****

Институт информационных и вычислительных технологий

Кафедра управления и интеллектуальных технологий

**Лабораторная работа №1**

**«Визуальный анализ временного ряда, методы обнаружения и выделения компонент временного ряда, анализ случайности и стационарности временных рядов»**

**Курс «Методы и алгоритмы обработки данных и изображений»**

Группа: А-02м-25

Студенты: Михайловский Михаил, Озеров Сергей

Бригада: №2

Проверяющий: Бородкин А.А.

**Москва, 2025**

Пункт 1.

a0 = 0.4, a1 = 0.015, a2 = 0.63

|  |  |
| --- | --- |
|  |  |
|  |  |

Пункт 2.

p=0.99

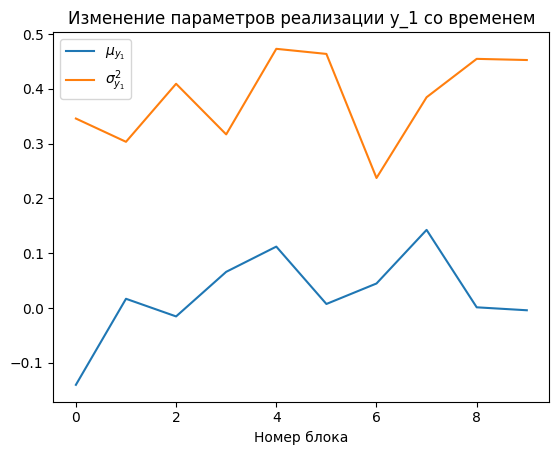
|  |  |
| --- | --- |
|  |  |
| Эллипс рассеивания для y1 | Эллипс рассеивания для y3 |

Размер эллипса скалируется в соответствии с заданной доверительной вероятностью, в предположении принадлежности прямой и конечной разности двумерному нормальному распределению. В данном случае эллипс охватывает 99% вероятности оценки нормального распределения.

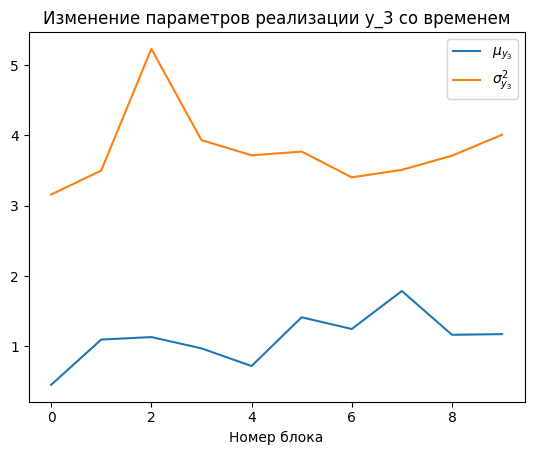
Пункт 3.

а) Критерий восходящих и нисходящих серий

Для y1. Критерий серий тренда на обнаружил



Для y3. Критерий серий тренда не обнаружил



б) Анализ АКФ и СПМ

Для у1

|  |  |
| --- | --- |
|  |  |
| Оценка АКФ | Оценка СПМ |

Для у3

|  |  |
| --- | --- |
|  |  |
| Оценка АКФ | Оценка СПМ |

Вывод о наличии тренда можно сделать только для по АКФ.

Пункт 4.

Характеристики y1

Математическое ожидание - 0.024

Дисперсия - 0.383

Асимметрия - 0.179

Эксцесс - 0.254

Тест на нормальность Колмогорова-Смирнова - pvalue=9.5e-08

Характеристики y3

Математическое ожидание - 1.121

Дисперсия - 3.844

Асимметрия - 0.049

Эксцесс - -0.132

Тест на нормальность Колмогорова-Смирнова - pvalue=7.4e-64

Пукт 5.

