PPB Moment-Conserving Advection for Multifluid Hydrodynamics

Paul R. Woodward
University of Minnesota

October 9, 2002

A family of moment-conserving advection schemes in 1-D was introduced by van Leer in the mid 1970s. Of these, MUSCL advection is the best known. That advection scheme represents the advected function, which we will call a, by a linear function within each grid cell. The linear function is determined by the values of the first two moments of this density distribution within the cell, namely the mass and the center of mass of the cell. Both these moments are updated by the scheme in a process that is equivalent to projecting the detailed advected function after each time step onto the subspace of piecewise linear functions (using a weighting function that treats every point within each cell as equally important). In this sense, the MUSCL advection scheme is similar to a Galerkin finite element method (without the constraint that the resulting representation of the function should be continuous). Schemes of this type have much later been called discontinuous Galerkin methods, with Godunov's method and MUSCL the first two schemes introduced of this type. MUSCL is also essentially a spectral element method, using within each grid cell only the first 2 terms in a spectral expansion of the advected function in Legendre polynomials. Van Leer demonstrated that, for linear advection in 1-D, the resulting scheme was formally second-order accurate, but it caused error to accumulate only at the rate of a third-order accurate scheme.

A next member in this family of 1-D advection schemes was introduced by van Leer that I later generalized to 2-D advection working with Rick White in the early 1980s. We will call that scheme PPB, for Piecewise-Parabolic Boltzmann method, since the original use I

intended was for a numerical treatment of the Boltzmann equation in several (4 or 6) dimensions. In 1-D, this PPB scheme takes 3 terms in the Legendre expansion of the density distribution within each grid cell, and it conserves exactly the first 3 moments of that distribution, the mass, the center of mass, and the moment of inertia. Although it is formally third-order accurate, in fact error accumulates at the smaller rate of a fifth-order accurate scheme, as was shown originally by van Leer. We can think of this scheme as using the best fit parabola to describe the density distribution within each grid cell. The scheme is implemented rather easily in 1-D, and with some difficulty in 2-D. As a practical matter, its accuracy is, in comparison to more standard methods like PPM, simply phenomenal; however, a manageable, constrained (as in the sense of monotonicity or similar concepts) implementation of the scheme for use in multidimensional hydrodynamical schemes has been elusive. This paper presents such a formulation.

A key concept in making an implementation of the PPB advection scheme practical is the decomposition of the advection operation not only into a symmetrized series of 1-D passes but also into a sequence of similar operations within each 1-D pass. The 2-D implementation described in my 1982 and 1986 papers employs 1-D passes, but it is extremely complicated – so complicated that extensions to higher dimensions were never carried out. That 1-D operator included the detailed effects of the transverse shear of each grid cell, which makes each 1-D pass effectively multi-dimensional. In 2-D the work involved is just manageable, but in 3-D, let alone in a 6-D phase space, the work is prohibitive. About 10 years ago, in an attempt to simplify the 2-D PPB scheme enough so that it could be taught to undergraduate students, I eliminated the treatment of shear in each of its 1-D passes. There remains a 2-D aspect to each of these simplified 1-D passes, because of course the y-moments in each cell must be updated. I built each 1-D pass in 2-D out of 3 successive applications of the same 1-D PPB advection operator, first for the x-distribution of a, averaged over the y-coordinate (the zeroth y-moment), second for the x-distribution of the first y-moment, and finally for the x-distribution of the second y-moment. This simplified advection scheme, referred to in this study

simply as PPB, gives surprisingly good results everywhere but very near the centers of vortices, where the shear that is neglected within the cells is most important. Even with this flaw, results are extremely good, since the centers of vortices are locations where numerical diffusion is unavoidable, and this inevitable diffusion covers over the failings of the difference scheme. In the summer of 2002, I fixed this small failing of the simplified scheme by devising a technique of splitting each strip of grid cells into a top and bottom half-strip, advecting each of these independently with the PPB scheme, and then recombining these top and bottom halves to produce a result in which the noticeable effects of the earlier scheme's ignoring grid cell shear are almost entirely removed. This new scheme is here referred to as PPBshear. The cost of its additional computational labor is roughly a factor of 2.5, but the benefit is noticeably increased accuracy.

The original 2-D PPB scheme discussed in my 1982 and 1986 articles conserved 9 moments of the density distribution, a, within each cell to machine round-off accuracy. These moments can be used to uniquely determine the 9 interpolation polynomial coefficients in the following representation of a within the grid cell:

$$a(\tilde{x},\tilde{y}) = a_{00} + a_{10}\tilde{x} + a_{20}\tilde{x}^2 + a_{01}\tilde{y} + a_{11}\tilde{x}\tilde{y} + a_{21}\tilde{x}^2\tilde{y} + a_{02}\tilde{y}^2 + a_{12}\tilde{x}\tilde{y}^2 + a_{22}\tilde{x}^2\tilde{y}^2$$

Here we define cell-centric scaled coordinates via:

$$\tilde{x} = (x - x_M)/\Delta x$$

$$\tilde{y} = (y - y_M)/\Delta y$$

The 9 moments updated by the advection scheme are defined by the following equation:

$$\left\langle a\tilde{x}^k\tilde{y}^l \right\rangle = \int_{-1/2}^{1/2} d\tilde{x} \int_{-1/2}^{1/2} d\tilde{y}\; a(\tilde{x},\tilde{y})\,\tilde{x}^k\tilde{y}^l$$

where the integers k and l each range from 0 to 2. We can easily find all the moments in terms of the polynomial coefficients by using the relationships:

$$\int_{-1/2}^{1/2} \tilde{x}^2 d\tilde{x} = 1/12$$

$$\int_{-1/2}^{1/2} \tilde{x}^4 d\tilde{x} = 1/80$$

Thus we find:

$$\langle a\tilde{x} \rangle = a_{10}/12 + a_{12}/144$$

$$\langle a\tilde{y} \rangle = a_{01}/12 + a_{21}/144$$

$$\langle a\tilde{x}\tilde{y} \rangle = a_{11}/144$$

$$\langle a\tilde{x}^2 \rangle = \left(a_{00}/3 + a_{20}/20 + a_{02}/36 + a_{22}/240 \right) / 4$$

$$\langle a\tilde{y}^2 \rangle = \left(a_{00}/3 + a_{20}/36 + a_{02}/20 + a_{22}/240 \right) / 4$$

$$\langle a\tilde{x}^2\tilde{y} \rangle = \left(a_{01}/3 + a_{21}/20 \right) / 48$$

$$\langle a\tilde{x}\tilde{y}^2 \rangle = \left(a_{10}/3 + a_{12}/20 \right) / 48$$

$$\langle a\tilde{x}^2\tilde{y}^2 \rangle = \left(a_{00}/9 + (a_{20} + a_{02})/60 + a_{22}/400 \right) / 16$$

$$\langle a \rangle = a_{00} + \left((a_{20} + a_{02}) + a_{22}/12 \right) / 12$$

We can invert these relationships in order to obtain the polynomial coefficients from the moments:

$$a_{11} = 144 \langle a\tilde{x}\tilde{y} \rangle$$

$$a_{12} = 180 \left(12 \langle a\tilde{x}\tilde{y}^2 \rangle - \langle a\tilde{x} \rangle \right)$$

$$a_{10} = 9 \left(3 \langle a\tilde{x} \rangle - 20 \langle a\tilde{x}\tilde{y}^2 \rangle \right)$$

$$a_{01} = 9 \left(3 \langle a\tilde{y} \rangle - 20 \langle a\tilde{x}^2\tilde{y} \rangle \right)$$

$$a_{21} = 180 \left(12 \langle a\tilde{x}^2\tilde{y} \rangle - \langle a\tilde{y} \rangle \right)$$

$$a_{00} = 81/16 \langle a \rangle - 135/4 \left(\langle a\tilde{x}^2 \rangle + \langle a\tilde{y}^2 \rangle \right) + 225 \langle a\tilde{x}^2\tilde{y}^2 \rangle$$

$$a_{22} = 225 \langle a \rangle - 2700 \left(\langle a\tilde{x}^2 \rangle + \langle a\tilde{y}^2 \rangle \right) + 32400 \langle a\tilde{x}^2\tilde{y}^2 \rangle$$

$$a_{20} = 405 \langle a\tilde{x}^2 \rangle + 225 \langle a\tilde{y}^2 \rangle - 135/4 \langle a \rangle - 2700 \langle a\tilde{x}^2\tilde{y}^2 \rangle$$

$$a_{02} \;=\; 225\left\langle a\tilde{x}^2\right\rangle \;+\; 405\left\langle a\tilde{y}^2\right\rangle \;-\; {}^{135}\!/_{4}\left\langle a\right\rangle \;-\; 2700\left\langle a\tilde{x}^2\tilde{y}^2\right\rangle$$

All these formulae are derived and explained fully in another document. In order to constrain this interpolation polynomial in the spirit of the monotonicity concept implemented in van Leer's MUSCL scheme or in PPM, we need to relate the polynomial coefficients and/or the 9 moments to 9 other quantities, the overall cell average of a, the averages of a along the 4 cell edges, and the values of a at the 4 cell corners:

$$\left\langle a\right\rangle \;=\; a_{00}+\frac{1}{12}\left(a_{20}+a_{02}+\frac{1}{12}a_{22}\right)$$

$$\left\langle a_L\right\rangle \;=\; \left(a_{00}+\frac{1}{12}a_{02}\right)\;-\;\frac{1}{2}\left(a_{10}+\frac{1}{12}a_{12}\right)\;+\;\frac{1}{4}\left(a_{20}+\frac{1}{12}a_{22}\right)$$

$$\left\langle a_R\right\rangle \;=\; \left(a_{00}+\frac{1}{12}a_{02}\right)\;+\;\frac{1}{2}\left(a_{10}+\frac{1}{12}a_{12}\right)\;+\;\frac{1}{4}\left(a_{20}+\frac{1}{12}a_{22}\right)$$

$$\left\langle a_B\right\rangle \;=\; \left(a_{00}+\frac{1}{12}a_{20}\right)\;-\;\frac{1}{2}\left(a_{01}+\frac{1}{12}a_{21}\right)\;+\;\frac{1}{4}\left(a_{02}+\frac{1}{12}a_{22}\right)$$

$$\left\langle a_T\right\rangle \;=\; \left(a_{00}+\frac{1}{12}a_{20}\right)\;+\;\frac{1}{2}\left(a_{01}+\frac{1}{12}a_{21}\right)\;+\;\frac{1}{4}\left(a_{02}+\frac{1}{12}a_{22}\right)$$

$$a_{BL} \;=\; a_{00}\;-\;\frac{1}{2}(a_{10}+a_{01})\;+\;\frac{1}{4}(a_{20}+a_{11}+a_{02})\;-\;\frac{1}{8}(a_{21}+a_{12})\;+\;\frac{1}{16}a_{22}$$

$$a_{TL} \;=\; a_{00}\;-\;\frac{1}{2}(a_{10}-a_{01})\;+\;\frac{1}{4}(a_{20}-a_{11}+a_{02})\;+\;\frac{1}{8}(a_{21}-a_{12})\;+\;\frac{1}{16}a_{22}$$

$$a_{BR} \;=\; a_{00}\;+\;\frac{1}{2}(a_{10}-a_{01})\;+\;\frac{1}{4}(a_{20}-a_{11}+a_{02})\;-\;\frac{1}{8}(a_{21}-a_{12})\;+\;\frac{1}{16}a_{22}$$

$$a_{TR} \;=\; a_{00}\;+\;\frac{1}{2}(a_{10}+a_{01})\;+\;\frac{1}{4}(a_{20}+a_{11}+a_{02})\;+\;\frac{1}{8}(a_{21}+a_{12})\;+\;\frac{1}{16}a_{22}$$

Relating these values of a averaged over the cell, its 4 edges, and evaluated at its 4 corners to the 9 moments of a over the grid cell, we obtain the following relations:

$$\left\langle a_L\right\rangle \;=\; -\frac{3}{2}\left\langle a\right\rangle \;-\; 6\left\langle a\tilde{x}\right\rangle \;+\; 30\left\langle a\tilde{x}^2\right\rangle$$

$$\left\langle a_R\right\rangle \;=\; -\frac{3}{2}\left\langle a\right\rangle \;+\; 6\left\langle a\tilde{x}\right\rangle \;+\; 30\left\langle a\tilde{x}^2\right\rangle$$

$$\langle a_B \rangle = -\frac{3}{2}\langle a \rangle - 6\langle a\tilde{y} \rangle + 30\langle a\tilde{y}^2 \rangle$$

$$\langle a_T \rangle = -\frac{3}{2}\langle a \rangle + 6\langle a\tilde{y} \rangle + 30\langle a\tilde{y}^2 \rangle$$

$$a_{BL} = \frac{9}{4}\langle a \rangle + 9\left(\langle a\tilde{x} \rangle + \langle a\tilde{y} \rangle\right) - 180\left(\langle a\tilde{x}\tilde{y}^2 \rangle + \langle a\tilde{x}^2\tilde{y} \rangle\right) + 36\langle a\tilde{x}\tilde{y} \rangle - 45\left(\langle a\tilde{x}^2 \rangle + \langle a\tilde{y}^2 \rangle\right) + 900\langle a\tilde{x}^2\tilde{y}^2 \rangle$$

$$a_{BR} = \frac{9}{4}\langle a \rangle - 9\left(\langle a\tilde{x} \rangle - \langle a\tilde{y} \rangle\right) + 180\left(\langle a\tilde{x}\tilde{y}^2 \rangle - \langle a\tilde{x}^2\tilde{y} \rangle\right) - 36\langle a\tilde{x}\tilde{y} \rangle - 45\left(\langle a\tilde{x}^2 \rangle + \langle a\tilde{y}^2 \rangle\right) + 900\langle a\tilde{x}^2\tilde{y}^2 \rangle$$

$$a_{TL} = \frac{9}{4}\langle a \rangle + 9\left(\langle a\tilde{x} \rangle - \langle a\tilde{y} \rangle\right) - 180\left(\langle a\tilde{x}\tilde{y}^2 \rangle - \langle a\tilde{x}^2\tilde{y} \rangle\right) - 36\langle a\tilde{x}\tilde{y} \rangle - 45\left(\langle a\tilde{x}^2 \rangle + \langle a\tilde{y}^2 \rangle\right) + 900\langle a\tilde{x}^2\tilde{y}^2 \rangle$$

$$a_{TR} = \frac{9}{4}\langle a \rangle - 9\left(\langle a\tilde{x} \rangle + \langle a\tilde{y} \rangle\right) + 180\left(\langle a\tilde{x}\tilde{y}^2 \rangle + \langle a\tilde{x}^2\tilde{y} \rangle\right) + 36\langle a\tilde{x}\tilde{y} \rangle - 45\left(\langle a\tilde{x}^2 \rangle + \langle a\tilde{y}^2 \rangle\right) + 900\langle a\tilde{x}^2\tilde{y}^2 \rangle$$

It is possible to invert these relationships in order to obtain the moments if only the cell average, the edge values, and the corner values are known:

$$\langle a\tilde{x} \rangle = \frac{1}{12}\left(\langle a_R \rangle - \langle a_L \rangle\right)$$

$$\langle a\tilde{y} \rangle = \frac{1}{12}\left(\langle a_T \rangle - \langle a_B \rangle\right)$$

$$\langle a\tilde{x}^2 \rangle = \frac{1}{20}\left[\langle a \rangle + \frac{1}{3}\left(\langle a_R \rangle + \langle a_L \rangle\right)\right]$$

$$\langle a\tilde{y}^2 \rangle = \frac{1}{20}\left[\langle a \rangle + \frac{1}{3}\left(\langle a_T \rangle + \langle a_B \rangle\right)\right]$$

$$\langle a\tilde{x}\tilde{y} \rangle = \frac{1}{144}\left(a_{TR} - a_{BR} - a_{TL} + a_{BL}\right)$$

$$\langle a\tilde{x}\tilde{y}^2 \rangle = \frac{1}{240}\left[\left(\langle a_R \rangle - \langle a_L \rangle\right) - \frac{1}{3}\left(a_{TR} + a_{BR} - a_{TL} - a_{BL}\right)\right]$$

$$\left\langle a\tilde{x}^2\tilde{y}\right\rangle \;=\; \frac{1}{240}\left[\left(\langle a_T\rangle-\langle a_B\rangle\right)-\frac{1}{3}\left(a_{TL}+a_{TR}-a_{BL}-a_{BR}\right)\right]$$

$$\left\langle a\tilde{x}^2\tilde{y}^2\right\rangle \;=\; \frac{1}{400}\left[\langle a\rangle+\frac{1}{3}\left(\langle a_R\rangle+\langle a_L\rangle+\langle a_T\rangle+\langle a_B\rangle\right)+\frac{1}{9}\left(a_{TR}+a_{BR}+a_{TL}+a_{BL}\right)\right]$$

We can also determine the interpolation polynomial coefficients from the cell average, the edge averages, and the corner values from the following formulae:

$$a_{00} \;=\; \frac{9}{4}\langle a\rangle+\frac{1}{16}\left(a_{TR}+a_{BR}+a_{TL}+a_{BL}\right)-\frac{3}{8}\left(\langle a_R\rangle+\langle a_L\rangle+\langle a_T\rangle+\langle a_B\rangle\right)$$

$$a_{10} \;=\; \frac{3}{2}\left(\langle a_R\rangle-\langle a_L\rangle\right)-\frac{1}{4}\left(a_{TR}+a_{BR}-a_{TL}-a_{BL}\right)$$

$$a_{01} \;=\; \frac{3}{2}\left(\langle a_T\rangle-\langle a_B\rangle\right)-\frac{1}{4}\left(a_{TR}-a_{BR}+a_{TL}-a_{BL}\right)$$

$$a_{20} \;=\; \frac{9}{2}\left(\langle a_R\rangle+\langle a_L\rangle\right)+\frac{3}{2}\left(\langle a_T\rangle+\langle a_B\rangle\right)-\frac{3}{4}\left(a_{TR}+a_{BR}+a_{TL}+a_{BL}\right)-9\langle a\rangle$$

$$a_{02} \;=\; \frac{3}{2}\left(\langle a_R\rangle+\langle a_L\rangle\right)+\frac{9}{2}\left(\langle a_T\rangle+\langle a_B\rangle\right)-\frac{3}{4}\left(a_{TR}+a_{BR}+a_{TL}+a_{BL}\right)-9\langle a\rangle$$

$$a_{11} \;=\; a_{TR}-a_{BR}-a_{TL}+a_{BL}$$

$$a_{21} \;=\; 3\left(a_{TR}-a_{BR}+a_{TL}-a_{BL}\right)-6\left(\langle a_T\rangle-\langle a_B\rangle\right)$$

$$a_{12} \;=\; 3\left(a_{TR}+a_{BR}-a_{TL}-a_{BL}\right)-6\left(\langle a_R\rangle-\langle a_L\rangle\right)$$

$$a_{22} \;=\; 36\langle a\rangle+9\left(a_{TR}+a_{BR}+a_{TL}+a_{BL}\right)-18\left(\langle a_R\rangle+\langle a_L\rangle+\langle a_T\rangle+\langle a_B\rangle\right)$$

The relationships given above that allow us to go back and forth between different and equivalent sets of 9 independent values for each grid cell are extremely useful. The 9 moments are considered by the PPB advection scheme to be the independent quantities that it advances in time. This is because it is the evaluation of these 9 moments after advection has taken place during a time step (or, more properly, during a single 1-D pass of the algorithm) that allows us to construct a new interpolation polynomial, continuous everywhere within the grid cell. The PPB advection scheme therefore performs the advection exactly, except for shear within a grid cell, and only

introduces error by conserving just the first 9 moments of the advected distribution rather than the complete infinite set of moments. However, it is difficult to apply monotonicity or other appropriate constraints to the moments directly, or even to the polynomial coefficients. Instead, the moment information is converted to cell average, edge average, and corner values that are easily constrained. The constrained values are then converted back to new values of the 9 moments. Also, in order to evaluate the moments of the advected distribution, which is discontinuous inside the grid cells, we find it most useful to convert the original moment information into interpolation polynomial coefficients, so that the appropriate moment integrals can be evaluated over the appropriate domains. This is all straightforward, but tedious and complex. We will see below that the process is rendered far simpler by our decision to develop the PPB scheme as a series of 1-D operators applied to various distributions derived from the full distribution of a within the grid cell.

