Fully conservative leak-proof treatment of thin solid structures immersed in compressible fluids

Jón Tómas Grétarsson*, Ron Fedkiw*

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

Abstract

We propose a novel high resolution conservative advection scheme that is suitable for thin, embedded moving solid structures. The scheme works by coupling together a high order flux-based method with a conservative semi-Lagrangian solver that is similar in spirit to that of [26], but modified to treat the cut cells and partial volumes that arise near a thin solid structure. The conservative semi-Lagrangian scheme is unconditionally stable, and so unlike previous methods no cell merging is required to compensate for the small cell volumes that arise. Furthermore, as the semi-Lagrangian scheme works via tracing characteristic curves, no special treatment is required either to enforce non-penetration through thin, moving solid structures, or to populate swept or uncovered degrees of freedom. For the flux-based solver, we use finite-difference ENO with Lax-Friedrich's diffusion (although any flux-based scheme works), and in doing so we found that a modification to the diffusion calculation leads to improved stability in its third order accurate variant. We integrate this novel hybrid advection scheme into a semi-implicit compressible flow solver, and modify the implicit pressure solver to work with cells of variable size. In addition, we propose an improvement to the semi-implicit compressible flow solver via a new method for computing a post-advected pressure. Finally, this hybrid conservative advection scheme is integrated into a semi-implicit fluid-structure solver, and a number of one-dimensional and two-dimensional examples are considered—in particular, showing that we can handle thin solid structures moving through the grid in a fully conservative manner, preventing fluid from leaking from one side of the structure to the other and without the need for cell merging or other special treatment of cut cells and partial volumes.

1. Introduction

The Direct Numerical Simulation (DNS) of fluid-structure interactions has recently received significant attention. Many of these works concern themselves with fluid flow in the incompressible flow regime, see for example [8, 22] and the references within, but researchers are increasingly giving attention to the two-way coupled interactions that arise in compressible flows, see for example [4, 17, 9]. If one desires to use a state-of-the-art Eulerian method on the fluid flow, and a state-of-the-art Lagrangian method for the structure solver, then this requires a numerical method for coupling these two solvers together. Fluid-based forces need to be transferred to the solid structure, and position and velocity-based boundary conditions must be applied to the fluid based on the current location and movement of the solid structure. One of the primary research areas in solid-fluid coupling concerns the stability of the numerical methods for coupling and is essentially focused on the feedback loop where pressure is applied to the solid, the solid structure reacts and deforms, and subsequently imposes position and velocity-based boundary conditions on the fluid. While the most straightforward approach is simply to treat the coupling in an explicit way, called a partitioned method [49, 41, 10], researchers have focused quite a bit of attention on so-called monolithic methods that employ higher degrees of implicit coupling [42, 14], in order to stabilize parts or all of this feedback loop. Another important issue regards the modifications that the Eulerian method requires to treat cells cut by

*{jontg,fedkiw}@cs.stanford.edu, Stanford University

the solid structure as well as those that are covered or uncovered as the structure sweeps across the Eulerian grid—especially in regards to stability and conservation. A common approach for treating these issues on the Eulerian grid is to fill the cells that are covered or partially covered by the solid structure with ghost values of some type, and then proceed in the standard way ignoring the solid all-together. This alleviates stability restrictions for cut cells, automatically creates new fluid in uncovered cells, and has been theme of the approach for the ghost fluid method [11] and the immersed boundary method (see [40] and the references therein, including [38, 39]). The fluid placed in these ghost cells must include the added mass effect of the solid, i.e. if the solid is heavier or lighter than the surrounding fluid, the ghost cells must properly represent that mass difference. The added mass can be accounted for in thin solid structures as well (see for example [50]), simply by adding that mass to the fluid cells that contain the solid structure. Whereas cut cell methods overcome stability restrictions for the cut cells, they do not maintain either conservation nor the ability for the fluid on one side of the structure to remain on that side, i.e. the fluid can leak across to the other side of the structure. In order to address these concerns, authors have focused on cut cell methods, see for example [16, 17] and the references therein. The main issue with these methods is in the treatment of small cell volumes, which can impose additional time step restrictions on the flow solver if special techniques such as cell merging near the structure interface are not used. Furthermore these methods can become extremely complex if the solid structure is sweeping across the grid. In fact, most approaches to treating covering and uncovering of cells are non-conservative, and even then there can be issues [45]. Generally speaking uncovered cells need to be replaced with a valid value, and one can do this with any number of methods that range from simply interpolating from nearby neighbors to using upwind information to populate these cells, see for example [28, 27, 47]. Our semi-Lagrangian approach also uses upwind information to fill uncovered cells, but with the aide of [26] more readily lends itself to a fully conservative approach than does a flux-based method.

We propose a novel treatment for cut cells and partial cell volumes near the structure interface. Unlike previous methods, this approach does not rely on cell merging to alleviate the time step restriction; instead we employ a conservative semi-Lagrangian scheme, similar in spirit to [26]. We make two major modifications to this method. First, the method is modified to support non-uniform grids, noting that care must be taken when a characteristic emanating from a large grid cell lands in the midsts of many small grid cells, and vice versa. Second, since the semi-Lagrangian method is low order accurate, we hybridize it with a high order accurate flux-based ENO method [46] (although any flux-based scheme works) so that high resolution can be obtained throughout the flow with the semi-Lagrangian method only being applied near the thin solid interface. As the semi-Lagrangian solver works by tracing along characteristic curves, we can use continuous collision-detection [7, 15] to guarantee that fluid does not penetrate into a volumetric solid or cross over from one side to the other on a thin solid. This works even when the solid is moving and is under-resolved by the grid. Notably, the resulting method requires no special treatment for swept or uncovered cells.

Using the semi-Lagrangian method to handle cells near the structure interface is similar in spirit to both volume of fluid (VOF) [20] and arbitrary Lagrange-Eulerian (ALE) [19] methods, which both explicitly move information along characteristics in a Lagrangian manner and both explicitly conserve the material. Although some versions of the volume of fluid scheme intersect flux-swept volumes with the volume fraction, others actually mesh up the volume fraction and move it through the grid in a Lagrangian fashion. If one treats each vertex of the meshed-up VOF polygon as a Lagrangian particle, continuous collision-detection can be applied to it in the same fashion as we do for our semi-Lagrangian rays. In this manner one can achieve conservation, stability and also prevent material from interpenetrating volumetric solids or crossing over thin solids. Afterwards, this advected polygon of volume needs to be deposited and stored on the grid so that it can be remeshed into the VOF representation at the next time step. The issue here comes in the representation; that is, if a cell is cut by a thin structure one needs to represent that volume fraction on the grid in a way that does not cross over the structure. The semi-Lagrangian method stores information at grid points (cell centers in our implementation) and therefore overcomes this, but a volume of fluid method would need to reconstruct the geometry in such a way that cuts the cells across the interface designated by the solid boundary. Similarly ALE methods push along the vertices of their mesh in a manner similar to both the VOF and semi-Lagrangian methods, and thus those vertices can be collided with the structure.

Again, one of the more complex aspects of this is in keeping the structure for the ALE mesh commensurate with the solid structure interface. Moreover, another issue with the ALE method is that pushing nodes around in a Lagrangian fashion and colliding them with the structure interface can result in inversion, and unless one wants to untangle the ALE mesh [48] and attempt to fit it to the solid structure, a remapping method needs to be employed where the material is dropped back down onto some Eulerian mesh and then remeshed in a way that fits the structure. In general we believe that both VOF and ALE methods could be applied in a manner similar to what we propose for our method, as long as one could work out the details for hybridization with the flux-based scheme and for redepositing the material near the solid interface onto an Eulerian grid. However, we feel that the conservative semi-Lagrangian approach of [26] is a very simple and straight-forward way to do this. We refer the interested reader to the following relevant VOF [18, 37, 2, 3, 30] and ALE papers [23, 34, 33, 35, 36, 5].

In order to capture the fluid-structure interactions we employ the flux-split compressible coupling methodology of [14], where the fluid flux terms are split into advective terms and pressure terms. The linearly degenerate advective terms are solved independently of the structure after which an implicit, monolithic coupled system is solved for the fluid pressures, the fluid-structure impulses and the structure velocity degrees of freedom. By treating the interactions implicitly, no new time step restrictions arise as a result of high density-to-mass ratios and we are free to work both with infinitesimally light structures as well as extremely heavy ones. We make two modifications to this method. The first modification addresses the fact that the original paper used a rasterized version of the solid structure, and so each grid cell is either completely full or completely empty of fluid. Second, we propose some modifications to the flux-splitting method of [24] that are primarily concerned with a better computation of what they refer to as the advected pressure.

The remainder of this paper is organized as follows; in Section 2 the conservative semi-Lagrangian advection scheme of [26] is extended to work with arbitrarily-sized control volumes, and then coupled together with a high order accurate flux-based solver in order to obtain a high resolution hybrid conservative advection scheme. Section 3 incorporates this hybrid conservative advection scheme together with the flux-split Eulerian flow solver of [24], makes note of a few algorithmic modifications that lead to better performance at a reduced computational cost, and alters the implicit pressure solve to account for non-uniformly sized control volumes. Section 4 couples together this Eulerian flow solver with the implicit fluid-structure interaction solver of [14], demonstrating how the hybrid advection flow solver can be used to treat cut cells and partial volumes. This section also covers how the volumetric conservative semi-Lagrangian advection scheme is modified to both guarantee collision-free advection near thin, moving solid structures, as well as maintain high resolution temporal accuracy in the flux-based region of the flow field. In Section 5 we demonstrate the resulting two-way coupled method for a variety of one-dimensional and two-dimensional examples.

2. Conservative semi-Lagrangian advection

We begin with a short review of [26] to lay the groundwork for conservative semi-Lagrangian advection before proposing our novel extension to non-uniform grids in Section 2.1, and then hybridizing the resulting method with a traditional flux-based method in Section 2.3. A traditional semi-Lagrangian scheme approximates

$$\phi_t + \vec{u} \cdot \nabla \phi = 0, \tag{1}$$

where ϕ is some passively advected scalar through a velocity field $\vec{u}$, by looking backward to a sample point x^- along characteristic lines and then interpolating to this sample point from nearby grid points. This can be written as

$$\phi_j^{n+1} = \phi\left(x_j, t^{n+1}\right) = \phi\left(x_j^-, t^n\right) = \sum_i w_{ij} \phi\left(x_i, t^n\right) \tag{2}$$

where w_{ij} are interpolation weights from some neighborhood of cells located at x_i to the sample point x_j^- (that is, $x_j^- = \sum_i w_{ij} x_i$). A first order accurate approximation may for example compute $x_j^- = x_j - \Delta t \vec{u}_j^n$, and then use bi-linear (or tri-linear, in three spatial dimensions) interpolation to compute $\phi\left(x_j^-, t^n\right)$; never explicitly computing w_{ij}.

In [26] this scheme is modified to instead solve the conservative form,

$$\hat{\phi}_t + \nabla \cdot (\hat{\phi} \vec{u}) = 0, \tag{3}$$

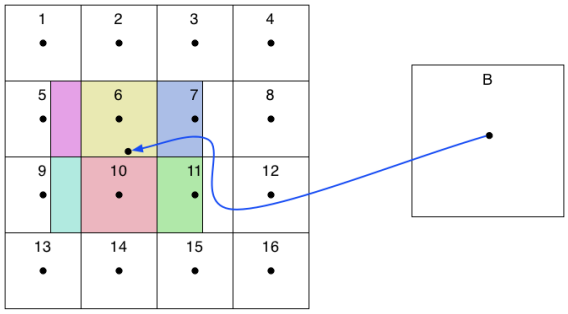
which can be derived by multiplying Equation (1) through by conservation of mass and setting $\hat{\phi} = \rho\phi$ (where ρ is density). They define $\sigma_i = \sum_j w_{ij}$ to be the fractional contribution that the time t^n data in cell i gives to the time t^{n+1} solution, and note that in order to remain conservative it is this term—rather than the sum over i—that should equal to one. That is, a given cell i must contribute all of its time t^n data to the time t^{n+1} solution; no more, and no less.

In order to strictly enforce conservation while remaining consistent with Equation (3), [26] computes modified weights $\hat{w}_{ij}$ such that $\sum_j \hat{w}_{ij} = 1$ for all cells i. This is done in three stages, beginning with a traditional semi-Lagrangian step. Rather than implicitly computing w_{ij} by way of computing $\phi\left(x_j^-, t^n\right)$, however, these weights are computed directly and saved. Next, cells that contribute too much to the time t^{n+1} solution (i.e. $\sigma_i > 1$) are clamped from above, and $\hat{w}_{ij} = w_{ij}/\sigma_i$ for these cells. Finally, cells that have any remaining material (i.e. $\sigma_i < 1$) have their remaining material pushed forward along characteristic curves. This is done through a second semi-Lagrangian step, this time advecting a point x_i *forward* along its characteristic curve to a sample point x_i^+; in a first order accurate method, for example, x_i^+ could be computed as $x_i + \Delta t \vec{u}_i^n$. Forward-advected weights f_{ij} are then computed as the interpolation weights to x_i^+ over a neighborhood of cells j (such that $x_i^+ = \sum_j f_{ij} x_j$). As $\sum_j f_{ij} = 1$, the remaining material is fully distributed to the time t^{n+1} solution if, for these cells, $\hat{w}_{ij} = w_{ij} + (1 - \sigma_i) f_{ij}$.

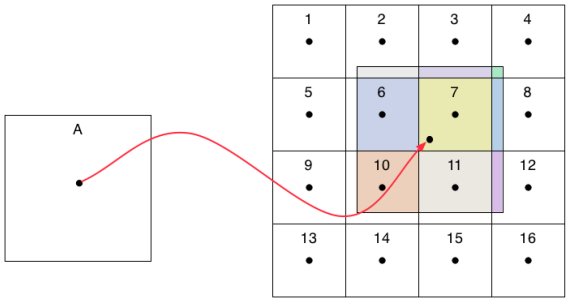
To summarize, they compute modified weights

$$\hat{w}_{ij} = \begin{cases} w_{ij}/\sigma_i & \sigma_i > 1 \\ w_{ij} + (1 - \sigma_i) f_{ij} & \sigma_i \leq 1 \end{cases} \tag{4}$$

and update $\hat{\phi}_j^{n+1} = \sum_i \hat{w}_{ij} \hat{\phi}_i^n$, to solve Equation (3). By moving data from time t^n to time t^{n+1} only along characteristic lines, they have consistency with Equation (3), and by enforcing that $\sum_j \hat{w}_{ij} = 1$ they attain numerical conservation. A key point is that Equation (3) is governed by the linearly degenerate eigenvalue u, and thus can also be used for the linearly degenerate velocity eigenvalue in the compressible flow equations. The interested reader is encouraged to read [25], which explores the conservative semi-Lagrangian approach in some detail, showing the benefits of using the conservative form over Equation (1) even for an incompressible flow.



(a) The control volume for a cell B is advected backward in time along its characteristic curve, and its overlap onto the fixed background grid is computed as $w_{jB} = \|\Omega_B^- \cap \Omega_j\|$. The resulting weights for this particular example are $w_{5B} = .396, w_{6B} = 1.00, w_{7B} = .604, w_{9B} = .396, w_{10B} = 1.00, w_{11B} = .604$. Note that these weights sum to four, which is the size of the original control volume $\|\Omega_B\|$ relative to the size of the smaller control volumes.



(b) The control volume for a cell A is advected forward in time along its characteristic curve, and its overlap onto the fixed background grid is computed as $f_{Aj} = \|\Omega_A^+ \cap \Omega_j\| / \|\Omega_A\|$. The resulting weights for this particular example are $f_{A2} = .034, f_{A3} = .039, f_{A4} = .006, f_{A6} = .212, f_{A7} = .250, f_{A8} = .037, f_{A10} = .179, f_{A11} = .211, f_{A12} = .032$. Note that unlike w_{ij} these weights sum to one.

Figure 1: Advected control volumes are moved backward and forward in space along its characteristic curve, and then distributed among control volumes in the fixed grid. Consider the examples above, where a large control volume four times the size of the smaller cells is advected into a region of smaller grid cells.

2.1. Non-uniform grids

We propose a modified version of this scheme that is suitable for non-uniform grids. In particular, we do not compute the backward-advected weights w_{ij} and forward-advected weights f_{ij} through an interpolation kernel (although, in the case of a uniform grid, this method does degenerate back to that of a bi-linear interpolation kernel). Instead these weights are computed as the overlap between a control volume that moves along characteristic lines and control volumes that remain fixed on a background grid. For brevity, we define a volume measure

$$\|\Omega\| = \int_\Omega dx.$$

When computing w_{ij}, the control volume for cell j, Ω_j, is moved backward in time along characteristic curves by a distance $|\vec{u}|\Delta t$. This translated volume Ω_j^- is then distributed among all overlapping control volumes from the background grid using our volume measure $\|\cdot\|$. These backward-advected weights are then given as $w_{ij} = \|\Omega_i \cap \Omega_j^-\|$, which is illustrated graphically for a simple example in Figure 1(a). These can be thought of as volume fractions of material moving from cell i to cell j. One could consider a control volume Ω_j^- that dilates by $(\nabla \cdot \vec{u})\Delta t$ as it moves along the characteristic curves, but we prefer the simplicity of a simple translating n-dimensional cube at the cost of some $\mathcal{O}(\Delta x)$ numerical errors. Note that a typical semi-Lagrangian method also has no mechanism to account for this dilation and therefore would possess similar errors. Those who pursue the semi-Lagrangian approach through a Lagrangian plus remap-style method can account for dilation and volume changes, and although significantly more complex than our approach, they bear some similarities; see for example [6] and the references therein.

For conservation we are interested in distributing all of the time t^n volume of material of cell i to the time t^{n+1} solution, or equivalently ensuring that the sum of volume fractions $\sum_j w_{ij}$ is equal to $\|\Omega_i\|$. In order to enforce this, we redefine $\sigma_i = \sum_j w_{ij}$ as the total volume of material entering the time t^{n+1} solution from the time t^n cell i, and use this term in three stages as before.

If $\sigma_i > \|\Omega_i\|$ then the cell is contributing too much volume of material to the time t^{n+1} solution, and all corresponding weights are scaled down accordingly by $\|\Omega_i\|/\sigma_i$. This enforces that no new material is created. Cells not contributing enough to the t^{n+1} solution (i.e. $\sigma_i < \|\Omega_i\|$) have their remaining volume advected forward along characteristic curves. This is done through a second advection step, moving the control volume for a given cell i forward in time along characteristic curves. This translated control volume

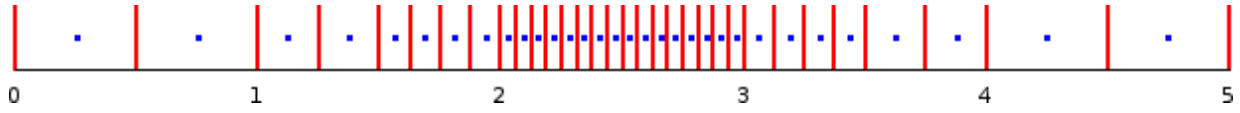


Figure 2: A non-uniform one-dimensional spatial grid, with flux boundaries shown as red vertical lines and cell-centered degrees of freedom as blue points. In the depicted grid, the smallest cell is of size $\Delta x_f = .0625$, while the largest cell is of size $\Delta x_c = .5$, $8\times$ larger than the smallest cells. When the grid resolution r is specified, we set $\Delta x_c = 5/r$ and scale the more refined regions appropriately.

Ω_i^+ is distributed among control volumes on a fixed background grid, and forward-advected weights are computed and normalized as $f_{ij} = \|\Omega_i^+ \cap \Omega_j\|/\|\Omega_i^+\|$; this is illustrated graphically in Figure 1(b). The remaining volume $\|\Omega_i\| - \sigma_i$ is distributed using these forward-advected weights, exploiting that $\sum_j f_{ij} = 1$.

To summarize, we solve Equation (3) on an arbitrary grid by computing backward-advected weights $w_{ij} = \|\Omega_i \cap \Omega_j^-\|$ and forward-advected weights $f_{ij} = \|\Omega_i^+ \cap \Omega_j\|/\|\Omega_i^+\|$ as necessary, and then compute modified weights

$$\hat{w}_{ij} = \begin{cases} (\|\Omega_i\|/\sigma_i) w_{ij} & \sigma_i > \|\Omega_i\| \\ w_{ij} + (\|\Omega_i\| - \sigma_i)\, f_{ij} & \sigma_i \leq \|\Omega_i\|. \end{cases} \tag{5}$$

The final update can then be written as

$$\hat{\phi}_j^{n+1} = \sum_i \frac{\hat{\phi}_i^n \hat{w}_{ij}}{\|\Omega_j\|}. \tag{6}$$

2.2. Examples

Equation (3) is solved for a pair of examples in one spatial dimension. The first is a sine-wave bump advected through a constant velocity field with $u = 1$, with initial state specified as

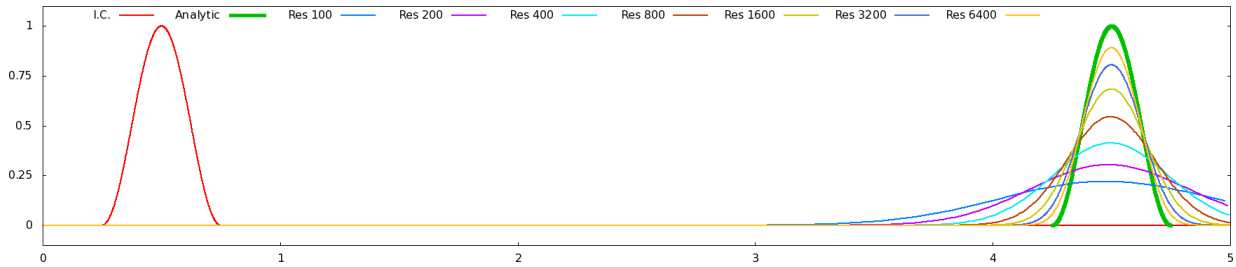
$$\hat{\phi}(x) = \begin{cases} \frac{1}{2}\left(1 + \sin\left[4\pi(x - \frac{3}{8})\right]\right) & x \in \left(\frac{1}{4}, \frac{3}{4}\right) \\ 0 & \text{otherwise} \end{cases}.$$

Figures 3(a) and 3(b) compare results at time $t = 4s$ between the volumetric conservative semi-Lagrangian advection and first order accurate ENO-LLF schemes on a uniform grid. In Figures 3(c) and 3(d) a non-uniform grid (shown in Figure 2) is utilized, and in Figure 3(e) the conservative semi-Lagrangian advection scheme is used with an effective CFL number of $\alpha = 4$ (well outside of the stability regime of a flux-based scheme) in order to demonstrate its unconditional stability. In all of these simulations, the results are qualitatively similar and the peak converges to its analytic solution at a rate of approximately .75.

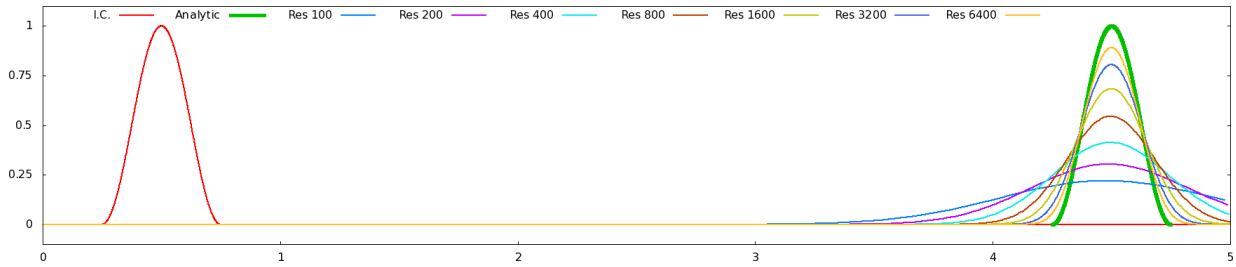
We also consider a square wave being advected through a divergent velocity field with $u(x) = \sin\left(\frac{\pi}{5}x\right)$. In Figures 4(a) and 4(b) the results at time $t = 5s$ are compared between the volumetric conservative semi-Lagrangian advection and first order accurate ENO-LLF schemes, on a uniform grid. In Figures 3(c) and 3(d) a non-uniform grid (shown in Figure 2) is utilized, and in Figure 3(e) the conservative semi-Lagrangian advection scheme is used with an effective CFL number of $\alpha = 4$ in order to again demonstrate its unconditional stability. We compute the analytic solution via the method of characteristics, noting that the total material between two characteristic curves does not change.

2.3. Hybrid flux / conservative semi-Lagrangian coupling

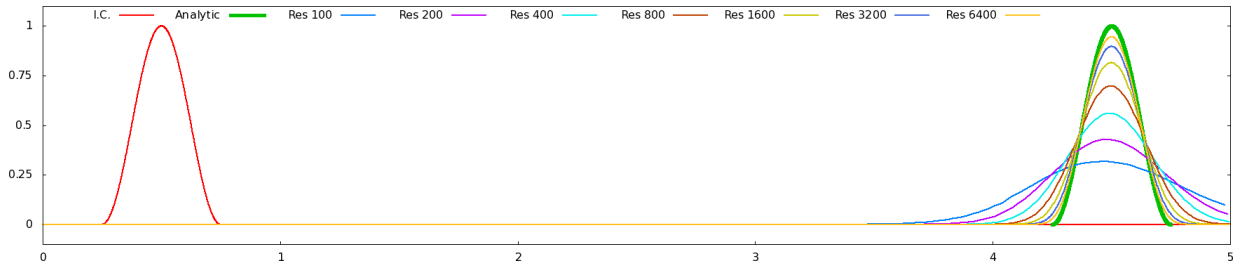
The conservative semi-Lagrangian advection scheme is unconditionally stable, which makes it ideal for regions of the flow field where small or irregular grid cells are necessary to resolve small-scale geometric detail. However, the method is limited to first order accuracy, and so is less desirable as a solver for the bulk of the flow. We therefore consider a hybrid formulation where the conservative semi-Lagrangian scheme is coupled with a more traditional high order accurate flux-based scheme. This hybrid formulation works by imposing fluxes as boundary conditions to the conservative semi-Lagrangian solver, and so can be used with any flux-based method.



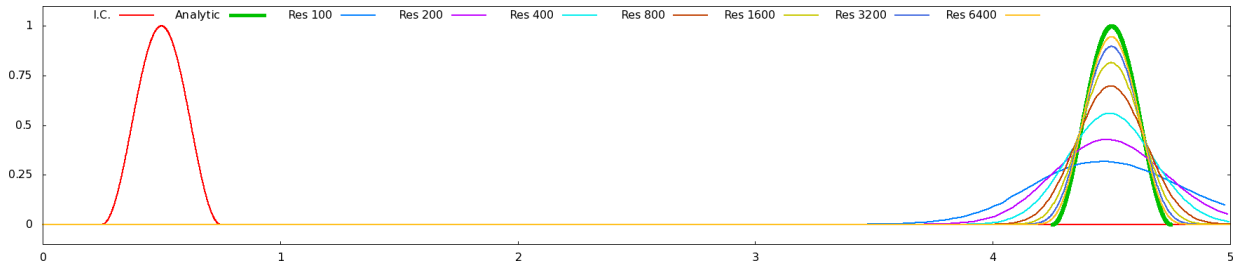
(a) First order accurate ENO-LLF on a uniform grid, with a CFL number $\alpha = .5$.



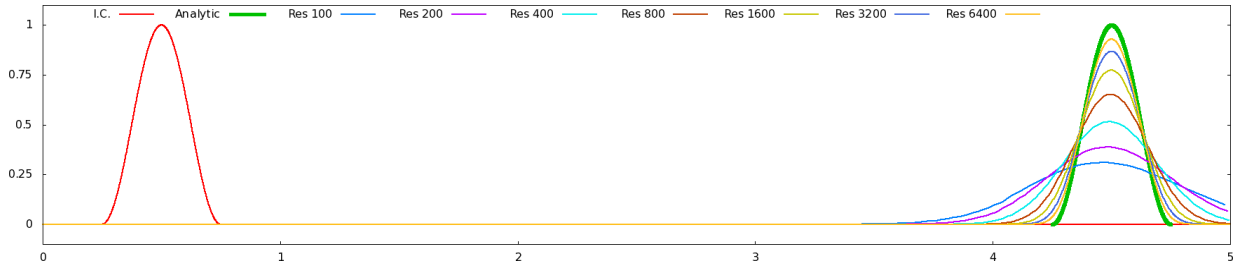
(b) Conservative semi-Lagrangian scheme on a uniform grid, with a CFL number $\alpha = .5$.



(c) First order accurate ENO-LLF on the non-uniform grid shown in Figure 2, with a CFL number $\alpha = .5$.

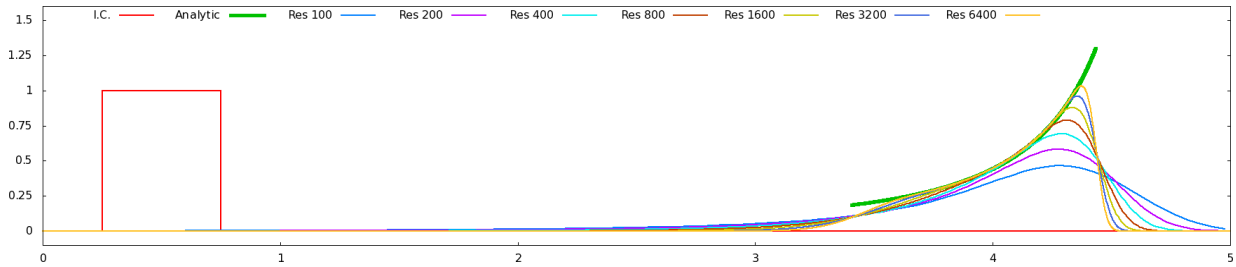


(d) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 2, with a CFL number $\alpha = .5$.

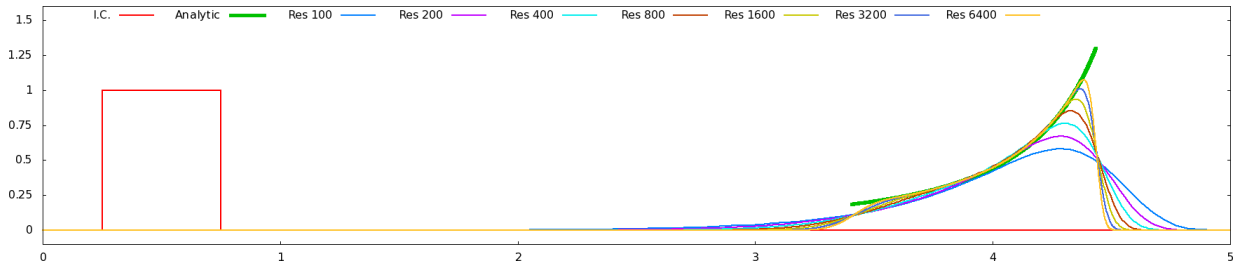


(e) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 2, with a CFL number $\alpha = .5$ taken with respect only to the coarse grid cells (and so the *effective* CFL number is 4).

