VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
INFORMATIKOS KATEDRA

Laboratorinis darbas

Diferencialinių lygčių sprendimas skaitiniais metodais

Atliko: 3 kurso 2 grupės studentė
Gabrielė Rinkevičiūtė

Vilnius
2024

Turinys

1. Užduoties sąlyga

Šio laboratorinio užduotis - išspręsti Koši uždavinį

$$\begin{cases} \frac{du}{dx} = e^x \ln(2u + x) + x \\ u(0) = 1, \end{cases} \tag{1}$$

kai x yra intervale $0 \leq x \leq 1$, naudojant du skirtingus skaitinius metodus, juos palyginti ir įvertinti jais gautus rezultatus.

Uždavinys buvo sprendžiamas naudojant Rungės-Kuto 2-pakopį ir 4-pakopį metodus. Algoritmai realizuoti naudojant MATLAB programavimo kalbą (5 skyrius).

2. Rungės-Kuto 2-pakopis metodas

Vienas iš realizuotų skaitinių metodų algoritmų Rungės-Kuto buvo 2-pakopis metodas.

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + \tau a_2, y_n + \tau b_{21} k_1) \end{cases}$$

$$y_{n+1} = y_n + \tau(\sigma_1 k_1 + \sigma_2 k_2)$$

Šiuo atveju $\sigma_1 = \sigma_2 = 0.5$, o $a_2 = \frac{1}{2\sigma_2} = 1$, $b_{21} = \frac{1}{2\sigma_2} = 1$, taigi galutinai metodas atrodo

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + \tau, y_n + \tau k_1) \end{cases}$$

$$y_{n+1} = y_n + \frac{\tau}{2}(k_1 + k_2)$$

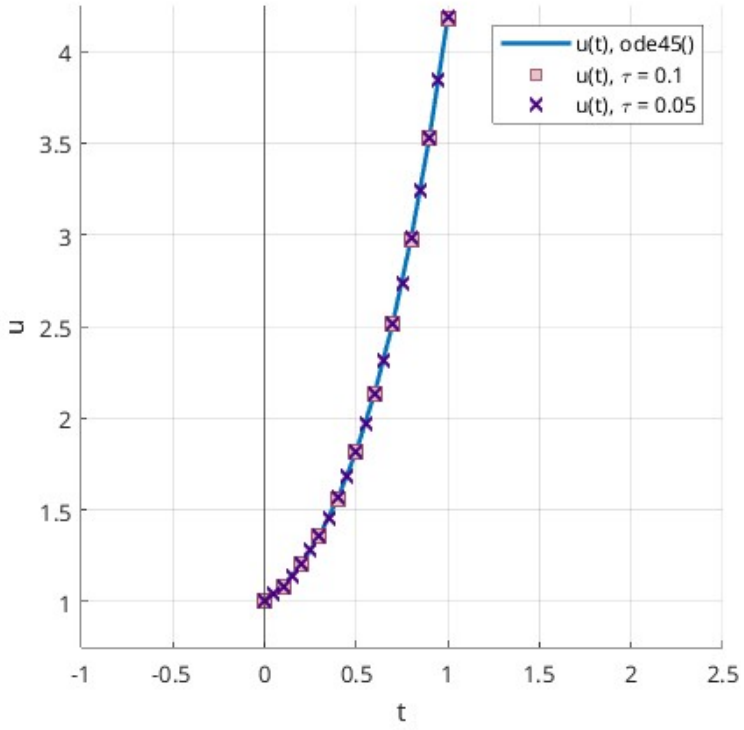
Pritaikius šį metodą aukščiau įvardintai užduočiai (1 lygčių sistema), gauname formules

$$\begin{cases} k_1 = e^{t_n} \ln(2y_n + t_n) + t_n \\ k_2 = e^{t_n + \tau} \ln(2y_n + 2\tau k_1 + t_n + \tau) + t_n + \tau \end{cases} \tag{2}$$

$$y_{n+1} = y_n + \frac{\tau}{2}(k_1 + k_2) \tag{3}$$

2.1. Rezultatai

Žemiau esančiame 1 paveikslėlyje matome Koši uždavinio sprendinį, gautą įgyvendinus aukščiau išvardintas 2 ir 3 formules MATLAB programavimo kalba.

