SHARIF UNIVERSITY OF TECHNOLOGY,ELECTRICAL ENGINEERING DEPARTMENT

EEG signal processing

Computer Assignment 1

Ali Jabbari

10/30/2023

INTRODUCTION

QUESTION 1: GENERATING THE SIGNAL

I begin by generating a signal with a time-varying frequency. The signal's frequency at each moment follows a quadratic function: $f(t) = f_0 + \beta t^2$. I set $f_0 = 100$ Hz and $\beta = 100$.

a) Signal Generation

I generate a signal of duration 2 s sampled at 1000 Hz. MATLAB code to generate the signal is as follows:

```
% Parameters
fs = 1000;          % Sampling frequency (Hz)
T = 2;              % Signal duration (s)
t = 0:1/fs:T;
f0 = 100;           % Initial frequency (Hz)
beta = 100;         % Frequency increase rate

% Generate the signal
f_t = (f0 + beta * t.^2);
signal = chirp(t, 100, 2, 500, 'quadratic');
```

b) Window Functions

I generate four window functions: rectangular, triangular, Gaussian, and Hamming, all of length 128. Let's compare them in terms of their time and frequency domain characteristics. MATLAB code to generate and compare window functions is as follows:

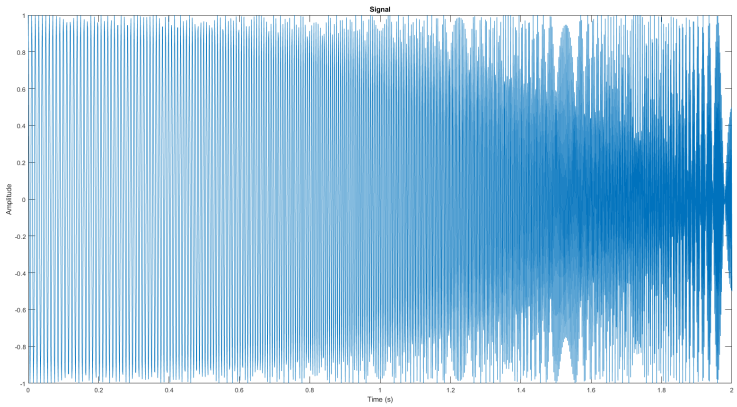

Figure 0.1: Generated Signal

```
% Define window length
L = 128;

% Generate window functions
rect_window = rectwin(L);
triangular_window = triang(L);
gauss_window = gausswin(L);
hamming_window = hamming(L);
```

Window Functions

Let's explore the four window functions:

- **Rectangular Window**: A simple window with constant amplitude.

$$w[n] = 1$$

- **Triangular Window (Triang)**: A triangular-shaped window that smoothly decreases towards its edges.

$$w[n] = 1 - \left| \frac{n - (N-1)/2}{(N-1)/2} \right|$$

- **Gaussian Window (Gausswin)**: A bell-shaped window that is often used to reduce spectral leakage.

$$w[n] = e^{-\left(\frac{n-(N-1)/2}{0.42(N-1)/2}\right)^2}$$

- **Hamming Window**: Another window with a bell-like shape that offers reduced side lobes.

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$

Question 2: Time-Frequency Analysis

Next, I perform time-frequency analysis of the signal using the Short-Time Fourier Transform (STFT) method.

