Metody Probabilistyczne i Statystyka

Zadanie domowe 1

Rafał Włodarczyk 2024-11-05

Spis treści

1 Implementacja Symulacji **1**
1.1 Generator liczb pseudolosowych 1

2 Obliczenia oraz wykresy **2**
2.1 Obliczenia oraz wykresy dla funkcji $f_1(x) = \sqrt[3]{x}$ 2
2.2 Obliczenia oraz wykresy dla funkcji $f_2(x) = \sin(x)$ 2
2.3 Obliczenia oraz wykresy dla funkcji $f_3(x) = 4x(1-x)^3$ 3

3 Wyznaczenie wartości liczby π **3**

4 Wnioski **4**

1 Implementacja Symulacji

Kod symulacji dostępny jest w pliku `main.py`, dołączonym do tego dokumentu.
Zadaną symulację zaimplementowałem w języku `python`. Do wykonania obliczeń oraz wyświetlenia wyników wykorzystałem biblioteki `numpy`[1] oraz `matplotlib`[2]. Moje środowisko w systemie Linux można odtworzyć przy użyciu następujących poleceń (zakładam zainstalowanego Pythona w wersji 3.12):

```
$ python -m venv venv # tworzę virtual environment
$ source venv/bin/activate # aktywuję virtual environment
(venv) $ pip install numpy matplotlib # instaluję biblioteki
```

1.1 Generator liczb pseudolosowych

Zgodnie z zaleceniami wykorzystałem Mersenne Twister PRNG[3] z biblioteki `numpy`. Funkcja generująca pseudolosowe punkty (fragment klasy Simulation) wygląda wobec tego następująco:

```
    def generate_points(self, n: int) -> List[Point]:
        """
        :param n: number of points to be generated
        :return: a list of n points, with x in the range and y in [0, M]
        """
        points = []
        rng = np.random.Generator(np.random.MT19937())
        for _ in range(n):
            x = rng.uniform(self.interval[0], self.interval[1])
            y = rng.uniform(0, self.M)
            points.append(Point(x, y))
        return points
```

[1] `https://numpy.org/doc/stable/`
[2] `https://matplotlib.org/stable/contents.html`
[3] `https://numpy.org/doc/stable/reference/random/bit_generators/mt19937.html`

2 Obliczenia oraz wykresy

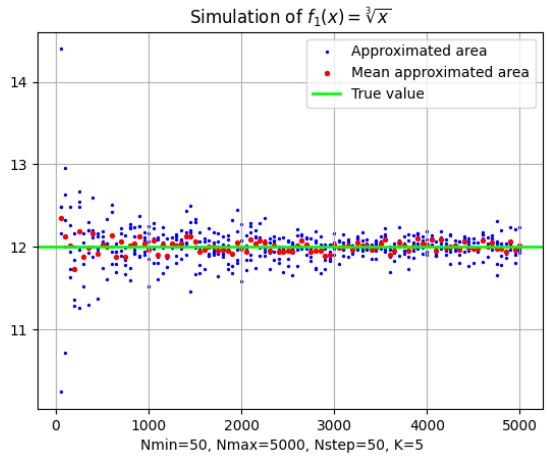
Dla każdej z zadanych funkcji wykonałem symulację z parametrem $k = 5$ oraz $k = 50$. Wygenerowałem wykresy, na których zaznaczyłem wartość średnią danej próbki oraz wartość faktyczną, zgodnie z wyliczeniami zamieszczonymi przy odpowiednich wykresach.

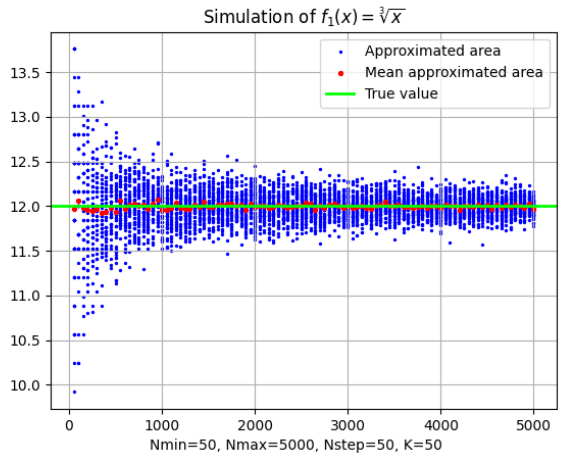
2.1 Obliczenia oraz wykresy dla funkcji $f_1(x) = \sqrt[3]{x}$

