федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий
Кафедра управления и интеллектуальных технологий

Отчёт по лабораторной работе №1
По дисциплине «Методы и алгоритмы обработки данных и изображений»

Выполнили студенты: Михайловский М. Ю., Озеров С. Д.
Группа: А-02м-25
Бригада: 2
Проверил: Бородкин А. А.

Москва 2025

Содержание

1 ПРЕДОБРАБОТКА И АНАЛИЗ СТАЦИОНАРНОСТИ ВРЕМЕННОГО РЯДА

1.1 Моделирование временного ряда

Для анализа было смоделировано два временных ряда (ВР) с числом отсчётов $N = 500$ и длительностью $T = 1$с.

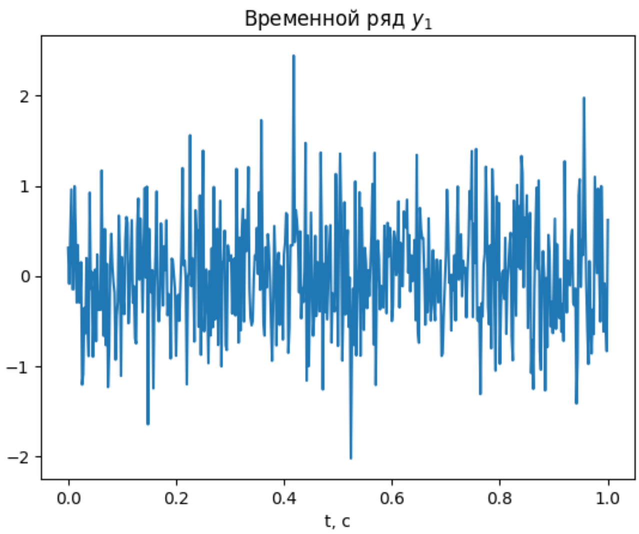
$$y_1 = a_0 \sin(a_1 \cdot 2\pi \cdot t) + a_2 \cdot e$$

$$y_3 = e^{0,1 \cdot t} + 2 \cdot e,$$

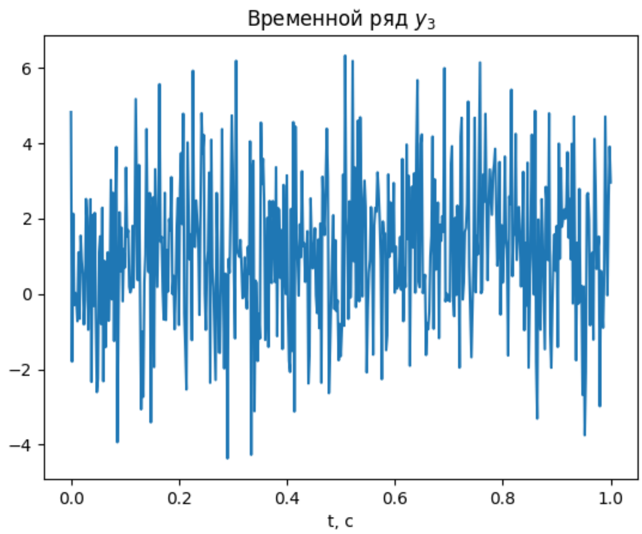
$$\text{где } e \sim N(0,1),\ a_0 = 0{,}4,\ a_1 = 0{,}015,\ a_2 = 0{,}63$$

Генерация данных ВР приведена в листинге 2. Полученные временные ряды представлены на рис. 1.1.

В структуре генерирующей формулы обоих временных рядов заложена неслучайная составляющая. Для $y_1(t)$ заложена периодическая составляющая, а для $y_3(t)$ заложен тренд, представляющий собой экспоненту. Визуально на графиках временных рядов сложно предположить о наличии этих составляющих.



а)



б)

Рис. 1.1. Графики: а) $y_1(t)$, б) $y_3(t)$

1.2 Обработка аномальных наблюдений

Для обнаружения аномалий построим эллипс рассеяния набора данных $(x\ y) = (\Delta y\ \ \nabla y)$. Для этого рассмотрим ковариационную матрицу нормированных данных C.

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \Rightarrow C = \Sigma\big|_{\sigma_x=\sigma_y=1} = \begin{bmatrix} \sigma_x^2 & \rho_{xy}\sigma_x\sigma_y \\ \rho_{xy}\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}\Bigg|_{\sigma_x=\sigma_y=1} = \begin{bmatrix} 1 & \rho_{xy} \\ \rho_{xy} & 1 \end{bmatrix}$$

Собственные значения такой матрицы будут равны:

$$\lambda_{1,2} = 1 \pm \rho_{xy} \tag{1}$$

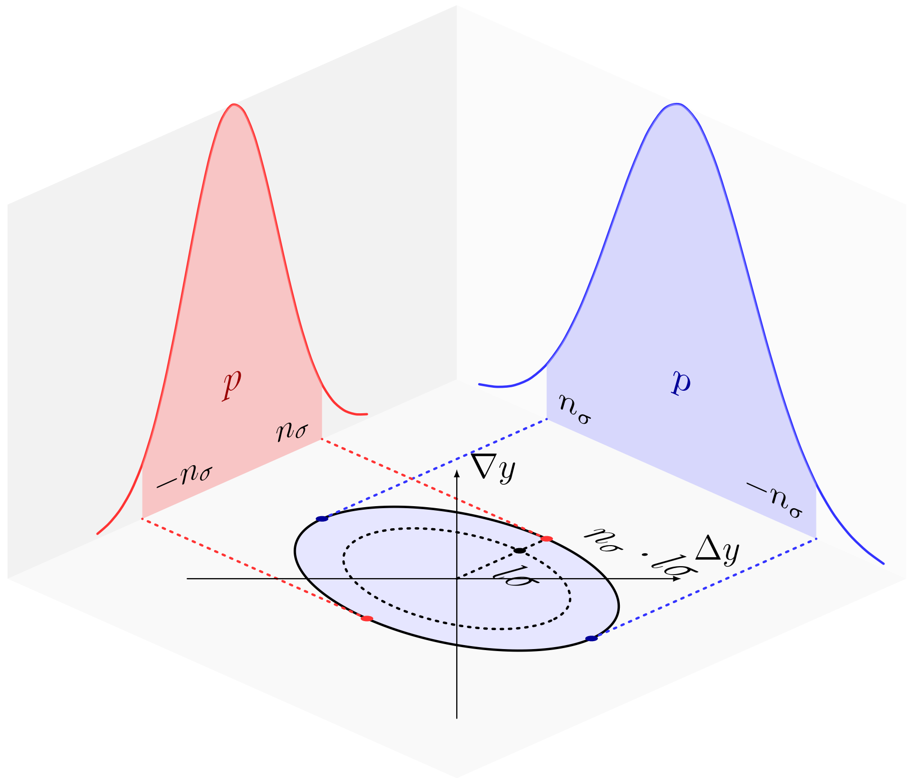


Рис. 1.2. Масштабирование эллипса рассеяния для заданной доверительной вероятности

Система собственных векторов в данном случае будет расположена под углом 45° относительно осей Oxy. Для облака рассеяния нормированных данных длины полуосей эллипса будут равны:

$$l_{1,2} = \sqrt{\lambda_{1,2}} \tag{2}$$

Для перехода от нормированных данных $(x_0\ y_0)$ к исходным $(x\ y)$ эллипс масштабируется с учётом стандартных отклонений при помощи растягивания S:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix}}_{S} \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{3}$$

Вспомним, что рассматривается набор данных $(\Delta y\ \ \nabla y)$. При сравнивании значений по модулю, все значения кроме Δy_N и ∇y_1 встречаются в обоих выборках Δy, ∇y. Это значит, что дисперсии практически равны: $\sigma_x^2 = \sigma_y^2$, и оператор растягивания S сохранит угол 45° между базисом собственных векторов и исходной системой координат Oxy.

В написанной функции построения эллипса рассеяния (листинг 3) можно задавать значение доверительной вероятности, по умолчанию $p = 0{,}99$. На вход функции подаётся вектор $(x\ y)$, используются предположения нормального распределения этого вектора и равенства дисперсий $\sigma_x^2 = \sigma_y^2$.

