Laboratorium 12

Całkowanie Monte Carlo

Adam Biśta, 06.06.2023

1 Treść zadań

Zadania
Tematem zadania będzie obliczanie metodami Monte Carlo całki funkcji:
1) $x^2 + x + 1$,
2) $\sqrt{1 - x^2}$
3) $\frac{1}{\sqrt{x}}$ w przedziale (0,1).
Proszę dla tych funkcji:

1. Napisać funkcję liczącą całkę metodą "hit-and-miss". Czy będzie ona dobrze działać dla funkcji 1/sqrt(x)?
2. Policzyć całkę przy użyciu napisanej funkcji. Jak zmienia się błąd wraz ze wzrostem liczby prób? Narysować wykres tej zależności przy pomocy Gnuplota. Przydatne będzie skala logarytmiczna.
3. Policzyć wartość całki korzystając z funkcji Monte Carlo z GSL. Narysować wykres zależności błędu od ilości wywołań funkcji dla różnych metod (PLAIN, MISER, VEGAS).

2 Rozwiązania zadań

Na początku policzmy wszystkie zadane całki (korzystam z programu Wolfram Alpha):

$$\int_0^1 (x^2 + x + 1)dx = \frac{11}{6} \approx 1.8333$$

$$\int_0^1 \sqrt{1 - x^2}, dx = \frac{\pi}{4} \approx 0.78540$$

$$\int_0^1 \frac{1}{\sqrt{x}}, dx = 2$$

1. **Metoda "hit-and-miss"**
   Metoda ta polega na losowym generowaniu punktów wewnątrz obszaru pod wykresem funkcji i liczeniu ilości punktów, które trafiają się pod wykresem. Dzięki temu możemy oszacować wartość całki.

   Poniżej prezentuję algorytm napisany w pythonie, który wyliczy całki metodą hit-and-miss:

```
 import numpy as np
from random import uniform
from math import sqrt

EPSILON = 10 ** -6

def f1(x: float) -> float:
    return x ** 2 + x + 1

def f2(x: float) -> float:
    return sqrt(1 - x ** 2)

def f3(x: float) -> float:
    if abs(x) >= EPSILON:
        return 1 / sqrt(x)
    else:
        return 1

def hit_and_miss(fun, a: float, b: float, N: int=1000, M:
↪   float=0.001) -> float:
    hits = 0
    h = max(fun(x) for x in np.arange(a, b, M))

    for i in range(N):
        x = uniform(a, b)
```

```
        y = uniform(0, h)

        if y <= fun(x):
            hits += 1

    return hits / N * (b - a) * h

if __name__ == "__main__":
    a = 0
    b = 1
    N = 1000
    M = 0.001

    result_f1 = hit_and_miss(f1, a, b, N, M)
    result_f2 = hit_and_miss(f2, a, b, N, M)
    result_f3 = hit_and_miss(f3, a, b, N, M)

    print("Dla f1(x) = x^2 + x + 1:")
    print("{:.10f}".format(result_f1))
    print("\nDla f2(x) = sqrt(1 - x^2):")
    print("{:.10f}".format(result_f2))
    print("\nDla f3(x) = 1 / sqrt(x):")
    print("{:.10f}".format(result_f3))
```

Należy zauważyć, że funkcja $f(x) = \frac{1}{\sqrt{x}}$ ma pionową asymptotę w punkcie $x_0 = 0$, a gdy x dąży do 0, wartość funkcji f dąży do $+\infty$. To powoduje trudności w wyborze odpowiedniej górnej granicy przedziału, z którego będziemy losować punkty metodą "hit-and-miss".

Dodatkowo, funkcja $f(x) = \frac{1}{\sqrt{x}}$ może prowadzić do problemów w obliczeniach numerycznych ze względu na operacje pierwiastkowania i dzielenia, które są podatne na błędy zaokrągleń. Funkcja taka jak na przykład $f(x) = x^2 + x + 1$ wymaga jedynie operacji mnożenia i dodawania, przez co jest mniej podatna na błędy numeryczne.

