Laboratorium 11 - Całkowanie Monte Carlo

Piotr Karamon

11.06.2024r.

Treści zadań

Tematem zadania będzie obliczanie metodą Monte Carlo całki funkcji:

1. $f(x) = x^2 + x + 1$
2. $g(x) = \sqrt{1 - x^2}$
3. $h(x) = \frac{1}{\sqrt{x}}$

w przedziale $(0, 1)$. Proszę dla tych funkcji:

1. Napisać funkcję liczącą całkę metodą hit-and-miss. Czy będzie ona dobrze działać dla funkcji $\frac{1}{\sqrt{x}}$?
2. Policzyć całkę przy użyciu napisanej funkcji. Jak zmienia się błąd wraz ze wzrostem liczby prób?
3. Policzyć wartość całki korzystając ze wzoru prostokątów dla dokładności (1e-3, 1e-4, 1e-5 i 1e-6). Porównać czas obliczenia całki metodą Monte Carlo i przy pomocy wzoru prostokątów dla tej samej dokładności, narysować wykres. Zinterpretować wyniki.

Rozwiązanie

Na początku definiujemy nasze funkcję oraz obliczamy dokładne wartości ich całek na przedziale (0,1) przy użyciu biblioteki sympy.

Zdefiniowanie potrzebnych funkcji i stałych

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

def g(x):
    return np.sqrt(1-x**2)

def h(x):
    return 1/np.sqrt(x)
```

```
x = sp.symbols('x')

f_int = float(sp.integrate(x**2 + x + 1, (x, 0,1)))
g_int = float(sp.integrate(sp.sqrt(1-x**2), (x, 0,1)))
h_int = float(sp.integrate(1/sp.sqrt(x), (x, 0,1)))


for f_name, integral in zip(['f', 'g', 'h'], [f_int, g_int, h_int]):
    print(f'I({f_name}) = {integral}')
```

```
I(f) = 1.8333333333333333
I(g) = 0.7853981633974483
I(h) = 2.0
```

Opis metody hit-and-miss

Metoda Monte Carlo hit-and-miss służy do numerycznego obliczania całek. Polega na losowym generowaniu punktów i sprawdzaniu, ile z nich znajduje się pod krzywą funkcji. Oto kroki tej metody:

1. **Wybór przedziału**: Wybieramy przedział całkowania $[a, b]$ oraz wartość maksymalną funkcji $f(x)$ w tym przedziale, czyli $y_{\max}$.
2. **Generowanie punktów**: Losujemy N punktów (x, y) w prostokącie ograniczonym przez a, b oraz $0, y_{\max}$.

3. **Sprawdzanie punktów**: Sprawdzamy, ile z wygenerowanych punktów znajduje się pod wykresem funkcji, tzn. spełnia warunek $y < f(x)$.

4. **Obliczanie całki**: Całkę przybliżamy jako

$$I \approx \hat{I} = \frac{\text{Liczba punktów pod wykresem}}{\text{Całkowita liczba punktów}} \times \text{Pole prostokąta}$$

Implementacja metody hit-and-miss

Funkcja realizująca metodę hit-and-miss.

```
def hit_and_miss(f, a, b, n):
    np.random.seed(42)
    xs = np.random.uniform(a, b, n)
    y_max = max(f(xs))
    ys = np.random.uniform(0, y_max, n)

