Kauno technologijos universitetas

Informatikos fakultetas

IV projektinio darbo ataskaita

Laboratorinio užduotis

Aistis Jakutonis

Studentas

Čalnerytė Dalia
Kriščiūnas Andrius

Dėstytojai

Kaunas, 2025

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. Užduoties aprašymas

Uždavinio variantas: 5.

1 Bendroji sąlyga.

1. Žemiau pateikti uždaviniai paprastųjų diferencialinių lygčių sistemų sprendimui. Remdamiesi tame pačiame faile pateiktų fizikinių dėsnių aprašymais, nurodytam variantui sudarykite diferencialinę lygtį arba lygčių sistemą. Lygties ar lygčių sistemos sudarymą paaiškinkite ataskaitoje.
2. Diferencialinę lygtį (arba lygčių sistemą) išspręskite Eulerio ir IV eilės Rungės ir Kutos metodais.
3. Keisdami metodo žingsnį įsitikinkite, kad gavote tikslų sprendinį. Atsakykite į uždavinyje pateiktus klausimus. Tuo pačiu metodu naudojant skirtingus žingsnius gautus sprendinius pavaizduokite viename grafike. Palyginkite metodus tikslumo prasme.
4. Keisdami metodo žingsnį nustatykite didžiausią žingsnį, su kuriuo metodas išlieka stabilus. Tuo pačiu metodu naudojant skirtingus žingsnius gautus sprendinius pavaizduokite viename grafike. Palyginkite metodus stabilumo prasme.
5. Patikrinkite gautą sprendinį su MATLAB standartine funkcija ode45, Python scipy.integrate bibliotekos funkcija solve_ivp ar kitais išoriniais šaltiniais. Tame pačiame grafike turi būti pateikti realizacijose ir naudojant išorinius šaltinius gauti sprendiniai.

2 pav. Bendros sąlygos/reikalavimai atliekant užduotį

2. Teorija

Paprastosios diferencialinės lygtys – tai tokios lygtis, kurios aprašo, kaip dydis keičiasi bėgant laikui. Vietoj to, kad žinotume tik patį dydį (pvz., kūno aukštį ar greitį), diferencialinė lygtis susieja šį dydį su jo kitimo greičiu.

Diferencialinė lygtis sudaroma pradedant nuo realios fizikinės situacijos ir pagrindinių dėsnių. Kadangi pagreitis yra greičio kitimo greitis, o greitis – padėties kitimo greitis, iš šių ryšių ir susiformuoja diferencialinė lygtis, kuri aprašo, kaip dydis keičiasi kiekvieną akimirką.

Eulerio metodas – paprasčiausias PDL skaitinio sprendimo metodas. Jis remiasi idėja, kad per nedidelį laiko žingsnį Δt sprendinį galime aproksimuoti tiesine priklausomybe. Metodas žingsnis po žingsnio „eina“ į priekį laike, kiekvieną kartą naudodamas esamo taško išvestinę kaip tiesinę prognozę kitam taškui.

IV eilės Rungės ir Kutos (RK4) metodas – tai tobulesnis skaitinis metodas diferencialinėms lygtims spręsti. Skirtingai nei Eulerio metodas, kuris kiekviename žingsnyje žiūri tik į vieną tašką, RK4 žingsnio metu įvertina keletą tarpinių taškų: pradžioje, viduryje žingsnio ir pabaigoje. Iš šių kelių „prognozių“ sudaro svorinį vidurkį. Dėl to gaunamas daug tikslesnis rezultatas, nors žingsnio dydis gali būti toks pats. Praktikoje tai reiškia, kad RK4, esant tam pačiam žingsniui, paprastai duoda daug geresnį priartėjimą prie tikrojo sprendinio nei Eulerio metodas.

Tikslumo atžvilgiu Eulerio ir RK4 metodai gana stipriai skiriasi. Eulerio metodas yra labai paprastas, bet netikslus: jei žingsnis nėra labai mažas, sprendinys greitai „nutolsta“ nuo tikslaus. Norint su juo gauti tvarkingus rezultatus, laiko žingsnį reikia rinktis labai mažą, tada skaičiavimų tampa daug. RK4 metodas, priešingai, yra daug tikslesnis: net ir su vidutinio dydžio žingsniu gauname gana gerą priartėjimą prie „tikro“ sprendinio.

Stabilumas parodo, ar metodas „neišprotėja“, kai skaičiuojame ilgiau arba naudojame didesnį laiko žingsnį. Eulerio metodas nėra itin stabilus: jei žingsnis parenkamas per didelis, sprendinys pradeda stipriai svyruoti, vertės gali nevaldomai augti arba tapti beprasmiškos. RK4 metodas yra stabilesnis – jis leidžia naudoti kiek didesnius žingsnius, išlaikant prasmingą sprendinį.

2.1. Fizikinis modelis:

Pagreitis (greičio kitimo greitis), įvertinus gravitaciją ir kvadratinį oro pasipriešinimą, aprašomas taip:

$$a(v) = \frac{dv}{dt} = -g - \frac{k}{m} v \mid v \mid.$$

Čia:

- g– laisvojo kritimo pagreitis,
- k– oro pasipriešinimo koeficientas,
- m– kūno masė,
- v– vertikalus greitis (aukštyn teigiamas).

Judėjimą aprašanti **diferencialinių lygčių sistema**:

$$\begin{cases} \dfrac{dh}{dt} = v, \\ \dfrac{dv}{dt} = -g - \dfrac{k}{m} v \mid v \mid. \end{cases}$$

2.2. Dviejų fazių modelis:

1 fazė (0 ≤ t ≤ t_s): abu kūnai sujungti į vieną, todėl masė ir pasipriešinimas:

$$M = m_1 + m_2, k = k_s.$$

Tada:

$$\begin{cases} \dfrac{dh}{dt} = v, \\ \dfrac{dv}{dt} = -g - \dfrac{k_s}{m_1 + m_2} v \mid v \mid. \end{cases}$$

2 fazė (t > t_s): kūnai juda atskirai.

Pirmas objektas:

$$\begin{cases} \dfrac{dh_1}{dt} = v_1, \\ \dfrac{dv_1}{dt} = -g - \dfrac{k_1}{m_1} v_1 \mid v_1 \mid. \end{cases}$$

Antras objektas:

$$\begin{cases} \dfrac{dh_2}{dt} = v_2, \\ \dfrac{dv_2}{dt} = -g - \dfrac{k_2}{m_2} v_2 \mid v_2 \mid. \end{cases}$$

Pradiniai duomenys 2 fazei paimami iš 1 fazės pabaigos:

$$h_1(t_s) = h_2(t_s) = h(t_s), v_1(t_s) = v_2(t_s) = v(t_s).$$

2.3. Eulerio metodas:

Eulerio žingsnis, skaičiuojant aukštį ir greitį:

$$\begin{aligned} h_{n+1} &= h_n + \Delta t \cdot v_n, \\ v_{n+1} &= v_n + \Delta t \cdot a(v_n) = v_n + \Delta t \left(-g - \frac{k}{m} v_n \mid v_n \mid \right). \end{aligned}$$

2.4. IV eilės Rungės ir Kutos metodas:

Sistema (h, v) sprendžiama 4-os eilės RK metodu. Vienam žingsniui nuo (h_n, v_n) iki $(h_{(n+1)}, v_{(n+1)})$ su žingsniu Δt:

Galutinis žingsnio atnaujinimas:

$$h_{n+1} = h_n + \frac{\Delta t}{6}(k_1^h + 2k_2^h + 2k_3^h + k_4^h),$$
$$v_{n+1} = v_n + \frac{\Delta t}{6}(k_1^v + 2k_2^v + 2k_3^v + k_4^v).$$

2.5. Aukščiausio taško radimas:

Kai ieškome aukščiausio taško:

- randame indeksą i, kai greitis keičia ženklą:
  $v_{i-1} > 0$ ir $v_i \leq 0$;
- tarp t_{i-1} ir t_i darome **linijinę interpoliaciją**, manydami, kad greitis keičiasi tolygiai.

