NUMERICAL METHODS FOR A ONE-DIMENSIONAL INTERFACE SEPARATING COMPRESSIBLE AND INCOMPRESSIBLE FLOWS

R. FEDKIW, B. MERRIMAN AND S. OSHER [0]
UCLA, Department of Mathematics,
405 Hilgard Ave., LA, CA 90095

Abstract.

We develop 1D numerical methods for treating an interface separating a liquid drop and a high speed gas flow. The droplet is an incompressible Navier-Stokes fluid. The gas is a compressible, multi-species, chemically reactive Navier-Stokes fluid (Fedkiw *et al.*, 1996; Fedkiw, 1996). The interface is followed with a marker particle, although the level set method will be used for the eventual 2D extension (Sussman, 1995). Away from the interface, we solve the equations with TVD Runge Kutta schemes in time and conservative finite difference ENO schemes in space (Shu and Osher, 1988). Near the interface, we cannot apply this discretization, since the equations differ in both number and type across the interface. Instead we use the interface location for domain decomposition, and apply a moving control volume formulation nearby. This is done in a conservative framework, compatible with the outer finite difference scheme. Full details are given for a simple forward Euler time stepping scheme, and this has direct, although algorithmically complicated, extensions to 2nd and 3rd order Runge Kutta methods. Future work will focus on the extension to 2D, and simplifications of the higher order time stepping algorithms.

[0]Research supported in part by ARPA URI-ONR-N00014-92-J-1890, NSF #DMS 94-04942, and ARO DAAH04-95-1-0155.

1. Introduction

The level set method has been used to model the interface between different fluids. In this procedure, each numerical flux function is constructed by staying on the appropriate side of the level set. These fluxes are then differenced to update the conserved variables. This method works well for most of the numerical domain. However, it does not work as well for points which are adjacent to the level set. A point which is adjacent to the level set will be updated using a flux from each side of it. Using these "straddled" fluxes can lead to problems.

If the media on opposite sides of the interface are very similar, then one is able to update a point adjacent to the level set by using "straddled" fluxes. In (Mulder *et al.*, 1991) the authors modeled two distinct gas phases (e.g. helium and air) and had little trouble with "straddled" fluxes. This is because the two gas phases are fairly similar. In (Sussman, 1995), dissimilar fluids are treated, such as water and air. In order to use "straddled" fluxes, conditions are imposed to make the fluids similar: both are treated as incompressible fluids, and their physical properties are averaged together over a few points (around 2 or 3) near the interface in order to smooth out the discrepancies.

There are times when the fluids on opposite sides of the interface are quite different and we are not free to simply average their properties. Even worse, the equations on opposite sides of the level set may differ in number and type. In these cases it does not make mathematical sense to use "straddled" fluxes; a new method must be employed. We construct a flux located on the moving interface, using sub-grid information. Then this *interface flux* can be utilized in a conservative way to update the points adjacent to the interface, on one or both sides as required by the equations posed on each side. In this way the different fluids do not incorrectly interact with each other through interior flux points. They interact with each other through the *interface flux* using the appropriate mathematical model designed for the interface.

We will study the 1D case of a liquid and gas, such as water and air. The liquid is an incompressible Navier-Stokes fluid. The gas is a compressible, multi-species, chemically reactive Navier-Stokes fluid (Fedkiw *et al.*, 1996; Fedkiw, 1996). The interface is followed with a marker particle instead of a level set. In 1D there is little advantage to using a level set. In 2D we will use the level set method, since it is the most efficient technique (Mulder *et al.*, 1991; Sussman, 1995). Away from the interface, we solve the equations with TVD Runge Kutta schemes in time and ENO schemes in space (Shu and Osher, 1988). The ENO schemes are implemented using Marquina's Jacobian method (Fedkiw *et al.*, 1996). Near the interface, we cannot employ standard methods, since the equations differ in both number and type. Instead we use the interface location for domain decomposition. A moving flux function is constructed for the interface in such a way that it is valid for both domains. We use this moving flux function in a conservative framework to update the conserved variables together with forward Euler time stepping. With some added algorithmic complexity, we can also directly extend this time stepping technique to methods based on a composite of Euler time steps, such as 2nd and 3rd order TVD Runge Kutta methods (Shu and Osher, 1988).

2. Equations

2.1. GAS PHASE MODEL EQUATIONS

The 1D Navier-Stokes equations for multi-species flow with chemical reactions are

$$\vec{U}_t + [\vec{F}(\vec{U})]_x = [\vec{F}_v(\vec{U})]_x + \vec{S} \tag{1}$$

$$E = -p + \frac{\rho u^2}{2} + \rho h \tag{2}$$

where

$$\vec{U} = \begin{pmatrix} \rho \\ \rho u \\ E \\ \rho Y_1 \\ \vdots \\ \rho Y_{N-1} \end{pmatrix}, \quad \vec{F}(\vec{U}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \\ \rho u Y_1 \\ \vdots \\ \rho u Y_{N-1} \end{pmatrix} \tag{3}$$

$$F_v(U) = \begin{pmatrix} 0 \\ \tau_{11} \\ u\tau_{11} + Q_1 \\ \rho D_{1,m}(Y_1)_x \\ \vdots \\ \rho D_{N-1,m}(Y_{N-1})_x \end{pmatrix}, \quad \vec{S} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \dot{\omega}_1 \\ \vdots \\ \dot{\omega}_{N-1} \end{pmatrix} \tag{4}$$

$$\tau_{11} = \frac{4}{3}\mu u_x, \quad Q_1 = kT_x + \rho \textstyle\sum_{i=1}^N h_i D_{i,m}(Y_i)_x \tag{5}$$

$$\dot{\omega}_i = \dot{\omega}_i(T, p, Y_1, \ldots, Y_{N-1}) \tag{6}$$

where t is time, x is the spatial dimension, ρ is the density, u is the velocity, E is the energy per unit volume, h is enthalpy per unit mass, p is the pressure, N is the number of species being considered, Y_i is the mass fraction of species i, μ is the mixture viscosity, k is the mixture thermal conductivity, $D_{i,m}$ is the mass diffusivity of species i into the mixture, and $\dot{\omega}_i$ is the mass production rate of species i. (Fedkiw *et al.*, 1996) Note that

$$Y_N = 1 - \sum_{i=1}^{N-1} Y_i \tag{7}$$

defines the last mass fraction.

2.1.1. *Energy and Enthalpy*

We write the enthalpy per unit mass as

$$h = e + \frac{p}{\rho} = \sum_{i=1}^{N} Y_i e_i + \frac{\sum_{i=1}^{N} p_i}{\rho} = \sum_{i=1}^{N} Y_i \left(e_i + \frac{p_i}{\rho Y_i} \right) = \sum_{i=1}^{N} Y_i h_i \tag{8}$$

where e_i, p_i, and h_i are the internal energy, partial pressure, and the enthalpy per unit mass of the ith gas respectively. Using the first equality in 8 gives an alternate form for equation 2

$$E = \rho e + \frac{\rho u^2}{2} \tag{9}$$

where e is the internal energy per unit mass.

In a *perfect gas*, the internal energy, enthalpy, and specific heats are functions of the temperature only. In this case we can write

$$h_i = h_i(T) \qquad e_i = e_i(T) \tag{10}$$

$$c_{p_i} = c_{p_i}(T) \qquad c_{vi} = c_{vi}(T) \tag{11}$$

$$dh_i(T) = c_{p_i}(T)dT \qquad de_i(T) = c_{vi}(T)dT \tag{12}$$

where c_{p_i} is the specific heat at constant pressure of the ith species, and c_{vi} is the specific heat at constant volume of the ith species (Anderson, 1989).

We can integrate the first equation in 12 starting from $T = 298K$ to get

$$h_i(T) = h_i^{298} + \int_{298}^{T} c_{p_i}(s)ds \tag{13}$$

where h_i^{298} is the enthalpy per unit mass at $298K$ for species i. This is also sometimes called the heat of formation at 298K, which is a standard constant that can be found in the JANAF Thermochemical Tables (Stall and Prophet, 1971). To speed up the actual implementation, we construct a table of $h_i(T)$ for each species including every integer temperature between $0K$ and $4800K$. We approximate the integral to desired accuracy, using the CHEMKIN code (Kee *et al.*, 1986) to give us the values of $c_{p_i}(T)$ when needed. This is done once at the beginning of the code. During computation, if we need $h_i(T)$ for a non-integral value of the temperature, we interpolate linearly. (Fedkiw *et al.*, 1996)

A *calorically perfect* gas has c_{p_i} constant, while a *thermally perfect* gas has $c_{p_i}(T)$ a function of the temperature (Anderson, 1989). In regimes where CHEMKIN does not have data for $c_{p_i}(T)$, we usually extrapolate with a calorically perfect assumption.

2.1.2. *Equation of State, Specific Heats, and Gamma*

The equation of state for multi-species flow is

$$p = \rho R T \tag{14}$$

where

$$R = \frac{R_u}{W} \tag{15}$$

where $R_u = 8314$ J/(kmol K) is the universal gas constant, and W is the molecular weight of the mixture given by

$$W = \frac{1}{\sum_{i=1}^{N} \frac{Y_i}{W_i}} \tag{16}$$

where W_i is the molecular weight of the ith species.

Gamma for the mixture is

$$\gamma = \frac{c_p}{c_p - \frac{R_u}{W}} \tag{17}$$

where c_p is the specific heat at constant pressure of the mixture given by

$$c_p = \sum_{i=1}^{N} Y_i c_{p_i} \tag{18}$$

(Fedkiw *et al.*, 1996).

2.1.3. *Temperature*

Since various thermodynamic quantities are functions of temperature, it is necessary to have a means of computing temperature from the primary conserved variables (mass, momentum and energy or enthalpy) evolved during the calculation.

We get an expression for the temperature by combining equation 2 with equation 14 to get

$$T = C_1 + C_2 h(T) \tag{19}$$

where C_1 and C_2 are constants if the conserved variables are fixed. This equation is implicit for temperature.

We rewrite equation 19 as,

$$f(T) = T - C_1 - C_2 h(T) = 0 \tag{20}$$

and note that

$$\frac{df(T)}{dT} = 1 - C_2 \frac{dh(T)}{dT} = 1 - C_2 c_p(T) = 1 - \frac{c_p(T)}{R} = \frac{-1}{\gamma(T) - 1} \tag{21}$$

where $\gamma(T)$ is a function of temperature. Since $\gamma(T)$ is always greater than one, this shows that $f(T)$ is a strictly decreasing function of temperature. We solve equation 20 with Newton-Raphson iteration. (Fedkiw *et al.*, 1996)

2.1.4. *Eigensystem*

The ENO spatial discretization is based on upwind discretization of the characteristic fields, and so requires full characteristic data for the convective part of the system of conservation laws, i.e. the eigensystem of $\vec{F}'(\vec{U})$. See appendix A for details.

2.2. LIQUID PHASE MODEL EQUATIONS

The incompressible liquid phase satisfies the general incompressibility condition $\nabla \cdot u = 0$, which in 1D reduces to the trivial form

$$u_x = 0 \tag{22}$$

with the density also a constant. For water,

$$\rho = 1000 \frac{kg}{m^3} \tag{23}$$

We will also assume the thermal conductivity is a constant, which is reasonable in the liquid phase. For water in the temperature range from range from 298K to 373K, the thermal conductivity is fixed at

$$k = .65 \frac{J}{mKsec} \tag{24}$$

instead of a varying function of temperature.

Using these simplifications, the 1D Navier-Stokes equations for a non-reacting, incompressible flow reduce to

$$\vec{U}_t + [\vec{F}(\vec{U})]_x = [\vec{F}_v(\vec{U})]_x \tag{25}$$

where

$$\vec{U} = \begin{pmatrix} \rho u \\ E \end{pmatrix}, \quad \vec{F}(\vec{U}) = \begin{pmatrix} p \\ (E+p)u \end{pmatrix}, \quad F_v(U) = \begin{pmatrix} 0 \\ kT_x \end{pmatrix} \tag{26}$$

where

$$E = \rho e + \frac{\rho u^2}{2} \tag{27}$$

which is in the form of equation 9. We use this form rather than the enthalpy form to avoid use of the pressure, since there is no local equation of state available for pressure anyway.

2.2.1. *Comments on the Equations*

The first equation in equation 25 can be written as

$$u_t = \frac{-p_x}{\rho} \tag{28}$$

and by taking an x-derivative of both sides of this equation we get

$$p_{xx} = 0 \tag{29}$$

implying that the pressure profile is linear.

Substituting equation 27 into the second equation in equation 25 yields

$$\rho e_t + \rho u u_t + u \rho e_x + u p_x = k T_{xx} \tag{30}$$

which can be simplified with the use of equation 28 to get

$$\rho e_t + u \rho e_x = k T_{xx} \tag{31}$$

or in conservation form as

$$(\rho e)_t + (u \rho e)_x = (k T_x)_x \tag{32}$$

for the convection and diffusion of internal energy per unit volume. It is interesting to note that equation 28 determines both the momentum and the kinetic energy. Thus, equation 28 cancels out the kinetic energy in equation 30 leaving only a conservation equation for internal energy, equation 32.

2.2.2. *Temperature*

As for the gas phase, we need to compute the liquid temperature from the primary variables (momentum and energy). However, the incompressible liquid does not have a local equation of state such as the gas law 14. Thus we cannot follow the temperature calculation used for the gas phase. Here we derive an appropriate relation for the temperature of the liquid phase.

