FULLY CONSERVATIVE LEAK-PROOF TREATMENT OF THIN SOLID STRUCTURES IMMERSED IN COMPRESSIBLE FLUIDS

A DISSERTATION
SUBMITTED TO THE INSTITUTE FOR COMPUTATIONAL
AND MATHEMATICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jón Tómas Grétarsson

March 2012

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ronald Fedkiw, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Eric Darve

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Charbel Farhat

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

In this dissertation we develop and present a leakproof approach to strongly coupled fluid-structure interactions in the presence of compressible fluids.

First we present a novel semi-implicit formulation of the Euler equations that separates the flux terms into an advection component and an acoustic component. The advection terms are treated explicitly using a standard flux-based scheme, and an implicit system of equations are derived for the pressure of the flow field. The implicit system of equations for pressure closely mirrors Poisson's equation that arises for *in*compressible flow, and indeed one obtains the Poisson equation for pressure in the limit as the sound speed goes to ∞. By treating the pressure implicitly we can alleviate the often-stringent acoustic component of the CFL restriction, and the resulting well-conditioned method depends only on the bulk velocity of the flow field for its time step restriction.

Next the implicit system of equations for pressure are integrated into a monolithic coupled system that robustly and stabily captures two-way coupled fluid-structure interactions. This formulation is quite general, and works with arbitrary fluid equations of state as well as both rigid and deforming structures without any special treatment. This tightly-coupled system captures the entire feedback loop that arises as fluid pressures and structure velocities interact, and so the method is suitable for capturing the behavior of flow near infinitesimally light structures (unlike partitioned methods) as well as extremely heavy structures. We exactly conserve momentum and kinetic energy within the coupled system, and hence naturally handle highly non-linear phenomena such as shocks, contacts and rarefactions near the fluid-structure interface. Note that this the method is *not* conservative near the interface during the advection step.

The advection stage is addressed in the third chapter, and a conservative semi-Lagrangian advection scheme is developed that works by supplementing a standard semi-Lagrangian advection with a conservative limiter and a forward-advection step. The conservative limiter clamps material motion from the first semi-Lagrangian step, guaranteeing that no new material is created, while the second forward-advection step is used to push forward any material that was left behind by the first step. We consider this advection scheme in its original habitat (incompressible flows), and

show that this method can be used to exactly conserve mass and momentum in such a flow. More interestingly, as the method works by tracing characteristic curves and interpolating values it is unconditionally stable. With this in mind we demonstrate that this *unconditionally stable* conservative advection scheme can be used to remove any and all remaining time step restrictions from the flux-split compressible flow previously introduced.

Finally, we introduce cut cells and partial volumes into the fluid-structure solver from Chapter 3 and modify the conservative semi-Lagrangian advection scheme to capture these small, irregular cell volumes without introducing any of the time step restrictions typically associated with cut cells. The semi-Lagrangian advection is limited to first order accuracy both in time and space, and so it is hybridized with a flux-based scheme and total variation-diminishing Runge-Kutta time integration, yielding a method that maintains high resolution accuracy in the bulk of the flow. The semi-Lagrangian method works by tracing characteristics, and so we modify it to enforce non-penetration through the structure interface by incorporating a temporal visibility map into the advection and clamping stages of the conservative semi-Lagrangian advection solver. Unlike previous methods, we do not require any complex geometric time evolution of volumes of material, nor do we require any special treatment for swept or uncovered cells. The resulting method can handle thin, moving solid structures in a fully conservative manner without any material leaking across the interface.

Acknowledgements

I would like to thank my advisor, Professor Ron Fedkiw, for providing guidance and advice at all stages of this study.

I would also like to extend my deepest gratitude to my reading committee, Charbel Farhat and Eric Darve, as well as my defense committee which also includes Margot Gerritsen and Adrien Lew.

I am indebted to Stanford University for Teaching Assistantship support during the Academic Years 2007-2008 and 2008-2009. The work presented here was supported in part by grants ONR N00014-06-1-0505 and ONR N00014-09-C-015.

I am humbled by the love, support and patience given to me by my family and friends. In particular, I would like to dedicate this thesis to my parents; my father, whose work in "tvífasastreymi" sparked my own interest in numerical methods, and my mother, whose creative talent and compassion is nothing short of divine.

Contents

List of Tables

List of Figures

Chapter 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 [10, 46] 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, 35, 20]. 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 [112, 84, 21], researchers have focused quite a bit of attention on so-called monolithic methods that employ higher degrees of implicit coupling [88, 31], 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 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 [23] and the immersed boundary method (see [82] and the references therein, including [80, 81]). 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 [114]), simply by adding that mass to the fluid cells that contain the solid structure. Whereas ghost 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 [34, 35] 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 [91]. 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 [61, 56, 100].

This dissertation builds towards and ultimately delivers a novel treatment for the cut cells and partial cell volumes near the structure interface. This is done in four stages; first, we propose a novel flux-split formulation of the governing equations for compressible flow and demonstrate that by treating the accoustic terms implicitly we can alleviate that component of the time step restriction, leaving only advection to limit the time step of the flow [50]. Next, the implicit accoustic solve is integrated into

a monolithic coupled fluid-structure solver, capturing the entire feedback loop between the fluid pressures and structure velocities. This implicit coupling appears to avoid adding any new time step restrictions to the flow, and conservatively captures the momentum and kinetic energy that moves between the structure and its surrounding fluid [31]. At this stage however the explicit advection solver is *not* conservative near the fluid-structure interface, and instead relies on ghost cells to populate swept and uncovered cells. In order to address this we then develop a conservative semi-Lagrangian advection scheme, and show that it is capable of conserving material as it moves around the grid [52]. Finally, we develop a number of extensions to this conservative advection scheme in order to make it suitable for use near the fluid-structure interface [30].

Using the semi-Lagrangian method to handle cells near the structure interface is similar in spirit to both volume of fluid (VOF) [38] and arbitrary Lagrange-Eulerian (ALE) [37] 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 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 [101] 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 is a very simple and straight-forward way to do this. We refer the interested reader to the following relevant VOF [36, 76, 2, 3, 64] and ALE papers [49, 70, 69, 74, 75, 7].

The material presented in this thesis is based on previously published works [50], [52], [31] and [30].

Chapter 2

Semi-implicit compressible flow

We propose a novel method for alleviating the stringent CFL condition imposed by the sound speed in simulating inviscid compressible flow with shocks, contacts and rarefactions. Our method is based on the pressure evolution equation, so it works for arbitrary equations of state, chemical species etc, and is derived in a straight-forward manner. Similar methods have been proposed in the literature, but the equations they are based on and the details of the methods differ significantly. Notably our method leads to a standard Poisson equation similar to what one would solve for incompressible flow, but has an identity term more similar to a diffusion equation. In the limit as the sound speed goes to infinity, one obtains the Poisson equation for incompressible flow. This makes the method suitable for two-way coupling between compressible and incompressible flows and fully implicit solid-fluid coupling, although both of these applications are left to future work. We present a number of examples to illustrate the quality and behavior of the method in both one and two spatial dimensions, and show that for a low Mach number test case we can use a CFL number of 300 (whereas previous work was only able to use a CFL number of 3 on the same example).

2.1 Introduction

In this paper, we focus on highly nonlinear compressible flows with shocks, contacts and rarefactions, for example the Sod shock tube. Traditionally these types of problems are solved with explicit time integration (Runge-Kutta methods, ENO, WENO etc, see e.g. [92, 93, 40]). Although these methods produce high quality results, small time steps are required in order to enforce the CFL condition of information moving only one grid cell per time step. While this is understandable for very high Mach number flow where $|u|$, $|u - c|$ and $|u + c|$ are all of similar magnitude, it is too restrictive for flows where the sound speed, c, may be much larger than $|u|$. Moreover some flow fields might have both high Mach number regions where shock waves are of interest as well as low Mach number regions where the material velocities are important. In this case, a large number of time steps are required if one is interested in the motion of the fluid particles over an appreciable distance in the low Mach number regions. Thus, it can be quite useful to have methods that avoid the stringent CFL time step restriction imposed by the acoustic waves and instead use only the material velocity CFL restriction (albeit one would expect some loss of quality because of the implicit treatment of the acoustic waves).

To alleviate the stringent CFL restriction, [41] proposed both a non-conservative and a conservative scheme. Their non-conservative scheme builds on the predictor-corrector type scheme of [109] to derive an elliptic pressure equation quite similar to ours, but for an adiabatic fluid. Our method is similar in spirit to [41, 103, 104, 107] where the calculation is divided into two parts: advection and non-advection. The advection terms are treated with explicit time integration, and thus the CFL restriction on the material velocity remains. Whereas one can use a standard method such as ENO in solving the advection terms, we found that when coupled to an implicit solution of the pressure equations (that is inherently central-differenced) the standard ENO method sometimes leads to spurious oscillatory behavior. Thus we designed a new ENO method geared towards a MAC grid discretization of the data, making it more similar to incompressible flow. We call this MAC-ENO or MENO. The remaining non-advection terms are solved using an implicit equation for the pressure

using a standard MAC grid type formulation. Since the MAC grid is dual in both velocity and pressure (noting that the MAC grid pressure needs to live at cell faces for flux based methods), one needs to interpolate data back and forth.

We base the derivation of our method on the pressure evolution equation as discussed in [24], thus making it valid for general equations of state, arbitrary chemical species etc. Thus, our derivation has fewer assumptions and is more straight forward than previous work, especially those based on preconditioners. For example, [103] makes two critical assumptions in their derivation of the implicit equation for pressure. In approximating the derivative of momentum they discard a $\Delta t \frac{\nabla p}{\rho}$ term, and their pressure evolution equation is missing the advection term. Also, our method is fully conservative and thus shocks are tracked at the right speed. We present a number of traditional examples for highly non-linear compressible flows including the Sod shock tube, interacting blast waves, and in two dimensions we show Flow Past a Step, Double Mach Reflection of a Strong Shock, and a Circular Shock. We also demonstrate that the method works well for low Mach number flow, taking an example from [42] where the authors obtain reasonable results with a CFL number of 3. Notably, our method allows a CFL number of 300 (two orders of magnitude more).

2.2 Numerical method

Let us consider the one dimensional Euler equations,

$$\begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ Eu + pu \end{pmatrix}_x = 0,$$

with ρ being the density, u the velocity, E the total energy per unit volume and p the pressure. The flux term can be separated into an advection part and a non-advection part,

$$\mathbf{F_1}(\mathbf{U}) = \begin{pmatrix} \rho u \\ \rho u^2 \\ Eu \end{pmatrix}, \mathbf{F_2}(\mathbf{U}) = \begin{pmatrix} 0 \\ p \\ pu \end{pmatrix}. \tag{2.1}$$

We first compute the Jacobian of the advection part

$$\mathbf{J} = \begin{pmatrix} 0 & 1 & 0 \\ -u^2 & 2u & 0 \\ -\frac{Eu}{\rho} & \frac{E}{\rho} & u \end{pmatrix}.$$

All the Jacobian's eigenvalues are equal to u, and it is rank deficient with left eigenvectors of $(u, -1, 0)$ and $(E/\rho, 0, -1)$ and right eigenvectors of $(1, u, 0)^T$ and $(0, 0, 1)^T$. Since all the characteristic velocities are identical, we can apply component wise upwinding to $\mathbf{F_1}(\mathbf{U})$ without having to transform into the characteristic variables first (as in [26]). Moreover, this advection part only requires a time step restriction based on u.

2.2.1 Implicit pressure update

The multi-dimensional Euler equations are

$$\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 \\ \rho uv \\ \rho uw \\ Eu \end{pmatrix}_x + \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ \rho vw \\ Ev \end{pmatrix}_y + \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 \\ Ew \end{pmatrix}_z + \begin{pmatrix} 0 \\ \nabla p \\ \nabla \cdot (p\vec{u}) \end{pmatrix} = 0,$$

where $\vec{u} = (u, v, w)$ are the velocities. Here we have advection components in each of the 3 spatial dimensions, and they can be handled as outlined previously in a dimension by dimension fashion (as in [93]).

We apply a time splitting as is typical for incompressible flow formulations, first updating the advection terms to obtain an intermediate value of the conserved variables $(\rho)^*$, $(\rho u)^*$, and E^*, and afterward correct these to time t^{n+1} using an implicit pressure. Since the pressure does not affect the continuity equation, $\rho^{n+1} = \rho^*$. The non-advection momentum and energy updates are

$$\frac{(\rho\vec{u})^{n+1} - (\rho\vec{u})^*}{\Delta t} = -\nabla p \tag{2.2}$$

and

$$\frac{E^{n+1} - E^*}{\Delta t} = -\nabla \cdot (pu). \tag{2.3}$$

Taking motivation from the standard incompressible flow formulation (which uses the momentum equation to derive an implicit equation for pressure), we divide equation (2.2) by ρ^{n+1},

$$\vec{u}^{n+1} = \vec{u}^\star - \Delta t \frac{\nabla p}{\rho^{n+1}}, \tag{2.4}$$

and take its divergence to obtain

$$\nabla \cdot \vec{u}^{n+1} = \nabla \cdot \vec{u}^\star - \Delta t \nabla \cdot \left(\frac{\nabla p}{\rho^{n+1}} \right). \tag{2.5}$$

In the case of incompressible flow, we would set $\nabla \cdot \vec{u}^{n+1} = 0$, but for compressible flow we instead use the pressure evolution equation derived in [24],

$$p_t + \vec{u} \cdot \nabla p = -\rho c^2 \nabla \cdot \vec{u}. \tag{2.6}$$

If we fix $\nabla \cdot \vec{u}$ to be at time $n+1$ through the time step (making an $\mathcal{O}(\Delta t)$ error), we can substitute in equation (2.5) to get

$$p_t + \vec{u} \cdot \nabla p = -\rho c^2 \nabla \cdot \vec{u}^\star + \rho c^2 \Delta t \nabla \cdot \left(\frac{\nabla p}{\rho^{n+1}} \right), \tag{2.7}$$

which is an advection-diffusion equation with a source term. Discretizing the $\vec{u} \cdot \nabla p$ advection term explicitly, using a forward Euler time step, and defining the diffusive pressure at time t^{n+1} as is typical for backward Euler discretization, gives after rearrangement

$$p^{n+1} - \rho^n (c^2)^n \Delta t^2 \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right) = (p^n - (\vec{u}^n \cdot \nabla p^n) \Delta t) - \rho^n (c^2)^n \Delta t \nabla \cdot \vec{u}^\star. \tag{2.8}$$

Note we have discretized ρc^2 at time t^n. This equation can be further simplified by using the advection equation for pressure,

$$\frac{p^a - p^n}{\Delta t} + \vec{u}^n \cdot \nabla p^n = 0$$

to obtain

$$p^a = p^n - (\vec{u}^n \cdot \nabla p^n)\Delta t, \tag{2.9}$$

where p^a is an advected pressure which can be computed using HJ ENO [79] or semi-Lagrangian advection [14]. Substituting in equation (2.8) we obtain

$$p^{n+1} - \rho^n (c^2)^n \Delta t^2 \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right) = p^a - \rho^n (c^2)^n \Delta t \nabla \cdot \vec{u}^\star. \tag{2.10}$$

We discretize this equation at cell centers (which is typical for advection-diffusion equations) and thus need to define velocities at cell faces for $\nabla \cdot \vec{u}^\star$. Consider two

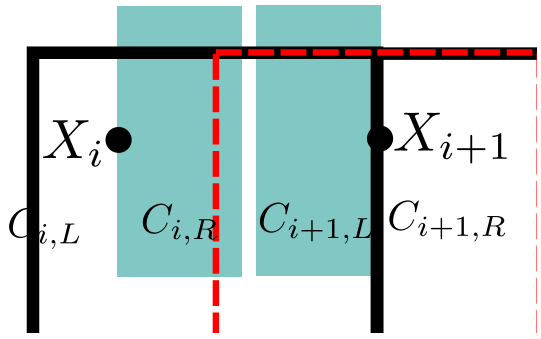


Figure 2.1:

adjacent grid cells, one centered at X_i and one centered at X_{i+1}. We divide these into four regions $C_{i,L}$, $C_{i,R}$, $C_{i+1,L}$, $C_{i+1,R}$, where $(C_{i,R} \cup C_{i+1,L})$ represents a dual cell (see figure 2.1). Then equation (2.2) for $C_{i,R}$ is

$$\frac{(\rho u)^{n+1}_{i,R} - (\rho u)^*_{i,R}}{\Delta t} = -\frac{p^{n+1}_{i+1/2} - p^{n+1}_i}{\Delta x/2}. \tag{2.11}$$

Similarly for $C_{i+1,L}$ we have

$$\frac{(\rho u)^{n+1}_{i+1,L} - (\rho u)^*_{i+1,L}}{\Delta t} = -\frac{p^{n+1}_{i+1} - p^{n+1}_{i+1/2}}{\Delta x/2}. \tag{2.12}$$

Adding these equations together and dividing by $(\rho_i + \rho_{i+1})$ yields

$$\frac{\hat{u}_{i+1/2}^{n+1} - \hat{u}_{i+1/2}^{*}}{\Delta t} = -\frac{p_{i+1}^{n+1} - p_i^{n+1}}{\Delta x \hat{\rho}^{n+1}}, \tag{2.13}$$

where $\hat{u}_{i+1/2} = \frac{(\rho u)_{i,R} + (\rho u)_{i+1,L}}{\rho_i + \rho_{i+1}} = \frac{(\rho u)_i + (\rho u)_{i+1}}{\rho_i + \rho_{i+1}}$ can be thought of as a density-weighted face velocity, and $\hat{\rho}_{i+1/2} = \frac{\rho_i + \rho_{i+1}}{2}$ is the cell face density. Note that we currently use $(\rho u)_{i,R} = (\rho u)_i$ and $(\rho u)_{i+1,L} = (\rho u)_{i+1}$, although higher order approximations could be used. Using this discretization on equation (2.10) yields

$$\left[I + \rho^n (c^2)^n \Delta t^2 G^T \left(\frac{1}{\hat{\rho}^{n+1}} G \right) \right] p^{n+1} = p^a + \rho^n (c^2)^n \Delta t G^T \vec{\hat{u}}^{\star}, \tag{2.14}$$

where G is our discretized gradient operator and $-G^T$ is our discretized divergence operator. This is solved to obtain p^{n+1} at cell centers.

In summary, instead of using an equation of state (EOS) to find the pressure for use as a flux in both conservation of momentum and energy, we use equation (2.14). The EOS still plays a role because it is used to determine the time t^n pressures which factor into p^a and is also used to determine $(c^2)^n$. In figure 2.2 we show an example calculation of the pressure for our Sod shock tube example. In the picture we plot the pressure using the equation of state at time t^n, i.e. p^n, the pressure calculated using equation (2.14), i.e. our p^{n+1}, and also the pressure calculated using the EOS applied to the conservative variables at time t^{n+1}, i.e. p_{EOS}^{n+1}. Notice in the figure that the pressure calculated from equation (2.14) is a good approximation to what the pressure will be at the next time step (i.e. p_{EOS}^{n+1}) emphasizing the implicit nature of our scheme. p^n is the pressure used in a typical explicit scheme.

It is interesting to note that this derivation does not require an ideal gas assumption, and hence should be general enough to work with any EOS (even multi-species flow [24]).

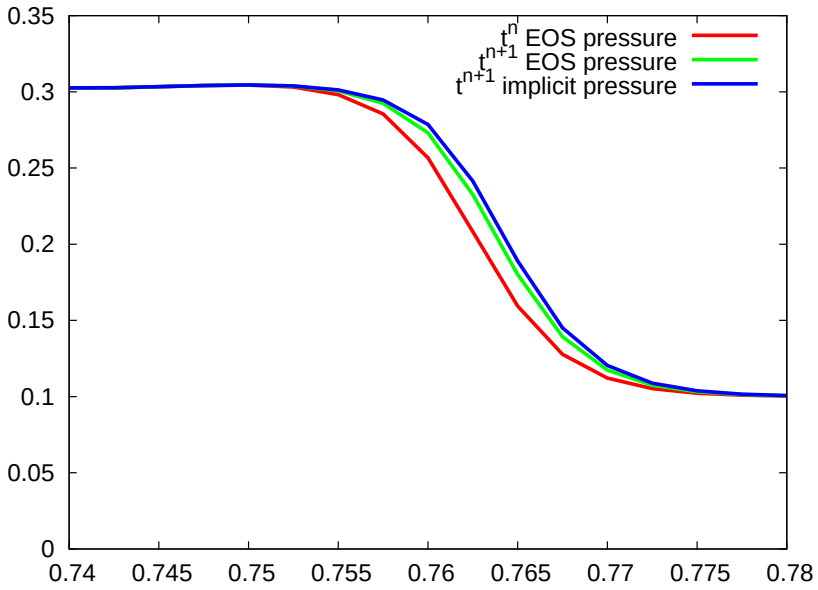


Figure 2.2: A blow-up of the pressure plot for example 6.1.1 at time $t(n) = .149s$ and $t(n+1) = .15s$, showing that the implicit pressure calculated in equation (2.14) is a good approximation to what the pressure will be at time t^{n+1} emphasizing the implicit nature of our scheme. p^n is also plotted to emphasize the difference between using an implicit and explicit pressure.

2.2.2 Updating momentum and energy

To obtain the correct shock speeds we use a flux based method and thus need the pressure at cell faces for equations (2.2) and (2.3), and the velocity at cell faces for equation (2.3). Applying conservation of momentum to the control volumes $C_{i,R}$ and $C_{i+1,L}$ (see figure 2.1) gives

$$Du_{i,R}/Dt = (p_i - p_{i+1/2})/(\Delta x \rho_{i,R}/2)$$

and

$$Du_{i+1,L}/Dt = (p_{i+1/2} - p_{i+1})/(\Delta x \rho_{i+1,L}/2).$$

The constraint that the interface remain in contact implies that $Du_{i,R}/Dt = Du_{i+1,L}/Dt$, which can be used with the aforementioned equations to solve for the pressure at the flux location $X_{i+1/2}$ as

$$p_{i+1/2} = \frac{p_{i+1}\rho_i + p_i\rho_{i+1}}{\rho_{i+1} + \rho_i}. \tag{2.15}$$

For solid wall boundaries, we reflect the pressure and density values as usual, and then use equation (2.15). The cell face velocity is computed via equation (2.13), and $p_{i+1/2}\hat{u}_{i+1/2}$ is used in equation (2.3).

2.3 Time step restriction

The eigenvalues of the Jacobian of the advection part of the flux are all u. Since we solve the acoustic component implicitly, we no longer have a severe time step restriction determined by the speed of sound c, and all that remains is to find an estimate for the maximum value of $|u|$ throughout the time step. Simply using u^n is not enough, since e.g. Sod shock tube starts out with an initial velocity identically zero and thus u^n would imply an infinite Δt. To alleviate this, we add a term that estimates the change in velocity over a time step similar to what was done in [43]. Assuming the flow is smooth, we combine conservation of mass and momentum to give an equation for the velocity, $u_t + u \cdot \nabla u + \frac{\nabla p}{\rho} = 0$. The temporal update of this equation would advect velocity based on the $u \cdot \nabla u$ term, but also increase the velocity by an amount equal to $\frac{\nabla p}{\rho}$. In one spatial dimension, we use this to estimate the velocity at the end of the time step as $\left(\frac{|u^n|_{max} + \frac{|p_x|}{\rho}\Delta t}{\Delta x}\right)$ and the CFL condition becomes

$$\Delta t \left(\frac{|u^n|_{max} + \frac{|p_x|}{\rho}\Delta t}{\Delta x}\right) \leq 1. \tag{2.16}$$

This is quadratic in Δt with solutions

$$\frac{-|u^n|_{max} - \sqrt{|u^n|^2_{max} + 4\frac{|p_x|}{\rho}\Delta x}}{2|p_x|/\rho} \leq \Delta t \leq \frac{-|u^n|_{max} + \sqrt{|u^n|^2_{max} + 4\frac{|p_x|}{\rho}\Delta x}}{2|p_x|/\rho}.$$

As the lower limit is always non positive and $\Delta t \geq 0$, we only need to enforce the upper bound. As $p_x \to 0$, both the numerator and denominator vanish and thus we obtain a more convenient time step restriction by replacing the 2^{nd} Δt in equation (2.16) with this upper bound to obtain

$$\frac{\Delta t}{2}\left(\frac{|u^n|_{max}}{\Delta x}+\sqrt{\left(\frac{|u^n|_{max}}{\Delta x}\right)^2+4\frac{|p_x|}{\rho\Delta x}}\right)\leq 1. \tag{2.17}$$

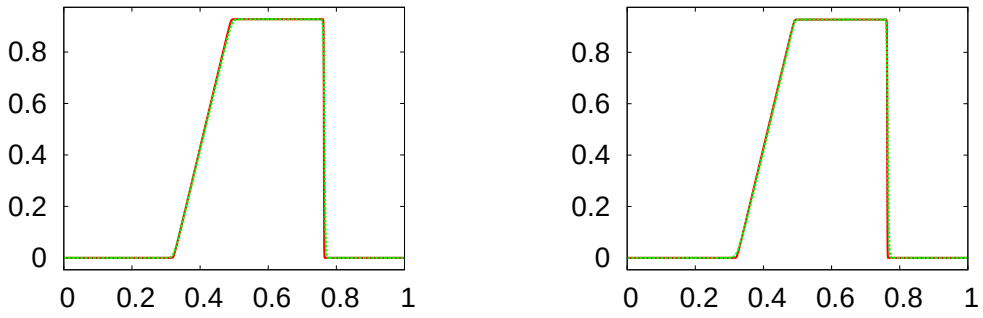
Note that this is not linear in Δx, but as $\Delta x \to 0$ we obtain a more typical CFL condition $\Delta t < \frac{\Delta x}{|u^n|_{max}}$. In two spatial dimensions our CFL follows along the lines of [43]'s equation 95 and is given by:

$$\frac{\Delta t}{2}\left(\frac{|u|_{max}}{\Delta x}+\frac{|v|_{max}}{\Delta y}+\sqrt{\left(\frac{|u|_{max}}{\Delta x}+\frac{|v|_{max}}{\Delta y}\right)^2+4\frac{|p_x|}{\rho\Delta x}+4\frac{|p_y|}{\rho\Delta y}}\right)\leq 1.$$

All of our examples are stable for CFL number $\alpha = .9$, and *all* of our examples were unstable for $\alpha = 1.3$. Some examples (e.g. example 6.1.8) blow up for $\alpha = 1$.

2.4 Modified ENO scheme

When using traditional ENO methods for the advection part of our equations (as in [93]), we obtained excessive spurious oscillations. This seems to be related to our dual cell center and MAC grid formulation, thus we device a new ENO scheme which better utilizes that dual formulation. We call this Mach-ENO or MENO. The main idea is to replace the advection velocity with the MAC grid value defined at the flux in question, i.e. $\hat{u}$. The lowest level of the divided difference table is typically constructed with the physical fluxes, i.e. ρu, ρu^2 and Eu for $\mathbf{F_1}(\mathbf{U})$ in equation (2.1). A dissipation term is added for the local and global Lax-Friedrichs versions. Consider constructing an ENO approximation for the flux at $X_{i+1/2}$. Locally, we would use a divided difference table with base values corresponding to the physical fluxes plus or minus the appropriate dissipation. Our modification is to replace $\rho_j u_j$, $\rho_j {u_j}^2$, and $E_j u_j$ with $\rho_j \hat{u}_{i+1/2}$, $\rho_j u_j \hat{u}_{i+1/2}$, and $E_j \hat{u}_{i+1/2}$ leaving the dissipation terms unaltered.



(a) Standard ENO-LLF (b) MENO-LLF

Figure 2.3: Sod shock tube problem at $t = .15s$. Left: Standard ENO-LLF (Local Lax-Friedrichs) using 401 grid points (green) and 1601 grid points (red). Right: The base 1601 grid points solution is the same as in the left figure, but the coarse grid calculation (with 401 grid points) is done with the new MENO scheme. Velocity is shown in both figures. Both simulations were done with explicit time integration and a full characteristic decomposition in order to demonstrate that the new ENO schemes performs similar to the old one when one is not using our new implicit discretization of the pressure.

Note that $\hat{u}_{i+1/2}$ is fixed throughout the divided difference table similar to the way one fixes the dissipation coefficient.

In order to validate our new MENO scheme, we compared it to the standard scheme from [93] for the standard Sod shock tube in Figure 2.3. For this problem and other fully explicit simulations the results were fairly similar, but when we ran the simulations with our semi-implicit formulation the MENO scheme performed much better, and in fact the standard ENO scheme was not successful in producing any solution whatsoever for figure 2.11 in our examples section.

2.5 Time integration

While the explicit component of our update is an upwind scheme, the implicit component is centrally-differenced. This tends to introduce more dispersive rather than

dissipative errors to the solution (i.e. there is more of an imaginary component to the eigenvalues), which suggests the use of Runge-Kutta over forward Euler.

We use two variations of the third order TVD Runge-Kutta scheme [92] in all of our examples. The first is to perform Runge-Kutta on just the advection part, $\mathbf{F_1}(\mathbf{U})$, with only one final implicit solve for $\mathbf{F_2}(\mathbf{U})$. The second variation is to carry out both $\mathbf{F_1}(\mathbf{U})$ and $\mathbf{F_2}(\mathbf{U})$ for each Runge-Kutta stage, noting that this has three times the computational cost as far as the implicit solution of $\mathbf{F_2}(\mathbf{U})$ is concerned. In general we observed better performance, especially in controlling overshoots, when using the second variation (see figure 2.4). However, some examples (in particular the high Mach number ones) do tend to show more oscillations (see figure 2.4, bottom). These oscillations are less predominant when combined with MENO, so we show all of our examples with the second variation.

2.6 Numerical results

2.6.1 One dimensional validation

For the one dimensional tests, we use a computational domain of $[0, 1]$, 401 grid points, and also plot a baseline solution using 1601 grid points in the standard fully explicit ENO method as in [93]. A second order ENO was used along with the CFL number of .5. Unless otherwise noted the maximum Mach number in each example lies within the range $(.9, 2.5)$. All units are in S.I. Generally speaking our method is a perturbation of those proposed by [103, 104] and thus demonstrates similar qualitative behavior. Timings are shown in table 2.1. In particular note that the implicit scheme is generally more efficient than the explicit scheme predominantly because we avoid the characteristic decomposition and can advect all three independent variables simultaneously because they all have the same eigenvalue u. At first glance one might assume that the necessity of a pressure Poisson equation would cancel out these efficiency gains, but practical experience shows only five or six iterations of conjugate gradients is required to reach a reasonable tolerance. It is unclear whether our newly proposed semi-implicit method would have these slight efficiency gains across

a wider number of examples and in multiple spatial dimensions, however for the low Mach number flow problems for which it was designed (such as example 6.1.8) it is significantly more efficient than the explicit method.

Sod shock tube

Our first test case is a standard Sod shock tube with initial conditions of

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

Our results are shown in Figure 2.5, which indicate well resolved shock, rarefaction and contact solutions. Since our method is conservative, we get the correct shock speeds. The results are comparable to that of [42] and [103].

Lax's shock tube

Lax's shock tube is similar in nature to Sod shock tube, except that the initial condition has a discontinuity in the velocity:

$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (.445,.698,3.528) & \text{if } x \leq .5, \\ (.5,0,.571) & \text{if } x > .5. \end{cases}$$

Our results are shown in Figure 2.6. Again, the results are comparable to the previous work.

Strong shock tube

The Strong shock tube problem poses initial conditions that generates a supersonic shock:

$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (1,0,10^{10}) & \text{if } x \leq .5, \\ (.125,0,.1) & \text{if } x > .5. \end{cases}$$

Our results are shown in Figure 2.7. The scheme admits some oscillations near the

rarefaction wave, and we see no notable difference in simulation time when compared to the explicit simulation. With that in mind, we note that the main advantage of the proposed method is to take time steps irrespective of the sound speed values; in cases of high Mach number flows (or high Mach number regions of the flow – if asynchronous time integration is used), one could use a typical ENO scheme.

Mach 3 shock test

The initial conditions for the Mach 3 shock test are:

$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (3.857, .92, 10.333) & \text{if } x \leq .5, \\ (1, 3.55, 1) & \text{if } x > .5. \end{cases}$$

Our results are shown in Figure 2.8. As above we do note some oscillations near the rarefaction wave.

High mach flow test

The initial conditions for the High mach flow test are:

$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (10, 2000, 500) & \text{if } x \leq .5, \\ (20, 0, 500) & \text{if } x > .5. \end{cases}$$

As noted in [42] the Mach number in this test can reach as high as 240. Our results are shown in Figure 2.9.

Interaction of blast waves

Here we present a test of two interacting blast waves. This problem was introduced by [102] and involves multiple strong shock waves. The initial conditions for the test

are:

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

We also have solid wall boundary conditions at $x = 0$ and $x = 1$. Our results are shown in Figure 2.10 which shows that we achieve very accurate results.

Two symmetric rarefaction waves

In this test there are two rarefaction waves going in opposite directions from the center of the domain. This causes very low density regions near the center of the domain. The initial conditions for the test are:

$$(\rho(x,0),u(x,0),p(x,0)) = \begin{cases} (1,-2,.4) & \text{if } x \le .5, \\ (1,2,.4) & \text{if } x > .5. \end{cases}$$

Our results are shown in Figure 2.11. Our results are comparable to that of [42] and [103]. Note that there is an unphysical pulse in the internal energy field near the low pressure region, caused by overheating (see e.g. [25]).

Smooth flow test (Mach zero limit)

The initial conditions for the zero mach limit test are given by:

$$\begin{aligned} u(x,0) &= 0 \\ p(x,0) &= p_0 + \epsilon p_1(x) \\ p_1(x) &= 60\cos(2\pi x) + 100\sin(4\pi x) \\ \rho(x,0) &= \left(\frac{p(x,0)}{p_0}\right)^{\frac{1}{\gamma}} \rho_0 \end{aligned}$$

Where $\rho_0 = 1$, $p_0 = 10^9$, and $\epsilon = 10^3$. Since the flow is smooth and there are no shocks in this test, we have used a single implicit solve per time step. This test is dominated by acoustic waves (as observed in [42]). We can take time steps as

large as is permitted by our CFL condition in equation (2.17). This permits time steps three orders of magnitude greater than those permitted by sound-speed based CFL. However, as with all implicit schemes, taking too large a time step can lead to inaccurate results. Thus, in order to get sufficient accuracy, we clamp our time step to be a fixed multiple of the explicit time step (which is calculated using the sound-speed based CFL). In figure 2.12 we use 3 times the explicit time step and show convergence via grid resolution.

In a second suit of tests we show that we can increase the grid resolution without the need to refine the time step. The timing results for this experiment are available in table 2.2, where Δt remains fixed as the grid resolution goes up as high as $320,000$ grid cells. At that point the effective sound speed CFL is 300. Numerical results are plotted in figure 2.13 and table 2.2 summarizes the results. In particular we note that the newly proposed implicit method permits a fixed time step all the way up to $320,000$ grid points. This allows the wall clock simulation time to scale approximately linear to the size of the problem (since we solve the Poisson equation using conjugate gradients, which has superlinear complexity – however, note that one only needs the solver to converge in the sense of truncation error as opposed to round-off error). On the other hand, in explicit methods the simulation time grows quadratically, becoming impractical at $320,000$ grid points. Note that since we are not refining the time step, we do not expect to see any further convergence in the solution.

2.6.2 Flow past a step test

Our first two dimensional experiment is similar to the one described in [25]. We assume an ideal gas with $\gamma = 1.4$. The test domain is 3 units long and 1 unit wide, with a .2 unit high step which is located .6 units from the left hand side of the tunnel. The initial conditions are $\rho = 1.4$, $p = 1$ and $u = 3$ and $v = 0$ everywhere in the domain. We apply an inflow boundary condition on the left hand side of the domain, and an outflow boundary condition on the right hand side of the domain. A reflective solid wall boundary condition is applied for the top and bottom boundaries of the domain. We show numerical results at $t = 4s$ on a grid of resolution 120x40 in

figure 2.14.

2.6.3 Double mach reflection of a strong shock

In a computational domain of $[0,4] \times [0,1]$, a planar Mach 10 shock hits a reflecting boundary that lies along the bottom wall of the domain along $x \in [\frac{1}{6}, 4]$. The plane of the shock begins at $(\frac{1}{6}, 0)$ and makes a 60° angle with the reflecting plane. The left and bottom (for $x \in (0, \frac{1}{6})$) boundary conditions are given by the postshock condition, the right boundary by a zero-gradient condition, and the top boundary is set to describe the exact motion of the Mach 10 shock. If we take $\vec{n}$ to be the unit vector that lies normal to the planar shock, then the initial values are given by:

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

Our method (see figure 2.15) compares well with those provided in [102], which provides a description of this example and presents numerical results comparing the performance of various methods in this problem. As is done in previous work we only show the domain of interest ($[0,3] \times [0,1]$).

2.6.4 Circular shock test

The circular shock test has an initial condition prescribed as

$$(\rho, u, v, p) = \begin{cases} (1,0,0,1) & \text{if } r \leq .4 \\ (.125,0,0,.1) & \text{if } r > .4, \end{cases}$$

where $r = \sqrt{x^2 + y^2}$. Numerical results are shown in figure 2.16. The same test was shown in [104]. Our results indicate well resolved shock and contact solutions along with correct speed shock calculations.

2.7 Conclusions and future work

We have presented a method for alleviating the stringent CFL condition imposed by the sound speed in highly non-linear compressible flow simulations. A fractional step procedure combined with the pressure evolution equation is used. The method works for arbitrary equations of state, and in the limit as the sound speed goes to infinity it yields the Poisson equation for incompressible flow. We also presented a Mach-ENO or MENO scheme which better utilizes a dual cell center and MAC grid formulation. The numerical experiments on various benchmark problems for one and two dimensions indicate that our semi-implicit method obtains well resolved shock, rarefaction and contact solutions. Since our method is conservative, we also obtain correct shock speeds. The smooth flow example illustrates the ability of our method to take significantly large time steps for low Mach number flows as compared to explicit methods. In future work we plan to extend our approach to handle two-way coupling between compressible and incompressible flows, as well as fully implicit solid-fluid coupling.

2.8 Appendix: boundary conditions

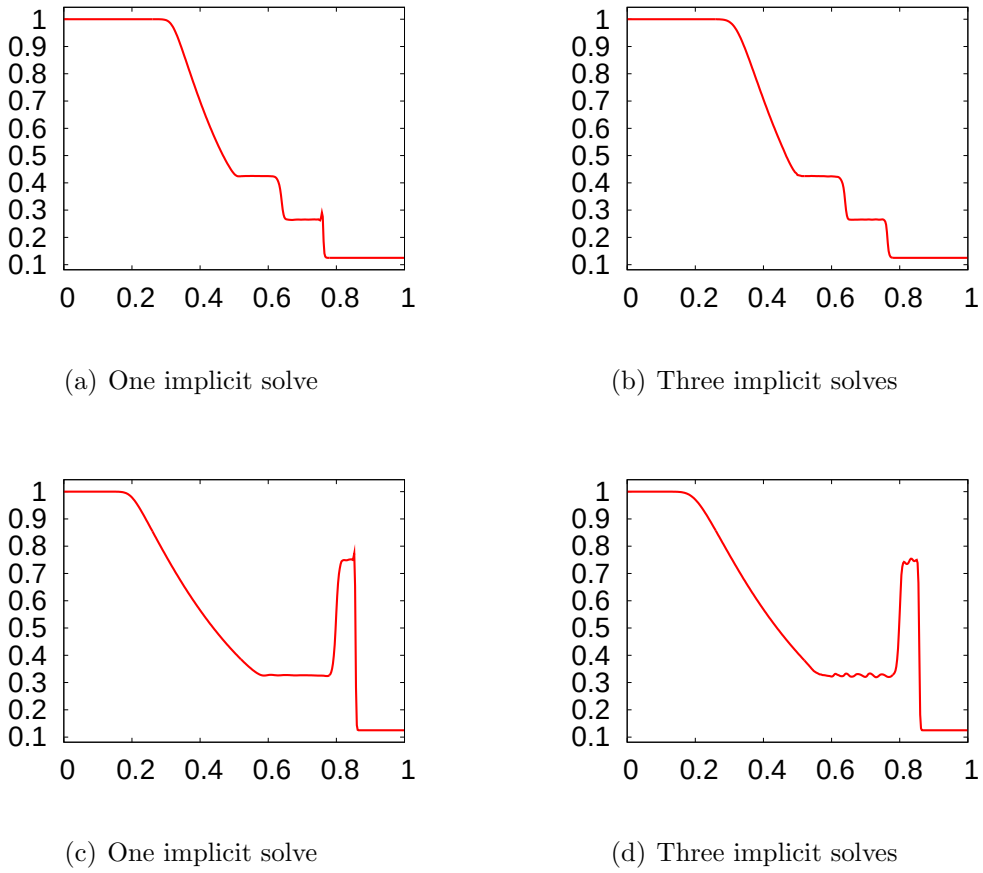
Figure 2.14 requires the handling of inflow and outflow boundary conditions. We define U_{out} to be the outgoing state and U_{in} to be the ingoing state. The outgoing state, U_{out}, is obtained by simple extrapolation whereas the ingoing state, U_{in}, is obtained by attenuating U_{out} towards specified far-field values. After defining U_{out} via extrapolation, we average the primitive variables to cell flux on the boundary of the domain, and use those values to compute a characteristic decomposition. If the p^{th} characteristic field indicates ingoing information, then when applying the ENO scheme in this characteristic field we use U_{in} for the ghost node values. Otherwise U_{out} is used. Note for higher order schemes boundary values will be needed for fluxes on the interior of the domain as well, and we choose the ghost nodes (as U_{in} or U_{out}) in the same fashion.

Our ingoing state, U_{in}, is obtained by attenuating the extrapolated state, U_{out},

towards a given far field state, U_{far}. This is accomplished by multiplying U_{out} with each of the left eigenvectors, attenuating if the eigenvalue in that characteristic field indicates an ingoing wave, and then multiplying by the right eigenvector. Defining the scalar characteristic information in each field as $\xi^p = L^p U_{out}$, we would attenuate ξ^p towards ξ^p_{far} using the analytic solution of the ODE

$$d\xi/dt = K(\xi - \xi_{far})$$

for time step Δt using initial data of $\xi = \xi_{out}$. We used an attenuation coefficient of $K = -.5$ in our examples.



(a) One implicit solve

(b) Three implicit solves

(c) One implicit solve

(d) Three implicit solves

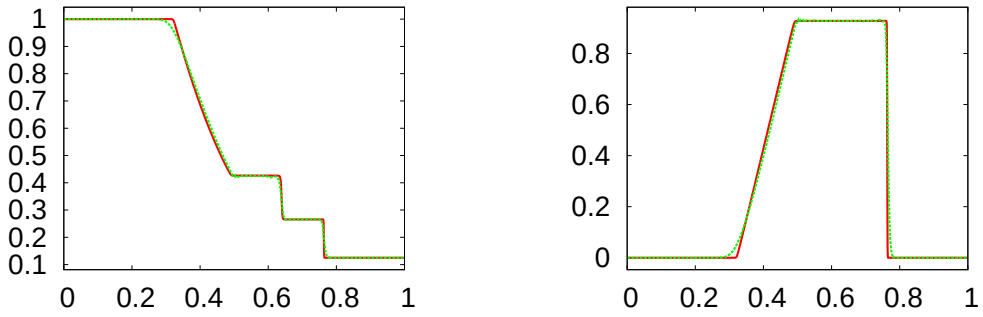
Figure 2.4: Numerical results comparing placing the implicit solve either inside each Runge-Kutta stage (b and d) or once after a full three stage Runge-Kutta cycle (a and c). The top two figures show the results for a Sod shock tube problem at $t = .15s$, the bottom two figures show the results for a strong shock tube problem at $t = 2.5 \times 10^{-6}s$. Density is shown in all figures. Note the spurious overshoots when the implicit solve is not included in the Runge-Kutta cycle (left two figures). Note that we use the standard ENO scheme from [93] (not MENO) for these four examples.

Test name	semi-implicit (seconds)	explicit (seconds)
Sod shock tube	2.95	3.69
Lax shock tube	2.71	4.53
Strong shock tube	2.43	3.43
Mach 3 shock test	2.90	3.59
High Mach flow test	3.75	3.29
Interaction of blast waves (Bang Bang)	5.28	9.86
Two symmetric rarefaction waves	3.52	4.15

Table 2.1: Wall clock times comparing the semi-implicit method with the fully explicit method, for 1-D examples. Simulations were run to the target times of each example as mentioned in their respective figures.

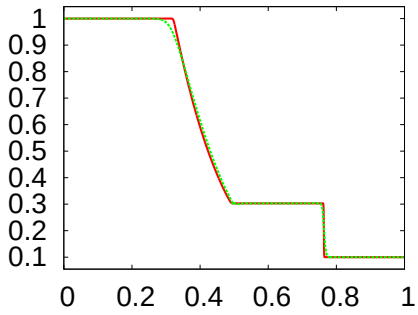
Grid Resolution	Effective sound speed CFL	Δt	Wall clock time (Implicit)	Wall clock time (Explicit)
3200	3	5.01e-08	63.41s	511.67s
32000	30	5.01e-08	810.03s	60498.49s
320000	300	5.01e-08	9976.58s	Impractical

Table 2.2: Timing results for smooth flow test, with Δt approximately constant. The wall clock times are shown for simulations till $t = 5 \times 10^{-5} s$.

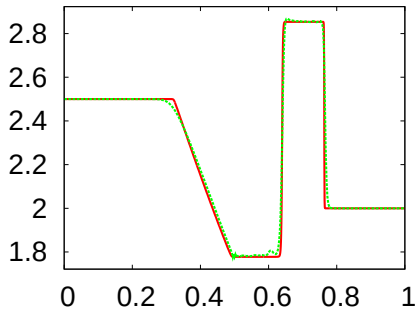


(a) Density

(b) Velocity

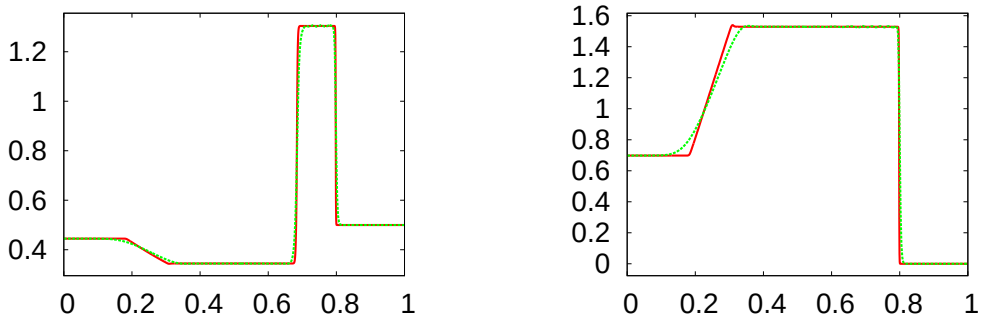


(c) Pressure



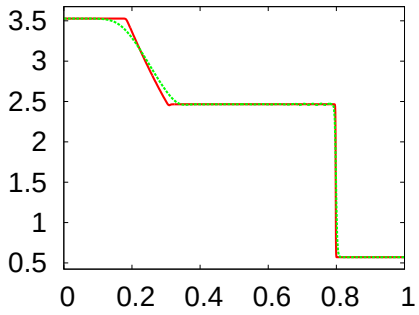
(d) Internal Energy

Figure 2.5: Numerical results of the Sod shock tube problem at $t = .15s$. The explicit baseline solution is plotted in red, and the solution from our method is plotted in dotted green.

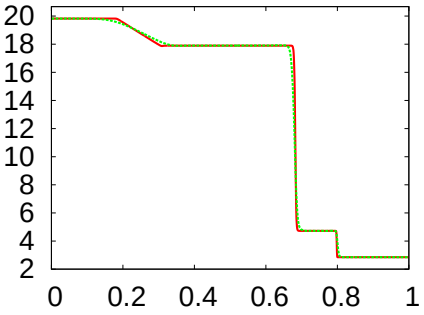


(a) Density

(b) Velocity

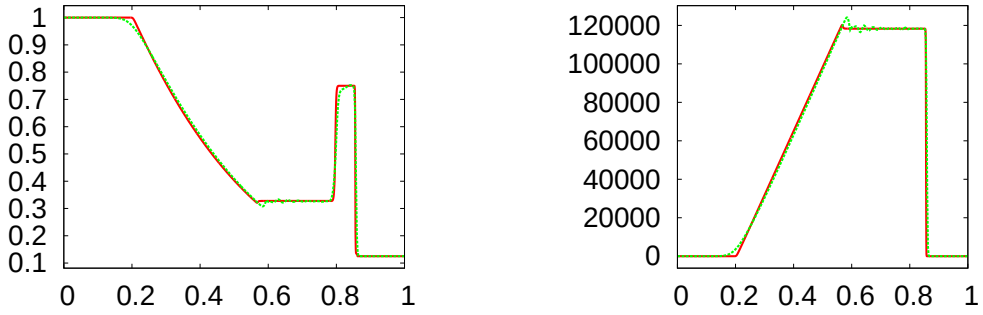


(c) Pressure



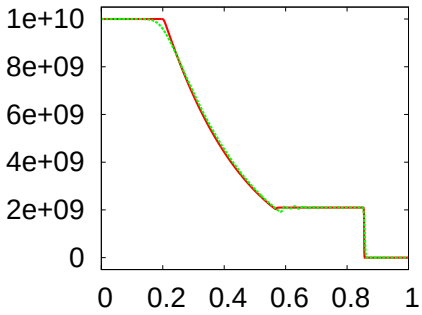
(d) Internal Energy

Figure 2.6: Numerical results of the Lax's shock tube problem at $t = .12s$. The explicit baseline solution is plotted in red, and the solution from our method is plotted in dotted green.

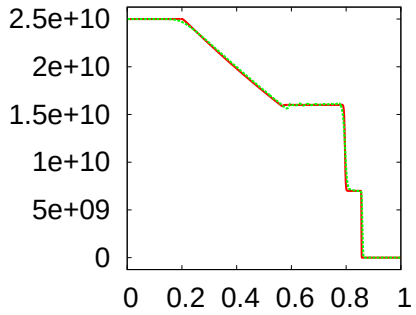


(a) Density

(b) Velocity

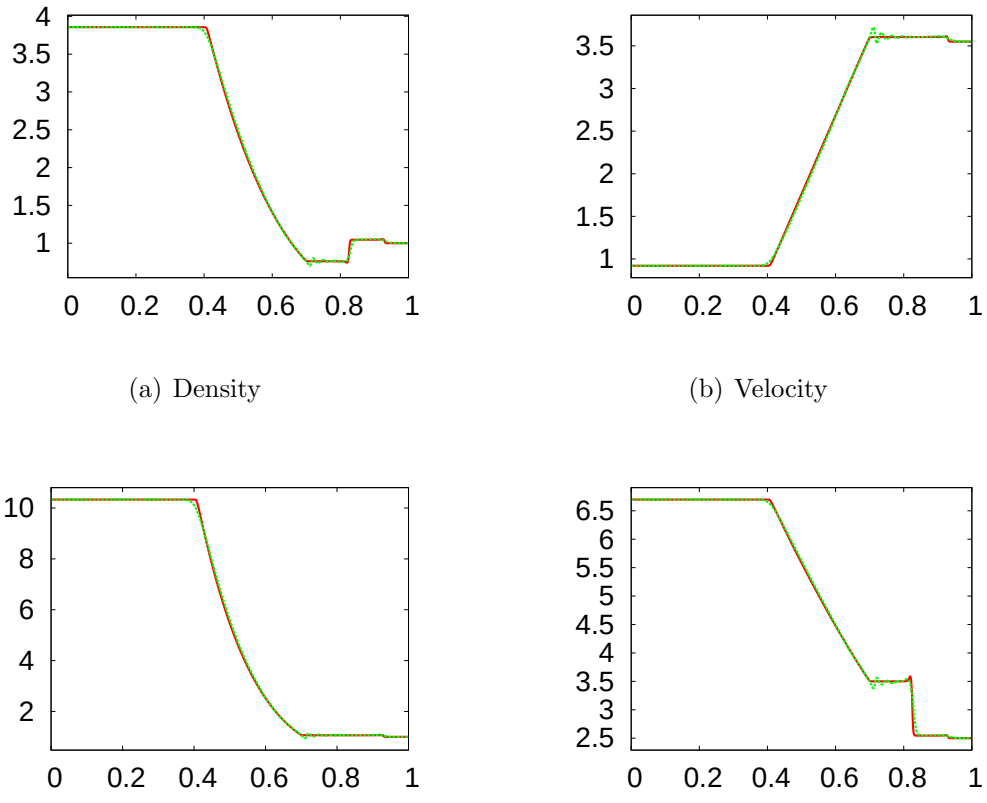


(c) Pressure



(d) Internal Energy

Figure 2.7: Numerical results of the strong shock tube problem at $t = 2.5 \times 10^{-6}s$. The explicit baseline solution is plotted in red, and the solution from our method is plotted in dotted green.



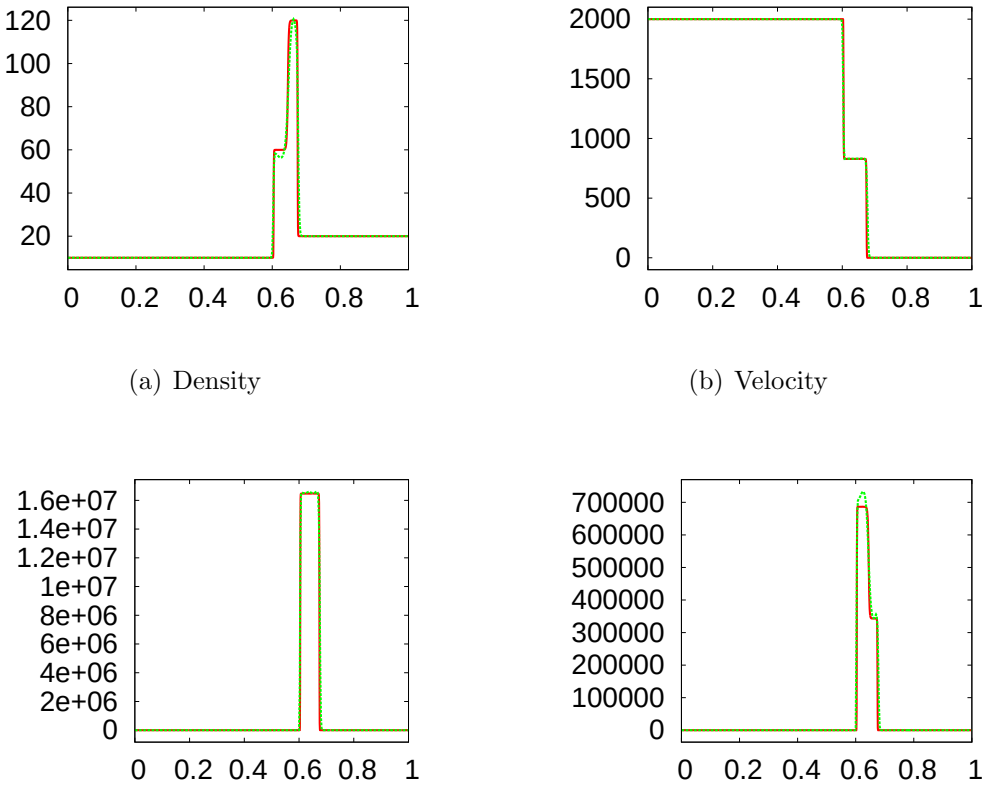
(a) Density

(b) Velocity

(c) Pressure

(d) Internal Energy

Figure 2.8: Numerical results of the Mach 3 shock tube problem at $t = .09s$. The explicit baseline solution is plotted in red, and the solution from our method is plotted in dotted green.



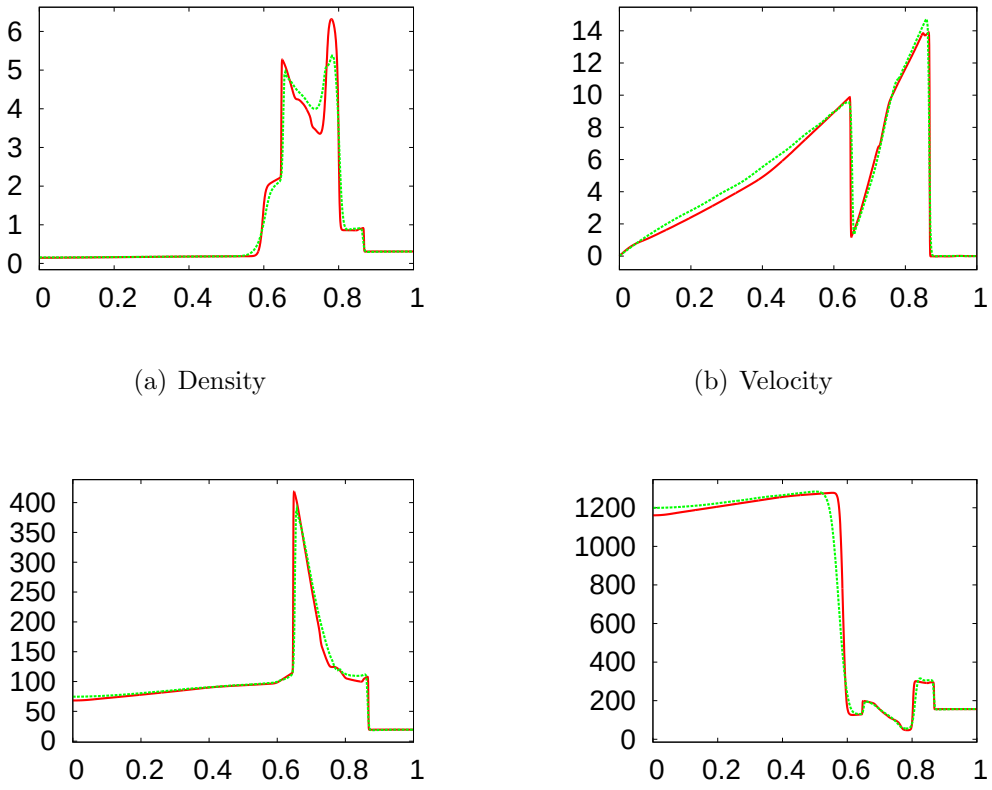
(a) Density

(b) Velocity

(c) Pressure

(d) Internal Energy

Figure 2.9: Numerical results of the High Mach shock tube problem at $t = 1.75 \times 10^{-4}s$. The explicit baseline solution is plotted in red, and the solution from our method is plotted in dotted green.



