Metody obliczeniowe w nauce i technice

Adam Naumiec
Marzec/kwiecień 2023

Laboratorium 5

Całkowanie numeryczne

Spis treści

1. Treść zadań 2
2. Rozwiązania 3
2.1. Zadanie 1. 3
2.1.1. Dokładna wartość całki 3
2.1.2. (a) Obliczenia wg wzoru prostokątów 3
2.1.3. (b) Obliczenia wg wzoru trapezów 3
2.1.4. (c1) Obliczenia wg wzoru Simpsona 4
2.1.5. Porównanie wyników i błędów 5
2.1.6. Wnioski 5
2.2. Zadanie 2. 7
2.2.1. Obliczenie całki 7
2.2.2. Wnioski 8
2.3. Zadanie 3. – zadanie domowe 1. 9
2.3.1. Dokładna wartość całki 9
2.3.2. (a) Obliczenia przy wykorzystaniu wzoru prostokątów 9
2.3.3. (b) Obliczenia przy wykorzystaniu wzoru trapezów 9
2.3.4. (c) Obliczenia przy wykorzystaniu wzoru Simpsona dla h=0,1 9
2.3.5. Wyniki 10
2.3.6. Wnioski 10
2.4. Zadanie 4. – zadanie domowe 2. 11
2.4.1. Dokładna wartość całki 11
2.4.2. Obliczenie całki metodą Gaussa dla n=4 11
2.4.3. Oszacowanie kwadratury 13
2.4.4. Wnioski 13
3. Bibliografia 14

1. Treść zadań

1. Zadanie 1. Obliczyć:

$$I = \int_0^1 \frac{1}{1+x} dx$$

wg wzoru:
   (a) prostokątów,
   (b) trapezów,
   (c) wzoru Simpsona zwykłego (dla $n = 3;\ 5$),
   (d) wzoru Simpsona złożonego (dla $n = 3;\ 5$).

   Porównać wyniki i błędy.

2. Zadanie 2. Obliczyć całkę:

$$I = \int_{-1}^1 \frac{1}{1+x^2} dx$$

   korzystając z wielomianów ortogonalnych (np. Legendre'a) dla $n = 8$.

3. Zadanie 3. – zadanie domowe 1. Obliczyć całkę:

$$I = \int_0^1 \frac{1}{1+x^2} dx$$

   korzystając z we wzoru:
   (a) prostokątów,
   (b) trapezów,
   (c) wzoru Simpsona

   dla $h = 0{,}1$.

4. Zadanie 4. – zadanie domowe 2. Metodą Gaussa obliczyć następującą całkę:

$$I = \int_0^1 \frac{1}{x+3} dx$$

   dla $n = 4$.
   Oszacować resztę kwadratury.

2. Rozwiązania

2.1. Zadanie 1.

2.1.1. Dokładna wartość całki

Dokładna wartość całki wynosi:

$$I = \int_0^1 f(x)dx = \int_0^1 \frac{1}{1+x}dx = [ln(x+1)]_0^1 = ln(2) - ln(1) = ln\left(\frac{2}{1}\right) = ln(2) \approx 0{,}6931.$$

2.1.2. (a) Obliczenia wg wzoru prostokątów

Metoda prostokątów polega na podziale przedziału całkowania $[a, b]$ na n równych części, a następnie przybliżeniu wartości całki jako sumy pól prostokątów o szerokości równej długości przedziału całkowania podzielonej przez n. W zależności od sposobu wyboru punktu, w którym przyjmowana jest wartość funkcji, można wyróżnić trzy warianty metody prostokątów: lewy, prawy i środkowy. W naszym przypadku wybrano metodą środkową i wyliczano wartości funkcji całkowanej dokładnie w środku ów przedziałów w punktach c_i, a następnie obliczono:

$$I_n = \sum_{i=0}^{n-1}\left(f(c_i)\cdot\frac{b-a}{n}\right) = \frac{b-a}{n}\cdot\sum_{i=0}^{n-1} f\left(x_i + \frac{1}{2}\frac{b-a}{n}\right) = h\cdot\sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right).$$

Wyniki obliczeń zaprezentowano w tabeli:

n	I_n	Błąd bezwzględny	Błąd względny
3	0.7833333333	0.0901861528	0.1301111154
5	0.7456349206	0.0524877401	0.07572380231
10	0.7187714032	0.0256242226	0.03696793889

