НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра технічної кібернетики

Звіти до комп’ютерних практикумів з кредитного модуля “Моделювання Систем”

Виконав

Студенти групи ІТ-02 Терешкович М.О.

Перевірила:

Стеценко І. В.

Київ – 2023

# Завдання 5-1:

A paper with math equations

Description automatically generated

**Виконання**

**Код Генератор 1:**

import random

import math

import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import chi2

n = 10000

lambdas = [0.5, 1.0, 2.0]

significance\_level = 0.05

bins = 20

for lam in lambdas:

    random\_numbers = []

    # Генератор(експоненціальне випадкове число)

    for \_ in range(n):

        Xi = random.random()

        exponential\_random = -(1 / lam) \* math.log(Xi)

        random\_numbers.append(exponential\_random)

    mean = np.mean(random\_numbers)

    std\_dev = np.std(random\_numbers)

    print(f'Lambda (λ) = {lam}')

    print(f'Mean: {mean:.3f}')

    print(f'Standard Deviation: {std\_dev:.3f}')

    #Побудова графіку для згенерованих чисел

    plt.hist(random\_numbers, bins, density=True, alpha=0.8, label=f'λ = {lam}')

    #Значення x для теоретичного PDF і розраховуємо його для лямбди (Probability Density Function)

    x = np.linspace(0, max(random\_numbers), 20)

    pdf = lam \* np.exp(-lam \* x)

    plt.plot(x, pdf, label=f'Theoretical PDF (λ = {lam})', linewidth=2)

    plt.title(f'Exponential Distribution (λ = {lam}')

    plt.xlabel('Random Number')

    plt.ylabel('Probability Density')

    plt.legend()

    # Підрахунок chi^2

    observed, \_ = np.histogram(random\_numbers, bins, density=True)

    expected = pdf \* np.sum(np.diff(x))

    chi\_square\_statistic = np.sum((observed - expected) \*\* 2 / expected)

    degrees\_of\_freedom = len(observed) - 1 - 1

    critical\_chi\_square = chi2.ppf(1 - significance\_level, degrees\_of\_freedom)

    print(f'Chi-Square Statistic: {chi\_square\_statistic:.3f}')

    print(f'Critical Chi-Square Value: {critical\_chi\_square:.3f}')

    print(f'---------------------------------')

    plt.show()

Вивід:   
A screen shot of a graph

Description automatically generated

A screen shot of a graph

Description automatically generated

A screen shot of a graph

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Код Генератор 2:**

import random

import math

import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import chi2

parameters = [

    {'sigma': 1, 'alpha': 0},

    {'sigma': 3, 'alpha': 2},

    {'sigma': 5, 'alpha': 5}

]

bins = 250

n = 10000

significance\_level = 0.05

#Генератор

for idx, param\_set in enumerate(parameters):

    sigma = param\_set['sigma']

    alpha = param\_set['alpha']

    # Генеруємо 12 рандомних чисел (від 0 до 1) xi і рахуємо mu\_i (сума 12 чисел)

    mu\_values = []

    for \_ in range(n):

        xi\_values = [random.random() for \_ in range(12)]

        mu\_i = sum(xi\_values) - 6

        mu\_values.append(mu\_i)

    #Генеруємо фінальні числа на основі

    random\_numbers = [sigma \* mu + alpha for mu in mu\_values]

    plt.figure(idx + 1)

    mean = np.mean(random\_numbers)

    std\_dev = np.std(random\_numbers)

    print(f'Sigma: {sigma:.3f}')

    print(f'Alpha: {alpha:.3f}')

    print(f'Mean: {mean:.3f}')

    print(f'Standard Deviation: {std\_dev:.3f}')

    #Нормальний розподіл

    def normal\_distribution(x, mu, sigma):

        return 1 / (sigma \* math.sqrt(2 \* math.pi)) \* math.exp(-((x - mu) \*\* 2) / (2 \* sigma \*\* 2))

