A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (The Ghost Fluid Method)

Ronald P. Fedkiw *
Tariq Aslam †
Barry Merriman *
Stanley Osher *

October 19, 1998

Abstract

While Eulerian schemes work well on most gas flows, they have been shown to admit nonphysical oscillations near some material interfaces, especially in multimaterial problems associated with deformable solids. In contrast, Lagrangian schemes work well at multimaterial interfaces, but suffer from their own difficulties in problems with large deformations characteristic of most gas flows. We believe that the most robust schemes will combine the best properties of Eulerian and Lagrangian schemes. In fact, this paper is dedicated to developing an Eulerian scheme which treats the interface in Lagrangian way, i.e. as a Heaviside function with no smearing along the lines of [10], [4] and [3].

In this paper, we use a level set function [24, 30, 25] to track the motion of a multimaterial interface in an Eulerian framework. In addition, the use of ghost cells (actually ghost nodes in our finite difference framework) and a new Isobaric Fix [6] technique allows us to keep the density profile from smearing out, while still keeping the scheme robust and easy to program along the lines of [29] with simple extensions to multidimensions and multilevel time integration, e.g. Runge

*UCLA - Research supported in part by ONR N00014-97-1-0027 and ONR N00014-97-1-0968

†Los Alamos National Laboratory - performed under the auspices of the U.S. Department of Energy

Kutta methods. In contrast, [3] and [4] all use ill-advised dimensional splitting for multidimensional problems and [10], [3], and [4] all suffer from great complexity when used in conjunction with multilevel time integrators.

1 Introduction

Eulerian schemes work well for most problems and can accurately and efficiently handle large deformations characteristic of gases. However, they can admit nonphysical oscillations near material interfaces due to the smeared out density profile and the radical change in equation of state across a material interface. Lagrangian schemes work well on material interfaces, since they do not smear out the density profile and it is clear which equation of state is valid at each point. Unfortunately, Lagrangian schemes have their own problems when subjected to large deformations such as those characteristic of gas flow. For a good summary of both Eulerian and Lagrangian schemes, see [2].

Our method consists of combining the robustness of an Eulerian scheme with a multimaterial interface method characteristic of a Lagrangian scheme. We do this by tracking the interface with a level set function [24, 30] which gives the exact subcell interface location. At this interface, we solve an approximate Riemann problem similar to the methods in [10], [4] and [3]. In [10], [4] and [3] the authors use schemes that are intricate in one dimension and can only be extended to multiple dimensions with dimensional splitting in time. In addition, multilevel time integrators, such as the Runge Kutta methods, are hard to implement for these methods. In contrast, our method draws on ideas from [29] which enables us to treat multidimensional calculations without time splitting and allows the easy and efficient implementation of Runge Kutta methods. This is done with an elegant use of ghost cells and the application of a new Isobaric Fix technique [6].

We make note of an alternative method of solving interface problems with Eulerian schemes. In [17], [14], and [5] the authors allow the errors in density associated with a smeared out material interface, and they attempt to fix these errors by modifying the internal energy to get exact cancelation of these error. While some of the preliminary results with a gamma law gas, see e.g. [31], are extremely promising, it is unclear that it will always be possible to remedy the errors associated with a smeared out density profile. In fact, the general pressure evolution equation [5] has a discontinuous coefficient with no meaningful regularization for general equations of state. We have pushed this equation to its limits in [21] and have been disappointed by its lack of robustness. In general, we advocate schemes which do not smear out

the density profile.

2 Equations

2.1 Euler Equations

The basic equations for two-dimensional compressible flow are the 2D Euler equations,

$$\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E+p)u \end{pmatrix}_x + \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E+p)v \end{pmatrix}_y = 0 \tag{1}$$

where t is time, x and y are the spatial dimensions, ρ is the density, u and v are the velocities, E is the total energy per unit volume, and p is the pressure. The total energy is the sum of the internal energy and the kinetic energy,

$$E = \rho e + \frac{\rho(u^2 + v^2)}{2} \tag{2}$$

where e is the internal energy per unit mass. The one-dimensional Euler equations are obtained by setting $v = 0$.

In general, the pressure can be written as a function of density and internal energy, $p = p(\rho, e)$, or as a function of density and temperature, $p = p(\rho, T)$. In order to complete the model, we need an expression for the internal energy per unit mass. Since $e = e(\rho, T)$ we write

$$de = \left(\frac{\partial e}{\partial \rho}\right)_T d\rho + \left(\frac{\partial e}{\partial T}\right)_\rho dT \tag{3}$$

which can be shown to be equivalent to

$$de = \left(\frac{p - Tp_T}{\rho^2}\right) d\rho + c_v dT \tag{4}$$

where c_v is the specific heat at constant volume. [1]

The sound speeds associated with the equations depend on the partial derivatives of the pressure, either p_ρ and p_e or p_ρ and p_T, where the change

of variables from density and internal energy to density and temperature is governed by the following relations

$$p_\rho \rightarrow p_\rho - \left(\frac{p - T p_T}{c_v \rho^2}\right) p_T \tag{5}$$

$$p_e \rightarrow p_\rho + \left(\frac{1}{c_v}\right) p_T \tag{6}$$

and the sound speed c is given by

$$c = \sqrt{p_\rho + \frac{p p_e}{\rho^2}} \tag{7}$$

for the case where $p = p(\rho, e)$ and

$$c = \sqrt{p_\rho + \frac{T(p_T)^2}{c_v \rho^2}} \tag{8}$$

for the case where $p = p(\rho, T)$.

The eigenvalues and eigenvectors for the Jacobian matrix of $\vec{F}(\vec{U})$ are obtained by setting $A = 1$ and $B = 0$ in the following formulas, while those for the Jacobian of $\vec{G}(\vec{U})$ are obtained with $A = 0$ and $B = 1$.

The eigenvalues are

$$\lambda^1 = \hat{u} - c, \quad \lambda^2 = \lambda^3 = \hat{u}, \quad \lambda^4 = \hat{u} + c, \tag{9}$$

and the eigenvectors are

$$\vec{L}^1 = \left(\frac{b_2}{2} + \frac{\hat{u}}{2c}, -\frac{b_1 u}{2} - \frac{A}{2c}, -\frac{b_1 v}{2} - \frac{B}{2c}, \frac{b_1}{2}\right), \tag{10}$$

$$\vec{L}^2 = (1 - b_2, b_1 u, b_1 v, -b_1), \tag{11}$$

$$\vec{L}^3 = (\hat{v}, B, -A, 0), \tag{12}$$

$$\vec{L}^4 = \left(\frac{b_2}{2} - \frac{\hat{u}}{2c}, -\frac{b_1 u}{2} + \frac{A}{2c}, -\frac{b_1 v}{2} + \frac{B}{2c}, \frac{b_1}{2}\right), \tag{13}$$

$$\vec{R}^1 = \begin{pmatrix} 1 \\ u - Ac \\ v - Bc \\ H - \hat{u}c \end{pmatrix}, \quad \vec{R}^2 = \begin{pmatrix} 1 \\ u \\ v \\ H - \frac{1}{b_1} \end{pmatrix}, \tag{14}$$

$$\vec{R}^3 = \begin{pmatrix} 0 \\ B \\ -A \\ -\hat{v} \end{pmatrix}, \quad \vec{R}^4 = \begin{pmatrix} 1 \\ u + Ac \\ v + Bc \\ H + \hat{u}c \end{pmatrix}, \tag{15}$$

where

$$q^2 = u^2 + v^2, \quad \hat{u} = Au + Bv, \quad \hat{v} = Av - Bu, \tag{16}$$

$$\Gamma = \frac{p_e}{\rho}, \quad c = \sqrt{p_\rho + \frac{\Gamma p}{\rho}}, \quad H = \frac{E + p}{\rho}, \tag{17}$$

$$b_1 = \frac{\Gamma}{c^2}, \qquad b_2 = 1 + b_1 q^2 - b_1 H. \tag{18}$$

The eigensystem for the one-dimensional Euler equations are obtained by setting $v = 0$.

2.2 Level Set Equation

We use the level set equation

$$\phi_t + u\phi_x + v\phi_y = 0 \tag{19}$$

to keep track of the interface location as the zero level of ϕ. In general ϕ starts out as the signed distance function, is advected by solving equation 19 using the methods in [11], and then is reinitialized using

$$\phi_t + S(\phi_o)\left(\sqrt{\phi_x^2 + \phi_y^2} - 1\right) = 0 \tag{20}$$

to keep ϕ approximately equal to the distance function near the interface where we need additional information. In this equation, $S(\phi_o)$ is the sign function of ϕ_o with appropriate numerical smearing. More details are given in the appendix.

We note that our method allows us to solve equation 19 independently of the Euler equations. That is, equation 19 can be solved directly using the method in [11], and the eigensystem for the Euler equations does not depend on ϕ, since we will be solving only one phase problems with any given eigensystem (see the later sections). For details on the level set function see [24, 30].

2.3 Equations of State

We will use the following equations of state in our numerical examples.

2.3.1 Gamma Law gas

For an ideal gas $p = \rho RT$ where $R = \frac{R_u}{M}$ is the specific gas constant, with $R_u \approx 8.31451 \frac{J}{molK}$ the universal gas constant and M the molecular weight of the gas. Also valid for an ideal gas is $c_p - c_v = R$ where c_p is the specific heat at constant pressure. Additionally, gamma as the ratio of specific heats $\gamma = \frac{c_p}{c_v}$. [1]

For an ideal gas, equation 4 becomes

$$de = c_v dT \tag{21}$$

and assuming that c_v does not depend on temperature (calorically perfect gas), we integrate to obtain

$$e = c_v T \tag{22}$$

where we have set e to be zero at $0K$. Note that e is not uniquely determined, and we could choose any value for e at $0K$ (although one needs to use caution when dealing with more than one material to be sure that integration constants are consistent with the heat release in any chemical reactions that occur).

Note that we may write

$$p = \rho RT = \frac{R}{c_v}\rho e = (\gamma - 1)\rho e \tag{23}$$

for use in the eigensystem.

2.3.2 Tait Equation of State for Water

We use a stiff equation of state for the water,

$$p = B\left(\frac{\rho}{\rho_o}\right)^{\gamma} - B + A \tag{24}$$

where $\gamma = 7.15$, $A = 10^5 Pa$, $B = 3.31 \times 10^8 Pa$, and $\rho_o = 1{,}000\frac{kg}{m^3}$. In addition, we define

$$e = \frac{B\rho^{\gamma-1}}{(\gamma-1)\rho_o^{\gamma}} + \frac{B-A}{\rho} \tag{25}$$

at the internal energy per unit mass. [32]

Note that this equation of state has $p_e = 0$ which causes a division by zero in the fourth component of $\vec{R}^2$. This can be avoiding with simple rescaling of $\vec{L}^2$ and $\vec{R}^2$ by dividing and multiplying by b_1 respectively. The new eigenvectors become

$$\vec{L}^2 = (-q^2 + H, u, v, -1) \tag{26}$$

and

$$\vec{R}^2 = \begin{pmatrix} b_1 \\ b_1 u \\ b_1 v \\ b_1 H - 1 \end{pmatrix} \tag{27}$$

In addition, to model cavitation, the minimum pressure is set to be $p_{min} = 22.0276 Pa$ [32]. That is, the equation of state becomes $p = p_{min}$ for all densities that would admit pressures lower than p_{min}. Thus, all partial derivatives of pressure are identically zero for densities below

$$\rho_{min} = \rho_o\left(\frac{p_{min} - A + B}{B}\right)^{\frac{1}{\gamma}} \tag{28}$$

and this causes problems in the eigensystem since the sound speed is now zero. To remedy this problem we use a central scheme [22] when $\rho < \rho_{min}$.

2.3.3 JWL Equation of state for Explosive Products

We use the following JWL (Jones-Wilkins-Lee) equation of state for explosive products,

$$p = A\left(1 - \frac{\omega\rho}{R_1\rho_o}\right)\exp\left(-\frac{R_1\rho_o}{\rho}\right) + B\left(1 - \frac{\omega\rho}{R_2\rho_o}\right)\exp\left(-\frac{R_2\rho_o}{\rho}\right) + \omega\rho e \quad (29)$$

where $A = 5.484 \times 10^{11} Pa$, $B = 9.375 \times 10^{9} Pa$, $R_1 = 4.94$, $R_2 = 1.21$, $\omega = .28$, and $\rho_o = 1,630 \frac{kg}{m^3}$. [32]

3 Numerical Method

We use the level set function to keep track of the interface. The zero level marks the location of the interface, while the positive values correspond to one fluid and the negative values correspond to the other. Each fluid satisfies the Euler equations described in the last section with different equations of state on each side. Based on the work in [11], the discretization of the level set function can be done independent of the two sets of Euler equations.

Besides discretizing equation 19 we need to discretize two sets of Euler equations. This will be done with the help of ghost cells. We will describe the scheme with an excessive use of ghost cells for the sake of clarity, and comment on efficiency later.

Given a level set function, it defines two separate domains for the two separate fluids, i.e. each point corresponds to one fluid or the other. Our goal is to define a ghost cell at every point in the computational domain. In this way, each grid point will contain the mass, momentum, and energy for the real fluid that exists at that point (according to the sign of the level set function) and a ghost mass, momentum, and energy for the other fluid that does not really exist at the point (it is on the other side of the interface). Once the ghost cells are defined, we can use standard methods, e.g. see [29], to update the Euler equations at every grid point for both fluids. Then we advance the level set function to the next time step and use this to determine which of the two multidimensional spatial discretizations to use at a given grid point. This makes multidimensional implementation trivial, since it is done in the usual straightforward way, i.e. in the usual way for a single phase fluid with no special concern for the interface, e.g. see [29]. In contrast, [10], [4] and [3] all need ill-advised dimensional splitting for multidimensional problems.

Consider a general time integrator for the Euler equations. In general, we construct right hand sides of the ordinary differential equation for both fluids (based on the methods in [29]), then we advance the level set to the next time level and pick one of the two right hand sides to use for the Euler equations based on the sign of the level set function. This can be done for every step and every combination of steps in a multistep method. Since both fluids are solved for at every grid point, we just choose the appropriate fluid based on the sign of the level set function. This is incredibly simple to

program and apply as opposed to complexity and decision making involved with the use of multilevel time integrators in [10], [4] and [3].

