The Astrophysical Journal, 715:1221–1238, 2010 June 1 doi:10.1088/0004-637X/715/2/1221

CASTRO: A NEW COMPRESSIBLE ASTROPHYSICAL SOLVER. I. HYDRODYNAMICS AND SELF-GRAVITY

A. S. Almgren[1], V. E. Beckner[1], J. B. Bell[1], M. S. Day[1], L. H. Howell[2], C. C. Joggerst[3,4], M. J. Lijewski[1], A. Nonaka[1], M. Singer[2], and M. Zingale[5]
[1] Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[2] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94550, USA
[3] Department of Astronomy & Astrophysics, The University of California, Santa Cruz, CA 95064, USA
[4] Los Alamos National Laboratory, Los Alamos, CA 87545, USA
[5] Department of Physics & Astronomy, Stony Brook University, Stony Brook, NY 11794-3800, USA
Received 2010 February 1; accepted 2010 April 2; published 2010 May 7

ABSTRACT

We present a new code, CASTRO, that solves the multicomponent compressible hydrodynamic equations for astrophysical flows including self-gravity, nuclear reactions, and radiation. CASTRO uses an Eulerian grid and incorporates adaptive mesh refinement (AMR). Our approach to AMR uses a nested hierarchy of logically rectangular grids with simultaneous refinement in both space and time. The radiation component of CASTRO will be described in detail in the next paper, Part II, of this series.

Key words: equation of state – gravitation – hydrodynamics – methods: numerical – nuclear reactions, nucleosynthesis, abundances

Online-only material: color figures

1. INTRODUCTION

In this paper, Part I of a two-part series, we present a new code, CASTRO, that solves the multicomponent compressible hydrodynamic equations with a general equation of state (EOS) for astrophysical flows. Additional physics include self-gravity, nuclear reactions, and radiation. CASTRO uses an Eulerian grid and incorporates adaptive mesh refinement (AMR). Our approach to AMR uses a nested hierarchy of logically rectangular grids with simultaneous refinement of the grids in both space and time. Spherical (in one dimension), cylindrical (in one dimension or two dimensions), and Cartesian (in one dimension, two dimensions, or three dimensions) coordinate systems are supported. The radiation component of CASTRO will be described in detail in the next paper, Part II, of this series.

There are a number of other adaptive mesh codes for compressible astrophysical flows, most notably, ENZO (O'Shea et al. 2005), FLASH (Fryxell et al. 2000), and RAGE (Gittings et al. 2008). CASTRO differs from these codes in several ways. CASTRO uses an unsplit version of the piecewise parabolic method, PPM, with new limiters that avoid reducing the accuracy of the scheme at smooth extrema; the other codes are based on operator-split hydrodynamics, though the most recent release of FLASH, version 3.2, includes an unsplit MUSCL-Hancock scheme. The different methodologies also vary in their approach to AMR. RAGE uses a cell-by-cell refinement strategy while the other codes use patch-based refinement. FLASH uses equal size patches whereas ENZO and CASTRO allow arbitrary sized patches. ENZO and FLASH enforce a strict parent–child relationship between patches; i.e., each refined patch is fully contained within a single parent patch; CASTRO requires only that the union of fine patches be contained within the union of coarser patches with a suitable proper nesting. Additionally, FLASH and RAGE use a single time step across all levels while CASTRO and ENZO support subcycling in time. All four codes include support for calculation of self-gravity.

It is worth noting that CASTRO uses the same grid structure as the low Mach number astrophysics code, MAESTRO (see, e.g., Nonaka et al. 2010). This will enable us to map the results from a low Mach number simulation, such as that of the convective period and ignition of a Type Ia supernova, to the initial conditions for a compressible simulation such as that of the explosion itself, thus taking advantage of the accuracy and efficiency of each approach as appropriate.

2. HYDRODYNAMICS

In CASTRO, we evolve the fully compressible equations forward in time. The equations expressing conservation of mass, momentum, and total energy are

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{u}) + S_{\mathrm{ext},\rho}, \tag{1}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} = -\nabla \cdot (\rho \mathbf{u}\mathbf{u}) - \nabla p + \rho \mathbf{g} + \mathbf{S}_{\mathrm{ext},\rho\mathbf{u}}, \tag{2}$$

$$\frac{\partial (\rho E)}{\partial t} = -\nabla \cdot (\rho \mathbf{u} E + p\mathbf{u}) + \rho H_{\mathrm{nuc}} + \rho \mathbf{u} \cdot \mathbf{g} + S_{\mathrm{ext},\rho E}. \tag{3}$$

Here ρ, $\mathbf{u}$, and E are the mass density, velocity vector, and total energy per unit mass, respectively. The total energy $E = e + \mathbf{u} \cdot \mathbf{u}/2$, where e is the specific internal energy. The pressure, p, is defined by a user-supplied EOS, and $\mathbf{g}$ is the gravitational acceleration vector. The source terms, $S_{\mathrm{ext},\rho}$, $S_{\mathrm{ext},\rho\mathbf{u}}$, and $S_{\mathrm{ext},\rho E}$ are user-specified external source terms for the mass, momentum, and energy equations, respectively. For reacting flows, we evolve equations for mass fractions, X_k:

$$\frac{\partial (\rho X_k)}{\partial t} = -\nabla \cdot (\rho \mathbf{u} X_k) + \rho \dot{\omega}_k + S_{\mathrm{ext},\rho X_k}, \tag{4}$$

where the production rates, $\dot{\omega}_k$, for species k are defined by a user-supplied reaction network. The reaction network also determines the energy generation rate H_{nuc}. The mass fractions are subject to the constraint that $\sum_k X_k = 1$. Again, a user-specified external source, $S_{\mathrm{ext},\rho X_k}$, may be specified. Finally, CASTRO includes passively advected quantities, C_k^{adv}, and

auxiliary variables, $C_k^{\rm aux}$ that satisfy

$$\frac{\partial\left(\rho C_k^{\rm adv}\right)}{\partial t} = -\nabla\cdot\left(\rho\mathbf{u}C_k^{\rm adv}\right) + S_{{\rm ext},\rho C_k^{\rm adv}}, \quad (5)$$

$$\frac{\partial\left(\rho C_k^{\rm aux}\right)}{\partial t} = -\nabla\cdot\left(\rho\mathbf{u}C_k^{\rm aux}\right) + S_{{\rm ext},\rho C_k^{\rm aux}}. \quad (6)$$

Advected and auxiliary variables are updated similarly, but they differ in their usage. In particular, auxiliary variables are passed into the EOS routines. Examples of auxiliary and advected variables, respectively, might include the electron fraction, Y_e, used in simulations of core-collapse supernovae, and angular momentum in two-dimensional simulations of a rotating star in cylindrical (axisymmetric) coordinates. Both of these evolution equations include user-specified sources, $S_{{\rm ext},\rho C_k^{\rm adv}}$ and $S_{{\rm ext},\rho C_k^{\rm aux}}$. We refer to $\mathbf{U} = (\rho, \rho\mathbf{u}, \rho E, \rho X_k, \rho C_k^{\rm adv}, \rho C_k^{\rm aux})$ as the conserved variables.

3. EQUATION OF STATE AND REACTION NETWORK

CASTRO is written in a modular fashion so that the routines for the EOS and reaction network can be supplied by the user. However, for the test problems presented later we use routines that come with the CASTRO distribution.

Each EOS must provide an interface for obtaining thermodynamic quantities from ρ, e, and X_k. One EOS which is supplied with the CASTRO distribution is the gamma-law EOS, which relates pressure and temperature, T, to ρ and e via:

$$p = \rho e(\gamma - 1) = \frac{\rho k_B T}{\mu m_p}. \quad (7)$$

Here, γ is the ratio of specific heats (e.g., $\gamma = 5/3$ for a monatomic gas), k_B is Boltzmann's constant, m_p is the mass of the proton, and the mean molecular weight, μ, is determined by

$$\frac{1}{\mu} = \sum_k \frac{X_k}{A_k}, \quad (8)$$

with A_k the atomic weight of species k.

The CASTRO distribution also includes more complex equations of state describing stellar matter, including the Helmholtz EOS (Timmes & Swesty 2000; Fryxell et al. 2000) which includes degenerate/relativistic electrons, ions (as a perfect gas), and radiation, and the Lattimer–Swesty EOS, which describes dense nuclear matter (Lattimer & Swesty 1991).[6] For tabular equations of state, it is common that ρ, T, and X_k are inputs, in which case a Newton–Raphson iteration is typically used to invert the EOS.

CASTRO can support any general reaction network that takes as inputs the density, temperature, and mass fractions, and returns updated mass fractions and the energy release (or decrease). The input temperature is computed from the EOS before each call to the reaction network. In general, we expect the reaction network to evolve the species according to

$$\frac{dX_k}{dt} = \dot{\omega}_k(\rho, X_k, T). \quad (9)$$

Reaction rates can be extremely temperature sensitive, so in most cases, the reaction network should be written to evolve the temperature for the purposes of evaluating the rates. Close to nuclear statistical equilibrium, the energy release and change in abundances can rapidly change sign if the rates are not evaluated with a temperature field consistent with the evolving energy (Müller 1986). At the end of the burning step, we use the energy release to update the total energy, E. The density remains unchanged during the burning.

[6] Code obtained from http://www.astro.sunysb.edu/dswesty/lseos.html.

4. GRAVITY

CASTRO supports several different options for how to specify and/or compute the gravitational acceleration. The simplest option is a gravitational field that is constant in space and time; this can be used for small-scale problems in which the variation of gravity throughout the computational domain is negligible. This option is available in one-dimensional Cartesian coordinates, two-dimensional cylindrical or Cartesian coordinates, and three-dimensional Cartesian coordinates.

A second approach uses a monopole approximation to compute a radial gravity field consistent with the mass distribution. Because the algorithm subcycles in time we construct a separate one-dimensional radial density profile at each level at each time needed. Once the radial density profile is defined, gravity is computed as a direct integral of the mass enclosed. This field is then interpolated back onto the original grids.

The most general option is to solve the Poisson equation for self-gravity, i.e., solve

$$\nabla^2\phi = 4\pi G\rho \quad (10)$$

for ϕ, and define $\mathbf{g} = -\nabla\phi$. This can be used in any of the coordinate systems. At boundaries away from the star, we set inhomogeneous Dirichlet boundary conditions for ϕ; these values are determined by computing the monopole approximation for $\mathbf{g}$ on the coarsest level, integrating this profile radially outward to create $\phi(r)$, and interpolating ϕ onto the domain boundaries to define the boundary conditions for the solve. Boundaries that pass through the center of the star use symmetry boundary conditions.

The Poisson equation is discretized using standard finite difference approximations and the resulting linear system is solved using geometric multigrid techniques, specifically V-cycles and red-black Gauss–Seidel relaxation. For multilevel calculations, special attention is paid to the synchronization of the gravitational forcing across levels, which will be discussed in Section 6.

There is also an option to add the gravitational forcing due to a specified point mass to either of the self-gravity options described above.

5. SINGLE-LEVEL INTEGRATION ALGORITHM

The time evolution of $\mathbf{U}$ can be written in the form

$$\frac{\partial\mathbf{U}}{\partial t} = -\nabla\cdot\mathbf{F} + \mathbf{S}_{\rm react} + \mathbf{S}, \quad (11)$$

where $\mathbf{F}$ is the flux vector, $\mathbf{S}_{\rm react}$ are the reaction source terms, and $\mathbf{S}$ are the non-reaction source terms, which includes any user-defined external sources, $\mathbf{S}_{\rm ext}$. We use Strang splitting (Strang 1968) to discretize the advection-reaction equations. In other words, to advance the solution, $\mathbf{U}$, by one time step, Δt, we first advance the nuclear reaction network by $\Delta t/2$,

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \frac{\Delta t}{2}\mathbf{S}_{\rm react}^n, \quad (12a)$$

then advect the solution by Δt, ignoring the reaction terms,

$$\mathbf{U}^{(2)} = \mathbf{U}^{(1)} - \Delta t \nabla \cdot \mathbf{F}^{n+1/2} + \Delta t \frac{\mathbf{S}^{(1)} + \mathbf{S}^{(2)}}{2}, \tag{12b}$$

and finally advance the nuclear reaction network by another $\Delta t/2$,

$$\mathbf{U}^{n+1} = \mathbf{U}^{(2)} + \frac{\Delta t}{2}\mathbf{S}^{(2)}_{\rm react}. \tag{12c}$$

The construction of $\mathbf{F}$ is purely explicit, and based on an unsplit Godunov method. The solution, $\mathbf{U}$, and source terms, $\mathbf{S}$, are defined on cell centers; we predict the primitive variables, $\mathbf{Q} = (\rho, \mathbf{u}, p, \rho e, X_k, C_k^{\rm adv}, C_k^{\rm aux})$, from cell centers at time t^n to edges at time $t^{n+1/2}$ and use an approximate Riemann solver to construct fluxes, $\mathbf{F}^{n+1/2}$, on cell faces. This algorithm is formally second order in both space and time.

5.1. Single-Level Flow Chart

At the beginning of each time step, we assume that, in the case of self-gravity, $\mathbf{g}$ is defined consistently with the current mass distribution in $\mathbf{U}$. The algorithm at a single level of refinement is composed of the following steps:

Step 1. Advance the nuclear reaction network through a time interval of $\Delta t/2$.

Define $\mathbf{U}^{(1)} = \mathbf{U}^n$ with the exception of

$$(\rho E)^{(1)} = (\rho E)^n + \frac{\Delta t}{2}(\rho H_{\rm nuc})^n, \tag{13}$$

$$(\rho X_k)^{(1)} = (\rho X_k)^n + \frac{\Delta t}{2}(\rho \dot{\omega}_k)^n, \tag{14}$$

where $(\rho H_{\rm nuc})^n$ and $(\rho \dot{\omega}_k)^n$ are computed using calls to the user-defined reaction network. Note that ρ is unchanged during this step.

Step 2. Advect the solution through Δt.

Advance the solution using time-centered fluxes and an explicit representation of the source term, neglecting the contribution from reactions which are taken into account in steps 1 and 4 (the asterisk superscript notation indicates that we will later correct this state to effectively time center the source terms):

$$\mathbf{U}^{(2,**)} = \mathbf{U}^{(1)} - \Delta t \nabla \cdot \mathbf{F}^{n+1/2} + \Delta t \mathbf{S}^{(1)}, \tag{15}$$

where

$$\mathbf{S}^{(1)}_{\mathbf{U}} = \begin{pmatrix} S_\rho \\ \mathbf{S}_{\rho \mathbf{u}} \\ S_{\rho E} \\ S_{\rho X_k} \\ S_{\rho C_k^{\rm adv}} \\ S_{\rho C_k^{\rm aux}} \end{pmatrix}^{(1)} = \begin{pmatrix} S_{{\rm ext},\rho} \\ (\rho \mathbf{g})^{(1)} + \mathbf{S}_{{\rm ext},\rho \mathbf{u}} \\ (\rho \mathbf{u} \cdot \mathbf{g})^{(1)} + S_{{\rm ext},\rho E} \\ S_{{\rm ext},\rho X_k} \\ S_{{\rm ext},\rho C_k^{\rm adv}} \\ S_{{\rm ext},\rho C_k^{\rm aux}} \end{pmatrix}^{(1)}. \tag{16}$$

The construction of the fluxes is described in detail in Section 5.2. We note that in the single-level algorithm we can use the gravitational forcing computed in step 3 of the previous time step, since the density has not changed.