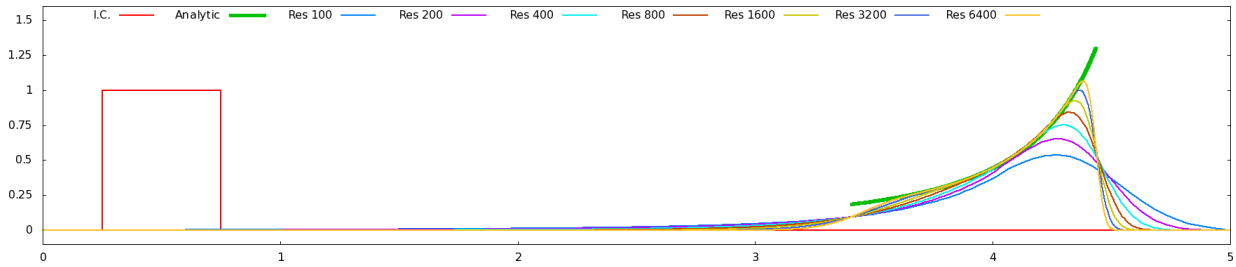
Figure 3: A sinusoidal wave is advected through a constant velocity field, $u = 1$, using a variety of spatial discretizations and grids, with TVD-RK3 time integration. Results are shown at $t = 4s$.



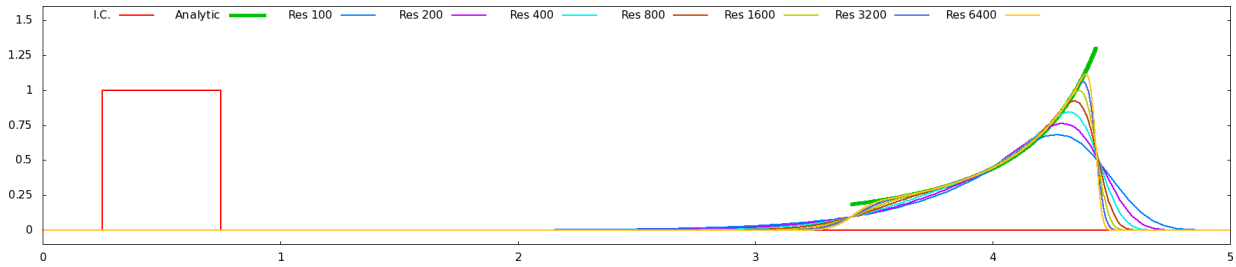
(a) First order accurate ENO-LLF on a uniform grid, with a CFL number $\alpha = .5$.



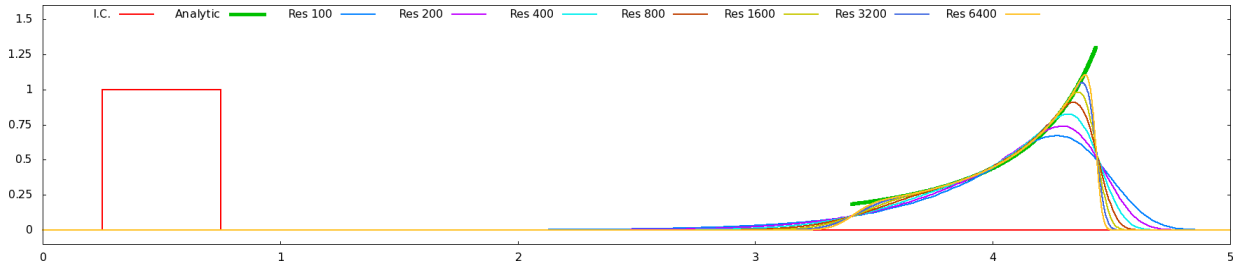
(b) Conservative semi-Lagrangian scheme on a uniform grid, with a CFL number $\alpha = .5$.



(c) First order accurate ENO-LLF on the non-uniform grid shown in Figure 2, with a CFL number $\alpha = .5$.



(d) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 2, with a CFL number $\alpha = .5$.



(e) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 2, with a CFL number $\alpha = .5$ taken with respect only to the coarse grid cells (and so the *effective* CFL number is 4).

Figure 4: A square wave is advected through a divergent velocity field, $u(x) = \sin\left(\frac{\pi}{5}x\right)$ using a variety of spatial discretizations and grids, with TVD-RK3 time integration. Results are shown at $t = 5s$.

The flow field is partitioned into two regions; a flux region, where the flow is updated entirely by fluxes computed on control volume boundaries, and a semi-Lagrangian region, where the flow is updated using the conservative semi-Lagrangian scheme. As flux-based schemes already prescribe material transport across control volume boundaries, we use the fluxes on faces that separate the two flow regions in order to determine how they interact. These boundary fluxes are computed using information from both sides of the interface, and completely determine how the two regions interact.

In the semi-Lagrangian region, the boundary flux must be accounted for in such a way that the method remains numerically conservative in its clamping and forward-advection stages. Consider a hybrid control volume boundary that separates a flux-region cell i and a semi-Lagrangian region cell j, where the flux $\vec{\mathcal{F}}_{ij}$ is imposed as a boundary condition on the semi-Lagrangian region. If the area-weighted surface normal of this control volume boundary is $d\vec{A}_{ij}$ (where the normal points into the semi-Lagrangian region), then the quantity of $\hat{\phi}$ moving across the hybrid control volume boundary can be computed from the fluxes as

$$\hat{\phi}_i^n \hat{w}_{ij} = \Delta t \vec{\mathcal{F}}_{ij} \cdot d\vec{A}_{ij} \qquad \text{or} \qquad \hat{\phi}_j^n \hat{w}_{ji} = -\Delta t \vec{\mathcal{F}}_{ij} \cdot d\vec{A}_{ij}$$

This material enters the semi-Lagrangian cell if $\vec{u}_f \cdot \vec{A}_{ij} \geq 0$, and leaves the cell otherwise, where $\vec{u}_f$ is the velocity at the flux face.

When incorporating the boundary conditions into the volumetric conservative semi-Lagrangian scheme, one might consider recovering $\hat{w}_{ij}$ by dividing through by $\hat{\phi}_i^n$ (or $\hat{\phi}_j^n$). However, if $\hat{\phi}_i^n = 0$ (as is the case for our one-dimensional advection examples) then this is infeasible. Instead we modify Equations (5) and (6), absorbing the $\hat{\phi}_i^n$ term into the $\hat{w}_{ij}$ term as $\tilde{w}_{ij}$. In the previous version of these equations $\hat{w}_{ij}$ had units of volume and was clamped against the volume of the cell, $\|\Omega_i\|$. With this new approach we are clamping $\hat{\phi}_i^n \hat{w}_{ij}$, which represents the fluxed material, against the total material in cell i (after accounting for boundary conditions).

The modified conservative semi-Lagrangian scheme accounts for boundary conditions in two ways. Hybrid boundary fluxes pushing material into a semi-Lagrangian cell j are handled by adding in new material weights $\tilde{w}_{\mathcal{F}ij} = \Delta t \vec{\mathcal{F}}_{ij} \cdot d\vec{A}_{ij}$, which are set aside until the final stage of the update. Hybrid boundary fluxes pulling material out of a semi-Lagrangian cell j are accounted for by subtracting off that material ($\tilde{w}_{j\mathcal{F}_i} = -\Delta t \vec{\mathcal{F}}_{ij} \cdot d\vec{A}_{ij}$) from the total amount of material in cell j, and so the remaining material in that cell is $K_j = \hat{\phi}_j^n \|\Omega_j\| - \sum_i \tilde{w}_{j\mathcal{F}_i}$. Cells that do not lie adjacent to the hybrid boundary remain unmodified and so the remaining material in that cell is simply $K_j = \hat{\phi}_j^n \|\Omega_j\|$.

Next, the backward-cast weights w_{ij} are computed as discussed in Figure 1(a). We only look back from cells in the semi-Lagrangian region and, as all of the material transport across the hybrid interface is accounted for by the fluxes, we discard any weights w_{ij} where either cell i or cell j are not semi-Lagrangian cells. These weights are then multiplied by $\hat{\phi}$, and we compare $\hat{\sigma}_i = \hat{\phi}_i^n \sum_j w_{ij}$ with K_i. If $|\hat{\sigma}_i| > |K_i|$—that is, if cell i gives too much material to the time t^{n+1} solution—then we scale the weights by $K_i/\hat{\sigma}_i$, setting $\tilde{w}_{ij} = (K_i/\hat{\sigma}_i)\hat{\phi}_i^n w_{ij}$ for these cells. Note that, unlike before, we clamp the magnitude of the sum of the weights. This is to properly account for the case where $\hat{\phi}_i^n < 0$.

If $|\hat{\sigma}_i| < |K_i|$—that is, the cell does not give sufficient material to the time t^{n+1} solution—then forward-advected weights f_{ij} are computed. If f_{ij} carries material across the hybrid boundary then it is discarded, and all remaining weights are scaled up accordingly (If there are no remaining weights then the remaining material is simply left in that cell, i.e. by setting $f_{ii} = 1$). These forward-advected weights are used to carry any remaining material forward as before, giving final weights for these cells as $\tilde{w}_{ij} = \hat{\phi}_i^n w_{ij} + (K_i - \hat{\sigma}_i)\tilde{f}_{ij}$, where $\tilde{f}_{ij}$ are the scaled up f_{ij} weights that carry material to cells that are in the semi-Lagrangian region at time t^{n+1} (i.e. $\sum_{j \in \text{s-L}} \tilde{f}_{ij} = 1$).

To summarize, the hybrid scheme first computes fluxes within the flux region, and at the hybrid boundary between the flux and semi-Lagrangian regions. The flux region is then updated using these flux values, completing the update for these cells. At the hybrid boundary, we compute $\tilde{w}_{\mathcal{F}ij} = \Delta t \vec{\mathcal{F}}_{ij} \cdot d\vec{A}_{ij}$ and $\tilde{w}_{j\mathcal{F}_i} = -\Delta t \vec{\mathcal{F}}_{ij} \cdot d\vec{A}_{ij}$ as discussed above. Hybrid boundaries that draw material out of the semi-Lagrangian region are then used to modify K_j by $\sum_i \tilde{w}_{j\mathcal{F}_i}$. Backward-cast weights w_{ij} are computed for semi-Lagrangian

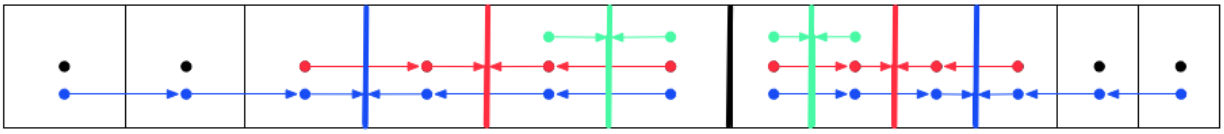


Figure 5: In the hybrid advection scheme, near areas where the cell size changes, we drop the stencil width (and therefore the order of accuracy) of the flux scheme to avoid crossing the refinement interface. In the one-dimensional example illustrated here, the blue flux faces are solved using a third order accurate scheme. The red faces are solved with a second order accurate scheme, while the green faces are computed with simple upwinding. The stencils for these faces are also illustrated, just to show that they do not cross the refinement interface. The thick black fluid face represents the refinement boundary between the larger cells on the left and the smaller cells on the right, and none of our first, second or third order accurate stencils cross that boundary. One could update this flux with a first order accurate ENO scheme, similar to the stencils shown in green to obtain what we refer to as a graded discretization near the refinement boundary. We instead use our conservative semi-Lagrangian scheme on the cells to the left and right of this face, with their K_j's modified based on the neighboring first order fluxes shown in green.

cells j, and any weights that cross the hybrid flux boundary are discarded. Forward-cast weights f_{ij} are computed for any cells where $|\hat{\sigma}_i| < |K_i|$, and again any weights that cross the hybrid flux boundary are discarded. The remaining forward-cast weights f_{ij} are scaled up to $\tilde{f}_{ij}$ such that $\sum_{j \in \text{s-L}} \tilde{f}_{ij} = 1$, and the material-weighted weights are

$$\tilde{w}_{ij} = \begin{cases} (K_i/\hat{\sigma}_i)\hat{\phi}_i^n w_{ij} & |\hat{\sigma}_i| > |K_i| \\ \hat{\phi}_i^n w_{ij} + (K_i - \hat{\sigma}_i)\,\tilde{f}_{ij} & |\hat{\sigma}_i| \leq |K_i|. \end{cases} \tag{7}$$

Finally we update the cells in the semi-Lagrangian region as

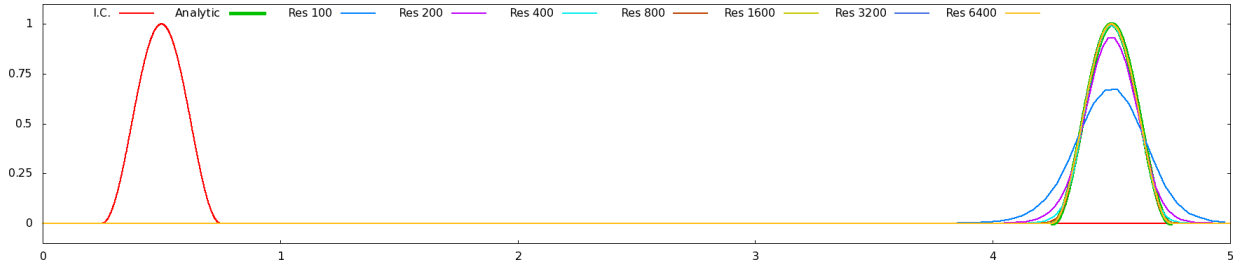
$$\hat{\phi}_j^{n+1} = \|\Omega_j\|^{-1} \left[\sum_i \tilde{w}_{ij} + \sum_i \tilde{w}_{\mathcal{F}ij} \right]. \tag{8}$$

As both the flux scheme and the conservative semi-Lagrangian scheme are conservative, the resulting hybrid scheme is also fully conservative.

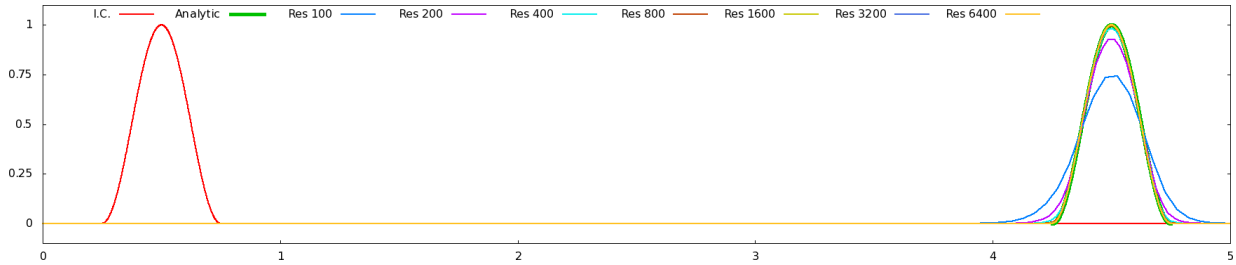
2.4. Examples

We noticed that, in the non-linear examples considered later in this paper, numerical artifacts manifest when the order of accuracy of the flow solver drops dramatically between neighboring cell faces – these include artificial reflected waves, kinks and overshoots. With this in mind, although the hybrid formulation is general enough to support any flux-based scheme, we prefer a flux-based scheme with a variable-width stencil over one that uses a fixed-width stencil such as WENO [29, 21]. That way, in the non-linear examples we can "step down" the stencil width (and therefore order of accuracy) of the flow solver as it approaches a hybrid flux boundary – see Figure 5. Unless otherwise stated we use ENO-LLF.

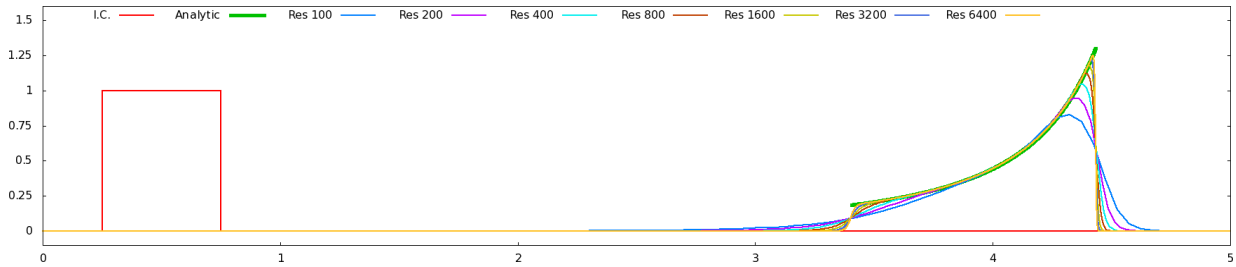
We return to the examples discussed in Section 2.2 on the non-uniform grid given in Figure 2, and utilize the proposed hybrid scheme in order to employ a third order accurate ENO-LLF scheme away from the refinement interfaces located at $x = \{1, 1.5, 2, 3, 3.5, 4\}$. Since these problems are linear the aforementioned difficulties with numerical artifacts do not exist, and therefore for the sake of exposition we take a more aggressive non-graded approach. The semi-Lagrangian scheme is only utilized on a lower-dimensional manifold of the computational domain—that is, only 36 cells—and so the results are expected to be high resolution in nature. Figures 6(b) and 6(d) show the results for the sine-wave and square-wave bumps respectively, and indeed the convergence of the peak value of the solution is only slightly degraded from that of a third order accurate ENO-LLF scheme on a uniform grid. The peak value of the sine-wave bump converges to its analytic value at a rate of 1.98 (as compared to 2.56), while the peak value of the square-wave bump converges to its analytic value at a rate of .63 (as compared to .57). Note that, for the square-wave, the discontinuity located immediately to the right of the peak value negatively impacts convergence even for a traditional scheme on a uniform grid.



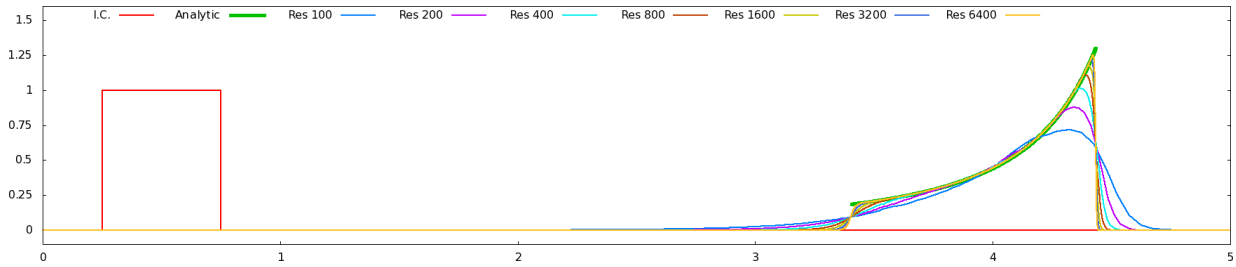
(a) Baseline sinusoidal wave advected through a constant velocity field computed using third order accurate ENO-LLF on a uniform grid, with a CFL number $\alpha = .5$.



(b) A sinusoidal wave advected through a constant velocity field, $u = 1$, to final time $t = 4s$.



(c) Baseline square wave advected through a divergent velocity field computed using third order accurate ENO-LLF on a uniform grid, with a CFL number $\alpha = .5$.



(d) A square wave advected through a divergent velocity field, $u(x) = \sin\left(\frac{\pi}{5}x\right)$, to final time $t = 5s$.

Figure 6: Conservative advection is solved on the non-uniform grid shown in Figure 2 using TVD-RK3 time integration and a hybrid spatial discretization. The semi-Lagrangian regions are limited to a three-cell band near refinement interfaces, and the bulk of the flow field is treated using third order accurate ENO-LLF.

3. Semi-implicit compressible flow formulation

As demonstrated in [26], a conservative semi-Lagrangian advection scheme can be combined with the implicit pressure solve of [24] to solve the compressible Euler equations. This first order accurate advection scheme alleviates the time step restriction imposed by the bulk advection, resulting in a method which imposes no time step restrictions for stability. The advection scheme tends to introduce significant numerical dissipation throughout the flow, however, and was previously limited to uniform grids. We modify the method of [24] and use our proposed modifications both for non-uniform grid refinement and hybrid advection.

Consider the Euler equations, given by

$$\begin{pmatrix} \rho \\ \rho\vec{u} \\ E \end{pmatrix}_t + \begin{pmatrix} \nabla \cdot \rho\vec{u} \\ \nabla \cdot (\rho\vec{u} \otimes \vec{u}) \\ \nabla \cdot E\vec{u} \end{pmatrix} + \begin{pmatrix} 0 \\ \nabla p \\ \nabla \cdot p\vec{u} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{9}$$

where we have split the flux terms into an advection and non-advection part. The advection part is integrated explicitly to give intermediate values $\rho^\star$, $(\rho\vec{u})^\star$ and $E^\star$, and since pressure does not affect the continuity equation we can set $\rho^{n+1} = \rho^\star$. The resulting momentum update equation is divided by ρ^{n+1}, giving

$$\vec{u}^{n+1} = \vec{u}^\star - \Delta t \frac{\nabla p^{n+1}}{\rho^{n+1}}, \tag{10}$$

and its divergence is taken to obtain

$$\nabla \cdot \vec{u}^{n+1} = \nabla \cdot \vec{u}^\star - \Delta t \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right). \tag{11}$$

The pressure evolution equation (see [12]), which is given by

$$p_t + \vec{u} \cdot \nabla p = -\rho c^2 \nabla \cdot \vec{u}, \tag{12}$$

is semi-discretized by fixing $\nabla \cdot \vec{u}$ to time t^{n+1} through the time step and by treating advection terms explicitly. Denoting the advected pressure field by $p^a = p^n - \Delta t \vec{u} \cdot \nabla p$ gives

$$p^{n+1} = p^a - \Delta t \rho c^2 \nabla \cdot \vec{u}^{n+1}. \tag{13}$$

Substituting this in to Equation (11) and rearranging gives

$$p^{n+1} - \Delta t^2 \rho^n (c^2)^n \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right) = p^a - \Delta t \rho^n (c^2)^n \nabla \cdot \vec{u}^\star, \tag{14}$$

where we have fixed ρc^2 to time t^n. We compose the $\rho^n (c^2)^n$ terms into a diagonal matrix $\mathbf{P} = \left[\Delta t^2 \rho^n (c^2)^n \right]$ and discretize the gradient and divergence operators, yielding

$$\left[\mathbf{P}^{-1} + G^T (\hat{\rho}^{n+1})^{-1} G \right] \hat{p}^{n+1} = \mathbf{P}^{-1} \hat{p}^a + G^T \vec{\hat{u}}^\star, \tag{15}$$

where G is the discretized gradient operator, $-G^T$ the corresponding discretized divergence operator, the pressures are scaled by Δt (i.e. $\hat{p} = p\Delta t$), and $\hat{\rho}$ and $\hat{u}$ represent variables interpolated to cell faces. This implicit system is solved to obtain p^{n+1} at cell centers. These time t^{n+1} pressures are then applied in a flux-based manner to the intermediate momentum and energy values to obtain time t^{n+1} quantities in a discretely conservative manner, giving correct shock speeds. This is done as follows. Pressures are averaged using a density weighting in order to compute the pressure at cell faces as

$$\hat{p}_{i+1/2} = \frac{\rho_i \hat{p}_{i+1} + \rho_{i+1} \hat{p}_i}{\rho_i + \rho_{i+1}} \tag{16}$$

and face velocities at time t^{n+1} are computed by rewriting the momentum update using face-averaged quantities as

$$\hat{u}_{i+1/2}^{n+1} = \hat{u}_{i+1/2}^{\star} - \hat{\rho}_{i+1/2}^{-1} G_{i+1/2} \hat{p}^{n+1}, \tag{17}$$

where $G_{i+1/2}$ is the row of G corresponding to face $i+1/2$ and $\hat{u}_{i+1/2}^{\star} = \frac{(\rho u)_i^{\star} + (\rho u)_{i+1}^{\star}}{\rho_i^{\star} + \rho_{i+1}^{\star}}$. The flux-based implicit update then takes the form

$$(\rho \vec{u})_i^{n+1} = (\rho \vec{u})_i^{\star} - \frac{\hat{p}_{i+1/2}^{n+1} - \hat{p}_{i-1/2}^{n+1}}{\Delta x}, \qquad E_i^{n+1} = E_i^{\star} - \frac{\hat{p}_{i+1/2}^{n+1}\hat{u}_{i+1/2}^{n+1} - \hat{p}_{i-1/2}^{n+1}\hat{u}_{i-1/2}^{n+1}}{\Delta x}. \tag{18}$$

3.1. Computing the advected pressure

In previous work, $p^a = p^n - \Delta t \vec{u} \cdot \nabla p$ was computed using Hamilton-Jacobi ENO [24]. When doing so they noticed Gibbs phenomena near the shock front (which can be seen in the left column of Figure 7), and in order to mitigate these oscillations a MAC grid-based ENO (or MENO) variant of ENO was introduced. We do not use MENO in any of our examples; instead, we compute the advected pressure as

$$p^a = p(\rho^{\star}, e^{\star}),$$

i.e. by using the equation of state directly on the post-advected flow-field. This appears to reduce oscillations near the shock front – see the right hand column of Figure 7. The reduced oscillations may be due to p^a being more consistent with the advection step, although some Gibbs phenomena features still appear in the momentum and energy, most likely due to the centrally differenced nature of the pressure update. This variant method of computing p^a appears to be beneficial in a number of our tests, but we did not extensively experiment for example with highly non-linear equations of state. This approach is also more efficient, having removed the Hamilton-Jacobi advection step.

3.2. Modified ENO-GLF scheme

In the higher spatial dimension examples considered later on in Section 5.3, we noticed that the third order accurate ENO-GLF (that is, global Lax-Friedrich's diffusion) scheme sometimes artificially cavitated (with internal energy going negative) during the flux-based advection update. This occured at Mach stems such as the one highlighted in Figure 8, and only for the third order accurate variant of ENO-GLF; first and second order accurate variants of ENO-GLF do not cavitate. We believe that this is due to dispersive errors in the flow field introduced by the Lax-Friedrich's diffusion term, due to those terms being evaluated for a higher order derivative than that of the flux gradient itself. Thus, when the dominant errors in the evaluation of the flux are dissipative, the dominant errors in the Lax-Friedrich's term are dispersive leading to numerical difficulties (of course, this analysis only rigorously applies in the linear sense).

Through experimentation, we found that truncating the divided difference table for the Lax-Friedrich's term to that of a lower order polynomial appears to alleviate this numerical cavitation. That is, we compute globally valid divided difference tables for $\mathcal{F}$ and $\hat{\phi}$ as

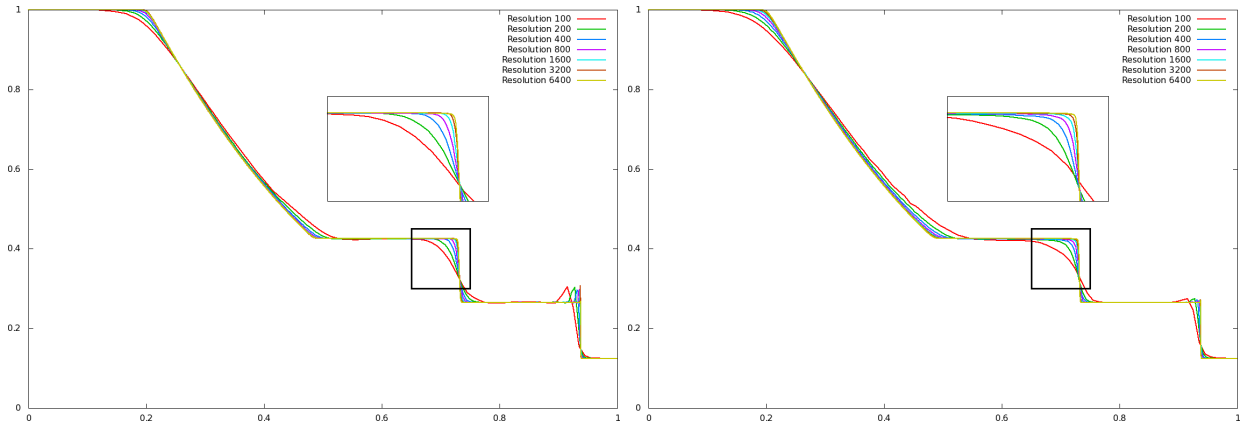
$$D_i^1 \mathcal{F} = \mathcal{F}(\hat{\phi}_i) \qquad\qquad D_i^1 \hat{\phi} = \hat{\phi}_i$$

$$D_{i+1/2}^2 \mathcal{F} = \frac{D_{i+1}^1 \mathcal{F} - D_i^1 \mathcal{F}}{2\Delta x} \qquad\qquad D_{i+1/2}^2 \hat{\phi} = \frac{D_{i+1}^1 \hat{\phi} - D_i^1 \hat{\phi}}{2\Delta x}$$

$$D_i^3 \mathcal{F} = \frac{D_{i+1/2}^2 \mathcal{F} - D_{i-1/2}^2 \mathcal{F}}{3\Delta x}$$

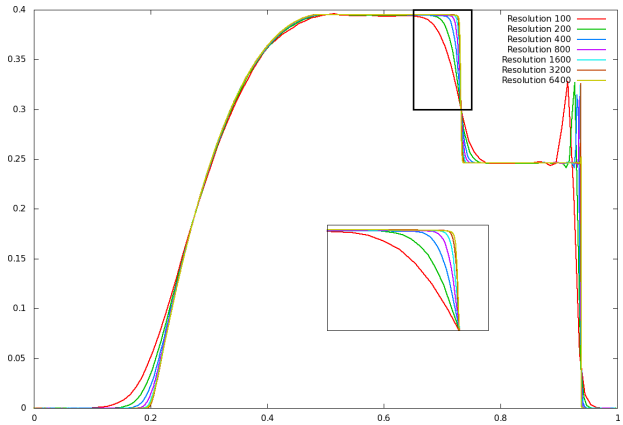
where, unlike the traditional third order accurate variant, we prescribe $D_i^3 \hat{\phi}$ to be zero; this is equivalent to fixing the degree of the polynomial approximating the diffusion term to be at most of order 2. The left-upwinded and right-upwinded flux terms ($\mathcal{F}_{i+1/2}^+$ and $\mathcal{F}_{i+1/2}^-$, respectively) are then computed using the locally valid divided difference tables, which are given by

$$D_i^1[\mathcal{F} \pm \alpha\hat{\phi}] = D_i^1 \mathcal{F} \pm \alpha D_i^1 \hat{\phi}, \qquad D_{i+1/2}^2[\mathcal{F} \pm \alpha\hat{\phi}] = D_{i+1/2}^2 \mathcal{F} \pm \alpha D_{i+1/2}^2 \hat{\phi}, \quad \text{and} \quad D_i^3[\mathcal{F} \pm \alpha\hat{\phi}] = D_i^3 \mathcal{F}.$$

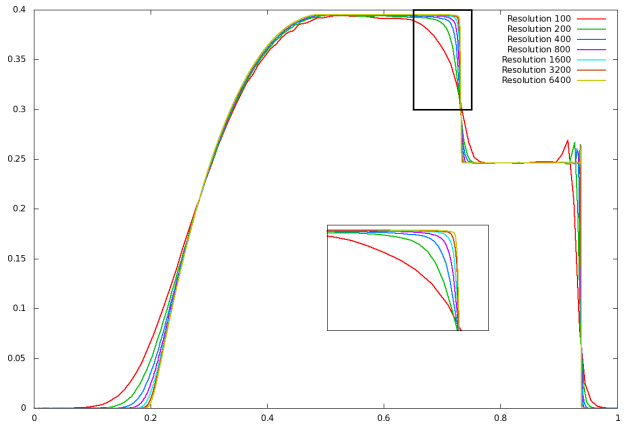


(a) Density from HJ-ENO.

