



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: Бородин А. А.

Содержание

1	Предобработка и анализ стационарности временного ряда	3
1.1	Моделирование временного ряда	3
1.2	Обработка аномальных наблюдений	3
1.3	Оценка стационарности	5
2	Корреляционный и спектральный анализ ВР	7
2.1	Построение корреляционной функции	7
2.2	Построение спектральной плотности мощности	10
A	Листинги	15

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.

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

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

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.

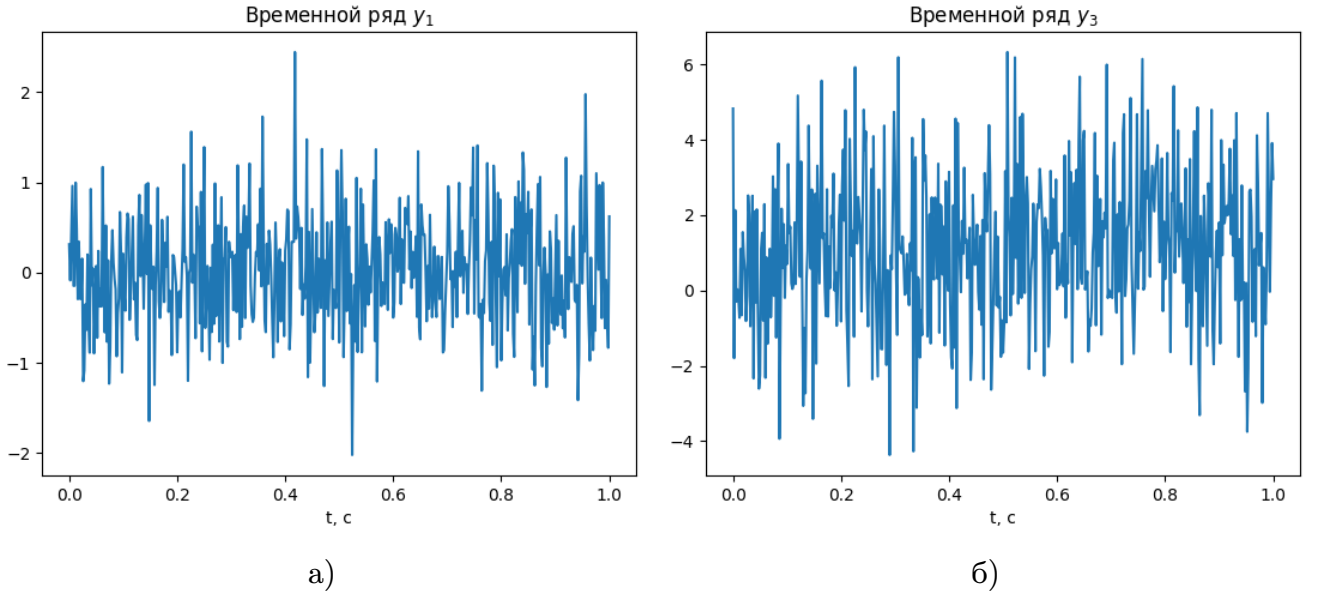


Рис. 1.1. Графики: а) $y_1(t)$, б) $y_3(t)$

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 .

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

Собственные значения такой матрицы будут равны:

$$\lambda_{1,2} = 1 \pm \rho_{xy} \quad (1)$$

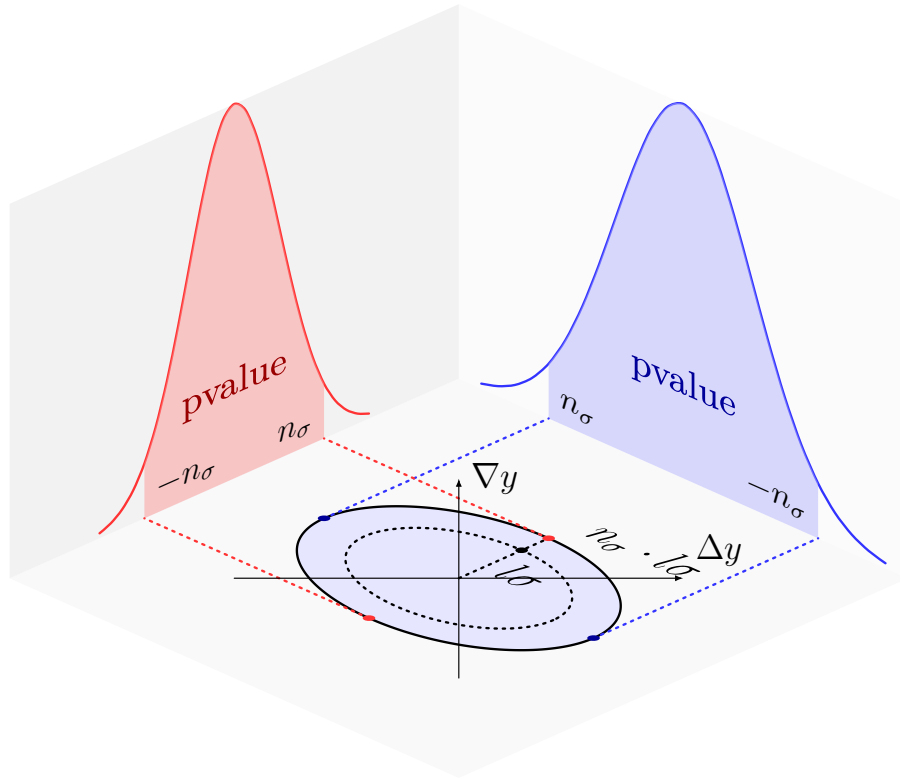


Рис. 1.2. Масштабирование эллипса рассеяния для заданной доверительной вероятности

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

$$l_{1,2} = \sqrt{\lambda_{1,2}} \quad (2)$$

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

$$\begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix}}_S \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (3)$$

Вспомним, что рассматривается набор данных $(\Delta y \ \nabla y)$. При сравнении значений по модулю, все значения кроме Δy_N и ∇y_1 встречаются в обеих 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 $(x \ y)$, и предполагается, что этот вектор имеет двумерное нормальное распределение.

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.

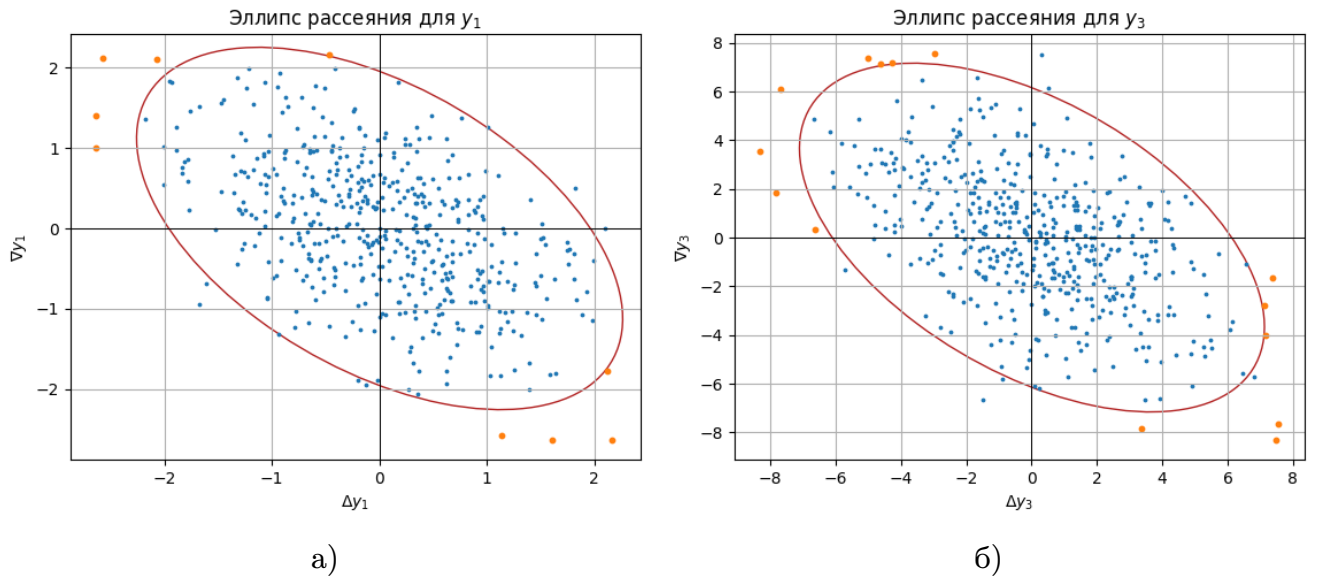


Рис. 1.3. Эллипс рассеяния: а) $y_1(t)$, б) $y_3(t)$

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):