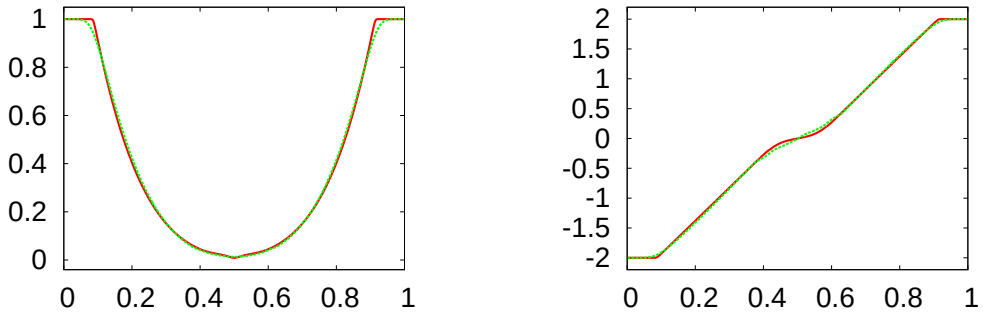
(a) Density

(b) Velocity

(c) Pressure

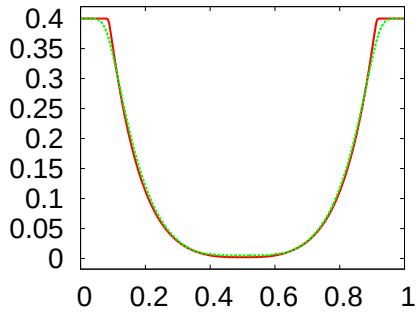
(d) Internal Energy

Figure 2.10: Numerical results of the interacting blasts shock tube problem at $t = .038s$. The explicit baseline solution is plotted in red, and the solution from our method is plotted in dotted green.

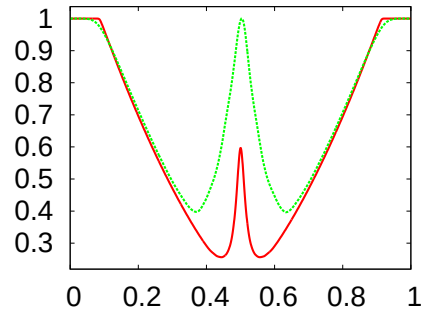


(a) Density

(b) Velocity



(c) Pressure



(d) Internal Energy

Figure 2.11: Numerical results of the symmetric rarefaction shock tube problem at $t = .15s$. The explicit baseline solution is plotted in red, and the solution from our method is plotted in dotted green.

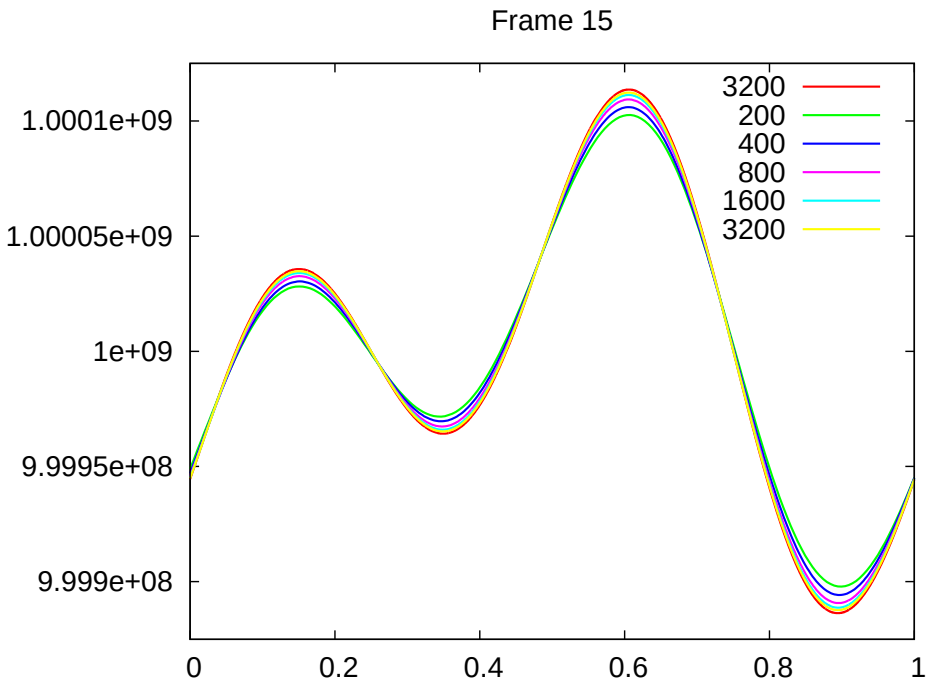


Figure 2.12: Numerical results comparing the pressure in smooth flow test at 200, 400, 800, 1600, and 3200 grid cells with an effective sound speed based CFL number 3 at $t = 1.5 \times 10^{-5}s$. The red curve is the explicit simulation run at 3200 grid cells with a CFL number .5.

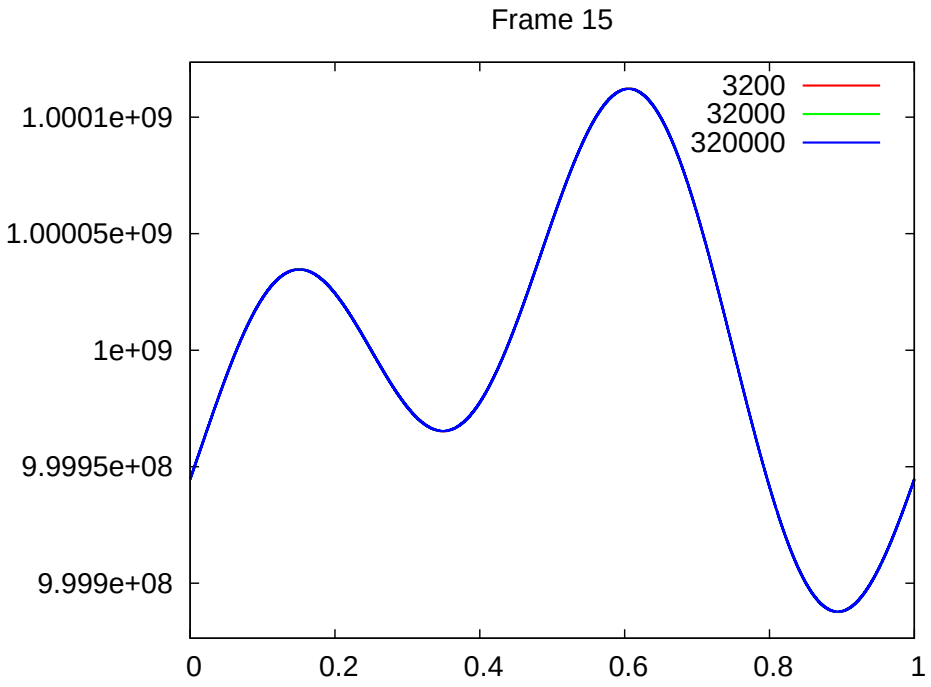


Figure 2.13: Numerical results showing pressure in the smooth flow test at 3200, 32000 and 320000 grid cells. We used an effective sound speed based CFL number of 3, 30 and 300 respectively at $t = 1.5 \times 10^{-5}s$. Since Δt stays constant, the solution remains relatively unchanged even as we get huge time step gains.

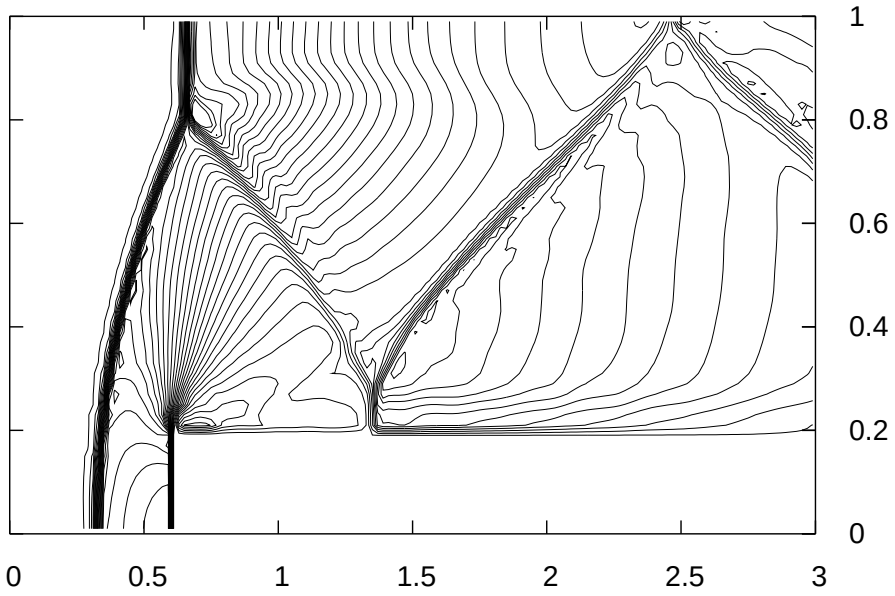


Figure 2.14: Numerical results showing the contour plots of density for the flow past a step test on a grid of size 120x40 at $t = 4s$. 30 contours are plotted in the range $[.2568, 6.067]$.

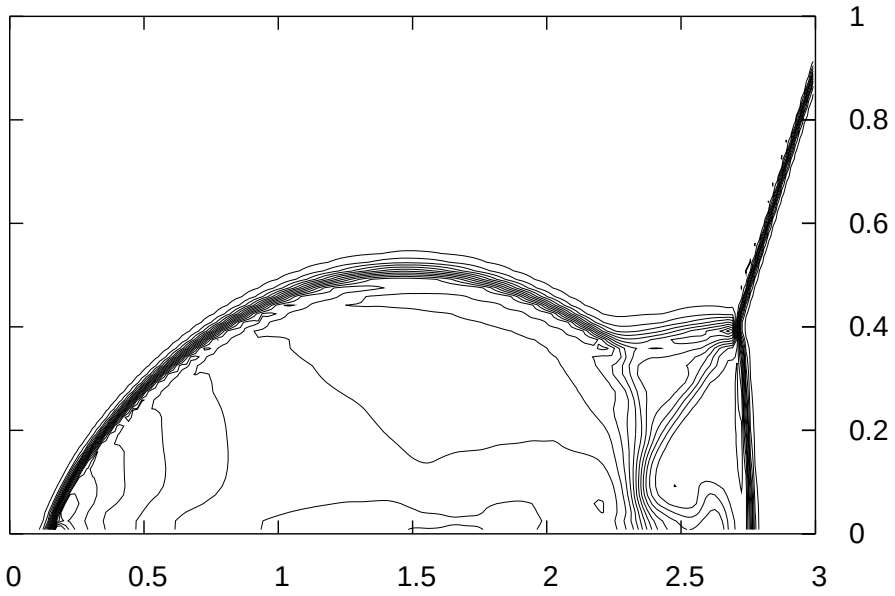
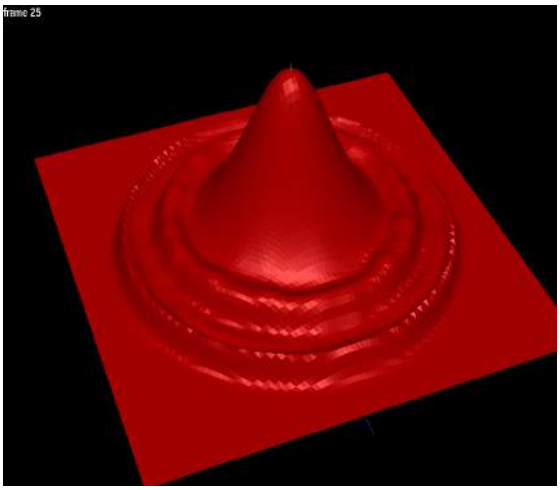
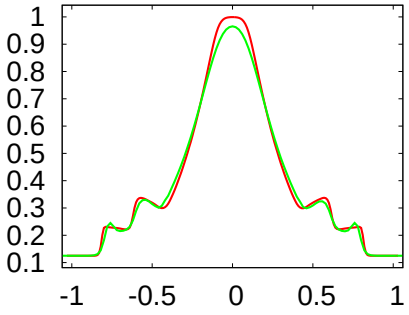


Figure 2.15: Numerical results showing the contour plots of density for the double mach reflection of a strong shock on a grid of size 240x60 at $t = .2s$. 30 contours are plotted within the range $[1.731, 20.92]$.



(a) Density



(b) 1-D slice of density, with the reference solution shown in red.

Figure 2.16: Numerical results for the circular shock test on a grid of size 100x100 at $t = .25s$.

Chapter 3

Compressible FSI

We propose a novel method to implicitly two-way couple Eulerian compressible flow to volumetric Lagrangian solids. The method works for both deformable and rigid solids and for arbitrary equations of state. The method exploits the formulation of [50] which solves compressible fluid in a semi-implicit manner, solving for the advection part explicitly and then correcting the intermediate state to time t^{n+1} using an implicit pressure, obtained by solving a modified Poisson system. Similar to previous fluid-structure interaction methods, we apply pressure forces to the solid and enforce a velocity boundary condition on the fluid in order to satisfy a no-slip constraint. Unlike previous methods, however, we apply these coupled interactions implicitly by adding the constraint to the pressure system and combining it with any implicit solid forces in order to obtain a strongly coupled, symmetric indefinite system (similar to [89], which only handles incompressible flow). We also show that, under a few reasonable assumptions, this system can be made symmetric positive-definite by following the methodology of [88]. Because our method handles the fluid-structure interactions implicitly, we avoid introducing any new time step restrictions and obtain stable results even for high density-to-mass ratios, where explicit methods struggle or fail. We exactly conserve momentum and kinetic energy (thermal fluid-structure interactions are not considered) at the fluid-structure interface, and hence naturally handle highly non-linear phenomenon such as shocks, contacts and rarefactions.

3.1 Introduction

Direct numerical simulations (DNS) are often used to study the interactions between fluid flows and solid structural models. Under certain assumptions these can be reduced to a one-way coupled system; for example if one wishes to determine the steady-state lift of an airfoil in subsonic flow, it is often reasonable to simulate the airfoil as a kinematic body. With a clever choice of boundary conditions, one can even begin to examine two-way coupled interactions, albeit in a limited fashion. In the more general case, these assumptions miss the interesting two-way coupled interactions between the fluid and the structure. These two-way coupled interactions can be quite important and, if not properly captured in the DNS, can lead to non-physical results. It is therefore important to have a robust numerical method that accurately captures two-way coupled interactions across a fluid-structure interface.

Methods to capture fluid-structure interactions can be broadly separated into two categories. Weakly coupled (partitioned) systems interleave the disparate subsystems by integrating them forward in time separately, using each others' results as boundary conditions in an alternating one-way coupled fashion (see e.g. [112, 84, 21]). This approach is appealing as it permits the use of specialized numerical methods for each of the different materials with only slight modifications to account for the modified time integration and changing boundaries. There are disadvantages to this approach, however, for example new and poorly understood stability restrictions arise independent of the individual subsystems, such as the lumped-mass instability discussed in [11]. The alternative is to employ a strongly coupled (monolithic) system, which are systems where the fluid and structure are evolved forward in time simultaneously using a solver specially crafted to incorporate phenomena from both fluid and solid phases. Our method is a hybrid of the two; the explicit components of both fluid and solid solvers are evolved forward independently, while the implicit components and interactions are coupled together in a monolithic solve.

State-of-the-art solvers typically use an Eulerian framework to treat fluid flows and a Lagrangian framework to treat solids, and so any coupled system must do one of three things: model the solid in an Eulerian framework, model the fluid in

a Lagrangian framework, or find a way to couple Eulerian fluids with Lagrangian solids. The first two options are undesirable as they impose significant limitations on the numerical method, for example Eulerian models only capture material properties (rather than tracking them) which makes it difficult to compute time history variables important to structural simulation, such as loading and damage. Many fluid Lagrangian models have difficulty in obtaining the correct shock speeds due to the lack of discrete flux differencing, and therefore resort to artificial viscosity methods that require a number of zones within a shock in order to obtain the right speed [5, 6]. Lagrangian fluid models also struggle with high-speed and deforming flows, as large deformations can cause significant numerical errors in the flow field and can drive the time step to zero. This can be partially alleviated by applying complex and expensive remeshing, but if the flow field tangles and inverts, the simulation can cease altogether. Arbitrary Lagrange-Eulerian (ALE) methods address the problem of a deforming Lagrangian fluid grid by permitting the fluid grid to move at some velocity other than the velocity of the fluid, but this can still lead to high aspect ratios that necessitate remeshing, especially in the presence of a fluid-structure interface. We address the challenge of coupling Eulerian fluids with Lagrangian solids by introducing an interpolation operator, which conservatively maps quantities from Eulerian boundaries to nearby Lagrangian boundary nodes, and vice versa.

At the fluid-structure interface there is a transfer of information. This information transfer can be handled by weakly coupling each separate subsystem using a one-sided estimate of the transfer, or by strongly coupling subsystems together and introducing new variables to the equations. Weakly coupled approaches have been shown to give high-fidelity results [1, 23, 22], but can struggle when applied to a system with high density-to-mass ratios (and are prone to going unstable, as we discuss in Section 3.4.3). These problems can be alleviated by using a better estimate of values at the interface, as suggested by [60], but this typically involves solving expensive general Riemann problems at every fluid-structure face. These problems can be avoided entirely by handling the interface in a strongly coupled fashion, but previous work has been limited to incompressible flows [82, 89]. Our method exploits the structure of [50], which treats the pressure flux of compressible flows implicitly. This permits us to

treat the fluid pressure as an implicit force on the solid, and use an implicit velocity boundary condition on the Poisson solve, much like previous strongly-coupled work.

Our fluid evolution is comprised of two steps: an advection stage and a pressure solver phase. This permits us to address the complexities arising from the truly non-linear components of the flow separately from the linearly degenerate components. In the pressure phase, we freeze everything to their time t^{n+1} location and perform an implicit solve for the fluid pressure and solid velocity. It is in this phase that we handle the transfer of momentum and kinetic energy across the fluid-structure interface, and as such it is important to be conservative in transferring information between the two sets of degrees of freedom. In the advection stage no information should be transmitted across the interface, but instead we must address the issues which arise by virtue of a moving solid (i.e. the covering and uncovering of fluid cells). There are many examples of how to address these problems in the literature, for example we could track cut cells, re-discretize the fluid in an ALE formulation—all of which significantly complicate the fluid evolution. Instead we make the key observation that since the interface is a contact discontinuity we can afford to be non-conservative, but only in the linearly degenerate components of the flow.

In a traditional explicit method the linearly degenerate and truly non-linear fluxes aren't separated, and as such these methods need to deal with all of the complexities of moving boundaries and information transmission at the same time. That is, they need to be conservative when dealing with information that crosses the interface while at the same time dealing with an interface that moves. Finally, the flux needs to be re-examined carefully in order to determine what forces should be applied to the interface. One could modify traditional methods by separating the conserved quantities into their Riemann invariants, and be conservative in the truly non-linear invariants while allowing the linearly degenerate invariants to be non-conservative —however this doesn't address the moving boundary, and still leaves us with the (poorly-understood) CFL restriction that arises from explicit fluid-structure interactions. Because of these complications, our method hinges on the existence of [50].

3.2 Semi-implicit compressible flow

We briefly describe the semi-implicit evolution for compressible flow [50] which forms the basis for our implicit coupling scheme. Consider the multi-dimensional Navier-Stokes equations, given by:

$$\begin{pmatrix} \rho \\ \rho\vec{u} \\ E \end{pmatrix}_t + \begin{pmatrix} \nabla \cdot \rho\vec{u} \\ \nabla \cdot (\rho\vec{u})\vec{u} \\ \nabla \cdot (E\vec{u}) \end{pmatrix} + \begin{pmatrix} 0 \\ \nabla p \\ \nabla \cdot (p\vec{u}) \end{pmatrix} = \mathbf{f} \tag{3.1}$$

where we have split the flux terms into an advection and non-advection part and lumped viscous terms into $\mathbf{f}$. The advection part (as well as any body forces) is integrated explicitly to give intermediate values $\rho^\star$, $(\rho\vec{u})^\star$ and $E^\star$. Since pressure does not affect the continuity equation, $\rho^{n+1} = \rho^\star$. The momentum update equation can be divided by ρ^{n+1} to obtain

$$\vec{u}^{n+1} = \vec{u}^\star - \Delta t \frac{\nabla p}{\rho^{n+1}}, \tag{3.2}$$

and taking its divergence gives

$$\nabla \cdot \vec{u}^{n+1} = \nabla \cdot \vec{u}^\star - \Delta t \nabla \cdot \left(\frac{\nabla p}{\rho^{n+1}} \right). \tag{3.3}$$

In the case of incompressible flow, we would set $\nabla \cdot \vec{u}^{n+1} = 0$, but for compressible flow we instead use the pressure evolution equation (see e.g. [24]),

$$p_t + \vec{u} \cdot \nabla p = -\rho c^2 \nabla \cdot \vec{u} \tag{3.4}$$

If we fix $\nabla \cdot \vec{u}$ to be at time t^{n+1} through the time step (making an $\mathcal{O}(\Delta t)$ error), discretize $p_t + \vec{u} \cdot \nabla p$ explicitly using a forward Euler time step (i.e. $\frac{p^{n+1} - p^n}{\Delta t} + \vec{u}^n \cdot \nabla p^n$), and define the advected pressure as $p^a = p^n - \Delta t(\vec{u}^n \cdot \nabla p^n)$ we obtain

$$p^{n+1} = p^a - \Delta t \rho c^2 \nabla \cdot \vec{u}^{n+1}. \tag{3.5}$$

Substituting this in Equation (3.3) and rearranging gives

$$p^{n+1} - \rho^n (c^2)^n \Delta t^2 \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right) = p^a - \rho^n (c^2)^n \Delta t \nabla \cdot \vec{u}^\star, \tag{3.6}$$

where we have defined ρc^2 at time t^n and the pressure p at time t^{n+1}. Discretizing the gradient and divergence operators yields

$$\left[I + \rho^n (c^2)^n \Delta t^2 G^T \left(\frac{1}{\hat{\rho}^{n+1}} G \right) \right] p^{n+1} = p^a + \rho^n (c^2)^n \Delta t G^T \vec{\hat{u}}^\star, \tag{3.7}$$

where G is our discretized gradient operator, $-G^T$ is our discretized divergence operator, and $\hat{\rho}$ and $\hat{u}$ represent variables interpolated to cell faces. This is solved to obtain p^{n+1} at cell centers. The 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 (thereby giving correct shock speeds). This is done by averaging the pressures to cell faces by $p^{n+1}_{i+1/2} = \frac{p^{n+1}_{i+1}\rho^{n+1}_i + p^{n+1}_i \rho^{n+1}_{i+1}}{\rho^{n+1}_i + \rho^{n+1}_{i+1}}$, rewriting Equation (3.2) using face-averaged quantities $\hat{u}_{i+1/2} = \hat{u}^\star_{i+1/2} - \Delta t \frac{G_{i+1/2} p^{n+1}}{\hat{\rho}_{i+1/2}}$ (where $\hat{\rho}_{i+1/2} = (\rho_i + \rho_{i+1})/2$), and updating the values using

$$(\rho \vec{u})^{n+1} = (\rho \vec{u})^\star - \Delta t \left(\frac{p^{n+1}_{i+1/2} - p^{n+1}_{i-1/2}}{\Delta x} \right), \qquad E^{n+1} = E^\star - \Delta t \left(\frac{(p\hat{u})^{n+1}_{i+1/2} - (p\hat{u})^{n+1}_{i-1/2}}{\Delta x} \right). \tag{3.8}$$

3.3 Solid evolution

We give a brief treatment of solid evolution with sufficient detail to properly handle the fluid-structure interactions. A solid state is completely described by its velocity and position. We update the position and velocities in a Newmark scheme in which velocity at time $t^{n+1/2}$ is used to update the position to time t^{n+1} in a second order update. Velocity is then updated from time t^n to time t^{n+1} in a separate step. We describe below the velocity update for deformable and rigid solids. The same procedure is used twice, once with a time step of $\Delta t/2$ to obtain $V^{n+1/2}$ for position update and then with a time step of Δt for the final velocity update.

Deformable body formulation: For deformable body evolution we need to handle

both elastic and damping forces. Damping forces can impose strict time step restrictions and are thus treated implicitly. We will describe a method which treats the elastic forces explicitly and damping forces implicitly although one could also incorporate implicit elasticity. The deformable body at a given time t can be described by a vector of positions of its nodes $X_s(t)$ and a vector of velocities of its nodes $V_s(t)$. The evolution of velocities can be described by Newton's second law as

$$M_s(V_s)_t = F(X_s, V_s), \tag{3.9}$$

where M_s is the mass matrix and F is the vector of all forces acting on the solid nodes. Discretizing and computing the elastic terms explicitly and damping terms explicit in position, but implicit in velocity, i.e. $F(X_s, V_s) = F(X_s^n, V_s^{n+1})$, we obtain

$$M_s V_s^{n+1} = M_s V_s^n + \Delta t F(X_s^n, V_s^{n+1}). \tag{3.10}$$

Using a Taylor series expansion on F yields

$$M_s V_s^{n+1} = M_s V_s^n + \Delta t(F(X_s^n, V_s^n) + D(V_s^{n+1} - V_s^n)). \tag{3.11}$$

where $D = \frac{\partial F}{\partial V_s}$. $F(X_s^n, V_s^n) - DV_s^n$ represents the elastic only (and, if present, any non-linear damping terms [94]) component of the force and one can write

$$M_s V_s^{n+1} = M_s V_s^\star + \Delta t D V_s^{n+1}, \tag{3.12}$$

where $V_s^\star$ denotes the velocity vector updated explicitly with the elastic terms only.
Rigid body formulation: For a rigid body we define the generalized velocity vector as $V_s = (V_{cm}^T, \omega^T)^T$, where V_{cm} is the velocity of its center of mass and ω is its angular velocity. The velocity evolution can then be described as

$$\begin{pmatrix} M_r & 0 \\ 0 & I_r \end{pmatrix} (V_s)_t = \begin{pmatrix} f \\ \tau \end{pmatrix}, \tag{3.13}$$

where M_r is a 3×3 diagonal matrix with the rigid body mass in the diagonals, I_r

is the inertia tensor and f, τ are the net force and torque acting on it. Writing the mass matrix as M_s and combining f, τ into F, we get a form similar to (3.9) which can be discretized using forward Euler to obtain

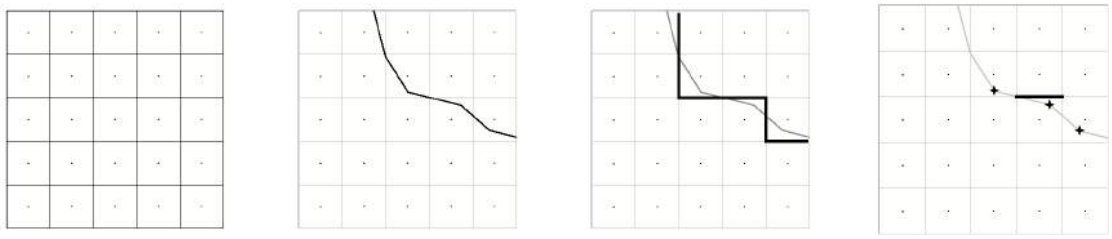
$$M_s V_s^{n+1} = M_s V_s^n + \Delta t F^n = M_s V_s^\star. \tag{3.14}$$

Where $V_s^\star$ denotes the velocity vector updated with the explicit forces. Note that this is the same as Equation (3.12) except without any damping term. We will therefore use Equation (3.12) as our general solid update equation below, as it covers both the rigid and deformable cases.

3.4 Fluid-structure interaction

We solve for the fluid on an Eulerian grid, and the solids on freely deforming Lagrangian meshes. The fluid and structure interact with each other by applying equal and opposite forces at the interface, satisfying physical boundary conditions (we use no-slip, no penetration boundary conditions) in the process. Immersed boundary methods induce extra force variables at the interface and apply a regularization operator to map these forces to fluid faces (see e.g. [97]). They also incorporate an interpolation operator to map fluid velocity to solid nodes for applying boundary conditions. We eliminate the extra interface force variables and conservatively map the fluid pressures directly to solid nodes, and solid velocities to fluid faces using an interpolation operator.

Figure 3.1 illustrates an example fluid grid which is coupled to a Lagrangian solid which occupies the upper right-hand corner of the grid. In our model, the fluid interacts with a voxelized version of the solid and the solid directly sees forces acting on its nodes. We define an interpolation operator W which maps solid node velocity to the fluid cell faces, where the rows correspond to fluid faces and the columns to solid nodes. W can be constructed in a row-by-row fashion: for each row, we identify the corresponding fluid face and locate the nearby solid nodes. The entry corresponding to each solid node is populated by a weight proportional to its contribution to the



(a) Eulerian fluid grid.

(b) Lagrangian solid which overlaps the fluid domain.

(c) Solid voxelized to fluid faces.

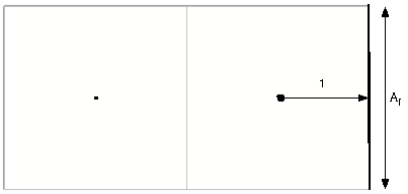
(d) Solid nodes which contribute to the rasterized face.

Figure 3.1: A common challenge with FSI problems is one of overlapping grids. We resolve this issue by voxelizing solid degrees of freedom to the fluid grid using an interpolation operator denoted by the matrix W. The row corresponding to a fluid face gets contributions from nearby solid nodes.

fluid face, and then finally the row is normalized to ensure that each row sums to one, making it an interpolation. This is done in a component-by-component manner, e.g. the x-component of solid velocity is voxelized to x-axis fluid faces but not y- or z-axis fluid faces, and so the solid velocity at fluid face $i + 1/2$ is $(WV_s)_{i+1/2}$. Since pressure is defined at cell centers, we also introduce an extrapolation operator B which maps cell-centered pressure to face pressures, as illustrated in Figure 3.2. These face pressures are then multiplied by the surface area of the cell face to get a force and distributed back to solid nodes using W^T. That is, W maps from solid node degrees of freedom to cell faces, and W^T maps back in the opposite direction. Note that since the rows of W sum to one, the columns of W^T sum to one and therefore the force felt due to the pressure on the face is fully and conservatively distributed to the solid node degrees of freedom.

3.4.1 The strongly coupled system

The fluid acts on solid degrees of freedom via pressure along the interface. The pressure exerts a force given by $W^T A_f B p$ on the solid degrees of freedom, where A_f is a diagonal matrix whose entries correspond to the areas of fluid-structure faces. We



$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Figure 3.2: Operator B maps pressure from cell centers to bordering fluid-structure faces. In this example there are x-direction faces, of which the one to the far right represents a rasterized solid face. Therefore B has three rows (one for each vertical face, with the top and the bottom rows corresponding to the far left and far right vertical faces respectively, and the middle row corresponding to the middle vertical face), and two columns (one for each pressure at each cell center). Since the only contribution to the solid is from the second pressure to the third face, B has the form shown above with a single non-zero element. Note that $(1/dx)B^T$ equals $-G_s^T$, as defined in Figure 3.3(b).

can incorporate these forces into the implicit solid system given by Equation (3.12):

$$M_s V_s^{n+1} = M_s V_s^\star + \Delta t D V_s^{n+1} + \Delta t W^T A_f B p. \tag{3.15}$$

The fluid sees a velocity boundary condition at the fluid-structure interface. To incorporate this into the fluid equations, we partition the discrete divergence operator $-G^T$ into two components. G_f^T operates over fluid-fluid faces, while G_s^T is the component of the divergence operator which operates on rasterized fluid-structure faces (as outlined in Figure 3.3), and $G^T = G_f^T + G_s^T$. We can then set fluid-structure faces to have implicit Neumann boundary conditions; that is,

$$\vec{u}^{n+1} = \begin{cases} \vec{u}^\star - \Delta t \frac{G_f p}{\hat{\rho}} & \text{at a fluid-fluid face; and} \\ W V_s^{n+1} & \text{at a fluid-structure face.} \end{cases} \tag{3.16}$$

Taking the divergence of the velocity field yields

$$G^T \vec{u}^{n+1} = G_f^T \vec{u}^\star - \Delta t G_f^T \frac{1}{\hat{\rho}} G p + G_s^T W V_s^{n+1} \tag{3.17}$$

Using this modified definition for $G^T\vec{u}^{n+1}$ in Equation (3.5) and substituting into Equation (2.5) gives

$$\left[\frac{1}{\Delta t\rho^n(c^n)^2}I + \Delta t G_f^T\frac{G}{\hat{\rho}^{n+1}}\right]p^{n+1} - G_s^T W V_s^{n+1} = \frac{p^a}{\Delta t\rho c^2} + G_f^T\vec{u}^\star. \tag{3.18}$$

If we define $V = \Delta x\Delta y\Delta z$ to be the volume of the fluid cell, then $VG_s^T = A_f B^T$. Combining equations (3.15) and (3.18), using scaled pressure $\tilde{p} = \Delta t p$ and scaled advected pressure $\tilde{p}^a = \Delta t p^a$, and rescaling the fluid equations by cell volume gives us our symmetric system

$$\begin{pmatrix} \frac{V}{\Delta t^2\rho c^2}I + VG_f^T\frac{1}{\rho}G_f & -A_f B^T W \\ -W^T B A_f & -M_s + \Delta t D \end{pmatrix}\begin{pmatrix}\tilde{p}^{n+1}\\ V_s^{n+1}\end{pmatrix} = \begin{pmatrix}\frac{V}{\Delta t^2\rho c^2}\tilde{p}^a + VG_f^T\vec{u}^\star \\ -M_s V_s^\star\end{pmatrix}. \tag{3.19}$$

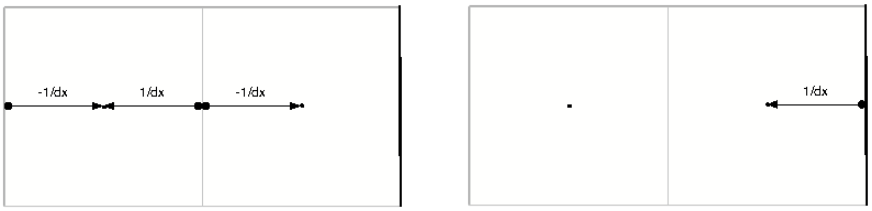
It is interesting to note that if we take the incompressibility assumption (i.e. $c\to\infty$) then this system reduces to one similar to [89].

The system in Equation (3.19) is symmetric but indefinite, and can be solved using efficient solvers such as Conjugate Residuals [72] to obtain the final time t^{n+1} solid velocity and pressure. The solid part of our update is now complete, but we still need to use the t^{n+1} pressure to update the fluid momentum and energy (noting that $\rho^{n+1} = \rho^\star$ is already done).

3.4.2 Updating fluid momentum and energy

To obtain correct shock speeds we use the flux-based method discussed above, with modifications to account for fluid-structure faces. At a fluid-structure face $i+1/2$, the fluid applied a force of $(BA_f p)_{i+1/2}$ to the solid. To conserve momentum, fluid face $i+1/2$ should apply an equal and opposite force $-(BA_f p)_{i+1/2}$ on fluid cell i. In our momentum update this is numerically equivalent to setting $p_{i+1/2} = (Bp)_{i+1/2}$ at fluid-structure faces.

Next, we need to consider the work done by the fluid on the solid at a fluid-structure face. We are applying an impulse $\Delta t(BA_f p)_{i+1/2}$ on the solid, which is equivalent to applying a constant force over the interval Δt. In order to compute the



$$G_f^T = \frac{1}{dx}\begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix} \qquad G_s^T = \frac{1}{dx}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 3.3: In our derivation, the divergence operator $-G^T$ is split into G_f^T (which operates only on fluid-fluid faces) and G_s^T (which operates only on fluid-structure faces). We show this splitting for a simple two cell example where the right-most face is a fluid-structure interface. The rows in the above matrices correspond to cells and columns to faces. The left most face corresponds to the first column of G_f^T and only has one non-zero element since it only borders one fluid cell. The middle face (which corresponds to the second column of G_f^T) contributes to both fluid cells and hence has two non-zero elements. The third column of G_f^T is zero, as the third face is a fluid-structure face and instead corresponds to G_s^T. Figure (b) depicts G_s^T, which is defined as $-(1/dx)B^T$ in Figure 3.2.

work done on the solid system by a single force $\vec{f}$ in the presence of other forces, we lump all forces acting on the solid into a vector $\vec{F}$ and examine

$$\int_0^{\Delta t} \vec{f}\cdot V_s(t)dt = \int_0^{\Delta t} \vec{f}\cdot(V_s^n + M_s^{-1}\vec{F}t)dt = \Delta t\vec{f}\cdot\left[V_s^n + M_s^{-1}\vec{F}\frac{\Delta t}{2}\right] = \Delta t\vec{f}\cdot\left[\frac{V_s^n + V_s^{n+1}}{2}\right], \tag{3.20}$$

where we take advantage of $\vec{F}$ and $\vec{f}$ being constant over the interval. We are interested in calculating the work done by a single fluid face on the solid, so if we take $W_{i+1/2}^T$ to be the column vector which distributes the pressure from cell face $i+1/2$ to the solid node degrees of freedom then $\vec{f} = W_{i+1/2}^T(BA_f p)_{i+1/2}$, and the work done on the solid by this face is exactly

$$\Delta t\left[W_{i+1/2}^T(BA_f p)_{i+1/2}\right]^T\left[\frac{V_s^n + V_s^{n+1}}{2}\right] = \Delta t\left[(BA_f p)_{i+1/2}\right]W_{i+1/2}\left[\frac{V_s^n + V_s^{n+1}}{2}\right]. \tag{3.21}$$

This, if $p_{i+1/2}$ is defined to be $(Bp)_{i+1/2}$ as suggested above in the momentum update, then we merely need to set $\vec{u}_{i+1/2} = (1/2)(W[V_s^n + V_s^{n+1}])_{i+1/2}$ in order to obtain a

flux $p\vec{u}$ which exactly conserves the kinetic energy transferred.

3.4.3 Time step restriction

In our method fluid-structure interactions are handled implicitly and thus we avoid introducing any new time step restrictions. The time step is therefore determined by the minimum of the time steps imposed by the fluid and the structure. For the structure update the time step restriction is determined by the elastic part only, as damping terms are handled implicitly, while our semi-implicit fluid update imposes a time step restriction dependent only on its bulk velocity. The time step restriction imposed by the semi-implicit flow formulation in two spatial dimensions is

$$\frac{\Delta t}{2}\left(\frac{|u|_{max}}{\Delta x}+\frac{|v|_{max}}{\Delta y}+\sqrt{\left(\frac{|u|_{max}}{\Delta x}+\frac{|v|_{max}}{\Delta y}\right)^2+4\frac{|p_x|}{\rho\Delta x}+4\frac{|p_y|}{\rho\Delta y}}\right)\leq 1, \qquad (3.22)$$

and we refer the interested reader to [50], which motivates this formulation.

We note that the implicit fluid-structure coupling gives stable results even for very high density-to-mass ratios, where explicit methods struggle even when the CFL restrictions of both solid and fluid systems are obeyed. We explore this in example 3.6.1.

3.5 Unified time integration

We employ a time integration scheme which incorporates fluid evolution into a Newmark-style solid evolution scheme. The scheme works by computing an intermediate velocity for the solid $V_s^{n+1/2}$, and applying this in a second order update to get solid positions at time t^{n+1}. Velocities are then updated from time t^n to t^{n+1} (discarding intermediate values), and so two linear systems are solved.

In order to compute the intermediate solid velocity $V_s^{n+1/2}$, we begin by applying all explicit solid forces to the system, which gives $V_s^{n+1/2\star}$. Explicit body forces such as gravity and viscosity are also applied to the fluid system, yielding $t^{n+1/2\star}$ fluid quantities. The coupled system (3.19) is solved in order to obtain $X_s^{n+1} =$

$X_s^n + \Delta t V_s^{n+1/2}$, and then the entire fluid state and all solid velocities are restored to their time t^n values.

These new positions are then used to compute an *effective* velocity for the solids, i.e. $(X_s^{n+1} - X_s^n)/\Delta t$. Using the effective velocity and then the time t^n position of the solid, we fill ghost cells. These ghost cells are used directly in the stencils of high-order methods, and provide a valid state for which to populate uncovered cells. In order to compute the ghost cell data at location $\vec{x}_g$, we begin by identifying the closest solid interface point $\vec{x}_I$, and reflecting across the interface. Density and pressure are interpolated to the reflected point $2\vec{x}_I - \vec{x}_g$ from neighboring cells and then copied to the ghost cell. The surface normal $\vec{N}$ at the interface is used to decompose the velocity at the reflected point $\vec{V}_r$ into its normal component $V_{rN} = \vec{V}_r \cdot \vec{N}$ and its tangential component $\vec{V}_{rT} = \vec{V}_r - V_{rN}\vec{N}$. In order to remain continuous with the effective velocity of the structure at the interface $\vec{V}_I$, V_{rN} is reflected across the interface, and so we compute $V_{gN} = 2\vec{V}_I \cdot \vec{N} - V_{rN}$. Tangential velocity is decoupled from the interface and thus we can use it directly, giving the final ghost cell velocity $\vec{V}_g = V_{gN}\vec{N} + \vec{V}_{rT}$.

Once ghost cells are filled, explicit body forces such as gravity and viscosity are integrated into the system, and the advection component of flux from Equation (3.1) is applied using a conservative flux-based method (see [50]. Explicit solid forces are applied in order to compute $V_s^{n+1\star}$, and then the coupled system (3.19) is solved to obtain V_s^{n+1} and p^{n+1}. This pressure is applied as per Section 3.4.2 to obtain time t^{n+1} fluid quantities.

We also fill the ghost cells inside the solid using time t^{n+1} data from the fluid and solid velocities, as described above. Although none of our examples use these ghost values, if an explicit body force such as viscosity were to be applied, its stencil would require valid ghost cells to be defined. Note that these are valid as instantaneous ghost cells, whereas the ghost cells above use the *effective* solid velocity, which is the actual motion of the solid through the mesh. Practical experience shows that this can make a meaningful difference.

3.6 Examples and validation

In order to compare our results with previous methods, we implement an explicit coupling scheme which integrates a fully explicit compressible flow evolution with a Newmark time integration for solids. This explicit method proceeds in a fashion similar to Section 3.5, except that instead of solving the system (3.19) we simply fill ghost cells inside the solid once and explicitly evolve the fluid once, while time t^n pressures along the fluid-structure interface are applied to the solid as explicit forces. This gives us an explicitly coupled time evolution scheme, such as the one described in [22].

Although one might assume that the implicit solve would cause efficiency bottlenecks, we observed relatively few Conjugate Residuals iterations per time step. This is likely due to the strongly diagonally dominant nature of Equation (3.19), and the good initial guess for pressure provided by the equation of state at time t^n. For all of our one dimensional examples the maximum number of iterations required per time step was 3. For the two dimensional examples, the rigid body coupling example required a maximum of 4 iterations, while the deformable coupling example required a up to 24 iterations per time step.

In all of the examples we consider the fluid is simulated using an ideal gas law, with $\gamma = 1.4$.

3.6.1 One-dimensional validation

We examine several one dimensional fluid-structure interactions to validate our method. A third order ENO scheme [93] is used along with an advection-based CFL number of .6. All quantities below are in SI units, with density as kg/m^3, pressure in Pa, lengths in m, spring coefficients in N/m, etc.

Sod shock coupled with a rigid body

Our first example is a Sod shock interacting with a rigid body, with open boundary conditions. The initial condition for the fluid is

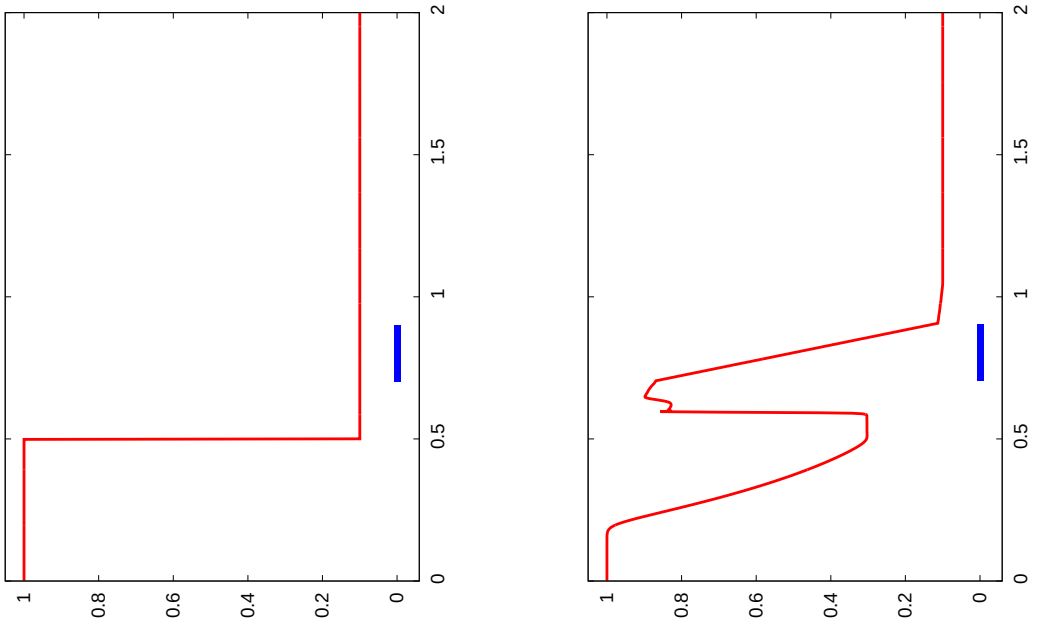
$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (1,0,1) & \text{if } x \leq .5, \\ (.125,0,.1) & \text{if } x > .5. \end{cases}$$

A rigid body of mass 1 and width .2 starts at rest with its center of mass a distance of .8 from the left of the domain. The domain is of length 2. The rigid body remains at rest until the shock hits it, at which point it accelerates by virtue of the pressure difference. The solid body continues to accelerate until it converges to a velocity of .927453, which is precisely the interfacial velocity of the Sod Riemann problem. Figure 3.4 shows snapshots of the pressure profile at various times through the simulation. For comparison, results with the explicit method are shown in Figure 3.5. We also do a convergence analysis of our method in Figure 3.6. The error in the position of the rigid body is computed at time .9 from the highest resolution grid simulated, which is 6401 grid cells. The convergence order of the error is estimated as 1.6.

It is interesting to consider this simple problem for a variety of density-to-mass ratios. Figure 3.7(a) shows the velocity of the rigid body as a function of time for a range of rigid body masses in the semi-implicit case. Figure 3.7(b) shows this in the explicit case. We note that the explicit simulation struggles with high density-mass ratios. In particular it appears as though the rigid body gains too much momentum in a single time step, causing the fluid on the other side to over-compress, leading to a very stiff oscillatory system, even though the time step obeyed CFL restrictions. We show snapshots of the pressure profile of simulations with a light solid of mass .0001, with semi-implicit and explicit schemes in Figure 3.8 and Figure 3.9, respectively.

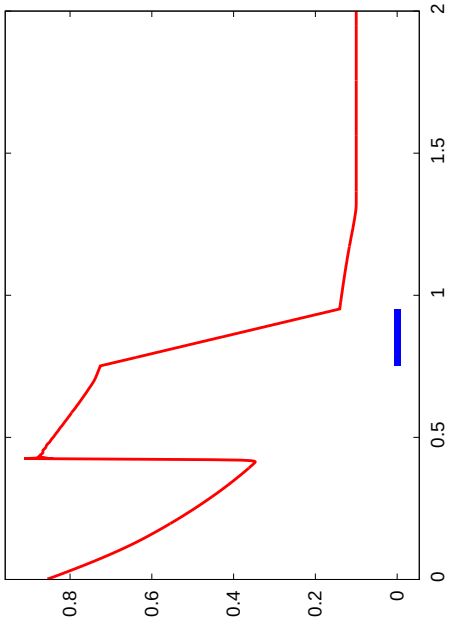
Sod shock interacting with a fluid piston

We consider a similar problem, this time with solid wall boundary conditions and a larger domain, with the initial discontinuity located at distance 1 from the left of the domain. The rigid body has a mass of 1, width .2 and starts at rest with its center

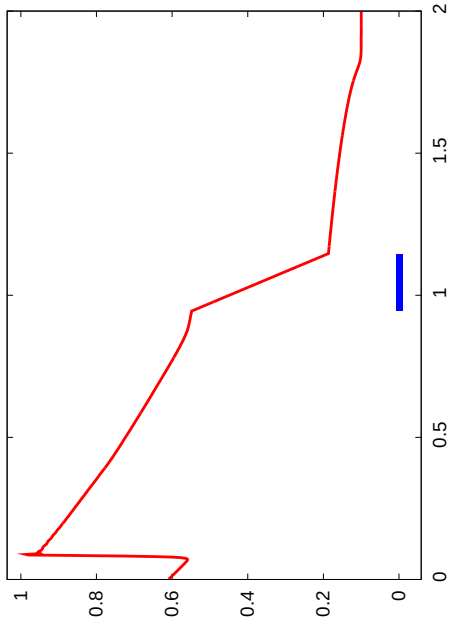


(a) $t = 0$

(b) $t = .25$

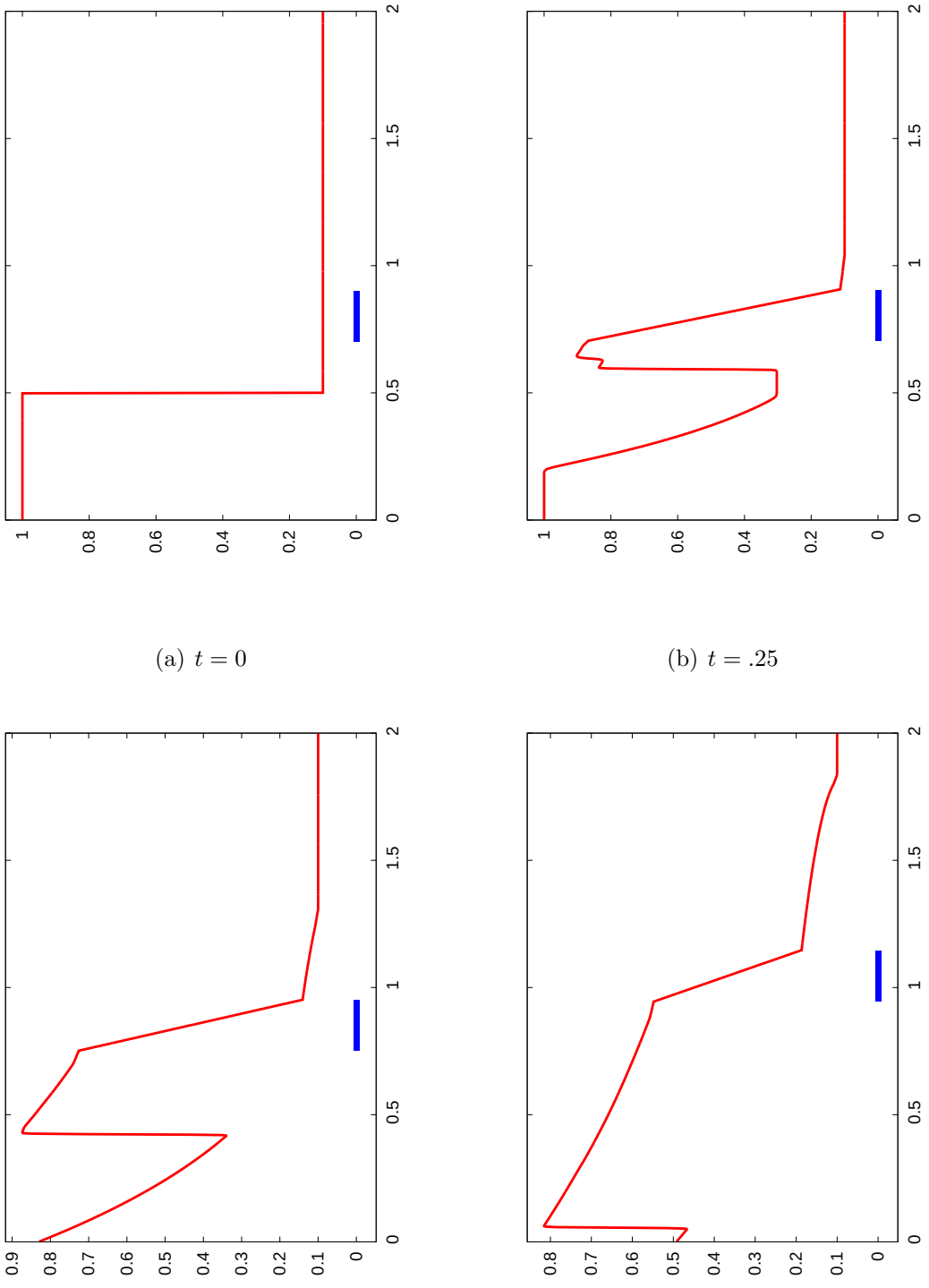


(c) $t = .5$



(d) $t = 1$

Figure 3.4: Semi-implicit simulation of a Sod shock hitting a rigid body of mass 1. Pressure profile of the fluid is shown at various times through the simulation. The 1-D rigid body is drawn as a blue line segment at the bottom of the plot, with pressure



(a) $t = 0$

(b) $t = .25$

(c) $t = .5$

(d) $t = 1$

Figure 3.5: Explicit simulation of a Sod shock hitting a rigid body of mass 1. Pressure profile of the fluid is shown at various times through the simulation. The 1-D rigid body is drawn as a blue line segment at the bottom of the plot, with pressure inside

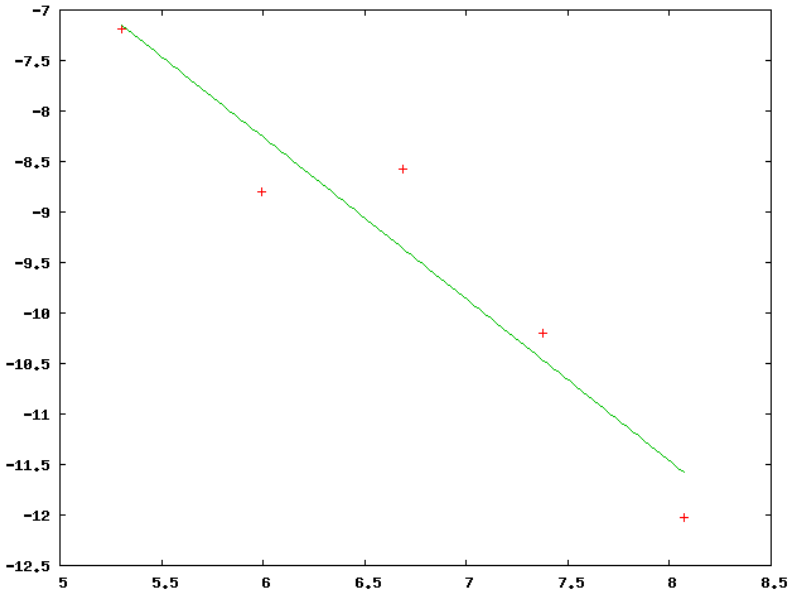
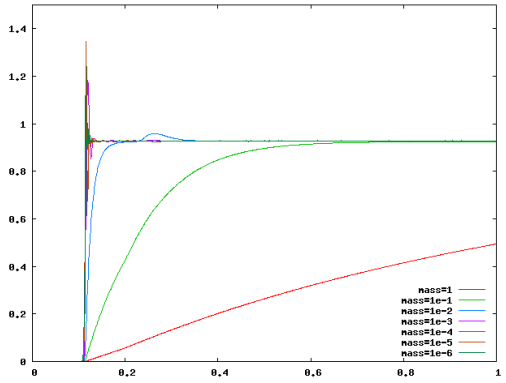


Figure 3.6: Position error of the center of mass of a rigid body hit by a Sod shock, as compared to a high-resolution simulation, at time $.9s$. We plot the log of the relative error, as a function of the log of the resolution of the underlying grid. The convergence rate is 1.6.

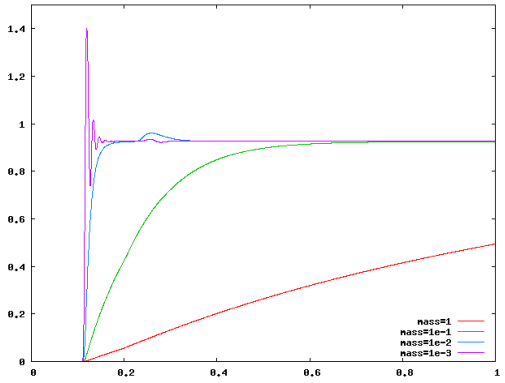
of mass at 1.5 from the left of the domain. The domain is of length 3. The shock imparts momentum to the rigid body which in turn compresses the fluid on its right. This compressed fluid creates a high pressure region which pushes back on the solid, in effect creating a "fluid spring." This causes the rigid body to oscillate, which plots the position of the center of mass of the rigid solid as a function of time. Figure 3.10 shows snapshots of the pressure profile at various times through the simulation. For comparison, results with the explicit method are shown in Figure 3.11. We also do a convergence analysis of our method in Figure 3.12. The error in the position of the rigid body is computed at time $4s$ from the highest resolution grid simulated, which is 6401 grid cells. The convergence order of the error is estimated as 1.03.

Sod shock coupled with a mass-spring system

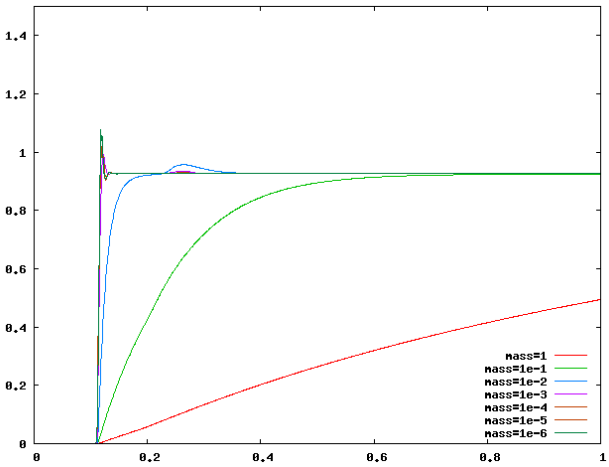
To conclude the one-dimensional examples, we consider the mass-spring system interacting with a high pressure gas described in [1] in order to provide validation for



(a) Semi-Implicit.

