$ for $\mathcal{F}^-$. Here $(\mathbf{R}^{-1}\vec{u})_x$ rotates a world space velocity into the object space of the grid according to Equation 9, and takes the first component of the resulting vector to represent the scalar velocity in the x-direction of that grid. Then, $\alpha_{i+\frac{1}{2},j}^n = \max(|(\mathbf{R}^{-1}\vec{u}_{i,j}^n)_x|, |(\mathbf{R}^{-1}\vec{u}_{i+1,j}^n)_x|)$ defines the standard high order diffusion coefficient. A first order accurate approximation of the numerical flux is then $F_{i+\frac{1}{2},j}^n = (D_{i,j}^1 \mathcal{F}^+ + D_{i+1,j}^1 \mathcal{F}^-)/2$, and higher order accurate approximations can be obtained by computing higher order divided differences in standard fashion; see [64] for more details.

For moving grids, we first compute an effective grid velocity $\vec{w}$ at the center of each cell using displacement of the world space locations,

$$\vec{w}_{i,j} = \frac{(\vec{x}_{\text{world}})_{i,j}^{n+1} - (\vec{x}_{\text{world}})_{i,j}^n}{\Delta t}. \tag{13}$$

This effective grid velocity is used to modify the characteristic velocity at cell (i, j) to be $\vec{u}_{i,j}^n - \vec{w}_{i,j}$. To obtain second order temporal accuracy, we replace $\mathbf{R}^{-1}\vec{u}_{k,j}^n$ in Equation 12 and in the definition of $\alpha_{i+\frac{1}{2},j}^n$ with

$$(\vec{u}_{\text{effective}})_{k,j}^n = \frac{(\mathbf{R}^n)^{-1}(\vec{u}_{k,j}^n - \vec{w}_{k,j}) + (\mathbf{R}^{n+1})^{-1}\left(\vec{u}^n\left((\mathbf{R}^n)^{-1}\left((\vec{x}_{\text{world}})_{k,j}^{n+1} - \vec{s}^n\right)\right) - \vec{w}_{k,j}\right)}{2}. \tag{14}$$

Here the time t^{n+1} location of the grid point $(\vec{x}_{\text{world}})_{k,j}^{n+1}$ is transformed into the time t^n object space of the grid according to Equation 8, i.e. $(\mathbf{R}^n)^{-1}((\vec{x}_{\text{world}})_{k,j}^{n+1} - \vec{s}^n)$, so that the fluid velocity at this point $\vec{u}^n((\mathbf{R}^n)^{-1}((\vec{x}_{\text{world}})_{k,j}^{n+1} - \vec{s}^n))$ can be computed using trilinear interpolation. Ignoring $(\mathbf{R}^n)^{-1}$ and $(\mathbf{R}^{n+1})^{-1}$, Equation 14 averages together the time t^n fluid velocity at the time t^n location and the time t^{n+1} location of the grid point, and subtracts the grid velocity $\vec{w}_{k,j}$ to make this a relative velocity. If the grid did not rotate, this average could simply be multiplied by $\mathbf{R}^{-1}$. However, the difference between the grid rotation at time t^n and time t^{n+1} requires further computation to determine which components of the velocity field correspond to the changing x-direction of the grid.

After completing the first Euler step for all the components of the state variables, we then further update ϕ^{n+1} to ϕ^{n+2} again using Equation 14 except replacing the time t^n fluid velocity values with the newly computed time t^{n+1} values. However, we stress that we use exactly the same values for the quantities related to the grid motion, i.e. $\mathbf{R}^n$, $\mathbf{R}^{n+1}$, $\vec{w}_{k,j}$, $(\vec{x}_{\text{world}})_{k,j}^{n+1}$, and $\vec{s}^n$. To understand this, we note that in the grid's object space the location $(\mathbf{R}^n)^{-1}((\vec{x}_{\text{world}})_{k,j}^{n+1} - \vec{s}^n)$ is displaced some distance from the grid point where $\vec{u}_{k,j}^n$ is stored, and that utilizing the same quantities related to the grid motion preserves this displacement. Thus, when updating ϕ^{n+1} to ϕ^{n+2} with $\vec{u}_{k,j}^{n+1}$ stored at this reference grid point, using the same displacement results in a velocity value valid for the time t^{n+2} grid point location. Furthermore, we do not change the values of the grid velocity from Equation 13, or the values of $(\mathbf{R}^n)^{-1}$ and $(\mathbf{R}^{n+1})^{-1}$ used to compute an average of the x-directions, as keeping these constant seems to be sufficient for second order accuracy.

After finishing the second Euler step, we average to obtain

$$\phi_{i,j}^{n+\frac{1}{2}} = \frac{3}{4}\phi_{i,j}^n + \frac{1}{4}\phi_{i,j}^{n+2}. \tag{15}$$

From these averaged ϕ values, we compute a new fluid velocity at time $t^{n+\frac{1}{2}}$ and take a subsequent Euler step to advance ϕ from time $t^{n+\frac{1}{2}}$ to time $t^{n+\frac{3}{2}}$, once again keeping all the quantities related to the grid motion constant. The final averaging step of

$$\phi_{i,j}^{n+1} = \frac{1}{3}\phi_{i,j}^n + \frac{2}{3}\phi_{i,j}^{n+\frac{3}{2}} \tag{16}$$

yields a second order accurate approximation of ϕ at time t^{n+1}. Note that we do not achieve the full third order accuracy due to our second order accurate approximation of the grid's motion. However, the stability region of third order accurate TVD Runge Kutta is still especially useful, since strong shocks in low dissipation situations have eigenvalues approaching purely imaginary behavior.

The time step restriction due to the CFL condition is computed using the same method as in [38] except replacing the fluid velocity with the effective velocity from Equation 14 yielding

$$\Delta t \left(\frac{|(\vec{u}_{\text{effective}})_x^n|_{\max} + \frac{|(\nabla p)_x^n|}{\rho^n}\Delta t}{\Delta x} + \frac{|(\vec{u}_{\text{effective}})_y^n|_{\max} + \frac{|(\nabla p)_y^n|}{\rho^n}\Delta t}{\Delta y} \right) \leq \text{CFL}, \tag{17}$$

where $(\nabla p)_x$ and $(\nabla p)_y$ are the pressure gradient components in the grid's x and y-directions. Note that $(\vec{u}_{\text{effective}})^n$ itself is a function of Δt and the grid motion. Thus, we apply a simple bisection approach as in [14] in order to find Δt. The number of ghost cells is chosen not only to accommodate what is needed for ENO-LLF and the three stages of third order accurate TVD Runge-Kutta, but also has to be large enough so that one can carry out the interpolations of velocities at $(\mathbf{R}^n)^{-1}((\vec{x}_{\text{world}})_{k,j}^{n+1} - \vec{s}^n)$ in Equation 14. This is further complicated by the grid motion as is pointed out in [14], and therefore we follow the approach of [14] here as well.

4.2. Conservative semi-Lagrangian advection

On a given grid, the flux based scheme may be replaced with the conservative semi-Lagrangian method [39] when solving Equation 10. We still use third order accurate TVD Runge-Kutta for the temporal

discretization, as we found that this significantly reduces artificial oscillations near the shock front. Similar to [14], we use world space velocities and locations. When taking large time steps, this seems to have smaller errors than those obtained using effective velocities in the grid's object space as per Equation 14. The main drawback of using world space velocities and locations is that one needs to conservatively remap conserved quantities in order to perform the Runge Kutta averaging in Equations 21 and 22, and it seems difficult in practice to robustly maintain second order accuracy during this combined remapping and Runge Kutta process. This is why we did not pursue this approach when carrying out flux based advection in Section 4.1 where our goal was to achieve second order accuracy.

In the first Euler step, in order to update grid cell i, we first compute a world space time t^n fluid velocity at $(\vec{x}_{\text{world}})_i^{n+1}$ by trilinearly interpolating $\vec{u}^n$ at the object space location $(\mathbf{R}^n)^{-1}((\vec{x}_{\text{world}})_i^{n+1} - \vec{s}^n)$ resulting in $\vec{u}^n((\mathbf{R}^n)^{-1}((\vec{x}_{\text{world}})_i^{n+1} - \vec{s}^n))$. Then, we trace a semi-Lagrangian ray back from $(\vec{x}_{\text{world}})_i^{n+1}$ to $(\vec{x}_{\text{world}})_i^{n+1} - \vec{u}^n((\mathbf{R}^n)^{-1}((\vec{x}_{\text{world}})_i^{n+1} - \vec{s}^n))\Delta t$, denoting this final location as $(\vec{x}_{\text{back}})_i$. Since ϕ^n is stored at the time t^n grid cell locations, we find the interpolation stencil at location $(\vec{x}_{\text{back}})_i$ in the grid's time t^n object space. That is, we find interpolation weights such that

$$(\mathbf{R}^n)^{-1}\Big((\vec{x}_{\text{back}})_i - \vec{s}^n\Big) = \sum_j W_{j,i}(\vec{x}_{\text{object}})_j, \tag{18}$$

where $W_{j,i}$ denotes the interpolation weight from cell j to cell i. After finding the interpolation weights for updating all the grid cells including any ghost cells that withdraw ϕ from the interior of the grid, we compute the total contribution from each cell k (also including ghost cells) as $\sigma_k = \sum_j W_{k,j}$, scaling down the weights $W_{k,j}$ to $W_{k,j}/\sigma_k$ when $\sigma_k \geq 1$. If $\sigma_k < 1$, we forward advect the remaining weight $1 - \sigma_k$ as follows. First, we trace forward a semi-Lagrangian ray from $(\vec{x}_{\text{world}})_k^n$ to $(\vec{x}_{\text{forward}})_k = (\vec{x}_{\text{world}})_k^n + \vec{u}_k^n \Delta t$. Since ϕ^{n+1} is stored at the time t^{n+1} grid cell locations, we find interpolation weights for $(\vec{x}_{\text{forward}})_k$ in the grid's time t^{n+1} object space as

$$(\mathbf{R}^{n+1})^{-1}\Big((\vec{x}_{\text{forward}})_k - \vec{s}^{n+1}\Big) = \sum_j \hat{W}_{j,k}(\vec{x}_{\text{object}})_j, \tag{19}$$

where $\hat{W}_{j,k}$ denotes the interpolation weight from cell j to cell k. Then, ϕ^{n+1} is computed as

$$\phi_i^{n+1} = \sum_j (W_{j,i} + (1 - \sigma_j)\hat{W}_{i,j})\phi_j^n. \tag{20}$$

See [39] for more details.

In the second Euler step, we update ϕ from time t^{n+1} to time t^{n+2}. Since a moving grid is like a window that stores a region of ϕ in the world space, we note that there can be many different valid choices of grid positions and orientations for time t^{n+2}, and what is important is that we interpolate and store values at correct world space locations based on the chosen grid position and orientation. For simplicity, we set the grid position and orientation at time t^{n+2} to be the same as that at time t^{n+1}, and found this sufficient for first order accuracy. In order to update grid cell i, we first compute $\vec{u}_i^{n+1}$ from ϕ_i^{n+1} and then trace back from $(\vec{x}_{\text{world}})_i^{n+1}$ using this velocity. The interpolation stencils and weights are computed using the grid's time t^{n+1} object space, and the weights are subsequently scaled down or forward advected as usual. For each cell k that needs forward advection, we trace forward from $(\vec{x}_{\text{world}})_k^{n+1}$ using $\vec{u}_k^{n+1}$, and then find the interpolation stencils and weights again using the grid's time t^{n+1} object space. Finally, ϕ_i^{n+2} is computed as usual along the lines of Equation 20.

It is not straightforward to average the values of ϕ^n and ϕ^{n+2} in world space, since they correspond to different grid positions and orientations. Moreover, trilinearly interpolating before averaging will violate conservation. Therefore, we conservatively advect ϕ^n at the time t^n grid cell locations to $(\phi_{\text{remap}})^n$ at the time t^{n+1} grid cell locations using a fluid velocity field that is identically zero everywhere. The implementation of this conservative remapping is exactly the same as the first Euler step except replacing $\vec{u}^n$ with $\vec{0}$ everywhere.

Then, we compute $\phi^{n+\frac{1}{2}}$ at the time t^{n+1} grid cell locations as

$$\phi_i^{n+\frac{1}{2}} = \frac{3}{4}(\phi_{\text{remap}})_i^n + \frac{1}{4}\phi_i^{n+2}. \tag{21}$$

Subsequently, we take the third Euler step to update $\phi^{n+\frac{1}{2}}$ to $\phi^{n+\frac{3}{2}}$, again setting $\phi^{n+\frac{3}{2}}$ to be stored at the time t^{n+1} grid cell locations. The implementation is the same as the second Euler step except replacing $\vec{u}^{n+1}$ with $\vec{u}^{n+\frac{1}{2}}$. The final averaging step

$$\phi_i^{n+1} = \frac{1}{3}(\phi_{\text{remap}})_i^n + \frac{2}{3}\phi_i^{n+\frac{3}{2}} \tag{22}$$

yields a first order accurate approximation to ϕ at time t^{n+1} stored at the time t^{n+1} grid cell locations.

Note that we need to allocate enough ghost cells so that the semi-Lagrangian rays from the interior of a grid do not reach outside the ghost domain; see [39] for more details. When using TVD Runge-Kutta, the fluid velocity may increase in any Euler step, and so a time step computed based on $\vec{u}^n$ may not be sufficient. Although there are many potential remedies, we simply use a smaller time step in this situation.

4.3. Numerical results

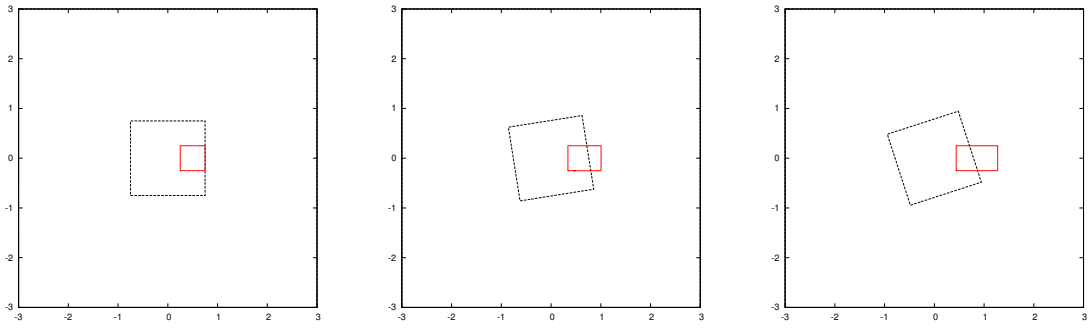
We carried out a number of examples in both one and two spatial dimensions. We observed second order accuracy when using the flux based scheme with at least second order accurate ENO-LLF, and first order accuracy when using the conservative semi-Lagrangian scheme on some of the finer grids. Here we show a typical example for illustrative purposes. Figure 5 shows the computational set up of two overlapping grids with the finer one rotating. The object space domains, cell sizes, positions, and orientations of the grids are listed in Table 1. The fluid velocity is varying both in time and space according to the function $\vec{u}(x, y, t) = (\cos(\frac{\pi}{4}t)\sin(\frac{\pi}{5}x), 0)$. The initial condition of the advected scalar field ϕ is specified as

$$\phi(x, y, t = 0) = \begin{cases} 1 & \text{if } |x - .5| < .25, |y| < .25, \\ 0 & \text{otherwise.} \end{cases}$$

The third order accurate ENO-LLF scheme is used on both of the grids with a global CFL number of .5. The order of accuracy for the results at time $t = 1$ is computed by averaging the pointwise order of accuracy at 3000 uniformly spacing points between $(.7, 0)$ and $(1, 0)$ resulting in second order accuracy as shown in Table 2(a). Figure 6 shows plots of convergence results on the slice $y = 0$. Next, using a global CFL number of 1.8, Table 2(b) and Figure 7 show the results when the conservative semi-Lagrangian scheme is used on the finer rotating grid. Table 2(c) and Figure 8 show the results from a global CFL number of 3 with the conservative semi-Lagrangian scheme used on both of the grids.

	Object space domain	Δx	$\vec{s}$	θ
1	$[-3, 3] \times [-3, 3]$	$6/n$	$(0, 0)$	0
2	$[-.75, .75] \times [-.75, .75]$	$3/n$	$(0, 0)$	$\frac{\pi}{10}t$

Table 1: The object space domains, cell sizes, positions, and orientation angles of the grids used in the advection tests. n indicates the number of grid cells in one spatial dimension.



(a) t=0 (b) t=.5 (c) t=1

Figure 5: Results of advecting a square bump through a time-varying divergent velocity field on two overlapping grids with grid resolution $n = 3200$ using the third order accurate ENO-LLF scheme on both grids. The dashed outline indicates the boundary of the rotating fine grid. The red outline is the $\phi = .3$ contour of the advected scalar field.

(a)		(b)		(c)	
n	Order	n	Order	n	Order
400	2.55	800	.97	1600	.92
800	1.86	1600	.64	3200	.93
1600	2.13	3200	.97	6400	.74
3200	1.99	6400	.98	12800	.87

Table 2: The convergence results at time $t = 1$ for the advection tests on two overlapping grids. (a) The third order accurate ENO-LLF scheme is used on both grids. (b) The conservative semi-Lagrangian scheme is used on the finer rotating grid. (c) The conservative semi-Lagrangian scheme is used on both grids. The ground truth used in computing the order of accuracy is the result on a single static grid of resolution $n = 51,200$ using the third order accurate ENO-LLF scheme.

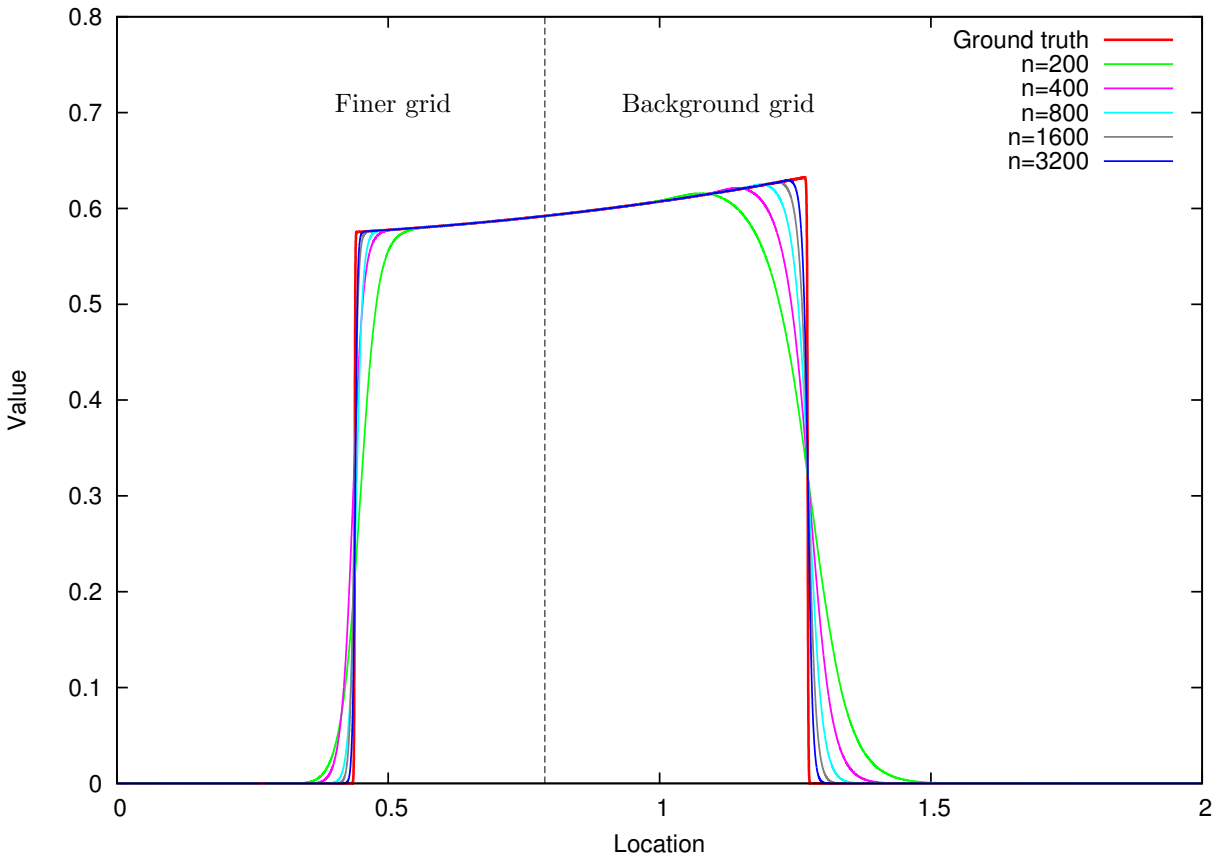


Figure 6: The convergence results at time $t = 1$ on the slice $y = 0$ for the advection tests on two overlapping grids using the third order accurate ENO-LLF scheme on both grids.

