Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Звіт

З лабораторної роботи №1

З курсу «Моделювання складних систем»

Варіант №4

Виконала:

Студентка групи ІПС-32

Челушкіна Валерія

Київ

2025

***Завдання.***

Визначити модель в класі функцій



для спостережуваної дискретної функції , 0.01, інтервал спостереження , 5.

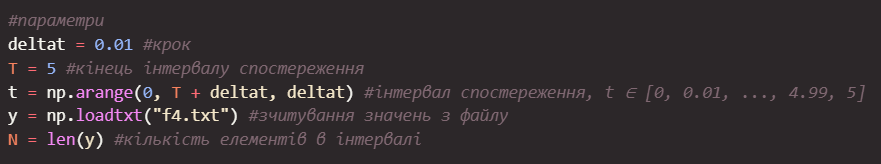
Дискретне перетворення Фур’є для дискретної послідовності 

.

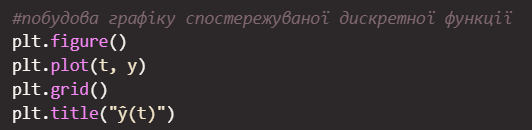
***Розв’язання.***

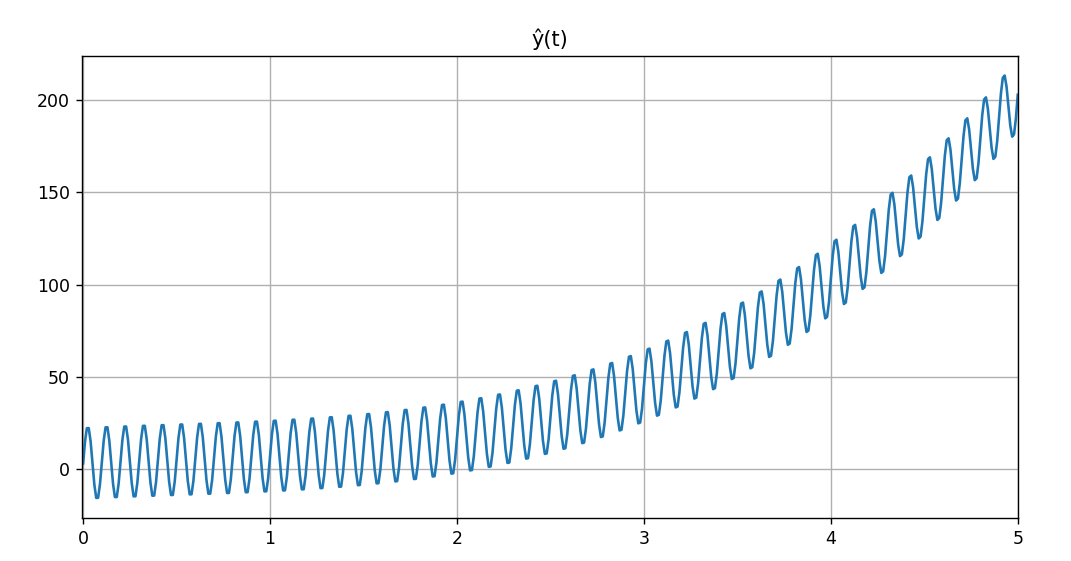
*\*Реалізовано мовою програмування Python.*

Почнемо з ініціалізації вже відомих нам змінних та зі зчитування даних із файлу. Маємо: інтервал спостереження t∈[0, T], T=5, крок ***∆****t* = 0,01 та N – кількість елементів в інтервалі спостереження.

