Audio processing

introduction

The goal of this project is to synthesise an audio signal that includes many frequency components and noise, and then add digital signal processing techniques to analyse and filter the signal. The goal is to understand the usage and application of the Discrete Fourier Transform (DFT) in signal processing (here we work an audio and reduce the noise of the input signal using filtering .)

DFT is a mathematical technique which we use to convert a discrete signal from the time domain to the frequency domain. This transform is important in signal processing because it enables us to analyse signal's frequency content, we can do tasks such as filtering, compression, and noise reduction.

DFT converts a signal in the time domain to a signal in the frequency domain and shows it with a sequence of complex numbers , each complex number relating to specific frequency which magnitude indicating the amplitude and its phase indicating the phase shift of that frequency component.This frequency-domain representation is necessary for identifying and modifying specific frequencies in a signal.

METHODOLOGY:

In this project, we synthesized an audio signal. The synthesized signal consists of three sine waves with (500 Hz, 1000 Hz, and 1500 Hz) frequencies and added noise. The generated signal with noise is plotted and saved as a .wav file, and played using MATLAB functions.

```
Fs = 8000;
t = 0:1/Fs:1-1/Fs;
f1 = 500;
f2 = 1000;
f3 = 1500;
noise_amp = 0.2;
signal = sin(2*pi*f1*t) + 0.5*sin(2*pi*f2*t) + 0.3*sin(2*pi*f3*t);
noise = noise_amp * randn(size(t));
audio = signal + noise;
plot(t, audio);
audiowrite('originalaudio.wav', audio, Fs);
sound(audio, Fs);
```

The function to transform from time-domain signal to frequency-domain is as follows :

```
function X = DFT(x)
    N = length(x);
    X = zeros(N, 1);
    for k = 0:N-1
        for n = 0:N-1
            X(k+1) = X(k+1) + x(n+1) * exp(-1i * 2 * pi * k * n / N);
        end
    end
end
```

Filter Design and Application :

In this section,focus is on identifying the noise components in the signal and designing different and varies filters to remove or reduce them. First, we use DFT to move the signal into the frequency-domain. Then, we design and apply appropriate filters.

```
band_low = 800;
band_high = 1200;

low = 800;
high = 1200;

%filtering

HLow = (f < low)';
HHigh = (f > high)';
HBand = (f > band_low & f < band_high)';
```

A band-pass filter allows frequencies within a specified band to pass through while eliminating frequencies outside that band. In this case, the band-pass filter is set to allow frequencies between 800 Hz and 1200 Hz to pass and to eliminate frequencies outside this band.
HBand is 0 for all frequencies except between 800 and 1200 Hz and for this range is 1.

A low-pass filter allows frequencies lower than a specified cutoff frequency to pass through while eliminating frequencies higher than that cutoff. In this case, the cutoff frequency is 800 Hz, so it means that the low-pass filter will allow frequencies below 800 Hz to pass and will eliminate frequencies above 800 Hz.
HLow is 1 for frequencies below 800 Hz and 0 for frequencies outside this range

High pass filter and HHigh is the reverse of the low pass filter pass frequencies above the given frequency.

the following code ensures that only the frequencies within the filter range remain and other frequencies are removed. Here, X is the Fourier transform of the original signal, and with these multiplications, only the frequencies within the filter range remain and the other frequencies are eliminated.

```
YLow = Y .* H_low;
YHigh = Y .* H_high;
YBand = Y .* H_band;
```

Low-pass, high-pass, and band-pass filters are designed and applied to the signal in the frequency domain. Then, using the inverse Fourier transform (synthesis), the signals are returned to the time domain and played back.

IDFT function is in charge to do this and the following lines are usage of this function

```
function x = IDFT(X)
   N = length(X);
   x = zeros(N, 1);
   for n = 0:N-1
      for k = 0:N-1
         x(n+1) = x(n+1) + X(k+1) * exp(1i * 2 * pi * k * n / N);
      end
      x(n+1) = x(n+1) / N;
   end
end
```

Usage of IDFT function:

```
lowFilterAudio= IDFT(YLow);
highFilterAudio = IDFT(YHigh);
bandFilterAudio= IDFT(YBand);
```

Outputs of this function are time-domain signals .

This part of the code normalizes the filtered audio signals to ensure that they are within a suitable range for playback or further processing. It divides each sample by its maximum absolute value. This operation scales the signal intensity such that the maximum absolute value is one. This action ensures that the signal intensity is normalized and remains within a range that does not exceed the limits for sound playback or subsequent processing.

```
lowAudioToSave = real(lowFilterAudio) / max(abs(lowFilterAudio));
HighAudioToSave= real(highFilterAudio) / max(abs(highFilterAudio));
BandAudioToSave = real(bandFilterAudio) / max(abs(bandFilterAudio));
```

Result:

Input file without filtering :

PLOT GENERATED SIGNAL

```
% polt the input signal

Y = DFT(audio');
a=Fs / length(Y);
b=0:length(Y)-1;
f = a*b;
figure;
plot(f, abs(Y));
title('Magnitude of the Audio Signal');
xlabel('"Hz"');
ylabel('measure');
```

Output file with different filters :

PLOT GENERATED SIGNAL

```
figure;
subplot(3,1,1);
plot(f, abs(DFT(lowAudioToSave)));
title('Low-pass Filtered');
xlabel('Hz');
ylabel('measure');

subplot(3,1,2);
plot(f, abs(DFT(HighAudioToSave)));
title('High-pass Filtered');
xlabel('Hz');
ylabel('measure');

subplot(3,1,3);
plot(f, abs(DFT(BandAudioToSave)));
title('Band-pass Filtered');
xlabel('Hz');
ylabel('measure');
```

High pass

- A high-pass filter allows frequencies higher than a specified cutoff frequency to pass through and removes lower frequencies.

Low pass

- A low-pass filter allows frequencies lower than a specified cutoff frequency to pass through and removes higher frequencies.

Band pass

- A band-pass filter allows frequencies within a specific range to pass through and removes frequencies outside that range.

Conclusion:

In this project, three types of filters (low-pass, high-pass, band-pass) were used to reduce noise in audio signals. Each of the filters successfully achieved a significant reduction in noise intensity in different frequency ranges, especially the low-pass filter which effectively removed low-frequency noise. The low-pass filters increased the clarity of the signal at low frequencies, while the high-pass filters led to a reduction in quality at low frequencies.

Insights gained from the project:

- Designing and using various filters to improve the quality and remove noise from audio signals.
- Understanding how the Fourier transform works in the frequency analysis of audio signals and its capabilities for noise reduction and quality improvement.
- Encountering challenges such as signal clipping and the need for proper normalization to maintain audio quality.