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
Совгирі Анни Олегівни
Варіант №5

Київ
2025

## Постановка задачі

Нехай є дискретна функція $\hat{y}(t_i)$, $i = 1,2,\ldots,N$, 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 $\hat{y}(t_i)$, $i = 1,2,\ldots,N$, (відповідний файл fk.txt), $t_{i+1} - t_i = \Delta t = 0.01$, інтервал спостереження $[0, T]$, $T = 5$.

Дискретне перетворення Фур'є для дискретної послідовності $x(i)$, $i = 0,1,2,\ldots,N-1$

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

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 $f^* = k^* \mathrm{f} = k^* / T$.

Знайшовши частоти із найбільшим вкладом, можна приступати до безпосереднього визначення невідомих параметрів $a_i$, $i = k+1$. Для їх визначення застосовуємо метод найменших квадратів. Для цього записуємо функціонал похибки:

$F(a_1, a_2, \ldots, a_{k+1}) = \frac{1}{2}\sum_{j=0}^{N-1}\left(a_1 t_j^3 + a_2 t_j^2 + a_3 t_j + \sum_{i=4}^{k} a_i \sin(2\pi f_{i-3} t_j) + a_{k+1} - \hat{y}(t_j)\right)^2$

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

$$F(a_1, a_2, \ldots, a_{k+1}) \to \min_{a_1, a_2, \ldots, a_{k+1}}$$

Для цього записуємо систему рівнянь:

