# Звіт до лабораторної 1

Студента групи ТТП-42

Остренка Олександра

### Підкидання монети.

#### Умова

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

#### Реалізація

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

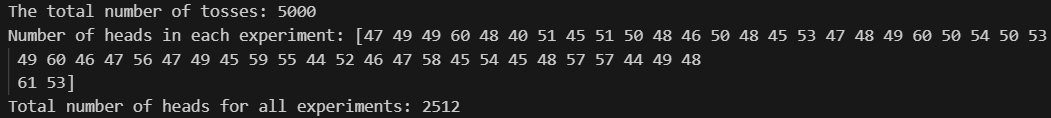
Автоматически созданное описание

За допомогою бібліотеки numpy можна з легкістю змоделювати підкидання монети, бо функція randint генерує цілу випадкову величину з рівномірного розподілу для проміжку [0;2). Змінна results являє собою матрицю K\*N, де один рядок – результати підкидань одного експеременту.

#### Тестування

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание



Изображение выглядит как снимок экрана, текст, шаблон, черно-белый

Автоматически созданное описание

Т.я. використовується функція, що моделює випадкові величини з рівномірного розподілу, то, в теорії, кількості появи герба та решки мають прямувати до одного і того самого числа. Можна спостерігати, що число появи герба і справді прямує до K\*N/2 при збільшені кількості підкидань, але на малих вибірках ми не отримаємо рівну кількість того і того через елемент випадковості при моделюванні.

### 1.2. Підкидання кубику.

#### Умова

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

#### Реалізація

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

За допомогою бібліотеки numpy можна з легкістю змоделювати підкидання кубику, бо функція randint генерує цілу випадкову величину з рівномірного розподілу для проміжку [1;7). Змінна results являє собою матрицю K\*N, де один рядок – результати підкидань одного експеременту.

#### Тестування

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Як і з підкиданням монети, в середньому, кожна цифра має випадати однакову кількість разів. В даному output кількість випадінь має прямувати до 18, але через випадкову складову моделювання кількості різняться.

### 2.2. Підкидання двох кубиків.

#### Умова

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

#### Реалізація

Для моделювання суми очок при підкиданні двох кубиків достатньо по прикладу задачі 1.2 змоделювати два рази один кубик і повернути суму очок.

Изображение выглядит как текст, Шрифт, снимок экрана

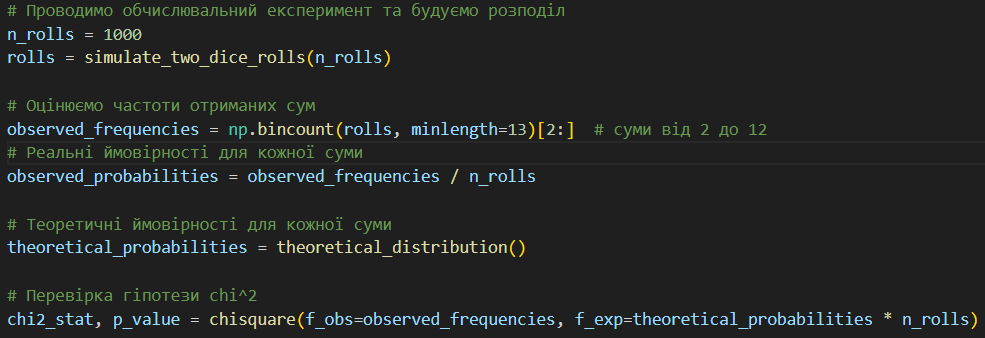
Автоматически созданное описание

Теоретичний розподіл знаходиться простим перерахуванням кількості комбінацій очок для конкретної суми.

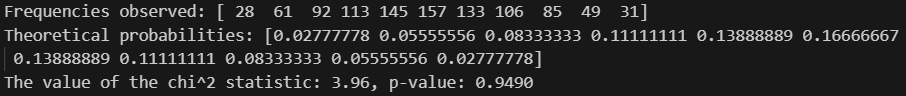
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Далі проводимо обчислюваний експеримент, де отримуємо частоти випадання різних сум та обчислюємо на основі цих частот відповідні ймовірності. Отримуємо теоретичні ймовірності до кожної суми. За допомогою функції «chisquare» з бібліотеки «scipy.stats» обчислюємо значення статистики chi^2, а також відповідне p-value для перевірки відповідності отриманих (експериментально) частот з теоретичним розподілом. В даному випадку теоретичний розподіл задається порахованими частотами випадінь відповідних сум.



