# The PPM Compressible Gas Dynamics Scheme
## (*This is a Draft. References especially are incomplete.*)

Paul R. Woodward
University of Minnesota, LCSE
*Feb. 10, 2005*

## 1. Introduction.

The development of the PPM gas dynamics scheme grew out of earlier work in the mid 1970s with Bram van Leer on the MUSCL scheme [1-3]. At Livermore, in the late 1970s, the use of moments of the internal cell distributions which makes MUSCL resemble finite element schemes was abandoned for reasons of compatibilit with Livermore production codes. In order to recapture the accuracy of MUSCL while using only cell averages as the fundamental data, PPM, the Piecewise-Parabolic Method, was developed in collaboration with Phil Colella [4-7]. Over the last 20 years, PPM has evolved considerably in order to address shortcomings of the scheme as it was laid out in [6] and [7]. PPM has also been extended to MHD [8-18] in collaboration with Wenlong Dai, applied extensively to the study of jets and supersonic shear layers [19-25] in collaboration with Karl-Heinz Winkler, Steve Hodson, Norm Zabusky, Jeffrey Pedelty, and Gene Bassett, used to simulate flow in disk galaxies and around stationary and moving objects [26-28] in collaboration with B. Kevin Edgar, applied extensively to convection and turbulence problems [29-56] in collaboration with David Porter, Annick Pouquet, and Igor Sytine (see later article in this volume), and most recently extended to multifluid gas dynamics problems. Implicit versions of the method have also been worked on [59-66] in collaboration with Bruce Fryxell and Karl-Heinz Winkler and, later, Wenlong Dai. A cell-based AMR version of PPM has been under development [see 67], in collaboration with Dennis Dinge. The PPM code has been implemented on many parallel computing systems, including some with thousands of processors [68-69; 54-55]. Versions of the PPM gas dynamics scheme have become incorporated into 3 community codes aimed at astrophysics applications: FLASH [refs], ENZO [refs], and VH-1 [refs].

Because the several improvements and modifications to the 1984 or 1986 versions of PPM in the literature have not been published, we here lay out the present scheme for simple, single-fluid, ideal gas dynamics from start to finish. Then selected results on homogeneous, isotropic, compressible turbulence in a cubical, periodic domain are briefly reviewed.

## 2. Design Constraints.

Before launching into a systematic description of the PPM algorithm, it is worthwhile to first explain the constraints that have influenced its design. These are:

1. Directional operator splitting.
2. Robustness for problems involving very strong shocks.
3. Contact discontinuity steepening.
4. Minimal dissipation.
5. Numerical errors dominated by dissipation, as opposed to dispersion.
6. Preservation of signals, if possible, even if their shapes are modified, so long as they travel at roughly the right speeds.
7. Minimal degradation of accuracy as the Courant number decreases toward 0.

The first of these design constraints was to guarantee very high processing speeds on all CPU designs. It has the side benefit of allowing the PPM scheme to employ complicated techniques to achieve high accuracy that would be impractical in a truly multi-dimensional implementation, especially in 3D. Accuracy is still limited in principle by the use of 1-D sweeps, but in practical problems errors produced by the use of 1-D sweeps have rarely, if ever, appeared to be important. The demand that the scheme be robust in the presence of very strong shocks has produced a general orientation that dissipation, in the right places, times, and amounts is good rather than bad. Early experience with Glimm's random choice scheme for multidimensional compressible gas dynamics in the 1970s illustrated the error of letting the formal dissipation of the scheme vanish for flows containing shocks. In low-speed turbulent flows, the

need for dissipation in the numerical scheme is less obvious but no less real. Contact discontinuity steepening is included in PPM in order to allow multifluid problems to be treated simply through the addition of passively advected constitutive properties, such as the constants of an analytic representation of the equation of state. The view of dissipation in PPM has already been mentioned. Naturally, a design goal has always been to minimize the dissipation consistent with an accurate solution. Nevertheless, it has also been a design goal that it is better to dissipate a signal than to significantly falsify its speed. This has led to an intolerance for signals with very short wavelengths, such as, for example, 4 cell widths. However, a further design goal was to preserve those signals that could be propagated accurately, even if this were to require a falsification of the signal shape. Such signals tend to move through the grid at the fluid velocity, since very short wavelength sound waves, with the exception of shocks, are difficult to propagate at the right speeds. The final design goal, that the accuracy of the scheme should be maintained in the limit of very many very small time steps is achieved through special features of the interpolation discussed below.

The attitude that has been taken in the PPM design toward numerical dissipation is the most relevant of these design constraints for the use of PPM in simulating turbulent flows. This point will be discussed after the scheme has been laid out, using simulations of decaying compressible turbulence as illustrations.

## 3. PPM Interpolation.

The heart of any numerical scheme is its approach to interpolation. PPM utilizes the cell averages of various quantities in a fairly broad stencil surrounding a cell in 3D in order to construct a picture of the internal structure of the cell. Because the equations of hydrodynamics are coupled, the PPM interpolation of the relevant quantities is also coupled. However, to see how the interpolation process works, it is easiest to consider it first for the simple case of a single variable, which we will represent by the symbol $a$. We will represent the cell averages of $a$ by $\langle a \rangle_i$. We write the value of $a$ at the left- and right-hand interfaces of the cells as $a_{L,\,i}$ and $a_{R,\,i}$. We will determine the coefficients of a parabola $a(\tilde{x}) = a_0 + a_1\tilde{x} + a_2\tilde{x}^2$ representing the distribution of the variable $a$ within the cell in terms of a cell-centered local coordinate $\tilde{x} = (x - x_M)/\Delta x$. Here $x_M$ is the cell center and $\Delta x$ is the cell width. For simplicity, we first assume a uniform grid. (In modern AMR codes, interpolation on non-uniform grids is unnecessary.) Our first task is to determine whether or not the function $a(x)$ is smooth in the region near our grid cell. To do this, we will essentially compare the first and third derivatives of the function in our cell. This process can be formulated in several different ways, all of which boil down to essentially the same criterion. We begin by determining the unique parabola that has the prescribed cell averages in our cell of interest and its two nearest neighbors.

If we write

$$\Delta a_{Li} = \langle a \rangle_i - \langle a \rangle_{i-1}$$

then we have

$$a_1 = \left(\Delta a_L + \Delta a_R\right)/2\,; \qquad a_2 = (\Delta a_R - \Delta a_L)/2$$

Here we have dropped the subscript $i$ by writing $\Delta a_R$ for $\Delta a_{L,\,i+1}$. Where possible, we will try to eliminate confusing subscripts by such devices. However, all our subscript indices will refer to cells, and none to interfaces (no half-indices will be used). If no subscript index is given, the subscript $i$ for our cell of interest will always be implied. In PPM, interpolated variables are discontinuous at interfaces, which makes the use of index subscripts such as $i+1/2$ ambiguous. Thus, $a_{R,i}$, or simply $a_R$, is not equivalent to $a_{L,\,i+1}$.

We will consider a function to be smooth near our grid cell if the above unique parabola, when extrapolated to the next nearest neighbor cells, gives a reasonable approximation to the behavior of the function there. This will not be true, of course, if the third derivative of the function is sufficiently large. We will consider the function to be smooth in our cell of interest if the parabola in its neighbor on the left, when extrapolated into our cell's neighbor on the right gives an accurate estimate of that cell's average, and if also that cell's parabola gives a good estimate of the average in our cell's neighbor on the left. Writing the average of the extrapolated parabola in the neighbor cell as $\langle a_i \rangle_{i\pm1}$ and in the next nearest neighbor cell as $\langle a_i \rangle_{i\pm2}$, we find that

$$\langle a_i \rangle_{i+1} - \langle a \rangle_{i+1} = 2\,(a_{2,\,i-1} - a_2)\,; \qquad \langle a \rangle_{i-1} - \langle a_i \rangle_{i-1} = 2\,(a_2 - a_{2,\,i+1})$$

Here we have used the fact that, by construction of these parabolae, $\langle a_i \rangle_{i\pm1} = \langle a \rangle_{i\pm1}$. We denote the fractional error in this extrapolation by $f_{err}$, given by

$$f_{err} = \max\left\{ \left| \left(a_{2,\,i+1} - a_2\right) / \left(\Delta a_R + sign(\alpha, \Delta a_R\right) \right|, \left| \left(a_2 - a_{2,\,i-1}\right) / \left(\Delta a_L + sign(\alpha, \Delta a_L\right) \right| \right\}$$

Here $sign$ is the Fortran sign transfer function, which applies the sign of its second argument to the absolute value of its first argument. We also denote a trivial value of the quantity $a$ by $\alpha$. Thus the term with the sign function is simply used to protect the divide operation in a Fortran program. Of course, we ignore this error estimate when this difference is very small, since in that case it is of no consequence whether we get a "good" extrapolation or not. To get some perspective on this measure, $f_{err}$, of the smoothness of the function $a$ on our grid, it is instructive to evaluate $f_{err}$ for sine waves of wavelengths $n\,\Delta x$. For the case where $a(x) = \sin(2\pi\, x/(n\,\Delta x))$, we have $\langle a \rangle = s\, a_0$, where $a_0$ is the value at the center of the cell and $s = (2/\Delta x)\sin(\Delta x/2)$. The higher-order coefficients of the interpolation parabola are given by $a_1 = s\,\sin(\Delta x)\cos(2\pi\, x_M/(n\,\Delta x))$, where $x_M$ is the coordinate of the cell center, and by $a_2 = s\,(\cos(\Delta x) - 1)\, a_0$. We therefore find that for values of $n$ ranging from 4 to 15, our fractional error, equal simply to $1 - \cos(\Delta x)$, assumes the values: 1.00, 0.69, 0.50, 0.38, 0.29, 0.23, 0.19, 0.16, 0.13, 0.11, 0.10, and 0.09. If we assert that a sine wave over 14 cells is smooth and one over 10 cells is not, then we may conclude that it is reasonable to assert that a function is smooth if $f_{err} \le 0.10$ and it is not smooth if $f_{err} \ge 0.20$. In between these two values, we may treat the function as partially smooth, by taking a linear combination of a smooth and an unsmooth interpolation parabola. The choice of which functions we will treat as smooth and which not comes from experience. Using only cell averages, it takes five cell values to determine, for example, whether one is really in a shock or not, as we will later see, or, as we have just seen, whether the function is locally smooth or not. Under such conditions, it is simply unreasonable to believe that one can properly treat a sine wave with a wavelength of only 6 cells. There are those in the community who believe this can be done, but PPM assumes this is impossible without further independent information provided upon which the numerical scheme can operate. Experience also shows that numerical noise, which can originate from a number of sources, some of which will be identified later, tends to show up principally at wavelengths between 4 and 8 cells. Signals that appear at these wavelengths are thus quite possibly noise, and therefore deserve to be treated as if they might actually be noise. PPM takes the view that a little noise is not bad, but too much is intolerable. This view is motivated by much experience indicating that elimination of noise is usually accompanied by significant collateral elimination of the signal.

In PPM, we interpolate several different functions. Some of them, like the Riemann invariants, are defined only as differences (the Riemann invariants are defined by inexact differentials). Other functions, like the transverse velocities, are defined by cell averages. Some may have sharp transitions, associated with contact discontinuities, that deserve special treatment. Below, we describe the most general algorithm, for the special case of a uniform grid. This is the algorithm that PPM uses to interpolate a sub-

grid structure for the Riemann invariant associated with the entropy, for which the differential form is given by

$$dA = d\rho - dp/c^2$$

From this most complicated interpolation algorithm, all other, simpler forms used in PPM can be derived as special cases. We will therefore describe the interpolation for the entropy Riemann invariant in detail, and later briefly note in which respects the other interpolations are degenerate cases of this one.

We begin the interpolation of $dA$ by defining a variable $A$ derived from it by determining the arbitrary constant of integration such that the cell average of $A$ vanishes. Thus each cell has its own local scaling for $A$ in which the cell average vanishes but the derivatives are correct. This local scaling is much like the local, cell-centered coordinate, $\tilde{x}$, that we use to define the interpolation parabola in a cell. There is a discontinuous jump in the integration constant when we go from one cell to its neighbor. This jump is

$$\Delta A_L = \Delta\rho_L - \Delta p_L/c_L^2 = \left(\langle\rho\rangle - \langle\rho\rangle_{i-1}\right) - \left(\langle p\rangle - \langle p\rangle_{i-1}\right)/c_L^2$$

For the gamma-law equation of state, we estimate $c_L^2$ via

$$c_L^2 = \frac{1}{2}\left[\left(\gamma\langle p\rangle_{i-1}/\langle\rho\rangle_{i-1}\right) + \left(\gamma\langle p\rangle/\langle\rho\rangle\right)\right]$$

Here we see a common practice used in PPM, namely the evaluation of an estimate of the cell average of a function of primitive fluid state variables as the function of the cell averages. Experience shows that attempts to do better than this are usually unrewarded by compensating increases in simulation accuracy. In the specific case of the kinetic energy, PPM does do a better job, but the effect on the flow accuracy is only marginal in this case, and the cost is not insignificant.

