Osnovi mehanike - vežbe 2
08. mart 2022.

1. Primenom Ojlerove metode:

   a) odrediti do koje visine će se popeti telo koje je brzinom v_0 bačeno sa Zamlje vertikalno u vis, kao i vreme penjanja;

```
a=-9.81

v=100
h=0

t=0

dt=1e-3 #vremenski korak

while v>=0:
    h=h+v*dt
    v=v+a*dt
    t+=dt

print('h=',h,  '\n v=', v ,'\n t=', t)
```

   b) odrediti kojoj brzinom će ovo telo udariti o tlo;

```
a=-9.81

v=100
h=0

t=0

dt=1e-3 #vremenski korak

while h>=0:
    h=h+v*dt
    v=v+a*dt
    t+=dt

print('h=',h,  '\n v=', v ,'\n t=', t)
```

   c) odrediti koji je odnos vremena koje telo provede u penjanju i u padanju.

```
a=-9.81
v=100
h=0
t=0

dt=1e-3 #vremenski korak

t_penjanja=0
t_padanja=0

while h>=0:
    h=h+v*dt
    v=v+a*dt
    t+=dt
    if v>0:
        t_penjanja+=dt
    else:
        t_padanja+=dt

print('odnos=', t_penjanja/t_padanja)
```

2. Prikazati zavisnost doleta od ugla i brzine za uglove iz skupa $\alpha_0[^o] \in \{0, 1, 2, ..., 90\}$ i brzine iz skupa $v_0[\text{km/s}] \in \{50, 51, 52, ..., 100\}$.

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

from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

v=np.arange(50,101)
alfa=np.deg2rad(np.arange(0,91))

domet=np.zeros([len(v), len(alfa)]) # alokacija

ax=0
ay=-9.81

dt=1e-1 #vremenski korak

for i in range(0, len(v)):
    for j in range(0,len(alfa)):

        #pocetni uslovi
        x=0; y=0
        vx=v[i]*np.cos(alfa[j]); vy=v[i]*np.sin(alfa[j])

        while y>=0:

            x+=vx*dt
            y+=vy*dt

            vx+=ax*dt
            vy+=ay*dt

        domet[i][j]=x

# konture
plt.figure()
plt.contourf(np.rad2deg(alfa), v, domet,20, cmap=cm.jet)
plt.xlabel('ugao')
plt.ylabel('brzina [m\s]')
plt.colorbar()
#plt.show()

# 3D plot
fig = plt.figure()
ax = fig.add_subplot(1,1,1, projection='3d')

alfa, v = np.meshgrid(alfa, v)

ax.plot_surface(v,np.rad2deg(alfa),domet,cmap=cm.jet,linewidth=0, antialiased=False)
ax.set_xlabel('brzina [m/s]')
ax.set_ylabel('ugao')
ax.set_zlabel('domet [m]')
plt.show()
```

3. Za maksimalni domet kosog hica odrediti putanje korišćenjem Ojlerove i Ojler-Kromerove metode. Koliko maksimalni domet odstupa od analitičkog rešenja i kako ovo odstupanje zavisi od vremenskog koraka?

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

```
v=1000 # pocetna brzina
alfa=np.deg2rad(45) # pocetni ugao koji daje maksimalni domet

DT=np.arange(1e-3,1, 1e-3) #niz vremenskih koraka

D_o=np.zeros(len(DT)) # Ojlerova metoda
D_ok=np.zeros(len(DT)) # Ojler-kromerova metoda

#analiticko resenje
domet=v**2/9.81

ax=0; ay=-9.81 #komponente ubrzanja
for dt in DT:

    # OJLEROVA METODA
    x = 0
    y = 0  # pocetni polozaj
    vx = v * np.cos(alfa)
    vy = v * np.sin(alfa)  # pocetna brzina
    while y>=0:
        x+=vx*dt
        y+=vy*dt
        vx+=ax*dt
        vy+=ay*dt
    D_o[np.argwhere(DT==dt)]=x

    # OJLER-KROMEROVA METODA
    x = 0
    y = 0  # pocetni polozaj
    vx = v * np.cos(alfa)
    vy = v * np.sin(alfa)  # pocetna brzina
    while y>=0:
        vx+=ax*dt
        vy+=ay*dt
        x+=vx*dt
        y+=vy*dt

