Metody Obliczeniowe i Symulacja, Laboratorium 5

Całkowanie Numeryczne (cz. I i II), Kamil Klimowicz, 2020-12-05

1. Zadania

Tematem zadania będzie obliczanie różnymi sposobami całki funkcji x^2 + x + 1 oraz 1/sqrt(x) w przedziale (0,1).

Cz. I:

1. Znaleźć dokładną wartość całki (całkując ręcznie)
2. Napisać program obliczający całkę metodą prostokątów. Program powinien mieć następujące parametry:
    1. float a - początek przedziału
    2. float b - koniec przedziału
    3. int n - ilość podprzedziałów, na które dzielimy przedział (a,b).
3. Zbadać, przy użyciu programu z poprzedniego punktu, jak zmienia się błąd całkowania wraz ze wzrostem liczby podprzedziałów. Kiedy błąd jest mniejszy niż $1e^{-3}$, $1e^{-4}$, $1e^{-5}$ i $1e^{-6}$?

Cz. II:

4. Obliczyć wartość całki korzystając z funkcji **gsl_integration_qag** metodą GSL_INTEG_GAUSS15 dla zadanych dokładności takich jak w p. 3. Sprawdzić, ile przedziałów (*intervals*) potrzebuje ta procedura, aby osiągnąć zadaną dokładność ($1e^{-3}$, $1e^{-4}$, $1e^{-5}$ i $1e^{-6}$). Porównać, ile przedziałów potrzebuje metoda prostokątów do osiągniecia podobnej dokładności. Patrz przykład w dokumentacji GSL.
5. Wykorzystać funkcję **gsl_integration_qag()** do obliczenia całki z wartościami błędów mniejszymi niż $1e^{-3}$, $1e^{-4}$, $1e^{-5}$ i $1e^{-6}$ dla funkcji:
    1. sin(x) na przedziale [0, pi]
    2. tan(x) na przedziale (0, pi/2]
    3. log(x+x^2) na przedziale [1,4]

2. Podejście do rozwiązania

Rozwiązania zostały wywołane/zaimplementowane w środowisku operacyjnym Linux Mint, Windows, wykorzystując języki skrypty odpowiednio przygotowane w C oraz Pyton, pozwalające na numeryczne obliczenia równań całkowych. Przygotowane pliki zostały skompilowane w ww systemach operacyjnych.

3. Wyniki

Cz. I:

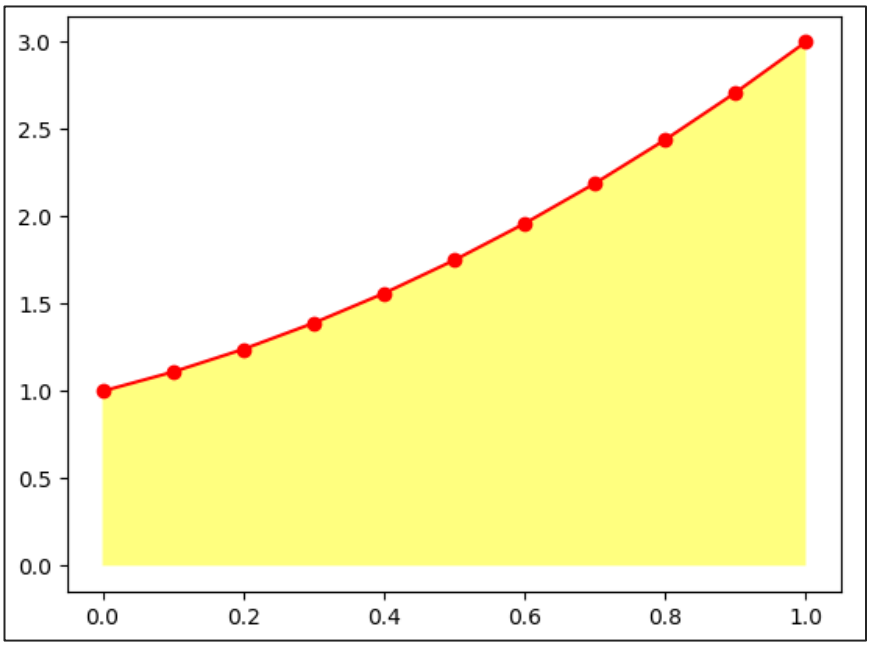
1. Dokładna wartość całki wynosi:

$$\int_0^1 (x^2 + x + 1)dx = \int_0^1 \left(\left(x + \frac{1}{2}\right)^2 + \frac{3}{2}\right)dx = \int_0^1 \left(x + \frac{1}{2}\right)^2 dx + \int_0^1 \frac{3}{4}dx =$$
$$= \frac{1}{3} \cdot \left|\left(x + \frac{1}{2}\right)^3\right|_0^1 + \left|\frac{3}{4}x\right|_0^1 =$$
$$= \frac{1}{3} \cdot \left(\frac{27}{8} - \frac{1}{8}\right) + \frac{3}{4} =$$
$$= \frac{11}{6} \approx 1.8333 \ldots$$