Žymime $\Delta t = t_i - t_{i-1}$ ir

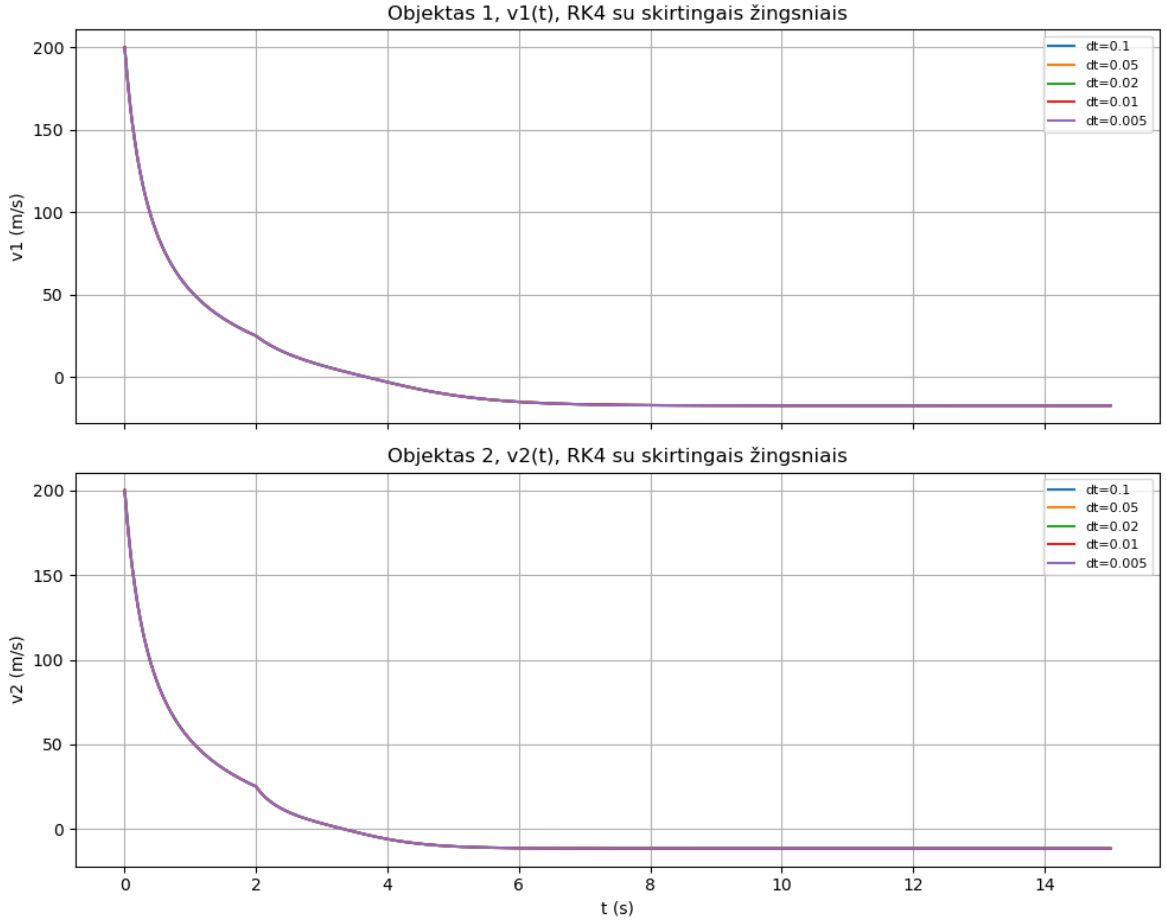
$$\text{frac} = -\frac{v_{i-1}}{v_i - v_{i-1}}.$$

Tada:

$$t_{\max} = t_{i-1} + \text{frac} \cdot (t_i - t_{i-1}),$$
$$h_{\max} = h_{i-1} + \text{frac} \cdot (h_i - h_{i-1}).$$

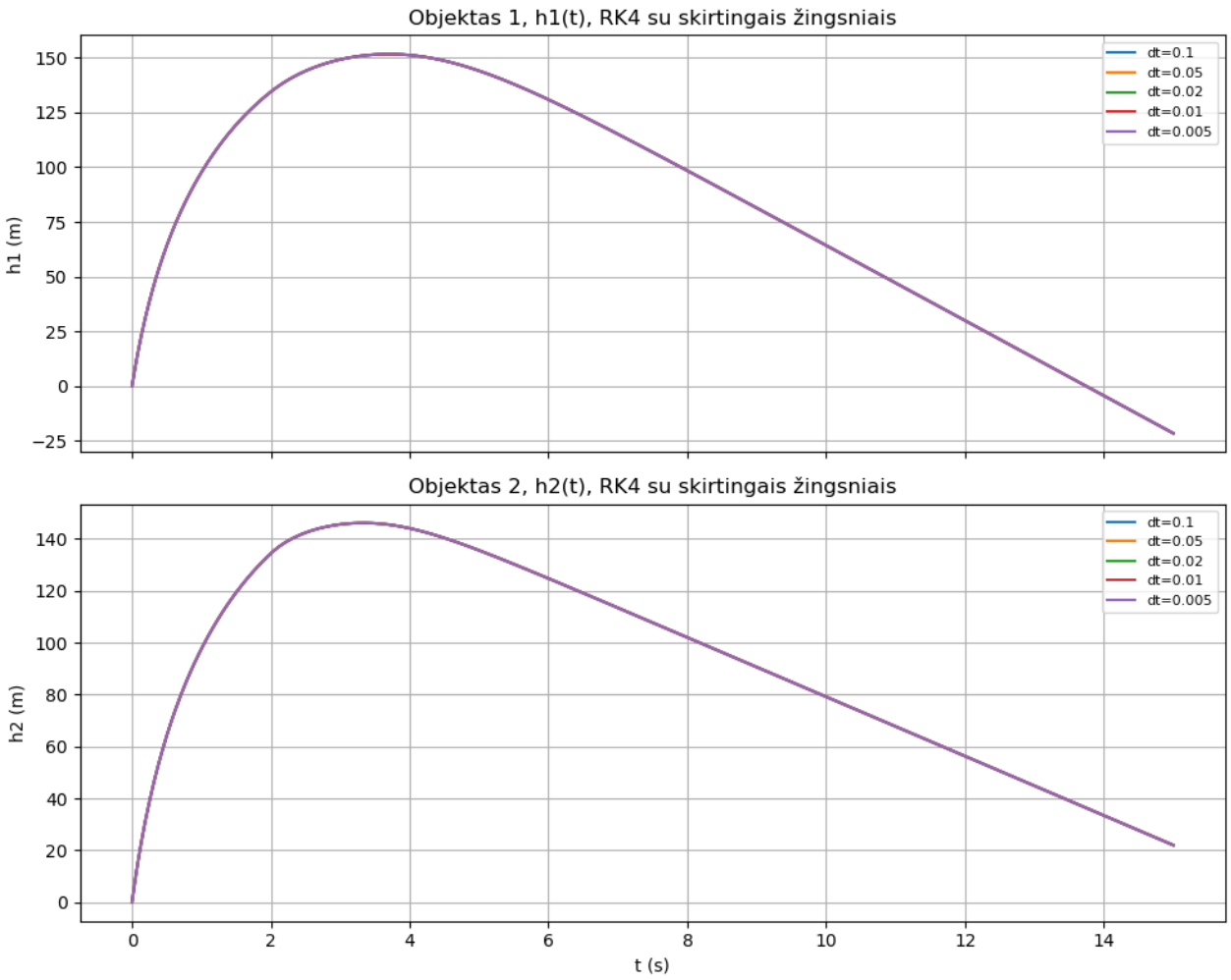
Tokiu būdu gauname tikslesnį aukščiausio taško laiką ir aukštį, nei vien tik imdami artimiausią tinklo tašką.

