Orbital Motion - Eris

Kaylin Shanahan 2023

This is a python programme which uses the Euler-Cromer technique to model the behaviour of a system over the next 1000 years. This system consists of the dwarf planet Eris and the Sun. The programme will output a graph of the orbit on an x-y plane, from which the aphelion distance and a number of other factors can be determined.

The dwarf planet this programme will be modelling follows a highly eccentric orbit. Its perihelion is 38.3 AU from the sun and it has a velocity of $1.22\ AU\,yr^{-1}$. Its highly erratic motion can be described using the following linked second order ODEs:

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

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

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

Where r is the orbital radius, G is the universal gravitational constant and M is the mass of the sun. In this case, the value of GM will be $4\pi^2\,AU^3\,yr^{-2}$.

The kinetic energy per unit mass and the gravitational potential energy per unit mass will also be determined by this programme, which will output calculte and then output a plot of these two energy terms, along with the total energy per unit mass, as a function of time. From this, whether or not energy is conserved can be determined. These energy terms can be determined using the following equations:

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

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

Finally, the orbital period of Eris will be determined using the above equations and output by this programme.

- Input initial conditions

- Input function to determine the number of steps required
- Initialise displacement, velocity and time values
- Calculate r^3 acceleration, velocity and displacement in each dimension using the Euler-Cromer technique
- Output graphs of x versus y and y versus x to display orbital motion
- Calculate and output the aphelion distance
- Calculate kinetic, gravitational potential and total energies
- Output graph of kinetic energy, gravitational potential energy and total energy as a function of time
- Determine whether energy is conserved
- Calculate and output orbital period of Eris around the Sun

In [228]:

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

# Import numpy for use in mathematics
import numpy as np

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

t_max = 1000             # Iterate up to a maximum time of t_max (yrs)
delta_t = 1              # Set the time step (yrs)
N = int(t_max/delta_t)   # Calculate the number of jumps

# Set up Numpy arrays to hold the displacement x, y, velocity v_x, v_y and time t values
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)
```

For the initial conditions for Earth, we start by placing the Earth on the x-axis at $x = 38.3$ and $y = 0$.

The velocity will be tangenital to this and have a magnitude of $1.22\ AU$ per year.

In [229]:
```
# Initialise the zeroth values of displacement x, y, velocity v_x, v_y and time t values
x[0] = 38.3 # AU
y[0] = 0.0
v_x[0] = 0.0
v_y[0] = 1.22 # AU per yr
t[0] = 0.0
```

Next the Euler-Cromer loop must be created. The r^3 value is calculated first, which is common to both differential equations. Then, acceleration, velocity and displacement in each dimension is calculated.

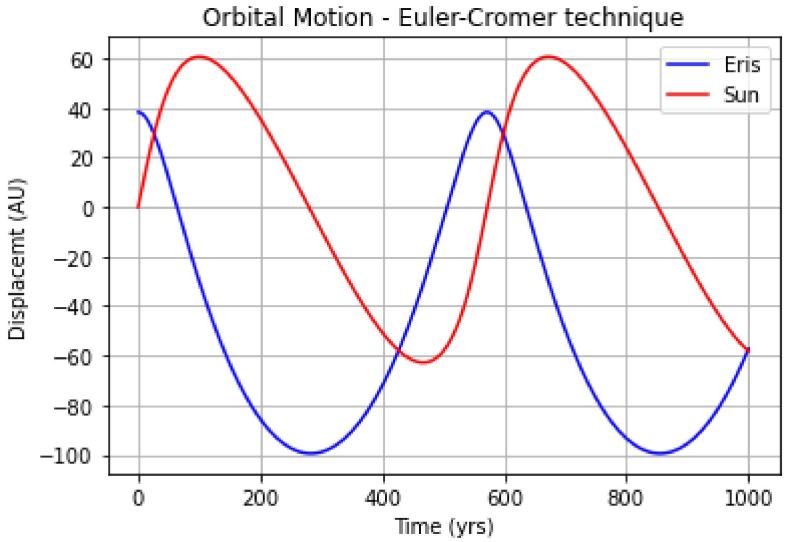
The same for loop must be used for the x-dimension and y-dimension calculations due to the linked nature of the ODEs.

In [230]:
```
# Use a for loop to step along the t-axis and apply Euler-Cromer technique
for i in range(N):
    r_cubed = (x[i]**2 + y[i]**2)**(3/2)   # Calculate r^3 for the current x and y values

