Laboratory # 7

Vowel Lab I: Source/Filter Vowel Models

7.1 Introduction

So far, we've been considering filters as systems that we design and then apply to signals to achieve a desired affect. However, filtering is something that occurs everywhere, without the intervention of a human filter designer. At sunset, the light of the sun is filtered by the atmosphere, often yielding a spectacular array of colors. A concert hall filters the sound of an orchestra before it reaches your ear, coloring the sound and adding pleasing effects like reverberation. Even our own head, shoulders, and ears form a pair of filters that allows us to localize sounds in space.

Quite often, we wish to recreate these filtering effects so that we can study them or apply them in different situations. One way to do this is to *model* these "natural" filters using simple digital filters. That is, if we can measure the response of a particular system, we would often like to design a filter that has the same (or a similar) response. In this lab, we will apply such system modeling to the signals produced when speaking vowels.

7.2 Background

7.2.1 The Source-Filter Model for Vowels Sounds

We begin with a physical description of how speech is produced. Traditional models of human speech production generally separate the speech production mechanism into two parts — the *glottal source* and the *vocal tract*. The glottal source is the stream of the air that emerges from the *larynx* (that is, the voice box). During the speaking or singing of a vowel, this stream of air is pushed through the *vocal cords*, which are held together. The resulting pressure causes the vocal cords to open and close rapidly; these oscillations give the glottal source a nearly periodic variation. For speech, this variation has a fundamental frequency of around 100 Hz for males and around 200 Hz for females. If the speaker whispers a vowel, the vocal cords are not held together and thus do not vibrate. Thus, the glottal source is not periodic for whispering; instead, it has a random, noise-like characteristic. In this lab, we will only consider the speaking voice, not singing or whispering.

The second part of the speech production mechanism is the vocal tract. The vocal tract is the airway that leads from the larynx up through the lips. See Figure 7.1 for a diagram of the vocal tract. The positions of the tongue, lips and jaw serve to shape the vocal tract, with different positions creating different vowel sounds. Try speaking a vowel for a couple of seconds and then changing to a different vowel. Notice how your tongue, lips, and jaw

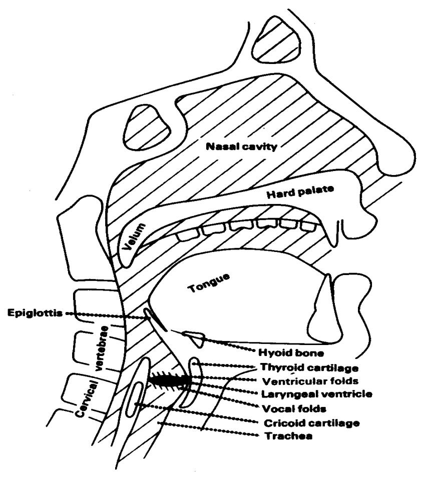


Figure 7.1: The human vocal tract.

change positions. Also try whispering the same vowels. Notice that the changes you make are the same whether you are whispering or speaking.

The two-part mechanism just described applies to vowels sounds, not consonants. For the most part, consonants are produced by using the tongue and lips to make sounds. For instance, the hiss of an "s" sound is made by forcing air past the tongue. A "p" or "t" sound, on the other hand, is produced by quickly releasing air from behind the lips or tongue (respectively). There are a wide variety of consonant sounds, but we will be restricting our attention in this lab to vowels.

We now discuss how vowel speech production is modeled in terms of signals and systems. The goal is a model for the signal $s(t)$ as measured by a microphone when someone speaks a sustained vowel. Following the physical model, the traditional signal/system model is that a glottal source signal $g(t)$ is the input to a vocal tract filter, whose output is $s(t)$.

As illustrated in the upper left panel Figure 7.2 the glottal source signal is modeled as a periodic pulse train

$$g(t) = \sum_{n=-\infty}^{\infty} p(t - nT) \ , \tag{7.1}$$

where T_0 is the period ($f_0 = 1/T + 0$) is the fundamental frequency of $g(t)$) and $p(t)$ is some basic pulse shape, called a *glottal pulse* with duration less than T_0.

When speaking a sustained vowel, it has been found that to a good approximation, the effect of the vocal tract on the glottal signal is well modeled as a linear time-invariant system, i.e. a filter in the usual sense. The irregular shape of the vocal tract causes it to act as a filter that enhances certain frequencies and attenuates others. Modifying the shape of the vocal tract (i.e. the positions of the tongue, lips and jaw) changes the frequency response of the filter, which in turn changes the spectrum of the output signal $s(t)$. Listeners have learned, in effect, to associate each vowel sounds with a distinct spectra. This is similar to the discrimination required to recognize the sounds of different musical instruments.

In this lab we will consider a discrete-time version of the source/filter model just described. Specifically, the sampled output speech signal, $s[n]$, is simply the convolution of the sampled glottal source signal, $g[n]$ with the sampled impulse response of the vocal tract, $v[n]$. That is,

$$s[n] = g[n] * v[n]. \tag{7.2}$$

For the time being, we will assume that $g[n]$ is periodic with period N, where $NT_s = T_0$ with T_s and T_0 being the sampling interval and the fundamental period of $g(t)$, respectively.

