Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za fiziku

Primjena programiranja u fizici

Prolazak kometa kroz Sunčev sustav

Mateo Turić

Split, 23.6.2025.

Sažetak

U ovom seminarskom radu proučit će se utjecaj gravitacijske sile na komet koji prolazi kroz Sunčev sustav. Također će se simulirati i slučaj u kojem se komet neelastično sudari s jednim od planeta. Simulacija je izvedena u programskom jeziku Python i prikazana u obliku animacije koristeći modul Matplotlib.

1 Uvod

Tijela će u ovoj simulaciji međudjelovati jedino silom gravitacije, dok će se sve ostale sile kojima tijela mogu međudjelovati smatrati zanemarivima. Gravitacijska sila dana je sa

$$\vec{F}_{12} = G \cdot \frac{m_1 m_2}{r^2} \hat{r} \tag{1}$$

Gdje je $G = 6.67430 \cdot 10^{-11}\mathrm{Nm}^2/\mathrm{kg}^2$ gravitacijska konstanta, m_1 i m_2 su mase tijela, a $\vec{r} = (x_2 - x_1, y_2 - y_1)$ je vektor udaljenosti između tijela.
Primjenom II. Newtonovog zakona dobije se izraz za akceleraciju tijela koja se direktno računa u simulaciji:

$$\vec{a}_1 = \frac{\vec{F}_{12}}{m_1} = G \cdot \frac{m_2}{r^2} \hat{r} \tag{2}$$

Iz principa superpozicije slijedi da se ukupna sila na tijelo može računati kao vektorski zbroj svih sila, a to povlači da isto vrijedi i za akceleraciju. Ukupna akceleracija tijela tada je dana izrazom

$$\vec{a}_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \vec{a}_{ij} \tag{3}$$

Gdje je $\vec{a}_{ij}$ doprinos j-tog tijela na ukupnu akceleraciju, a n je broj tijela. Brzina i položaj tijela računaju se Eulerovom metodom.
Sudar dvaju tijela u simulaciji se računa kao potpuno neelastičan sudar, pa zakon očuvanja količine gibanja glasi

$$m_1 \vec{v}_1 + m_2 \vec{v}_2 = \vec{v}(m_1 + m_2) \tag{4}$$

iz čega slijedi da je brzina novog tijela nakon sudara

$$\vec{v} = \frac{m_1 \vec{v}_1 + m_2 \vec{v}_2}{m_1 + m_2} \tag{5}$$

2 Diskusija

Simulacija je napravljena tako da može podržati proizvoljan broj planeta koji se jednostavno mogu dodati kao objekti klase *Tijelo*. Svaki objekt inicijalizira se u klasi *Tijelo* sa sljedećim parametrima:

- Masa
- Početni položaj u obliku liste $[x, y]$, gdje su x i y koordinate tijela
- Početna brzina u obliku liste $[v_{0x}, v_{0y}]$, gdje su v_{0x} i v_{0y} komponente početne brzine
- Ime tijela koje služi isključivo za prikaz u legendi

- Debljina linije kojom će se putanja tijela prikazati u animaciji

Odmah po inicijalizaciji tijelo se dodaje u listu *objekti* koja sadrži sve stvorene objekte, što će biti potrebno da bi se računao doprinos svih ostalih tijela na akceleraciju objekta. U metodi *akceleracija* se prolaskom kroz listu *objekti* računa ukupna akceleracija za pojedino tijelo na način dan u izrazima (1), (2) i (3)

```
def akceleracija(self, G=6.67408e-11):
    # m*a=G mM/r**2
    # a_x = a*cosfi = a* x/d
    # a_y = a* y/d

    self.a=[0,0]
    for obj in Tijelo.objekti:
        if obj != self and obj.flag == 0: #Da se ne računa utjecaj objekta samog na sebe, bude 0 udaljenost
            d = [obj.r[-1][0]-self.r[-1][0], obj.r[-1][1]-self.r[-1][1]] # vektor udaljenosti od tijela do nekog objekta
            d_iznos= mt.sqrt(d[0]**2+d[1]**2)
            #print(obj.r)
            self.a=[self.a[0]+G*obj.m/(d_iznos**2)*(d[0]/d_iznos), self.a[1]+G*obj.m/(d_iznos**2)*(d[1]/d_iznos)] #pridonos akceleracije jednog objekta na tijelo
            #Unutar jedne kalkulacije akceleracije krece se od 0, pa se svaku iteraciju zbraja doprinos svakog objekta
    return 0
```

