ПЗ 1

ДОСЛІДЖЕННЯ ЗАКОНІВ РОЗПОДІЛУ ВИПАДКОВИХ ВЕЛИЧИН. МОДЕЛЮВАННЯ ВИПАДКОВИХ ВЕЛИЧИН ЗА РІВНОМІРНИМ ТА НОРМАЛЬНИМ ЗАКОНАМИ РОЗПОДІЛУ

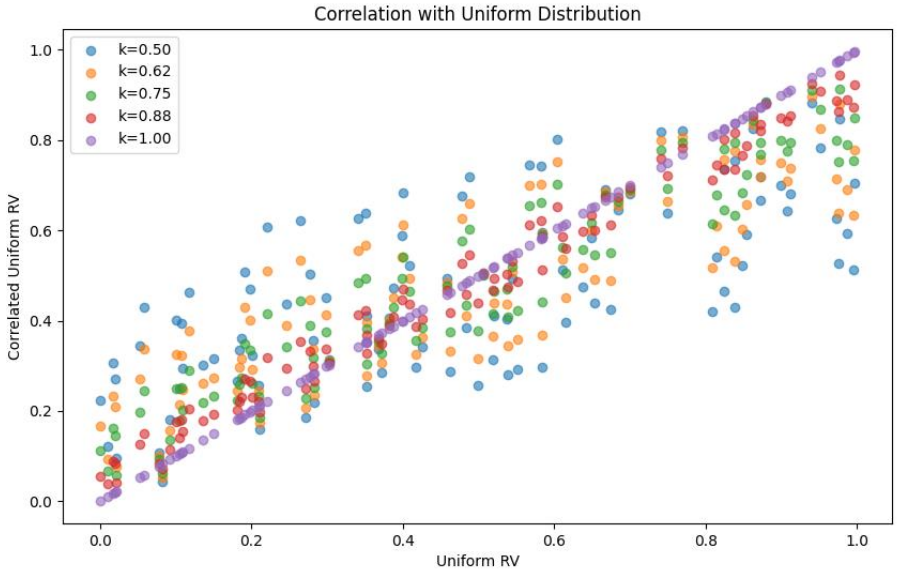
- Вибрати п’ять значень k для випадкових величин, де коефіцієнт кореляції становить:

– для рівномірного закону розподілу $0,5<k<1$;

– для нормального закону розподілу $0<k<1$.

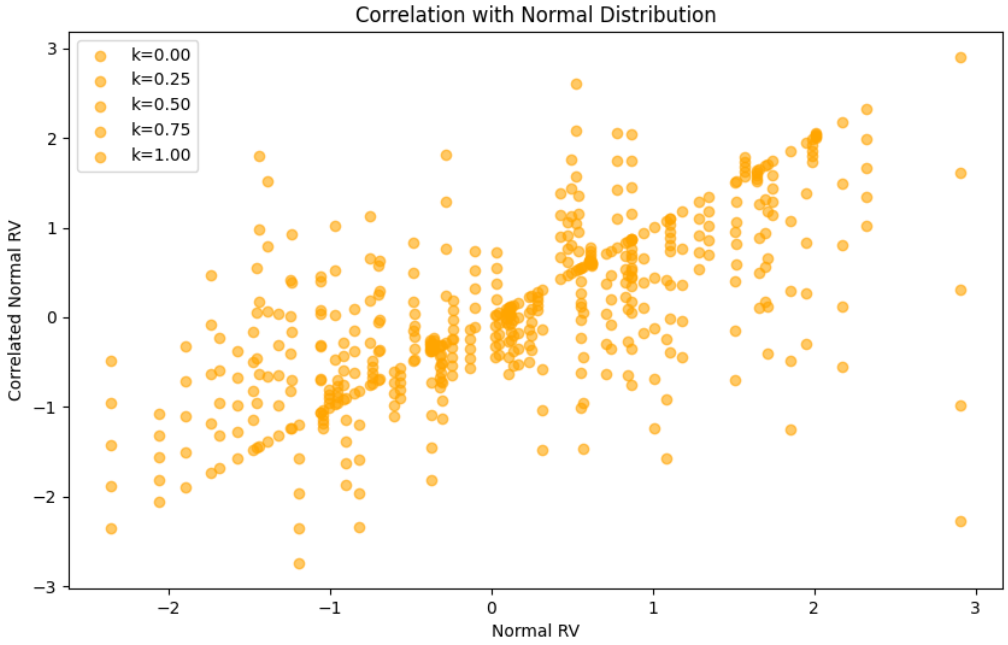
1. Код для кореляції випадкових величин за рівномірним законом розподілу