    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

    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 x at the end of the interval

    t[i+1] = t[i] + delta_t                # Calculate t at the end of the interval
```

Firstly, the results can be plotted as in the below graph, as in one-dimensional systems, by plotting x and y as functions of time. This will produce one and a half wave cycles for 1000 years of motion.

In [231]:

```
import matplotlib.pyplot as plt

# Plot the x and y values against time
plt.plot(t, x, "b-", label="Eris")
plt.plot(t, y, "r-", label="Sun")
plt.title("Orbital Motion - Euler-Cromer technique")
plt.xlabel("Time (yrs)")
plt.ylabel("Displacemt (AU)")
plt.grid()
plt.legend()
plt.show()
```

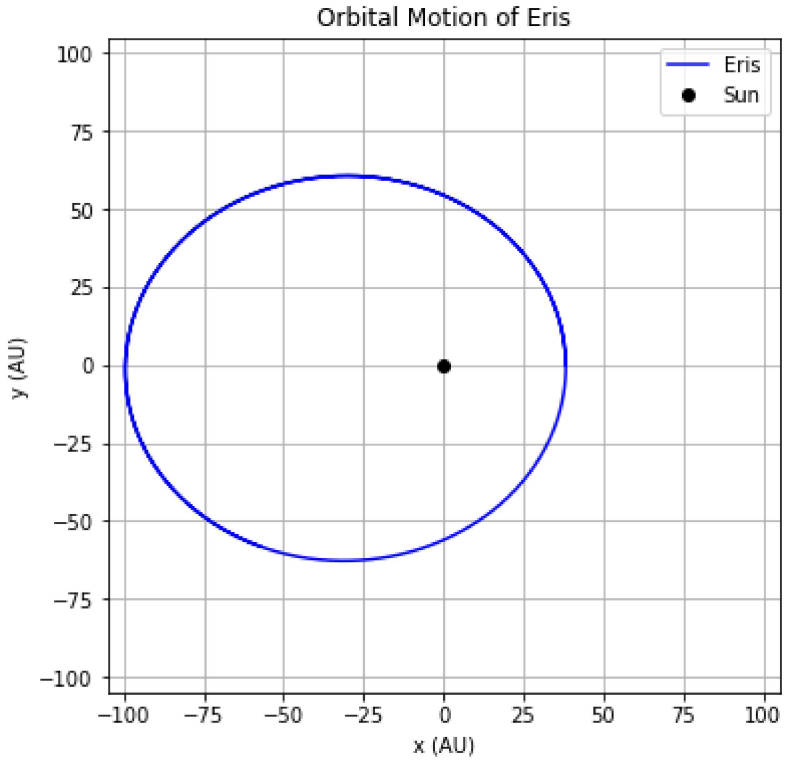


Secondly, the results can be plotted to observe the shape of the orbit by plotting y against x. This will produce an elliptical orbit; a stable orbit with the Sun at one of the foci.

From the graph below, it can be noted that the velocity of Eris, in an elliptical orbit, is not constant. Thus, the kinetic energy is also not constant.

In [232]:

```
# Plot y against x
plt.figure(figsize=(6,6))             # Make the plot square
plt.plot(x, y, "b-", label="Eris")    # The orbital path taken by the planet Eris
plt.plot(0, 0, "ko", label="Sun")     # The location of the Sun at the origin
plt.xlim(-105, 105)                   # The range of x-values diplayed, slightly larger than that of the or
plt.ylim(-105, 105)                   # The range of y-values diplayed
plt.title("Orbital Motion of Eris")
plt.xlabel("x (AU)")
plt.ylabel("y (AU)")
plt.legend()
plt.grid()
```



In [233]:

```
# Determine the aphelion distance, the maximum distance between Eris and Sun,
# which is equal to the distance from the origin to the minimum of the function
print("The aphelion distance of Eris and the Sun is: {0:10.2f} AU.".format(abs(x.min())))
```