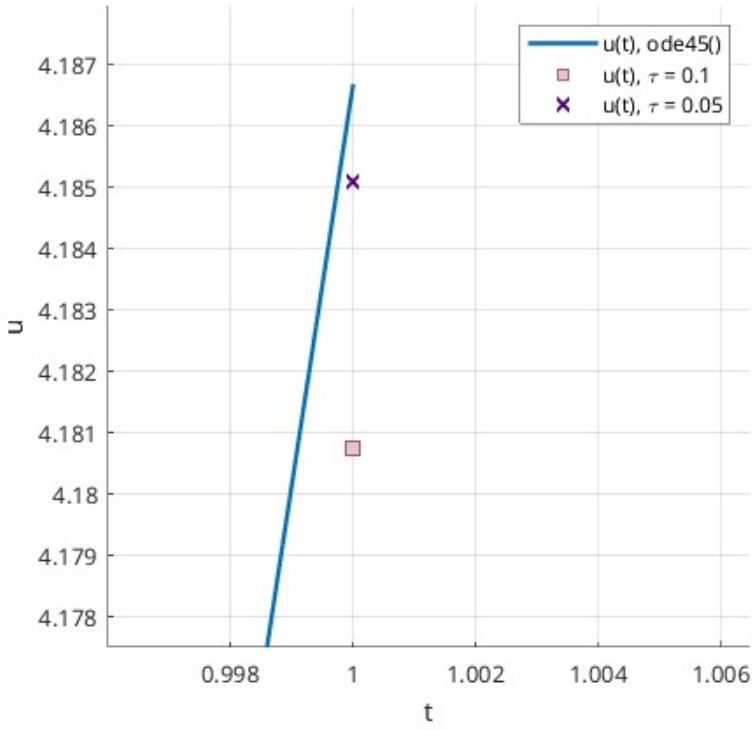


1 pav. Koši uždavinio (1 lygčių sistema) sprendinys, gautas Rungės-Kuto 2-pakopiu metodu

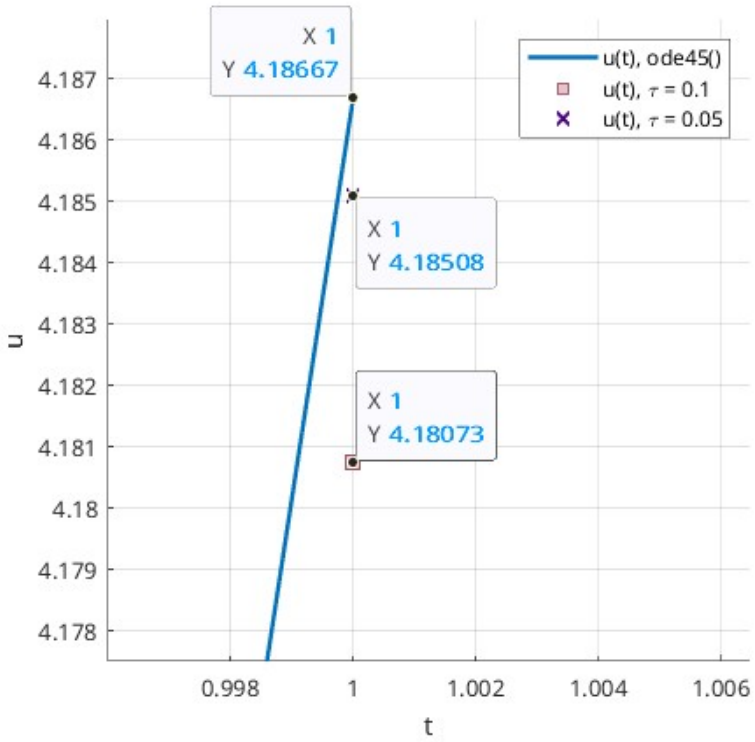
Metodo analizei buvo paimti du žingsnių dydžiai: $\tau = 0.1$ (pažymėta rusvais kvadratėliais) bei $\tau = 0.05$ (pavaizduota violetiniais kryžiukais). Taip pat palyginimui buvo nubrėžta kiek tikslesnė sprendinio kreivė, naudojant MATLAB standartinę funkciją ode45. Iš pateikto paveikslėlio (1 pav.) matome, kad abiem atvejai: tiek kai $\tau = 0.1$, tiek kai $\tau = 0.05$, gautos taškų aibės atitinka tendenciją, kurią atvaizduoja kreivė, nubrėžta naudojant ode45 funkciją.

