KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

SKAITINIAI METODAI IR ALGORITMAI(P170B115)

4 LABORATORINIS DARBAS

Varianto Nr. 15

Atliko:

IFF-1/8 gr. studentas

Matas Palujanskas

Priėmė:

Prof. Rimantas Barauskas

Doc. Andrius Kriščiūnas

KAUNAS

2023

1. Turinys

Matas Palujanskas

1. Užduotis

1.1 Užduoties tikslas

Remdamiesi pateiktų fizikinių dėsnių aprašymais, gautajam **15**-ąjam variantui sudaryti diferencialinę lygtį arba lygčių sistemą bei ją paaiškinti. Diferencialinę lygtį arba lygčių sistemą įspręsti skaitinių metodų pagalba tai yra Eulerio ir IV eilės Rungės ir Kutos metodais. Suprasti kaip žingsnis gali daryti įtaką uždavinio rezultatų tikslumui. Palyginti metodų tikslumo prasmę bei patikrinti gautą rezultatą su MATLAB standartine funkcija ode45 arba kitais išoriniais šaltiniais.

1.2 Užduoties sąlyga

4 Uždavinys variantams 11-20

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?

2 Lentelė. Uždavinyje naudojami dydžiai.

Varianto numeris	m_1, kg	m_2, kg	h_0, m	t_g, s	k_1, kg/m	k_2, kg/m
11	100	15	3000	40	0,5	10
12	70	15	4000	40	0,1	5
13	50	15	3500	35	0,1	7
14	125	25	2000	20	0,5	10
15	120	10	2800	25	0,25	10
16	90	15	3500	40	0,5	3
17	85	10	2500	35	0,2	10
18	60	15	3500	25	0,1	7
19	75	10	2200	30	0,3	10
20	120	15	2800	35	0,15	10

pav. 1 Užduoties sąlyga

15	120	10	2800	25	0,25	10

pav. 2 15 varianto duomenys

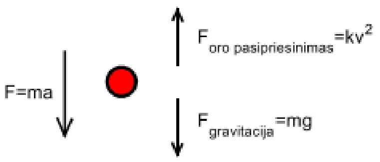
Matas Palujanskas

2. Užduoties sprendimas

2.1. Teorinė dalis

Pagal užduoties sąlygą kūnas juda su pagreičiu, todėl remsiuosi antruoju Niutono dėsniu, kuris teigia, kad pagreitis $\vec{a}$, kuriuo juda kūnas yra tiesiogiai proporcingas kūną veikiančiai jėgai $\vec{F}$ A ir atvirkščiai proporcingas to kūno masei.

$$\vec{F}\text{A} = m\vec{a}$$



pav. 3 Kūną veikiančių jėgų schema

Taipogi žinau, kad greitis yra pirmoji kelio funkcijos s(t) išvestinė, kaip pagreitis yra pirmoji greičio funkcijos išvestinė. Todėl gauname:

$$\frac{ds}{dt} = v, \frac{d^2s}{dt^2} = \frac{dv}{dt} = a$$

Remdamasis šiuo dėsniu susidarau lygtį, kuri rodo parašiutininką veikiančia gravitaciją bei oro pasipriešinimo jėgą.

$$-m * g + F = m * a$$

Visas kūnas kurį veikia jėga yra m1+m2 (parašiutininko masė bei pačio parašiuto), kadangi pasipriešinimo koeficientą turiu, galiu išsireikšti lygtį.

$$a = \frac{dv}{dt} = \frac{-(m1 + m2) * g - k * v * |v|}{m1 + m2}$$

Pasipriešinimo koeficientas po parašiuto išskleidimo pakis, todėl pradžioje jis yra **0.25**, o po parašiuto išskleidimo jis tampa **10**.