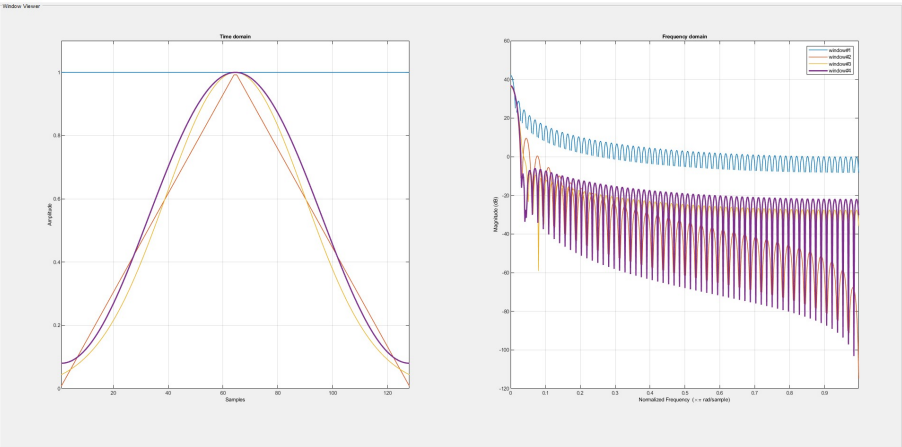

Figure 0.2: Comparison of Different Window Functions

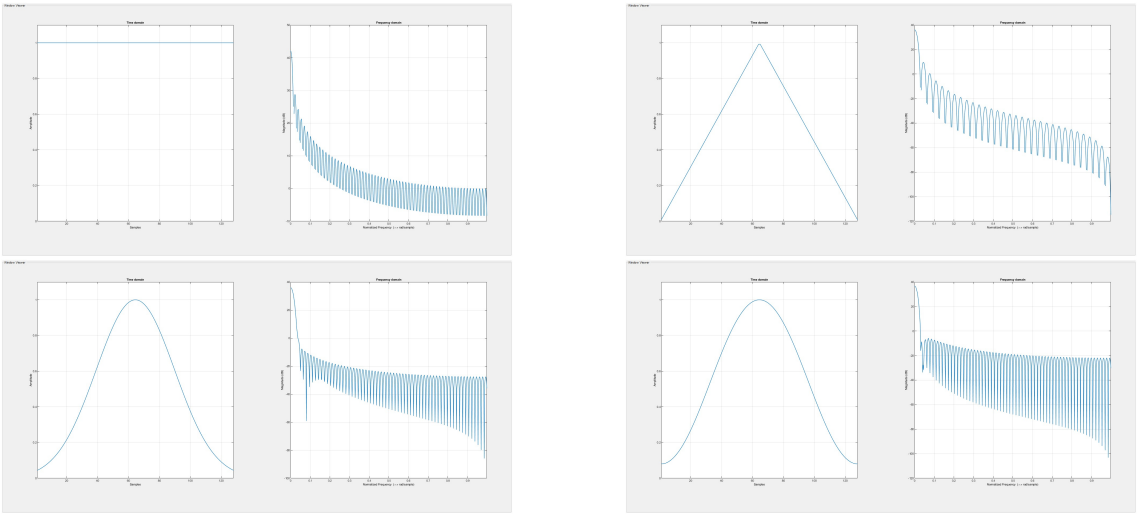

Figure 0.3: Comparison of Different Window Functions in Time and Frequency

c) Spectrogram Using Different Windows

I calculate the spectrogram using four different windows (rectangular, triangular, Gaussian, and Hamming) and compare the results. MATLAB code for calculating and plotting the spectrograms is as follows:

```
% Parameters for STFT
nfft = L;       % Number of DFT points
noverlap = 0;  % Number of overlapping points

% Plot the spectrograms
for i = 1:4
    switch i
        case 1
            window = rect_window;
            window_name = 'Rectangular Window';
        case 2
            window = triangular_window;
            window_name = 'Triangular Window';
```

```
        case 3
            window = gauss_window;
            window_name = 'Gaussian Window';
        case 4
            window = hamming_window;
            window_name = 'Hamming Window';
    end

    [S, F, T, P] = spectrogram(signal, window, noverlap, nfft, fs);

    % Plot the spectrogram
    figure;
    imagesc(T, F, 10 * log10(abs(P)));
    title(['Spectrogram using ', window_name]);
    xlabel('Time (s)');
    ylabel('Frequency (Hz)');
    axis xy;
    colormap('jet');
    colorbar;
end
```

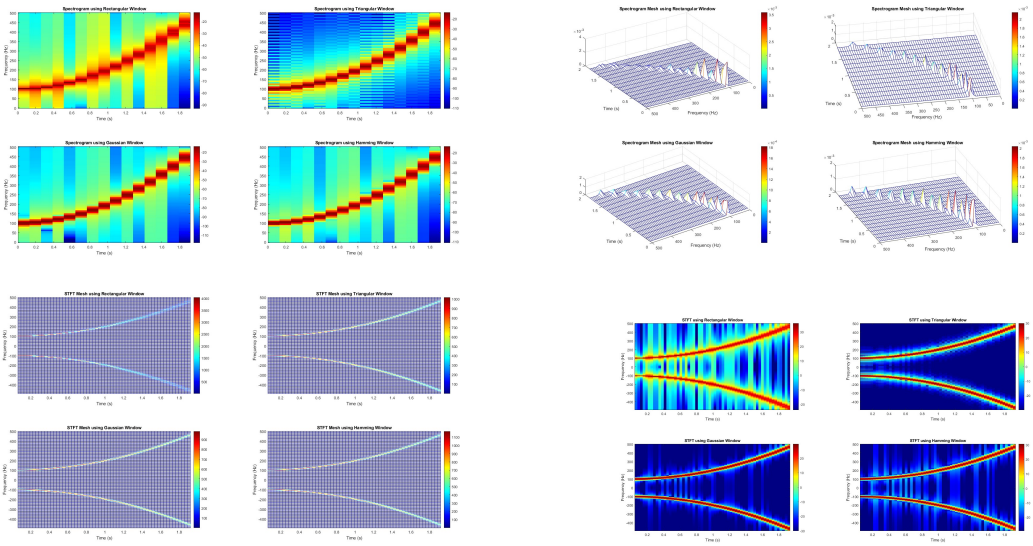

Figure 0.4: Spectrograms Using Different Windows and Different plots (Heat Map of Spectrogram ,Mech plot of spectrogram ,Mech plot of SFTF ,Heatmap of STFT)

