

# Лабораторная работа №4

Липатов Данила Вячеславович  
МСМТ243

## Постановка задачи:

- 1) Необходимо выполнить ССА(Сингулярный спектральный анализ) предложенного и своего (моделью из ЛР 1 из трех гармоник) сигналов.
- 2) Добиться разделимости компонент своего сигнала ( нужным образом подобрать L и сгруппировать)
- 3) Сравнить с фурье и вейвлет-анализом

## Пункт 1

ССА - используется для анализа временных рядов, выделения трендов, гармоник, шума и других составляющих сигналов.

Для анализа надо выполнить несколько пунктов, а точнее:

- 1) Создание траекторной матрицы.
- 2) С помощью SVD (сингулярное разложение) выполнить разложение траекторной матрицы.
- 3) Сгруппировать сингулярные компоненты.
- 4) Выполнить обратное преобразование (Ханкелизация).

Предложенный сигнал генерируется из двух гармоник, тренда и шума.

Параметры сигнала следующие:

$$\begin{aligned}N_{\text{signal}} &= 1024 \\dt &= 1 / 12 \\P1 &= 10 / dt \\P2 &= 1 / dt\end{aligned}$$

Где:

$N_{\text{signal}}$  - задаёт длину временного ряда.

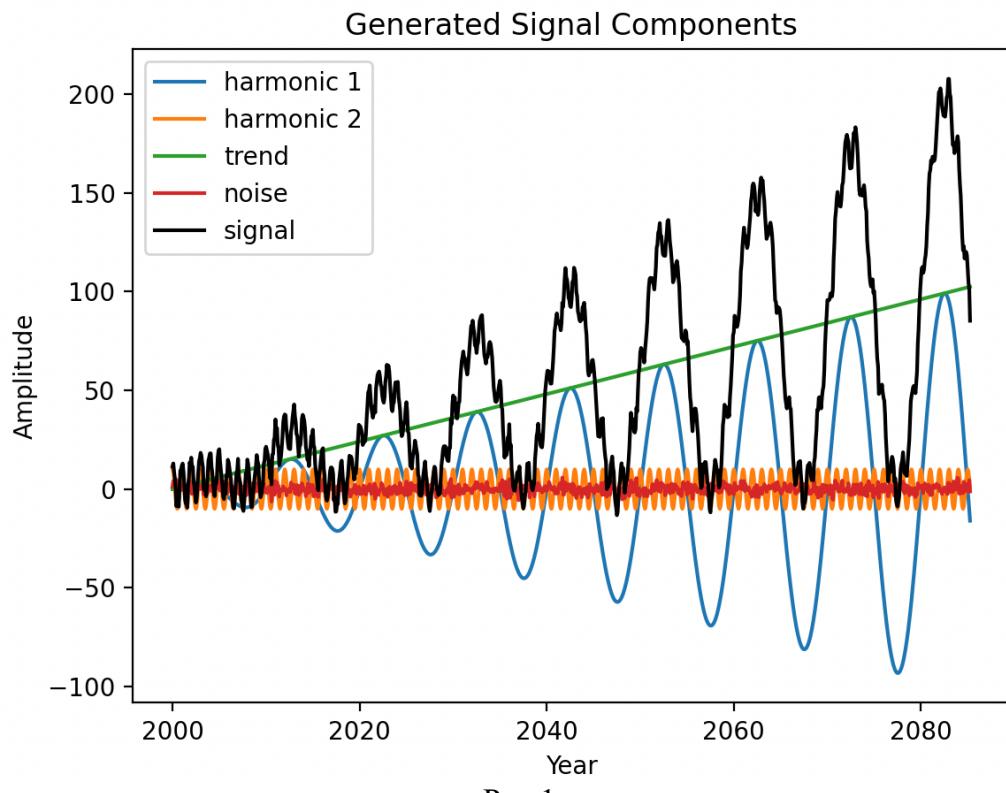
$dt$  - определяет временной шаг, связывающий индекс времени с реальным временем.

$P1$  и  $P2$  – гармоники

Все компоненты задаются в виде:

$$\begin{aligned}garm_1 &= 0.1 \cdot (k + 1) \cdot \sin\left(\frac{2\pi k}{P_1}\right) \\garm_2 &= 10 \cdot \cos\left(\frac{2\pi k}{P_2}\right) \\trend &= 0.1(k + 1)\end{aligned}$$

Отобразим наш смоделированный сигнал на Рис.1



И выполним ССА по описанному выше алгоритму (код программы matlab был адаптирован на Python, в Приложении 1 продемонстрирована его реализация)  
Параметры ССА были выбраны следующие:

$L = 300$

Groups = [[0,1], [2,3], [4,5]]

