Numerična matematika - domača naloga 3

Matematično nihalo

Gregor Kovač

Simulirati želimo nedušeno nihanje matematičnega nihala, ki je podano z diferencialno enačbo $\frac{g}{l}\sin\theta(t) + \theta''(t) = 0$ z začetnima pogojema $\theta(0) = \theta_0$ in $\theta'(0) = \theta'_0$, kjer je $g = 9.80665\frac{m}{s^2}$ gravitacijski pospešek in l dolžina nihala.

Uvedemo spremenljivki $\theta_1(t) = \theta(t)$ in $\theta_2(t) = \theta'_1(t)$. Od tu dobimo sistem dveh diferencialnih enačb prvega reda:

$\theta'_1(t) = \theta_2(t)$

$\theta'_2(t) = -\frac{g}{l}\sin(\theta_1(t))$.

Ta sistem rešujemo z metodo Runge-Kutta reda 4.

Osnovna simulacija

In [ ]:
```
import numpy as np
from nihalo import *
import matplotlib.pyplot as plt

plt.rc('text', usetex=True)
plt.rc('font', family='serif')
plt.rc('font', size=16)
```

Simulirajmo nihanje in izrišimo nihalo ter spremembo kota skozi čas.

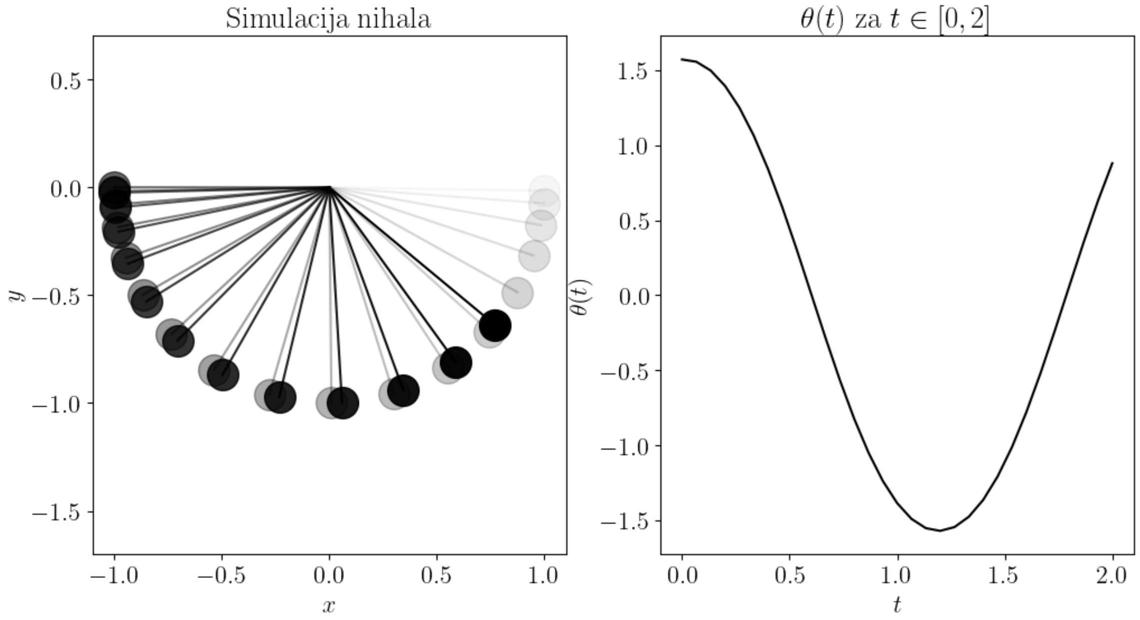
Opomba: na sliki, ki simulira nihalo, prosojnost predstavlja točko v času. Bolj kot je nihalo prosojno, bliže je $t = 0$.