2.2. Paklaidos įvertinimas

Kiek įdomiau yra įvertinti, kokia yra aukščiau pateiktų rezultatų paklaida. Žemiau pateiktuose 2 ir 3 paveikslėliuose matome, kad gautas rezultatas su žingsniu $\tau = 0.05$ yra artimesnis gautam su ode45, o vadinasi artimesnis realiam rezultatui.



2 pav. Rungės-Kuto 2-pakopiu metodu gauti paskutiniai taškai



3 pav. Rungės-Kuto 2-pakopiu metodu gautų paskutinių taškų etiketės

Pasitelkus Rungės paklaidos įvertinimo metodą

$$|u(T) - y_\tau| \approx \frac{|y_{2\tau} - y_\tau|}{2^P - 1},$$

galime nustatyti, su kokia apytiksliai paklaida buvo apskaičiuotas rezultatas su žingsniu $\tau = 0.05$. Šiuo atveju $P = 2$, kadangi buvo naudotas Rungės-Kuto 2-pakopis metodas. Pritaikius šią formulę gautiems rezultatams, gaunam, kad paklaida yra maždaug $1.449 \cdot 10^{-3}$.

Pasinaudojus skirtumu tarp gautų rezultatų, buvo apskaičiuota, kad paklaida, kai $\tau = 0.1$ yra maždaug $5.799 \cdot 10^{-3}$. Matome, kad pamažinus žingsnį per pusę, stipriai nepasikeitė - liko tos pačios eilės.

3. Rungės-Kuto 4-pakopis metodas

Antrasis realizuotas skaitinis metodas buvo Rungės-Kuto buvo 4-pakopis metodas.

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + \tau a_2, y_n + \tau b_{21} k_1) \\ k_3 = f(t_n + \tau a_3, y_n + \tau(b_{31} k_1 + b_{32} k_2)) \\ k_4 = f(t_n + \tau a_4, y_n + \tau(b_{41} k_1 + b_{42} k_2 + b_{43} k_3)) \end{cases}$$

$$y_{n+1} = y_n + \tau(\sigma_1 k_1 + \sigma_2 k_2 + \sigma_3 k_3 + \sigma_4 k_4)$$

Šio laboratorinio kontekste buvo naudojamas konkrečiai šis jo variantas

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2} k_1) \\ k_3 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2} k_2)) \\ k_4 = f(t_n + \tau, y_n + \tau k_3) \end{cases}$$

$$y_{n+1} = y_n + \frac{\tau}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

