Київський національний університет імені Тараса Шевченка

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Моделювання складних систем

Звіт

з лабораторної роботи №1

студента 3-го курсу

групи ІПС-31

Іванченка Владислава Віталійовича

Варіант №13

Київ

2025

Постановка задачі

Нехай є дискретна функція $\hat{y}(t_{i})$, $i=1,2,...,N$, що подана у вигляді значень у і-й момент часу.

Потрібно визначити модель в класі функцій

$y(t)=a_{1}t^{3}+a_{2}t^{2}+a_{3}t+\sum_{i=4}^{k}a_{i}sin(2\pi f_{i-3}t)+a_{k+1}$

для спостережуваної дискретної функції $\tilde{y}(t_{i})$, $i=1,2,...,N$, (відповідний файл f13.txt), $t_{i+1}-t_{i}=\Delta t=0.01$, інтервал спостереження [0, T], $T=5$.

Дискретне

перетворення

$x(i)$, $i=0,1,2,...,N-1$

Фур'є для дискретної послідовності

$c_{x}(k)=\frac{1}{N}\sum_{m=0}^{N-1}x(m)e^{-i2\pi im/N}$

Далі потрібно визначити за модулем дискретного перетворення Фур'є (далі модуль ДПФ) частоти із найбільшим вкладом.

Для цього визначаємо момент, у який модуль ДПФ приймає найбільше значення, тобто є локальним екстремумом. Позначимо такі моменти через k^{*}. Щоб знайти саме частоти із найбільшим вкладом, які позначимо через f^{*} потрібно виконати множення $f^{*}=k^{*}f=k^{*}/T.$

Знайшовши частоти із найбільшим вкладом, можна приступати до безпосереднього визначення невідомих параметрів a, $i=k+I$. Для їх

визначення застосовуємо метод найменших квадратів. Для цього записуємо функціонал похибки:

$F(a_{1},a_{2},...,a_{k+1})=\frac{1}{2}\sum_{j=0}^{N-1}(a_{1}t_{j}^{3}+a_{2}t_{j}^{2}+a_{3}t_{j}+\sum_{i=4}^{k}a_{i}sin(2\pi f_{i-3}t_{j})+a_{k+1}-\hat{y}(t_{j}))^{2}$

Параметри a, $i=1,2,...,k+1$ шукаємо з умови:

$F(a_{1},a_{2},...,a_{k+1})\rightarrow min_{a_{1},a_{2},...,a_{k+1}}$

Для цього записуємо систему рівнянь:

$\frac{\partial F(a_{1},a_{2},...,a_{k+1})}{\partial a_{j}}=0$

Ця система є системою лінійних алгебраїчних рівнянь. Розв'язавши цю систему

одним з відомих методів, знаходимо a, $i=1,2,...,k+1$.

Хід роботи

Програмна реалізація виконана мовою MATLAB.

Метою даної лабораторної роботи є побудова математичної моделі для апроксимації спостережуваної дискретної функції.

Модель шукається у класі функцій, що є сумою полінома третього ступеня та синусоїдальних компонент.

Для визначення частот синусоїд використовується дискретне перетворення Фур'є (ДПФ), а для знаходження невідомих коефіцієнтів моделі метод найменших квадратів (МНК).

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 г.