Effects of Window Functions

The choice of the window function has a significant impact on the spectrogram. Here's how each window function affects the analysis:

- **Rectangular Window**: The rectangular window has high sidelobes in the frequency domain, leading to spectral leakage. It provides poor frequency resolution but has the best time localization.
- **Triangular Window**: The triangular window has a smoother transition from the main lobe to the sidelobes, reducing spectral leakage compared to the rectangular window. It offers a trade-off between frequency resolution and time localization.
- **Gaussian Window**: The Gaussian window further reduces sidelobes and spectral leakage, resulting in better frequency resolution. However, it has a wider main lobe in the time domain.
- **Hamming Window**: The Hamming window provides reduced sidelobes and spectral leakage while maintaining reasonable time localization. It is a good compromise between frequency resolution and time localization.

d) Varying Overlapping Points

I investigate the effect of changing the number of overlapping points (Noverlap) on the spectrogram using the triangular window. MATLAB code for varying the overlapping points is as follows:

```
% Values for Noverlap
Noverlaps = [0, 64, 127];

% Plot spectrograms with different Noverlap values
for i = 1:numel(Noverlaps)
    nooverlap = Noverlaps(i);

    [S, F, T, P] = spectrogram(signal, triangular_window, nooverlap, nfft, fs);

    % Plot the spectrogram
    figure;
    imagesc(T, F, 10 * log10(abs(P)));
    title(['Noverlap = ', num2str(nooverlap), ' with triangular window']);
    xlabel('Time (s)');
    ylabel('Frequency (Hz)');
    axis xy;
    colormap('jet');
    colorbar;
end
```

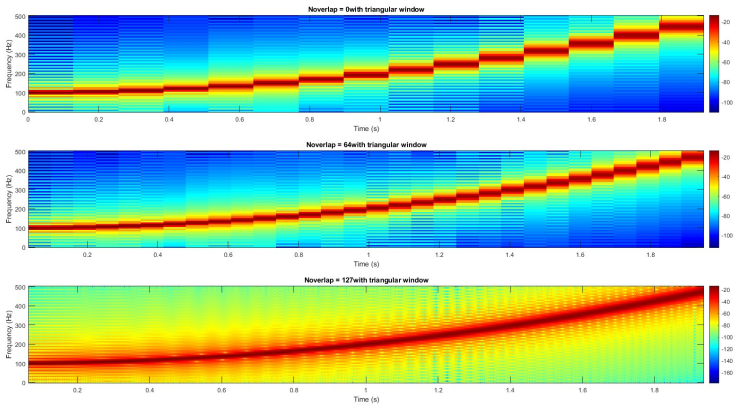

Figure 0.5: Effect of Varying Overlapping Points on Spectrogram

Effects of Overlapping Points

The number of overlapping points (Noverlap) affects the trade-off between time resolution and frequency resolution in the spectrogram. Here's how changing Noverlap influences the analysis:

- **Noverlap = 0**: No overlap between adjacent windows results in the highest time resolution but lower frequency resolution. Each window provides information at distinct time points but with reduced spectral information.
- **Noverlap = 64**: A moderate overlap balances time and frequency resolution. Overlapping windows provide better frequency information while maintaining reasonable time localization.
- **Noverlap = 127**: Maximum overlap between windows results in the best frequency resolution but sacrifices time resolution. Spectral information is preserved, but time localization is reduced.

e) Varying Window Length

I study the impact of changing the window length on the spectrogram with Noverlap fixed at L-1. MATLAB code for varying the window length is as follows:

```
% Values for window length
L_values = [32, 128, 512];
Noverlap = L_values - 1;

% Plot spectrograms with different window lengths
for i = 1:3
    % Spectogram using triangular window
    win = triang(L_values(i));
    [S, F, T, P] = spectrogram(signal, win, Noverlap(i), nfft(i), fs, 'yaxis');

    % Plot the spectrogram
    figure;
    imagesc(T, F, 10 * log10(abs(P)));
    str = sprintf('Spectrogram of Quadratic Chirp with Triangular Window & L = %d, Noverlap
    title(str);
    xlabel('Time (s)');
    ylabel('Frequency (Hz)');
    axis xy;
    colormap('jet');
    colorbar;
end
```

