<table>
<tr><td colspan="3">LODZ UNIVERSITY OF TECHNOLOGY
Faculty of Technical Physics, Information Technology and Applied Mathematics
Field of study: Modelling and Data Science
Course name: Numerical Methods</td></tr>
<tr><td>Academic year:
Semester:
Lab day and time:</td><td colspan="2">Task’s no:</td></tr>
<tr><td>Student ID number:</td><td>First name, last name
Marta Haik</td><td>Report grade:</td></tr>
<tr><td>Date of submission of the report:</td><td colspan="2"></td></tr>
</table>

1. Problem description - Kirkwood gaps

Knowing the parameters of orbits of Jupiter and Mars (assuming they are circular and lay in the same plane) simulate orbital movement of a few planetoids. Assume that the planetoids don't influence the movement of the planets. In particular show the orbits of the following planetoids (radii of orbits given in *AU*, orbital velocities to be calculated) [1]:

$\mathbf{r_1}$ = 3.000 AU,

$\mathbf{r_2}$ = 3.276 AU,

$\mathbf{r_3}$ = 3.700 AU.

2. Equations

Knowns: **r1, r2, r3**

Unknowns $\mathbf{v_{orb}}$**, forces**

To do: **plot**

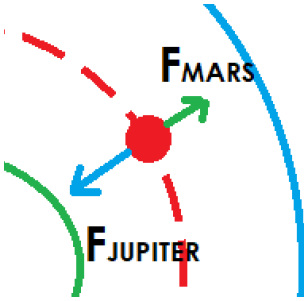


Figure 1 Simplified representation of forces acting on planetoid

The planetoid is influenced by Mars and Jupiter simultaneously, so we need to consider them both. The plot should display the two planets and the planetoid, first simulating one round around the Sun, then increasing the amount of rounds to see what will be the behavior of the planetoid.