$$\frac{\partial F(a_1, a_2, \ldots, a_{k+1})}{\partial a_j} = 0$$

Ця система є системою лінійних алгебраїчних рівнянь. Розв'язавши цю систему одним з відомих методів, знаходимо $a_i$, 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 t. Далі з файлу *f5.txt* було завантажено масив спостережуваних значень y(t).

```
%% 1. ІНІЦІАЛІЗАЦІЯ ПАРАМЕТРІВ
% 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('f5.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:

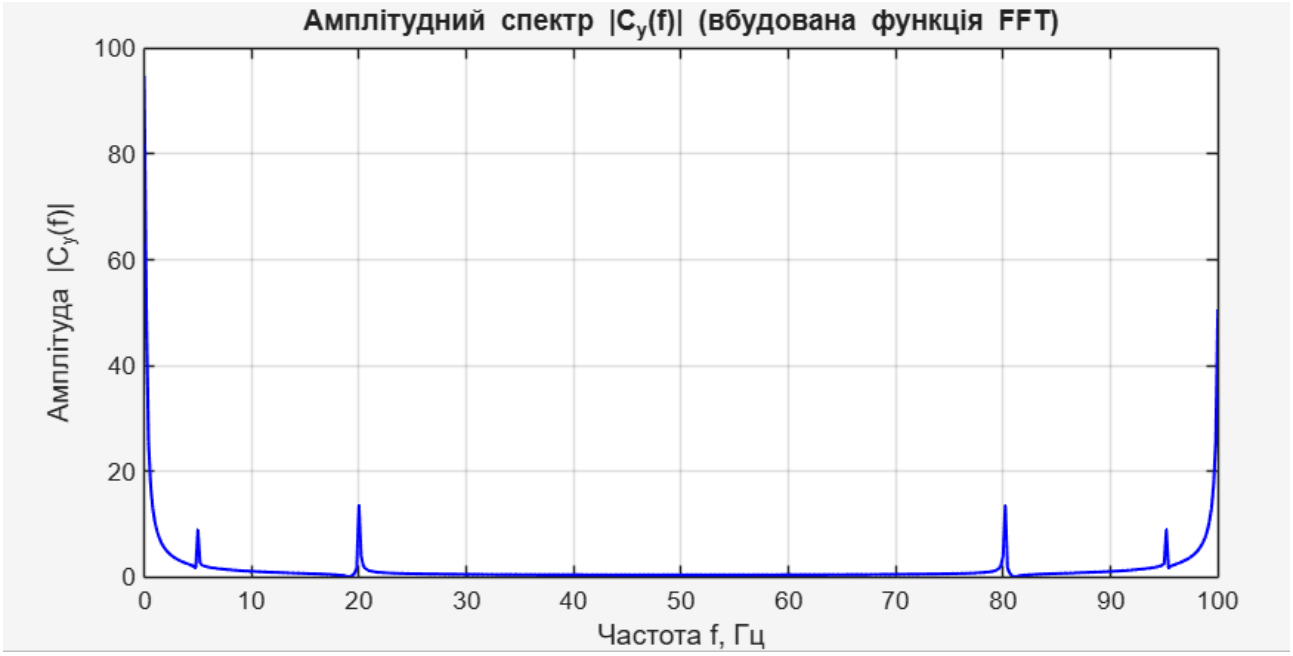
1. За допомогою вбудованої функції MATLAB *fft*.
2. Шляхом програмної реалізації алгоритму ДПФ вручну.

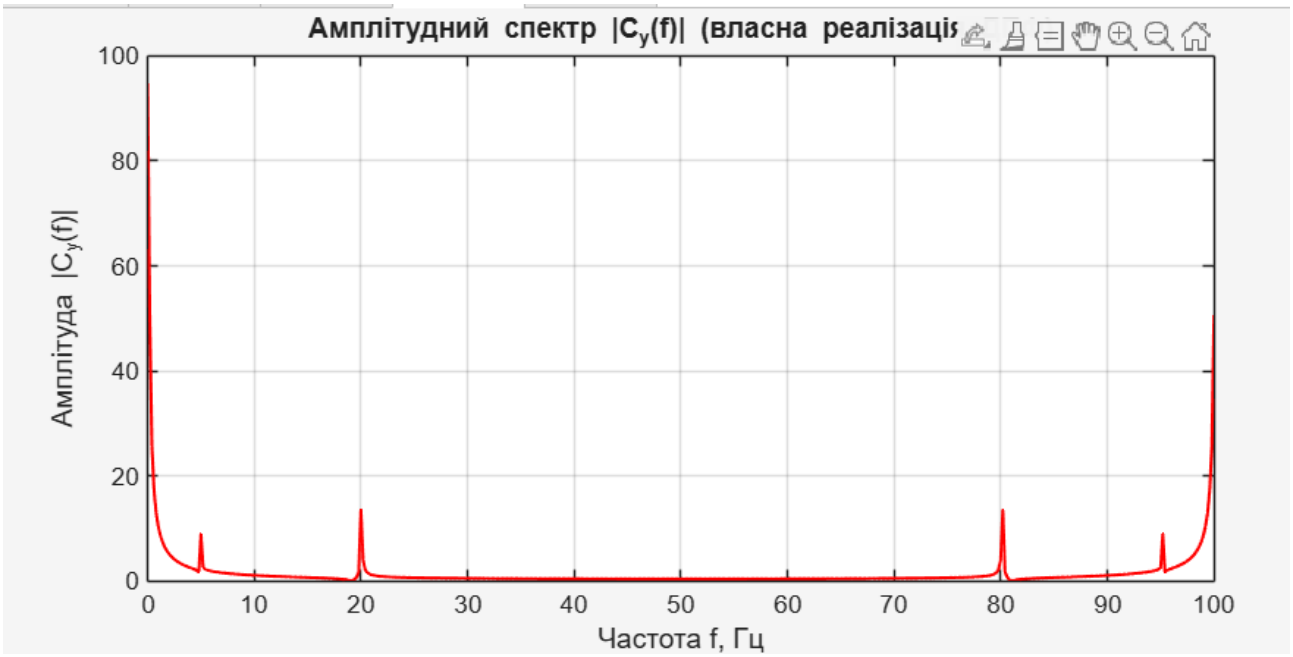
```
%% 4. ДИСКРЕТНЕ ПЕРЕТВОРЕННЯ ФУР'Є (вбудована функція)
fft_y_raw = fft(y);          % БЕЗ нормалізації (для пошуку піків)
fft_y = fft_y_raw / N;       % З нормалізацією (для відображення)

