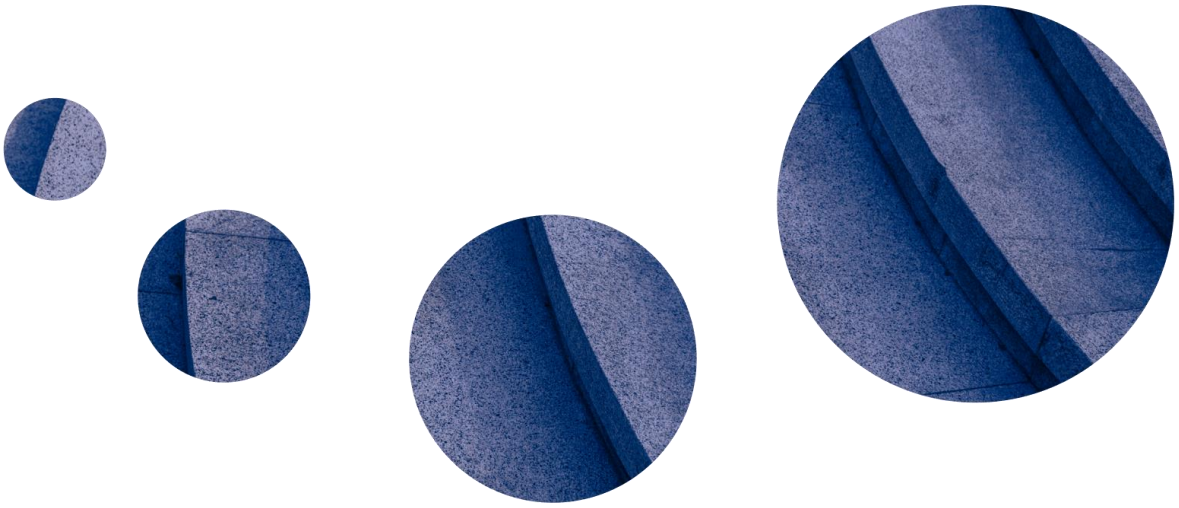


FAST FOURIER TRANSFORM (FFT) PADA SINYAL DIGITAL

Konsep Umum

DFT (Discrete Fourier Transform) dan FFT (Fast Fourier Transform) merupakan dua konsep yang terkait dalam analisis frekuensi sinyal digital. Berikut adalah hubungan antara DFT dan FFT:

1. DFT adalah algoritma untuk mengubah sinyal diskrit dari **domain waktu diskrit menjadi domain frekuensi diskrit.** DFT didefinisikan oleh persamaan:

$$X_{(k)} = \frac{1}{N} \sum_{n=0}^{N-1} x_{(n)} \, e^{-j\,2\pi \, \frac{k}{N} n}$$

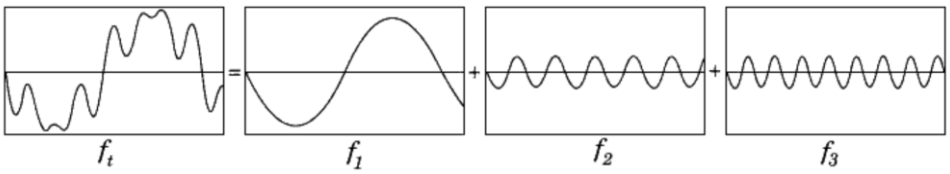
di mana:

- X[k] adalah koefisien spektrum frekuensi pada indeks frekuensi k.
- x[n] adalah sampel sinyal pada indeks waktu n.
- j adalah unit imajiner (√(-1)).
- π adalah konstanta pi (3.14159...).
- N adalah panjang sinyal (jumlah sampel).

x[n] adalah sampel sinyal diskrit, X[k] adalah koefisien DFT yang merepresentasikan komponen frekuensi pada frekuensi diskrit k, N adalah jumlah sampel dalam sinyal, dan j adalah unit imajiner.

