Laboratorinis darbas #2

Domantas Keturakis

Lapkritis 2024

Užduotys

1. Išspręsi paprastąją pirmos eilės, netiesinę diferencialinę lygtį $\frac{du}{dx} = x^2 \ln(u + x) - x$, su Koši sąlyga $u(0) = u_0$ intervale 0 ≤ x ≤ 1, taikant:

   a. 4-pakopi Rungės-Kuto metodą ir

   b. dvipakopi Rungės-Kuto (σ = 0.5).

   Su pradiniu tašku $u_0 = 1$, žingsniais (τ) 0.1 ir 0.05.

2. Įvertinti paklaidą, intervale $(0, 1]$, Rungės metodu.

Rungės-Kuto 4-kopis metodas

$$
\begin{aligned}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{\tau}{2}, y_n + \frac{\tau}{2} k_1) \\
k_3 &= f(x_n + \frac{\tau}{2}, y_n + \frac{\tau}{2} k_2) \\
k_4 &= f(x_n + \tau, y_n + \tau k_3) \\
y_{n+1} &= y_n + \frac{\tau}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

Rungės-Kuto 2-kopis metodas

$$
\begin{aligned}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{\tau}{2}, y_n + \frac{\tau}{2} k_1) \\
y_{n+1} &= y_n + \frac{\tau}{2}(k_1 + k_2)
\end{aligned}
$$

Sprendimai

Sprendimas (kodas prikabintas 4 psl.), kartu su Python `ScyPy` bibliotekos pateiktu sprendiniu naudojant `solve_ivp` funkciją.