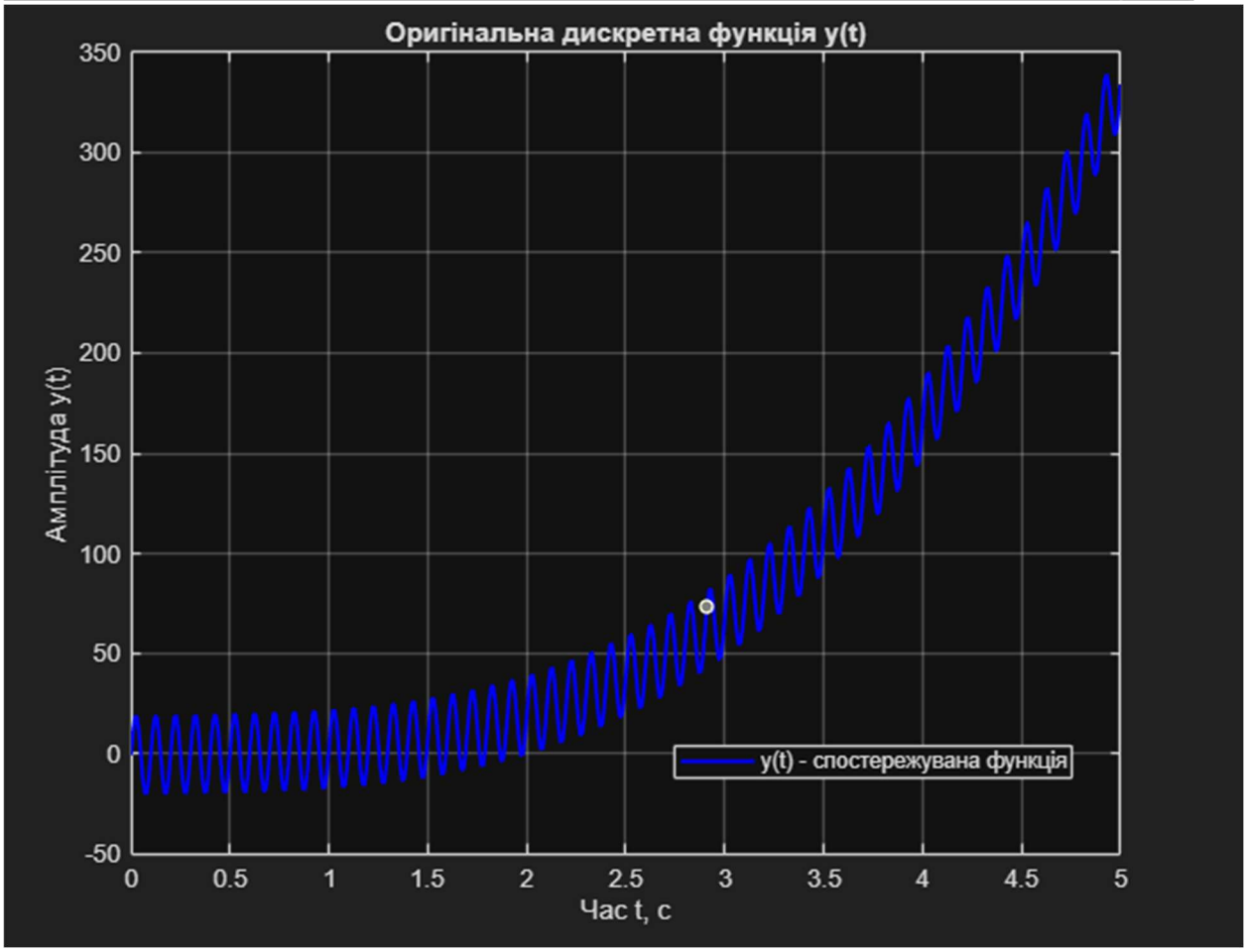
```
%% 1. ІНІЦІАЛІЗАЦІЯ ПАРАМЕТРІВ
clear all;
clc;
close all;

% Крок дискретизації та інтервал спостереження
dt = 0.01;        % Крок дискретизації
T = 5;            % Інтервал спостереження
t = 0:dt:T;       % Масив часових точок
N = length(t);    % Кількість точок
```

Далі з файлу **f13.txt** було завантажено масив спостережуваних значень $y(t)$.

```
%% 2. ЗАВАНТАЖЕННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДАНИХ

y = dlmread('f13.txt', '');
```

Для візуального аналізу поведінки сигналу було побудовано графік вихідної функції.