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

# Функція для генерації випадкових величин
def generate_uniform_variables(n, a, b):
    return np.random.uniform(a, b, n)

# Функція для генерації корельованих випадкових величин
def generate_correlated_variables(n, k, rv1, rv2):
    return k * rv1 + (1 - k) * rv2

# Параметри випадкових величин
n = 100
a, b = 0, 1  # для рівномірного розподілу

# Випадкові величини
uniform_rv = generate_uniform_variables(n, a, b)
normal_rv = generate_uniform_variables(n, a, b)  # друга ВВ теж рівномірна для кореляції

# Вибір значень k для рівномірного розподілу
k_values_uniform = np.linspace(0.5, 1, 5)

# Кореляція та візуалізація
plt.figure(figsize=(10, 6))
for i, k in enumerate(k_values_uniform):
    correlated_uniform = generate_correlated_variables(n, k, uniform_rv, normal_rv)
    plt.scatter(uniform_rv, correlated_uniform, alpha=0.6, label=f'k={k:.2f}')

plt.title('Correlation with Uniform Distribution')
plt.xlabel('Uniform RV')
plt.ylabel('Correlated Uniform RV')
plt.legend()
plt.show()
```



2. Код для кореляції випадкових величин за нормальним законом розподілу

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

# Функція для генерації випадкових величин
def generate_normal_variables(n, mean, std):
    return np.random.normal(mean, std, n)

# Функція для генерації корельованих випадкових величин
def generate_correlated_variables(n, k, rv1, rv2):
    return k * rv1 + (1 - k) * rv2

# Параметри випадкових величин
n = 100
mean, std = 0, 1   # для нормального розподілу

# Випадкові величини
normal_rv = generate_normal_variables(n, mean, std)
uniform_rv = generate_normal_variables(n, mean, std)  # друга ВВ теж нормальна для кореляції

# Вибір значень k для нормального розподілу
k_values_normal = np.linspace(0, 1, 5)

# Кореляція та візуалізація
plt.figure(figsize=(10, 6))
for i, k in enumerate(k_values_normal):
    correlated_normal = generate_correlated_variables(n, k, normal_rv, uniform_rv)
    plt.scatter(normal_rv, correlated_normal, alpha=0.6, label=f'k={k:.2f}', color='orange')

plt.title('Correlation with Normal Distribution')
plt.xlabel('Normal RV')
plt.ylabel('Correlated Normal RV')
plt.legend()
plt.show()
```



ПЗ 2
ДОСЛІДЖЕННЯ ПОМИЛОК КВАНТУВАННЯ У ДИСКРЕТНИХ ЦИФРОВИХ СИСТЕМАХ

Номер варіанта	Функція	Діапазон x_0	Кількість рівнів квантування N
1	$\sin(x)$	[0;15]	10

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

# Функція для створення синусоїди
def generate_sine_wave(freq, sample_rate, end_time):
    t = np.linspace(0, end_time, int(sample_rate * end_time), endpoint=False)
    sine_wave = np.sin(2 * np.pi * freq * t)
    return t, sine_wave

# Функція для квантування сигналу
def quantize_signal(signal, num_levels):
    min_val = np.min(signal)
    max_val = np.max(signal)      You, 20 minutes ago • new subject
    step = (max_val - min_val) / num_levels
    quantized_signal = np.round((signal - min_val) / step) * step + min_val
    return quantized_signal, step

# Параметри синусоїди
freq = 1 / (2 * np.pi)  # Частота, щоб період був 2*pi (один повний цикл sin за 2*pi)
end_time = 15  # Кінець діапазону для x
sample_rate = 1000  # Частота дискретизації

t, sine_wave = generate_sine_wave(freq, sample_rate, end_time)

# Квантування синусоїди
num_levels = 10  # Кількість рівнів квантування
quantized_signal, step = quantize_signal(sine_wave, num_levels)

# Візуалізація
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
plt.plot(t, sine_wave, label='Original Sine Wave')
plt.title('Original Sine Wave')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')

plt.subplot(2, 1, 2)
plt.plot(t, quantized_signal, label='Quantized Sine Wave', color='orange')
plt.title('Quantized Sine Wave')
plt.xlabel('Time [s]')
plt.ylabel('Quantized Amplitude')

plt.tight_layout()
plt.show()
```