РАБОТА №5. ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

Цель работы: формирование навыков применения прямого и обратного преобразования Фурье для дискретных сигналов, а также их использования для получения частотного спектра сигнала.

Планируемая продолжительность: 2 академических часа.

Тип работы:  с использованием компьютерных средств.

Основные понятия преобразования Фурье для дискретных сигналов

В данной работе будут употреблены следующие сокращения:

ДПФ – дискретное преобразование Фурье (преобразование Фурье для сигналов, заданных массивами чисел);

ОДПФ – обратное дискретное преобразование Фурье (восстановление исходного массива из коэффициентов, полученных с помощью ДПФ);

БПФ – быстрое преобразование Фурье (является разновидностью ДПФ, значительно ускоренной, но только для массивов, количество элементов которых равна 2^m, где m=1,2,…);

ОБПФ – обратное быстрое преобразование Фурье (восстановление исходного массива из коэффициентов, полученных с помощью БПФ);

Преобразование Фурье переводит дискретный или непрерывный сигнал из временной в частотную область (по оси x вместо времени t следует откладывать частоту ω), причем полученные коэффициенты имеют комплексный вид. Модуль этих коэффициентов показывает распределение амплитуд по частотам в сигнале, а аргумент – распределение фазовых сдвигов по частотам.

Начнем с рассмотрения ДПФ. Формула для его реализации известна (задает k-й коэффициент Фурье на основе n-го элемента исходного массива x, который состоит из N элементов):

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-\frac{2\pi i n k}{N}}. \quad (5.1)$$

Восстановить же сигнал можно согласно формуле ОДПФ:

$$x_k = \frac{1}{N} \sum_{k=0}^{N-1} X_k \, e^{\frac{2\pi i n k}{N}}. \quad (5.2)$$

БПФ является частным случаем ДПФ, когда $N=2^m$, где m – произвольное натуральное число. Выигрыш достигается за счет отдельного расчета четных и нечетных коэффициентов. После разделения ДПФ на четные и нечетные коэффициенты можно к следующему выражению:

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n}\, e^{-\frac{4\pi i n k}{N}} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}\, e^{-\frac{2\pi\,(2n+1)k}{N}}. \qquad (5.3)$$

которое и задает БПФ.

Число математических операций при этом значительно уменьшается по сравнению с ДПФ. ОБПФ можно получить аналогично ОДПФ:

$$X_n = \sum_{k=0}^{\frac{N}{2}-1} x_{2k}\, e^{\frac{4\pi i n k}{N}} + \sum_{k=0}^{\frac{N}{2}-1} x_{2k+1}\, e^{-\frac{2\pi\,(2k+1)n}{N}}. \qquad (5.4)$$

Реализация преобразования Фурье в Python

Начнём подготавливать окружение. Установим необходимые библиотеки.

```
! pip install -U pip
! pip install -U matplotlib
! pip install -U numpy
! pip install -U scipy
```

Рис. 5.1. Установка библиотек

Импортируем следующие зависимости:

```
import numpy as np
import math
import scipy

from matplotlib import pyplot as plt
%matplotlib widget

from scipy.fft import fftfreq, fft
from typing import Callable
from random import randint
```

Рис. 5.2. Импортирование зависимостей

Из предыдущей работы возьмём метод plot и немного изменим, чтобы можно было отрисовывать на графике N сигналов.

```
def plot(*args):
    ax = plt.figure()
    for idx in range(0, len(args), 2):
        x, y = args[idx], args[idx + 1]
        plt.plot(x, y)
    plt.grid(True)
    plt.ylabel('y')
    plt.xlabel('x')
    plt.show()
```

Рис 5.3. Функция отрисовки графиков

Реализуем функцию сигнала. Пусть это будет простейший y(x)=sin(2πx).

```
def y_1(x):
    return math.sin(2*x*math.pi)
```

Рис 5.4. Функция сигнала

Теперь нужно создать дискретные отсчёты на основе этой функции, для этого нужно задать количество дискретных отсчётов и диапазон отсчётов, при помощи функции np.arange.

```
DISCRETE_COUNT_MAX = 32
discrete_range = np.arange(stop=DISCRETE_COUNT_MAX)
```

Рис. 5.5. Дискретные отсчёты

На основе полученного диапазона и нашей функции создадим массив дискретных значений нашей функции через отдельный метод. Доесть каждое значений диапазона передаём в функцию, разделив на общее количество значений.

```
def discrete_values(
    func: Callable,
    discrete_range: np.ndarray,
) -> np.ndarray:
    result = np.zeros(discrete_range.shape)
    max_value = discrete_range.size
    for idx, value in enumerate(discrete_range):
        result[idx] = func((value) / max_value)
    return result
```

Рис 5.6. Функция дискретных значений

Также можете заметить, result является np.ndarray (потому что сгенерирован через np.zeros), в отличие от предыдущей работы, где в качестве результата был нативный list. Вам никто не запрещает использовать списковое включение (list comprehension), но помните, работать это будет медленнее, python для списков выделяет память динамически, и не всегда блок памяти будет в одной области, в отличие от np.zeros, где память резервируется под размер нашего массива. Как пример из жизни, нативный список – это как набитая полка с книгами, куда вам нужно положить ещё и свои книги, вы определяете, где свободное место, если места нет, то какие книги можно выбросить. Numpy массив – грубо говоря выкидывает все книги, и сразу же их складывает.

