**Kauno technologijos universitetas**
Informatikos fakultetas

# Skaitiniai metodai ir algoritmai (P170B115)

4-osios projektinės užduoties ataskaita

10-as variantas

Atliko:

**IFF-1/6 gr. studentas**

**Lukas Kuzmickas**

2023 m. December 20 d.

Priėmė:

**Prof. Rimantas Barauskas**

**Doc. Andrius Krisčiūnas**

**Kaunas 2023**

TURINYS

# 1. Užduotis

### 3 Uždavinys variantams 1-10

Sujungti $m_1$ ir $m_2$ masių objektai iššaunami vertikaliai į viršų pradiniu greičiu $v_0$. Oro pasipriešinimo koeficientas sujungtiems kūnams lygus $k_s$. Praėjus laikui $t_s$, objektai pradeda judėti atskirai. Oro pasipriešinimo koeficientai atskirai judantiems objektams atitinkamai yra $k_1$ ir $k_2$. Oro pasipriešinimas proporcingas objekto greičio kvadratui. Raskite, kaip kinta objektų greičiai nuo 0 s iki $t_{max}$. Kada kiekvienas objektas pasieks aukščiausią tašką ir pradės leistis?

1 Lentelė. Uždavinyje naudojami dydžiai.

| Varianto numeris | $m_1$, kg | $m_2$, kg | $v_0$, m/s | $k_s$, kg/m | $t_s$, s | $k_1$, kg/m | $k_2$, kg/m | $t_{max}$, s |
|---|---|---|---|---|---|---|---|---|
| 1 | 0,2 | 0,4 | 80 | 0,015 | 1 | 0,02 | 0,005 | 15 |
| 2 | 0,15 | 0,2 | 70 | 0,01 | 2 | 0,05 | 0,001 | 10 |
| 3 | 0,07 | 0,2 | 50 | 0,015 | 3 | 0,05 | 0,01 | 10 |
| 4 | 0,5 | 0,25 | 100 | 0,002 | 2 | 0,02 | 0,04 | 15 |
| 5 | 0,6 | 0,2 | 200 | 0,01 | 2 | 0,02 | 0,015 | 15 |
| 6 | 0,1 | 0,5 | 60 | 0,01 | 1 | 0,01 | 0,005 | 10 |
| 7 | 0,3 | 0,3 | 60 | 0,005 | 2 | 0,05 | 0,01 | 10 |
| 8 | 0,05 | 0,3 | 100 | 0,01 | 3 | 0,05 | 0,01 | 10 |
| 9 | 0,4 | 0,8 | 50 | 0,001 | 2 | 0,02 | 0,02 | 10 |
| 10 | 0,8 | 0,8 | 200 | 0,01 | 2 | 0,02 | 0,005 | 15 |

*1 pav. Mūsų užduoties variantas (10 nr.).*

Pradiniai dydžiai, pagal varianto numerį:

$$m_1 = 0.8kg \;\; m_2 = 0.8kg \;\; v_0 = 200\frac{m}{s} \;\; k_s = 0.01\frac{kg}{m} \;\; t_s = 2s \;\; k_1 = 0.02\frac{kg}{m}$$

$$k_2 = 0.005\frac{kg}{m} \;\; t_{max} = 15s$$

## 2. Lygties sudarymas

Antrasis Niutono dėsnis teigia, kad pagreitis $\vec{a}$, kuriuo juda kūnas yra tiesiogiai proporcingas kūną veikiančiai jėgai $\vec{F}$ ir atvirkščiai proporcingas to kūno masei $m$:

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

Žinome, kad greitis yra pirmoji kelio funkcijos $s(t)$ išvestinė, o pagreitis – pirmoji greičio funkcijos $v(t)$ išvestinė (antroji kelio funkcijos $s(t)$ išvestinė), t.y.:

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

Uždaviniuose naudojamas laisvojo kritimo pagreitį: $g = 9.8\frac{m}{s^2}$

Remiantis Niutono dėsniai sudaroma lygtis:

$$F = -F_{gravitacija} - F_{oro\ pasipriešinimas}$$

$$F_{gravitacija} = mg \textit{ (kitaip vadinama sunkio jėga)}$$

$$F_{oro\ pasipriešinimas} = kv^2 \textit{(oro pasipriešinimo jėga)}$$

*Petvarkome visa lygtį:*

$ma = -mg - kv^2$ (mums nereikia dydžio m)

$$a = -g - \frac{kv^2}{m}$$

Kadangi: $\frac{dv}{dt} = a$, tai:

$$\frac{dv}{dt} = -g - \frac{kv^2}{m}$$

Sudarėme mūsų sprendžiamą diferencialinę lygtį.

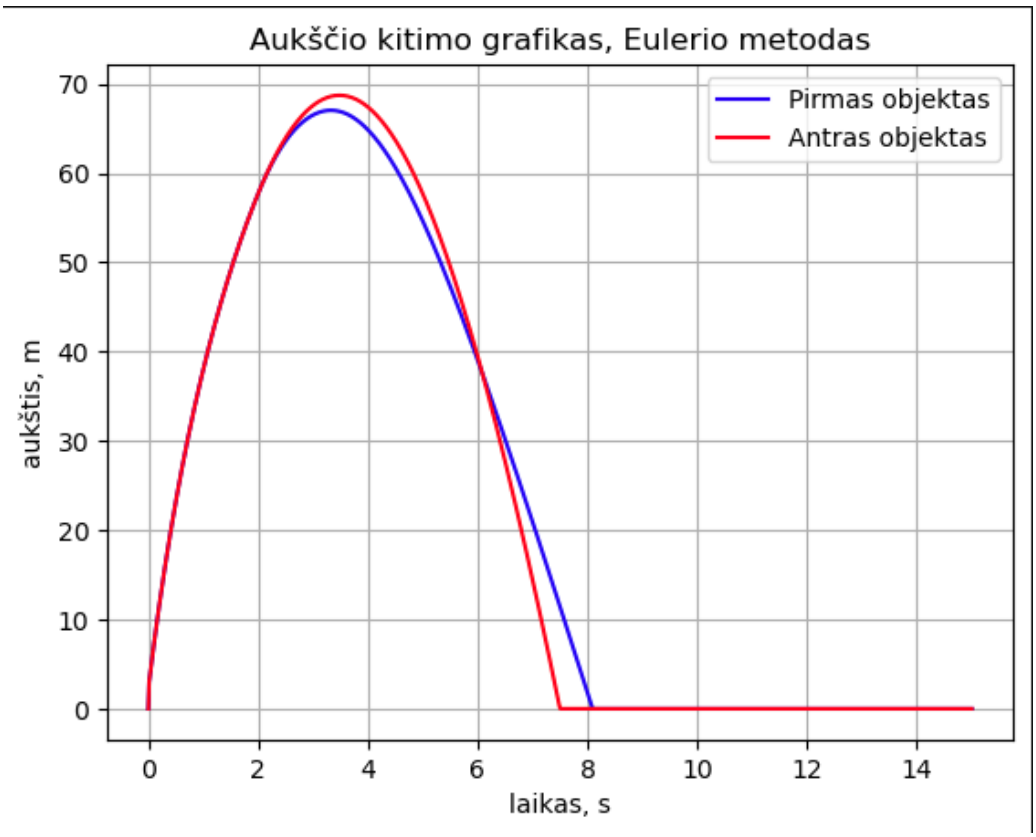
# 3. Lygties sprendimas Eulerio ir IV eilės Rungės ir Kutos metodais

## 3.1. Eulerio metodo realizacija:

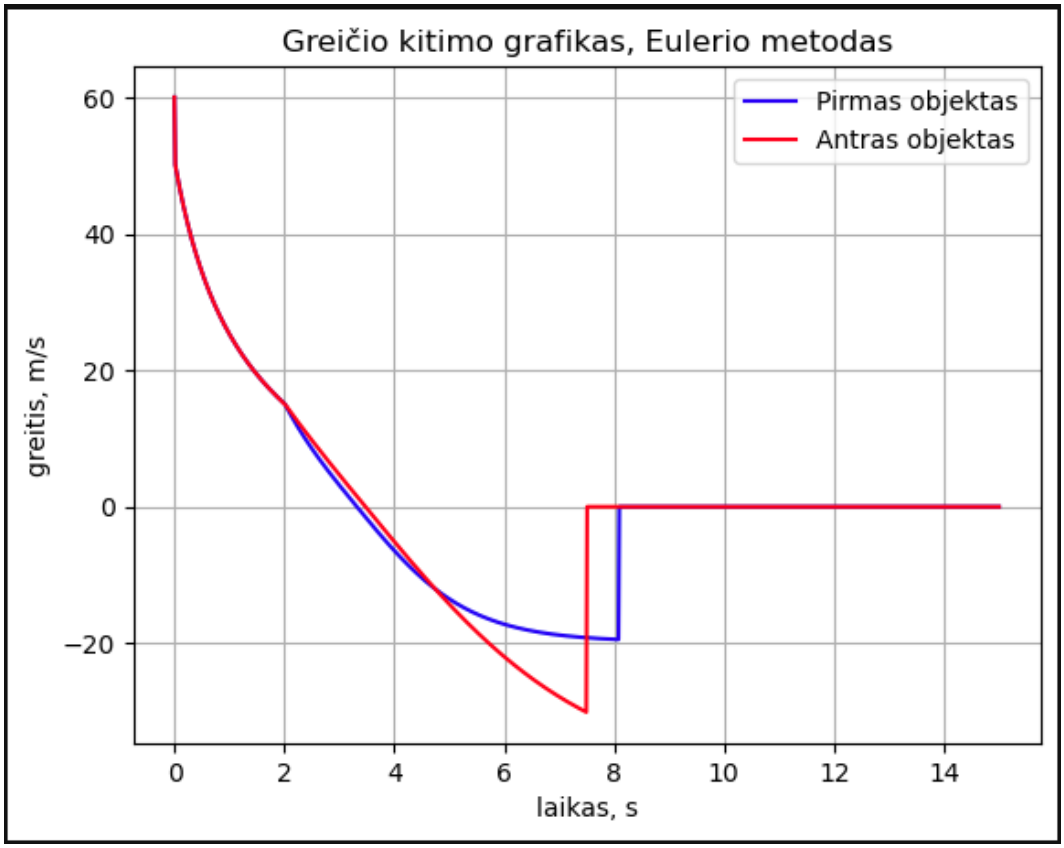
```
for i in range(N - 1):
  rez[:, i + 1] = rez[:, i] + funk(rez[:, i], t[i]) * dt
  if rez[0, i + 1] <= 0 : rez[[0, 2], i + 1] = 0
  if rez[1, i + 1] <= 0 : rez[[1, 3], i + 1] = 0
```

*2 pav. Eulerio metodas.*

Sekantis Eulerio metodo rezultatas apskaičiuojamas iš paskutinio gauto rezultato sudėties su mūsų funkcijos rezultato ir pasirinkto žingsnio sandauga.