```
%% 3. ВІЗУАЛІЗАЦІЯ ОРИГІНАЛЬНОЇ ФУНКЦІЇ
figure;
plot(t, y, 'b', 'LineWidth', 1.5, 'DisplayName', 'y(t) - спостережувана функція')
grid on;
title('Оригінальна дискретна функція y(t)');
xlabel('Час t, с');
ylabel('Амплітуда y(t)');
legend('Location', 'best');
```



Маємо поліноміальну функцію із частотними вкладеннями у вигляді коливань.

2. Частотний аналіз сигналу

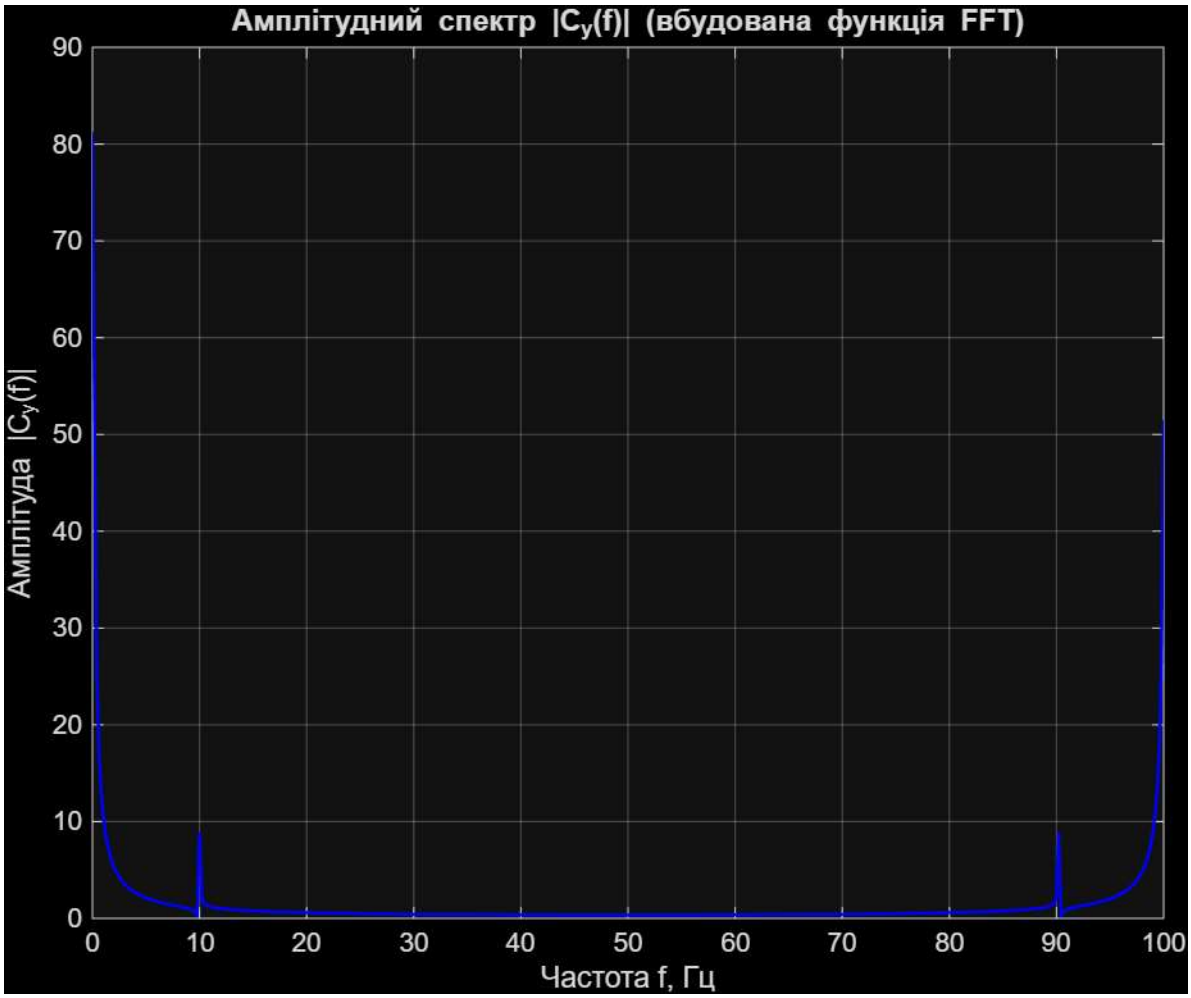
Для ідентифікації основних гармонійних складових сигналу було застосовано дискретне перетворення Фур'є.