Теперь вызовем нашу функцию discrete_values, напечатаем значения и нарисуем график данных

```
1  dv = discrete_values(y_1, discrete_range)
2  print(dv)
3  plot(discrete_range, dv)
```

Рис 5.7. Вызов и отрисовка дискретных значений

Получим следующее:

```
[ 0.00000000e+00  1.95090322e-01  3.82683432e-01  5.55570233e-01
  7.07106781e-01  8.31469612e-01  9.23879533e-01  9.80785280e-01
  1.00000000e+00  9.80785280e-01  9.23879533e-01  8.31469612e-01
  7.07106781e-01  5.55570233e-01  3.82683432e-01  1.95090322e-01
  1.22464680e-16 -1.95090322e-01 -3.82683432e-01 -5.55570233e-01
 -7.07106781e-01 -8.31469612e-01 -9.23879533e-01 -9.80785280e-01
 -1.00000000e+00 -9.80785280e-01 -9.23879533e-01 -8.31469612e-01
 -7.07106781e-01 -5.55570233e-01 -3.82683432e-01 -1.95090322e-01]
```

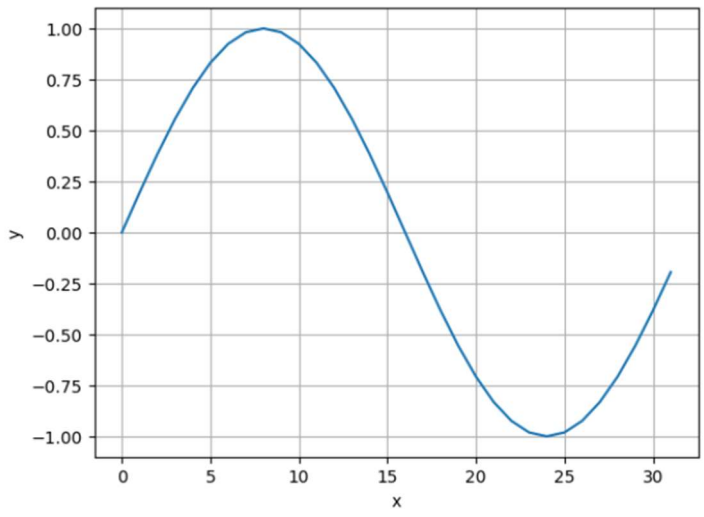

Рис. 5.8. Данные и их график

Реализуем метод расчёта ДПФ в соответствии с формулой 5.1. Как видим, функция должна принимать номер выходного коэффициента k и имя входного массив в качестве параметров возвращает мнимую сумму

```
def dft_func(
    array: np.ndarray,
    k: int
) -> np.imag:
    def F(item, n):
        return item * math.e ** (
            (-1) * ((2 * math.pi * k * n * 1j) /
                    array.size - 1)
             )

    result = np.zeros(array.size, dtype=np.complex_)
    for idx in range(array.size):
        result[idx] = F(array[idx], idx)
    return np.sum(result)
```

Рис. 5.9. Метод дискретного преобразования Фурье

Теперь используя реализованный метод dft_func создадим ещё один метод формирования массива ДПФ.

```
def count_discrete_fourier_on(
    array: np.ndarray,
) -> np.ndarray:
    result = np.zeros(array.shape, dtype=np.complex_)
    for idx in range(array.size):
        result[idx] = discrete_fourier_transform(array, idx)
    return result
```

Рис. 5.10. Метод формирования массива ДПФ

Нарисуем по абсолютному значению результат работы этого массива для наших дискретных значений.

```
fourier_array = count_discrete_fourier_on(discrete_values)
plot(discrete_range, np.abs(fourier_array))
```

Рис 5.11. – Вызов и отрисовка функции
формирования массива Фурье

Получим следующий график:

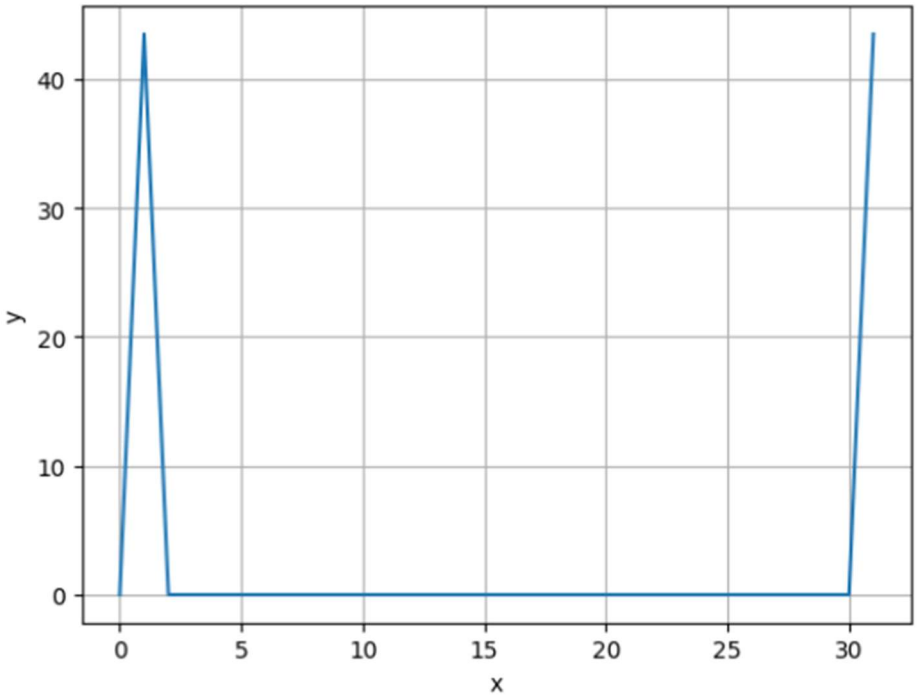

Рис. 5.12. Спектр сигнала

Как видим, мы получили спектр сигнала. Спектр сигнала симметричен (если не считать самый первый коэффициент, который характеризует наличие постоянной составляющей). Это объясняется тем, что преобразование Фурье, в общем случае, осуществляется в пределах от -∞ до ∞, и правая половина спектра неинформативна.

В данном случае, максимальным информативным коэффициентом является (N/2+1)-й, поскольку он соответствует половине частоты дискретизации, выше этого значения физически полезные данные не могут быть найдены.

При помощи инструмента масштабирования приблизим первый пик на графике.