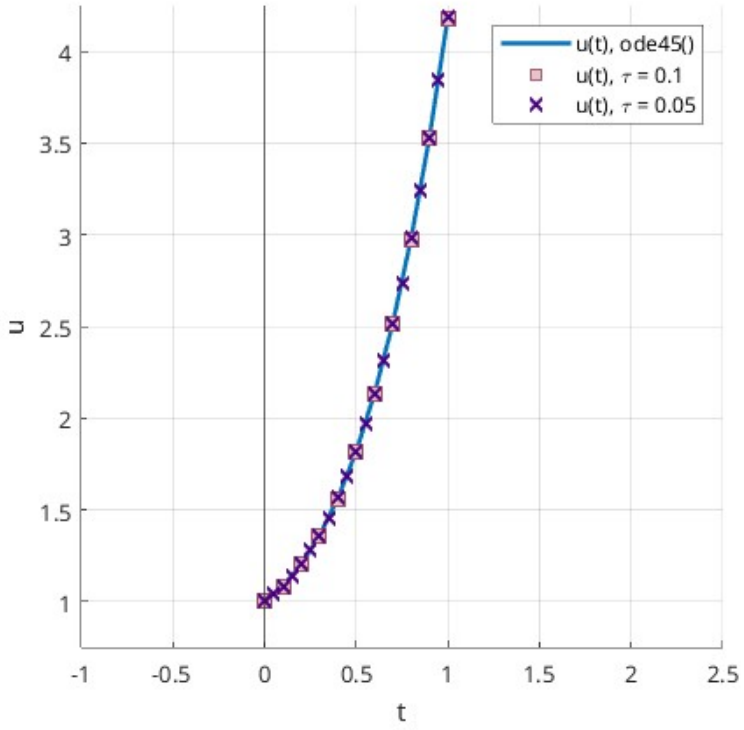
Pritaikius šį metodą konkrečiai šio laboratorinio uždaviniui, gauname

$$\begin{cases} k_1 = e^{t_n} \ln(2y_n + t_n) + t_n \\ k_2 = e^{t_n + \frac{\tau}{2}} \ln(2y_n + \tau k_1 + t_n + \frac{\tau}{2}) + t_n + \frac{\tau}{2} \\ k_3 = e^{t_n + \frac{\tau}{2}} \ln(2y_n + \tau k_2 + t_n + \frac{\tau}{2}) + t_n + \frac{\tau}{2} \\ k_4 = e^{t_n + \tau} \ln(2y_n + 2\tau k_3 + t_n + \tau) + t_n + \tau \end{cases} \tag{4}$$

$$y_{n+1} = y_n + \frac{\tau}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{5}$$

3.1. Rezultatai

Įgyvendinus praeitame skyrelyje gautas 4 ir 5 formules buvo gautas Koši uždavinio (1 lygčių sistema) sprendinys, pateiktas žemiau esančiame 4 paveikslėlyje.

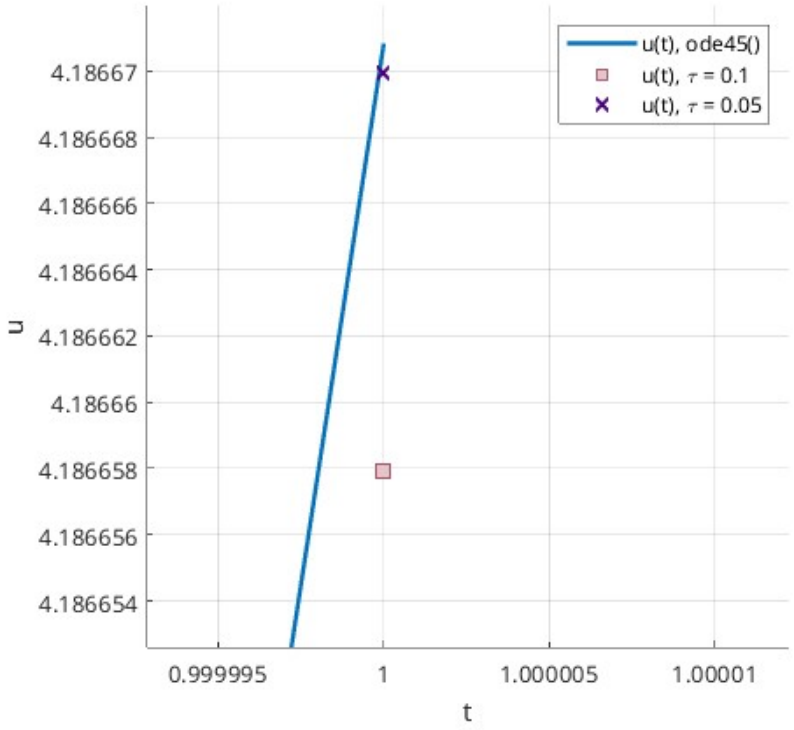


4 pav. Koši uždavinio (1 lygčių sistema) sprendinys, gautas Rungės-Kuto 4-pakopiu metodu