2.2. Sprendimas Eulerio metodu

2.2.1 Kodas

Eulerio.ipynb:

```
import numpy as np
import matplotlib.pyplot as plt

def parašiutininko_kritimas(X, t):
    h, v = X
    m1 = 120  # parašiutininko masė (kg)
    m2 = 10   # įrangos masė (kg)
    k1 = 0.25  # oro pasipriešinimo koeficientas laisvo kritimo metu (kg/m)
    k2 = 10    # oro pasipriešinimo koeficientas išskleidus parašiutą (kg/m)
    g = 9.81   # gravitacijos pagreitis (m/s^2)

    if t < 25:
        # Laisvas kritimas iki parašiuto išsiskleidimo
        F = (m1 + m2) * g - k1 * v**2 * np.sign(v)
    else:
        # Parašiutas išsiskleidęs
        F = (m1 + m2) * g - k2 * v**2 * np.sign(v)

    dhdt = -v
    dvdt = F / (m1 + m2)

    return np.array([dhdt, dvdt])

# Laiko nustatymai
viso_laikas = 146  # Visas skaičiavimo laikas (s)
dt = 0.1  # Laiko žingsnis Eulerio metodu

# Pradinės sąlygos
h0 = 2800  # Pradinis aukštis (m)
v0 = 0     # Pradinis greitis (m/s)

# Rezultatų masyvas
N = int(viso_laikas / dt) + 1
t = np.linspace(0, viso_laikas, N)
rezultatas = np.zeros([2, N])
rezultatas[:, 0] = np.array([h0, v0])

# Skaičiavimas naudojant Eulerio metodą
for i in range(N - 1):
    išvestinė = parašiutininko_kritimas(rezultatas[:, i], t[i])
    rezultatas[:, i + 1] = rezultatas[:, i] + išvestinė * dt


# Analizuojame rezultatus
pasiekimo_momento_indeksas = np.argmax(rezultatas[0] <= 0)  # Indeksas, kai aukštis
tampa neigiamas
pasiekimo_laikas = t[pasiekimo_momento_indeksas]
pasiekimo_greitis = rezultatas[1, pasiekimo_momento_indeksas]

išskleidimo_indeksas = np.argmax((t >= 25) & (rezultatas[0] > 0))  # Indeksas, kai
parašiutas išsiskleidžia

išskleidimo_laikas = t[išskleidimo_indeksas]
išskleidimo_aukštis = rezultatas[0, išskleidimo_indeksas]
```

```
# Spausdiname rezultatus
print(f"Parašiutininkas pasiekia žemę laiko t = {pasiekimo_laikas:.2f} s metu su greičiu {pasiekimo_greitis:.2f} m/s.")
print(f"Parašiutas išsiskleidžiamas laiko t = {išskleidimo_laikas:.2f} s metu, esant aukštyje {išskleidimo_aukštis:.2f} m.")

# Braižome rezultatus
fig, ax = plt.subplots()
ax.plot(t, rezultatas[0, :], label='Aukštis')
ax.set_xlabel('Laikas (s)')
ax.set_ylabel('Aukštis (m)')
ax.legend()
plt.show()
```

2.2.2 Testavimas

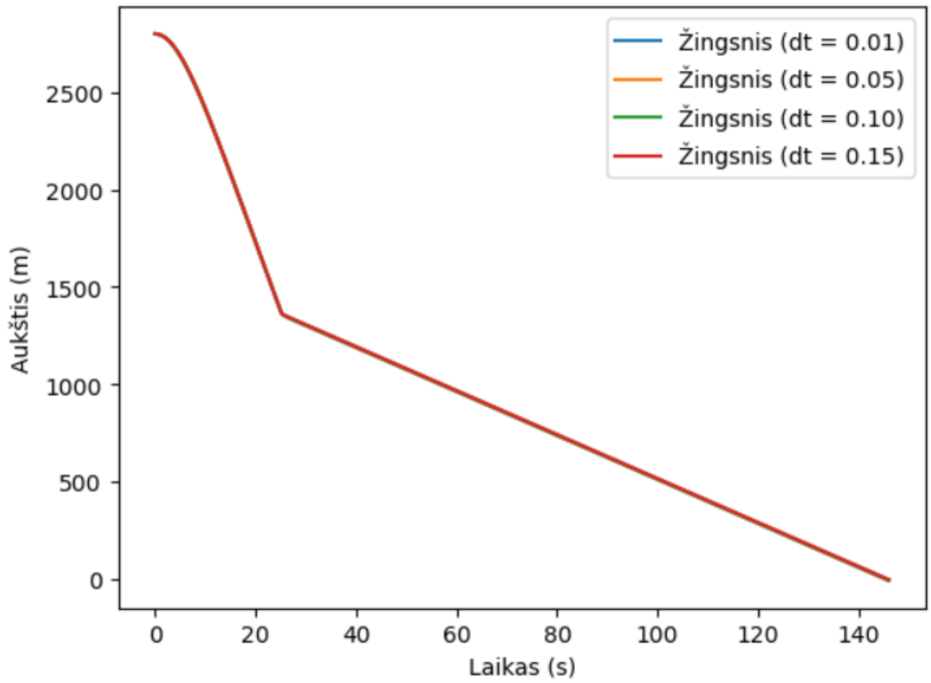
Sprendimas Eulerio metodu bus testuojamas 4 kartus su skirtingais žingsniais

