Osnovi mehanike - vežbe 5
29. mart 2022.

1. Telo A slobodno pada sa visine H. Sa rastojanja d od vertikale duž koje se kreće telo A, u trenutku kada ono počinje da pada ispaljen je hitac sa Zemlje. Primenom Ojler-Kromerove metode odrediti:

   a) putanju tela uz analitička rešenja za početnu brzinu i ugao hica i prikazati – **prethodne vežbe**;

   b) putanje tela za različite brzine i različite uglove i prikazati iste;

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

g=-9.81
Hsp=100
d=15

#################
#analiticka resenja
#alfa=np.arctan(Hsp/d)
###v0 moze biti u opsegu od ove vrednosti pa do previse (uslov)
#v0=np.sqrt(-g*Hsp/(2*(np.sin(alfa))**2))
###############

V0=np.arange(10,35,5)
Alfa=np.deg2rad(np.arange(60,90,5))

dt=1e-3

#NESTED LISTS - za upisivanje rezultata:
#(alternativno, moze i np.array (biće u d),e),f)))
# U svaku praznu listu se upisuju koodinate za svaku iteraciju dt u while petlji.
# Praznih lista ima len(Alfa)*len(V0), gde kolone odgovaraju jednoj
# vrednosti ugla, a redovi jednoj vrednosti brzina

X = [[[] for column in range(len(Alfa))]for row in range(len(V0))]
Y = [[[] for column in range(len(Alfa))]for row in range(len(V0))]
Xsp = [[[] for column in range(len(Alfa))]for row in range(len(V0))]
Ysp = [[[] for column in range(len(Alfa))]for row in range(len(V0))]

for i, v0 in enumerate(V0):
    for j, alfa in enumerate(Alfa):

        x=0; y=0
        vx=v0*np.cos(alfa)
        vy=v0*np.sin(alfa)

        xsp=d #zauvek
        ysp=Hsp
        vsp=0

        while x<d and ysp>=0 and y>=0:
            #hitac

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

            #slobodan pad
```

```
            vsp+=g*dt
            ysp+=vsp*dt

            X[i][j].append(x)
            Y[i][j].append(y)
            Xsp[i][j].append(xsp)
            Ysp[i][j].append(ysp)


#PLOTOVANJE - prikaz vise grafika ja jednom plotu -> subplot!
fig = plt.figure(figsize=(14, 12), layout='constrained')
for i in range(len(V0)):
    for j in range(len(Alfa)):
        ax=fig.add_subplot(len(V0),len(Alfa), len(Alfa)*i+j+1)
        ax.plot(X[i][j], Y[i][j], '.r')
        ax.plot(Xsp[i][j], Ysp[i][j], '.')
        a = np.rad2deg(Alfa[j])
        v = V0[i]
        ax.set_title(f'ugao = {"%0.1f" % a},brzina = {"%0.1f" % v}')
plt.show()
```

c) kojom brzinom i pod kojim uglom treba ispaliti hitac da bi pogodio telo A;

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

g=-9.81
Hsp=100
d=15

dt=1e-3

#################
#analiticka resenja
#alfa=np.arctan(Hsp/d)
###v0 moze biti u opsegu od ove vrednosti pa do previse (uslov)
#v0=np.sqrt(-g*Hsp/(2*(np.sin(alfa))**2))
################

V0=np.arange(10,40,2)
Alfa=np.deg2rad(np.arange(60,90,2))

#NESTED LISTS
R = [[[] for column in range(len(Alfa))]for row in range(len(V0))]

for i, v0 in enumerate(V0):
    for j, alfa in enumerate(Alfa):

        x=0; y=0
        vx=v0*np.cos(alfa)
        vy=v0*np.sin(alfa)

        xsp=d #zauvek
        ysp=Hsp
        vsp=0

        while x<d and ysp>=0 and y>=0:
            #hitac

            vy+=g*dt
            x+=vx*dt
```

```
            y+=vy*dt

            #slobodan pad

            vsp+=g*dt
            ysp+=vsp*dt

        #najmanje rastojanje, najblizi su sudaru tad
        R[i][j]=np.sqrt((xsp-x)**2+(ysp-y)**2)

#print(R)
#print(type(R))
R=np.array([np.array(i) for i in R])
#print(type(R))
result = np.where(R == np. amin(R))
print('optimalna brzina =',V0[result[0][0]], 'm/s','\n',
      'optimalni ugao =',np.rad2deg(Alfa[result[1][0]]))
```

d) na koji način optimalni ugao zavisi od početne brzine;

e) na koji način brzina pogotka (kosog hica) zavisi od početne brzine;

f) na koji način visina pogotka zavisi od početne brzine?