Before describing the PPB scheme, we note that the 9 moments can be used to generate average values of the distribution a in subsections of a grid cell. This can be extremely useful for plotting purposes, since the moments contain considerable information that is worth displaying. It is most natural to convert the 9 moments into average values in 9 equal rectangular subregions of the grid cell. We can divide the cell into ninths using the following relations:

$$3\int_{1/6}^{1/2}\tilde{x}\,d\tilde{x} \;=\; 1/3 \;=\; 3/8\,(1-1/9)$$

$$3\int_{-1/6}^{1/6}\tilde{x}^2\,d\tilde{x} \;=\; 1/108$$

$$3\int_{1/6}^{1/2}\tilde{x}^2\,d\tilde{x} \;=\; 13/108 \;=\; 1/8\,(1-1/27)$$

We therefore obtain:

$$\langle a\rangle_{BL} \;=\; a_{00} \;-\; 1/3\,(a_{10}+a_{01}) \;+\; 1/9\,a_{11} \;+\; 13/108\,(a_{20}+a_{02}) \;-\; 13/324\,(a_{21}+a_{12}) \;+\; 169/11664\,a_{22}$$

$$\langle a\rangle_{BM} \;=\; a_{00} \;-\; 1/3\,a_{01} \;+\; 1/108\,(a_{20}+13\,a_{02}) \;-\; 1/324\,a_{21} \;+\; 13/11664\,a_{22}$$

$$\langle a\rangle_{BR} = a_{00} + \tfrac{1}{3}(a_{10}-a_{01}) - \tfrac{1}{9}a_{11} + \tfrac{13}{108}(a_{20}+a_{02}) - \tfrac{13}{324}(a_{21}-a_{12}) + \tfrac{169}{11664}a_{22}$$

$$\langle a\rangle_{ML} = a_{00} - \tfrac{1}{3}a_{10} + \tfrac{1}{108}(13a_{20}+a_{02}) - \tfrac{1}{324}a_{12} + \tfrac{13}{11664}a_{22}$$

$$\langle a\rangle_{MM} = a_{00} + \tfrac{1}{108}(a_{20}+a_{02}) + \tfrac{1}{11664}a_{22}$$

$$\langle a\rangle_{MR} = a_{00} + \tfrac{1}{3}a_{10} + \tfrac{1}{108}(13a_{20}+a_{02}) + \tfrac{1}{324}a_{12} + \tfrac{13}{11664}a_{22}$$

$$\langle a\rangle_{TL} = a_{00} - \tfrac{1}{3}(a_{10}-a_{01}) - \tfrac{1}{9}a_{11} + \tfrac{13}{108}(a_{20}+a_{02}) + \tfrac{13}{324}(a_{21}-a_{12}) + \tfrac{169}{11664}a_{22}$$

$$\langle a\rangle_{TM} = a_{00} + \tfrac{1}{3}a_{01} + \tfrac{1}{108}(a_{20}+13a_{02}) + \tfrac{1}{324}a_{21} + \tfrac{13}{11664}a_{22}$$

$$\langle a\rangle_{TR} = a_{00} + \tfrac{1}{3}(a_{10}+a_{01}) + \tfrac{1}{9}a_{11} + \tfrac{13}{108}(a_{20}+a_{02}) + \tfrac{13}{324}(a_{21}+a_{12}) + \tfrac{169}{11664}a_{22}$$

We can invert these relationships to obtain:

$$a_{00} = \tfrac{169}{144}\langle a\rangle_{MM} + \tfrac{1}{576}\left(\langle a\rangle_{BL}+\langle a\rangle_{BR}+\langle a\rangle_{TL}+\langle a\rangle_{TR}\right) - \tfrac{13}{288}\left(\langle a\rangle_{ML}+\langle a\rangle_{MR}+\langle a\rangle_{BM}+\langle a\rangle_{TM}\right)$$

$$16\,a_{10} = 26\left(\langle a\rangle_{MR}-\langle a\rangle_{ML}\right) - \langle a\rangle_{TR} + \langle a\rangle_{TL} - \langle a\rangle_{BR} + \langle a\rangle_{BL}$$

$$16\,a_{01} = 26\left(\langle a\rangle_{TM}-\langle a\rangle_{BM}\right) - \langle a\rangle_{TR} + \langle a\rangle_{BR} - \langle a\rangle_{TL} + \langle a\rangle_{BL}$$

$$\tfrac{4}{9}a_{11} = \langle a\rangle_{TR} - \langle a\rangle_{TL} - \langle a\rangle_{BR} + \langle a\rangle_{BL}$$

$$a_{20} = \tfrac{9}{2}\left(\langle a\rangle_{ML}+\langle a\rangle_{MR}\right) - \tfrac{39}{4}\langle a\rangle_{MM} - \tfrac{3}{16}\left(\langle a\rangle_{BL}+\langle a\rangle_{BR}+\langle a\rangle_{TL}+\langle a\rangle_{TR}\right) + \tfrac{3}{8}\left(\langle a\rangle_{ML}+\langle a\rangle_{MR}+\langle a\rangle_{BM}+\langle a\rangle_{TM}\right)$$

$$a_{02} = -\tfrac{9}{2}\left(\langle a\rangle_{ML}+\langle a\rangle_{MR}\right) - \tfrac{39}{4}\langle a\rangle_{MM} - \tfrac{3}{16}\left(\langle a\rangle_{BL}+\langle a\rangle_{BR}+\langle a\rangle_{TL}+\langle a\rangle_{TR}\right) + \tfrac{39}{8}\left(\langle a\rangle_{ML}+\langle a\rangle_{MR}+\langle a\rangle_{BM}+\langle a\rangle_{TM}\right)$$

$$\tfrac{4}{27}a_{12} = \langle a\rangle_{TR} - \langle a\rangle_{TL} + \langle a\rangle_{BR} - \langle a\rangle_{BL} - 2\left(\langle a\rangle_{MR}-\langle a\rangle_{ML}\right)$$

$$\tfrac{4}{27}a_{21} = \langle a\rangle_{TR} - \langle a\rangle_{BR} + \langle a\rangle_{TL} - \langle a\rangle_{BL} - 2\left(\langle a\rangle_{TM}-\langle a\rangle_{BM}\right)$$

$$a_{22} = 81\langle a\rangle_{MM} + \tfrac{81}{4}\left(\langle a\rangle_{BL}+\langle a\rangle_{BR}+\langle a\rangle_{TL}+\langle a\rangle_{TR}\right) - \tfrac{81}{2}\left(\langle a\rangle_{ML}+\langle a\rangle_{MR}+\langle a\rangle_{BM}+\langle a\rangle_{TM}\right)$$

For plotting we generally will wish to decompose each cell only into 4 equal subcells. To evaluate these subcell averages, we will use the following relationships:

$$2\int_0^{1/2}\tilde{x}\,d\tilde{x} \;=\; 1/4$$

$$2\int_0^{1/2}\tilde{x}^2\,d\tilde{x} \;=\; 1/12$$

Together with the odd or even nature of these integrals when $\tilde{x}$ *is replaced by* $-\tilde{x}$ *, these imply:*

$$\langle a\rangle_{BL} \;=\; a_{00} \;-\; \tfrac{1}{4}(a_{10}+a_{01}) \;+\; \tfrac{1}{16}a_{11} \;+\; \tfrac{1}{12}(a_{20}+a_{02}) \;-\; \tfrac{1}{48}(a_{21}+a_{12}) \;+\; \tfrac{1}{144}a_{22}$$

$$\langle a\rangle_{BR} \;=\; a_{00} \;+\; \tfrac{1}{4}(a_{10}-a_{01}) \;-\; \tfrac{1}{16}a_{11} \;+\; \tfrac{1}{12}(a_{20}+a_{02}) \;-\; \tfrac{1}{48}(a_{21}-a_{12}) \;+\; \tfrac{1}{144}a_{22}$$

$$\langle a\rangle_{TL} \;=\; a_{00} \;-\; \tfrac{1}{4}(a_{10}-a_{01}) \;-\; \tfrac{1}{16}a_{11} \;+\; \tfrac{1}{12}(a_{20}+a_{02}) \;+\; \tfrac{1}{48}(a_{21}-a_{12}) \;+\; \tfrac{1}{144}a_{22}$$

$$\langle a\rangle_{TR} \;=\; a_{00} \;+\; \tfrac{1}{4}(a_{10}+a_{01}) \;+\; \tfrac{1}{16}a_{11} \;+\; \tfrac{1}{12}(a_{20}+a_{02}) \;+\; \tfrac{1}{48}(a_{21}+a_{12}) \;+\; \tfrac{1}{144}a_{22}$$

While we are discussing subdividing grid cells, we will include the formulae we use to divide a grid cell into a top and bottom cell, each with its own set of 9 derived moments referred to rescaled coordinates in the y-direction that range from -½ to +½ across the height of each subscell. If we define:

$$\tilde{y}_{\pm} = 2\left(\tilde{y} \pm \tfrac{1}{4}\right)$$

then the subscript + corresponds to the bottom subcell, while the subscript - corresponds to the top subcell (sorry for the bad choice of notation here). We may invert the above relationship, and then substitute the resulting equation, $\tilde{y} = \frac{1}{2}\tilde{y}_{\pm} \mp \frac{1}{4}$ *, into the general form for our interpolation polynomial in order to determine the equivalent coefficients of this polynomial, with respect to the rescaled y-coordinates, in each subcell. The most useful relationships turn out to be those which give us the 9 moments with respect to the rescaled coordinates within each of the top and bottom subcells:*

$$\langle a \rangle_{\pm} = \langle a \rangle \mp 3\langle a\,\tilde{y} \rangle$$

$$\langle a\,\tilde{x} \rangle_{\pm} = \langle a\,\tilde{x} \rangle \mp 3\langle a\,\tilde{x}\,\tilde{y} \rangle$$

$$\langle a\,\tilde{x}^2 \rangle_{\pm} = \langle a\,\tilde{x}^2 \rangle \mp 3\langle a\,\tilde{x}^2\tilde{y} \rangle$$

$$\langle a\,\tilde{y} \rangle_{\pm} = \frac{1}{2}\langle a\,\tilde{y} \rangle \mp \left(\frac{15}{4}\langle a\,\tilde{y}^2 \rangle - \frac{5}{16}\langle a \rangle \right)$$

$$\langle a\,\tilde{x}\,\tilde{y} \rangle_{\pm} = \frac{1}{2}\langle a\,\tilde{x}\,\tilde{y} \rangle \mp \left(\frac{15}{4}\langle a\,\tilde{x}\,\tilde{y}^2 \rangle - \frac{5}{16}\langle a\,\tilde{x} \rangle \right)$$

$$\langle a\,\tilde{x}^2\tilde{y} \rangle_{\pm} = \frac{1}{2}\langle a\,\tilde{x}^2\,\tilde{y} \rangle \mp \left(\frac{15}{4}\langle a\,\tilde{x}^2\tilde{y}^2 \rangle - \frac{5}{16}\langle a\,\tilde{x}^2 \rangle \right)$$

$$\langle a\,\tilde{y}^2 \rangle_{\pm} = \frac{1}{16}\langle a \rangle + \frac{1}{4}\langle a\,\tilde{y}^2 \rangle \mp \frac{1}{4}\langle a\,\tilde{y} \rangle$$

$$\langle a\,\tilde{x}\,\tilde{y}^2 \rangle_{\pm} = \frac{1}{16}\langle a\,\tilde{x} \rangle + \frac{1}{4}\langle a\,\tilde{x}\,\tilde{y}^2 \rangle \mp \frac{1}{4}\langle a\,\tilde{x}\,\tilde{y} \rangle$$

$$\langle a\,\tilde{x}^2\tilde{y}^2 \rangle_{\pm} = \frac{1}{16}\langle a\,\tilde{x}^2 \rangle + \frac{1}{4}\langle a\,\tilde{x}^2\tilde{y}^2 \rangle \mp \frac{1}{4}\langle a\,\tilde{x}^2\,\tilde{y} \rangle$$

Using these formulae, we can split up a grid cell into top and bottom sections. After performing the PPB advection step on each slice independently, we need to recombine the resulting subcells. We can do this operation using the following formulae:

$$\langle a \rangle = \frac{1}{2}\left(\langle a \rangle_{+} + \langle a \rangle_{-} \right)$$

$$\langle a\,\tilde{x} \rangle = \frac{1}{2}\left(\langle a\,\tilde{x} \rangle_{+} + \langle a\,\tilde{x} \rangle_{-} \right)$$

$$\langle a\,\tilde{x}^2 \rangle = \frac{1}{2}\left(\langle a\,\tilde{x}^2 \rangle_{+} + \langle a\,\tilde{x}^2 \rangle_{-} \right)$$

$$\langle a\,\tilde{y} \rangle = \frac{1}{4}\left(\langle a\,\tilde{y} \rangle_{+} + \langle a\,\tilde{y} \rangle_{-} \right) + \frac{1}{8}\left(\langle a \rangle_{-} - \langle a \rangle_{+} \right)$$

$$\langle a\,\tilde{x}\,\tilde{y} \rangle = \frac{1}{4}\left(\langle a\,\tilde{x}\,\tilde{y} \rangle_{+} + \langle a\,\tilde{x}\,\tilde{y} \rangle \right) + \frac{1}{8}\left(\langle a\,\tilde{x} \rangle_{-} - \langle a\,\tilde{x} \rangle_{+} \right)$$

$$\langle a\,\tilde{x}^2\,\tilde{y} \rangle = \frac{1}{4}\left(\langle a\,\tilde{x}^2\tilde{y} \rangle_{+} + \langle a\,\tilde{x}^2\tilde{y} \rangle_{-} \right) + \frac{1}{8}\left(\langle a\,\tilde{x}^2 \rangle_{-} - \langle a\,\tilde{x}^2 \rangle_{+} \right)$$

$$\left\langle a\,\tilde{y}^2\right\rangle \;=\; \frac{1}{8}\left(\left\langle a\,\tilde{y}^2\right\rangle_+ + \left\langle a\,\tilde{y}^2\right\rangle_-\right) \;+\; \frac{1}{32}\left(\langle a\rangle_+ + \langle a\rangle_-\right) \;+\; \frac{1}{8}\left(\langle a\,\tilde{y}\rangle_- - \langle a\,\tilde{y}\rangle_+\right)$$

$$\left\langle a\,\tilde{x}\,\tilde{y}^2\right\rangle \;=\; \frac{1}{8}\left(\left\langle a\,\tilde{x}\,\tilde{y}^2\right\rangle_+ + \left\langle a\,\tilde{x}\,\tilde{y}^2\right\rangle_-\right) \;+\; \frac{1}{32}\left(\langle a\,\tilde{x}\rangle_+ + \langle a\,\tilde{x}\rangle_-\right) \;+\; \frac{1}{8}\left(\langle a\,\tilde{x}\,\tilde{y}\rangle_- - \langle a\,\tilde{x}\,\tilde{y}\rangle_+\right)$$

$$\left\langle a\,\tilde{x}^2\tilde{y}^2\right\rangle \;=\; \frac{1}{8}\left(\left\langle a\,\tilde{x}^2\tilde{y}^2\right\rangle_+ + \left\langle a\,\tilde{x}^2\tilde{y}^2\right\rangle_-\right) \;+\; \frac{1}{32}\left(\left\langle a\,\tilde{x}^2\right\rangle_+ + \left\langle a\,\tilde{x}^2\right\rangle_-\right) \;+\; \frac{1}{8}\left(\left\langle a\,\tilde{x}^2\tilde{y}\right\rangle_- - \left\langle a\,\tilde{x}^2\tilde{y}\right\rangle_+\right)$$

And we must be sure to remember that

$$\tilde{y}_\pm = 2\left(\tilde{y} \pm \tfrac{1}{4}\right)$$

so that the upper sign goes with the lower half of the grid cell.

The above formulae for combining the upper and lower halves of the grid cell may seem at first glance to be overly complex. This is not the case. The 18 subgrid cell moments can be used to obtain 9 merged cell moments in many ways. The formulae above come from evaluating the 9 moments of the combined cell directly from the implied interpolation polynomials in each of the subcell sections. This method is essential in maintaining the property of the PPB scheme that, if we are able to advect each cell exactly before reconstructing a new, continuous interpolation polynomial, the 9 moments of the overall distribution are exactly conserved.

In this paper, I will discuss PPB advection in 2-D. There are no new issues that enter for PPB advection in additional numbers of dimensions. The reason for this is our decomposition of the advection process into 1-D passes, and our decomposition of each 1-D pass into a series of simple 1-D advection operations. Even monotonicity or other constraints can be applied incrementally, through a series of 1-D operations. However, as the number of dimensions increases, the number of moments that we must update in a PPB advection scheme mounts dramatically. In 1-D, we have only 3 moments to update. In 2-D, we have 9, which is still a manageable number. However, in 3-D we have 27 moments to update. We can perform the update in a series of 1-D operations, but even if the machine time is not a concern (CPU computation comes nearly free on most modern machines, due to their fundamental lack of computational design balance), the memory required for all these moments at each grid point can

become excessive. For a Boltzmann equation simulation, for example to describe the dynamics of a self-gravitating disk of stars, we would need to update $3^6 = 729$ moments for each grid cell! Can this really be necessary? Can it possibly be worthwhile? In the scheme comparisons presented below, the necessary data for you to make this decision for yourself will be provided.

Our interpolation polynomial is "bi-quadratic." That is, it includes all the terms we get when we construct such a polynomial through the product of a parabola in x and a parabola in y. Of course, such a product polynomial, although it has 9 coefficients, contains only 6 independent combinations of its coefficients. Since we have not restricted our polynomial to be such a product, our interpolation polynomial is more general, and it has 9 independent coefficients. But do we really need all 9 of these coefficients? Although our rescaled x- and y-coordinates each range from $-½$ to $+½$, our cells are of course small, so that these values are ultimately multiplied by the small parameters Δx and Δy. Therefore our interpolation polynomial contains 3 terms which are all higher than second-order small. In fact, along the cell diagonals, our interpolation function takes the form of a quartic polynomial, which is surely overkill. If we are satisfied with a parabola for interpolation in both the x- and y-directions, why not settle for parabolae along the cell diagonals as well? If we were to do so, we could set the interpolation polynomial coefficients a_{12}, a_{21}, and a_{22} to zero. This would leave us with only 6 coefficients, which we could determine from only 6 moments rather than 9. In 2-D, this would be considerably less work, about half as much work in fact. However, in 3-D we would have only 10 moments to contend with instead of 27, and our savings would be proportionally greater. In 6-D, there simply would be no worthwhile comparison, it would seem, between the two approaches.

The issue of 6 moments or 9, or of 10 moments or 27, is not so clear as it might seem. The advection experiments discussed below will establish that the 9-moment method, which we here call simply PPB, is considerably more accurate than the 6-moment method, here called PPB6. Apparently, formal order of accuracy is not everything. The key difference between these two schemes seems to be that the PPB scheme has greater flexibility to fit the behavior of the

interpolated function near the cell corners. In fact, we have seen above that the 9-moment interpolation function can attain any values at the cell corners and along the cell edges that we might care to prescribe. The 6-moment interpolation function can fit any prescribed average values along the cell edges, but it can only represent the difference of the sums of the two corner values along the two cell diagonals through its coefficient a_{11} *or its moment* $\langle a\tilde{x}\tilde{y}\rangle$. *We should not be overly surprised that the flexibility to provide a quality fit along the cell diagonals is important, since, after all, the distance along the diagonals is 41% greater than along the principal grid directions. The question, which surely is application dependent, of whether the extra moments, the extra complexity, the extra labor, and the extra computer memory is worthwhile is here left to the reader to decide. Evidence of the relative merits of the two approaches, at least in 2-D, is provided in order to assist in making this decision.*

Before launching into the description of the advection algorithm for updating the moments, we note that in the 6-moment approach, or PPB6, the 3 interpolation polynomial coefficients a_{12}, a_{21}, *and* a_{22} *are all assumed to vanish. Through the formulae given earlier for the moments in terms of the polynomial coefficients, this constraint implies that the 3 highest-order moments can be derived from the lower-order moments as follows:*

$$\langle a\tilde{x}\tilde{y}^2\rangle = \langle a\tilde{x}\rangle/12$$

$$\langle a\tilde{x}^2\tilde{y}\rangle = \langle a\tilde{y}\rangle/12$$

$$\langle a\tilde{x}^2\tilde{y}^2\rangle = \tfrac{1}{12}\left(\langle a\tilde{x}^2\rangle + \langle a\tilde{y}^2\rangle\right) - \tfrac{1}{144}\langle a\rangle$$

We have the simpler set of relationships between the 6 moments and the 6 interpolation polynomial coefficients:

$$\langle a\rangle = a_{00} + (a_{20} + a_{02})\ /\ 12$$

$$\langle a\tilde{x}\rangle = a_{10}\ /12$$

$$\langle a\tilde{y}\rangle = a_{01}\ /12$$

$$\langle a\tilde{x}\tilde{y}\rangle = a_{11}/144$$

$$\langle a\tilde{x}^2\rangle = (a_{00}/3 + a_{20}/20 + a_{02}/36)/4$$

$$\langle a\tilde{y}^2\rangle = (a_{00}/3 + a_{20}/36 + a_{02}/20)/4$$

We can invert these relationships in order to obtain the polynomial coefficients from the moments:

$$a_{00} = \tfrac{7}{2}\langle a\rangle - 15\left(\langle a\tilde{x}^2\rangle + \langle a\tilde{y}^2\rangle\right)$$

$$a_{10} = 12\langle a\tilde{x}\rangle$$

$$a_{01} = 12\langle a\tilde{y}\rangle$$

$$a_{11} = 144\langle a\tilde{x}\tilde{y}\rangle$$

$$a_{20} = 180\langle a\tilde{x}^2\rangle - 15\langle a\rangle$$

$$a_{02} = 180\langle a\tilde{y}^2\rangle - 15\langle a\rangle$$

We can also relate the cell edge and corner values to the 6 moments via:

$$\langle a\rangle_L = 30\langle a\tilde{x}^2\rangle - \frac{3}{2}\langle a\rangle - 6\langle a\tilde{x}\rangle$$

$$\langle a\rangle_R = 30\langle a\tilde{x}^2\rangle - \frac{3}{2}\langle a\rangle + 6\langle a\tilde{x}\rangle$$

$$\langle a\rangle_B = 30\langle a\tilde{y}^2\rangle - \frac{3}{2}\langle a\rangle - 6\langle a\tilde{y}\rangle$$

$$\langle a\rangle_T = 30\langle a\tilde{y}^2\rangle - \frac{3}{2}\langle a\rangle + 6\langle a\tilde{y}\rangle$$

$$a_{BL} = 30\left(\langle a\tilde{x}^2\rangle + \langle a\tilde{y}^2\rangle\right) - 4\langle a\rangle - 6\left(\langle a\tilde{x}\rangle + \langle a\tilde{y}\rangle\right) + 36\langle a\tilde{x}\tilde{y}\rangle$$

$$a_{TR} = 30\left(\langle a\tilde{x}^2\rangle + \langle a\tilde{y}^2\rangle\right) - 4\langle a\rangle + 6\left(\langle a\tilde{x}\rangle + \langle a\tilde{y}\rangle\right) + 36\langle a\tilde{x}\tilde{y}\rangle$$

$$a_{TL} = 30\left(\langle a\tilde{x}^2\rangle + \langle a\tilde{y}^2\rangle\right) - 4\langle a\rangle - 6\left(\langle a\tilde{x}\rangle - \langle a\tilde{y}\rangle\right) - 36\langle a\tilde{x}\tilde{y}\rangle$$

$$a_{BR} = 30\left(\langle a\tilde{x}^2\rangle + \langle a\tilde{y}^2\rangle\right) - 4\langle a\rangle + 6\left(\langle a\tilde{x}\rangle - \langle a\tilde{y}\rangle\right) - 36\langle a\tilde{x}\tilde{y}\rangle$$

and we note that:

$$a_{11} \;=\; a_{BL} + a_{TR} - a_{TL} - a_{BR}$$

If in this formula we approximate each corner value by a quarter of the sum of the average values in the 4 cells adjacent to that corner, then in the resulting expression for a_{11} *the contribution from the central cell will cancel (2 plus and 2 minus) and for each nearest-neighbor cell we will have one positive contribution from one corner and a negative one for the other. It is therefore clear that the formula for this interpolation polynomial coefficient in terms of neighbor cell averages must be:*

$$a_{11} \;=\; \tfrac{1}{4}\left(\langle a\rangle_{ZBL} + \langle a\rangle_{ZTR} - \langle a\rangle_{ZTL} - \langle a\rangle_{ZBR}\right)$$

We can obtain this same formula by extrapolating our assumed interpolation polynomial into the 8 surrounding cells, integrating it over the 4 corner neighboring cells, and summing those results as in the formula above. All contributions of all other coefficients in our assumed functional form must then vanish by symmetry, and we are left with a contribution from the coefficient a_{11} *alone.*

We may also determine the average values in the cell quadrants from the 6 moments:

$$\langle a\rangle_{BL} \;=\; \langle a\rangle \;-\; 3\left(\langle a\tilde{x}\rangle + \langle a\tilde{y}\rangle\right) \;+\; 9\,\langle a\tilde{x}\,\tilde{y}\rangle$$

$$\langle a\rangle_{TR} \;=\; \langle a\rangle \;+\; 3\left(\langle a\tilde{x}\rangle + \langle a\tilde{y}\rangle\right) \;+\; 9\,\langle a\tilde{x}\,\tilde{y}\rangle$$

$$\langle a\rangle_{TL} \;=\; \langle a\rangle \;-\; 3\left(\langle a\tilde{x}\rangle - \langle a\tilde{y}\rangle\right) \;-\; 9\,\langle a\tilde{x}\,\tilde{y}\rangle$$

$$\langle a\rangle_{BR} \;=\; \langle a\rangle \;+\; 3\left(\langle a\tilde{x}\rangle - \langle a\tilde{y}\rangle\right) \;-\; 9\,\langle a\tilde{x}\,\tilde{y}\rangle$$

Constraining the Moments in a 1-D Pass of the PPB Advection Scheme

In the previous section, we set out the basis for the PPB moment-conserving advection scheme. It remains to lay out the details of how the moments conserved by the scheme are updated in a single 1-D pass of the algorithm.

The first operation we perform in each 1-D pass is to apply constraints to the interpolation polynomial implied by the values of the 9 or 6 moments prescribed for each grid cell at the beginning of the 1-D pass. Because the advection in the x-direction deals only with functions that are averaged over the y-coordinate (even if such functions may turn out to be the first or second y-moments of the distribution in the cell), it is appropriate for the x-pass only to constrain the x-behavior of the average of the function over y. The behavior of the distribution within the cell in the y-direction will be constrained in the y-pass. Nevertheless, we will need to apply any desired constraints on the behavior of the function in the directions of the cell diagonals in both the x- and the y-passes. Any such constraints will affect the "corner transport" implied by the scheme.

