Laboratorium 4

Całkowanie numeryczne

Jan Rajczyk

12 kwietnia 2021

1 Treści zadań

1. Obliczyć $I = \int_0^1 \frac{1}{1+x}\,dx$ wg. wzoru prostokątów, trapezów i wzoru Simpsona (zwykłego i złożonego $n = 3, 5$). Porównać wyniki i błędy.

2. Obliczyć całkę $I = \int_{-1}^1 \frac{1}{1+x^2}\,dx$ korzystając z wielomianów ortogonalnych (np. Legandre'a) dla $n = 8$.

3. Obliczyć całkę:
$$\int_0^1 \frac{1}{1+x^2}\,dx$$
korzystając ze wzoru prostokątów, trapezów i wzoru Simpsona dla $h = 0.1$.

4. Metodą Gaussa obliczyć następującą całkę:
$$\int_0^1 \frac{1}{x+3}\,dx$$
dla $n = 4$. Oszacować resztę kwadratury.

2 Rozwiązania zadań

2.1 Zadanie 1

Celem zadania było aproksymacja całki $\int_0^1 \frac{1}{1+x}$ trzema metodami:

- metodą prostokątów,
- metodą trapezów,
- metodą Simpsona.

Całkowanie zostało przeprowadzone dla $N = 3{,}5$ (N - liczba przedziałów) i dla każdej metody obliczone zostały wartości funkcji oraz błędy bezwzględne.

$$S(f) = \int_0^1 f(x)\,dx = \int_0^1 \frac{1}{1+x}\,dx = \ln 2 \approx 0.69314718056$$

1. **Metoda prostokątów**
   Wartość funkcji definiujemy tutaj jako:

   $$S(f) = \int_a^b f(x)\,dx = \frac{b-a}{n}\sum_{i=0}^{n-1} f(x_i + \frac{1}{2}\cdot\frac{b-a}{n})$$

   W naszym przypadku $a = 0$, $b = 1$ stąd otrzymujemy przybliżenie funkcji:

   $$S(f) = \frac{1}{n}\sum_{i=0}^{n-1} f(x_i + \frac{1}{2n})$$

   Otrzymaliśmy następujące wyniki:

n	$S(f)$	Δ
1	0.6666666666666666	0.026480513893278657
3	0.6897546897546897	0.00339249080525561
5	0.6919078857159353	0.001239294844009975

Tablica 1: Porównanie wyników dla metody prostokątów

2. **Metoda trapezów**
   Wartość funkcji definiujemy tutaj jako:

$$S(f) = \int_a^b f(x)\,dx = \frac{1}{2} \cdot \frac{b-a}{n} \sum_{i=0}^{n-1} [f(x_i) + f(x_{i+1})]$$

   W naszym przypadku $a = 0$,$b = 1$ stąd otrzymujemy przybliżenie funkcji:

$$S(f) = \frac{1}{2n} \sum_{i=0}^{n-1} [f(x_i) + f(x_{i+1})]$$

   Otrzymaliśmy następujące wyniki:

n	$S(f)$	Δ
1	0.75	0.056852819440054714
3	0.7	0.006852819440054669
5	0.6956349206349206	0.00248774007497532

Tablica 2: Porównanie wyników dla metody trapezów

3. **Metoda Simpsona (parabol)**
   Wartość funkcji definiujemy tutaj jako:

$$S(f) = \int_a^b f(x)\,dx = \frac{h}{3} \cdot \sum_{i=1}^{n/2} [f(x_{2*i-2}) + 4f(x_{2*i-1}) + f(x_{2*i})],$$

   gdzie $h = \frac{b-a}{n}$ W naszym przypadku $a = 0$, $b = 1$, zaś $n = n+1$, ponieważ n musi być liczba parzystą. Stąd otrzymujemy przybliżenie funkcji:

$$\frac{h}{3} \cdot \sum_{i=1}^{n/2} [f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})],$$

   Otrzymaliśmy następujące wyniki:

n	$S(f)$	Δ
3	0.6932539682539682	0.0001067876940229473
5	0.6931697931697932	0.00002612609847927345