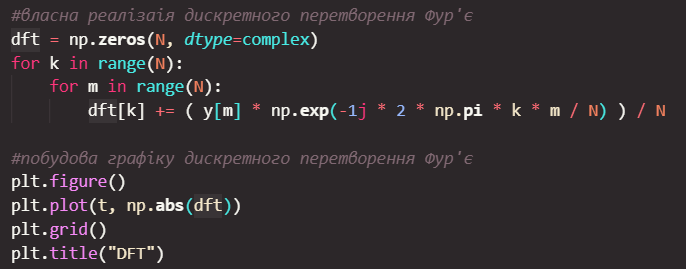


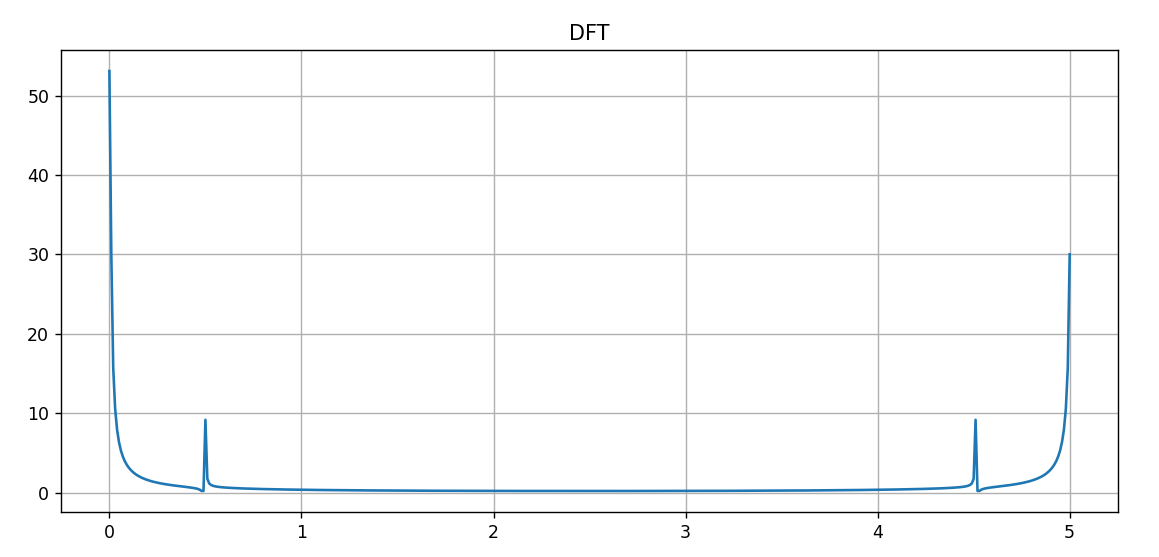
Побудуємо графік функції ŷ(t)





Реалізовуємо власноруч дискретне перетворення Фур’є у програмі за відомою формулою: . Побудуємо графік його модуля в залежності від часу t.





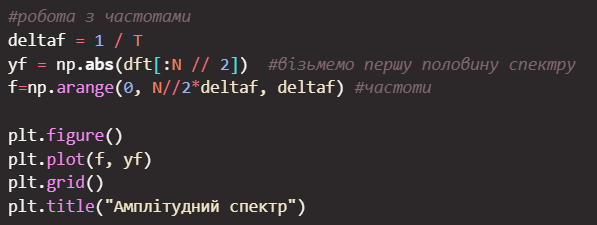
На графіку можна побачити, що функція майже симетрична.

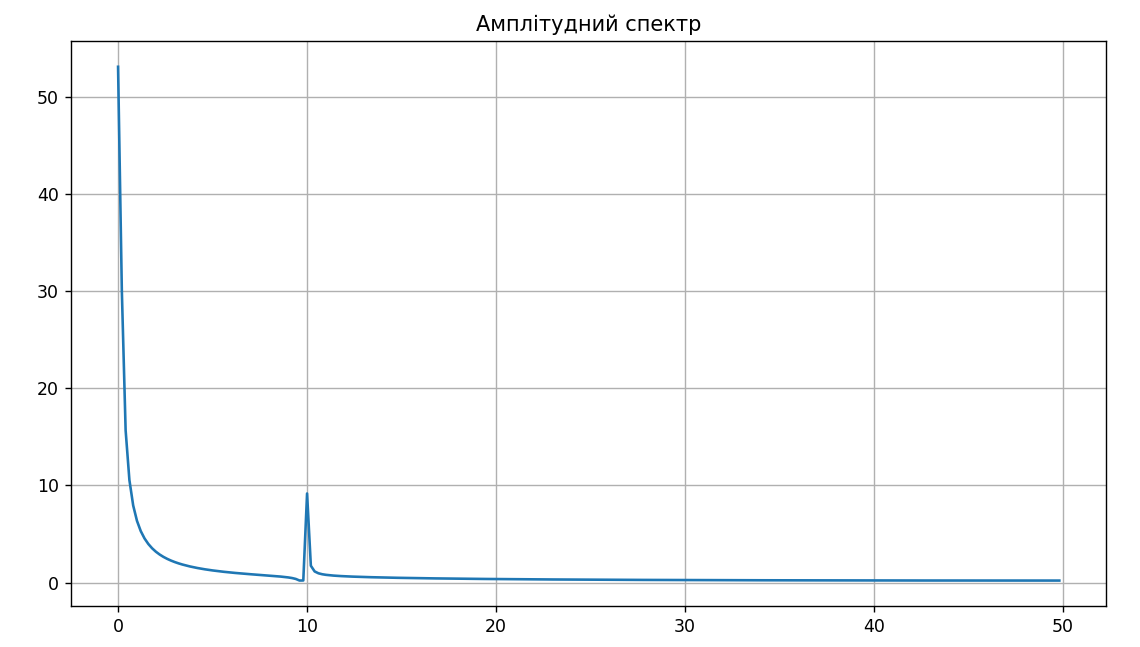
Знаходимо ∆f = 1/T.