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

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

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

```
По результату критерий серий "Тренд в m(y_1) отсутствует - True"
По результату критерий серий "Тренд в var(y_1) отсутствует - True"
По результату критерий серий "Тренд в m(y_3) отсутствует - True"
По результату критерий серий "Тренд в var(y_3) отсутствует - True"
```

Характеристики y_1

Математическое ожидание - 0.024

Дисперсия - 0.383

Асимметрия - 0.179

Экссесс - 0.254

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

Характеристики y_3

Математическое ожидание - 1.121

Дисперсия - 3.844

Асимметрия - 0.049

Экссесс - -0.132

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

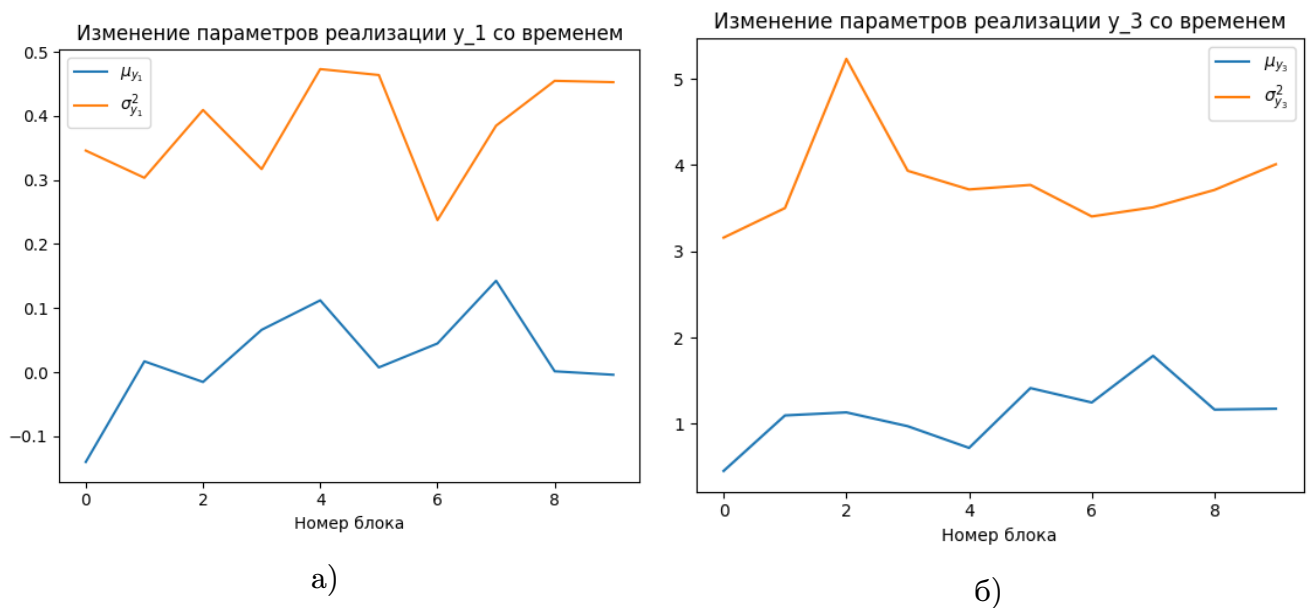


Рис. 1.4. Изменение параметров реализации: а) $y_1(t)$, б) $y_3(t)$

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.

