Laboratorium 10 - Rozwiązywanie równań różniczkowych zwyczajnych

Piotr Karamon

03.06.2024r.

Treści zadań

Zadanie 1

Dane jest równanie różniczkowe (zagadnienie początkowe):

$$y' + y\cos x = \sin x \cos x, \quad y(0) = 0$$

Znaleźć rozwiązanie metodą Rungego-Kutty i metodą Eulera.
Porównać otrzymane rozwiązanie z rozwiązaniem dokładnym:

$$y(x) = e^{-\sin x} + \sin x - 1.$$

Zadanie 2

Dane jest zagadnienie brzegowe:

$$y'' + y = x, \quad y(0) = 1, \quad y(0.5\pi) = 0.5\pi - 1$$

Znaleźć rozwiązanie metodą strzałów.
Porównać otrzymane rozwiązanie z rozwiązaniem dokładnym:

$$y(x) = \cos x - \sin x + x.$$

Zadanie 1

Rozwiązanie

Przekształcamy równanie tak by po lewej stronie została jedynie pochodna.

$$y' = \sin x \cos x - y\cos x$$

Metoda Eulera jest jedną z najprostszych metod numerycznych do rozwiązywania równań różniczkowych zwyczajnych (ODE).

1. Musimy mieć równanie postaci $y' = f(x, y)$ oraz warunek początkowy $y(x_0) = y_0$
2. Wybieramy krok h.
3. Stosujemy iteracyjny wzór
$$y_{n+1} = y_n + h \cdot f(x_n, y_n)$$
gdzie $x_{n+1} = x_n + h$

Istnieje wiele wersji algorytmu Rungego-Kutty różniących się między sobą własnościami (jak stabilność, jawność, niejawność, metody osadzone, szybkość działania itp.) tutaj stosujemy zdecydowanie najbardziej popularną, czyli metodę rzędu 4. Metoda Rungego-Kutty czwartego rzędu (RK4) jest jedną z najczęściej używanych metod numerycznych ze względu na jej dokładność i stabilność.

1. Jak w metodzie Eulera, musimy mieć równanie postaci $y' = f(x, y)$ oraz warunek początkowy $y(x_0) = y_0$.
2. Wybieramy krok h.
3. Wzór RK4 używa czterech współczynników do obliczenia wartości y w następnym punkcie

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = hf(x_n + h, y_n + k_3)$$

Następnie wartośc y w punkcie x_{n+1} obliczamy jako:

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

Najpierw tworzymy funkcję $f(x, y)$ oraz funkcję opisującą dokładne rozwiązanie.

```
def f(x,y):
    return np.sin(x)*np.cos(x) - y *np.cos(x)

def exact_solution(x):
    return np.e**(-np.sin(x)) +np.sin(x) - 1
```

Bazując na wzorach powyżej tworzymy funkcję dla metody Eulera.

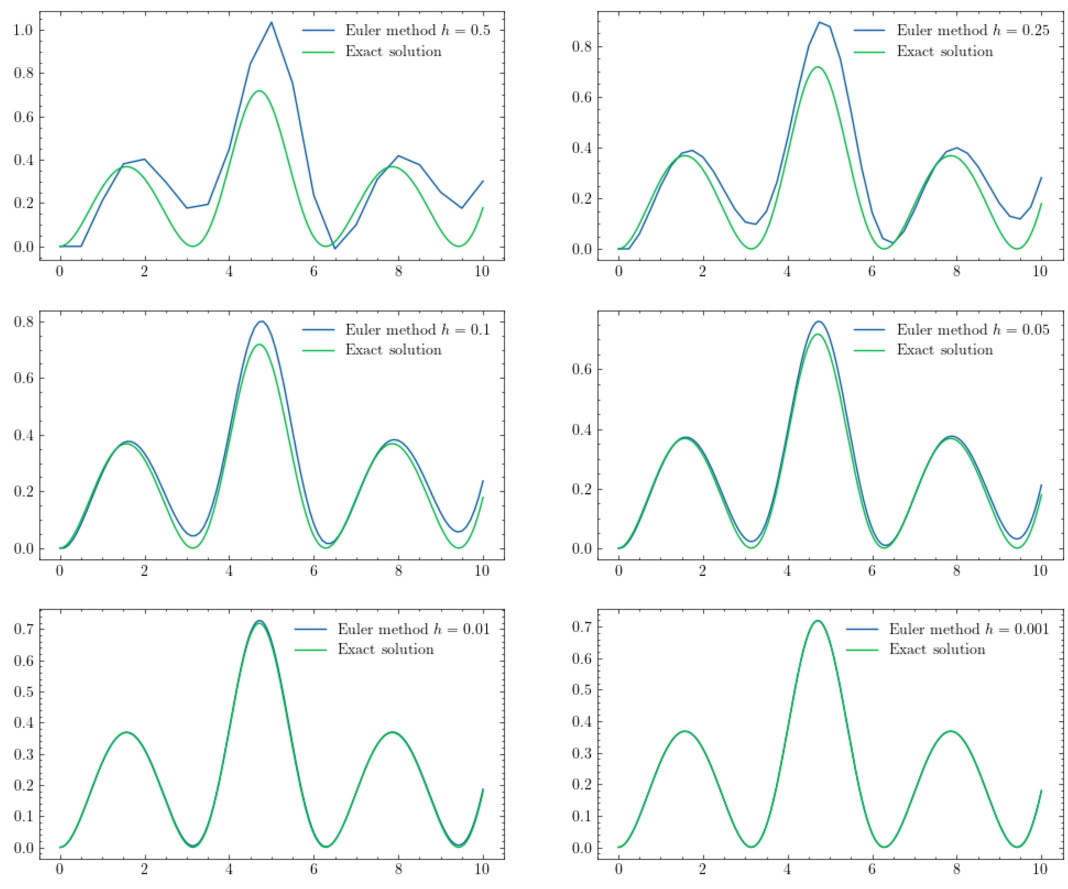
```
def euler_method(f, x0, y0, h, n):
    xs = [x0]
    ys = [y0]

    for _ in range(n):
        x = xs[-1]
        y = ys[-1]
        y += h * f(x, y)
        x += h
        xs.append(x)
        ys.append(y)
    return xs, ys
```

