Lab 4
October 3, 2016
Filters

INSTRUCTIONS:
All lab submissions include a written report and source code in the form of an m-file. The report contains all plots, images, and figures specified within the lab. All figures should be labeled appropriately. Answers to questions given in the lab document should be answered in the written report. ***The written report must be in PDF format***. Submissions are done electronically through **my.ECE**.

1 Ideal Filters

A filter $H_d(\omega)$ has can be decomposed into magnitude and phase components,i.e. $H_d(\omega) = |H_d(\omega)|e^{i\angle H_d(\omega)}$. For an ideal filter, $\angle H_d(\omega) = 0$. That is, there is no phase shift associated with any ω. The most 3 basic types of filters are highpass, lowpass, and bandpass filters. The magnitude response of each of these filters are:

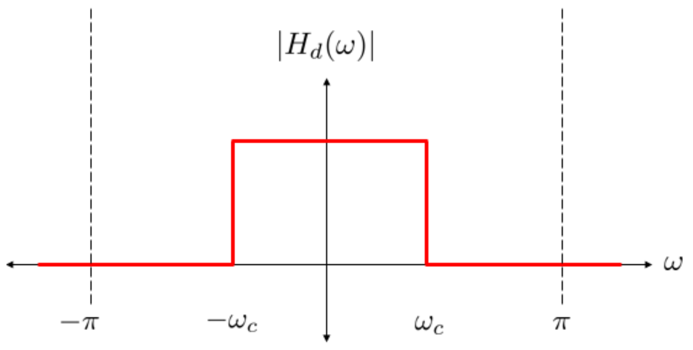


Figure 1: Ideal LPF magnitude response.

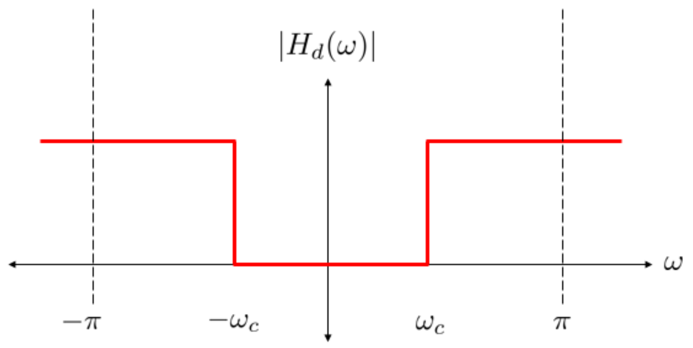


Figure 2: Ideal HPF magnitude response.

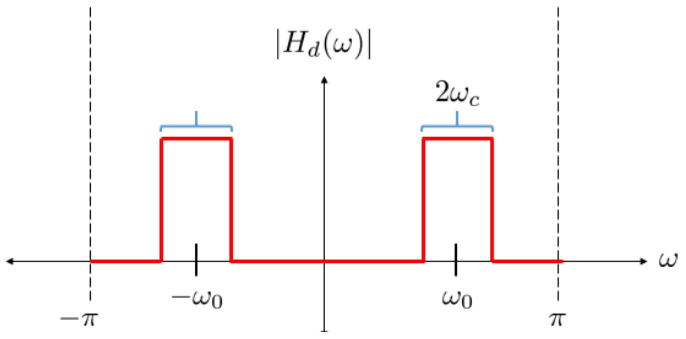


Figure 3: Ideal BPF magnitude response.

where the corresponding impulse response for each is

$$h(n) = \frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c}{\pi}n\right) \qquad \text{(Lowpass)}$$
$$h(n) = \delta(n) - \frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c}{\pi}n\right) \qquad \text{(Highpass)}$$
$$h(n) = \cos(\omega_0 n)\frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c}{\pi}n\right) \qquad \text{(Bandpass)}$$

and the definition of sinc is the same as MATLAB's.

Report Item: Implement the impulse response for the ideal lowpass, highpass, and bandpass filters. Let $n = -N : N$ for $N = 20$. Plot each impulse response using **stem** and the magnitude response using **plot**. Repeat for $N = 40$. How does the magnitude response differ from the ideal and why?

In general, an ideal filter cannot be realized due to the fact that it is infinite in length and non-causal. A remedy is to first perform a shift followed by a truncation. This makes the filter both finite and causal at the cost of adding ripples to the magnitude response and linear phase to the phase response. The formal name for these ripples is *Gibb's Phenomenon*. Truncation also affects the transition width of the filter. The transition width is the bandwidth between the passband and stopband edge frequencies. In general, the longer the filter is, the smaller the transition width.