In [ ]:
```
# Simuliramo nihanje s spodnjimi parametri
l = 1
t = 2
theta0 = np.pi / 2
dtheta0 = 0.1
n = 30

theta, dtheta = nihalo(l = l, t = t, theta0 = theta0, dtheta0 = dtheta0,
```

In [ ]:
```
x = np.cos(theta)
y = np.sin(theta)
colors = np.linspace(0, 1, len(x))

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
```

```
# Subplot 1
for i in range(len(x)):
    plt.plot([0, y[i]], [0, -x[i]], color=(0, 0, 0, colors[i]))
    plt.plot(y[i], -x[i], 'o', color=(0, 0, 0, colors[i]), markersize=20)

plt.title('Simulacija nihala')
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.axis('equal')

# Subplot 2
plt.subplot(1, 2, 2)
plt.plot(np.linspace(0, t, len(theta)), theta, color='black')
plt.xlabel(r'$t$')
plt.ylabel(r'$\theta(t)$')
plt.title(rf'$\theta(t)$ za $t \in [0, {t}]$')
```

Out[ ]:
```
Text(0.5, 1.0, '$\\theta(t)$ za $t \\in [0, 2]$')
```

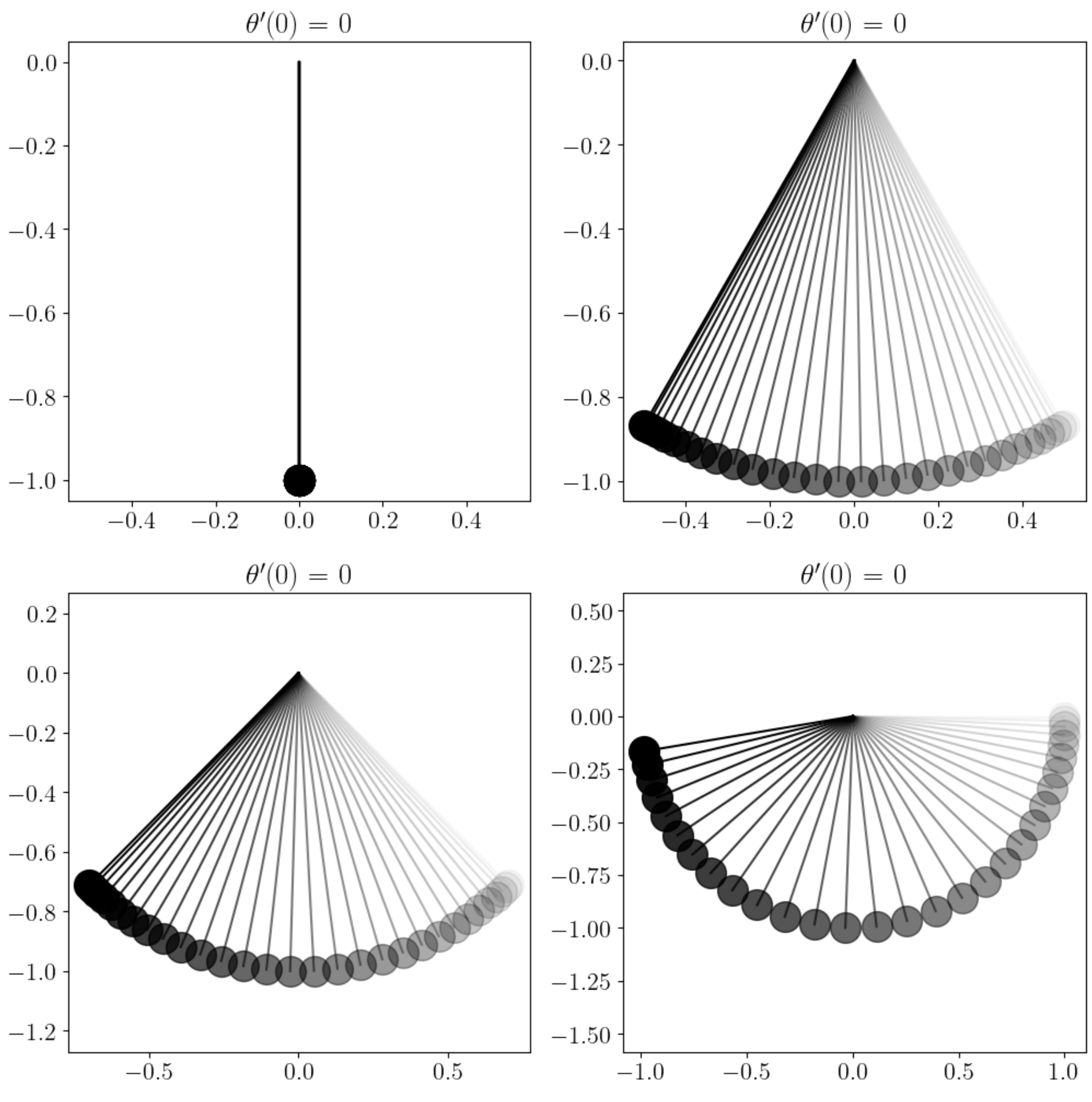


Primerjava nihal z različnimi začetnimi pogoji

Primerjajmo 4 nihala z enako začetno kotno hitrostjo $\theta_0' = 0$ in različnimi začetnimi legami.

In [ ]:
```
l = 1
t = 1
dtheta0 = 0
n = 30

plt.figure(figsize=(12, 12))

i = 1
for theta0 in [0, np.pi/6, np.pi/4, np.pi/2]:
    theta, dtheta = nihalo(l = l, t = t, theta0 = theta0, dtheta0 = dthet

    plt.subplot(2, 2, i)
    plt.title(rf"$\theta'(0)$ = {dtheta0}")
    i += 1
```

```
    x = np.cos(theta)
    y = np.sin(theta)
    colors = np.linspace(0, 1, len(x))

    for j in range(len(x)):
        c = (0, 0, 0, colors[j])

        plt.plot([0, y[j]], [0, -x[j]], color=c)
        plt.plot(y[j], -x[j], 'o', color=c, markersize=20)

    plt.axis('equal')
```