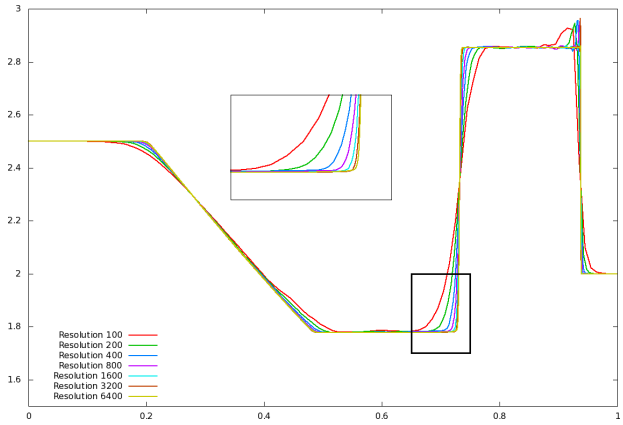
(b) Density from $p^a = p(\rho^\star, e^\star)$.



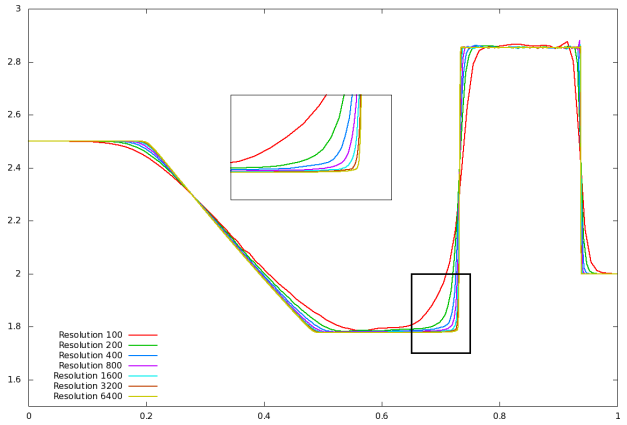
(c) Momentum from HJ-ENO.



(d) Momentum from $p^a = p(\rho^\star, e^\star)$.



(e) Internal energy from HJ-ENO.



(f) Internal energy from $p^a = p(\rho^\star, e^\star)$.

Figure 7: A comparison of the impact of how p^a is computed. On the left column a third order accurate Hamilton-Jacobi ENO scheme is used to compute $p^a = p^n - \Delta t \vec{u}^n \cdot \nabla p^n$. On the right, the advected pressure is computed directly from the post-advected fluid state $\vec{U}^\star$ – that is, $p^a = p(\rho^\star, e^\star)$. Both methods capture the shock location properly, by virtue of being conservative, but the second approach appears to have significantly reduced overshoots at the shock front. Both solutions are computed on uniform grids using TVD-RK3 time integration and standard third order accurate ENO-GLF to handle advection. The CFL number α is .5, and results are shown for $t = .25s$.

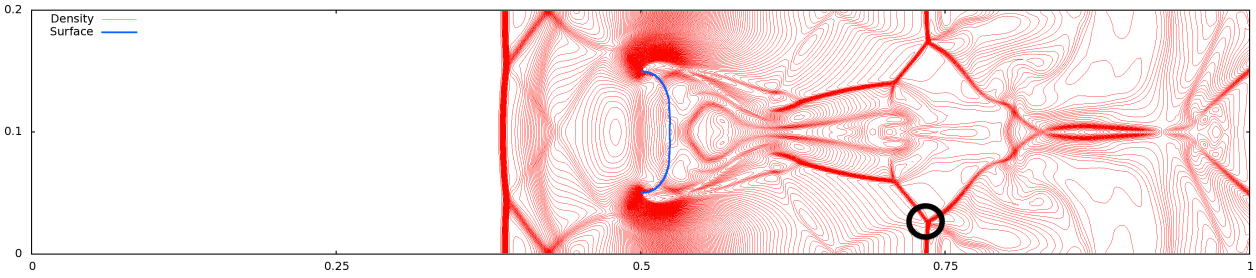


Figure 8: In this two-dimensional example (explored in further detail in Section 5.3), artificial cavitation occurs in the circled region, behind a Mach stem, when the standard Lax-Friedrich's third order accurate variant of ENO is used to compute $\vec{U}^\star$. Shown are isocontours for density for the example described in Section 5.3.5 at time $t = .2s$.

We designate this *truncated* diffusion version of ENO-GLF as ENO-GLFT. We compare ENO-GLFT with the standard second order accurate variant of ENO-GLF in Figure 9 to show the impact of factoring in the third order accurate terms for $\mathcal{F}$ but not the third order accurate diffusion term. The standard second order accurate variant is shown in the left column, and ENO-GLFT is shown in the right column. There are no significant differences in the rarefaction region, $x \leq .5$, and minimal differences near the shock front near $x = .937$. However, near the linearly degenerate contact discontinuity at $x = .734$ significant improvement in accuracy can be seen. Although not shown, no discernible difference is seen in the pressure or velocity – both of which are continuous at linearly degenerate contact discontinuities. The ENO-GLFT method also compares well with standard third order accurate ENO-GLF, shown as the right-hand column of Figure 7. The most noticeable difference appears at the shock front, where the overshoots for ρ and e are reduced in magnitude as a result of this modification. This benefit is secondary, however, to the improved stability for the examples discussed in Section 5.3. Using our modified variant to third order accurate ENO-GLF no artificial cavitation occurs, unlike in the case where the standard third order accurate ENO-GLF is used.

3.3. Non-uniform grids

In order to write the momentum update for a dual cell, we begin by distributing each cell-centered momentum term evenly between their axis-appropriate left and right cell faces. That is, ρu is evenly distributed to the x-axis cell faces and ρv to the y-axis cell faces. The momentum for an x-axis cell face can then be written as

$$\beta_{i+1/2}\hat{u}_{i+1/2} = \frac{1}{2}\left[V_i(\rho u)_i + V_{i+1}(\rho u)_{i+1}\right]$$

where $\hat{u}$ is the velocity component associated with the dual cell, V are the cell volumes, and β is the effective mass of the dual cell computed as the sum of the masses related to each component of momentum, i.e. $\beta_{i+1/2} = \frac{1}{2}(V_i\rho_i + V_{i+1}\rho_{i+1})$. Note that if there is a Neumann boundary on a cell face, only the density and momentum from the fluid side is used. Next we define the volume-weighted divergence operator as $-\hat{G}^T$ by multiplying the regular divergence operator through by the volume of the cell (see for example [31, 32]). Then the negative transpose of this is the gradient operator $\hat{G}$, which can be used to write the momentum update for the dual cells as

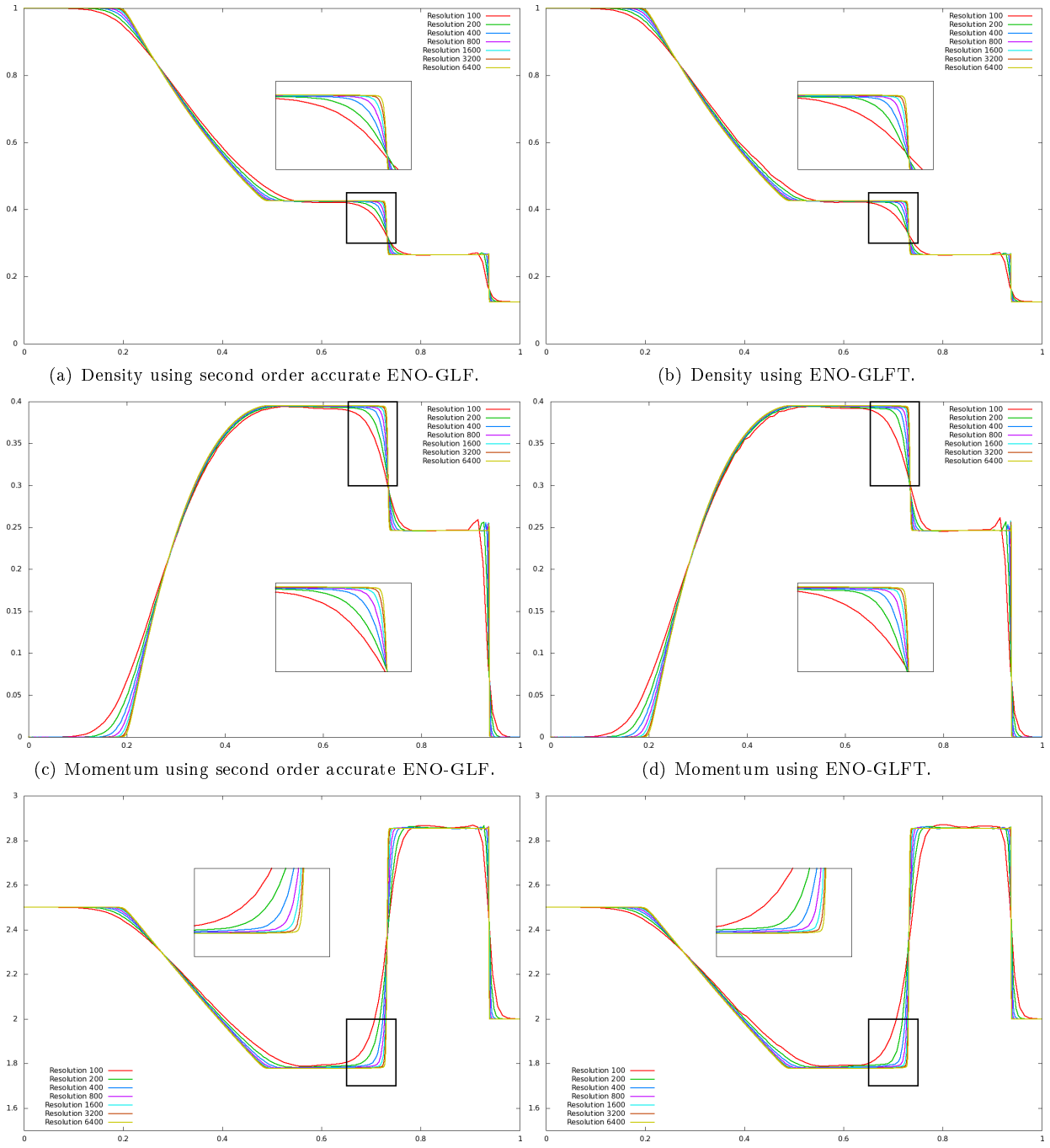
$$\vec{\hat{u}}^{n+1} = \vec{\hat{u}}^\star - \beta^{-1}\hat{G}\hat{p}^{n+1}. \tag{19}$$

Taking its volume-weighted divergence with $-\hat{G}^T$ gives

$$-\hat{G}^T\vec{\hat{u}}^{n+1} = -\hat{G}^T\vec{\hat{u}}^\star + \hat{G}^T\beta^{-1}\hat{G}\hat{p}^{n+1}. \tag{20}$$

And following the derivation of Equation (15) then leads to

$$\left[\hat{\mathbf{P}}^{-1} + \hat{G}^T\beta^{-1}\hat{G}\right]\hat{p}^{n+1} = \hat{\mathbf{P}}^{-1}\hat{p}^a + \hat{G}^T\vec{\hat{u}}^\star \tag{21}$$



(a) Density using second order accurate ENO-GLF.

(b) Density using ENO-GLFT.

(c) Momentum using second order accurate ENO-GLF.

(d) Momentum using ENO-GLFT.

(e) Internal energy using second order accurate ENO-GLF.

(f) Internal energy using ENO-GLFT.

Figure 9: A comparison of the impact on how the advection step is spatially discretized. In the left column a standard second order accurate ENO-GLF is used to compute the advective fluxes, while in the right column the ENO-GLFT of Section 3.2 is used. There is no significant difference near the rarefaction region, $x \leq .5$, and the difference near the shock (located near $x = .937$) is minimal. At the contact discontinuity, however, a significantly faster convergence rate is seen – indeed these results compare qualitatively well with those depicted in the right-hand column of Figure 7, where an *unmodified* version of the third order accurate ENO-GLF scheme is used. The CFL number α is .5, and results are shown for $t = .25s$.

where G and G^T are replaced by $\hat{G}$ and $\hat{G}^T$, the density and velocity at cell faces are computed using the volume-lumping explained in the beginning of this subsection, and the diagonal matrix $\hat{\mathbf{P}}^{-1} = V/[\Delta t^2 \rho^n (c^2)^n]$ is the volume-scaled version of $\mathbf{P}^{-1}$ above.

Once this implicit pressure $\hat{p}^{n+1}$ is computed, we apply it back to the conserved variables in a flux-based manner in order to remain conservative. The updates for momentum and energy for a given cell i can be written

$$(\rho\vec{u})_i^{n+1} = (\rho\vec{u})_i^{\star} - \frac{\hat{p}_{i+1/2}^{n+1} - \hat{p}_{i-1/2}^{n+1}}{\Delta} \qquad \text{and} \qquad E_i^{n+1} = E_i^{\star} - \frac{\hat{p}_{i+1/2}^{n+1}\hat{u}_{i+1/2}^{n+1} - \hat{p}_{i-1/2}^{n+1}\hat{u}_{i-1/2}^{n+1}}{\Delta} \tag{22}$$

where both $\hat{p}^{n+1}$ and $\hat{u}^{n+1}$ are evaluated at face locations, and the discretization width, Δ, can be recovered by dividing the volume associated with the cell V_i by the dual cell face area $A_{i-1/2} = A_{i+1/2}$, i.e. $\Delta = V_{i+1/2}/A_{i+1/2}$. Note our cross-sectional areas A do not change, and are always equal to either Δx or Δy depending on the dimension. The pressure for a given cell face $i+1/2$, $\hat{p}_{i+1/2}^{n+1}$, is computed as a density-weighted average of the two cells adjacent to the face as in Equation (16), and the face velocity $\hat{u}^{n+1}$ is recovered from Equation (19) as

$$\hat{u}_{i+1/2}^{n+1} = \hat{u}_{i+1/2}^{\star} - \beta_{i+1/2}^{-1}\hat{G}_{i+1/2}\hat{p}^{n+1}, \tag{23}$$

similar to Equation (17) with G replaced by $\hat{G}$, $\hat{\rho}$ replaced by β, and the $\vec{\hat{u}}^{\star}$ computed using the volume-lumping explained in the beginning of this subsection.

If one knows the Neumann velocity $\hat{u}_{i+1/2}^{n+1}$ for a constrained face, then the pressure gradient $\hat{G}_{i+1/2}\hat{p}^{n+1}$ at that face can be recovered from Equation (19) using $\beta_{i+1/2}$ and $\hat{u}_{i+1/2}^{\star}$ from above. At a constrained Neumann face we can compute

$$\hat{p}_{i+1/2}^{n+1} = \hat{p}_i^{n+1} + d\hat{G}_{i+1/2}\hat{p}^{n+1},$$

where d is the distance from the cell center i to the cell face $i+1/2$ (see for example [32]). Note, however that d is a $\mathcal{O}(\Delta x)$ term and that typically only a lower-dimensional manifold of cell faces are constrained as Neumann faces, and thus one could also set $\hat{p}_{i+1/2}^{n+1} = \hat{p}_i$.

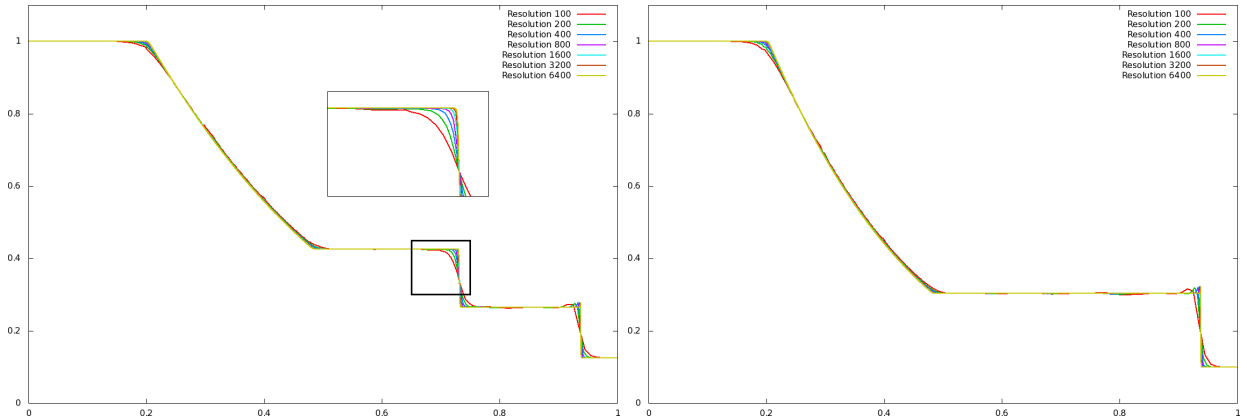
3.4. Sod's shock example

We consider Sod's shock with initial conditions specified over a computational domain of $[0,1]$ as

$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (1,0,1) & \text{if } x \leq .5, \\ (.125,0,.1) & \text{if } x > .5, \end{cases}$$

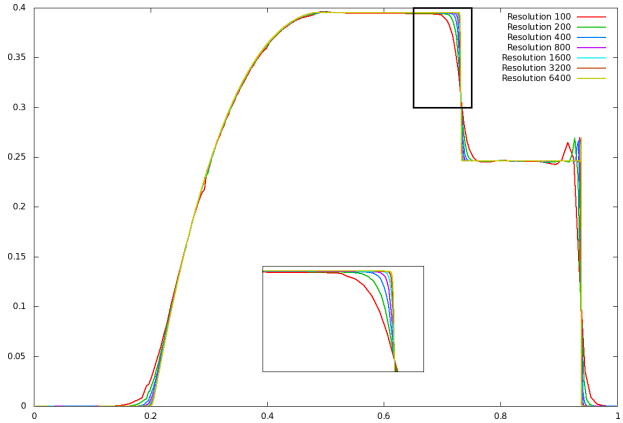
where the baseline solution, computed on a uniform grid using ENO-GLFT and the improved computation for advected pressure, was shown in the right column of Figure 9. Here we consider Sod's shock under non-uniform grid refinement using the grid pattern shown in Figure 2, except that the domain is $[0,1]$ rather than the depicted $[0,5]$. Each of the refinement boundaries, now located at $x \in \{.2,.3,.4,.6,.7,.8\}$, are treated as described in Figure 5—that is, ENO stencils in the vicinity of these refinement boundary faces are reduced in width so as not to cross refinement faces, and the cells immediately to the left and right of the refinement face represent the semi-Lagrangian region. The resulting flow field at $t = .25s$ is shown in Figure 10. We could also treat the region within $x \in [.2,.8]$ as the semi-Lagrangian region and take 8 times larger time steps, using only the coarsest grid cells to dictate the time step. The resulting flow field is shown in Figure 11 and demonstrates the unconditional stability of the hybrid semi-implicit solver on the non-uniform grid.

It is interesting to note the over-heating that appears in the internal energy shown in Figure 11(e), near $x = .875$. This seems to be generated when a shock passes from the fully refined semi-Lagrangian region in $x \in [.2,.8]$ into the flux-based region. After the shock passes through that hybrid interface, the overheating peak passively advects with the flow. An isobaric fix such as the one discussed in [13] may alleviate these peaks, but doing so would sacrifice conservation. Similar but less pronounced issues occur in Figure 10(e) when the scheme used to capture the numerical shock changes.

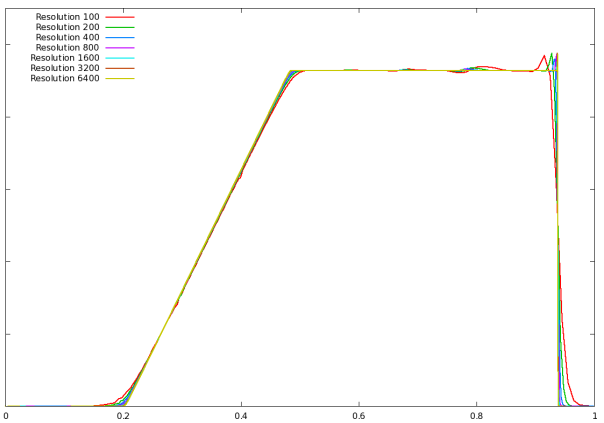


(a) Density

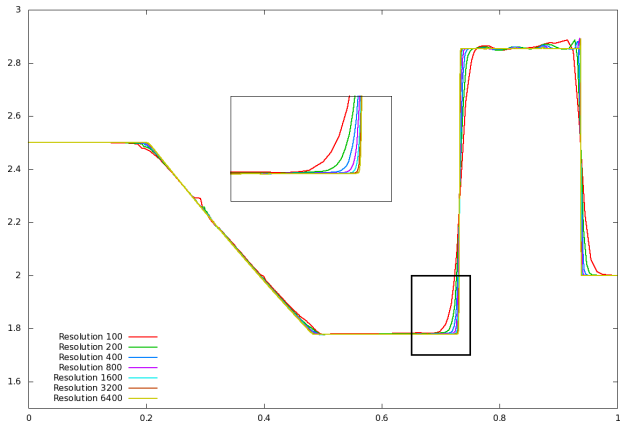
(b) Pressure



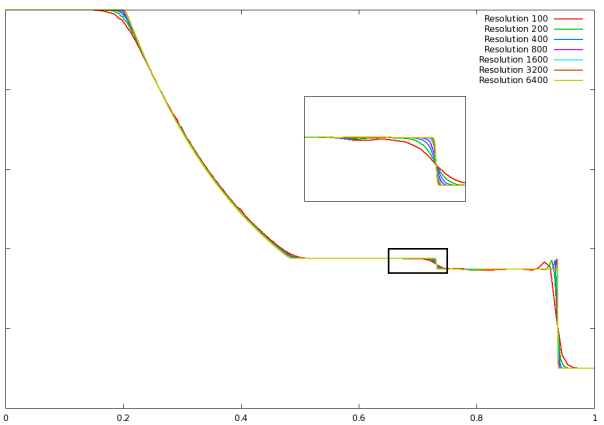
(c) Momentum



(d) Velocity

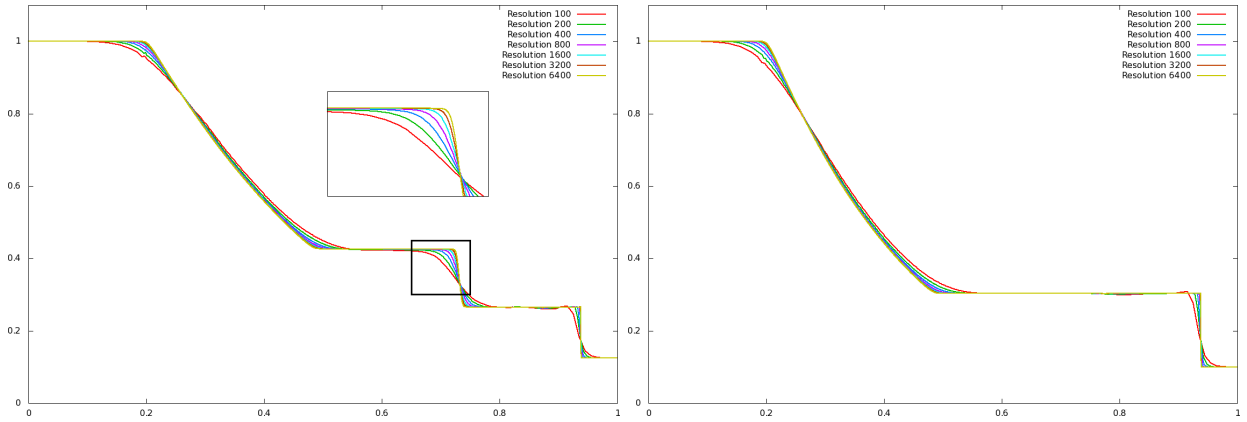


(e) Internal Energy

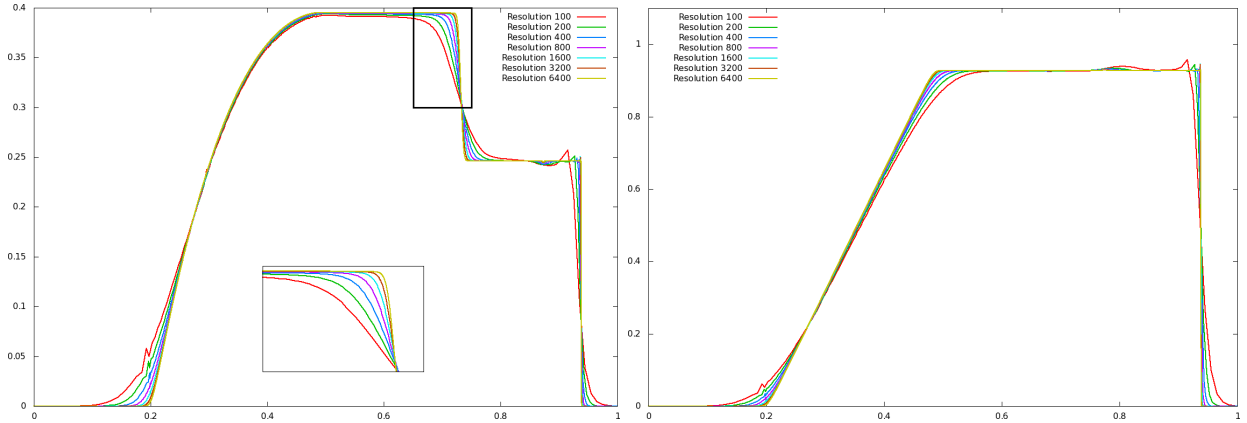


(f) Total Energy

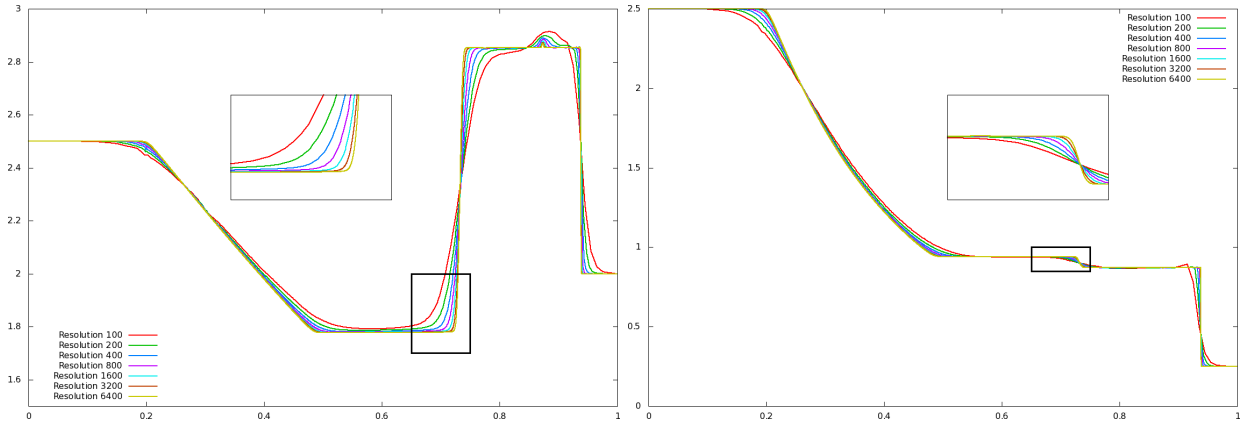
Figure 10: Results for a Sod shock at $t = .25s$, when computed via the semi-implicit formulation on a non-uniformly refined grid, using the hybrid advection scheme and TVD-RK3 time integration. The twelve cells at the refinement boundary represent the semi-Lagrangian region, and the remainder of computational domain are solved using ENO-GLFT (dropping the order of accuracy locally as discussed in Figure 5).



(a) Density (b) Pressure



(c) Momentum (d) Velocity



(e) Internal Energy (f) Total Energy

Figure 11: Results for a Sod shock at $t = .25s$, when computed via the semi-implicit formulation on a non-uniformly refined grid, using the hybrid advection scheme and TVD-RK3 time integration. All cells between $x = .2$ and $x = .8$ are treated using the conservative semi-Lagrangian advection scheme, and the time step is dictated only by the coarse grid cell sizes.

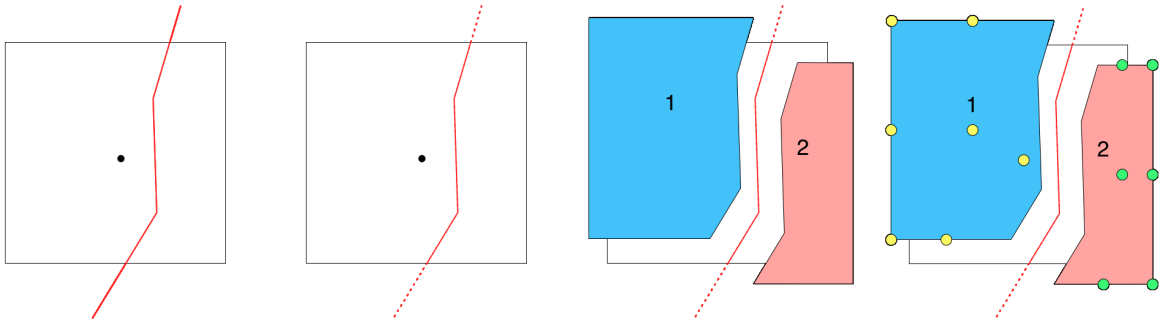


Figure 12: When a grid cell is cut by a structure interface (shown as a red segmented curve), we first clip the segments of the curve against cell boundaries as shown in the second figure. Then the clipped interface is stitched together with contiguous components of the cell volume boundary, yielding the cut cells shown in as the red and blue polygons in the third figure. Finally, the visibility sample points are computed as the significant features of the polygon that do not lie on the structure interface, and lie inside the cut cell polygon Ω or on the cut cell's boundary $\partial\Omega$ (see the yellow and green dots in the last figure).

4. Fluid-structure interactions

We modify the fluid-structure solver of [14] by incorporating cut cells and partial cell volumes into both the explicit advection and the implicitly coupled stage of the solver. This is done without introducing any new degrees of freedom; instead cut cells are populated on the fly, advected using the conservative semi-Lagrangian scheme, and then their material is redistributed back to cell-centered degrees of freedom. Since the semi-Lagrangian scheme is only used in a thin band of cells near the structure interface, standard high resolution results are obtained in the bulk of the flow field.

4.1. Computing cut cells

Fluid grid cells that are cut by the structure interface are divided into a number of partial cell volumes, and each of these polygonal regions are assigned some sample points to aide in the computation of visibility (see Sections 4.2 and 4.3). While a number of techniques exist in the literature, we use a straightforward approach where the simplices of the structure interface are first clipped to a cell volume and then stitched together with the cell volume boundary to form the cut cell volumes. This is trivial in one spatial dimension, and illustrated for two spatial dimensions in Figure 12. Visibility sample points are computed for these polygonal regions in an ad hoc fashion by identifying the significant features such as the nodes and face centers of the cut cell geometry, centroid, etc., that do not lie along the structure interface—see e.g. the yellow and green dots shown in the fourth subfigure of Figure 12. For all non-cut cells in the semi-Lagrangian region, we use the cell center for visibility as necessary. In general any approximate interface reconstruction suffices so long as some estimate of cell volume can be given, and some ability to determine visibility is provided. Note that we do not consider three spatial dimensions in this paper, however there is no intrinsic three-dimensional limitation to the algorithm as described.

We do not consider three spatial dimensions in this paper, and instead refer the interested reader to [4] or [45] where cut cell generation techniques are discussed.