З метою перевірки коректності обчислень, ДПФ було виконано двома способами:

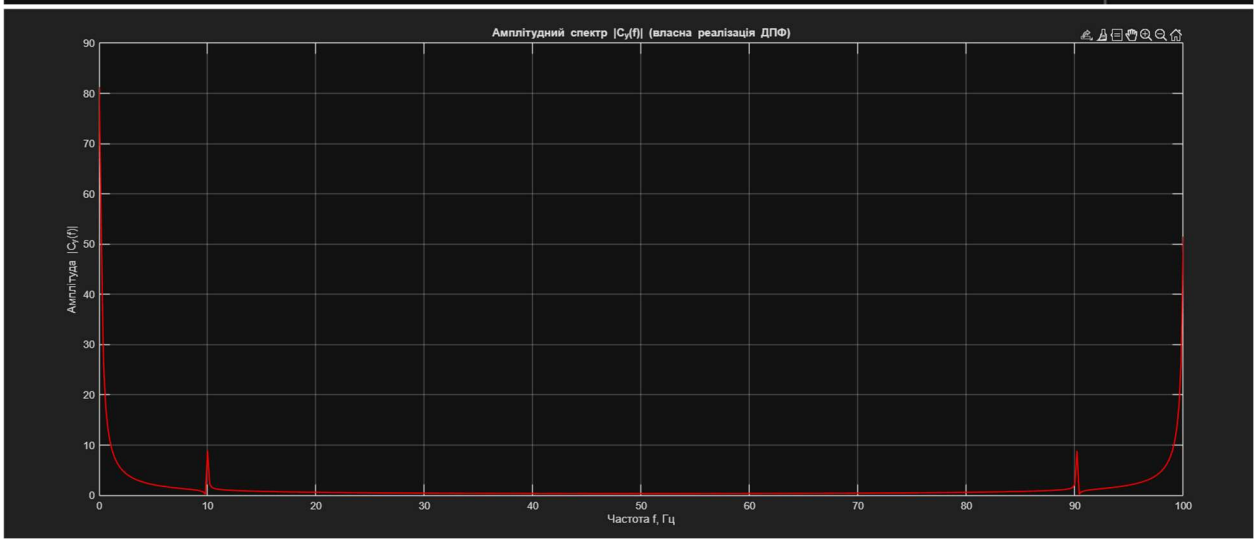
1. За допомогою вбудованої функції MATLAB fft.
2. Шляхом програмної реалізації алгоритму ДПФ вручну.

```
%% 4. ДИСКРЕТНЕ ПЕРЕТВОРЕННЯ ФУР'Є (вбудована функція)
fft_y_raw = fft(y);       % БЕЗ нормалізації (для пошуку піків)
fft_y = fft_y_raw / N;  % З нормалізацією (для відображення)

% Побудова амплітудного спектру (FFT)
figure;
freq_axis = (0:N-1) / T; % Частотна вісь в Гц
plot(freq_axis, abs(fft_y), 'b', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (вбудована функція FFT)');
xlabel('Частота f, Гц');
ylabel('Амплітуда |C_y(f)|');
```

[СЮДИ



```
%% 5. ВЛАСНА РЕАЛІЗАЦІЯ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ФУР'Є
% (Для перевірки коректності)
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 результату

% Побудова амплітудного спектру (власна реалізація)
figure;
plot(freq_axis, abs(fft_manual), 'r', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (власна реалізація ДПФ)');
xlabel('Частота f, Гц');
ylabel('Амплітуда |C_y(f)|');
```



Графіки амплітудних спектрів, отримані обома методами, виявились ідентичними, що підтвердило правильність реалізації.

