GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Spring 2003
Lab #10: PeZ - The z, n, and $\hat{\omega}$ Domains

Date: 25–31 Mar 2003

You should read the Pre-Lab section of the lab and do all the exercises in the Pre-Lab section before your assigned lab time. You **MUST** complete the online Pre-Post-Lab exercise on Web-CT at the beginning of your scheduled lab session. You can use MATLAB and also consult your lab report or any notes you might have, but you cannot discuss the exercises with any other students. You will have approximately 20 minutes at the beginning of your lab session to complete the online Pre-Post-Lab exercise. The Pre-Post-Lab exercise for this lab includes some questions about concepts from the previous Lab report as well as questions on the Pre-Lab section of this lab.

The Warm-up section of each lab must be completed **during your assigned Lab time** and the steps marked *Instructor Verification* must also be signed off **during the lab time**. After completing the warm-up section, turn in the verification sheet to your TA.

Forgeries and plagiarism are a violation of the honor code and will be referred to the Dean of Students for disciplinary action. You are allowed to discuss lab exercises with other students and you are allowed to consult old lab reports but the submitted work should be original and it should be your own work.

The lab report for this week will be an **Informal Lab Report.** It is only necessary to turn in Section 4 as this week's lab report. The report will **due the next time your lab meets: 1–7 April**.

---

1 Introduction & Objective

The objective for this lab is to build an intuitive understanding of the relationship between the location of poles and zeros in the z-domain, the impulse response $h[n]$ in the n-domain, and the frequency response $H(e^{j\hat{\omega}})$ (the $\hat{\omega}$-domain). A graphical user interface (GUI) called **PeZ** was written in MATLAB for doing interactive explorations of the three domains.[1] **PeZ** is based on the system function, represented as a ratio of polynomials in z^{-1}, which can be expressed in either factored or expanded form as:

$$H(z) = \frac{B(z)}{A(z)} = G\,\frac{\prod_{k=1}^{M}(1 - z_k z^{-1})}{\prod_{\ell=1}^{N}(1 - p_\ell z^{-1})} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{\ell=1}^{N} a_\ell z^{-\ell}} \tag{1}$$

2 Pre-Lab

There are two version of the **PeZ** GUI: the original one written for versions 4 and 5 of MATLAB; and a newer one for version 6. Both versions are contained in the *SP-First* toolbox. To run **PeZ**, type `pezdemo` at the command prompt and you will see the GUI shown in Fig. 1.[2]

[1] The original **PeZ** was written by Craig Ulmer; a later version by Koon Kong is the one that we will use in this lab.

[2] The command `pez` will invoke the older version of **PeZ** which is distinguished by a black background in all the plot regions.