The two-stream instability test problem that is described below involves only linear advection. That is, all grid cells in each grid strip travel to the left or right at precisely the same velocity. However, we will also describe the application of PPB advection to more complex problems in nonlinear fluid dynamics, where this simple behavior does not generally occur. One of the most potentially useful applications of PPB advection in fluid dynamics simulations performed with difference schemes such as PPM is the advection of variables that give the fraction of the fluid in each cell that is of a particular type. Such fluid "types" could be different materials, such as sulfur hexaflouride and air, concentrations of constituents such as water vapor or sulfur dioxide, or simply concentrations of passively advected tracers, such as smoke or dye, introduced into the fluid to trace its motion in a subsequent or even real-time visualization of the flow. For such advected quantities, it is most appropriate to apply the obvious constraint that the values of these variables must everywhere lie within the range from 0 to 1. One could consider applying monotonicity

constraints, as is standard practice for difference schemes such as PPM. However, the PPB schemes have such great resolving power that they can easily be catastrophically degraded by the application of monotonicity constraints that introduce little additional error into far less accurate schemes such as PPM. It is best to demand that these PPB schemes produce no obviously wrong values but otherwise to let them do their magic unimpeded. The monotonicity constraint that the interpolated polynomial must attain its extreme values for the grid cell only at the edges of the cell is inappropriate for a PPB scheme that can accurately advect sine wave disturbances with wavelengths of only 4 or 5 grid cells. Already the modern versions of PPM in use at the LCSE perform complicated inspections of the local behavior of any function to be interpolated in order to determine whether or not the function is smooth, so that monotonicity constraints that clip extrema can be dispensed with. For the PPB schemes, we would have to redesign such tests for function smoothness in order to let functions pass as "smooth" which schemes like PPM would regard as close to discontinuous. Hence, we describe below only the much weaker constraint that the interpolation function should assume values only in the range from 0 to 1.

For the specific case of advection of the fraction of a given fluid in a multifluid hydrodynamics problem, especially in the common case that this fluid fraction should be strictly conserved along streamlines, we note that there are few situations in which we need be concerned about the generation of inappropriate oscillations in the advected distribution. The features to be advected consist most importantly of edges that separate regions which are entirely composed of one type of fluid from regions composed entirely of another. These edges will tend to be steepened by the action of shear, but chances are that the edges were already as steep as they can be at the outset of the problem, so that this steepening process will be balanced immediately by numerical diffusion. Our constraint that all values of the function lie between 0 and 1 will apply just the monotonicity constraint to these edge structures that we wish.

In multifluid problems of the type where at the outset of the problem the fluids are completely unmixed and where physical diffusion of the fluids is negligible, we do not expect to find

edges developing between regions of constant fluid fraction unless the constant values on either side of the edges are the values 0 and 1. In these problems, the fluids can mix as the result of a variety of instabilities, such as the Kelvin-Helmholtz, Rayleigh-Taylor, and Richtmyer-Meshkov instabilities. All these instabilities involve deposition and/or amplification (especially in 3-D) of vorticity along the region of the multifluid interface. In the nonlinear regime, this results in entrainment of one fluid into the region of another in vortex structures (the typical Rayleigh-Taylor or Richtmyer-Meshkov spikes or plumes rapidly develop ring vortices at their tips). This process does not bring about induction of broad regions of pure fluid A into surrounding regions of pure fluid B. Instead, it results in thin tongues of each fluid being drawn into the regions of the other. These thin tongues are continually stretched, so that they tend to become thinner without limit. Thus, aside from the simple edge, in which the fluid fraction jumps from 0 to 1, the typical unresolvable structures in such problems are relatively long, thin strips (or possibly sheets) where the fluid fraction is either 0 or 1. These thin strips or sheets are drawn into regions where the fluid fraction is, respectively, either 1 or 0. For a single such strand where the value should be 1, for example, surrounded by a broad field at the value 0, our constraint that all values lie between 0 and 1 will guarantee that no oscillations will occur along the edges of this strand. Where the strand of fluid becomes unresolved on the mesh, our constraints will allow the PPB method to describe it with parabolae inside the grid cells that can assume maximum values in those cells. We will not clip those maxima, and the result will be a much better description of the fluid mixing on the subgrid scale, with no unrealistic oscillations generated in the process. The test problems discussed below illustrate this point quite well. To clip the maxima along these strands of fluid, as would occur if we were to use the PPM advection scheme, would be to destroy the resolving power of the PPB approach. Thus, our decision to enforce only the constraint to values between 0 and 1 is application dependent, but it is nevertheless strongly motivated. We should remember that the standard monotonicity constraint used in shock problems is not a law of nature but instead an application dependent design choice.

We will apply our monotonicity constraints in 1-D passes, and therefore it is appropriate to examine the dependence on x of the distribution averaged over y. For this distribution, our interpolation polynomial takes the form

$$a(\tilde{x}) \;=\; a_0 \;+\; a_1 \tilde{x} \;+\; a_2 \tilde{x}^2$$

where again $\tilde{x}$ is the cell-centric scaled coordinate:

$$\tilde{x} \;=\; (x - x_M) / \Delta x$$

We can relate the interpolation parabola's coefficients to the x-moments of the distribution, averaged over y, via:

$$a_1 \;=\; 12 \langle a \tilde{x} \rangle$$

$$a_2 \;=\; 15 \left(12 \langle a \tilde{x}^2 \rangle \;-\; \langle a \rangle \right)$$

$$a_0 \;=\; \langle a \rangle \;-\; a_2 / 12$$

We can invert these relationships to find the x-moments of the distribution, averaged over y, from the interpolation parabola's coefficients:

$$\langle a \tilde{x} \rangle \;=\; a_1 / 12$$

$$\langle a \rangle \;=\; a_0 \;+\; a_2 / 12$$

$$\langle a \tilde{x}^2 \rangle \;=\; \left(\langle a \rangle \;+\; a_2 / 15 \right) \;/\; 12$$

We can also relate the interpolation parabola's coefficients to the values of the distribution, averaged over y, at the cell edges. Here, for this distribution averaged over y, we denote these cell edge values simply by a_L and a_R :

$$a_L \;=\; a_0 \;-\; \frac{1}{2} a_1 \;+\; \frac{1}{4} a_2$$

$$a_R \;=\; a_0 \;+\; \frac{1}{2} a_1 \;+\; \frac{1}{4} a_2$$

We can invert these relationships to find the interpolation parabola's coefficients from the edge values and the cell-averaged value as follows:

$$a_1 = a_R - a_L$$

$$a_2 = 3\left(a_R + a_L - 2\langle a\rangle\right)$$

$$a_0 = \frac{3}{2}\langle a\rangle - \frac{1}{4}\left(a_R + a_L\right)$$

In terms of the moments, the edge values of the y-averaged distribution are:

$$a_L = \frac{3}{2}\left(20\left\langle a\tilde{x}^2\right\rangle - \langle a\rangle\right) - 6\langle a\tilde{x}\rangle$$

$$a_R = \frac{3}{2}\left(20\left\langle a\tilde{x}^2\right\rangle - \langle a\rangle\right) + 6\langle a\tilde{x}\rangle$$

We can invert these relations to obtain the first and second x-moments in terms of the edge values and the cell average:

$$\langle a\tilde{x}\rangle = \left(a_R - a_L\right)/12$$

$$\left\langle a\tilde{x}^2\right\rangle = \left(3\langle a\rangle + \left(a_R + a_L\right)\right)/\ 60$$

We begin our application of constraints to the interpolation polynomial for the distribution averaged over y knowing the cell average, $\langle a\rangle$, and the first 2 x-moments, $\langle a\tilde{x}\rangle$ and $\left\langle a\tilde{x}^2\right\rangle$. The constraint we wish to apply is that no value of this interpolation parabola within the cell shall lie outside the range from 0 to 1. We therefore first examine the cell average itself. If it is negative, we reset it to 0, and then we obviously should also reset both $\langle a\tilde{x}\rangle$ and $\left\langle a\tilde{x}^2\right\rangle$ to 0. Clearly, it is also a good idea to reset $\langle a\tilde{x}\tilde{y}\rangle$, $\langle a\tilde{y}\rangle$, and $\left\langle a\tilde{y}^2\right\rangle$ to 0. If $\langle a\rangle$ exceeds 1, then we reset it to 1, we reset $\langle a\tilde{x}\rangle$, $\langle a\tilde{y}\rangle$, and $\langle a\tilde{x}\tilde{y}\rangle$ to 0, and we reset $\left\langle a\tilde{x}^2\right\rangle$ and $\left\langle a\tilde{y}^2\right\rangle$ to 1/12. For the 9-moment method, the 3 high-order moments are also reset.

At this point, we calculate the values of the distribution (averaged over y) at the left- and right-hand cell edges using the formulae given above. If either value lies outside the range from 0 to 1, we reset it appropriately. Now it proves most useful to compute the coefficients, a_1 and a_2,

of the interpolation parabola. We could compute other quantities as well, but this would be wasteful. We use the formulae given earlier:

$$a_1 = a_R - a_L$$

$$a_2 = 3\left(a_R + a_L - 2\langle a \rangle\right)$$

We now examine the interpolation parabola to see if it has an extremum within the cell which exceeds the allowed range of values. If it does, we will flatten the parabola until the value at its extremum is permissible. First we consider the case of a minimum inside the cell where the value is negative. To have a minimum inside the cell, we need to have the derivative negative at the left-hand edge and positive at the right-hand edge. The derivative values at the cell edges are:

$$\left.\frac{\partial a}{\partial \tilde{x}}\right|_L = a_1 - a_2$$

$$\left.\frac{\partial a}{\partial \tilde{x}}\right|_R = a_1 + a_2$$

Therefore, if $a_2 > |a_1|$, we clearly have a minimum inside the cell. We also note that if $-a_2 > |a_1|$, we have a maximum inside the cell. If there is an extremum inside the cell, then it occurs at a value of $\tilde{x}$ where $\partial a / \partial \tilde{x}$ vanishes. Since

$$\frac{\partial a}{\partial \tilde{x}} = a_1 + 2 a_2 \tilde{x}$$

it is clear that the extremum occurs at $\tilde{x}_{ext}$, given by

$$\tilde{x}_{ext} = -\frac{a_1}{2 a_2}$$

The value of the interpolation parabola at this point is

$$a_{ext} = a_0 - \frac{a_1^2}{4 a_2}$$

If the value of our interpolation parabola is out of range at such an extremum, we will reduce the magnitudes of both a_1 and a_2 together by a common reduction factor, f_{reduce}, so that this

extremum is brought just to the limit of our allowable range of values. By reducing both a_1 *and* a_2 *together, it is clear from our formula for* $\tilde{x}_{ext}$ *that this point at which the extremum occurs does not change. In the reduction process, which causes the interpolation parabola to become flattened, we must be careful not to alter the value of the cell average,* $\langle a \rangle$ *. If this cell average had been out of the allowable range at the outset of this constraint procedure, of course we would have reset it. Hence sufficient flattening to bring the entire parabola into the allowable range must be possible. To preserve the value of the cell average, we have only to determine the constant coefficient in the interpolation parabola by the formula*

$$a_0 \;=\; \langle a \rangle \;-\; a_2/12$$

that was given earlier. Actually, we will have no need to evaluate this coefficient during our constraint procedure.

In order to determine the proper value of the reduction factor f_{reduce} *, we require that, for positive values of* a_2 *, the minimum value must vanish. Hence*

$$a_{0reduced} \;=\; \langle a \rangle \;-\; \frac{f_{reduce}\, a_2}{12} \;=\; f_{reduce} \left(\frac{a_1^2}{4 a_2} \right)$$

so that

$$f_{reduce} \;=\; 12 \langle a \rangle \;/\; \left\{ a_2 \left[1 \;+\; 3 \; (a_1 / a_2)^2 \right] \right\}$$

It is also worth noting, with respect to the efficiency of the computation, that the division in the denominator in this formula need not be removed, since the ratio appearing there will already have been computed in the evaluation of the (out of range) extremum value.

For the case where a_2 *is negative and we have a maximum value greater than 1 inside the cell, we arrive at similar formulae. In this case we determine* f_{reduce} *from the demand that the maximum equal 1. This requirement translates into the demand that*

$$a_{0reduced} \;-\; 1 \;=\; \langle a \rangle \;-\; 1 \;-\; \frac{f_{reduce}\, a_2}{12} \;=\; f_{reduce} \left(\frac{a_1^2}{4 a_2} \right)$$

so that

$$f_{reduce} \;=\; 12\,\left(\langle a\rangle - 1\right) \,/\, \left\{a_2\left[1 \,+\, 3\,(a_1/a_2)^2\right]\right\}$$

It is useful to determine f_{reduce} without having to first evaluate the extremum value. From our earlier formula for the extremum value, we see that we can express it as follows:

$$a_{ext} \;=\; a_0 \;-\; \frac{a_1^2}{4\,a_2} \;=\; \langle a\rangle \;-\; \frac{a_2}{12} \;-\; \frac{a_1^2}{4\,a_2} \;=\; \langle a\rangle \;-\; \frac{a_2}{12}\left[1 \,+\, 3\,(a_1/a_2)^2\right]$$

Clearly, this extremum value will be negative if:

$$a_2 \;>\; |a_1| \qquad \text{and} \qquad \frac{a_2}{12}\left[1 \,+\, 3\,(a_1/a_2)^2\right] \;>\; \langle a\rangle$$

The extremum value will exceed unity if:

$$-a_2 \;>\; |a_1| \qquad \text{and} \qquad -\,\frac{a_2}{12}\left[1 \,+\, 3\,(a_1/a_2)^2\right] \;>\; 1 \,-\, \langle a\rangle$$

We can eliminate division operations by noting that the extremum will be negative if:

$$a_2 \;>\; |a_1| \qquad \text{and} \qquad a_2^2 \,+\, 3\,a_1^{\,2} \;>\; 12\,a_2\,\langle a\rangle$$

In this case, we set

$$f_{reduce} \;=\; 12\,a_2\,\langle a\rangle \,/\, \left(a_2^2 \,+\, 3\,a_1^{\,2}\right)$$

The extremum value will exceed unity if:

$$-a_2 \;>\; |a_1| \qquad \text{and} \qquad a_2^2 \,+\, 3\,a_1^{\,2} \;>\; -12\,a_2\left(1 \,-\, \langle a\rangle\right)$$

In this case, we set

$$f_{reduce} \;=\; -12\,a_2\left(1 \,-\, \langle a\rangle\right) \,/\, \left(a_2^2 \,+\, 3\,a_1^{\,2}\right)$$

In all other cases, we of course set f_{reduce} to unity. Then, naturally, we modify a_1 and a_2 by multiplying each of them by the reduction factor f_{reduce}. However, we want to make these adjustments of a_1 and a_2 only in cases where we have not reset either of the edge values to values at the limits of our allowable range, that is, to 0 or to 1. Thus, before applying the reduction

factor, we check for these cases and set f_{reduce} to 1 if either of the edge values have been reset. In the case that we have reset one of the edge values, we will apply a different constraint, described below, to our interpolation polynomial.

The idea here is that we reduce the magnitudes of a_1 and a_2 together, flattening the interpolation parabola, in cases where our grid cell is likely to be located in between regions where a is either 0 or 1. Then we are likely to have in our cell a segment of a thin, unresolved strip in which the distribution a is either 1 or 0 (respectively). It is entirely appropriate to describe this unresolved strip segment using a parabola that has an extremum inside our grid cell. We must however be careful not to let the extremum value poke outside of our allowable range. If, however, we have reset either edge value to one of our limiting allowable values, that is, to either 0 or 1, then we expect that our cell is located next to a region in which the distribution a is either 0 or 1 (respectively). In this case, it would be inappropriate to simply flatten our interpolation parabola, which would cause the value at the cell edge to move away from the limiting value shared by the region adjacent to it. Instead, we would like to keep the edge value at this limit, if that is possible.

Although we could devise a test for this last case discussed above by examining the average values of a in adjacent cells, we will nevertheless simply assume that if we have reset one of our edge values, either a_L or a_R, to either of our limiting values, 0 or 1, then our cell is located adjacent to a solid region in which the distribution has this limiting value. In this case, we will demand that our interpolation parabola be monotone within the cell. It is somewhat tedious, but nevertheless enlightening, to consider all the cases that might arise when we have reset the value at the right-hand cell edge, a_R, to 1. In this case, we can be certain that the average slope of our interpolation parabola within the cell, a_1, is non-negative. If $|a_2| \le |a_1|$, we know that the interpolation parabola is monotone inside our cell. Since both edge values and the cell average have been constrained to the allowable range, this monotone interpolation parabola must be acceptable. If, however, $-a_2 > a_1 \ge 0$, then the interpolation parabola assumes a

maximum value inside the cell that exceeds 1. In this case, we reset the left-hand edge value, a_L, so that the slope of the parabola vanishes at the right-hand edge of the cell. We therefore reset a_L so that $a_L = 3\langle a \rangle - 2a_R = 3\langle a \rangle - 2$. In terms of the parabola's coefficients, we reset a_1 and a_2 so that $a_1 = -a_2 = 3(1 - \langle a \rangle)$. Now we consider the case where $a_2 > a_1 \geq 0$. In this case the interpolation parabola assumes a minimum value inside the cell. We could consider adjusting the left-hand edge value, a_L, so that this minimum value, if originally negative, is just 0. However, this would not be an appropriate interpolation polynomial if indeed our cell is at the edge of a region, to its right, where the distribution a is unity. Therefore, in this case we reset the right-hand edge value, a_R, so that the slope of the parabola vanishes at the left-hand edge of the cell. We therefore reset a_R so that $a_R = 3\langle a \rangle - 2a_L$. In terms of the parabola's coefficients, we reset a_1 and a_2 so that $a_1 = a_2 = 3(\langle a \rangle - a_L)$. Of course, in all cases we demand that the cell average be unchanged, which is guaranteed so long as we set the constant coefficient of the interpolation parabola to the value $a_0 = \langle a \rangle - a_2/12$.

This resetting of the interpolation parabola's coefficients that we have described above for the case where the value at the right-hand edge of the cell is unity is just the procedure that we use to constrain interpolation parabolae in the PPM scheme (although in PPM this is only performed if a careful examination of the local behavior of the interpolated function reveals that it cannot be considered as smooth). In our context, we begin the application of this PPM monotonicity constraint only in the case where we have a cell edge value equal to either 0 or 1, and we know at this point that both edge values and the cell average are in the permissible range. Then we simply examine the magnitudes of the coefficients a_1 and a_2. If $|a_2| > |a_1|$, it is clear that we want the slope of our interpolation parabola to vanish at the cell edge where its value is closest to the average value in the cell, $\langle a \rangle$. This will of course guarantee that the value at this edge is also the extremum value, so that the parabola will be monotone within our cell. Now, the demand of a vanishing derivative and a particular limiting value at one cell edge, plus the demand that the

cell average remain unchanged, serves to completely determine our interpolation parabola. The demand that the slope of the parabola vanishes at one cell edge implies that $|a_2| = |a_1|$. The value at the opposite cell edge, along with the value of the cell average, $\langle a \rangle$, then determines the magnitude of these interpolation parabola coefficients. From the various relevant relations, we find the following constraint procedure, familiar from the PPM scheme, where the first 2 cases are for negative and the second 2 cases for positive second derivatives in the cell:

If $a_L = 0$ or $a_R = 1$, and $-a_2 > |a_1|$, we set $-a_2 = a_1 = 3\left(a_R - \langle a \rangle\right)$.

If $a_L = 1$ or $a_R = 0$, and $-a_2 > |a_1|$, we set $-a_2 = -a_1 = 3\left(a_L - \langle a \rangle\right)$.

If $a_L = 0$ or $a_R = 1$, and $a_2 > |a_1|$, we set $a_2 = a_1 = 3\left(\langle a \rangle - a_L\right)$.

If $a_L = 1$ or $a_R = 0$, and $a_2 > |a_1|$, we set $a_2 = -a_1 = 3\left(\langle a \rangle - a_R\right)$.

At the end of this resetting procedure, we recompute the moments of the distribution via

$$\langle a\tilde{x} \rangle = a_1/12$$

$$\langle a\tilde{x}^2 \rangle = \left(\langle a \rangle + a_2/15\right) / 12$$

so that we are ready to proceed with the advection algorithm, which we will formulate to work from a prescribed set of moments.

The above constraint procedure stands in strong contrast to that used in the PPM scheme. Because the PPB scheme does such a careful job of computing the moments of the distribution within each grid cell, we do not need to examine the behavior of the distribution outside of a single cell in order to determine appropriate constraints to apply. We are here considering the case of a distribution whose values are known to lie strictly between 0 and 1. If we did not have any such knowledge of an allowable range of values, we would have two choices: (1) we could simply let the advected function oscillate near unresolved sharp features, or (2) we could examine the behavior of the function within the cell of interest and its two neighbors in the direction of the present pass in order to determine whether the function was smooth, so that no constraints need be applied, or not,

so that we might apply the standard PPM monotonicity constraint. I would suggest that the second choice would not be a good one, since it would tend to destroy the resolving power of the PPB approach. One might tone down the drastic dissipation that the PPM monotonicity constraint would imply by instead blending the constrained distribution together with the original one with weight factors of, say, 0.25 and 0.75, respectively. One would want, as in the PPM scheme, to change these weight factors continuously over a range in which the interpolated function goes from smooth to unresolved, with weights of 0.25 and 0.75, or the like, applying only at the unresolved end of this range. However, the PPB approach, without any such constraints, does not result in oscillations of very significant size, and therefore one might prefer simply to live with them. Such a choice, of course, would depend upon the application. We have performed no tests with choice number (2) above, since we target either advection of a phase space fluid, in which we would want to apply only a positivity constraint, and advection of the fractional volume of a particular type of fluid or material, in which the constraints described above in detail seem most appropriate. Tests for both these types of problems are presented below.

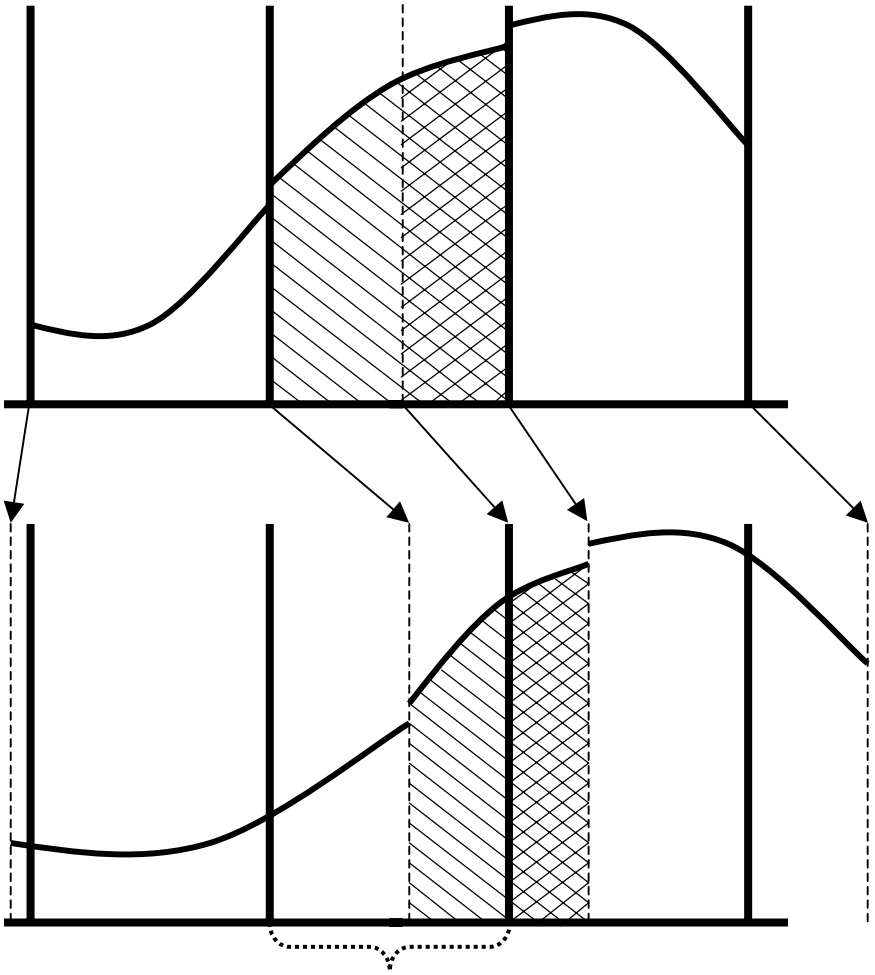
Updating the Moments in a 1-D Pass of the PPB Advection Scheme

In the previous section, we described the application of constraints on the interpolation polynomials used by the PPB moment-conserving advection schemes. It remains to lay out the details of how the moments conserved by the schemes are updated in a single 1-D pass of the algorithm. We will specialize this discussion to the case of the PPB6 scheme, which updates only 6 rather than the full 9 moments. This is the only scheme that has been implemented in the wind tunnel flow simulation program described later. I will try to point out where appropriate how the 6-moment PPB6 scheme can be extended to update all 9 moments in 2-D.