[0,1] – сигнал

[2,3] – harmonic 1

[4,5] – harmonic 2

$N_{ev} = 7$

Результаты изображены на Рис.2

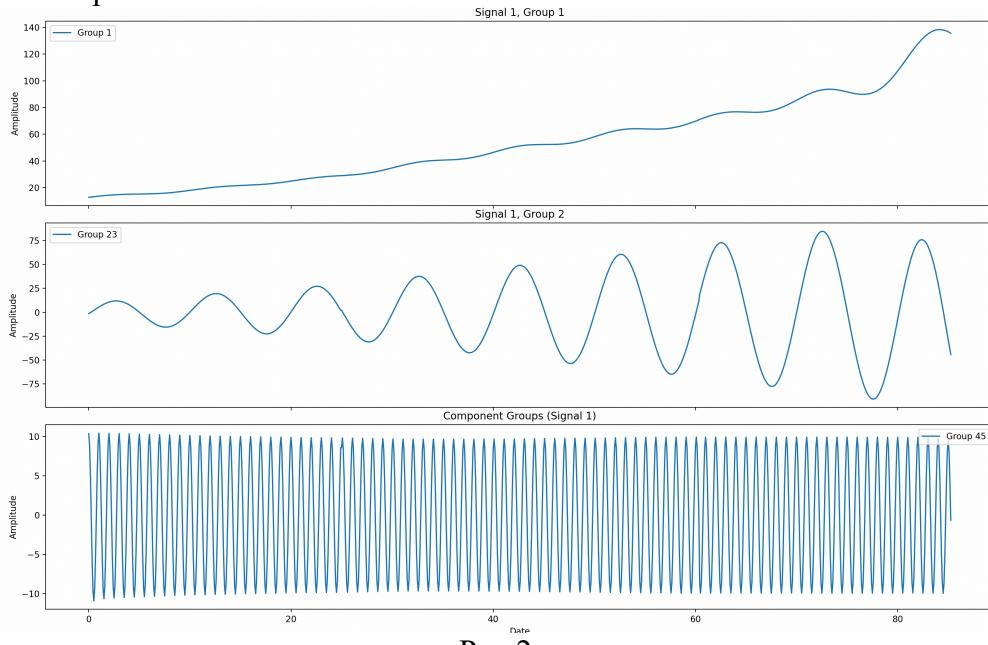


Рис.2

Для смоделированного сигнала из ЛР1 имеем следующий график (Рис.3)

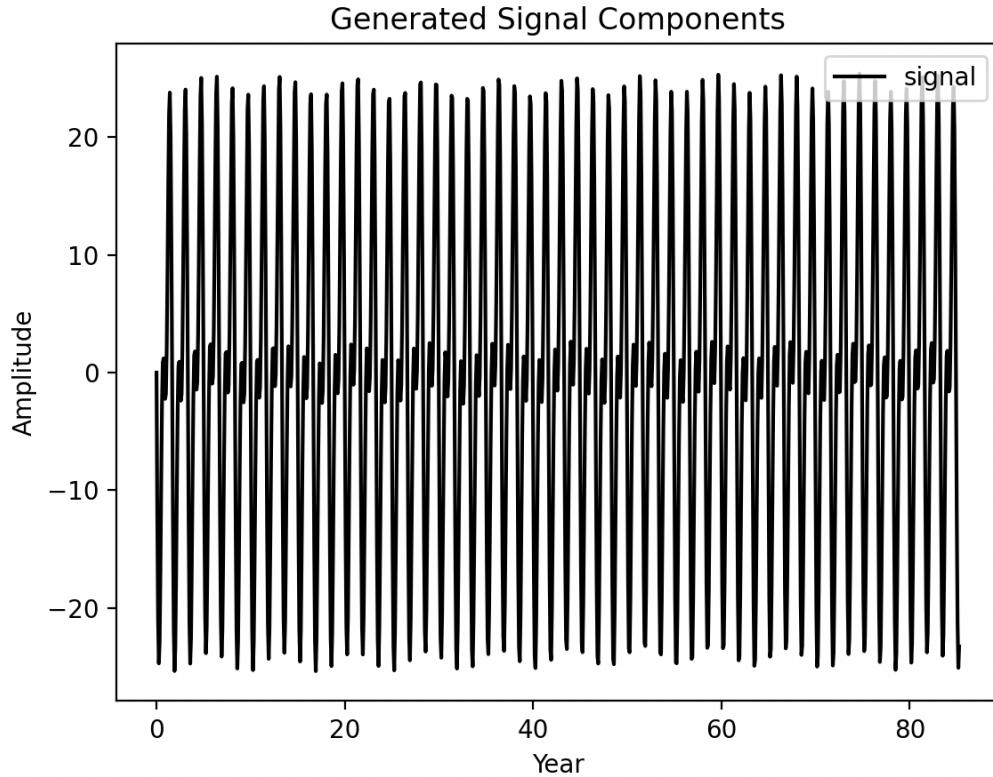


Рис.3

Как и в ЛР1 параметры сигнала следующие:

$$a = 18, b = 10, c = 2002$$

# Периоды (в годах) T1 = 0.5

T2 = 1

T3 = 4.6

$$A1 = (a / 31) * 20$$

$$A2 = (b / 12) * 20$$

$$A3 = ((c - 2000) / 50) * 20$$

$$X_{\text{model}} = (A1 * \text{np.cos}(2 * \text{np.pi} * t / T1 + \phi_1)) +$$

$$A2 * \text{np.cos}(2 * \text{np.pi} * t / T2 + \phi_2) + A3 * \text{np.cos}(2 * \text{np.pi} * t / T3 + \phi_3))$$

Для ССА сначала применим ту же группировку и в целом те же параметры, получим следующий результат на изображении Рис.4