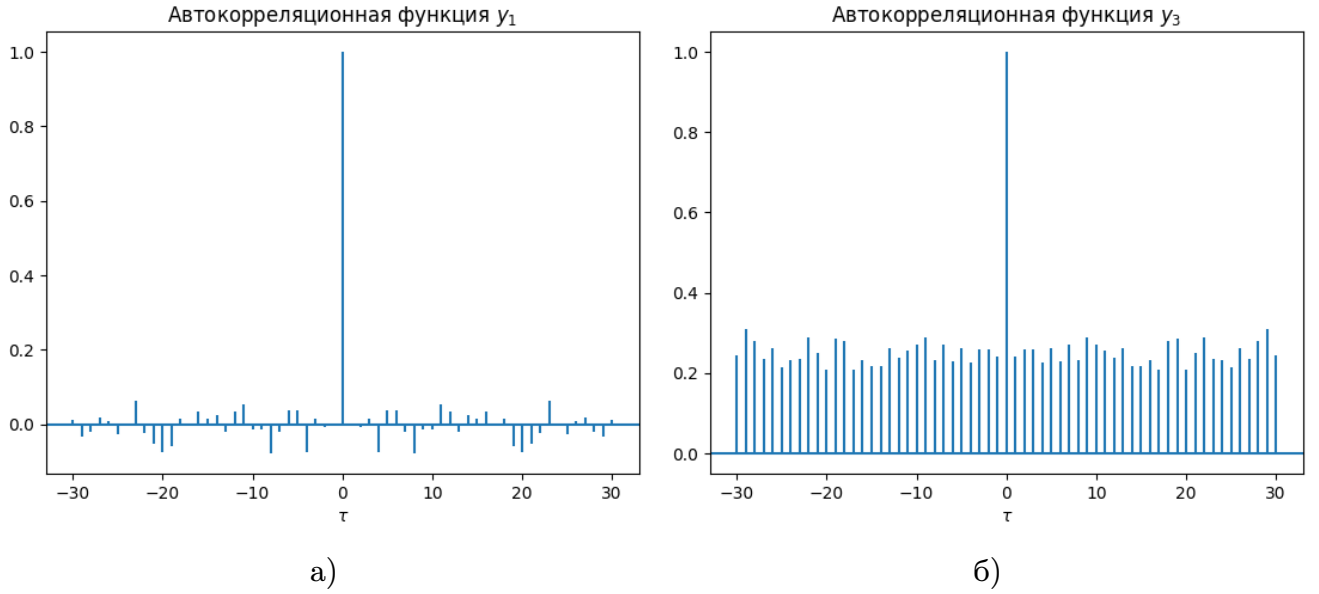


Рис. 1.5. Автокорреляционные функции: а) $y_1(t)$, б) $y_3(t)$

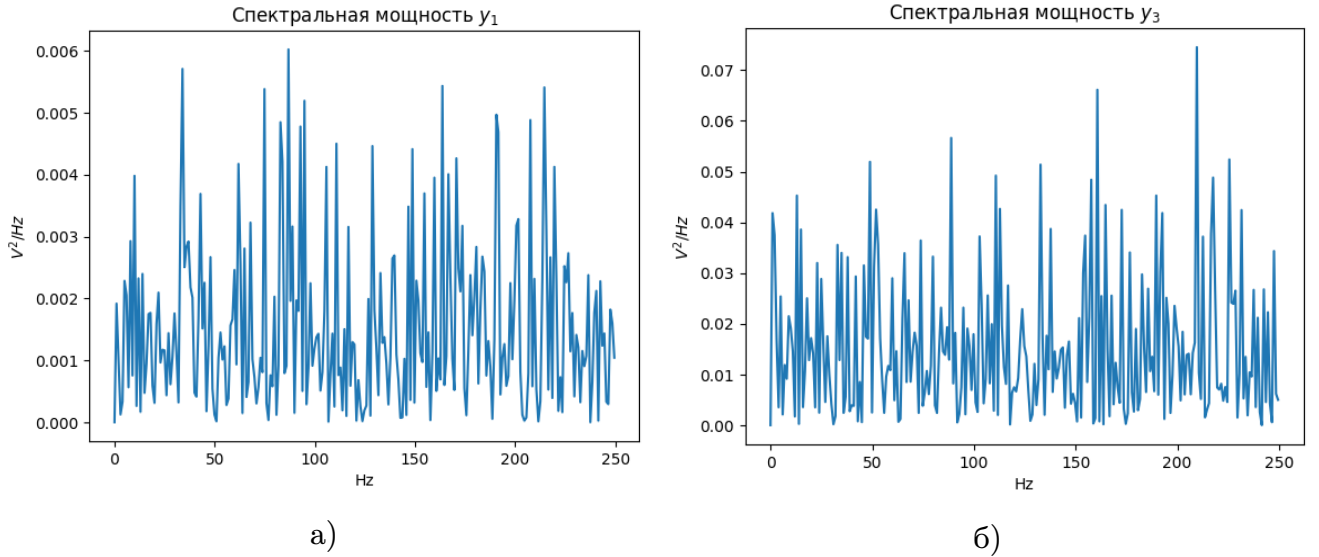


Рис. 1.6. Спектральная плотность мощности: а) $y_1(t)$, б) $y_3(t)$

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):

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

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.