Consider equation 8 for the vapor phase of the liquid,

$$h = e + \frac{p}{\rho} \tag{33}$$

The vapor phase is a gas and thus satisfies the gas law, 14. Using this yields

$$h = e + \left(\frac{R_u}{W}\right) T \tag{34}$$

where W is the molecular weight of the liquid molecules. Thus the internal energy of the vapor phase is

$$e(T) = h(T) - \left(\frac{R_u}{W}\right) T \tag{35}$$

and we can write the internal energy of liquid phase as that of the vapor minus the heat of vaporization,

$$e(T) = h(T) - \left(\frac{R_u}{W}\right) T - H_{vap}(T) \tag{36}$$

where $H_{vap}(T)$ is the latent heat of vaporization of the liquid as a function of temperature. This quantity is tabulated for water and many other liquids.

To speed up the actual implementation, we construct a table of $e(T)$ for the liquid, and handle it in the same way as the enthalpy tables constructed for each species. In order to construct this table consistent with the gas, we first construct a table of enthalpy $h(T)$ for liquid vapor. Then we construct as much of a table as possible for the latent heat of vaporization $H_{vap}(T)$ (this obviously isn't tabulated for very high or low values). Then we can use equation 36 to construct a partial table of internal energies $e(T)$ for the liquid. We need to complete the table outside the range of tabulated latent heat of vaporization $H_{vap}(T)$. To do this, we assume that the liquid is calorically perfect outside the temperature range of tabulated $H_{vap}(T)$. We integrate the second equation in 12 twice. First starting from $T = T_{min}$ and then again starting from $T = T_{max}$

$$e(T) = e(T_{min}) + c_v(T_{min})(T - T_{min}) \tag{37}$$

$$e(T) = e(T_{max}) + c_v(T_{max})(T - T_{max}) \tag{38}$$

where T_{min} and T_{max} were the lowest and highest temperature in the already completed portion of the table of $e(T)$. Note that $c_v(T_{min})$ and $c_v(T_{max})$ are two different constants. We than use equation 37 to fill in the low temperature portion of the table, and equation 38 to fill in the high temperature portion of the table.

We arrive at an implicit expression for the temperature from 27, which can be written as

$$f(T) = e(T) + C = 0 \tag{39}$$

where C is a constant if the conserved variables are fixed. Note that

$$\frac{df(T)}{dT} = \frac{de(T)}{dT} = c_v(T) \tag{40}$$

shows that $f(T)$ is a strictly increasing function of temperature. We then solve equation 39 for the temperature via Newton-Raphson iteration.

3. Numerical Methods

We attempt to construct numerical flux functions at the midpoint of every pair of grid nodes. They are constructed using 3rd order conservative finite difference ENO (Shu and Osher, 1988) with Marquina's Jacobian (Fedkiw *et al.*, 1996) on the convection terms along with conservative central differencing on the diffusive terms. These fluxes are then differenced to update the conserved variables. This method works well for most of the numerical domain. However, it does not work as well for points which are adjacent to the interface. A point which is adjacent to the interface would be updated using a flux from each side of the interface. Since our equations differ in number and type, it does not make sense to use these "straddled" fluxes. Instead, we construct a flux located on the moving interface. This *interface flux* is constructed directly from the model equations valid at the interface. Then the *interface flux* is used in a conservative way to update the points adjacent to the interface.

3.1. CONSERVATION METHOD

Our information is stored as point values on the grid. For normal nodes—those not adjacent to the interface—we construct ENO numerical flux functions which update the point values to the desired order of accuracy. In a cell which contains the interface, we do not construct a numerical flux function. Instead, we construct a physical flux at the interface.

We wish to use moving physical fluxes to update the points adjacent to the interface. For this purpose, we use the cell average framework. Thus, a grid point which is adjacent to the interface has two stored values. The point value is used to find numerical flux functions for updating the interior domain, away from the interface. The cell average value is used to update the grid point itself.

Consider updating the cell average for a point adjacent to the interface. We do this with two physical fluxes. One of them is obviously the moving physical flux on the interface. The other should be constructed away from the interface, where a numerical flux function already exists. To maintain conservation and compatibility with the outer finite difference ENO scheme, we use the pre-existing numerical flux function as an approximation to the desired physical flux function at this point. Since the numerical flux is only a second order accurate approximation to the physical flux, this degrades the local order of our numerical method. Here a choice must be made between order and conservation. We could get higher order by violating conservation, but we favor preserving discrete conservation to capture unresolved phenomena that may arise from abrupt interface motion.

Since the interface moves at the speed of the fluid which it separates, the CFL condition limits its travel to one grid cell per time step. We will eliminate the case where the interface lies exactly on a grid point. We can always move it off the grid point by some small amount based on the order of the method or on machine precision. Thus, there are two cases to consider:

1. The interface does not cross a grid point during the course of an Euler time step. It ends up in the the same cell in which it started.
2. The interface crosses a grid point during the course of an Euler time step. It appears in an adjacent cell at the end of the time step.

For the first case construction, refer to Case 1 in Figure 1 for illustration. In this case the interface remains between the grid points x_i and x_{i+1}. The pointwise values of the conserved variables here are $\vec{U}_i$ and $\vec{U}_{i+1}$, while the cell average values are $(\vec{U}_i)_{ave}$ and $(\vec{U}_{i+1})_{ave}$. We do not construct a numerical flux function $\vec{F}_{i+\frac{1}{2}}$ in the cell which contains the interface. Instead, we construct an interface flux, $\vec{F}_I$. We still construct numerical fluxes away from the interface, including $\vec{F}_{i-\frac{1}{2}}$ and $\vec{F}_{i+\frac{3}{2}}$. At time n, we define h_i^n and h_{i+1}^n as the sizes (lengths) of the cells over which the cell averages $(\vec{U}_i^n)_{ave}$ and $(\vec{U}_{i+1}^n)_{ave}$ are respectively defined. At time $n+1$, we define h_i^{n+1} and h_{i+1}^{n+1} as the sizes of the cells over which the cell averages $(\vec{U}_i^{n+1})_{ave}$ and $(\vec{U}_{i+1}^{n+1})_{ave}$ are respectively defined.

Since the interface moves at the speed of the fluid, we know that the velocity profile is continuous at the interface, although the derivatives of the velocity are not necessarily continuous. At time n, we can interpolate to find the interface velocity v_I^n. For our specific problem, equation 22 tells us that the velocity profile in the liquid is constant. Thus, v_I^n is identical to the liquid velocity at time n, and the interface moves a distance of $v_I^n dt$ during an Euler time step. So it is possible to find the future location of the interface, before updating the conserved variables. Note that a level set representation of the interface will have the same property in multiple dimensions. This ability is essential for the numerical method presented here.

We write the numerical scheme as

$$(\vec{U}_i^{n+1})_{ave}h_i^{n+1} = (\vec{U}_i^n)_{ave}h_i^n + dt(\vec{F}_{i-\frac{1}{2}} - \vec{F}_I) \tag{41}$$

$$(\vec{U}_{i+1}^{n+1})_{ave}h_{i+1}^{n+1} = (\vec{U}_{i+1}^n)_{ave}h_{i+1}^n + dt(\vec{F}_I - \vec{F}_{i+\frac{3}{2}}) \tag{42}$$

which both can be read as: the new “stuff” equals the old “stuff”, plus the flux in from the right, minus the flux out to the left. We can solve equations 41 and 42 for $(\vec{U}_i^{n+1})_{ave}$ and $(\vec{U}_{i+1}^{n+1})_{ave}$ by dividing by h_i^{n+1} and

h_{i+1}^{n+1} respectively. Remember that h_i^{n+1} and h_{i+1}^{n+1} are known, since we are able to move the interface first. Once we find $(\vec{U}_i^{n+1})_{ave}$ and $(\vec{U}_{i+1}^{n+1})_{ave}$ we still need to find the pointwise values $\vec{U}_i^{n+1}$ and $\vec{U}_{i+1}^{n+1}$.

Consider the second case where the interface moves into an adjacent cell, and refer to Case 2 in Figure 1 for illustration. Since moving to the left and to the right are symmetric, we only consider the case where the interface moves into the cell to the right. It starts between x_i and x_{i+1} and ends up between x_{i+1} and x_{i+2}. The pointwise values of the conserved variables here are $\vec{U}_i$, $\vec{U}_{i+1}$, and $\vec{U}_{i+2}$, while the cell average values are $(\vec{U}_i)_{ave}$, $(\vec{U}_{i+1})_{ave}$, and $(\vec{U}_{i+2})_{ave}$. Note that we do not know $(\vec{U}_{i+2})_{ave}$ and must construct it. Once again, we construct an interface flux, $\vec{F}_I$, to replace the numerical flux $\vec{F}_{i+\frac{1}{2}}$. Also, we still construct fluxes away from the interface, including $\vec{F}_{i-\frac{1}{2}}$, $\vec{F}_{i+\frac{3}{2}}$, and $\vec{F}_{i+\frac{5}{2}}$. At time n, we have h_i^n, h_{i+1}^n, and dx as the size of the cells over which the cell averages $(\vec{U}_i^n)_{ave}$, $(\vec{U}_{i+1}^n)_{ave}$, and $(\vec{U}_{i+2}^n)_{ave}$ are respectively defined. At time $n+1$, we have dx, h_{i+1}^{n+1}, and h_{i+2}^{n+1} as the size of the cells over which the cell averages $(\vec{U}_i^{n+1})_{ave}$, $(\vec{U}_{i+1}^{n+1})_{ave}$, and $(\vec{U}_{i+2}^{n+1})_{ave}$ are respectively defined.

We write the numerical scheme as

$$(\vec{U}_i^{n+1})_{ave}dx + (\vec{U}_{i+1}^{n+1})_{ave}h_{i+1}^{n+1} = (\vec{U}_i^n)_{ave}h_i^n + dt(\vec{F}_{i-\frac{1}{2}} - \vec{F}_I) \tag{43}$$

$$(\vec{U}_{i+2}^{n+1})_{ave}h_{i+2}^{n+1} = (\vec{U}_{i+1}^n)_{ave}h_{i+1}^n + (\vec{U}_{i+2}^n)_{ave}dx + dt(\vec{F}_I - \vec{F}_{i+\frac{5}{2}}) \tag{44}$$

which can both be read as: the new “stuff” equals the old “stuff”, plus the flux in from the right, minus the flux out to the left. In equation 43, we need to find $(\vec{U}_i^{n+1})_{ave}$ and $(\vec{U}_{i+1}^{n+1})_{ave}$ along with their pointwise values $\vec{U}_i^{n+1}$ and $\vec{U}_{i+1}^{n+1}$. In equation 44, we find $(\vec{U}_{i+2}^{n+1})_{ave}$ by dividing by h_{i+2}^{n+1}. After this, we still need to find the pointwise value $\vec{U}_{i+2}^{n+1}$.

A note on the CFL condition. The size of the cells adjacent to the interface will not necessarily be dx, like the cells in the rest of the domain. However, it is easy to see that their size will always be larger than $\frac{dx}{2}$ and smaller than $\frac{3dx}{2}$. Thus, the minimum cell size for use in the CFL conditions $\frac{dx}{2}$, and this determines the largest stable time step.

3.2. FINDING UNKNOWN CELL AVERAGES AND POINT VALUES

In equation 44, we need $(\vec{U}_{i+2}^n)_{ave}$. Since the cell averages are only known for the cells adjacent to the interface, we need to construct $(\vec{U}_{i+2}^n)_{ave}$. We use a 2nd order, piecewise linear construction given by

$$(\vec{U}_{i+2}^n)_{ave} = \frac{\vec{U}_{i+1}^n + 6\vec{U}_{i+2}^n + \vec{U}_{i+3}^n}{8} \tag{45}$$

which is depicted graphically in Figure 2.

In equations 41, 42, and 44 we can solve for $(\vec{U}_i^{n+1})_{ave}$, $(\vec{U}_{i+1}^{n+1})_{ave}$, and $(\vec{U}_{i+2}^{n+1})_{ave}$. From these we have to reconstruct the point values $\vec{U}_i^{n+1}$, $\vec{U}_{i+1}^{n+1}$, and $\vec{U}_{i+2}^{n+1}$ respectively. We only show how to reconstruct the point value $\vec{U}_i^{n+1}$ from the cell average $(\vec{U}_i^{n+1})_{ave}$, since the other two reconstructions are symmetric to this one. We use a 2nd order, linear reconstruction which is consistent with the cell average,

$$\vec{U}_i^{n+1} = \left(\frac{(\vec{U}_i^{n+1})_{ave} - \vec{U}_{i-1}^{n+1}}{\frac{dx}{2} + \frac{h_i^{n+1}}{2}} \right) dx + \vec{U}_{i-1}^{n+1} \tag{46}$$

which is depicted graphically in Figure 2. Since these reconstructions could give non-physical values, we find the pointwise value at the interface using the same reconstruction

$$\vec{U}_I = \left(\frac{(\vec{U}_i^{n+1})_{ave} - \vec{U}_{i-1}^{n+1}}{\frac{dx}{2} + \frac{h_i^{n+1}}{2}} \right) \left(\frac{dx}{2} + h_i^{n+1} \right) + \vec{U}_{i-1}^{n+1} \tag{47}$$

and check to be sure this value is within the physically allowed range (positive densities, etc). If not, then we replace equation 46 with

$$\vec{U}_i^{n+1} = (\vec{U}_i^{n+1})_{ave} \tag{48}$$

which is a 1st order accurate reconstruction.