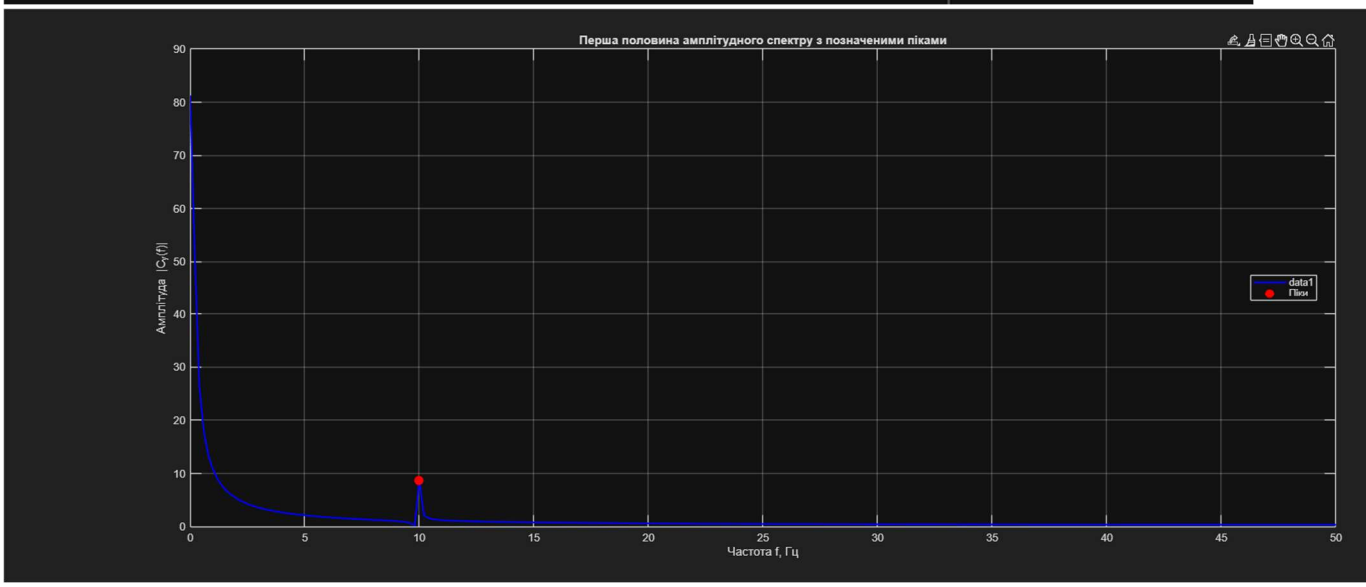
3. Визначення значущих частот

Оскільки спектр ДПФ є симетричним, для подальшого аналізу

використовувалась лише його перша половина. На цьому спектрі за допомогою функції findpeaks було здійснено пошук локальних максимумів ("піків"), які відповідають частотам, що роблять найбільший внесок у вихідний сигнал.

Індекси знайдених піків були перераховані у фізичні частоти (в Гц) для подальшого використання в моделі.

```
%% 6. АНАЛІЗ ЧАСТОТНОГО СПЕКТРУ
% Визначення частотних компонент
df = 1 / T; % Частотне розділення
f = 0:df:(N-1)*df; % Частотна вісь

% Беремо тільки першу половину спектру (оскільки він симетричний)
fft_half_raw = abs(fft_y_raw(1:round(N/2))); % Перша половина БЕЗ нормалізації
fft_half_norm = abs(fft_y(1:round(N/2)));   % Перша половина З нормалізацією

% Пошук піків (екстремумів) у спектрі
% 'MinPeakHeight' - фільтр, щоб відсіяти малий шум. Можливо, доведеться змінити це значення.
[maxima, locs] = findpeaks(fft_half_raw, 'MinPeakHeight', 10);

% Виведення спектру з позначеними піками (показуємо нормалізований)
figure;
plot(f(1:round(N/2)), fft_half_norm, 'b', 'LineWidth', 1.5);
grid on;
hold on;
% Для графіку використовуємо нормалізовані значення
plot(f(locs), maxima/N, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r', 'DisplayName', 'Піки');
hold off;
title('Перша половина амплітудного спектру з позначеними піками');
xlabel('Частота f, Гц');
ylabel('Амплітуда |C_y(f)|');
legend('Location', 'best');
```



На графіку видно значущі локальні максимуми («піки»), що відповідають основним гармонійним коливанням.

```
ЗНАЙДЕНІ ДОМІНУЮЧІ ЧАСТОТИ
========================================
Кількість піків: 1

Пік 1: f = 10.00 Гц, амплітуда = 8.7385
========================================
```

4. Побудова моделі методом найменших квадратів

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

$y(t)=a_{1}t^{3}+a_{2}t^{2}+a_{3}t+\sum_{i=4}^{k}a_{i}sin(2\pi f_{i-3}t)+a_{k+1}$

Для цього було сформовано матрицю системи A, стовпцями якої є вектори базових функцій (t3, t2, t, $sin(2\pi f_{j-3}t),$ 1), та вектор спостережень y.