*3 pav. Eulerio metodo aukščio kitimo grafikas.*




*4 pav. Eulerio metodo greičio kitimo grafikas.*

Kada kiekvienas objektas pasieks aukščiausią tašką ir pradės leistis?

```
Pirmas objektas maksimaliai pasiekė: 67.03940405742102 m., ir pradėjo leistis laiko momentu: 3.3333333333333335 s.
Antras objektas maksimaliai pasiekė: 68.71991285550911 m., ir pradėjo leistis laiko momentu: 3.4834834834834836 s.
```

*5 pav. Eulerio metodo kiekvieno objekto didžiausias aukštis ir laikas kada pradėjo leistis.*

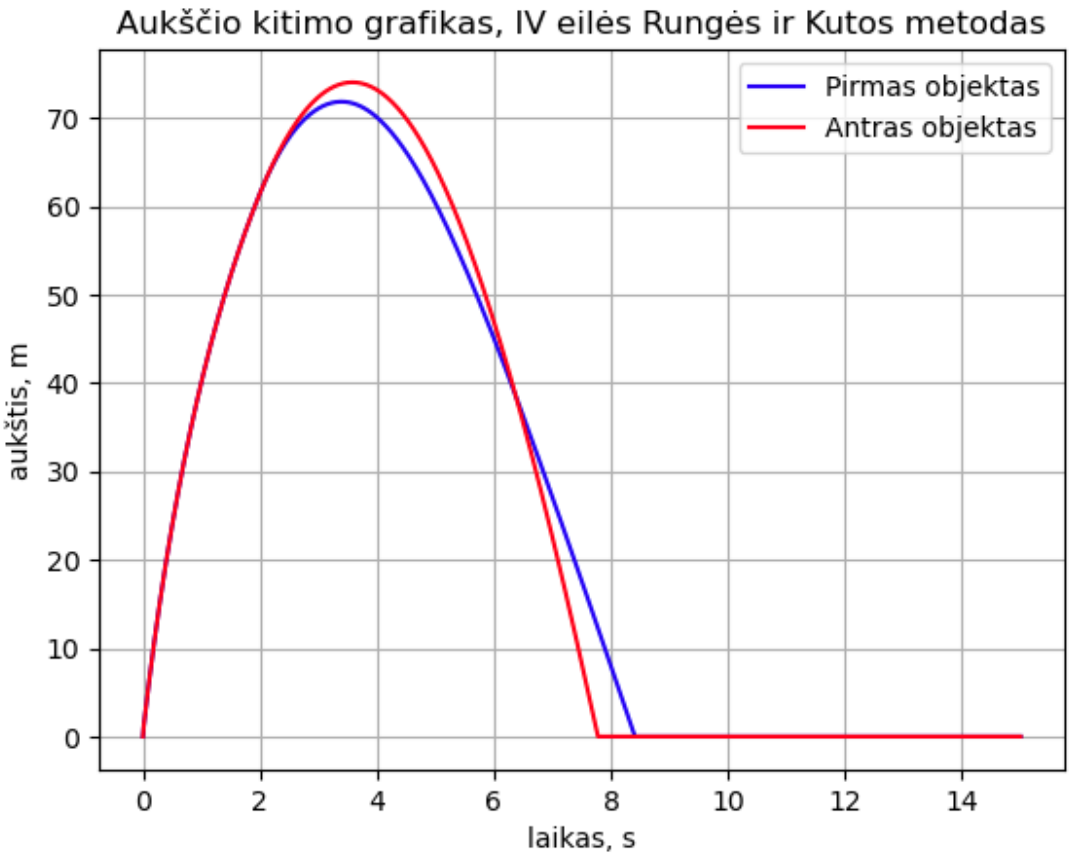
### 3.2. IV eilės Rungės ir Kutos metodo realizacija

```
for i in range (N-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, 2], i + 1] = 0
    if rez[1, i + 1] <= 0 : rez[[1, 3], i + 1] = 0
```

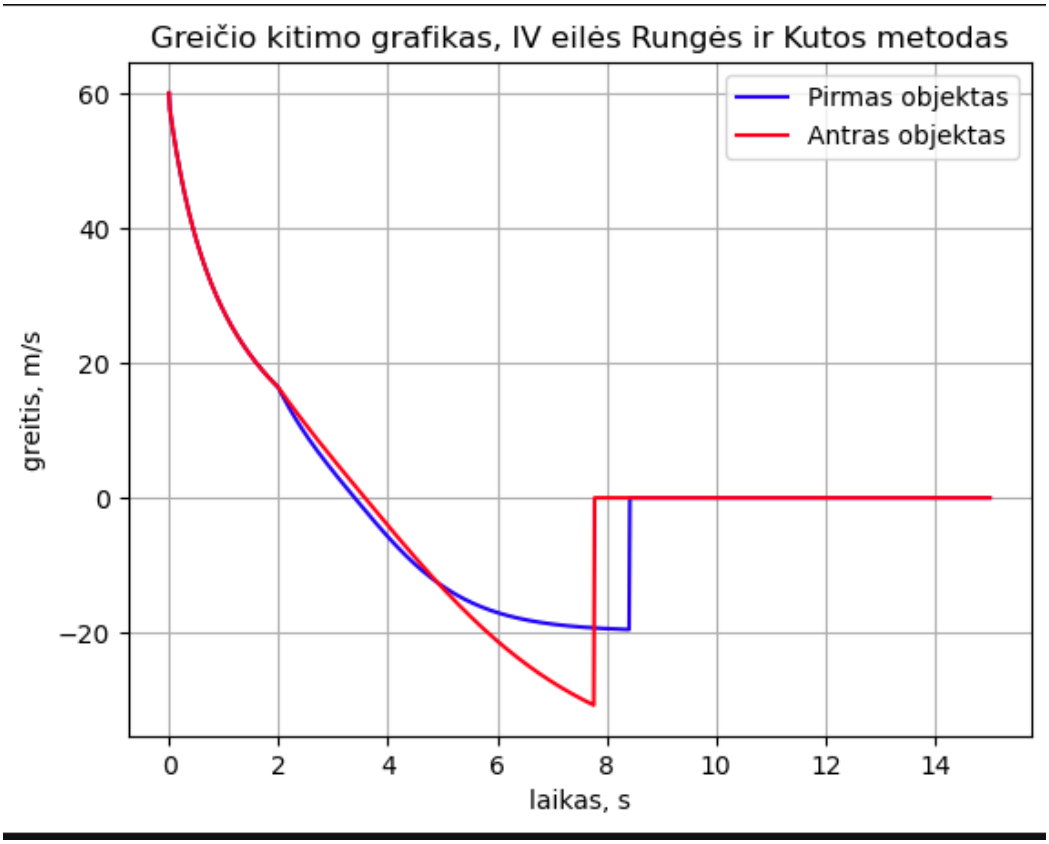
*6 pav. IV eilės Rungės ir Kutos metodo realizacija.*

IV eilės Rungės ir Kutos metodo veikimas:

- **Pirmas etapas**: prognozuojame, taikydami Eulerio metodą žingsniu dx/2;
- **Antras etapas:** koreguojame, taikydami atgalinį Eulerio metodą žingsniu dx/2;
- **Trečias etapas:** prognozuojame, taikydami vidurinio taško formulę žingsniu dx;
- **Ketvirtas etapas:** koreguojame, taikydami Simpsono koreguojančią formulę žingsniu dx.




*7 pav. IV eilės Rungės ir Kutos metodo aukščio kitimo grafikas.*




*8 pav. IV eilės Rungės ir Kutos metodo greičio kitimo grafikas.*

Kada kiekvienas objektas pasieks aukščiausią tašką ir pradės leistis?

```
Pirmas objektas maksimaliai pasiekė: 71.8444245798354 m., ir pradėjo leistis laiko momentu: 3.400900900900901 s.
Antras objektas maksimaliai pasiekė: 74.03682727484295 m., ir pradėjo leistis laiko momentu: 3.581081081081081 s.
```

*9 pav. IV eilės Rungės ir Kutos metodo kiekvieno objekto didžiausias aukštis ir laikas kada pradėjo leistis.*

