An Unconditionally Stable Fully Conservative Semi-Lagrangian Method

Michael Lentine*, Jón Tómas Grétarsson*, Ronald Fedkiw*

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

Abstract

Semi-Lagrangian methods have been around for some time, dating back at least to [3]. 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.

1. Introduction

The idea of applying the method of characteristics to advect quantities forward in time dates back at least as far as [3] and has gained popularity in many areas, such as atmospheric sciences [46]. 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 [38]. The simplicity of these schemes makes them quite useful for adaptive grids and other data structures, see for example [7, 31, 30, 47, 16]. Recently, authors have considered using semi-Lagrangian methods as building blocks in other schemes, for example [17, 18, 6] showed that the second order accurate BFECC method of [5] can be made unconditionally stable using the first order accurate semi-Lagrangian method as a building block. In addition, [44] showed that the original scheme of MacCormack [36] 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 [23, 26, 24, 55], as well as the compressible flow community [25, 43] 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 [49, 52, 39, 51, 48].

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

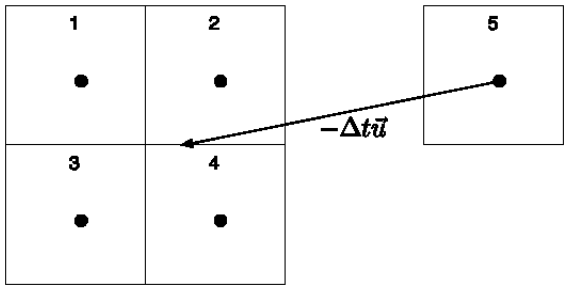
*{mlentine,jontg,fedkiw}@cs.stanford.edu, Stanford University

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 [9, 35, 20, 19, 53, 4, 2, 28, 32, 41, 10]. 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 [40, 42, 27, 14, 29, 56]). 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 [15, 37, 33, 34, 1]. 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.

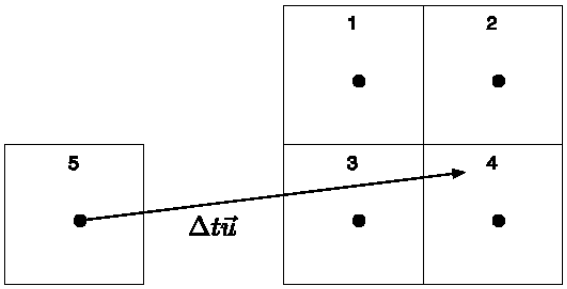
Notably, our method is quite simple, both conceptually and as far as implementation is concerned—it requires only a small modification to a standard semi-Lagrangian scheme and utilizes most of the functionality already present. The standard semi-Lagrangian method updates the value at a grid point by 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.



(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 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.

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 [21] 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.

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 $\vec{u}$,

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

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

$$(\rho\phi)_t + \nabla \cdot (\phi\rho\vec{u}) = 0. \tag{2}$$

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 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

$$\hat{\phi}(\vec{x}_j, t^{n+1}) = \hat{\phi}(\vec{x}, t^n) = \sum_i w_{ij} \hat{\phi}(\vec{x}_i, t^n), \tag{3}$$

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 (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 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

$$\hat{\phi}_j^{n+1} = \sum_i \hat{w}_{ij} \hat{\phi}(x_i, t^n). \tag{4}$$

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.

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:

$$\text{Error}(t^n) = \Sigma\hat{\phi}(t^n) - \left[\Sigma\hat{\phi}(t^0) + \Sigma_{in} - \Sigma_{out}\right], \tag{5}$$

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. [12]). Typically, when performing *interpolation* as in [12] 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.

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 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. Figures 3, 4 and 5 demonstrate the significant error improvement by using this interpolation scheme. Note that in these 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 [44] and the modified BFECC scheme of [6]. Another interesting approach would be to apply a second order non-conservative correction to a full conservative first order accurate scheme.

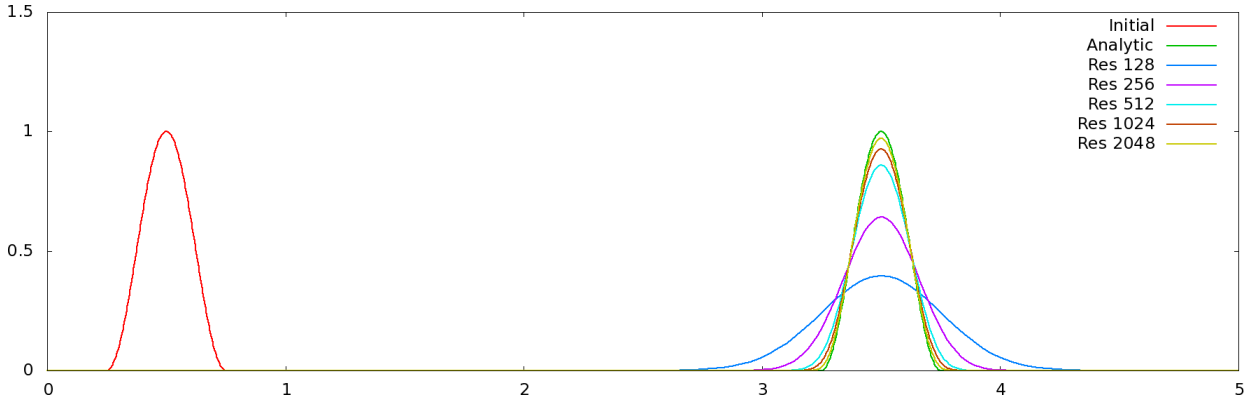
2.3. Examples

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

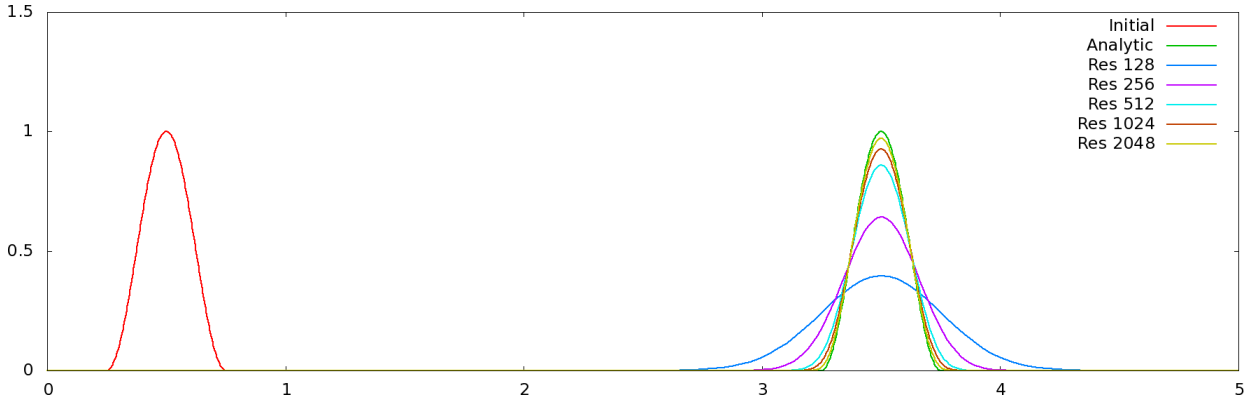
$$\hat{\phi}(x,0) = \begin{cases} \frac{1}{2}\left(1 + \sin\left(4\pi * (x - \frac{3}{8})\right)\right) & .25 \le x \le .75 \\ 0 & \text{else} \end{cases} \tag{6}$$

with $u = 1$. The problem is discretized over the domain $[0, 5]$, and we solve Equation (2) with a CFL number of .9. In Figure 2(a), we show the solution as computed by a standard non-conservative semi-Lagrangian advection, while Figure 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 3 shows a comparison between using linear and quadratic interpolation in our method. Figure 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 5 we run this simulation three times longer with a CFL of 2.9 showing errors more commensurate with Figure 3 as expected. We also demonstrate the order of convergence in Table 1 which shows that our algorithm gives first order convergence.

Figure 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 8 shows dramatic loss of conservation in the standard semi-Lagrangian method as compared to the conservative version which maintains exact value up

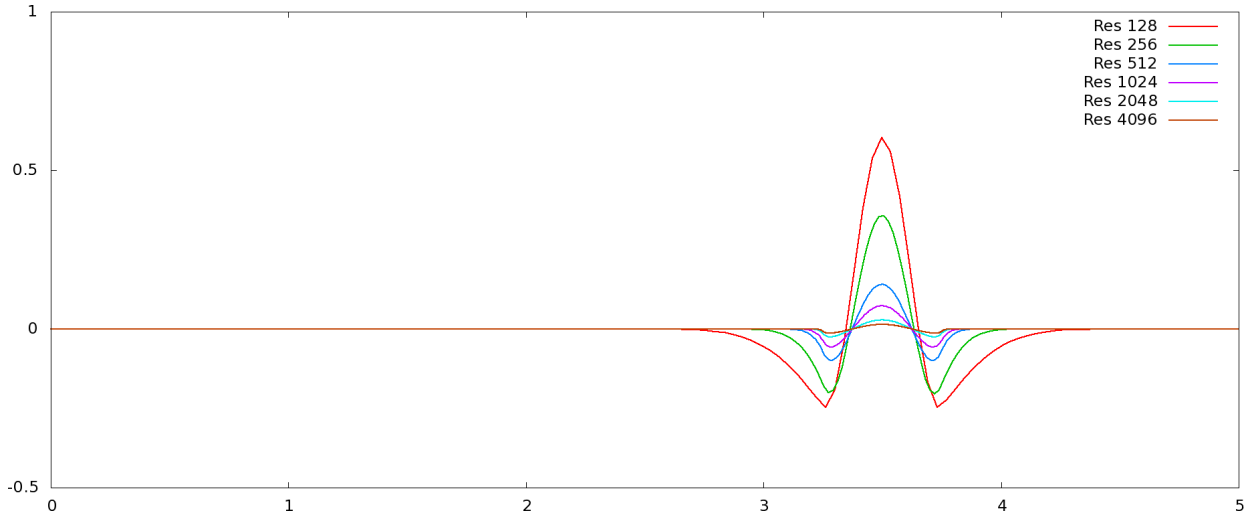


(a) Standard semi-Lagrangian advection.

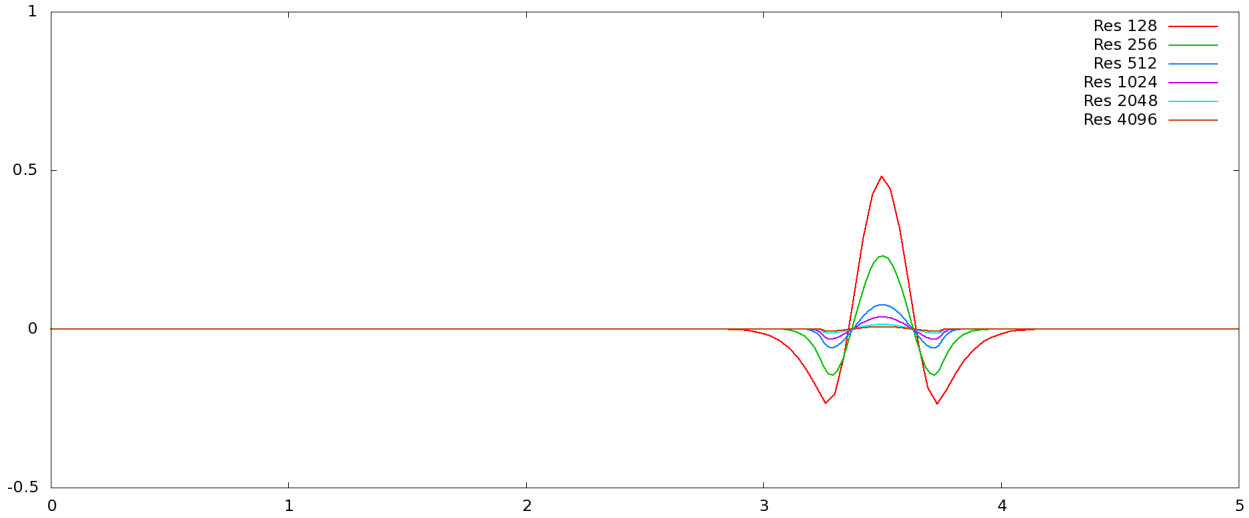


(b) Conservative semi-Lagrangian advection.

