Lab P-11: Frequency Response: Hearing Test

Pre-Lab: Read the Pre-Lab and do all the exercises in the Pre-Lab section ***prior to attending lab.***

Verification: The Exercise section of each lab should be completed **during your assigned Lab time** and the steps marked *Instructor Verification* signed off **during the lab time**. When you have completed a step that requires verification, demonstrate the result to your instructor and answer any questions about it. Turn in the completed verification sheet before you leave the lab.

Lab Report: The project requires a MATLAB programming effort and should be documented with a lab report. A good report should include the following items: a cover sheet, commented MATLAB code, explanations of your approach, and conclusions.

1 Introduction

The goal of this lab is to study the frequency response. For FIR filters this is the response to inputs such as complex exponentials and sinusoids. You can use `firfilt()`, or `conv()`, to implement filters and `freqz()` to obtain the filter's frequency response.[1] As a result, you should learn how to characterize a filter by knowing how it reacts to different frequency components in the input.

2 Pre-Lab

2.1 Frequency Response of FIR Filters

The output or *response* of a filter for a complex sinusoid input, $e^{j\hat{\omega}n}$, is a complex exponential at the same frequency. The magnitude and phase of the output will be different, and that change depends on the frequency, $\hat{\omega}$. The dependence of these magnitude and phase changes versus frequency is called the *frequency response*. In effect, the filter is described by how it affects different input frequencies. For example, the frequency response of the two-point averaging filter

$$y[n] = \tfrac{1}{2}x[n] + \tfrac{1}{2}x[n-1]$$

can be found by using a general complex exponential as an input $x[n]$ and observing the output or *response.*

$$x[n] = Ae^{j(\hat{\omega}n+\varphi)} \tag{1a}$$

$$y[n] = \tfrac{1}{2}Ae^{j(\hat{\omega}n+\varphi)} + \tfrac{1}{2}Ae^{j(\hat{\omega}(n-1)+\varphi)} \tag{1b}$$

$$= Ae^{j(\hat{\omega}n+\varphi)} \underbrace{\left(\tfrac{1}{2} + \tfrac{1}{2}e^{-j\hat{\omega}}\right)}_{\mathcal{H}(\hat{\omega}),\ \text{function of } \hat{\omega}} \tag{1c}$$

In (1c) there are two terms, the original input, and a term that is a function only of $\hat{\omega}$. This second term is the frequency response and it is commonly denoted[2] by $H(e^{j\hat{\omega}})$.

$$H(e^{j\hat{\omega}}) = \mathcal{H}(\hat{\omega}) = \tfrac{1}{2}\left(1 + e^{-j\hat{\omega}}\right) \tag{2}$$

[1] If you do not have the function `freqz.m`, there is a substitute available called `freekz.m` in the *DSP-First* (or *SP-First*) Toolbox.

[2] The notation $H(e^{j\hat{\omega}})$ is used in place of $\mathcal{H}(\hat{\omega})$ for the frequency response because we will eventually connect this notation with the z-transform, $H(z)$, in later chapters.


The general form of the frequency response for an M-th order FIR linear time-invariant system with filter coefficients $\{b_k\}$ is

$$H(e^{j\hat{\omega}}) = \sum_{k=0}^{M} b_k e^{-j\hat{\omega}k} \tag{3}$$

Once the frequency response, $H(e^{j\hat{\omega}})$, has been determined, the effect of the filter on any complex exponential input may be determined by evaluating $H(e^{j\hat{\omega}})$ at the corresponding input frequency. The output signal, $y[n]$, is also a complex exponential whose complex amplitude has a constant magnitude and phase. The phase of $H(e^{j\hat{\omega}})$ describes the phase change of the complex sinusoid and the magnitude of $H(e^{j\hat{\omega}})$ describes the "gain" applied to the complex sinusoid.