From equation 43, we need to find $(\vec{U}_i^{n+1})_{ave}$ and $(\vec{U}_{i+1}^{n+1})_{ave}$ along with their pointwise values $\vec{U}_i^{n+1}$ and $\vec{U}_{i+1}^{n+1}$. To do this, we we define the total cell average through

$$(\vec{U}_T)_{ave}(dx + h_{i+1}^{n+1}) = (\vec{U}_i^{n+1})_{ave} dx + (\vec{U}_{i+1}^{n+1})_{ave} h_{i+1}^{n+1} \tag{49}$$

where $(\vec{U}_T)_{ave}$ is the total cell average. We can combine equations 43 and 49 to get

$$(\vec{U}_T)_{ave}(dx + h_{i+1}^{n+1}) = (\vec{U}_i^{n})_{ave} h_i^n + dt(\vec{F}_{i-\frac{1}{2}} - \vec{F}_I) \tag{50}$$

and then solve for $(\vec{U}_T)_{ave}$ by dividing by $dx + h_{i+1}^{n+1}$. We use a 2nd order, linear reconstruction which is consistent with the total cell average,

$$\vec{U}_i^{n+1} = \left(\frac{(\vec{U}_T)_{ave} - \vec{U}_{i-1}^{n+1}}{\frac{dx}{2} + \frac{dx + h_{i+1}^{n+1}}{2}} \right) dx + \vec{U}_{i-1}^{n+1} \tag{51}$$

$$\vec{U}_{i+1}^{n+1} = 2\vec{U}_i^{n+1} - \vec{U}_{i-1}^{n+1} \tag{52}$$

$$(\vec{U}_i^{n+1})_{ave} = \vec{U}_i^{n+1} \tag{53}$$

$$(\vec{U}_{i+1}^{n+1})_{ave} = \frac{(\vec{U}_T)_{ave}(dx + h_{i+1}^{n+1}) - (\vec{U}_i^{n+1})_{ave} dx}{h_{i+1}^{n+1}} \tag{54}$$

which is depicted graphically in Figure 2. Note that equation 54 comes from a rearrangement of equation 49. Since these reconstructions could give non-physical values, we find the pointwise value at the interface using the same reconstruction

$$\vec{U}_I = \left(\frac{(\vec{U}_T)_{ave} - \vec{U}_{i-1}^{n+1}}{\frac{dx}{2} + \frac{dx + h_{i+1}^{n+1}}{2}} \right) \left(\frac{3dx}{2} + h_{i+1}^{n+1} \right) + \vec{U}_{i-1}^{n+1} \tag{55}$$

and check to be sure this value is within the physically allowed range. If not, then we replace equations 51, 52, 53, and 54 with

$$\vec{U}_i^{n+1} = \vec{U}_{i+1}^{n+1} = (\vec{U}_i^{n+1})_{ave} = (\vec{U}_{i+1}^{n+1})_{ave} = (\vec{U}_T)_{ave} \tag{56}$$

which is a 1st order accurate reconstruction.

3.3. INTERFACE FLUX

We have discussed how to implement the *interface flux*, F_I, in the numerical method. Next we will consider its construction. The interface flux will model the physics of the interface. The general physical flux looks like,

$$F_I = \vec{F}(\vec{U}) - \vec{F}_v(\vec{U}) = \begin{pmatrix} \rho u \\ \rho u^2 + p - \tau_{11} \\ (E+p)u - u\tau_{11} - Q_1 \\ \rho u Y_1 - \rho D_{1,m}(Y_1)_x \\ \vdots \\ \rho u Y_{N-1} - \rho D_{N-1,m}(Y_{N-1})_x \end{pmatrix} \tag{57}$$

but this can be greatly simplified.

Since the interface moves with the speed of the fluid, and it separates two fluids, there are no particles crossing the interface. Thus, there is no convection of mass, momentum, or energy across the interface. Likewise, there is no mass diffusion across the interface. So, equation 57 reduces to

$$F_I = \begin{pmatrix} 0 \\ p - \tau_{11} \\ u(p - \tau_{11}) - Q_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ p - \frac{4}{3}\mu u_x \\ u(p - \frac{4}{3}\mu u_x) - kT_x \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{58}$$

where $p - \frac{4}{3}\mu u_x$ is the net force on the interface. This force changes both the momentum and the kinetic energy. Also, $-kT_x$ is the diffusive heat flux across the interface.

Since we have a conservative scheme, the interface flux must have some consistency with both sides of the interface. It makes physical sense that both temperature and velocity are continuous. In addition to this, we note two principles of the interface which will help us:

1. The net stress on the interface is zero.
2. Heat which flows into the interface flows back out.

These are equivalent to

$$\left(p - \frac{4}{3}\mu u_x\right)_{gas} = \left(p - \frac{4}{3}\mu u_x\right)_{liquid} \tag{59}$$

$$(kT_x)_{gas} = (kT_x)_{liquid} \tag{60}$$

and obviously also give jump conditions for the interface. With these conditions, we see the F_I in equation 58 is consistent with both sides of the interface. Note that in our case equation 59 reduces to

$$\left(p - \frac{4}{3}\mu u_x\right)_{gas} = p_{liquid} \tag{61}$$

since the velocity in the liquid is constant.

3.4. EVALUATION OF THE INTERFACE FLUX

We will consider Figure 1 with gas on the left and liquid on the right. The opposite case is symmetric.

3.4.1. *Mass Fractions*

In order to evaluate the viscosity and thermal conductivity on the gas side of the interface, we need to know the mass fractions. The first $N-1$ mass fractions are interpolated from the gas, since they do not exist in the liquid. The Nth mass fraction is formed from those using equation 7. We interpolate to high order, but reduce the order if non-physical values are predicted at the interface, i.e. any negative mass fraction. Mass fractions may also be predicted to be greater than one, but equation 7 shows that we cannot have a mass fraction greater than one without another mass fraction being negative. The third, second, and first order interpolations for the mass fractions are $\vec{Y} = (Y_1, \cdots, Y_{N-1})$ are

$$\vec{Y}_I = \left(\frac{\vec{Y}_i - 2\vec{Y}_{i-1} + \vec{Y}_{i-2}}{2dx^2}\right)\left(h_i^n - \frac{dx}{2}\right)^2 +$$

$$\left(\frac{3\vec{Y}_i - 4\vec{Y}_{i-1} + \vec{Y}_{i-2}}{2dx}\right)\left(h_i^n - \frac{dx}{2}\right) + \vec{Y}_i \tag{62}$$

$$\vec{Y}_I = \left(\frac{\vec{Y}_i - \vec{Y}_{i-1}}{dx}\right)\left(h_i^n - \frac{dx}{2}\right) + \vec{Y}_i \tag{63}$$

$$\vec{Y}_I = \vec{Y}_i \tag{64}$$

respectively.

3.4.2. *Temperature and Thermal Conductivity*

Given the interface temperature T_I, we can compute discrete approximations to T_x in both the liquid and gas phases using one sided differences. Note we should not difference across the interface, since T is not necessarily smooth across the interface.

Thus the discretized form of the energy flux equation 60 yields an implicit equation for T_I,

$$f(T_I) = (kT_x)_{gas} - (kT_x)_{liquid} = 0 \tag{65}$$

(the thermal conductivities are functions of T_I as well, in general, and also of the known mass fractions at the interface in the gas phase).

In equation 65, we replace T_x on the gas side of the interface with the following third order discretization,

$$A_1 = \frac{T_I - T_i}{h_i^n - \frac{dx}{2}}, \qquad A_2 = \frac{T_I - T_{i-1}}{h_i^n + \frac{dx}{2}}, \qquad A_3 = \frac{T_I - T_{i-2}}{h_i^n + \frac{3dx}{2}}$$

$$A_4 = \frac{\left(h_i^n + \frac{dx}{2}\right)A_1 - \left(h_i^n - \frac{dx}{2}\right)A_2}{dx}, A_5 = \frac{\left(h_i^n + \frac{3dx}{2}\right)A_1 - \left(h_i^n - \frac{dx}{2}\right)A_3}{2dx}$$

$$T_x = \frac{\left(h_i^n + \frac{3dx}{2}\right)A_4 - \left(h_i^n + \frac{dx}{2}\right)A_5}{dx} \tag{66}$$

where T_x is a function of T_I. We replace T_x on the liquid side of the interface using the symmetric, third order equivalent to equation 66.

Finally, we solve equation 65 for T_I using Newton-Raphson iteration. As an initial guess for the interface temperature we use the average

$$(T_I)_o = \frac{T_i + T_{i+1}}{2} \tag{67}$$

although a k-weighted harmonic average would yield and even better guess.

The resulting T_I is used to evaluate the kT_x term in the interface flux. It is important to use the exact same numerical discretization of T_x to evaluate the interface flux as was used in the Newton-Raphson iteration, i.e. equation 66. If we used a different discretization for the interface flux evaluation, say fourth order, than there is no guarantee that equation 60 holds in the discretized form. If equation 60 does not hold in discretized form, there could be problems with both accuracy and conservation, depending on how the interface flux is treated.

3.4.3. *Velocity and Viscosity*

Now that we know the mass fractions and the temperature at the interface, we can compute the viscosity on the gas side of the interface using Chemkin (Kee *et al.*, 1986).

The velocity will be continuous at the interface, although its derivative is usually not. Since the velocity in the liquid is constant, the interface velocity,u_I, will be identical to the liquid velocity. Also, $u_x = 0$ on the liquid side of the interface. To compute u_x on the gas side of the interface, we use the known interface velocity, u_I, to interpolate u_x to third order

$$A_1 = \frac{u_I - u_i}{h_i^n - \frac{dx}{2}}, \qquad A_2 = \frac{u_I - u_{i-1}}{h_i^n + \frac{dx}{2}}, \qquad A_3 = \frac{u_I - u_{i-2}}{h_i^n + \frac{3dx}{2}}$$

$$A_4 = \frac{\left(h_i^n + \frac{dx}{2}\right) A_1 - \left(h_i^n - \frac{dx}{2}\right) A_2}{dx}, A_5 = \frac{\left(h_i^n + \frac{3dx}{2}\right) A_1 - \left(h_i^n - \frac{dx}{2}\right) A_3}{2dx}$$

$$u_x = \frac{\left(h_i^n + \frac{3dx}{2}\right) A_4 - \left(h_i^n + \frac{dx}{2}\right) A_5}{dx} \tag{68}$$

3.4.4. *Pressure*

The pressure at the interface is interpolated from the gas phase, since we do not have a local equation of state for the pressure in the liquid. We interpolate to high order, but reduce the order if non-physical values are predicted at the interface, i.e. if $p \le 0$. The third, second, and first order interpolations for the pressure are

$$\begin{aligned} p_I \quad = \quad & \left(\frac{p_i - 2p_{i-1} + p_{i-2}}{2dx^2}\right)\left(h_i^n - \frac{dx}{2}\right)^2 + \\ & \left(\frac{3p_i - 4p_{i-1} + p_{i-2}}{2dx}\right)\left(h_i^n - \frac{dx}{2}\right) + p_i \end{aligned} \tag{69}$$

$$p_I = \left(\frac{p_i - p_{i-1}}{dx}\right)\left(h_i^n - \frac{dx}{2}\right) + p_i \tag{70}$$

$$p_I = p_i \tag{71}$$

respectively. Note that this is the same as that for the mass fractions.

Since we know p, μ, and u_x in the gas, we can use equation 61 to find the pressure on the liquid side of the interface. In 2D this would also include any pressure jump due to surface tension forces. This in turn provides the pressure boundary condition needed to solve the global equation for pressure inside the incompressible liquid.

Note that our procedure of interpolating pressure in from the gas phase, and using the pressure balance across the interface to deduce pressure on the liquid side, makes the implicit assumption that a pressure shock does not exist at the interface. If one does, a brief transient error may result during the time it takes for the shock to be transmitted one cell.

3.5. PRACTICAL USE OF THE INTERFACE FLUX

Once the interface flux has been constructed, it has a single value which is used on both sides of the interface. Most of its terms are identically zero, but we do have a non-zero flux for both momentum and energy.

If momentum, ρu, and total energy, E, are included in the conserved variables on both sides of the interface, then we just apply the interface flux to update the conserved variables according to equations 41 and 42 or equations 43 and 44, depending on which are relevant. In equation 1, for the gas side of the interface, $\vec{U}$ contains momentum and total energy as conserved variables, as can be seen in $\vec{U}$ in equation 3. In equation 25, for the liquid side of the interface, $\vec{U}$ contains momentum and total energy as conserved variables, as can be seen in $\vec{U}$ in equation 26. Thus, if equations 1 and 25 are our conservation equations for gas and liquid, then no special treatment is needed.

Suppose that we used equations 28 and 32 for the liquid. Then one has to be careful when updating. Equation 28 can be rewritten as

$$(\rho u)_t = -p_x \tag{72}$$

since ρ is a constant. Then, we can update the momentum using the interface flux. This is straight forward. It is not as easy to see how to handle equation 32. In fact, there are two choices:

1. We can maintain conservation of *total energy*, E, which will maintain global conservation of energy.

2. We can maintain conservation of *internal energy*, ρe, which is dictated by the special incompressible assumptions leading to equation 32. This will not maintain global conservation of energy.