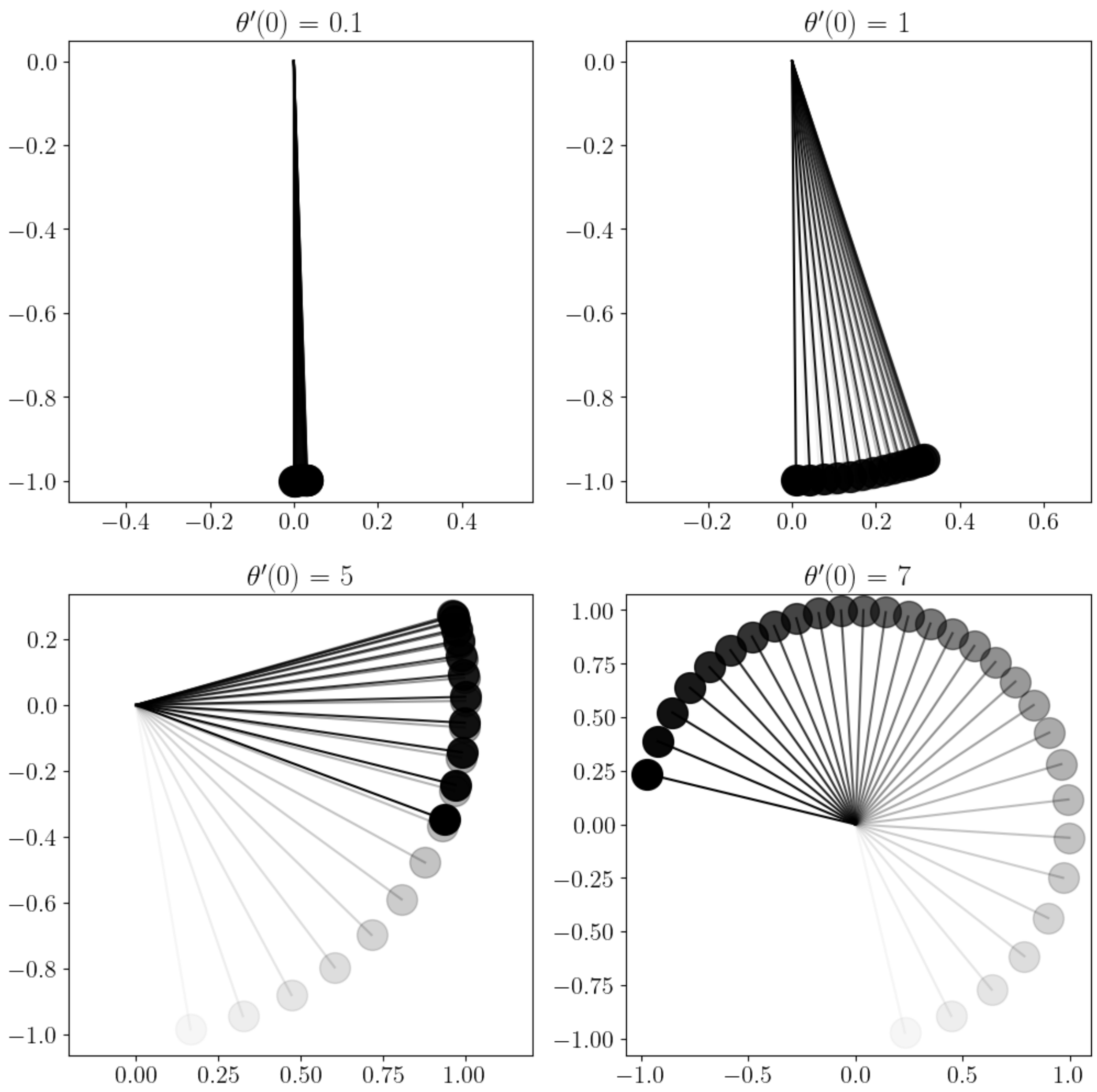


Kot pričakovano se nihalo z večjim začetnim kotom zaziba dlje.

Simulirajmo še 4 različna nihala, ki vsa začnejo v najnižji legi ($\theta_0 = 0$), a imajo različne količine začetne kotne hitrosti.

In [ ]:
```
l = 1
t = 1
theta0 = 0
n = 30

plt.figure(figsize=(12, 12))
```

```
i = 1
for dtheta0 in [0.1, 1, 5, 7]:
    theta, dtheta = nihalo(l = l, t = t, theta0 = theta0, dtheta0 = dthet

    plt.subplot(2, 2, i)
    plt.title(rf"$\theta'(0)$ = {dtheta0}")
    i += 1

    x = np.cos(theta)
    y = np.sin(theta)
    colors = np.linspace(0, 1, len(x))

    for j in range(len(x)):
        c = (0, 0, 0, colors[j])

        plt.plot([0, y[j]], [0, -x[j]], color=c)
        plt.plot(y[j], -x[j], 'o', color=c, markersize=20)

    plt.axis('equal')
```



Večja kot je začetna kotna hitrost, višje gre nihalo, kar smo pričakovali. V prvem primeru se skoraj ne premakne, v zadnjem pa naredi skoraj cel krog.