Using the above definitions, we evaluate a fractional error estimate as follows:

$$f_{err} = \frac{1}{2}\,\frac{\left|\Delta A_{L,\,i+2} - 2\,\Delta A_R + \Delta A_L\right| + \left|\Delta A_R - 2\,\Delta A_L + \Delta A_{L,\,i-1}\right|}{\left|\Delta A_L\right| + \left|\Delta A_R\right| + d\alpha}$$

Here $d\alpha$ represents a trivial Riemann invariant difference that is used to protect the divide in the Fortran program. We note that this error estimate is very similar to but not exactly the same as the one discussed earlier. Using this error estimate, we construct a measure, $\Omega$, varying between 0 and 1, of the roughness of the function near this grid cell as follows:

$$\Omega = \min\left\{1, \max\left\{0, 10\left(f_{err} - 0.1\right)\right\}\right\}$$

We now construct estimates $A_{s1}$ and $A_{m1}$ of the first-order parabola coefficient, which gives the change of the variable across the cell. The first, $A_{s1}$, applies to the parabola that has the 3 prescribed average values in this and the nearest neighbor cells. The second, $A_{m1}$, applies to this parabola after the application of a monotonicity constraint. These quantities will be used in constructing further quantities below. We have already seen that $A_{s1} = \left(\Delta A_L + \Delta A_R\right)/2$. Defining a sign variable, $s$, of absolute value unity to have the sign of $A_{s1}$, we may then compute $A_{m1}$ in the following sequence of steps:

$$\tilde{A}_{m1} = 2\,\min\left\{s\,\Delta A_L, s\,\Delta A_R\right\}, \qquad \tilde{\tilde{A}}_{m1} = s\,\max\left\{0, \min\left\{s\,A_{s1}, \tilde{A}_{m1}\right\}\right\}$$

$$A_{m1} = \Omega\,A_{s1} + \left(1-\Omega\right)\tilde{\tilde{A}}_{m1}$$

(The superiority of Fortran over mathematical notation for the description of numerical algorithms is here becoming apparent.) We now define two estimates, $A_{sL}$ and $A_{mL}$, of the value at the left-hand cell interface. The first of these, $A_{sL}$, corresponds to an unconstrained interpolation polynomial, while $A_{mL}$ corresponds to one that is constrained.

$$A_{sL} = \frac{1}{6}\left(A_{s1,\,i-1} - A_{s1}\right) - \frac{1}{2}\Delta A_L\,, \qquad A_{mL} = \frac{1}{6}\left(A_{m1,\,i-1} - A_{m1}\right) - \frac{1}{2}\Delta A_L$$

We note that these interface value estimates involve the cell averages in two cells on either side of the interface. The unconstrained estimate is the value of the unique cubic curve that has the prescribed average values in these 4 cells. The constrained value is guaranteed to lie within the range defined by the cell averages adjacent to the interface. Of course, both these values refer to the choice of integration constant that gives a vanishing cell average to the right of this interface (in the cell of interest, $i$ ). We desire a linear combination of these two interface value estimates, which we will regard as a provisional estimate $A_{pL} = (1-\Omega)A_{sL} + \Omega A_{mL}$. Rather than build this value $A_{pL}$ by blending the unconstrained and constrained estimates, we find it more useful to build blended parabola coefficients, $A_{p1} = (1-\Omega)A_{s1} + \Omega A_{m1}$, and to form $A_{pL}$ from these. This procedure allows us to make use of the coefficients $A_{p1}$ later in our contact discontinuity detection and steepening algorithm. We now proceed to modify our provisional interface values $A_{pL}$ in cells where the function is not smooth. First, we set our interpolation function to a constant in cells where extrema occur, unless of course the function is smooth there. Thus, if $A_{pL}\,A_{pR} \ge 0$, we set new provisional values, indicated by a subscript $q$, as follows: $A_{qL} = (1-\Omega)\,A_{pL}$, $A_{qR} = (1-\Omega)\,A_{pR}$. Otherwise we simply set $A_{qL} = A_{pL}$ and $A_{qL} = A_{pL}$. Now, in cells where the function is not smooth, we constrain the internal structure to be monotone. We must make sure in this process not to revise the cell structure further in the cells containing extrema. We therefore construct limiting values, $A_{lL}$ and $A_{lR}$, which we must be careful to set to 0 in cells containing extrema. In the remaining cells, these values are: $A_{lL} = -2A_{pR}$ and $A_{lR} = -2A_{pL}$. This reflects the fact that a parabola having zero slope at one side of the cell will assume at the opposite interface the negative of twice this value, so long as the cell average is zero. If these limiting values are exceeded in cells where the function is not smooth, we must reset them to these limits. Therefore we construct new, but still provisional, values $A_{rL}$ and $A_{rR}$ (in Fortran, of course, no new names are required), which reflect this additional constraint:

$$\textit{if}\quad \left(A_{qR} - A_{qL}\right)\left(A_{qL} - A_{lL}\right) < 0\,, \qquad \textit{then}\quad A_{rL} = (1-\Omega)A_{qL} + \Omega A_{lL}\,, \qquad \textit{otherwise}\quad A_{rL} = A_{qL}$$

$$\textit{if}\quad \left(A_{qR} - A_{qL}\right)\left(A_{lR} - A_{qR}\right) < 0\,, \qquad \textit{then}\quad A_{rR} = (1-\Omega)A_{qR} + \Omega A_{lR}\,, \qquad \textit{otherwise}\quad A_{rR} = A_{qR}$$

From these provisional interface values, we construct provisional interpolation parabola coefficients:

$$A_{r1} = A_{rR} - A_{rL}\,, \qquad A_{r2} = 3\left(A_{rL} + A_{rR}\right), \qquad A_{r0} = -A_{r2}/12$$

For interpolation of sound-wave Riemann invariant differences, for which we do no contact discontinuity steepening, the above values are our final results. However, for the entropy Riemann invariant differences, we continue as described below. We also note that interpolation of a variable, such as the transverse velocity component, which is not expressed as a differential form, we may first construct differences of the cell averages and then proceed as outlined above.

We wish to detect cells that are inside sharp jumps in the entropy Riemann invariant that are associated with contact discontinuities. In such cells, it is inappropriate to try to fit smooth curves to determine the subgrid structure, since it is actually discontinuous (PPM is solving the Euler equations). For the purpose of computation, we will nevertheless require that the distribution inside the grid cell be a

parabola; however, we may use the knowledge that the actual structure is discontinuous to construct an appropriate choice for this parabola. The idea here is quite simple. We build a test that detects cells that are within contact discontinuity structures. Then we obtain estimates of the interface values for such cells by extrapolating to the cell interfaces presumably smoother structures from outside the cell and hence, hopefully, from outside the discontinuity. Together with the prescribed cell averages, these more appropriate cell interface values allow us to build an improved, steeper parabola to describe the distribution within the cell. Obviously, the function in such cells is not smooth, so we apply monotonicity constraints to the new edge values and also to this improved parabola. This procedure works very well in practice. A similar procedure can be adapted for use with other numerical schemes that employ linear interpolation functions for the cell structures, but it is less effective. Apparently, the curvature provided by the use of parabolae allows the dimensionless constants that will be introduced below to make the steepening process effective for contact discontinuities without requiring that the steepening process be applied for marginal cases, which can occasionally turn out not to actually be contact discontinuities. That is, using parabolae we are able to apply the steepening process only in unequivocal circumstances and thus to avoid steepening structures by mistake that should not actually be steepened.

We begin, as with our test for function smoothness, with a measurement of the size of the third derivative of the function relative to its first derivative. Reusing the symbol $s$ for a different sign variable, we build a steepness measure $S$ in a sequence of steps as follows:

$$S_1 = \left( A_{sR} - A_{sL} - A_{s1} \right) / \left( A_{s1} + \ 10 \ s \ d\alpha \right)$$

Here $s$ has absolute value unity and has the same sign as $A_{s1}$ in order to protect the divide without altering the sign of the overall expression. As before, $d\alpha$ is a trivial Riemann invariant difference. We must take care to realize that the numerator in this expression is not necessarily zero, since the edge values that appear there come from cubic curves rather than the parabolae that define the $A_{s1}$ values. We must also take care to realize that because each cell has its own value of the integration constant for this Riemann invariant's differential form, we must set $A_{sR} = A_{sL,\, i+1} + \Delta A_R$. With those cautions, we see that this expression for $S_1$ is very similar to our measure of the lack of function smoothness, except that no absolute value signs appear in it. The first expression in the numerator for the variable difference across the cell involves cubic interpolation, which accounts for the third derivative of the function. From this we subtract $A_{s1}$, which removes the part that is due to the first 2 function derivatives. To obtain $S_1$, we divide by an estimate of the first derivative. The lack of absolute value operations in this formula for $S_1$ is caused by our desire to detect sharp jumps in the function, and to reject sudden flat spots. Flat spots will correspond to negative values of $S_1$. We now scale $S_1$ and limit it to the range from 0 to 1:

$$S_2 = \max\left\{0, \min\left\{1, \ 20 \left( S_1 - 0.05 \right)\right\}\right\}$$

To guard against applying our contact discontinuity steepening anywhere but at true sharp jumps in the entropy, we reset $S_2$ to zero where the second derivative of the function does not change sign and also where the amplitude of the jump is not sufficient to warrant this special treatment (we do not want to preserve, or worse, to amplify small numerical glitches). Thus:

$$\textit{if} \qquad A_{s2,\, i-1} \, A_{s2,\, i+1} \geq 0, \qquad \textit{then} \quad S_3 = 0, \qquad \textit{otherwise} \quad S_3 = S_2$$

Here we use the earlier definition of the coefficients of the parabola that has the prescribed 3 cell averages, writing $A_{s2} = \left( \Delta A_R - \Delta A_L \right) / 2$. To eliminate trivial jumps from consideration, we form $S_4$:

$$\textit{if} \qquad \left| \Delta A_L \right| + \left| \Delta A_R \right| < d\alpha, \qquad \textit{then} \quad S_4 = 0, \qquad \textit{otherwise} \quad S_4 = S_3$$

Before proceeding with the steepening algorithm, we make one further test to make certain that steepening is appropriate. This test makes sense for contact discontinuities, but for other variables, such

as a fractional volume of a second fluid, it might not. Therefore this last test is optional, but we do use it for the entropy. The calling program to the interpolation routine provides two variable differences along with the differences $\Delta A_L$. If the jump is one of the type we seek to detect, the differences $\Delta B$ will be very small compared with the differences $\Delta D$. Both these sets of differences are cell centered and have the same units. We enforce this demand by constructing our final steepness measure, $S$, as follows:

$$if \qquad \Delta D = 0 \;\; or \;\; \Delta B / \Delta D \geq 0.1, \qquad then \quad S = 0, \qquad otherwise \quad S = S_4$$

In previous versions of PPM, the $\Delta B$ were pressure differences and the $\Delta D$ were estimates of $c^2 \Delta \rho$, while the density was the quantity being interpolated with potential contact discontinuity steepening. We still perform this additional test, although it is somewhat redundant when interpolating the entropy Riemann invariant, for which the differential form is just $\Delta A = \Delta \rho - \Delta p / c^2$.

With our contact discontinuity detection process complete, and all cells inside such structures marked by $S$, varying from 0 to 1, we are at last ready to begin the steepening operation. Much earlier, we computed constrained variable differences $A_{m1}$ across the cells. In principle, these variable differences could be only partially constrained, but it is safe to assume that if our cell is in a contact discontinuity, the parameter $\Omega$ that measured the function roughness assumes the value unity in this and the neighboring cells. We define cell interface values, $A_{cL}$ and $A_{cR}$, appropriate for a contact discontinuity structure by extrapolating the constrained slopes of the function in neighboring cells to the interfaces of our cell of interest: $A_{cL} = -\Delta A_L + A_{m1,\, i-1}/2$ and $A_{cR} = \Delta A_R - A_{m1,\, i+1}/2$. We then construct our nearly final estimates for the interface values as:

$$A_{tL} = (1-S)\, A_{rL} + S\, A_{cL}\,, \qquad\qquad A_{tR} = (1-S)\, A_{rR} + S\, A_{cR}$$

We must once again apply the constraints on the implied parabola and then compute the coefficients of that parabola, just as before (however, hardly any cells are steepened, and hence this extra work occurs in a scalar loop for only these cells, which incurs hardly any additional computational cost). These operations will not be repeated here. They are just like those we performed to arrive at first $A_{qL}$ and then $A_{rL}$ beginning with $A_{pL}$.