Robimy analogicznie dla metody RK4.

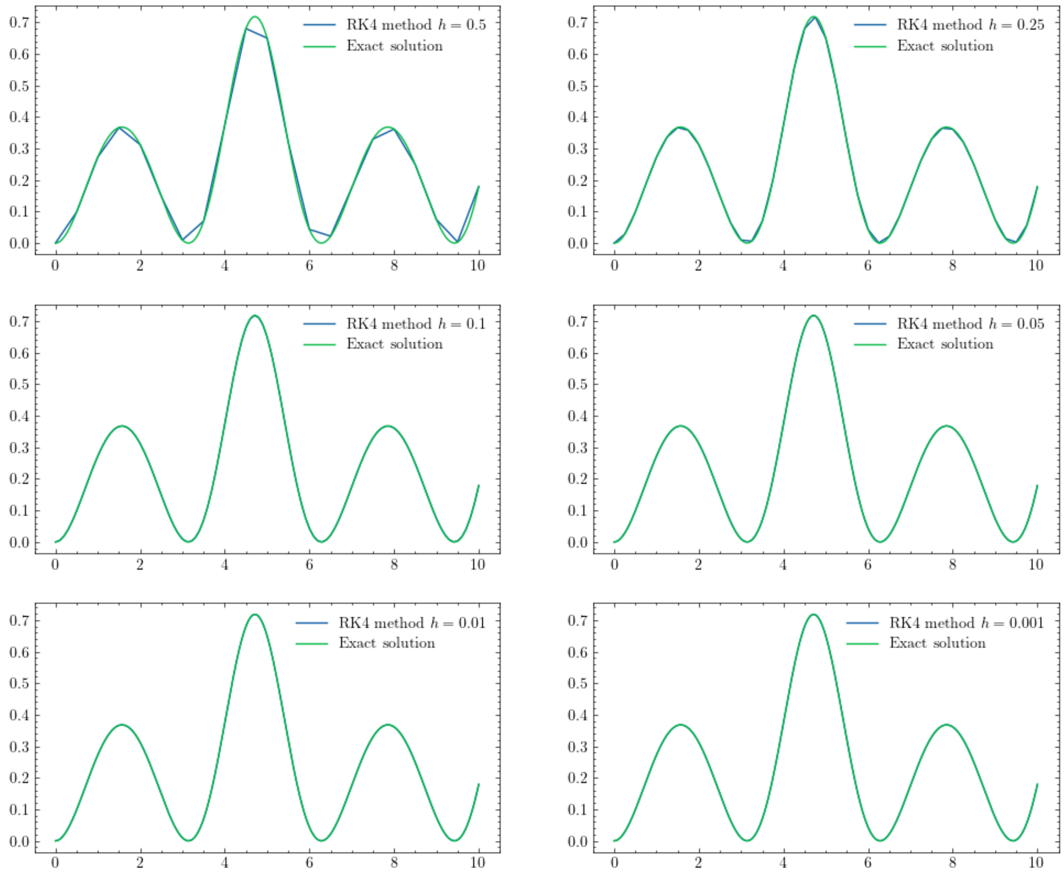
```
def rk4_method(f, x0, y0, h, n):
    xs = [x0]
    ys = [y0]
    for _ in range(n):
        x = xs[-1]
        y = ys[-1]
        k1 = h * f(x, y)
        k2 = h * f(x + h/2, y + k1/2)
        k3 = h * f(x + h/2, y + k2/2)
        k4 = h * f(x + h, y + k3)
        y += (k1 + 2*k2 + 2*k3 + k4)/6
        x += h
        xs.append(x)
        ys.append(y)
    return xs, ys
```