#### Тестування



Таке високе p-value (явно більше за прийняте 0.05) говорить про те, що немає підстав відкидати гіпотезу у тому, що експериментальні дані слідують теоретичному розподілу.

Для наочності я також відобразив теоретичні та емпіричні дані у вигляді стовпчастої діаграми.

Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание

З неї добре видно, що експериментальні ймовірності ідуть бік о бік з теоретичними, а сума в 7 очок буде випадати в більшій кількості підкидань.

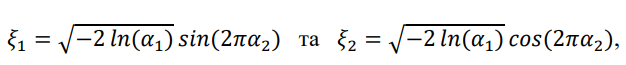
### 3.4. Моделювання випадкових величин з нормального розподілу.

#### Умова

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

#### Реалізація

Для функціонального моделювання було використано метод оберненої функції, а саме - перетворення Бокса-Мюллера, в якому пара випадкових величин генеруються за наступною формулою:  


Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Далі моделюємо випадкові величини з використанням бібліотечної функції random.gauss().

Изображение выглядит как текст, Шрифт, снимок экрана, линия

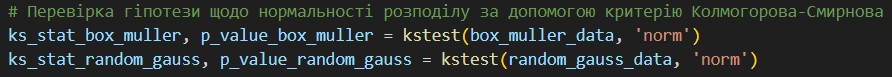
Автоматически созданное описание

Після отримання 1000 випадкових величин обома підходами, оцінюємо математичне сподівання та дисперсію отриманих вибірок.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

За допомогою критерію Колмогорова-Смирнова перевіряємо гіпотезу про нормальність отриманих вибірок.



#### Тестування

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Можна побачити, що обидва методи дають дуже хороші результати – мат. сподівання прямує до нуля, дисперсія до одиниці, а p-value набагато перевищує 0.05, що означає відсутність підстав відкидати гіпотезу про нормальність обох вибірок.

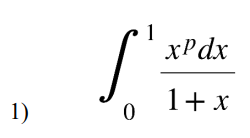
Візуалізована гістограма наглядно показує, що отримані вибірки дотримуються нормального розподілу

Изображение выглядит как диаграмма, снимок экрана, График, текст

Автоматически созданное описание

### 4.1.1. Обчислити інтеграл методом Монте-Карло

#### Умова



#### Реалізація

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Даний метод генерує 10млн точок та рахує співвідношення тих, що знаходяться під заданою функцією, до загальної кількості точок. Т.я. точки генеруються в квадрати 1\*1, то додаткових масштабувань робити нема потреби.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Щоб оцінити успішність такого інтегрування, я використав бібліотеку scipy.integrate для отримання «справжнього» значення шуканого інтегралу.

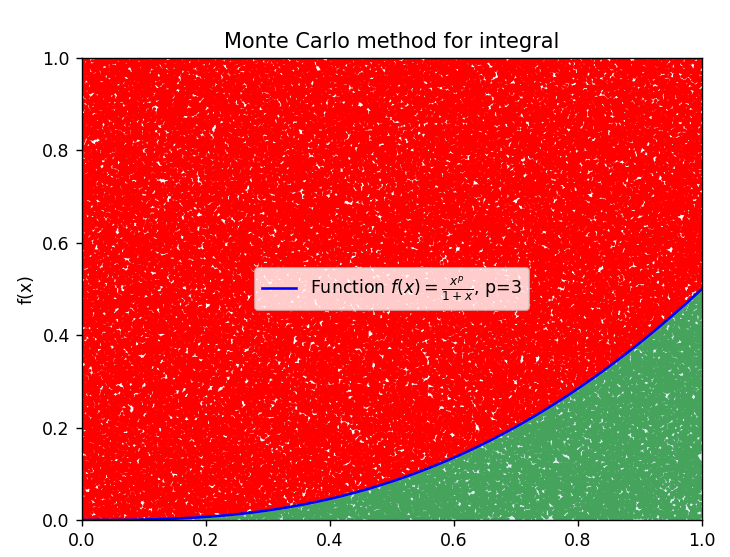
#### Тестування



Для 10млн точок бачимо правильність з точністю до третього знаку після коми.