3. Sprendimas



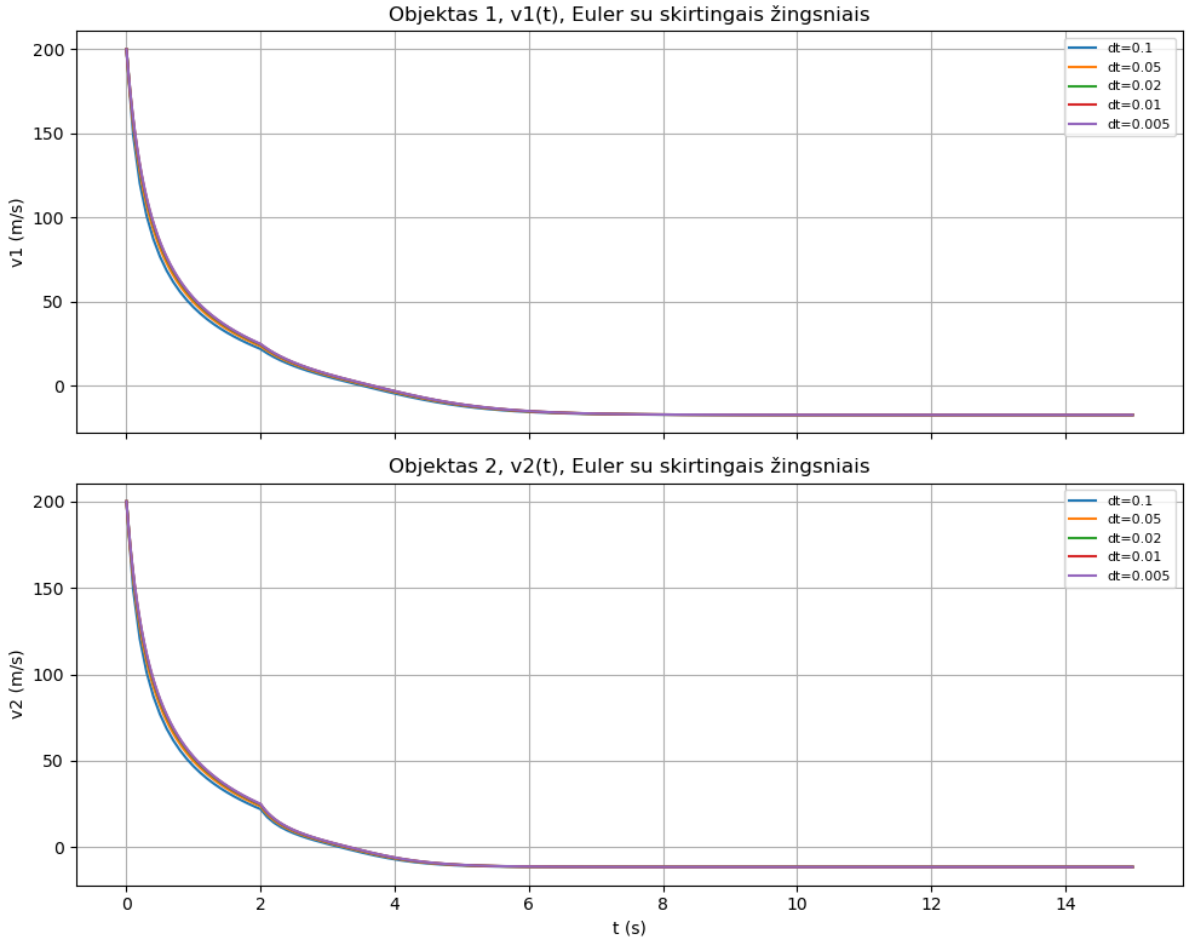
3 pav. RK4 objektų greičiai su skirtingais žingsniais

Grafike pavaizduoti abiejų objektų greičiai, apskaičiuoti RK4 metodu su skirtingais laiko žingsniais. Matyti, kad mažinant žingsnį kreivės beveik sutampa tarpusavyje – tai reiškia, kad metodas yra stabilus ir konverguoja į tą patį sprendinį. Skirtumai tarp žingsnių labiau išryškėja tik ties staigesniais pokyčiais, tačiau ir ten jie išlieka nedideli.



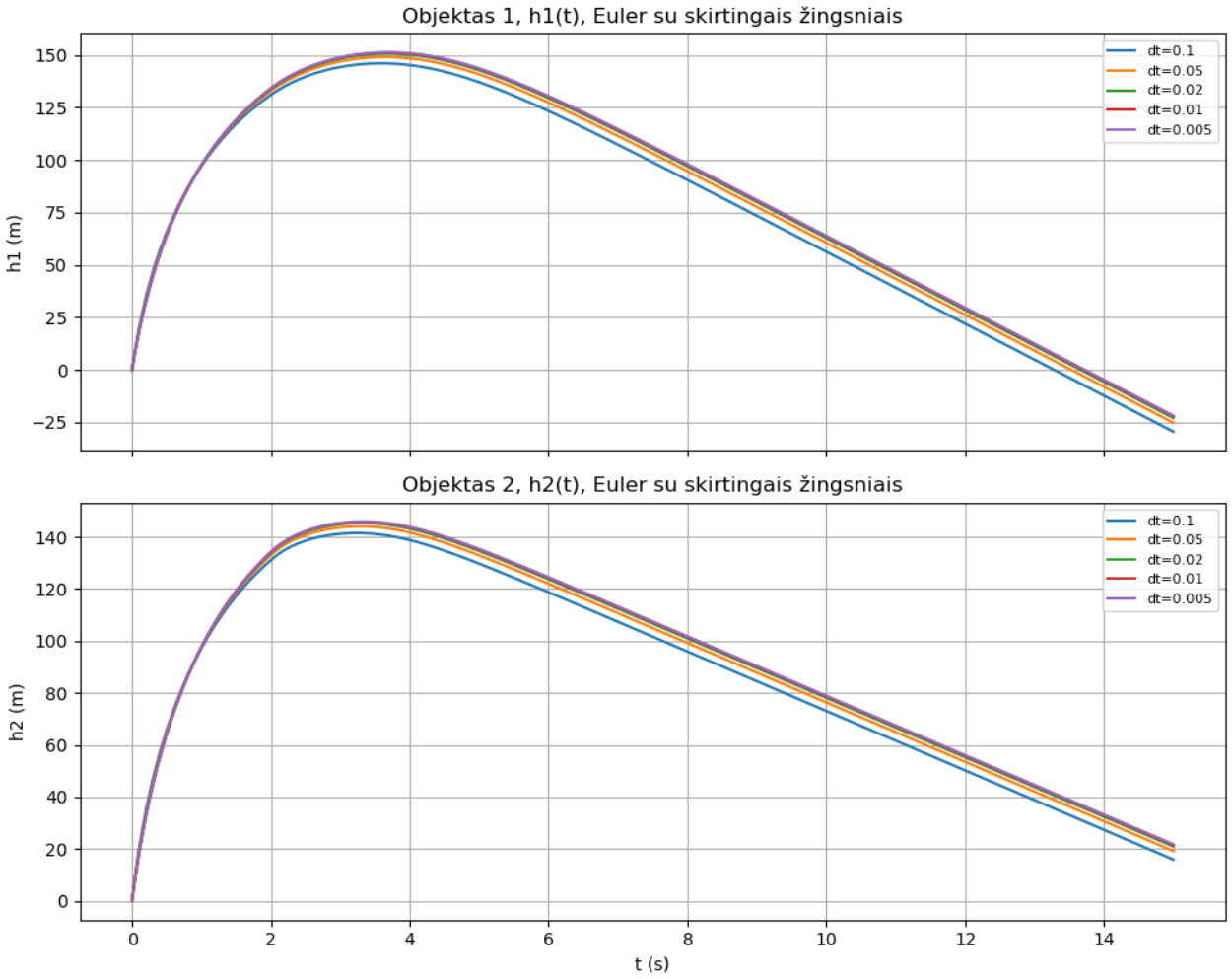
4 pav. RK4 objektų aukščiai su skirtingais žingsniais

