9. Obične diferencijalne jednačine, problem početne vrednosti

1. Drugi Njutnov zakon

Zadatak 1

Ako na telo mase $1\,kg$, koje je u trenutku $0\,s$ imalo položaj $0\,m$ i brzinu $0\,\frac{m}{s}$, deluje konstantna sila od $10\,N$, naći položaj tela svake sekunde tokom narednih $10\,s$.

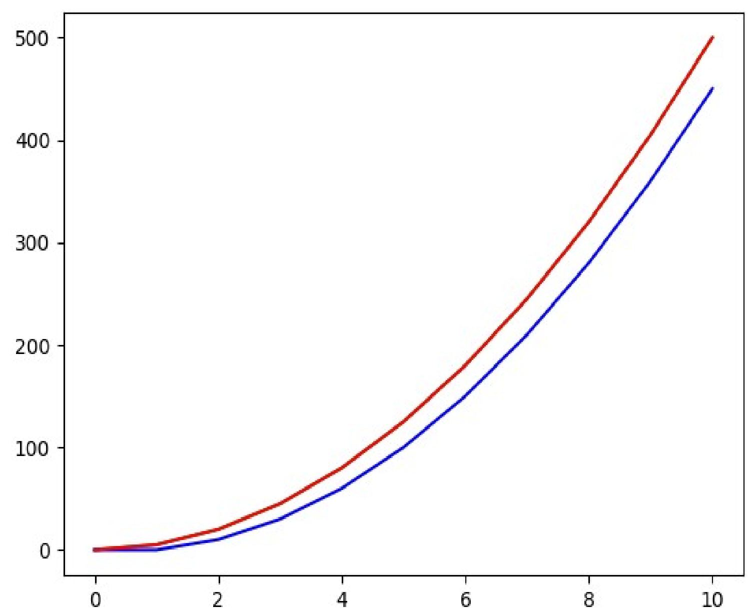
$$\frac{d^2 s}{dt^2}=\frac{F}{m}$$
$$s''(t)=\frac{F}{m}=\frac{10}{1}=10$$
$$s(0)=0$$
$$s'(0)=v(0)=0$$

Naći numeričko rešenje Ojlerovim metodom i metodom RK4 u 10 tačaka, zadajući korak h. Na istom grafiku nacrtati i uporediti 2 numerička i analitičko rešenje ($s(t)=v_0 t+\frac{F}{m}\frac{t^2}{2}$).

Uporediti nalaženje numeričkih rešenja u 10, 100, 1000, 10000 tačaka, menjajući korak h.

Rešenje (za 10 tačaka):

```
sTTrue =
     0     5    20    45    80   125   180   245   320   405   500
sTEuler =
     0     0    10    30    60   100   150   210   280   360   450
sTRK4 =
     0     5    20    45    80   125   180   245   320   405   500
```



Ojlerova metoda greši za veliki korak h (tj. za mali broj tačaka)!

