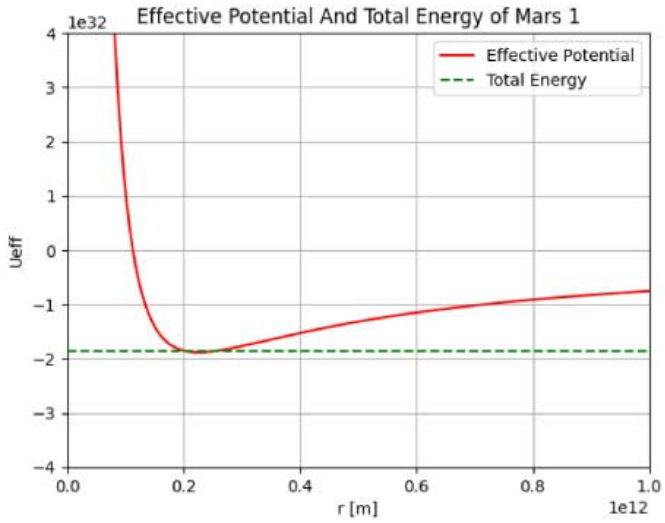
PHYS 35100 - FALL 2024

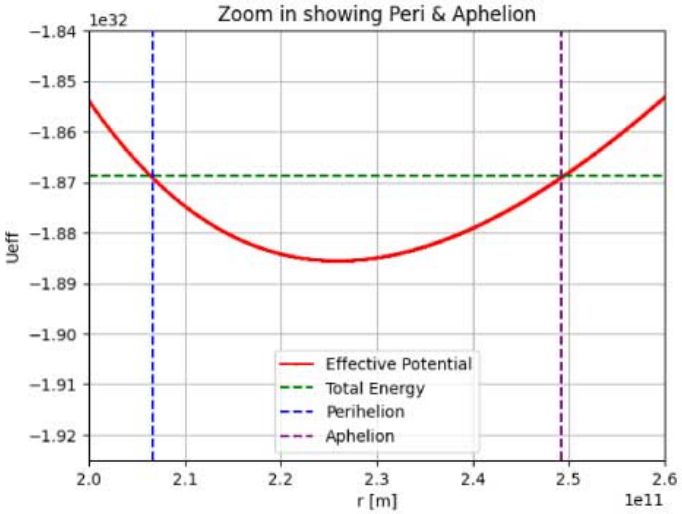
Homework Set 4 - Solutions

1. Halley's Comet Stats

a	Energy	-5.44×10^{21} J
b	Angular Momentum	1.04×10^{30} kg m^2 s^{-1}
c	Period (using a)	75.9 years

2. Effective Potential





3. Spring and 2 masses

In the center of mass frame, we can construct the lagrangian:

$$\mathcal{L} = \frac{1}{2}M\dot{\mathbf{R}}^2 + \frac{1}{2}\mu\dot{\mathbf{r}}^2 - \frac{1}{2}kr^2$$

where, $M = m_a + m_b$, $\mu = \frac{m_a m_b}{m_a + m_b}$, $\mathbf{R}$ is the center of mass location, and $\mathbf{r}$ is the relative position of the two masses.

There is no $\mathbf{R}$ in the lagrangian, so we can quickly say that $\ddot{\mathbf{R}} = 0$

For the $\mathbf{r}$ components of the Euler-Lagrange equation, we get:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{r}} = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{r}}}\right)$$

which quickly turns into:

$$\mu\ddot{\mathbf{r}} = -k\mathbf{r}$$

which we identify as a simple harmonic oscillator with frequency:

$$\omega = \sqrt{k/\mu}$$

4. Ping... Ping... Ping...

First, figure out the distance to the center of the Earth at perigee, using the average radius of the Earth of 6371 km:

$$6371\,\text{km} + 270\,\text{km} = 6641\,\text{km}$$

The velocity at perigee was quoted as 8700 m/s. Thus we can calculate the angular momentum as

$$l = m_{sat} v_{peri} r_{peri}$$

Using the $r(\phi)$ equation for an orbit we found:

$$r = \frac{l^2}{G m_E m_{sat}^2} \left(\frac{1}{1 + \epsilon \cos\phi} \right)$$

we can say at r_{peri}:

$$r_{peri} = \frac{m_{sat}^2 v_{peri}^2 r_{peri}^2}{G m_E m_{sat}^2} \left(\frac{1}{1 + \epsilon} \right)$$