Čia matome abiejų objektų aukščio priklausomybę nuo laiko, apskaičiuotą RK4 metodu su keliais skirtingais žingsniais. Kreivės rodo skrydį aukštyn, pasiekimą maksimalios padėties ir kritimą žemyn. Kaip ir greičio atveju, mažesni žingsniai nedaug keičia rezultatą – tai parodo, kad RK4 metodas aukščiui skaičiuoti taip pat yra pakankamai tikslus ir jau su vidutiniu žingsniu duoda labai panašų sprendinį.



5 pav. Eulerio objektų greičiai su skirtingais žingsniais

Grafike pavaizduoti abiejų objektų greičiai, apskaičiuoti Eulerio metodu su tais pačiais laiko žingsniais kaip ir RK4 atveju. Jau galima pastebėti, kad didesni žingsniai sukelia didesnius nuokrypius: kreivės tampa šiurkštesnės, labiau „banguotos“, o jų forma prasčiau atitinka fizinę intuiciją. Tai iliustruoja, jog Eulerio metodas yra jautrus žingsnio dydžiui ir tiksliems rezultatams reikalauja gerokai smulkesnio laiko tinklelio.



6 pav. Eulerio objektų aukščiai su skirtingais žingsniais

Čia pateikti abiejų objektų aukščiai, apskaičiuoti Eulerio metodu su skirtingais žingsniais. Lyginant su RK4 rezultatais, matyti, kad didesni žingsniai lemia labiau iškraipytas trajektorijas: maksimalūs aukščiai ir nusileidimo momentai šiek tiek „slankioja“. Mažinant žingsnį, grafikai artėja prie tikrojo sprendinio formos, bet tam reikia daugiau skaičiavimų, todėl Eulerio metodas šiuo atveju yra mažiau efektyvus.

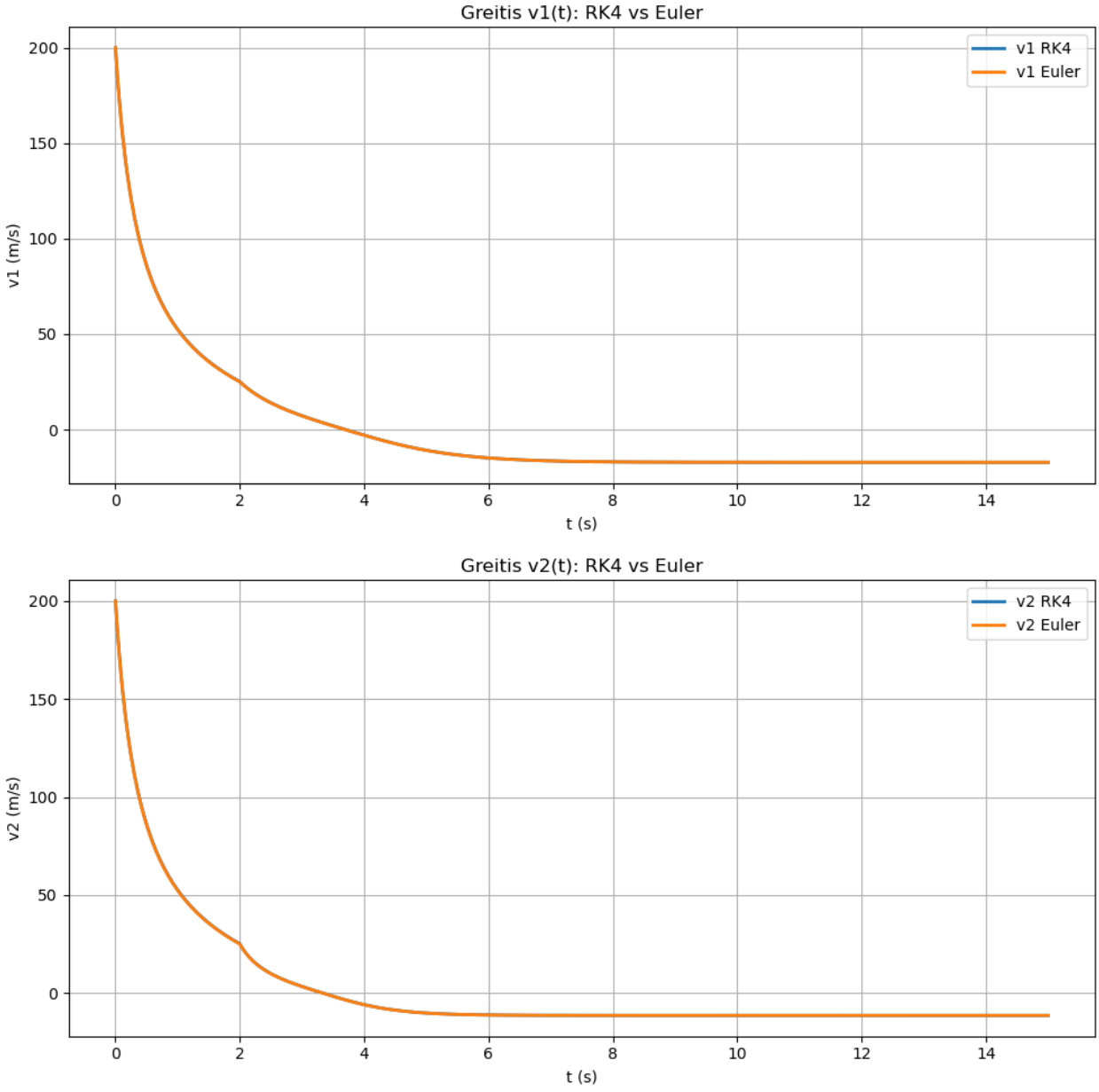
```
== Aukščiausi taškai (RK4, dt = 0.0050) ==
Objektas 1: t = 3.6996 s, h = 151.5982 m
Objektas 2: t = 3.3334 s, h = 146.1566 m

== Aukščiausi taškai (Euler, dt = 0.0050) ==
Objektas 1: t = 3.6919 s, h = 151.3679 m
Objektas 2: t = 3.3264 s, h = 145.9642 m
```