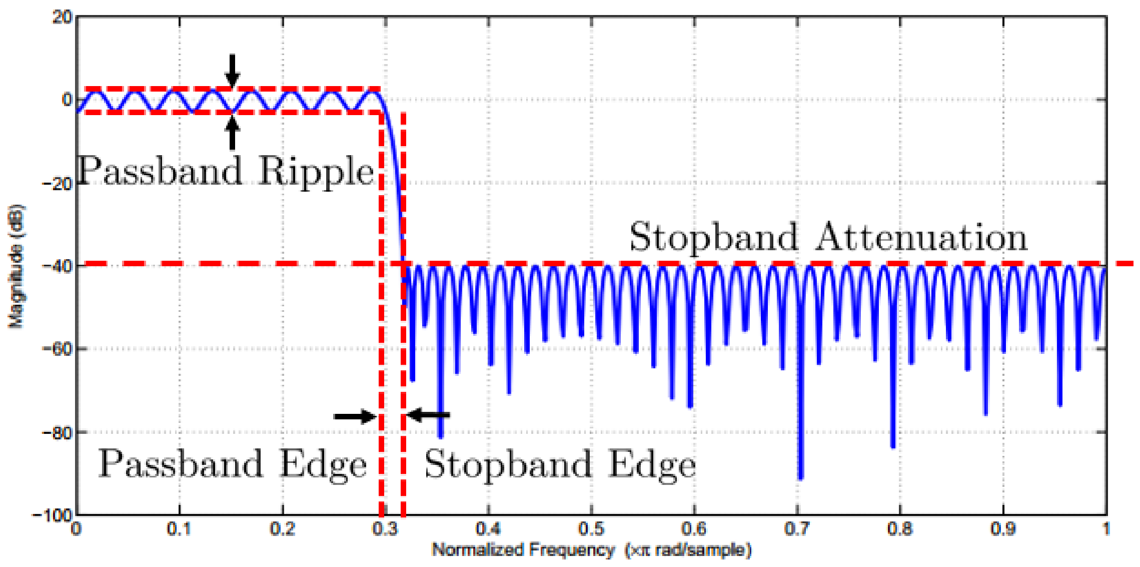


Figure 4: Filter metrics.

Passband ripple is the peak-to-peak ripple in the passband in dB. Stopband attenuation is the highest peak in the stopband. The transition width is the difference between the stopband and passband edges. The location for the stopband and passband edges are arbitrary and depend on the convention used.

Report Item: Load *impulseresponse.mat* which contains the impulse response of a filter h. Use **stem** to plot the impulse response. Find the magnitude response (in dB) and phase response of this filter and **plot** both. What are the stopband and passband ripples? What is the transition bandwidth?

2 Generalized Linear Phase FIR Filters

Filters can be separated into two classes: finite impulse response (FIR) and infinite impulse response (IIR). As implied by their names, FIR filters have a finite length impulse response and IIR filters have an infinite length impulse response. The transfer function for an FIR filter can be expressed as a polynomial of the form:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_{N-1}(x - N + 1) \tag{1}$$

$$H(z) = b_0 + b_1 z^{-1} + \cdots + b_{N-1} z^{-(N-1)} \tag{2}$$

where the output only depends on the input. Using the discrete-time delay property of z-transforms, the transfer function for FIR filters can easily be determined. FIR filters are inherently stable because they have no poles (except the ones at zero). A symmetric FIR filter always has linear phase. FIR filters can be designed using several methods. Some common approaches are the windowing method, frequency sampling method, and Parks-McClellan method.

Consider a filter with a real-valued, length N impulse response $\{h_n\}_{n=0}^{N-1}$ which has a DTFT of the form

$$H_d(\omega) = R(\omega) e^{i(\alpha - \omega M)} \tag{3}$$

where $R(\omega)$ is real-valued, α is a real-valued constant, and $M = \frac{N-1}{2}$. Note that

$$R(\omega) = \begin{cases} |R(\omega)| & \text{if } R(\omega) > 0 \\ e^{\pm j\pi}|R(\omega)| & \text{if } R(\omega) < 0. \end{cases}$$

Thus, a sign change in $R(\omega)$ results in a $\pm\pi$ jump in the phase of $H_d(\omega)$. Phase jumps of π may not be 'unwrapped' like jumps of 2π, but the filter $H_d(\omega)$ will still have generalized linear phase.