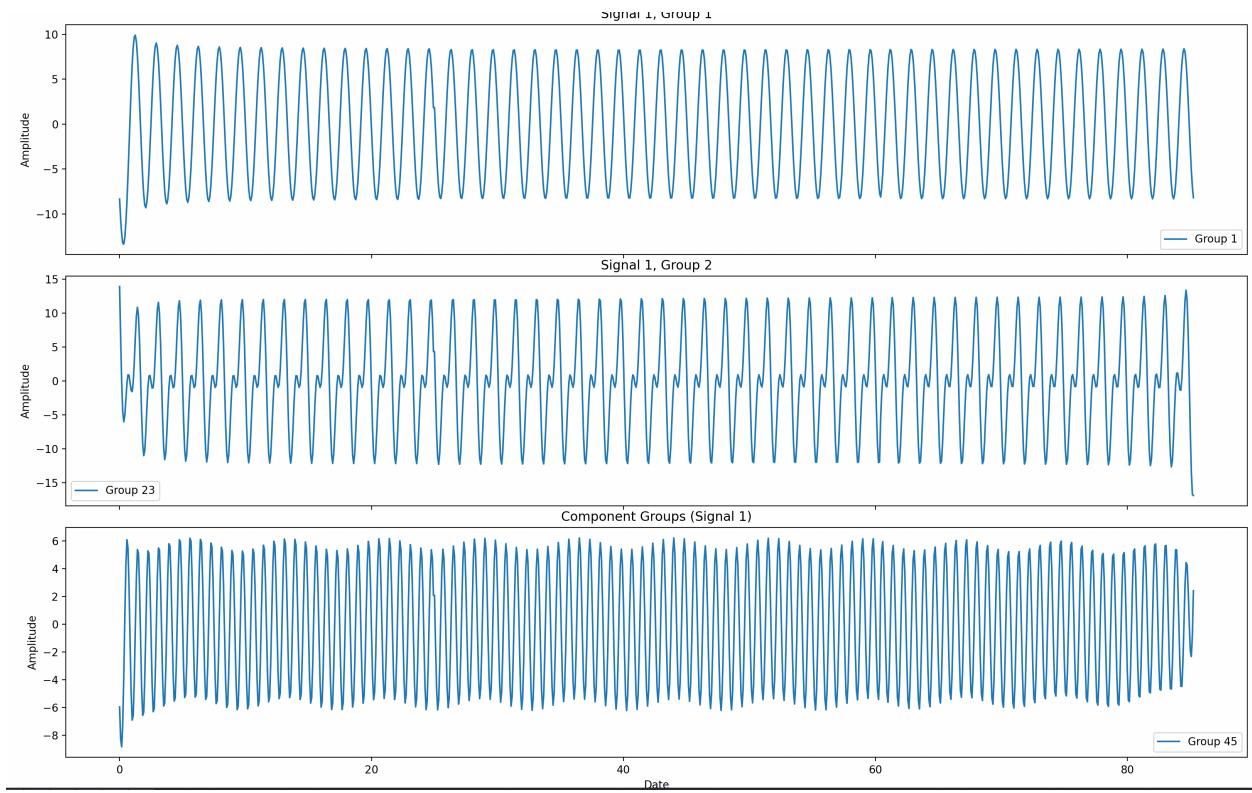


Рис.4

На данном изображении мы видим ряд артефактов, а также не совсем понятно верное ли разложение у нас вышло.