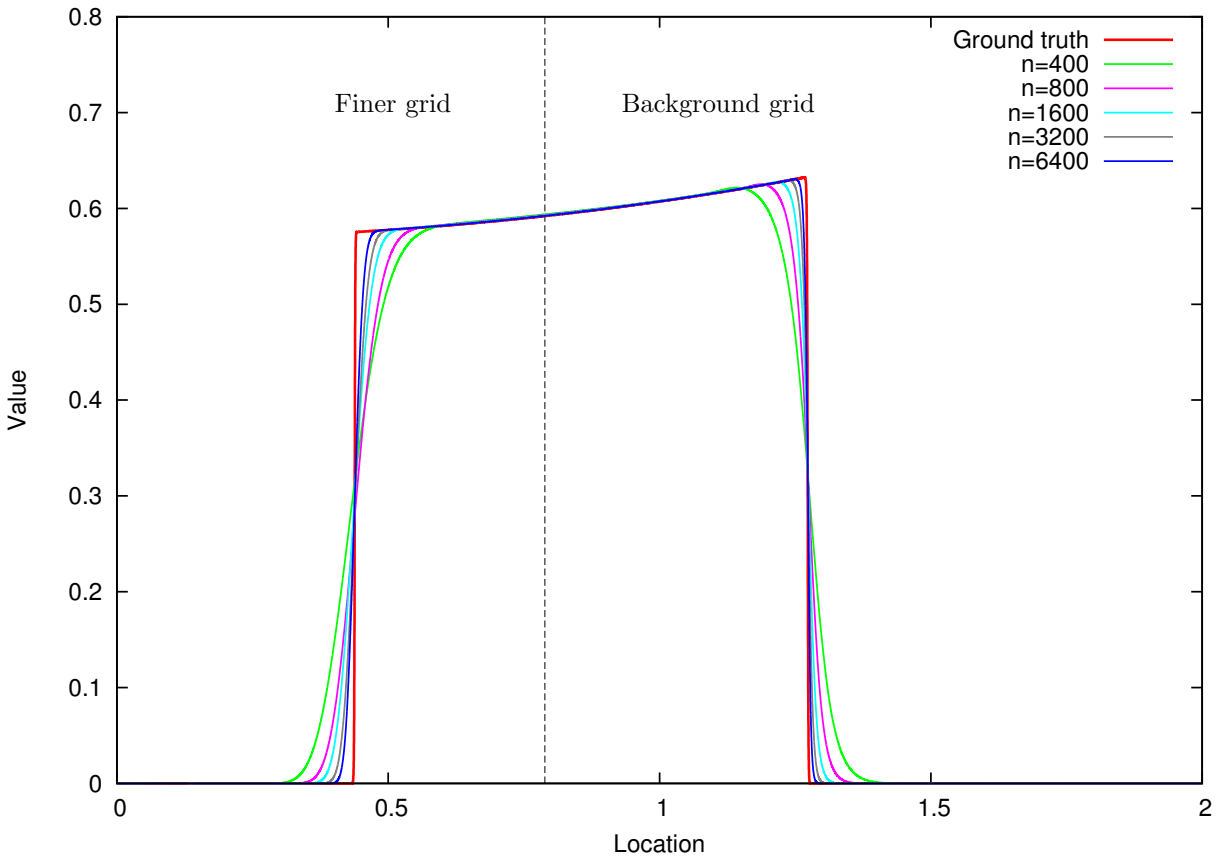


Figure 7: The convergence results at time $t = 1$ on the slice $y = 0$ for the advection tests on two overlapping grids using the conservative semi-Lagrangian scheme on the finer rotating grid.

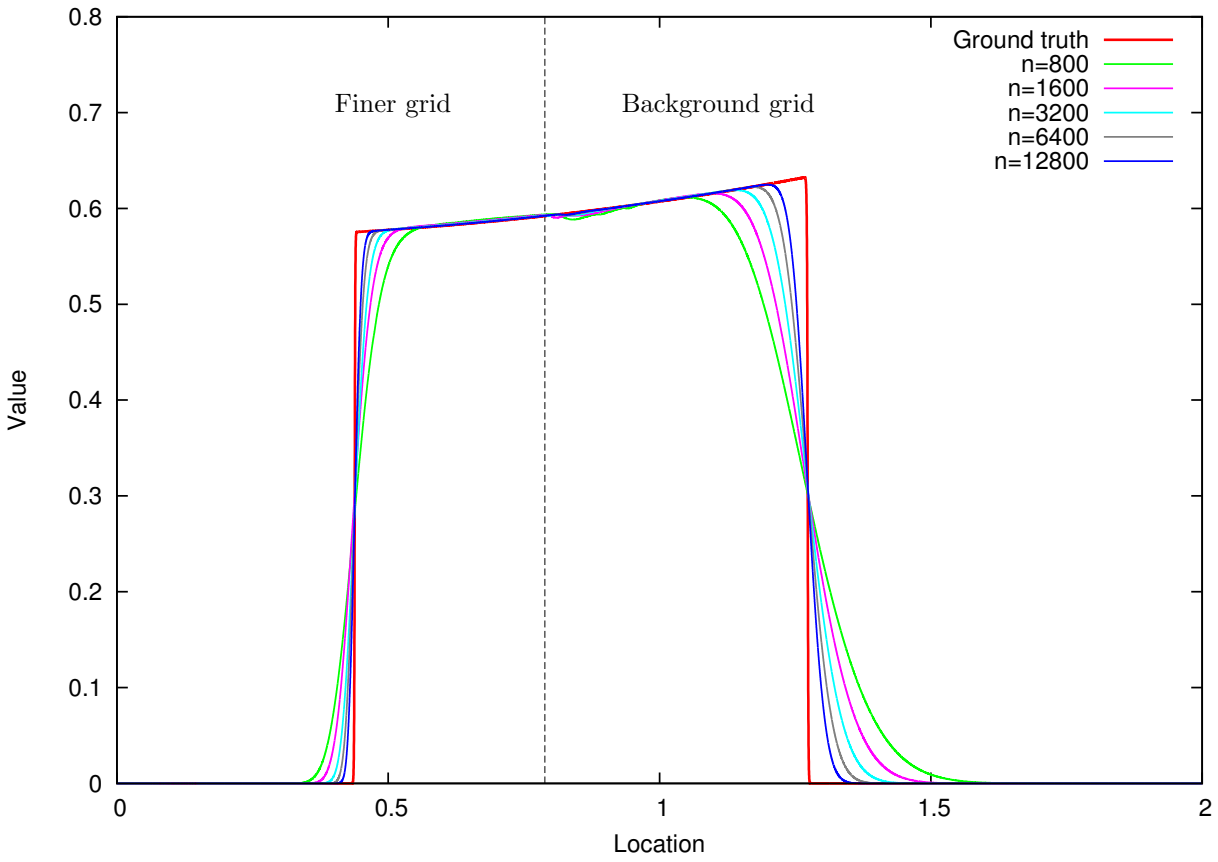


Figure 8: The convergence results at time $t = 1$ on the slice $y = 0$ for the advection tests on two overlapping grids using the conservative semi-Lagrangian scheme on both grids.

5. Solving for the Pressure

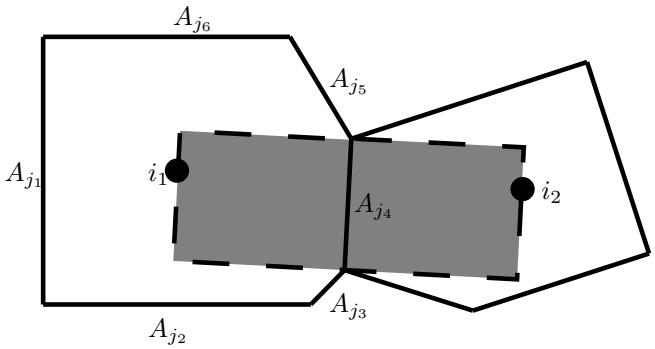


Figure 9: Two adjacent Voronoi cells i_1 and i_2. A_{j_1}, A_{j_2}, ..., A_{j_6} denote the areas of the Voronoi faces adjacent to Voronoi cell i_1. The shaded region denotes the volume of the dual cell of face j_4.

We solve for the pressure by discretizing Equation 7 on the contiguous Voronoi mesh as shown for example in Figure 3. The pressure p^{n+1} is defined at Voronoi cell centers, and we follow the method proposed in [14] for the discretization of the divergence and gradient operators yielding

$$[\mathbf{N} + \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G}]\hat{\mathbf{p}}^{n+1} = \mathbf{N}\hat{\mathbf{p}}^a + \mathbf{G}^T\mathbf{u}_f^*, \tag{23}$$

where $\mathbf{N} = [\frac{V_i}{\rho_i^n (c_i^n)^2 \Delta t^2}]$ is a diagonal matrix with V_i the volume of Voronoi cell i. ρ_i^n and c_i^n are evaluated using the conserved variables at time t^n and the equation of state noting that the cell centers of the Voronoi mesh are cell centers of Cartesian grids so the conserved variables are defined there. $\mathbf{G}$ is the discrete volume-weighted gradient operator, $\boldsymbol{\beta}$ is a diagonal matrix with each diagonal entry representing the fluid mass in the dual cell of a Voronoi face (including the regular Cartesian grid faces that remain on the Voronoi mesh), $\hat{\mathbf{p}}$ is the vector of discrete pressure values scaled by Δt, and $\mathbf{u}_f^*$ is a vector of the Voronoi face normal components of $\vec{u}^*$.

Each row of the volume weighted divergence operator $-\mathbf{G}^T$ corresponds to one Voronoi cell. Consider the volume weighted divergence of cell i_1 in Figure 9. According to the divergence theorem, the volume weighted divergence can be discretized as $\sum_{j \in \text{Face}(i_1)} A_j \vec{n}_j^{i_1} \cdot \vec{u}_j$, where A_j is the area of face j and $\vec{n}_j^{i_1}$ is the outward face normal. Note that the outward face normal $\vec{n}_{j_4}^{i_1}$ for cell i_1 and $\vec{n}_{j_4}^{i_2}$ for cell i_2 are in opposite directions on face j_4. In order to compute a unique scalar component of $\vec{u}_{j_4}$ to store on face j_4, we need to pick a single direction for face j_4, and thus we define a unit vector $\vec{e}_{j_4}$ such that

$$\vec{n}_{j_4}^k = s_{j_4}^k \vec{e}_{j_4}, \quad k = i_1 \text{ or } i_2,$$

where $s_{j_4}^k = \pm 1$ is the sign variable. Using an arbitrarily chosen $\vec{e}_j$ for each face of the Voronoi mesh, we store a scalar component on each face j as $u_j = \vec{e}_j \cdot \vec{u}_j$, and express the volume weighted divergence of cell i_1 as $\sum_{j \in \text{Face}(i_1)} A_j s_j^{i_1} u_j$.

Next, consider the negated transpose of $-\mathbf{G}^T$, i.e. the volume weighted gradient operator $\mathbf{G}$. Taking Voronoi face j_4 in Figure 9 as an example and assuming $\vec{e}_{j_4}$ points from cell i_1 to cell i_2, the row of $\mathbf{G}$ corresponding to face j_4 has an entry of $-A_{j_4}$ in the column corresponding to cell i_1 and an entry of A_{j_4} in the column corresponding to cell i_2. Thus, $\mathbf{Gp}$ computes the face normal component of the volume weighted pressure gradient at face j_4 as $(p_{i_2} - p_{i_1})A_{j_4}$, which scales the face normal component of the pressure gradient $\frac{p_{i_2} - p_{i_1}}{||\vec{x}_{i_2} - \vec{x}_{i_1}||}$ by the volume $||\vec{x}_{i_2} - \vec{x}_{i_1}||A_{j_4}$. This volume is shaded in Figure 9, and we define this volume as the volume of the dual cell of face j_4. When computing the fluid mass in the dual cell of a Voronoi face, i.e.

$\boldsymbol{\beta}$, it is important that we consistently define the volume of a dual cell in this manner. For example, for face j_4, we compute the fluid mass in its dual cell as

$$\frac{\rho_{i_1} + \rho_{i_2}}{2} ||\vec{x}_{i_2} - \vec{x}_{i_1}|| A_{j_4},$$

since face j_4 is the bisector of the segment between $\vec{x}_{i_1}$ and $\vec{x}_{i_2}$ and thus the volume of the dual cell is half inside cell i_1 and half inside cell i_2. The velocity field $\mathbf{u}_f^*$ used in Equation 23 is computed in a similar fashion. That is, the fluid momentum in the dual cell of face j_4 is computed as

$$\frac{\rho_{i_1}\vec{u}_{i_1} + \rho_{i_2}\vec{u}_{i_2}}{2} ||\vec{x}_{i_2} - \vec{x}_{i_1}|| A_{j_4},$$

which can be divided by the fluid mass in the dual cell to obtain

$$\vec{u}_{j_4} = \frac{\rho_{i_1}\vec{u}_{i_1} + \rho_{i_2}\vec{u}_{i_2}}{\rho_{i_1} + \rho_{i_2}}. \tag{24}$$

In order to update the momentum and energy in Voronoi cell i_1 using the pressure resulting from Equation 23, we first compute the time t^{n+1} pressure on the cell's incident Voronoi faces following the same strategy as computing the face pressure on Cartesian grid faces in [38]. For example, the pressure on Voronoi face j_4 is computed as

$$p_{j_4}^{n+1} = \frac{p_{i_2}^{n+1}\rho_{i_1}^{n+1} + p_{i_1}^{n+1}\rho_{i_2}^{n+1}}{\rho_{i_2}^{n+1} + \rho_{i_1}^{n+1}}. \tag{25}$$

Using the time t^{n+1} pressure on all the incident Voronoi faces, we update the momentum in Voronoi cell i_1 via

$$(\rho\vec{u})_{i_1}^{n+1} = (\rho\vec{u})_{i_1}^{*} - \frac{\Delta t}{V_{i_1}} \sum_{j \in \text{Face}(i_1)} p_j^{n+1} A_j \vec{n}_j^{i_1}. \tag{26}$$

In order to update the energy, we first compute the time t^{n+1} scalar component of the fluid velocity on all incident Voronoi faces. For example, for face j_4 assuming $\vec{e}_{j_4}$ points from cell i_1 to cell i_2, we obtain

$$u_{j_4}^{n+1} = u_{j_4}^{*} - \frac{\Delta t}{\rho_{j_4}^{n+1} ||\vec{x}_{i_2} - \vec{x}_{i_1}||}(p_{i_2}^{n+1} - p_{i_1}^{n+1}), \tag{27}$$

where the face density $\rho_{j_4}^{n+1}$ is computed by averaging the density from the two neighboring Voronoi cells. After obtaining the time t^{n+1} scalar component of the fluid velocity on every Voronoi face, the energy in Voronoi cell i_1 is updated via

$$E_{i_1}^{n+1} = E_{i_1}^{*} - \frac{\Delta t}{V_{i_1}} \sum_{j \in \text{Face}(i_1)} p_j^{n+1} u_j^{n+1} A_j s_j^{i_1}. \tag{28}$$

Equations 26 and 28 provide time t^{n+1} values for momentum and energy on the Voronoi mesh that can be used to interpolate values onto the Cartesian grids; however, this violates conservation. Instead, in order to preserve conservation on each Cartesian grid including the background grid, we interpolate pressure from the Voronoi mesh to removed Cartesian grid cells as well as one layer of ghost cells using the method in Section 3. Then, each Cartesian grid independently follows the method from [38]. For example, in one spatial dimension, we compute the pressure on the grid face

$$p_{k+\frac{1}{2}}^{n+1} = \frac{p_{k+1}^{n+1}\rho_k^{n+1} + p_k^{n+1}\rho_{k+1}^{n+1}}{\rho_{k+1}^{n+1} + \rho_k^{n+1}}, \tag{29}$$

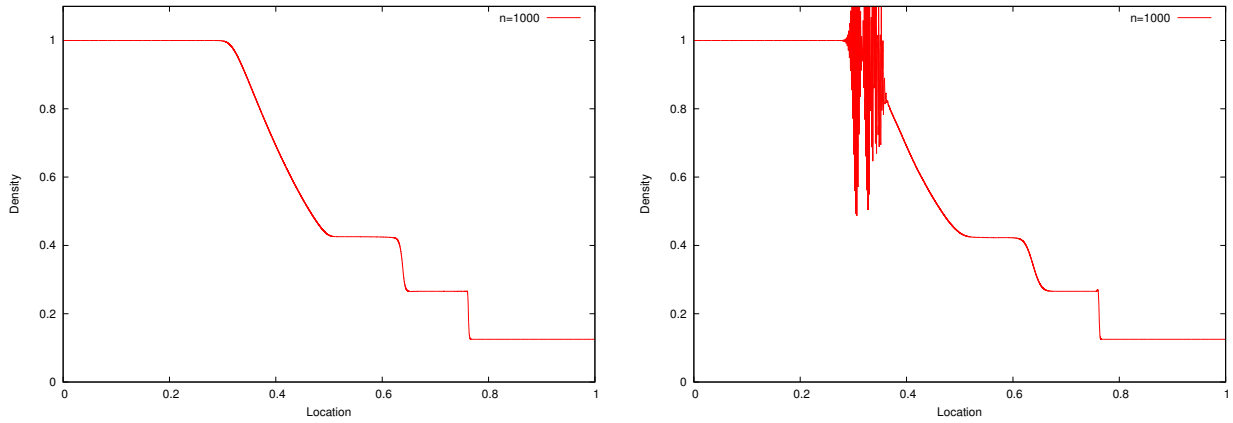


Figure 10: A standard Sod shock tube test on a single static one-dimensional grid computed using the second order accurate ENO-LLF scheme (left) or the conservative semi-Lagrangian scheme (right) for advection, followed by an implicit pressure update. The grid domain is $[0, 1]$ and the number of grid cells is 1000. The CFL number is .5 in both tests. Severe spurious ocillations are observed when using the conservative semi-Lagrangian scheme.

from which one can update the momentum

$$(\rho u)_k^{n+1} = (\rho u)_k^* - \frac{\Delta t}{\Delta x}(p_{k+\frac{1}{2}}^{n+1} - p_{k-\frac{1}{2}}^{n+1}). \tag{30}$$

Then, the face velocity is computed via

$$u_{k+\frac{1}{2}}^{n+1} = u_{k+\frac{1}{2}}^* - \frac{\Delta t}{\rho_{k+\frac{1}{2}}^{n+1}\Delta x}(p_{k+1}^{n+1} - p_k^{n+1}), \tag{31}$$

where the face density $\rho_{k+\frac{1}{2}}^{n+1}$ is computed by averaging the density from the two neighboring grid cells. Finally, the energy is updated via

$$E_k^{n+1} = E_k^* - \frac{\Delta t}{\Delta x}(p_{k+\frac{1}{2}}^{n+1} u_{k+\frac{1}{2}}^{n+1} - p_{k-\frac{1}{2}}^{n+1} u_{k-\frac{1}{2}}^{n+1}). \tag{32}$$

6. High Order Diffusion

The implicit discretization of the pressure terms in Section 5 contains inherent central differencing which can lead to spurious oscillations. This problem is exacerbated when using the conservative semi-Lagrangian scheme for discretizing the advection terms, as shown in Figures 10 and 11. We surmise that the ENO-LLF scheme behaves better than the conservative semi-Lagrangian scheme because of the extra high order dissipation. However, the ENO-LLF scheme is only using eigenvalues from the advection terms, without including the sound speeds, making its diffusion too small as well.

Noting that we may want to add even more high order diffusion than the ENO-LLF scheme typically does, we proceed with an implicit discretization, which of course needs to be globally coupled. Thus, we utilize the Voronoi mesh obtaining

$$(\hat{\mathbf{V}} + \mathbf{G}^T \hat{\boldsymbol{\mu}} \mathbf{G})\boldsymbol{\phi}^{n+1} = \hat{\mathbf{V}}\tilde{\boldsymbol{\phi}}, \tag{33}$$

where $\tilde{\boldsymbol{\phi}}$ and $\boldsymbol{\phi}^{n+1}$ are vectors of ϕ values before and after applying the high order diffusion, respectively, both discretized at Voronoi cell centers. $\hat{\mathbf{V}} = [\frac{V_i}{\Delta t}]$ is a diagonal matrix, and $\hat{\boldsymbol{\mu}} = [\frac{\mu_j}{(V_f)_j}]$ is a diagonal matrix

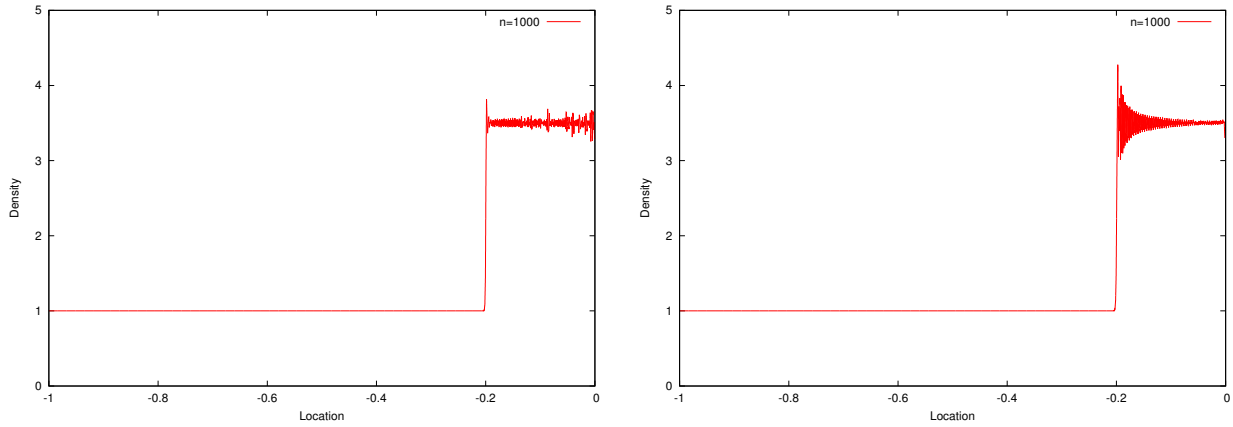


Figure 11: A shock is reflected from a wall on a single static one-dimensional grid computed using the second order accurate ENO-LLF scheme (left) or the conservative semi-Lagrangian scheme (right) for advection, followed by an implicit pressure update. The grid domain is $[-1, 0]$ and the number of grid cells is 1000. The initial condition has density 1, velocity 1, and pressure .2 throughout the domain. A standard reflective solid wall boundary condition is applied to the right-hand boundary of the domain. The CFL number is .5 in both tests.

with μ_j the high order diffusion coefficient at Voronoi face j and $(V_f)_j$ the volume of the dual cell of face j (the shaded area in Figure 9).