Tabela 1. Wyniki całkowania numerycznego z wykorzystaniem metody prostokątów

2.1.3. (b) Obliczenia wg wzoru trapezów

Metoda trapezów jest techniką numeryczną stosowaną do przybliżonego obliczania wartości całki oznaczonej z funkcji na przedziale $[a, b]$. Przedział ten jest dzielony na $n-1$ równych podprzedziałów za pomocą n punktów, a następnie wartości funkcji są obliczane w pewnych punktach ci, które znajdują się na krańcach tych podprzedziałów oddalonych o h. Wzór, który jest używany do obliczenia wartości całki, jest następujący:

$$I_n = \sum_{i=1}^{n-1}\frac{h}{2}\left(f(c_{i-1}) + f(c_i)\right).$$

Wyniki obliczeń zaprezentowano w tabeli:

n	I_n	Błąd bezwzględny	Błąd względny
3	0.6970238095	0.0038766290	0.0098865286
5	0.6956349206	0.0024877401	0.0035890503
10	0.6937714032	0.0006242226	0.0009005629

Tabela 2. Wyniki całkowania numerycznego z wykorzystaniem metody trapezów

2.1.4. (c1) Obliczenia wg wzoru Simpsona

Metoda Simpsona jest jedną z metod numerycznych stosowanych do obliczania przybliżonego całkowania funkcji. Metoda ta polega na przybliżeniu funkcji przez parabolę na każdym z podziałów przedziału całkowania i całkowaniu tej paraboli. Wzór na całkowanie paraboli jest znany analitycznie, co pozwala na dokładne obliczenie całki na danym przedziale. Następnie wyniki całkowania parabol na każdym podprzedziale są sumowane, aby uzyskać przybliżoną wartość całki na całym przedziale. Metoda Simpsona jest bardziej dokładna niż metoda prostokątów i trapezów, ponieważ uwzględnia krzywiznę funkcji. Jednakże, do zastosowania tej metody, funkcja musi być wystarczająco gładka i paraboliczna na każdym podprzedziale, co nie zawsze jest spełnione. Sumę liczymy za pomocą wzoru:

$$I_n = \sum_{i=3}^{n} \frac{h}{3}\big(f(c_i) + 4f(c_{i-1}) + f(c_{i-2})\big).$$

n	I_n	Błąd bezwzględny	Błąd względny
3	0.6931545307	0.0000073501	0.0000106039
5	0.6931502307	0.0000030501	0.0000044004

Tabela 3. Wyniki całkowania numerycznego z wykorzystaniem metody Simpsona.

Złożona metoda Simpsona to rozszerzenie metody Simpsona na przypadki, gdy funkcja jest bardziej skomplikowana i nie można jej całkować w jednym przedziale. Złożona metoda Simpsona polega na podziale przedziału całkowania na kilka mniejszych przedziałów i zastosowaniu metody Simpsona do każdego z nich, a następnie zsumowaniu wyników.

Wyniki dla wzoru Simpsona złożonego:

n	I_n	Błąd bezwzględny	Błąd względny
3	0.6931471824	0.0000000019	0.0000000027
5	0.6931471813	0.0000000008	0.0000000011

Tabela 4. Wyniki całkowania numerycznego z wykorzystaniem złożonej metody Simpsona.

2.1.5. Porównanie wyników i błędów

Metoda Simpsona okazała się dawać najlepsze wyniki, najbliższe wartościom dokładnym. Metoda trapezów okazała się lepsza od metody prostokątów. Złożona metoda Simpsona dała wyniki nader satysfakcjonujące.

2.1.6. Wnioski

Wszystkie metody dawały wyniki dość bliskie wartościom rzeczywistym, więc można twierdzić, że programy działały poprawnie. Najlepsza okazała się metoda Simpsona.

