KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Skaitiniai metodai ir algoritmai (P170B115)

4 Laboratorinio darbo ataskaita

Atliko:

IFF-1/1 gr. studentas

Arnas Bradauskas

Priėmė:

Andrius Kriščiūnas

Rimantas Barauskas

KAUNAS 2023

TURINYS

1. Užduotis **3**
2. Teorija **3**
3. Rezultatai **4**

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
4	0,5	0,25	100	0,002	2	0,02	0,04	15

2. Teorija

Diferencialinė lygtis:

$$\frac{dv}{dt} = -g - \frac{kv^2 \cdot \mathrm{sgn}(v)}{m}$$

```
def motion(y, m, k):
    dvdt = -g - (k * y[1] ** 2 * np.sign(y[1])) / m
    return [y[1], dvdt]
```

- dv/dt - pagreitis (nurodo, kaip greitis keičiasi per laiką)
- -g - gravitacijos pagreitis (neigiamas, nes visados traukia žemyn)
- k*v^2 - k yra oro pasipriešinimas. v^2 nurodo, jog oro pasipriešinimas didėja proporcingai greičio kvadratui.
- sgn(v) - skirtas užtikrinti, kad pasipriešinimas visada veikia priešinga kryptimi, nei greitis.
- m - objekto masė. Kuo masyvesnis objektas, tuo mažiau jis jaučia pagreitį (antras Niutono dėsnis F=ma)

Eulerio metodas:

- Pradedame taške, kuriame žinome poziciją ir greitį.
- Apskaičiuojame pagreitį tame taške.
- Pajudame mažą žingsnį, koreguojame greitį ir poziciją pagal pagreitį, kurį apskaičiavome.
- Kartojame toliau.

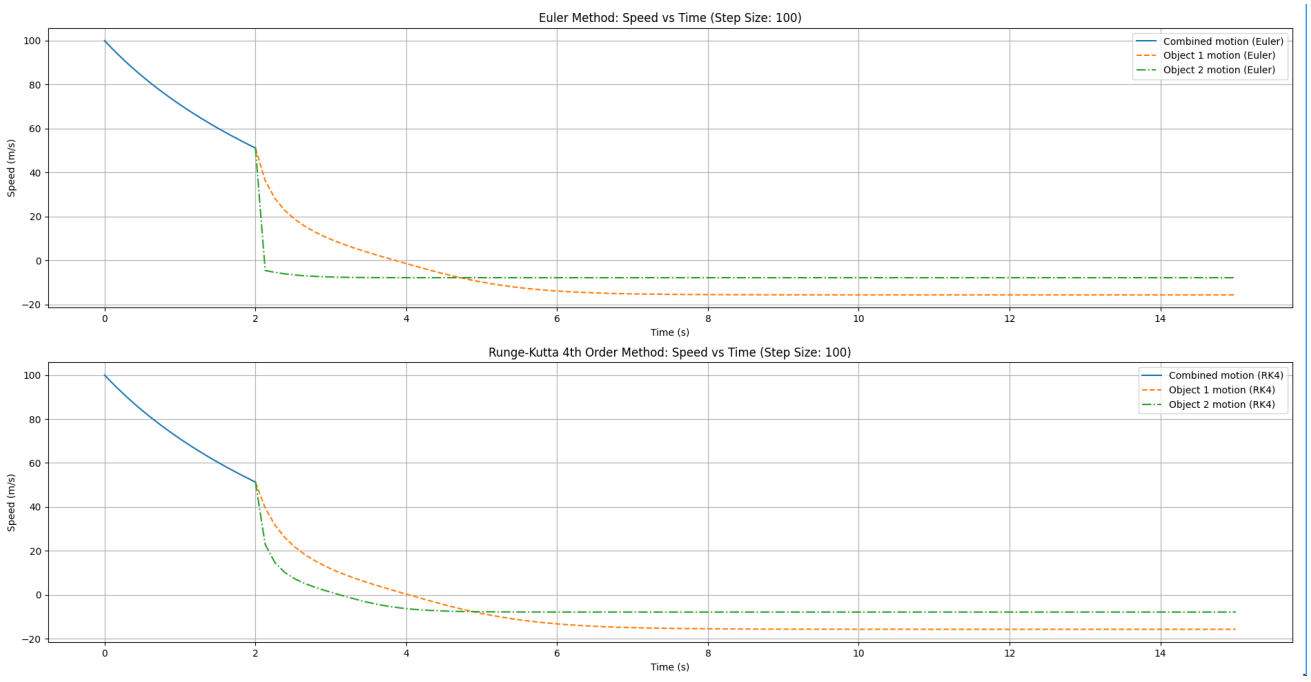
```
# Euler method implementation
def euler_method(f, t_span, y0, args, steps):
    t0, tf = t_span
    h = (tf - t0) / steps
    t_values = np.linspace(t0, tf, steps + 1)
    y_values = np.zeros((len(y0), steps + 1))
    y_values[:, 0] = y0

    for i in range(steps):
        y_values[:, i + 1] = y_values[:, i] + h * np.array(
            f(t_values[i], y_values[:, i], *args)
        )

    return t_values, y_values
```