The high order diffusion coefficient μ at for example Voronoi face j_4 in Figure 9 is computed using the state variables in the two neighboring Voronoi cells i_1 and i_2. For each of these two Voronoi cells, we compute the eigenvalues of the full Jacobian matrix of the flux terms in the direction of $\vec{e}_{j_4}$. For cell i_1, we obtain $\vec{u}_{i_1} \cdot \vec{e}_{j_4} - c_{i_1}$, $\vec{u}_{i_1} \cdot \vec{e}_{j_4}$, and $\vec{u}_{i_1} \cdot \vec{e}_{j_4} + c_{i_1}$ and choose the eigenvalue with the largest absolute value denoting this as $(|\lambda|_{\max})_{i_1}$. Then,

$$\mu_{j_4} = \frac{||\vec{x}_{i_2} - \vec{x}_{i_1}||}{2} \max\Big((|\lambda|_{\max})_{i_1}, (|\lambda|_{\max})_{i_2}\Big) \tag{34}$$

is the high order diffusion coefficient proportional to Δx, so the high order diffusion vanishes as $\Delta x \to 0$. Note that the same diffusion coefficients are used for all the components of the state variables, since transformations into the characteristic fields are not utilized.

The ϕ^{n+1} values resulting from Equation 33 may not be exactly conservative due to the numerical errors and tolerances of the linear solver, and thus we instead use these ϕ^{n+1} values to compute numerical fluxes at Voronoi faces. For example, for face j_4, we compute a numerical flux of

$$(F_{\text{diffusion}})_{j_4} = -\mu_{j_4} \frac{\phi_{i_2}^{n+1} - \phi_{i_1}^{n+1}}{||\vec{x}_{i_2} - \vec{x}_{i_1}||}. \tag{35}$$

Using the numerical fluxes on all the incident Voronoi faces, the ϕ value in cell i_1 can be updated via

$$\phi_{i_1}^{n+1} = \tilde{\phi}_{i_1} - \frac{\Delta t}{V_{i_1}} \sum_{j \in \text{Face}(i_1)} (F_{\text{diffusion}})_j A_j s_j^{i_1}. \tag{36}$$

However, in order to maintain conservation on each Cartesian grid, including the background grid, we can instead interpolate the numerical fluxes in Equation 35 from the Voronoi mesh to the removed Cartesian grid faces using the method from Section 3. Then, the ϕ values on each Cartesian grid can be updated independently using the standard flux based method. Note that Equation 35 uses only first divided differences, which typically results in more diffusion than using higher order divided differences in the standard ENO-LLF scheme. Thus, our preferred method interpolates ϕ^{n+1} from the Voronoi mesh to the removed

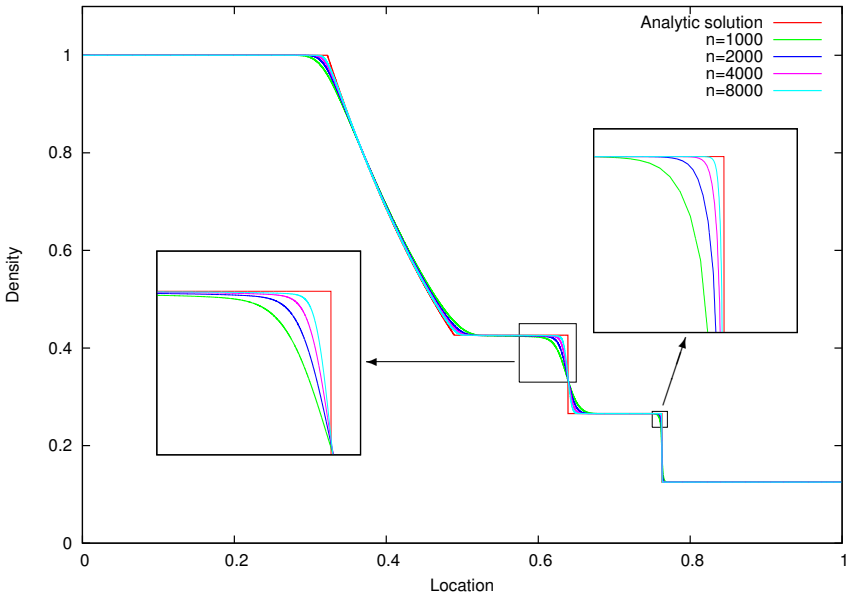


Figure 12: Same as Figure 10 except using the high order diffusion and showing the convergence results.

Cartesian grid cells, and uses these new ϕ^{n+1} values to construct higher order divided differences that are used to compute new diffusion fluxes on the Cartesian grid faces that are also Voronoi faces. The Cartesian grid faces that are not Voronoi faces are still updated with the interpolated values of numerical fluxes from Equation 35. This applies more diffusion near intergrid boundaries, which is often desirable. Furthermore, we have found that the increased diffusion is often desirable for grids with effective CFL numbers larger than 4, and therefore we do not use higher order divided differences for any of the faces of these grids. Figures 12 and 13 show that high order diffusion applied in the aforementioned manner alleviates the issues depicted in Figures 10 and 11.

7. Conservation

The methods proposed in Sections 4, 5, and 6 for advection, pressure based fluxes, and high order diffusion are all fully conservative on the background grid as well as on the interior of every other grid neglecting the lower dimensional influence of ghost cells. In regions where grids overlap, duplicate values of the conserved variables tend to drift apart. Although this could be rectified by filling overlapped cells with interpolated values, that violates conservation. Instead, we treat interpolated values as target values and conservatively remap values in overlapped regions. This is similar in spirit to [37, 47, 46, 45, 9]. The spatially contiguous target values are defined on the Voronoi mesh and created by applying the pressure fluxes specified in Equations 26 and 28 as well as the high order diffusion specified in Equation 36. Moreover, on the Voronoi faces which are also Cartesian grid faces where the higher order divided differences were used to compute diffusion fluxes, we utilize these higher order fluxes in Equation 36. A similar consideration is not necessary for the pressure fluxes, since Equations 26 and 28 reduce to Equations 30 and 32 where the Voronoi mesh and Cartesian grid faces agree.

For the Cartesian grid cell centers that agree in value with their Voronoi cell center counterparts, nothing needs to be done. However, the values at the Cartesian grid cell centers that were removed from the Voronoi mesh will not generally agree with the values interpolated from the Voronoi mesh, so these cells require conservative remapping. In addition, the Cartesian grid cell centers that remain on the Voronoi mesh but do not degenerate to Cartesian grid cells on the Voronoi mesh, rather having irregular boundaries, will

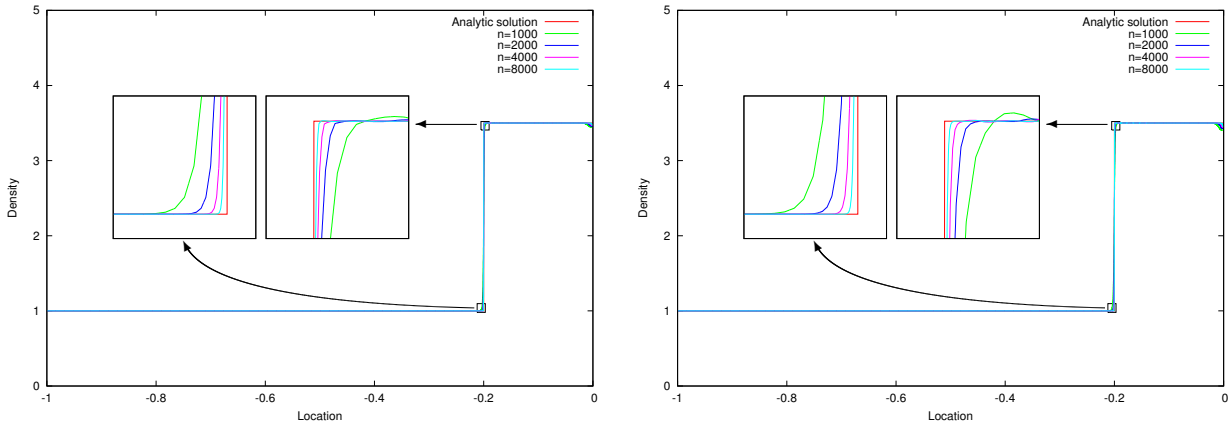


Figure 13: Same as Figure 11 except using the high order diffusion and showing the convergence results.

also disagree in value with their counterparts on the Voronoi mesh. Thus, these cells require conservative remapping as well.

Consider each grid independently. We conservatively remap the values of ϕ such that the net change is identically zero, i.e. $\sum \Delta\phi_i = 0$, where we have eliminated the volume of each grid cell since it is the same for every cell of a particular Cartesian grid. We proceed as follows. First, we use the target value or an interpolated target value in order to compute a target $\Delta\phi_i$ for every cell under consideration. Then, we compute the gain σ_{gain} as the sum of all $\Delta\phi_i$ where $\Delta\phi_i > 0$ and the loss σ_{loss} as the absolute value of the sum of all $\Delta\phi_i$ where $\Delta\phi_i < 0$. If $\sigma_{\text{gain}} > \sigma_{\text{loss}}$, then we need not adjust any of the cells where $\Delta\phi_i < 0$, since they obtain their target values by losing material when there is a net gain of material. In this case we achieve conservation simply by scaling down $\Delta\phi_i$ for all cells where $\Delta\phi_i > 0$ using the factor $\sigma_{\text{loss}}/\sigma_{\text{gain}}$. Similarly, if $\sigma_{\text{gain}} < \sigma_{\text{loss}}$, we scale down $\Delta\phi_i$ for all cells where $\Delta\phi_i < 0$ using the factor $\sigma_{\text{gain}}/\sigma_{\text{loss}}$. The resulting values are always bounded between the original values and the target values, see for example Figure 14. Since the values on both the Voronoi mesh and Cartesian grid converge to the analytic solution under grid refinement, bounding the remapped values between them guarantees convergence while minimizing the creation of extrema. This conservative remapping method is applied to density, momentum, and energy at the end of each time step.

7.1. Positivity preservation

Since the conservative remapping method bounds the remapped values between the original values and the target values, the remapped density and energy remain positive. However, the internal energy computed from the remapped conserved variables does not necessarily stay positive. This is rectified as follows. We clamp the proposed change of the conserved variables $\Delta\rho_i$, $\Delta(\rho\vec{u})_i$, and ΔE_i using a scaling factor $\lambda_i \in [0, 1]$ such that the resulting internal energy is bounded from below, i.e.

$$\frac{E_i + \lambda_i \Delta E_i}{\rho_i + \lambda_i \Delta\rho_i} - \frac{||(\rho\vec{u})_i + \lambda_i \Delta(\rho\vec{u})_i||^2}{2(\rho_i + \lambda_i \Delta\rho_i)^2} \geq (e_{\min})_i, \tag{37}$$

where

$$(e_{\min})_i = .9 \ \min\{e_i, (e_{\text{target}})_i\}, \tag{38}$$

and $(e_{\text{target}})_i$ is computed from the target conserved variables. Equation 37 can be rearranged into the form

$$A\lambda_i^2 + B\lambda_i + C \geq 0, \tag{39}$$

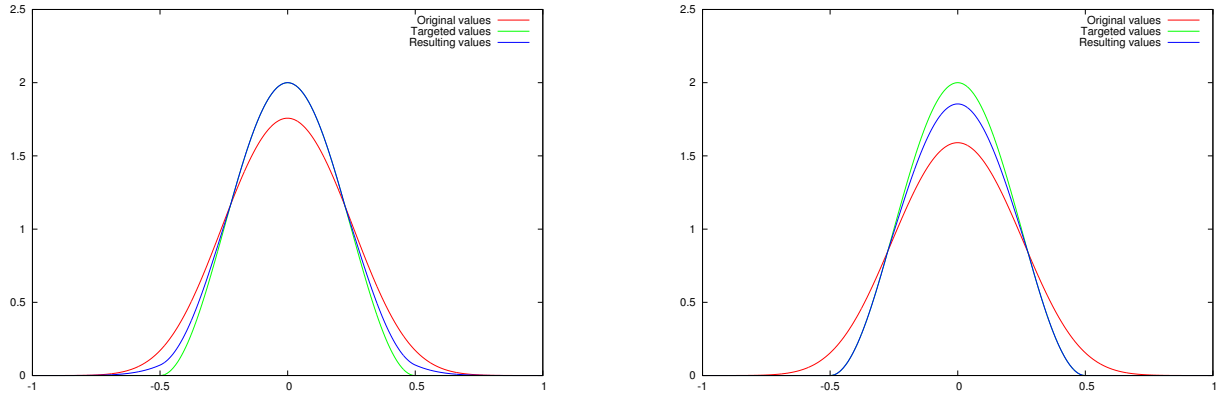


Figure 14: Two examples of conservatively remapping the original values toward the target values. (Left) $\sigma_{\text{gain}} < \sigma_{\text{loss}}$, so the original values increase to meet their target values where necessary, while the original values that attempt to decrease toward their target values have their decrease limited. (Right) $\sigma_{\text{gain}} > \sigma_{\text{loss}}$, so the original values decrease to meet their target values where necessary, while the original values that attempt to increase toward their target values have their increase limited.

where

$$
\begin{aligned}
A &= 2\Delta E_i \Delta\rho_i - ||\Delta(\rho\vec{u})_i||^2 - 2(e_{\min})_i(\Delta\rho_i)^2, \\
B &= 2\rho_i \Delta E_i + 2E_i\Delta\rho_i - 2(\rho\vec{u})_i \cdot \Delta(\rho\vec{u})_i - 4(e_{\min})_i\rho_i\Delta\rho_i, \\
C &= 2E_i\rho_i - ||(\rho\vec{u})_i||^2 - 2(e_{\min})_i(\rho_i)^2 = 2(\rho_i)^2(e_i - (e_{\min})_i).
\end{aligned}
$$

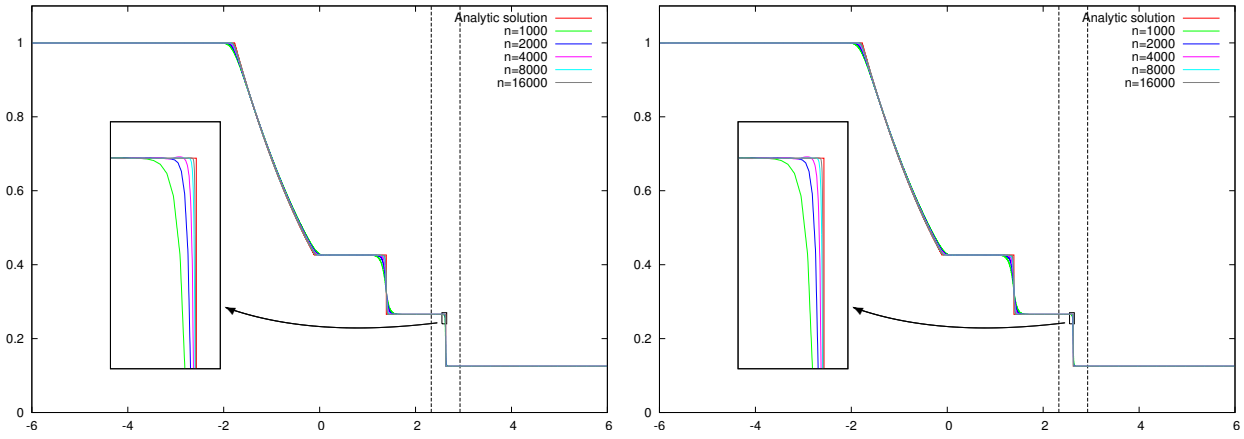
The second equality in the definition of C is obtained using $E_i = \rho_i e_i + \frac{||(\rho\vec{u})_i||^2}{2\rho_i}$. This, along with Equation 38, shows that C is strictly positive and thus a solution for Equation 39 always exists. We solve Equation 39 to find the maximum allowable value of λ_i using the robust method proposed in [52], which is especially important if any of the coefficients approach zero.

Using λ_i to scale down the change of the conserved variables preserves positivity but violates conservation. Thus, we once again reenforce conservation by recomputing σ_{gain} and σ_{loss} and proceeding as above. This entire process can be iteratively repeated, and in practice we have rarely seen the necessity of more than two iterations.

7.2. Numerical results

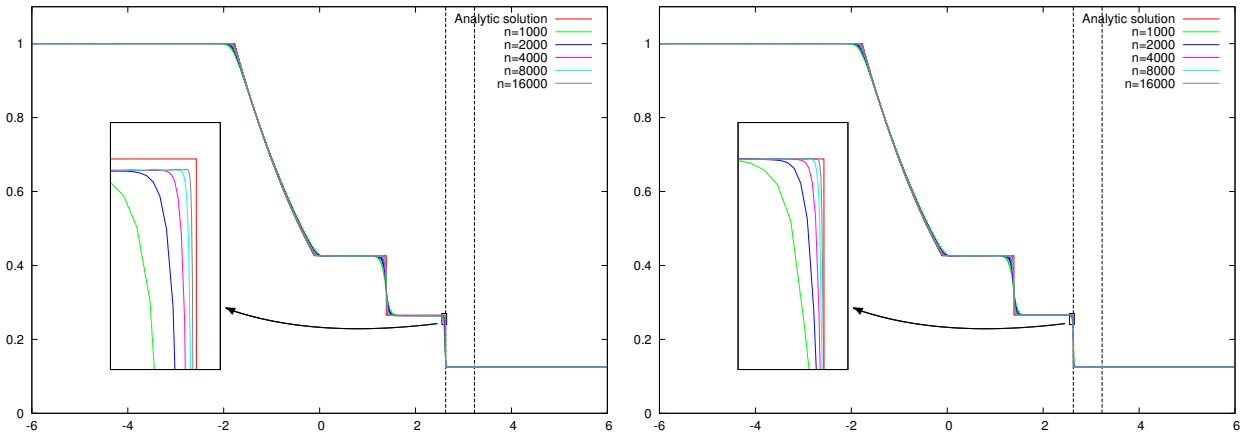
We carried out a number of convergence tests in both one and two spatial dimensions. Here we show a few typical examples for illustrative purposes. In all these examples, we use the second order accurate ENO-LLF scheme for advection followed by the implicit pressure update and the implicit high order diffusion.

First, consider a standard one-dimensional Sod shock tube test with the initial conditions (ρ, u, p) equal to $(1, 0, 1)$ when $x \leq 0$ and $(.125, 0, .1)$ when $x > 0$. There are two overlapping grids with their object space domains and cell sizes listed in Table 3. In Figure 15 the initially stationary finer grid moves with a velocity of 1.7522 when $t \geq .329$ so that the shock front remains interior to that grid. In Figure 16 the finer grid starts moving earlier when $t = .158$ so that the shock front is coincident with the left-hand boundary of the grid. While both methods converge when the shock is kept interior to the finer grid, linear interpolation does not adequately converge when the shock front is aligned with the left-hand grid boundary. Similar results were obtained if the finer grid begins to move at a later time in order to keep the shock coincident with its right-hand boundary. In a second series of tests, we used an underlying leftward moving fluid velocity in order to create a stationary Sod shock. The finer grid was placed such that the shock front was on its left-hand boundary, right-hand boundary, or interior. The proposed conservative remapping method converged in all three cases.



(a) Linear interpolation (b) Conservative remapping

Figure 15: Convergence results of the density field for the one-dimensional Sod shock tube test at time $t = 1.5$. The dashed lines indicate the boundaries of the finer grid.



(a) Linear interpolation (b) Conservative remapping

Figure 16: Convergence results of the density field for the one-dimensional Sod shock tube test at time $t = 1.5$. The dashed lines indicate the boundaries of the finer grid.

	Object space domain	Δx	s
1	$[-6, 6]$	$12/n$	0
2	$[-.3, .3]$	$6/n$	.576

Table 3: The object space domains, cell sizes, and initial positions of the two grids used in the one-dimensional Sod shock tube test. n indicates the number of cells on the background grid.

Next, consider a two-dimensional Sod shock tube test with the same initial conditions and two overlapping grids as specified in Table 4. In Figure 17 (top), we show the convergence results holding the finer grid stationary as the shock wave passes over it. The rest of Figure 17 shows the results when the finer grid moves in order to keep the shock front aligned with the center, left-hand boundary, or right-hand boundary similar to Figures 15 and 16 except that the finer grid does not extend all the way to the top and bottom boundary of the coarser grid illustrating behaviors tangential to a shock that crosses from the finer to the coarser grid. All the simulations converge as expected. Figure 18 rotates the finer grid to an angle of $\pi/3$ and either holds it fixed or translates it with the velocity of the shock. Again, the simulations converge as expected, although one can see a small kink near the transition from the finer to the coarser grid on the lower resolution simulations.

	Object space domain	Δx	$\vec{s}$
1	$[-4, 4] \times [-.6, .6]$	$8/n$	$(0, 0)$
2	$[-.3, .3] \times [-.3, .3]$	$4/n$	$(.576, 0)$

Table 4: The object space domains, cell sizes, and initial positions of the two grids used in the two-dimensional Sod shock tube test. n indicates the number of cells in x-direction on the background grid.