3.5.1. *Conservation of Internal Energy in the Liquid*
The kinetic energy is defined as

$$KE = \frac{\rho u^2}{2} \tag{73}$$

where ρ is a constant in the liquid. Thus u uniquely determines both the momentum and the kinetic energy. Therefore, the entire interface flux for energy is not needed. On the liquid side, we rewrite the interface flux as

$$F_I = \begin{pmatrix} 0 \\ p_{liquid} \\ (-kT_x)_{liquid} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{74}$$

with the kinetic energy portion of the energy flux deleted.

One should be careful when using this method since it is not conservative. Using the unaltered interface flux yields a change in kinetic energy of

$$\triangle KE = dt(up) \tag{75}$$

across the interface. Using the altered interface flux in equation 74, the change in velocity is

$$\triangle u = dt\left(\frac{p}{\rho}\right) \tag{76}$$

across the interface. The corresponding change in kinetic energy is

$$\triangle KE = \frac{\rho(u+\triangle u)^2}{2} - \frac{\rho u^2}{2} = dt(up) + dt^2\left(\frac{p^2}{2\rho}\right) \tag{77}$$

across the interface. Note that equation 75 gives the conservative update of the kinetic energy, while equation 77 differs from the conservative update in the second order term. These updates were done for Euler's Method. For third order Runge-Kutta, the error in conservation would be fourth order. This method is only conservative up to the order of the temporal scheme.

3.5.2. *Global Conservation of Total Energy*

To keep the scheme entirely conservative, we have to move the truncation error seen in equation 77 to a different location. This is done by using the same value of the interface flux on both sides of the interface. Again, we only need modifications on the liquid side.

The velocity is used to compute the kinetic energy which is then added to the internal energy to get the total energy,

$$E^n = \frac{\rho(u^n)^2}{2} + (\rho e)^n \tag{78}$$

which is updated to E^{n+1} using the interface flux. We also update the momentum with the interface flux and find the new velocity. Then the internal energy is reconstructed as

$$(\rho e)^{n+1} = E^{n+1} - \frac{\rho(u^{n+1})^2}{2} \tag{79}$$

to finish the conservative scheme. The change in internal energy across the interface is

$$\triangle(\rho e) = \triangle E - \left(\frac{\rho(u^{n+1})^2}{2} - \frac{\rho(u^n)^2}{2}\right) = -kT_x - dt^2\left(\frac{p^2}{2\rho}\right) \tag{80}$$

when it should only have been $-kT_x$. Here the second order error is in the non-physical conversion of kinetic energy to internal energy, which is not dictated by equation 32.

3.6. DISCRETIZATION IN THE LIQUID

We use equations 1 and 25 as our conservation equations for gas and liquid, so that momentum and total energy are included in the conserved variables on both sides of the interface. Thus, we use the same value of the interface flux on both sides of the interface. The gas equations are easily discretized (Fedkiw *et al.*, 1996), but the liquid equations require some thought.

Consider the the first row in equation 25,

$$(\rho u)_t + p_x = 0 \tag{81}$$

which can be updated once we know the pressure. Since the interface evaluation process gave us the pressure on both sides of the droplet, we can compute p_x as a constant since the pressure profile in the droplet is known to be linear from equation 29. Thus p_x is just a source term. Since fluxes are needed to properly do the interface portion of the calculation, we decompose this spatially constant source term into fluxes. Each flux is assigned the known pressure value at the flux point. Then, differencing the fluxes gives us back our spatially constant source term p_x.

Consider the last row in equation 25,

$$E_t + (uE + up)_x = (kT_x)_x \tag{82}$$

which can be simplified using the fact that density is constant in the droplet. Constant density, along with $u_x = 0$ from equation 22 implies that $(KE)_x = 0$ in the droplet. These considerations simplify equation 82 to

$$E_t + (u\rho e)_x + up_x = (kT_x)_x \tag{83}$$

where up_x is a constant, since velocity is constant in the liquid, and pressure is linear. Thus up_x is a spatially constant source term. Once again we decompose the source term up_x into fluxes, where each flux is evaluated as the known value of up. Differencing the fluxes gives us back our spatially constant source term up_x, since u is constant. The diffusion term kT_x is discretized with conservative central differencing. The convection term $(u\rho e)_x$ is just the convection of internal energy and can be discretized with ENO, using u as the upwind direction.

In equation 81 and equation 83, we have written all the terms defining the fluxes at the cell walls. This allows us to easily apply our interface method.

3.7. SPECIAL STABILITY CONSIDERATIONS

Th numerical method designed above generates a small amount of noise which propagates into the surrounding gas. To stop this, we lower the accuracy of the flux adjacent to the interface so that it is only second order. This single flux acts as some sort of a filter. The reasons for this mild instability are still unclear at this time.

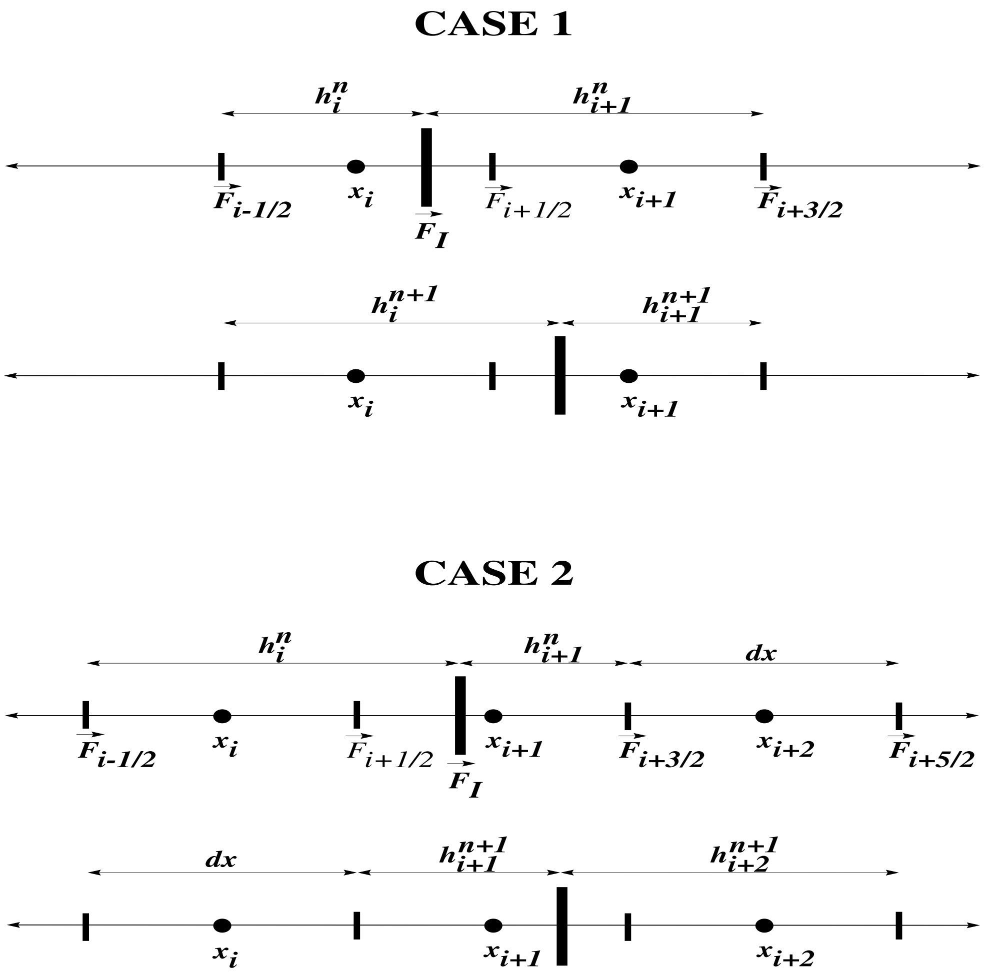

Figure 1. In case 1, the interface stays between grid nodes. In case 2, the interface crosses a grid node and ends up in an adjacent cell.

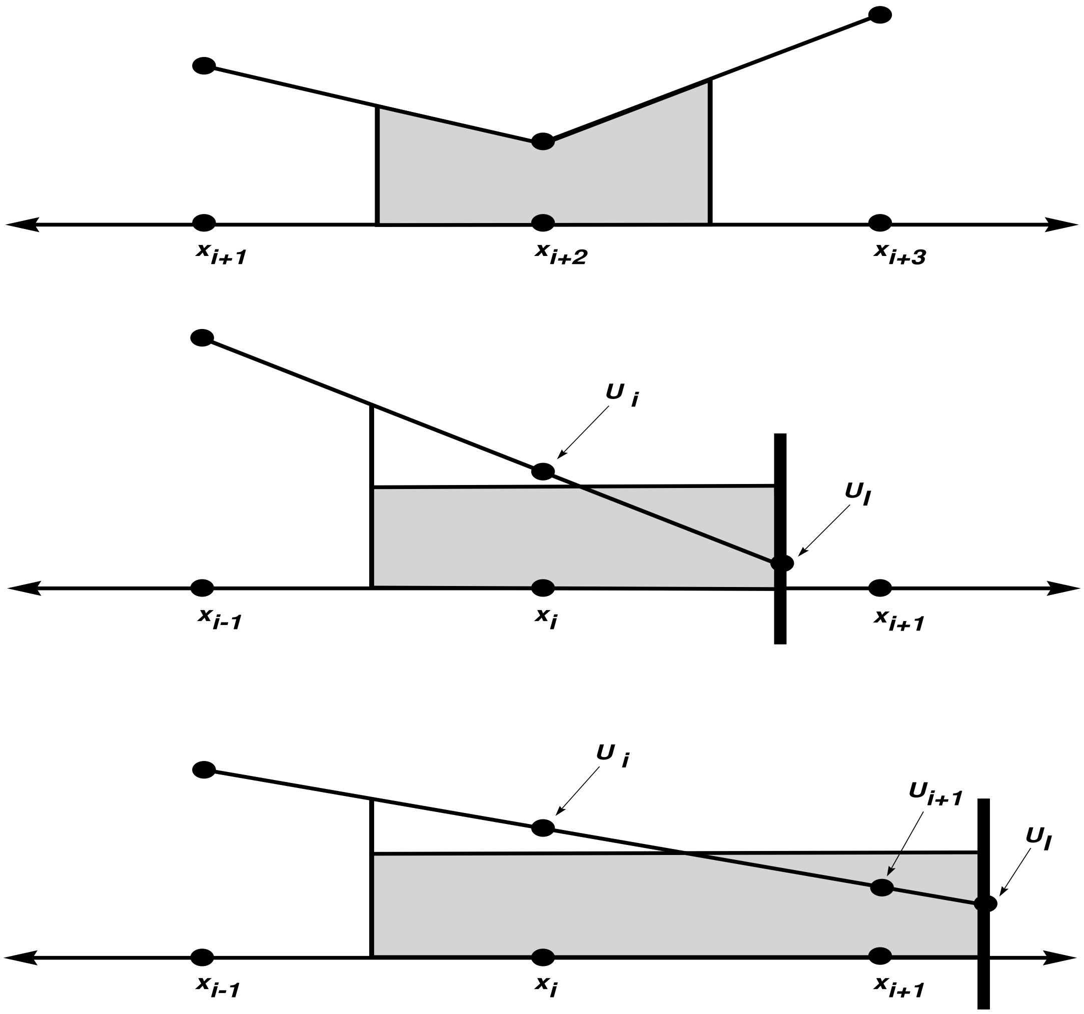

Figure 2. Geometric representation of the transition back and forth from cell averages to point values.

4. Runge-Kutta Methods

For time integration of an equation of the form

$$\vec{U}_t = \vec{f}(\vec{U}) \tag{84}$$

we use the TVD Runge Kutta methods (Shu and Osher, 1988). First order TVD Runge Kutta is forward Euler,

$$\vec{U}^{n+1} = \vec{U}^n + \triangle t \vec{f}(\vec{U}^n) \tag{85}$$

Second order TVD Runge Kutta is Heun's predictor-corrector method,

$$\vec{U}^{\star} = \vec{U}^n + \triangle t \vec{f}(\vec{U}^n) \tag{86}$$

$$\vec{U}^{n+1} = \vec{U}^n + \triangle t \left(\frac{1}{2} \vec{f}(\vec{U}^n) + \frac{1}{2} \vec{f}(\vec{U}^{\star}) \right) \tag{87}$$

A third order TVD Runge Kutta method is given by,

$$\vec{U}^{\star} = \vec{U}^n + \triangle t \vec{f}(\vec{U}^n) \tag{88}$$

$$\vec{U}^{\star\star} = \vec{U}^n + \triangle t \left(\frac{1}{4} \vec{f}(\vec{U}^n) + \frac{1}{4} \vec{f}(\vec{U}^{\star}) \right) \tag{89}$$

$$\vec{U}^{n+1} = \vec{U}^n + \triangle t \left(\frac{1}{6} \vec{f}(\vec{U}^n) + \frac{1}{6} \vec{f}(\vec{U}^{\star}) + \frac{2}{3} \vec{f}(\vec{U}^{\star\star}) \right) \tag{90}$$

The scheme in the previous section was described using Euler's method. Our scheme can be extended to second and third order TVD Runge Kutta as well. Each partial Runge Kutta step is seen as an "Euler type" step where the right hand side is the average of the right hand sides from the appropriate time levels. For points "away" from the interface, the right hand side is evaluated in the standard way. However, one must be careful for points "near" the interface.