    D_ok[np.argwhere(DT==dt)]=x

plt.figure()
plt.plot(DT,D_o)
plt.plot(DT,D_ok,'r')
plt.grid()
#plt.show()

plt.figure()
plt.plot(DT,np.abs(D_o-domet)) # apsolutno odstupanje Ojlerove metode
plt.plot(DT,np.abs(D_ok-domet), 'r') # apsolutno odstupanje Ojler-kromerove metode
#plt.show()

plt.figure()
plt.plot(DT,np.abs(D_o-domet)) # apsolutno odstupanje Ojlerove metode
plt.plot(DT,np.abs(D_ok-domet), 'r') # apsolutno odstupanje Ojler-kromerove metode

plt.show()
```

4. Primenom Ojler-Kromerove metode:

   a) odrediti do koje visine će se popeti telo koje je brzinom v_0 bačeno sa Zemlje vertikalno u vis, ako na njega deluje sila obrnuto srazmerna kvadratu rastojanja od centra Zemlje:

$$F = \gamma \frac{mM}{r^2},$$

gde su $\gamma = 6,67 \times 10^{-11} \frac{\mathrm{m}^3}{\mathrm{kg\ s}^2}$ gravitaciona konstanta, $M = 5,972 \times 10^{24}$kg masa Zemlje, $R = 6,371 \times 10^6$m srednji poluprečnik Zemlje, r rastojanje od centra Zemlje, a m masa tela;

```
# konstante
GM=3.983324e14
R=6.371e6

v=3000 #pocetna brzina

dt=1e-2

r=R #pocetno rastojanje od centra Zemlje
while v>0:
    v-=GM/r**2*dt
    r+=v*dt

print(v)
print((r-R)/1000) #visina u km
```

b) odrediti i prikazati na koji način ova visina zavisi od početne brzine (uporediti sa analitičkim rešenjem $h = \frac{v_0^2}{2g}$ dobijenim za $g = 9,81 = \text{const.}$);

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

# konstante
GM=3.983324e14
R=6.371e6

V=np.arange(100,11300,100) #pocetna brzina
H=np.zeros(len(V))

Hh=V**2/(2*9.81) # analiticko resenje

dt=1 #vremenski korak

for i, v in enumerate(V):

    r=R #pocetno rastojanje od centra Zemlje
    while v>0:
        v-=GM/r**2*dt
        r+=v*dt
    H[i]=(r-R)/1000

plt.figure()
plt.plot(V,H, 'r')
plt.plot(V,Hh/1000, 'b')
plt.xlabel('brzina [m/s]')
plt.ylabel('visina penjanja [km]')
plt.show()
```

c) odrediti i prikazati na koji način vreme penjanja zavisi od brzine (uporediti sa analitičkim rešenjem $t = \frac{v_0}{g}$ dobijenim za $g = 9,81 = \text{const.}$);

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

# konstante
GM=3.983324e14
```

```
R=6.371e6

V=np.arange(100,5000,100) #pocetna brzina
T=np.zeros(len(V))

Th=V/9.81 # analiticko resenje

dt=1e-2 #vremenski korak

for i, v in enumerate(V):

    r=R #pocetno rastojanje od centra Zemlje
    t=0
    while v>0:
        v-=GM/r**2*dt
        r+=v*dt
        t += dt
    T[i]=t

plt.figure()
plt.plot(V,T, 'r')
plt.plot(V,Th, 'b')
plt.xlabel('brzina [m/s]')
plt.ylabel('vreme penjanja [s]')
plt.show()
```

d) odrediti i prikazati na koji način vreme izvršavanja koda zavisi od brzine;

```
import numpy as np
from matplotlib import pyplot as plt
import time

# konstante
GM=3.983324e14
R=6.371e6

V=np.arange(100,5000,100) #pocetna brzina
vreme=np.zeros(len(V))


dt=1e-3 #vremenski korak

for i, v in enumerate(V):

    r=R #pocetno rastojanje od centra Zemlje
    t=0
    pocetak = time.time()
    while v>0:
        v-=GM/r**2*dt
        r+=v*dt
        t += dt

    vreme[i]=time.time()-pocetak

plt.figure()
plt.plot(V,vreme, 'r')
plt.xlabel('brzina [m/s]')
plt.ylabel('vreme izvrsavanja [s]')
plt.show()
```

e) odrediti do koje početne brzine je razlika u visini dobijena Ojler-Kromerovom metodom manja od 1% u odnosu na analitički dobijenu; **-DOMAĆI**

f) odrediti kojom brzinom će ovo telo udariti u tlo. **-DOMAĆI**