KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Modulio P175B015 „Skaitiniai metodai ir algoritmai “

Laboratorinio darbo ataskaita

Ketvirtas laboratorinis darbas

Dėstytojas
Doc. Andrius Kriščiūnas
Doc. Dalia Čalnerytė

Studentas
Rokas Puzonas IF-1/1

KAUNAS, 2023

Turinys

1. Diferencialinės lygties sudarymas

m_1 masės parašiutininkas su m_2 masės įranga iššoka iš lėktuvo, kuris skrenda aukštyje h_0. Po t_g laisvo kritimo parašiutas išskleidžiamas. Oro pasipriešinimo koeficientas laisvo kritimo metu lygus k_1, o išskleidus parašiutą - k_2. Tariama, kad paliekant lėktuvą parašiutininko greitis lygus 0 m/s, o oro pasipriešinimas proporcingas parašiutininko greičio kvadratui. Raskite, kaip kinta parašiutininko greitis nuo 0 s iki nusileidimo. Kada ir kokiu greičiu parašiutininkas pasiekia žemę? Kokiame aukštyje išskleidžiamas parašiutas?

Varianto numeris	m_1, kg	m_2, kg	v_0, m	t_g, s	k_1, kg/m	k_2, kg/m
20	120	15	2800	35	0.15	10

If $t < t_g$:

$$\begin{cases} \dfrac{dh}{\mathrm{dt}} = v \\ (\mathrm{m}_1 + \mathrm{m}_2)\dfrac{dv}{\mathrm{dt}} = -(\mathrm{m}_1 + \mathrm{m}_2)g - k_1 v^2 sign(v) \end{cases}$$

$$\frac{d}{\mathrm{dt}}\begin{Bmatrix} h \\ v \end{Bmatrix} = \begin{Bmatrix} v \\ -g - \dfrac{k_1 v^2 sign(v)}{\mathrm{m}_1 + \mathrm{m}_2} \end{Bmatrix}, \quad \begin{Bmatrix} h \\ v \end{Bmatrix}\Bigg|_{t=0} = \begin{Bmatrix} 2800 \\ 0 \end{Bmatrix}$$

Else:

$$\begin{cases} \dfrac{dh}{\mathrm{dt}} = v \\ (\mathrm{m}_1 + \mathrm{m}_2)\dfrac{dv}{\mathrm{dt}} = -(\mathrm{m}_1 + \mathrm{m}_2)g - k_2 v^2 sign(v) \end{cases}$$

$$\frac{d}{\mathrm{dt}}\begin{Bmatrix} h \\ v \end{Bmatrix} = \begin{Bmatrix} v \\ -g - \dfrac{k_2 v^2 sign(v)}{\mathrm{m}_1 + \mathrm{m}_2} \end{Bmatrix}, \quad \begin{Bmatrix} h \\ v \end{Bmatrix}\Bigg|_{t=0} = \begin{Bmatrix} 2800 \\ 0 \end{Bmatrix}$$