7 pav. Aukščiausi taškai ir laikas iš RK4 bei Eulerio metodų

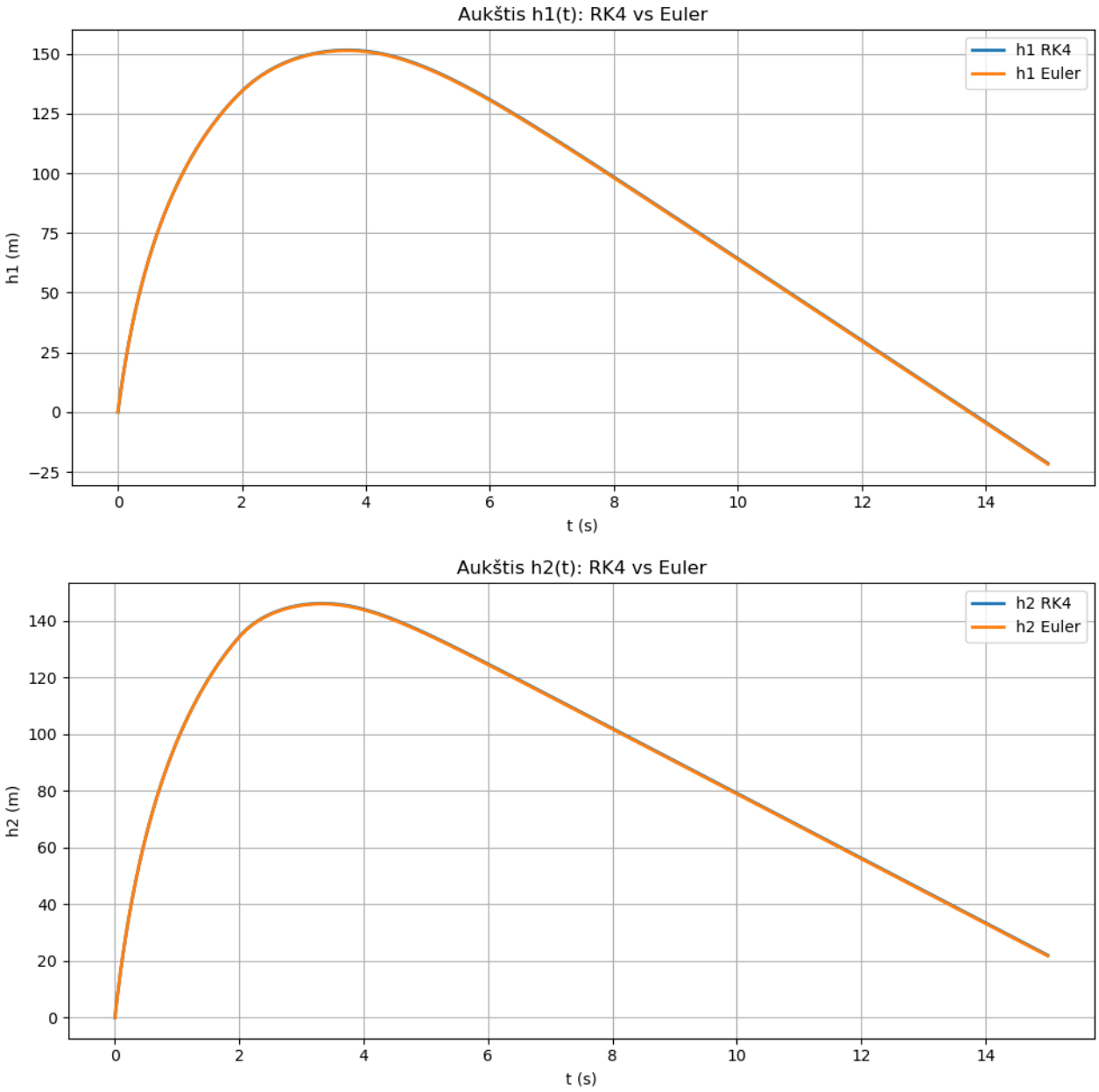
Čia yra palyginami aukščiausi taškai ir laikai, gauti naudojant RK4 ir Eulerio metodus. Matyti, kad RK4 rezultatai labai artimi vienas kitam ir mažai priklauso nuo žingsnio pasirinkimo, o Eulerio

metodo atveju didesni žingsniai duoda labiau išsiskiriančias viršūnes. Tai rodo, kad maksimalų aukštį ir laiką iki jo RK4 metodas apskaičiuoja patikimiau, ypač kai nenorime naudoti labai smulkaus žingsnio.



8 pav. Gautų greičių palyginimai tarp metodų

Čia tiesiogiai palyginti abiejų metodų gauti greičiai laiko atžvilgiu. Nors iš pateiktų grafikų neišeina aiškiai matyti didelių skirtumų, bet atsižvelgus į skaitines vertes yra matyti, kad RK4 kreivė yra „lygesnė“ ir labiau tikėtina fizikiniu požiūriu, o Eulerio kreivė su tuo pačiu žingsniu dažnai šiek tiek nukrypsta – tiek pagal reikšmes, tiek pagal momentus, kada pasiekiami ekstremumai.



