Mars orbit simulator using Python 3.6

Department of Physics, Imperial College London

Tuesday, 7^{th} of February 2017

Abstract

The Python programming language has been used to simulate the trajectories, energy changes and angle deviations of a 260 kg satellite moving around Mars. This was done by programming a numerical model to solve differential equations for a range of initial conditions. Simulations of the various initial conditions agreed with analytical predictions. However, when plotting the total energy of the system the limitation of this method was shown as there was fluctuations in the total energy.

Introduction

Analysing orbit trajectories is important for space experiments where satellites are modelled to follow elongated orbits so that they can aid radio experiments [1]. It is used to simulate asteroids and help predict whether they will collide with Earth and if so, the time period we have until collision [2]. In this report, the orbit of a satellite of mass 260 kg is computed numerically by using Newton's laws and simplifying the associated differential equations. Newton's law of Gravitation and Newton's second law of motion describe the dynamics of a mass moving in a gravitational field. The aim is to simulate scenarios which reveal the properties of a satellite's motion around Mars. Energy properties of orbits and different trajectories can be used to check the simulations. Properties of the motion include trajectories where the satellite is captured and not captured, the angle deviation, and the systems energy. As some parts of this problem can be solved analytically, analysing the programs output by comparing it to expected outputs will help aid understanding in the capabilities of using computer simulations.

Theory

Newton's law of Gravitation states that the force between two bodies is inversely proportional to the distance between the masses and proportional to the product of the masses. The force on each mass is given by,

$$F = G\left(\frac{M_1 M_2}{r^2}\right)\hat{r}, \tag{1}$$

where G is the gravitational constant, M_1 and M_2 are the masses of the bodies, r is the separation of the bodies, and $\hat{r}$ is a unit vector in the radial direction. This force acts in the radial direction towards the centre of Mars.

Using Newton's second law and equation 1 results in an equation for acceleration, a,

$$a = G\left(\frac{M_1 M_2}{r^3}\right)\cdot\begin{pmatrix} x \\ y \end{pmatrix}, \tag{2}$$

where x is the x position and y is the y position of the satellite relative to the centre of Mars. This can be done because this problem is being done in two dimensions.

The kinetic energy of the satellite can be shown to be,

$$E_k = \frac{1}{2}M_2(v_x^2 + v_y^2), \tag{3}$$

and the potential energy of the satellite can be shown to be,

$$E_p = -\frac{GM_1M_2}{\sqrt{v_x^2 + v_y^2}}, \tag{4}$$

where M_1 is the mass of Mars and M_2 is the mass of the satellite, v_x is the x component and v_y is the y component of the satellite's velocity.

Method

There were two motions for Mars, one was Mars remaining stationary and the other was Mars moving with a constant velocity of 24 kms^{-1}. A Cartesian co-ordinate system was set-up for both instances, when Mars was treated as being stationary, Mars' centre was the origin, and when it was moving, the (0,0) point was treated as the origin. The force was projected along a unit vector $\hat{r}$ which is in the direction of the centre of Mars, so that the x and y components for velocity, v, and acceleration, a, could be separated (see equation 2).

The numerical model for solving these equations was based on the fact that the integral of acceleration is velocity, and that the integral of velocity is displacement. Python's Scipy module, 'integrate', was used to integrate x and y components of velocity and acceleration. A function was used which had an array where each column contained values for x, v_x, y, v_y. A time parameter, t, was involved and for this a range of time values were set-up. Arrays for x, v_x, y, v_y, and initial conditions were produced. The initial conditions were fed into the function by the 'odeint()' function. The 'odeint()' function takes 3 variables: the function that includes the values to be returned, initial conditions and the time range. It integrates the returned values of the function which is inserted as an argument for 'odeint()'. Then the arrays of displacements and velocities were used to make plots for different properties. This numerical model was tested by using initial conditions that would reproduce Mars' moon Phobos' orbit. Phobos' orbit is known to be nearly circular [1] and that is what the numerical model produced.

