A Primer on Eulerian Computational Fluid Dynamics for Astrophysics

HY TRAC

Department of Astronomy and Astrophysics, University of Toronto, Toronto, ON M5S 3H8, Canada; trac@cita.utoronto.ca

AND

UE-LI PEN

Canadian Institute for Theoretical Astrophysics, 60 St. George Street, Toronto, ON M5S 3H8, Canada; pen@cita.utoronto.ca

ABSTRACT. We present a pedagogical review of some of the methods employed in Eulerian computational fluid dynamics (CFD). Fluid mechanics is governed by the Euler equations, which are conservation laws for mass, momentum, and energy. The standard approach to Eulerian CFD is to divide space into finite volumes or cells and store the cell-averaged values of conserved hydro quantities. The integral Euler equations are then solved by computing the flux of the mass, momentum, and energy across cell boundaries. We review both first-order and second-order flux assignment schemes. All linear schemes are either dispersive or diffusive. The nonlinear, second-order accurate total variation diminishing (TVD) approach provides high-resolution capturing of shocks and prevents unphysical oscillations. We review the relaxing TVD scheme, a simple and robust method to solve systems of conservation laws such as the Euler equations. A three-dimensional relaxing TVD code is applied to the Sedov-Taylor blast-wave test. The propagation of the blast wave is accurately captured and the shock front is sharply resolved. We apply a three-dimensional self-gravitating hydro code to simulating the formation of blue straggler stars through stellar mergers and present some numerical results. A sample three-dimensional relaxing TVD code is provided in the Appendix.

1. INTRODUCTION

Astrophysical structure formation and the dynamics of astrophysical systems involve nonlinear gasdynamical processes that cannot be modeled analytically but require numerical methods. One would like to address the challenging problem of star formation and how this process produces planetary systems. Observations of the X-ray emission from hot gas in galaxy clusters, the Sunyaev-Zeldovich effect in the cosmic microwave background (CMB) spectrum, and the Lyα forest in the spectra of quasars are meaningful only if we understand the gasdynamical processes involved. The evolution of complex systems is best modeled using numerical simulations.

A large class of astrophysical problems involve collisional systems where the mean free path is much smaller than all length scales of interest. Hence, one can appropriately adopt an ideal fluid description of matter where the thermodynamical properties of the fluid obey well-known equations of state. Conservation of mass, momentum, and energy allows one to write the Euler equations that govern fluid mechanics (see Landau & Lifshitz 1987). This formalism is an ideal basis for simulating astrophysical fluids.

Hydrodynamical simulation is faced with challenging problems, but advancements in the field have made it an important tool for theoretical astrophysics. One of the main challenges in simulating complex fluid flows is the capturing of strong shocks, which frequently occur and play an important role in gasdynamics. The differential Euler equations are ill defined at shock discontinuities where derivatives are infinite. Much effort has been devoted to solving this problem, and a field of work has resulted from it. Computational fluid dynamics (CFD) is a powerful approach to simulating fluid flow with emphasis on high-resolution capturing of shocks and prevention of numerical instabilities. Both Eulerian and Lagrangian methods have been developed.

Lagrangian methods based on smoothed-particle hydrodynamics (SPH; Gingold & Monaghan 1977; Lucy 1977) consider a Monte Carlo approximation to solving the fluid equations, somewhat analogous to *N*-body methods for the Vlasov equation. SPH schemes follow the trajectories of particles of fixed mass that represent fluid elements. The Lagrangian forms of the Euler equations are solved to determine smoothed fluid variables such as density, velocity, and temperature. The particle formulation does not naturally capture shocks, and artificial viscosity is added to prevent unphysical oscillations. However, the addition of artificial viscosity broadens shocks over several smoothing lengths and degrades the resolution. The Lagrangian approach has a large dynamic range in length but not in mass. It achieves good spatial resolution in high-density regions but does poorly in low-density regions. SPH schemes must smooth over a large number of neighboring par-

303

ticles, making it computationally expensive and challenging to implement in parallel.

The standard approach to Eulerian methods is to discretize the problem and solve the integral Euler equations on a Cartesian grid by computing the flux of mass, momentum, and energy across grid cell boundaries. In conservative schemes, the flux taken out of one cell is added to the neighboring cell, and this ensures the correct shock propagation. Flux assignment schemes based on the *total variation diminishing condition* (Harten 1983) have been designed for high-order accuracy and high-resolution capturing of shocks, while preventing unphysical oscillations. The Eulerian approach has a large dynamic range in mass but not in length, opposite to that of Lagrangian schemes. In general, Eulerian algorithms are computationally faster by several orders of magnitude. They are also easy to implement and to parallelize.

The purpose of this paper is to present a pedagogical review of some of the methods employed in Eulerian computational fluid dynamics. In § 2, we briefly review the Euler equations and discuss the standard approach to discretizing conservation laws. We describe traditional central differencing methods such as the Lax-Wendroff scheme in § 3 and more modern flux assignment methods such as the total variation diminishing (TVD) scheme in § 4. In § 5, we review the relaxing TVD method for systems of conservation laws such as the Euler equations, which have been successfully implemented for simulating cosmological astrophysical fluids by Pen (1998). In § 6, we apply a self-gravitating hydro code to simulating the formation of blue straggler stars through stellar mergers. A sample three-dimensional relaxing TVD code is provided in the Appendix.

2. EULERIAN HYDRODYNAMICS

The Euler equations, which govern hydrodynamics, are a system of conservation laws for mass, momentum, and energy. In differential conservation form, the continuity equation, momentum equation, and energy equation are given as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho v_j) = 0, \tag{1}$$

$$\frac{\partial(\rho v_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho v_i v_j + P\delta_{ij}) = 0, \tag{2}$$

$$\frac{\partial e}{\partial t} + \frac{\partial}{\partial x_j}[(e + P)v_j] = 0. \tag{3}$$

We have omitted gravitational and other source terms such as heating and cooling. The physical state of the fluid is specified by its density ρ, velocity field v, and total energy density

$$e = \frac{1}{2}\rho v^2 + \epsilon. \tag{4}$$

In practice, the thermal energy ϵ is evaluated by subtracting

the kinetic energy from the total energy. For an ideal gas, the pressure $P(\epsilon)$ is related to the thermal energy by the equation of state

$$P = (\gamma - 1)\epsilon, \tag{5}$$

where γ is the ratio of specific heats. Another thermodynamic variable that is of importance is the sound speed c_s, which is given by

$$c_s^2 \equiv \frac{\partial P}{\partial \rho} = \frac{\gamma P}{\rho}. \tag{6}$$

The thermodynamical properties of an ideal gas obey well-known equations of state, which we do not fully list here.

The differential Euler equations require differentiable solutions and therefore are ill defined at jump discontinuities where derivatives are infinite. In the literature, nondifferentiable solutions are called *weak solutions*. The differential form gives a complete description of the flow in smooth regions, but the integral form is needed to properly describe shock discontinuities. In integral conservation form, the rate of change in mass, momentum, and energy is equal to the net flux of those conserved quantities through the surface enclosing a control volume. For simplicity of notation, we will continue to express conservation laws in differential form, as a shorthand for the integral form.

2.1. Computational Fluid Dynamics

The standard approach to Eulerian computational fluid dynamics is to discretize time into discrete steps and space into finite volumes or cells, where the conserved quantities are stored. In the simplest case, the integral Euler equations are solved on a Cartesian cubical lattice by computing the flux of mass, momentum, and energy across cell boundaries in discrete time steps. Consider the Euler equations in vector differential conservation form:

$$\frac{\partial u}{\partial t} + \frac{\partial F_i(u)}{\partial x_i} = 0, \tag{7}$$

where $u = (\rho, \rho v_x, \rho v_y, \rho v_z, e)$ contains the conserved physical quantities and $F(u)$ represents the flux terms. In practice, the conserved cell-averaged quantities $u_n \equiv u(x_n)$ and fluxes F_n are defined at integer grid cell centers x_n. The challenge is to use the cell-averaged values to determine the fluxes $F_{n+1/2}$ at cell boundaries.

In the following sections, we describe flux assignment methods designed to solve conservation laws such as the Euler equations. For ease of illustration, we begin by considering a

one-dimensional scalar conservation law:

$$\frac{\partial u}{\partial t} + \frac{\partial F(u)}{\partial x} = 0, \tag{8}$$

where $F(u) = vu$ and v is a constant advection velocity. Equation (8) is referred to as a linear advection equation and has the analytical solution

$$u(x, t) = u(x - vt, 0). \tag{9}$$

The linear advection equation describes the transport of the quantity u at a constant velocity v.

In integral flux conservation form, the one-dimensional scalar conservation law can be written as

$$\frac{\partial}{\partial t} \int_{x_1}^{x_2} u(x, t)dx + \int_{x_1}^{x_2} \frac{\partial F(u)}{\partial x} dx = 0, \tag{10}$$

where $x_1 \equiv x_{n-1/2}$ and $x_2 \equiv x_{n+1/2}$ for our control cells. Let $F_{n+1/2}^t$ denote the flux of u through cell boundary $x_{n+1/2}$ at time t. Note then that the second integral is simply equal to $F_{n+1/2}^t - F_{n-1/2}^t$. The rate of change in the cell-integrated quantity $\int u \, dx$ is equal to the net flux of u through the control cell. For a discrete time step, the discretized solution for the cell-averaged quantity u_n is given by

$$u_n^{t+\Delta t} = u_n^t - \left(\frac{F_{n+1/2}^t - F_{n-1/2}^t}{\Delta x}\right) \Delta t. \tag{11}$$

The physical quantity u is conserved since the flux taken out of one cell is added to the neighboring cell that shares the same boundary. Note that equation (11) has the appearance of being a finite-difference scheme for solving the differential form of the one-dimensional scalar conservation law. This is why the differential form can be used as a shorthand for the integral form.

3. CENTERED FINITE-DIFFERENCE METHODS

Central-space finite-difference methods have ease of implementation but at the cost of lower accuracy and stability. For illustrative purposes, we start with a simple first-order centered scheme to solve the linear advection equation. The discretized solution is given by equation (11), where the fluxes at cell boundaries

$$F_{n+1/2}^t = \frac{F_{n+1}^t + F_n^t}{2} \tag{12}$$

are obtained by taking an average of cell-centered fluxes $F_n^t = vu_n^t$. The discretized first-order centered scheme can be

equivalently written as

$$u_n^{t+\Delta t} = u_n^t - \left(\frac{F_{n+1}^t - F_{n-1}^t}{2\Delta x}\right) \Delta t. \tag{13}$$

In this form, the discretization has the appearance of using a central difference scheme to approximate spatial derivatives. Hence, centered schemes are often referred to as central difference schemes. In practice, when using centered schemes, the discretization is done on the differential conservation equation rather than the integral equation.