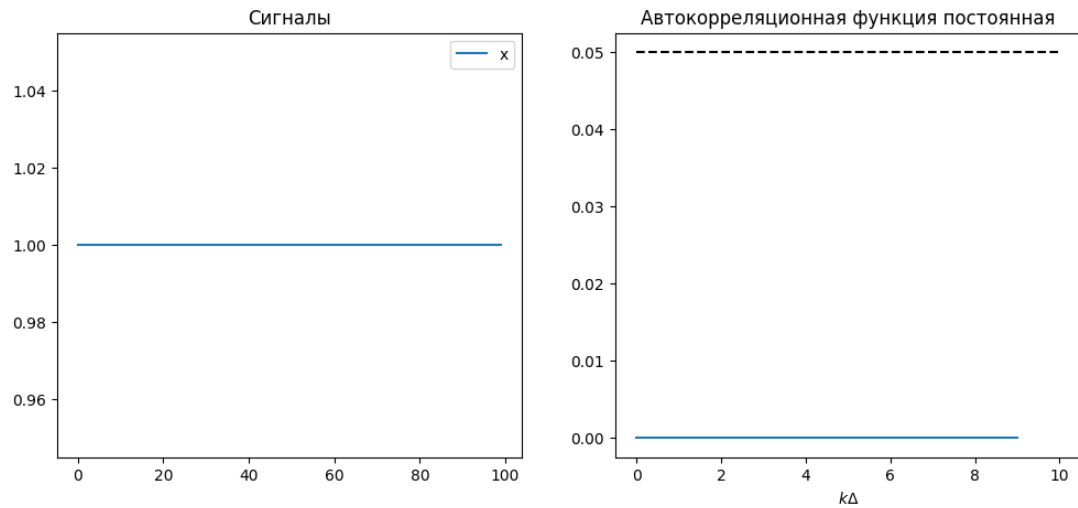


Рис. 2.1. Корреляционная функция, пример 1

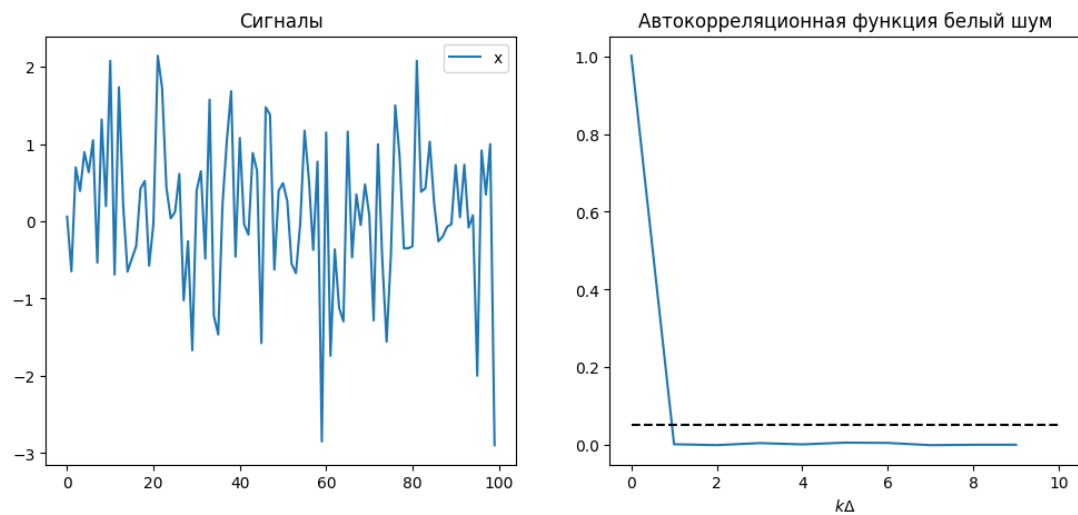


Рис. 2.2. Корреляционная функция, пример 2

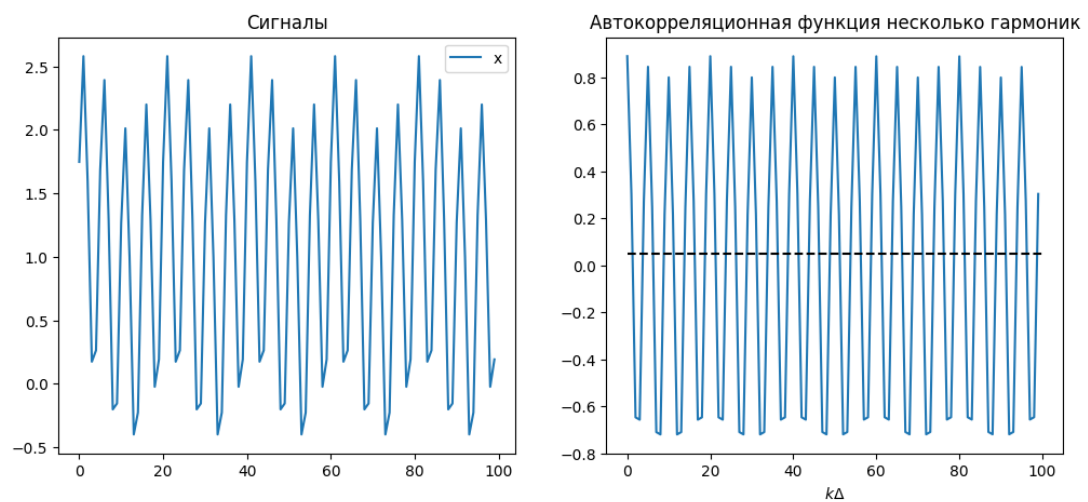


Рис. 2.3. Корреляционная функция, пример 3

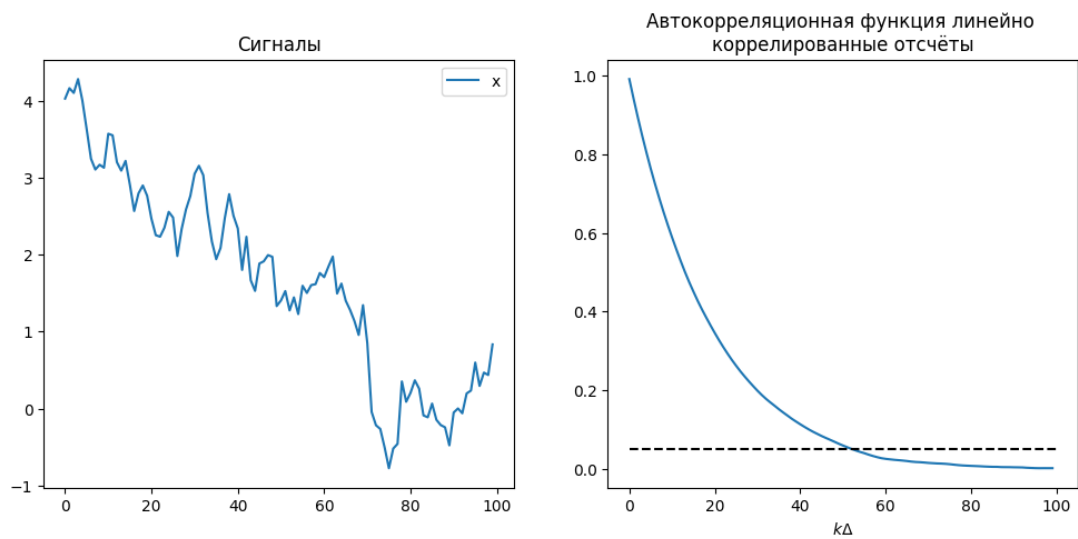


Рис. 2.4. Корреляционная функция, пример 4

