EE213 Assignment 3: Digital Filtering

In this assignment, you will:

- Use MATLAB to load and play a short audio signal.
- Compute and plot the Discrete Time Fourier Transform (DTFT) of the signal to identify “noise”.
- Inspect the magnitude response of a Finite Impulse Response (FIR) filter, and apply that filter to remove the “noise”.
- Compute and plot the DTFT of the filtered signal to see the effect of the filtering.

The prerequisite theory for completing this assignment can be found in **Lecture 6: Frequency Analysis**, **Lecture 10: Digital Filters**, and **Lecture 11: Noise and Distortion**.

You should upload **a single PDF file** to Moodle. This PDF file should include your MATLAB code, figures, and answers to all the tasks and the questions therein. Make an effort to communicate your answers clearly, concisely, and with precision. Ensure that all figures have appropriate titles, labels, and units. Marks will be deducted if these components are absent.

Getting Started

Additional Files

Before you begin, there are a few files you should download from Moodle. Save them all to your working directory for the assignment (i.e. make sure everything is in the same folder so you can just use filenames in your code and won't have to specify paths to files). The files are:

- DTFT.m
- noisysig1.m
- noisysig2.m

Loading Data into MATLAB

As you might expect, MATLAB has functions for loading data from files. MATLAB's own file format for data is .mat. We can use the load command to load a file into the workspace.

Playing Audio Files in MATLAB

We can also use MATLAB to play audio files using the sound command. A digital audio file is a one- dimensional discrete signal, which is just a sequence of numbers.

```
load('noisysig1')         % Loads noisysig1.mat into the MATLAB workspace.
sound(noisysig1)          % Plays noisysig1 as an audio file.
```

If you run this code, you should hear a voice saying “Please get rid of this beep!”, with a beep overlaid on the signal (the beep is actually a sine wave).

Using the DTFT Function

The DTFT.m file you downloaded from Moodle is an implementation of the DTFT as a function.

- The syntax you should use to call the function is [X,w]=DTFT(x,M)
- ‘x’ is the input signal.
- ‘M’ specifies the number of output points in the DTFT.
- ‘w’ is a vector of frequencies from $-\pi$ to π generated by the function (for plotting the DTFT).
- ‘x’ is a vector of the DTFT magnitudes (i.e. the transformed signal).

```
help DTFT
```

```
This function computes samples of the DTFT of x.
To compute the DTFT of x, use

            [X,w] = DTFT(x,0)

where X is the vector of DTFT samples and w is the
vector of radial frequencies. To compute at least
M samples of the DTFT, you may use the command

            [X,w] = DTFT(x,M)

This is useful when the plot of X versus w does
not contain a sufficient number of points.
```

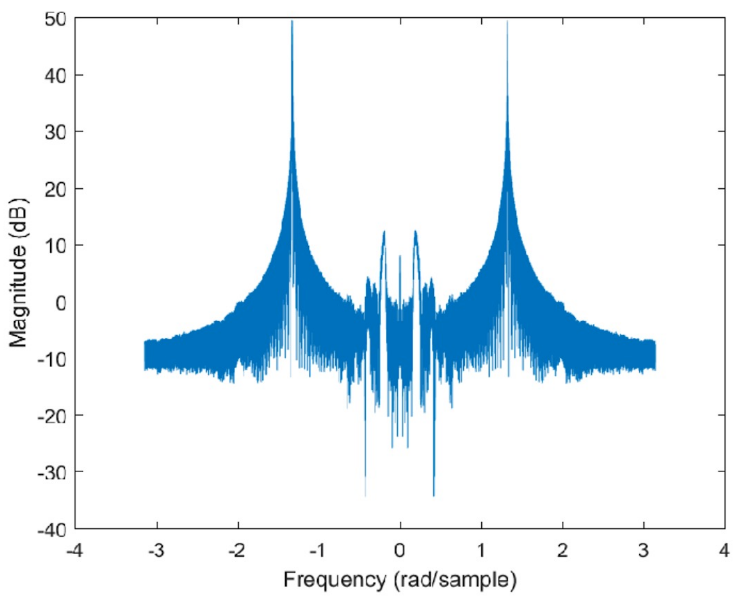
Note: If the length of the input signal is small, you may increase M to achieve a smoother plot in the frequency domain.

Task 1: DTFT of Signal 1

1. Compute the magnitude of the DTFT of 1001 samples of the audio signal noisysig1 for the time indices (100:1100). Set the number of samples in the frequency domain to ‘M=5000’.
2. Plot the magnitude of the DTFT (in decibels) versus frequency for $\omega \in [-\pi, \pi]$.
3. You should arrive at the figure shown on the top of the next page.

The two large peaks in the spectrum correspond to the sinusoidal interference signal (the beep).

Hint: Recall that the decibel conversion is given by $|H_{\text{dB}}(\omega)| = 20\log_{10}(|H(\omega)|)$. MATLAB has built-in functions for computing logarithms run ‘help log’ for more information.



```
% Starter code for Task 1
load('noisysig1');

x = noisysig1(100:1100);
M = 5000;
[X,w] = DTFT(x,M);
```

Effort Required: This task can be completed by adding 1 line to the starter code given above (not including axis labels etc.).

Task 2: Simple FIR Filter Design

- We would like to tune a simple filter to remove the interference signal.
- In order to do this, we must first determine the frequency of the interference signal.
- Because the interference signal has more power than the underlying speech signal, the locations of the peaks in the spectrum indicate the frequency of the interference signal. We can use MATLAB's max function to determine this frequency.