Изобразим сигналы по отдельности на Рис.5 – Рис.7

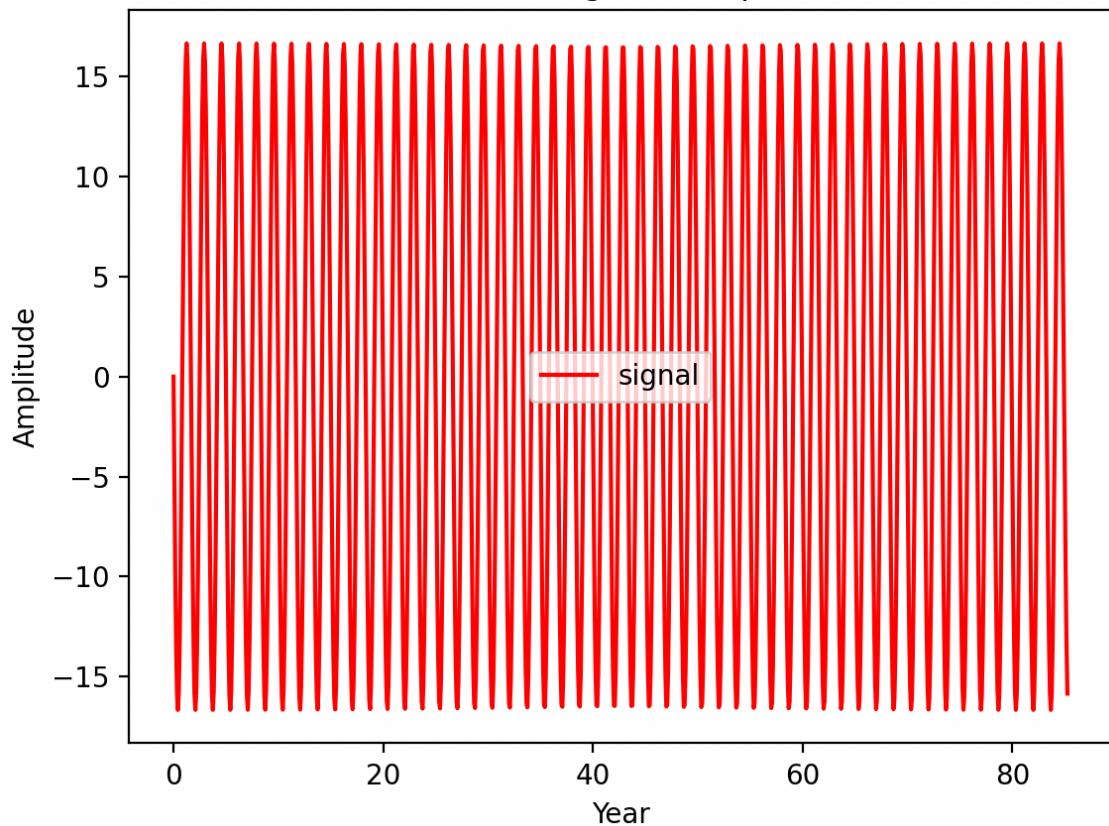


Рис.5

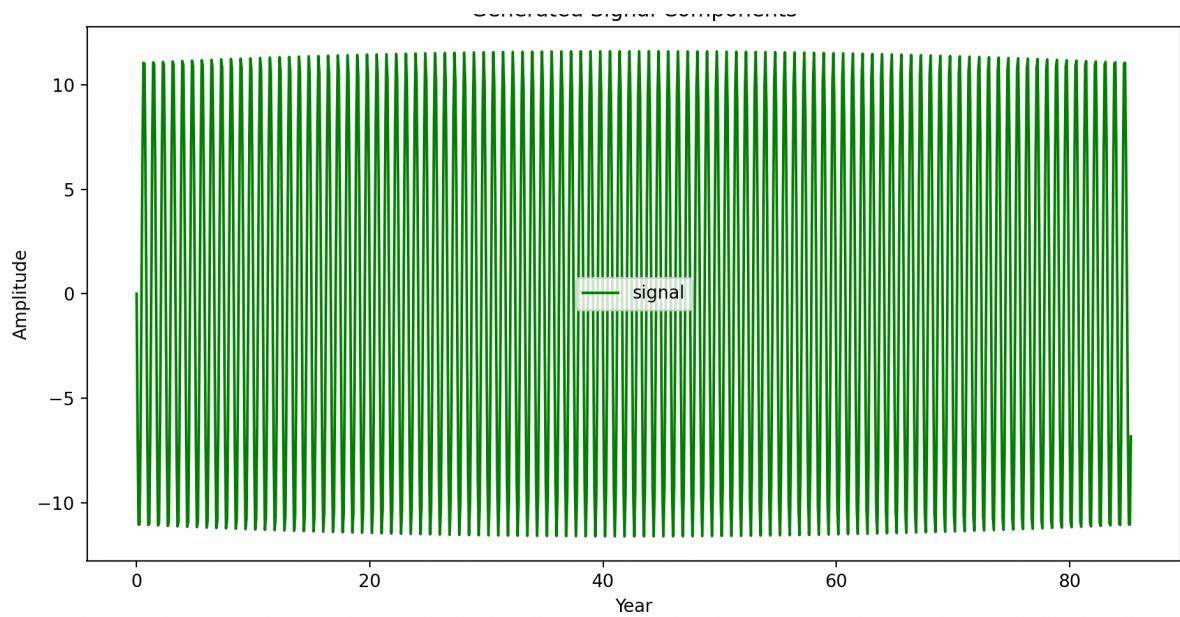


Рис.6

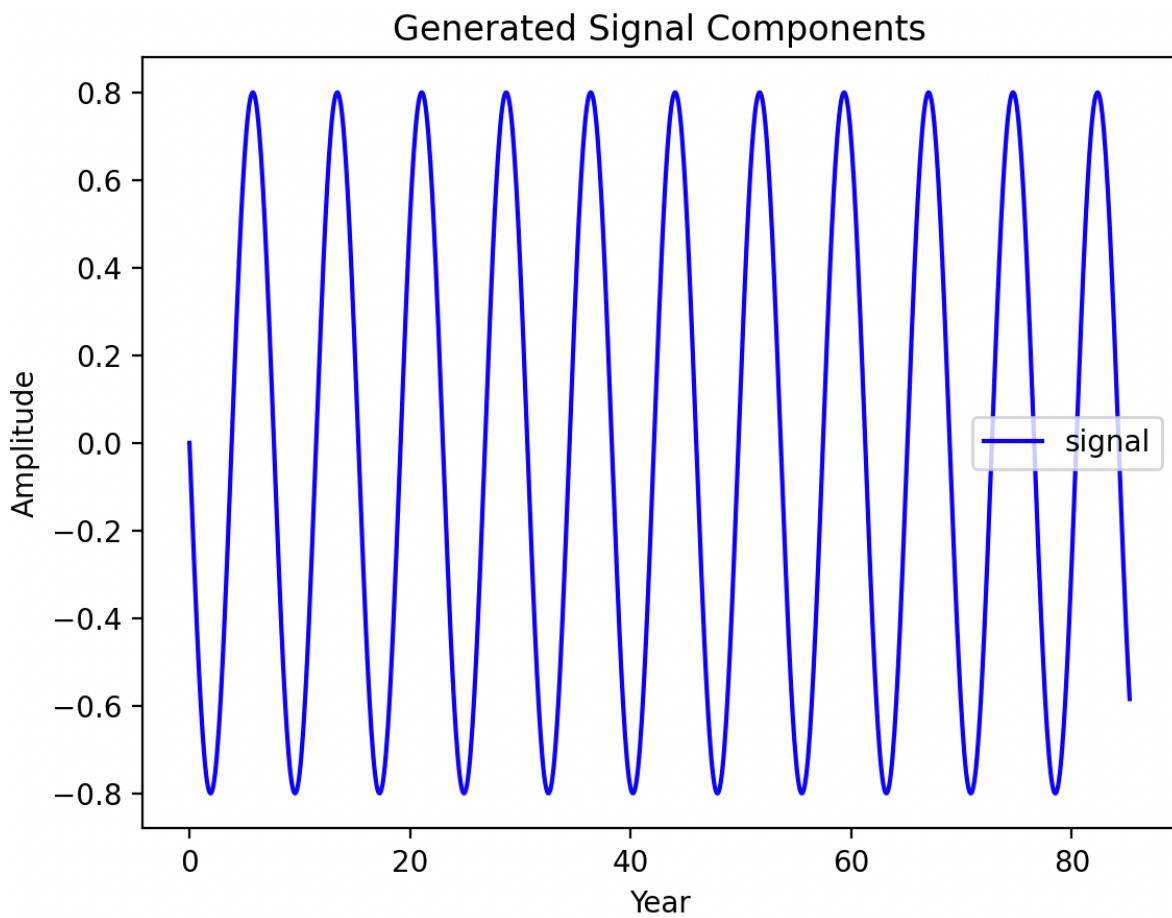


Рис.7

В целом, сразу заметно, что у 3 группы уж слишком расходится графики, а также значение амплитуд у остальных.

## Пункт 2.

Результаты расчета из п.1 могут свидетельствовать о том, что была выбрана неверная группировка или иные параметры ( $L$ ), перегруппируем компоненты следующим образом:  $[[1, 2], [3, 4], [5, 6]]$ .

