



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ МК «Машиностроительный»

КАФЕДРА МК10 «Высшая математика и физика»

Домашняя работа №1

«Одномерные вейвлет-преобразования»

ДИСЦИПЛИНА: «Вейвлет-преобразование сигналов»

Выполнил: студент гр. ИУК4-11М _____ (_____)
(подпись) (Ф.И.О.)

Проверил: _____ (_____)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Задание 1

Функция $f(t) = 4(t-7)^2 + t$ задана на промежутке $[0, 14]$.

Используя пакет SymPy, найдём разложение функции в ряд Фурье в символьном виде, упростим разложение.

```
fourier = sp.fourier_series(f, (t, *t_range))  
fourier
```

$$-\frac{14 \sin\left(\frac{\pi t}{7}\right)}{\pi} - \frac{7 \sin\left(\frac{2\pi t}{7}\right)}{\pi} + \frac{784 \cos\left(\frac{\pi t}{7}\right)}{\pi^2} + \frac{196 \cos\left(\frac{2\pi t}{7}\right)}{\pi^2} + \frac{217}{3} + \dots$$

Рисунок 1.1 – Разложение функции в ряд Фурье

```
sp.simplify(fourier).truncate(5)
```

$$-\frac{14 \sin\left(\frac{\pi t}{7}\right)}{\pi} - \frac{7 \sin\left(\frac{2\pi t}{7}\right)}{\pi} - \frac{14 \sin\left(\frac{3\pi t}{7}\right)}{3\pi} - \frac{7 \sin\left(\frac{4\pi t}{7}\right)}{2\pi} + \frac{784 \cos\left(\frac{\pi t}{7}\right)}{\pi^2} + \frac{196 \cos\left(\frac{2\pi t}{7}\right)}{\pi^2} + \frac{784 \cos\left(\frac{3\pi t}{7}\right)}{9\pi^2} + \frac{49 \cos\left(\frac{4\pi t}{7}\right)}{\pi^2} + \frac{217}{3}$$

Рисунок 1.2 – Частичная сумма s_4

Построим графики функции и частичных сумм $(f(t), S_0(t)); (f(t), S_1(t)); (f(t), S_2(t)); (f(t), S_{10}(t))$

