KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Skaitiniai metodai ir algoritmai (P170B115)

Laboratorinių darbų ataskaita

Atliko:

IFF-1/4 gr. studentas

Mildaras Karvelis

2023 m. gruodžio 22 d.

Priėmė:

Prof. Barauskas Rimantas

KAUNAS 2023

#### TURINYS

1. Pirma užduotis. Interpoliavimas daugianariu Error! Bookmark not defined.

1.1. a dalis. Taškai pasiskirstę tolygiai **Error! Bookmark not defined.**

1.2. b dalis. Taškai apskaičiuojami naudojant Čiobyševo abscises **Error! Bookmark not defined.**

1.3. Programos kodas **Error! Bookmark not defined.**

2. Antra užduotis. Interpoliavimas splainu per duotus taškus Error! Bookmark not defined.

2.1. Programos kodas **Error! Bookmark not defined.**

3. Trečia užduotis. Aproksimavimas Error! Bookmark not defined.

3.1. Programos kodas **Error! Bookmark not defined.**

4. Ketvirta užduotis. Parametrinis aproksimavimas Error! Bookmark not defined.

4.1. Pradiniai duomenys **Error! Bookmark not defined.**

4.2. Aproksimavimo rezultatai **Error! Bookmark not defined.**

4.3. Programos kodas **Error! Bookmark not defined.**

# Užduotis

A close-up of a text

Description automatically generated

A screenshot of a table

Description automatically generated

# Diferencialinė lygtis

Išreiškiame judančio objekto pagreitį pasinaudoję antrąjį Niutono dėsnį:

A math equation with black text

Description automatically generated with medium confidence

Oro pasipriešinimo jėga yra proporcinga greičio kvadratui:

A math equation with black text

Description automatically generated

Iki objektų atsiskyrimo jėgų pusiausvyros lygtis :

A mathematical equation with black letters

Description automatically generated with medium confidence

Sudarome diferencialinę lygtį:

A black line with a black line in the middle

Description automatically generated with medium confidence

# Sprendimas

## Žingsnis-46

Kūnas m1 (Euler) pasiekia max aukštį: 2.0 s

Kūnas m2 (Euler) pasiekia max aukštį: 2.0 s

Kūnas m1 (RK) pasiekia max aukštį: 2.444444444444444 s

Kūnas m2 (RK) pasiekia max aukštį: 2.444444444444444 s

Kūnas m1 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

Kūnas m2 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

A graph of different colored lines

Description automatically generated

## Žingsnis-47

Kūnas m1 (Euler) pasiekia max aukštį: 1.9565217391304348 s

Kūnas m2 (Euler) pasiekia max aukštį: 1.9565217391304348 s

Kūnas m1 (RK) pasiekia max aukštį: 2.608695652173913 s

Kūnas m2 (RK) pasiekia max aukštį: 2.608695652173913 s

Kūnas m1 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

Kūnas m2 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

A graph of different colored lines

Description automatically generated

## Žingsnis-80

Kūnas m1 (Euler) pasiekia max aukštį: 2.278481012658228 s

Kūnas m2 (Euler) pasiekia max aukštį: 2.278481012658228 s

Kūnas m1 (RK) pasiekia max aukštį: 2.5316455696202533 s

Kūnas m2 (RK) pasiekia max aukštį: 2.5316455696202533 s

Kūnas m1 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

Kūnas m2 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

A graph of different colored lines

Description automatically generated

## Žingsnis-100

Kūnas m1 (Euler) pasiekia max aukštį: 2.4242424242424243 s

Kūnas m2 (Euler) pasiekia max aukštį: 2.4242424242424243 s

Kūnas m1 (RK) pasiekia max aukštį: 2.525252525252525 s

Kūnas m2 (RK) pasiekia max aukštį: 2.525252525252525 s

Kūnas m1 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

Kūnas m2 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

A screenshot of a graph

Description automatically generated

## Žingsnis-300

Kūnas m1 (Euler) pasiekia max aukštį: 2.5418060200668897 s

Kūnas m2 (Euler) pasiekia max aukštį: 2.5418060200668897 s

Kūnas m1 (RK) pasiekia max aukštį: 2.608695652173913 s

Kūnas m2 (RK) pasiekia max aukštį: 2.608695652173913 s

Kūnas m1 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

Kūnas m2 (solve\_ivp) pasiekia max aukštį: 2.610441767068273 s

A graph of different colored lines

Description automatically generated

# Programos kodas

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from scipy.integrate import solve\_ivp  # Duotos konstantos  m1 = 0.05 # kg  m2 = 0.3 # kg  v0 = 100 # m/s  ks = 0.01 # kg/m sujungtiems objektams  ts = 3 # s, laikas po kurio objektai atskiriami  k1 = 0.05 # kg/m m1 masėi  k2 = 0.01 # kg/m m2 masėi  tmax = 10 # s, bendras laiko tarpas  # Gravitacija  g = 9.8 # m/s^2  # Diferencialinės lygties funkcija  def diferencialine\_lygtis(t, Y):  v1, v2, h1, h2 = Y  f = np.zeros(4)  if t < ts:  f[0] = f[1] = -g - (ks \* v1\*\*2 \* np.sign(v1)) / (m1 + m2)  else:  f[0] = -g - (k1 \* v1\*\*2 \* np.sign(v1)) / m1  f[1] = -g - (k2 \* v2\*\*2 \* np.sign(v2)) / m2  f[2] = v1  f[3] = v2  return f  # Eulerio metodo funkcija  def Eulerio\_Zingsnis(Y, diferencialine\_lygtis, t, dt):  return Y + diferencialine\_lygtis(t, Y) \* dt  # IV eilės Rungės-Kutos metodo funkcija  def IV\_RK(Y, diferencialine\_lygtis, t, dt):  k1 = diferencialine\_lygtis(t, Y)  k2 = diferencialine\_lygtis(t + dt / 2, Y + dt / 2 \* k1)  k3 = diferencialine\_lygtis(t + dt / 2, Y + dt / 2 \* k2)  k4 = diferencialine\_lygtis(t + dt, Y + dt \* k3)  return Y + dt / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4)  # Integracija naudojant scipy solve\_ivp  sp\_ivp\_sprendinys = solve\_ivp(diferencialine\_lygtis, [0, tmax], [v0, v0, 0, 0], t\_eval=np.linspace(0, tmax, 250), method='RK45')  # Pradinės sąlygos ir parametrai  Y0 = [v0, v0, 0, 0]  NSteps = 300  t\_eval = np.linspace(0, tmax, NSteps)  dt = t\_eval[1] - t\_eval[0]  Y\_euler = np.zeros((4, NSteps))  Y\_rk = np.zeros((4, NSteps))  Y\_euler[:, 0] = Y0  Y\_rk[:, 0] = Y0  # Integracijos metodų taikymas  for i in range(NSteps - 1):  Y\_euler[:, i + 1] = Eulerio\_Zingsnis(Y\_euler[:, i], diferencialine\_lygtis, t\_eval[i], dt)  Y\_rk[:, i + 1] = IV\_RK(Y\_rk[:, i], diferencialine\_lygtis, t\_eval[i], dt)  # Randame laiką, kada aukštis didžiausias  def rasti\_didziausio\_aukscio\_laika(greiciai, laikai):  didziausio\_aukscio\_indeksai = np.where(np.diff(np.sign(greiciai)))[0]  if len(didziausio\_aukscio\_indeksai) > 0:  return laikai[didziausio\_aukscio\_indeksai[0]]  else:  return None  # Taikome funkciją kiekvienam metodui  m1\_max\_aukstis\_laikas\_euler = rasti\_didziausio\_aukscio\_laika(Y\_euler[0, :], t\_eval)  m2\_max\_aukstis\_laikas\_euler = rasti\_didziausio\_aukscio\_laika(Y\_euler[1, :], t\_eval)  print(f"Kūnas m1 (Euler) pasiekia max aukštį: {m1\_max\_aukstis\_laikas\_euler} s")  print(f"Kūnas m2 (Euler) pasiekia max aukštį: {m2\_max\_aukstis\_laikas\_euler} s")  m1\_max\_aukstis\_laikas\_rk = rasti\_didziausio\_aukscio\_laika(Y\_rk[0, :], t\_eval)  m2\_max\_aukstis\_laikas\_rk = rasti\_didziausio\_aukscio\_laika(Y\_rk[1, :], t\_eval)  print(f"Kūnas m1 (RK) pasiekia max aukštį: {m1\_max\_aukstis\_laikas\_rk} s")  print(f"Kūnas m2 (RK) pasiekia max aukštį: {m2\_max\_aukstis\_laikas\_rk} s")  m1\_max\_aukstis\_laikas\_solve\_ivp = rasti\_didziausio\_aukscio\_laika(sp\_ivp\_sprendinys.y[0, :], sp\_ivp\_sprendinys.t)  m2\_max\_aukstis\_laikas\_solve\_ivp = rasti\_didziausio\_aukscio\_laika(sp\_ivp\_sprendinys.y[1, :], sp\_ivp\_sprendinys.t)  print(f"Kūnas m1 (solve\_ivp) pasiekia max aukštį: {m1\_max\_aukstis\_laikas\_solve\_ivp} s")  print(f"Kūnas m2 (solve\_ivp) pasiekia max aukštį: {m2\_max\_aukstis\_laikas\_solve\_ivp} s")  # Grafikų braižymas  plt.figure(figsize=(12, 10))  # Greičio grafikai  plt.subplot(2, 1, 1)  plt.plot(t\_eval, Y\_euler[0], label='Euler m1 greitis')  plt.plot(t\_eval, Y\_euler[1], label='Euler m2 greitis')  plt.plot(sp\_ivp\_sprendinys.t, sp\_ivp\_sprendinys.y[0], label='solve\_ivp m1 greitis')  plt.plot(sp\_ivp\_sprendinys.t, sp\_ivp\_sprendinys.y[1], label='solve\_ivp m2 greitis')  plt.plot(t\_eval, Y\_rk[0], label='RK m1 greitis')  plt.plot(t\_eval, Y\_rk[1], label='RK m2 greitis')  plt.xlabel('Laikas (s)')  plt.ylabel('Greitis (m/s)')  plt.title('Objektų Greitis Laikui Bėgant')  plt.legend()  # Aukščio grafikai  plt.subplot(2, 1, 2)  plt.plot(t\_eval, Y\_euler[2], label='Euler m1 aukštis')  plt.plot(t\_eval, Y\_euler[3], label='Euler m2 aukštis')  plt.plot(sp\_ivp\_sprendinys.t, sp\_ivp\_sprendinys.y[2], label='solve\_ivp m1 aukštis')  plt.plot(sp\_ivp\_sprendinys.t, sp\_ivp\_sprendinys.y[3], label='solve\_ivp m2 aukštis')  plt.plot(t\_eval, Y\_rk[2], label='RK m1 aukštis')  plt.plot(t\_eval, Y\_rk[3], label='RK m2 aukštis')  plt.xlabel('Laikas (s)')  plt.ylabel('Aukštis (m)')  plt.title('Objektų Aukštis Laikui Bėgant')  plt.legend()  plt.tight\_layout()  plt.show() |