2.0.1 Type I Generalized Linear Phase

An FIR filter with real-valued impulse response $\{h_n\}_{n=0}^{N-1}$ as specified above has type I generalized linear phase if and only if it has even symmetry given by

$$h[n] = h[N-1-n] = \begin{cases} n = 0, 1, \ldots, N/2 - 1 & N \text{ even} \\ n = 0, 1, \ldots, (N-1)/2 & N \text{ odd.} \end{cases} \tag{4}$$

Considering Eq. 3, it may be proved that for type I generalized linear phase $\alpha = 0$ and $R(\omega)$ must be an even function (e.g. $R(\omega) = R(-\omega)$, a sum of cosine functions).

An example of type I generalized linear phase for an FIR filter is shown in Fig. 5.

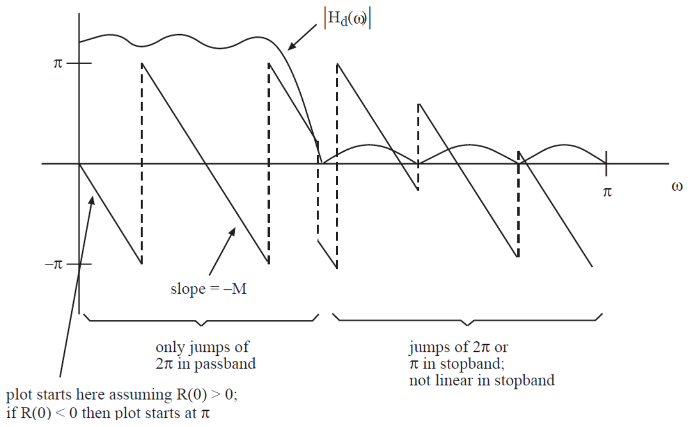


Figure 5

2.0.2 Type II Generalized Linear Phase

An FIR filter with real-valued impulse response $\{h_n\}_{n=0}^{N-1}$ has type II generalized linear phase if and only if it has odd symmetry given by

$$h[n] = -h[N-1-n] = \begin{cases} n = 0, 1, \ldots, N/2 - 1 & N \text{ even} \\ n = 0, 1, \ldots, (N-1)/2 & N \text{ odd} \end{cases} \tag{5}$$

Considering Eq. 3, it may be proved that for type II generalized linear phase $\alpha = \pm\frac{\pi}{2}$ and $R(\omega)$ must be an odd function (e.g. $R(\omega) = -R(-\omega)$, a sum of sine functions). If N is odd, $h[M]$ ($M = (N-1)/2$) must be zero to satisfy the odd symmetry condition.

3 Windowing Method

Let us begin with the ideal lowpass filter, shown below (left), with the desired frequency response $D(\omega)$. We might choose the filter coefficients to be the inverse DTFT of the ideal lowpass filter, $d[n] = \frac{\omega_c}{\pi}\text{sinc}[\omega_c n]$, as shown below (right).

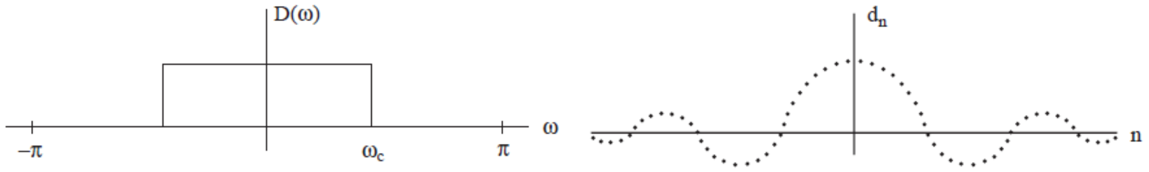


Figure 6

However, the sequence $d[n]$ is infinitely long and non-causal. To obtain an FIR filter, it is necessary to shift the sequence $d[n]$ and truncate it in time. The shift of the impulse response adds linear phase to the complex frequency response. The truncation causes sidelobes in the magnitude response, known as Gibb's phenomenon. These sidelobes can be reduced by windowing the impulse response (in the time domain), at the cost of increasing the transition bandwidth in the magnitude frequency response.