It is worthwhile to note the ways in which the above interpolation algorithm addresses the previously listed design constraints for PPM. Some of the points are obvious, but others are less so. In particular, it is not obvious that dissipation errors will dominate dispersion errors, although this is in fact the case. The reason for this is the action of the monotonicity constraints on all short wavelength disturbances, which always fail our test for function smoothness. The monotonicity constraints are nonlinear, in that they alter the shape of a sine wave by introducing higher frequency components through effects such as the clipping of extrema. Their overall effect is strongly dissipative for very short wavelength signals, which of course are the ones for which the numerical scheme would otherwise introduce significant dispersion errors. This dominance of dissipation over dispersion error as a design goal may seem incompatible with the goal of minimal dissipation. Our task is to deliver as little dissipation as possible while still having dissipation overwhelm dispersion. PPM's use of parabolae rather than the more commonly used linear interpolation functions is meant to address this design goal. The use of cubic interpolation functions to help define these parabolae through interpolation of cell interface values reduces dissipation errors still further, since not all parabolae provide equally good fits. However, even more importantly this use of cubic interpolation in PPM preserves the accuracy of the scheme in the limit of vanishing Courant number. Since, as is common practice, we will determine a single time step value for the entire grid based upon Courant number limitations for the single most demanding cell, the bulk of the cells we update will have very small Courant numbers. When AMR is added to the scheme, the Courant numbers used for the bulk of the cells are likely to become even smaller. Therefore it is very important for the accuracy of the calculation to hold up in this limit. The use of cubic interpolation of cell interface values is our way of

addressing this issue in PPM. Over many years of applying PPM to a wide variety of flow problems, very little Courant number sensitivity has ever appeared.

The monotonicity constraints and contact discontinuity detection and steepening algorithms incorporated in PPM have important consequences for the propagation of barely resolved or effectively unresolved signals. Our design goal is to preserve these signals, so long as we can propagate them at roughly the proper speeds, rather than to destroy them through numerical diffusion processes. The monotonicity constraints and, to a far greater degree, the contact discontinuity steepening in PPM have this effect. They act upon passively advected signals, such as entropy variations, that can easily be propagated at the correct speeds. They apply an artificial compression along with their other effects, such as maintaining positivity where appropriate. This tends to maintain signal amplitude while altering signal shape when the signal is not adequately resolved. As the description of the contact discontinuity detection algorithm shows, we take great care to apply the steepening method only where appropriate, since some small signals, better described as numerical glitches of various types and causes, deserve to be dissipated.

## 4. Using the Interpolation Operators to Build a Subgrid-Scale Model for a Cell.

The above interpolation process is quite elaborate. Nevertheless, it is not the entire story of PPM interpolation. As we remarked earlier, the equations of gas dynamics are coupled, and therefore the PPM interpolations of the fluid state variables are also coupled. The goal is to come up with a complete and consistent picture of the internal structure of a grid cell – that is, to come up with a subgrid-scale model. This is not a turbulence model, but it *is* a subgrid-scale model. This model must satisfy, for consistency, certain primary constraints. First, the integrated mass, momenta, and total energy must be consistent with the prescribed cell averages of these quantities. This reflects the need for the numerical scheme to be in strict conservation form in order to correctly capture and propagate shocks. PPM regards the fundamental cell averages on which it operates to be the cell averages of density and pressure (both volume-weighted) and those of the 3 velocity components (all mass-weighted). From these, PPM defines the mass, momenta, and total energy of a cell to be: $\Delta m = \langle \rho \rangle \Delta x$, $\langle u_x \rangle \Delta m$, $\langle u_y \rangle \Delta m$, $\langle u_z \rangle \Delta m$, and $\frac{\langle p \rangle}{\gamma - 1} \Delta x + \frac{1}{2}\left( \langle u_x \rangle^2 + \langle u_y \rangle^2 + \langle u_z \rangle^2 \right) \Delta m$. Here we do not bother to include the cell widths in the transverse directions, $\Delta y$ and $\Delta z$, because they cancel out of all our equations for the x-pass. The last expression, for the cell's total energy, is misleading. At the beginning of the grid cell update for a 1-D pass, we compute the total energy of the cell in just this way. However, once we have done this, we make a more accurate estimate of the cell's kinetic energy and then revise its thermal energy, keeping this total constant. We do this by using nearest neighbor cells to compute cell-centered slope estimates for all 3 velocity components. We then apply the standard monotonicity constraints to these slopes. All these computations are similar to the following steps, for the slope of $u_y$ in the $z$ direction:

$$\Delta_z u_y = \frac{1}{2}\left( \langle u_y \rangle_{k+1} - \langle u_y \rangle_{k-1} \right), \qquad \left( \Delta_z u_y \right)_{\max} = 2 \min\left\{ s\left( \langle u_y \rangle_{k+1} - \langle u_y \rangle \right),\ s\left( \langle u_y \rangle - \langle u_y \rangle_{k-1} \right) \right\}$$

$$\Delta_{mz} u_y = s \max\left\{ 0,\ \min\left\{ s\,\Delta_z u_y,\ \left( \Delta_z u_y \right)_{\max} \right\} \right\}$$

Here $s$ has absolute value unity and the same sign as $\Delta_z u_y$. We now estimate the cell's average kinetic energy to be given by

$$
\begin{aligned}
2\left\langle E_{kin}\right\rangle &= \left\langle u_x\right\rangle^2 + \frac{1}{12}\left[\left(\Delta_{mx}u_x\right)^2 + \left(\Delta_{my}u_x\right)^2 + \left(\Delta_{mz}u_x\right)^2\right] \\
&+ \left\langle u_y\right\rangle^2 + \frac{1}{12}\left[\left(\Delta_{mx}u_y\right)^2 + \left(\Delta_{my}u_y\right)^2 + \left(\Delta_{mz}u_y\right)^2\right] \\
&+ \left\langle u_z\right\rangle^2 + \frac{1}{12}\left[\left(\Delta_{mx}u_z\right)^2 + \left(\Delta_{my}u_z\right)^2 + \left(\Delta_{mz}u_z\right)^2\right]
\end{aligned}
$$

This process captures the first correction to the average kinetic energy that arises from the internal structure of the velocity within the grid cell. The method would still be second-order accurate without this correction, but this correction proves to be of some marginal value (relative to its implementation cost) in flows involving strong shear layers. Terms of this sort make up part of some turbulence closure models, although one might argue that this is inappropriate, since they have nothing specifically to do with turbulence. In PPM we must compute estimates for cell averages of several other quantities, and including terms like these for all those computations would roughly double the cost of the scheme. This has been tried, and little noticeable benefit is delivered in practical problems. Therefore only these velocity terms are included in PPM. One can see why these terms are important as follows. The velocity slopes can be very large in a cell, because shear layers that are stretching due to the local flow or due to their own instability naturally become ever thinner. The contribution to the kinetic energy from the velocity slope terms can therefore be significant, and can thus have a significant effect upon the pressure (since the total energy is prescribed), and hence upon the dynamics. Nevertheless, this effect upon the dynamics is strongly localized, so that omitting the terms, as was done in PPM for many years, only tends to increase by roughly 50% the numerical friction in thin, strong (roughly sonic or supersonic) shear layers.

In order to propagate signals, we need to interpolate their subgrid structures. We therefore apply the previously described interpolation algorithm to the 5 Riemann invariants of the Euler equations for 3-D compressible flow. For the two transverse components of the velocity, $u_y$ and $u_z$, which are advected passively with the fluid velocity in the x-pass, we form differences and then apply the interpolation scheme described above. We do no contact discontinuity detection or steepening for these variables, even though they may in fact jump at such discontinuities. We have already discussed in detail the treatment for the entropy Riemann invariant. The sound wave Riemann invariants, which we denote by $R_\pm$, are defined by the following differential forms and their corresponding numerical approximations:

$$
\begin{aligned}
dR_\pm &= du_x \pm \frac{dp}{C} \\
\Delta R_{\pm L} &= \Delta u_{xL} \pm \frac{\Delta p_L}{C_L} = \left\langle u_x\right\rangle - \left\langle u_x\right\rangle_{i-1} \pm 2\left(\left\langle p\right\rangle - \left\langle p\right\rangle_{i-1}\right) / \left(\left\langle C\right\rangle + \left\langle C\right\rangle_{i-1}\right)
\end{aligned}
$$

Here we introduce the Lagrangian sound speed $C = \rho c$, and we approximate its cell average as $\left\langle C\right\rangle = \left\langle \rho\right\rangle\left\langle c\right\rangle = \sqrt{\gamma\left\langle p\right\rangle\left\langle \rho\right\rangle}$. The pressure, of course, is $p$, and only the x-component of velocity appears because we are describing the x-pass of the directionally split PPM algorithm. It is disturbances in $R_+$ that propagate to the right at speed $s_+ = u_x + c$ and disturbances in $R_-$ that propagate to the left at speed $s_- = u_x - c$. We interpolate parabolae to describe the structure of these sound wave signals in the grid cells, using $\Delta R_{\pm L}$ in place of $\Delta A_L$ in the algorithm previously described, and of course performing no contact discontinuity detection or steepening. Performing interpolations for the Riemann invariants attempts to uncouple the gas dynamic equations as much as possible. However, we will not work in terms of these variables directly. Instead, we will immediately go about constructing interpolation parabolae for the primary fluid state variables – the density, pressure, and 3 components of velocity – in a manner that attempts to achieve as much consistency as possible with the interpolated

structures of the Riemann invariant signals. To perform our hydrodynamical cell update, we will require internal cell structures for all these variables as well as the total energy. Any choice of structures for 5 variables will imply structures for all the others, but those implied structures may not satisfy reasonable constraints, such as monotonicity or positivity when appropriate. PPM attempts to achieve consistency between interpolated and implied structures by first interpolating constrained parabolae for the Riemann invariant signals, constructing the implied parabolae for the primary state variables, and then constraining those implied parabolae where appropriate. In a sense this is an attempt to "have it both ways," which is of course impossible. However, experience has shown that the extra labor involved in this process delivers additional accuracy and robustness of a value worthy of its computational and programming cost.

From the interpolated parabolae for the sound wave Riemann invariants, we may compute the cell interface values of the pressure and x-velocity as follows:

$$p_L = \langle p \rangle + C_L \left( R_{+L} - R_{-L} \right) / 2 \ , \qquad p_R = \langle p \rangle + C_R \left( R_{+R} - R_{-R} \right) / 2$$

$$u_{xL} = \langle u_x \rangle + \left( R_{+L} + R_{-L} \right) / 2 \ , \qquad u_{xR} = \langle u_x \rangle + \left( R_{+R} + R_{-R} \right) / 2$$

We also compute a measure, $\Omega$, of the lack of smoothness of the associated functions as the maximum of the values found for the two sound wave Riemann invariants separately during the process of their interpolation: $\Omega = \max \left\{ \Omega_{R_+}, \Omega_{R_-} \right\}$. We will use this measure $\Omega$ to control the application of constraints to our interpolation parabolae for $p$ and $u_x$. We first apply monotonicity constraints, controlled by $\Omega$, to the cell interface pressures and x-velocities obtained above. When $\Omega = 1$, we demand that the interface values lie within the ranges defined by the averages in adjacent cells. We then, again when $\Omega = 1$, demand that the parabolae defined by these constrained interface values and the cell averages are monotone. These constraints are essentially the same as those described earlier for the interpolation of the entropy Riemann invariant, and hence they will not be stated in detail here. Once the interpolation parabola for the pressure has been so derfined, we may use the interpolated interface values for the entropy to determine interface values for the density. Using the maximum of the $\Omega$ measures for the pressure and for the entropy, we may then constrain the interface densities and ultimately the parabolae that they and the cell averages define.

At the end of this lengthy process, we have interpolation parabolae defined for the density, pressure, and all 3 velocity components. The formulae presented assume uniform cell sizes, although more general formulae are easily derived and were presented in the description of PPM in [whatever]. We can think of these interpolations as occurring in a cell number variable, so that they are valid, although potentially somewhat less accurate, even if the cell size is smoothly varying. Nevertheless, we must take care in interpreting these parabolae. Some of the variables described by them, such as the velocity components, must be considered to vary according to mass fraction across each cell, since the (mass-weighted) cell average is associated directly with the conserved cell momentum. For the velocities, therefore, we take the interpolation variable $\tilde{x}$ within the cell to have its origin at the center of mass and to describe fractions of the cell mass rather than of the cell volume. For the pressure or the density, whose volume integrals are directly associated with the mass and internal energy, we must interpret $\tilde{x}$ as having its origin at the center of the cell in a volume coordinate and to describe volume fractions within the cell. Our use of the uniform grid formulae therefore reflects an assumption that all interpolated quantities are smoothly varying in a cell fraction variable, be it a volume or a mass fraction. We know this assumption to be false at contact discontinuities and slip surfaces (for some of the variables) and at shocks, but we have augmented our interpolation procedure to deal with these discontinuities. The interpolation is therefore valid, so long as we interpret it properly.