### 3.3. Kodo fragmentas

L4-1dalis.py

```
import matplotlib.pyplot as plt
import numpy as np
import time
from scipy.integrate import solve_ivp
obj1maxHeight = 0
obj2maxHeight = 0
obj1Time = 0
obj2Time = 0
def funk(X, t):
  obj1Height=X[0]
  obj2Height=X[1]
  obj1Speed=X[2]
  obj2Speed=X[3]
  global obj1maxHeight, obj2maxHeight
  global obj1Time, obj2Time
  obj1M = 0.8 #kg
  obj2M = 0.8 #kg
  startingSpeed = 200 #m/s
  resistanceCombined = 0.01 #kg/m
  resistanceObj1 = 0.02 #kg/m
  resistanceObj2 = 0.005 #kg/m
```

```
  ts = 2 #s
  g = 9.8
  Fgravity = (obj1M + obj2M) * g
  rez=np.zeros(4,dtype=float)

  if t == 0:
    obj1Speed = startingSpeed
    obj2Speed = startingSpeed

  if t < ts:
    Fdrag = resistanceCombined * obj1Speed**2
    F = (Fgravity - Fdrag) / (obj1M + obj2M)

    rez[0] = obj1Speed
    rez[1] = obj2Speed
    rez[2] = -g + (F-resistanceObj1*obj1Speed**2*np.sign(obj1Speed))/(obj1M +
obj2M)
    rez[3] = rez[2]
  else:
    rez[0] = obj1Speed
    rez[1] = obj2Speed
    rez[2] = -g - resistanceObj1 * obj1Speed**2 * np.sign(obj1Speed) / obj1M
    rez[3] = -g - resistanceObj2 * obj2Speed**2 * np.sign(obj2Speed) / obj2M

  if obj1Height >= obj1maxHeight:
    obj1maxHeight = obj1Height
    obj1Time = t

  if obj2Height >= obj2maxHeight:
    obj2maxHeight = obj2Height
    obj2Time = t

  return rez

ttt = 15
t_span = (0, ttt)
t = np.linspace(0, ttt, 1000)
dt = t[1] - t[0]
h0 = 0
N = len(t);
rez = np.zeros([4, N], dtype=float)
rez[:, 0] = np.array([h0,h0,60,60]);

for i in range(N - 1):
  rez[:, i + 1] = rez[:, i] + funk(rez[:, i], t[i]) * dt
  if rez[0, i + 1] <= 0 : rez[[0, 2], i + 1] = 0
  if rez[1, i + 1] <= 0 : rez[[1, 3], i + 1] = 0
print('Pirmas objektas maksimaliai pasiekė:', obj1maxHeight, 'm., ir pradėjo
leistis laiko momentu:', obj1Time, 's.')
print('Antras objektas maksimaliai pasiekė:', obj2maxHeight, 'm., ir pradėjo
leistis laiko momentu:', obj2Time, 's.')

# rezultatu pavaizdavimas:
fig1=plt.figure(1); ax1=fig1.add_subplot(1,1,1); ax1.set_xlabel('laikas,
s');ax1.set_ylabel('aukštis, m');ax1.grid();plt.title('Aukščio kitimo grafika,
Eulerio metodas')
ax1.plot(t,rez[0,:],'b-');ax1.plot(t,rez[1,:],'r-'); plt.legend(['Pirmas
objektas','Antras objektas']);plt.show()

fig2=plt.figure(1); ax2=fig2.add_subplot(1,1,1); ax2.set_xlabel('laikas,
s');ax2.set_ylabel('greitis, m/s');ax2.grid();plt.title('Greičio kitimo grafikas,
Eulerio metodas')
ax2.plot(t,rez[2,:],'b-');ax2.plot(t,rez[3,:],'r-');plt.legend(['Pirmas
objektas','Antras objektas']);plt.show()
```

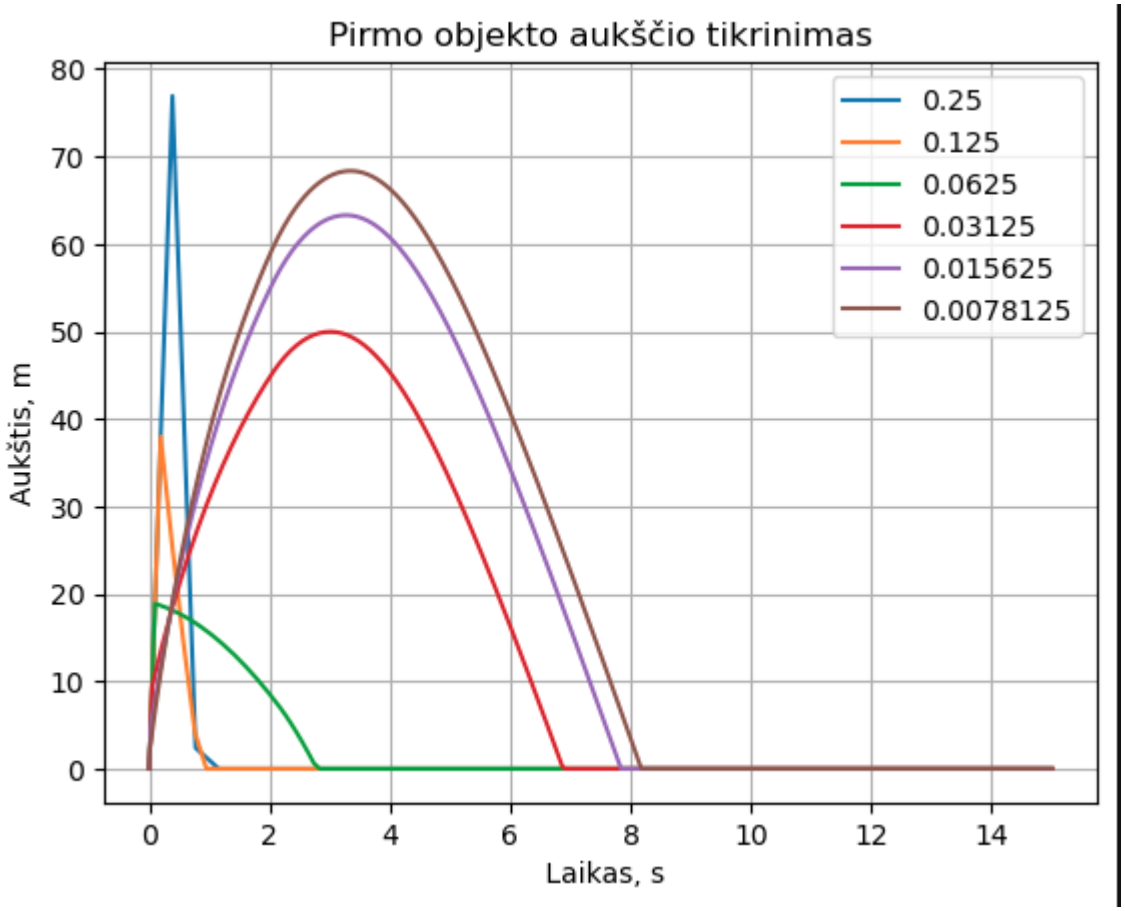
```
for i in range (N-1) :
    fz = rez[:,i] + funk(rez[:,i], t[i]) * dt / 2
    fzz = rez[:,i] + funk(fz, t[i] + dt / 2) * dt / 2
    fzzz = rez[:,i] + funk(fzz, t[i] + dt / 2) * dt
    rez[:, i + 1] = rez[:, i] + dt / 6 * (funk(rez[:, i], t[i]) + 2 * funk(fz,
t[i] + dt / 2) + 2 * funk(fzz, t[i] + dt / 2) + funk(fzzz, t[i] + dt))
    if rez[0, i + 1] <= 0: rez[[0, 2], i + 1] = 0
    if rez[1, i + 1] <= 0 : rez[[1, 3], i + 1] = 0
print('Pirmas objektas maksimaliai pasiekė:', obj1maxHeight, 'm., ir pradėjo
leistis laiko momentu:', obj1Time, 's.')
print('Antras objektas maksimaliai pasiekė:', obj2maxHeight, 'm., ir pradėjo
leistis laiko momentu:', obj2Time, 's.')

# rezultatu pavaizdavimas:
fig1=plt.figure(1); ax1=fig1.add_subplot(1,1,1); ax1.set_xlabel('laikas,
s');ax1.set_ylabel('aukštis, m');ax1.grid();plt.title('Aukščio kitimo grafika, IV
eilės Rungės ir Kutos metodas')
ax1.plot(t,rez[0,:],'b-');ax1.plot(t,rez[1,:],'r-'); plt.legend(['Pirmas
objektas','Antras objektas']);plt.show()

fig2=plt.figure(1); ax2=fig2.add_subplot(1,1,1); ax2.set_xlabel('laikas,
s');ax2.set_ylabel('greitis, m/s');ax2.grid();plt.title('Greičio kitimo grafikas,
IV eilės Rungės ir Kutos metodas')
ax2.plot(t,rez[2,:],'b-');ax2.plot(t,rez[3,:],'r-');plt.legend(['Pirmas
objektas','Antras objektas']);plt.show()
```

# 4. Sprendinio tikslumo įvertinimas

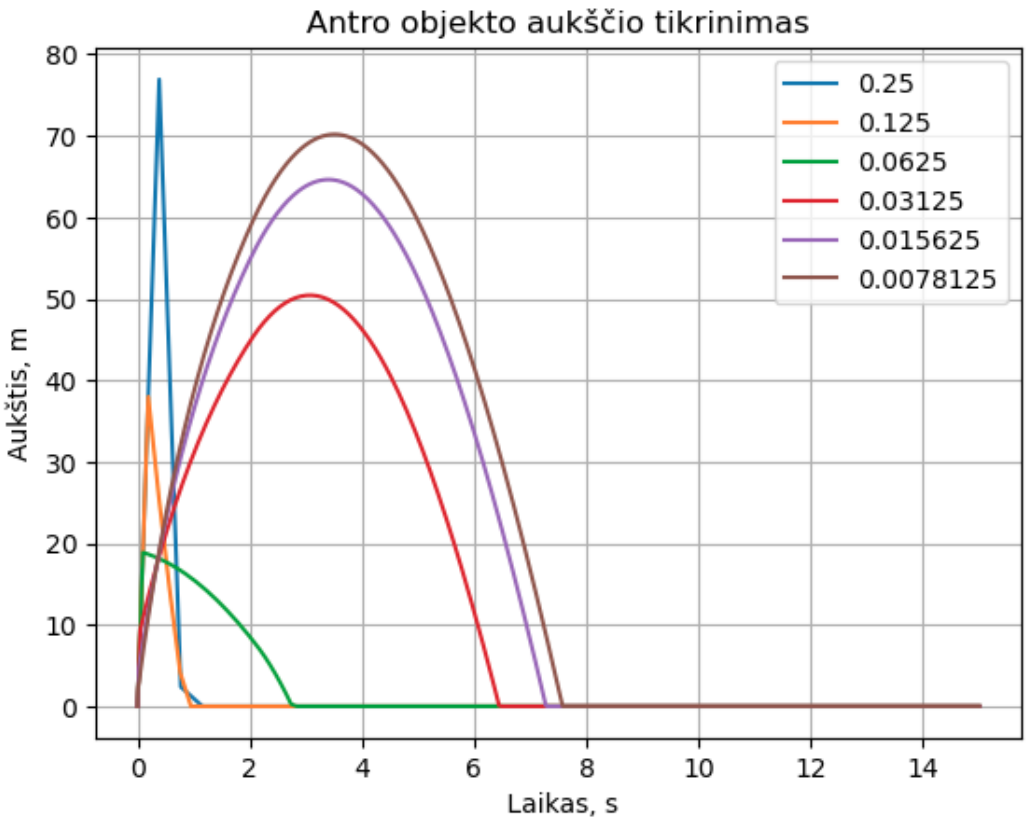
Pasirenkame pradines žingsnių vertes su kuriomis testuosime metodų tikslumo įvertinimus:

Žingsnio dydžiai = { '0.25','0.125', '0.0625', '0.03125', '0.015625', '0.0078125' }

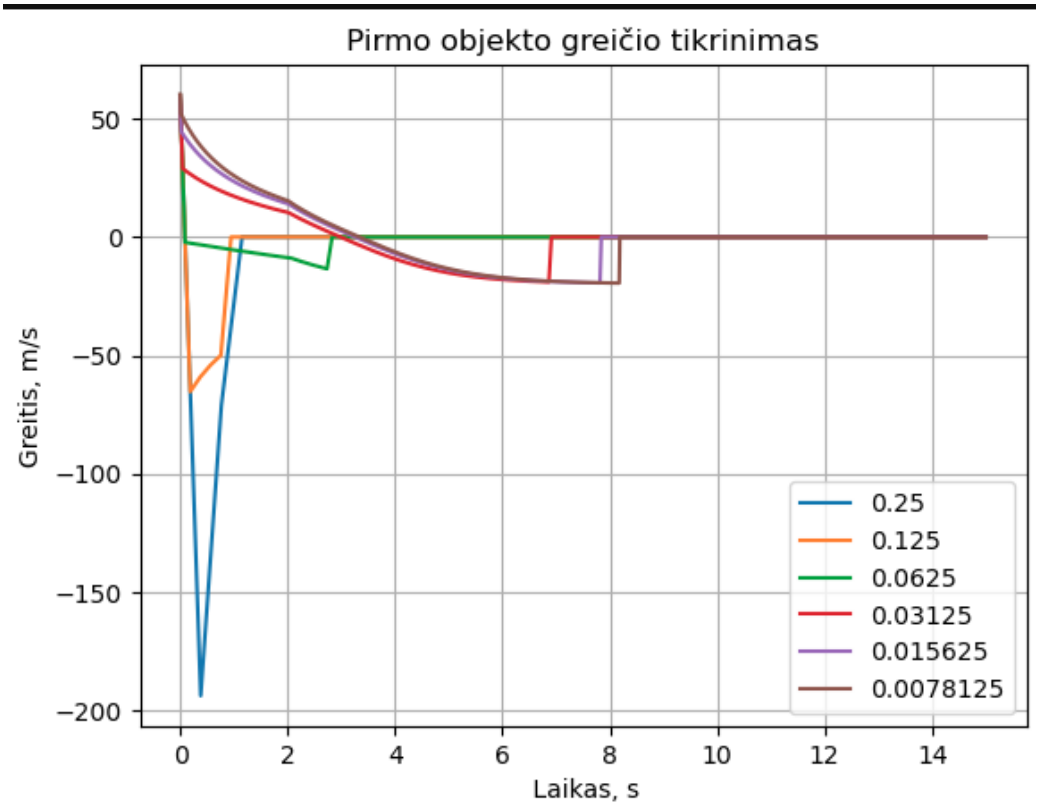
## 4.1. Sprendinio tikslumo įvertinimas Eulerio metodu