After the advective update, we ensure that the solution is physically meaningful by forcing the density to exceed a non-negative, user-defined minimum value. We also ensure that the mass fractions are all non-negative and sum to one.

We also have an option for a user-defined sponge in order to prevent the velocities in the upper atmosphere from becoming too large, and subsequently, the time step from becoming too small. We multiply the velocity by $1/(1+\Delta t\ \kappa\ f_{\rm damp}(\rho))$, where κ is the sponge strength, and $f_{\rm damp}$ is a smooth function of density that varies from 0 to 1. Full details of the sponge are given in Zingale et al. (2009). Finally, we adjust $(\rho E)^{(2,**)}$ to be consistent with $\mathbf{u}^{(2,**)}$.

Step 3. Correct the solution with time-centered source terms and compute gravity at t^{n+1}.

We correct the solution by effectively time-centering the source terms. First, we correct $\mathbf{U}$ with updated external sources:

$$\mathbf{U}^{(2),*} = \mathbf{U}^{(2,**)} + \frac{\Delta t}{2}\left(S^{(2,**)}_{{\rm ext},\mathbf{U}} - S^{(1)}_{{\rm ext},\mathbf{U}}\right). \tag{17}$$

Next, we evaluate gravity using $\rho^{(2,*)}$. If using full gravity we solve

$$\mathbf{g}^{(2,*)} = -\nabla \phi^{(2,*)}, \qquad \nabla^2 \phi^{(2,*)} = 4\pi G \rho^{(2,*)}, \tag{18}$$

where we supply an initial guess for $\phi^{(2,*)}$ from the previous solve. In the single-level algorithm described here, $\mathbf{g}^{(2,*)}$ is saved to be used as $\mathbf{g}^{(1)}$ in step 2 of the next time step. This suffices in the single-level algorithm because ρ does not change between the end of step 3 of one time step and the start of step 2 of the next time step.

We then correct the solution with the updated gravity:

$$(\rho \mathbf{u})^{(2)} = (\rho \mathbf{u})^{(2,*)} + \frac{\Delta t}{2}[(\rho \mathbf{g})^{(2,*)} - (\rho \mathbf{g})^{(1)}], \tag{19}$$

$$(\rho E)^{(2)} = (\rho E)^{(2,*)} + \frac{\Delta t}{2}[(\rho \mathbf{u} \cdot \mathbf{g})^{(2,*)} - (\rho \mathbf{u} \cdot \mathbf{g})^{(1)}]. \tag{20}$$

For all other conserved variables other than $\rho \mathbf{u}$ and ρE, $\mathbf{U}^{(2)} = \mathbf{U}^{(2,*)}$. We note here that the time discretization of the gravitational forcing terms differs from that in the FLASH (Fryxell et al. 2000) and ENZO (O'Shea et al. 2005) codes, where the gravitational forcing at $t^{n+1/2}$ is computed by extrapolation from values at t^n and t^{n-1} (see also Bryan et al. 1995). Our discretization of the gravitational terms is consistent with our predictor-corrector approach in the handling of other source terms.

Step 4. Advance the nuclear reaction network through a time interval of $\Delta t/2$.

Define $\mathbf{U}^{n+1} = \mathbf{U}^{(2)}$ with the exception of

$$(\rho E)^{n+1} = (\rho E)^{(2)} + \frac{\Delta t}{2}(\rho H_{\rm nuc})^{(2)}, \tag{21}$$

$$(\rho X_k)^{n+1} = (\rho X_k)^{(2)} + \frac{\Delta t}{2}(\rho \dot{\omega}_k)^{(2)}. \tag{22}$$

We also include an option to modify any component of the new-time state as needed to account for special user requirements.

Step 5. Compute the new time step.

The time step is computed using the standard CFL condition for explicit methods, with additional constraints (such as one based on rate of burning) possible as needed. The user sets a CFL factor, $\sigma^{\rm CFL}$, between 0 and 1. The sound speed, c, is computed by the EOS, and for a calculation in $n_{\rm dim}$ dimensions,

$$\Delta t = \sigma^{\rm CFL} \min_{i=1\ldots n_{\rm dim}} \{\Delta t_i\}, \tag{23}$$

where

$$\Delta t_i = \min_{\mathbf{x}} \left\{ \frac{\Delta x_i}{|\mathbf{u}_i| + c} \right\}. \quad (24)$$

$\min_{\mathbf{x}}$ is the minimum taken over all computational grid cells in the domain.

This concludes the single-level algorithm description. We note that whenever the kinetic energy dominates the total energy, making the calculation of e from E numerically unreliable, we use a method similar to the dual-energy approach described in Bryan et al. (1995) to compute the internal energy with sufficient precision. In practice, this involves evolving ρe in time and using this solution when appropriate.

5.2. Construction of Fluxes

We use an unsplit Godunov method with characteristic tracing and full corner coupling in three dimensions (Miller & Colella 2002) to compute time-centered edge states. We have replaced the PPM limiters in Miller & Colella (2002) with an updated PPM algorithm that is designed to preserve accuracy at smooth extrema and is insensitive to asymmetries caused by roundoff error (Colella & Sekora 2008; McCorquodale & Colella 2010). CASTRO also has options to use the unsplit piecewise-linear algorithm described in Colella (1990), Saltzman (1994), or to retain the PPM limiters in Miller & Colella (2002), which were originally developed in Colella & Woodward (1984) using a split integrator.

There are four major steps in the construction of the face-centered fluxes, $\mathbf{F}^{n+1/2}$, that are used in step 2 in Section 5.1 to update the solution. We also include details on the solution of the Riemann problem. In summary,

Step 2.1. Rewrite the state, $\mathbf{U}^{(1)}$, in terms of primitive variables, $\mathbf{Q}^{(1)}$.

Step 2.2. Construct a piecewise parabolic approximation of $\mathbf{Q}^{(1)}$ within each cell.

Step 2.3. Predict average values of $\mathbf{Q}^{(1)}$ on edges over the time step using characteristic extrapolation.

Step 2.4. Compute fluxes, $\mathbf{F}^{n+1/2}$, using an approximate Riemann problem solver.

We expand each of these steps in more detail below.

Step 2.1. Compute primitive variables and source terms.

We define $\mathbf{Q}^{(1)} = (\rho, \mathbf{u}, p, \rho e, X_k, C_k^{\rm adv}, C_k^{\rm aux})^{(1)}$. The pressure, p, is computed through a call to the EOS using ρ, e, and X_k. Note that we also include ρe in $\mathbf{Q}$; this quantity is used in the approximate Riemann solver to avoid an EOS call to evaluate the energy flux, analogous to the effective dynamics for $\gamma = p/(\rho e) + 1$ in the Colella & Glaz (1985) approximate Riemann solver.

For the overall integration algorithm, we want to include the effect of source terms except for reactions in the characteristic tracing (step 2.2). (Reactions are treated separately using a symmetric operator-split approach in steps 1 and 4 of the algorithm.) The time evolution equations written in terms of the primitive variables, $\mathbf{Q}$, and omitting contributions from reactions, are

$$\frac{\partial \mathbf{Q}}{\partial t} = \begin{pmatrix} -\mathbf{u}\cdot\nabla\rho - \rho\nabla\cdot\mathbf{u} \\ -\mathbf{u}\cdot\nabla\mathbf{u} - \frac{1}{\rho}\nabla p \\ -\mathbf{u}\cdot\nabla p - \rho c^2\nabla\cdot\mathbf{u} \\ -\mathbf{u}\cdot\nabla(\rho e) - (\rho e + p)\nabla\cdot\mathbf{u} \\ -\mathbf{u}\cdot\nabla X_k \\ -\mathbf{u}\cdot\nabla C_k^{\rm adv} \\ -\mathbf{u}\cdot\nabla C_k^{\rm aux} \end{pmatrix} + \mathbf{S}_{\mathbf{Q}},$$

where

$$\mathbf{S}_{\mathbf{Q}} = \begin{pmatrix} S_\rho \\ \mathbf{S}_{\mathbf{u}} \\ S_p \\ S_{\rho e} \\ S_{X_k} \\ S_{C_k^{\rm adv}} \\ S_{C_k^{\rm aux}} \end{pmatrix} = \begin{pmatrix} S_{{\rm ext},\rho} \\ \mathbf{g} + \frac{1}{\rho}\mathbf{S}_{{\rm ext},\rho\mathbf{u}} \\ \frac{p_e}{\rho} S_{{\rm ext},\rho E} + p_\rho S_{{\rm ext}\rho} + \frac{p_{X_k}}{\rho} S_{{\rm ext},\rho X_k} \\ S_{{\rm ext},\rho E} \\ \frac{1}{\rho} S_{{\rm ext},\rho X_k} \\ \frac{1}{\rho} S_{{\rm ext},\rho C_k^{\rm adv}} \\ \frac{1}{\rho} S_{{\rm ext},\rho C_k^{\rm aux}} \end{pmatrix}. \quad (25)$$

Here, c is the sound speed, defined as $c = \sqrt{\Gamma_1 p/\rho}$, with $\Gamma_1 = d\log p/d\log\rho|_s$, with s the entropy. The remaining thermodynamic derivatives are $p_e = \partial p/\partial e|_{\rho,X_k}$, $p_\rho = \partial p/\partial\rho|_{e,X_k}$, and $p_{X_k} = \partial p/\partial X_k|_{\rho,e,X_{j,(j\neq k)}}$. Often, the EOS is a function of ρ, T, and X_k, and returns derivatives with these quantities held constant. In terms of the latter derivatives, our required thermodynamic derivatives are

$$p_e = \left(\left.\frac{\partial e}{\partial T}\right|_{\rho,X_k}\right)^{-1} \left.\frac{\partial p}{\partial T}\right|_{\rho,X_k}, \quad (26a)$$

$$p_\rho = \left.\frac{\partial p}{\partial \rho}\right|_{T,X_k} - \left(\left.\frac{\partial e}{\partial T}\right|_{\rho,X_k}\right)^{-1} \left.\frac{\partial p}{\partial T}\right|_{\rho,X_k} \left.\frac{\partial e}{\partial \rho}\right|_{T,X_k}, \quad (26b)$$

$$p_{X_k} = \left.\frac{\partial p}{\partial X_k}\right|_{\rho,T,X_{j,(j\neq k)}} - \left(\left.\frac{\partial e}{\partial T}\right|_{\rho,X_k}\right)^{-1} \left.\frac{\partial p}{\partial T}\right|_{\rho,X_k} \left.\frac{\partial e}{\partial X_k}\right|_{\rho,T,X_{j,(j\neq k)}}. \quad (26c)$$

Step 2.2. Reconstruct parabolic profiles within each cell.

In this step, we construct a limited piecewise parabolic profile of each q in $\mathbf{Q}$ (we use q to denote an arbitrary primitive variable from $\mathbf{Q}$). These constructions are performed in each coordinate direction separately. The default option in CASTRO is to use a new limiting procedure that avoids reducing the order of the reconstruction at smooth local extrema. The details of this construction are given in Colella & Sekora (2008) and McCorquodale & Colella (2010). In summary,

1. *Step 2.2a.* For each cell, we compute the spatial interpolation of q^n to the high and low faces of cell q_i using a limited cubic interpolation formula. These interpolants are denoted by $q_{i,+}$ and $q_{i,-}$.
2. *Step 2.2b.* Construct quadratic profiles using $q_{i,-}$, q_i, and $q_{i,+}$.

$$q_i^{\rm quad}(x) = q_{i,-} + \xi(x)\{q_{i,+} - q_{i,-} + q_{6,i}[1 - \xi(x)]\}, \quad (27)$$

$$q_6 = 6q_i - 3(q_{i,-} + q_{i,+}), \quad (28)$$

$$\xi(x) = \frac{x - ih}{h}, \quad 0 \leqslant \xi(x) \leqslant 1, \quad (29)$$

where h is the mesh spacing in the direction of interpolation. Also, as in Miller & Colella (2002), we compute a flattening coefficient, $\chi \in [0, 1]$, used in the edge state prediction to further limit slopes near strong shocks. The computation of χ is identical to the approach used in FLASH (Fryxell et al. 2000), except that a flattening coefficient of 1 indicates that no additional limiting takes place, whereas a flattening coefficient of 0 means we effectively drop order to a first-order Godunov scheme, which is opposite of the convention used in FLASH.

Step 2.3. Characteristic extrapolation.

We begin by extrapolating $\mathbf{Q}^{(1)}$ to edges at $t^{n+1/2}$. The edge states are dual valued, i.e., at each face, there is a left state and a right state estimate, denoted $q_{L,i+1/2}$ and $q_{R,i+1/2}$ (we write the equations in one dimension for simplicity). The spatial extrapolation is one dimensional, i.e., transverse derivatives are omitted and accounted for later.

1. *Step 2.3a.* Integrate the quadratic profiles. We are essentially computing the average value swept out by the quadratic profile across the face assuming the profile is moving at a speed λ_k, where λ_k is a standard wave speed associated with gas dynamics.
   Define the following integrals, where $\sigma_k = |\lambda_k|\Delta t/h$:

$$\mathcal{I}_{i,+}(\sigma_k) = \frac{1}{\sigma_k h}\int_{(i+1/2)h-\sigma_k h}^{(i+1/2)h} q_i^{\rm quad}(x)dx, \tag{30a}$$

$$\mathcal{I}_{i,-}(\sigma_k) = \frac{1}{\sigma_k h}\int_{(i-1/2)h}^{(i-1/2)h+\sigma_k h} q_i^{\rm quad}(x)dx. \tag{30b}$$

   Substituting Equation (27) gives

$$\mathcal{I}_{i,+}(\sigma_k) = q_{i,+} - \frac{\sigma_k}{2}\left[q_{i,+} - q_{i,-} - \left(1 - \frac{2}{3}\sigma_k\right)q_{6,i}\right], \tag{31a}$$

$$\mathcal{I}_{i,-}(\sigma_k) = q_{i,-} + \frac{\sigma_k}{2}\left[q_{i,+} - q_{i,-} + \left(1 - \frac{2}{3}\sigma_k\right)q_{6,i}\right]. \tag{31b}$$

2. *Step 2.3b.* Obtain a left and right edge state at $t^{n+1/2}$ by applying a characteristic tracing operator (with flattening) to the integrated quadratic profiles. Note that we also include the explicit source term contribution:

$$q_{L,i+1/2} = q_i - \chi_i \sum_{k:\lambda_k \geqslant 0} \mathbf{l}_k \cdot [q_i - \mathcal{I}_{i,+}(\sigma_k)]\mathbf{r}_k + \frac{\Delta t}{2}S^n_{q,i}, \tag{32a}$$

$$q_{R,i-1/2} = q_i - \chi_i \sum_{k:\lambda_k \leqslant 0} \mathbf{l}_k \cdot [q_i - \mathcal{I}_{i,-}(\sigma_k)]\mathbf{r}_k + \frac{\Delta t}{2}S^n_{q,i}. \tag{32b}$$

   In non-Cartesian coordinates, volume source terms are added to the traced states. Here, $\mathbf{r}_k$ and $\mathbf{l}_k$ are the standard right column and left row eigenvectors associated with the equations of gas dynamics (see Toro 1997).
   An unsplit approximation that includes full corner coupling is constructed by constructing increasingly accurate approximations to the transverse derivatives. The details follow exactly as given in Section 4.2.1 in Miller & Colella (2002), except for the solution of the Riemann problem, which is described in step 2.4.

