Raport zadania 10
Metody Numeryczne

Łukasz Chmielowski (307713)

2023

1 Treść zadania

Rozwiąż równanie różniczkowe i wyznacz pierwiastki rozwiązania leżące w przedziale $x = [0, 4]$

$$y' - 2t + y^2 + 1 = 0, y(0) = 0 \tag{1}$$

2 Rozumowanie wykonania i teoria

Na początku dla własnej wygody zmieniłem oznaczenia z x na t. Następnie zapisałem równanie w następujący sposób:

$$\frac{dy}{dt} = 2t - y^2 - 1 \tag{2}$$

Jest to równanie różniczkowe nieliniowe pierwszego rzędu, skorzystałem zatem z metody Rungego-Kutty 4 rzędu. Wybrałem ją, ze względu na relatywną prostotę wzórów, oraz szybkość wykonania.

2.1 Metoda Rungego-Kutty 4 rzędu

Żeby zastosować tą metodę, musimy mieć równanie w postaci:

$$y' = f(x, y) \tag{3}$$

oraz znać wartosć $y(x_0) = y_0$.

Następnie musimy przyjąć dowolną wartość h, czyli kroku całkowania. Mając te rzeczy, możemy zapisać wzory iteracyjne:

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

gdzie:

$$k1 = hf(x_n, y_n)$$
$$k2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k4 = hf(x_n + h, y_n + k_3)$$

W ten sposób uzyskamy zestaw punktów dających rozwiązanie w przybliżeniu.

3 Kod

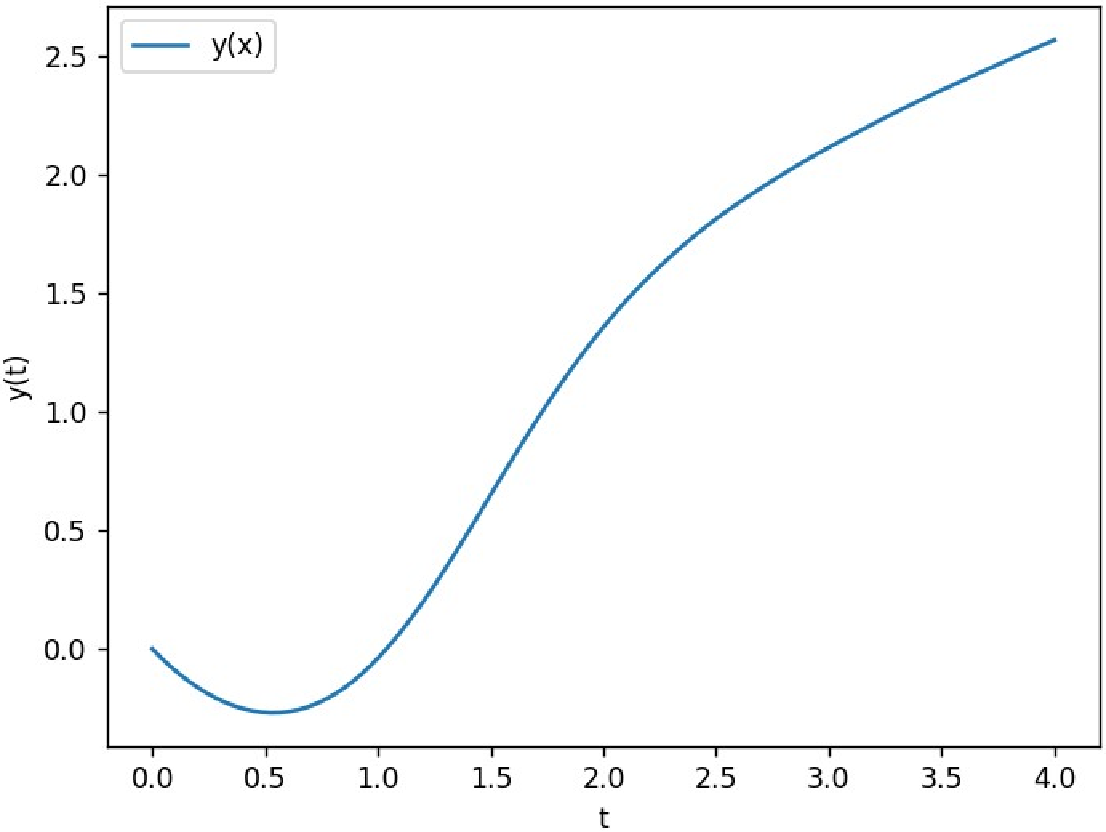
Po wybraniu metody przystąpiłem do pisania kodu Python, który w wersji końcowej wygląda następująco:

```
import numpy as np
from scipy.integrate import solve_ivp
from scipy.interpolate import CubicSpline
import matplotlib.pyplot as plt

def dydt(t, y):
    return 2*t - y**2 - 1

y0 = 0
t = np.linspace(0, 4, 100)

sol = solve_ivp(dydt, t_span=(0,max(t)), y0=[y0], t_eval=t)

y_sol = sol.y[0]

spl = CubicSpline(t, y_sol)
print(spl.roots())

plt.plot(t, y_sol, label='y(x)')
plt.legend(loc='best')
plt.ylabel('y(t)')
plt.xlabel('t')
plt.show()
```

Na początku załadowałem potrzebne biblioteki. Następnie zdefiniowałem funkcję *dydt*, która zwraca prawą stronę równania 2. Zdefiniowałem wartość funkcji $y(t)$ w punkcie $y(0)$, oraz utworzyłem tablicę wartości czasu, po których program będzie wyliczał rozwiązanie.

Do wyliczenia wartości równania, skorzystałem z funkcji *solve_ivp* z pakietu *SciPy* podanej na wykładzie.

Po otrzymaniu wyników wydobyłem tablicę rozwiązań dla y i wykonałem wykres z otrzymanych danych:



Na wykresie możemy zauważyć, że w zakresie $t = [0, 4]$ znajdują 2 pierwiastki tego rozwiązania. Pierwszy został podany w poleceniu, natomiast do wyliczenia drugiego korzystałem z funkcji $CubicSpline$ z pakietu $scipy.interpolate$ za pomocą modułu $roots$. Wartości liczbowe pierwiastków wynoszą:

$$y_1 = 0$$
$$y_2 = 1.03851753$$

4 Wnioski

Otrzymany wykres porównałem z rozwiązaniem tego równania za pomocą programu Mathematica. Po porównaniu wyników stwierdziłem, że wynik funkcji $solve_ivp$ jest bardzo podobny do wyniku Mathematici, przez co mogę stwierdzić, że otrzymane wyniki są poprawne i bardzo zbliżone do rozwiązania rzeczywistego.