Orbital motion - Eris: Jessica Murphy

06/03/2023

Homework 18

This python programme involves an Initial Value Problem where it uses a technique to model the orbital motion of the dwarf planet Eris around the Sun - it is called the **Euler-Cromer** technique and is implemented through **linked differential equations**.

Discovered in 2005, the dwarf planet Eris follows a highly eccentric orbit. At its perihelion (the point where it is closest to the Sun) it is 38.3 AU from the Sun and is travelling with a velocity of 1.22 AU yr^{-1}

The motion is described by the linked second order ODEs:

$$\frac{d^2x}{dt^2} = -\frac{GM}{r^3}x$$

$$\frac{d^2y}{dt^2} = -\frac{GM}{r^3}y$$

$$r = \sqrt{x^2 + y^2}$$

They are linked because the x-acceleration depends on x and y (via r)and the y acceleration depends on both x and y. A python programme is written which models the behaviour of the system over the next 1000 years. The time unit used is years and the distance unit used in the Astronomical Unit, AU.The value of GM is then $4\pi^2$ AU^3 yr^{-2}. The variables invlolved here are the distances of Eris to the Sun in the x and y directions, the velocity in the x and y directions and time.

The initial values of the variables are stated, a time step is fixed, arrays to hold the changing variables of x, y, v_x, v_y, and t are created, and a for loop is determined to iterate along the t-axis to model the behavior of the planets orbit. To look at the shape of the orbit a graph of y is plotted against x. The aphelion distance is determined in AU, this is the furthest radial distance Eris orbits the Sun at a specific point on the x-y plane.

In a new cell, the kinetic energy per unit mass and the gravitational potential energy per unit mass are calculated and graphed. Energy per unit mass will be measured in AU2 yr^{-2}.

The equations for these are as follows:

$$\frac{KE}{m} = \frac{1}{2}({v_x}^2 + {v_y}^2)$$

$$\frac{PE}{m} = -\frac{GM}{r}$$

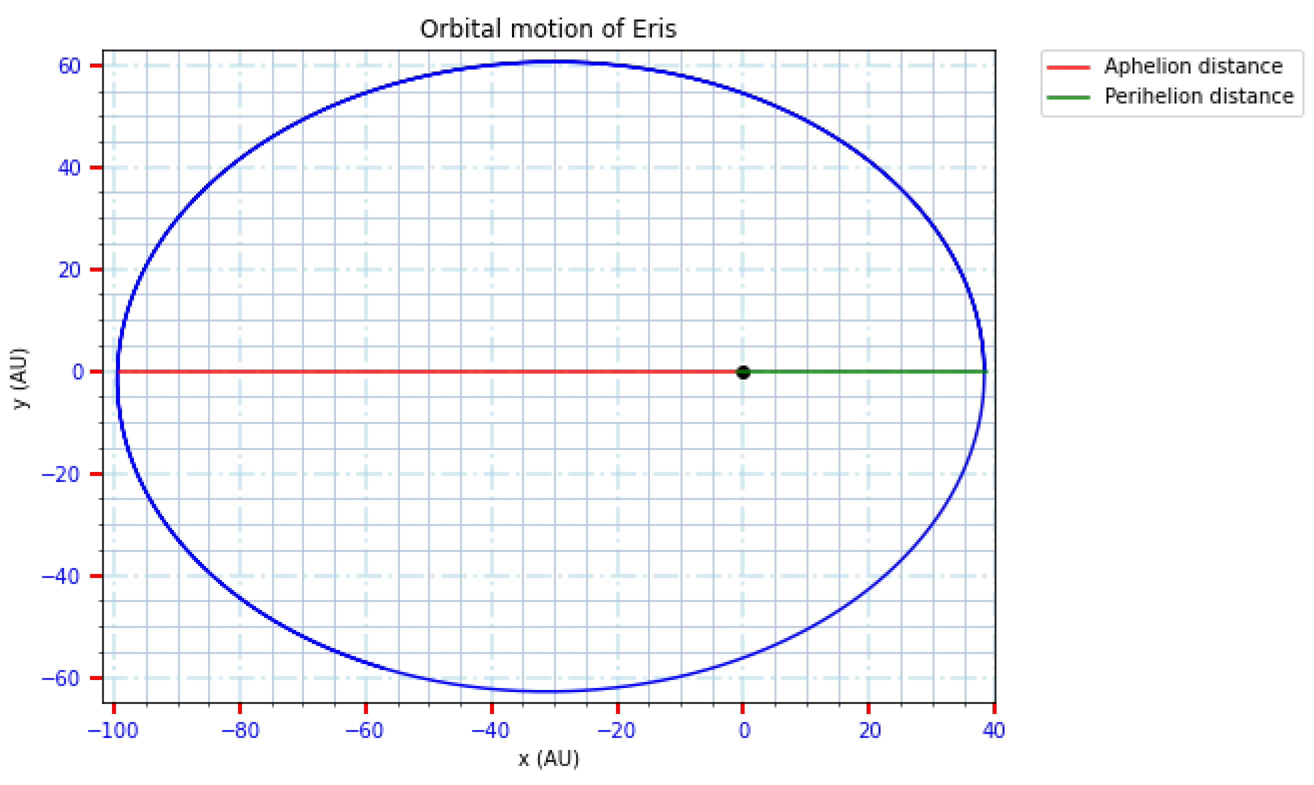
A few remarks about energy conservation are made about the graph at the end of the notebook. The orbital period is determined from the graph.