Step 2.4. Compute fluxes

The fluxes are computed using an approximate Riemann solver. The solver used here is essentially the same as that used in P. Colella et al. (1997, unpublished), which is based on ideas discussed in Bell et al. (1989). This solver is computationally faster and considerably simpler than the approximate Riemann solver introduced by Colella & Glaz (1985). The Colella and Glaz solver was based on an effective dynamics for γ and was designed for real gases that are well approximated by this type of model. The approximate Riemann solver used in CASTRO is suitable for a more general convex EOS.

As with other approximate Riemann solvers, an important design principle is to avoid additional evaluations of the EOS when constructing the numerical flux. For that reason, we include ρe in $\mathbf{Q}$ and compute $(\rho e)_{L,R}$. The information carried in ρe is overspecified but it allows us to compute an energy flux without an inverse call to the EOS.

The numerical flux computation is based on approximating the solution to the Riemann problem and evaluating the flux along the $x/t = 0$ ray. The procedure is basically a two-step process in which we first approximate the solution in phase space and then interpret the phase space solution in real space.

1. *Step 2.4a.* To compute the phase space solution, we first solve for p^* and u^*, the pressure between the two acoustic waves and the velocity of the contact discontinuity, respectively. These quantities are computed using a linearized approximation to the Rankine–Hugoniot relations. We first define $\Gamma_{1,L/R}$ by using the cell-centered values on either side of the interface. Next, we compute Lagrangian sound speeds, $W_L = \sqrt{\Gamma_{1,L}p_L\rho_L}$ and $W_R = \sqrt{\Gamma_{1,R}p_R\rho_R}$, and the corresponding Eulerian sound speeds $c_{L,R} = \sqrt{\Gamma_{1,L,R}p_{L,R}/\rho_{L,R}}$. Then,

$$p^* = \frac{W_L p_R + W_R p_L + W_L W_R(u_L - u_R)}{W_L + W_R}, \tag{33a}$$

$$u^* = \frac{W_L u_L + W_R u_R + (p_L - p_R)}{W_L + W_R}. \tag{33b}$$

   From u^* and p^*, we can compute

$$\rho^*_{L,R} = \rho_{L,R} + \frac{p^* - p_{L,R}}{c^2_{L,R}}, \tag{34a}$$

$$(c^*_{L,R})^2 = \Gamma_{1,L,R}p^*_{L,R}/\rho^*_{L,R}, \tag{34b}$$

$$(\rho e)^*_{L,R} = (\rho e)_{L,R} + (p^* - p_{L,R})\frac{(e + p/\rho)_{L,R}}{c^2_{L,R}}, \tag{34c}$$

$$v^*_{L,R} = v_{L,R}, \tag{34d}$$

   where v generically represents advected quantities (which includes transverse velocity components). Here, the notation $^*_{L,R}$ refers to values on the left and right side of the contact discontinuity.
2. *Step 2.4b.* The next step in the approximate Riemann solver is to interpret this phase space solution. If $u^* > 0$ then the contact discontinuity is moving to the right and numerical flux depends on the speed and structure of the acoustic wave connecting $\mathbf{Q}_L$ and $\mathbf{Q}^*_L$ associated with the $\lambda = u - c$ eigenvalue. Similarly, if $u^* < 0$ then the contact is moving to the left and the numerical flux depends on the speed and structure of the acoustic wave connecting $\mathbf{Q}_R$ and $\mathbf{Q}^*_R$ associated with the $\lambda = u + c$ eigenvalue. Here, we discuss in detail the case in which $u^* > 0$; the other case is treated analogously.
   For $u^* > 0$, we define $\lambda_L = u_L - c_L$ and $\lambda^*_L = u^*_L - c^*_L$. If $p^*_L > p_L$ then the wave is a shock wave and we define a

shock speed $\sigma = 1/2(\lambda_L + \lambda_L^*)$. For that case if $\sigma > 0$ then the shock is moving to the right and we define the Godunov state $\mathbf{Q}_G = \mathbf{Q}_L$; otherwise $\mathbf{Q}_G = \mathbf{Q}_L^*$. The rarefaction case is somewhat more complex. If both λ_L and λ_L^* are negative, then the rarefaction fan is moving to the left and $\mathbf{Q}_G = \mathbf{Q}_L^*$. Similarly, if both λ_L and λ_L^* are positive, then the rarefaction fan is moving to the right and $\mathbf{Q}_G = \mathbf{Q}_L$. However, in the case in which $\lambda_L < 0 < \lambda_L^*$, the rarefaction spans the $x/t = 0$ ray and we need to interpolate the solution. For this case, we define

$$\mathbf{Q}_G = \alpha \mathbf{Q}_L^* + (1 - \alpha)\mathbf{Q}_L, \tag{35}$$

where $\alpha = \lambda_L/(\lambda_L - \lambda_L^*)$. This choice of α corresponds to linearly interpolating $\mathbf{Q}$ through the rarefaction to approximate the state that propagates with zero speed.

As noted above the case in which $u^* < 0$ is treated analogously. When $u^* = 0$ we compute $\mathbf{Q}_G$ by averaging $\mathbf{Q}_L^*$ and $\mathbf{Q}_R^*$. For the Riemann problem approximation, we allow for user-specified floors for ρ, p, and c to prevent the creation of non-physical values.

The fluxes can then be evaluated from the final $\mathbf{Q}_G$. A small quadratic artificial viscosity that is proportional to the divergence of the velocity field is added to the flux in order to add additional dissipation at strong compressions. We also scale all the species fluxes so that they sum to the density flux, as in the sCMA algorithm described by Plewa & Müller (1999).

6. AMR

Our approach to AMR in CASTRO uses a nested hierarchy of logically rectangular grids with simultaneous refinement of the grids in both space and time. The integration algorithm on the grid hierarchy is a recursive procedure in which coarse grids are advanced in time, fine grids are advanced multiple steps to reach the same time as the coarse grids and the data at different levels are then synchronized.

The AMR methodology was introduced by Berger & Oliger (1984); it has been demonstrated to be highly successful for gas dynamics by Berger & Colella (1989) in two dimensions and by Bell et al. (1994) in three dimensions.

6.1. Creating and Managing the Grid Hierarchy

6.1.1. Overview

The grid hierarchy is composed of different levels of refinement ranging from coarsest ($\ell = 0$) to finest ($\ell = \ell_{\rm finest}$). The maximum number of levels of refinement allowed, $\ell_{\rm max}$, is specified at the start of a calculation. At any given time in the calculation, there may not be that many levels in the hierarchy, i.e., $\ell_{\rm finest}$ can change dynamically as the calculation proceeds as long as $\ell_{\rm finest} \leqslant \ell_{\rm max}$. Each level is represented by the union of non-overlapping rectangular grids of a given resolution. Each grid is composed of an even number of cells in each coordinate direction; cells are the same size in each coordinate direction but grids may have different numbers of cells in each direction. Figure 1 shows a cartoon of AMR grids in two dimensions with two levels of refinement.

In this implementation, the refinement ratio between levels ℓ and $\ell+1$, which we call r_ℓ, is always two or four, with the same factor of refinement in each coordinate direction. The grids are properly nested, in the sense that the union of grids at level $\ell + 1$ is contained in the union of grids at level ℓ. Furthermore, the containment is strict in the sense that, except at physical boundaries, the level ℓ grids are large enough to guarantee that there is a border at least $n_{\rm proper}$ level ℓ cells wide surrounding each level $\ell + 1$ grid (grids at all levels are allowed to extend to the physical boundaries so the proper nesting is not strict there). The parameter $n_{\rm proper}$ is two for factor 2 refinement, and one for factor 4 refinement, since four ghost cells are needed for the PPM algorithm.

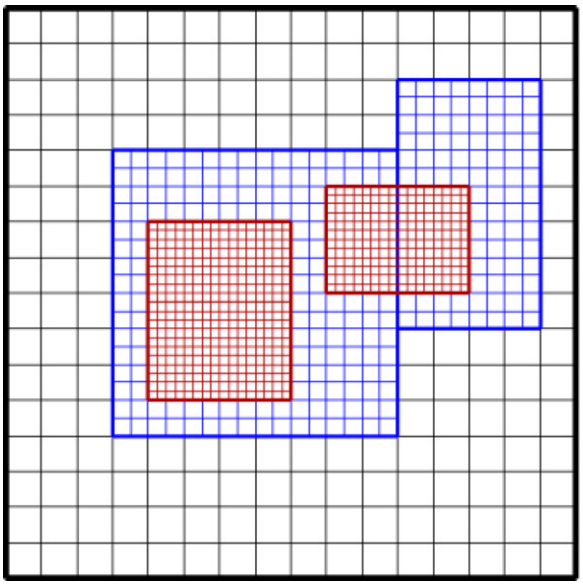

Figure 1. Cartoon of AMR grids with two levels of factor 2 refinement. The coarsest grid covers the domain with 16^2 cells. Bold lines represent grid boundaries. The two intermediate resolution grids are at level 1 and the cells are a factor of two finer than those at level 0. The two finest grids are at level 2 and the cells are a factor of 2 finer than the level 1 cells. Note that the level 2 grids are properly nested within the union of level 1 grids, but there is no direct parent–child connection.

(A color version of this figure is available in the online journal.)

6.1.2. Error Estimation and Regridding

We initialize the grid hierarchy and regrid following the procedure outlined in Bell et al. (1994). Given grids at level ℓ, we use an error estimation procedure to tag cells where the error, as defined by user-specified routines, is above a given tolerance. Typical error criteria include first or second derivatives of the state variables or quantities derived from the state variables, or the state variables or derived quantities themselves. A user can specify that any or all of the criteria must be met to refine the cell; one can also specify criteria that ensure that a cell not be refined. For example, one could specify that a cell be refined if $\rho > \rho_{\rm crit}$ and $((\nabla^2 T) > (\nabla^2 T)_{\rm crit}$ or $|\nabla p| > |\nabla p|_{\rm crit})$, where $\rho_{\rm crit}$, $(\nabla^2 T)_{\rm crit}$, and $|\nabla p|_{\rm crit}$ are constants specified by the user.

The tagged cells are grouped into rectangular grids at level ℓ using the clustering algorithm given in Berger & Rigoutsos (1991). These rectangular patches are refined to form the grids at level $\ell + 1$. Large patches are broken into smaller patches for distribution to multiple processors based on a user-specified *max_grid_size* parameter.

At the beginning of every k_ℓ level ℓ time steps, where $k_\ell \geqslant 1$ is specified by the user at run time, new grid patches are defined at all levels $\ell + 1$ and higher if $\ell < \ell_{\rm max}$. In regions previously covered by fine grids, the data are simply copied from old grids to new; in regions which are newly refined, data are interpolated from underlying coarser grids.

6.1.3. Enlarging the Domain

The finest resolution of a calculation can vary in time; however, the coarsest resolution covering the domain does not change during a single run. However, a feature has been added to the CASTRO distribution that allows a user to restart a calculation in a larger domain covered by a coarser resolution, provided the data exist to initialize the larger domain. This is useful in simulations during which a star expands dramatically, for example. Using this strategy, one could periodically stop the simulation, double the domain size, and restart the calculation in the larger domain.

6.2. Multilevel Algorithm

6.2.1. Overview

The multilevel time stepping algorithm can most easily be thought of as a recursive procedure. In the case of zero or constant gravity, to advance level ℓ, $0 \leqslant \ell \leqslant \ell_{\max}$ the following steps are taken. Here, the phrase, "Advance **U**" refers to steps 1–4 of the single-level algorithm described in the previous section.

1. If $\ell = 0$, compute the new time steps for all levels as follows:
   (a) Compute the appropriate time step for each level, $\Delta t^{\ell',*}$ using the procedure described in step 5 of the previous section.
   (b) Define $R_{\ell'}$ as the ratio of the level 0 cell size to the level ℓ' cell size.
   (c) Define $\Delta t^0 = \mathbf{min}_{\ell'}(R_{\ell'}\Delta t^{\ell',*})$.
   (d) Define $\Delta t^{\ell'} = \Delta t^0/R_{\ell'}$ for all ℓ', $0 \leqslant \ell' \leqslant \ell_{\max}$.
2. Advance **U** at level ℓ in time as if it is the only level, filling boundary conditions for **U** from level $\ell - 1$ if level $\ell > 0$, and from the physical domain boundaries.
3. If $\ell < \ell_{\max}$,
   (a) Advance **U** at level $(\ell + 1)$ for r_ℓ time steps with time step $\Delta t^{\ell+1} = \frac{1}{r_\ell}\Delta t^{\ell}$.
   (b) Synchronize the data between levels ℓ and $\ell + 1$.
      i. Volume average **U** at level $\ell + 1$ onto level ℓ grids.
      ii. Correct **U** in all level ℓ cells adjacent to but not covered by the union of level $\ell+1$ grids through an explicit refluxing operation as described in Berger & Colella (1989).

6.2.2. Monopole Gravity

When we use the monopole gravity assumption in a multilevel simulation, we can no longer exploit the fact that ρ at level ℓ at the end of step 3 of one time step is unchanged when one reaches the beginning of step 2 of the next level ℓ time step. If $\ell < \ell_{\max}$, then potential changes in ρ come from two sources:

1. ρ at level ℓ under the level $\ell + 1$ grids is replaced by the volume average of ρ at level $\ell + 1$.
2. The explicit refluxing step between levels ℓ and $\ell + 1$ modifies ρ on all level ℓ cells adjacent to but not covered by the union of level $\ell + 1$ grids.

In addition, because the grids are dynamically created and destroyed through regridding, at the beginning of step 2 of a level ℓ time step, there may not be a value for **g** from the previous step, because this region of space was previously not covered by level ℓ grids.

In order to address all of these changes, we simply compute $\mathbf{g}^{(1)}$ at the beginning of step 2 of each time step at each level, rather than copying it from $\mathbf{g}^{(2,*)}$ from step 3 of the previous time step as in the single-level algorithm. This captures any changes in grid structure due to regridding, and reflects any changes in density due to refluxing or volume averaging.