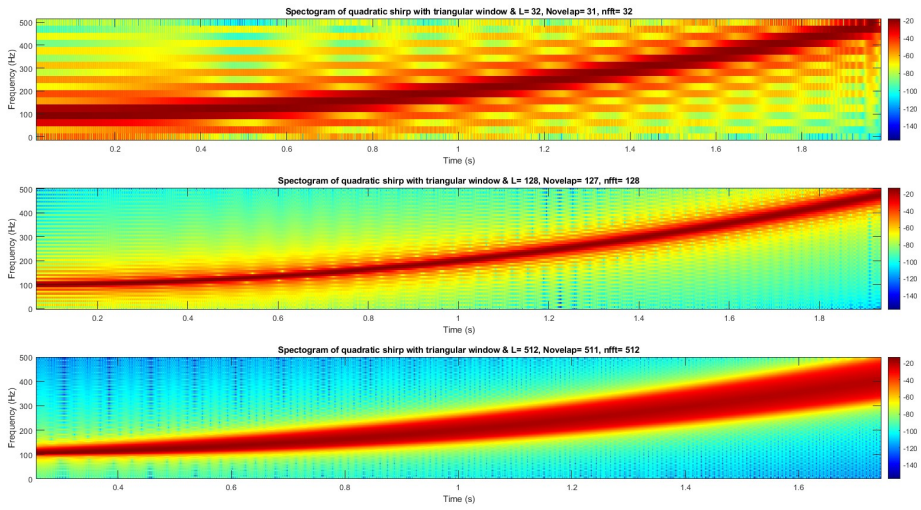

Figure 0.6: Effect of Varying Window Length Points on Spectrogram

Effects of Window Length

Changing the window length influences the time and frequency resolution of the spectrogram. Here's how different window lengths affect the analysis:

- **Window Length = 32**: Shorter windows result in better time resolution but poorer frequency resolution. This means that you can detect rapid changes in the signal's frequency but have a limited ability to distinguish closely spaced frequency components.
- **Window Length = 128**: A medium-length window balances time and frequency resolution. It provides a reasonable trade-off between the two aspects, making it a common choice for general analysis.
- **Window Length = 512**: Longer windows provide better frequency resolution but sacrifice time resolution. These windows are useful for analyzing signals with fine frequency details but may lose information about rapid changes in the signal.

f) Varying DFT Points (nfft)

I investigate the effect of changing the number of Discrete Fourier Transform (DFT) points (nfft) on the spectrogram with a fixed window length and overlapping points. MATLAB code for varying nfft is as follows:

```
% Values for nfft
L = 128;
nffts = [L, 2 * L, 4 * L];

% Plot spectrograms with different nfft values
for i = 1:numel(nffts)
    nfft = nffts(i);
    nooverlap = L / 2;

    [S, F, T, P] = spectrogram(signal, hamming_window, nooverlap, nfft, fs);

    % Plot the spectrogram
    figure;
    imagesc(T, F, 10 * log10(abs(P)));
    title(['nfft = ', num2str(nfft)]);
    xlabel('Time (s)');
    ylabel('Frequency (Hz)');
    axis xy;
    colormap('jet');
    colorbar;
end
```

Effects of DFT Points (nfft)

The choice of the number of DFT points (nfft) impacts the spectrogram's frequency resolution. Here's how different nfft values affect the analysis:

- **nfft = 128**: Using a lower number of DFT points results in lower frequency resolution. The spectrogram shows frequency components with larger bins, making it difficult to distinguish closely spaced frequencies.

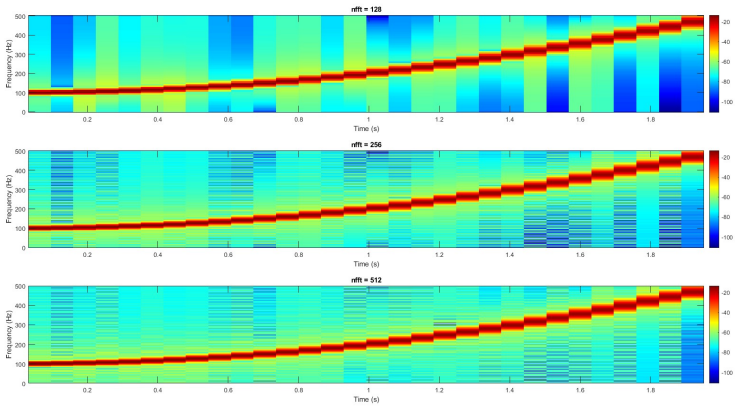

Figure 0.7: Effect of Varying DFT Points (nfft) on Spectrogram

- **nfft = 256**: Doubling the number of DFT points improves frequency resolution. The spectrogram can distinguish between closer frequency components, providing better spectral information.
- **nfft = 512**: Further increasing the number of DFT points enhances frequency resolution. The spectrogram becomes more capable of resolving fine frequency details and is useful for analyzing signals with closely spaced components.

