Київський національний університет імені Тараса Шевченка
Факультет комп`ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем
Моделювання складних систем

Звіт
з лабораторної роботи №1
студентки 3-го курсу
групи ІПС-31
Ляшенка Матвія Олексійовича
Варіант №14

Київ
2025

Хід роботи

У ході роботи передбачається створення математичної моделі, яка буде апроксимувати задану дискретну функцію. Модель розглядається як сума полінома третього ступеня та кількох синусоїдальних складових. Частоти цих синусоїд визначаються за допомогою дискретного перетворення Фур'є (ДПФ), після чого невідомі параметри моделі обчислюються методом найменших квадратів. Реалізація всіх етапів виконується у середовищі MATLAB.

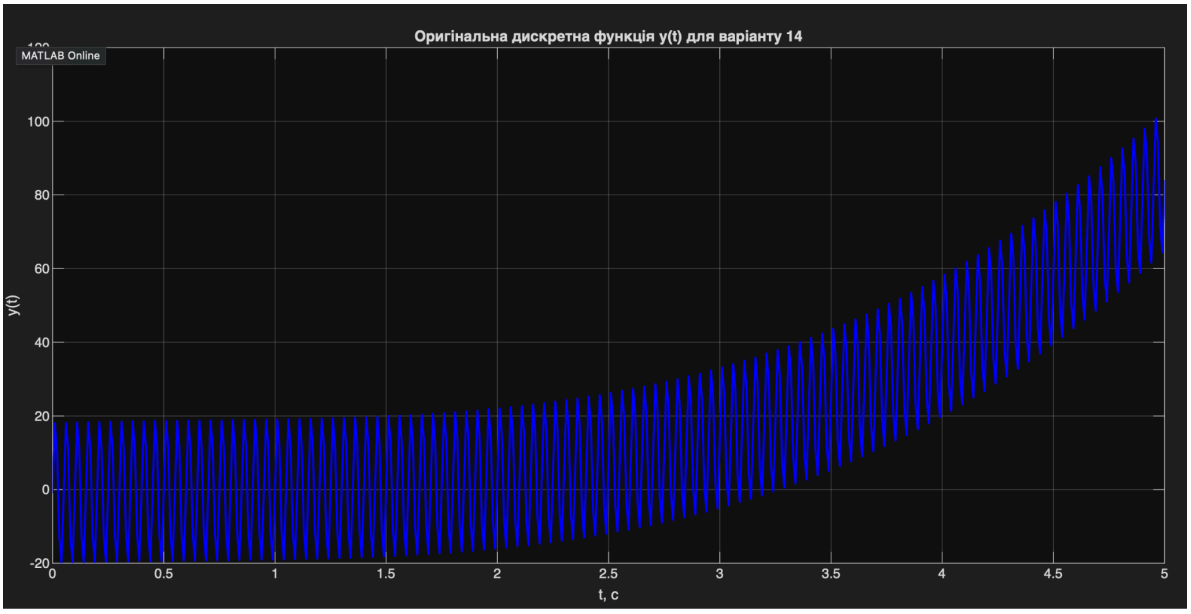
1 . Завантаження та візуалізація вихідних даних

На першому етапі було ініціалізовано початкові параметри: крок дискретизації dt=0.01, інтервал спостереження T=5 секунд та відповідний вектор часу t. Далі з файлу *f14.txt* було завантажено масив спостережуваних значень y(t).

```
1   dt = 0.01;
2   T = 5;
3   t = 0:dt:T;
4   N = length(t);
5
6   y = dlmread('f14.txt',' ');
```

Для візуального аналізу поведінки сигналу було побудовано графік вихідної функції.

```
8   figure;
9   plot(t, y, 'b-', 'LineWidth', 1.5);
10  grid on;
11  title('Оригінальна дискретна функція y(t) для варіанту 14');
12  xlabel('t, c');
13  ylabel('y(t)');
```