Nr.	Žingsnis	Kada parašiutininkas pasiekia žemę?(s)	Kokiu greičiu jis pasiekia žemę?(m/s)	Kokiame aukštyje išskleidžiamas parašiutas?(m)
1	0.01	145.24	11.29	1374.47
2	0.05	145.35	11.29	1375.04
3	0.1	145.60	11.29	1375.76
4	0.15	145.85	11.29	1372.92

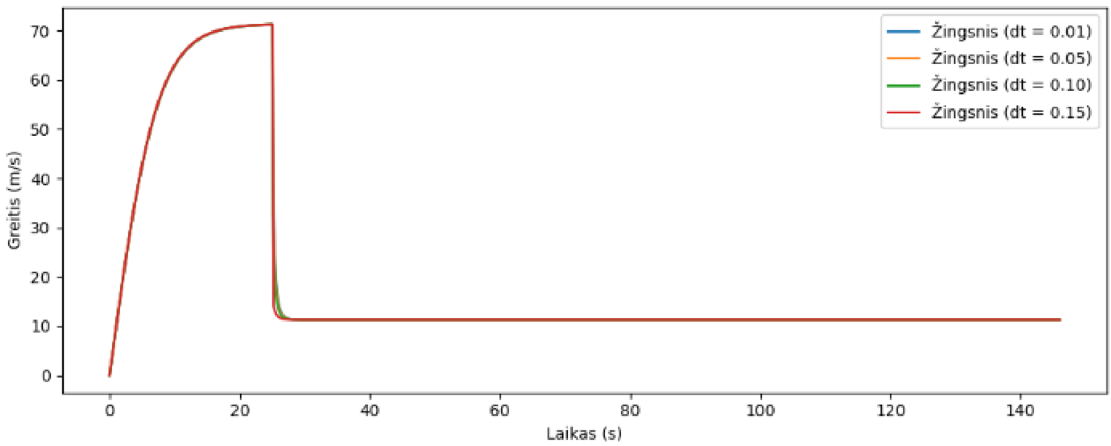
```
Su dt = 0.01:
Parašiutininkas paliečia žemę laiko t = 145.24 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.00 s metu, esant aukštyje 1374.47 m.

Su dt = 0.05:
Parašiutininkas paliečia žemę laiko t = 145.35 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.00 s metu, esant aukštyje 1375.04 m.

Su dt = 0.10:
Parašiutininkas paliečia žemę laiko t = 145.60 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.00 s metu, esant aukštyje 1375.76 m.

Su dt = 0.15:
Parašiutininkas paliečia žemę laiko t = 145.85 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.06 s metu, esant aukštyje 1372.92 m.
```

pav. 4 Gauti programos rezultatai

2.2.3 Grafikai



pav. 5 Eulerio aukščio ir laiko priklausomybė



pav. 6 Eulerio greičio ir laiko priklausomybė

Išvada: keičiant žingsnio dydį grafikas ir rezultatai skiriasi nežymiai, grafikai perdengia vienas kitą.

