Computing project: Planet (sheet 2)

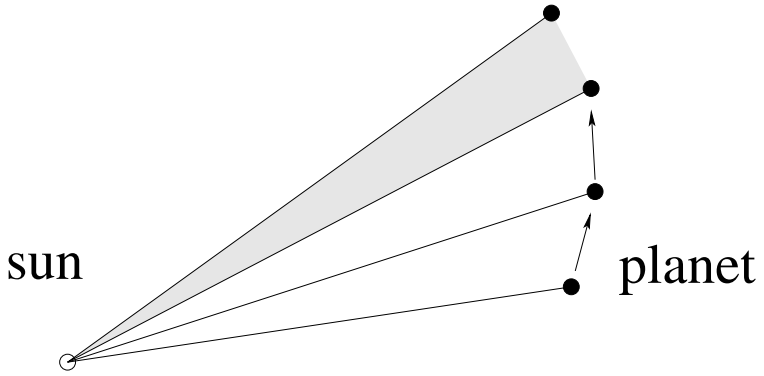
Some suggestions to help you think about the physics

1. **Remember where the sun is.** The acceleration is always directed towards the sun. Add to your graph of x_1 versus x_2 a yellow point at that location.

2. It is important that your plot of x_1 versus x_2 has '`equal`' axes. (See sheet 1.)

3. Think about the speed of the planet. Does the speed of the planet vary? **Display the velocity in your plots**. One way to visualize the speed of the planet is to plot *points* showing where the planet was at times $t = 0, 10, 20, 30, \ldots$. In what situations does the planet's speed increase? In what situations does it decrease?

4. Try drawing lines from the planet to the sun at times $t = 0, 10, 20, 30, \ldots$.



   Do you notice anything about the areas of the resulting triangles?

5. Here are some interesting functions that you could compute and plot.

   - The angular momentum of a mass about a point $\mathbf{o}$ is the cross product

     $$\mathbf{J_o}(\mathbf{x}, \mathbf{v}) = m(\mathbf{x} - \mathbf{o}) \times \mathbf{v}.$$

     In this problem, the angular momentum about the sun is an interesting quantity, so set $\mathbf{o}$ to the origin. You can define the mass of the planet to be $m = 1$ unit. And because $\mathbf{x}$ and $\mathbf{v}$ are two-dimensional, the angular momentum is a simple scalar

     $$J = m(x_1 v_2 - x_2 v_1).$$

   - The kinetic energy is

     $$T(\mathbf{v}) = \frac{1}{2} m \mathbf{v}^2.$$

   - The potential energy is

     $$U(\mathbf{x}) = -\frac{Am}{|\mathbf{x}|}.$$

     (You should confirm that this is consistent with the force, $\mathbf{f} = -\partial U / \partial \mathbf{x}$.

   - The total energy is

     $$E(\mathbf{x}, \mathbf{v}) = T(\mathbf{v}) + U(\mathbf{x}).$$

6. Optional extra: try changing the power in the inverse-square force law from 2 to another value such as 1.9.

Here are some programming suggestions

1. If your code is *modular*, you may find that you work more efficiently. For example, define your planet *simulation* in one script file or function file, and your *planet-plotting commands* in another file.

2. As Alan showed you, it is possible to define functions

   - by typing the function directly into octave;
   - by writing a script file and reading that script file into octave; or
   - by writing a function file. [1]

   When I use octave, I use *only the third method.* I define all my functions by creating function files.

   Each function file defines one function,
   and the name of the function is the same as the name of the function file.

   For example when I write a function `blah(a,b,n)`, I put it in a file called `blah.m`.

   This constrained programming style has some advantages:

   - Because I can have only one file called `blah.m` in a directory, I can have only one definition of a function in any given directory, so I don't get confused about which definition applies.
   - I don't have to remember to read in the script file to ensure a function is defined. Whenever I use a function name (for example `blah`), octave looks for the file `blah.m` and reads the function definition automatically.
   - I don't need to re-read a script file whenever I edit the function's definition.
   - When I come back and look at my programs in one month, I can easily find any function, because the filenames are the function names.

   One disadvantage of this programming style is it may require me to open lots of files and jump between them in my editor.

[1] What is the difference between a script file and a function file? You can find out by reading the octave manual on the computer. One difference is: a script file must not begin with the word `function`.