KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

4 Laboratorinis darbas
Nr. 10

Atliko:
IFK-8 grupės stud.
Tomas Odinas

Priėmė
lekt. Andrius Kriščiūnas

KAUNAS, 2020

TURINYS

1. UŽDUOTIS

Paprastųjų diferencialinų lygčių sprendimas

3 Uždavinys variantams 6-10

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
6	100	15	3000	40	0,5	10
7	70	15	4000	40	0,1	5
8	50	15	3500	35	0,1	7
9	125	25	2000	20	0,5	10
10	120	10	2500	25	0,25	10

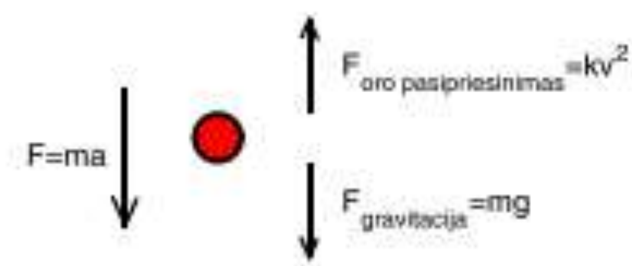
Pav. 1 Užduoties sąlyga

2. PAGRINDINĖ DALIS

Nagrinėsime krentančio kūno greitį. Pav. 2 pavaizduota nagrinėjamos sistemos schema. Remiantis Niutono dinamikos dėsniais sudaroma lygtis $F = F_{gravitacija} - F_{oro\ pasipriešinimas}$.

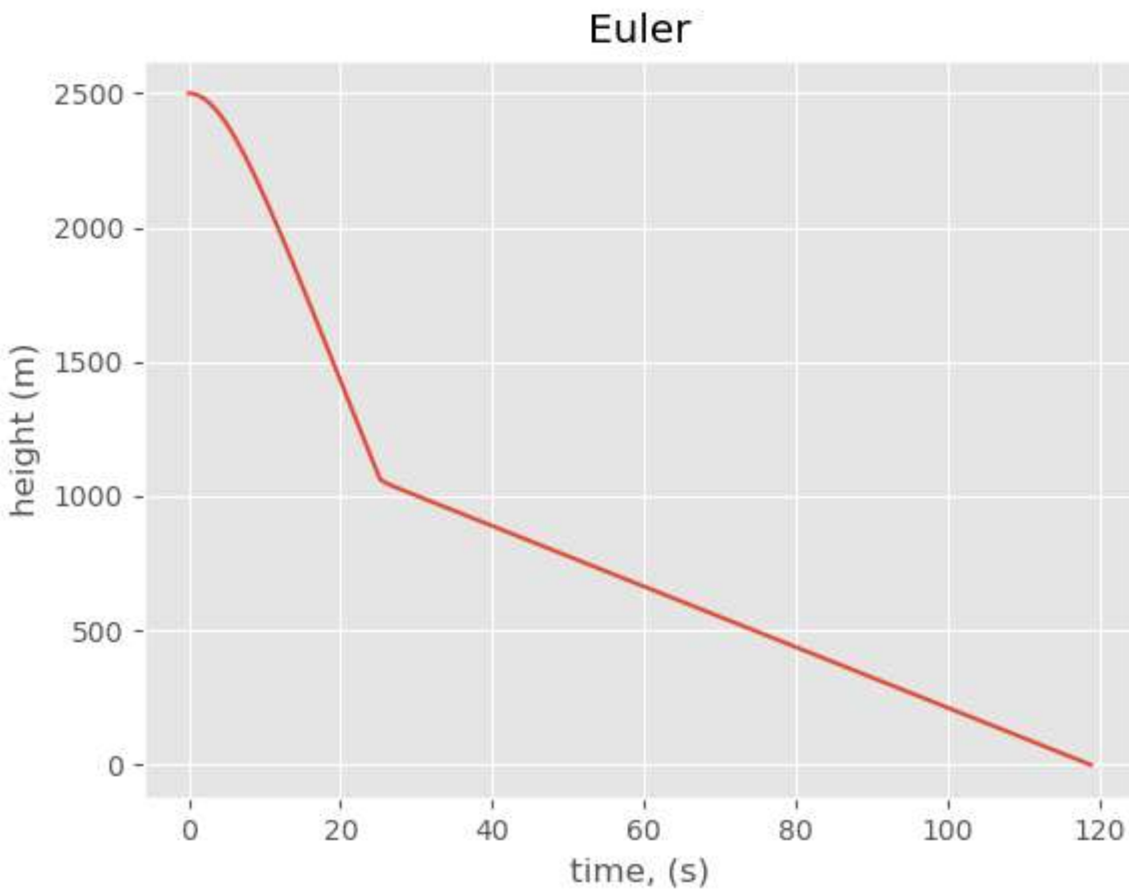
Iš šios lygties seka, jog spręsime diferencialinę lygtį $\frac{dv}{dt} = \frac{mg-kv^2}{m}$. Čia g-laisvojo kritimo pagreitis, k-oro pasipriešinimo koeficientas (išsiskleidus parašiutui jis pakinta), v – krentančio kūno greitis, - m bendra krentančio kūno masė (parašiutininko masė + jo įrangos masė). Atsižvelgiant į tai, jog pasipriešinimo koeficientas keisis – bendrasis lygties pavidalas atrodo taip:

$$\frac{dv}{dt} = \begin{cases} \dfrac{mg - k_1 v^2}{m}, kai\ t < t_g \\ \dfrac{mg - k_2 v^2}{m}, kai\ t \geq t_g \end{cases}$$

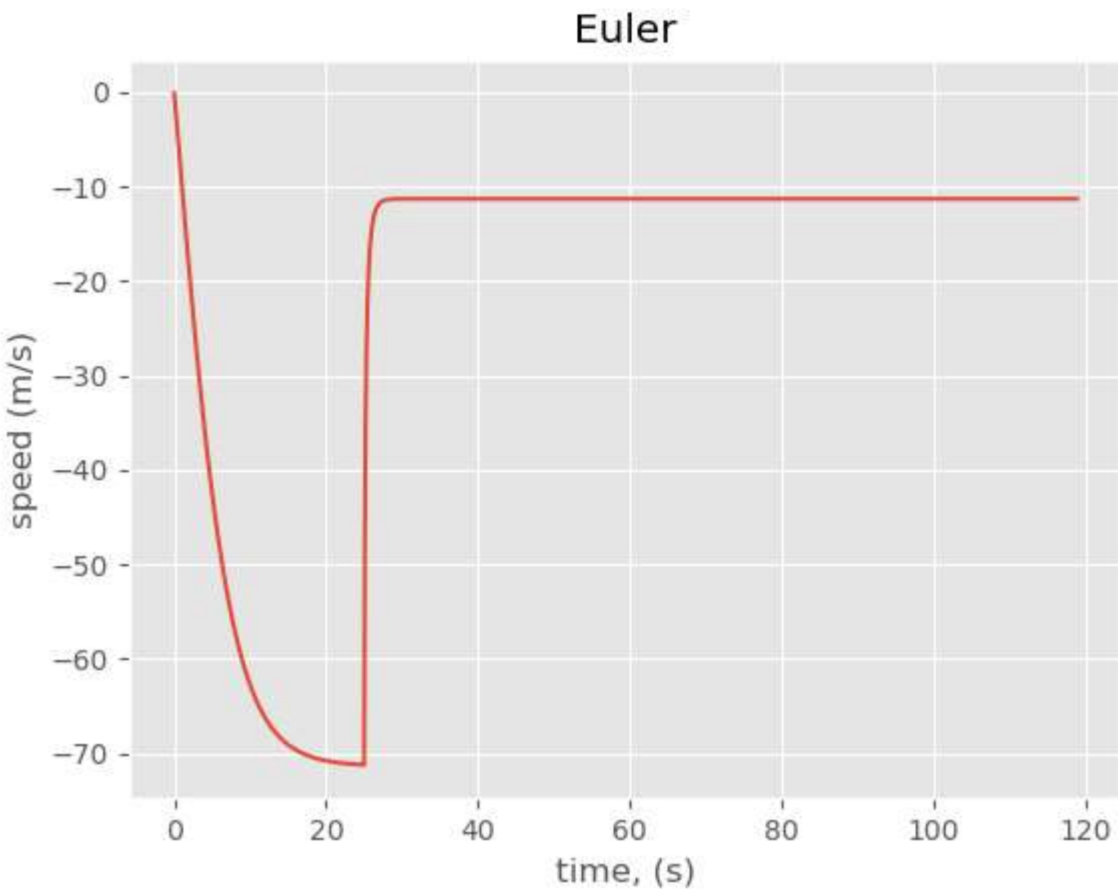


Pav. 2 Nagrinėjama fizikinė sistema