2.

Wyniki dla różnych N:

Tabela 1: Wyniki dla różnych wartości N

N	$f_1(x) = x^2 + x + 1$	$f_2(x) = \sqrt{1 - x^2}$	$f_3(x) = \frac{1}{\sqrt{x}}$
10	2.0979007000	0.7000000000	3.1622776602
100	2.0079906700	0.7600000000	2.8460498942
1000	1.8011976010	0.7670000000	1.7076299365
10000	1.8116871045	0.7897000000	1.9953972036
100000	1.8375212531	0.7837800000	1.9565011883
1000000	1.8348778982	0.7853170000	1.9615608326

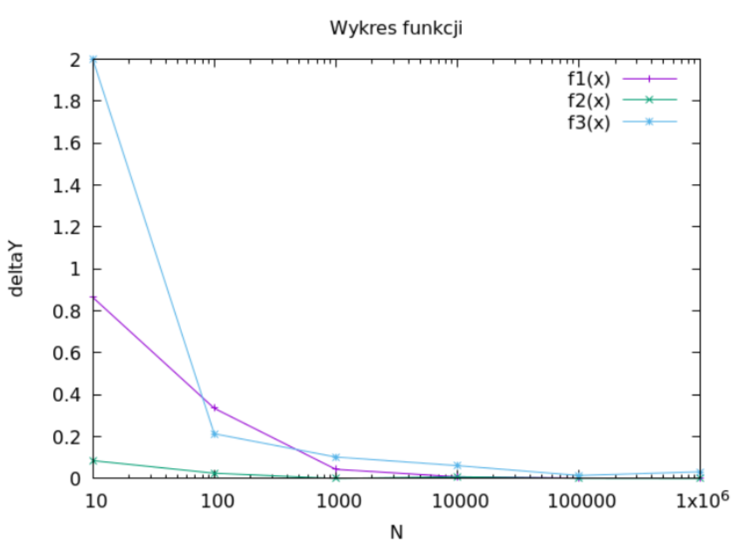
Zmodyfikowany program w pythonie:

```python
if __name__ == "__main__":
    a = 0
    b = 1

    N = 10
    M = 0.001
    real_res_f1:float = 11/6
    real_res_f2:float = pi/4
    real_res_f3:float = 2.0
    while N < 10e6:
        result_f1 = hit_and_miss(f1, a, b, N, M)
        result_f2 = hit_and_miss(f2, a, b, N, M)
        result_f3 = hit_and_miss(f3, a, b, N, M)

        print("N=", N)
        print("f1(x) = x^2 + x + 1:",result_f1,"real_f1:", real_res_f1, " deltaY: ", abs(result_f1 - real_res_f1))
        print("f2(x) = sqrt(1 - x^2):",result_f2,"real_f2:", real_res_f2, " deltaY: ", abs(result_f2 - real_res_f2))
        print("f3(x) = 1 / sqrt(x):",result_f3,"real_f3:", real_res_f3, " deltaY: ", abs(result_f3 - real_res_f3))
        print("====================================")
        N*=10
```

Wyniki

N	$f(x)$	Wartość	Wartość rzeczywista ΔY
10	$f_1(x) = x^2 + x + 1$	2.6973009	0.8639675666666669
	$f_2(x) = \sqrt{1 - x^2}$	0.7	0.08539816339744832
	$f_3(x) = 1/\sqrt{x}$	0.0	2.0
100	$f_1(x) = x^2 + x + 1$	1.4985005	0.33483283333333325
	$f_2(x) = \sqrt{1 - x^2}$	0.76	0.02539816339744827
	$f_3(x) = 1/\sqrt{x}$	2.213594362117866	0.21359436211786598
1000	$f_1(x) = x^2 + x + 1$	1.789209597	0.04412373633333333
	$f_2(x) = \sqrt{1 - x^2}$	0.785	0.0003981633974482479
	$f_3(x) = 1/\sqrt{x}$	1.8973665961010278	0.10263340389897224
10000	$f_1(x) = x^2 + x + 1$	1.8254733091	0.007860024233333318
	$f_2(x) = \sqrt{1 - x^2}$	0.7768	0.008598163397448233
	$f_3(x) = 1/\sqrt{x}$	1.9384762056832168	0.06152379431678323
100000	$f_1(x) = x^2 + x + 1$	1.83395482193	0.0006214885966666639
	$f_2(x) = \sqrt{1 - x^2}$	0.78468	0.0007181633974482349
	$f_3(x) = 1/\sqrt{x}$	1.9852779150537088	0.014722084946291236
1000000	$f_1(x) = x^2 + x + 1$	1.8328788985709998	0.00045443476233342217
	$f_2(x) = \sqrt{1 - x^2}$	0.785319	7.916339744828971e-05
	$f_3(x) = 1/\sqrt{x}$	1.967664028486571	0.03233597151342904