2. Diskretne jednačine

Zadatak 2

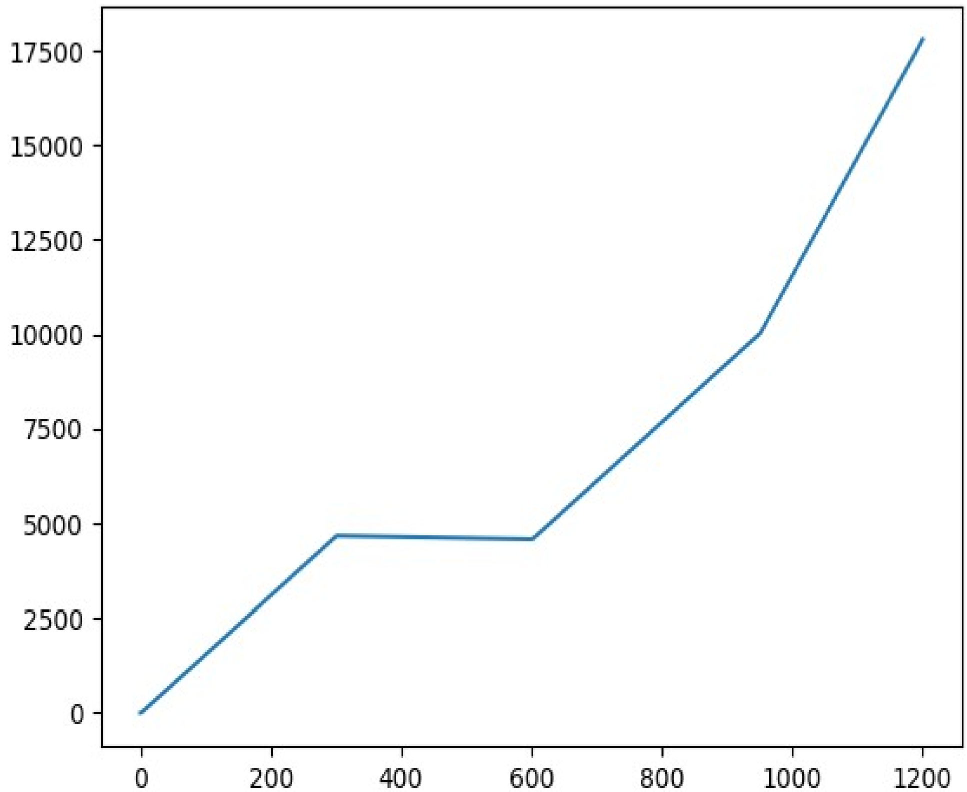
Pri dodavanju gasa vozlio ubrzava $5\frac{m}{s^2}$. Pri kočenju vozlio usporava $-10\frac{m}{s^2}$. Ako se vozilo iz stanja mirovanja kretalo po deonici puta dužine $10\,km$ sa ograničenjem brzine $60\frac{km}{h}$, a zatim ostatak puta sa ograničenjem brzine od $120\frac{km}{h}$ i ako je napravilo pauzu između 5. i 10. min., koliki put je prešlo nakon 20min?

a) Formulisati diferencijalnu jednačinu u posebnoj MATLAB datoteci (uz pomoć `if` selekcije definisati diskretne uslove kretanja):

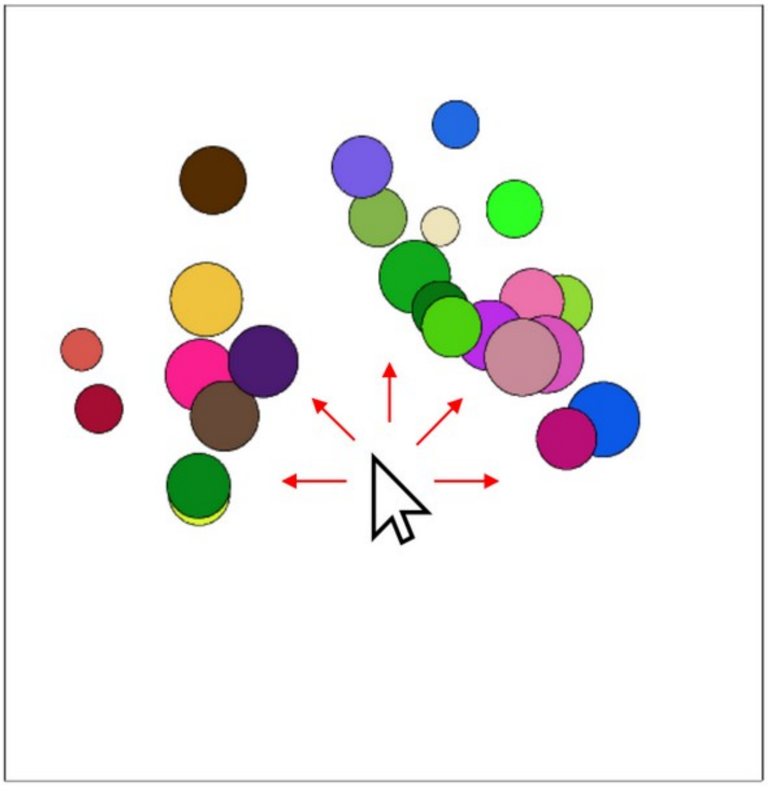
```
def fp2(t, sT, dsT):
    #  t - vreme proteklo od pocetka putovanja
    #  sT - predjeni put
    #  dsT - trenutna brzina vozila
    v = dsT
    #  ogranicenje brzine
    speedLimit = 60*1000/3600 # 60km/h
    if sT > 10*1000: # nakon 10km
        speedLimit = 120*1000/3600 #  120km/h
    acc = 5 # 5m/(s^2)
    dec = -10 # -10m/(s^2)
    if t > 5*60 and t < 10*60: # pauza izmedju 5. i 10. min.
        if v >= 0: # usporavati sve dok vozilo ne stane
            a = dec
        else: # ako vozilo "krene u nazad" usled negativnog ubrzanja, ubrzati ponovo do 0
            a = acc
    else:
        if v <= speedLimit: # ubrzavati ako je brzina ispod ograni?enja
            a = acc
        else:
            a = dec # usporiti ako je brzina preko ograni?enja
    ddsT = a
    return ddsT
```