2.3. Sprendimas IV eilės Rungės ir Kutos metodus

2.3.1 Kodas

IV RK.ipynb:

```
import numpy as np
import matplotlib.pyplot as plt

def parašiutininko_kritimas(X, t):
    h, v = X
```

```
    m1 = 120   # parašiutininko masė (kg)
    m2 = 10    # įrangos masė (kg)
    k1 = 0.25  # oro pasipriešinimo koeficientas laisvo kritimo metu (kg/m)
    k2 = 10    # oro pasipriešinimo koeficientas išskleidus parašiutą (kg/m)
    g = 9.81   # gravitacijos pagreitis (m/s^2)

    if t < 25:
        # Laisvas kritimas iki parašiuto išsiskleidimo
        F = (m1 + m2) * g - k1 * v**2 * np.sign(v)
    else:
        # Parašiutas išsiskleidęs
        F = (m1 + m2) * g - k2 * v**2 * np.sign(v)

    dhdt = -v
    dvdt = F / (m1 + m2)

    return np.array([dhdt, dvdt])

def runge_kutta_4(func, y0, t):
    N = len(t)
    y = np.zeros((len(y0), N))
    y[:, 0] = y0

    for i in range(N - 1):
        h = t[i + 1] - t[i]
        k1 = h * func(y[:, i], t[i])
        k2 = h * func(y[:, i] + k1 / 2, t[i] + h / 2)
        k3 = h * func(y[:, i] + k2 / 2, t[i] + h / 2)
        k4 = h * func(y[:, i] + k3, t[i] + h)
        y[:, i + 1] = y[:, i] + (k1 + 2 * k2 + 2 * k3 + k4) / 6

    return y

# Laiko nustatymai
viso_laikas = 170  # Visas skaičiavimo laikas (s)
dt_values = [0.01, 0.1, 0.5, 1.2]  # Laiko žingsniai Eulerio metodu

# Pradinės sąlygos
h0 = 2800  # Pradinis aukštis (m)
v0 = 0     # Pradinis greitis (m/s)

# Braižome rezultatus su skirtingais laiko žingsniais
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

for dt in dt_values:
    # Laiko vektorius
    t = np.arange(0, viso_laikas, dt)

    # Skaičiavimas naudojant Rungės-Kutos metodą
    rezultatai_rk4 = runge_kutta_4(parašiutininko_kritimas, [h0, v0], t)

    # Analizuojame rezultatus
    pasiekimo_momento_indeksas = np.argmax(rezultatai_rk4[0] <= 0)  # Indeksas, kai aukštis tampa neigiamas
    pasiekimo_laikas = t[pasiekimo_momento_indeksas]
    pasiekimo_greitis = rezultatai_rk4[1, pasiekimo_momento_indeksas]

    išskleidimo_indeksas = np.argmax((t >= 25) & (rezultatai_rk4[0] > 0))  # Indeksas, kai parašiutas išsiskleidžia
    išskleidimo_laikas = t[išskleidimo_indeksas]
    išskleidimo_aukštis = rezultatai_rk4[0, išskleidimo_indeksas]
```

```
    # Braižome rezultatus aukščio grafike
    ax1.plot(t, rezultatai_rk4[0, :], label=f'Aukštis (dt = {dt:.2f})')

    # Braižome rezultatus greičio grafike
    ax2.plot(t, rezultatai_rk4[1, :], label=f'Greitis (dt = {dt:.2f})')

# Plot nustatymai aukščio grafikui
ax1.set_xlabel('Laikas (s)')
ax1.set_ylabel('Aukštis (m)')
ax1.legend()

# Plot nustatymai greičio grafikui
ax2.set_xlabel('Laikas (s)')
ax2.set_ylabel('Greitis (m/s)')
ax2.legend()

plt.tight_layout()
plt.show()
```