Za pomocą gnuplot stworzyłem następujący wykres zależności błędu od liczby prób:



Rysunek 1

Do stworzenia wykresu posłużyłem się następującymi komendami:

```
set term pngcairo
set output 'output.png'

set title "Wykres funkcji"
set xlabel "N"
set ylabel "deltaY"
set logscale x

plot '-' using 1:2 title 'f1(x)' with linespoints, '-' using
↪  1:2 title 'f2(x)' with linespoints, '-' using 1:2 title
↪  'f3(x)' with linespoints
10 0.8639675666666669
100 0.33483283333333325
1000 0.04412373633333333
10000 0.007860024233333318
100000 0.0006214885966666639
1000000 0.00045443476233342217
e
10 0.08539816339744832
100 0.02539816339744827
1000 0.0003981633974482479
10000 0.008598163397448233
100000 0.0007181633974482349
1000000 7.916339744828971e-05
e
10 2.0
100 0.21359436211786598
1000 0.10263340389897224
10000 0.06152379431678323
100000 0.014722084946291236
1000000 0.03233597151342904
e
```

3. Do obliczenia całek wykorzystamy następujący program w języku c:

```
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_integration.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_monte.h>
#include <gsl/gsl_monte_plain.h>
#include <gsl/gsl_monte_miser.h>
#include <gsl/gsl_monte_vegas.h>

double f1 (double *x, size_t dim, void *params) {
    return x[0] * x[0] + x[0] + 1;
```

```
}

double f2 (double *x, size_t dim, void *params) {
    return sqrt(1 - x[0] * x[0]);
}

double f3 (double *x, size_t dim, void *params) {
    return 1 / sqrt(x[0]);
}

void monte_carlo(int N, double (*func)(double *, size_t,
↪   void *)) {
    double res, err;
    double xl[1] = {0};
    double xu[1] = {1};
    const gsl_rng_type *T;
    gsl_rng *r;
    gsl_monte_function G = { func, 1, 0 };

    gsl_rng_env_setup ();

    T = gsl_rng_default;
    r = gsl_rng_alloc (T);

    {
        gsl_monte_plain_state *s = gsl_monte_plain_alloc
↪   (1);
        gsl_monte_plain_integrate (&G, xl, xu, 1, N, r, s,
↪   &res, &err);
        gsl_monte_plain_free (s);

        printf ("PLAIN RESULT: %.8f +/- %.8f (estimated
↪   error)\n", res, err);
    }

    {
        gsl_monte_miser_state *s = gsl_monte_miser_alloc
↪   (1);
        gsl_monte_miser_integrate (&G, xl, xu, 1, N, r, s,
↪   &res, &err);
        gsl_monte_miser_free (s);

        printf ("MISER RESULT: %.8f +/- %.8f (estimated
↪   error)\n", res, err);
    }
```

```
    {
        gsl_monte_vegas_state *s = gsl_monte_vegas_alloc
↪   (1);
        gsl_monte_vegas_integrate (&G, xl, xu, 1, N, r, s,
↪   &res, &err);
        gsl_monte_vegas_free (s);

        printf ("VEGAS RESULT: %.8f +/- %.8f (estimated
↪   error)\n", res, err);
    }

    gsl_rng_free (r);
}

int main(void) {
    for (int N = 10; N <= 1000000; N *= 10) {
        printf("N = %d\n", N);
        printf("Function f1(x) = x^2 + x + 1\n");
        monte_carlo(N, f1);
        printf("\nFunction f2(x) = sqrt(1 - x^2)\n");
        monte_carlo(N, f2);
        printf("\nFunction f3(x) = 1 / sqrt(x)\n");
        monte_carlo(N, f3);
        printf("\n------------------------\n");
    }
    return 0;
}
```

