Metody Monte Carlo

Laboratorium 7

Zadanie 1 i 2

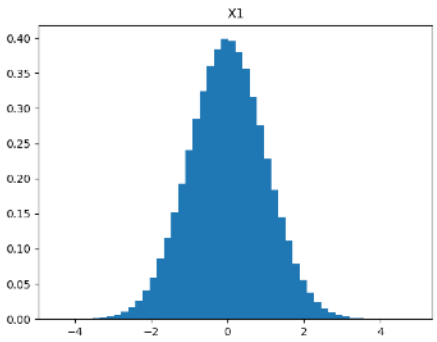
Kod (Python) współdzielony

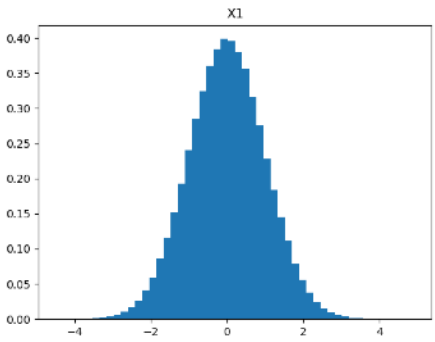
```python
import numpy as np
from matplotlib import pyplot as plt


def generate_normal_distribution(lower, upper):
  return np.random.normal(lower, upper)


def generate_uniform_distribution(lower, upper):
  return np.random.uniform(lower, upper)


def odchylenie_standardowe(M, estymator_y, results):
  return np.sqrt(1 / (M - 1) * np.sum([(y - estymator_y)**2 for y in results]))


def estymator(M, results):
  return 1 / M * np.sum(results)


def przedzial_rozszerzenia(results, p, M):
  q = int(p * M)
  output_sorted = results.copy()
  output_sorted.sort()
  r = int((M - q) / 2)
  if not isinstance(r, int):
    r = int((M - q + 1) / 2)
  y_min = output_sorted[r]
  y_max = output_sorted[r + q]

  return y_min, y_max


def _gum_odchylenie_czastkowe(x, mean_x):
  return np.sqrt(1 / len(x) * np.sum([(x - mean_x)**2 for x in x]))


def gum_odchylenie_standardowe(X):
  x1, x2 = zip(*X)
  mean_x1 = np.mean(x1)
  mean_x2 = np.mean(x2)
  std_dev_x1 = _gum_odchylenie_czastkowe(x1, mean_x1)
  std_dev_x2 = _gum_odchylenie_czastkowe(x2, mean_x2)
  std_dev = np.sqrt(std_dev_x1**2 + std_dev_x2**2)

  return std_dev


def gum_przedzial_rozszerzenia(results, std_dev):
  y_mean = np.mean(results)
  y_min = y_mean - 2 * std_dev
  y_max = y_mean + 2 * std_dev

  return y_min, y_max


def calculate_results(PAIRS, SUMS, M, p):
  estymator_y = estymator(M, SUMS)
  std_dev = odchylenie_standardowe(M, estymator_y, SUMS)
  y_min, y_max = przedzial_rozszerzenia(SUMS, p, M)
  gum_std_dev = gum_odchylenie_standardowe(PAIRS)
  gum_y_min, gum_y_max = gum_przedzial_rozszerzenia(SUMS, gum_std_dev)

  print('Estymator:', estymator_y)
```