The process described above has enabled us to construct a subgrid-scale model for each cell. Since each fluid state variable varies as a simple parabola inside a cell, this subgrid-scale model cannot possibly describe subgrid-scale turbulent motions. To perform that function, the model would have to be augmented by the addition of one or more new state variables, such as a subgrid-scale turbulent kinetic energy variable, which could also be given an interpolation parabola within the cell. Nevertheless, our demand that the structure of a velocity component within a cell be no more complicated than a parabola,

and our constraints on that parabola, provide a powerful mechanism to dissipate kinetic energy of small-scale motions unresolvable or marginally resolvable on our grid into heat. This is one of the important functions of turbulence closure models, especially when incorporated into difference schemes that lack any other method to accomplish this very necessary dissipation. In this respect, turbulence models can be used to stabilize otherwise nonlinearly unstable numerical schemes, a role that has nothing to do with turbulence itself and that confuses the purpose and function of a turbulence model. As we will see later, with PPM we may compute turbulent flow directly from the Euler equations that govern it, and we accomplish the dissipation of small-scale motions through the truncation errors of our numerical scheme rather than from differencing an explicit viscous diffusion term. We take the view that the details of this dissipation are unimportant so long as kinetic energy turning up on the smallest possible scales in our computation is dissipated into heat with no unphysical side effects. We presume that in a far more expensive and careful simulation of all relevant physics, this same kinetic energy would be dissipated in any event, and the same amount of heat generated (since total energy is conserved), but the (hopefully unimportant) details on tiny length and time scales would differ. The assumption here is that the kinetic energy that appears on scales where the numerical dissipation comes into play has arrived there due to a physical process, such as a turbulent cascade, that is correctly simulated in our calculation because it has nothing to do with molecular viscosity.

## 5. The PPM Approximate Riemann Solver

We will update the cell averages in time by applying the laws of conservation of mass, momentum, and total energy. To do this, for a 1-D x-pass of the algorithm, we must compute time-averaged fluxes at the left- and right-hand cell interfaces of these 5 conserved quantities. This is done with the help of an approximate Riemann solver. A fundamental approximation that we make is that the solution to the Riemann problem for left and right states representing spatial averages over the appropriate domains of dependence is roughly equivalent to the average in time of the many separate Riemann problem solutions that each express the interaction of the specific Riemann invariant values arriving at the interface at that particular time. The Riemann solver is a nonlinear operator, and we assume that its output when operating on the averaged inputs is equivalent to the average of the outputs from the separate, time-varying inputs. This assumption makes the Riemann solver relatively simple and efficient. Our elaborate construction of subgrid-scale structures allows us to easily arrive at the spatial averages in the appropriate domains of dependence. In the linear regime, when we have simple advection of the Riemann invariants, we get the full advantage of or interpolation parabolae for the Riemann invariant signals. Experience shows that when variations in $R_{\pm}$ are large, shocks rapidly develop, and these demand techniques only loosely related to concepts of formal order of accuracy.

Here we will discuss how we handle the domain of dependence for the Riemann invariant $R_+$. The treatment for $R_-$ is similar, and will not be described. We will treat the characteristic speed $s_+ = u + c$ to be constant both in space and time within each cell for this single time step. Accounting for the space and time variation of $s_+$ is demanded only if we desire formal third-order accuracy. We will focus our discussion on the left-hand interface of our cell of interest. We calculate a Courant number, $\sigma_{+L}$, which applies to one or the other of the adjacent cells, depending upon the signs of $s_+$ in them:

$$\sigma_{+L} = s_{+,i-1}\,\Delta t/\Delta x, \quad \textit{if} \quad s_{+,i-1} \geq 0; \quad \textit{otherwise} \quad \sigma_{+L} = -s_+\,\Delta t/\Delta x, \quad \textit{if} \quad s_+ \leq 0$$

We note that the two possibilities listed above are not exhaustive. In the case where $s_+$ is directed away from the interface on both sides, we will have a centered rarefaction there. We will handle this case later. For the moment it will be sufficient to set $\sigma_{+L}$ to zero in this rare event. The Courant condition demands that we control the time step $\Delta t$ so that $\sigma_{+L}$ does not exceed unity. We now estimate the spatial averages of the density and pressure, $\langle \rho \rangle_{+L}$ and $\langle p \rangle_{+L}$, in the domain of dependence via:

$$\langle\rho\rangle_{+L} = \rho_{R,i-1} - \frac{\sigma_{+L}}{2}\left[\rho_{1,i-1} + \left(1-\frac{2\sigma_{+L}}{3}\right)\rho_{2,i-1}\right], \qquad if \quad s_{+,i-1} \geq 0;$$

$$\langle\rho\rangle_{+L} = \rho_L + \frac{\sigma_{+L}}{2}\left[\rho_1 - \left(1-\frac{2\sigma_{+L}}{3}\right)\rho_2\right], \quad otherwise.$$

We obtain $\langle p\rangle_{+L}$ using similar formulae. From $\langle\rho\rangle_{+L}$ we may calculate a mass fraction Courant number, $\sigma_{m+L}$. When $s_{+,i-1} \geq 0$, we have $\sigma_{m+L} = \sigma_{+L}\langle\rho\rangle_{+L} / \langle\rho\rangle_{i-1}$, and otherwise we have $\sigma_{m+L} = \sigma_{+L}\langle\rho\rangle_{+L} / \langle\rho\rangle$. We can then obtain $\langle u_x\rangle_{+L}$ using the same formulae as for the density or pressure, but with $\sigma_{+L}$ replaced by $\sigma_{m+L}$. Note that we choose the upwind domain of dependence in the case when $R_+$ Riemann invariant signals reach the interface from both adjacent cells. This can happen if the interface is inside a shock structure, in which case the $R_+$ signals from the cell on the right will become lost in the shock transition. Hence we take the signals from the cell on the left in this case, which will correspond to the proper post-shock state. For the $R_-$ signals, we prefer signals from the cell on the right, if such signals can reach the interface from both adjacent cells. In the case when $s_{+,i-1} < 0$ and also $s_+ > 0$, the left-hand interface of our cell is located inside a centered rarefaction fan. We desire Riemann invariant information corresponding to a vanishing characteristic speed. We interpolate $\langle\rho\rangle_{+L}$, $\langle p\rangle_{+L}$, and $\langle u_x\rangle_{+L}$ linearly between the two edge values, as in: $\langle p\rangle_{+L} = f_{+L}\, p_{R,i-1} + (1-f_{+L})\, p_L$, where $f_{+L} = s_+ / (s_+ - s_{+,i-1})$.

A simple, linearized solution to the Riemann problem at the cell interface can be obtained from the demands that the time-averaged state at the interface have the same values of $R_\pm$ as the domains of dependence which we have just determined. At this point we desire only the time-averaged pressure and x-velocity at the interface, which we denote by $\overline{p}_L$ and $\overline{u}_{xL}$. Using the differential forms that define the Riemann invariants, we thus have: $\overline{u}_{xL} - \langle u_x\rangle_{\pm L} \pm (\overline{p}_L - \langle p\rangle_{\pm L})/C_L = 0$. Therefore:

$$\overline{u}_{xL} = \frac{1}{2}\left(\langle u_x\rangle_{+L} + \langle u_x\rangle_{-L}\right) + \frac{1}{C_L}\left(\langle p\rangle_{+L} - \langle p\rangle_{-L}\right)$$

$$\overline{p}_L = \langle p\rangle_{-L} + C_L\left(\overline{u}_{xL} - \langle u_x\rangle_{-L}\right)$$

Because this is a numerical scheme, and anything can happen, we subsequently make sure that $\overline{p}_L$ is at least as big as some trivial, floor value. This simple, linear Riemann solver suffices for almost all the cell interfaces, but it is not good enough for interfaces that are in strong shock structures. For such cells, which are very small in number, we can do much better. However, first we must determine which cells are in shock structures. In PPM, we go to considerable lengths to identify these cells, and we will use the information we glean in the process to build a "smart" diffusion velocity that we will use to greatly diminish noise that can be generated at shocks in numerical computations.

Experience has shown that shock detection must be done using information from all 3 dimensions. This fact means that the difference stencil of the PPM 1-D pass extends into the transverse dimensions. This explicit acknowledgement that the 1-D pass is actually part of a multi-D computation allows PPM to avoid a number of numerical pathologies that would otherwise occur. Several of these have been discussed in the original paper laying out the design philosophy and major techniques of PPM [W&C]. (These pathologies were dubbed "Cray instabilities," since at the time one needed to have access to a Cray-1 computer in order to afford grids fine enough to observe them.) As with the detection of function

roughness and sudden jumps associated with contact discontinuities, detection of shocks requires the inspection of data from 2 cells on either side of the cell of interest. For the case of shock detection, we use 2 cells on either side in each of the 3 dimensions. Taking differences over 4 cell widths allows us to get a good estimate of the entire shock transition, and hence of its propagation speed. We will need to know the propagation speed relative to the grid as well as the shock orientation relative to the grid in order to detect those situations which require diffusion at the shock, and hence to be able to construct our smart diffusion velocity. Our approach is to first compute in vectorizable loops quantities that allow us to eliminate most candidate cells, and then to do the more involved computations determining shock orientation and propagation speed only for the very few cells that require this. We will demand that to be considered as within a shock structure a cell must be in a region of compression and in a sudden pressure jump of at least 25%. These demands will allow us to eliminate from consideration the great bulk of our cells.

First, in each of the 3 dimensions, we evaluate the jump between the next nearest neighbor cell averages of the pressure on either side of our cell. In this process, we also determine which of these next nearest neighbor cells has the lower pressure value. We then find the largest of these pressure jumps and divide it by the lower pressure value in that dimension. For our cell to be considered inside a shock, this ratio must exceed 0.25, although this particular value is not critical to the successful operation of the scheme (a value of 0.33, for example, works well too). The choice of 0.25 indicates a Mach number, which one could derive, for a shock below which no special treatment by PPM is necessary. Setting this value too low is harmless, but it would cause the scheme to devote considerable unnecessary labor to weak disturbances. Setting the value too high will let small, incorrect signals be generated at some shocks under pathological circumstances. The choice 0.25 has served well for decades in a wide variety of problems. In the same vectorizable loop, we evaluate simple difference approximations to the divergence of the velocity. One of these uses differences between velocity component averages in nearest neighbors to our cell and the other uses differences of such averages in next nearest neighbors. If either of these estimates of the velocity divergence is positive, we eliminate our cell as a candidate for being inside a shock structure. We mark our candidate cells, those with negative velocity divergence estimates and with 25% or greater pressure jumps, with a flag vector that we call $shocked$. This flag array vanishes everywhere but in our candidate cells, where it has the value unity.

For the cells marked by $shocked$, we now embark upon a complicated calculation described below. First we use averages in next nearest neighbor cells to evaluate the contributions from each dimension to the velocity divergence estimate. We will use weight factors, $f_x$, $f_y$, and $f_z$, for each dimension according to the relative size of these contributions. Because we desire no negative weight factors, we define the contributions to the velocity divergence as $\Delta_{2y}u_y = \min\left\{0, \left(\left\langle u_y \right\rangle_{j+2} - \left\langle u_y \right\rangle_{j-2}\right)\right\}$, with similar formulae for the $x$ and $z$ dimensions. We then define the weight factors as follows:

$$f_y = \frac{\left(\Delta_{2y}u_y\right)^2}{\left(\Delta_{2x}u_x\right)^2 + \left(\Delta_{2y}u_y\right)^2 + \left(\Delta_{2z}u_z\right)^2}, \qquad f_z = \frac{\left(\Delta_{2z}u_z\right)^2}{\left(\Delta_{2x}u_x\right)^2 + \left(\Delta_{2y}u_y\right)^2 + \left(\Delta_{2z}u_z\right)^2}$$

We define the remaining weight factor by subtraction: $f_x = 1 - f_y - f_z$. Looking at values in next nearest cells in each dimension, we select the pre- and post-shock cells in each. We then blend these 3 pre- and 3 post-shock cell averages using the above weight factors to obtain estimates of the pre- and post-shock densities, pressures, and 3 velocity components. Using the jumps across the shock in the 3 velocity components computed from these blended values, we get the x-, y-, and z-components of a unit vector pointing in the direction of the velocity jump, hence in the direction normal to the shock front. We now project the pre- and post-shock velocities onto this normal direction by taking the dot products of those velocities with the unit vector. We also compute the pre- and post-shock Lagrangian sound speeds. The nonlinear Lagrangian wave speed, $W$, for the shock is computed via:

$$\widetilde{W} = \sqrt{\left|p_{post} - p_{pre}\right| \;/\; \left(\left|V_{pre} - V_{post}\right| \;+\; 0.00000001\; V_{post}\right)}$$