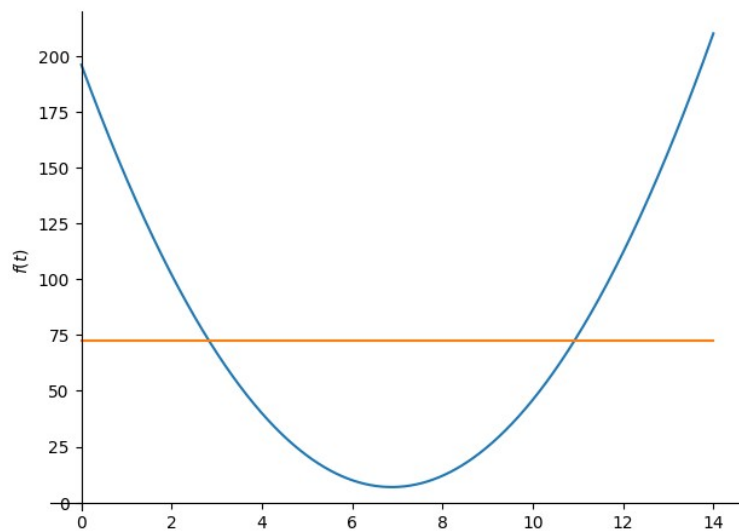


Рисунок 1.3 – График функции и частичная сумма s_0

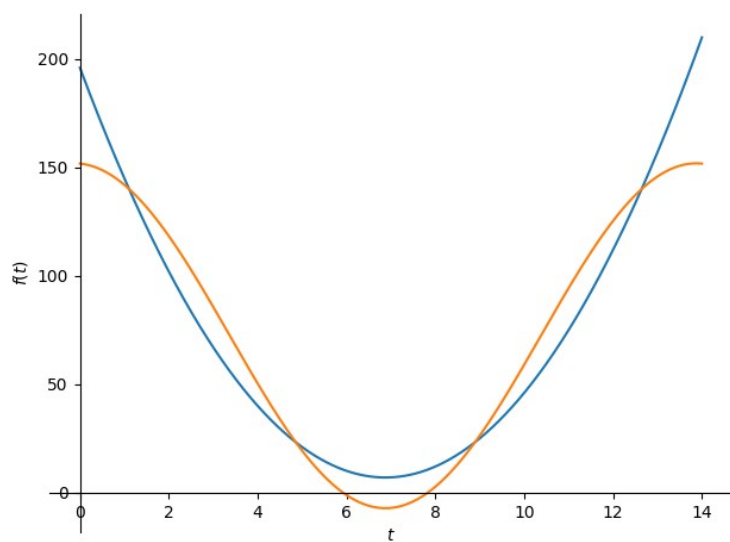


Рисунок 1.4 – График функции и частичная сумма s_1

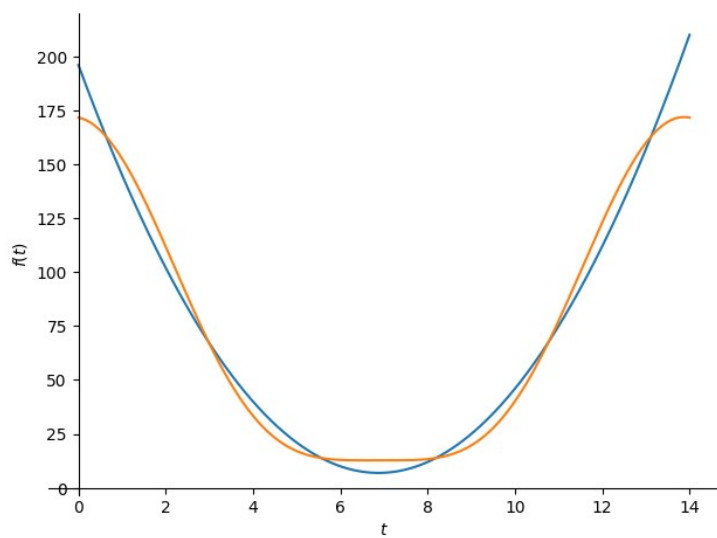


Рисунок 1.5 – График функции и частичная сумма s_2

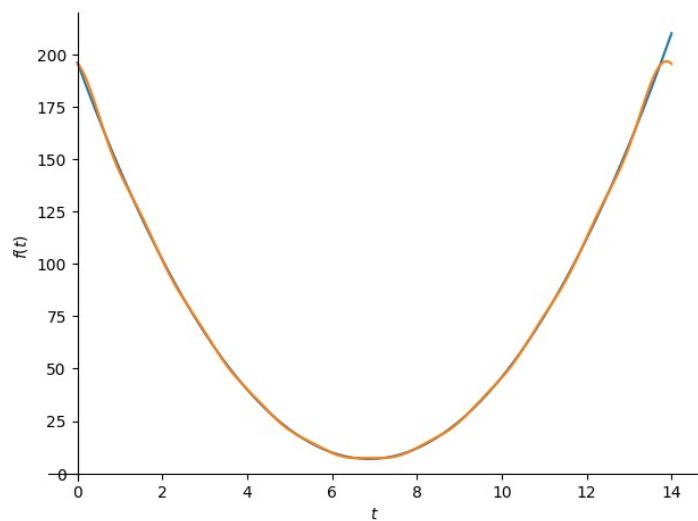


Рисунок 1.6 – График функции и частичная сумма s_{10}

Найдём максимальную погрешность приближения функции $f(t)$ частичной суммой $S_{50}(t)$.

```
lambdified = sp.lambdify(t, sp.Abs(f - fourier.truncate(51)).evalf())
np.max(lambdified(np.arange(*t_range, 1e-3)))

np.float64(8.416341780039858)
```

Рисунок 1.7 – Максимальная погрешность приближения функции

Задание 2

Функция без случайной составляющей:

$$t(t+7)^{0.5} + \begin{cases} -8 & \text{for } t = \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} + \begin{cases} \cos(8t+14) & \text{for } t \leq \pi \\ \cos(4t+7) & \text{otherwise} \end{cases}$$