Figure 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.

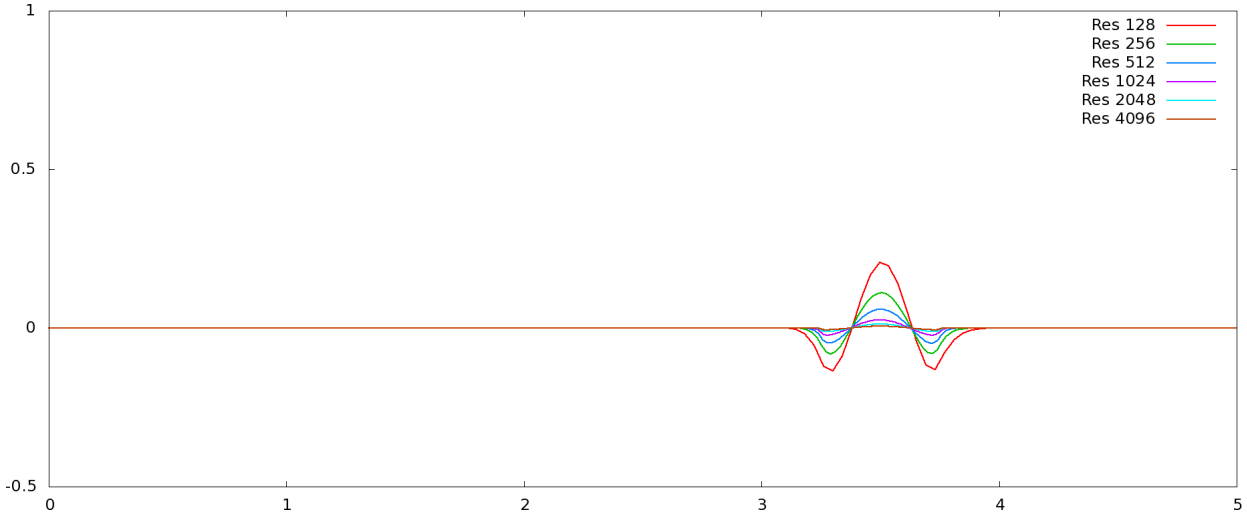


(a) Linear interpolation.

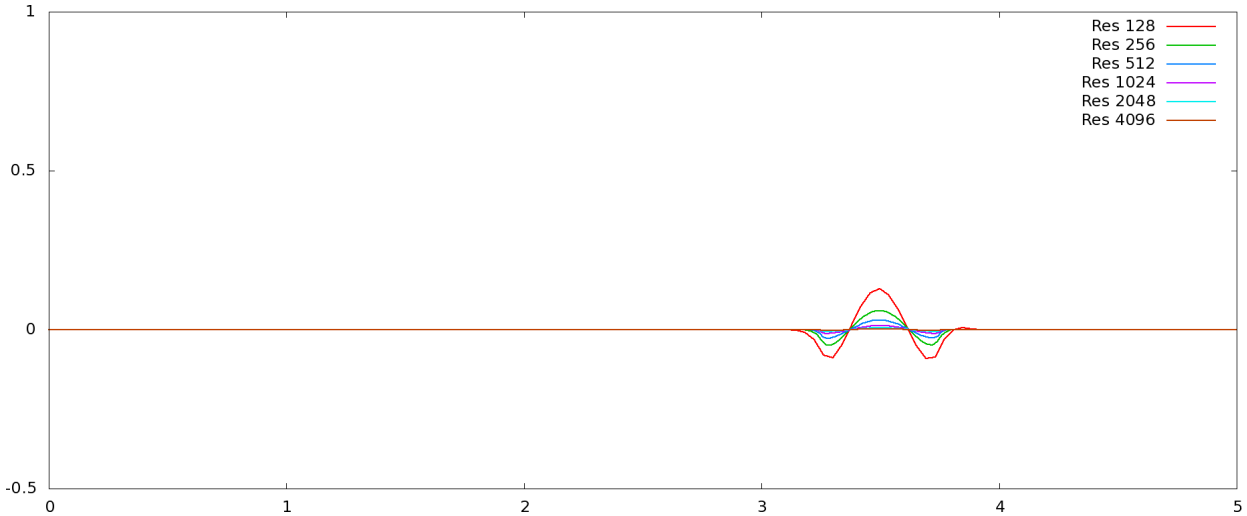


(b) Quadratic interpolation.

Figure 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.

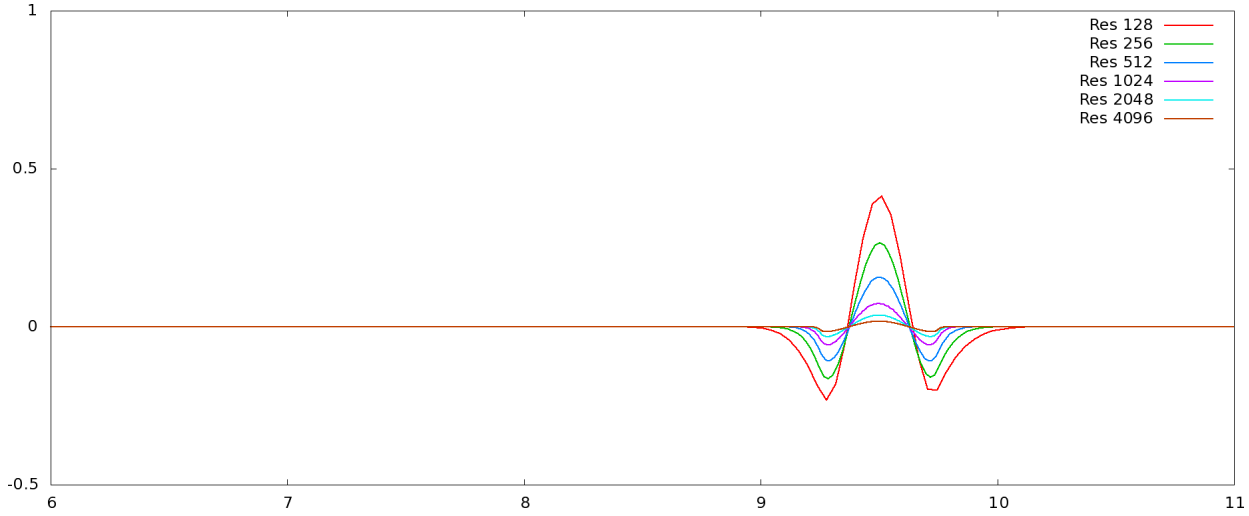


(a) Linear interpolation.

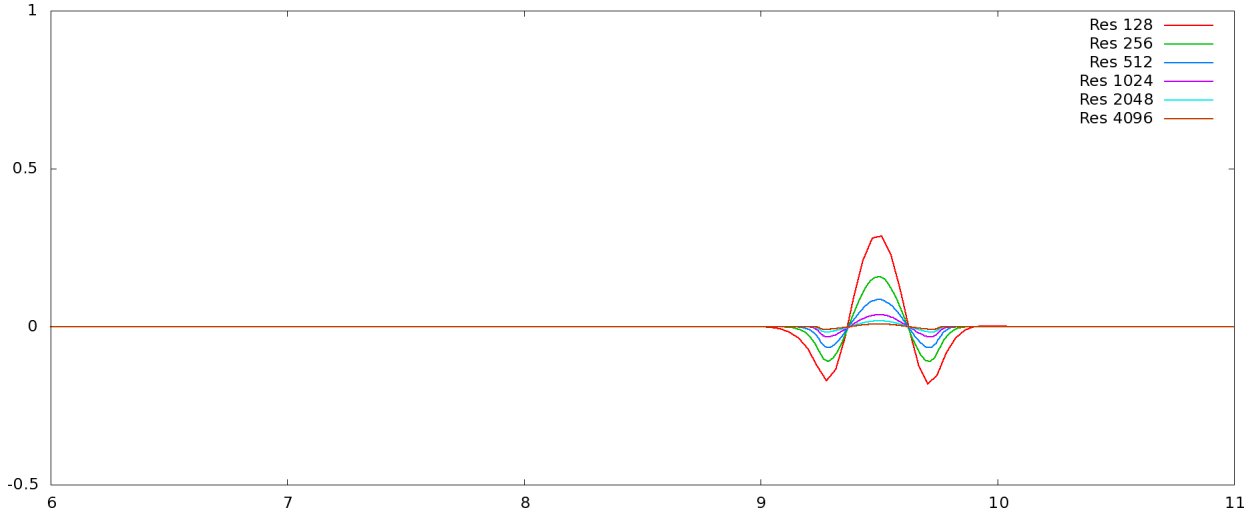


(b) Quadratic interpolation.

Figure 4: 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 3, and so the error introduced by the interpolation scheme are significantly smaller.



(a) Linear interpolation.



(b) Quadratic interpolation.

Figure 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 3).

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 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.

to round off error. Figure 7 shows a convergence analysis for the two schemes using a high resolution full conservative ENO method [45] as a ground truth. As is typical the non-conservative method converges to the wrong solution whereas our new 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 (1) whereas our method solves Equation (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 [54]. In this example a notched disk is advected through a velocity field specified by

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

Shown in Figure 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 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.

3. Incompressible flow

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

$$\begin{cases} \vec{u}_t + \vec{u} \cdot \nabla \vec{u} + \frac{\nabla p}{\rho} = \frac{1}{\rho} \nabla \cdot (\mu \nabla \vec{u}) \\ \nabla \cdot \vec{u} = 0 \end{cases} \tag{7}$$

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

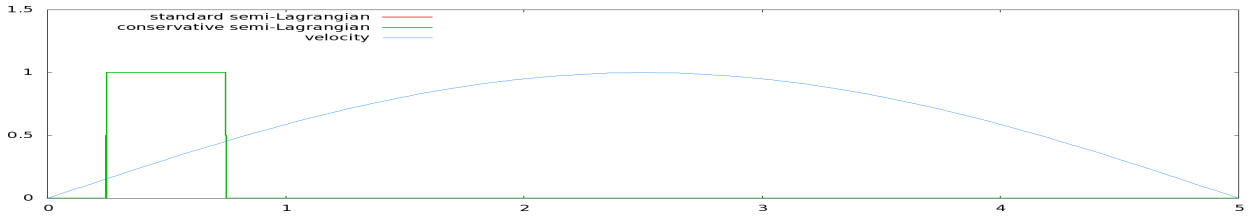
$$\Delta t \nabla \cdot \frac{1}{\rho} \nabla p = \nabla \cdot \vec{u}^\star \tag{8}$$

to obtain a pressure, which is then applied via:

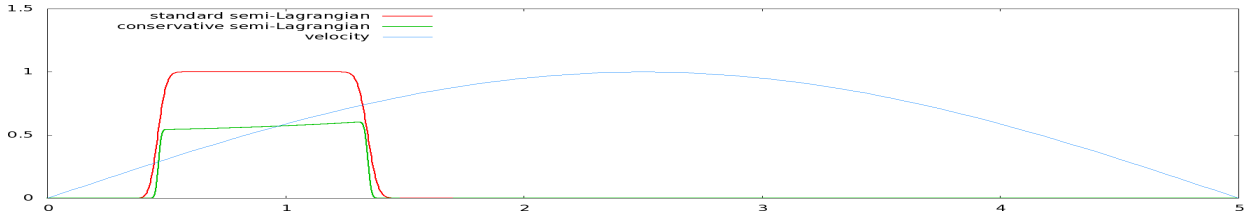
$$\vec{u}^{\star\star} = \vec{u}^\star - \frac{\Delta t}{\rho} \nabla p. \tag{9}$$

Viscous forces are next implicitly accounted for by solving

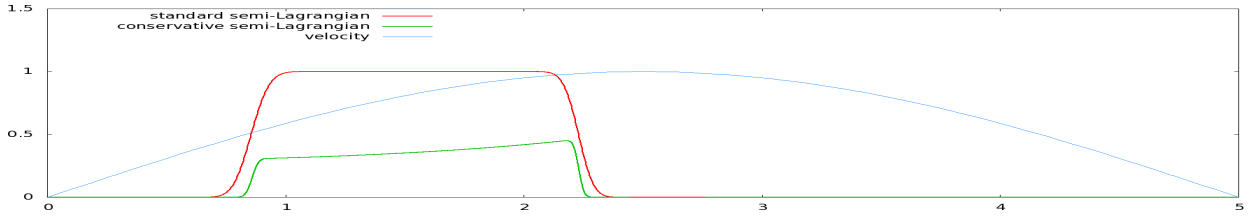
$$\tilde{\vec{u}}^{n+1} = \vec{u}^{\star\star} + \frac{\Delta t}{\rho} \nabla \cdot (\mu \nabla \tilde{\vec{u}}^{n+1}), \tag{10}$$



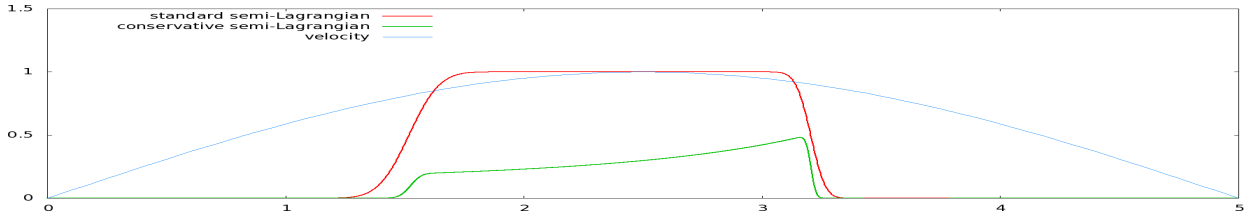
(a) $t = 0s$.