1. Import the python library NumPy
2. Display the equation for GM which is $4\pi^2$ AU^3 yr^{-2}
3. Set the number of steps required
4. Set up NumPy arrays to hold the distances (***x***) and (***y***), velocity(***v_x***) and (***v_y***) and time (***t***) values and initialise.
5. Use a for loop to step along the t-axis. Use the Euler-cromer technique to obtain a generalised formula to simulate more accurate physical conditions by using the velocity at the end of the time interval, when estimating the distance. Do this for x and y directions.
6. Plot the numerical solution on a graph of the x-y plane to show the shape of the orbit.
   - import matplotlib
   - print aphelion calculated value, display perihelion to show accuracy of graph
7. Calculate the kinetic energy per unit mass, the gravitational potential energy per unit mass and the total energy per unit mass
8. Plot these energy terms as a function of time on the same graph
9. Discussion of energy conservation and determine the orbital period from graph

In [1]:

```python
# Orbital motion using Euler-Crometechnique
# The linked 2nd order ODEs are:
# d^2x/dt^2 = -GM x/(r^3), d^2y/dt^2 = -GM y/(r^3)

import numpy as np # import python libraries as needed

GM = 4 * np.pi**2 # GM in units of AU^3 yr^-3

t_max = 1000 # iterate up to a maximum of t_max in years
delta_t = 1 # set the time step (years)
N = int(t_max/delta_t) # calculate the number of time jumps

# set up numpy arrays to hold the displacement x, y, velocity v_x, v_y and time t values
# need one more element in each array than the number of jumps, to include the zeroth element
x = np.zeros(N + 1)
y = np.zeros(N + 1)
v_x = np.zeros(N + 1)
v_y = np.zeros(N + 1)
t = np.zeros(N + 1)

# initialise the zeroth values, explicitly stating variables y, velocity in x direction and t start at 0
x[0] = 38.3
y[0] = 0
v_x[0] = 0
v_y[0] = 1.22 # a non-zero, tangential velocity to go around sun once in a year
t[0] = 0

# use a for loop to step along the t-axis N times and apply the Euler-Cromer technique
# notice symmetry in x acceleration and y acceleration - linked ODEs
for i in range(N):
    r_cubed = (x[i]**2 +y[i]**2)**(3/2) # calculate r^3 for cuyrrent x and y positions

    a_x = -GM * x[i] / r_cubed # linked ODE for x acceleration
    v_x[i+1] = v_x[i] + a_x *delta_t # Estimate v_x at the end of the interval
    x[i+1] = x[i] + v_x[i+1] * delta_t # Estimate x at the end of the interval, Euler-Cromer technique applied to the index of v_x
                                       # estimates x at the end of the time interval using v_x at the end of the time interval

    a_y = -GM * y[i] / r_cubed # linked ODE for y acceleration
    v_y[i+1] = v_y[i] + a_y *delta_t # Estimate v_y at the end of the interval
    y[i+1] = y[i] + v_y[i+1] * delta_t # Estimate y at the end of the interval, Euler-Cromer technique applied to the index of v_y
                                       # estimates y at the end of the time interval using v_y at the end of the time interval
    t[i+1] = t[i] + delta_t# calculate t at the end of the interval

# import matplotlib library to plot graph
import matplotlib.pyplot as plt

#plot the y against x
plt.figure(figsize=(8,6)) # increase the figure size
plt.plot(x,y, 'b-') # no time value as on x-y plane
plt.plot(0, 0, 'ko') # make the location of the Sun at the origin
plt.axhline(y= 0, xmin=.02, xmax=.71, color = 'r',linestyle='-', label = 'Aphelion distance') # add a line to show the max distance
plt.axhline(y= 0, xmin=.71, xmax=.99, color = 'g',linestyle='-', label = 'Perihelion distance') # add a line to show the min distance
plt.xlim(-102,40) # optimise graph area
plt.ylim(-65, 63)
plt.title('Orbital motion of Eris') # title graph
plt.xlabel('x (AU)')
plt.ylabel('y (AU)')
plt.tick_params(axis='both', direction='out', length=6, width=2, labelcolor='b', colors='r', grid_color='gray', grid_alpha=0.5) # make axes ticks red and writing blue
plt.grid(visible=True, color='lightblue',linestyle='-.', linewidth=2) # major grid lines
plt.minorticks_on()
plt.grid(visible=True, which='minor', color='lightsteelblue', alpha=0.8, ls='-', lw=1) # minor grid lines
plt.legend(bbox_to_anchor = (1.05, 1), loc = 'upper left', borderaxespad=0) # adding legend
plt.show()

# state the radius r, the distance Eris orbits around the sun at a given point in x-y plane
```

```python
r = (x**2 + y**2)**0.5
# plot the aphelion distance in AU - futhest from sun, and point where it is closest to the sun - perihelion
print('The aphelion distance is: {0:4.2f} AU'.format(max(r)))
print('The perihelion distance is: {0:4.2f} AU'.format(min(r))) # double checks if programme is correct
```