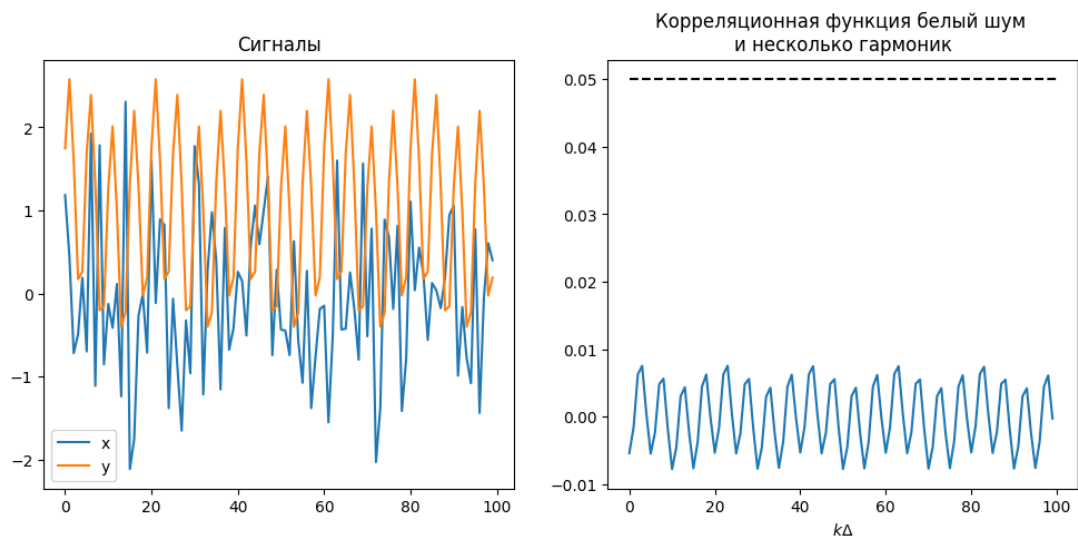


Рис. 2.5. Корреляционная функция, пример 5

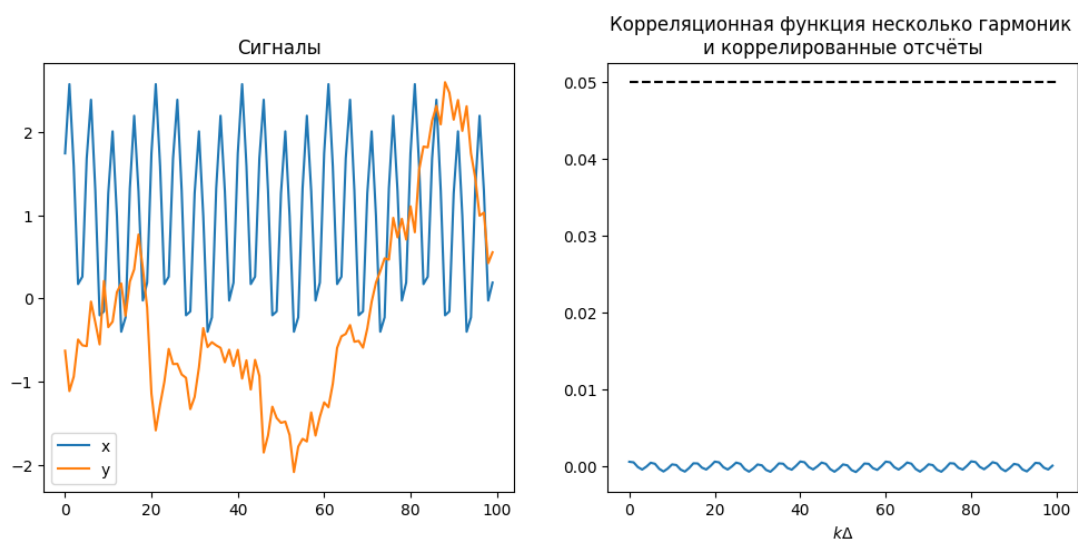


Рис. 2.6. Корреляционная функция, пример 6

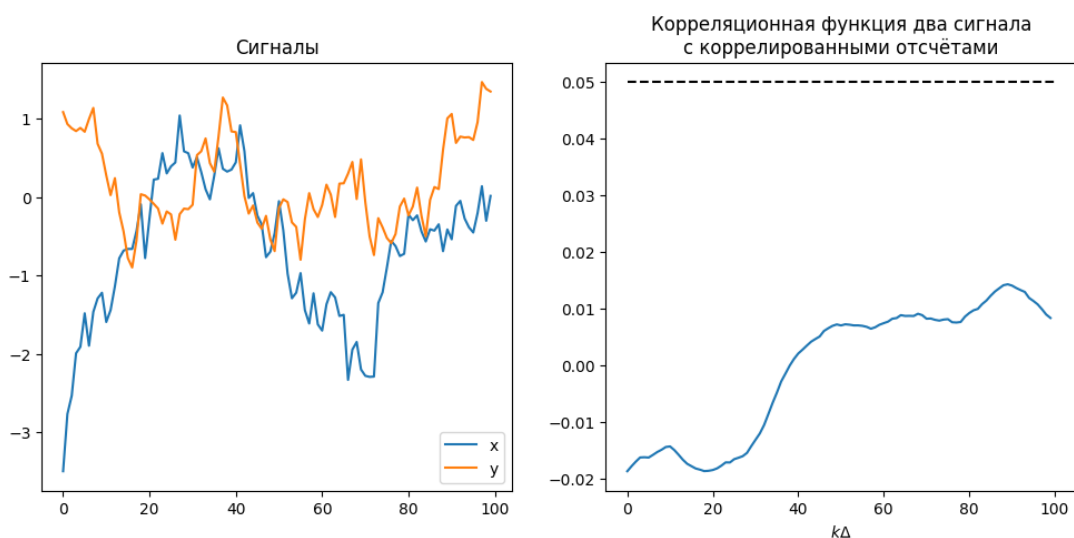


Рис. 2.7. Корреляционная функция, пример 7

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 $M = 100$.