    #Підрахунок chi^2

    observed, bin\_edges = np.histogram(random\_numbers, bins)

    expected = n \* np.diff(bin\_edges) \* np.array([normal\_distribution(x, alpha, sigma) for x in bin\_edges[:-1]])

    degrees\_of\_freedom = len(observed) - 1 - 1 - 1

    chi2\_statistic = np.sum((observed - expected) \*\* 2 / expected)

    critical\_chi2 = chi2.ppf(1 - significance\_level, degrees\_of\_freedom)

    print(f'Chi^2 Statistic: {chi2\_statistic:.3f}')

    print(f'Critical Chi^2 Value: {critical\_chi2:.3f}')

    print(f'-----------------------------------')

    #Гістограма випадкових чисел

    plt.hist(random\_numbers, bins, density=True, alpha=0.8, color='g')

    #Графік нормального розподілу

    x = np.linspace(min(random\_numbers), max(random\_numbers), 1000)

    y = [normal\_distribution(xi, alpha, sigma) for xi in x]

    plt.plot(x, y, 'r--')

    plt.xlabel('Random Numbers')

    plt.ylabel('Probability Density')

    plt.title(f'Generated Random Numbers vs. Normal Distribution (σ={sigma}, α={alpha})')

    plt.legend(['Expected Normal Distribution', 'Generated Random Numbers'])

plt.show()

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Код Генератор 3:**

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

bins = 200

params = [

    (5\*\*13, 2\*\*31),

    (7\*\*14, 4\*\*33),

    (10\*\*15, 9\*\*29)

]

n = 10000

significance\_level = 0.95

for i, (a, c) in enumerate(params):

    # Ініціалізація порожнього списку

    z\_i = 1

    random\_numbers = []

    # Генератор

    for \_ in range(n):

        z\_i = (a \* z\_i) % c

        x\_i = z\_i / c

        random\_numbers.append(x\_i)

    mean = np.mean(random\_numbers)

    std\_dev = np.std(random\_numbers)

    print(f'Mean: {mean:.3f}')

    print(f'Standard Deviation: {std\_dev:.3f}')

    # Підрахунок chi^2

    observed, \_ = np.histogram(random\_numbers, bins)

    expected = np.full(bins, n / bins)

    chi2, p = stats.chisquare(observed, f\_exp=expected)

    degrees\_of\_freedom = len(observed) - 1 - 1 - 1

    critical\_chi2 = stats.chi2.ppf(significance\_level, degrees\_of\_freedom)

    print(f'Chi^2 Statistic: {chi2:.3f}')

    print(f'Critical Chi^2 Value ({significance\_level}): {critical\_chi2:.3f}')

    print(f'-----------------------------------')

    # Побудова гістограми

    plt.hist(random\_numbers, bins, density=True, alpha=0.8, color='g', label=f'Random Numbers ({i+1})')

    plt.xlabel('Random Number')

    plt.ylabel('Frequency')

    plt.title(f'Random Number Distribution ({i+1})')

    # Побудуйте щільність розподілу для рівномірного розподілу на інтервалі (0, 1)

    x = np.linspace(0, 1, 100)

    plt.plot(x, np.ones\_like(x), 'r', label='Uniform PDF (0-1)')

    plt.xlabel('Random Numbers')

    plt.ylabel('Probability Density')

    plt.show()

A screen shot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

# Висновок 5-1:

В даній лабораторній роботі нам було надано завдання згенерувати випадкові числа за допомогою трьох різних методів і перевірити їх відповідність різним законам розподілу. Ми також мали побудувати гістограми частот для цих чисел і визначити їх вид закону розподілу, а також застосувати критерій згоди χ² для перевірки відповідності згенерованих даних заданим законам розподілу.

У результаті цієї лабораторної роботи ви навчилися генерувати випадкові числа за допомогою різних методів і перевіряти їх відповідність різним законам розподілу. Ви також отримали практичний досвід роботи з гістограмами частот і застосуванням критерію згоди для аналізу випадкових даних.