This formula clearly involves some defenses against potential pathologies. In multidimensional flows next to walls where Mach reflections of shocks can occur, some rather wild cases can arise to which our formulae must be insensitive. The tilde in the formula indicates that this is a provisional value. To force this wave speed estimate to be reasonable, we additionally execute the following constraints, resulting ultimately in our final estimate, $W$:

$$\tilde{\tilde{W}} = \min\left\{\tilde{W}, \max\left\{C_{pre}, C_{post}\right\}\right\}, \qquad W = \max\left\{\tilde{\tilde{W}}, \min\left\{C_{pre}, C_{post}\right\}\right\}$$

We now compute the Eulerian shock speed, $w$, by using the post-shock density and normal velocity:

$$w = W / \rho_{post} + u_{\perp post}$$

One might wonder why we need all this information about the shock. We will use it to compute a diffusion speed that will determine the amount of additional dissipation that we introduce here. As discussed in the original articles on PPM [W&C, C&W, W], shock representations on our grid tend to become too thin when the shocks move slowly relative to the grid. Such a shock can linger near a grid line, and then the action of the focusing characteristics can cause the shock jump to occur mostly across that single cell interface. As the shock moves so that it becomes centered on a grid cell instead, it is forced to have a quite different and broader numerical representation. The slow oscillation of the shock structure from thicker to thinner numerical representation causes small disturbances to be emitted in all the characteristic families that cross the shock (i.e. in our case in the entropy and in the oppositely propagating sound wave Riemann invariants). This happens in 1-D problems, but in 2- or 3-D problems we can get glitches caused by oversteepening of the shock structure as it crosses grid lines obliquely. At each grazing intersection, a tiny shock and contact discontinuity pair can be emitted. The solution to all these problems is to broaden the numerical shock representation so that it is roughly independent of phase relative to the grid. The glitches and noise emission can be reduced dramatically by only a modest broadening of the shock. We do this by means of a diffusion operation that will be described later. The diffusion coefficient, which we evaluate as a diffusion velocity, must relate to the potential severity of the numerical problems for a given shock in a given cell. We find that the diffusion coefficient needs to reflect two independent factors – the shock strength and the wavelength of the fundamental noise signals that it tends to emit. Our diffusion velocity, which we denote by $u_{diff}$, has a factor equal to the negative of the velocity divergence in order to reflect the strength of the shock. It also has a factor arising from the fundamental wavelength of noise emission. The fundamental period for noise emission by the shock is the interval between grid line crossings. The wavelength of any such signals, measured in grid cell widths, is then dependent upon the Riemann invariant family involved and the post-shock fluid velocity and sound speeds. The longest fundamental wavelength of signal emission, measured in grid cell widths, will apply to the sound wave Riemann invariant, and it can be estimated as follows:

$$\lambda_{noise} = \frac{\left(W / \rho_{post}\right) + c_{post}}{\left|\left(W / \rho_{post}\right) + u_{\perp post}\right| + 0.00000001\ c_{pre}}$$

When the shock moves rapidly across the grid, this wavelength is very short, and any noise signals are immediately damped by the numerical scheme, if they can propagate at all. It is only as this wavelength exceeds 2 that we need become concerned. We therefore compute our diffusion velocity in the following steps:

$$\tilde{u}_{diff} = 0.3\ (shockd) \left|\left\langle u_x\right\rangle_{i+2} - \left\langle u_x\right\rangle_{i-2} + \left\langle u_y\right\rangle_{j+2} - \left\langle u_y\right\rangle_{j-2} + \left\langle u_z\right\rangle_{k+2} - \left\langle u_z\right\rangle_{k-2}\right|$$

$$\Theta = \max\left\{0, \left(\lambda_{noise} - 2\right)\right\}, \qquad \Xi = \Theta^3 / \left(\Theta^3 + 1\right), \qquad u_{diff} = 0.1\ \left(1 + 9\Xi\right)\ \tilde{u}_{diff}$$

Some aspects of these formulae may seem a bit arbitrary, and no doubt other variations might work equally well. Nevertheless, the above formulation is quite serviceable, and has kept numerical glitches and noise at bay in PPM simulations for at least 15 years. Changing the 0.3 to 0.5 in the formula for

$\widetilde{u}_{diff}$ will roughly double the thicknesses of shocks, almost completely eliminate numerical noise, and reduce the overall quality of the solution by an amount comparable to coarsening the grid by a factor of 2 in every spatial dimension and time. Setting this constant to 0.1 will cause the noise to appear and to mar the solution. The recommended value of 0.3 is thus a compromise, giving excellent results with thin shock structures and crisply resolved details but allowing very low level signals of numerical origin into the simulation at levels almost always below about 1.5% of the amplitude of the relevant, emitting shock jump. The smart diffusion described here, and applied at the end of the grid cell update, supersedes all previous formulations in the literature on PPM. Earlier formulations went to significant lengths to avoid the use of any construct that could directly be interpreted as an artificial viscosity. This was a consequence of a bet made with Bill Noh at Livermore. Since his death many years ago, there has been no satisfaction from doing without explicit diffusion, and the present formulation, in use for about 15 years, is easily made compatible with other code packages.

Now that we have identified the cells inside shock structures and computed the diffusion velocities to be used there, we proceed, in a scalar loop for only these cells, to apply a nonlinear Riemann solver to compute the fluxes at their interfaces. A fundamental approximation we will make is to use the shock formulae for pressure and velocity differences across rarefaction fans. For weak rarefactions, this approximation is easy to justify. Strong rarefactions at single cell interfaces are always transient phenomena, with the exception of centered rarefactions attached to features in walls or solid objects. We take the view that it is impossible to do a good job for these exceptional cases in any event, save to refine the grid there using AMR or other techniques. We will describe the nonlinear Riemann solver for gamma-law gases, but a generalization of it to arbitrary equations of state has been given in [W86]. We will perform the following operations only at those interfaces for which one of the adjacent cells is marked by the flag array $shocked$. We will do the first step of what could be used as an iteration. Experience has shown that there is essentially no value in performing additional steps of such an iteration. The use of a nonlinear Riemann solver allows us to compute a portion of the entropy jump in the shock from shock jump conditions rather than obtaining it from a diffusion process that smears the shock structure. However, the shock jump conditions that apply at a single cell interface are essentially never the proper ones for the shock as a whole, and as a result it is not worthwhile to take them too seriously. A single nonlinear iteration therefore suffices. We begin by computing positive-definite Lagrangian wave speeds in the gas immediately to the left, $W_{LL}$, and to the right, $W_{RL}$, of the contact discontinuity that initially forms at the interface $L$. To improve our estimate, $\overline{p}_L$, of the pressure at this contact discontinuity, we also evaluate the slopes, $(\partial p / \partial u_x)_{LL}$ and $(\partial p / \partial u_x)_{RL}$, of the tangent lines at the pressure $\overline{p}_L$ to the Hugoniot curves in the velocity-pressure plane:

$$f_{LL} = (\gamma - 1) \left\langle p \right\rangle_{+L} + (\gamma + 1) \overline{p}_L , \qquad f_{RL} = (\gamma - 1) \left\langle p \right\rangle_{-L} + (\gamma + 1) \overline{p}_L$$

$$W_{LL} = \sqrt{f_{LL} \left\langle \rho \right\rangle_{+L} / 2} , \qquad W_{RL} = \sqrt{f_{RL} \left\langle \rho \right\rangle_{-L} / 2}$$

$$(\partial p / \partial u_x)_{LL} = \frac{f_{LL} W_{LL}}{f_{LL} - \frac{1}{2}(\gamma + 1)\left(\overline{p}_L - \left\langle p \right\rangle_{+L}\right)} , \qquad (\partial p / \partial u_x)_{RL} = \frac{f_{RL} W_{RL}}{f_{RL} - \frac{1}{2}(\gamma + 1)\left(\overline{p}_L - \left\langle p \right\rangle_{-L}\right)}$$

We now compute the point of intersection of the two tangents to the Hugoniot curves, which gives us an improved estimate, $\overline{p}_{shkL}$, for $\overline{p}_L$. The tangent lines pass through the points $(\overline{u}_{xLL}, \overline{p}_L)$ and $(\overline{u}_{xRL}, \overline{p}_L)$ in the $u_x - p$ plane. They intersect at the point $(\overline{u}_{x\,shkL}, \overline{p}_{shkL})$.

$$\overline{u}_{xLL} = \left\langle u_x \right\rangle_{+L} - \left(\overline{p}_L - \left\langle p \right\rangle_{+L}\right) / W_{LL} , \qquad \overline{u}_{xRL} = \left\langle u_x \right\rangle_{-L} + \left(\overline{p}_L - \left\langle p \right\rangle_{-L}\right) / W_{RL}$$

$$\Psi = \frac{\left(\overline{u}_{xRL} - \overline{u}_{xLL}\right)\left(\partial p / \partial u_x\right)_{RL}}{\left(\partial p / \partial u_x\right)_{LL} + \left(\partial p / \partial u_x\right)_{RL}}$$

$$\overline{p}_{shkL} = \max\left\{\phi, \left[\overline{p}_L - \Psi\left(\partial p/\partial u_x\right)_{LL}\right]\right\}, \qquad \overline{u}_{xshkL} = \overline{u}_{xLL} + \Psi$$

We must now take care to handle slowly moving strong shocks properly. From the above results for the pressure and x-velocity at the contact discontinuity in our Riemann problem, we can compute the Eulerian wave speeds, $w_{LL}$ and $w_{RL}$, for both the leftward and rightward facing waves:

$$w_{LL} = \left\langle u_x \right\rangle_{+L} - W_{LL}/\left\langle \rho \right\rangle_{+L}, \qquad w_{RL} = \left\langle u_x \right\rangle_{-L} + W_{RL}/\left\langle \rho \right\rangle_{-L}$$

Using this information, we now select from the 3 possibilities our improved estimates for $\overline{u}_{xL}$ and $\overline{p}_L$, which we will adorn with primes:

$$\textit{if} \quad \overline{u}_{xshkL} < 0 \quad \textit{and} \quad w_{RL} < 0, \quad \textit{then} \qquad \overline{u}'_{xL} = \left\langle u_x \right\rangle_{-L}, \qquad \overline{p}'_L = \left\langle p \right\rangle_{-L}$$

$$\textit{if} \quad \overline{u}_{xshkL} > 0 \quad \textit{and} \quad w_{LL} > 0, \quad \textit{then} \qquad \overline{u}'_{xL} = \left\langle u_x \right\rangle_{+L}, \qquad \overline{p}'_L = \left\langle p \right\rangle_{+L}$$

$$\textit{otherwise} \qquad \overline{u}'_{xL} = \overline{u}_{xshkL}, \qquad \overline{p}'_L = \overline{p}_{shkL}$$

Getting good estimates of the nonlinear wave speeds is crucial here, although in principle the fluxes of mass, momentum, and total energy on either side of a stationary strong shock are identical.

Up to this point, we have worked only with the time-averaged x-velocities and pressures at the cell interfaces. Now that we know the time-averaged interface velocities, we can trace back along the fluid streamlines from the interfaces into the adjoining cells in order to estimate what fraction of which cell crosses the interface during the time step. We may also estimate the time-averaged density at the interface by taking the spatial average at the beginning of the time step of this material that crosses the interface and compressing or expanding it *using the shock jump conditions* to the time-averaged pressure at the interface, which has just been computed. In the absence of shocks, this compression or expansion should be adiabatic, but if a strong shock is involved, the behavior could be extremely different. Using the shock jump conditions will give us an acceptable result in either case. We begin by computing the spatial averages, $\left\langle \rho \right\rangle_{0L}$, $\left\langle p \right\rangle_{0L}$, $\left\langle u_y \right\rangle_{0L}$, and $\left\langle u_z \right\rangle_{0L}$, in essentially the same way that we computed similar spatial averages earlier over the domains of dependence of the $R_{\pm}$ Riemann invariant characteristics. Here we use the subscript 0 to denote the streamline characteristics. The time-averaged density at the cell interface is now:

$$\overline{\rho}_L = \left\langle \rho \right\rangle_{0L} \frac{(\gamma+1)\,\overline{p}'_L + (\gamma-1)\left\langle p \right\rangle_{0L}}{(\gamma-1)\,\overline{p}'_L + (\gamma+1)\left\langle p \right\rangle_{0L}}$$

## 6. Construction of the Time-Averaged Interface Fluxes and Application of the Conservation Laws.

Now that we have evaluated the time averages of the fluid state variables at the cell interfaces, it is a simple matter to construct from them the time-averaged fluxes of the conserved quantities. At each interface, we compute an advected volume, $dx_L$, an advected mass, $dm_L$, 3 advected momenta, $d\mu_{xL}$, $d\mu_{xL}$, $d\mu_{xL}$, and an advected total energy, $dE_L$. These are all signed quantities, with positive signs applying to advection to the right:

$$dx_L = \overline{u}'_{xL}\,\Delta t, \qquad dm_L = \overline{\rho}_L\,dx_L, \qquad d\mu_{xL} = \overline{u}'_{xL}\,dm_L + \overline{p}'_L\,\Delta t, \qquad d\mu_{yL} = \overline{u}_{yL}\,dm_L$$

$$d\mu_{zL} = \overline{u}_{zL}\,dm_L, \qquad dE_L = \frac{1}{2}\left[\left(\overline{u}'_{xL}\right)^2 + \left(\overline{u}'_{xL}\right)^2 + \left(\overline{u}'_{xL}\right)^2\right]dm_L + \frac{\gamma}{\gamma-1}\,\overline{p}'_L\,dx_L$$

To guard against the production of negative densities, we also demand that $dm_L$ not exceed advection of 95% of the upstream cell's mass. Our time step controls should prevent this situation, but this constraint provides additional code robustness.