4.2. Material lumping

The goal of material lumping is to make sure that all cut cells are associated with a cell-centered degree of freedom. The left-hand polygonal region shown in Figure 12 contains its own cell center, and so is trivially associated with it. The right-hand polygonal region shown in Figure 12 on the other hand does not contain a cell center, and moreover cannot be lumped onto the cell center associated with its own grid cell as that cell center lies on the wrong side of the structure interface. Instead this polygon is lumped onto visible adjacent degrees of freedom. In order to determine which adjacent degrees of freedom are visible we look at all orthogonally adjacent neighbors and cast rays from the sample points of the cut cell to the adjacent cell centers; if any of the rays do not intersect the solid interface, which is fixed in time, then that neighbor is

visible (note that adjacent cells may also be a cut cell that contains its own cell center degree of freedom, and so the left-hand polygon from Figure 12 could also absorb material from neighboring cut cells). If none of the orthogonally adjacent neighbors are visible we look to diagonally adjacent neighbors, and if none of those are visible we discard the polygon and its volume. Note that this does not violate conservation except at time t^0 as we are careful not to assign any material to these cut cells during advection—see Section 4.4. If this is a problem at time t^0 one could still enforce conservation by manually moving this material to adjacent degrees of freedom.

Once we have a list of all visible neighbors, whether it be a number of orthogonally adjacent neighbors or alternatively a number of diagonally adjacent neighbors, we compute a conservative material distribution operator. This is done by putting an axis-aligned bounding box around the polygon, and using the ratio between the sides of the rectangle to determine how much volume is distributed to each of the visible adjacent cells. Let A_L and A_R be the lengths of the left and right sides of the rectangle (where $A_L = A_R$), and A_T and A_B are the lengths of the top and bottom sides of the rectangle (where $A_T = A_B$), setting the corresponding value to zero if that orthogonal cell center is not visible. The remaining weights are normalized so that they sum to one, and then used to conservatively distribute the cut cell's volume to surrounding cell-centered degrees of freedom. In the case where no orthogonally adjacent cells are visible and instead only diagonally adjacent cells are visible, we set all weights equal, i.e. the volume is equally distributed to all visible diagonal cells. After distribution the per-unit-volume quantities at the cell-centered degrees of freedom are changed in order properly account for the lumped material. For example, the new density ρ at the cell center can be computed as

$$\rho^{\text{new}} = \frac{M + \bar{M}_{\text{cut}}}{V + \bar{V}_{\text{cut}}} \tag{24}$$

where M and V is the cell mass and volume respectively before lumping, $\bar{M}_{\text{cut}}$ is the mass lumped onto the cell center from all neighboring cut cells and $\bar{V}_{\text{cut}}$ is the total volume lumped onto the cell center from neighboring cut cells. Equation (24) is also applied to the momentum and energy. At this point all material and volume has been associated with a standard Cartesian degree of freedom.

4.3. Temporal Visibility

Now that we have computed our cut cells and lumped orphaned volumes onto neighboring degrees of freedom where possible, we next re-address visibility from the standpoint of what is required for advection. Advection occurs between time t^n and time t^{n+1} and thus we build a temporal visibility map using continuous collision detection [7, 15] as opposed to the static solid positions used above in Section 4.2. Recall that all cells near the structure interface will be treated with the semi-Lagrangian method; then we say that a time t^n full or cut cell i and a time t^{n+1} full or cut cell j can interact with each of other if there exists at least one pair of sample points that one can travel between during the time step without colliding with the moving structure—this is where continuous collision detection is used. Conceptually speaking we place a particle at the time t^n sample point and give it a constant velocity equal to the displacement vector to the time t^{n+1} sample point divided by the size of the time step. Then if this particle collides with the moving solid interface at some time τ, these sample points cannot reach each other. For our purposes we linearize the motion of each segment of the solid structure and check the particle collision against all segments which are near it (e.g. that are within one grid cell of the particle over the time step; this can be determined quickly via acceleration structures). If a given time t^n cut cell cannot see anything at time t^{n+1}, then that cut cell and its volume is discarded; this does not violate conservation as the conserved quantities can be left on cell-centered degrees of freedom (i.e. that material is never "unlumped" from the cell-centered degree of freedom). Note that a collision can only occur if the three involved points become colinear at some time $\tau \in [t^n, t^n + \Delta t]$, and a secondary check must be done to see if the colinear particle lies within the line segment. Very robust algorithms exist for doing this, as can be seen by the collisions in the cloth simulations of [7] and the water and cloth simulations in [15]. In three spatial dimensions one must determine τ as the times of coplanarity for four particles, and this requires solving for the roots of a cubic polynomial. We caution the reader that those authors discovered that double precision was required to solve the resulting cubic as opposed to single

precision, which was not accurate enough to capture all collisions. Moreover closed-form solutions for the cubic are not accurate enough and one must instead use an iterative solver.

4.4. Advection

As discussed in Section 4.2 volume from cut cells was lumped on to cell-centered degrees of freedom. At the beginning of the advection stage these degrees of freedom return the same exact volume that they received back to their cut cells, along with a proportional amount of their current material. That is, the mass given to a cut cell is $\rho^n V_{\text{cut}}$, where V_{cut} was the volume lumped onto the degree of freedom from that cut cell and ρ^n is the current density at the cell center. Momentum and energy is handled in the same manner. In particular, note that cut cells receive the same volume that they gave, but different amounts of material.

For any cut cell that has a computed geometry, we use its bounding box to transform the complicated geometry into a Cartesian cell (i.e. a rectangle). Thus at time t^n we have a collection of Cartesian cells, those from the regular grid along with the cut cells bounding boxes, which we note may overlap each other. Then for a given full or cut cell at time t^{n+1}, the center of its rectangle is traced backwards along its characteristic curve and is intersected against time t^n full and cut cells bounding boxes (as described in Figure 1). Note that overlapping cut cells do not change our algorithm, we simply calculate the overlap with every cell and apply this scheme as if there were no overlap. The backward-cast ray may cross over the structure interface and permit flow to leak across it, and one could improve the guess for the end-point of the backward-cast ray by taking into account the moving structure interface via continuous collision detection, recording the collision location on the structure interface, and then following that position on the structure back to time t^n. In fact, as we are tracing volumes back in time one might even consider using continuous collision detection on the boundary of the volume to compute a squished backward-cast volume, similar in spirit to VOF and ALE methods, but this quickly becomes quite complicated (especially in three spatial dimensions). A somewhat simpler approach would be to only collide the cell center and then try to reconstruct a traced-back Cartesian cell by sending out rays to each of its four corners, colliding these rays with the time t^n structure interface to create a quadrilateral—though even that quadrilateral's overlap with other cells requires scrutiny as parts of the quadrilateral volume may lie on the wrong side of the interface even if the four corners do not. We instead propose a simple approach that uses the temporal visibility map from Section 4.3. After tracing back the cell center, whether one wishes to collide it with the moving solid structure or not (for more accuracy), we simply place the orthogonally aligned cell (or cut cell bounding box) at the foot of the characteristic as in Figure 1. Then when calculating overlap, again as in Figure 1, we simply ignore cells that are not visible to the original cell according to the visibility map. The clamping step of advection does not change, and all of the statements made above for backward advection also hold for the forward advection step. That is, we simply push our point forward along its characteristic curve, potentially colliding it with the time-evolved structure for more accuracy, and compute the weights as the overlap of rectangles discarding any that are not visible according to the temporal visibility map. Unlike backward-advection if weights are discarded in the forward advection step then the remaining forward-advected weights f_{ij} are scaled up accordingly so that no material is lost, providing for conservation. If all weights are discarded then we can distribute material to any nearby cells which are allowed for by the temporal visibility map. One could consider not only those near the destination, but also those along the characteristic curve and near the original cell itself. If this fails one might need to go back and reconsider the temporal visibility map itself, possibly placing more sample points in the cells or expanding how one searches. Generally speaking we are providing a strategy, and have found that the simplest possible version of that strategy works for our examples, however for completeness we are describing how one should proceed as examples become more complex.

Whereas advection proceeds with every full and cut cell treated as a normal cell, afterwards the material and volume in cut cells are distributed back to cell-centered degrees of freedom as described above in Section 4.2. If the visibility map at time t^{n+1} contains a cut cell that currently has material in it but cannot be lumped onto any adjacent neighbors, then we alter the previously described advection scheme to not advect material into this cell—otherwise that material would be lost, violating conservation. Note that our treatment works by computing weights locally, and so does not work in the case of some structure-structure collision where material is instantaneously forced to move rather large distances as it is squeezed out of a

region of the computational domain, or in areas of unresolved curvature where a structure folds in upon itself. This represents an area of future work, however hypothetically speaking our strategy seems to extend to these cases, although the required code might contain complexities we have not anticipated. It is worth noting that this problem does not seem to have been addressed in the literature.

4.5. High resolution time integration near the fluid-structure interface

As the volume associated with a given fluid degree of freedom changes or vanishes, the state averaging that Runge-Kutta time integrators rely on to achieve high order accuracy breaks down, violating conservation and—if the structure is thin—mixing flow variables from both sides of the structure interface. One might consider a sophisticated solver that volume-weights the state variables, but unfortunately this still fails when the volume associated with a particular degree of freedom vanishes entirely. This issue only arises near the fluid-structure interface and so we still use Runge-Kutta in the flux-based region of the flow.

We couple the semi-Lagrangian solver for the region near the structure interface with the flux-based Runge-Kutta solver in the bulk of the domain as follows. First consider second order Runge-Kutta (RK2), which can be applied by taking two forward-Euler steps and then linearly interpolating between the solution at time t^n and time t^{n+2} to obtain the solution at time t^{n+1}. To hybridize this with our semi-Lagrangian scheme we take the first Euler substep for both the flux-based and semi-Lagrangian regions as usual. Then the second Euler step can be taken for the flux-based scheme and the averaging can be performed to obtain a final time t^{n+1} solution in that region. Note that the actual flux that takes one from time t^n to the final time t^{n+1} in RK2 is simply the average of the two computed fluxes, and thus we can use that averaged flux as the effective flux to step forward the flux-based region, obtaining the same answer. Thus we can use that averaged flux as the boundary condition for the semi-Lagrangian scheme, which can then take an Euler step forward from time t^n to time t^{n+1} in order to obtain the final solution in that region.

Third order TVD Runge-Kutta (TVD-RK3) proceeds similarly as follows. Similar to RK2 one takes two Euler steps, and one semi-Lagrangian step would be needed to compute the second Euler step in the flux-based region, and then time averaging is done in that region to obtain a solution at time $t^{n+1/2}$. The effective flux for this is the same as above, albeit for half a time step, and so exactly as in RK2 we use this as a boundary condition to evolve the semi-Lagrangian region forward by $\Delta t/2$ to time $t^{n+1/2}$. Then we have data everywhere in the domain in order to take another Euler step in the flux-based region to go from time $t^{n+1/2}$ to time $t^{n+3/2}$, at which point one can take a linear average between that solution and the time t^n solution to compute the final solution in the flux-based region at time t^{n+1}. Once again this can be seen as a single time step with an averaged flux, which is 1/6 of the first and second fluxes and 2/3 of the third flux, and use this averaged flux as a boundary condition on the semi-Lagrangian region to evolve that region of the flow to time t^{n+1}. Unlike RK2 where all the temporal visibility information from time t^n to time t^{n+1} is enough, as the semi-Lagrangian region only evolves between these two states, in this case one needs to also compute all the temporal visibility information at the time $t^{n+1/2}$, as the semi-Lagrangian region must be evolved to this state in TVD-RK3. The solid evolution also needs to be done to time $t^{n+1/2}$ and we simply take the linearly averaged state (e.g. $X_S^{n+1/2} = \frac{1}{2}[X_S^n + X_S^{n+1}]$) and use effective velocities. Finally, note that our treatment of time integration near the flux boundary is not aimed particularly at addressing accuracy, rather the goal of having a TVD-RK3 method is as much of the domain as possible is to provide an enhanced stability region—one typically applies RK3 to other compressible flow problems for the same reason, an enhanced stability region.

4.6. Solid evolution

The solid state is completely described by its velocity $V_S(t)$ and position $X_S(t)$, and we update the position and velocities separately in a Newmark-style scheme. Note that throughout the paper when we refer to an *effective* velocity, we do not mean V_S but rather $(X_S^{n+1} - X_S^n)/\Delta t$, a velocity based entirely on displacements. First the velocity is evolved for half a time step to $V_S^{n+1/2}$, and then this intermediate velocity is used to update the position via $X_S^{n+1} = X_S^n + \Delta t V_S^{n+1/2}$. Finally, the velocity is reset to time

t^n and evolved for a full time step Δt to compute the time t^{n+1} state. For deforming bodies we update the velocity via

$$M_S V_S^{n+1} = M_S V_S^n + \Delta t F(X_S^n, V_S^{n+1}).$$

where M_S is the mass matrix and F are the forces, which are treated explicitly in position via X_S^n and implicitly in velocity via V_S^{n+1}. Applying Taylor series on the velocity term of $F(\cdot,\cdot)$ gives

$$M_S V_S^{n+1} = M_S V_S^n + \Delta t(F(X_S^n, V_S^n) + D(X_S^n)[V_S^{n+1} - V_S^n])$$

where $D = \frac{\partial F}{\partial V_S}$ are the linear damping terms. We then explicitly compute

$$M_S V_S^\star = M_S V_S^n + \Delta t\left[F(X_S^n, V_S^n) - D(X_S^n)V_S^n\right]$$

and implicitly solve

$$M_S V_S^{n+1} = M_S V_S^\star + \Delta t D(X_S^n)V_S^{n+1}. \tag{25}$$

Note that since X_S^{n+1} was already computed before solving for V_S^{n+1} one could also use X_S^{n+1} rather than X_S^n when computing $F(\cdot,\cdot)$ and $D(\cdot)$, for enhanced stability. Note also that $V_S^{n+1/2}$ is computed in the same manner as V_S^{n+1} including both linearization and the implicit solve, however for $V_S^{n+1/2}$ only X_S^n exists and therefore one cannot use a future X_S for enhanced stability. A rigid body has significantly fewer degrees of freedom than a deforming body with its center-of-mass and orientation as positional degrees of freedom, and center-of-mass velocity and angular velocity as velocity degrees of freedom. Using generalized mass and velocity its equations can be treated in a manner similar to the deformable case, except that there is no intrinsic damping and no need for the implicit solve. For more details on rigid and deformable body simulation in regards to two-way solid-fluid coupling see [42].

4.7. Coupled time evolution

Our time integration scheme proceeds as follows. Using the time t^n solid position, we first determine all fluid faces which need to be constrained to the solid as discussed in Section 4.8. Note that a later step will consist of determining those at the solid's time t^{n+1} position, and therefore one can use the time t^{n+1} constraints from one time step as the time t^n constraints at the next time step (i.e. this only needs to be done once per time step). Next we explicitly compute the intermediate solid momentum $M_S V_S^\star$, after which we formulate and solve the solid-fluid coupled system (see Section 4.8) to find the intermediate solid momentum $M_S V_S^{n+1/2}$, which is used to update the position from X_S^n to X_S^{n+1} before the velocity is reset to its original time t^n values. This gives our final time t^{n+1} position of the solid, which we then use to once again determine which fluid cell faces need to be constrained. Next we need to update both the fluid state and solid momentum to time t^{n+1}. This is done by first advecting the fluid forward in time using the fully conservative interpenetration-free advection algorithm of Section 4.4 in the Runge-Kutta style time integration of Section 4.5. We also compute the intermediate momentum for the solid $M_S V_S^\star$. Finally we once again solve the coupled system (of Section 4.8) to find the final time t^{n+1} momentum of the solid, as well as the final time t^{n+1} fluid state.

4.8. Implicit monolithic system

Constraints are identified through a ray-casting approach, where for every fluid cell we cast a ray from its cell center to the center of each of its orthogonal neighboring cells; if the ray intersects the structure interface then a constraint is introduced. We use a restriction operator $\mathbf{W}$ that acts on fluid grid cell faces and returns a vector of constrained faces. Note that there may be up to two constraints per cell face, if there is fluid on both sides of the structure interface. These constraints are treated independently, and do not interact except through the solid constitutive model. The other matrix of interest is $\mathbf{J}$, which is a conservative interpolation operator that maps solid velocity degrees of freedom to constraint locations [43].

At each constrained face we enforce velocity continuity. Writing this as an equation for all constrained faces results in

$$\mathbf{J}V_S^{n+1} - \mathbf{W}\hat{u}^{n+1} = 0. \tag{26}$$

This constraint is enforced using a set of impulses λ that are exchanged between the fluid and the structure. Equation (25) for the structure becomes

$$M_S V_S^{n+1} = M_S V_S^\star + \Delta t D V_S^{n+1} - \mathbf{J}^T \lambda. \tag{27}$$

and the corresponding equations for the fluid become

$$\vec{\hat{u}}^{n+1} = \vec{\hat{u}}^\star - \beta^{-1} \hat{G} \hat{p} + \beta^{-1} \mathbf{W}^T \lambda, \tag{28}$$

Since the row sums of both $\mathbf{J}^T$ and $\mathbf{W}^T$ are unitary, any momentum lost by the structure is recovered by the fluid and vice versa, making the method fully conservative in momentum.

Then we assemble the following linear system

$$\begin{pmatrix} \hat{\mathbf{P}}^{-1} + \hat{G}^T \beta^{-1} \hat{G} & -\hat{G}^T \beta^{-1} \mathbf{W}^T & 0 \\ -\mathbf{W} \beta^{-1} \hat{G} & \mathbf{W} \beta^{-1} \mathbf{W}^T & -\mathbf{J} \\ 0 & -\mathbf{J}^T & -M_S + \Delta t D \end{pmatrix} \begin{pmatrix} \hat{p}^{n+1} \\ \lambda \\ V_S^{n+1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{P}}^{-1} \hat{p}^a + \hat{G}^T \vec{\hat{u}}^\star \\ -\mathbf{W} \vec{\hat{u}}^\star \\ -M_S V_S^\star \end{pmatrix}, \tag{29}$$

where the equations for the fluid pressure come from the derivation of Equation (21) incorporating the additional λ term from Equation (28), the equations for the solid velocity come from Equation (27), and the middle row of equations come from substituting Equation (28) in to Equation (26). For newly swept or uncovered pressure degrees of freedom, we replace the $[\rho^n (c^2)^n]$ terms of $\mathbf{P}$ with $[\rho^\star (c^2)^\star]$. This is the *only* special treatment given to swept and uncovered cell-centered degrees of freedom.

Following [42] we also assume that $-\Delta t D = \hat{C}^T \hat{C}$, i.e. that the structure damping matrix is symmetric negative definite. We introduce new degrees of freedom $f = \hat{C} V_S$, and Equation (27) can be premultiplied by M_S^{-1}, giving

$$V_S^{n+1} = V_S^\star - M_S^{-1} \hat{C}^T f^{n+1} - M_S^{-1} \mathbf{J}^T \lambda, \tag{30}$$

Substituting Equation (30) into the middle set of rows in Equation (29) and multiplying the third row by $\hat{C} M_S^{-1}$ gives

$$\begin{pmatrix} \hat{\mathbf{P}}^{-1} + \hat{G}^T \beta^{-1} \hat{G} & -\hat{G}^T \beta^{-1} \mathbf{W}^T & 0 \\ -\mathbf{W} \beta^{-1} \hat{G} & \mathbf{W} \beta^{-1} \mathbf{W}^T + \mathbf{J} M_S^{-1} \mathbf{J}^T & \mathbf{J} M_S^{-1} \hat{C}^T \\ 0 & \hat{C} M_S^{-1} \mathbf{J}^T & I + \hat{C} M_S^{-1} \hat{C}^T \end{pmatrix} \begin{pmatrix} \hat{p}^{n+1} \\ \lambda \\ f^{n+1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{P}}^{-1} \hat{p}^a + \hat{G}^T \vec{\hat{u}}^\star \\ \mathbf{J} V_S^\star - \mathbf{W} \vec{\hat{u}}^\star \\ f^\star \end{pmatrix}. \tag{31}$$

After solving this symmetric positive-definite system we recover the solid velocities via Equation (30), and are careful to keep the momentum and kinetic energy exchange conservative as explained in detail in [14].

5. Examples

We consider a number of one-dimensional and two-dimensional fluid-structure examples. In the discussion below all units are presented in SI (that is, mass in kg, velocity in m/s etc.). In all of the examples discussed, we found the implicit system to be well-behaved as a result of the large diagonal terms, and so we use a simple diagonal preconditioner when solving Equation (31), i.e. we row-scale and column-scale the matrix. The matrix form of our preconditioner is

$$\begin{pmatrix} \hat{\mathbf{P}} & 0 & 0 \\ 0 & \left(\mathbf{W}\beta^{-1} + \mathbf{J}M_S^{-1}\right)^{-1} & 0 \\ 0 & 0 & I \end{pmatrix},$$

and the resulting system requires only between 11 and 36 iterations of Conjugate Gradients to converge to numerical round-off, depending on the structural model. The $\hat{\mathbf{P}}$ term alone brought down the number of iterations from on the order of 150 to around 13, and adding the $\left(\mathbf{W}\beta^{-1} + \mathbf{J}M_S^{-1}\right)^{-1}$ only reduced the average by about 2 iterations. We did not further experiment with preconditioning, as the $\hat{\mathbf{P}}$ alone seems good enough, however we did use the matrix given above in our simulations, even though the middle term provided only marginal improvement.

5.1. One-dimensional Sod shock coupled with a thin rigid body of varying mass

A one-dimensional rigid point-mass is inserted into a Sod shock tube simulation, where the flow is initially prescribed over $x \in [-1, 3]$ as

$$(\rho, u, p) = \begin{cases} (1, 0, 1) & \text{if } x \leq .5, \\ (.125, 0, .1) & \text{if } x > .5 \end{cases}.$$

The solid is initially positioned at $X_S = 1.3001$, and is hit by the shock at approximately $t = .57s$. We consider a broad range of mass choices for the point mass, from infinitesimally light ($M_S = 10^{-6}$) to extremely heavy ($M_S = 10^6$), and a selection in-between ($M_S \in \{10^{-2}, 10^{-1}, .25, .5, .75, 1, 2.5, 7.5, 10, 10^2\}$). In the dynamic examples, where the solid is light enough ($M_S < 10$), we check the convergence of the position of the solid body and verify that in each case convergence is linear (see e.g. Figure 14). We show some results in Figures 15, 16, 17 and 18.

For each choice in solid mass, we verify that no fluid mass moves across the structure interface, and we also verify that within the computational domain no momentum or energy is created or destroyed. In order to verify conservation of mass, we compute

$$\mathcal{M}_L = \sum_{x_i < X_S} \rho_i V_i \qquad \mathcal{M}_R = \sum_{x_i > X_S} \rho_i V_i,$$

the total fluid masses to the left and right of the solid, respectively, and plot a time history of the variation from initial value in Figure 13(c). The figure shows the time history for $M_S = 1$ which is representative of other mass choices, with the total left fluid mass never varying more than 2×10^{-13} and right fluid mass never varying more than 4×10^{-14} for any selection of solid mass, within numerical round-off.

In order to compute the total error in momentum of the system, we note that until the rarefaction and shock reach the domain boundaries the expected total momentum introduced into the system by time t^N is exactly $.9 \cdot t^N$, where .9 is the difference in pressure values from the left and right boundary conditions. Then the total error in conservation of momentum at time t^N can be written as

$$\sum_i (\rho u)_i^N V_i^N + M_S V_S^N - .9 \cdot t^N,$$

and we show a time history of this error (for $M_S = 1$) in Figure 13(a). This is representative of other mass choices, with the total momentum of the system for any mass choice never varying more than 1.2×10^{-13}.

As the velocity remains zero at the left and right boundaries of the domain, no work is done and the total energy of the system should not vary from its initial value. We compute the total energy of the system as

$$\sum_i E_i V_i + \frac{1}{2} M_S V_S^2$$

and show a time history of the difference of this value from its original value (for $M_S = 1$) in Figure 13(b). This is representative of other mass choices, with the total energy of the system never varying more than 2.5×10^{-13} from its initial value.

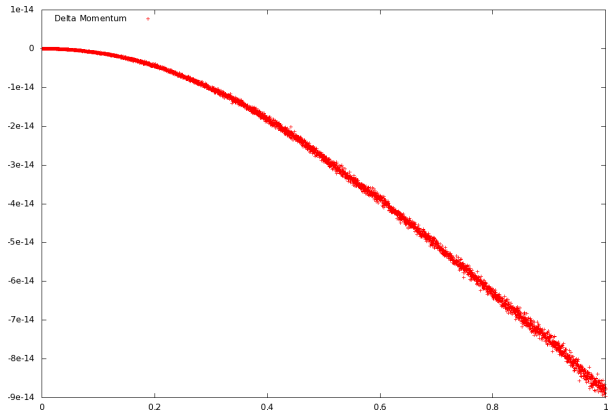
5.2. Stationary fluid damping a mass-spring system

In order to validate our results, we also consider the fluid-damped mass-spring system introduced in [1], where a high pressure fluid acts as a damping force on a spring fixed to the right side of the domain. The spring is of length 1 with spring coefficient $k = 10^7$. The fluid state is initialized over $x \in [0, 20]$ as

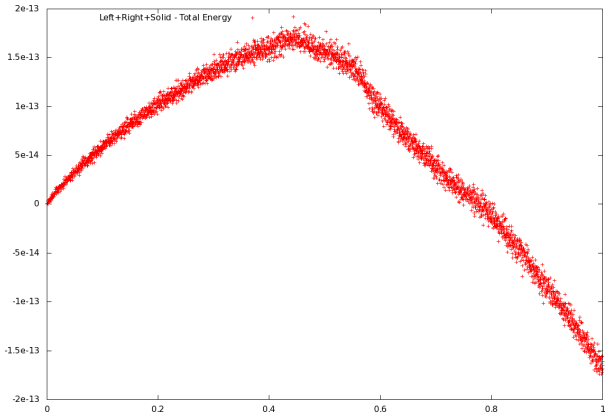
$$(\rho, u, p) = \left(4, 0, 10^6\right),$$

and at $t = 0$ the spring is released from rest, resulting in over-damped behavior. We show the position of the free end of the spring in Figure 20, and plot the fluid pressure for a selection of times in Figure 19.

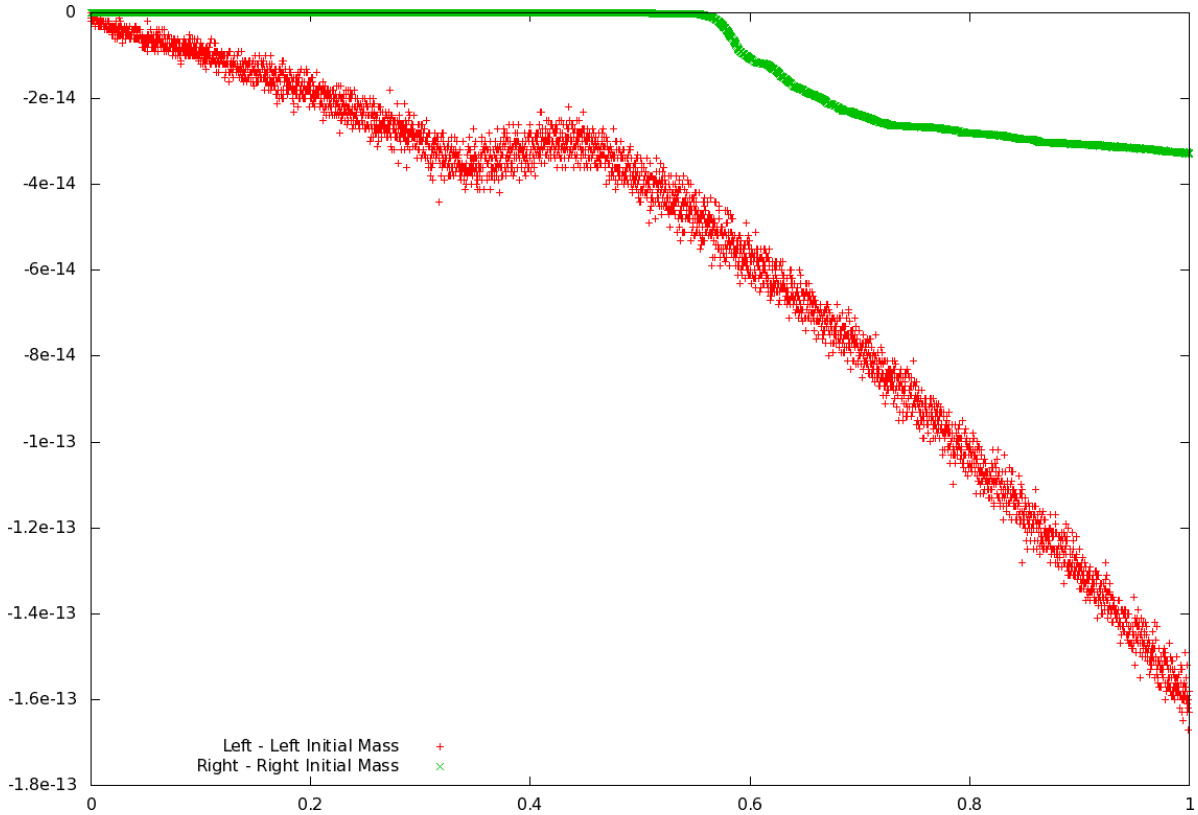
The convergence of the free end of the spring to the analytic solution at times $t = .0075s$ and $t = .008$ are shown in Figure 21, suggesting quadratic convergence to the analytic solution. This represents a significant improvement over [14], which only achieves convergence at a rate of 1.16 for this example, and is likely a result of the pressure gradient and divergence operators being discretized to high order accuracy as the result of carefully considering cut cells. Interestingly, unlike [14] no oscillations are seen in the left-moving shock front that spontaneously forms at $t = .0045s$; this is likely due to the improvements in the flow solver discussed in Sections 3.1 and 3.2.



(a) Error in conservation of momentum of the system, computed as $\sum_i (\rho u)_i^N V_i^N + M_S V_S^N - (.9 \cdot t^N)$, where $(.9 \cdot t^N)$ is the increase in total momentum of the system at time t^N due to pressure differences at the boundary.

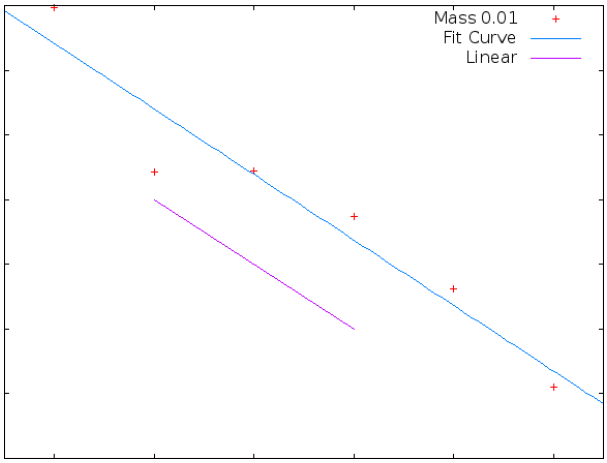


(b) Error in conservation of energy of the system, computed as $\sum_i E_i V_i + \frac{1}{2} M_S V_S^2$. As the velocity of the fluid at the boundary remains zero, no work is done and the total energy of the system is constant.

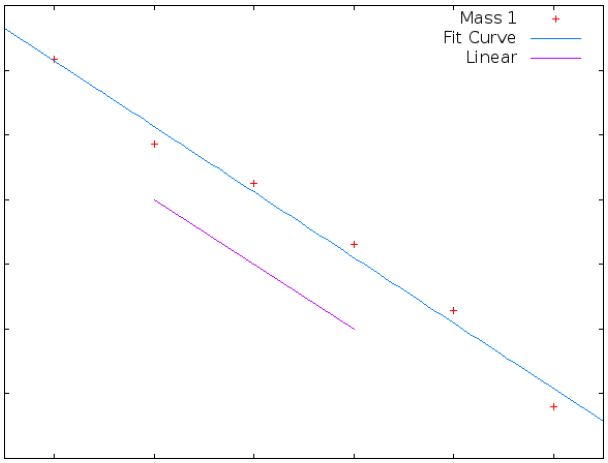


(c) Error in conservation of mass for the left (green) and right (red) sides of the domain.