b) Rešiti PPV

rešenje: $s(20\,min)=17.802\,km$


17500
15000
12500
10000
7500
5000
2500
0
0
200
400
600
800
1000
1200

3. Fizika u video igrama (interaktivno kretanje)



Slika 1. Interaktivno kretanje

Poznavajući fizičke karakteristike objekata i početne uslove kretanja, potrebno je simulirati njihovo kretanje u proizvoljnom vremenskom intervalu, pri čemu uslovi kretanja mogu da se menjaju u svakom vremenskom trenutku (slika 1). Radi jednostavnosti primera odabrane su sfere.
Jednačina položaja takvog kretanja (2. Njutnov zakon) je:

$$\frac{d^2\vec{p}(t)}{dt^2}=\frac{\vec{F}(t)}{m} \tag{1}$$

, gde su $\vec{p}$ trenutni položaj tela, m je masa tela, $\vec{F}$ je sila koja deluje na telo, a t proteklo vreme. $\vec{F}(t)$ je veličina koja menja uslove kretanja u svakom vremenskom trenutku. $\vec{F}(t)$ je nepoznata funkcija i zavisi od korisničke interakcije. Nije moguće izraziti ovu funkciju analitički.

Funkciju položaja je međutim u svakom vremenskom trenutku $(t+\Delta t)$ moguće razviti u Tejlorov red:

$$\vec{p}(t+\Delta t)=\vec{p}(t)+\Delta t\frac{d\vec{p}(t)}{dt}+\frac{\Delta t^2}{2}\frac{d^2\vec{p}(t)}{dt^2}+\ldots+\frac{\Delta t^n}{n!}\frac{d^n\vec{p}(t)}{dt^n}$$

Ograničavanjem beskonačne sume Tejlorovog reda na konačnu, moguće je numeričkim putem doći do rešenja diferencijalne jednačine u uzastopnim vremenskim trenucima.
Ojlerov metod uzima u obzir prva 2 člana reda:

$$\vec{p}(t+\Delta t)\approx\vec{p}(t)+\Delta t\frac{d\vec{p}(t)}{dt}$$

Diferenciranjem obe strane jednačine dobija se:

$$\vec{p}(t+\Delta t)=\vec{p}(t)+\Delta t\frac{d\vec{p}(t)}{dt}$$

$$\frac{d\vec{p}(t+\Delta t)}{dt}=\frac{d\vec{p}(t)}{dt}+\Delta t\frac{d^2\vec{p}(t)}{dt^2}$$

Uvođenjem smene $\frac{d\vec{p}(t)}{dt}=\vec{v}(t)$, dobija se:

$$\vec{p}(t+\Delta t)=\vec{p}(t)+\Delta t\,\vec{v}(t)$$

$$\vec{v}(t+\Delta t)=\vec{v}(t)+\Delta t\frac{d^2\vec{p}(t)}{dt^2} \qquad (2)$$

Zamenom (1) u (2), dobija se:

$$\vec{p}(t+\Delta t)=\vec{p}(t)+\Delta t\,\vec{v}(t)$$

$$\vec{v}(t+\Delta t)=\vec{v}(t)+\Delta t\frac{\vec{F}(t)}{m}$$

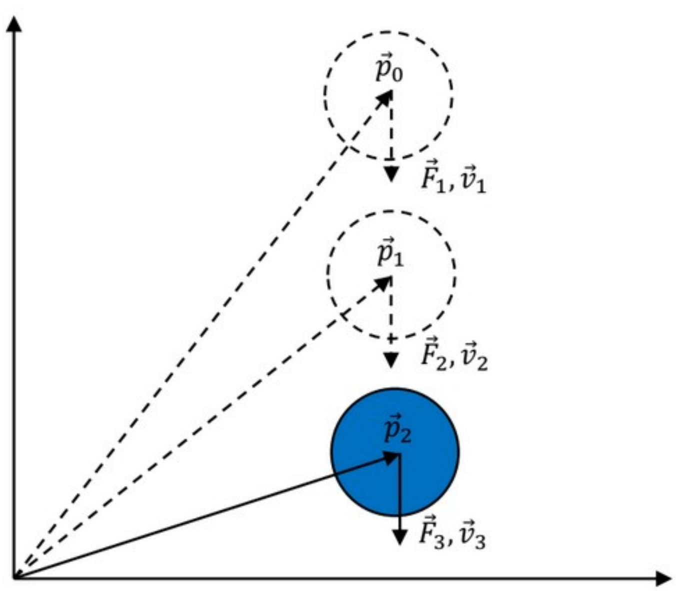
Prevođenjem u iterativni zapis, dobija se:

$$\vec{p}_i=\vec{p}_{i-1}+\Delta t\,\vec{v}_{i-1}$$

$$\vec{v}_i=\vec{v}_{i-1}+\Delta t\frac{\vec{F}_{i-1}}{m}$$

Masa tela m je konstantna. Ako je silu $\vec{F}_{i-1}$ u svakom trenutku moguće izraziti i izračunati i ako su početni položaj $\vec{p}_0$ i početna brzina $\vec{v}_0$ tela poznati, tada je **Ojlerovom integracijom** moguće naći položaj $\vec{p}_i$ i brzinu $\vec{v}_i$ u svakom vremenskom trenutku $(0+i\,\Delta t)$:

0	$0+\Delta t$	$0+2\Delta t$	$0+i\,\Delta t$
$\vec{p}_0$	$\vec{p}_1=\vec{p}_0+\Delta t\,\vec{v}_0$	$\vec{p}_2=\vec{p}_1+\Delta t\,\vec{v}_1$	$\vec{p}_i=\vec{p}_{i-1}+\Delta t\,\vec{v}_{i-1}$
$\vec{v}_0$	$\vec{v}_1=\vec{v}_0+\Delta t\frac{\vec{F}_0}{m}$	$\vec{v}_2=\vec{v}_1+\Delta t\frac{\vec{F}_1}{m}$	$\vec{v}_i=\vec{v}_{i-1}+\Delta t\frac{\vec{F}_{i-1}}{m}$



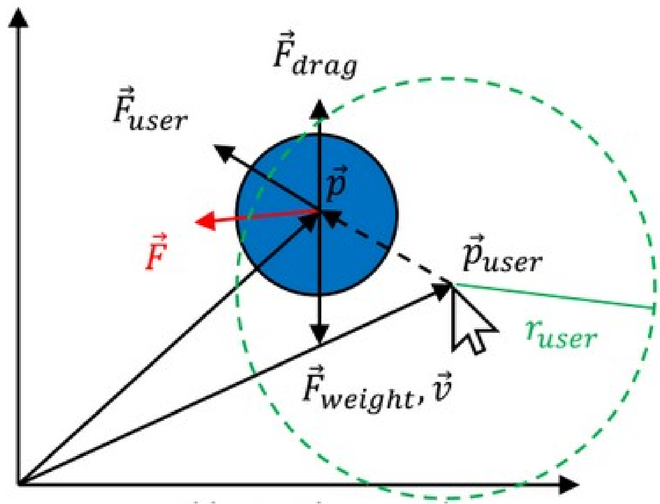
Slika 2. Ojlerova integracija

RK4 metoda podrazumeva iste parametre, pa se može upotrebiti na **isti način**.

Ako su poluprečnik r i gustina sfere ρ poznati (za gumu npr. $\rho_{rubber}=1522\frac{kg}{m^3}$), masa sfere se može izračunati na sledeći način:

$$m=\rho_{rubber}V=\rho_{rubber}\frac{4}{3}r^3\pi$$

Ukupna sila $\vec{F}$ koja deluje na telo u svakom trenutku se može izračunati kao linearna kombinacija različitih komponenata:



Slika 3. Ukupna sila

$$\vec{F}=\vec{F}_{weight}+\vec{F}_{drag}+\vec{F}_{user}$$

Težina tela $\vec{F}_{weight}$ se izračunava na sledeći način:

$$\vec{F}_{weight}=m\vec{g}$$

, gde je m masa tela, a $\vec{g}=[0, \quad -9.81]\frac{m}{s^2}$ je gravitaciono ubrzanje.

