Lab 6 Information

- The material in this section is informational. Please read through the section as it helps you work on the lab exercises in the next section. There may be code examples in this informational section. You are welcome to copy-and-paste them to MATLAB to run the code, but no submission is needed on any test run.

Calculating DTFT in MATLAB

It is not hard to see that you may use the function `freq_resp` you wrote in Lab 5 to calculate the DTFT of any finite-length signal (why?). Hereafter in this lab, rename the function as below:

```
X = dtft(x,w)
```

where `x` is the input signal vector, `w` is a (normalized radian) frequency vector, and `X` is the DTFT vector of `x` calculated at the frequencies specified in `w`. For an infinite-length signal, we have to truncate it in order to use the function `dtft` to approximately calculate the DTFT of the signal.

Note that the method used in `dtft` is a computationally inefficient way to calculate the DTFT of a signal, as you will see in the latter parts of the lab. We will talk about how to do DTFT calculations in a much more computationally efficient way later in the semester.

Table of Discrete-Time Fourier Transform Pairs

$$\text{Discrete-Time Fourier Transform} \quad : \quad X(e^{j\widehat{\omega}}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\widehat{\omega}n}$$

$$\text{Inverse Discrete-Time Fourier Transform} \quad : \quad x[n] = \frac{1}{2\pi}\int_{2\pi} X(e^{j\widehat{\omega}})e^{j\widehat{\omega}n}\, d\widehat{\omega}$$

$$\text{Dirac delta function that repeat every } 2\pi \quad : \quad \delta_{2\pi}(\widehat{\omega}) = \sum_{k=-\infty}^{\infty} \delta(\widehat{\omega} - 2\pi k)$$

$x[n]$	$X(e^{j\widehat{\omega}})$	condition
$a^n u[n]$	$\dfrac{1}{1-ae^{-j\widehat{\omega}}}$	$\lvert a\rvert < 1$
$(n+1)a^n u[n]$	$\dfrac{1}{(1-ae^{-j\widehat{\omega}})^2}$	$\lvert a\rvert < 1$
$\dfrac{(n+r-1)!}{n!(r-1)!}a^n u[n]$	$\dfrac{1}{(1-ae^{-j\widehat{\omega}})^r}$	$\lvert a\rvert < 1$
$\delta[n]$	1	
$\delta[n-n_0]$	$e^{-j\widehat{\omega}n_0}$	
$x[n] = 1$	$2\pi\delta_{2\pi}(\widehat{\omega})$	
$u[n]$	$\dfrac{1}{1-e^{-j\widehat{\omega}}} + \pi\delta_{2\pi}(\widehat{\omega})$	
$e^{j\widehat{\omega}_0 n}$	$2\pi\delta_{2\pi}(\widehat{\omega} - \widehat{\omega}_0)$	
$\cos(\widehat{\omega}_0 n)$	$\pi\left[\delta_{2\pi}(\widehat{\omega} - \widehat{\omega}_0) + \delta_{2\pi}(\widehat{\omega} + \widehat{\omega}_0)\right]$	
$\sin(\widehat{\omega}_0 n)$	$\dfrac{\pi}{j}\left[\delta_{2\pi}(\widehat{\omega} - \widehat{\omega}_0) + \delta_{2\pi}(\widehat{\omega} + \widehat{\omega}_0)\right]$	
$\displaystyle\sum_{k=-\infty}^{\infty}\delta[n-kN]$	$\dfrac{2\pi}{N}\displaystyle\sum_{k=-\infty}^{\infty}\delta\left(\widehat{\omega} - \dfrac{2\pi k}{N}\right)$	
$x[n] = \begin{cases} 1 & , \quad \lvert n\rvert \le N \\ 0 & , \quad \lvert n\rvert > N \end{cases}$	$\dfrac{\sin(\widehat{\omega}(N+1/2))}{\sin(\widehat{\omega}/2)}$	
$\dfrac{\sin(Wn)}{\pi n} = \dfrac{W}{\pi}\text{sinc}\left(\dfrac{Wn}{\pi}\right)$	$X(e^{j\widehat{\omega}}) = \begin{cases} 1 & , \quad 0 \le \lvert\widehat{\omega}\rvert \le W \\ 0 & , \quad W < \lvert\widehat{\omega}\rvert \le \pi \end{cases}$	
	$X(e^{j\widehat{\omega}})$ is periodic with period 2π	