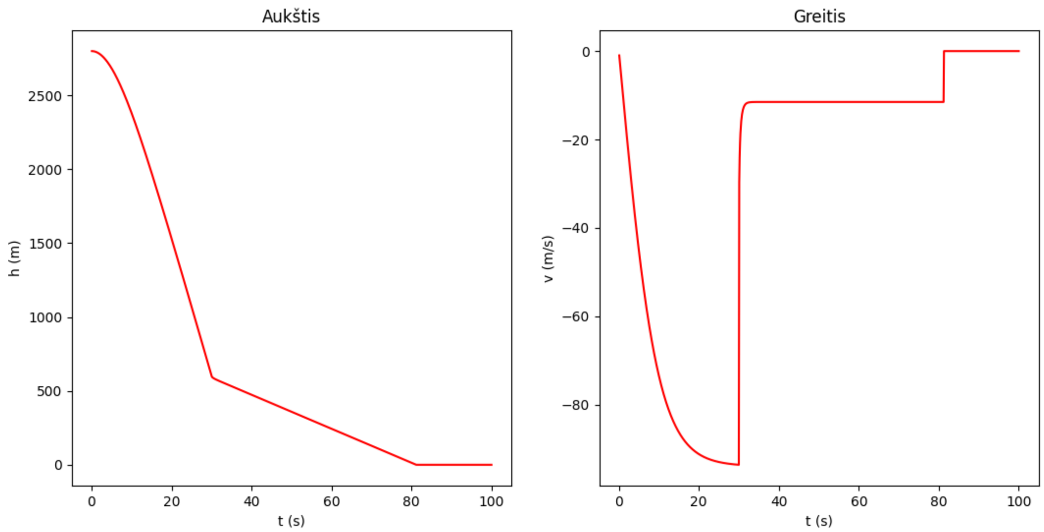
$$if\ h \leq 0: h = 0, v = 0$$

2. Diferencialinės lygties sprendimas

a. Spredimas naudojant eulerio metodą

Parametrai:
Žignsio dydis: 0.005

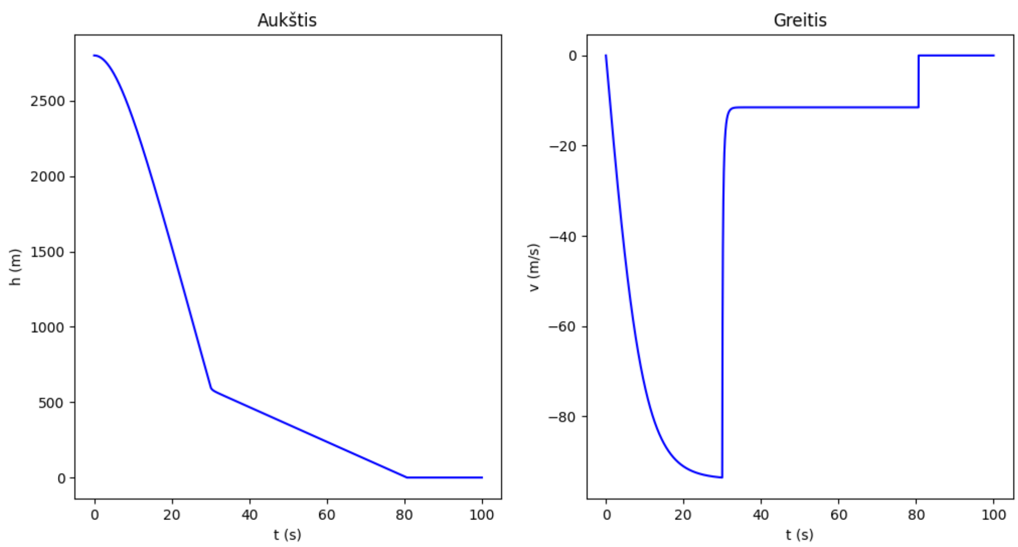
Rezultatai:
Greitis: 11.5080m/s
Laiko momentas: 80.62s