Normal fluxes are only computed in cells which do not contain the interface. Cells which contain the interface contain the special moving interface flux. To compute the right hand side for a partial step of a Runge Kutta method, we intersect the normal fluxes across all time levels that are used in the right hand side of the partial step. Then we use these intersected fluxes to find the convective derivative on the right hand side for each time level. Then the convective derivatives are averaged in the appropriate way.

There is a grid point or two left which are not updated. These are updated using the interface method with the new averaged interface flux and its nearest averaged neighbor which lives in the intersection described above. Essentially, these points are updated in a conservative way using some of the interpolation described in the previous section. We have employed second and third order Runge Kutta methods and prefer third order. We will not spell out the details of the higher order Runge Kutta here, since there are many cases and it is quite lengthy. (More to come on higher order Runge Kutta in a future report.)

5. Numerical Examples

In the figures, we do not show the actual values of density and pressure for the water droplet. Density is not shown, since it is is off the scale reasonable for the gas. Pressure is not shown, since it is not computed in the 1D water model. It has a linear profile and must be calculated using the pressure in the neighboring air. We use "place holder" values for the density and the pressure, just to show the location of the water droplet. However, the values for the velocity and the temperature are unaltered.

5.1. EXAMPLE 1

We take a 1-D domain of length $10cm$ and a grid with 100 points. We have solid wall boundaries on both sides of the domain. Inside the domain we have an incompressible water droplet and a compressible, thermally perfect gas. We will assume that they are both inviscid for this calculation.

The water droplet is five grid cells long and starts at rest near the left hand side of the domain with an initial temperature of $298K$. The rest of the chamber is filled with argon gas which also starts at rest with a temperature of $1000K$. The gas to the right of the drop is at standard atmospheric pressure, while the gas to the left of the drop has a pressure over five times higher than atmospheric pressure.

We expect the high pressure air to drive the droplet to the right. The air on the left will expand and cool, while the air on the right will be compressed and heated. Eventually the droplet will compress the air to the right enough to allow a net force to the left which will slow the droplet and reverse its direction. This process will repeat and the droplet will bounce back and forth in a spring like fashion.

Third order ENO with Marquina's Jacobian is used to discretize the relevant fluxes. Then our phase interface method is coupled to third order TVD Runge Kutta in the appropriate way.

Figure 3 shows the droplet after a short time. It is being driven to the left by the pressure difference. The droplet continues to the right. The air to the left is decompressed and cooled, while the air to the right is compressed and heated. This can be seen in Figure 4 where the velocity of the droplet is over $30\frac{m}{s}$. Note that the pressure on the right now exceeds that on the left, and the net force is now to the left slowing down the droplet. The droplet will keep slowing down until it comes to a complete rest for an brief instant, before it starts moving in the opposite direction. Figure 5 shows the droplet almost at rest, just before it turns around in the other direction. Figure 6 shows the droplet moving in the opposite direction Now it is decompressing and cooling the air on the right while it compresses and heats the air on the left. This process continues, mimicking a spring-like phenomena.

Note that we conserve total energy. Thus there will be an error in the computation of the internal energy in the water. Equation 80 shows that the internal energy will decrease in a nonphysical way proportional to $\frac{p^2}{2\rho}$. Since $\frac{p^2}{2\rho}$ is larger on the right hand side of the droplet then on the left, for most of our computation, we would expect the temperature on the right of the droplet to be lower than that on the left. This can be seen in figures 4, 5, and 6.

5.2. EXAMPLE 2

We take a 1-D domain of length $10cm$ and a grid with 100 points. We have solid wall boundaries on both sides of the domain. Inside the domain we have an incompressible water droplet and a compressible, thermally perfect gas. We will assume that they are both inviscid for this calculation.

The water droplet is five grid cells long and starts at rest near the right hand side of the domain with an initial temperature of $298K$. The rest of the chamber is filled with argon gas which also starts at rest with a temperature of $1000K$. Near the center of the domain, we impose a pressure jump. The gas to the right of this jump is at standard atmospheric pressure, while the gas to the left of this jump has a pressure over five times higher than atmospheric pressure. This is a "Sod" shock tube problem with a water droplet included in the tube.

We expect the pressure jump to split into a shock, a contact discontinuity, and a rarefaction. The shock will travel to the right of the domain and collide with the water droplet, reflecting off in the opposite direction.

Third order ENO with Marquina's Jacobian is used to discretize the relevant fluxes. Then our phase interface method is coupled to third order TVD Runge Kutta in the appropriate way.

Figure 7 shows the shock traveling to the right toward the water droplet. It hits the droplet, reflects off, and then travels in the opposite direction as shown in figure 8. Note that the shock will impart momentum to the droplet forcing it to the right with high pressure. This can be seen slightly in figure 8. Figure 9 shows the solution at a later time, where it is more apparent that the droplet is moving to the right. Here, the velocity of the droplet is positive and the velocity of the air to the right of the droplet has a linear profile.

Figures 10 and 11 show the same example with 400 points, just to illustrate convergence. Note that there is a peak in the density and a dip in the temperature by the droplet after reflection. This is the standard effect due to loss of room to interpolate.

5.3. EXAMPLE 3

We repeat example 1, except this time we allow the flow to be fully viscous.

Once again the droplet is driven to the left, slows down, and reverses direction. Figure 12 shows the droplet after reversing direction. Comparing this to figure 6, we can see that the cold droplet is absorbing heat from the hot gas. The gas temperature profile near the droplet is falling, while the corresponding density profile is increasing.

Figure 13 shows the drop after reversing direction once again, now traveling to the right.

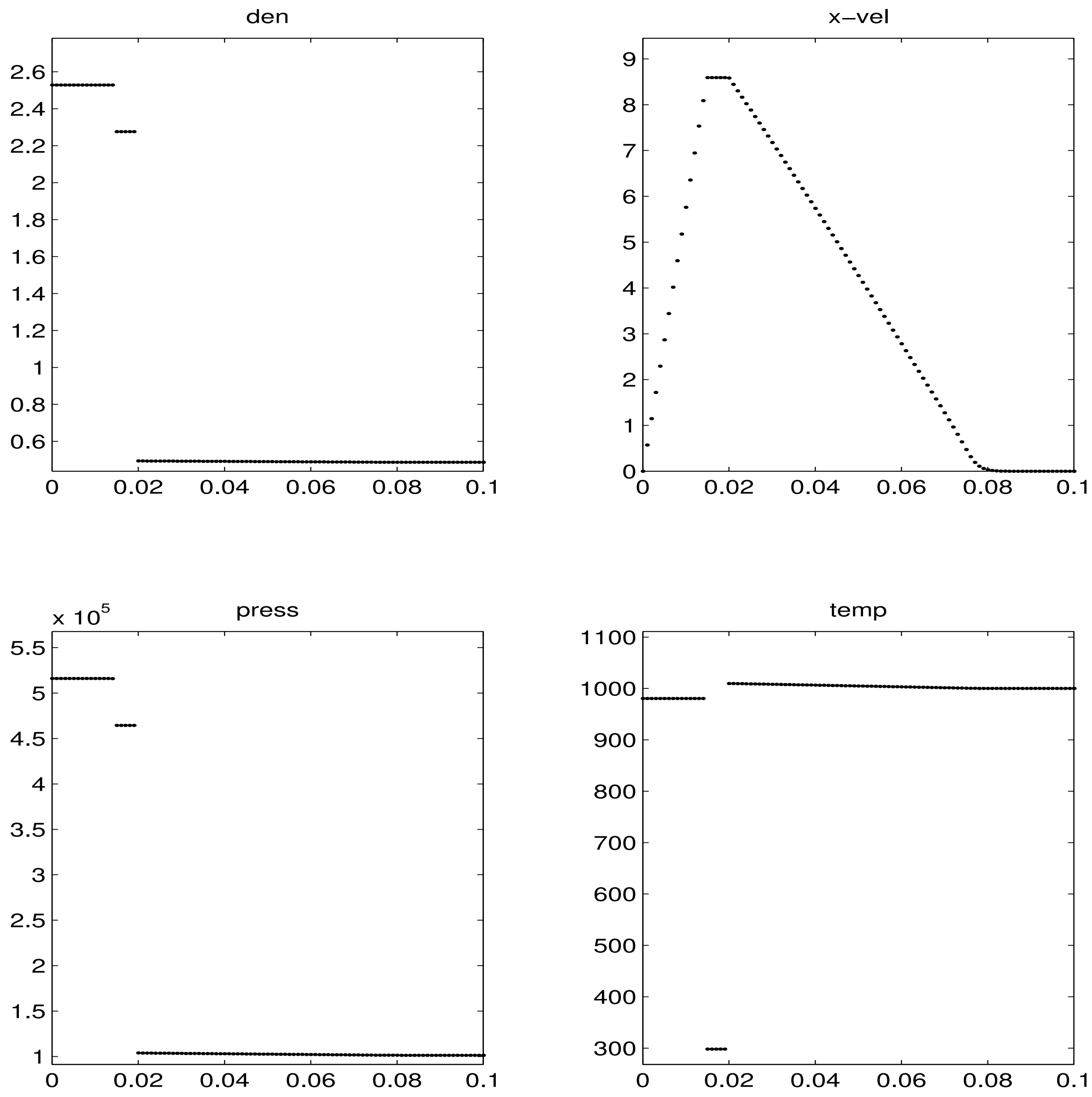

Figure 3. The water droplet is initially driven to the right from the pressure difference in the air. Inviscid case.

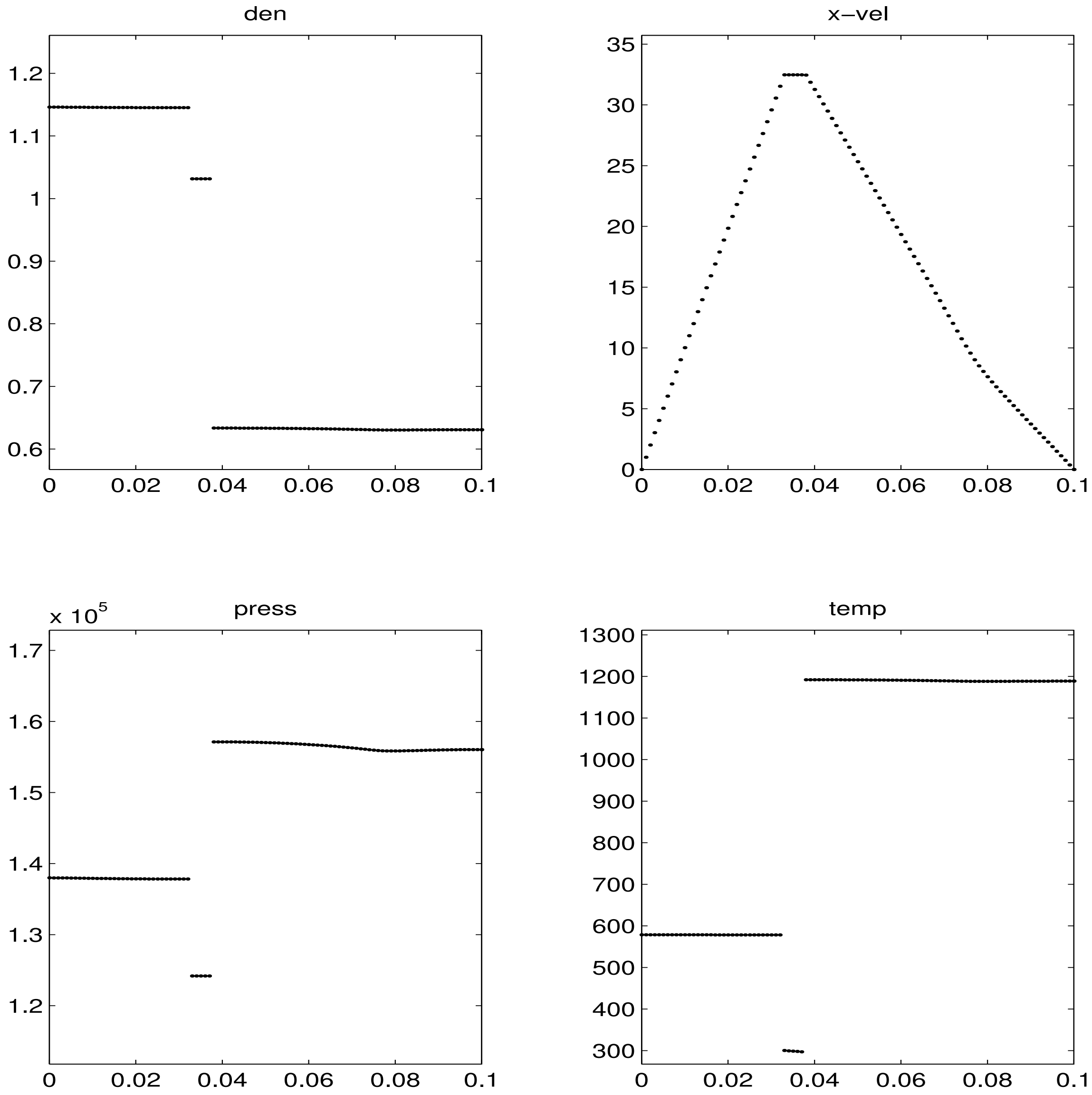

Figure 4. The gas on the left has been decompressed and cooled, while the gas on the right has been compressed and heated. Now, the net force is to the left and the droplet will start to slow down. Inviscid case.