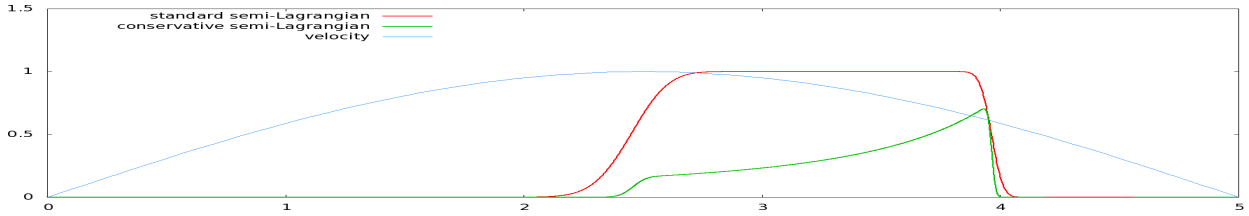
(b) $t = 1s$.



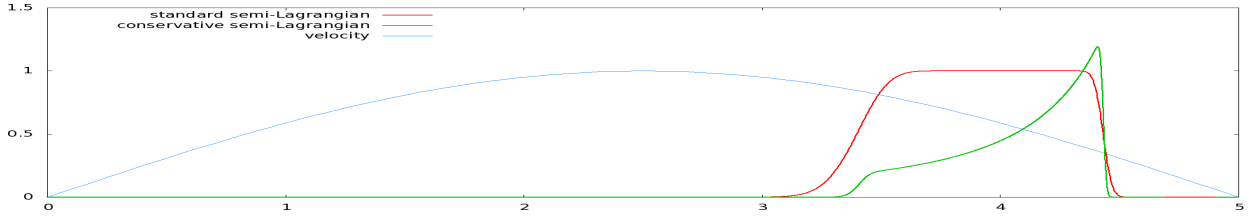
(c) $t = 2s$.



(d) $t = 3s$.

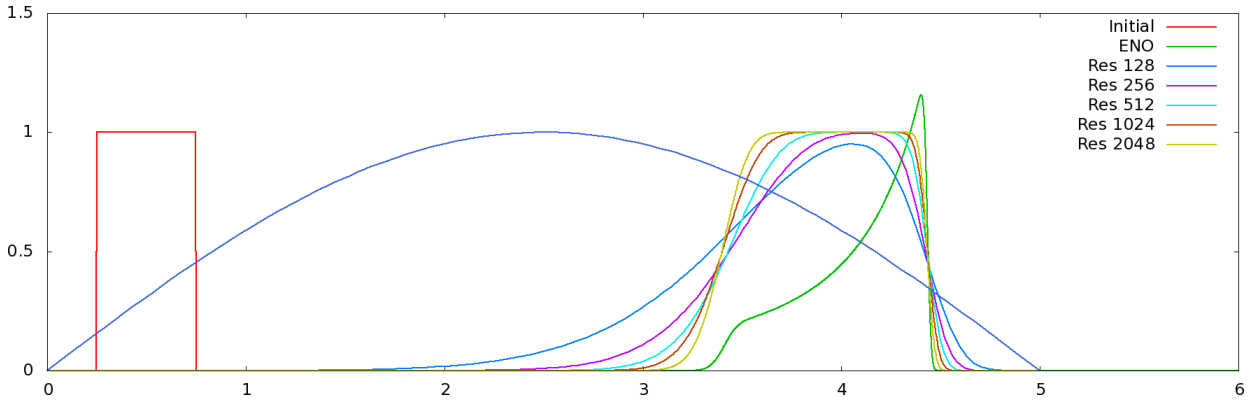


(e) $t = 4s$.

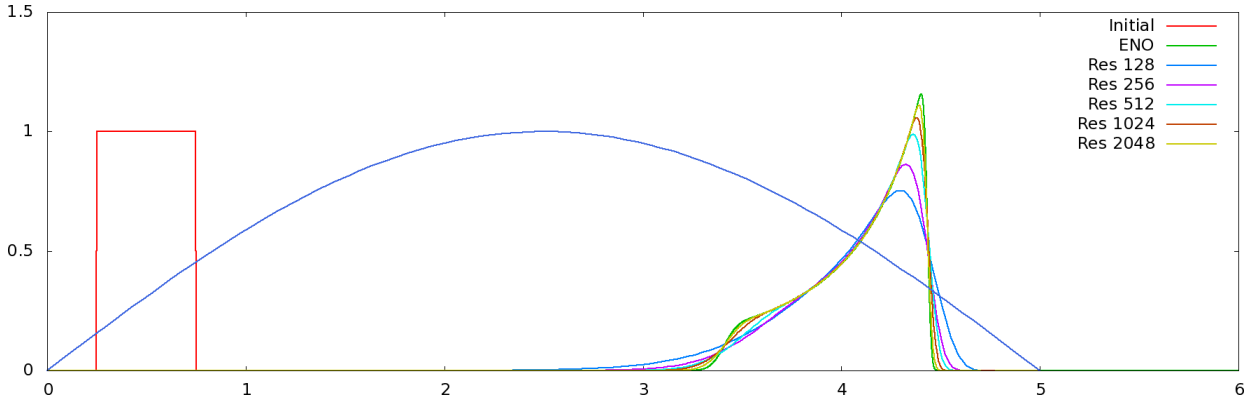


(f) $t = 5s$.

Figure 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 $\Delta x = 5/8192$.



(a) Standard semi-Lagrangian advection.



(b) Conservative semi-Lagrangian advection.

Figure 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.

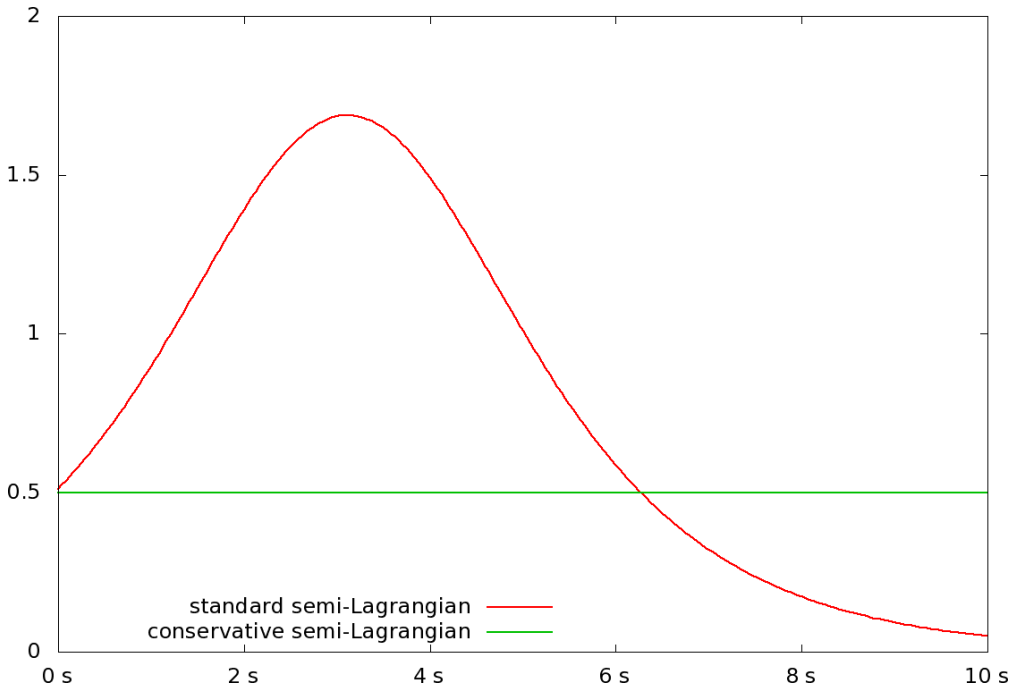


Figure 8: Shown is the time history of $\sum_i \Delta x \hat{\phi}_i$ for a square wave that is evolved through a divergent velocity field with $u = \sin\left(\pi \frac{x}{5}\right)$. Solutions for both the standard semi-Lagrangian advection scheme and our proposed conservative semi-Lagrangian advection scheme are shown at high-resolution with $\Delta x = 5/8192$.

after which we project the flow field again by solving Equation (8) (replacing $\vec{u}^\star$ with $\tilde{\vec{u}}$), and then finally updating the flow field to time t^{n+1} via

$$\vec{u}^{n+1} = \tilde{\vec{u}}^{n+1} - \frac{\Delta t}{\rho} \nabla p. \tag{11}$$

A standard Marker and Cell (MAC, [13]) 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.

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 (7) through by density, giving the following equations in two spatial dimensions:

$$\rho u_t + \rho u u_x + \rho v u_y + p_x = (\mu u_x)_x + (\mu u_y)_y, \tag{12}$$
$$\rho v_t + \rho v u_x + \rho v v_y + p_y = (\mu v_x)_x + (\mu v_y)_y. \tag{13}$$

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

$$(L_u)_t + (L_u u)_x + (L_u v)_y + p_x = (\mu u_x)_x + (\mu u_y)_y, \tag{14}$$
$$(L_v)_t + (L_v u)_x + (L_v v)_y + p_y = (\mu v_x)_x + (\mu v_y)_y. \tag{15}$$