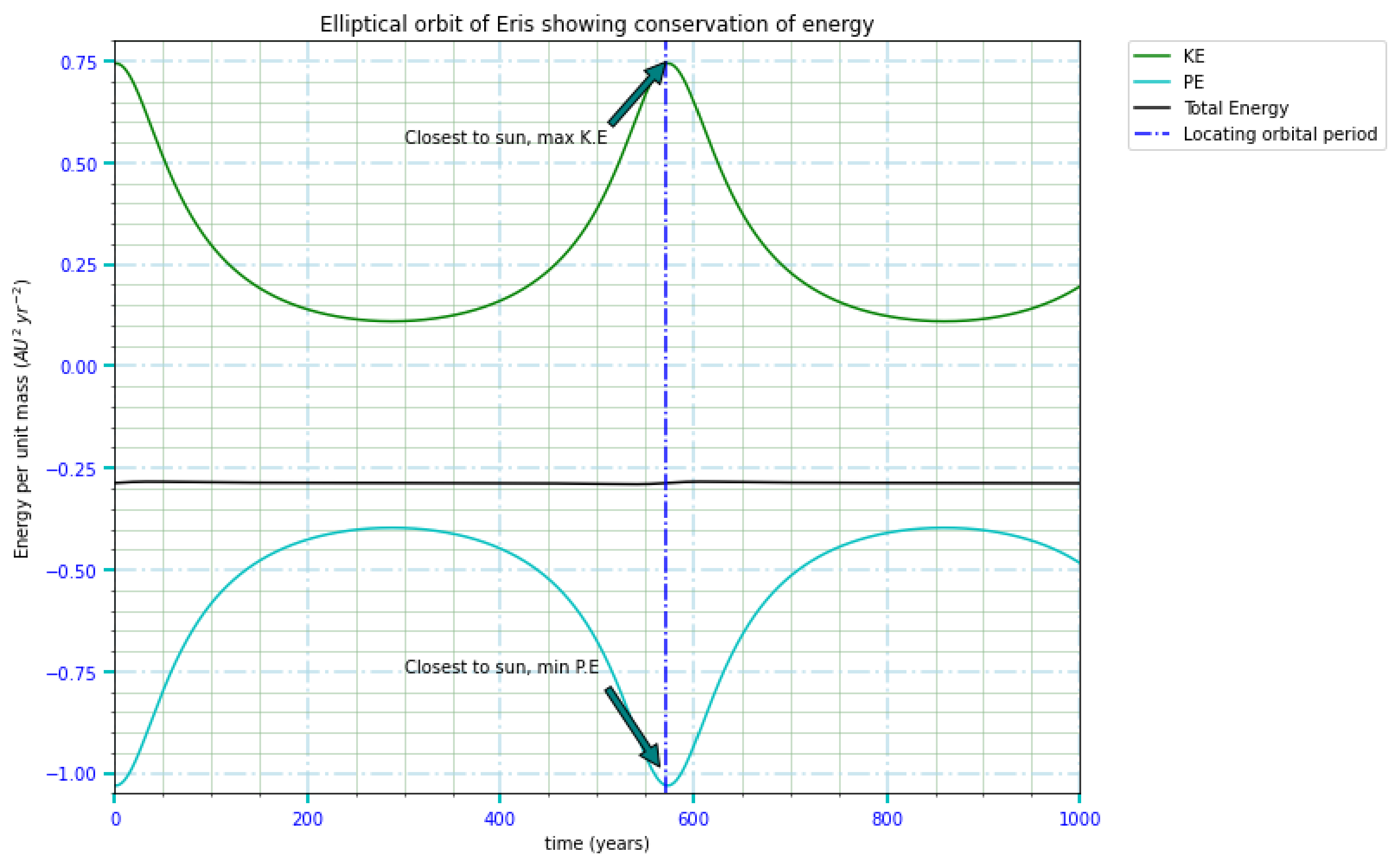


```
The aphelion distance is: 99.55 AU
The perihelion distance is: 38.29 AU
```