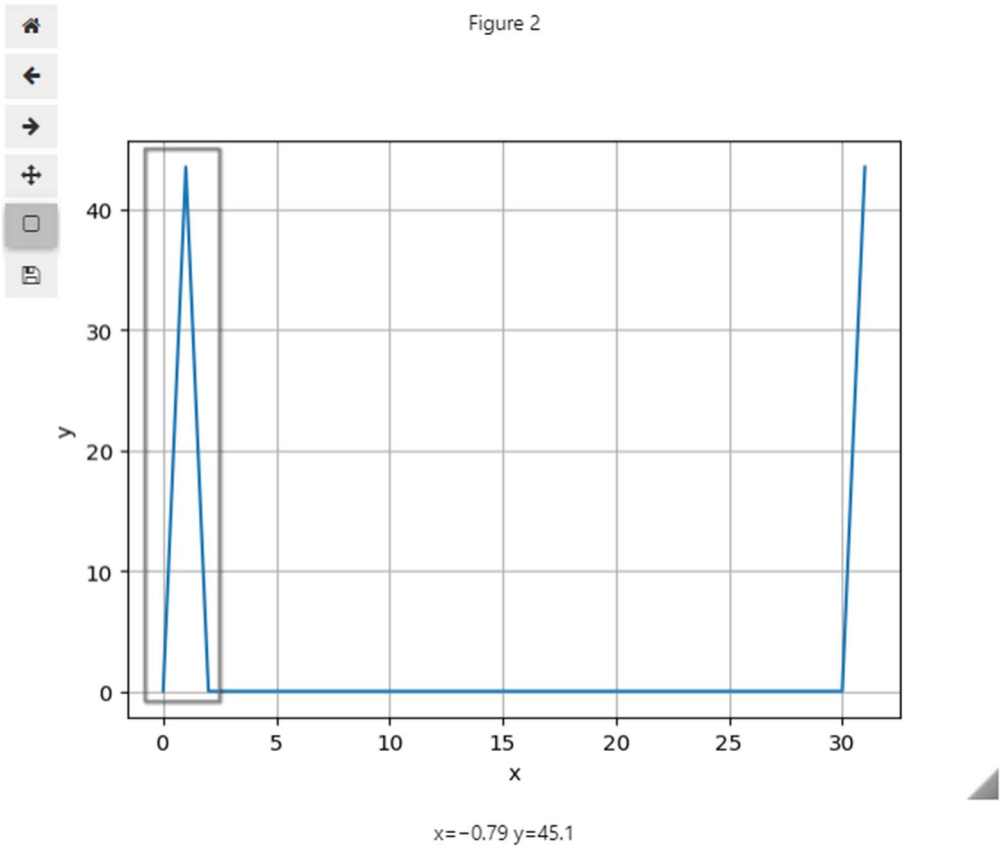

Рис. 5.13. Масштабирование графика

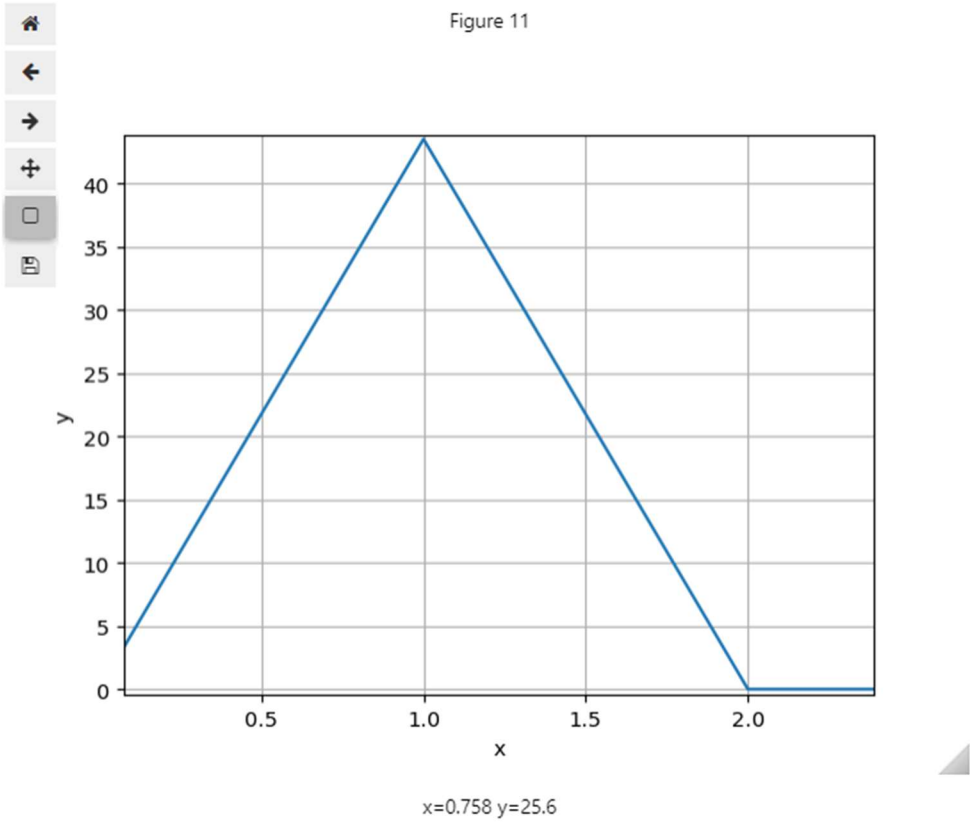

Рис. 5.14. Первый пик спектра

Анализируя спектр, можно сказать, что в сигнале присутствует только одна частота, соответствующая второму коэффициенту Фурье (первому, если обозначить коэффициент, соответствующий постоянной составляющей, как нулевой). Это суждение корректно, поскольку исходный сигнал сформирован с помощью функции $\sin(2\pi x)$, то есть содержит одну частоту, равную 1 Гц.

Теперь реализуем обратную задачу, а именно ОДПФ по формуле 5.2. Повнимательнее приглядимся к функциям 5.1 и 5.2, у 5.2 суммирующая часть ничем не отличается от 5.1, кроме знака у степени, поэтому мы можем использовать ту же функцию dft_func, только на вход передавать отрицательный индекс и дописать деление

```
1  def reverse_fourier_transform(
2      array: np.ndarray
3  ) -> np.array:
4      result = np.zeros(array.shape, dtype=np.complex_)
5      for idx in range(array.size):
6          result[idx] = (1 / len(array)) * \
7              (discrete_fourier_transform(array, (-1) * idx))
8      return result
```

Рис. 5.15. Метод формирования массива ОДПФ

Вызовем эту функцию на полученном ранее массиве ДПФ и выведем график:

```
1  reversed_fourier_array = reverse_fourier_transform(fourier_array)
2  plot(discrete_range, reversed_fourier_array)
```

Рис. 5.16. Вызов ОДПФ с вызовом отрисовки графика

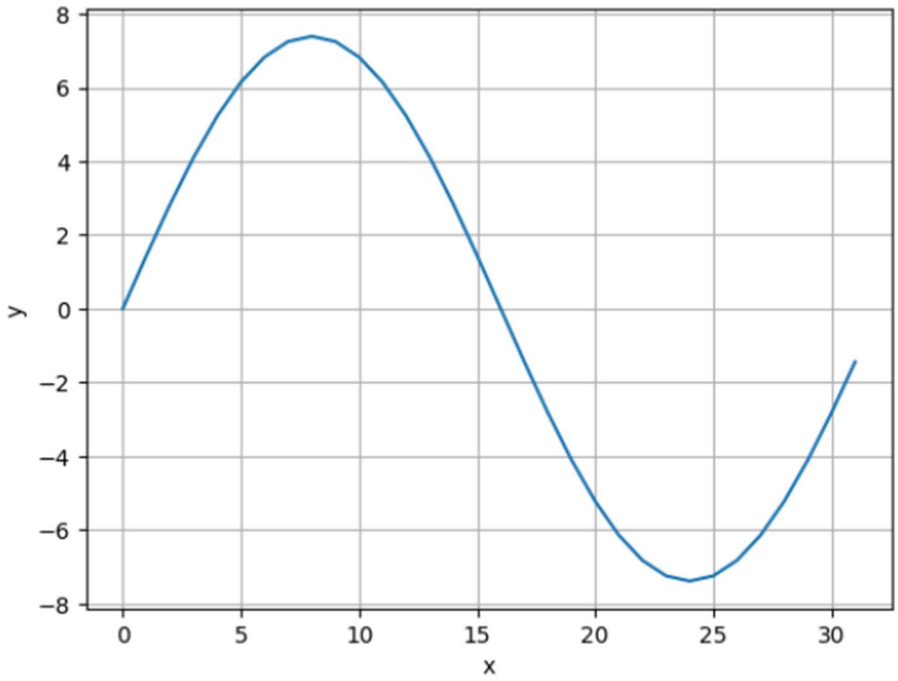

Рис. 5.17. График восстановленной по ОДПФ функции

Таким образом получили график исходной функции на основе ряда Фурье.

Чаще всего, в математических пакетах уже существует реализация БПФ, как одного из наиболее значимых инструментов анализа сигналов. Для этой задачи будем использовать SciPy и его функции fft и fftreq.