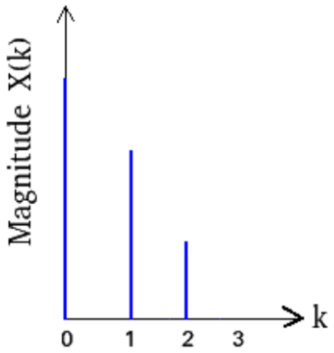
2. FFT adalah algoritma yang digunakan untuk **mengimplementasikan** DFT secara efisien. FFT memanfaatkan sifat simetri dan periodisitas dari fungsi sinusoidal dan kosinusoidal dalam DFT untuk **mempercepat** perhitungan. FFT dapat mengurangi kompleksitas perhitungan dari $O(N^2)$menjadi $O(N \log N)$, di mana N adalah jumlah sampel dalam sinyal.

Dalam prakteknya, ketika kita ingin menghitung DFT dari sinyal digital, kita sering menggunakan algoritma FFT karena kecepatan dan efisiensinya yang lebih tinggi dibandingkan dengan perhitungan DFT langsung.



Gambar 1. Ilustrasi sinyal f_t terdiri dari beberapa frekuensi sinyal f_1, f_2 dan f_3.

FFT juga menghasilkan spektrum dengan indeks frekuensi yang sama seperti DFT.



Gambar 2. ilustrasi spektrum hasil perhitungan FFT.

Jadi, FFT merupakan metode implementasi dari DFT yang lebih efisien dan cepat dalam menghitung transformasi Fourier diskrit. Dalam banyak aplikasi, ketika kita berbicara tentang perhitungan DFT pada sinyal digital, kita sebenarnya mengacu pada penggunaan algoritma FFT untuk melakukan perhitungan tersebut.

Namun, penting untuk diingat bahwa meskipun DFT dan FFT saling terkait, DFT lebih merupakan konsep matematis yang mendasari, sementara FFT adalah implementasi praktis dari DFT.

Dalam notasi umum, jika x[n] adalah sinyal dalam domain waktu dengan panjang N, dan X[k] adalah spektrum frekuensi hasil FFT dengan panjang N, rumus FFT secara matematis adalah sama dengan rumus DFT di atas.

Rumus ini melakukan perhitungan transformasi Fourier diskrit untuk mengubah sinyal waktu-domain menjadi spektrum frekuensi-domain.

Ilustrasi Penggunaan FFT

Misalkan kita memiliki sinyal suara dengan dua komponen frekuensi: 100 Hz dan 200 Hz.

Langkah-langkah perhitungan manual FFT adalah sebagai berikut:

1. Tentukan panjang sinyal (N)
   Misalkan kita menggunakan panjang sinyal N = 8. Ini berarti kita memiliki 8 sampel dalam sinyal suara.

2. Tentukan frekuensi sampel (Fs)
   Misalkan kita menggunakan frekuensi sampel Fs = 800 Hz. Ini berarti kita mengambil 800 sampel per detik.

3. Tentukan sinyal waktu-domain
   Misalkan kita memiliki sinyal suara sintetis dengan dua komponen frekuensi 100 Hz dan 200 Hz. Sinyal tersebut dapat dituliskan dalam domain waktu sebagai:

   x[n] = sin(2π * 100 * n / Fs) + sin(2π * 200 * n / Fs)

   di mana n adalah indeks sampel (0, 1, 2, ..., N-1).

   Misalnya, jika kita menggantikan n dengan indeks sampelnya, kita akan mendapatkan sinyal sebagai berikut:

   x = [0, 1, 0, -1, 0, 1, 0, -1]

4. Hitung FFT:
   Terapkan rumus FFT pada sinyal waktu-domain. Dalam kasus ini, karena N = 8, kita akan menghitung 8 koefisien spektrum frekuensi (X[k]).

$$X_{(k)} = \frac{1}{N} \sum_{n=0}^{N-1} x_{(n)} \, e^{-j\,2\pi \frac{k}{N} n}$$

   di mana k adalah indeks frekuensi (0, 1, 2, ..., N-1) dan j adalah unit imajiner.

