Laboratorium 5

Całkowanie numeryczne

Krzysztof Solecki

1 Kwietnia 2023

1 Treści zadań

1.1 Zadania

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. Legendre'a) dla n=8.

1.2 Zadania domowe

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

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

2 Rozwiązania - zadania laboratoryjne

2.1 Obliczyć $\mathrm{I} = {}_0\int^1 1/(1+\mathrm{x})\mathrm{dx}$ wg wzoru prostokątów, trapezów i wzoru Simpsona (zwykłego i złożonego $\mathrm{n} = 3, 5$). Porównać wyniki i błędy.

Dokładna wartość całki wynosi:

$$\int_0^1 1/(1+x)dx = \ln(x+1)|_0^1 = \ln 2 \approx 0.69314718056$$

Całkowanie metodą prostokatów na zadanym przedziale [a,b] polega na obliczeniu następującej sumy:

$$I_n = \sum_{i=1}^{n} f(c_i) * \frac{|b-a|}{n}$$

gdzie: c_i to środek każdego z n podprzedziałów przedziału $[a, b]$ Całka jest więc równa sumie pól prostokątów o szerokości każdego podprzedziału i wysokości równej wartości funkcji całkowanej w środku tego przedziału.

Całkowanie metodą trapezów polega na policzeniu poniższej sumy:

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

gdzie: c_i to każdy z n punktów, powstałych poprzez podział przedziału [a,b] na $n-1$ przedziałów. Punkty oddalone są od siebie o h

Całkowanie metoda Simpsona polega na policzeniu poniższej sumy:

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

gdzie: c_i to każdy z n punktów, gdzie $n = 2k+1$, powstałych poprzez podział przedziału [a,b] na $n-1$ równych podprzedziałów.

1. Zwykła kwadratura prostokatów jest tożsama ze złożoną kwadraturą prostokątów dla $n = 1$
2. Zwykla kwadratura trapezów jest tożsama ze złożoną kwadraturą trapezów dla $n = 2$
3. Zwykla kwadratura Simpsona jest tożsama ze złożoną kwadraturą Simpsona dla $n = 3(k = 1)$

Obliczenia powyższych kwadratur wykonałem przy pomocy poniższego programu w Pythonie:

```
import numpy as np
from math import log


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


def rectangle_method():
    print("Metoda prostokątów")
    for steps in [1, 3, 5]:
        length = 1
        xs = np.linspace(0, 0 + length, num=steps + 1, endpoint=False)
        Int = sum(map(lambda x: f(x) * length / (steps), xs[1::]))
        print("n = ", steps, ":", Int, "blad bezwzgledny: ", Int - log(2))


def trapeze_method():
    print("Metoda trapezów")
    for steps in [2, 3, 5]:
        length = 1
        h = length / (steps - 1)
        xs = np.linspace(0, 0 + length, num=steps, endpoint=True)
        Int = sum(map(lambda x: (f(x) - f(x - h)) * h / 2, xs[1::]))
        print("n = ", steps, ":", Int, "blad bezwzgledny: ", Int - log(2))


def simpson_method():
    print("Metoda Simpsona")
    for steps in [3, 5]:
        length = 1
        h = length / (steps - 1)
        xs = np.linspace(0, 0 + length, num=steps // 2 + 1, endpoint=True)
        Int = sum(map(lambda x: (f(x) + 4 * f(x - h) + f(x - 2 * h)) * (h / 3), xs[1::]))
        print("n = ", steps, ":", Int, "blad bezwzgledny: ", Int - log(2))
```

Otrzymano wyniki (odpowiednio dla metod prostokatów, trapezów, Simpsona):

```
Metoda prostokątów
n =  1 : 0.6666666666666666 blad bezwzgledny:  -0.026480513893278657
n =  3 : 0.6793650793650794 blad bezwzgledny:  -0.013782101194865892
n =  5 : 0.6838528138528138 blad bezwzgledny:  -0.009294366707131463
Metoda trapezów
n =  2 : -0.25 blad bezwzgledny:  -0.9431471805599453
n =  3 : -0.125 blad bezwzgledny:  -0.8181471805599453
n =  5 : -0.0625 blad bezwzgledny:  -0.7556471805599453
Metoda Simpsona
n =  3 : 0.6944444444444443 blad bezwzgledny:  0.0012972638844990225
n =  5 : 0.6932539682539682 blad bezwzgledny:  0.0001067876940229473
```

2.2 Obliczyć całkę $\mathrm{I} = {}_{-1}\int^{1} 1/\left(1+\mathrm{x}^2\right)$ dx korzystając z wielomianów ortogonalnych (np. Legendre'a) dla $\mathrm{n} = 8$.