b. Sprendimas naudojant IV eilės Rungės ir Kutos metodą

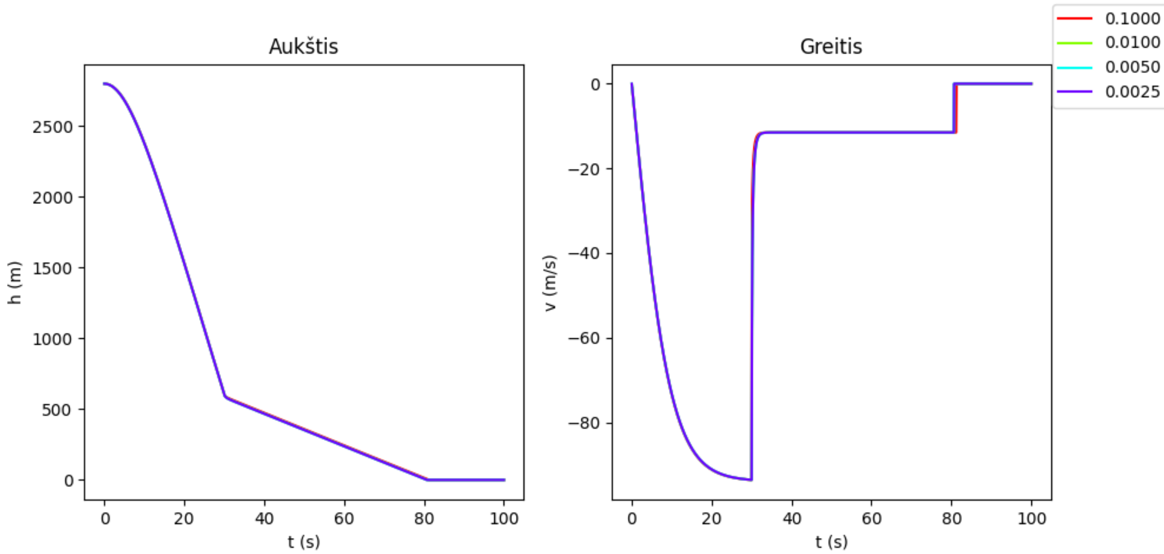
Parametrai:
Žignsio dydis: 0.005

Rezultatai:
Greitis: 11.5080m/s
Laiko momentas: 80.62s

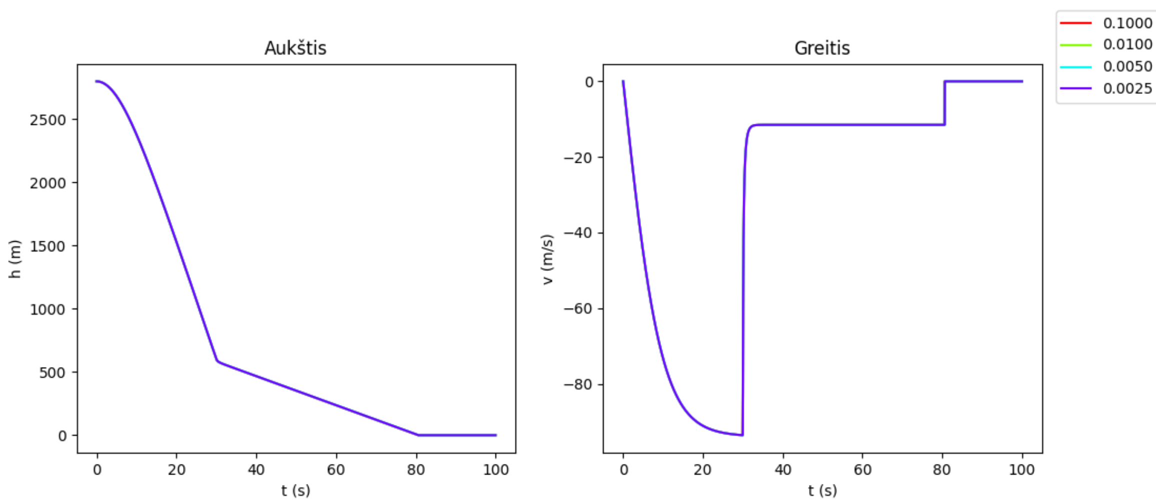


3. Žingsnio dydžio keitimas

Eulerio metodo grafikas su įvairiai žingsnio dydžiais.



Rungės ir Kutos metodo grafikas su įvairiai žingsnio dydžiais.

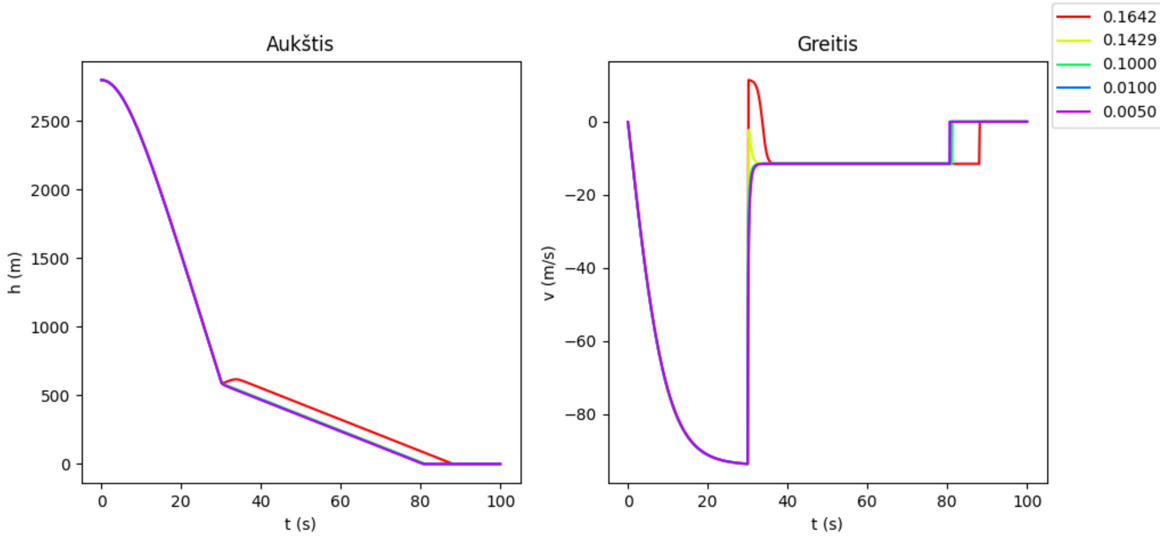


Įsitikinta, kad parinktas pakankamai mažas žingsnio dydis 0.005, nes naudojant Eulerio ar Rungės ir Kutos metodą gauname atsakymus tarp abiejų metodų kurie sutampa 3 skaičiais po kablelio tikslumu.