$L$  возьмем равным 350 (В идеале, оно должно делиться на периоды гармоник)  
И остальные значения пока не трогаем. Получим следующие результаты на Рис. 8

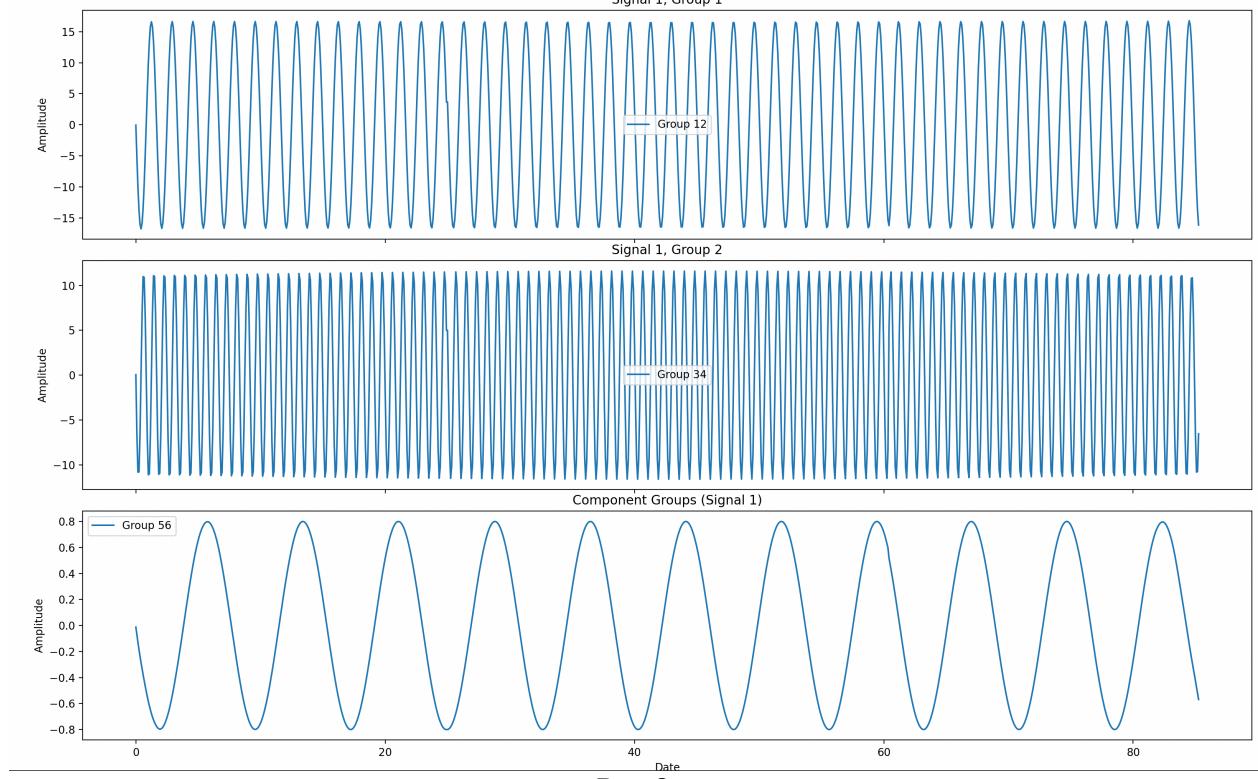


Рис.8

Данные результаты расчета уже совпали, как по значениям, так и по поведению в целом.  
Стоит еще добавить, что  $L$  должно быть достаточно большим, чтобы улавливать основные тренды и гармоники.

Если  $L$  слишком мал, разложение может потерять информацию о, например, трендах.

Если  $L$  слишком велик, может стать сложнее выделить гармоники, и могут появиться шумы.

В целом,  $L$ , как и группы стоит выбирать так, чтобы не графике не было каких-либо артефактов и они совпадали с каждой компонентой по отдельности. В нашем случае это проверить возможно, в общем случае это может оказаться довольно сложно, поэтому алгоритм должен работать безотказно.

### Пункт 3.

Для данного пункта выполним БПФ и Вейвлет преобразование для смоделированного сигнала из п.1 аналогично тому, что делали в предыдущих ЛР.

Получим следующие результаты (Рис.9 и Рис.10)