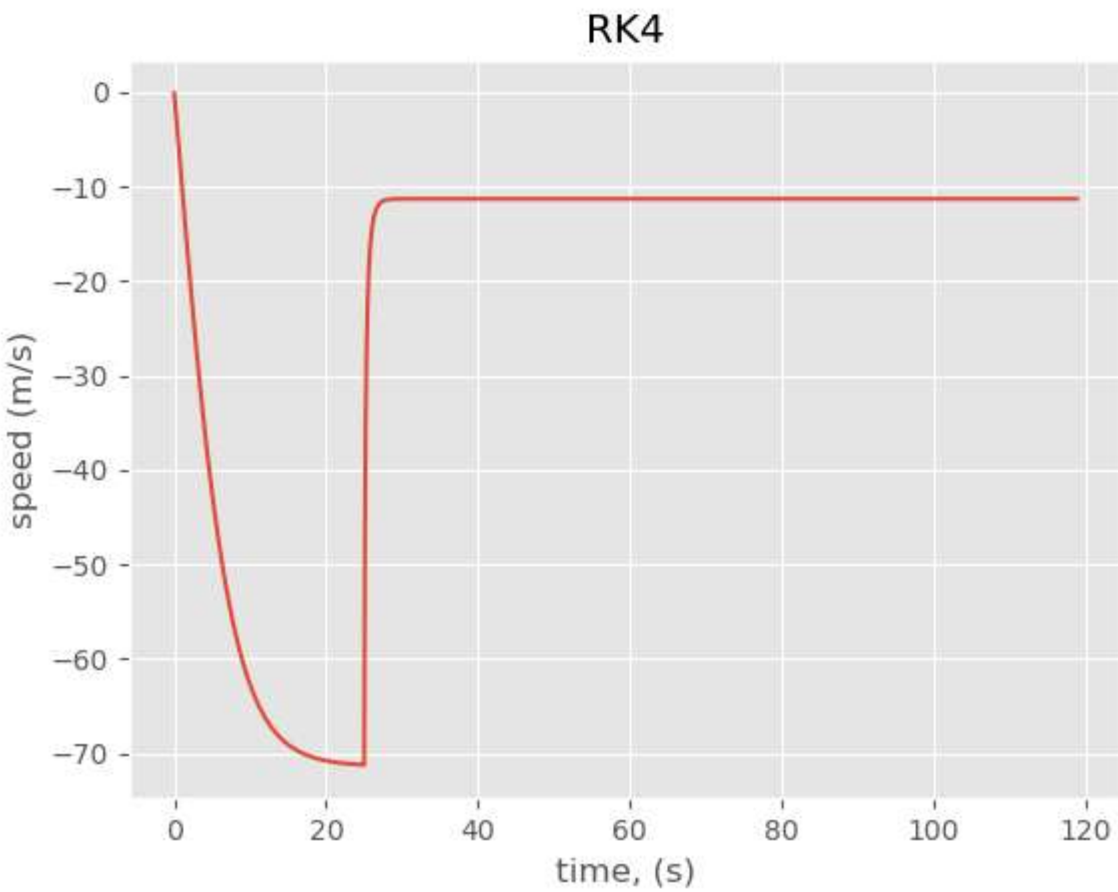
Sudarytą sistemą sprendžiame Eulerio ir 4 eilės Rungės ir Kuto metodais. Žemiau pavaizduoti parašiutininko aukščio ir greičio priklausomybės nuo laiko sprendžiant sistemą abiem metodais.