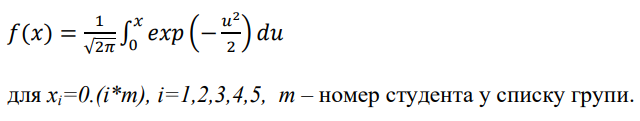
Для 100млн точок – до четвертого, але при цьому програма помітно зависає секунд на 10.



Також я візуалізував сам розподіл точок.

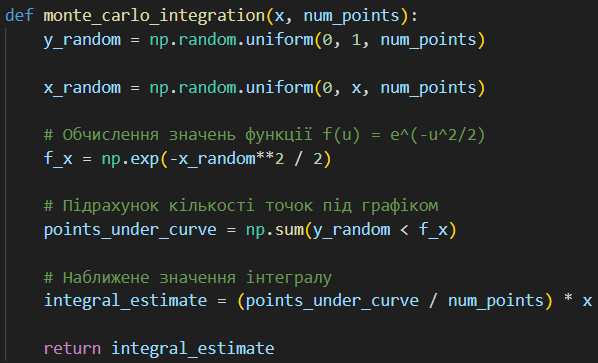
### 4.2. Обчислити значення функції

#### Умова



m=20.

#### Реалізація



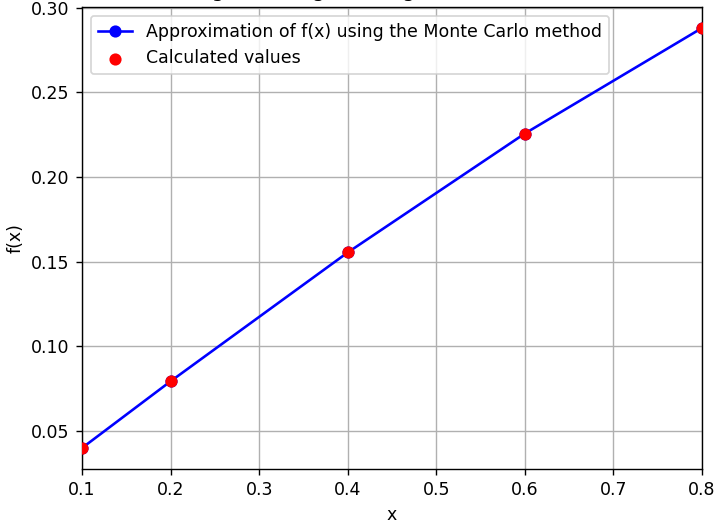
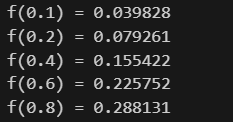
Ідея реалізації як і в завданні 4.1.1, тільки в даному випадку співвідношення кількості точок треба масштабувати до площі прямокутника, тому і домножуємо на його ширину – х.

Изображение выглядит как текст, снимок экрана, Шрифт

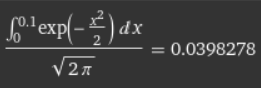
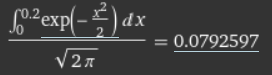
Автоматически созданное описание

Далі для кожного х рахуємо значення f(x).

#### Тестування

Щоб перевірити правдивість розрахунків, можна використати електронних ресурс WolframAlpha:

  Изображение выглядит как рукописный текст, Шрифт, текст, грифельная доска

Автоматически созданное описание Изображение выглядит как Шрифт, рукописный текст, текст, линия

Автоматически созданное описание

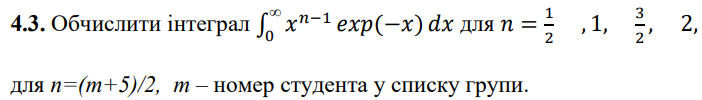
Изображение выглядит как Шрифт, текст, рукописный текст, линия

Автоматически созданное описание

Бачимо, що розрахунки співпадають з точністю до четвертого знаки після коми, враховуючи, що для моделювання було використано 10млн точок.