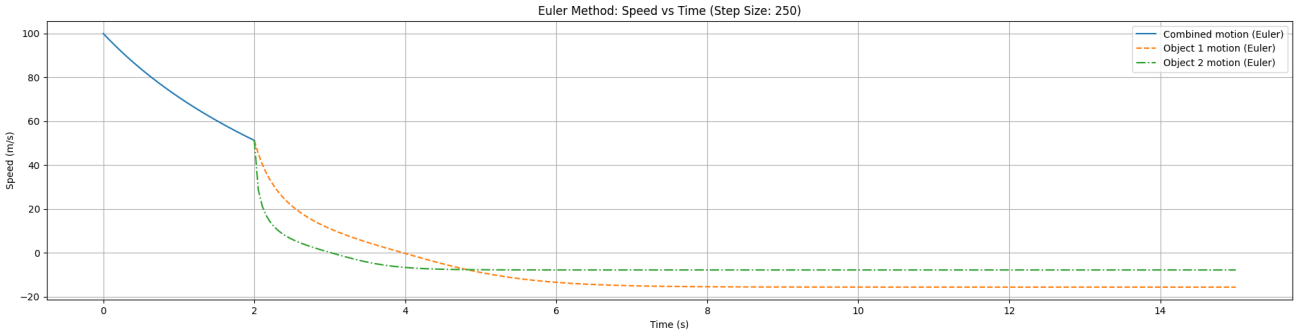
IV eilės Rungės ir Kutos metodas:

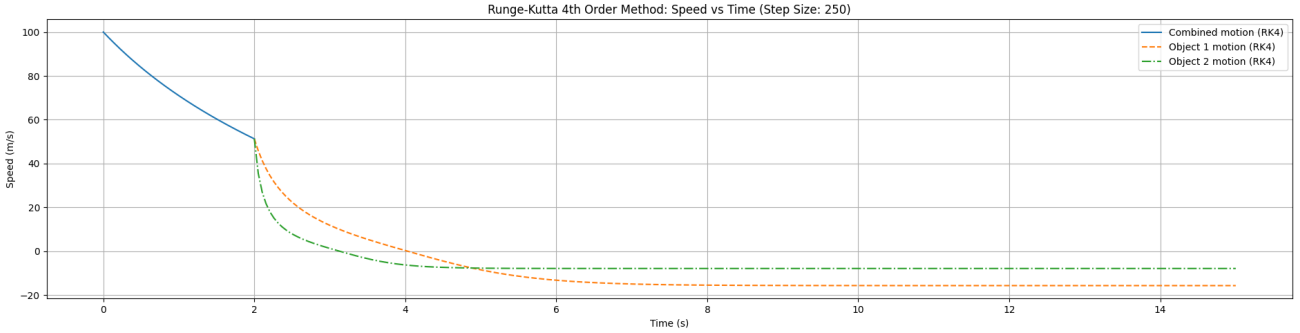
- Apskaičiuojame pagreitį matuojamo intervalo pradžioje (kaip ir Eulerio metode).
- Įvertiname pagreitį dalinai praėjus intervalą naudojant pirmame žingsnyje apskaičiuotą pagreitį.
- Dar kartą įvertiname pagreitį, vėl dalinai praėjus intervalą, bet su pagreičiu iš antro punkto.
- Apskaičiuojame pagreitį gale intervalo naudodami pagreitį iš 3 punkto.
- Apskaičiuojame šių pagreičių svorinį vidurkį.
- Keičiame poziciją ir greitį atsižvelgiant į šį svorinį vidurkį.