Sprawdzenie poprawności rozwiązania

Najpierw przetestujemy działanie metod wizualnie, będziemy szukać rozwiązania na przedziale $[0, 10]$.



Rysunek 1: Wykresy pokazujące rozwiązanie uzyskane metodą Eulera w zależności od kroku h.



Rysunek 2: Wykresy pokazujące rozwiązanie uzyskane metodą RK4 w zależności od kroku h.

Wykresy sugerują, że metoda RK4 jest bardziej dokładnie niż metoda Eulera. By zweryfikować to ilościowo obliczymy RMSE(Root Mean Squared Error) dla obu metod w zależności od kroku h.

Tablica 1: RMSE dla metody Eulera oraz RK4 w zależności od kroku h.

h	RMSE Euler	RMSE RK4
0.5	$1.68491 \cdot 10^{-1}$	$3.34763 \cdot 10^{-4}$
0.25	$9.79064 \cdot 10^{-2}$	$1.32932 \cdot 10^{-5}$
0.1	$4.43530 \cdot 10^{-2}$	$3.86852 \cdot 10^{-7}$
0.05	$2.32629 \cdot 10^{-2}$	$2.71244 \cdot 10^{-8}$
0.01	$4.84605 \cdot 10^{-3}$	$4.78514 \cdot 10^{-11}$
0.001	$4.89223 \cdot 10^{-4}$	$2.20822 \cdot 10^{-13}$

Wnioski

Przeprowadzona analiza numeryczna za pomocą metody Eulera i metody Rungego-Kutty czwartego rzędu (RK4) dla równania różniczkowego:

$$y' = \sin x \cos x - y \cos x$$

pokazała wyraźne różnice w dokładności tych metod.

Dokładność Metody Eulera

Metoda Eulera wykazuje znaczące błędy w przybliżeniu rozwiązania w porównaniu z metodą RK4. Przy większych krokach h, błędy te są szczególnie widoczne, co wynika z liniowego przybliżenia funkcji. Nawet przy zmniejszaniu kroku h do bardzo małych wartości (np. $h = 0.001$), metoda Eulera wciąż generuje błędy rzędu 10^{-4}, co może być niewystarczające dla bardziej precyzyjnych zastosowań.

Dokładność Metody RK4

Metoda Rungego-Kutty czwartego rzędu (RK4) znacznie przewyższa metodę Eulera pod względem dokładności. Nawet dla większych kroków h, metoda RK4 generuje bardzo małe błędy, co wynika z jej zaawansowanego podejścia, uwzględniającego więcej punktów w obliczeniach. Dla kroków h takich jak 0.01 czy 0.001, błędy są rzędu 10^{-11} i 10^{-13} odpowiednio, co wskazuje na bardzo wysoką precyzję tej metody.

Wpływ Kroku h na Dokładność

Obie metody wykazują zależność dokładności od wielkości kroku h, jednak RK4 radzi sobie znacznie lepiej przy większych krokach niż metoda Eulera.

Zastosowania Praktyczne

Z powyższych analiz wynika, że metoda Eulera może być użyteczna dla szybkich i zgrubnych obliczeń, gdzie wysoka precyzja nie jest wymagana. Natomiast metoda RK4 jest preferowana w sytuacjach wymagających dużej dokładności i stabilności, takich jak symulacje fizyczne, inżynieryjne oraz w naukach przyrodniczych.

Zadanie 2

Opis metody strzałów

Metoda strzałów jest numeryczną techniką stosowaną do rozwiązywania równań różniczkowych z warunkami brzegowymi. Procedura polega na przekształceniu problemu brzegowego na problem początkowy, a następnie iteracyjnym dostosowywaniu warunków początkowych, aż rozwiązanie spełni warunki brzegowe.