This simple scheme is numerically unstable and we can show this using the *von Neumann* linear stability analysis. Consider writing $u(x, t)$ as a discrete Fourier series:

$$u_n^t = \frac{1}{N} \sum_{k=-N/2}^{N/2} c_k^t \exp\left(\frac{2\pi ikn}{N}\right), \tag{14}$$

where N is the number of cells in our periodic box. In plane-wave solution form, we can write this as

$$u_n^t = \frac{1}{N} \sum_{k=-N/2}^{N/2} c_k^0 \exp\left[\frac{2\pi i(kn - \omega t)}{N}\right], \tag{15}$$

where c_k^0 are the Fourier series coefficients for the initial conditions $u(x, 0)$. Equivalently, the time evolution of the Fourier series coefficients in equation (14) can be cast into a plane-wave solution of the form

$$c_k^t = \exp\left(\frac{-2\pi i\omega t}{N}\right) c_k^0, \tag{16}$$

where the numerical dispersion relation $\omega(k)$ is complex in general. The imaginary part of ω represents the growth or decay of the Fourier modes, while the real part describes the oscillations. A numerical scheme is linearly stable if $\mathrm{Im}(\omega) \leq 0$. Otherwise, the Fourier modes will grow exponentially in time and the solution will blow up.

The exact solution to the linear advection equation can be expressed in the form of equation (9) or by a plane-wave solution where the dispersion relation is given by $\omega_0 = vk$. The waves all travel at the same phase velocity $\omega_0/k = v$ in the exact case.

The centrally discretized linear advection equation (eq. [13]) is exactly solvable. After m time steps, the time evolution of the independent Fourier modes is given by

$$c_k^{m\Delta t} = (1 - i\lambda \sin \phi)^m c_k^0, \tag{17}$$

where $\lambda \equiv v\Delta t/\Delta x$ and $\phi = 2\pi k\Delta x/N$. The dispersion relation

is given by

$$\omega = \frac{N}{2\pi\Delta t}\left[\tan^{-1}(\lambda\sin\phi) + \frac{i}{2}\ln(1 + \lambda^2\sin^2\phi)\right]. \quad (18)$$

For any time step $\Delta t > 0$, the imaginary part of ω will be greater than 0. The Fourier modes will grow exponentially in time and the solution will blow up. Hence, the first-order centered scheme is numerically unstable.

3.1. Lax-Wendroff Scheme

The Lax-Wendroff scheme (Lax & Wendroff 1960) is second-order accurate in time and space, and the idea behind it is to stabilize the unstable first-order scheme from the previous section. Consider a Taylor series expansion for $u(x, t + \Delta t)$:

$$u(x, t + \Delta t) = u(x, t) + \frac{\partial u}{\partial t}\Delta t + \frac{\partial^2 u}{\partial t^2}\frac{\Delta t^2}{2} + O(\Delta t^3), \quad (19)$$

and replace the time derivatives with spatial derivatives using the conservation law to obtain

$$u(x, t + \Delta t) = u(x, t) - \frac{\partial F}{\partial x}\Delta t$$
$$+ \frac{\partial}{\partial x}\left(\frac{\partial F}{\partial u}\frac{\partial F}{\partial u}\frac{\partial u}{\partial x}\right)\frac{\Delta t^2}{2} + O(\Delta t^3). \quad (20)$$

For the linear advection equation, the eigenvalue of the flux Jacobian is $\partial F/\partial u = v$. Discretization using central differences gives

$$u_n^{t+\Delta t} = u_n^t - \frac{F_{n+1}^t - F_{n-1}^t}{2\Delta x}\Delta t$$
$$+ \left(\frac{F_{n+1}^t - F_n^t}{\Delta x} - \frac{F_n^t - F_{n-1}^t}{\Delta x}\right)\frac{v\Delta t^2}{2\Delta x}. \quad (21)$$

In conservation form, the solution is given by equation (11), where the fluxes at cell boundaries are defined as

$$F_{n+1/2}^t = \frac{1}{2}(F_{n+1}^t + F_n^t) - (F_{n+1}^t - F_n^t)\frac{v\Delta t}{2\Delta x}. \quad (22)$$

Compare this with the boundary fluxes for the first-order scheme (eq. [12]). The Lax-Wendroff scheme obtains second-order fluxes,

$$F^{(2)} = F^{(1)} - \frac{\partial F}{\partial u}\frac{\partial F}{\partial x}\frac{\Delta t}{2}, \quad (23)$$

by modifying the first-order fluxes $F^{(1)}$ with a second-order correction.

The stability of the Lax-Wendroff scheme to solve the linear advection equation can also be determined using the von Neumann analysis. The discretized Lax-Wendroff equation (eq. [21]) is exactly solvable and, after m time steps, the Fourier modes evolve according to

$$c_k^{m\Delta t} = \left[1 - \lambda^2(1 - \cos\phi) - i\lambda\sin\phi\right]^m c_k^0, \quad (24)$$

where $\lambda \equiv v\Delta t/\Delta x$ is called the *Courant* number and $\phi = 2\pi k\Delta x/N$. The dispersion relation is given by

$$\omega = \frac{N}{2\pi\Delta t}\tan^{-1}\left[\frac{\lambda\sin\phi}{1 - \lambda^2(1 - \cos\phi)}\right]$$
$$+ \frac{iN}{4\pi\Delta t}\ln\left[1 - 4\lambda^2(1 - \lambda^2)\sin^4\left(\frac{\phi}{2}\right)\right]. \quad (25)$$

It is important to note three things. First, the Lax-Wendroff scheme is conditionally stable provided that $\text{Im}(\omega) \leq 0$, which is satisfied if

$$\frac{v\Delta t}{\Delta x} \leq 1. \quad (26)$$

This constraint is a particular example of a general stability constraint known as the *Courant-Friedrichs-Lewy* (CFL) condition. The Courant number λ is also referred to as the CFL number. Second, for $\lambda = 1$, the dispersion relation is exactly identical to that of the exact solution, and the numerical advection is exact. This is a special case, however, and it does not test the ability of the Lax-Wendroff scheme to solve general scalar conservation laws. Normally, one chooses $\lambda < 1$ to satisfy the CFL condition. Last, for $\lambda < 1$, the dispersion relation $\omega(k)$ for the Lax-Wendroff solution is different from the exact solution where $\omega_0 = vk$. The dispersion relation relative to the exact solution can be parameterized by

$$\Delta\omega \equiv \omega - \omega_0. \quad (27)$$

The second-order truncation of the Taylor series (eq. [20]) results in a phase error $\text{Re}(\Delta\omega)$, which is a function of frequency. In the Lax-Wendroff solution, the waves are damped and travel at different speeds. Hence, the scheme is both diffusive and dispersive.

In Figure 1, we plot the phase error $\text{Re}(\Delta\omega)$ and the amplification term $\text{Im}(\Delta\omega)$ for the Lax-Wendroff scheme with parameters $N = 100$, $v = 1$, and $\lambda = 0.9$. A negative value of $\text{Re}(\Delta\omega)$ represents a lagging phase error, while a positive value indicates a leading phase error. For the chosen CFL number, the high-frequency modes have the largest phase errors, but they are highly damped. Some of the modes having lagging phase errors are not highly damped. We will subsequently see how this becomes important.

A rigorous test of the one-dimensional Lax-Wendroff scheme and other flux assignment schemes we will discuss is the linear
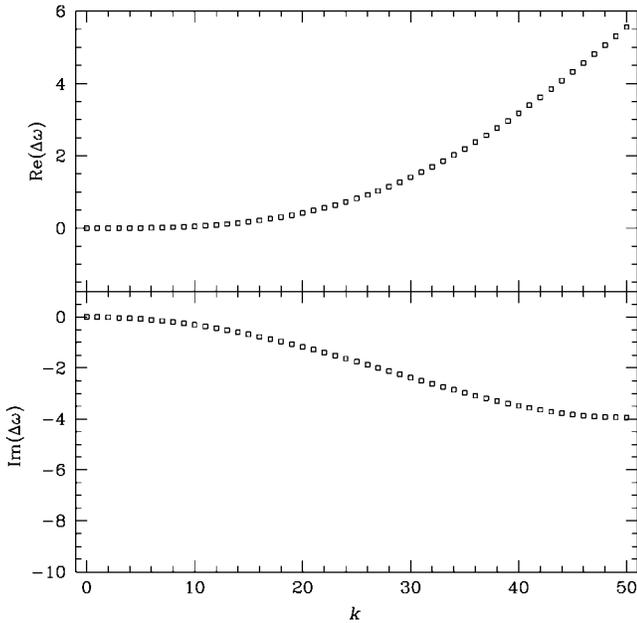
FIG. 1.—Phase error $\text{Re}(\Delta\omega)$ and the amplification factor $\text{Im}(\Delta\omega)$ for the Lax-Wendroff scheme with parameters $N = 100$, $v = 1$, and $\lambda = 0.9$.

FIG. 2.—Lax-Wendroff scheme used to linearly advect a square wave (*solid line*) once (*dashed line*) and 10 times (*dotted line*) through a box of 100 grid cells at speed $v = 1$.

advection of a square wave. The challenge is to accurately advect this discontinuous function where the edges mimic Riemann shock fronts. In Figure 2, we show how the Lax-Wendroff scheme does at advecting the square wave once (*dashed line*) and 10 times (*dotted line*) through a periodic box of 100 grid cells at speed $v = 1$ and $\lambda = 0.9$. Note that this scheme produces numerical oscillations. Recall that a square wave can be represented by a sum of Fourier or sine waves. These waves will be damped and disperse when advected using the Lax-Wendroff scheme. Figure 1 shows that the modes having lagging phase errors are not damped away. Hence, the Lax-Wendroff scheme is highly dispersive and the oscillations in Figure 2 are due to dispersion. We leave it as an exercise for the reader to advect a sine wave using the Lax-Wendroff scheme. Since there is only one frequency mode in this case, there will be no spurious oscillations due to dispersion, but a phase error will be present. For a comprehensive discussion on the family of Lax-Wendroff schemes and other centered schemes, see Hirsch (1990) and Laney (1998).

4. UPWIND METHODS

Upwind methods take into account the physical nature of the flow when assigning fluxes for the discrete solution. This class of flux assignment schemes, whose origin dates back to the work of Courant, Isaason, & Reeves (1952), has been shown to be excellent at capturing shocks and also being highly stable.

We start with a simple first-order upwind scheme to solve the linear advection equation. Consider the case where the advection velocity is positive and flow is to the right. The flux of the physical quantity u through the cell boundary $x_{n+1/2}$ will

originate from cell n. The upwind scheme proposes that, to first order, the fluxes $F_{n+1/2}^{t}$ at cell boundaries be taken from the cell-centered fluxes $F_n^t = vu_n^t$, which is in the upwind direction. If the advection velocity is negative and flow is to the left, the boundary fluxes $F_{n+1/2}^t$ are taken from the cell-centered fluxes $F_{n+1}^t = vu_{n+1}^t$. The first-order upwind flux assignment scheme can be summarized as follows:

$$F_{n+1/2}^t = \begin{cases} F_n^t & \text{if } v > 0, \\ F_{n+1}^t & \text{if } v < 0. \end{cases} \tag{28}$$

Unlike central difference schemes, upwind schemes are explicitly asymmetric.