Slika 1: Metoda *akceleracija*

Sljedeća metoda je *korak* u kojoj se događa promjena akceleracije, brzine i položaja. Na početku se provodi detekcija sudara tako što se provjeri postoje li dva objekta čija je međusobna udaljenost manja od predefinirane vrijednosti *d_sudar*. Potreba za definicijom proizvoljne udaljenosti proizlazi iz toga što objekti nemaju definiran radijus, već se računaju kao materijalne točke. Bez nje bi, zbog numeričke pogreške, bilo jako teško uskladiti položaje objekata i detektirati sudar. Ako je sudar detektiran, sudarenim objektima je svojstvo *flag* postavljeno na vrijednost 1, što ih izuzima iz svih daljnih provjera i računa i vodi ih kao "uništene". Stvara se novo tijelo kao produkt neelastičnog sudara, a njegova se svojstva računaju po izrazu (5).
Sljedeći blok koda osigurava pravilno crtanje novostvorenog objekta tako što mu se lista položaja i brzina do trenutka u kojem je detektiran sudar puni neodređenom vrijednošću *NaN*, jer u tim vremenima tijelo nije postojalo. Preostali kod Eulerovom metodom računa novu brzinu i položaj za svaki objekt.

```
def __korak(d_sudar,t,dt):
    #SUDAR
    dodavanje=[]
    for obj in Tijelo.objekti:
        for meta in Tijelo.objekti:
            if obj != meta and ((obj.r[-1][0]-meta.r[-1][0])**2+(obj.r[-1][1]-meta.r[-1][1])**2)<d_sudar**2 and meta.flag==0 and obj.flag==0:
                #Napravimo novi objekt ; neelasticni sudar, mase i brzine se zbroje, brisemo obj i metu
                obj.flag=1
                meta.flag=1
                #v = p_uk/m_uk
                #r = centar mase
                dodavanje.append([obj.m+meta.m,
                                  [(obj.m*obj.r[-1][0]+meta.m*meta.r[-1][0])/(obj.m+meta.m),(obj.m*obj.r[-1][1]+meta.m*meta.r[-1][1])/(obj.m+meta.m)],
                                   [(obj.m*obj.v[-1][0]+meta.m*meta.v[-1][0])/(obj.m+meta.m), (obj.m*obj.v[-1][1]+meta.m*meta.v[-1][1])/(obj.m+meta.m)],
                                   obj.ime+" + "+meta.ime])
    for el in dodavanje:
        novo_tijelo= Tijelo(el[0],[float("nan"), float("nan")], [float("nan"), float("nan")],el[3])
        for i in np.arange(0,t-dt,dt):
            novo_tijelo.v.append([float("nan"), float("nan")])
            novo_tijelo.r.append([float("nan"), float("nan")])
        novo_tijelo.v.append(el[2])
        novo_tijelo.r.append(el[1])


    for obj in Tijelo.objekti:
        if obj.flag == 0:
            obj.akceleracija()
        else:
            obj.a=[float("nan"),float("nan")]
    for obj in Tijelo.objekti:
        if obj.flag == 0:
            obj.v.append([obj.v[-1][0]+obj.a[0]*dt, obj.v[-1][1]+obj.a[1]*dt])
        else:
            obj.v.append([float("nan"), float("nan")])
    for obj in Tijelo.objekti:
        if obj.flag == 0:
            obj.r.append([obj.r[-1][0]+obj.v[-1][0]*dt, obj.r[-1][1]+obj.v[-1][1]*dt])
        else:
            obj.r.append([float("nan"), float("nan")])
```

Slika 2: Metoda *korak*

Metoda *gibanje* poziva metodu *korak* sve dok $t < t\_fin$, odnosno dok je trenutno vrijeme manje od finalnog i izrađuje animaciju koristeći modul Matplotlib.

```
def gibanje(t_fin,dt):

    t=0
    while t<t_fin:
        Tijelo.__korak(1e7,t,dt)
        t+=dt

    crtaj_svakih_n=2 #Izgleda optimalno, crta svaku 2. tocku
    frames = range(0, len(Tijelo.objekti[0].r), crtaj_svakih_n)
    anim = animation.FuncAnimation(Tijelo.fig,Tijelo.crtanje,init_func=Tijelo.init, frames=frames, interval=20, blit=False, repeat=False)
    #plt.legend()
    plt.show()
```