2.1.1 MATLAB Function for Frequency Response

MATLAB has a built-in function for computing the frequency response of a discrete-time LTI system. The following MATLAB statements show how to use `freqz` to compute and plot both the magnitude (absolute value) and the phase of the frequency response of a two-point averaging system as a function of $\hat{\omega}$ in the range $-\pi \leq \hat{\omega} \leq \pi$:

```
bb = [0.5, 0.5];        %-- Filter Coefficients
ww = -pi:(pi/100):pi;   %-- omega hat frequency vector
H = freqz(bb, 1, ww);   %<--freekz(bb,1,ww) is an alternative
subplot(2,1,1);
plot(ww, abs(H)), grid on
subplot(2,1,2);
plot(ww, angle(H)), grid on
xlabel('Normalized Radian Frequency')
```

For FIR filters, the second argument of `freqz( _, 1, _ )` must always be equal to `1`. The frequency vector `ww` should cover an interval of length 2π for $\hat{\omega}$, and its spacing must be fine enough to give a smooth curve for $H(e^{j\hat{\omega}})$. Note: we always use capital `H` for the frequency response.[3]

2.2 Periodicity of the Frequency Response

The frequency responses of discrete-time filters are *always* periodic with period equal to 2π. Explain why this is the case by using the definition of the frequency response (3) and then considering two input sinusoids whose frequencies are $\hat{\omega}$ and $\hat{\omega} + 2\pi$.

$$x_1[n] = e^{j\hat{\omega}n} \qquad \text{versus} \qquad x_2[n] = e^{j(\hat{\omega}+2\pi)n}$$

It should be easy to prove that $x_2[n]$ is equal to $x_1[n]$. Consult Chapter 6 for a mathematical proof that the outputs from each of these signals are identical **The implication of periodicity is that a plot of $H(e^{j\hat{\omega}})$ only has to be made over the interval $-\pi \leq \hat{\omega} \leq \pi$.**

2.3 Frequency Response of the Four-Point Averager

In Chapter 6 we examined filters that compute the average of input samples over an interval. These filters are called "running average" filters or "averagers" and they have the following form for the L-point averager:

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[n-k] \tag{4}$$

[3] If the output of the `freqz` function is not assigned, then plots are generated automatically; however, the magnitude is given in decibels which is a logarithmic scale. For linear magnitude plots a separate call to `plot` is necessary.


(a) Use Euler's formula and complex number manipulations to show that the frequency response for the 4-point running average operator is given by:

$$H(e^{j\hat{\omega}}) = \mathcal{H}(\hat{\omega}) = \left(\frac{2\cos(0.5\hat{\omega}) + 2\cos(1.5\hat{\omega})}{4}\right) e^{-j1.5\hat{\omega}} = C(\hat{\omega})e^{j\psi(\hat{\omega})} \tag{5}$$

(b) Implement (5) directly in MATLAB. Use a vector that includes 400 samples covering the interval $[-\pi, \pi)$ for $\hat{\omega}$. Make plots of $C(\hat{\omega})$ and $\psi(\hat{\omega})$ versus $\hat{\omega}$. It is tempting to think that $C(\hat{\omega})$ is the magnitude of the frequency response, but $C(\hat{\omega})$ can go negative, so these are not plots of the magnitude and phase. You would have to use `abs()` and `angle()` to extract the magnitude $|H(e^{j\hat{\omega}})|$ and phase $\angle H(e^{j\hat{\omega}})$ of the frequency response for plotting.

(c) In this part, use `freqz.m` or `freekz.m` in MATLAB to compute $H(e^{j\hat{\omega}})$ numerically (from the filter coefficients) and plot its magnitude and phase versus $\hat{\omega}$. Write the appropriate MATLAB code to plot both the magnitude and phase of $H(e^{j\hat{\omega}})$. Follow the example in Section 2.1.1. The filter coefficient vector for the 4-point averager is defined via:

```
bb = 1/4*ones(1,4);
```

Recall that the function `freqz(bb,1,ww)` evaluates the frequency response for all frequencies in the vector `ww`. It uses the summation in (3), not the formula in (5). The filter coefficients are defined in the assignment to vector `bb`. How do your results compare with part (b)?
Note: the plots should not be identical, but you should be able to explain why they are equivalent by converting the minus sign in the negative values of $C(\hat{\omega})$ to a phase of π radians.

2.4 The MATLAB FIND Function

Often signal processing functions are performed in order to extract information that can be used to make a decision. The decision process inevitably requires logical tests, which might be done with `if-then` constructs in MATLAB. However, MATLAB permits vectorization of such tests, and the `find` function is one way to determine which elements of a vector meet a certain logical criterion. In the following example, `find` extracts all the numbers that "round" to 3:

```
xx = 1.4:0.33:5, jkl = find(round(xx)==3), xx(jkl)
```

The argument of the `find` function can be any logical expression, and `find` returns a list of indices where that logical expression is true, and `xx(jkl)` gives the values that round to 3. See `help` on `relop` for more information.
Now, suppose that you have a frequency response:

```
ww = -pi:(pi/500):pi; HH = freqz( 1/4*ones(1,4), 1, ww );
```

Use the `find` command to determine the indices where `HH` is zero, or very small, e.g., $|H(e^{j\hat{\omega}})| \leq 1 \times 10^{-8}$. Then use those indices to display the list of frequencies where `HH` is "zero." Since there might be round-off error in calculating `HH`, the logical test should be a test for those indices where the magnitude (absolute value in MATLAB) of `HH` is less than 1×10^{-8}. Compare your answer to the frequency response that you plotted for the four-point averager in Section 2.3.