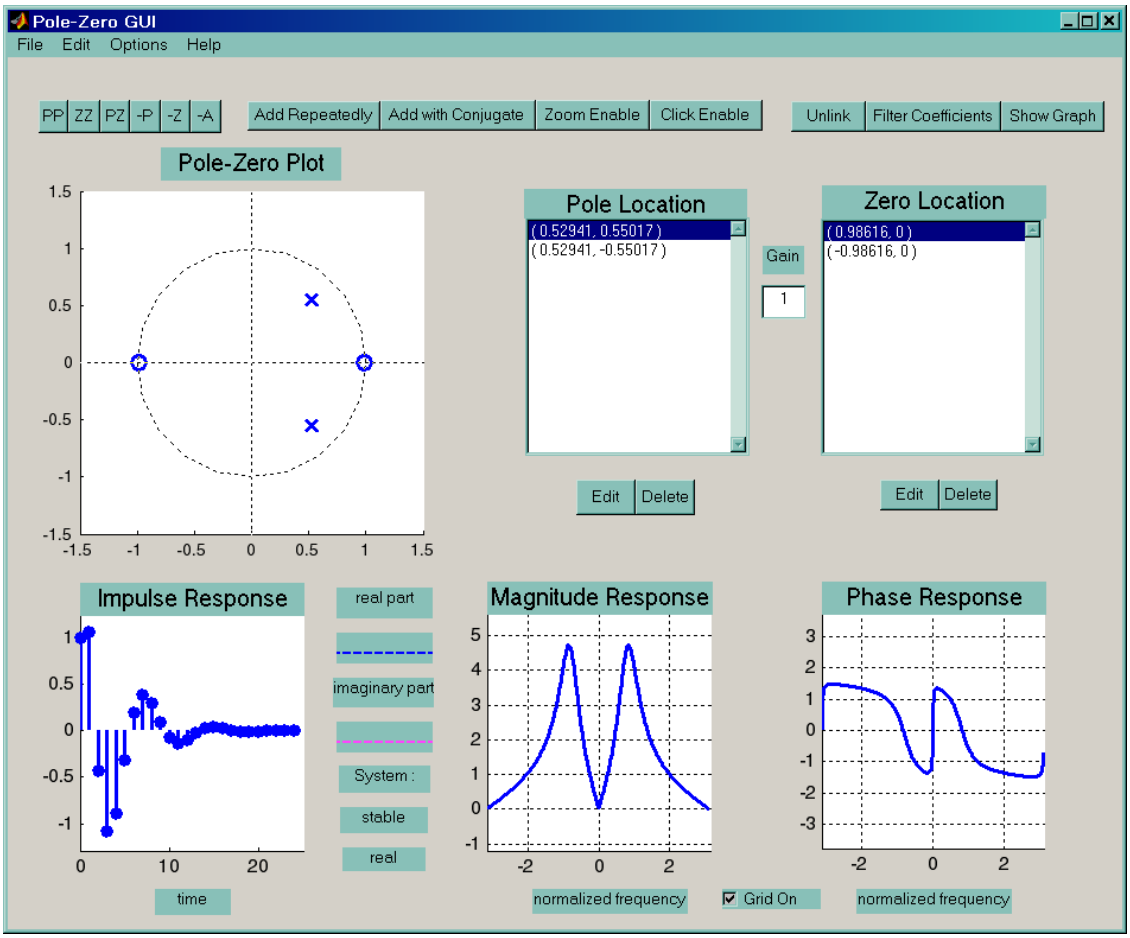

Figure 1: GUI interface for `pezdemo` running in MATLAB version 6. A 2nd-order bandpass filter is shown.

2.1 Controls for PeZ using `pezdemo`

The **PeZ** GUI is controlled by the `Pole-Zero Plot` where the user can add (or delete) poles and zeros, as well as move them around with the pointing device. For example, Fig. 1 shows a case where two (complex-conjugate) poles have been added, along with single zeros at $z = 1$ and $z = -1$. The buttons named `PP` and `ZZ` were used to add these poles and zeros. By default, the `Add with Conjugate` property is turned on, so poles and zeros are typically added in pairs to satisfy the complex-conjugate property:

> A polynomial with real coefficients has roots that are real, or occur in complex-conjugate pairs.

To learn about the other controls in `pezdemo`, access the menu item called "Help" for extensive information about all the **PeZ** controls and menus.

Here are a few things to try. You can use the `Pole-Zero Plot` to selectively place poles and zeros in the z-plane, and then observe (in the other plots) how their placement affects the impulse and frequency responses. In **PeZ** an individual pole/zero pair can be moved around and the corresponding $H(e^{j\hat{\omega}})$ and $h[n]$ plots will be updated as you drag the pole (or zero). Since exact placement of poles and zeros with the mouse is difficult, an `Edit` button is provided for numerical entry of the real and imaginary parts. Before you can edit a pole or zero, however, you must first select it in the list of `Pole Locations` or `Zero Locations`. Removal of individual poles or zeros can also be performed by using the `-P` or `-Z` buttons, or with the `Delete` button. Note that all poles and/or zeros can be easily cleared by clicking on the `-A` button.

2.2 Create an IIR Filter with PeZ

Play around with **PeZ** for a few minutes to gain some familiarity with the interface. Implement the following first-order system:

$$H(z) = \frac{1 - z^{-1}}{1 + 0.9z^{-1}}$$

by placing its pole and zero at the correct location in the z-plane. First try placing the pole and zero with the mouse, and then use the `Edit` feature to get exact locations. Since **PeZ** wants to add complex-conjugate pairs, you might have to delete one of the poles/zeros that were added; or you can turn off the `Add with Conjugate` feature. Look at the frequency response and determine what kind of filter you have.

Now, use the mouse to "grab" the pole and move it from $z = -0.9$ to $z = +0.8$. Observe how the frequency response changes. Describe the type of filter that you have created.