We now apply the conservation laws for mass, momentum, and total energy in order to arrive at provisional new values for the cell averages at the end of the time step for this x-pass. We use the subscript $N$ to denote the new time level and the superscript $(1)$ to denote this first, provisional value. Despite our application of monotonicity constraints it is still possible in this Eulerian treatment for the new cell mass to be negative. We therefore demand that the new cell mass not fall below a floor value, given by $\rho_{\min}\Delta x$. The value of $\rho_{\min}$ is of course problem dependent, and it must be supplied by the user for any given run, along with trivial values for other positive definite quantities such as pressure and energy, $p_{\min}$ and $E_{\min}$. To prevent problems of negative cell masses from occurring, we will compute a cell mass Courant number that we will use to help control the time step. This time step control is the Eulerian equivalent of the one used in Lagrangian calculations to prevent the tangling of grid cells.

$$\Delta m_N^{(1)} = \max\left\{\left(\Delta m + dm_L - dm_R\right), \rho_{\min}\Delta x\right\}, \qquad \left\langle \rho \right\rangle_N^{(1)} = \Delta m_N^{(1)} / \Delta x$$

The cell mass Courant number we use is $0.2\left(\Delta m - \Delta m_N^{(1)}\right) / \Delta m_N^{(1)}$, and we use a similar cell volume Courant number equal to $0.2\left(\overline{u}'_{xL} - \overline{u}'_{xR}\right)\Delta t / \left[\Delta x - \left(\overline{u}'_{xL} - \overline{u}'_{xR}\right)\Delta t\right]$. These Courant numbers assert that we do not want either the mass of the Eulerian cell or the volume of its Lagrangian counterpart to decrease by more than 80% in a single time step. We usually find that if the Courant number exceeds unity on any time step, even though we have taken measures to see that no disasters, such as square roots of negative numbers, will ensue, it is best to have the computation halve the time step and try again. This feature is built into our PPM code, including its massively parallel versions, and occurs automatically, although, gratefully, very rarely.

The provisional new cell velocities are now obtained from the momentum conservation laws:

$$\left\langle u_x \right\rangle_N^{(1)} = \left(\left\langle u_x \right\rangle \Delta m + d\mu_{xL} - d\mu_{xR}\right) / \Delta m_N^{(1)}$$

with similar equations for $\left\langle u_y \right\rangle_N^{(1)}$ and $\left\langle u_z \right\rangle_N^{(1)}$. Although it is optional, we usually choose to reset trivial cell interface and cell average velocities to zero. This prevents the generation of underflows, and it can allow us in some problems to avoid doing any computational labor in cells where nothing is happening yet. From these new velocities, we obtain the provisional new cell kinetic energy, $\left\langle E_{kin} \right\rangle_N^{(1)}$.

$$\left\langle E_{kin} \right\rangle_N^{(1)} = \frac{1}{2}\left[\left(\left\langle u_x \right\rangle_N^{(1)}\right)^2 + \left(\left\langle u_y \right\rangle_N^{(1)}\right)^2 + \left(\left\langle u_z \right\rangle_N^{(1)}\right)^2\right]$$

Note that at this point we make only a simple estimate of the cell average of the kinetic energy. It would make the scheme a great deal more complex and expensive to attempt to determine the new internal cell velocity structures at this stage of the calculation. That will be done at the outset of the next 1-D pass, where the total energy in the cell will first be computed according to the above prescription, so that the total energy conservation law, stated below, will be obeyed.

$$\left\langle E \right\rangle_N^{(1)} = \left(\left\langle E \right\rangle \Delta m + dE_L - dE_R\right) / \Delta m_N^{(1)}, \qquad \left\langle \varepsilon \right\rangle_N^{(1)} = \max\left\{E_{\min}, \left\langle E \right\rangle_N^{(1)} - \left\langle E_{kin} \right\rangle_N^{(1)}\right\}$$

Here we use the symbol $\varepsilon$ to represent the internal energy. We now compute the new provisional cell pressure from the gamma-law equation of state:

$$\left\langle p \right\rangle_N^{(1)} = (\gamma - 1)\left\langle \rho \right\rangle_N^{(1)} \left\langle \varepsilon \right\rangle_N^{(1)}$$

We now have a complete set of provisional new cell averages. These will actually be our final results for the overwhelming majority of the cells. However, near slowly moving strong shocks we still need to apply our smart diffusion.

## 7. Smart Diffusion and Reapplication of the Conservation Laws.

The final step of the PPM 1-D pass is applied only to those cells with interfaces for which a smart diffusion velocity was computed earlier. There are very few of these cells, and therefore the computational cost of this final step is small. However, this step increases the numerical difference stencil by one cell on either side of our cell of interest. Hence when the scheme is implemented on multiprocessor or cluster systems, the principal cost of this additional step of the algorithm is the increased quantity of data that must be communicated from one processor to another. We reiterate that the smart diffusion is multidimensional. The detection of shocks does not favor any single dimension, such as the dimension of the present 1-D pass, and the diffusion is applied in all directional passes, not just those in which the shock jump is seen as sudden. This multidimensional character of the smart shock dissipation in PPM avoids multiple numerical pathologies, and the required size of the difference stencil in the 3 dimensions is therefore a small price to pay for this benefit. We begin by computing diffusive fluxes at those cell interfaces adjacent to cells in which the diffusion velocity, $u_{diff}$, does not vanish. We also compute a diffusion Courant number equal to double the advected cell fraction for this diffusion step.

$$u_{diffL} = \max\left\{u_{diff,\,i-1},\, u_{diff}\right\}, \qquad dm_{diffLL} = \Delta t\, u_{diffL}\, \langle \rho \rangle^{(1)}_{N,\,i-1}, \qquad dm_{diffRL} = \Delta t\, u_{diffL}\, \langle \rho \rangle^{(1)}_{N}$$

$$dm_{diffL} = dm_{diffLL} - dm_{diffRL}, \qquad d\mu_{xdiffL} = dm_{diffLL} \langle u_x \rangle^{(1)}_{N,\,i-1} - dm_{diffRL} \langle u_x \rangle^{(1)}_{N}$$

$$d\mu_{ydiffL} = dm_{diffLL} \langle u_y \rangle^{(1)}_{N,\,i-1} - dm_{diffRL} \langle u_y \rangle^{(1)}_{N}, \qquad d\mu_{zdiffL} = dm_{diffLL} \langle u_z \rangle^{(1)}_{N,\,i-1} - dm_{diffRL} \langle u_z \rangle^{(1)}_{N}$$

$$dE_{diffL} = dm_{diffLL} \langle E \rangle^{(1)}_{N,\,i-1} - dm_{diffRL} \langle E \rangle^{(1)}_{N}$$

With these fluxes, for the affected cells only, we now go through the same steps described earlier using the non-diffusive fluxes, applying the conservation laws to obtain the new cell averages: $\langle \rho \rangle_N$, $\langle p \rangle_N$, $\langle u_x \rangle_N$, $\langle u_y \rangle_N$, and $\langle u_z \rangle_N$. This completes the x-pass of the PPM single-fluid gas dynamics algorithm.

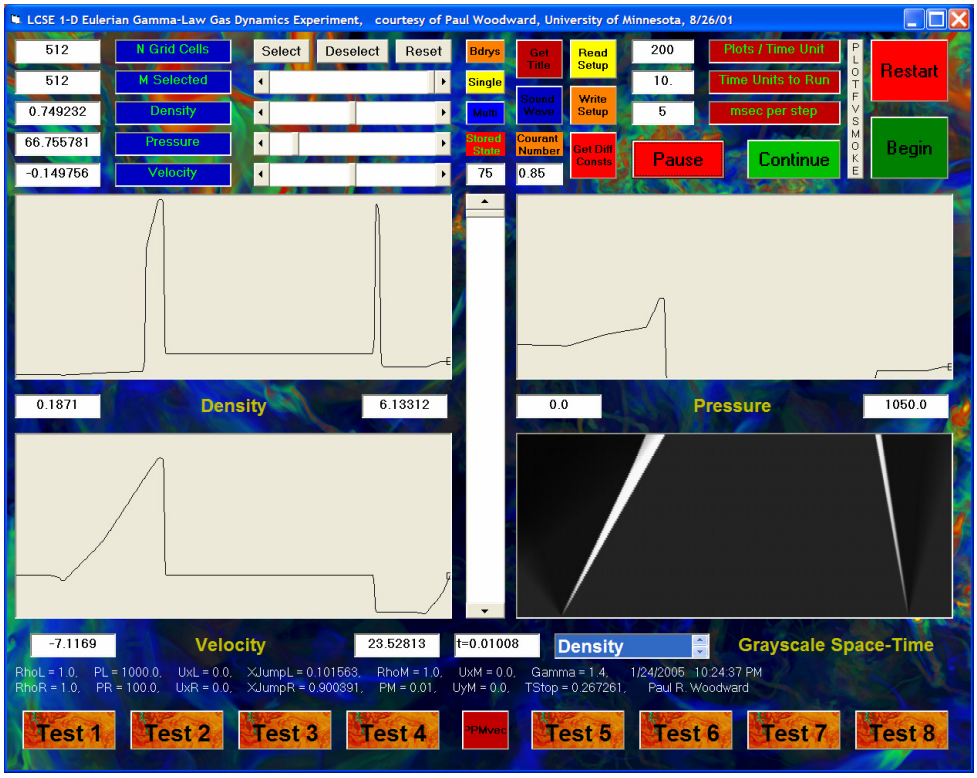
The full 3-D algorithm consists of six, symmetrized 1-D sweeps, in an x-y-z-z-y-x sequence and with a constant value of the time step.

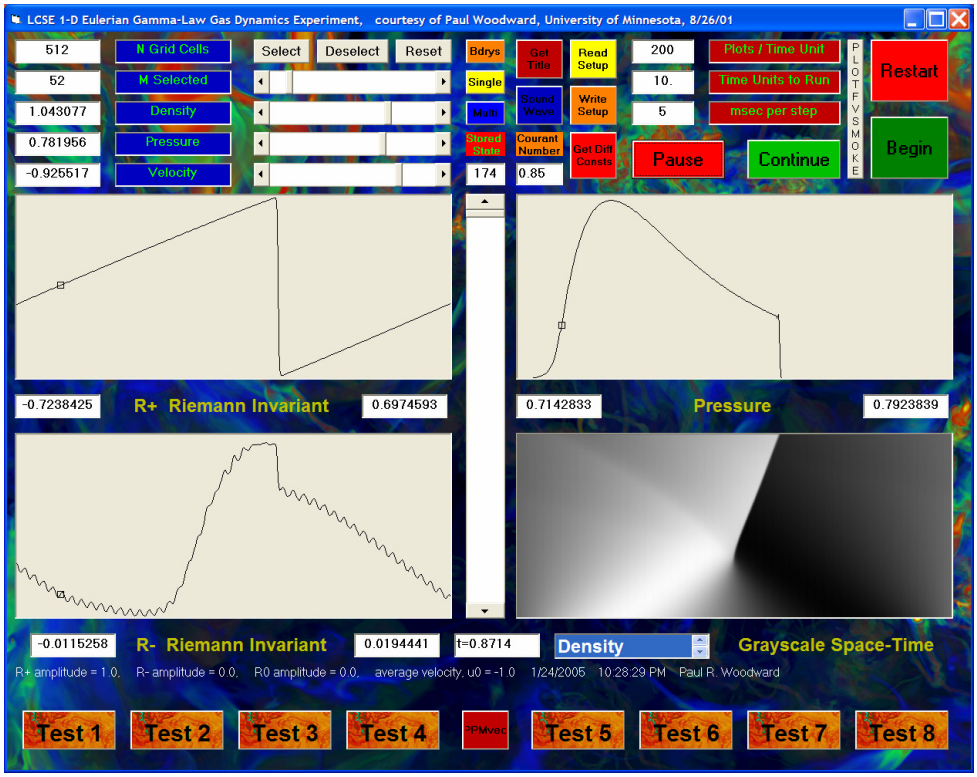
## 8. PPM Code Performance and Scalability on Modern Multiprocessing Systems.