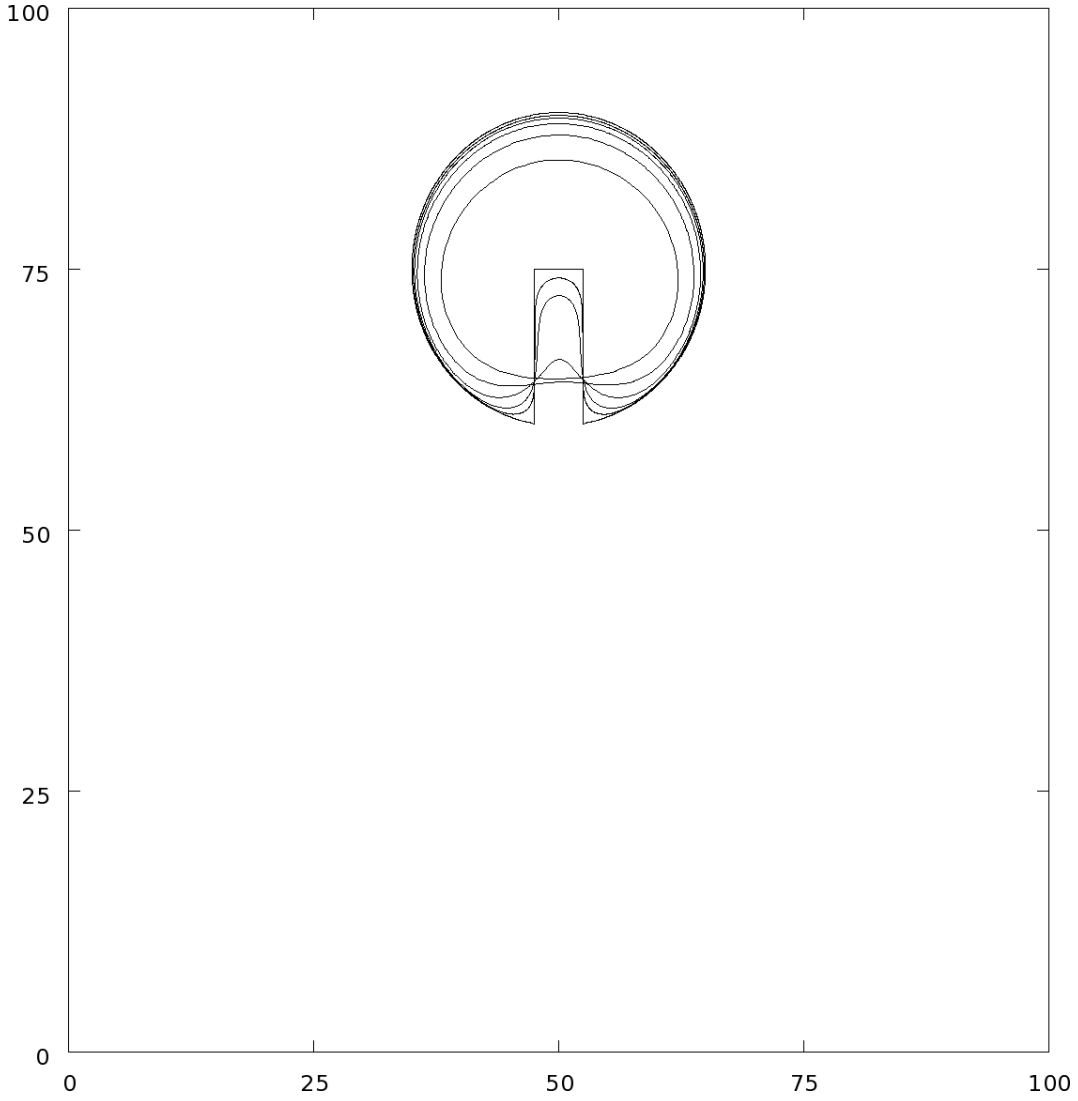


Figure 9: After one full rotation of the Zalesak disk [54] using our proposed conservative semi-Lagrangian advection scheme, for a variety of grid resolutions. Shown is the .5 isocontour for grid resolutions $\Delta x = 2^{-7}$, 2^{-8}, 2^{-9}, 2^{-10}, and 2^{-11}, in addition to the analytic solution. The mass of the disk is properly conserved using our method (this is verified in Figure 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.

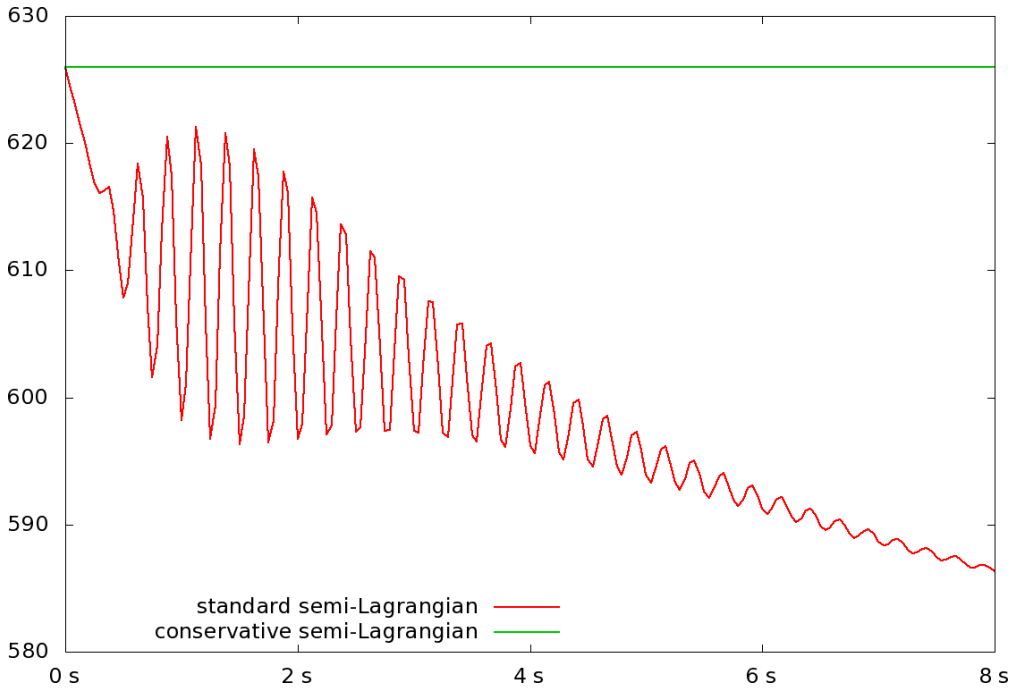


Figure 10: Shown is the time history of $\Sigma_i \Delta x \hat{\phi}_i + \Sigma_{out} - \Sigma_{in}$ for Zalesak Disk with $\Delta x = 2^{-7}$. Time history for the standard semi-Lagrangian advection scheme is shown in red, while our proposed conservative semi-Lagrangian advection scheme is shown in green.

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 (9) and (12), (13), (14), (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 (9) to the velocity field $\vec{u}^\star$.

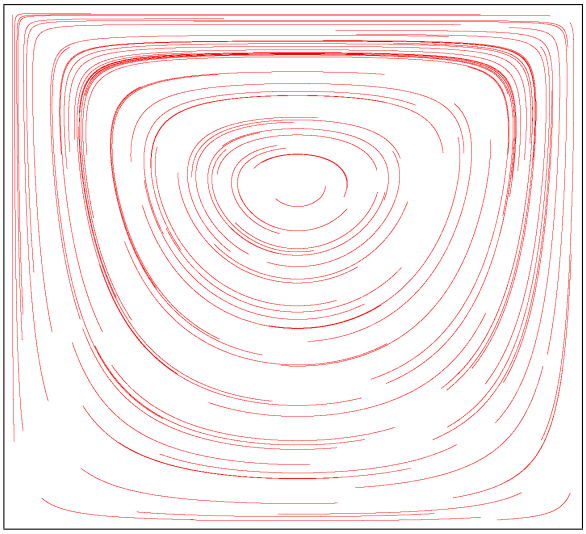
The viscous terms are treated implicitly by solving $\frac{\rho \tilde{u}^{n+1} - \rho \tilde{u}^{\star\star}}{\Delta t} = (\mu \tilde{u}_x^{n+1})_x + (\mu \tilde{u}_y^{n+1})_y$ which for constant density and viscosity becomes

$$\tilde{L}_u^{n+1} = L_u^{\star\star} + \Delta t \mu \nabla^2 \tilde{u}^{n+1}, \tag{16}$$

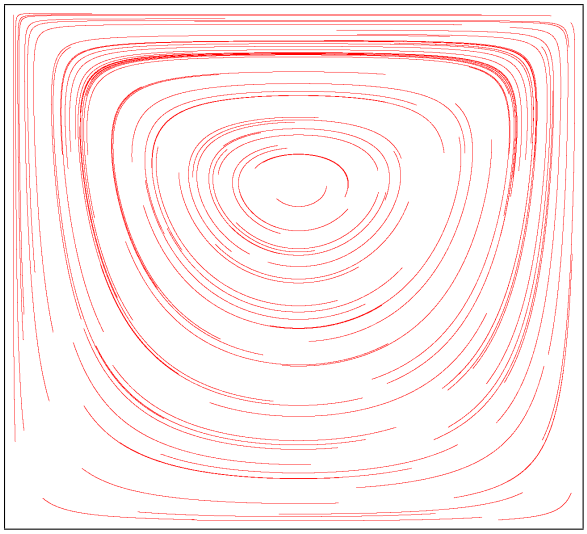
similar to Equation (10) above. In order to properly account for momentum transfer during the viscous stage, we are careful to view this viscosity update in a flux-based form. That is, μu_x acts as a momentum flux in between the MAC grid stencil locations of u values, and μu_y acts as a momentum flux 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

$$\tilde{L}_v^{n+1} = L_v^{\star\star} + \Delta t \mu \nabla^2 \tilde{v}^{n+1}. \tag{17}$$

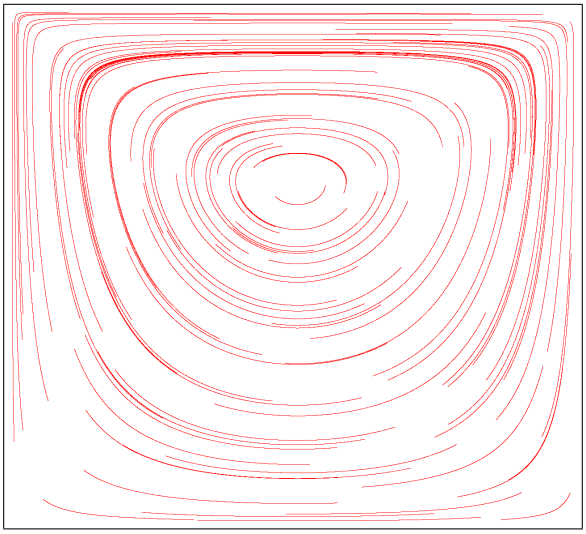
These intermediate quantities are again projected by solving Equation (8) (replacing $\vec{u}^\star$ with $\tilde{\vec{u}}^{n+1}$). The time t^{n+1} velocity field is computed using Equation (11), and momentum is updated as $L_u^{n+1} = \rho u^{n+1}$ and $L_v^{n+1} = \rho v^{n+1}$.



(a) Standard semi-Lagrangian scheme.



(b) Momentum-conserving scheme.



(c) Kinetic energy-conserving scheme.

Figure 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 $\Delta x = 2^{-7}$.

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 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 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 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 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 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 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. 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 (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

$$K_t+\nabla\cdot[(K+p)\vec{u}]=\vec{u}\cdot(\nabla\cdot\tau). \tag{18}$$

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 $v^\star$.

4.2. Projection

The modified $\vec{u}^\star$ values are used in Equation (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 (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

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

where $\hat{u}=\frac{u^{\star\star}+u^\star}{2}$ and $\hat{v}=\frac{v^{\star\star}+v^\star}{2}$, and we use Equation (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 [22], 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

$$\hat{u}^{new} = \hat{u}^{old} + \frac{\hat{p}_i}{m}, \tag{19}$$

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.

4.3. Viscosity

After the projection in Equation (9) we compute the viscous forces via Equation (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 (10) to eliminate the $\left(\tilde{u}^{n+1} - u^{\star\star}\right)$ 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. Examples

We first reconsider the driven cavity case from Section 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 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 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 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 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 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 16 shows the kinetic energy advected into and out of the domain across inflow and outflow boundaries. Figure 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 18 for each of the three schemes. Figure 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.

