Metody Numeryczne
Instrukcja do laboratorium 3

Andrii Shekhovtsov

9 grudnia 2022

1 Zasady ogólne

Do wykonania i przesłania jest 6 zadań. Ocena za laboratorium zależy od liczby dobrze zrobionych zadań, skala ocen jest pokazana w Tabeli 1.

Tabela 1: Skala ocen.

Liczba zrobionych zadań domowych	Ocena
1-2	3.0
3	3.5
4	4.0
5	4.5
6	5.0

W przypadku nie oddania zadania w terminie ustawionym w systemie Moodle, uzyskana ocena jest zmniejszana o 0,5 za każdy zaczęty tydzień opóźnienia. Jako termin oddania liczy się data wgrania pliku z kodem źródłowym na platformie Moodle.

Kod wszystkich wykonanych programów proszę wkleić do jednego pliku, podpisując poszczególne zadania za pomocą komentarzy. Zadanie proszę wklejać po kolei (1, 2, 3...). Plik z zadaniami proszę przesłać na platformie Moodle. Plik musi mieć nazwę `numeralbumu_lab3.py`. **Plik musi być plikiem tekstowym z rozszerzeniem `.py`**. Wysłanie pracy w nieodpowiedniej formie może skutkować wystawieniem oceny niedostatecznej za te laboratorium.

Listing 1: **Przykład dobrze sformatowanego pliku z zadaniami.**

```python
# Zadanie 1
a = 4
b = 2
print(a + b)

# Zadanie 2
# Brak

# Zadanie 3
a = 3
b = 2
print(a ** b)

# ...
```

2 Zakres tematyczny

- Obliczenie pierwiastka (miejsca zerowego) funkcji za pomocą metody połowienia.
- Obliczenie pierwiastka (miejsca zerowego) funkcji za pomocą metody Newtona.
- Obliczenie lokalnego minimum funkcji za pomocą metody złotego podziału.

3 Zadania do wykonania

Zadanie 1

Napisać funkcje implementującą metodę prostokątów służącą do całkowania numerycznego.

Przy implementacji proszę założyć że liczymy tylko pola funkcji znajdujących się w I ćwiartce układu kartezjańskiego (mamy tylko dodatnie wartości dla X i Y).

Przykładowe argumenty: funkcja przyjmuje funkcje f pole pod którą ma być obliczone, przedział całkowania $[a, b]$, oraz liczbę prostokątów (ewentualnie krok, zależy od sposobu implementacji). Liczba prostokątów (krok) wypływa na dokładność rozwiązania. Funkcja ma zwrócić obliczone pole (jedna wartość).

Przykład:

Obliczymy pole pod funkcją $f(x) = x^3$ w przedziale $[0, 2]$ za pomocą 20 prostokątów (krok około 0.1).

Listing 2: Przykładowe wywołanie funkcji.

```
def f(x):
    return x ** 3

# Jako argumenty podajemy zakres calkowania oraz liczbe prostokatow.
print(rectangle_method(f, 0, 2, 20))
# 3.995
```

Zrobimy to też analitycznie:

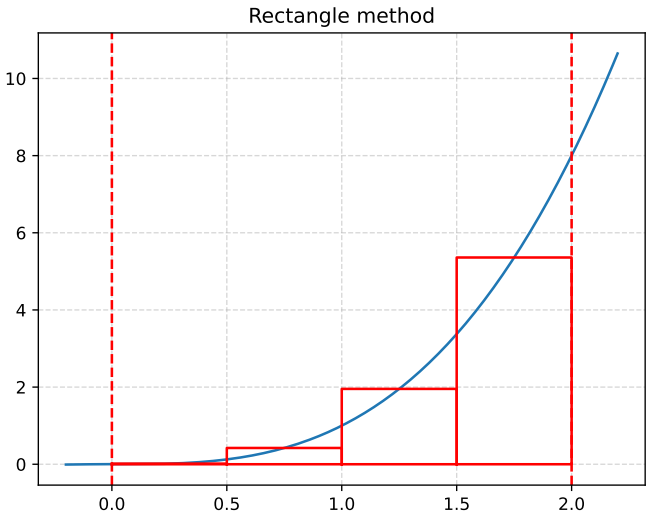
$$P_{f(x)} = \int_0^2 x^3 dx = \frac{x^4}{4}\Big|_0^2 = \frac{2^4}{4} - \frac{0^4}{4} = \frac{16}{4} = 4$$

Jak widzimy, przy obliczaniu tej całki metodą prostokątów dostaliśmy rozwiązanie przybliżone. Przy zwiększeniu dokładności uzyskalibyśmy bardziej dokładne rozwiązanie.

Zadanie 2

Rozbudować funkcjonalność funkcji z **Zadania 1** tak żeby była pokazywana wizualizacja działania zaimplementowanej metody. Proszę zwrócić uwagę, że do wizualizacji używamy mniejszej liczby prostokątów (większy krok).

Przykład takiej wizualizacji jest pokazany na Rysunku poniżej.



Rysunek 1: Przykładowa wizualizacja metody prostokątów.

Zadanie 3

Napisać funkcje implementującą metodę trapezów służącą do całkowania numerycznego.

Przy implementacji proszę założyć że liczymy tylko pola funkcji znajdujących się w I ćwiartce układu kartezjańskiego (mamy tylko dodatnie wartości dla X i Y).

Przykład:

Obliczymy pole pod funkcją $f(x) = x^3$ w przedziale $[0, 2]$ używając 20 trapezów (krok około 0.1).

Listing 3: Przykładowe wywołanie funkcji.