Solving this for ϵ, we get:

$$\epsilon = \frac{v_{peri}^2 * r_{peri}}{G m_E} - 1$$

The distance from the center at apogee is found by setting $\phi = \pi$ in the $r(\phi)$ equation:

$$r_{apo} = \frac{l^2}{G m_E m_{sat}^2} \left(\frac{1}{1 - \epsilon} \right)$$

Replacing the angular momentum (which is constant) yields:

$$r_{apo} = \frac{v_{peri}^2 r_{peri}^2}{G m_E} \left(\frac{1}{1 - \epsilon} \right)$$

a. ϵ = 0.261
b. r_{apo} = 4983 km above the surface
c. The period from Kepler's 3rd law is aprox. 8496.46 seconds, or 2.36 hours.

```
import matplotlib.pyplot as plt
import numpy as np
```

Problem 1: Halley's Comet

a) Find total energy

```
MSun = 1.989E30 #kg
MHalley = 2.2E14 #kg

G = 6.67408E-11 #m^3 kg^-1 s^-2
oneAUinMeters = 149597870691 #meters
a = 17.93003431157555 * oneAUinMeters #m
```

The easiest thing to do would be use the equation that relates Energy to the masses and semi-major axis of the orbital ellipse:

$$E = -\frac{GM\mu}{2a}$$

```
def calcEnergy(m1,m2,a):
  M = m1+m2 #kg
  mu = m1*m2/(m1+m2) #reduced mass
  energy = -G*M*mu/(2*a) #equation derived in the ellipse math pdf
  return energy
```

```
print("energy in Joules: ",calcEnergy(MSun,MHalley,a))
```

```
energy in Joules:  -5.443927636628031e+21
```

Or we could do it by finding the kinetic and potential energy at some time then computing the total based on these two values. We'll choose two times, just to compare the numbers. (Should they be the same?)

If we retrieve some values for Halleys comet on two special days, when it is at perihelion and aphelion, for example, but really any two days will be work, we get 4 vectors:

```
 ******************************************************************************
            JDUT ,            Calendar Date (UT ),                     X,
 ******************************************************************************
 $$SOE
 2446471.000000000, A.D. 1986-Feb-09 12:00:00.0000,  4.895818068130744E+07, -6.71547052707
 2460288.000000000, A.D. 2023-Dec-09 12:00:00.0000, -2.970250750791217E+09,  4.07485758202
 $$EOE
 ******************************************************************************
```

```
rPerihelionVector = np.array([4.895818068130744E+07,-6.715470527075997E+07,2.4831
rPerihelion = np.linalg.norm(rPerihelionVector)*1E3 #find the magnitude and conve

rAphelionVector = np.array([-2.970250750791217E+09,+4.074857582022480E+09,-1.4889
rAphelion = np.linalg.norm(rAphelionVector)*1E3

vPerihelionVector = np.array([-4.276208662598826E+01,-3.326983810318622E+01,-6.21
vPerihelion = np.linalg.norm(vPerihelionVector)*1E3

vAphelionVector = np.array([7.199968937077390E-01,5.448161762044478E-01,1.0697863
vAphelion = np.linalg.norm(vAphelionVector)*1E3


print(rPerihelion)
print(rAphelion)
print(vPerihelion)
print(vAphelion)
```

```
86736801353.2043
5257742433754.372
54534.81470052535
909.2109884196084
```

```
def CalcMechPlusPot(m1,m2,r1,v1):
  #m1 is sun
  #m2 is orbiter (smaller than sun)
  kinetic = 0.5*m2*(v1**2)
  potential = -G*m1*m2/r1
  totalEnergy = kinetic + potential
  return totalEnergy
```

```
print("energy in Joules: ",CalcMechPlusPot(MSun,MHalley,rPerihelion,vPerihelion))
```

```
energy in Joules:  -9.556762880088532e+21
```

```
print("energy in Joules: ",CalcMechPlusPot(MSun,MHalley,rAphelion,vAphelion))
```

```
energy in Joules:  -5.46362564607888e+21
```

Interesting, the energy is different at perihelion vs aphelion. What could cause this?

b) Find angular momentum, l.