Dokładna wartość naszej całki wynosi:

$$I = \int_{-1}^{1} 1/\left(1+x^2\right) dx = \operatorname{arctg}(x)|_{-1}^{1} = \frac{\pi}{2}$$

Wykorzystuje wielomiany Legendre'a, które zadane są wzorem rekurencyjnym:

$$\begin{gathered}(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \\ P_0(x) = 1 \\ P_1(x) = x\end{gathered}$$

Są one ortogonalne na przedziale całkowania $[-1, 1]$ z wagą $w(x) = 1$.

Całkę będziemy przybliżać całką wielomianu aproksymującego szukaną funkcję, w naszym przypadku funkcja aproksymująca będzie mieć postać:

$$F(x) = \sum_{i=0}^{2} c_i P_i(x), \quad x \in [-1, 1]$$

Wyznaczam współczynniki c_i ze wzoru:

$$c_i = \frac{\int_{-1}^{1} f(x)L_i(t)dx}{\int_{-1}^{1} L_i^2(x)dx}$$

Następnie przy pomocy programu napisanego w języku Python wygenerowałem wielomiany Legendre'a (def genLegendre), wyznaczyłem współczynniki c_i (def approxF). Kod poniżej.

```
import numpy as np
from scipy.integrate import quadrature
def f(x):
    return 1/(1+x**2)
def genLegendre(n):
    p = []
    p.append(np.poly1d([1]))
    p.append(np.poly1d([1,0]))
    x = np.poly1d([1,0])
    for i in range(1,n):
        p.append((2*i+1)/(i+1)*p[i]*x-(i/(i+1))*p[i-1])
    return p
def approxF(n):
    L = genLegendre(n)
    c = []
    fA = []
    fA.append(np.poly1d([0]))
    for i in range(0,n):
```

```
        c.append(quadrature((lambda x: f(x)*L[i](x)), -1,1)[0]/quadrature((lambda x: L[i](x)**2),
        fA.append(c[i]*L[i])
    return sum(fA)
def Int(n):
    i = approxF(n).integ()
    return i(1) - i(-1)
```

Dzięki funkcji genLegendre() otrzymuję funkcję aproksymującą, której całka będzie przybliżeniem całki, którą mam obliczyć. Wynik programu:

```
Wartość obliczona:  1.5707963251251216 błąd bezwględny:  -1.6697749849470256e-09
```

3 Rozwiązania - zadania domowe

3.1 Obliczyć całkę $_0\int^1 1/\left(1+\mathrm{x}^2\right)$ dx korzystając ze wzoru prostokątów, trapezów i wzoru Simpsona dla $\mathrm{h}=0.1$

Dokładna wartość całki:

$$\int_0^1 \frac{1}{(1+x^2)} = \tan^{-1}(x) = \frac{\pi}{4} \approx 1.5707963267948966192$$

Analogicznie do zadania 1, po niewielkich modyfikacjach, korzystam z przygotowanego wcześniej programu:

```
import numpy as np
from math import log
def f(x):
    return (1/(1+x**2))
def rectangle_method():
    length = 1
    h=0.1
    xs = np.linspace(0, 1, num=10, endpoint=False)
    Int = sum(map(lambda x: f(x)*h, xs[1::]))
    print("wartość: ", Int, "bład bezwględny: ", Int - np.pi/4)
def trapez_method():
    length = 1
    h = 0.1
    xs = np.linspace(0, 1, num=11, endpoint=True)
    Int = sum(map(lambda x: (f(x)+f(x-h))*h/2, xs[1::]))
    print("wartość: ", Int, "bład bezwględny: ", Int - np.pi/4)
def simpson_method():
    length = 1
    h = 0.1
    xs = np.linspace(0, 1, num=6, endpoint=True)
    Int = sum(map(lambda x: (f(x)+4*f(x-h)+f(x-2*h))*(h/3), xs[1::]))
    print("wartość: ", Int, "bład bezwględny: ", Int - np.pi/4)
```

Otrzymuję następujące wyniki (odpowiednio dla metod prostokątów, trapezów i Simpsona):

```
Metoda prostokątów
wartość:  0.7099814972267896 bład bezwględny:  -0.07541666617065867
Metoda trapezów
wartość:  0.7849814972267897 bład bezwględny:  -0.00041666617065860834
Metoda Simpsona
wartość:  0.7853981534848038 bład bezwględny:  -9.912644483023314e-09
```

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.

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

Dokładna wartość całki:

$$\int_0^1 \frac{1}{(x+3)} = ln(4) - ln(3) \approx 0.28768207245178092743$$

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

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

Rys. 1: Wagi oraz pierwiastki 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.

4 Bibliografia

1. Katarzyna Rycerz: Materiały wykładowe z przedmiotu Metody Obliczeniowe w Nauce i Technice
2. https://www.wolframalpha.com/
3. https://pomax.github.io/bezierinfo/legendre-gauss.html
4. https://en.wikipedia.org/wiki/Numerical_integration