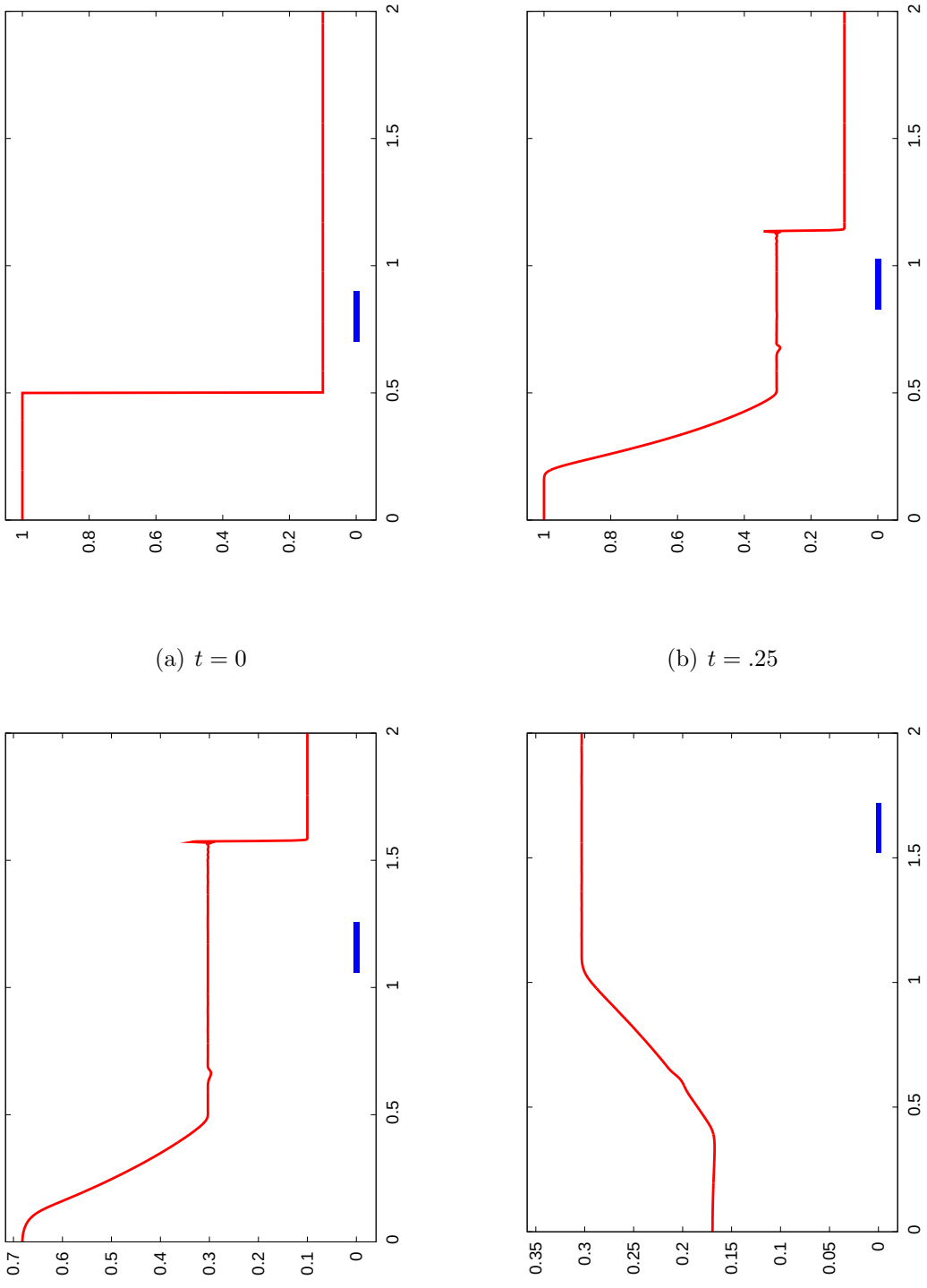


(b) Explicit.



(c) Semi-Implicit symmetric positive-definite formulation.

Figure 3.7: Velocity of a 1-D rigid body hit by a Sod shock, as a function of time. Simulations were done on a grid of resolution 1601. All simulations were run with a CFL number of .6, where the explicit simulation CFL is based on $|u| \pm c$ and the semi-implicit simulation was run with the CFL condition specified in Equation (3.22). The explicit simulations grow increasingly unstable as mass tends to zero, giving unusable results when mass reaches .0001 (these results are shown in Figure 3.9), and crashes for lighter masses. As mass tends to zero, the momentum absorbed by the solid tends to zero and the shock passes through the solid relatively unperturbed, and so the flat line to which solid velocities appear to converge is in fact the post-shock velocity of the fluid.



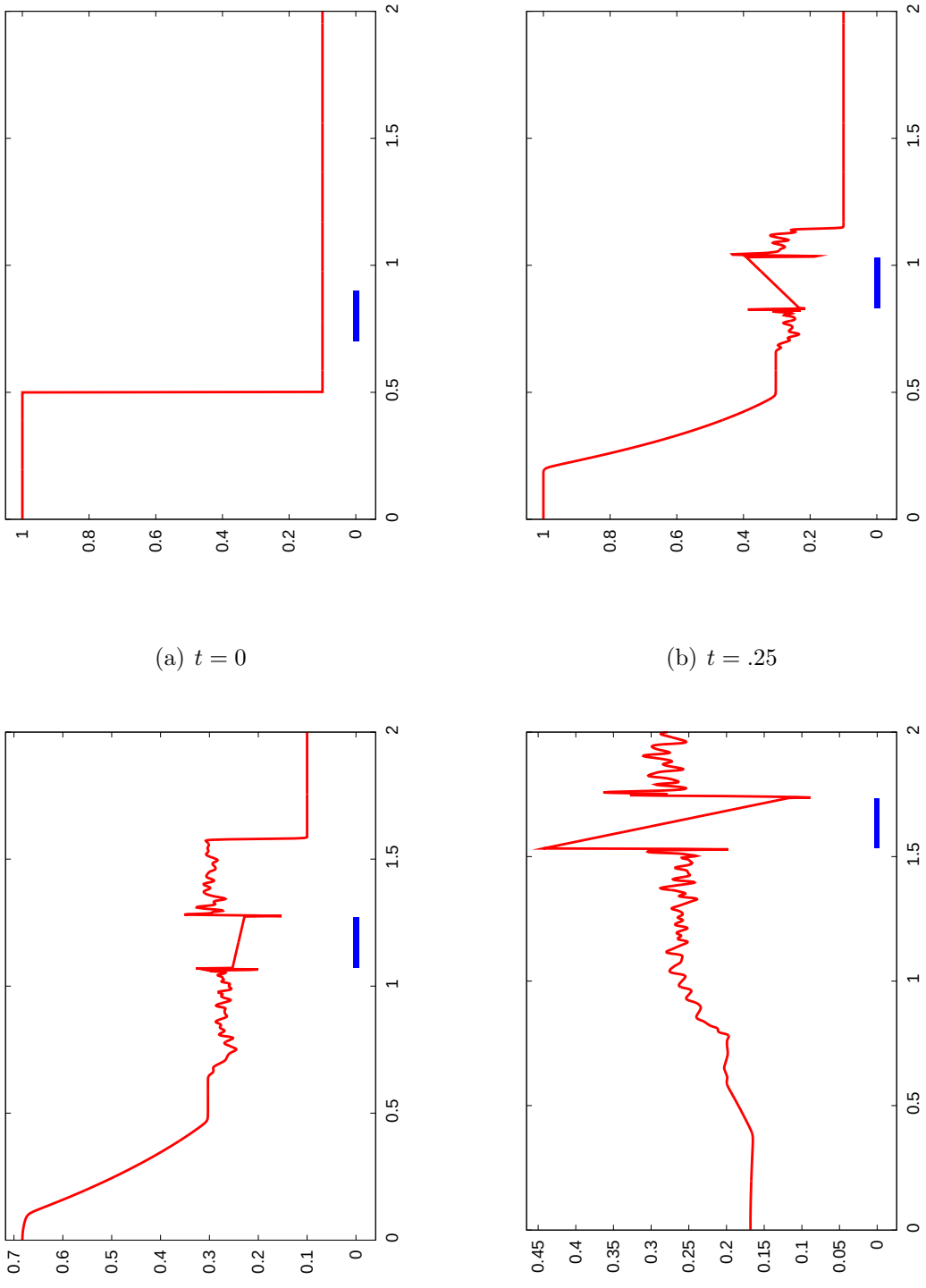
(a) $t = 0$

(b) $t = .25$

(c) $t = .5$

(d) $t = 1$

Figure 3.8: Semi-implicit simulation of a Sod shock hitting a light solid of mass .0001. Pressure profile of the fluid is shown at various times through the simulation. The 1-D rigid body is drawn as a blue line segment at the bottom of the plot. The



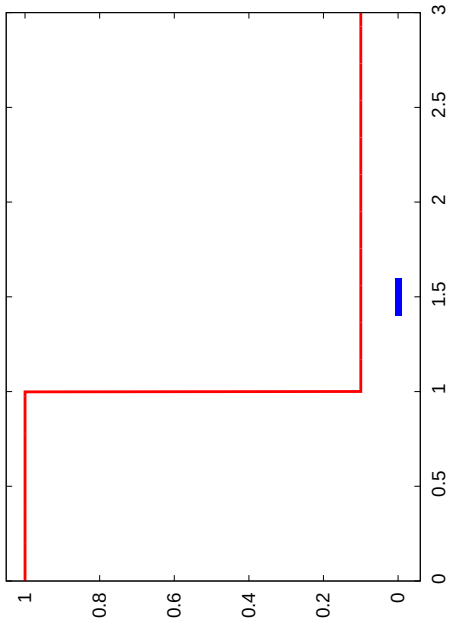
(a) $t = 0$

(b) $t = .25$

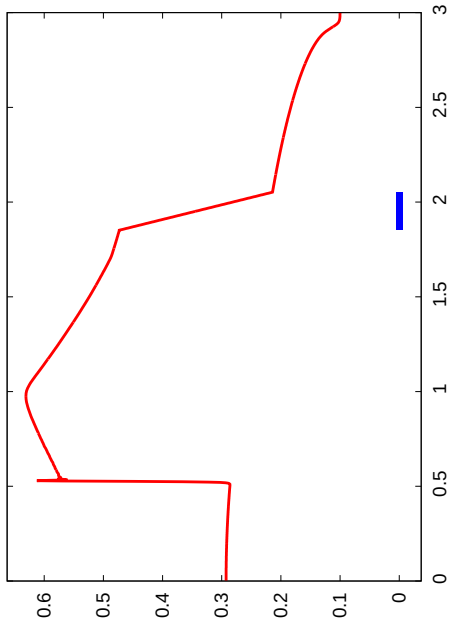
(c) $t = .5$

(d) $t = 1$

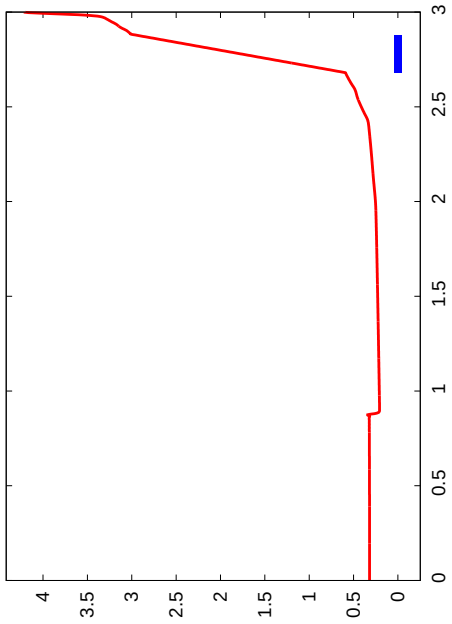
Figure 3.9: Explicit simulation of a Sod shock hitting a light solid of mass .0001. Pressure profile of the fluid is shown at various times through the simulation. The 1-D rigid body is drawn as a blue line segment at the bottom of the plot. The simulation



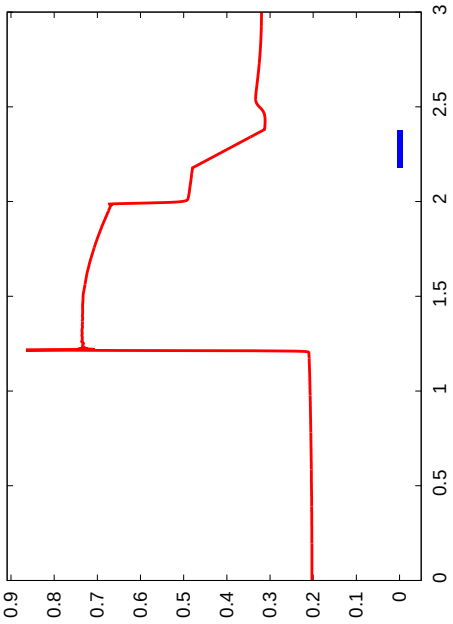
(a) $t = 0$



(b) $t = 1.5$

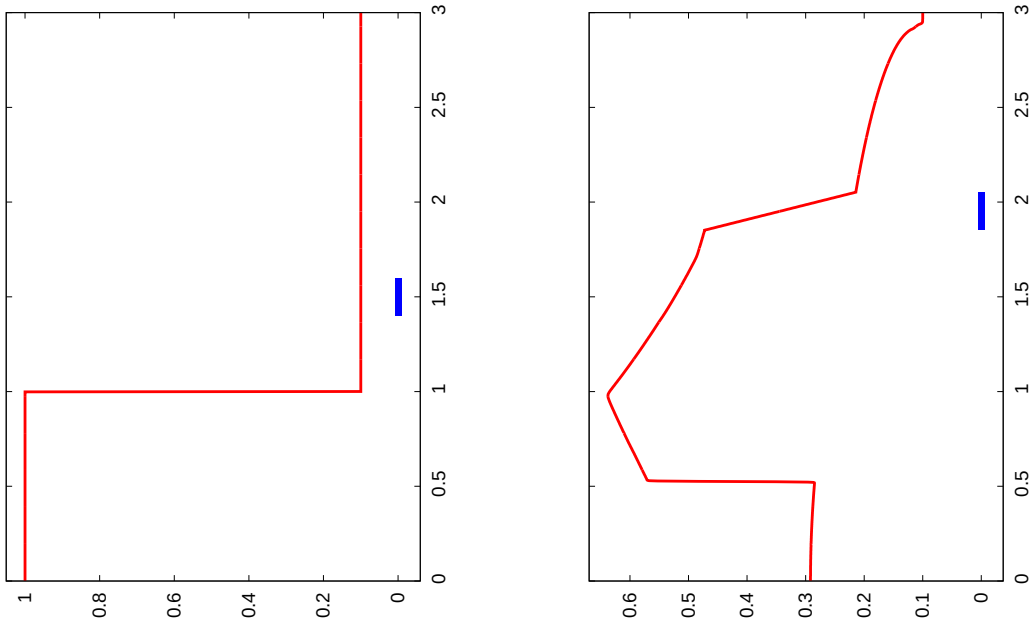


(c) $t = 2.87$



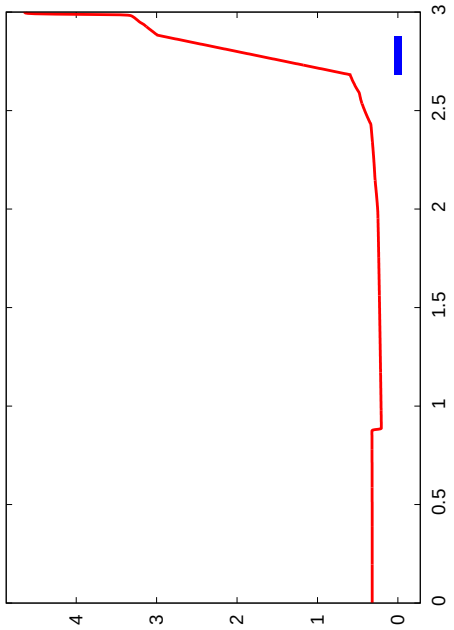
(d) $t = 4$

Figure 3.10: Semi-implicit simulation of a piston hit by a Sod shock, with closed-wall boundary conditions on both sides. Pressure profile of the fluid is shown at various times through the semi-implicit simulation. The 1-D rigid body is drawn as a blue

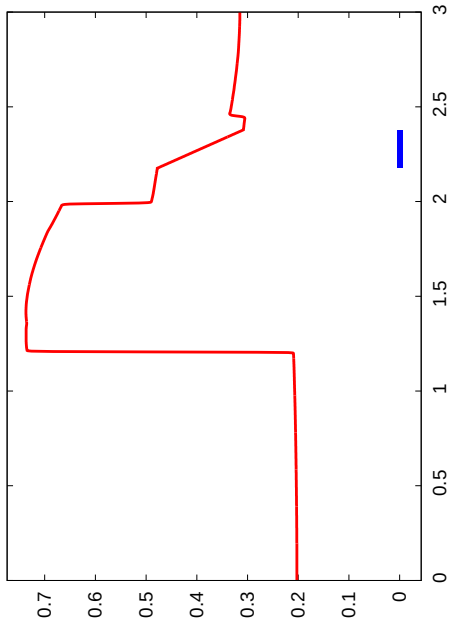


(a) $t = 0$

(b) $t = 1.5$



(c) $t = 2.87$



(d) $t = 4$

Figure 3.11: Explicit simulation of a piston hit by a Sod shock, with closed-wall boundary conditions on both sides. Pressure profile of the fluid is shown at various times through the explicit simulation. The 1-D rigid body is drawn as a blue line

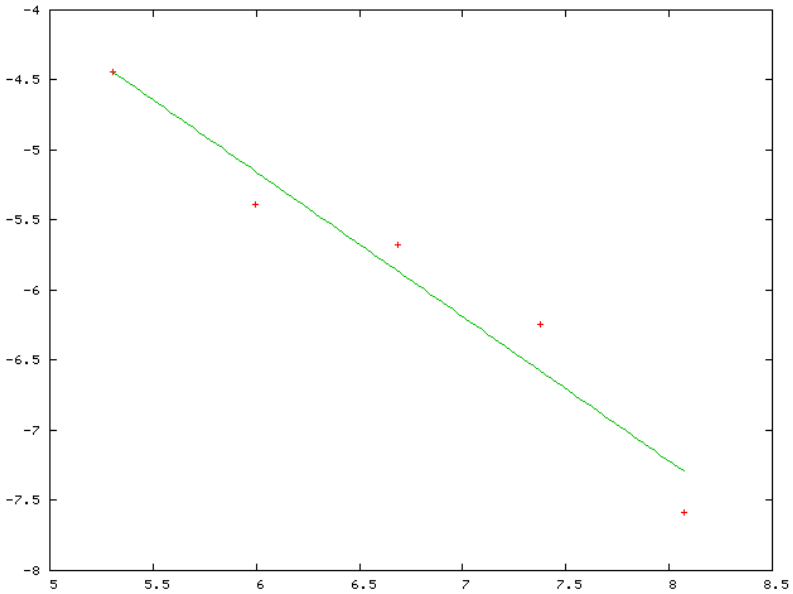
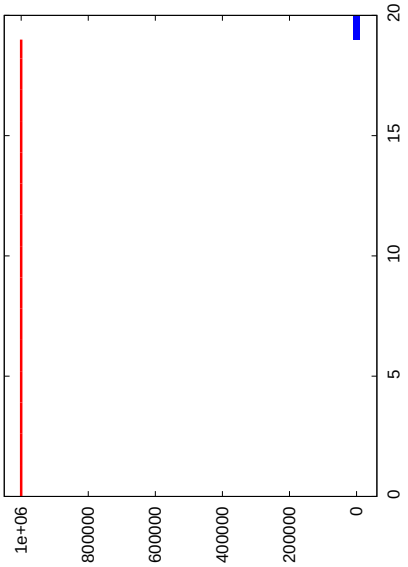


Figure 3.12: Position error of the center of mass of the piston (Section 3.6.1), as compared to a high-resolution simulation, at time $4s$. We plot the log of the relative error, as a function of the log of the resolution of the underlying grid. The convergence rate is 1.03.

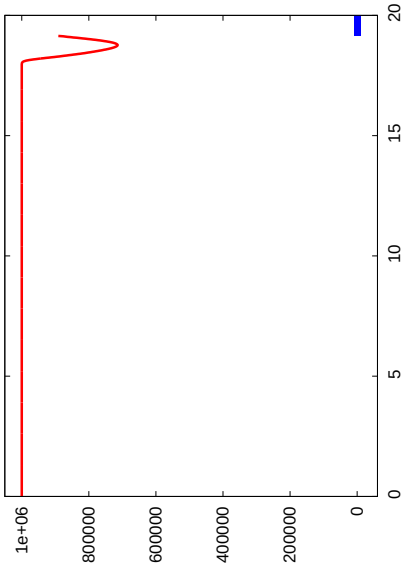
our approach against an analytic solution. The domain is of length 20, and a spring is fixed to the right side of the domain which has a rest length of 1, a stiffness of 10^7, no damping and a mass of 3. The fluid is given by

$$(\rho, p, \vec{u}) = (4, 10^6, 0)$$

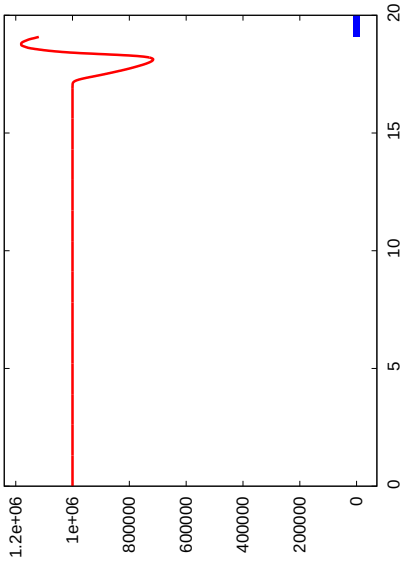
An outflow boundary condition is used for the left side of the domain. The spring starts at rest length and is compressed by the gas. Figure 3.13 shows snapshots of the pressure profile at various times through the simulation. The position of the moving end of the spring as a function of time is shown in Figure 3.14(a), and a convergence analysis in Figure 3.14(b). The error in the position of the free end of the spring is computed at time .008, and is compared against the analytic solution provided in [1]. The convergence order of the error is estimated as 1.16.



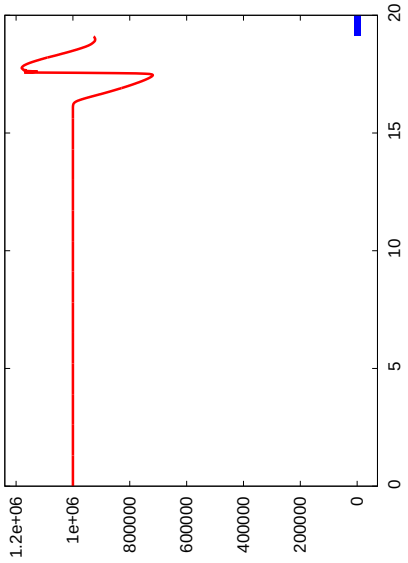
(a) $t = 0$



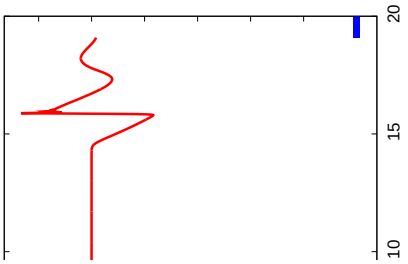
(b) $t = .0015$

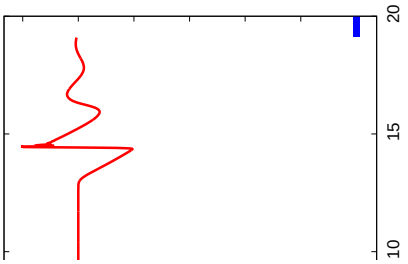


(c) $t = .003$



(d) $t = .0045$





3.6.2 Two-dimensional validation

In this section we validate our method for the multidimensional case, and briefly describe a symmetric positive-definite reformulation of the Equation (3.19). We consider interactions with both rigid and deformable solids. A second order ENO scheme was used along with an advection-based CFL number of .6.

Rigid Cylinder lift-off

This example, which is suggested by [19, 27, 1], examines the interaction of a Mach 3 shock with a rigid cylinder initially at rest on the floor of a rectangular channel. The cylinder is lifted by the shock, due to an asymmetric reflection of the incident wave. The test domain is $1 \times .2$, with the initial shock front positioned at .08 from the left boundary and the remaining domain is filled with the gas at pressure 1 and density 1.4. The top and bottom of the domains are rigid walls, the left boundary is fixed to be the post shock state and an outflow boundary condition is used for the right boundary. The cylinder has a density of 10.77, a radius of .05 and is initially located at $(.15, .05)$. Figure 3.15 shows the snapshot of the simulation for a selection of times. Our results compare favorably to those shown in [1], and converges at a rate of .93.

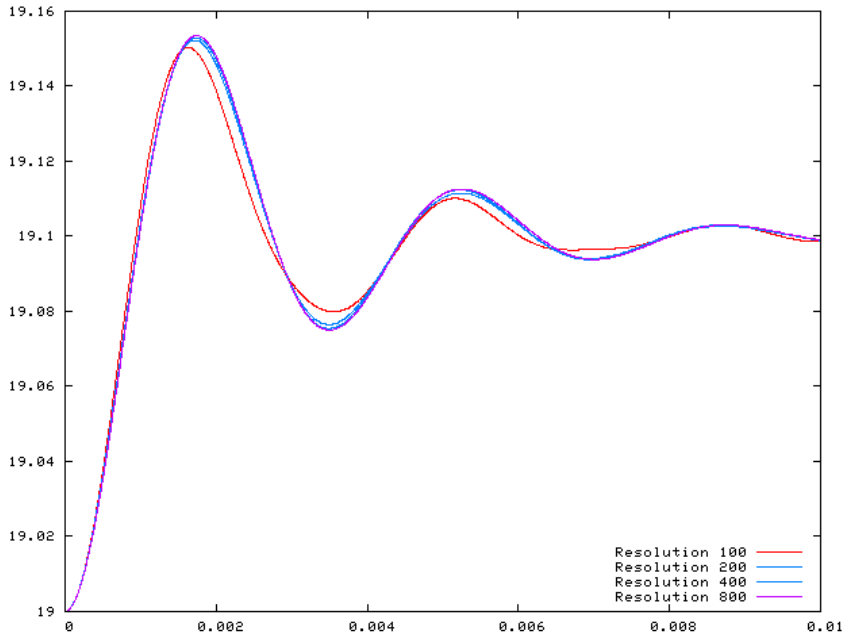
Deforming cylinder lift-off

This example is similar to the one described above (in Section 3.6.2), except that the rigid cylinder is replaced by a deformable mass-spring system with 222 triangles, and edge- and altitude-springs with a stiffness of .3. The density of the sphere is 10.77, has a radius of .05 and the center of mass is initially located at $(.15, .05)$. Figure 3.16 shows snapshots of the simulation for a selection of times. As the shock front passes through the deforming body, it dissipates, scatters and is partially absorbed by the body. The example converges at a rate of .99.

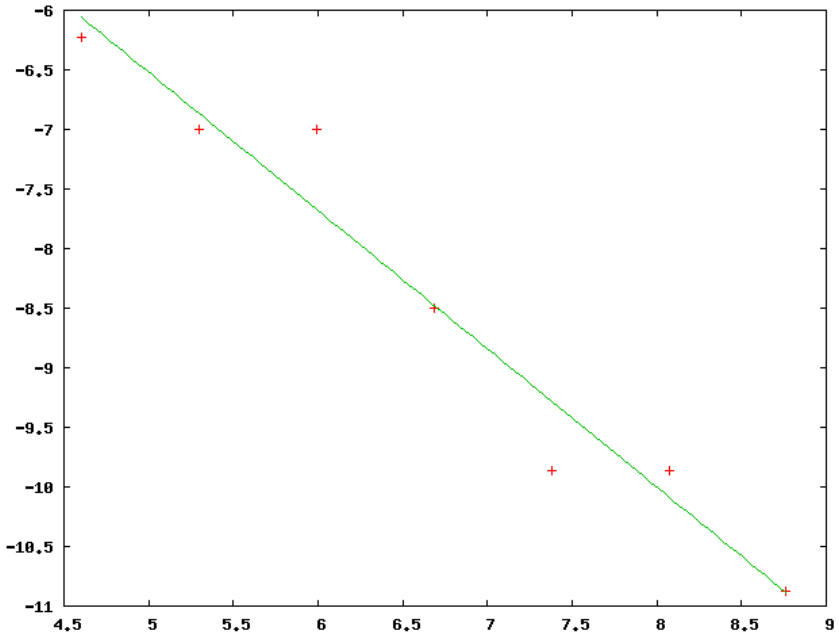
Heavy deforming cylinder lift-off

We next consider a heavy deforming cylinder, in the same setup as described in Section 3.6.2 and Section 3.6.2 above. In this case, the cylinder matches the cylinder

from Section 3.6.2, except the density is set to 100. As the body absorbs the shock wave, it compresses and delays the shock. Some of the shock is reflected, but most of the shock passes through the cylinder. Figure 3.17 shows snapshots of the simulation at a selection of times. The example converges at a rate of 1.01.

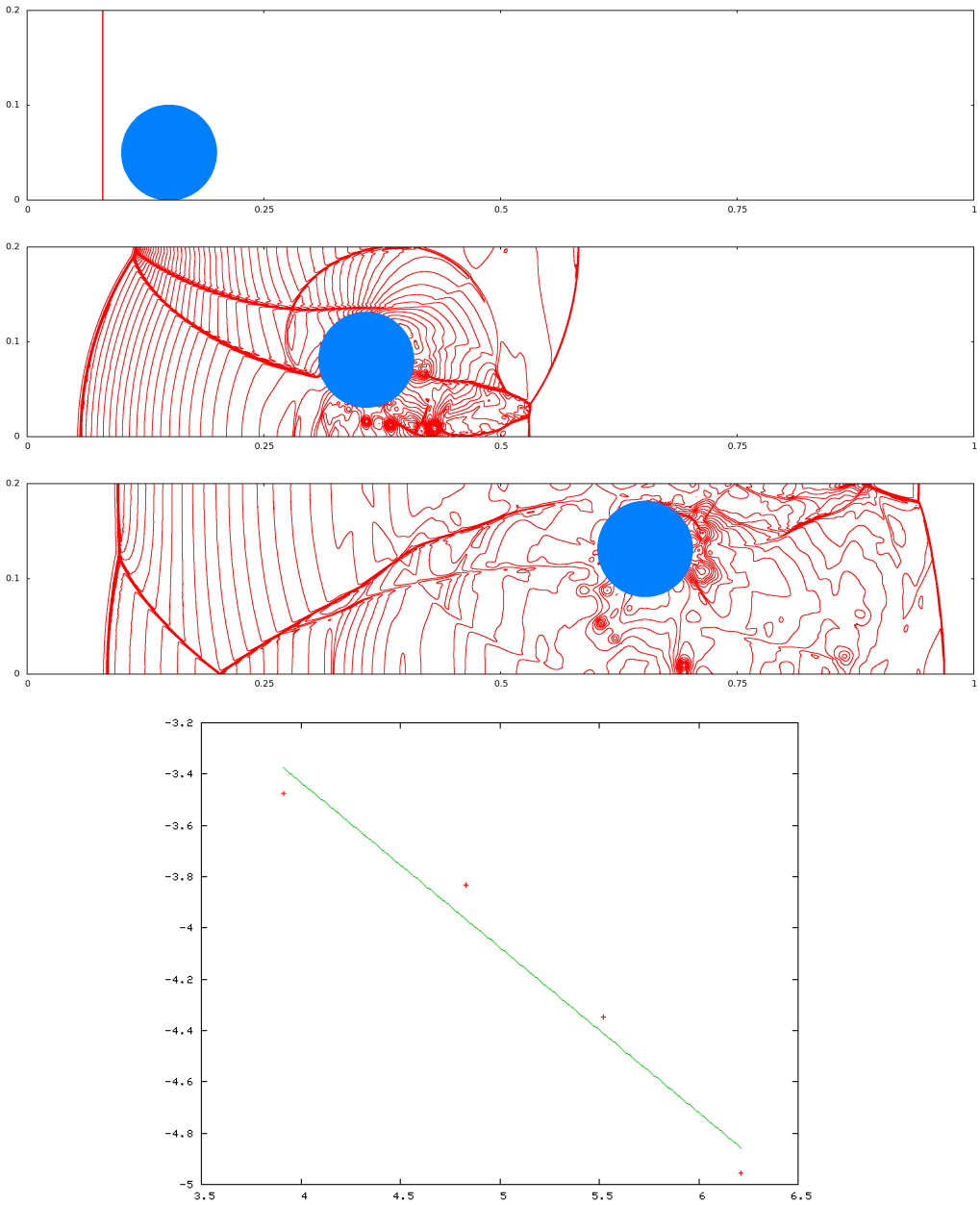


(a) Position of the free end of the spring, as a function of time.



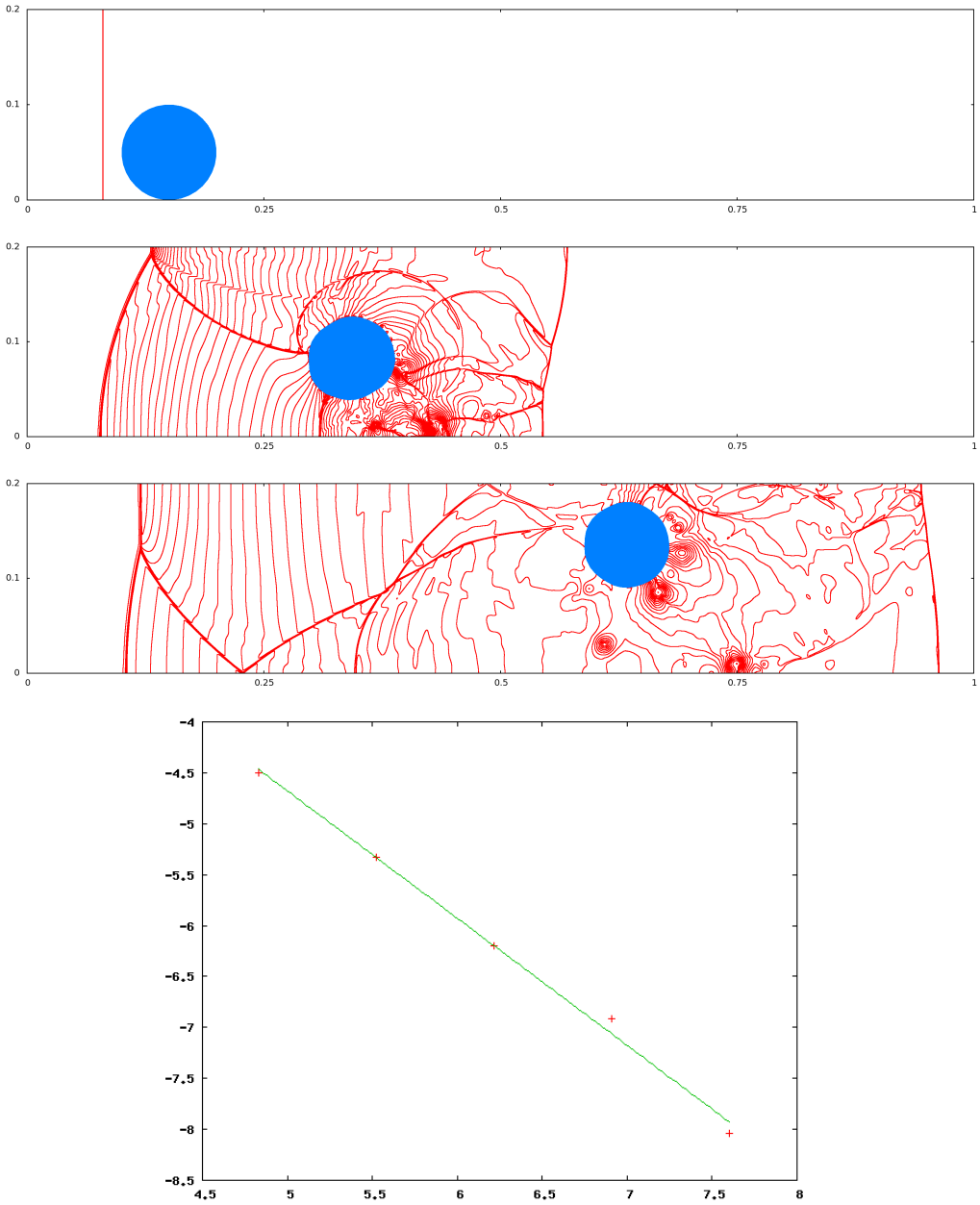
(b) Position error for the left-most side of the mass-spring system, as compared to the analytic solution provided in [1], at time .008s. We plot the log of the relative error, as a function of the log of the resolution of the underlying grid. The convergence rate is 1.16.

Figure 3.14: 1-D mass-spring system hit by a Sod shock wave.



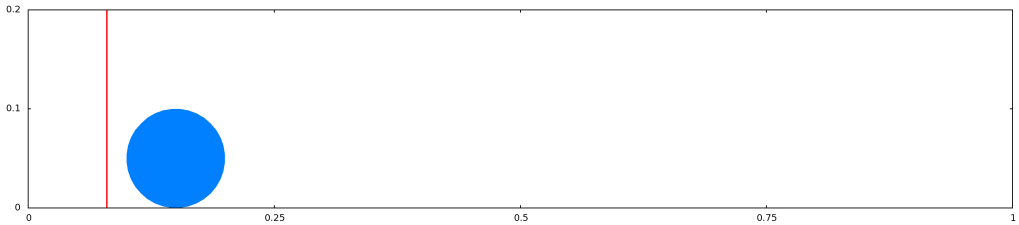
(d) Position error of the center of mass of the cylinder hit by a planar shock, as compared to a high-resolution simulation, at time $t = .15s$, with a convergence of .96.

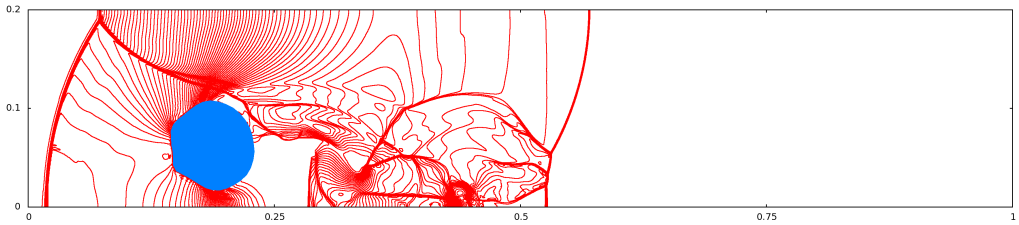
Figure 3.15: Pressure contours for semi-implicit simulation of rigid cylinder lift off are shown at $t = 0$, $t = .164$ and $t = .301$. The simulation is run with a CFL number of .6, using the CFL restriction discussed in Equation 3.22.

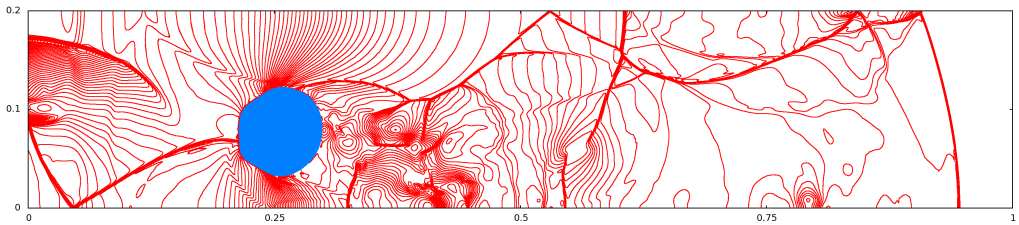


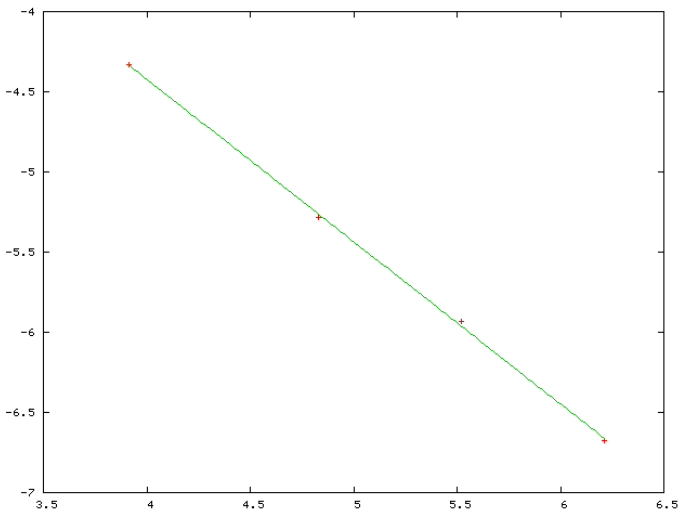
(d) Position error of the center of mass of the deformable cylinder hit by a planar shock, as compared to a high-resolution simulation, at time $t = .15s$. We plot the log of the relative error, as a function of the log of the resolution of the underlying grid. The convergence rate is .99.

Figure 3.16: Pressure contours for semi-implicit simulation of deformable cylinder lift off are shown at $t = 0$, $t = .164$ and $t = .301$. The simulation is run with a CFL number of .6, using the CFL restriction discussed in Equation 3.22.









(d) Position error of the center of mass of the heavy deformable cylinder hit by a planar shock, as compared to a high-resolution simulation, at time $t = .15s$. We plot the log of the relative error, as a function of the log of the resolution of the underlying grid. The convergence rate is 1.01.

Figure 3.17: Pressure contours for semi-implicit simulation of deformable cylinder lift off are shown at $t = 0$, $t = .164$ and $t = .301$. The simulation is run with a CFL number of .6, using the CFL restriction discussed in Equation 3.22.

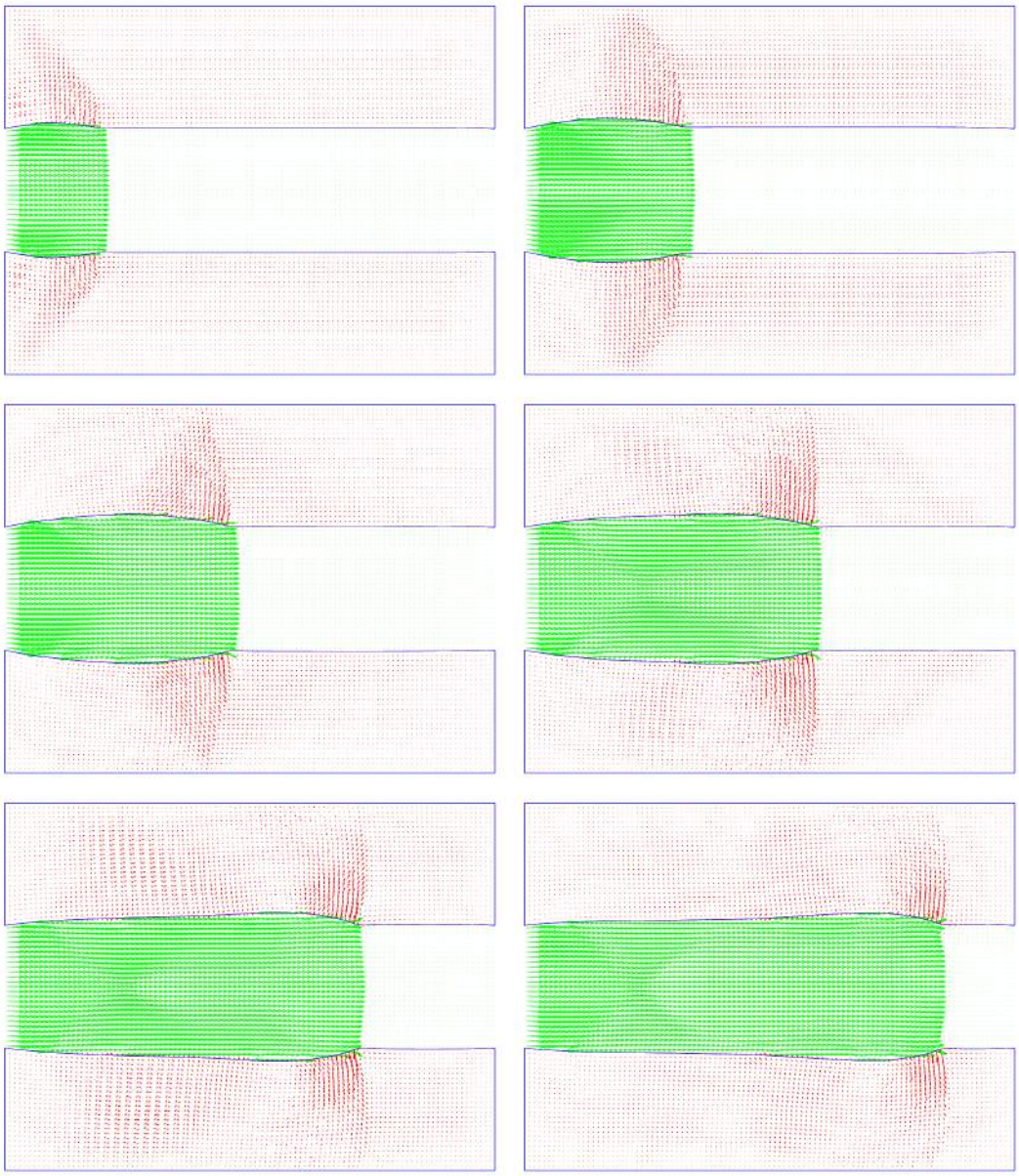


Figure 3.18: A planar shock travels down a deformable bladder. Shown are the velocity field of the fluid in green and the velocities of the deformable nodes in red at times $t = .0001$, $t = .0002$, $t = .0003$, $t = .0004$, $t = .0005$ and $t = .0006$.

Shock traveling down a deformable tube

This example is similar to the inflatable bladder examples suggested in [1] and [22] in which a shock wave travels through a deformable tube causing large deformation of the walls. Our results are shown in Figure 3.18. We also do a convergence analysis of our method, with convergence 1.18. The error in the position of a particle on the deformable tube is computed at time $.00049s$ (which is the approximate time of maximum deformation of that particle in the highest resolution simulation) from the highest resolution grid simulated, which is 800×600 grid cells. The convergence order of the error is estimated as 1.18.

Symmetric positive-definite reformulation

Our numerical method is symmetric, but not positive-definite. Recent developments in [88] discuss a modification of the implicit coupling methodology for incompressible flow by separating out the coupling forces as implicit variables λ (similar to immersed boundary methods), decomposing the symmetric damping force into $D = C^T C$ and solving for $\hat{V}_s = CV_s^{n+1}$. The symmetric positive-definite system they obtain can be modified for compressible flow in a manner similar to Section 3.4.1 to obtain

$$\begin{pmatrix} \frac{V}{\Delta t^2 \rho c^2} I + \hat{G}^T \beta^{-1} \hat{G} & -\hat{G}^T \beta^{-1} K^T & 0 \\ -K\beta^{-1}\hat{G} & K(\beta^{-1} + WM^{-1}W^T)K^T & KWM^{-1}C^T \\ 0 & CM^{-1}W^T K^T & I + CM^{-1}C^T \end{pmatrix} \begin{pmatrix} \tilde{p} \\ \lambda \\ \hat{V}_s \end{pmatrix} = \begin{pmatrix} \frac{V}{\Delta t^2 \rho c^2} \tilde{p}^a + \hat{G}^T u^\star \\ KWV_s^\star - Ku^\star \\ CV_s^\star \end{pmatrix}, \tag{3.23}$$

where $\hat{G}$ and $-\hat{G}^T$ are the volume weighted gradient and divergence operators respectively, β is the diagonal matrix of fluid dual cell masses, and K^T is the matrix of 1s and 0s mapping λ to the appropriate fluid velocity scalars (see [88] for more details). Note that in order to avoid confusion in notation we renamed a few operators. In particular W and J in [88] correspond to the K and W we use here, respectively. This system is both symmetric and positive-definite. We demonstrate the viability of this modified method in another example, where we've replaced the implicit coupled solve with Equation (3.23). Our example is similar to the example in Section 3.6.2 except that the sphere is replaced with a diamond whose major axis is of length .1

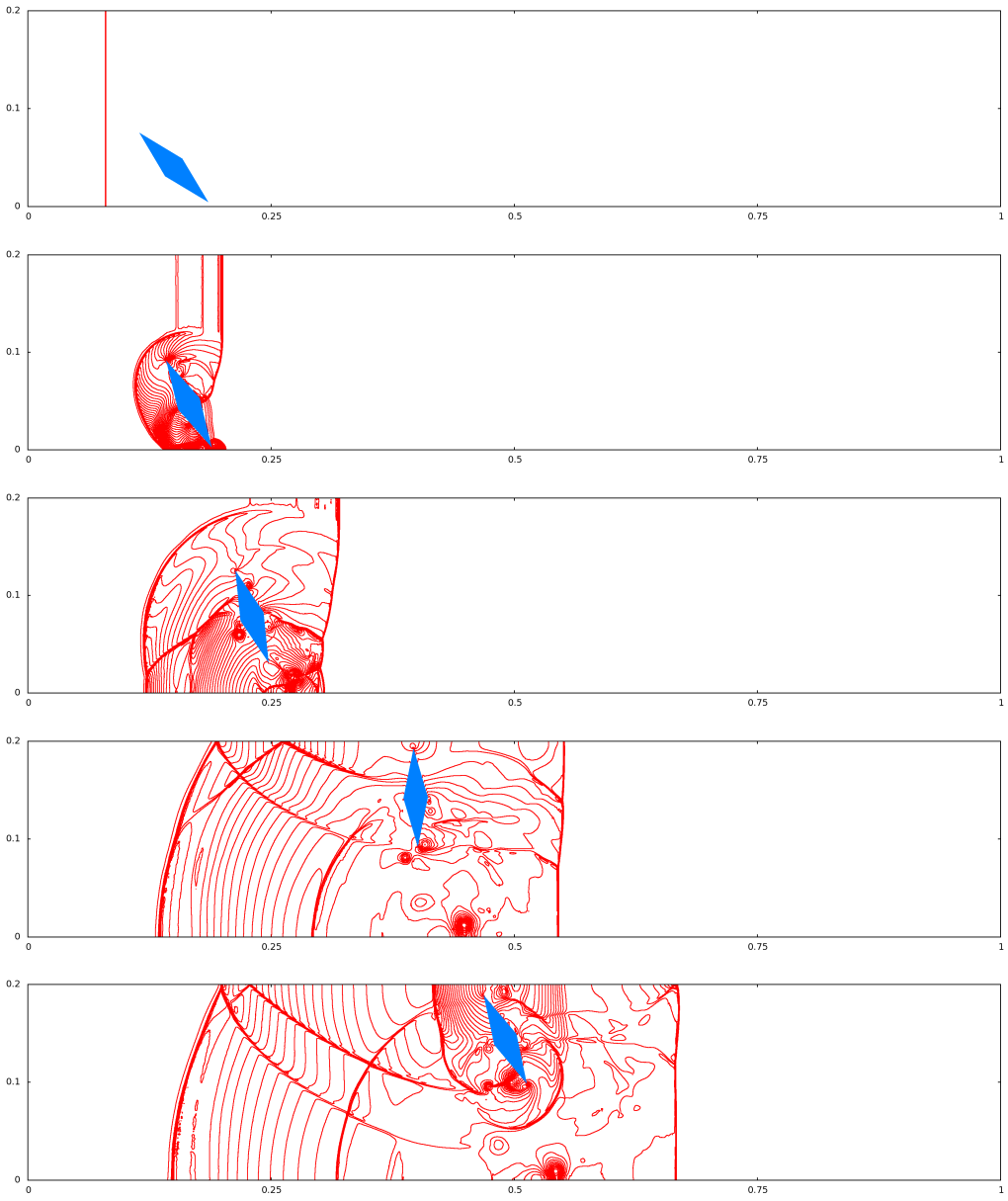


Figure 3.19: A diamond is hit by a planar shock, and then collides with the top of the channel. Shown are pressure contours at $t = 0$, $t = .04$, $t = .08$, $t = .16$ and $t = .2$.

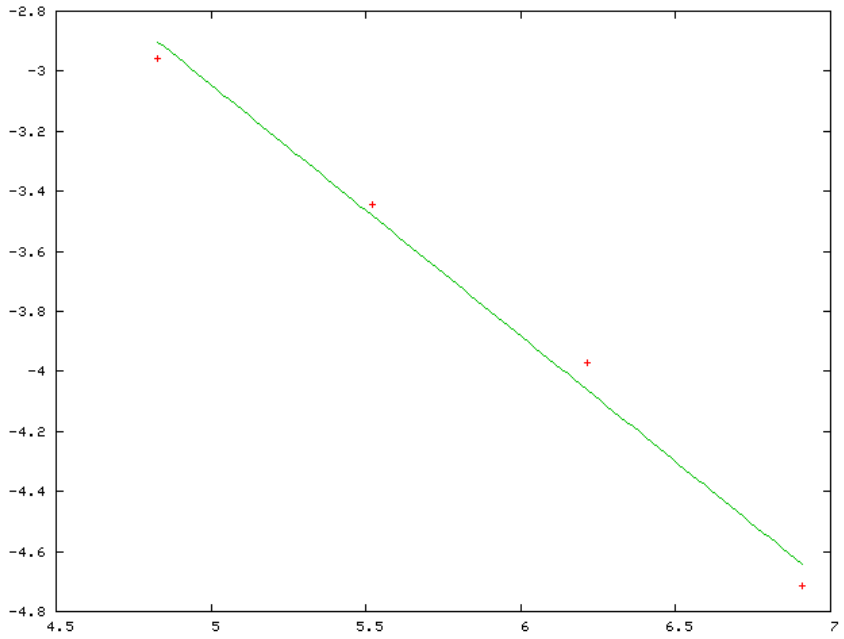


Figure 3.20: Position error of the center of mass of the diamond hit by a planar shock, as compared to a high-resolution simulation, at time $.15s$. We plot the log of the relative error, as a function of the log of the resolution of the underlying grid. The convergence rate is .84.

and minor axis is of length .025. The diamond begins rotated by $\pi/4$, with a center of mass at $(.15, .04)$. Snapshots of the resulting simulation are shown in Figure 3.19. The convergence analysis for this example is shown in Figure 3.20 which estimates the convergence order of the error as .84.

3.7 Conclusions and future work

We have presented a first order method which implicitly couples compressible flow with solid bodies with arbitrary constitutive models. We show that this method is robust, numerically conservative, and avoids the numerical instabilities which comparable explicit methods suffer from in the presence of high density-to-mass ratios. The same methodology can be applied to reformulate our implicit system into a symmetric positive-definite system.

There are several interesting avenues of future work which we wish to explore. Given the promising results which arise from handling fluid-structure interactions implicitly, we believe that an alternative approach would split the fluid flux along Riemann invariants–rather than by pressure–and solve for the Riemann invariant which interacts with the solid implicitly. Our method also relies on the assumption that the solid has some thickness where ghost cells can be filled, and we believe that the method can be made to work for thin shell structures (such as parachutes). Given the utility of the scheme proposed in [50] in handling fluid-structure interactions, it becomes imperative to address the issues of that original scheme. In particular, the implicit component of the method is overly centrally-differenced, which tends to introduce Gibbs phenomena at shocks. It would be better to add upwind biasing, although it is unclear how to do so.

Chapter 4

Conservative Advection

Semi-Lagrangian methods have been around for some time, dating back at least to [14]. Researchers have worked to increase their accuracy, and these schemes have gained newfound interest with the recent widespread use of adaptive grids where the CFL-based time step restriction of the smallest cell can be overwhelming. Since these schemes are based on characteristic tracing and interpolation, they do not readily lend themselves to a fully conservative implementation. However, we propose a novel technique that applies a conservative limiter to the typical semi-Lagrangian interpolation step in order to guarantee that the amount of the conservative quantity does not increase during this advection. In addition, we propose a new second step that forward advects any of the conserved quantity that was not accounted for in the typical semi-Lagrangian advection. We show that this new scheme can be used to conserve both mass and momentum for incompressible flows. For incompressible flows, we further explore properly conserving kinetic energy during the advection step, but note that the divergence free projection results in a velocity field which is inconsistent with conservation of kinetic energy (even for inviscid flows where it should be conserved). For compressible flows, we rely on a recently proposed splitting technique that eliminates the acoustic CFL time step restriction via an incompressible-style pressure solve. Then our new method can be applied to conservatively advect mass, momentum and total energy in order to exactly conserve these quantities, and remove the remaining time step restriction based on fluid velocity that the original scheme

still had.

4.1 Introduction

The idea of applying the method of characteristics to advect quantities forward in time dates back at least as far as [14] and has gained popularity in many areas, such as atmospheric sciences [95]. Although the simplest schemes trace back along straight line characteristics and use low order interpolation to estimate the data, one can trace back arbitrarily high order curved characteristics and use arbitrarily high order interpolation, see for example [77]. The simplicity of these schemes makes them quite useful for adaptive grids and other data structures, see for example [18, 66, 65, 96, 39]. Recently, authors have considered using semi-Lagrangian methods as building blocks in other schemes, for example [44, 45, 17] showed that the second order accurate BFECC method of [16] can be made unconditionally stable using the first order accurate semi-Lagrangian method as a building block. In addition, [90] showed that the original scheme of MacCormack [73] can be made unconditionally stable in a similar way. A notable feature of the semi-Lagrangian method is that it relieves the time step restriction. This is part of the reason why it has received such interest from the atmospheric sciences community [53, 57, 54, 111], as well as the compressible flow community [55, 87] where the acoustic time step restrictions can be severe. We refer the reader to a particularly interesting body of work that considers a number of methods for making semi-Lagrangian schemes conservative, considering one spatial dimension, multiple spatial dimensions with splitting, multiple spatial dimensions without splitting, and even obtaining conservation from a non-conservative form [99, 106, 78, 105, 98].

Intuitively, the idea behind a fully conservative semi-Lagrangian scheme is simply to advect the conserved quantities along characteristic paths in a way that is careful to respect conservation. Many numerical methods are based on this principle, for example SPH methods push around chunks of mass, momentum and energy assigned to particles, and have been used to solve both compressible and incompressible flows,

including flows with shock waves, see for example [28, 71, 48, 47, 108, 15, 13, 59, 68, 85, 29]. In fact, the idea of pushing around conserved quantities is the basis for volume of fluid methods, which attempt to conserve volume (see for example [83, 86, 58, 36, 63, 113]). In addition, ALE methods also push material around using a moving grid, and some of those methods use a background grid along with a two-step procedure where the material is first advected forward on a moving grid, and then remapped or redistributed to the background mesh in a conservative fashion, see for example [37, 75, 69, 70, 12]. Obviously this idea of pushing around mass in a conservative way respecting propagational characteristics for the sake of consistency has received quite a bit of attention.