    hits = ys < f(xs)
    return hits.sum() / n * (b-a) * y_max
```

Czy metoda zadziała dla $\frac{1}{\sqrt{x}}$?

Metoda Monte Carlo hit-and-miss napotka problemy przy obliczaniu całki z funkcji $\frac{1}{\sqrt{x}}$ na przedziale $(0, 1)$. Główne powody to:

1. **Asymptota w punkcie $\mathbf{x = 0}$**: Funkcja $\frac{1}{\sqrt{x}}$ dąży do nieskończoności, gdy x zbliża się do 0. To sprawia, że oszacowanie wartości maksymalnej funkcji $y_{\max}$ staje się niepraktyczne. Ponieważ $y_{\max}$ jest nieskończona, nie można wygenerować losowych punktów w ograniczonym zakresie y, co jest niezbędne do zastosowania metody hit-and-miss

2. **Błędy obliczeniowe wynikające ze wzoru**: By obliczyć wartość $\frac{1}{\sqrt{x}}$ musimy obliczyć pierwiastek a następnie wykonać dzielenie, te operacje w arytmetyce zmiennoprzecinkowej powodują o wiele większe błędy, niż np. samo dodawanie i mnożenie jak w przypadku wielomianów.

Metoda zwróci wynik sensowny, jednakże obarczony sporym błędem. Metoda bardzo możliwe, że nie będzie zbieżna dla tej funkcji(w raz ze wzrostem N błąd nie będzie maleć).

Liczenie całek przy użyciu hit-and-miss

Obliczenia wykonujemy dla różnych wartości n i porównujemy wyniki z analitycznie policzonymi wartościami całek.

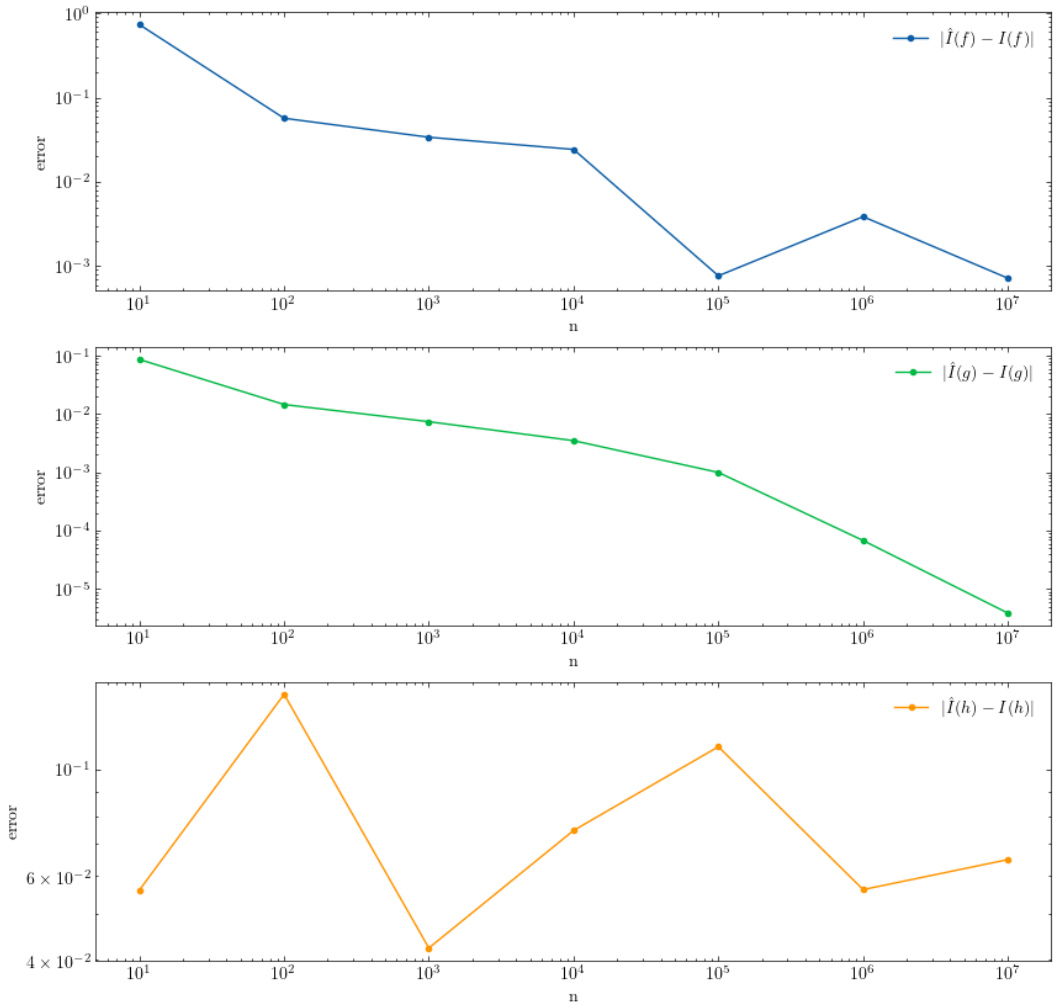
Dla przypomnienia oznaczenia funkcji:

$$f(x) = x^2 + x + 1$$
$$g(x) = \sqrt{1 - x^2}$$
$$h(x) = \frac{1}{\sqrt{x}}$$

Tablica 1: Wyniki obliczeń całek metodą hit-and-miss oraz porównanie z rzeczywistymi wartościami całek