Зададим новый сигнал, причем состоящий из двух синусоид различных частот и шумовой компонент.

```
1  def y_2(x):
2      return np.sin(2 * math.pi * 50 * x) + np.sin(2 * math.pi * 120 * x) + (randint(0, 100) / 1000)
```

Рис. 5.18. Функция сигнала

Введём переменные количество точек сигнала – SIGNAL_POINTS, частоту дискретизации – SAMPLE_SPACING, x – в данном случае будет массив отсчётов, этот массив мы передаём в нашу функцию, выведем график функции. В fft передаём массив данных получаем массив из БПФ, далее на основе этого массива создаём массив частот при помощи fftreq. Так как полезная часть находится в первой половине – делим массив частот по полам.

```
SIGNAL_POINTS = 600
SAMPLE_SPACING = 1.0 / 800
x = np.arange(0.0, stop=SIGNAL_POINTS*SAMPLE_SPACING, step=SAMPLE_SPACING)
y = y_2(x)
plot(x, y)
yf = fft(y)
xf = fftfreq(SIGNAL_POINTS, SAMPLE_SPACING)[:SIGNAL_POINTS//2]
plot(xf, 2.0/SIGNAL_POINTS * np.abs(yf[0:SIGNAL_POINTS//2]))
```

Рис. 5.19. БПФ со спектром сигнала

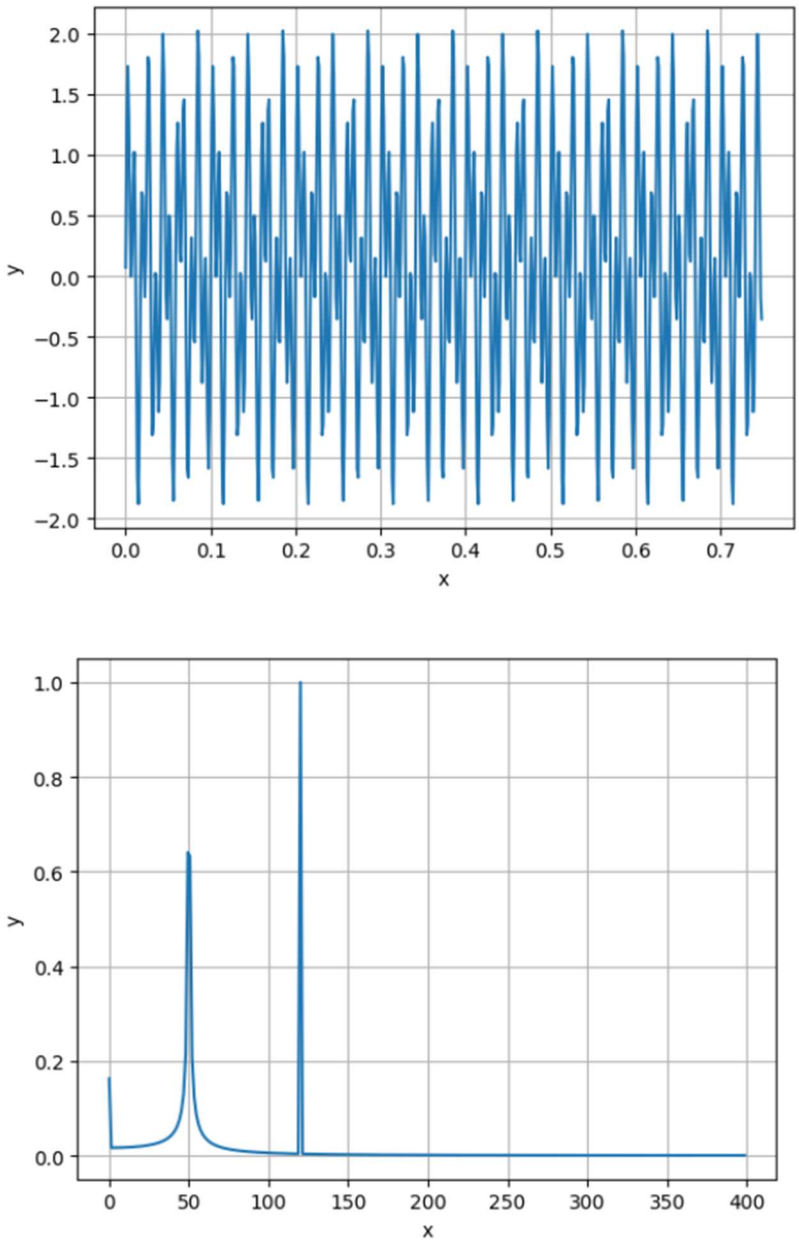

Рис. 5.20. График сигнала и его спектра

Можно заметить, что на исходном сигнале визуально едва ли заметны доминирующие частоты, тогда как на спектре сигнала четко представлены два пика, соответствующие этим частотам.

Задание 1

В соответствии с описанным материалом реализуйте ДПФ, ОДПФ и БПФ, построив соответствующие графики. При этом, используйте в качестве y(x) и f(x) указанные в таблице ниже функции, в соответствии с вариантом. Для всех вариантов в f(x) присутствует случайная аддитивная помеха `(randint(0, 100) / 1000)`.