Wyniki wywołania: N = 10
Function $f1(x) = x^2 + x + 1$
PLAIN RESULT: 2.06927598 +/- 0.21807705 (estimated error)
MISER RESULT: 1.70606768 +/- 0.14412567 (estimated error)
VEGAS RESULT: 1.83557493 +/- 0.03615733 (estimated error)

Function $f2(x) = sqrt(1 - x^2)$
PLAIN RESULT: 0.66072102 +/- 0.10711350 (estimated error)
MISER RESULT: 0.85371327 +/- 0.04479919 (estimated error)
VEGAS RESULT: 0.74295996 +/- 0.00469245 (estimated error)

Function $f3(x) = 1/sqrt(x)$
PLAIN RESULT: 1.46042346 +/- 0.16241692 (estimated error)
MISER RESULT: 1.64334697 +/- 0.14719164 (estimated error)
VEGAS RESULT: 1.25572423 +/- 0.16237665 (estimated error)

N = 100
Function $f1(x) = x^2 + x + 1$
PLAIN RESULT: 1.70763599 +/- 0.05183088 (estimated error)
MISER RESULT: 1.83709096 +/- 0.05850292 (estimated error)
VEGAS RESULT: 1.83368262 +/- 0.00063653 (estimated error)

Function $f2(x) = sqrt(1 - x^2)$
PLAIN RESULT: 0.83565228 +/- 0.01928683 (estimated error)
MISER RESULT: 0.78148434 +/- 0.02341260 (estimated error)
VEGAS RESULT: 0.78518177 +/- 0.00024031 (estimated error)

Function $f3(x) = 1/sqrt(x)$
PLAIN RESULT: 1.87182739 +/- 0.09291807 (estimated error)
MISER RESULT: 2.54539879 +/- 0.64165755 (estimated error)
VEGAS RESULT: 1.99859895 +/- 0.00329076 (estimated error)

N = 1000
Function $f1(x) = x^2 + x + 1$
PLAIN RESULT: 1.83576236 +/- 0.01823297 (estimated error)
MISER RESULT: 1.82970029 +/- 0.00704102 (estimated error)
VEGAS RESULT: 1.83335516 +/- 0.00001646 (estimated error)

Function $f2(x) = sqrt(1 - x^2)$
PLAIN RESULT: 0.78492993 +/- 0.00707491 (estimated error)
MISER RESULT: 0.78133830 +/- 0.00512957 (estimated error)
VEGAS RESULT: 0.78538844 +/- 0.00000708 (estimated error)

Function $f3(x) = 1/sqrt(x)$
PLAIN RESULT: 1.95655087 +/- 0.07780969 (estimated error)
MISER RESULT: 2.01028714 +/- 0.04871837 (estimated error)
VEGAS RESULT: 1.99872907 +/- 0.00014435 (estimated error)

N = 10000
Function $f1(x) = x^2 + x + 1$
PLAIN RESULT: 1.83685152 +/- 0.00582712 (estimated error)
MISER RESULT: 1.83293303 +/- 0.00039965 (estimated error)
VEGAS RESULT: 1.83333324 +/- 0.00000052 (estimated error)

Function $f2(x) = sqrt(1 - x^2)$
PLAIN RESULT: 0.78376798 +/- 0.00225178 (estimated error)
MISER RESULT: 0.78504926 +/- 0.00039241 (estimated error)
VEGAS RESULT: 0.78539819 +/- 0.00000022 (estimated error)