Система лінійних алгебраїчних рівнянь (СЛАР) A a y була розв'язана відносно вектора коефіцієнтів a за допомогою оператора \ в MATLAB.

```
%% 7. ФОРМУВАННЯ СИСТЕМИ ДЛЯ МЕТОДУ НАЙМЕНШИХ КВАДРАТІВ
% Побудова матриці синусоїдальних компонентів на знайдених частотах
sinf_components = zeros(length(locs), N);
for i = 1:length(locs)
    sinf_components(i, :) = sin(2 * pi * f(locs(i)) * t);
end

% Формування матриці системи A = [t³, t², t, sin(2πf₁t), ..., sin(2πfₙt), 1]
A = [t.^3; t.^2; t; sinf_components; ones(1, N)]';

% Розв'язання системи A*coeff = y методом найменших квадратів
coeff = A \ y';


% Виведення знайдених коефіцієнтів
fprintf('========================================\n');
fprintf('ЗНАЙДЕНІ КОЕФІЦІЄНТИ\n');
fprintf('========================================\n');
fprintf('a1 (коеф. при t^3):     %+.6f\n', coeff(1));
fprintf('a2 (коеф. при t^2):     %+.6f\n', coeff(2));
fprintf('a3 (коеф. при t):       %+.6f\n', coeff(3));
for i = 1:length(locs)
    fprintf('a%d (коеф. при sin(2pi*%.2f*t)): %+.6f\n', 3+i, f(locs(i)), coeff(3+i));
end
fprintf('a%d (вільний член):       %+.6f\n', 3+length(locs)+1, coeff(end));
fprintf('========================================\n\n');
```

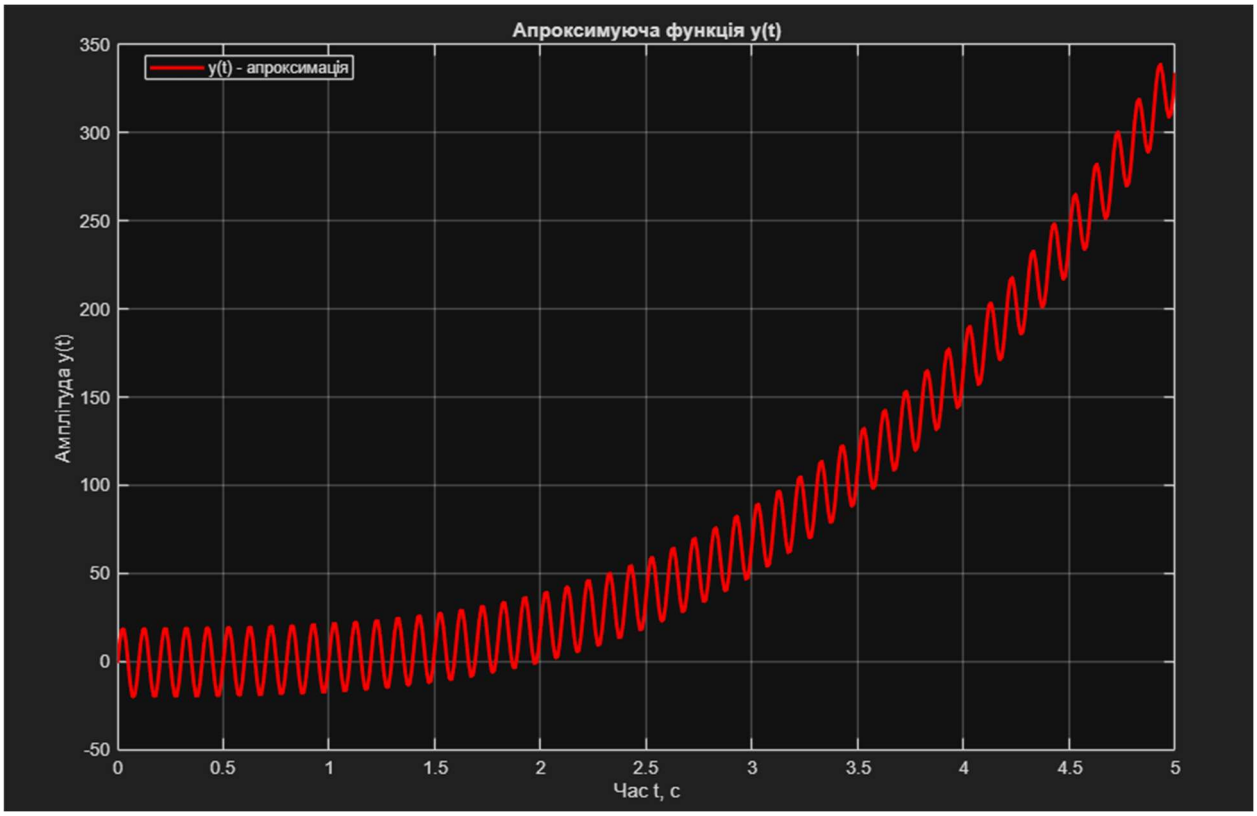
```
АПРОКСИМУЮЧА ФУНКЦІЯ
========================================
y(t) = 3.0000*t^3 + -2.0000*t^2 + 2.0000*t + 20.0000*sin(2pi*10.00*t) + -1.0000
========================================
```

```
ЗНАЙДЕНІ КОЕФІЦІЄНТИ
========================================
a1 (коеф. при t^3):     +3.000000
a2 (коеф. при t^2):     -2.000000
a3 (коеф. при t):       +2.000001
a4 (коеф. при sin(2pi*10.00*t)): +20.000003
a5 (вільний член):       -1.000000
========================================
```