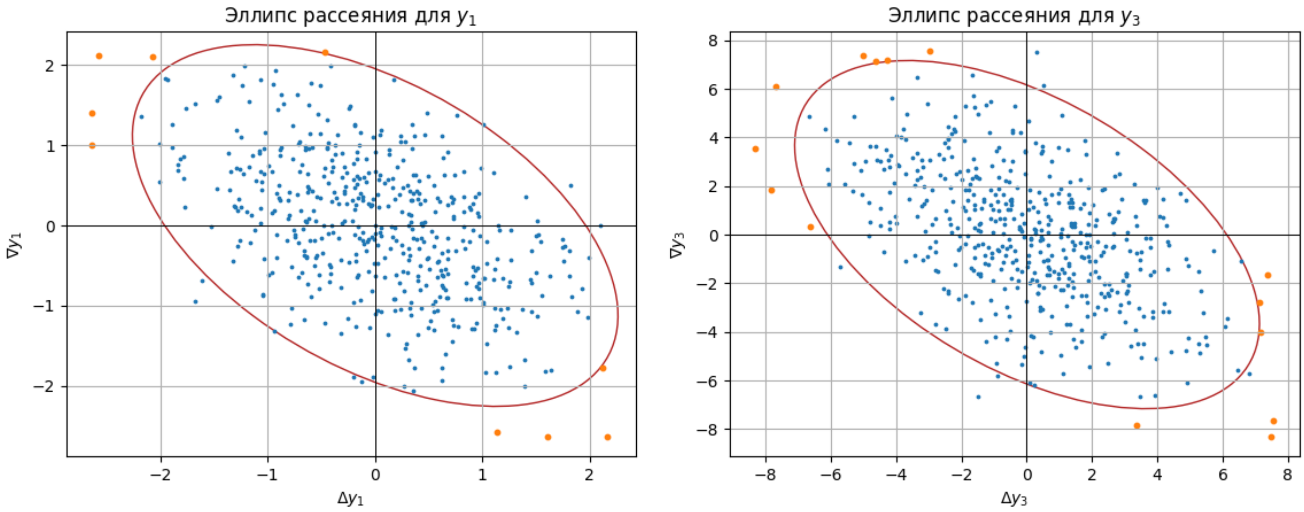
Для учёта доверительной вероятности эллипс рассеяния масштабируется в соответствии с множителем стандартного отклонения (рис. 1.2), показывающим сколько стандартных отклонений должно быть охвачено:

$$n_\sigma = q_{\frac{1+p}{2}},$$

где $q_{\frac{1+p}{2}}$ – квантиль для стандартного нормального распределения. Такое масштабирование сделает сам эллипс кривой, охватывающей p вероятности двумерного нормального распределения.

В качестве входных данных при построении эллипса рассеяния используются передние и задние конечные разности. По оси X имеем $\Delta y = y[k+1] - y[k]$, по оси Y: $\nabla y = y[k] - y[k-1]$. Все значения, которые оказываются вне эллипса рассеяния во II и IV квадрантах – считаются аномальными. Аномальные значения заменяются средним значением между соседними отсчётами во временном ряду.

Полученные эллипсы рассеяния для $y_1(t)$, $y_3(t)$ представлены на рис. 1.3. Они были получены в результате программы, представленной в листинге 4. На графике оранжевым представлены наблюдения, которые были приняты аномальными и заменены средним значением.



а) б)

Рис. 1.3. Эллипс рассеяния: а) $y_1(t)$, б) $y_3(t)$

1.3 Оценка стационарности

Для оценки стационарности временных рядов они были разбиты на 10 блоков по 50 значений. В каждом блоке была рассчитана оценка математического ожидания и дисперсии (листинг 5):

$$\hat{m}_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4}$$

$$\hat{\sigma}_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{m}_x)^2 \tag{5}$$

Полученные графики оценок представлены на рис. 1.4. Визуально по ним сложно утверждать о том, что параметры как либо изменяются. Лишь для $y_3(t)$ можно предположить о слабом изменении математического ожидания.

Критерий серий говорит о стационарности математического ожидания и дисперсии по результату анализа этих же графиков (листинг 1).

Для обоих сигналов критерий Колмогорова-Смирнова для нормального распределения не выполняется. Это говорит о том, что сами сигналы нельзя считать просто нормально распределёнными.

Проанализируем автокорреляционные функции и спектральные плотности мощности для этих сигналов. Автокорреляционные функции представлены на рис. 1.5, а спектральные плотности мощности на рис. 1.6.

Листинг 1. Вывод критерия серий

```
Характеристики y1
Математическое ожидание - 0.024
Дисперсия - 0.383
Асимметрия - 0.179
Эксцесс - 0.254
Тест на нормальность Колмогорова-Смирнова - pvalue=8.1e-01

Характеристики y3
Математическое ожидание - 1.121
Дисперсия - 3.844
Асимметрия - 0.049
Эксцесс - -0.132
Тест на нормальность Колмогорова-Смирнова - pvalue=9.1e-01
```

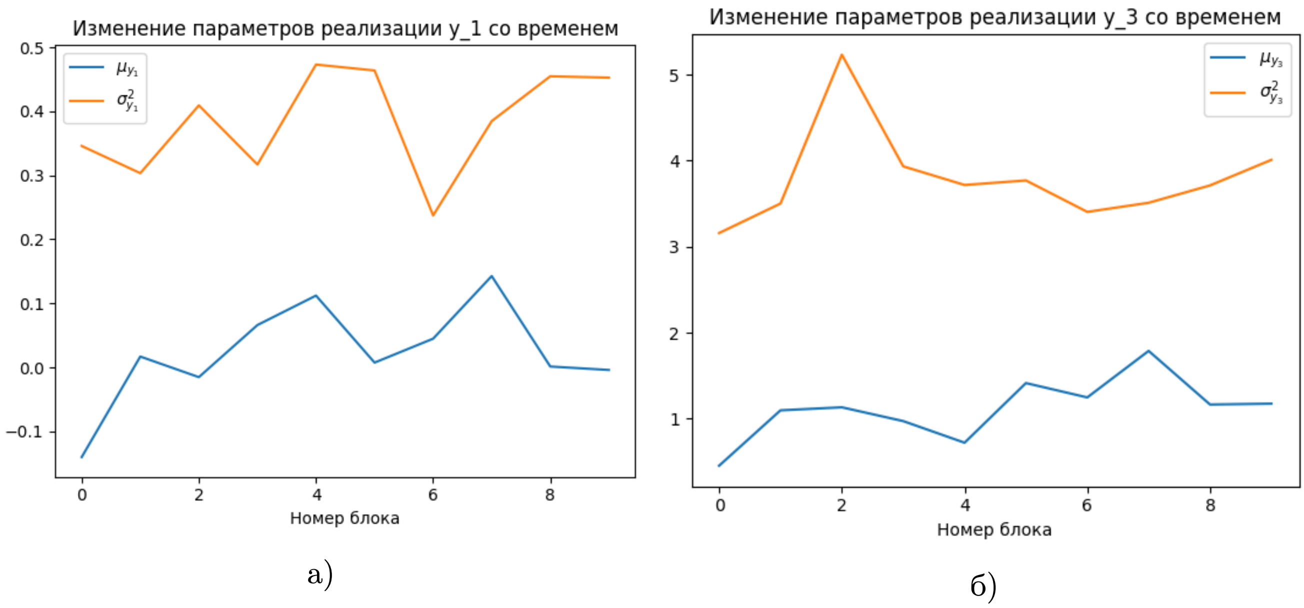
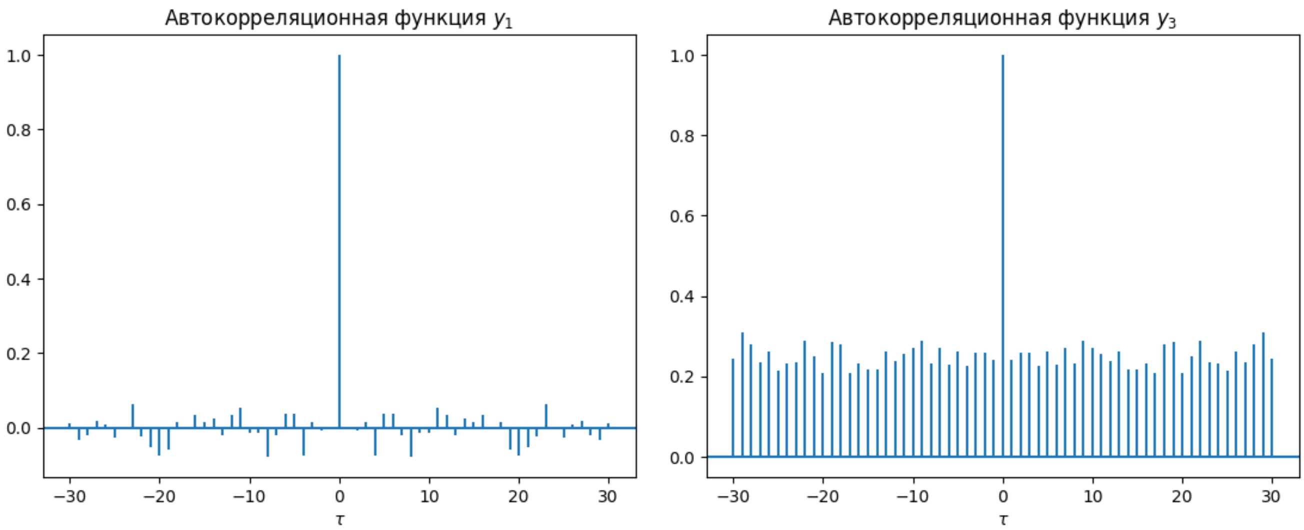