The CFL condition for the first-order upwind scheme can be determined from the von Neumann analysis. We consider the case of a positive advection velocity. After m time steps, the Fourier modes evolve according to

$$c_k^{m\Delta t} = [1 - \lambda(1 - \cos\phi) - i\lambda\sin\phi]^m c_k^0, \tag{29}$$

where $\lambda \equiv v\Delta t/\Delta x$ and $\phi = 2\pi k\Delta x/N$. The dispersion relation is given by

$$\omega = \frac{N}{2\pi\Delta t}\tan^{-1}\left[\frac{\lambda\sin\phi}{1 - \lambda(1 - \cos\phi)}\right]$$

$$+ \frac{iN}{4\pi\Delta t}\ln\left[1 - 4\lambda(1 - \lambda)\sin^2\left(\frac{\phi}{2}\right)\right]. \tag{30}$$

The CFL condition for solving the linear advection equation
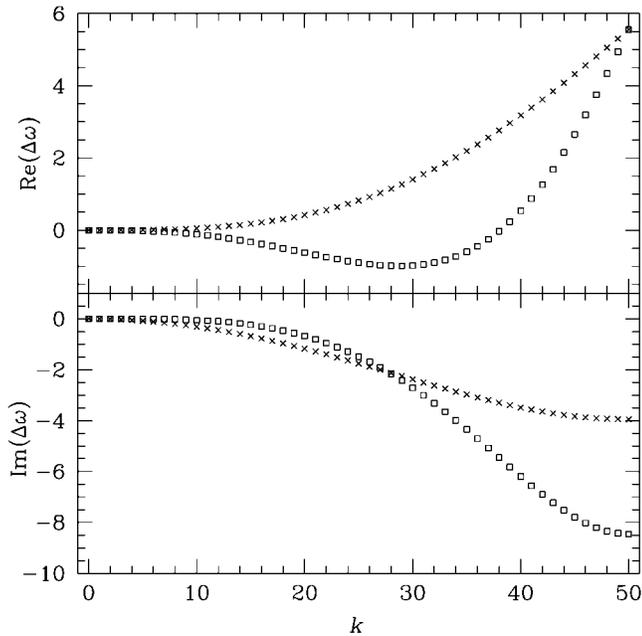
FIG. 3.—Phase error Re($\Delta\omega$) and the amplification term Im($\Delta\omega$) for the Lax-Wendroff scheme (*boxes*) and the first-order upwind scheme (*crosses*) with parameters $N = 100$, $v = 1$, and $\lambda = 0.9$.

FIG. 4.—First-order upwind scheme used to linearly advect a square wave (*solid line*) once (*dashed line*) and 10 times (*dotted line*) through a box of 100 grid cells at speed $v = 1$.

with this scheme is to have $\lambda \leq 1$, identical to that for the Lax-Wendroff scheme. For $\lambda < 1$, the dispersion relation $\omega(k)$ for the first-order upwind scheme is different from the exact solution where $\omega_0 = vk$. This scheme is both diffusive and dispersive. Since it is only first-order accurate, the amount of diffusion is large. In Figure 3, we compare the dispersion relation of the upwind scheme to that of the Lax-Wendroff scheme. The Fourier modes in the upwind scheme also have phase errors, but they will be damped away. The low-frequency modes that contribute to the oscillations in the Lax-Wendroff solution are more damped in the upwind solution. Hence, one does not expect to see oscillations resulting from phase errors.

In Figure 4, we show how the first-order upwind scheme does at advecting the Riemann shock wave. This scheme is well behaved and produces no spurious oscillations, but since it is only first order, it is highly diffusive. The first-order upwind scheme has the property of having monotonicity preservation. When applied to the linear advection equation, it does not allow the creation of new extrema in the form of spurious oscillations. The Lax-Wendroff scheme does not have the property of having monotonicity preservation.

The flux assignment schemes that we have discussed so far are all linear schemes. Godunov (1959) showed that all linear schemes are either diffusive or dispersive or a combination of both. The Lax-Wendroff scheme is highly dispersive, while the first-order upwind scheme is highly diffusive. Godunov's theorem also states that linear monotonicity preserving schemes are only first-order accurate. In order to obtain higher order

accuracy and prevent spurious oscillations, nonlinear schemes are needed to solve conservation laws.

4.1. Total Variation Diminishing Schemes

Harten (1983) proposed the TVD condition, which guarantees that a scheme have monotonicity preservation. Applying Godunov's theorem, we know that all linear TVD schemes are only first-order accurate. In fact, the only linear TVD schemes are the class of first-order upwind schemes. Therefore, higher order accurate TVD schemes must be nonlinear.

The TVD condition is a nonlinear stability condition. The total variation of a discrete solution, defined as

$$\text{TV}(u^t) = \sum_{i=1}^{N} |u_{i+1}^t - u_i^t|, \qquad (31)$$

is a measure of the overall amount oscillations in u. The direct connection between the total variation and the overall amount of oscillations can be seen in the equivalent definition

$$\text{TV}(u^t) = 2\left(\sum u_{\max} - \sum u_{\min} \right), \qquad (32)$$

where each maxima is counted positively twice and each minima is counted negatively twice (see Laney 1998). The formation of spurious oscillations will contribute new maxima and minima and the total variation will increase. A flux

assignment scheme is said to be TVD if

$$TV(u^{t+\Delta t}) \leq TV(u^t), \qquad (33)$$

which signifies that the overall number of oscillations is bounded. In linear flux-assignment schemes, the von Neumann linear stability condition requires that the Fourier modes remain bounded. In nonlinear schemes, the TVD stability condition requires that the total variation diminishes.

We now describe a nonlinear second-order accurate TVD scheme that builds upon the first-order monotone upwind scheme described in the previous section. The second-order accurate fluxes $F_{n+1/2}^t$ at cell boundaries are obtained by taking first-order fluxes $F_{n+1/2}^{(1),t}$ from the upwind scheme and modifying it with a second-order correction. First, consider the case where the advection velocity is positive. The first-order upwind flux $F_{n+1/2}^{(1),t}$ comes from the averaged flux F_n^t in cell n. We can define two second-order flux corrections,

$$\Delta F_{n+1/2}^{L,t} = \frac{F_n^t - F_{n-1}^t}{2}, \qquad (34)$$

$$\Delta F_{n+1/2}^{R,t} = \frac{F_{n+1}^t - F_n^t}{2}, \qquad (35)$$

using three local cell-centered fluxes. We use cell n and the cells immediately left and right of it. If the advection velocity is negative, the first-order upwind flux comes from the averaged flux F_{n+1}^t in cell $n + 1$. In this case, the second-order flux corrections,

$$\Delta F_{n+1/2}^{L,t} = -\frac{F_{n+1}^t - F_n^t}{2}, \qquad (36)$$

$$\Delta F_{n+1/2}^{R,t} = -\frac{F_{n+2}^t - F_{n+1}^t}{2}, \qquad (37)$$

are based on cell $n + 1$ and the cells directly adjacent to it. Near extrema where the corrections have opposite signs, we impose no second-order correction, and the flux assignment scheme reduces to first-order. A flux limiter ϕ is then used to determine the appropriate second-order correction,

$$\Delta F_{n+1/2}^t = \phi(\Delta F_{n+1/2}^{L,t}, \Delta F_{n+1/2}^{R,t}), \qquad (38)$$

which still maintains the TVD condition. The second-order correction is added to the first-order fluxes to get second-order fluxes. The first-order upwind scheme and second-order TVD scheme will be referred to as *monotone upwind schemes for conservation laws* (MUSCL).

Time integration is performed using a second-order Runge-Kutta scheme. We first do a half time step:

$$u_n^{t+\Delta t/2} = u_n^t - \left(\frac{F_{n+1/2}^t - F_{n-1/2}^t}{\Delta x}\right)\frac{\Delta t}{2}, \qquad (39)$$

using the first-order upwind scheme to obtain the half-step values $u^{t+\Delta t/2}$. A full time step is then computed:

$$u_n^{t+\Delta t} = u_n^t - \left(\frac{F_{n+1/2}^{t+\Delta t/2} - F_{n-1/2}^{t+\Delta t/2}}{\Delta x}\right)\Delta t, \qquad (40)$$

using the TVD scheme on the half-step fluxes $F^{t+\Delta t/2}$. The reader is encouraged to show that it is second-order accurate.

We briefly discuss three TVD limiters. The minmod flux limiter chooses the smallest absolute value between the left and right corrections:

$$\text{minmod}(a, b) = \frac{1}{2}[\text{sign}(a) + \text{sign}(b)]\min(|a|, |b|). \qquad (41)$$

The superbee limiter (Roe 1985) chooses between the larger correction and 2 times the smaller correction, whichever is smaller in magnitude:

$$\text{superbee}(a, b) = \begin{cases} \text{minmod}(a, 2b) & \text{if } |a| \geq |b|, \\ \text{minmod}(2a, b) & \text{otherwise.} \end{cases} \qquad (42)$$

The Van Leer limiter (Van Leer 1974) takes the harmonic mean of the left and right corrections:

$$\text{vanleer}(a, b) = \frac{2ab}{a + b}. \qquad (43)$$

The minmod limiter is the most moderate of all second-order TVD limiters. In Figure 5, we see that it does not do much better than first-order upwind for the square-wave advection test. Superbee chooses the maximum correction allowed under the TVD constraint. It is especially suited for piecewise linear conditions and is the least diffusive for this particular test, as can be seen in Figure 6. Note that no additional diffusion can be seen by advecting the square wave more than once through the box. It can be shown that the minmod and superbee limiters are extreme cases that bound all other second-order TVD limiters. The Van Leer limiter differs from the previous two in that it is analytic. This symmetrical approach falls somewhere in between the other two limiters in terms of moderation and diffusion, as can be seen in Figure 7. It can be shown that the CFL condition for the second-order TVD scheme is to have $\lambda < 1$. For a comprehensive discussion on TVD limiters, see Hirsch (1990) and Laney (1998).
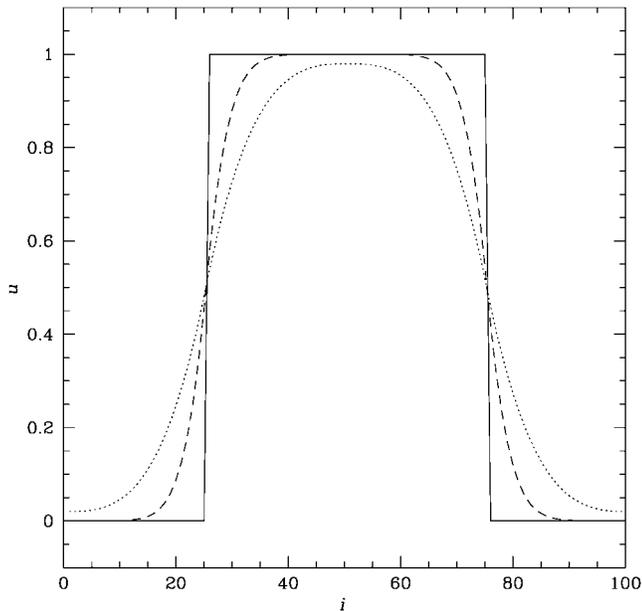
FIG. 5.—TVD scheme using the minmod flux limiter is applied to the advection of a square wave.