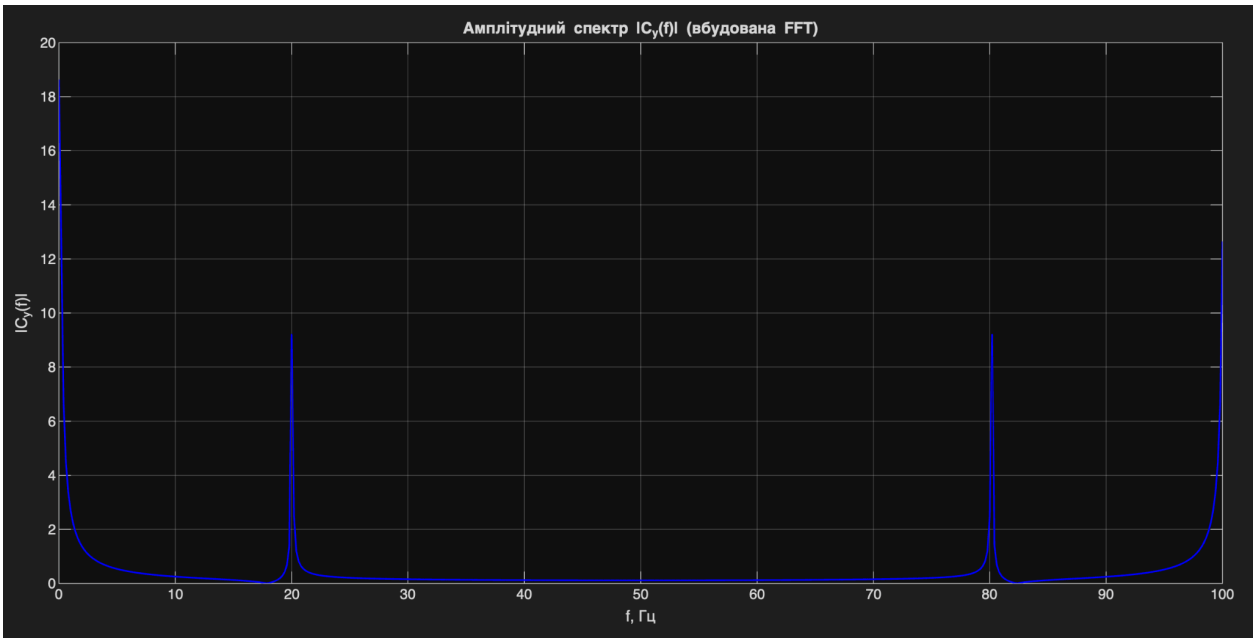
Маємо дискретну функцію, яка поєднує поліноміальний тренд із вираженим високочастотним гармонійним складником.