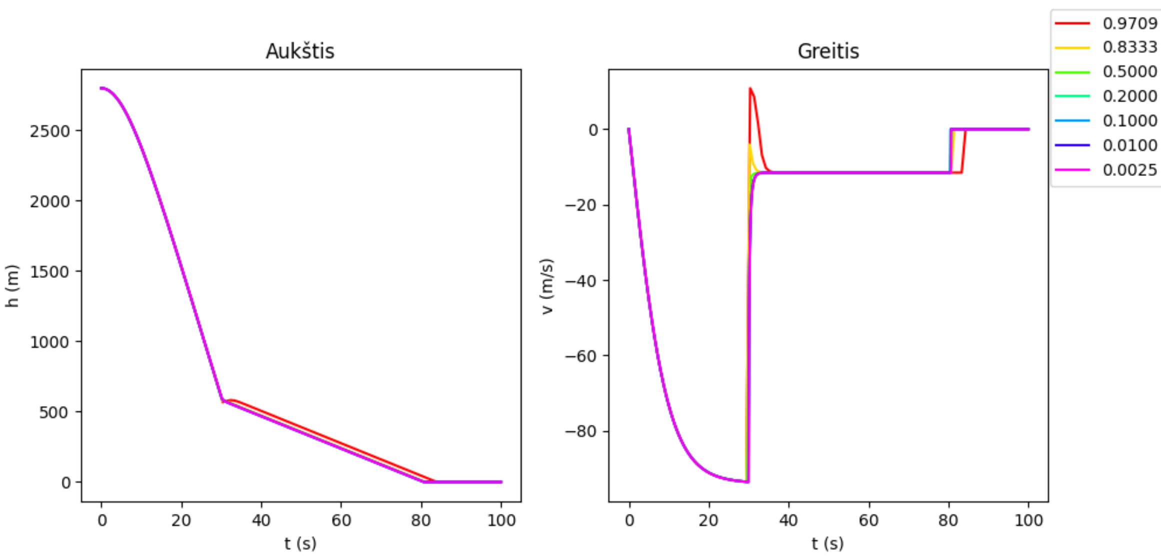
Matome, kad dauguma grafikų sutampa, kad sunku atskirti juos grafike.

4. Didžiausio žingsnio dydžio nustatymas

Naudojant Eulerio metodą didžiausias žigsnio dydis yra ~0.1642.

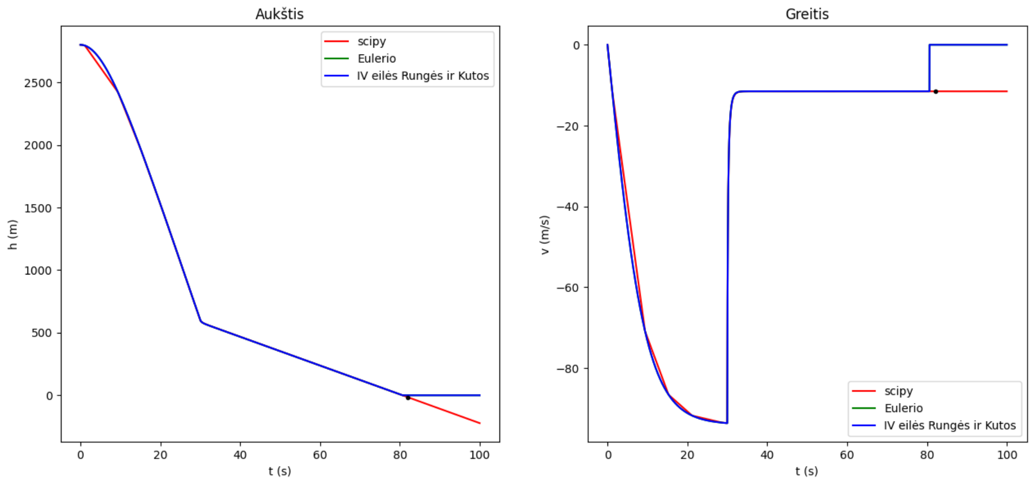


Naudojant Rungės ir Kutos metodą didžiausias žigsnio dydis yra ~0.9709.