Slika 3: Metoda *gibanje*

Preostale dvije metode, *crtanje* i *init* potrebne su za crtanje animacije.

```
def crtanje(frame):
    """for i, obj in enumerate(Tijelo.objekti):
        x_data = [r[0] for r in obj.r[:frame]]
        y_data = [r[1] for r in obj.r[:frame]]
        Tijelo.lines[i].set_data(x_data, y_data)""" #Za crtanje pune crte; bez repa
    rep = 100  # broj zadnjih točaka koje crtamo
    for i, obj in enumerate(Tijelo.objekti):
        x_data = [r[0] for r in obj.r[max(0,frame - rep):frame]]
        y_data = [r[1] for r in obj.r[max(0,frame - rep):frame]] #Mora bit max jer prije 100 frameova ne bi bilo slike, od 0 do frame dok je nframes<100
        Tijelo.lines[i].set_data(x_data, y_data)
    return Tijelo.lines
@staticmethod
def init():
    Tijelo.lines = []
    for obj in Tijelo.objekti:
        (line,) = Tijelo.ax.plot([], [], label=obj.ime, lw=obj.lw)
        Tijelo.lines.append(line)
    Tijelo.ax.set_xlim(-5e11, 5e11)
    Tijelo.ax.set_ylim(-5e11, 5e11)
    Tijelo.ax.set_xlabel("x/m")
    Tijelo.ax.set_ylabel("y/m")
    Tijelo.ax.legend()
    return Tijelo.lines
```

Slika 4: Metode *crtanje* i *init*

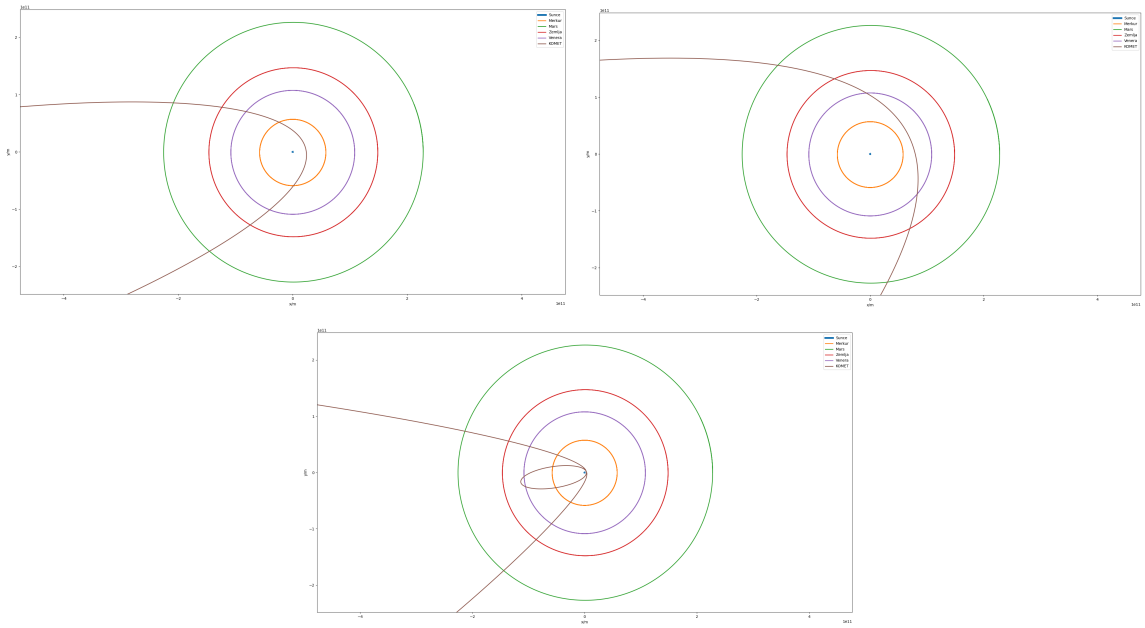
U ovom primjeru konstruiran je Sunčev sustav koji se sastoji od Sunca, Merkura, Venere, Zemlje, Marsa i kometa koji se giba prema Sunčevom sustavu. Njihove početne karakteristike dane su u slici dolje, a gibanje se provodilo za period od dvije godine.

```
Sunce = Tijelo(1.989e30,[0,0], [0,0], "Sunce",5)
Merkur = Tijelo(3.3011e23, [5.791e10,0], [0, 4.787e4], "Merkur")
Mars = Tijelo(6.4171e23, [2.2794e11,0], [0, 2.407e4], "Mars")
Zemlja = Tijelo(5.9742e24,[1.486e11,0],[0, 2.9783e4], "Zemlja")
Venera = Tijelo(4.8675e24, [1.0821e11,0],[0,3.502e4], "Venera")
Komet = Tijelo(1e14,[-4.5*149.6e9,5.791e10],[17.107e3,2210], "KOMET") #Valja se igrat sa masom  kometa, da se vidi sudar
Tijelo.gibanje(365.242*24*60*60*2, 3600*12) #2 godine
```