Since we assume the vocal tract is linear and time-invariant, it follows that $s[n]$ will also be periodic with period N.

Let us now consider the frequency domain representations of each of these signals. Let $S[k]$ be the DFT of the speech signal, let $G[k]$ be the DFT of the source signal, and let $V(\widehat{\omega})$ be the frequency response of the vocal tract filter. Then, as derived in class,

$$S[k] = G[k]V(\tfrac{2\pi}{N}k) \ , \quad k = 0, \ldots, N-1 \ . \tag{7.3}$$

Note that this holds in the ideal case of $g[n]$ and $s[n]$ periodic with period N. It turns out that the way a vowel sounds is principally determined by the magnitudes of the $S[k]$'s. Thus it is sufficient to consider the magnitudes of $S[k]$, $G[k]$, and $V(\frac{2\pi}{N}k)$, which are related via

$$\big|S[k]\big| = \big|G[k]\big| \ \big|V(\tfrac{2\pi}{N}k)\big| \ , \quad k = 0, \ldots, N-1 \ . \tag{7.4}$$

Typically, we will know $|S[k]|$ and $|G[k]|$ but we want to estimate $|V(\omega)|$. (There will be a different $|S[k]|$ and $|V(\widehat{\omega})|$ corresponding to each type of vowel, but $|G[k]|$ will be essentially the same.) From equation (7.4), we can determine magnitude of the frequency response of the vocal tract filter at the DFT frequencies simply as

$$\big|V(\tfrac{2\pi}{N}k)\big| = \frac{|S[k]|}{|G[k]|} \ , \quad k = 0, \ldots, N-1 \ . \tag{7.5}$$

Unfortunately, there are some practical problems with this model. First, we need to realize that real-world signals are never truly periodic. Some signals (like vowel sounds) are *approximately* periodic, but the signal actually varies somewhat between each "period." Figure 7.2 shows a part of an actual speech signal in the lower left panel. Notice the small variations between each "period" of the signal. For a signal that is nearly periodic, we

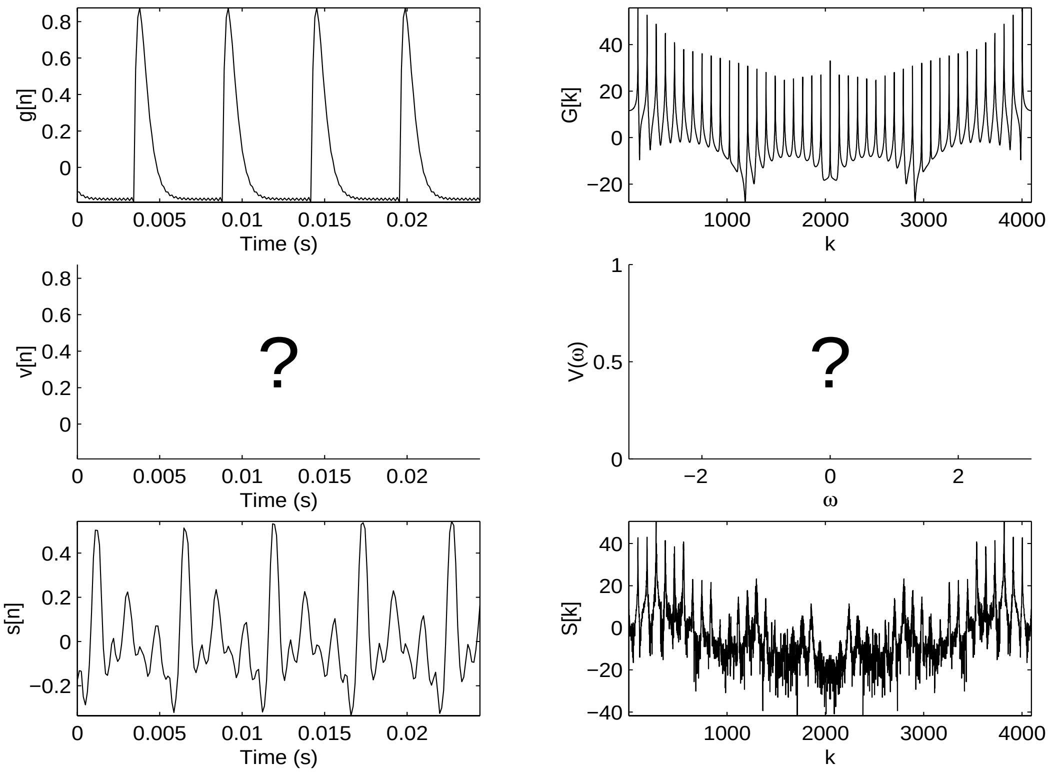


Figure 7.2: The components of the source-filter model system in the time-domain and the frequency-domain. We need to determine $v[n]$ and $V(\omega)$.

can still say that it has a fundamental period; however, this will never be more than an approximation. Another problem arises when we realize that the fundamental "period" is unlikely to be exactly an integral number of samples. The idealized glottal source signal shown in the upper panel left of Figure 7.2 was truly periodic in continuous time. However, its period was not an integral number of samples long, so the sampled signal is not truly periodic either. However, as we will see in the next section, it turns out that we will still be able to use the DFT to analyze the frequency-domain properties of our signals.