### 4.3. Обчислити інтеграл.

#### Умова



m=20 => n=17.5

#### Реалізація

Один з підінтегральних множників – щільність ймовірності експоненційного розподілу з lambda=1: exp(-x). Відповідно згідно з методом Монте-Карло, можемо розглядати даний інтеграл як мат. сподівання функції f(x)=x^(n-1) для випадкової величини Х, що розподілена за експоненційним законом.

Изображение выглядит как Шрифт, линия, текст, диаграмма

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Таким чином головна функція генерує 10млн точок з експоненційного розподілу, підставляє них в досліджувану функцію та рахує мат. сподівання.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Для перевірки успішності такого моделювання було використано вбудовану функцію з бібліотеки scipy.integrate для підрахунку інтегралу.

#### Тестування

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Окрім сухих чисел була також побудована гістограма, на якій зіставляються результати моделювання методом Монте-Карло та вбудованою функцією. Як можна спостерігати, значення збігаються з доволі хорошою точністю, але треба розуміти, що при більших X точність буде дуже сильно падати.

Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

### 5.5. Знайти екстремум (мінімум, максимум) функції методом випадкового пошуку

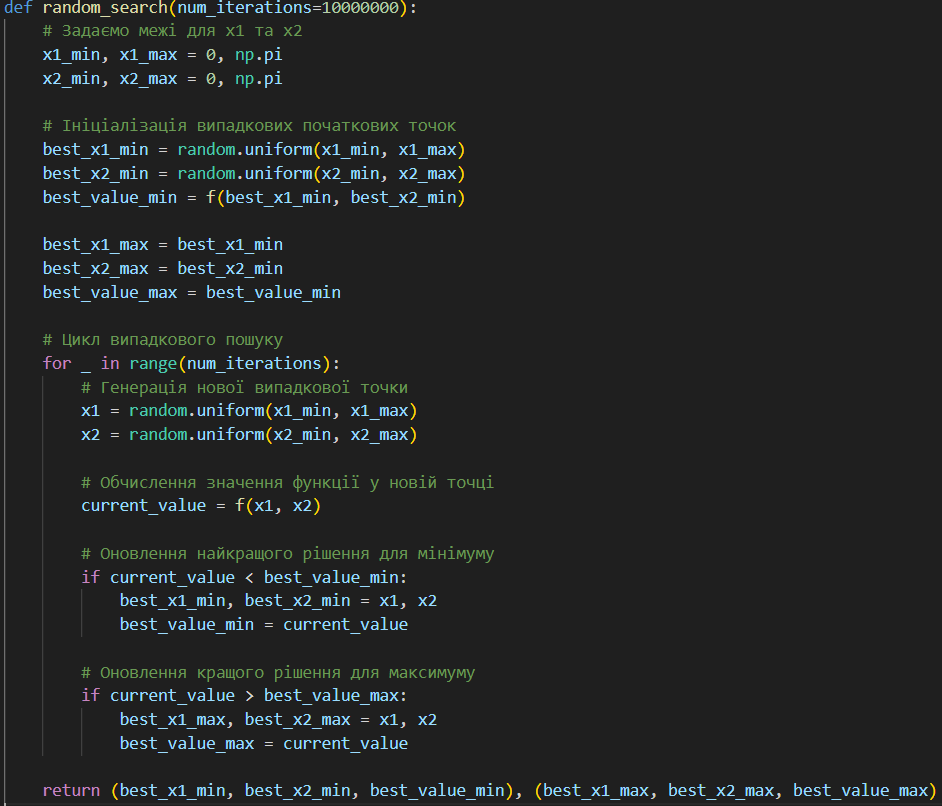
#### Умова

Изображение выглядит как текст, диаграмма, снимок экрана, График

Автоматически созданное описание

#### Реалізація

Сам алгоритм достатньо простий з точки зору реалізації – просто перебирається безліч точок з відповідного проміжку і оновлюються локальні мінімуми та максимуми.