5. Аналіз результатів апроксимації

Знайдені коефіцієнти a були підставлені у рівняння моделі для побудови апроксимуючої функції $y^{\wedge}(t)$ Також знаходимо квадратичну похибку функції:

```
%% 9. ВИВЕДЕННЯ АПРОКСИМУЮЧОЇ ФУНКЦІЇ
% Виведення апроксимуючої функції у символьному вигляді
fprintf('========================================\n');
fprintf('АПРОКСИМУЮЧА ФУНКЦІЯ\n');
fprintf('========================================\n');
fprintf('y(t) = %.4f*t^3 + %.4f*t^2 + %.4f*t', coeff(1), coeff(2), coeff(3));
for i = 4:length(coeff)-1
    fprintf(' + %.4f*sin(2pi*%.2f*t)', coeff(i), f(locs(i-3)));
end
fprintf(' + %.4f\n', coeff(end));
fprintf('========================================\n\n');
% ===> КІНЕЦЬ БЛОКУ <===
```

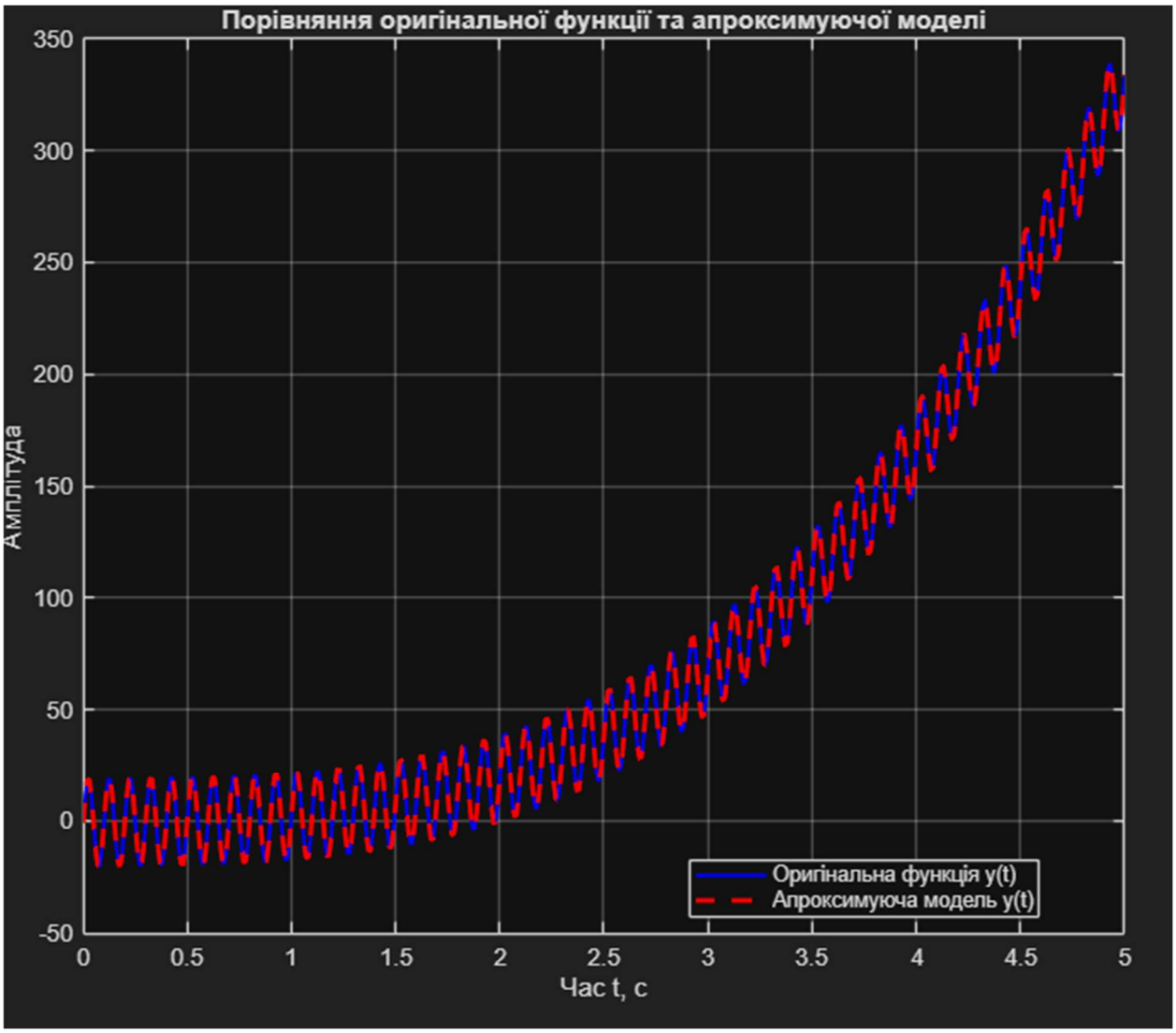



```
%% 11. ОЦІНКА ЯКОСТІ АПРОКСИМАЦІЇ
% Квадратична похибка
error_squared = sum((y_model' - y).^2);
% Середньоквадратична похибка (MSE)
mse = error_squared / N;
% Відносна похибка
relative_error = sqrt(error_squared / sum(y.^2)) * 100;

% ===> СКОПІЮЙТЕ ЦЕЙ БЛОК У ЗВІТ <===
fprintf('========================================\n');
fprintf('ОЦІНКА ТОЧНОСТІ МОДЕЛІ\n');
fprintf('========================================\n');
fprintf('Сумарна квадратична похибка (SSE): %.6e\n', error_squared);
fprintf('Середньоквадратична похибка (MSE):  %.6e\n', mse);
fprintf('Відносна похибка:                   %.4f%%\n', relative_error);
fprintf('========================================\n\n');
% ===> КІНЕЦЬ БЛОКУ <===
```

```
========================================
ОЦІНКА ТОЧНОСТІ МОДЕЛІ
```

```
=======================================
Сумарна квадратична похибка (SSE): 7.946919e+06
Сумарна квадратична похибка (SSE): 7.045867e+06
Сумарна квадратична похибка (SSE): 6.557998e+06
Сумарна квадратична похибка (SSE): 6.556795e+06
Сумарна квадратична похибка (SSE): 7.041830e+06
Сумарна квадратична похибка (SSE): 7.939062e+06
Сумарна квадратична похибка (SSE): 8.974369e+06
Сумарна квадратична похибка (SSE): 9.682570e+06
Сумарна квадратична похибка (SSE): 9.680797e+06
Сумарна квадратична похибка (SSE): 8.969421e+06
Сумарна квадратична похибка (SSE): 7.931853e+06
Сумарна квадратична похибка (SSE): 7.033172e+06
Сумарна квадратична похибка (SSE): 6.546800e+06
Сумарна квадратична похибка (SSE): 6.545765e+06
Сумарна квадратична похибка (SSE): 7.029708e+06
І так далі…
```