5. RELAXING TVD

We now describe a simple and robust method to solve the Euler equations using the MUSCL from the previous section. The relaxing TVD method (Jin & Xin 1995) provides high-resolution capturing of shocks using computationally inexpensive algorithms that are straightforward to implement and to parallelize. It has been successfully implemented for simulating cosmological astrophysical fluids by Pen (1998).

The MUSCL scheme is straightforward to apply to conservation laws such as the advection equation since the velocity alone can be used as a marker of the direction of flow. However, applying the MUSCL scheme to solve the Euler equations is made difficult by the fact that the momentum and energy fluxes depend on the pressure. In order to determine the direction upwind of the flow, it becomes necessary to calculate the flux Jacobian eigenvectors using Riemann solvers. This step requires computationally expensive algorithms. The relaxing TVD method offers an attractive alternative.

5.1. One-dimensional Scalar Conservation Law

We first present a motivation for the relaxing scheme by again considering the one-dimensional scalar conservation law. The MUSCL scheme for solving the linear advection equation is explicitly asymmetric in that it depends on the sign of the advection velocity. We now describe a symmetrical approach that applies to a general advection velocity.

The flow can be considered as a sum of a right-moving wave u^R and a left-moving wave u^L. For a positive advection velocity, the amplitude of the left-moving wave is zero, and for a negative advection velocity, the amplitude of the right-moving wave

is zero. In compact notation, the waves can be defined as

$$u^R = \left(\frac{1 + v/c}{2}\right) u, \tag{44}$$
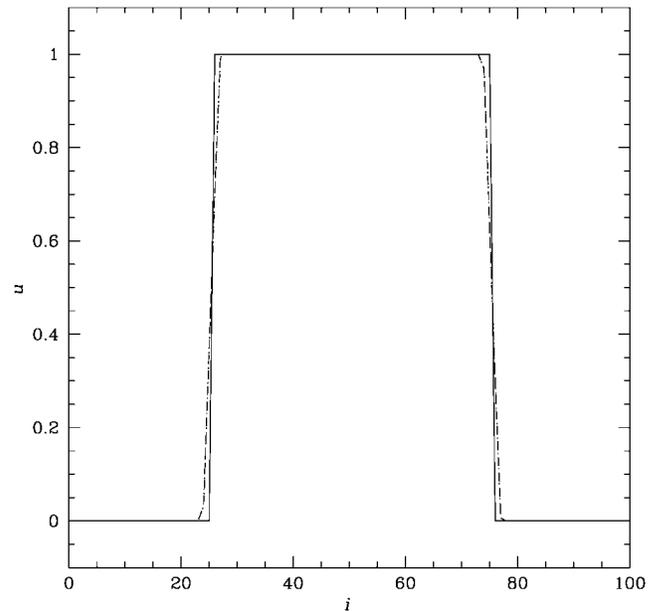
$$u^L = \left(\frac{1 - v/c}{2}\right) u, \tag{45}$$

FIG. 6.—TVD scheme using the superbee flux limiter is applied to the advection of a square wave.
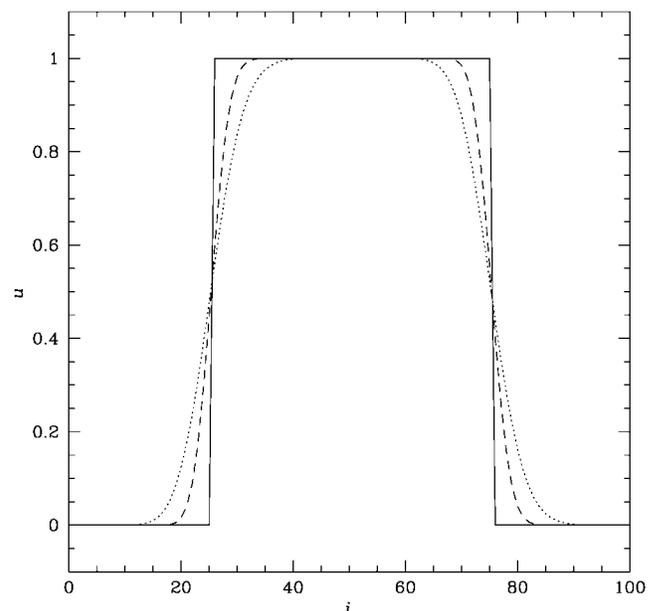
FIG. 7.—TVD scheme using the Van Leer flux limiter is applied to the advection of a square wave.

where $c = |v|$. The two waves are traveling in opposite directions with advection speed c and can be described by the advection equations:

$$\frac{\partial u^R}{\partial t} + \frac{\partial}{\partial x}(cu^R) = 0, \tag{46}$$

$$\frac{\partial u^L}{\partial t} - \frac{\partial}{\partial x}(cu^L) = 0. \tag{47}$$

The MUSCL scheme is straightforward to apply to solve equations (46) and (47) since the upwind direction is left for the right-moving wave and right for the left-moving wave. The one-dimensional relaxing advection equation then becomes

$$\frac{\partial u}{\partial t} + \frac{\partial F^R}{\partial x} - \frac{\partial F^L}{\partial x} = 0, \tag{48}$$

where $F^R = cu^R$ and $F^L = cu^L$. For the discretized solution given by equation (11), the boundary fluxes $F_{n+1/2}$ are now a sum of the fluxes $F_{n+1/2}^{R,t}$ and $F_{n+1/2}^{L,t}$ from the right-moving and left-moving waves, respectively. Note that the relaxing advection equation will correctly reduce to the linear advection equation for any general advection velocity.

Using this symmetrical approach, a general algorithm can be written to solve the linear advection equation for an arbitrary advection velocity. This scheme is indeed inefficient for solving the linear advection equation since one wave will have zero amplitude. However, the Euler equations can have both right-moving and left-moving waves with nonzero amplitudes.

5.2. One-dimensional Systems of Conservation Laws

We now discuss the one-dimensional relaxing TVD scheme and later generalize it to higher spatial dimensions. Consider a one-dimensional system of conservation laws:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{F}(u)}{\partial x} = 0, \tag{49}$$

where for the Euler equations, we have $\boldsymbol{u} = (\rho, \rho v, e)$ and $\boldsymbol{F}(\boldsymbol{u})$, the corresponding flux terms. We now replace the vector conservation law with the relaxation system

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial}{\partial x}(c\boldsymbol{w}) = 0, \tag{50}$$

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial}{\partial x}(c\boldsymbol{u}) = 0, \tag{51}$$

where $c(x, t)$ is a free positive function called the freezing speed. The relaxation system contains two coupled-vector linear advection equations. In practice, we set $\boldsymbol{w} = \boldsymbol{F}(\boldsymbol{u})/c$ and use it as an auxiliary vector to calculate fluxes. Equation (50) reduces to our one-dimensional vector conservation law, and equation (51) is a vector conservation law for $\boldsymbol{w}$.

In order to solve the relaxed system, we decouple the equations through a change of variables:

$$\boldsymbol{u}^R = \frac{\boldsymbol{u} + \boldsymbol{w}}{2}, \tag{52}$$

$$\boldsymbol{u}^L = \frac{\boldsymbol{u} - \boldsymbol{w}}{2}, \tag{53}$$

which then gives us

$$\frac{\partial \boldsymbol{u}^R}{\partial t} + \frac{\partial}{\partial x}(c\boldsymbol{u}^R) = 0, \tag{54}$$

$$\frac{\partial \boldsymbol{u}^L}{\partial t} - \frac{\partial}{\partial x}(c\boldsymbol{u}^L) = 0. \tag{55}$$

Equations (54) and (55) are vector linear advection equations, which can be interpreted as right-moving and left-moving flows with advection speed c. Note the similarity with their scalar counterparts, equations (46) and (47). The one-dimensional vector-relaxing conservation law for $\boldsymbol{u}$ becomes

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{F}^R}{\partial x} - \frac{\partial \boldsymbol{F}^L}{\partial x} = 0, \tag{56}$$

where $\boldsymbol{F}^R = c\boldsymbol{u}^R$ and $\boldsymbol{F}^L = c\boldsymbol{u}^L$. The vector-relaxing equation can now be solved by applying the MUSCL scheme to equations (54) and (55). Again, note the similarity between the vector-relaxing equation and its scalar counterpart, equation (48).

The relaxed scheme is TVD under the constraint that the freezing speed c be greater than the characteristic speed given by the largest eigenvalue of the flux Jacobian $\partial \boldsymbol{F}(\boldsymbol{u})/\partial \boldsymbol{u}$. For the Euler equations, this is satisfied for

$$c = |v| + c_s. \tag{57}$$

Jin & Xin (1995) considered the freezing speed to be a positive constant in their relaxing scheme, while we generalize it to be a positive function. Time integration is again performed using a second-order Runge-Kutta scheme, and the time step is determined by satisfying the CFL condition

$$\frac{c_{max}\Delta t}{\Delta x} \le 1. \tag{58}$$

Note that a new freezing speed is computed for each partial step in the Runge-Kutta scheme. The CFL number $\lambda = c_{max}\Delta t/\Delta x$ should be chosen such that c_{max} will be larger than $\max(c_n^t)$ and $\max(c_n^{t+\Delta t/2})$.

We now summarize the steps needed to numerically solve the one-dimensional Euler equations. At the beginning of each partial step in the Runge-Kutta time integration scheme, we need to calculate the cell-averaged variables defined at grid cell centers. First for the half time step, we calculate the fluxes

$F(u_n^t)$ and the freezing speed c_n^t. We then set the auxiliary vector $w_n^t = F(u_n^t)/c_n^t$ and construct the right-moving waves $u_n^{R,t}$ and left-moving waves $u_n^{L,t}$. The half time step is given by

$$u_n^{t+\Delta t/2} = u_n^t - \left(\frac{F_{n+1/2}^t - F_{n-1/2}^t}{\Delta x}\right)\frac{\Delta t}{2}, \tag{59}$$

where

$$F_{n+1/2}^t = F_{n+1/2}^{R,t} - F_{n+1/2}^{L,t}. \tag{60}$$

The first-order upwind scheme is used to compute fluxes at cell boundaries for the right-moving and left-moving waves. For the full time step, we construct the right-moving waves $u_n^{R,t+\Delta t/2}$ and left-moving waves $u_n^{L,t+\Delta t/2}$, using the half-step values of the appropriate variables. The full time step,

$$u_n^{t+\Delta t} = u_n^t - \left(\frac{F_{n+1/2}^{t+\Delta t/2} - F_{n-1/2}^{t+\Delta t/2}}{\Delta x}\right)\Delta t, \tag{61}$$

is computed using the second-order TVD scheme. This completes the updating of u^t to $u^{t+\Delta t}$.