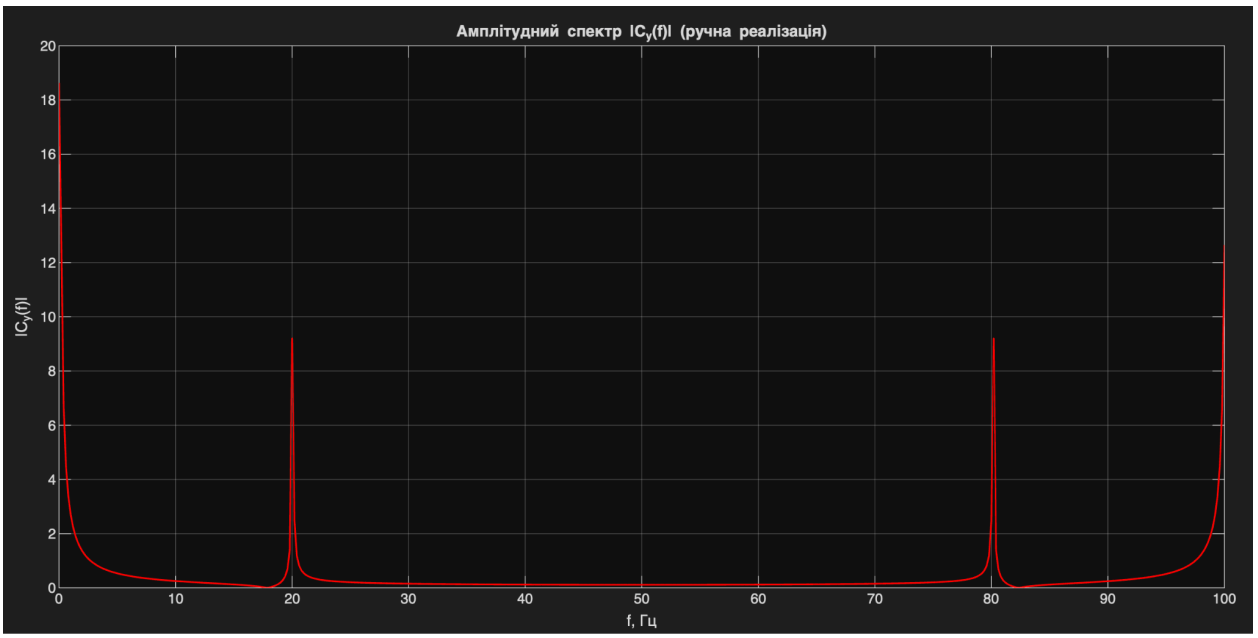
2. Частотний аналіз сигналу

Для визначення гармонійних складових спостережуваного сигналу було проведено дискретне перетворення Фур'є. Щоб переконатися у правильності спектрального аналізу, ДПФ було виконано у двох формах:

1. із використанням вбудованої функції MATLAB **fft**,
2. а також шляхом власної реалізації дискретного перетворення Фур'є у вигляді подвійного циклу.

```
%2
% --- FFT (вбудована функція)
fft_y_raw = fft(y);                % необроблений спектр
fft_y = fft_y_raw / N;             % нормалізація

freq_axis = (0:N-1) / T;           % частотна вісь

figure;
plot(freq_axis, abs(fft_y), 'b-', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (вбудована FFT)');
xlabel('f, Гц');
ylabel('|C_y(f)|');

% --- Ручна реалізація ДПФ
fft_manual = zeros(1, N);
for k = 1:N
    for m = 1:N
        fft_manual(k) = fft_manual(k) + y(m) * exp(-1i * 2*pi*(k-1)*(m-1)/N);
    end
end
fft_manual = fft_manual / N;       % нормалізація

figure;
plot(freq_axis, abs(fft_manual), 'r-', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (ручна реалізація)');
xlabel('f, Гц');
ylabel('|C_y(f)|');
```