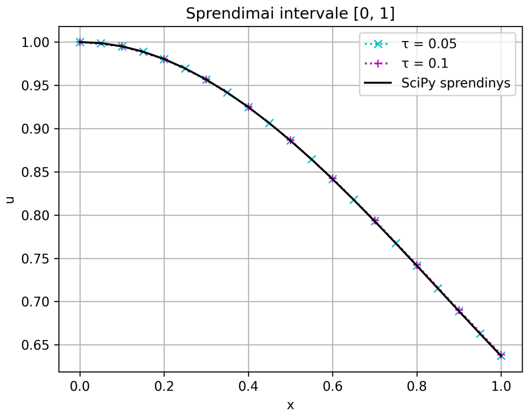

Fig. 1: Sprendimas 2-to laipsnio Rungės Kuto metodu, su žingsniais 0.1 ir 0.05

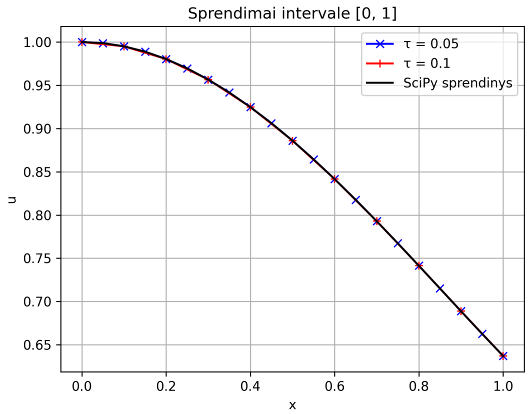

Fig. 2: Sprendimas 4-to laipsnio Rungės Kuto metodu, su žingsniais 0.1 ir 0.05

Palyginimas

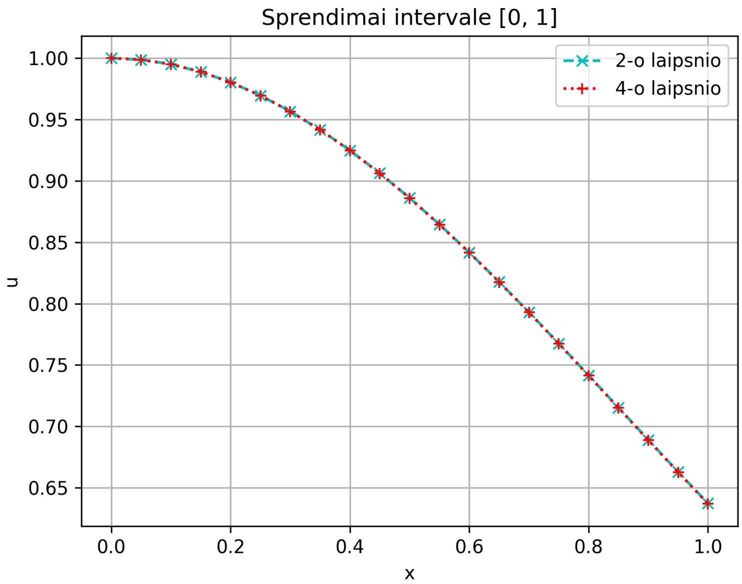

Fig. 3: Sprendimai paskaičiuoti naudojant 2-to ir 4-to laipsnio Rungės Kuto metodus, kai žingsnis τ = 0.05

Paklaidos vertinimas

Paklaida įvertinta Rungės metodu $|u(T) - y_\tau| \approx \dfrac{|y_{2\tau} - y_\tau|}{2^p - 1}$, kur:

y_τ – skaitinis sprendinys taške t = T, apskaičiuotas su žingsniu τ,

$y_{2\tau}$ – skaitinis sprendinys taške t = T, apskaičiuotas su žingsniu 2τ,

p – metodo tikslumo eilė.

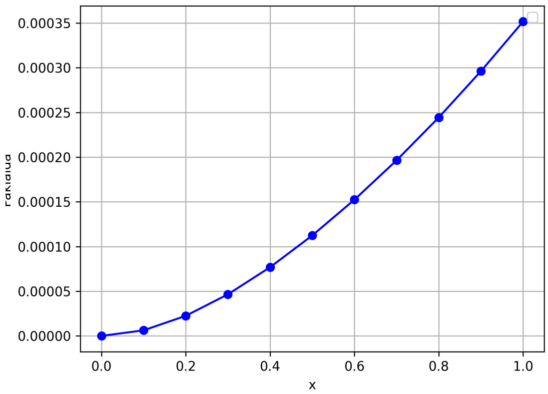

Fig. 4: Paklaidos naudojant 2-to laipsnio Rungės Kuto metodus, su žingsniu τ = 0.05

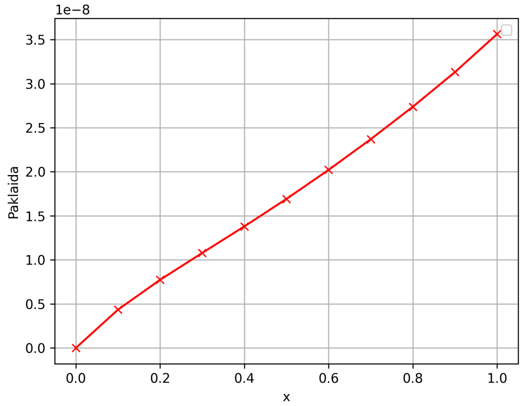

Fig. 5: Paklaidos naudojant 4-to laipsnio Rungės Kuto metodus, su žingsniu τ = 0.05

Priedai

Naudojami įtraukimai:

```py
import numpy as np
import matplotlib.pyplot as plt
from pipe import take_while
from scipy.integrate import solve_ivp
```

Sprendimai

Sprendimas 4-to laipsnio Rungės Kuto metodu:

```py
def RungeKutta4(fn, u0, tau=1e-5, lower_bound = 0, upper_bound=1):
  ys = [u0]
  ts = np.arange(lower_bound, upper_bound + tau, tau)

  for t_n in ts[:-1]:
    y_ = ys[-1]
    k1 = fn(t_n,         y_         )
    k2 = fn(t_n + tau/2, y_ + tau*k1/2)
    k3 = fn(t_n + tau/2, y_ + tau*k2/2)
    k4 = fn(t_n + tau,   y_ + tau*k3  )
    y_ = y_ + tau/6*(k1 + 2*k2 + 2*k3 + k4)
    ys.append(y_)

  return (ts, ys)
```

