How to Write a Hydrodynamics Code

INTRODUCTION

In this note, I describe how to write an Eulerian hydrodynamics code. In §1, I show how to write a one-dimensional first-order hydrodynamics code based on an approximate Riemann solver. In §2, I show how to extend the 1D first-order code to high (2nd) order. In §3, I describe how to write a two-dimensional high-order hydrodynamics code. §4 lists a few additional test problems, and §5 briefly mentions some possible extensions and relevant references.

(Please contact me if any of the links die or you spot any typos.)

1 A ONE-DIMENSIONAL FIRST-ORDER METHOD

The equations of one-dimensional hydrodynamics can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0, \tag{1}$$

where $\mathbf{U} = (\rho, \rho v, E)$ are the conserved variables and $\mathbf{F} = (\rho v, \rho v^2 + P, v(E + P))$ are the fluxes; ρ is the fluid density, v is the fluid velocity, P is the pressure, and $E = \rho e + \frac{1}{2}\rho v^2$ is the total energy density, and each of the aforementioned quantities is a function of space and time, or x and t. The equations are closed by an equation of state (EOS) given by $P = P(\rho, e)$.[1] For an ideal gas, the EOS reads

$$P = (\gamma - 1)\rho e, \tag{2}$$

where γ is the adiabatic index of the ideal gas.

To numerically approximate solutions to the above equation, we can rewrite Equation 1 in semi-discrete form:[2]

$$\frac{\partial \mathbf{U}_i}{\partial t} = L(\mathbf{U}) = -\frac{\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2}}{\Delta x}, \tag{3}$$

where i denotes the cell with its center at x_i, Δx is the cell width, and $\mathbf{F}_{i\pm1/2}$ are the fluxes at the cell interfaces.

The time integration can be accomplished using the first-order forward Euler method,

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t L(\mathbf{U}^n), \tag{4}$$

where $\mathbf{U}^n$ are the conserved variables and timestep n and $\mathbf{U}^{n+1}$ are the conserved variables after advancing one timestep of size Δt.

To obtain $\mathbf{F}_{i\pm1/2}$, the fluxes at each cell interface, we can solve the so-called Riemann problem. Given variables at cell i and cell $i+1$, we can calculate the flux at the interface, $x = x_{i+1/2}$. This problem can be solved nearly exactly numerically, but iteratively and at substantial cost. Instead, we can make good progress with an approximate Riemann solver, of which there are many. As an example, we'll use the HLLE

[1] In general, the equation of state can also be a function of composition, but in many cases a single-component gas is good enough.

[2] That is to say, substituting approximate numerical spatial derivative for the ∂_x operator while leaving the ∂_t operator without approximation for the time being. Note that Equation 3 simply replaces ∂_x with a 2nd-order central finite difference operator.

Riemann solver (after Harten, Lax, van Leer, and Einfeldt). Given the left state $\mathbf{U}^L$ and right state $\mathbf{U}^R$, the HLLE flux can be written as

$$F^{\mathrm{HLLE}} = \frac{\alpha^+ \mathbf{F}^L + \alpha^- \mathbf{F}^R - \alpha^+\alpha^-(\mathbf{U}^R - \mathbf{U}^L)}{\alpha^+ + \alpha^-}, \tag{5}$$

where α^+ and α^- are related to the minimum and maximum eigenvalues of the Jacobians of the left and right states,

$$\alpha^{\pm} = \max\{0, \pm\lambda^{\pm}(\mathbf{U}^{\mathrm{L}}), \pm\lambda^{\pm}(\mathbf{U}^{\mathrm{R}}). \tag{6}$$

The minimum and maximum eigenvalues $\lambda^{\pm}$ are given by

$$\lambda^{\pm} = v \pm c_s, \tag{7}$$

where $c_s = \sqrt{\gamma P/\rho}$ is the sound speed.[3] As an example of how to employ the HLLE approximate flux, one obtain $\mathbf{F}_{i+1/2}$ in Equation 5 by substituting $\mathbf{U}(x_i)$ for $\mathbf{U}^L$ and $\mathbf{U}(x_{i+1})$ for $\mathbf{U}^R$ in Equation 3.