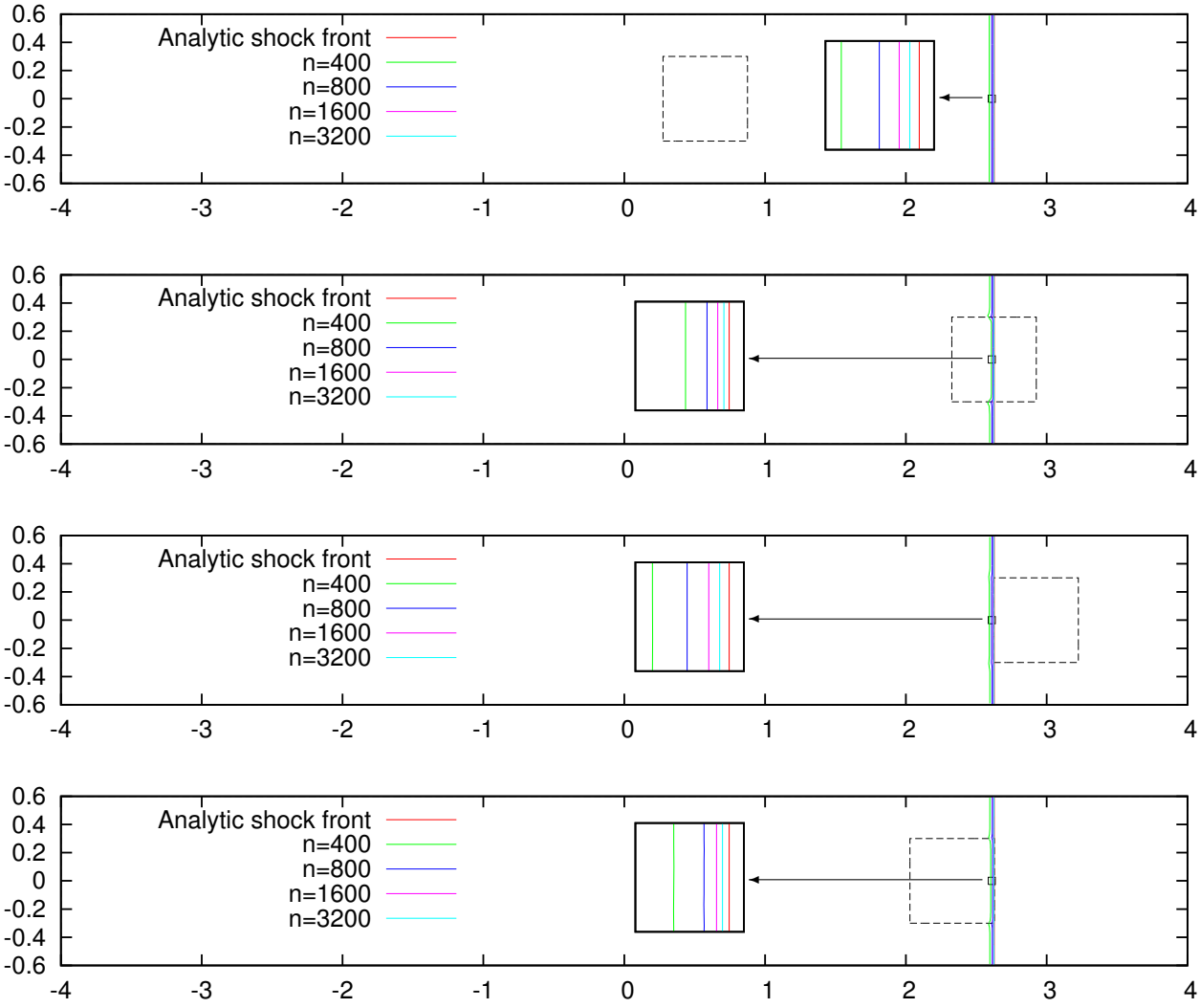


Figure 17: Convergence of the $\rho = .25$ density contour for the two-dimensional Sod shock tube test at time $t = 1.5$. This density value is between the postshock density .2656 and the preshock density .125, and thus its contour should converge to the location of the analytic shock front. The dashed outline indicates the fine grid boundary at time $t = 1.5$.

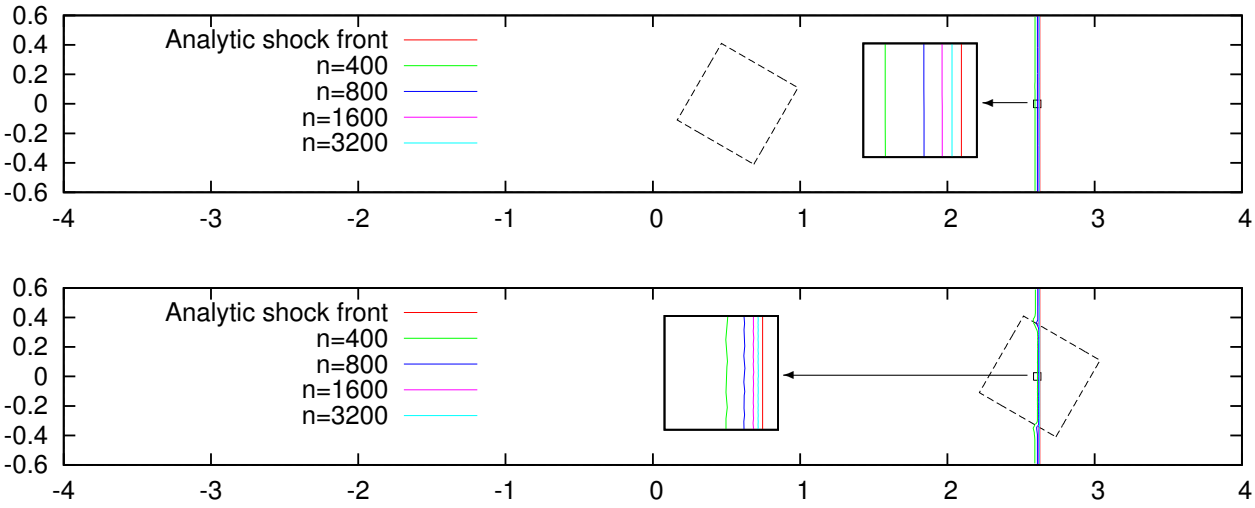


Figure 18: Convergence of the $\rho = .25$ density contour for the two-dimensional Sod shock tube test at time $t = 1.5$. This density value is between the postshock density .2656 and the preshock density .125, and thus its contour should converge to the location of the analytic shock front. The dashed outline indicates the fine grid boundary at time $t = 1.5$.

8. Examples

At this point we pause to present a few examples before considering two-way solid fluid coupling in Section 9. We solve the symmetric positive-definite systems for both the pressure and the high order diffusion via the preconditioned conjugate gradient method with diagonal preconditioners. In the examples below, all quantities are in SI units, and gravity is set to zero.

We rasterize the various stationary solid walls and domain boundaries onto the grids by classifying grid cell centers as inside the fluid, inside one of the solids, or outside the domain, where we denote the grid cells inside the fluid as fluid grid cells and the grid cells inside the solids as solid grid cells. The fluid state variables in grid cells that are inside the solids or outside the domain are filled with appropriate boundary conditions before advection. When discretizing the pressure terms using the Voronoi mesh, we rasterize the various stationary solid walls and domain boundaries onto the Voronoi mesh by classifying Voronoi cell centers in a similar fashion. The pressure degrees of freedom are only defined in the Voronoi cells whose centers are inside the fluid. Note that when interpolating pressure from the Voronoi mesh to removed Cartesian grid cells, we discard the cells that are inside the solids or outside the domain from the interpolation stencils and renormalize the weights.

We have noticed that artificial oscillations occasionally originate from solid boundaries which are overlapped and rasterized differently by different grids. This is remedied by identifying fluid grid cells adjacent to solid grid cells and updating their values using interpolation from the updated values of momentum and energy on the contiguous Voronoi mesh. Note that the Voronoi momentum and energy is updated via Equations 26 and 28. Additionally, these grid cells do not participate in the conservative remapping discussed in section 7, but rather have their values directly replaced with the target values from the Voronoi mesh.

8.1. Two interacting blast waves

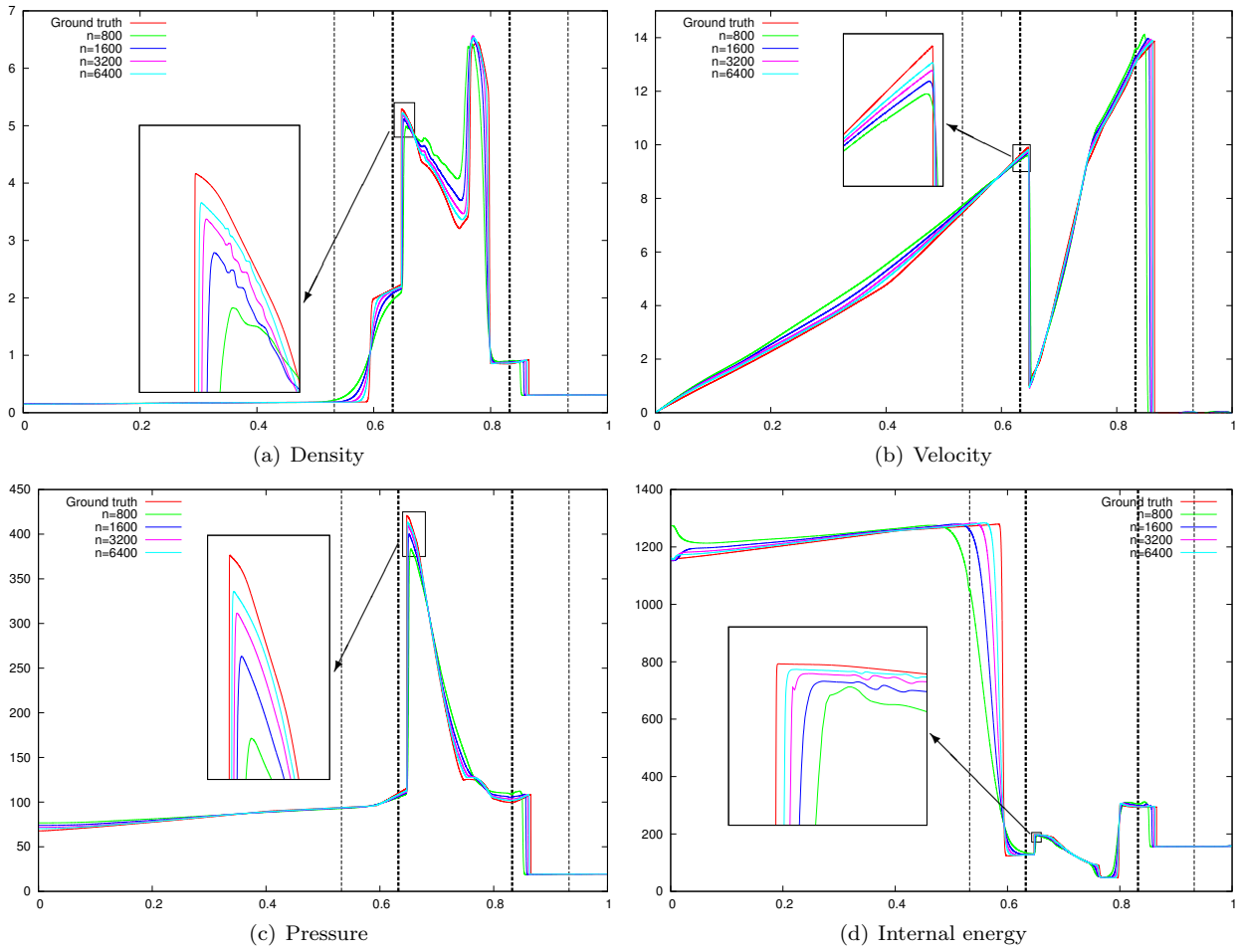
Consider the one-dimensional test of two interacting blast waves introduced by [70] which involves multiple strong shocks. The initial conditions are specified as

$$(\rho(x), u(x), p(x)) = \begin{cases} (1, 0, 10^3) & \text{if } 0 \leq x < .1, \\ (1, 0, 10^{-2}) & \text{if } .1 \leq x < .9, \\ (1, 0, 10^2) & \text{if } .9 \leq x < 1. \end{cases} \tag{40}$$

Standard reflective solid wall boundary conditions are applied to both the left and the right boundaries. There are three overlapping grids listed in Table 5, where the two finer grids are moving. We use the second order accurate ENO-LLF scheme for advection on all grids with a global CFL number of 3, and subdivide the time steps during advection. Figure 19 shows that the solutions correctly converge to a ground truth result computed using a single static grid at resolution $n = 40{,}000$ and third order accurate ENO-LLF with a CFL number of .5. Figure 20 shows the results using the conservative semi-Lagrangian scheme and a global CFL number of 3, where we do not need to subdivide the time steps during advection.

	Object space domain	Δx	s
1	$[0, 1]$	$1/n$	0
2	$[.3, .7]$	$.5/n$	$.25\sin(10\pi t)$
3	$[.4, .6]$	$.25/n$	$.25\sin(10\pi t)$

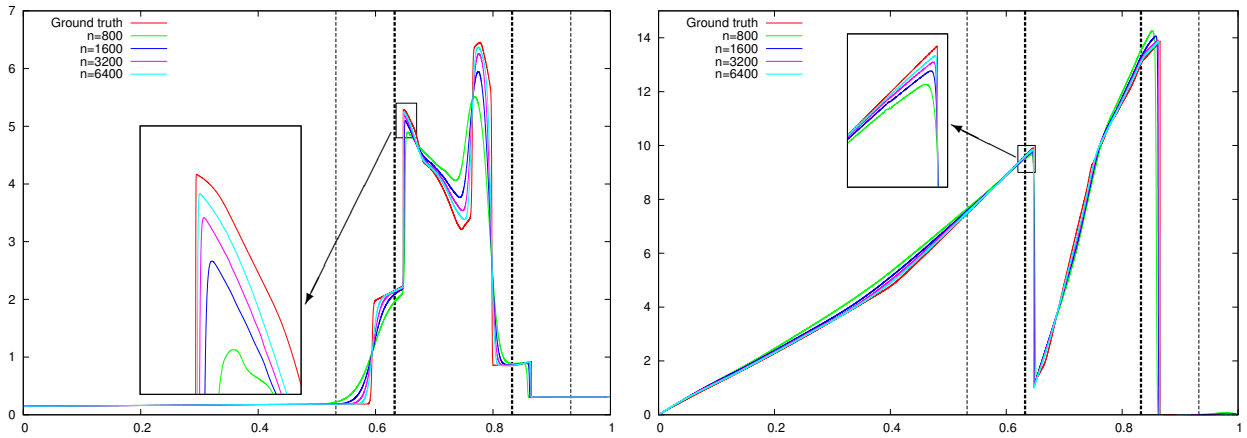
Table 5: The object space domains, cell sizes, and positions of the grids used in the two interacting blast waves example. n indicates the number of grid cells on the background grid. In this example, the two finer grids are not meant to track features, but are merely meant to test the efficacy of our method in terms of complex features crossing grid boundaries and vice versa.



(a) Density (b) Velocity

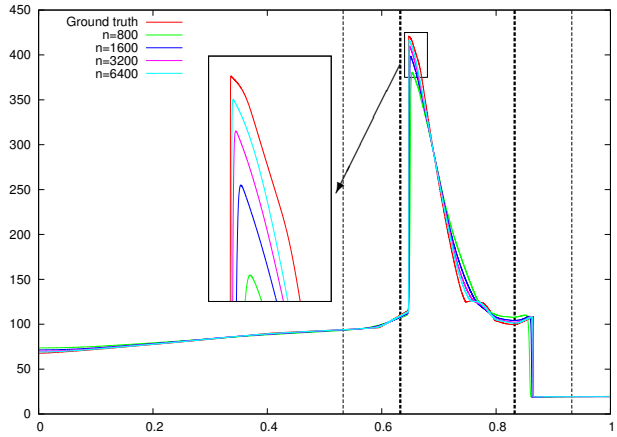
(c) Pressure (d) Internal energy

Figure 19: Convergence results for the two interacting blast waves at time $t = .038$, where the inner two thicker dashed lines indicate the boundaries of grid 3 and the outer two thinner dashed lines indicate the boundaries of grid 2. The second order accurate ENO-LLF scheme is used for advection.

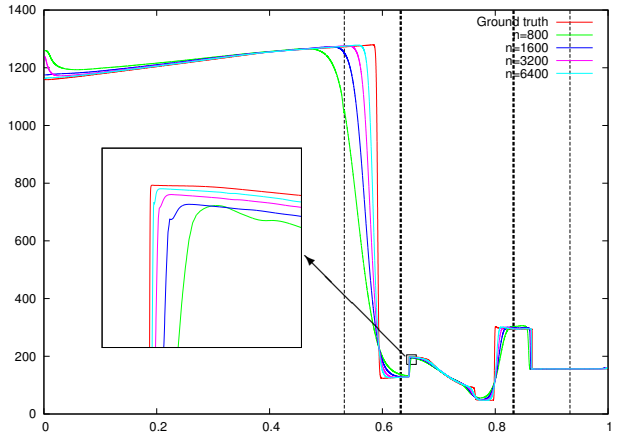


(a) Density

(b) Velocity



(c) Pressure



(d) Internal energy

Figure 20: Convergence results for the two interacting blast waves at time $t = .038$, where the inner two thicker dashed lines indicate the boundaries of grid 3 and the outer two thinner dashed lines indicate the boundaries of grid 2. The conservative semi-Lagrangian scheme is used for advection.

8.2. Isentropic Vortex

Consider the two-dimensional test of an inviscid vortex in a free stream as described in [71]. The free stream conditions are specified as density $\rho_\infty = 1$, velocity $(u_\infty, v_\infty) = (1, 0)$, and pressure $p_\infty = 1$. As an initial condition, an isentropic vortex with no perturbation in entropy is added to the free stream, so that the analytic solution is the pure advection of the vortex at the free stream velocity. In a computational domain of $[-25, 25] \times [-25, 25]$, the perturbation values are given by

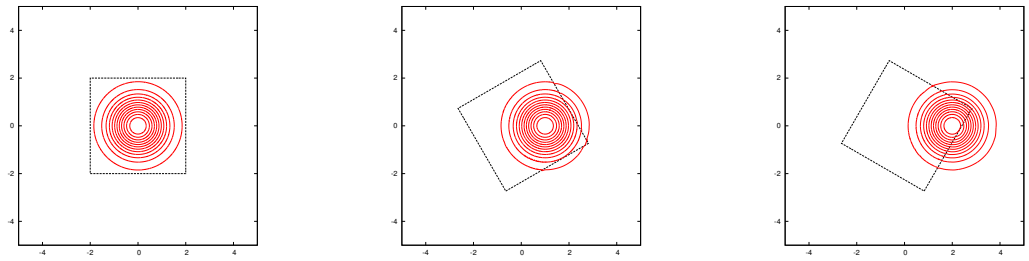
$$(\delta u, \delta v) = \frac{\beta}{2\pi} e^{(1-r^2)/2} (-y, x), \tag{41}$$

$$\delta T = -\frac{(\gamma - 1)\beta^2}{8\gamma\pi^2} e^{1-r^2}, \tag{42}$$

where $r^2 = x^2 + y^2$, $\beta = 5$ is the vortex strength, and T is temperature. Since the entire flow field is isentropic, $T = \frac{p}{\rho}$ and $p = \rho^\gamma$ for the ideal gas. There are two overlapping grids with the finer one rotating as listed in Table 6. We use the second order accurate ENO-LLF scheme for advection on both grids with a global CFL number of 3, and subdivide the time steps during advection. Figure 21 shows the snapshots of the simulation, and Table 7(a) shows the convergence of the density field as compared to the analytic solution. Next, we use the conservative semi-Lagrangian scheme for advection and a global CFL number of 3, where we do not need to subdivide the time steps during advection, and Table 7(b) shows the convergence of the density field as compared to the analytic solution.

	Object space domain	Δx	$\vec{s}$	θ
1	$[-25, 25] \times [-25, 25]$	$50/n$	$(0, 0)$	0
2	$[-2, 2] \times [-2, 2]$	$16/n$	$(.1 \sin t, 0)$	$\frac{\pi}{6} t$

Table 6: The object space domains, cell sizes, positions, and orientation angles of the grids used in the isentropic vortex tests. n indicates the number of grid cells in one spatial dimension of the background grid.



(a) t=0 (b) t=1 (c) t=2

Figure 21: The density contour plots for the isentropic vortex test on two overlapping grids with grid resolution $n = 3200$ using the second order accurate ENO-LLF scheme for advection on both grids and a global CFL number of 3. The dashed outline indicates the boundary of the rotating fine grid. 12 contours are plotted within the density value range $[.54, .98]$.