Using the max function

- The following code snippet indicates how the max function works.
- The max function returns two values: the maximum value; and the location (or index) of that value.
- **Note:** If the maximum value is not unique, then the function returns the index of the first instance of the maximum value. If you run this code snippet a few times, you will observe this behavior.

```
% Generate some random integers.
nums = randi(10,1,10)
```

```
nums = 1x10 double

     7     1     9    10     7     8     8     4     7     2
```

```
% Find the maximum value, and its location in the vector.
[max_val, max_ind] = max(nums)
```

```
max_val = 10
max_ind = 4
```

Finding the interference signal frequency

- We apply this to the spectrum calculated earlier and convert the returned index to a frequency.
- This should return the frequency corresponding to the peak on the left.

```
% Compute max
[Xmax, ind_Xmax] = max(X);

% Convert max index to a frequency
wc = (1-ind_Xmax/(length(w)/2))*(-pi)
```

```
wc = -1.3223
```

Filter specification

To remove an interference signal at a frequency ω_c, we can use the following FIR filter, which has 3 coefficients.

$$h = [1, -2\cos(\omega_c), 1]$$

Note: Recall that $\cos(-x) = \cos(x)$, so it does not matter that we are using the peak from the negative half of the frequency domain.

Implementation

1. Compute the filter coefficients and plot them using the stem function.
2. Compute and plot the magnitude of the filter's frequency response.
3. Use the filter to filter the noisy signal. Recall that the process of filtering is an application of convolution. Thus, to apply the filter, use MATLAB's ‘conv’ command, i.e. ‘y = conv(h, noisysig1)’.
4. Play the filtered signal using the sound command. Comment on how the filtering changes the quality of the audio signal.
5. Plot the magnitude (in dB) of the filtered signal. Comment on how the filtering changes the frequency content of the signal.

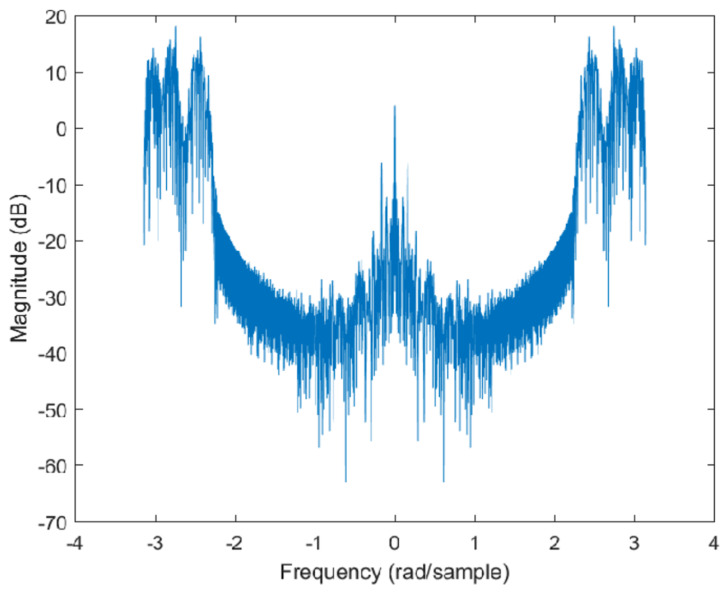
Effort required: This task can be completed using 8 lines of code (not including axis labels, etc.), plus your comments.

Task 3: DTFT of Signal 2

Inspecting the noisy signal

In the preceding two tasks, you removed noise at a particular frequency from a signal. In this task, you will remove noise at a range (a.k.a. a window) of high frequencies.

1. Load ‘noisysig2’.
2. Play the signal using the sound command.
3. Comment on the quality of the speech signal and the backgroud noise.
4. Compute the magnitue of the DTFT of 1001 samples of the audio signal noisysig2 for the time indices (100:1100). Set the number of samples in the frequency domain to ‘M=5000’.
5. You should arrive at the figure shown below.



Effort Required: This task can be completed in 6 lines of code (not including axis labels etc.), plus your comments.

Task 4: FIR Filter Design using Window Method

Filter Design

From the spectrum you just calculated, it should be clear that the noise content of the signal occupies a range of frequencies. As such, a filter which removes a single frequency (like the one in Task 2) is unsuitable for this application. Instead, we must use the **Window Method**. This is discussed in detail in Section 4.1 of Lecture 10.

- We would like to eliminate what we consider to be the noise in this signal i.e. frequencies above 2rad/ sample.
- As such, we should use a low-pass filter, which should have a cut-off

frequency $\omega_c = 2$.

- In this assignment, we use a Rectangular window in the design of the filter.
- Let N be the number of coefficients in the designed FIR filter. (In Task 2, $N = 3$).

The coefficients of an FIR filter with a cutoff frequency ω_c, using a rectangular window, are given by

$$h[n] = \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c(n - n_0)}{\pi}\right), \qquad n = 0, \ldots, N - 1$$

where $n_0 = (N - 1)/2$. The detailed derivation of this filter design can be found in Lecture 10.

Implementation

Given this filter design, complete the following tasks for $N = 21$ and for $N = 101$:

1. Plot the filter coefficients using the stem function.
2. Compute and plot the magnitude of the DTFT of the filter.
3. Use ‘conv’ as before to filter the audio signal ‘noisysig2’, and play the filtered signal using the ‘sound’ command.
4. Comment on difference in the quality of the filtered signal.
5. Compute and plot the magnitude of the DTFT of the filtered signal.

In your comments, you should (qualitatively) compare the performance of the filter for both values of N.

Effort Required: This task can be completed in approximately 15 lines of code (for one value of N, not including axis labels etc.), plus your comments.