These properties were investigated by launching the satellite at different positions and velocities. Plots were then produced which showed the different aspects. For stationary Mars, plots were made for the satellite's trajectories when it was captured and not captured, the distance of closest approach to Mars, collision with Mars, and kinetic, potential and total energy. The escape velocity was calculated analytically using $v = \sqrt{\frac{2GM_{satellite}}{r}}$ and was 1816 ms^{-1} from position ($7R$,$3R$). Plots around this velocity range were made. For the distance of closest approach, a trial and error approach was taken where plots were made until a minimum distance was obtained. This minimum distance was defined as the distance from Mars' surface which the satellite could complete an orbit.

Energy plots were made using the array value and equations for kinetic energy and potential energy (see equations 3 and 4). These plots were made for captured and not captured, and elongated and circular orbits.

Angular deviation was calculated for trajectories which the satellite was not captured. This was done using the vector dot product and re-arranging to give: $\theta = \arccos\left(\frac{(v_0)(v_\infty)}{|v_0||v_\infty|}\right)$, where the subscript denotes the velocity at that time. The initial velocity was dotted with the velocity at a time which the satellite was very distant from Mars. Then the angle deviation was plotted against initial velocity.

Mars was made to move along the x-axis with constant velocity of 24 kms^{-1}. This was done by using a for loop to iterate through each time value and change the x-position by $v_{Mars}t$. Two arrays were created for Mars' x and y positions. The motion of the satellite behind and ahead of Mars was investigated. To produce a trajectory ahead of Mars v_x of the satellite would need to be greater than v_{Mars} and the y component needed to be low. The opposite was done to produce a trajectory behind Mars. The program will run following the logical steps below:

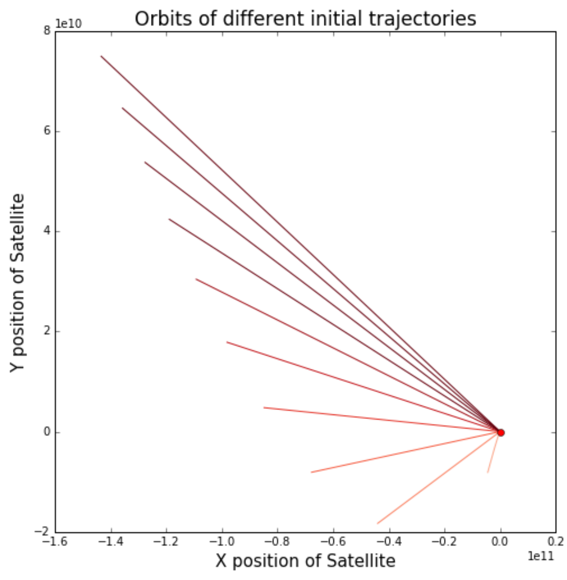
[Define constants to be used] $\rightarrow$ [Define function that integrates x,y,v$_x$,v$_y$] $\rightarrow$ [Check whether Mars is chosen to be moving or stationary] $\rightarrow$ [Make plots of position, energy and angle deviation according to this check].

Results and Discussion

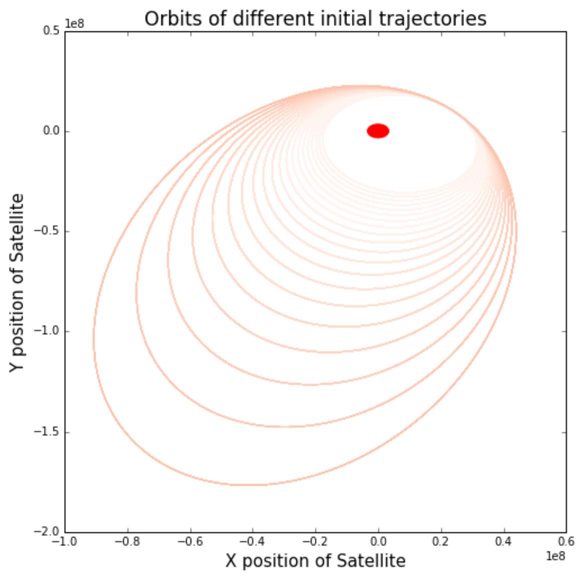
The satellite escaped with initial velocities above 1816 ms^{-1} as shown in figure 1.(a). The initial x and y velocities ranged from 1284 to 2200 ms^{-1} where the lighter coloured lines represent slower velocities. Figure 1.(a) shows the trajectories tending towards the initial launch direction as the initial velocity increases. This pattern is expected because with a velocity much faster than the escape velocity, the time spent in the gravitational field is reduced. So, the satellite is not effected as much. Initial velocities below 1816 ms^{-1} were captured by Mars and so completed orbits as shown in figure 1.(b).This is expected because as the satellite's velocity increases so does it's angular momentum. As the angular momentum increases the orbit trajectory becomes larger as the artificial outward force produced by the angular momentum increases as well. For both of the above cases the algorithm shows the pattern clearly, however one limitation is that the smaller velocity lines are shorter than larger velocity lines. This does not show that the smaller line does not complete an orbit.

