An Example Implementation of a 1D Lagrangian Hydrodynamics Code

The "blcode" – Bruenn's Lagrangian Code

An implementation by Christian D. Ott, Viktoriya Morozova, and Tony Piro (all TAPIR, Caltech) based on Mezzacappa & Bruenn 1993, ApJ 405, 669, [1]

July 24, 2014

Abstract

We present an example implementation of the Mezzacappa & Bruenn 1993, ApJ 405, 669, [1], spherically-symmetric (1D) Lagrangian hydrodynamics scheme that employs artificial viscosity for shock capturing. We follow the original manuscript closely and deviate only in details. This is not a state-of-the art numerical approach to the 1D Lagrangian hydrodynamics problem, but it can serve as a good learning example for researchers interested in studying hydrodynamic phenomena. We call this code `blcode`.

1 Basic Equations Solved and Implementation Details

`blcode` solves the equations of Langrangian hydrodynamics (e.g., [2]). In the following, the mass coordinate m is the mass enclosed by radius r.

Mass conservation:

$$\frac{\partial r}{\partial m} = \frac{1}{4\pi r^2 \rho} \ . \tag{1}$$

Momentum conservation:

$$\frac{Dv}{Dt} = -\frac{Gm}{r^2} - 4\pi r^2 \frac{\partial P}{\partial m} \ , \tag{2}$$

where Dv/Dt is the Lagrangian time derivative at constant mass.

Energy conservation:

$$\frac{D\epsilon}{Dt} = -P \frac{D}{Dt}\left(\frac{1}{\rho}\right) \ , \tag{3}$$

which is just a reformulation of the first law of thermodynamics. ϵ is the specific internal energy (energy per mass) of a mass element.

1.1 Basic Conventions and Grid Setup

The grid is divided into `imax` computational cells. Quantities with subscript i live at the inner edge of computaitonal cell i. Quantities with subscript $i+1/2$ live at the center (in mass coordinate) of cell i. `blcode` **presently assumes uniform spacing in mass** (see `grid.F90`). This is unlikely to be optimal for many situations and the user should modify the gridding according to their needs.

- The mass coordinate m_i is the mass enclosed by the inner boundary of cell i. In the code, this is variable `mass(i)`.
- The mass point $m_1 = 0$ corresponds to the origin, $r = 0$, but often m_1 is set to some large mass, for example the mass enclosed inside a piston or inside a thermal bomb location.
- The mass contained in each cell is

$$\Delta m_{i+1/2} = m_{i+1} - m_i \ , \tag{4}$$

  for $i = 1, 2, ..., \text{imax} - 1$.
- The mass enclosed by the center of each zone is given by

$$m_{i+1/2} = m_i + \frac{1}{2}\Delta m_{i+1/2} \ , \tag{5}$$

  for $1, 2, ..., \text{imax} - 1$. And the mass difference between two consecutive cell centers is given by

$$\Delta m_i = m_{i+1/2} - m_{i-1/2} = \frac{1}{2}(\Delta m_{i-1/2} + \Delta m_{i+1/2}) \ , \tag{6}$$

  for $i = 2, 3, ..., \text{imax}$ and we set $\Delta m_{\text{imax}+1/2} = \Delta m_{\text{imax}-1/2}$.

1.2 Finite-Difference Form of the Equations

The guts of `blcode` are in `hydro.F90`. In the following, we present the evolved equations in finite-difference form in the order of the update sequence.

The velocity is kept at half timesteps ($n + 1/2$, indicated as a superscript) from all other variables. This is a way of making the evolution second-order accurate in time (see, e.g., [3]). The velocity at the inner edge of cell i at time $n + 1/2$ is given by

$$\begin{aligned} v_i^{n+1/2} &= v_i^{n-1/2} - \Delta t_v \frac{G m_i}{(r_i^n)^2} - \Delta t_v 4\pi (r_i^n)^2 \frac{P_{i+1/2}^n - P_{i-1/2}^n}{\Delta m_i} \\ &\quad - \Delta t_v 4\pi \frac{(r_{i+1/2}^n)^2 Q_{i+1/2}^{n-1/2} - (r_{i-1/2}^n)^2 Q_{i-1/2}^{n-1/2}}{\Delta m_i} \ , \end{aligned} \tag{7}$$

where $v_i^{n-1/2}$ is the old velocity and the second and third terms correspond to the right-hand-side of Eq. 2. The fourth term is due to the artificial viscosity, which is needed to stabilize the numerical evolution at discontinuities (= shocks). This is the von Neumann-Richtmyer approach to handling numerical hydrodynamics (e.g., [3, 4]). The artificial viscosity should be zero wherever the flow is smooth and non-zero where there are discontinuous changes in the state variables to damp numerical noise that will arise there. The detailed form of Q is given in §1.3 and in the momentum equation (Eq. 7) it is time-lagged for additional stability. The velocity update in Eq. 7 is spatially centered, thus is second-order also in space.