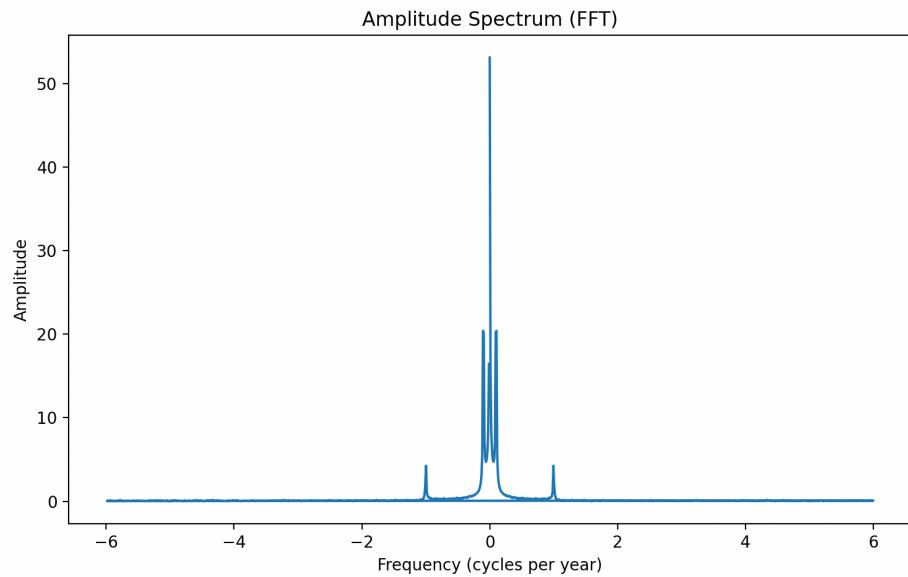


Рис.9

Scalogram (CWT) with Period on Y-axis

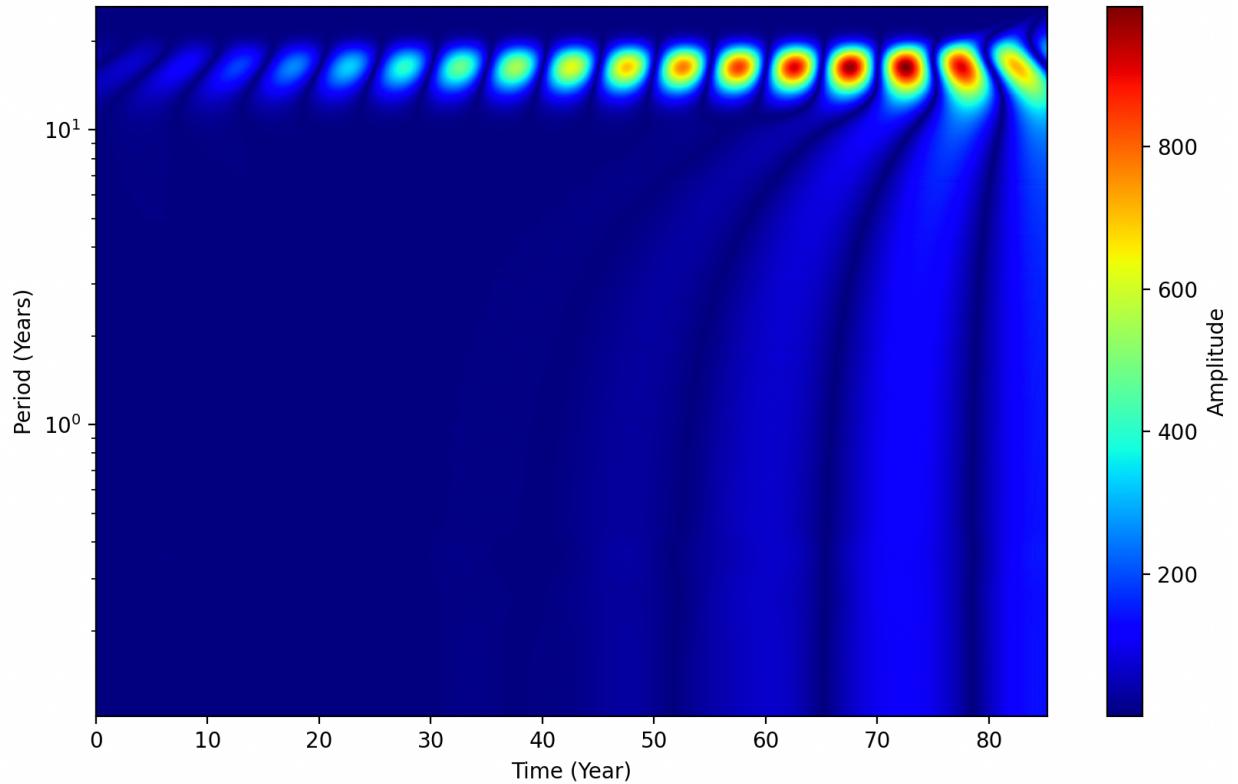


Рис. 10

Стоит сразу отметить, что в отличии от реализации в Matlab, здесь реализован обычный вейвлет Морле. Комплексный вейвлет Морле даст несколько измененную скейлограмму, которая довольно сильно расходится с версией в Matlab (Рис.11)