Program, którym realizowano obliczenia:

```
import numpy as np
from math import log

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

def rectangle(a, b, n):
    h = (b - a) / n
    integral = 0

    for i in range(n):
        xi = a + i * h
        integral += f(xi) * h

    absolute_error = abs(integral - log(2))
    relative_error = absolute_error / log(2)

    return format(integral, '.10f'), format(absolute_error, '.10f'),
format(relative_error, '.10f')

def trapezoidal(a, b, n):
    h = (b - a) / n
    integral = (f(a) + f(b)) / 2

    for i in range(1, n):
        xi = a + i * h
        integral += f(xi)
    integral *= h

    absolute_error = abs(integral - log(2))
    relative_error = absolute_error / log(2)

    return format(integral, '.10f'), format(absolute_error, '.10f'),
format(relative_error, '.10f')

def simpson1(a, b, n):
    h = (b - a) / n
    x = [a + i * h for i in range(n + 1)]
    y = [f(xi) for xi in x]
    integral = y[0] + y[-1]

    for i in range(1, n, 2):
        integral += 4 * y[i]
    for i in range(2, n - 1, 2):
```

```
        integral += 2 * y[i]

    integral *= h / 3

    absolute_error = abs(integral - log(2))
    relative_error = absolute_error / log(2)

    return format(integral, '.10f'), format(absolute_error, '.10f'),
format(relative_error, '.10f')

def simpson2(a, b, n, eps=1e-8):
    m = n // 2
    dx = (b - a) / n
    x = a + dx
    sum1 = 0
    sum2 = 0

    for i in range(1, m + 1):
        sum1 += f(x)
        x += 2 * dx
    x = a + 2 * dx
    for i in range(1, m):
        sum2 += f(x)
        x += 2 * dx
    integral = dx * (f(a) + 4 * sum1 + 2 * sum2 + f(b)) / 3

    err = eps + 1
    while err > eps:
        old_integral = integral
        n *= 2
        m *= 2
        dx /= 2
        x = a + dx
        sum1 = 0
        sum2 = 0

        for i in range(1, m + 1):
            sum1 += f(x)
            x += 2 * dx
        x = a + 2 * dx
        for i in range(1, m):
            sum2 += f(x)
            x += 2 * dx

        integral = dx * (f(a) + 4 * sum1 + 2 * sum2 + f(b)) / 3
        err = abs(integral - old_integral) / 15

    absolute_error = abs(integral - log(2))
    relative_error = absolute_error / log(2)
    return format(integral, '.10f'), format(absolute_error, '.10f'),
format(relative_error, '.10f')

if __name__ == "__main__":
    a = 0
    b = 1
    h = 0.1
    n = 10
    print("Rectangle method: ", rectangle(a, b, n))
    print("Trapezoidal method: ", trapezoidal(a, b, n))
    print("Simpson method: ", simpson1(a, b, n))
    print("Composite Simpson method: ", simpson2(a, b, n))
```

2.2. Zadanie 2.

2.2.1. Obliczenie całki

$$I = \int_{-1}^{1} \frac{1}{1+x^2} dx = [arctg(x)]_{-1}^{1} = \frac{\pi}{2} \approx 1{,}5707963268.$$

Kolejne wielomiany Legendre'a można wliczyć z wzoru rekurencyjnego:

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x),$$

gdzie:

$$P_0(x) = 1,$$
$$P_1(x) = x.$$

Liczymy całkę na przedziale $[-1,1]$ z wagą $w(x) = 1$, zatem wielomiany te ortogonalne, wniosek ten wykorzystamy w obliczeniach.

Wielomian wykorzystany do aproksymacji jest postaci:

$$F(x) = \sum_{i=0}^{n} c_i P_i(x),$$

a współczynniki c_i są postaci:

$$c_i = \frac{\int_a^b f(x)L_i(t)dx}{\int_a^b L_i^2(x)dx}.$$

W naszym przypadku:

$$F(x) = \sum_{i=0}^{2} c_i P_i(x),\ x \in [-1,1]; \qquad c_i = \frac{\int_{-1}^{1} f(x)L_i(t)dx}{\int_{-1}^{1} L_i^2(x)dx}.$$

Po dokonaniu obliczeń otrzymujemy:

$$F(x) = 0{,}1705x^8 - 0{,}5668x^6 + 0{,}8856x^4 - 0{,}9885x^2 + 0{,}9998x.$$

Przybliżamy całkowaną funkcję funkcją aproksymującą, możemy zapisać:

$$I = \int_{-1}^{1} \frac{1}{1+x^2} dx \approx \int_{-1}^{1} F(x)dx \approx$$

$$\approx \int_{-1}^{1} (0{,}1705x^8 - 0{,}5668x^6 + 0{,}8856x^4 - 0{,}9885x^2 + 0{,}9998x)dx.$$