It is easiest to first consider how we might update the moments $\langle a \rangle$, $\langle a\tilde{x} \rangle$, and $\langle a\tilde{x}^2 \rangle$ in a 1-D pass treating advection only in the x-direction. We will assume that some other computation, such as for example PPM hydrodynamics, produces for us the time-averaged x-velocities at the cell interfaces. We will also assume that we have constrained the parabolae in all the grid cells, and that the moments of the distribution in each grid cell reflect this. The diagram on the following page indicates this situation. Three contiguous grid cells are shown, and in each the interpolation parabola determined by the moments $\langle a \rangle$, $\langle a\tilde{x} \rangle$, and $\langle a\tilde{x}^2 \rangle$ is shown. The motion of the grid cells under the action of the prescribed velocity field is also shown, and we have indicated that each interpolation parabola is simply either stretched or compressed in the horizontal dimension by the action of this velocity field. We assume that the values of the distribution a are conserved along stream lines in the flow and that we may use a simple linear interpolation to obtain the time-averaged velocity at any value of the x-coordinate. We will also simplify our calculation by assigning a velocity, so interpolated, to each point along the x-dimension at the beginning of the 1-D pass and holding that velocity constant in time along the streamline emanating from that location. Thus we assign the prescribed time-averaged interface velocity to each streamline emating from a cell interface. We thus equate the time-averaged cell interface

velocity in the Lagrangian coordinate with that in the Eulerian one. This procedure is of course not correct, but it will do well enough for our purpose.

In the figure we can clearly see that after each cell has moved in the x-direction, the distributions inside the new Eulerian grid cells are discontinuous. These new distributions must be replaced by continuous parabolae. To determine these parabolae, we evaluate the moment integrals for each Eulerian grid cell interval. This procedure is simple in concept, but complicated in practice. In general, we need to evaluate up to 3 separate contributions to each moment integral for the new Eulerian grid cell: a contribution from each neighboring cell and one from the original cell. To evaluate these individual contributions to each moment integral,



The interpolation parabolae for 3 grid cells are shown at the beginning of the x-pass in the upper part of the figure. The motion of the cell interfaces is indicated, and the new stretched or squashed parabolae are shown in the lower part of the figure. The two portions of the central cell that become parts of the new central cell and of its neighbor on the right are indicated by the diagonal and cross-hatched shading patterns in both parts of the figure. To obtain the new interpolation parabola for the central cell, we must evaluate the moment integrals over the cell domain in x, which is indicated by the bracket at the bottom of the figure.

we need expressions for the integrals of the distribution, $a(\tilde{x})$, multiplied by powers of $\tilde{x}$. These integrals must be carried out over subregions of the grid cell such as the areas shown in the figure with diagonal shading and with cross-hatched shading. In the upper part of the figure, these two regions indicate portions of the original cell volume that will be advected into the new Eulerian cell on the right (the cross-hatched section) and that will remain in the original cell (diagonal shading). In the most complicated case, there are 3 regions of the grid cell over which we need to evaluate contributions to moment integrals in new Eulerian cells. The grid cell on the left in the upper portion of the figure is such a complex case. Material within it will contribute to the new moment integrals in both neighbor cells as well as the cell itself.

Using the subscript N to denote quantities taken at the new time level, the new moments we need to evaluate in the Eulerian grid cells are:

$$\left\langle a \tilde{x}^k \right\rangle_N \;=\; \left\langle a_N \tilde{x}_N^k \right\rangle \;=\; \int_{-\frac{1}{2}}^{\frac{1}{2}} d\tilde{x}_N \; a_N(\tilde{x}_N)\, \tilde{x}_N^k$$

where k ranges from 0 to 2. We denote the time-averaged velocities of the left- and right-hand edges of the grid cell by $\bar{u}_{xL}$ and $\bar{u}_{xR}$. The bar over these symbols is intended to indicate that they represent time averages over the time step. Now, writing

$$\bar{u}_{x0} \;=\; (\bar{u}_{xL} \,+\, \bar{u}_{xR}) \;/\; 2$$

$$\Delta\bar{u}_x \;=\; \bar{u}_{xR} \,-\, \bar{u}_{xL}$$

we can express the new coordinate $\tilde{x}_N$ associated with the old coordinate $\tilde{x}$ by

$$\tilde{x}_N \;=\; \tilde{x} \;+\; (\bar{u}_{x0} \,+\, \Delta\bar{u}_x\, \tilde{x})\; (\Delta t / \Delta x)$$

We will find it convenient to denote the factor by which the grid cell expands during the time step by $f_{\exp}$ and the average distance, measured in cell widths, that the material in our grid cell moves to the right during the time step (the average Courant number) by σ_0. Thus

$$f_{\exp} \;=\; 1 \;+\; \Delta\bar{u}_x\, \Delta t / \Delta x$$

$$\sigma_0 \;=\; \bar{u}_{x0}\, \Delta t / \Delta x$$

We may therefore write

$$\tilde{x}_N \;=\; f_{\exp}\,\tilde{x} \;+\; \sigma_0$$

In evaluating the contributions from our grid cell of interest (the central one in the figure shown earlier) to the new moments in the new Eulerian grid cells, we will use the assumption that the value of the distribution a is conserved along streamlines. For example, we can write the contribution, $dmom_{kR}$, to the k^{th} moment in the cell to the right from the cross-hatched region in our cell of interest as follows:

$$dmom_{kR} \;=\; \int_{\xi_R}^{1/2} d\tilde{x}\,(d\tilde{x}_{NZR}/d\tilde{x})\,a(\tilde{x})\,\tilde{x}_{NZR}^{k}$$

In this formula, we have used the subscript ZR to denote the zone (or cell) on the right. We can simplify this formula by recognizing that $(d\tilde{x}_{NZR}/d\tilde{x})$ is just $f_{\exp}$ and that

$$\tilde{x}_{NZR} \;=\; \tilde{x}_N \;-\; 1 \;=\; f_{\exp}\,\tilde{x} \;+\; \sigma_0 \;-\; 1$$

The lower limit on the integration, ξ_R, is the value of $\tilde{x}$ from which the streamline emanates that just reaches the cell interface at the end of the time step. Thus

$$1/2 \;=\; f_{\exp}\,\xi_R \;+\; \sigma_0$$

Hence

$$\xi_R \;=\; \left(1/2 \;-\; \sigma_0\right) \,/\, f_{\exp}$$

We will define the Courant number σ_R for the right-hand cell edge by

$$\sigma_R \;=\; 1/2 \;-\; \xi_R$$

We now rewrite the contribution to the new k^{th} moment in the cell to the right as

$$dmom_{kR} \;=\; f_{\exp} \int_{\xi_R}^{1/2} d\tilde{x}\; a(\tilde{x})\; \left(f_{\exp}\,\tilde{x} \;+\; \sigma_0 \;-\; 1\right)^k$$

It is a straightforward matter to write this integral in terms of the simpler integrals

$$dmm_{kR} \;=\; \int_{\xi_R}^{1/2} d\tilde{x}\; a(\tilde{x})\;\; \tilde{x}^k$$

These simpler integrals are given by

$$dmm_{kR} \;=\; \sigma_R \left(a_0 \, D_k \;+\; a_1 \, D_{k+1} \;+\; a_2 \, D_{k+2} \right)$$

where the constants D_k *can be evaluated recursively via* $D_0 \;=\; 1$ *and*

$$D_k \;=\; \frac{1}{(k+1)\, 2^k} \;+\; \frac{k}{k+1} \xi_R \, D_{k-1}$$

Therefore:

$$D_0 \;=\; 1$$

$$D_1 \;=\; \frac{1}{4} \;+\; \frac{1}{2} \xi_R \, D_0$$

$$D_2 \;=\; \frac{1}{12} \;+\; \frac{2}{3} \xi_R \, D_1$$

$$D_3 \;=\; \frac{1}{32} \;+\; \frac{3}{4} \xi_R \, D_2$$

$$D_4 \;=\; \frac{1}{80} \;+\; \frac{4}{5} \xi_R \, D_3$$

We may now write

$$dmom_{0R} \;=\; f_{\exp} \, dmm_{0R}$$

$$dmom_{1R} \;=\; f_{\exp}^2 \, dmm_{1R} \;+\; (\sigma_0 - 1) \, dmom_{0R}$$

$$dmom_{2R} \;=\; f_{\exp}^3 \, dmm_{2R} \;+\; (\sigma_0 - 1) \left(f_{\exp}^2 \, dmm_{1R} \;+\; dmom_{1R} \right)$$

In the above formulae, we have taken special care to write the expressions in a form that is highly efficient for computation.

It is easiest to obtain the contribution, $dmom_{kC}$, *from the diagonally shaded region in our figure to the new* k^{th} *moment in the central Eulerian grid cell by subtracting the portion from the cross-hatched region, which is advected into the cell on the right, from the new total* k^{th} *moment integral below as follows:*

$$dmom_{kC} \;=\; f_{\exp} \int_{-1/2}^{1/2} d\tilde{x} \; a(\tilde{x}) \left(f_{\exp} \, \tilde{x} \;+\; \sigma_0 \right)^k \;-\; f_{\exp} \int_{\xi_R}^{1/2} d\tilde{x} \; a(\tilde{x}) \left(f_{\exp} \, \tilde{x} \;+\; \sigma_0 \right)^k$$

The trick here is to relate the second integral on the right to $dmom_{kR}$, which we have already evaluated, and to relate the first integral on the right to the original moments in the cell. Denoting the first integral on the right above by $dmom_{kT}$, we have

$$dmom_{kT} = f_{\exp} \int_{-1/2}^{1/2} d\tilde{x}\; a(\tilde{x}) \left(f_{\exp} \tilde{x} + \sigma_0 \right)^k$$

so that:

$$dmom_{0T} = f_{\exp} \langle a \rangle$$

$$dmom_{1T} = f_{\exp}^2 \langle a \tilde{x} \rangle + \sigma_0\, dmom_{0T}$$

$$dmom_{2T} = f_{\exp}^3 \langle a \tilde{x}^2 \rangle + \sigma_0 \left(f_{\exp}^2 \langle a \tilde{x} \rangle + dmom_{1T} \right)$$

Denoting the second integral on the right in the above expression for $dmom_{kC}$ by $dmom_{kRC}$, we have

$$dmom_{kRC} = f_{\exp} \int_{\xi_R}^{1/2} d\tilde{x}\; a(\tilde{x}) \left(f_{\exp} \tilde{x} + \sigma_0 \right)^k$$

so that:

$$dmom_{0RC} = dmom_{0R}$$

$$dmom_{1RC} = dmom_{1R} + dmom_{0RC}$$

$$dmom_{2RC} = dmom_{2R} + dmom_{1R} + dmom_{1RC}$$

The above integrals could of course also be computed directly, as we did for the integrals in $dmom_{kR}$, but that would involve more arithmetic than the formulae given here.

If we were to encounter only advection to the right in every problem, our job of describing how to update the x-moments of the distribution of the variable a in a single 1-D pass would now be complete, since we would have only to add the contributions in $dmom_{kR}$ from the zone on the left to those in $dmom_{kC}$ in order to obtain $\langle a \tilde{x}^k \rangle_N$. However, of course we must also account for leftward advection. We will do this in a manner that will allow the resulting program to employ vectorizable logic. Thus we will seek formulae for leftward advection that resemble as

closely as possible the formulae given above for advection to the right. Most every CPU chip now in production will be able to capitalize upon our vectorizable form of the computation, using short 1-D scratch vectors stored in its off-chip cache memory.

To calculate the leftward advection in the same vectorizable loop with the rightward advection, it is most natural to loop over the cell interfaces, computing at each interface the contribution, either $dmom_{kR}$ above or $dmom_{kLZR}$, to the new k^{th} moment in the cell to the right or left of the interface. Just as $dmom_{kR}$ is the contribution from the cell of interest to the new k^{th} moment in the cell to the right, $dmom_{kLZR}$ is the contribution to the new k^{th} moment in the cell of interest from the cell on the right. (Here the subscript LZR refers to the left-hand interface of the zone, or cell, on the right.) In the formulae for leftward advection that follow, we will instead refer to advection out of the cell of interest across its left-hand interface and into the cell on the left. This will allow us to avoid using clumsey subscripts. In putting these formulae together with those given earlier for rightward advection, it is a simple matter to get the subscripts correct and to implement differences between the signs in the formulae via multiplications by a variable that is ± 1 according to the sign of the time-averaged advection velocity at the interface.

In the case of leftward advection across the left-hand interface, we will have $\overline{u}_{xL}$ negative. Iti is not necessary that $\overline{u}_{x0}$ also be negative, but it is likely. We can think of the left-most cell in our earlier figure as an example, and in this case, although $\overline{u}_{xL}$ is indeed negative, $\overline{u}_{x0}$ is positive. Understanding that σ_0 may be negative, we may still write

$$\tilde{x}_N \;=\; f_{\exp}\,\tilde{x} \;+\; \sigma_0$$

We can write the contribution, $dmom_{kL}$, to the k^{th} moment in the cell to the left from the appropriate region in our cell of interest as follows:

$$dmom_{kL} \;=\; \int_{-\frac{1}{2}}^{\xi_L} d\tilde{x}\,\left(d\tilde{x}_{NZL}/d\tilde{x}\right)a(\tilde{x})\,\tilde{x}_{NZL}^{k}$$

In this formula, we have used the subscript ZL to denote the zone (or cell) on the left. Once again we notice that $(d\tilde{x}_{NZL}/d\tilde{x})$ is just $f_{\exp}$ and that

$$\tilde{x}_{NZL} \;=\; \tilde{x}_N \;+\; 1 \;=\; f_{\exp}\,\tilde{x} \;+\; \sigma_0 \;+\; 1$$

The upper limit on the integration, ξ_L, is the value of $\tilde{x}$ from which the streamline emanates that just reaches the left-hand cell interface at the end of the time step. Thus

$$-\tfrac{1}{2} \;=\; f_{\exp}\,\xi_L \;+\; \sigma_0$$

Hence

$$\xi_L \;=\; -\left(\tfrac{1}{2} \;+\; \sigma_0\right) \,/\, f_{\exp}$$

We define the Courant number σ_L for the left-hand cell edge by

$$\sigma_L \;=\; \tfrac{1}{2} \;+\; \xi_L$$

We now rewrite the contribution to the new k^{th} moment in the cell to the left as

$$dmom_{kL} \;=\; f_{\exp} \int_{-\frac{1}{2}}^{\xi_L} d\tilde{x}\; a(\tilde{x})\; \left(f_{\exp}\,\tilde{x} \;+\; \sigma_0 \;+\; 1\right)^k$$

It is a straightforward matter to write this integral in terms of the simpler integrals

$$dmm_{kL} \;=\; \int_{-\frac{1}{2}}^{\xi_L} d\tilde{x}\; a(\tilde{x})\; \tilde{x}^k$$

These simpler integrals are given by

$$dmm_{kL} \;=\; \sigma_L \left((-1)^k a_0\, C_k \;+\; (-1)^{k+1} a_1\, C_{k+1} \;+\; (-1)^{k+2} a_2\, C_{k+2}\right)$$

where the constants C_k can be evaluated recursively via

$$C_k \;=\; \frac{1}{(k+1)\,2^k} \;-\; \frac{k}{k+1}\,\xi_L\, C_{k-1}$$

Therefore:

$$C_0 \;=\; 1$$

$$C_1 \;=\; \frac{1}{4} \;-\; \frac{1}{2}\,\xi_L\, C_0$$

$$C_2 \;=\; \frac{1}{12} \;-\; \frac{2}{3}\,\xi_L\, C_1$$

$$C_3 = \frac{1}{32} - \frac{3}{4}\xi_L\, C_2$$

$$C_4 = \frac{1}{80} - \frac{4}{5}\xi_L\, C_3$$

If we carry out a careful comparison of these formulae for the C_k *with the earlier formulae for the* D_k, *we realize that these constants are essentially the same things, except for a substitution of* $-\xi_L$ *in the place of* ξ_R. *The signs in front of the terms in our expression for* dmm_{kL} *come from the odd or even symmetries about the origin of the integrands involved. We may now write*

$$dmom_{0L} = f_{\exp}\, dmm_{0L}$$

$$dmom_{1L} = f_{\exp}^2\, dmm_{1L} + (\sigma_0 + 1)\, dmom_{0L}$$

$$dmom_{2L} = f_{\exp}^3\, dmm_{2L} + (\sigma_0 + 1)\left(f_{\exp}^2\, dmm_{1L} + dmom_{1L}\right)$$

In a single vectorizable loop it is a straightforward matter to compute at each cell interface the contribution to the new k^{th} *moment in the downstream grid cell that arises from advection across the interface from the upstream grid cell. This contribution will be either the quantity we have called above* $dmom_{kL}$ *or it will be* $dmom_{kR}$. *Once this computation is complete, we execute a second vectorizable loop over grid cells rather than grid cell interfaces in which we accumulate the various contributions to the new moments in the cells. For leftward advection, the computations in this second loop are described below.*

We once again obtain the contribution, $dmom_{kC}$, *from the cell of interest to the new* k^{th} *moment in this same cell by subtracting the contribution from the region just discussed, which is advected into the cell on the left, from the new total* k^{th} *moment integral below as follows:*

$$dmom_{kC} = f_{\exp} \int_{-1/2}^{1/2} d\tilde{x}\; a(\tilde{x})\, \left(f_{\exp}\, \tilde{x} + \sigma_0\right)^k - f_{\exp} \int_{-1/2}^{\xi_L} d\tilde{x}\; a(\tilde{x})\, \left(f_{\exp}\, \tilde{x} + \sigma_0\right)^k$$

The trick is once again to relate the second integral on the right to $dmom_{kL}$, *which we have already evaluated, and to relate the first integral on the right to the original moments in the cell. The treatment of the first integral on the right, which results in the computation of the quantities*

we have called $dmom_{kT}$, is identical to that discussed earlier for rightward advection. Denoting the second integral on the right in the above expression for $dmom_{kC}$ by $dmom_{kLC}$, we have

$$dmom_{kLC} \;=\; f_{\exp} \int_{-1/2}^{\xi_L} d\tilde{x}\; a(\tilde{x})\; \left(f_{\exp}\, \tilde{x} \;+\; \sigma_0 \right)^k$$

so that:

$$dmom_{0LC} \;=\; dmom_{0L}$$

$$dmom_{1LC} \;=\; dmom_{1L} \;-\; dmom_{0LC}$$

$$dmom_{2LC} \;=\; dmom_{2L} \;-\; dmom_{1L} \;-\; dmom_{1LC}$$

To compute the new moments in the grid cells is now simple. Because of the nomenclature we have used, which, believe it or not, has been chosen to avoid confusion, we give the four possible cases:

$$\left\langle a\tilde{x}^k \right\rangle_N \;=\; dmom_{kT} \;-\; dmom_{kLC} \;+\; dmom_{kLZR}\,, \qquad when \quad \bar{u}_{xL} < 0 \quad and \quad \bar{u}_{xR} < 0$$

$$\left\langle a\tilde{x}^k \right\rangle_N \;=\; dmom_{kT} \;-\; dmom_{kLC} \;-\; dmom_{kRC}\,, \qquad when \quad \bar{u}_{xL} < 0 \quad and \quad \bar{u}_{xR} > 0$$

$$\left\langle a\tilde{x}^k \right\rangle_N \;=\; dmom_{kT} \;+\; dmom_{kRZL} \;+\; dmom_{kLZR}\,, \qquad when \quad \bar{u}_{xL} > 0 \quad and \quad \bar{u}_{xR} < 0$$

$$\left\langle a\tilde{x}^k \right\rangle_N \;=\; dmom_{kT} \;+\; dmom_{kRZL} \;-\; dmom_{kRC}\,, \qquad when \quad \bar{u}_{xL} > 0 \quad and \quad \bar{u}_{xR} > 0$$

These four cases are easily handled all at once, using multiplications by sign variables and vectorizable logic.

The new x-moments that we have just computed above may not imply a new interpolation parabola that satisfies the constraints we so carefully applied at the beginning of this 1-D pass. However, we will apply those constraints again at the beginning of the next 1-D pass, so there is no need to be concerned with these constraints further here. The computations set out above give us 3 new moments out of the 6 or 9 that we require for a 2-D advection problem. In the PPB method that updates 9 moments independently, we have only to repeat the above computations

precisely two times, once for the moments $\langle a\tilde{y}\rangle$, $\langle a\tilde{x}\tilde{y}\rangle$, $\langle a\tilde{x}^2\tilde{y}\rangle$ serving in place of the moments $\langle a\rangle$, $\langle a\tilde{x}\rangle$, $\langle a\tilde{x}^2\rangle$ and once again for the moments $\langle a\tilde{y}^2\rangle$, $\langle a\tilde{x}\tilde{y}^2\rangle$, $\langle a\tilde{x}^2\tilde{y}^2\rangle$ serving in place of the moments $\langle a\rangle$, $\langle a\tilde{x}\rangle$, $\langle a\tilde{x}^2\rangle$. For these calculations, we do not need to apply the constraints described for our first set of 3 moments. The reason is that the y-moments are constrained only in the y-pass, and constraining the mixed moments, which describe the behavior of the distribution along the cell diagonals, is optional. In determining the algorithm for the PPB6 method, which conserves only 6 moments, we insert into the 9-moment algorithm the constraint that the 3 highest-order moments are determined in terms of the other 6. This results in dramatic simplification of the procedure for updating the three remaining moments, $\langle a\tilde{y}\rangle$, $\langle a\tilde{x}\tilde{y}\rangle$, and $\langle a\tilde{y}^2\rangle$. The PPB6 scheme involves roughly half the computational labor in 2-D of the PPB scheme that updates all 9 moments.