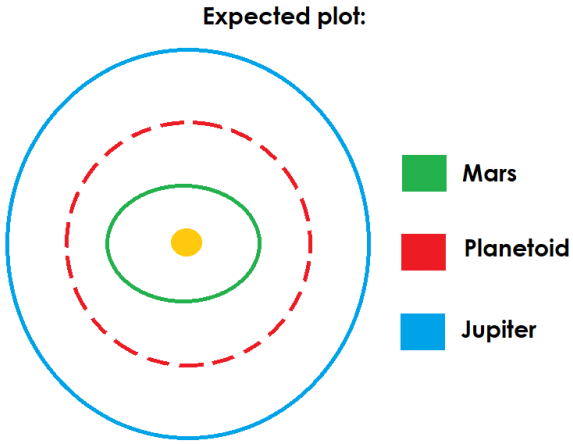


Figure 2 The plot that I will try to achieve

The program:

```
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from scipy.linalg import norm

plt.close("all")

# parameters

# distances of orbits from the sun in [AU]
sun_radius = 0.0  # lays in the center
jupiter_radius = 5.2
mars_radius = 1.52
r1 = 3.0  # planetoid 1
r2 = 3.276  # planetoid 2
r3 = 3.7  # planetoid 3

R = r1  # variable choosing the planetoid to test

GMs = 1.327e20  # heliocentric gravitational constant [m^3/s^2]
G = 6.674e-11  # gravitational constant [m^3/kg*s^2]
m_sun = 1.989e30  # mass of sun [kg]

# info Jupiter
ecc_jup = 0.0489 #0
semimajor_jup = 5.2044
a_jup = semimajor_jup
perihelion_jup = a_jup*(1-ecc_jup)
m_jup = 1.8982e27 # mass [kg]

vp_jup = math.sqrt(GMs) * math.sqrt((1+ecc_jup) / (a_jup*(1-ecc_jup)) *
(1+(m_jup/m_sun)))

# info Mars
ecc_mars = 0.0934 #0
semimajor_mars = 1.5237 # [10^6 km]
a_mars = semimajor_mars
perihelion_mars = a_mars * (1 - ecc_mars)
m_mars = 6.4171e23 # mass [kg]
```

```
vp_mars = math.sqrt(GMs) * math.sqrt((1+ecc_mars) / (a_mars*(1-ecc_mars)) *
(1+(m_mars/m_sun)))

# info planetoid
ecc_pl = 0  # circular
semimajor_pl = 2.5 # [10^6 km]
a_pl = semimajor_pl
perihelion_pl = a_pl*(1-ecc_pl)
m_pl = 1e10  # mass is arbitarary since << Msun

# initial parameters
# for planetoid
vp_pl = math.sqrt(GMs) * math.sqrt((1+ecc_pl) / (a_pl*(1-ecc_pl)) *
(1+(m_pl/m_sun)))
vx = 0
vy = vp_pl
x = -perihelion_pl
y = 0
# for Jupiter
vx2 = 0
vy2 = vp_jup
x2 = -perihelion_jup
y2 = 0
# for Mars
vx3 = 0
vy3 = vp_mars
x3 = -perihelion_mars
y3 = 0

starting = np.array([
    x2,y2,  #position of jup
    x3,y3,  #position of mars
    x,y,    #position of planetoid

    vx2,vy2,  #velocity of jup
    vx3,vy3,  #velocity of mars
    vx,vy     #velocity of planetoid
])

def system(t, Y, G, m1, m2, m3):

    # position vectors of all 3 planets
    x1 = Y[:2]
    x2 = Y[2:4]
    x3 = Y[4:6]
```

```
    # calculcating the square of distance vecotrs
    d12 = (x1 - x2)/np.power(np.linalg.norm(x1 - x2),3)
    d13 = (x1 - x3)/np.power(np.linalg.norm(x1 - x3),3)
    d23 = (x2 - x3)/np.power(np.linalg.norm(x2 - x3),3)

    d1s = x1/np.power(np.linalg.norm(x1),3)
    d2s = x2/np.power(np.linalg.norm(x2),3)
    d3s = x3/np.power(np.linalg.norm(x3),3)

    m_sun = 1.989e30  # mass of sun [kg]

    # net forces
    f1 = G*(-m2*d12 - m3*d13-m_sun*d1s)
    f2 = G*(m1*d12 - m3*d23-m_sun*d2s)
    f3 = G*(m1*d13 + m2*d23-m_sun*d3s)

    # return the right hand side of the equation
    return np.hstack((Y[6:],f1,f2,f3))


t = 2*np.math.pi/(-vy/x)
n_steps = 1000      # the number of steps
n_periods = 100     # number of full circles
ts = np.linspace(0,n_periods*t,n_steps)

y = solve_ivp(lambda t,Y : system(t, Y, G, m_jup, m_mars, m_pl), (0,t*n_periods),
starting, t_eval=ts, max_step=t/n_steps)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(y.y[0],y.y[1],'b')
ax.plot(y.y[2],y.y[3],'g')
ax.plot(y.y[4],y.y[5],'r')
ax.legend(['jupiter','mars','planetoid'])
ax.scatter(0,0,s=200,c='y')
ax.set_aspect("equal")
ax.set(title='Planetoid: r = {} AU after {} cycle(s)'.format(R, n_periods),
xlabel='Distance (AU)', ylabel='Distance (AU)')
plt.show()
```