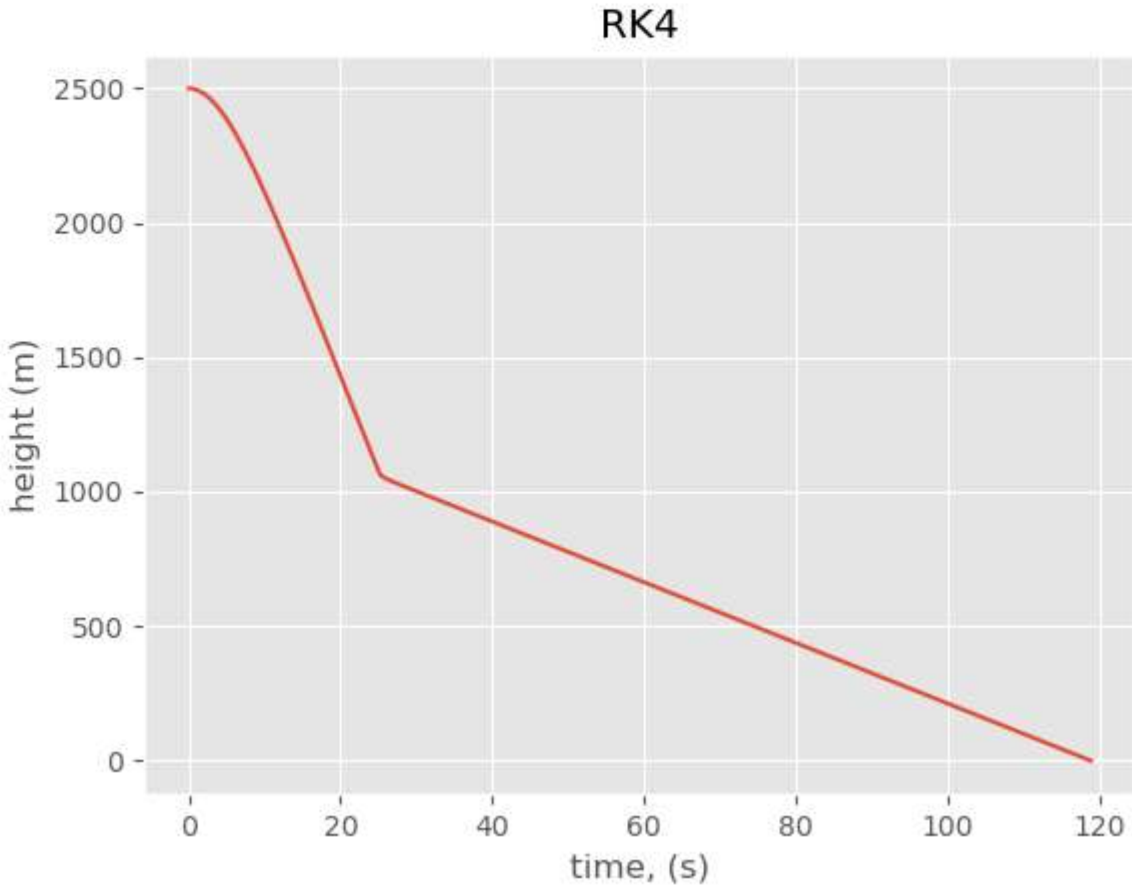
Pav. 3 Aukščio priklausomybė nuo laiko (Eulerio metodas)



Pav. 4 Greičio priklausomybė nuo laiko (Eulerio metodas)