CONCLUSION

In this section, I explored some signal processing techniques, including signal generation, window functions, and time-frequency analysis using the Short-Time Fourier Transform (STFT). I observed the effects of different window functions, overlapping points, window length, and the number of DFT points on the spectrogram's time and frequency resolution. The choice of window functions and parameter values significantly influences the analysis, and researchers should select them based on the specific characteristics of signals and the desired outcomes of the analysis.

ANALYSIS OF AN EEG SIGNAL

In this section, I analyze an EEG signal with a sampling frequency of 256 Hz. The EEG signal is processed through various techniques, including time domain analysis, frequency domain analysis, low-pass filtering, downsampling, zero-padding, and the Discrete Fourier Transform (DFT). Each section includes MATLAB code, descriptions, and figures to visualize the results.

0.1 Time Domain Analysis

I begin by examining the EEG signal in the time domain and Frequency content with Spectrogram. Figure 0.8 shows the time-domain plot of the EEG signal.

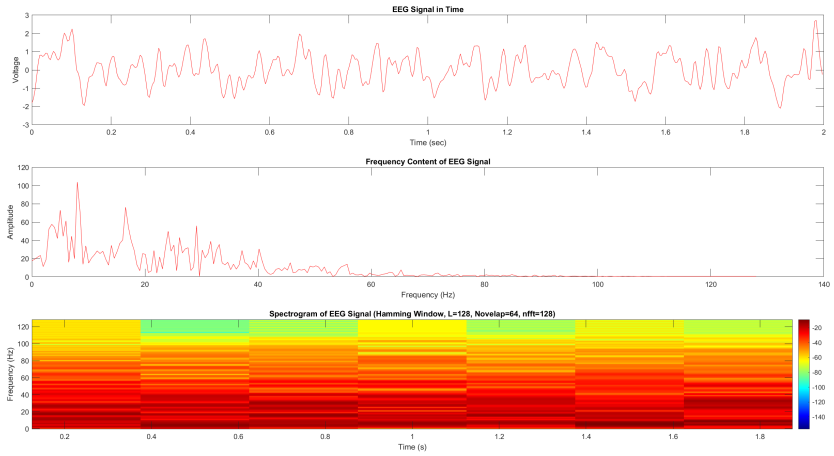

Figure 0.8: EEG Signal in Time Domain and Frequency and Spectrom of Signal

0.2 Frequency Domain Analysis

Next, I perform a Fourier Transform of the EEG signal to analyze its frequency components. Figure **??** displays the frequency content of the EEG signal.
To calculate the FFT, I use the following MATLAB code:

```
% MATLAB code for Fourier Transform
fx = fft(EEG_signal');
m = numel(fx);
fx = abs(fx(1:floor(m/2)));
f = linspace(0, fs/2, floor(m/2));
```

0.3 Short Time Fourier Transform (STFT)

I apply the Short Time Fourier Transform (STFT) to the EEG signal using a Hamming window with specific parameters (Hamming Window, $L = 128$, Novelap = 64, nfft = 128). Figure 0.9 shows the spectrogram.
To create the spectrogram, I use the following MATLAB code:

```
% MATLAB code for Spectrogram
L = 128;
wind = hamming(L);
```

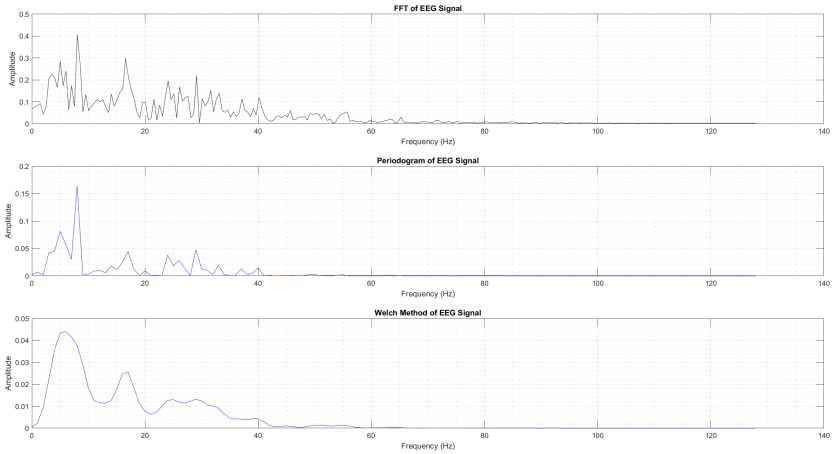

Figure 0.9: Frequency Content of EEG Signal with STFT and FFT and Welch method

```
Noverlap = L/2;
nfft = floor(m/2);
spectrogram(EEG_signal, wind, Noverlap, nfft, fs, 'yaxis');
```

0.4 Discussion

In the time domain analysis, I observe the raw EEG signal, and in the frequency domain analysis, I identify its frequency components. The STFT provides a time-frequency representation of the signal.