To summarize, the method described here is trivial to implement. Use ghost cells to define each fluid at every point in the computational domain. Update each fluid separately in multidimensional space for one time step or one substep of a multistep time integrator with standard methods. Then update the level set function independently using the real fluid velocities and the sign of the level set function to decide which of the two answers is the valid answer at each grid point. Keep the valid answer and discard the other so that only one fluid is defined at each grid point. Note that multistep time integrators will also require one to save the right hand side of the ordinary differential equation for both fluids for possible use at a later time level. Then define new ghost cells and start over. In this we have regulated all the difficult decision making about special cases on interface crossing, cut cells, etc. to the subroutine that decides how to define the ghost cells. In fact, the entire method relies on the ability to produce ghost cells that satisfy the appropriate boundary conditions for the Euler equations. In this way, one can compute solutions to multiphase flow problems with one's own favorite single phase solver by adding a new routine to define and deal with ghost cells. We chose to use the ENO scheme and TVD Runge Kutta methods from [29].

Lastly, we note that only a band of 3 to 5 ghost cells on each side of the interface is actually needed by the computational method depending on the stencil and movement of the interface. One can optimize the code accordingly.

4 Defining the Ghost Cells

Since a standard one phase solver will be used, the ghost nodes are the key to the numerical method. We have discovered that a straightforward boundary condition capturing approach yields surprisingly good results as is demonstrated by our numerical examples.

4.1 One Dimension

To define the ghost nodes in one spatial dimension, three quantities must be defined in the ghost region, then the equation of state along with the appropriate algebraic relations can be used to get the mass, momentum, and energy.

We choose pressure and velocity as two of our three variables for physical reasons. In many problems, pressure and velocity are continuous across the interface and we can set the pressure and velocity of the ghost fluid identically equal to the pressure and velocity of the real fluid at each point. That is, node by node we can copy the real fluid values of pressure and velocity into the ghost fluid values of pressure and velocity. In this way we capture the interface boundary conditions for the pressure and velocity without explicitly identifying the interface location. Some modification of this procedure is necessary when the pressure and velocity are discontinuous as will be discussed in a future paper.

Once the pressure and velocity have been defined at each ghost node, one more quantity needs to be defined. In [6], it was shown that one degree of freedom exists at a material interface or contact discontinuity. This degree of freedom corresponds to the advection of entropy in the linearly degenerate field. Note that entropy is generally discontinuous at a contact discontinuity. When one applies a standard finite difference scheme to a discontinuous function, large errors result since the truncation error is not small. Shock capturing methods have traditionally avoided the large dispersive errors with a myriad of special techniques while still allowing the large dissipative errors that are usually harmless in a one phase computation. However, these large dissipative errors can be the source of spurious oscillations in a two phase computation.

We eliminate the dissipative errors in the numerical method by using one sided extrapolation of the entropy. Defining the ghost cells with one sided extrapolation of the entropy will create a continuous entropy profile and remove the large errors due to numerical dissipation. Note that the discontinuous nature of the entropy profile dictates that one sided extrapolation will capture the appropriate boundary condition. As discussed in [6], there is a true degree of freedom at a contact discontinuity and one has some choice as to which variable to extrapolate, although one needs to use caution since there will be different degrees of "overheating" errors depending on the variable chosen. See [6] for details.

At this point, we describe the method in detail. Suppose that the zero level of the level set function lies between nodes i and $i+1$, i.e. the level set function changes sign between these nodes. Then fluid 1 is defined at node i and to the left of node i, while fluid 2 is defined at node $i+1$ and to the right of node $i+1$. In order to update fluid 1, we need to define ghost fluid values of fluid 1 at nodes to the right and including node $i+1$. For each of these nodes, we define the ghost fluid value by combining fluid 2's pressure and velocity at each node with the entropy of fluid 1 from node i. This is constant extrapolation of entropy which is actually preferred over high order extrapolation since our interface will behave in a fashion similar to the piston in [6] suffering from "overheating" errors. In fact we always use constant extrapolation of entropy to minimize the "overheating" errors. Likewise, we create a ghost fluid for fluid 2 in the region to the left and including node i. This is done by combining fluid 1's pressure and velocity at each node with the entropy of fluid 2 from node $i+1$.

As discussed in [6], the isobaric fix technique can be used to reduce the "overheating" errors. This technique allows the entropy in real fluid values to change. In order to apply our isobaric fix technique, we change the entropy at node i to be equal to the entropy at node $i-1$ without modifying the values of the pressure and velocity at node i. Likewise, we change the entropy at node $i+1$ to be equal to the entropy at node $i+2$. This completes the isobaric fix, and then the ghost cells are defined as outlined above using these new values for the entropy.

Note that the isobaric fix can be combined with ghost node population in a simple way. For the nodes to the right and including node i, combine the pressure and velocity of each node with the entropy from node $i-1$. This defines fluid 1 to the right and including node i. For the nodes to the left and including node $i+1$, combine the pressure and velocity of each node with the entropy of node $i+2$. This defines fluid 2 to the left and including node $i+1$.

This method is especially effective in multidimensional implementation.

An important aspect of this method is its simplicity. We do not need to solve a Riemann problem, consider the Rankine-Hugoniot jump conditions, or solve an initial boundary value problem at the interface. We capture the appropriate interface conditions by defining a fluid that has the pressure and velocity of the real fluid at each point, but the entropy of some other fluid. Consider the case of air and water. In order to solve for the air, we replace the water with *ghost air* that acts like the water in every way (pressure and velocity) but appears to be air (entropy). In order to solve for the water, we replace the air with *ghost water* that acts like the air in every way (pressure and velocity) but appears to be water (entropy). Since the ghost fluids behave in a fashion consistent with the real fluids that they are replacing, the appropriate boundary conditions are captured. Since the ghost fluids have the same entropy as the real fluid that is not replaced, we are solving a one phase problem. We name this method the "Ghost Fluid Method", not to be confused with ghost cells or ghost nodes which are used in the implementation of the method and have been in use for quite some time.

4.2 Justification

Here we provide a justification of why our method works. Consider the case of a solid wall boundary, where a reflection condition is used for the ghost cells. One can think of this as prescribing waves in the ghost region which are identical to those in the real fluid so that the real fluid does not escape when it interacts with the boundary. Instead, it sees its reflected twin and behaves as if the boundary was impenetrable [33]. Now consider an interface anywhere in a fluid. We want the fluid on one side of the interface to behave in the appropriate way when we add our ghost cells, and thus the easiest thing to do is to let all the ghost values be equal to the real fluid values at that point. In this way, the ghost cells do nothing and the scheme is just the standard Eulerian scheme.

Unfortunately this standard Eulerian scheme does not behave well in certain situations, just as the piston does not behave well in [6] due to "overheating". This implies that a simple modification of the ghost cells is needed similar to [6]. We noticed in [6], that the only modification necessary to cure "overheating" was an isobaric fix. If one thinks of the smearing out of the density profile in a contact discontinuity as a phenomena similar to "over-

heating" than it becomes obvious that the isobaric fix technique will work well here. Thus, the only modification in the ghost cells is to use the isobaric fix technique, while leaving the pressure and velocity unchanged.

4.3 Multidimensions

The above procedure is for one dimensional problems and as trivial as it seems, it does the job. For multidimensional problems, the ghost cells become more involved. We have more than one velocity to deal with, and we need to make some choices for the direction of extrapolation. In multidimensions, we treat the pressure and the three dimensional velocity field in the usual way, just defining the ghost values equal to the real fluids values. In order to finish the ghost cell procedure, we need to apply the isobaric fix technique to all the cells bordering the interface, and we need to extend the isobaric fix variable into the ghost region in a fashion that resembles the constant extrapolation done in the one dimensional case.

A natural way applying the isobaric fix technique exists because of the level set formulation. Using the level set function, we can define the unit normal at every grid point as

$$\vec{N} = \frac{\vec{\bigtriangledown}\phi}{|\vec{\bigtriangledown}\phi|} \tag{30}$$

and then solve a partial differential equation for constant extrapolation in the normal direction. This equation is

$$I_t \pm \vec{N} \cdot \vec{\bigtriangledown} I = 0 \tag{31}$$

where I is the isobaric fix variable, e.g. the entropy. Note that the normal, $\vec{N}$, always points from the negative fluid into the positive fluid. We use the "+" sign in equation 31 to populate a ghost fluid in the region where $\phi > 0$ with the values of I from the region where $\phi < 0$, while keeping the real fluid values of I fixed in the region where $\phi < 0$. Likewise, we use the "-" sign in equation 31 to populate a ghost fluid in the region where $\phi < 0$ with the values of I from the region where $\phi > 0$, while keeping the real fluid values of I fixed in the region where $\phi > 0$. This equation only needs to be solved for a few time steps to populate a thin band of ghost cells needed for the numerical method. Once the ghost cells are populated we can reassemble the conserved variables.

Note that the above procedure does not apply an isobaric fix to the cells in the real fluid which border the interface. In order to apply the isobaric fix, we keep the real fluid values of I fixed in the region where $\phi < -\epsilon$ when using the "+" sign in equation 31, and we keep the real fluid values of I fixed in the region where $\phi > \epsilon$ when using the "-" sign in equation 31. Since ϕ is an approximate distance function, we choose ϵ to be the thickness of the band in which we wish to apply the isobaric fix. We use $\epsilon = 1.5\triangle x$.

4.4 Boundary Conditions

In some multimaterial problems, large jumps in tangential velocity exist at the interface similar to the jumps in density and equation of state that we remedy in this paper. Most schemes will smear out this jump in tangential velocity due to numerical dissipation. An extension of our method allows one to avoid this smearing.

We use the interface normal, $\vec{N}$, to separate the three component velocity field into a tangential and a normal component. The normal component is treated in the same fashion as the velocity in one dimension, i.e. we copy the normal velocity directly into the ghost cells with no change. The tangential velocity is handled in the same way as the isobaric fix variable, i.e. the goal is to extrapolate it or extend it as a constant in the normal direction. In two dimensions, a tangent vector must be chosen consistently in one direction or the other. In three dimensions, one has a difficult time choosing a consistent two dimensional basis for the tangent plane. We remove the difficulty in extension to higher dimensions by applying a basis free projection method similar to the CPM (Complementary Projection Method) [8].

We define the normal at each point by equation 30 and the velocity as $\vec{V} =< u, v, w >$. Once these are defined, we solve the propagation equation 31 where $\vec{I}$ is now a column vector of length four which contains the three dimensional velocity field and the isobaric fix variable. Then at every cell in the ghost region we have two separate velocity fields, one from the real fluid and one from the ghost fluid. Then for each velocity field, the normal component of velocity, $V_N = \vec{V} \cdot \vec{N}$, is put into a three component vector, $V_N \vec{N}$, and then we use a complementary projection idea to define the two dimensional velocity field in the tangent plane by another three component vector, $\vec{V} - V_N \vec{N}$. Then we take the normal component of velocity, $V_N \vec{N}$, from the real fluid and the tangential component of velocity, $\vec{V} - V_N \vec{N}$, from the ghost fluid and add them back together to get our new velocity to occupy

the ghost cell.

We present a simple model to illustrate how the method works. Consider a line in two dimensions or a plane in three dimensions which defines a material interface. Assume that the normal component of velocity is constant across the interface, while the tangential velocity is constant on each side of the interface but jumps across the interface. Consider populating one of the ghost regions with the velocity from the other side of the interface. Since the entire velocity field on one side of the interface is a constant, we are just advecting that constant value into the ghost region. Then we split the velocity field into a normal and tangential component for both the ghost cells and the real fluid. We keep the tangential component from the ghost fluid and the normal component from the real fluid. Since the real fluid has the same normal velocity on both sides of the interface, our procedure is equivalent to just keeping both components of the ghost cell velocity field. This is equivalent to using a constant velocity field, and our method has no knowledge of a jump in velocity at the interface. This allows our method to completely avoid smearing and leads to exact modeling of planar shear waves.

Shear waves may or may not be stable [23]. For example, shear waves are stable in high Mach number flows and when materials have strength (such as steel). Besides the obvious smearing errors, standard schemes may suffer other problems due to their inability to correctly model these shear waves. For example, a shear wave moving across the grid will suffer from a pressure overshoot, while our scheme does not have this problem. In addition, there are many large forces that may be incorrectly excited in material models due to erroneous smeared out velocity profiles. For example, consider two pieces of steel slowly sliding past each other at room temperature. The velocity profile will smear, inducing a continuous, non-constant velocity profile in each piece of steel. This erroneous non-constant velocity profile will induce large non-physical resistant forces from a continuum model.

In many cases a jump in tangential velocity is not stable, and will lead to a Kelvin-Helmholtz instability. This instability is not well posed for the Euler equations, and only becomes well posed when viscosity (or some other regularization such as surface tension) is added, e.g. Navier-Stokes flow.

4.5 A Note On Conservation and Convergence

Here, we briefly discuss the issues of conservation and convergence of the numerical algorithm. The method described in this paper breaks the computational domain into two separate fluids. Within each fluid a standard conservative flux differencing scheme is used. At the interface between the two fluids, there is formally a lack of discrete conservation on a set of measure zero. Fluxes that exists between two different fluids are not unique. Since the pressure and normal velocity in each fluid are the same, these fluxes do have a unique pressure and normal velocity. However, they differ in their values of entropy and tangential velocities. We note that entropy and tangential velocities move with the speed of the fluid, so they do not cross the interface. In addition, there will be a lack of conservation due to the advection of the level set function, ϕ, similar to the area loss problem seen in incompressible flow calculations [30].

Since the scheme is formally non-conservative at the interface, we expect our scheme to behave like a fully conservative scheme with an $O(\Delta x^n)$ source term acting over the material interface. Here n is related to the order at which we are specifying the ghost node states and the order in which we implement the level set method. If the interface length, $L(t)$, is independent of resolution, then the overall lack of conservation will be of $O(\Delta x^n \int_0^t L(t)dt)$. Clearly if $n > 0$, one will achieve conservation. See Section 5.7 for an example where it appears that $n = 2$. Since in this case, conservation is achieved under resolution, and since our discretization is numerically consistent with the Euler equations, we expect to also get convergence to the proper weak solution. Again this is seen in Section 5.7.