6.2.3. Full Gravity Solve

Overview. Solving the Poisson equation for self-gravity on a multilevel grid hierarchy introduces additional complications. We start by defining some necessary notation. We define L^ℓ as an approximation to ∇^2 at level ℓ, with the assumption that Dirichlet boundary conditions are supplied on the boundary of the union of level ℓ grids (we allow more general boundary conditions at physical boundaries), and define a *level solve* as the process of solving

$$L^\ell \phi^\ell = 4\pi G \rho^\ell$$

at level ℓ.

We define $L^{\rm comp}_{\ell,m}$ as the composite grid approximation to ∇^2 on levels ℓ through m, and define a *composite solve* as the process of solving

$$L^{\rm comp}_{\ell,m} \phi^{\rm comp} = 4\pi G \rho^{\rm comp}$$

on levels ℓ through m. The solution to the composite solve satisfies

$$L^m \phi^{\rm comp} = 4\pi G \rho^m$$

at level m, but satisfies

$$L^{\ell'} \phi^{\ell'} = 4\pi G \rho^{\ell'}$$

for $\ell \leqslant \ell' < m$ only on the regions of each level *not* covered by finer grids or adjacent to the boundary of the finer grid region. In regions of a level ℓ' grid covered by level $\ell'+1$ grids, the solution is defined as the volume average of the solution at $\ell'+1$; in level ℓ' cells immediately adjacent to the boundary of the union of level $\ell' + 1$ grids, a modified interface operator is used that reflects the geometry of the interface (see, e.g., Almgren et al. 1998 for details of the multilevel cell-centered interface stencil).

In an algorithm without subcycling one can perform a composite solve at every time step, as described in Ricker (2008), to solve for ϕ on all levels. Because the CASTRO algorithm uses subcycling in time, however, we must use level solves at times when the solution is not defined at all levels, and then synchronize the solutions at different levels as appropriate. Even without changes in ρ due to volume averaging and refluxing, replacing a composite solve by separate level solves generates a mismatch in the normal gradient of ϕ at the boundary between each level. We correct these mismatches with a multilevel *correction solve*, which is a two-level composite solve for a correction to ϕ. In addition to correcting the solutions once the mismatch is detected, we add a correction term to later level solve solutions in order to minimize the magnitude of the correction that will be needed.

Multilevel algorithm. At the start of a calculation, we perform a composite solve from level 0 through $\ell_{\rm finest}$ to compute ϕ at all levels. In addition, after every regridding step that creates new grids at level $\ell + 1$ and higher, a composite solve from level ℓ through $\ell_{\rm finest}$ is used to compute ϕ at those levels.

Following an approach similar to that described in Miniati & Colella (2007), at the start and end of each level ℓ time step we

perform a level solve to compute ϕ^ℓ. The difference between $\phi_\ell^{\rm comp}$ and ϕ^ℓ at the start of the time step is stored in $\phi^{\ell,{\rm corr}}$. This difference is added to ϕ^ℓ at the beginning *and* end of this level ℓ time step. Thus, $\phi^\ell + \phi^{\ell,{\rm corr}}$ is identical to $\phi_\ell^{\rm comp}$ at the start of the time step; at the end of the time step it is an approximation to what the solution to the composite solve would be. In the event that the density does not change over the course of the time step, the effect of this lagged correction is to make $\phi^\ell + \phi^{\ell,{\rm corr}}$ at the end of the time step identical to $\phi_\ell^{\rm comp}$ at that time, thus there is no mismatch between levels to correct. In general, when the density is not constant, the effect of the lagged correction is to make the correction solve that follows the end of the time step much quicker. We now describe the two-level correction step. In the discussion below, we will refer to the two levels involved in a correction solve as the "coarse" and "fine" levels.

At the end of r_ℓ level $\ell+1$ time steps, when the level $\ell+1$ solution has reached the same point in time as the level ℓ solution, and after the volume averaging and refluxing steps above have been performed, we define two quantities on the coarse grid. The first is the cell-centered quantity, $(\delta\rho)^c$, which carries the change in density at the coarse level due only to refluxing. The second is the face-centered flux register,

$$\delta F_\phi^\ell = -A^c \frac{\partial \phi^c}{\partial n} + \sum A^f \frac{\partial \phi^f}{\partial n}, \tag{36}$$

which accounts for the mismatch in the normal gradient of ϕ at coarse–fine interfaces. Here, A^c and A^f represent area weighting factors on the coarse and fine levels, respectively. We define the composite residual, $R^{\rm comp}$, to be zero in all fine cells and in all coarse cells away from the union of fine grids, and

$$R^{\rm comp} = 4\pi G(\delta\rho)^c - (\nabla\cdot \delta F_\phi)|^c, \tag{37}$$

on all cells adjacent to the union of fine grids, where $(\nabla\cdot)|^c$ refers to the discrete divergence at the coarse level, where the only non-zero contribution comes from δF_ϕ on the coarse–fine interface. We then solve

$$L^{\rm comp}_{\ell,\ell+1}\ \delta\phi = R^{\rm comp} \tag{38}$$

and define the update to gravity at both levels,

$$\delta \mathbf{g} = -\nabla(\delta\phi). \tag{39}$$

This update is used to correct the gravitational source terms. We define the new-time state after volume averaging but before refluxing as $(\overline{\rho}, \overline{\mathbf{u}}, \overline{\rho E}, \ldots)$, and the contributions to the solution on the coarse grid from refluxing as $((\delta\rho)^c, \delta(\rho\mathbf{u})^c, \delta(\rho E)^c, \ldots)$. Then we can define the sync sources for momentum on the coarse and fine levels, $S_{\rho\mathbf{u}}^{{\rm sync},c}$, and $S_{\rho\mathbf{u}}^{{\rm sync},f}$, respectively as follows:

$$\begin{aligned} S_{\rho\mathbf{u}}^{{\rm sync},c} &= (\overline{\rho}^c + (\delta\rho)^c)(\mathbf{g}^{c,n+1} + \delta\mathbf{g}^c) - \overline{\rho}^c\ \mathbf{g}^{c,n+1} \\ &= [(\delta\rho)^c \mathbf{g}^{c,n+1} + (\overline{\rho}^c + (\delta\rho)^c)\ \delta\mathbf{g}^c)], \\ S_{\rho\mathbf{u}}^{{\rm sync},f} &= \overline{\rho}^f\ \delta\mathbf{g}^f. \end{aligned}$$

These momentum sources lead to the following energy sources:

$$\begin{aligned} S_{\rho E}^{{\rm sync},c} &= S_{\rho\mathbf{u}}^{{\rm sync},c} \cdot \left(\overline{\mathbf{u}}^c + 1/2\ \Delta t_c\ S_{\rho\mathbf{u}}^{{\rm sync},c}/\overline{\rho}^c\right), \\ S_{\rho E}^{{\rm sync},f} &= S_{\rho\mathbf{u}}^{{\rm sync},f} \cdot \left(\overline{\mathbf{u}}^f + 1/2\ \Delta t_f\ S_{\rho\mathbf{u}}^{{\rm sync},f}/\overline{\rho}^f\right). \end{aligned}$$

The state at the coarse and fine levels is then updated using

$$\begin{aligned} (\rho\mathbf{u})^{c,n+1} &= (\rho\mathbf{u})^c + \delta(\rho\mathbf{u})^c + 1/2\Delta t_c S_{\rho\mathbf{u}}^{{\rm sync},c}, \\ (\rho\mathbf{u})^{f,n+1} &= (\overline{\rho\mathbf{u}})^f + 1/2\Delta t_f S_{\rho\mathbf{u}}^{{\rm sync},f}, \\ (\rho E)^{c,n+1} &= (\rho E)^c + \delta(\rho E)^c + 1/2\Delta t_c S_{\rho E}^{{\rm sync},c}, \\ (\rho E)^{f,n+1} &= (\overline{\rho E})^f + 1/2\Delta t_f S_{\rho E}^{{\rm sync},f}. \end{aligned}$$

(The factor of 1/2 follows from the time-centering of the sources.)

To complete the correction step,

1. We add $\delta\phi$ directly to ϕ^ℓ and $\phi^{\ell+1}$ and interpolate $\delta\phi$ to any finer levels and add it to the current ϕ at those levels. We note that at this point ϕ at levels ℓ and $\ell+1$ is identical to the solution that would have been computed using a two-level composite solve with the current values of density. Thus the new, corrected, ϕ at each level plays the role of $\phi^{\rm comp}$ in the next time step.
2. If level $\ell > 0$, we transmit the effect of this change in ϕ to the coarser levels by updating the flux register between level ℓ and level $\ell-1$. In particular, we set

$$\delta F_\phi{}^{\ell-1} = \delta F_\phi{}^{\ell-1} + \sum A^c \frac{\partial(\delta\phi)^{c-f}}{\partial n}. \tag{40}$$

Performance issues. The multilevel algorithm is not as computationally expensive as it might appear. Because multigrid is an iterative solver, the cost of each solve is proportional to the number of V-cycles, which is a function of the desired reduction in residual. We can reduce the number of V-cycles needed in two ways. First, we can supply a good initial guess for the solution; second, we can lower the desired reduction in residual.

In the case of level solves, we always use ϕ from a previous level solve, when available, as a guess in the current level solve. Thus, even in a single-level calculation, ϕ from the beginning of the time step is used as a guess in the level solve at the end of the time step. If no regridding occurs, then ϕ at the end of one time step can be used as a guess for the level solve at the start of the next time step. The extent to which ρ changes in a time step dictates the extent to which a new computation of gravity is needed, but this also dictates the cost of the update.

Similarly, there is no point in solving for $\delta\phi$ to greater accuracy than we solve for ϕ. When we do the correction solve for $\delta\phi$, we require only that the residual be reduced to the magnitude of the final residual from the level solve, *not* that we reduce the correction residual by the same factor. Thus, if the right-hand side for the correction solve is already small, the cost of the correction solve will be significantly less than that of the initial level solve.

7. SOFTWARE DESIGN AND PARALLEL PERFORMANCE

7.1. Overview

CASTRO is implemented within the BoxLib framework, a hybrid C++/Fortran90 software system that provides support for the development of parallel structured-grid AMR applications. The basic parallelization strategy uses a hierarchical programming approach for multicore architectures based on both MPI and OpenMP. In the pure-MPI instantiation, at least one grid at each level is distributed to each core, and each core communicates with every other core using only MPI. In the hybrid

approach, where on each socket there are n cores which all access the same memory, we can instead have one larger grid per socket, with the work associated with that grid distributed among the n cores using OpenMP.

In BoxLib, memory management, flow control, parallel communications, and I/O are expressed in the C++ portions of the program. The numerically intensive portions of the computation, including the multigrid solvers, are handled in Fortran90. The fundamental parallel abstraction in both the C++ and the Fortran90 is the MultiFab, which holds the data on the union of grids at a level. A MultiFab is composed of FABs; each FAB is an array of data on a single grid. During each MultiFab operation the FABs composing that MultiFab are distributed among the cores. MultiFabs at each level of refinement are distributed independently. The software supports two data distribution schemes, as well as a dynamic switching scheme that decides which approach to use based on the number of grids at a level and the number of processors. The first scheme is based on a heuristic knapsack algorithm as described in Crutchfield (1991) and in Rendleman et al. (2000). The second is based on the use of a Morton-ordering space-filling curve.

Each processor contains *metadata* that is needed to fully specify the geometry and processor assignments of the MultiFabs. At a minimum, this requires the storage of an array of boxes specifying the index space region for each AMR level of refinement. One of the advantages of computing with fewer, larger grids in the hybrid OpenMP–MPI approach is that the size of the metadata is substantially reduced.

7.2. Parallel Output

Data for checkpoints and analysis are written in a self-describing format that consists of a directory for each time step written. Checkpoint directories contain all necessary data to restart the calculation from that time step. Plotfile directories contain data for postprocessing, visualization, and analytics, which can be read using `amrvis`, a customized visualization package developed at LBNL for visualizing data on AMR grids, or VisIt (VisIt User's Manual 2005). Within each checkpoint or plotfile directory is an ASCII header file and subdirectories for each AMR level. The header describes the AMR hierarchy, including number of levels, the grid boxes at each level, the problem size, refinement ratio between levels, step time, etc. Within each level directory are the MultiFab files for each AMR level. Checkpoint and plotfile directories are written at user-specified intervals.

For output, each processor writes its own data to the appropriate MultiFab files. The output streams are coordinated to allow only one processor to write to a file at one time and to try to maintain maximum performance by keeping the number of open data streams, which is set at run time, equal to the number of files being written. Data files typically contain data from multiple processors, so each processor writes data from its associated grid(s) to one file, then another processor can write data from its associated grid(s) to that file. A designated I/O Processor writes the header files and coordinates which processors are allowed to write to which files and when. The only communication between processors is for signaling when processors can start writing and for the exchange of header information. We also use the C++ `setbuf` function for good single file performance. While I/O performance even during a single run can be erratic, recent timings on the Franklin machine (XT4) at NERSC indicate that CASTRO's I/O performance, when run with a single level composed of multiple uniformly sized grids, matches some of the top results for the N5 IOR benchmark (roughly 13 GB s^{-1}; Franklin Performance Monitoring 2010). For more realistic simulations with multiple grids at multiple levels, CASTRO is able to write data at approximately 5 GB s^{-1} sustained, over half of the average I/O benchmark reported speed.

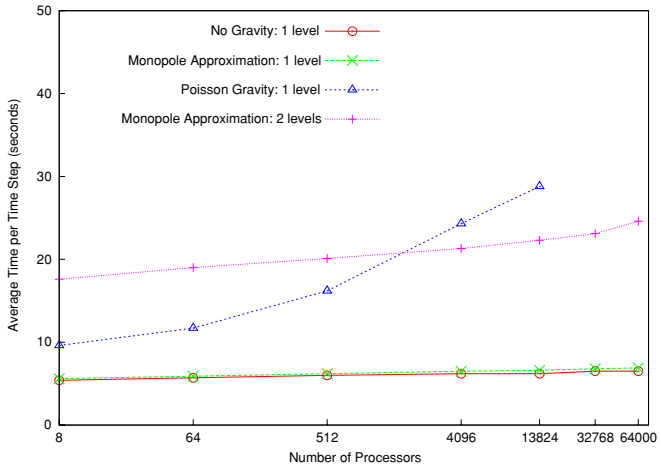

Figure 2. Weak scaling behavior of the CASTRO code on the jaguarpf machine at the OLCF. For the two-level simulation, the number of cells that are advanced in a time step increases by a factor of 3 because of subcycling. To quantify the overall performance, we note that for the 64,000 processor case without gravity, the time for a single core to advance one cell for one time step is 24.8 μs.

(A color version of this figure is available in the online journal.)

7.3. Parallel Restart

Restarting a calculation can present some difficult issues for reading data efficiently. In the worst case, all processors would need data from all files. If multiple processors try to read from the same file at the same time, performance problems can result, with extreme cases causing file system thrashing. Since the number of files is generally not equal to the number of processors and each processor may need data from multiple files, input during restart is coordinated to efficiently read the data. Each data file is only opened by one processor at a time. The IOProcessor creates a database for mapping files to processors, coordinates the read queues, and interleaves reading its own data. Each processor reads all data it needs from the file it currently has open. The code tries to maintain the number of input streams to be equal to the number of files at all times.