Kroki metody strzałów:

1. **Formułowanie równania różniczkowego i warunków brzegowych:** Mamy równanie różniczkowe drugiego rzędu:

   $$y'' + y = x$$

   z warunkami brzegowymi:

   $$y(0) = 1, \quad y(0.5\pi) = 0.5\pi - 1$$

2. **Zamiana problemu brzegowego na układ równań pierwszego rzędu:** Zamieniamy równanie różniczkowe drugiego rzędu na układ równań pierwszego rzędu. Wprowadzamy nową funkcję:

   $$v(x) = y'(x)$$

   Wówczas układ równań różniczkowych wygląda następująco: $\begin{cases} y' = v \\ v' = x - y \end{cases}$ z warunkami początkowymi:

   $$y(0) = 1, \quad v(0) = s$$

3. **Rozwiązanie problemu początkowego:** Rozwiązujemy układ równań różniczkowych z warunkami początkowymi $y(0) = 1$ oraz $v(0) = s$. Otrzymujemy rozwiązania $y(x; s)$ oraz $v(x; s)$.

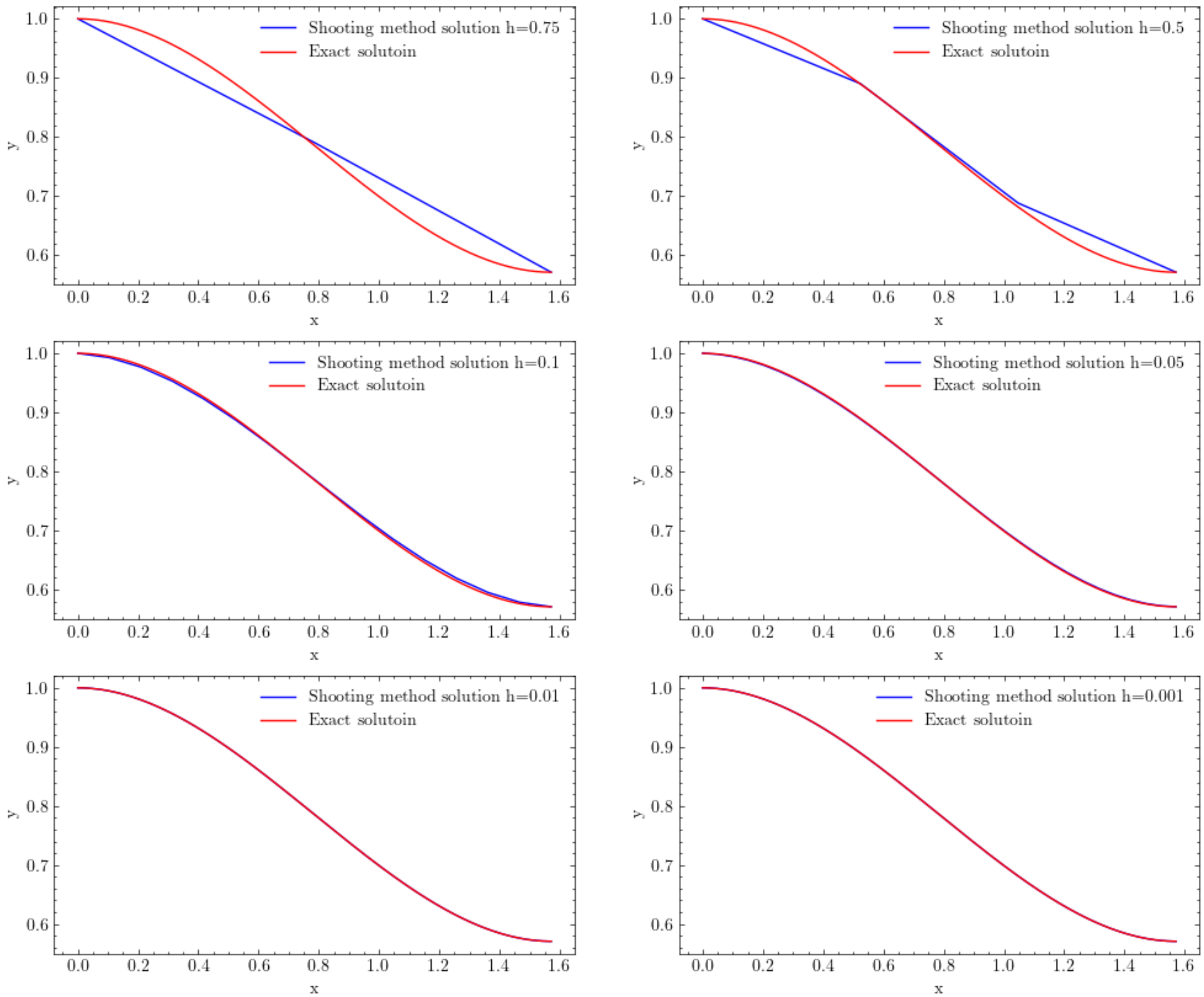
4. **Dopasowanie warunków brzegowych:** Iteracyjnie zmieniamy wartość s w taki sposób, aby rozwiązanie $y(x; s)$ spełniało drugi warunek brzegowy $y(0.5\pi) = 0.5\pi - 1$. W tym przypadku będziemy dobierać s używając bisekcji.