The timestep for the velocity update Δt_v is given as the average of the previous and the current timestep,

$$\Delta t_v \equiv \frac{1}{2}(\Delta t^{n-1/2} + \Delta t^{n+1/2}) \ , \tag{8}$$

where

$$\Delta t^{n+1/2} \equiv t^{n+1} - t^n \ . \tag{9}$$

We elaborate in §1.4 on how we chose $\Delta t^{n+1/2}$.

Once the new velocity at $n + 1/2$ is known, the radial coordinates are updated according to

$$r_i^{n+1} = r_i^n + \Delta t^{n+1/2} v_i^{n+1/2} \ . \tag{10}$$

Because the cell center is defined to contain half the mass in a cell, the discrete radial coordinate at time $n + 1$ is computed via a volume average (assuming that the density is constant across a zone):

$$r_{i+1/2}^{n+1} = \left[\frac{(r_i^{n+1})^3 + (r_{i+1}^{n+1})^3}{2} \right]^{1/3} \ . \tag{11}$$

Next, we use the updated radial coordinates to update the cell densities,

$$\rho_{i+1/2}^{n+1} = \frac{\Delta m_{i+1/2}}{\frac{4}{3}\pi[(r_{i+1}^{n+1})^3 - (r_i^{n+1})^3]} \ . \tag{12}$$

Note that the finite difference representation of the density equation is spatially centered and, hence, has a second-order truncation error in Δm.

Finally, we must update the specific internal energy. The finite-difference representation of the energy equation Eq. 3 is

$$\begin{aligned} \epsilon_{i+1/2}^{\star} \ = \ & \epsilon_{i+1/2}^{n} - \frac{1}{2}(P_{i+1/2}^{\star} + P_{i+1/2}^{n}) \left(\frac{1}{\rho_{i+1/2}^{n+1}} - \frac{1}{\rho_{i+1/2}^{n}} \right) \\ & -4\pi\Delta t (\frac{1}{2}[r_{i+1/2}^{n+1} + r_{i+1/2}^{n}])^2 Q_{i+1/2}^{n+1/2} \frac{v_{i+1}^{n+1/2} - v_i^{n+1/2}}{\Delta m_{i+1/2}} \ . \end{aligned} \tag{13}$$

In the above, the first term is the old specific internal energy, the second term is the PdV work, and the third term arises from the artificial viscosity (at timestep n). The new pressure $P_{i+1/2}^{n+1}$ will generally be a function of the new internal energy $\epsilon_{i+1/2}^{n+1}$. Hence, both quantities are marked with a star in Eq. 13 and the equation must be iterated to convergence.

`blcode` has two ways of handling this. For equations of state that do not explicitely include temperature, that is, $P = P(\rho, \epsilon)$, e.g., the common gamma-law $P = (\Gamma - 1)\rho\epsilon$, one just loops over Eq. 13 and updates the pressure $P_{i+1/2}^{\star}$ until $(|\epsilon_{i+1/2}^{\star,j+1} - \epsilon_{i+1/2}^{\star,j}|)/|\epsilon_{i+1/2}^{\star,j}|$, where j is the iteration number, is smaller than some tolerance `EPSTOL` set in `hydro.F90`. This way of updating the specific internal energy is chosen by setting `energy_update = ''epsilon''` in `parameters`.

The second way of updating the specific internal energy, `energy_update = ''temperature''`, is for normal stellar equations of state for which $P = P(\rho, T, \{X_l\})$ (where $\{X_l\}$ is the set of variables specifying the composition of the gas). In this case, first an intermediate temperature $T_{i+1/2}^{\star}$ must be found to obtain the intermediate pressure $P_{i+1/2}^{\star}$. In `blcode`, this is all formulated as a univariate root finding problem and solved via Newton-Raphson iteration.

The reader may recall that the root-finding problem in one variable is to find x_r for which $f(x_r) = 0$. In Newton-Raphson, we expand f about its root x_r to first order,

$$f(x_r) = f(x) + (x_r - x) f'(x) = 0 \ . \tag{14}$$

Now we interpret $f(x_r)$ as the trial value for the true root at the n-th step of an iterative procedure. The $n+1$-th step is then

$$f(x_{n+1}) = f(x_n) + \underbrace{(x_{n+1} - x_n)}_{\delta x} f'(x_n) \approx 0 \ , \tag{15}$$

and, thus,

$$x_{n+1} = x_n + \delta x = x_n - \frac{f(x_n)}{f'(x_n)} \ . \tag{16}$$

The iteration is stop when $f(x_{n+1}) = 0$ or when the fractional change from between iteration n and $n+1$ is smaller than some small number.