There is one last problem with our model. Equation (7.5) actually only gives *samples* of the frequency response $V(\widehat{\omega})$. This is generally not sufficient to completely characterize the filter. After all, the filter has a response at frequencies other than just those we've sampled $(0, 2\pi/N, 4\pi/N, \ldots 2\pi(N-1)/N)$. However, if we assume that the frequency response of the filter is varying slowly compared to the spacing of the samples of $V(\widehat{\omega})$, those samples should be sufficient to allow us to model $V(\widehat{\omega})$.

7.2.2 The Spectrum of Arbitrary Signals via the DFT

We have seen in class and in lab that the frequency domain (the frequency spectra of signals and the frequency response of filters) provides a powerful framework for analysis and synthesis of signals and filters. However, the theory developed in class has been exclusively limited to periodic signals, whereas, as mentioned above, real-world signals are never perfectly periodic. Indeed, many are very far from periodic. As we now discuss, frequency domain methods can be applied to arbitrary discrete-time signals by applying the DFT to finite segments of signals.

Consider an arbitrary signal $x[n]$. We can choose some integer N and apply the DFT to the N samples $x[0], \ldots, x[N-1]$, giving the N DFT coefficients

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{k}{N} n} \ , \quad k = 0, \ldots, N-1 \ . \tag{7.6}$$

Of what value are these coefficients? According to the basic DFT synthesis formula,

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{j2\pi \frac{k}{N} n} \ , \quad n = 0, \ldots, N-1 \ . \tag{7.7}$$

That is, $x[n]$ is the sum of N complex exponentials with frequencies 0, $2\pi\frac{1}{N}$, $2\pi\frac{2}{N}$, $2\pi\frac{3}{N}$, $\ldots$, $2\pi\frac{(N-1)}{N}$ and with complex amplitudes $X[0], \ldots, X[N-1]$, respectively. To emphasize this, let us rewrite the synthesis formula as

$$x[n] = X[0] + X[1]e^{j2\pi\frac{1}{N}n} + X[2]e^{j2\pi\frac{2}{N}n} + \ldots + X[N-1]e^{j2\pi\frac{N-1}{N}n} \ , \quad n = 0, \ldots, N-1 \ . \tag{7.8}$$

Thus $X[0], \ldots, X[N-1]$ tell us the spectrum of $x[n]$ in the time interval $0, 1, \ldots, N-1$.

Notice, however, that if, as usual, $x[n]$ is not periodic with period N, then equations (7.7) and (7.8) do not hold when $n < 0$ or $n \geq N$. The righthand side is periodic with period N and the lefthand side is not. The idea is that we have analyzed the frequency content of the signal only within the time interval $0, 1, \ldots, N-1$. One can similarly analyze the frequency content in any interval of length N, e.g. $n = N_0, N_0+1, \ldots, N_0+N-1$.

Another important thing to notice is that the *resolution* of the frequency domain analysis increases with N. By this we mean that spacing between adjacent frequency components, namely, $\frac{2\pi}{N}$ decreases with N.

In the following we summarize a number of important properties of the DFT applied in the way just described.

1. If you take the DFT of N samples of a signal, namely $x[0], \ldots, x[N-1]$, the result is a vector of N samples $X[0], \ldots, X[N-1]$. (Note that in MATLAB we access $x[n]$ and $X[k]$ with `x(n+1)` and `X(k+1)`, respectively, since there are no `x(0)` and `X(0)` elements.)

2. $X[0]$ is the average value (the DC term) of the input signal.

3. In general, $X[k]$ is complex. Thus, if we want to display the $X[k]$, we need to take it's magnitude first (i.e., `abs` in MATLAB).

4. Because $e^{2\pi \frac{N-k}{N} n} = e^{2\pi \frac{-k}{N} n}$ for every n, the coefficient $X[N-k]$ can be interpreted as the spectral component at frequency $-2\pi \frac{k}{N}$.

5. If $x[n]$ is real (which it usually is), $X[k]$ will be conjugate symmetric. That is, $X[k] = X[N-k]^*$. It also implies that $|X[k]| = |X[N-k]|$ and $\angle X[k] = -\angle X[N-k]$. Generally, these facts mean that we are only interested in the first half of the DFT.

6. To determine the frequency in Hz, f_k, represented by sample $X[k]$, use the relation

$$f_k = \frac{kf_s}{N}. \tag{7.9}$$

   This is easy to remember. Just think that $X[N]$ would correspond to the sampling frequency, f_s; to convert N to f_s, we simply divide by N and multiply by f_s.

7. The following are the DFT's of some commonly occuring idealized signals.

   (a) $x[n] = \cos\left(\frac{2\pi}{N} kn\right)$

$$\longrightarrow (X[0], \ldots, X[N-1]) = (0, \ldots, \frac{1}{2}, 0, \ldots, 0, \frac{1}{2}, 0, \ldots, 0)\ , \tag{7.10}$$

   where the nonzero coefficients are at positions k and $N-k$.