We have found that a minor modification to the implementation described above gives more accurate results. Consider writing the flux of the right-moving and left-moving waves as

$$F^R = cG^R, \tag{62}$$

$$F^L = cG^L, \tag{63}$$

where G^R is the flux of $\mu^R = u^R/c$ and G^L is the flux of $\mu^L = u^L/c$. The linear advection equations for μ^R and μ^L are similar to equations (54) and (55), but where we replace u^R with μ^R and u^L with μ^L. For each partial step in the Runge-Kutta scheme, the net fluxes at cell boundaries are then taken to be

$$F_{n+1/2} = c_{n+1/2}(G_{n+1/2}^R - G_{n+1/2}^L), \tag{64}$$

where we use $c_{n+1/2} = (c_{n+1} + c_n)/2$. In practice, this modified implementation has been found to resolve shocks with better accuracy in certain cases. Note that the two different implementations of the relaxing TVD scheme are identical when a constant freezing speed is used.

5.3. Multidimensional Systems of Conservation Laws

The one-dimensional relaxing TVD scheme can be generalized to higher dimensions using the dimensional splitting technique by Strang (1968). In three dimensions, the Euler equations can be dimensionally split into three separate one-dimensional equations that are solved sequentially. Let the operator L_i represent the updating of u^t to $u^{t+\Delta t}$ by including the flux in the i direction. We first complete a forward sweep:

$$u^{t+\Delta t} = L_z L_y L_x u^t, \tag{65}$$

and then perform a reverse sweep:

$$u^{t+2\Delta t} = L_x L_y L_z u^{t+\Delta t}, \tag{66}$$

using the same time step Δt to obtain second-order accuracy. We will refer to the combination of the forward and reverse sweeps as a double sweep.

A more symmetrical sweeping pattern can be used by permutating the sweeping order when completing the next double time step. The dimensional splitting or operator splitting technique can be summarized as follows:

$$u^{t_2} = u^{t_1+2\Delta t_1} = L_x L_y L_z L_z L_y L_x u^{t_1}, \tag{67}$$

$$u^{t_3} = u^{t_2+2\Delta t_2} = L_z L_x L_y L_y L_x L_z u^{t_2}, \tag{68}$$

$$u^{t_4} = u^{t_3+2\Delta t_3} = L_y L_z L_x L_x L_z L_y u^{t_3}, \tag{69}$$

where Δt_1, Δt_2, and Δt_3 are newly determined time steps after completing each double sweep.

The CFL condition for the three-dimensional relaxing TVD scheme is similarly given by equation (58), but with

$$c_{max} = \max[(c_x)_{max}, (c_y)_{max}, (c_z)_{max}]. \tag{70}$$

where $c_i = |v_i| + c_s$. Note that since $\max(|v_i|)$ is on average a factor of $\sqrt{3}$ smaller than $\max(|v|)$, a dimensionally split scheme can use a longer time step compared to an unsplit scheme.

The dimensional splitting technique also has other advantages. The decomposition into a one-dimensional problem allows one to write short one-dimensional algorithms, which are easy to optimize to be cache efficient. A three-dimensional hydro code is straightforward to implement in parallel. When sweeping in the x direction, for example, one can break up the data into one-dimensional columns and operate on the independent columns in parallel. A sample three-dimensional relaxing TVD code, implemented in parallel using OpenMP directives, is provided in the Appendix.

6. SEDOV-TAYLOR BLAST-WAVE TEST FOR THREE-DIMENSIONAL HYDRO

A rigorous and challenging test for any three-dimensional Eulerian or Lagrangian hydrodynamic code is the Sedov-Taylor blast-wave test. We set up the simulation box with a homogeneous medium of density ρ_1 and negligible pressure and introduce a pointlike supply of thermal energy E_0 in the center of the box at time $t = 0$. The challenge is to accurately capture the strong spherical shock that sweeps along material as it propagates out into the ambient medium. The Sedov-Taylor test is used to model nuclear-type explosions. In astrophysics,
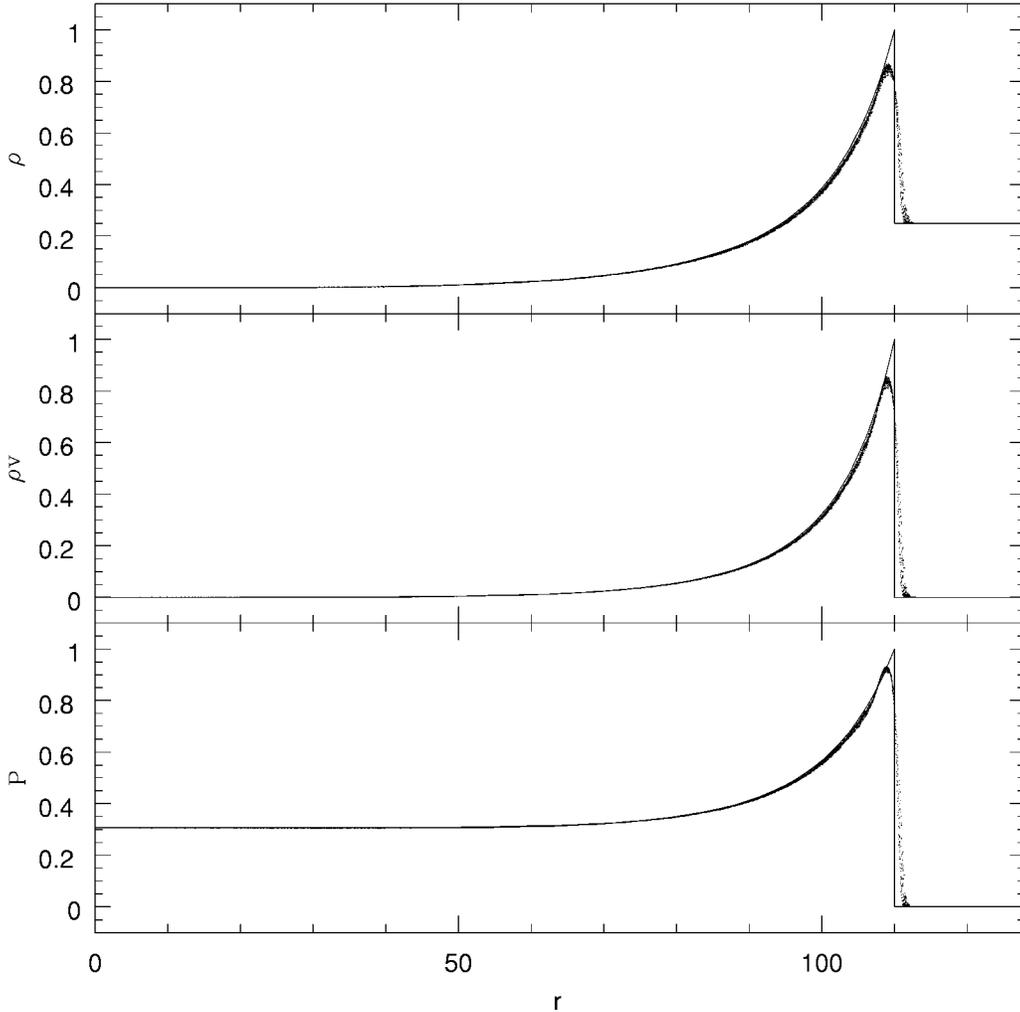
Fig. 8.—Sedov-Taylor blast-wave test conducted in a box with 256^3 cells. The data points are taken from a random subset of cells, and the solid lines are the analytical self-similar solutions.

it is often used as a basic setup to model supernova explosions and the evolution of supernova remnants (see Shu 1992).

The analytical Sedov solution uses the self-similar nature of the blast-wave expansion (see Landau & Lifshitz 1987). Consider a frame fixed relative to the center of the explosion. The spherical shock front propagates outward, and the distance from the origin is given by

$$r_{sh}(t) = \xi_0 \left(\frac{E_0 t^2}{\rho_1}\right)^{1/5},$$ (71)

where $\xi_0 = 1.15$ for an ideal gas with $\gamma = 5/3$. The velocity of the shock $v_{sh} = \partial r_{sh}/\partial t$ is given by

$$v_{sh}(t) = \frac{2}{5}\frac{r_{sh}(t)}{t}.$$ (72)

Since the ambient medium has negligible pressure, the shocks will be very strong. The density ρ_2, velocity v_2, and pressure P_2 directly behind the shock front are

$$\rho_2 = \left(\frac{\gamma + 1}{\gamma - 1}\right)\rho_1,$$ (73)

$$v_2 = \left(\frac{2}{\gamma + 1}\right)v_{sh},$$ (74)

$$P_2 = \left(\frac{2}{\gamma + 1}\right)\rho_1 v_{sh}^2.$$ (75)

The immediate postshock gas density is constant in time, while the shocked gas velocity v_2 and pressure P_2 decrease as $t^{-3/5}$ and $t^{-6/5}$, respectively. The full analytical Sedov-Taylor solutions can be found in Landau & Lifshitz (1987).
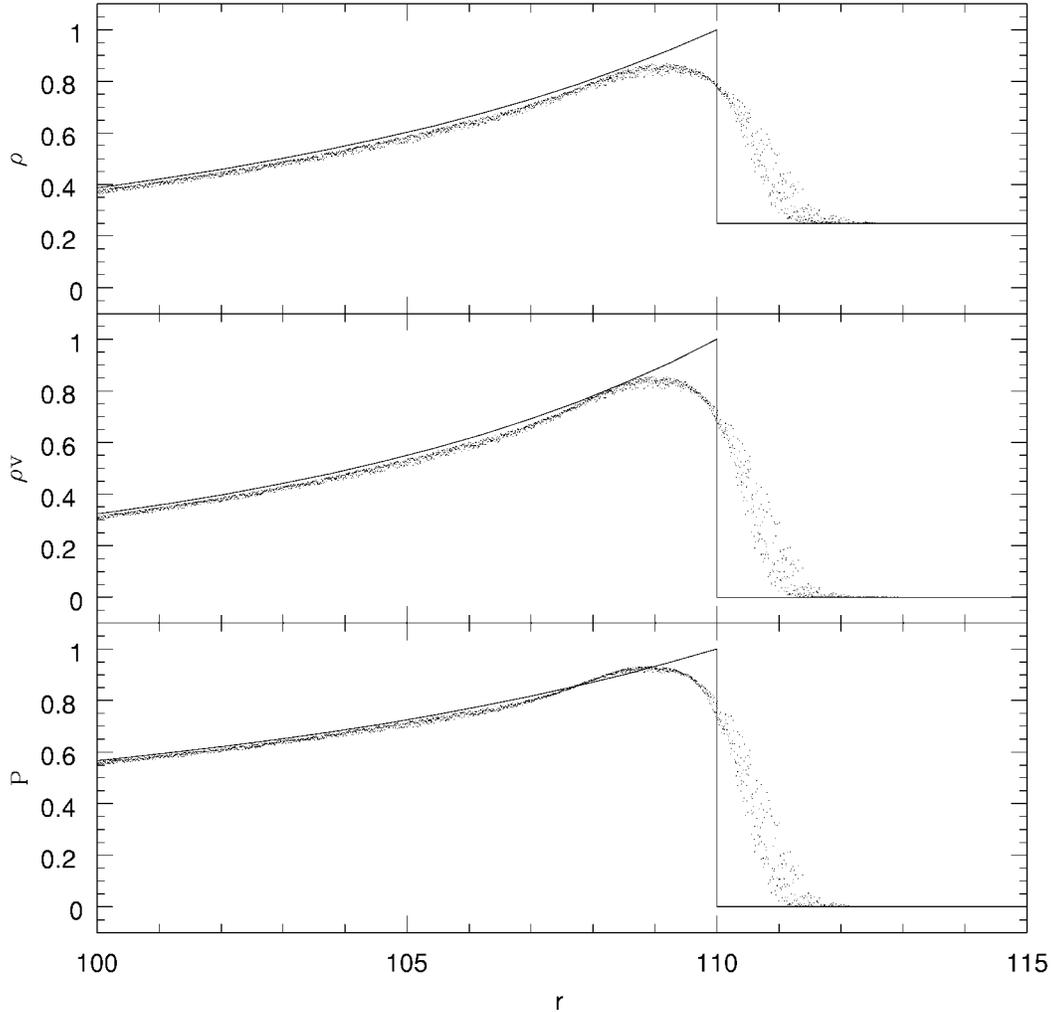
Fig. 9.—Closeup of the Sedov-Taylor blast wave. The resolution of the shock front is roughly two grid cells, and the anisotropic scatter is less than one grid cell.