5. Sprendimo patikrinimas

Naudotas žingsnio dydis yra: 0.005



6. Kodas

```
from typing import Literal
import numpy as np
import matplotlib.pyplot as plt
from dataclasses import dataclass
import scipy.integrate

@dataclass
class Salyga:
    m1: float # Parašiutininko masė
    m2: float # Įrangos masė
    h0: float # Iššokimo aukštis
    tg: float # Laikas iki parašiuto išskleidimo
    k1: float # Oro pasipriešinimas be parašiuto
    k2: float # Oro pasipriešinimas su parašiutu
    g = 9.81

def ispresti_euleriu(salyga: Salyga, iteracijos: float, simuliacijos_laikotarpis: float):
    t_history = []
    h_history = []
    v_history = []

    m = salyga.m1 + salyga.m2
    dt = simuliacijos_laikotarpis / iteracijos
    h = salyga.h0
    t = 0
    v = 0
    for _ in range(iteracijos):
        k = salyga.k1 if t < salyga.tg else salyga.k2
        pagreitis = salyga.g - k * v ** 2 / m

        h += v * dt
        v += -pagreitis * dt

        if h <= 0:
            h = 0
            v = 0

        t_history.append(t)
        h_history.append(h)
        v_history.append(v)

        t += dt
```

```python
    return t_history, h_history, v_history

def ispresti_rk4(salyga: Salyga, iteracijos: int, simuliacijos_laikotarpis: float):
    def funk(X, t):
        nonlocal salyga
        k = salyga.k1 if t < salyga.tg else salyga.k2
        v = X[1]
        m = salyga.m1 + salyga.m2
        pagreitis = -salyga.g + k * v ** 2 / m

        return np.array([v, pagreitis])

    t = np.linspace(0, simuliacijos_laikotarpis, iteracijos)
    dt = t[1]-t[0]
    rez = np.zeros([2, iteracijos], dtype=float)
    rez[:,0] = np.array([salyga.h0, 0])

    for i in range(iteracijos-1):
        fz   = rez[:,i] + funk(rez[:,i],t[i]     ) * dt/2
        fzz  = rez[:,i] + funk(fz      ,t[i]+dt/2) * dt/2
        fzzz = rez[:,i] + funk(fzz     ,t[i]+dt/2) * dt

        rez[:,i+1] = rez[:,i] + dt/6 * (
                funk(rez[:,i],t[i]     ) +
            2 * funk(fz      ,t[i]+dt/2) +
            2 * funk(fzz     ,t[i]+dt/2) +
                funk(fzzz    ,t[i]+dt  )
        )

        if rez[0,i+1] <= 0:
            rez[0,i+1] = 0
            rez[1,i+1] = 0

    return t, rez[0,:], rez[1,:]

def main_1(salyga: Salyga, iteracijos: int, simuliacijos_laikotarpis):
    t_history_e  , h_history_e  , v_history_e   = ispresti_euleriu(salyga, iteracijos, simuliacijos_laikotarpis)
    t_history_rk4, h_history_rk4, v_history_rk4 = ispresti_rk4(salyga, iteracijos, simuliacijos_laikotarpis)

    print("Žingsnio dydis: ", simuliacijos_laikotarpis / iteracijos)

    for i, h in enumerate(h_history_e):
        if h == 0:
            print("Eulerio", v_history_e[i-1], t_history_e[i-1])
            break

    for i, h in enumerate(h_history_rk4):
        if h == 0:
            print("rk4", v_history_rk4[i-1], t_history_rk4[i-1])
            break

    fig1=plt.figure(1)

    ax1 = fig1.add_subplot(1,2,1)
    #ax1.plot(t_history_e, h_history_e, 'r-', label="Eulerio")
    ax1.plot(t_history_rk4, h_history_rk4, 'b-', label="IV eilės Rungės ir Kutos")
    #ax1.legend()
    ax1.set_xlabel("t (s)")
    ax1.set_ylabel("h (m)")
    ax1.set_title("Aukštis")

    ax2 = fig1.add_subplot(1,2,2)
    #ax2.plot(t_history_e, v_history_e, 'r-', label="Eulerio")
    ax2.plot(t_history_rk4, v_history_rk4, 'b-', label="IV eilės Rungės ir Kutos")
    #ax2.legend()
    ax2.set_xlabel("t (s)")
    ax2.set_ylabel("v (m/s)")