   (b) $x[n] = \sin\left(\frac{2\pi}{N} kn\right)$

$$\longrightarrow (X[0], \ldots, X[N-1]) = (0, \ldots, \frac{1}{2j}, 0, \ldots, 0, -\frac{1}{2j}, 0, \ldots, 0)\ , \tag{7.11}$$

   where the nonzero coefficients are at positions k and $N-k$.

   (c) $x[n] = \cos\left(\frac{2\pi}{N} kn + \phi\right)$

$$\longrightarrow (X[0], \ldots, X[N-1]) = (0, \ldots, \frac{1}{2}e^{j\phi}, 0, \ldots, 0, \frac{1}{2}e^{j\phi}, 0, \ldots, 0)\ , \tag{7.12}$$

   where the nonzero coefficients are at positions k and $N-k$.

   (d) $x[n] = \cos\left(\left(\frac{2\pi}{N}k + \epsilon\right) n\right) \quad \longrightarrow \quad (X[0], \ldots, X[N-1])$, where typically all the $X[k]$'s are nonzero, and typically all have small magnitudes except those corresponding to frequencies closest to $\frac{2\pi}{N}k + \epsilon$.

   (e) $x[n] = \delta[n] \quad \longrightarrow \quad (X[0], \ldots, X[N-1]) = \left(\frac{1}{N}, \ldots, \frac{1}{N}\right)$.

8. In MATLAB, we perform the DFT by using the function `fft`. FFT stands for the *fast fourier transform*; this is a particularly efficient implementation of the DFT that is $O(N \log N)$ rather than $O(N^2)$. It is often advantageous to compute DFTs with lengths equal to powers of 2 (i.e., 256, 512, 1024, 2048, 4096, etc.). This speeds the calculation of the FFT considerably.

The top subplot of Figure 7.3 shows a portion of a real-world DTMF signal, `x`. Note that this signal is not truly periodic, nor do these 1024 samples represent one period of a larger signal. However, we can still take the signal's DFT. The magnitude of the signal's DFT is shown in the lower subplot of Figure 7.3.

This plot illustrates many of the properties enumerated above. First, you can clearly see the symmetry in the DFT magnitude plot. The two sinusoids that make up the DTMF tone can be seen clearly. However, notice that they do not show up as true impulses; instead they

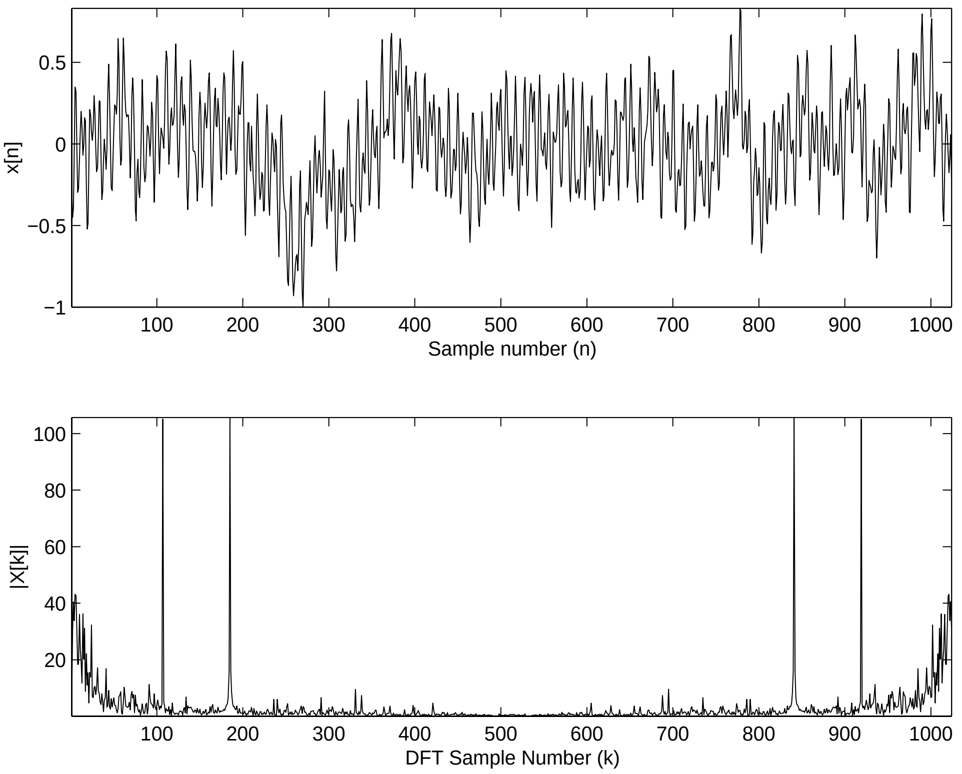


Figure 7.3: A "real-world" DTMF tone with 1024 samples (top) and the magnitude of the DFT of this signal (bottom).

have some width. This is because the DFT was not calculated over an integral number of periods of these sinusoids. Because of this, we see some "smearing" in the frequency domain. Also, notice the rough ("noise-like") characteristic of both the signal and the spectrum, especially at low frequencies. This is inevitable whenever we deal with real signals. The main thing that you should take away from this figure is that the DFT is rarely as clean as we would like it to be, and so we must interpret it to find relevant features.