The three-dimensional relaxing TVD code based on the Van Leer flux limiter is applied to capturing the Sedov-Taylor blast wave. We set up a box with 256^3 cells and constant initial density $\rho_1 = 1$. At time $t = 0$, we inject a supply of thermal energy $E_0 = 10^5$ into one cell at the center of the box. The simulation is stopped at time $t = 283$, in which the shock front has propagated out to a distance of $r_{sh} = 110$ cells from the center. In Figures 8 and 9, we plot the radial distributions of density, momentum, and pressure, normalized to ρ_2, $\rho_2 v_2$, and P_2, respectively. The data points are taken from a random subset of cells, and the solid lines are the analytical Sedov-Taylor solutions. Despite solving a spherically symmetric problem on an explicitly nonrotationally invariant Cartesian grid, the anisotropic scatter in the results is small. The distance of the shock front from the center of the explosion as a function of time is indeed given by equation (71), demonstrating that the

three-dimensional relaxing TVD code ensures the correct shock propagation. The resolution of the shock front is roughly two grid cells. The numerical shock jump values of ρ_2, v_2, and P_2 are resolution dependent and come close to the theoretical values for our test with 256^3 cells. We leave it as an exercise for the reader to test the code using the minmod and superbee flux limiters.

7. SELF-GRAVITATING HYDRO FOR ASTROPHYSICAL APPLICATIONS

For astrophysical applications, both hydrodynamical and gravitational forces are included. The gravitational forces arise from the self-gravity of the fluid and can also come from an external field. The Euler equations with the gravitational source
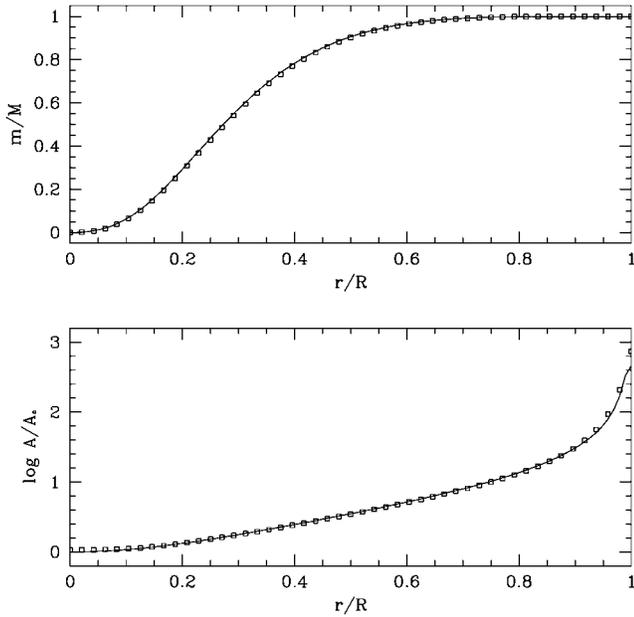
FIG. 10.—Advection of a self-gravitating polytrope in a periodic box with 256^3 cells. We compare the mass and entropy profiles of the initial (*solid line*) and advected polytrope after 1000 time steps in which the polytrope has moved 256 cells in each direction.

terms included are given as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho v_j) = 0, \tag{76}$$

$$\frac{\partial(\rho v_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho v_i v_j + P\delta_{ij}) = -\rho \frac{\partial \phi}{\partial x_i}, \tag{77}$$

$$\frac{\partial e}{\partial t} + \frac{\partial}{\partial x_j}[(e + P)v_j] = -\rho v_i \frac{\partial \phi}{\partial x_i}, \tag{78}$$

where ϕ is the gravitational potential. Poisson's equation

$$\nabla^2 \phi = 4\pi G \rho \tag{79}$$

relates the gravitational potential to the density field. The general solution can be written as

$$\phi(\boldsymbol{x}) = \int \rho(\boldsymbol{x}')w(\boldsymbol{x} - \boldsymbol{x}')d^3x', \tag{80}$$

where the kernel is given by

$$w(\boldsymbol{x}) = -\frac{G}{|\boldsymbol{x}|}. \tag{81}$$

In the discrete case, the integral in equation (80) becomes a

sum and Poisson's equation can be solved using fast Fourier transforms (FFTs) to do the convolution. The forces are then calculated by finite differencing the potential (see Hockney & Eastwood 1988).

The addition of gravitational source terms in the Euler equations is easily handled using the operator splitting technique described in § 5.3. Consider the double sweep:

$$u^{t+2\Delta t} = L_x L_y L_z GGL_z L_y L_x u^t, \tag{82}$$

where the operator L_i represents the updating of u by including the flux in the i direction and the operator G represents the gravitational acceleration of the fluid. During the gravitational step, the flux terms in the Euler equations are ignored. The density distribution does not change and only the fluid momenta and total energy density are updated.

7.1. Astrophysical Formation of Blue Stragglers through Stellar Collisions

The stellar density in the cores of globular and open clusters is high enough for stellar collisions to take place with significant frequency (Hills & Day 1976). Current observations and simulations suggest that the merger of two main-sequence stars produces a blue straggler (Sills et al. 1997; Sandquist, Bolte, & Hernquist 1997). The blue stragglers are outlying main-sequence stars that lie beyond the main-sequence turnoff in the color-magnitude diagram (CMD) of a star cluster. The blue stragglers are more massive, brighter, and bluer than the turnoff stars. Since more massive stars evolve faster than lower mass stars and are not expected to lie beyond the turnoff, this suggests that blue stragglers must have formed more recently.

In principle, the merger of two main-sequence stars can produce a young remnant star provided that significant mixing occurs in the process. The mixing produces a higher hydrogen fraction in the core of the remnant than that of the parent stars, which have already burned most of the hydrogen to helium in their cores. Benz & Hills (1987) used low-resolution SPH simulations with $\sim 10^3$ particles to simulate the merging of $n = 3/2$ polytropes and found that they fully mixed. However, medium-resolution SPH simulations with $\sim 10^4$ particles of $n = 3/2$ or $n = 3$ polytropes showed only weak mixing (Lombardi, Rasio, & Shapiro 1996; Sandquist et al. 1997). It is worth noting that $n = 3/2$ polytropes are more representative of low-mass main-sequence stars with large convective envelopes, while $n = 3$ polytropes resemble main-sequence stars near the turnoff that have little mass in their convective envelopes. High-resolution SPH simulations involving $\sim 10^5$–10^6 particles have now been applied to simulating stellar collisions (Sills et al. 2002).

The merging stars process is mostly subsonic, and strong shocks are not expected. In the absence of shocks, SPH particles will follow flow lines of constant entropy as a result of the Lagrangian nature of the method. As a result, the particles may
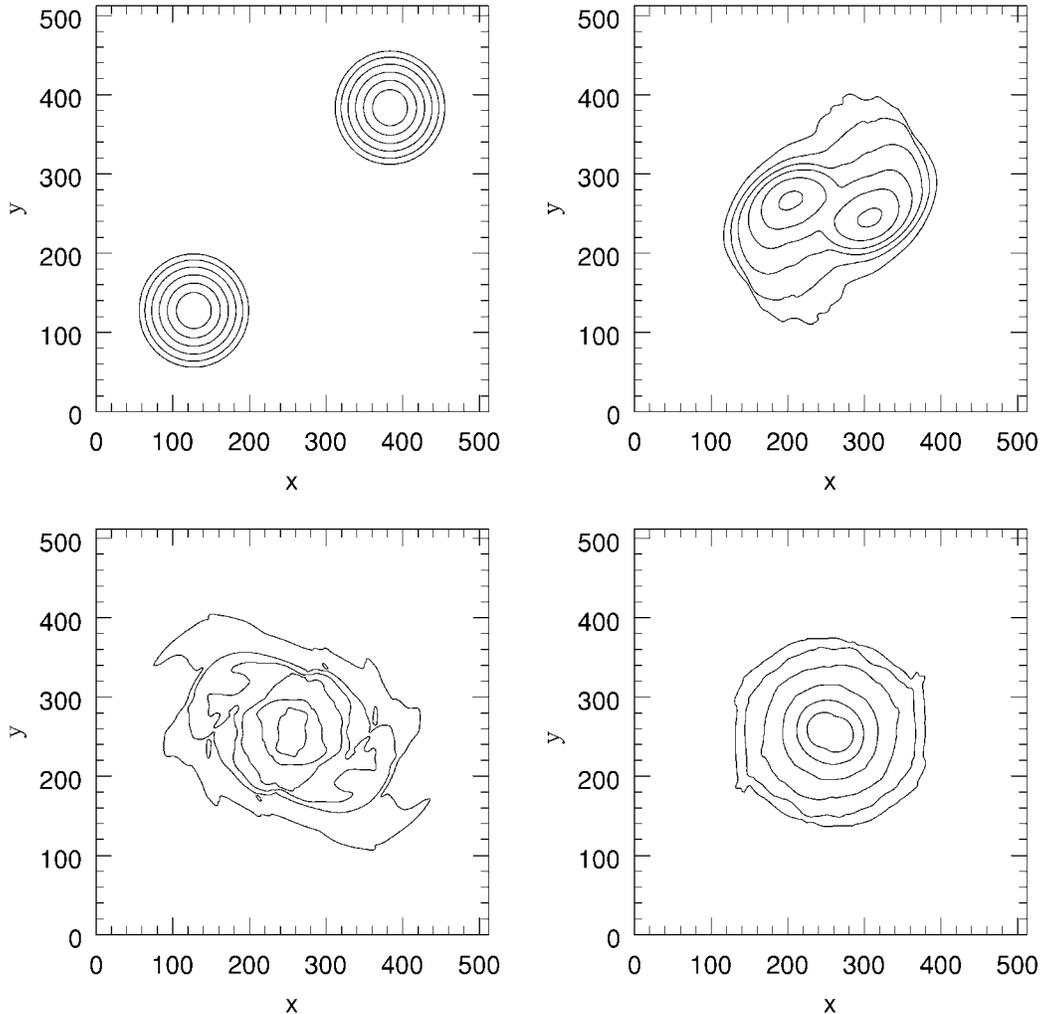
FIG. 11.—Snapshots of the merging process taken at time $t = 0$, 2, 4, and $8\tau_{\mathrm{dyn}}$. The density contours are spaced logarithmically with two per decade and covering three decades down from the maximum.

experience sedimentation. In addition, the mixing can also depend on the adopted smoothing length and the form of artificial viscosity. For an SPH fluid, the Reynolds number is of the order of $(N_p/N_s)^{1/3}$, where N_p is the total number of particles and N_s is the number of particles over which the smoothing is done. For $N_p \sim 10^5$ and $N_s \sim 10^2$, the Reynolds number is ~10. However, a fluid with a low Reynolds number will tend to experience laminar flow. Hence, SPH may undermix.