In general, we expect that for stable interface flows, the above arguments will hold, and the algorithm should achieve both convergence and conservation under mesh refinement. Unfortunately, the inviscid Euler equations will generally be unstable at material interfaces due to either Kelvin-Helmholtz [12] or Richtmyer-Meshkov [27] types of instabilities. In these cases, the growth rate of an infinitesimal disturbance is usually proportional to the wavenumber of the disturbance [23] and $L(t)$ is resolution dependent. Since under refinement finer scales are introduced, it is likely that $L(t) \propto 1/\Delta x^m$. Here, it is most likely that $m > 0$, and the length of the interface will become larger under refinement. In this case the error in conservation would be of $O(\Delta x^{n-m} \int_0^t L_*(t)dt)$, where L_* is the length of the interface at a particular resolution (i.e. fixed). The fact that the interface may be growing is brought outside the integral and is grouped with n. We expect conservation under

mesh refinement when $n > m$, and expect to lose conservation when $n < m$. Note that it is very difficult in general to determine both n and m just given some initial/boundary value problem. It may be possible that for a physically unstable problem that $n = m$, in which case under refinement one may observe a fixed (and possibly small) error in conservation. The example in Section 5.8 may be of this type, but it is probably best to simply monitor the error in conservation for each problem and to attempt to determine $n - m$ numerically.

It should be noted that many "fully conservative" schemes may conserve overall mass, but may not conserve mass of each constituent [24]. In addition, problems where the Euler equations have instabilities at all wavelengths will never be a resolved even with a perfectly conservative scheme. For the method described in this paper, conservation is achieved under resolution for problems that have a resolvable solution.

5 Examples

Unless otherwise noted the calculations are done with 3rd order ENO-LLF (essentially non-oscillatory - local Lax-Friedrichs) and 3rd order TVD RK (total variation diminishing Runge-Kutta) [29], except where the water cavitates where the 3rd order central scheme [22] is used for the spatial discretization.

5.1 Example 1

In this first example, we explore a simple one phase problem where an Eulerian scheme works well with no oscillations. We will compare our scheme to the standard Eulerian scheme.

This problem was taken from [32]. Consider a gamma law gas with $\gamma = 1.4$ on a $4m$ domain with 100 grid points. The interface is located midway between the 50th and 51st grid points with left and right states defined as $\rho_L = 2\frac{kg}{m^3}$, $\rho_R = 1\frac{kg}{m^3}$, $p_L = 9.8 \times 10^5 Pa$, $p_R = 2.45 \times 10^5 Pa$, and $u_L = u_R = 0\frac{m}{s}$. We ran the code to a final time of .0022 seconds.

The results in figure 1 were obtained with the standard scheme while the results in figure 2 were obtained with the use of the new ghost cell technique where we choose the isobaric fix variable to be entropy for extrapolation, but do not use the isobaric fix itself. Notice the slight overheating near the contact discontinuity. In figure 3, we add the constant entropy isobaric fix to clean up the overheating. All three sets of results are plotted on top of the exact solution.

Note that we still capture the shock and still generate the large dissipative errors characteristic of shock capturing schemes. However, our contact discontinuity no longer suffers from this dilemma.

5.2 Example 2

In this example we compute solutions to "Test A", "Test B", "Test C", and the two cases of "Test D" from [17] where we have redimensionalized the problems. Note that all of these examples have solutions where the *pressure is constant* across the contact discontinuity. Because of this, the pressure

evolution equation gives good results which are also shown in [17], although there are large smearing errors due to numerical dissipation.

5.2.1 "Test A"

For "Test A" we use a $1m$ domain with 100 grid points. The interface is located midway between the 50th and 51st grid points with left and right states defined as $\gamma_L = 1.4, \gamma_R = 1.2, \rho_L = 1\frac{kg}{m^3}, \rho_R = .125\frac{kg}{m^3}, p_L = 1\times 10^5 Pa$, $p_R = 1 \times 10^4 Pa$, and $u_L = u_R = 0\frac{m}{s}$. We ran the code to a final time of .0007 seconds and the results with the standard scheme from [24] are shown in figure 4, while the results using our new scheme with entropy as the isobaric fix variable are shown in figure 5. Both sets of results are plotted on top of the exact solution.

Note that the contact discontinuity is shifted one grid point to the left, since we estimate its speed with the local fluid velocity when advecting the level set function. During wave interactions, the actual velocity of a contact discontinuity can vary slightly from the local fluid velocity. We have performed a grid refinement analysis and the contact discontinuity seems to be off by one grid cell for all levels of grid refinement yielding first order convergence in location as expected for a discontinuity where exact conservation is relaxed slightly. A more resolved solution with 400 grid points is shown in figure 6.

5.2.2 "Test B"

For "Test B" we use a $1m$ domain with 100 grid points. A right going shock is located midway between the 5th and 6th grid points and an interface is located midway between the 50th and 51st grid points. The left, middle, and right states are defined as $\gamma_L = 1.4, \gamma_M = 1.4, \gamma_R = 1.2, \rho_L = 1.3333\frac{kg}{m^3}$, $\rho_M = 1\frac{kg}{m^3}, \rho_R = .1379\frac{kg}{m^3}, p_L = 1.5\times 10^5 Pa, p_M = 1\times 10^5 Pa, p_R = 1\times 10^5 Pa$, $u_L = .3535\sqrt{10^5}\frac{m}{s}$, and $u_M = u_R = 0\frac{m}{s}$. We ran the code to a final time of .0012 seconds and the results with the standard scheme from [24] are shown in figure 7, while the results using our new scheme with entropy as the isobaric fix variable are shown in figure 8. Both sets of results are plotted on top of the exact solution.

In this case, the contact discontinuity is located in the correct cell. Note that the weak rarefaction wave (located to the left) and the weak shock wave

(located to the right) both suffer from numerical dissipation at this level of resolution, independent of the sharp contact discontinuity. A more resolved solution with 400 grid points is shown in figure 9.

5.2.3 "Test D", Case 1

This is similar to "Test B", except that we increase the strength of the shock with $\rho_L = 4.3333\frac{kg}{m^3}$, $p_L = 1.5 \times 10^6 Pa$, and $u_L = 3.2817\sqrt{10^5}\frac{m}{s}$. We ran the code to a final time of .0005 seconds and the results with the standard scheme from [24] are shown in figure 10, while the results using our new scheme with entropy as the isobaric fix variable are shown in figure 11. Both sets of results are plotted on top of the exact solution.

In this case, the contact discontinuity is located in the correct cell. Note that the glitch near $x = .2m$ is due to the capturing of perfect shock initial data by a shock capturing scheme. This is more pronounced in this example, since the shock wave is quite strong. If one starts with a smoothed out shock profile, this glitch is no longer present. A more resolved solution with 400 grid points is shown in figure 12.

5.2.4 "Test C"

This is similar to "Test B", except that we change the fluid on the right to $\gamma_R = 1.249$, $\rho_R = 3.1538\frac{kg}{m^3}$, $p_R = 1 \times 10^5 Pa$, and $u_R = 0\frac{m}{s}$. We ran the code to a final time of .0017 seconds and the results with the standard scheme from [24] are shown in figure 13, while the results using our new scheme with entropy as the isobaric fix variable are shown in figure 14. Both sets of results are plotted on top of the exact solution.

In this case is located in the correct cell, although the shock wave located to the left is shifted two grid points to the right. Once again, these errors are consistent under grid refinement yielding first order accuracy in location. In addition, note that these errors do not increase in time, since they are the result of estimating the velocity of the contact discontinuity by the local fluid velocity during wave interactions. A more resolved solution with 400 grid points is shown in figure 15.

5.2.5 "Test D", Case 2

This is similar to "Test C", except that we increase the strength of the shock with $\rho_L = 4.3333\frac{kg}{m^3}$, $p_L = 1.5 \times 10^6 Pa$, and $u_L = 3.2817\sqrt{10^5}\frac{m}{s}$. We ran the code to a final time of .0007 seconds and the results with the standard scheme from [24] are shown in figure 16, while the results using our new scheme with entropy as the isobaric fix variable are shown in figure 17. Both sets of results are plotted on top of the exact solution.

In this case is located in the correct cell, although the shock wave located to the left is shifted two grid points to the right and the shock wave located to the right is shifted one grid point to the right. Note that the glitches near $x = .3m$ and $x = .7m$ are due to the capturing of perfect shock initial data by a shock capturing scheme. If one starts with a smoothed out shock profile, these glitches are no longer present. A more resolved solution with 400 grid points is shown in figure 18.

5.3 Example 3

We take the initial data for the shock tube problem from "Test A" in [17] as in Example 2. This time we compute in two spatial dimensions on a 200 by 200 grid with the shock tube aligned in the diagonal direction. In figure 19 we show output from the off-diagonal direction. Note that we ran the code for $\sqrt{2}$ times longer in order to get a good comparison with "Test A" in Example 2. The results are plotted on top of the exact solution.

5.4 Example 4

This problem was taken from [32]. Consider a $4m$ domain with 100 grid points and the interface located midway between the 50th and 51st grid points. There is a JWL gas on the left and water on the right with initial states of $\rho_L = 1630\frac{kg}{m^3}$, $\rho_R = 1000\frac{kg}{m^3}$, $p_L = 7.81 \times 10^9 Pa$, $p_R = 1.0 \times 10^5 Pa$, and $u_L = u_R = 0\frac{m}{s}$.

Since the equation of state for water has pressure as a function of density only, one needs to be careful when choosing the isobaric fix variable. The most natural choice for water is the internal energy. For simplicity, we do not use the isobaric fix technique for the JWL gas, and we extend density directly into the ghost cells.

We ran the code to a final time of .0005 seconds and the results using our new scheme are shown in figure 20. The results compare well with the exact solution in [32].

5.5 Example 5

This problem was taken from [32]. Consider a $10m$ domain with 500 grid points and reflection boundary conditions applied to both sides of the domain. The interface is located midway between the 250th and 251st grid points, with a gamma law gas, $\gamma = 1.25$, on the left and water on the right. The initial states are $\rho_L = 1\frac{kg}{m^3}$, $\rho_R = 1000\frac{kg}{m^3}$, $p_L = 1.0 \times 10^5 Pa$, $p_R = 1.0 \times 10^5 Pa$, and $u_L = u_R = 0\frac{m}{s}$. In addition, we have a shock in each fluid. Grid points 1 to 48 have $\rho = 8.26605505\frac{kg}{m^3}$, $p = 1.0 \times 10^7 Pa$, and $u = 2949.97131\frac{m}{s}$. Grid points 481 to 500 have $\rho = 1004.1303\frac{kg}{m^3}$, $p = 1.0 \times 10^7 Pa$, and $u = -6.3813588\frac{m}{s}$.

Since the equation of state for water has pressure as a function of density only, one needs to be careful when choosing the isobaric fix variable. The most natural choice for water is the internal energy. We use entropy as the isobaric fix variable in the gas.

We ran the code to a final time of .003 seconds and the results using our new scheme are shown in figure 21 where we plot $\log_{10} \rho$ instead of the density, so that one may see the shock in the gas. In the figure, we use **RED** for the gas and **GREEN** for the water. In addition, note that the entropy field in the water is not used, so we set it to zero for graphing purposes. The results compare well with the solution computed in [32].

Note that the pressure evolution equation method in [17] has a difficult time dealing with these sorts of contact discontinuities where the velocity and pressure are not both constant.

5.6 Example 6

This problem was taken from [32]. Consider a $10m$ domain with 400 grid points. A reflection boundary condition is applied to the left hand side of the domain, while an outflow boundary condition applied to the right hand side of the domain. Water is located in the central part of the domain surrounded by a gamma law gas, $\gamma = 1.3$, on each side. There are two interfaces, one between the 40th and 41st cell and one in between the 120th and 121st cell.

The gas has initial values of $\rho = 35\frac{kg}{m^3}$, $p = 1.0 \times 10^7 Pa$, and $u = 500\frac{m}{s}$, while the water has initial values of $\rho = 1004.1303\frac{kg}{m^3}$, $p = 1.0 \times 10^7 Pa$, and $u = 500\frac{m}{s}$.

This problem is computationally challenging so we modify our numerical method slightly by choosing the high order viscosity for the ENO-LLF scheme as the largest of the three separate field viscosities as opposed to the usual field by field choice. In addition, we only use the second order accurate version of the spatial method in the water.

Since the equation of state for water has pressure as a function of density only, one needs to be careful when choosing the isobaric fix variable. The most natural choice for water is the internal energy. We use entropy as the isobaric fix variable in the gas.

We ran the code to a final time of .007 seconds and the results using our new scheme are shown in figure 22 where we plot $\log_{10} \rho$ instead of the density, so that one may see the shock in the gas. In the figure, we use **RED** for the gas and **GREEN** for the water. In addition, note that the entropy field in the water is not used, so we set it to zero for graphing purposes. The results compare well with the solution computed in [32].

Note that the pressure evolution equation method in [17] has a difficult time dealing with these sorts of contact discontinuities where the velocity and pressure are not both constant.

5.7 Example 7

We examine the convergence and conservation of a stable flow field with an interface. The problem is linear advection of a helium bubble in air and the nondimensionalized initial conditions are,

$$(\rho = 1, u = 1, v = 1, p = 1, \gamma = 1.4) \qquad \text{air} \tag{32}$$

$$(\rho = .138, u = 1, v = 1, p = 1, \gamma = 1.67) \qquad \text{helium} \tag{33}$$

$$\phi = -0.2 + \sqrt{(x - .25)^2 + (y - .25)^2} \qquad \text{level set} \tag{34}$$

where $\phi \leq 0$ represents helium and $\phi > 0$ represents the air. No reinitialization of the level set function was done. For this advection problem our scheme achieves the exact state in each of the fluid regions, and the only

error incurred is from the advection of the level set function. This may seem like a trivial example, but most standard conservative or pressure evolution schemes would smear out the density and possibly create spurious pressure oscillations.

A series of experiments were carried out on the unit square to measure the convergence to the exact solution and to analyze how well the scheme conserves the mass of each fluid. Zero gradient boundary conditions were used for the conservative fluid variables, and linear extrapolation at the boundaries was used for ϕ. We used a centrally biased ENO scheme [28] applied in a central framework [22, 33] with third order TVD Runge-Kutta time integration and third order WENO for the advection of ϕ [15]. We used $\Delta t = 0.1\Delta x$ and integrated to $t = 0.5$. We measured two discrete errors, namely the L1 error in the density field, E_ρ, and the relative error in total mass of helium, E_{He}, at $t = 0.5$. The errors and numerical rates of convergence, R_c, are given in Table I. Clearly the errors in both the density field and in total mass conservation converge at second order.

TABLE I: Numerical accuracy for helium advection in air.