8.3. Double mach reflection of a strong shock

Consider the two-dimensional test introduced in [70] which involves double Mach reflection of a strong shock. In a computational domain of $[0, 4] \times [0, 1]$, a planar Mach 10 shock hits a reflective solid wall boundary that lies along the bottom of the domain, i.e. $y = 0$, in the range of $x \in [1/6, 4]$. The plane of the shock front is initially incident to the reflective solid wall at $(1/6, 0)$ with a angle of $\pi/3$ with respect to the x-direction. The left and bottom (for $x \in (0, 1/6)$) boundary conditions are given by the postshock condition, and a standard zero-gradient outflow boundary condition is applied at the right boundary. The

	(a)		(b)	
n	Error	Order	Error	Order
200	4.03×10^{-3}	-	9.78×10^{-3}	-
400	1.61×10^{-3}	1.32	5.11×10^{-3}	.94
800	6.74×10^{-4}	1.26	2.60×10^{-3}	.98
1600	3.17×10^{-4}	1.09	1.30×10^{-3}	1.00
3200	1.61×10^{-4}	.98	6.49×10^{-4}	1.00

Table 7: The convergence of the density field at time $t = 2$ for the isentropic vortex test on two overlapping grids. The errors are computed by averaging the pointwise errors at 1000×1000 uniformly spacing points in the domain $[-3, 7] \times [-5, 5]$. (a) The second order accurate ENO-LLF scheme is used for advection. (b) The conservative semi-Lagrangian scheme is used for advection.

top boundary conditions are set to describe the exact motion of the Mach 10 shock. The initial conditions are specified as

$$(\rho(x,y), \vec{u}(x,y), p(x,y)) = \begin{cases} (1.4, \vec{0}, 1) & \text{preshock}, \\ (8, 8.25\vec{n}, 116.5) & \text{postshock}, \end{cases} \tag{43}$$

where $\vec{n}$ denotes the unit vector in the initial shock speed direction. Table 8 shows the six moving grids used to track important features of the flow. The results obtained using a grid resolution of $n = 240$ and a global CFL number of 3 are shown in Figure 22, which compares well with [70]. Figure 23 shows convergence results of the $\rho = 9$ density contour as compared to a ground truth computed using a single static grid of resolution $n = 960$ and third order accurate ENO-LLF with a CFL number of .5.

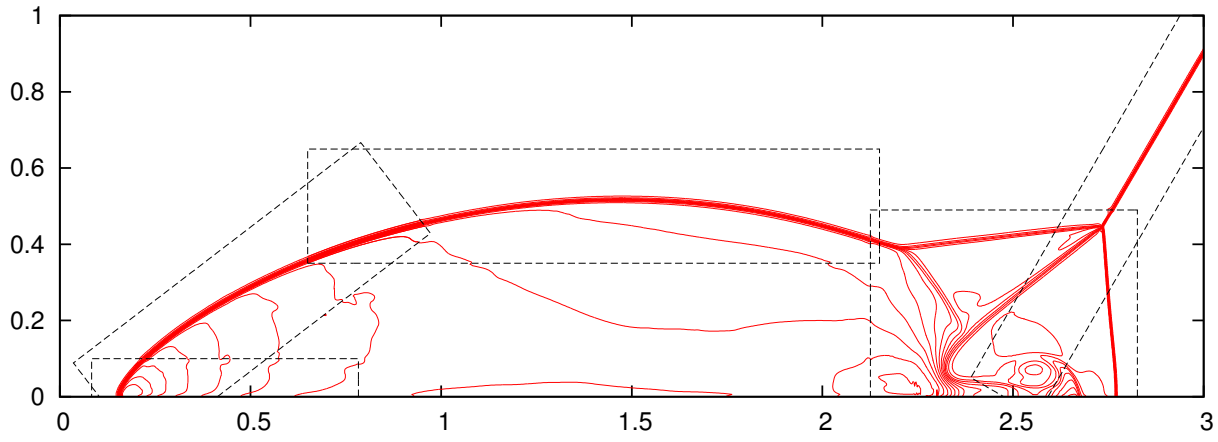
	Object space domain	Δx	$\vec{s}$	θ
1	$[-.2, 4.2] \times [-.2, 1.2]$	$1/n$	$(0, 0)$	0
2	$[0, 1.2] \times [-.1, .1]$	$.5/n$	$(\frac{1}{6} + \frac{20\sqrt{3}}{3}t, 0)$	$\frac{\pi}{3}$
3	$[-.35, .35] \times [-.25, .25]$	$.4/n$	$(\frac{1}{6} + \frac{20\sqrt{3}}{3}t, .24)$	0
4	$[-.05, .9] \times [-.15, .15]$	$.5/n$	$(\frac{1}{6}, 0)$	$\frac{25\pi}{18}t$, $t \leq .15$ $\frac{5\pi}{24}$, $t > .15$
5	$[-.75, .75] \times [-.15, .15]$	$.5/n$	$(1.4 + 2.5t, 0)$	0
6	$[-.35, .35] \times [-.1, .1]$	$.4/n$	$(\frac{13}{30}, 0)$	0

Table 8: The object space domains, cell sizes, positions, and orientations of the grids used in the double Mach reflection of a strong shock example. n indicates the number of grid cells in one axial direction in a length of 1 on the background grid.

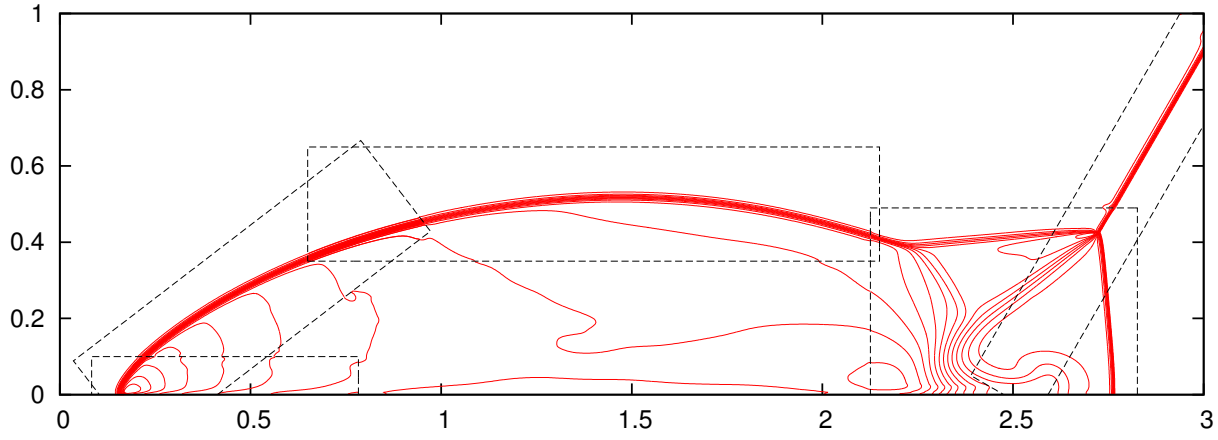
8.4. Mach 3 wind tunnel with a step

Consider a two-dimensional Mach 3 wind tunnel with a solid step as described in [70, 20]. The computational domain is $[0, 3] \times [0, 1]$, and the solid step is .2 units high located at the bottom of the domain .6 units from the left boundary. Initially, the wind tunnel is filled with gas with initial conditions $\rho = 1.4$, $p = 1$, and $\vec{u} = (3, 0)$, and gas with these conditions is continually fed in the left boundary while the right boundary is set to a zero-gradient outflow boundary condition. Standard reflective solid wall boundary conditions are applied to the top and bottom boundaries of the computational domain. We applied the entropy and enthalpy fix around the solid step corner as in [70] and the isobaric fix near solid boundaries as in [20]. In this example, the grid motions are specified manually, and Table 9 lists the locations of the 9 grids at time $t = 4$.

The results obtained using the second order accurate ENO-LLF scheme are shown in Figure 24, which compares well to [70]. Figure 26 shows the convergence of the $\rho = 4$ density contour at time $t = 4$ as



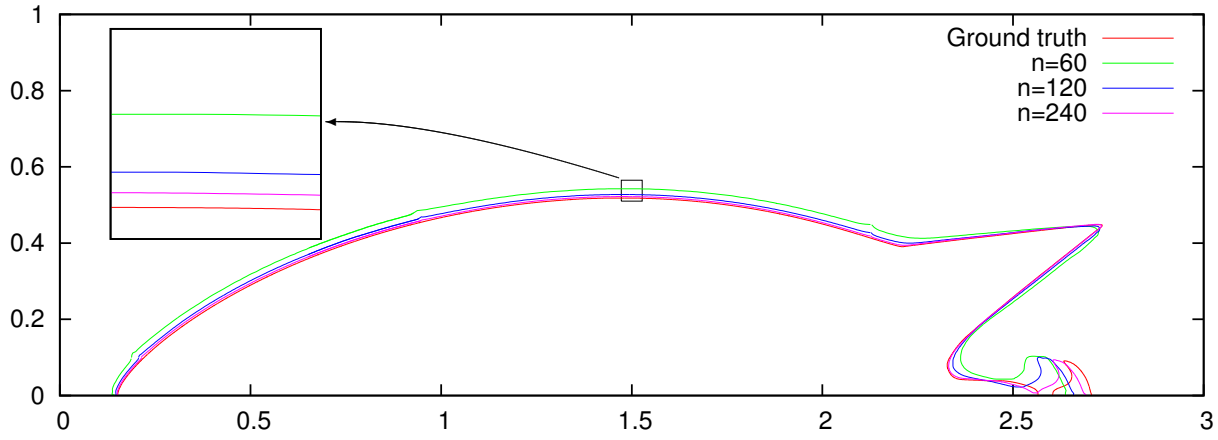
(a) ENO-LLF



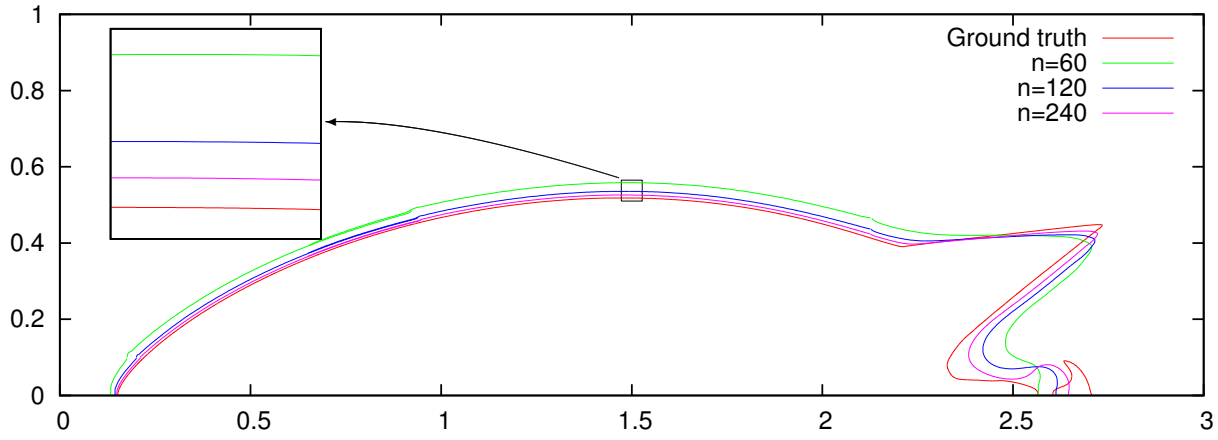
(b) Conservative semi-Lagrangian scheme

Figure 22: The density contour plots for the double Mach reflection of a strong shock example at time $t = .2$, where the dashed outlines indicate the grid boundaries. 30 contours are plotted within the range $[1.731, 20.92]$.

compared to the ground truth computed using a single static grid at resolution $n = 1280$ and third order accurate ENO-LLF with a CFL number of .5. The results obtained using the conservative semi-Lagrangian scheme are shown in Figure 25. The shock reflection off the solid step near $x = 1.25$ in Figure 25(a) is problematic, and we note that this behavior not only subsides when reducing the CFL number to .5 but also seems to depend on the placement of the grids with regards to the difficult corner. Shrinking the fine grid to uncover the corner in Figure 25(b) alleviates the issues. The convergence result is shown in Figure 26(c).



(a) ENO-LLF

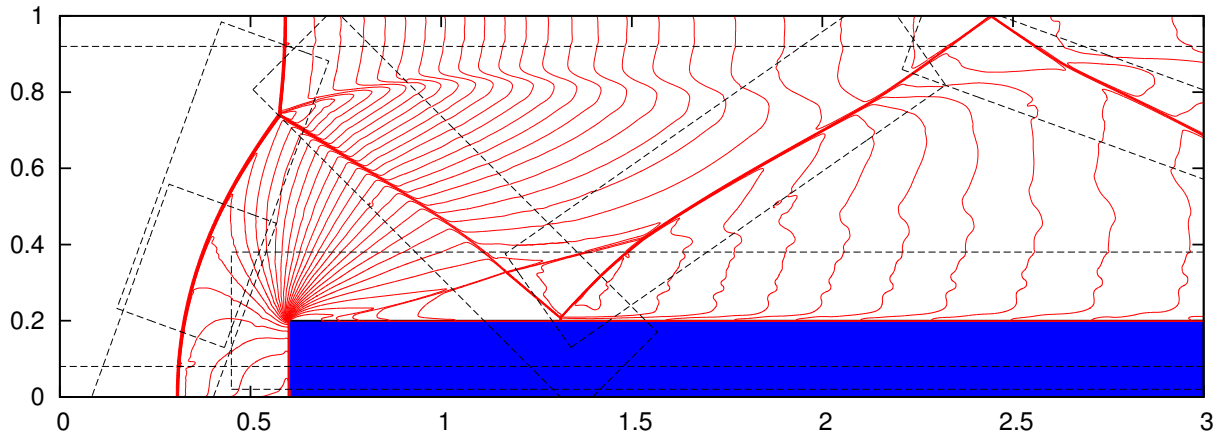


(b) Conservative semi-Lagrangian scheme

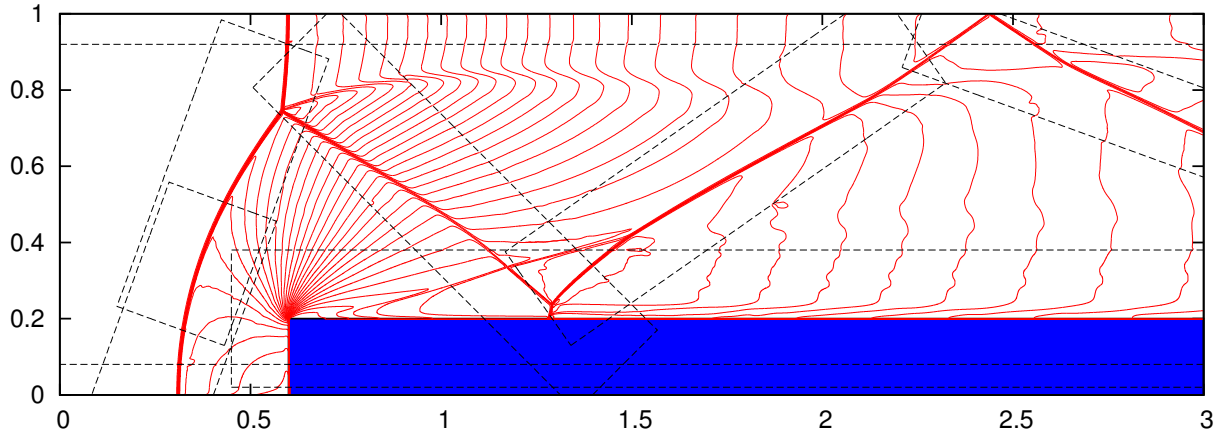
Figure 23: Convergence of the $\rho = 9$ density contour for the double Mach reflection of a strong shock example at time $t = .2$.

	Object space domain	Δx	$\vec{s}$	θ
1	$[-.2, 3.2] \times [-.2, 1.2]$	$1/n$	$(0, 0)$	0
2	$[-.1, 3.1] \times [-.08, .08]$	$.356/n$	$(0, 1)$	0
3	$[-.1, 3.1] \times [-.08, .08]$	$.356/n$	$(0, 0)$	0
4	$[-.1, .7] \times [-.15, .15]$	$.4/n$	$(.325, .275)$	1.22
5	$[0, 1.2] \times [-.15, .15]$	$.4/n$	$(.612, .913)$	$-.785$
6	$[-.3, .9] \times [-.15, .15]$	$.4/n$	$(1.5, .425)$	.611
7	$[-.11, .77] \times [-.11, .11]$	$.44/n$	$(2.35, .925)$	$-.349$
8	$[-.3, .3] \times [-.15, .15]$	$.4/n$	$(.325, .225)$	1.22
9	$[-.15, 2.55] \times [-.18, .18]$	$.24/n$	$(.6, .2)$	0

Table 9: The object space domains, cell sizes, positions, and orientations of the grids at time $t = 4$ in the Mach 3 wind tunnel with a step example. n indicates the number of grid cells in one axial direction in a length of 1 on the background grid.

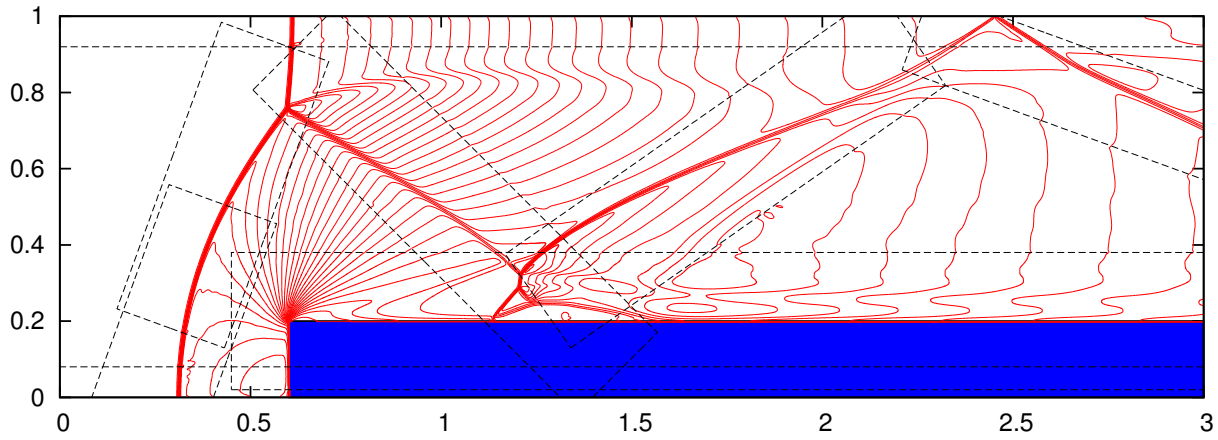


(a) ENO-LLF, CFL=.5

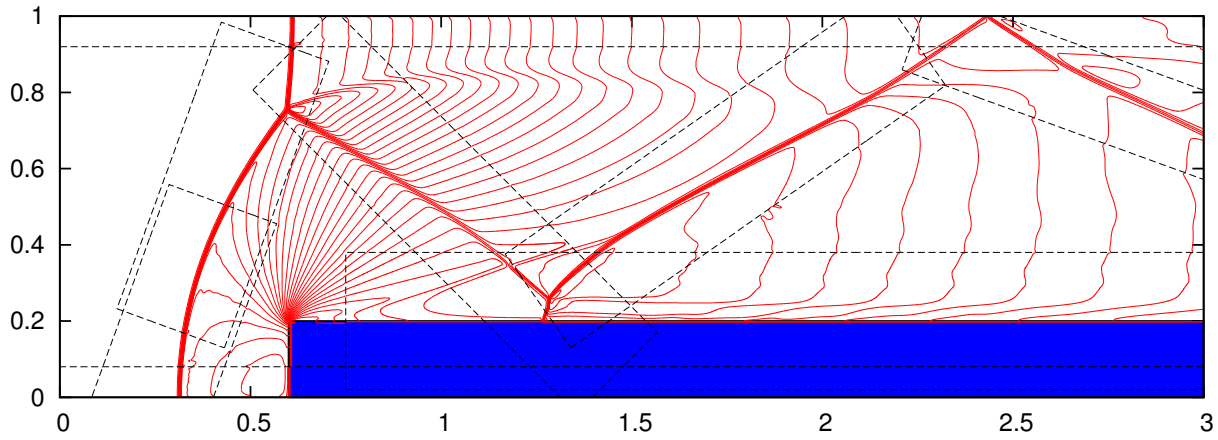


(b) ENO-LLF, CFL=3

Figure 24: The density contour plots for the Mach 3 wind tunnel with a step at time $t = 4$ at grid resolution $n = 320$, where the dashed outlines indicate the grid boundaries. 30 contours are plotted within the range $[.2568, 6.067]$.