It is a worthwhile exercise to model the merging process using Eulerian hydrodynamical simulations. The differences between Eulerian and Lagrangian approaches may lead to very different results on mixing. At present, no such work has been reported in the literature.

7.2. Numerical Method

We consider the off-axis collision of two main-sequence stars with $M = 0.8\,M_\odot$ and $R = 0.955\,R_\odot$, which are modeled using

$n = 3$ polytropes. A polytrope with polytropic index n has equilibrium density and pressure profiles, which are related by

$$P \propto \rho^{1+1/n}. \tag{83}$$

The density profile is determined by solving the Lane-Emden equation (see Chandrasekhar 1957). We adopt an ideal gas equation of state with adiabatic index $\gamma = 5/3$. Note that for an $n = 3$ polytrope, 90% of the total mass is contained within $r \lesssim 0.5R$. We define the dynamical time to be

$$\tau_{\mathrm{dyn}} \equiv \frac{1}{\sqrt{G\bar{\rho}}}, \tag{84}$$

where $\bar{\rho}$ is the average density. For the chosen parent stars, the dynamical time is approximately 1 physical hour.

The collision is simulated in a box with $512 \times 512 \times 256$
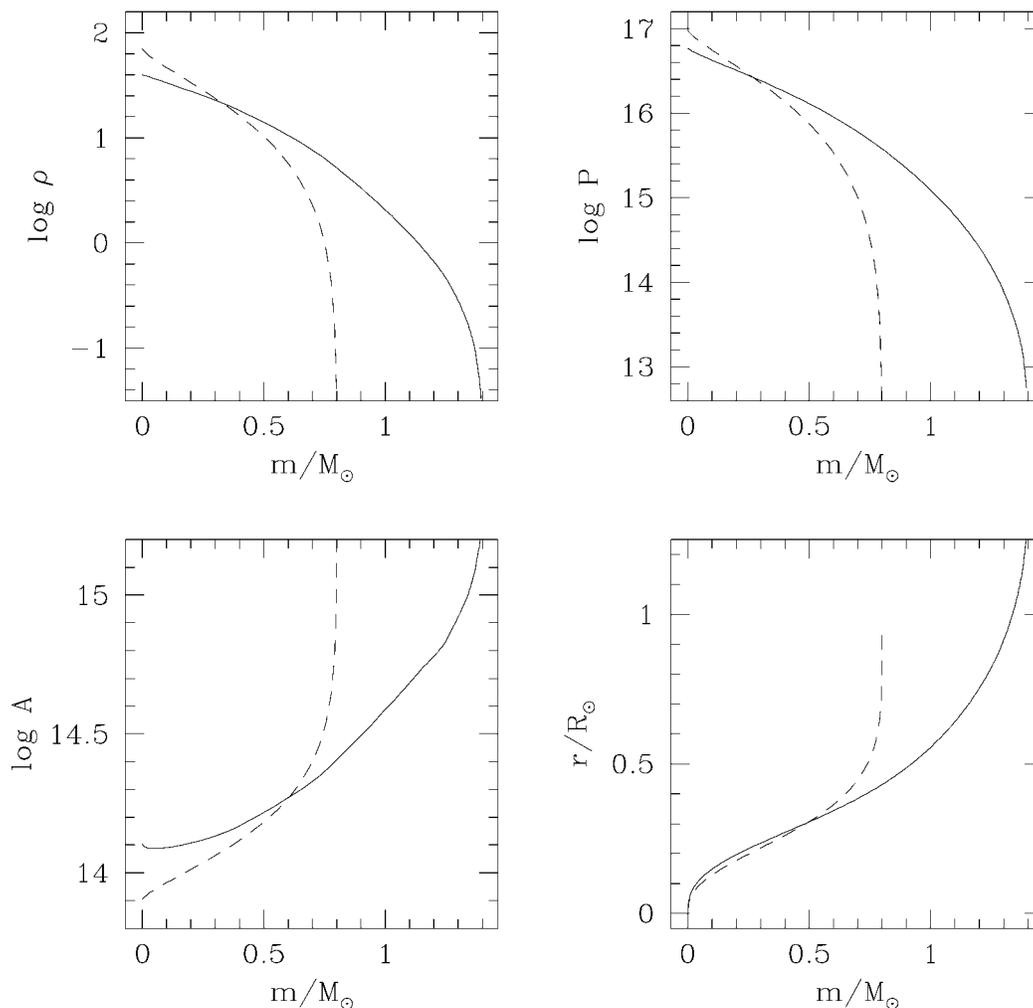
FIG. 12.—Thermodynamic profiles of the merger remnant (*solid line*) and the parent stars (*dashed line*). Units are cgs. The radial plot is included for comparison.

cells, and the orbital plane coincides with the *x-y* plane. Initially, each parent star has a radius of 96 grid cells. The stars are set up on zero-energy parabolic orbits with a pericenter separation equal to $0.25R$. The initial trajectories are calculated assuming point masses. In a Eulerian simulation, the vacuum cannot have zero density. We set the minimum density of the cells to be 10^{-8} of the central density of the parent stars.

7.3. Numerical Results

A nontrivial test of a self-gravitating Eulerian hydro code is the advection of an object in hydrostatic equilibrium. The challenge is to maintain the equilibrium profile over a large number of time steps. One of the parent stars is placed in a periodic box with 256^3 cells and given some initial momentum. We make the test rigorous by having the polytrope move in all three directions. In Figure 10, we compare the mass and entropy profiles of the initial and advected polytrope. The entropic variable $A \equiv P/\rho^\gamma$ is used in place of the specific entropy. The

parameter A_0 is defined to be the minimum entropy of the parent polytrope. After 1000 time steps in which the polytrope has moved 256 cells in each direction, the advected polytrope has still retained its equilibrium profile. Shock heating can occur in the outer envelope as the polytrope moves through the false vacuum. However, by setting the density of the false vacuum to be 10^{-8} of the central density of the polytrope, we can minimize the spurious shock heating.

In Figure 11, we show four snapshots of the merging process taken at time $t = 0, 2, 4,$ and $8\tau_{\mathrm{dyn}}$. The two-dimensional density maps are created by averaging over four planes taken about the orbital midplane. The density contours are spaced logarithmically with two per decade and covering three decades down from the maximum. The parent stars are initially separated by $3.75R$ and placed on zero-energy orbits with a pericenter separation of $0.25R$. During the collision process, the outer envelopes of the parent stars are shock-heated and material gets ejected. In less than $10\tau_{\mathrm{dyn}}$, the merger remnant

establishes hydrostatic equilibrium. The merger remnant is a rotating oblate with mass approximately 90% of the combined mass of the parent stars.

In Figure 12, we plot the thermodynamic profiles of the merger remnant and the parent stars. The central density and pressure in the core of the merger remnant is lower than the corresponding values in the parent stars by approximately half. The entropy floor has risen by a factor of 1.6. Shock heating is expected to be minimal in the core, so a change in entropy suggests that some mixing has taken place. However, it is difficult to quantify the amount of mixing from examining the thermodynamic profiles alone.

7.4. Future Work

To help address the question of mixing, we are implementing a particle-mesh (PM) scheme where test particles can be used to track passively advected quantities such as chemical composition. Initially, each parent star is assigned a large number of particles with known chemical composition. The test particles are passively advected along velocity field lines. For each time step, the velocity of each particle is interpolated from the grid using a "cloud-in-cell" (CIC) scheme (Hockney & Eastwood 1988), and the equations of motions are solved using second-order Runge-Kutta integration. The CIC interpolation scheme is also used to determine the local density, pressure, and entropy associated with each particle. With this setup, we have the benefit of being able to track thermodynamic quantities like in an SPH scheme but avoid the undermixing problem since the fluid equations are solved using the Eulerian scheme.

Future work (H. Trac, A. Sills, & U.-L. Pen 2003, in preparation) will have higher resolution simulations. Collisions will be simulated in a box with $1024 \times 1024 \times 512$ cells. Each parent star will have a radius of 192 grid cells and be assigned 256^3 test particles.

The self-gravitating hydro code used for the simulations is very memory friendly. For the $1024^2 \times 512$ grid, 10 GBytes is required to store the hydro variables, 2 GBytes for the potential, and less than 1 GByte for the test particles. For every double time step, approximately 1000 floating point operations per grid cell are needed to carry out the TVD hydro calculations. The potential is computed once for every double step and this requires two FFTs. Since Eulerian codes are very memory friendly, have low floating point counts, are easily parallelized, and scale very well on shared-memory, multiple-processor machines, they can be used to run very high resolution simulations.

8. SUMMARY

We have presented several numerical schemes for solving the linear advection equation and given the CFL stability conditions for each scheme. We have implemented the relaxing TVD scheme to solve the Euler system of conservation laws. The second-order accurate TVD scheme provides high-resolution capturing of shocks, as can be seen in the Riemann shock test and the Sedov-Taylor blast-wave test. The one-dimensional relaxing TVD scheme can be easily generalized to higher dimensions using the dimensional splitting technique. A dimensionally split scheme can use longer time steps and is straightforward to implement in parallel. We have presented a sample astrophysical application. A three-dimensional self-gravitating Eulerian hydro code is used to simulate the formation of blue straggler stars through stellar mergers. We hope to have convinced the reader that Eulerian computational fluid dynamics is a powerful approach to simulating complex fluid flows because it is simple, fast, and accurate.

APPENDIX

THREE-DIMENSIONAL RELAXING TVD CODE

We provide a sample three-dimensional relaxing TVD code written in Fortran 90. The code is implemented using OpenMP directives to run in parallel on shared memory machines. The code is fast and memory friendly. The array `u(a,i,j,k)` stores the five conserved hydro quantities $a = (\rho, \rho vx, \rho vy, \rho vz, e)$ for each cell (i,j,k) in the Cartesian cubical lattice with side length `nc`. For each sweep, we first call the subroutine `timestep` to determine the appropriate time step `dt` that satisfies the CFL condition. The updating of `u` by including the flux in the `x` direction is performed by the `sweepx` subroutine. The data array `u` is divided into one-dimensional array sections `u1d(a,i)`, which are operated on by the `relaxing` subroutine. The independent columns are distributed amongst multiple processors on a shared memory machine by the OpenMP directives.