3 Warm-up

The first objective of this warm-up is to use a MATLAB GUI to demonstrate the frequency response. The frequency response demo, `dltidemo`, is part of the *DSP-First* Toolbox.

3.1 LTI Frequency Response Demo

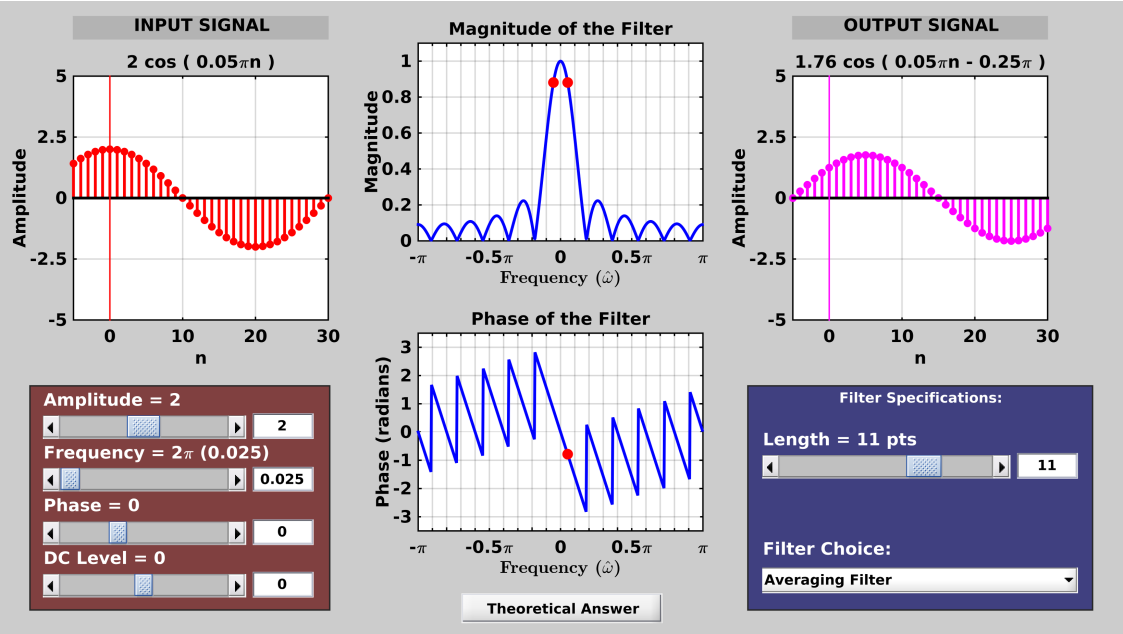


Figure 1: DLTI demo interface showing the output when $x[n] = 2\cos(0.05\pi n)$ is filtered by an length-11 averager. The frequency response (magnitude and phase) of the averager is shown in the middle panels. The frequency response demo, `dltidemo`, is part of the *SP-First* (or *DSP-First*) Toolbox.

The `dltidemo` GUI illustrates the "sinusoid-IN gives sinusoid-OUT" property of LTI systems. In this demo, you can change the amplitude, phase and frequency of an input sinusoid, $x[n]$, and you can change the digital filter that processes the signal. Then the GUI will show the output signal, $y[n]$, which is also a sinusoid (at the same frequency). Figure 1 shows the interface for the `dltidemo` GUI. It is possible to see the formula for the output signal, if you click on the `Theoretical Answer` button located at the bottom-middle part of the window. The digital filter can be changed by choosing different options in the `Filter Specifications` box in the lower right-hand corner.
In the Warm-up, you should perform the following steps with the `dltidemo` GUI:

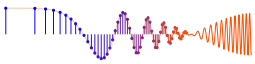
(a) Set the input to $x[n] = 3.6\cos(0.1\pi(n-6))$; note that this sinusoid has a peak at $n = 6$.

(b) Set the digital filter to be a 9-point averager.