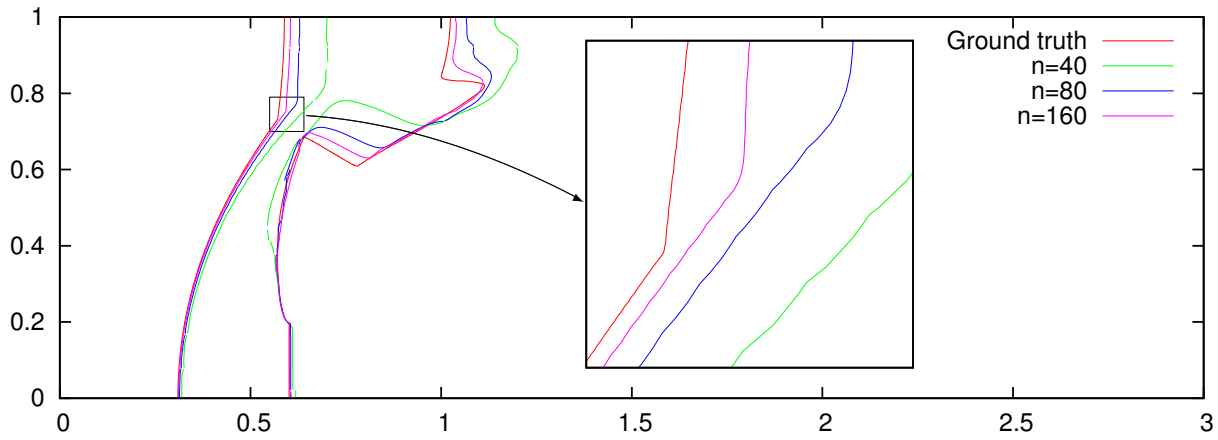


(a) Conservative semi-Lagrangian scheme, CFL=3

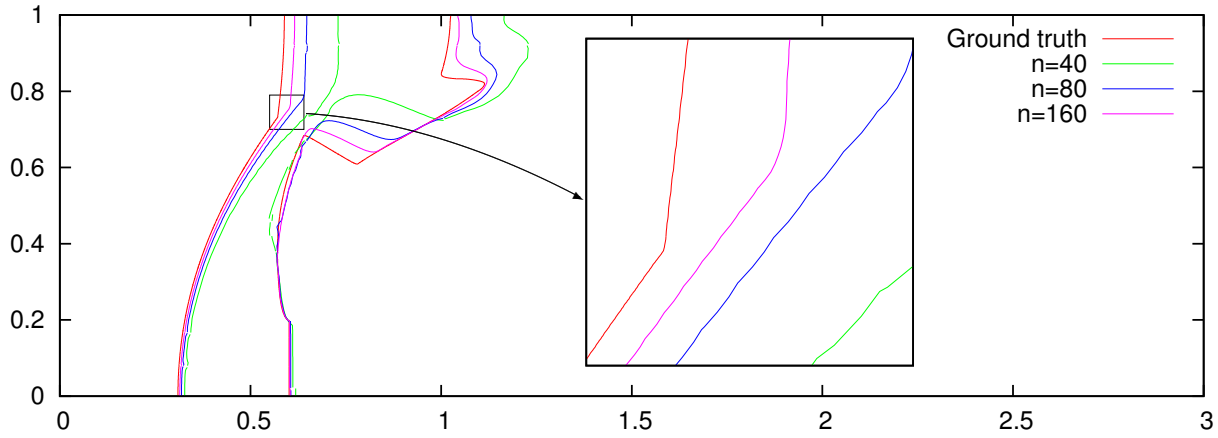


(b) Conservative semi-Lagrangian scheme, CFL=3 (shrinking the fine grid around the solid step)

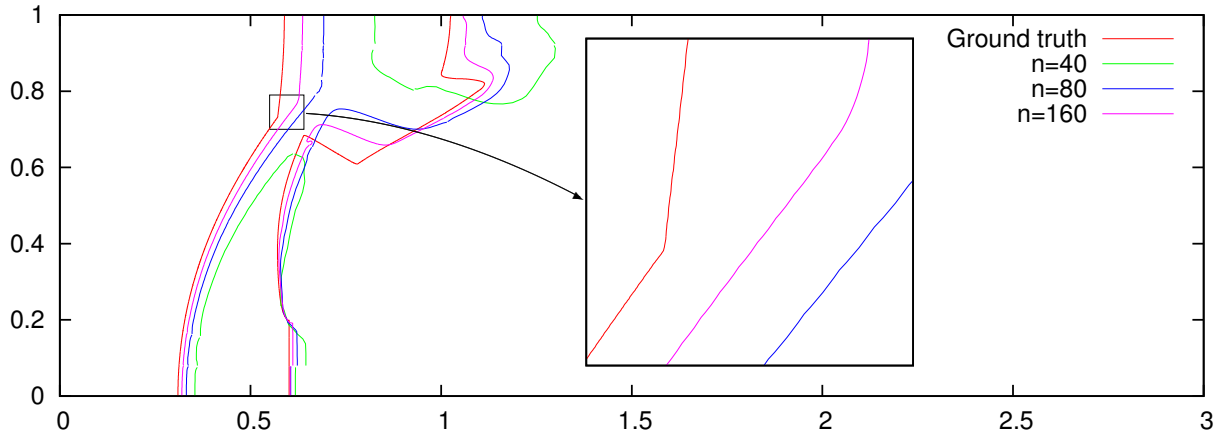
Figure 25: The density contour plots for the Mach 3 wind tunnel with a step at time $t = 4$ at grid resolution $n = 320$, where the dashed outlines indicate the grid boundaries. Note that in (b) the fine grid around the solid step is shrunk so that the corner of the step is exposed to the background grid. 30 contours are plotted within the range $[.2568, 6.067]$.



(a) ENO-LLF, CFL=.5



(b) ENO-LLF, CFL=3



(c) Conservative semi-Lagrangian scheme, CFL=3 (shrinking the fine grid around the solid step)

Figure 26: Convergence of the $\rho = 4$ density contour for the Mach 3 wind tunnel with a step at time $t = 4$.

9. Two-Way Solid Fluid Coupling

We adapt the two-way solid fluid coupling approach for compressible flow from [24, 23] to our Chimera grid framework. A deformable body can be described by vectors of nodal positions $\mathbf{x}(t)$ and velocities $\mathbf{v}(t)$. The evolution of positions and velocities satisfies Newton's second law

$$\mathbf{x}_t = \mathbf{v} \tag{44}$$
$$\mathbf{M}\mathbf{v}_t = \mathbf{F}(\mathbf{x}, \mathbf{v}), \tag{45}$$

where $\mathbf{M}$ is the solid mass matrix and $\mathbf{F}$ is the vector of all forces acting on the solid. We discretize and compute elastic and body forces explicitly, and discretize damping terms explicitly in position but implicitly in velocity. This results in

$$\mathbf{v}^{n+1/2} = \mathbf{v}^n + \tfrac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^n) + \tfrac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{D}(\mathbf{x}^n)\mathbf{v}^{n+1/2} \tag{46}$$
$$\tilde{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}^{n+1/2} \tag{47}$$
$$(\mathbf{x}^{n+1}, \tilde{\mathbf{v}}^n) = \text{CollisionAndContact}(\tilde{\mathbf{x}}^{n+1}, \mathbf{v}^n) \tag{48}$$
$$\mathbf{v}^{n+1} = \tilde{\mathbf{v}}^n + \Delta t\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^{n+1}) + \Delta t\mathbf{M}^{-1}\mathbf{D}(\mathbf{x}^{n+1})\mathbf{v}^{n+1} \tag{49}$$

where $\mathbf{F}_e(\mathbf{x})$ is the sum of elastic and body forces, and $\mathbf{D}(\mathbf{x})$ is the damping matrix. Collision and contact are handled for deformable and rigid bodies as outlined in [10], [25], and [62].

For a rigid body, we define the generalized position vector as $\mathbf{x} = (\mathbf{x}_{cm}^T, \boldsymbol{\theta}^T)^T$ and the generalized velocity vector as $\mathbf{v} = (\mathbf{v}_{cm}^T, \boldsymbol{\omega}^T)^T$ where $\mathbf{x}_{cm}$ and $\mathbf{v}_{cm}$ are the position and velocity of the center of mass, and $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are the angular position and velocity. The rigid body equivalent of Equation 45 is

$$\left(\begin{bmatrix}\mathbf{M}_r & 0\\ 0 & \mathbf{I}_r\end{bmatrix}\mathbf{v}\right)_t = \begin{bmatrix}\mathbf{f}_{cm}\\ \boldsymbol{\tau}\end{bmatrix}, \tag{50}$$

where $\mathbf{M}_r$ is a 3×3 diagonal matrix with the rigid body mass on the diagonal, and $\mathbf{f}_{cm}$ and $\boldsymbol{\tau}$ are the net force and torque. Note that the inertia tensor $\mathbf{I}_r$ is a function of $\boldsymbol{\theta}$ and thus a function of time t. The rigid body equivalents of Equations 46 – 49 are

$$(\mathbf{M}\mathbf{v})^{n+1/2} = (\mathbf{M}\mathbf{v})^n + \tfrac{\Delta t}{2}\mathbf{F}_e(\mathbf{x}^n) + \tfrac{\Delta t}{2}\mathbf{D}(\mathbf{x}^n)\mathbf{v}^{n+1/2} \tag{51}$$
$$\tilde{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}^{n+1/2} \tag{52}$$
$$(\mathbf{x}^{n+1}, (\widetilde{\mathbf{M}\mathbf{v}})^n) = \text{CollisionAndContact}(\tilde{\mathbf{x}}^{n+1}, (\mathbf{M}\mathbf{v})^n) \tag{53}$$
$$(\mathbf{M}\mathbf{v})^{n+1} = (\widetilde{\mathbf{M}\mathbf{v}})^n + \Delta t\mathbf{F}_e(\mathbf{x}^{n+1}) + \Delta t\mathbf{D}(\mathbf{x}^{n+1})\mathbf{v}^{n+1} \tag{54}$$

Since we keep the solid positions frozen at time t^n when solving Equation 51 and frozen at time t^{n+1} when solving Equation 54, Equations 51 and 54 degenerate into Equations 46 and 49.

After obtaining the time $t^{n+1/2}$ solid velocity $\mathbf{v}^{n+1/2}$, we update the solid positions using Equations 47 and 48 (or Equations 52 and 53) in conjunction with the fluid advection. In each subdivided time step Δt_i for the purpose of advection, we first update the solid positions using Equation 47 (or Equation 52) and Δt_i and then process collision and contact using Equation 48 (or Equation 53) obtaining the solid positions at time $t^n + \sum_{j=1}^{i} \Delta t_j$. Subsequently, we follow [24] filling ghost cells inside solids based on the solid positions at time $t^n + \sum_{j=1}^{i-1} \Delta t_j$ using the reflective boundary condition and an effective pointwise solid velocity which is computed for any point on the solid surface as the displacement of that point from time $t^n + \sum_{j=1}^{i-1} \Delta t_j$ to time $t^n + \sum_{j=1}^{i} \Delta t_j$ divided by Δt_i. See also [40, 69, 42]. These filled ghost cells inside solids provide boundary conditions for advection as well as valid fluid states for populating cells uncovered as solids move. Next, we update all the grids' positions and orientations while solving for fluid advection as discussed in Section 4.

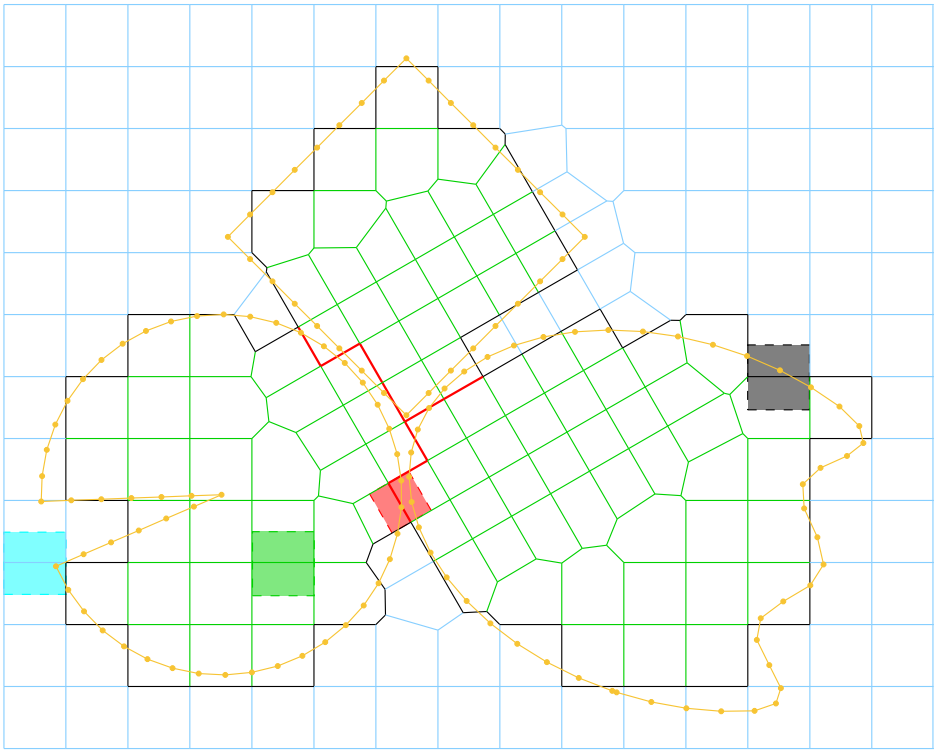


Figure 27: The blue lines indicate faces between two fluid cells. The black lines indicate faces between a solid cell and a fluid cell. The green lines indicate faces between two cells inside the same solid. The red lines indicate faces between two cells inside two different solids. The dual cell of one face of each type is shaded as an example. The dots are the vertices of the solids' surfaces.

In Equations 46 and 49, the $\mathbf{F}_e$ terms are first incorporated explicitly to get an intermediate velocity $\mathbf{v}^*$. Then, instead of solving for the damping terms implicitly as in Equations 46 and 49, we solve a two-way coupled system. The solution procedure for the first two-way coupled system in Equation 46 is the same as that for the second one in Equation 49, except that $\frac{\Delta t}{2}$ is replaced with Δt, $\mathbf{x}^n$ is replaced with $\mathbf{x}^{n+1}$, and the fluid state after advection is used. Thus, we only focus on the formulation of the second two-way coupled system in Section 9.1. After obtaining the time t^{n+1} fluid state and solid velocity, we apply the high order diffusion terms in Section 6 and conservatively remap the values in overlapped regions as discussed in Section 7.

9.1. Two-way coupled system

Extending the method from [24], we rasterize the solids onto the Voronoi mesh by classifying the Voronoi cell centers as inside the fluid or inside one of the solids, resulting in two types of Voronoi cells: fluid cells and solid cells. In addition, there are four types of Voronoi faces: fluid-fluid faces, solid-fluid faces, faces inside a single solid (both adjacent Voronoi cell centers are inside the same solid), and faces between two solids (the adjacent Voronoi cell centers are inside different solids). See Figure 27. On each solid-fluid face a coupling constraint enforces equality between the face normal component of the fluid velocity and the face normal component of the solid velocity interpolated to the center of that Voronoi face. This can be described as

$$\mathbf{W}\mathbf{u}_f^{n+1} - \mathbf{J}\mathbf{v}^{n+1} = 0 \tag{55}$$

where $\mathbf{W}$ is a matrix of zeros and ones that picks out the solid fluid coupling faces, and $\mathbf{J}$ can be decomposed into

$$\mathbf{J} = \mathbf{W}\mathbf{N}\hat{\mathbf{J}}\mathbf{R} \tag{56}$$

where $\mathbf{R}$ samples the solids' velocity, $\hat{\mathbf{J}}$ maps these samples to the center of the cell faces, and $\mathbf{N}$ uses face normals to extract the normal component of the velocity. For deformable bodies, $\mathbf{R} = \mathbf{I}$ since we use the surface particles of deformable bodies as velocity samples. For a rigid body, the velocity of a rigid body b at a sample point p is given by $\mathbf{v}_p = \mathbf{v}_b + [\mathbf{r}_{pb}]_\times^T \boldsymbol{\omega}_b$, where $\mathbf{r}_{pb}$ is the offset from the rigid body center of mass to the point p where the velocity is measured and $[\mathbf{r}_{pb}]_\times$ is the skew-symmetric matrix such that $[\mathbf{r}_{pb}]_\times \boldsymbol{\omega} = \mathbf{r}_{pb} \times \boldsymbol{\omega}$. Thus, the sub-matrix $\mathbf{R}_{pb} = \begin{pmatrix} \mathbf{I} & [\mathbf{r}_{pb}]_\times^T \end{pmatrix}$ maps from the six degree of freedom velocity of a rigid body ($\mathbf{v}_b$ and $\boldsymbol{\omega}_b$) to the sample point p at the center of a Voronoi face.

For deformable bodies, $\hat{\mathbf{J}}$ is an interpolation operator interpolating from surface particles of deformable bodies to Voronoi faces. Each row of $\hat{\mathbf{J}}$ corresponds to one Voronoi face and is constructed by clipping surface triangles of nearby solids into the dual cell of the Voronoi face, computing weights as the orthogonally projected areas of the subpolygons resulting from clipping, normalizing weights, and assigning weights to related surface particles. We refer readers to [58] for a detailed description of this process. For rigid bodies, each row of $\hat{\mathbf{J}}$ is constructed by computing weights using the same method as was used for deformable bodies except that weights are assigned to rigid bodies instead of surface particles. If a dual cell intersects only a single rigid body, then the row of $\hat{\mathbf{J}}$ corresponding to that dual cell has only a single nonzero value of 1 corresponding to the velocity sample of that rigid body at the center of the Voronoi face. If multiple rigid bodies intersect a dual cell, then there will be multiple nonzero entries in that row of $\hat{\mathbf{J}}$ with each corresponding to a rigid body velocity sample at the Voronoi face center. That is, $\hat{\mathbf{J}}$ interpolates a velocity from the multiple rigid body velocities sampled at the center of the Voronoi face. If a dual cell intersects both rigid and deformable bodies, $\hat{\mathbf{J}}$ is an interpolation operator that considers all relevant deformable body surface particles as well as all relevant rigid bodies with their velocity point-sampled to the center of the Voronoi face. Since the dual cell of a Voronoi face can be complex, we instead use a Cartesian approximation when computing $\hat{\mathbf{J}}$. This approximation is centered at the Voronoi face center defined as the midpoint of the line segment connecting the centers of the two adjacent Voronoi cells. Although this approximation still extends from cell center to cell center, the cross sectional shape is approximated as a square with an area identical to the area of the face.

Denoting $\boldsymbol{\lambda}$ as the impulse transfer between the solid and the fluid, the fluid momentum update in Equation 2 becomes

$$\boldsymbol{\beta}\mathbf{u}_f^{n+1} = \boldsymbol{\beta}\mathbf{u}_f^* - \mathbf{G}\hat{\mathbf{p}}^{n+1} + \mathbf{W}^T\boldsymbol{\lambda}, \tag{57}$$

where $\mathbf{G}$ is the volume weighted gradient operator as defined in Section 5 with solid cell columns dropped, and the solid momentum update becomes

$$\mathbf{M}\mathbf{v}^{n+1} = \mathbf{M}\mathbf{v}^* + \Delta t\mathbf{D}\mathbf{v}^{n+1} - \mathbf{J}^T\boldsymbol{\lambda}, \tag{58}$$

where $\mathbf{J}^T$ conservatively distributes the impulse $\boldsymbol{\lambda}$ defined on the solid fluid coupling faces to the solid velocity degrees of freedom. Assembling Equations 57 and 58 with Equation 55 and the matrix form of Equation 5 yields the following system of equations

$$\begin{bmatrix} \mathbf{N} + \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & 0 \\ -\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{W}^T + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T & \mathbf{J}\mathbf{M}^{-1}\mathbf{C}^T \\ 0 & \mathbf{C}\mathbf{M}^{-1}\mathbf{J}^T & \mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T \end{bmatrix} \begin{bmatrix} \hat{\mathbf{p}}^{n+1} \\ \boldsymbol{\lambda} \\ \hat{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{N}\hat{\mathbf{p}}^a + \mathbf{G}^T\mathbf{u}_f^* \\ \mathbf{J}\mathbf{v}^* - \mathbf{W}\mathbf{u}_f^* \\ \mathbf{C}\mathbf{v}^* \end{bmatrix} \tag{59}$$

where $\hat{\mathbf{v}} = \mathbf{C}\mathbf{v}^{n+1}$, and $\mathbf{C}$ is computed by a factorization $-\Delta t\mathbf{D} = \mathbf{C}^T\mathbf{C}$ as in [59]. This is a symmetric positive-definite (SPD) system that is invertible via the preconditioned conjugate gradient method.

In order to update the momentum and energy in a fluid Voronoi cell, we apply the method in Section 5 with modifications to account for the impulses $\boldsymbol{\lambda}$ at solid-fluid faces. Take face j_4 in Figure 9 as an example

assuming cell i_1 is a fluid cell and cell i_2 is a solid cell. At face j_4, the fluid transfers an impulse of $-\lambda_{j_4}\vec{e}_{j_4}$ to the solid. Thus in order to conserve momentum, we modify Equation 26 replacing $-\Delta t p_{j_4}^{n+1} A_{j_4} \vec{n}_{j_4}^{i_1}$ in the summation by $\lambda_{j_4}\vec{e}_{j_4}$. In order to conserve energy, we first note that $s_{j_4}^{i_1} = \vec{n}_{j_4}^{i_1} \cdot \vec{e}_{j_4}$, so that we can similarly replace $-\Delta t p_{j_4}^{n+1} A_{j_4} \vec{n}_{j_4}^{i_1}$ in Equation 28 with $\lambda_{j_4}\vec{e}_{j_4}$. In addition, along the lines of [24], we replace $u_{j_4}^{n+1}$ in Equation 28 with $(v_{j_4}^n + v_{j_4}^{n+1})/2$ where v_{j_4} is the face normal component of the interpolated solid velocity at the center of face j_4.

In order to preserve conservation on each Cartesian grid including the background grid, we interpolate pressure from the Voronoi mesh to removed Cartesian grid cells as well as one layer of ghost cells discarding the cells inside the solids or outside the domain from the interpolation stencils and renormalizing the weights, and then update the momentum and energy on each Cartesian grid. Moreover, as in Section 8 we again interpolate the momentum and energy from the Voronoi mesh to a lower dimensional set of fluid grid cells that are adjacent to solid grid cells, and do not include these fluid grid cells (that are adjacent to solid grid cells) in the conservative remapping.

9.2. Numerical results

We solve Equation 59 via the preconditioned conjugate gradient method with a diagonal preconditioner. In the examples below, all quantities are in SI units, and gravity is set to zero.

9.2.1. Rigid cylinder lift-off

In this example, suggested by [16, 21, 3, 24], a rigid cylinder initially at rest on the floor of a rectangular channel is hit and lift off by a Mach 3 shock. The computational domain is $[0, 1] \times [0, .2]$ with the initial shock front positioned at .08 from the left boundary, and the rest of the domain initially filled with gas with $\rho = 1.4$, $p = 1$, and $\vec{u} = (0, 0)$. The standard reflective solid wall boundary conditions are applied to the top and bottom boundaries of the domain, while the left boundary is fixed to be the post shock state and the right boundary has a zero-gradient outflow boundary condition. The cylinder has density 10.77, radius .05, and is initially located at $(.15, .05)$. Table 10 lists the overlapping grids, where the position of the finer grid exactly follows that of the cylinder. The results obtained using second order accurate ENO-LLF with a global CFL number of 3 are shown in Figure 28, which compares well to [3, 24]. Note that there are small discrepancies between [3] and [24] as well as between our results and their results, which probably results from the different truncation errors inherent in the different numerical methods used. Figure 29 shows the convergence of the position of the center of mass of the cylinder as compared to a ground truth computed using a single static grid at resolution $n = 6400$ and third order accurate ENO-LLF with a CFL number of .5.