PPM follows sound wave signals explicitly, so that all its computations, save the calculation of the time step value, are local. The scheme is very computationally intensive, because it expends so much effort to ensure that the various special operations such as monotonicity constraints, contact discontinuity steepening, and special treatment of strong shocks are applied only when and where they should be. On all CPUs from the Cray-1 to the modern Intel-based laptop machine or ASCI supercomputer, PPM has achieved excellent performance. This is because almost all of the arithmetic can be performed in vector mode, and the code has always been built to perform the entire algorithm for each strip of grid cells at once, thoroughly exploiting modern cache-based memory systems. On multiprocessor systems, each CPU updates an entire 3-D grid brick, augmented by enough extra grid cells to make either 3 or 6 1-D passes possible without any further information. During this brick update, the results of the previous brick update are written back to global memory and a new brick of data is fetched. Thus, even when the global memory is implemented on disks, all CPUs are always kept busy. The above description has been long and involved, and it may therefore seem that an undue amount of computation is required by the PPM scheme. Hence it is useful to quantify just how much computation is actually involved. On a difficult flow problem involving many strong shocks, on the average each cell update for a 1-D pass involves 921 flops

and 249 vectorizable logical operations (which do not count as flops). This work is performed at a speed of 1255 Mflop/s on a 1.7 GHz Intel Pentium-M CPU working from its cache memory in a laptop machine. This laptop performance degrades to 954 Mflop/s for a full $128^3$ 3-D grid brick update, so that full $128^3$ test runs can be performed overnight at a hotel. These speeds are for 32-bit arithmetic, which is all that PPM ever requires. On a 3.2 GHz Intel Pentium-4 CPU, these 32-bit rates become 1607 Mflop/s and 1273 Mflop/s, respectively, while the corresponding rates for 64-bit arithmetic, which is unnecessary but quoted for comparison purposes with other applications, are 1132 Mflop/s and 938 Mflop/s, respectively. PPM performance has scaled essentially linearly on every multiprocessing machine on which the code has ever been implemented, including ASCI machines with over 6000 CPUs.

Below, we review some results from applying the PPM gas dynamics scheme to compressible turbulent flow problems. However, one can download a Windows PC version of the code from the LCSE Web site at the University of Minnesota and try it out for oneself. From links on the main page at www.lcse.umn.edu one can download 1-D, 2-D, and 3-D test versions to play with that have been used in teaching and code development at the University of Minnesota. User guides are even online for the 1-D and 2-D versions. All versions online can perform either single fluid or 2-fluid simulations. A library of compiled PPM code modules, PPMLIB, is also available from the LCSE Web site, although these modules are at present not compiled for the latest machines. These codes, distributed in binary form from the LCSE Web site, are designed to run a variety of test problems that have proven useful in our code development at the LCSE. The 1-D and 2-D codes run, among other things, test problems introduced in the original PPM papers in 1980 and 1984. The 3-D code runs a 3-D shear layer instability problem. Try them; they're fun. Versions of PPM are incorporated in the community codes FLASH, ENZO, and VH-1 for the more serious user.

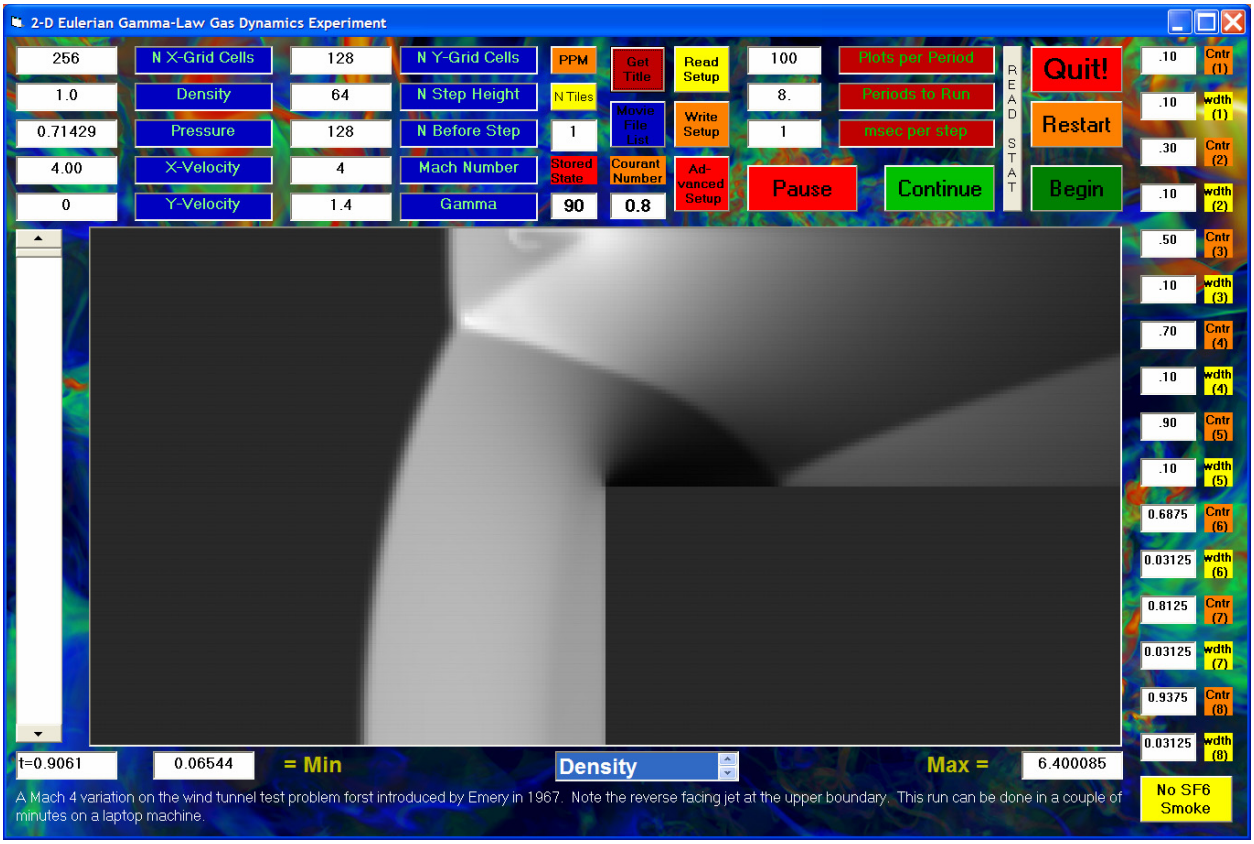



At the left we see the user interface to the 1D PPM gas dynamics test program available at the LCSE Web site. It is set up to perform by default the dual blast wave problem from the early PPM comparison with other difference schemes in 1984. The space-time wave diagram at the bottom right can be enlarged to fill the entire window area by double clicking on it. Display in this window of any of a whole list of variables may be generated.

Below at the left is this same application, but the user has clicked on the "Sound Wave" button to set up a sound wave steepening test problem. For this type of problem, the Riemann invariants are shown in the plotting windows, since these variables are the most sensitive indicators of shock-emitted noise, some of which can be seen at very low amplitude in the lower left-hand plotting window. This tiny disturbance is not noticeable at all in the space-time plot. Runs of this type were used to help determine the optimal values of the dimensionless constants in the PPM smart shock diffusion algorithm described in the text.

The buttons at the bottom of the user interface form set up a series of test problems from a comparison of gas dynamics schemes carried out by Burton Wendroff at Los Alamos




The user interface to the 2-D PPM gas dynamics code is shown here in the process of computing a type of wind tunnel test problem originally introduced by Emery in 1967. This test code can perform multifluid variations on this problem, and it can do single-fluid computations that inject smoke streams, set using the little text windows at the right of the form, and track them in order to visualize the flow. Any of a large number of possible displays can be selected using the list-box at the bottom center of the form.

9. References.


1. van Leer, B., *J. Comput. Phys.*, **32**, 101 (1979).
2. van Leer, B., and P. R. Woodward, "The MUSCL Code for Compressible Flow: Philosophy and Results," Proc. of the TICOM Conf., March, 1979.
3. van Leer, B., and P. R. Woodward, Proc. Internat. Conf. Comput. Meth. Nonlinear Mech., edited by J. T. Oden (Amsterdam:North-Holland), p. 234.
4. Woodward, P. R., and P. Colella, "High-Resolution Difference Schemes for Compressible Gas Dynamics," *Lecture Notes in Phys.* **141**, 434 (1981).
5. Woodward, P. R., and P. Colella, "The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks," *J. Comput. Phys.* **54**, 115-173 (1984).
6. Colella, P., and P. R. Woodward, "The Piecewise-Parabolic Method (PPM) for Gas Dynamical Simulations," *J. Comput. Phys.* **54**, 174-201 (1984).
7. Woodward, P. R., "Numerical Methods for Astrophysicists," in *Astrophysical Radiation Hydrodynamics*, eds. K.-H. Winkler and M. L. Norman, Reidel, 1986, pp. 245-326.
8. Wenlong Dai and P. R. Woodward, "An Approximate Riemann Solver for Ideal Magnetohydrodynamics," *Journal Computational Physics* **111**, 354-372 (1994).

9. Wenlong Dai and P. R. Woodward, "Extension of the Piecewise-Parabolic Method to Multidimensional Ideal Magnetohydrodynamics," *Journal of Computational Physics* **115**, 485-514 (1994).
10. Wenlong Dai and P. R. Woodward, "Structures of Reconnection Layers based on the Ideal Magnetohydrodynamic Equations," *J. Plasma Physics* **51**, 381-398, (1994).
11. Wenlong Dai and P. R. Woodward, "Interactions between Magnetohydrodynamical Shocks and Denser Clouds," *Astrophysical Journal* **436**, 776-783 (1994).
12. Wenlong Dai and P. R. Woodward, "Interactions between Magnetohydrodynamical Discontinuities," *Phys. Plasmas* **1**, 3662 (1994).
13. W. Dai and P. R. Woodward, "A Simple Riemann Solver and High-Order Godunov Schemes for Hyperbolic Systems of Conservation Laws," *Journal of Computational Physics* **121**, 51 (1995).
14. Dai, Wenlong, and P. R. Woodward, "A High-Order Godunov Scheme for Ideal Magnetohydrodynamics Equations Based on a Nonlinear Riemann Solver," in *Proceedings of the Fifth International Conf. On Hyperbolic Problems: Theory, Numerics, Applications*, pp. 293-299, eds. J. Glimm, M. J. Graham, J. W. Grove, and B. J. Plohr, World Scientific (1996).
15. Wenlong Dai and P. R. Woodward, "A High-Order Godunov-Type Scheme for Shock Interactions in Ideal Magnetohydrodynamics," *SIAM Journal on Scientific Computing* **18**, 957-981 (1997).
16. Dai, Wenlong, and P. R. Woodward, "A Simple Finite Difference Scheme for Multidimensional Magnetohydrodynamical Equations," *Journal of Computational Physics* **142**, 331-369 (1998).
17. Dai, Wenlong, and P. R. Woodward, "On the Divergence-Free Condition and Conservation Laws in Numerical Simulations for Supersonic Magnetohydrodynamical Flows," *Astrophysical J.*, **493** (1998).
18. Wenlong Dai and P. R. Woodward, "Second Order Godunov Schemes for 2D and 3D MHD Equations and Divergence-Free Condition," in *Barriers and Challenges in Computational Fluid Dynamics*, eds. V. Venkatakrishnan, M. D. Salas, and S. R. Chakravarthy, Kluwer Academic Publishers, pp. 283-297 (1998).
19. Woodward, P. R., "Simulation of the Kelvin-Helmholtz Instability of a Supersonic Slip Surface with the Piecewise-Parabolic Method (PPM)," in *Numerical Methods for the Euler Equations of Fluid Dynamics*, eds. Angrand, Dervieux, Desideri, and Glowinski, SIAM, 1985.
20. Woodward, P. R., "Simulations of Supersonic Jet Instability using the Piecewise-Parabolic Method (PPM)," in *High Speed Computing, Scientific Applications and Algorithm Design*, R. B. Wilhelmson, editor, Univ. of Illinois Press, 1988.
21. Woodward, P. R., D. H. Porter, M. Ondrechen, J. Pedelty, K.-H. Winkler, J. W. Chalmers, S. W. Hodson, and N. J. Zabusky, "Simulations of Unstable Fluid Flow Using the Piecewise-Parabolic Method (PPM)," in *Science and Engineering on Cray Supercomputers*, published by Cray Research, Inc., Minneapolis, 1987.
22. Winkler, K.-H., J. W. Chalmers, S. W. Hodson, P. R. Woodward, and N. J. Zabusky, "A Numerical Laboratory," *Physics Today*, Oct., 1987.
23. Pedelty, J. A., and P. R. Woodward, "Numerical Simulations of the Nonlinear Kink Modes in Linearly Stable Supersonic Slip Surfaces," *J. Fluid Mech.*, **225**, 101-120 (1991).
24. Bassett, G. M., and P. R. Woodward, "Numerical Simulation of Nonlinear Kink Instabilities of Supersonic Shear Layers," *Journal of Fluid Mechanics* **284**, 323-340 (Feb. 10, 1995).
25. Bassett, G. M., and P. R. Woodward, "Simulation of the Instability of Mach 2 and Mach 4 Gaseous Jets in 2 and 3 Dimensions," *Astrophysical Journal* **441**, (March 10, 1995).
26. Edgar, B. K., "," Ph.D. thesis, University of Minnesota.
27. Edgar, B. K., and P. R. Woodward, "Diffraction of a Shock Wave by a Wedge: Comparison of PPM Simulations with Experiment," AIAA Paper 91-0696, and *Video Journal of Engineering Research*, **3**, 25-33+cover illustration (1993).