2.3 Importing Filters into PeZ

One useful option for connecting **PeZ** to other programs is the "Import/Export" feature under the `Filter Coefficients` button. The technique for exporting and importing is to use a "cell array" called `filter_co`, where `filter_co{1}` contains the denominator coefficients for $A(z)$, and `filter_co{2}` the numerator coefficients for $B(z)$. When exporting from **PeZ**, the `filter_co` cell array is saved into a `.mat` file which can be loaded and read into filter coefficient vectors via:

```
load filename.mat
a = sscanf(filter_co{1},'%f');
b = sscanf(filter_co{2},'%f');
```

The `filter_co` array cannot be used directly because it contains strings, not numbers, but `sscanf` reads the strings and converts them to a vector of numbers.

If you want to import filter coefficients into **PeZ**, then it is necessary to create the `filter_co` array and save it to a `.mat` file. For example,

```
filter_co{1} = num2str(a,'%15g');   % a must be a row
filter_co{2} = num2str(b,'%15g');   % b must be a row
save filename.mat filter_co
```

When the `Filter Coefficients` button is used and "import" is chosen, a file open dialog will be presented from which `filename.mat` can be selected.[3]

Use the "import coefficients" feature to create the pole-zero diagram for a length-11 FIR filter whose impulse response is given by $h[n] = \cos(4\pi n/11)$ for $n = 0, 1, 2, \ldots, 10$. From the frequency response, determine the type of filter (e.g., LPF, HPF, or BPF).

3 Warm-up

The lab verification requires that you write down your observations when using the PeZ GUI.

3.1 Relationships between z, n, and $\hat{\omega}$ domains

Work through the following exercises and keep track of your observations by filling in the worksheet at the end of this assignment. In general, you want to make note of the following quantities:

- How does $h[n]$ change with respect to its rate of decay? For example, when $h[n] = a^n u[n]$, the impulse response will fall off more rapidly when a is smaller.

[3] The "Import" under the "File" menu cannot be used, because that import uses a different format.

- If $h[n]$ exhibits an oscillating component, what is the period of oscillation? Also, estimate the decay rate of the "envelope" that overlays the oscillation.
- How does $H(e^{j\hat{\omega}})$ change with respect to peak location and peak width?

Note: review the "Three-Domains - FIR" under the Demos link for chapter 7 and "Three-Domains - IIR" under the Demos link for chapter 8 for movies and examples of these relationships.

3.2 Real Poles

(a) Use **PeZ** to place a single pole at $z = \frac{1}{2}$. You may have to use the `Edit` button to get the location exactly right. Use the plots for this case as the reference for answering the next five parts.

(b) Move the pole close to the origin (still on the real axis). You can do this by dragging the pole to the new location. Describe the changes in the impulse response $h[n]$ and the frequency response $H(e^{j\hat{\omega}})$.

(c) When you move poles and zeros, the impulse response and frequency response plots are updated continually. Select the pole you want to move and start to drag it slowly. Watch for the update of the plots in the secondary window.

Move the real pole slowly from $z = \frac{1}{2}$ to $z = 1$ and observe the changes in the impulse response $h[n]$ and the frequency response $H(e^{j\hat{\omega}})$.

(d) Place the pole exactly on the unit circle (or maybe just inside at a radius of 0.99999999). Describe the changes in $h[n]$ and $H(e^{j\hat{\omega}})$. What do you expect to see for $H(e^{j\hat{\omega}})$?

(e) Move the pole outside the unit circle. Describe the changes in $h[n]$. Explain how the appearance of $h[n]$ validates the statement that the system is not stable. In this case, the frequency response $H(e^{j\hat{\omega}})$ is not legitimate because the system is no longer stable.

(f) In general, where should poles be placed to guarantee system stability? By stability we mean that the system's output does not blow up.

3.3 Complex Poles and Zeros

PeZ assumes real coefficients for the numerator and denominator polynomials when the `Add with Conjugate` mode is on (which it is by default). Therefore, if we enter a complex pole or zero, **PeZ** will automatically insert second root at the conjugate location. For example, if we place a root at $z = \frac{1}{3} + j\frac{1}{2}$, then we will also get one at $z = \frac{1}{3} - j\frac{1}{2}$.

(a) What property of the polynomial coefficients of $A(z) = 1 - a_1 z^{-1} - a_2 z^{-2}$ will guarantee that the roots come in conjugate pairs?