The angle deviation of the satellite for a range of initial velocities, which were above the escape velocity, was also plotted. Figure 2.(c) shows that as the initial velocity increases the angle deviation tends towards zero. For velocities closer to the escape velocity, the angle deviation increases dramatically. This is because the force on the satellite is inversely proportional to r^2 (see equation 1), and so the slower initial velocities mean that the satellite is closer to Mars which causes a steep increase in angle deviation. These two results are what we expect to see because the satellites trajectory is less effected for further distances from Mars. This is for the same reason that the escape trajectories tend towards the initial trajectory.

The distance of closest approach calculated by the numerical model was 794 m. This result was obtained by running the program with different ranges of initial velocities. Velocities from the range 471.8 to 600.0 ms^{-1} produced the above minimum distance. Limitations of the numerical model are that it did not model Mars' surface, so no mountains or regions of high land were considered. Therefore, the distance of closest approach in this model was for a smooth spherical Mars, and would not reflect reality where Mars s not a smooth sphere. For a collision with Mars, the condition were shown to be a range of initial velocities up until the minimum distance of approach was reached.

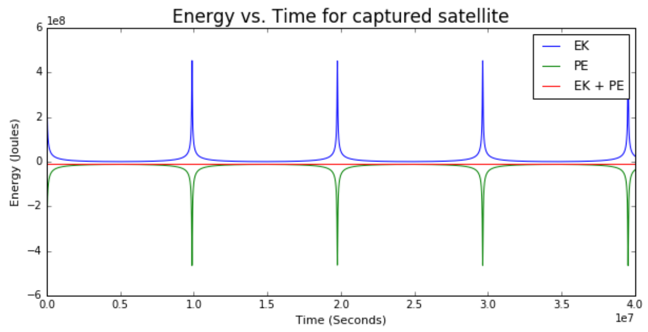


(a) Escape trajectories

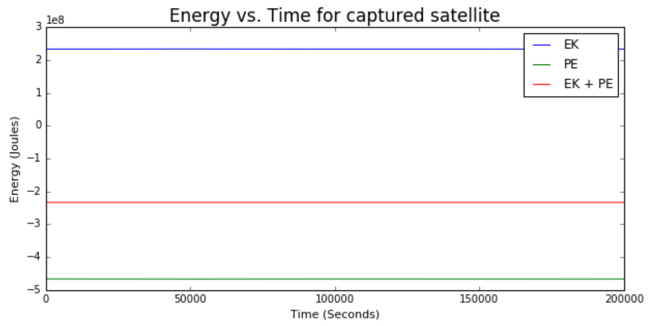


(b) Captured trajectories

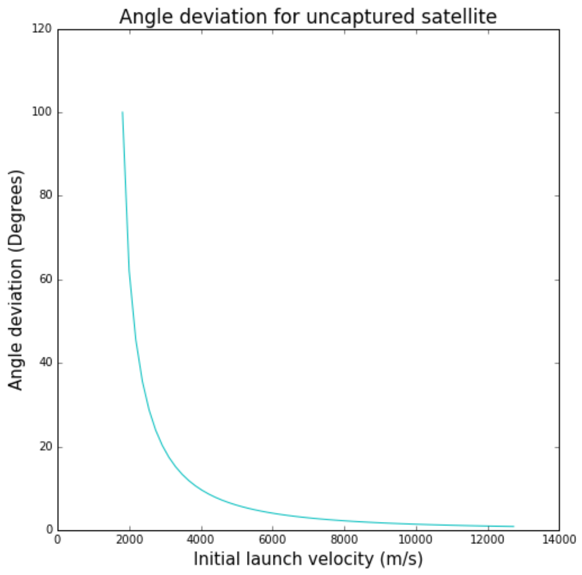
Figure 1: (a) shows the escape trajectories for initial x and y velocities in the range 1284 to 2200 ms^{-1}. The lighter coloured velocities are slower. This shows that as the initial velocities increase, the trajectory tends towards the initial launch direction. (b) shows complete orbits for velocities in the range 900 to 1200 ms^{-1}. The larger the initial velocity the more elongated the orbit. The smaller orbits reach a point where the orbit is nearly circular.



(a) Elongated orbit



(b) Circular Orbit



(c) Angle deviation

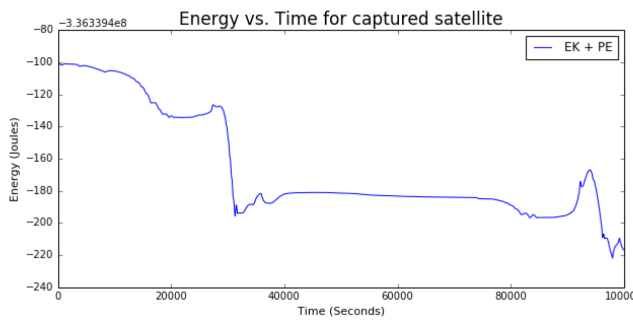