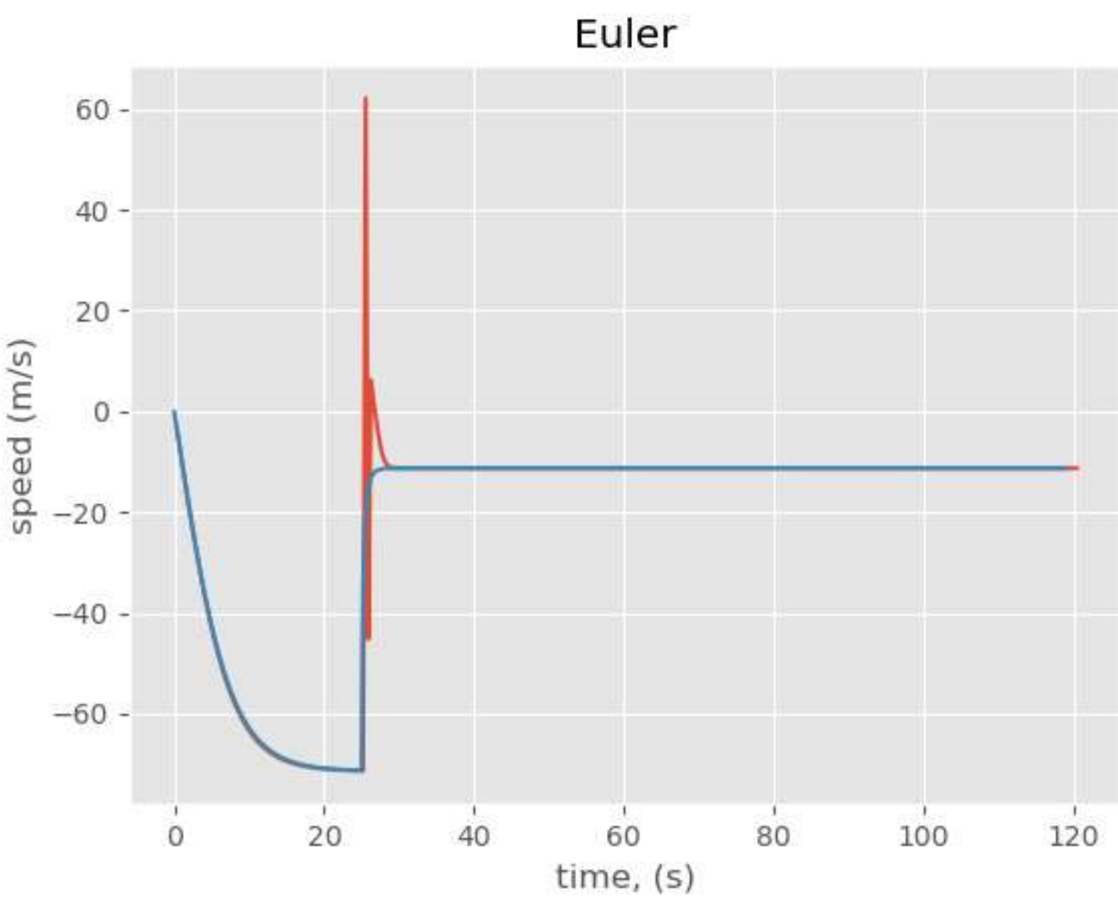


Pav. 5 Aukščio priklausomybė nuo laiko (4 eilės Rungės ir Kuto metodas)



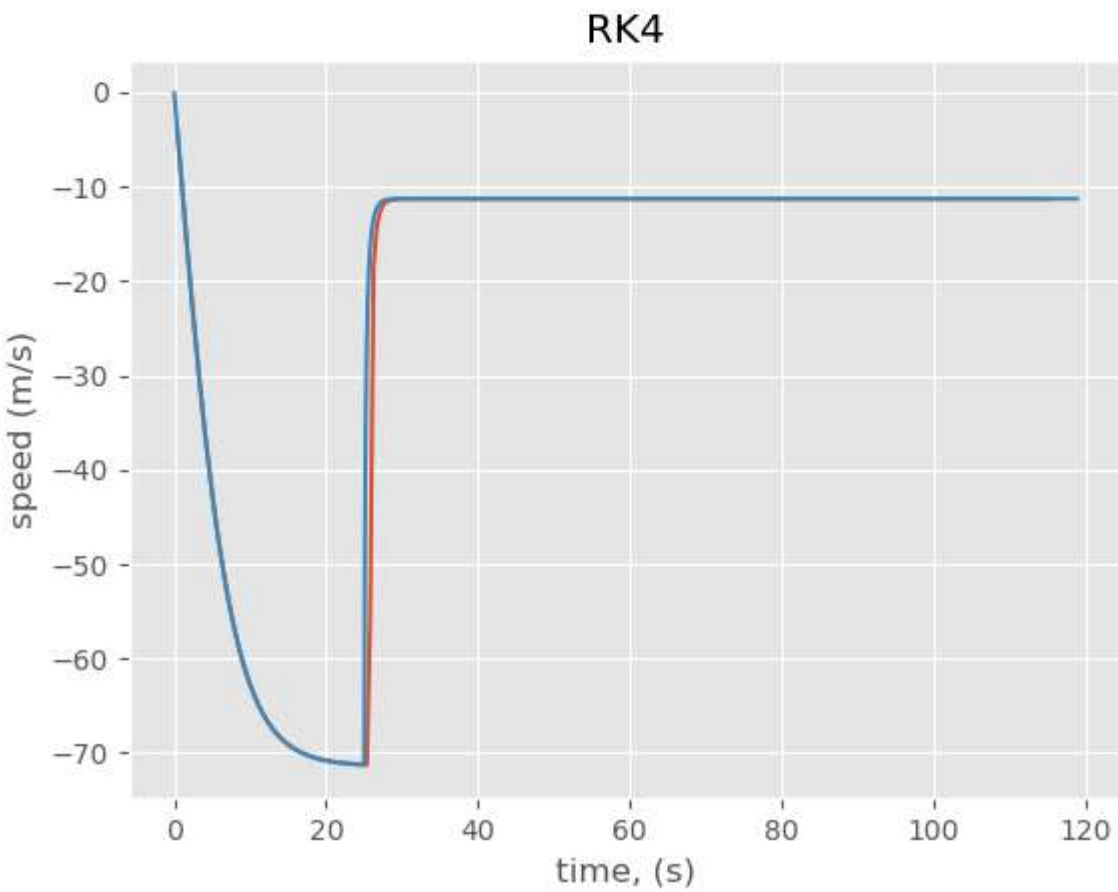
Pav. 6 Greičio priklausomybė nuo laiko (4 eilės Rungės ir Kuto metodas)