Let's use the relation between eccentricity and angular momentum:

$$\epsilon \equiv \sqrt{1 + \frac{2El^2}{G^2 M^2 m^3}}$$

Solving for l, we have:

$$l = \sqrt{\frac{G^2 M^2 \mu^3}{2E}(\epsilon^2 - 1)}$$

Double-click (or enter) to edit

```
def calcAngularMomentum(m1,m2,a,eccentricity):
  energy = calcEnergy(m1,m2,a)
  M = m1+m2 #kg
  mu = m1*m2/(m1+m2) #reduced mass
  angularMomentum = np.sqrt((G**2)*M**2*(mu**3)/(2*energy)*(eccentricity**2-1))
  return angularMomentum
```

```
print("angular momentum: ",calcAngularMomentum(MSun,MHalley,a,0.9679221169240834)
```

```
angular momentum:  1.0430277092543943e+30
```

c) Kepler's 3rd Law

Kepler's 3rd law as we derived it indicates:

$$T = \frac{2\pi}{\sqrt{GM}} a^{3/2}$$

Double-click (or enter) to edit

```
def calcPeriod(m1,m2,a):
  M = m1+m2 #kg
  period = 2*np.pi/np.sqrt(G*M)*np.power(a,1.5)
  return period
```

```
print("period in seconds: ",calcPeriod(MSun,MHalley,a))
```

```
period in seconds:  2395667668.45799
```

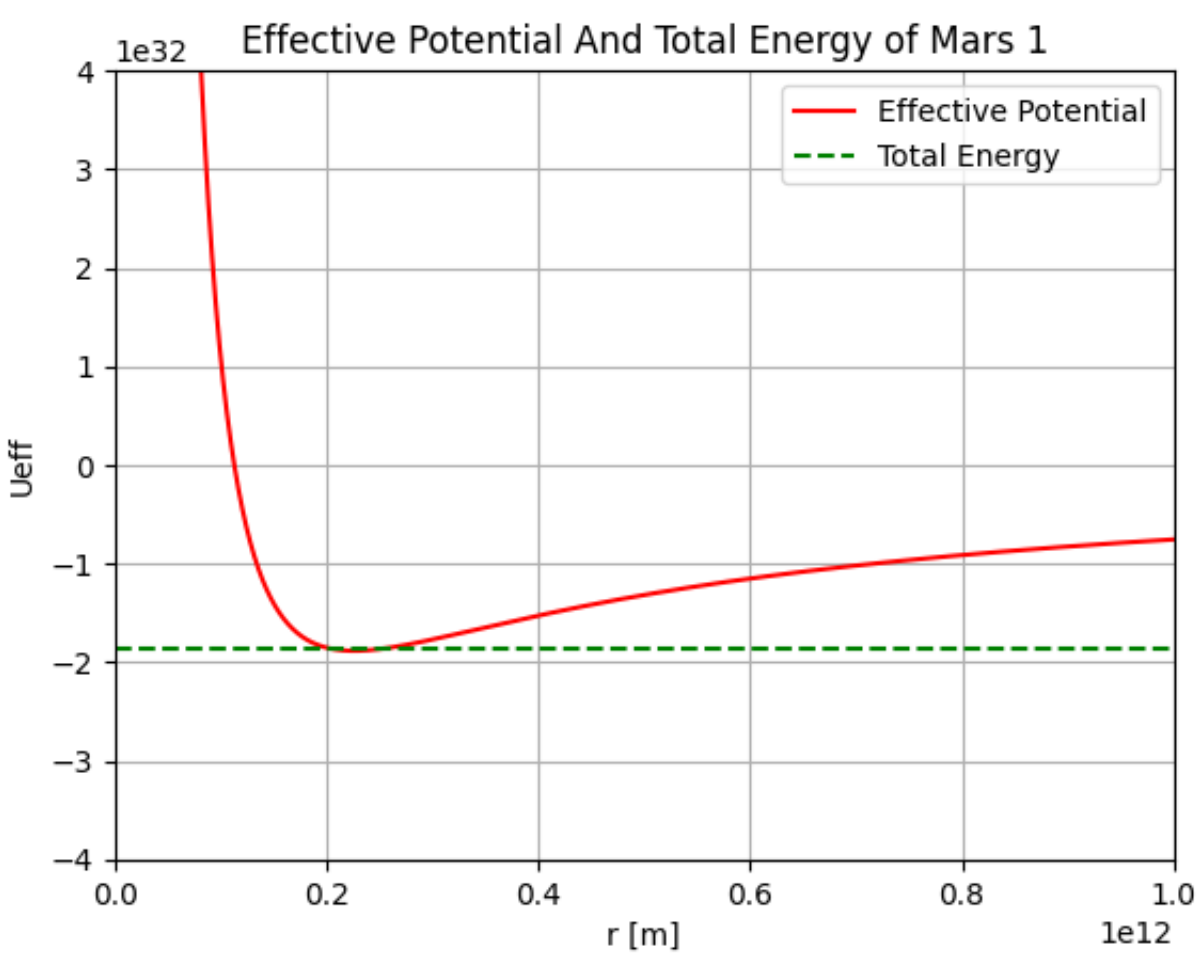
```
print("period in years: ",calcPeriod(MSun,MHalley,a)/(365.242374*24*60*60))
```

```
period in years:  75.91571248410818
```