Sprendimas 2-to laipsnio Rungės Kuto metodu:

```py
def RungeKutta2(fn, u0=1, lower_bound=0, upper_bound=1, tau=0.1):
  y = [u0]
  ts = np.arange(lower_bound, upper_bound + tau, tau)

  for t_n in ts[:-1]:
    k1 = fn(t_n, y[-1])
    k2 = fn(t_n + tau, y[-1] + tau * k1)
    y_ = y[-1] + (tau/2) * (k1 + k2)
    y.append(y_)

  return ts, y
```

Sprendimas naudojant ScyPy:

```py
def SciPy(fn, u0 = 1, lower_bound = 0, upper_bound = 1, tau = 0.1):
  ts = np.arange(lower_bound, upper_bound + tau, tau)
  sol = solve_ivp(f, (lower_bound, upper_bound), [u0], t_eval=ts)
  return ts, sol['y'][0]
```

Vizualizacijos

```py
f = lambda x, u: np.pow(x, 2) * np.log(u + x) - x

ts1, rk4_ys1 = RungeKutta4(f, tau = 0.05)
ts2, rk4_ys2 = RungeKutta4(f, tau = 0.1)

_, rk2_ys1 = RungeKutta2(f, tau = 0.05)
_, rk2_ys2 = RungeKutta2(f, tau = 0.1)

_, scipy1 = SciPy(f, tau = 0.05)
plt.plot(ts1, rk4_ys1, 'b', marker='x', label='τ = 0.05')
plt.plot(ts2, rk4_ys2, 'r', marker='+', label='τ = 0.1')
plt.plot(ts1, scipy1, color='black', label='SciPy sprendinys')
plt.xlabel('x')
plt.ylabel('u')
plt.title('Sprendimai intervale [0, 1]')
plt.legend()
plt.grid(True)
plt.savefig('rk4.png', dpi=300)
plt.show()

plt.plot(ts1, rk2_ys1, 'c:', marker='x', label='τ = 0.05')
plt.plot(ts2, rk2_ys2, 'm:', marker='+', label='τ = 0.1')
plt.plot(ts1, scipy1, color='black', label='SciPy sprendinys')
plt.xlabel('x')
plt.ylabel('u')
plt.title('Sprendimai intervale [0, 1]')
plt.legend()
plt.grid(True)
plt.savefig('rk2.png', dpi=300)
plt.show()

plt.plot(ts1, rk2_ys1, 'c--', marker='x', label='2-o laipsnio')
plt.plot(ts1, rk4_ys1, 'r:', marker='+', label='4-o laipsnio')
plt.xlabel('x')
plt.ylabel('u')
plt.title('Sprendimai intervale [0, 1]')
plt.legend()
```

```
plt.grid(True)
plt.savefig('both.png', dpi=300)
plt.show()
```

Paklaidos skaičiavimas ir vizualizacija

```py
def error(y_tau, y_2tau, tau = 0.1, order = 2):
  y_tau  = np.array(y_tau)
  y_2tau = np.array(y_2tau[::2])

  return np.abs(y_2tau - y_tau) / (2**order - 1)

rk2_err = error(rk2_ys2, rk2_ys1, order = 2)
rk4_err = error(rk4_ys2, rk4_ys1, order = 4)

plt.plot(ts2, rk2_err, 'b', marker='o')
plt.xlabel('x')
plt.ylabel('Paklaida')
plt.legend()
plt.grid(True)
plt.savefig('error_rk2.png', dpi=300)
plt.show()

plt.plot(ts2, rk4_err, 'r', marker='x')
plt.xlabel('x')
plt.ylabel('Paklaida')
plt.legend()
plt.grid(True)
plt.savefig('error_rk4.png', dpi=300)
plt.show()
```