	Object space domain	Δx	$\vec{s}$	θ
1	$[0, 1] \times [-.04, .24]$	$1/n$	$(0, 0)$	0
2	$[-.08, .08] \times [-.08, .08]$	$.267/n$	follows cylinder	0

Table 10: The object space domains, cell sizes, positions, and orientations of the grids in the rigid cylinder lift-off example. n indicates the number of grid cells in x-direction on the background grid.

9.2.2. Deformable cylinder lift-off

We repeat the simulation in Section 9.2.1 replacing the rigid cylinder by a deformable mass-spring system with 216 triangles. We create an edge spring on each triangle edge with Young's modulus .3 and damping equal to 1.5 times critical damping. We also create an altitude spring [61] between each particle of a triangle and a virtual node projected onto the opposite face with Young's modulus .3 and damping equal to 2 times critical damping. The overlapping grids are the same as those in Table 10 except that the finer grid now follows the center of mass of the deformable cylinder. The results obtained using second order accurate ENO-LLF with a global CFL number of 3 are shown in Figure 30, which compares well to [24]. Figure 31

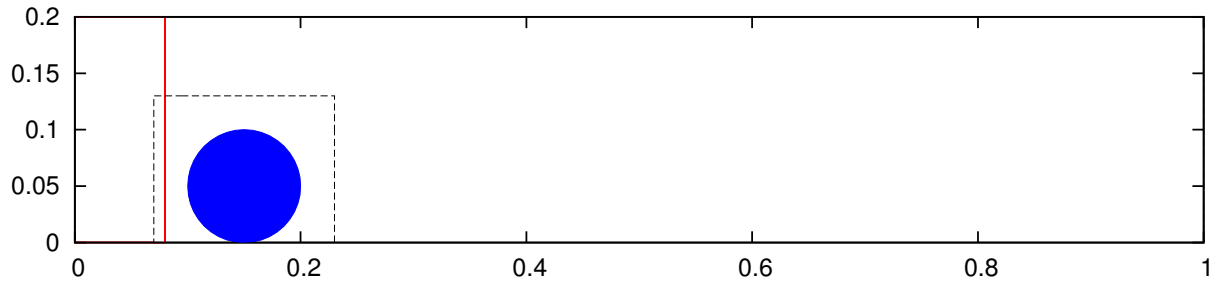
shows the convergence of the positions of the particles of the deformable cylinder as compared to a ground truth computed using a single static grid at resolution $n = 6400$ and third order accurate ENO-LLF with a CFL number of .5.

9.2.3. Rigid diamond lift-off

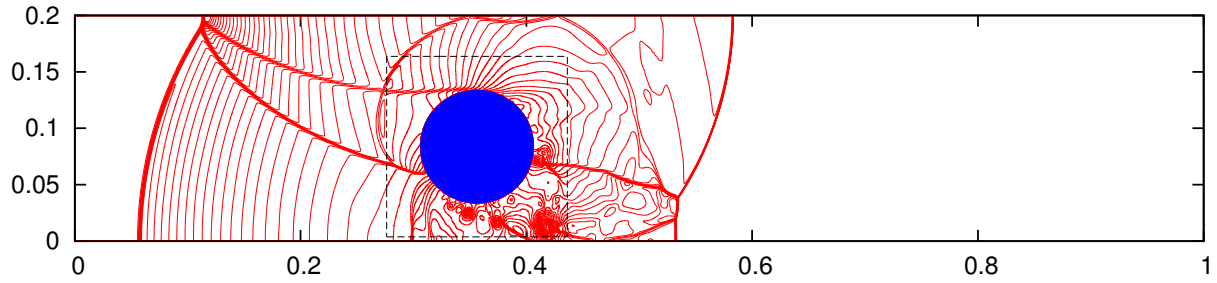
We repeat the simulation in Section 9.2.1 replacing the rigid cylinder by a rigid diamond whose major axis is of length .1 and minor axis is of length .025. The diamond has mass 2.22×10^{-2}, moment of inertia 1.96×10^{-5}, and is initially positioned at $(.15, .04)$ with a rotation angle of $\pi/4$. The friction coefficient and the restitution coefficient of the diamond and the solid walls are both set to be 1. The computational domain and the boundary conditions remain the same as those in Section 9.2.1. Table 11 lists the overlapping grids, where the finer grid exactly follows the motion of the diamond. We use the second order accurate ENO-LLF scheme for advection on all grids with a global CFL number of 3. Figure 32 shows the snapshots of the simulation, and Figure 33 shows the convergence of the positions of the corners of the diamond as compared a ground truth computed using a single static grid at resolution $n = 6400$ and third order accurate ENO-LLF with a CFL number of .5.

	Object space domain	Δx	$\vec{s}$	θ
1	$[0, 1] \times [-.04, .24]$	$1/n$	$(0, 0)$	0
2	$[-.036, .036] \times [-.072, .072]$	$.36/n$	follows diamond	follows diamond

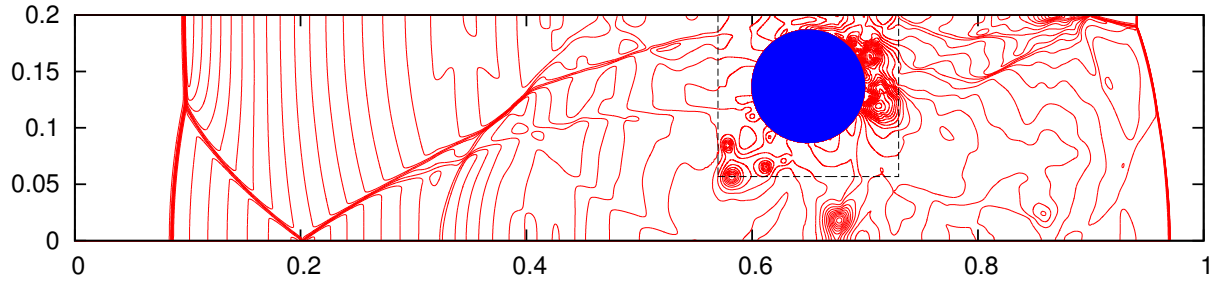
Table 11: The object space domains, cell sizes, positions, and orientations of the grids in the rigid diamond lift-off example. n indicates the number of grid cells in x-direction on the background grid.



(a) t=0

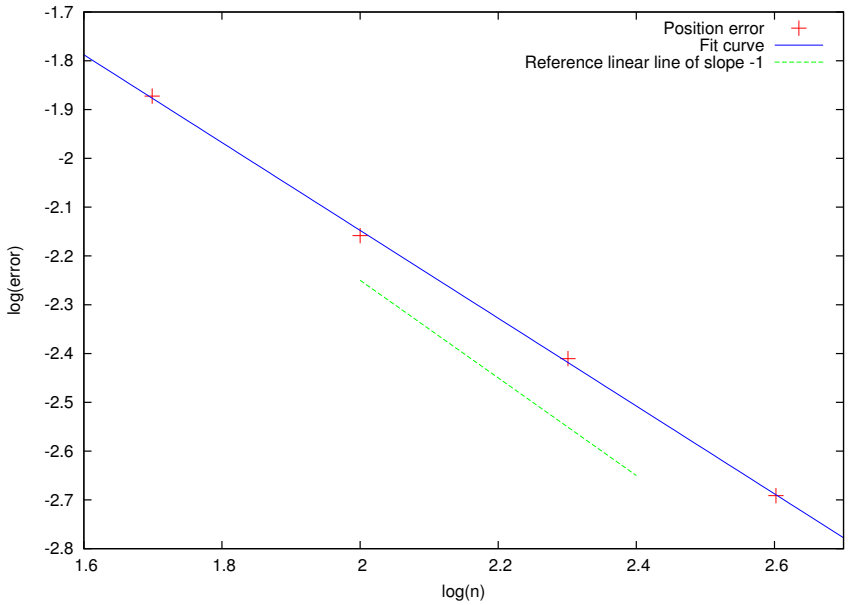


(b) t=.164

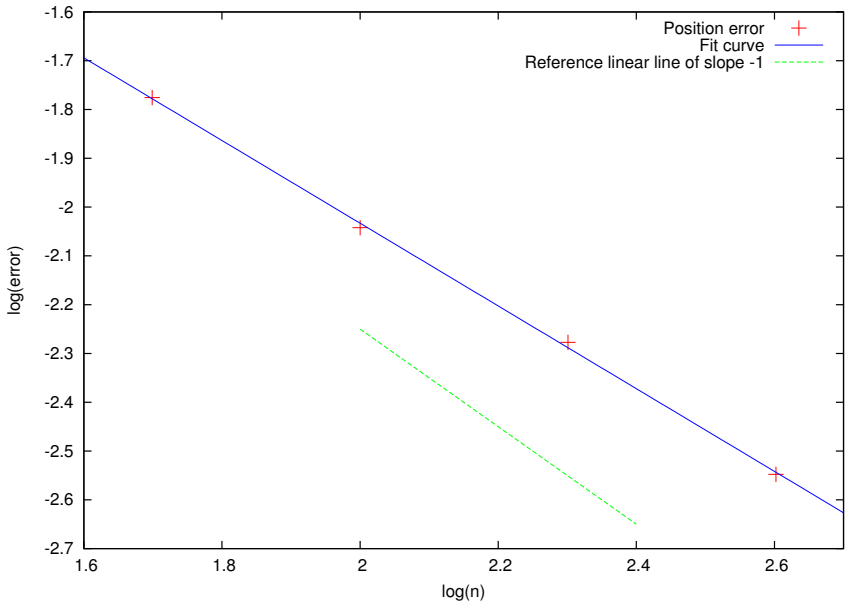


(c) t=.301

Figure 28: Pressure contours for the rigid cylinder lift-off example at grid resolution $n = 3200$ using second order accurate ENO-LLF with a global CFL number of 3. The dashed outline indicates the boundary of the finer grid that follows the cylinder. 53 contours are plotted within the range $[2, 28]$.

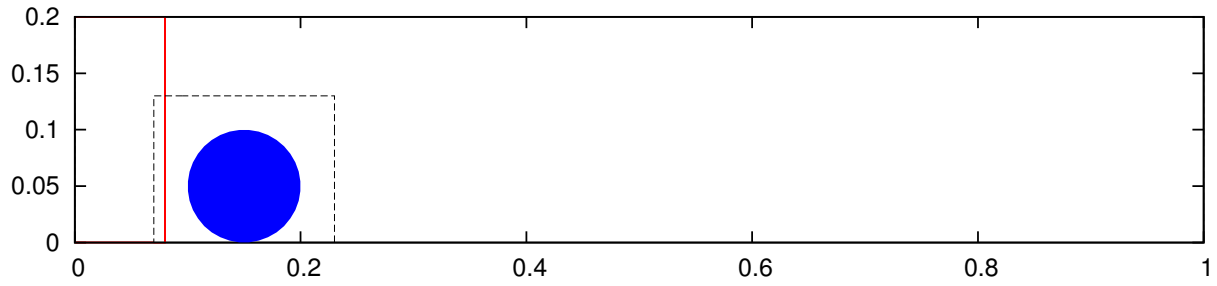


(a) ENO-LLF, CFL=3, convergence rate=.90

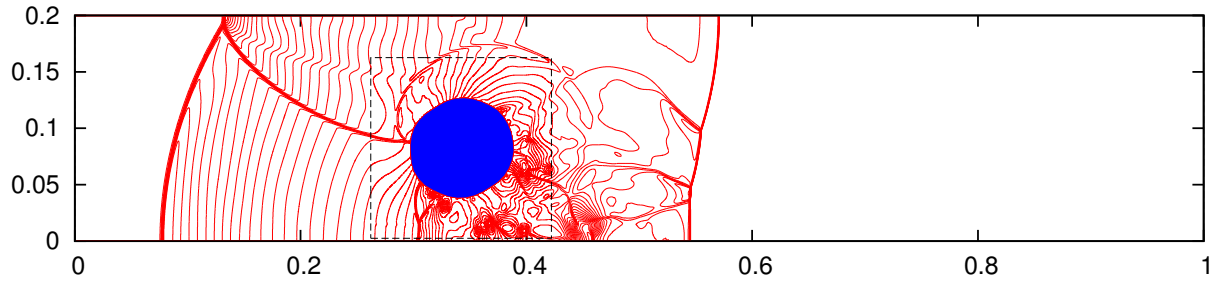


(b) Conservative semi-Lagrangian scheme, CFL=3, convergence rate=.85

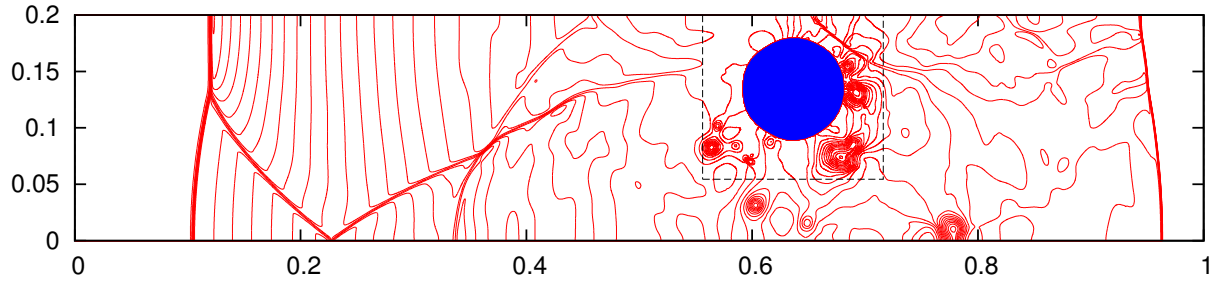
Figure 29: Convergence of the position errors of the center of mass of the cylinder at $t = .15$ in the rigid cylinder lift-off example. The grid resolutions for the data points in the plots are $n = 50$, 100, 200, and 400.



(a) t=0

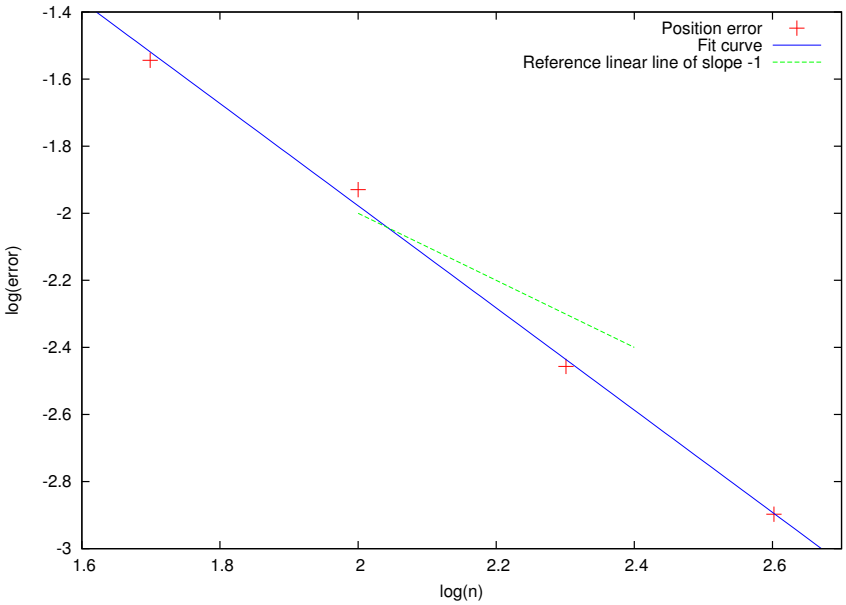


(b) t=.164

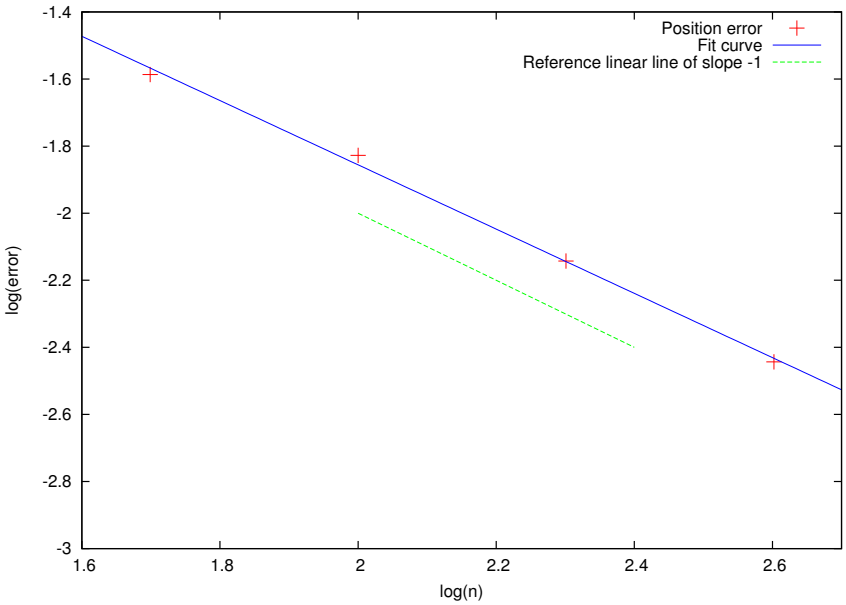


(c) t=.301

Figure 30: Pressure contours for the deformable cylinder lift-off example at grid resolution $n = 3200$ using second order accurate ENO-LLF with a global CFL number of 3. The dashed outline indicates the boundary of the finer grid that follows the center of mass of the deformable cylinder. 53 contours are plotted within the range $[2, 28]$.



(a) ENO-LLF, CFL=3, convergence rate=1.52



(b) Conservative semi-Lagrangian scheme, CFL=3, convergence rate=.96

Figure 31: Convergence of the average errors of the positions of the particles of the deformable cylinder at $t = .15$ in the deformable cylinder lift-off example. The grid resolutions for the data points in the plots are $n = 50$, 100, 200, and 400.

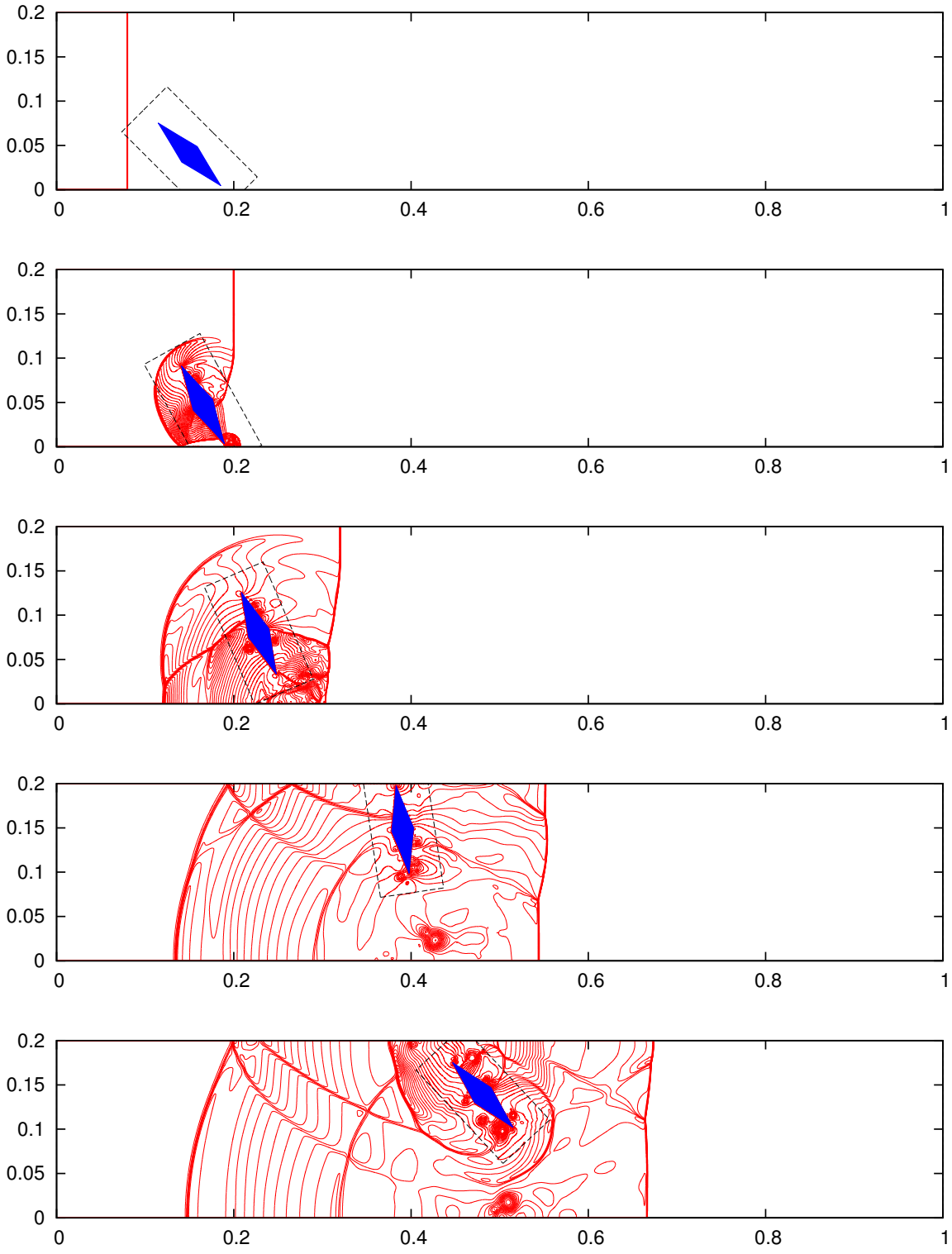


Figure 32: Pressure contours for the rigid diamond lift-off example at time $t = 0$, .04, .08, .16, and .2 at grid resolution $n = 3200$ using second order accurate ENO-LLF with a global CFL number of 3. The dashed outline indicate the boundary of the finer grid that follows the diamond. 53 contours are plotted within the range $[2, 28]$.

