ASTR 4260: Problem Set #4

Due: Wednesday, October 11, 2023

In this problem set, we will solve a set of ordinary differential equations to model the structure of solid planets. We assume that the planet is perfectly spherical, so that the pressure $P(r)$ and density $\rho(r)$ at any point of the planet depends only on the distance to the center of the planet (the radius r). If we consider a spherical shell of the planet to be in mechanical equilibrium, with pressure balanced against gravity, we obtain the equation of hydrostatic equilibrium:

$$\frac{dP(r)}{dr} = -\frac{Gm(r)\rho(r)}{r^2} \tag{1}$$

where G is the gravitational constant, $m(r)$ is the mass of the planet interior to r and the minus sign indicates the pressure decreases as r increases. The enclosed mass $m(r)$ satisfies the differential equation

$$\frac{dm(r)}{dr} = 4\pi\rho(r)r^2. \tag{2}$$

These two equations are two coupled first order differential equations that determine the structure of the planet for a given equation of state relating the density ρ to the pressure P. One example of an equation of state is the familiar ideal gas relation

$$\rho(P) = \frac{\mu}{kT}P. \tag{3}$$

However, terrestrial ("rocky") planet interiors are not well described by this relation (although it is appropriate for some gas giants). Instead, we will use a fit to experimental data from Seager et al (2007, ApJ 669, 1279), which is of the form:

$$\rho(P) = \rho_0 + cP^n \tag{4}$$

where ρ_0, c and n are constants that depend on the material being modeled. For example, for iron (Fe), $\rho_0 = 8300$ kg m^{-3}, $c = 0.00349$ kg m^{-3}Pa^{-n} and $n = 0.528$. Other materials can be found in Table 3 of the above paper.

The values of the dependent variables at $r = 0$ are: $P = P_c$ (the central pressure), and $m = 0$. Integrating outward in r gives the density profile, the outer radius R, being determined by the point at which the pressure goes to zero. The total mass of the planet is then $M = m(R)$. Since both the mass and the radius depend on the central pressure, P_c, variation of this parameter allows planets of different masses to be studied.

Problem 1

Write an ODE solver that implements the fourth-order Runge-Kutta technique, and use it to solve this set of ODE's. Although you can do just the two equations, if you can generalize to solving any number of equations with a separate module that calculates the derivatives for any particular problem and number of equations, you will save a lot of time in a subsequent problem set (or at the end of this set for graduates).

You can start by writing a routine that just takes one RK4 step for all the functions, and then iterate calls to it. Note that, to do the integration, you will need to choose a radial step Δr – this should be some small fraction of an earth radius in the correct units (since we expect "rocky" planets to be approximately earth sized).

Note that you don't know the final radius of the planet in advance—instead, when the radius r of your integration is too large, the pressure will go negative. You should stop the integration (just) before that happens. One way to do this is to check if the pressure will go negative at $P(r + \Delta r)$ before you take a step of size Δr with your integrator, and, if so, stop.[1] Finally, note that you will not be able to evaluate dP/dr exactly at $r = 0$, so you will need to start the integration at some small r (say, $r = \Delta r$)[2]

In order to implement the required equations of state (which require parameters that are inconvenient to pass through the argument list), you could use any of the solutions we discussed in class. One recommended way is to define a class for the materials. Instance attributes can represent the changeable constants for each material. Class methods can get and set those attributes for different materials, as well as compute the equation of state by taking a pressure as input and returning a density as output using those constants. See sections 7.1.1-2 in the *Primer on Scientific Programming* (or any discussion of object-oriented programming in Python) for more discussion of this idea.

Apply your routine to the case $P_c = 10^{12}$ Pa, using Fe for the equation of state, plot the variation of Pressure vs. radius and density vs. radius, and report your final radius (where the pressure goes to zero), and mass, in terms of the Earth's radius and mass.

Problem 2

Repeat the above integration for a range of central pressures to find the relation between planet *total* mass M and the *outer* radius R, again for Fe. Pick a range of central pressures such that the total mass corresponding to that central pressure, after integrating, covers the range from 0.1 to 100 earth masses. Plot the resulting radius - mass relation (in units of earth radii and earth masses, using a logarithmic scale). Is it easier to detect high mass planets or low mass planets through the transit technique (and why)?

Problem 3

Repeat problem 2, but instead of Fe, use H_20 and $MgSi0_3$ (see Table 3 of the Seager et al. paper referenced earlier). Plot the three resulting M-R relations (for the three materials

[1] Alternately, instead of stopping, you could adaptively reduce Δr before taking the step – this will allow you to determine the edge very accurately, although note that eventually you will have to stop reducing Δr.

[2] You can evaluate the required value of $m(\Delta r)$ by assuming the density/pressure is constant over that first step.

mentioned) and also plot points for the solar system's terrestrial planets (Mercury, Venus, Earth and Mars), and some exoplanet data – see, for example, Table A.1 of Otegi et al. (2020, A&A, 634, A43). You don't have to plot all of those exoplanets – a small selection across the mass range is fine. Use the routine `matplotlib.pyplot.errorbar` to plot the errors in mass (note that you'll want to specify something like `linestyle = 'None', marker = 'x', color = 'red'`, since `errorbar` doesn't take specifications like `'rx'` like `plot` does). Comment on the results that you find.

Problem 4: Graduates

Another alternative to improve the Euler method and ensure energy conservation is the Euler-Cromer or semi-implicit Euler method. This uses the time-advanced value of the velocity to compute the new position. That improvement makes the method *symplectic*, which conserves the Hamiltonian of the approximate system, with positive consequences for energy conservation.

(1) Demonstrate this by taking the standard Euler method presented in class and computing the nonlinear pendulum with it. Then modify the method to use the new value computed for the velocity $d\theta/dt$ to compute the position θ at each time step. Compare the behavior for a problem with the same parameters as presented in class.

(2) Run the same pendulum simulation with the RK4 solver you wrote for problem 1. Compare energy conservation for Euler, Euler-Cromer, and RK4. Run for long enough time and with large enough time steps to clearly differentiate the behavior of the different algorithms. The energy for this system is

$$E = \frac{1}{2} m\ell^2 \dot{\theta}^2 + mg\ell(1 - \cos\theta). \tag{5}$$

How do the different methods behave if you vary the resolution? Plot your results and draw conclusions from them about accuracy and energy conservation of the different methods.