Zadanie Numeryczne 14

Mateusz Wojtyna

1. Wstęp

Należało narysować wykres funkcji

$$F(x) = \int_{-\infty}^{x} \cos\left(\frac{1+t}{t^2+0.04}\right) e^{-t^2}\,\mathrm{d}t$$

oraz obliczyć $\lim_{x\to\infty} F(x)$ z dokładnością 10^{-8}.

2. Opis

Całkę możemy podzielić na 3 części:

$$\int_{-\infty}^{A} g(x)\,\mathrm{d}t + \int_{A}^{B} g(x)\,\mathrm{d}t + \int_{B}^{\infty} g(x)\,\mathrm{d}t$$

gdzie $g(x)$ jest funkcją podcałkową z treści zadania. Można zauważyć, że

$$\left|\cos\left(\frac{1+t}{t^2+0.04}\right) e^{-t^2}\right| \leq e^{-t^2}$$

oraz

$$\int_{-\infty}^{A} e^{-t^2}\,\mathrm{d}t \leq \frac{e^{-A^2}}{2A}$$

Chcemy więc, żeby $\frac{e^{-A^2}}{2A} \leq 10^{-8}$, rozwiązaniem czego jest $A \leq -4.03373$. Postępując analogicznie dla B otrzymujemy $B \geq 4.03373$. Zaokrąglając, wybieramy $A \leq -5$ oraz $B \geq 5$.

Całkę liczymy numerycznie korzystając z metody Romberga. Aby policzyć granicę funkcji w zadanej dokładności, wystarczy policzyć całkę w granicach od A do B.

3. Kod

Program napisano w Pythonie z użyciem pakietów *NumPy* oraz *Matplotlib*.

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.typing import NDArray


num = np.float64
array = NDArray[num]

A = -5
B = 5


def f(x: num) -> num:
    return np.cos((1 + x) / (x * x + 0.04)) * np.exp(-(x * x))


def trapezoidal_method(a: num, b: num, h: num) -> num:
    n = int((b - a) / h)
    sum = f(a) / 2 + f(b) / 2
    for _ in range(1, n):
        sum += f(a + h)
        a += h
    sum *= h
    return sum


def romberg(a: num, b: num, eps: num = 1e-8, limit: int = 25) -> num:
    h = (b - a) / 2

    # indeksy odwrócone: A[k][n] = A[n][k] z wykładu
    # base case: A[0][0] = 1 podział
    prev = [h * (f(a) + f(b))]

    for k in range(1, limit + 1):
        h /= 2

        # A[k+1][0] = złożona metoda trapezów
        cur = [trapezoidal_method(a, b, h)]

        # Wypełnienie wiersza korzystając ze wzoru na A[k][n]
        factor = 1
        for n in range(1, k + 1):
            factor *= 4
```

```
            cur.append((factor * cur[n - 1] - prev[n - 1]) / (factor - 1))

        if np.abs(cur[k] - prev[k - 1]) <= eps:
            return cur[-1]

        prev = cur

    return prev[-1]


def F(x: num, limit: num):
    if A < x < B:
        return romberg(A, x)
    elif x <= A:
        return 0
    else:
        return limit


def main():
    np.set_printoptions(suppress=True)

    # integral(-inf, inf) = integral(-inf, A) + integral(A, B) + integral(B, inf)
    # A < -4.03773
    # B > 4.03773

    limit = romberg(A, B)
    print("Granica w ∞:", limit)

    nodes = np.linspace(A, B, 1000)
    values = [F(x, limit) for x in nodes]

    plt.plot(nodes, values)
    plt.xlabel("x")
    plt.ylabel("F(x)")
    plt.title("Wykres funkcji F(x)")
    plt.grid(True)
    plt.show()


if __name__ == "__main__":
    main()
```

4. Wyniki

$$\lim_{x \to \infty} F(x) = 0.21961195$$