2. Program realizujący całkowanie funkcji z zadania:

```
import numpy as np
import cv2

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

def calcInterval(a, step):
   return 0.5 * f(a) + f(a+step) * step

def calcIntegral (a, n, step):
   integral = 0
   for i in range(n):
      integral = calcInterval(a, step)
      a = a + step
   return integral

def main():
   a = 0
   b = 1
   n = 10
   step = (b-a) / n
   print(calcIntegral(a, n, step))

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

Rys.1. Kod źródłowy.



Rys.2. Wykres argumentów i wartości funkcji.

3. Analiza błędu całkowania została zamieszczona w poniższej tabeli.

W analizie została założona dokładna wartość całki S(f) wyznaczona ręcznie: $S(F) = \frac{11}{6}$ wyrażona w zmiennej typu float. Błąd przybliżenia został wyrażony w postaci różnicy wartości dokładnej i wynikowej: $S(F) - Q(f)$.

Tabela 1. Wyniki błędu całkowania dla różnych wielkości podziałów funkcji całkowanej.

n	2	4	6	8	10	20	50
Err	-0.5417	-0.0729	0.0694	0.1380	0.1783	0.2571	0.3031
		$1e^{-3}$=0.0498		$1e^{-4}$=0.0182		$1e^{-5}$=0.0069	$1e^{-6}$=0.0025

Legenda:

n – ilość przedziałów

Err – błąd przybliżenia

Zaimplementowane rozwiązanie nie pozwala na uzyskanie dokładności poniżej e^{-6}. Najlepsza wartość przybliżenia wynosi e^{-2}. Co więcej wraz ze wzrostem liczby przedziałów, wartość błędu rośnie. Jest to

wynikiem niedokładności powodowanej przez zwiększającą się liczbę iteracji przy rosnącej liczbie przedziałów.

Cz. II:

4. Wyniki całkowania funkcji zrealizowane z wykorzystaniem biblioteki **gsl_integration_qag.**

Wykorzystana zasada kwadratury funkcji Gauss'a-Kronrod'a rzędu 1, zapewnia uzyskanie wyników obarczonych błędem poniżej e^{-10}. Znakomita rezultat jest uzyskany bez względu na przyjęty stopień podziału, gdyż przyjęta metoda wykorzystuje algorytm adaptacyjny.

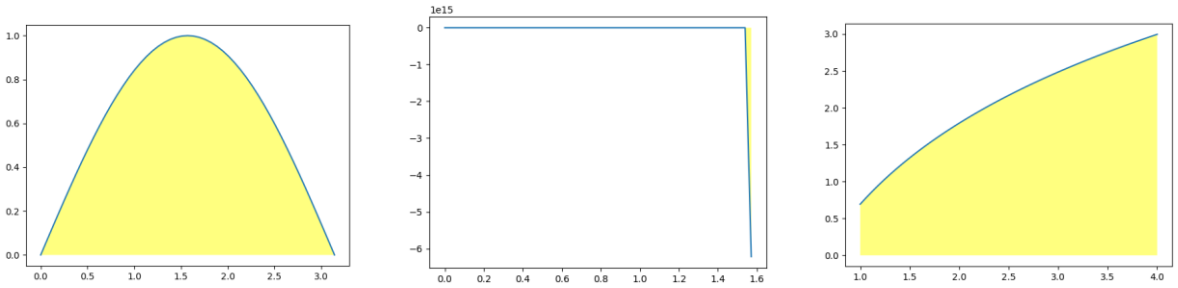
```c
#include <stdio.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_integration.h>

double f(double x, void *p) {
    return x*x + x + 1;
}

int main (void) {
    //int a=0, b=M_PI/2;
    int a = 1, b=4;
    int i, n=10;

    gsl_integration_workspace *iw = gsl_integration_workspace_alloc(10)

    gsl_function F;
    F.function = &f;
    F.params = 0;

    double result, error;

    double x = i / (double)n;
    gsl_integration_qag(&F,
                        a, b,
                        1e-3, 1e-3, n,
                        GSL_INTEG_GAUSS15,
                        iw,
                        &result, &error);

    printf ("result = % .20f\n", result);
    printf ("error = % .20f\n", error);
    //float Err = (11.0/6.0)  - result;
    //printf ("Err = % .20f\n", Err);

    gsl_integration_workspace_free (iw);

    return 0;
}
```

Rys.3. Implementacja biblioteki gsl_integration_qag.

5. Poniżej przedstawiono wyniki całkowania funkcji trygonometrycznych i logarytmicznej wykorzystując bibliotekę gsl_integration_qag.

Tabela 2. Funkcje całkowane i przyjęte wartości błędu ograniczenia.

		$1e^{-3}$	$1e^{-4}$	$1e^{-5}$	$1e^{-6}$
sin(x)	$\in$ [0,pi]	0.00000000000018318680			
tan(x)	$\in$ (0,pi/2]	0.00000000163700368514			
$\log(x+x^2)$	$\in$ [1,4]	0.00000000163700368514			



Rys.5. Funkcje całkowe z tabeli 2 dla zadanej dziedziny.

4. Wnioski

Metoda całkowania funkcji przez numeryczną analizę pola powierzchni funkcji dla zadanej dziedziny może wydawać się mało dokładna. Jednakże rozważając trudności w implementacji (a czasami wręcz całkowity jej brak) pochodnej funkcji podcałkowej, widzimy, że wspomniana metoda może okazać się zdecydowanie skuteczniejszą i jedyną dostępną opcją (szczególnie dla funkcji nieciągłych w całej dziedzinie).

Analizując stopień dokładności uzyskiwanych wyników w stosunku do metody klasycznej, widzimy, że popełniany błąd jest zależy od metody implementacji rozwiązania. Biblioteka GNU znakomicie rozwinięte metody adaptacyjne obliczania równań całkowych (szczególnie odnosząc się do własnej implementacji z wykorzystaniem aproksymacji trapezoidalnej).

5. Bibliografia

[1] https://artemis.wszib.edu.pl/~funika/mois/lab5/

[2] Wykład Systemy i metody obliczeniowe, Dr inż. Włodzimierz Funika

[3] http://www.gnuplot.info/

[4] https://www.wolframalpha.com

[5] http://wazniak.mimuw.edu.pl