Function $f3(x) = 1/sqrt(x)$

PLAIN RESULT: 2.02831391 +/- 0.04969080 (estimated error)
MISER RESULT: 1.99812190 +/- 0.01049800 (estimated error)
VEGAS RESULT: 1.99996378 +/- 0.00001869 (estimated error)

N = 100000
Function $f1(x) = x^2 + x + 1$
PLAIN RESULT: 1.83205070 +/- 0.00183895 (estimated error)
MISER RESULT: 1.83335301 +/- 0.00002037 (estimated error)
VEGAS RESULT: 1.83333332 +/- 0.00000002 (estimated error)

Function $f2(x) = sqrt(1 - x^2)$
PLAIN RESULT: 0.78585744 +/- 0.00070598 (estimated error)
MISER RESULT: 0.78540224 +/- 0.00004445 (estimated error)
VEGAS RESULT: 0.78539817 +/- 0.00000001 (estimated error)

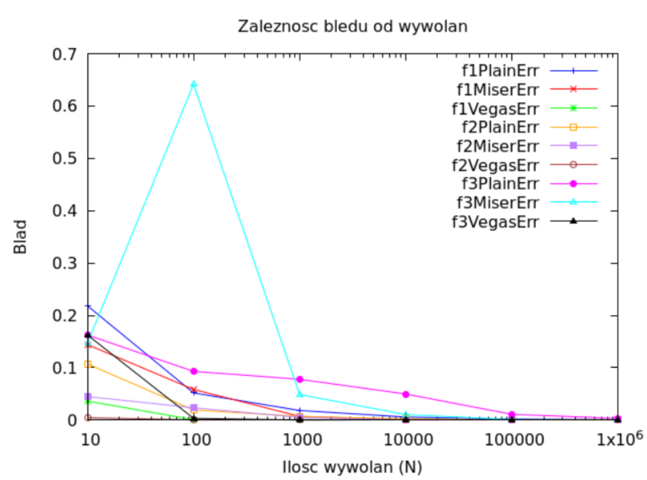
Function $f3(x) = 1/sqrt(x)$
PLAIN RESULT: 1.99822188 +/- 0.01093539 (estimated error)
MISER RESULT: 1.99763911 +/- 0.00110308 (estimated error)
VEGAS RESULT: 1.99996800 +/- 0.00000260 (estimated error)

N = 1000000
Function $f1(x) = x^2 + x + 1$
PLAIN RESULT: 1.83301876 +/- 0.00058199 (estimated error)
MISER RESULT: 1.83333298 +/- 0.00000096 (estimated error)
VEGAS RESULT: 1.83333333 +/- 0.00000000 (estimated error)
Function $f2(x) = sqrt(1 - x^2)$
PLAIN RESULT: 0.78552354 +/- 0.00022316 (estimated error)
MISER RESULT: 0.78540126 +/- 0.00000450 (estimated error)
VEGAS RESULT: 0.78539816 +/- 0.00000000 (estimated error)
Function $f3(x) = 1/sqrt(x)$
PLAIN RESULT: 1.99930684 +/- 0.00319187 (estimated error)
MISER RESULT: 2.00066247 +/- 0.00036835 (estimated error)
VEGAS RESULT: 1.99999541 +/- 0.00000088 (estimated error)

Aby stworzyć wykres, muszę stworzyć plik dane.txt z powyższymi danymi dla wszystkich dziewięciu funkcji (mamy 3 funkcje i dla każdej sprawdzamy, jak się zachowa dla wybranej metody całkowania). Ustawię sobie skalę logarytmmiczną dla osi X, gdyż wykres stanie się dzięki temu zabiegowi czytelniejszy.