% 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)|');
```

```
%% 5. ВЛАСНА РЕАЛІЗАЦІЯ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ФУР'Є
fft_manual = zeros(1, N);
for k = 1:N
    for m = 1:N
        % Формула ДПФ: C_y(k) = (1/N) * sum(y(m) * exp(-i*2π*k*m/N))
        fft_manual(k) = fft_manual(k) + y(m) * exp(-1i * 2 * pi * (k - 1) * (m - 1)
    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)|');
```



В обох випадках результат було нормалізовано діленням на кількість точок N для отримання коректних амплітуд. 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.

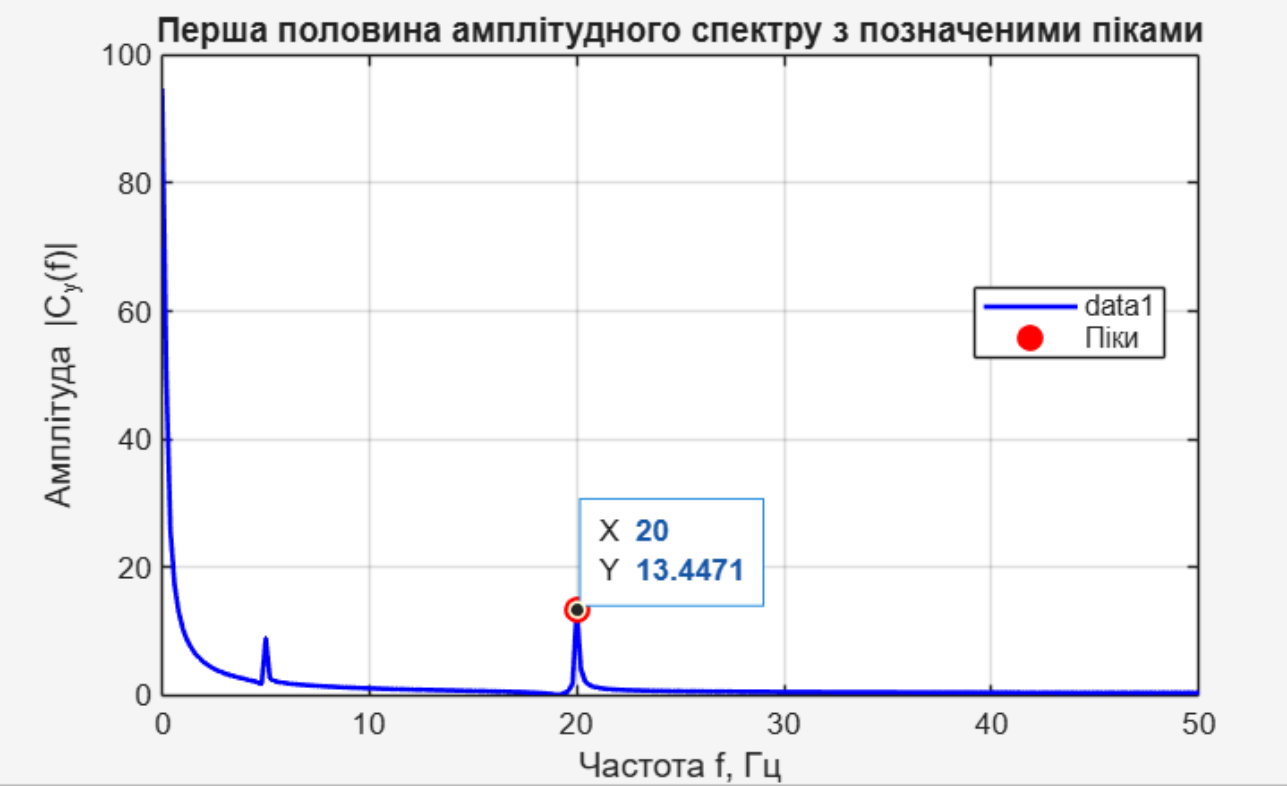
Індекси знайдених піків були перераховані у фізичні частоти (в Гц) для подальшого використання в моделі.

```
%% 6. АНАЛІЗ ЧАСТОТНОГО СПЕКТРУ
% Визначення частотних компонент
df = 1 / T;                                 % Частотне розділення
f = 0:df:(N-1)*df;                          % Частотна вісь
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
[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');
```

```
% Виведення знайдених частот у консоль
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');
```




На графіку видно значущі локальні максимуми («піки»), що відповідають основним гармонійним коливанням. Ці піки розташовані на частотах:

- **$f_1 \approx 20$ Гц**
- **$f_2 \approx 5$ Гц**
-

```
========================================
ЗНАЙДЕНІ ДОМІНУЮЧІ ЧАСТОТИ
========================================
Кількість піків: 2

