Python 3 Tutorial 3: Writing a script to solve a problem

Patrick Champey & Jacob Heerikhuisen

May 31, 2017

1 An Orbit Around the Sun

We will compute the orbit of an object (assumed massless) around the Sun. You will want to create a new Python script. For this problem, you will want to make use of NumPy and MatPlotLib.PyPlot.

1.1 Setting up the problem

Next let's define the length of an astronomical unit (AU) and the number of seconds in a day:
au = 1.49598e11 # m
day = 60*60*24 # s

We are going to integrate the equations of motion in the xy-plane:

$\frac{dx}{dt} = v_x; \quad \frac{dy}{dt} = v_y; \quad \frac{dv_x}{dt} = a_x; \quad \frac{dv_y}{dt} = a_y$

The acceleration can be computed from the gravitational force:

$\vec{a} = \frac{(-GM)}{r^2}\vec{\hat{r}}$

where the vector r is a unit vector, and the mass of the object cancels out. A unit vector is a vector of length 1, obtained by taking the position vector $\vec{r}$ and dividing it by its length —r—. Using this information, and the fact that $\vec{r} = (x, y)$ gives:

$\vec{a} = (a_x, a_y) = \frac{(-GM)}{r^2}\frac{(x,y)}{|r|}$
or
$a_x = \frac{(-GM)}{r^3}x; \quad a_y = \frac{(-GM)}{r^3}y$

The four differential equations require four initial conditions in order to have a unique solution. Let's choose the following: xx = 1 * au; yy = 0; vx = 0; vy = 29800

As we integrate over the orbit, we need to keep track of the position of the object, so that we can plot out the orbit once it's been calculated. It's most efficient to "declare" the arrays where we will store the x & y positions up-front:

```
npoints = 1000
x_vect = np.zeros(npoints)
y_vect = np.zeros(npoints)
```

Finally, we need to choose a time-step for the integration procedure. dt = 1*day

1.2 Solving the integrals

We are going to use the simplest integration method, known as "Euler's method". The basic idea is that at each point in time you take a small step in the direction of the local velocity, and then update the velocity according to the local force. The rest of the program looks something like:

```
x_vect[0] = xx/au, y_vect[0] = yy/au

for k in range(1,npoints-1):

    ax, ay = orbit_func(xx, yy, vx, vy)
    xx = xx + vx*dt; yy = yy + vy*dt
    vx = vx + ax*dt; vy = vy + ay*dt
    x_vect[k] = xx/au; y_vect[k] = yy/au

plt.figure()
plt.plot(x_vect, y_vect, '-k')
plt.xlim(-1.5, 1.5)
plt.ylim(-1.5, 1.5)
plt.show()
```

Here we make use of an external function that computes the accelerations at a given point:

```
def orbit_func(x, y, vx, vy):
    G = 6.6742E-11 # m^3 kg^-1 s^-2
    M_s = 1.9889E30 # kg
    r = np.sqrt(x**2+y**2)
    F_factor = -G*M_s / r**3
    ax = F_factor*x; ay = F_factor*y
    return ax, ay
```

Run the program to see the orbit. Try changing the step size. Is a smaller step size more accurate? How can you tell?

The above method can be greatly improved by using a more accurate method. The next simplest is known as "second order Runge-Kutta". The method involves taking the average of the current velocity with an estimate of the "future" velocity:

```
xx1 = xx + vx*dt
vx1 = vx + ax*dt
xx = xx + 0.5*(vx + vx1)*dt
```

Similar terms are required for the y position and the x & y velocities the latter requiring additional accelerations ax1 & ay1 at the positions xx1 & yy1.

Compare the accuracy of the Euler & RK2 methods for a number of orbits and a number of time-step sizes.

2 Tasks

Use your code to answer the following:

1. Suppose a comet has a perihelion of 0.2 AU, and an aphelion of 100AU. What speed will the comet have at perihelion?
2. Suppose we want to send a spacecraft to Mars (assume a distance of 1.5 AU from the Sun). What is the minimum launch speed, relative to Earth, that could get the spacecraft to Mars?
3. What is the escape speed for the solar system?

Try the next two tasks if you have time.

4. Extend your code to two finite masses orbiting around each other.
5. Make a movie of the two orbiting masses.

Whoever figures out step 5, I will buy you lunch! I have not had much success with making movies in Python. Though, I'm confident there is a way, I just have not yet figured it out!