Obliczamy całkę z wielomianu aproksymującego i otrzymujemy wynik przybliżenia:

$$[0{,}0189x^9 - 0{,}0809x^7 + 0{,}1771x^5 - 0{,}3295x^3 + 0{,}9998x]_{-1}^{1} \approx 1{,}5707963251.$$

Błąd bezwzględny wynosi w przybliżeniu:

$$|1{,}5707963268 - 1{,}5707963251| \approx -1{,}7e - 9.$$

2.2.2. Wnioski

Otrzymany wynik okazał się być bliski wartości rzeczywistej. Sposób ten wymagał jednak trochę więcej obliczeń co w przypadku bardziej skomplikowanych zadań może narażać nas na błędy. Niemniej udało się pokazać, że jest to skuteczna metoda całkowania numerycznego z dużą dokładnością.

Program napisany w języku Python z wykorzystaniem biblioteki scipy do obliczania wartości liczbowych:

```
from scipy.special import roots_legendre


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


def legendre_integrate(n=8):
    x, w = roots_legendre(n)

    integral = 0
    for i in range(n):
        integral += w[i] * f(x[i])

    return integral


if __name__ == "__main__":
    n = 8
    integral = legendre_integrate(n)
    print("Wynik całkowania: ", integral)
```

2.3. Zadanie 3. – zadanie domowe 1.

2.3.1. Dokładna wartość całki

Dokładna wartość całki wynosi:

$$I = \int_0^1 f(x)dx \int_0^1 \frac{1}{1+x^2}dx = [arctg(x)]_0^1 = arctg(1) - arctg(0) = \frac{\pi}{4} - 0 = \frac{\pi}{4} \approx 0{,}7854.$$

W przedstawionych programach w tym zadaniu wykorzystano język Python i bibliotekę numpy oraz math. Zdefiniowano także całkowaną funkcję.

```
import numpy as np
from math import pi

def f(x):
    return 1 / ((x**2) + 1)
```

2.3.2. (a) Obliczenia przy wykorzystaniu wzoru prostokątów

Funkcja obliczająca całkę przy użyciu metody prostokątów:

```
def rectangle(a, b, h=0.1, n=10):
    array = np.linspace(a, b, num=n, endpoint=False)
    integral = sum(map(lambda x: f(x) * h, array))
    absolute_error = abs(integral - (pi / 4))

    return integral, absolute_error
```

2.3.3. (b) Obliczenia przy wykorzystaniu wzoru trapezów

Funkcja obliczająca całkę przy użyciu metody trapezów:

```
def trapezoidal(a, b, h=0.1, n=10):
    array = np.linspace(a, b, num=n + 1, endpoint=True)
    integral = sum([(f(x) + f(x - h)) * h / 2 for x in array[1:]])
    absolute_error = abs(integral - (pi / 4))

    return integral, absolute_error
```

2.3.4. (c) Obliczenia przy wykorzystaniu wzoru Simpsona dla h=0,1

Funkcja obliczająca całkę przy użyciu metody Simpsona:

```
def simpson(a, b, h=0.1):
    n = int((b - a) / (2 * h))
    x = np.linspace(a, b, 2 * n + 1)
    y = f(x)
    integral = h / 3 * np.sum(y[0:-1:2] + 4 * y[1::2] + y[2::2])
    absolute_error = abs(integral - (pi / 4))

    return integral, absolute_error
```

2.3.5. Wyniki

Otrzymane wyniki wraz z błędem bezwzględnym:

- Metoda prostokątów: 0.8099814972, 0.0245833338.
- Metoda trapezów: 0.7849814972, 0.0004166662.
- Metoda Simpsona: 0.7853981535, 0.0000000099.

2.3.6. Wnioski

Najdokładniejsza okazała się metoda Simpsona, dawała wyniki zdecydowanie bliższe wartościom rzeczywistym. Metoda trapezów okazała się być lepsza od metody prostokątów.