Checkpoint and plotfiles are portable to machines with a different byte ordering and precision from the machine that wrote the files. Byte order and precision translations are done automatically, if required, when the data are read.

7.4. Parallel Performance

In Figure 2, we show the scaling behavior of the CASTRO code, using only MPI-based parallelism, on the jaguarpf machine at the Oak Ridge Leadership Computing Facility (OLCF). A weak scaling study was performed, so that for each run there was exactly one 64^3 grid per processor. We ran the code with gravity turned off, with the monopole approximation to gravity, and with the Poisson solve for gravity. The monopole approximation to gravity adds very little to the run time of the code; with and without the monopole approximation the code scales excellently from 8 to 64,000 processors. For the 64,000 processor case without gravity, the time for a single core to advance one cell for one time step is 24.8 μs.

Good scaling of linear solves is known to be much more difficult to achieve; we report relatively good scaling up to only 13,824 processors in the pure-MPI approach. An early strong scaling study contrasting the pure-MPI and the hybrid-MPI–OpenMP approaches for a 768^3 domain shows that one can achieve at least a factor of 3 improvement in linear solver time by using the hybrid approach at large numbers of processors. Improving the performance of the linear solves on the new multicore architectures is an area of active research; more extensive development and testing is underway.

We also ran a scaling study with a single level of local refinement using the monopole gravity approximation. In this MPI-only study, there is one 64^3 grid at each level for each processor. Because of subcycling in time, a coarse time step consists of a single step on the coarse grid and two steps on the fine grid. Thus, we would expect that the time to advance the multilevel solution by one coarse time step would be a factor of 3 greater than the time to advance the single-level coarse solution by one coarse time step, plus any additional overhead associated with AMR. From the data in the figure, we conclude that AMR introduces a modest overhead, ranging from approximately 5% for the 8 processor case to 19% for the 64,000 processor case. By contrast, advancing a single-level calculation at the finer resolution by the same total time, i.e., two fine time steps, would require a factor of 16 more resources than advancing the coarse single-level solution.

8. TEST PROBLEMS

In this section, we present a series of calculations demonstrating the behavior of the hydrodynamics, self-gravity, and reaction components of CASTRO. The first set contains three one-dimensional shock-tube problems, including Sod's problem, a double rarefaction problem, and a strong shock problem. We follow this with Sedov–Taylor blast waves computed in one-dimensional spherical coordinates, two-dimensional cylindrical and Cartesian coordinates, and three-dimensional Cartesian coordinates. Our final pure-hydrodynamics test is a two-dimensional Rayleigh–Taylor (R–T) instability. We use this problem to contrast the differences in the flow found using dimensionally split and unsplit methods with piecewise linear, PPM with the old limiters, and PPM with the new limiters.

We then present two examples that test the interaction of the self-gravity solvers with the hydrodynamics in three-dimensional Cartesian coordinates. In the first case, a star is initialized in hydrostatic equilibrium and we monitor the maximum velocities that develop; in the second, the homologous dust collapse test problem, a uniform-density sphere is initialized at a constant low pressure, and collapses under its own self-gravity. These tests more closely examine the three-dimensional spherical behavior we expect to be present in simulations of Type Ia and Type II supernovae.

We perform a test of the coupling of the hydrodynamics to reactions. This test consists of a set of buoyant reacting bubbles in a stratified stellar atmosphere. We compare the CASTRO results to those of the FLASH code.

Finally, we note that a previous comparison of CASTRO to our low Mach number hydrodynamics code, MAESTRO, can be found in Nonaka et al. (2010). In that test, we took a one-dimensional spherical, self-gravitating stellar model and watched it hydrostatically adjust as we dumped energy into the center of the star. The resulting temperature, pressure, and density profiles agreed very well between the two codes.

8.1. Shock-tube Problems

To test the behavior of the hydrodynamics solver, we run several different one-dimensional shock-tube problems. The setup for these problems consists of a left and right state, with the interface in the center of the domain. All calculations use a gamma-law EOS with $\gamma = 1.4$. We show results from each problem run using one-dimensional Cartesian coordinates, but we have verified that the results are identical when each problem is run in two-dimensional or three-dimensional Cartesian coordinates and the interface is normal to a coordinate axis. The length of the domain is always taken as 1.0, with the interface in the center. We use a base grid of 32 cells, with two additional levels of factor 2 refinement, for an effective resolution of 128 cells. The refinement criteria are based on gradients of density and velocity. In the case of the double rarefaction, we also present results from runs with two levels of factor 4 refinement (effective resolution of 512 cells) and three levels of factor 4 refinement (effective resolution of 2048 cells). In each case, analytic solutions are found using the exact Riemann solver from Toro (1997). All calculations are run with the new PPM limiters and a CFL number of 0.9. For each problem, we show density, pressure, velocity, and internal energy.

8.1.1. Sod's Problem

Sod's problem (Sod 1978) is a simple shock-tube problem that exhibits a shock, contact discontinuity, and a rarefaction wave. The non-dimensionalized initial conditions are

$$\begin{array}{ll} \rho_L = 1 & \rho_R = 0.125, \\ u_L = 0 & u_R = 0, \\ p_L = 1 & p_R = 0.1. \end{array} \tag{41}$$

This results in a rightward moving shock and contact discontinuity, and a leftward moving rarefaction wave. Figure 3 shows the resulting pressure, density, velocity, and internal energy at $t = 0.2$ s. We see excellent agreement with the exact solution.

8.1.2. Double Rarefaction

The double rarefaction problem tests the behavior of the hydrodynamics algorithm in regions where a vacuum is created. We run the problem as described in Toro (1997). The non-dimensionalized initial conditions are

$$\begin{array}{ll} \rho_L = 1 & \rho_R = 1, \\ u_L = -2 & u_R = 2, \\ p_L = 0.4 & p_R = 0.4. \end{array} \tag{42}$$

This results in two rarefaction waves propagating in opposite directions away from the center. As a result, matter is evacuated from the center, leaving behind a vacuum. Figure 4 shows the CASTRO solutions at $t = 0.15$ s. The agreement with the exact solution is excellent at the 128-cell resolution for density, pressure, and velocity; the internal energy is more sensitive, but clearly converges to the analytic solution except at the center line. This is a very common pathology for this problem, since the internal energy, e, is derived from Equation (7) using values of p and ρ which are both approaching zero in the center of the domain (Toro 1997).

8.1.3. Strong Shock

The final shock-tube problem we try is a strong shock. We initialize the problem as described in Toro (1997). The initial

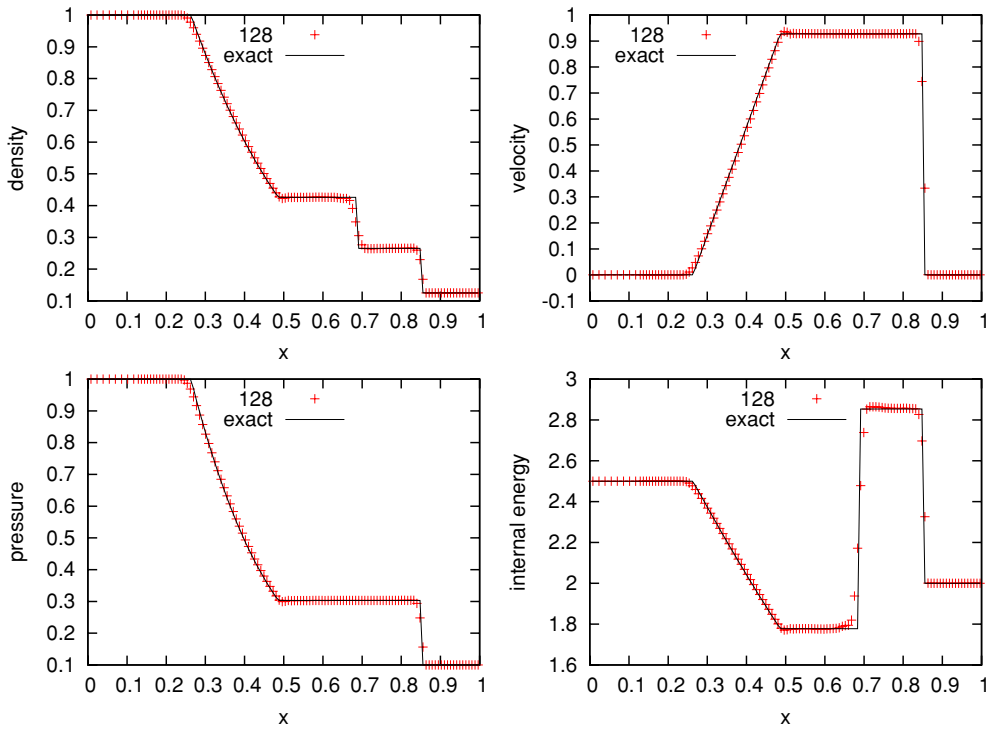

Figure 3. Adaptive CASTRO solution vs. analytic solution for Sod's problem run in one dimension at an effective resolution of 128 cells.
(A color version of this figure is available in the online journal.)

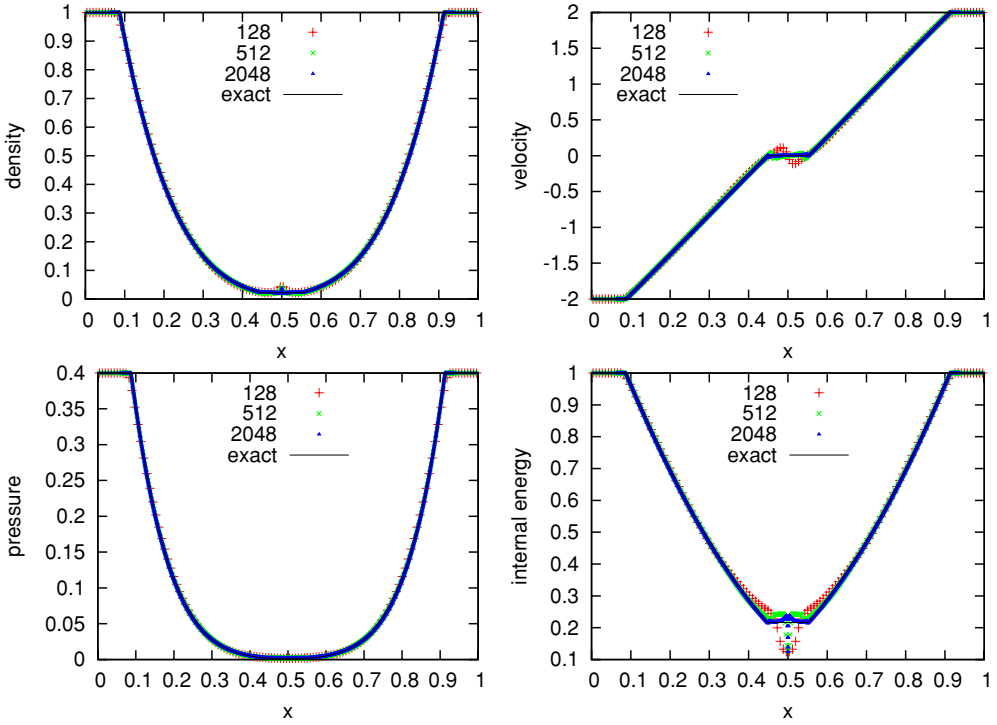

Figure 4. Adaptive CASTRO solutions vs. analytic solution for the double rarefaction problem run in one dimension at effective resolutions of 128, 512, and 2048 cells.
(A color version of this figure is available in the online journal.)

conditions are

$$\begin{array}{ll} \rho_L = 1 & \rho_R = 1, \\ u_L = 0 & u_R = 0, \\ p_L = 1000 & p_R = 0.01. \end{array} \tag{43}$$

The initial pressure jump of 6 orders of magnitude results in a strong rightward moving shock. This large dynamic range can cause trouble for some hydrodynamics solvers. The shock is followed very closely by a contact discontinuity. A leftward moving rarefaction is also present. Figure 5 shows the CASTRO

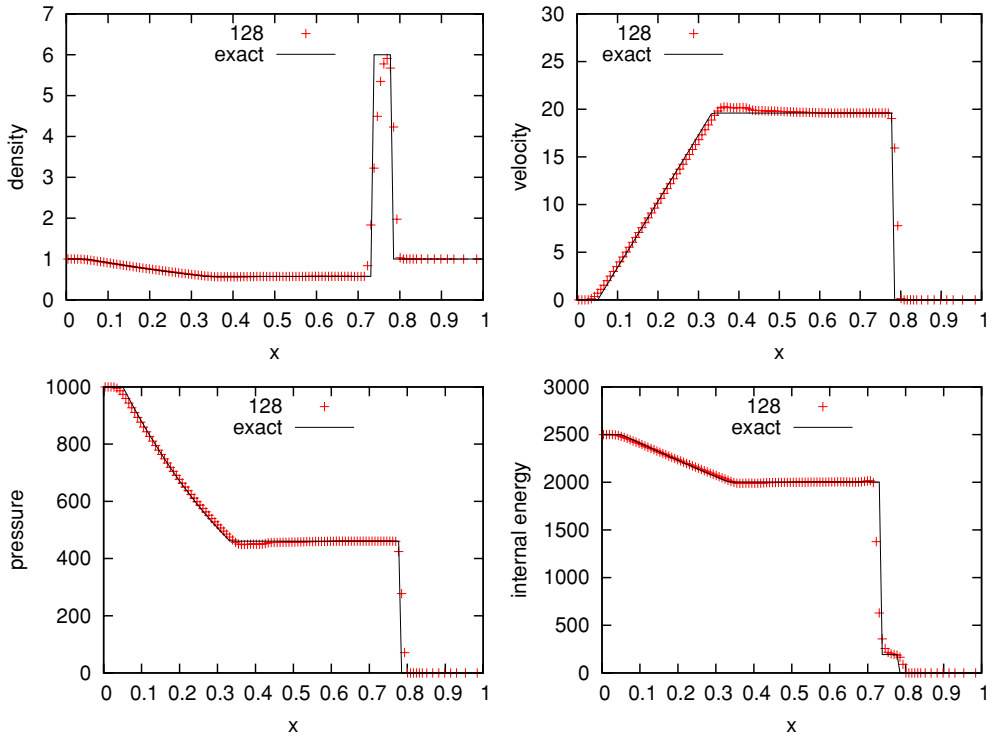

Figure 5. Adaptive CASTRO solution vs. analytic solution for the strong shock problem run in one dimension at an effective resolution of 128 cells.
(A color version of this figure is available in the online journal.)

results at $t = 0.012$ s. We see good agreement between the CASTRO results and the exact solution.

8.2. Sedov

Another standard hydrodynamics test is the Sedov–Taylor blast wave. The problem setup is very simple: a large amount of energy is deposited into the center of a uniform domain. This drives a blast wave (spherical or cylindrical, depending on the domain geometry). An analytic solution is provided by Sedov (1959). We use a publicly available code described by Kamm & Timmes (2007) to generate the exact solutions.