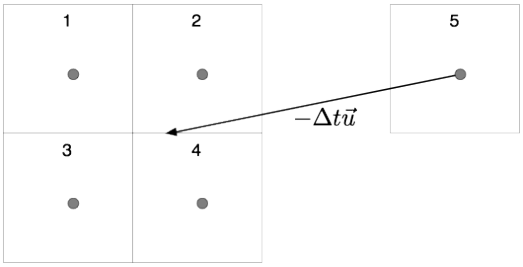
Notably, our method is quite simple, both conceptually and as far as implementation is concerned—it requires only a small modification to a standard semi-Lagrangian scheme and utilizes most of the functionality already present. The standard semi-Lagrangian method updates the value at a grid point by tracing a possibly curved characteristic backwards in time to find its point of origin, interpolating the surrounding data to that point, and placing the result of the interpolation at the original grid point. In this manner, a grid point is updated with a linear combination of data from other points. One can view this as placing some fraction of the data from other grid points at this point, and then consider what this means from the point of conservation of this data. Considering the grid as a whole, each grid point traces back some characteristic and obtains some fraction of data stored at other grid points. One can see that the scheme is not conservative, since certain grid points contribute to multiple interpolations and the sum of all the weights *from* that grid point *to* all the points where the interpolations were performed can be larger than one. This means that the data contained at the grid point has been over-depleted, violating conservation. Similarly, some grid points may not be asked for any of their data at all, or the sum of the inquisitive weights may be less than one. This also violates conservation, in the sense that data has been left at that grid point and not advected forward. Of course, we could simply account for this data by leaving it at that grid point, but then the scheme would be inconsistent as this data needs to be advected forward. We note that it is trivial to cure both of these pathologies in the semi-Lagrangian scheme by

simply ensuring that the sum of the interpolation weights *from* every point adds to one, and that any data that wasn't advected forward is pushed forward in our second semi-Lagrangian step.

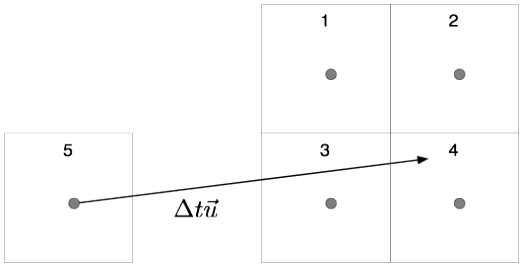
To summarize, we make the following modifications to a standard semi-Lagrangian method. Each grid point is thought of as a control volume, containing a certain amount of conserved quantity similar to any other conservation law solver. We trace back the potentially curved semi-Lagrangian rays in the usual manner, perform the interpolation in the usual manner, but add the additional step of recording all the interpolation weights for every grid point so that we may check whether or not they are equal to one. Our first correction requires sweeping through the grid, identifying any grid node which has been asked for more information than it contains (i.e. sum of the weights is greater than one), and subsequently scaling down these weights such that their sum is exactly equal to one. Then these corrected weights are used in place of the standard weights in the semi-Lagrangian advection scheme. At this point, the standard semi-Lagrangian scheme is completed, however as mentioned above, we have not advected all of the conserved quantity forward in time. Thus, for each grid point whose weights sum to less than one, we need to advect the remaining conserved data forward in time for consistency. This is done via a second application of the semi-Lagrangian method starting at that grid point and tracing a potentially curved characteristic *forward* in time, to see where it lands (exactly opposite of the standard semi-Lagrangian method). The remaining data at that point is placed at its new location, however this new location will not lie at a grid point but will instead lie inside some grid cell. We distribute the remaining data to the surrounding grid points by noting that the transpose of an interpolation operator is a conservative distribution operator. That is, we simply calculate the interpolation weights at the new point, just as one would in a standard semi-Lagrangian interpolation, and use those weights to determine how much of the quantity is distributed to each of the surrounding points. Notably, the building blocks for the second step already exist in most implementations, only the tracing of a characteristic and the computation of interpolation weights are needed in the algorithm.

In this paper, we consider the application of our method to both incompressible

and compressible flow. As far as mass is concerned, treating a variable-density incompressible flow and the density equation in compressible flow requires only straightforward application of the method. As far as momentum and energy are concerned, we take an approach which is similar for both incompressible and compressible flows. In particular we use the method of [50] in order to solve the compressible flow equations in a way that requires an advection step followed by a pressure solve similar to incompressible flow, but which contains an identity term since pressure is based on the time dependent pressure evolution equation. Thus, both methods consist of a conservative advection step, followed by an implicit solve for the pressure, and a final pressure correction step. In the case of incompressible flow, our new semi-Lagrangian method can be used to exactly conserve the momentum of the fluid, and if the pressure correction is viewed as a flux, then one can conserve momentum in that step as well. In addition, we show how to account for stationary walls and potentially moving solid object boundaries. The treatment for compressible flow is similar, except mass, momentum and energy are conservatively advected with our semi-Lagrangian scheme before the pressure is solved for and the correction is applied. We show how to apply the pressure correction in such a way so that both the momentum and total energy are conserved, especially near solid walls and object boundaries. Finally, we note that a conservation style equation can be formulated for the kinetic energy of an incompressible flow. This equation is similar to that for compressible flow, with total energy replaced by kinetic energy along with the appearance of a source term for losses due to viscosity. Although our scheme can be used to conservatively advect kinetic energy, and accounting for the viscous source term is straight-forward, the pressure projection step is inconsistent with the conservation of kinetic energy and therefore the resulting divergence-free velocity field disagrees with that predicted by kinetic energy conservation. We provide some analysis of this along with quantitative results.



(a) Cell 5 casts a ray backward, $-\Delta t\vec{u}$, which lands between cells 1, 2, 3 and 4. Using a standard bilinear interpolation scheme, the weights are calculated to be $w_{15} = .125$, $w_{25} = .375$, $w_{35} = .125$, $w_{45} = .375$.



(b) Cell 5 casts a ray forward, $\Delta t\vec{u}$, which lands between cells 1, 2, 3 and 4. We again use standard bilinear interpolation, giving forward-cast weights $f_{51} = .06$, $f_{52} = .24$, $f_{53} = .14$, $f_{54} = .56$.

Figure 4.1: Standard semi-Lagrangian advection schemes cast rays either forward or backward along characteristic lines in order to determine time t^{n+1} values at cell centers. We take advantage of this in our scheme, making use of the computed weights w_{ij} and f_{ij} as appropriate. The notation w_{ij} and f_{ij} denote the contribution that cell i gives to cell j over a time step.

4.2 Conservative semi-Lagrangian method

We begin by discussing the standard semi-Lagrangian method as applied in the simplest case of a passively advected scalar ϕ, in a velocity field $\vec{u}$,

$$\phi_t + \vec{u} \cdot \nabla\phi = 0. \tag{4.1}$$

Combining this equation with conservation of mass, $\rho_t + \nabla \cdot (\rho\vec{u}) = 0$, leads to the conservative form of the same equation:

$$(\rho\phi)_t + \nabla \cdot (\phi\rho\vec{u}) = 0. \tag{4.2}$$

For the sake of exposition, we define $\hat{\phi} = \rho\phi$ as the conserved quantity; this allows us to interchangeably talk about ϕ, the passively advected scalar, and $\hat{\phi}$, the conserved quantity. For each grid point $\vec{x}_j$, the semi-Lagrangian method would trace a potentially curved characteristic ray backward in time to some position $\vec{x}$, and use an interpolation kernel to obtain a value of $\hat{\phi}$ at $\vec{x}$. This value is then used as $\hat{\phi}(\vec{x}_j, t^{n+1})$. The first order accurate case is illustrated by Figure 4.1(a), where a straight line

characteristic is traced backward in time from cell 5 to find $\vec{x}$ in-between cells 1, 2, 3 and 4. In equation form, this is given by

$$\hat{\phi}(\vec{x}_j, t^{n+1}) = \hat{\phi}(\vec{x}, t^n) = \sum_i w_{ij} \hat{\phi}(\vec{x}_i, t^n), \tag{4.3}$$

where w_{ij} are interpolation weights such that $\vec{x} = \sum_i w_{ij} \vec{x}_i$. Dimension-by-dimension linear interpolation yields a first order method. Notably, $\sum_j w_{ij} = 1$ for any consistent interpolation operator, regardless of the size of the stencil or order of accuracy.

After updating $\hat{\phi}$ at every grid point, we can then define the total contribution from cell i to the time t^{n+1} data as $\sigma_i = \sum_i w_{ij}$, noting that this is not expected to sum to 1 due to numerical truncation errors. In fact, since $\hat{\phi}$ is conserved as shown in Equation (4.2), in order to exactly conserve data during the semi-Lagrangian update, σ_i should be exactly 1. Fixing this is the key idea of our numerical method. This is accomplished by visiting each donor grid cell i, examining σ_i, and scaling down the weights w_{ij} to $\hat{w}_{ij} = w_{ij}/\sigma_i$ when $\sigma_i \geq 1$, which guarantees explicitly that we do not artificially create $\hat{\phi}$.

Next we treat the cells for which $\sigma_i < 1$. At these cells we apply a second pass of the standard semi-Lagrangian scheme, casting rays *forward*, as illustrated for a first order accurate method in Figure 4.1(b), yielding forward-cast weights f_{ij}. Noting that the transpose of an interpolation operator is a conservative distribution operator, we use these weights f_{ij} to distribute the remaining $\hat{\phi}$, i.e. $(1 - \sigma_i)\hat{\phi}_i$, to the cells j used to perform the interpolation. This can be seen as incrementing the unclamped w_{ij} weights from the first step by an amount equal to $(1 - \sigma_i) f_{ij}$, so that the final weights are $\hat{w}_{ij} = w_{ij} + (1 - \sigma_i) f_{ij}$. Our update is then given as

$$\hat{\phi}_j^{n+1} = \sum_i \hat{w}_{ij} \hat{\phi}(x_i, t^n). \tag{4.4}$$

At the end of our two applications of the standard semi-Lagrangian steps, we now have modified weights $\hat{w}_{ij}$ to satisfy $\sum_i \hat{w}_{ij} = 1$. That is, every cell on the grid contributes exactly everything it has at time t^n to the time t^{n+1} solution along the characteristic lines which pass through the cell.

To summarize, when $\sigma_i \geq 1$, we clamp the w_{ij} to obtain $\hat{w}_{ij} = w_{ij}/\sigma_i$; using these new $\hat{w}_{ij}$ weights leads to $\sigma_i = 1$. Otherwise if $\sigma_i < 1$, we forward advect the non-advected data at each grid point and use it's placement to calculate the new weights $\hat{w}_{ij}$ which also lead to $\sigma_i = 1$. We note that in the $\sigma_i \geq 1$ case, one could also forward advect and interpolate. In this fashion, one would be advecting negative material to cancel out the excess of positive material that was advected by the first semi-Lagrangian step. However, when this negative material is place at surrounding grid nodes using the f_{ij} weights, it is possible for the target grid node x_j to lose more of the conserved quantity than it originally had. Thus, for now, we only consider the method of clamping even though it seems to limit the method to first order accuracy.

4.2.1 Boundary conditions

In the application of our method, we consider a number of different boundary conditions. For open boundaries, inflow and outflow are treated by adding and filling the appropriate number of ghost cells. For inflow boundary conditions, rays which extend out of the domain are treated in the standard semi-Lagrangian fashion, and the amount of material donated from ghost cells to points interior to the domain is considered to be our inflow. One could modify the inflow scheme to not simply perform semi-Lagrangian interpolation but instead conservatively advect the sum of the ghost cell data, however this requires careful accounting since, as ghost cells, some of these are not solved for.

Unlike the standard scheme where only interior points need to be updated, our outflow boundaries require evaluation of ghost nodes in the numerical scheme to ensure that they withdraw the correct amount of $\hat{\phi}$ from the interior of the grid. Moreover one needs to ensure that enough ghost cells are updated, such that the information is correctly withdrawn from the interior of the domain. After clamping, one also needs to consistently advect data from interior nodes to ghost cells when $\sigma_i < 1$.

Throughout the paper, we measure our conservation error at time t^n using the

following equation:

$$\text{Error}(t^n) = \Sigma\hat{\phi}(t^n) - \left[\Sigma\hat{\phi}(t^0) + \Sigma_{in} - \Sigma_{out}\right], \tag{4.5}$$

where the first term on the right-hand side represents the current amount of $\hat{\phi}$ on the grid and the second term represents the initial amount. These should only vary through inflow and outflow which are represented by the third and fourth terms. When updating $\hat{\phi}_i^{n+1}$ from $\hat{\phi}^n$, if a semi-Lagrangian ray reaches back to ghost cells and pulls information into the domain, then we track that for the Σ_{in} term. If information is transported from the interior of our grid to the ghost cells, we track that for the Σ_{out} term. That is, $\hat{w}_{ij}$'s which contribute to a ghost cell j are accounted for in Σ_{out}. There is rich literature on treating inflow and outflow boundary conditions for fluid flows, and we imagine that many variations of our method could be designed in such a way that is consistent with our treatment of the interior of the domain. However, we found this sufficient for our examples.

Near solid walls and moving object boundaries, one must be careful not to interpolate across or into the wall or object. All rays that are cast are done in a collision-aware manner, stopping any rays early if they would pass through the interface, similar to the computational geometry approach detailed in the computer graphics literature (see e.g. [32]). Typically, when performing *interpolation* as in [32] we use information from the solid, such as its velocity. However, that would transfer information from the solid to the fluid, for example, during momentum advection one would be interpolating momentum from the solid. This is non-physical since advection should not transport conserved quantities across material interfaces. Any transmission of momentum from the solid to the fluid should instead occur when considering the acoustic characteristics, for example when solving for the pressure (which we consider later). Thus, for our scheme we simply set $w_{ij} = 0$ for any interpolation point which is not visible. At this point one could consider scaling up the remaining weights to get interpolation weights such that $\Sigma_i w_{ij} = 1$, although we have not experimented numerically with this option. Finally, in the forward-casting step of the scheme, in order to guarantee conservation we set $f_{ij} = 0$ if cell j is not visible from

the interpolation point, and the remaining weights are then scaled up to account for the missing material.

4.2.2 Interpolation

For our new conservative semi-Lagrangian approach we require an interpolation scheme to determine the weights. A simple method uses linear weights between the nearest points as shown in Figure 4.1. While this works rather well for conserving energy as well as converging to the correct solution, the interpolation error can be reduced through the use of higher order interpolation. Consider for example quadratic interpolation. If our point of interest x lies between cells i and $i+1$, then we have available two valid quadratic functions: a left-biased one which interpolates across the range (x_{i-1}, x_i, x_{i+1}), and a right-biased one that interpolates across the range (x_i, x_{i+1}, x_{i+2}). The left-biased quadratic produces weights for an interpolated point x as:

$$\alpha_{i-1,L} = \frac{\bar{x}(\bar{x}-1)}{2}, \qquad \alpha_{i,L} = 1 - \bar{x} - \bar{x}(\bar{x}-1), \qquad \alpha_{i+1,L} = \bar{x} + \frac{\bar{x}(\bar{x}-1)}{2}$$

while the right-biased quadratic produces weights for an interpolated point x as:

$$\alpha_{i,R} = 1 - \bar{x} + \frac{\bar{x}(\bar{x}-1)}{2}, \qquad \alpha_{i+1,R} = \bar{x} - \bar{x}(\bar{x}-1), \qquad \alpha_{i+2,R} = \frac{\bar{x}(\bar{x}-1)}{2}$$

where $\bar{x} = (x - x_i)/\Delta x$. While sufficient for a standard semi-Lagrangian scheme, these interpolations will produce negative weights on the outlying cells ($i-1$ for the left-biased one, $i+2$ for the right-biased one) when $x_i < x < x_{i+1}$. To alleviate these negative weights we instead always zero the weight on the outlying cell and push the missing contribution inward. That is, for the left-biased polynomial the weights would be

$$\tilde{\alpha}_{i-1,L} = 0, \qquad \tilde{\alpha}_{i,L} = 1 - \bar{x} - \frac{\bar{x}(\bar{x}-1)}{2}\left[2 - \frac{\hat{\phi}_{i-1}}{\hat{\phi}_i}\right], \qquad \tilde{\alpha}_{i+1,L} = \bar{x} + \frac{\bar{x}(\bar{x}-1)}{2}$$

Similarly, for the right-biased polynomial the weights would be

$$\tilde{\alpha}_{i,R} = 1 - \bar{x} + \frac{\bar{x}(\bar{x}-1)}{2}, \qquad \tilde{\alpha}_{i+1,R} = \bar{x} - \frac{\bar{x}(\bar{x}-1)}{2}\left[2 - \frac{\hat{\phi}_{i+2}}{\hat{\phi}_{i+1}}\right], \qquad \tilde{\alpha}_{i+2,R} = 0.$$

This preserves the *value* given by the higher-order interpolation scheme and significantly reduces the likelihood of a negative weight. Note that both of the quadratic interpolations provide a second order correction to a linear interpolation. If we take *both* of these interpolation schemes and average them, we get the weights that we use in the quadratic version of our scheme:

$$\alpha_i = 1 - \bar{x} - \frac{\bar{x}(\bar{x}-1)}{4}\left(1 - \frac{\hat{\phi}_{i-1}}{\hat{\phi}_i}\right), \qquad \alpha_{i+1} = \bar{x} - \frac{\bar{x}(\bar{x}-1)}{4}\left(1 - \frac{\hat{\phi}_{i+2}}{\hat{\phi}_{i+1}}\right).$$

Using these modified weights we can then perform our semi-Lagrangian steps as discussed earlier. conservative-sl-figures 4.3, 4.4 and 4.5 demonstrate the significant error improvement by using this interpolation scheme. Note that in these conservative-sl-figures, using second-order Runge-Kutta to trace characteristic lines gives no numerical differences, as the first order approximation is already exact.

Whereas arbitrarily high order characteristics can be traced using our semi-Lagrangian scheme, it is this negativity in the interpolation weights which so far has restricted our method to first order accuracy. Negative weights are not entirely detrimental, and in fact the quadratic version of our scheme admits that to some lesser degree. If the sum of all the weights at a grid node is equal to some $\epsilon < 0$ at a grid node, then we simply forward-advect $1 + \epsilon$ amount of material. The problem is that a typical quadratic interpolation scheme can have rather large positive weights balancing out rather large negative weights on the side of the interval from which two points are used, and this seems to lead to difficulties. Our process of merging the weights to form $\tilde{\alpha}$ from α tends to cancel out these large positive and negative values making the result more reasonable. Of course one can guarantee that the weights never become negative by simply using standard multi-linear interpolation.

It may be possible to make a second order accurate scheme using only order

first order accurate interpolation stencils, as was done in the modified MacCormack scheme of [90] and the modified BFECC scheme of [17]. Another interesting approach would be to apply a second order non-conservative correction to a full conservative first order accurate scheme.

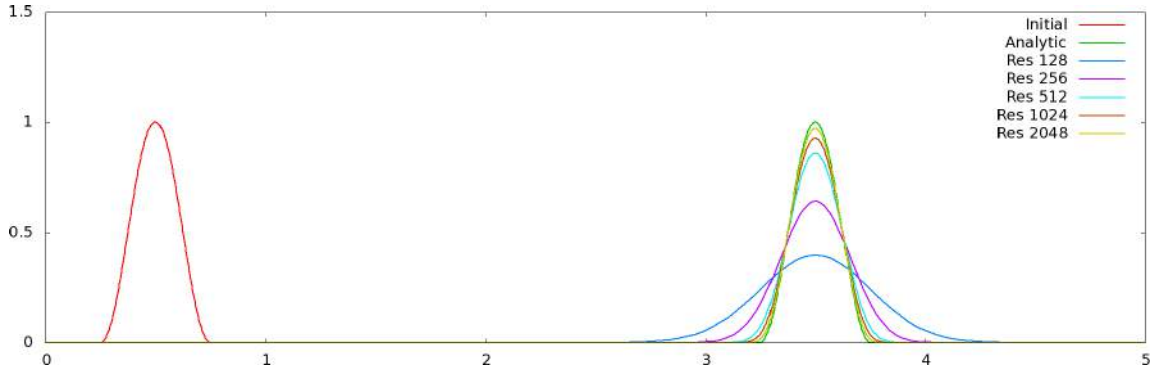
4.2.3 Examples

In order to demonstrate the conservation properties of our scheme, we consider an advected sine-wave "bump" using a constant velocity field. That is,

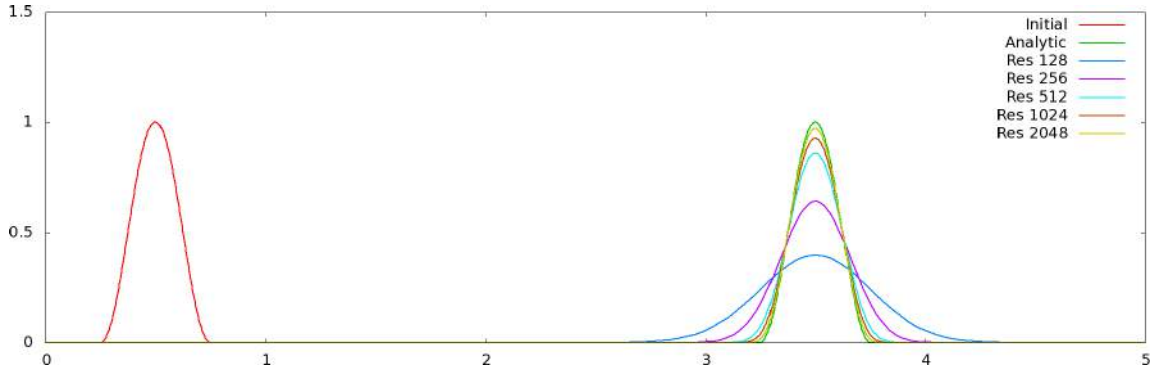
$$\hat{\phi}(x,0) = \begin{cases} \frac{1}{2}\left(1+\sin\left(4\pi * (x-\frac{3}{8})\right)\right) & .25 \leq x \leq .75 \\ 0 & \text{else} \end{cases} \tag{4.6}$$

with $u = 1$. The problem is discretized over the domain $[0, 5]$, and we solve Equation (4.2) with a CFL number of .9. In Figure 4.2(a), we show the solution as computed by a standard non-conservative semi-Lagrangian advection, while Figure 4.2(b) shows the solution computed by our new scheme. As we expect, the solutions of the two methods agree and both converge to the analytic solution. Figure 4.3 shows a comparison between using linear and quadratic interpolation in our method. Figure 4.4 shows the same comparison using a CFL number of 2.9 instead of 0.9. Note that the errors are much smaller since approximately three times fewer time steps (and thus interpolations) are needed. In Figure 4.5 we run this simulation three times longer with a CFL of 2.9 showing errors more commensurate with Figure 4.3 as expected. We also demonstrate the order of convergence in Table 4.1 which shows that our algorithm gives first order convergence.

Figure 4.6 considers a square wave in the divergent velocity field $u = \sin(\pi x/5)$. Note the marked difference between the conservative and non-conservative method. Figure 4.8 shows dramatic loss of conservation in the standard semi-Lagrangian method as compared to the conservative version which maintains exact value up to round off error. Figure 4.7 shows a convergence analysis for the two schemes using a high resolution full conservative ENO method [93] as a ground truth. As is typical the non-conservative method converges to the wrong solution whereas our new



(a) Standard semi-Lagrangian advection.



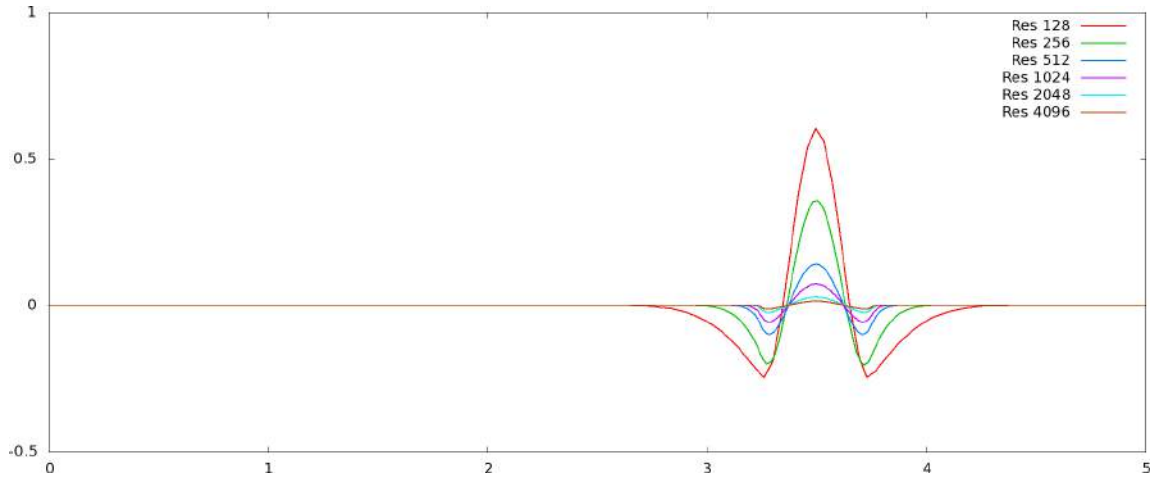
(b) Conservative semi-Lagrangian advection.

Figure 4.2: A sine-wave "bump" is advected through a uniform velocity field. Shown is the solution at time $t = 3s$. We apply the first order version of both the standard semi-Lagrangian advection, as well we our proposed conservative semi-Lagrangian advection scheme.

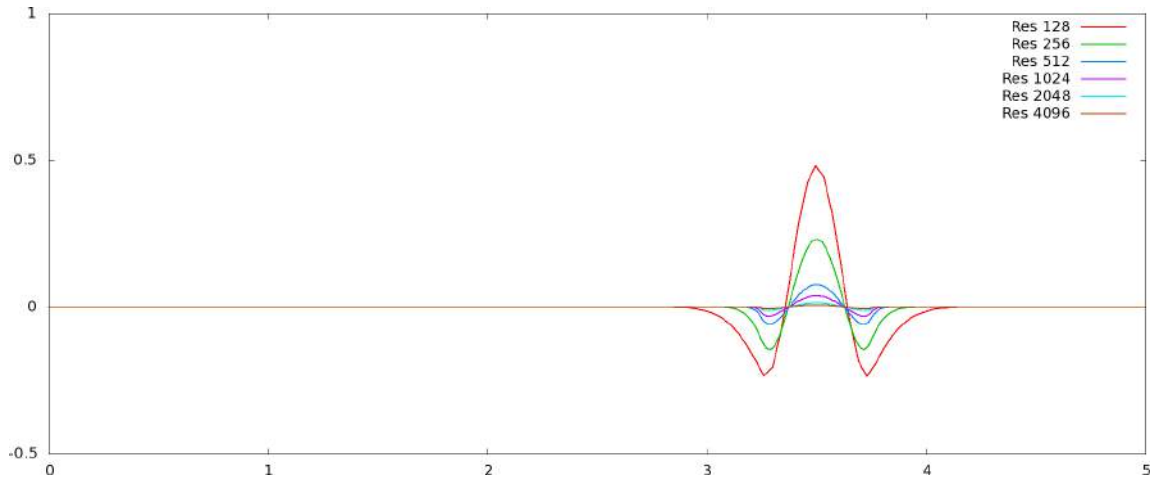
method converges to the result obtained via ENO. The reason the non-conservative method converges to the wrong solution in this case is that it solves Equation (4.1) whereas our method solves Equation (4.2). In comparing these two equations, standard semi-lagrangian advection is missing the $\hat{\phi}(\nabla \cdot \vec{u})$ term.

We also consider the Zalesak disc example, discussed in [110]. In this example a notched disk is advected through a velocity field specified by

$$u = (\pi/314)(50 - y)$$
$$v = (\pi/314)(x - 50)$$

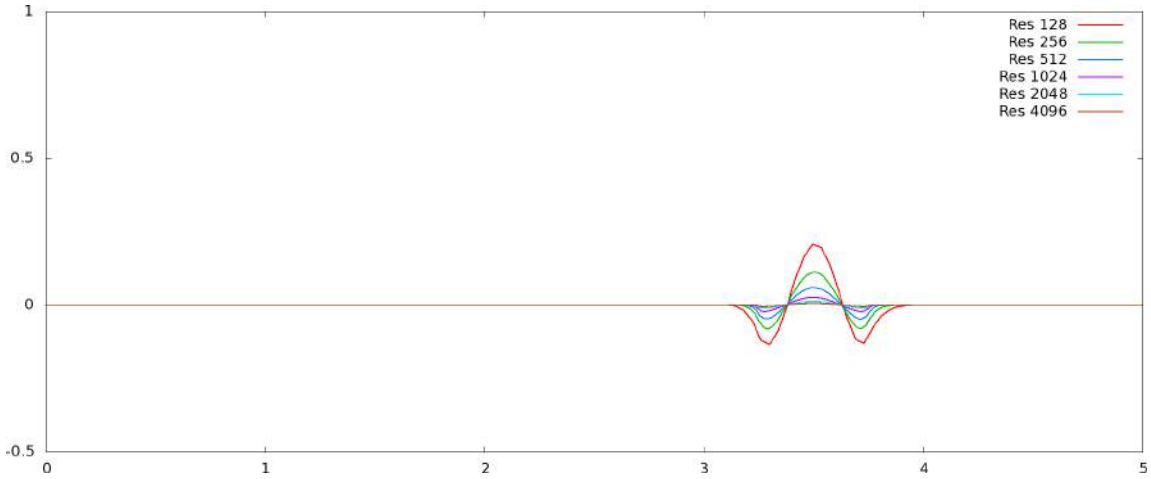


(a) Linear interpolation.

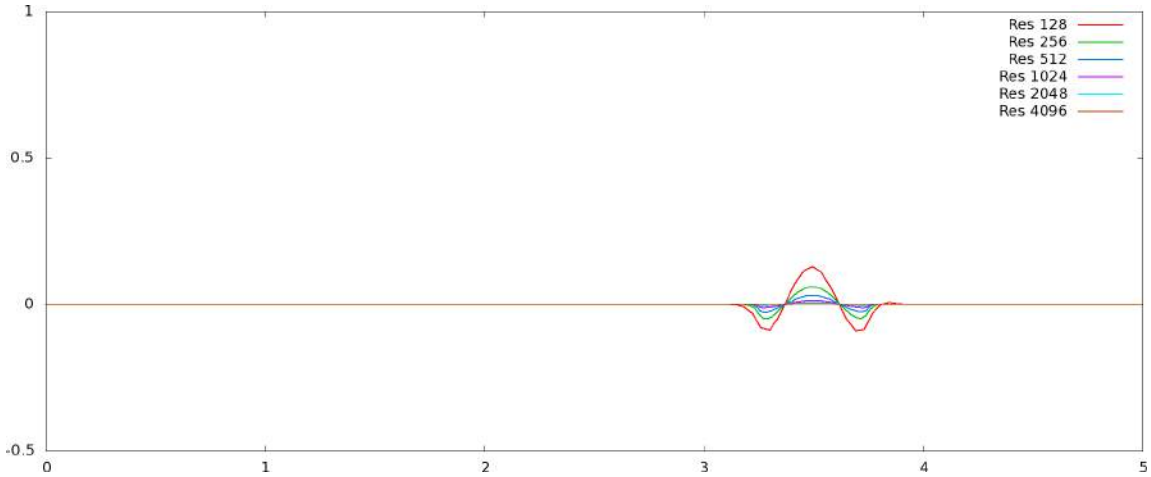


(b) Quadratic interpolation.

Figure 4.3: Error curve for the advected sine-wave "bump" in a constant velocity field $u = 1$ at time $t = 3s$ for linear and quadratic interpolation using our proposed conservative semi-Lagrangian advection scheme, run with a CFL number .9. Using a higher-order interpolation scheme gives noticeably reduced error; for example at $\Delta x = 5/256$ the peak error for the linear interpolation scheme is .111, while the quadratic interpolation scheme has a peak error of .060.

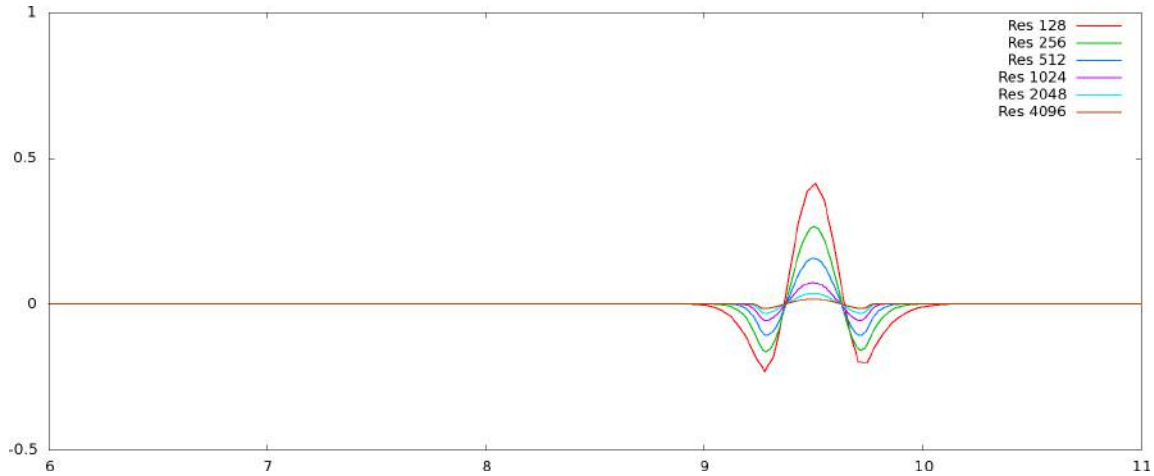


(a) Linear interpolation.

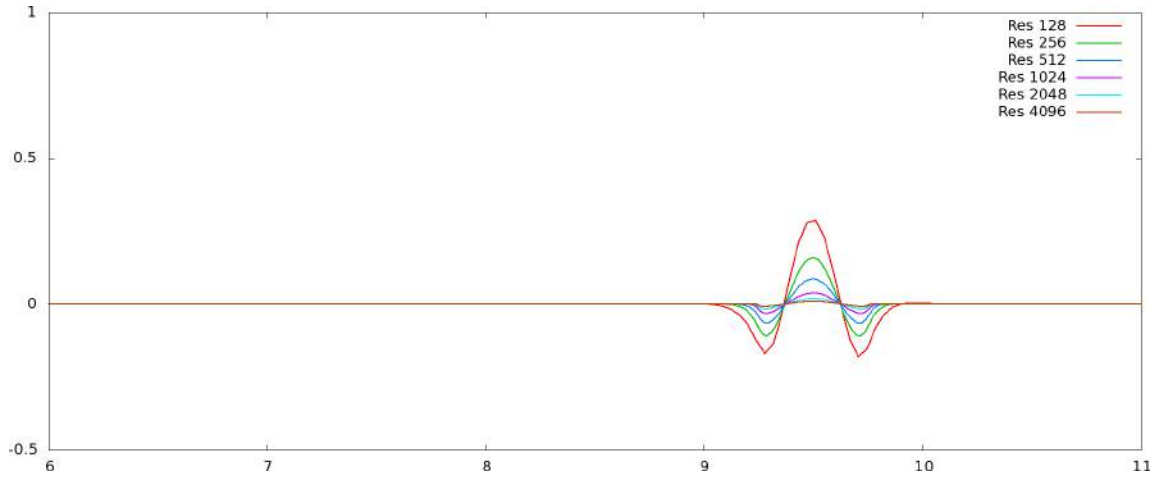


(b) Quadratic interpolation.

Figure 4.4: Error curve for the advected sine-wave "bump" in a constant velocity field $u = 1$ at time $t = 3s$ for linear and quadratic interpolation using our proposed conservative semi-Lagrangian advection scheme, run with a CFL number 2.9. As there are no temporal errors (as any semi-Lagrangian ray exactly captures the characteristic curve), all errors are due to the application of an interpolation scheme. The larger CFL number permits time steps almost three times larger than those taken for Figure 4.3, and so the error introduced by the interpolation scheme are significantly smaller.

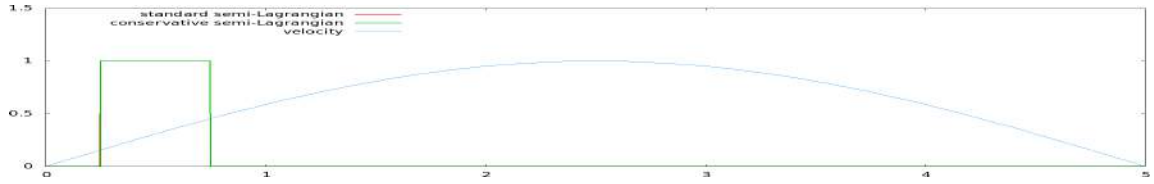


(a) Linear interpolation.

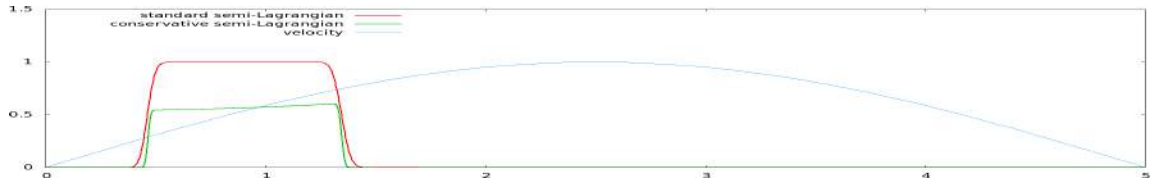


(b) Quadratic interpolation.

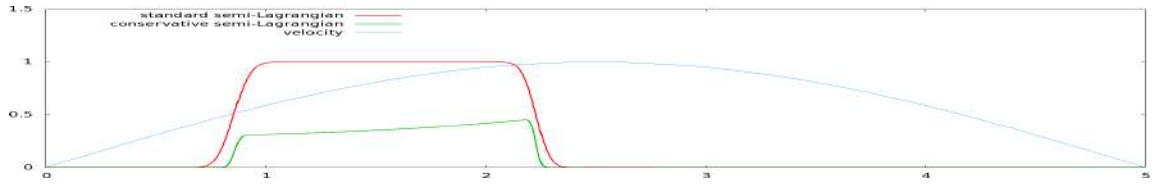
Figure 4.5: Error curve for the advected sine-wave "bump" in a constant velocity field $u = 1$ at time $t = 9s$ for linear and quadratic interpolation using our proposed conservative semi-Lagrangian advection scheme, run with a CFL number 2.9. As there are no temporal errors (as any semi-Lagrangian ray exactly captures the characteristic curve), all errors are due to the application of an interpolation scheme. As such the number of interpolations needed decreases as the CFL number increases, and the error goes down proportionally. If we run the same simulation with a larger CFL number and a proportionally longer period of time, the errors become similar (see Figure 4.3).



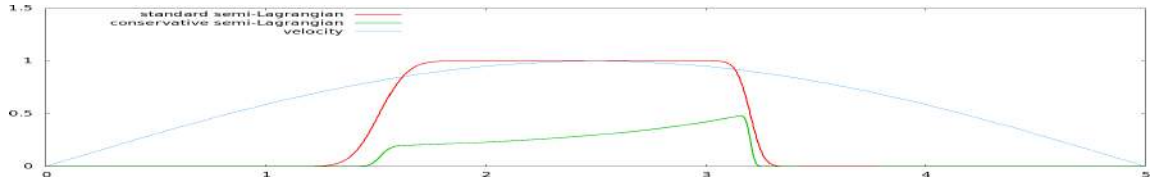
(a) $t = 0s$.



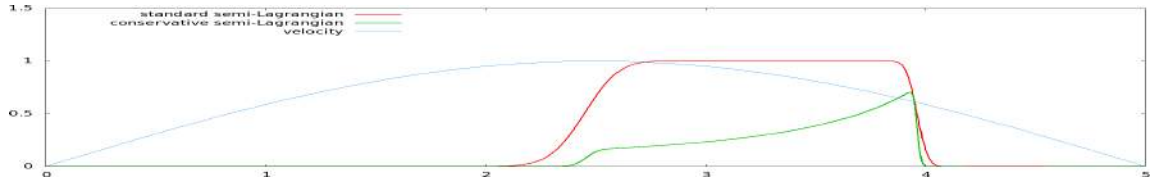
(b) $t = 1s$.



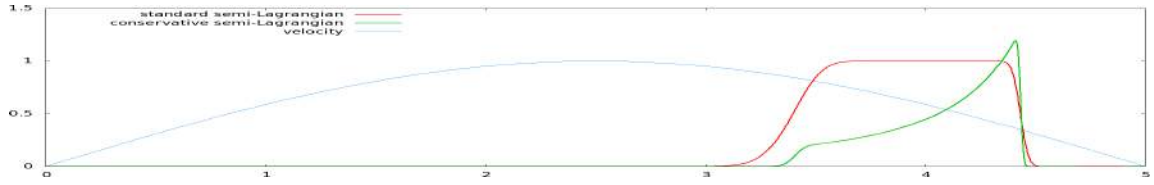
(c) $t = 2s$.



(d) $t = 3s$.



(e) $t = 4s$.

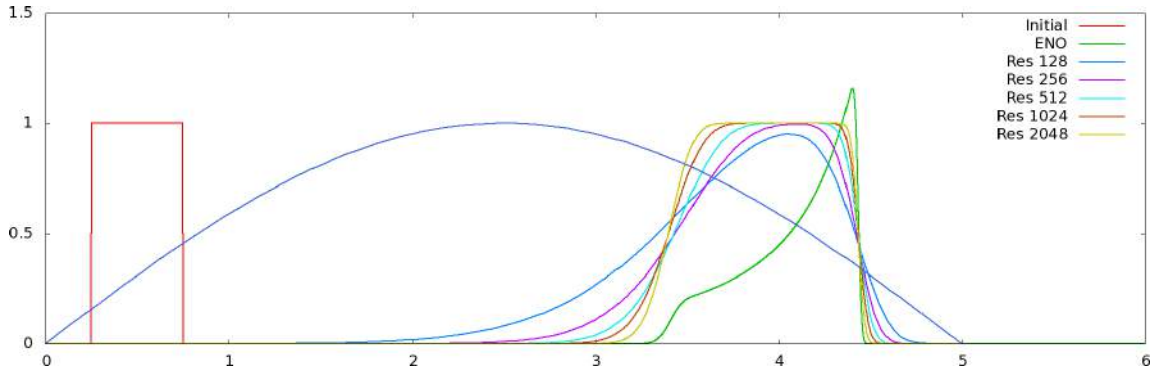


(f) $t = 5s$.

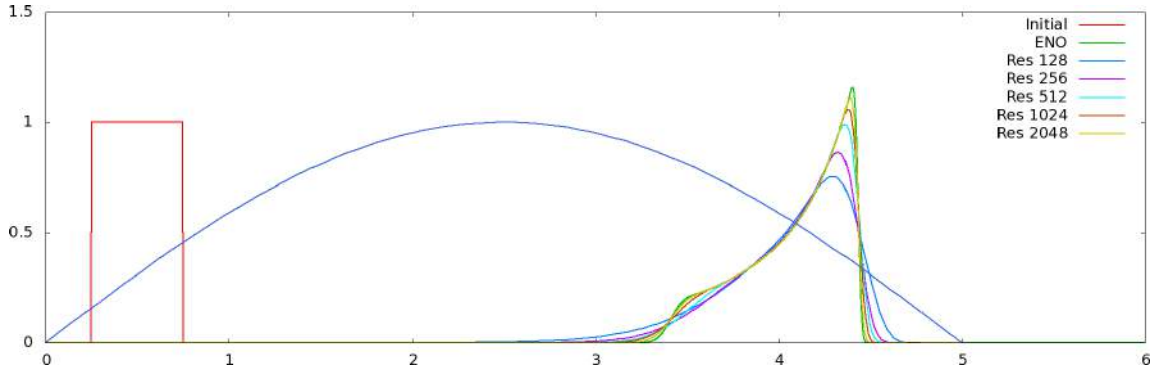
Figure 4.6: We consider the evolution of density in a velocity field that is specified by $u(x) = \sin\left(\pi\frac{x}{5}\right)$. In such a velocity field, the standard semi-Lagrangian approach fails to capture the rarefaction and converges to a non-physical solution. This simulation is run with $\Delta x = 5/8192$.

Coarse Res	Fine Res	Convergence Order
128	256	0.9384
256	512	1.2062
512	1024	1.1614
1024	2048	1.1260
2048	4096	0.9938

Table 4.1: Convergence order is computed by taking the $log_2(c_e/f_e)$ where c_e is the error in the coarse resolution simualtion and f_e is the error in the fine resoltuion simulation. The order is averaged over all relevant points.



(a) Standard semi-Lagrangian advection.



(b) Conservative semi-Lagrangian advection.

Figure 4.7: A square wave that evolves with a divergent velocity field $u = \sin\left(\pi\frac{x}{5}\right)$. Shown is the solution at time $t = 3s$. We apply the first order version of both the standard semi-Lagrangian advection, as well as our proposed conservative semi-Lagrangian advection scheme. In this example, we see the standard semi-Lagrangian advection scheme converges to the wrong solution.

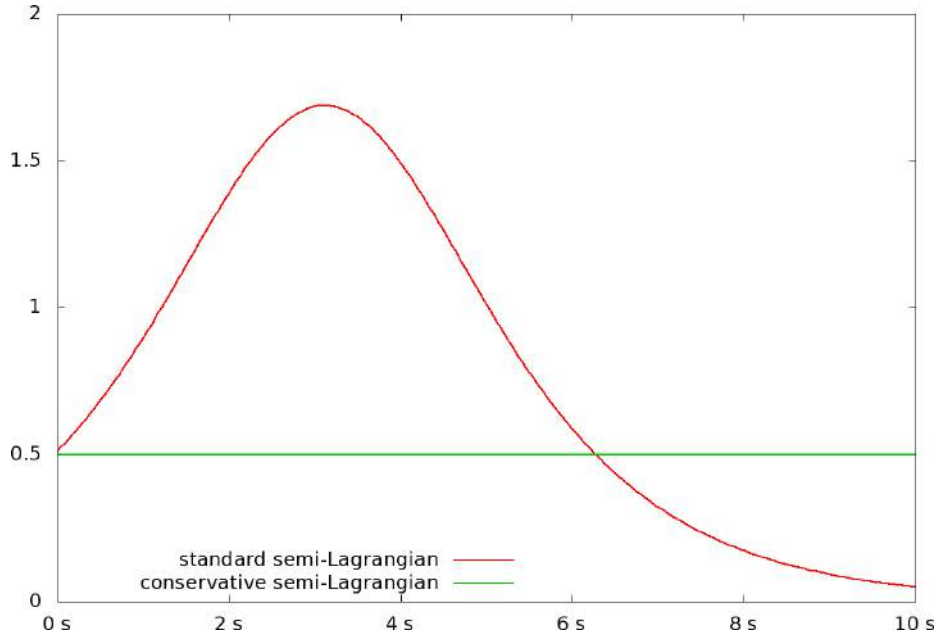


Figure 4.8: Shown is the time history of $\sum_i \Delta x \hat{\phi}_i$ for a square wave that is evolved through a divergent velocity field with $u = \sin\left(\pi \frac{x}{5}\right)$. Solutions for both the standard semi-Lagrangian advection scheme and our proposed conservative semi-Lagrangian advection scheme are shown at high-resolution with $\Delta x = 5/8192$.

Shown in Figure 4.9 is the disk after one rotation, for a variety of resolutions. We also plot the total mass of the system as a function of time, in Figure 4.10; note that a standard semi-Lagrangian scheme fails to conserve the mass of the disk. The conservative semi-Lagrangian scheme conserves the mass of the disk up to roundoff error.

4.3 Incompressible flow

We model incompressible flow using the viscous Navier-Stokes equations, given by

$$\begin{cases} \vec{u}_t + \vec{u} \cdot \nabla \vec{u} + \frac{\nabla p}{\rho} = \frac{1}{\rho} \nabla \cdot (\mu \nabla \vec{u}) \\ \nabla \cdot \vec{u} = 0 \end{cases} \tag{4.7}$$

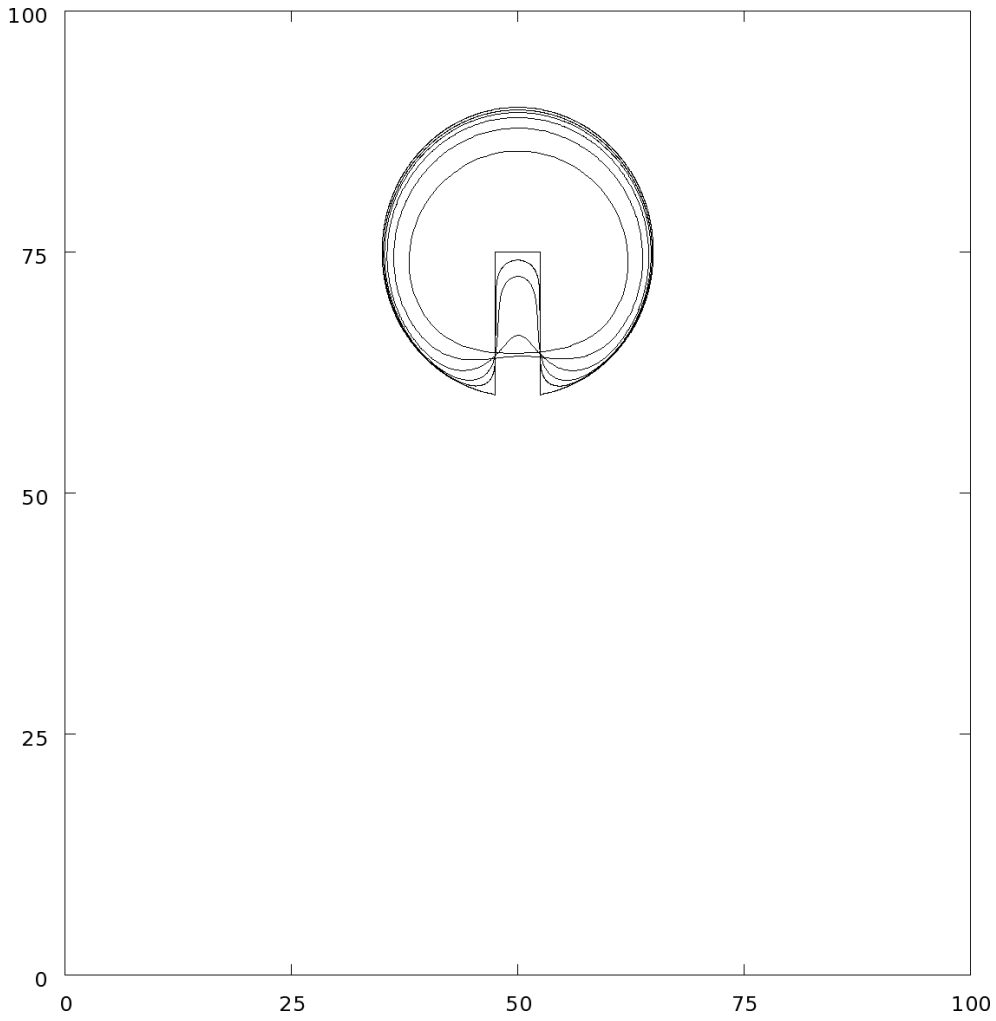


Figure 4.9: After one full rotation of the Zalesak disk [110] using our proposed conservative semi-Lagrangian advection scheme, for a variety of grid resolutions. Shown is the .5 isocontour for grid resolutions $\Delta x = 2^{-7}$, 2^{-8}, 2^{-9}, 2^{-10}, and 2^{-11}, in addition to the analytic solution. The mass of the disk is properly conserved using our method (this is verified in Figure 4.10), while the standard semi-Lagrangian advection scheme loses significant mass. In this light, our scheme can be thought of as the conservative advection of a smeared-out Heaviside color function.

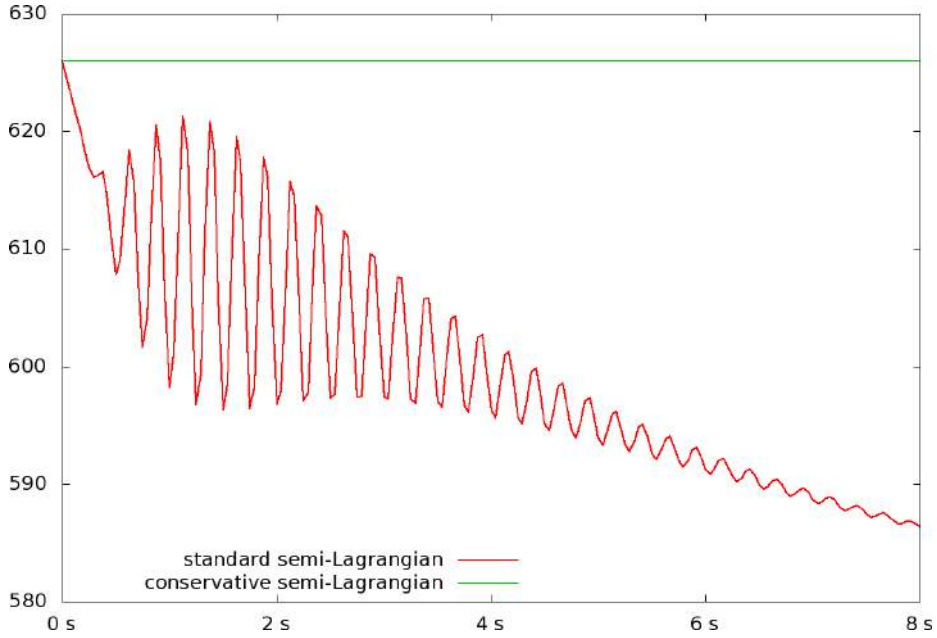


Figure 4.10: Shown is the time history of $\Sigma_i \Delta x \hat{\phi}_i + \Sigma_{out} - \Sigma_{in}$ for Zalesak Disk with $\Delta x = 2^{-7}$. Time history for the standard semi-Lagrangian advection scheme is shown in red, while our proposed conservative semi-Lagrangian advection scheme is shown in green.

where $\vec{u}$ is the fluid velocity, p is the pressure and μ is the coefficient of viscosity (which is taken to be constant). For the sake of illustration, we use a fairly simple time discretization scheme. First we account for the $\vec{u} \cdot \nabla \vec{u}$ term by advecting $\vec{u}^n$ forward in time using the incompressible velocity field $\vec{u}^n$ with a semi-Lagrangian advection scheme, giving an advected velocity $\vec{u}^\star$. This velocity field is projected and made incompressible by solving

$$\Delta t \nabla \cdot \frac{1}{\rho} \nabla p = \nabla \cdot \vec{u}^\star \tag{4.8}$$

to obtain a pressure, which is then applied via:

$$\vec{u}^{\star\star} = \vec{u}^\star - \frac{\Delta t}{\rho} \nabla p. \tag{4.9}$$

Viscous forces are next implicitly accounted for by solving

$$\tilde{\vec{u}}^{n+1} = \vec{u}^{\star\star} + \frac{\Delta t}{\rho}\nabla \cdot (\mu \nabla \tilde{\vec{u}}^{n+1}), \tag{4.10}$$

after which we project the flow field again by solving Equation (4.8) (replacing $\vec{u}^{\star}$ with $\tilde{\vec{u}}$), and then finally updating the flow field to time t^{n+1} via

$$\vec{u}^{n+1} = \tilde{\vec{u}}^{n+1} - \frac{\Delta t}{\rho}\nabla p. \tag{4.11}$$

A standard Marker and Cell (MAC, [33]) grid discretization is used, storing fluid velocity in a component-by-component fashion on cell faces. By treating the viscous forces implicitly, we alleviate the viscous time step restriction.

4.3.1 Momentum-conserving scheme

In order to derive a completely conservative scheme for the momentum, we reformulate the incompressible flow equations slightly. First, we multiply Equation (4.7) through by density, giving the following equations in two spatial dimensions:

$$\rho u_t + \rho u u_x + \rho v u_y + p_x = (\mu u_x)_x + (\mu u_y)_y, \tag{4.12}$$

$$\rho v_t + \rho v u_x + \rho v v_y + p_y = (\mu v_x)_x + (\mu v_y)_y. \tag{4.13}$$

Next, we make use of conservation of mass, given in two spatial dimensions as $\rho_t + (\rho u)_x + (\rho v)_y = 0$, noting that for incompressible flow this is identical to $\rho_t + u\rho_x + v\rho_y = 0$. If we combine this with the equations above, we can introduce the momentum $L_u = \rho u$, $L_v = \rho v$ and derive the conservation form of the incompressible flow equations as

$$(L_u)_t + (L_u u)_x + (L_u v)_y + p_x = (\mu u_x)_x + (\mu u_y)_y, \tag{4.14}$$

$$(L_v)_t + (L_v u)_x + (L_v v)_y + p_y = (\mu v_x)_x + (\mu v_y)_y. \tag{4.15}$$

For advection we solve $(L_u)_t + (L_u u)_x + (L_u v)_y = 0$ for $L_u^{\star}$ using our new conservative semi-Lagrangian scheme. Similarly, $(L_v)_t + (L_v u)_x + (L_v v)_y = 0$ is solved for

$L_v^\star$. This small change in form of the equations yields an advection scheme which is robust to the numerical viscosity effects typically seen in a semi-Lagrangian advection solver.

We use the standard pressure update to compute the pressure, where the intermediate velocity field is computed as $u^\star = L_u^\star/\rho$ and $v^\star = L_v^\star/\rho$. Equation (4.9) and (4.12), (4.13), (4.14), (4.15) illustrate that the pressure already acts as a conservative momentum flux between fluid cells. For fluid cells which lie along the fluid-structure interface, pressure acts as a momentum flux from the fluid cell faces to the solid, and vice versa. Thus, after projection we can simply update our x-momentum as $L_u^{\star\star} = \rho u^{\star\star}$ and y-momentum as $L_v^{\star\star} = \rho v^{\star\star}$, after applying the correction defined in Equation (4.9) to the velocity field $\vec{u}^\star$.

The viscous terms are treated implicitly by solving $\frac{\rho\tilde{u}^{n+1} - \rho\tilde{u}^{\star\star}}{\Delta t} = (\mu\tilde{u}_x^{n+1})_x + (\mu\tilde{u}_y^{n+1})_y$ which for constant density and viscosity becomes

$$\tilde{L}_u^{n+1} = L_u^{\star\star} + \Delta t\mu\nabla^2\tilde{u}^{n+1}, \tag{4.16}$$

similar to Equation (4.10) above. In order to properly account for momentum transfer during the viscous stage, we are careful to view this viscosity update in a flux-based form. That is, μu_x acts as a momentum flux in between the MAC grid stencil locations of u values, and μu_y acts as a momentum flux in between MAC grid u stencil locations in the other direction. The same approach is used to update v velocities, using μv_x and μv_y as momentum fluxes between MAC grid v stencil locations. This gives

$$\tilde{L}_v^{n+1} = L_v^{\star\star} + \Delta t\mu\nabla^2\tilde{v}^{n+1}. \tag{4.17}$$

These intermediate quantities are again projected by solving Equation (4.8) (replacing $\vec{u}^\star$ with $\tilde{\vec{u}}^{n+1}$). The time t^{n+1} velocity field is computed using Equation (4.11), and momentum is updated as $L_u^{n+1} = \rho u^{n+1}$ and $L_v^{n+1} = \rho v^{n+1}$.