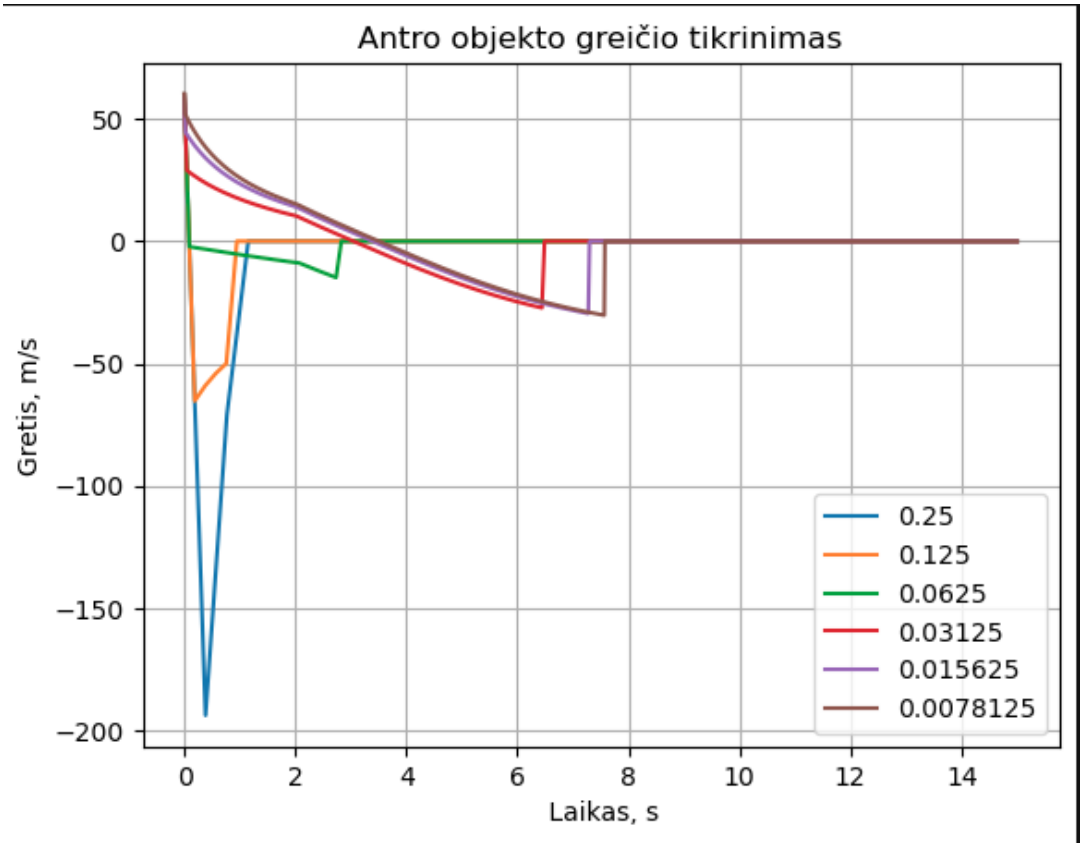

*10 pav. Pirmo objekto aukščio tikrinimas naudojant Eulerio metodą.*




*11 pav. Antro objekto aukščio tikrinimas naudojant Eulerio metodą.*

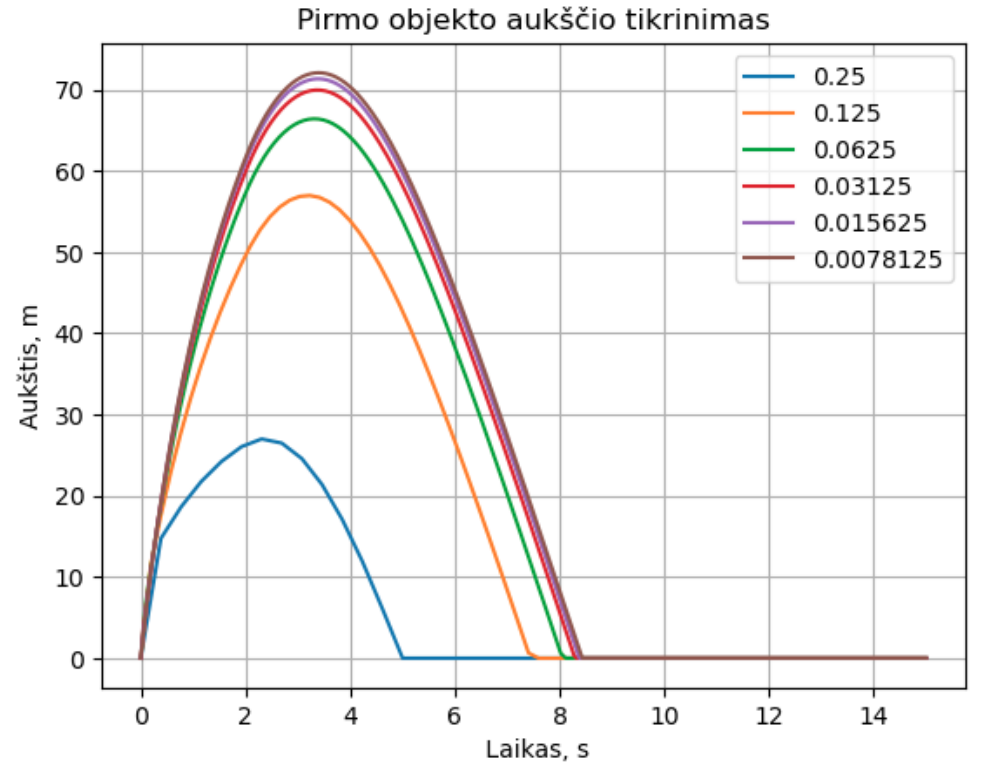



*12 pav. Pirmo objekto greičio tikrinimas naudojant Eulerio metodą.*

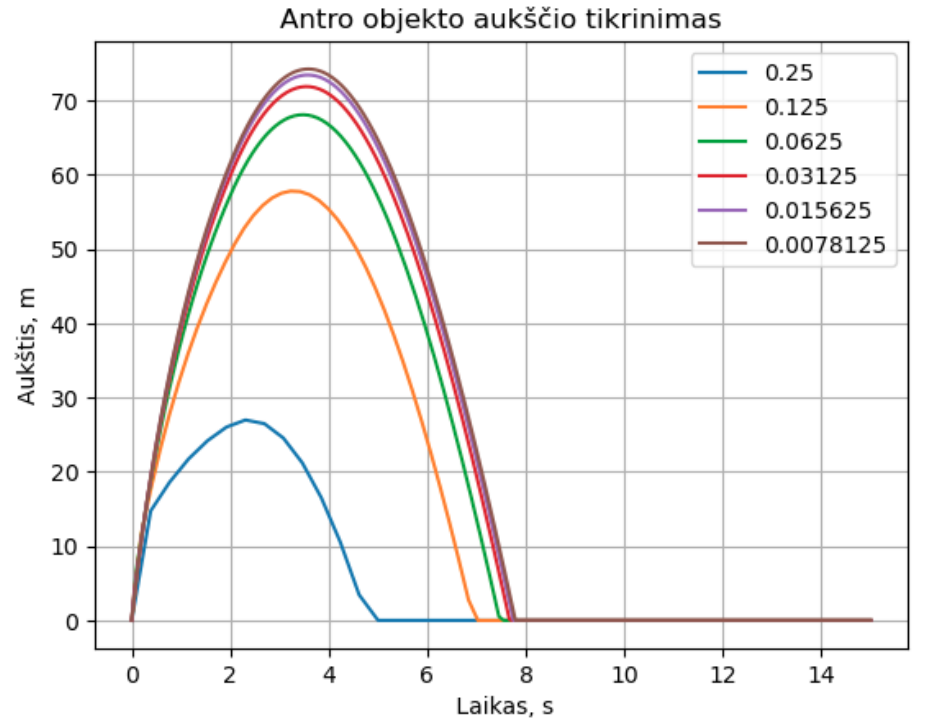



*13 pav. Antro objekto greičio tikrinimas naudojant Eulerio metodą.*

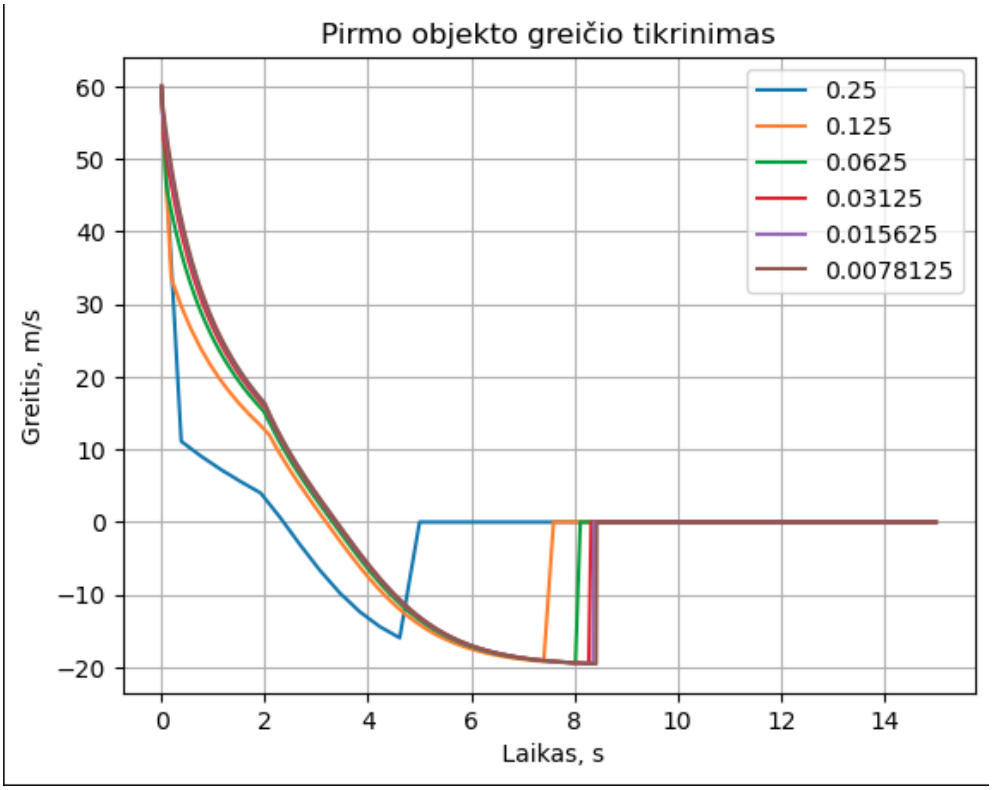
## 4.2. Sprendinio tikslumo įvertinimas IV eilės Rungės ir Kutos metodu



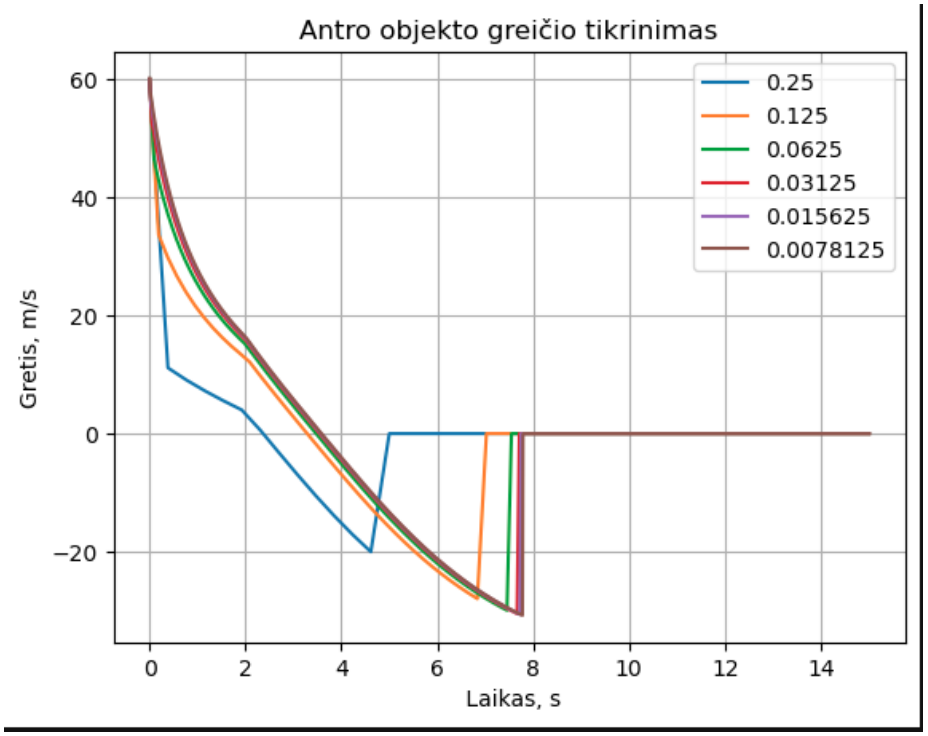

*14 pav. Pirmo objekto aukščio tikrinimas naudojant IV eilės Rungės ir Kutos metodą.*




*15 pav. Antro objekto aukščio tikrinimas naudojant IV eilės Rungės ir Kutos metodą.*




*16 pav. Pirmo objekto greičio tikrinimas naudojant IV eilės Rungės ir Kutos metodą.*




*17 pav. Antro objekto greičio tikrinimas naudojant IV eilės Rungės ir Kutos metodą.*

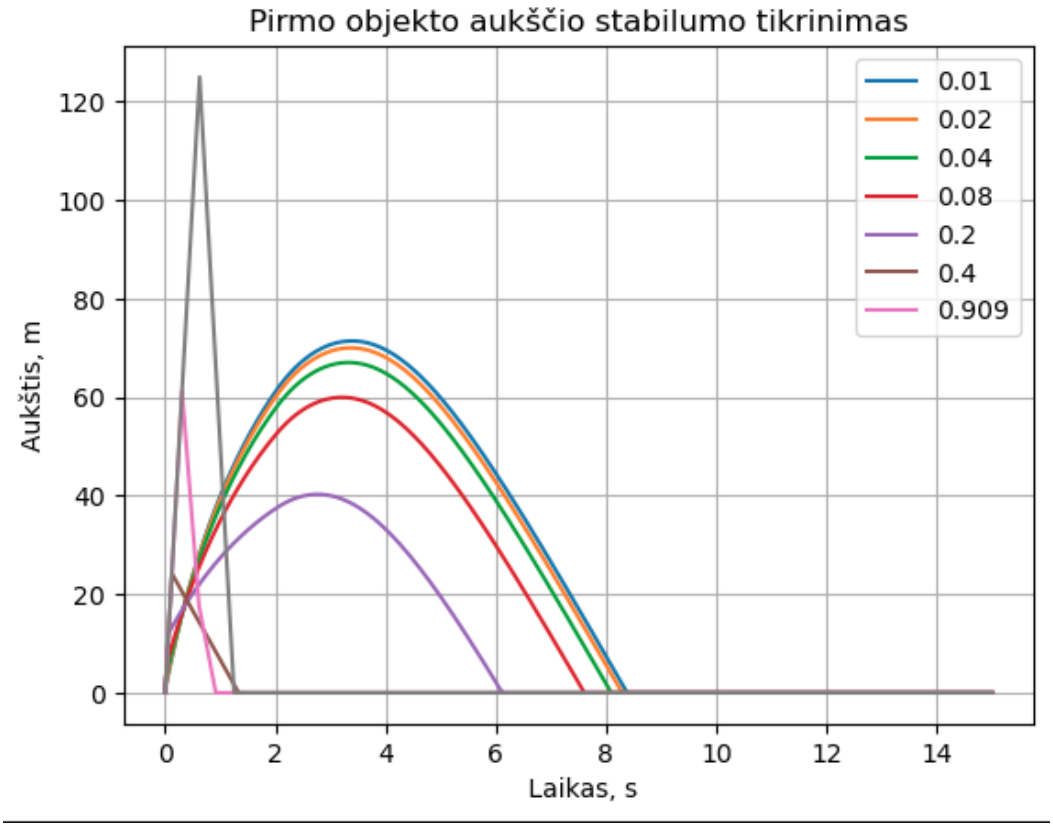
Taikant IV eilės Rungės ir Kutos metodą gauname tikslesnes paklaidas.