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

# konstante
g=9.81
h1=500
d=1000

# pocetni uslovi
alfa_a=np.rad2deg(np.arctan(h1/d)) #analiticko resenje
alfa=np.deg2rad(np.arange(10,80,2))
v=np.arange(200,500,10)

#NUMPY ARRAY (umesto lista)
alfa_optimalno=np.zeros(len(v))
brzina_pogotka=np.zeros(len(v))
visina_pogotka=np.zeros(len(v))

dt=1e-3 # vremenski korak

# pocetni uslovi
for i in range(len(v)):
    # donje np.array će se upisivati u gornje definisane
    # na taj način imamo zavisnost i od pocetne brzine i od pocetnog ugla

    # razlika visina dva tela kada hitac dostigne rastojanje d
    razlika_visina=np.zeros(len(alfa))
    # visina na kojoj je telo (slobodni pad) kada dostigne trazeno rastojanje
    h_pogotka=np.zeros(len(alfa))
    # brzina na kojoj je telo (kosi hitac) kada dostigne trazeno rastojanje
    v_pogotka=np.zeros(len(alfa))

    for j in range(len(alfa)):
        # pocetni uslovi
        # hitac
        x=0; y=0;
        vx=v[i]*np.cos(alfa[j]); vy=v[i]*np.sin(alfa[j])
```

```
            # slobodan pad
            yt=h1
            vt=0
            while x<=d and y>=0 and yt>=0:

                # hitac
                vy-=g*dt
                x+=vx*dt
                y+=vy*dt
                # slobodan pad
                vt-=g*dt
                yt+=vt*dt

            razlika_visina[j]=np.abs(y-yt)
            h_pogotka[j]=yt
            v_pogotka[j]=np.sqrt(vx**2+vy**2)

        alfa_optimalno[i]=alfa[np.argmin(razlika_visina)]
        visina_pogotka[i]=h_pogotka[np.argmin(razlika_visina)]
        brzina_pogotka[i]=v_pogotka[np.argmin(razlika_visina)]

    # minimalna brzina (ona pri kojoj se pogodak
    # desava kada je abs(visina_pogotka) minimalna)
    v_min=v[np.argmin(np.abs(visina_pogotka))]

    #zavisnost optimalnog ugla od brzine
    plt.figure()
    plt.plot(v,np.rad2deg(alfa_optimalno))
    plt.xlabel('brzina')
    plt.ylabel('pocetni ugao')

    #zavisnost brzine pogotka hica u telo od pocetne brzine
    plt.figure()
    plt.plot(v,brzina_pogotka)
    plt.xlabel('brzina')
    plt.ylabel('brzina pogotka')

    # zavisnost visine pogotka od pocetne brzine
    plt.figure()
    plt.plot(v,visina_pogotka)
    plt.plot(v_min, visina_pogotka[np.argmin(np.abs(visina_pogotka))],'or')
    plt.xlabel('brzina')
    plt.ylabel('visina pogotka')