(b) Clear all the poles and zeros from **PeZ**. Now place a pole pair with magnitude 0.85 at angles of $\pm 45°$; and then two zeros at the origin. Note that **PeZ** automatically places a conjugate pole in the z-domain. The frequency response has a peak—record the frequency (location) of this peak.

(c) Change the angle of the pole: move the pole to 90°, then 135°. Describe the changes in $|H(e^{j\hat{\omega}})|$. Concentrate on the location of the peak.

Next, we will put complex zeros on the unit circle to see the effect on $|H(e^{j\hat{\omega}})|$. **PeZ** will automatically put poles at the origin to keep the filter causal.[4]

(d) Clear all poles and zeros from **PeZ**. Now place zeros at the following locations: $z_1 = -1$, $z_2 = 0 - j$ and $z_3 = 0 + j$ (remember that conjugate pairs such as z_2 and z_3 will be entered simultaneously). Judging from the impulse and frequency responses what type of filter have you just implemented?

[4]Recall that a length-L FIR filter whose impulse response extends from $n = 0$ to $N = L-1$ will have $L-1$ poles at $z = 0$.

4 Lab Exercise: Filter Design

In this section, we will use **PeZ** to place the poles and zeros of $H(z)$ to make a filter with a desirable frequency response. Then similar filters will be used to remove interfering sinusoids from a speech signal.

4.1 Notch Filter with Poles and Zeros

Filter design is a process that selects the filter coefficients $\{a_\ell\}$ and $\{b_k\}$ in (1) to accomplish a given task. The task here is to create a filter that has a very narrow "notch." Such a filter would be useful for removing one frequency component while leaving others undisturbed. One example of a *notch filter* can be synthesized from the cascade of two simpler filters shown in Fig. 2.

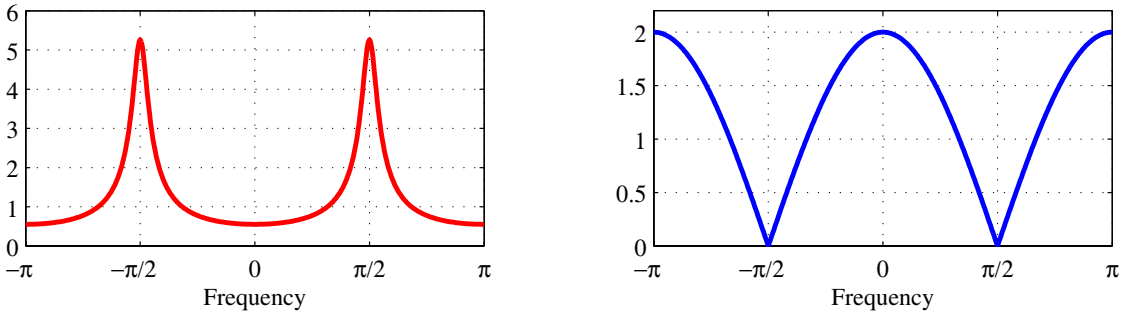

Figure 2: Magnitude response of two unknown filters. (a) Second-order IIR filter. (b) Second-order FIR filter. Frequency axis is normalized $\hat{\omega}$. Use **PeZ** to help you find the filter coefficients that will match these frequency responses as closely as possible.

(a) Start the process by using **PeZ** to design each of the filters given in Fig. 2. You will have to determine the locations of the poles and zeros by "trial and error" to match the plots in Fig. 2. Both filters are second-order, i.e. they have two zeros or two poles. Make sure that you enter the poles and zeros precisely. **PeZ** will do the conversion between between root locations and polynomial coefficients, but you could also do this with the MATLAB commands `roots` and `poly`. You can check your results by also calculating the filter coefficients by hand (see the next section on polynomials with complex coefficients). Determine the coefficients of each filter.
Note: Use **PeZ** or `freqz()` to verify that the frequency response of each filter is correct.

(b) Now use **PeZ** to connect the filters together in a cascade. Place the poles and zeros together, and then view the frequency response. Determine the filter coefficients for the overall cascaded filter $H(z)$.

(c) Use `freqz` to determine the frequency response of the cascade of the two filters that you "designed" in the previous part. Plot the magnitude of the overall frequency response of the cascade system for $-\pi < \hat{\omega} < \pi$, and include the plot for your lab report.[5] Explain briefly why the frequency response magnitude has a notch, and explain why the values of $|H(e^{j\hat{\omega}})|$ at $\hat{\omega} = 0$ and $\hat{\omega} = \pi$ are the same.