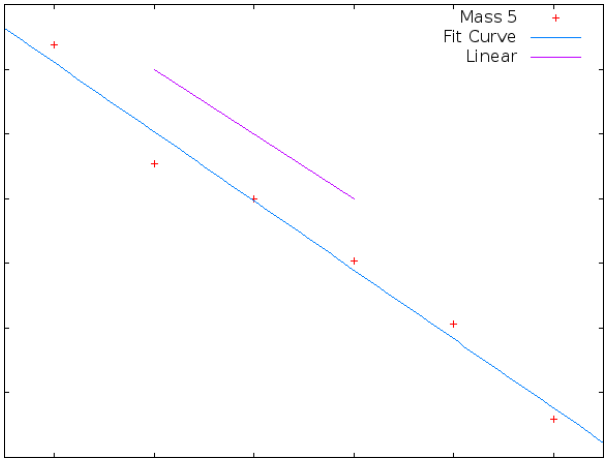
Figure 13: Conservation error of a Sod shock tube interacting with a rigid point-mass, with $M_S = 1$. Note that the scale in the dependent axis is 10^{-14}, showing that all of the errors in conservation lie in the round-off error.



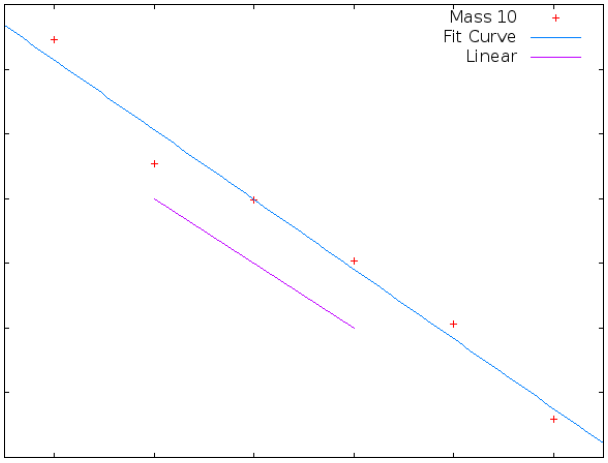
(a) $M_S = .01$ converges at 1.02.



(b) $M_S = 1$ converges at 1.01.

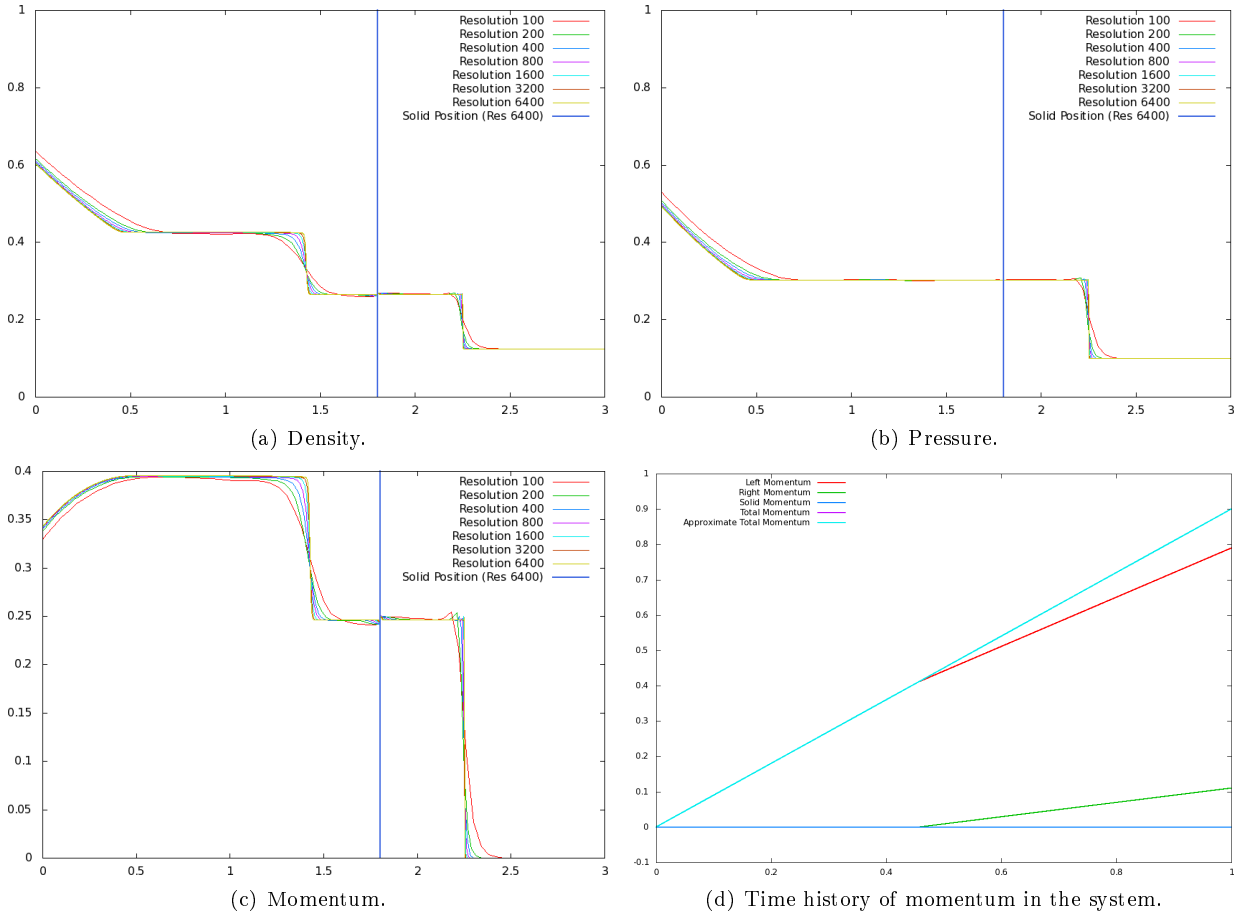


(c) $M_S = 5$ converges at 1.07.



(d) $M_S = 10$ converges at 1.08.

Figure 14: Convergence rate for the position of a rigid point-mass, where error is computed as the L^2 error in position against the position from a highly refined solution. The rate of convergence is computed as the slope of the best-fit line on the log-log scale of error as a function of grid refinement, and this best-fit line is shown in blue. This can be compared with a reference line of slope 1, shown as the purple line.



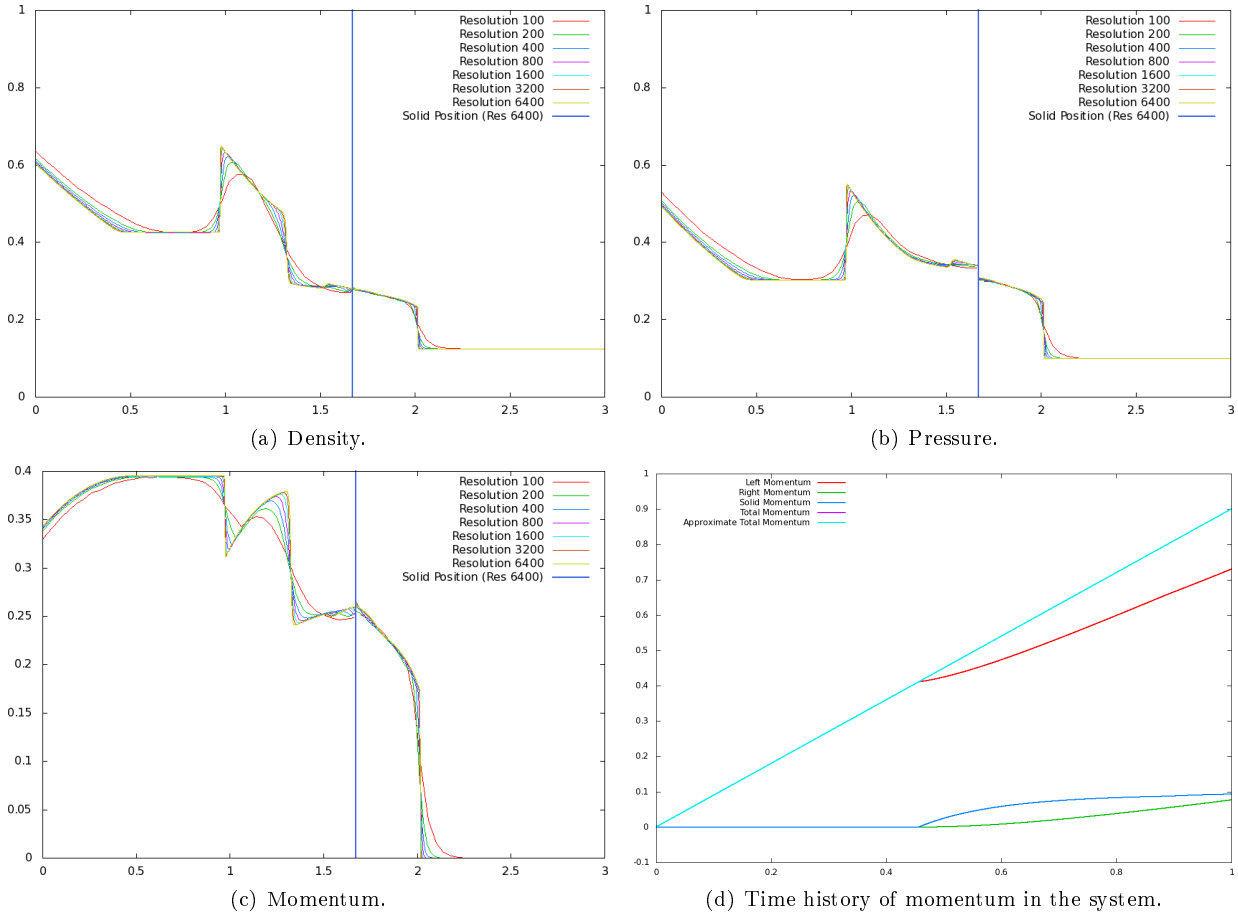
(a) Density.

(b) Pressure.

(c) Momentum.

(d) Time history of momentum in the system.

Figure 15: A Sod shock interacts with a rigid point-mass of mass 10^{-6}, where the blue vertical line represents the position of the solid; we show a snapshot of density (a), pressure (b) and momentum at $t = 1s$. The time history of momentum is depicted in (d), where the fluid momentum to the left of the solid is shown in red, the fluid momentum to the right of the solid is shown in green, and the momentum of the rigid point-mass is the dark blue line. The sum of these quantities are shown in the purple line, and the light blue line represents the expected momentum of the system based on the pressure difference at the boundary—note that these lines coincide exactly.



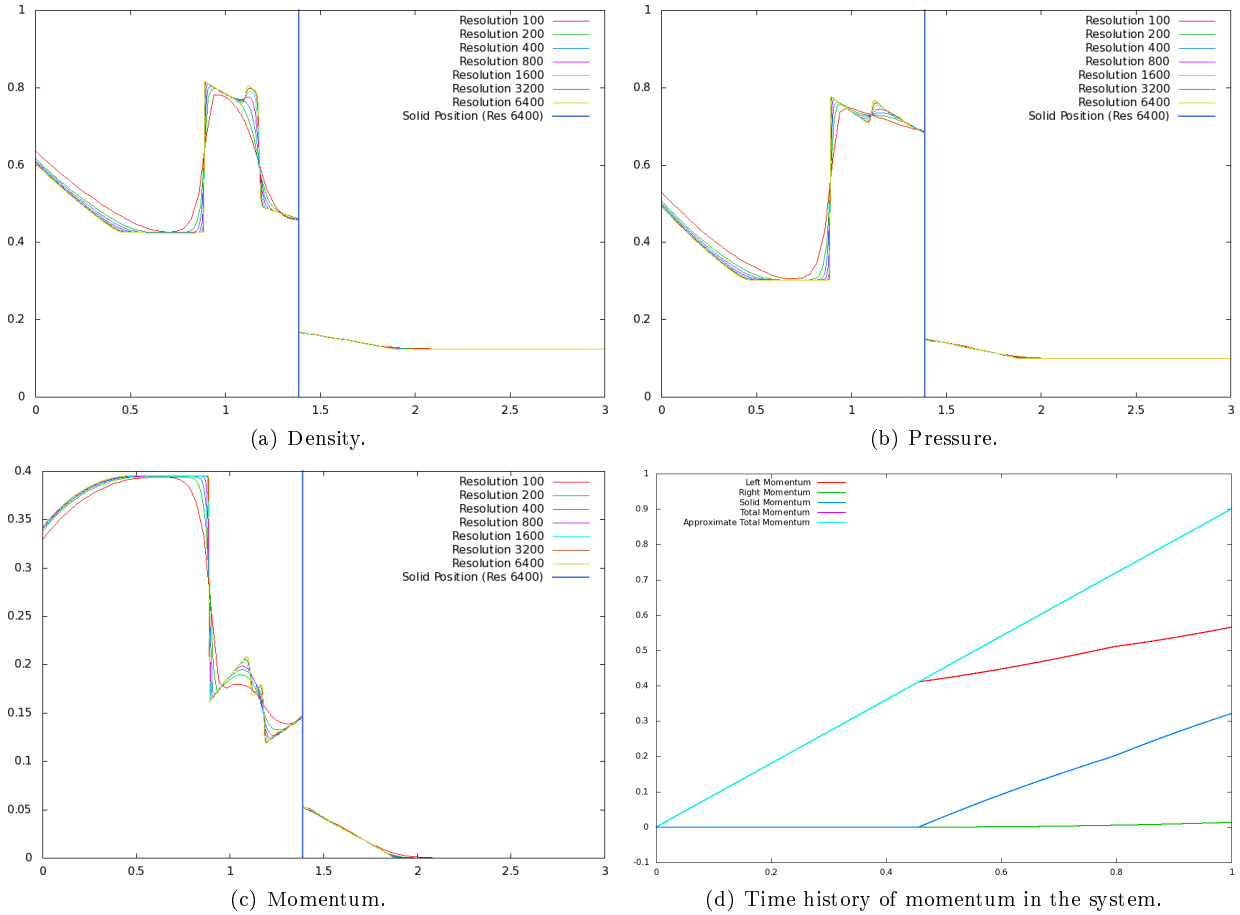
(a) Density.

(b) Pressure.

(c) Momentum.

(d) Time history of momentum in the system.

Figure 16: A Sod shock interacts with a rigid point-mass of mass 10^{-1}, where the blue vertical line represents the position of the solid; we show a snapshot of density (a), pressure (b) and momentum at $t = 1s$. The time history of momentum is depicted in (d), where the fluid momentum to the left of the solid is shown in red, the fluid momentum to the right of the solid is shown in green, and the momentum of the rigid point-mass is the dark blue line. The sum of these quantities are shown in the purple line, and the light blue line represents the expected momentum of the system based on the pressure difference at the boundary—note that these lines coincide exactly.



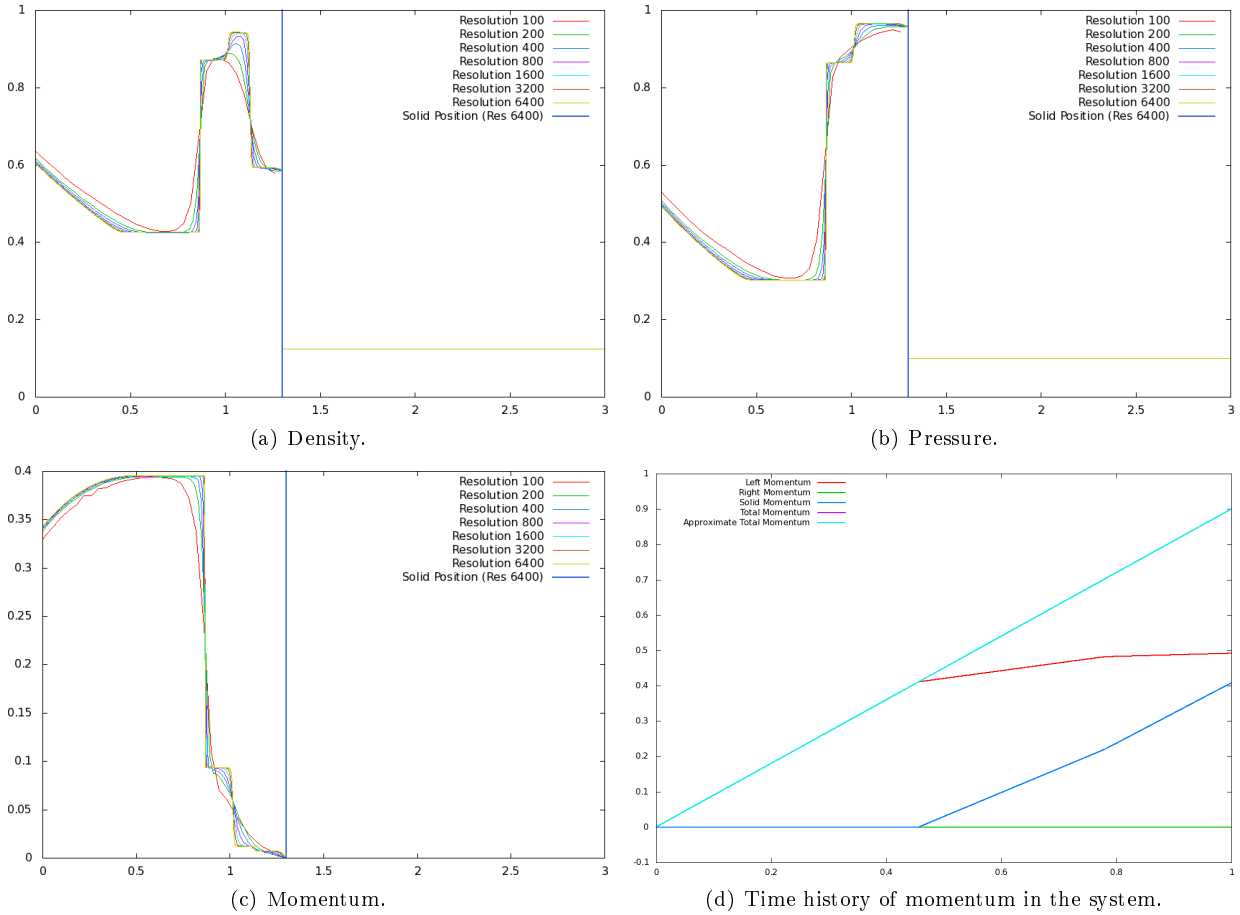
(a) Density.

(b) Pressure.

(c) Momentum.

(d) Time history of momentum in the system.

Figure 17: A Sod shock interacts with a rigid point-mass of mass 1, where the blue vertical line represents the position of the solid; we show a snapshot of density (a), pressure (b) and momentum at $t = 1s$. The time history of momentum is depicted in (d), where the fluid momentum to the left of the solid is shown in red, the fluid momentum to the right of the solid is shown in green, and the momentum of the rigid point-mass is the dark blue line. The sum of these quantities are shown in the purple line, and the light blue line represents the expected momentum of the system based on the pressure difference at the boundary—note that these lines coincide exactly.



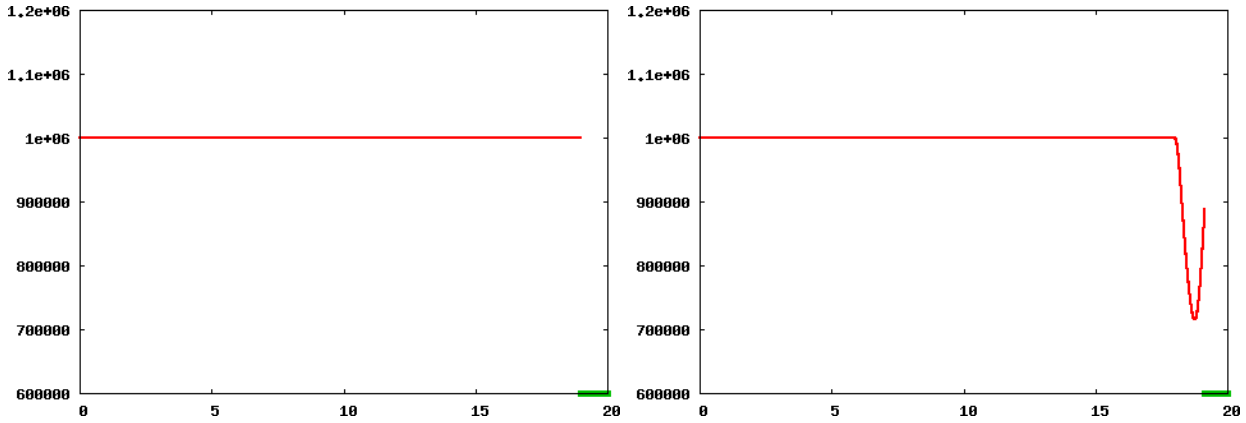
(a) Density.

(b) Pressure.

(c) Momentum.

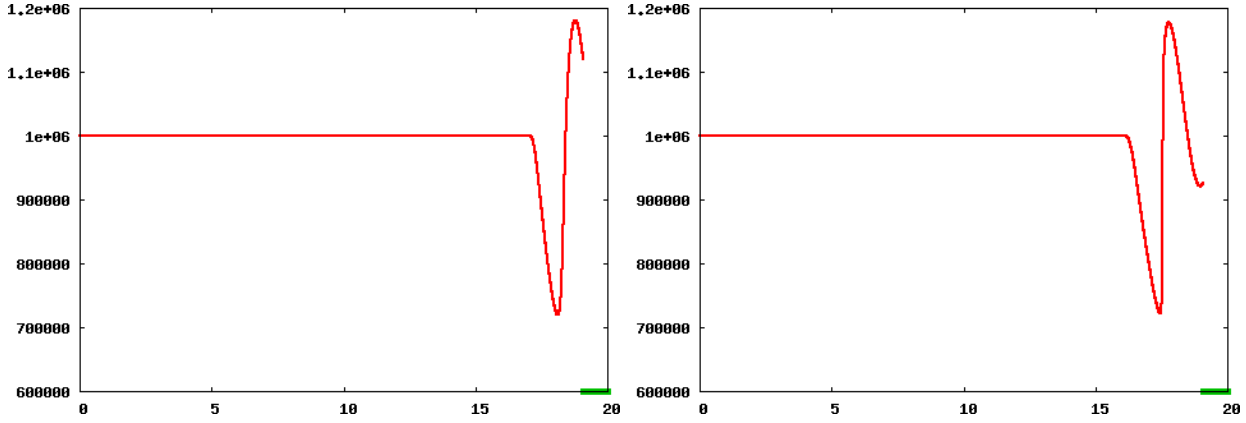
(d) Time history of momentum in the system.

Figure 18: A Sod shock interacts with a rigid point-mass of mass 10^6, where the blue vertical line represents the position of the solid; we show a snapshot of density (a), pressure (b) and momentum at $t = 1s$. The time history of momentum is depicted in (d), where the fluid momentum to the left of the solid is shown in red, the fluid momentum to the right of the solid is shown in green, and the momentum of the rigid point-mass is the dark blue line. The sum of these quantities are shown in the purple line, and the light blue line represents the expected momentum of the system based on the pressure difference at the boundary—note that these lines coincide exactly.



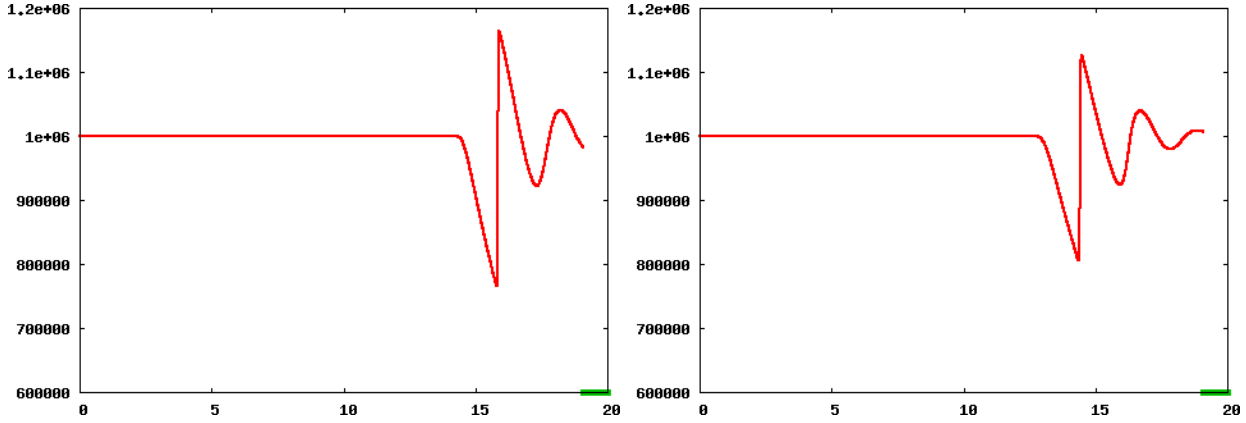
(a) $t = 0s$

(b) $t = .0015s$



(c) $t = .003s$

(d) $t = .0045s$



(e) $t = .0075s$

(f) $t = .01s$

Figure 19: Pressure of the flow field for Section 5.2, for a selection of times.

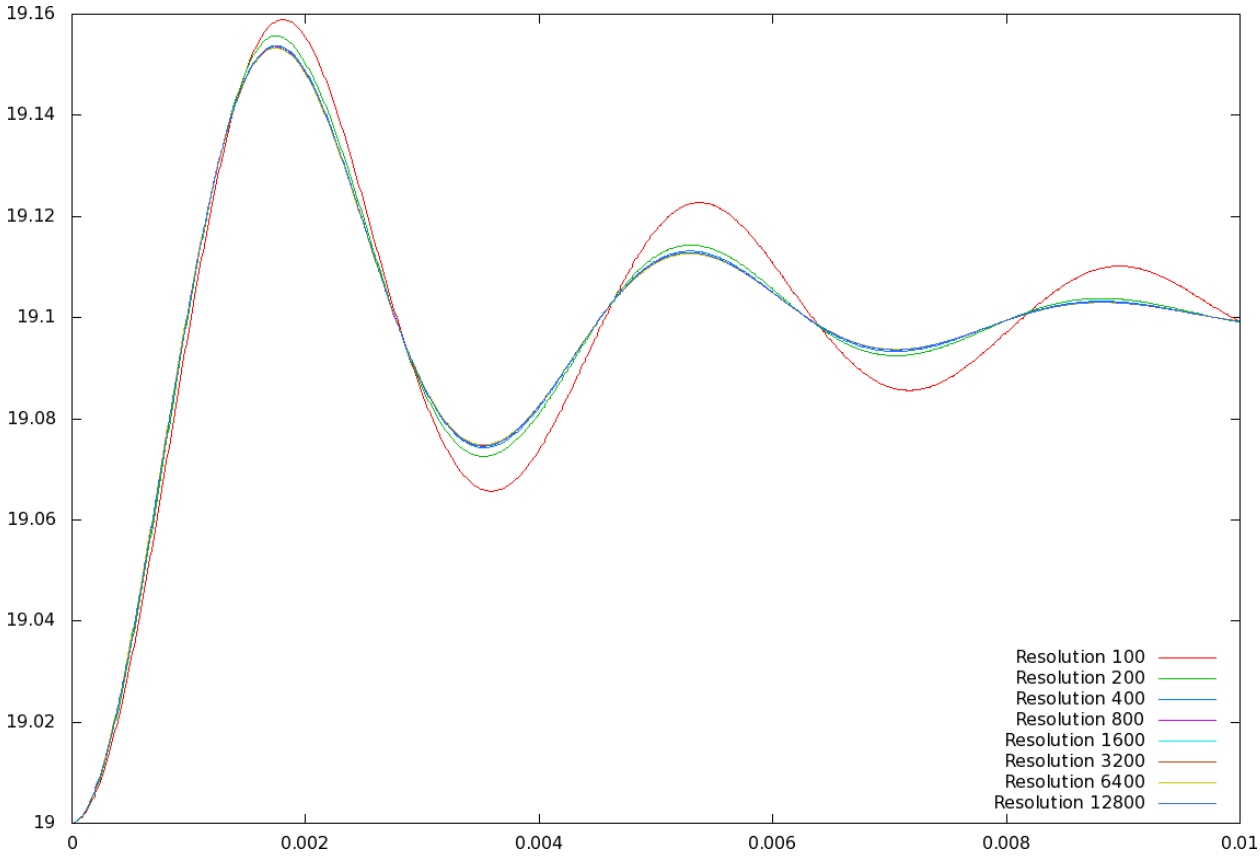
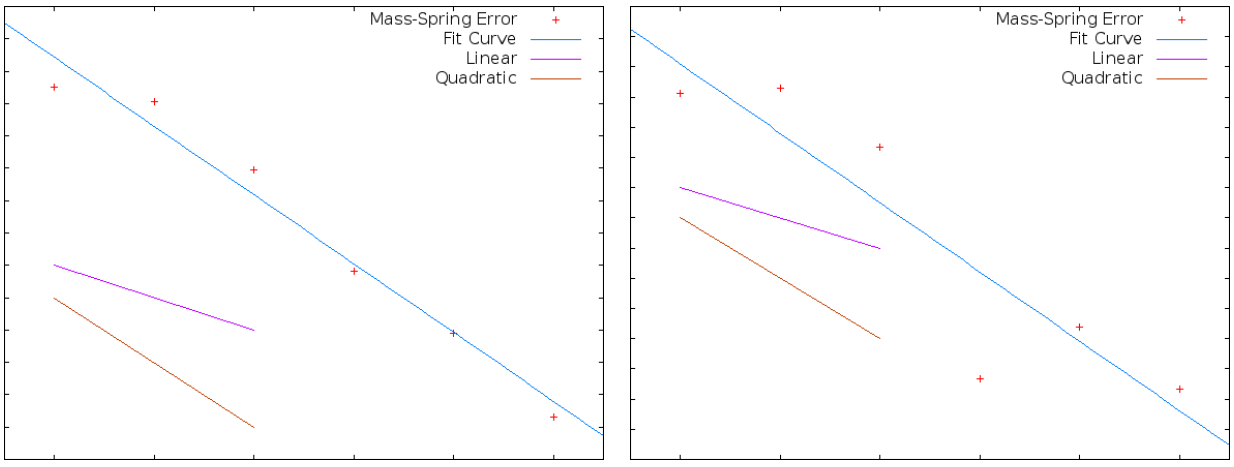


Figure 20: Position of the free end of the spring for Section 5.2, as a function of time.



(a) $t = .0075s$ with a best-fit convergence of 2.12.

(b) $t = .008s$ with a best-fit convergence of 2.29.

Figure 21: Convergence of the free end of the spring to the analytic solution, for Section 5.2. The rate of convergence is computed as the slope of the best-fit line on the log-log scale, and this best-fit is shown above in blue. This can be compared with a reference line of slope 1 (representing linear convergence), shown as the purple line, and a reference line of slope 2 (representing quadratic convergence), shown as the brown line.

5.3. Two-dimensional examples

We consider a variety of solid bodies submerged in an ideal gas, with initial conditions specified by

$$(\rho, u, v, p) = \begin{cases} (5.4, \frac{20}{9}, 0, \frac{31}{3}) & x \leq x_s \\ (1.4, 0, 0, 1) & x > x_s \end{cases}$$

where the initial location x_s of a Mach 3 rightward-moving shock varies per example. For each example involving a dynamic moving structure we compute the convergence of the center of that structure with respect to a high-resolution simulation via its L^2 error. The resulting errors are plotted as a function of grid refinement on a log-log scale in Figure 22 and the line of best fit is computed and shown as a blue line. The slope of this line gives the convergence rate and can be compared against a line of slope 1 (shown in purple), which is representative of the behavior of a linearly converging solution.