The timestep used in Equation 4 must satisfy the Courant-Friedrich-Levy condition for the evolution to be stable. Thus, the following condition must be satisfied,

$$\Delta t < \Delta x / \max(\alpha^{\pm}). \tag{8}$$

To test the code, there are a serial of tests one can run. The first test is usually the Sod shock tube problem. In this test, the one-dimensional numerical region ($0 \le x \le 1$) initially consists of two constant states: $P_L = 1.0$, $\rho_L = 1.0$, $v_L = 0.0$ and $P_R = 0.125$, $\rho_R = 0.1$, $v_R = 0.0$, where L stands for the left state, and R the right state. The fluid is assumed to be an ideal gas with an adiabatic index $\gamma = 1.4$. The initial discontinuity is at x = 0.5. In this test problem, the evolution of the initial discontinuity gives rise to a shock, a rarefaction wave, and a contact discontinuity in between. This is a fairly easy test. All modern hydrodynamics codes should be able to capture the expected features, acquire correct positions of the shock front, contact discontinuity and rarefaction wave.

You should run the Sod problem and compare the numerical results with the exact solutions. You should also try a harder shock tube problem, with $P_L = 100$, $\rho_L = 10$, $v_L = 0$ and $P_R = 1$, $\rho_R = 1$, $v_R = 0.0$. To calculate the analytical solution, you can download Bruce Fryxell's code (hosted by Frank Timmes) at https://cococubed.com/code_pages/exact_riemann.shtml.

2 A ONE-DIMENSIONAL SECOND-ORDER METHOD

It is fairly straightforward to extend the first-order method presented in Section 2 to second order, which can very significantly reduce errors.

For time integration, we can simply replace the 1st-order Euler's method, Equation 4, with a higher-order Runge-Kutta scheme. One popular approach is Heun's method, which uses Euler's method as an initial guess, which is then corrected. The method is the following:

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \Delta t L(\mathbf{U}^n) \tag{9}$$

$$\mathbf{U}^{n+1} = \frac{1}{2}\left(\mathbf{U}^n + \mathbf{U}^{(1)} + \Delta t L(\mathbf{U}^{(1)})\right) = \mathbf{U}^n + \frac{\Delta t}{2}\left(L(\mathbf{U}^n) + L(\mathbf{U}^{(1)})\right). \tag{10}$$

Because Equation 3 is already spatially accurate to second order, the necessary step in achieving *overall*

[3] A few exegetic remarks on Equation 5 may be helpful. The final term in its numerator, containing the difference in the left and right conserved variables, is a form of numerical dissipation that disappears when the left and right states are identical; without that dissipative term, the HLLE flux is essentially a weighted sum of the individual fluxes. Careful examination of the definitions of α will reveal that each coefficient encodes a notion akin to 'what direction information is flowing' or "upwinding;" the right state's flux is weighted by the maximum wavespeed of waves moving from right to left, and the left state's flux is weighted by the maximum speed of waves moving from left to right.

second order accuracy is to determine higher-order approximations of $\mathbf{U}^L$ and $\mathbf{U}^R$. The treatment in Section 2, using the values $\mathbf{U}(x_i \pm 1/2)$, assumes that the value of each quantity is constant within each cell. Higher-order accuracy can be achieved by using higher-order interpolation from the cell centers to cell faces; linear interpolation is sufficient to achieve second-order accuracy. However, care must be taken to approach this in a stable way by limiting extreme gradients (such as those near discontinuities). To obtain the pressure, density, and velocity of the let and right states at the cell interface $i+1/2$, we need the states at $i-1$, i, $i+1$, and $i+2$. Given the values c_{i-1}, c_i, c_{i+1}, the left and right interface values are given by