Cały program:

```
import numpy as np
from math import pi


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


def rectangle(a, b, h=0.1, n=10):
    array = np.linspace(a, b, num=n, endpoint=False)
    integral = sum(map(lambda x: f(x) * h, array))
    absolute_error = abs(integral - (pi / 4))

    return format(integral, '.10f'), format(absolute_error, '.10f')


def trapezoidal(a, b, h=0.1, n=10):
    array = np.linspace(a, b, num=n + 1, endpoint=True)
    integral = sum([(f(x) + f(x - h)) * h / 2 for x in array[1:]])
    absolute_error = abs(integral - (pi / 4))

    return format(integral, '.10f'), format(absolute_error, '.10f')


def simpson(a, b, h=0.1):
    n = int((b - a) / (2 * h))
    x = np.linspace(a, b, 2 * n + 1)
    y = f(x)
    integral = h / 3 * np.sum(y[0:-1:2] + 4 * y[1::2] + y[2::2])
    absolute_error = abs(integral - (pi / 4))

    return format(integral, '.10f'), format(absolute_error, '.10f')


if __name__ == "__main__":
    a = 0
    b = 1
    h = 0.1
    n = 10

    print("Rectangle method: ", rectangle(a, b, h, n))
    print("Trapezoidal method: ", trapezoidal(a, b, h, n))
    print("Simpson method: ", simpson(a, b, h))
```

2.4. Zadanie 4. – zadanie domowe 2.

2.4.1. Dokładna wartość całki

Dokładna wartość całki wynosi:

$$I = \int_0^1 f(x)dx \int_0^1 \frac{1}{x+3}dx = [ln(x+3)]_0^1 = ln(4) - ln(3) = ln\left(\frac{4}{3}\right) \approx 0{,}2877.$$

2.4.2. Obliczenie całki metodą Gaussa dla n=4

Do obliczeń korzystamy z kwadratury Gaussa-Legendre'a dla $n = 4$. Współczynnik tej kwadratury wygląda następująco:

$$\begin{bmatrix} \phi_0(x_1) & \phi_0(x_2) & \phi_0(x_3) & \phi_0(x_4) \\ \phi_1(x_1) & \phi_1(x_2) & \phi_1(x_3) & \phi_1(x_4) \\ \phi_2(x_1) & \phi_2(x_2) & \phi_2(x_3) & \phi_2(x_4) \\ \phi_3(x_1) & \phi_3(x_2) & \phi_3(x_3) & \phi_3(x_4) \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \int_{-1}^{1} w(x)\phi_0(x)dx \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

oraz możemy zapisać:

$$\begin{gathered} w(x) = 1, \\ \phi_0(x) = 1, \\ \phi_1(x) = x, \\ \phi_2(x) = \frac{1}{2}(3x^2 - 1), \\ \phi_3(x) = \frac{1}{2}(5x^3 - 3x), \\ \phi_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3); \end{gathered}$$

gdzie: x_1, x_2, x_3, x_4 to miejsca zerowe wielomianu ϕ_4.

Otrzymujemy:

$$\frac{1}{8}(35x^4 - 30x^2 + 3) = 0.$$

Rozwiązujemy równanie i otrzymujemy:

$$x_i = \pm\sqrt{\frac{\frac{30}{8} \pm \sqrt{\frac{15}{2}}}{\frac{70}{8}}} = \pm\frac{\sqrt{525 \pm 70\sqrt{30}}}{35},$$

zatem:

$$x_1 \approx -0{,}339981,$$
$$x_2 \approx 0{,}339981,$$
$$x_3 \approx -0{,}861136,$$
$$x_4 \approx 0{,}861136.$$

Zapisujemy:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{\sqrt{525-70\sqrt{30}}}{35} & \frac{\sqrt{525-70\sqrt{30}}}{35} & -\frac{\sqrt{525+70\sqrt{30}}}{35} & \frac{\sqrt{525-70\sqrt{30}}}{35} \\ \frac{5-3\sqrt{30}}{35} & \frac{5-3\sqrt{30}}{35} & \frac{5+3\sqrt{30}}{35} & \frac{5+3\sqrt{30}}{35} \\ \frac{3\sqrt{5}+5\sqrt{6}}{35}\sqrt{\frac{1}{7}(15-2\sqrt{30})} & -\frac{3\sqrt{5}+5\sqrt{6}}{35}\sqrt{\frac{1}{7}(15-2\sqrt{30})} & \frac{3\sqrt{5}+5\sqrt{6}}{35}\sqrt{\frac{1}{7}(15+2\sqrt{30})} & -\frac{3\sqrt{5}+5\sqrt{6}}{35}\sqrt{\frac{1}{7}(15-2\sqrt{30})} \end{bmatrix}$$