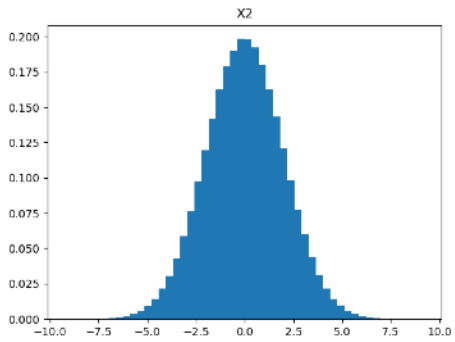
```
  print('Odchylenie standardowe:', std_dev)
  print('Przedział rozszerzenia: [y_min:', y_min, 'y_max:', y_max, ']')
  print('Odchylenie standardowe GUM:', gum_std_dev)
  print('Przedział rozszerzenia GUM: [y_min:', gum_y_min, 'y_max:', gum_y_max,
        ']')


def plot(data, title, filename):
  plt.figure()
  plt.hist(data, bins=50, density=True)
  plt.title(title)
  plt.savefig(filename)


def main():
  # 1. Okreslenie liczby prob monte carlo
  M = 10**6
  p = 0.9545

  print("--- ZADNAIE 1 ---")

  # 2. Wygenerowanie M wektorow danych wejsciowych X
  PAIRS = []
  SUMS = []
  for _ in range(M):
    x1 = generate_normal_distribution(0, 1)
    x2 = generate_normal_distribution(0, 2)
    PAIRS.append((x1, x2))
    SUMS.append(x1 + x2)

  calculate_results(PAIRS, SUMS, M, p)

  plot([a for a, b in PAIRS], "X1", "x1_zad1.png")
  plot([b for a, b in PAIRS], "X2", "x2_zad1.png")
  plot([a + b for a, b in PAIRS], "Y", "y_zad1.png")

  print("--- ZADANIE 2 ---")

  # 2. Wygenerowanie M wektorow danych wejsciowych X
  PAIRS = []
  SUMS = []
  for _ in range(M):
    x1 = generate_uniform_distribution(0, 4)
    x2 = generate_uniform_distribution(5, 6)
    PAIRS.append((x1, x2))
    SUMS.append(x1 + x2)

  calculate_results(PAIRS, SUMS, M, p)

  plot([a for a, b in PAIRS], "X1", "x1_zad2.png")
  plot([b for a, b in PAIRS], "X2", "x2_zad2.png")
  plot([a + b for a, b in PAIRS], "Y", "y_zad2.png")


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

Zadanie 1 - Wyniki

--- ZADNAIE 1 ---
Estymator: -0.0035459335726771546
Odchylenie standardowe: 2.2347857268690956
Przedział rozszerzenia: [y_min: -4.472455702300937 y_max: 4.464023240725864]
Odchylenie standardowe GUM: 2.2349096190841076
Przedział rozszerzenia GUM: [y_min: -4.473365171740892 y_max: 4.466273304595538]



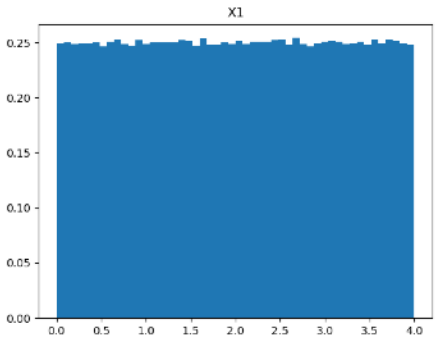


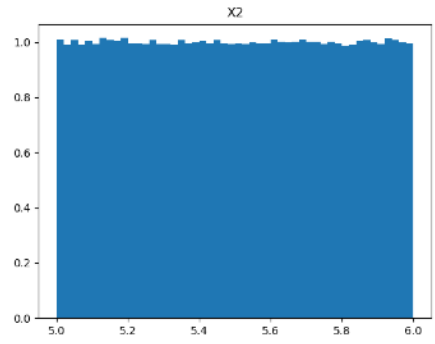


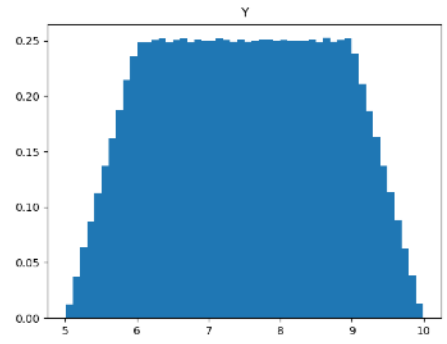
Wartości obliczone z metody Monte Carlo i GUM, przy parametrach M = 10^6 i p = 9545 są do siebie zbliżone, gdy zmienne losowe zostały wygenerowane z rozkładu normalnego. Minimalne odchylenia w przedziałach rozszerzeń mogą wynikach z pseudolosowego charakteru otrzymanych zmiennych losowych, co jest raczej oczekiwanym skutkiem.

Zadanie 2 - Wyniki

--- ZADANIE 2 ---
Estymator: 7.5008020001049776
Odchylenie: 1.1897098976013072
Przedział rozszerzenia: [y_min: 5.4282782972715955 y_max: 9.57423634310727]
Odchylenie standardowe GUM: 1.1899168873558477
Przedział rozszerzenia GUM: [y_min: 5.120968225393282 y_max: 9.880635774816673]







Tym razem, w odróżnieniu do zadanie nr 1, zmienne losowe zostały wyznaczone z pseudolosowego rozkładu równomiernego. Co ciekawe, przedziały rozszerzenia nie są już do siebie tak zbliżone, jak w zadaniu 1.
Wynika to prawdopodobnie z liczby **p**, która w zadaniu nr 1 wskazywała zakres procentowy dolnego i górnego zakresu w sposób najbardziej prawidłowy. Rozkład prostokątny (równomierny) wprowadza większy błąd do naszych wyliczeń, i to zostaje odzwierciedlone w propagacji zaburzeń GUM.s