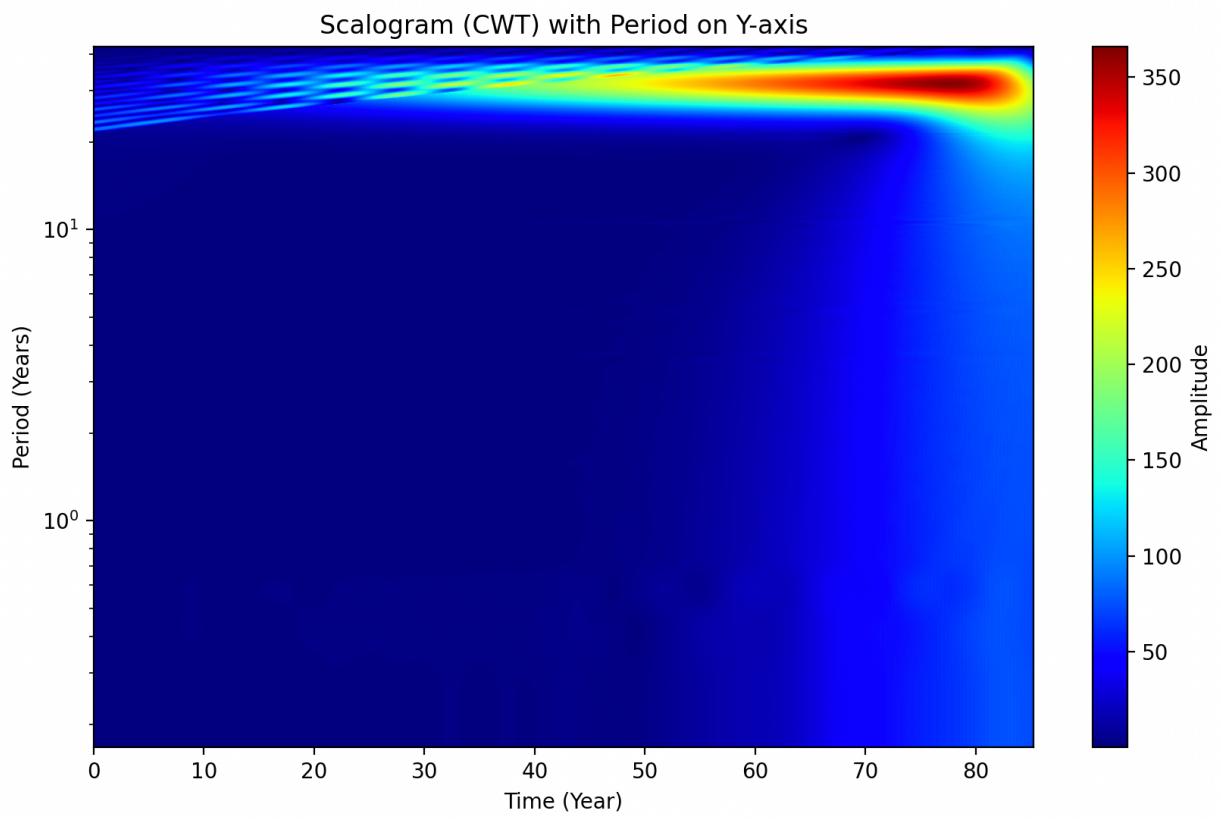


Рис.11

Здесь явно выраженное отклонение по амплитуде, в отличии от версии amor.  
Проблема может быть в самой реализации встроенных функций библиотеки Pywavelet.

Аналогично для сигнала из LP1 получим следующее (Рис.12 – Рис.13)