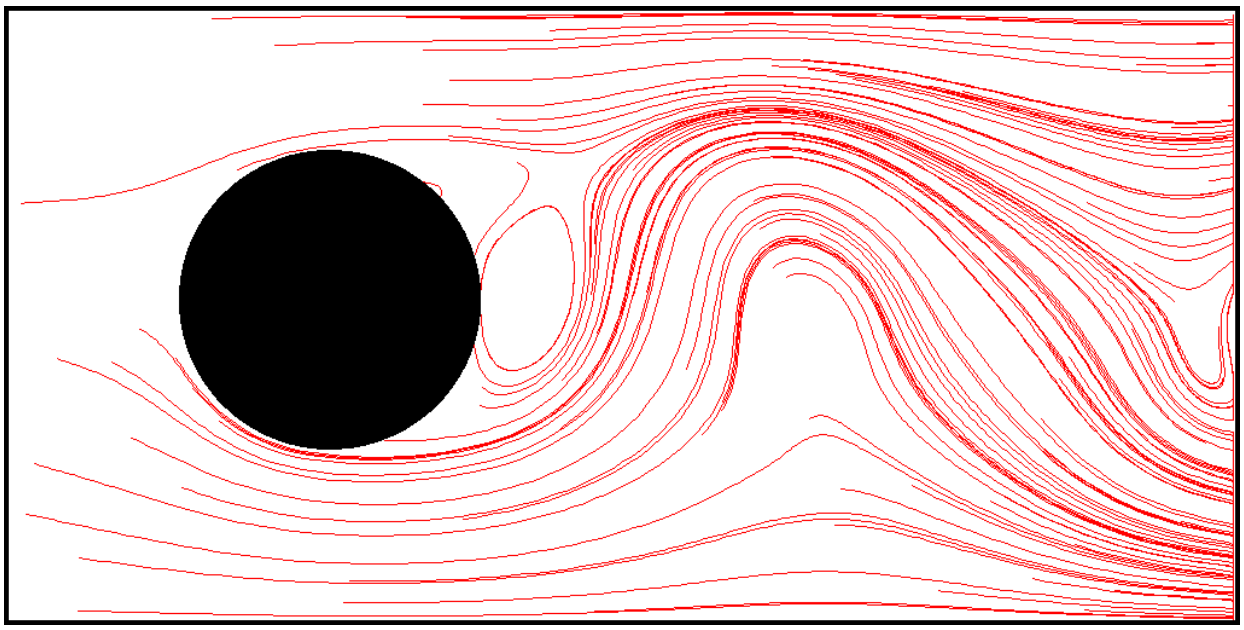


Figure 12: Stream-line visualization of flow past a sphere.

5. Compressible flow

We model compressible flow using the inviscid Euler equations,

$$\begin{pmatrix} \rho \\ \rho\vec{u} \\ E \end{pmatrix}_t + \begin{pmatrix} \nabla\cdot[\rho\vec{u}] \\ \nabla\cdot[\rho\vec{u}\otimes\vec{u}] + \nabla p \\ \nabla\cdot[(E+p)\vec{u}] \end{pmatrix} = 0, \tag{20}$$

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 [21]. 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 $F_2(\vec{U})$:

$$F_1(\vec{U}) = \begin{pmatrix} \nabla\cdot[\rho\vec{u}] \\ \nabla\cdot[\rho\vec{u}\otimes\vec{u}] \\ \nabla\cdot[E\vec{u}] \end{pmatrix}, \qquad F_2(\vec{U}) = \begin{pmatrix} 0 \\ \nabla p \\ \nabla\cdot[p\vec{u}] \end{pmatrix}. \tag{21}$$

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:

$$\nabla\cdot\vec{u}^{n+1} - \nabla\cdot\vec{u}^\star = -\Delta t\nabla\cdot\frac{1}{\rho^{n+1}}\nabla p. \tag{22}$$

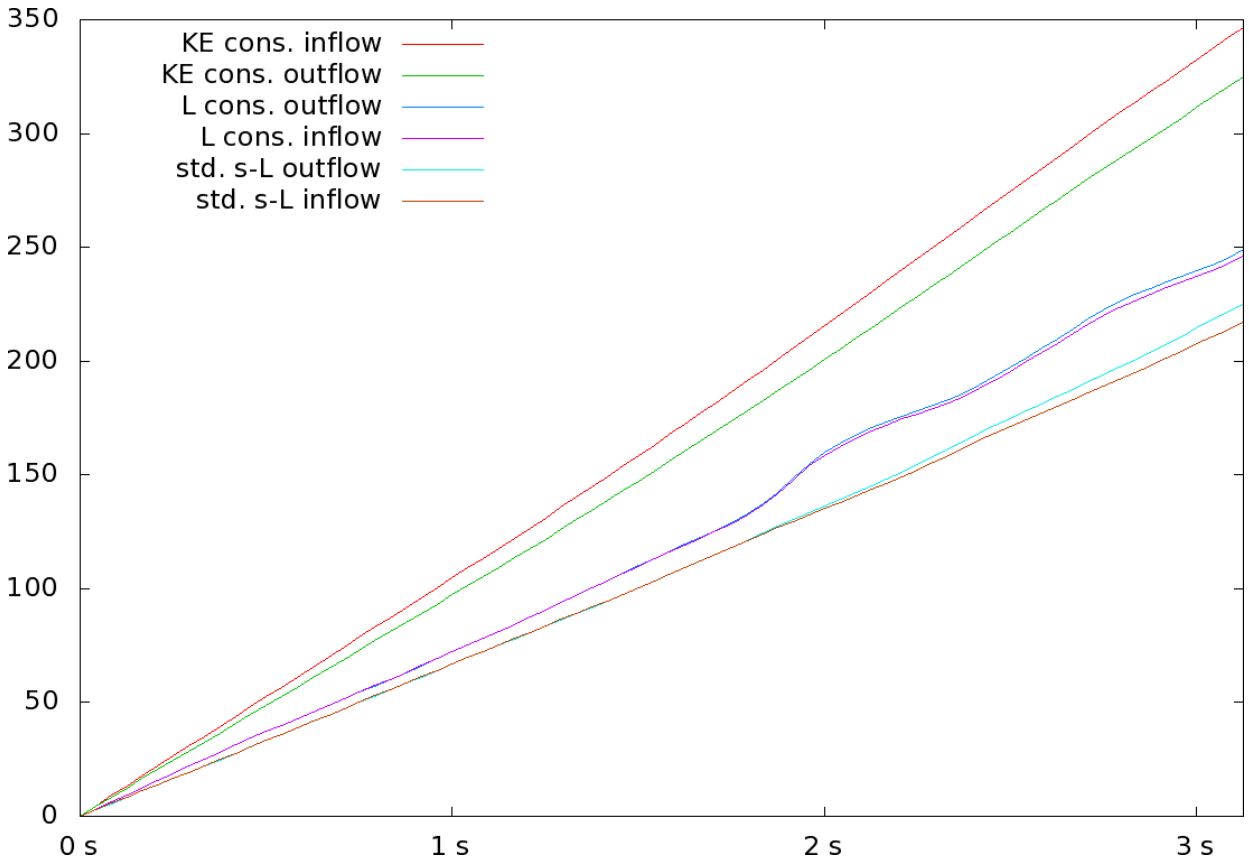


Figure 13: Total momentum fluxing into the computational domain and total momentum fluxing out of the computational domain, plotted as a function of time for a standard semi-Lagrangian scheme and our proposed momentum-conserving scheme.

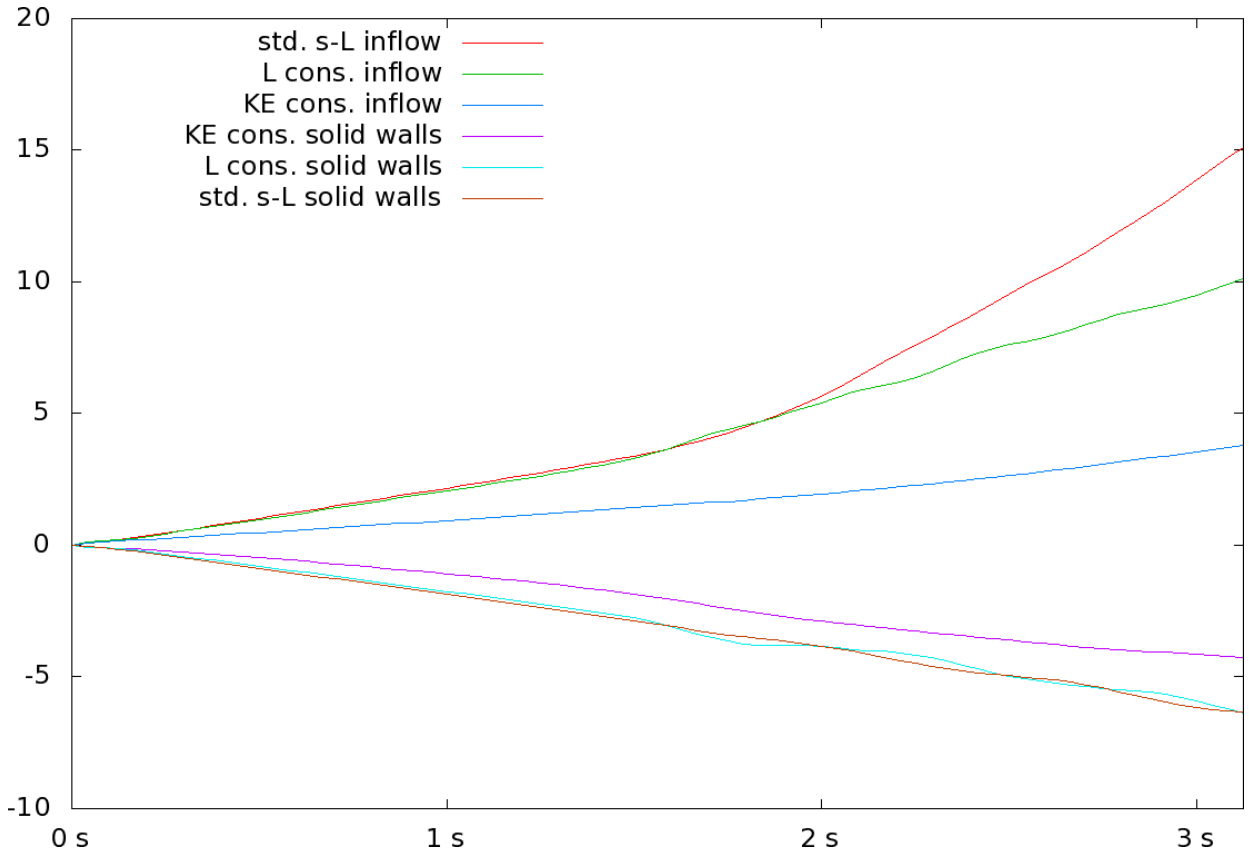


Figure 14: Pressure momentum flux into solid wall boundaries, and pressure momentum flux entering the computational domain from the inflow boundary condition, plotted as a function of time for a standard semi-Lagrangian scheme and our proposed momentum-conserving scheme.

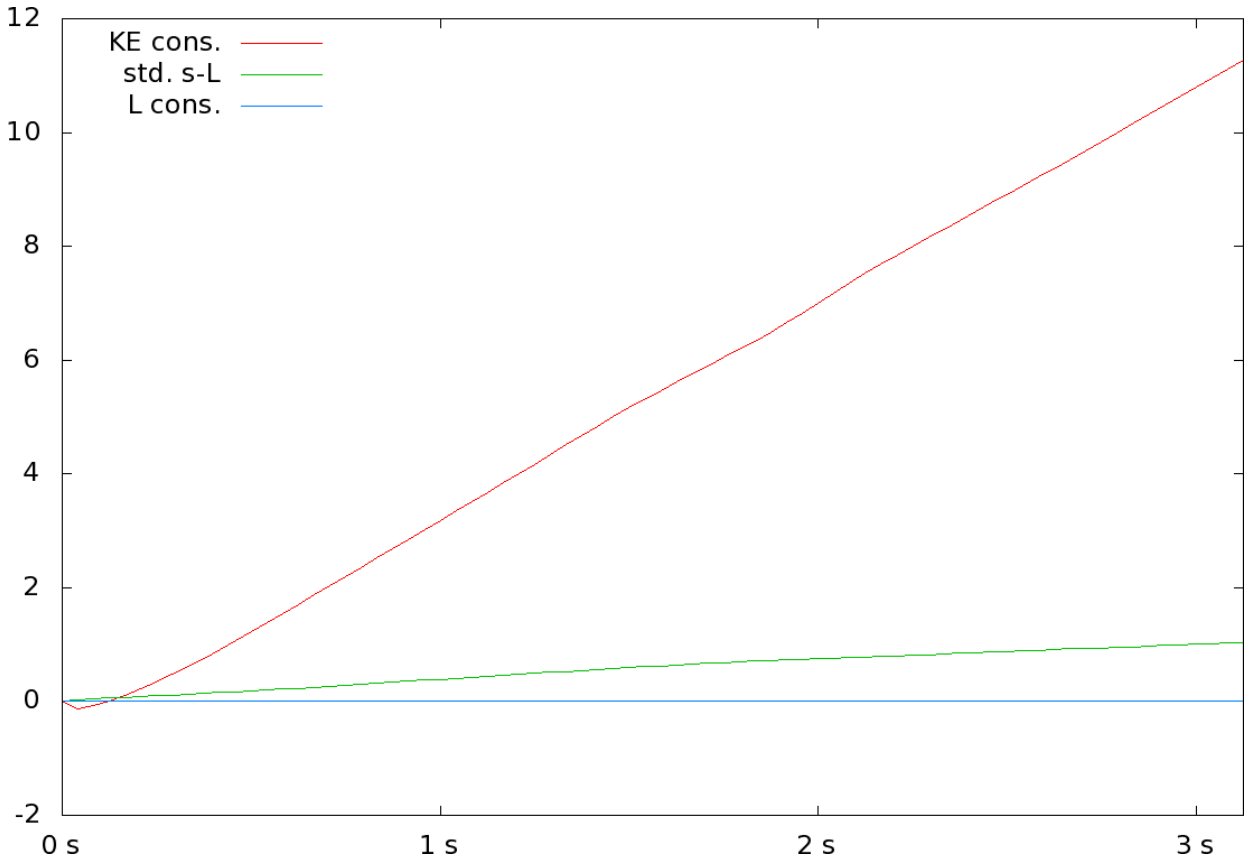


Figure 15: Sum total of momentum in the domain, plus momentum fluxed out of the domain (through outflow and solid wall boundaries), minus momentum fluxed into the domain (through inflow), plotted as a function of time for a standard semi-Lagrangian scheme and our proposed momentum-conserving scheme.