For the 1-D pass of a 3-D calculation, we would, in the 27-moment PPB scheme, simply update 6 additional sets of 3 moments using precisely the same algorithm just described. This could be done by simply calling the same subroutine 6 more times with 6 new sets of arguments. These 6 additional sets of moments would give the x-distributions of the first and second z-moments and of the yz-moment, the y^2z-moment, and the yz^2-moment. The 3-D version of the PPB6 scheme would update 10 moments. A 1-D pass of this algorithm would update the x-dependence of the z- and z^2-moments just as the x-dependence of the y- and y^2-moments is treated in the 2-D algorithm. Only a single moment, the yz-moment, would remain to be updated, and that could be handled using a simple donor-cell differencing scheme.

```
C                                 Copyright 2002 Paul R. Woodward.
      small = 0.0000001
      almost1 = 0.999999
      third = 1. / 3.
      sixth = 0.5 * third
      twelth = 0.25 * third
      fftnth = 0.2 * third
c                                 This code will soon be made available under
c     nrecips = nrecips + 1       GNU "Open Source" licensing.
c     nmults = nmults + 3
c     nadds = nadds + 0
c
      do   i = 0,n+1
        if (rho(i) .lt. small)   rhoxy(i) = 0.
        if (rho(i) .lt. small)   rho(i) = 0.
        if (rho(i) .lt. small)   rhox(i) = 0.
        if (rho(i) .lt. small)   rhox2(i) = 0.
        if (rho(i) .lt. small)   rhoy(i) = 0.
        if (rho(i) .lt. small)   rhoy2(i) = 0.
        if (rho(i) .gt. almost1)   rhoxy(i) = 0.
        if (rho(i) .gt. almost1)   rho(i) = 1.
        if (rho(i) .gt. almost1)   rhox(i) = 0.
        if (rho(i) .gt. almost1)   rhox2(i) = twelth
        if (rho(i) .gt. almost1)   rhoy(i) = 0.
        if (rho(i) .gt. almost1)   rhoy2(i) = twelth
        thyng = 30. * rhox2(i) - 1.5 * rho(i)
        thang = 6. * rhox(i)
        rhol(i) = thyng - thang
        rhor(i) = thyng + thang
        if (rhol(i) .lt. smlrho)   rhol(i) = 0.
        if (rhol(i) .gt. 1.)   rhol(i) = 1.
        if (rhor(i) .lt. smlrho)   rhor(i) = 0.
        if (rhor(i) .gt. 1.)   rhor(i) = 1.
        rho1(i) = rhor(i) - rhol(i)
        rho2(i) = 3. * (rhol(i) + rhor(i) - 2. * rho(i))
        freduce(i) = 1.
        thing(i) = rho(i) * rho2(i)
        if (rho2(i) .lt. 0.)   thing(i) = thing(i) - rho2(i)
        thang = twelth * rho2(i) * rho2(i)  +  0.25 * rho1(i) * rho1(i)
        if (thang .ne. 0.)   freduce(i) = thing(i) / thang
        if (thang .le. thing(i))   freduce(i) = 1.
        absrho1(i) = rho1(i)
        if (rho1(i) .lt. 0.)   absrho1(i) = - rho1(i)
        absrho2(i) = rho2(i)
        if (rho2(i) .lt. 0.)   absrho2(i) = - rho2(i)
        if (absrho2(i) .le. absrho1(i))   freduce(i) = 1.
        if (rhol(i) .lt. smlrho)   freduce(i) = 1.
        if (rhol(i) .gt. almost1)   freduce(i) = 1.
        if (rhor(i) .lt. smlrho)   freduce(i) = 1.
        if (rhor(i) .gt. almost1)   freduce(i) = 1.
        rho1(i) = freduce(i) * rho1(i)
        rho2(i) = freduce(i) * rho2(i)
        flag(i) = 0.
        if (rhol(i) .lt. smlrho)   flag(i) = 1.
        flagl(i) = 0.
        if (rho2(i) .gt. rho1(i))   flagl(i) = flag(i)
        flag(i) = 0.
```

```
        if (rhol(i) .ge. 1.)   flag(i) = 1.
        if (rho2(i) .lt. rho1(i))   flagl(i) = flagl(i) + flag(i)
        if (flagl(i) .gt. 0.)   rho2(i) = 3. * (rho(i) - rhol(i))
        if (flagl(i) .gt. 0.)   rho1(i) = rho2(i)
        flag(i) = 0.
        if (rhor(i) .lt. smlrho)   flag(i) = 1.
        flagr(i) = 0.
        if (rho2(i) .gt. -rho1(i))   flagr(i) = flag(i)
        flag(i) = 0.
        if (rhor(i) .ge. 1.)   flag(i) = 1.
        if (-rho2(i) .gt. rho1(i))   flagr(i) = flagr(i) + flag(i)
        if (flagr(i) .gt. 0.)   rho2(i) = 3. * (rho(i) - rhor(i))
        if (flagr(i) .gt. 0.)   rho1(i) = - rho2(i)
        rhox(i) = twelth * rho1(i)
        rhox2(i) = twelth * (rho(i)  +  fftnth * rho2(i))
      enddo
c
      nrecips = nrecips + n + 3
      ncvmgms = ncvmgms + n*37 + 74
      nmults = nmults + n*18 + 42
      nadds = nadds + n*30 + 69
c
      third = 1. / 3.
      twothird = 2. * third
      sixth = 0.5 * third
      twelth = 0.25 * third
      dtbydx = dt / (xl(2) - xl(1))
c
      do   i = 1,n+1
        dchil(i) = uxavl(i) * dtbydx
        uxavfar(i) = uxavl(i-1)
        if (uxavl(i) .lt. 0.)   uxavfar(i) = uxavl(i+1)
        dchifar = uxavfar(i) * dtbydx
        dchi0upstrm = 0.5 * (dchil(i) + dchifar)
        sl(i) = 1.
        if (uxavl(i) .lt. 0.)   sl(i) = -1.
        ddchiupstrm = sl(i) * (dchil(i) - dchifar)
        fupstrm(i) = 1. + ddchiupstrm
        xil = (sl(i) * 0.5  -  dchi0upstrm) / fupstrm(i)
        sxil = sl(i) * xil
        sigmal(i) = 0.5 - sxil
        d1(i) = 0.25 + 0.5 * sxil
        d2(i) = twelth + twothird * sxil * d1(i)
        d3 = 0.03125 + 0.75 * sxil * d2(i)
        d4 = 0.0125 + 0.8 * sxil * d3
        rhoupstrm(i) = rho(i-1)
        if (uxavl(i) .lt. 0.)   rhoupstrm(i) = rho(i)
        rhoxupstrm(i) = rhox(i-1)
        if (uxavl(i) .lt. 0.)   rhoxupstrm(i) = rhox(i)
        rhox2upstrm(i) = rhox2(i-1)
        if (uxavl(i) .lt. 0.)   rhox2upstrm(i) = rhox2(i)
        rho1upstrm = 12. * rhoxupstrm(i)
        rho2upstrm = 15. * (12. * rhox2upstrm(i) - rhoupstrm(i))
        rho0upstrm = rhoupstrm(i) - twelth * rho2upstrm
        dmm0l(i) = sigmal(i) * (rho0upstrm
     &                      +  sl(i) * d1(i) * rho1upstrm
     &                      +  d2(i) * rho2upstrm)
```

```
         dmm1l(i) = sigmal(i) * (sl(i) * d1(i) * rho0upstrm
     &                                 +  d2(i) * rho1upstrm
     &                          +  sl(i) * d3 * rho2upstrm)
         dmm2l(i) = sigmal(i) * (d2(i) * rho0upstrm
     &                          +  sl(i) * d3 * rho1upstrm
     &                          +  d4 * rho2upstrm)
         dmom0l(i) = fupstrm(i) * dmm0l(i)
         fupstrmsq(i) = fupstrm(i) * fupstrm(i)
         dchil1(i) = dchi0upstrm - sl(i)
         dmom1l(i) = fupstrmsq(i) * dmm1l(i)  +  dchil1(i) * dmom0l(i)
         dmom2l(i) = fupstrm(i) * fupstrmsq(i) * dmm2l(i)
     &             +  dchil1(i) * (fupstrmsq(i) * dmm1l(i) + dmom1l(i))
       enddo
c
       nrecips = nrecips + n + 3
       ncvmgms = ncvmgms + n*5 + 5
       nmults = nmults + n*38 + 45
       nadds = nadds + n*21 + 30
c
c

       do   i = 1,n
         dchi0(i) = 0.5 * (dchil(i) + dchil(i+1))
         ddchi = dchil(i+1) - dchil(i)
         f(i) = 1. + ddchi
         fsq(i) = f(i) * f(i)
         totmom0nu = f(i) * rho(i)
         totmom1nu = fsq(i) * rhox(i)  +  dchi0(i) * totmom0nu
         totmom2nu = f(i) * fsq(i) * rhox2(i)
     &             +  dchi0(i) * (fsq(i) * rhox(i) + totmom1nu)
         rhonu(i) = totmom0nu + sl(i) * dmom0l(i)
         rhoxnu(i) = totmom1nu + sl(i) * dmom1l(i)
         rhox2nu(i) = totmom2nu + sl(i) * dmom2l(i)
         if (uxavl(i) .lt. 0.)   rhoxnu(i) = rhoxnu(i) + dmom0l(i)
         thyng = 2. * dmom1l(i)  -  dmom0l(i)
         if (uxavl(i) .lt. 0.)   rhox2nu(i) = rhox2nu(i) + thyng
         rhonu(i) = rhonu(i)  -  sl(i+1) * dmom0l(i+1)
         rhoxnu(i) = rhoxnu(i)  -  sl(i+1) * dmom1l(i+1)
         rhox2nu(i) = rhox2nu(i)  -  sl(i+1) * dmom2l(i+1)
         if (uxavl(i+1) .gt. 0.)   rhoxnu(i) = rhoxnu(i) - dmom0l(i+1)
         thyng = 2. * dmom1l(i+1)  +  dmom0l(i+1)
         if (uxavl(i+1) .gt. 0.)   rhox2nu(i) = rhox2nu(i) - thyng
       enddo
c
       ncvmgms = ncvmgms + n*4
       nmults = nmults + n*16 + 3
       nadds = nadds + n*18 + 5
c
c
       do   i = 1,n+1
         rhoyupstrm(i) = rhoy(i-1)
         if (uxavl(i) .lt. 0.)   rhoyupstrm(i) = rhoy(i)
         rhoxyupstrm(i) = rhoxy(i-1)
         if (uxavl(i) .lt. 0.)   rhoxyupstrm(i) = rhoxy(i)
         rhoy1upstrm = 12. * rhoxyupstrm(i)
c        rhoy0upstrm = rhoyupstrm(i)
         dmm0l(i) = sigmal(i) * (rhoyupstrm(i)
```

```
     &                        +  sl(i) * d1(i) * rhoy1upstrm)
        dmm1l(i) = sigmal(i) * (sl(i) * d1(i) * rhoyupstrm(i)
     &                              +  d2(i) * rhoy1upstrm)
        dmom0l(i) = fupstrm(i) * dmm0l(i)
        dmom1l(i) = fupstrmsq(i) * dmm1l(i)  +  dchil1(i) * dmom0l(i)
       enddo
c
       do   i = 1,n
        totmom0nu = f(i) * rhoy(i)
        totmom1nu = fsq(i) * rhoxy(i)  +  dchi0(i) * totmom0nu
        rhoynu(i) = totmom0nu + sl(i) * dmom0l(i)
        rhoxynu(i) = totmom1nu + sl(i) * dmom1l(i)
        if (uxavl(i) .lt. 0.)   then
          rhoxynu(i) = rhoxynu(i) + dmom0l(i)
        endif
        rhoynu(i) = rhoynu(i)  -  sl(i+1) * dmom0l(i+1)
        rhoxynu(i) = rhoxynu(i)  -  sl(i+1) * dmom1l(i+1)
        if (uxavl(i+1) .gt. 0.)   then
           rhoxynu(i) = rhoxynu(i) - dmom0l(i+1)
        endif
       enddo
c
       ncvmgms = ncvmgms + n*4 + 2
       nmults = nmults + n*17 + 13
       nadds = nadds + n*10 + 8
c
c
       do   i = 1,n+1
        rhoy2upstrm(i) = rhoy2(i-1)
        if (uxavl(i) .lt. 0.)   rhoy2upstrm(i) = rhoy2(i)
c       rhoy21upstrm = rhoxupstrm(i)
        rhoy22upstrm = 15. * (rhox2upstrm(i) - twelth * rhoupstrm(i))
        rhoy20upstrm = rhoy2upstrm(i)  -  twelth * rhoy22upstrm
        dmm0l(i) = sigmal(i) * (rhoy20upstrm
     &                        +  sl(i) * d1(i) * rhoxupstrm(i)
     &                        +  d2(i) * rhoy22upstrm)
        dmom0l(i) = fupstrm(i) * dmm0l(i)
       enddo
c
c      ncvmgms = ncvmgms + (n+1)*1
c      nmults = nmults + (n+1)*8 + 3
c      nadds = nadds + (n+1)*4 + 5
c
       do   i = 1,n
c       totmom0nu = f(i) * rhoy2(i)
        rhoy2nu(i) = f(i) * rhoy2(i)  +  sl(i) * dmom0l(i)
     &                                -  sl(i+1) * dmom0l(i+1)
       enddo
c
       ncvmgms = ncvmgms + n + 1
       nmults = nmults + n*11 + 11
       nadds = nadds + n*6 + 9
c
c
       return
       End
C                    Copyright 2002  Paul R. Woodward.
```

Comparison of Results for the Various Advection Schemes

In order to test and demonstrate the performance and accuracy of each advection scheme discussed above, we have embedded these in a gravitational two-stream instability program that was written for use in a computational methods course at the University of Minnesota. This application is written in Visual Basic 6.0 and runs on Microsoft Windows machines. The executable code, along with essential dynamic link libraries (DLLs) can be downloaded from the LCSE Web site at www.lcse.umn.edu/TwoStreamPPB. Each of the advection implementations discussed here has been compiled into a Fortran DLL to run inside this two-stream instability application. This allows visualization of the results of the advection calculations and also measurements, through this user interface, of performance on Windows platforms. In order to measure performance from within this larger application, an option is selected through the user interface that causes the time spent in the Fortran DLL implementing the advection scheme to be measured and converted to a number of Mflop/s. Because PCs do not allow for extremely accurate time measurements, the advection routine is called many times rather than just once, so that enough work is performed to yield a meaningful time interval measurement. Measurements of time intervals smaller than a few msec. are possible on PCs, but we have found that they are not reliable and repeatable. Consequently the advection routine is called a sufficient number of times to get above this measuring threshold. It is important to understand that measuring the code performance in this way enables the main memory traffic to be reduced, since the algorithm operates repeatedly on the same data. Although this may distort the performance measurement on a laptop computer, it gives a better indication of performance on PC workstations or Itanium machines, where main memory bandwidth is 4 or 10 times greater.

Our two-stream instability application simulates the advection of a fluid in a 2-D phase space. The horizontal dimension, x, is a spatial dimension, while the vertical dimension, u, is a velocity dimension. The location of a point in the phase space plane gives its spatial location,

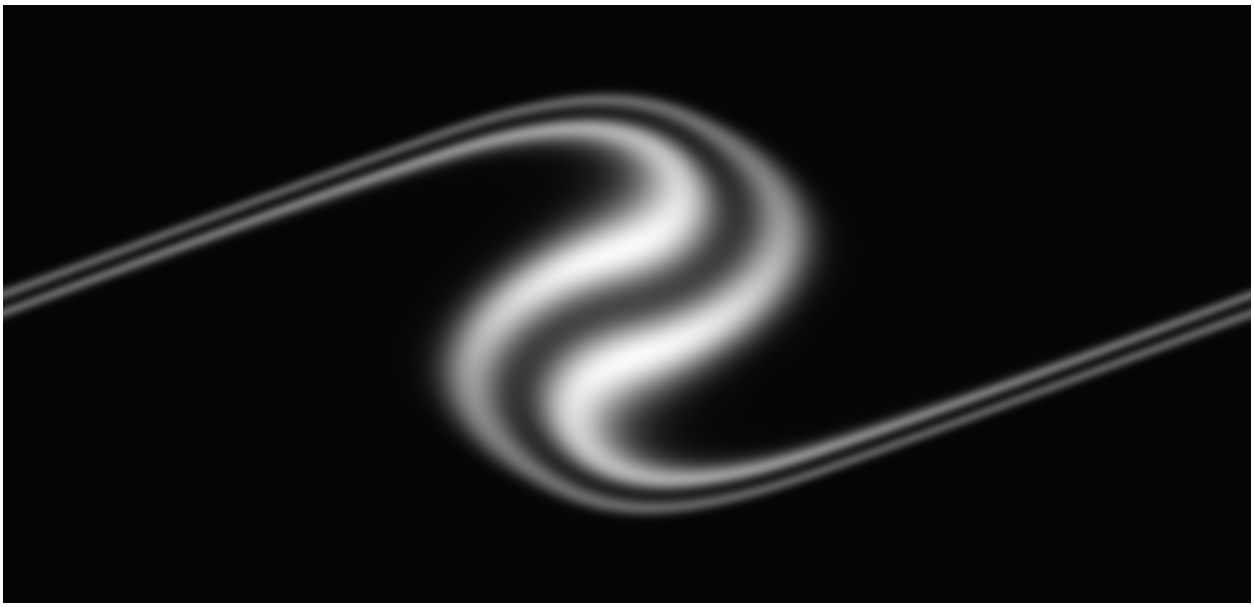


Figure 1a. The phase space distribution function at time 0.2 in the gravitational two-stream instability problem described in the text. This advected distribution was computed using the PPBshear advection scheme on a grid of 512x512 cells.

x, and its velocity in the x-direction, u. If we plot the phase-space density, f, as a function of u at any particular location x, then this plot gives us what physicists commonly call the velocity distribution function. The integral of the phase-space distribution function f over velocity is the density ρ in configuration space. This density is interpreted in this problem as a density of stars, each of which is a point mass far too small to collide with any other star during the time of our simulation. These stars interact through the gravitational force, which is computed by finding the Fourier transform of the density distribution ρ.

In our two-stream instability experiment, we assume periodic boundary conditions in the spatial dimension, x. We initially prescribe the spatial density ρ as a constant plus a cosine function that makes this density 50% larger in the center of our problem domain than it is at the edges. This large initial perturbation gets the problem moving interestingly right away. We also initially set up the phase space density f so that the velocity distribution function at each value of x consists of the superposition of two Gaussians, centered at velocities of $\pm u_0$ and with standard deviations of c_0. Each Gaussian thus corresponds to a group of stars moving either to

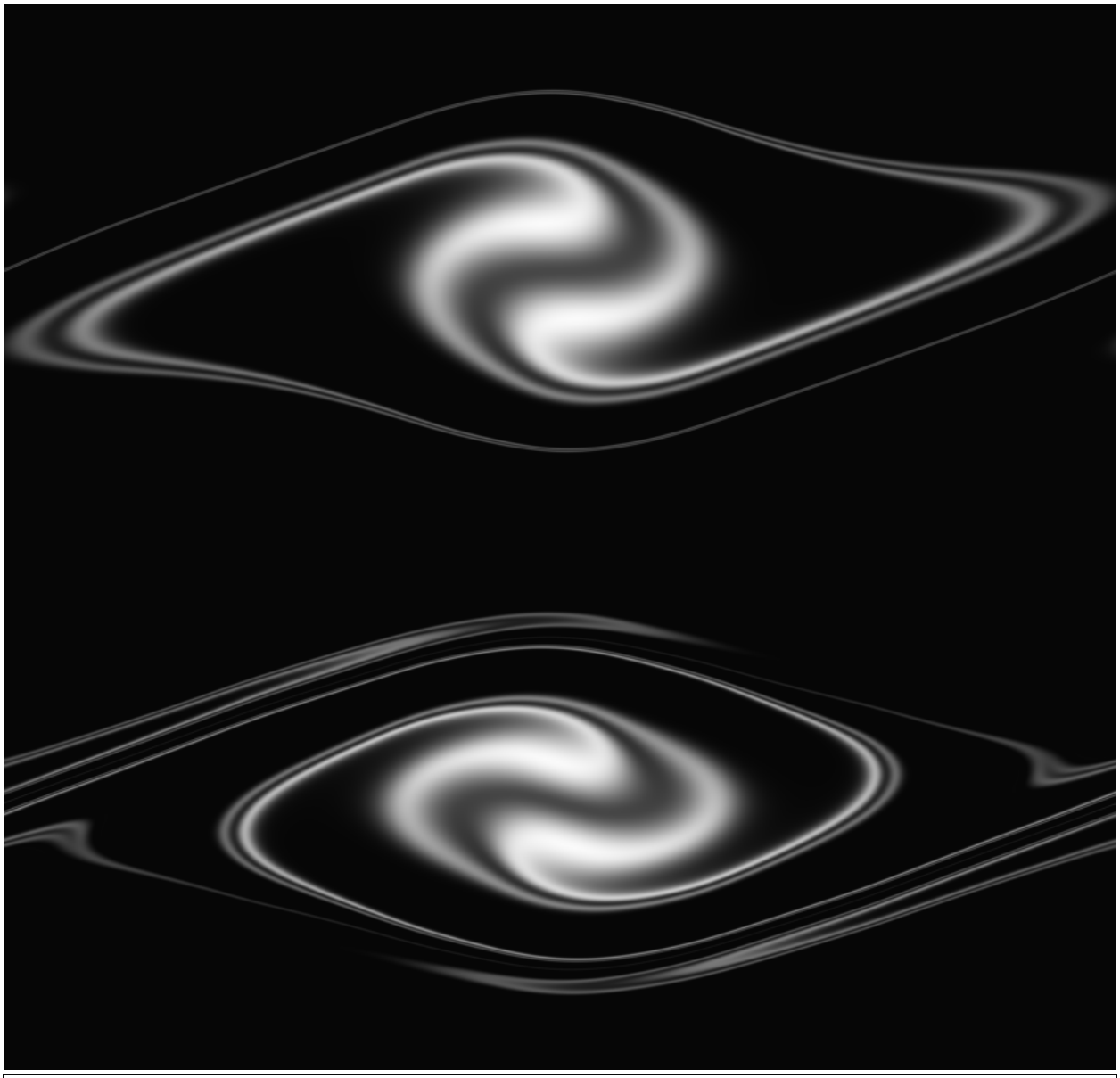


Figure 1b-c. The phase space distribution function at times 0.4 and 0.6 in the gravitational two-stream instability problem described in the text. This advected distribution was computed using the PPBshear advection scheme on a grid of 512x512 cells.

the left or to the right. For the cases we will present here, we choose $u_0 = 4$ *and* $c_0 = 2$, *so that these groups of stars are initially very well separated in velocity. Each group is 50% denser at the center of our problem domain, and therefore all stars are accelerated to one mild degree or another toward this point. This acceleration makes the phase space distribution move upward in*

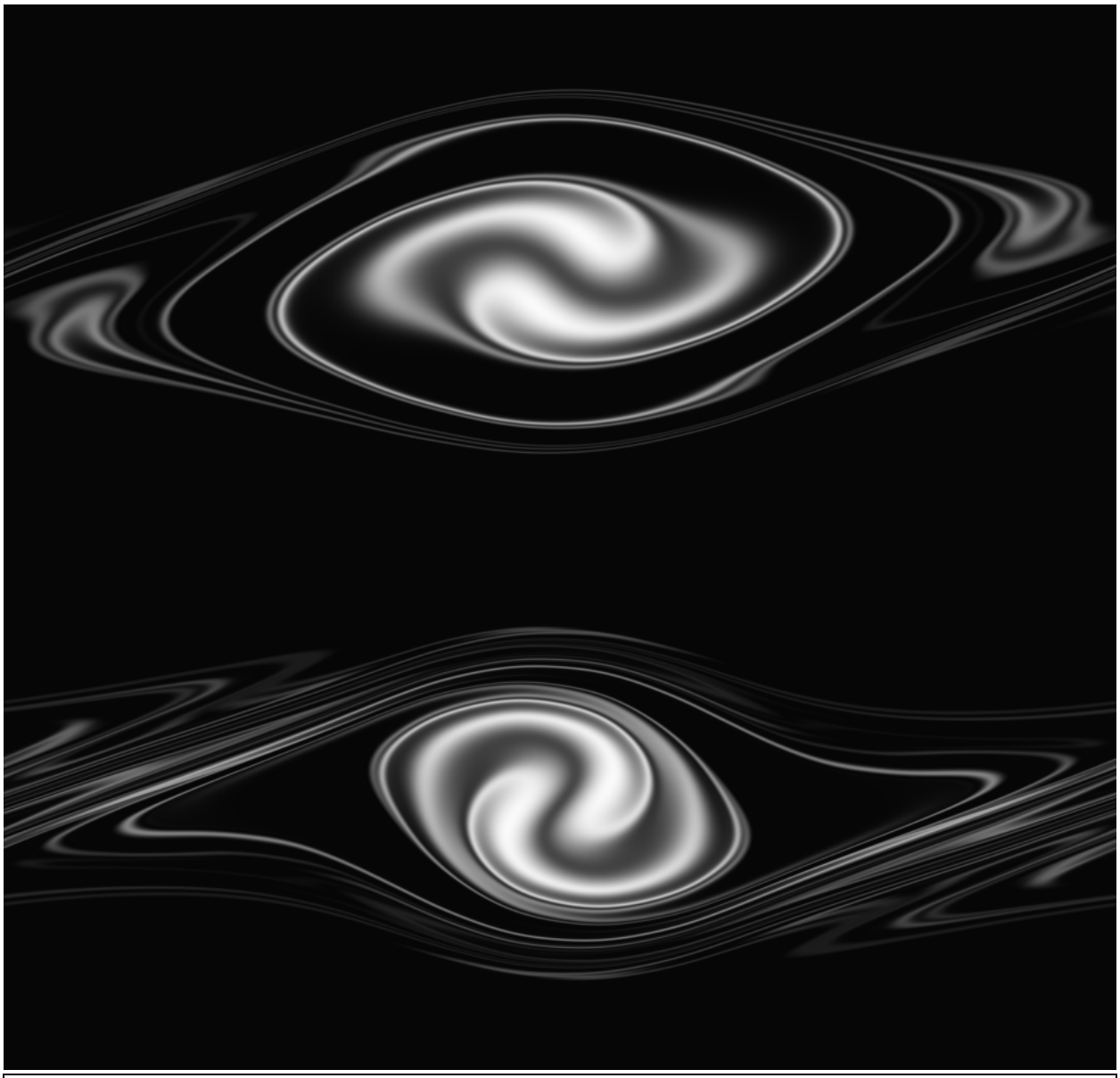


Figure 1d-e. The phase space distribution function at times 0.8 and 1.4 in the gravitational two-stream instability problem described in the text. This advected distribution was computed using the PPBshear advection scheme on a grid of 512x512 cells.