Tablica 3: Porównanie wyników dla metody trapezów

Obliczenia zostały wykonane z użyciem kodu w języku *Python*:

```
import numpy as np
from math import log

def fun(x):
    return 1 / (1 + x)


b: int = 1
a: int = 0
N = [1, 3, 5]


def rectangle_rule():
    print("Rectangle: ")
    for n in N:
        width = (b - a) / n
        pieces = np.linspace(a, b - width, n)
        pieces += width / 2
        f = lambda x: fun(x) * width
        approx = sum(f(pieces))
        print("n:", n, "S = ", approx, "E = ", abs(approx - log(2)))


def trapezoidal_rule():
    print("Trapezoidal: ")
    for n in N:
        width = (b - a) / n
        pieces = np.linspace(a, b, n + 1)
        approx: float = 0
        for i in range(1, n + 1):
            approx += width * (fun(pieces[i]) + fun(pieces[i - 1])) / 2

        print("n:", n, "S= ", approx, "E = ", abs(approx - log(2)))


def simpson_rule():
    print("Simpson: ")
    for n in N:
        m = n + 1
        width = (b - a) / m
        pieces = np.linspace(a, b, m + 1)
        approx: float = 0
        for i in range(1, m // 2 + 1):
            approx += width * (fun(pieces[2 * i - 2]) +
                      4 * fun(pieces[2 * i - 1]) + fun(pieces[2 * i])) / 3
        print("n:", n, "S = ", approx, "E = ", abs(approx - log(2)))
```

Wnioski: Jak można się było spodziewać metoda Simpsona daje najlepsze oszacowanie, natomiast pomiędzy dwiema pozostałymi metodami, czyli metoda prostokątów oraz trapezów nie widać znaczących różnic w aproksymacji. Widzimy również, że im większe jest n tym lepsza jest aproksymacja.

2.2 Zadanie 2

Najpierw obliczymy dokładną wartość całki:

$$\int_{-1}^{1} f(x)\,dx = \int_{0}^{1} \frac{1}{1+x^2}\,dx = \arctan 1 - \arctan -1 = \frac{\pi}{2} \approx 1.570796326794$$

Dla $N = 8$ aproksymujemy wartość całki jako:

$$\int_{-1}^{1} f(t)\,dt \approx S(f) = \sum_{k=1}^{8} w_k f(t_k)$$

k	w_k	t_k
1	0.3626837833783620	-0.1834346424956498
2	0.3626837833783620	0.1834346424956498
3	0.3137066458778873	-0.5255324099163290
4	0.3137066458778873	0.5255324099163290
5	0.2223810344533745	-0.7966664774136267
6	0.2223810344533745	0.7966664774136267
7	0.1012285362903763	-0.9602898564975363
8	0.1012285362903763	0.9602898564975363

Tablica 4: Wagi w_k oraz pierwiastki t_k dla $N = 8$

Korzystając z powyższej tabelki wstawiamy wartości do wzoru i otrzymujemy:

$$S(f) \approx 1.5707944125460624$$

oraz błąd bezwzględny równy: 0.00000191424883.
Do rozwiązania zadania użyłem programu napisanego w języku *Python*:

```
from math import pi
def f(x):
    return 1/(1+x**2)

A = [(0.3626837833783620, -0.1834346424956498),
(0.3626837833783620,0.1834346424956498),
(0.3137066458778873,-0.5255324099163290),
(0.3137066458778873,0.5255324099163290),
(0.2223810344533745,-0.7966664774136267),
```

```
(0.2223810344533745,0.7966664774136267),
(0.1012285362903763,-0.9602898564975363),
(0.1012285362903763,0.9602898564975363)]

approx: float = 0
for (w, t_k) in A:
    approx += w*f(t_k)

print("Result:", approx, "error: ", abs(approx-pi/2))
```

Wnioski: Użycie kwadratury Gaussa-Legendre'a daje znakomite wyniki, a także jest proste w implementacji, stąd nie dziwi taka popularność tej kwadratury.

2.3 Zadanie 3

Wzory, z których korzystaliśmy przy rozwiązywaniu problemów były identyczne jak te w zadaniu 1, a jedyne co uległo zmianie to n, które było równe 10.