Zapis algorytmu

```
import numpy as np
import matplotlib.pyplot as plt

# Definiowanie parametrów
a = 0
b = 0.5 * np.pi
alpha = 1
beta = 0.5 * np.pi - 1
tolerance = 1e-6


# Funkcja do rozwiązania układu równań różniczkowych za pomocą metody Rungego-Kutty 4 rzędu
def rk4_shooting(s, h, n):
    x = np.linspace(a, b, n)
    y = np.zeros(n)
    v = np.zeros(n)
    y[0] = alpha
    v[0] = s

    for i in range(1, n):
        k1_y = h * v[i - 1]
        k1_v = h * (x[i - 1] - y[i - 1])

        k2_y = h * (v[i - 1] + 0.5 * k1_v)
        k2_v = h * (x[i - 1] + 0.5 * h - (y[i - 1] + 0.5 * k1_y))

        k3_y = h * (v[i - 1] + 0.5 * k2_v)
        k3_v = h * (x[i - 1] + 0.5 * h - (y[i - 1] + 0.5 * k2_y))

        k4_y = h * (v[i - 1] + k3_v)
        k4_v = h * (x[i - 1] + h - (y[i - 1] + k3_y))

        y[i] = y[i - 1] + (1 / 6) * (k1_y + 2 * k2_y + 2 * k3_y + k4_y)
        v[i] = v[i - 1] + (1 / 6) * (k1_v + 2 * k2_v + 2 * k3_v + k4_v)

    return y, v

def shooting_method(s1, s2, tolerance, h, n):
    while np.abs(s2 - s1) > tolerance:
        smid = (s1 + s2) / 2
        y1, _ = rk4_shooting(s1, h, n)
        ymid, _ = rk4_shooting(smid, h, n)

        if (y1[-1] - beta) * (ymid[-1] - beta) < 0:
            s2 = smid
        else:
            s1 = smid

    return smid
```

Sprawdzenie poprawności rozwiązania

Na początku zweryfikujemy naszą metodę wizualnie, tworząc wykresy które porównają otrzymane rozwiązanie z rozwiązaniem dokładnym.



Rysunek 3: Wykresy pokazujące rozwiązanie uzyskane metodą strzałów w zależności od kroku h.

Sprawdzamy jak dobrze obliczona przez nas funkcja jest zgodna z rzeczywistym rozwiązaniem, poprzez obliczenie RMSE(Root Mean Squared Error).

Tablica 2: RMSE dla metody strzałów w zależności od kroku h.

h	RMSE
0.75	$3.37779 \cdot 10^{-3}$
0.5	$2.97418 \cdot 10^{-3}$
0.1	$2.80038 \cdot 10^{-3}$
0.05	$8.38422 \cdot 10^{-4}$
0.01	$3.25396 \cdot 10^{-5}$
0.001	$3.27207 \cdot 10^{-5}$

Wnioski

Metoda strzałów, nawet przy relatywnie dużych krokach, generuje rozwiązanie bliskie dokładnemu, co pokazuje jej skuteczność w przybliżaniu wyników. Dla większych wartości kroku h, RMSE jest większe, co wskazuje na mniejszą dokładność rozwiązania. Przy mniejszych wartościach h, RMSE znacząco się zmniejsza, wskazując na wyższą dokładność. Optymalny krok obliczeniowy zależy od balansu między dokładnością rozwiązania a czasem obliczeń. Przy bardzo małych krokach, czas obliczeń znacząco wzrasta, mimo że dokładność się poprawia. Metoda strzałów pozwala na efektywną zamienię zagadnienia brzegowego w zagadnienie początkowe.

Bibliografia

- Marian Bubak, Katarzyna Rycerz: *Metody Obliczeniowe w Nauce i Technice. Metody całkowania równań różniczkowych zwyczajnych*

- Michael T. Heath: *Scientific Computing: An Introductory SurveyChapter 10 – Boundary Value Problems for Ordinary Differential Equations*

- Michael T. Heath: *Scientific Computing: An Introductory SurveyChapter 9 – Initial Value Problems forOrdinary Differential Equations*