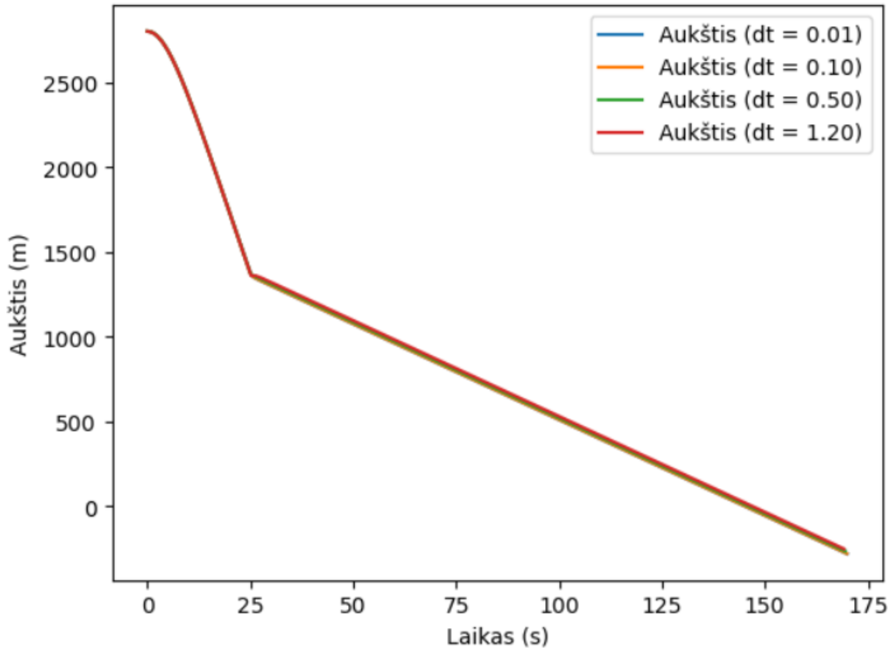
2.3.2 Testavimas

Nr.	Žingsnis	Kada parašiutininkas pasiekia žemę?(s)	Kokiu greičiu jis pasiekia žemę?(m/s)	Kokiame aukštyje išskleidžiamas parašiutas?(m)
1	0.01	145.22	11.29	1374.33
2	0.1	145.30	11.29	1375.33
3	0.5	146.50	11.29	1374.33
4	1.2	147.60	11.29	1360.08

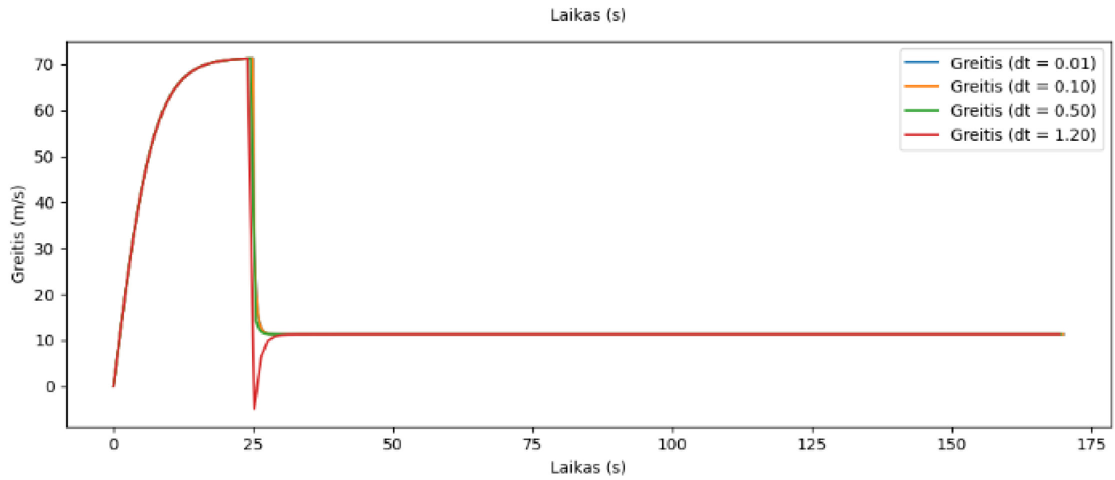
```
Su dt = 0.01:
Parašiutininkas paliečia žemę laiko t = 145.22 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.00 s metu, esant aukštyje 1374.33 m.

Su dt = 0.10:
Parašiutininkas paliečia žemę laiko t = 145.30 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.00 s metu, esant aukštyje 1374.33 m.

Su dt = 0.50:
Parašiutininkas paliečia žemę laiko t = 146.50 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.00 s metu, esant aukštyje 1374.33 m.

Su dt = 1.20:
Parašiutininkas paliečia žemę laiko t = 147.60 s su greičiu 11.29 m/s.
Parašiutas išskleidžiamas laiko t = 25.20 s metu, esant aukštyje 1360.08 m.
```

pav. 7 Programiškai gauti rezultatai

2.3.3 Grafikai



pav. 8 IV eilės Rungės ir Kutos aukščio ir laiko priklausomybės grafikai



pav. 9 IV eilės Rungės ir Kutos greičio ir laiko priklausomybės grafikai

Išvada: keičiant žingsnio dydį grafikas ir rezultatai skiriasi nežymiai, grafikai perdengia vienas kitą. Tačiau kai žingsnis didesnis 1.20 jau matosi nuokrypis.