n	$\hat{I}(f)$	$\lvert\hat{I}(f) - I(f)\rvert$	$\hat{I}(g)$	$\lvert\hat{I}(g) - I(g)\rvert$	$\hat{I}(h)$	$\lvert\hat{I}(h) - I(h)\rvert$
10	2.569115	0.735781	0.698818	0.086580	2.055964	0.055964
100	1.776500	0.056834	0.799988	0.014590	1.856292	0.143708
1000	1.799492	0.033841	0.777992	0.007407	1.957624	0.042376
10000	1.809089	0.024244	0.788900	0.003502	1.925381	0.074619
100000	1.832565	0.000768	0.784400	0.000998	1.888359	0.111641
1000000	1.837197	0.003864	0.785466	0.000068	1.943869	0.056131
10000000	1.832615	0.000719	0.785394	0.000004	1.935147	0.064853

Narysujemy teraz wykres dla każdej funkcji zależności pomiędzy błędem a n w skali log-log.



Rysunek 1: Wykresy obrazujące zależność błędu od n dla każdej z funkcji.

Widzimy, że metoda jest wolno zbieżna. Metoda najlepiej radzi sobie z funkcją g jednakże błędy rzędu 10^{-5} są nadal znaczące. Metoda hit-and-miss nie jest dobrym wyborem jeżeli zależy nam na szybkości oraz dokładności. Dla funkcji h metoda nie jest zbieżna. Błąd zamiast spadać z większą liczbą iteracji, zachowuje się w sposób mniej lub bardziej losowy. To oznacza, że choć metoda hit-and-miss jest bardzo ogólna nie zawsze jest ona zbieżna, w szczególności w przypadku funkcji z osobliwościami.

Porównanie z metodą prostokątów

Funkcja realizująca metodę prostokątów.

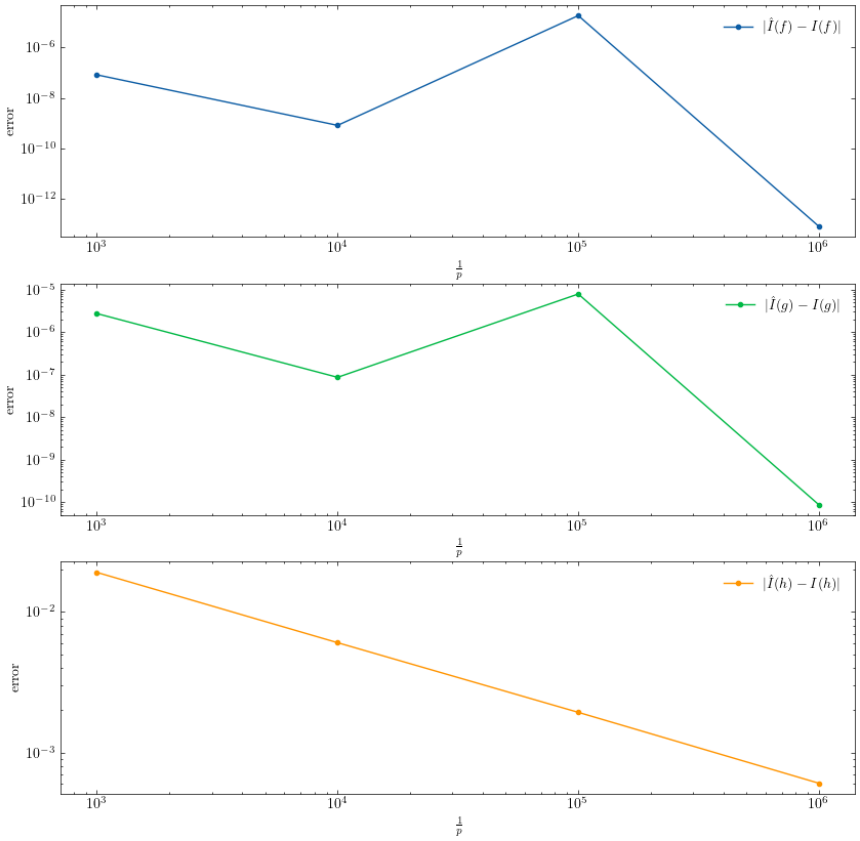
```
def rectangular_rule(f, a, b, h):
    n = int((b-a)/h)
    xs = np.linspace(a+h/2, b-h/2, n)
    return h * np.sum(f(xs))
```