Рисунок 2.1 – Функция без случайной составляющей

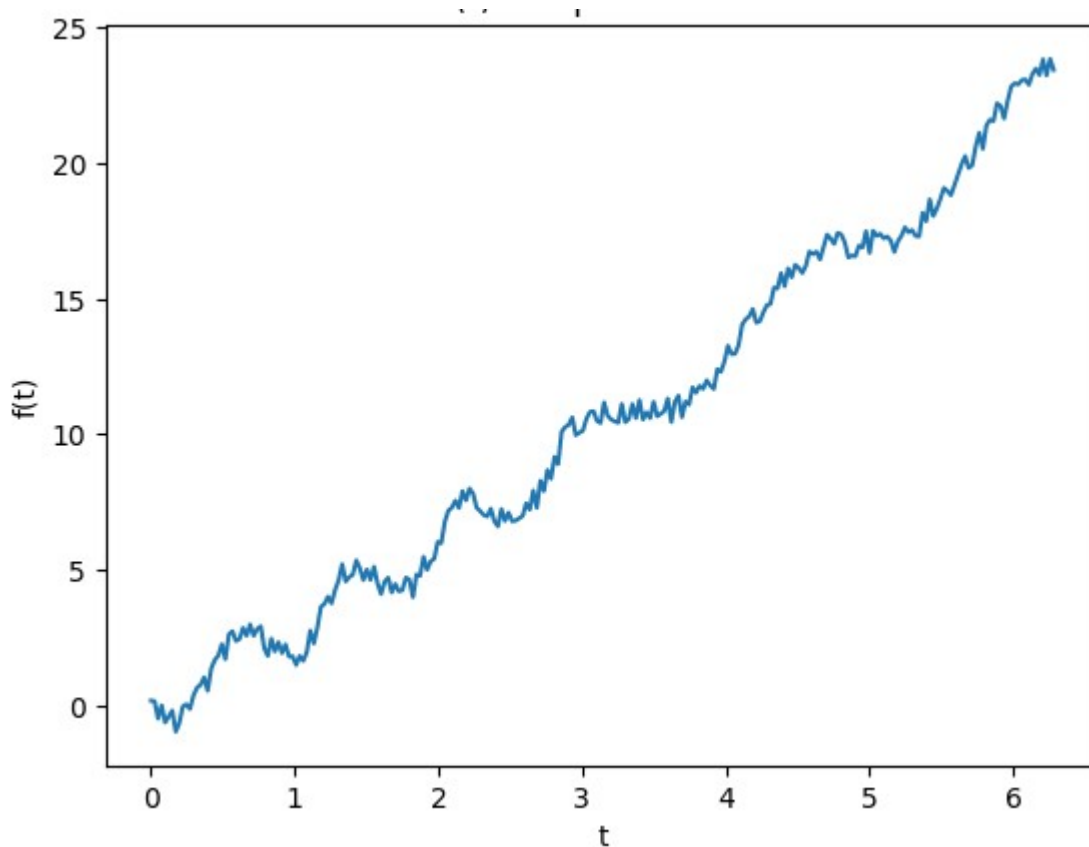


Рисунок 2.2 – Функция

```
frequencies = np.fft.fftfreq(num_points, (t_values[1] - t_values[0]))
plt.plot(frequencies[:num_points // 2], np.abs(fourier)[:num_points // 2], color='r')
plt.title("Амплитудный спектр сигнала f(t)")
plt.xlabel("Частота (Hz)")
plt.ylabel("Амплитуда")
```

```
Text(0, 0.5, 'Амплитуда')
```

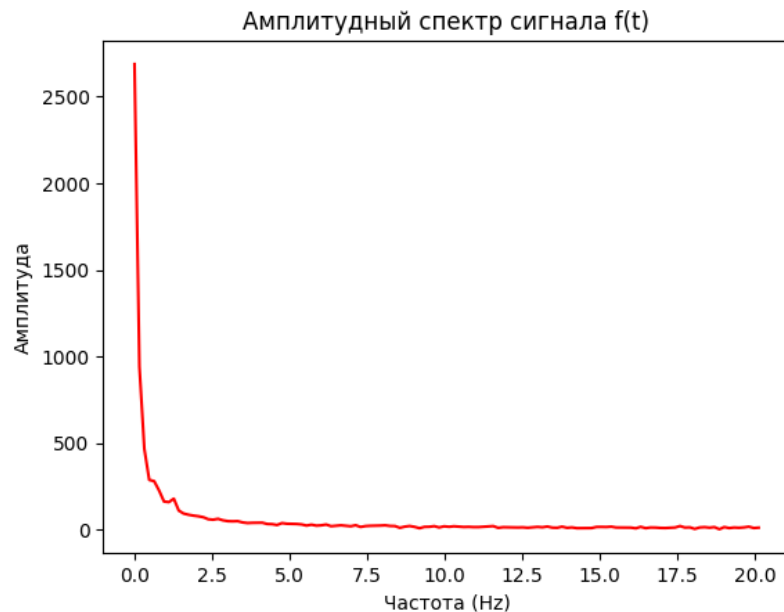


Рисунок 2.3 – График амплитудного спектра преобразования Фурье

Оставим 10% наибольших по абсолютной величине коэффициентов и обнулим остальные:

```
sorted_indices = np.argsort(np.abs(fourier))
num_to_keep = int(0.1 * num_points)
threshold_indices = sorted_indices[-num_to_keep:]

fourier_filtered = np.zeros_like(fourier)
fourier_filtered[threshold_indices] = fourier[threshold_indices]
```

Рисунок 2.4 – Обнуление 90% наименьших коэффициентов

Найдём разницу между исходным (без учета случайной составляющей) и восстановленным сигналами, используя L1-норму:

```
f_values_filtered = np.fft.ifft(fourier_filtered).real
```

```
f_no_noise = f_num(t_values)
difference = f_no_noise - f_values_filtered
L1_norm = np.sum(np.abs(difference))
L1_norm
```

```
np.float64(167.2929421565357)
```

Рисунок 2.5 – L1-норма разницы между исходным и восстановленным сигналами

Получаем, что $L1\text{-норма}=167.29$.

Построим графики исходного (без учета случайной составляющей) и восстановленного сигналов:

```
plt.plot(t_values, f_no_noise, label="Исходный сигнал без случайного шума", color='b')
plt.plot(t_values, f_values_filtered, label="Восстановленный сигнал (10% коэффициентов)", color='g', linestyle='--')
plt.title("Сравнение исходного и восстановленного сигналов")
plt.xlabel("t")
plt.ylabel("Амплитуда")
plt.legend()
```

<matplotlib.legend.Legend at 0x716222e4ed50>



Рисунок 2.6 – Графики исходного и восстановленного сигналов

Задание 3

Функция без случайной составляющей:

$$t(t+7)^{0.5} + \begin{cases} -8 & \text{for } t = \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} + \begin{cases} 2 \cos(8t + 14) & \text{for } t \leq \pi \\ 3 \cos(4t + 7) & \text{otherwise} \end{cases}$$

