Computational methods in Astrophysics

Homework #3

Programs can be written in c, c++, Matlab, Python or Python notebook (preferably the last). In addition to the program, you should have a writeup that contains plots requested, answers to any analytical or explanation questions, and a short description of your code and how to run it. This can be done in, e.g., LATEX, WORD, etc. If your program requires a compilation to run, add the executable file as well. Code and writeup should be submitted to the Moodle site.

1) *ODE solution of earth orbit around the sun:* (based on Zingale exercise 1.8). Consider the orbit of Earth around the Sun. If we work in astronomical units, years, and solar masses, then Newton's gravitational constant and the solar mass together are simply $GM = 4\pi^2$ (this should look familiar as Kepler's third law). The ODE system describing the motion of Earth is:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= -\frac{GM\mathbf{r}}{r^3} \end{aligned} \tag{1}$$

If we take the coordinate system such that the Sun is at the origin, then, $\mathbf{r} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}}$ and $\mathbf{v} = v_x\hat{\mathbf{x}} + v_y\hat{\mathbf{y}}$ are the radius vector pointing from the Sun to Earth and Earth velocity. Take as initial conditions the planet at perihelion:

$$\begin{aligned} x_0 &= 0 \\ y_0 &= a(1-e) \\ v_{x,0} &= -\sqrt{\frac{GM}{a}\frac{1+e}{1-e}} \\ v_{y,0} &= 0 \end{aligned} \tag{2}$$

where a is the semi-major axis and e is the orbit eccentricity. Integrate this system for a single orbital period with the first-order Euler and the RK4 method and measure the convergence by integrating at a number of different Δt's. Note: you'll need to define some measure of error, you can consider a number of different metrics, e.g., the change in radius after a single orbit.

2) *The advection equation in 1D with finite differences* (based on Garcia 7.1).

a) The "upwind" scheme for solving the advection equation uses a left derivative for the $\partial/\partial x$ term,

$$\frac{a_i^{n+1} - a_i^n}{\Delta\tau} = -u\frac{a_i^n - a_{i-1}^n}{h} \tag{3}$$

where $\Delta\tau$ and h are the time step and grid space respectively. Use this method to solve the advection equation for the initial conditions used in the class (a top-hat function, with a CFL $= 0.5$, a grid range of $[x_0, x_{N-1}] = [0,1]$ with $N = 64$ cells. Set up periodic BCs, and integrate the equation over a single period ($\tau = 1/u$). For what values of $\Delta\tau$ is this

method stable? Define ϵ the same way as we defined in class, and make a simple conversion test and show how ϵ changes with N. Supply plots of the initial with the final states in your solution and a plot of $\epsilon(N)$.

b) The leap-frog scheme for solving the advection equation uses centered derivatives for both terms:

$$\frac{a_i^{n+1} - a_i^{n-1}}{2\Delta\tau} = -u\frac{a_{i+1}^n - a_{i-1}^n}{2h} \tag{4}$$

Note that this a three time-level scheme as it uses a_i^{n+1}, a_i^n, a_i^{n-1}.To get it started you need to use for the first time step a different scheme (e.g., Lax, see Garcia 7.1). Repeat the same exercise as before using the leapfrog.

3) *The advection equation in 1D with finite volume* (based on Zingale ex. 5.1). The finite volume conservative method to solve the advection equation is:

$$\frac{a_i^{n+1} - a_i^n}{\Delta\tau} = -u\frac{a_{i+1/2}^n - a_{i-1/2}^n}{h} \tag{5}$$

Use this method to solve the advection equation. For the Reimann problem on the cell interfaces use a uniform approximation of a_i^n (first order accuracy) and a linear piecewise approximation (second order accuracy, here you will need two ghost cells). Repeat the same exercise as before on both ways of solution.