Metody Numeryczne - Zadanie XIV

Mateusz Kamiński

December 2025

1 Wstęp

W zadaniu należało obliczyć całkę nieoznaczoną funkcji,

$$F(x) = \int_{-\infty}^{x} \cos\left(\frac{1+t}{t^2+0.04}\right) e^{-t^2}\,\mathrm{d}t \tag{1.1}$$

narysować wykres funkcji oraz policzyć granicę F(x)

2 Opis

Całkę liczono metodą Romberga, która łączy metodę trapezów z ekstrapolacją Richardsona. Ograniczono przedział całkowania do $[-5, 5]$, który wyznaczono z zależności.

$$\int_{-\infty}^{\infty} e^{-t^2}\,\mathrm{d}t \le \frac{e^{-A^2}}{2A}, \quad A > 0, \tag{2.1}$$

$$\frac{e^{-25}}{10} \approx 1.338 \cdot 10^{-12} \ll 10^{-8} \tag{2.2}$$

Wkład „ogona” i „przodu” całki poza tym przedziałem jest zaniedbywalny i dla A = 5 ich wkład jest mniejszy niż wymagana dokładność 10^{-8}.

3 Implementacja

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

```
num = np.float64

def f(t):
    return np.cos((1 + t) / (t**2 + 0.04))*np.exp(-(t**2))

def romberg(a: num, b: num, tol=1e-8, max_k=25):
    # Tabela Romberga: R[k] przechowuje wiersz przybliżeń dla 2^k
przedziałów
    # R[k][0] to wynik złożonego wzoru trapezów (odpowiada A0,k)
    # R[k][n] to wynik po n-tej ekstrapolacji (odpowiada An,k)

    R = []
    # k = 0: Metoda trapezów dla 1 przedziału
    h = b - a
    R.append([(h / 2) *(f(a) + f(b))])

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

        # z punktów {0, 1} zagęszczamy do {0, 0.5, 1} itd...
        points = np.arange(a + h, b, 2*h)
        trapez_sum = R[k - 1][0] / 2 + h * np.sum(f(points))

        row = [trapez_sum]

        # Ekstrapolacja Richardsona: wypełniamy wiersz po prawej
        for n in range(1, k + 1):
            # Wzór: An,k = (4^n * An-1,k+1 - An-1,k) / (4^n - 1)
            factor = 4**n
            value = (factor * row[n - 1] - R[k - 1][n - 1]) / (factor - 1)
            row.append(value)

        R.append(row)

        # Kryteriu stopu
        if np.abs(R[k][k] - R[k - 1][k - 1]) < tol:
            return R

    return R

def compute_limit(A: num) -> num:
    R = romberg(-A, A, 1e-8)
    return R[-1][-1]

def F(x: num, A: num) -> num:
    if x <= -A:
        return 0.0
    if x >= A:
        return compute_limit(A)
    R = romberg(-A, x, 1e-8)
    return R[-1][-1]
```

```
def main():
    A = 5.0

    limit = compute_limit(A)
    print(f"Limit funkcji f(x) = cos(x) / (x^2 + 0.04) w przedziale [{-A},
{A}] = {limit}")

    # wykres
    x_i = np.linspace(-A, A, 1000)
    Fx = [F(x, A) for x in x_i]

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

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

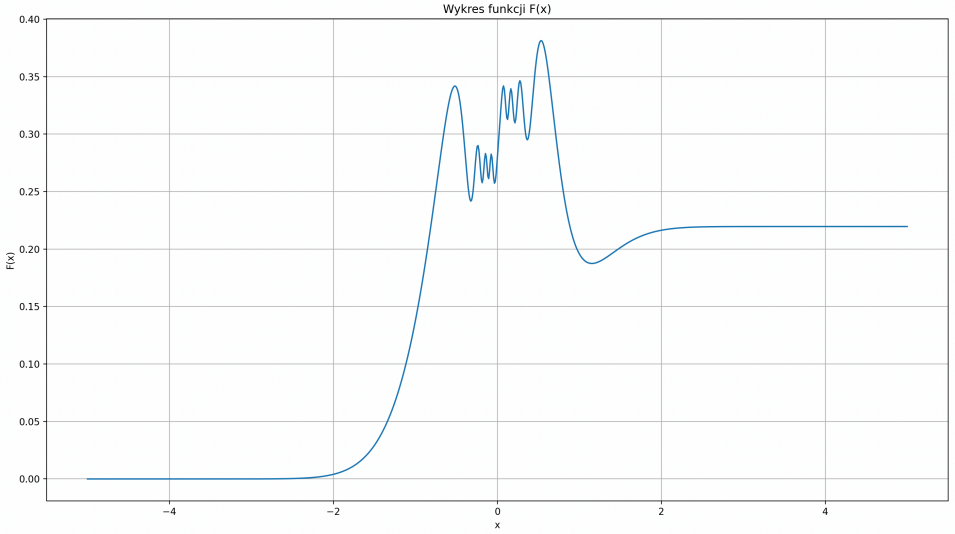
4 Wyniki

4.1 Granica funkcji

$$\lim_{x \to \infty} F(x) = 0.2196119 \tag{4.1}$$

4.2 Wykres

Wykres funkcji pierwotnej $F(x)$:


Wykres funkcji F(x)
F(x)
x
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4