7.2.3 FIR Filters on the Z-plane

In Chapter 7, you've seen that if we have an FIR filter of the form

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + b_3x[n-3] + \cdots + b_Mx[n-M], \tag{7.13}$$

we can take the Z transform of this filter response and write a corresponding system function, $H(z)$, as

$$H(z) = b_0 + b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + \cdots + b_Mz^{-M}. \tag{7.14}$$

We can further factor this complex-valued polynomial as

$$H(z) = (1 - a_1z^{-1})(1 - a_2z^{-1})(1 - a_3z^{-1}) \cdots (1 - a_Mz^{-1}). \tag{7.15}$$

The the complex values $\{a_1 \dots a_M\}$ are *zeros* of the system function $H(z)$. Much like the zeros of a real-valued polynomial, these are values of the independent variable z for which the system function goes to zero. Also like real-valued polynomials, the set of zeros completely characterizes the system function. This is especially useful because the location of zeros on the z-plane provides a wide variety of information about the system under consideration, including an intuitive understanding of the system's frequency response.

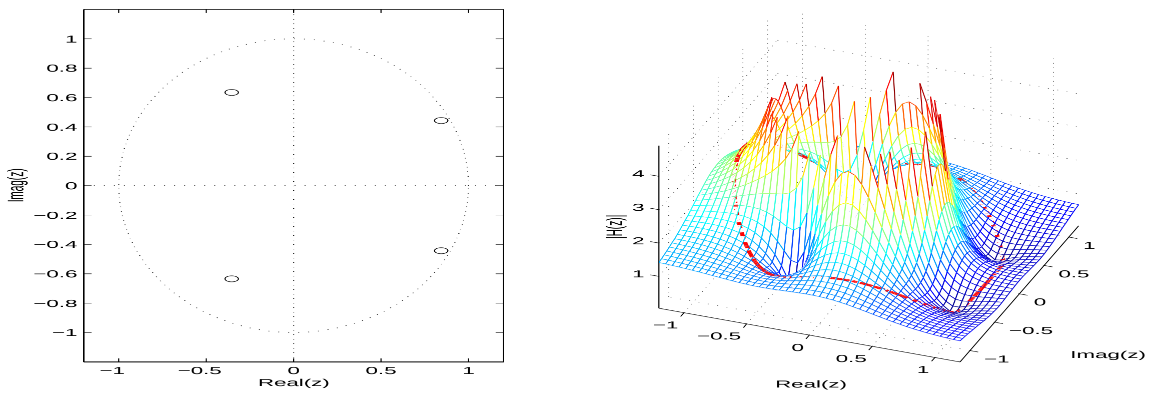


Figure 7.4: A plot of zeros on the z-plane with the unit circle (left); a plot of the magnitude of the equivalent system function, with the response at around the unit circle outlined in red (right).

Figure 7.4 shows the plot of the zeros of a four-zero system function. It also shows the magnitude of the system function on the z-plane. Notice how the zeros "push down" the system function in the right plot. On the z-plane, the most important area is the unit circle, because the value of the system response on the unit circle is equal to the system's frequency response. That is, the magnitude of $H(z)$ at $z = 1\angle\omega$ is equal to the magnitude of the frequency response at a frequency of ω. The magnitude of the system response at the unit circle has been outlined in red in Figure 7.4. This idea allows us to obtain an intuitive understanding of the effects of zeros on the frequency response of the system. If a zero is placed close to the unit circle, there will be a "low point" in the magnitude response near that frequency.

It is often useful to design a filter by placing zeros on the z-plane. The effect of zero locations on the system's frequency response is much clearer than the effect of changes in the filter's coefficients, and this allows for rapid filter design. This design method is also useful if we wish to fit a low-order digital filter to a more complicated frequency response. We can often find a simple design that sufficiently models the frequency response without a great deal of complexity. In this lab, we will use the zero placement design methodology to fit low-order filter models to estimated vocal tract transfer functions.

Before we conclude, we will briefly summarize a number of important points about z-plane representations of FIR filters.

1. An FIR filter of order M will have M zeros.

2. If the FIR coefficients $\{b_0, \ldots, b_M\}$ are real (which they usually are), then all zeros will either be real or will appear in complex conjugate pairs.

3. The value of $H(z)$ at any $z = e^{j\widehat{\omega}}$ is equivalent to the system's frequency response at discrete radian frequency $\widehat{\omega}$. That is, the system's discrete frequency response is located on the unit circle.

4. Zeros placed near the unit circle will tend to "pull down" the frequency response near them and will tend to "push up" the frequency response on the opposite side of the unit circle.