Реализация корреляционной функции.

def estimate\_m(arr: np.array) -> float:

    N = arr.shape[0]

    return (arr @ np.ones(N)) / N

def correlation\_function(x: np.array, y: np.array, lags\_count: int):

    if x.shape[0] != y.shape[0]:

        raise ValueError('x and y must be the same size')

    N = x.shape[0]

    if lags\_count > N:

        raise ValueError('lags\_count should be less than x, y size')

    mx = estimate\_m(x)

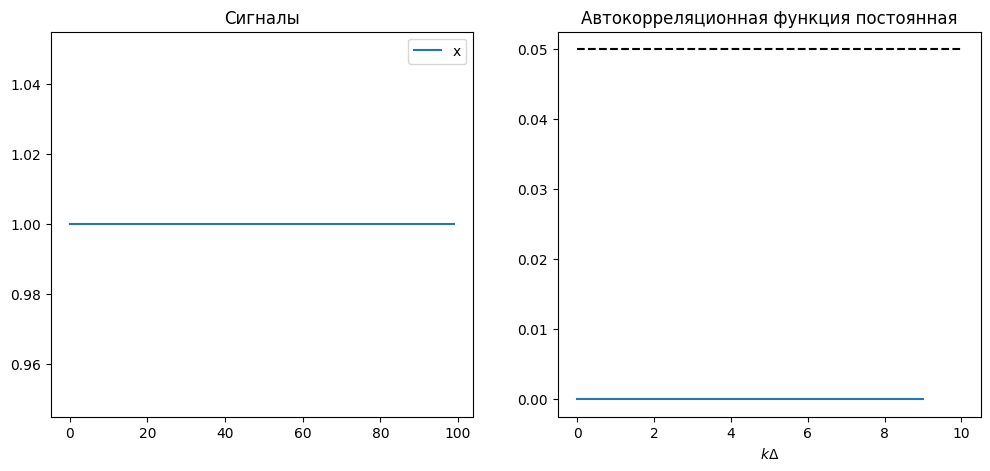
    my = estimate\_m(y)

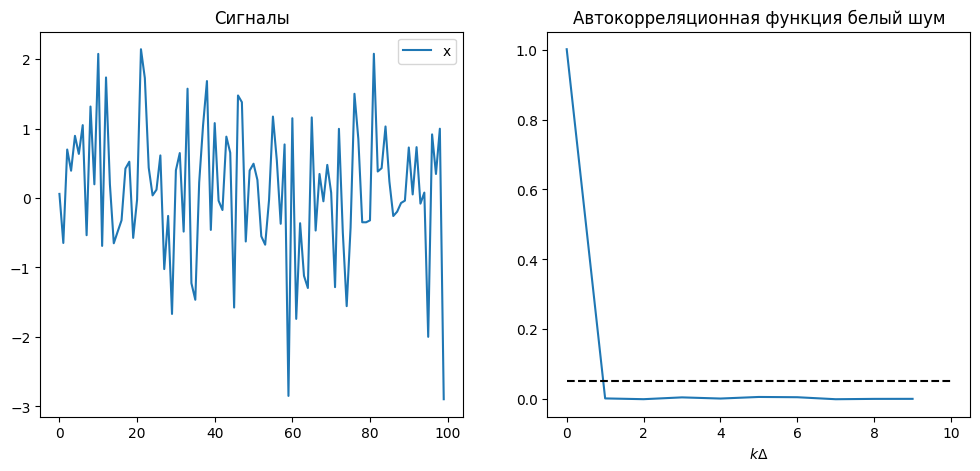
    Rxy = []

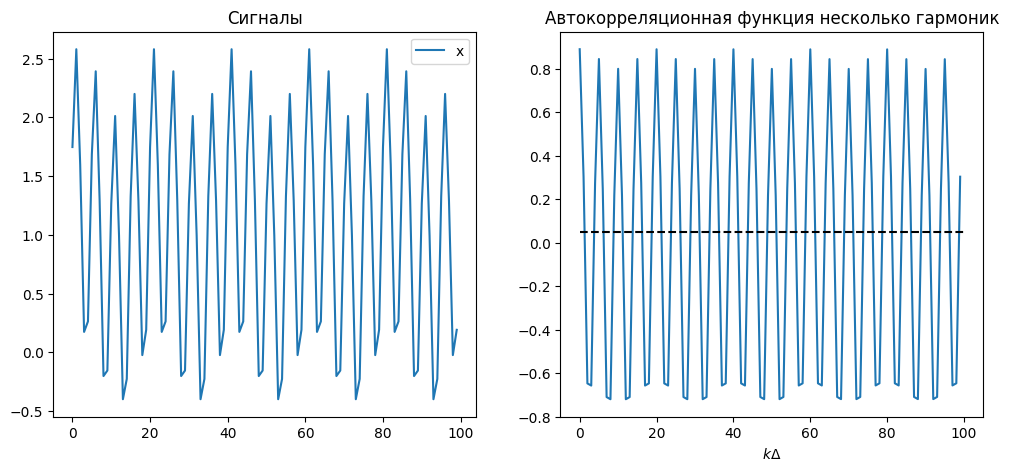
    for k in range(lags\_count):