Low-Pass Filtering and Downsampling

In this section, I apply low-pass filtering to the EEG signal and downsample it. Figure 0.10 shows the downsampled EEG signal and its frequency content.

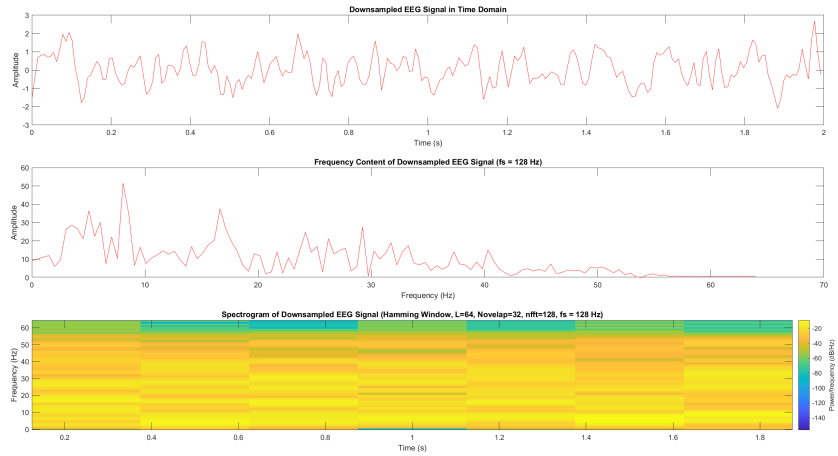

Figure 0.10: Downsampeled EEG Signal in Time Domain and Frequency and Spectrom of Signal

The low-pass filtering is implemented using the Butterworth filter, and downsampling is done by a specified factor. The resulting signal is named `EEG_ds`.
MATLAB code for low-pass filtering and downsampling:

```
[b, a] = butter(3, 64 / (fs/2), 'low');
filtered_EEGsig = filtfilt(b, a, EEG_signal);

M = 2;  % Downsampling factor
EEG_ds = decimate(filtered_EEGsig, M);
```

0.5 Spectrogram of Downsampled EEG Signal

I generate the spectrogram of the downsampled EEG signal with different parameters (Hamming Window, $L = 64$, Novelap = 32, nfft = 128, $fs = 128$ Hz). Figure 0.11 illustrates the spectrogram.

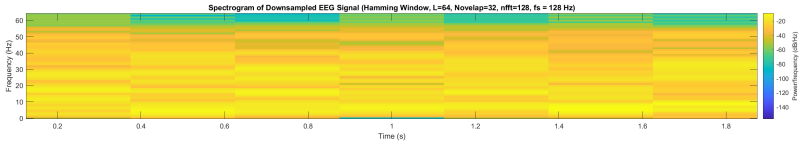

Figure 0.11: Spectrogram of Downsampled EEG Signal

Analysis with Different Downsampling Factors

In this section, I explore the effects of various downsampling factors (1, 2, 3, 4, 6, 8, 10) on the EEG signal. I compare the original EEG signal with the downsampled versions. Figure 1.1 presents these comparisons for different downsampling factors.

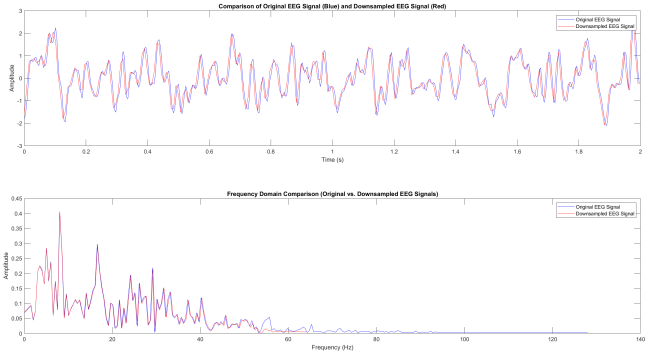

Figure 0.12: Comparison of Original EEG Signal and Downsampled EEG Signals (Different Downsampling Factors)

I examine the impact of different downsampling factors on the EEG signal, providing insights into how signal properties change with reduced sampling rates.
In this Figure 1.2, we can see that by increasing the downsampling rate, more frequency information is removed and the downsampling factor 2 was chosen to continue the work. This downsampling process results in retaining the primary information and frequency peaks. Comparatively, it offers a smoother frequency spectrum while maintaining critical features of the signal.