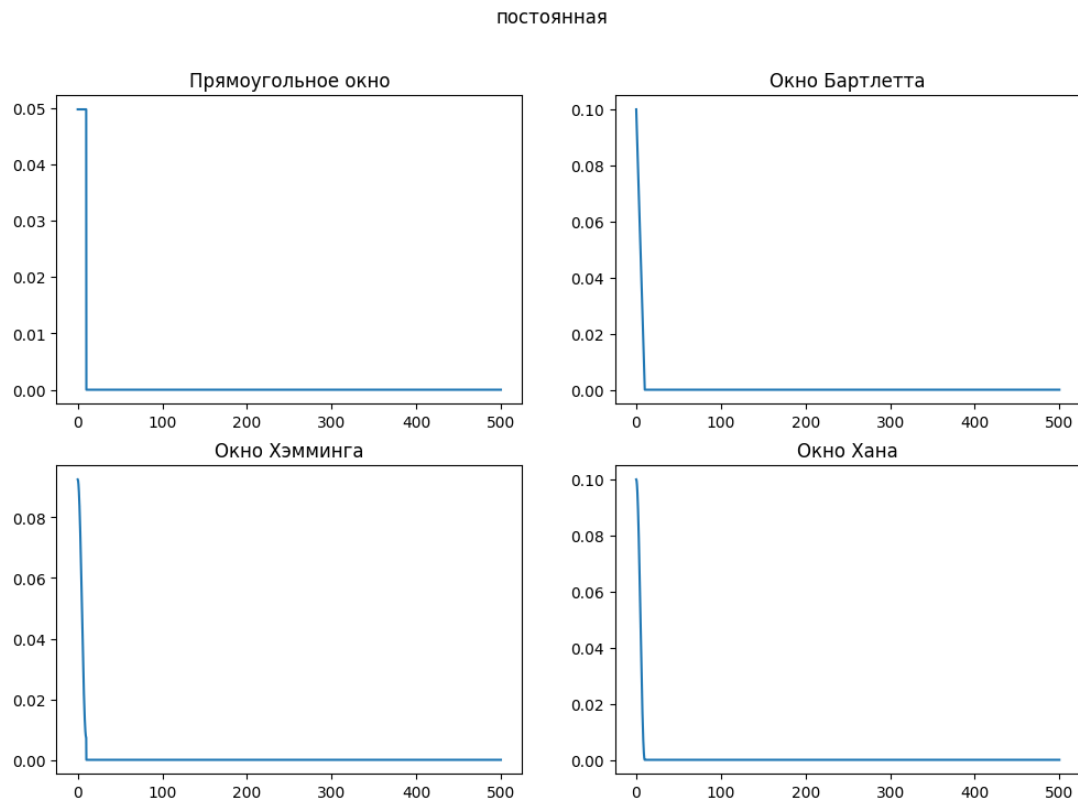


Рис. 2.8. Спектральная плотность мощности, пример 1

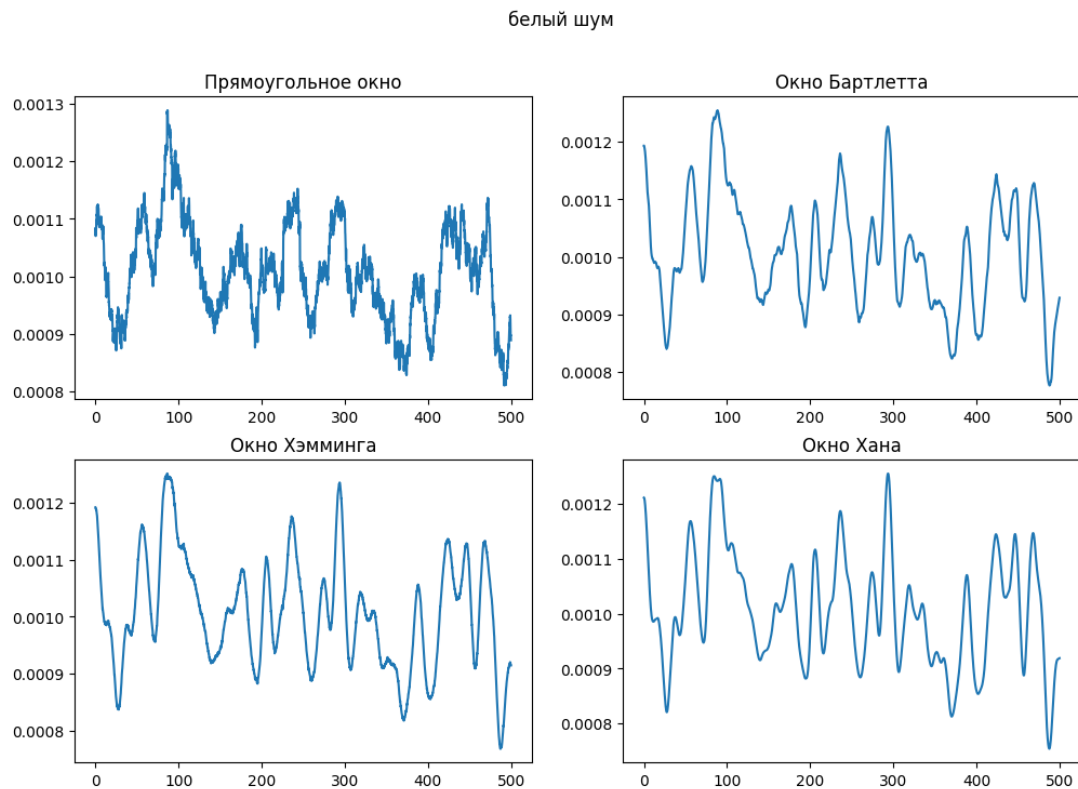


Рис. 2.9. Спектральная плотность мощности, пример 2

несколько гармоник

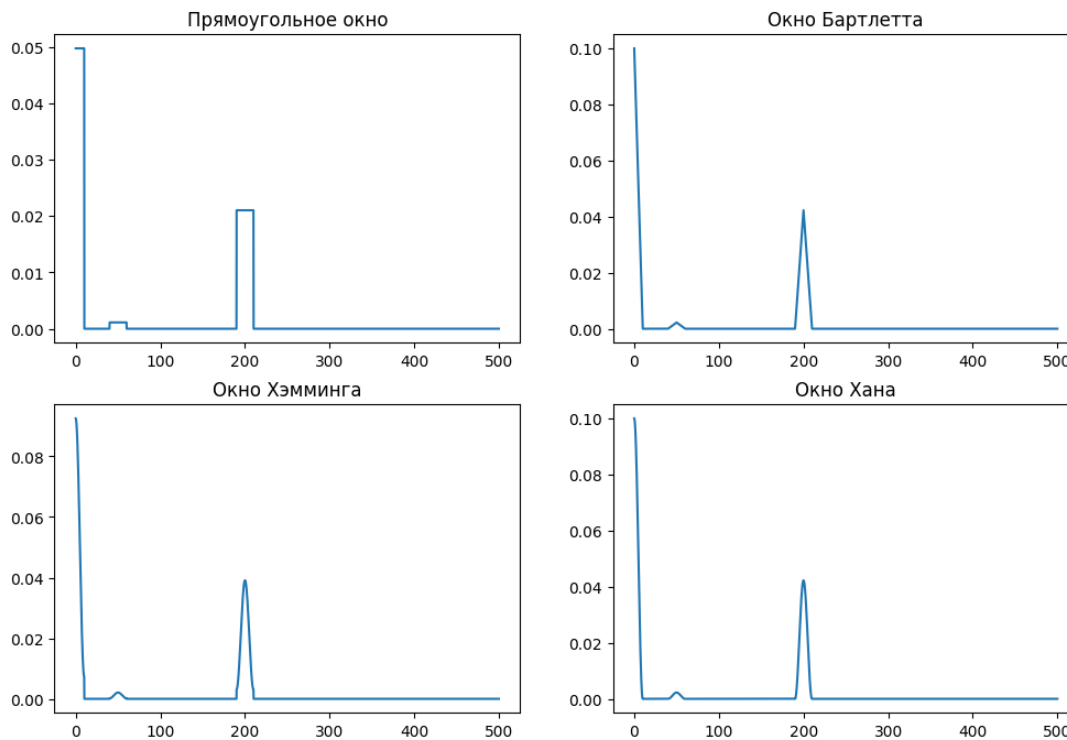


Рис. 2.10. Спектральная плотность мощности, пример 3

линейно
коррелированные отсчёты

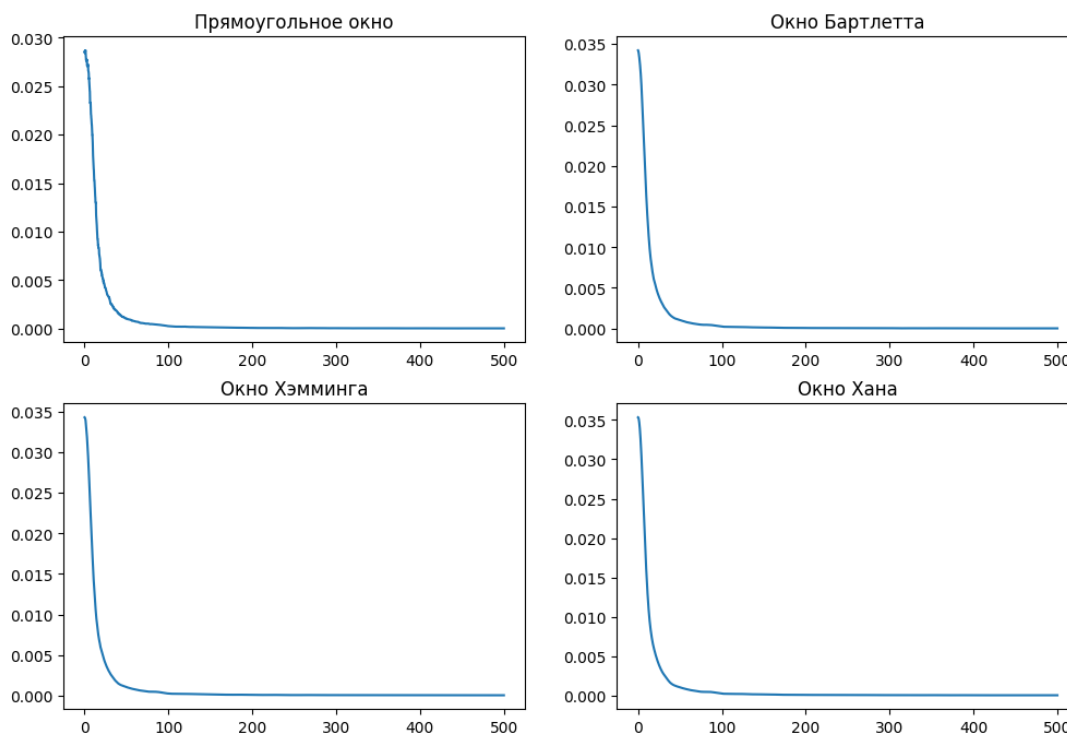


Рис. 2.11. Спектральная плотность мощности, пример 4

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.

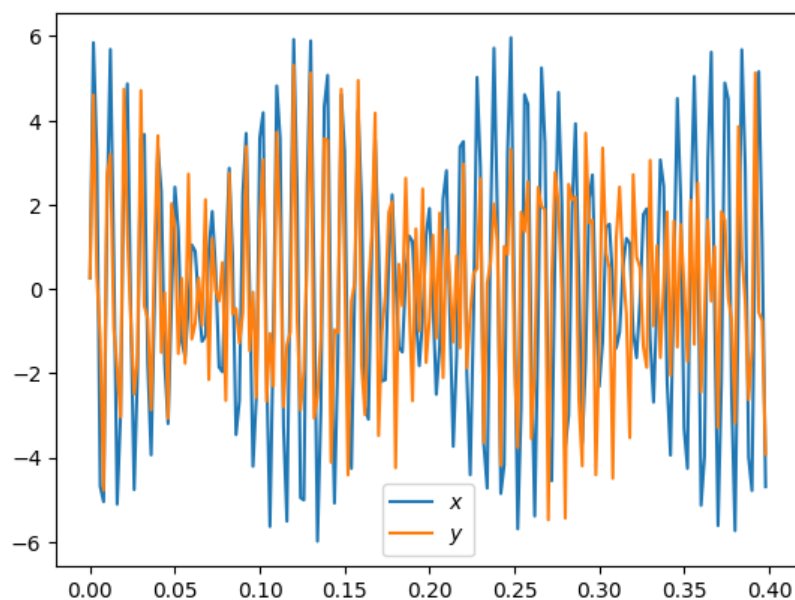


Рис. 2.12. График сигналов x , y .

