Digital Filter Design and Application using MATLAB

John Bobby

Contents

1 Objective

To design a digital filter using MATLAB's filter Designer tool and applying it to filter a mixture of sine waves. The FFT plot of the original and filtered signal is compared.

2 Filter Design Process

The filter is designed using filterDesigner tool in MATLAB. The filter coefficients are stored in .m file.

```
function Hd = lowpass
%LOWPASS Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 24.2 and Signal Processing Toolbox 24.2.
% Generated on: 01-Apr-2025 22:18:56

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

Fpass = 500;              % Passband Frequency
Fstop = 600;              % Stopband Frequency
Dpass = 0.057501127785;   % Passband Ripple
Dstop = 0.0001;           % Stopband Attenuation
dens  = 20;               % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop]/(Fs/2), [1 0], [Dpass, Dstop])
    ;

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```

Listing 1: lowpass.m that stores the filter details

2.1 Filter Parameters

- Type of Filter: LowPass
- F_{pass} : $500Hz$
- F_{stop} : $600Hz$
- Filter Order: Minimum
- Density Factor: 20
- Sampling Frequency: 48,000 Hz

2.2 Filter Magnitude Response

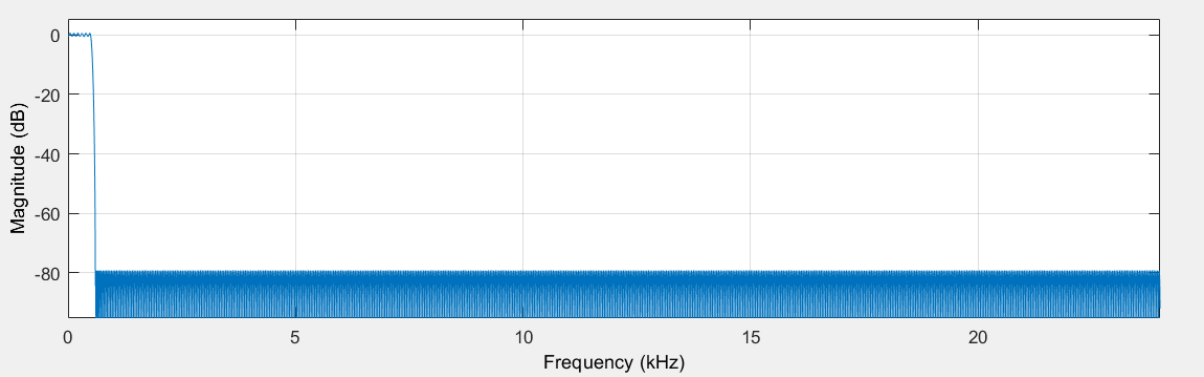

Figure 1: Magnitude Response for the low pass filter

From the above plot we can see that after $0.5kHz$ the gain reaches high negative values thus attenuating that signal in that region.

3 Signal Generation and Analysis

3.1 Signal Generation

A mixture of 10 sinusoidal signals of frequencies ranging from $100Hz$ to $1000Hz$ is generated.

```
Fs=48000;
T=1;
t=0:1/(Fs):1;

frequencies=[100,200,300,400,500,600,700,800,900,1000];

signal=zeros(size(t));

for i=1:length(frequencies)
    signal=signal+sin(2*pi*frequencies(i)*t);
end
```

Listing 2: singal generation in MATLAB

- As the sampling frequency is $4800Hz$, it means that 4800 samples are taken in one second. Thus the array t has a step size of $\frac{1}{F_s}$
- *signal* is a array of same size as t. The loop adds up all the individual frequency signals.

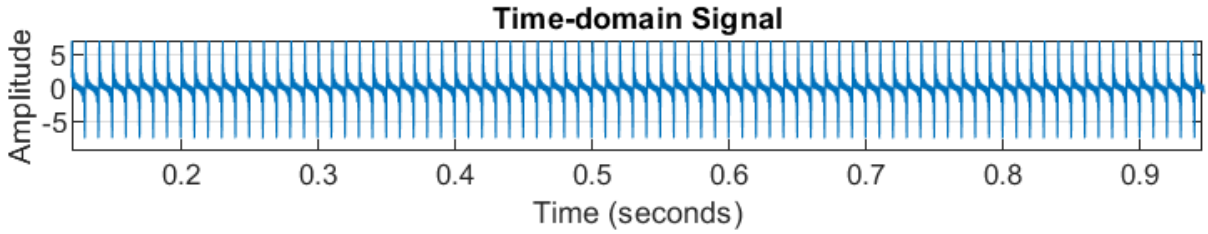

Figure 2: Generated Signal in time domain

3.2 Signal Spectral Analysis using FFT

The Fast Fourier Transform (FFT) is an efficient algorithm used to compute the Discrete Fourier Transform (DFT) and its inverse. The FFT algorithm reduces the computational complexity, making it much faster for large datasets.

