Metody Numeryczne - Zadanie XIII

Mateusz Kamiński

December 2025

1 Wstęp

W zadaniu należało obliczyć całkę z dokładnością do 10^{-7}

$$\int_0^\infty \sin\left(\pi \frac{1+\sqrt{x}}{1+x^2}\right) e^{-x}\,\mathrm{d}x \tag{1.1}$$

metodą Romberga, która wykorzystuje również metodę trapezów.

2 Opis

Całkę dzielimy na część główną i „ogon":

$$I = \int_0^A f(x)\,\mathrm{d}x + \int_A^\infty f(x)\,\mathrm{d}x \tag{2.1}$$

Rozwiązując równanie $e^{-A} < 10^{-7}$ dostajemy A ≈ 16.118
Wybieramy $A = 17$, aby całka ogonowa spełniała $< 10^{-7}$.

2.1 Metoda Romberga

Metoda Romberga opiera się na wyznaczeniu kolejnych przybliżeń w tabeli, gdzie pierwsza kolumna to wyniki złożonego wzoru trapezów.

Kolejne elementy tabeli wyznaczane są za pomocą ekstrapolacji Richardsona zgodnie ze wzorem:

$$A_{n,k} = \frac{4^n \cdot A_{n-1,k+1} - A_{n-1,k}}{4^n - 1} \tag{2.2}$$

W zaimplementowanym algorytmie:

- Parametr `max_k` ogranicza maksymalną liczbę podziałów.
- Kryterium stopu sprawdzane jest jako:

$$|A_{k,k} - A_{k-1,k-1}| < \varepsilon \tag{2.3}$$

3 Implementacja

```
import numpy as np
from numpy.typing import NDArray

vector = NDArray[np.float64]
matrix = NDArray[np.float64]
num = np.float64

def f(x: num) -> num:
    return np.sin(np.pi * (1 + np.sqrt(x)) / (1 + x**2)) * np.exp(-x)

def romberg(a: num, b: num, tol=1e-7, 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 main():
    # exp(-A) < 10^-7
    # Wyznaczone A ~ -16.118
    A = 17.0

    tabelau = romberg(0, A)

    for i, row in enumerate(tabelau):
        print(f"k={i}: " + " ".join(f"{val:.10f}" for val in row))

    print(tabelau[-1][-1])

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

4 Wyniki

Wynik całki z pominięciem ogona, który można zaniedbać:

$$I = -0.2172750475 \tag{4.1}$$

4.1 Tabelau

k	Wartość $R_{k,k}$
0	0.0000000195
1	0.0003854301
2	0.0417589014
3	0.3700452339
4	0.1903045877
5	−0.1386427368
6	−0.1783582379
7	−0.2064448178
8	−0.2135061248
9	−0.2159571299
10	−0.2168117564
11	−0.2171117373
12	−0.2172174157

13	−0.2172547116
14	−0.2172678860
15	−0.2172725417
16	−0.2172741874
17	−0.2172747692
18	−0.2172749748
19	−0.2172750475