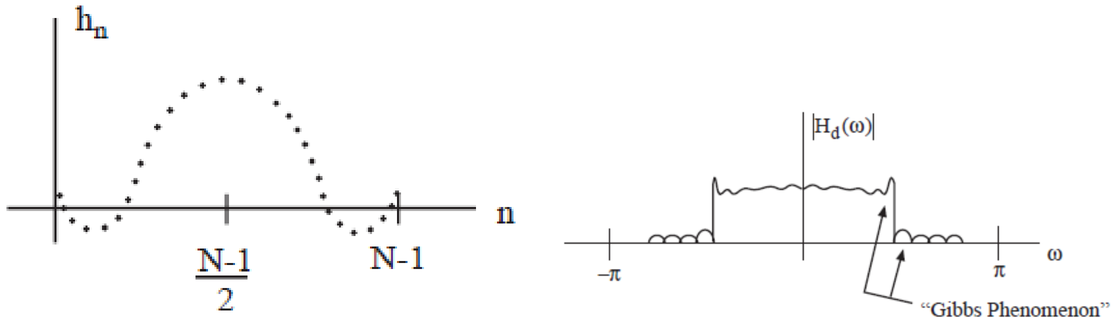


Figure 7

To design a generalized linear phase filter of coefficients $\{h_n\}_{n=0}^{N-1}$ such that $|H_d(\omega)|$ approximates the desired frequency response:

1. Let $G_d(\omega) = D(\omega)e^{-jM\omega}$, where $M = \frac{N-1}{2}$
2. Find $g[n] = \text{DTFT}^{-1}\{G_d(\omega)\}$
3. Let $h[n] = w[n]g[n]$, where $w[n]$ is the window function (e.g. Hamming window centered at $n = M$)

Report Item: Design and analyze the following generalized linear phase FIR filters using the windowing method. Design a lowpass filter of length $N = 25$ with cutoff frequency $\omega_c = \pi/3$ using a Hamming window. First derive the closed-form solution of $g[n]$ and apply the required window, then calculate and plot the impulse response, and the magnitude (in dB) and phase of the frequency response of the designed FIR filter. Find the passband ripple, stopband attenuation, passband edge frequency, and stopband edge frequency.

4 Frequency Sampling Method

As the name implies, this design method is based on sampling the ideal frequency response. Let $G_d(\omega)$ be the ideal frequency response, with linear phase. We then force $H_d(\omega)$ to agree with $G_d(\omega)$ at frequency-sampled points $\omega = \frac{2\pi m}{N}$ for $0 \leq m \leq N-1$. The impulse response is determined by the inverse DFT

$$h[n] = \tfrac{1}{N} \textstyle\sum_{m=0}^{N-1} G_d(\tfrac{2\pi m}{N}) e^{j\frac{2\pi mn}{N}} \quad 0 \leq n \leq N-1.$$

The ideal and frequency-sampled filters are shown in Fig. 8.

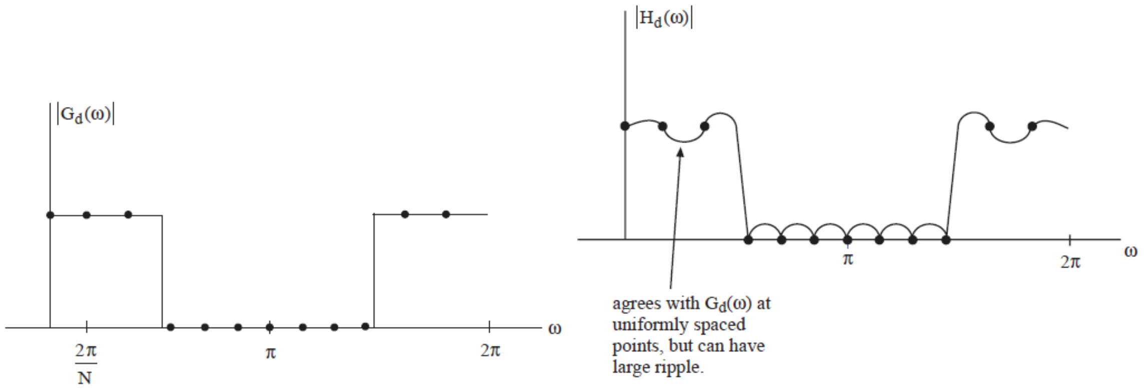

|Gd(ω)|
2π/N
π
2π
ω
|Hd(ω)|
π
2π
ω
agrees with Gd(ω) at uniformly spaced points, but can have large ripple.

Figure 8

Note that the two frequency responses $H_d(\omega)$ and $G_d(\omega)$ are only identical for the sampled values. The implementable frequency response $H_d(\omega)$ can have significant ripples. One way to reduce the ripples is to allow for a variable transition sample in the transition band, whose amplitude can be chosen to utilize tradeoffs between ripple amplitudes and the transition bandwidth. Such a sample is represented with an 'x' in the figure below.