Policzmy zadaną całkę:

$$\int_0^8 \sqrt[3]{x}\,dx = \left[\frac{3}{4}x^{\frac{4}{3}}\right]_0^8 = \frac{3}{4}\cdot\left(8^{\frac{4}{3}} - 0^{\frac{4}{3}}\right) = 12 \tag{1}$$

Wykresy wygenerowane dla funkcji f_1 w `main.py`:



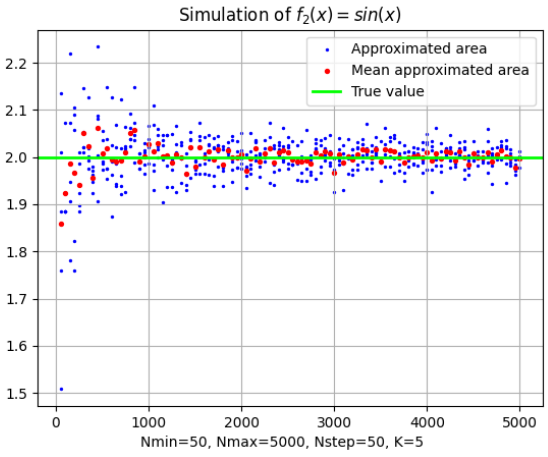


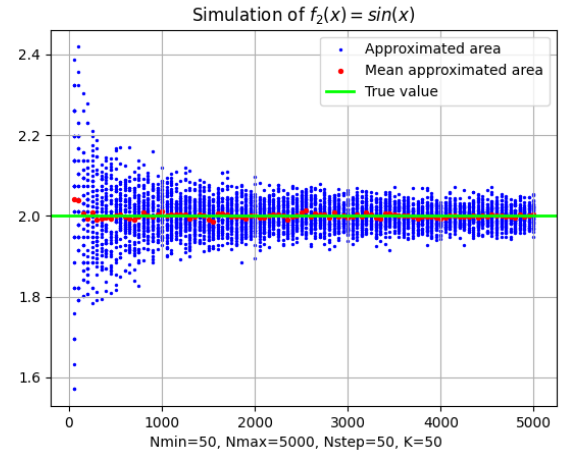
2.2 Obliczenia oraz wykresy dla funkcji $f_2(x) = \sin(x)$

Policzmy zadaną całkę:

$$\int_0^\pi \sin(x)dx = [-\cos(x)]_0^\pi = -(\cos(\pi) - \cos(0)) = -(-1 - 1) = 2 \tag{2}$$

Wykresy wygenerowane dla funkcji f_2 w `main.py`:





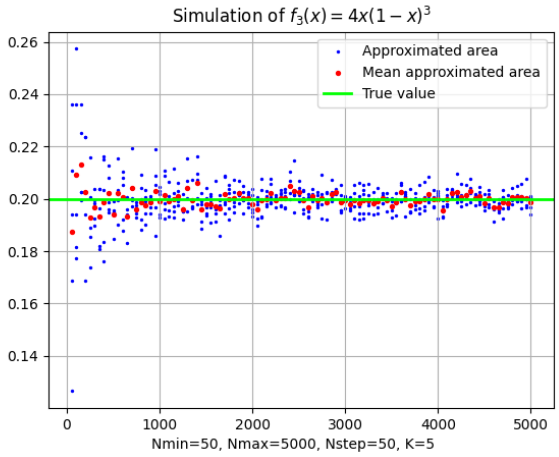
2.3 Obliczenia oraz wykresy dla funkcji $f_3(x) = 4x(1-x)^3$

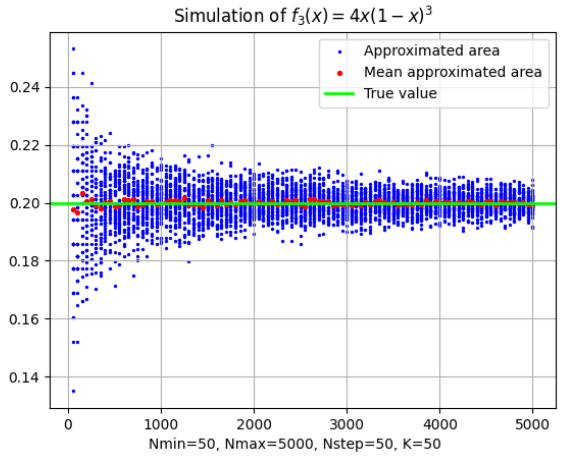
$$\int_0^1 4x(1-x)^3 dx = 4 \cdot \int_0^1 x - 3x^2 + 3x^3 - x^4 dx = \tag{3}$$