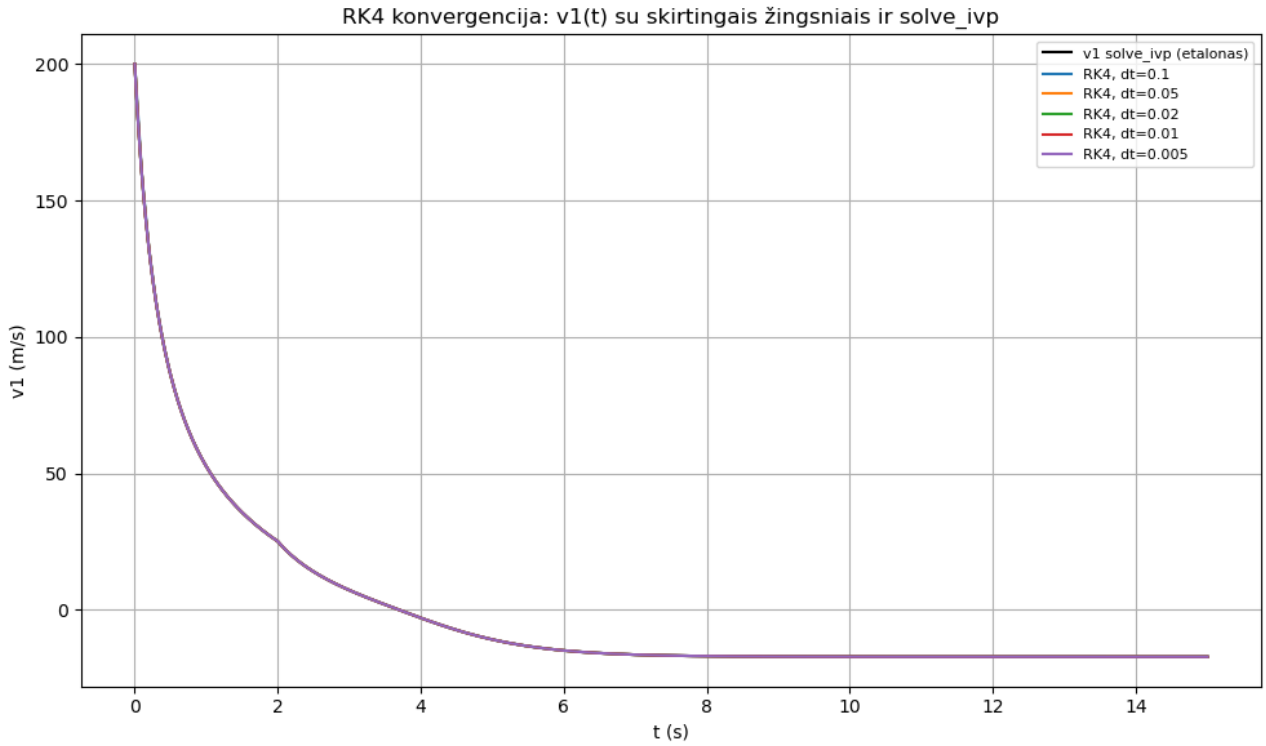
9 pav. Aukščių palyginimai tarp metodų

Šiame grafike lyginami abiejų metodų gauti aukščio grafikai. Matyti, kad RK4 ir Eulerio rezultatai iš pradžių gana panašūs, tačiau ilgėjant laikui Eulerio sprendinys ima labiau skirtis: maksimalūs aukščiai šiek tiek skiriasi, taip pat skiriasi nusileidimo laikas. Tai rodo, kad aukščio skaičiavimui Eulerio metodas labiau „kaupia“ paklaidą, o RK4 išlaiko trajektoriją arčiau tikslaus sprendinio.

```
== Tikslumo palyginimas su solve_ivp (objektas 1, greitis v1) ==
      dt max|v_RK4-v_ivp| max|v_Eu-v_ivp|
  0.1000 2.2694e+00 1.1670e+01
  0.0500 6.6628e-01 5.1557e+00
  0.0200 1.1835e-01 1.9434e+00
  0.0100 3.0688e-02 9.5427e-01
  0.0050 7.8135e-03 4.7292e-01
```

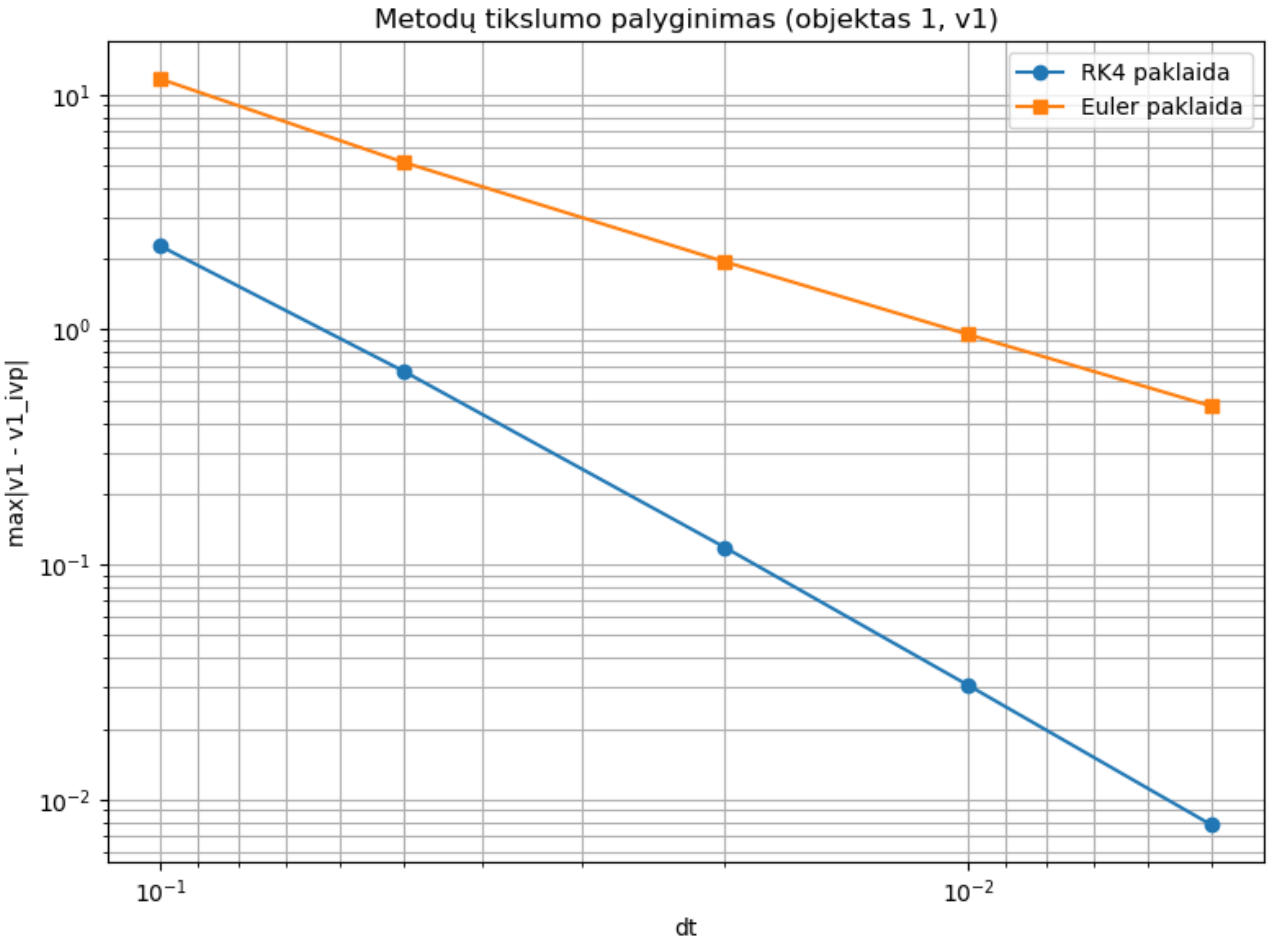
10 pav. Tikslumo palyginimas su solve_ivp

Čia pavaizduotas tikslumo palyginimas su `solve_ivp` funkcija, kuri laikoma etaloniniu sprendiniu. Grafike matyti, kaip RK4 sprendiniai su skirtingais žingsniais priglunda prie `solve_ivp` kreivės. Mažinant žingsnį, RK4 rezultatai praktiškai sutampa su etaloniniu grafiku, o tai patvirtina, kad parinktas modelis ir įgyvendinimas yra teisingi, o RK4 metodas – tinkamai realizuotas.



11 pav. RK4 greičiai su žingsniais ir solve_ivp

Šiame paveiksle detaliau parodomi RK4 metodu gauti greičiai su skirtingais žingsniais ir palyginami su `solve_ivp` rezultatais. Ryškiai matyti, kad net ir naudojant gana „stambius“ žingsnius, RK4 kreivės išlieka arti etaloninio sprendinio, o smulkesni žingsniai kreives priartina praktiškai idealiai. Tai iliustruoja RK4 metodo konvergenciją: didinant skaičiavimo tikslumą, metodas greitai artėja prie „tikro“ sprendinio.