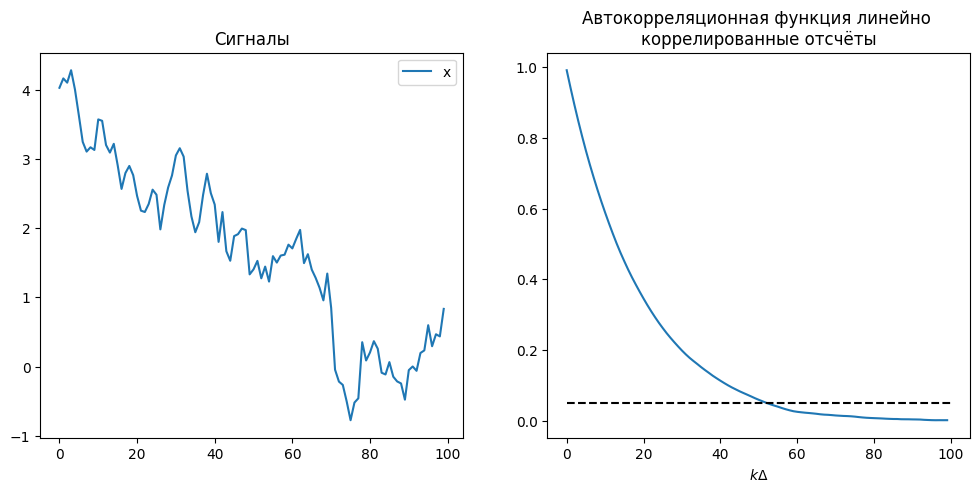
        Rxy.append((x[:N-k] - mx).T @ (y[k:] - my) / N)