Kinetic energy per unit mass and the gravitational potential energy per unit mass

In [2]:

```python
# calculate the kinetic energy per unit mass, the gravitational potential energy per unit mass and the total energy per unit mass
# Energy per unit mass will be measured in AU^2 yr^-2

KE = 0.5 * (v_x**2 + v_y**2) # kinetic energy per unit mass
PE = (- GM) / r # potential energy per unit mass
E_total = KE + PE # the total energy per unit mass

# plot KE, PE and total energy as a function of time in years
plt.figure(figsize=(10,8)) # increase the figure size
plt.plot(t, KE, 'g-', label='KE') # plot KE
plt.plot(t, PE, 'c-', label='PE') # plot PE
plt.plot(t, E_total, 'k-', label='Total Energy') # plot total energy
plt.title('Elliptical orbit of Eris showing conservation of energy') # title graph
plt.xlabel('time (years)')
plt.ylabel('Energy per unit mass ($AU^2$ $yr^{-2}$)')
plt.axvline(x = 572, color = 'b',linestyle='-.', label = 'Locating orbital period') # establish a line that determines the orbital period of Eris, at peak of KE and min of PE
plt.xlim(0, t_max) # maximise area of graph
plt.ylim(-1.05, .8)
plt.tick_params(axis='both', direction='out', length=6, width=2, labelcolor='b', colors='c', grid_color='gray', grid_alpha=0.6) # make axes ticks cyan and writing blue
plt.grid(visible=True, color='lightblue',linestyle='-.', linewidth=2) # major grid lines
plt.minorticks_on()
plt.grid(visible=True, which='minor', color='darkseagreen', alpha=0.8, ls='-', lw=0.6) # minor grid lines
plt.legend(bbox_to_anchor = (1.05, 1), loc = 'upper left', borderaxespad=0) # adding legend
plt.annotate('Closest to sun, min P.E', xy=(572, -1.01), xytext=(300, -.75), # adding arrow to point where the minimum potential energy is
             arrowprops=dict(facecolor='teal', shrink = 0.1),
             )
plt.annotate('Closest to sun, max K.E', xy=(572, 0.75), xytext=(300, .55), # adding arrow to point where the maximum kinetic energy is
             arrowprops=dict(facecolor='teal', shrink = 0.01),
             )
plt.show()
```



Determining from the graph, it is seen that energy is conserved because of the horizontal, flat line showing **destructive interference** . This shows where there is a lack in kinetic energy, there is additional potential energy and visa versa. Shows a stable orbit.

The orbital period of Eris is approximately 572 years, read from the graph.