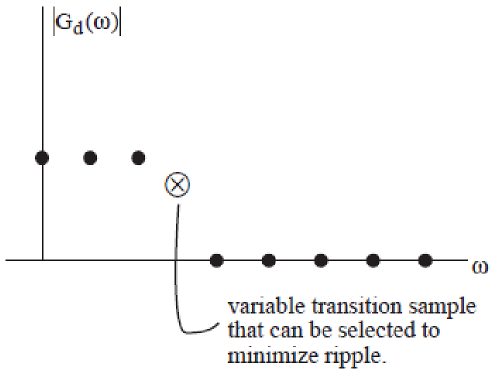

$|G_d(\omega)|$
ω
variable transition sample that can be selected to minimize ripple.

Figure 9

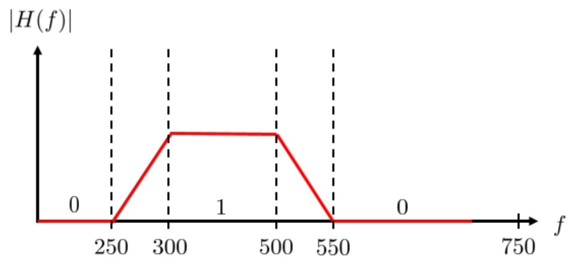
5 Parks-McClellan FIR Filter Design

The Parks-McClellan FIR filter produces a minimum length, linear phase filter with minimal error between the desired response and the actual response using the Remez exchange algorithm. In MATLAB, a PM filter can be designed using **firpmord**. The input to this command are the band edges, desired amplitude, and ripple levels. A bandpass filter can be specified by **firpm**(f, a, rp, fs) where $f = [250, 300, 500, 550]$, $a = [0, 1, 0]$, $rp = [0.001, 0.1, 0.001]$, and $f_s = 1500$. This corresponds to an ideal bandpass filter shown in Figure 11a. The output of **firpmord** are the filter order, frequency band edges, desired amplitudes, and weighting function. These are given to **firpm** which returns the FIR filter coefficients b. The filter returned from **firpm** can be applied to an input signal using **filter**.

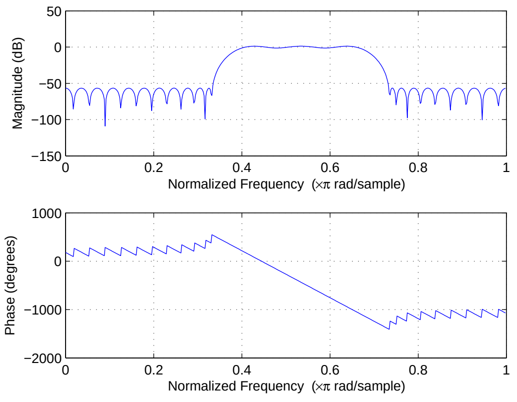
```
clc, clear all, close all

f = [250,300,500,550];
a = [0,1,0];
rp = [0.001 0.1 0.001];
fs = 1500;
[n,fo,mo,w] = firpmord(f, a, rp, fs);
b = firpm(n,fo,mo,w);
freqz(b,1);
```

Figure 10: Using firpm to produce a bandpass filter using **firpm**.



(a) Desired response.



(b) Actual response.

Figure 11: A comparison between the desired response vs the actual response produced by the PM filter method.

Report Item: Design a low-pass filter using the PM filter method with a passband edge frequency of 0.3π and a stopband edge frequency of 0.36π with a stopband attenuation of -50 dB and passband ripple of 2 dB peak-to-peak. Use **freqz** to plot the magnitude and phase response of this filter. Use **impz** to plot the impulse response of this filter.