Рис. 1.4. Изменение параметров реализации: а) $y_1(t)$, б) $y_3(t)$

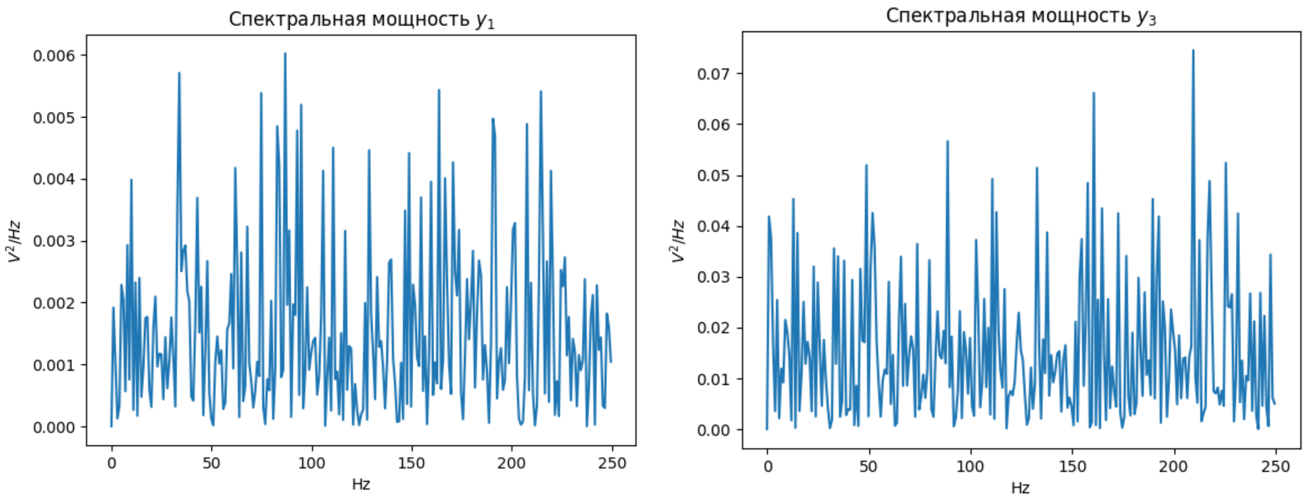
По автокорреляционным функциям можно сказать о наличии тренда в сигнале $y_3(t)$, т.к. эта функция явно не затухает до нуля, что говорит о наличии коррелированности между отсчётами временного ряда. Для $y_1(t)$ нельзя сказать о наличии тренда.

По графикам спектральной плотности мощности рассматриваемых сигналов нельзя сказать о наличии тренда или конкретной частоты. Мощность частот распределена довольно равномерно, что говорит о близости сигналов с белому шуму. В данном случае видно явное преобладание аддитивного шума, который был заложен в модель.



а) б)

Рис. 1.5. Автокорреляционные функции: а) $y_1(t)$, б) $y_3(t)$



а) б)

Рис. 1.6. Спектральная плотность мощности: а) $y_1(t)$, б) $y_3(t)$

2 КОРРЕЛЯЦИОННЫЙ И СПЕКТРАЛЬНЫЙ АНАЛИЗ ВР

2.1 Построение корреляционной функции

Была написана функция *correlation_function*, которая строит корреляционную функцию (КФ) для двух данных сигналов. Реализация функции представлена в листинге 6. Используется формула (6):

$$\hat{R}_{xy}(k\Delta) = \frac{1}{N}\sum_{i=1}^{N-k}(x_i - \hat{m}_x)(y_{i+k} - \hat{m}_y) \tag{6}$$

Полученные сигналы и корреляционные функции представлены на рис. 2.1-2.7. Как видим, получили графики соответствующие теоретическим. Для белого шума автокорреляционная функция (АКФ) $R_{xx}(\tau) = \delta(\tau)$, для нескольких гармоник АКФ представляет собой сигнал тоже из нескольких гармоник, для сигнала с линейно-коррелированными отсчётами АКФ представляет собой затухающую экспоненту.

Таблица полученных интервалов максимальной корреляции $\tau_{\text{м.к.}}$ представлена на рис. 1.