$$c^L_{i+1/2} = c_i + 0.5\text{minmod}\left(\theta(c_i - c_{i-1}),\, 0.5(c_{i+1} - c_{i-1}), \theta(c_{i+1} - c_i)\right) \tag{11}$$

$$c^R_{i+1/2} = c_{i+1} - 0.5\text{minmod}\left(\theta(c_{i+1} - c_i),\, 0.5(c_{i+2} - c_i), \theta(c_{i+2} - c_{i+1})\right), \tag{12}$$

where the $1 \le \theta \le 2$, and the minmod function, important for preserving monotonicity, reads

$$\text{minmod}(a,b,c) = \frac{1}{4}\left|\text{sgn}(a) + \text{sgn}(b)\right|\left((\text{sgn}(a) + \text{sgn}(c))\right)\min\left(|a|, |b|, |c|\right) \tag{13}$$

and the sgn function returns its argument's sign. Equation 13 becomes the more diffusive 'minmod' limiter when $\theta = 1$, and the less diffusive 'monotonized central' limiter when $\theta = 2$.

Equations 11—13 have assumed a constant grid spacing, but can easily be extended by reintroucing the elided factors of Δx. Note, for example, how 11 and 12 are essentially applications of a Taylor series given a central value of a function at each cell extrapolated half a cell width using an approximate values of the slope at the center of the cell.

3 A TWO-DIMENSIONAL METHOD

It is straightforward to extend the above method to multiple dimensions. Using two dimensions as an example, the equations of gas dynamics can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0. \tag{14}$$

Now, the conserved variables are $\mathbf{U} = (\rho, \rho v_x, \rho v_y, E)$, the flux in the x-direction is

$$\mathbf{F} = \left(\rho v_x, \rho v_x^2 + P, \rho v_x v_y, (E+P)v_x\right), \tag{15}$$

and the flux in the y-direction is

$$\mathbf{G} = \left(\rho v_y, \rho v_x v_y, \rho v_y^2 + P, (E+P)v_y\right), \tag{16}$$

and the total energy is now given by $E = \rho e + \frac{1}{2}\rho(v_x^2 + v_y^2)$.

Using the method of lines, we can approximate Equation 14 in semi-discrete form according to

$$\frac{\partial \mathbf{U}_{i,j}}{\partial t} = L(\mathbf{U}_{i,j}) = -\frac{\mathbf{F}_{i+1/2,j} - \mathbf{F}_{i-1/2,j}}{\Delta x} - \frac{\mathbf{G}_{i,j+1/2} - \mathbf{G}_{i,j-1/2}}{\Delta y}, \tag{17}$$

where $\mathbf{F}_{i\pm1/2,j}$ and $\mathbf{G}_{i,j\pm1/2}$ are the fluxes at each cell interface or the x and y directions respectively, which can be averaged as in the 1D case. The same time-integration procedures can be employed here without any modifications. You should be able to write a 2D hydro code base on this note.

4 ADDITIONAL PROBLEMS

In the following, I'll describe a series of one-dimensional test problems. However, these can serve as nontrivial two-dimensional tests by applying a simple coordinate transformation.

A Linear Wave Test

Earlier we asserted that the method would be 1st- or 2nd-order, but never actually checked. If nothing else, checking the order of convergence is a good way to identify typos, and build trust in your implementation when applied to problems without known solutions (which is to say, most of them). A simple test is linear advection.

The pressure should be constant, and the fluid velocity should be constant. However, the fluid density should be some periodic and non-negative function, be it a Gaussian bump or something like $1+0.2\sin(2\pi x)$. The final density distribution after some time should simply be the initial distribution translated by tv, making it trivial to check the numerical solution against the analytical one. Often one will use periodic boundary conditions and run for some integer number of cycles. One can integrate the error across the domain, using some error norm, e.g. $E = \int |\rho - \rho_0| dx$, and should find that $E \propto (\Delta x)^p$ where p is the order of the scheme.