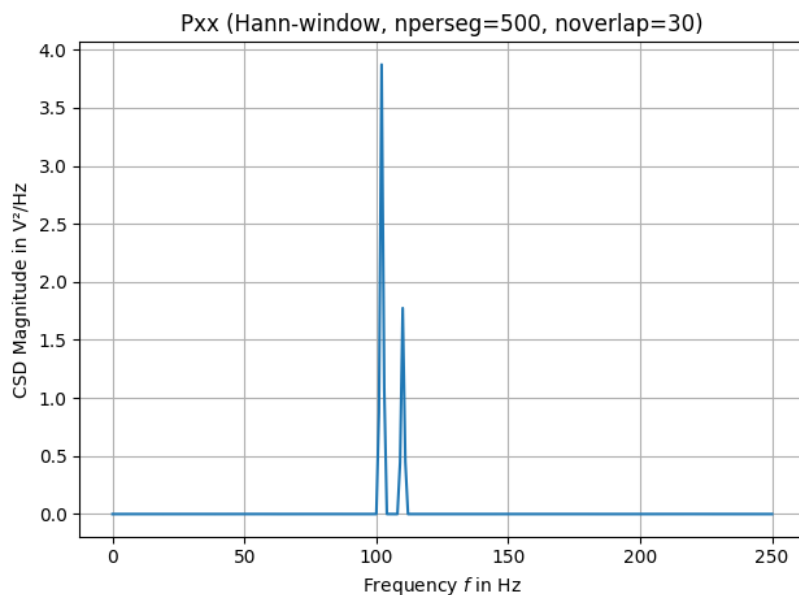


Рис. 2.13. Спектральная плотность мощности

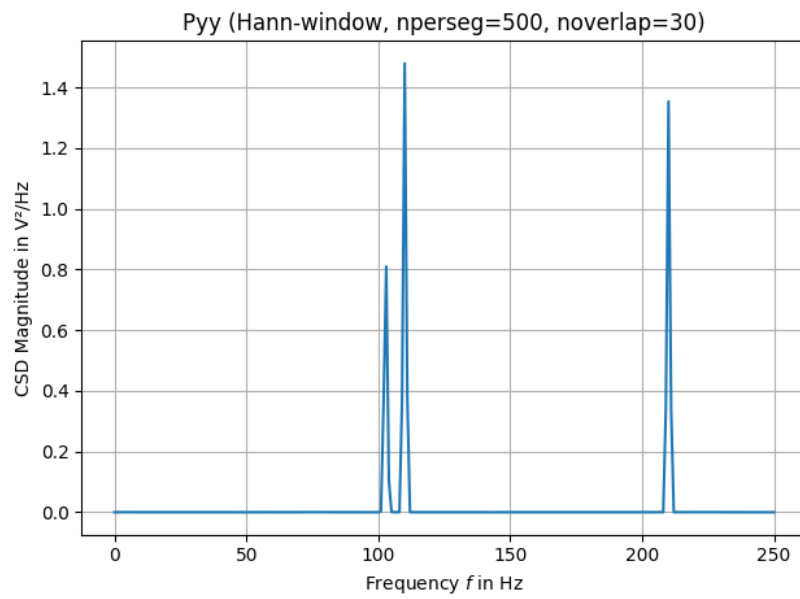


Рис. 2.14. Спектральная плотность мощности

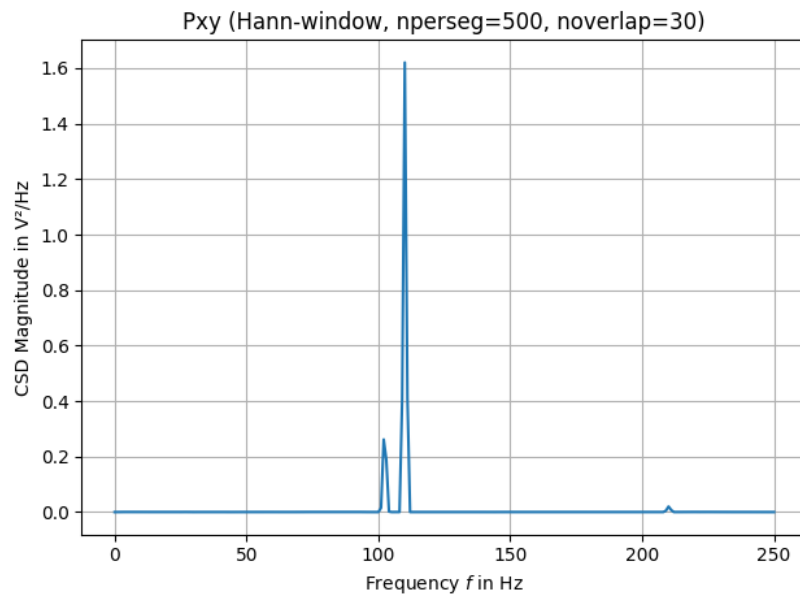


Рис. 2.15. Спектральная плотность мощности

Приложение А. Листинги

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

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  from matplotlib.patches import Ellipse
4  import matplotlib.transforms as transforms
5  import scipy.stats as stats
6  from scipy import signal
7  from collections import namedtuple
8
9  N = 500
10 T = 1 # seconds
11 delta = T / N
12 a0 = 0.4
13 a1 = 0.015
14 a2 = 0.63
15
16 np.random.seed(42)
17 e1 = np.random.standard_normal(N + 1)
18 e2 = np.random.standard_normal(N + 1)
19
20 k = np.array(range(N + 1))
21 t = k * delta
22 y1 = a0 * np.sin(a1 * 2 * np.pi * t) + a2 * e1
23
24 y3 = np.exp(0.1 * t) + 2 * e2

```

Листинг 3. Функция построения эллипса рассеяния

```

1  def confidence_ellipse(x, y, ax, p=0.99, facecolor='none', **kwargs):
2      if x.size != y.size:
3          raise ValueError("x and y must be the same size")
4
5      cov = np.cov(x, y)
6      pearson = cov[0, 1]/np.sqrt(cov[0, 0] * cov[1, 1])
7      # Используется частный случай получения собственных значений
8      ell_radius_x = np.sqrt(1 + pearson)
9      ell_radius_y = np.sqrt(1 - pearson)
10     ellipse = Ellipse((0, 0), width=ell_radius_x * 2, height=ell_radius_y * 2,
11                       facecolor=facecolor, **kwargs)
12
13     # Масштабирование в соответствии с данным pvalue в предположении (x, y) ~ N
14     n_std_for_quantile = stats.norm.ppf((1 + p) / 2)
15
16     scale_x = np.sqrt(cov[0, 0]) * n_std_for_quantile
17     mean_x = np.mean(x)
18
19     scale_y = np.sqrt(cov[1, 1]) * n_std_for_quantile
20     mean_y = np.mean(y)
21
22     # A, такой что xAx^T = 1 описывает эллипс рассеяния до масштабирования и поворота
23     A = np.array([[ 1/((ell_radius_x) ** 2), 0], [0, 1/((ell_radius_y) ** 2)]])

```

```

24     scale = np.array([[1/(scale_x ** 2), 0], [0, 1/ (scale_y ** 2)]])
25
26     angle = np.pi / 4
27     rotation = np.array([ [np.cos(angle), -np.sin(angle)], [np.sin(angle), np.cos(angle)] ])
28
29     A = rotation @ scale @ A @ rotation.T
30     mu = np.array([mean_x, mean_y])
31
32     transf = transforms.Affine2D() \
33         .rotate_deg(45) \
34         .scale(scale_x, scale_y) \
35         .translate(mean_x, mean_y)
36
37     ellipse.set_transform(transf + ax.transData)
38     return ax.add_patch(ellipse), A, mu

```

Листинг 4. Построение эллипса рассеяния

```

1  def replace_with_mean(arr: np.array, indices):
2      result = np.copy(arr)
3      for i in indices:
4          if i == 0 or (i == result.shape[0] - 1):
5              continue
6
7          result[i] = 0.5 * (result [i - 1] + result[i + 1])
8      return result
9
10 def ellipse(y, title: str):
11     differences = y[1:] - y[:-1]
12     forward_diff = differences[1:]
13     backward_diff = differences[:-1]
14
15     fig, ax = plt.subplots()
16
17     _, A, mu = confidence_ellipse(forward_diff, backward_diff, ax, edgecolor='firebrick')
18
19     ell_x = np.c_[forward_diff, backward_diff]
20     diagonal_values = np.diag((ell_x - mu) @ A @ (ell_x - mu).T) # вектор длины n
21     signes = np.sign(forward_diff * backward_diff)
22     anomalies = ell_x[(diagonal_values > 1) & (signes < 1)]
23     anomalies_indices = np.where((diagonal_values > 1) & (signes < 1))[0]
24     ell_x_clean = replace_with_mean(ell_x, anomalies_indices)
25
26     # Добавление на график ell_x_clean, anomalies
27
28     ellipse(y1, '$y_1$')
29     ellipse(y3, '$y_3$')

```

Листинг 5. Оценка стационарности рядов

```

1  def estimate_m(arr: np.array) -> float:
2      N = arr.shape[0]
3      return (arr @ np.ones(N)) / N
4

```

```

5  def estimate_m_var(arr: np.array):
6      N = arr.shape[0]
7      m = estimate_m(arr)
8      return m, ((arr - m).T @ (arr - m)) / (N - 1)
9
10 def series_test(arr: np.array):
11     N = arr.shape[0]
12     arr = arr[1:] - arr[:-1]
13
14     v, t_list = 1, []
15     counter, elem_sign = 1, np.sign(arr[0])
16     for x in np.nditer(arr[1:]):
17         if elem_sign * x >= 0:
18             counter += 1
19         else:
20             v += 1
21             t_list.append(counter)
22             counter, elem_sign = 1, np.sign(x)
23     t_list.append(counter)
24     t = np.max(t_list)
25     if N <= 6:
26         tmax = 5
27     elif N <= 154 and N > 6:
28         tmax = 6
29     else:
30         tmax = 7
31
32     no_trend = (v > ((2 * (N - 1)) / 3 - 1.96 * np.sqrt((16 * N - 29) / 90))) and t < tmax
33     return no_trend, v, t
34
35 def trend_test(y, title):
36     mus = []
37     vars = []
38     for i in range(10):
39         arr = y[50 * i: 50 * (i + 1)]
40         mu, var = estimate_m_var(arr)
41         mus.append(mu)
42         vars.append(var)
43
44
45     no_trend, _, _ = series_test(np.array(mus))
46     print(f'По результату критерий серий "Тренд в m({title.replace("$", "")}) отсутствует -
47     ↪ {no_trend}')
48     no_trend, _, _ = series_test(np.array(vars))
49     print(f'По результату критерий серий "Тренд в var({title.replace("$", "")}) отсутствует -
50     ↪ {no_trend}')
51     plt.plot(mus, label="$\mu_{ " + title + " }$")
52     plt.plot(vars, label="$\sigma_{ " + title + " }^2$")
53     plt.legend()
54     plt.title(f'Изменение параметров реализации {title} со временем')
55     plt.xlabel('Номер блока')
56     plt.show()
57
58 def corr_spectrum(y, title):
59     plt.acorr(y, maxlags=30, normed=True)

```

```

58     plt.title(f'Автокорреляционная функция {title}')
59     plt.xlabel('$\\tau$')
60     plt.show()
61
62     f, Pxx = signal.periodogram(y.flatten(), scaling='density', fs=1/delta)
63     plt.plot(f, Pxx)
64     plt.xlabel('Hz')
65     plt.ylabel('$V^2/Hz$')
66     plt.title(f'Спектральная мощность {title}')
67     plt.show()
68
69     trend_test(y1, 'y_1')
70     corr_spectrum(y1, '$y_1$')
71
72     trend_test(y3, 'y_3')
73     corr_spectrum(y3, '$y_3$')
74
75     def print_characteristics(arr: np.array):
76         mu, var = estimate_m_var(arr)
77         skew = stats.skew(arr)
78         kurtosis = stats.kurtosis(arr)
79         kstest = stats.kstest(arr, 'norm', alternative='two-sided')
80
81     print(f'Математическое ожидание - {mu:.3f}')
82     print(f'Дисперсия - {var:.3f}')
83     print(f'Асимметрия - {skew:.3f}')
84     print(f'Экцесс - {kurtosis:.3f}')
85     print(f'Тест на нормальность Колмогорова-Смирнова - pvalue={kstest.pvalue:.1e}')
86
87     print('Характеристики y1')
88     print_characteristics(y1)
89
90     print('\\nХарактеристики y3')
91     print_characteristics(y3)

```

Листинг 6. Построение корреляционных функций для разных сигналов