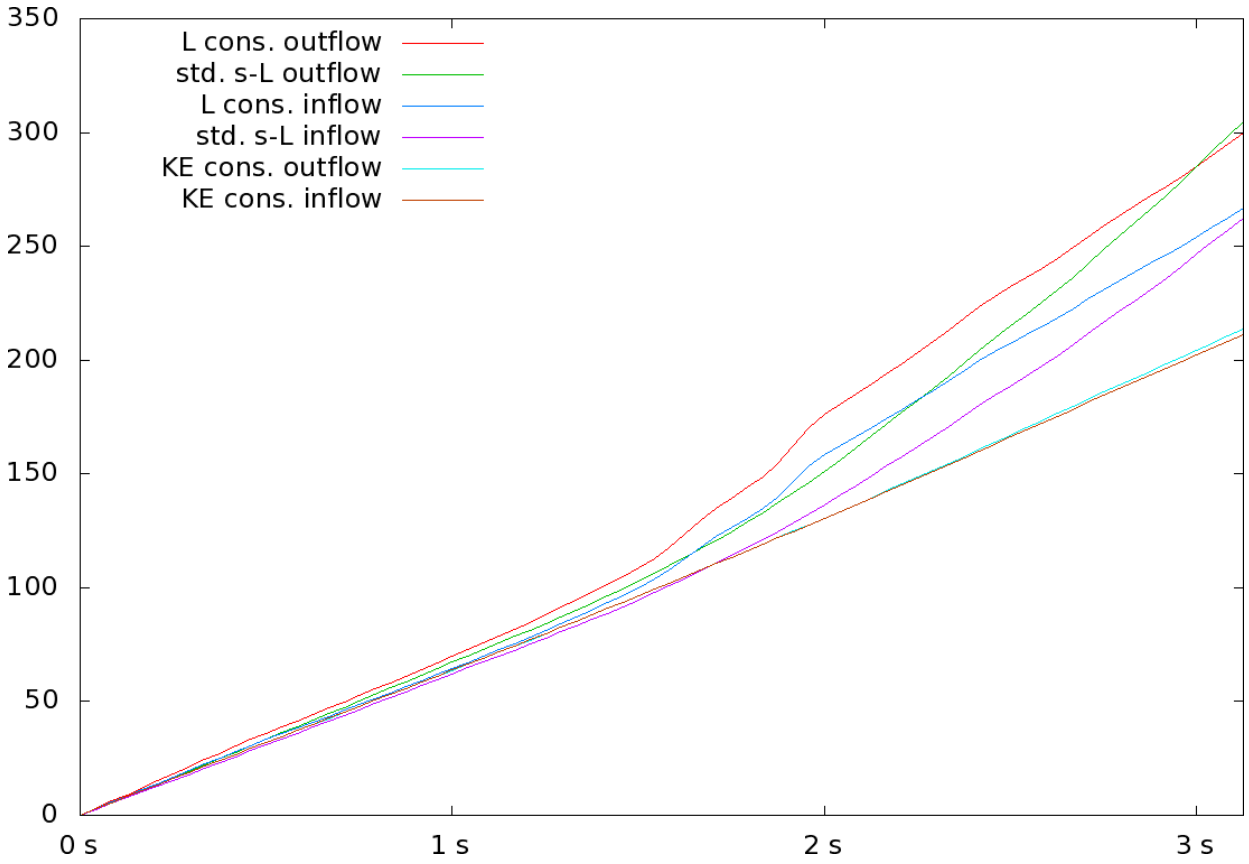


Figure 16: Total kinetic energy fluxing into the computational domain and total kinetic energy fluxing out of the computational domain, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme.

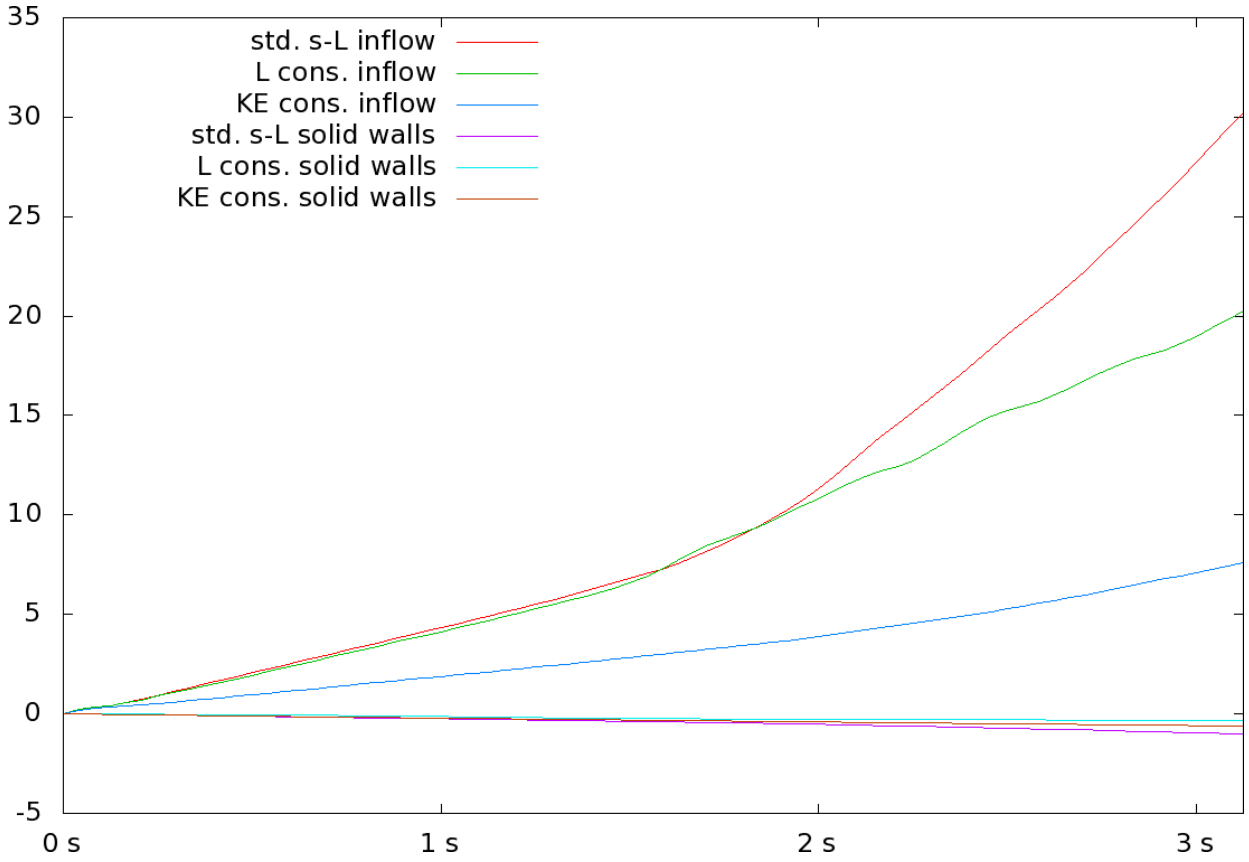


Figure 17: Energy flux into solid wall boundaries, and energy flux entering the computational domain from the inflow boundary condition, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme.

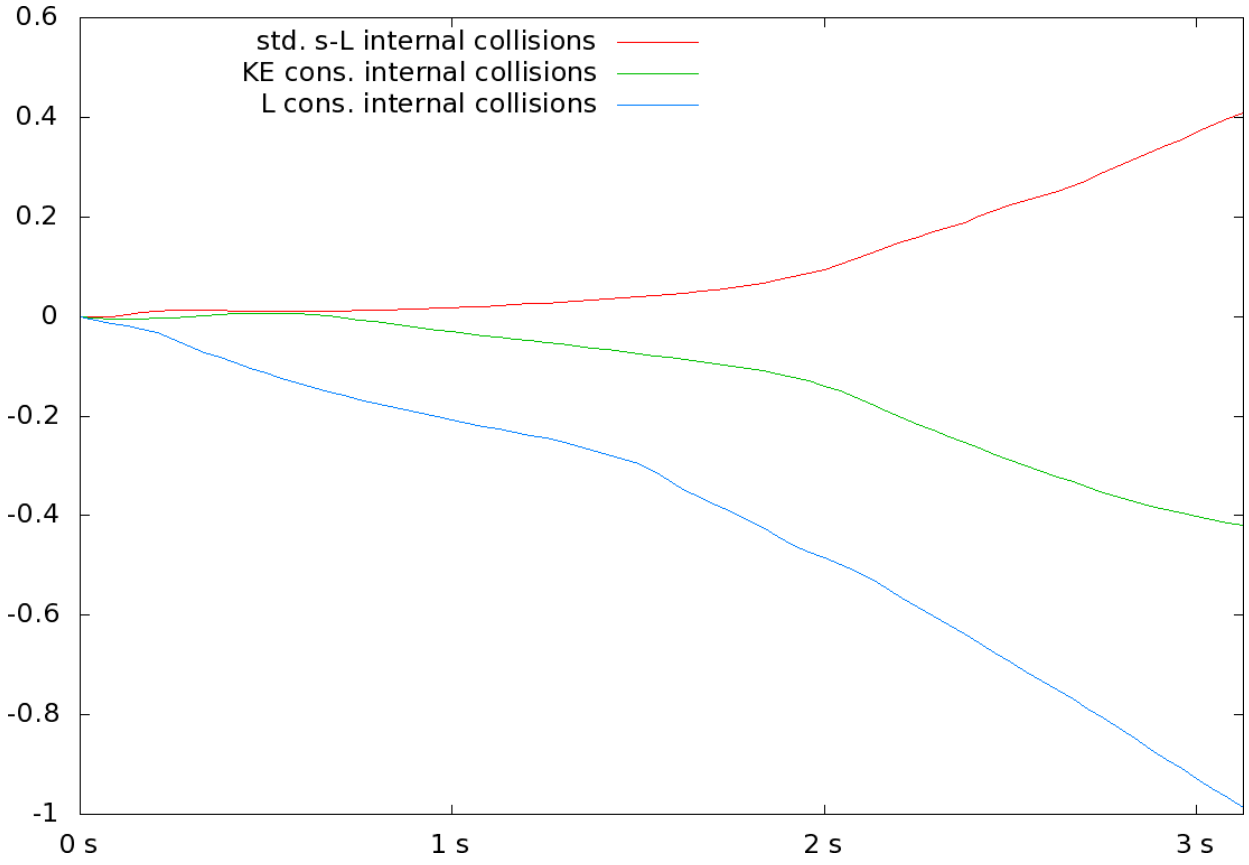


Figure 18: Change in kinetic energy due to the pressure projection step away from boundaries, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme. Note that in all three schemes the change in momentum due to the pressure projection step away from boundaries is zero.