the left half-plane and downward in the right half-plane of our problem. By symmetry, accelerations vanish both at the center of the domain in x *and at its edges.*

The initial phase space distribution function is

$$f(x,u)=\left(\frac{1+0.5\cos(\pi x/L)}{2\sqrt{2\pi}\,c_0}\right)\left\{\exp\left[-\frac{(u-u_0)^2}{2{c_0}^2}\right]+\exp\left[-\frac{(u+u_0)^2}{2{c_0}^2}\right]\right\}$$

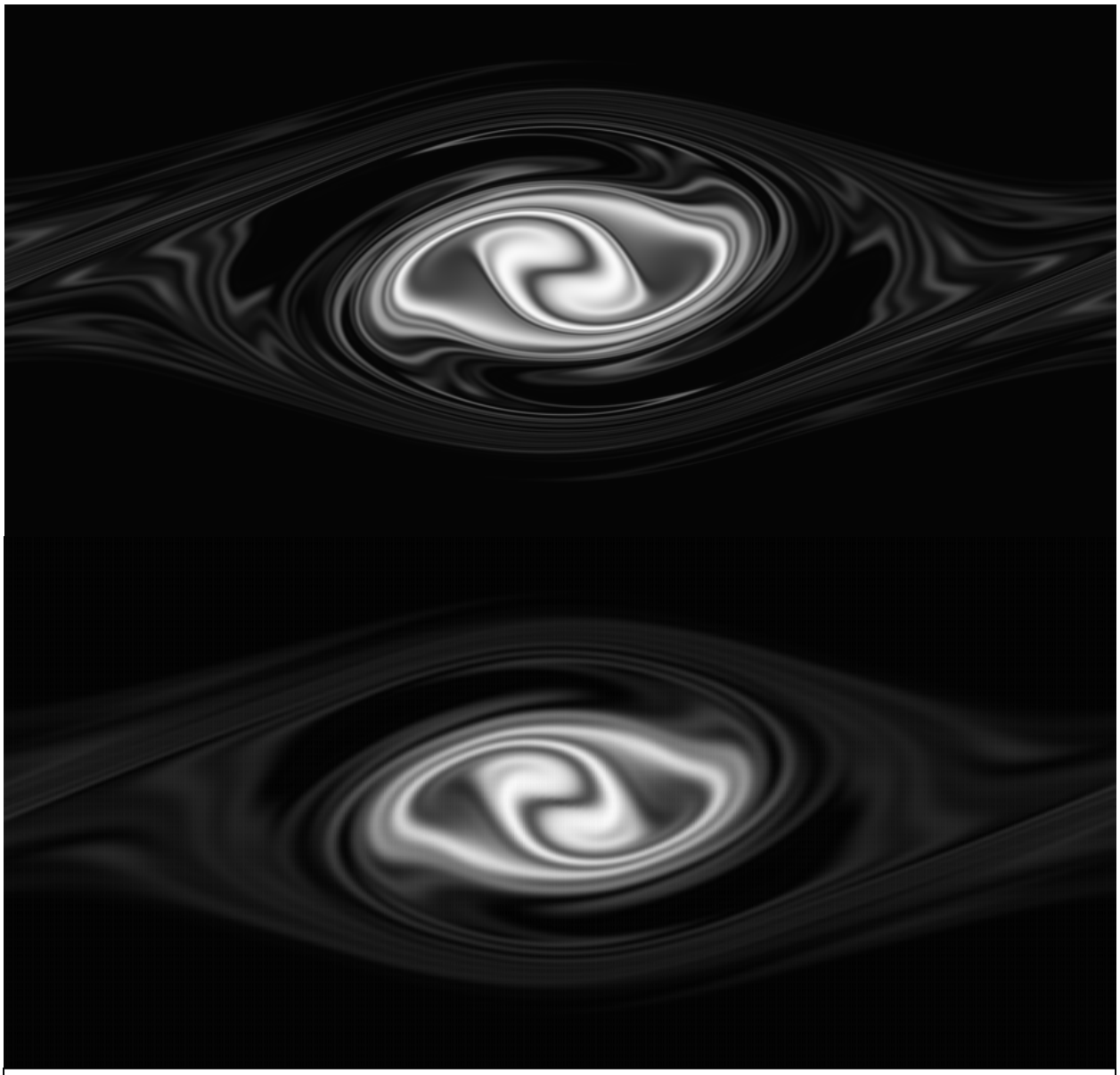


Figure 1f. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. In the upper panel, this advected distribution was computed using the PPBshear advection scheme on a grid of 512x512 cells, while in the lower panel a grid of only 128x128 cells was used. The extent to which the solution has converged on the finer grid is thus fairly evident.

The problem is further specified by setting L, *half the periodic length in the spatial dimension, to 6 and by setting the value of the gravitational constant to* 2π.

The time development of this gravitational two-stream instability, which is very similar to that first presented in Woodward 1982 and 1986, is shown below using the Visual Basic program

running the PPBshear scheme discussed earlier. These results, computed with the most accurate of the advection schemes presented here and on a very fine grid of 512×512 cells, may be regarded as showing the correct time development of the phase space density f for this problem. The first of the images from this run already shows that the two Gaussians have moved upward in the left half-plane and downward in the right half-plane. As this problem proceeds, a circulation about the center of the problem domain develops. This circular motion in phase space corresponds, of course, to a back-and-forth oscillatory motion in configuration space for the two groups of stars. The Gaussian strands, or streams, in phase space thus wind about each other ever more tightly as the evolution proceeds. Ultimately, at any given resolution we can find a time so late that the strands can no longer be distinguished. The system thus relaxes to a much hotter system of stars in which two separate velocity groups of stars can no longer be distinguished. Diffusion, which is inevitable on scales near that of our grid cells, must then ultimately bring this system into a steady, fully relaxed, equilibrium state.

This two-stream instability problem is well known in plasma physics, where the force law has the opposite sign. Although it is less familiar in astrophysics, it is in fact responsible for the process called dynamical friction that causes randomization of stellar velocities in galaxy mergers near the centers of rich clusters of galaxies. For this problem, the advection in phase space is particularly simple. Along each horizontal strip of grid cells, the fluid all moves to the left or right at the identical speed. Similarly, along each vertical column of grid cells, the fluid all moves upward or downward at the same speed as well. Nevertheless, because the total amount of phase space fluid is strictly conserved, the swirling flow in phase space stretches the original Gaussian streams out into ever thinner bands, which become progressively more and more tightly spun around the center of the problem domain, like a watch spring. As time advances, the advection problem therefore becomes progressively more difficult, and any advection scheme must ultimately break down no matter how fine a grid we choose at the outset. We can see this behavior

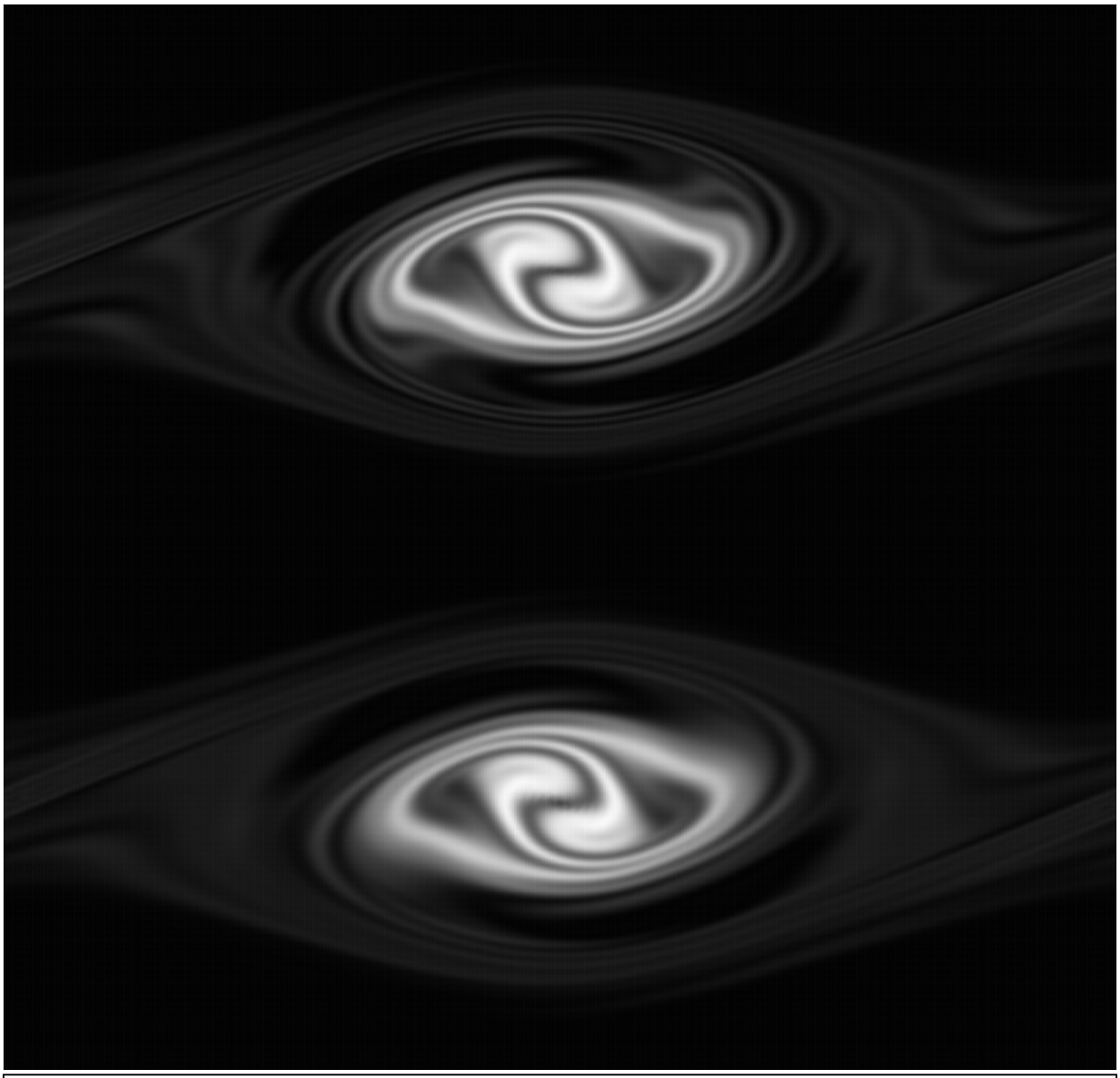


Figure 2a-b. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. In the upper panel, this advected distribution was computed using the PPBshear advection scheme on a grid of 128x128 cells, while in the lower panel the PPB6 advection scheme was used on this same grid. The lack of a treatment of subgrid-scale shear is evident in the central region of the lower panel, where saw-tooth structures arise inside the cells where the internal shear is strongest. Each cell is represented by plotted values for 16 equal subcells in this display. The saw-tooth structures would not be evident if only the cell averages were displayed.

in the snap shots shown here from the PPBshear run on the 512×512 grid. The developing flow is easily resolved on this grid in the first snap shot, but in the second, at problem time 0.4, the Gaussian streams have become extremely thin near the edges of the problem domain, so that their separate identities can no longer be maintained by the advection scheme. Nevertheless, on this fine

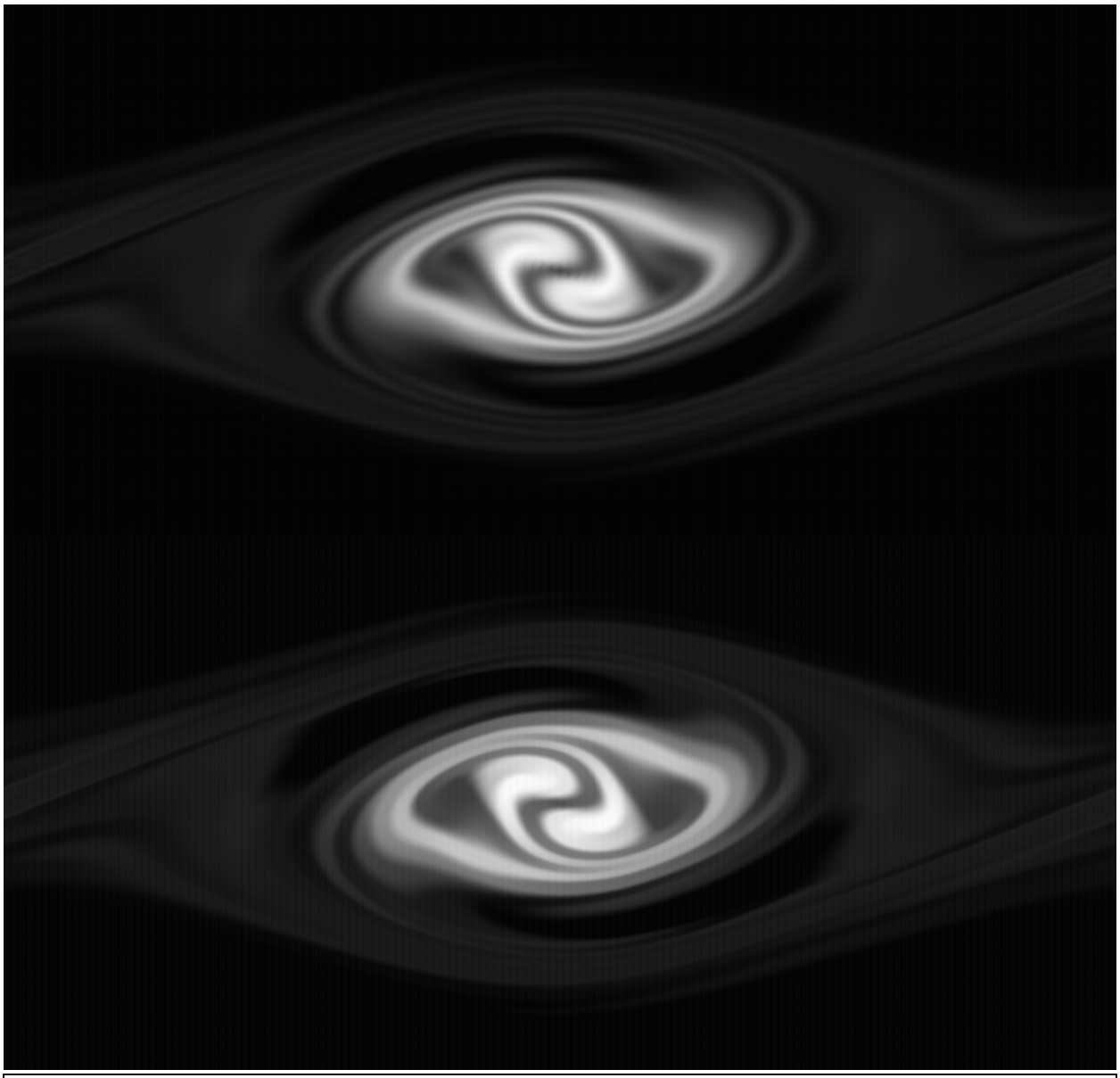


Figure 3a-b. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. In the upper panel, this advected distribution was computed using the PPB6 advection scheme on a grid of 128x128 cells, while in the lower panel the PPM advection scheme was used on a 384x384 grid. PPB6 requires 215 flops per cell update, while PPM requires only 109. However, PPB6 runs at 658 Mflop/s on a 1 GHz Pentium-III laptop, while PPM runs only at 492 Mflop/s on that machine. Thus, while the results of the two calculations are roughly equivalent, the PPM computation requires more than 18 times the computer time. In 3-D this cost advantage would increase by an additional factor of 0.6x3, so that PPB6 would be 33 times more efficient than PPM for this problem.

grid and with this very accurate advection scheme, these two streams are still well resolved near the center of the problem domain for the entire duration of this experiment.

The version of the PPB scheme used in this calculation enforces a positivity constraint.

When no monotonicity constraints of any kind are applied to this scheme, and when a Gaussian waveform becomes unresolvable on the mesh, a characteristic waveform is established by the action of this PPB advection scheme. This characteristic signature of an unresolved pulse has its centroid in the proper location, a feature that is guaranteed for linear advection by the conservation of the pulse center of mass by the scheme to machine round-off accuracy. The amplitude of the pulse is diminished, and the pulse develops wings that dip below zero to about 10% of the pulse height in magnitude. Oscillations about zero stretch out from the pulse centroid, but their amplitudes die off exponentially. As the pulse continues very, very slowly to spread out, this characteristic waveform persists, so that the negative values never much exceed 10% of the pulse height in magnitude no matter how far the pulse is propagated through the mesh by this advection scheme.

The behavior just described for the PPB scheme as it becomes unable to correctly describe the evolving phase space distribution function is quite benign. In contrast, we will see presently that other high order advection schemes break down in much less favorable ways when features reach the resolving limit of the grid. It is worth noting that in this problem, when individual streams become so thin that they can no longer be accurately tracked, they tend to be located near other such streams, and the streams tend to be merged together by the advection scheme. This behavior can be seen in the snap shots above at problem times 0.4 and 0.6. Although the two separate streams can no longer be distinguished along some of their lengths, a merged stream has been produced that is being quite accurately tracked as it moves through the grid. As described earlier, this PPB advection scheme conserves 9 moments of the phase space distribution function to machine round-off accuracy. As a result, the single stream created by merging the two separate, unresolved streams must preserve all those 9 moments of the more complex, unresolved structure. In our two-stream instability problem, this is true except insofar as errors in the phase space distribution function result in errors in the density ρ in configuration space, which in turn create

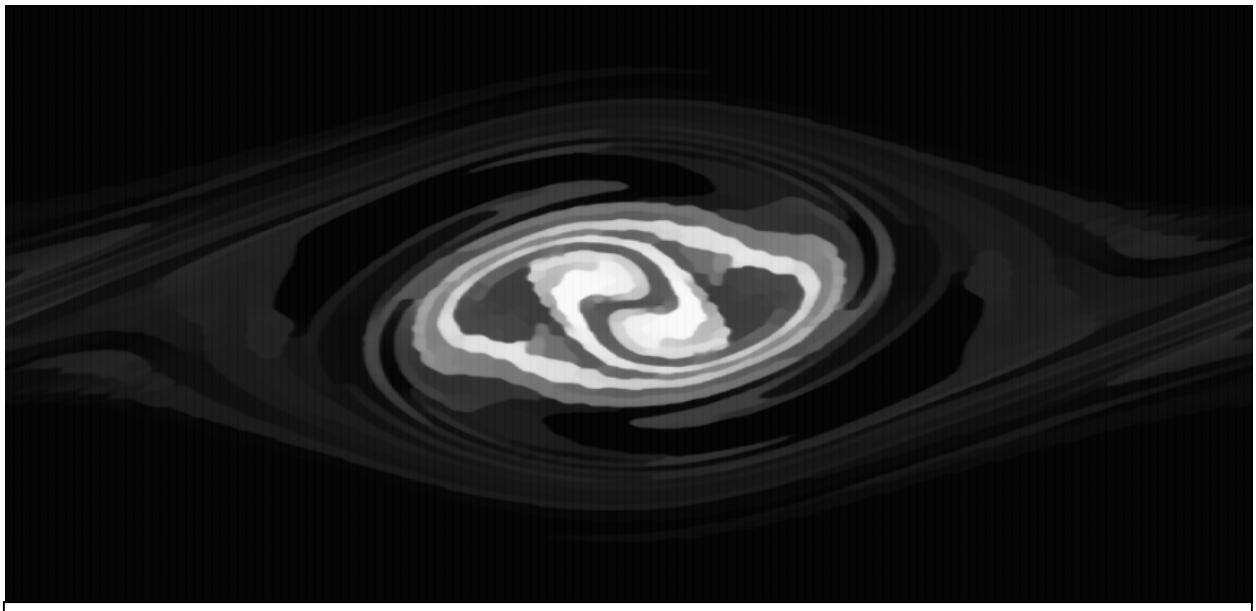


Figure 4. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. This advected distribution was computed using the PPMstpn advection scheme on a grid of 384x384 cells. The contact discontinuity steepening algorithm has introduced some stair-step features by enhancing sharp edges detected in the distribution when those edges were advected in directions near the principal grid diagonals. Although tweaking of the dimensionless constants that control the aggressiveness of this steepening is possible, the generation of such false signals is difficult to avoid without eliminating the benefits of the technique. Especially in situations where these false signals can stimulate fluid instabilities, such as Kelvin-Helmholtz instabilities, one is better off using one of the PPB methods to make sure that features that should stay sharp do so.

errors in the gravitational accelerations, which are the advection speeds in the vertical direction in our phase space.

We can consider the PPBshear results of the two-stream instability problem on the 512×512 grid that are presented above to be representations of the exact solution of this sample problem, since they are demonstrably far more accurate than any we will present below. At the same time, it is clear from the results shown here that there is a convergence of the results of this and the other advection schemes as the grid is progressively refined. Evidence of that assertion for the PPBshear scheme is given in the final panel, Figure 1f, where results from running the same problem on a much coarser grid of only 128×128 cells are shown below those obtained on the 512×512 grid. The flow is displayed at a problem time of 3.0, which we use for comparing results from the four different advection schemes discussed in this paper.

This concludes our introduction of the physical problem that we will use to test and demonstrate our advection schemes in terms of both accuracy and performance. The two-stream instability program, which is available for download at www.lcse.umn.edu/two-stream-test, measures algorithm performance by counting flops executed and by measuring the time required to perform grid updates via many repeated calls to a grid update program compiled into a Fortran DLL. Results from these measurements on a Windows 2000 laptop computer with a 1 GHz Intel mobile Pentium-III CPU with a 128 KB cache memory are given in the table below. A second table is included with results from my twelve-year-old's new desktop computer. This machine is much faster, even though it cost 5 times less. It has a 2.53 GHz Intel Pentium-4 CPU (not a Xeon model, since he is only 12). The performance numbers for this machine were obtained with Fortran DLLs compiled for the earlier Pentium-3. The flop counts refer to a single grid cell update for 2-D advection. All computations were performed with 32-bit arithmetic, which vectorizes when expressed appropriately in Fortran under the Intel Fortran 5.0 compiler (although during its development this program exposed a number of interesting bugs in that compiler involving subroutine inlining (don't use it) and the treatment of dynamic arrays beginning with negative index values (don't use them)). Mflop/s performance reflects the repeated calls to the grid update functions, which therefore do not need to do as much main memory access. This measurement strategy gives a better idea of algorithm performance on machines with larger cache memories, where none of these schemes require much main memory traffic. Mflop/s are largest for algorithms that involve very little logic, such as the PPB schemes without monotonicity or positivity constraints. PPB schemes involving positivity constraints, which are all that is appropriate for this two-stream instability problem, are labeled in the table with the suffix "pos." The PPM scheme with its contact discontinuity steepening algorithm activated is labeled "PPMstpn." The scheme labeled "RKV1D" is a fifth-order in space, third-order in time Runge-Kutta scheme that is used fairly widely in the meteorological community, but here it

is implemented in a series of 1-D passes, which greatly enhances its accuracy, speed, and stability.