```

1  def correlation_function(x: np.array, y: np.array, lags_count: int):
2      if x.shape[0] != y.shape[0]:
3          raise ValueError('x and y must be the same size')
4      N = x.shape[0]
5
6      if lags_count > N:
7          raise ValueError('lags_count should be less than x, y size')
8
9      mx = estimate_m(x)
10     my = estimate_m(y)
11     Rxy = []
12     for k in range(lags_count):
13         Rxy.append((x[:N-k] - mx).T @ (y[k:] - my) / N)
14
15     return Rxy
16
17 def plot_corr_function(x: np.array, y: np.array, lags_count: int, title: str):
18     is_autocorr = np.all(x == y)

```

```

19
20     if is_autocorr:
21         Rxy = correlation_function(x, x, lags_count)
22     else:
23         Rxy = correlation_function(x, y, lags_count)
24
25     fig, axs = plt.subplots(1, 2, figsize=(12, 5))
26     axs[0].set_title('Сигналы')
27     axs[0].plot(x[:100], label='x')
28     if not is_autocorr:
29         axs[0].plot(y[:100], label='y')
30     axs[0].legend()
31
32     axs[1].plot(Rxy)
33     axs[1].set_title(f'{"Автокорреляционная" if is_autocorr else "Корреляционная"} функция
34     ↪ {title}')
35     axs[1].set_xlabel('$k \Delta$')
36     axs[1].hlines(0.05, 0, lags_count, color='k', linestyle='dashed')
37     plt.show()
38
39 def signal_with_garmonics(N, delta = 1e-3):
40     k = np.array(range(N))
41     return 1 * np.ones(N) + 1.3 * np.sin(200 * 2 * np.pi * k * delta + 0.4) + 0.3 * np.sin(50
42     ↪ * 2 * np.pi * k * delta + 0.93)
43
44 def signal_linear_correlated(N, alpha=0.95):
45     w = np.random.standard_normal(N)
46
47     x = np.zeros(N)
48     x[0] = w[0] / np.sqrt(1 - alpha**2)
49     for n in range(1, N):
50         x[n] = alpha * x[n-1] + np.sqrt(1 - alpha**2) * w[n]
51     return x
52
53 Realization = namedtuple('Realization', 'generate_x generate_y lags_to_plot')
54 realizations = {
55     'постоянная': Realization(np.ones, None, 10),
56     'белый шум': Realization(np.random.standard_normal, None, 10),
57     'несколько гармоник': Realization(signal_with_garmonics, None, 100),
58     'линейно \nкоррелированные отсчёты': Realization(signal_linear_correlated, None, 100),
59     'белый шум \nни несколько гармоник': Realization(np.random.standard_normal,
60     ↪ signal_with_garmonics, 100),
61     'несколько гармоник \nни коррелированные отсчёты': Realization(signal_with_garmonics,
62     ↪ signal_linear_correlated, 100),
63     'два сигнала \nс коррелированными отсчётами': Realization(signal_linear_correlated,
64     ↪ signal_linear_correlated, 100)
65 }
66
67 N = 100_000
68
69 for title, realization in realizations.items():
70     x = realization.generate_x(N)
71     if realization.generate_y is None:
72         plot_corr_function(x, x, realization.lags_to_plot, title)
73     else:

```

```

69         y = realization.generate_y(N)
70         plot_corr_function(x, y, realization.lags_to_plot, title)

```

Листинг 7. Построение спектральных плотностей мощности для разных сигналов

```

1  def periodogram(x: np.array, delta):
2      N = x.shape[0]
3      I = []
4
5      i = np.array(range(N))
6      for k in range(int(N)):
7          alpha = x.T @ np.cos( (2 * np.pi * k * i) / N )
8          beta = x.T @ np.sin( (2 * np.pi * k * i) / N )
9          I.append(delta / N * (alpha ** 2 + beta ** 2))
10
11     return I
12
13 # spectral density
14 def esimate_spe(x: np.array, delta, window):
15     N = x.shape[0]
16     Sxx = []
17     I = np.array(periodogram(x, delta))
18
19     j = np.arange(-N / 2, N / 2, 1)
20     for k in range(int(N / 2)):
21         indices = np.abs( np.astype((k - j), np.int16) )
22         Sxx.append( I[indices].T @ window(j) )
23
24     return Sxx
25
26 def rect_window(j, M = 100):
27     w = np.where(np.abs(j) <= M, 1/(2*M + 1), 0)
28     return w
29
30 def bartlett_window(j, M = 100):
31     w = np.where(np.abs(j) <= M, (1 - np.abs(j) / M) / M, 0)
32     return w
33
34 def hamming_window(j, M = 100):
35     w = np.where(np.abs(j) <= M, ( 0.54 + 0.46 * np.cos(np.pi * j / M) ) / (1.08 * M + 0.08),
36                  ↪ 0)
37     return w
38
39 def han_window(j, M = 100):
40     w = np.where(np.abs(j) <= M, (1 + np.cos(np.pi * j / M)) / (2 * M), 0)
41     return w
42
43 realizations = {
44     'постоянная': Realization(np.ones, None, 10),
45     'белый шум': Realization(np.random.standard_normal, None, 10),
46     'несколько гармоник': Realization(signal_with_garmonics, None, 100),
47     'линейно \nкоррелированные отсчёты': Realization(signal_linear_correlated, None, 100),
48 }
49
50 windows = {

```

```

50     'Прямоугольное окно': rect_window,
51     'Окно Бартлетта': bartlett_window,
52     'Окно Хэмминга': hamming_window,
53     'Окно Хана': han_window,
54 }
55
56 N = 10000
57 delta = 1e-3
58 for title, realization in realizations.items():
59     x = realization.generate_x(N)
60
61     fig, axs = plt.subplots(2, 2, figsize=(12, 8))
62     for i, window_name in enumerate(windows):
63         Sxx = estimate_spe(x, delta, windows[window_name])
64         f = np.arange(len(Sxx)) / (delta * N)
65
66         plot_index = int(i / 2), i % 2
67         axs[plot_index].set_title(window_name)
68         axs[plot_index].plot(f, Sxx)
69
70     plt.suptitle(title)
71     plt.show()

```

Листинг 8. Построение спектральных плотностей мощности для разных сигналов

```

1  N = 1500
2  T = 3 # seconds
3  delta = T / N
4  fs = 1 / delta
5
6  k = np.array(range(N + 1))
7  t = k * delta
8
9  e = np.random.standard_normal(N + 1)
10 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
11 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
12 plt.plot(t[:200], x[:200], label='$x$')
13 plt.plot(t[:200], y[:200], label='$y$')
14 plt.legend()
15 plt.show()
16
17 # Compute and plot the magnitude of the cross spectral density:
18 nperseg, noverlap, win = 500, 30, 'hann'
19
20
21 def plot_CSD(csd, csd_name: str):
22     fig0, ax0 = plt.subplots(tight_layout=True)
23     ax0.set_title(f"{csd_name} ({win.title()}-window, {nperseg=}, {noverlap=})")
24     ax0.set_xlabel="Frequency $f$ in Hz", ylabel="CSD Magnitude in V2/Hz"
25     ax0.plot(f, np.abs(csd))
26     ax0.grid()
27     plt.show()
28

```

```
29 f, Pxx = signal.csd(x, x, fs, win, nperseg, noverlap)
30 plot_CSD(Pxx, 'Pxx')
31 f, Pyy = signal.csd(y, y, fs, win, nperseg, noverlap)
32 plot_CSD(Pyy, 'Py')
33 f, Pxy = signal.csd(x, y, fs, win, nperseg, noverlap)
34 plot_CSD(Pxy, 'Pxy')
```