Tworzymy analogiczną tabelę jak w przypadku metody hit-and-miss. p to precyzja.

Tablica 2: Wyniki obliczeń całek metodą prostokątów oraz porównanie z rzeczywistymi wartościami całek.

$\frac{1}{p}$	$\hat{I}(f)$	$\lvert\hat{I}(f) - I(f)\rvert$	$\hat{I}(g)$	$\lvert\hat{I}(g) - I(g)\rvert$	$\hat{I}(h)$	$\lvert\hat{I}(h) - I(h)\rvert$
1000.0	1.8333332	8.3333333e-08	0.78540089	2.7228308e-06	1.9808714	0.019128554
10000.0	1.8333333	8.3333318e-10	0.78539825	8.6108711e-08	1.993951	0.0060489862
100000.0	1.833315	1.8333325e-05	0.78539031	7.8512871e-06	1.9980672	0.0019328103
1000000.0	1.8333333	8.3266727e-14	0.78539816	8.6108787e-11	1.9993951	0.00060489864

Jak widzimy metoda prostokątów daje nam mniejsze błędy niż metoda hit-and-miss. Podobnie jak poprzednio rysujemy wykres błędów w tym przypadku od $1/p$



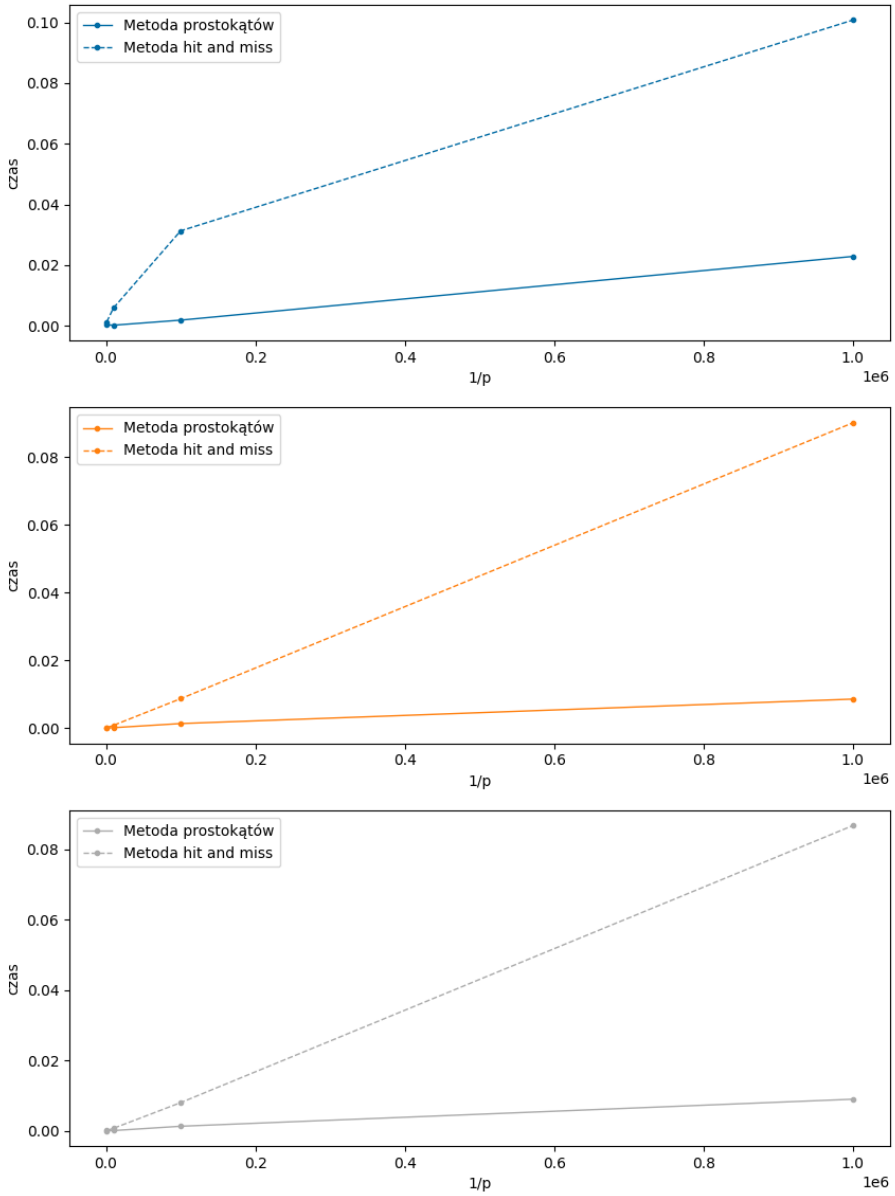
Porównanie czasu obliczenia

Tworzymy tabelę z wynikami pomiaru czasu:

Tablica 3: Wyniki pomiaru czasu dla obu metod. HM - hit-and-miss, Rect - metoda prostokątów. Czasy są podane w sekundach.

p	Rect I(f)	HM I(f)	Rect I(g)	HM I(g)	Rect I(h)	HM I(h)
0.001000	0.000843	0.002029	0.000147	0.000226	0.000077	0.000124
0.000100	0.000330	0.001325	0.000099	0.001022	0.000071	0.000811
0.000010	0.000862	0.010701	0.000649	0.008110	0.000380	0.007743
0.000001	0.008065	0.125657	0.008876	0.089052	0.004378	0.081908

W celu lepszego zobrazowania zależności wykonujemy wykres.



Rysunek 2: Wykresy pokazujące zależność czasu od odwrotności precyzji w skali log-log.