4.3.2 Examples

We consider a cavity with high viscous forces that is driven by a flat, horizontal velocity profile with magnitude 1m/s on the top boundary of the domain. All walls in the domain are closed, the computational domain is $1m \times 1m$ with $\Delta x = .01$, and a driving flow moving at speed 1 m/s. Density is 1, and the viscous forces are determined by $\mu = 100\ Pa \cdot s$. Viscosity causes a vortex to form in the cavity, which quickly settles to steady-state. The resulting steady-state solutions are shown in Figure 4.11 for the standard semi-Lagrangian advection scheme and our momentum-conserving semi-Lagrangian advection scheme. Examining the pressure along the internal boundary, it is interesting to note that *both* schemes produce 0 net force acting on the solid boundary (i.e. $\sum_{\partial\Omega} p d\vec{A} = 0$), but the magnitude of the force isn't (i.e. $\sum_{\partial\Omega} |p| \neq 0$), suggesting that we properly capture linear momentum but angular momentum remains an issue.

Next we consider the simple case of flow past a sphere with closed walls on the top and bottom of the domain, inflow velocity with magnitude 2 m/s from the left side of the domain and an open outflow boundary on the right side of the domain with $p = 0$. For this example we used a domain of $(0, 0) \times (2, 1)$ (in meters), no viscosity and a grid size defined by $\Delta x = .01$. The solution at time $t = 9s$ is shown in Figure 4.12, using our proposed momentum-conserving scheme. The results of a standard semi-Lagrangian scheme are qualitatively (but not quantitatively) similar as expected.

We also carry out a detailed study of the momentum, for both our scheme and the standard semi-Lagrangian scheme. The bottom two lines in Figure 4.13 show the cumulative momentum advected into and out of our of the domain for the semi-Lagrangian scheme, while the middle two lines show these same quantities for our momentum-conserving scheme. Since the flow is divergence free, one would generally expect these lines to be commensurate, however, due to numerical errors in interpolation there is some drift, which accumulates as the simulation carries forward. As pointed out above, the pressure acts as a conservative flux between fluid velocity degrees of freedom. Along solid wall boundaries, such as the top and bottom of the domain and around the sphere, the pressure can be scaled by the cell face

size and Δt to give an impulse, suggesting that it represents a momentum-preserving collision between the solid and the fluid. However, since these walls are stationary, i.e. they have infinite mass, they remove momentum from the flow. If we sum the momentum lost along the walls we obtain the bottom two lines shown in Figure 4.14 for the semi-Lagrangian scheme and our momentum-conserving scheme respectively. Momentum is also introduced into the domain via pressure at the inflow boundary condition, where an upstream pressure profile is used to maintain a constant inflow velocity. The gains due to inflow are shown in Figure 4.14 for the semi-Lagrangian method on the top curve, and the second curve from the top is for our momentum-conserving method. Note that since $p = 0$ at the outflow boundary, no momentum is lost. Figure 4.15 shows the result if we sum the previous graphs accounting for all the momentum advected into and out of the domain as well as the pressure fluxes at the walls and inflow. The bottom line, which is 0 to numerical roundoff, shows that our momentum-conserving scheme does indeed conserve momentum during the semi-Lagrangian advection step (which is the only term not accounted for in the previous graphs). In contrast, the standard semi-Lagrangian scheme gains momentum during the advection step. Although we did not compute the momentum gained by carefully looking at that step, we can accurately compute it by accounting for all the remaining terms and seeing what is left over.

4.4 Treating kinetic energy

It is interesting to consider incompressible flow from the standpoint of kinetic energy. Although kinetic energy is not conserved for a viscous fluid, it *is* conserved for an inviscid fluid. Moreover, it should be conserved during the semi-Lagrangian advection stage, even though it typically is not. We begin by deriving the time derivative of $K = \frac{1}{2}\rho\vec{u}\cdot\vec{u}$ as

$$\begin{aligned} K_t &= \frac{1}{2}\left(\rho\vec{u}\cdot\vec{u}\right)_t = \frac{1}{2}\vec{u}\cdot\vec{u}\rho_t + \rho\vec{u}\cdot\vec{u}_t \\ &= \frac{1}{2}\vec{u}\cdot\vec{u}\left(-\vec{u}\cdot\nabla\rho\right) + \vec{u}\cdot\left(\nabla\cdot\tau - \rho\vec{u}\cdot\nabla\vec{u} - \nabla p\right). \end{aligned}$$

We take advantage of Equation (4.7) from above, and observe that

$$\begin{aligned}\frac{1}{2}(\vec{u}\cdot\vec{u})\vec{u}\cdot\nabla\rho+\rho\vec{u}\cdot(\vec{u}\cdot\nabla\vec{u}) &= \frac{1}{2}(\vec{u}\cdot\vec{u})\vec{u}\cdot\nabla\rho+\frac{1}{2}\rho\vec{u}\cdot\nabla\left[\vec{u}\cdot\vec{u}\right]\\ &= \frac{1}{2}\vec{u}\cdot\nabla\left[\rho\vec{u}\cdot\vec{u}\right]=\vec{u}\cdot\nabla K\\ &= \nabla\cdot(K\vec{u}).\end{aligned}$$

Note that we can freely add in $p\nabla\cdot\vec{u}$ and $K\nabla\cdot\vec{u}$, which are both analytically zero. This gives time evolution of kinetic energy in conservative form as

$$K_t+\nabla\cdot[(K+p)\vec{u}]=\vec{u}\cdot(\nabla\cdot\tau). \tag{4.18}$$

4.4.1 Advection

We compute and store kinetic energy on horizontal u faces as $K_u=\frac{1}{2}\rho u^2$, and at vertical v faces as $K_v=\frac{1}{2}\rho v^2$, and evolve them forward in time separately as they only couple together through pressure fluxes, similar to the advection of the velocity field.

For advection, we solve $(K_u)_t+(K_uu)_x+(K_uv)_y=0$ for $K_u^\star$ and $(K_v)_t+(K_vu)_x+(K_vv)_y=0$ for $K_v^\star$, using the time t^n velocity field $\vec{u}^n$. In doing so, we explicitly conserve the kinetic energy of the system during the advection step, which has the effect of relieving the artificial viscosity effects typically seen when using a standard semi-Lagrangian advection scheme.

Once we compute $K^\star$ advected quantities, we use these kinetic energies to determine the magnitudes of the intermediate fluid velocity field $\vec{u}^\star$. That is, $u^\star=\pm\sqrt{2K_u^\star/\rho}$ and $v^\star=\pm\sqrt{2K_v^\star/\rho}$. We also advect fluid velocities forward in time (using either the semi-Lagrangian scheme or the momentum-conserving scheme) and use the sign of the resulting velocity field to determine the sign of $u^\star$ and $v^\star$.

4.4.2 Projection

The modified $\vec{u}^\star$ values are used in Equation (4.8) to compute the fluid pressure. Unlike the momentum update, where the pressure itself acts as a momentum flux

and the result of the projection does conserve momentum, for kinetic energy we not only don't have good values for the flux $p\vec{u}$ in Equation (4.18), but the resulting post-velocity projection does not have the same kinetic energy as the pre-projected velocity. Indeed, the change in kinetic energy due to the projection is

$$\begin{aligned}\Delta K_u &= \frac{\rho}{2}\left((u^{\star\star})^2 - (u^{\star})^2\right) \\ &= \frac{\rho}{2}\left(u^{\star\star} + u^{\star}\right)\left(u^{\star\star} - u^{\star}\right) \\ &= \frac{\rho}{2}\left(u^{\star\star} + u^{\star}\right)\left(-\frac{\Delta t}{\rho}p_x\right) \\ &= -\Delta t\hat{u}\left(p_x\right).\end{aligned} \qquad \begin{aligned}\Delta K_v &= \frac{\rho}{2}\left((v^{\star\star})^2 - (v^{\star})^2\right) \\ &= \frac{\rho}{2}\left(v^{\star\star} + v^{\star}\right)\left(v^{\star\star} - u^{\star}\right) \\ &= \frac{\rho}{2}\left(v^{\star\star} + v^{\star}\right)\left(-\frac{\Delta t}{\rho}p_y\right) \\ &= -\Delta t\hat{v}\left(p_y\right).\end{aligned}$$

where $\hat{u} = \frac{u^{\star\star}+u^{\star}}{2}$ and $\hat{v} = \frac{v^{\star\star}+v^{\star}}{2}$, and we use Equation (4.9) to replace $(u^{\star\star} - u^{\star})$ and $(v^{\star\star} - v^{\star})$ terms respectively. Then ΔK_u and ΔK_v look like $\Delta t\hat{u}p_x$ and $\Delta t\hat{v}p_y$ respectively, overall accounting for the $\vec{u}\cdot\nabla p$ component of $\nabla\cdot(p\vec{u})$. Analytically we would expect this to be sufficient in an incompressible flow, as $p\nabla\cdot\vec{u} = 0$, but examining this update from the discrete standpoint we note that $\nabla\cdot\hat{\vec{u}} = \frac{1}{2}\nabla\cdot\vec{u}^{\star} + \frac{1}{2}\nabla\cdot\vec{u}^{\star\star} = \frac{1}{2}\nabla\cdot\vec{u}^{\star} \neq 0$ in general and some kinetic energy is lost. If we examine the sum of the terms of the update for cell faces $i-1/2$ and $i+1/2$,

$$\hat{u}_{i+1/2}\frac{p_{i+1}-p_i}{\Delta x} + \hat{u}_{i-1/2}\frac{p_i - p_{i-1}}{\Delta x},$$

we can rearrange terms slightly, giving

$$\frac{\hat{u}_{i+1/2}p_{i+1}}{\Delta x} - \boxed{p_i\frac{\hat{u}_{i+1/2}-\hat{u}_{i-1/2}}{\Delta x}} - \frac{\hat{u}_{i-1/2}p_{i-1}}{\Delta x},$$

where the boxed term, when summed over all spatial dimensions for cell i, gives a discrete approximation of $-p\nabla\cdot\hat{\vec{u}}$. That is, by performing an update using ΔK_u and ΔK_v, we are losing exactly this component of the flux. If we view each individual component of the boxed term, $p_i u_{i+1/2}/\Delta x$, these can be thought of as fluxes between cell *face* $i+1/2$ and cell *center* i; then the kinetic energy that has been lost in this update is precisely the kinetic energy that accumulates to a cell center, rather than

being fully distributed to our degrees of freedom.

Various strategies can be taken to address this, such as taking the accumulated cell-centered kinetic energy and distributing it equally to all of the surrounding cell faces. We plan on looking into this more in future work [51], but for now we accept the loss of kinetic energy due to projection and incorporate the change in kinetic energy by simply using ΔK_u and ΔK_v as computed above.

If we consider the pressure at a grid cell i and scale it up by the area of a cell face and Δt, we get the impulse $\hat{p}_i$ between dual-cells $i - 1/2$ and $i + 1/2$. In multiple spatial dimensions, this impulse couples together the orthogonal directions, involving every cell face incident to cell i. This impulse exchange can be thought of as a collision between neighboring dual cells. Along this line of reasoning, it is interesting to note that while collisions preserve momentum and total energy, they do *not* conserve kinetic energy unless the coefficient of restitution is 1. Typically in a collision kinetic energy is lost, and the collisions in this system—with one exception—are no different. In the special case where $(\nabla \cdot \vec{u}^\star)_i = 0$, then the multi-dimensional collision that occurs at cell i does indeed conserve kinetic energy, and can be thought of as a fully elastic collision with a coefficient of restitution equal to 1.

For the momentum update, the application of these collision-based impulses can be done in any order; that is, we can freely iterate over impulses, updating the momentum by applying impulses in a Gauss-Seidel manner. This is not the case for the energy update, as the application of one impulse changes the energy updated by all subsequent impulses due to the cross-terms which arise. If we let $\rho = \frac{m}{\Delta x \Delta y}$, then the update takes the form

$$\hat{u}^{new} = \hat{u}^{old} + \frac{\hat{p}_i}{m}, \tag{4.19}$$

where $\hat{p}_i$ is the impulse defined above. If we consider the change in kinetic energy

after these updates,

$$\begin{aligned}
\Delta KE &= \frac{1}{2}m(\hat{u}^{newer})^2 - \frac{1}{2}m(\hat{u}^{old})^2 \\
&= \frac{1}{2}m\left[\left(\hat{u}^{old} + \frac{\hat{p}_i}{m} - \frac{\hat{p}_{i+1}}{m}\right)^2 - (\hat{u}^{old})^2\right] \\
&= \frac{1}{2}m\left[(\hat{u}^{old})^2 + \hat{u}^{old}\left(\frac{\hat{p}_i}{m} - \frac{\hat{p}_{i+1}}{m}\right) + \left(\frac{\hat{p}_i}{m} - \frac{\hat{p}_{i+1}}{m}\right)^2 - (\hat{u}^{old})^2\right] \\
&= \hat{u}^{old}\left(\frac{\hat{p}_i - \hat{p}_{i+1}}{2}\right) + \frac{1}{2m}\left(\hat{p}_i^2 + \hat{p}_{i+1}^2\right) - \boxed{\frac{\hat{p}_i\hat{p}_{i+1}}{m}},
\end{aligned}$$

then the boxed term is the cross-term which arises from the sequential application of impulse updates to the fluid volume. Note that the result is the same regardless of which impulse $\hat{p}_i$ or $\hat{p}_{i+1}$ is applied first. However, one might misconstrue the gain in kinetic energy due to each impulse depending on the order in which they were applied.

4.4.3 Viscosity

After the projection in Equation (4.9) we compute the viscous forces via Equation (4.10) and then compute the kinetic energy as seen by the fluid velocity field:

$$\begin{aligned}
\Delta K &= \frac{\rho}{2}\left((\tilde{u}^{n+1})^2 - (u^{\star\star})^2\right) \\
&= \frac{\rho}{2}\left(u^{\star\star} + \tilde{u}^{n+1}\right)\left(\tilde{u}^{n+1} - u^{\star\star}\right) \\
&= \frac{\rho}{2}\left(u^{\star\star} + \tilde{u}^{n+1}\right)\left(\frac{\Delta t}{\rho}\nabla\cdot(\mu\nabla u^{\star\star})\right) \\
&= \Delta t\tilde{u}\nabla\cdot(\mu\nabla u^{\star\star}),
\end{aligned}$$

where $\tilde{u} = \frac{u^{\star\star}+\tilde{u}^{n+1}}{2}$ and we use Equation (4.10) to eliminate the $(\tilde{u}^{n+1} - u^{\star\star})$ term. This gives us $K^{\star\star} = \tilde{K}^{n+1} + \Delta K$, the loss of kinetic energy due to viscous effects (noting in the case of inviscid flow that $\Delta K = 0$ and $K^{\star\star} = \tilde{K}^{n+1} = K^{n+1}$). Once $K^{\star\star}$ is computed, it is projected again as discussed above.

4.4.4 Examples

We first reconsider the driven cavity case from Section 4.3.2 using our kinetic energy-conserving semi-Lagrangian advection scheme. For this simple case, we do not attempt to correct for the kinetic energy losses due to the inelastic collisions dictated by the $\hat{p}$ discussed above. That is, kinetic energy is lost during the projection step, even though we know how much is lost to each cell center, as adding this kinetic energy back into the flow field would lead to a divergent velocity field. The simple case of the driven cavity is shown in Figure 4.11, showing that the kinetic energy-conserving scheme compares well with the other two schemes. Unfortunately, for more interesting cases such as the one shown in Figure 4.12, the inability to create a divergence-free velocity field that is consistent with the kinetic energy posses an issue, and the results are qualitatively different.

In spite of that we carry out an analysis for the momentum and kinetic energy in all three schemes: the original semi-Lagrangian scheme, the momentum-conserving scheme, and the kinetic energy-conserving advection scheme, which correctly conserves kinetic energy during advection but fails to account for kinetic energy loses during projection. The reason for this quantitative analysis is to illustrate where the kinetic energy goes, in each of these schemes. We begin by considering the momentum. The top two lines in Figure 4.13 represent the momentum advected into and out of the domain across the inflow and outflow for the kinetic energy-conserving scheme. The middle two lines in Figure 4.14 account for the momentum fluxing through solid wall boundaries due to pressure, as well as the pressure flux at the inflow of the domain. For Figure 4.15, the top line is the sum that represents the momentum loss during advection. Note that this is rather large when compared to the other two schemes, in part because the advected velocity is not consistent with kinetic energy.

Finally, we consider the kinetic energy transfer of all three schemes. Figure 4.16 shows the kinetic energy advected into and out of the domain across inflow and outflow boundaries. Figure 4.17 shows the energy gained due to the pressure interacting with both the solid wall boundaries and pressure flux at the inflow boundary. Note that in this case the pressure acts as a collision between a fluid cell and a solid wall boundary, and that collisions influence both the momentum and the kinetic energy.

Similar collisions happen at the inflow, where kinematically moving ghost cells collide with our fluid domain. A new term that we didn't consider for the momentum is the loss of kinetic energy during projection, where fluid cells collide with each other in a partially elastic way, losing kinetic energy; these loses are shown in Figure 4.18 for each of the three schemes. Figure 4.19 shows the sum of all these terms discussed, leaving only losses which occur during the advection stage of our schemes. Note that our kinetic energy-conserving advection scheme does indeed conserve kinetic energy during advection, whereas both the standard semi-Lagrangian scheme as well as the momentum-conserving scheme lose kinetic energy in this step.

4.5 Compressible flow

We model compressible flow using the inviscid Euler equations,

$$\begin{pmatrix} \rho \\ \rho\vec{u} \\ E \end{pmatrix}_t + \begin{pmatrix} \nabla \cdot [\rho\vec{u}] \\ \nabla \cdot [\rho\vec{u} \otimes \vec{u}] + \nabla p \\ \nabla \cdot [(E+p)\vec{u}] \end{pmatrix} = 0, \tag{4.20}$$

where ρ is the fluid density, $\rho\vec{u}$ is the momentum, $E = \rho e + \frac{1}{2}\rho\vec{u} \cdot \vec{u}$ is the total energy per unit volume and e is the internal energy per unit mass. These are solved using the splitting proposed in [50]. Defining the state vector as $\vec{U} = (\rho, \rho\vec{u}, E)^T$, the flux is split into its advective component, $F_1(\vec{U})$, and acoustic component $F_2(\vec{U})$:

$$F_1(\vec{U}) = \begin{pmatrix} \nabla \cdot [\rho\vec{u}] \\ \nabla \cdot [\rho\vec{u} \otimes \vec{u}] \\ \nabla \cdot [E\vec{u}] \end{pmatrix}, \qquad F_2(\vec{U}) = \begin{pmatrix} 0 \\ \nabla p \\ \nabla \cdot [p\vec{u}] \end{pmatrix}. \tag{4.21}$$

The method first computes $F_1(\vec{U})$ explicitly with the MENO advection scheme, which uses density-averaged velocities at cell faces, advecting the state variables to

an intermediate state $\vec{U}^\star$. That is,

$$\begin{aligned}\rho^\star &= \rho^n - \Delta t \nabla \cdot (\rho \vec{u}) \\ \rho \vec{u}^\star &= \rho \vec{u}^n - \Delta t \nabla \cdot [\rho \vec{u} \otimes \vec{u}] \\ E^\star &= E^n - \Delta t \nabla \cdot (E \vec{u}).\end{aligned}$$

Note that $\rho^\star = \rho^{n+1}$, as the first term in $F_2(\vec{U})$ is zero. Next, we examine the remaining component of the momentum equation,

$$\rho^{n+1}\vec{u}^{n+1} - \rho^{n+1}\vec{u}^\star = -\Delta t \nabla p.$$

We divide through by ρ^{n+1} and take its divergence, yielding an implicit equation for pressure:

$$\nabla \cdot \vec{u}^{n+1} - \nabla \cdot \vec{u}^\star = -\Delta t \nabla \cdot \frac{1}{\rho^{n+1}} \nabla p. \tag{4.22}$$

In order to remain conservative, we discretize $\nabla \cdot \vec{u}^\star$ by computing $\vec{u}^\star$ at faces. That is, we compute $\nabla \cdot \vec{u}^\star = -G^T \hat{\vec{u}}^\star$, where $-G^T$ is the discretized divergence operator and $\hat{\vec{u}}^\star$ are $\vec{u}^\star$ velocities averaged to faces. Then we next eliminate the $\nabla \cdot \vec{u}^{n+1}$ term by considering the pressure evolution equation (see [24]):

$$p_t + \vec{u} \cdot \nabla p + \rho c^2 \nabla \cdot \vec{u} = 0. \tag{4.23}$$

This is discretized as $p^{n+1} = p^a - \Delta t \rho^n (c^n)^2 \nabla \cdot \vec{u}^{n+1}$, where p^a is an advected p^n pressure using the $\vec{u}^n$ velocity field. Plugging this into (4.22), discretizing the gradient ∇ as G and the divergence $\nabla\cdot$ as $-G^T$ gives the following implicit pressure equation:

$$\left[I + \rho^n (c^2)^n \Delta t^2 G^T \left(\frac{1}{\hat{\rho}^{n+1}} G \right) \right] p^{n+1} = p^a + \rho^n (c^2)^n \Delta t G^T \hat{\vec{u}}^\star. \tag{4.24}$$

where $\hat{\rho}^{n+1}$ are densities averaged to cell faces.

Finally these pressures are applied to the $\vec{U}^\star$ state to get time t^{n+1} quantities. Since pressure values and momentum quantities are collocated, we average pressures to faces as $p^{n+1}_{i+1/2} = \frac{p^{n+1}_{i+1}\rho^{n+1}_i + p^{n+1}_i \rho^{n+1}_{i+1}}{\rho^{n+1}_i + \rho^{n+1}_{i+1}}$, permitting us to evaluate ∇p for the momentum

update in a flux-based manner. We also want to evaluate $p\vec{u}$ at cell faces in order to numerically conserve total energy, and so we update the $\hat{\vec{u}}^{\star}$ velocities from Equation (4.22) as $\hat{\vec{u}}^{n+1} = \hat{\vec{u}}^{\star} - \Delta t \frac{Gp^{n+1}}{(\rho_i + \rho_{i+1})/2}$. This permits us to write the numerically conservative flux-based update as

$$(\rho\vec{u})^{n+1} = (\rho\vec{u})^{\star} - \Delta t \left(\frac{p^{n+1}_{i+1/2} - p^{n+1}_{i-1/2}}{\Delta x} \right), \qquad E^{n+1} = E^{\star} - \Delta t \left(\frac{(p\hat{u})^{n+1}_{i+1/2} - (p\hat{u})^{n+1}_{i-1/2}}{\Delta x} \right). \tag{4.25}$$

In order to demonstrate our new conservative semi-Lagrangian advection, we use it to replace the MENO advection scheme when solving $F_1(\vec{U})$. Notably the method of [50] was able to stably compute the solutions of compressible flow ignoring the CFL restriction due to the acoustic wave because of the implicit treatment of pressure in Equation (4.24). However, they were still limited by a CFL restriction based on the fluid velocity. Using our unconditionally stable advection scheme, we are no longer restricted to a fluid velocity-based CFL.

4.5.1 Example

We solve the classic one-dimensional Sod shock tube [102] using the advection-based CFL condition given by

$$\frac{\Delta t}{2} \left(\frac{|u|_{max}}{\Delta x} + \sqrt{\frac{|u|^2_{max}}{\Delta x^2} + 4\frac{|p_x|}{\rho \Delta x}} \right) \leq 1.$$

as defined in [50]. The Sod shock tube takes as initial conditions

$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (1,0,1) & \text{if } x \leq .5, \\ (.125,0,.1) & \text{if } x > .5. \end{cases} \tag{4.26}$$

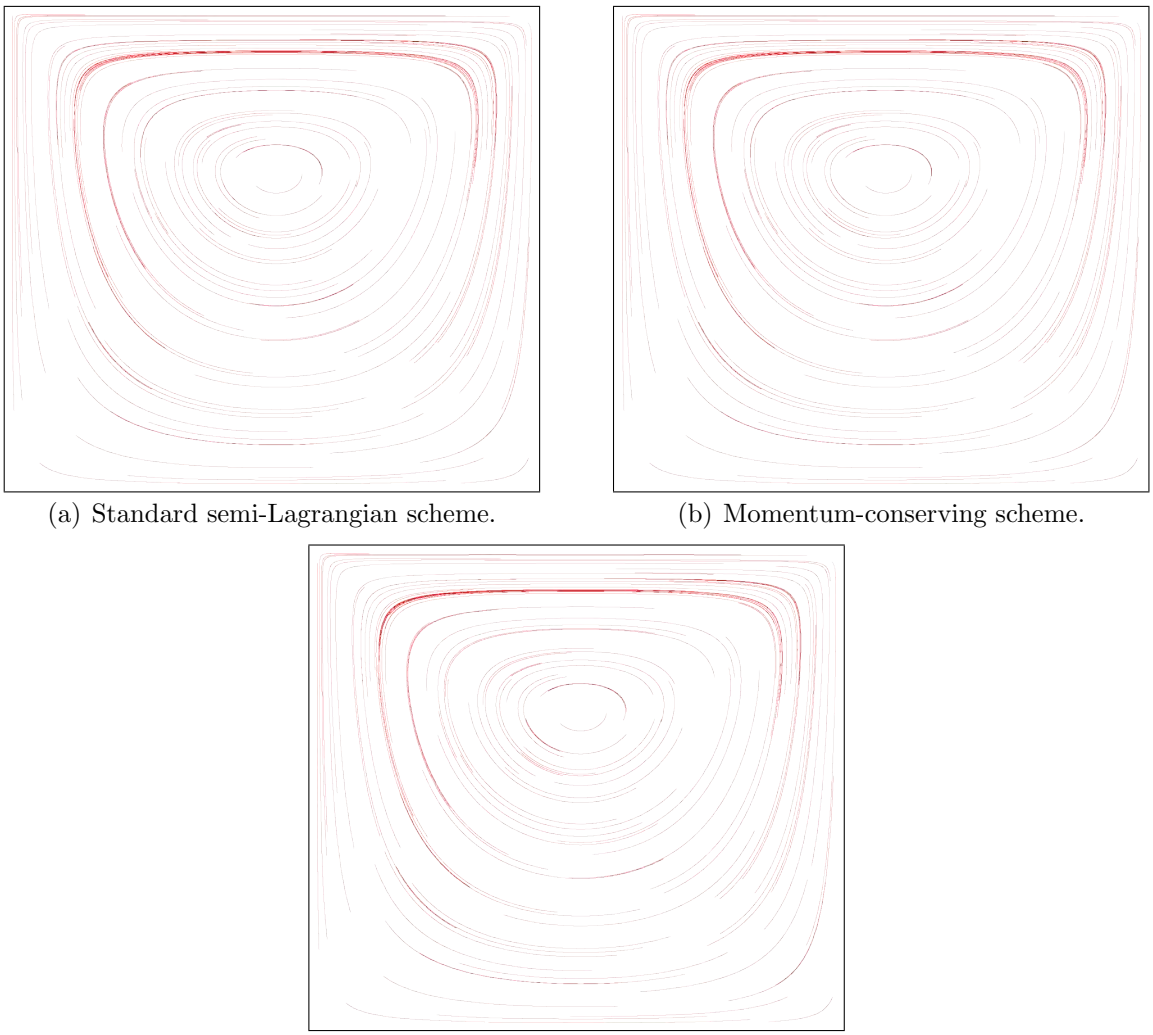
This example is solved on a computational domain of $x \in (0,1)$, with $\Delta x = 2.5 \times 10^{-3}$. We compare the results of an approach using MENO and a CFL number of .9 with the results of an approach using our conservative semi-Lagrangian advection scheme with a CFL number of 3. To illustrate convergence, we show a plot of density at

time $t = .15s$ for a selection of grid resolutions in Figure 4.20, where conservative semi-Lagrangian advection is used with the semi-implicit compressible flow solver with a CFL number of .5. Figure 4.21 shows convergence when a CFL number of 3 is used. We show the same quantities for $t = .8s$ in conservative-sl-figures 4.23 and 4.24. Each figure also shows the resulting solution when solved using a traditional, fully explicit compressible flow solver with 3rd order accuracy in time and space, for comparison. We stress that the over-shoots near the shock front are a consequence of the semi-implicit discretization of the equations discussed in [50], as the implicit pressure system is centrally biased; to illustrate this point, we show in Figure 4.22 the results generated when a third order MENO advection scheme is used instead, which suffers from these same overshoots.

Currently, in the context of compressible flows, we are working to extend our method in a fashion that hybridizes it with a flux-based scheme such as that of [93]. The goal here would be to apply high order accurate flux-based discretization in most of the domain (albeit with a restrictive CFL condition), yet apply our method near moving solid boundaries and especially thin shells, see [30].

4.6 Conclusion

We have presented a conservative, unconditionally stable semi-Lagrangian advection scheme. The method is built from simple, first order semi-Lagrangian building blocks. We show that the method is beneficial in the simulation of both incompressible and compressible flows.



(a) Standard semi-Lagrangian scheme.

(b) Momentum-conserving scheme.

(c) Kinetic energy-conserving scheme.

Figure 4.11: Streamlines for the driven cavity example using standard semi-Lagrangian advection, our proposed momentum-converging method, and our proposed kinetic energy-conserving method. All simulations are run with $\Delta x = 2^{-7}$.

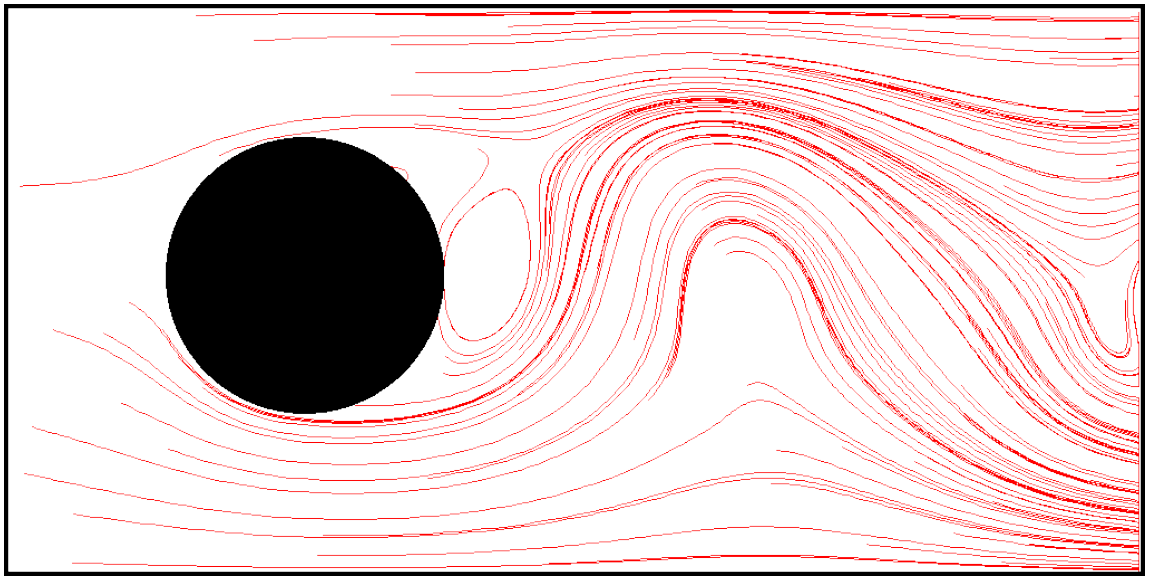


Figure 4.12: Stream-line visualization of flow past a sphere.

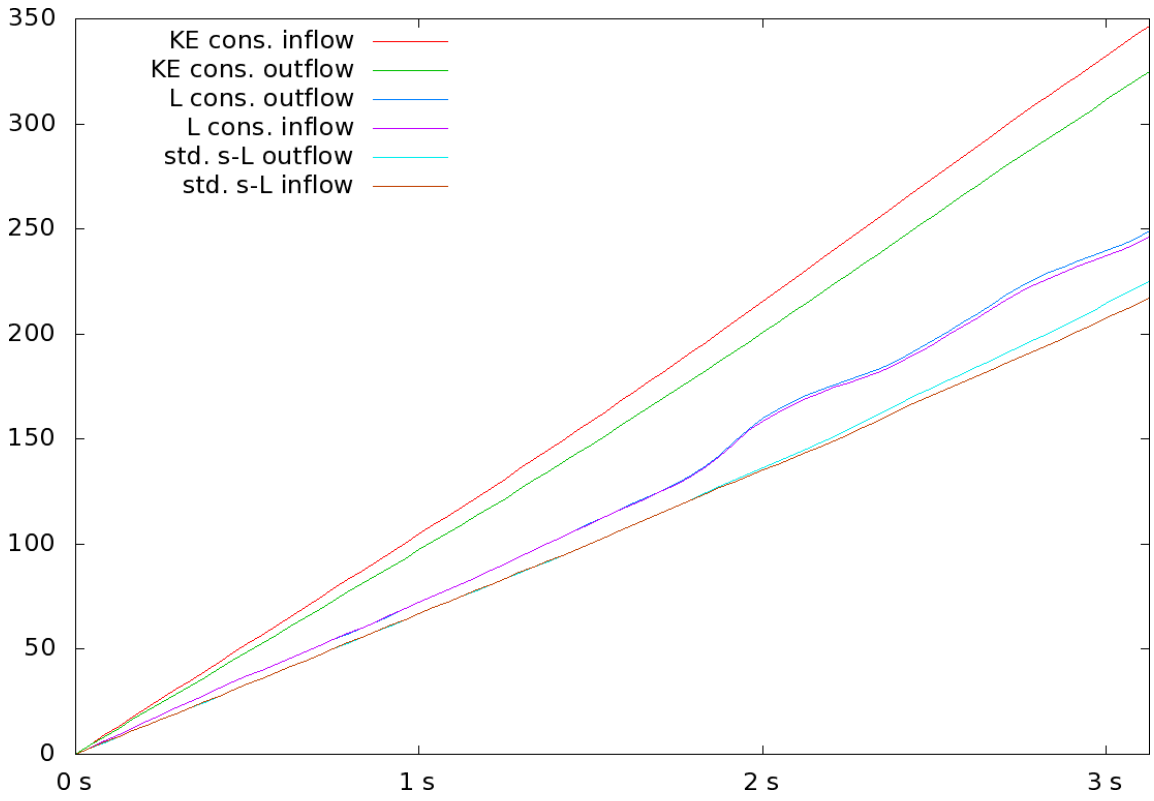


Figure 4.13: Total momentum fluxing into the computational domain and total momentum fluxing out of the computational domain, plotted as a function of time for a standard semi-Lagrangian scheme and our proposed momentum-conserving scheme.

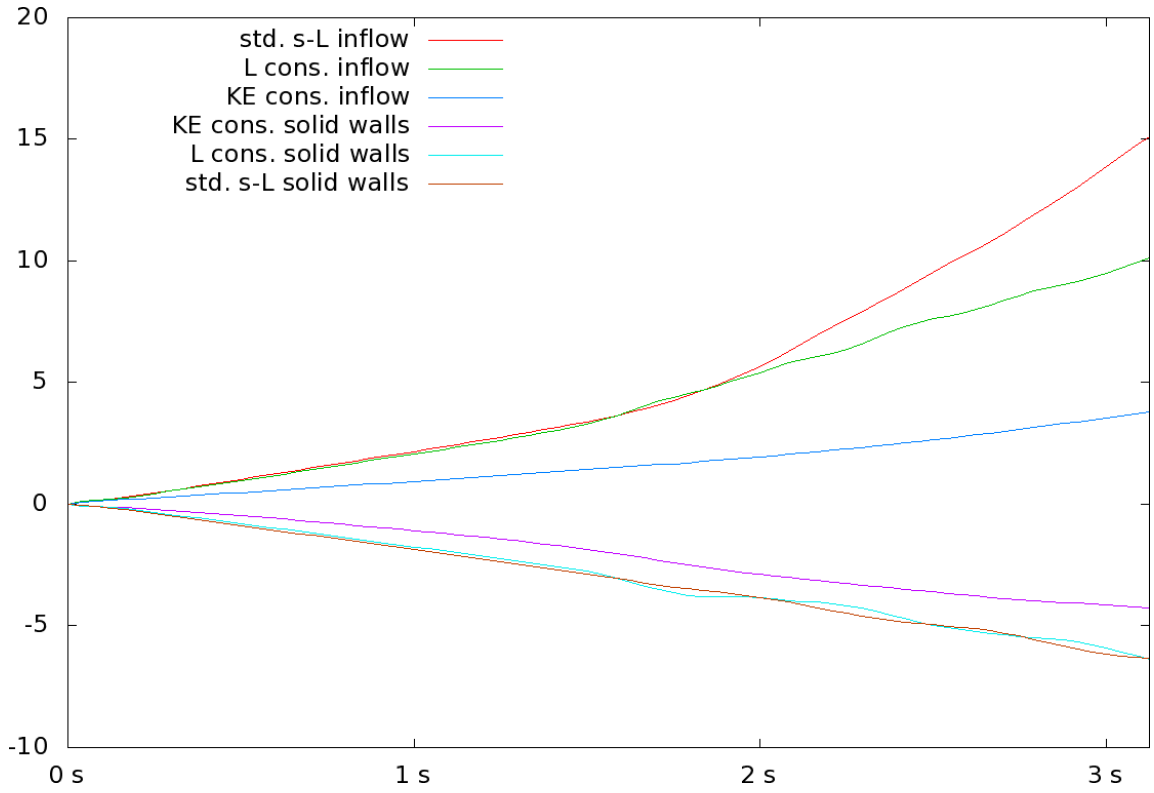


Figure 4.14: Pressure momentum flux into solid wall boundaries, and pressure momentum flux entering the computational domain from the inflow boundary condition, plotted as a function of time for a standard semi-Lagrangian scheme and our proposed momentum-conserving scheme.

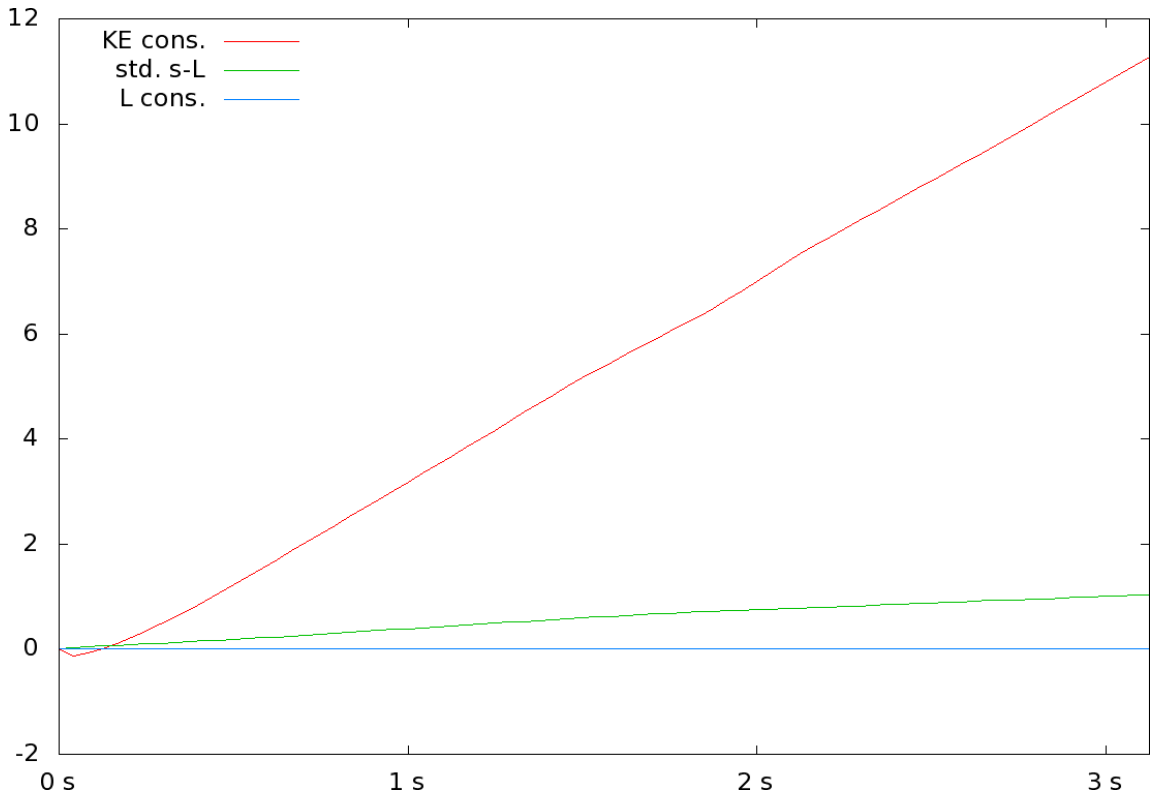


Figure 4.15: Sum total of momentum in the domain, plus momentum fluxed out of the domain (through outflow and solid wall boundaries), minus momentum fluxed into the domain (through inflow), plotted as a function of time for a standard semi-Lagrangian scheme and our proposed momentum-conserving scheme.

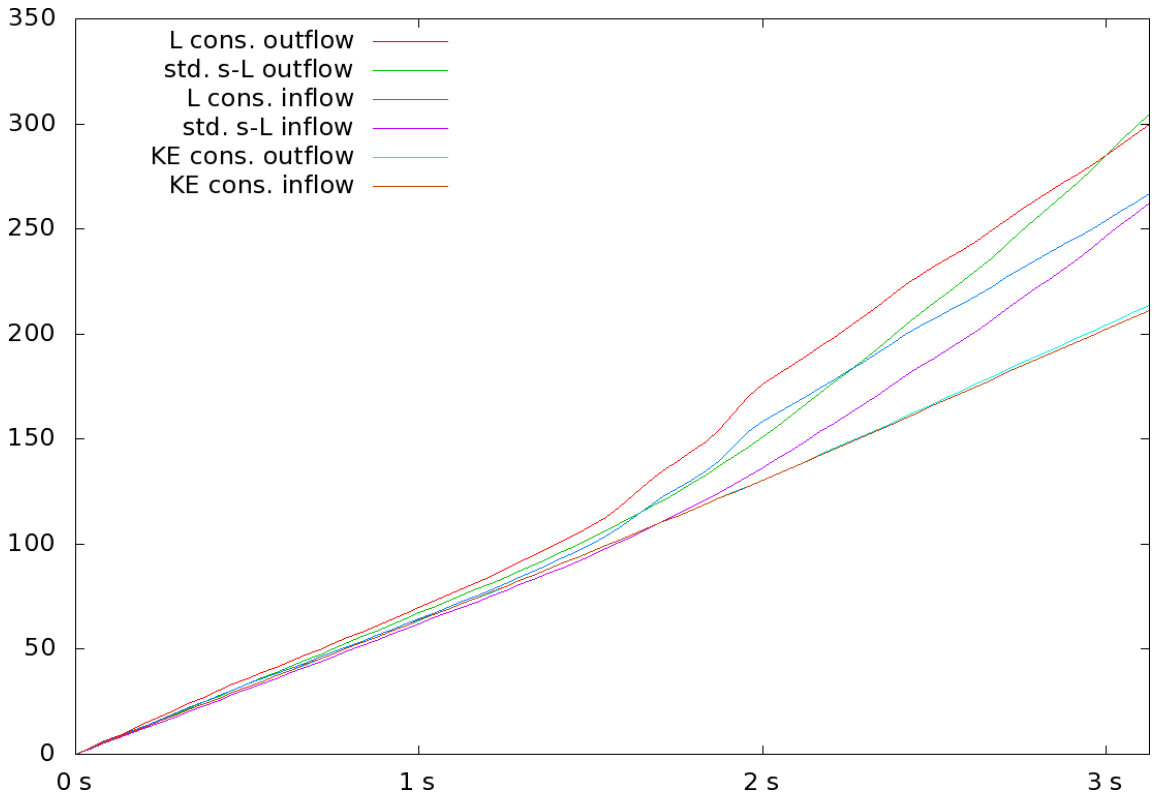


Figure 4.16: Total kinetic energy fluxing into the computational domain and total kinetic energy fluxing out of the computational domain, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme.

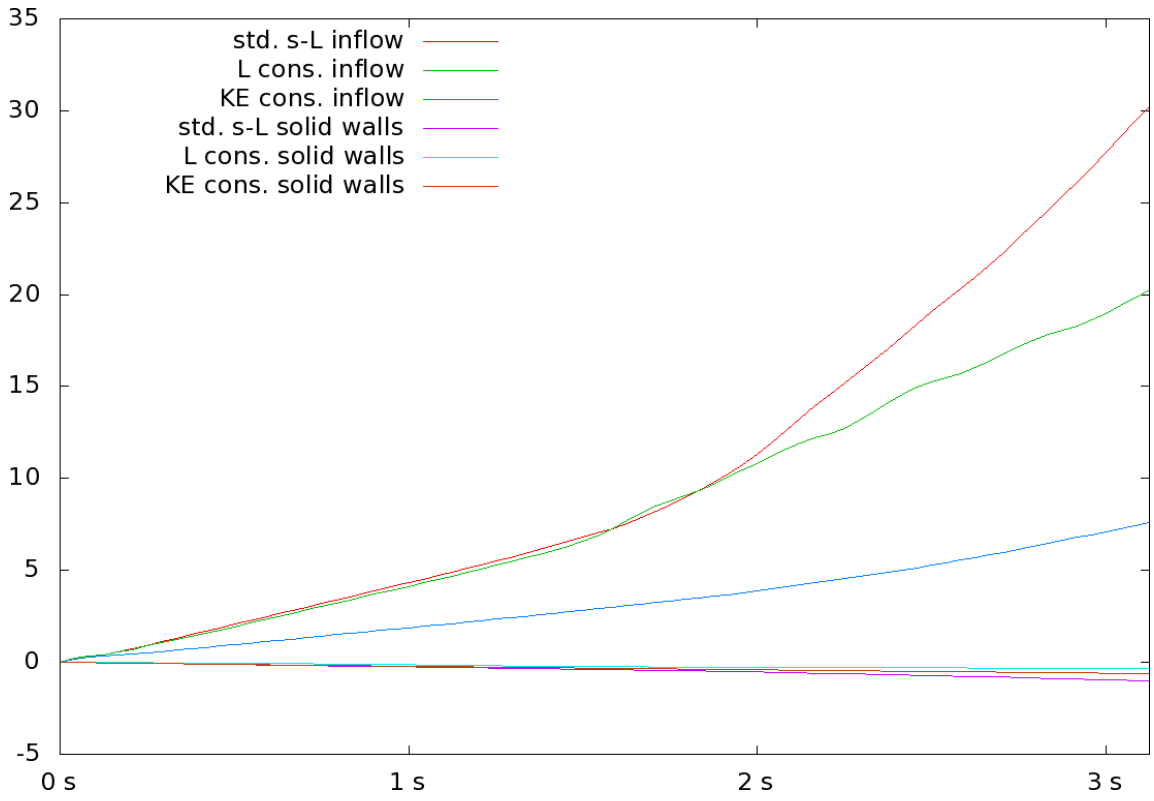


Figure 4.17: Energy flux into solid wall boundaries, and energy flux entering the computational domain from the inflow boundary condition, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme.

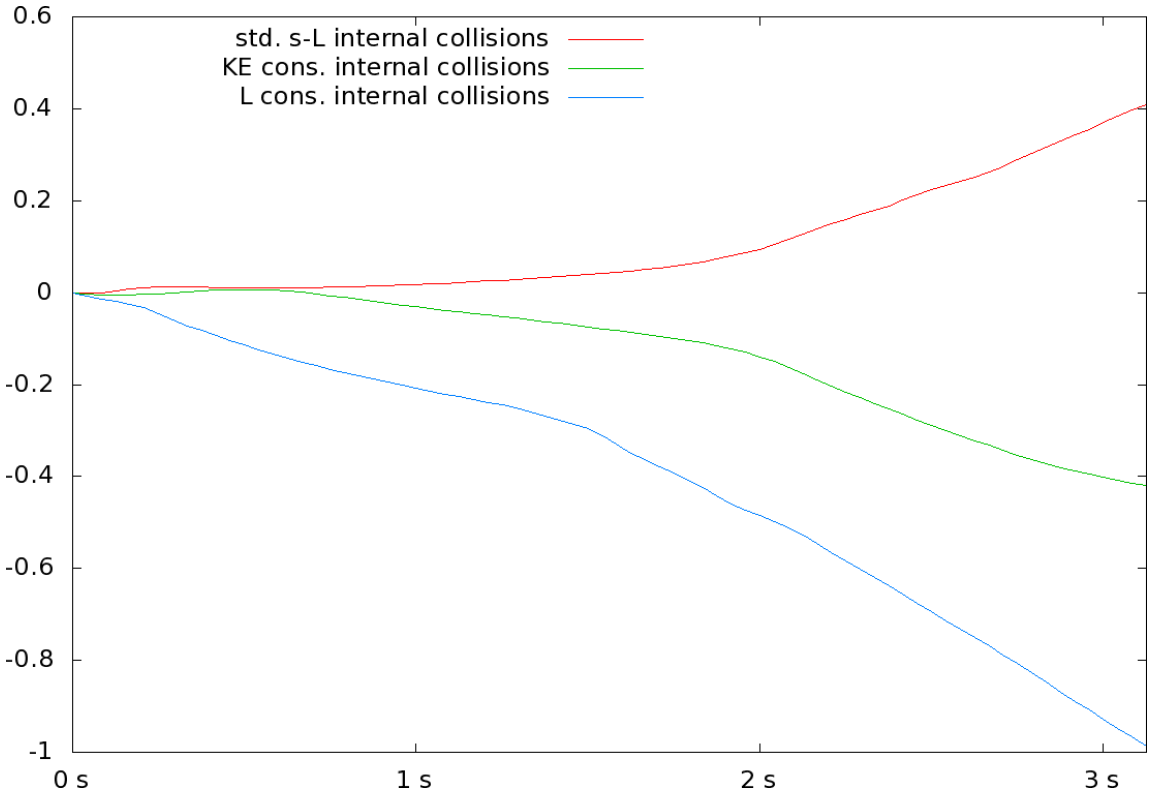


Figure 4.18: Change in kinetic energy due to the pressure projection step away from boundaries, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme. Note that in all three schemes the change in momentum due to the pressure projection step away from boundaries is zero.

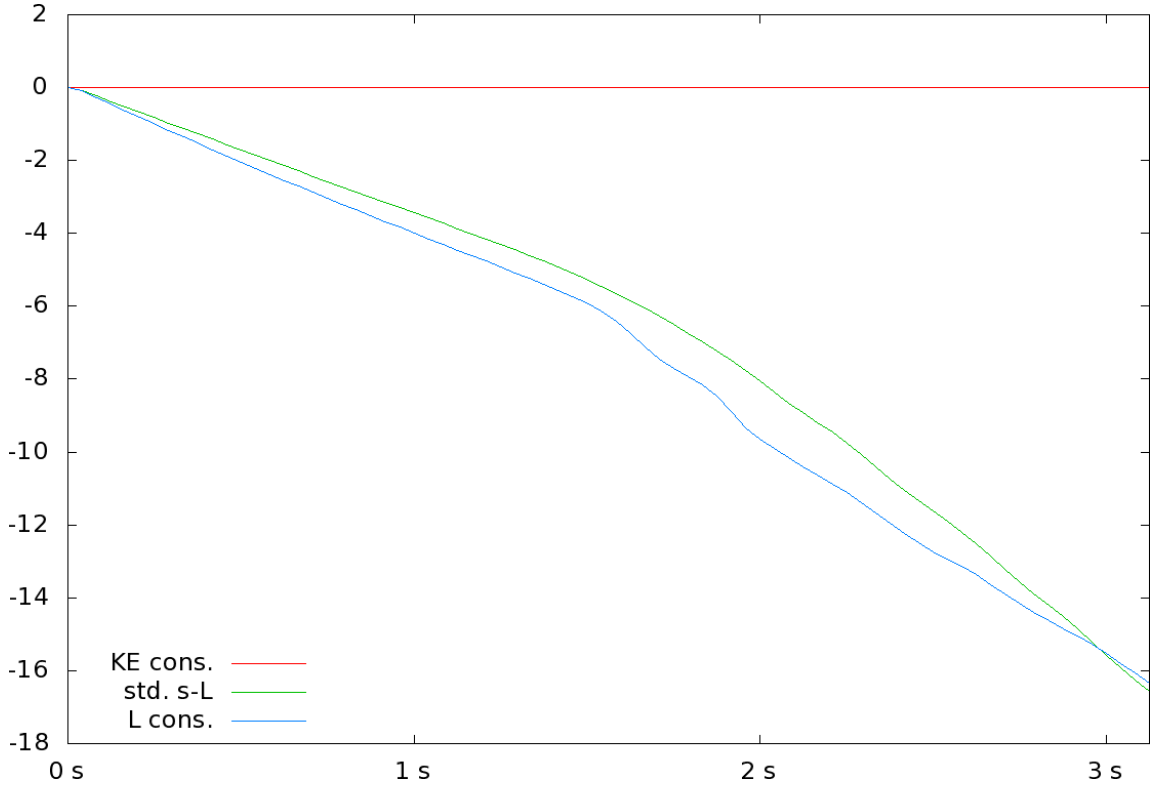


Figure 4.19: Sum total of kinetic energy in the domain, plus kinetic energy fluxed out of the domain (through outflow and solid wall boundaries), minus kinetic energy fluxed into the domain (through inflow), plus kinetic energy lost in the projection step away from boundaries, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme.