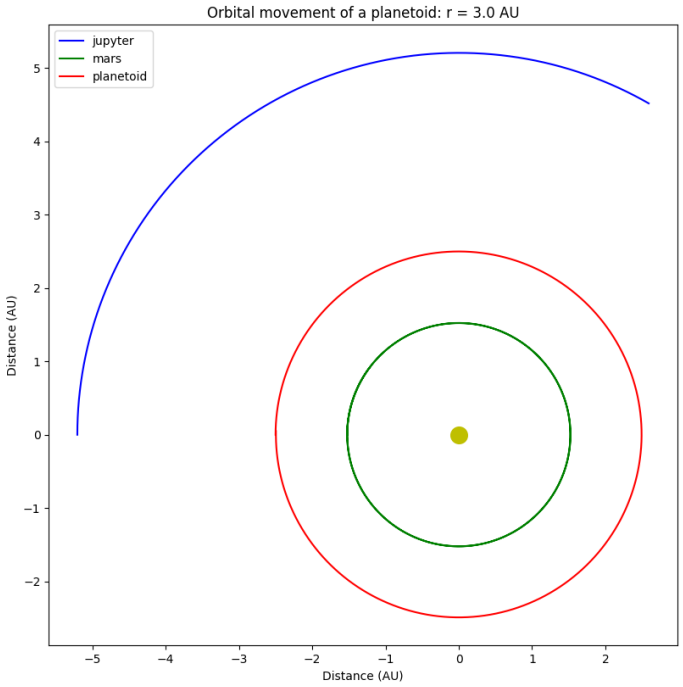
3. Typical parameters

The parameters for the program have been taken from official scientific tables [2], [3], [4] when it comes to Mars, Jupiter and the Sun (and some constants). But, we can experiment with planetoid data as it is not fixed.

4. Results