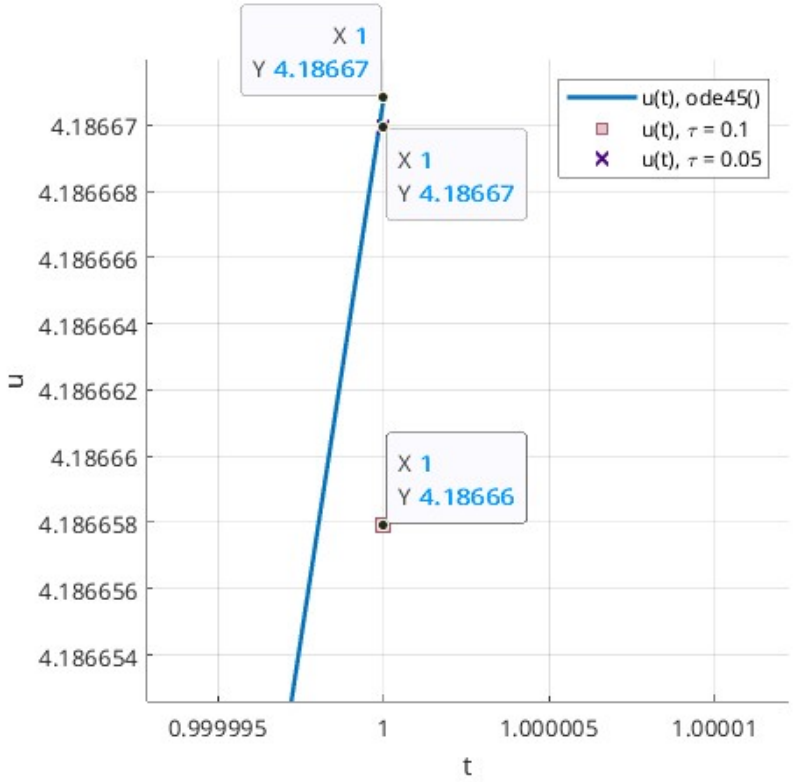
Analogiškai 2 skyriuje nagrinėtam metodui, Rungės-Kuto 4-pakopiui metodui buvo paimti du žingsnių dydžiai: $\tau = 0.1$ (pažymėta rusvais kvadratėliais) bei $\tau = 0.05$ (pavaizduota violetiniais kryžiukais). Taip pat palyginimui buvo nubrėžta kreivė, naudojant funkciją ode45. Iš pateikto 4 paveikslėlio matome, kad tiek su $\tau = 0.1$, tiek su $\tau = 0.05$, gautos taškų aibės atitinka tendenciją, gautą su funkcija ode45.

3.2. Paklaidos įvertinimas

Taip pat kaip ir 2.2 skyrelyje, taip ir čia matome, kad rezultatas su žingsniu $\tau = 0.05$ yra artimesnis nubrėžtai kreivei, negu rezultatas, gautas paėmus žingsnį $\tau 0.1$. Šią įžvalgą vaizdžiai iliustruoja žemiau pateikti 5 ir 6 paveikslėliai.



5 pav. Rungės-Kuto 4-pakopiu metodu gauti paskutiniai taškai



6 pav. Rungės-Kuto 4-pakopiu metodu gautų paskutinių taškų etiketės

Pritaikius gautiems rezultatams jau minėtą Rungės paklaidos įvertinimo metodą, buvo nusta-

tyta, kad rezultato, kai žingsnis yra $\tau = 0.05$, paklaida yra maždaug $8.0086 \cdot 10^{-7}$.