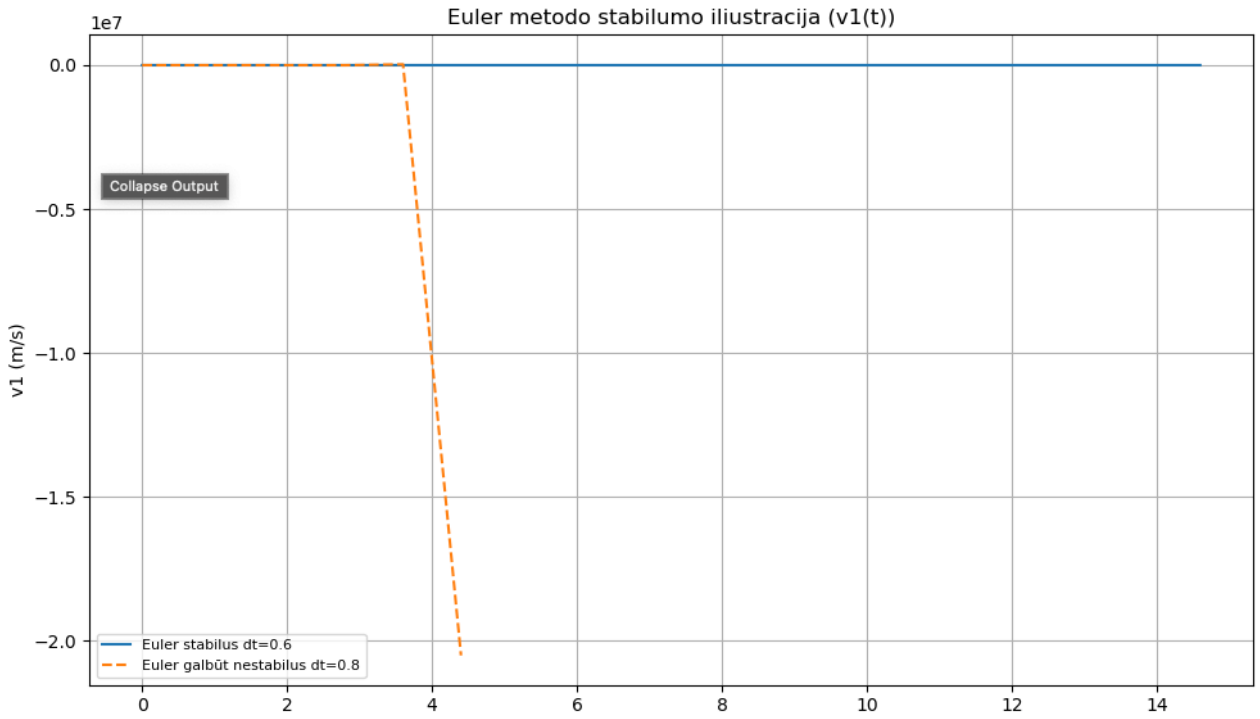
12 pav. Metodų tikslumo palyginimas (pirmo objekto greitis)

Čia pateiktas metodų tikslumo palyginimas, žiūrint į pirmo objekto greitį. Grafikas rodo paklaidą priklausomai nuo laiko žingsnio – tiek RK4, tiek Eulerio metodams. Linijos aiškiai rodo, kad Eulerio metodui paklaida krenta daug lėčiau, todėl jam reikia labai mažo žingsnio, kad pasiektų priimtiną tikslumą. RK4 paklaida mažėja daug sparčiau, tad net su palyginti dideliais žingsniais galima gauti gerą rezultatą.

```
== Stabilumo tyrimas (heuristinis) ==
Euler metodas: didžiausias stabilus dt ≈ 0.6
RK4 metodas  : didžiausias stabilus dt ≈ 0.8
```

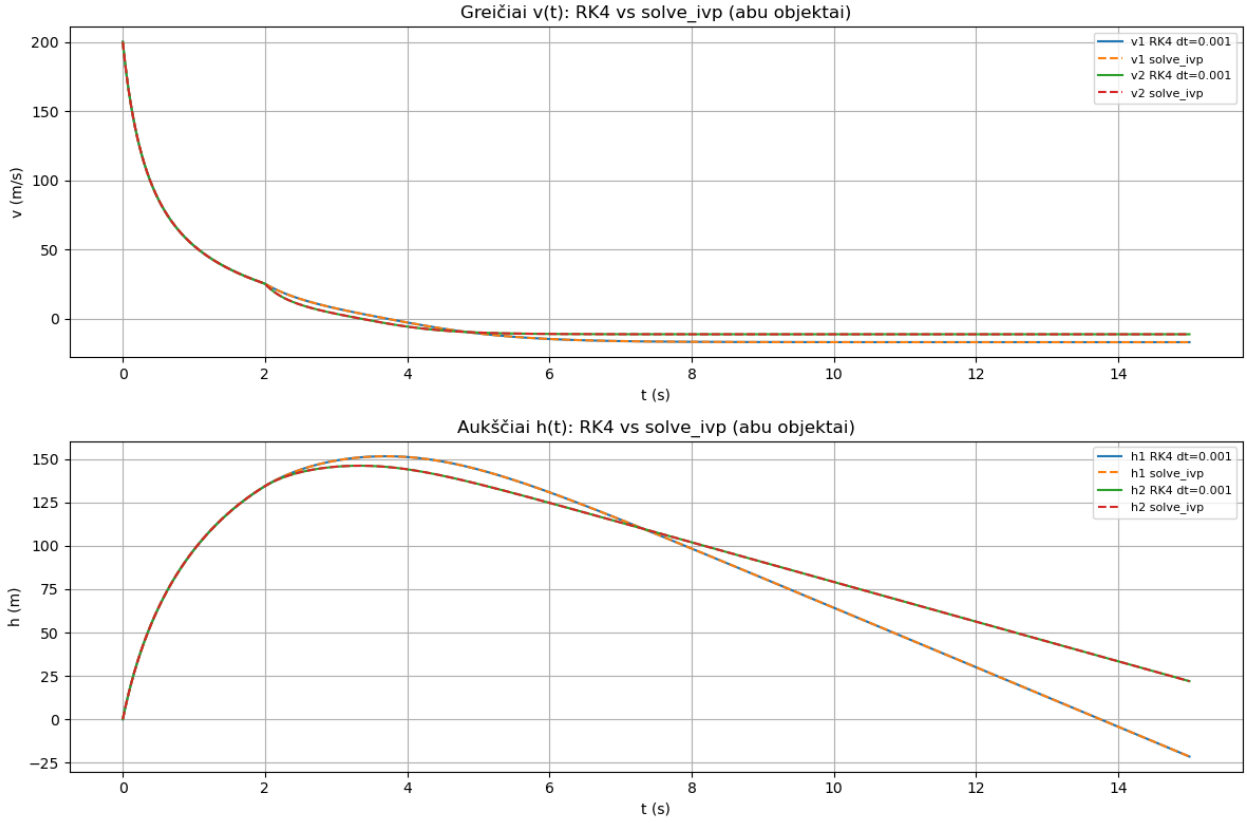
13 pav. Stabilumo tyrimo rezultatai

Šiame paveiksle matomi stabilumo tyrimo rezultatai. Čia pateikta, kokie didžiausi žingsniai dar leidžia gauti stabilų sprendinį Eulerio ir RK4 metodams. Rezultatai parodo, kad Eulerio metodas „sugriūva“ jau esant vidutiniams žingsnio dydžiams – sprendinys tampa nestabilus, reikšmės šokinėja arba išauga iki nerealių dydžių. Tuo tarpu RK4 metodas išlieka stabilus ir su didesniais žingsniais.



14 pav. Eulerio metodo stabilumo iliustracija

Šis grafikas iliustruoja Eulerio metodo stabilumo problemą konkrečiu pavyzdžiu. Pavaizduota, kaip atrodo greičio trajektorija su dar „leistinu“ žingsniu ir su šiek tiek didesniu, jau nestabiliu žingsniu. Esant per dideliam žingsniui, kreivė pradeda elgtis nebe fizikai būdingu būdu – atsiranda dideli svyravimai, nuokrypiai arba iškreiptos, nelogiškos trajektorijos. Tai parodo, kad Eulerio metodą reikia naudoti atsargiai, atidžiai parenkant žingsnį.