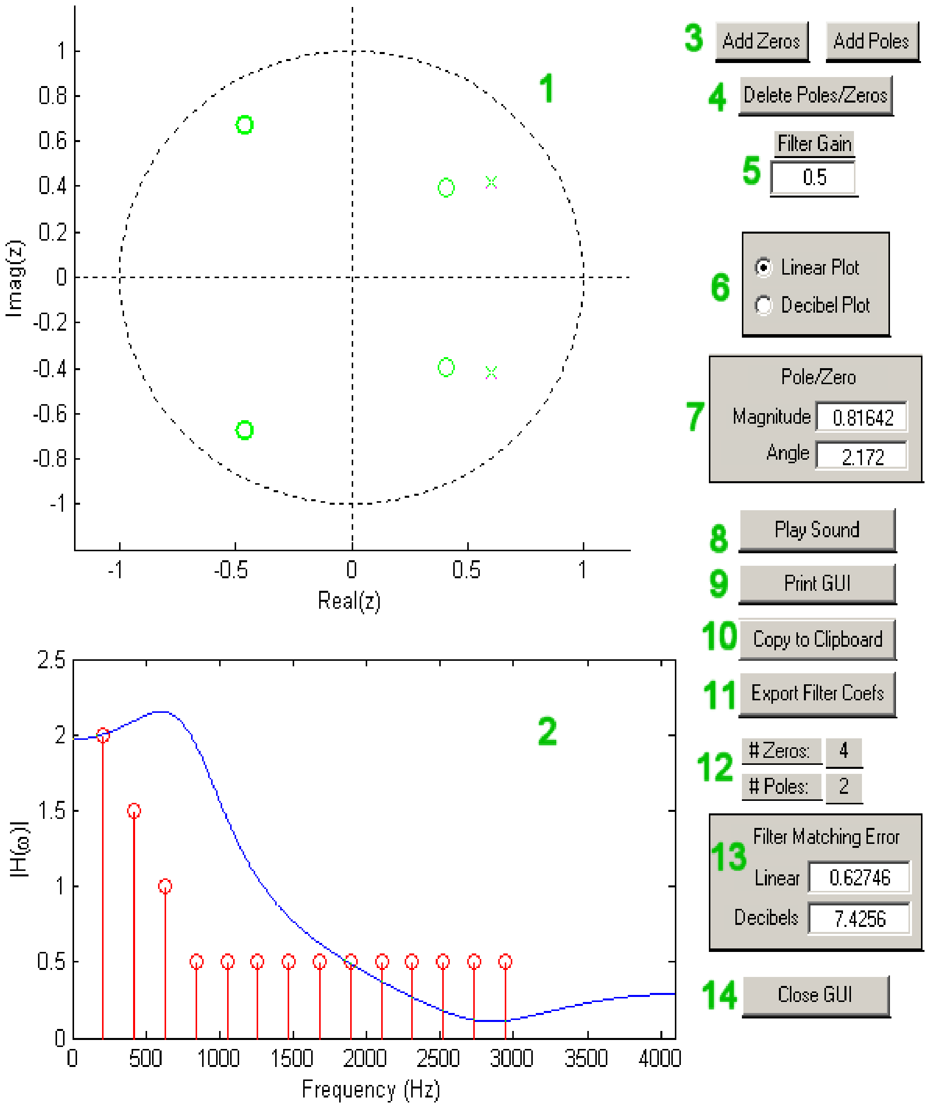
Though we will not discuss them in depth, you will also be experimenting with the use of *poles* on the z-plane. Poles are one of the fundamental building blocks of IIR filters. They have many of the same properties of zeros, but in another sense they are the opposite of zeros; they "push up" the spectrum where zeros pull it down. In fact, poles are locations on the z-plane where the system function goes to infinity, so they also present some stability issues. When using poles for filter design, make sure that you *always* keep the poles inside the unit circle.

7.2.4 *Pole-Zero Place* – A Filter Design GUI

Pole-Zero Place is a MATLAB[1] graphical user interface that will help to design discrete-time filters to match a particular frequency response. It does this by letting you place poles and zeros on the z-plane and watch the effect they have on the frequency response of the filter being designed. In the lab assignment, you will use *Pole-Zero Place* to match filters to an estimated vocal tract transfer function.

Two files make up *Pole-Zero Place* – `pole_zero_place.m` and `pole_zero_place.fig`. You must have both files to run the program. To execute *Pole-Zero Place*, execute the

[1]Note that *Pole-Zero Place* requires MATLAB version 6.0 or higher. Notably, this means that it *cannot* be run on the Macintosh



Filter Gain: 0.5

[✓ Linear Plot] Decibel Plot

Pole/Zero
Magnitude: 0.81642
Angle: 2.172

Zeros: 4
Poles: 2

Filter Matching Error
Linear: 0.62746
Decibels: 7.4256

Figure 7.5: The *Pole-Zero Place* GUI for discrete-time filter design.

command

```
pole_zero_place(gains,fund_frq);
```

where `gains` is a vector of gains at harmonically related frequencies for a desired frequency response and `fund_frq` is the fundamental frequency in Hz (assuming a sampling frequency of 8192 Hz). For example, if the magnitude of our desired frequency response is 10 at 300 Hz, 1 at 600 Hz, and 0.1 at 900 Hz, we would use the command

```
pole_zero_place([10, 1, .1],300);
```

The function also has optional third and fourth parameters that allow you to specify the feedforward and feedback filter coefficients (i.e., `B` and `A`), so that the GUI will start with an initial filter configuration. (This is useful if we want to "save" the state of the GUI and come back to a particular design later).