5.3.1. Leakproof validation with an infinite-mass rigid thin shell

The discontinuity is initialized at $x_s = .475$, and reflects off of an infinitesimally thin, slanted rigid body that separates two regions of the flow. The body is assigned an infinite mass (by setting $M_S^{-1} = 0$), and so the fluid to the right of the body (in Ω_R) remains quiescent. We examine the total material to the right of the body, calculating

$$\sum_{i \in \Omega_R} \hat{\phi}_i V_i.$$

At $t = 0$, the mass in Ω_R is .112, with .2 total energy and no momentum, and over the course of the simulation these terms vary by less than 10^{-18}.

Figures 23 and 24 show the time evolution of the flow field for pressure and density, respectively. When the shock initially makes contact with the rigid body the shock reflects, and as it nears the funnel point of the channel it forms a Mach stem along the top edge of the channel that travels to the left. A similar feature begins to form along the rigid body, but is quickly overtaken by the reflected shock front.

5.3.2. Leakproof validation with a constrained deforming thin shell

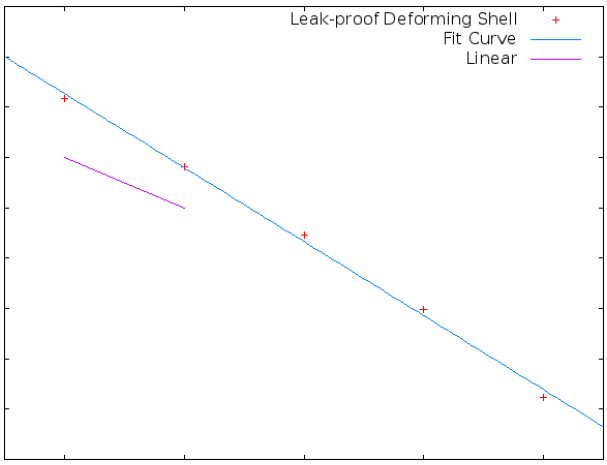
A thin deforming shell separates the channel into two regions, and the planar shock is initially at $x_s = .475$. The thin deforming shell is comprised of 21 line segments, 2 constrained nodes and 20 unconstrained point masses. Each dynamic node is assigned a mass of .0476, and the top and bottom nodes are fixed to the walls by assigning them an infinite mass (by setting $M_S^{-1} = 0$), ensuring that they do not move. The nodes are connected together by springs with a stiffness of $k = 15$ whose initial configuration dictates their rest length.

Figures 25 and 26 show the time evolution of pressure and density. The shock reflects off of the structure and causes it to buckle to the right. As the shell moves to the right a region of low mass and pressure gradually forms to its left as seen at $t = .16s$. The constraints fix the two ends to the walls of the channel, and at $t = .2s$ these constraints cease the rightward-motion of the shell. As the solid structure slows, a strong pressure and mass spike forms to its left, while a corresponding mass and pressure vacuum begin to form to its right.

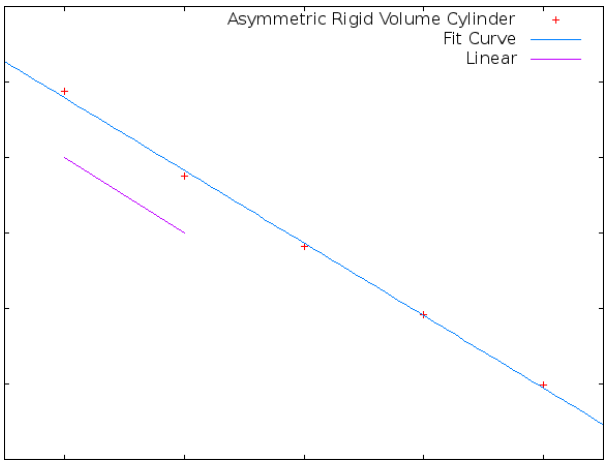
The position of the center-of-mass of the deforming body converges at a rate of 1.471, shown in Figure 22(a). The super-linear convergence is likely aided by the constrained nodes, whose influence does not change under grid refinement. We verify that the mass to the right of the deforming plate is non-changing, and the mass does not change (up to numerical round-off) from 0.13972. Unlike the previous example, however, the momentum and energy of Ω_R do change (as these quantities appropriately transfer across the solid structure).

5.3.3. Asymmetric shock reflection off a rigid cylinder

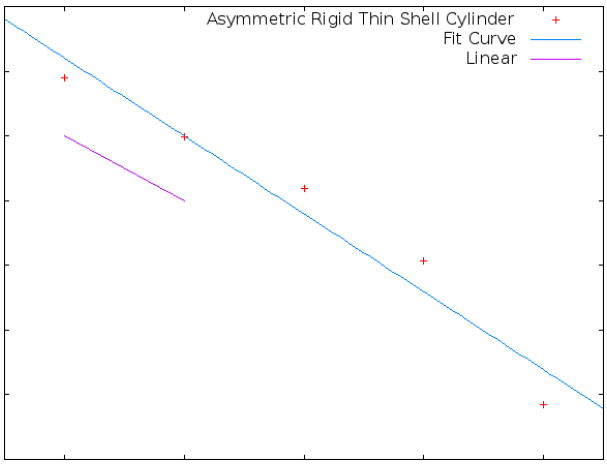
Similar to [14], we consider a rigid cylinder of radius .05, initially positioned at (.15, .06) with density 10.77. The shock is initially positioned at $x_s = .08$ and asymmetrically reflects off of the cylinder and walls. This asymmetry imparts lift on the cylinder, driving it up as it travels to the right. Under grid refinement the L^2 error of the position of its center converges at a rate of .962, shown in Figure 22(b), and the flow



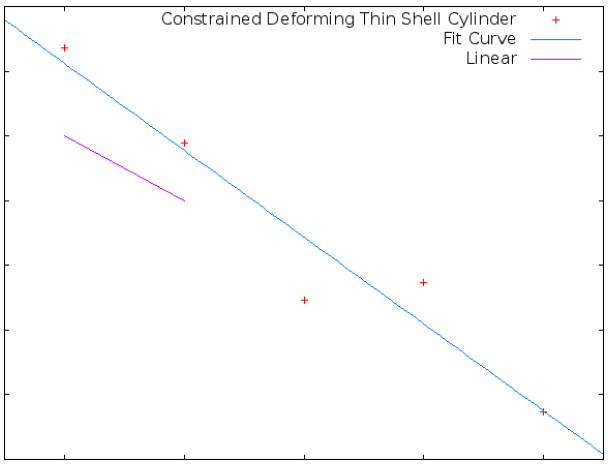
(a) Example from Section 5.3.2 converges at a rate of 1.471.



(b) Example from Section 5.3.3 converges at a rate of 0.962.

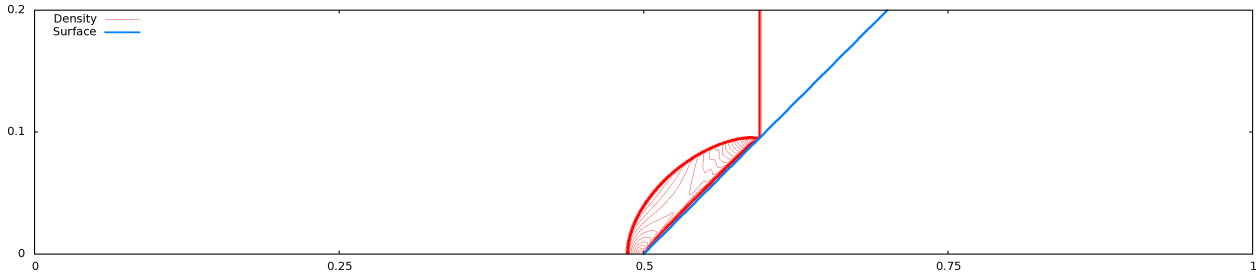


(c) Example from Section 5.3.4 converges at a rate of 1.204.

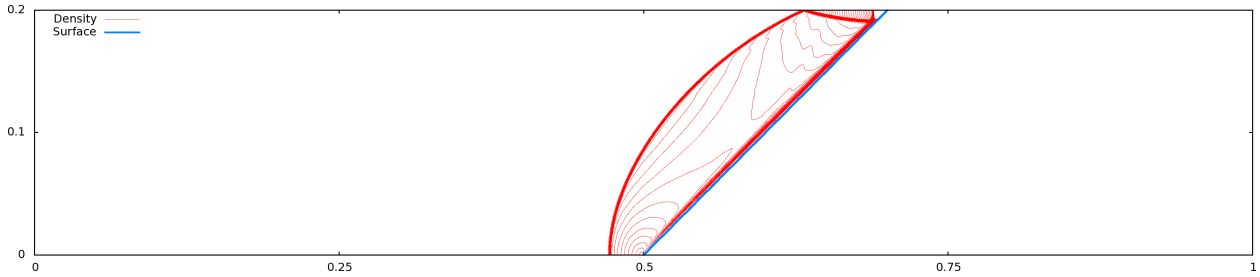


(d) Example from Section 5.3.5 converges at a rate of 1.342.

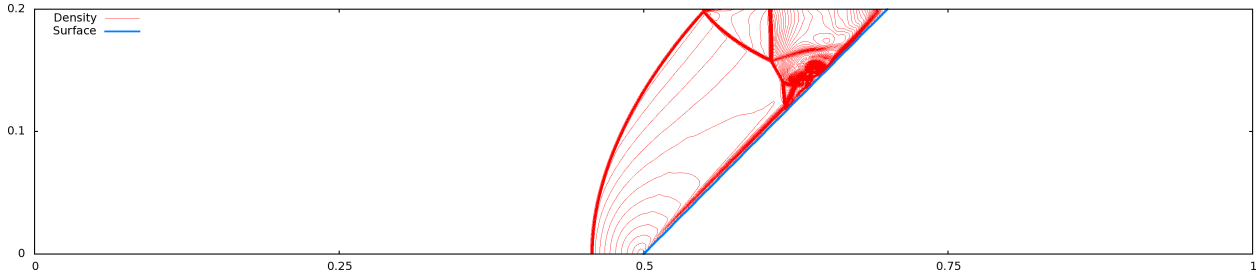
Figure 22: Convergence rate for the center of the dynamic structure is computed using the L^2 error in position against the position from a highly refined solution. The rate of convergence is computed as the slope of the best-fit line on the log-log scale of the L^2 error as a function of grid refinement, and this best-fit line is shown in blue. This can be compared with a reference line of slope 1, shown in purple.



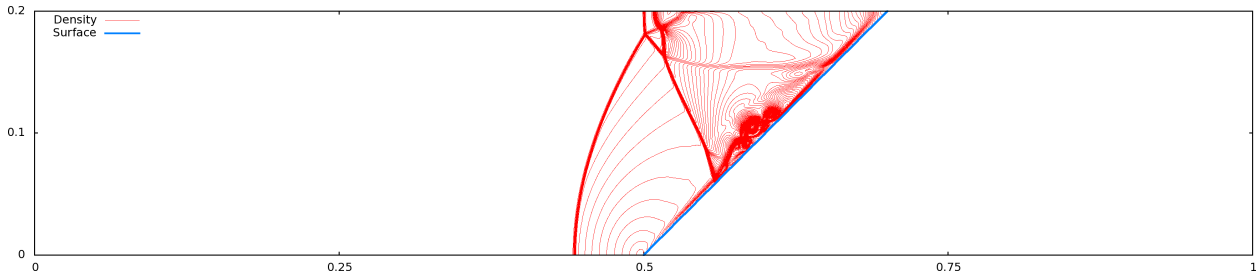
(a) $t = .04s$



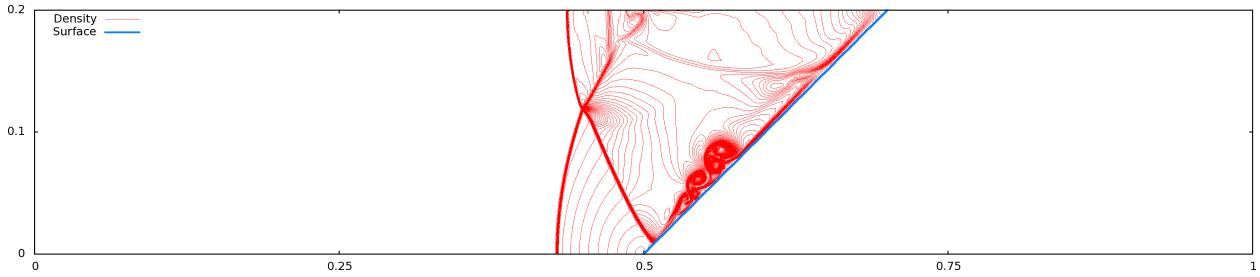
(b) $t = .08s$



(c) $t = .12s$

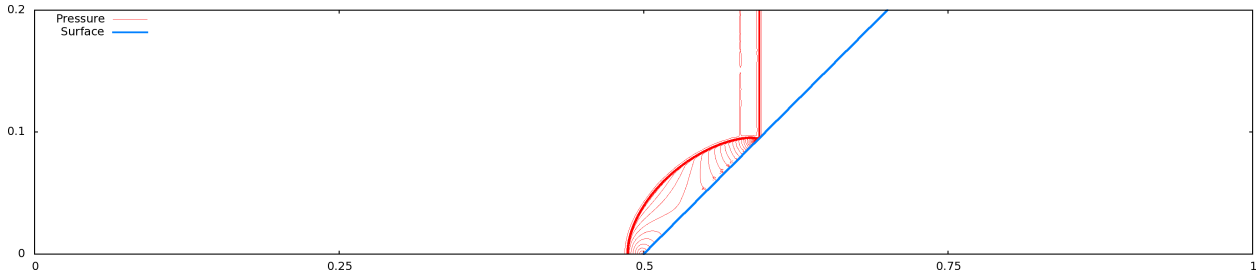


(d) $t = .16s$

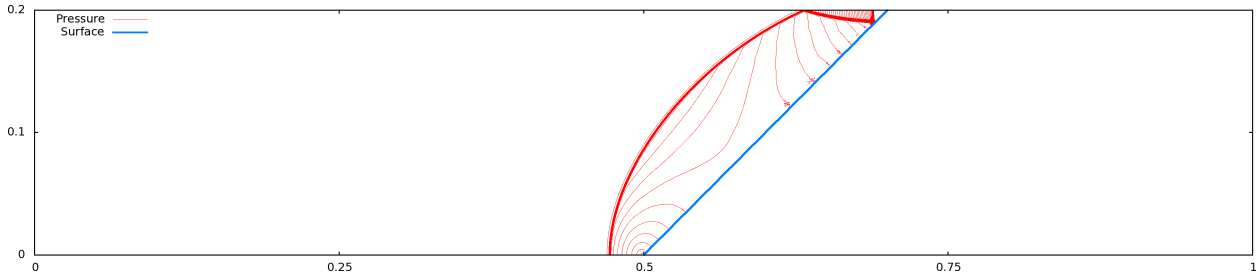


(e) $t = .2s$

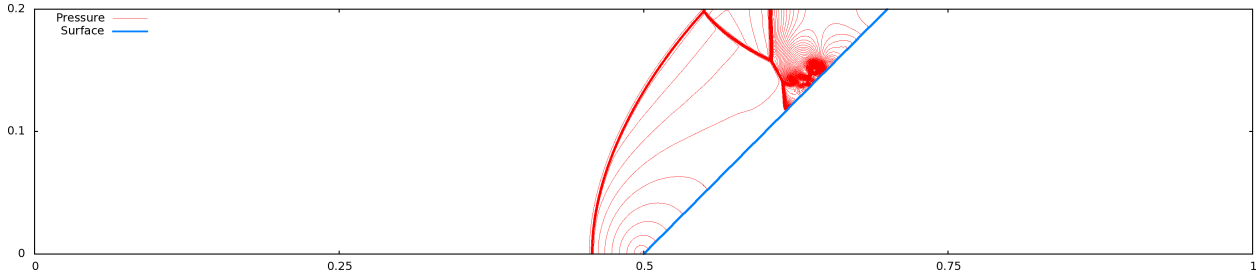
Figure 23: Time evolution of density in the example described in Section 5.3.1, on a 2560×512 grid.



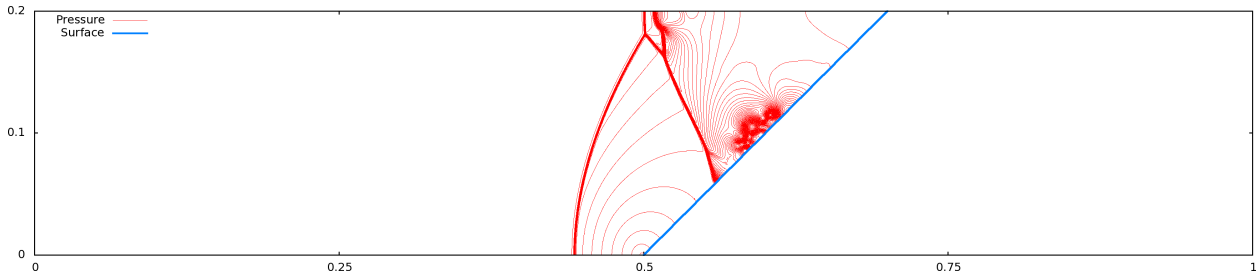
(a) $t = .04s$



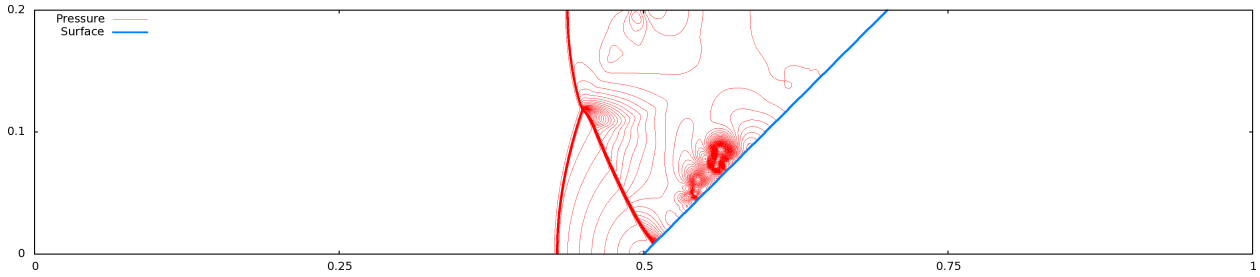
(b) $t = .08s$



(c) $t = .12s$

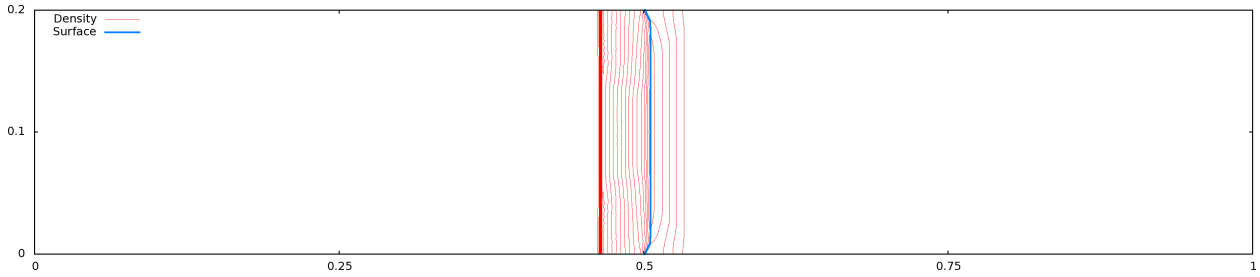


(d) $t = .16s$

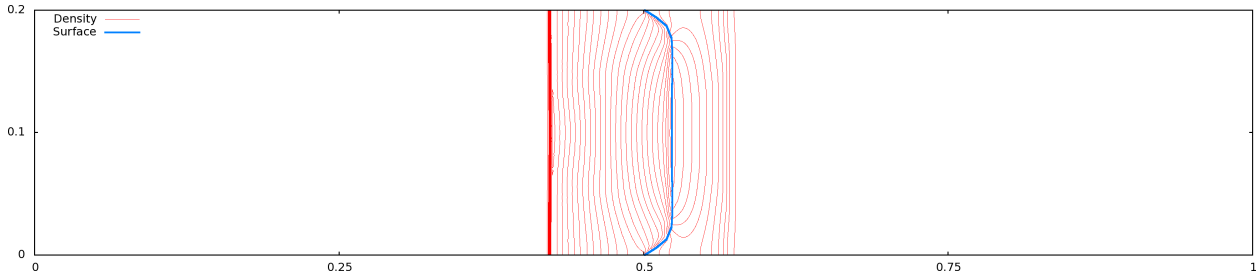


(e) $t = .2s$

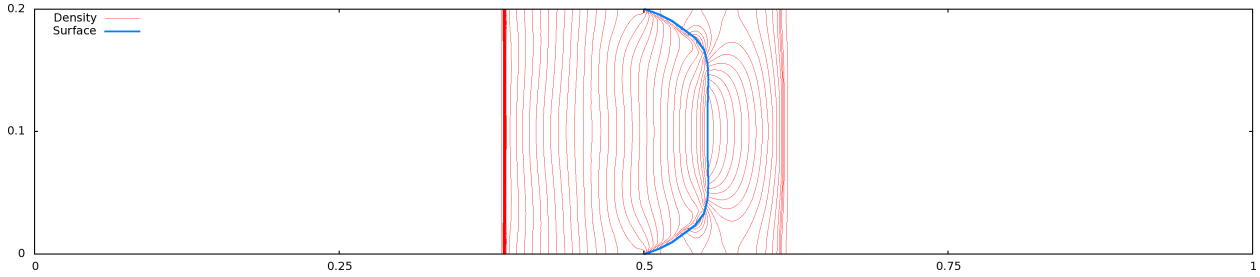
Figure 24: Time evolution of pressure in the example described in Section 5.3.1, on a 2560×512 grid.



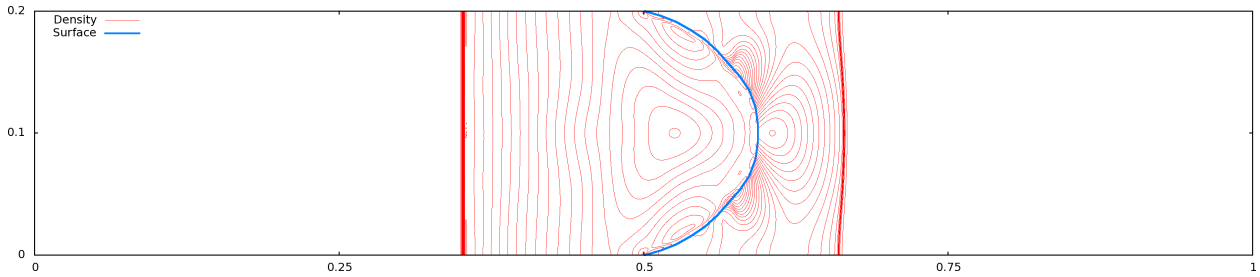
(a) $t = .04s$



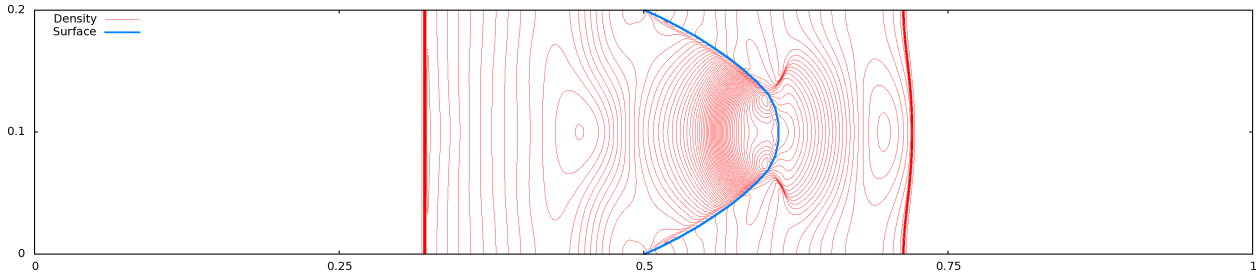
(b) $t = .08s$



(c) $t = .12s$

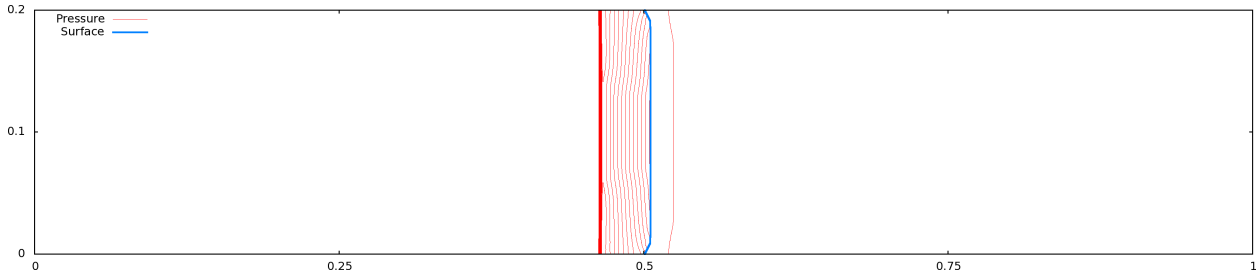


(d) $t = .16s$

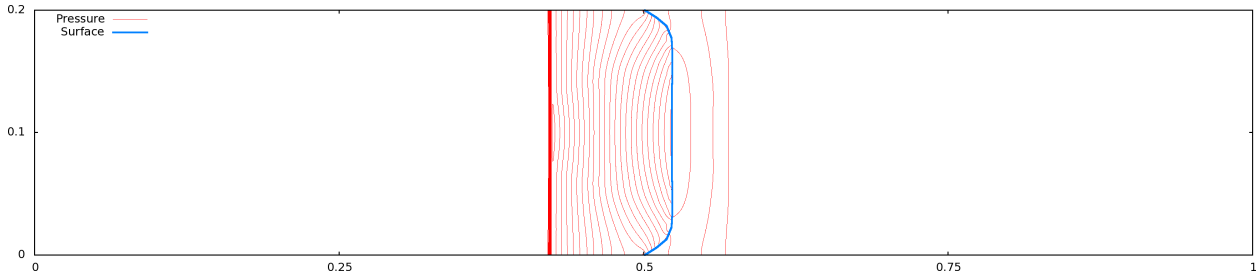


(e) $t = .2s$

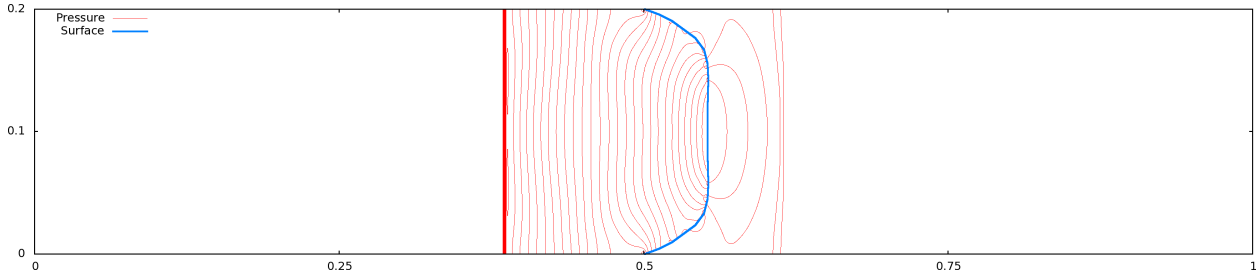
Figure 25: Time evolution of density in the example described in Section 5.3.2, on a 2560×512 grid.



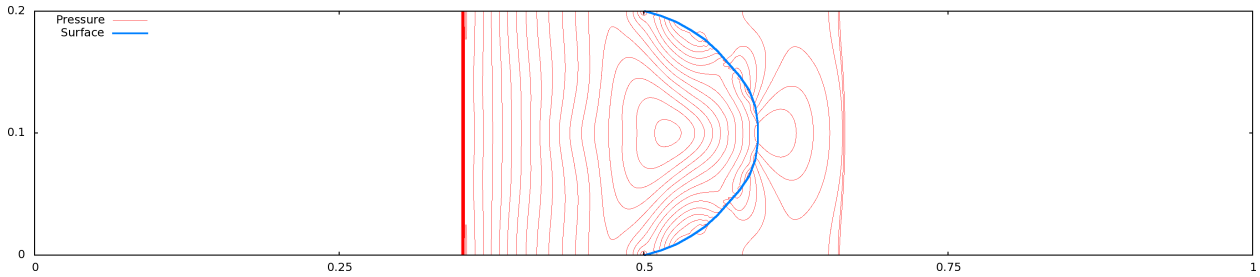
(a) $t = .04s$



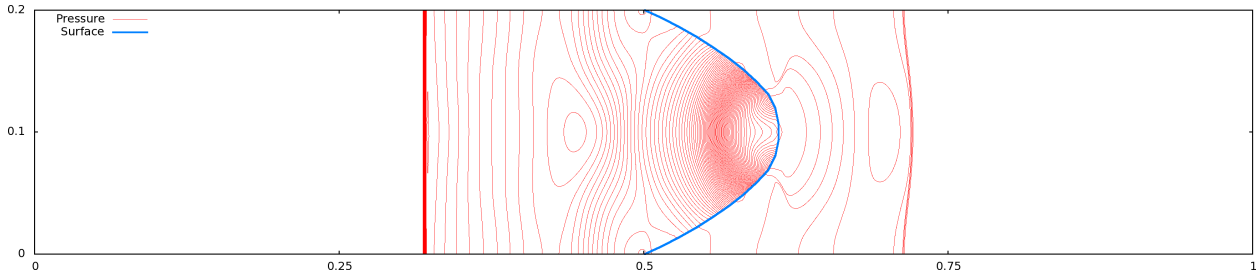
(b) $t = .08s$



(c) $t = .12s$



(d) $t = .16s$



(e) $t = .2s$

Figure 26: Time evolution of pressure in the example described in Section 5.3.2, on a 2560×512 grid.

field converges as shown in Figure 27. The time evolution is shown in Figures 28 and 29, for pressure and density respectively.

5.3.4. Asymmetric shock reflection off a thin rigid shell

The cylinder from Section 5.3.3 is hollowed out and filled with a dense fluid specified by

$$(\rho, u, v, p) = \left(\frac{31}{3}, 0, 0, \frac{31}{3}\right)$$

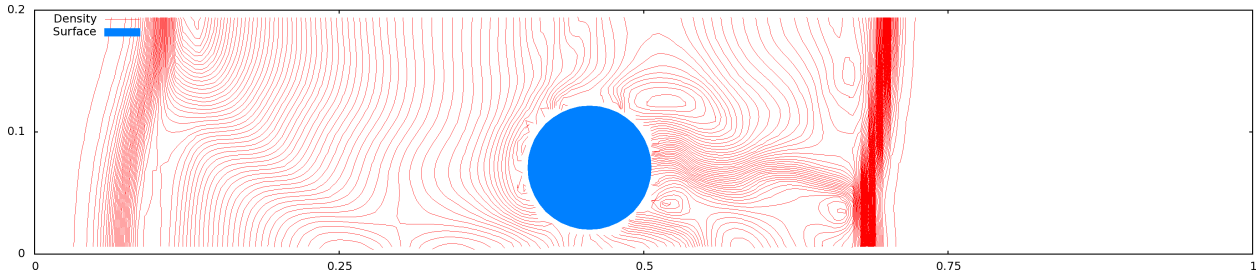
whose density and pressure are significantly higher than that of the pre-shock state. The asymmetric reflection still imparts lift on the cylinder, but the motion is delayed until the shock *inside* the solid hits the far wall of the cylinder. The internal fluid motion causes the cylinder first to delay any rightward motion, then (when the internal shock hits the right boundary at $t = 1s$) jump to the right and up. The time evolution of pressure and density are shown in Figures 30 and 31, respectively. Under grid resolution, the position of the center of the cylinder converges at a rate of 1.204, shown in Figure 22(c), and the flow field converges as shown in Figure 32. The total fluid state inside the hollow cylinder is shown in Figure 33; note that the total mass of fluid inside the cylinder does not change. The momentum and energy are more interesting, exchanging information with the structure.

