

0.1 Homework 6

Due: Monday, October 5, 2009, 10:00 pm

Note that this week we did not make testing the errors and figuring out an appropriate time step size an explicit part of these problems. However, by now you are advanced enough to know that you are expected to check these things. In your report, describe how you tested your programs, and how you determined an appropriate time step size, and make an estimate of the likely errors in the results you report.

This assignment requires you to use three different expressions for the drag force. As before, you can save a lot of trouble by putting these in separate functions, and plugging in the one you want. Your code should use a function for the acceleration (i.e., dv/dt), but it is ok to have dx/dt coded into the ODE solver, as this is generic for solving Newton's laws.

1) Write programs to use the second order Runge-Kutta method to simulate the motion of a falling body. Do this both for drag linear in the velocity and drag quadratic in the velocity. See the class notes for a discussion of the equations you must solve.

Use your programs to compute the time required for an object to fall to the ground for each drag form. The object falls from a height of 30 meters, starting at rest. Use a terminal velocity of 25 m/s for both the linear and quadratic drag. Note that, just as in the case of the coffee cooling last week, to get an answer with the desired accuracy you will have to determine (approximately) where in the last time step the object hit a height of zero. There is more discussion of this in last week's class notes — it's the "last step problem".

2) For the case of drag linear in the velocity, compare your answers to the expected exact answer. Use your root finding program from two weeks ago to find the time at which the height is zero in the exact solution.

3) Two identical balls start from a height $h = 10$ meters. One starts at rest, and falls straight down. The second starts with an initial horizontal velocity of 30 m/s. Use a terminal velocity of 10 m/s for both the linear and quadratic drag. Write a program to calculate the motion of both balls for each of three cases — no drag, linear drag and quadratic drag. (For no drag, you can just set the drag constant to zero in your program. In fact, you should have already done that as a simple test of your program.) Without drag, which of the two balls hits first? With linear drag, which of the two hits first? And, with quadratic drag, which ball hits first? Explain why you get these answers.

Note that you have to do something new here, namely handle two dimensions at once. For a single ball, you will now have four variables: two components of the position, and two components of the velocity. One possible solution would be to have separate functions to return the x and y accelerations. A better solution is to have a

single function but use arrays, pointers, or structures to allow that function to return both the x and y acceleration.

For this week, it is ok to have the program solve one ball at a time and compare the results by hand. Next week, we will start to handle multiple bodies.

4) Add Philsplot calls to your solution to part 3 so as to animate the motion of the ball.

5) The world record for a free-fall parachute jump was a jump from a height of 24,483 meters. The skydiver then fell to a height of about 1000 meters, where he opened his parachute. Modify your program to calculate the fall of this skydiver up to the point his parachute opened. Assume that the drag force is quadratic in the velocity and decreases with altitude because the density of the air decreases. In fact, the density of the atmosphere is approximately proportional to $\exp(-y/y_s)$, where the “scale height”, y_s , is approximately 8000 meters. Specifically, assume that

$$|F_{drag}| = mg \left(\frac{v}{v_{term}} \right)^2 \exp(-y/y_s) \quad (1)$$

where the terminal velocity (at sea level) is $v_{term} = 65$ meters/second.

Make a graph of the skydiver’s speed as a function of time; to do this, you may use Philsplot calls or write out a table for use in a graphing program. What is the maximum velocity reached by the skydiver? At what height does he reach his maximum velocity? Calculate the time required to reach the height of 1000 meters.

When computing the maximum velocity, it is enough to quote the maximum value at a time step. That is, you do not need to worry about interpolating between the time steps in order to find a slightly better maximum (unlike how we have to handle things like finding the time to match a given height). Simply keep a variable that contains the biggest velocity to date, and in each step, check whether the new velocity is bigger and if so update the variable.