Пік 1: f = 5.00 Гц, амплітуда = 8.8747
Пік 2: f = 20.00 Гц, амплітуда = 13.4471
========================================
```

**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';
```

```
%% 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 РЕЗУЛЬТАТІВ У КОНСОЛЬ
% Виведення апроксимуючої функції у символьному вигляді
disp('========================================');
disp('Апроксимуюча функція:');
disp('========================================');
fprintf('y(t) = %.4f·t³ + %.4f·t² + %.4f·t ', coeff(1), coeff(2), coeff(3));
for i = 4:length(coeff)-1
    fprintf('+ %.4f·sin(2π·%.2f·t) ', coeff(i), f(locs(i-3)));
end
fprintf('+ %.4f\n', coeff(end));
disp('========================================');
```

```
========================================
АПРОКСИМУЮЧА ФУНКЦІЯ
========================================
y(t) = 2.0000·t³ + 2.0000·t² + 5.0000·t + 22.0000·sin(2π·5.00·t) + 30.0000·sin(2π·20.00·t) + 3.0000
========================================
```

```
========================================
СИСТЕМА РІВНЯНЬ (метод найменших квадратів)
========================================
Розмірність матриці A: 501 × 6
Розмірність вектора y: 501 × 1

Структура базисних функцій:
  ϕ₁(t) = t³
  ϕ₂(t) = t²
  ϕ₃(t) = t
  ϕ4(t) = sin(2π·5.00·t)
  ϕ5(t) = sin(2π·20.00·t)
  ϕ6(t) = 1 (вільний член)

Система: A·a = y
де a = [a₁, a₂, a₃, a4, a5, a6]ᵀ
========================================

========================================
ЗНАЙДЕНІ КОЕФІЦІЄНТИ
========================================
a₁ (коеф. при t³):   +2.000001
a₂ (коеф. при t²):   +1.999993
a₃ (коеф. при t):    +5.000013
a4 (коеф. при sin(2π·5.00·t)): +22.000003
a5 (коеф. при sin(2π·20.00·t)): +30.000000
a6 (вільний член):  +2.999993
```

**5. Аналіз результатів апроксимації**

Знайдені коефіцієнти a були підставлені у рівняння моделі для побудови апроксимуючої функції y^(t). Також знаходимо квадратичну похибку функції:


```
========================================
ОЦІНКА ТОЧНОСТІ МОДЕЛІ
========================================
Сумарна квадратична похибка (SSE): 4.556110e-07
Середньоквадратична похибка (MSE):  9.094032e-10
Відносна похибка:                    0.0000%
========================================
```

**Додатково: порівняння двох графіків:**

# Код повністю:

```
%% ========================================================================
%  ЛАБОРАТОРНА РОБОТА №1: Апроксимація дискретної функції
%  Моделювання складних систем
%% ========================================================================

%% 1. ІНІЦІАЛІЗАЦІЯ ПАРАМЕТРІВ
% 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('f5.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
fft_manual = zeros(1, N);
for k = 1:N
    for m = 1:N
        % Формула ДПФ: C_y(k) = (1/N) * sum(y(m) * exp(-i*2π*k*m/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
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
[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');

% Виведення знайдених частот у консоль
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');

%% 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(2πfₙ t), 1]
A = [t.^3; t.^2; t; sinf_components; ones(1, N)]';

% Виведення структури системи рівнянь
fprintf('========================================\n');
fprintf('СИСТЕМА РІВНЯНЬ (метод найменших квадратів)\n');
fprintf('========================================\n');
fprintf('Розмірність матриці A: %d × %d\n', size(A, 1), size(A, 2));
fprintf('Розмірність вектора y: %d × 1\n\n', length(y));
fprintf('Структура базисних функцій:\n');
fprintf('  ϕ₁(t) = t³\n');
fprintf('  ϕ₂(t) = t²\n');
fprintf('  ϕ₃(t) = t\n');
for i = 1:length(locs)
    fprintf('  ϕ%d(t) = sin(2π·%.2f·t)\n', 3+i, f(locs(i)));
end
fprintf('  ϕ%d(t) = 1 (вільний член)\n\n', 3+length(locs)+1);
fprintf('Система: A·a = y\n');
fprintf('де a = [a₁, a₂, a₃');
for i = 1:length(locs)
    fprintf(', a%d', 3+i);
end
fprintf(', a%d]ᵀ\n', 3+length(locs)+1);
fprintf('========================================\n\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 t³):   %+.6f\n', coeff(1));
fprintf('a₂ (коеф. при t²):   %+.6f\n', coeff(2));
fprintf('a₃ (коеф. при t):    %+.6f\n', coeff(3));
for i = 1:length(locs)
    fprintf('a%d (коеф. при sin(2π·%.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·t³ + %.4f·t² + %.4f·t', coeff(1), coeff(2), coeff(3));
for i = 4:length(coeff)-1
    fprintf(' + %.4f·sin(2π·%.2f·t)', coeff(i), f(locs(i-3)));
end
fprintf(' + %.4f\n', coeff(end));
fprintf('=======================================\n\n');

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

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');
```

```
%% 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 ŷ(t)');
hold off;
grid on;
title('Порівняння оригінальної функції та апроксимуючої моделі');
xlabel('Час t, с');
ylabel('Амплітуда');
legend('Location', 'best');
```