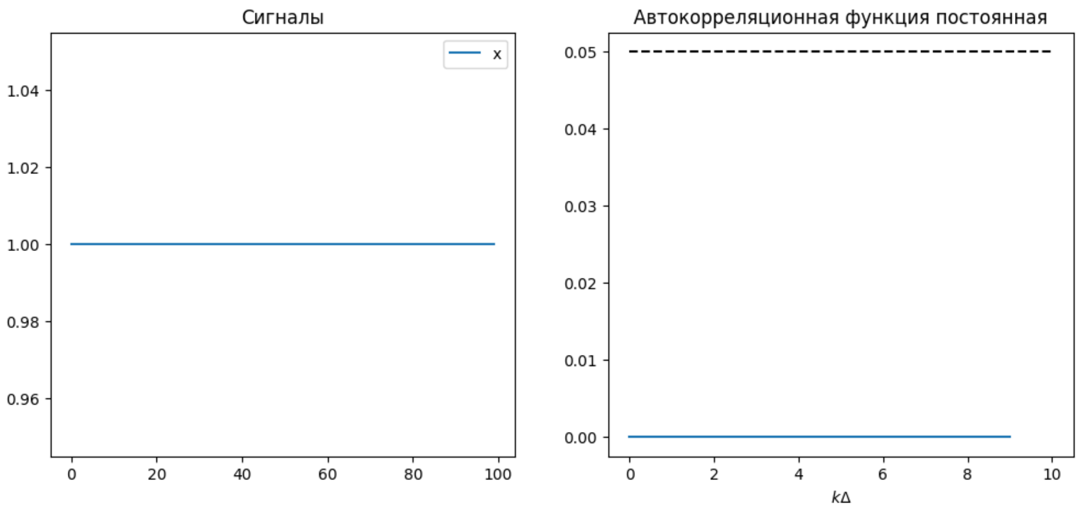


Рис. 2.1. Корреляционная функция, пример 1

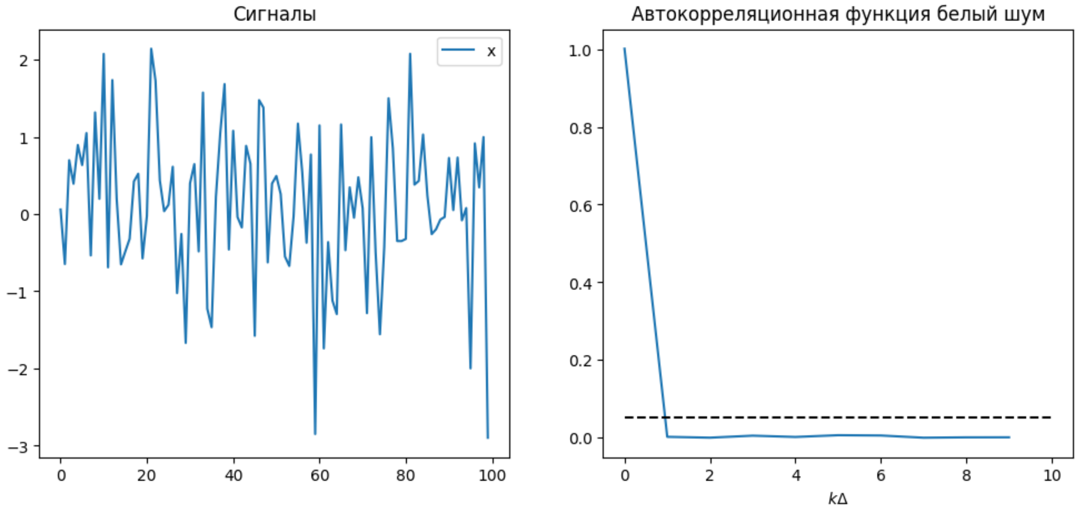


Рис. 2.2. Корреляционная функция, пример 2

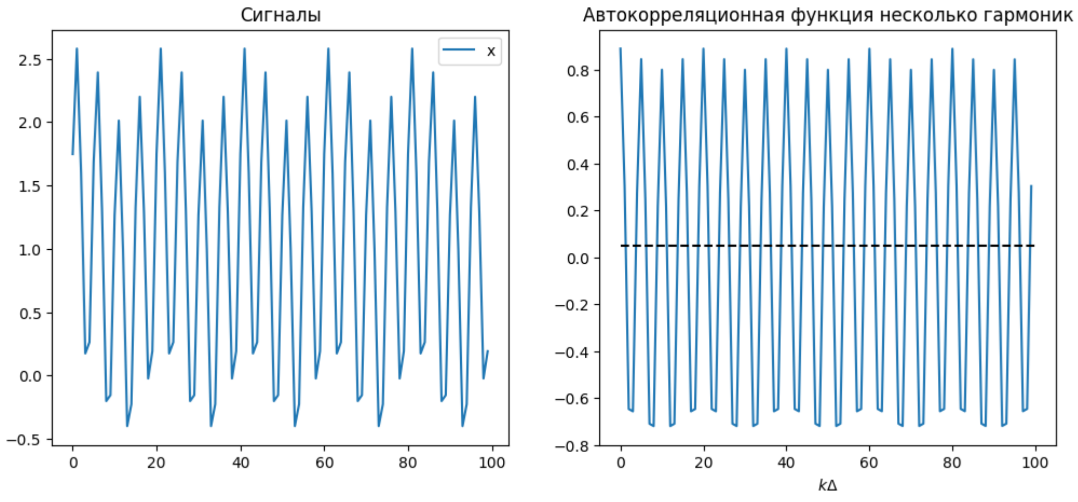


Рис. 2.3. Корреляционная функция, пример 3

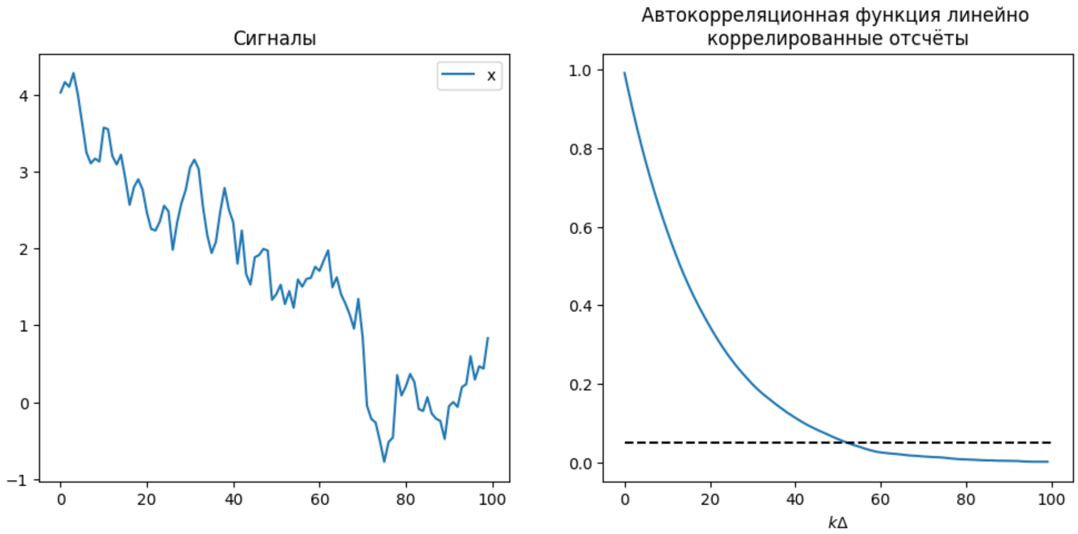


Рис. 2.4. Корреляционная функция, пример 4

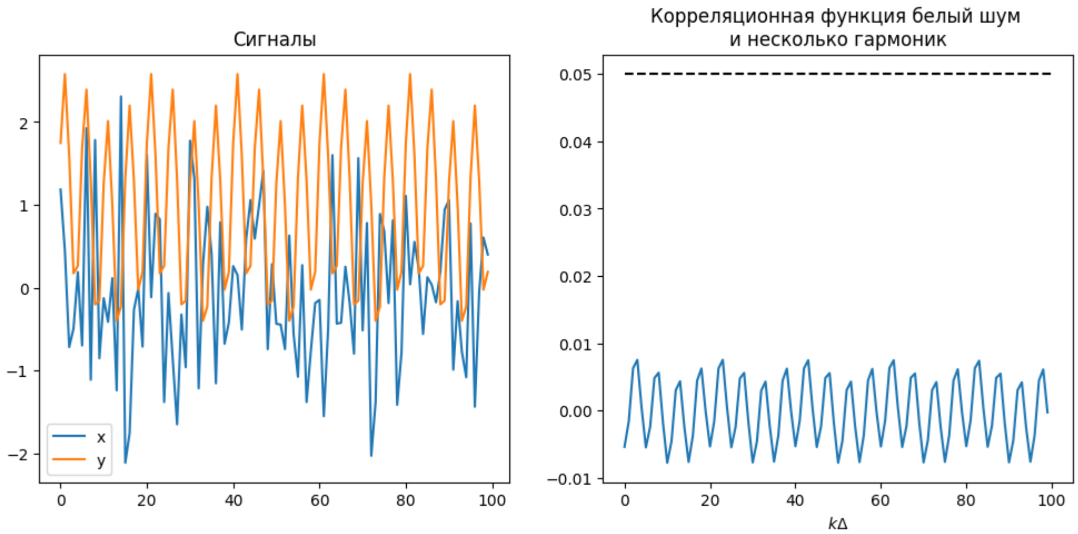


Рис. 2.5. Корреляционная функция, пример 5

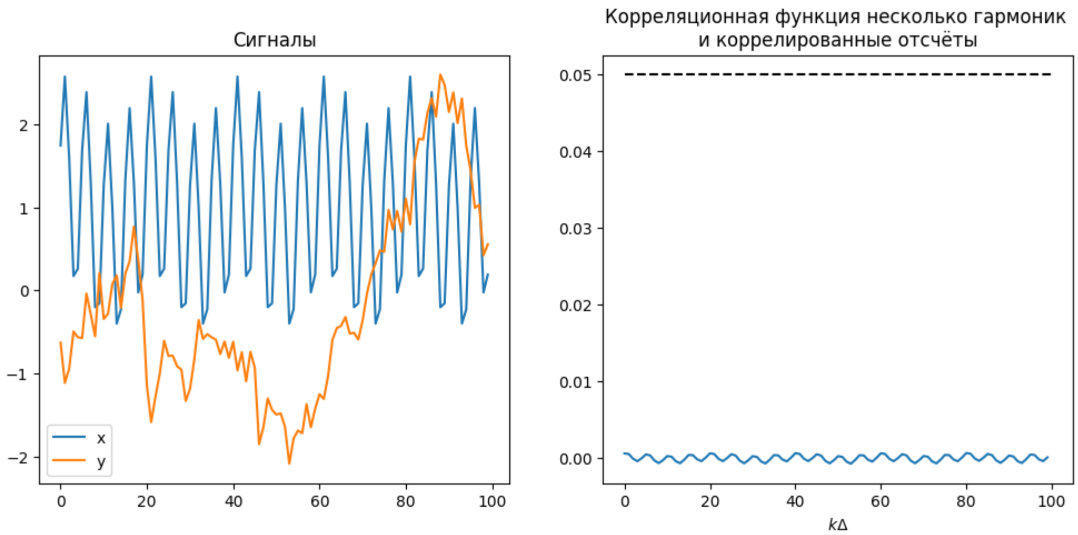


Рис. 2.6. Корреляционная функция, пример 6

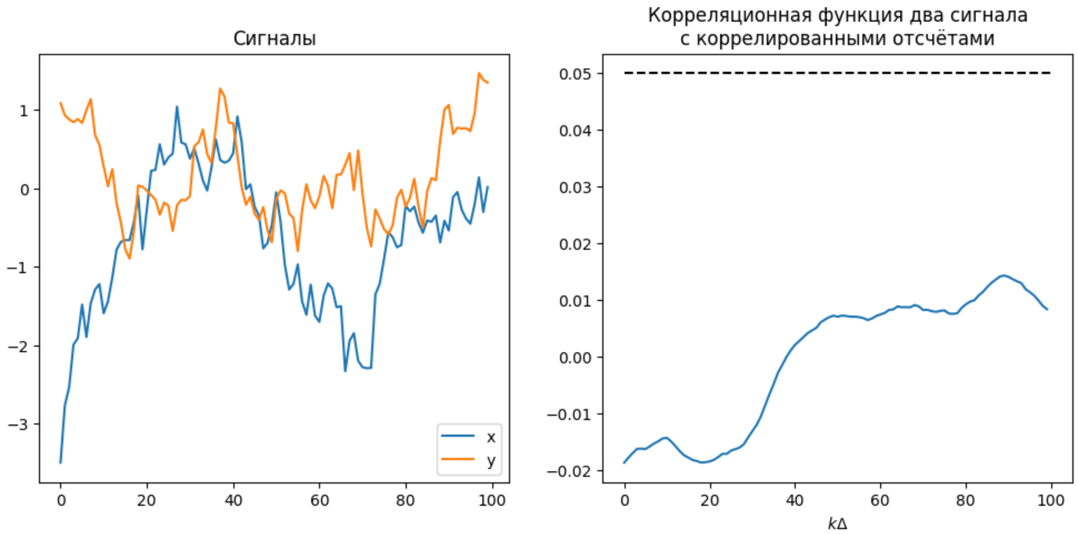


Рис. 2.7. Корреляционная функция, пример 7

Таблица 1. Значение максимальных интервалов корреляции $\tau_{\text{м.к.}}$

Тип сигнала	$\tau_{\text{м.к.}}$
Постоянный	0
Белый шум	1
Несколько гармоник	–
Коррелированные отсчёты	51
Белый шум – несколько гармоник	0
Несколько гармоник – коррелированные отсчёты	0
Коррелированные отсчёты – коррелированные отсчёты	0

2.2 Построение спектральной плотности мощности

Были написаны функции *periodogram* и *estimate_spe*, для построения периодограммы и применения к ней окон. Их реализация приведена в листинге 7. Получили графики спектральных плотностей мощности, представленные на рис. 2.8-2.11. Для всех окон параметр $M = 100$.