    ax2.set_title("Greitis")

    plt.show()

def main_2(salyga: Salyga, metodas: Literal["euler", "rk4"], iteracijos: list[int], simuliacijos_laikotarpis):
    fig1=plt.figure(1)

    ax1 = fig1.add_subplot(1,2,1)
    ax1.set_xlabel("t (s)")
    ax1.set_ylabel("h (m)")
    ax1.set_title("Aukštis")

    ax2 = fig1.add_subplot(1,2,2)
    ax2.set_xlabel("t (s)")
    ax2.set_ylabel("v (m/s)")
    ax2.set_title("Greitis")

    cmap = plt.cm.get_cmap('hsv', len(iteracijos)+1)
    for i, iteraciju_kiekis in enumerate(iteracijos):
```

```python
        if metodas == "euler":
            t_history, h_history, v_history = ispresti_euleriu(salyga, iteraciju_kiekis, simuliacijos_laikotarpis)
        elif metodas == "rk4":
            t_history, h_history, v_history = ispresti_rk4(salyga, iteraciju_kiekis, simuliacijos_laikotarpis)
        zingsnis = simuliacijos_laikotarpis / iteraciju_kiekis
        ax1.plot(t_history, h_history, c=cmap(i))
        ax2.plot(t_history, v_history, c=cmap(i), label=f"{zingsnis:.4f}")

    fig1.legend()
    plt.show()

def main_3(salyga: Salyga, iteracijos, simuliacijos_laikotarpis: float):
    print("Žingsnio dydis: ", simuliacijos_laikotarpis / iteracijos)

    t_history_e  , h_history_e  , v_history_e   = ispresti_euleriu(salyga, iteracijos, simuliacijos_laikotarpis)
    t_history_rk4, h_history_rk4, v_history_rk4 = ispresti_rk4(salyga, iteracijos, simuliacijos_laikotarpis)

    def funk(t, X):
        nonlocal salyga
        k = salyga.k1 if t < salyga.tg else salyga.k2
        v = X[1]
        m = salyga.m1 + salyga.m2
        pagreitis = -salyga.g + k * v ** 2 / m

        return np.array([v, pagreitis])

    tspan = np.array([0, simuliacijos_laikotarpis])
    Y = scipy.integrate.solve_ivp(funk, tspan, [salyga.h0, 0])

    fig1=plt.figure(1)

    zero_point = 0
    for i, h in enumerate(Y.y[0,:]):
        if h <= 0:
            zero_point = i
            break

    ax1 = fig1.add_subplot(1,2,1)
    ax1.set_xlabel("t (s)")
    ax1.set_ylabel("h (m)")
    ax1.set_title("Aukštis")
    ax1.plot(Y.t, Y.y[0,:], color="r", label="scipy")
    ax1.plot(t_history_e, h_history_e, color="g", label="Eulerio")
    ax1.plot(t_history_rk4, h_history_rk4, color="b", label="IV eilės Rungės ir Kutos")
    ax1.plot(Y.t[zero_point], Y.y[0, zero_point], 'k.')
    ax1.legend()

    ax2 = fig1.add_subplot(1,2,2)
    ax2.set_xlabel("t (s)")
    ax2.set_ylabel("v (m/s)")
    ax2.set_title("Greitis")
    ax2.plot(Y.t, Y.y[1,:], color="r", label="scipy")
    ax2.plot(t_history_e, v_history_e, color="g", label="Eulerio")
    ax2.plot(t_history_rk4, v_history_rk4, color="b", label="IV eilės Rungės ir Kutos")
    ax2.plot(Y.t[zero_point], Y.y[1,zero_point], 'k.')
    ax2.legend()

    plt.show()

# Variantas 20
salyga = Salyga(
    m1 = 120.0,
    m2 = 15.0,
    h0 = 2800.0,
    tg = 30.0,
    k1 = 0.15,
    k2 = 10.0
)

main_1(
    salyga,
    iteracijos = 20000,
    simuliacijos_laikotarpis = 100
)

main_2(
    salyga,

    # Žemi žingsniai
    # metodas="rk4",
    # iteracijos = [1000, 10000, 20000, 40000],
    # metodas="euler",
    # iteracijos = [1000, 10000, 20000, 40000],

    # Aukšti žingsniai
    metodas="rk4",
    iteracijos = [103, 120, 200, 500, 1000, 10000, 40000],
```

```
    # metodas="euler",
    # iteracijos = [609, 700, 1000, 10000, 20000],

    simuliacijos_laikotarpis = 100
)

main_3(
    salyga,
    iteracijos = 20000,
    simuliacijos_laikotarpis = 100
)
```