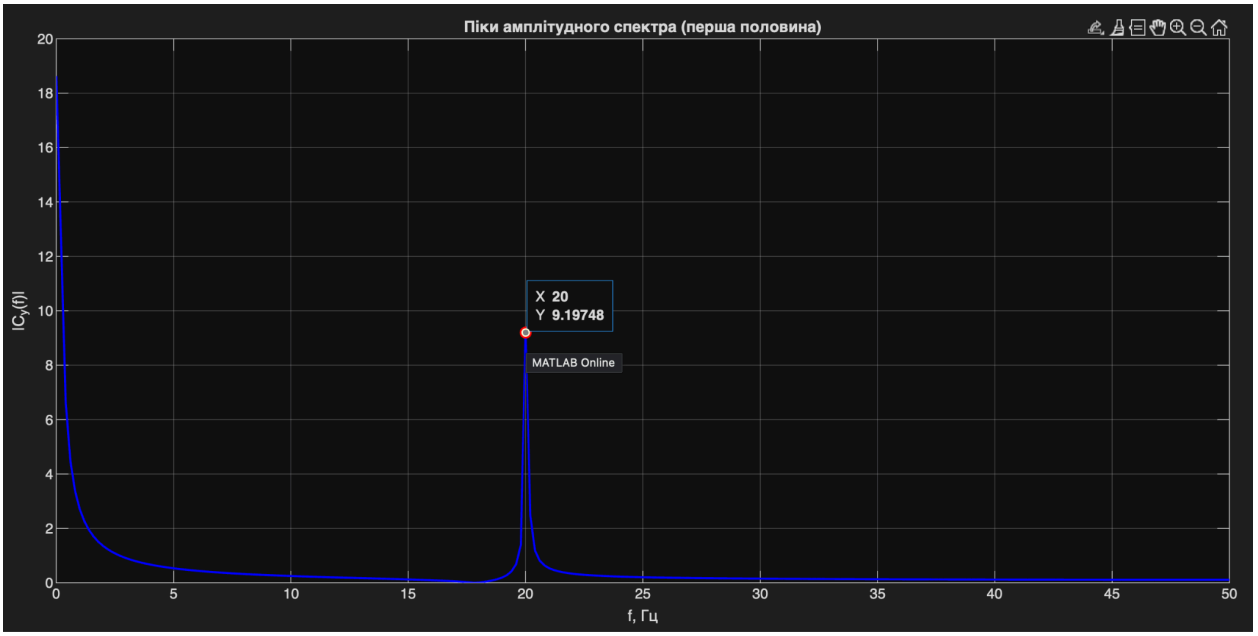


В обох випадках спектральні коефіцієнти було нормалізовано шляхом поділу на кількість точок N, що забезпечило коректне відображення амплітуд. Отримані амплітудні спектри для вбудованого та ручного обчислення повністю збіглися, що підтвердило правильність реалізації ДПФ та достовірність подальшого частотного аналізу.

3. Визначення значущих частот

Оскільки спектр дискретного перетворення Фур'є є симетричним, для подальшого аналізу було використано лише його першу половину. На цій ділянці за допомогою функції **findpeaks** було виконано пошук локальних максимумів, що відповідають гармонійним складовим із найбільшим внеском у сигнал. Індекс виявленого піка було перетворено у фізичну частоту (у герцах), яку надалі використано як частотний параметр синусоїдальної компоненти моделі.

```
47
48  %3
49  % --- Перша половина спектра
50  fft_half_raw = abs(fft_y_raw(1:round(N/2)));
51
52  % --- Масив частот
53  df = 1 / T;
54  f = 0:df:(N-1)*df;
55
56  % --- Пошук піків
57  [maxima, locs] = findpeaks(fft_half_raw, 'MinPeakHeight', 5);
58
59  % --- Графік з піками
60  figure;
61  plot(f(1:round(N/2)), abs(fft_y(1:round(N/2))), 'b-', 'LineWidth', 1.5);
62  grid on;
63  hold on;
64  plot(f(locs), maxima / N, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');
65  hold off;
66  title('Піки амплітудного спектра (перша половина)');
67  xlabel('f, Гц');
68  ylabel('|C_y(f)|');
69
70  % --- Вивід частот
71  disp('--- Знайдені пікові частоти ---');
72  for i = 1:length(locs)
73      fprintf('Пік %d: частота %.4f Гц, амплітуда %.4f\n', ...
74          i, f(locs(i)), maxima(i)/N);
75  end
76
```



На отриманому спектрі спостерігається виражений локальний максимум, який відповідає основній гармонійній складовій сигналу. Єдиний значущий пік розташований на частоті:

- $f_1 \approx 20$ **Гц**

Ця частота використовується як базова синусоїдальна компонента у подальшому побудуванні моделі.

```
>> main
--- Знайдені пікові частоти ---
Пік 1: частота 20.0000 Гц, амплітуда 9.1975
```

4. Побудова моделі методом найменших квадратів

Для знаходження невідомих коефіцієнтів апроксимуючої функції було застосовано метод найменших квадратів. Модель шукалася у вигляді:

$$y(t)=a1t3+a2t2+a3t+a4\sin(2\pi\cdot 20\,t)+a5.$$

Для цього була сформована матриця системи A, стовпцями якої є вектори базових функцій
(t3, t2, t, sin(2π·20 t), 1), а також вектор спостережень yy.
Отримана система лінійних алгебраїчних рівнянь $A\cdot a=y$ була розв'язана відносно вектора коефіцієнтів ***a*** за допомогою оператора \ у MATLAB.

```
%4
% --- Частота синусоїди
f1 = 20;

% --- Формування матриці системи
s1 = sin(2*pi*f1*t);
A = [t'.^3, t'.^2, t', s1', ones(N,1)];

% --- Розв'язання A*a = y
coeff = A \ y';

disp('--- Знайдені коефіцієнти ---');
fprintf('a1 (t^3): %.6f\n', coeff(1));
fprintf('a2 (t^2): %.6f\n', coeff(2));
fprintf('a3 (t):   %.6f\n', coeff(3));
fprintf('a4*sin:   %.6f\n', coeff(4));
fprintf('a5 const: %.6f\n', coeff(5));

% --- Побудова моделі
y_model = coeff(1)*t.^3 + coeff(2)*t.^2 + coeff(3)*t + coeff(4)*sin(2*pi*f1*t) + coeff(5);

% --- Похибки
SSE = sum((y_model - y).^2);
MSE = SSE / N;
rel_err = sqrt(SSE / sum(y.^2)) * 100;

disp('--- Похибки моделі ---');
fprintf('SSE: %.6f\n', SSE);
fprintf('MSE: %.6f\n', MSE);
fprintf('Відносна похибка: %.4f %%\n', rel_err);

% --- Графік порівняння
figure;
plot(t, y, 'b-', 'LineWidth', 1.5);
hold on;
plot(t, y_model, 'r--', 'LineWidth', 2);
grid on;
legend('Оригінал', 'Модель');
title('Порівняння y(t) та апроксимації y_{model}(t)');
xlabel('t, c');
ylabel('y');
```

```
--- Знайдені коефіцієнти ---
a1 (t^3): 1.000001
a2 (t^2): -2.000006
a3 (t):   2.000012
a4*sin:   20.000000
a5 const: -1.000006
--- Похибки моделі ---
SSE: 0.000000
MSE: 0.000000
Відносна похибка: 0.0001 %
>>
```

Апроксимуюча функція:

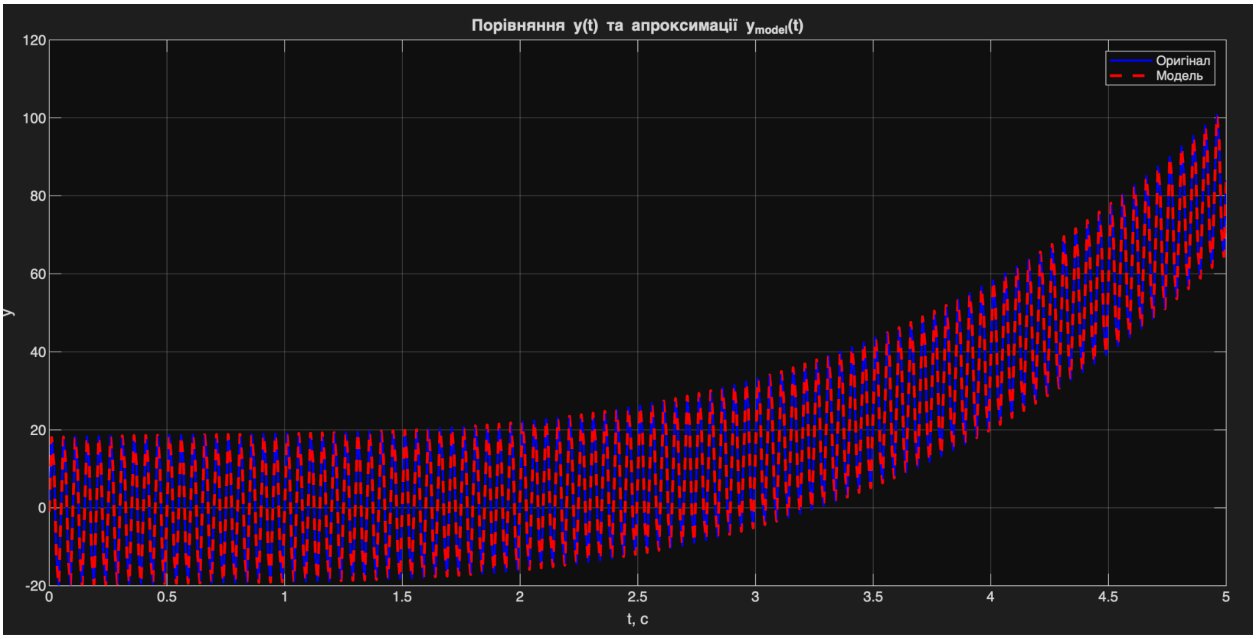
$y(t)=1.000001t3-2.000006t2+2.000012t+20\sin(40\pi t)-1.000006$