Apskaičiavus paklaidą, kai $\tau = 0.1$, buvo gauta, kad ji yra lygi maždaug $1.08 \cdot 10^{-5}$. Taigi gauname, kad pamažinus žingsnį per pusę, rezultato tikslumas pagerėjo per 2 eiles.

4. Metodų palyginimas

Suvedus į vieną lentelę (1 lentelė) rezultatus, gautus abiem metodais, matome, kad Rungės-Kuto 4-pakopis metodas yra tikslesnis. 2-pakopiu metodu buvo gauta -3 eilės paklaida, tuo tarpu 4-pakopiu metodu buvo gautos -5 ir -7 eilės paklaidos, priklausomai nuo parinkto žingsnio.

	Rungės-Kuto 2-pakopis metodas		Rungės-Kuto 4-pakopis metodas	
Žingsnis (τ)	0.1	0.05	0.1	0.05
Rezultatas	4.18073	4.18508	4.18666	4.18667
Paklaida	$5.799 \cdot 10^{-3}$	$1.449 \cdot 10^{-3}$	$1.08 \cdot 10^{-5}$	$8.008 \cdot 10^{-7}$

1 lentelė. Metodų palyginimo lentelė

Taip pat iš šios lentelės yra pastebima, jog sumažinus žingsnį perpus 4-pakopiame metode yra laimima daugiau tikslumo, tuo tarpu, kai 2-pakopiam metode tikslumas padidėja, bet pasilieka tos pačios eilės. Taigi sprendžiant diferencialines lygtis skaitiniais metodais, gautų rezultatų tikslumas priklauso nuo pasirinkto metodo bei jam pateiktų parametrų.

5. Priedai

5.1. Failas „plot_prep.m“

```
function plot_prep(x_limits, y_limits)
    axis equal; hold on;

    xline(0, 'HandleVisibility', 'off'); hold on;
    yline(0, 'HandleVisibility', 'off'); hold on;

    xlim(x_limits);
    ylim(y_limits);

    xlabel('x'); hold on;
    ylabel('y'); hold on;

    grid on;
    legend();
end
```

5.2. Failas „runge_approx.m“

```
function error = runge_approx(y_2tau, y_tau, p)
    error = abs(y_2tau - y_tau) / (2^p - 1);
end
```

5.3. Failas „runge_kutta_2o_method.m“

```
function results = runge_kutta_2o_method(f, initial_point, tau, T)
    iter_num = T / tau - 1;

    t_n = initial_point(1);
    y_n = initial_point(2);

    t_val = [t_n];
    y_val = [y_n];
    for i = 0:iter_num
        k1 = f(t_n, y_n);
        k2 = f(t_n + tau, y_n + k1*tau);
```

```
        y_n = y_n + (k1 + k2)*tau/2;
        t_n = t_n + tau;

        t_val = [t_val, t_n];
        y_val = [y_val, y_n];
    end

    results = [t_val; y_val];
end
```