Slika 5: Zadana tijela i njihove početne karakteristike

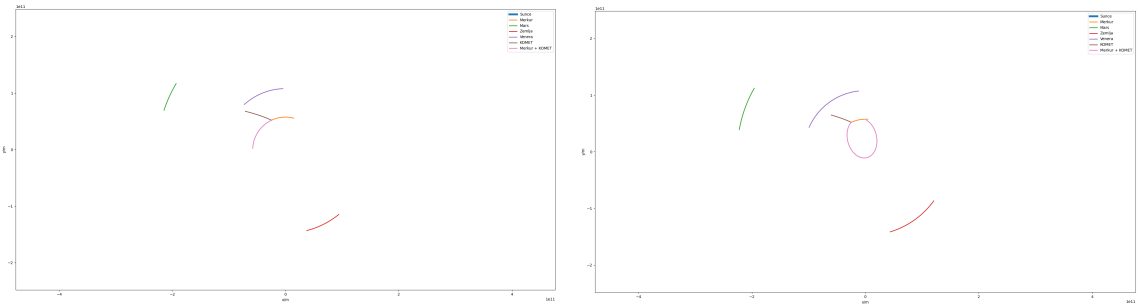
3 Rezultati

Na slici 6 prikazane su putanje kometa koji prolazi kroz sunčev sustav za različite početne pozicije i brzine. Podatci o masi, udaljenosti i srednoj orbitalnoj brzini planeta uzeti su iz stvarnih podataka [1] i ostaju isti u svim primjerima, a udaljenost kometa od Sunca je u svim primjerima približno jednaka 4.5AU.
U prvom primjeru (gore lijevo) za početnu y koordinatu uzet je radijus orbite Merkura. Komponente brzine kometa su $v_x = 17103\text{m/s}$, $v_y = 2210\text{m/s}$, a vremenski korak dt je 12 sati.
U drugom primjeru (gore desno) za početnu y koordinatu uzet je radijus orbite Zemlje, a x komponenta brzine je povećana na $v_x = 20000\text{m/s}$. y komponenta brzine i vremenski korak isti su kao u prvom primjeru.
U trećem primjeru (dolje) početni položaj kometa isti je kao u drugom primjeru, a brzina je smanjena i iznosi $v_x = 15000\text{m/s}$, $v_y = -2000\text{m/s}$. Vremenski korak u ovom primjeru je 6 sati.



Slika 6: Primjeri putanja kometa za različite početne parametre

Primjeri u kojima dolazi do sudara kometa i Merkura prikazani su na slici 7. Početni položaji i brzine kometa isti su u oba primjera i iznose $x = -4.5\text{AU}$, $y = 5.791 \cdot 10^{10}\text{m}$, $v_x = 17050.435\text{m/s}$, $v_y = 2200\text{m/s}$, a udaljenost za detekciju sudara *d_sudar* je $3.5 \cdot 10^8\text{m}$. Razlog za tako veliku udaljenost za detekciju sudara je taj što se, moguće zbog numeričke pogreške, pokazalo izrazito teško namjestiti početne parametre kometa tako da prođe bliže Merkuru. U prvom primjeru (lijevo) masa kometa je postavljena na 10^{14}kg. Uočljivo je da nema vidljive promjene u orbiti Merkura, jer je masa kometa 9 redova veličine manja od mase Merkura. U drugom primjeru (desno) masa kometa postavljena je na 10^{23}kg, što je približno 1/3 mase Merkura. U ovom je primjeru vidljiva značajna promjena orbite Merkura.



Slika 7: Primjeri u kojima dolazi do sudara kometa i Merkura

4 Zaključak

Simulacija prolaska kometa kroz Sunčev sustav može se na relativno jednostavan način prikazati u programskom jeziku Python. Ova simulacija prikazuje gibanje objekata samo u ravnini, što nije potpuno ispravan prikaz, ali se otvara mogućnost da se kod proširi na način da prikazuje i gibanje u trećoj dimenziji, što bi bilo fizikalno točnije. Osim toga, moguće je implementacijom drugih metoda za računanje brzine i položaja smanjiti numeričku pogrešku da se lakše pronađu početni uvjeti koji bi uzrokovali sudar dvaju ili više tijela.

Literatura

[1] Williams, D.R., NASA(2025) *Planetary Fact Sheet - Metric.* Dostupno na https://nssdc.gsfc.nasa.gov/planetary/factsheet/ (Pristupljeno 24.6.2025.)

[2] Dulčić, A., Poljak, N., Požek, M. , 2023., *Mehanika*, Zagreb: Školska knjiga