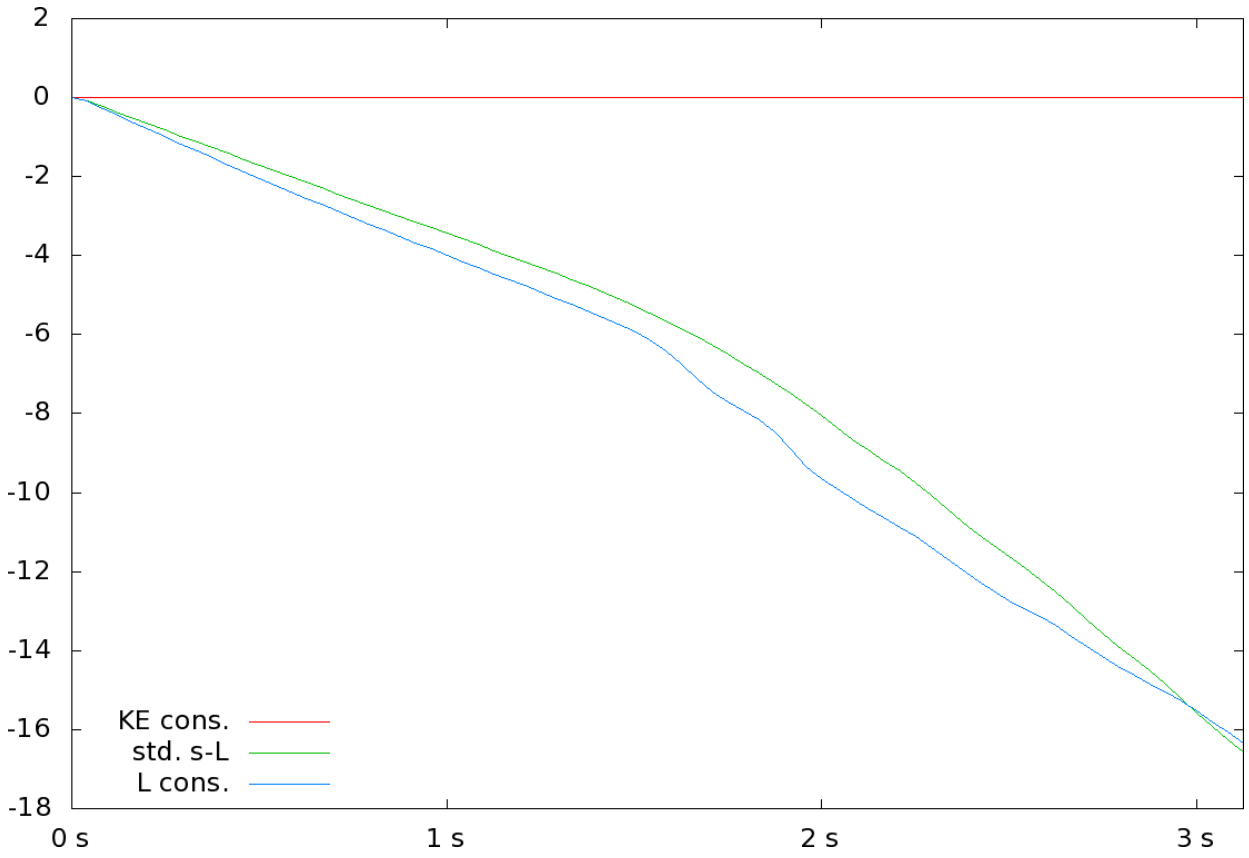


Figure 19: Sum total of kinetic energy in the domain, plus kinetic energy fluxed out of the domain (through outflow and solid wall boundaries), minus kinetic energy fluxed into the domain (through inflow), plus kinetic energy lost in the projection step away from boundaries, plotted as a function of time for a standard semi-Lagrangian scheme, our proposed momentum-conserving scheme, and our proposed kinetic energy-conserving scheme.

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 [8]):

$$p_t + \vec{u} \cdot \nabla p + \rho c^2 \nabla \cdot \vec{u} = 0. \tag{23}$$

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 (22), discretizing the gradient ∇ as G and the divergence $\nabla\cdot$ as $-G^T$ gives the following implicit pressure equation:

$$\left[I + \rho^n (c^2)^n \Delta t^2 G^T \left(\frac{1}{\hat{\rho}^{n+1}} G \right) \right] p^{n+1} = p^a + \rho^n (c^2)^n \Delta t G^T \hat{\vec{u}}^\star. \tag{24}$$

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 (22) as $\hat{\vec{u}}^{n+1} = \hat{\vec{u}}^\star - \Delta t \frac{G p^{n+1}}{(\rho_i + \rho_{i+1})/2}$. This permits us to write the numerically conservative flux-based update as

$$(\rho\vec{u})^{n+1} = (\rho\vec{u})^\star - \Delta t \left(\frac{p^{n+1}_{i+1/2} - p^{n+1}_{i-1/2}}{\Delta x} \right), \qquad E^{n+1} = E^\star - \Delta t \left(\frac{(p\hat{u})^{n+1}_{i+1/2} - (p\hat{u})^{n+1}_{i-1/2}}{\Delta x} \right). \tag{25}$$

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 [21] 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 (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.

5.1. Example

We solve the classic one-dimensional Sod shock tube [50] 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 [21]. The Sod shock tube takes as initial conditions

$$(\rho(x,0), u(x,0), p(x,0)) = \begin{cases} (1,0,1) & \text{if } x \leq .5, \\ (.125,0,.1) & \text{if } x > .5. \end{cases} \tag{26}$$

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 20, where conservative semi-Lagrangian advection is used with the semi-implicit compressible flow solver with a CFL number of .5. Figure 21 shows convergence when a CFL number of 3 is used. We show the same quantities for $t = .8s$ in Figures 23 and 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 [21], as the implicit pressure system is centrally biased; to illustrate this point, we show in

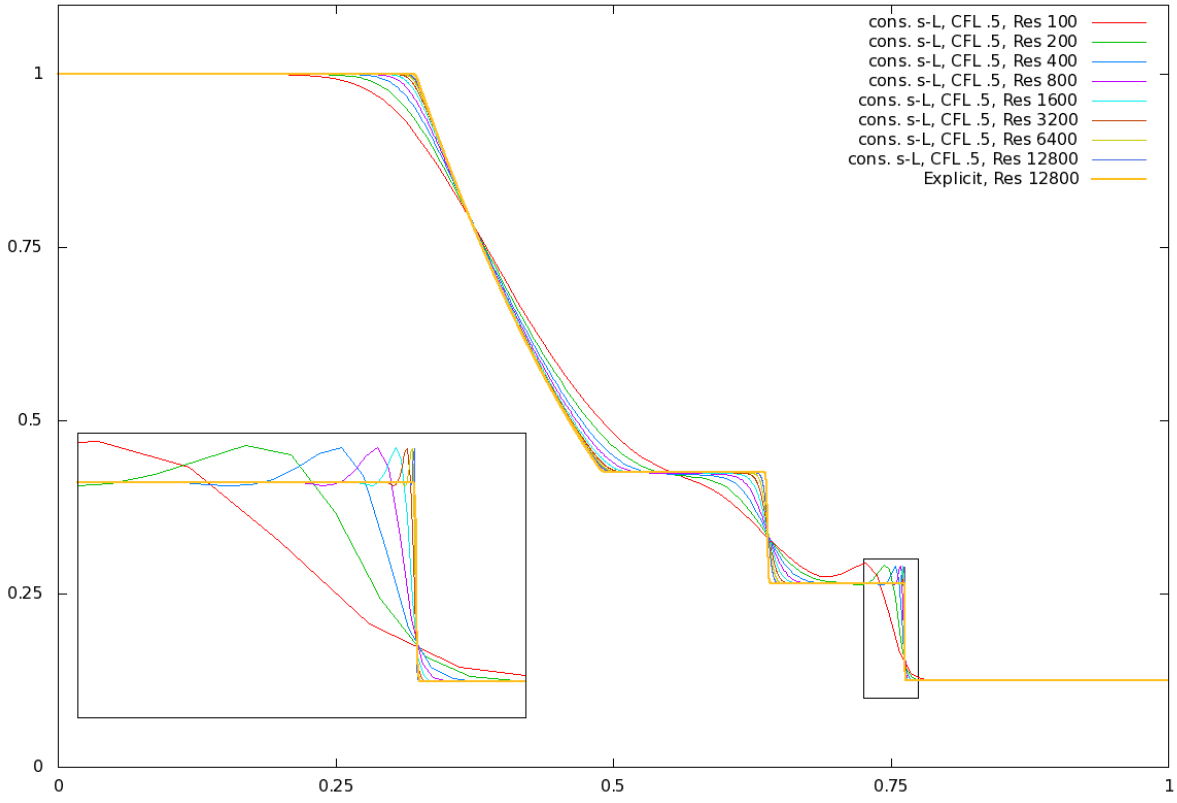


Figure 20: Density profile of a SOD shock tube at $t = .15s$, as generated by the scheme detailed in [21], using our new conservative semi-Lagrangian scheme and a CFL number of .5. We zoom in to the box $[.725, .775] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

Figure 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 [45]. 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 [11].

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.

7. Acknowledgements

Research supported in part by ONR N0014-06-1-0393, ONR N00014-06-1-0505, ONR N00014-05-1-0479 for a computing cluster, and King Abdullah University of Science and Technology (KAUST) 42959. J.G. was supported in part by, and computational resources were provided in part by ONR N00014-06-1-0505 and ONR N00014-09-C-015. M.L. was supported in part by an Intel Ph.D. Fellowship.

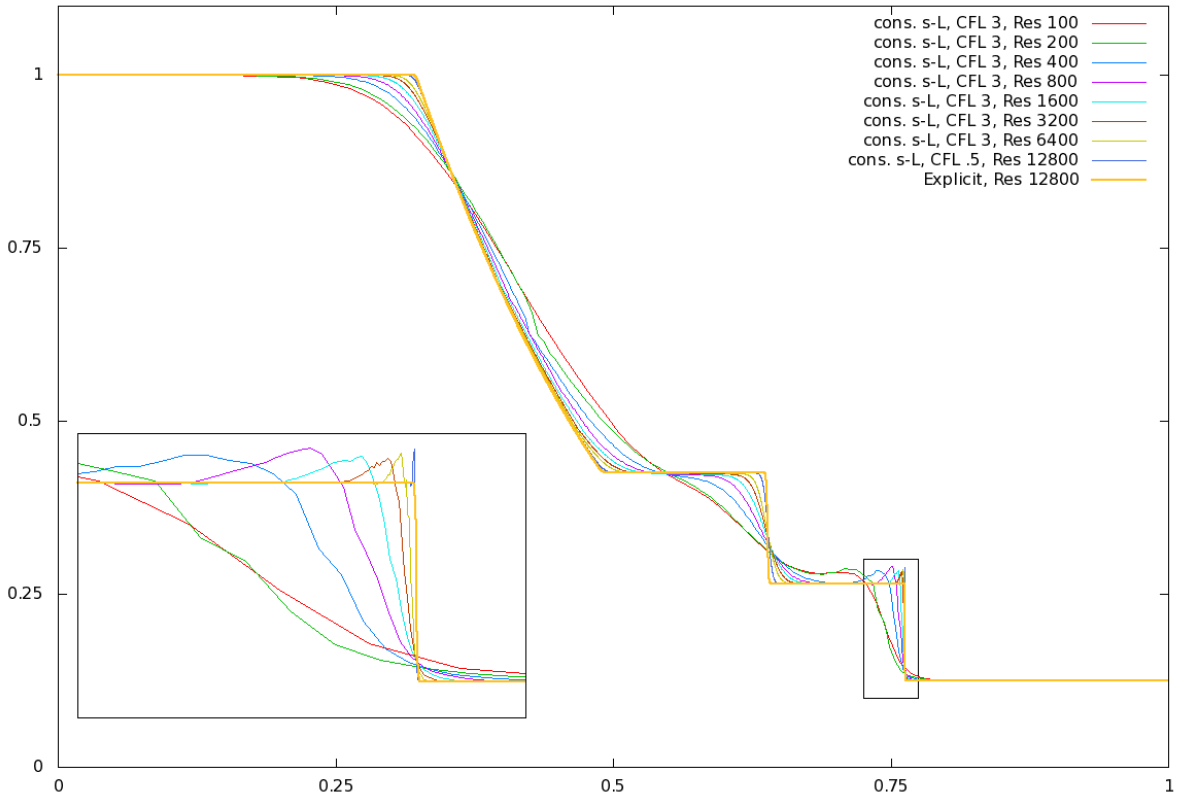


Figure 21: Density profile of a SOD shock tube at $t = .15s$, as generated by the scheme detailed in [21], using our new conservative semi-Lagrangian scheme and a CFL number of 3. We zoom in to the box $[.725, .775] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