$$= 4 \cdot \left(\int_0^1 x dx - \int_0^1 3x^2 dx + \int_0^1 3x^3 dx - \int_0^1 x^4 dx \right) = \tag{4}$$

$$= 4 \cdot \left(\frac{1}{2} - 1 + \frac{3}{4} - \frac{1}{5} \right) = \frac{1}{5} \tag{5}$$

Wykresy wygenerowane dla funkcji f_3 w `main.py`:





3 Wyznaczenie wartości liczby π

Do wyznaczenia wartości liczby π posłużyłem się funkcją $f_{cw}(x) = \sqrt{1-x^2}$, która na przedziale $[0,1]$ opisuje ćwiartkę okręgu jednostkowego. Zauważmy, że:

$$\int_0^1 4 \cdot f_{cw}(x)\, dx = 4 \cdot \int_0^1 \sqrt{1-x^2}\, dx = \tag{6}$$

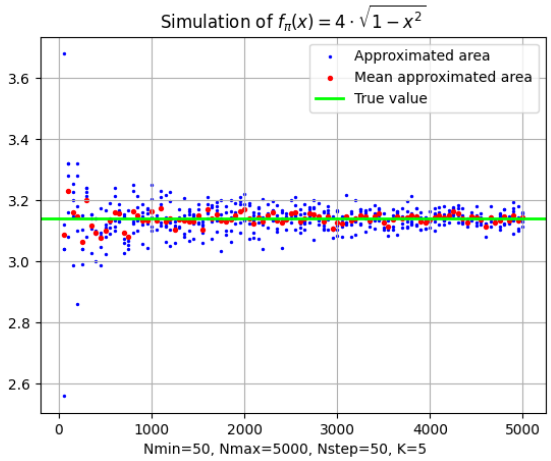
$$= 4 \cdot \int_0^{\frac{\pi}{2}} \sqrt{1-\sin^2(x)} \cdot \cos(x) dx = \tag{7}$$

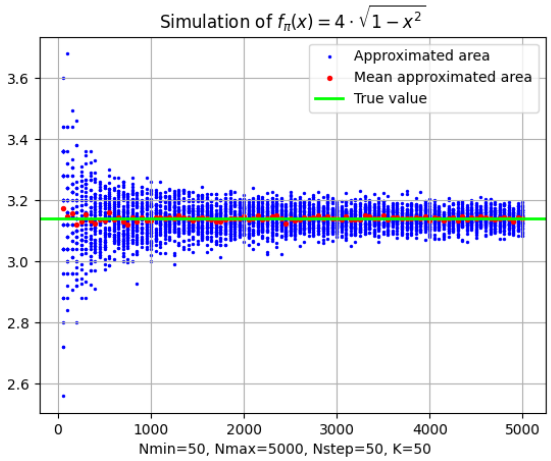
$$= 4 \cdot \int_0^{\frac{\pi}{2}} \cos^2(x) dx = \tag{8}$$

$$= 4 \cdot \frac{1}{2} \cdot \left[x + \frac{\sin(2x)}{2} \right]_0^{\frac{\pi}{2}} = \tag{9}$$

$$= \pi \tag{10}$$

Wykresy wygenerowane dla funkcji f_π w `main.py`:





4 Wnioski

Metoda Monte Carlo sprawda się w przybliżaniu wartości całek zadanych funkcji. Wzrost liczby próbek n w pojedyńczej symulacji umożliwia uzyskanie dokładniejszej esytmacji prawdziwej wartości.

Średnie wyniki uzyskane z $k = 50$ powtórzeń są bliższe prawdziwym wartościom, niż wyniki uzyskiwane z $k = 5$ powtórzeń - zwięszenie liczby niezależnych symulacji pozwala zmniejszyć wpływ skrajnych wyników.

Zastosowanie dobrego generatora liczb pseudolosowych pozwala poprawić dokładność wyniku.

Metoda Monte Carlo nie jest w stanie wyznaczyć dokładnego wyniku, a zebranie satysfakcjonującej ilości próbek jest zadaniem wymagającym sporej ilości zasobów obliczeniowych.