Metoda hit-and-miss jest wolniejsza niż metoda prostokątów.

Wnioski

1. **Dokładność metod**:
   - Metoda hit-and-miss:
     - Jest mało dokładna i wolno zbieżna, szczególnie dla funkcji z asymptotą, jak $\frac{1}{\sqrt{x}}$.
     - Dla funkcji $f(x) = x^2 + x + 1$ oraz $g(x) = \sqrt{1 - x^2}$ metoda daje sensowne wyniki, ale wymaga bardzo dużej liczby prób n, aby osiągnąć niski błąd.
     - Błąd nie zawsze maleje z liczbą prób, może wykazywać oscylacje jak w przypadku funkcji $\frac{1}{\sqrt{x}}$.
   - Metoda prostokątów: - Jest znacznie bardziej dokładna niż metoda hit-and-miss.
     - Dla pierwszych dwóch funkcji metoda prostokątów osiąga niski błąd nawet przy relatywnie małej liczbie podziałów.
     - Dla funkcji $\frac{1}{\sqrt{x}}$ metoda ma już dużo większe błędy niż w przypadku dwóch pozostałych funkcji, lecz błąd wraz z ilością podziałów maleje, nie wykazuje oscylacji jak w przypadku metody hit-and-miss.
2. **Czas obliczeń**: Metoda prostokątów jest szybsza niż metoda hit-and-miss.
3. **Wybór metody**:
   - Metoda hit-and-miss może być użyteczna w sytuacjach, gdzie inne metody numeryczne są trudne do zastosowania lub gdy funkcja jest trudna do zintegrowania analitycznie.
   - Metoda prostokątów jest preferowana ze względu na swoją dokładność i efektywność czasową, szczególnie w przypadkach, gdy funkcja jest dobrze zdefiniowana i nie posiada asymptot.

Bibliografia

- Marian Bubak, Katarzyna Rycerz: *Metody Obliczeniowe w Nauce i Technice. Obliczanie całek metodami Monte Carlo*