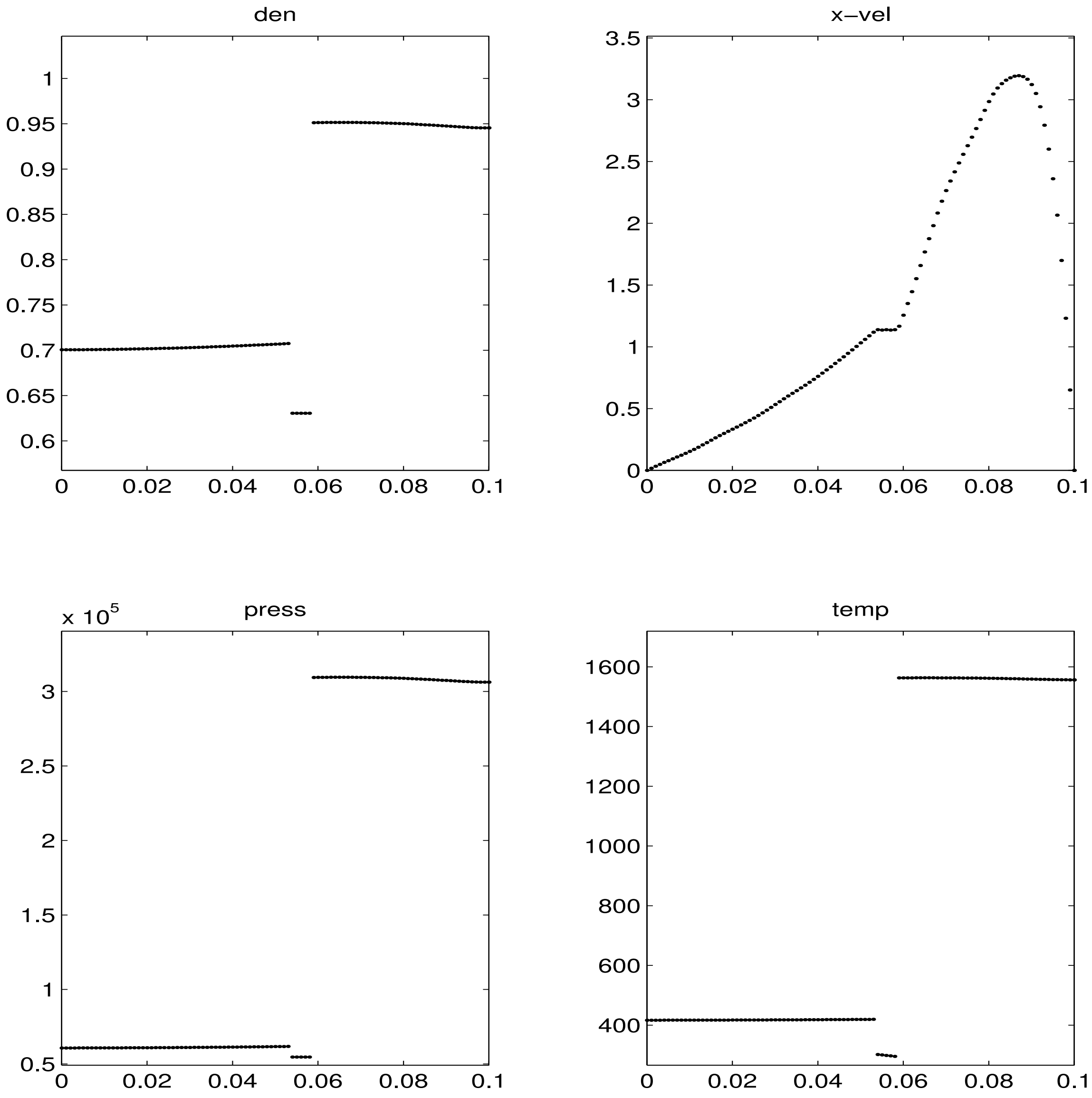

Figure 5. The droplet has almost come to a complete rest, after which it will begin to move in the opposite direction. Inviscid case.

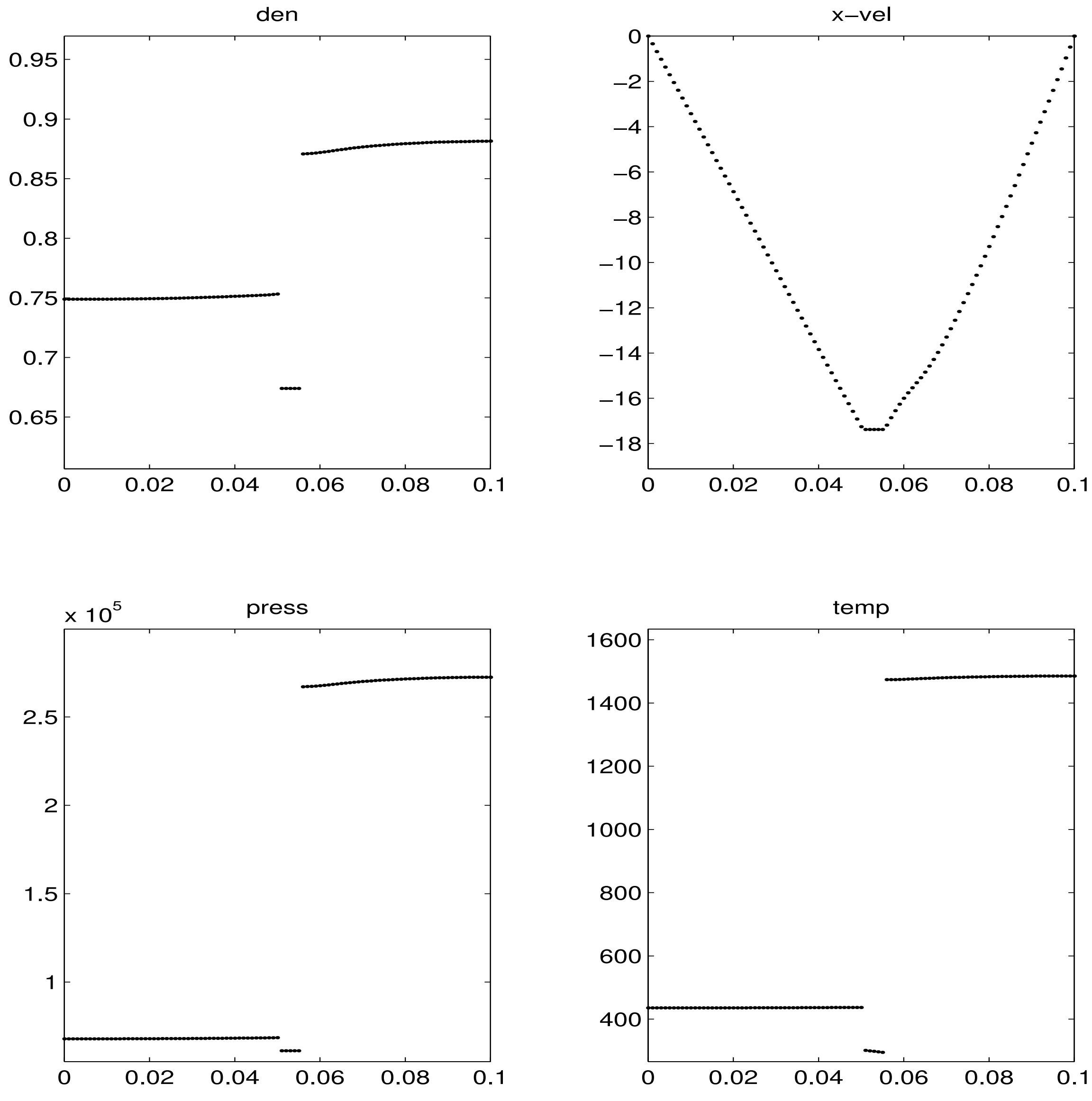

Figure 6. Now, the droplet moves to the left. It decompresses and cools the gas to the left, while it compresses and heats the gas to the right. Inviscid case.

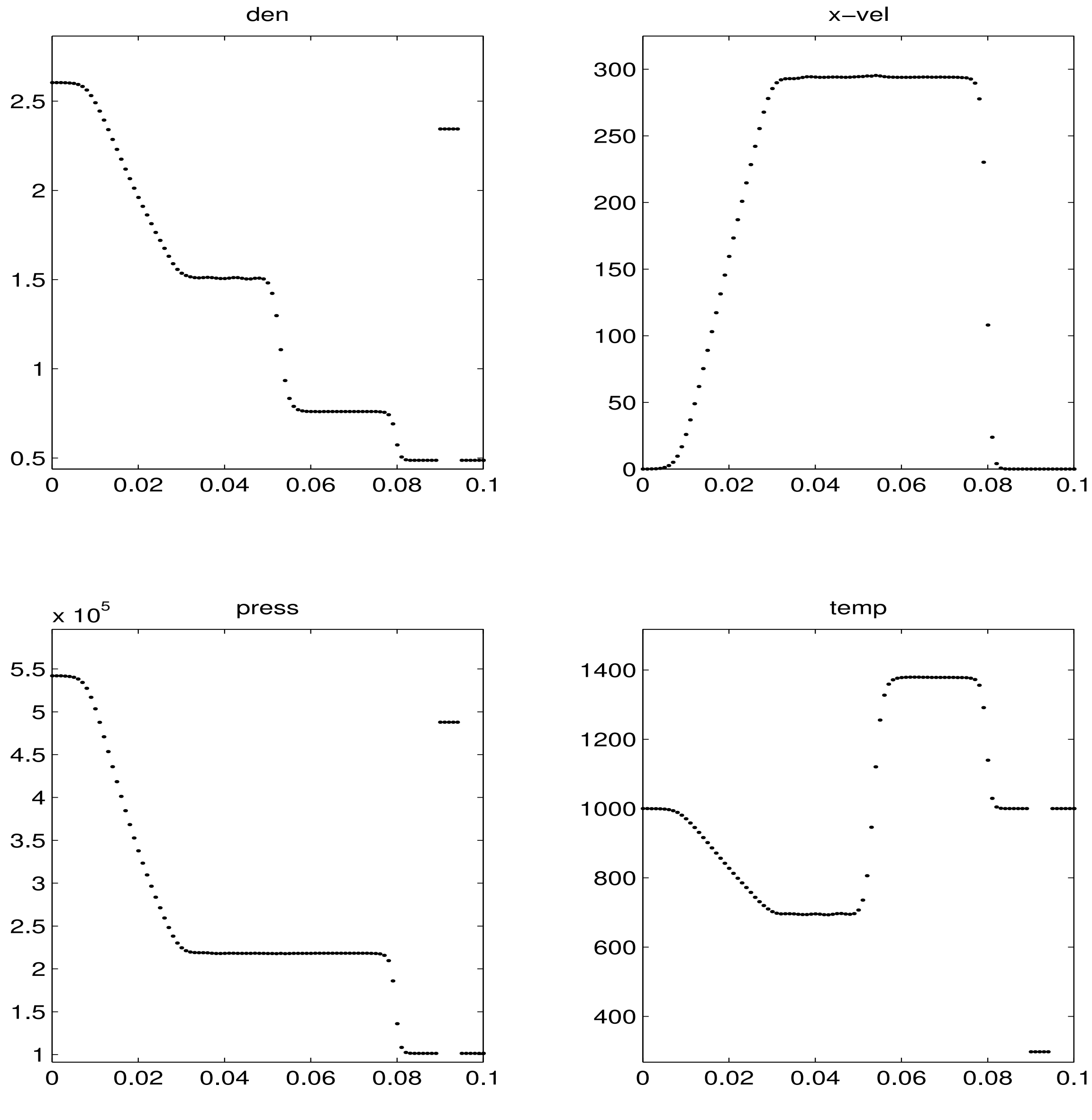

Figure 7. The shock is traveling to the right and is about to collide with the water droplet. 100 grid points.