```
Y=fft(signal);
f=0:Fs;
plot(f(1:Fs/2), abs(Y(1:Fs/2)));
```

Listing 3: computing FFT of the generated signal

- fft( ) function calculates the Fast Fourier Transform of a signal. Y is a complex array whose real part is the amplitude of the cosine wave and imaginary part is the amplitude of the sine wave. Only the magnitude of the complex number is plotted.
- The maximum frequency that can be measured properly when the sampling frequency is F_s is $\frac{F_s}{2}$.

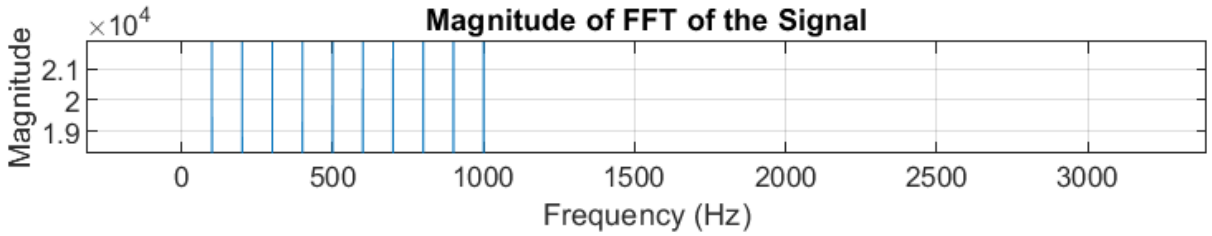

Figure 3: Magnitude of the FFT of generated signal

4 Signal Filtering

On saving the lowpass.m in the same directory we get an filter object Hd that has the filter coefficients.

```
Hd=lowpass();
filtered_signal=filter(Hd,signal);
plot(t,filtered_signal);
Y_filter=fft(filtered_signal);
subplot(4,1,4);
plot(f(1:Fs/2),abs(Y_filter(1:Fs/2)));
```

Listing 4: MATLAB code for applyig filter to signal

- filter(Hd,signal) applies the filter Hd to the signal.
- the time and frequency domain plots are plotted.

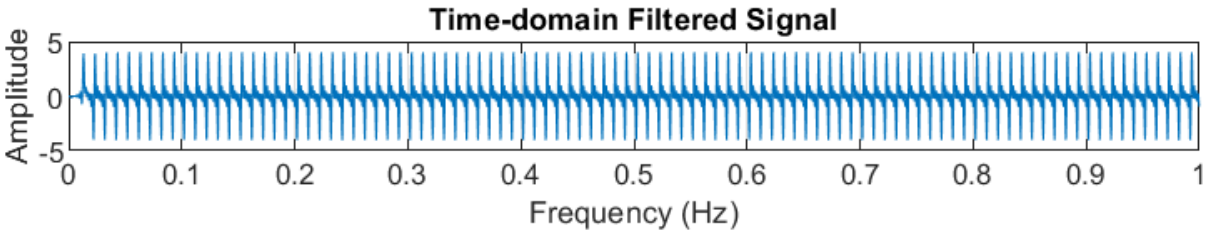

Figure 4: Time domain plot of the filtered signal

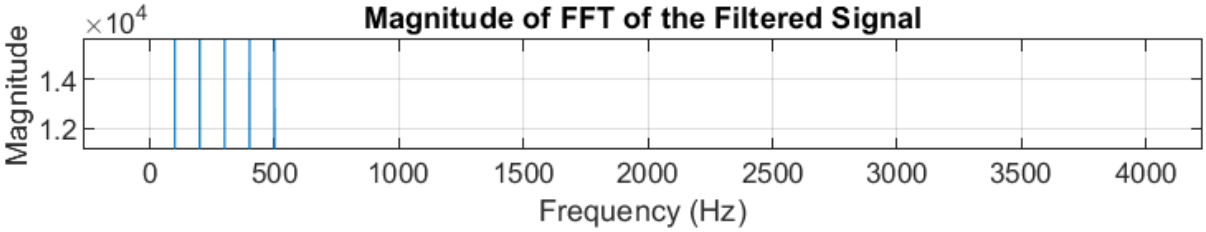

Figure 5: Magnitude of FFT of the filtered signal

Observations in Time Domain

Before Filtering (Original Signal):

- The original signal is a mixture of sinusoidal waves, with frequencies ranging from 100 Hz to 1000 Hz.
- The waveform might appear jagged or complex, with high-frequency oscillations or sharp changes in amplitude due to the higher-frequency components.

After Filtering (Filtered Signal):

- After applying the low-pass filter with a cutoff of 500 Hz, the high-frequency components (above 500 Hz) are attenuated or removed.
- The waveform becomes smoother, with fewer high-frequency fluctuations.
- High-frequency noise or sharp oscillations will be reduced or eliminated, resulting in a cleaner signal.

Observations in Frequency Domain (FFT)

Before Filtering (Original Signal):

- The FFT of the original signal shows peaks at all the frequencies present in the mixture, ranging from 100 Hz to 1000 Hz.

- The amplitude of these peaks corresponds to the strength of the respective sinusoidal components.

After Filtering (Filtered Signal):

- The FFT of the filtered signal shows a significant reduction or elimination of frequencies above 500 Hz.
- The amplitude of FFT at frequencies below 500Hz do not change that much.

5 Complete MATLAB Code

```
Fs=48000;
T=1;
t=0:1/(Fs):1;

frequencies=[100,200,300,400,500,600,700,800,900,1000];

signal=zeros(size(t));

for i=1:length(frequencies)
    signal=signal+sin(2*pi*frequencies(i)*t);
end
Y=fft(signal);
f=0:Fs;

figure;
subplot(4,1,1);
plot(t, signal);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Time-domain Signal');
grid on;

subplot(4,1,2);
plot(f(1:Fs/2), abs(Y(1:Fs/2)));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Magnitude of FFT of the Signal');
grid on;



Hd=lowpass();
filtered_signal=filter(Hd,signal);

subplot(4,1,3);
plot(t,filtered_signal);
xlabel('Frequency (Hz)')
ylabel('Amplitude');
title('Time-domain Filtered Signal');
grid on;

Y_filter=fft(filtered_signal);
subplot(4,1,4);
```

```
plot(f(1:Fs/2),abs(Y_filter(1:Fs/2)));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Magnitude of FFT of the Filtered Signal');
grid on;
```

Listing 5: Complete MATLAB code