Таблица 5.1

Варианты заданий

Варианты	**y(x)**	**f(x)**
1,6,11, 16,21,26	2*(Sin[4πx])	$\sin(2\cdot\pi\cdot14\cdot x)+\sin(2\cdot\pi\cdot70\cdot x)+\frac{\text{random}(100)}{1000}$
2,7,12, 17,22,27	Cos[3πx]	$\cos(2\cdot\pi\cdot25\cdot x)+0,5\cdot\sin(2\cdot\pi\cdot70\cdot x)+\frac{\text{random}(100)}{1000}$
3,8,13, 18,23,28	0.5*(Sin[8πx])	$0,3\cdot\cos(2\cdot\pi\cdot31\cdot x)+1,5\cdot\cos(2\cdot\pi\cdot39\cdot x)+\frac{\text{random}(100)}{1000}$
4,9,14, 19,24,29	0.5*(Cos[10πx])	$0,1\cdot\sin(2\cdot\pi\cdot11\cdot x)+\cos(2\cdot\pi\cdot100\cdot x)+\frac{\text{random}(100)}{1000}$
5,10,15, 20,25,30	1.3*(Sin[14πx])	$\sin(2\cdot\pi\cdot73\cdot x)+\sin(2\cdot\pi\cdot83\cdot x)+\frac{\text{random}(100)}{1000}$

Применение корреляционной функции и преобразования фурье для исследования сигналов

Рассмотренные ранее преобразования крайне часто встречаются на практике для анализа различных сигналов – как детерминированных, так и стохастических.

Для анализа будем использовать сигнал, заданный выражением

```
def signal(x):
    return ((np.exp(9.24 * x * 1j) - np.exp((-1j) * 9.24 * x)) / (
        2 * 1j
    ))
```

На его основе создадим дискретный массив из 64 значений с шагом dx и наложенными на них случайными значениями из массива rnd и выведем .

```
SIGNAL_LEN = 64
STEP = 0.0312
rnd = np.random.randint(low=-1, high=1, size=SIGNAL_LEN)
x = np.arange(start=0, stop=SIGNAL_LEN) * STEP
y_1 = signal(x) + rnd
plot(x, y_1)
```

Рис. 5.21. Создание массива значений сигнала

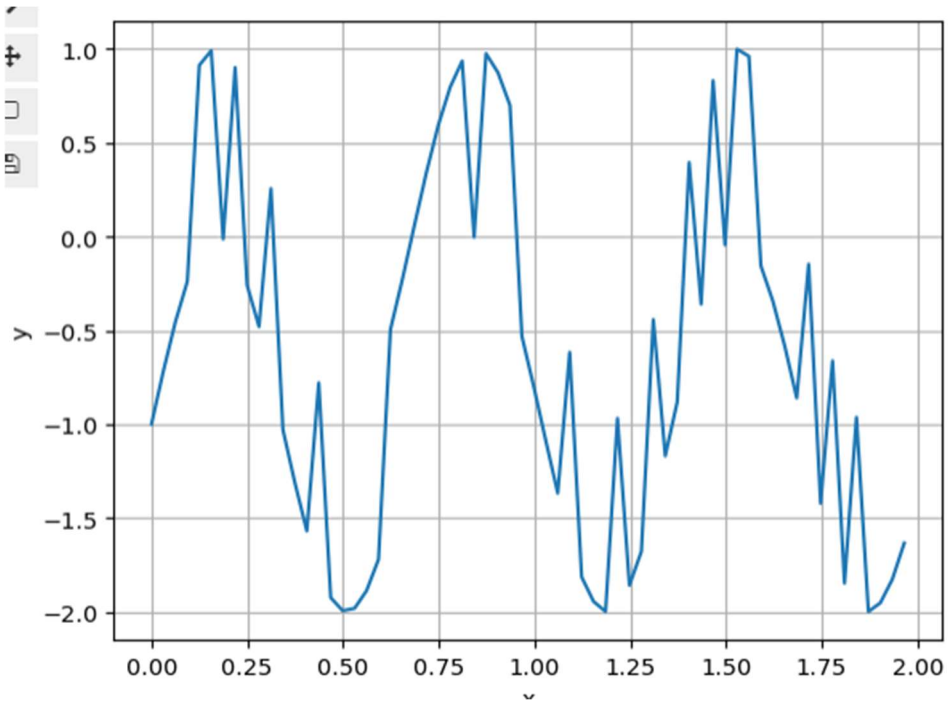

Рис. 5.22. График сигнала

При поиске полезной информации в такого рода сигналах (визуально идентичных шуму), чаще всего исходят из предположения, что полезная информация имеет периодический характер. Для выявления периодических составляющих сигнала следует воспользоваться АКФ, поскольку данное преобразование, за счет «сдвигания» сигнала относительно самого себя позволяет подчеркнуть повторяющиеся области.

Как было сказано в предыдущей работе, АКФ нельзя применять к сигналу, смещенному относительно оси x. Для устранения постоянной составляющей, вычтем из каждого значения массива y1 его математическое ожидание, сформировав массив y2.

```
1  mean = np.mean(y_1)
2  y_2 = y_1 - mean
```

Рис. 5.23. Устранение постоянной составляющей

Для формирования массива АКФ воспользуемся функцией corelate из модуля numpy после чего выведем график получившийся график.

```
1  auto_corelation_y_2 = np.correlate(y_2, y_2, mode='full')
2  auto_corelation_x = np.arange(start=-SIGNAL_LEN + 1, stop=SIGNAL_LEN)
3  plot(auto_corelation_x, auto_corelation_y_2)
```