i rozwiązujemy układy równań:

$$a_1 = \frac{18+\sqrt{30}}{36} \approx 0{,}652145,$$

$$a_2 = \frac{18+\sqrt{30}}{36} \approx 0{,}652145,$$

$$a_3 = \frac{18-\sqrt{30}}{36} \approx 0{,}652145,$$

$$a_4 = \frac{18-\sqrt{30}}{36} \approx 0{,}652145.$$

Wielomiany ortogonalne określone są w przedziale $[-1,1]$, a nasza całka jest liczona w przedziale $[0,1]$. „Przechodzimy” (jak wcześniej) na przedział $[0,1]$.

Niech $x = 2t - 1$, a $t \in [-1,1]$.

Zatem:

$$t = \frac{x+1}{2},$$

oraz:

$$\int_0^1 \frac{1}{x+3}\,dx = \int_{-1}^1 \frac{1}{t+3}\,dt = \int_{-1}^1 \frac{1}{\frac{x+1}{2}+3}\,dx,$$

otrzymujemy, że funkcja f ma postać:

$$f(x) = \frac{1}{\frac{x+1}{2} + 3}.$$

Ostatecznie z powyższych rozważań otrzymujemy:

$$\int_0^1 \frac{1}{x+3} dx \approx \frac{1-0}{2} \sum_{i=1}^{n} a_i f(x_i) = \frac{1}{2} \sum_{i=1}^{4} a_i f(x_i) =$$

$$= a_1 f(x_1) + a_2 f(x_2) + a_3 f(x_3) + a_4 f(x_4).$$

Co po wyliczeniu daje wynik:

$$\int_0^1 \frac{1}{x+3} dx \approx 0{,}2876820721.$$

2.4.3. Oszacowanie kwadratury

Obliczamy błąd kwadratury:

$$|0{,}2876820724 - 0{,}2876820721| = 2.9e - 9.$$

2.4.4. Wnioski

Kwadratura dawała bardzo dokładne wyniki, można uznać ją za nieskomplikowaną metodę dającą satysfakcjonujące rezultaty przy prostszych obliczeniach numerycznych nawet przy niewielkich parametrach. Możliwe jest zwiększenie dokładności kosztem liczby koniecznych obliczeń i czasu potrzebnego na ich wykonanie.

3. Bibliografia

1. Wykłady dr inż. Katarzyny Rycerz z przedmiotu *Metody obliczeniowe w nauce i technice* na czwartym semestrze kierunku Informatyka w AGH w Krakowie
2. Wykresy kreślono za pomocą internetowego programu GeoGebra: https://www.geogebra.org/calculator
3. Obliczenia wykonywano za pomocą internetowego programu WolframAlpha: https://www.wolframalpha.com/ oraz programu Microsoft Excel: https://www.microsoft.com/pl-pl/microsoft-365/excel
4. Programy napisane zostały w języku Python w wersji 3.11: https://www.python.org/
5. Wykorzystano bibliotekę NumPy dla języka Python w wersji 1.24: https://numpy.org/doc/stable/index.html
6. Wykorzystano bibliotekę SciPy dla języka Python w wersji 1.10.1: https://scipy.org/
7. Kalkulator całek – *WolframAlpha Online Integral Calculator:* https://www.wolframalpha.com/calculators/integral-calculator/
8. https://home.agh.edu.pl/~funika/mownit/lab5/calkowanie.pdf
9. https://home.agh.edu.pl/~funika/mownit/lab5/what_is_gauss.html
10. ChatGPT: https://chat.openai.com/
11. https://en.wikipedia.org/wiki/Numerical_integration
12. https://en.wikipedia.org/wiki/Simpson%27s_rule
13. https://en.wikipedia.org/wiki/Riemann_sum
14. https://en.wikipedia.org/wiki/Trapezoidal_rule