$\Delta x = \Delta y$	E_ρ	R_c	E_{He}	R_c
1/10	5.17×10^{-2}		5.00×10^{-1}	
1/20	8.62×10^{-3}	2.58	8.16×10^{-2}	2.62
1/40	2.15×10^{-3}	2.00	2.03×10^{-2}	2.01
1/80	5.39×10^{-4}	2.00	5.02×10^{-3}	2.02

5.8 Example 8

In this example, we will illustrate the difficulty in computing shear waves with shock capturing schemes and demonstrate the potential benefits of our new method. A full computational analysis of these issues will be treated in a future paper on viscous flow.

Consider a $1m$ square domain with $\gamma = 1.4$, $\rho = 1\frac{kg}{m^3}$, and $p_L = 1 \times 10^5 Pa$ everywhere. An interface is located at $x = .5m$ with tangential velocities of $v = 300\frac{m}{s}$ on the left and $v = 200\frac{m}{s}$ on the right. In addition, we impose a normal velocity of $u = 50\frac{m}{s}$ directed to the right. The flow is inviscid and a shear wave should be advected to the right with a perfect slip boundary condition. We use a coarse mesh of 20 grid points in each direction and plot

the $y = \frac{10/19}{m}$ cross-section of this initial data in figure 23. A shock capturing scheme will smear out this shear wave creating the errors shown in figure 24 at .0005 seconds and figure 25 at .005 seconds. Using the slip boundary condition in our new scheme results in an extremely sharp solution as shown in figure 26 at .005 seconds.

5.9 Example 9

We examine a Mach 1.22 air shock collapse of a helium bubble. Experimental results may be found in [13] and a numerical solution using adaptive mesh refinement (AMR) may be found in [31]. The physical initial conditions for this problem are given in figure 27, where the upper and lower boundary conditions are reflection for solid wall boundaries. The left and right boundary conditions were zero gradient for the flow variables and linear extrapolation for ϕ. The nondimensionalized initial conditions are,

$$(\rho = 1, u = 0, v = 0, p = 1, \gamma = 1.4) \qquad \text{pre-shocked air} \tag{35}$$

$$(\rho = 1.3764, u = -.394, v = 0, p = 1.5698, \gamma = 1.4) \qquad \text{post-shocked air} \tag{36}$$

$$(\rho = .138, u = 0, v = 0, p = 1, \gamma = 1.67) \qquad \text{helium} \tag{37}$$

$$\phi = -25 + \sqrt{(x - 175)^2 + y^2} \qquad \text{level set} \tag{38}$$

where $\phi \leq 0$ represents helium and $\phi > 0$ represents the air. The post-shock air state was given for $x > 225$. No reinitialization of the level set function or isobaric fix was done. We used a centrally biased ENO scheme [28] applied in a central framework [22, 33] with third order TVD Runge-Kutta time integration and third order WENO for the advection of ϕ [15]. Note that the computational domain was only the top half of the physical domain with a reflection condition applied at $x = 0$. A series of experiments were carried out at different resolutions ($\Delta x = 2, 1, 0.5, 0.25$) at $CFL = 0.8$.

Figure 28 shows an idealized Schlieren image corresponding to $427\mu s$ after the air shock encounters the helium bubble ($\Delta x = 0.25$). The image was generated in the exact same manner as described in Section 3.3 of [31]. Also shown in figure 28 is a circle representing the original helium-air interface to make the comparison easier with figure 9(h) of [31] and figure 7(h) of

[13]. The comparisons between the previous AMR solution and experiment are in good agreement for the general bubble shape and position. There are differences in the details of the interface, which is to be expected since for this problem the interface is unstable, and without some regularization there will be no unique or resolved answer to the Euler equations. For this problem the series of resolutions ($\Delta x = 2, 1, 0.5, 0.25$) gave (2.5%, 0.78%, 0.42%, 0.43%) as the time averaged relative percent errors in helium mass, respectively. Clearly this error in conservation of mass is not very significant, and although it appears to be generally getting better with resolution, we make no conjecture that conservation will be achieved under resolution to unstable problems.