Figure 2: (a) shows E_k and E_p exchanging. There are sharp peaks in both energies and long durations of much lower energies. Total energy is negative, so it is captured. (b) shows constant E_k and E_p. Total energy is shown to be negative and appears to be a straight line. (c) shows the angle deviation tends towards zero for large values of initial velocity, and it is much greater for velocities nearer to the escape velocity.

As the initial velocity was increased from 480 to 1283 ms^{-1}, a wide range of orbits were produced. Elongated orbits were shown to have much wider time gaps between kinetic and potential energy transfers (see figure 2.(a)). There are also peaks in the kinetic energy which shows that for a short time the satellite speeds up drastically. This agrees with what should happen in an elongated orbit where the satellite slows down for most of the journey and speeds up when it is near Mars. The total energy is close to zero, however it is still negative. This shows that the overall energy is gravitational potential energy and so the satellite has been captured. When comparing this result to the expected outcome it agrees well. We expect the total energy to be negative for captured satellites, and we expect the energy to be transferred from kinetic to potential energy. For more circular orbits the kinetic and potential energy is shown to be constant from the numerical model (see figure 2.(b)). This is expected because in circular motion the velocity of the satellite remains constant and so the kinetic energy is constant, and the distance from Mars remains constant and so the potential energy is constant.

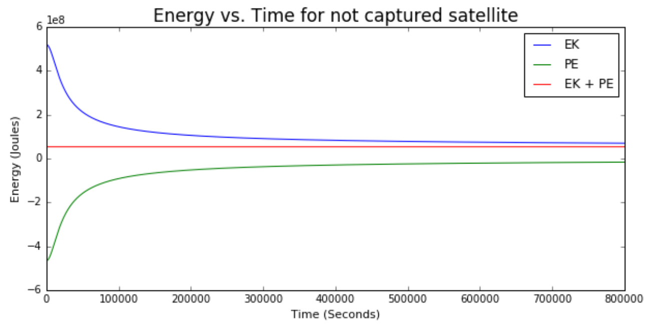
When running the program with initial velocities greater than the escape velocity the total energy is shown to be positive. This is expected because now the satellite has enough initial kinetic energy to escape Mars' attraction. As time increases, the potential energy tends towards zero and the kinetic energy tends towards the total energy of the satellite. This agrees with what should happen as when the satellite is very distant the potential approaches zero (see equation 2), therefore the only energy left is kinetic and so the kinetic tends towards the total energy.