Для того, щоб визначити локальні максимуми модуля перетворення Фур’є , k=0, 1, …, [N/2]-1, відкинемо праву половину функції та працюватимемо із лівою.

Знаходимо частоти , де

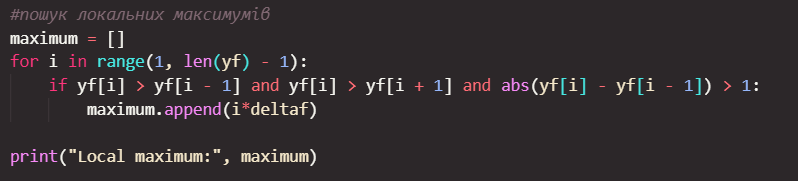
Побудуємо графік.





З графіку видно, найбільший вклад мають частоти 0 і 10. 0 можна відкинути, оскільки це є поліноміальний вклад. Тоді, маємо єдиний значущий вклад – 10гц: f=10.

В програмі це реалізовано таким чином:



Наша модель в загальному вигляді представлена так: 

Підставимо замість , *f*=10. Тепер вигляд моделі: .

Залишилось знайти інші параметри. Оскільки маємо лише один значущий вклад, то i=1 => k-3=1 => k=4.

Тобто потрібно знайти параметри => .

Для цього застосовуємо метод найменших квадратів. Для цього записуємо функцiонал похибки:

Параметри шукаємо з умови: .

Для цього записуємо систему рiвнянь:

Розв’яжемо для

=

Аналогічно: для вийде, що обидві частини потрібно буде множити на ,

Маємо формулу для системи лінійних рівнянь: Aa=y.

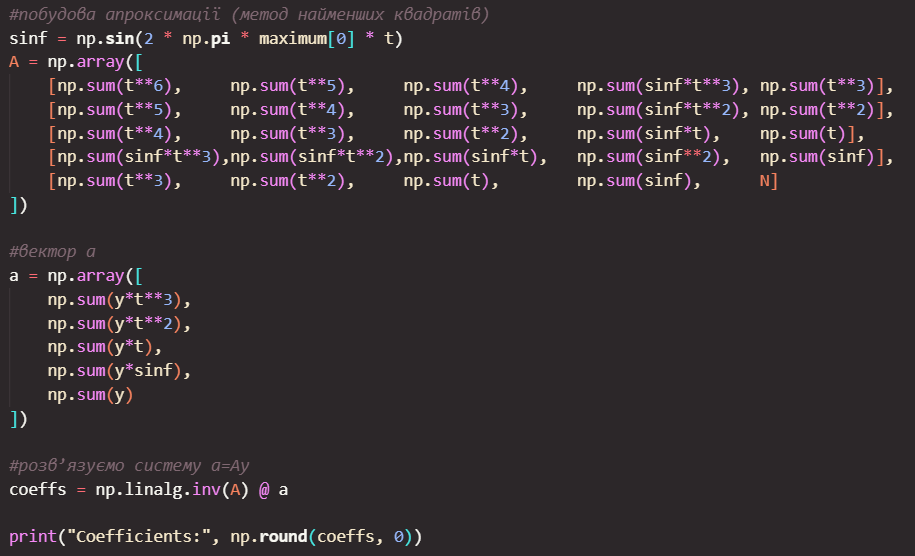
Зробимо заміну, нехай = g.

Тоді матриця A буде розміру 5х5 і матиме вигляд: , де в g – кожен окремий доданок стоїть у своєму стовпчику.

Вектор a= – вектор параметрів *a*, вектор y – розв’язки системи.