Рис. 5.24. Создание АКФ и его отображение на графике

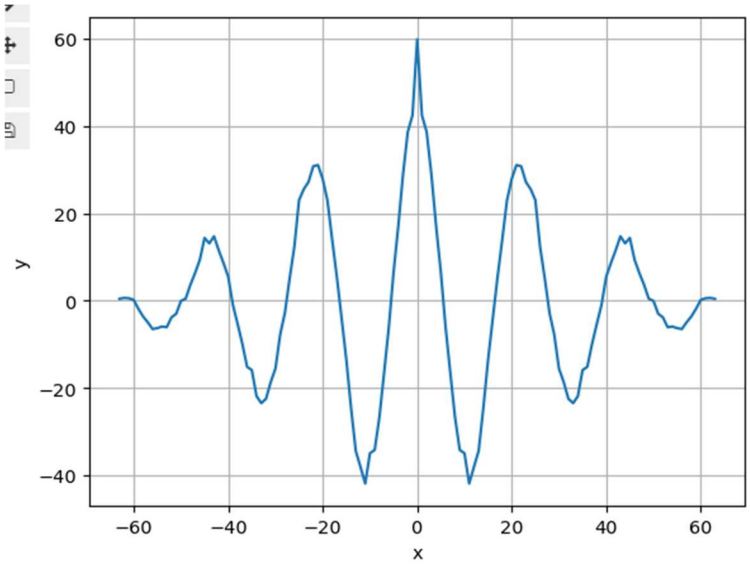

Рис. 5.24. График АКФ

На основе полученного графике можно сделать исследование. В общем случае, если уровень шума не чрезмерно высок, а в сигнале присутствует периодическая составляющая, то на графике АКФ различить ее будет очень просто. Если же уровень шума значительный (что может приводить к потере полезной информации), либо периодических составляющих в сигнале не содержится – график АКФ будет представлять собой затухающий по обе стороны от нуля импульс с заполнением реализацией случайной величины (затухающий от центра к краям шум). И только в нулевой точке будет наблюдаться пик, поскольку реализация случайного процесса значительно коррелирует сама с собой только при отсутствии сдвига. По графику АКФ ниже очевидно, что сигнал содержит периодическую полезную составляющую.

Выявив, имеется ли полезная составляющая в сигнале, следует определить ее свойства, при наличии. Поскольку в данной работе под полезной понимается периодическая составляющая, задача определения свойств сводится к вычислению характерной частоты сигнала. Характерной частотой (или характерными частотами) называют доминирующую частоту или группу частот, сопровождающих весь исследуемый процесс. В данном случае, для выявления характерной частоты лучше всего воспользоваться исследованием спектра как модуля ДПФ.

```
1  SIGNAL_SIDE = np.arange(0, (SIGNAL_LEN - 1) / 2)
2  fs = SIGNAL_SIDE * (1 / (SIGNAL_LEN * STEP))
3  fourier_array = np.abs(count_discrete_fourier_on(y_1))[:SIGNAL_SIDE.size]
```

Рис. 5.25. Массивы частоты и спектра по ДПФ

Нулевая частота есть постоянная составляющая, поэтому ее учитывать при выявлении характерной частоты не следует. Чтобы выявить характерную частоту достаточно найти максимум на графике (не соответствующий постоянной составляющей). Обычно характерная частота сильно, в два и более раз выделяется на фоне прочих, шумовых составляющих.

Построив график спектра, видим присутствие в нем характерной частоты между 0 и 4 Гц.

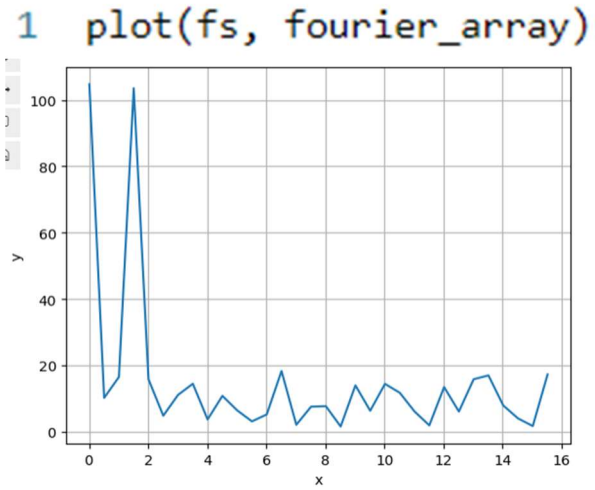

Рис. 5.26. Спектр сигнала

Приближаем график в данной области, для определения характерной частоты. Становится ясно, что она равна 1,5 Гц.