4.1.1 Filter Coefficients from Roots

Compute the filter coefficients for the denominator $A(z) = 1 - a_1 z^{-1} - a_2 z^{-2}$ and numerator $B(z) = b_0 + b_1 z^{-1} + b_2 z^{-2}$ when the poles are:

$$p_1 = 0.75e^{j\pi/4}, \quad p_2 = 0.75e^{-j\pi/4},$$

[5] **PeZ** uses the normalized frequency scale: $-0.5 < \hat{\omega}/(2\pi) < 0.5$, which is the same as $-\pi < \hat{\omega} < \pi$.

and the zeros are:

$$z_1 = 1, \quad z_2 = -1$$

Use the following relationship:

$$H(z) = \frac{B(z)}{A(z)} = G\frac{(1 - z_1 z^{-1})(1 - z_2 z^{-1})}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})}$$

where z_1 and z_2 are the zeros, and p_1 and p_2 are the poles defined above. Determine G so that the maximum magnitude of $H(e^{j\omega})$ is one. (Remember that MATLAB can multiply polynomials via its `conv` function.)

4.2 Removing Hum from a Speech Signal

FIR filters can be used to reject interfering signals that are sinusoidal. One situation where this might occur is in a tape recording of speech where the recorder is not adequately isolated from the power line signal which is a 60-Hz sinusoid. The recorded signal is actually the sum of two signals: the desired speech signal and a sinusoid, $A\cos(120\pi t + \phi)$. In this section, you will design an IIR notch filter to remove two interfering sinusoids, and then assess how much the desired signal is distorted by the filtering process.

(a) Load the file `lab10sig.mat` which contains two signals, `spc48`, which is the sum of a speech signal, `s1`, plus a interfering sinusoids at 400 and 800 Hz. The original utterance in `s1` was "the pipe began to rust while new" spoken by a female. The sampling rate of this signal is 8000 Hz. Listen to this signal to verify that the interference is so strong that the speech is not recognizable. You will have to process the signal `spc48` with *two different* filters.

(b) Use the example done in **PeZ** to write a simple MATLAB function that will design a second-order notch filter by finding the numerical values of the filter coefficients for the "notch frequency." The poles must be inside the unit circle (for stability), but you can experiment with different pole radii such as 0.9, 0.95, 0.98, and so on. Scale the filter coefficients for $B(z)$ so that the frequency response has a magnitude of one at $\hat{\omega} = \pi$.

(c) Design an IIR notch filter that will remove both sinusoids completely. You will need the cascade of two notch filters—one for each interfering sinusoid. Use the MATLAB function from the previous part. Plot the frequency response of the notch filter designed in the previous part.

(d) Process the corrupted signal, `spc48`, through the notch filter. Listen to the result and assess how successful the processing was. How well was the interference removed? When compared to the original, `s1`, how much was the speech signal degraded in the process?
Note: the MATLAB function `soundsc()` takes its scaling from the largest value in the output signal. Therefore, it will be necessary to ignore the beginning of the output signal because the transient region produced by the notch filter can be long. Examine the first 100 or 200 points of the output to see how much of the output must be ignored, i.e., use the colon notation in MATLAB to get `yy(start:end)` where `start` is some index after the "start-up" region.

(e) Process the corrupted signal, `spc48`, again, but use only the numerators of the notch filters. In effect, this will process the signal through two FIR *nulling* filters. Plot the frequency response (magnitude) of the cascaded nulling filters and determine the region(in hertz) that would be considered its stopband. Listen to the result, and compare this output to the output from the notch filter.

(f) In both cases (notch vs. nulling), use the frequency response (magnitude) to explain how the close the output signal is to the original speech signal. Your explanation will be qualitative, but should focus on how much of the low frequency and/or high frequency regions of the speech were altered.

Lab #10
ECE-2025
Spring-2003
WORKSHEET & VERIFICATION PAGE

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

Name: _____ Date of Lab: _____

Part	Observations
3.2(a)	$h[n]$ decays exponentially with no oscillations, $H(e^{j\hat{\omega}})$ has a hump at $\hat{\omega} = 0$
3.2(b)	
3.2(c)	
3.2(d)	
3.2(e)	
3.2(f)	
3.3(a)	
3.3(b)	
3.3(c)	
3.3(d)	