4.
```
# wykres.gnu
set terminal pngcairo enhanced
set output 'wykres.png'

set title 'Zaleznosc bledu od ilosci wywołan'
```

```
set xlabel 'Ilość wywołań (N)'
set ylabel 'Błąd'

set logscale x

set key autotitle columnheader

plot 'dane.txt' using 1:2 with linespoints lc rgb 'blue' pt
↪  1 ps 1, \
     '' using 1:3 with linespoints lc rgb 'red' pt 2 ps 1, \
     '' using 1:4 with linespoints lc rgb 'green' pt 3 ps 1,
↪  \
     '' using 1:5 with linespoints lc rgb 'orange' pt 4 ps
↪  1, \
     '' using 1:6 with linespoints lc rgb 'purple' pt 5 ps
↪  1, \
     '' using 1:7 with linespoints lc rgb 'brown' pt 6 ps 1,
↪  \
     '' using 1:8 with linespoints lc rgb 'magenta' pt 7 ps
↪  1, \
     '' using 1:9 with linespoints lc rgb 'cyan' pt 8 ps 1,
↪  \
     '' using 1:10 with linespoints lc rgb 'black' pt 9 ps 1
```



Rysunek 2

Wnioski:

(a) Metoda Monte Carlo jest potężnym narzędziem do numerycznego przybliżania wartości całek, szczególnie w przypadku funkcji, dla których trudno jest znaleźć analityczne rozwiązania.

(b) Metoda PLAIN (prosta metoda Monte Carlo) polega na losowaniu punktów wewnątrz przedziału całkowania i obliczaniu wartości funkcji w tych punktach. Jest to najprostsza do zaimplementowania metoda, ale może wymagać dużego N (liczby prób) dla uzyskania dokładnych wyników.

(c) Metoda MISER (metoda Monte Carlo z adaptacyjnym podziałem) stosuje adaptacyjny podział przedziału całkowania na podprzedziały w celu zoptymalizowania próbkowania. Dzięki temu można uzyskać dokładniejsze wyniki przy mniejszej liczbie prób niż w przypadku metody PLAIN.

(d) Metoda VEGAS (metoda Monte Carlo z adaptacyjnym podziałem z wagami) również stosuje adaptacyjny podział przedziału całkowania, ale uwzględnia również wagę próbek w celu lepszego dopasowania do kształtu funkcji. Jest to najbardziej zaawansowana i efektywna metoda, która może osiągnąć bardzo dokładne wyniki przy stosunkowo niewielkiej liczbie prób.

(e) Wszystkie trzy metody mają swoje zalety i wady, i wybór odpowiedniej metody zależy od konkretnego przypadku i oczekiwanego poziomu dokładności. Metoda VEGAS jest zazwyczaj preferowaną metodą, jeśli dostępne są wystarczające zasoby obliczeniowe, ponieważ może osiągnąć najmniejsze błędy przybliżenia. Metoda MISER jest dobrym kompromisem między dokładnością a efektywnością obliczeniową, szczególnie dla funkcji o skomplikowanych kształtach. Metoda PLAIN jest najprostszą do zaimplementowania, ale może dawać mniej precyzyjne wyniki niż MISER i VEGAS.

(f) W przypadku bardziej złożonych funkcji, których kształt może wpływać na efektywność próbkowania, istnieje możliwość konieczności dostosowania parametrów metody (np. liczby prób, podziału przedziału całkowania) w celu uzyskania optymalnych wyników.

Podsumowując, metody Monte Carlo z GSL (PLAIN, MISER, VEGAS) są wszechstronnymi narzędziami do numerycznego obliczania całek. Wybór odpowiedniej metody zależy od charakterystyki funkcji, dostępnych zasobów obliczeniowych i oczekiwanego poziomu dokładności. Dzięki tym metodom można uzyskać przybliżone wartości całek dla różnych funkcji bez konieczności znajomości ich analitycznych rozwiązań.

3 Bibliografia

- Katarzyna Rycerz: "Wykład z przedmiotu Metody Obliczeniowe w Nauce i Technice"
- https://home.agh.edu.pl/ funika/mownit/lab12/
- https://www.gnu.org/software/gsl/doc/html/montecarlo.html
- https://www.taygeta.com/rwalks/node3.html
- https://mathworld.wolfram.com/MonteCarloMethod.html