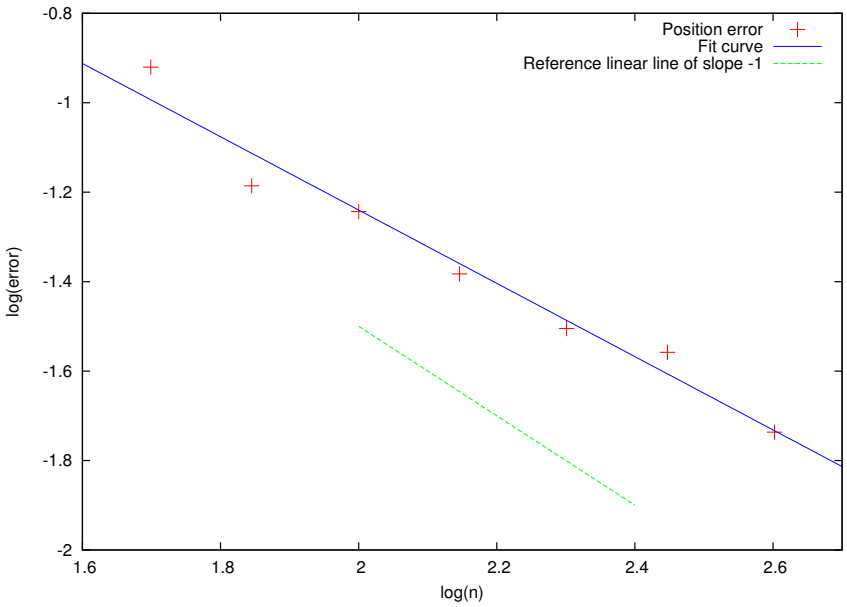


Figure 33: Convergence of the total errors of the positions of the corners of the diamond at $t = .1$ in the rigid diamond lift-off example. The convergence rate is .82. The grid resolutions for the data points are $n = 50$, 70, 100, 140, 200, 280, and 400.

10. Conclusion

We presented a method for simulating compressible flow on overlapping Cartesian grids that can rotate and translate. To handle advection we devised a second order accurate flux based scheme using an effective characteristic velocity that accounts for the grid motion without any extra remapping. Strategies were proposed to allow a fluid velocity CFL number larger than 1 in order to alleviate the stringent time step restriction imposed by very fine grids. The pressure was solved for by formulating an implicit linear system which avoids any time step restrictions related to the sound speed. The implicit discretization was accomplished via the Voronoi mesh, resulting in a symmetric positive-definite system that is solved via the preconditioned conjugate gradient method. Furthermore, in order to alleviate the spurious oscillations caused by the inherent central differencing in the implicit discretization of the pressure terms, we proposed adding high order diffusion terms similar in spirit to ENO-LLF but solved for implicitly avoiding any associated time step restrictions. Finally, a conservative interpolation operator was devised to enforce the consistency of the values in overlapped regions. The method is globally conservative on the background grid and locally conservative in the interior of each Cartesian grid. In addition, we demonstrated the ability to handle two-way solid fluid coupling problems.

There are numerous avenues for future work. For example, in order to use a fluid velocity CFL number larger than 1, we either subdivided the time steps during advection or used an unconditionally stable conservative semi-Lagrangian method. Since the conservative semi-Lagrangian method is only first order accurate, we observed an increase in errors when using it. This may be alleviated by utilizing higher order accurate conservative semi-Lagrangian methods. Another issue is that the current method assumed that the flux based schemes used for updating the conserved variables on each Cartesian grid are positivity preserving, which may not be true, so incorporating robust positivity preserving strategies into the current framework is also important. Furthermore, it would be useful to design strategies that allow moving grids to automatically keep track of interesting fluid features such as shock fronts as well as automatically expand or shrink.

11. Acknowledgements

Research supported in part by ONR N00014-13-1-0346, 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.

Appendix. Compressible Gap Flow

Here we briefly comment on our efforts to extend the gap flow formulation of [58] from incompressible flow to compressbile flow as well as from a single static grid to Chimera grids. We carried out a number of numerical tests in two and three spatial dimensions similar to those in [58]. Our two-way coupled gap flow solver passed all such tests for arbitrarily large fluid pressures, various overlapping fluid grids, various solid meshes, and various spatial configurations and geometries of rigid bodies. See [57] for more details.

Since the solid fluid two-way coupled solve is discretized on the Voronoi mesh, we create virtual solid-fluid faces and virtual fluid cells at Voronoi faces between two different solids. In order to simulate compressible flow in the gap region, we further define fluid state variables, i.e. density, momentum, and energy, on each of the pressure degree of freedom particles on solid surfaces. These fluid state variables are used to compute the fluid velocity tangential to the solid surface required for computing divergence in the gap region. As in [58], the two-way coupled system (Equation 59) is modified by augmenting vectors and matrices to account for the additional pressure and velocity degrees of freedom on solid surfaces, while retaining the symmetric positive-definiteness of the system. After the two-way coupled solve, the fluid momentum and energy on solid surfaces are updated using the gap flow gradient and divergence operators discussed in [58] except that the surface normal component of the momentum is set to be the density times the surface normal component of the pointwise solid velocity in order to enforce velocity compatibility on solid surfaces. Note that the current approach does not exactly conserve the conserved variables in the gap region – this is left as future work.

For advection, we split the flux terms into a non-conservative advection term and an expansion term yielding

$$\phi_t + \vec{u} \cdot \nabla\phi + \phi\nabla \cdot \vec{u} = 0. \tag{60}$$

The non-conservative advection term $\vec{u} \cdot \nabla\phi$ is solved using the method in [58] after interpolating density, momentum, and energy from the finest grid that has a valid interpolation stencil to surface particles that are not pressure degree of freedom particles in order to provide boundary conditions. When interpolating momentum to a surface particle, we only keep its tangential component while modifying its normal component to be the interpolated density times the surface normal component of the pointwise solid velocity. The kinetic energy is likewise modified for consistency. After applying the non-conservative advection term and obtaining intermediate values ϕ^*, we discretize the expansion term $\phi\nabla \cdot \vec{u}$ as follows so that no time step restriction is required in order to preserve the positivity of ϕ,

$$\begin{cases} \phi^{n+1} = \phi^*/(1 + \Delta t\nabla \cdot \vec{u}^n) & \text{if } \nabla \cdot \vec{u}^n > 0, \\ \phi^{n+1} = \phi^*(1 - \Delta t\nabla \cdot \vec{u}^n) & \text{if } \nabla \cdot \vec{u}^n \leq 0. \end{cases} \tag{61}$$

This is essentially backward Euler when $\nabla \cdot \vec{u}^n > 0$ and forward Euler when $\nabla \cdot \vec{u}^n \leq 0$.

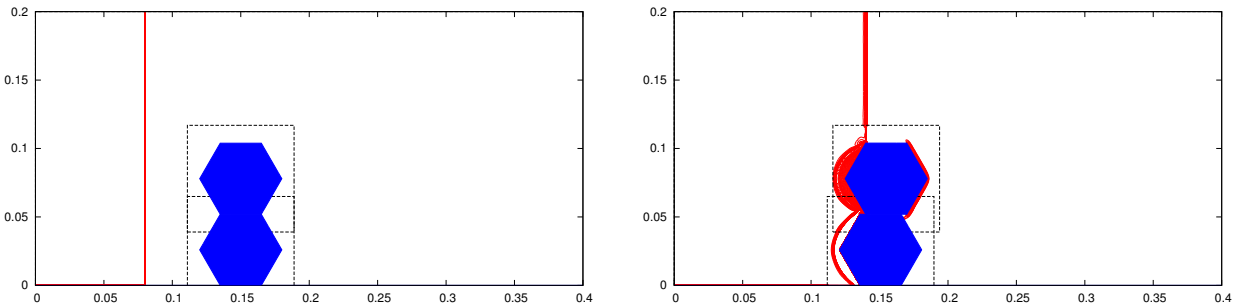
In order to enhance stability, we add artificial diffusion to fluid state variables on surface particles simply by averaging the value of each particle with the values of its one-ring neighbors weighted by the control volumes. Moreover, we average the fluid state varibles on the two sides of the gap.

We briefly report on the results of two rigid hexagons hit by a Mach 3 shock wave using the aforementioned gap flow extensions. The computational domain, initial conditions, and boundary conditions for this example are the same as those in Section 9.2.1. The two hexagons both have a uniform side length $l = .03$ and are positioned at $(.15, 1.5h)$ and $(.15, .5h)$, where $h = \sqrt{3}l$ is the height of the hexagon. The density of the top hexagon is 1, while the density of the bottom hexagon is 10. The friction coefficient and the restitution coefficient are both set to zero. The grids are specified in Table 12. Figure 34 shows snapshots of the simulation at grid resolution $n = 1600$ using second order accurate ENO-LLF with a global CFL number of 3. Convergence analysis indicates that the positions of the two hexagons converge with first order accuracy when compared to a higher resolution simulation on a single static grid. If we do not use the gap flow solver in this example, the simulations often crash when the shock wave lifts the hexagons, since the newly uncovered solid cells deep within the gap region have no nearby valid fluid state variables that can be used for filling them. One can avoid the crashing by extrapolating fluid state variables from fluid cells exterior

to the gap region. However, since the pressure forces in the gap region are not resolved until fluid degrees of freedom appear in the gap between the two solids, the solids are spuriously pushed against each other by the exterior pressure forces resulting in frequent spurious collisions adversely affecting convergence.

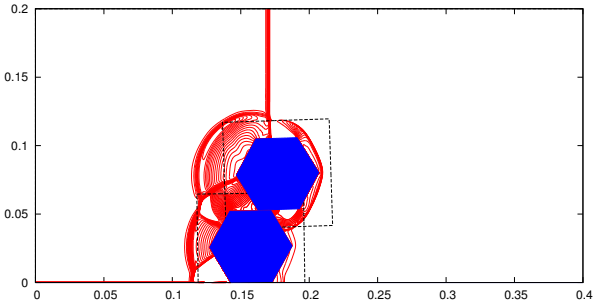
	Object space domain	Δx	$\vec{s}$	θ
1	$[0,1] \times [-.04, .24]$	$1/n$	$(0,0)$	0
2	$[-.75h, .75h] \times [-.75h, .75h]$	$.225\sqrt{3}/n$	follows top hexagon	follows top hexagon
3	$[-.75h, .75h] \times [-.75h, .75h]$	$.225\sqrt{3}/n$	follows bottom hexagon	follows bottom hexagon

Table 12: The object space domains, cell sizes, positions, and orientations of the grids in the hexagon stack example. n indicates the number of grid cells in x-direction on the background grid. The edge length of the solid mesh is .01 at grid resolution $n = 50$ and refined at the same rate as the grids.

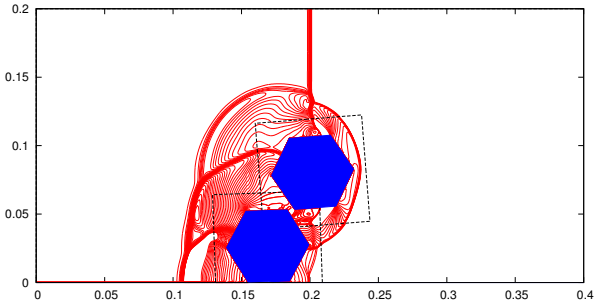


(a) t=0

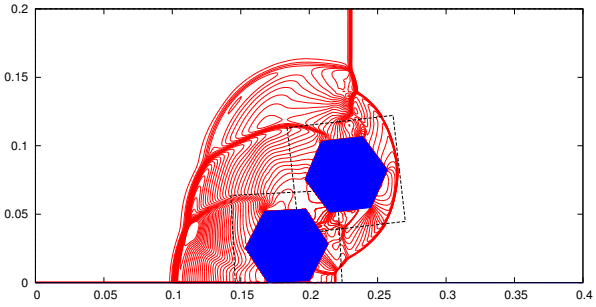
(b) t=.02



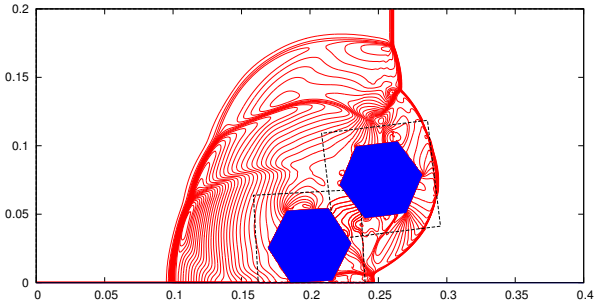
(c) t=.03



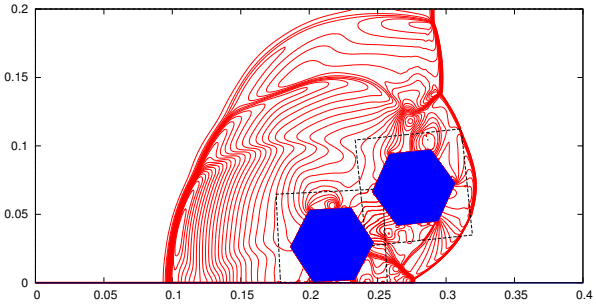
(d) t=.04



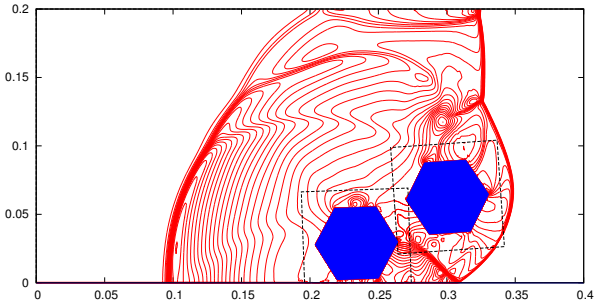
(e) t=.05



(f) t=.06



(g) t=.07



(h) t=.08

Figure 34: Pressure contours for the hexagon stack example at grid resolution $n = 1600$ using second order accurate ENO-LLF with a global CFL number of 3. The dashed outlines indicate the boundaries of the finer grids that follow the hexagons. 53 contours are plotted within the range $[2, 28]$.

Bibliography

[1] Mridul Aanjaneya, Saket Patkar, and Ronald Fedkiw. A monolithic mass tracking formulation for bubbles in incompressible flow. *J. Comput. Phys.*, 247:17–61, 2013.

[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] M. Arienti, P. Hung, E. Morano, and J. Shepherd. A level set approach to Eulerian-Lagrangian coupling. *J. Comput. Phys.*, 185:213–251, 2003.

[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] Markus Berndt, JéRôMe Breil, StéPhane Galera, Milan Kucharik, Pierre-Henri Maire, and Mikhail Shashkov. Two-step hybrid conservative remapping for multimaterial arbitrary Lagrangian-Eulerian methods. *J. Comput. Phys.*, 230(17):6664–6687, July 2011.

[10] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):594–603, 2002.

[11] 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.

[12] 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.

[13] R. DeBar. Fundamentals of the KRAKEN code. Technical report, Lawrence Livermore National Laboratory (UCID- 17366), 1974.

[14] R.E. English, L. Qiu, Y. Yu, and R. Fedkiw. An adaptive discretization of incompressible flow using a multitude of moving Cartesian grids. *J. Comput. Phys.*, 254(0):107 – 154, 2013.

[15] R.E. English, L. Qiu, Y. Yu, and R. Fedkiw. Chimera grids for water simulation. In *SCA '13: Proceedings of the 2013 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2013.

[16] J. Falcovitz, G. Alfandary, and G. Hanoch. A two-dimensional conservation laws scheme for compressible flows with moving boundaries. *J. Comput. Phys.*, 138(1):83–102, 1997.

[17] C. Farhat, M. Lesoinne, and P. Le Tallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity. *Comp. Meth. Appl. Mech. Eng.*, 157(1-2):95–114, 1998.

[18] Charbel Farhat and Vinod K. Lakshminarayan. An ALE formulation of embedded boundary methods for tracking boundary layers in turbulent fluidstructure interaction problems. *J. Comput. Phys.*, 263(0):53 – 70, 2014.

[19] R. Fedkiw, X.-D. Liu, and S. Osher. A general technique for eliminating spurious oscillations in conservative schemes for multiphase and multispecies Euler equations. *Int. J. Nonlinear Sci. and Numer. Sim.*, 3:99–106, 2002.

[20] R. Fedkiw, A. Marquina, and B. Merriman. An isobaric fix for the overheating problem in multimaterial compressible flows. *J. Comput. Phys.*, 148:545–578, 1999.

[21] H. Forrer and M. Berger. Flow simulations on Cartesian grids involving complex moving geometries. In *Hyperbolic Problems: Theory, Numerics, Applications; Seventh International Conference in Zürich, February 1998*, page 315. Birkhauser, 1999.

[22] G. Golub and U. von Matt. Tikhonov regularization for large scale problems. *Workshop on Scientific Computing*, pages 3–26, 1997.

[23] J. Grétarsson and R. Fedkiw. Fully conservative, robust treatment of thin shell fluid-structure interactions in compressible flows. *J. Comput. Phys.*, 245:160–204, 2013.

[24] J.T. Grétarsson, N. Kwatra, and R. Fedkiw. Numerically stable fluid-structure interactions between compressible flow and solid structures. *J. Comput. Phys.*, 230:3062–3084, 2011.

[25] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.

[26] 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.

[27] 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.

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

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

[30] C. Hirt, A. Amsden, and J. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J. Comput. Phys.*, 14(3):227–253, 1974.

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

[32] Matthew Jemison, Mark Sussman, and Marco Arienti. Compressible, multiphase semi-implicit method with moment of fluid interface representation. *J. Comput. Phys.*, 279:182–217, 2014.

[33] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126:202–228, 1996.

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

[35] 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.

[36] 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.

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

[38] N. Kwatra, J. Su, J.T. Grétarsson, and R. Fedkiw. A method for avoiding the acoustic time step restriction in compressible flow. *J. Comput. Phys.*, 228(11):4146–4161, 2009.

[39] M. Lentine, J.T. Grétarsson, and R. Fedkiw. An unconditionally stable fully conservative semi-lagrangian method. *J. Comput. Phys.*, 230(8):2857–2879, April 2011.

[40] R.J. LeVeque and K.-M. Shyue. Two-dimensional front tracking based on high resolution wave propagation methods. *J. Comput. Phys.*, 123(2):354–368, 1996.

[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] W. Liu, L. Yuan, and C.-W. Shu. A conservative modification to the ghost fluid method for compressible multiphase flows. *Commun. Comput. Phys.*, 10(4):785–806, 2011.

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

[44] 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.

[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] 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.

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

[48] 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.

[49] 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.

[50] 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.

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

[52] S. Patkar, M. Aanjaneya, W. Lu, and R. Fedkiw. Positivity preservation for monolithic two-way solid-fluid coupling. *(In Preparation)*, 2014.

[53] 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.

[54] N Anders Petersson. An algorithm for assembling overlapping grid systems. *SIAM Journal on Scientific Computing*, 20(6):1995–2022, 1999.

[55] N Anders Petersson. Hole-cutting for three-dimensional overlapping grids. *SIAM Journal on Scientific Computing*, 21(2):646–665, 1999.

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

[57] L. Qiu. *An adaptive discretization for incompressible and compressible flow using a multitude of moving Cartesian grids with gap flow treatment*. PhD thesis, Stanford University, June 2015.

[58] L. Qiu, Y. Yu, and R. Fedkiw. On thin gaps between rigid bodies two-way coupled to incompressible flow. *J. Comput. Phys.*, 292(0):1 – 29, 2015.

[59] A. Robinson-Mosher, C. Schroeder, and R. Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. *J. Comput. Phys.*, 230(4):1547–1566, 2011.

[60] 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.

[61] A. Selle, M. Lentine, and R. Fedkiw. A mass spring model for hair simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 27(3):64.1–64.11, August 2008.

[62] T. Shinar, C. Schroeder, and R. Fedkiw. Two-way coupling of rigid and deformable bodies. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '08, pages 95–103, 2008.

[63] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

[64] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes II (two). *J. Comput. Phys.*, 83:32–78, 1989.

[65] 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.

[66] 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.

[67] 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.

[68] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963.

[69] H.S. Udaykumar, H.-C. Kan, W. Shyy, and R. Tran-Son-Tay. Multiphase dynamics in arbitrary geometries on fixed cartesian grids. *J. Comput. Phys.*, 137(2):366–405, 1997.

[70] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comput. Phys.*, 54:115–173, April 1984.

[71] H.C Yee, N.D Sandham, and M.J Djomehri. Low-dissipative high-order shock-capturing methods using characteristic-based filters. *J. of Comp. Phys.*, 150(1):199 – 238, 1999.

[72] D. L. Youngs. Time-dependent multi-material flow with large fluid distortion. In *Numerical Methods for Fluid Dynamics*, page 273, New York, 1982. Academic Press.

[73] D. L. Youngs. An interface tracking method for a 3D Eulerian hydrodynamics code. Technical report, AWRE (44/92/35), 1984.