# 5. Sprendinio stabilumo įvertinimas

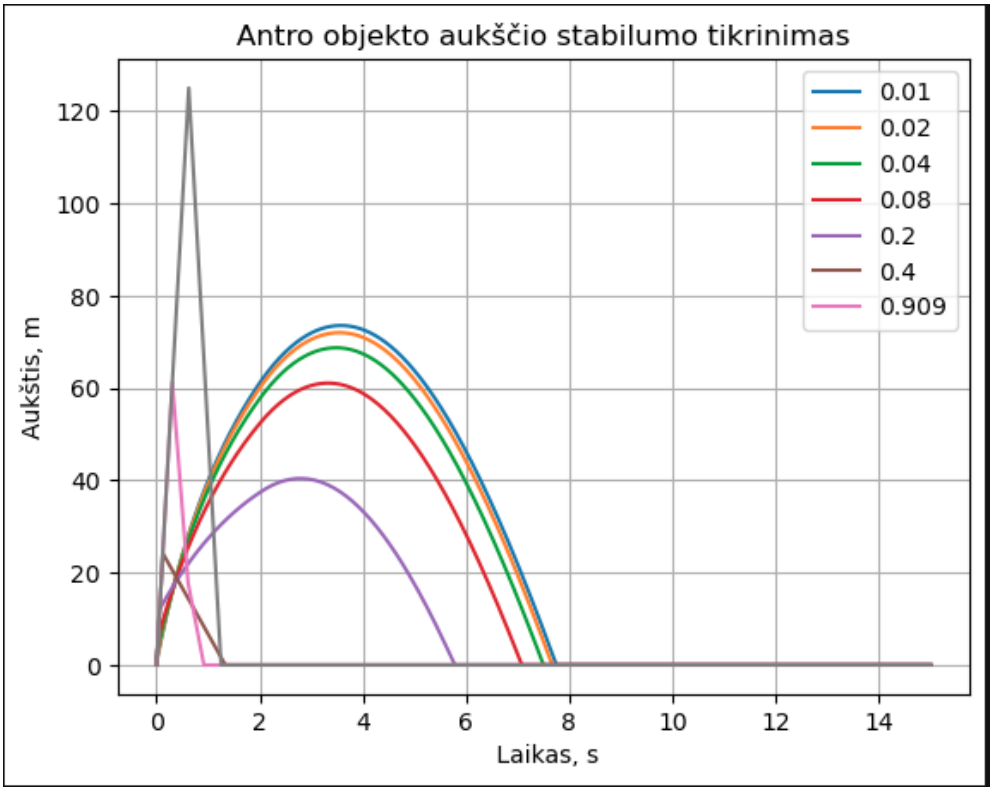
Pasirenkame pradines žingsnių vertes su kuriomis testuosime metodų stabilumo įvertinimus:

Žingsnio dydžiai = { '0.01', '0.02', '0.04', '0.08', '0.2', '0.4', '0.909'}

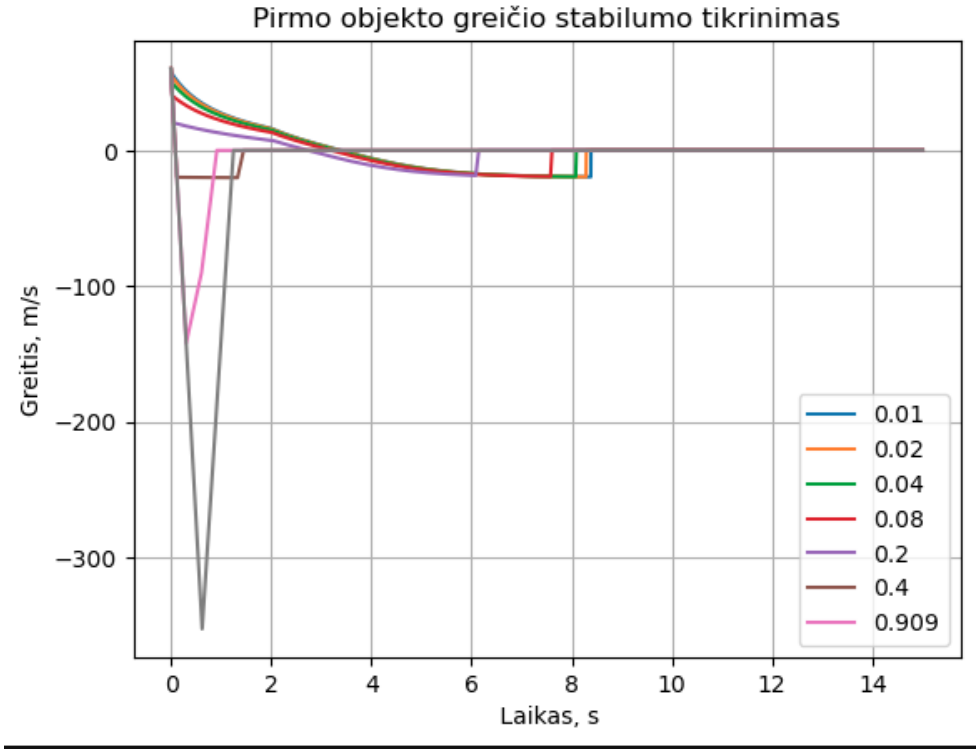
## 5.1. Sprendinio stabilumo įvertinimas Eulerio metodu