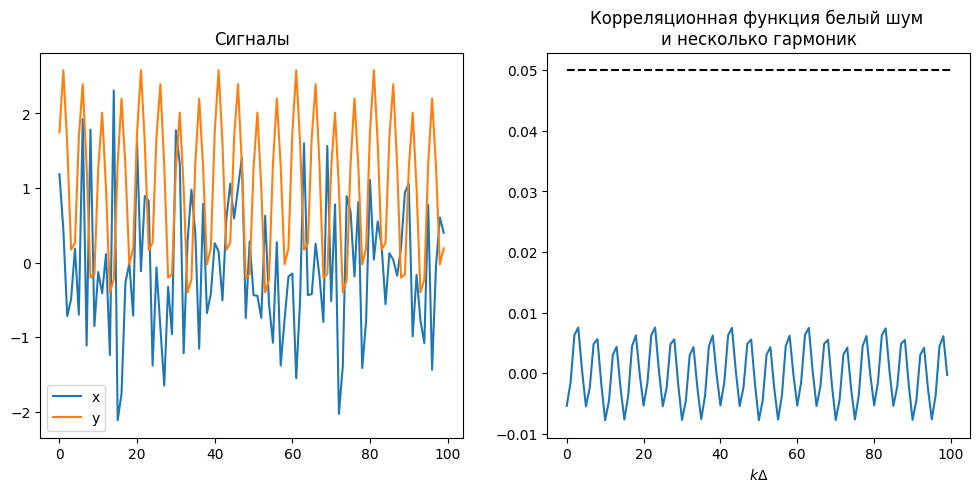
    return Rxy

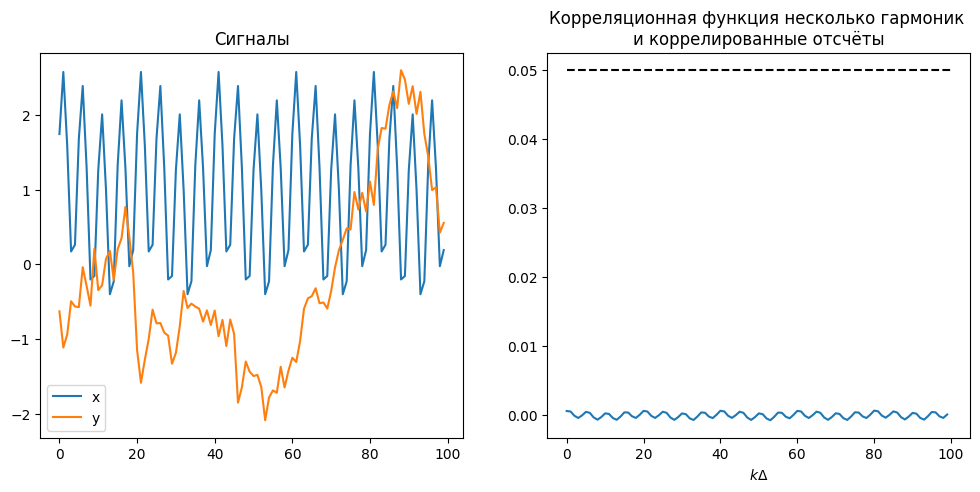


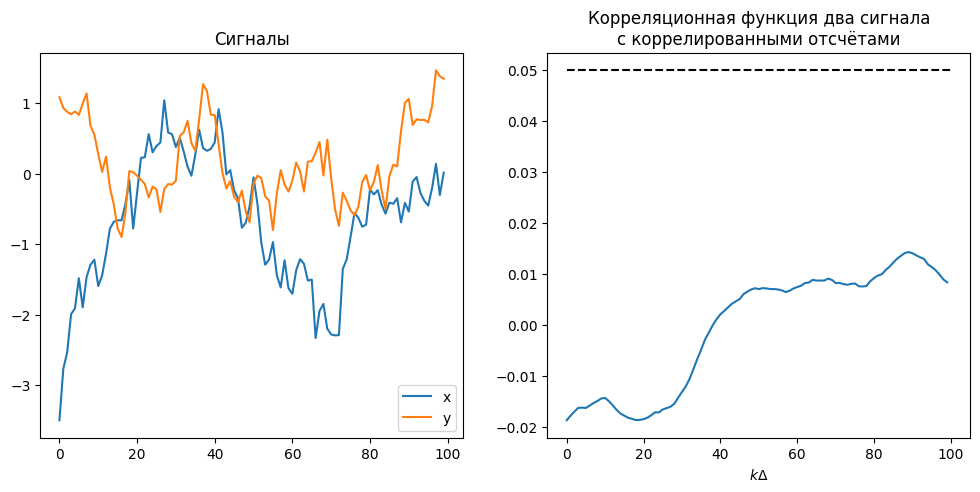












Полученные АКФ соответствуют их теоретическому виду. Время максимальной корреляции можно определить по пересечению графиком уровня 0.05.

Пункт 6.

Реализация периодограммной оценки СПМ

def periodogram(x: np.array, delta):

    N = x.shape[0]

    I = []

    i = np.array(range(N))

    for k in range(int(N / 2)):

        alpha = x.T @ np.cos( (2 \* np.pi \* k \* i) / N )

        beta = x.T @ np.sin( (2 \* np.pi \* k \* i) / N )

        I.append(delta / N \* (alpha \*\* 2 + beta \*\* 2))

    return I

# spectral density

def esimate\_spe(x: np.array, delta, window):

    N = x.shape[0]

    Sxx = []

    I = np.array(periodogram(x, delta))

    j = np.arange(-N / 4 + 1, N / 4, 1)

    for k in range(int(N / 4)):

        indices = np.abs( np.astype((k - j), np.int16) )

        Sxx.append( I[indices].T @ window(j) )

    return Sxx

def rect\_window(j, M = 100):

    w = np.where(np.abs(j) <= M, 1/(2\*M + 1), 0)

    return w

def bartlett\_window(j, M = 100):

    w = np.where(np.abs(j) <= M, (1 - np.abs(j) / M) / M, 0)

    return w

def hamming\_window(j, M = 100):

    w = np.where(np.abs(j) <= M, ( 0.54 + 0.46 \* np.cos(np.pi \* j / M) ) / (1.08 \* M + 0.08), 0)

    return w

def han\_window(j, M = 100):

    w = np.where(np.abs(j) <= M, (1 + np.cos(np.pi \* j / M)) / (2 \* M), 0)

    return w