6 Spectrograms

A spectrogram is constructed from the short-time DFT of a signal. A short-time DFT is essentially identical to an ordinary fourier transform except a sliding window $w(n)$ is applied to the input. A generic expression for the short-time DFT is:

$$s(m,k) = \sum_{n=0}^{M-1} w(n)x(n+mD)e^{-i2\pi kn/M} \tag{6}$$

where M is the approximate support of $w(n)$ and D is a positive integer signifying how much to shift $w(n)$ for a given m. In the case of a rectangular window of length M, the short-time DFT can be written as

$$s(m,k) = \sum_{n=0}^{M-1} x(n+mD)e^{-i2\pi kn/M} \tag{7}$$

The utility of the short-time DFT is that it enables one see both the frequency components of a signal as well as the approximate time in which that frequency occurs. For example, suppose we have the following signal:

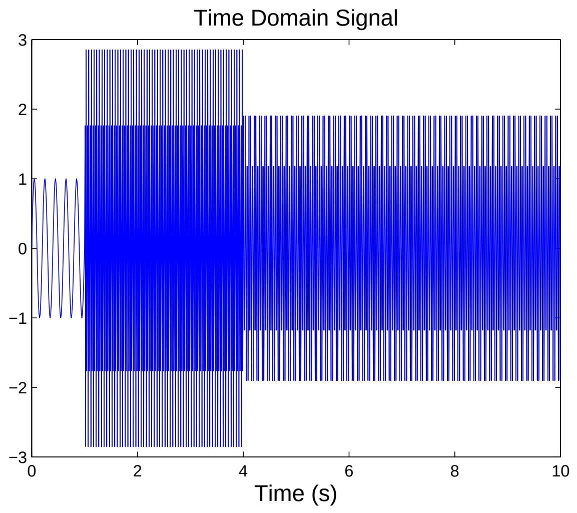


Figure 12

This signal contains 3 frequencies which occur at different times. From the magnitude spectrum in Figure 13, we are able to identify what these frequencies are but we cannot determine at what time they occur. By using short-time DFT, we can determine the approximate time at which each frequency component occurs.

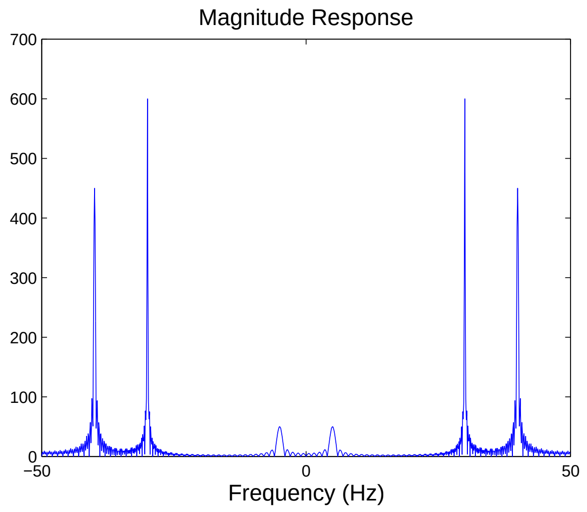


Figure 13

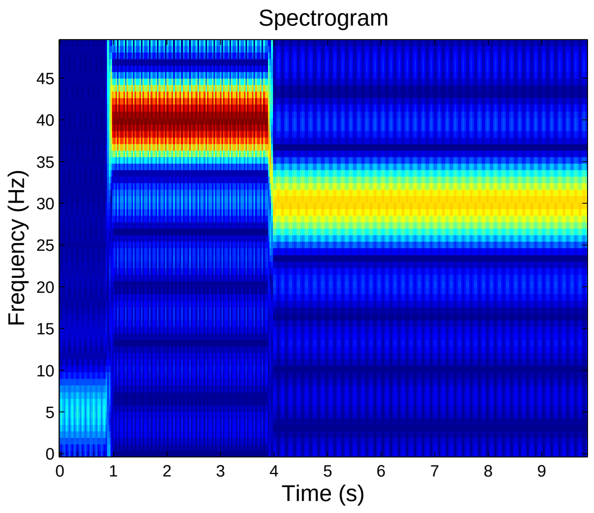


Figure 14: The short-time DFT of the signal with $M = 15$, $D = 3$, and $P = 128$.

The short-time DFT has more knobs to turn compared to the ordinary DFT. For example, when M is chosen to be large, one gains better frequency resolution but loses time resolution. Conversely, when M is small, one gains better time resolution but loses frequency resolution. Additionally, the window length M implies that there is some minimum frequency which can be captured for each time slice. This minimum frequency can be determined using the *Rayleigh Limit*. A high-pass filter can be applied to the overall signal to eliminate frequencies below the minimum frequency seen by the window.