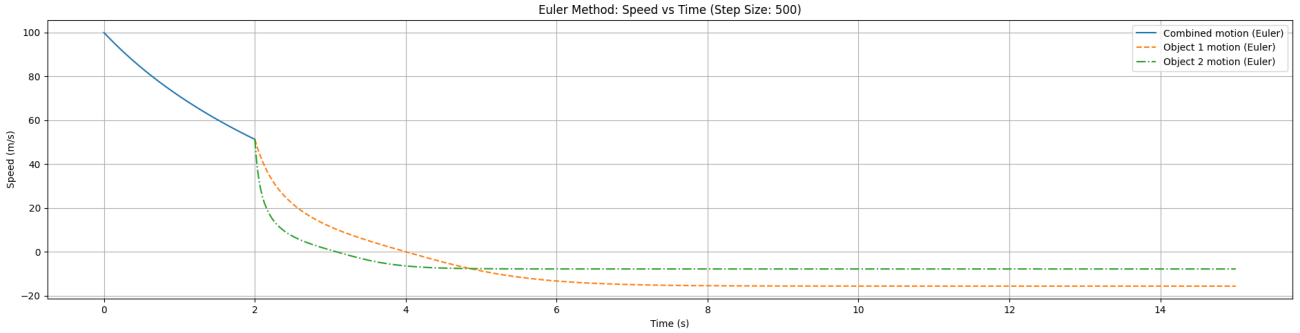
```python
# Runge-Kutta 4th order method implementation
def runge_kutta_4th_order(f, t_span, y0, args, steps):
    t0, tf = t_span
    h = (tf - t0) / steps
    t_values = np.linspace(t0, tf, steps + 1)
    y_values = np.zeros((len(y0), steps + 1))
    y_values[:, 0] = y0

    for i in range(steps):
        k1 = np.array(f(t_values[i], y_values[:, i], *args))
        k2 = np.array(f(t_values[i] + h / 2, y_values[:, i] + h / 2 * k1, *args))
        k3 = np.array(f(t_values[i] + h / 2, y_values[:, i] + h / 2 * k2, *args))
        k4 = np.array(f(t_values[i] + h, y_values[:, i] + h * k3, *args))

        y_values[:, i + 1] = y_values[:, i] + h / 6 * (k1 + 2 * k2 + 2 * k3 + k4)

    return t_values, y_values
```