Figure 3.(a) shows that the total energy of the satellite is not constant because it fluctuates. The range of total energy fluctuation for the time range 0 to 100000 seconds is around 120 J. This does not agree with the expected constant total energy. This fluctuation in energy is much greater than what was expected, however it is a small fluctuation relative to the actual kinetic and potential energies. From this, another limitation is shown to be in the numerical models ability to produce numbers which are accurate. The time division was decreased to check whether this would give a lower fluctuation but it showed a greater fluctuation.

In reality Mars is also moving. For this motion, the satellite is required to also increase its velocity in order to escape. This is because the escape velocity is relative to Mars being stationary but now it is moving at 24.1 kms^{-1}.



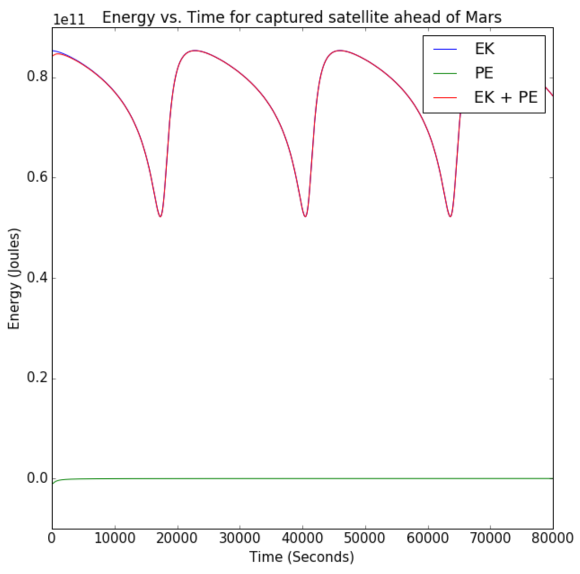
(a) Total energy



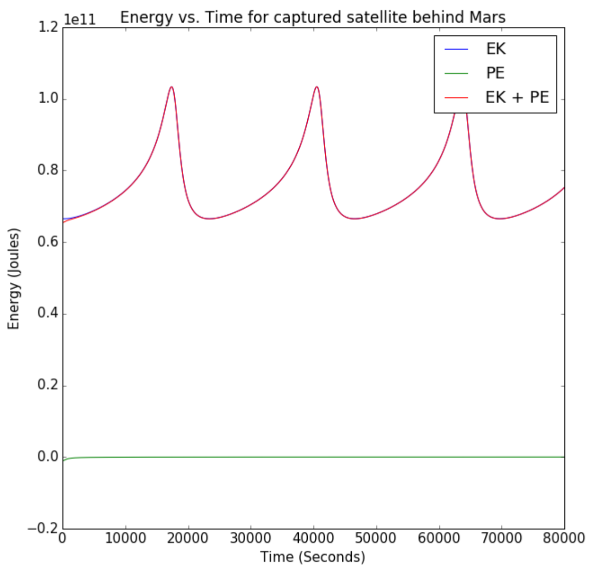
(b) Energy of escape trajectory

Figure 3: (a) shows that the total energy fluctuates and is not constant. (b) shows total energy is more kinetic for escape trajectories. E_k tends towards total energy, and E_p tends towards zero.

As the satellite moves ahead of Mars, the kinetic energy is expected to decrease because the satellite experiences a force which opposes it's motion. Therefore, the kinetic energy should increase for a satellite behind Mars as it experiences as force in the same direction as it's motion. These two outcomes are shown by figure 4, and so the numerical model is producing correctly shaped graphs. The potential energy should oscillate, however the numerical model shows a line tending towards zero for large values of time. When only potential energy is plotted the same pattern remains. These results can be trusted as far as the shape of the kinetic energies, however the potential energy for a satellite orbiting a moving Mars is not correct. Based on this result and the small fluctuation of the total energy these results are an accurate representation.



(a) Ahead of Mars



(b) Behind Mars

Figure 4: (a) shows that E_k decreases first and then increases, with the decrease being more steady and the increase being much steeper. (b) shows that E_k increases steadily and then decreases steeply. E_p tends towards zero in both cases. In both, the kinetic energy oscillates with a constant time period of about 16000 seconds.