The first plot is after one round around the Sun. The two below are after 100 cycles. The difference between them are orbit eccentricity: the left one is taken from tables and right one is equal to 0 (because they were supposed to be circular).

a) Planetoid 1 of $\mathbf{r_1}$ = 3.000 AU



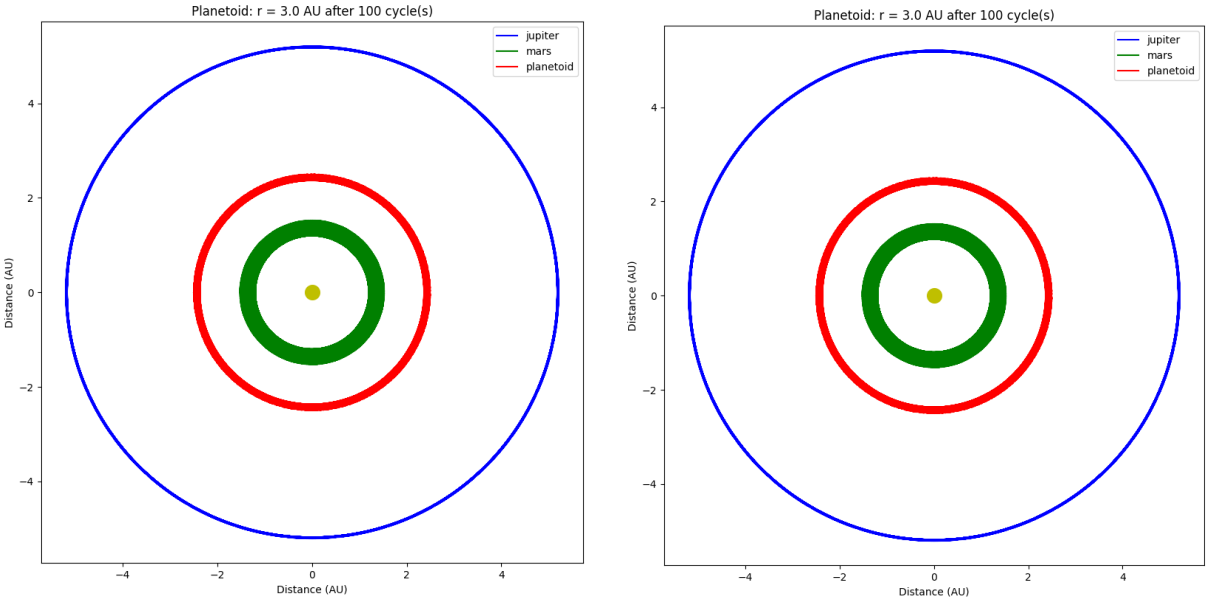
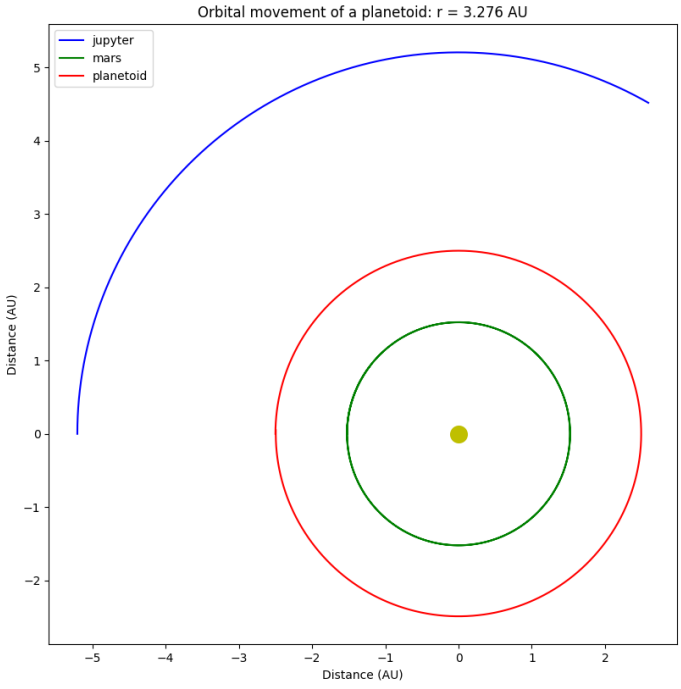


