Modelling single and double body rocket orbits in Python

Peter Spencer-Smith

Introduction:

In this exercise the orbital motion of an object around a single and double body, is to be modelled using the 4th order Runge-Kutta method for solving the differential equation which describes the motion.

Runge-Kutta methods are improvements upon the Euler method and perform more evaluations at each step allowing for grater accuracy in the solution and increased complexity in the differential equation.

Equation (1) gives the general form of the 2nd order differential equation to be solved, with $\mathbf{x}$ a vector of multiple variables.

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, (\mathbf{x})) \qquad (1)$$

The 4th order Runge-Kutta method is then given by equations (2) & (3) with the K's being the standard Runge-Kutta coefficients.

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \qquad (2)$$

$$t_{i+1} = t_i + h \quad (3)$$

Part a) of the problem models the motion of a body in both a circular and elliptical, stable orbit around an earth like body. The differential equation (4) describes this motion and is solved by the RK4.

$$\ddot{\mathbf{r}} = -\frac{MG}{|\ \mathbf{r}\ |^3}\mathbf{r} \qquad (3)$$

Part b) models the motion of an object as it travels between an earth-moon system in a slingshot round trip, akin to that of the Apollo missions. In this problem the differential equation is more complex given by (4) and is again solved by the RK4.

$$\ddot{\mathbf{r}} = -\frac{M_E G}{|\ \mathbf{r} - \mathbf{R}_E\ |^3}(\mathbf{r} - \mathbf{R}_E) - \frac{M_M G}{|\ \mathbf{r} - \mathbf{R}_M\ |^3}(\mathbf{r} - \mathbf{R}_M) \qquad (4)$$

Part a: Single body orbit

The orbit of an object around the Earth was modelled for a circular and elliptic trajectory and the energy of the system was examined .

The program:

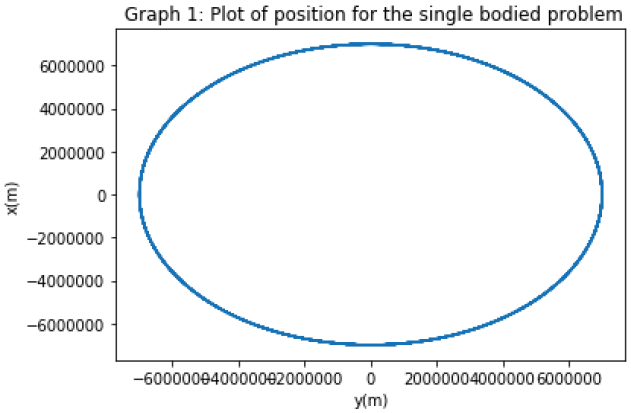
In order to model the orbit of the object around the earth a while loop was employed to iterate through each time step using the RK4. Four functions for x-y position and velocity were coded and then called within the loop to maximise the efficiency of the script.

Empty lists for position and energy were populated by the .append function within the loop and then used to plot the required orbital position and energy graphs.

An extra set of functions was written to calculate the initial velocity required for a stable orbit, adapted from the Vis-Viva equation. This value automatically then sets the initial conditions for velocity. A user generated pre-factor was added to provide an easy way of generating elliptical orbits.

Results & Discussion:

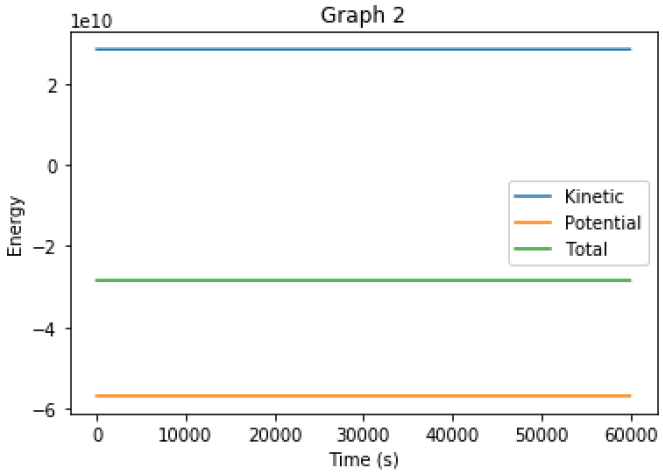
Graphs 1 & 2 display the position and energy plots, respectively, for a circular orbit. Here, the pre-factor is set to one and the automatically generated initial velocity for a



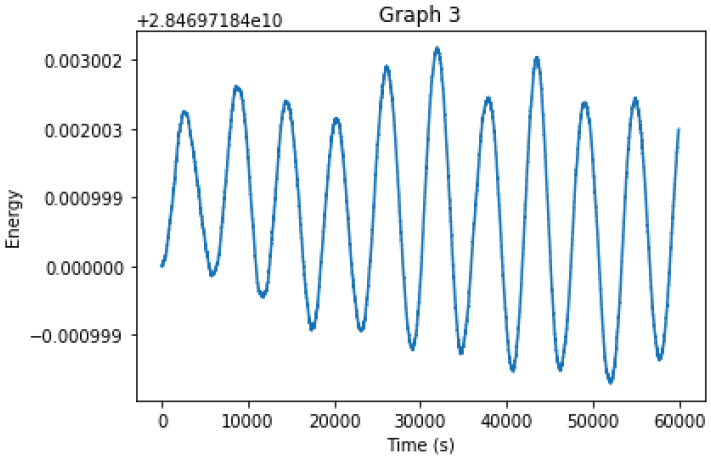
circular orbit is left unaltered.