Matome, kad tinkamai parinkus žingsnį abu metodai duoda tuos pačius rezultatus. Tačiau galime įsitikinti, jog parinkus didesnį skaičiavimo žingsnį abiejų metodų tikslumas gali nukentėti. Žemiau pateikiami metodų grafikai sistemą sprendžiant parinkus skirtingo dydžio žingsnius. Raudonas grafikas žymi santykinai didelį, o mėlynas mažą žingsnį.



Pav. 7 Greičio priklausomybė nuo laiko (Eulerio metodas)

Matome, kad išsiskleidus parašiutui atsiranda didelės skaičiavimo paklaidos dėl greitai pakitusio oro pasipriešinimo koeficiento.

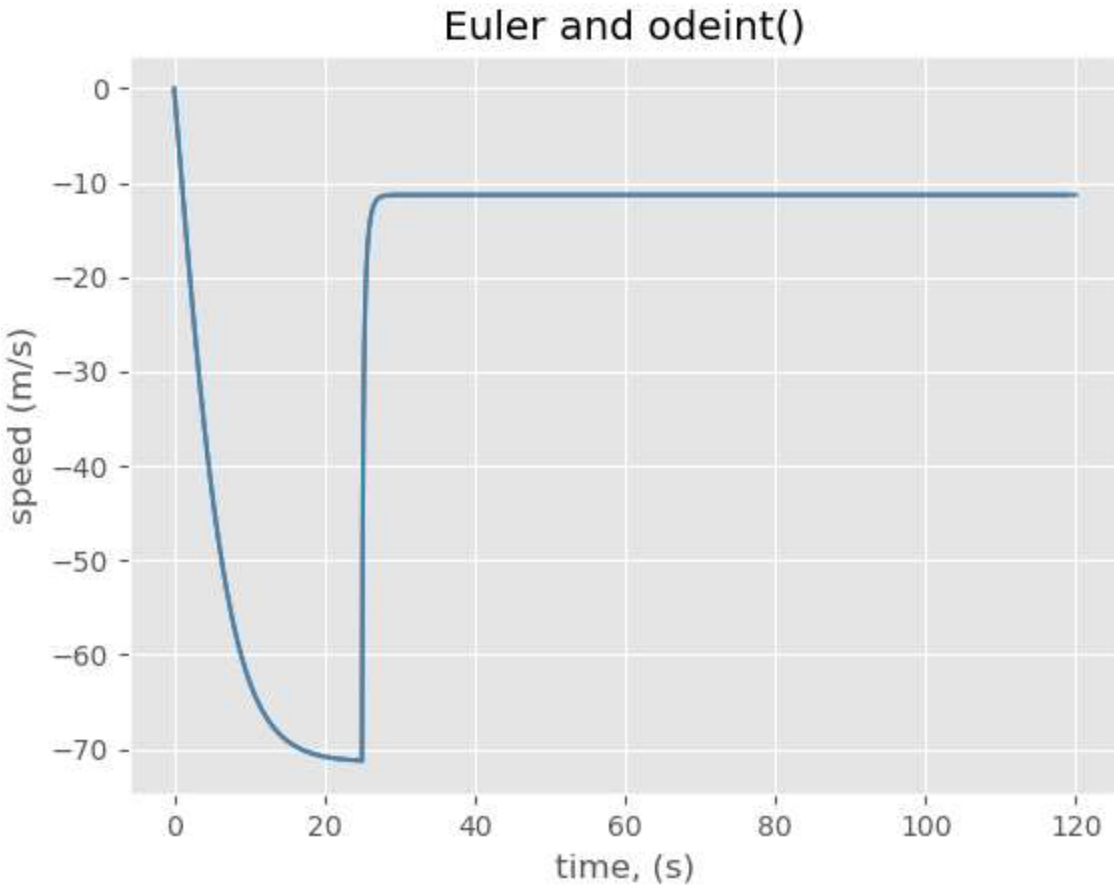


Pav. 8 Greičio priklausomybė nuo laiko (Rungės ir Kuto metodas)

Rungės ir Kuto metodas duoda geresnį rezultatą, tačiau svarbu pabrėžti jog jis atlieka daugiau skaičiavimų kiekviename žingsnyje lyginant su Eulerio metodu.

Parašiutas išsiskleidžia 1074 metrų aukštyje. Parašiutininkas žemę pasiekia po 118 sekundžių, krisdamas 11 m/s greičiu.

Uždavinį dar kartą išsprendžiame pasinaudodami funkcija odenint() iš scipy.integrate paketo. Gautą greičio priklausomybę pavaizduojame grafike. Tame pačiame grafike pavaizduojame ir Eulerio metodo rezultatą. Rezultatai labai artimi, todėl grafikai persidengia. Tai tik patvirtina, jog uždavinys išspręstas teisingai.



Pav. 9 Greičio priklausomybė nuo laiko (Eulerio metodas – raudona, odeint() funkcija - mėlyna)

3. IŠVADOS

Aukštos eilės paprastoji diferencialinė lygtis visada gali būti pakeista pirmos eilės dif. lygčių sistema.

Rungės ir Kuto metodas yra tikslesnis už Eulerio metodą, tačiau atlieka daugiau veiksmų.

4 Eilės Rungės ir Kuto metodo pranašumas prieš kitus aukštesniosios eilės metodus yra tai, jog nereikia analitiškai apskaišiuoti aukštesniųjų eilių išvestinių.