In `blcode`, we set

$$\begin{aligned} f(T^\star_{i+1/2}) \quad = \quad & \epsilon^\star_{i+1/2} - \epsilon^n_{i+1/2} \\ & + \frac{1}{2}(P^\star_{i+1/2} + P^n_{i+1/2}) \left(\frac{1}{\rho^{n+1}_{i+1/2}} - \frac{1}{\rho^n_{i+1/2}} \right) \\ & + 4\pi\Delta t (\frac{1}{2}[r^{n+1}_{i+1/2} + r^n_{i+1/2}])^2 Q^{n+1/2}_{i+1/2} \frac{v^{n+1/2}_{i+1} - v^{n+1/2}_i}{\Delta m_{i+1}} \ . \end{aligned} \tag{17}$$

$f'(T^\star_{i+1/2})$ is then given by

$$f'(T^\star_{i+1/2}) = \left(\frac{\partial \epsilon}{\partial T} \right)^\star_{i+1/2} + \frac{1}{2} \left(\frac{\partial P}{\partial T} \right)^\star_{i+1/2} \left(\frac{1}{\rho^{n+1}_{i+1/2}} - \frac{1}{\rho^n_{i+1/2}} \right) \ . \tag{18}$$

The derivatives of ϵ and P with respect to T are obtained from the equation of state.

1.3 Artificial Viscosity

The Euler equations admit weak solutions, i.e. solutions for which the integral form of the conservation laws hold, but the differential form is violated, because of discontinuities in the state variables. As a consequence, a naive finite-difference treatment of the Euler equations will lead to the growth and eventual blow-up of oscillations near discontinuities (e.g., shocks, contact discontinuities, surfaces of stars etc.). The idea of von Neumann & Richtmyer [4] is to avoid such oscillations by artificially spreading out discontinuities over multiple grid cells and thus avoiding the development of unstable oscillations. The *artificial viscosity* used for this should be non-zero only at discontinuities and zero everywhere else.

Mezzacappa & Bruenn [1] use an artificial viscosity prescription optimized for collapse. However, in `blcode`, we use the simpler original von Neumann & Richtmyer form,

$$Q \equiv \begin{cases} c_0^2 \rho (\partial v / \partial k)^2 & \partial v / \partial k < 0 \\ 0 & \text{otherwise} \ , \end{cases} \tag{19}$$

where $\partial . / \partial k$ denotes the deriviate with respect to an integer Lagrangian coordinate (some cell index k). The finite-difference form of this implemented in `artificial_viscosity.F90` is

$$Q^{n+1/2}_{i+1/2} = \begin{cases} c_0^2 \rho^{n+1}_{i+1/2} (v^{n+1/2}_{i+1} - v^{n+1/2}_i)^2 & \text{if } (v^{n+1/2}_{i+1} - v^{n+1/2}_i) < 0 \\ 0 & \text{otherwise} \ . \end{cases} \tag{20}$$

We set $c_0^2 = 2$, following [1, 5].

1.4 Timestep

The timestep used in the update of the hydrodynamic equations must not violate causality, that is, it must be no larger than the time it takes a sound wave to travel across a grid cell. In addition to the causality constraint, the timestep may further need to be limited to ensure stability of the numerical implementation (see, e.g., [2–4]).

In `blcode`, the timestep is determined in `timestep.F90` according to the following prescription. We first compute a local timestep for each cell i,

$$\Delta t^{n+1/2}_{i+1/2} = \min\left(\frac{r^n_{i+1} - r^n_i}{|v^{n-1/2}_i + c^n_{s,i+1/2}|}, \frac{r^n_{i+1} - r^n_i}{|v^{n-1/2}_i - c^n_{s,i+1/2}|}\right) , \tag{21}$$

where c_s is the speed of sound obtained from the EOS. We then use the global minimum as the new timestep,

$$\Delta t^{n+1/2} = CFL \min(\{t^{n+1/2}_i\}) , \tag{22}$$

where CFL is the Courant-Friedrichs-Lewy factor, which we set to 0.95 for stability.

1.5 Boundary Conditions

We must specify boundary conditions for coordinates and some hydrodynamics/thermodynamic state variables at both the inner and the outer boundaries of our grid. The default boundary conditions in `blcode` are the following. The user should feel free to experiment and change them according to their need.