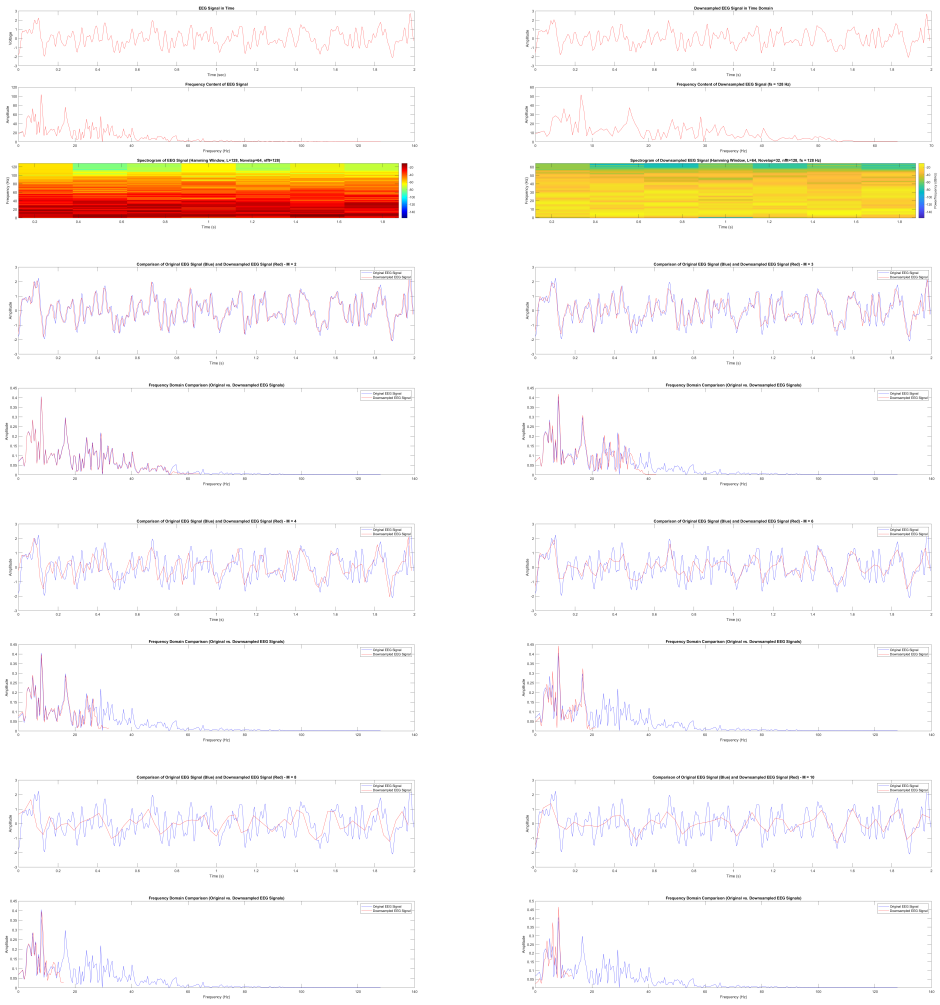

Figure 0.13: Downsampling with different M(Downsampling Factor)

0.6 Zero Padding and DFT

Next, I discuss the application of zero padding to the downsampled EEG signal. I analyze the DFT using different fractions of N and observe the effects on the frequency domain representation.

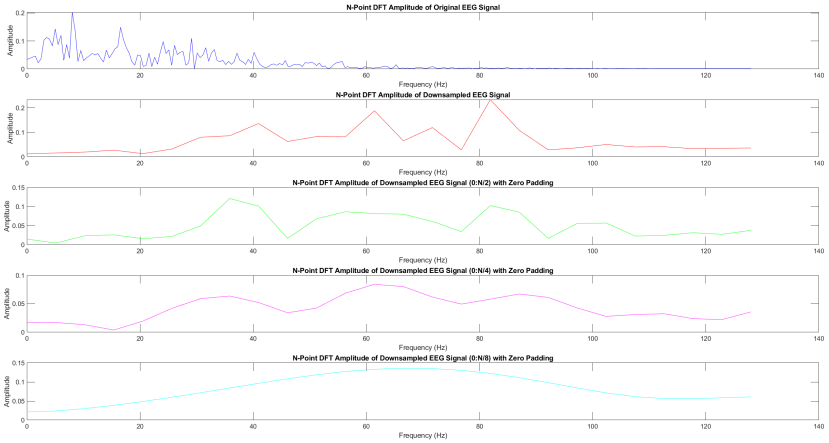

Figure 0.14: DFT Analysis of EEG Signals without Zero Padding ($N/2$, $N/4$, $N/8$)

we explore the application of zero padding to the downsampled EEG signal and analyze the Discrete Fourier Transform (DFT) using different fractions of the signal length. The objective is to observe how zero padding influences the frequency domain representation. We then performed downsampling at various rates, including N, $N/2$, $N/4$, and $N/8$. The results demonstrated that reducing the number of sampled points in the DFT led to a loss of information, especially frequency peaks present in the original signal.