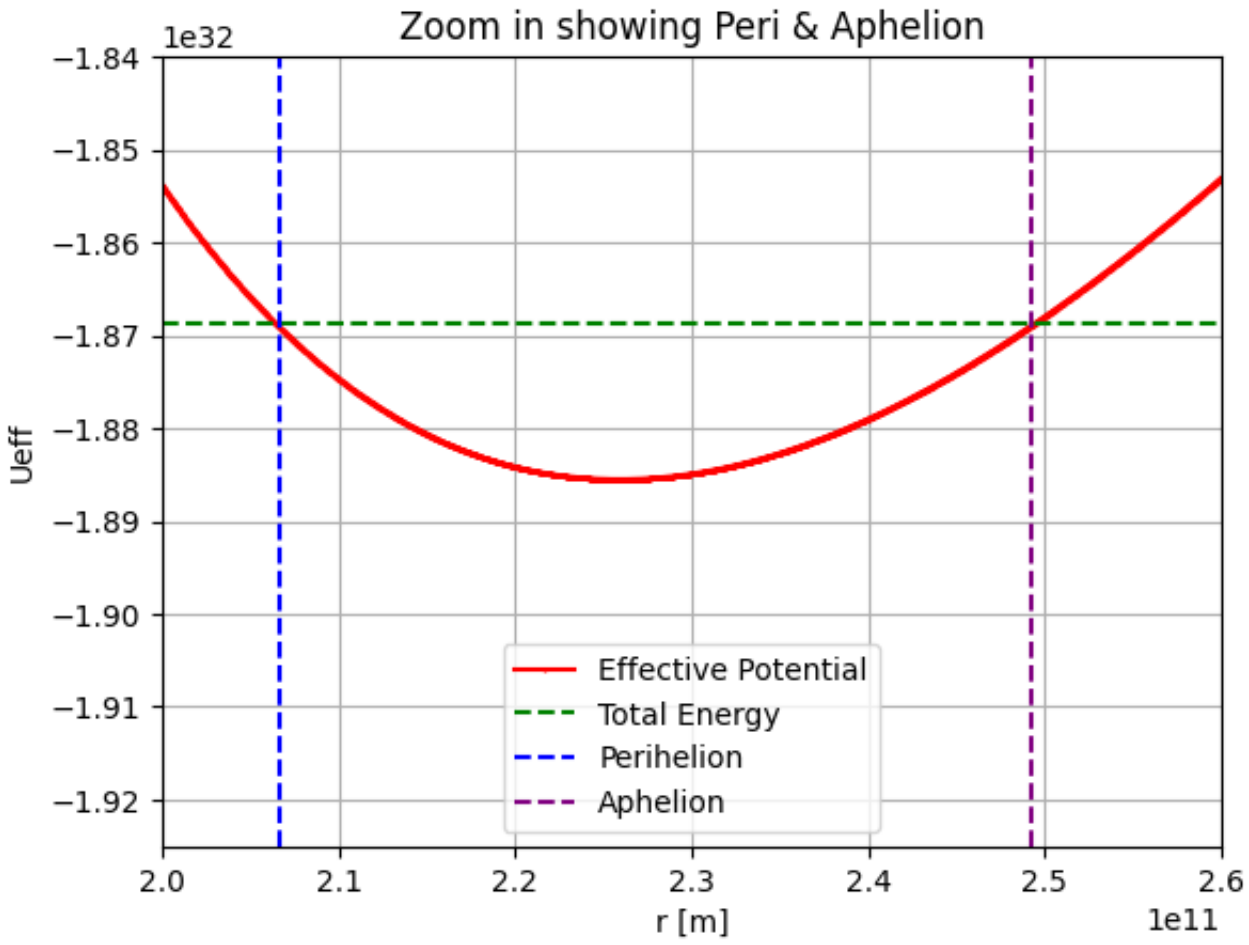
This period of 75.9 years agrees that data found in the JPL small bodies database.

```
import matplotlib.pyplot as plt
import numpy as np



G = 6.67430E-11
oneAU = 149597870700
mSun = 1.989E30
mMars = 6.4171E23
rM = oneAU*1.52368055
vM = 2*np.pi*rM/(686.980*24*60*60)
perihelionMars = 206650000000
aphelionMars = 249261000000
vMMax = 26.50E3 #m/s
lM = perihelionMars*mMars*vMMax
muMars = mSun*mMars/(mSun+mMars)
a_Mars = 2.27939366000E11


r = np.linspace(1E9,2E12,100000)
Ueff_1d = ((lM**2)/(2*muMars*np.power(r,2)))-G*mSun*mMars/np.power(r,1)
totalE_Mars = -(G*mSun*muMars)/(2*a_Mars)
```

```
fig, ax = plt.subplots()

ax.plot(r, Ueff_1d,'-',label="Effective Potential",c="red")
ax.axhline(y=totalE_Mars, c='green',linestyle="--",label="Total Energy")
ax.set_xlabel('r [m]')
ax.set_ylabel('Ueff')
ax.set_title('Effective Potential And Total Energy of Mars')
ax.set_ylim(-4E32,4E32)
ax.set_xlim(0,1E12)
ax.legend()
ax.grid()
#plt.savefig("effPot.svg")
plt.show()
```



```
fig, ax = plt.subplots()