Додатково: порівняння двох графіків:



Цілий код:

```
%% 1. ІНІЦІАЛІЗАЦІЯ ПАРАМЕТРІВ
clear all;
clc;
```

```
close all;

% Крок дискретизації та інтервал спостереження
dt = 0.01;       % Крок дискретизації
T = 5;           % Інтервал спостереження
t = 0:dt:T;      % Масив часових точок
N = length(t);   % Кількість точок

%% 2. ЗАВАНТАЖЕННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДАНИХ

y = dlmread('f13.txt', '');

%% 3. ВІЗУАЛІЗАЦІЯ ОРИГІНАЛЬНОЇ ФУНКЦІЇ
figure;
plot(t, y, 'b', 'LineWidth', 1.5, 'DisplayName', 'y(t) - спостережувана функція');
grid on;
title('Оригінальна дискретна функція y(t)');
xlabel('Час t, с');
ylabel('Амплітуда y(t)');
legend('Location', 'best');

%% 4. ДИСКРЕТНЕ ПЕРЕТВОРЕННЯ ФУР'Є (вбудована функція)
fft_y_raw = fft(y);       % БЕЗ нормалізації (для пошуку піків)
fft_y = fft_y_raw / N;  % З нормалізацією (для відображення)

% Побудова амплітудного спектру (FFT)
figure;
freq_axis = (0:N-1) / T; % Частотна вісь в Гц
plot(freq_axis, abs(fft_y), 'b', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (вбудована функція FFT)');
xlabel('Частота f, Гц');
ylabel('Амплітуда |C_y(f)|');

%% 5. ВЛАСНА РЕАЛІЗАЦІЯ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ФУР'Є
% (Для перевірки коректності)
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 результату

% Побудова амплітудного спектру (власна реалізація)
figure;
plot(freq_axis, abs(fft_manual), 'r', 'LineWidth', 1.2);
grid on;
title('Амплітудний спектр |C_y(f)| (власна реалізація ДПФ)');
xlabel('Частота f, Гц');
ylabel('Амплітуда |C_y(f)|');

%% 6. АНАЛІЗ ЧАСТОТНОГО СПЕКТРУ
% Визначення частотних компонент
df = 1 / T; % Частотне розділення
f = 0:df:(N-1)*df; % Частотна вісь

% Беремо тільки першу половину спектру (оскільки він симетричний)
fft_half_raw = abs(fft_y_raw(1:round(N/2))); % Перша половина БЕЗ нормалізації
fft_half_norm = abs(fft_y(1:round(N/2)));   % Перша половина З нормалізацією

% Пошук піків (екстремумів) у спектрі
```

```
% 'MinPeakHeight' - фільтр, щоб відсіяти малий шум. Можливо, доведеться змінити це
значення.
[maxima, locs] = findpeaks(fft_half_raw, 'MinPeakHeight', 10);

% Виведення спектру з позначеними піками (показуємо нормалізований)
figure;
plot(f(1:round(N/2)), fft_half_norm, 'b', 'LineWidth', 1.5);
grid on;
hold on;
% Для графіку використовуємо нормалізовані значення
plot(f(locs), maxima/N, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r', 'DisplayName',
'Піки');
hold off;
title('Перша половина амплітудного спектру з позначеними піками');
xlabel('Частота f, Гц');
ylabel('Амплітуда |C_y(f)|');
legend('Location', 'best');


fprintf('\n========================================\n');
fprintf('ЗНАЙДЕНІ ДОМІНУЮЧІ ЧАСТОТИ \n');
fprintf('========================================\n');
fprintf('Кількість піків: %d\n\n', length(locs));
for i = 1:length(locs)
    fprintf('Пік %d: f = %.2f Гц, амплітуда = %.4f\n', i, f(locs(i)), maxima(i)/N);
end
fprintf('========================================\n\n');
% ===> КІНЕЦЬ БЛОКУ <===

%% 7. ФОРМУВАННЯ СИСТЕМИ ДЛЯ МЕТОДУ НАЙМЕНШИХ КВАДРАТІВ
% Побудова матриці синусоїдальних компонентів на знайдених частотах
sinf_components = zeros(length(locs), N);
for i = 1:length(locs)
    sinf_components(i, :) = sin(2 * pi * f(locs(i)) * t);
end

% Формування матриці системи A = [t³, t², t, sin(2πf₁t), ..., sin(2πfₙt), 1]
A = [t.^3; t.^2; t; sinf_components; ones(1, N)]';

% Розв'язання системи A*coeff = y методом найменших квадратів
coeff = A \ y';


% Виведення знайдених коефіцієнтів
fprintf('========================================\n');
fprintf('ЗНАЙДЕНІ КОЕФІЦІЄНТИ\n');
fprintf('========================================\n');
fprintf('a1 (коеф. при t^3):     %+.6f\n', coeff(1));
fprintf('a2 (коеф. при t^2):     %+.6f\n', coeff(2));
fprintf('a3 (коеф. при t):       %+.6f\n', coeff(3));
for i = 1:length(locs)
    fprintf('a%d (коеф. при sin(2pi*%.2f*t)): %+.6f\n', 3+i, f(locs(i)), coeff(3+i));
end
fprintf('a%d (вільний член):       %+.6f\n', 3+length(locs)+1, coeff(end));
fprintf('========================================\n\n');


%% 8. ПОБУДОВА АПРОКСИМУЮЧОЇ МОДЕЛІ
% Обчислення апроксимованих значень
y_model = coeff(1) * t.^3 + coeff(2) * t.^2 + coeff(3) * t;
% Додавання синусоїдальних компонент
for i = 4:length(coeff)-1
```

```
    y_model = y_model + coeff(i) * sin(2 * pi * f(locs(i-3)) * t);
end
y_model = y_model + coeff(end); % Додавання вільного члена

%% 9. ВИВЕДЕННЯ АПРОКСИМУЮЧОЇ ФУНКЦІЇ
% Виведення апроксимуючої функції у символьному вигляді
fprintf('=======================================\n');
fprintf('АПРОКСИМУЮЧА ФУНКЦІЯ\n');
fprintf('=======================================\n');
fprintf('y(t) = %.4f*t^3 + %.4f*t^2 + %.4f*t', coeff(1), coeff(2), coeff(3));
for i = 4:length(coeff)-1
    fprintf(' + %.4f*sin(2pi*%.2f*t)', coeff(i), f(locs(i-3)));
end
fprintf(' + %.4f\n', coeff(end));
fprintf('=======================================\n\n');
% ===> КІНЕЦЬ БЛОКУ <===

%% 10. ВІЗУАЛІЗАЦІЯ АПРОКСИМУЮЧОЇ МОДЕЛІ
figure;
plot(t, y_model, 'r', 'LineWidth', 2, 'DisplayName', 'y(t) - апроксимація');
grid on;
title('Апроксимуюча функція y(t)');
xlabel('Час t, с');
ylabel('Амплітуда y(t)');
legend('Location', 'best');

%% 11. ОЦІНКА ЯКОСТІ АПРОКСИМАЦІЇ
% Квадратична похибка
error_squared = sum((y_model' - y).^2);
% Середньоквадратична похибка (MSE)
mse = error_squared / N;
% Відносна похибка
relative_error = sqrt(error_squared / sum(y.^2)) * 100;

% ===> СКОПІЮЙТЕ ЦЕЙ БЛОК У ЗВІТ <===
fprintf('=======================================\n');
fprintf('ОЦІНКА ТОЧНОСТІ МОДЕЛІ\n');
fprintf('=======================================\n');
fprintf('Сумарна квадратична похибка (SSE): %.6e\n', error_squared);
fprintf('Середньоквадратична похибка (MSE):  %.6e\n', mse);
fprintf('Відносна похибка:                   %.4f%%\n', relative_error);
fprintf('=======================================\n\n');
% ===> КІНЕЦЬ БЛОКУ <===

%% 12. ПОРІВНЯННЯ ОРИГІНАЛУ ТА МОДЕЛІ
% Графік порівняння оригіналу та моделі
figure;
plot(t, y, 'b', 'LineWidth', 1.5, 'DisplayName', 'Оригінальна функція y(t)');
hold on;
plot(t, y_model, 'r--', 'LineWidth', 2, 'DisplayName', 'Апроксимуюча модель y(t)');
hold off;
grid on;
title('Порівняння оригінальної функції та апроксимуючої моделі');
xlabel('Час t, с');
ylabel('Амплітуда');
legend('Location', 'best');
```