Можно отметить, что прямоугольное окно является самым простым, но при сглаживании периодограммы в интервале $[f_0 - M,\ f_0 + M]$ то, насколько далеко значения сглаживаемой функции от f_0, придавая всем одинаковый вес. А за счёт разрыва 1-го рода на границе носителя функции оценка получается кусочно-непрерывной, и разрывный характер визуально заметен.

Окно Хэмминга тоже имеет разрыв 1-го рода на границе носителя функции, но кусочная-непрерывность оценки не так заметна, график выглядит более сглаженным.

Окно Бартлетта имеет излом в нуле, а окно Ханна на границе носителя за счёт чего оценки с этими оконными функциями получаются кусочно-гладкими.

Если применять окно Бартлетта, пики получаются более острыми, за счёт чего они более характерно выделяются, а окно Ханна визуально даёт самые гладкие оценки.

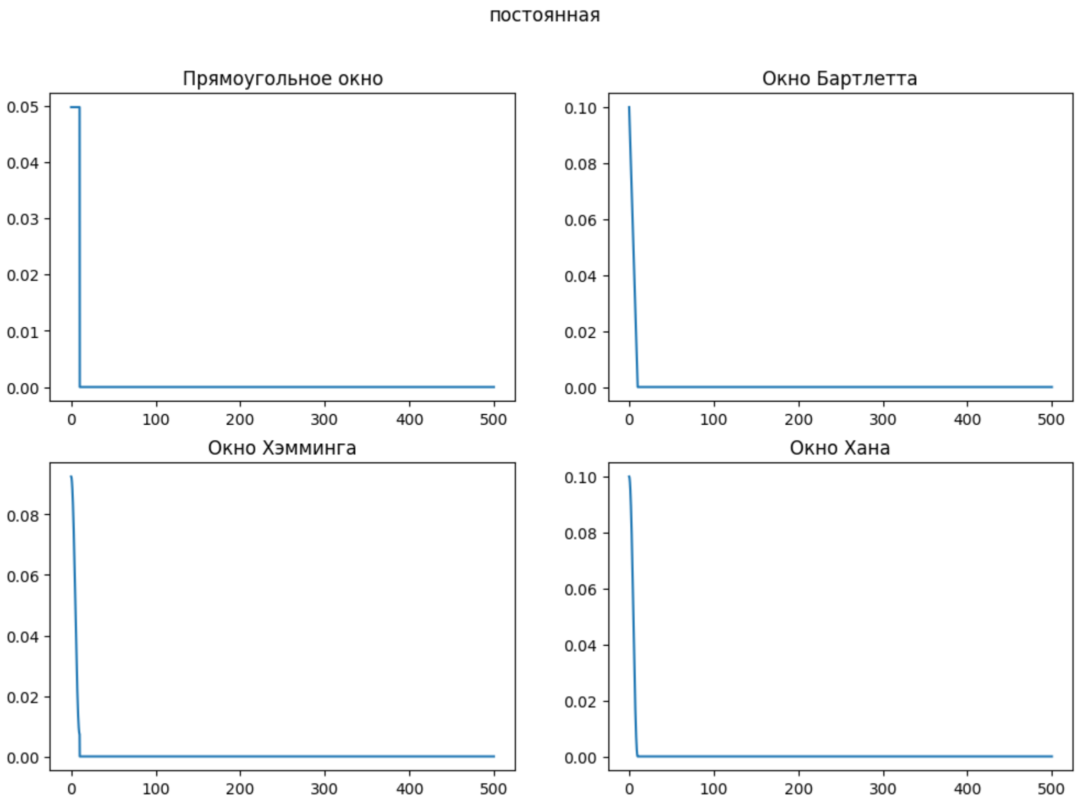


Рис. 2.8. Спектральная плотность мощности, пример 1

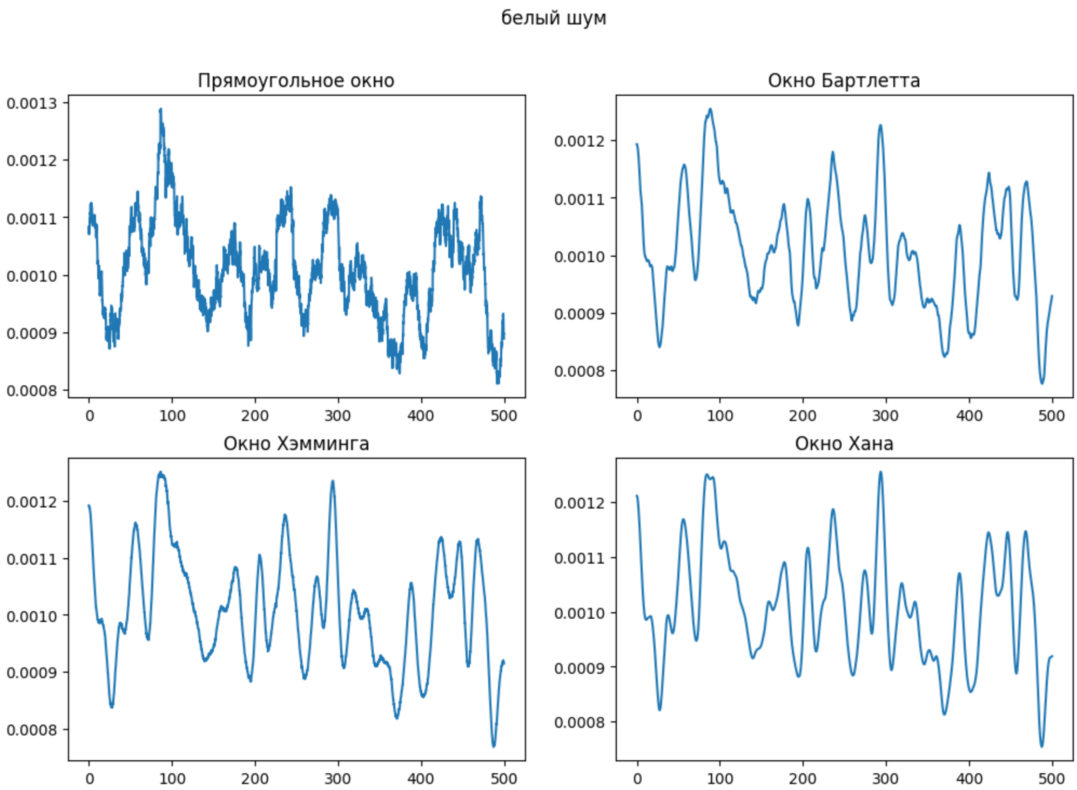