Once the program starts, the GUI window will appear. This window is shown in Figure 7.5. Hopefully you will find the interface to be intuitive. Important features are labeled with numbers and described below.

1. The z-plane plot. The locations of current poles (x) and zeros (o) are plotted here. If a conjugate pair is currently selected, it will be highlighted. On this plot, you can select poles and zeros and drag them around the z-plane. As you move them, you can see the effect on the frequency response (2).
2. The filter frequency response is plotted here in blue. The stem plot in red indicates the positions of the desired transfer function gains that we input to GUI when we started it. Note that this plot ranges from 0 to $f_s/2$, where our $f_s = 8192$ Hz.
3. Press the "Add zeros" or "Add poles" button and then click on the z-plane plot (1) to add a conjugate pair of zeros or poles.
4. Press the "Delete poles/zeros" button to delete a selected pair of zeros.
5. The filter gain determines the overall height of the frequency response. Change it by entering a different number in this box.
6. These radio buttons switch the frequency response plot (2) between linear-scale magnitude and decibels.
7. These edit boxes indicate the location of the currently selected zero. You can edit the magnitude and angle to change those values for the selected zero.
8. This button lets you hear the result of your filter on a glottal source signal (requires the `glottal_source.m` file).
9. This button opens a dialog so that you can print the GUI window.
10. This button copies the GUI into the Windows clipboard (only on Windows machines).
11. This button displays the filter coefficients to the command window so that you can copy them into a variable and use them or export them to another program. The vector labeled `B` contains the *feedforward* coefficients of the filter function (i.e., the regular FIR filter coefficients that multiply $x[n - n_0]$). The `A` vector contains *feedback* terms introduced by the inclusion of poles on the z-plane. These terms multiply previous values of the output, $y[n - n_0]$.
12. These indicate how many poles and zeros you have on the plane.
13. These edit boxes show the RMS error between your filter's frequency response and the desired response.
14. This button displays the filter coefficients and then closes the GUI window.

7.2.5 MATLAB Command Refresher

There are a number of MATLAB commands that you'll need for this lab. You have probably used most of them we've used before, but they are included here for reference. Note that this list is *not* meant to be exhaustive.

- To compute the DFT of a vector `x`, use the command

  ```
  X = fft(x);
  ```

- To compute the magnitude of the DFT of a vector `x`, we use the command

  ```
  X = abs(fft(x));
  ```

- To convert a value (or a vector of values) `x` to decibels, use the command

  ```
  xdb = 20*log10(x);
  ```

- To convert a value (or vector of values) `xdb` back from decibels, use the command

  ```
  x = 10.^(xdb/20);
  ```

- To listen to a signal in a vector `x` with a sampling frequency of `fs`, use the command

  ```
  soundsc(x,fs);
  ```

7.3 Demonstrations in the lab session

1. Interpreting the DFT
2. Source-filter models
3. Filter design on the z-plane
4. *Pole Zero Place*
5. A quick introduction to IIR Filters

7.4 Laboratory assignment

Download the file `lab7_vowels.mat` and load it into your workspace. This .MAT file contains two vowel signals, `ee_8192` and `oh_8192`, both sampled at 8192 Hz. These signals are actual recordings, not synthesized vowels like those we dealt with in Lab #3. Among other things, this means that the signals are not strictly periodic. These signals will be our $s[n]$ as described Section 7.2.1. Our ultimate goal is to find low-order filter models for the vocal tracts that were used to produce these signals. There are a number of steps in this process, and we will address each one in turn. We will focus on the signal `ee_8192` for this lab, but you might be interested in analyzing `oh_8192` as well.

1. In order to estimate the vocal tract transfer function, we first need to determine the glottal source signal, $g[n]$. We will assume that most of the parameters of the source signal are known, but you still need to determine an appropriate fundamental frequency. Recall that the period of the glottal source will be the same as the period of the speech signal. A slight difficulty arises because the speech signal does not have a constant period.

   (a) For the `ee_8192` signal, take the DFT of the first 4096 samples of the signal. Take the magnitude of the resulting signal, and convert to decibels.

- [4] In a single figure, plot the magnitude of the DFT in one subplot and the magnitude of the DFT in decibels in another subplot below it. (Note: to prevent confusion later, you should use the vector `0:4095` as your x-axis in both plots).

(b) This is a lot happening in this signal, but the relevant features of the DFT are the harmonic peaks, which should be evident in both figures (though perhaps somewhat more so in the decibel plot). Estimate the frequencies (in Hz) and amplitudes of the first 24 harmonic peaks. (Hint: Remember that you should only consider peaks on the first half of the DFT plot. Also, consider using the `ginput` to speed up the process.)

- [8] Produce a table that includes the *harmonic number* of each peak (where the n^{th} harmonic has a harmonic number of n), its location in the DFT vector (i.e., its sample number), and its estimated frequency in Hz. Also include the magnitude of the DFT at each harmonic and this magnitude expressed in decibels.
- [2] Use `stem` to plot the harmonic magnitudes (in decibels) versus their estimated frequencies in Hz.