15 pav. Gautų RK4 greičių ir aukščių palyginimai su solve_ivp

Paskutiniame grafike kartu pavaizduoti RK4 metodu gauti abiejų objektų greičiai ir aukščiai bei juos atitinkantys `solve_ivp` sprendiniai. Matyti, kad kreivės beveik sutampa – tiek kylant, tiek krintant, tiek ties atsiskyrimo momentu. Tai patvirtina, kad sudarytas fizikinis modelis yra tinkamas, o pasirinktas RK4 metodas, su parinktu žingsniu, duoda labai gerą priartėjimą prie etaloninio sprendinio tiek greičio, tiek aukščio atžvilgiu.

Pagrindinis sprendimo kodas:

```
def accel(v, k, m):
    # |v| reikalingas oro pasipriešinimo modeliui
    v_abs = abs(v)
    # Jeigu greitis tampa absurdiškai didelis (dėl nestabilaus metodo),
    # geriau grąžinti NaN, kad sustabdytume skaičiavimus.
    if v_abs > 1e6:
        return np.nan

    # Pagreitis dv/dt sudarytas iš dviejų jėgų:
    #   1) gravitacija:      -g        (visada veikia žemyn)
    #   2) oro pasipriešinimas: -(k/m) * v * |v|
    # Galutinis pagreitis
    return -g - (k / m) * v * v_abs

def rk4_step(h, v, dt, k, m):
    # sistema: dh/dt = v; dv/dt = accel(v)
    # k1 (pradžios taške)
    k1_h = v
    k1_v = accel(v, k, m)
    # k2 (pusę žingsnio toliau)
    v2 = v + 0.5 * dt * k1_v
    k2_h = v + 0.5 * dt * k1_v
    k2_v = accel(v2, k, m)
    # k3 (vėl per pusę žingsnio)
    v3 = v + 0.5 * dt * k2_v
    k3_h = v + 0.5 * dt * k2_v
    k3_v = accel(v3, k, m)
    # k4 (žingsnio pabaigoje)
    v4 = v + dt * k3_v
    k4_h = v + dt * k3_v
    k4_v = accel(v4, k, m)
    # update (svertinis vidurkis)
    h_new = h + dt/6.0 * (k1_h + 2*k2_h + 2*k3_h + k4_h)
    v_new = v + dt/6.0 * (k1_v + 2*k2_v + 2*k3_v + k4_v)
    return h_new, v_new

def euler_step(h, v, dt, k, m):
    # dh/dt = v  → aukščio pokytis yra tiesiog greitis
    dh = v
    dv = accel(v, k, m)
    # Euler metodas sako:
    #   nauja reikšmė = sena reikšmė + dt * išvestinė
    return h + dt * dh, v + dt * dv

def simulate_two_stage(method='RK4', dt=0.001):
    # laiko tinklelis fazei 1
    t_arr1 = np.arange(0.0, ts + 1e-12, dt)
    N1 = len(t_arr1)
    # pradinės sąlygos
```

```
    h = 0.0
    v = v0
    h_hist1 = np.zeros(N1)
    v_hist1 = np.zeros(N1)
    h_hist1[0] = h
    v_hist1[0] = v
    M = m1 + m2

    # 1 FAZĖ: abu kūnai sujungti į vieną (masė M = m1 + m2, pasipriešinimas
ks)
    for i in range(N1-1):
        if method.upper() == 'RK4':
            h, v = rk4_step(h, v, dt, ks, M)
        else:
            h, v = euler_step(h, v, dt, ks, M)

        # Stabilumo patikra: jeigu skaičiai pavirto NaN/inf → laikome, kad
sprendinys subyrėjo
        if (not np.isfinite(h)) or (not np.isfinite(v)):
            # užpildome likusią trajektorijos dalį NaN
            h_hist1[i+1:] = np.nan
            v_hist1[i+1:] = np.nan
            break

        # Išsaugome naujas reikšmes istorijoje
        h_hist1[i+1] = h
        v_hist1[i+1] = v

    # Fazės 1 pabaigos reikšmės (ties t = ts) – jos taps pradinėmis fazei 2
    h_ts = h_hist1[-1]
    v_ts = v_hist1[-1]

    # 2 FAZĖ: kūnai atsiskiria, ir toliau judame su skirtingomis masėmis
    # ir pasipriešinimo koeficientais
    t_arr2 = np.arange(ts + dt, tmax + 1e-12, dt)
    N2 = len(t_arr2)

    # Pilnas laiko masyvas: pirma fazė + antra fazė vienoje ašyje
    t_full = np.concatenate((t_arr1, t_arr2))
    h1 = np.zeros_like(t_full)
    v1 = np.zeros_like(t_full)
    h2 = np.zeros_like(t_full)
    v2 = np.zeros_like(t_full)

    # pirmos fazės kopija
    h1[:N1] = h_hist1
    v1[:N1] = v_hist1
    h2[:N1] = h_hist1
    v2[:N1] = v_hist1

    # Pradinės sąlygos fazei 2: abiejų objektų padėtis ir greitis tokie patys
ties t = ts
    h1_cur = h_ts
    v1_cur = v_ts
    h2_cur = h_ts
    v2_cur = v_ts

    # Fazės 2 integravimas (nuo ts iki tmax) abiem objektams atskirai
    for j in range(N2):
```

```
        # Žingsnis kiekvienam objektui su atitinkamu k ir m
        if method.upper() == 'RK4':
            h1_cur, v1_cur = rk4_step(h1_cur, v1_cur, dt, k1, m1)
            h2_cur, v2_cur = rk4_step(h2_cur, v2_cur, dt, k2, m2)
        else:
            h1_cur, v1_cur = euler_step(h1_cur, v1_cur, dt, k1, m1)
            h2_cur, v2_cur = euler_step(h2_cur, v2_cur, dt, k2, m2)

        # Indeksas bendrame masyve (po fazės 1)
        idx = N1 + j

        # Jeigu bent vienam objektui atsirado NaN/inf → laikome, kad
sprendinys nestabilus
        if (not np.isfinite(h1_cur)) or (not np.isfinite(v1_cur)) \
           or (not np.isfinite(h2_cur)) or (not np.isfinite(v2_cur)):
            h1[idx:] = np.nan
            v1[idx:] = np.nan
            h2[idx:] = np.nan
            v2[idx:] = np.nan
            break

        # Išsaugome naujas reikšmes bendruose masyvuose
        h1[idx] = h1_cur
        v1[idx] = v1_cur
        h2[idx] = h2_cur
        v2[idx] = v2_cur

    # Grąžiname pilną laiką ir abiejų objektų aukščius bei greičius
    return t_full, h1, h2, v1, v2

def find_peak_time_and_height(t, h, v):
    # Einame per visą greičio vektorių nuo antro elemento
    for i in range(1, len(t)):
        # Tikriname, ar greitis peržengė nulį
        if v[i-1] > 0 and v[i] <= 0:
            dv = v[i] - v[i-1]
            # Jei dv = 0, negalime interpoliuoti – tiesiog laikome viršūnę
v[i-1] taške
            if dv == 0:
                frac = 0.0
            else:
                # Linijinės interpoliacijos koeficientas:
                # frac = dalis intervalo, kur greitis buvo lygus 0
                frac = -v[i-1] / dv
            t_peak = t[i-1] + frac * (t[i] - t[i-1])
            h_peak = h[i-1] + frac * (h[i] - h[i-1])
            # Grąžiname aukščiausio taško laiką ir aukštį
            return t_peak, h_peak
    return np.nan, np.nan
```