Wartość całki wynosi:

$$\int_0^1 f(x)\,dx = \int_0^1 \frac{1}{1+x^2}\,dx = \arctan 1 = \frac{\pi}{4} \approx 0.785398163397$$

Wyniki przedstawia tabelka:

Metoda	$S(f)$	Δ
Metoda prostokątów	0.7856064962502743	0.00020833285282606528
Metoda trapezów	0.7849814972267897	0.00041666617065860834
Metoda Simpsona	0.7853981534848038	0.00000000991264448302

Tablica 5: Porównanie wyników dla trzech różnych metod

Podobnie jak w zadaniu 1 skorzystałem z kodu napisanego w języku *Python*:

```
import numpy as np
from math import log
from math import pi


def fun(x):
    return 1 / (1 + x**2)


b: int = 1
a: int = 0
n = 10


def rectangle_rule():
    print("rectangle: ")
    width: float = (b - a) / n
    pieces = np.linspace(a, b - width, n)
    pieces += width / 2
    f = lambda x: fun(x) * width
    approx = sum(f(pieces))
    print("n:", n, "S=", approx, "E=", abs(approx - pi/4))


def trapezoidal_rule():
    print("trapezoidal: ")
    width = (b - a) / n
    pieces = np.linspace(a, b, n + 1)
    approx: float = 0
    for i in range(1, n + 1):
        approx += width * (fun(pieces[i]) + fun(pieces[i - 1])) / 2
    print("n:", n, "S=", approx, "E=", abs(approx - pi/4))



def simpson_rule():
    print("Simpson: ")
    width = (b - a) / n
    pieces = np.linspace(a, b, n + 1)
    approx: float = 0
    for i in range(1, n // 2 + 1):
        approx += width * (fun(pieces[2 * i - 2]) +
                4 * fun(pieces[2 * i - 1]) + fun(pieces[2 * i])) / 3
    print("n:", n, "S=", approx, "E=", abs(approx - pi/4))
```

Wnioski: Podobnie jak w zadaniu 1 metoda Simpsona daje najlepsze oszacowanie, natomiast pomiędzy dwiema pozostałymi metodami, czyli metoda prostokątów oraz trapezów nie widać znaczących różnic w aproksymacji.
Warto również nadmienić, że przy relatywnie małej liczbie przedziałów wartość wyliczona za pomocą całkowania numerycznego różni się o bardzo niewiele od wartości rzeczywistej.

2.4 Zadanie 4

Obliczamy dokładną wartość całki:

$$\int_0^1 f(x)\,dx = \int_0^1 \frac{1}{x+3}\,dx = \ln(4) - \ln(3) \approx 0.28768207245178$$

Musimy dokonać przekształcenia przedziału $(0,1)$ na przedział $(-1,1)$ i stąd otrzymujemy wzór na wartość aproksymowaną postaci:

$$\int_0^1 f(t) \approx \frac{1-0}{2} \int_{-1}^1 g(x)\,dx,$$

gdzie $g(x) = f(\frac{1-0}{2}x + \frac{1+0}{2}) = f(\frac{1}{2}x + \frac{1}{2})$.

$$S(f) = \sum_{k=1}^{4} w_k g(t_k)$$

k	w_k	t_k
1	0.6521451548625461	-0.3399810435848563
2	0.6521451548625461	0.3399810435848563
3	0.3478548451374538	-0.8611363115940526
4	0.3478548451374538	0.8611363115940526

Tablica 6: Wagi w_k oraz pierwiastki t_k dla $N = 4$

Po wstawieniu wartości otrzymujemy: $S(f) = 0.2876820721506314$, zaś błąd bezwzględny równy: 0.000000000301149327697.

Wnioski: Widzimy, że już dla $n = 4$ otrzymujemy świetne przybliżenie wartości całki.

3 Bibliografia

1. Włodzimierz Funika *Materiały ze strony*
2. Katarzyna Rycerz *Wykład z przedmiotu Metody Obliczeniowe w Nauce i Technice*
3. `https://www.wolframalpha.com`
4. `https://pomax.github.io/bezierinfo/legendre-gauss.html`
5. `https://en.wikipedia.org/wiki/Numerical_integration`