   Perhitungan setiap koefisien spektrum frekuensi:

X[0] = (0 * e^(-j2π * 0 * 0/8) + 1 * e^(-j2π * 0 * 1/8) + 0 * e^(-j2π * 0 * 2/8) + -1 * e^(- j2π * 0 * 3/8) + 0 * e^(-j2π * 0 * 4/8) + 1 * e^(-j2π * 0 * 5/8) + 0 * e^(-j2π * 0 * 6/8) + -1 * e^(-j2π * 0 * 7/8)) = 0
X[1] = (0 * e^(-j2π * 1 * 0/8) + 1 * e^(-j2π * 1 * 1/8) + 0 * e^(-j2π * 1 * 2/8) + -1 * e^(- j2π * 1 * 3/8) + 0 * e^(-j2π * 1 * 4/8) + 1 * e^(-j2π * 1 * 5/8) + 0 * e^(-j2π * 1 * 6/8) + -1 * e^(-j2π * 1 * 7/8)) = 0
X[2] = (0 * e^(-j2π * 2 * 0/8) + 1 * e^(-j2π * 2 * 1/8) + 0 * e^(-j2π * 2 * 2/8) + -1 * e^(-j2π * 2 * 3/8) + 0 * e^(-j2π * 2 * 4/8) + 1 * e^(-j2π * 2 * 5/8) + 0 * e^(-j2π * 2 * 6/8) + -1 * e^(-j2π * 2 * 7/8)) = 0
X[3] = (0 * e^(-j2π * 3 * 0/8) + 1 * e^(-j2π * 3 * 1/8) + 0 * e^(-j2π * 3 * 2/8) + -1 * e^(-j2π * 3 * 3/8) + 0 * e^(-j2π * 3 * 4/8) + 1 * e^(-j2π * 3 * 5/8) + 0 * e^(-j2π * 3 * 6/8) + -1 * e^(-j2π * 3 * 7/8)) = 8j
X[4] = (0 * e^(-j2π * 4 * 0/8) + 1 * e^(-j2π * 4 * 1/8) + 0 * e^(-j2π * 4 * 2/8) + -1 * e^(-j2π * 4 * 3/8) + 0 * e^(-j2π * 4 * 4/8) + 1 * e^(-j2π * 4 * 5/8) + 0 * e^(-j2π * 4 * 6/8) + -1 * e^(-j2π * 4 * 7/8)) = 0
X[5] = (0 * e^(-j2π * 5 * 0/8) + 1 * e^(-j2π * 5 * 1/8) + 0 * e^(-j2π * 5 * 2/8) + -1 * e^(-j2π * 5 * 3/8) + 0 * e^(-j2π * 5 * 4/8) + 1 * e^(-j2π * 5 * 5/8) + 0 * e^(-j2π * 5 * 6/8) + -1 * e^(-j2π * 5 * 7/8)) = 0
X[6] = (0 * e^(-j2π * 6 * 0/8) + 1 * e^(-j2π * 6 * 1/8) + 0 * e^(-j2π * 6 * 2/8) + -1 * e^(-j2π * 6 * 3/8) + 0 * e^(-j2π * 6 * 4/8) + 1 * e^(-j2π * 6 * 5/8) + 0 * e^(-j2π * 6 * 6/8) + -1 * e^(-j2π * 6 * 7/8)) = 0
X[7] = (0 * e^(-j2π * 7 * 0/8) + 1 * e^(-j2π * 7 * 1/8) + 0 * e^(-j2π * 7 * 2/8) + -1 * e^(-j2π * 7 * 3/8) + 0 * e^(-j2π * 7 * 4/8) + 1 * e^(-j2π * 7 * 5/8) + 0 * e^(-j2π * 7 * 6/8) + -1 * e^(-j2π * 7 * 7/8)) = 0