Figure 3 The first planetoid behaviour

b) Planetoid 2 of $\mathbf{r_2}$ = 3.276 AU



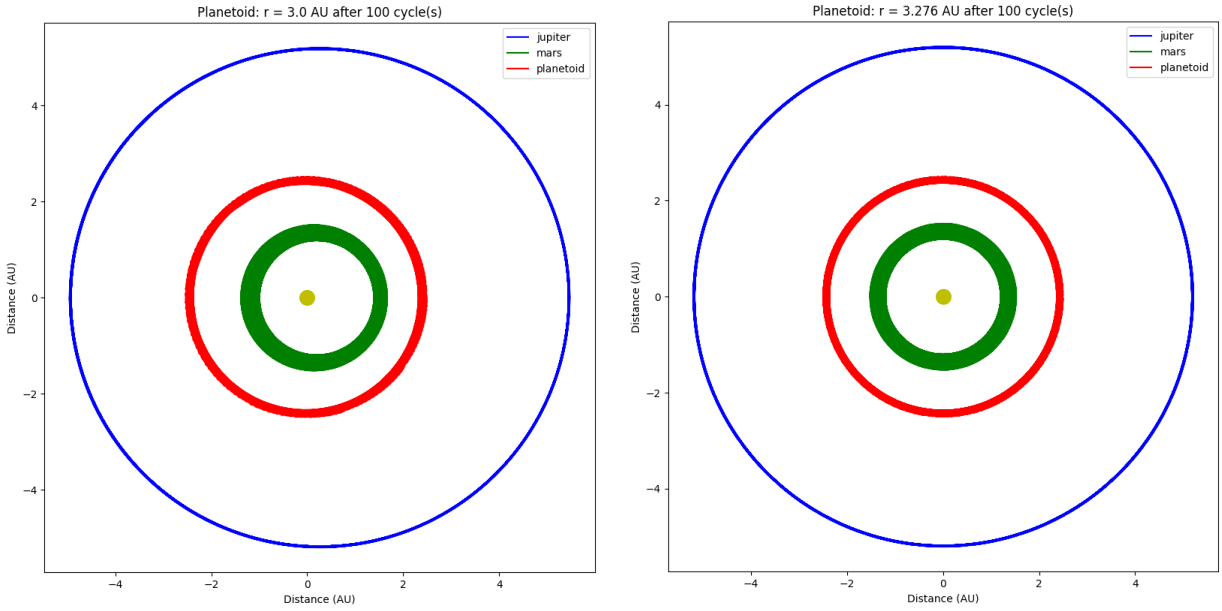
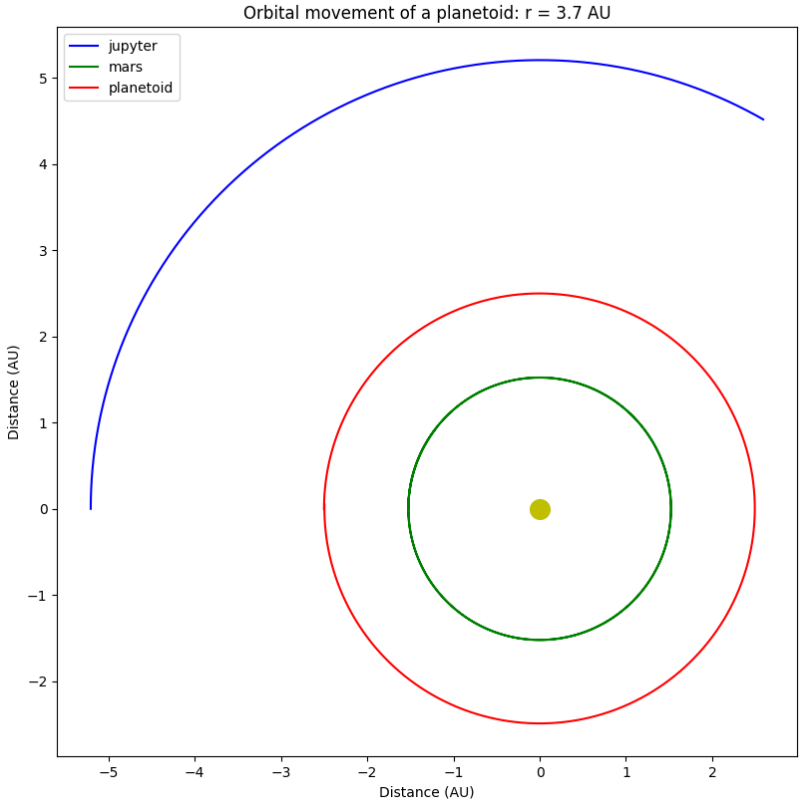