5.3.5. Planar shock interacting with a constrained, immersed deforming thin shell

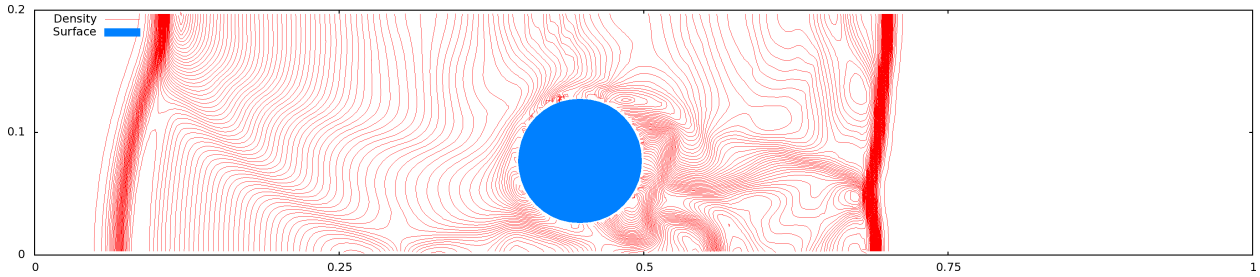
In this example, a thin deforming shell is placed at $x = .5$ of length .5, and the planar shock is initially located at $x_s = .475$. The thin deforming shell is comprised of 21 line segments, 2 constrained nodes and 20 unconstrained point masses. Each dynamic node is assigned a mass of .0952, and the top and bottom nodes are assigned an infinite mass (by setting $M_S^{-1} = 0$), ensuring that they do not move. The nodes are connected together by springs with a stiffness of $k = 10$ and whose initial configuration dictates their rest length.

The shock reflects off of the deforming body and causes the deforming shell to buckle to the right, while rarefaction fans form around the constrained nodes as flow passes by. As the deforming body reaches its right-most location and bends back a second, weaker shock pushes out and to the left and forming a vacuum behind the thin shell. Interesting vorticial structures start to form behind the thin shell, most easily seen near $x = .75$ at $t = .35s$ in Figure 34 and Figure 35. Under grid refinement (shown in Figure 36), note the regime change from laminar flow to separating flow. This is a result of artificial viscosity vanishing under grid refinement and the lack of any real viscosity by virtue of our simulating an inviscid flow. As the grid becomes more refined we can expect the observed Reynolds number to go even further up, and more refined detail should appear.

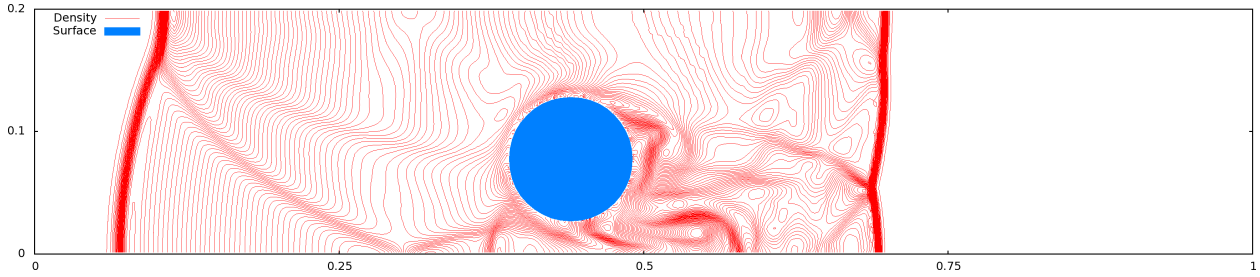
The position of the center-of-mass of the deforming body converges at a rate of 1.342, shown in Figure 22(d), and the flow field converges as shown in Figure 36. This is likely aided by the constrained nodes, whose influence does not change under refinement. No material flows through the thin solid structure.



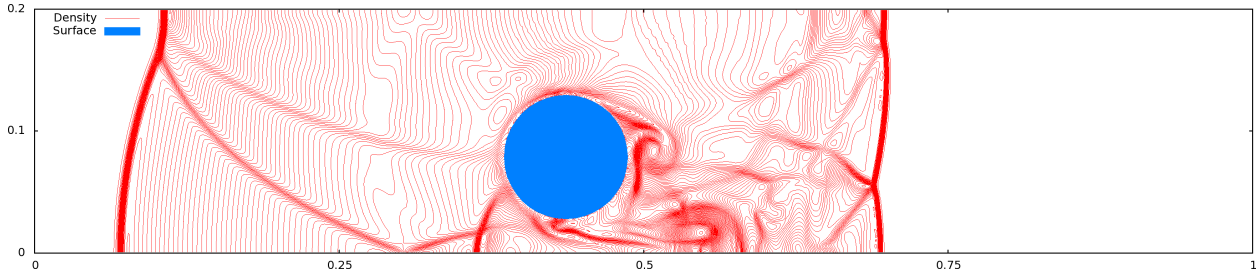
(a) Resolution 160×32



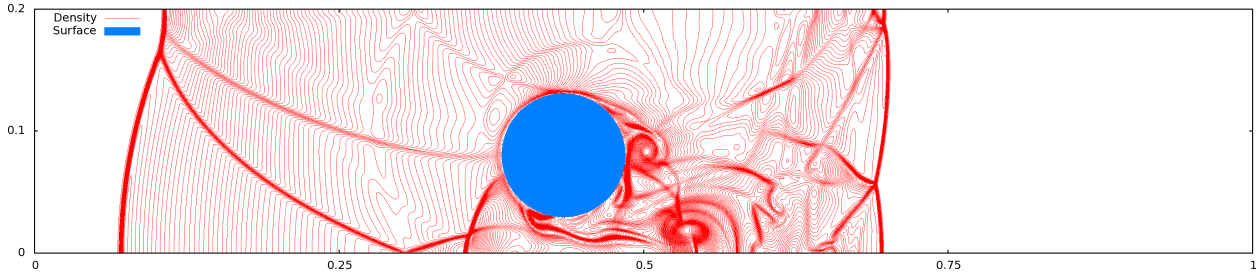
(b) Resolution 320×64



(c) Resolution 640×128

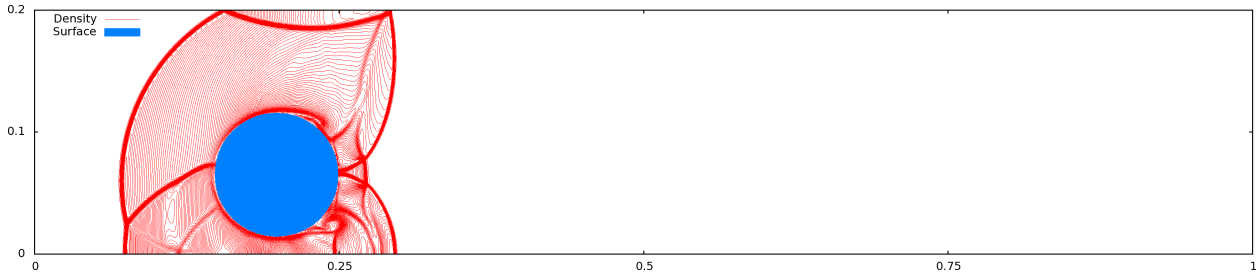


(d) Resolution 1280×256

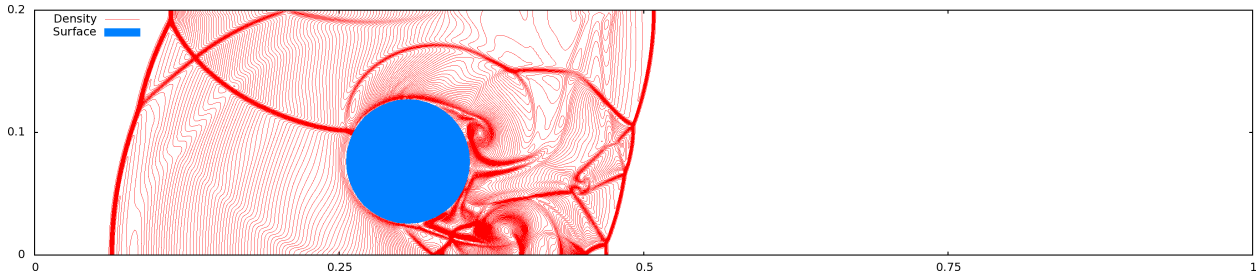


(e) Resolution 2560×512

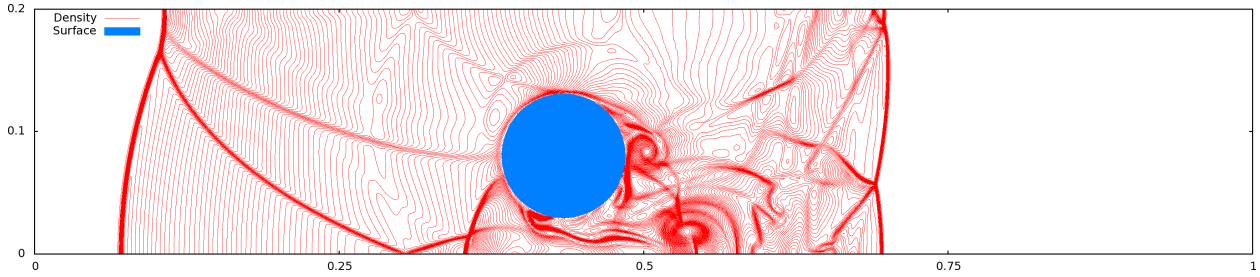
Figure 27: Grid convergence of the example described in Section 5.3.3, at time $t = .21s$.



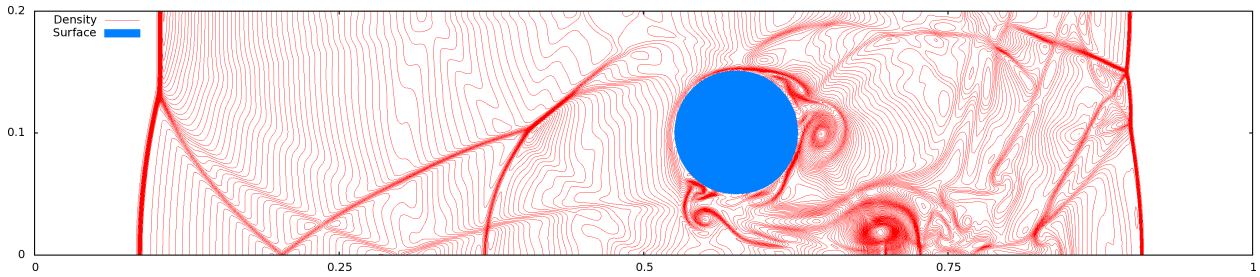
(a) $t = .07s$



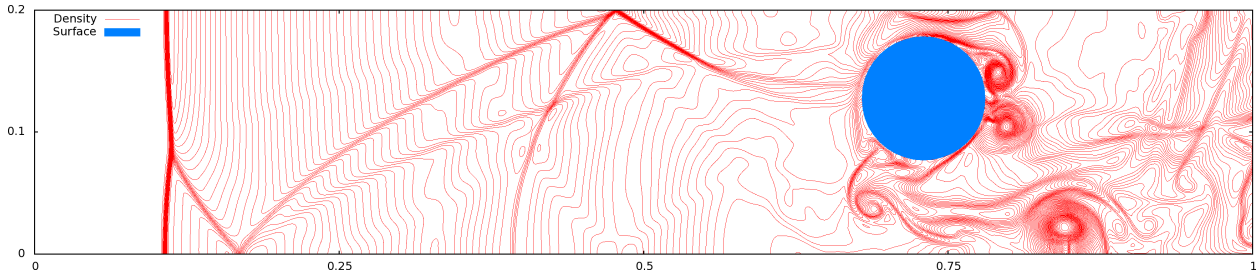
(b) $t = .14s$



(c) $t = .21s$

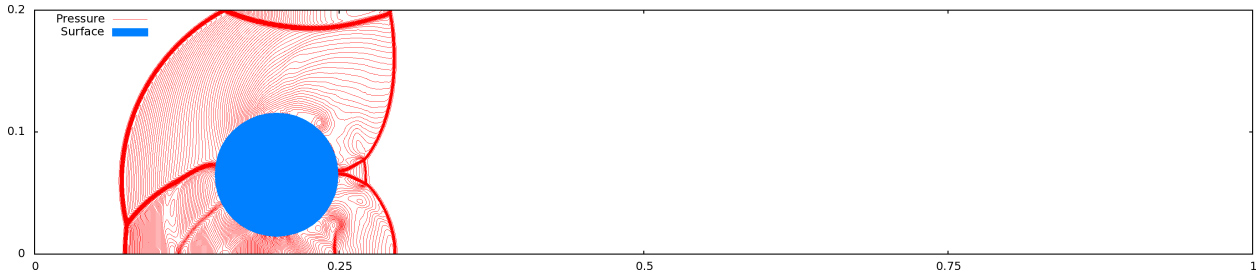


(d) $t = .28s$

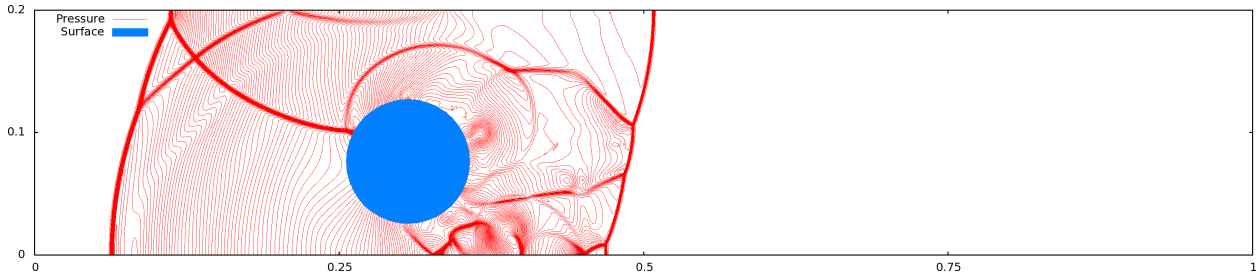


(e) $t = 35s$

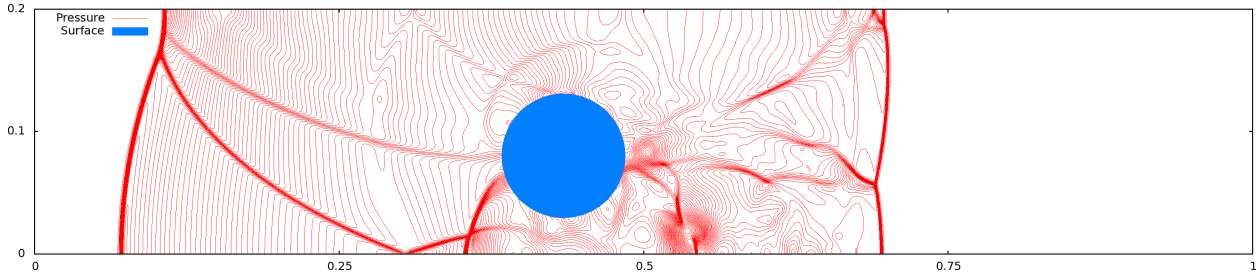
Figure 28: Time evolution of density in the example described in Section 5.3.3, on a 2560×512 grid.



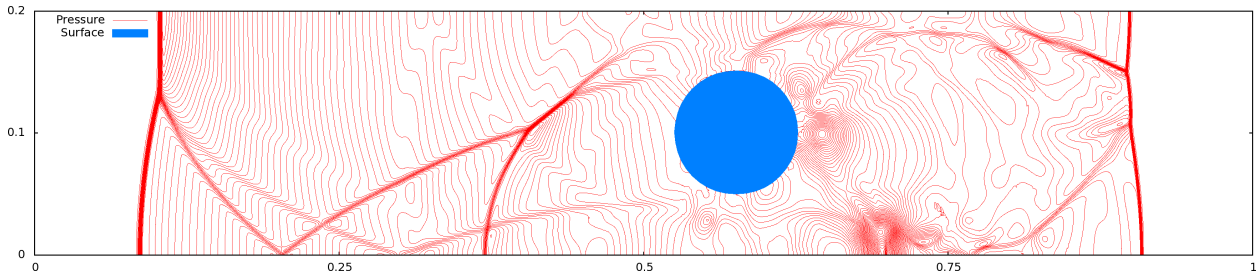
(a) $t = .07s$



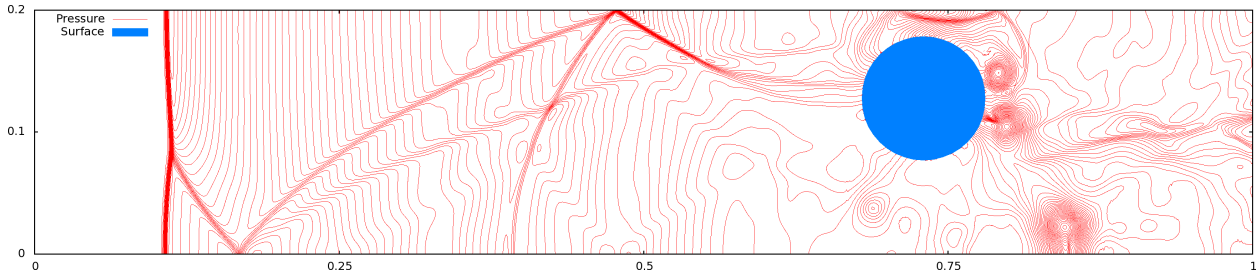
(b) $t = .14s$



(c) $t = .21s$

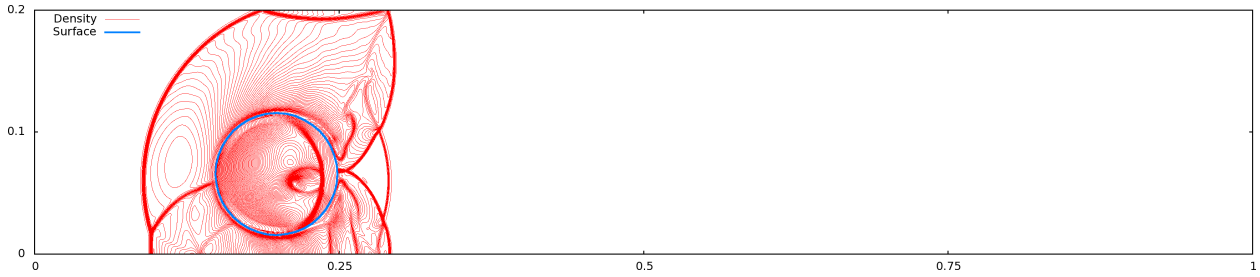


(d) $t = .28s$

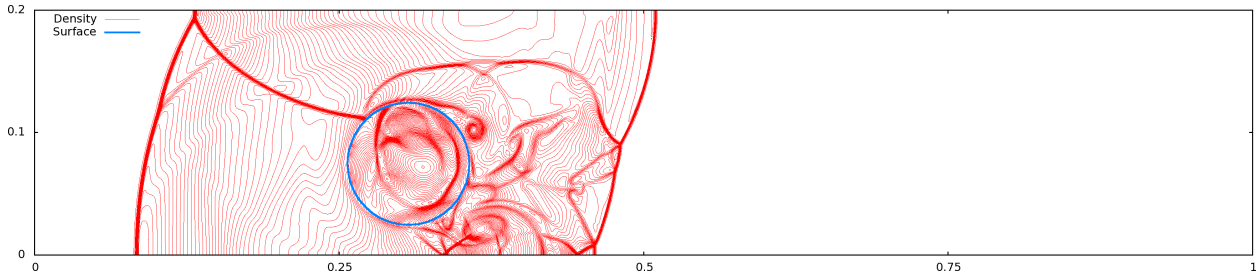


(e) $t = 35s$

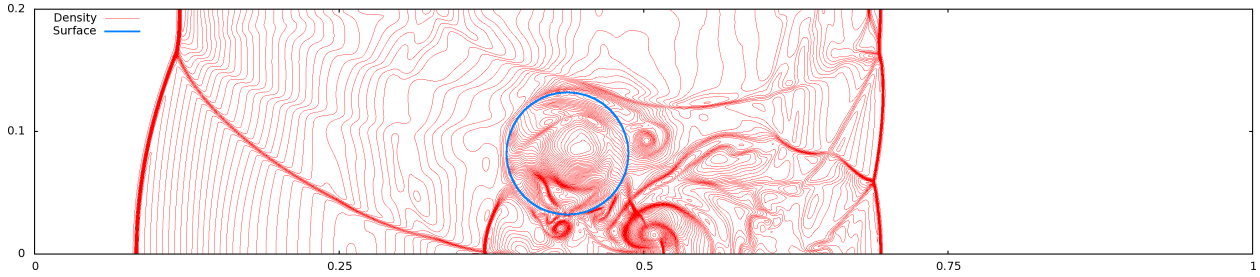
Figure 29: Time evolution of pressure in the example described in Section 5.3.3, on a 2560×512 grid.



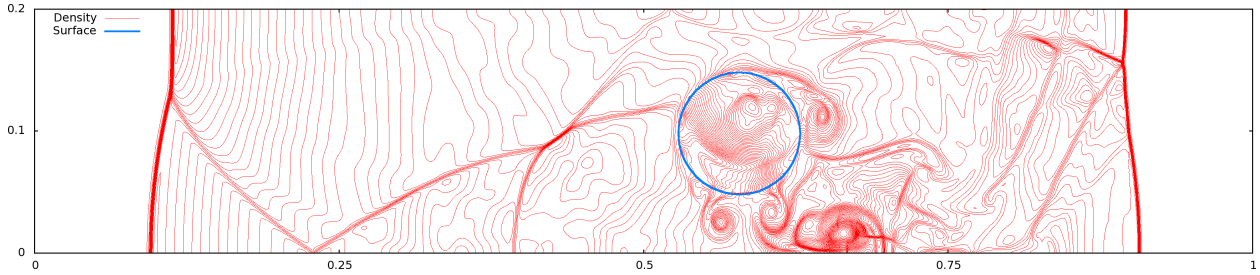
(a) $t = .07s$



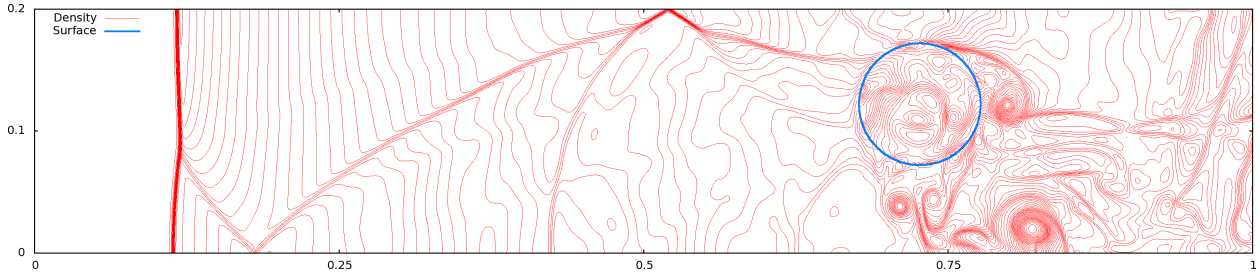
(b) $t = .14s$



(c) $t = .21s$

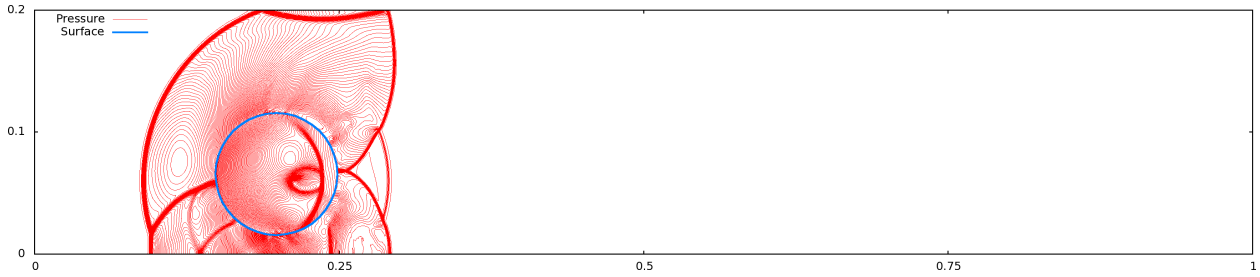


(d) $t = .28s$

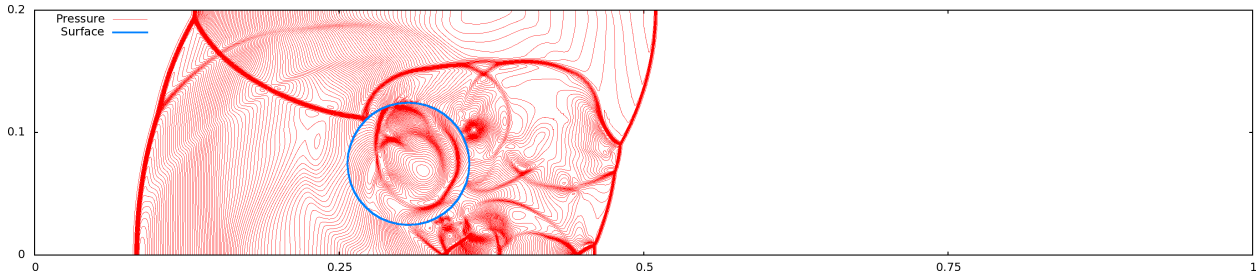


(e) $t = 35s$

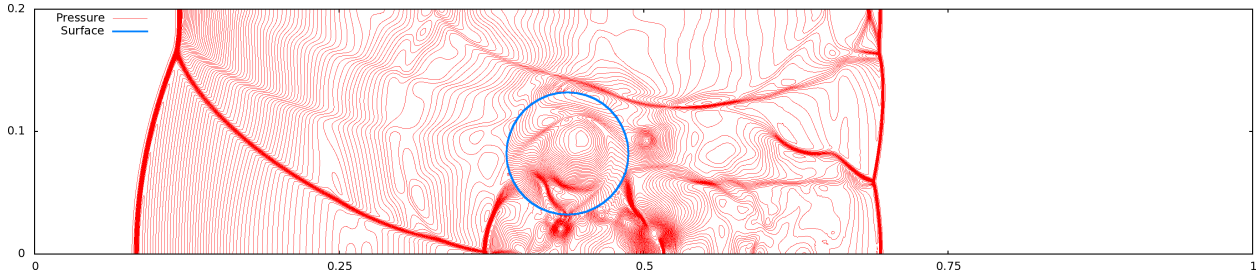
Figure 30: Time evolution of density in the example described in Section 5.3.4, on a 2560×512 grid.



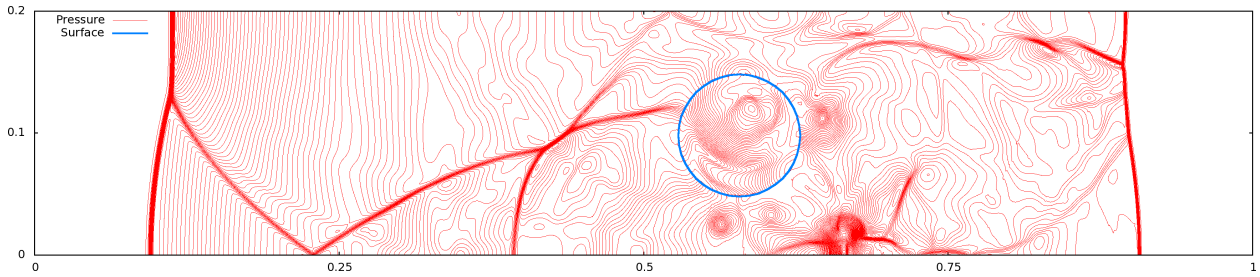
(a) $t = .07s$



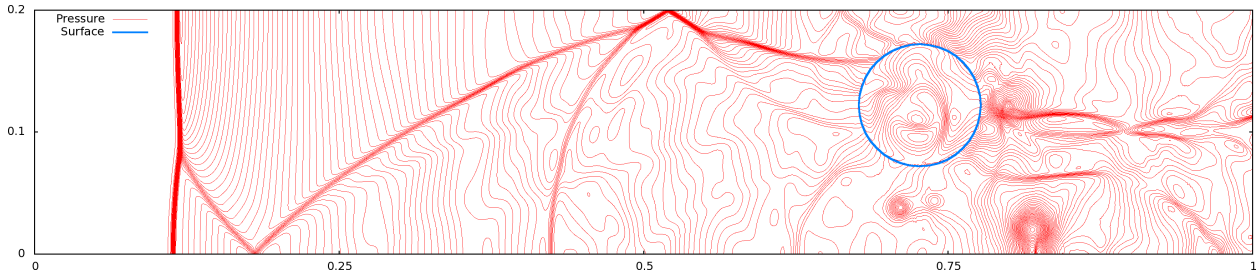
(b) $t = .14s$



(c) $t = .21s$

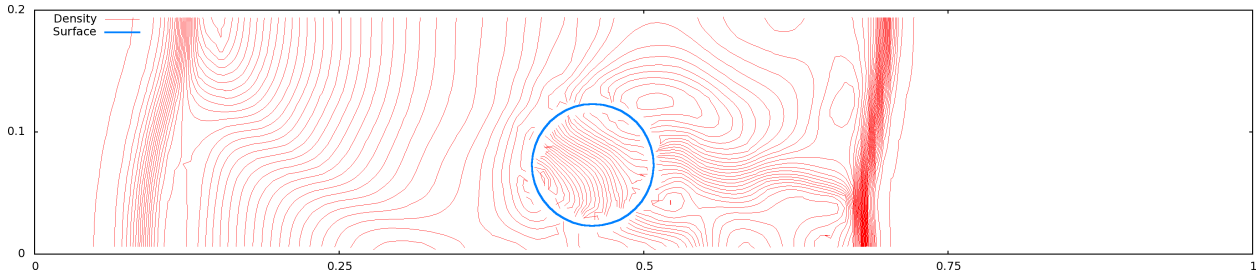


(d) $t = .28s$

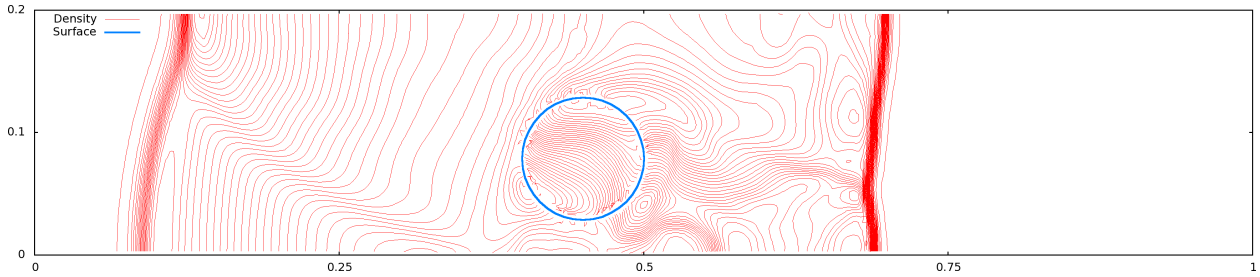


(e) $t = 35s$

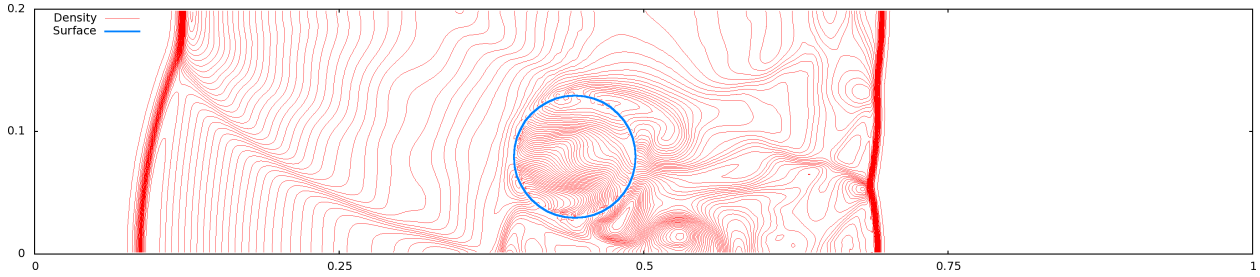
Figure 31: Time evolution of pressure in the example described in Section 5.3.4, on a 2560×512 grid.