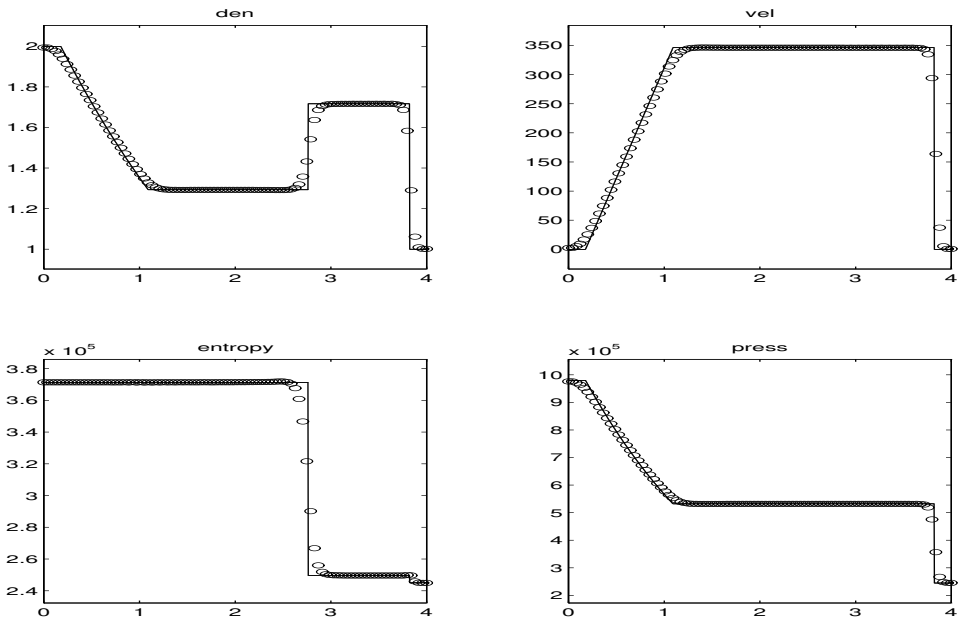

Figure 1: Standard Scheme - 100 points

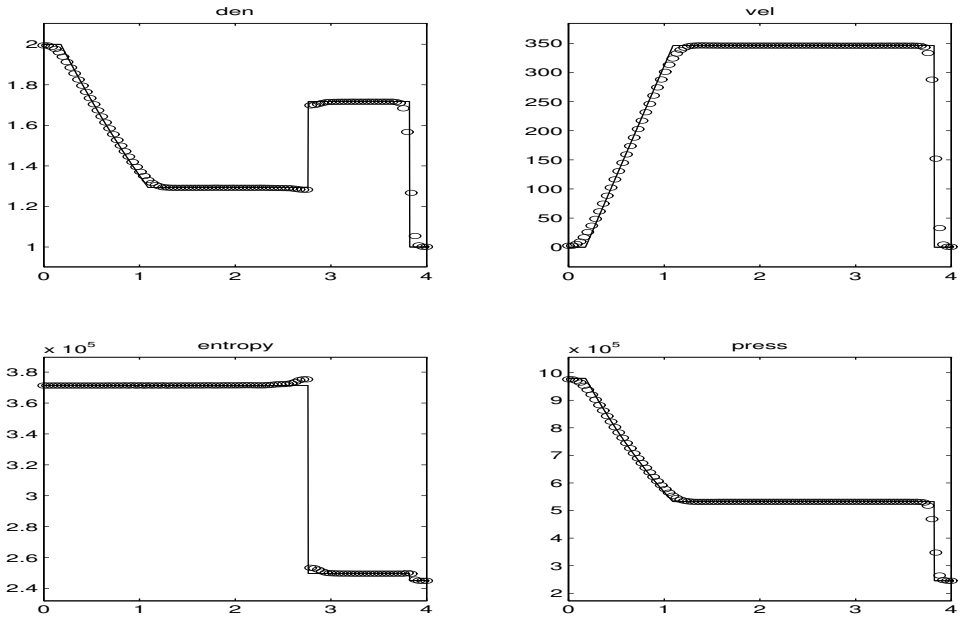

Figure 2: Ghost Fluid Method - without isobaric the fix - 100 points

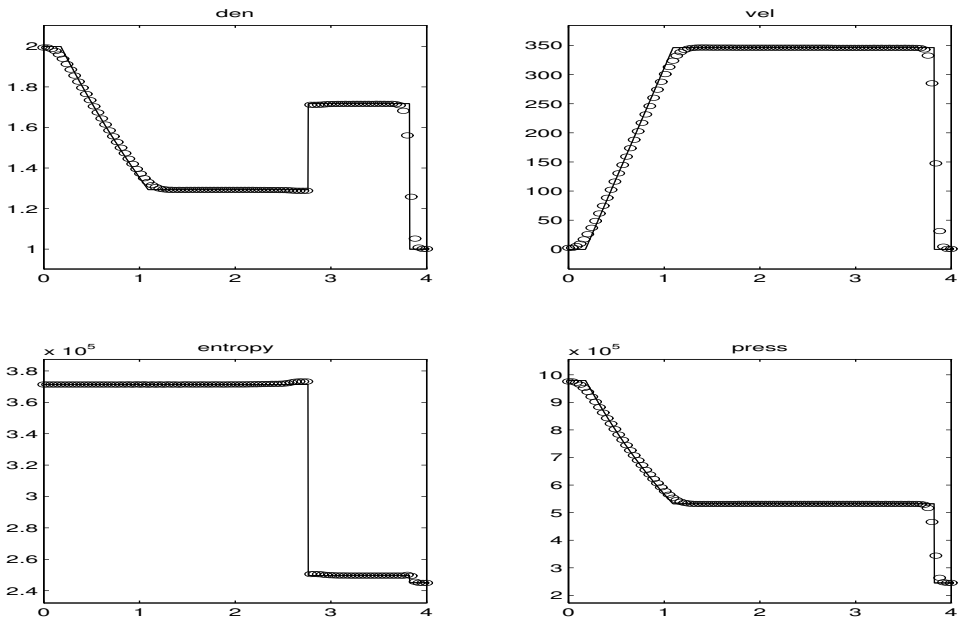

Figure 3: Ghost Fluid Method - with the isobaric fix - 100 points

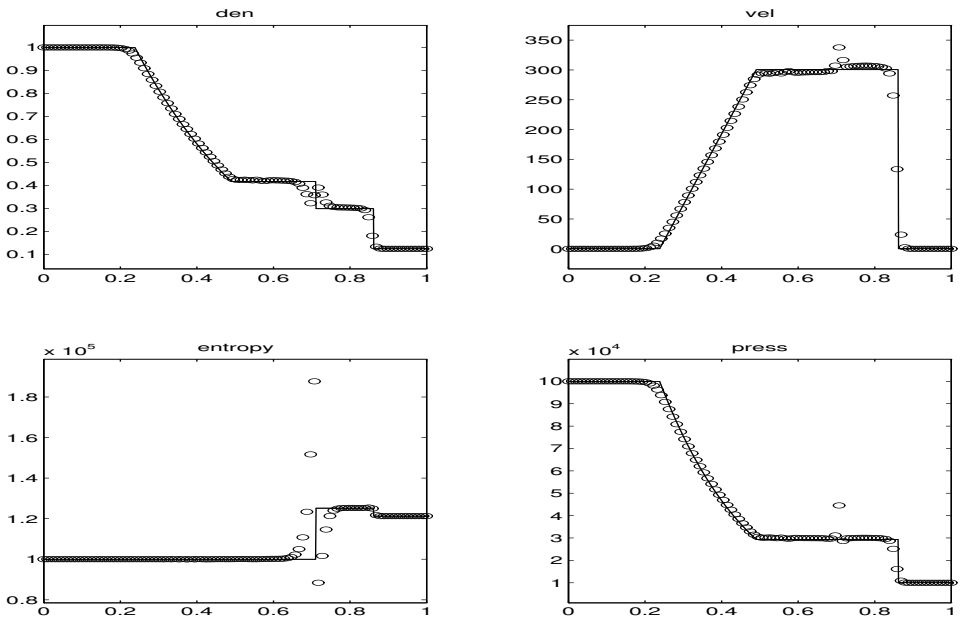

Figure 4: Test A - Standard Scheme - 100 points

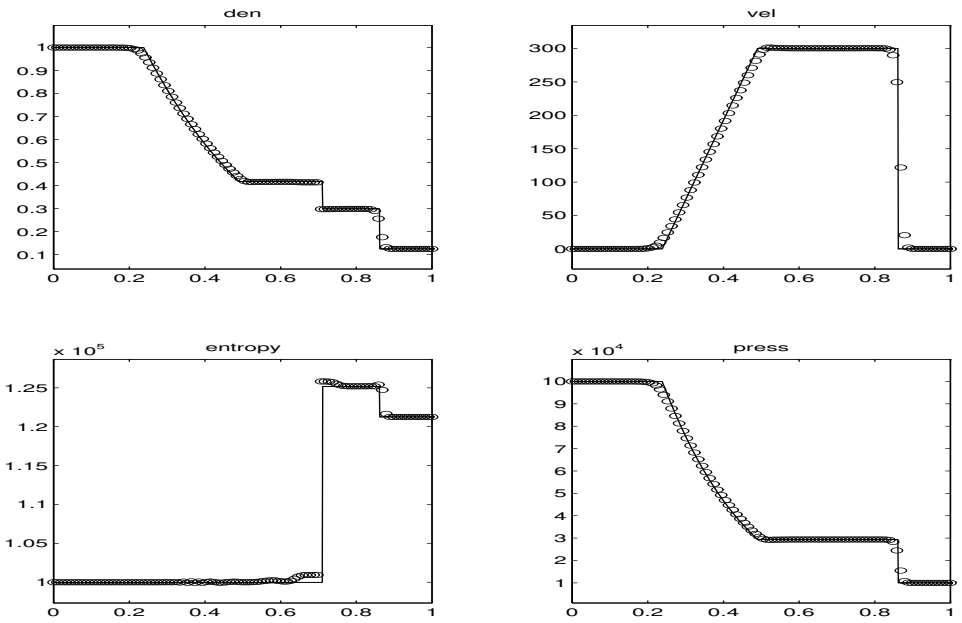

Figure 5: Test A - Ghost Fluid Method - 100 points

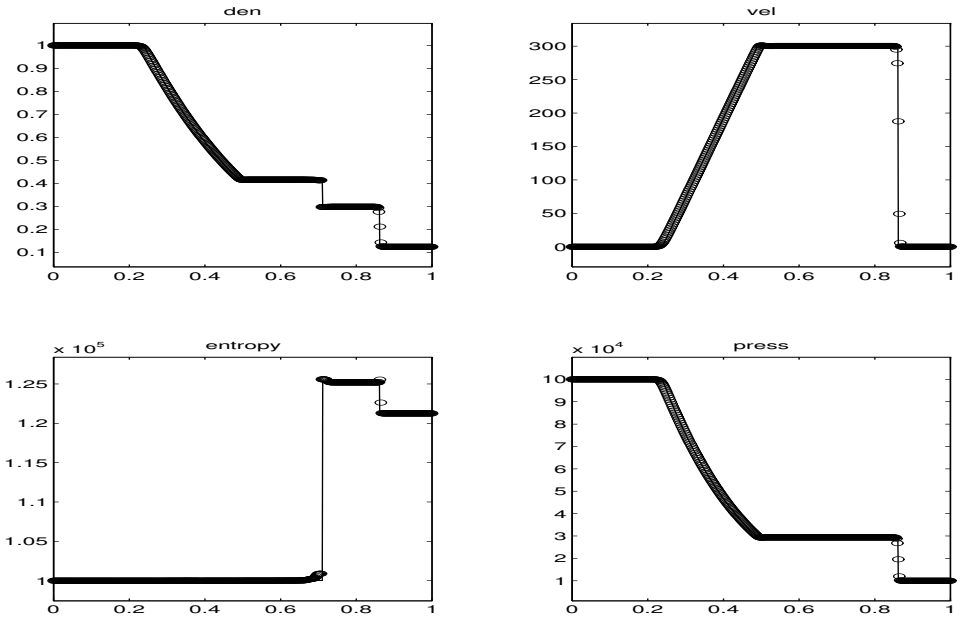

Figure 6: Test A - Ghost Fluid Method - 400 points

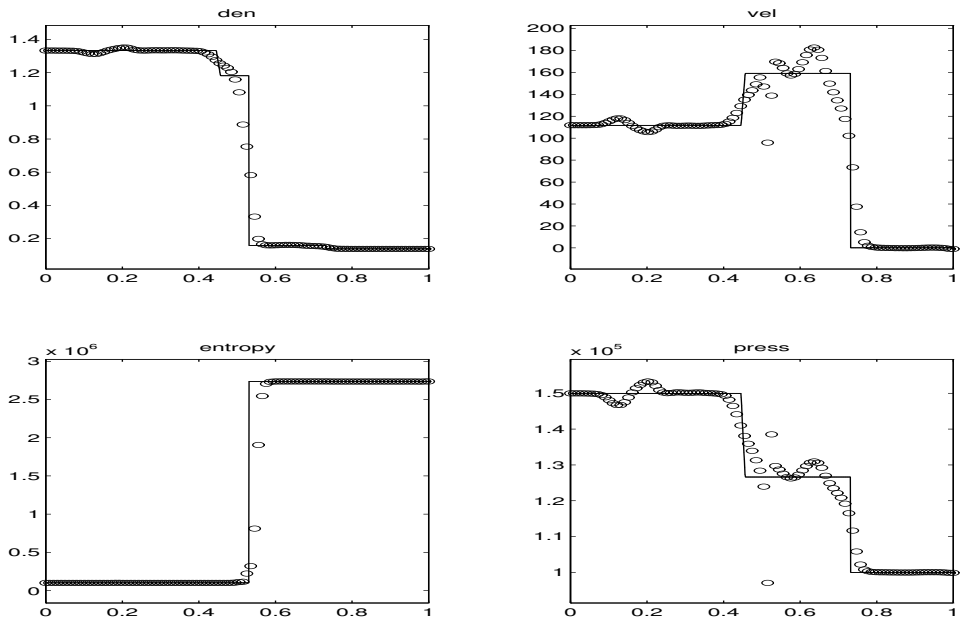

Figure 7: Test B - Standard Scheme - 100 points

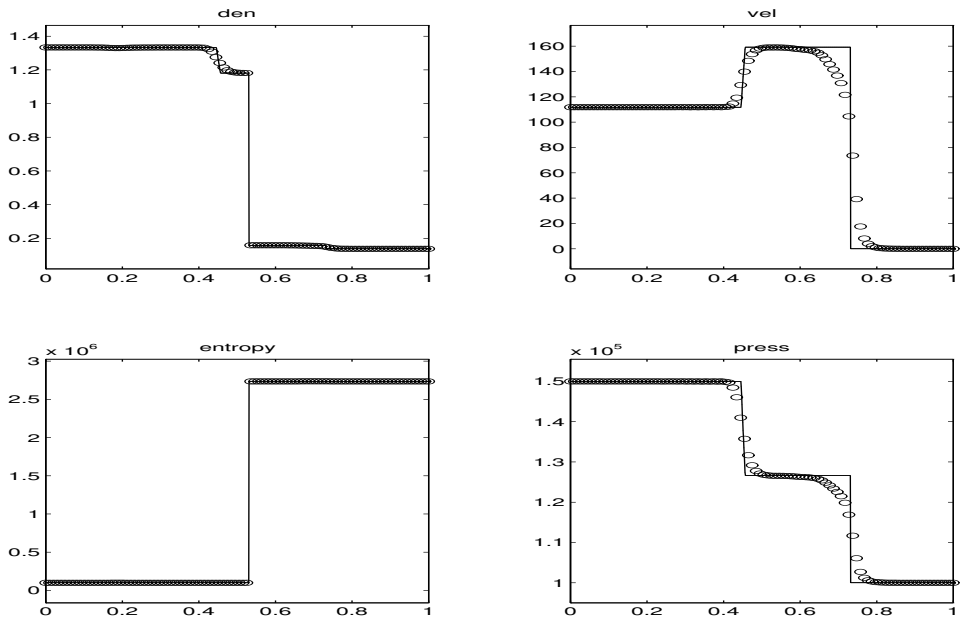

Figure 8: Test B - Ghost Fluid Method - 100 points

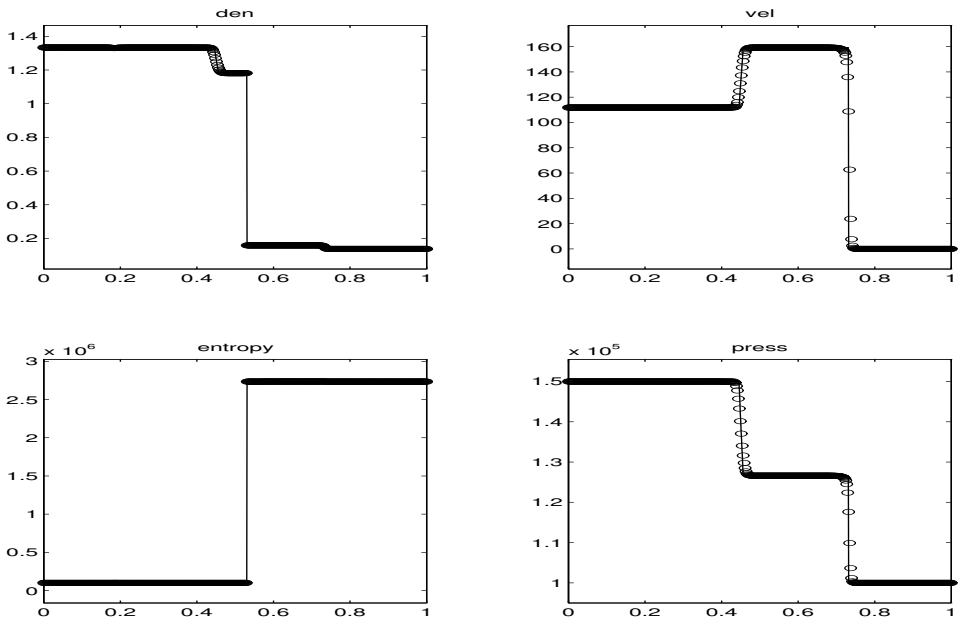

Figure 9: Test B - Ghost Fluid Method - 400 points

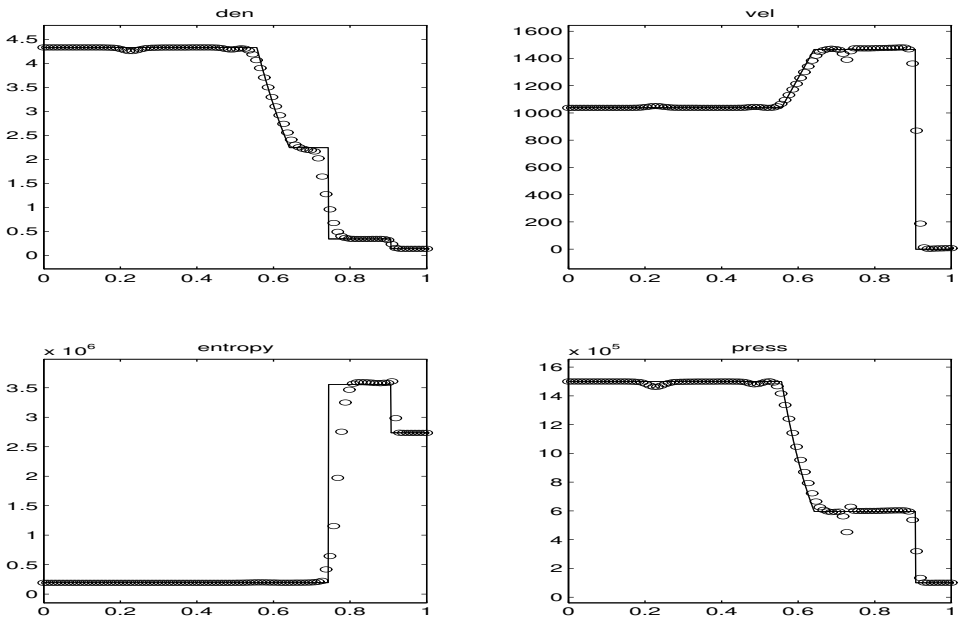

Figure 10: Test D, Case 1 - Standard Scheme - 100 points

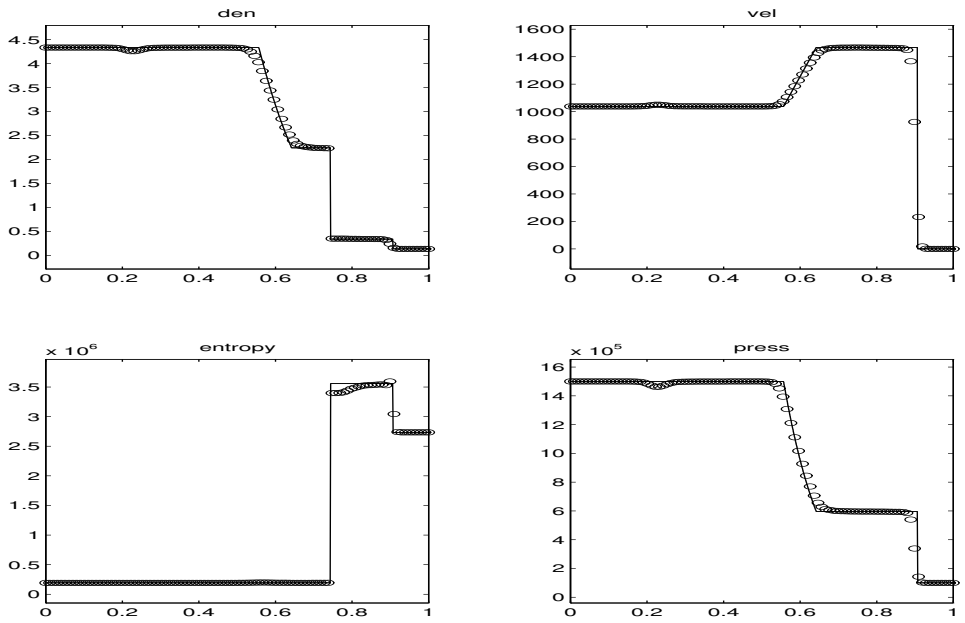

Figure 11: Test D, Case 1 - Ghost Fluid Method - 100 points