Table of Discrete-Time Fourier Transform Properties

For each property, assume

$$x[n] \overset{DTFT}{\longleftrightarrow} X(e^{j\widehat{\omega}}) \quad \text{and} \quad y[n] \overset{DTFT}{\longleftrightarrow} Y(e^{j\widehat{\omega}})$$

Property	Time domain	DTFT domain
Linearity	$Ax[n] + By[n]$	$AX(e^{j\widehat{\omega}}) + BY(e^{j\widehat{\omega}})$
Time Shifting	$x[n - n_0]$	$X(e^{j\widehat{\omega}})e^{-j\widehat{\omega}n_0}$
Frequency Shifting	$x[n]e^{j\widehat{\omega}_0 n}$	$X(\widehat{\omega} - \widehat{\omega}_0)$
Conjugation	$x^*[n]$	$X^*(e^{-j\widehat{\omega}})$
Time Reversal	$x[-n]$	$X(e^{-j\widehat{\omega}})$
Convolution	$x[n] * y[n]$	$X(e^{j\widehat{\omega}})Y(e^{j\widehat{\omega}})$
Multiplication	$x[n]y[n]$	$\frac{1}{2\pi}\int_{2\pi} X(e^{j\theta})Y(e^{j(\widehat{\omega}-\theta)})d\theta$
Differencing in Time	$x[n] - x[n-1]$	$(1 - e^{-j\widehat{\omega}})X(e^{j\widehat{\omega}})$
Accumulation	$\sum_{k=-\infty}^{\infty} x[k]$	$\frac{1}{1-e^{-j\widehat{\omega}}} + \pi X(0)\sum_{k=-\infty}^{\infty}\delta(\widehat{\omega} - 2\pi k)$
Frequency Differentiation	$nx[n]$	$j\frac{dX(e^{j\widehat{\omega}})}{d\widehat{\omega}}$
Parseval's Relation for Aperiodic Signals	$\sum_{k=-\infty}^{\infty} \lvert x[k]\rvert^2$	$\frac{1}{2\pi}\int_{2\pi} \lvert X(e^{j\widehat{\omega}})\rvert^2 d\widehat{\omega}$

Lab 6 Exercises

- Unless stated otherwise, you must submit your solutions to all the lab exercises in this section.
- Your laboratory solutions should be submitted on Canvas as a single PDF. The simplest way is to put your codes and answers for all the lab exercises in a single MATLAB Publisher script and use `%%` to separate the codes and answers for different exercises into different sections as described in the information section of Lab 1.
- **Plot all magnitude spectrums in this lab in dB scale.**

Exercise 6.1:

This exercise shows you how to carefully interprete the results obtained from your MATLAB function `dtft`, particularly for the cases of infinite-length signals.

Use the tables in the information section, or otherwise, to analytically determine the expressions of the DTFTs for the signals below. Use your MATLAB function `dtft` to calculate and plot (magnitude and angle) the DTFTs. For each signal, truncate its length to 10, if needed, when using your function `dtft` to calculate the DTFT. Compare your plot with the DTFT expression you obtain analytically. Do they match? Repeat the same MATLAB calculation of the DTFT and comparison with the analytic expression by truncating each signal to length 100 instead. Explain the effects of increasing the truncation length.

(a) $x[n] = \delta[n-3]$

(b) $x[n] = (8/9)^n u[n]$

(c) $x[n] = (-8/9)^n u[n]$

(d) $x[n] = u[n] - u[n-5]$

(e) $x[n] = \cos((\pi/4)n)$

Exercise 6.2: *(Nulling Filter)*

This exercise shows you how to employ DTFT to analyze an audio signal corrupted by a sinusoidal interfering signal. Specifically, you will examine the DTFT to determine the frequency of the sinusoidal interference and then use a nulling filter to remove the interference. The corrupted audio signal is provided in the WAV file `corrupted_wannabe.wav`.