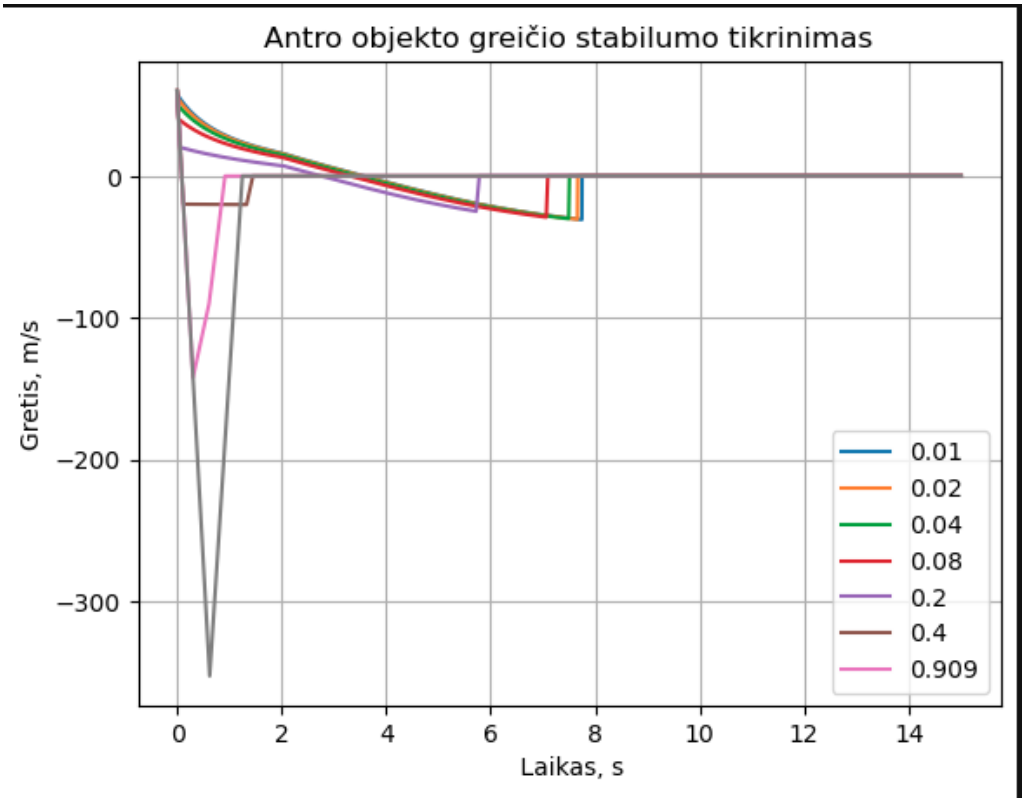

*18 pav. Pirmo objekto aukščio stabilumo įvertinimas naudojant Eulerio metodą.*




*19 pav. Antro objekto aukščio stabilumo įvertinimas naudojant Eulerio metodą.*



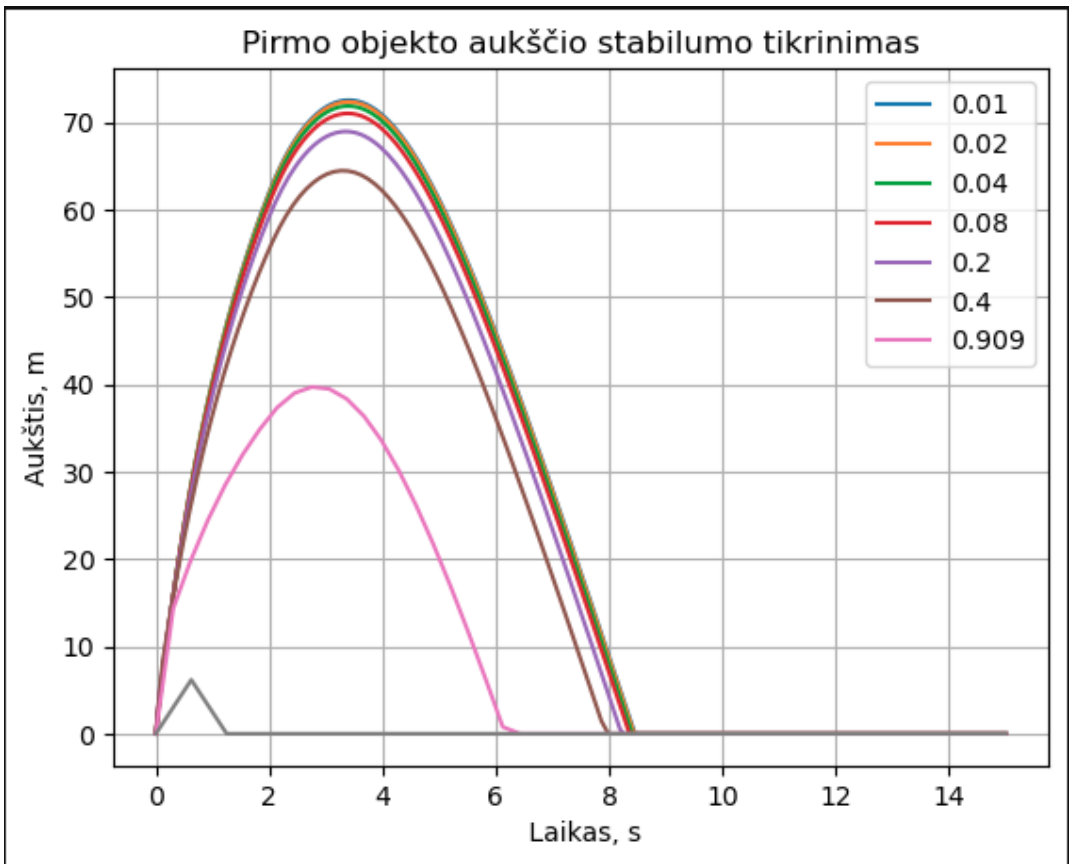

*20 pav. Pirmo objekto greičio stabilumo įvertinimas naudojant Eulerio metodą.*



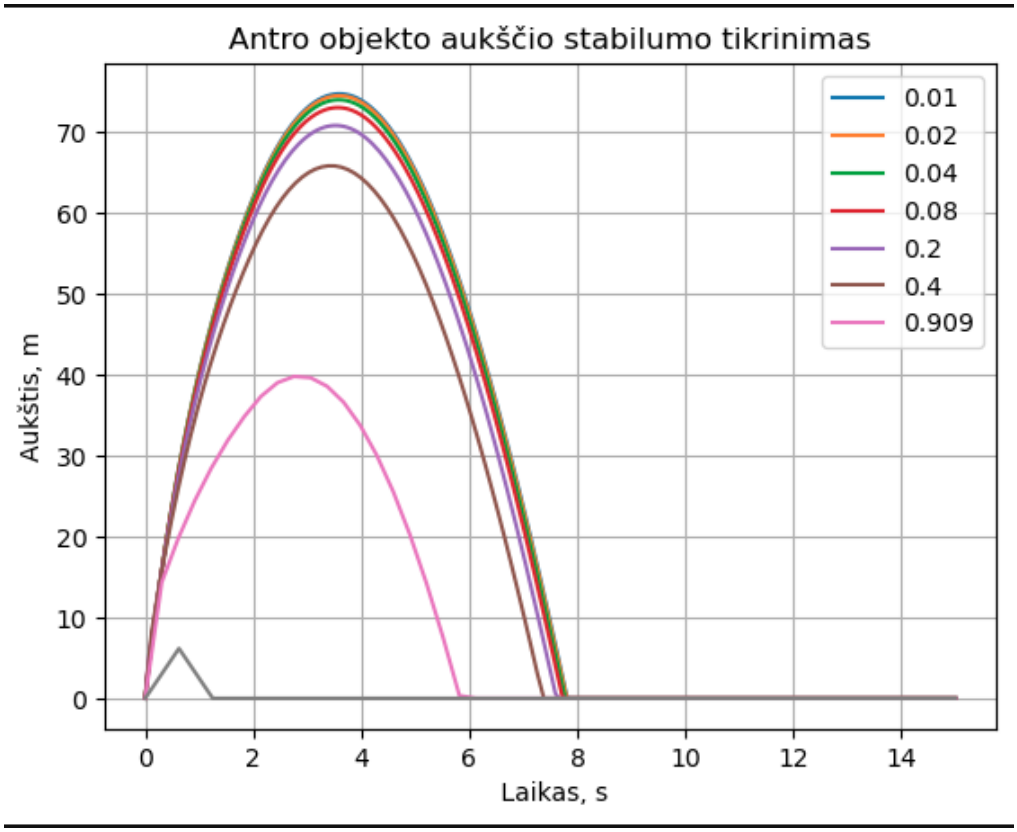

*21 pav. Antro objekto greičio stabilumo įvertinimas naudojant Eulerio metodą.*

Eulerio metodas yra stabilus, kai žingsnis yra ne didesnis už 0.2.

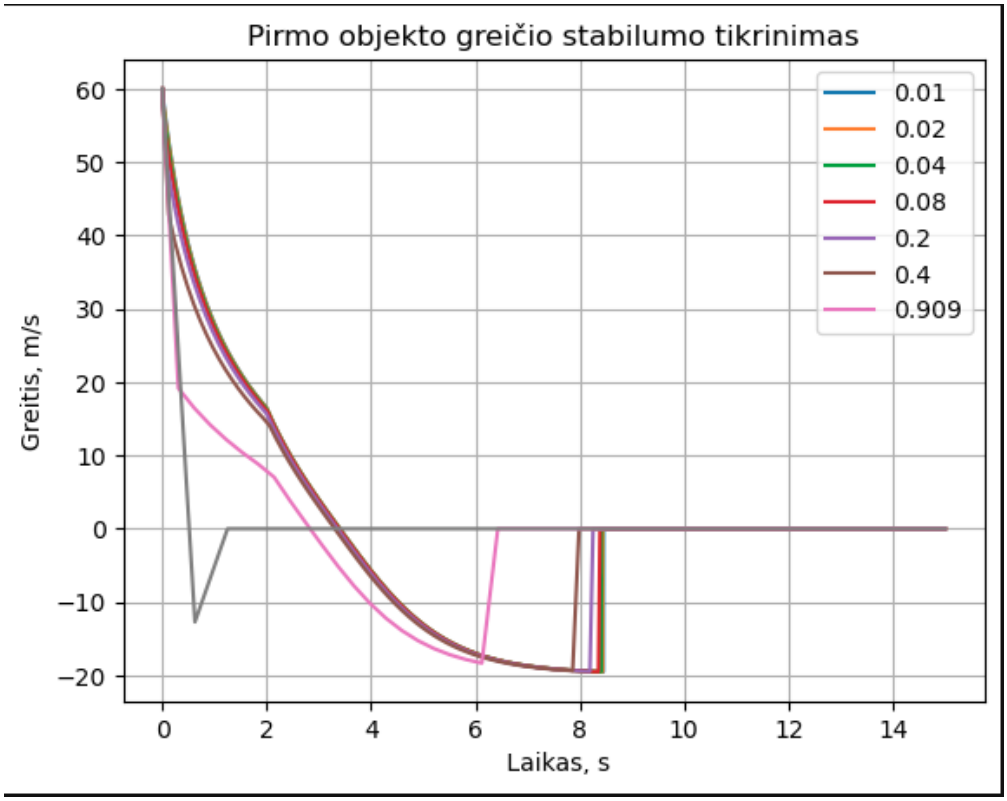
### 5.2. Sprendinio stabilumo įvertinimas IV eilės Rungės ir Kutos metodu



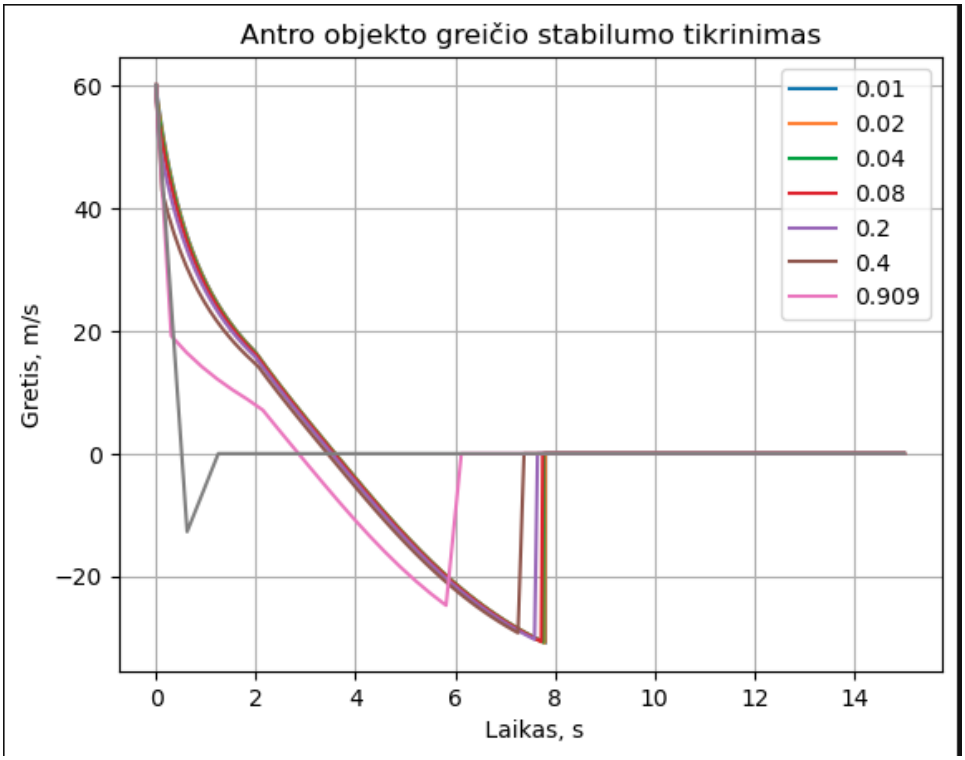

*22 pav. Pirmo objekto aukščio stabilumo įvertinimas naudojant IV eilės Rungės ir Kutos metodą.*




*23 pav. Antro objekto aukščio stabilumo įvertinimas naudojant IV eilės Rungės ir Kutos metodą.*




*24 pav. Pirmo objekto greičio stabilumo įvertinimas naudojant IV eilės Rungės ir Kutos metodą.*

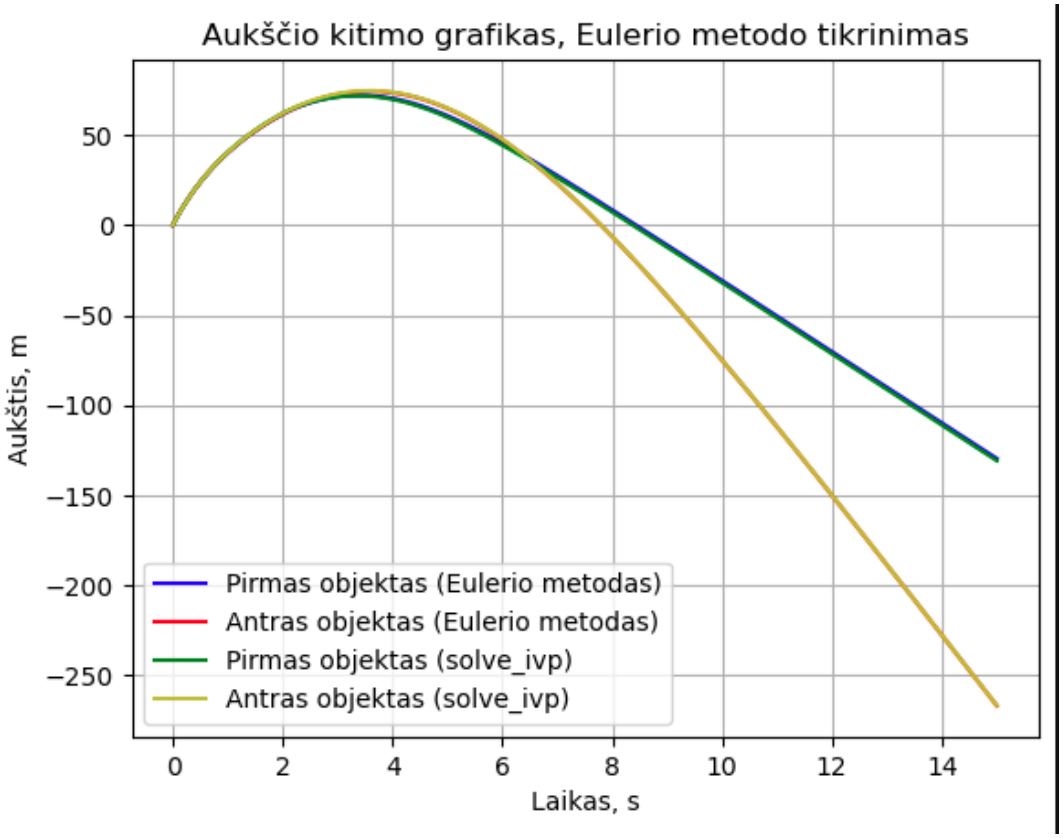



*25 pav. Antro objekto greičio stabilumo įvertinimas naudojant IV eilės Rungės ir Kutos metodą.*