The relaxing TVD subroutine in this sample code is written for ease of readability and therefore is not fully optimized. At the beginning of each partial step in the Runge-Kutta time integration scheme, the cell-averaged variables defined at grid cell centers are calculated by the `averageflux` subroutine. The fluxes at cell boundaries for the right-moving and left-moving waves are stored in `fr` and `fl`, respectively. We have implemented the minmod, superbee, and Van Leer flux limiters, and the user of the code can easily switch between them.

We have provided some initial conditions for the Sedov-Taylor blast-wave test. The reader is encouraged to test the code and compare how the various flux limiters do at resolving strong shocks. This sample code does not implement the modified relaxing TVD scheme described at the end of § 5.2, which has been found to work very well with the Van Leer flux limiter but unstable with superbee for the three-dimensional Sedov Taylor test. We have found that the superbee limiter is often unstable for three-dimensional fluid simulations. Please contact the authors regarding any questions on the implementation of the relaxing TVD algorithm.

```fortran
program main
    implicit none
    integer, parameter :: nc = 64,hc = nc/2
    real, parameter :: gamma = 5./3,cfl = 0.9

    real, dimension(5,nc,nc,nc) :: u

    integer nsw,stopsim
    real t,tf,dt,E0,rmax

    t = 0
    dt = 0
    nsw = 0
    stopsim = 0

    E0 = 1e5
    rmax = 3*hc/4
    tf = sqrt((rmax/1.15)**5/E0)
    call sedovtaylor
    do
        call timestep
        call sweepx
        call sweepy
        call sweepz
        call sweepz
        call sweepy
        call sweepx
        if (stopsim .eq. 1) exit
        call timestep
        call sweepz
        call sweepx
        call sweepy
        call sweepy
        call sweepx
        call sweepz
        if (stopsim .eq. 1) exit
        call timestep
        call sweepy
        call sweepz
        call sweepx
        call sweepx
        call sweepz
```

```fortran
        call sweepy
        if (stopsim .eq. 1) exit
    enddo
    call outputresults

contains

    subroutine sedovtaylor
        implicit none
        integer i,j,k

        do k = 1,nc
            do j = 1,nc
                do i = 1,nc
                    u(1,i,j,k) = 1
                    u(2:4,i,j,k) = 0
                    u(5,i,j,k) = 1e-3
                enddo
            enddo
        enddo
        u(5,hc,hc,hc) = E0
        return
    end subroutine sedovtaylor

    subroutine outputresults
        implicit none
        integer i,j,k
        real r,x,y,z

        open(1,file = 'sedovtaylor.dat',recl = 200)
        do k = 1,nc
            z = k-hc
            do j = 1,nc
                y = j-hc
                do i = 1,nc
                    x = i-hc
                    r = sqrt(x**2+y**2+z**2)
                    write(1,*) r,u(:,i,j,k)
                enddo
            enddo
        enddo
        close(1)
        return
    end subroutine outputresults

    subroutine timestep
        implicit none
        integer i,j,k
        real P,cs,cmax
        real v(3)

        cmax = 1e-5
        !$omp parallel do default(shared) private(i,j,k,v,cs,P)
            reduction(max:cmax)
```

```fortran
      do k=1,nc
        do j=1,nc
          do i=1,nc
            v=abs(u(2:4,i,j,k)/u(1,i,j,k))
            P=max((gamma-1)*(u(5,i,j,k)-u(1,i,j,k)
                *sum(v**2)/2),0.)
            cs=sqrt(gamma*P/u(1,i,j,k))
            cmax=max(cmax,maxval(v)+cs)
          enddo
        enddo
      enddo
      !$omp end parallel do

      dt=cfl/cmax
      if (t+2*dt .gt. tf) then
        dt=(tf-t)/2
        stopsim=1
      endif
      t=t+2*dt
      nsw=nsw+1
      write(*,"(a7,i3,a8,f7.5,a6,f8.5)") 'nsw = ',nsw,'
          dt = ',dt,' t = ',t
      return
    end subroutine timestep

    subroutine sweepx
      implicit none
      integer j,k
      real u1d(5,nc)

      !$omp parallel do default(shared) private(j,k,u1d)
      do k=1,nc
        do j=1,nc
          u1d=u(:,:,j,k)
          call relaxing(u1d)
          u(:,:,j,k)=u1d
        enddo
      enddo
      !$omp end parallel do
      return
    end subroutine sweepx

    subroutine sweepy
      implicit none
      integer i,k
      real u1d(5,nc)

      !$omp parallel do default(shared) private(i,k,u1d)
      do k=1,nc
        do i=1,nc
          u1d((/1,3,2,4,5/),:)=u( :,i,:,k)
          call relaxing(u1d)
          u(:,i,:,k)=u1d((/1,3,2,4,5/),:)
        enddo
```

```fortran
      enddo
      !$omp end parallel do
      return
    end subroutine sweepy

    subroutine sweepz
      implicit none
      integer i,j
      real u1d(5,nc)

      !$omp parallel do default(shared) private(i,j,u1d)
      do j=1,nc
        do i=1,nc
          u1d((/1,4,3,2,5/),:)=u(:,i,j,:)
          call relaxing(u1d)
          u(:,i,j,:)=u1d((/1,4,3,2,5/),:)
        enddo
      enddo
      !$omp end parallel do
      return
    end subroutine sweepz

    subroutine relaxing(u)
      implicit none
      real, dimension(nc) :: c
      real, dimension(5,nc) :: u,u1,w,fu,fr,fl,dfl,dfr

      !! Do half step using first-order upwind scheme
      call averageflux(u,w,c)
      fr=(u*spread(c,1,5)+w)/2
      fl=cshift(u*spread(c,1,5)-w,1,2)/2
      fu=(fr-fl)
      u1=u-(fu-cshift(fu,-1,2))*dt/2

      !! Do full step using second-order TVD scheme
      call averageflux(u1,w,c)

      !! Right-moving waves
      fr=(u1*spread(c,1,5)+w)/2
      dfl=(fr-cshift(fr,-1,2))/2
      dfr=cshift(dfl,1,2)
      call vanleer(fr,dfl,dfr)
      !call minmod(fr,dfl,dfr)
      !call superbee(fr,dfl,dfr)

      !! Left-moving waves
      fl=cshift(u1*spread(c,1,5)-w,1,2)/2
      dfl=(cshift(fl,-1,2)-fl)/2
      dfr=cshift(dfl,1,2)
      call vanleer(fl,dfl,dfr)
      !call minmod(fl,dfl,dfr)
      !call superbee(fl,dfl,dfr)

      fu=(fr-fl)
```

```fortran
      u = u-(fu-cshift(fu,-1,2))*dt
      return
      end subroutine relaxing


      subroutine averageflux(u,w,c)
         implicit none
         integer i
         real P,v
         real u(5,nc),w(5,nc),c(nc)

         !! Calculate cell-centered fluxes and freezing speed
         do i = 1,nc
            v = u(2,i)/u(1,i)
            P = max((gamma-1)*(u(5,i)-
               sum(u(2:4,i)**2)/u(1,i)/2),0.)
            c(i) = abs(v)+max(sqrt(gamma*P/u(1,i)),1e-5)
            w(1,i) = u(1,i)*v
            w(2,i) = (u(2,i)*v+P)
            w(3,i) = u(3,i)*v
            w(4,i) = u(4,i)*v
            w(5,i) = (u(5,i)+P)*v
         enddo
         return
      end subroutine averageflux


      subroutine vanleer(f,a,b)
         implicit none
         real, dimension(5,nc) :: f,a,b,c

         c = a*b
```

```fortran
         where (c .gt. 0)
            f = f+2*c/(a+b)
         endwhere
         return
      end subroutine vanleer


      subroutine minmod(f,a,b)
         implicit none
         real, dimension(nc) :: f,a,b

         f = f+(sign(1.,a)+sign(1.,b))*min(abs(a),abs(b))/2.
         return
      end subroutine minmod


      subroutine superbee(f,a,b)
         implicit none
         real, dimension(5,nc) :: f,a,b

         where (abs(a) .gt. abs(b))
            f = f+(sign(1.,a)+sign(1.,b))*min(abs(a),
               abs(2*b))/2.
         elsewhere
            f = f+(sign(1.,a)+sign(1.,b))*min(abs(2*a),
               abs(b))/2.
         endwhere
         return
      end subroutine superbee


      end program main
```

REFERENCES

Benz, W., & Hills, J. G. 1987, ApJ, 323, 614

Chandrasekhar, S. 1957, An Introduction to the Study of Stellar Structure (New York: Dover)

Courant, R., Isaacson, E., & Reeves, M. 1952, Commun. Pure Appl. Math., 5, 243

Gingold, R. A., & Monaghan, J. J. 1977, MNRAS, 181, 375

Godunov, S. K. 1959, Mat. Sb., 47, 271

Harten, A. 1983, J. Comput. Phys., 49, 357

Hills, J. G., & Day, C. A. 1976, Astrophys. Lett., 17, 87

Hirsch, C. 1990, Numerical Computation of Internal and External Flows, Vol. 2, Computational Methods for Inviscid and Viscous Flows (New York: Wiley)

Hockney, R. W., & Eastwood, J. W. 1988, Computer Simulation Using Particles (Philadelphia: IOP)

Jin, S., & Xin, Z. 1995, Commun. Pure Appl. Math., 48, 235

Landau, L. D., & Lifshitz, E. M. 1987, Fluid Mechanics (2d ed.; Oxford: Pergamon)

Laney, C. B. 1998, Computational Gas Dyanamics (Cambridge: Cambridge Univ. Press)

Lax, P. D., & Wendroff, B. 1960, Commun. Pure Appl. Math., 10, 537

Lombardi, J. C., Jr., Rasio, F. A., & Shapiro, S. L. 1996, ApJ, 468, 797

Lucy, L. B. 1977, AJ, 82, 1013

Pen, U. 1998, ApJS, 115, 19

Roe, P. L. 1985, in Large-Scale Computations in Fluid Mechanics, Lectures in Applied Mathematics, ed. B. E. Engquist, S. Osher, & R. C. J. Somerville (Providence: Am. Math. Soc.)

Sandquist, E. L., Bolte, M., & Hernquist, L. 1997, ApJ, 477, 335

Shu, F. H. 1992, Gas Dynamics (Mill Valley: University Science Books)

Sills, A., Adams, T., Davies, M. B., & Bate, M. R. 2002, MNRAS, 332, 49

Sills, A., Lombardi, J. C., Jr., Bailyn, C. D., Demarque, P. D., Rasio, F. A., & Shapiro, S. L. 1997, ApJ, 487, 290

Strang, G. 1968, SIAM J. Numer. Anal., 5, 506

Van Leer, B. 1974, J. Comput. Phys., 14, 361