#### Тестування

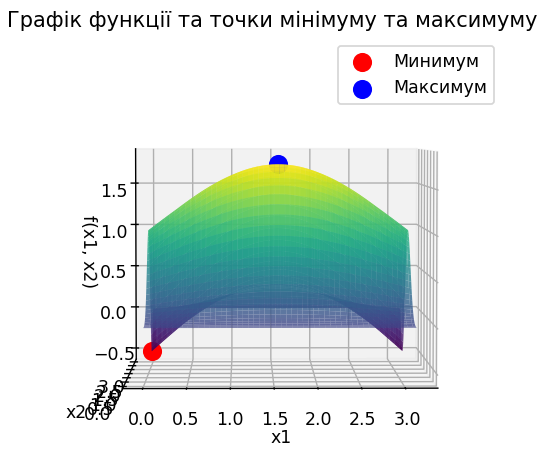


Изображение выглядит как текст, снимок экрана, Красочность, диаграмма

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, диаграмма, График

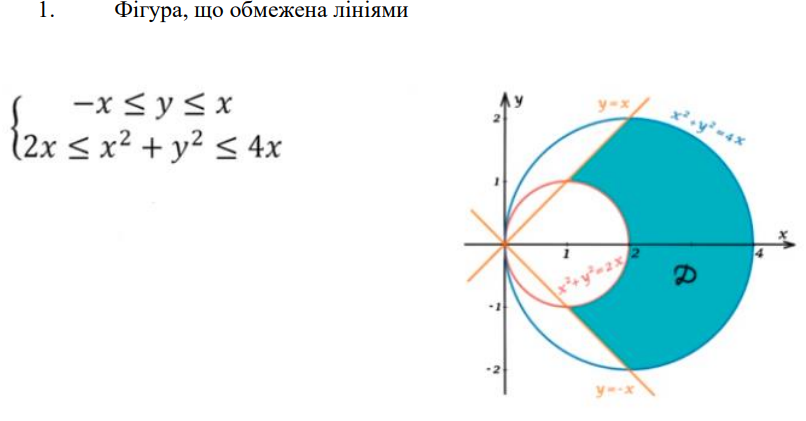
Автоматически созданное описание



Якщо вірити графічному відображенню даної функції і двох знайдених точок, то вони і справді знаходяться в околі екстремумів.

### 6.1.1. Обчислити площу

#### Умова



#### Реалізація

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

Функція для визначення «чи попала точка в фігуру». По суті перевіряє дві умови системи.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Дана функція генерує точки з відповідного квадрату, який обмежує задану фігуру, після чого находить співвідношення точок «всередині» фігури до загальної кількості точок. Даним співвідношенням і оцінюється площа фігури.

#### Тестування



Изображение выглядит как текст, снимок экрана, Красочность, Графика

Автоматически созданное описание

Після виконання можна побачити наступну оцінку площі та графічний вигляд фігури, отриманої по згенерованим точкам.

### 6.2.1. Обчислити об’єм перетину двох циліндрів

#### Умова

Изображение выглядит как аксессуар

Автоматически созданное описание

#### Реалізація

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Для початку перевіряємо чи належить точка (x, y, z) першому та другому циліндрам з радіусом r.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Далі генеруємо точки в кубі 2r\*2r і визначаємо, чи згенерована точка належить перетину циліндрів. Відповідним співвідношенням «в перетині циліндрів»/«не в перетині циліндрів» ми і находимо оцінку об’єму даної фігури.

Изображение выглядит как Шрифт, зарисовка, диаграмма, белый

Автоматически созданное описание

Для перевірки успішності даного моделювання було пораховано оцінка стандартного відхилення для випадкової величини «1, якщо точка потрапляє у фігуру, і 0 – якщо не потрапляє» і таким чином оцінити наскільки реальний об’єм може відрізнятись від змодельованого на 10млн точках.

#### Тестування.



Виходячи з розрахунків, отриманий розв’язок правильний з точністю до третього знаку після коми.