By increasing the total duration of orbit by a factor of 10 from 6000-60,000 s, both

displayed in graph 1, we see no change in the orbital pattern and conclude that the orbit exhibits a high degree of stability. Providing re-runs of each scenario resulted, as expected, in identical results.

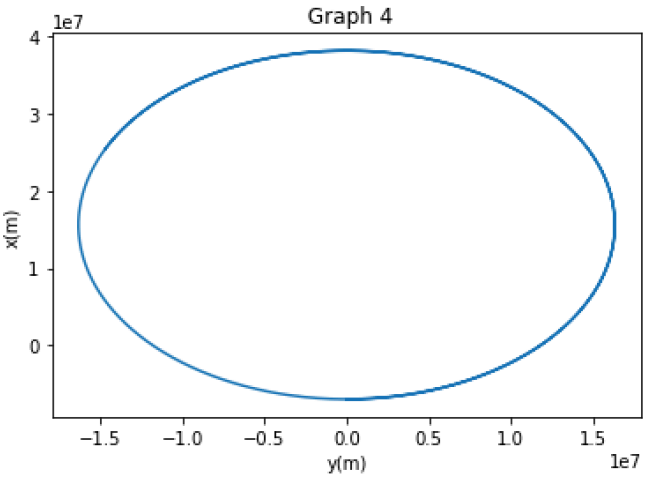


Graph 2 displays the potential, kinetic and total energy for the circular orbit. Each of the three quantities remain constant throughout the orbital period which is conducive with the conservation of energy for a circular orbit. The time step was set to 0.5's, which provided an accurate representation of the constant energy values of the system

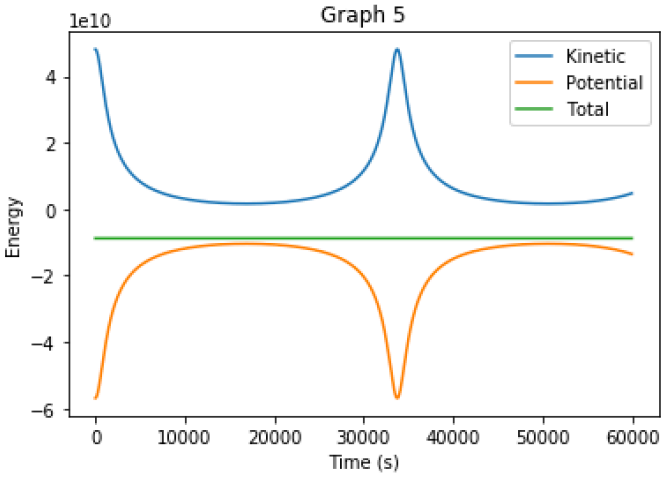


By altering the time step from 0.5 to 1's the kinetic energy of the system loses accuracy, as displayed in graph 3.

In order to create an elliptic orbit the cofactor was increased to 1.3. Again the stability of the system is evident from graph 4, which displays overlays of orbits over different time periods.



Now in the elliptical regime, conservation of energy is displayed in graph 5, which shows the oscillation of kinetic and potential energy which sum to give a constant, conserved, total energy.



Part b: Slingshot to the Moon and back!

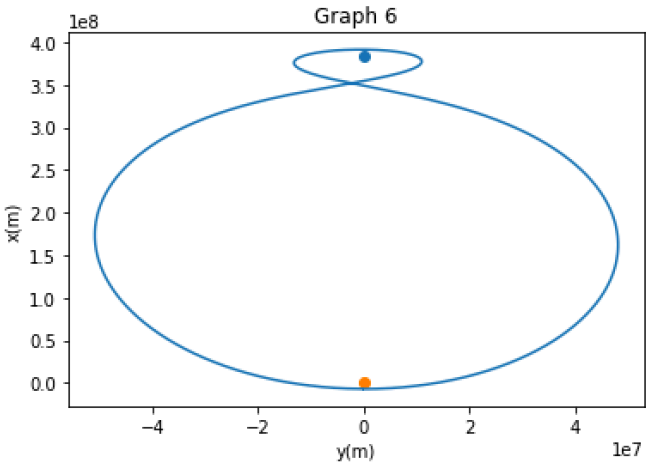
The journey of the object as it slingshots between the earth and moon was successfully modelled and various initial conditions explored in order to stabilise the trajectory.

The program:

The architecture of the script used in part a) was employed again with the alteration of the functions used to model the position of the object. Equation (4) was adapted and

implemented in functions 3 and 4, where the position of the earth was set to (0,0) and the Moon aligned with the y-axis.

Graph 6 displays the path taken by the object as it journeys between the Earth and Moon in a figure of eight pattern.



The initial conditions for the journey were set by positioning the object on the far side of the Earth, with a positive - horizontal x-velocity. The initial velocity was set through a process of trial and error, iterating upon a stable, circular Earth orbit, by a pre-factor. The pre-factor was found to be 1.3995 and the stable Earth orbital velocity was taken from the results of part a).

It was found that the closest approach to the moon possible, without crashing, was given by the pre-factor given above.

The time taken for the one complete trip was found to be 845000's, taken from when the object returned to the far side of the Earth. This time duration was set as the end condition for the journey and was coded as the break clause within the while loop when it reached the number of time steps required.

Conclusion:

The motion of an object in a circular and elliptic orbit was successfully modelled using the RK4 method. The energy of the system was examined and conservation principles were demonstrated where the correct time step was employed. A two bodied slingshot orbit was also achieved by the RK4 method and various parameters explored to achieve a successful simulation.