Рис. 2.9. Спектральная плотность мощности, пример 2

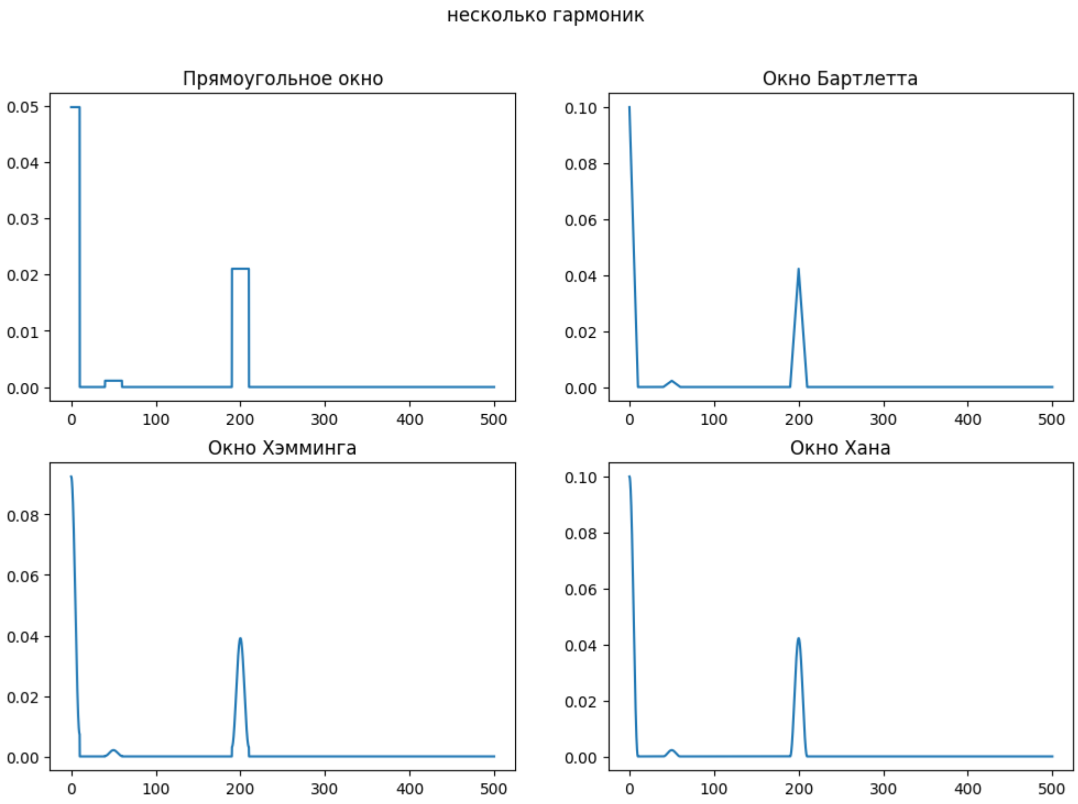


Рис. 2.10. Спектральная плотность мощности, пример 3

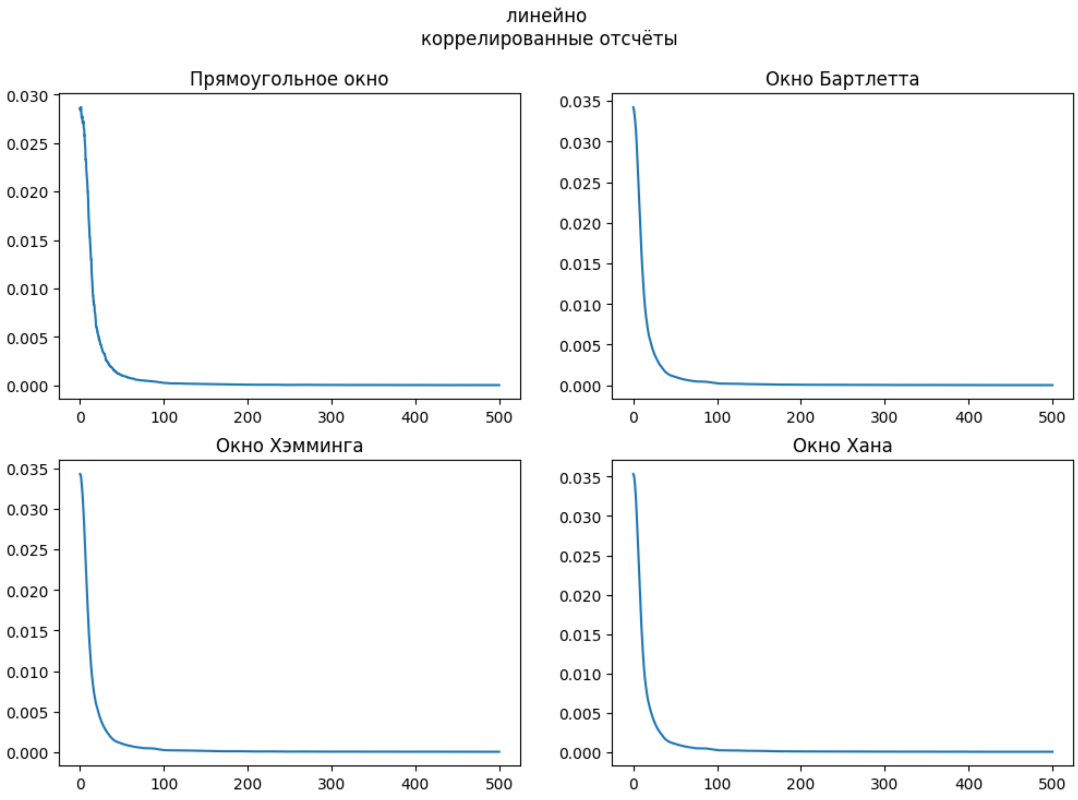


Рис. 2.11. Спектральная плотность мощности, пример 4

Рассмотрим следующие сигналы:

$$x = 2\sin(102t \cdot 2\pi) + 1{,}7\sin(102{,}08t \cdot 2\pi) + 2{,}3\sin(110t \cdot 2\pi) + 0{,}2e$$

$$y = 1{,}6\sin(102{,}8t \cdot 2\pi) + 2{,}1\sin(110t \cdot 2\pi) + 2{,}0\sin(210t \cdot 2\pi) + 0{,}2e$$

где $e \sim N(0, 1)$

Построим для этих сигналов графики оценок спектральной плотности мощности (СПМ) ($\hat{S}_{xx}(f)$, $\hat{S}_{yy}(f)$) и взаимной СПМ ($\hat{S}_{xy}(f)$). Полученные графики представлены на рис. 2.12-2.15.

Как и ожидалось, мы видим пики на частотах, которые были заложены в исходные сигналы. На графике взаимной СПМ пики присутствуют для частот, которые встречаются в обоих сигналах.

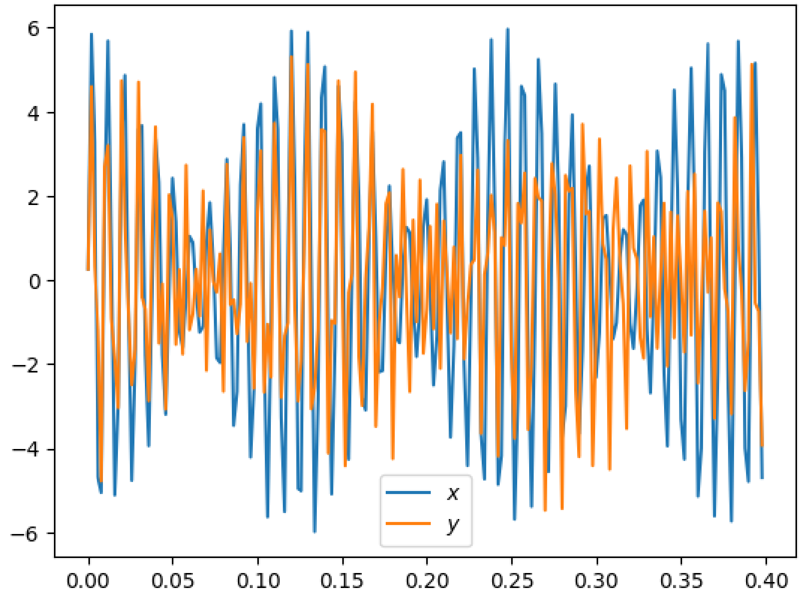


Рис. 2.12. График сигналов x, y.