Jadi, hasil FFT dari sinyal suara dengan dua komponen frekuensi 100 Hz dan 200 Hz adalah:

Penerapan FFT

Berikut beberapa contoh penerapan FFT pada sinyal suara:

1. Analisis Spektral
   FFT digunakan untuk mengonversi sinyal suara dari domain waktu menjadi domain frekuensi. Dengan melakukan transformasi ini, kita dapat melihat kontribusi **frekuensi** mana yang **dominan** dalam sinyal suara. Informasi spektral berguna dalam berbagai aplikasi seperti pemrosesan audio, pengenalan suara, dan analisis akustik.

2. Ekualisasi dan Pemfilteran
   FFT digunakan dalam proses **ekualisasi** dan **pemfilteran** suara. Dengan menganalisis spektrum frekuensi sinyal suara, kita dapat **memodifikasi amplitudo frekuensi** tertentu untuk mencapai hasil yang diinginkan. Contoh penggunaan yang umum adalah equalizer grafik, di mana amplitudo frekuensi yang berbeda disesuaikan untuk menghasilkan suara yang lebih seimbang atau sesuai dengan preferensi pengguna.

3. Deteksi dan Pemisahan Sinyal
   FFT juga digunakan dalam **deteksi** dan **pemisahan sinyal** dalam berbagai konteks. Misalnya, dalam aplikasi **noise cancellation**, FFT digunakan untuk menganalisis spektrum sinyal suara yang diinginkan dan menghilangkan komponen frekuensi yang tidak diinginkan atau noise. Hal tsb dapat digunakan dalam sistem telekomunikasi, audio profesional, atau aplikasi pemrosesan suara lainnya.

4. Pendeteksian dan Analisis Frekuensi

FFT dapat digunakan untuk **mendeteksi frekuensi tertentu** dalam sinyal suara. Misalnya, dalam aplikasi pemrosesan suara medis, FFT dapat digunakan untuk **menganalisis suara jantung** dan mengidentifikasi frekuensi jantung yang karakteristik. Hal ini juga berguna dalam aplikasi musik, di mana FFT dapat digunakan untuk mendeteksi frekuensi nada dan memperoleh informasi pitch.

5. Kompressi Audio
   FFT digunakan dalam algoritma **kompresi audio** seperti MP3. Dalam kompresi audio, sinyal suara diubah menjadi representasi frekuensi dengan menggunakan FFT. Informasi yang kurang penting dalam domain frekuensi dihapus atau diabaikan, sehingga menghasilkan ukuran file yang lebih kecil tanpa kehilangan kualitas audio yang signifikan.