The Sedov explosion can test the geometrical factors in the hydrodynamics scheme. A cylindrical blast wave (e.g., a point explosion in a two-dimensional plane) can be modeled in two-dimensional Cartesian coordinates. A spherical blast wave can be modeled in one-dimensional spherical, two-dimensional axisymmetric (cylindrical r–z), or three-dimensional Cartesian coordinates.

In the Sedov problem, the explosion energy, $\mathcal{E}_{\rm exp}$ (in units of energy, not energy/mass or energy/volume), is deposited into a single point, in a medium of uniform ambient density, $\rho_{\rm ambient}$, and pressure, $p_{\rm ambient}$. Initializing the problem can be difficult because the small volume is typically only one cell in extent, which can lead to grid imprinting in the solution. A standard approach (see, for example, Fryxell et al. 2000, Omang et al. 2006, and the references therein) is to convert the explosion energy into a pressure contained within a certain volume, $V_{\rm init}$, of radius $r_{\rm init}$ as

$$p_{\rm init} = \frac{(\gamma - 1)\mathcal{E}_{\rm exp}}{V_{\rm init}}. \quad (44)$$

This pressure is then initialized to $p_{\rm init}$ in all of the cells where $r < r_{\rm init}$. We use the gamma-law EOS with $\gamma = 1.4$.

To further minimize any grid effects, we do subsampling in each cell: each cell is divided it into $N_{\rm sub}$ subcells in each coordinate direction, each subcell is initialized independently, and then the subcells are averaged together (using volume weighting for spherical or cylindrical coordinates) to determine the initial state of the full cell.

For these runs, we use $\rho_{\rm ambient} = 1\,{\rm g\,cm^{-3}}$, $p_{\rm ambient} = 10^{-5}\,{\rm dyn\,cm^{-2}}$, $\mathcal{E}_{\rm exp} = 1$ erg, $r_{\rm init} = 0.01$ cm, and $N_{\rm sub} = 10$. A base grid with $\Delta x = 0.03125$ cm is used with three levels of factor 2 refinement. For most geometries, we model the explosion in a domain ranging from 0 to 1 cm in each coordinate direction. In this case, the base grid would have 32 cells in each coordinate direction and the finest mesh would correspond to 256 cells in each coordinate direction. For the two-dimensional axisymmetric case, we model only one quadrant, and the domain ranges from 0 to 0.5 cm. All calculations were run with a CFL number of 0.5, and the initial time step was shrunk by a factor of 100 to allow the point explosion to develop. We refine on regions where $\rho > 3\,{\rm g\,cm^{-3}}$, $\nabla\rho > 0.01\,{\rm g\,cm^{-3}\,cm^{-1}}$, $p > 3\,{\rm dyn\,cm^{-2}}$, or $\nabla p > 0.01\,{\rm dyn\,cm^{-2}\,cm^{-1}}$.

Figure 6 shows the CASTRO solution to a spherical Sedov explosion at time $t = 0.01$ s, run in one-dimensional spherical, two-dimensional cylindrical, and three-dimensional Cartesian coordinates. For the two-dimensional and three-dimensional solutions, we compute the radial profile by mapping each cell into its corresponding radial bin and averaging. The radial bin width was picked to match the width of a cell at the finest level of refinement in the CASTRO solution. The density, velocity, and pressure plots match the exact solution well. As with the double rarefaction problem, the internal energy is again the most difficult quantity to match due to the vacuum region created at the origin. Figure 7 shows the same set of calculations run with four levels of factor 2 refinement. Here, the agreement is even better. Figure 8 shows the CASTRO solution at time $t = 0.1$ s to a cylindrical Sedov explosion, run in two-dimensional Cartesian coordinates.

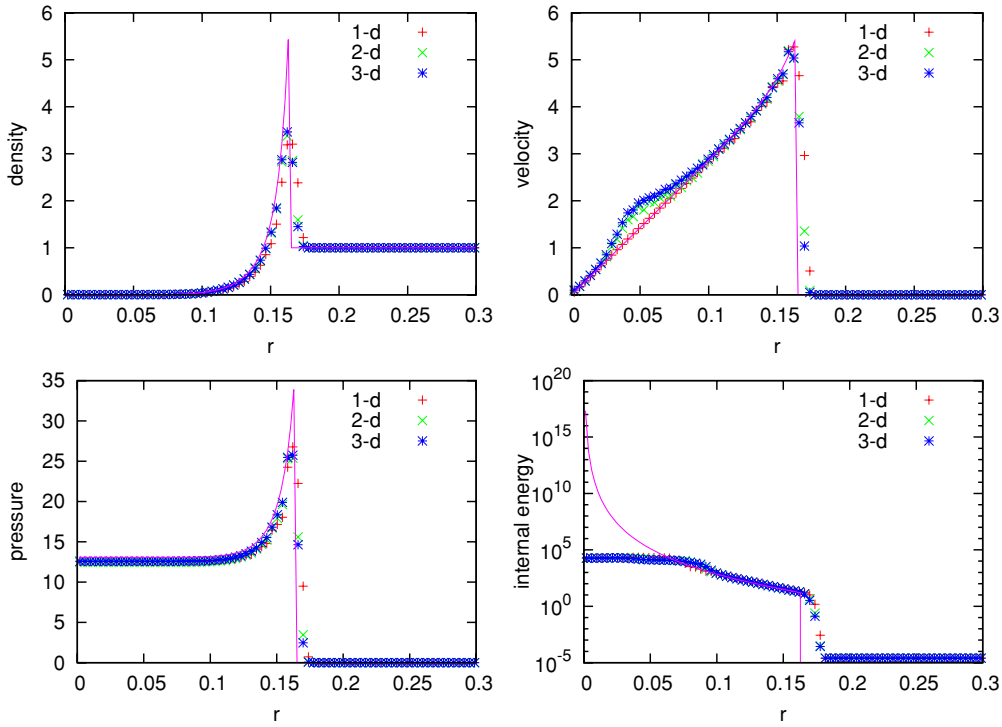

Figure 6. CASTRO solution at $t = 0.01$ s for the spherical Sedov blast wave problem run in one-dimensional spherical, two-dimensional axisymmetric, and three-dimensional Cartesian coordinates. This was run with a base grid with $\Delta x = 0.03125$ cm and three levels of factor 2 refinement for an effective resolution of $\Delta x = 0.00390625$ cm.

(A color version of this figure is available in the online journal.)

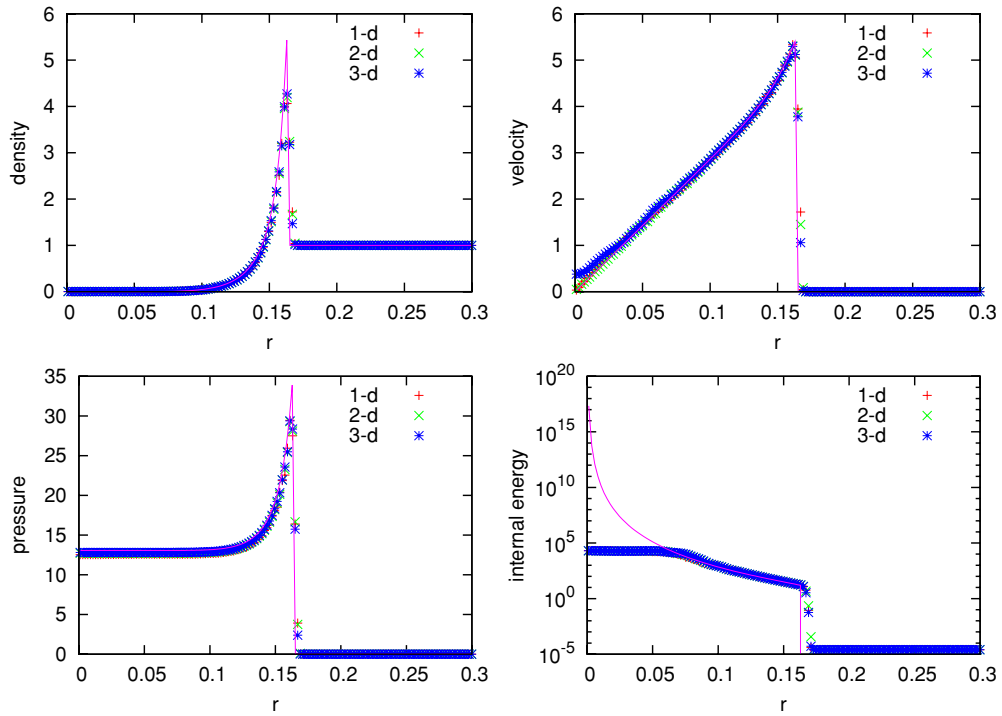

Figure 7. CASTRO solution at $t = 0.01$ s for the spherical Sedov blast wave problem run in one-dimensional spherical, two-dimensional axisymmetric, and three-dimensional Cartesian coordinates. This was run with a base grid with $\Delta x = 0.03125$ cm and four levels of factor 2 refinement for an effective resolution of $\Delta x = 0.001953125$ cm.

(A color version of this figure is available in the online journal.)

8.3. Rayleigh–Taylor

The R–T instability results when a dense fluid is placed over a less-dense fluid in a gravitational field (Taylor 1950; Layzer 1955; Sharp 1984). The interface is unstable and a small perturbation will result in the growth a buoyant uprising bubbles and dense, falling spikes of fluid. This instability provides a mechanism for mixing in many astrophysical systems. Despite

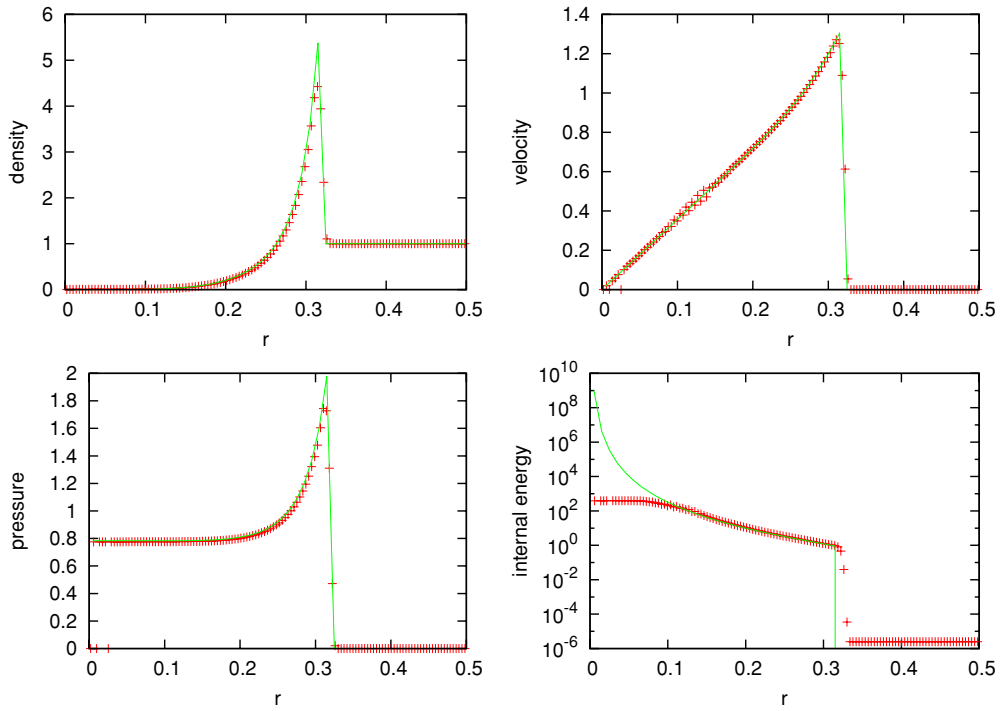

Figure 8. CASTRO solution at $t = 0.1$ s for the cylindrical Sedov blast wave problem run in two-dimensional Cartesian coordinates. This was run with a base grid with $\Delta x = 0.03125$ cm and three levels of factor 2 refinement for an effective resolution of $\Delta x = 0.00390625$ cm.

(A color version of this figure is available in the online journal.)

its seemingly simplistic nature, only the linear growth regime is understood analytically (see, for example, Chandrasekhar 1961). In the nonlinear regime, R–T instability calculations are often used as a means of code validation (Dimonte et al. 2004).

For our purposes, the R–T instability provides a good basis to compare different choices of the advection algorithm. We model a single-mode R–T instability—a perturbation consisting of a single wavelength that disturbs the initial interface. Short-wavelength perturbations have a faster growth rate than long-wavelength perturbations, so grid effects can easily drive the instability on smaller scales than our initial perturbation. No viscous terms are explicitly modeled.

We choose the density of the dense fluid to be $\rho_2 = 2\,\mathrm{g\,cm^{-3}}$ and the light fluid is $\rho_1 = 1\,\mathrm{g\,cm^{-3}}$. The gravitational acceleration is taken to be $g = -1\,\mathrm{cm\,s^{-2}}$ in the vertical direction. The gamma-law EOS is used with $\gamma = 1.4$. Our domain has a width of $L_x = 0.5$ cm and a height of $L_y = 1$ cm. The initial interface separating the high and low-density fluid is centered vertically at $L_y/2$, with the density in the top half taken to be ρ_2 and the density in the lower half ρ_1. Since g and ρ_1, ρ_2 are constant, we can analytically integrate the equation of hydrostatic equilibrium to get the pressure in both the high- and low-density regions of the domain:

$$p(y) = \begin{cases} p_{\mathrm{base}} + \rho_1 g y & y < L_y/2 \\ p_{\mathrm{base}} + \rho_1 g L_y/2 + \rho_2 g (y - L_y/2) & y > L_y/2, \end{cases} \tag{45}$$

where y is the vertical coordinate and p_{base} is the pressure at the base of the domain. We take $p_{\mathrm{base}} = 5\,\mathrm{dyn\,cm^{-2}}$.

To initiate the instability, the interface is perturbed by slightly shifting the density, keeping the interface centered vertically in the domain. We define the perturbed interface height, ψ, to be a function of position in the x-direction as

$$\psi(x) = \frac{A}{2}\left[\cos\left(\frac{2\pi x}{L_x}\right) + \cos\left(\frac{2\pi(L_x - x)}{L_x}\right)\right] + \frac{L_y}{2}, \tag{46}$$

with the amplitude $A = 0.01$ cm. We note that the cosine part of the perturbation is done symmetrically, to prevent roundoff error from introducing an asymmetry in the flow. The density is then perturbed as

$$\rho(x, y) = \rho_1 + \frac{\rho_2 - \rho_1}{2}\left[1 + \tanh\left(\frac{y - \psi(x)}{h}\right)\right]. \tag{47}$$

The tanh profile provides a slight smearing of the initial interface, over a smoothing length h. We take $h = 0.005$ cm.

In Figure 9, we present simulation results for the R–T problem at $t = 2.5$ s for several different variants of the hydrodynamics. All calculations were run with 256×512 grid cells. In the bottom right image, we show the results obtained using the unsplit PPM with the new limiter used in CASTRO. The left and middle images on the bottom row are results using the unsplit piecewise-linear method and unsplit PPM with limiters as in Miller & Colella (2002), respectively. The results with all three methods are reasonably good; however, the piecewise linear and original PPM limiter both exhibit mild anomalies at the tip of both the bubble and the spike.

