

# Computing project: Planet

Write an `octave` program that simulates a planet moving around a sun, assuming an inverse-square force law. The differential equations for the position  $\mathbf{x}$  and velocity  $\mathbf{v}$  are

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \quad (1)$$

$$\frac{d\mathbf{v}}{dt} = -\frac{A}{|\mathbf{x}|^3}\mathbf{x}, \quad (2)$$

where  $A = GM$  is the product of the gravitational constant and the mass of the sun.

## Note your dimensions

Next to *every* equation in your program, include a comment that says the dimensions of the quantities in the equation. (Comments begin with the ‘#’ character.) For example, a program that simulates motion of a particle in the absence of any force might look like this:<sup>1</sup>

```
# Initial condition for position and velocity
x = [ 93 , 0 ] ;
# x has dimensions [L]
v = [ 0 , 1.3 ] ;
# v has dimensions [L/T]

# Duration of simulation, and time-step; and initial time
T = 1000 ; dt = 5 ; t = 0 ;
# T and dt and t have dimensions [T]

while( t < T )
    t = t + dt ;
# [T] = [T] + [T]
    x = x + v * dt ;
# [L] = [L] + [L/T] * [T]
endwhile
```

Always check that the **two rules of dimensions** are true:

1. All quantities connected by =, +, -, >, or < must have the same dimensions.
2. All quantities appearing inside `log`, `sin`, `cos`, `exp` must be dimensionless.

Checking dimensions is a good habit because it helps you catch programming errors.

## What to do

Set the parameter  $A$  to:

$$A = 238 \text{ [L}^3\text{T}^{-2}\text{]}$$

---

<sup>1</sup>For information about `while` loops, which are even simpler than the `for` loops you learnt about last week, see the web page [http://www.aims.ac.za/wiki/index.php/Octave:Loops\\_and\\_conditions](http://www.aims.ac.za/wiki/index.php/Octave:Loops_and_conditions).

Try these initial conditions first

$$(a) \quad \begin{array}{rcll} \mathbf{x} & = & [93,0] & [L] \\ \mathbf{v} & = & [0,1.1] & [L/T] \\ dt & = & 1 & [T] \quad (\text{timestep}) \\ T & = & 1000 & [T] \quad (\text{duration}) \end{array}$$

Make two plots. Plot  $x_1$  versus  $x_2$ . And plot  $x_1$  and  $x_2$  as functions of time.

Then try changing the initial conditions like this

```
theta = 0.5 ; # angle for rotating the velocity, dimensionless [1]
M = [ cos(theta) sin(theta) ;
      -sin(theta) cos(theta) ] ; # rotation matrix, dimensionless [1]
```

$$\begin{array}{rcll} (b) \quad \mathbf{v} & = & [0,1.1] * M & [L/T] \\ (c) \quad \mathbf{v} & = & [0,1.6] & [L/T] \\ (d) \quad \mathbf{v} & = & [0,1.6] * M & [L/T] \\ (e) \quad \mathbf{v} & = & [0,2.0] * M & [L/T] \\ (f) \quad \mathbf{v} & = & [0,2.5] * M & [L/T] \end{array}$$

Investigate what happens if you make the timestep  $dt$  bigger or smaller.

## Tips

TIP 1: For the plot of  $x_1$  versus  $x_2$ , you may find the command

```
axis('equal')
```

is useful. This command tries to make the plot have equal size units on both axes. To switch this effect off again, use:

```
axis('normal')
```

TIP 2: To keep a record of the sequence of values of  $\mathbf{x}$  and  $\mathbf{v}$ , you can use an array like this:

```
i = 0 ; ## index for rows of the history array
clear history ; ## this makes sure there is nothing in the history
## array when we start the simulation
while( t < T )
    i ++ ;
    history( i , : ) = [ t , x , v ] ; ## put t, x, and v into columns
    ... ## 1, 2-3, and 4-5 of history
```

(Each row of the history array is a vector that contains quantities that have different dimensions. This does not break the rules of dimensions. It is OK to have a vector, such as  $(x, v)$ , whose components have different dimensions.)

TIP 3: If we define our time unit to be 1 day, and our distance unit to be the megamile (1 megamile =  $10^6$  miles) then the initial conditions in part (c) correspond to earth's orbit.

So for initial condition (c), the orbit should be roughly circular, with period 365.25 time-units.