Radiative Gas Dynamics
Problem Set 5
Due Thursday, March 1

1-d Hydro Code: Artifical Viscosity and Shocks

The goal of this problem set is to extend the 1-d hydrodynamics code that you wrote for Problem Set 3 in several ways, so that it can handle some more interesting physical processes. *Save the old version of your code.* You will want it to test your modifications against.

(a) Evolving the thermal energy

Previously you evolved only the density and velocity fields and computed pressure assuming a polytropic equation of state. Modify the code so that it also evolves the thermal energy equation:

$$\frac{\partial \epsilon}{\partial t} = -u\frac{\partial \epsilon}{\partial x} - \frac{P}{\rho}\frac{\partial u}{\partial x}. \tag{1}$$

Note that this is very similar to the equation you are already evolving for the density, so you can adapt your code fairly easily (including all of the timestepping details) by copying the density evolution lines and modifying them accordingly.

When you set up initial values of ρ and u, you now also need to set up initial values of ϵ using

$$\epsilon = \frac{1}{\gamma - 1}\frac{P}{\rho} = \frac{1}{\gamma - 1}T_1\rho^{\gamma-1}, \tag{2}$$

where the second equality makes use of the system of units adopted in the code (see equation 4 of PS 3). You can make this change to the initial conditions in whatever way is most convenient for you. I actually use a separate program to generate my initial conditions, which the hydro code then reads in. (And we will be trying a variety of initial conditions in this problem set, so you may want to implement this now if you didn't before.) I decided it was easiest to have my initial conditions program generate temperatures,

$$T = T_1\rho^{\gamma-1}, \tag{3}$$

then have the hydro code convert these to $\epsilon = T/(\gamma - 1)$ when it reads them in. We will restrict ourselves to $\gamma = 5/3$ in this problem set; $\gamma = 1$ obviously has some complications once one takes the thermal energy equation seriously.

Now that you have the thermal energy, you should calculate the pressure (and pressure gradients) from the equation of state,

$$P = (\gamma - 1)\rho\epsilon, \tag{4}$$

instead of using $P = T_1\rho^\gamma$. At this stage, there is no obvious need to define a whole pressure array p_i, since you can just calculate the gradient of $(\gamma - 1)\rho\epsilon$ on the fly. However, we will be adding complications to the pressure in part (d) below, and I recommend that you create a pressure array and just add a separate loop that calculates p_i from ρ_i and ϵ_i before taking the steps to advance ρ, u, and ϵ. (If this were a 3-d code instead of a 1-d code, we might be more reluctant to create extra arrays, but as it is we have plenty of memory.) Remember to apply the periodic boundary

condition to p_i after the loop. Also, remember that you need to calculate p_i twice per timestep, once to advance ρ, u, and ϵ to the midstep, and once (using the midstep values of ρ and ϵ) to take the full step. However, you don't need two p_i arrays, since you don't need to save values.

Modify the output of your program so that it prints out the temperature $T = (\gamma - 1)\epsilon$ in addition to ρ and u.

You don't need to attach any plots or compute any numbers for this part. However, you do need to get this right, or else you won't be able to do subsequent parts. To check that you have coded the thermal energy evolution correctly, rerun the low amplitude perturbations that you did with the code before and (a) check that you get the same evolution of the perturbations, (b) check that at the final time $T = T_1\rho^{\gamma-1}$ in every cell. Since the evolution is adiabatic, there should be no change in the specific entropy

$$s = c_V \ln(P\rho^{-\gamma}) = c_V \ln(T\rho^{1-\gamma}) \tag{5}$$

(where I have dropped additive constants). Also, modify your program so that before each output time it calculates the total thermal plus kinetic energy,

$$E = \sum_i \rho_i \left[\tfrac{1}{2}u_i^2 + \epsilon_i\right]. \tag{6}$$

It should write the time and this energy to a diagnostic output file (which should probably be different from the file where it writes the density, velocity, and temperature fields). Check that your modified program conserves energy.

There is one additional change that I recommend making to your program at this point. Add a loop after each update to the fields that checks whether ρ or ϵ is negative in any cell. If so, the calculation has gone wrong, and the program should halt – it is about to run into some mathematical catastrophe like square-root of a negative number anyway. Your code may crash of its own accord, but modern compilers often have the machine grind away computing useless "`nan`" values, very slowly, and your life will be happier if you stop the madness yourself.

(b) Some new initial conditions

With $N = 200$, $\Delta t = 10^{-3}$, $T_1 = 1$, $\gamma = 5/3$, evolve the initial conditions

$$\rho(x, t = 0) = 1 + \Delta \sin(kx), \tag{7}$$

$$u(x, t = 0) = a_s \Delta \sin(kx), \tag{8}$$

where the wavenumber $k = 2\pi/\lambda$, the wavelength $\lambda = 0.5$, $a_s = (\gamma T_1)^{1/2}$ is the sound speed at $\rho = 1$, and $\Delta = 0.02$ is the fractional amplitude of the perturbation.

Evolve to $t = 1.0$, with outputs every $\Delta t = 0.1$. Make an illustrative plot (e.g., time evolution of the density field, but you don't need to show every output time) and describe (in a couple of sentences) what is going on. (You may find it easier to interpret the results after you have also done the three cases below.)