	Mflop/s	Flops/Cell	Adds/Cell	Mults/Cell	Cvmgms/Cell	Recips/Cell
PPM	712	109	58	46	27	2
PPMstpn	520	149	83	61	35	2
RKV1D	669	87	50	37	19	0
PPB6pos	658	215	101	108	39	2
PPBpos	815	388	190	192	55	2
PPB6ShearPos	619	526	258	256	77	4
PPBShearPos	661	928	459	457	110	4

The above table gives data for advection scheme performance on a 1 GHz Pentium-III laptop with 128 KB cache.

	Relative Mflop/s	Relative Flops/Cell	Relative Cvmgms/Cell	Mults/Cell	MCells/sec	Relative Cells/sec
PPM	1.08	0.12	0.25	46	6.53	9.17
PPMstpn	0.79	0.16	0.32	61	3.49	4.90
RKV1D	1.01	0.094	0.17	37	7.69	10.80
PPB6pos	0.995	0.23	0.35	108	3.06	4.30
PPBpos	1.23	0.42	0.50	192	2.10	2.95
PPB6ShearPos	0.94	0.57	0.70	256	1.18	1.66
PPBShearPos	1.00	1.00	1.00	457	0.712	1.00

We see that the PPB6pos scheme, given that it is as accurate as the PPM scheme using 3 times as many cells in each dimension, is, in 3D, 38 times more computationally efficient than the PPM scheme.

	Mflop/s	Flops/Cell	Adds/Cell	Mults/Cell	Cvmgms/Cell	Recips/Cell
PPM	1521	109	58	46	27	2
PPMstpn	1270	149	83	61	35	2
RKV1D	768	87	50	37	19	0
PPB6pos	1428	215	101	108	39	2
PPBpos	1604	388	190	192	55	2
PPB6ShearPos	1165	526	258	256	77	4
PPBShearPos	998	928	459	457	110	4
PPB6	1848	140	69	71	0	0
PPB	1929	255	121	134	0	0
PPM hydro64	775	984	481	442	269	16
PPM hydro32	961	964	471	434	265	16
PPM Slow Flow hydro32	1317	699	303	256	144	20

The above table gives data for advection scheme performance on a 2.53 GHz Pentium-IV desktop PC purchased at CompUSA for just under $1000. The schemes were compiled for the Pentium-III with the Intel 5.0 Fortran compiler. All, except for "PPM hydro64," use 32-bit floating point arithmetic, which delivers all the accuracy that could possibly be required for an advection or a hydrodynamics computation with these schemes.

The data given in the second table above performs a comparison of the various advection schemes in terms of 2-D grid cell update speed, measured in Mcells/sec, in terms of Mflop/s, and in terms of the number of flops required to perform a 2-D grid cell update. Unfortunately, accuracy comparisons are subjective, but we have seen that the PPB6pos scheme is very nearly as accurate as the PPM scheme using 3 times as many grid cells in each dimension. The RKV1D scheme is a little bit more accurate than PPM, but it has a stencil of 19 grid cells

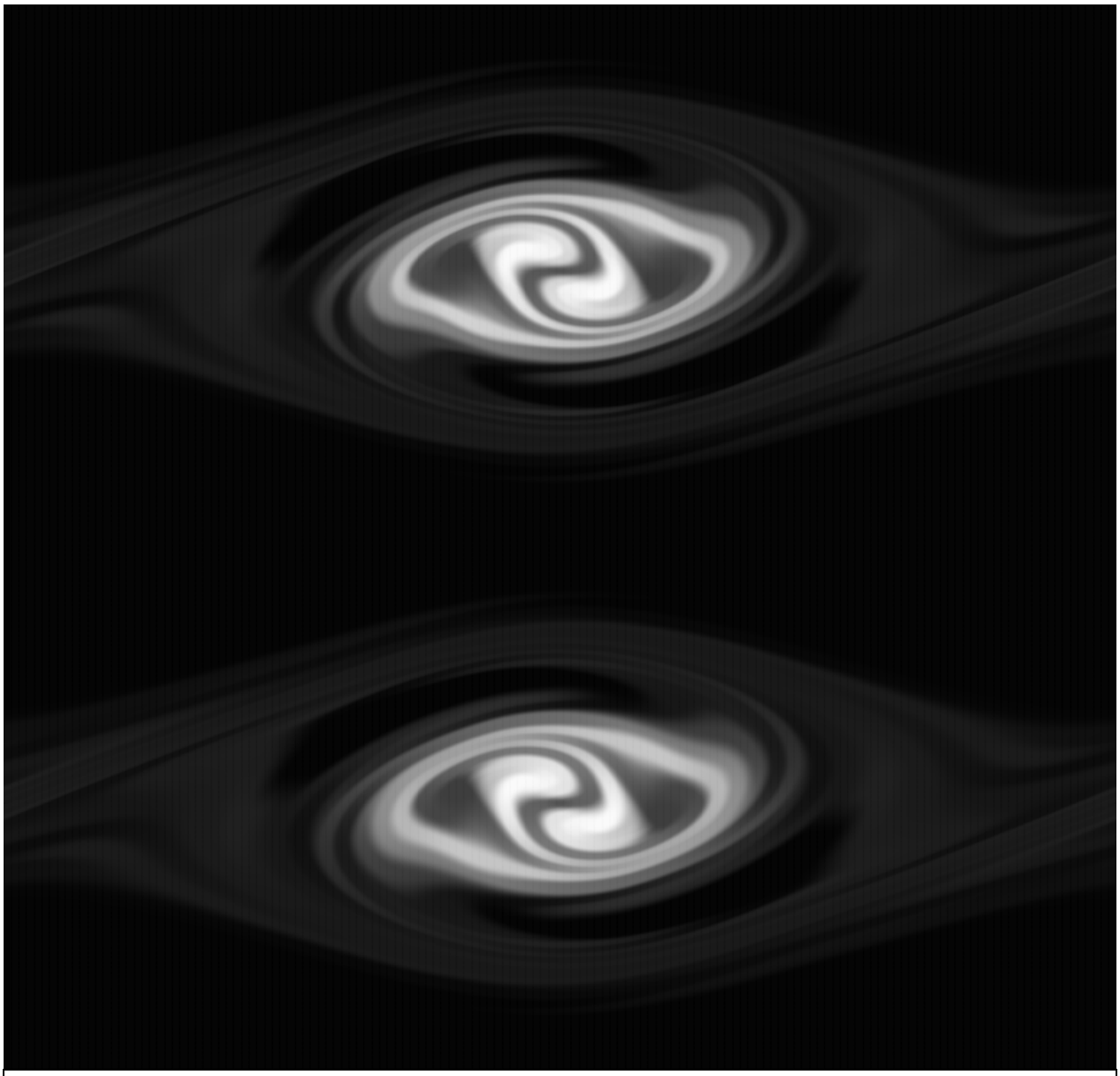


Figure 5a-b. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. In the upper panel, this advected distribution was computed using the RKV1D advection scheme on a grid of 384x384 cells, while in the lower panel the PPM advection scheme was used on a 384x384 grid. PPM is slightly less accurate and requires a bit more computation, but it has a very much smaller difference stencil, requiring data from only a third as many grid cells on each side of a given cell (namely 3 instead of 9) in order to update that cell.

in each 1-D sweep, compared to PPM's stencil of 7 grid cells. This large stencil tends to introduce a very high cost in a parallel implementation on a distributed memory computing platform, and hence the RKV1D scheme is not discussed here. For this problem, the contact discontinuity steepener in the PPMstpn scheme produces some stair-stepping of artificially

steepened gradients, a flaw of this sort of scheme for transport at nearly 45° to the mesh that is hard to remove without eliminating all benefits of the discontinuity steepening.

It should also be noted that, although we have not shown the result here, adding to the PPB6pos scheme the treatment of grid cell shear that was discussed earlier eliminates the slight effects of grid cell shear that are apparent in the figure shown earlier. However, those results are not shown here, since the improvement does not appear to be worth its cost. The most common application of an advection scheme is for it to be embedded in a larger computation. This larger computation will generally produce the velocity field for the advection. Most hydrodynamic methods will involve unavoidable numerical viscosity effects which limit the extent to which subgrid-scale shear can play a meaningful role in any advection computation using the computed velocity field. For this reason, we have not dwelt in detail here on the algorithm mentioned earlier for treating subgrid-scale shear. There is not space to show all the various results of all these different advection schemes here, but the reader is invited to download the two-stream instability program from the LCSE Web site and to run any experiment with it that he or she desires. The test program is written in Visual Basic 6.0, calling Fortran DDLs embodying the various schemes, and it should run on just about any Windows machine. Experiments take only a few minutes each, although the run presented here to show the converged behavior on this problem ran overnight on my laptop machine. The user can play around with different physical parameters for this two-stream instability problem and discover a wealth of interesting behavior.

Results for the PPB6 Advection Scheme Embedded in a PPM Gas Dynamics Code

The moment-conserving advection schemes discussed here have been developed with two principal applications in mind. First is the advection of a distribution function of particles in a 4- or 6-D phase space. In such an application, the advection has very special properties. Along each grid strip in any of the 6 dimensions, all grid cells move at precisely the same velocity in precisely the same direction. This fact can be exploited to collapse some of the computation involved in the PPB schemes and thus to make them more cost effective. Also, for such applications a simple constraint the the particle distribution function be positive is sufficient, so that the PPB schemes can be made more cost effective still. Given the high dimensionality of these problems, these simplifying features are very welcome. Also, the high resolving power of the PPB schemes is welcome, since without this property the number of grid points required to span the appropriate regions of a 6-dimensional phase space would be prohibitive. Given the accuracy of the PPB schemes and the expected clumping of the particle distribution function in limited regions of the velocity dimensions for, say, simulations of normal spiral galaxies, it should be possible to obtain an adequate description with only 20 to 30 grid cells per velocity dimension. Since the PPB6 scheme with positivity constraints executes on a 2.5 GHz Intel Pentium-4 CPU at over 1.4 Gflop/s, it should be practical on modern supercomputers to use up to 10 billion grid cells. This would allow 20 cells in each of 3 velocity dimensions and 100 cells in each of 3 spatial dimensions, with 2 billion cells left over. The computations of the two-stream instability problem we have just seen on 128×128 grids give some idea of the resolution we would be able to achieve in such a calculation.

The second targeted application for the PPB schemes is advection of especially important variables within fluid dynamics simulations. Such a special variable could be the entropy, or, as the meteorologists call it, the potential temperature. Also, wind borne pollutants or chemically reacting trace concentrations could be such special variables. A particularly interesting potential

application is to use the PPB or PPB6 scheme to advect the fractional volumes of different gases in multifluid gas dynamics problems. These fractional volume variables play a special role deserving of highly accurate treatment because they tend to delineate multifluid interfaces where a variety of fluid instabilities develop that can profoundly affect the global development of a fluid flow. Numerical diffusion at these interfaces is especially to be avoided, since it is not only unphysical but it also can profoundly alter the simulated flow behavior.

Like the Boltzmann (or Vlasov) equation problems of stellar dynamics discussed above, advection of fluid fractions in gas dynamics problems involves certain simplifications that tend to counter the potential cost of using the PPB schemes. In most such problems we know that in the vast bulk of the grid cells in the simulation the fluid fraction variables will have either the value 0 or 1. In such regions there is really no work to do. All the computational labor in such advection problems can be concentrated on the multifluid interface and mixing regions, which are likely to occupy only a small fraction of the problem domain. This special feature of multifluid advection can reduce not only the computational requirements but also the memory requirements. In any cell where the fluid fraction is either 0 or 1 (the vast bulk of the cells), there is no point in storing any high-order moments of the distribution.

The key concern for using an advection scheme such as PPB or PPB6 for multifluid problems is numerical diffusion of the multifluid interfaces. Multifluid gas dynamics has for many decades been pursued either by use of Lagrangian coordinates, in one form or another, so that numerical diffusion of multifluid interfaces vanishes, or it has been pursued using interface tracking techniques of one type or another. We will not discuss Lagrangian techniques, since they make little sense in 3-D unless the flow is of such a character that it is basically 1-D in any event. Interface tracking techniques attempt to introduce subgrid-scale information about the configuration of the fluids in mixed cells. The demand that the reconstructed multifluid interface be infinitely thin in such approaches introduces the potential for a variety of numerically generated

glitches, since at the scale of a grid cell the representation of the multifluid interface, or at least its numerical treatment, simply cannot be smooth.

We adopt here the philosophical position that a grid cell represents a sort of quantum limit for a numerical simulation, and it therefore does not make sense to construct elaborate and detailed representations of a grid cell's interior. One might object that by, potentially, prescribing up to 27 moments of the subgrid-scale structure we are violating our own philosophy. However, these 10 or 27 moments serve only to determine a smooth, internal structure, with no sudden internal jumps and, in the 10-moment case, with the charater of a simple parabola along any particular line through the grid cell. Our large quantity of subgrid-scale information serves only to make the determination of our smooth internal representation of the function highly accurate. We do not use this information to insert into the cell interior sudden jumps or any other truly subgrid-scale feature. Using PPB or PPB6 to describe a multifluid interface through the advection of the fluid fraction will therefore result in the smooth smearing out of the multifluid interface so that it becomes roughly one or two cells wide. This is appropriate, because our view of the grid cell as a quantum limit argues that to make the interface thinner would be meaningless. We use the PPB scheme to advect this interface with great accuracy, so that it does not rapidly grow thicker. Hence a critical test of the value of this approach is to see if such interfaces, where a variable advected by the PPB6 scheme (the least accurate of the PPB family we are discussing) jumps over one or two grid cells from the value 0 to 1, remain this sharp over the course of a long and complex fluid flow simulation.

I have decided to design such a test of PPB6 advection within the classic wind tunnel simulation problem that was originally introduced by Emery in 1967 and which I discussed at great length in my 1980 and 1984 articles with Phil Colella. In this 2-D gas dynamics problem, air (or any other gamma-law gas) enters a 2-D duct, or wind tunnel, supersonically at the left, travels down the duct, striking and flowing around a forward facing step on its way, and flows out of the duct supersonically at the right. The supersonic flow-in boundary condition at the

left is trivially implemented, the supersonic outflow at the right is also easily treated, and the walls of the duct are modeled as impenetrable, thermally insulating, perfectly reflecting surfaces. In the 1984 discussion with Phil Colella, we pointed out that the corner of the step in this wind tunnel is a singular point, where the gas is invited to turn a corner instantly – a feat that, although possible in theory, is impossible in practice (and, for that matter, also impossible in nature). In order to make 7 different numerical schemes all converge to a single result on grids that today seem ludicrously coarse, we decided to force the flow solution at the corner of the step to behave "properly." In the test program implemented here, I do not do anything special at the corner of the step, leaving the numerical scheme to generate entropy there through the action of numerical viscosity. This creates a boundary layer along the top of the step, and the presence of this boundary layer, of numerical origin, profoundly affects the macroscopic flow behavior. Nevertheless, the simulated flow converges as the grid is refined, and it is interesting to speculate whether or not this converged solution represents a limit of Navier-Stokes viscous flows as the viscosity tends toward zero. However, these details are irrelevant to our desire and ability to test the capability of the PPB6 advection scheme to keep multifluid interfaces sharp.

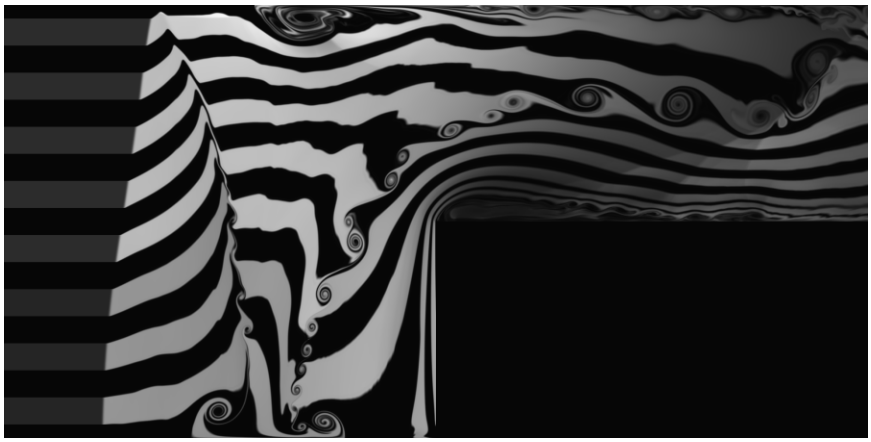
The PPB6 advection scheme has been built into the PPM gas dynamics code running the wind tunnel test problem. A graphical user interface offering real time interaction and display has been built as a Visual Basic 6.0 program which calls the PPM and PPB6 codes through Fortran DLLs compiled under the Intel Fortran 5.0 compiler. This software, and a detailed user guide to the code, is available for download from the LCSE Web site at www.lcse.umn.edu/windtunnel. Although this user interface appears to offer alternative gas dynamics schemes, only the default selection, 32-bit vectorized PPM with PPB6, will operate properly (my apologies). This advection problem, which has been discussed at length for one special case in Woodward and Colella 1984 and which is discussed in a rather different fashion for an intended audience of University of Minnesota freshmen in the Wind Tunnel User's Guide, can be substantially modified through the provided user interface. The resulting flow

behaviors are fascinating, but irrelevant to our concerns here. To make the otherwise boring initial development of this wind tunnel flow more visually exciting, I added a shear layer in the duct, which can be repositioned and modified through the "Advanced Setup" button on the user interface. I also added 8 user configurable streams of smoke which enter the duct through the left-hand boundary. These smoke streams are passively advected with the flow, and they are advected by the PPB6 scheme described at great length earlier.

I have chosen a wind tunnel problem that exercises the PPB6 advection scheme especially well, including testing its behavior tracking interfaces caught up in developing Kelvin-Helmholtz instabilities. The smoke streams near the outset of this problem are shown along with the user interface controls in the figure at the top of this page. Air enters the wind tunnel at the left at Mach 4. There are 8 streams of smoke, inside each of which the fractional volume variable that we advect with PPB6 has the value 1. We generate a smoke visualization by rendering an image of the product of this smoke fractional volume and the density of the air. Therefore, for

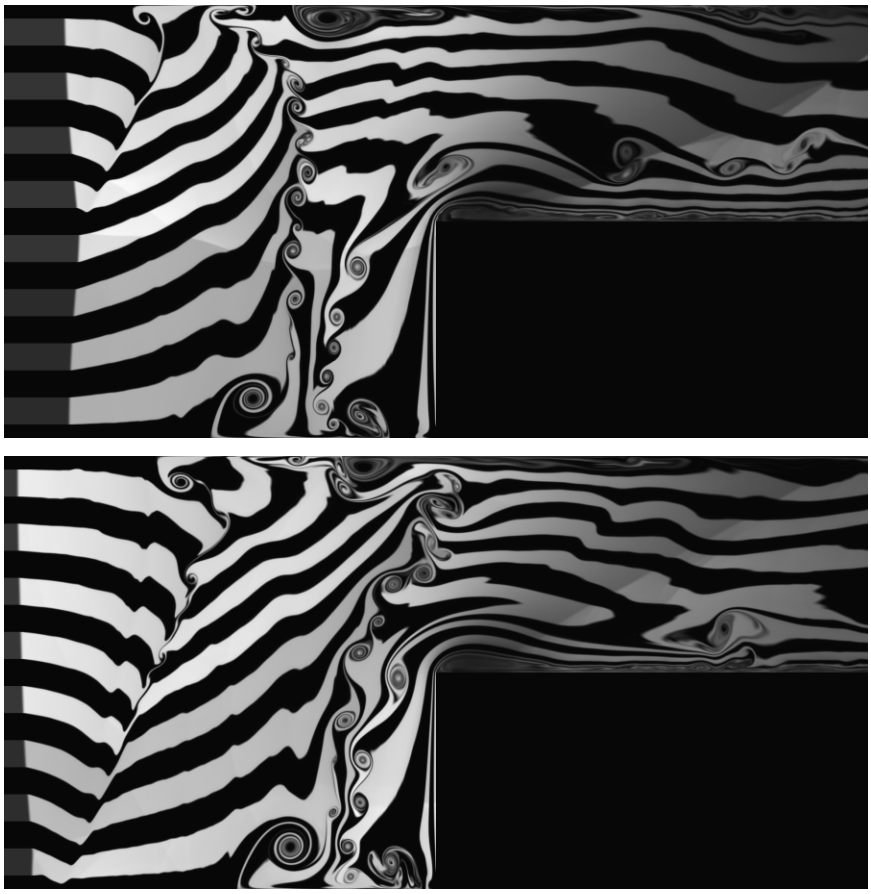
example, the developing bow shock in front of the step is clearly seen by the sudden brightening of the smoke streams that pass through this front. To add visual interest, the initial density inside the wind tunnel was given a smooth variation, assuming its smallest value, 1, at the lower left corner, and its largest value, more than 2, at the upper right corner. The air entering the wind tunnel, however, has a uniform density of 1 in the bottom 3/8 of the duct width and a uniform density of 1.25 in the upper 5/8. The Mach number of this entering air is everywhere 4, so that the denser, upper air is traveling less rapidly down the duct.

The initial pressure and density mismatches along the wind tunnel entrance cause, along its entire length, shocks to run into the entering air and rarefactions to run into the air originally in the duct, leaving contact discontinuities at the original location of the entrance in the inflowing air. These three wave fronts can be distinguished from the shade variations in the smoke streams in the



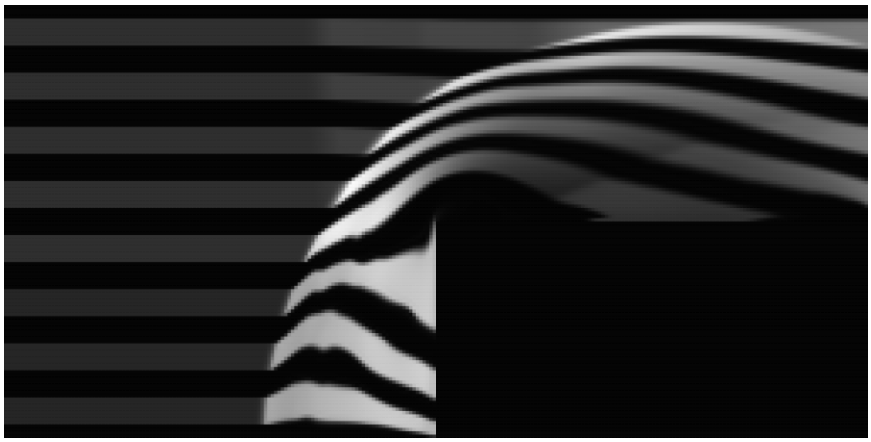
first figure of this sequence, which shows the flow at time 0.12. Because of the slower flow down the duct in its upper 5/8, all three of these wave fronts jog to the left 3/8 of the way up from the bottom toward the top of the duct. The development of this wind tunnel flow, as visualized by the 8 smoke streams advected by the PPB6 method, is shown in the image sequence on this and the next few pages. These images, which, except for the first one at time 0.12, show the flow at time intervals of 0.3 (beginning at time 0.3), come from a simulation on a grid of 1024×512 cells, for which each of the entering smoke streams is 32 cells wide.