Shock-Wave Interactions

A particularly interesting and challenging test problem was introduced by Shu & Osher (1989), which consists of a Mach 3 shock wave interacting with a smoothly varying density distribution. This tests the ability of a code to capture small-scale smooth fluctuations simultaneously with strong shocks. This test assumes a domain $x \in [-5, 5]$, with initial conditions (typically setting $\alpha = 0.1$)

$$\begin{cases} \rho = 3.857143,\ v_x = 2.629369,\ P = 10.33333 & \text{for } x < -4 \\ \rho = 1 + \alpha \sin(5x),\ v_x = 0,\ P = 1. & \text{for } x \geq -4 \end{cases}$$

5 EXTENSIONS

There are many different ingredients in this code that we could improve. These include improving its accuracy (for smooth flows, for shocks, or in general) or adding additional physics.

Accuracy

We make a number of approximations as we *discretize* differential equations on a computer.

Time

A relatively simple one is the discretization in time, which we approximated using the 1st-order-accurate forward Euler method in Equation 4 and the 2nd-order-accurate Heun's method in Equation 9; as we decrease the timestep, the error in this stage of the algorithm would decrease as $\propto \Delta t$ or $\propto \Delta t^2$ respectively. A very useful third-order method was introduced by Shu & Osher (1988), and a useful fourth-order method was introduced in (Kraaijevanger 1991). A good review is provided by Gottlieb et al. (2009), which is available here.

Space

When writing Equation 3 we used a 2nd-order central finite difference approximation, and in §2 we treated the quantities within each cell as varying linearly (in space). We did not need to worry about what the values stored by each cell (e.g. $\mathbf{U}_i$) really meant: they could represent the average value of each field within the cell, or the point value of the field at the center of the cell, since for linear functions those are the same. At higher-order, more accurate approximations, we need to make a choice. If you have a copy lying around, Chapter 9 of Rezzolla & Zanotti (2013) provides a decent overview. Balsara (2017) provides a good review of finite-volume methods (those evolving cell-averaged values), and is available here.

A major consideration is how to *reconstruct* variables from cell centers/averages in an accurate and stable way, as we did with slope limiting in §2 at second order. One early and quite popular method was introduced in Colella & Woodward (1984), assuming a *piecewise parabolic* representation. Another useful class of methods (weighted essentially-non-oscillatory or WENO) is based on using a weighted combination of trial reconstructions, where the weights are chosen to achieve an optimally high order in smooth regions but achieve behavior similar to §2 in the presence of shocks. The first hallmark work on WENO schemes was Jiang & Shu (1996), but a whole zoo of schemes have been developed introducing various improvements, including Borges et al. (2008).

Discontinuities

The HLLE Riemann solver we implemented earlier works pretty well. But it could do better. As an example, try setting up a shock-like initial condition with left state $P_L = 1.0$, $\rho_L = 10.0$, $v_L = 0.0$ and right state $P_R = 1.0$, $\rho_R = 0.1$, $v_R = 0.0$, and then evolving the system with the method developed above. Even though $\mathbf{F} = 0$ (initially), the system evolves! This is because the HLLE *approximate* Riemann solver is a bit heavy-handed when it comes to the dissipative term $\propto (\mathbf{U}^R - \mathbf{U}^L)$ in Equation 5. This problem was tackled in Toro et al. (1994), which introduced the HLL +Contact (or HLLC) approximate Riemann solver, which is able to more accurately model "contact discontinuities" like the one we set up above (with discontinuous density but constant pressure and velocity). Another method, introduced by Roe (1981), uses an exact solution to a linearized Riemann problem, which typically results in less dissipation than HLLE.

As a word of caution, these one-dimensional low-dissipation approximate Riemann solvers can lead to numerical instabilities when simulation multidimensional shocks. See Appendix C of Stone et al. (2008) for further discussion.