```
The aphelion distance of Eris and the Sun is:      99.52 AU.
```

In [234]:

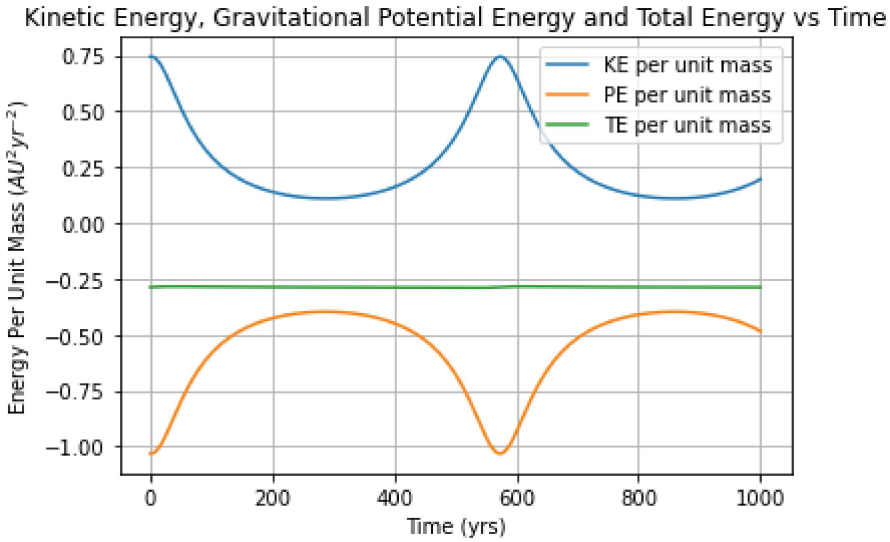
```
# Determine r, using equation for orbital radius
r = (x**2 + y**2) ** (1/2)

# Kinetic energy per unit mass
KE_per_m = (1/2) * ((v_x**2) + (v_y**2))
KE = np.array(KE_per_m)

# Gravitational potential energy per unit mass
PE_per_m = - GM / r
PE =  np.array(PE_per_m)

# Total energy per unit mass
TE_per_m = KE_per_m + PE_per_m
TE =  np.array(KE_per_m + PE_per_m)
```

Kinetic, Gravitational Potential and Total Energy

The planet, Eris, on its elliptical orbit of the sun, converts energy back and forth between potential energy and kinetic, which will decrease and increase throughout its orbit.

From the graph produced below, it can be observed that the peaks in kinetic energy and troughs in potential energy indicate the points at which Eris is closest to the sun (perihelion), and the peaks in potential energy and troughs in kinetic energy indicate the points at which Eris is farthest from the sun (aphelion).

In [235]:
```
# Plot kinetic energy, gravitational potential energy and total energy as a function of time
plt.plot(t, KE_per_m, label="KE per unit mass")
plt.plot(t, PE_per_m, label="PE per unit mass")
plt.plot(t, TE_per_m, label="TE per unit mass")
plt.title("Kinetic Energy, Gravitational Potential Energy and Total Energy vs Time")
plt.xlabel("Time (yrs)")
plt.ylabel("Energy Per Unit Mass ($AU^2 yr^{-2}$)")
plt.grid()
plt.legend()
plt.show()
```



Conservation of Energy

Is energy conserved in this system, consisting of the planet Eris orbiting around the Sun?

- To determine whether or not the energy of this system is conserved, one must consider the sum of all the kinetic and potential energies.
  - $Total\ Energy = Kinetic\ Energy + Potential\ Energy$
- The total energy must always remain the same for energy to be conserved
- We can observe from the graph above, that the total energy remains the same throughout the orbit of Eris around the sun, at approximately $-0.29\ AU^2 yr^{-2}$

It can therefore be concluded that energy is conserved

Orbital Period

The orbital period of Eris, the amount of time taken for Eris to complete one orbit around the sun, can also be determined from our previous kinetic energy equation.

In [236]:
```
# Determine the orbital period of Eris
print("The orbital period of Eris is:", np.argmax(KE_per_m), "years.")
```

```
The orbital period of Eris is: 574 years.
```