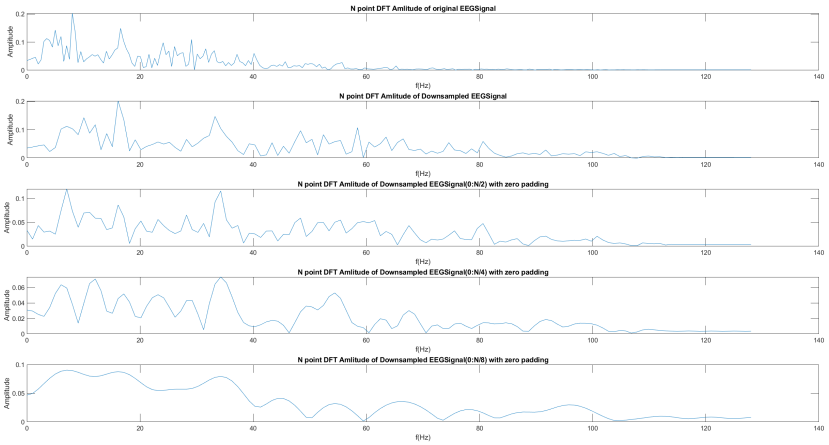

Figure 0.15: DFT Analysis of EEG Signals with Zero Padding ($N/2$, $N/4$, $N/8$)

Figure 1.4 presents the DFT amplitude of the original EEG signal, downsampled signal, and the downsampled signal with different levels of zero padding, including $N/2$, $N/4$, and $N/8$. By analyzing these figures, we can gain insights into how zero padding affects the spectral information and frequency domain representation.
The MATLAB code used for this analysis is as follows:

```
% Original EEG Signal DFT
L = length(EEG_signal);
fft_EEGSignal = fft(EEG_signal, L) / L;
fft_EEGSignal = abs(fft_EEGSignal(1:floor(L/2)));
```

```
f = linspace(0, fs/2, floor(L/2));

% DFT of Downsampled EEG Signals with Zero Padding
N = length(EEG_ds);
fft_EEG_ds1 = fft(EEG_ds, N) / N;

EEG_ds2 = zeros(1, N);
N1 = N/2;
EEG_ds2(1:N1) = EEG_ds(1:N1);
fft_EEG_ds2 = fft(EEG_ds2, N) / N;

EEG_ds3 = zeros(1, N);
N2 = N/4;
EEG_ds3(1:N2) = EEG_ds(1:N2);
fft_EEG_ds3 = fft(EEG_ds3, N) / N;

EEG_ds4 = zeros(1, N);
N3 = N/8;
EEG_ds4(1:N3) = EEG_ds(1:N3);
fft_EEG_ds4 = fft(EEG_ds4, N) / N;
```

We observed that by increasing the number of points with zero-padding, each figure corresponding to the DFT analysis had more detailed information. Zero-padding, besides increasing the number of points in the frequency domain, also improves the continuity of the spectrum. However, it doesn't affect the frequency resolution. The frequency resolution primarily depends on the number of samples and the sampling frequency.

CONCLUSION

In this comprehensive analysis of the EEG signal, we explored various signal processing techniques, including time domain analysis, frequency domain analysis, low-pass filtering, downsampling, and zero-padding. Each of these techniques plays a crucial role in understanding the signal and its transformation.

When downsampling the EEG signal, it's essential to first pass it through a low-pass filter with a cutoff frequency of 64 Hz. This step helps preserve most of the signal's frequency information while avoiding aliasing effects. The majority of the signal's spectral content lies within the $[0, 60]$ Hz range, and the low-pass filtering strategy effectively retains this information.

After the low-pass filtering, we decimated the signal to a 128 Hz sampling rate, which is more than twice the maximum frequency content in the original signal (60 Hz). This downsampling process results in retaining the primary information and frequency peaks. Comparatively, it offers a smoother frequency spectrum while maintaining critical features of the signal.

We then performed downsampling at various rates, including N, $N/2$, $N/4$, and $N/8$. The results demonstrated that reducing the number of sampled points in the DFT led to a loss of information, especially frequency peaks present in the original signal.

In the final part, we applied zero-padding. We observed that by increasing the number of points with zero-padding, each figure corresponding to the DFT analysis had more detailed information. Zero-padding, besides increasing the number of points in the frequency domain, also improves the continuity of the spectrum. However, it doesn't affect the frequency resolution. The frequency resolution primarily depends on the number of samples and the sampling frequency.

In conclusion, the techniques demonstrated in this report serve as vital tools for understanding EEG signals and other time-domain signals. Careful selection of sampling rates, filtering, and zero-padding can help balance between data preservation, computational efficiency, and spectral analysis. These tools are essential for various applications, such as brain activity monitoring and medical diagnostics.