Conclusion

Simulations are good for captured and not captured trajectories. They show that the satellite escapes when it's velocity is larger than the escape velocity, and that it completes orbits when launch below the escape velocity (1816 ms^{-1}). The angle deviation for not captured trajectories shows that large initial velocities lead to a smaller angle deviations, this agrees with what should happen. Energy is shown to follow the correct pattern for most simulations, however the total energy is not shown to be conserved as it fluctuates. Energy graphs for circular, elongated and escape trajectories shows the expected pattern. Circular orbits have constant kinetic and potential energy, elongated orbits have oscillating energies, and escape trajectories have energies that tend toward the kinetic energy which decreases with time. When simulating Mars as a moving body the kinetic energy graphs for passing behind and ahead agree with what should happen, however the potential energy tends towards zero. Collisions and distances of closest approach were well simulated, when it collides it remains stationary and the distance of closest approach is 794 m. However, the model did not take into account that Mars' surface is not smooth. To improve this numerical model, the surface of Mars should be included in the model, and the numerical model should run with smaller time divisions so that the accuracy of the solution gets closer to the true value. This showed that computers are a powerful tool for simulating physical scenarios and it highlighted the fact that the quality of the software and hardware limit how well the simulation agrees with reality.

Bibliography

[1]George Davis (Emergent Space Technologies, LLC), Michael Moreau, Russell Carpenter, Frank Bauer (NASA Goddard Space Flight Center),"GPS-BASED NAVIGATION AND ORBIT DETERMINATION FOR THE AMSAT AO-40 SATELLITE", American Institute of Aeronautics and Astronautics

[2]Martin Jutzi et Al., "Modelling asteroid collisions and impact processes", arXiv:1502.01844v1 [astro-ph.EP], 6 Feb 2015

[3]NASA, http://solarsystem.nasa.gov/planets/phobos/facts.