(c) Determine the formula for the output signal and write it in the form: $y[n] = A\cos(\hat{\omega}_0(n - n_d))$.

(d) Using n_d from $y[n]$ and the fact that the input signal had a peak at $n = 6$, determine how much the peak of the cosine wave has shifted. This is called the *delay* through the filter. Give an equation that explains how the delay is related to the phase of $H(e^{j\hat{\omega}})$ at $\hat{\omega} = 0.1\pi$.

Instructor Verification (separate page)

(e) Now, change the length of the averaging filter so that the output will be zero, i.e., $y[n] = 0$. Use the GUI to show that you have the correct filter to zero the output. If the filter length is more than 15, you will have to enter the "Filter Specifications" with the `user Input` option.
Hint: Recall the Dirichlet form for the frequency response of the averaging filter, and where it is zero.

(f) When the output is zero, the filter acts as a *Nulling Filter,* because it eliminates the input at $\hat{\omega} = 0.1\pi$. Sinusoidal components at other frequencies $\hat{\omega}$ will also be nulled—make a list of these frequencies.

Instructor Verification (separate page)

3.2 Ideal Filters and Practical Filters

The running average FIR filter is a lowpass filter, but it is not a very good one. Better filters can be designed with computer-aided optimization algorithms. In `dltidemo`, it is possible to choose:

Ideal Filters which are given by their frequency response, but are not actually FIR filters. In other words, there are no filter coefficients that will produce the ideal frequency response(s). In the GUI, you can find ideal lowpass filters (LPF), highpass filters (HPF) and bandpass filters (BPF). The ideal LPFs and HPFs have one parameter for the cutoff frequency which determines the boundary between the passband and stopband. The ideal BPF has a parameter for center frequency which determines where the band is located. All the ideal filters have an additional parameter for the slope of the phase of $H(e^{j\hat{\omega}})$.

Practical Filters which are approximations to the ideal filters by a length-L FIR filter. The ones shown in `dltidemo` were designed using MATLAB's `fir1` function for filter design. The GUI has length-15 LPFs and HPFs, and length-21 BPFs. The LPF and HPF designs have one parameter for the cutoff frequency which determines the boundary between the passband and stopband. The BPF has a parameter for center frequency which determines where the band is located.

Notation: the cutoff frequency for HPFs and LPFs will be called $\hat{\omega}_c$.

(a) Define the input to be $x[n] = 3.6\cos(0.55\pi n)$.

(b) Set the filter to be an ideal HPF with a cutoff frequency of $\hat{\omega}_c = 2\pi(0.2)$. Determine a value for the phase slope so that the output will be $y[n] = 3.6\cos(0.55\pi n - 1.1\pi)$.

(c) Still using the ideal HPF, determine a cutoff frequency ($\hat{\omega}_c$) so that the output will be zero.

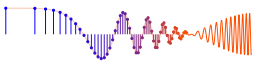
Instructor Verification (separate page)

(d) Now switch to the length-15 HPF, using the same input signal. Determine the output when the cutoff frequency is $2\pi(0.2)$. Is the signal still *passed* by the HPF?

(e) Again with the length-15 HPF, use the cutoff frequency ($\hat{\omega}_c$) from part (c) and determine the formula for the output signal. Why is the output not exactly zero?

(f) Determine a cutoff frequency for the length-15 HPF so that the output signal is nearly zero. Give the formula for the output signal, and compare the cutoff frequency ($\hat{\omega}_c$) to part (c).

Instructor Verification (separate page)

4 Lab Exercises

Unlike most labs written for a first-year or second-year student this lab does not have specific steps with detailed instructions. Instead, there is only one requirement:

> The task for this lab project can be stated simply as: conduct a test of your hearing, and present the results as a frequency response plot.

4.1 Hints and Links

Perhaps the description above is too minimal, so a few hints and links are given below.

1. ***Be kind to your ears: make sure that you test the sound levels without headphones first since your program will be moving the amplitude up and down according to your calculations.***

2. We use MATLAB so you will be writing a short MATLAB program to conduct the test.

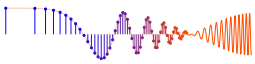
3. "Pure tone hearing threshold measurements have a bias in terms of the amplitude trajectory of the tone presented. If the subject hears the tone and its amplitude is gradually decreased until it is inaudible, a level can be recorded. If instead, the tone starts out sub-threshold and its amplitude is gradually increased until it is audible, a higher level will result than that established earlier. In practice, as a result of this phenomenon, hearing thresholds are measured using ***Békésy audiometry***. When the subject presses a signal button, the amplitude of the tone decreases as long as the button is depressed and increases when it is released. The midpoint of these two levels is identified as the threshold," Prof. Mark Clements.