Рисунок 3.1 – Функция без случайной составляющей

Введём функции для построения графиков и устранения шумов:

```
def plot_result(t_values, f_no_noise, f_restored):
    plt.figure(figsize=(10, 6))

    plt.subplot(2, 1, 1)
    plt.plot(t_values, f_no_noise, label="Исходный сигнал без случайного шума", color='b')
    plt.title("Исходный сигнал")
    plt.xlabel("t")
    plt.ylabel("Амплитуда")
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.title("Восстановленный сигнал")
    plt.plot(t_values, f_restored, label="Восстановленный сигнал (10% коэффициентов)", color='g')
    plt.xlabel("t")
    plt.ylabel("Амплитуда")
    plt.legend()

    plt.tight_layout()
    plt.show()

def denoise(f_values: np.array, wavelet_type: str):
    coeffs = pywt.wavedec(f_values, wavelet_type)

    coeff_arr, coeff_slices = pywt.coeffs_to_array(coeffs)
    threshold = np.percentile(np.abs(coeff_arr), 90)
    coeff_arr[np.abs(coeff_arr) < threshold] = 0
    coeffs_thresholded = pywt.array_to_coeffs(coeff_arr, coeff_slices, output_format='wavedec')

    return pywt.waverec(coeffs_thresholded, wavelet_type)
```

Рисунок 3.2 – Утилитарные функции

Восстановим сигнал, используя вейвлет Добеши 2 и получим L1-норму для него.

```
f_restored = denoise(f_values, 'db2')

l1_norm = np.sum(np.abs(f_values - f_restored))
print("L1-норма разности:", l1_norm)

L1-норма разности: 95.66175096830705
```

Рисунок 3.3 – Значение L1-нормы вейвлета Добеши 2

Построим графики исходного и восстановленного сигналов:

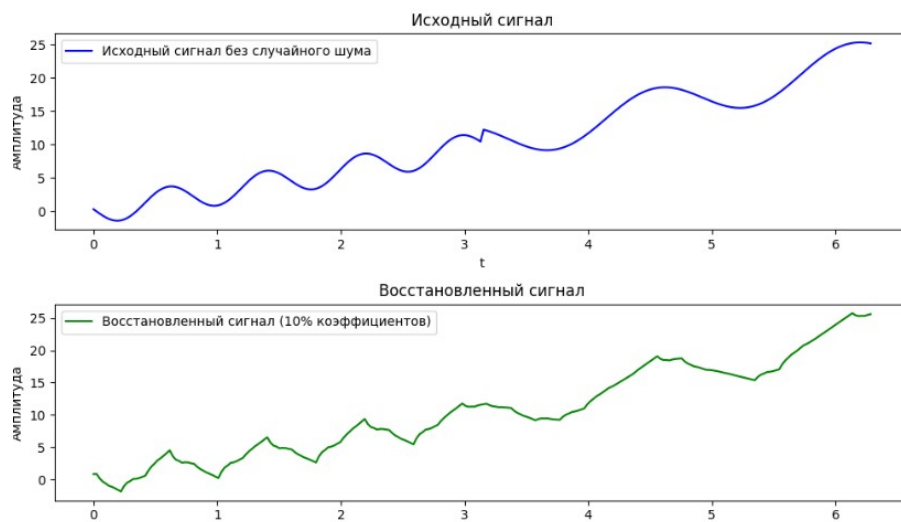


Рисунок 3.4 – Исходный сигнал и восстановленный сигнал вейвлет-преобразования Добеши 2

Восстановим сигнал, используя вейвлет Добеши 10 и получим L1-норму для него.

```
f_restored = denoise(f_values, 'db10')
```

```
l1_norm = np.sum(np.abs(f_values[:len(f_restored)] - f_restored))  
print("L1-норма разности:", l1_norm)
```

```
L1-норма разности: 72.50519895455824
```

```
f_no_noise = f_num(t_values)  
plot_result(t_values, f_no_noise, f_restored)
```

Рисунок 3.5 – Значение L1-нормы вейвлета Добеши 10

Построим графики исходного и восстановленного сигналов:

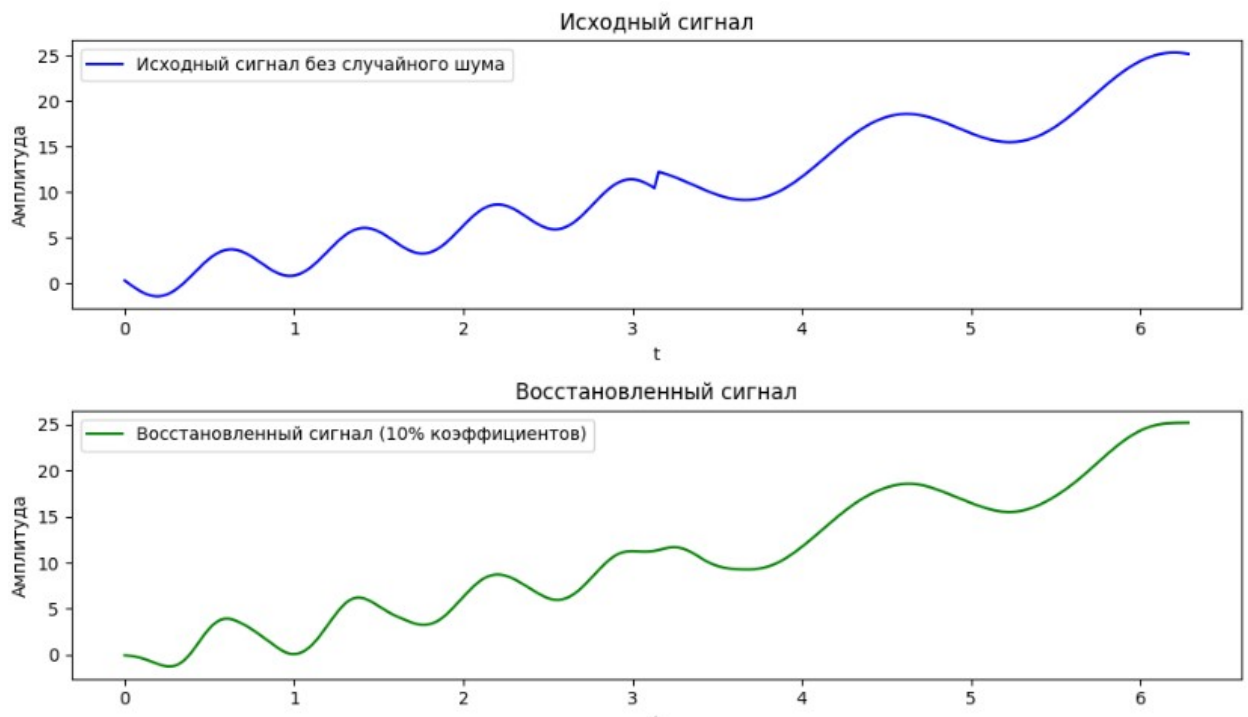


Рисунок 3.6 – Исходный сигнал и восстановленный сигнал вейвлет-преобразования Добеши 10

Восстановим сигнал, используя биортогональный вейвлет 2,2 и получим L1-норму для него.

```
f_restored = denoise(f_values, 'bior2.2')
```

```
l1_norm = np.sum(np.abs(f_values[:len(f_restored)] - f_restored))  
print("L1-норма разности:", l1_norm)
```

L1-норма разности: 69.27611724185996

Рисунок 3.7 – Значение L1-нормы биортогонального вейвлета 2,2

Построим графики исходного и восстановленного сигналов:

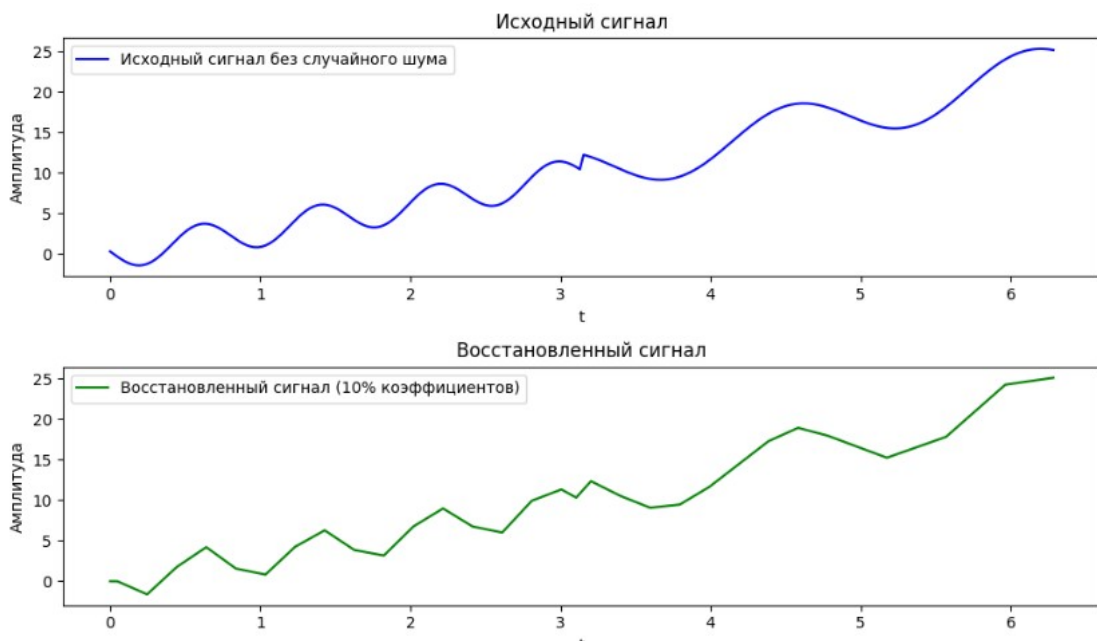


Рисунок 3.8 – Исходный сигнал и восстановленный сигнал биортогонального вейвлет-преобразования 2,2

Восстановим сигнал, используя симметричного вейвлет 4 и получим L1-норму для него.

```
f_restored = denoise(f_values, 'sym4')
```

```
l1_norm = np.sum(np.abs(f_values[:len(f_restored)] - f_restored))  
print("L1-норма разности:", l1_norm)
```

L1-норма разности: 73.42999539117159

Рисунок 3.9 – Значение L1-нормы симметричного вейвлета 4

Построим графики исходного и восстановленного сигналов:

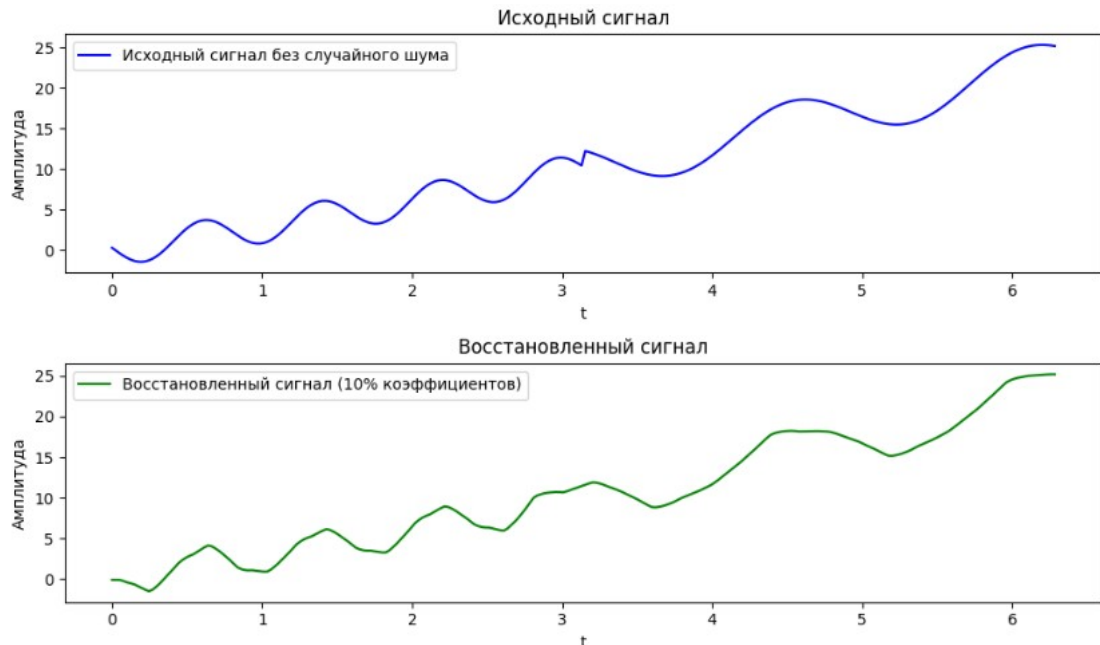


Рисунок 3.10 – Исходный сигнал и восстановленный сигнал симметричного вейвлет-преобразования 4

Таблица 1

Значения L1-нормы для вейвлет-преобразований

	Добеши 2	Добеши 10	Биортогональный 2,2	Симметричный 4
L1-норма	95.66	72.50	69.27	73.42

Биортогональный 2.2 имеет наименьшую L1-норму (69.27), что свидетельствует о том, что этот вейвлет наилучшим образом подходит для восстановления сигнала в данном случае. Добеши 10 также демонстрирует хороший результат, но уступает биортогональному вейвлету. Добеши 2 оказался наименее эффективным с точки зрения восстановления сигнала, так как имеет наибольшую L1-норму.

Задание 4

Функция без случайной составляющей:

$$t(t+7)^{0.5} + \begin{cases} 2 \cos(8t+14) & \text{for } t \leq \pi \\ 3 \cos(4t+7) & \text{otherwise} \end{cases}$$

Рисунок 4.1 – Функция без случайной составляющей

Построим графики самого исходного сигнала и его же без шумов:

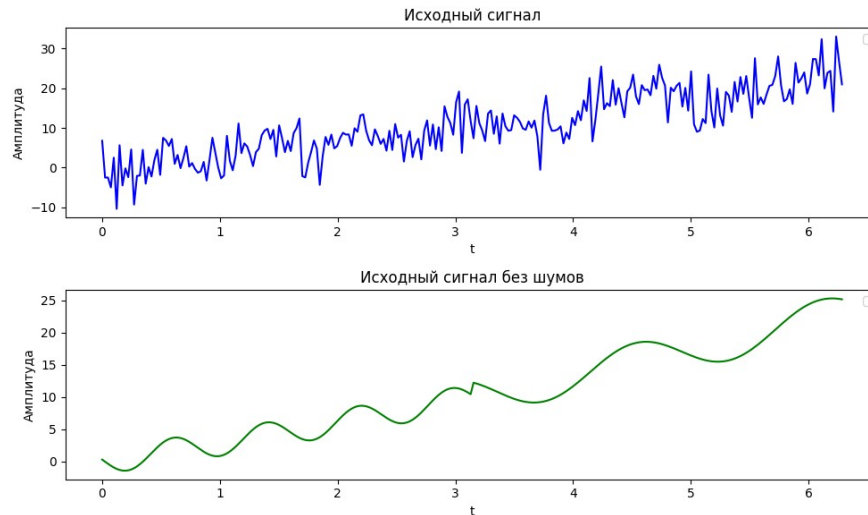


Рисунок 4.2 – График самого исходного сигнала и его же без шумов

Введём функцию для подсчёта соотношения сигнала к шумам

$$ОСШ = 10 * \log_{10} \frac{A_{ср. сигнала}^2}{A_{ср. шума}^2}:$$

```
def calculate_snr(signal: np.array, noise):
    power_signal = np.mean(signal ** 2)
    power_noise = np.mean(noise ** 2)
    snr = 10 * np.log10(power_signal / power_noise)
    return snr
```