5.4. Failas „runge_kutta_2o_analysis.m“

```
%% clearing old values, closing figures
clc, clear, close all
addpath('../utils');
addpath('numerical-methods/');

% defining function
f = @(t, u) exp(t) .* log(2 .* u + t) + t;
t_span = [0, 1];
initial_point = [0, 1]; % u(0) = 1

% solving differential equation using standard MATLAB methods
ode45_results = ode45(f, t_span, initial_point(2));

% solving differential equation using 2nd order Runge-Kutta method
rk_2o_results1 = runge_kutta_2o_method(f, initial_point, 0.1, 1);
rk_2o_results2 = runge_kutta_2o_method(f, initial_point, 0.05, 1);

%% plotting
figure(1);
plot_prep([-1, 2.5], [0.75, 4.25]);
xlabel('t');
ylabel('u');

hold on;

% plotting ode45 results
graph = plot(ode45_results.x, ode45_results.y);
```

```
graph.LineStyle = '-';
graph.LineWidth = 1.8;
graph.DisplayName = "u(t), ode45()";

% plotting 2nd order Runge-Kutta method results
t_values = rk_2o_results1(1, :);
y_values = rk_2o_results1(2, :);
y_2tau = y_values(end); % saving for error evaluation

graph = scatter(t_values, y_values, 60);
graph.Marker = 'square';
graph.MarkerFaceColor = '#e5c5c5';
graph.MarkerEdgeColor = '#A0526D';
graph.DisplayName = "u(t), \tau = 0.1";

t_values = rk_2o_results2(1, :);
y_values = rk_2o_results2(2, :);
y_tau = y_values(end); % saving for error evaluation

graph = scatter(t_values, y_values, 40);
graph.LineWidth = 1.5;
graph.Marker = 'x';
graph.MarkerEdgeColor = '#4B0082';
graph.DisplayName = "u(t), \tau = 0.05";

hold off;

%% evaluating approximation
error = runge_approx(y_2tau, y_tau, 2);
disp("error = " + num2str(error))
```

5.5. Failas „runge_kutta_method.m“

```
function results = runge_kutta_method(f, initial_point, tau, T)
    iter_num = T / tau - 1;

    t_n = initial_point(1);
    y_n = initial_point(2);
```

```
    t_val = [t_n];
    y_val = [y_n];
    for i = 0:iter_num
        k1 = f(t_n, y_n);
        k2 = f(t_n + tau/2, y_n + k1*tau/2);
        k3 = f(t_n + tau/2, y_n + k2*tau/2);
        k4 = f(t_n + tau, y_n + k3*tau);

        y_n = y_n + (k1 + 2*k2 + 2*k3 + k4)*tau/6;
        t_n = t_n + tau;

        t_val = [t_val, t_n];
        y_val = [y_val, y_n];
    end

    results = [t_val; y_val];
end
```

5.6. Failas „runge_kutta_analysis.m“

```
%% clearing old values, closing figures
clc, clear, close all
addpath('../utils');
addpath('numerical-methods/');

% defining function
f = @(t, u) exp(t) .* log(2 .* u + t) + t;
t_span = [0, 1];
initial_point = [0, 1]; % u(0) = 1

% solving differential equation using standard MATLAB methods
ode45_results = ode45(f, t_span, initial_point(2));

% solving differential equation using Runge-Kutta method
rk_results1 = runge_kutta_method(f, initial_point, 0.1, 1);
rk_results2 = runge_kutta_method(f, initial_point, 0.05, 1);

%% plotting
figure(1);
```

```
plot_prep([-1, 2.5], [0.75, 4.25]);
xlabel('t');
ylabel('u');

hold on;

% plotting ode45 results
graph = plot(ode45_results.x, ode45_results.y);
graph.LineStyle = '-';
graph.LineWidth = 1.8;
graph.DisplayName = "u(t), ode45()";

% plotting Runge-Kutta method results
t_values = rk_results1(1, :);
y_values = rk_results1(2, :);
y_2tau = y_values(end); % saving for error evaluation

graph = scatter(t_values, y_values, 60);
graph.Marker = 'square';
graph.MarkerFaceColor = '#e5c5c5';
graph.MarkerEdgeColor = '#A0526D';
graph.DisplayName = "u(t), \tau = 0.1";

t_values = rk_results2(1, :);
y_values = rk_results2(2, :);
y_tau = y_values(end); % saving for error evaluation

graph = scatter(t_values, y_values, 40);
graph.LineWidth = 1.5;
graph.Marker = 'x';
graph.MarkerEdgeColor = '#4B0082';
graph.DisplayName = "u(t), \tau = 0.05";

hold off;

%% evaluating approximation
error = runge_approx(y_2tau, y_tau, 4);
disp("error = " + num2str(error))
```