2.4. Tikrinimas su solve_ivp

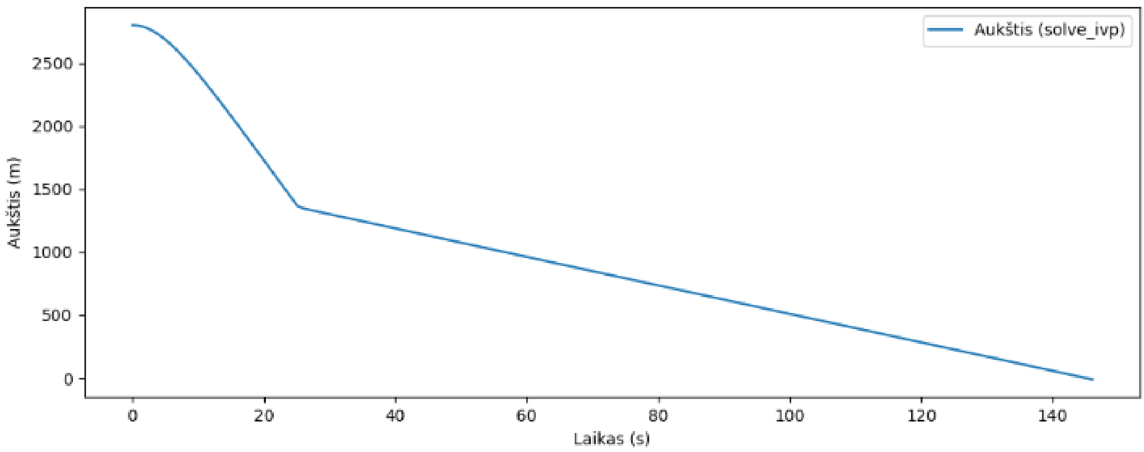
2.4.1 Tikrinimo kodas

Tikrinimas.ipynb:

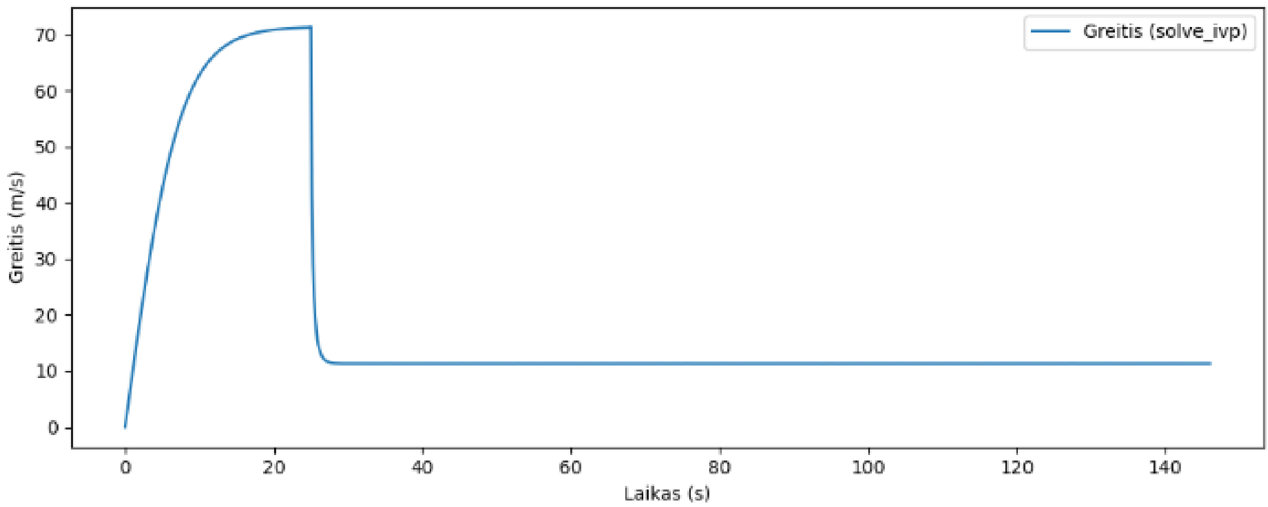
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

def parašiutininko_kritimas(t, X):
    h, v = X
    m1 = 120  # parašiutininko masė (kg)
    m2 = 10   # įrangos masė (kg)
    k1 = 0.25  # oro pasipriešinimo koeficientas laisvo kritimo metu (kg/m)
    k2 = 10    # oro pasipriešinimo koeficientas išskleidus parašiutą (kg/m)
    g = 9.81   # gravitacijos pagreitis (m/s^2)

    if t < 25:
        # Laisvas kritimas iki parašiuto išsiskleidimo
        F = (m1 + m2) * g - k1 * v**2 * np.sign(v)
    else:
        # Parašiutas išsiskleidęs
        F = (m1 + m2) * g - k2 * v**2 * np.sign(v)

    dhdt = -v
    dvdt = F / (m1 + m2)

    return [dhdt, dvdt]