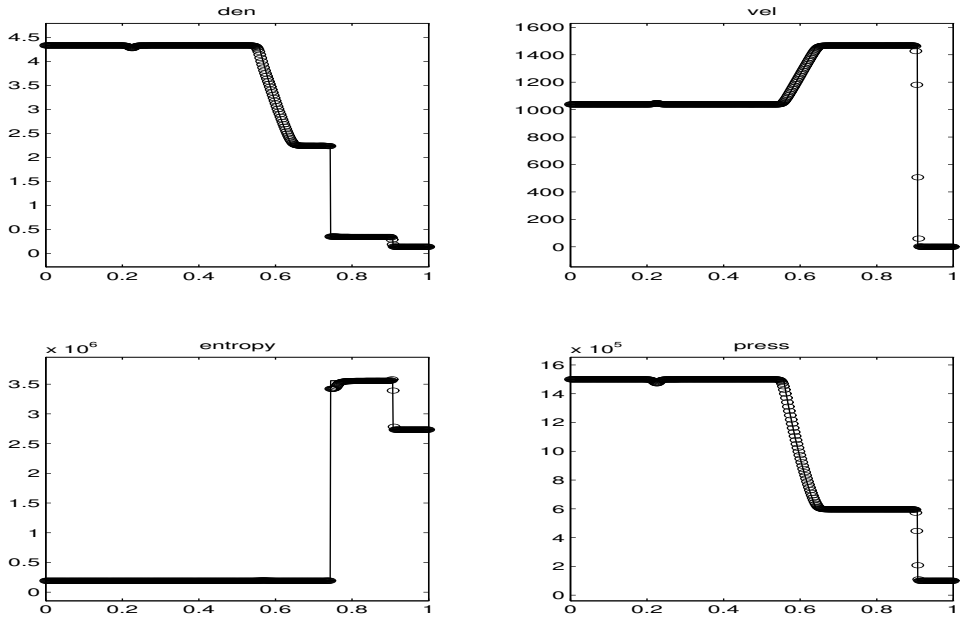

Figure 12: Test D, Case 1 - Ghost Fluid Method - 400 points

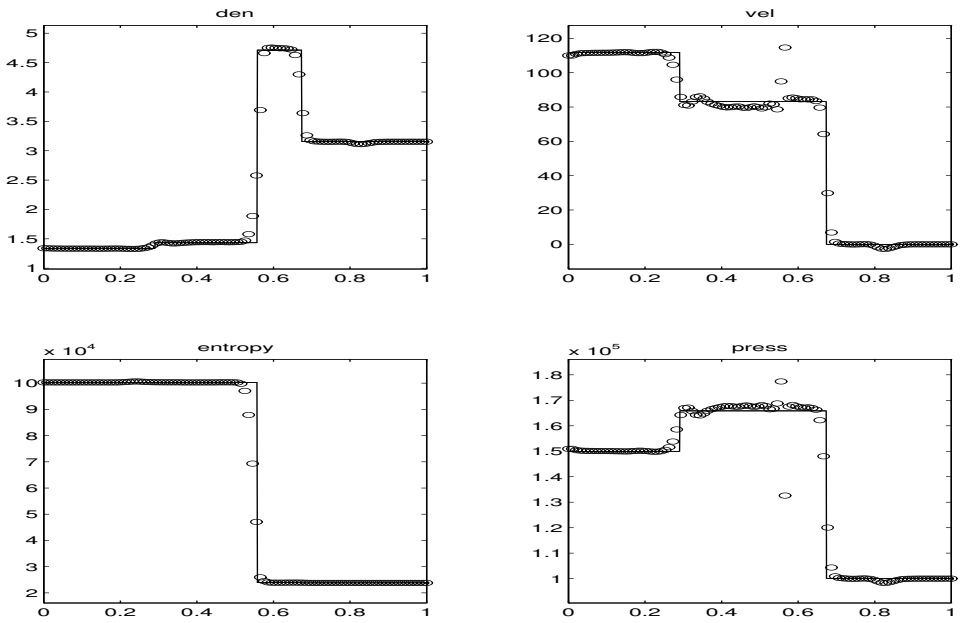

Figure 13: Test C - Standard Scheme - 100 points

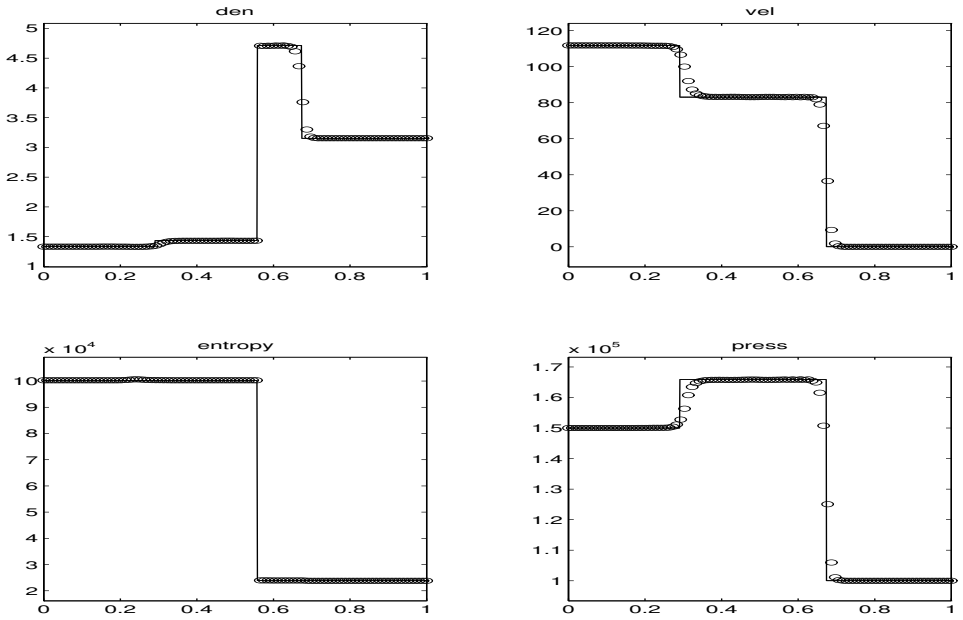

Figure 14: Test C - Ghost Fluid Method - 100 points

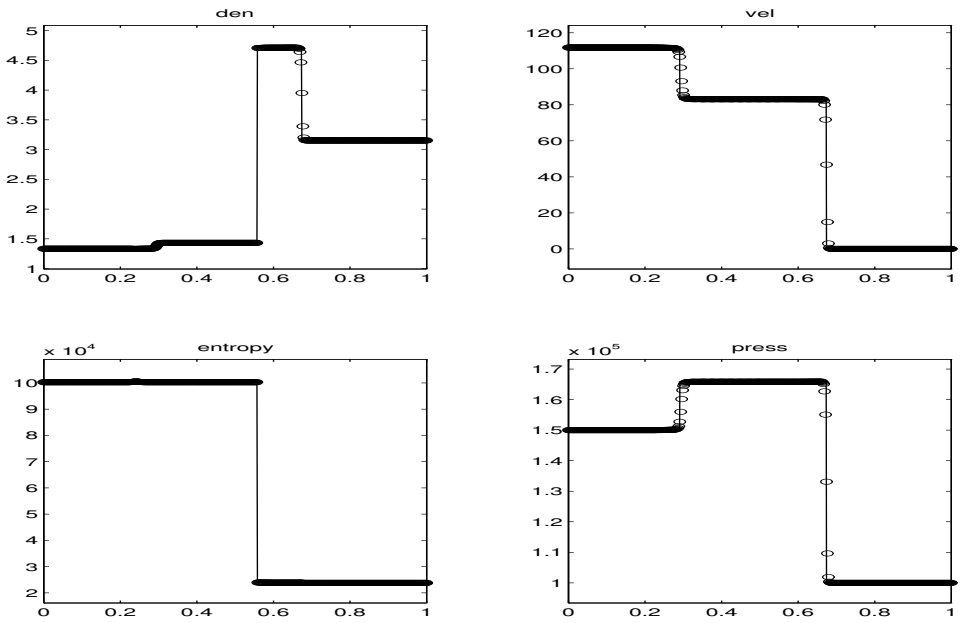

Figure 15: Test C - Ghost Fluid Method - 400 points

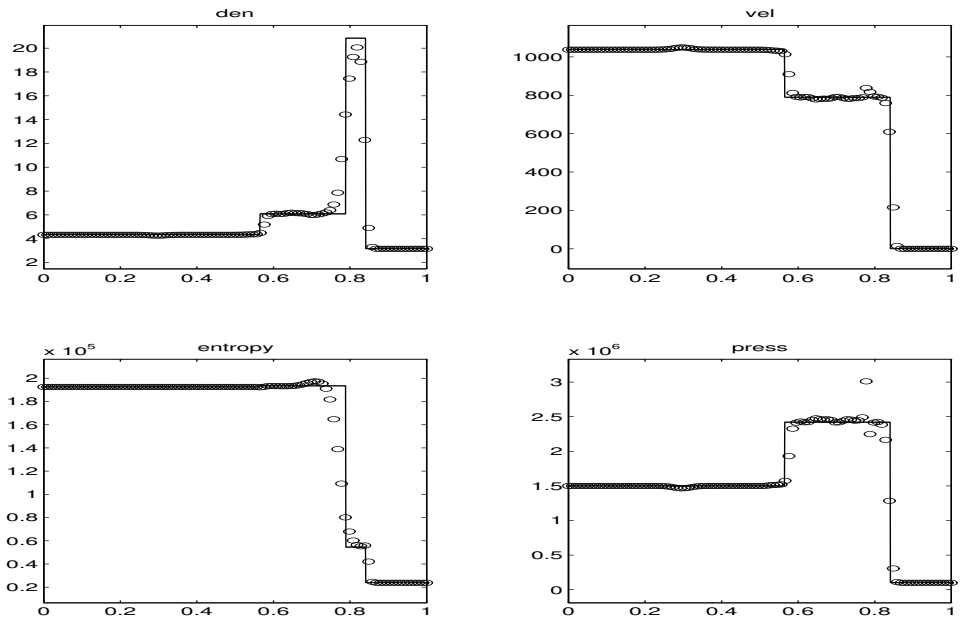

Figure 16: Test D, Case 2 - Standard Scheme - 100 points

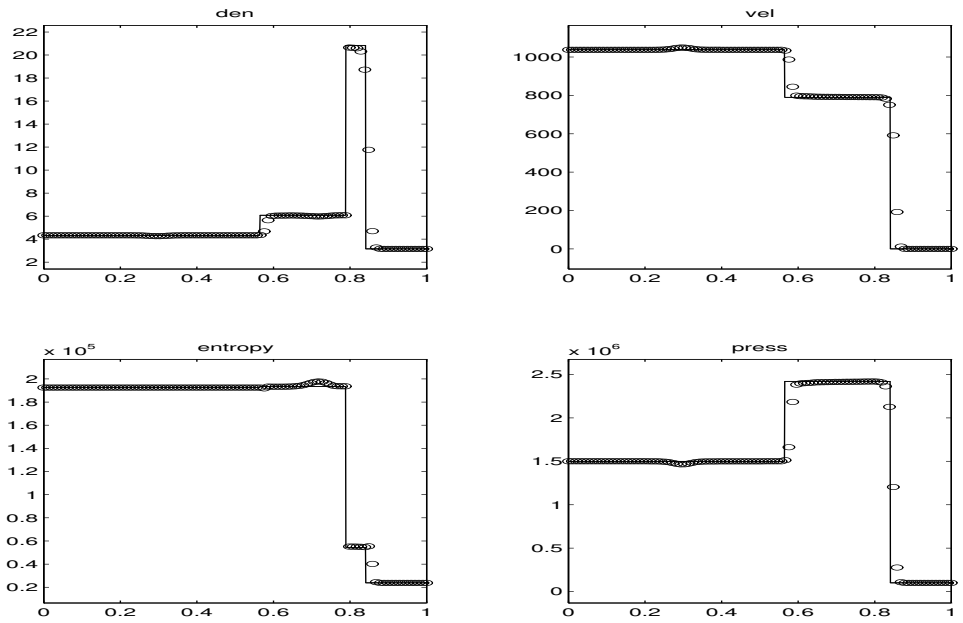

Figure 17: Test D, Case 2 - Ghost Fluid Method - 100 points

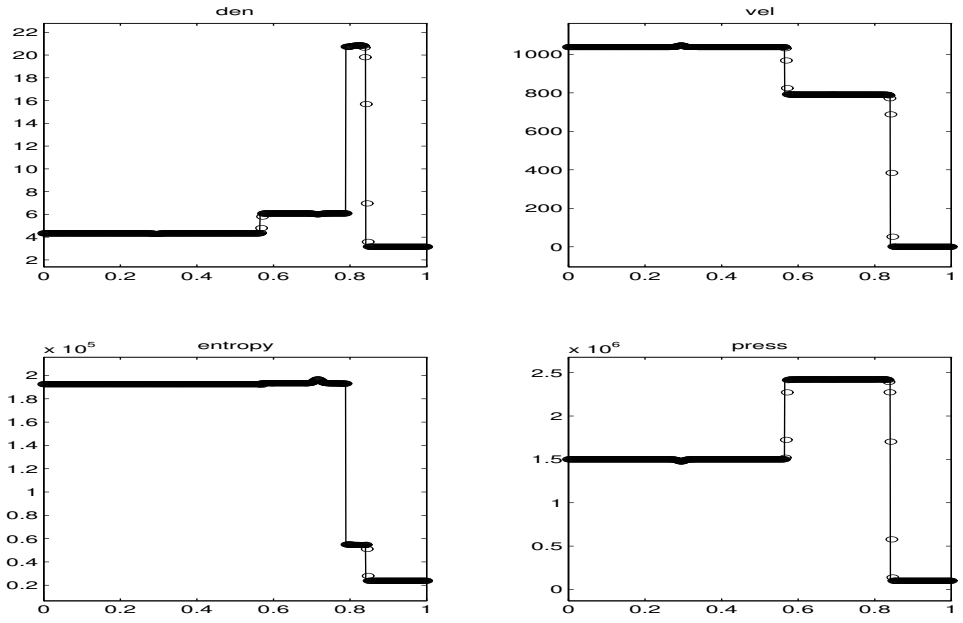

Figure 18: Test D, Case 2 - Ghost Fluid Method - 400 points

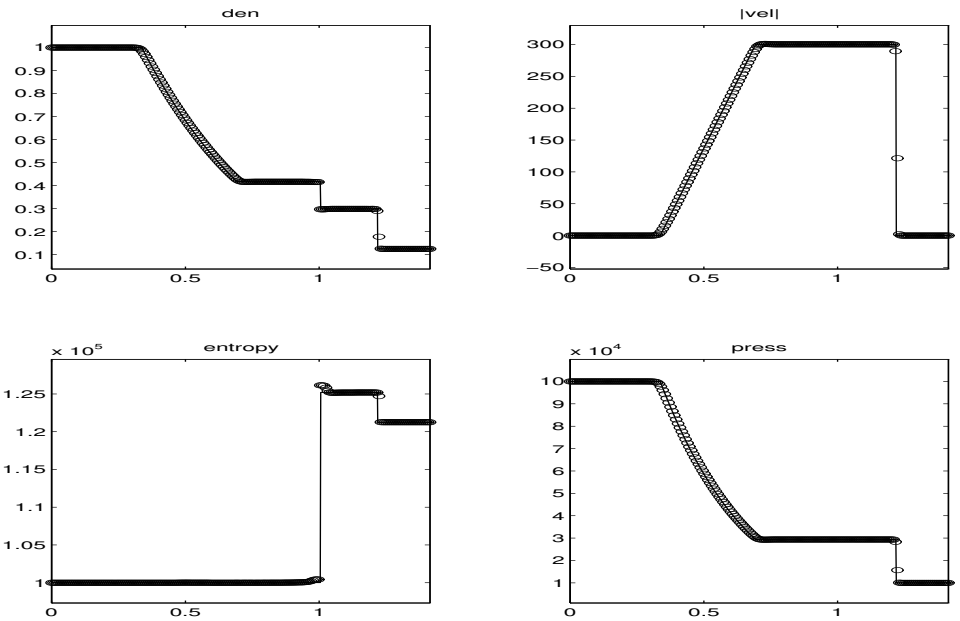

Figure 19: Test A - 2D calculation - diagonal cross-section

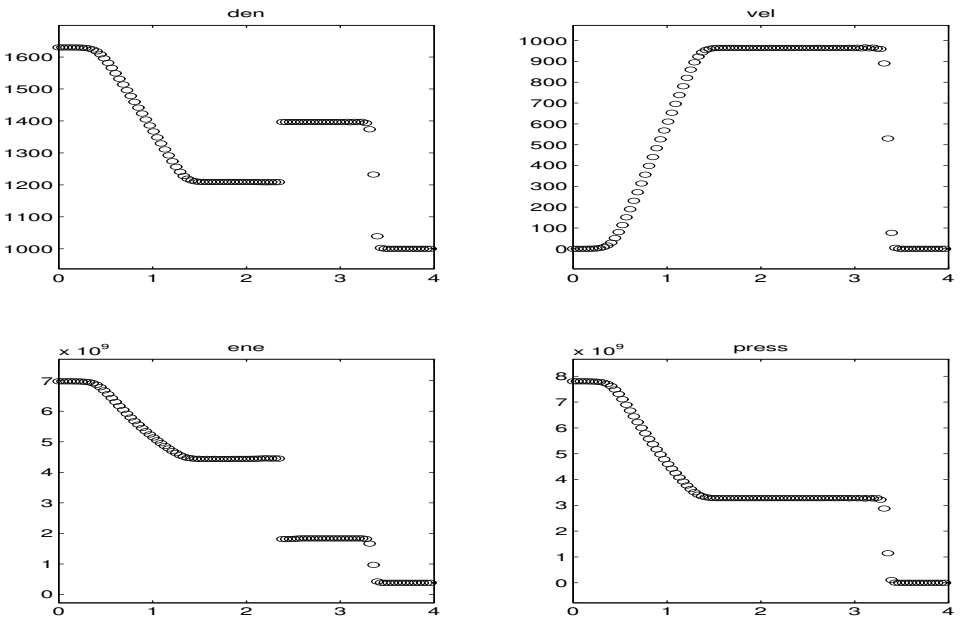

Figure 20: JWL gas on the left & water on the right

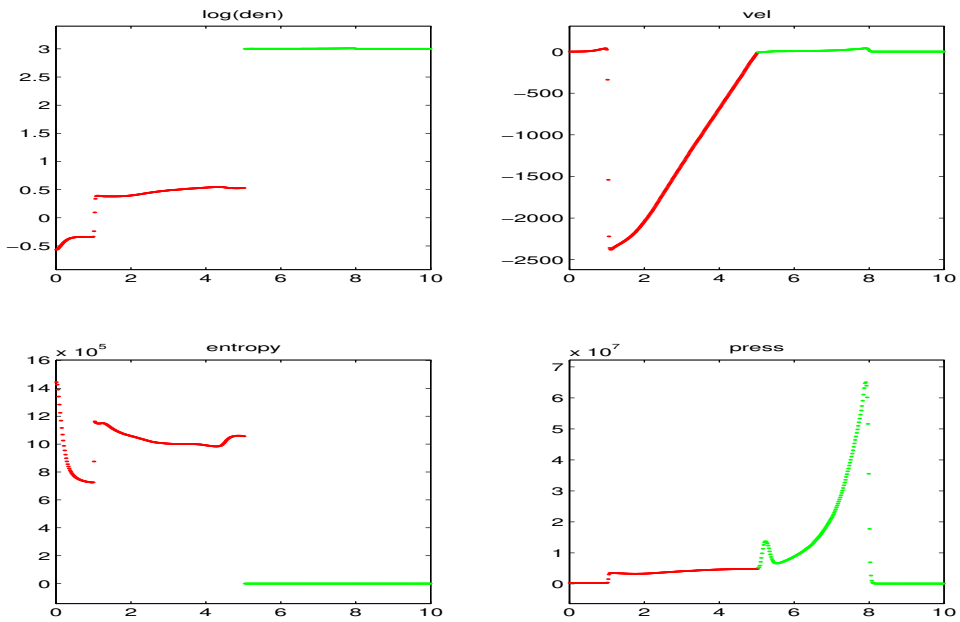

Figure 21: gamma law gas (RED) & water (GREEN)

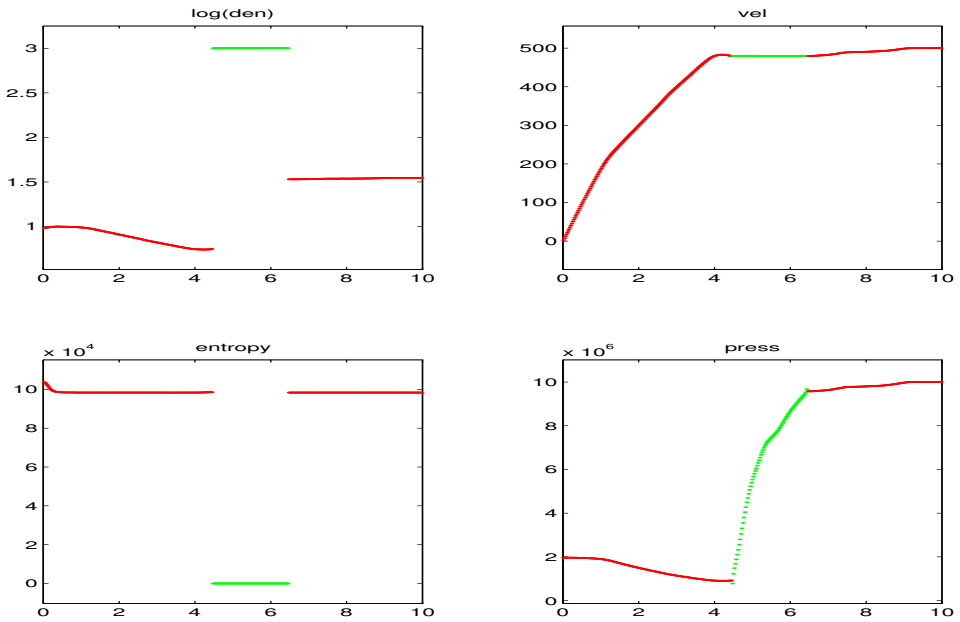