Nihajni čas

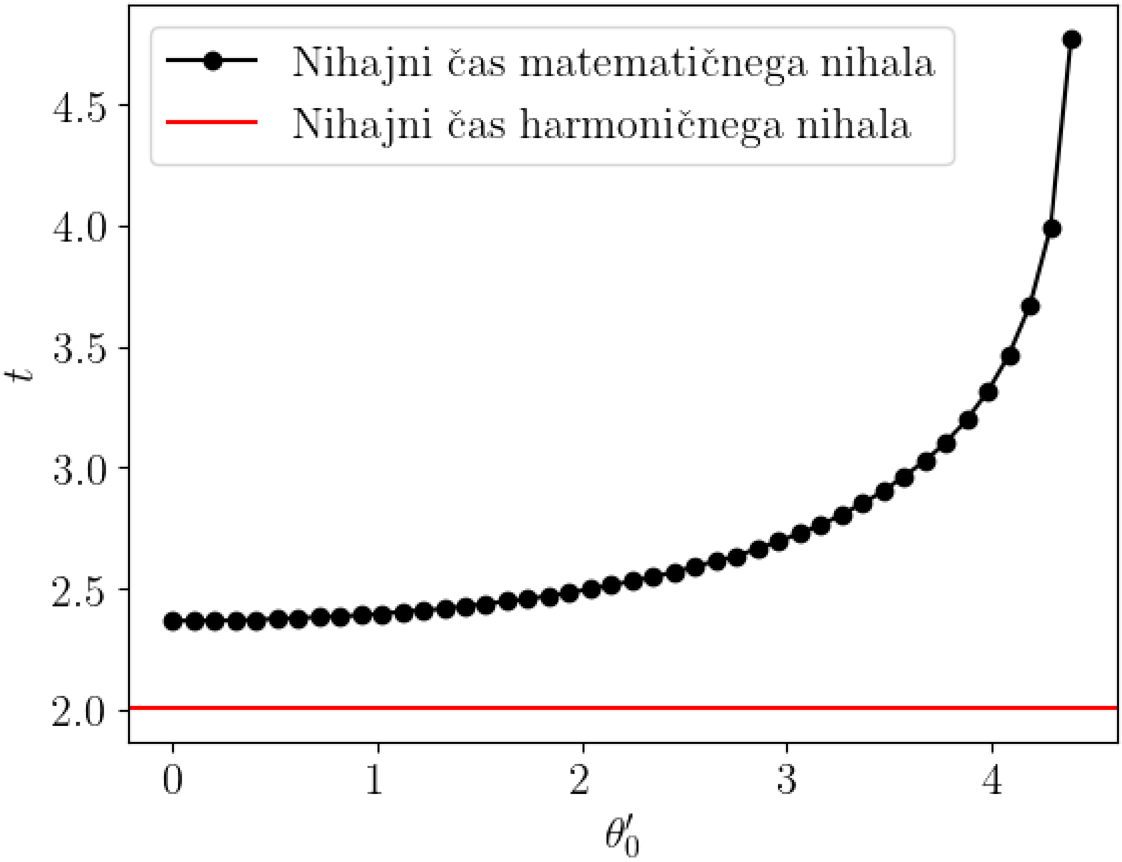
Napišimo funkcijo za izračun nihajnega časa, ki ga najlažje računamo tako, da spremljamo prehod nihala preko $\theta(t) = 0$.

In [ ]:
```
"""
nihajni_cas izračuna nihajni čas nihala
Vhod:
    l ... dolžina nihala
    theta0 ... začetni kot
    dtheta0 ... začetna kotna hitrost
    t ... končni čas nihanja
    n ... število korakov

Izhod:
    (float) ... nihajni čas
"""
def nihajni_cas(l, theta0, dtheta0, t = 10, n = 1000):
    # Simuliramo nihanje in beležimo vse korake
    theta, _ = nihalo(l = l, t = t, theta0 = theta0, dtheta0 = dtheta0, n

    # Poiščemo prehode nihala čez kot 0
    crossings = np.where(np.diff(np.sign(theta)))[0]

    # Zanemarimo robni primer
    if len(crossings) < 2:
        return np.nan

    # Izračunamo časovni korak
    dt = t / n

    # Izračunamo in vrnemo povprečni nihajni čas.
    # Rezultat množimo z 2, saj se za en nihaj upošteva prehod od 0,
    # do skrajne lege, do 0, do skrajne lege in zopet do 0.
    return 2 * np.mean(np.diff(crossings)) * dt
```

Opazujmo, kaj se zgodi z nihajnim časom pri različnih začetnih energijah. Primerjajmo ga še s harmoničnim nihalom, ki ima konstanten nihajni čas $t_0 = 2\pi\sqrt{\frac{\ell}{g}}$.

In [ ]:
```
nihajni_casi = []

for dtheta0 in np.linspace(0, 5, 50):
    nt = nihajni_cas(1, np.pi/2, dtheta0)
    if nt is not None:
        nihajni_casi.append((dtheta0, nt))

plt.plot(*zip(*nihajni_casi), 'o-', color='black', label='Nihajni čas mat

g = 9.80665
t_0_harm = 2 * np.pi * np.sqrt(1 / g)

plt.axhline(t_0_harm, color='red', label='Nihajni čas harmoničnega nihala

plt.legend()

plt.xlabel(r"$\theta'_0$")
plt.ylabel(r'$t$')
```

```
Out[ ]: Text(0, 0.5, '$t$')
```



Nihajni čas harmoničnega nihala je konstanten, za matematično nihalo pa se povečuje z večanjem začetne energije. To je smiselno, saj se nihalo z večjo začetno kotno hitrostjo višje zaziba in posledično potrebuje dlje za en nihaj.