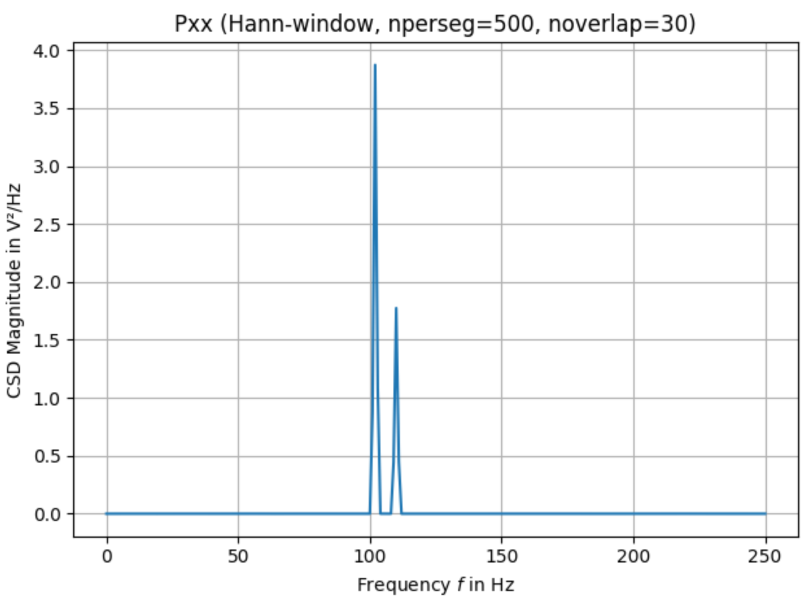


Рис. 2.13. Спектральная плотность мощности

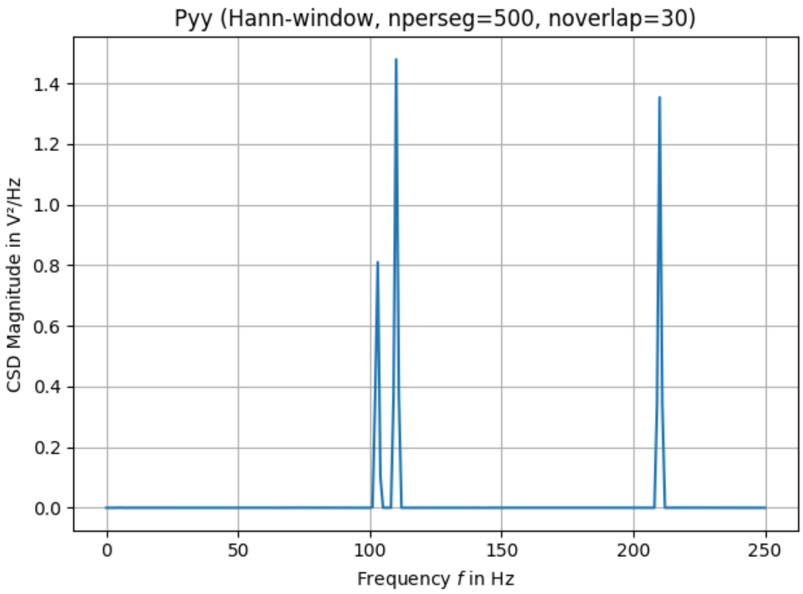


Рис. 2.14. Спектральная плотность мощности

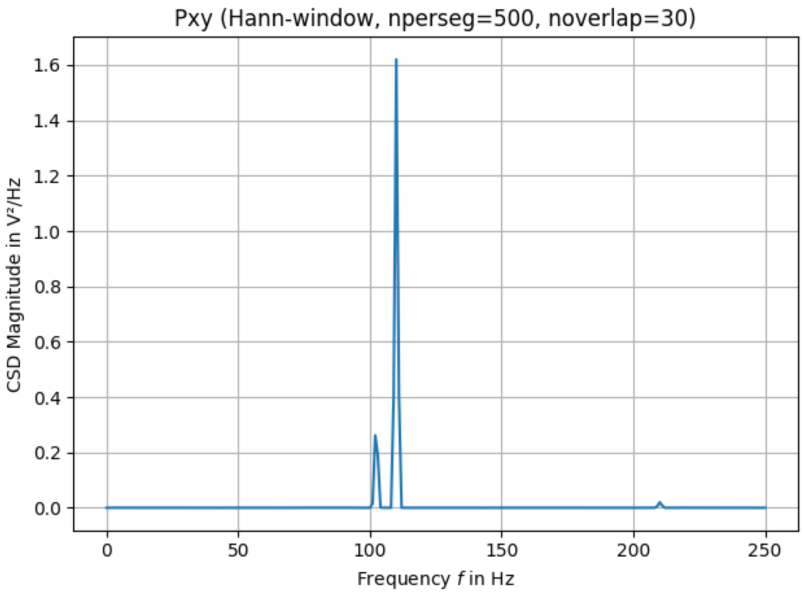


Рис. 2.15. Спектральная плотность мощности

3 ВЫВОДЫ

В данной работе была рассмотрена методика анализа временных рядов. Был рассмотрен и реализован подход к выделению аномальных наблюдений с помощью эллипса рассеяния. Сформировали процедуры определения стационарности временного ряда, за счёт анализа блочных оценок математического ожидания $\hat{m}_y$ и дисперсии $\hat{\sigma}_y^2$. Для рассмотренных временных рядов критерий серий не определил наличия нестационарности. На наличие тренда в сигнале $y_3(t)$ указывал только график автокорреляционной функции.

Были реализованы функции нахождения корреляционной функции $\hat{R}_{xy}(\tau)$ и спектральной плотности мощности $\hat{S}_{xx}(f)$. Для их проверки построили и проанализировали оценки этих характеристик на типовых сигналах. Также сравнили использование различных оконных функций при построении периодограммных оценок.

Приложение А. Листинги

Листинг 2. Генерация временных рядов $y_1(t)$, $y_3(t)$

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
import matplotlib.transforms as transforms
import scipy.stats as stats
from scipy import signal
from collections import namedtuple

N = 500
T = 1 # seconds
delta = T / N
a0 = 0.4
a1 = 0.015
a2 = 0.63

np.random.seed(42)
e1 = np.random.standard_normal(N + 1)
e2 = np.random.standard_normal(N + 1)

k = np.array(range(N + 1))
t = k * delta
y1 = a0 * np.sin(a1 * 2 * np.pi * t) + a2 * e1

y3 = np.exp(0.1 * t) + 2 * e2
```

Листинг 3. Функция построения эллипса рассеяния

```
def confidence_ellipse(x, y, ax, p=0.99, facecolor='none', **kwargs):
        if x.size != y.size:
                raise ValueError("x and y must be the same size")

        cov = np.cov(x, y)
        pearson = cov[0, 1]/np.sqrt(cov[0, 0] * cov[1, 1])
        # Используется частный случай получения собственных значений
        ell_radius_x = np.sqrt(1 + pearson)
        ell_radius_y = np.sqrt(1 - pearson)
        ellipse = Ellipse((0, 0), width=ell_radius_x * 2, height=ell_radius_y * 2,
        facecolor=facecolor, **kwargs)

        # Масштабирование в соответствии с данным pvalue в предположении (x, y) ~ N
        n_std_for_quantile = stats.norm.ppf((1 + p) / 2)

        scale_x = np.sqrt(cov[0, 0]) * n_std_for_quantile
        mean_x = np.mean(x)

        scale_y = np.sqrt(cov[1, 1]) * n_std_for_quantile
        mean_y = np.mean(y)

        # A, такой что xAx^T = 1 описывает эллипс рассеяния до масштабирования и поворота
        A = np.array([ [1/((ell_radius_x) ** 2), 0], [0, 1/((ell_radius_y) ** 2)]])
```

```
        scale = np.array([[1/(scale_x ** 2), 0], [0, 1/ (scale_y ** 2)]])

        angle = np.pi / 4
        rotation = np.array([ [np.cos(angle), -np.sin(angle)], [np.sin(angle), np.cos(angle)] ])

        A = rotation @ scale @ A @ rotation.T
        mu = np.array([mean_x, mean_y])

        transf = transforms.Affine2D() \
                .rotate_deg(45) \
                .scale(scale_x, scale_y) \
                .translate(mean_x, mean_y)

        ellipse.set_transform(transf + ax.transData)
        return ax.add_patch(ellipse), A, mu
```

Листинг 4. Построение эллипса рассеяния

```
def replace_with_mean(arr: np.array, indices):
        result = np.copy(arr)
        for i in indices:
                if i == 0 or (i == result.shape[0] - 1):
                        continue

                result[i] = 0.5 * (result [i - 1] + result[i + 1])
        return result

def ellipse(y, title: str):
        differences = y[1:] - y[:-1]
        forward_diff = differences[1:]
        backward_diff = differences[:-1]

        fig, ax = plt.subplots()

        _, A, mu = confidence_ellipse(forward_diff, backward_diff, ax, edgecolor='firebrick')

        ell_x = np.c_[forward_diff, backward_diff]
        diagonal_values = np.diag((ell_x - mu) @ A @ (ell_x - mu).T)  # вектор длины n
        signes = np.sign(forward_diff * backward_diff)
        anomalies = ell_x[(diagonal_values > 1) & (signes < 1)]
        anomalies_indices = np.where((diagonal_values > 1) & (signes < 1))[0]
        ell_x_clean = replace_with_mean(ell_x, anomalies_indices)

        # Добавление на график ell_x_clean, anomalies

ellipse(y1, '$y_1$')
ellipse(y3, '$y_3$')
```

Листинг 5. Оценка стационарности рядов

```
def estimate_m(arr: np.array) -> float:
        N = arr.shape[0]
        return (arr @ np.ones(N)) / N

```