Sila otpora vazduha $\vec{F}_{drag}$ se izračunava na sledeći način:

$$\vec{F}_{drag}=\vec{v}^2\frac{1}{2}\rho_{air}C_d A=-v|\vec{v}|\frac{1}{2}\rho_{air}C_d A$$

, gde je $\rho_{air}=1.225\frac{kg}{m^3}$ gustina vazduha, $\vec{v}$ je brzina tela, C_d je koeficijent aerodinamičnosti tela (za sfere $C_d=0.47$), a A je poprečni presek tela normalan na pravac kretanja (za sfere $A=r^2\pi$). Primetiti da je sila otpora vazduha po pravcu ista, a po smeru suprotna brzini kretanja.

Korisnička sila $\vec{F}_{user}$ se izračunava na sledeći način:

$$\vec{F}_{user}=\begin{cases} 0 & , \quad |\vec{p}-\vec{p}_{user}|>r_{user} \\ \frac{\vec{p}-\vec{p}_{user}}{|\vec{p}-\vec{p}_{user}|}f_{user} & , \quad |\vec{p}-\vec{p}_{user}|\leq r_{user} \end{cases}$$

, gde je $\vec{p}$ položaj tela, $\vec{p}_{user}$ je položaj pokazivača miša, r_{user} poluprečnik kruga delovanja korisničke sile, a f_{user} je intenzitet korisničke sile.

Zadatak 3

Simulirati slobodan pad gumenih sfera nasumično generisanih položaja iz stanja mirovanja kroz vazduh. Omogućiti da korsinik pokazivačem miša može da unese dodatnu silu u simulaciju.

Konstante prostora:

dimenzije prostora	$(width, height)=[10, \quad 10]m$
gravitaciono ubrzanje	$\vec{g}(x,y)=[0, \quad -9.81]\frac{m}{s^2}$
gustina vazduha	$\rho_{air}=1.225\frac{kg}{m^3}$
gustina gume	$\rho_{rubber}=1522\frac{kg}{m^3}$
koeficijent aerodinamičnosti sfere	$C_d=0.47$

Sfere:

veličina	min. nasumično generisana vrednost	maks. nasumično generisana vrednost
broj	25	25
poluprečnik	$r_{min}=0.25m$	$r_{max}=0.5m$
početne brzine sfera	$\vec{v}_{min}(x,y)=[0, \quad 0]\frac{m}{s}$	$\vec{v}_{max}(x,y)=[0, \quad 0]\frac{m}{s}$
položaji sfera	$\vec{p}_{min}(x,y)=[2.5 \quad 7.5]m$	$\vec{p}_{max}(x,y)=[7.5 \quad 12.5]m$

Korisnička sila:

poluprečnik kruga delovanja korisničke sile	$r_{user}=2m$
intenzitet korisničke sile	$f_{user}=15000\,N$

c) Definisati konstante prostora:

```
worldSize = [10.0, 10.0]  # [m]; dimenzije prostora

g = 9.81  # [m/s^2]; gravitaciono ubrzanje
air_density = 1.225  # [kg/m^3]; gustina vazduha
rubber_density = 1522  # [kg/m^3]; gustina gume
drag_coefficient = 0.47  # koeficijent aerodinamičnosti sfera
```