Звідси параметри *a* можна знайти: => => .

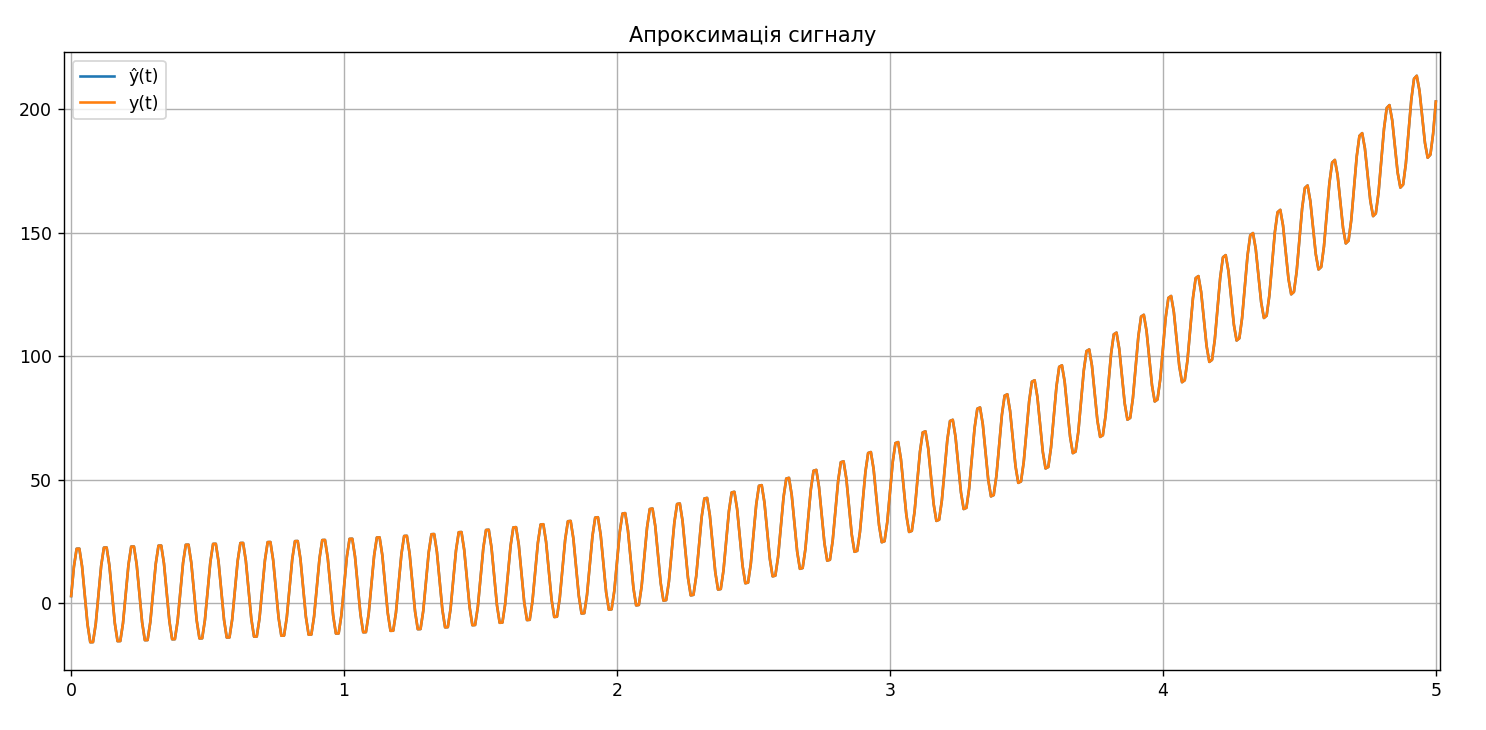
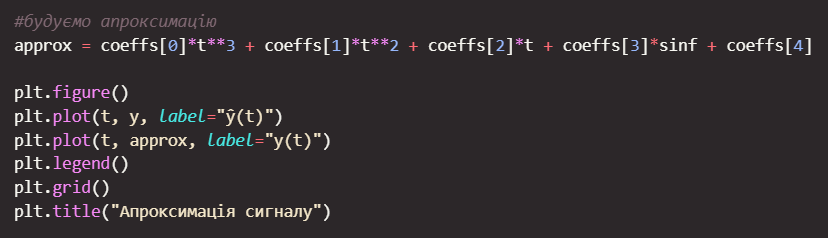
Реалізація в коді:



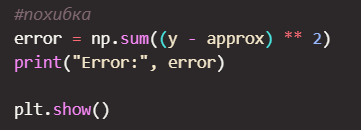
Coefficients: [ 2, -3, 5, 20, 3]

Підставимо у нашу модель і отримаємо.