```python
def estimate_m_var(arr: np.array):
    N = arr.shape[0]
    m = estimate_m(arr)
    return m, ((arr - m).T @ (arr - m)) / (N - 1)

def series_test(arr: np.array):
    N = arr.shape[0]
    arr = arr[1:] - arr[:-1]

    v, t_list = 1, []
    counter, elem_sign = 1, np.sign(arr[0])
    for x in np.nditer(arr[1:]):
        if elem_sign * x >= 0:
            counter += 1
        else:
            v += 1
            t_list.append(counter)
            counter, elem_sign = 1, np.sign(x)
    t_list.append(counter)
    t = np.max(t_list)
    if N <= 6:
        tmax = 5
    elif N <= 154 and N > 6:
        tmax = 6
    else:
        tmax = 7

    no_trend = (v > ((2 * (N - 1)) / 3 - 1.96 * np.sqrt((16 * N - 29) / 90))) and t < tmax
    return no_trend, v, t

def trend_test(y, title):
    mus = []
    vars = []
    for i in range(10):
    arr = y[50 * i: 50 * (i + 1)]
    mu, var = estimate_m_var(arr)
    mus.append(mu)
    vars.append(var)


    no_trend, _, _ = series_test(np.array(mus))
    print(f'По результату критерий серий "Тренд в m({title.replace("$", "")}) отсутствует - {no_trend}"')
    no_trend, _, _ = series_test(np.array(vars))
    print(f'По результату критерий серий "Тренд в var({title.replace("$", "")}) отсутствует - {no_trend}"')
    plt.plot(mus, label="$\mu_{" + title + "}$")
    plt.plot(vars, label="$\sigma_{" + title + "}^2$")
    plt.legend()
    plt.title(f'Изменение параметров реализации {title} со временем')
    plt.xlabel('Номер блока')
    plt.show()

def corr_spectrum(y, title):
    plt.acorr(y, maxlags=30, normed=True)
```

```
        plt.title(f'Автокорреляционная функция {title}')
        plt.xlabel('$\\tau$')
        plt.show()

        f, Pxx = signal.periodogram(y.flatten(), scaling='density', fs=1/delta)
        plt.plot(f, Pxx)
        plt.xlabel('Hz')
        plt.ylabel('$V^2/Hz$')
        plt.title(f'Спектральная мощность {title}')
        plt.show()

trend_test(y1, 'y_1')
corr_spectrum(y1, '$y_1$')

trend_test(y3, 'y_3')
corr_spectrum(y3, '$y_3$')

def print_characteristics(arr: np.array):
        mu, var = estimate_m_var(arr)
        skew = stats.skew(arr)
        kurtosis = stats.kurtosis(arr)
        arr_std = (arr - np.mean(arr)) / np.std(arr, ddof=1)
        kstest = stats.kstest(arr_std, 'norm', alternative='two-sided')

print(f'Математическое ожидание - {mu:.3f}')
print(f'Дисперсия - {var:.3f}')
print(f'Асимметрия - {skew:.3f}')
print(f'Эксцесс - {kurtosis:.3f}')
print(f'Тест на нормальность Колмогорова-Смирнова - pvalue={kstest.pvalue:.1e}')

print('Характеристики y1')
print_characteristics(y1)

print('\nХарактеристики y3')
print_characteristics(y3)
```

Листинг 6. Построение корреляционных функций для разных сигналов

```
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

def plot_corr_function(x: np.array, y: np.array, lags_count: int, title: str):
```

```
        is_autocorr = np.all(x == y)

        if is_autocorr:
                Rxy = correlation_function(x, x, lags_count)
        else:
                Rxy = correlation_function(x, y, lags_count)

        fig, axs = plt.subplots(1, 2, figsize=(12, 5))
        axs[0].set_title('Сигналы')
        axs[0].plot(x[:100], label='x')
        if not is_autocorr:
                axs[0].plot(y[:100], label='y')
        axs[0].legend()

        axs[1].plot(Rxy)
        axs[1].set_title(f'{"Автокорреляционная" if is_autocorr else "Корреляционная"} функция {title}')
        axs[1].set_xlabel('$k\\Delta$')
        axs[1].hlines(0.05, 0, lags_count, color='k', linestyles='dashed')
        plt.show()

def signal_with_garmonics(N, delta = 1e-3):
        k = np.array(range(N))
        return 1 * np.ones(N) + 1.3 * np.sin(200 * 2 * np.pi * k * delta + 0.4) + 0.3 * np.sin(50 * 2 * np.pi * k * delta + 0.93)

def signal_linear_correlated(N, alpha=0.95):
        w = np.random.standard_normal(N)

        x = np.zeros(N)
        x[0] = w[0] / np.sqrt(1 - alpha**2)
        for n in range(1, N):
                x[n] = alpha * x[n-1] + np.sqrt(1 - alpha**2) * w[n]
        return x

Realization = namedtuple('Realization', 'generate_x generate_y lags_to_plot')
realizations = {
        'постоянная': Realization(np.ones, None, 10),
        'белый шум': Realization(np.random.standard_normal, None, 10),
        'несколько гармоник': Realization(signal_with_garmonics, None, 100),
        'линейно \nкоррелированные отсчёты': Realization(signal_linear_correlated, None, 100),
        'белый шум \nи несколько гармоник': Realization(np.random.standard_normal, signal_with_garmonics, 100),
        'несколько гармоник \nи коррелированные отсчёты': Realization(signal_with_garmonics, signal_linear_correlated, 100),
        'два сигнала\nс коррелированными отсчётами': Realization(signal_linear_correlated, signal_linear_correlated, 100)
}

N = 100_000

for title, realization in realizations.items():
        x = realization.generate_x(N)
        if realization.generate_y is None:
                plot_corr_function(x, x, realization.lags_to_plot, title)
```

```
        else:
                y = realization.generate_y(N)
                plot_corr_function(x, y, realization.lags_to_plot, title)
```

Листинг 7. Построение спектральных плотностей мощности для разных сигналов

```
def periodogram(x: np.array, delta):
        N = x.shape[0]
        I = []

        i = np.array(range(N))
        for k in range(int(N)):
        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 / 2, N / 2, 1)
        for k in range(int(N / 2)):
        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

realizations = {
        'постоянная': Realization(np.ones, None, 10),
        'белый шум': Realization(np.random.standard_normal, None, 10),
        'несколько гармоник': Realization(signal_with_garmonics, None, 100),
        'линейно \nкоррелированные отсчёты': Realization(signal_linear_correlated, None, 100),
}

```

```
windows = {
        'Прямоугольное окно': rect_window,
        'Окно Бартлетта': bartlett_window,
        'Окно Хэмминга': hamming_window,
        'Окно Хана': han_window,
}

N = 10000
delta = 1e-3
for title, realization in realizations.items():
        x = realization.generate_x(N)

        fig, axs = plt.subplots(2, 2, figsize=(12, 8))
        for i, window_name in enumerate(windows):
                Sxx = esimate_spe(x, delta, windows[window_name])
                f = np.arange(len(Sxx)) / (delta * N)

                plot_index = int(i / 2), i % 2
                axs[plot_index].set_title(window_name)
                axs[plot_index].plot(f, Sxx)

        plt.suptitle(title)
        plt.show()
```

Листинг 8. Построение спектральных плотностей мощности для разных сигналов

```
N = 1500
T = 3 # seconds
delta = T / N
fs = 1 / delta

k = np.array(range(N + 1))
t = k * delta

e = np.random.standard_normal(N + 1)
x = 2 * np.sin(102 * t * 2 * np.pi) + 1.7 * np.sin(102.08 * t * 2 * np.pi) + 2.3 * np.sin(110 * t * 2 * np.pi) + 0.2 * e
y = 1.6 * np.sin(102.8 * t * 2 * np.pi) + 2.1 * np.sin(110 * t * 2 * np.pi) + 2 * np.sin(210 * t * 2 * np.pi) + 0.2 * e
plt.plot(t[:200], x[:200], label='$x$')
plt.plot(t[:200], y[:200], label='$y$')
plt.legend()
plt.show()

# Compute and plot the magnitude of the cross spectral density:
nperseg, noverlap, win = 500, 30, 'hann'


def plot_CSD(csd, csd_name: str):
        fig0, ax0 = plt.subplots(tight_layout=True)
        ax0.set_title(f"{csd_name} ({win.title()}-window, {nperseg=}, {noverlap=})")
        ax0.set(xlabel="Frequency $f$ in Hz", ylabel="CSD Magnitude in V²/Hz")
        ax0.plot(f, np.abs(csd))
        ax0.grid()
        plt.show()
```

```python

f, Pxx = signal.csd(x, x, fs, win, nperseg, noverlap)
plot_CSD(Pxx, 'Pxx')
f, Pyy = signal.csd(y, y, fs, win, nperseg, noverlap)
plot_CSD(Pyy, 'Pyy')
f, Pxy = signal.csd(x, y, fs, win, nperseg, noverlap)
plot_CSD(Pxy, 'Pxy')
```