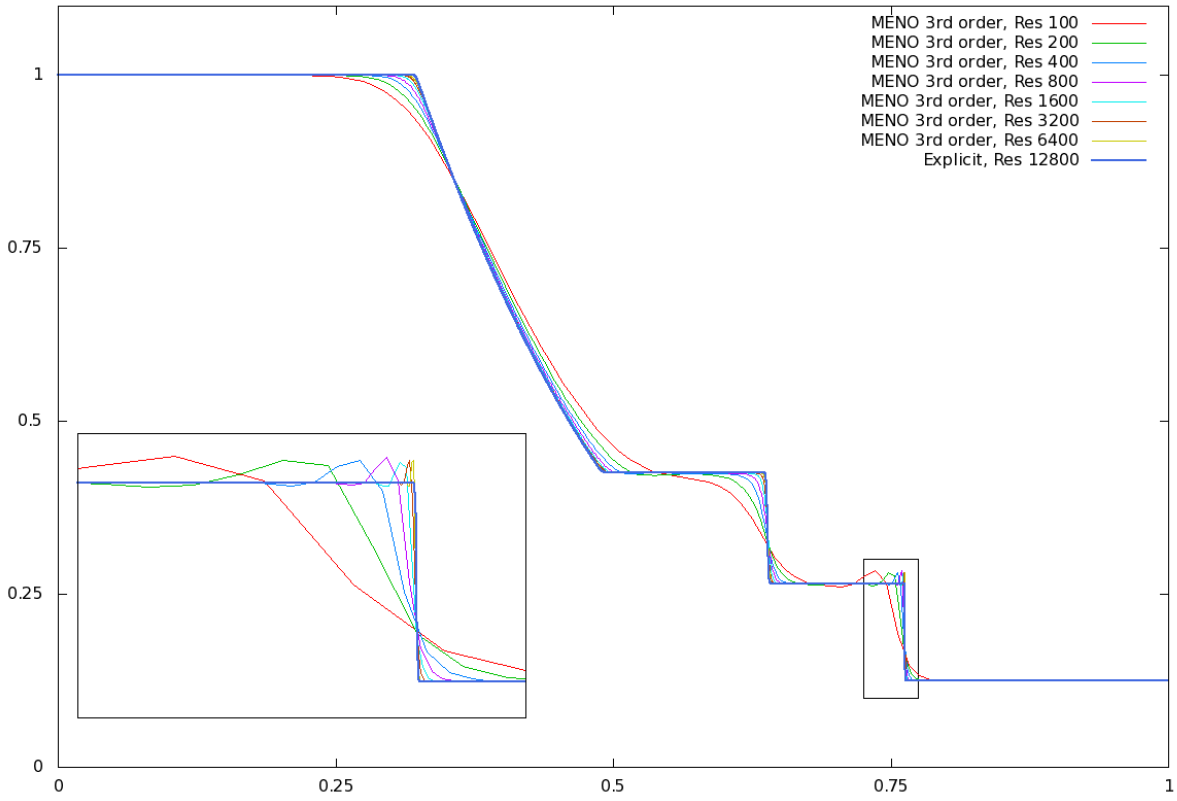


Figure 22: Density profile of a SOD shock tube at $t = .15s$, as generated by the scheme detailed in [21], using a third order MENO advection scheme and a CFL number of .5. We zoom in to the box $[.725, .775] \times [.1, .3]$, showing the shock front in greater detail and highlighting convergence at the discontinuity.

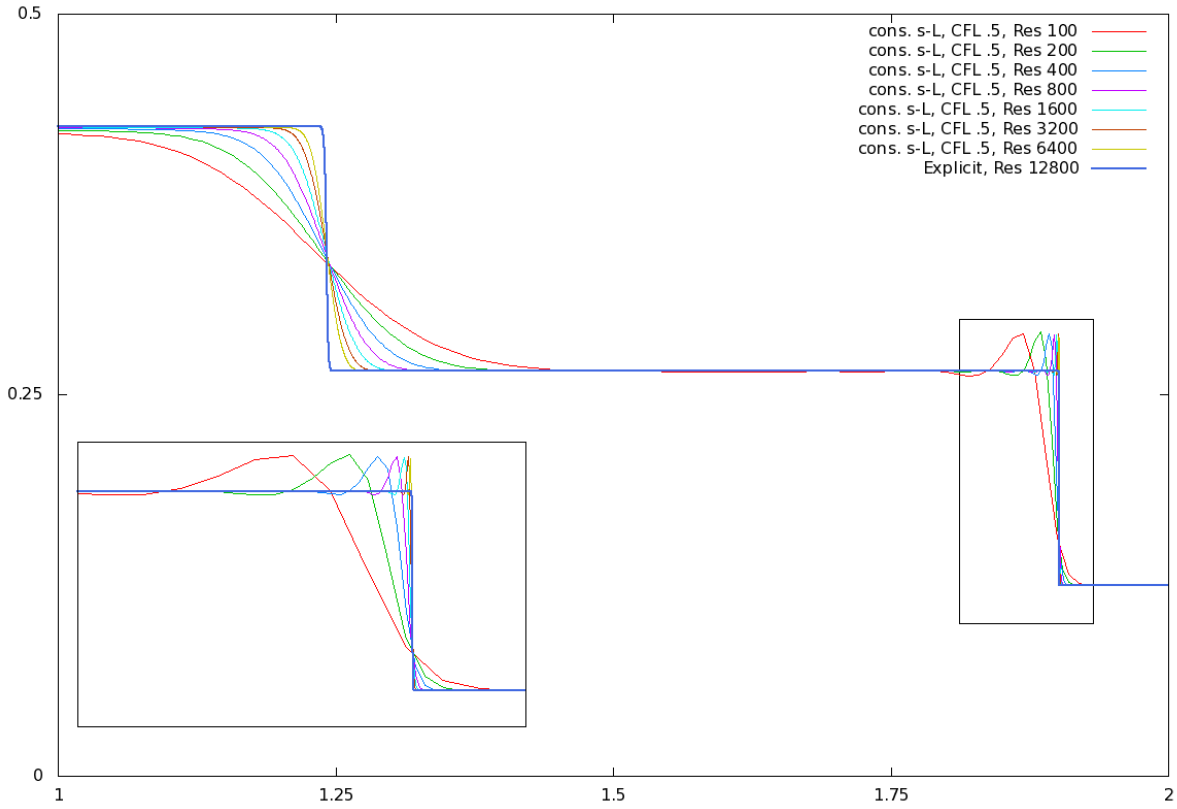


Figure 23: Density profile of a SOD shock tube at $t = .8s$, as generated by the scheme detailed in [21], using our new conservative semi-Lagrangian scheme and a CFL number of .5. In order to capture this later time, we extend the computational domain to $x \in (-1, 2)$ and show only $x \in (1, 2)$ to 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.

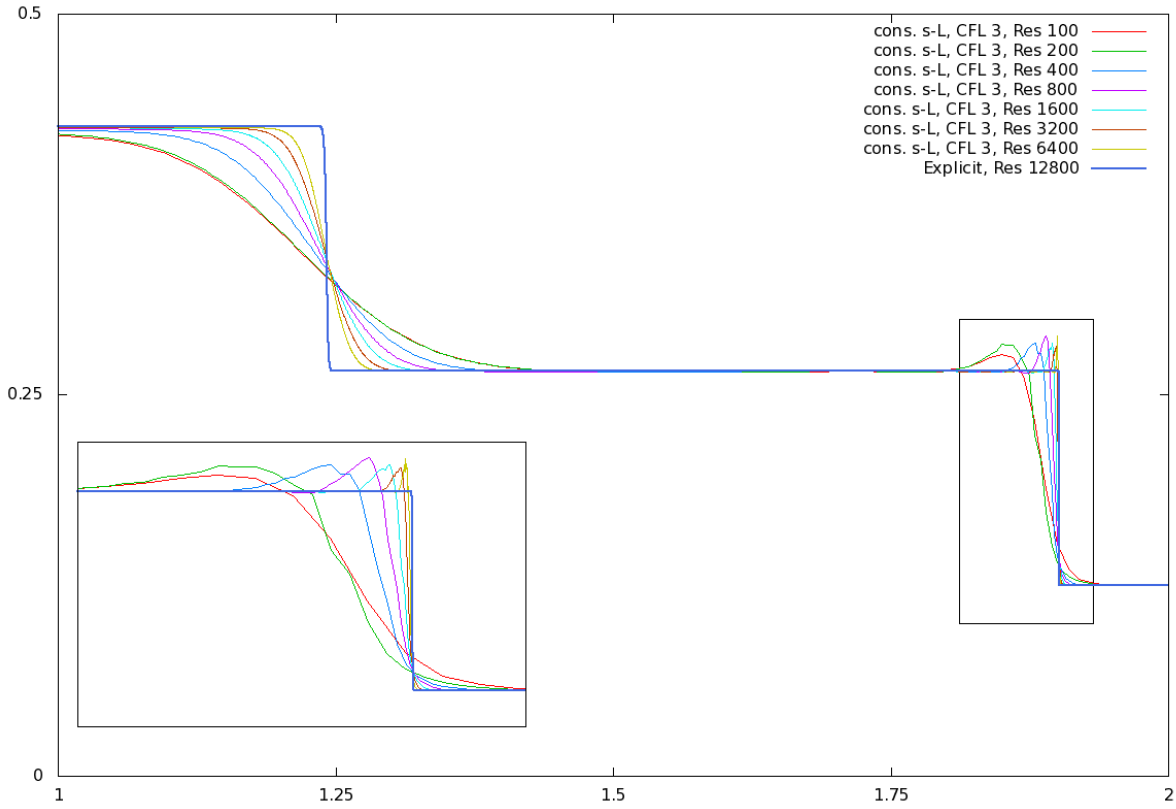


Figure 24: Density profile of a SOD shock tube at $t = .8s$, as generated by the scheme detailed in [21], using our new conservative semi-Lagrangian scheme and a CFL number of 3. In order to capture this later time, we extend the computational domain to $x \in (-1, 2)$ and show only $x \in (1, 2)$ to 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.

8. Bibliography

[1] 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. of Comp. Phys.*, 228(5):1591–1611, 2009.

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

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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. of Comput. Phys.*, 228(22):8380–8393, 2009.

[11] J.T. Grétarsson and R. Fedkiw. Two-way coupling of compressible flows and thin deforming shells. *(In Preparation)*, 2010.

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

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

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

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

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

[22] M. Lentine and R. Fedkiw. Treating kinetic energy in incompressible flows. *(In Preparation)*, 2010.

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

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

[25] 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.

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

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.

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

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

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

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

[37] LG Margolin and M. Shashkov. Remapping, recovery and repair on a staggered grid. *Computer Methods in Applied Mechanics and Engineering*, 193(39-41):4139–4155, 2004.

[38] 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.

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

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

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

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

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

[44] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. An unconditionally stable MacCormack method. *J. of Sci. Comp.*, 35(2):350–371, 2008.

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

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

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

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

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

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

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

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

[53] 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.

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

[55] 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. of Comput. Phys.*, 225(1):935–948, 2007.

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