Physics

Above, we limited ourselves to the equations of ideal hydrodynamics and assumed an idea equation of state. We can relax both of these assumptions. This will often involve thinking about not just conserved variables and fluxes, but *source terms*. The governing equations will then look something like

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{S}. \tag{18}$$

(Self-)Gravity

Fluids can be affected by both external gravitational fields (from a potential Φ_{ext}) or from their own weight (resulting in a potential Φ_s). These potentials will modify the momentum of the fluid, since their gradient is a gravitational force; this can introduce challenges to conserving linear momentum and energy. See Hanawa & Mullen (2025) for a cutting-edge approach to this problem. Another challenge when dealing with self-gravitating fluids is finding Φ_s given the instantaneous mass distribution. This requires solving Poisson's equation,

$$\nabla^2 \Phi = 4\phi G \rho. \tag{19}$$

This type of equation ("Elliptic") calls for different methods than the ones used above: see Chapter 9 of this book for an introduction.

Dissipation

Oftentimes, people add viscous dissipation to the hydrodynamics equations. Sometimes this emulates some unresolved process (an "eddy viscosity") and other times it is meant to provide a more physical dissipation mechanism (than numerical dissipation) and is useful for simulations of turbulence. Viscosity tends to smooth out velocity gradients, and the resulting change in the kinetic energy of the fluid is typically transformed into internal energy, i.e. heat. A good introduction is provided by Chapter 10 of this book.

Heating and Cooling

On occasion we would like to model how fluids are heated or cooled over time by processes internal or external. The method typically depends on the system at hand, but one ad-hoc model is that of Newtonian cooling, or thermal relaxation. In this case, we add an energy source term $\sim \rho(e_0 - e)/\tau_c$ where ρe_0 is a (potentially spatially dependent) target internal energy and τ_c is a characteristic cooling timescale. For this to be captured accurately, the timestep must be less than τ_c. If $\tau_c \ll \Delta x/|\alpha^{\pm}|$, then implicit time integration might be necessary. See, for example, Chapter 1.2.5 of this book for a friendly introduction.

Magnetic Fields

Many astrophysical systems are magnetized, and it is relatively straightforward to add magnetic fields to the Equation 1 (as long as the fluid is fully ionized). However, because magnetic fields need to satisfy $\nabla \cdot \mathbf{B} = 0$, it can be tricky to implement the equations of magneto-hydrodynamics on a computer. A nice introduction to astrophysical plasmas is given in Kulsrud (2005), and a description of a modern code to solve the MHD equations is given by Stone et al. (2008).

ACKNOWLEDGMENTS

This note is based on an original (covering §1-3) by Weiqun Zhang, which was passed on to me as a fledgling graduate student by Geoff Ryan. All errors are my own.

REFERENCES

Balsara D. S., 2017, Living Reviews in Computational Astrophysics, 3, 2
Borges R., Carmona M., Costa B., Don W. S., 2008, Journal of Computational Physics, 227, 3191
Colella P., Woodward P. R., 1984, Journal of Computational Physics, 54, 174
Gottlieb S., Ketcheson D. I., Shu C.-W., 2009, Journal of Scientific Computing, 38, 251
Hanawa T., Mullen P. D., 2025, The Astrophysical Journal Supplement Series, 277, 53
Jiang G.-S., Shu C.-W., 1996, Journal of Computational Physics, 126, 202
Kraaijevanger J. F. B. M., 1991, BIT Numerical Mathematics, 31, 482
Kulsrud R. M., 2005, Plasma Physics for Astrophysics. Princeton University Press, `http://www.jstor.org/stable/j.ctvzsmf0w`
Rezzolla L., Zanotti O., 2013, Relativistic Hydrodynamics
Roe P., 1981, Journal of Computational Physics, 43, 357
Shu C.-W., Osher S., 1988, Journal of Computational Physics, 77, 439
Shu C.-W., Osher S., 1989, Journal of Computational Physics, 83, 32
Stone J. M., Gardiner T. A., Teuben P., Hawley J. F., Simon J. B., 2008, The Astrophysical Journal Supplement Series, 178, 137
Toro E. F., Spruce M., Speares W., 1994, Shock Waves, 4, 25