Inner Boundary, `i=1` (quantities not specified are computed and not fixed)

Quantity	Description/Notes	Source File
$m_1 = 0$	Mass interior to inner zone; set to $m_1 = M_{\rm interior}$ for piston or thermal-bomb explosions.	`problem.F90`
$r_1 = 0$	Inner radius, in explosion case set to location of inner mass boundary	`read_profile.F90` and `hydro.F90`
$v_1 = 0$	Set to $v_{\rm piston}$ only when piston active.	`hydro.F90`

Outer Boundary, `i=imax` (quantities not specified are computed and not fixed)

Quantity	Description/Notes	Source File
$Q_{imax} = 0$	Artificial viscosity	`artificial_viscosity.F90`
$\rho_{imax+1/2} = \rho_{imax+1/2-1}$	Density	`hydro.F90`
$\epsilon_{imax+1/2} = \epsilon_{imax-1}$	Spec. int. energy.	`hydro.F90`
$T_{imax+1/2} = T_{imax-1}$	Temperature	`hydro.F90`
$P_{imax+1/2} = 0$	Pressure. Set to zero.	`hydro.F90`

Note that P_{imax} and Q_{imax} are the only important boundary conditions, since the other quantities at `imax` are never used in the evolution.

2 Basic Control and Setup of Problems

2.1 `parameters` file

Much of what `blcode` will do can be controlled via settings read in by `input_parser.F90` from a text file called `parameters` in the code's main directory. There are various parameters that steer

when the code will do output (their rather trivial meaning can be inferred from `blcode.F90` and `output.F90`).

The parameter `imax` sets the number of grid cells to be used in the simulation.

The parameter `energy_update` controls if the energy equation is updated via an iteration over temperature (`''temperature''`; as in [1]) or via an iteration over specific internal energy alone (`''epsilon''`). The latter is needed for equations of state that do not explicitely include a temperature (e.g., a gamma-law, $P = (\Gamma - 1)\rho\epsilon$).

Of particular importance is the parameter `initial_data` that will determine, which branch of `problem.F90` is executed and what problem `blcode` will solve (see §2.2).

Parameter `profile_name` sets the name of the profile that is read in by `read_profile.F90` and mapped to `blcode`'s grid. `blcode` uses the familiar `.short` format whose details can be inferred from `read_profile.F90`. This profile just contains the basic hydrodynamics and thermodynamics of the star. Parameters `ncomps` and `comp_profile_name` set the number of isotopes tracked and the name of the isotope profile file, respectively. The isotope profile is read and mapped by `read_profile.F90` from which the detailed contents of the isotope profile file can be inferred.

The equation of state (EOS) to be used is specified by parameter `eoskey`. See §3.

2.2 `problem.f90`

`problem.F90` is a central routine of the code, since it controls the problem solved, including EOS (see §3) and grid parameters.

Currently, `blcode` is set up for thermal bomb (`initial_data = ''Thermal_Bomb''`) and piston-driven (`initial_data = ''Piston_Explosion''`) explosions.

3 Equations of State

An equation of state (EOS) is crucial for closing the system of hydrodynamics equations. `blcode` implements a number of equations of state. Not all will work for all applications.

EOS parameters may vary from problem to problem. Hence, they are set in `problem.F90`. The top-level EOS routine is `eos.F90`, from which lower-level routines for the various EOS are called. Basic EOS information is also provided in the comment section at the top of file `eos.F90`. The EOS implementation currently present in the code is rather inefficient and could be optimized easily.

The EOS is selected with the `eoskey` variable that is set in the `parameters` file.