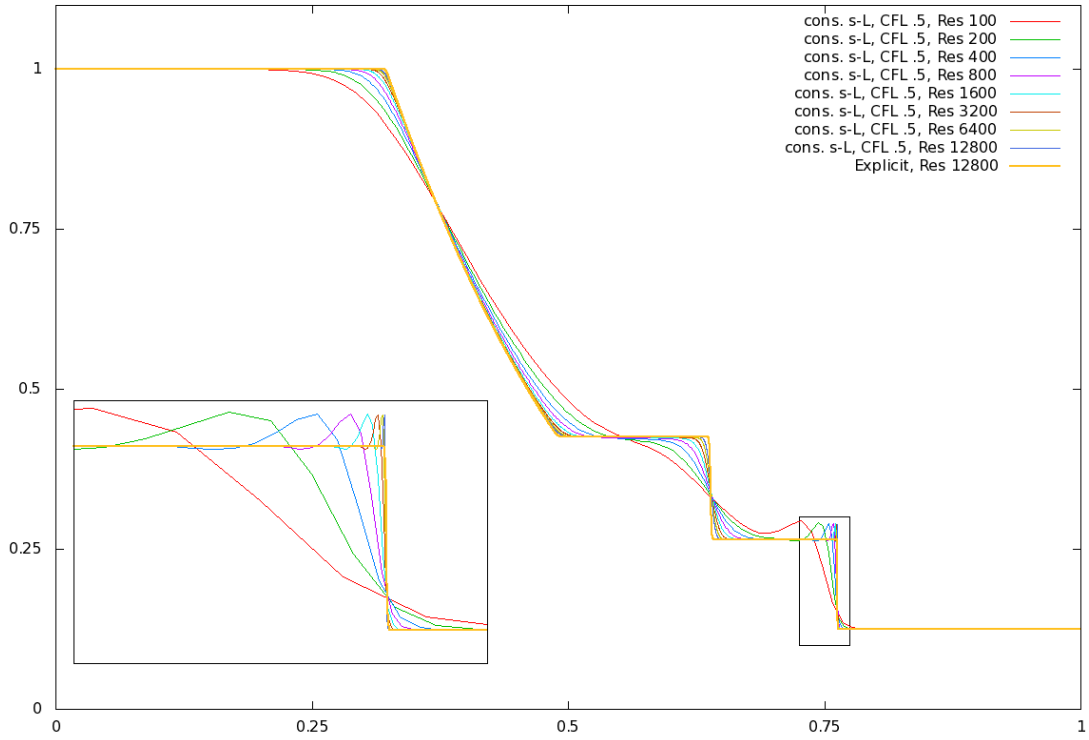


Figure 4.20: Density profile of a SOD shock tube at $t = .15s$, as generated by the scheme detailed in [50], using our new conservative semi-Lagrangian scheme and a CFL number of .5. We zoom in to the box $[.725, .775] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

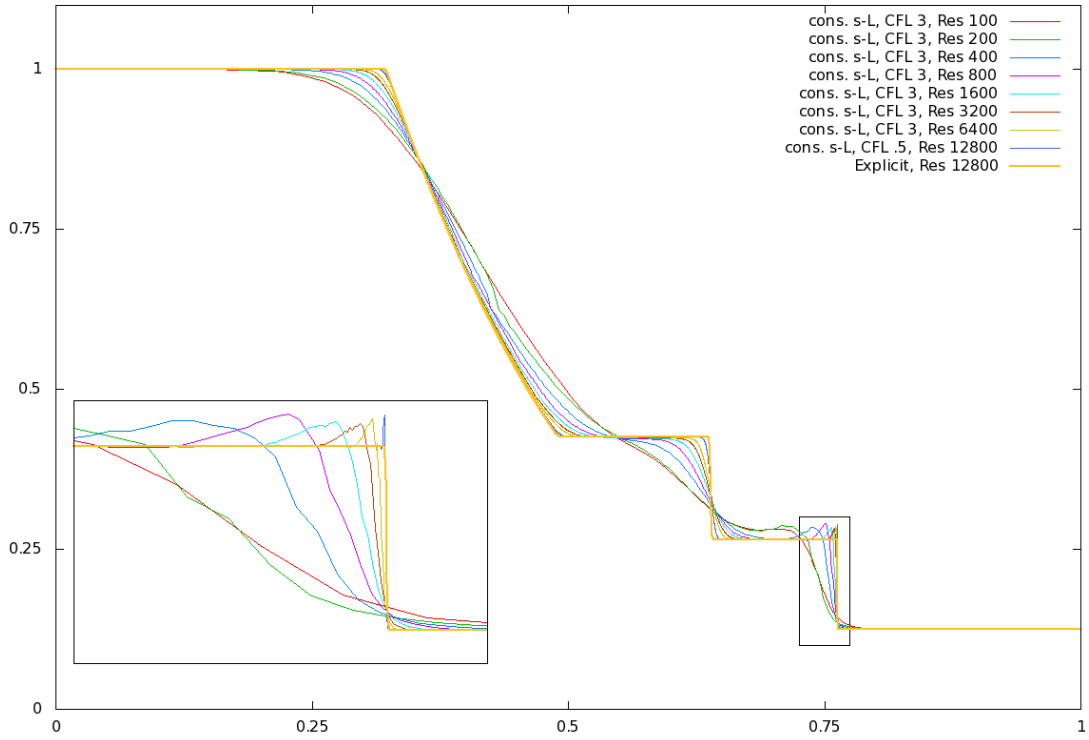


Figure 4.21: Density profile of a SOD shock tube at $t = .15s$, as generated by the scheme detailed in [50], using our new conservative semi-Lagrangian scheme and a CFL number of 3. We zoom in to the box $[.725, .775] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

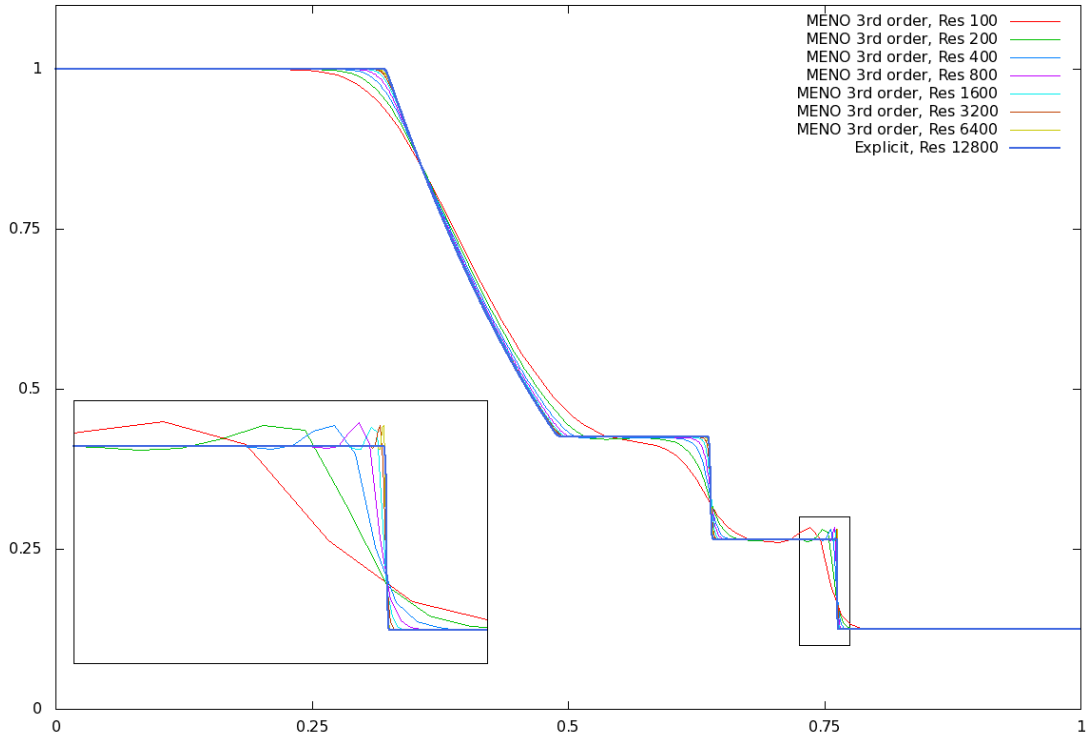


Figure 4.22: Density profile of a SOD shock tube at $t = .15s$, as generated by the scheme detailed in [50], using a third order MENO advection scheme and a CFL number of .5. We zoom in to the box $[.725, .775] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

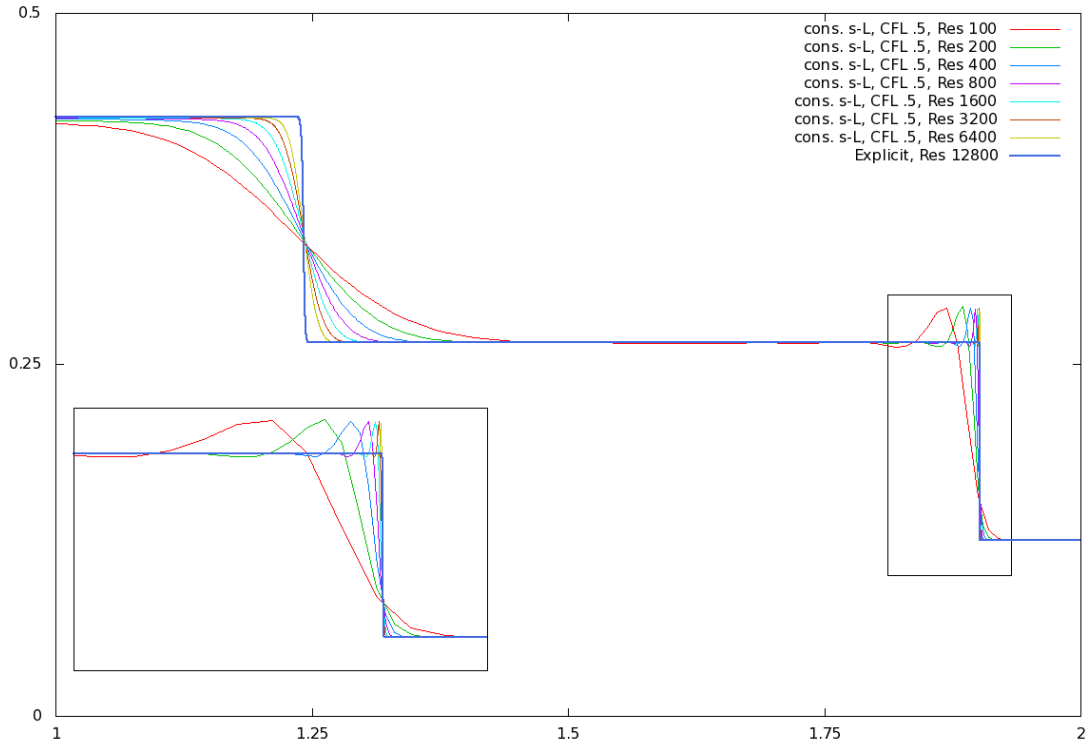


Figure 4.23: Density profile of a SOD shock tube at $t = .8s$, as generated by the scheme detailed in [50], using our new conservative semi-Lagrangian scheme and a CFL number of .5. In order to capture this later time, we extend the computational domain to $x \in (-1, 2)$ and show only $x \in (1, 2)$ to illustrate shock front convergence. We zoom in to the box $[1.812, 1.932] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

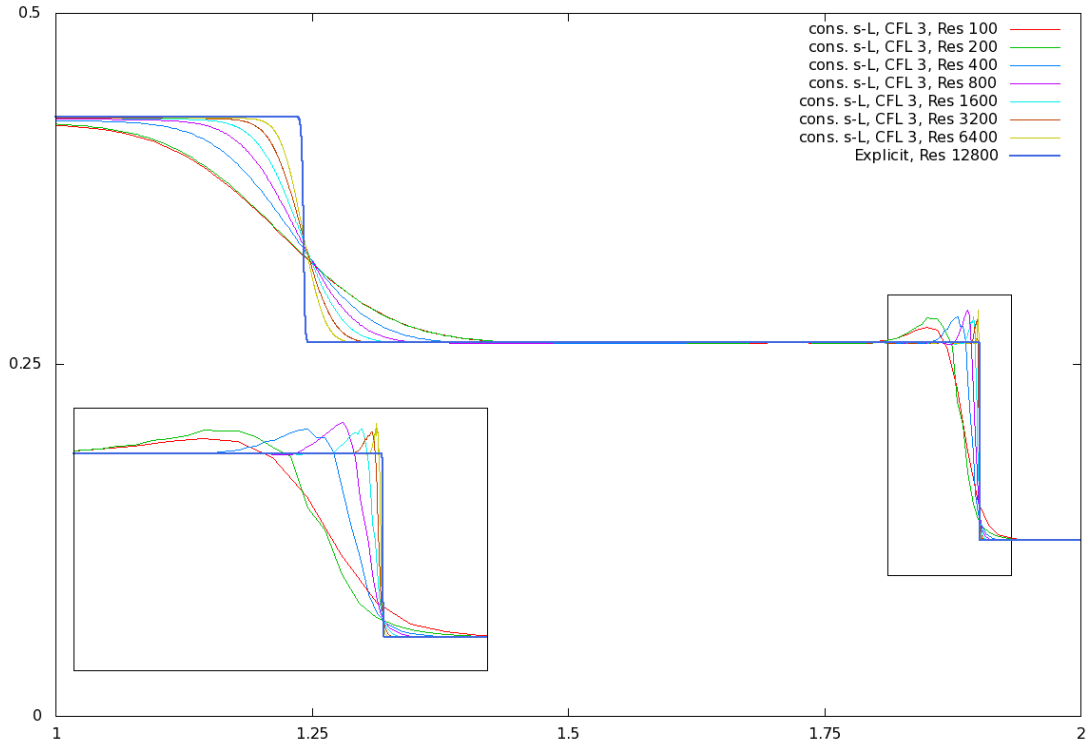


Figure 4.24: Density profile of a SOD shock tube at $t = .8s$, as generated by the scheme detailed in [50], using our new conservative semi-Lagrangian scheme and a CFL number of 3. In order to capture this later time, we extend the computational domain to $x \in (-1, 2)$ and show only $x \in (1, 2)$ to illustrate shock front convergence. We zoom in to the box $[1.812, .932] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

Chapter 5

Thin-shell Conservative FSI

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 [52], 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.

5.1 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 [10, 46] 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, 35, 20]. 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 [112, 84, 21], researchers have focused quite a bit of attention on so-called monolithic methods that employ higher degrees of implicit coupling [88, 31], 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 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 [23] and the immersed

boundary method (see [82] and the references therein, including [80, 81]). 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 [114]), simply by adding that mass to the fluid cells that contain the solid structure. Whereas ghost cell methods overcome stability restrictions for 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 [34, 35] 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 [91]. 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 [61, 56, 100]. Our semi-Lagrangian approach also uses upwind information to fill uncovered cells, but with the aide of [52] 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 [52]. 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 [93] (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 [9, 32] 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) [38] and arbitrary Lagrange-Eulerian (ALE) [37] 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 [101] 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 [52] is a very simple and straight-forward way to do this. We refer the interested reader to the following relevant VOF [36, 76, 2, 3, 64] and ALE papers [49, 70, 69, 74, 75, 7].

In order to capture the fluid-structure interactions we employ the flux-split compressible coupling methodology of [31], 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 [50] 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 [52] 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 [50], 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 [31], 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.

5.2 Conservative semi-Lagrangian advection

We begin with a short review of [52] to lay the groundwork for conservative semi-Lagrangian advection before proposing our novel extension to non-uniform grids in Section 5.2.1, and then hybridizing the resulting method with a traditional flux-based method in Section 5.2.3. A traditional semi-Lagrangian scheme approximates

$$\phi_t + \vec{u} \cdot \nabla \phi = 0, \tag{5.1}$$

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

$$\phi_j^{n+1} = \phi\left(x_j, t^{n+1}\right) = \phi\left(x_j^-, t^n\right) = \sum_i w_{ij} \phi\left(x_i, t^n\right) \tag{5.2}$$

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

$$\hat{\phi}_t + \nabla \cdot (\hat{\phi} \vec{u}) = 0, \tag{5.3}$$

which can be derived by multiplying Equation (5.1) 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.

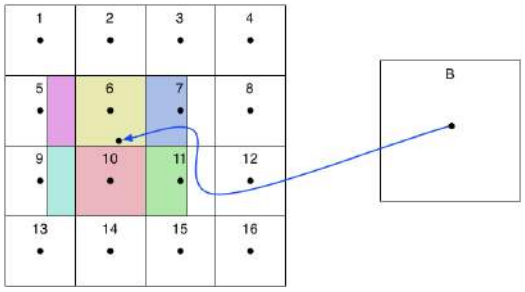
In order to strictly enforce conservation while remaining consistent with Equation (5.3), [52] computes modified weights $\hat{w}_{ij}$ such that $\sum_j \hat{w}_{ij} = 1$ for all cells

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, $\hat{w}_{ij} = w_{ij} + (1 - \sigma_i) f_{ij}$.

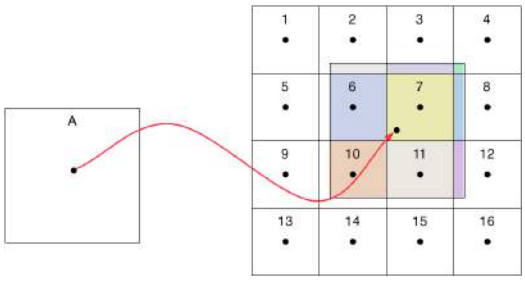
To summarize, they compute modified weights

$$\hat{w}_{ij} = \begin{cases} w_{ij}/\sigma_i & \sigma_i > 1 \\ w_{ij} + (1 - \sigma_i) f_{ij} & \sigma_i \leq 1 \end{cases} \tag{5.4}$$

and update $\hat{\phi}_j^{n+1} = \sum_i \hat{w}_{ij} \hat{\phi}_i^n$, to solve Equation (5.3). By moving data from time t^n to time t^{n+1} only along characteristic lines, they have consistency with Equation (5.3), and by enforcing that $\sum_j \hat{w}_{ij} = 1$ they attain numerical conservation. A key point is that Equation (5.3) 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 [51], which explores the conservative semi-Lagrangian approach in some detail, showing the benefits of using the conservative form over Equation (5.1) 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 5.1: 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.

5.2.1 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 5.1(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 [8] 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 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 5.1(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 (5.3) on an arbitrary grid by computing backward-advected weights $w_{ij} = \|\Omega_i \cap \Omega_j^-\|$ and forward-advected weights $f_{ij} = \|\Omega_i^+ \cap \Omega_j\|/\|\Omega_i^+\|$

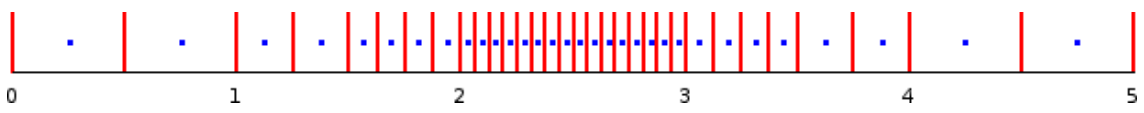


Figure 5.2: A non-uniform one-dimensional spatial grid, with flux boundaries shown as red vertical lines and cell-centered degrees of freedom as blue points. In the depicted grid, the smallest cell is of size $\Delta x_f = .0625$, while the largest cell is of size $\Delta x_c = .5$, $8\times$ larger 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.

as necessary, and then compute modified weights

$$\hat{w}_{ij} = \begin{cases} (\|\Omega_i\|/\sigma_i) w_{ij} & \sigma_i > \|\Omega_i\| \\ w_{ij} + (\|\Omega_i\| - \sigma_i)\, f_{ij} & \sigma_i \leq \|\Omega_i\|. \end{cases} \tag{5.5}$$

The final update can then be written as

$$\hat{\phi}_j^{n+1} = \sum_i \frac{\hat{\phi}_i^n \hat{w}_{ij}}{\|\Omega_j\|}. \tag{5.6}$$

5.2.2 Examples

Equation (5.3) 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 5.3(a) and 5.3(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 5.3(c) and 5.3(d) a non-uniform grid (shown in Figure 5.2) is utilized, and in Figure 5.3(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.

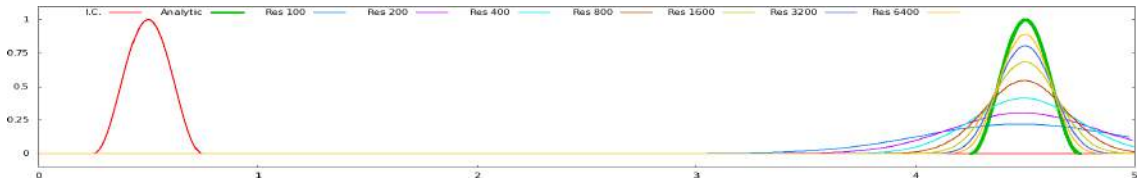
We also consider a square wave being advected through a divergent velocity field with $u(x) = \sin\left(\frac{\pi}{5}x\right)$. In Figures 5.4(a) and 5.4(b) the results at time $t = 5s$ are compared between the volumetric conservative semi-Lagrangian advection and first order accurate ENO-LLF schemes, on a uniform grid. In Figures 5.3(c) and 5.3(d) a non-uniform grid (shown in Figure 5.2) is utilized, and in Figure 5.3(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.

5.2.3 Hybrid flux / conservative semi-Lagrangian coupling

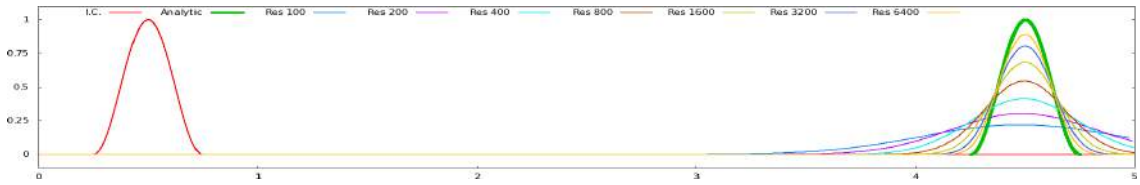
The conservative semi-Lagrangian advection scheme is unconditionally stable, which makes it ideal for regions of the flow field where small or irregular grid cells are necessary to resolve small-scale geometric detail. However, the method is limited to first order accuracy, and so is less desirable as 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.

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.

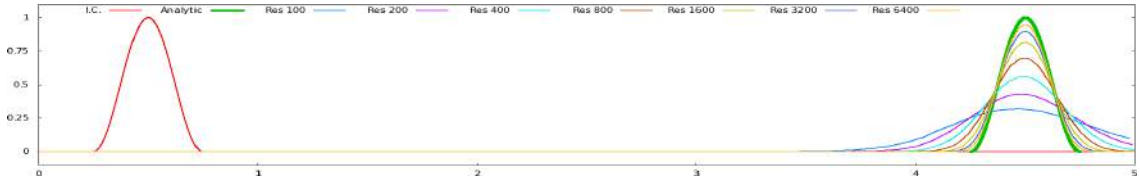
In the semi-Lagrangian region, the boundary flux must be accounted for in such a



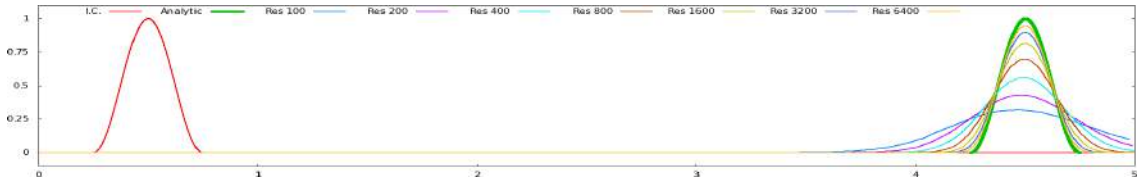
(a) First order accurate ENO-LLF on a uniform grid, with a CFL number $\alpha = .5$.



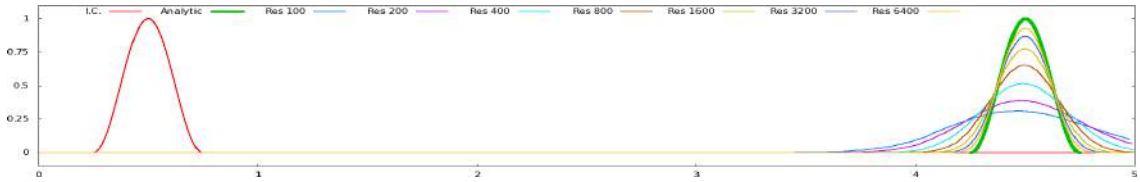
(b) Conservative semi-Lagrangian scheme on a uniform grid, with a CFL number $\alpha = .5$.



(c) First order accurate ENO-LLF on the non-uniform grid shown in Figure 5.2, with a CFL number $\alpha = .5$.

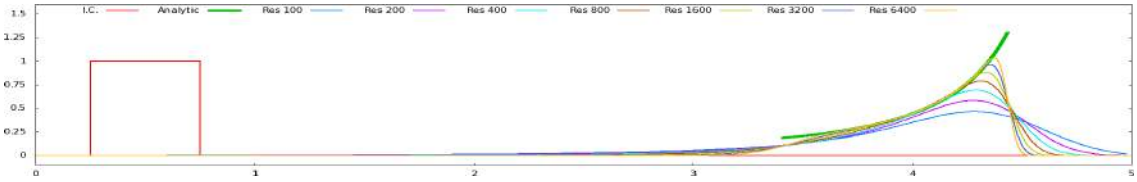


(d) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 5.2, with a CFL number $\alpha = .5$.

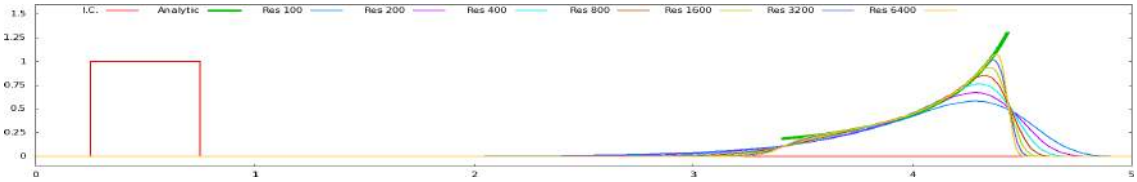


(e) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 5.2, with a CFL number $\alpha = .5$ taken with respect only to the coarse grid cells (and so the *effective* CFL number is 4).

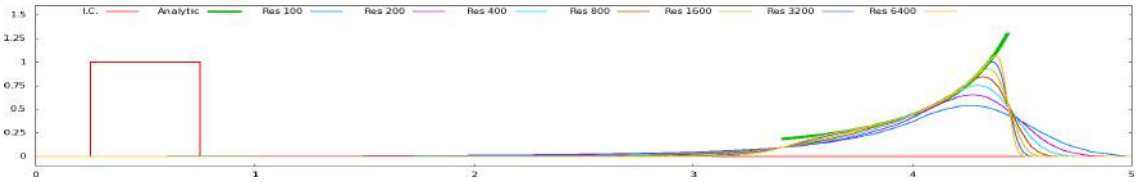
Figure 5.3: A sinusoidal wave is advected through a constant velocity field, $u = 1$, using a variety of spatial discretizations and grids, with TVD-RK3 time integration. Results are shown at $t = 4s$.



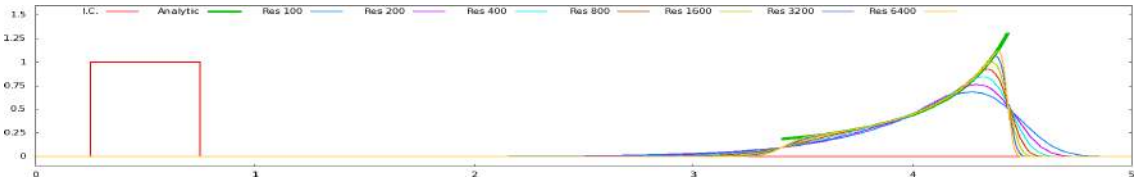
(a) First order accurate ENO-LLF on a uniform grid, with a CFL number $\alpha = .5$.



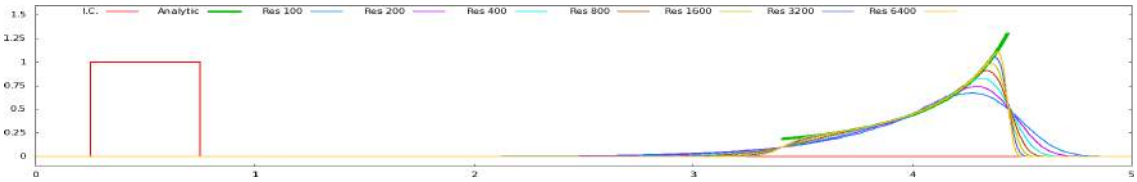
(b) Conservative semi-Lagrangian scheme on a uniform grid, with a CFL number $\alpha = .5$.



(c) First order accurate ENO-LLF on the non-uniform grid shown in Figure 5.2, with a CFL number $\alpha = .5$.



(d) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 5.2, with a CFL number $\alpha = .5$.



(e) Conservative semi-Lagrangian scheme on the non-uniform grid shown in Figure 5.2, with a CFL number $\alpha = .5$ taken with respect only to the coarse grid cells (and so the *effective* CFL number is 4).

Figure 5.4: 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 $t = 5s$.

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 (5.5) and (5.6), 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 5.1(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 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

$$\tilde{w}_{ij} = \begin{cases} (K_i/\hat{\sigma}_i)\hat{\phi}_i^n w_{ij} & |\hat{\sigma}_i| > |K_i| \\ \hat{\phi}_i^n w_{ij} + (K_i - \hat{\sigma}_i)\,\tilde{f}_{ij} & |\hat{\sigma}_i| \leq |K_i|. \end{cases} \tag{5.7}$$

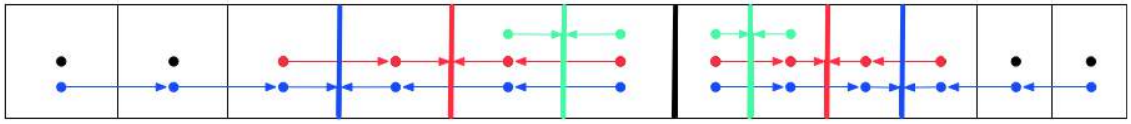


Figure 5.5: In the hybrid advection scheme, near areas where the cell size changes, we drop the stencil width (and therefore the order of accuracy) of the flux scheme to avoid crossing the refinement interface. In the one-dimensional example illustrated here, the blue flux faces are solved using a third 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.

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

$$\hat{\phi}_j^{n+1} = \|\Omega_j\|^{-1} \left[\sum_i \tilde{w}_{ij} + \sum_i \tilde{w}_{\mathcal{F}_i j} \right]. \tag{5.8}$$

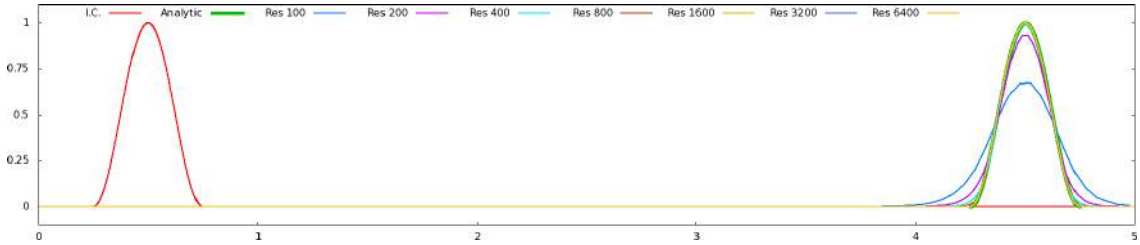
As both the flux scheme and the conservative semi-Lagrangian scheme are conservative, the resulting hybrid scheme is also fully conservative.

5.2.4 Examples

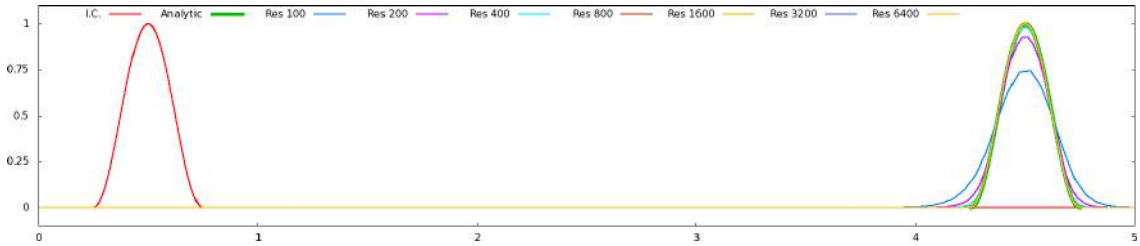
We noticed that, in the non-linear examples considered later in this paper, numerical artifacts manifest when the order of accuracy of the flow solver drops dramatically between neighboring cell faces – these include artificial reflected waves, kinks and overshoots. With this in mind, although the hybrid 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 [62, 40]. 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 5.5. Unless otherwise stated we use ENO-LLF.

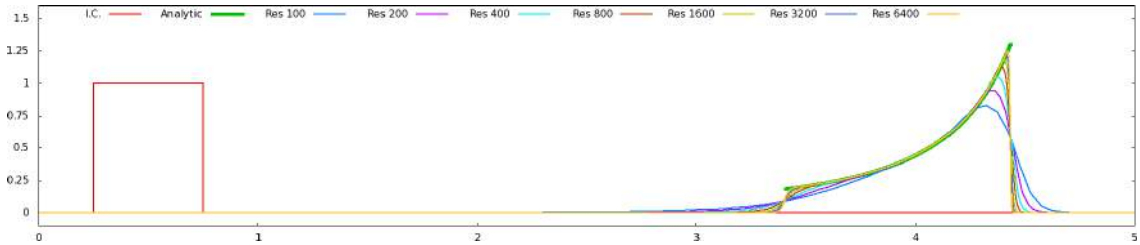
We return to the examples discussed in Section 5.2.2 on the non-uniform grid given in Figure 5.2, and utilize the proposed hybrid scheme in order to employ a third order accurate ENO-LLF scheme away from the refinement interfaces located at $x = \{1, 1.5, 2, 3, 3.5, 4\}$. Since these problems are linear 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 5.6(b) and 5.6(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.



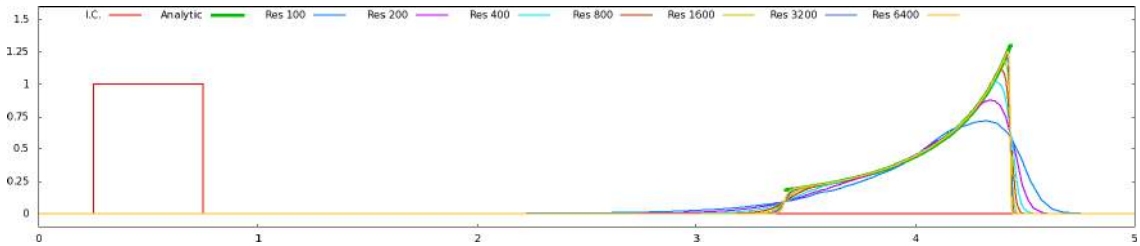
(a) Baseline sinusoidal wave advected through a constant velocity field computed using third order accurate ENO-LLF on a uniform grid, with a CFL number $\alpha = .5$.



(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 5.6: Conservative advection is solved on the non-uniform grid shown in Figure 5.2 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.

5.3 Semi-implicit compressible flow formulation

As demonstrated in [52], a conservative semi-Lagrangian advection scheme can be combined with the implicit pressure solve of [50] 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 [50] and use our proposed modifications both for non-uniform grid refinement and hybrid advection.

Consider the Euler equations, given by

$$\begin{pmatrix} \rho \\ \rho\vec{u} \\ E \end{pmatrix}_t + \begin{pmatrix} \nabla \cdot \rho\vec{u} \\ \nabla \cdot (\rho\vec{u} \otimes \vec{u}) \\ \nabla \cdot E\vec{u} \end{pmatrix} + \begin{pmatrix} 0 \\ \nabla p \\ \nabla \cdot p\vec{u} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{5.9}$$

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

$$\vec{u}^{n+1} = \vec{u}^\star - \Delta t \frac{\nabla p^{n+1}}{\rho^{n+1}}, \tag{5.10}$$

and its divergence is taken to obtain

$$\nabla \cdot \vec{u}^{n+1} = \nabla \cdot \vec{u}^\star - \Delta t \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right). \tag{5.11}$$

The pressure evolution equation (see [24]), which is given by

$$p_t + \vec{u} \cdot \nabla p = -\rho c^2 \nabla \cdot \vec{u}, \tag{5.12}$$

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

$$p^{n+1} = p^a - \Delta t \rho c^2 \nabla \cdot \vec{u}^{n+1}. \tag{5.13}$$

Substituting this in to Equation (5.11) and rearranging gives

$$p^{n+1} - \Delta t^2 \rho^n (c^2)^n \nabla \cdot \left(\frac{\nabla p^{n+1}}{\rho^{n+1}} \right) = p^a - \Delta t \rho^n (c^2)^n \nabla \cdot \vec{u}^\star, \tag{5.14}$$

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

$$\left[\mathbf{P}^{-1} + G^T (\hat{\rho}^{n+1})^{-1} G \right] \hat{p}^{n+1} = \mathbf{P}^{-1} \hat{p}^a + G^T \vec{\hat{u}}^\star, \tag{5.15}$$

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

$$\hat{p}_{i+1/2} = \frac{\rho_i \hat{p}_{i+1} + \rho_{i+1} \hat{p}_i}{\rho_i + \rho_{i+1}} \tag{5.16}$$

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

$$\hat{u}^{n+1}_{i+1/2} = \hat{u}^\star_{i+1/2} - \hat{\rho}^{-1}_{i+1/2} G_{i+1/2} \hat{p}^{n+1}, \tag{5.17}$$

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

$$(\rho \vec{u})^{n+1}_i = (\rho \vec{u})^\star_i - \frac{\hat{p}^{n+1}_{i+1/2} - \hat{p}^{n+1}_{i-1/2}}{\Delta x}, \qquad E^{n+1}_i = E^\star_i - \frac{\hat{p}^{n+1}_{i+1/2} \hat{u}^{n+1}_{i+1/2} - \hat{p}^{n+1}_{i-1/2} \hat{u}^{n+1}_{i-1/2}}{\Delta x}. \tag{5.18}$$

5.3.1 Computing the advected pressure

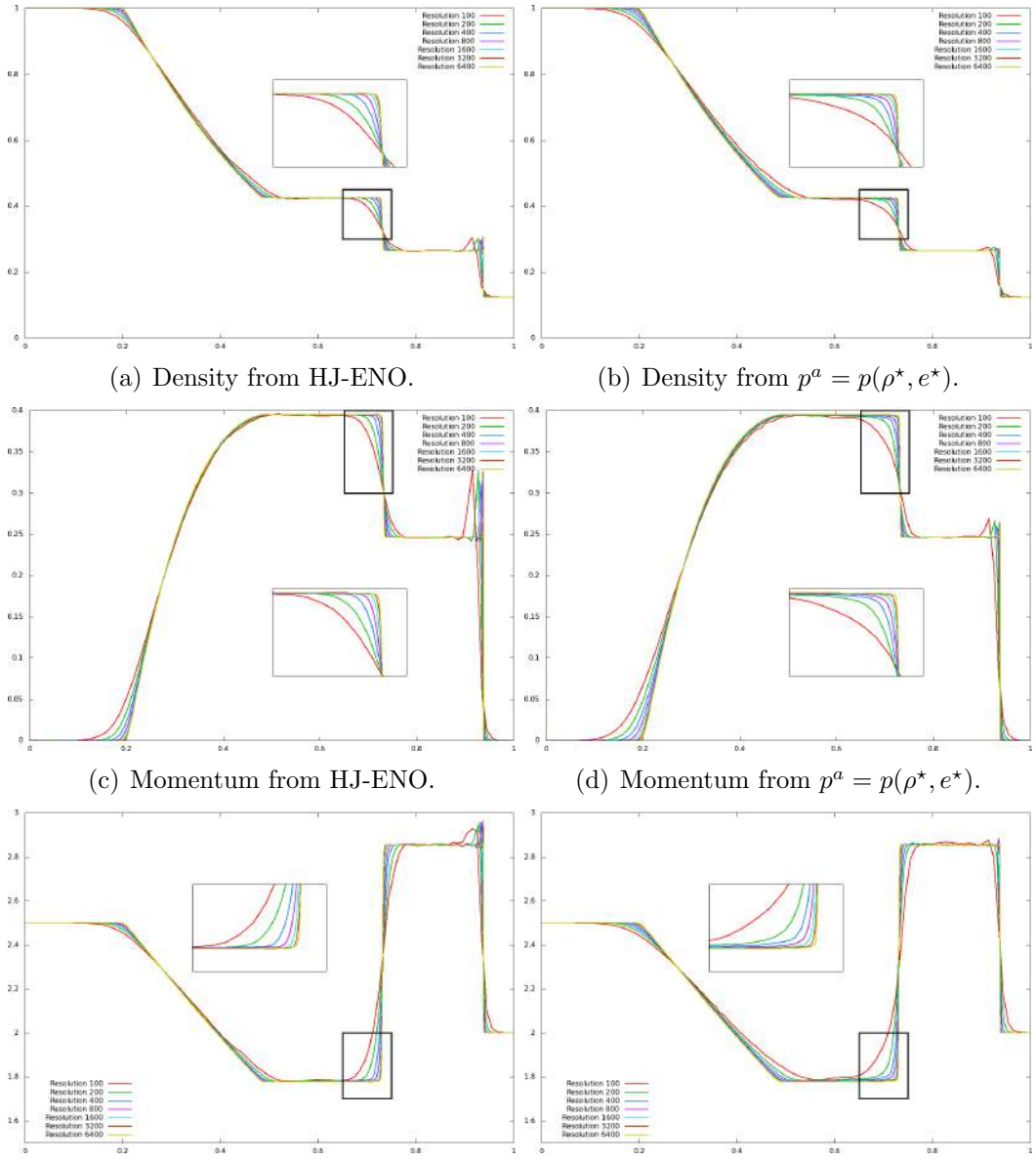
In previous work, $p^a = p^n - \Delta t \vec{u} \cdot \nabla p$ was computed using Hamilton-Jacobi ENO [50]. When doing so they noticed Gibbs phenomena near the shock front (which can be seen in the left column of Figure 5.7), and in order to mitigate these oscillations a MAC grid-based ENO (or MENO) 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 5.7. 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.

5.3.2 Modified ENO-GLF scheme

In the higher spatial dimension examples considered later on in Section 5.5.3, 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 5.8, 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



(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 5.7: 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 $t = .25s$.

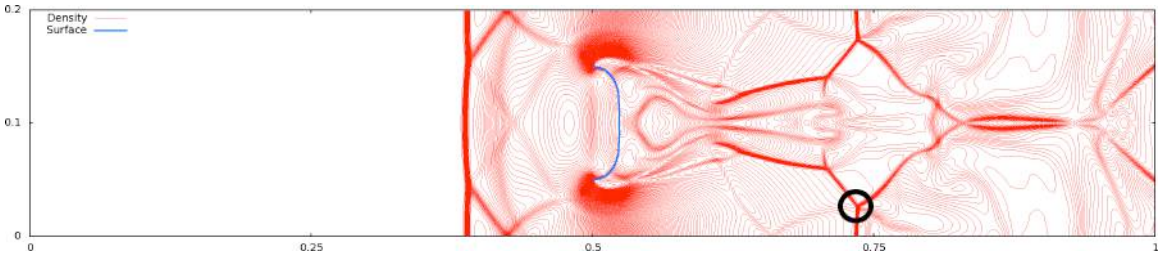


Figure 5.8: In this two-dimensional example (explored in further detail in Section 5.5.3), artificial cavitation occurs in the circled region, behind a Mach stem, when the standard Lax-Friedrich's third order accurate variant of ENO is used to compute $\vec{U}^\star$. Shown are isocontours for density for the example described in Section 5.5.3 at time $t = .2s$.

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

$$D_i^1\mathcal{F} = \mathcal{F}(\hat{\phi}_i) \qquad D_i^1\hat{\phi} = \hat{\phi}_i$$
$$D_{i+1/2}^2\mathcal{F} = \frac{D_{i+1}^1\mathcal{F} - D_i^1\mathcal{F}}{2\Delta x} \qquad D_{i+1/2}^2\hat{\phi} = \frac{D_{i+1}^1\hat{\phi} - D_i^1\hat{\phi}}{2\Delta x}$$
$$D_i^3\mathcal{F} = \frac{D_{i+1/2}^2\mathcal{F} - D_{i-1/2}^2\mathcal{F}}{3\Delta x}$$

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

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

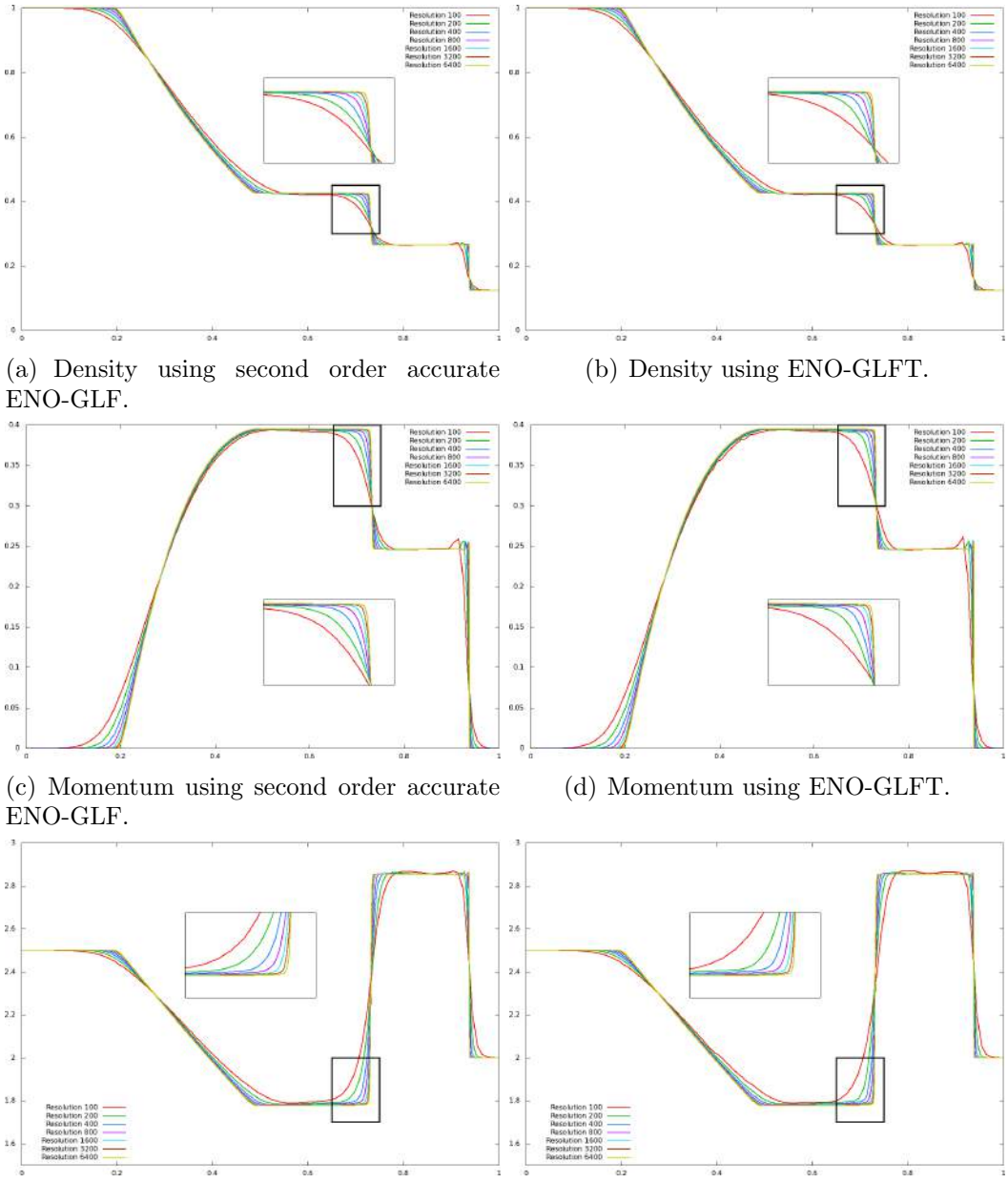
We designate this *truncated* diffusion version of ENO-GLF as ENO-GLFT. We compare ENO-GLFT with the standard second order accurate variant of ENO-GLF in Figure 5.9 to show the impact of factoring in the third order accurate terms for $\mathcal{F}$ but not the third order accurate diffusion term. The 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 5.7. 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 5.5.3. 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.

5.3.3 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 $-\hat{G}^T$



(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 5.9: 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 5.3.2 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 5.7, 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 $t = .25s$.

by multiplying the regular divergence operator through by the volume of the cell (see for example [65, 67]). 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

$$\vec{\hat{u}}^{n+1} = \vec{\hat{u}}^{\star} - \beta^{-1}\hat{G}\hat{p}^{n+1}. \tag{5.19}$$

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

$$-\hat{G}^T\vec{\hat{u}}^{n+1} = -\hat{G}^T\vec{\hat{u}}^{\star} + \hat{G}^T\beta^{-1}\hat{G}\hat{p}^{n+1}. \tag{5.20}$$

And following the derivation of Equation (5.15) then leads to

$$\left[\hat{\mathbf{P}}^{-1} + \hat{G}^T\beta^{-1}\hat{G}\right]\hat{p}^{n+1} = \hat{\mathbf{P}}^{-1}\hat{p}^{a} + \hat{G}^T\vec{\hat{u}}^{\star} \tag{5.21}$$

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

$$(\rho\vec{u})_i^{n+1} = (\rho\vec{u})_i^{\star} - \frac{\hat{p}_{i+1/2}^{n+1} - \hat{p}_{i-1/2}^{n+1}}{\Delta} \qquad \text{and} \qquad E_i^{n+1} = E_i^{\star} - \frac{\hat{p}_{i+1/2}^{n+1}\hat{u}_{i+1/2}^{n+1} - \hat{p}_{i-1/2}^{n+1}\hat{u}_{i-1/2}^{n+1}}{\Delta} \tag{5.22}$$

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 (5.16), and the

face velocity $\hat{u}^{n+1}$ is recovered from Equation (5.19) as

$$\hat{u}^{n+1}_{i+1/2} = \hat{u}^{\star}_{i+1/2} - \beta^{-1}_{i+1/2}\hat{G}_{i+1/2}\hat{p}^{n+1}, \tag{5.23}$$

similar to Equation (5.17) 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 $\hat{u}^{n+1}_{i+1/2}$ 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 (5.19) using $\beta_{i+1/2}$ and $\hat{u}^{\star}_{i+1/2}$ from above. At a constrained Neumann face we can compute

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

where d is the distance from the cell center i to the cell face $i + 1/2$ (see for example [67]). 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 $\hat{p}^{n+1}_{i+1/2} = \hat{p}_i$.

5.3.4 Sod's shock example

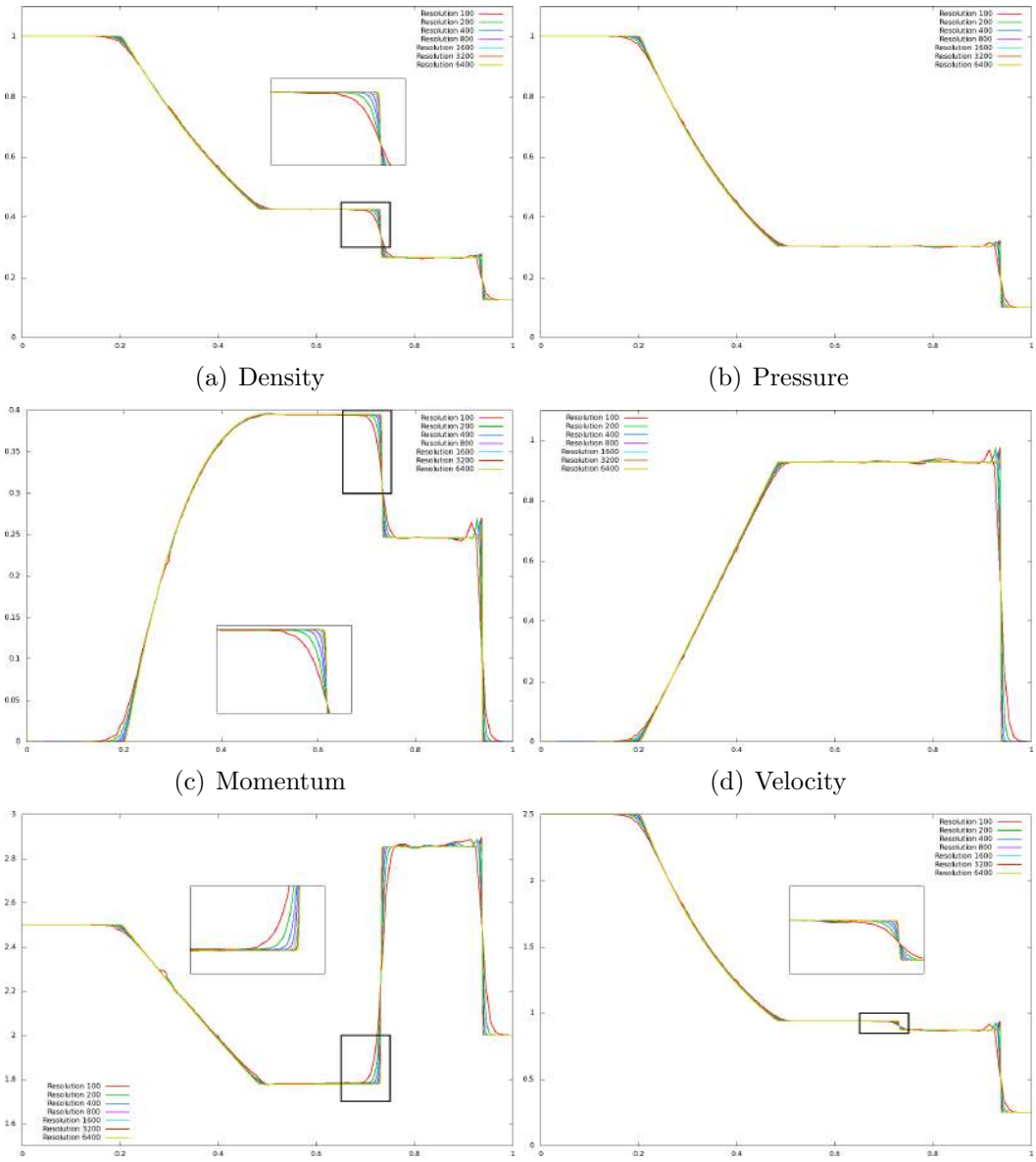
We consider Sod's shock with initial conditions specified over a computational domain of $[0, 1]$ as

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

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 5.9. Here we consider Sod's shock under non-uniform grid refinement using the grid pattern shown in Figure 5.2, 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 5.5—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 5.10. 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 5.11 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 5.11(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 [25] may alleviate these peaks, but doing so would sacrifice conservation. Similar but less pronounced issues occur in Figure 5.10(e) when the scheme used to capture the numerical shock changes.

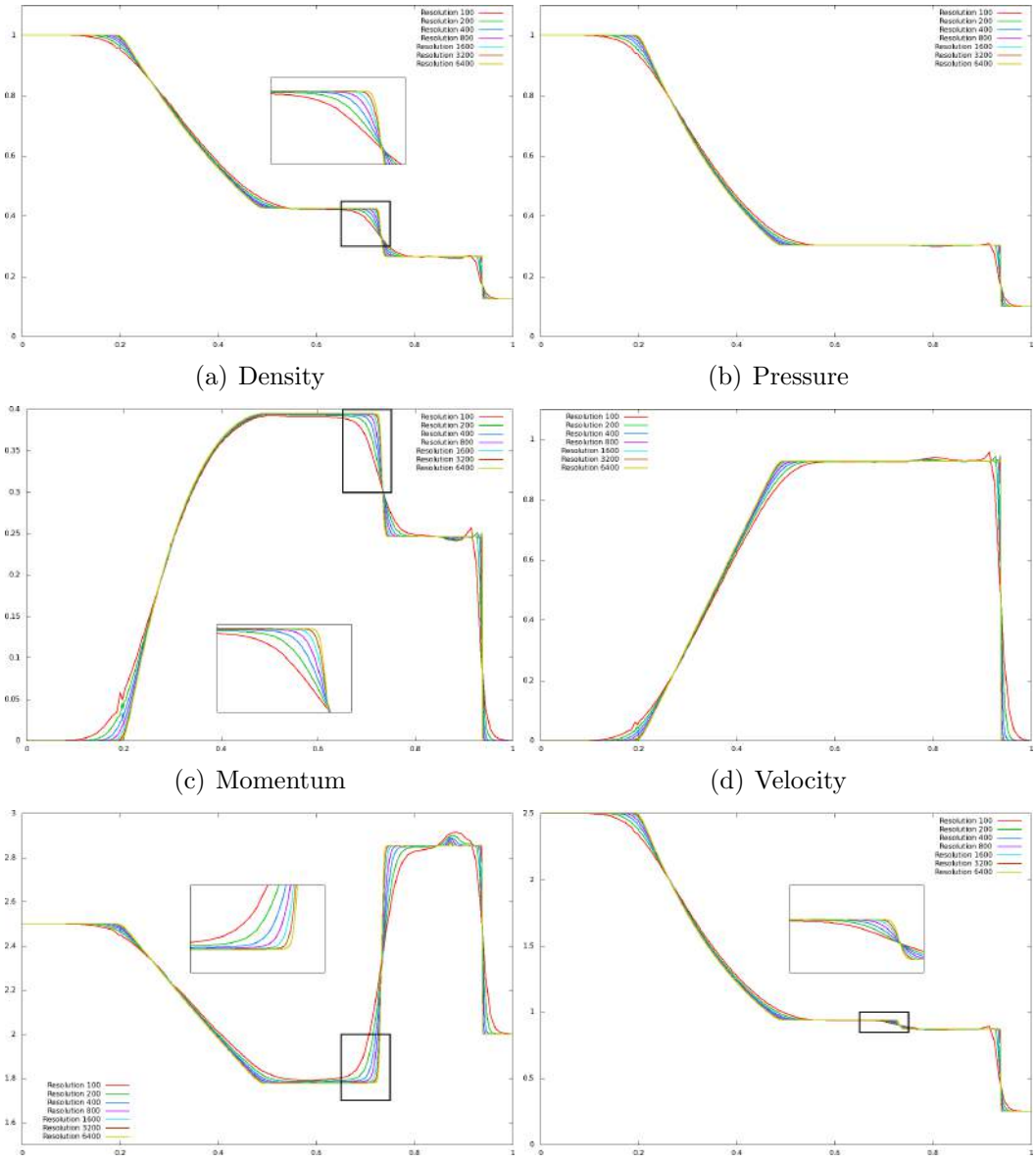


(a) Density (b) Pressure

(c) Momentum (d) Velocity

(e) Internal Energy (f) Total Energy

Figure 5.10: 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 5.5).



(a) Density (b) Pressure

(c) Momentum (d) Velocity

(e) Internal Energy (f) Total Energy

Figure 5.11: 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.

5.4 Fluid-structure interactions

We modify the fluid-structure solver of [31] 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.

5.4.1 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 5.4.2 and 5.4.3). 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 5.12. 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 5.12. 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 [91] where cut cell generation techniques are discussed.

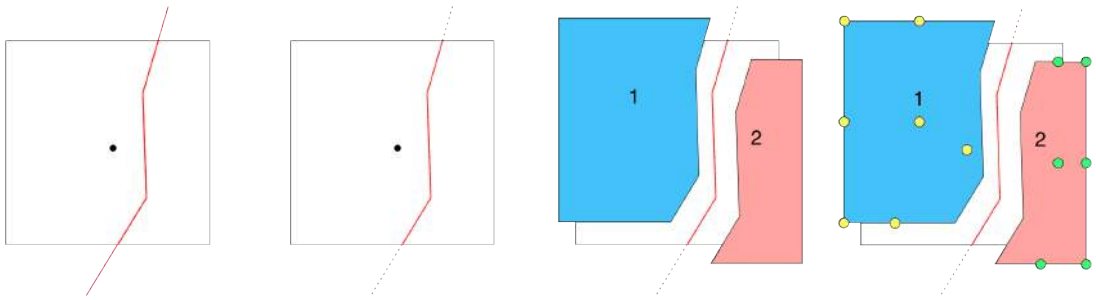


Figure 5.12: When a grid cell is cut by a structure interface (shown as a red segmented curve), we first clip the segments of the curve against cell boundaries as shown in the second figure. Then the clipped interface is stitched together with contiguous components of the cell volume boundary, yielding 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).

5.4.2 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 5.12 contains its own cell center, and so is trivially associated with it. The right-hand polygonal region shown in Figure 5.12 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 5.12 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 5.4.4. 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

$$\rho^{\text{new}} = \frac{M + \bar{M}_{\text{cut}}}{V + \bar{V}_{\text{cut}}} \tag{5.24}$$

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 (5.24) 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.

5.4.3 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 [9, 32] as opposed to the static solid positions used above in Section 5.4.2. 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 [9] and the water and cloth simulations in [32]. 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.

5.4.4 Advection

As discussed in Section 5.4.2 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 5.1). 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 5.4.3. 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 5.1. Then when calculating overlap, again as in Figure 5.1, 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 5.4.2. 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.

5.4.5 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.

5.4.6 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

$$M_S V_S^{n+1} = M_S V_S^\star + \Delta t D(X_S^n) V_S^{n+1}. \tag{5.25}$$

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 [88].

5.4.7 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 5.4.8. 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 5.4.8) 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 5.4.4 in the Runge-Kutta style time integration of Section 5.4.5. We also compute the intermediate momentum for the solid $M_S V_S^\star$. Finally we once again solve the coupled system (of Section 5.4.8) to find the final time t^{n+1} momentum of the solid, as well as the final time t^{n+1} fluid state.

5.4.8 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 [89].

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

$$\mathbf{J}V_S^{n+1} - \mathbf{W}\hat{u}^{n+1} = 0. \tag{5.26}$$

This constraint is enforced using a set of impulses λ that are exchanged between the fluid and the structure. Equation (5.25) for the structure becomes

$$M_S V_S^{n+1} = M_S V_S^{\star} + \Delta t D V_S^{n+1} - \mathbf{J}^T\lambda. \tag{5.27}$$

and the corresponding equations for the fluid become

$$\vec{\hat{u}}^{n+1} = \vec{\hat{u}}^{\star} - \beta^{-1}\hat{G}\hat{p} + \beta^{-1}\mathbf{W}^T\lambda, \tag{5.28}$$

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

$$\begin{pmatrix} \hat{\mathbf{P}}^{-1} + \hat{G}^T\beta^{-1}\hat{G} & -\hat{G}^T\beta^{-1}\mathbf{W}^T & 0 \\ -\mathbf{W}\beta^{-1}\hat{G} & \mathbf{W}\beta^{-1}\mathbf{W}^T & -\mathbf{J} \\ 0 & -\mathbf{J}^T & -M_S + \Delta t D \end{pmatrix} \begin{pmatrix} \hat{p}^{n+1} \\ \lambda \\ V_S^{n+1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{P}}^{-1}\hat{p}^a + \hat{G}^T\vec{\hat{u}}^{\star} \\ -\mathbf{W}\vec{\hat{u}}^{\star} \\ -M_S V_S^{\star} \end{pmatrix}, \tag{5.29}$$

where the equations for the fluid pressure come from the derivation of Equation (5.21) incorporating the additional λ term from Equation (5.28), the equations for the solid velocity come from Equation (5.27), and the middle row of equations come from substituting Equation (5.28) in to Equation (5.26). 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 [88] 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 (5.27) can be premultiplied by M_S^{-1}, giving

$$V_S^{n+1} = V_S^\star - M_S^{-1} \hat{C}^T f^{n+1} - M_S^{-1} \mathbf{J}^T \lambda, \tag{5.30}$$

Substituting Equation (5.30) into the middle set of rows in Equation (5.29) and multiplying the third row by $\hat{C} M_S^{-1}$ gives

$$\begin{pmatrix} \hat{\mathbf{P}}^{-1} + \hat{G}^T \beta^{-1} \hat{G} & -\hat{G}^T \beta^{-1} \mathbf{W}^T & 0 \\ -\mathbf{W} \beta^{-1} \hat{G} & \mathbf{W} \beta^{-1} \mathbf{W}^T + \mathbf{J} M_S^{-1} \mathbf{J}^T & \mathbf{J} M_S^{-1} \hat{C}^T \\ 0 & \hat{C} M_S^{-1} \mathbf{J}^T & I + \hat{C} M_S^{-1} \hat{C}^T \end{pmatrix} \begin{pmatrix} \hat{p}^{n+1} \\ \lambda \\ f^{n+1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{P}}^{-1} \hat{p}^a + \hat{G}^T \vec{u}^\star \\ \mathbf{J} V_S^\star - \mathbf{W} \vec{u}^\star \\ f^\star \end{pmatrix}. \tag{5.31}$$

After solving this symmetric positive-definite system we recover the solid velocities via Equation (5.30), and are careful to keep the momentum and kinetic energy exchange conservative as explained in detail in [31].