    plt.show()
```

3. 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, $M = 5,972 \times 10^{24}\mathrm{kg}$ masa Zemlje, $R = 6,371 \times 10^{6}\mathrm{m}$ srednji poluprečnik Zemlje, r rastojanje od centra Zemlje, a m masa tela;

   a) korišćenjem konstantnog vremenskog koraka Δt, kao i korišćenjem koraka koji je funkcija ubrzanja tela dat kao $\frac{C}{a}$, gde je a ubrzanje tela usled Zemljine gravitacije, a C konstanta takva da u početnom trenutku vremeski korak bude Δt, a zatim odrediti relativnu grešku svake visine u odnosu na analitičku za datu zavisnost ubrzanja, kao i vremena izvršavanja kodova i njihov odnos;

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

# konstante
GM=3.983324e14
R=6.371e6
g=9.81
V=10000 #pocetna brzina

#analiticko resenje
h=V**2*R**2/(2*GM-R*V**2)

# KONSTANTAN VREMENSKI KORAK
pocetak = time.time()
v=V #pocetna brzina
dt=1e-2
r=R #pocetno rastojanje od centra Zemlje
while v>0:
    v-=GM/r**2*dt
    r+=v*dt
h1=r-R
vreme1=time.time()-pocetak

print(h1/1000) #visina u km
print(vreme1)
print('-----------------')

# VREMENSKI KORAK KOJI ZAVISI OD UBRZANJA
pocetak = time.time()
v=V #pocetna brzina
c=GM*dt/R**2
r=R #pocetno rastojanje od centra Zemlje
while v>0:
    dt=c*r**2/GM
    v-=GM/r**2*dt
    r+=v*dt
h2=r-R
vreme2=time.time()-pocetak

print(h2/1000) #visina u km
print(vreme2)
print('-----------------')

#relativne greske
print('relativne greske')
print('konstantan korak: ',abs(h1-h)/h*100)
print('promenljiv korak: ',abs(h2-h)/h*100)
print('odnos vremena izvrsavanja koda: ', vreme1/vreme2)
```

b) prikazati odnos vremena izvršavanja kodova (oba slučaja vremenskog koraka) od početne brzine v_0;

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

# konstante
GM=3.983324e14
R=6.371e6
g=9.81
V=np.arange(0,11200,200) #pocetna brzina
```

```
odnos=[]

for i in range(len(V)):
    print(V[i])
    # KONSTANTAN VREMENSKI KORAK
    pocetak = time.time()
    v=V[i] #pocetna brzina
    dt=1e-2
    r=R #pocetno rastojanje od centra Zemlje
    while v>0:
        v-=GM/r**2*dt
        r+=v*dt
    h1=r-R
    vreme1=time.time()-pocetak

    # VREMENSKI KORAK KOJI ZAVISI OD UBRZANJA
    pocetak = time.time()
    v=V[i] #pocetna brzina
    c=GM*dt/R**2
    r=R #pocetno rastojanje od centra Zemlje
    while v>0:
        dt=c*r**2/GM
        v-=GM/r**2*dt
        r+=v*dt
    h2=r-R
    vreme2=time.time()-pocetak

    odnos.append(vreme1/vreme2)

plt.figure()
plt.plot(V,odnos)
plt.xlabel('pocetna brzina')
plt.ylabel('odnos vremena izvrsavanja koda')
plt.grid()
plt.show()
```

c) prikazati razlike visina (u odnosu na analitičko rešenje) u zavisnosti od početne brzine.

```
 import numpy as np
from matplotlib import pyplot as plt
import time
# konstante
GM=3.983324e14
R=6.371e6
g=9.81
V=np.arange(0,11200,200) #pocetna brzina
razlika1=[]
razlika2=[]


#analiticko resenje
h=V**2*R**2/(2*GM-R*V**2)


for i in range(len(V)):
    print(V[i])
    # KONSTANTAN VREMENSKI KORAK
    v=V[i] #pocetna brzina
    dt=1e-2
    r=R #pocetno rastojanje od centra Zemlje
    while v>0:
```

```
            v-=GM/r**2*dt
            r+=v*dt
        h1=r-R
        razlika1.append((h1-h[i])/h[i]*100)

        # VREMENSKI KORAK KOJI ZAVISI OD UBRZANJA
        v=V[i] #pocetna brzina
        c=GM*dt/R**2
        r=R #pocetno rastojanje od centra Zemlje
        while v>0:
            dt=c*r**2/GM
            v-=GM/r**2*dt
            r+=v*dt
        h2=r-R
        razlika2.append(abs(h2-h[i])/h[i]*100)


plt.figure()
plt.plot(V,razlika1)
plt.plot(V,razlika2, 'r')
plt.xlabel('pocetna brzina')
plt.ylabel('relativne greske [\%]')
plt.grid()
plt.show()
```