Svarbu parinkti tinkamus metodų žingsnius, jog skaitinės paklaidos būtų minimalios.

4. PRIEDAI

4.1 Programinio kodo fragmentai

Eulerio metodas:

```
def euler():
    t = t0
    v = v0
    h = h0
    m = m1 + m2

    vv = []
    tt = []
    hh = []

    ddd = []

    parachute_deployed = False
    hp = -1
    while h > 0:
        k = k1 if t < tg else k2
        if parachute_deployed == (t < tg):
            parachute_deployed = True
            hp = h

        dv_dt = (m * g + k * v * np.abs(v)) / m
        ddd.append(dv_dt)
        t += dt
        v = v - dt * dv_dt
        h = h + dt * v

        vv.append(v)
        tt.append(t)
        hh.append(h)

    show_results(vv, tt, hh, v, t, hp, 'Euler')
```

4 eilės Rungės ir Kuto metodas:

```
def rk_4():
    t = t0
    v = v0
    h = h0
    m = m1 + m2

    vv = []
    tt = []
    hh = []

    parachute_deployed = False
    hp = -1
    while h > 0:
        k = k1 if t < tg else k2
        if parachute_deployed == (t < tg):
            parachute_deployed = True
            hp = h

        t += dt

        dv_dt1 = (m * g + k * v * np.abs(v)) / m
        v1 = v - dt / 2 * dv_dt1
        dv_dt2 = (m * g + k * v1 * np.abs(v1)) / m
        v2 = v - dt / 2 * dv_dt2
        dv_dt3 = (m * g + k * v2 * np.abs(v2)) / m
        v3 = v - dt * dv_dt3

        dv_dt4 = (m * g + k * v3 * np.abs(v3)) / m
        v = v - dt / 6 * (dv_dt1 + 2 * dv_dt2 + 2 * dv_dt3 + dv_dt4)

        h = h + dt * v

        vv.append(v)
        tt.append(t)
        hh.append(h)

    show_results(vv, tt, hh, v, t, hp, 'RK4')
```

4.2 Nuoroda į programinio kodo repozitoriją

https://github.com/d0ubletr0uble/Skaitiniai-metodai-ir-algoritmai