Report Item: Write a function called *mySTDFT* that accepts an input signal x, rectangular window length M, shift D, zero-padding length P, and sampling frequency f_s. The 3 outputs to this function are: (1) a $\lceil P/2 \rceil \times \left(\lfloor \frac{N-M}{D} \rfloor + 1\right)$ matrix where each column corresponds to the M-point DFT padded to P for a particular time-slice, (2) a vector which stores Ω, and (3) a vector of time shifts mD/f_s. *Do not fftshift the output of the fft.* Instead, take just the first $\lceil P/2 \rceil$ points of the DFT corresponding to the positive frequency components. Additionaly, Ω should also correspond to just the positive frequency components. Apply your spectrogram function to the signal in *spectrogram.mat* for $P = 128$, $M = 5$, and $D = 5$. Plot the resulting spectrogram using **imagesc**. What frequencies are present and at what time do they occur? Change M to 30 and use **imagesc** to plot the spectrogram. What differences do you notice? How is this related to the *Rayleigh limit*?

7 Fourier De-noising

In this section, we apply our knowledge about filters to remove noise from a signal. But first of all, what is noise? Well in the case of audio, it can be considered sound which our ears find unpleasant. Mathematically there is little difference between signal and noise and sometimes it can be hard to separate the two from each other. A particular kind of noise known as *white noise* is broadband in frequency. That is, if you take the DFT of white noise, the frequency spectrum has similar amplitude for all frequencies and random phase.

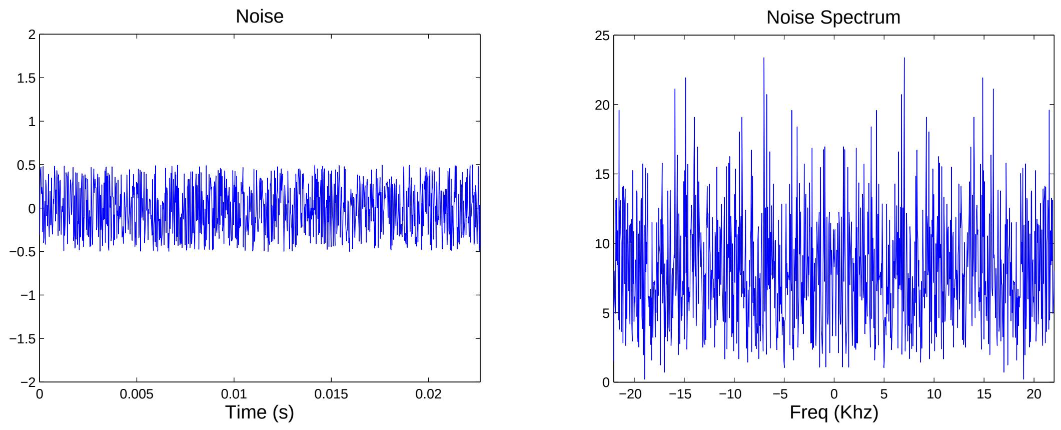


Figure 15: White noise.

Report Item: Load *sound1.wav* using **wavread**. How many seconds is the total sound file? Plot the magnitude spectrum along with the spectrogram of the signal. Write the parameters you choose for the spectrogram. Use **firpm** to create a filter that will remove the noise observed in the frequency response. Plot the magnitude spectrum and spectrogram of the filtered signal using the same parameters as before. Additionally, use **soundsc** to listen to the filtered signal. Comment on the effectiveness of the filter.

Using filter techniques, it is possible to remove noise within certain frequency bands. However, there are limitations to this technique. If the noise is truly broadband, then noise components can overlap with the frequency components of the actual signal. It is difficult to remove this kind of noise using Fourier de-noising techniques. An alternative method for de-noising is wavelets.

Report Item: Load *sound2.wav* using **wavread**. Plot the magnitude spectrum along with the spectrogram of the signal. Write the parameters you choose for the spectrogram. From the magnitude spectrum, identify the frequency band where white noise is present. Then, examine the spectrogram. Can you find more sources of noise in the spectrogram? Why does the magnitude spectrum fail to capture this? Use **firpm** to create a filter to remove the noise observed from the magnitude response. Plot the magnitude spectrum and spectrogram of the filtered signal using the same parameters as before. Additionally, use **soundsc** to listen to the filtered signal. Comment on the effectiveness of the filter.

8 Windowed Spectrogram

In the real world, signals are often only periodic for a finite length of time. Using music as an example, every note played by an instrument has a dominant frequency. When doing analysis on these signals, we are only interested in signals that are periodic in this short time frame, and we wish to vizualize how these frequencies change over time. To visualize the variation of frequencies in short time periods a spectrogram is often used. Figure 16 shows such an example. This is the spectrogram of the speech signal from Lab 1.