Appendix

```
import scipy as sp
import numpy as np
import pylab as pl
import matplotlib
import matplotlib.pyplot as plt
import scipy.integrate as spi
G= 6.67e-11 #define G - constant of gravity
M = 6.4e23  #define M - mass of Mars
m = 260     #define m - mass of satellite
R = 3.4e6   #define mars' radius
mv = 24100  #mars' velocity
marsmoving = input(str("Do you want to simulate mars moving? Enter y for yes, enter n for no:")) #allows user to
def f(x,t): #called with the variables to be integrated
    if marsmoving == "yes":
        xx=x[0] - mv*t #updating satellites' position relative to mars
    elif marsmoving == "no":
        xx=x[0] # updates position of satellite relative to origin
    vx=x[1] # x component of velocity
    yy=x[2] # y component of distance
    vy=x[3] # y component of velocity
    r = (xx**2 +yy**2)**0.5   #radial distance from centre of Mars
    if r < R: #dont plot - zero values for [xx,vx,yy,vy]
        if marsmoving == "yes": #keeps satellite moving with mars' velocity when it crashes
            return[mv,0,0,0]
        elif marsmoving == "no": #keeps satellite stationary when crashes on mars
            return[0,0,0,0]
    ax=-xx*(G*M)/(r**3)  #acceleration in x direction
    ay=-yy*(G*M)/(r**3) #acceleration in y direction
    return [vx,ax,vy,ay] #actually returns [xx,vx,yy,vy] - integrates returned values
timelimit = input("Please enter the time limit you would like to run the simulation:") #user option for time scal
initialv = input("Enter initial velocity:") #user option for initial velocity
finalv = input("Enter final velocity:") #user option for final velocity
t=sp.linspace(0.,timelimit,4000) # solving every t per second
Ang = [] # empty list for angle deviation
vlist = [] # empty list for velocites to be used for angle deviation
dist =[]  # for minimum dist
for i in sp.linspace(initialv,finalv,10):
    if marsmoving == "yes":
        IC=[0,mv+i,4*R,i] # set initial conditions [x,vx,y,vy] for satellite when mars is moving
    elif marsmoving =="no":
        IC=[7*R,-i,3*R,i] # set initial conditions [x,vx,y,vy] for satellite when mars is stationary
    soln=spi.odeint(f,IC,t) # integrates vx,vy,ax,ay
    x=soln[:,0] #takes xx solution and places it in first column of array
    vx=soln[:,1] #takes vx solution and places it in second column of array
    y=soln[:,2] #takes yy solution and places it in third column of array
    vy=soln[:,3] #takes vy solution and places it in fourth column of array
    c = pl.get_cmap('Reds')((i - initialv)/finalv) # colouring different velocities, second brackets give rgb col
    dotproduct = (vx[0]*vx[-1]+vy[0]*vy[-1]) / (((vx[0]**2+vy[0]**2)**0.5)*((vx[-1]**2+vy[-1]**2)**0.5)) #dot pro
    O = 180*np.arccos(dotproduct)/np.pi # angle deviation for specific initial conditions
    Ang.append(O) #array of angle deviations
    v = (vx[0]**2+vy[0]**2)**0.5   # initial velocities
    vlist.append(v) #array of initial velocities to be used for angle deviation plot
    rdist =(x**2+y**2)**0.5 # for use in closest aproach
    if rdist[i] > R:
        dist.append(rdist)
    if marsmoving == "no":
        plt.plot(x,y,color=c) # Plot the x and y components of satellite for multpile trajctories in the initial
        if finalv <650: #this was the finalv that gave close aproach
            zz = ", Closest distance:" + str(int(np.amin(dist))-int(R)) #calculates the minimum distance from the
yM = [] #empty array for y position of mars
xM = [] #empty array for x position of mars
for i in range(len(t)): #for loop that plots mars's position at same time as the satelittes motion
    yM.append(0) #keep adding zero's to mars' y position to keep mars on the x-axis so moves straight line
    xM.append(mv*t[i]) # appends the updated version of mars' position as time moves forward
plt.figure(1)
if marsmoving == "yes":
    plt.plot(x,y)
    plt.plot(xM,yM, 'ro') # plots mars moving
elif marsmoving == "no":
    mars=plt.Circle((0,0),R,color='r',label="Mars") # creates circle radius of mars
    plt.gcf().gca().add_artist(mars) # plots the cirlce
plt.xlabel("X position of Satellite",size=15)
plt.ylabel("Y position of Satellite",size=15)
if marsmoving == "no":
    if finalv < 600:                       #
        plt.xlim(-1e7,3e7)                 #
        plt.ylim(-1e7,3e7)                 #
    elif finalv >= 600 and finalv < 1200: ##    setting the time scale for plots based on final velocity as this
        plt.xlim(-5e7,8e7)                 #
        plt.ylim(-5e7,8e7)                 #
    elif finalv > 1200:                    #
        plt.xlim(-2e8,2e8)                 #
        plt.ylim(-2e8,2e8)                 #
plt.title('Satellite moving around Mars' + zz, size=17)
fig = plt.gcf() #takes figure plotted and lets edit
fig.set_size_inches((8, 8))
fig.savefig("Satellite's trajcetory")
plt.show()
plt.figure(2) ##################Angle Deviation------------------------------------------
plt.plot(vlist,Ang, color = 'c')
plt.xlabel("Initial launch velocity (m/s)",size=15)
plt.ylabel("Angle deviation (Degrees)",size=15)
plt.title("Angle deviation for satellite",size=17)
fig = pl.gcf() #takes figure plotted and lets edit
plt.savefig("Angle deviation")
fig.set_size_inches((8, 8)) #sets size of figure
plt.figure(3)##################Energy Plots --------------------------------------------
EK = 0.5*m*(vx**2+vy**2) # kinetic energy formula
r = (x**2+y**2)**0.5 #radial distance
PE = -G*M*m/r  # potential energy
plt.plot(t,EK, label = "EK") #plot kinetic
plt.plot(t,PE, label = "PE") #plot potential
plt.plot(t,EK+PE, label ="EK + PE") # plot total enegry
plt.title("Energy vs. Time for captured satellite ahead of Mars",size=13)
matplotlib.rcParams.update({'font.size': 13})
plt.xlabel("Time (Seconds)",size=15)
plt.ylabel("Energy (Joules)",size=15)
plt.legend()
fig = plt.gcf() #takes figure plotted and lets edit
fig.set_size_inches((8, 8))
plt.savefig("Energy Plot")
plt.show()
```