# Laiko nustatymai
viso_laikas = 146  # Visas skaičiavimo laikas (s)

# Pradinės sąlygos
h0 = 2800  # Pradinis aukštis (m)
v0 = 0     # Pradinis greitis (m/s)

# Laiko vektorius
t_span = (0, viso_laikas)

# Skaičiavimas naudojant solve_ivp
rezultatai_ivp = solve_ivp(parašiutininko_kritimas, t_span, [h0, v0],
t_eval=np.linspace(0, viso_laikas, 1000), method='RK45')

# Analizuojame rezultatus
pasiekimo_momento_indeksas = np.argmax(rezultatai_ivp.y[0] <= 0)  # Indeksas, kai
aukštis tampa neigiamas
pasiekimo_laikas = rezultatai_ivp.t[pasiekimo_momento_indeksas]
pasiekimo_greitis = rezultatai_ivp.y[1, pasiekimo_momento_indeksas]

išskleidimo_indeksas = np.argmax((rezultatai_ivp.t >= 25) & (rezultatai_ivp.y[0] >
0))  # Indeksas, kai parašiutas išsiskleidžia
išskleidimo_laikas = rezultatai_ivp.t[išskleidimo_indeksas]
išskleidimo_aukštis = rezultatai_ivp.y[0, išskleidimo_indeksas]

# Spausdiname rezultatus
print(f"Parašiutininkas pasiekia žemę laiko t = {pasiekimo_laikas:.2f} s metu su
greičiu {pasiekimo_greitis:.2f} m/s.")
print(f"Parašiutas išsiskleidžiamas laiko t = {išskleidimo_laikas:.2f} s metu, esant
aukštyje {išskleidimo_aukštis:.2f} m.")
```

```
# Braižome rezultatus
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

# Aukščio grafikas
ax1.plot(rezultatai_ivp.t, rezultatai_ivp.y[0], label='Aukštis (solve_ivp)')
ax1.set_xlabel('Laikas (s)')
ax1.set_ylabel('Aukštis (m)')
ax1.legend()

# Greičio grafikas
ax2.plot(rezultatai_ivp.t, rezultatai_ivp.y[1], label='Greitis (solve_ivp)')
ax2.set_xlabel('Laikas (s)')
ax2.set_ylabel('Greitis (m/s)')
ax2.legend()

plt.tight_layout()
plt.show()
```

2.4.2 Tikrinimo grafikai



pav. 10 Tikrinimas su solve_ivp aukščio ir laiko priklausomybė



pav. 11 Tikrinimas su solve_ivp greičio ir laiko priklausomybė

Išvada: naudojant Pyhton scipy.integrate bibliotekos funkciją solve_ivp buvo įsitikinta, jog buvo gauti teisingi rezultatai.

2.5. Bendros išvados

Išvada: Darant šį darbą buvo įsisavinti paprastųjų diferencialinių lygčių sudarymo ir sprendimo metodai. Buvo prisimintas fizikos kursas. Išnagrinėti Eulerio ir 4 eilės Rungės ir Kutos metodai. Nustatytas pastarojo pranašumas tikslumo atžvilgiu prieš Eulerio metodą. To buvo galima tikėtis, nes skaičiuojant sprendinį 4 eilės Rungės ir Kutos metodu yra daromi papildomi tarpiniai skaičiavimai. Taip pat panaudota Pyhton scipy.integrate bibliotekos funkcija solve_ivp patikrinti sprendinių tikslumą.

3. Literatūros sąrašas

1. „Skaitiniai metodai ir algoritmai“ „Moodle“ aplinkoje HTTPS://MOODLE.KTU.EDU/COURSE/VIEW.PHP?ID=7639