Основна частота сигналу:

$f*=20$ Гц

Кінцева похибка моделі:

- SSE = 0
- MSE = 0
- Відносна похибка = 0.0001%



Повний код:

```
dt = 0.01;
T = 5;
t = 0:dt:T;
N = length(t);
y = dlmread('f14.txt',' ');
%1
figure;
plot(t, y, 'b-', 'LineWidth', 1.5);
grid on;
title('Оригінальна дискретна функція y(t) для варіанту 14');
xlabel('t, c');
ylabel('y(t)');
%2
% --- FFT (вбудована функція)
fft_y_raw = fft(y);                % необроблений спектр
fft_y = fft_y_raw / N;             % нормалізація
freq_axis = (0:N-1) / T;           % частотна вісь
figure;
plot(freq_axis, abs(fft_y), 'b-', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (вбудована FFT)');
xlabel('f, Гц');
ylabel('|C_y(f)|');
% --- Ручна реалізація ДПФ
fft_manual = zeros(1, N);
for k = 1:N
   for m = 1:N
       fft_manual(k) = fft_manual(k) + y(m) * exp(-1i * 2*pi*(k-1)*(m-1)/N);
   end
end
fft_manual = fft_manual / N;       % нормалізація
figure;
plot(freq_axis, abs(fft_manual), 'r-', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (ручна реалізація)');
xlabel('f, Гц');
ylabel('|C_y(f)|');
%3
% --- Перша половина спектра
fft_half_raw = abs(fft_y_raw(1:round(N/2)));
% --- Масив частот
df = 1 / T;
f = 0:df:(N-1)*df;
% --- Пошук піків
[maxima, locs] = findpeaks(fft_half_raw, 'MinPeakHeight', 5);
% --- Графік з піками
figure;
plot(f(1:round(N/2)), abs(fft_y(1:round(N/2))), 'b-', 'LineWidth', 1.5);
grid on;
hold on;
plot(f(locs), maxima / N, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');
hold off;
title('Піки амплітудного спектра (перша половина)');
xlabel('f, Гц');
ylabel('|C_y(f)|');
% --- Вивід частот
disp('--- Знайдені пікові частоти ---');
for i = 1:length(locs)
   fprintf('Пік %d: частота %.4f Гц, амплітуда %.4f\n', ...
       i, f(locs(i)), maxima(i)/N);
end
%4
% --- Частота синусоїди
f1 = 20;
% --- Формування матриці системи
s1 = sin(2*pi*f1*t);
A = [t'.^3, t'.^2, t', s1', ones(N,1)];
% --- Розв'язання A*a = y
coeff = A \ y';
disp('--- Знайдені коефіцієнти ---');
fprintf('a1 (t^3): %.6f\n', coeff(1));
fprintf('a2 (t^2): %.6f\n', coeff(2));
fprintf('a3 (t):   %.6f\n', coeff(3));
fprintf('a4*sin:   %.6f\n', coeff(4));
fprintf('a5 const: %.6f\n', coeff(5));
% --- Побудова моделі
y_model = coeff(1)*t.^3 + coeff(2)*t.^2 + coeff(3)*t + coeff(4)*sin(2*pi*f1*t) + coeff(5);
% --- Похибки
SSE = sum((y_model - y).^2);
MSE = SSE / N;
rel_err = sqrt(SSE / sum(y.^2)) * 100;
```

```
disp('--- Похибки моделі ---');
fprintf('SSE: %.6f\n', SSE);
fprintf('MSE: %.6f\n', MSE);
fprintf('Відносна похибка: %.4f %%\n', rel_err);
% --- Графік порівняння
figure;
plot(t, y, 'b-', 'LineWidth', 1.5);
hold on;
plot(t, y_model, 'r--', 'LineWidth', 2);
grid on;
legend('Оригінал', 'Модель');
title('Порівняння y(t) та апроксимації y_{model}(t)');
xlabel('t, c');
ylabel('y');
```