Рисунок 4.3 – Функция подсчёта ОСШ

Построим график восстановленного сигнала для вейвлета Добеши 2:

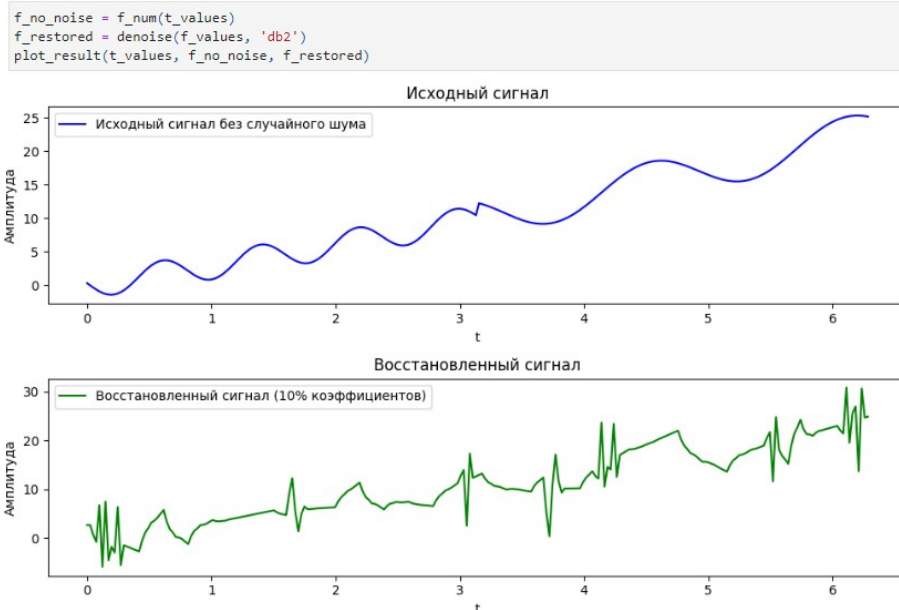


Рисунок 4.4 – Исходный сигнал и восстановленный сигнал вейвлет-преобразования Добеши 2

Вычислим значение ОСШ для исходного сигнала и вейвлета Добеши 2:

```
calculate_snr(f_values, r_values)

np.float64(10.809635880458037)

calculate_snr(f_restored, f_no_noise - f_restored)

np.float64(14.073055816432916)
```

Рисунок 4.5 – Значение ОСШ вейвлета Добеши 2 и исходного сигнала

Построим график восстановленного сигнала для вейвлета Добеши 10:

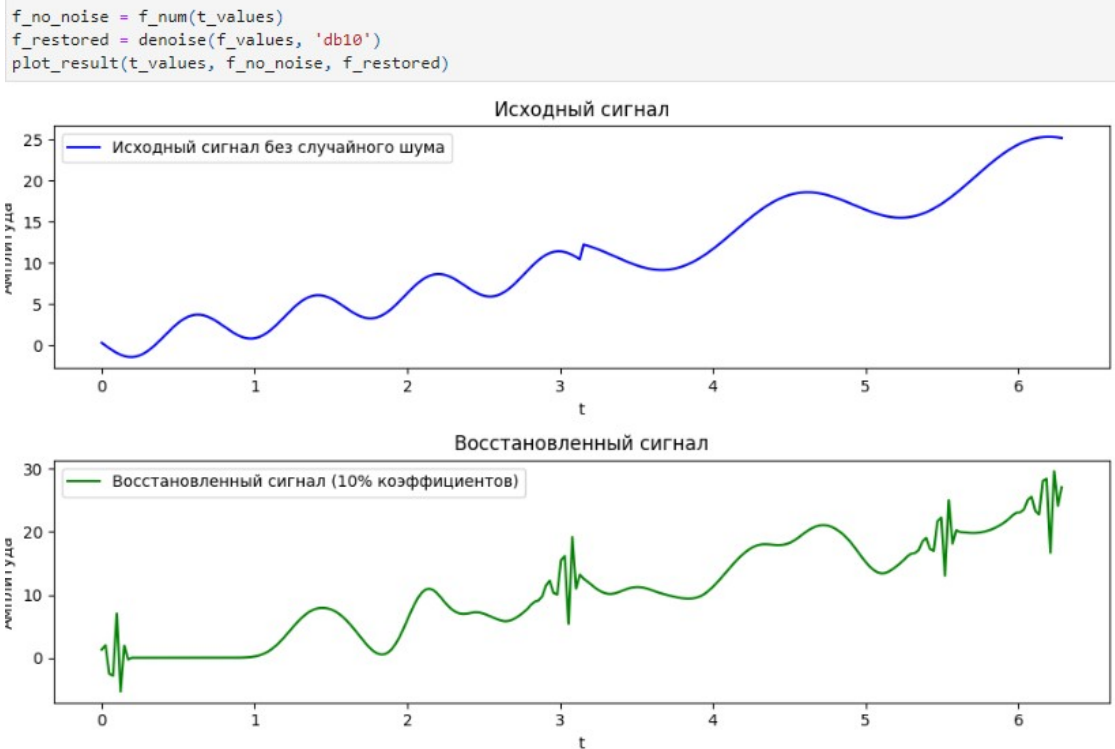


Рисунок 4.6 – Исходный сигнал и восстановленный сигнал вейвлет-преобразования Добеши 10