Figure 22: gamma law gas (RED) & water (GREEN)

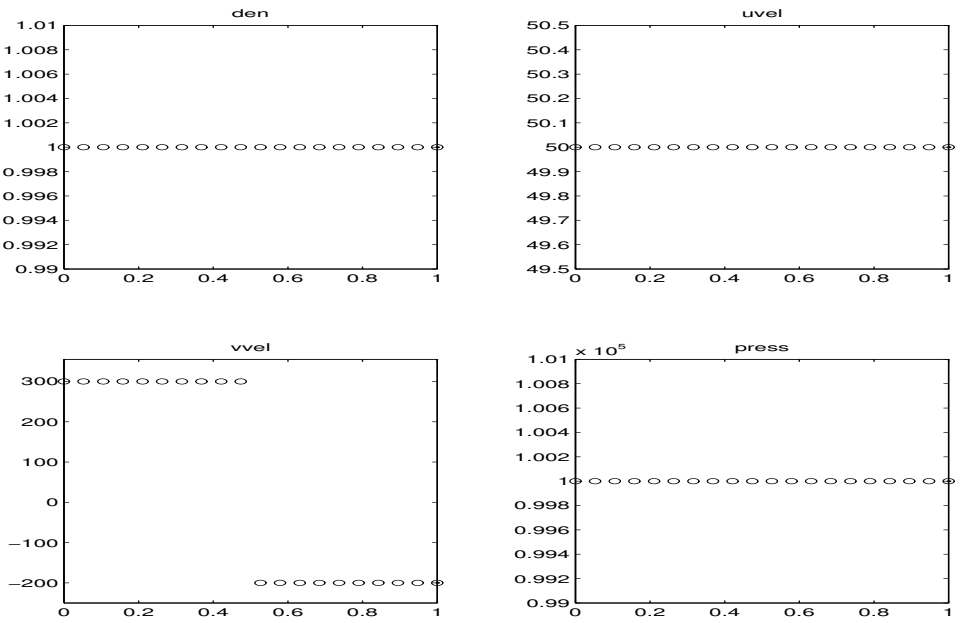

Figure 23: Shear Wave - initial data

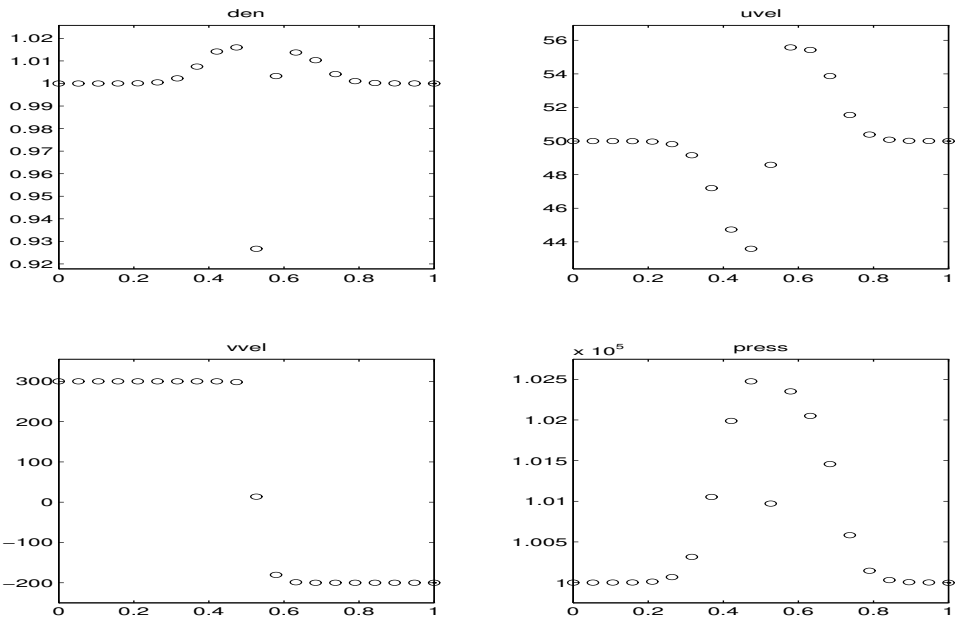

Figure 24: Shear Wave - Standard Scheme

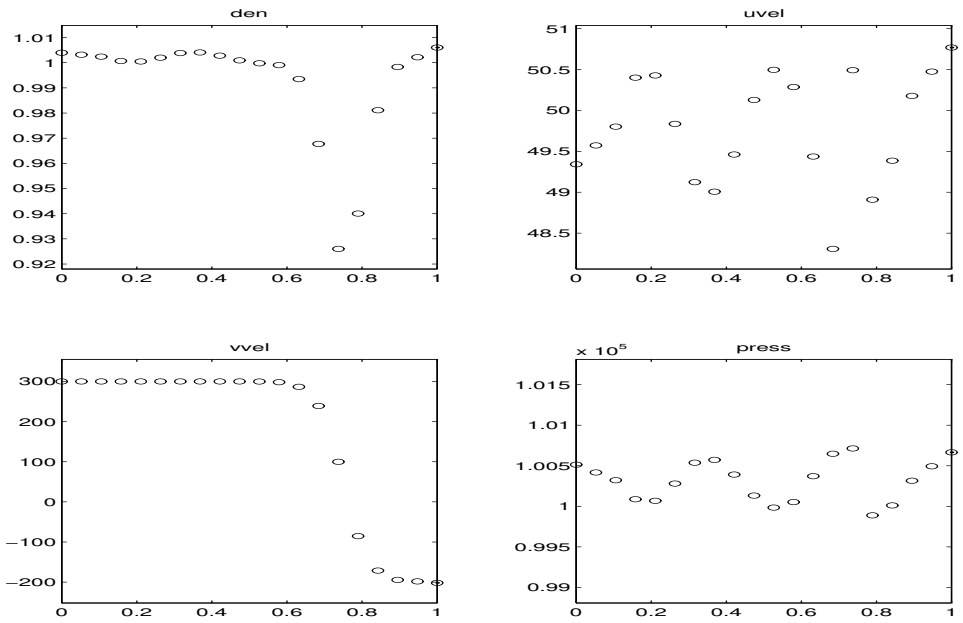

Figure 25: Shear Wave - Standard Scheme

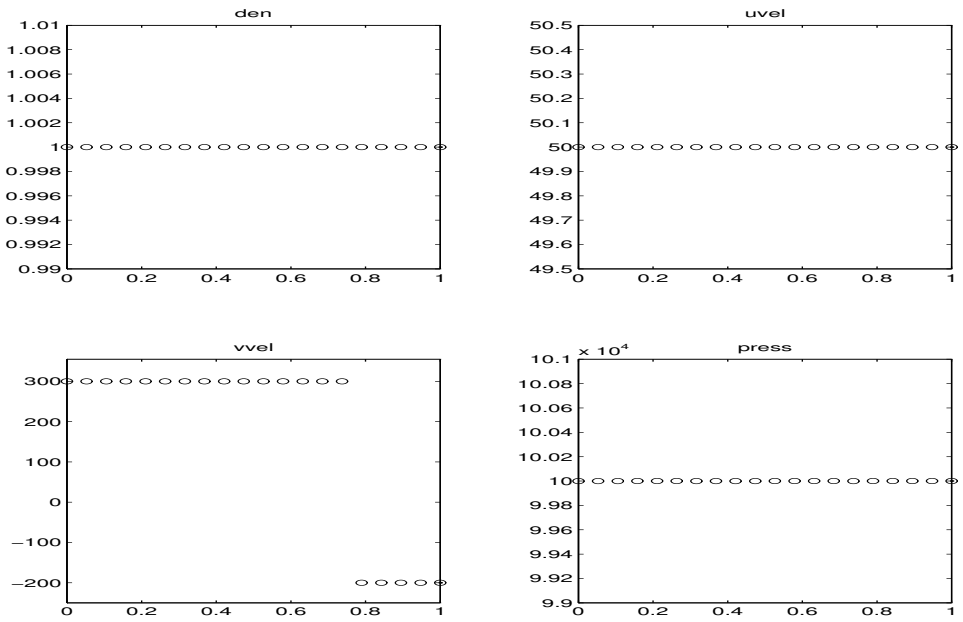

Figure 26: Shear Wave - Ghost Fluid Method

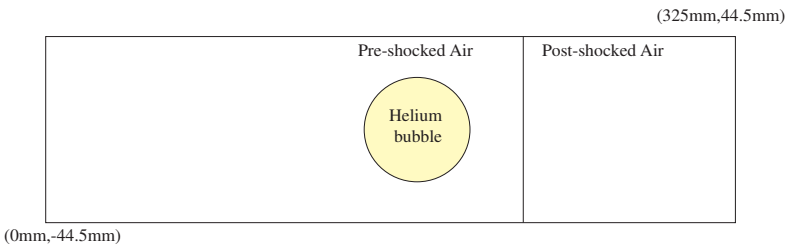

Figure 27: Physical Domain for Example 8

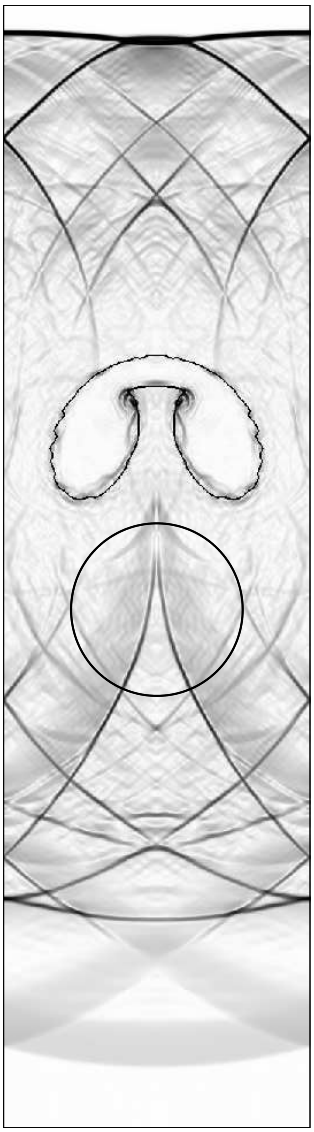

Figure 28: Schlieren image for Example 8 at $t = 427\mu s$ (Rotated 90^o clockwise)

A Level Set Methods

In this section, we present some of the relevant ideas for discretization of the level set advection equation

$$\phi_t + u\phi_x + v\phi_y = 0 \tag{39}$$

and the reinitialization equation

$$\phi_t + S(\phi_o)\left(\sqrt{\phi_x^2 + \phi_y^2} - 1\right) = 0 \tag{40}$$

We also discuss the advection equation for ghost cells population and the isobaric fix

$$I_t \pm n_x I_x \pm n_y I_y = 0 \tag{41}$$

where we have used $\vec{N} =< n_x, n_y >$.