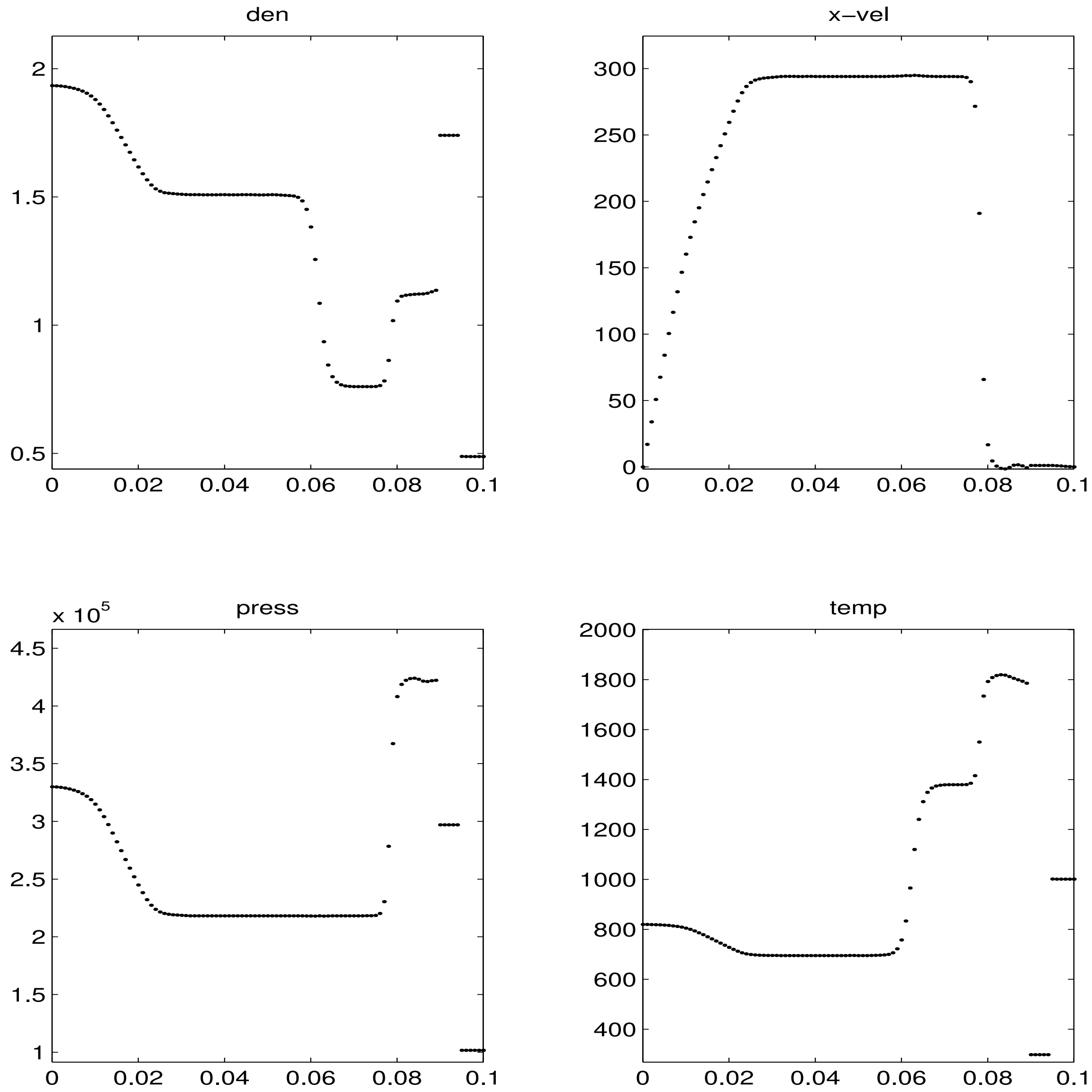

Figure 8. The shock has reflected off the water droplet and is now traveling to the left. 100 grid points.

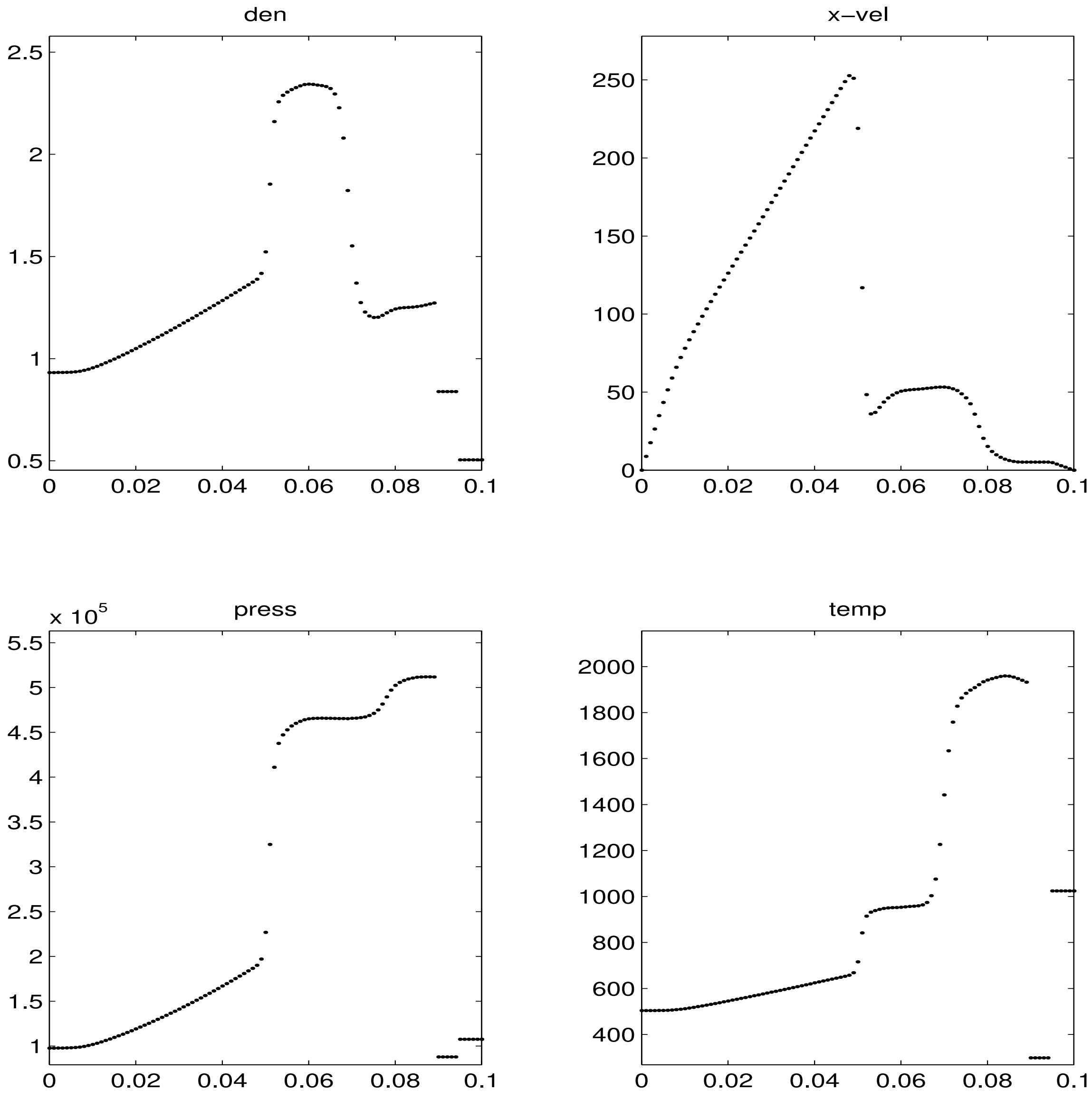

Figure 9. The droplet has a positive velocity and is moving slowly to the right. The air to the right of the droplet has a linear velocity profile. 100 grid points.

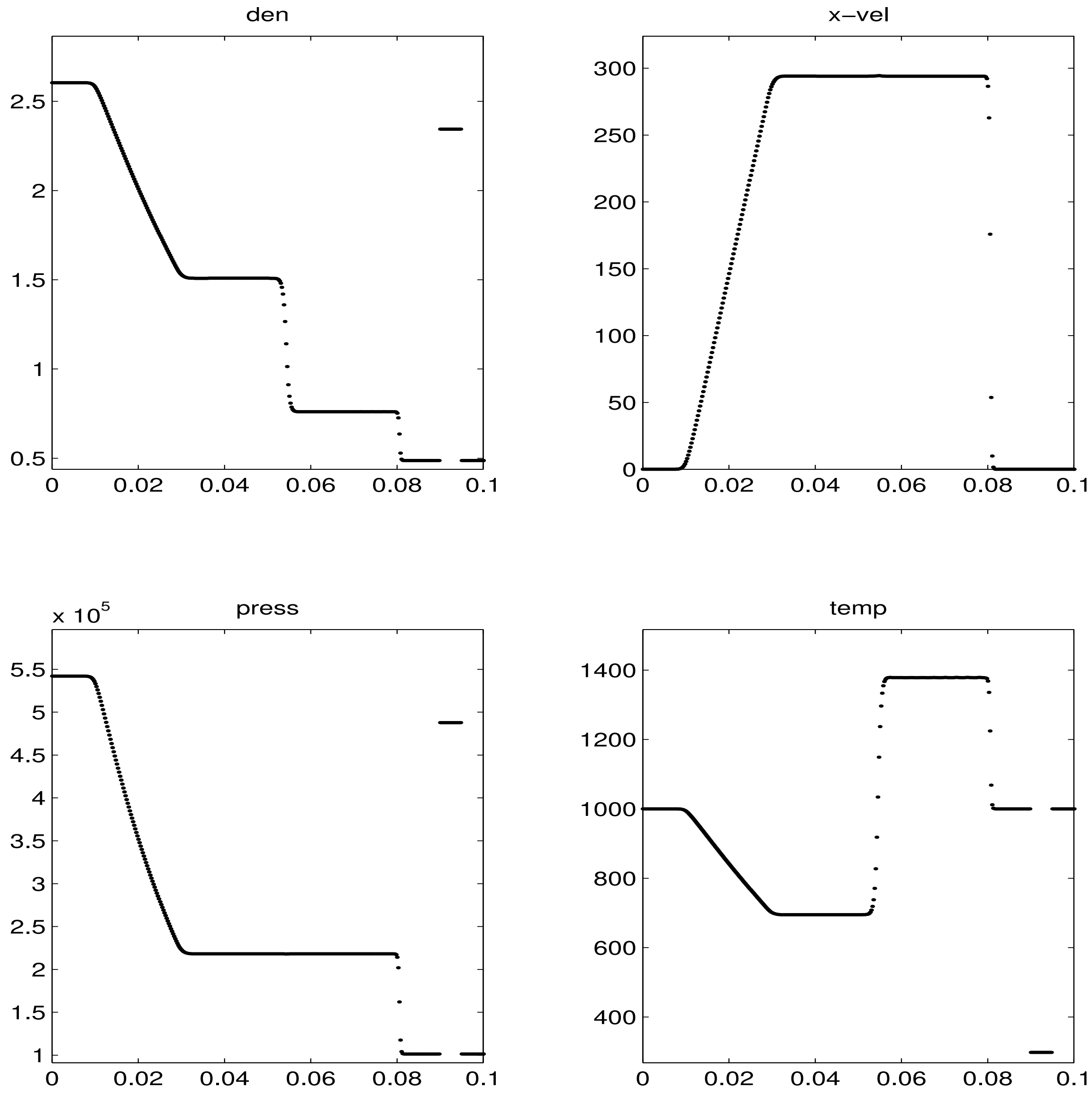

Figure 10. The shock is traveling to the right and is about to collide with the water droplet. 400 grid points.

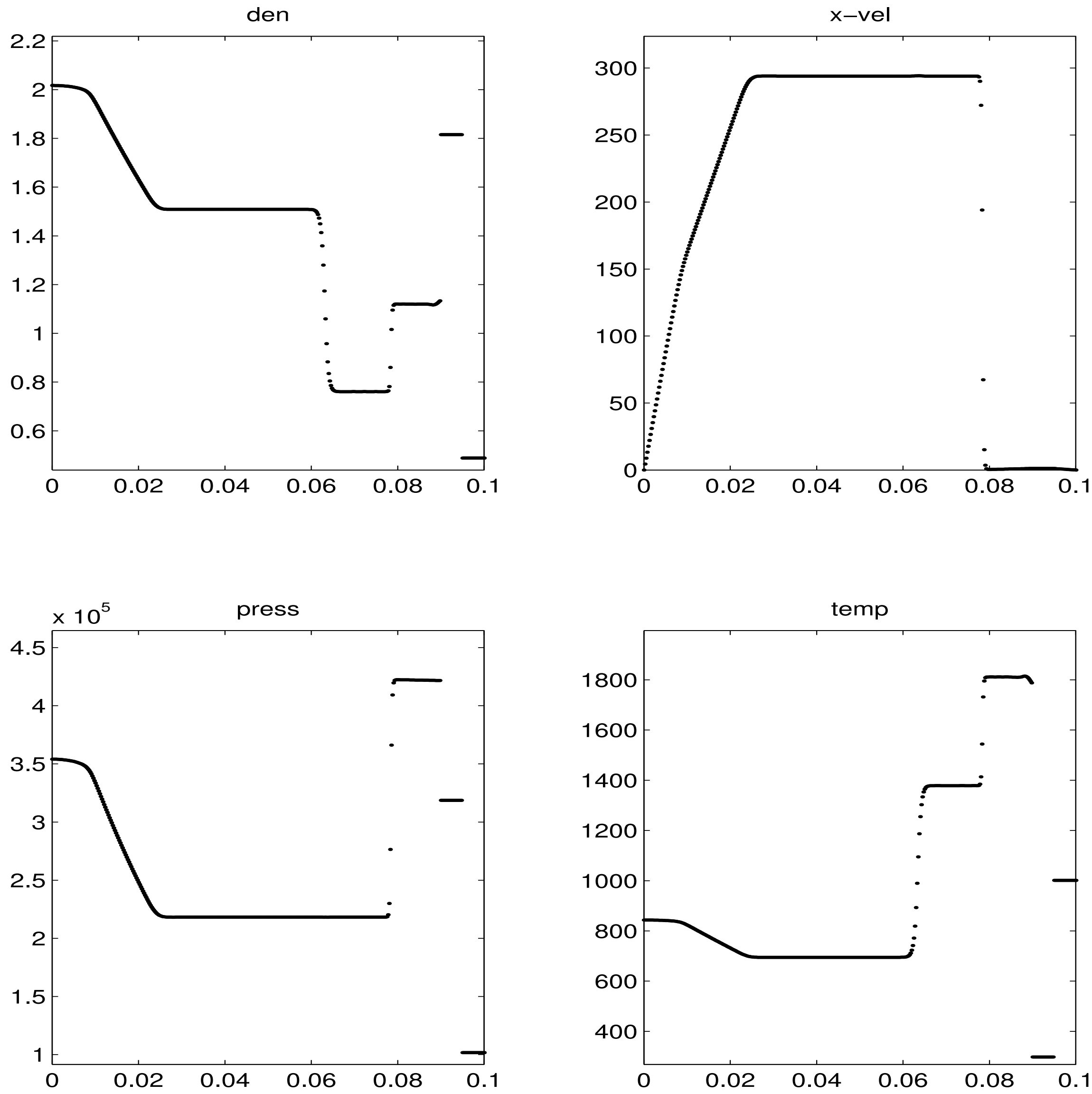

Figure 11. The shock has reflected off the water droplet and is now traveling to the left. 400 grid points.