Вычислим значение ОСШ для вейвлета Добеши 10:

```
calculate_snr(f_restored, f_no_noise - f_restored)

np.float64(16.152874218066692)
```

Рисунок 4.7 – Значение ОСШ вейвлета Добеши 10

Построим график восстановленного сигнала для биортогонального вейвлета 2,2:

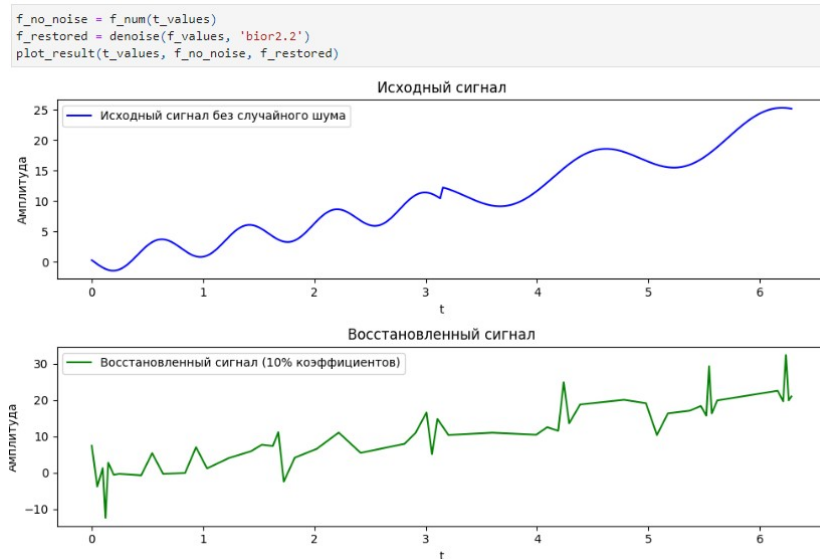


Рисунок 4.8 – Исходный сигнал и восстановленный сигнал биортогонального вейвлет-преобразования 2,2

Вычислим значение ОСШ для биортогонального вейвлета 2,2:

```
calculate_snr(f_restored, f_no_noise - f_restored)

np.float64(14.570048257135104)
```

Рисунок 4.9 – Значение ОСШ биортогонального вейвлета 2,2

Построим график восстановленного сигнала для симметричного вейвлета 4:

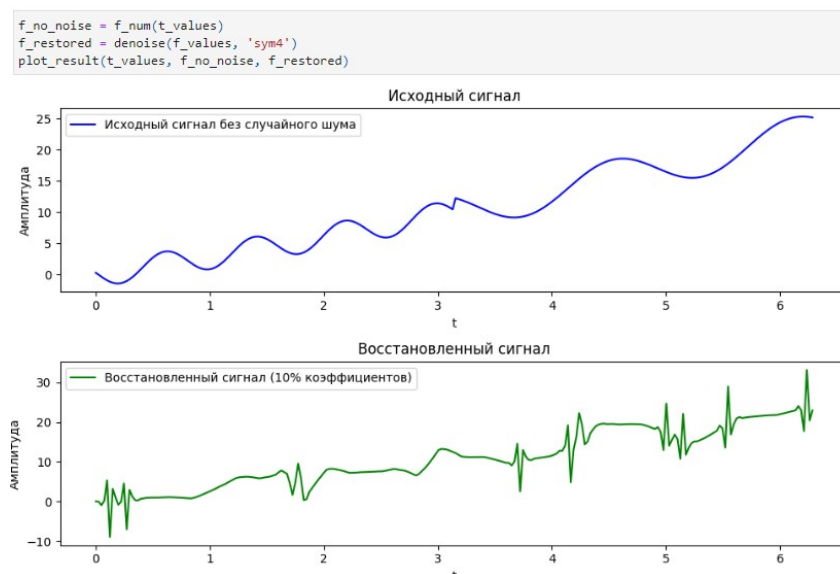


Рисунок 4.10 – Исходный сигнал и восстановленный сигнал симметричного вейвлета 4

Вычислим значение ОСШ для симметричного вейвлета 4:

```
calculate_snr(f_restored, f_no_noise - f_restored)  
  
np.float64(14.737411907816341)
```

Рисунок 4.11 – Значение ОСШ симметричного вейвлета 4

Таблица 2

Значения ОСШ для исходного и восстановленных сигналов

	Исходный	Добеши 2	Добеши 10	Биортогональный 2,2	Симметричный 4
ОСШ, дБ	10.81	14.07	16.15	14.57	14.73

На основе данных таблицы наилучший вейвлет для устранения шума — Добеши 10. Он обеспечивает наибольшее значение ОСШ (16.15 дБ), что указывает на лучшее подавление шума и сохранение полезного сигнала.