A.1 Hamilton Jacobi Discretization

Following [25, 26], we need to find a left sided and right sided discretization for ϕ_x which we call ϕ_x^- and ϕ_x^+. The same procedure is applied to ϕ_y in the obvious fashion.

A.1.1 3rd Order ENO

We proceed along the lines of [29]. We will use polynomial interpolation to find ϕ and then differentiate to get ϕ_x.

The zeroth order divided differences, D_i^0, and all higher order *even* divided differences of ϕ exist at the grid points and will have the subscript i. The first order divided differences, $D_{i+\frac{1}{2}}^1$, and all higher order *odd* divided differences of ϕ exist at the cell walls and will have the subscript $i \pm \frac{1}{2}$.

Consider a specific grid point i_0. To find ϕ_x^-, set $k = i_0 - 1$. To find ϕ_x^+, set $k = i_0$.

Define

$$Q_1(x) = (D_{k+\frac{1}{2}}^1 \phi)(x - x_{i_0}). \tag{42}$$

If $|D_k^2\phi| \leq |D_{k+1}^2\phi|$, then $c = D_k^2\phi$ and $k^\star = k-1$. Otherwise, $c = D_{k+1}^2\phi$ and $k^\star = k$. Define

$$Q_2(x) = c(x - x_k)(x - x_{k+1}). \tag{43}$$

If $|D^3_{k^\star - \frac{1}{2}}\phi| \leq |D^3_{k^\star + \frac{1}{2}}\phi|$, then $c^\star = D^3_{k^\star - \frac{1}{2}}\phi$. Otherwise, $c^\star = D^3_{k^\star + \frac{1}{2}}\phi$. Define

$$Q_3(x) = c^\star(x - x_{k^\star})(x - x_{k^\star+1})(x - x_{k^\star+2}). \tag{44}$$

And then $(\phi_x^\pm)_{i_0}$ is

$$D^1_{k+\frac{1}{2}}\phi + c\,(2(i_0 - k) - 1)\,\triangle x + c^\star\,\left(3(i_0 - k^\star)^2 - 6(i_0 - k^\star) + 2\right)(\triangle x)^2. \tag{45}$$

A.1.2 5th Order WENO

We proceed along the lines of [16] and [15]. Consider a specific grid point i_0.
To find ϕ_x^-, set

$$v_1 = \frac{\phi_{i_0-2} - \phi_{i_0-3}}{\triangle x}, \qquad v_2 = \frac{\phi_{i_0-1} - \phi_{i_0-2}}{\triangle x} \tag{46}$$

$$v_3 = \frac{\phi_{i_0} - \phi_{i_0-1}}{\triangle x}, \qquad v_4 = \frac{\phi_{i_0+1} - \phi_{i_0}}{\triangle x} \tag{47}$$

$$v_5 = \frac{\phi_{i_0+2} - \phi_{i_0+1}}{\triangle x} \tag{48}$$

and to find ϕ_x^+, set

$$v_1 = \frac{\phi_{i_0+3} - \phi_{i_0+2}}{\triangle x}, \qquad v_2 = \frac{\phi_{i_0+2} - \phi_{i_0+1}}{\triangle x} \tag{49}$$

$$v_3 = \frac{\phi_{i_0+1} - \phi_{i_0}}{\triangle x}, \qquad v_4 = \frac{\phi_{i_0} - \phi_{i_0-1}}{\triangle x} \tag{50}$$

$$v_5 = \frac{\phi_{i_0-1} - \phi_{i_0-2}}{\triangle x} \tag{51}$$

Next we define the smoothness

$$S_1 = \frac{13}{12}(v_1 - 2v_2 + v_3)^2 + \frac{1}{4}(v_1 - 4v_2 + 3v_3)^2 \tag{52}$$

$$S_2 = \frac{13}{12}(v_2 - 2v_3 + v_4)^2 + \frac{1}{4}(v_2 - v_4)^2 \tag{53}$$

$$S_3 = \frac{13}{12}(v_3 - 2v_4 + v_5)^2 + \frac{1}{4}(3v_3 - 4v_4 + v_5)^2 \tag{54}$$

and the weights

$$a_1 = \frac{1}{10}\frac{1}{(\epsilon + S_1)^2}, \qquad w_1 = \frac{a_1}{a_1 + a_2 + a_3} \tag{55}$$

$$a_2 = \frac{6}{10}\frac{1}{(\epsilon + S_2)^2}, \qquad w_2 = \frac{a_2}{a_1 + a_2 + a_3} \tag{56}$$

$$a_3 = \frac{3}{10}\frac{1}{(\epsilon + S_3)^2}, \qquad w_3 = \frac{a_3}{a_1 + a_2 + a_3} \tag{57}$$

to finally get $(\phi_x^{\pm})_{i_0} =$

$$w_1(\frac{v_1}{3} - \frac{7v_2}{6} + \frac{11v_3}{6}) + w_2(\frac{-v_2}{6} + \frac{5v_3}{6} + \frac{v_4}{3}) + w_3(\frac{v_3}{3} + \frac{5v_4}{6} - \frac{v_5}{6}) \tag{58}$$

Note that we use $\epsilon = 10^{-6}$.

A.2 Convection

In order to solve,

$$\phi_t + u\phi_x + v\phi_y = 0 \tag{59}$$

we look at the velocities. If $u_{i_0} > 0$, then we use ϕ_x^-. If $u_{i_0} < 0$, then we use ϕ_x^+. If $u_{i_0} = 0$, then we do not need to choose either. The same considerations apply to v and ϕ_y.

A.3 Reinitialization

In order to solve,

$$\phi_t + S(\phi_o)\left(\sqrt{\phi_x^2 + \phi_y^2} - 1\right) = 0 \tag{60}$$

we can rewrite it as

$$\phi_t + \left(\frac{S(\phi_o)\phi_x}{\sqrt{\phi_x^2 + \phi_y^2}}\right)\phi_x + \left(\frac{S(\phi_o)\phi_y}{\sqrt{\phi_x^2 + \phi_y^2}}\right)\phi_y = S(\phi_o) \tag{61}$$

and consider $S(\phi_o)\phi_x$ and $S(\phi_o)\phi_y$ evaluated at i_0 to determine the upwind directions. [30]

We use a modification of Godunov's method [26]. If $S(\phi_o)\phi_x^+ \geq 0$ and $S(\phi_o)\phi_x^- \geq 0$, then we use ϕ_x^-. If $S(\phi_o)\phi_x^+ \leq 0$ and $S(\phi_o)\phi_x^- \leq 0$, then we use ϕ_x^+. If $S(\phi_o)\phi_x^+ > 0$ and $S(\phi_o)\phi_x^- < 0$, then we use $\phi_x = 0$. If $S(\phi_o)\phi_x^+ < 0$ and $S(\phi_o)\phi_x^- > 0$, we define

$$s = \frac{S(\phi_o)(|\phi_x^+| - |\phi_x^-|)}{\phi_x^+ - \phi_x^-} \tag{62}$$

and if $s > 0$, then we use ϕ_x^-. Otherwise we use ϕ_x^+.

The same procedure is repeated for $S(\phi_o)\phi_y$ and the appropriate values for ϕ_x and ϕ_y are plugged into equation 60.

Note that we smear out the sign function and define

$$S(\phi_o) = \frac{\phi}{\sqrt{\phi^2 + (\triangle x)^2}} \tag{63}$$

instead of the exact sign function.

We also use a limiter in the time evolution of the distance function to keep the interface from crossing grid points.

A.4 Ghost Cells

In order to solve,

$$I_t + n_x I_x + n_y I_y = 0 \tag{64}$$

we use a first order ENO approximation to I_x^+ and I_x^- as outlined above for ϕ. Note that we use first order since theoretically this equation is solved to steady state.

We will also need to evaluate the unit normal

$$\vec{N} =< n_x, n_y >= \left\langle \frac{\phi_x}{\sqrt{\phi_x^2 + \phi_y^2}}, \frac{\phi_y}{\sqrt{\phi_x^2 + \phi_y^2}} \right\rangle \tag{65}$$

at i_0, e.g. using central differencing. It is usually helpful to use a temporary variable, $\hat{\phi}$, when computing the normal. First copy ϕ into $\hat{\phi}$, reinitialize $\hat{\phi}$ to produce an approximate distance function, then compute the normals using $\hat{\phi}$. In this way one can get improved values for the normal without losing accuracy in the original level set function, ϕ. Note that caution should be used to avoid division by zero when $\phi_x = \phi_y = 0$.

If $n_x > 0$, then we use I_x^-. If $n_x < 0$, then we use I_x^+. If $n_x = 0$, then we do not need to choose either. The same considerations apply to the $n_y I_y$ term.

In order to solve,

$$I_t - n_x I_x - n_y I_y = 0 \tag{66}$$

we use $-n_x$ instead of n_x and $-n_y$ instead of n_y in the procedure above.

We use $+\vec{N}$ to update the ghost cells with $\phi > 0$ and $-\vec{N}$ to update the ghost cells with $\phi \leq 0$. where we have chosen the convention that $\phi = 0$ belongs to the fluid with $\phi < 0$. To apply the isobaric fix, we allow a band of the real fluids cells near the interface to be populated along with the cells on the other side of the interface. In this case, we use $+\vec{N}$ to update the cells with $\phi > -\epsilon$ and $-\vec{N}$ to update the cells with $\phi < +\epsilon$ where ϵ determines the thickness of the band. For example, choose $\epsilon = 1.5\triangle x$.

A.5 Time Evolution

The advection equation for the level set function is updated together with the Euler equations.

The reinitialization equation is usually solved in fictitious time after each *fully complete time step* for the Euler equations. For example, set $\triangle\tau = \frac{\triangle x}{2}$ and take 10 τ-steps with a 3rd order TVD Runge Kutta method to reinitialize about 5 cells on each side of the interface to be approximately a distance function.

The advection equation for the population of ghost cells must be done after each *substep* of the time discretization for the Euler equations, in contrast to the reinitialization. For example, the ghost cells must be populated

after each Euler substep of a Runge Kutta method, whereas the reinitialization is done after each full Runge Kutta step. This update is also done in fictitious time. For example, set $\triangle\tau = \frac{\triangle x}{2}$ and take about 20 τ-steps with a 3rd order TVD Runge Kutta method to populate a small band of ghost cells. We caution the reader that numerical dissipation could affect this ghost cell population and that they may need more than 20 steps on occasion.

A.6 Boundaries

The following boundary condition keeps the characteristics flowing outward for the level set function. After updating the interior of the domain, we update the boundary points with

$$\phi_B = \phi_{B-1} + S(\phi_{B-1})|\phi_{B-1} - \phi_{B-2}| \tag{67}$$

where ϕ_B lies on the boundary and ϕ_{B-1} and ϕ_{B-2} are the adjacent points in a given coordinate direction.

References

[1] Atkins, P., *Physical Chemistry*, 5th edition, Freeman, 1994.

[2] Benson, D., *Computational Methods in Lagrangian and Eulerian hydrocodes*, Computer Methods in Applied Mechanics and Engineering, 99 (1992), pp. 235-394.

[3] Cocchi, J.-P., Saurel, S., *A Riemann Problem Based Method for the Resolution of Compressible Multimaterial Flows*, Journal of Computational Physics, vol. 137, (1997) pp. 265-298.

[4] Davis, S., *An interface tracking method for hyperbolic systems of conservation laws*, Applied Numerical Mathematics, 10 (1992) 447-472.

[5] Fedkiw, R., Liu, X.-D., Osher, S., *A General Technique for Eliminating Spurious Oscillations in Conservative Schemes for Multiphase and Multispecies Euler Equations*, UCLA CAM Report 97-27, June 1997.

[6] Fedkiw, R., Marquina, A., and Merriman, B., *An Isobaric Fix for the Overheating Problem in Multimaterial Compressible Flows*, UCLA CAM Report 98-5, February 1998, J. Computational Physics (to appear).

[7] Fedkiw, R., Merriman, B., Donat, R., and Osher, S., *The Penultimate Scheme for Systems of Conservation Laws: Finite Difference ENO with Marquina's Flux Splitting*, UCLA CAM Report 96-18, July 1996, http://www.math.ucla.edu/applied/cam/.

[8] Fedkiw, R., Merriman, B., and Osher, S., *Efficient characteristic projection in upwind difference schemes for hyperbolic systems (The Complementary Projection Method)*, J. Computational Physics, vol. 141, 22-36 (1998).

[9] Fedkiw, R., Merriman, B., and Osher, S., *High accuracy numerical methods for thermally perfect gas flows with chemistry*, J. Computational Physics 132, 175-190 (1997).

[10] Fedkiw, R., Merriman, B., and Osher, S., *Numerical methods for a one-dimensional interface separating compressible and incompressible flows*,

Barriers and Challenges in Computational Fluid Dynamics, pp 155-194,