Побудуємо тепер графіки y(t) і ŷ(t), і порівняємо.



Візуально обидва графіки співпали, отже можемо сказати, що всі розрахунки були проведені правильно. Але тепер ще знайдемо похибку.



Error: 4.502554073057709e-07.

Отже, модель при знайдених параметрах матиме вигляд:



***Повний код:***

import numpy as np

import matplotlib.pyplot as plt

*#параметри*

deltat = 0.01 *#крок*

T = 5 *#кінець інтервалу спостереження*

t = np.arange(0, T + deltat, deltat) *#інтервал спостереження, t ∈ [0, 0.01, ..., 4.99, 5]*

y = np.loadtxt(" f4.txt") *#зчитування значень з файлу*

N = len(y) *#кількість елементів в інтервалі*

*#побудова графіку спостережуваної дискретної функції*

plt.figure()

plt.plot(t, y)

plt.grid()

plt.title("ŷ(t)")

*#власна реалізаія дискретного перетворення Фур'є*

dft = np.**zeros**(N, *dtype*=complex)

for k in range(N):

    for m in range(N):

        dft[k] += ( y[m] \* np.**exp**(-1j \* 2 \* np.**pi** \* k \* m / N) ) / N

*#побудова графіку дискретного перетворення Фур'є*

plt.figure()

plt.plot(t, np.**abs**(dft))

plt.grid()

plt.title("DFT")

*#робота з частотами*

deltaf = 1 / T

yf = np.**abs**(dft[:N // 2])  *#візьмемо першу половину спектру*

f=np.arange(0, N//2\*deltaf, deltaf) *#частоти*

plt.figure()

plt.plot(f, yf)

plt.grid()

plt.title("Амплітудний спектр")

*#пошук локальних максимумів*

maximum = []

for i in range(1, len(yf) - 1):

    if yf[i] > yf[i - 1] and yf[i] > yf[i + 1] and abs(yf[i] - yf[i - 1]) > 1:

        maximum.append(i\*deltaf)

print("Local maximum:", maximum)

*#побудова апроксимації (метод найменших квадратів)*

sinf = np.**sin**(2 \* np.**pi** \* maximum[0] \* t)

A = np.array([

    [np.sum(t\*\*6),     np.sum(t\*\*5),     np.sum(t\*\*4),     np.sum(sinf\*t\*\*3), np.sum(t\*\*3)],

    [np.sum(t\*\*5),     np.sum(t\*\*4),     np.sum(t\*\*3),     np.sum(sinf\*t\*\*2), np.sum(t\*\*2)],

    [np.sum(t\*\*4),     np.sum(t\*\*3),     np.sum(t\*\*2),     np.sum(sinf\*t),    np.sum(t)],

    [np.sum(sinf\*t\*\*3),np.sum(sinf\*t\*\*2),np.sum(sinf\*t),   np.sum(sinf\*\*2),   np.sum(sinf)],

    [np.sum(t\*\*3),     np.sum(t\*\*2),     np.sum(t),        np.sum(sinf),      N]

])

*#вектор a*

a = np.array([

    np.sum(y\*t\*\*3),

    np.sum(y\*t\*\*2),

    np.sum(y\*t),

    np.sum(y\*sinf),

    np.sum(y)

])

*#розв’язуємо систему a=Ay*

coeffs = np.linalg.inv(A) @ a

print("Coefficients:", np.**round**(coeffs, 0))

*#будуємо апроксимацію*

approx = coeffs[0]\*t\*\*3 + coeffs[1]\*t\*\*2 + coeffs[2]\*t + coeffs[3]\*sinf + coeffs[4]

plt.figure()

plt.plot(t, y, *label*="ŷ(t)")

plt.plot(t, approx, *label*="y(t)")

plt.legend()

plt.grid()

plt.title("Апроксимація сигналу")

*#похибка*

error = np.sum((y - approx) \*\* 2)

print("Error:", error)

plt.show()

***Результати:***

Local maximum: [10.0]

Coefficients: [ 2. -3.  5. 20.  3.]

Error: 4.502554073057709e-07