(a) Load `corrupted_wannabe.wav` into MATLAB. Use your function `dtft` to calculate the DTFT of the audio signal. Plot the magnitude of the DTFT. Hint: You should use a frequency vector that has at least 2000 points to calculate the DTFT so as to properly identify the interference frequency in part (b) below. It may take a minute or so to compute the DTFT.

(b) Identify the frequency of the sinusoidal interference. Give this frequency in normalized radian frequency as well as continuous-time cyclic frequency. Hint: You may use "Tools→Data Tips" to identify the value of interference frequency from the DTFT magnitude plot. You may also use the MATLAB function `find` (see Lab 5) to more accurate determine the frequency of interest.

(c) Consider the following nulling FIR filter:

$$y[n] = x[n] - 2\cos(\widehat{w}_0)x[n-1] + x[n-2], \tag{1}$$

This filter can remove a sinusoid at the normalized radian frequency specified by $\widehat{w}_0$. Design this filter to remove the interference that you identified in (b), i.e. specify the impulse response of the resulting filter in the vector `h`. Use your function `dtft` to calculate and plot the magnitude response and phase response of the filter.

(d) Apply the nulling filter to the corrupted audio signal. Listen to the filtered audio and save it to the WAV file `filtered_wannabe.wav`. Submit the WAV file. Use your `dtft` function to calculate and plot the magnitude of the DTFT of the filtered audio signal. Describe the differences between the DTFTs of the original audio and the output of the nulling filter, particularly with respect to the interference signal.

Exercise 6.3 (Extra credits: +20 points):

This exercise is similar to Exercise 6.2. An audio signal (not the same as the one in Exercise 6.2) is corrupted by a very strong artificial noise (not a simple sinusoidal interference). You again need to use your `dtft` function to study the frequency contents of the corrupted audio signal, identify the frequency range of the noise components, and design a filter to remove the noise. The theory behind how to design a filter that can remove the noise here is a bit too advance for us. Instead of going through the theory, this exercise shows you how to use a MATLAB filter design GUI to work out the required filter.

Load in the noisy audio signal from the WAV file `what_did_he_say.wav`. Use your `dtft` function to calculate and plot the magnitude of the signal's DTFT. Identify the frequency range of the noise. Express your answer in both normalized radian frequency and continuous-time cyclic frequency. To remove the noise in this case, you will need to employ a filter different than the nulling filter in Exercise 6.2. State what kind of filter (lowpass, highpass, bandpass) is needed to remove the noise. Determine the start and end frequencies of the passband and stopband of the filter.

MATLAB has a GUI `filterDesigner` that helps you design a filter. Type `filterDesigner` in the command window to invoke the GUI. You may select the type of the filter that you want to design. Based on your answer above, choose whether you want to design a lowpass, highpass, or bandpass filter. Choose **FIR Equiripple**. Choose the **Filter Order** to be **Minimum order**. Specify the sampling frequency, start and end frequencies of the passband and stopband. You will also need to specify the ripple tolerance in the passband (**Apass**) and the magnitude attenuation from the passband to stopband (**Astop**). It suffices to choose **Apass** = 1 dB. You will need to choose a suitable value for **Astop**. For example, **Astop** = 60 dB means that the value of the magnitude response in the stopband is 60 dB below that in the passband. After specifying all the parameters, click the **Design Filter** button to generate the filter impulse response. You can save the generated impulse response to your workspace by choosing "File→Export", and then specifying **Export to Workspace**, **Export as Coefficients**, and setting `Numerator` to your choice of a variable name to hold the impulse response. Click the **Export** button to export the impulse response to your workspace.

Use the GUI to design a filter that can remove the noise. **Report all the parameters that you specify in the GUI.** Export the impulse response of the filter to your MATLAB workspace. Apply this filter to the noisy audio signal. Listen to the filtered signal and save the filtered audio to the WAV file `filtered_what.wav`. Submit the WAV file. **Write down the sentence dictated in the audio file.** Use your `dtft` function to calculate and plot the magnitude of the DTFT of the filtered audio signal. Describe the differences between the DTFTs of the original audio and the output of the noise removal filter, highlighting what and how much noise components are removed by the filter.

Note that you may need to iterate the above filter design process a few times to generate a relatively clean and audible audio signal.