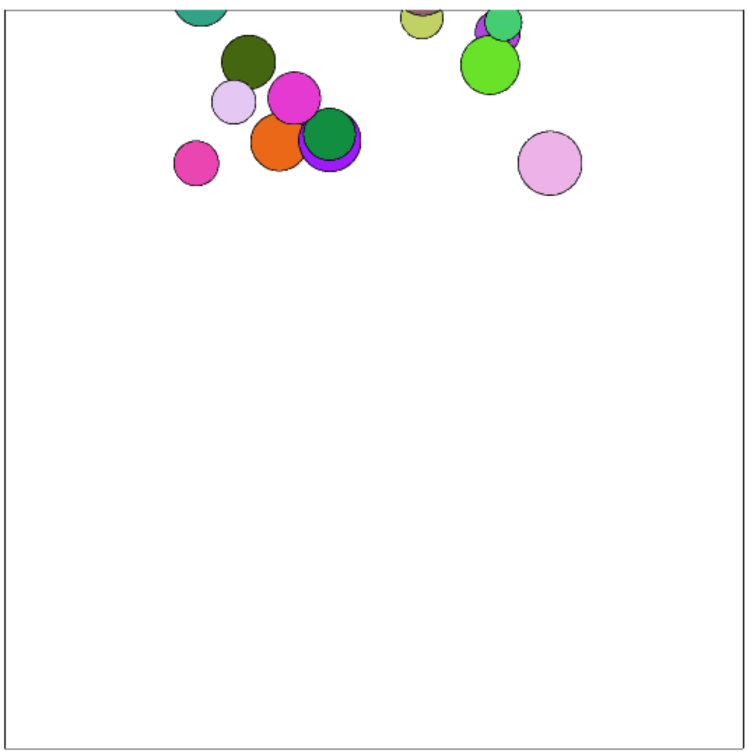
d) Definisati sfere:

```
# sfere
sphere_count = 25
r = (0.5 + np.random(sphere_count)*0.5)*0.5  # [m]; dimenzije sfera
A = r**2*np.pi  # [m^2]; poprečni preseci sfera
m = rubber_density*4/3*r**3*np.pi  # [kg]; mase (gumenih) sfera
v = np.zeros((sphere_count, 2))  # [m/s]; trenutne brzine sfera
# [m]; trenutni polozaji sfera
p = np.random.rand(sphere_count, 2)*0.5
p[:, 0] = (0.25 + p[:, 0])*world_size[0]
p[:, 1] = (0.75 + p[:, 1])*world_size[1]
colors = np.random.rand(sphere_count, 3)  # (R,G,B); boje sfera
```

e) Inicijalizovati grafički interfejs:

```
% GUI
fig, ax = plt.subplots()  # prozor
plt.axis((0, world_size[0], 0, world_size[1]))  # ogranicavanje prikaza u okviru dimenzija prostora
ax.set_aspect('equal')  # sprecavanje reskaliranja prikaza
plt.axis('off')  # sakrivanje osa


# ivice
plt.plot([0, world_size[0]], [0, 0], c='k')
plt.plot([0, world_size[0]], [world_size[1], world_size[1]], c='k')
plt.plot([0, 0], [0, world_size[1]], c='k')
plt.plot([world_size[0], world_size[0]], [0, world_size[1]], c='k')


# sfere
spheres = []
for sphere in range(sphere_count):  # za svaku sferu(sphere)
    location = p[sphere, :]
    radius = r[sphere]
    diameter = 2*radius
    x = location[0] - radius
    y = location[1] - radius
    color = colors[sphere, :]
    spheres.append(plt.Circle((x, y), radius, facecolor=colors[sphere], edgecolor='black'))
```

Rezultat:



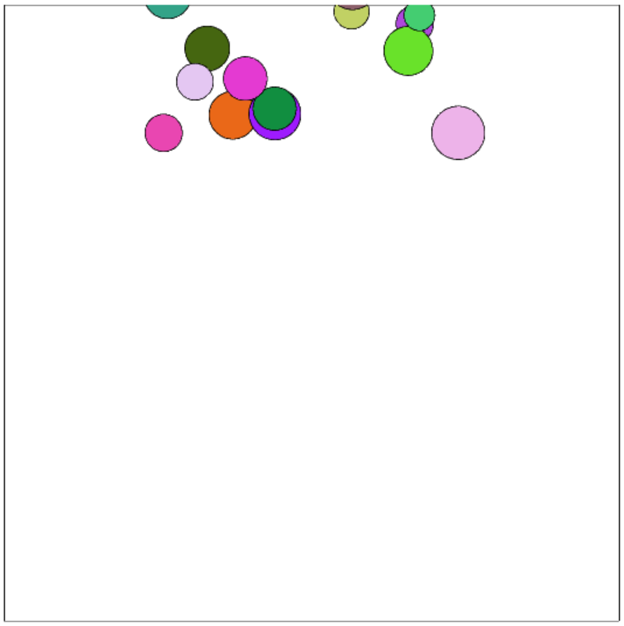
Slika 4. Početak simulacije

f) Simulirati kretanje sfera Ojlerovom integracijom:

```
fps = 60  # broj osvežavanja prikaza u sekundi
timeScale = 1.0  # brzina simulacije
t1 = 0  # [s]; početni vremenski trenutak
dt = 1/fps  # [s]; vremenska razlika između koraka


def init(): # inicijalizacija pocetnih pozicija
    for sphere in spheres:
        ax.add_patch(sphere)
    return spheres


# funkcija koja se poziva prilikom iscrtavanja svakog frame-a
def animate(f):
    t2 = t1 + dt*time_scale  # naredni vremenski trenutak

    # azuriranje polozaja i iscrtavanje
    for idx, sphere in enumerate(spheres):
        # sile
        F = [0, 0]


        # integracija
        #
-------------------------------------------------------------
        ddpX = lambda t, p, v: F[0]/m[idx]  # p"(t) = F(t)/m) funkcija kretanja (x)
        ddpY = lambda t, p, v: F[1]/m[idx]  # p"(t) = F(t)/m) funkcija kretanja (y)
        _, pnX = eulerN.eulerN(t1, t2, t2 - t1, np.array([p[idx, 0], v[idx, 0]]), ddpX, 0.0)# integracija(x)
        _, pnY = eulerN.eulerN(t1, t2, t2 - t1, np.array([p[idx, 1], v[idx, 1]]), ddpY, 0.0)# integracija(y)
        p[idx, :] = [pnX[0, -1], pnY[0, -1]]  # trenutni polozaj
        v[idx, :] = [pnX[1, -1], pnY[1, -1]]  # trenutna brzina

        # prikaz
        # -------------------------------------------------------------
        location = p[idx, :]
        radius = r[idx]
        x = location[0] - radius
        y = location[1] - radius
        sphere.center = (x, y)
        ax.add_patch(sphere)
    return spheres
```

$$(p_x(t)) = \begin{bmatrix} p_x(t_1) \\ v_x(t_1) \end{bmatrix} \begin{bmatrix} p_x(t_2) \\ v_x(t_2) \end{bmatrix}$$

Rezultat:

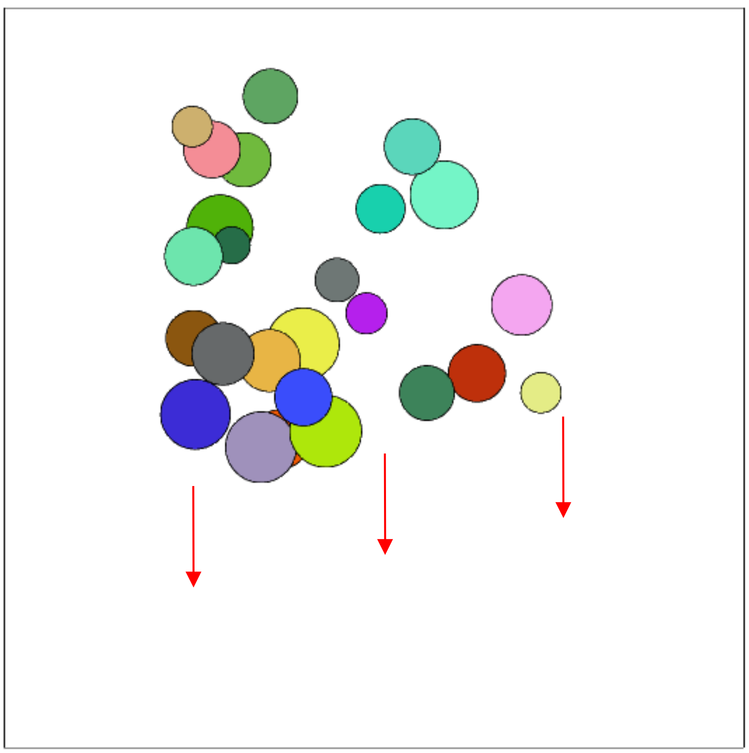


Slika 5. Bez sile kretanje nije moguće

g) Uvesti sile težine tela i otpora vazduha:

```
    .
    .
    .
    # azuriranje polozaja i iscrtavanje
    for idx, sphere in enumerate(spheres):
        # sile
        # ---------------------------------------------------------------
        F_weight = [0, -m[idx]*g]  # tezina tela (x, y)

        velocity = v[idx, :]
        F_drag = -velocity*np.linalg.norm(velocity, 2)*0.5*air_density*drag_coefficient*A[idx]  # otpor
vazduha (x, y)

        F = F_weight + F_drag

        # integracija
        # -----------------------------------------------------------------------
        .
        .
        .
```

Rezultat:



Slika 6. Slobodan pad

h) Definisati funkciju `normalize(in_vector)` koja za ulazni vektor `in_vector` vraća jedinični vektor `out` istog pravca i smera:

```
def normalize(in_vector):
    magnitude = np.linalg.norm(in_vector, 2)
    if magnitude == np.inf or magnitude <= 0:
        out = [1, 0]
    else:
        out = in_vector/magnitude
    return out
```

i) Definisati funkciju `gui_mouse_move(event)` koja na događaj pomeranja pokazivača miša čuva položaj miša u globalnoj promenljivoj `mouse_location`:

```
mouse_location = [None, None] # inicijalizacija globalne promenljive - pozicija misa
def gui_mouse_move(event):
    global mouse_location
    mouse_location = [event.xdata, event.ydata]
```

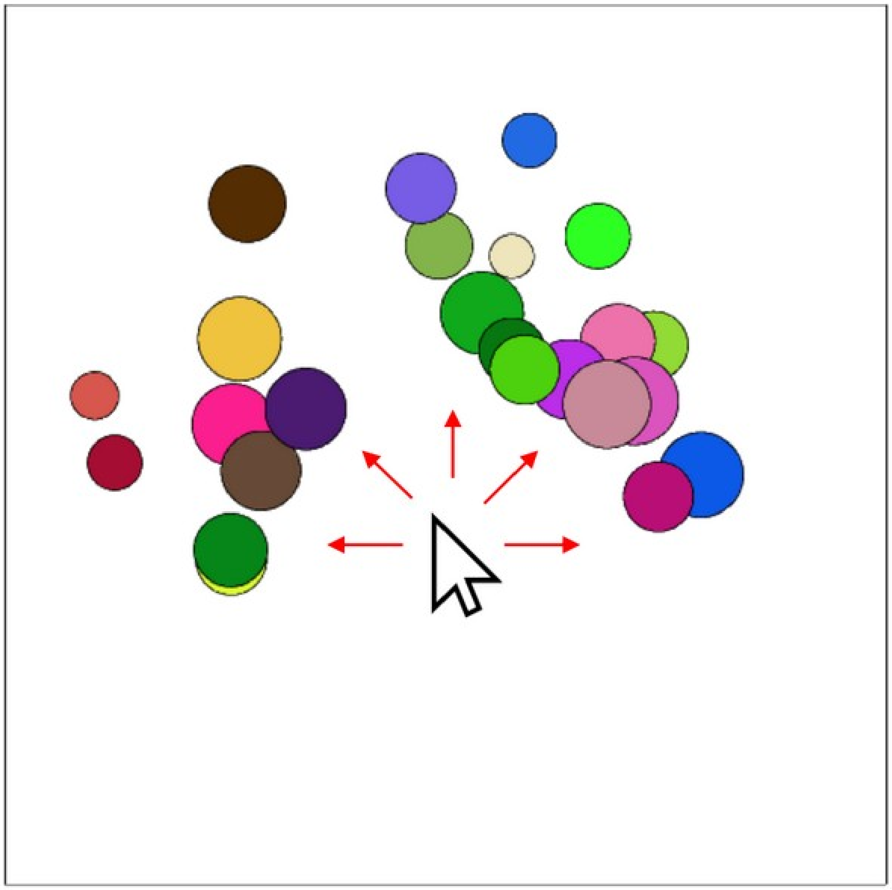
j) Pre procedure iz koraka d), definisati parametre korisničke sile i registrovati funkciju `gui_mouse_move` da reaguje na događaj pomeranja pokazivača miša:

```
# user force
r_user = min(world_size) * 0.2  # [m]
f_user = 15000  # [N = kg * m / s^2]
p_user = [-np.inf, -np.inf]
plt.connect('motion_notify_event', gui_mouse_move)  # registrovanje funkcije
```

k) Uvesti delovanje korisničke sile:

```
 .
 .
 .
 pUser = mouse_location  # čitanje vrednosti globalne promenljive

 # ažuriranje položaja i iscrtavanje
 # azuriranje polozaja i iscrtavanje
for idx, sphere in enumerate(spheres):
    # sile
    # --------------------------------------------------------------
     .
     .
     .
     F_user = [0, 0]
     try:
         direction = p[idx, :] - p_user
         if np.linalg.norm(direction, 2) <= r_user:
             F_user = normalize(direction)*f_user  # korisnicka sila (x, y)
     except:
         pass

     F = F_weight + F_drag + F_user

     # integracija
     # -----------------------------------------------------------------------
     .
     .
     .
```

Rezultat:



Slika 7. Interaktivno kretanje