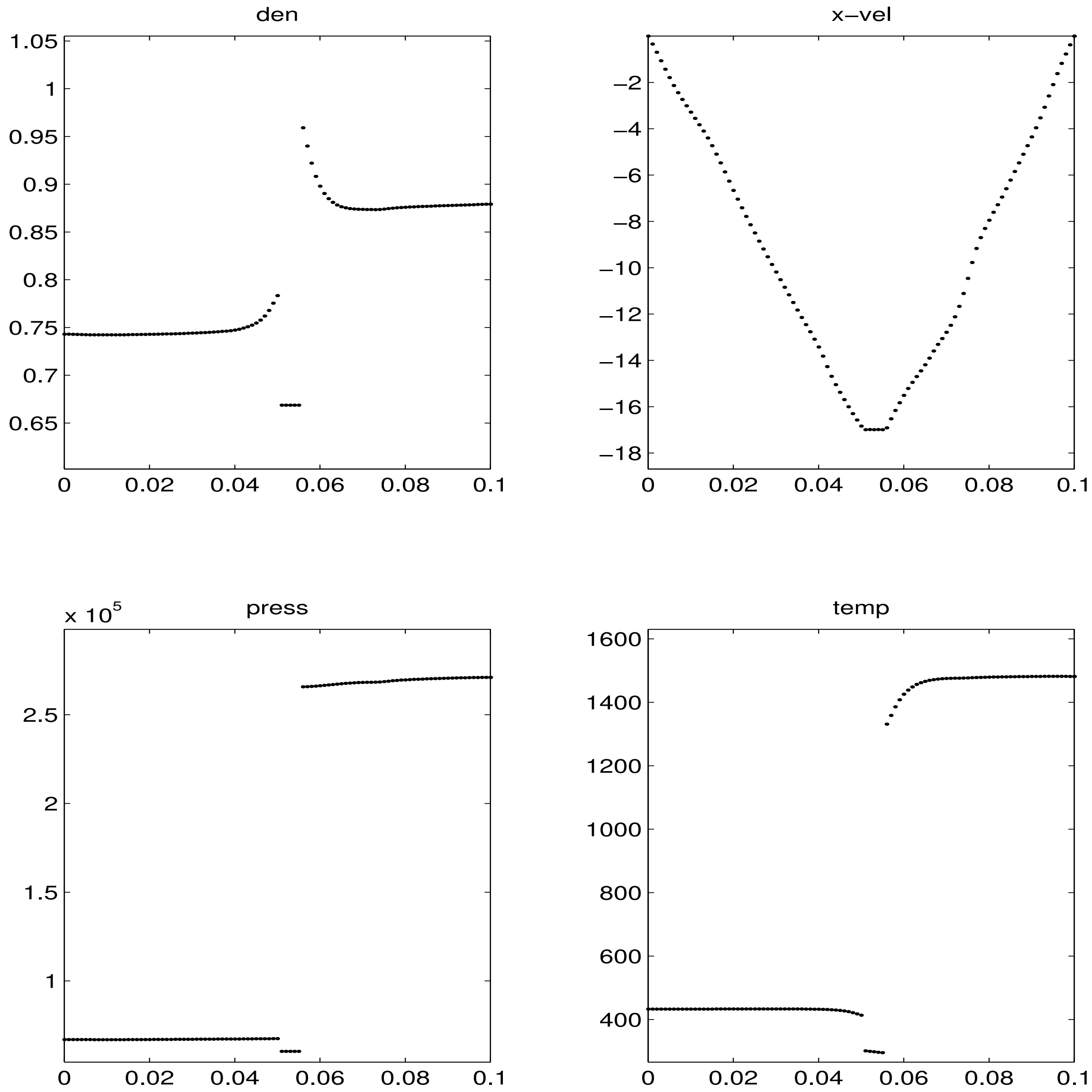

Figure 12. Now, the droplet moves to the left. It decompresses and cools the gas to the left, while it compresses and heats the gas to the right. Viscous case.

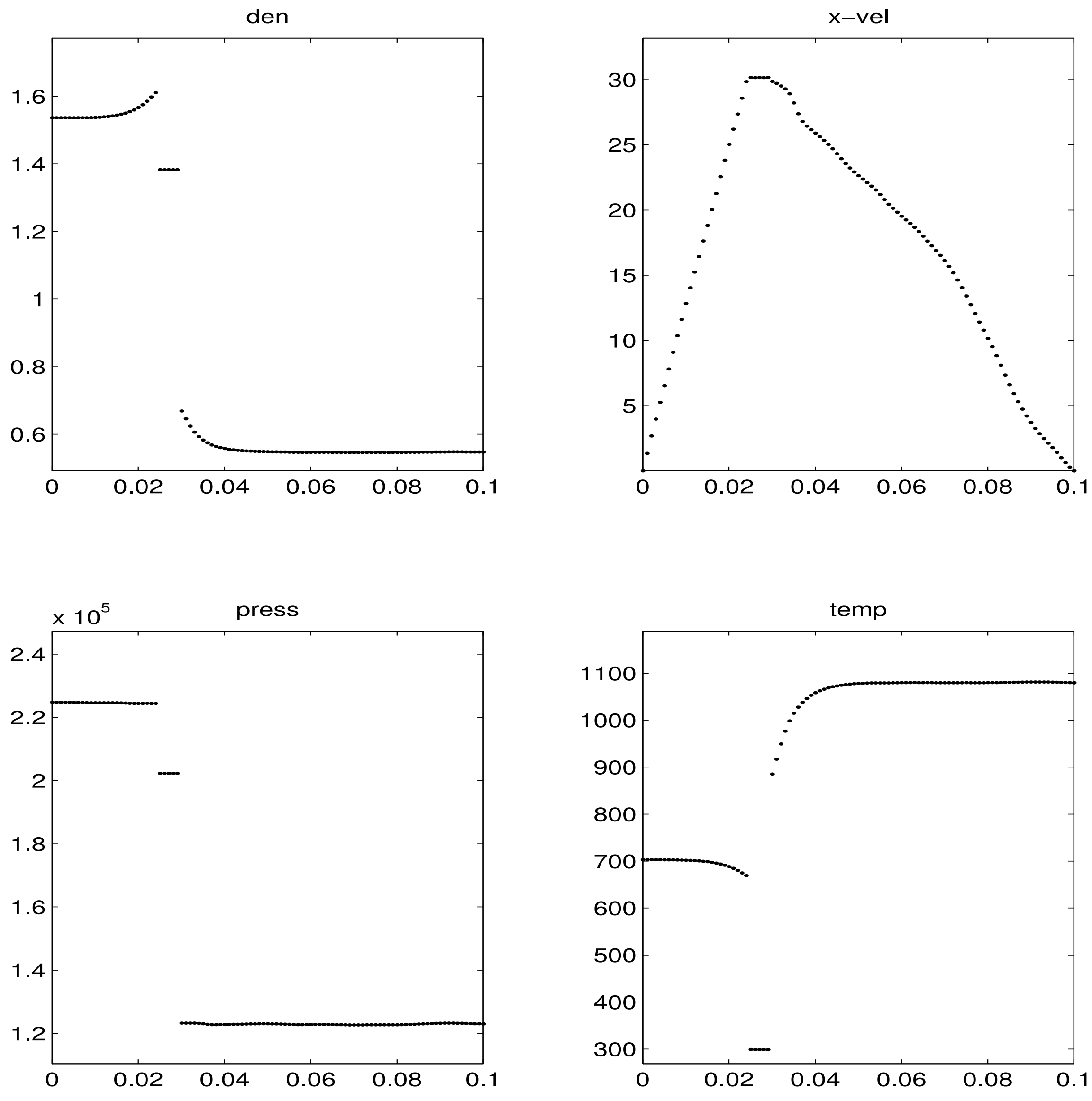

Figure 13. Once again, the droplet changes direction. It is now move to the right. Viscous case.

6. Conclusion

We have developed a new approach to multiphase flow involving a liquid droplet hit by a high speed gas flow, or more generally for any mixed compressible-incompressible flows. Our approach is based on decomposing the domain into distinct regions where the respective, different, model equations apply, and discretizing each region with the most appropriate techniques. Proper boundary conditions are imposed on the internal interface boundary, mainly in terms of continuity or conservation conditions. In order to maintain discrete conservation and use finite difference ENO methods in the gas phase, we developed a novel formulation using a Lagrangian control volume method near the interface. Novel time stepping schemes are also developed for nodes near the interface, since a given node will change its unknowns and governing equations as the interface passes over it.

For basic proof of principle and to investigate the details of of the novel spatial discretization and time stepping required near the interface, we did this initial theory and computations in 1D, Our experiments showed good results for both smooth flows and flows with shocks reflecting from the liquid surface, Future work will extend the method to 2D, using a level set representation of the interface.

A. Eigensystem

Consider $\vec{F}(\vec{U})$ defined in equation 3.

The eigenvalues of the Jacobian matrix of $\vec{F}(\vec{U})$ are

$$\lambda^1 = u - c \tag{91}$$

$$\lambda^2 = \cdots = \lambda^{N+2} = u \tag{92}$$

$$\lambda^{N+3} = u + c \tag{93}$$

The left eigenvectors $\vec{L}^p$, are the rows of the following matrix.

$$\begin{pmatrix} \frac{b_2}{2} + \frac{u}{2c} + \frac{b_3}{2} & -\frac{b_1 u}{2} - \frac{1}{2c} & \frac{b_1}{2} & \frac{-b_1 z_1}{2} & \cdots & \frac{-b_1 z_{N-1}}{2} \\ 1 - b_2 - b_3 & b_1 u & -b_1 & b_1 z_1 & \cdots & b_1 z_{N-1} \\ -Y_1 & 0 & 0 & & & \\ \vdots & \vdots & \vdots & & I & \\ -Y_{N-1} & 0 & 0 & & & \\ \frac{b_2}{2} - \frac{u}{2c} + \frac{b_3}{2} & -\frac{b_1 u}{2} + \frac{1}{2c} & \frac{b_1}{2} & \frac{-b_1 z_1}{2} & \cdots & \frac{-b_1 z_{N-1}}{2} \end{pmatrix} \tag{94}$$

The right eigenvectors $\vec{R}^p$, are the columns of the following matrix.

$$\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 1 \\ u - c & u & 0 & \cdots & 0 & u + c \\ H - uc & H - \frac{1}{b_1} & z_1 & \cdots & z_{N-1} & H + uc \\ Y_1 & Y_1 & & & & Y_1 \\ \vdots & \vdots & & I & & \vdots \\ Y_{N-1} & Y_{N-1} & & & & Y_{N-1} \end{pmatrix} \tag{95}$$

Here I is the $N-1$ by $N-1$ identity matrix, and

$$c = \sqrt{\frac{\gamma p}{\rho}}, \qquad H = \frac{E + p}{\rho} \tag{96}$$

$$b_1 = \frac{\gamma - 1}{c^2}, \quad b_2 = 1 + b_1 u^2 - b_1 H, \quad b_3 = b_1 \sum_{i=1}^{N-1} Y_i z_i \tag{97}$$

$$z_i = h_i - h_N - c_p W \left(\frac{1}{W_i} - \frac{1}{W_N} \right) T \tag{98}$$

For more on the eigensystem, see (Fedkiw *et al.*, 1996).

References

Anderson, John D., *Hypersonic and High Temperature Gas Dynamics*, McGraw-Hill, 1989.

Atkinson, Kendall E., *An Introduction to Numerical Analysis*, Wiley, 1989.

Fedkiw, Ronald P., *A Survey of Chemically Reacting, Compressible Flows*, UCLA (Dissertation), 1996.

Fedkiw, R., Merriman, B., and Osher, S., *Numerical methods for a mixture of thermally perfect and/or calorically perfect gaseous species with chemical reactions*, UCLA CAM Report 96-1, January 1996.

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.

Kee, Miller and Jefferson, *CHEMKIN: A General Purpose Problem Independent, Transportable Fortran Chemical Kinetics Code Package*, SAND 80-8003 Sandia National Laboratories, March 1986.

W. Mulder, S. Osher, and J. A. Sethian, *Computing Interface Motion in Compressible Gas Dynamics*, J. Numer. Analysis, 30, 542, 1991.

Shu, C.W. and Osher, S., *Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes II (two)*, UCLA CAM Report 88-12, April 1988.

Stall, D.R. and Prophet, H., *JANAF Thermochemical Tables*, National Standard Reference Data Series, 1971.

M. Sussman *A Level Set Approach to Computing Incompressible Two Phase Flow*, Ph.D. Thesis, UCLA Department of Mathematics, 1995.