Figure 4 The second planetoid behaviour

c) Planetoid 3 of $\mathbf{r_3}$ = 3.700 AU



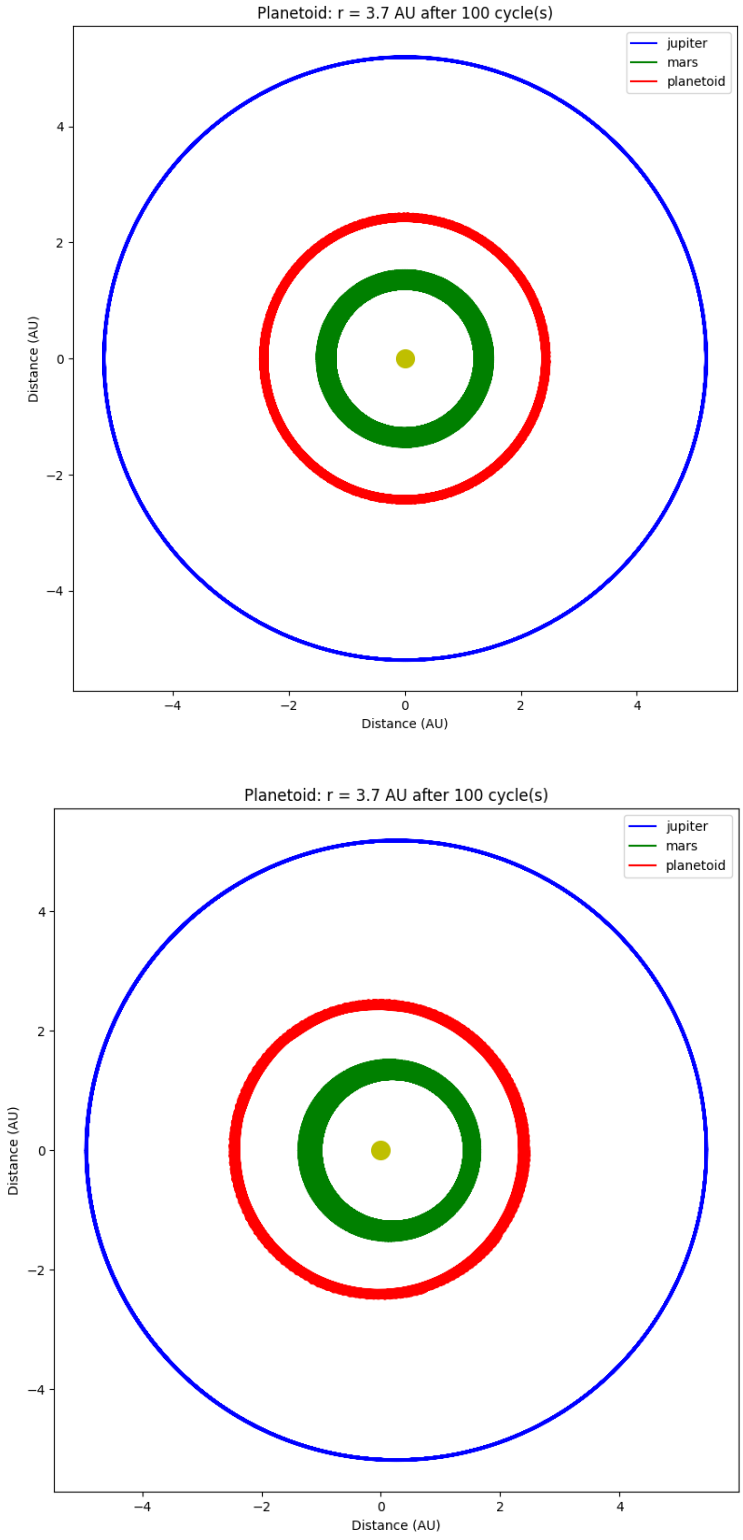


Figure 5 The third planetoid behaviour

A remark: the difference between the presence (or its lack) of orbital eccentricity increases with the distance from Sun.

I also tested higher amount of cycles and different mass of planetoid but it didn't have much impact for the graph.

5. Discussion of correctness

The program appears to be correct in terms of syntax and logic. However, there are no visible Kirkwood gaps. It was not implied in the task itself that they have to appear, but its title suggested that at least one of the distances will be corresponding to said gaps.

One of the most prominent Kirkwood gaps (which are at similar distance to the ones given: 3.000, 3.276 and 3.700 AU) are located at [5]:

- 2.958 AU (7:3 resonance),
- 3.279 AU (2:1 resonance),
- 3.972 AU (3:2 resonance)

These gaps occur at specific resonances with Jupiter, where the gravitational effects of Jupiter's orbit disrupt the orbits of asteroids, causing them to be cleared out of those regions [5]. The graphs should have had dispersion of points (imitating the dips in asteroids distribution).

Nonetheless, the program correctly uses the input data and produces results with the use of *solve_ivp* function, producing the graphs.

6. Sources

[1] pdf file "assignment equations" on the FTIMS WIKAMP platform
[2] NASA Jupiter parameters: Jupiter Fact Sheet (nasa.gov)
[3] NASA Mars parameters: Mars Fact Sheet (nasa.gov)
[4] NASA Sun parameters: Sun Fact Sheet (nasa.gov)
[5] Wikipedia page about Kirkwood gaps Kirkwood gap - Wikipedia