ax.plot(r, Ueff_1d,'-',label="Effective Potential",c="red", marker='.', markersiz
ax.axhline(y=totalE_Mars,c='green',linestyle="--",label="Total Energy")
ax.axvline(x=206650000000,c='blue',linestyle="--",label="Perihelion")
ax.axvline(x=249261000000,c='purple',linestyle="--",label="Aphelion")

ax.set_xlabel('r [m]')
ax.set_ylabel('Ueff')
ax.set_title('Zoom in showing Peri & Aphelion')
ax.set_xlim(0.2E12,0.26E12)
ax.grid()
ax.legend()
ax.set_ylim(-1.925E32,-1.84E32)
plt.savefig("zoomin.svg")
plt.show()
```



Start coding or generate with AI.

```
import matplotlib.pyplot as plt
import numpy as np


rperi = (6371+270)*1000
vperi = 8700
MEarth = 5.972e+24
G = 6.67e-11
```

```


epsilon = ((vperi**2*rperi)/(G*MEarth))-1
print(epsilon)
```

```
0.261904103206267
```

```
rapo = ((vperi**2 * rperi**2)/(G*MEarth))*(1/(1-epsilon))
print(rapo)
```

```
11353951.682696814
```

```
(rapo/1000)-6371
```

```
4982.951682696814
```

```
period = (2*np.pi/np.sqrt(G*MEarth))*((rperi+rapo)/2)**(3/2)
print(period)
print(period/(60*60))
```

```
8496.461341085409
2.3601281503015024
```