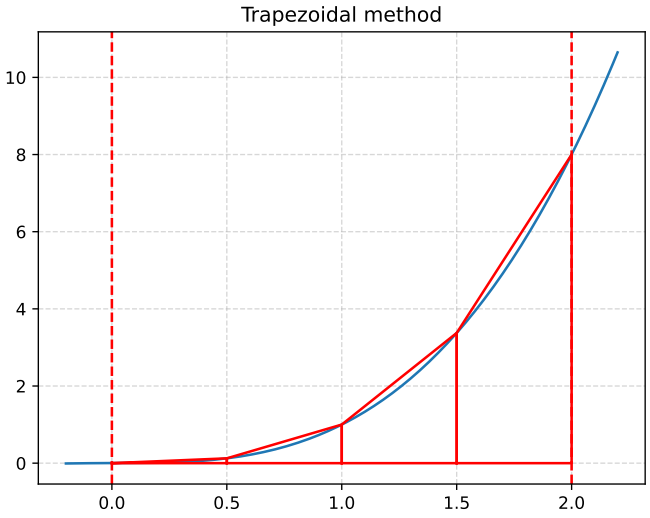
```
# Funkcja f jest zdefiniowana poprzednio
print(trapezoidal_method(f, 0, 2, 20))
# 4.01
```

Jak widzimy, dostaliśmy wynik bliski temu wyniku który uzyskaliśmy całkując tą funkcję analitycznie, ale w dalszym ciągu mamy małe odchylenie.

Zadanie 4

Rozbudować funkcjonalność funkcji z **Zadania 3** tak żeby była pokazywana wizualizacja działania zaimplementowanej metody.

Przykład takiej wizualizacji jest pokazany na Rysunku poniżej.



Rysunek 2: Przykładowa wizualizacja metody trapezów.

Zadanie 5

Napisać funkcje implementującą metodę Monte Carlo służącą do całkowania numerycznego.

Przy implementacji proszę założyć że liczymy tylko pola funkcji znajdujących się w I ćwiartce układu kartezjańskiego (mamy tylko dodatnie wartości dla X i Y).

Przykład:

Obliczymy pole pod funkcją $f(x) = x^3$ w przedziale $[0, 2]$, używając do tego 10000 wylosowanych punktów:

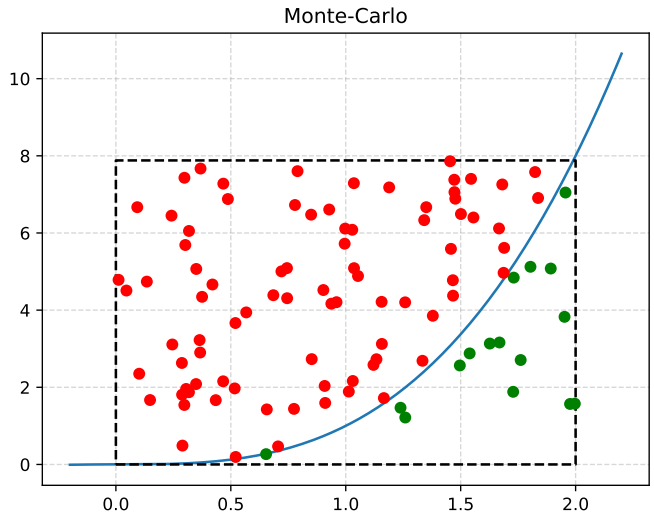
Listing 4: Przykładowe wywołanie funkcji.

```
# Jako argumenty podajemy funkcje, zakres, oraz liczbe strzałów
print(monte_carlo(f, 0, 2, 10000))
# 4.078998042399999
```

Zadanie 6

Rozbudować funkcjonalność funkcji z **Zadania 5** tak żeby była pokazywana wizualizacja działania zaimplementowanej metody.

Przykład takiej wizualizacji jest pokazany na Rysunku poniżej.



Rysunek 3: Przykładowa wizualizacja metody Monte-Carlo.

4 Informacje pomocnicze

Wzór na pole trapezu:

$$P_{trap} = \frac{a \cdot b}{2} h,$$

gdzie a i b to są podstawy trapezu, a h to jego wysokość.

Narysować kreskę pionową można za pomocą polecenia: `plt.axvline()` przykładowo `plt.axvline(x=0)` rysuje kreskę pionową w punkcie $x = 0$.

Przy wykonaniu wizualizacji dla metody prostokątów i metody trapezów można użyć zwykły `plt.plot()`, w sposób pokazany niżej (jako argumenty podajemy dwie listy współrzędnych x i y które definiują nasz prostokąt lub trapez, pierwszy i ostatni punkt muszą być jednakowe by linia była zamknięta):

Listing 5: Przykład rysowania kształtu za pomocą funkcji plot.

```
plt.plot([0, 1, 1, 0, 0], [0, 0, 0.5, 1, 0],
         marker='o', markersize=10)
plt.grid(linestyle='--', alpha=0.5)
plt.show()
```

Proszę zwrócić uwagę, że w metodzie Monte Carlo konieczne jest także górne ograniczenie dla strzałów (losowanych punktów). Może ono być ustawione arbitralnie, lecz lepszym rozwiązaniem będzie użycie np. metody złotego podziału w celu wyszukania wartości ekstremum w zadanym przedziale. Żeby wyszukać wartości maksymalnej wystarczy odwrócić funkcję: czyli żeby wyszukać lokalne maksimum x_{max} dla funkcji $f(x)$ wystarczy że odszukamy lokalne minimum funkcji $g(x) = -f(x)$. Wartość $f(x_{max})$ znaleziona w ten sposób będzie właśnie górnym ograniczeniem dla losowanych punktów.