As seen in the previous exercises, if the frequencies of interest in the time window that we choose do not 'work well' with the number of samples, we will be seeing lots of spectral leakage. Another way to deal with spectral leakage is to apply a windowing function on the signal before taking its Fourier Transform. One way to think of this is giving an extra 'weight' to signals near the center of the time period.

In order to compute the spectrogram, we need to define a few parameters. The FFT length, N, step size, m, and windowing function w. Each column (denoted by X_i) of the spectrogram is computed as follows:

$$X_i = abs(fft(w \circ x_i)) \tag{8}$$

Where x_i is the i-th window into the signal. (`x_i = x(((i-1)*m+1):(i-1)*m+N)`, in MATLAB) and $\circ$ is the element wise multiplication operator.

You can use a for-loop to compute each column of the spectrogram. The expected result, X will be a $((N/2)+1) \times l$ matrix (see Note 1), where l is the total number of windows. This image can be displayed using `imagesc` or other image displaying functions provided by MATLAB. You should choose the appropriate labels for the axes.

Note 1: Since the FFT is amplitude spectrum symmetrical about π, in practice, we only need the first half of the amplitude spectrum. i.e. For length $N = 128$, we only need the first $N/2 + 1$ values.

Note 2: Step sizes are chosen depending on how much you want each window to overlap. With step size the same as the FFT length, you will have non-overlapping windows.

Note 3: You actual colors might be different depending on the color map you have selected. Different colors map to different magnitudes, choose one that helps you see the differences in the magnitude bests.

Report Item: Load the speech signal data from Lab 1, 'speechsig.mat'. Using a step size of 2 and FFT length of 128, (a) Apply the Hamming window to the signal and plot it's spectrogram. (b) Apply the Rectangular window to the signal and plot it's spectrogram. (c) Write a short comment on the differences between the spectrograms. Your plots should look similar to Figure 16

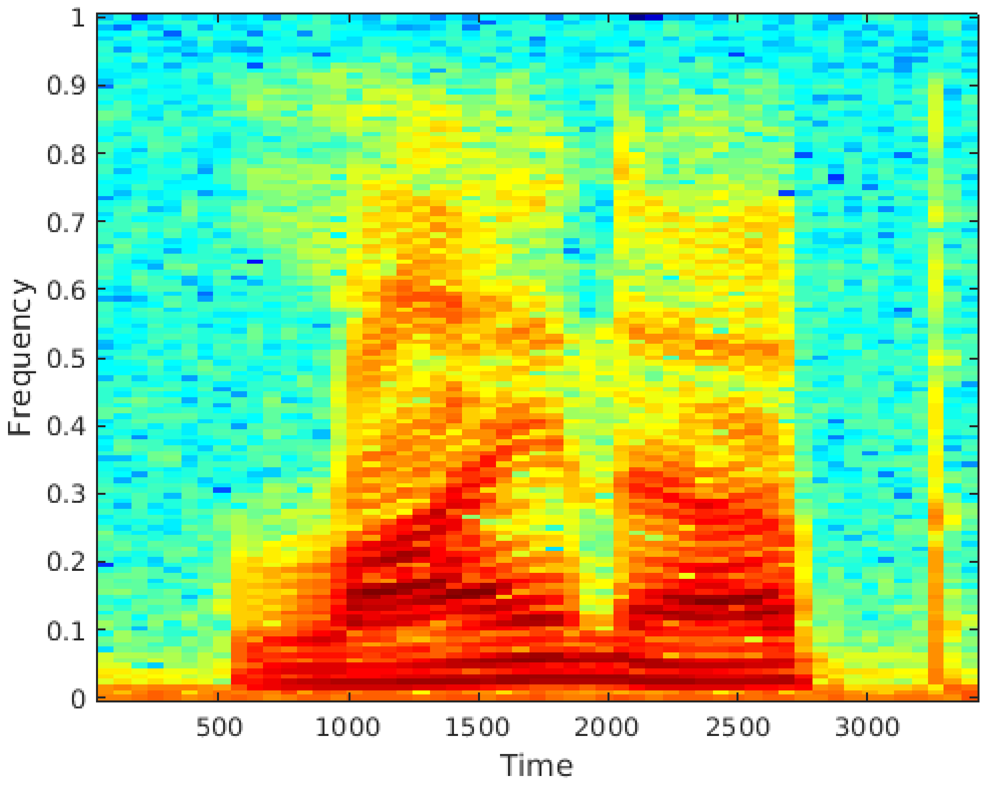


Figure 16: Spectrogram of Speech Signal