In the upper row, we present results for the R–T problem using operator-split analogs of the unsplit methods. The details of the algorithms such as limiters, Riemann solver, etc., are the same as in the unsplit methods; the only difference is the use of operator splitting. We note that all three of the operator-split methods produce spurious secondary instabilities. This behavior is a direct result of the operator-split approach. Physically, for these low Mach number flows, the density field is advected

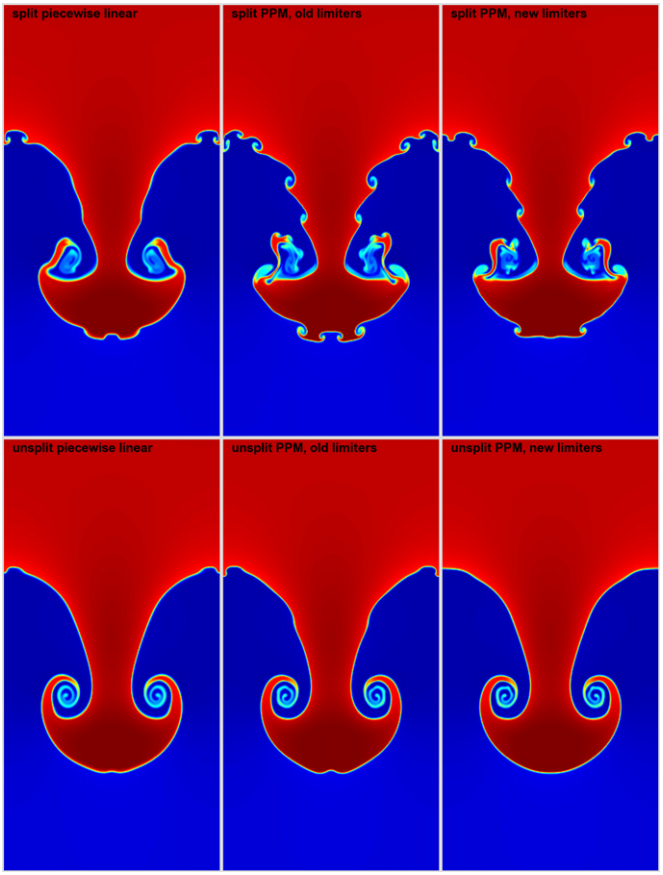

Figure 9. Density in a single-mode Rayleigh–Taylor simulation for a variety of advection schemes. Dimensionally split method results are shown on the top row; unsplit method results are shown on the bottom row. We see that the unsplit methods do better at suppressing the growth of high-wavenumber instabilities resulting from grid effects.

by a nearly incompressible flow field, and remains essentially unchanged along Lagrangian trajectories. However, in regions where there is significant variation in the local strain rate, an operator-split integration approach alternately compresses and expands the fluid between subsequent sweeps. This alternating compression/expansion provides the seed for the anomalies observed with operator-split methods.

We note that both the CPU time and the memory usage are roughly a factor of 2 larger for the unsplit algorithm than for the split algorithm in this two-dimensional implementation. For a pure-hydrodynamics problem with gamma-law EOS this factor is nontrivial; for a simulation that uses the full self-gravity solver, a realistic reaction network, a costly EOS, or significant additional physics, the additional cost of the hydrodynamic solver may be negligible.

In three dimensions, one might expect the ratio of CPU time for the unsplit algorithm relative to the split algorithm to be even larger than in two dimensions because of the additional Riemann solves required to construct the transverse terms. However, this effect is counterbalanced by the need to advance ghost cells in the split algorithm to provide boundary conditions for subsequent sweeps. Consequently, we observe an increase in CPU time that is slightly less than the factor of 2 observed in two dimensions. The three-dimensional implementation of the unsplit algorithm in CASTRO uses a strip-mining approach that only stores extra data on a few planes at a time, so we see an increase of less than 10% in the memory required for the unsplit integrator compared to the split integrator in three dimensions.

8.4. Stationary Star Gravity

A challenging problem for a hydrodynamics code is to keep a star in hydrostatic equilibrium. Because of the different treatment of the pressure, density, and gravitational acceleration by the hydrodynamics algorithm, small motions can be driven by the inexact cancellation of ∇p and $\rho\mathbf{g}$. This is further exaggerated by modeling a spherical star on a three-dimensional Cartesian grid. Here, we test the ability of CASTRO to maintain hydrostatic equilibrium for a spherical, self-gravitating star.

Our initial model is a nearly Chandrasekhar mass, carbon–oxygen white dwarf, which is generated by specifying a core density (2.6×10^9 g cm^{-3}), temperature (6×10^8 K), and a uniform composition ($X(^{12}\mathrm{C}) = 0.3$, $X(^{16}\mathrm{O}) = 0.7$) and integrating the equation of hydrostatic equilibrium outward while constraining the specific entropy, s, to be constant. In discrete form, we solve

$$p_{0,j+1} = p_{0,j} + \frac{1}{2}\Delta r(\rho_{0,j} + \rho_{0,j+1})g_{j+1/2}, \tag{48}$$

$$s_{0,j+1} = s_{0,j}, \tag{49}$$

with $\Delta r = 1.653125 \times 10^5$ cm. We begin with a guess of $\rho_{0,j+1}$ and $T_{0,j+1}$ and use the EOS and Newton–Raphson iterations to find the values that satisfy our system. Since this is a spherical, self-gravitating star, the gravitation acceleration, $g_{j+1/2}$, is updated each iteration based on the current value of the density. Once the temperature falls below 10^7 K, we keep the temperature constant, and continue determining the density via hydrostatic equilibrium until the density falls to 10^{-4} g cm^{-3}, after which we hold the density constant. This uniquely determines the initial model. We note that this is the same procedure we follow to initialize a convecting white dwarf for the multilevel low Mach number code, MAESTRO, described in Nonaka et al. (2010).

We map the model onto a $(5 \times 10^8$ cm$)^3$ domain with 192^3, 384^3, and 768^3 grid cells, and center the star in the domain. We let the simulation run to 1 s, and compare the maximum magnitude of velocity versus time and the magnitude of velocity versus radius at $t = 1$ s, a time greater than two sound-crossing times. We only consider regions of the star at $r < 1.8 \times 10^8$ cm, which corresponds to a density of $\rho \approx 5.4 \times 10^5$ g cm^{-3}. Note that the density reaches the floor of 10^{-4} g cm^{-3} at $r = 1.9 \times 10^8$ cm. We turn on the sponge at the radius where $\rho = 100$ g cm^{-3} and the sponge reaches its full strength at the radius where $\rho = 10^{-4}$ g cm^{-3} with a sponge strength of $\kappa = 1000\,\mathrm{s}^{-1}$. We use a CFL of 0.9 and no refinement. We use the Helmholtz EOS (Timmes & Swesty 2000; Fryxell et al. 2000) and no reactions are modeled.

Figure 10 shows a plot of the maximum magnitude of velocity versus time. At each of the three resolutions, we show the results using a monopole gravity approximation and Poisson solve for gravity. We note that in each simulation, the maximum velocity is not strictly increasing, leading us to believe that over longer periods of time the velocities will remain small. We note that sound speed at the center of the star is approximately 9.4×10^8 cm s^{-1}, so at the highest resolution, the peak velocity is less than 1% of the sound speed. The monopole and Poisson cases match up very well, except for the finest resolution. The reason why we see larger peak velocities in the finest resolution Poisson solver simulation is due to the large velocities at the edge of the star.

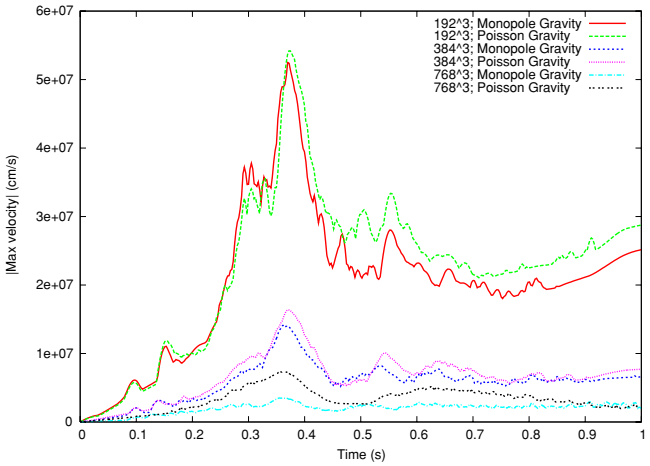

Figure 10. Maximum magnitude of velocity vs. time for the stationary star gravity problem. At each of the three resolutions, we show the results using a monopole gravity approximation and Poisson solve for gravity. We note that in each simulation, the maximum velocity is not strictly increasing, leading us to believe that over longer periods of time the velocities will remain small. We note that sound speed at the center of the star is approximately 9.4×10^8 cm s^{-1}, so at the highest resolution, the peak velocity is less than 1% of the sound speed. The solutions in the monopole and Poisson cases match up very well; the discrepancy we see at the finest resolution is due to large velocities at the edge of the star, which is typically outside the region of interest.

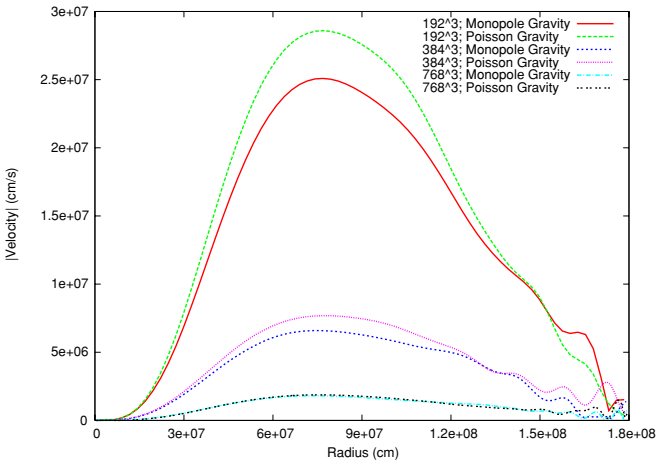

Figure 11. Magnitude of velocity vs. radius at $t = 1$ s for the stationary star gravity problem. At each of the three resolutions, we show the results using a monopole gravity approximation and Poisson solve for gravity. Here, we see clear second-order convergence in the max norm, and the monopole and Poisson simulations agree best at the highest resolution.

Figure 11 shows a plot of the magnitude of velocity versus radius at $t = 1$ s. Again, at each of the three resolutions, we show the results using a monopole gravity approximation and Poisson solve for gravity. Here, we see clear second-order convergence in the max norm, and the monopole and Poisson simulations agree best at the highest resolution. We also see how in the finest resolution runs, the velocities at the edge of the star can become large, but this is likely outside the region of interest for a typical simulation.

8.5. Homologous Dust Collapse

As a second test of the gravity solver in CASTRO, we implement the homologous dust collapse test problem, a "pressureless" configuration that collapses under its own self-gravity. An analytic solution that describes the radius of the sphere as a function of time is found in Colgate & White (1966). Our implementation of this problem follows that described in FLASH 3.2 User's Guide (2009) and Monchmeyer & Muller (1989). The problem is initialized with a sphere with a large, uniform density, ρ_0, of radius r_0. The pressure everywhere should be negligible, i.e., the sound-crossing time should be much longer than the free-fall collapse time (see, for example, FLASH 3.2 User's Guide 2009). Colgate & White (1966) use $p = 0$. We choose a value that does not appear to affect the dynamics. As the sphere collapses, the density inside should remain spatially constant, but increase in value with time.

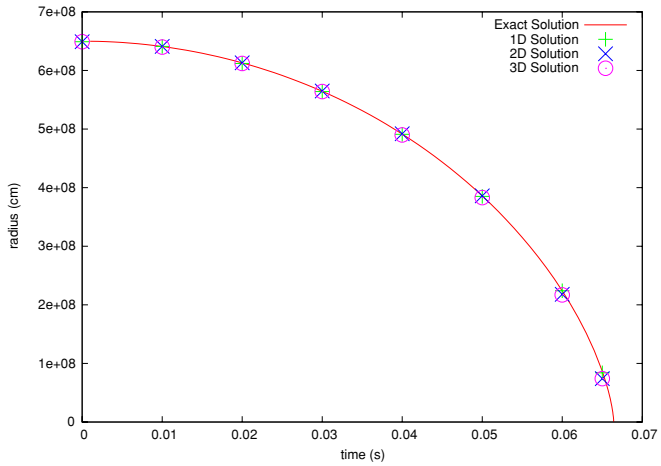

Figure 12. Radius vs. time for the homologous dust collapse problem in one-dimensional, two-dimensional, and three-dimensional simulations as compared to the exact solution. In all three cases, we see excellent agreement with the exact solution.

Following FLASH 3.2 User's Guide (2009), we take $\rho_0 = 10^9$ g cm^{-3} and $r_0 = 6.5 \times 10^8$ cm. The pressure is not specified, so we take it to be 10^{15} dyn cm^{-2}. Outside of the sphere, we set the density to $\rho_{\rm ambient} = 10^{-5}$ g cm^{-3}. Finally, since the sharp cutoff at the edge of the sphere is unphysical, we smooth the initial profile by setting

$$\rho = \rho_0 - \frac{\rho_0 - \rho_{\rm ambient}}{2}\left[1 + \tanh\left(\frac{r - r_0}{h}\right)\right] \qquad (50)$$

with the smoothing length, $h = 4 \times 10^6 \ll r_0$. We use the gamma-law EOS with $\gamma = 1.66$.

Figure 12 shows the radius versus time for the one-dimensional, two-dimensional, and three-dimensional simulations as compared to the exact solution. In all three cases, we see excellent agreement with the exact solution.