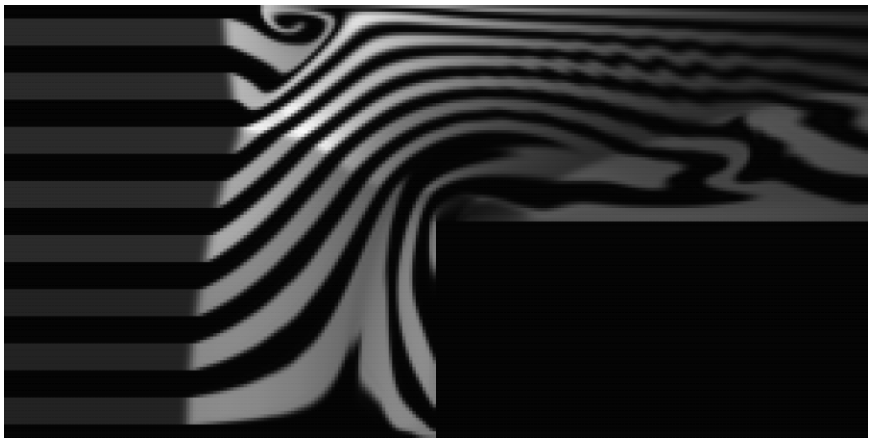
The images from the 1024×512 simulation give a sense of the "correct" flow behavior; they are not intended to show off the accuracy of the PPB6 advection. Nevertheless, one can marvel at the entrainment of these smoke streams in the many vortices, both small and large, that



develop within this flow. Also, the definition of the very thin streams of smoke that result from the strong shear associated with the double Mach reflection of the bow shock at the upper wall of the duct is exceptional. The tracing of the strands of smoke entrained in the vortex at the tip of the jet that squirts forward along the upper wall also poses a strong challenge to any advection scheme. What is absolutely absent from these images is any noticeable diffusion of the edges of the smoke streams. To see that such diffusion is truly absent, we must examine runs of this same complex flow problem on much coarser grids, where any errors will appear much more glaringly to the eye.

The images in the sequence just discussed are generated by representing each grid cell with 4 subcell values. These values are interpolated for the smoke fractional volume using the moment data, carefully constrained in 2-D to lie in the range from 0 to 1. The density values in the 4 subcells come from PPM cell average data that has been interpolated with the PPM algorithm in both the x- and y-dimensions. The behavior of the subcell density distribution is not constrained along the cell diagonals, so it is possible to obtain small negative density values from this interpolation process. These were not removed, since they appear at roughly the 1% level, and they do not marr the visualizations. However, if we are to assess the resolving power of the PPB6

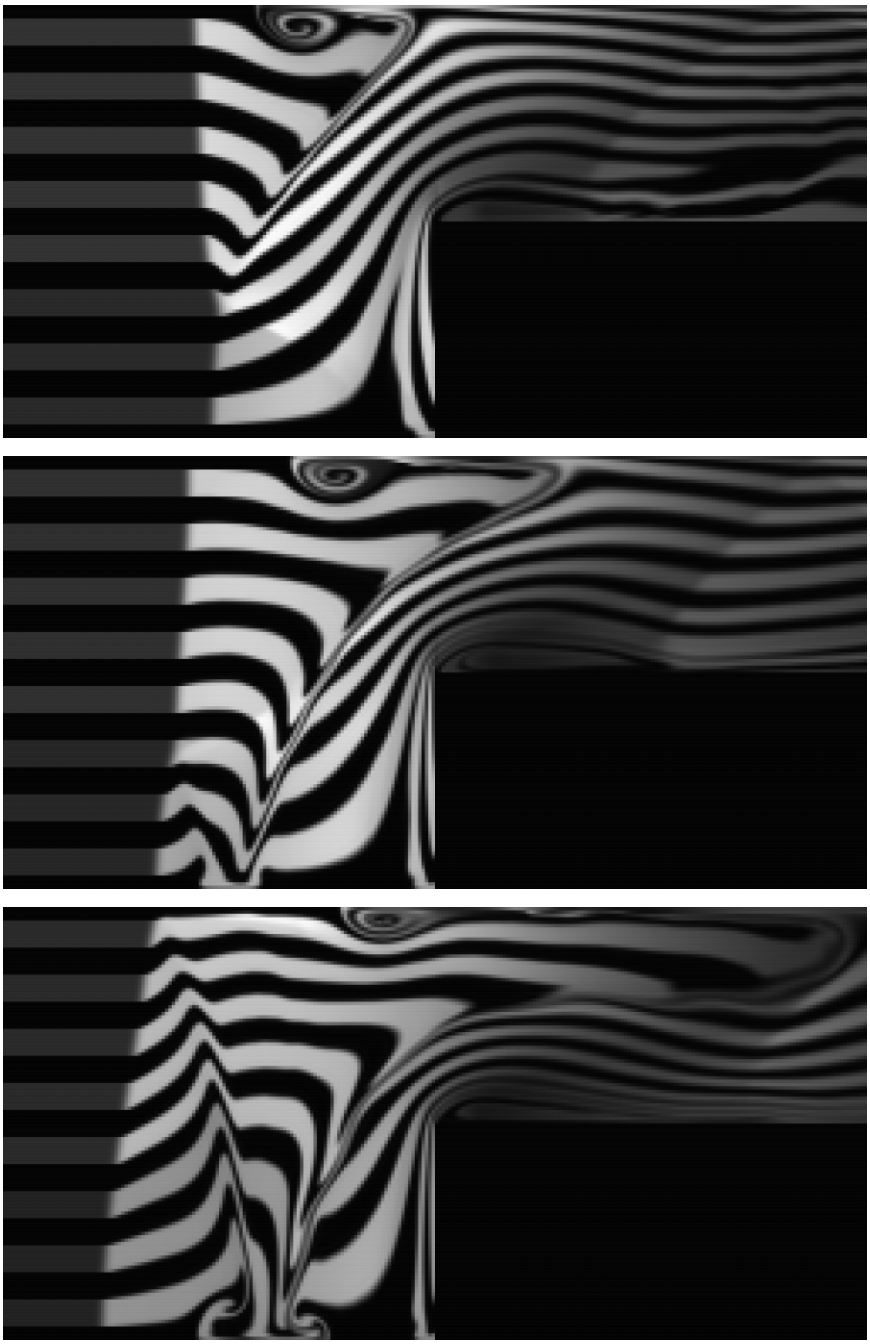




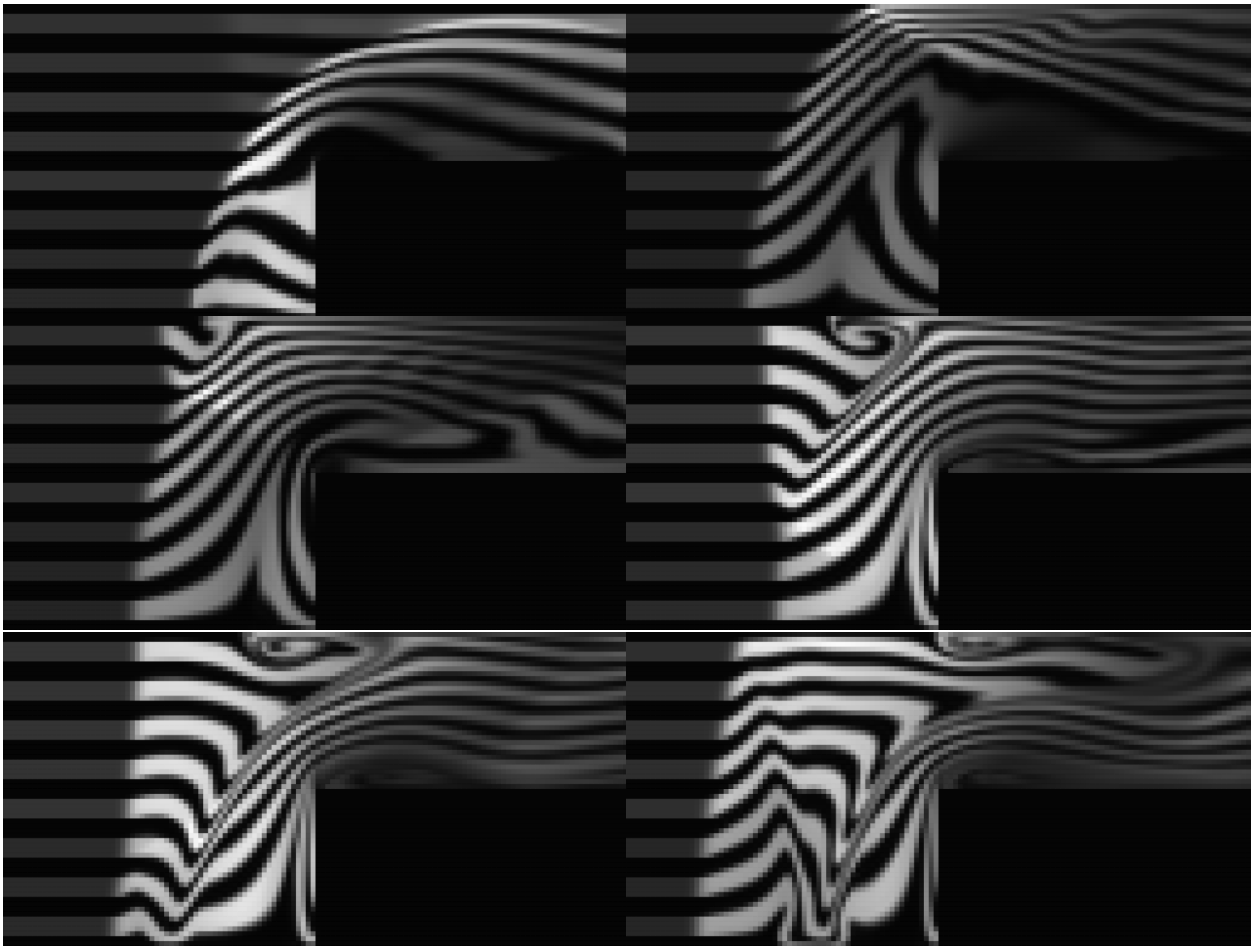
advection in detail, it is best to change the wind tunnel simulation program so that only cell averaged values are displayed. This will also help us to see the individual grid cells in the resulting displays, and thus to assess more precisely the behavior of the advection scheme.

To see clearly what the PPB6 advection scheme is doing at the scale of individual grid cells, we need to reduce the resolution of the wind tunnel simulation. On this page and the next are a series of smoke visualizations in which only grid cell averages of the product of the density and the fractional volume of smoke are shown. The grid used in this simulation is 256×128 cells, and it is possible by careful inspection of these images to pick out the individual cells. At this grid resolution, each smoke stream enters the duct at the left precisely 8 cells wide. Although a simulation on a 128×64 grid would make the individual

cell values clearer (see the next set of images), its smoke streams, at only 4 cells wide, would pose an unreasonable challenge to the advection scheme.



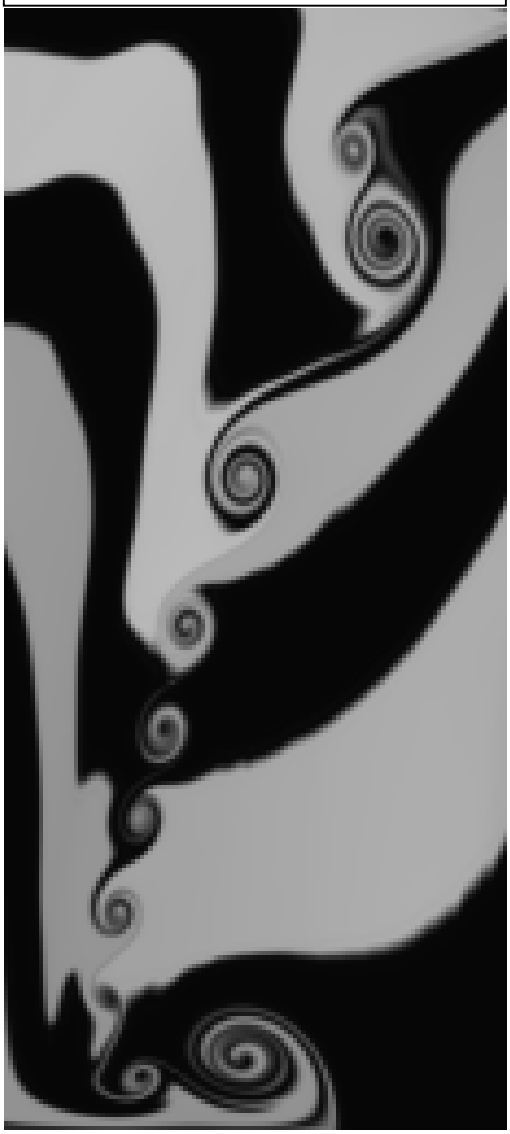
The third image in this sequence clearly shows that the smoke stream nearest to the upper wall of the duct is sheared and compressed into a stream less than 2 cells wide which is transported in a swirling flow without loss of definition. This behavior of the PPB6 advection scheme is remarkable. One might argue that the path traversed by this smoke stream is rather short, so that diffusion does not have so long to act in spreading the stream out, but the various smoke streams do not appear to become much less distinct as they travel down the length of the wind tunnel, they only become dimmer, since the gas density decreases in this region. In the fifth image of this sequence, the smoke streams are sheared into thin strips that are only one cell



wide and that are separated by only one cell. It is truly remarkable that these thin strips remain distinct until they are crowded together even more closely than this.

In this coarser flow simulation a minor flaw, caused by PPM and not by PPB6, is very noticeable in the second and third images of the sequence. This is a brief stretch of several of the smoke streams that develops a zig-zag shape after passing through the bow shock. An examination at this point in the problem of the distribution of the vertical component of velocity reveals that this has developed a brief region of oscillatory behavior just behind the bow shock. This is a numerical instability first identified in my article with Colella in 1984. It is caused by a slowly moving strong shock front that is nearly aligned with the mesh. In 1984 we called this instability a Cray instability, because at the time one needed to own a Cray supercomputer in order to perform

Close-up views from time 1.8 of the run on the 1024×512 grid. Each original pixel is slightly larger than a grid cell.



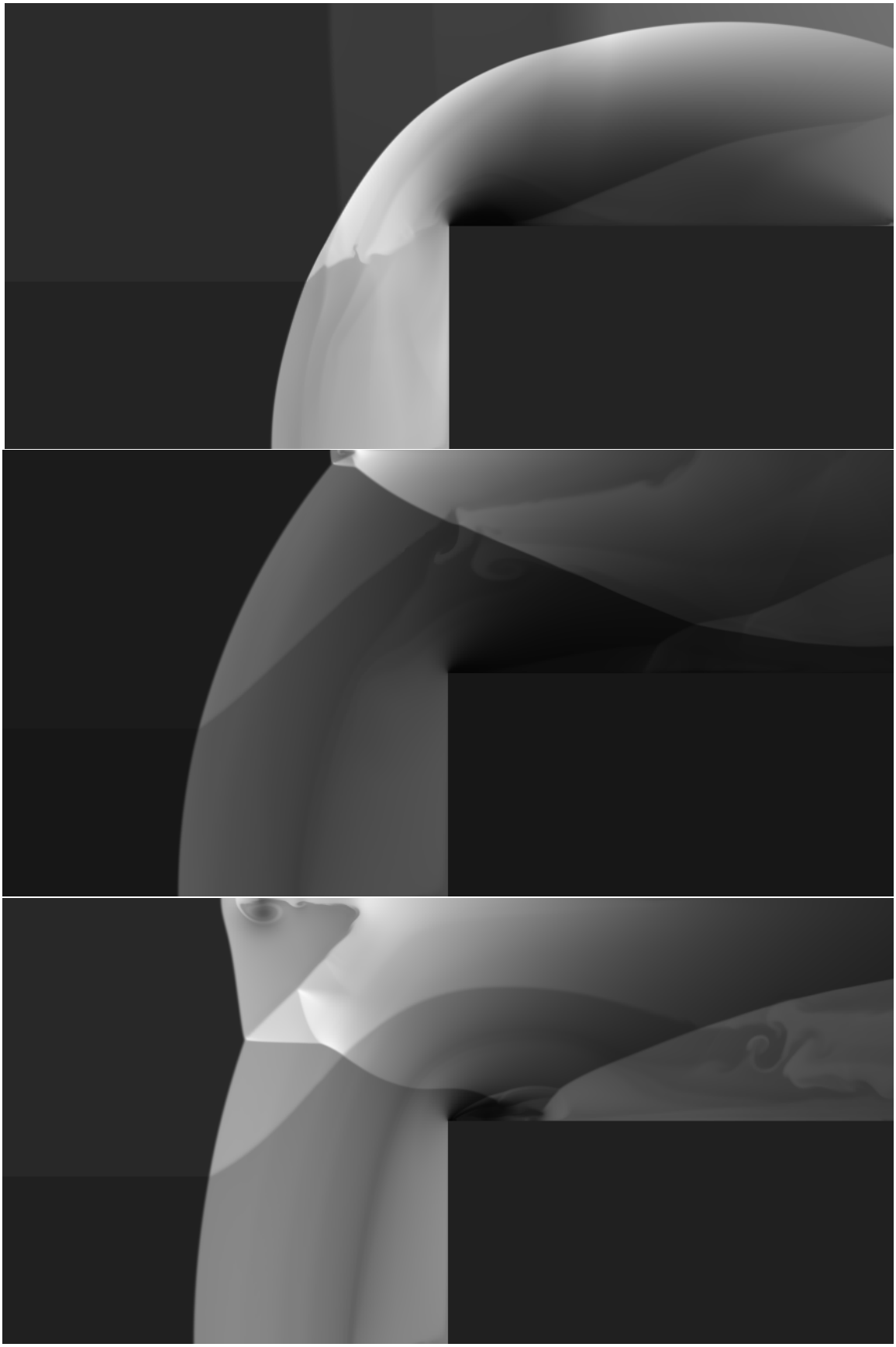
simulations on sufficiently fine meshes to see it develop. Here, in 2002, it has developed clearly on my laptop machine. Now that's progress!

In 1984 we identified the cause and a solution for this instability. In the present version of PPM, a still more effective form of this solution approach has been implemented, which explains why this oscillatory behavior of the post-shock velocity field develops only briefly at one time and location in this problem. Setting the coefficient of the smart diffusion in PPM a bit larger, from its default value of 0.3 to, say, 0.5, completely eliminates this behavior. However, it does so at the cost of spreading out the bow shock so that the definition of the resulting solution is degraded. Therefore, I have chosen to live with this flaw. You may decide otherwise, download the wind tunnel program, change the diffusion constant, and compute away if you please. In any event, this zig-zag feature of

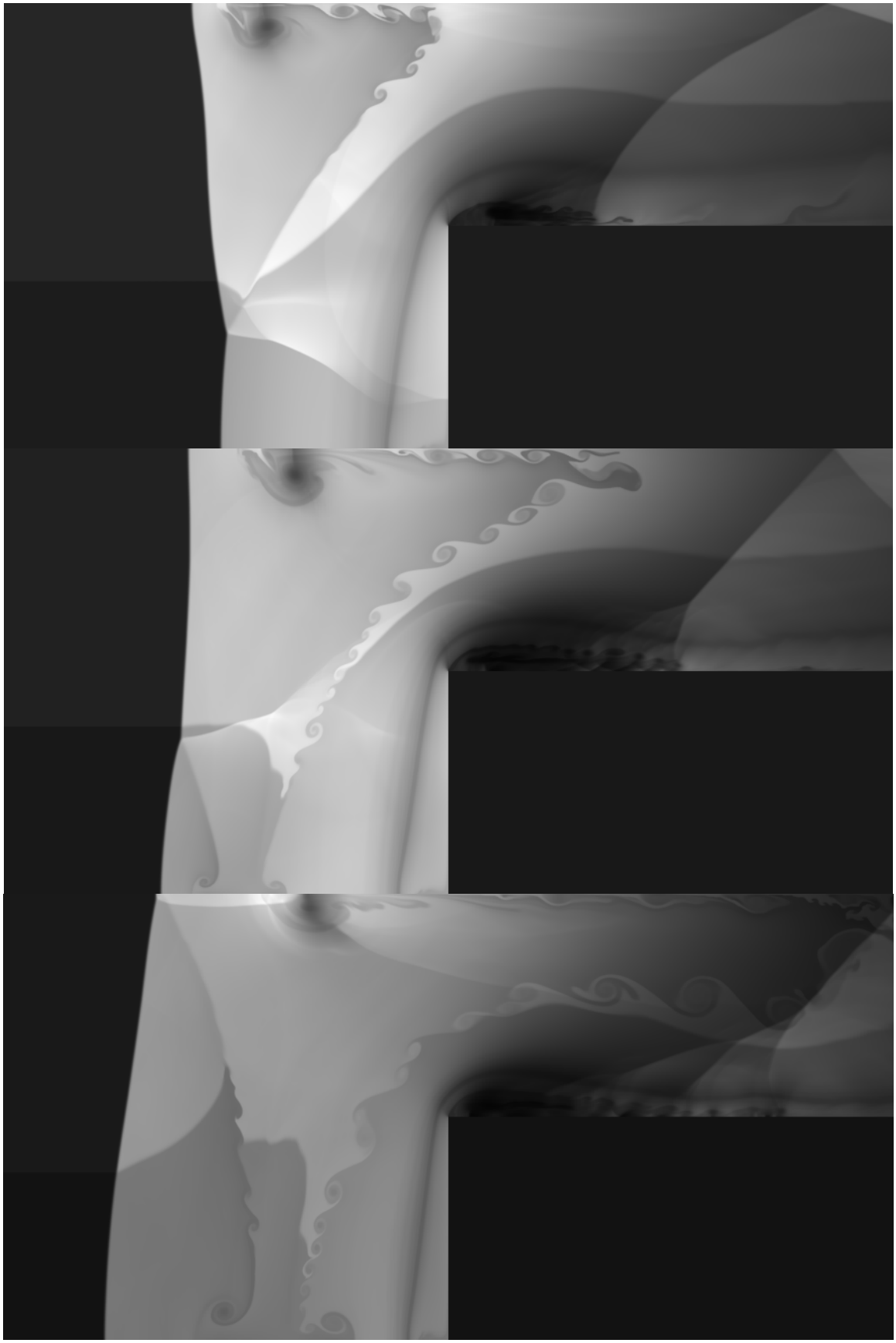
the smoke streams is a numerical error, but it is not an error of the PPB6 scheme, it is an error of the PPM scheme, which is not under discussion in this paper.

It is worth noting that the oscillating velocities should set up sound waves that should propagate throughout this post-shock flow. However, they do not propagate away from the immediate region of the shock front. The reason for this is that their wavelengths are too small, about 3 or 4 cells, to avoid very strong damping by the PPM scheme. But these wavelengths do not deter the PPM6 scheme, with its far greater resolving power. Therefore, for example, the post-shock density and pressure fields are completely unmarred by this slight error of the PPM computation. The zig-zags in the smoke streams show that when we incorporate into PPM the far more accurate PPB6 advection scheme, we must take care that slight errors that we expect the high frequency dissipation of PPM to eliminate are not propagated or, worse, amplified by PPB6. It might therefore be wise to use PPM with a higher setting of its smart shock dissipation, and at the same time to use adaptive mesh refinement right along the shock front so that the greater smearing of this front does not degrade the overall solution.

In order to gain a better understanding of this wind tunnel problem, I have included below sequences of images, taken at the same times as those shown above, showing the density distribution.



PPB Advection Within PPM Gas Dynamics
10/9/02



PPB Advection Within PPM Gas Dynamics
10/9/02