28. Edgar, B. K., Woodward, P. R., and Anderson, S. E., PPM code library, with documentation and examples, available at *www.lcse.umn.edu/PPMlib*.
29. Porter, D. H., A. Pouquet, and P. R. Woodward, "Supersonic Homogeneous Turbulence," *Lecture Notes in Physics*, **392**, 105-125 (1991).
30. Porter, D. H., A. Pouquet, and P. R. Woodward, "A Numerical Study of Supersonic Turbulence," *Theoretical and Computational Fluid Dynamics*, **4**, 13-49 (1992).
31. Porter, D. H., A. Pouquet, and P. R. Woodward, "Three-Dimensional Supersonic Homogeneous Turbulence: A Numerical Study," *Phys. Rev. Lett.*, **68**, 3156-3159 (1992).
32. Porter, D. H., A. Pouquet, and P. R. Woodward, "Kolmogorov-Like Spectra in Decaying Three-Dimensional Supersonic Flows," *Phys. Fluids A*, **6**, 2133-2142 (1994).
33. Woodward, P. R., "Superfine Grids for Turbulent Flows," *IEEE Computational Science & Engineering*, Vol. **1**, No. 4, pp. 4-5+cover illustration, (December, 1994).
34. Woodward, P. R., D. H. Porter, B. K. Edgar, S. E. Anderson, and G. Bassett, "Parallel Computation of Turbulent Fluid Flow," Proc. Parallel CFD '94 Conference, Kyoto, Japan, May, 1994; published in *Computing Applications Mathematics*, Vol. **14**, no. 1, pp 97-105 (1995).
35. Porter, D. H., A. Pouquet, and P. R. Woodward, "Compressible Flows and Vortex Stretching," in *Small-Scale Structures in Three-Dimensional Hydrodynamic and MHD Turbulence*, ed. A. Pouquet, M. Meniguzi, & P-L. Sulem, Springer Verlag, p. 51-58 (1995).
36. Dannevik, W. P., A. M. Dimits, R. H. Cohen, D. Eliason, O. Schilling, D. H. Porter and P. R. Woodward, "Three-Dimensional Compressible Rayleigh-Taylor Simulation on the ASCI Blue-Pacific ID System," VHS Video (1996), LLNL Report UCRL-MI-125449.
37. Mirin, A. A., R. H. Cohen, W. P. Dannevik, A. M. Dimits, D. E. Eliason, D. H. Porter, O. Schilling and P. R. Woodward, "Three-Dimensional Simulations of Compressible Turbulence on High-Performance Computing Systems," Eighth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis (1997), LLNL Report UCRL-JC-125949.
38. Schilling, O., R. H. Cohen, W. P. Dannevik, A. M. Dimits, D. E. Eliason, A. A. Mirin, D. H. Porter and P. R. Woodward, "Three-Dimensional High-Resolution Simulations of Compressible Rayleigh-Taylor Instability and Turbulent Mixing," Sixth Int'l. Workshop on the Physics of Compressible Turbulent Mixing, Marseille, France (1997), LLNL Report UCRL-JC-125308.
39. Cohen, R. H., W. P. Dannevik, A. M. Dimits, D. E. Eliason, A. A. Mirin, D. H. Porter, O. Schilling, and P. R. Woodward, "Three-dimensional high-resolution simulations of the Richtmyer-Meshkov mixing and shock-turbulence interaction," in Proc. Of the 6th International Workshop on the Physics of Compressible Turbulent Mixing, Marseille, France, 18-21 June, 1997, pp. 128-133. LLNL Report UCRL-JC-125309.
40. Cohen, R. H., A. M. Dimits, A. A. Mirin, W. P. Dannevik, R. G. Eastman, D. E. Eliason, O. Schilling, D. H. Porter and P. R. Woodward, "Three-Dimensional PPM Simulations of Re-shocked Richtmyer-Meshkov Instability," VHS Video (1997), LLNL Report UCRL-MI-128783.
41. Woodward, P. R., D. H. Porter, B. K. Edgar, W. Dai, G. Bassett, S. E. Anderson, "Application of Monotonicity-Preserving Godunov Schemes to Compressible Flow Simulation," Proceedings of the Godunov Conference, Univ. of Michigan, Ann Arbor, May, 1997.
42. Porter, D. H., P. R. Woodward, and A. Pouquet, "Inertial Range Structures in Decaying Turbulent Flows," *Physics of Fluids* **10**, 237-245 (1998).
43. Porter, D. H., A. Pouquet, and P. R. Woodward, "Intermittency in Compressible Flows," Proc. Seventh European Turbulence Conference, St. Jean C. F., France, July, 1998, ed. Uriel Frisch.
44. Mirin, A. A., R. H. Cohen, B. C. Curtis, W. P. Dannevik, A. M. Dimits, M. A. Duchaineau, D. E. Eliason, D. R. Schikore, S. E. Anderson, D. H. Porter, P. R. Woodward, L. J. Shieh, and S. W. White, "Very High Resolution Simulation of Compressible Turbulence on the IBM-SP System," the IEEE's 1999 Gordon Bell Award was conferred in the performance category for this work; also available as Lawrence Livermore National Laboratory Report UCRL-MI-134237, July, 1999.

45. Porter, D. H., A. Pouquet, I. Sytine, and P. R. Woodward, "Turbulence in Compressible Flows," *Physica A* **263**, 263-270 (1999).

46. Sytine, I. V., D. H. Porter, P. R. Woodward, S. W. Hodson, and K.-H. Winkler 2000, "Convergence Tests for Piecewise Parabolic Method and Navier-Stokes Solutions for Homogeneous Compressible Turbulence," *J. Comput. Phys.*, **158**, 225-238 (2000).

47. Cohen, R. H., W. P. Dannevik, A. M. Dimits, D. E. Eliason, A. A. Mirin, Y. Zhou, D. H. Porter, and P. R. Woodward, "Three-Dimensional Simulation of a Richtmyer-Meshkov Instability with a Two-Scale Initial Perturbation," *Physics of Fluids*, **14**, 3692-3709 (2002); also available as LLNL Preprint UCRL-JC-144836 (2000).

48. Vetter, M., and Sturtevant, B., 1995. "Experiments on the Richtmyer-Meshkov Instability of an Air/SF6 Interface," *Shock Waves*, **4**, 247-252.

49. Porter, D. H., Pouquet, A., and Woodward, P. R., "Measures of Intermittency in Driven Supersonic Flows," *Pysical Review E* **66**, 026301 (2002).

50. Porter, D. H., and P. R., Woodward, "3-D Simulations of Turbulent Compressible Convection," *Astrophysical Journal Suppl.* **127**, 159-187 (2000); available at *www.lcse.umn.edu/conv3d*.

51. Porter, D. H., P. R. Woodward, and M. L. Jacobs, "Convection in Slab and Spheroidal Geometries," Proc. 14th International Florida Workshop in Nonlinear Astronomy and Physics: Astrophysical Turbulence and Convection, Univ. of Florida, Feb., 1999; in *Annals of the New York Academy of Sciences* **898**, 1-20 (2000); available at *www.lcse.umn.edu/convsph*.

52. Woodward, P. R., D. H. Porter, I. Sytine, S. E. Anderson, A. A. Mirin, B. C. Curtis, R. H. Cohen, W. P. Dannevik, A. M. Dimits, D. E. Eliason, K.-H. Winkler, and S. W. Hodson, "Very High Resolution Simulations of Compressible, Turbulent Flows," in *Computational Fluid Dynamics,* Proc. of the 4th UNAM Supercomputing Conference, Mexico City, June, 2000, edited by E. Ramos, G. Cisneros, R. Fernández-Flores, A. Santillan-González, World Scientific (2001); available at *www.lcse.umn.edu/mexico*.

53. Woodward, P. R., Porter, D. H., and Jacobs, M., "3-D Simulations of Turbulent, Compressible Stellar Convection," Proc. 3-D Stellar Evolution Workshop, Univ. of Calif. Davis I.G.P.P., July, 2002; also available at *www.lcse.umn.edu/3Dstars*.

54. Woodward, P. R., Anderson, S. E., Porter, D. H., and Iyer, A., "Cluster Computing in the SHMOD Framework on the NSF TeraGrid," LCSE internal report, April, 2004, available on the Web at *www.lcse.umn.edu/turb2048*.

55. Porter, D. H., P. R. Woodward, and A. Iyer, "Initial experiences with grid-based volume visualization of fluid flow simulations on PC clusters," accepted for publ. in Proc. Visualization and Data Analysis 2005 (VDA2005), San Jose, CA, Jan., 2005.

56. LCSE movies from a variety of turbulent fluid flow simulations can be downloaded and/or viewed at *www.lcse.umn.edu/MOVIES*.

57. Yokokawa, M., Itakura, K., Uno, A., Ishihara, T., and Kaneda, Y., "16.4 Tflops Direct Numerical Simulation of Turbulence by a Fourier Spectral Method on the Earth Simulator," Proc. Supercomputing 2002, Baltimore, Nov., 2002.

58. http://www.lcse.umn.edu and links from this main page.

59. Fryxell, B., P. R. Woodward, P. Colella, and K.-H. Winkler, "An Implicit-Explicit Extension of the PPM Scheme for Lagrangian Hydrodynamics in One Dimension," *J. Comput. Phys.* **63**, 283 (1986).

60. Dai, Wenlong, and P. R. Woodward, "A High-Order Iterative Implicit-Explicit Hybrid Scheme for 2-D Hydrodynamics," in *Numerical Methods for Fluid Dynamics V*, pp. 377-383, eds. K. W. Morton and M. J. Baines, Oxford Science Publications (1995).

61. Dai, Wenlong, and P. R. Woodward, "Iterative Implementation of an Implicit-Explicit Hybrid Scheme for Hydrodynamics," *Journal of Computational Physics* **124**, 217-229 (1996).

62. Dai, Wenlong, and P. R. Woodward, "A Second-Order Iterative Implicit-Explicit Hybrid Scheme for Hyperbolic Systems of Conservation Laws," *Journal of Computational Physics* **128**, 181-196 (1996).

63. Dai, Wenlong, and P. R. Woodward, "Numerical Simulations for Nonlinear Heat Transfer in a System of Multimaterials," *Journal of Computational Physics* **139**, 58-78 (1998).

64. Dai, Wenlong, and P. R. Woodward, "Numerical Simulations for Radiation Hydrodynamics. I. Diffusion Limit," *Journal of Computational Physics* **141**, 182-207 (1998).

65. Dai, Wenlong, and P. R. Woodward, "Numerical Simulations for Radiation Hydrodynamics II. Transport Limit," *Journal of Computational Physics* **157**, 199-233 (2000).

66. Dai, Wenlong, and P. R. Woodward, "Implicit-Explicit Hybrid Schemes for Radiation Hydrodynamics Suitable for Distributed Computer Systems," in Parallel Computational Fluid Dynamics, Elsevier, Eds. Keyes, Ecer, Satofuka, Fox, and Periaux, p. 179-187 (2000).

67. Dinge, D., and P. R. Woodward, "A Parallel Cell-by-Cell AMR Method for the PPM Hydrodynamics Code," *International J. Modern Phys. C*, **14**, 1-24 (2003).

68. Anderson, S. E., P. R. Woodward, and D. H. Porter 1995, "A Simplified 3-D PPM Implementation for the Clustered SMP Architecture, a Parallel Programming Template," 3-D sPPM template code, and associated materials, available at *www.lcse.umn.edu/research/-sppm* (the LCSE Web site).

69. Woodward, P. R., and S. E. Anderson, "Portable Petaflop/s Programming: Applying Distributed Computing Methodology to the Grid Within a Single Machine Room," Proc. of the 8th IEEE International Conference on High Performance Distributed Computing, Redondo Beach, Calif., Aug., 1999; available at *www.lcse.umn.edu/HPDC8*.