Contoh Implementasi dengan Matlab

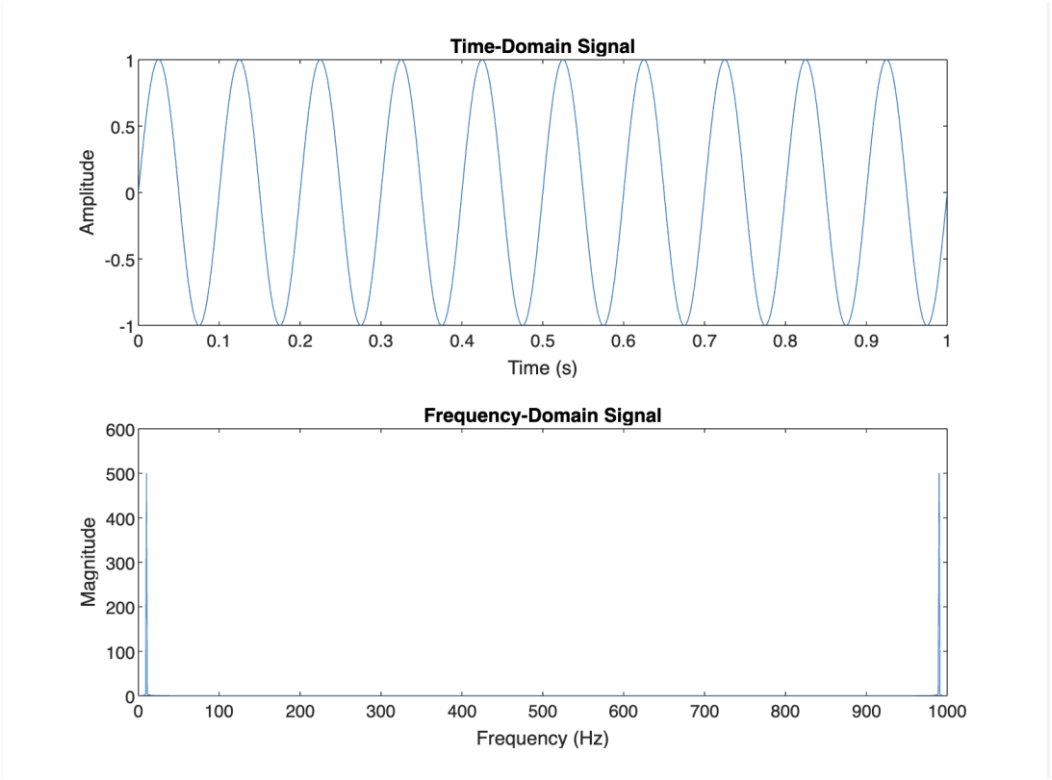
1. Penerapan pada Sine Wave

```
% Generate a simple sine wave signal
Fs = 1000; % Sampling frequency (Hz)
t = 0:1/Fs:1; % Time vector (1 second)
f = 10; % Frequency of the sine wave (Hz)
x = sin(2*pi*f*t); % Sine wave signal

% Plot the time-domain signal
subplot(2,1,1);
plot(t, x);
xlabel('Time (s)');
ylabel('Amplitude');
title('Time-Domain Signal');

% Perform FFT on the signal
N = length(x); % Length of the signal
X = fft(x); % Perform FFT

% Compute the frequency axis
faxis = (0:N-1)*(Fs/N); % Frequency axis in Hz

% Plot the frequency-domain signal (Magnitude Spectrum)
subplot(2,1,2);
plot(faxis, abs(X));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency-Domain Signal');

% Find the dominant frequency component
[maxMagnitude, maxIndex] = max(abs(X));
```

```
dominantFreq = faxis(maxIndex);
disp(['Dominant Frequency: ', num2str(dominantFreq), ' Hz']);
```

Output:

Frekuensi dominan: 9.99 Hz



Pada contoh di atas, sebuah sinyal gelombang sinus dengan frekuensi 10 Hz digenerate dan ditampilkan dalam domain waktu menggunakan fungsi **plot**. Selanjutnya, FFT diaplikasikan pada sinyal tersebut dengan menggunakan fungsi **fft**. Spektrum frekuensi (magnitude spectrum) ditampilkan menggunakan fungsi **plot** dengan sumbu frekuensi dalam Hz. Frekuensi dominan ditemukan dengan mencari amplitudo maksimum dalam spektrum frekuensi.