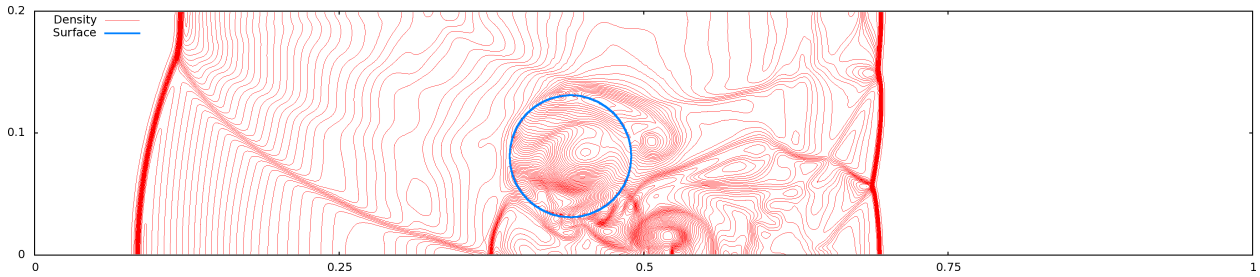
(a) Resolution 160×32



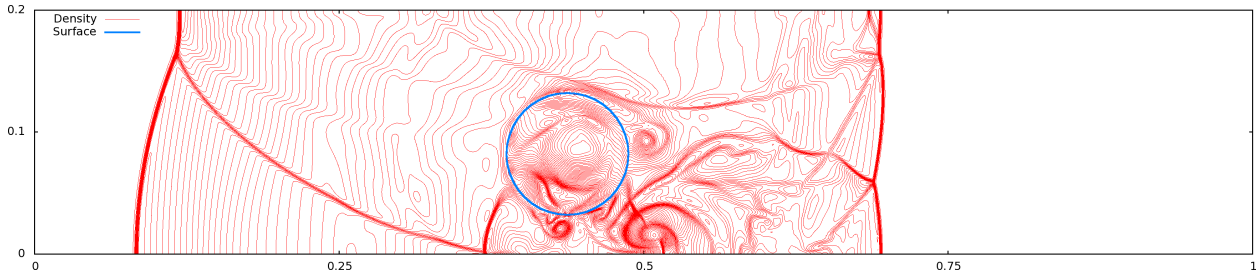
(b) Resolution 320×64



(c) Resolution 640×128

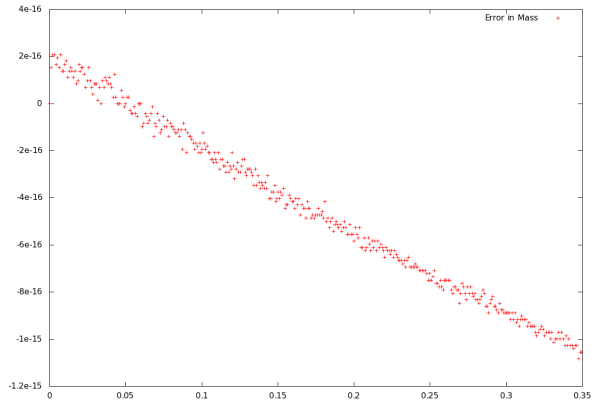


(d) Resolution 1280×256

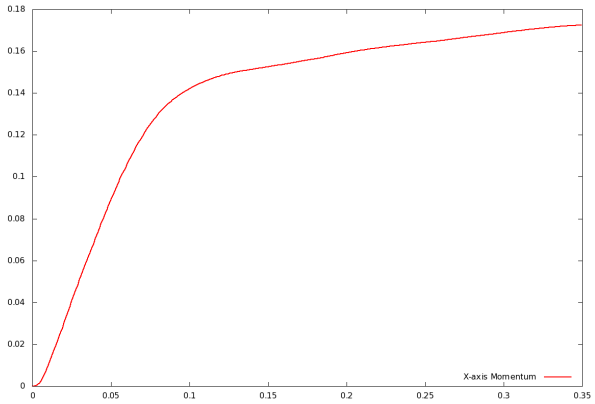


(e) Resolution 2560×512

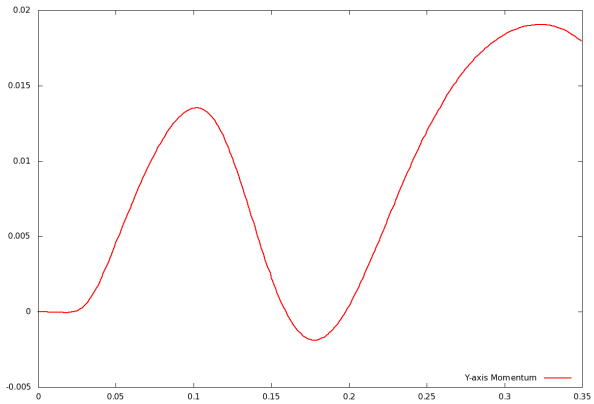
Figure 32: Grid convergence of the example described in Section 5.3.4, at time $t = .21s$.



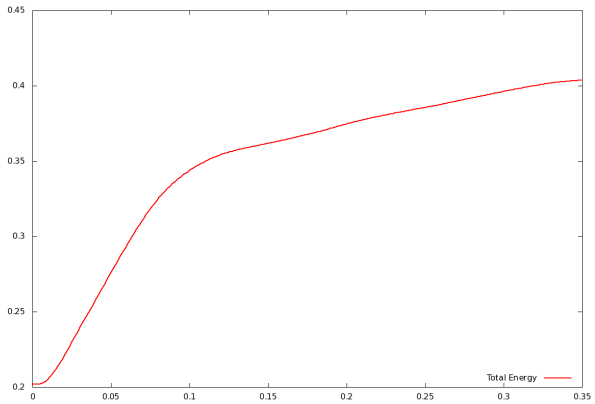
(a) Time evolution of mass error from the initial configuration.



(b) Time evolution of x-axis fluid momentum.

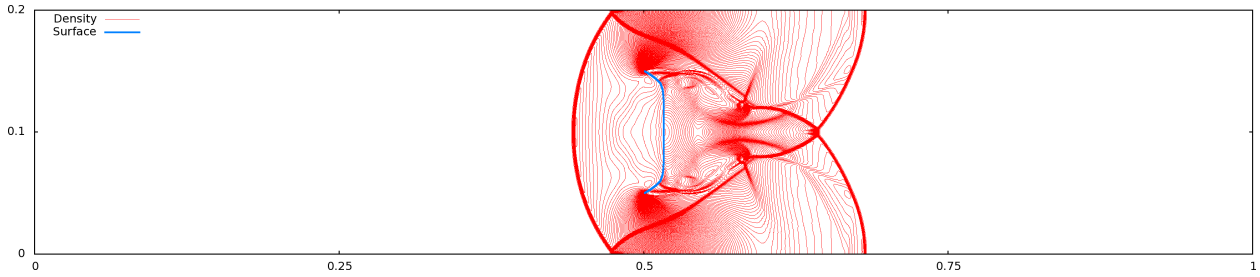


(c) Time evolution of y-axis fluid momentum.

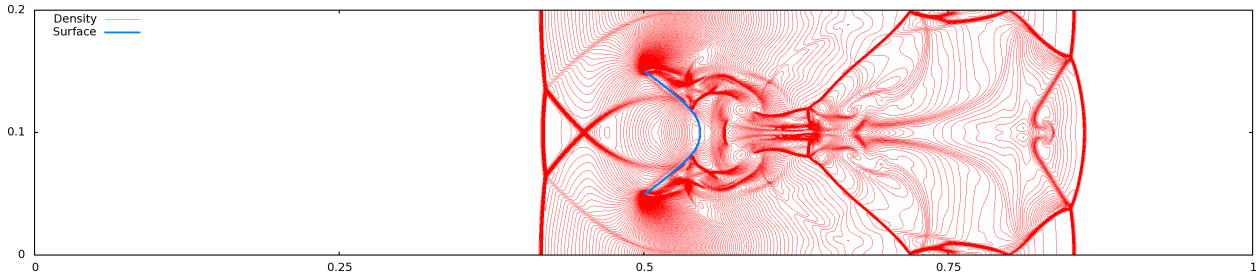


(d) Time evolution of total fluid energy.

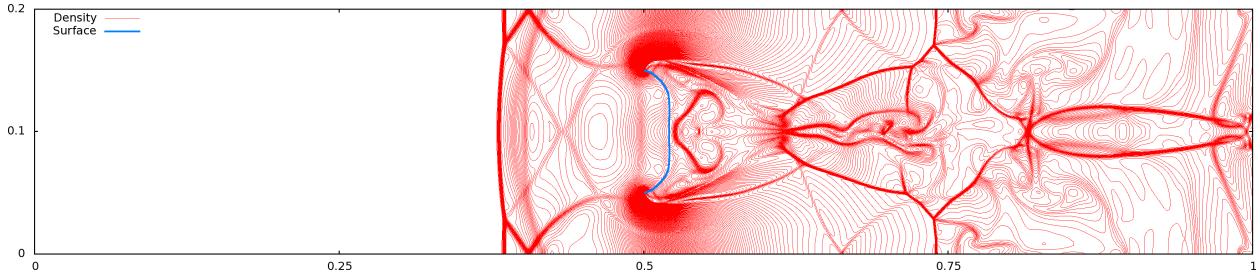
Figure 33: Time evolution of conserved material inside the hollow rigid cylinder from Section 5.3.4. In (a), we show the time history of the error in total mass inside the cylinder, while (b), (c) and (d) show time history of fluid momentum and energy inside the cylinder. Note that the scale in the dependent axis of (a) is 10^{-16}, showing that the errors in conservation lie well within round-off error.



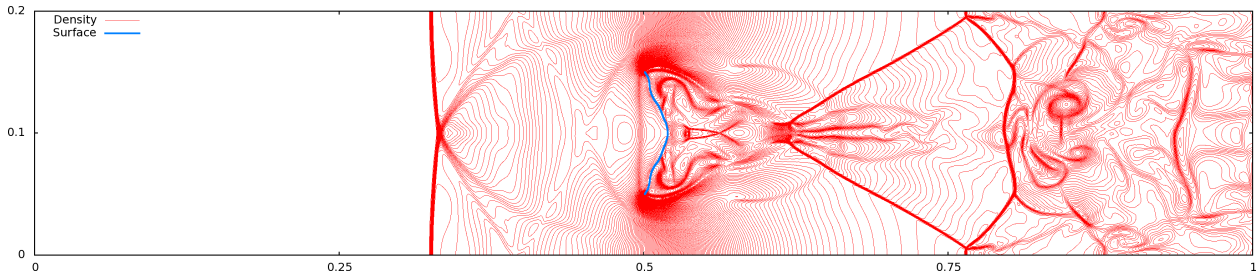
(a) $t = .07s$



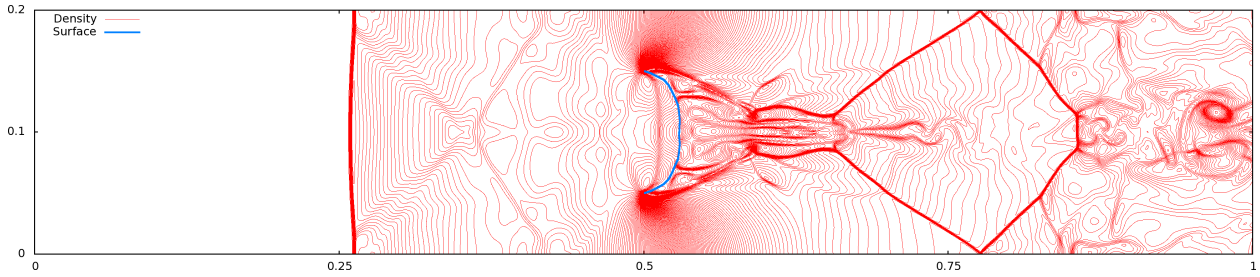
(b) $t = .14s$



(c) $t = .21s$

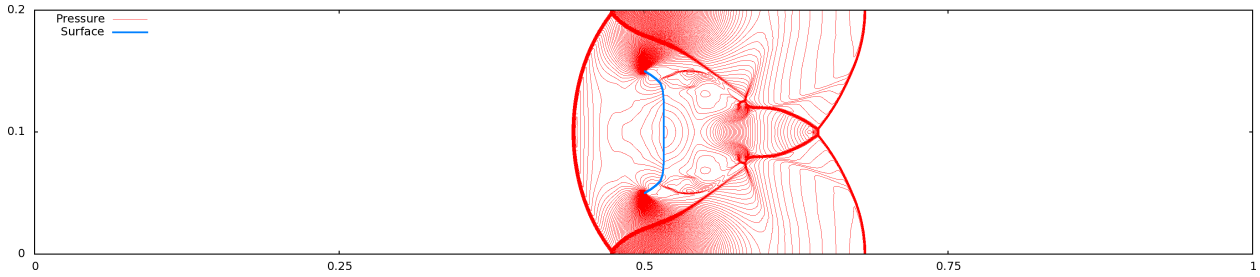


(d) $t = .28s$

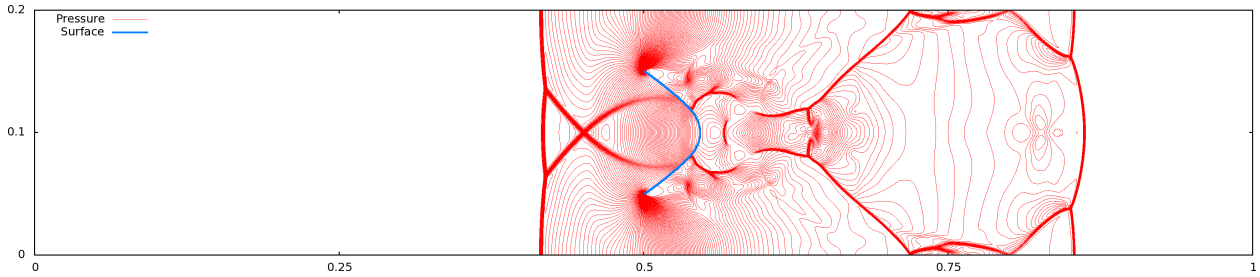


(e) $t = 35s$

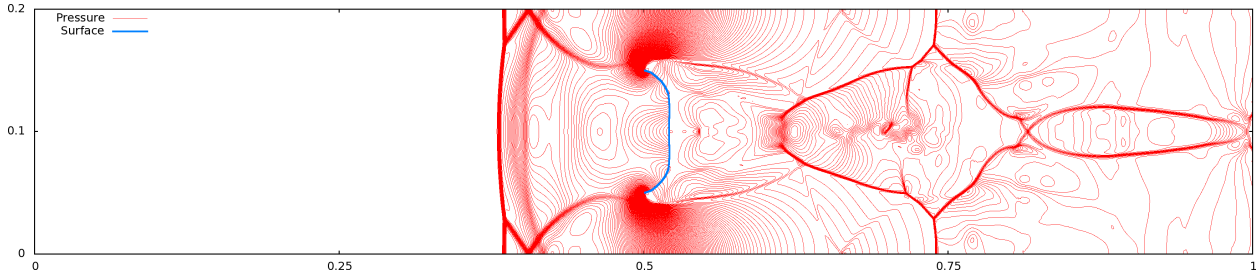
Figure 34: Time evolution of density in the example described in Section 5.3.5, on a 2560×512 grid.



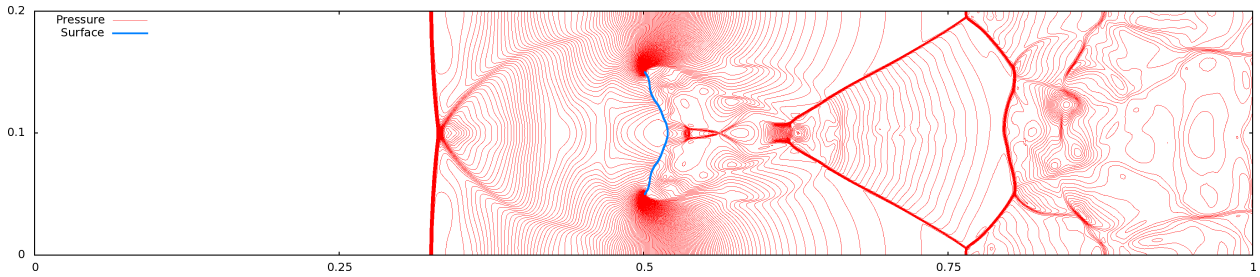
(a) $t = .07s$



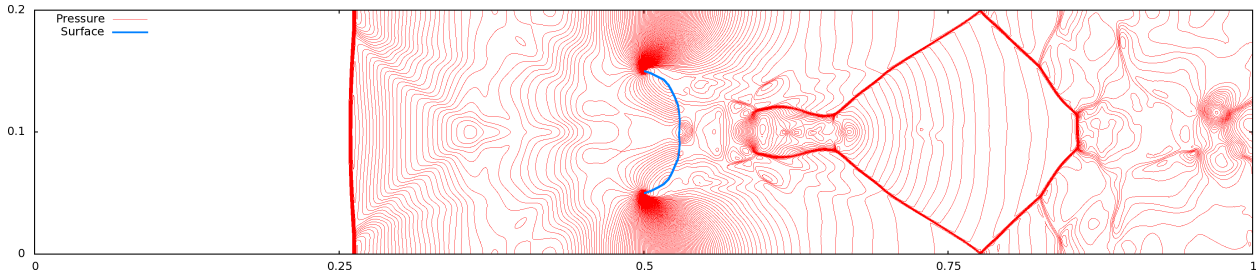
(b) $t = .14s$



(c) $t = .21s$

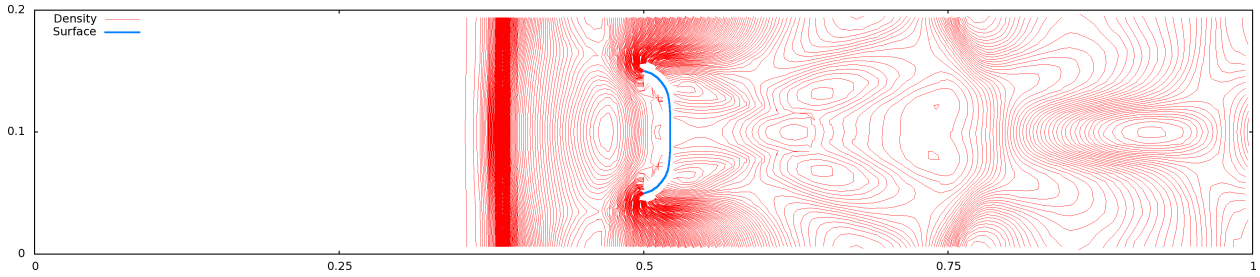


(d) $t = .28s$

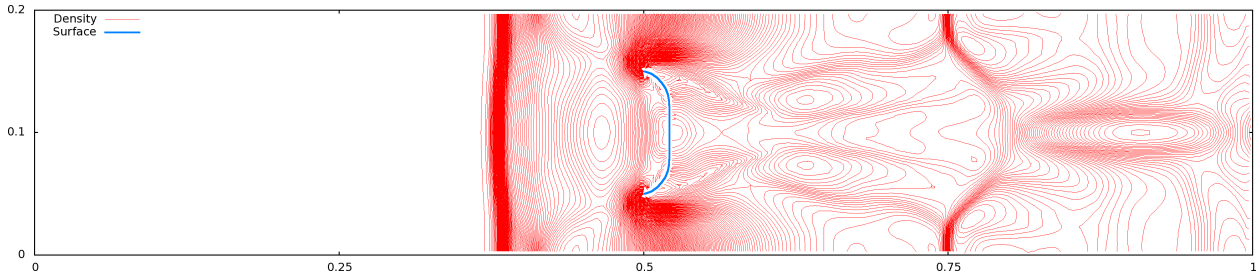


(e) $t = 35s$

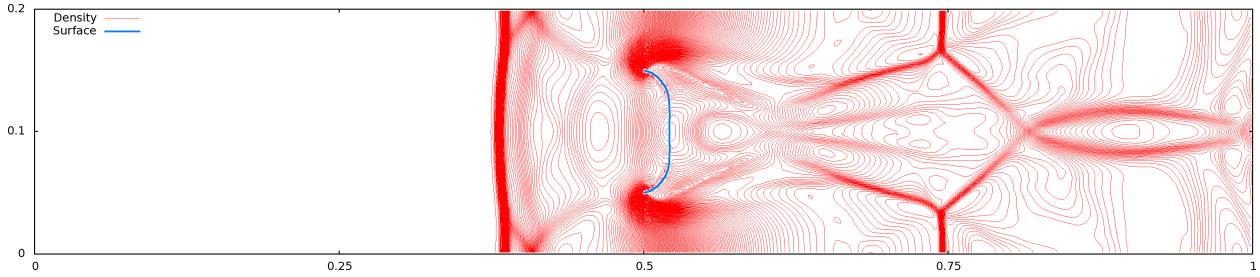
Figure 35: Time evolution of pressure in the example described in Section 5.3.5, on a 2560×512 grid.



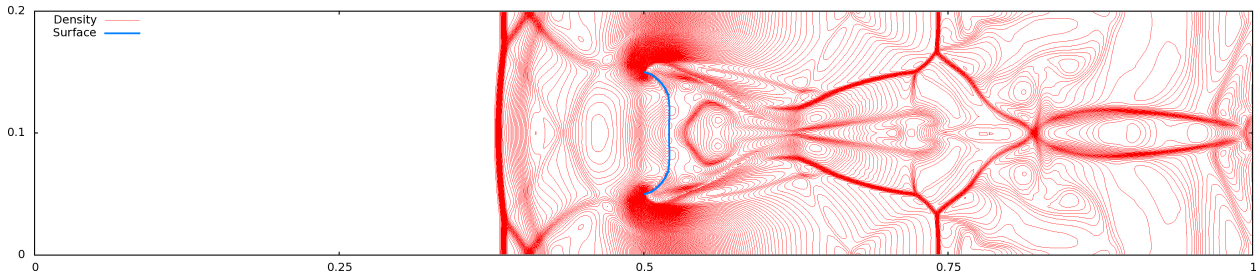
(a) Resolution 160×32



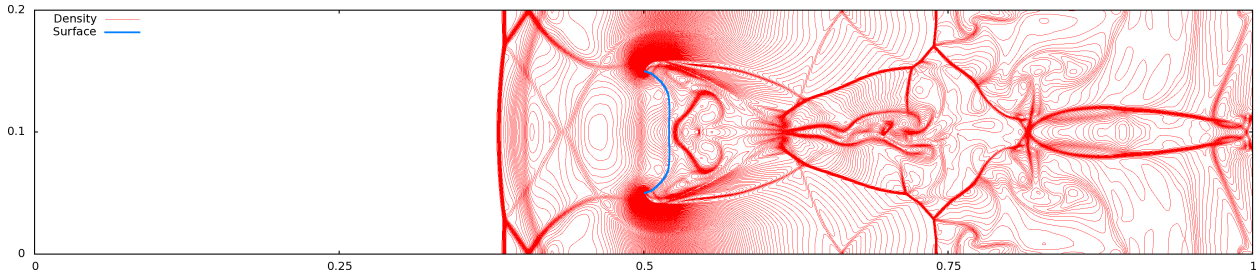
(b) Resolution 320×64



(c) Resolution 640×128



(d) Resolution 1280×256



(e) Resolution 2560×512

Figure 36: Grid convergence of the example described in Section 5.3.5, at time $t = .21s$.

6. Conclusions and future work

We have presented a fully conservative high resolution method for the simulation of two-way coupled fluid-structure phenomena. The method uses cut cells to track partial cell volumes, but requires no new time step restrictions to remain stable. Furthermore, it is robust to high fluid density to solid mass ratios and requires no special treatment for swept or uncovered cells. It is also collision-aware and therefore suitable for thin, impermeable solid structures separating disparate fluids. The resulting monolithic implicit system is symmetric positive-definite, and well-conditioned.

There are several interesting avenues of future work that we believe are worth exploring. The advection scheme drops to first order accuracy near the fluid-structure interface, introducing numerical error into the flow field, which may be mitigated by higher order accurate conservative semi-Lagrangian methods. One potential candidate would be the MacCormack method of [44]. The hybrid flux formulation also opens up some interesting possibilities, such as treating the flux region of the flow with a fully explicit method (i.e. fully explicit in the pressure as well) and enforcing pressure boundary conditions at the boundary, significantly reducing the fluid degrees of freedom of the coupled implicit solver. Moreover this would increase the accuracy in the flux-based region.

The implicit system also has some directions worth pursuing, such as the better treatment of slip boundary conditions within the solve by enforcing velocity compatibility at the structure interface, rather than at cell faces. For example in the implicit update, rather than lumping cut cell volumes into neighboring cells and enforcing compatibility through fluid grid cell faces (which causes geometric errors to appear in the flow due to the "stair-stepping" of the interface), one might consider enforcing compatibility on the structural interface directly, treating cut cells and partial volumes as additional degrees of freedom.

7. Acknowledgements

Research supported in part by ONR N00014-06-1-0505, ONR N00014-09-1-0101, ONR N-00014-11-1-0027, ARL AHPCRC W911NF-07-0027, NSF IIS-1048573, and the Intel Science and Technology Center for Visual Computing. Computing resources were provided in part by ONR N00014-05-1-0479.

8. Bibliography

[1] M. Arienti, P. Hung, E. Morano, and J.E. Shepherd. A level set approach to Eulerian–Lagrangian coupling. *J. Comput. Phys.*, 185(1):213–251, 2003.

[2] E. Aulisa, S. Manservisi, and R. Scardovelli. A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. *J. Comput. Phys.*, 188:611–639, 2003.

[3] E. Aulisa, S. Manservisi, and R. Scardovelli. A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking. *J. Comput. Phys.*, 197:555–584, 2004.

[4] P.T. Barton, B. Obadia, and D. Drikakis. A conservative level-set based method for compressible solid/fluid problems on fixed grids. *J. Comput. Phys.*, 230:7867–7890, 2011.

[5] D.J. Benson. An efficient, accurate, simple ALE method for nonlinear finite element programs. *Comput. Meth. in Appl. Mech. Eng.*, 72(3):305 – 350, 1989.

[6] M. Berndt, J. Breil, S. Galera, M. Kucharik, P.-H. Maire, and M. Shashkov. Two-step hybrid conservative remapping for multimaterial arbitrary Lagrangian-Eulerian methods. *J. Comput. Phys.*, 230:6664–6687, 2011.

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

[8] D. Calhoun. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *J. Comput. Phys.*, 176(2):231–275, 2002.

[9] C. Farhat, A. Rallu, and S. Shankaran. A higher-order generalized ghost fluid method for the poor for the three-dimensional two-phase flow computation of underwater implosions. *J. Comput. Phys.*, 227(16):7674–7700, 2008.

[10] C. Farhat, K. G. van der Zee, and P. Geuzaine. Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Comp. Meth. Appl. Mech. Eng.*, 195(17-18):1973 – 2001, 2006.

[11] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[12] 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.

[13] 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.

[14] 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.

[15] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.*, 24(3):973–981, 2005.

[16] D. Hartmann, M. Meinke, and W Schröder. A Cartesian cut-cell solver for compressible flows. *Comp. Meth. Appl. Mech. Eng.*, 200:1038–1052, 2011.

[17] D. Hartmann, M. Meinke, and W. Schröder. A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids. *Comp. Meth. App. Mech. Eng.*, 200(9–12):1038–1052, 2011.

[18] D.J.E. Harvie and D.F. Fletcher. A new volume of fluid advection algorithm: the stream scheme. *J. Comput. Phys.*, 162(1):1–32, 2000.

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

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

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

[22] Y. Kim and C.S. Peskin. 3-D parachute simulation by the immersed boundary method. *Comput. and Fluids*, 38(6):1080 – 1090, 2009.

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

[24] 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.

[25] M. Lentine, M. Aanjaneya, and R. Fedkiw. Mass and momentum conservation for fluid simulation. *SCA '11: Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 91–100, 2011.

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

[27] R.J. Leveque and K.M. Shyue. Two-dimensional front tracking based on high resolution wave propagation methods. *J. Comput. Phys*, 123:354–368, 1994.

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

[29] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 126:202–212, 1996.

[30] J. López, J. Hernández, P. Gómez, and F. Faura. An improved PLIC-VOF method for tracking thin fluid structures in incompressible two-phase flows. *J. Comput. Phys.*, 208(1):51 – 74, 2005.

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

[32] F. Losasso, G. Irving, E. Guendelman, and R. Fedkiw. Melting and burning solids into liquids and gases. *IEEE Trans. on Vis. and Comput. Graph.*, 12(3):343–352, 2006.

[33] 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.

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

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

[36] LG Margolin and M. Shashkov. Remapping, recovery and repair on a staggered grid. *Comput. Meth. in Appl. Mech. Eng.*, 193(39-41):4139–4155, 2004.

[37] V. Maronnier, M. Picasso, and J. Rappaz. Numerical simulation of free surface flows. *J. Comput. Phys.*, 155:439–455, 1999.

[38] C.S. Peskin. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10:252–271, 1972.

[39] C.S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[40] C.S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[41] S. Piperno, C. Farhat, and B. Larrouturou. Partitioned procedures for the transient solution of coupled aroelastic problems part I: Model problem, theory and two-dimensional application. *Comp. Meth. Appl. Mech. Eng.*, 124(1-2):79 – 112, 1995.

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

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

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

[45] J.H. Seo and R. Mittal. A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. *J. Comput. Phys.*, 230:7347–7363, 2011.

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

[47] H. Udaykumar, H. Kan, W. Shyy, and R. Tran-Son-Tay. Two-dimensional front tracking based on high resolution wave propagation methods. *J. Comput. Phys*, 123:354–368, 1994.

[48] P. Vachal, R. Garimella, and M. Shashkov. Untangling of 2D meshes in ALE simulation. *J. Comput. Phys.*, 196:627–644, 2004.

[49] H. Zhang, X. Zhang, S. Ji, Y. Guo, G. Ledezma, N. Elabbasi, and H. deCougny. Recent development of fluid-structure interaction capabilities in the ADINA system. *Comput. and Struct.*, 81(8-11):1071 – 1085, 2003. K.J Bathe 60th Anniversary Issue.

[50] L. Zhu and C.S. Peskin. Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method. *J. Comput. Phys.*, 179:452–468, 2002.