8.6. Reacting Bubbles in a Stellar Atmosphere

A final test is a code comparison of the evolution of three reacting bubbles in a plane-parallel stellar atmosphere. This problem is almost identical to the setup described in Section 4.2 of Almgren et al. (2008) with two minor differences. First, we eliminate the stably stratified layer at the base of the atmosphere by setting the lower y extrema of the domain to 5.00625×10^7 cm—this way, the bottom-most row of cells in the domain is initialized with the specified base density (2.6×10^9 g cm^{-3}) and temperature. Second, we set the base temperature of the atmosphere to 6×10^8 K (instead of 7×10^8 K) to minimize the amount of reactions occurring near the lower domain boundary. Three temperature perturbations are seeded in pressure equilibrium with a range of heights and widths as

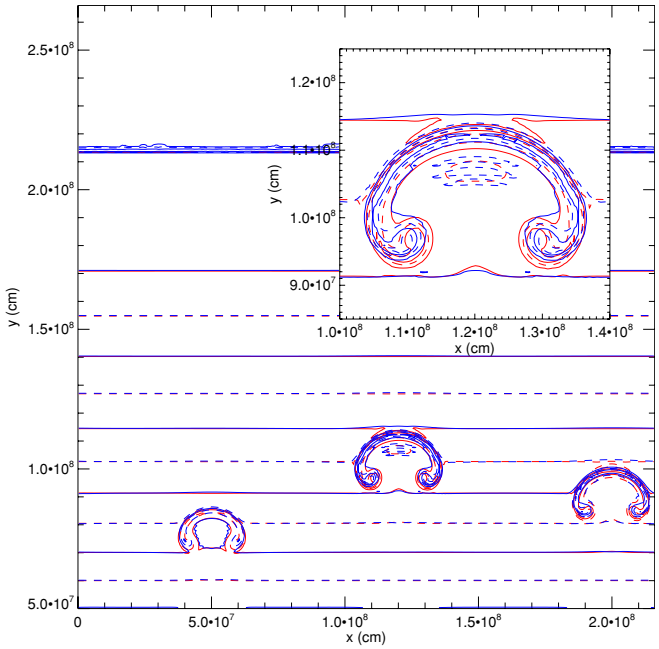

Figure 13. Comparison of FLASH (red) and CASTRO (blue) temperature contours for the reacting bubble test. Temperature contours at 10^8, 1.5×10^8, 2×10^8, 2.5×10^8, 3×10^8, 3.5×10^8, 4×10^8, 4.5×10^8, 5×10^8, 5.5×10^8, 6×10^8, 6.5×10^8, 7×10^8, 7.5×10^8, and 8×10^8 K are shown, drawn with alternating solid and dashed lines. The inset shows the detail of the middle bubble. We see good agreement between FLASH and CASTRO.

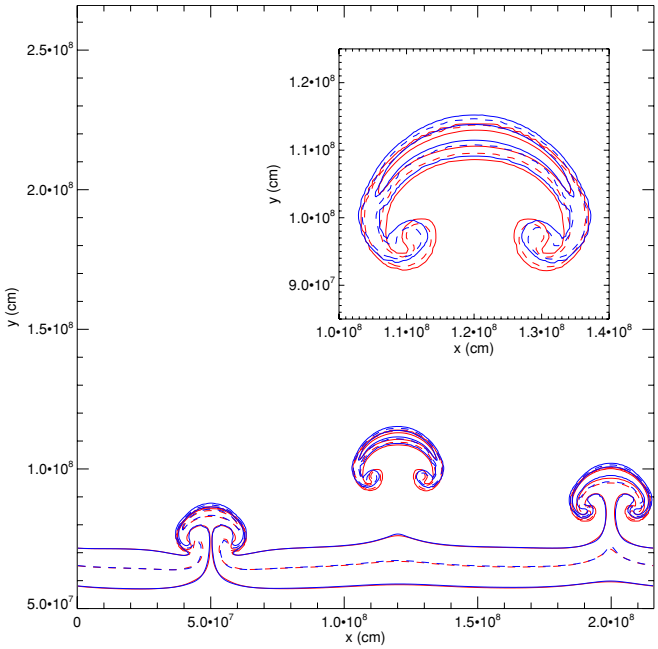

Figure 14. Comparison of FLASH (red) and CASTRO (blue) ^{24}Mg mass fraction contours for the reacting bubble test. Contours are drawn at values of X of 5×10^{-9}, 5×10^{-8}, 5×10^{-7}, and 5×10^{-6}, with alternating solid and dashed lines. The inset shows the detail of the middle bubble. As with the temperature, we see good agreement between FLASH and CASTRO.

specified by Equation (87) and Table 1 of Almgren et al. (2008). We use a uniform computation grid of 384×576 cells and a domain width of 2.16×10^8 cm.

We compare the evolution to the FLASH code (Fryxell et al. 2000), version 2.5, using the standard dimensionally split PPM hydrodynamics module that comes with FLASH. The lower boundary condition in both cases provides hydrostatic support by integrating the equation of hydrostatic equilibrium together with the equations of state into the ghost cells, assuming a constant temperature, as described in Zingale et al. (2002). The left and right boundary is periodic. We use the same single step (^{12}C + ^{12}C $\rightarrow$ ^{24}Mg) reaction module described in Almgren et al. (2008). Both codes use the general stellar EOS described in Fryxell et al. (2000) and Timmes & Swesty (2000) with the Coulomb corrections enabled.

Figures 13 and 14 show contours of the temperature and $X(^{24}\mathrm{Mg})$ after 2.5 s of evolution for both FLASH and CASTRO. We see excellent agreement between the two codes in terms of bubble heights and contour levels.

8.7. Type Ia Supernova

As a final example, in Figure 15 we show a two-dimensional snapshot of temperature from a three-dimensional calculation of a Type Ia supernova (H. Ma & A. J. Aspden 2010, private communication; Ma et al. 2010). This simulation uses a realistic stellar EOS and a turbulent flame model, and is typical of more realistic CASTRO applications. The domain is 5.12×10^8 cm on a side, and is covered with 512 64^3 grids. There are two levels of factor 2 refinement, with approximately 1.8% of the domain covered by level 2 grids with an effective resolution of 2.5×10^5 cm. Figure 16 is a close-up of the center of the domain so that the level 2 grids are more visible.

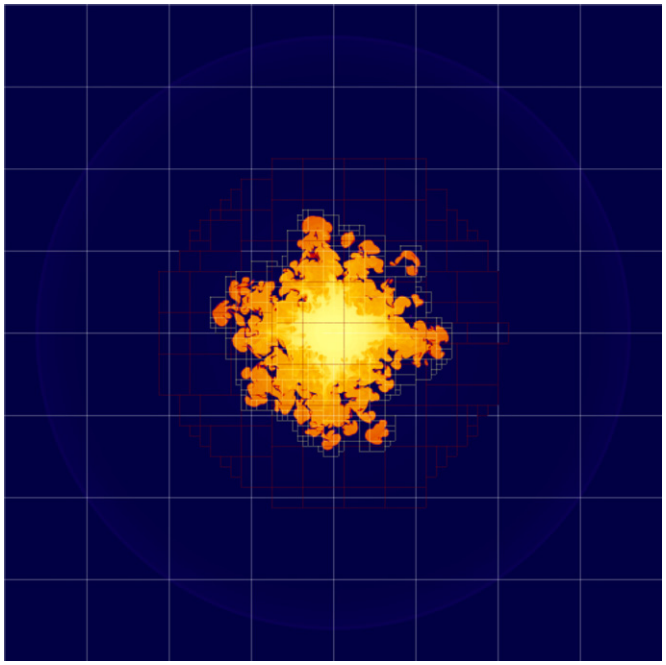

Figure 15. Two-dimensional slice of the temperature field from a three-dimensional calculation of a Type Ia supernova with two levels of refinement. There are 512 grids, each containing 64^3 cells, at the coarsest level, over 1000 grids at level 1 and over 2000 grids at level 2. Approximately 1.8% of the domain is at the finest resolution.

9. SUMMARY

We have described a new Eulerian adaptive mesh code, CASTRO, for solving the multicomponent compressible hydrodynamic equations with a general EOS for astrophysical flows. CASTRO differs from existing codes of its type in that it uses unsplit PPM for its hydrodynamic evolution, subcycling in time, and a nested hierarchy of logically rectangular grids. Additional

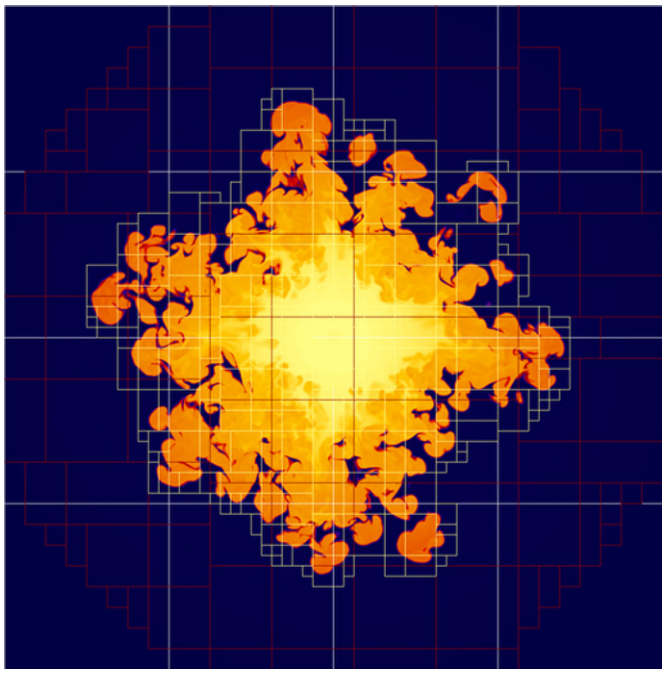

Figure 16. Close-up of the previous figure, showing more detail of the level 2 grids.

physics includes self-gravitation, nuclear reactions, and radiation. Radiation will be described in detail in the next paper, Part II, of this series.

CASTRO is currently being used in simulations of Type Ia supernovae and core-collapse supernovae; examples of simulations done using CASTRO can be found in Joggerst et al. (2009) and Woosley et al. (2009). Further details on the CASTRO algorithm can be found in the CASTRO User Guide (2009).

We thank Alan Calder for useful discussions on test problems and Stan Woosley for numerous invaluable interactions. In comparing to other codes, we benefited from helpful discussions with Brian O'Shea about Enzo, Paul Ricker about gravity in FLASH, and Michael Clover about RAGE. Finally, we thank Haitao Ma, Jason Nordhaus, and Ken Chen for being patient early users of CASTRO. The work at LBNL was supported by the Office of High Energy Physics and the Office of Mathematics, Information, and Computational Sciences as part of the SciDAC Program under the U.S. Department of Energy under contract No. DE-AC02-05CH11231. The work performed at LLNL was under the auspices of the U.S. Department of Energy under contract No. DE-AC52-07NA27344. M.Z. was supported by Lawrence Livermore National Lab under contracts B568673, B574691, and B582735. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the Oak Ridge Leadership Computational Facility (OLCF), which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

REFERENCES

Almgren, A. S., Bell, J. B., Colella, P., Howell, L. H., & Welcome, M. L. 1998, J. Comput. Phys., 142, 1
Almgren, A. S., Bell, J. B., Nonaka, A., & Zingale, M. 2008, ApJ, 684, 449
Bell, J., Berger, M., Saltzman, J., & Welcome, M. 1994, SIAM J. Sci. Statist. Comput., 15, 127
Bell, J. B., Colella, P., & Trangenstein, J. A. 1989, J. Comput. Phys., 82, 362
Berger, M. J., & Colella, P. 1989, J. Comput. Phys., 82, 64
Berger, M. J., & Oliger, J. 1984, J. Comput. Phys., 53, 484
Berger, M. J., & Rigoutsos, J. 1991, IEEE Trans. Syst. Man Cybern., 21, 1278
Bryan, G. L., Norman, M. L., Stone, J. M., Cen, R., & Ostriker, J. P. 1995, Comput. Phys. Commun., 89, 149
CASTRO User Guide 2009, https://ccse.lbl.gov/Research/CastroUserGuide.pdf
Chandrasekhar, S. 1961 (ed.), Hydrodynamic and Hydromagnetic Stability (Oxford: Clarendon; Dover reprint, 1981)
Colella, P. 1990, J. Comput. Phys., 87, 171
Colella, P., & Glaz, H. M. 1985, J. Comput. Phys., 59, 264
Colella, P., & Sekora, M. D. 2008, J. Comput. Phys., 227, 7069
Colella, P., & Woodward, P. R. 1984, J. Comput. Phys., 54, 174
Colgate, S. A., & White, R. H. 1966, ApJ, 143, 626
Crutchfield, W. Y. 1991, Load Balancing Irregular Algorithms, Tech. Rep. UCRL-JC-107679, LLNL
Dimonte, G., et al. 2004, Phys. Fluids, 16, 1668
FLASH 3.2 User's Guide 2009, http://flash.uchicago.edu/website/codesupport/
Franklin Performance Monitoring 2010, N5 IOR Aggregate Write, http://www.nersc.gov/nusers/systems/franklin/monitor.php
Fryxell, B., et al. 2000, ApJS, 131, 273
Gittings, M., et al. 2008, Comput. Sci. Discovery, 1, 015005
Joggerst, C. C., Almgren, A., Bell, J., Heger, A., Whalen, D., & Woosley, S. E. 2009, ApJ, 709, 11
Kamm, J. R., & Timmes, F. X. 2007, ApJS, submitted, see http://cococubed.asu.edu/papers/la-ur-07-2849.pdf
Lattimer, J. M., & Swesty, F. D. 1991, Nucl. Phys. A, 535, 331
Layzer, D. 1955, ApJ, 122, 1
Ma, H., Woosley, S., Almgren, A., & Bell, J. 2010, BAAS, 41, 448
McCorquodale, P., & Colella, P. 2010, J. Comput. Phys., submitted
Miller, G. H., & Colella, P. 2002, J. Comput. Phys., 183, 26
Miniati, F., & Colella, P. 2007, J. Comput. Phys., 227, 400
Monchmeyer, R., & Muller, E. 1989, A&A, 217, 351
Müller, E. 1986, A&A, 162, 103
Nonaka, A., Almgren, A. S., Bell, J. B., Lijewski, M. J., Malone, C., & Zingale, M. 2010, ApJS, in press
Omang, M., Børve, S., & Trulsen, J. 2006, J. Comput. Phys., 213, 391
O'Shea, B. W., Bryan, G., Bordner, J., Norman, M. L., Abel, T., Harkness, R., & Kritsuk, A. 2005, in Lecture Notes in Computational Science and Engineering, Vol. 41, Adaptive Mesh Refinement—Theory and Applications, ed. T. Plewa, T. Linde, & V. G. Weirs (Berlin: Springer), 341
Plewa, T., & Müller, E. 1999, A&A, 342, 179
Rendleman, C. A., Beckner, V. E., Lijewski, M., Crutchfield, W. Y., & Bell, J. B. 2000, Comput. Vis. Sci., 3, 147
Ricker, P. M. 2008, ApJS, 176, 293
Saltzman, J. 1994, J. Comput. Phys., 115, 153
Sedov, L. I. 1959, Similarity and Dimensional Methods in Mechanics (Translated from the 4th Russian ed.; New York: Academic)
Sharp, D. H. 1984, Phys. D Nonlinear Phenom., 12, 3
Sod, G. A. 1978, J. Comput. Phys., 27, 1
Strang, G. 1968, SIAM J. Numer. Anal., 5, 506
Taylor, G. 1950, Proc. R. Soc. A, 201, 192
Timmes, F. X., & Swesty, F. D. 2000, ApJS, 126, 501
Toro, E. F. 1997, Riemann Solvers and Numerical Methods for Fluid Dynamics (Berlin: Springer)
VisIt User's Manual 2005, https://wci.llnl.gov/codes/visit/home.html
Woosley, S. E., et al. 2009, J. Phys. Conf. Ser., 180, 012023
Zingale, M., Almgren, A. S., Bell, J. B., Nonaka, A., & Woosley, S. E. 2009, ApJ, 704, 196
Zingale, M., et al. 2002, ApJS, 143, 539