5.5 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 (5.31), 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.

5.5.1 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 5.14). We show some results in Figures 5.15, 5.16, 5.17 and 5.18.

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 5.13(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 5.13(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 5.13(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.

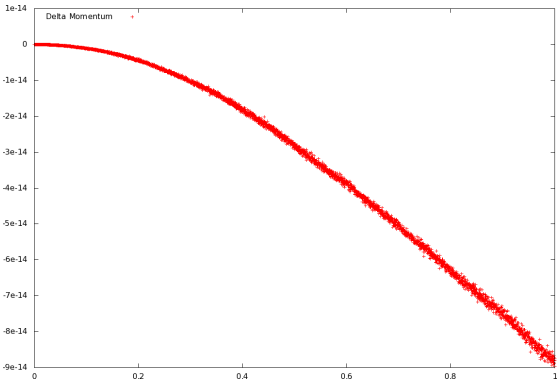
5.5.2 Stationary fluid damping a mass-spring system

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

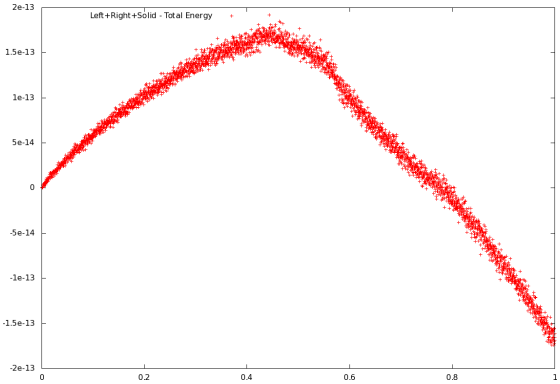
$$(\rho, u, p) = \left(4, 0, 10^6\right),$$

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 5.20, and plot the fluid pressure for a selection of times in Figure 5.19.

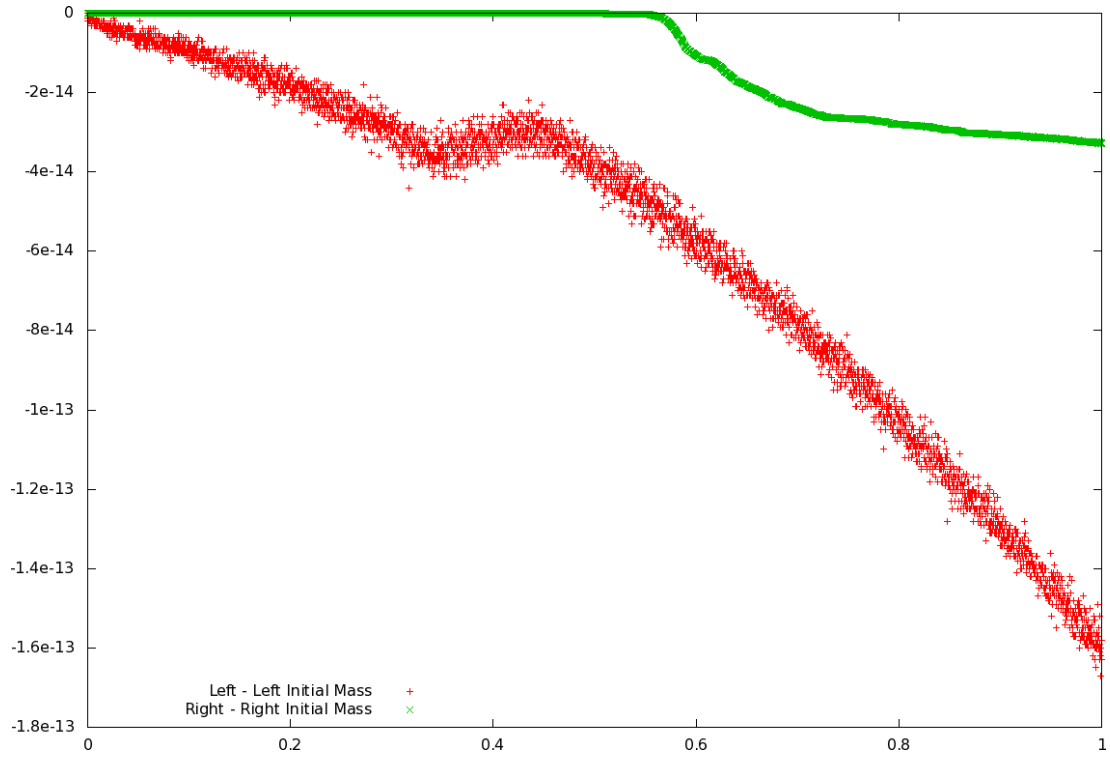
The convergence of the free end of the spring to the analytic solution at times $t = .0075s$ and $t = .008$ are shown in Figure 5.21, suggesting quadratic convergence to the analytic solution. This represents a significant improvement over [31], which only achieves convergence at a rate of 1.16 for this 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 [31] 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 5.3.1 and 5.3.2.



(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.

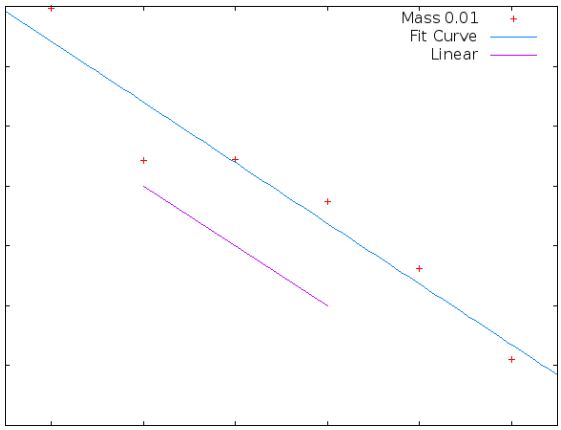


(b) Error in conservation of energy of the system, computed as $\sum_i E_i V_i + \frac{1}{2} M_S V_S^2$. As the velocity of the fluid at the boundary remains zero, no work is done and the total energy of the system is constant.

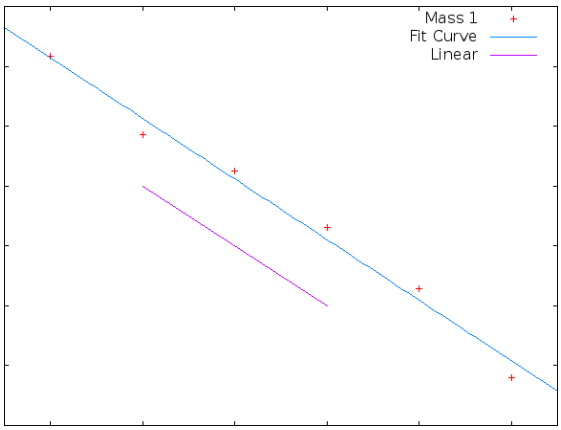


(c) Error in conservation of mass for the left (green) and right (red) sides of the domain.

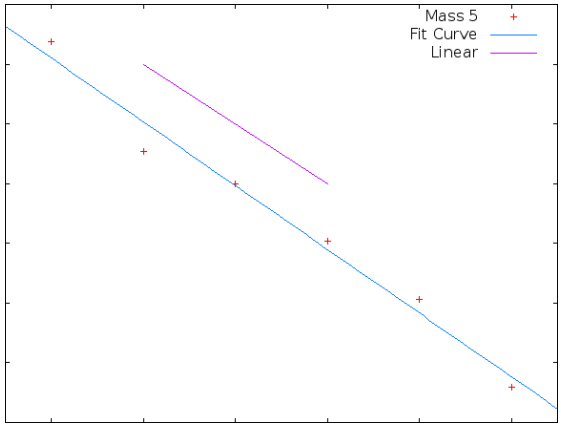
Figure 5.13: Conservation error of a Sod shock tube interacting with a rigid point-mass, with $M_S = 1$. Note that the scale in the dependent axis is 10^{-14}, showing that all of the errors in conservation lie in the round-off error.



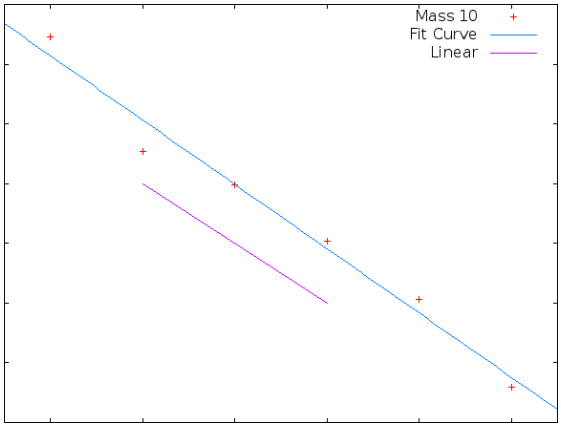
(a) $M_S = .01$ converges at 1.02.



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

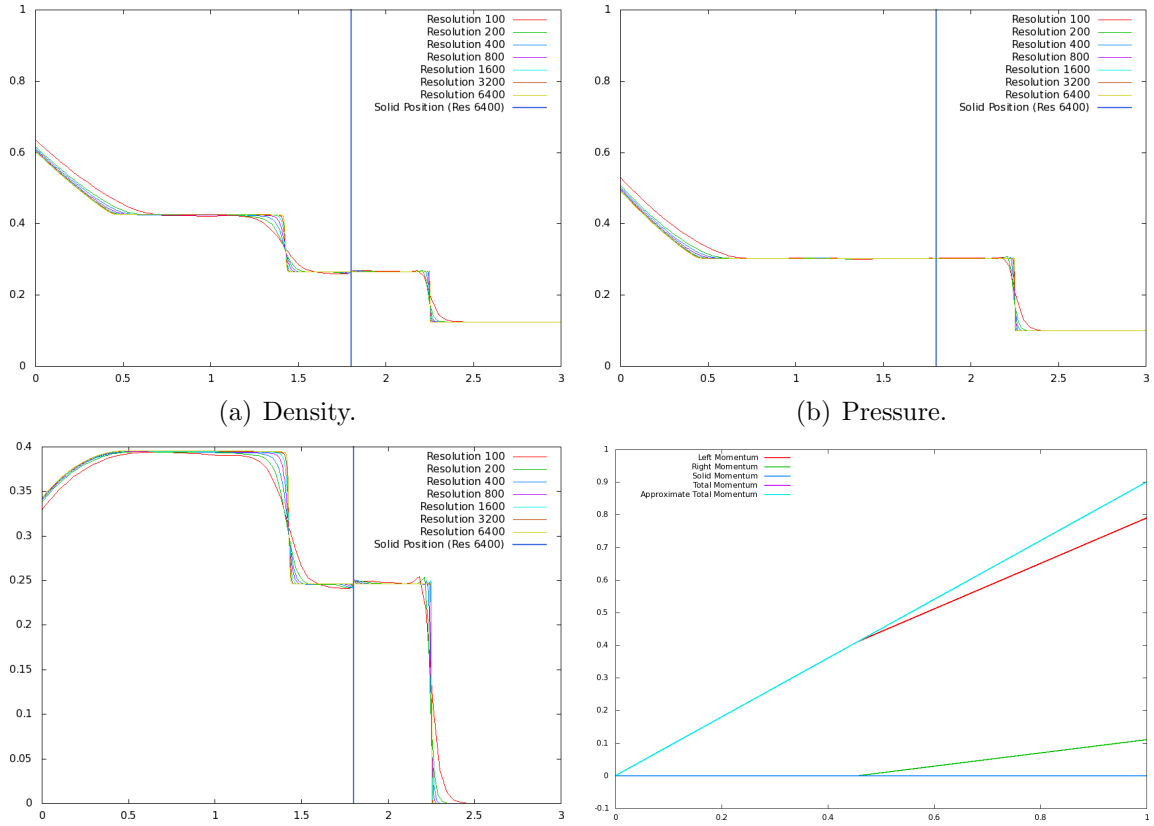


(c) $M_S = 5$ converges at 1.07.



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

Figure 5.14: 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.



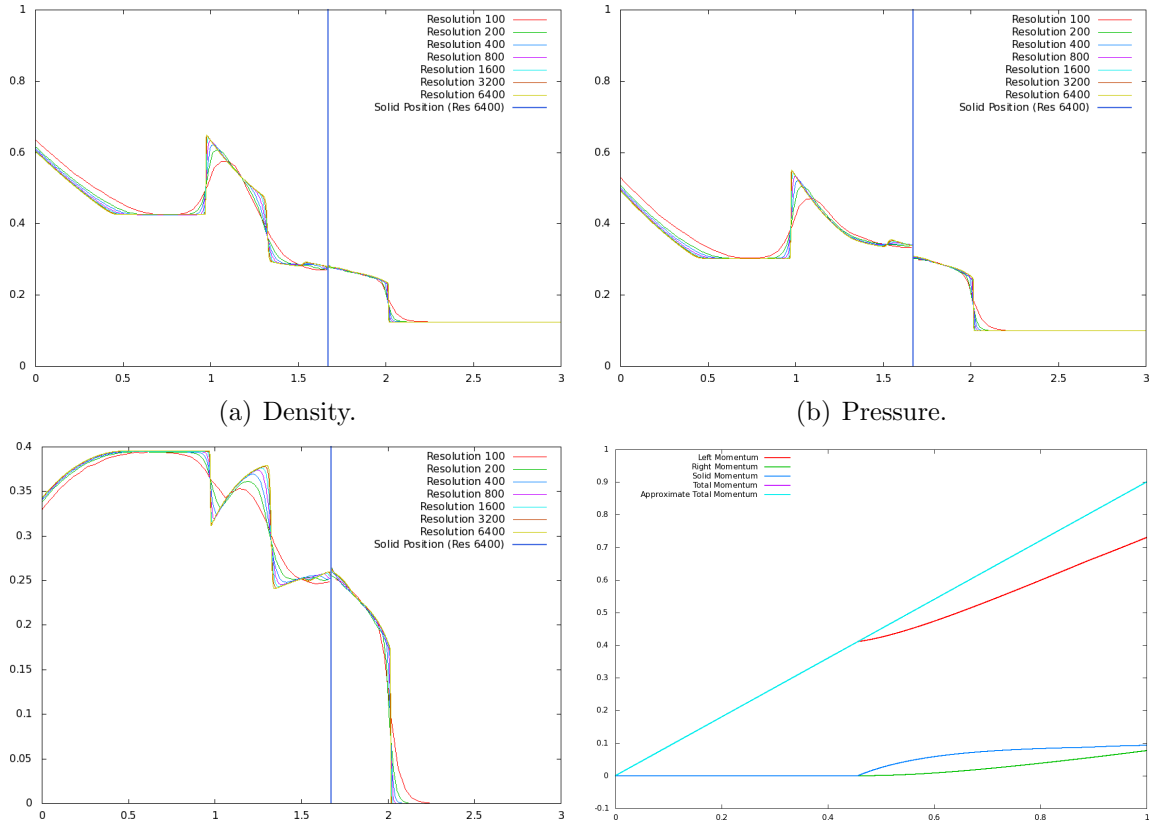
(a) Density.

(b) Pressure.

(c) Momentum.

(d) Time history of momentum in the system.

Figure 5.15: 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.



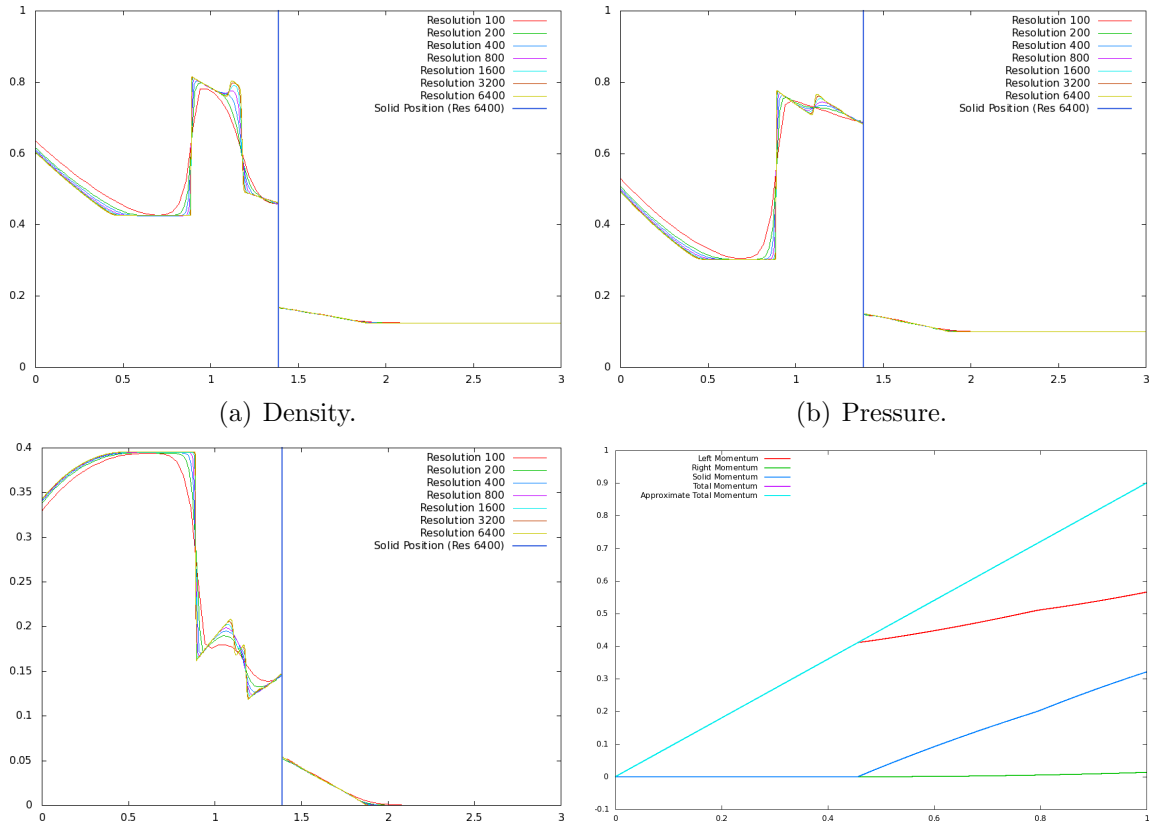
(a) Density.

(b) Pressure.

(c) Momentum.

(d) Time history of momentum in the system.

Figure 5.16: 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.



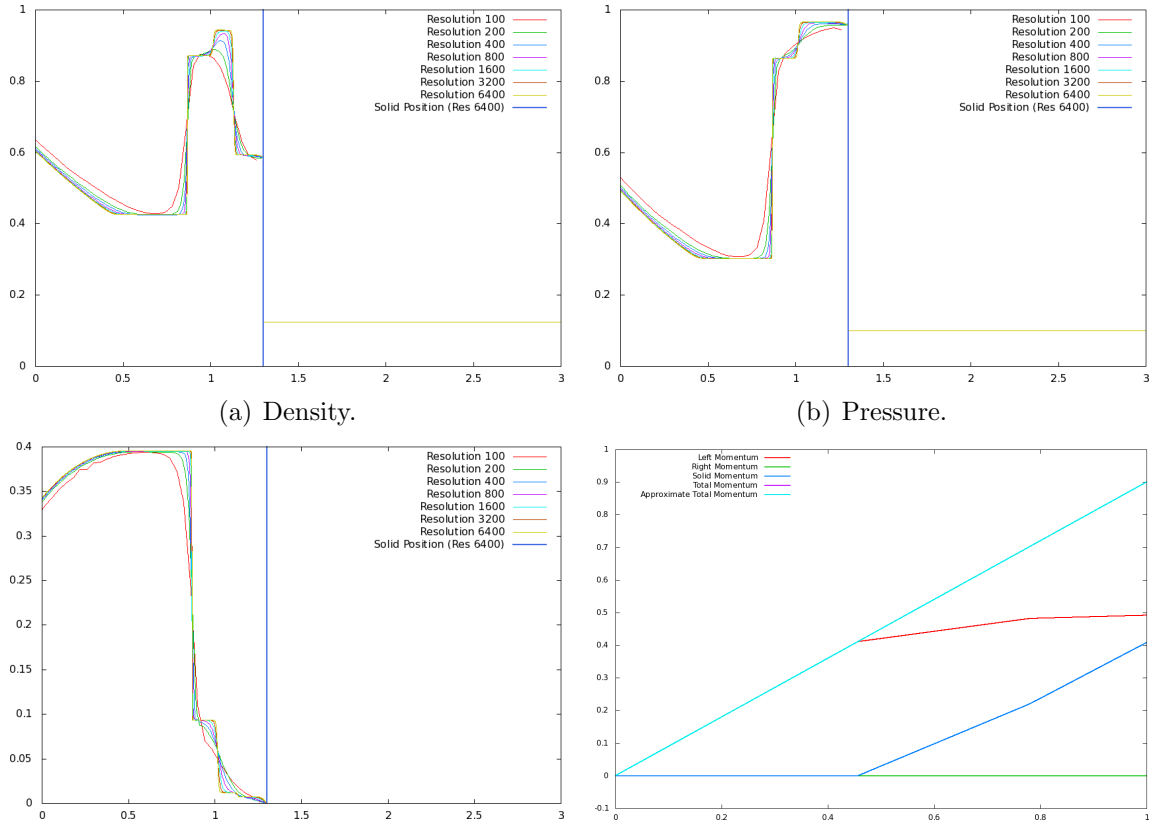
(a) Density.

(b) Pressure.

(c) Momentum.

(d) Time history of momentum in the system.

Figure 5.17: 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.



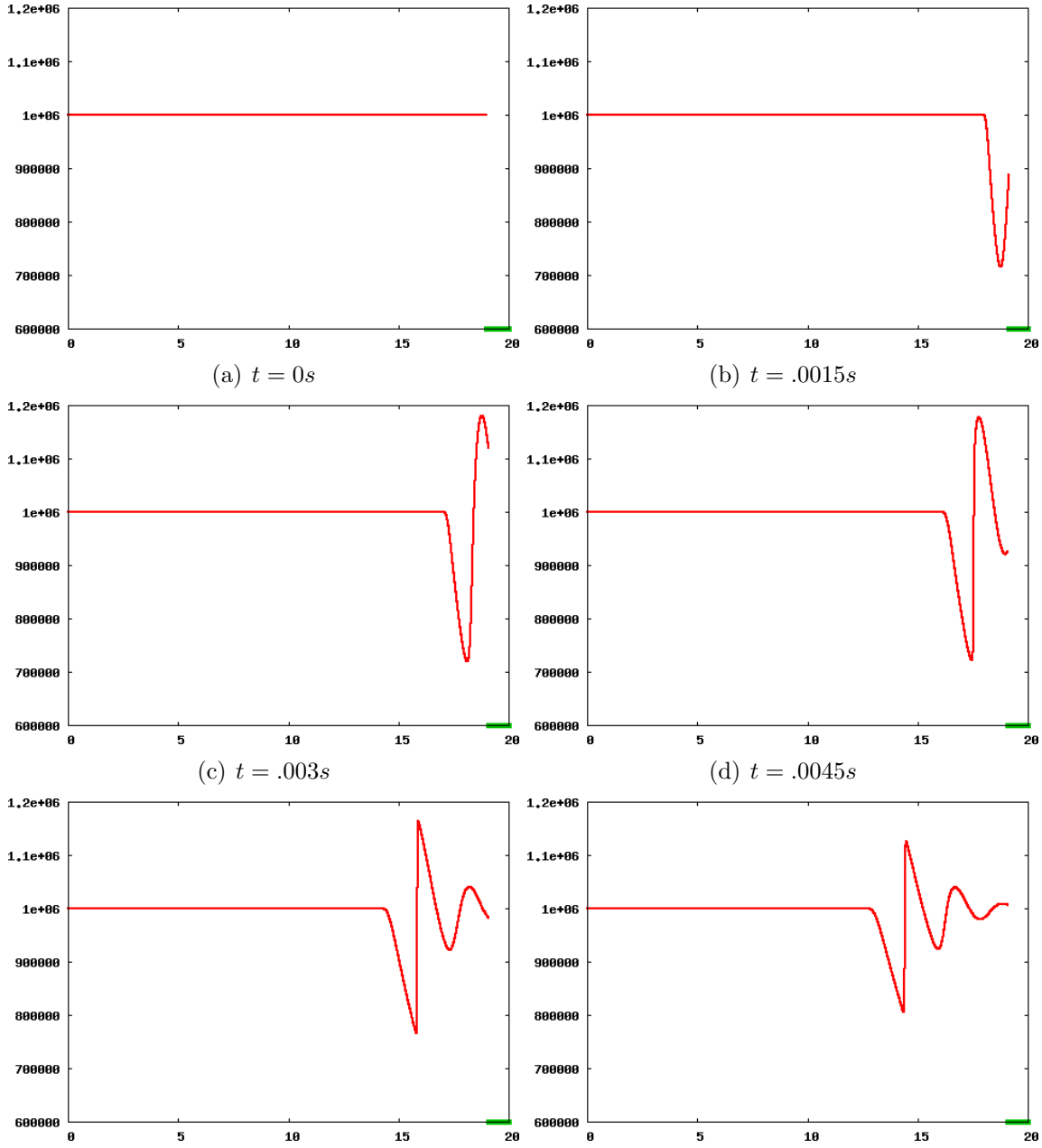
(a) Density.

(b) Pressure.

(c) Momentum.

(d) Time history of momentum in the system.

Figure 5.18: 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 5.19: Pressure of the flow field for Section 5.5.2, for a selection of times.

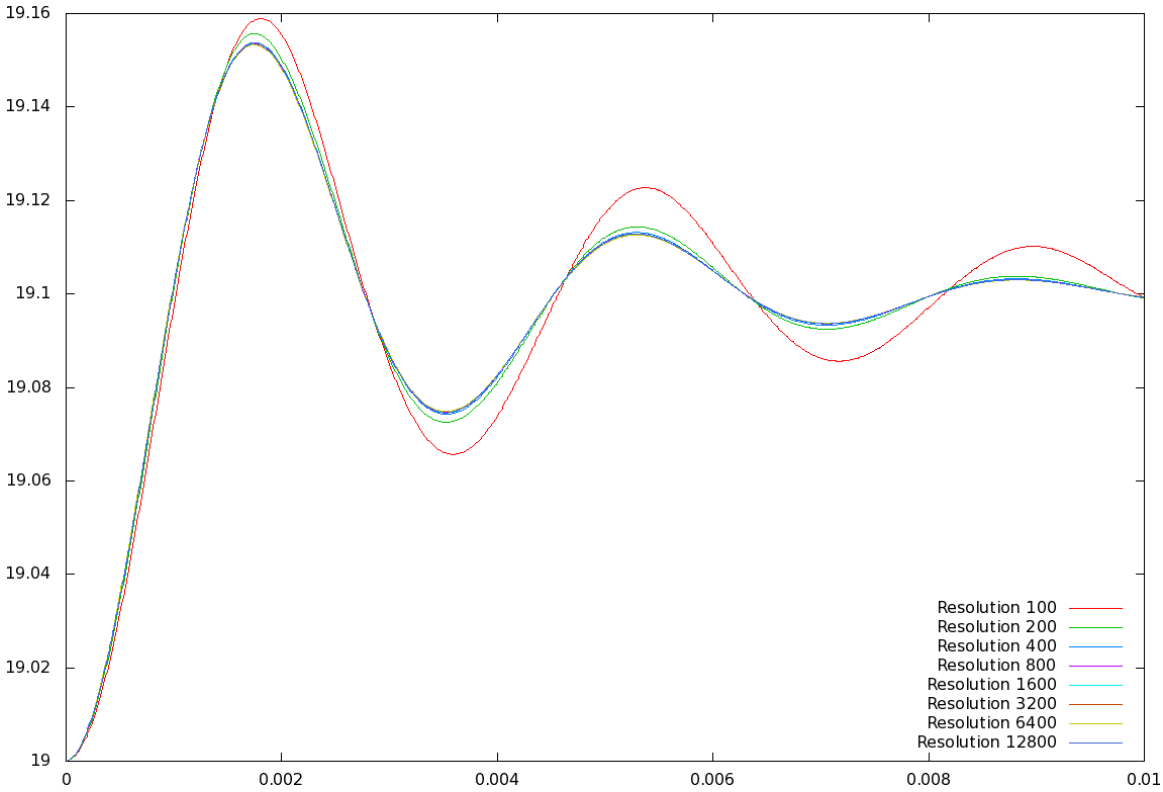
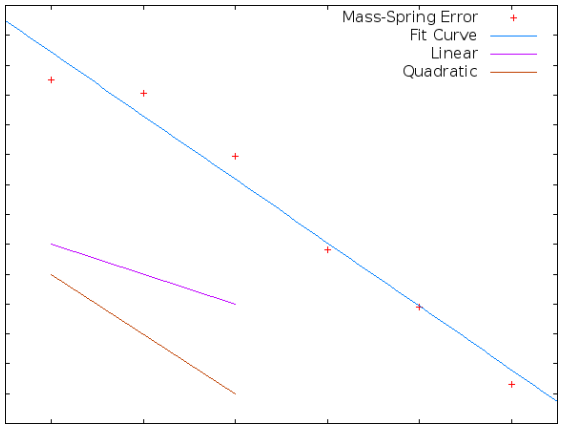
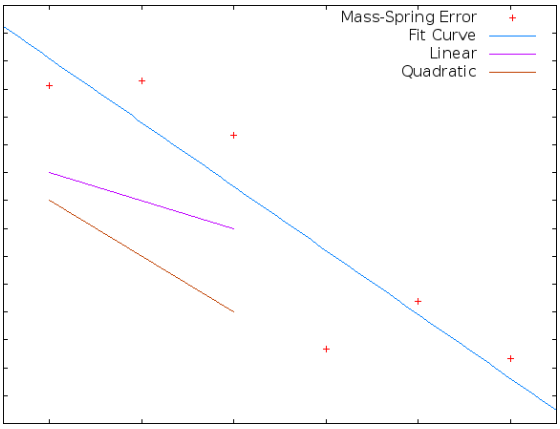


Figure 5.20: Position of the free end of the spring for Section 5.5.2, as a function of time.



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



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

Figure 5.21: Convergence of the free end of the spring to the analytic solution, for Section 5.5.2. 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.

5.5.3 Two-dimensional examples

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

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

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 5.22 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.

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 5.23 and 5.24 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.

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 5.25 and 5.26 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 5.22(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).

Asymmetric shock reflection off a rigid cylinder

Similar to [31], 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 5.22(b), and the flow field converges as shown in Figure 5.27. The time evolution is shown in

Figures 5.28 and 5.29, for pressure and density respectively.

Asymmetric shock reflection off a thin rigid shell

The cylinder from Section 5.5.3 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 5.30 and 5.31, respectively. Under grid resolution, the position of the center of the cylinder converges at a rate of 1.204, shown in Figure 5.22(c), and the flow field converges as shown in Figure 5.32. The total fluid state inside the hollow cylinder is shown in Figure 5.33; 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.

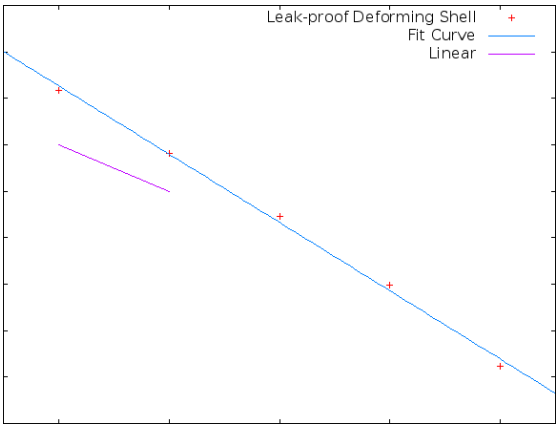
Planar shock interacting with a constrained, immersed deforming thin shell

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.

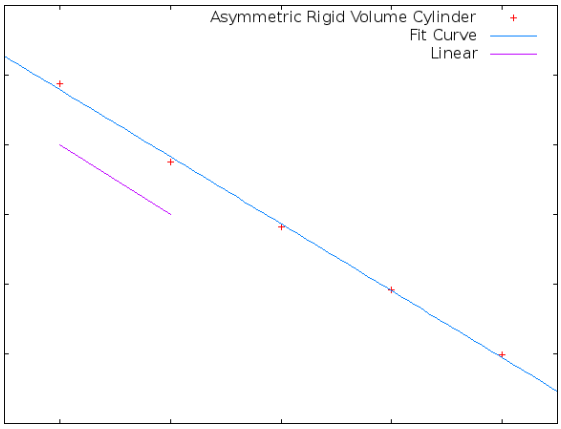
The shock reflects off of the deforming body and causes the deforming shell to buckle to the right, while rarefaction fans form around the constrained nodes as flow passes by. As the deforming body reaches its right-most location and bends back a

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 5.34 and Figure 5.35. Under grid refinement (shown in Figure 5.36), note the regime change from laminar flow to separating flow. This is a result of artificial viscosity vanishing under grid refinement and simulating inviscid flows; and so as the grid becomes more refined the observed Reynolds number goes up.

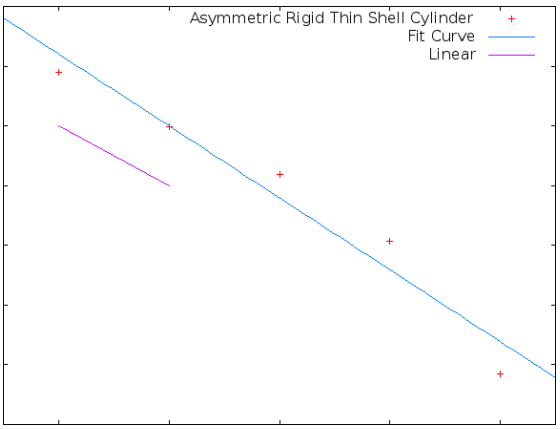
The position of the center-of-mass of the deforming body converges at a rate of 1.342, shown in Figure 5.22(d), and the flow field converges as shown in Figure 5.36. This is likely aided by the constrained nodes, whose influence does not change under refinement. No material flows through the thin solid structure.



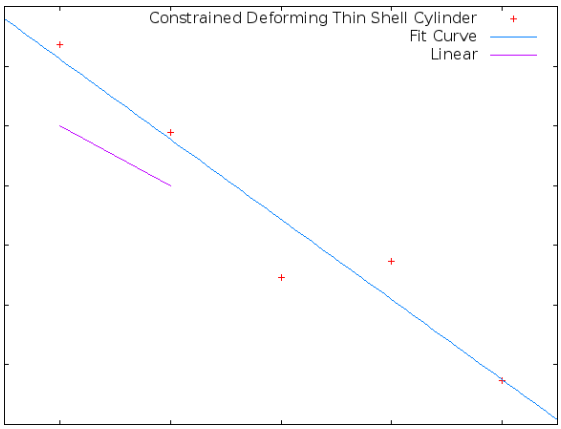
(a) Example from Section 5.5.3 converges at a rate of 1.471.



(b) Example from Section 5.5.3 converges at a rate of 0.962.

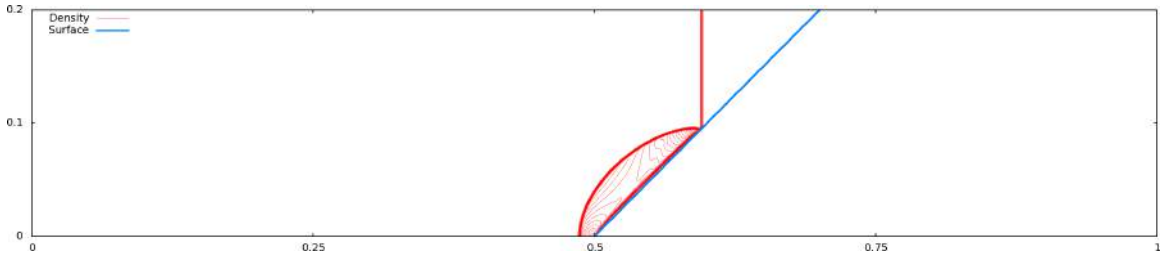


(c) Example from Section 5.5.3 converges at a rate of 1.204.

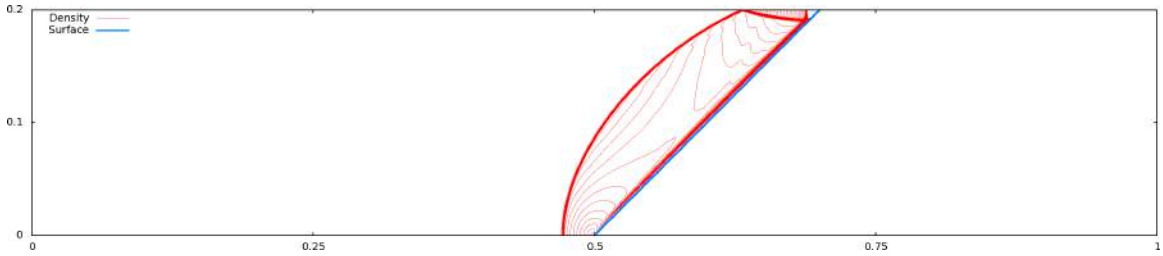


(d) Example from Section 5.5.3 converges at a rate of 1.342.

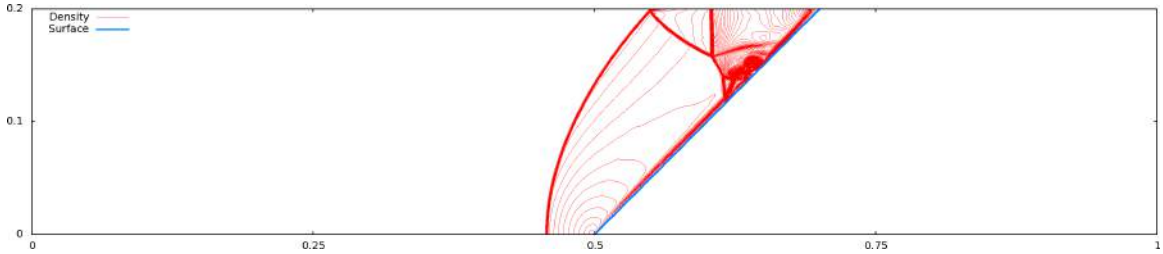
Figure 5.22: Convergence rate for the center of the dynamic structure is computed using the L^2 error in position against the position from a highly refined solution. The rate of convergence is computed as the slope of the best-fit line on the log-log scale of the L^2 error as a function of grid refinement, and this best-fit line is shown in blue. This can be compared with a reference line of slope 1, shown in purple.



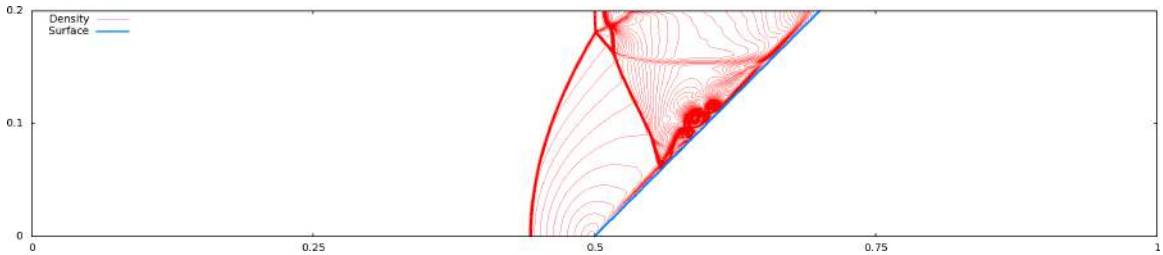
(a) $t = .04s$



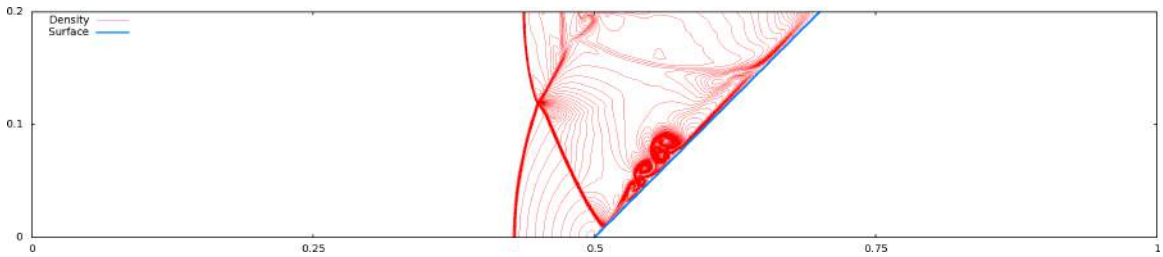
(b) $t = .08s$



(c) $t = .12s$

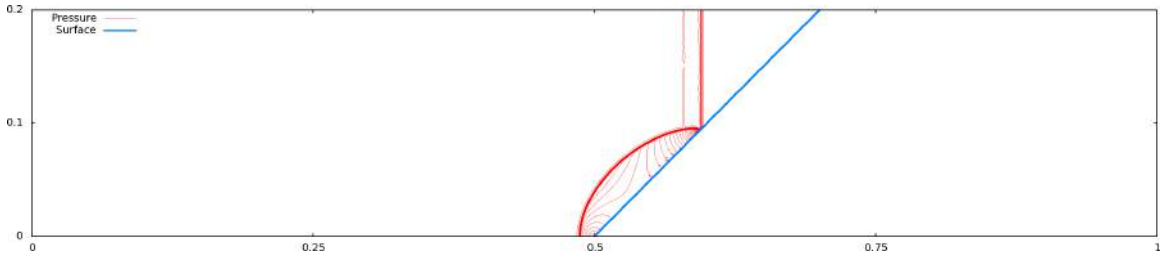


(d) $t = .16s$

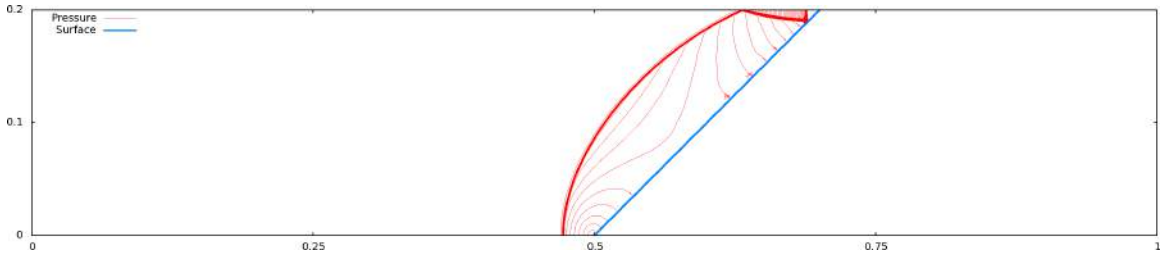


(e) $t = .2s$

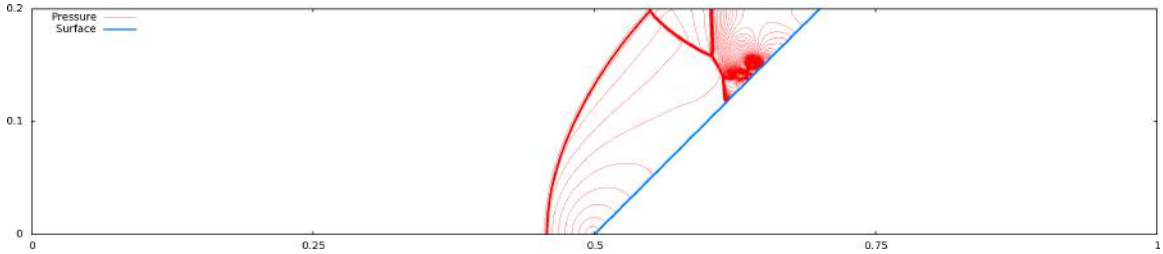
Figure 5.23: Time evolution of density in the example described in Section 5.5.3, on a 2560×512 grid.



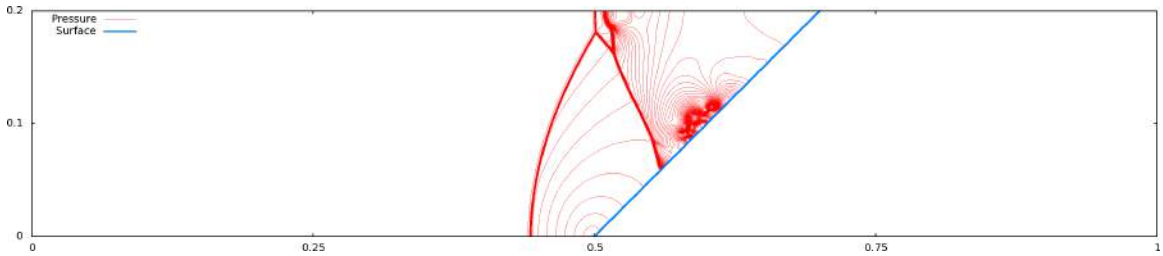
(a) $t = .04s$



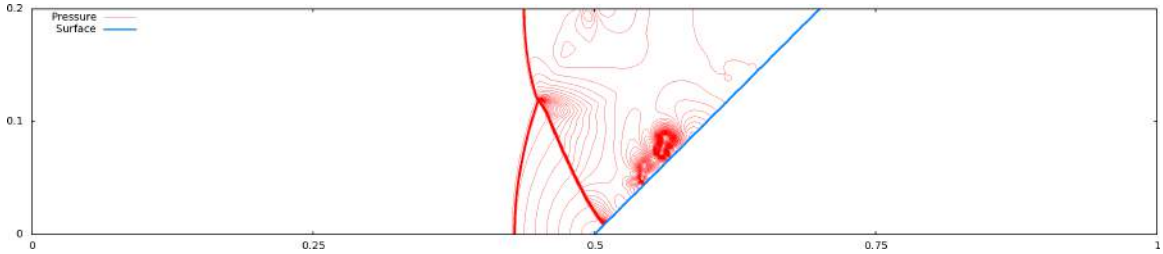
(b) $t = .08s$



(c) $t = .12s$

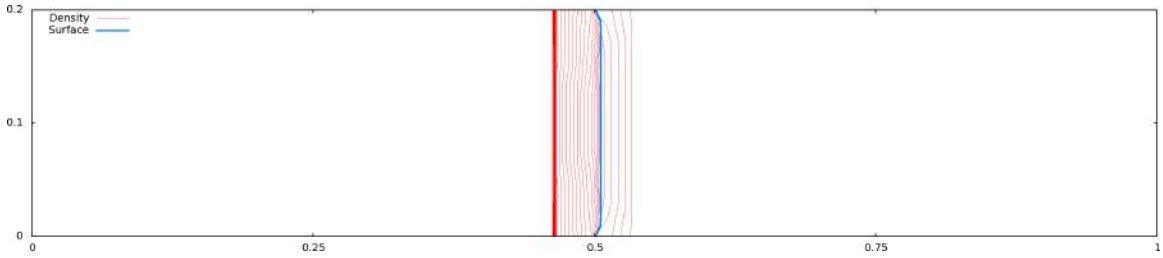


(d) $t = .16s$

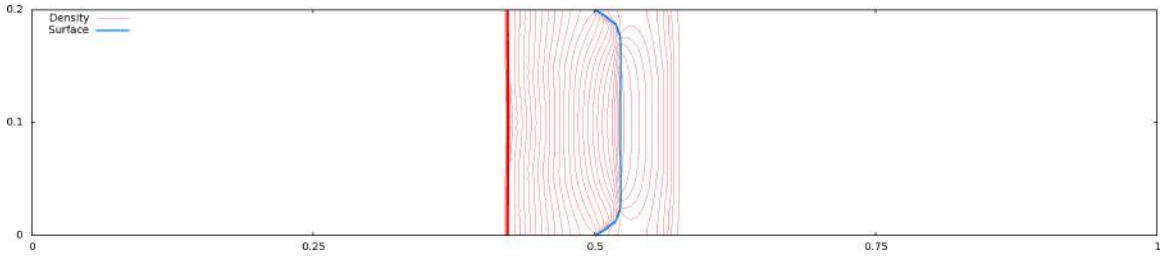


(e) $t = .2s$

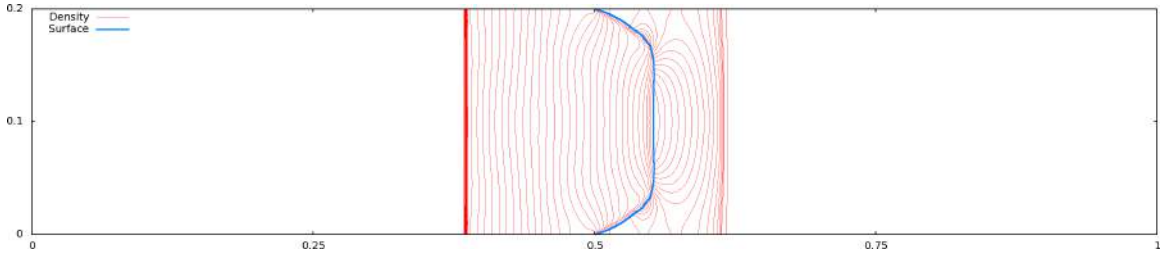
Figure 5.24: Time evolution of pressure in the example described in Section 5.5.3, on a 2560×512 grid.



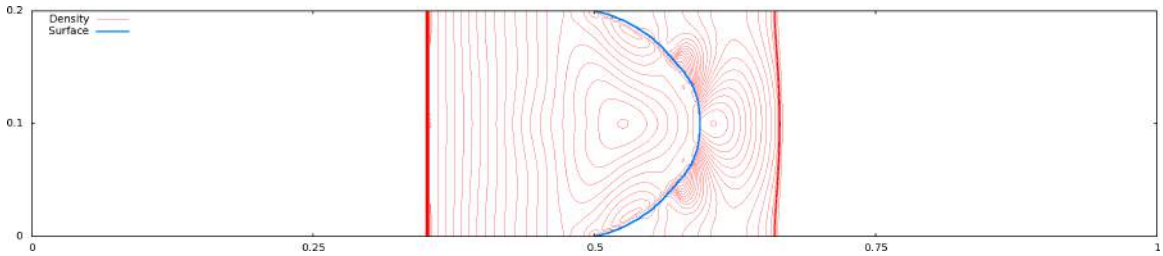
(a) $t = .04s$



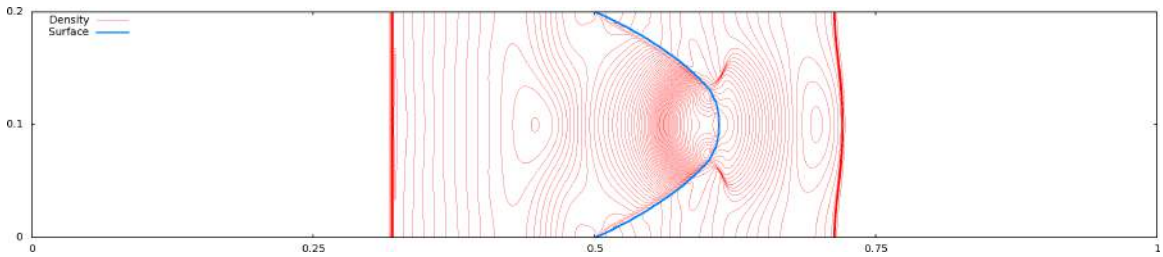
(b) $t = .08s$



(c) $t = .12s$

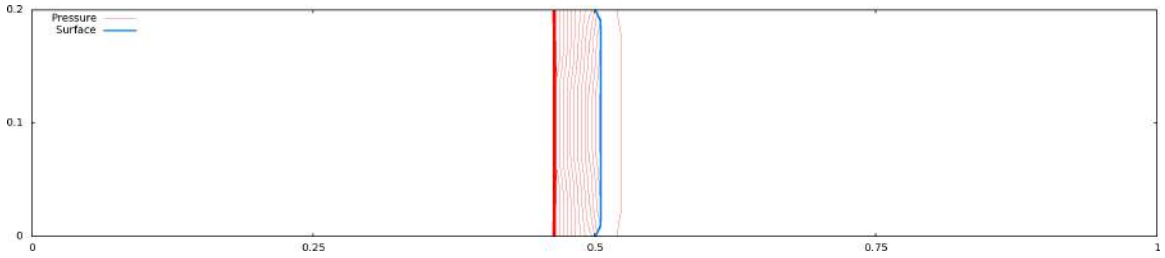


(d) $t = .16s$

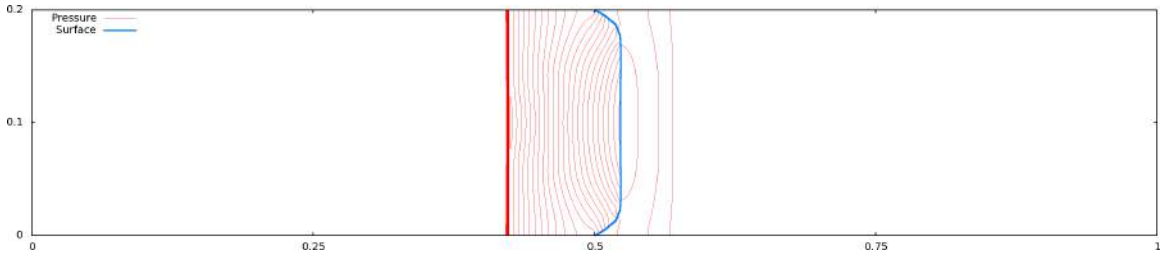


(e) $t = .2s$

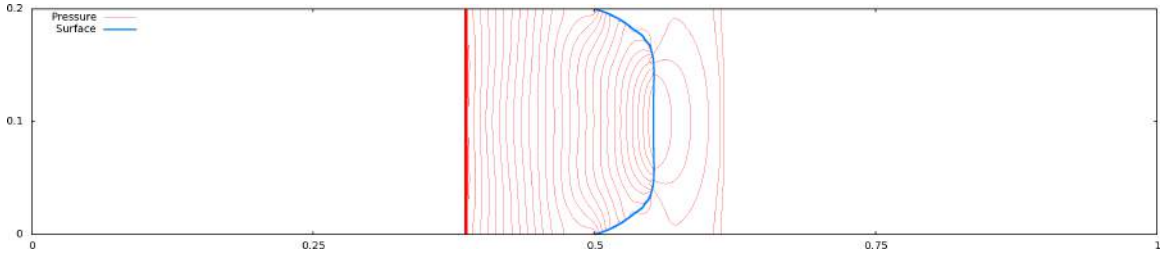
Figure 5.25: Time evolution of density in the example described in Section 5.5.3, on a 2560×512 grid.



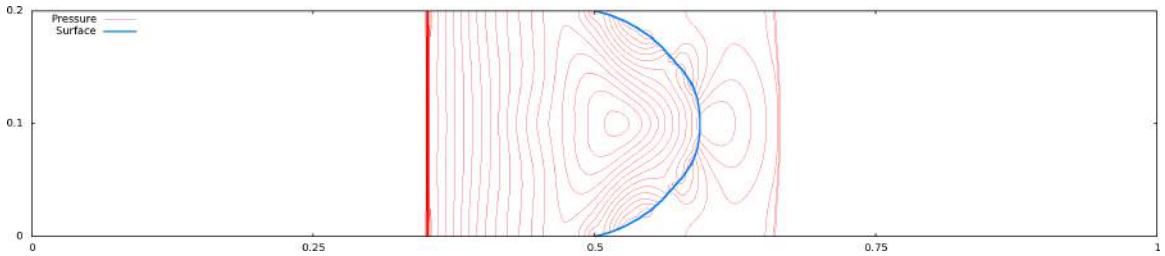
(a) $t = .04s$



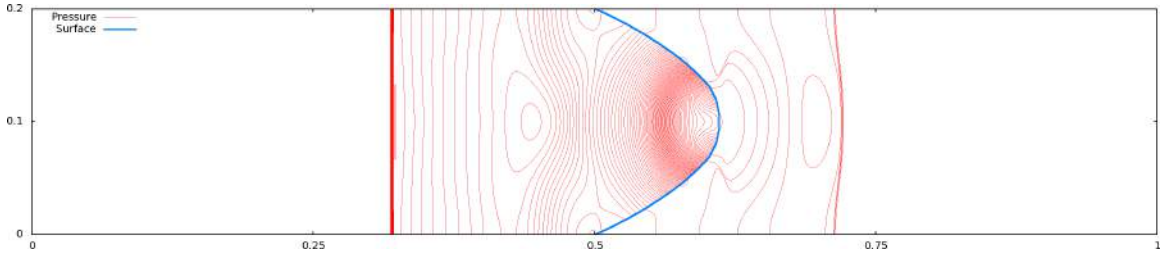
(b) $t = .08s$



(c) $t = .12s$

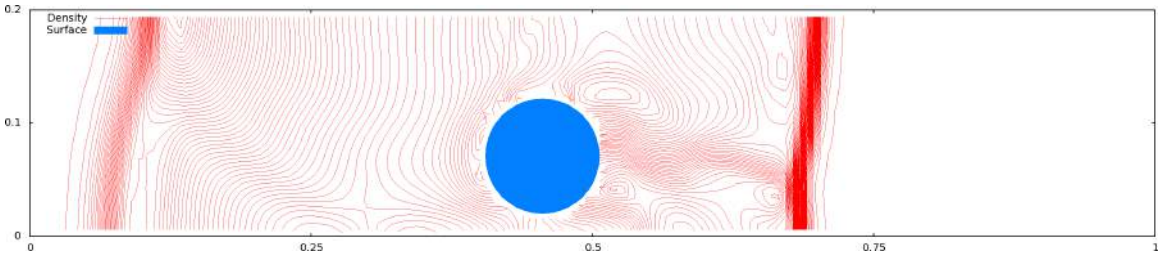


(d) $t = .16s$

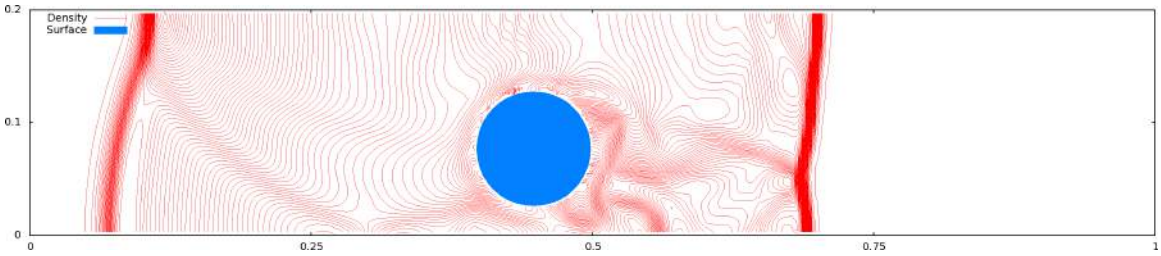


(e) $t = .2s$

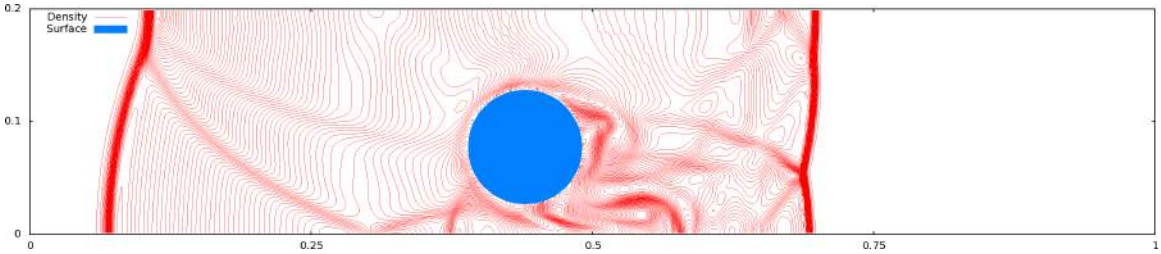
Figure 5.26: Time evolution of pressure in the example described in Section 5.5.3, on a 2560×512 grid.