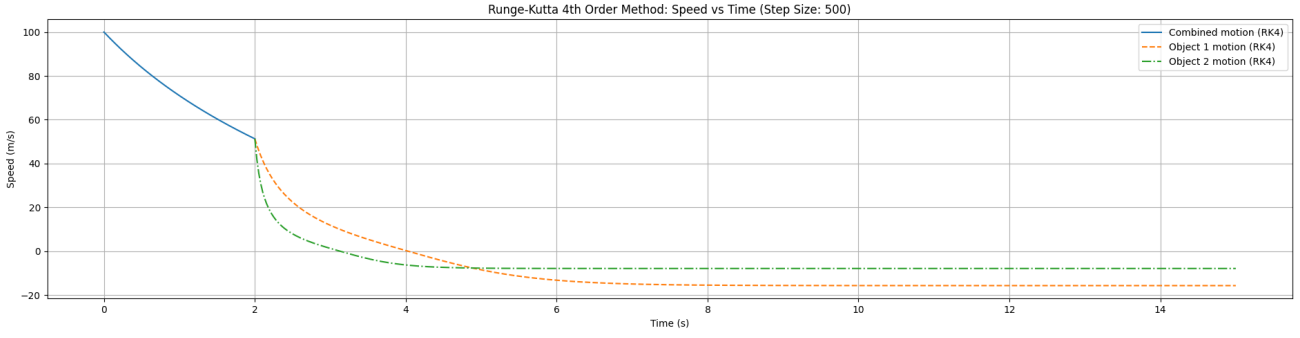
3. Rezultatai

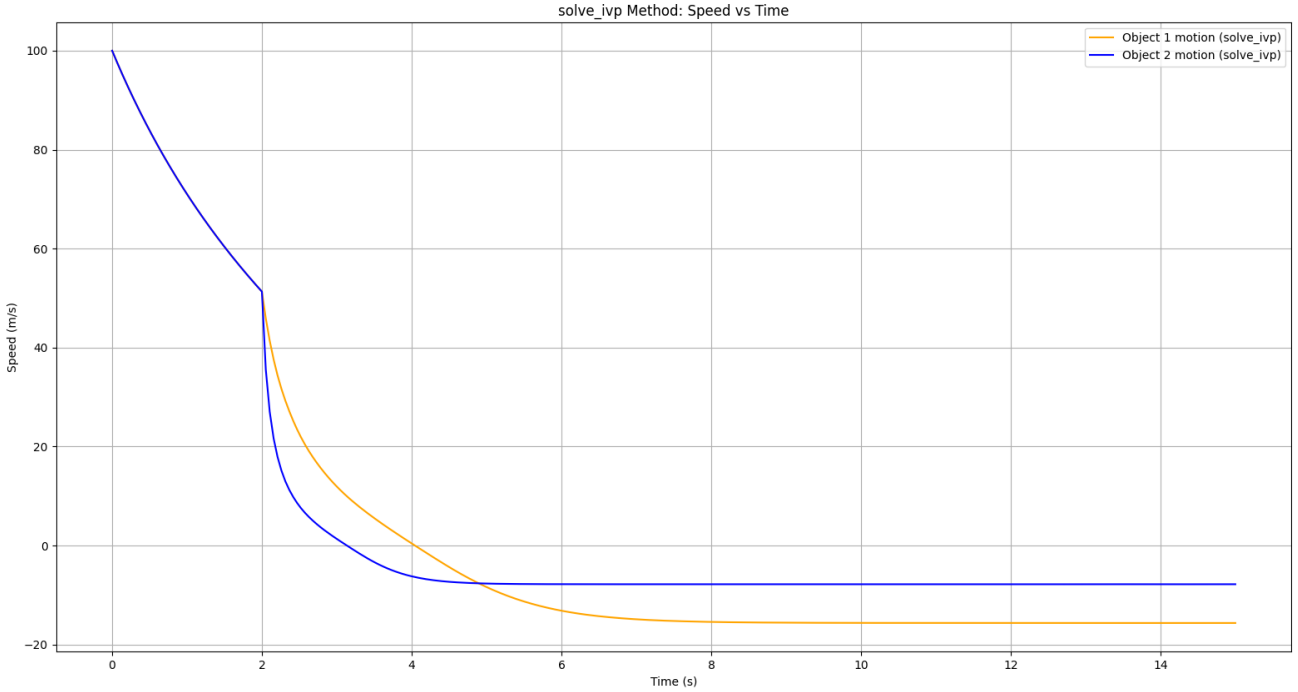



Euler Method: Speed vs Time (Step Size: 250)
Combined motion (Euler)
Object 1 motion (Euler)
Object 2 motion (Euler)
Speed (m/s)
Time (s)


Runge-Kutta 4th Order Method: Speed vs Time (Step Size: 250)
Combined motion (RK4)
Object 1 motion (RK4)
Object 2 motion (RK4)
Speed (m/s)
Time (s)


Euler Method: Speed vs Time (Step Size: 500)
Combined motion (Euler)
Object 1 motion (Euler)
Object 2 motion (Euler)
Speed (m/s)
Time (s)


Runge-Kutta 4th Order Method: Speed vs Time (Step Size: 500)
Combined motion (RK4)
Object 1 motion (RK4)
Object 2 motion (RK4)
Speed (m/s)
Time (s)




```
Step Size: 100
Time highest point (Euler) - Object 1: 3.95 s, Object 2: 2.13 s
Time highest point (RK4) - Object 1: 4.08 s, Object 2: 3.17 s

Step Size: 250
Time highest point (Euler) - Object 1: 3.98 s, Object 2: 3.04 s
Time highest point (RK4) - Object 1: 4.08 s, Object 2: 3.14 s

Step Size: 500
Time highest point (Euler) - Object 1: 4.00 s, Object 2: 3.09 s
Time highest point (RK4) - Object 1: 4.05 s, Object 2: 3.14 s
```