Now evolve initial conditions that are the same as equation (7) and (8) for $x \leq 2\pi/k$ but have $\rho = 1$, $u = 0$ for $x > 2\pi/k$, i.e., just a single sine "pulse." Make a similar plot to the one you made for the continuous sine train.

Rerun the single pulse, but with amplitude $\Delta = 0.06$. How is the result different?

Finally, evolve the initial density field defined by equation (7) (for all x, not just $x \leq 2\pi/k$), with $\Delta = 0.02$, but with $u = 0$ everywhere. Interpret the result.

(c) A new boundary condition

So far we have used a periodic boundary condition. Now change the boundary condition to "hard walls" at the edges of the box by setting $u_1 = u_N = 0$ after each midstep and step (i.e., every time you update the u array or calculate the values of u at midstep, set the values in cells 1 and N to zero). Since you still want your centered differences to work, you should set $u_0 = u_{N+1} = 0$ as well. Because we will make use of periodic boundary conditions again later, I recommend implementing this hard boundary condition as an *option* to the code that you can control by a flag, rather than recompiling every time or making two different programs.

Rerun the single pulse using these new boundary conditions. Adopt $\Delta = 0.002$ (not $\Delta = 0.02$ as before) and run to $t = 1.7$. Attach a plot and interpret the results. It may help to look at the velocity field as well as the density field.

(d) Artificial viscosity and shocks

Now things get interesting. From the $\Delta = 0.06$ sine pulse of (b), it is clear that the code as it exists breaks down when a shock arises. In order to deal with shocks, we will make use of the *artificial viscosity* technique invented by von Neumann and Richtmeyer in the early 1950s.

In place of the thermal pressure $p_i = (\gamma - 1)\rho_i \epsilon_i$, use $p_i + q_i$ where the artificial viscosity contribution to the pressure is:

$$\begin{aligned} q_i &= \rho_i \left[C_q (u_{i+1} - u_{i-1})^2 - D_q (u_{i+1} - u_{i-1}) \sqrt{\gamma(\gamma-1)\epsilon_i} \right] \qquad \text{if } u_{i+1} - u_{i-1} < 0, \\ q_i &= 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{if } u_{i+1} - u_{i-1} > 0. \end{aligned} \tag{9}$$

C_q and D_q are constants to be specified later. The quantity in $\sqrt{\ }$ is the local speed of sound a_s. There are several things to note about this definition. First, the artificial viscosity contribution to pressure is zero when the velocity field is diverging, $\frac{\partial u}{\partial x} > 0$. Second, the artificial viscosity contribution is large and positive when there is a strong negative gradient (convergence) of the velocity field. Thus, the artificial viscosity is a form of pressure that suppresses strong negative velocity gradients. Third, the artificial viscosity depends not on $\frac{\partial u}{\partial x}$ itself but on the difference between neighboring *cells*; the first term is $\propto \left(\frac{\partial u}{\partial x}\Delta x\right)^2$ and the second term is $\propto \left(\frac{\partial u}{\partial x}\Delta x\right) a_s$. In the limit of high resolution ($N \to \infty$, $\Delta x \to 0$), the artificial viscosity is negligible everywhere except where the velocity field is nearly discontinuous (i.e., in the immediate neighborhood of shocks).

You must use $p_i + q_i$ in place of p_i in *both* the momentum equation and the energy equation; otherwise the code will not conserve energy.

Using $N = 400$ cells (not $N = 200$), set up initial conditions

$$\begin{aligned} \rho &= 1 && \text{for all } x, \\ u &= 0 && \text{for all } x, \\ T &= 10 && \text{for } x \le 0.2, \\ T &= 10 - 9(x - 0.2)/0.02 && \text{for } 0.2 < x \le 0.22, \\ T &= 1 && \text{for } x > 0.22. \end{aligned} \tag{10}$$

The density and velocity field are unperturbed, but the temperature has a sharp drop between $x = 0.2$ and $x = 0.22$. (We use a ramp rather than a truly discontinuous drop to avoid numerical problems.)

Evolve these initial conditions to $t = 0.2$ using the hard boundary implemented in part (c), with viscosity parameters $C_q = 4$, $D_q = 2$, and timestep $\Delta t = 10^{-4}$. Plot $\rho(x)$, $u(x)$, and $T(x)$ at $t = 0.02$, 0.04, 0.08, 0.12, 0.16, 0.20. In addition, plot the specific entropy

$$s(x) = 1.5 \ln(P\rho^{-\gamma}) = 1.5 \ln(T\rho^{1-\gamma}). \tag{11}$$

Write a brief interpretation of your results.

By visually estimating the position of the shock front as a function of time in your plots, determine the Mach number M_1 of the shock. (Remember that the relevant sound speed is $a_s = \sqrt{\gamma P/\rho} = \sqrt{\gamma T}$, not the isothermal sound speed.) Are the shock jump conditions for density, velocity, and temperature satisfied? Is energy conserved?

Some optional experiments (if you have time):

Try $N = 200$ instead of $N = 400$.

Experiment with C_q and D_q and describe the effects of the two terms on the calculation.

Rerun the $\Delta = 0.06$ sine pulse with $C_q = 4$, $D_q = 0.1$, and compare to the previous results with no viscosity.

Rerun the $\Delta = 0.06$ sine pulse with $C_q = 4$, $D_q = 2$.

See what happens if you make the temperature truly discontinuous at $x = 0.2$.