(c) The next step is to estimate the fundamental frequency of the underlying glottal source signal. If we know the harmonic number and the frequency of a single harmonic of the speech signal, we can easily come up with an estimate of the fundamental frequency. Come up with a way to use your estimated frequencies for *all* of the harmonics to produce an estimate of the glottal source fundamental frequency.

- [4] Describe how you did this.
- [2] What is your estimated fundamental frequency?

2. Download the file `glottal_source.m`. This function generates a simple glottal source signal. It takes two parameters – a fundamental frequency, and a signal length (in seconds). Once we have this signal, we can estimate the frequency response of the vocal tract at the harmonics of our fundamental frequency.

(a) Create a glottal source signal, `g_e`, at the fundamental frequency that you found for the signal `ee_8192`. Your signal should be 0.5 seconds long.

- [4] In a single figure, plot the magnitude of the DFT of the source signal in one subplot and the magnitude of the DFT of the signal in decibels in a second subplot. (Note: Again, use the vector `0:4095` as your x-axis in both plots).
- [2] Verify that the fundamental frequency of this source signal is the same as the one that you estimated in Problem 1c. Briefly describe how you did this.

(b) Now, you need to determine the magnitudes of the first 24 harmonics of `g_e`. Identify the magnitudes of each harmonic of `g_e` in decibels.

- [2] Use `stem` to plot the magnitude of the harmonics in decibels versus their frequency. (Hint: You can assume that the harmonics are strict multiples of the fundamental frequency. You don't need to re-estimate their frequencies.)

(c) Finally, we can estimate the magnitude frequency response of the vocal tract filter at the harmonics of our fundamental frequency. Calculate these values by dividing the magnitude of the n^{th} harmonic of `ee_8192` by the magnitude of the n^{th} harmonic of `g_e`. (Hint: You can also perform this calculation in decibels. Division is the same as subtraction in decibels.)

- [6] Produce a table that includes the harmonic number, the harmonic frequencies in Hz, the magnitude frequency response of the vocal tract, and the magnitude frequency response in decibels.

- [2] Use `stem` to plot the resulting magnitude frequency response of the vocal tract in decibels versus the harmonic frequencies in Hz.

3. Now, download the files `pole_zero_place.m` and `pole_zero_place.fig`. This is the *Pole-Zero Place* program described in Section 7.2.4. The function `pole_zero_place.m` takes two parameters. The first is a vector of target vocal tract gains (which you calculated in Problem 2c). These gains should *not* be represented in decibels. The second is the fundamental frequency of these harmonics (which you calculated in 1c).

   (a) Execute `pole_zero_place` with the vocal tract gains and fundamental frequency that you have calculated. Find a design using *six zeros* (i.e., three conjugate pairs) that matches the vocal tract gains as closely as possible. Do not use any poles. You can try to minimize one of the two error metrics, or you can try to find a filter that "sounds the best." Your resulting resynthesis (when you hit "Play Sound") should roughly like the vowel "ee,"as in the word "street."

   - [10] Include the GUI window containing your filter design in your report.
   - [2] What criteria did you use for optimizing your filter?
   - [2] What are the FIR coefficients (i.e., the `B` vector) for your filter?

   (b) Repeat for a design using *twelve zeros* (i.e., six conjugate pairs). Again, do not use any poles.

   - [10] Include the GUI window containing your filter design in your report.
   - [2] What criteria did you use for optimizing your filter?
   - [2] What are the FIR coefficients (i.e., the `B` vector) for your filter?

4. Using IIR filters provides us with a new range of flexibility in filter design. Unlike FIR filter design, where it is difficult to get high gain at a single frequency, IIR filters excel at providing a great deal of localized gain. Execute `pole_zero_place.m` again. This time, we will use poles as well as zeros to design our filters.

   (a) Use `pole_zero_place` to find filter design with *six poles or zeros* (i.e., three conjugate pairs). You can use all poles, some poles and some zeros, or all zeros; just make sure that you have a combined total of six poles and zeros. Optimize your design with respect to one of the criteria listed in Problem 3a.

   - [10] Include the GUI window containing your filter design in your report.
   - [2] What criteria did you use for optimizing your filter?
   - [2] What are the filter coefficients (both `B` and `A`) for your design?

   (b) Repeat the design for *twelve poles or zeros* (i.e., six conjugate pairs). That is, make sure that the combined total number of poles and zeros is twelve.

   - [10] Include the GUI window containing your filter design in your report.
   - [2] What criteria did you use for optimizing your filter?
   - [2] What are the filter coefficients (both the `B` and `A` vectors) for your design?

   (c) Consider your experiences with zero-only and pole-zero filter design.

   - [4] Compare the zero-only design process to the pole-and-zero design process. What was difficult about each task? Which was easier? Which produced better overall results?
   - [2] Do you think that zero-only design allows you to find a good approximation to your desired transfer function with a small number of zeros?
   - [2] Do you think that pole-zero design allows you to find a good approximation to your desired transfer function with a small number of poles and zeros?

- [2] Do you think you need more poles or zeros to achieve a “good” approximation? How many do you need? (Hint: try some designs with more poles and zeros to determine this.)