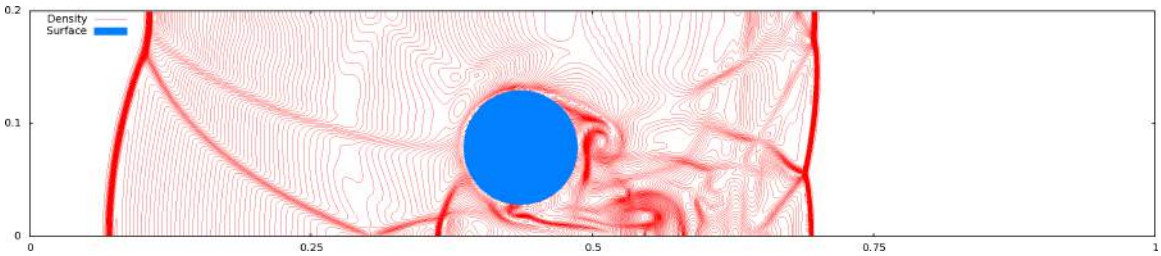
(a) Resolution 160×32



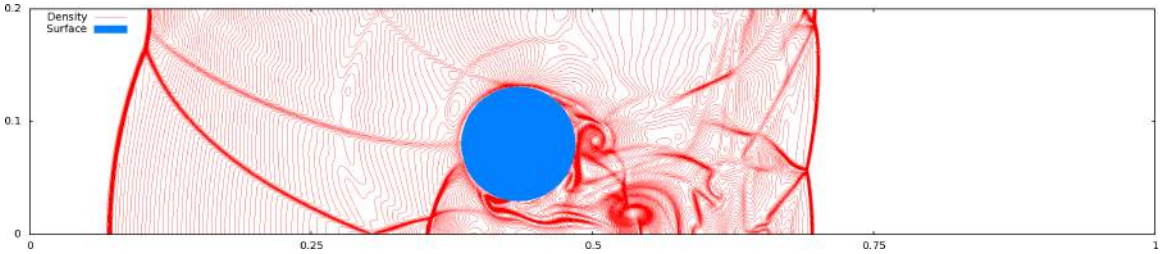
(b) Resolution 320×64



(c) Resolution 640×128

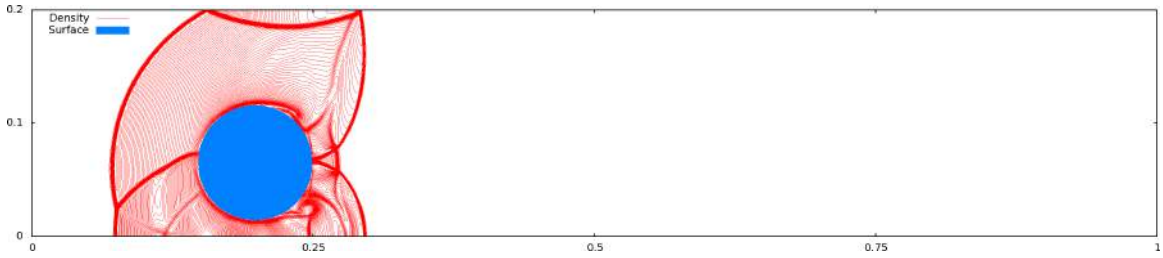


(d) Resolution 1280×256

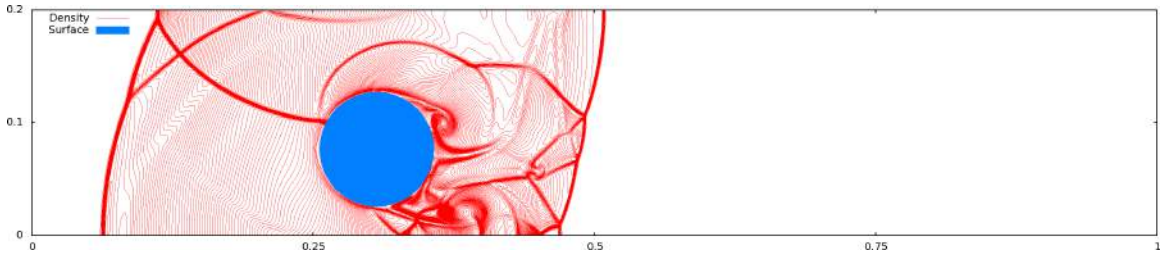


(e) Resolution 2560×512

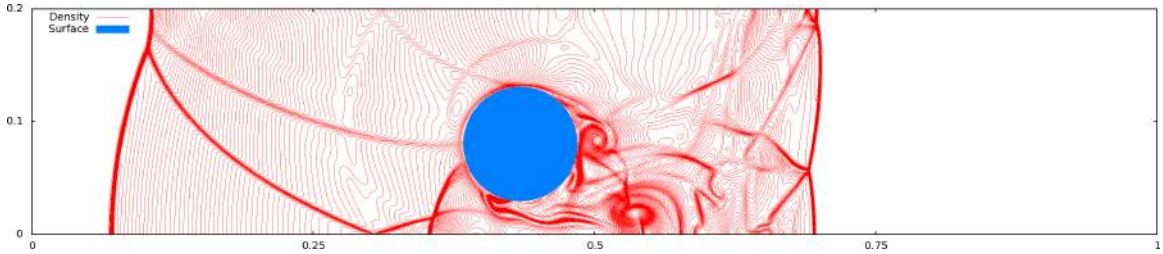
Figure 5.27: Grid convergence of the example described in Section 5.5.3, at time $t = .21s$.



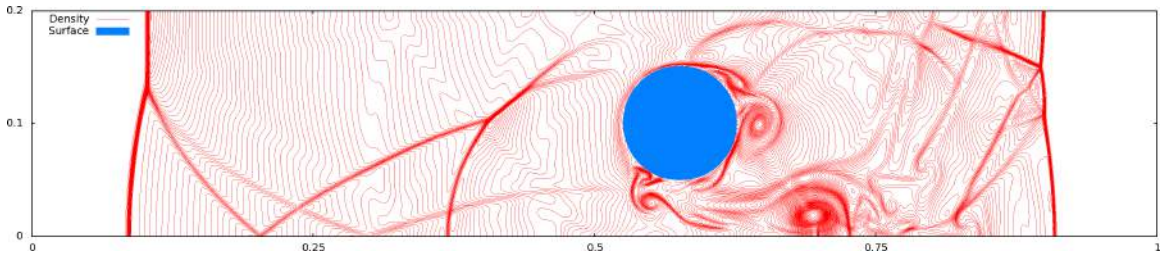
(a) $t = .07s$



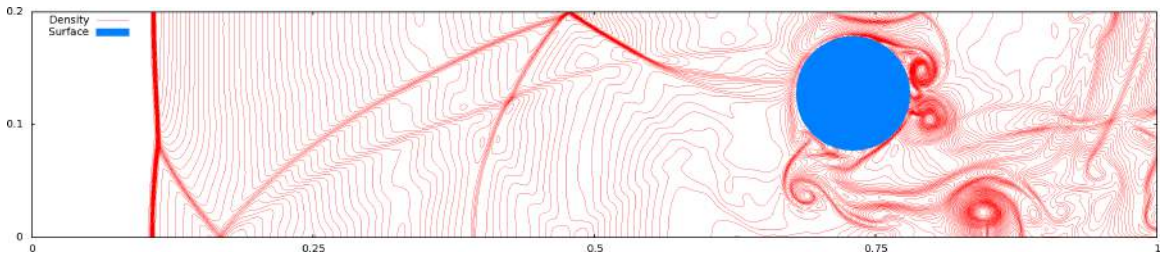
(b) $t = .14s$



(c) $t = .21s$

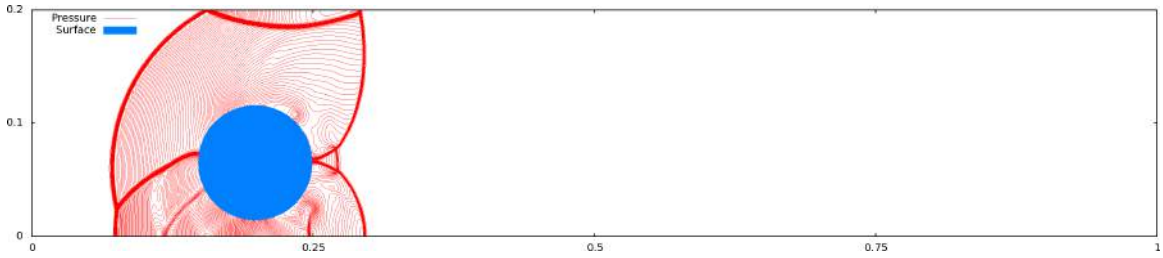


(d) $t = .28s$

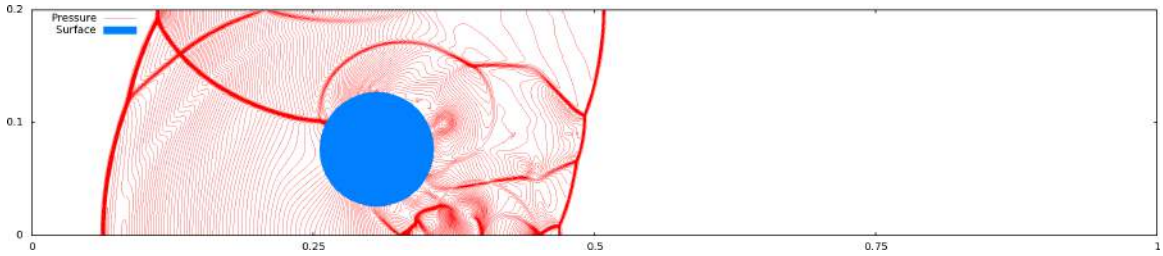


(e) $t = 35s$

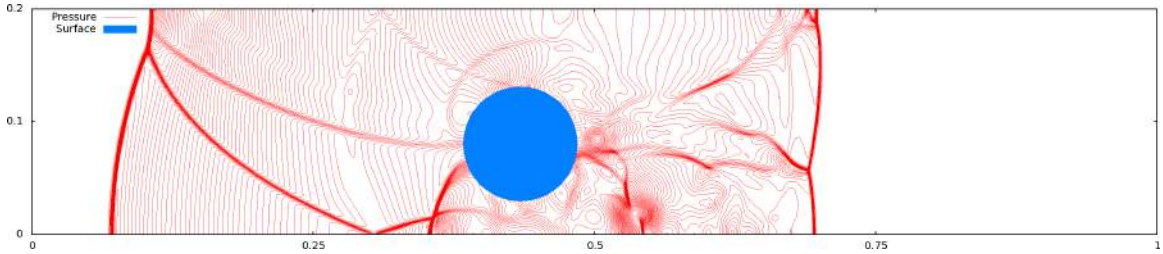
Figure 5.28: Time evolution of density in the example described in Section 5.5.3, on a 2560×512 grid.



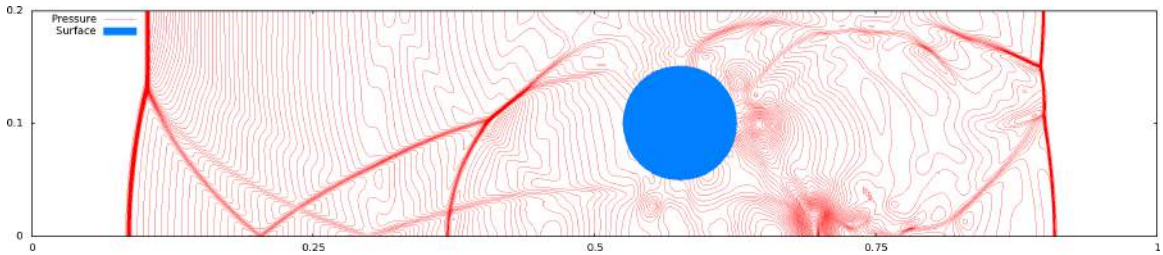
(a) $t = .07s$



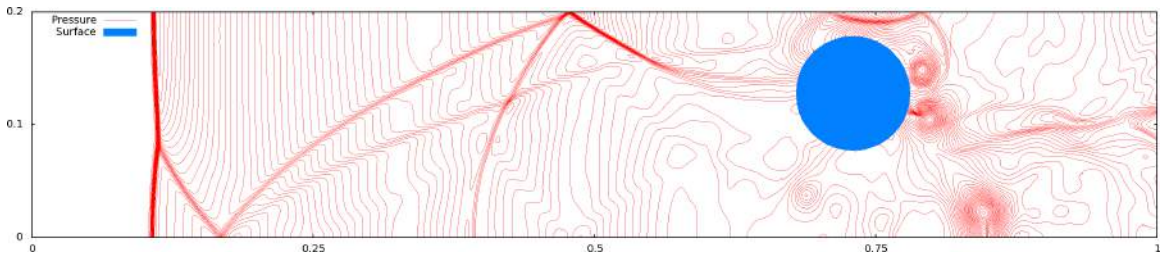
(b) $t = .14s$



(c) $t = .21s$

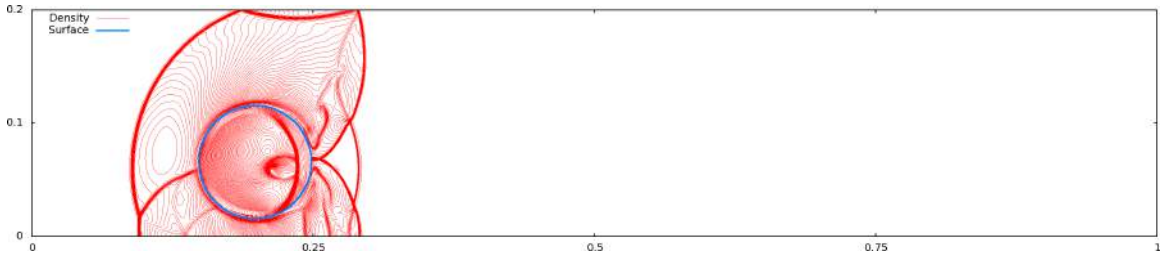


(d) $t = .28s$

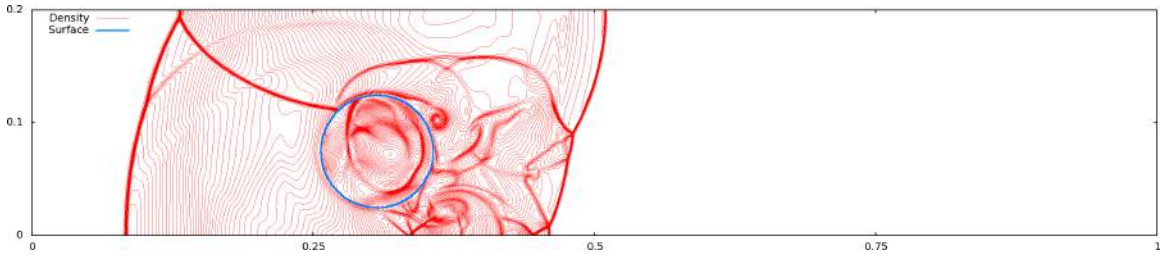


(e) $t = 35s$

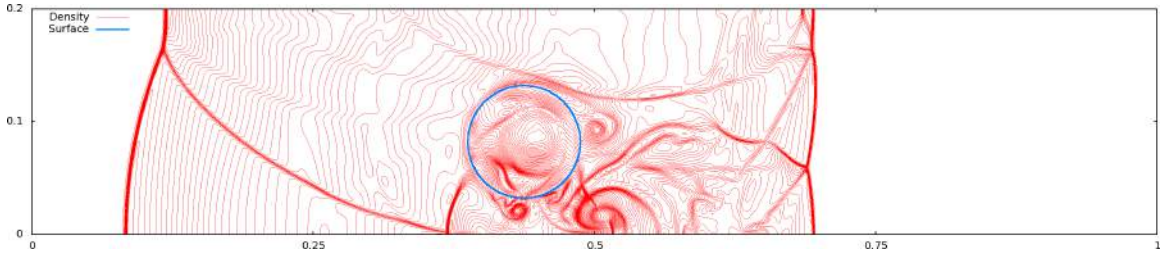
Figure 5.29: Time evolution of pressure in the example described in Section 5.5.3, on a 2560×512 grid.



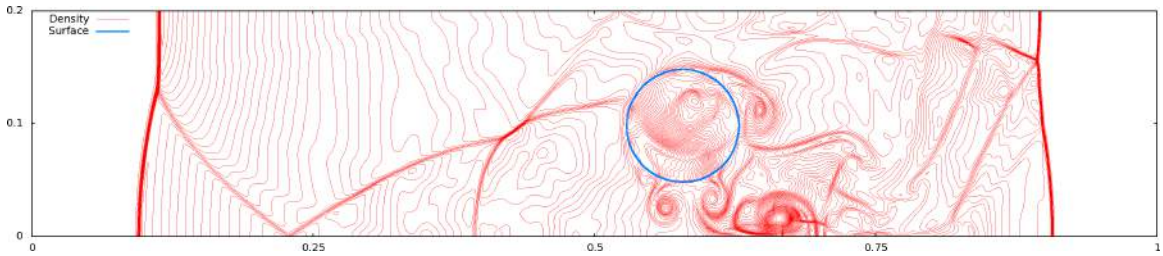
(a) $t = .07s$



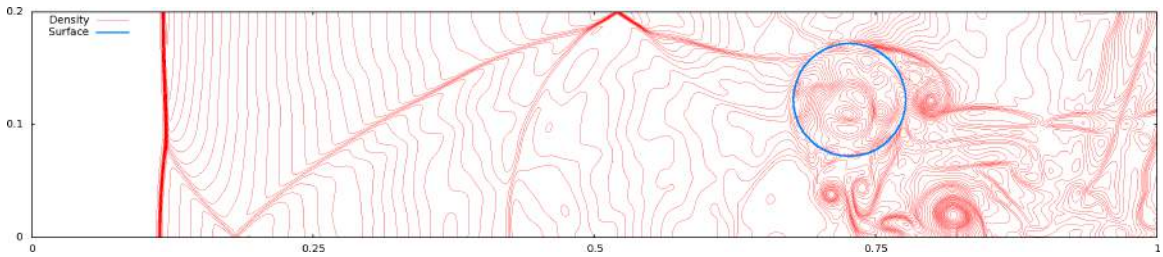
(b) $t = .14s$



(c) $t = .21s$

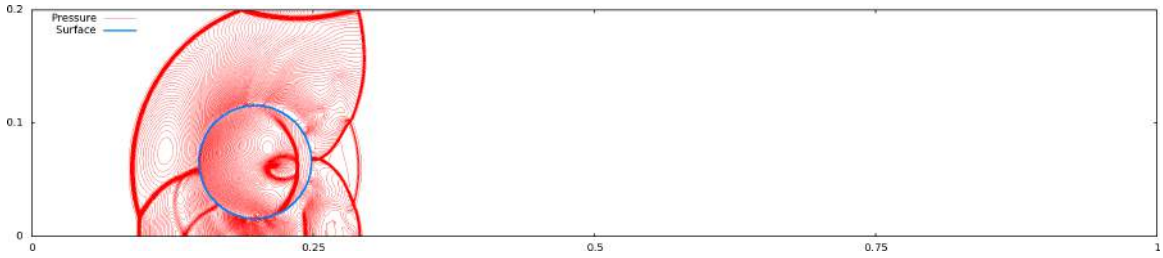


(d) $t = .28s$

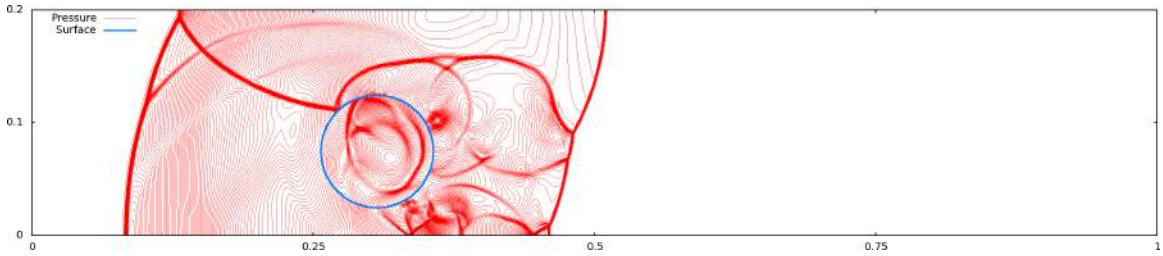


(e) $t = 35s$

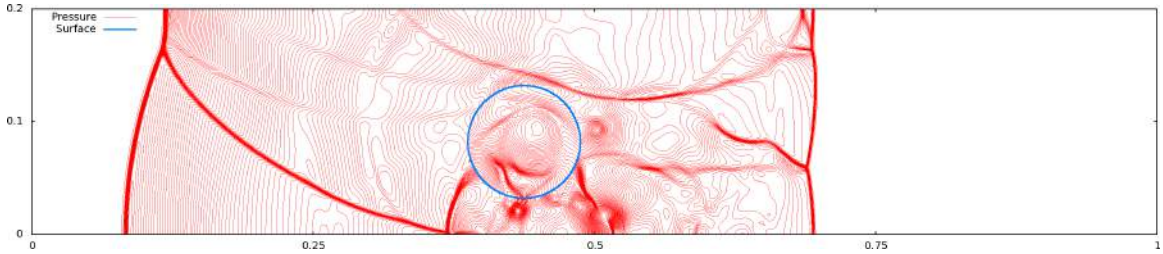
Figure 5.30: Time evolution of density in the example described in Section 5.5.3, on a 2560×512 grid.



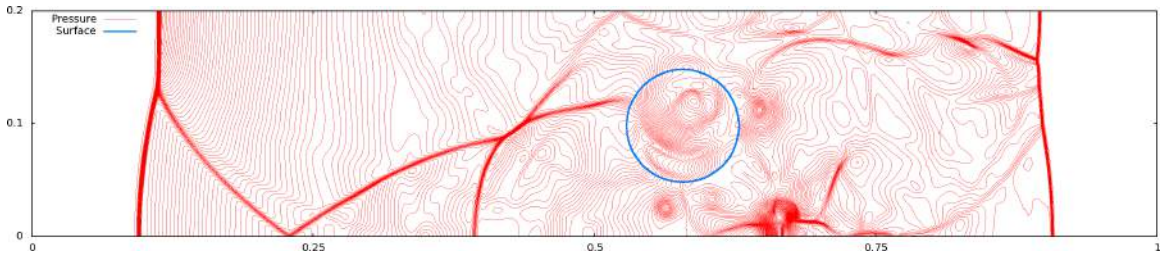
(a) $t = .07s$



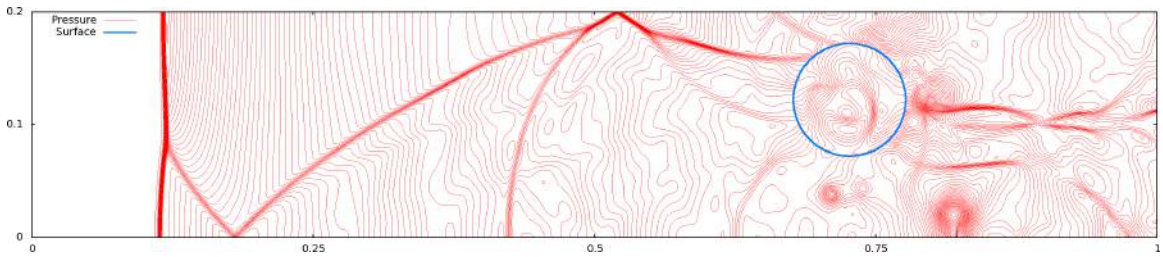
(b) $t = .14s$



(c) $t = .21s$

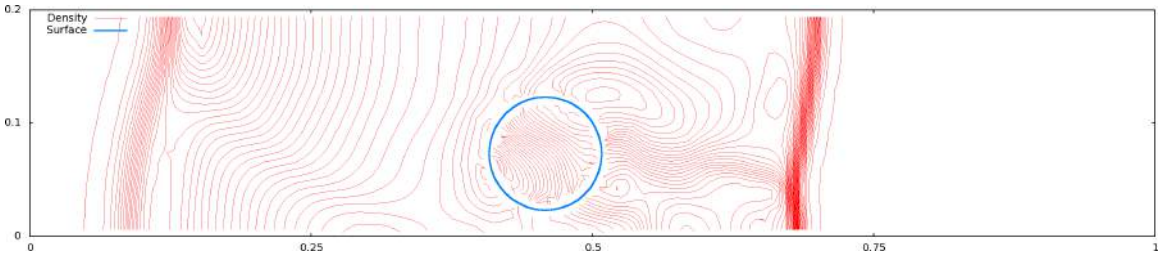


(d) $t = .28s$

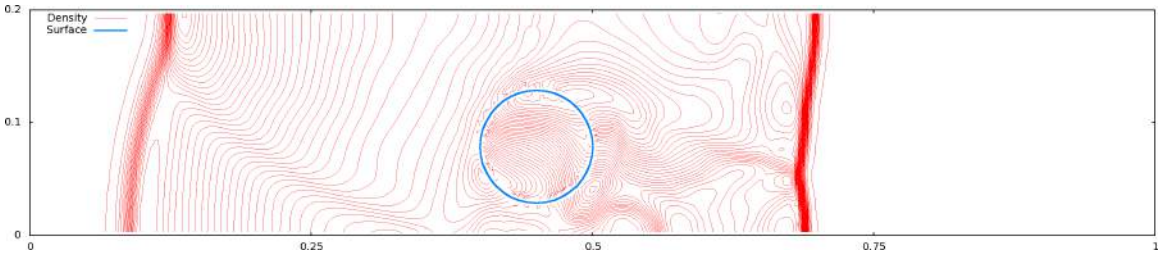


(e) $t = 35s$

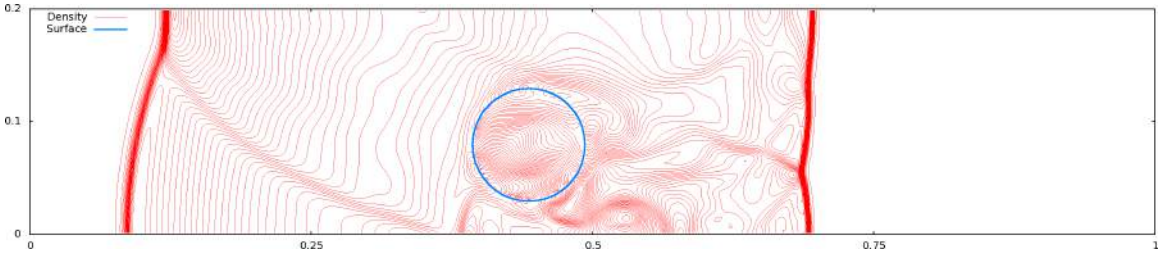
Figure 5.31: Time evolution of pressure in the example described in Section 5.5.3, on a 2560×512 grid.



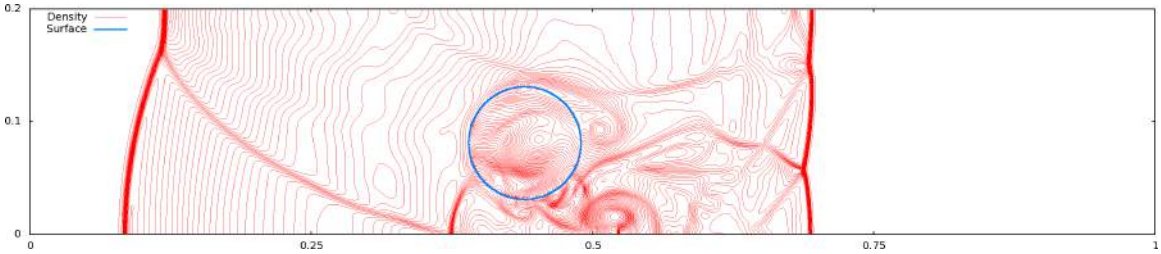
(a) Resolution 160×32



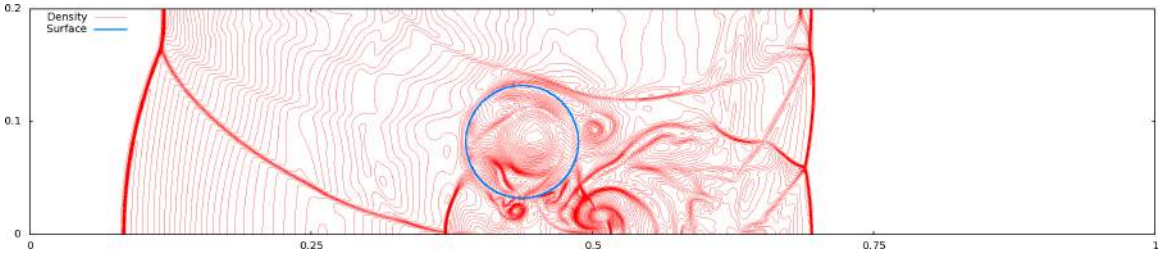
(b) Resolution 320×64



(c) Resolution 640×128

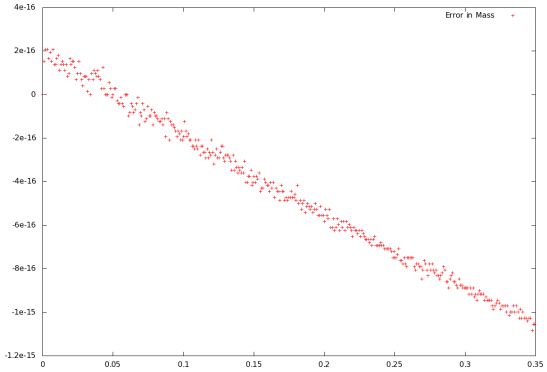


(d) Resolution 1280×256

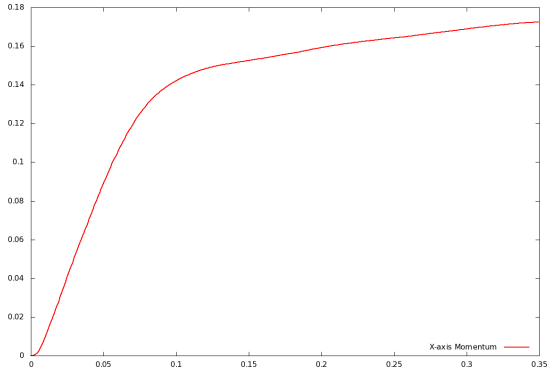


(e) Resolution 2560×512

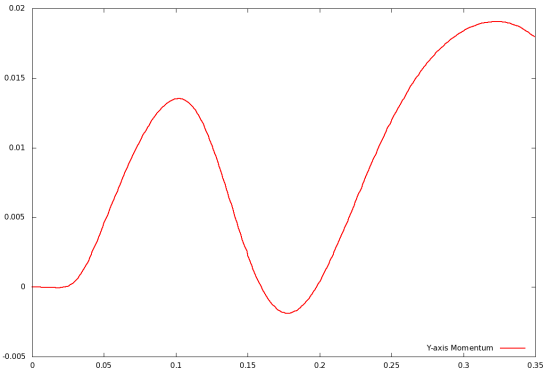
Figure 5.32: Grid convergence of the example described in Section 5.5.3, at time $t = .21s$.



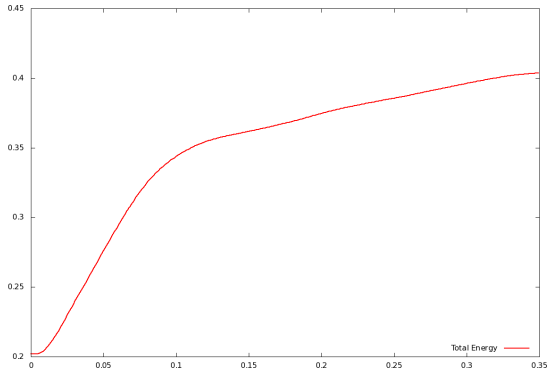
(a) Time evolution of mass error from the initial configuration.



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

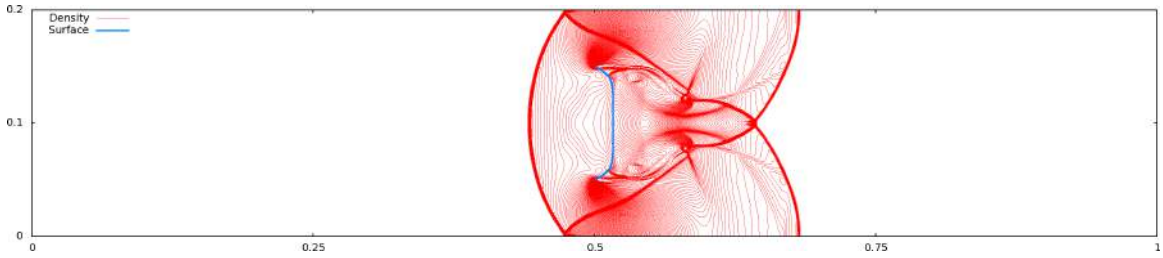


(c) Time evolution of y-axis fluid momentum.

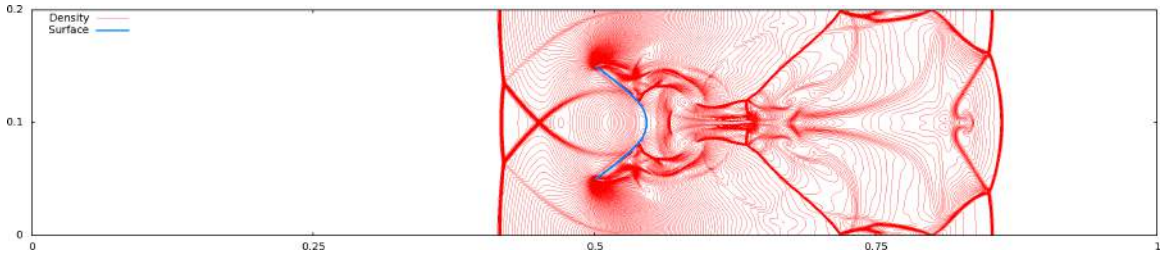


(d) Time evolution of total fluid energy.

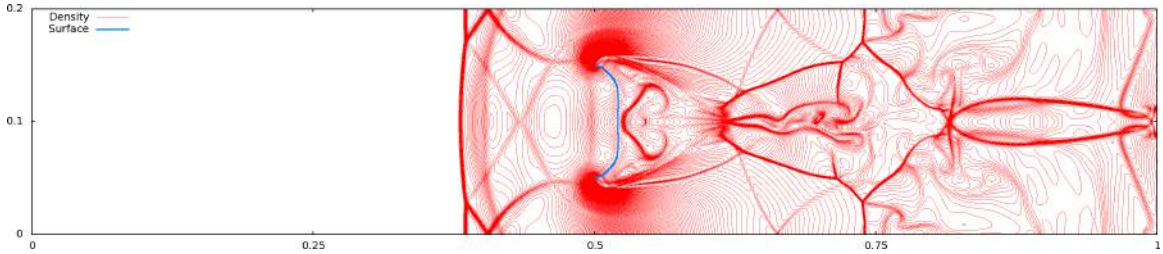
Figure 5.33: Time evolution of conserved material inside the hollow rigid cylinder from Section 5.5.3. In (a), we show the time history of the error in total mass inside the cylinder, while (b), (c) and (d) show time history of fluid momentum and energy inside the cylinder. Note that the scale in the dependent axis of (a) is 10^{-16}, showing that the errors in conservation lie well within round-off error.



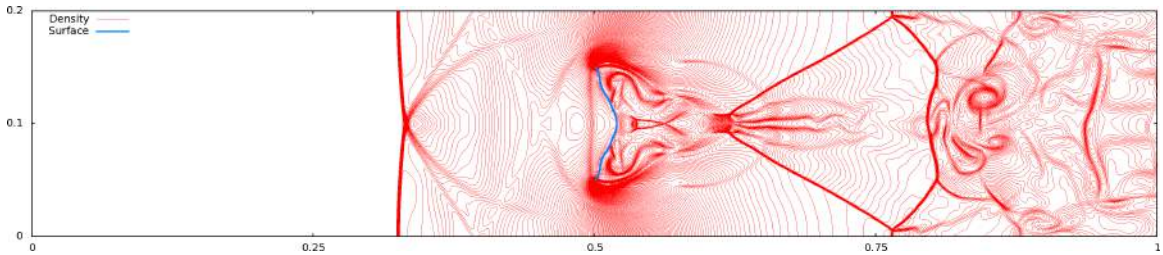
(a) $t = .07s$



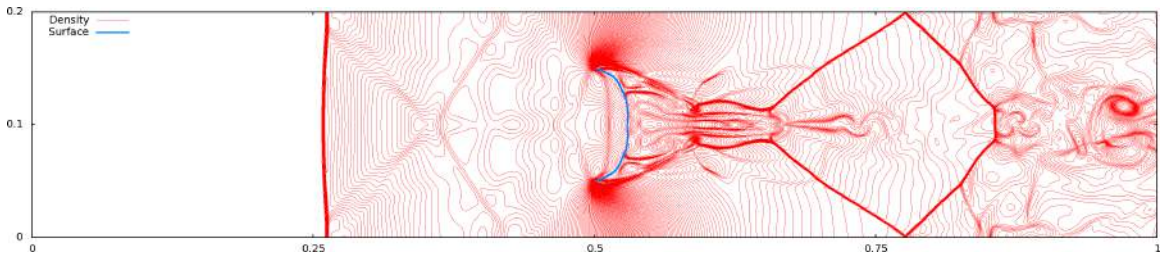
(b) $t = .14s$



(c) $t = .21s$

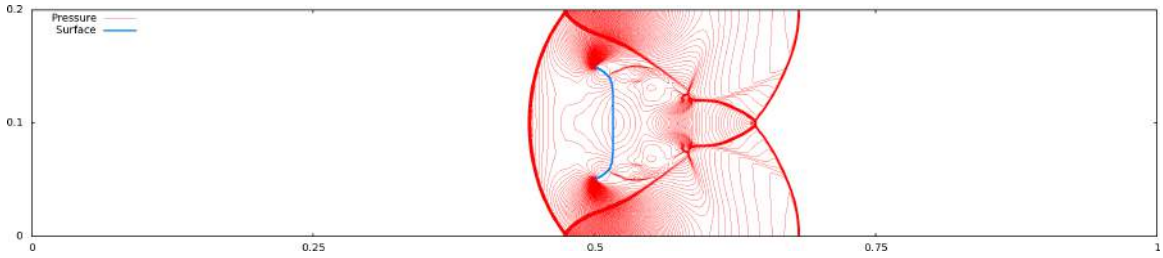


(d) $t = .28s$

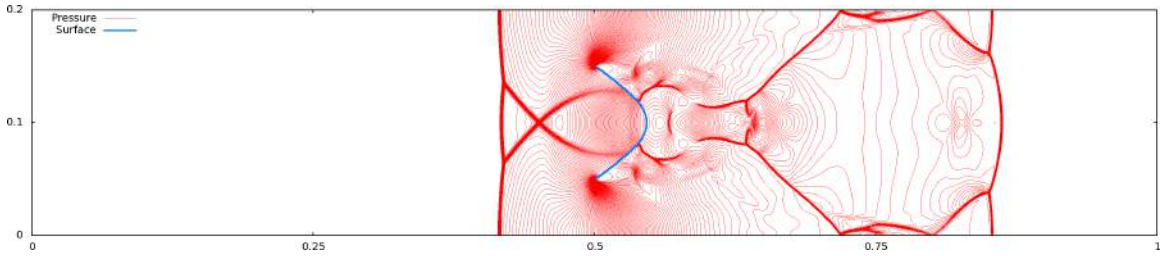


(e) $t = 35s$

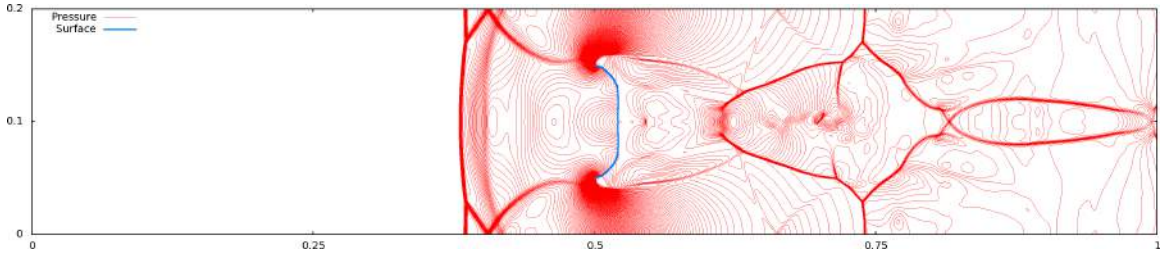
Figure 5.34: Time evolution of density in the example described in Section 5.5.3, on a 2560×512 grid.



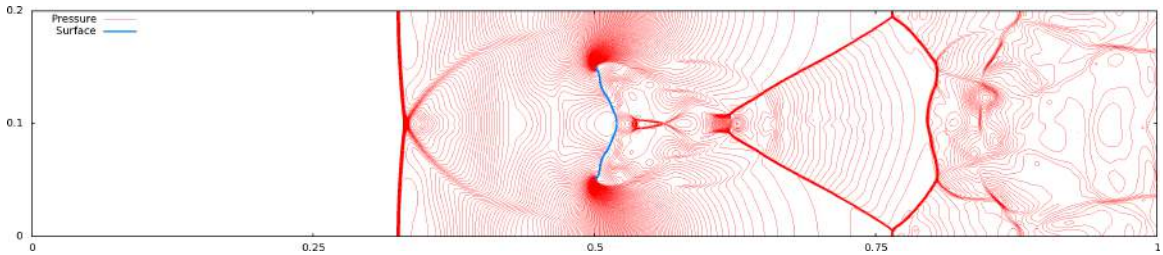
(a) $t = .07s$



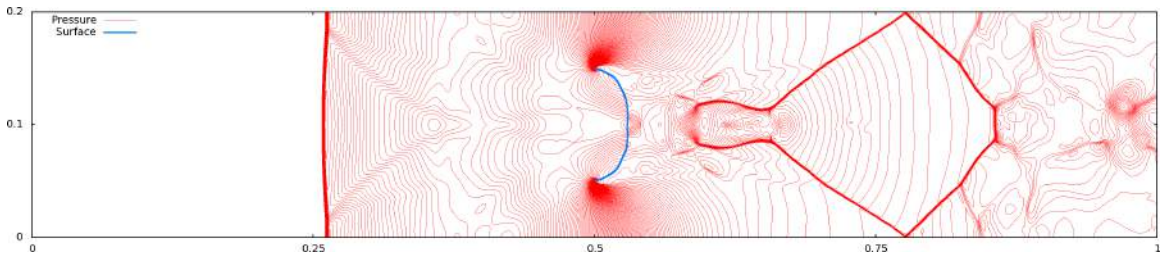
(b) $t = .14s$



(c) $t = .21s$

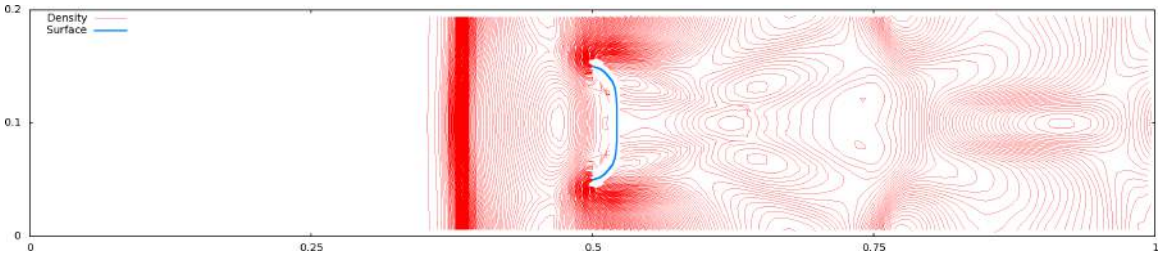


(d) $t = .28s$

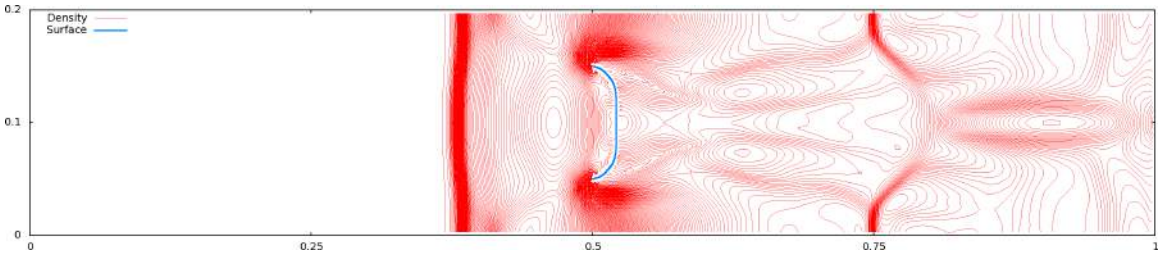


(e) $t = 35s$

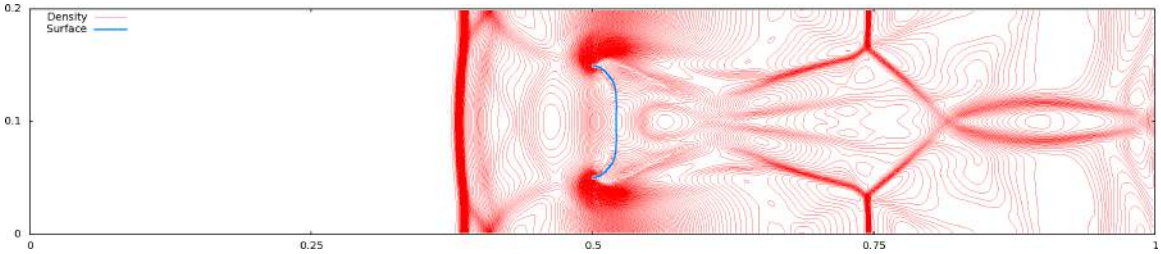
Figure 5.35: Time evolution of pressure in the example described in Section 5.5.3, on a 2560×512 grid.



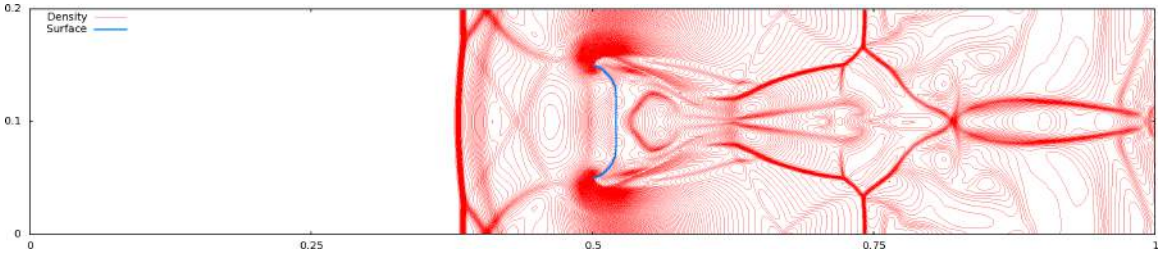
(a) Resolution 160×32



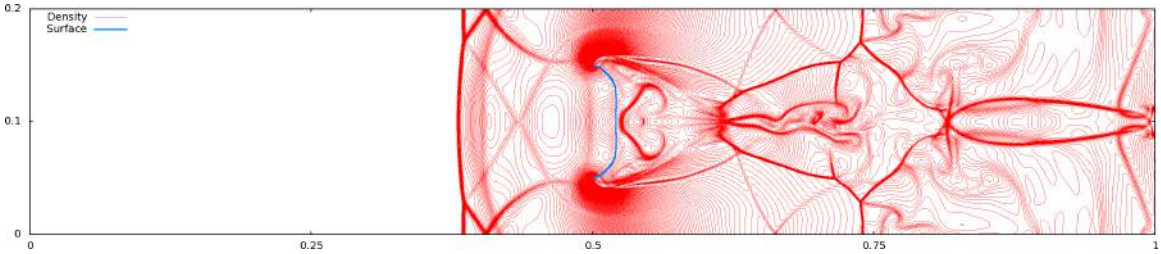
(b) Resolution 320×64



(c) Resolution 640×128



(d) Resolution 1280×256



(e) Resolution 2560×512

Figure 5.36: Grid convergence of the example described in Section 5.5.3, at time $t = .21s$.

5.6 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 [90]. 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.

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. Benson. A new two-dimensional flux-limited shock viscosity for impact calculations. *Comput. Meth. Appl. Mech. Eng.*, 93(1):39–95, 1991.

[6] D. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Comput. Meth. Appl. Mech. Eng.*, 99:235–394, 1992.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] P. Causin, J.-F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Comp. Meth. Appl. Mech. Eng.*, 194(42-44), 2005.

[12] R. Codina, G. Houzeaux, H. Coppola-Owen, and J. Baiges. The fixed-mesh ALE approach for the numerical approximation of flows in moving domains. *J. Comput. Phys.*, 228(5):1591–1611, 2009.

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

[14] R. Courant, E. Issacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Comm. Pure and Applied Math*, 5:243–255, 1952.

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

[16] T. Dupont and Y. Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys.*, 190/1:311–324, 2003.

[17] T. Dupont and Y. Liu. Back and forth error compensation and correction methods for semi-Lagrangian schemes with application to level set interface computations. *Math. Comp.*, 76(258):647–668, 2007.

[18] D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures*, 83:479–490, 2005.

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

[20] 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.

[21] 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.

[22] R. Fedkiw. Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175:200–224, 2002.

[23] 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.

[24] 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.

[25] 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.

[26] R. Fedkiw, B. Merriman, and S. Osher. Efficient characteristic projection in upwind difference schemes for hyperbolic systems (the complementary projection method). *J. Comput. Phys.*, 141:22–36, 1998.

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

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

[29] N. Grenier, M. Antuono, A. Colagrossi, D. Le Touzé, and B. Alessandrini. An Hamiltonian interface SPH formulation for multi-fluid and free surface flows. *J. Comput. Phys.*, 228(22):8380–8393, 2009.

[30] J.T. Grétarsson and R. Fedkiw. Fully conservative leak-proof treatment of thin solid structures immersed in compressible fluids. *Submitted to J. Comput. Phys.*, 2012.

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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.

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

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

[39] G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 25(3):805–811, 2006.

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

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

[42] S.Y. Kadioglu, M. Sussman, S. Osher, J. Wright, and M. Kang. A second order primitive preconditioner for solving all speed multi-phase flows. *J. Comput. Phys.*, 209:477–503, 2005.

[43] M. Kang, R. Fedkiw, and X.-D. Liu. A boundary condition capturing method for multiphase incompressible flow. *J. Sci. Comput.*, 15:323–360, 2000.

[44] B.-M. Kim, Y. Liu, I. Llamas, and J. Rossignac. Using BFECC for fluid simulation. In *Eurographics Workshop on Natural Phenomena 2005*, 2005.

[45] B.-M. Kim, Y. Liu, I. Llamas, and J. Rossignac. Advections with significantly reduced dissipation and diffusion. *IEEE Trans. on Vis. and Comput. Graph.*, 13(1):135–144, 2007.

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

[47] S. Koshizuka, A. Nobe, and Y. Oka. Numerical analysis of breaking waves using the moving particle semi-implicit method. *Int. J. Num. Meth. in Fluids*, 26:751–769, 1998.

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

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

[50] 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.

[51] 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.

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

[53] BP Leonard, AP Lock, and MK MacVean. Conservative explicit unrestricted-time-step multidimensional constancy-preserving advection schemes. *Mon. Weather Rev.*, 124:2588–2606, 1996.

[54] L.M. Leslie and R.J. Purser. Three-dimensional mass-conserving semi-lagrangian scheme employing forward trajectories. *Mon. Weather Rev.*, 123(8), 1995.

[55] R.J. Leveque. A large time step generalization of godunov's method for systems of conservation laws. *SIAM J. Num. Analysis*, 22(6):1051–1073, 1985.

[56] 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.

[57] S.J. Lin and R.B. Rood. Multidimensional flux-form semi-lagrangian transport schemes. *Mon. Weather Rev.*, 24(9):2046–2070, 1996.

[58] P. Liovic, M. Rudman, J.L. Liow, D. Lakehal, and D. Kothe. A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction. *Computers & Fluids*, 35(10):1011–1032, 2006.

[59] J. Liu, S. Koshizuka, and Y. Oka. A hybrid particle-mesh method for viscous, incompressible, multiphase flows. *J. Comput. Phys.*, 202(1):65–93, 2005.

[60] T.G. Liu, B.C. Khoo, and W.F. Xie. The modified ghost fluid method as applied to extreme fluid-structure interaction in the presence of cavitation. *Commun. Comput. Phys*, 1(5):898–919, 2006.

[61] 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.

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

[63] J. López, J. Hernández, P. Gómez, and F. Faura. A volume of fluid method based on multidimensional advection and spline interface reconstruction. *J. Comput. Phys.*, 195(2):718–742, 2004.

[64] 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.

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

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

[67] 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.

[68] Frank Losasso, Jerry Talton, Nipun Kwatra, and Ronald Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Trans. on Vis. and Comput. Graph.*, 14(4):797–804, 2008.

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

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

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

[72] D.G. Luenberger. The conjugate residual method for constrained minimization problems. *SIAM J. on Numer. Anal.*, pages 390–398, 1970.

[73] R. MacCormack. The effect of viscosity in hypervelocity impact cratering. In *AIAA Hypervelocity Impact Conference*, 1969. AIAA paper 69-354.

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

[75] 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.

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

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

[78] T. Nakamura, R. Tanaka, T. Yabe, and K. Takizawa. Exactly conservative semi-Lagrangian scheme for multi-dimensional hyperbolic equations with directional splitting technique. *J. Comput. Phys.*, 174(1):171–207, 2001.

[79] S. Osher and C.-W. Shu. High order essentially non-oscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Num. Anal.*, 28:902–921, 1991.

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

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

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

[83] J.E. Pilliod et al. Second-order accurate volume-of-fluid algorithms for tracking material interfaces* 1. *J. Comput. Phys.*, 199(2):465–502, 2004.

[84] 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.

[85] D.J. Price. Modelling discontinuities and Kelvin-Helmholtz instabilities in SPH. *J. Comput. Phys.*, 227(24):10040–10057, 2008.

[86] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *J. Comput. Phys.*, 141:112–152, 1998.

[87] P.J. Roache. A flux-based modified method of characteristics. *Int. J. Num. Meth. in Fluids*, 15(11):1259–1275, 1992.

[88] 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.

[89] 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.

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

[91] 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.

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

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

[94] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *J. Sci. Comput.*, 19(1):439–456, 2003.

[95] A. Staniforth and J. Cote. Semi-Lagrangian integration schemes for atmospheric models: A review. *Mon. Weather Rev.*, 119:2206–2223, 1991.

[96] J. Strain. Tree methods for moving interfaces. *J. Comput. Phys.*, 151:616–648, 1999.

[97] K. Taira and T. Colonius. The immersed boundary method: A projection approach. *J. Comput. Phys*, 225(2):2118–2137, 2007.

[98] K. Takizawa, T. Yabe, and T. Nakamura. Multi-dimensional semi-Lagrangian scheme that guarantees exact conservation. *Computer Physics Communications*, 148(2):137–159, 2002.

[99] R. Tanaka, T. Nakamura, and T. Yabe. Constructing exactly conservative scheme in a non-conservative form. *Computer Physics Communications*, 126(3):232–243, 2000.

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

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

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

[103] F. Xiao. Unified formulation for compressible and incompressible flows by using multi-integrated moments I: one-dimensional inviscid compressible flow. *J. Comput. Phys.*, 195:629–5654, 2004.

[104] F. Xiao, R. Akoh, and S. Ii. Unified formulation for compressible and incompressible flows by using multi-integrated moments II: Multi-dimensional version for compressible and incompressible flows. *J. Comput. Phys.*, 213:31–56, 2006.

[105] F. Xiao and T. Yabe. Completely conservative and oscillationless semi-Lagrangian schemes for advection transportation. *J. Comput. Phys.*, 170(2):498–522, 2001.

[106] T. Yabe, R. Tanaka, T. Nakamura, and F. Xiao. An exactly conservative semi-Lagrangian scheme (CIP–CSL) in one dimension. *Mon. Weather Rev.*, 129:332–344, 2001.

[107] T. Yabe and P.Y. Wang. Unified numerical procedure for compressible and incompressible fluid. *Journal of the Physical Society of Japan*, 60:2105–2108, July 1991.

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

[109] S.Y. Yoon and T. Yabe. The unified simulation for incompressible and compressible flow by the predictor-corrector scheme based on the CIP method. *Comput. Phys. Commun.*, 119:149–158, 1999.

[110] S.T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31(3):335–362, 1979.

[111] M. Zerroukat, N. Wood, and A. Staniforth. Application of the parabolic spline method (psm) to a multi-dimensional conservative semi-lagrangian transport scheme (slice). *J. Comput. Phys.*, 225(1):935–948, 2007.

[112] 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.

[113] Q. Zhang and P.L.F. Liu. A new interface tracking method: The polygonal area mapping method. *J. Comput. Phys.*, 227(8):4063–4088, 2008.

[114] 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.

Jón Tómas Grétarsson

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Ronald Fedkiw) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Charbel Farhat)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Eric Darve)

Approved for the University Committee on Graduate Studies