4. Georg von Békésy won the Nobel Prize in Medicine. The problem is still challenging. Wikipedia attributes the following quote to Békésy, "In time, I came to the conclusion that the dehydrated cats and the application of Fourier analysis to hearing problems became more and more a handicap for research in hearing."

5. MATLAB has some trouble interacting with sound, so the program has to ask the user whether a sound was heard. One way to "ask" is to require a keystroke. Here's some code to do that; the variable names should be self-explanatory.

```
function yesno = hear_sinus(amplitude,frequency,duration,samplingRate)
%
%  yesno = true or false. True when the sound is heard
%
  tt = 0:1/samplingRate:duration;
  xx = amplitude*cos(2*pi*frequency*tt + rand(1)*2*pi);
  sound(xx,samplingRate)
  pause(duration)   %--wait for the sound to end
  aa = input('Can U hear me now? (<CR>=no, y=yes) ','s')
  yesno = length(aa)>0;
  if yesno
    yesno = upper(aa(1))=='Y';
  end
```

6. Consider making a protocol for your testing; here are some considerations but more are possible. Presenting tones at random inhibits guessing by the user being tested. Making multiple measurements at each frequency increases confidence via averaging.

7. You have to use (good) headphones to perform the test, so it will be essential to do some research on the frequency response of your headphones. When you submit your results, include documentation of the type of headphones used. If possible, find information about the frequency response of the headphones.

8. Professional hearing tests are done in a "quiet room." Technical specifications would define quiet.

9. Your plot will be the *threshold of hearing* versus frequency over some range. It is said that humans can hear from 20 Hz to 20,000 Hz, but you might find that range too large. Whatever range you pick, you should make the plot versus $\log(f)$. Use the hearing test to determine the frequency where your hearing sensitivity starts to drop significantly.

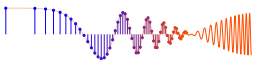
10. Think about how the headphone frequency response would affect your results.

11. A common way to present amplitude is to use a log scale in the form of a *decibel* plot. The definition of decibels is $20\log_{10} A$.

12. Do some research on the term "pure tone audiometry." If you can find a typical plot that an audiologist makes by hand, then you would see a log-log scale.

13. Here are some links (c. 2015) and search phrase, but these might disappear with time:

    - `http://hyperphysics.phy-astr.gsu.edu/hbase/sound/maxsens.html`
    - `http://hyperphysics.phy-astr.gsu.edu/hbase/sound/earsens.html`
    - Search for "pure tone audiogram images"
    - "audiology headphones"
    - "headphone frequency response test"

Lab: Hearing Test via Frequency Response
INSTRUCTOR VERIFICATION PAGE

For each verification, be prepared to explain your answer and respond to other related questions that the instructor might ask. Turn this page in at the end of your lab period.

Name: _____ Date of Lab: _____

Part	Observations
3.1(c)	Formula for output from length-9 averager as $y[n] = A\cos(\hat{\omega}_0(n-n_d))$
3.1(d)	Give an equation that explains how the filter's *delay* is related to the phase of $H(e^{j\hat{\omega}})$ at $\hat{\omega} = 0.1\pi$.

Verified:_____ Date/Time:_____

3.1(e)	Length L of running average FIR filter that nulls $x[n] = 3.6\cos(0.1\pi(n-6))$
3.1(f)	List of all frequencies nulled by the running-average FIR filter in part (e), or give a formula.

Verified:_____ Date/Time:_____

3.2(b)	Phase slope for Ideal HPF to get $y[n] = 3.6\cos(0.55\pi n - 1.1\pi)$
3.2(c)	Cutoff frequency for Ideal HPF to get $y[n] = 0$.

Verified:_____ Date/Time:_____

3.2(d)	Output of length-15 HPF with $\hat{\omega}_c = 0.4\pi$. Give a formula, i.e., $A\cos(\hat{\omega}_0 n + \varphi)$
3.2(e)	Output formula when using cutoff frequency from part (c). Explain why $y[n]$ is not exactly zero.
3.2(f)	Cutoff frequency for length-15 HPF to make output nearly zero. Give formula for $y[n]$.

Verified:_____ Date/Time:_____