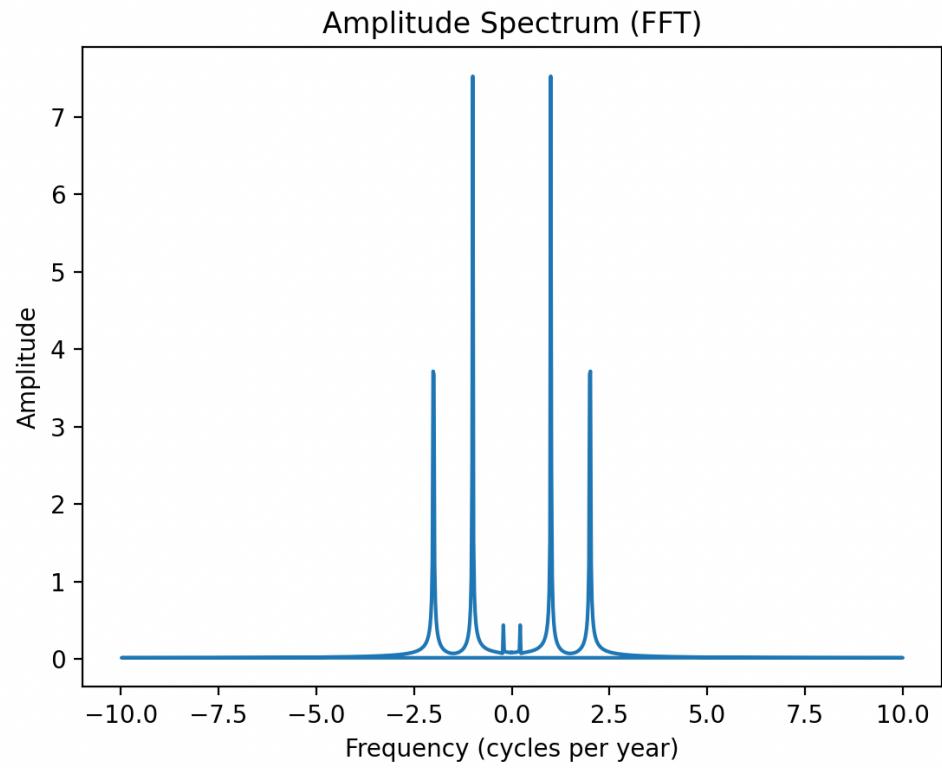


Рис.12

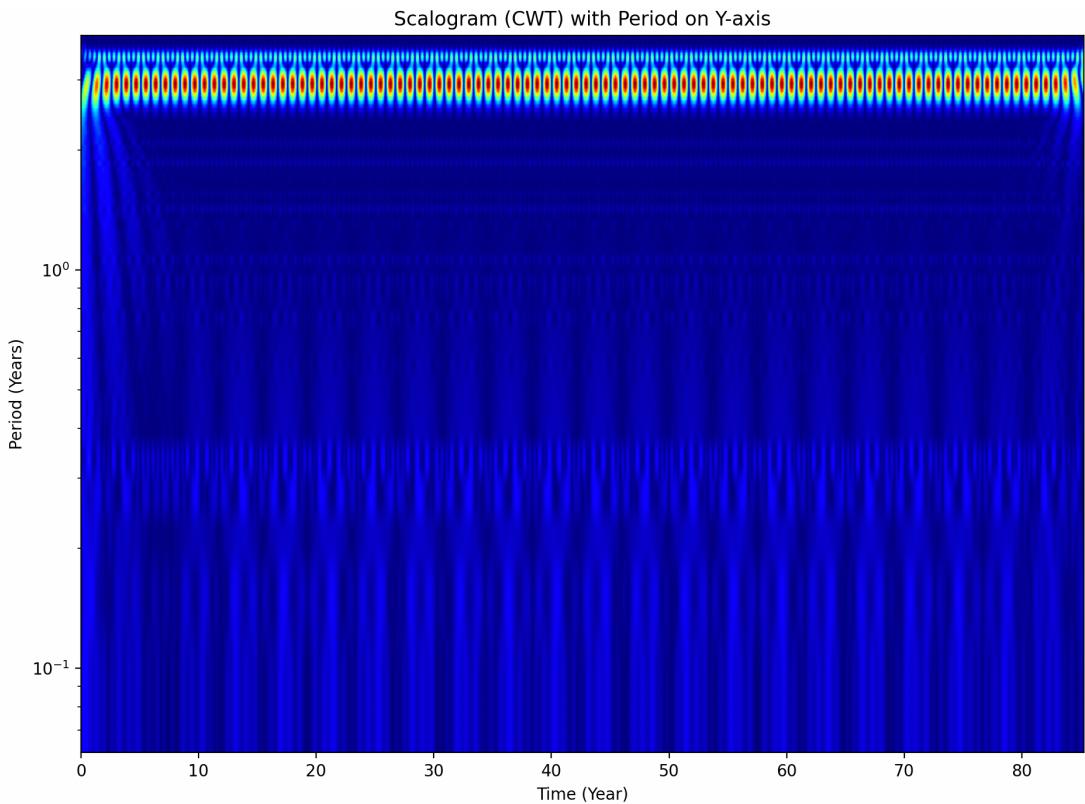


Рис.13

При сравнении БПФ и ССА обратим внимание на то, что в нашем варианте нет шума и тренда, и в целом на спектрограмме сигнала из ЛР1 выделены 3 гармоники, ССА также разложил исходный сигнал на 3 составляющие без дополнительных шумов и трендов. (Доп. компоненты в группировке отобразили с амплитудой порядка  $10^{-13}$ , что очень незначительно)

Вейвлет преобразование в целом отобразило поведение нашего сигнала с его амплитудой (энергией), а также мы видим основное колебание при малых масштабах равных около 64. В целом, как и в ЛР3 здесь выделяется поведение сигнала, как не странно, при больших скейлах не наблюдаются колебания, наш сигнал не затухает и не может распознаваться однозначно при больших масштабах.

#### **Вывод 4.**

В данной работе продемонстрировано применение ССА на двух смоделированных сигналах, один из которых из ЛР1.

Была произведена адаптация исходного программного кода с matlab на python и проанализированы результаты двух программных кодов.

Результаты разошлись только в реализации вейвлет преобразования, в остальном все совпало.

Были подобраны параметры для разложения сигнала по составляющим без каких-либо артефактов.

Для проверки верного разложения строились отдельные компоненты сигнала и результат ССА для смоделированных сигналов. Результаты сошлись.

Проанализированы результаты ССА для смоделированного сигнала и из ЛР1 в сравнении с БПФ и вейвлет преобразованием. Все методы были реализованы верно.

#### **Приложение 1.**

```

def ampl_fft(signal, dT):
    N = len(signal)
    fourier_transform = np.fft.fft(signal)
    spectr = fourier_transform / N

    omega = np.zeros(N)
    t = np.zeros(N)

    for j in range(N):
        if j == 0:
            omega[j] = 0
        elif j <= N // 2:
            t[j] = N * dT / (j)
            omega[j] = 2 * np.pi / t[j]
        else:
            t[j] = N * dT / (N - j)
            omega[j] = -2 * np.pi / t[j]

    return spectr, omega


def hankelization(G1, L, N, L_min, K_max):
    if L != L_min:
        return np.zeros(N)

    X1 = np.zeros(N)

    # Краевые элементы
    for ii in range(1, L_min + 1):
        for j in range(ii):
            X1[ii - 1] += G1[ii - j - 1, j]
            X1[N - ii] += G1[L_min - (ii - j), K_max - j - 1]
        X1[ii - 1] /= ii
        X1[N - ii] /= ii

    # Центральные элементы
    for ii in range(L_min, K_max):
        for j in range(L_min):
            X1[ii] += G1[L_min - j - 1, ii - L_min + j]
        X1[ii] /= L_min

    return X1


def mssa(Date, SIGNAL, N_loc, N, L, N_ev, coef, pathout, group_seq):
    K = N - L + 1
    A = np.zeros((L * N_loc, K))

    for ii in range(L):
        for j in range(K):
            for k in range(N_loc):
                A[k * L + ii, j] = SIGNAL[k, ii + j]

    U, S, Vt = np.linalg.svd(A, full_matrices=False)
    V = Vt.T

    RX = np.zeros((N_ev, N_loc, N))
    for ii in range(N_ev):
        G = np.outer(U[:, ii], S[ii] * V[:, ii])
        for k in range(N_loc):
            Block = G[k * L:(k + 1) * L, :]
            RX[ii, k, :] = hankelization(Block, L, N, min(L, K), max(L, K))

    components = np.zeros((N_loc, len(group_seq), N))

```

```

    # fig, axes = plt.subplots(N_loc, len(group_seq), figsize=(15, 5 *
N_loc), sharex=True)
    fig, axes = plt.subplots(N_loc * len(group_seq), 1, figsize=(10, 5 *
N_loc * len(group_seq)), sharex=True)
    for l in range(N_loc):
        for m, group in enumerate(group_seq):
            label = ""
            for ind in group:
                if ind > 0:
                    components[l, m, :] += RX[ind - 1, l, :]
                    label += str(ind)
            # plt.plot(Date, components[l, m, :], label=f"Group {label}")
            # ax = axes[l, m] if N_loc > 1 else axes[m] # Если только 1
компонент, индексация упрощается
            # ax.plot(Date, components[l, m, :], label=f"Group {label}")
            ax_idx = l * len(group_seq) + m # Индекс текущего графика
            ax = axes[ax_idx] if len(axes) > 1 else axes # Если всего один
график, убрать массивность
            ax.plot(Date, components[l, m, :], label=f"Group {label}")
            ax.legend()
            ax.set_title(f"Signal {l + 1}, Group {m + 1}")
            ax.set_ylabel("Amplitude")
            if ax_idx == len(axes) - 1:
                ax.set_xlabel("Date")
    # plt.legend()
    plt.tight_layout()
    plt.title(f"Component Groups (Signal {l + 1})")
    plt.show()

return components

```