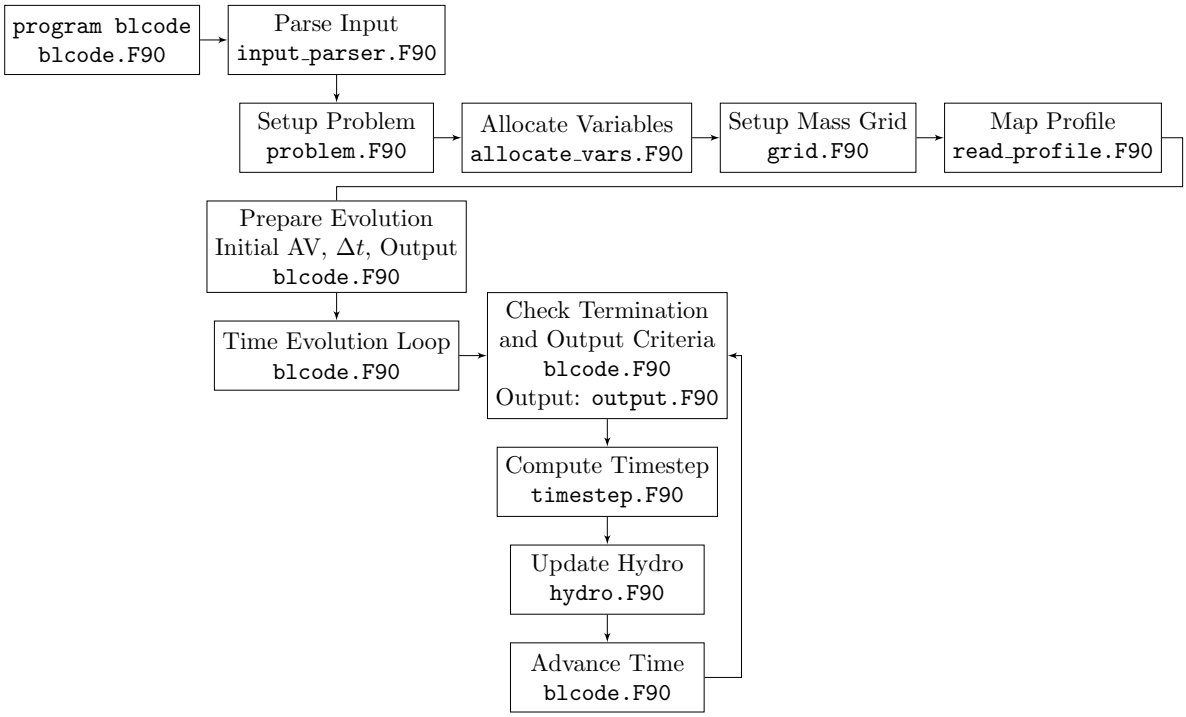
- `eoskey = 1` **Hybrid Polytropic/Gamma-Law EOS**: useful for simple stellar collapse problems, see Janka *et al.* 1993, `http://adsabs.harvard.edu/abs/1993A%26A...268..360J`. This equation of state does not define temperature and requires the parameter setting `energy_update = ''epsilon''`.
- `eoskey = 2` **Polytropic EOS**: $P = K\rho^{\Gamma}$. This EOS does not define temperature and requires the parameter setting `energy_update = ''epsilon''`.
- `eoskey = 3` **Ideal single-particle Boltzmann Gas EOS**

- `eoskey = 4` **Helmholtz EOS**: ideal gases of ions, photons, electrons, and positrons. Based on Timmes & Swesty 2000, http://adsabs.harvard.edu/abs/2000ApJS..126..501T.

- `eoskey = 5` **Gamma-Law EOS**: $P = (\Gamma - 1)\rho\epsilon$. This equation of state does not define temperature and requires the parameter setting `energy_update = ''epsilon''`.

- `eoskey = 6` **Paczynski EOS**: Simplified EOS for a mixture of ions, photons, and semi-degenerate/semi-relativistic electrons. Based on Paczynski 1983, http://adsabs.harvard.edu/abs/1983ApJ...267..315P. Note that the speed of sound for this EOS is currently defined as $c_s^2 = P/\rho$, which is *not* the correct adiabatic speed of sound. It is somewhat involved to work out the correct expression from Paczynski's approximate expressions and we leave this to future revisions of the code. Since the speed of sound is used only for determining the timestep and does not enter the evolution equations, we find that our simple approximation does not distort the hydrodynamic evolution. We recommend to test the sensitivity of simulations to changes in the Courant factor (parameter `dtfac` in `timestep.F90`).

4 Code Flow Chart



References

[1] A. Mezzacappa and S. W. Bruenn. A numerical method for solving the neutrino Boltzmann equation coupled to spherically symmetric stellar core collapse. *Astrophys. Journal*, 405:669, March 1993.

[2] D. Mihalas and B. Weibel-Mihalas. *Foundations of Radiation Hydrodynamics*. Dover Publications, Mineola, NY, USA, 1999.

[3] R. L. Bowers and J. R. Wilson. *Numerical modeling in applied physics and astrophysics*. Jones and Bartlett, Boston, MA, USA., 1991.

[4] J. Von Neumann and R. D. Richtmyer. A Method for the Numerical Calculation of Hydrodynamic Shocks. *J. Appl. Phys.*, 21:232, March 1950.

[5] M. C. Bersten, O. Benvenuto, and M. Hamuy. Hydrodynamical Models of Type II Plateau Supernovae. *Astrophys. Journal*, 729:61, March 2011.