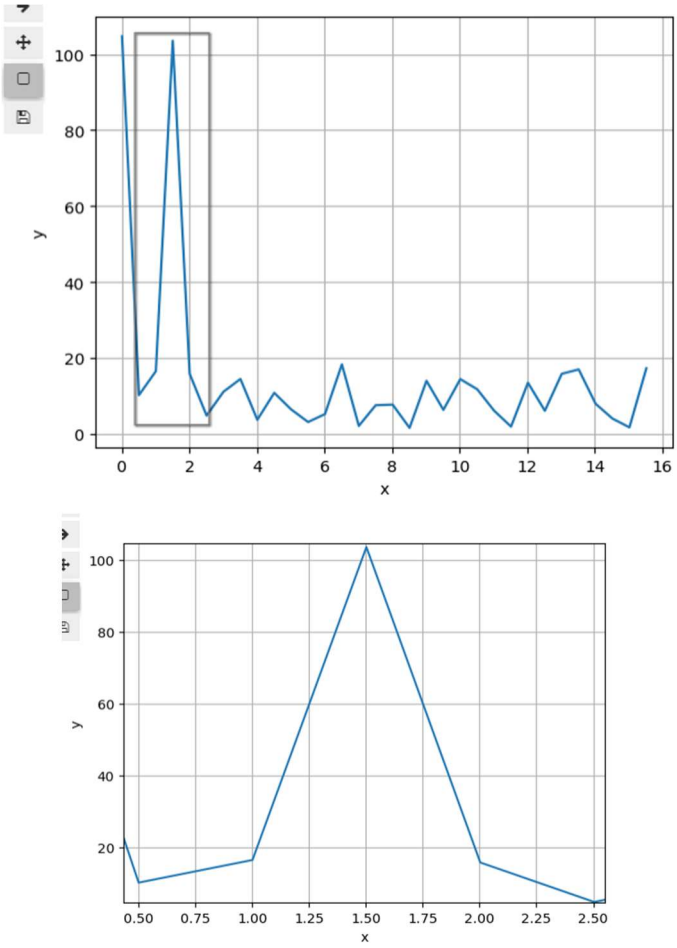

Рис. 5.27. Фрагмент спектра

Для иллюстрации результата задаем сигнал с такой частотой, $sin(1{,}5 \cdot 2\pi \cdot x)$, и построим его график вместе с графиком исходных данных.

```
1  sin_func = np.sin(1.5 * 2 * math.pi * x)
2  plot(x, y_1, x, sin_func)
```

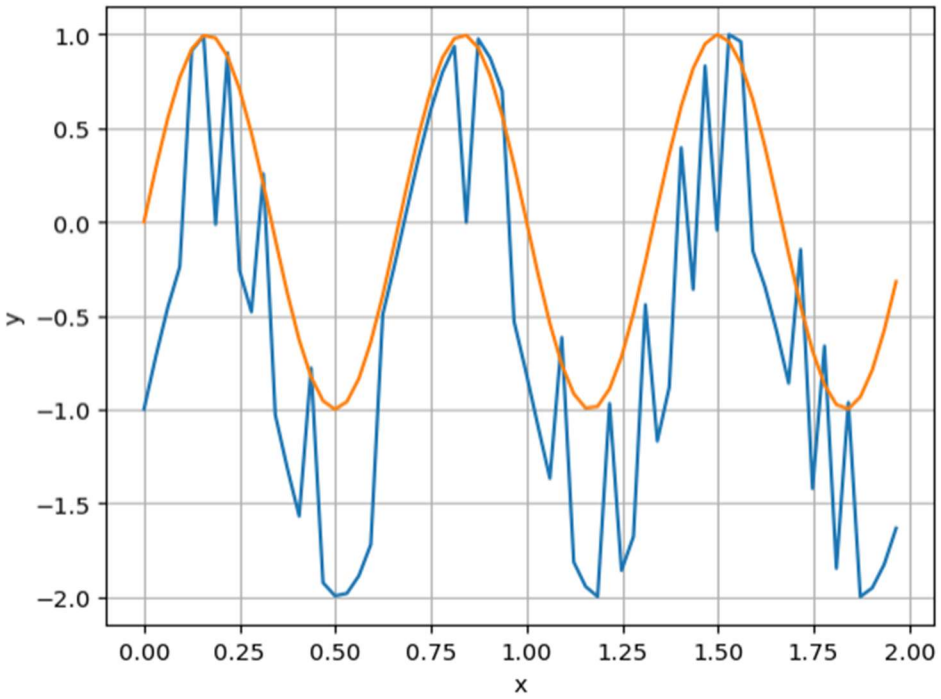

Рис. 5.28. Исследуемый сигнал с наложенной полезной составляющей

Задание 2

1. Задать указанные параметры из собственного варианта.
2. Построить график получившегося сигнала y1.
3. Выявить наличие/отсутствие периодической составляющей в заданном сигнале.
4. При наличии периодической составляющей определить значение характерной частоты для заданного сигнала.
5. Построить график сигнала с найденной характерной частотой вместе с графиком исходных данных.

Варианты ниже меняют вид выражения signal, а также параметр dx. Прочие переменные из примера выше остаются прежними.

Таблица 8.1

Варианты задания

Вариант	**signal(x)**	**dx**
1,6,11, 16,21,26	$\frac{\exp(i\cdot 25{,}1327\cdot x)-\exp(-i\cdot 25{,}1327\cdot x)}{2\cdot i}$	0.0117
2,7,12, 17,22,27	$\frac{\exp(i\cdot 31{,}4159\cdot x)-\exp(-i\cdot 31{,}4159\cdot x)}{2\cdot i}$	0.0094
3,8,13, 18,23,28	$\frac{\exp(i\cdot 6{,}2832\cdot x)-\exp(-i\cdot 6{,}2832\cdot x)}{2\cdot i}$	0.0469
4,9,14, 19,24,29	$\frac{\exp(i\cdot 18{,}8496\cdot x)-\exp(-i\cdot 18{,}8496\cdot x)}{2\cdot i}$	0.0156
5,10,15, 20,25,30	$\frac{\exp(i\cdot 12{,}5664\cdot x)-\exp(-i\cdot 12{,}5664\cdot x)}{2\cdot i}$	0.0234