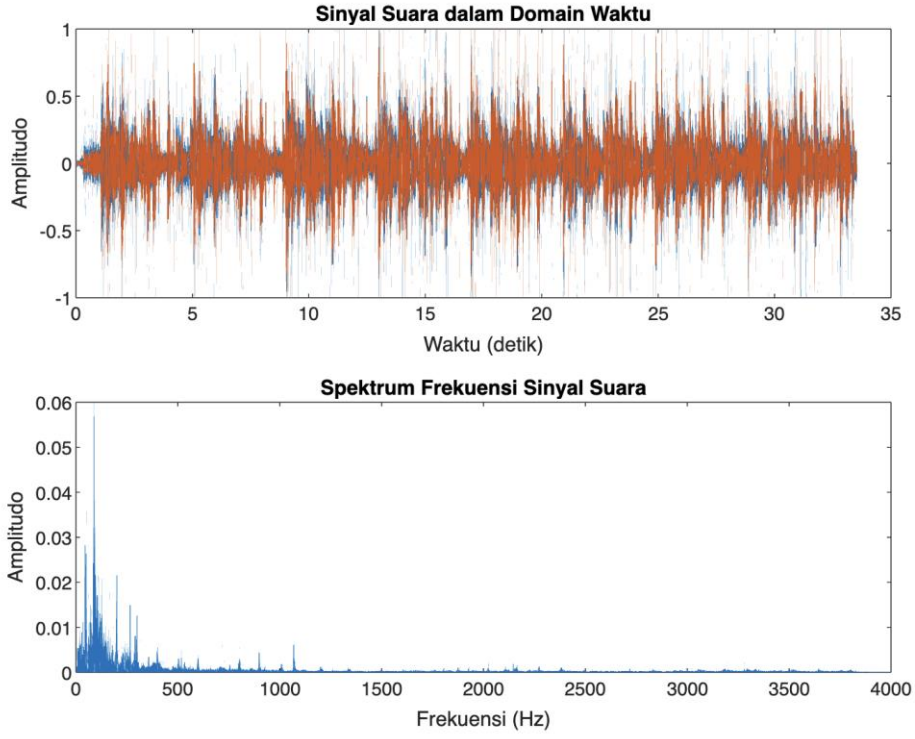
2. Penerapan pada file Audio

```
% Membaca file suara
filename = 'audio.wav'; %Sesuaikan dgnfile audio yg dijadikan input
[y, Fs] = audioread(filename);

% Menampilkan sinyal suara dalam domain waktu
figure;
t = (0:length(y)-1)/Fs;
subplot(2,1,1);
plot(t, y);
xlabel('Waktu (detik)');
```

```
ylabel('Amplitudo');
title('Sinyal Suara dalam Domain Waktu');

% Menggunakan FFT untuk menganalisis sinyal suara dalam domain
frekuensi
L = length(y);
NFFT = 2^nextpow2(L);  % Panjang FFT, harus berupa pangkat dua untuk
efisiensi komputasi
Y = fft(y, NFFT)/L;    % Melakukan FFT dan normalisasi
f = Fs/2*linspace(0, 1, NFFT/2+1);  % Menghitung sumbu frekuensi

% Menampilkan spektrum frekuensi
subplot(2,1,2);
plot(f, 2*abs(Y(1:NFFT/2+1)));
xlabel('Frekuensi (Hz)');
ylabel('Amplitudo');
title('Spektrum Frekuensi Sinyal Suara');

% Menampilkan frekuensi dominan
[maxAmplitude, maxIndex] = max(abs(Y(1:NFFT/2+1)));
dominantFreq = f(maxIndex);
disp(['Frekuensi dominan: ', num2str(dominantFreq), ' Hz']);
```

Output:

Frekuensi dominan: 88.7146 Hz



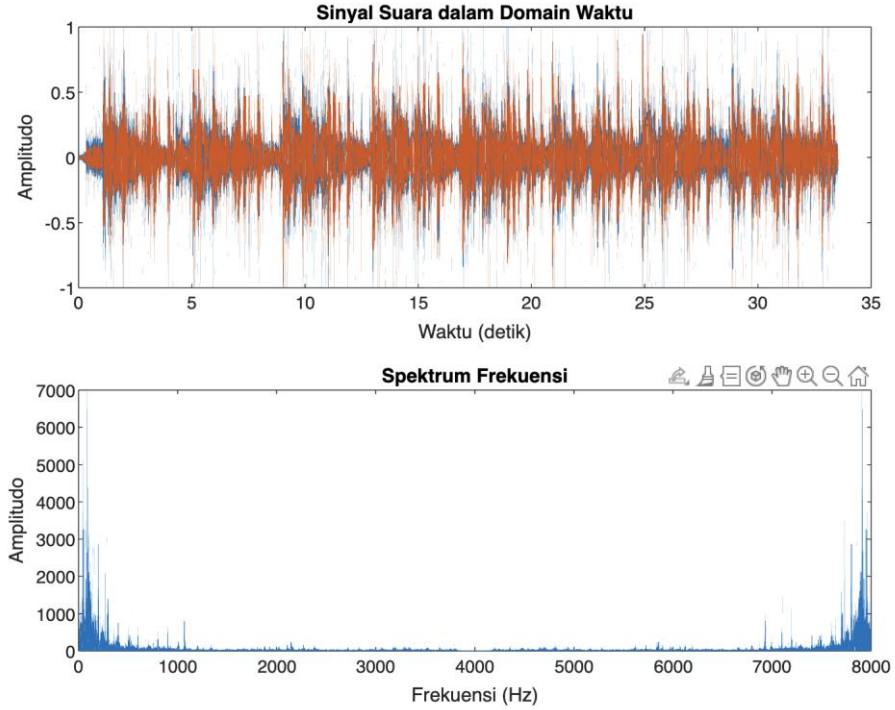
Dalam contoh di atas, file suara "audio.wav" dibaca menggunakan fungsi **audioread**. Kemudian, sinyal suara ditampilkan dalam domain waktu menggunakan plot. Selanjutnya, FFT diaplikasikan pada sinyal suara dengan menggunakan fungsi **fft**. Hasil FFT ditampilkan sebagai spektrum frekuensi dengan menggunakan plot. Akhirnya, frekuensi dominan ditemukan dengan mencari amplitudo maksimum dalam spektrum frekuensi.

Studi Kasus

```
% Baca file suara
[soundData, Fs] = audioread('audio.wav');

% Ambil satu saluran (jika suara stereo)
soundData = soundData(:, 1);

% Panjang sinyal
N = length(soundData);

% Hitung FFT
fftData = fft(soundData);

% Hitung amplitudo spektrum frekuensi
amplitude = abs(fftData);

% Hitung sumbu frekuensi
f = (0:N-1)*(Fs/N);

% Plot spektrum frekuensi
plot(f, amplitude);
xlabel('Frekuensi (Hz)');
ylabel('Amplitudo');
title('Spektrum Frekuensi');

% Menampilkan frekuensi dominan
[maxAmplitude, idx] = max(amplitude);
dominantFrequency = f(idx);
fprintf('Frekuensi dominan: %.2f Hz\n', dominantFrequency);
```

Output:

Frekuensi dominan: 88.73Hz



Pada contoh di atas, langkah-langkah yang dilakukan adalah sebagai berikut:

1. Membaca file suara menggunakan fungsi **audioread** dalam MATLAB. File suara disimpan dalam variabel **soundData** dan frekuensi sampel disimpan dalam variabel **Fs**.
2. Jika file suara memiliki dua saluran (stereo), kita dapat memilih salah satu saluran menggunakan **soundData(:, 1)**.
3. Menghitung panjang sinyal dengan **length(soundData)**.
4. Menggunakan fungsi **fft** untuk menghitung FFT dari sinyal suara. Hasil FFT disimpan dalam variabel **fftData**.
5. Menghitung amplitudo spektrum frekuensi dengan mengambil nilai absolut dari **fftData**.
6. Menghitung sumbu frekuensi dengan menggunakan rumus **(0:N-1)*(Fs/N)**. Ini akan memberikan array frekuensi yang sesuai dengan amplitudo.
7. Membuat plot spektrum frekuensi menggunakan fungsi **plot**. Sumbar x adalah frekuensi dan sumbar y adalah amplitudo.
8. Mencari frekuensi dominan dengan mencari nilai maksimum amplitudo menggunakan **max** dan mencari indeksnya menggunakan **idx**. Frekuensi dominan adalah nilai frekuensi pada indeks tersebut.
9. Menampilkan frekuensi dominan menggunakan **fprintf**.