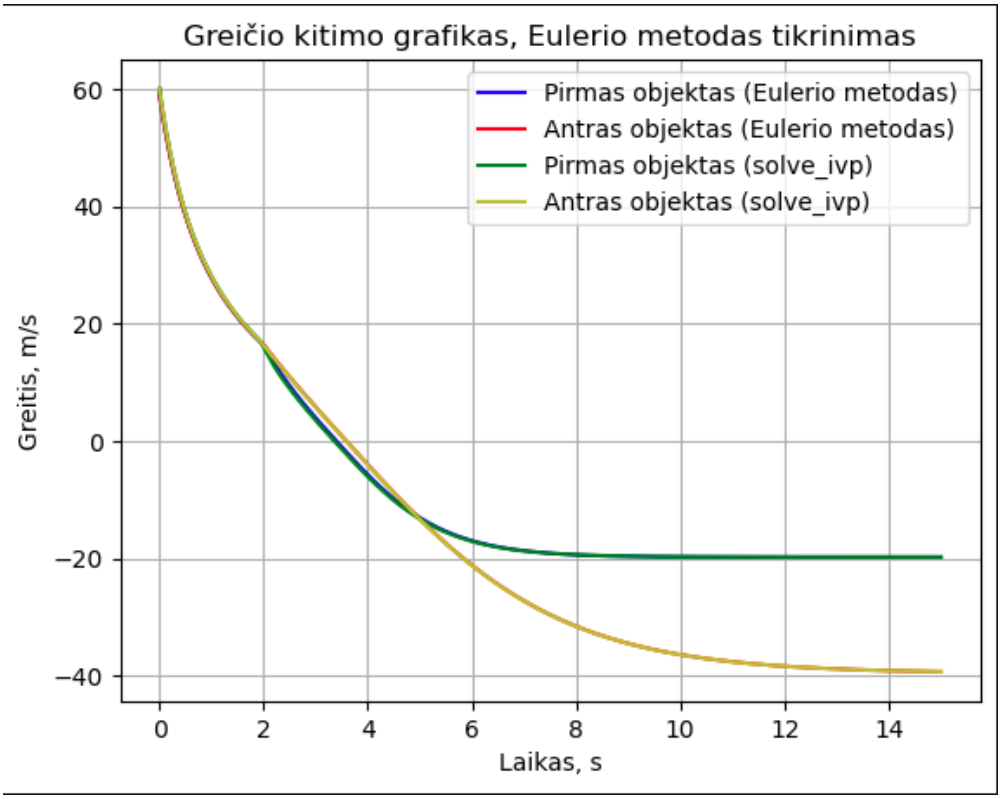
Taikant IV eilės Rungės ir Kutos metodą pastebime, kad jis yra žymiai stabilesnis už Eulerio metodą, didžiausias stabilus žingsnis ne didesnis už 0.4.

## 6. Gauto sprendinio patikrinimas su standartine Python funkcija solve_ivp.

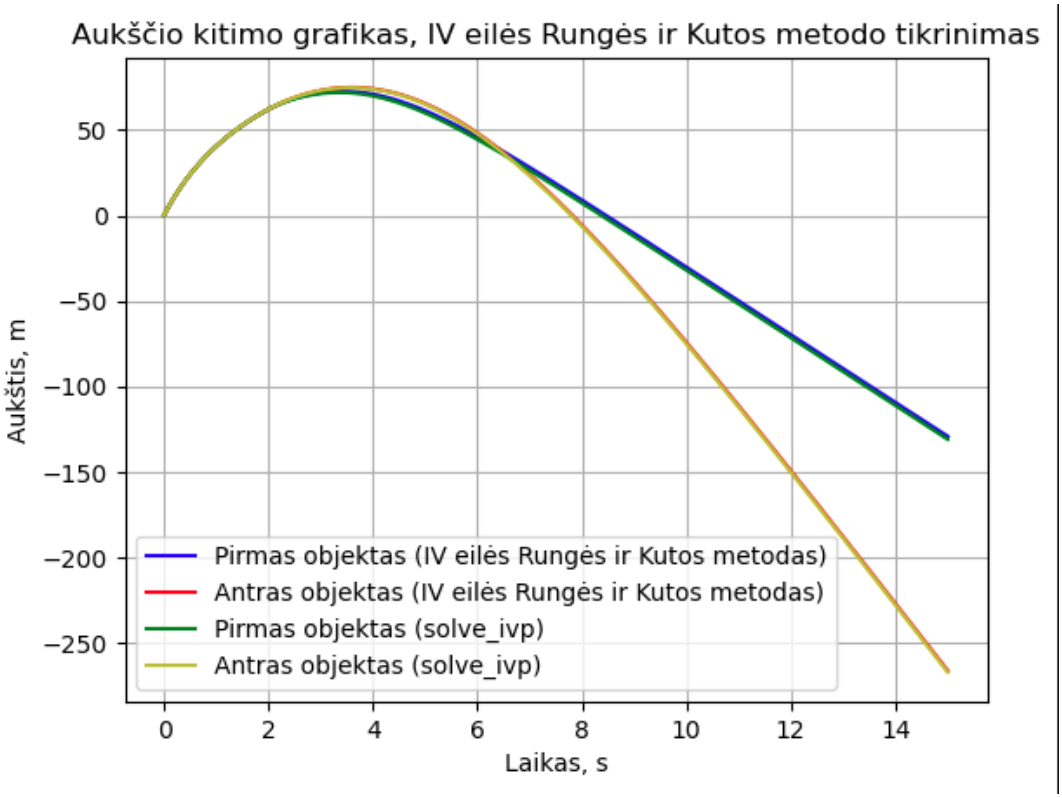
Naudojame Python scipy.integrate bibliotekos funkcija solve_ivp.



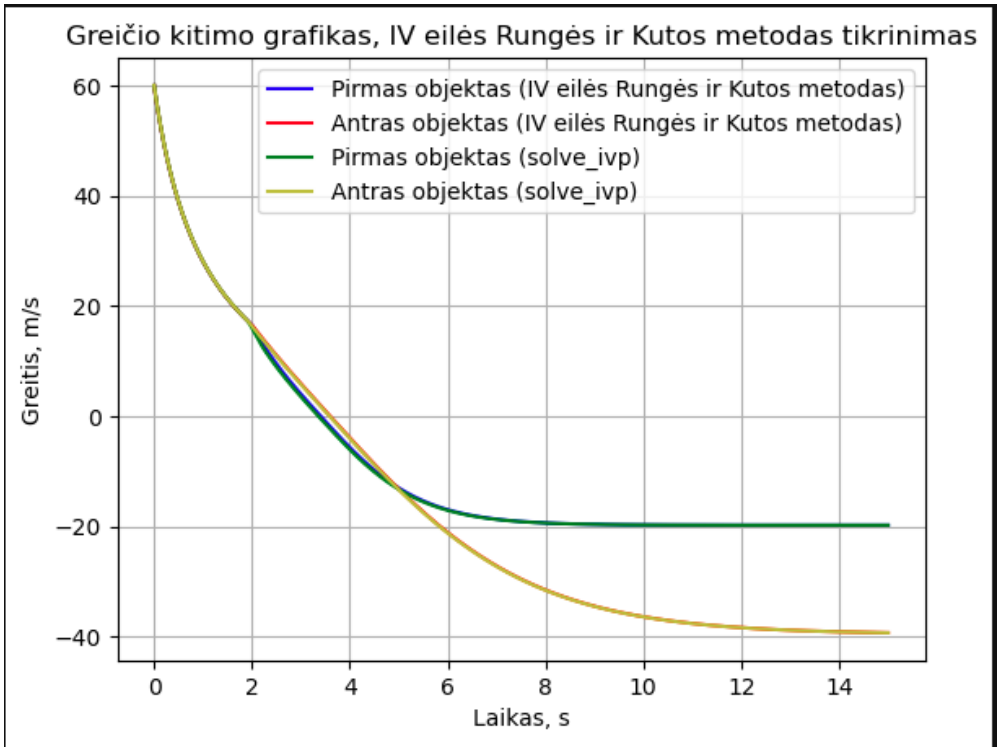

*26 pav. Aukščio kitimo grafikas, naudojant Eulerio metodą ir solve_ivp.*




*27 pav. Greičio kitimo grafikas, naudojant Eulerio metodą ir solve_ivp.*




*28 pav. Aukščio kitimo grafikas, naudojant IV eilės Rungės ir Kutos metodą ir solve_ivp.*




*29 pav. Greičio kitimo grafikas, naudojant IV eilės Rungės ir Kutos metodą ir solve_ivp.*

Python funkcija solve_ivp patvirtino Eulerio ir IV eilės Rungės ir Kutos metodų tikslumą.

## 7. Išvados

Susipažinta su Eulerio ir IV eilės Rungės ir Kutos metodo veikimo principais. Padaryta išvada, dėl IV eilės Rungės ir Kutos metodo pranašumo prieš Eulerio metodą.