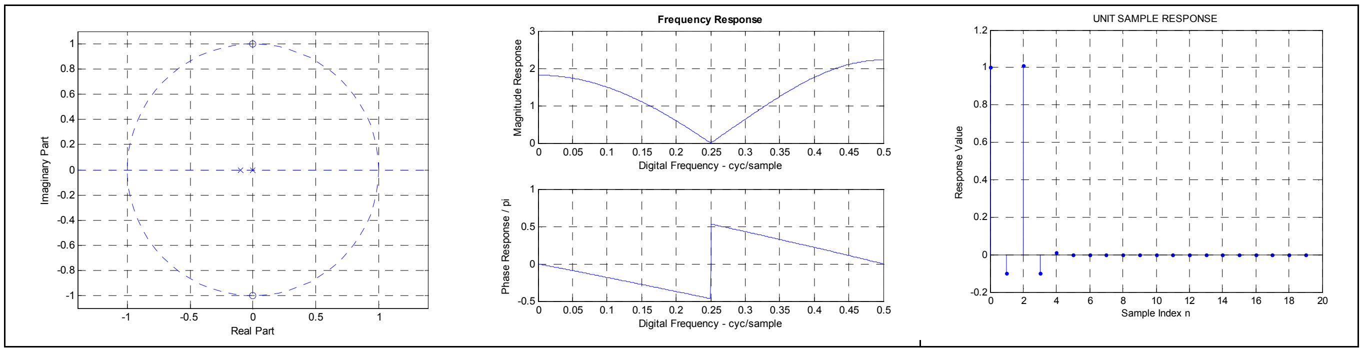
# EE 419 - Project 2
# Developing a Filter Analysis Program in Matlab



**Learning Objectives:**

**Use Matlab to quantify the performance of a discrete-time filter design, including:**

- **Pole-zero diagram**
- **Frequency Response plots (linear and dB magnitude and phase)**
- **Unit Sample Response**
- **Algorithmic determination of filter type**
- **Numeric quantification of filter bandwidth**

REMINDERS:
1) When you are finished, be sure to always copy your m-files to a place that you can retrieve them again in the future (copy them to a memory stick, email them to yourself, etc.).
2) Please then delete your files from the Lab Computer (so that everyone else has to work as hard as you!)

## 1) [Matlab] Creating a Unit Sample Response sequence

Using Matlab, create a **function** that plots and returns a Unit Sample Response sequence (the infamous and illusive "h[n]") for an arbitrary filter; beginning at n=0, and extending for the sequence length specified by a user. The function should have the form:

```
[hn, n]=unit_sample_response(Bk, Ak, number_of_samples, figure_number);
```

Where the values returned are:

```
hn – has the unit sample response sequence values
n – has the corresponding sample indices (starting at 0);
```

Where the function parameters (arguments) are:

```
Bk = a list of the Bk coefficients of the filter difference equation
       (coefficients of the "x" terms)
Ak = a list of the Ak coefficients of the filter difference equation
       (coefficients of the "y" terms)
number_of_samples = # of unit sample response sequence samples to find
figure_number = # figure for the hn sequence plot
```

Your Matlab function should

- Create a unit sample sequence using the function you created last week.
- Apply the unit sample sequence to a filter defined by the Ak and Bk coefficients.
- Create a stem plot of the response hn vs. n, in Figure # = *figure_number*
  - with small dots '.' at ends of the stems
  - with a title and all axes labeled

Test your program by finding the first 40 points of the impulse response using a filter with the following difference equation:

```
y[n] + 0.8 y[n-1] = x[n]
```

**Turn in the following:**

- A printout of the m-file for your function
- A printout of the figure created by your function for the test filter given above.

  **HELPFUL HINTS:** You can copy and paste a Matlab Figure into a document simply by choosing "Copy Figure" from the "File" menu of the Figure window created by Matlab. Then paste it into your document. You can then resize the figure easily in your document.

  If you are using MS Word, and you want to modify the figure (font sizes, *etc*), you right-click on the figure in your document, and select "Edit Picture." This allows you to convert the figure into a Microsoft Office drawing object. Then you can select and edit individual parts of the figure.

Refer to the Matlab Help information provided for: filter, stem, figure

## 2) [Matlab] Analyzing and Plotting the Frequency Response of a Filter

Using Matlab, create an m-file FUNCTION to analyze and provide frequency response plots for an arbitrary digital filter (FIR or IIR, any length); starting with a specification of the filter coefficients (Ak's and Bk's).

Your Matlab m-file should use the following function call format:

```
[poles,zeros,HF,Fd,hn,n] = show_filter_responses(Ak,Bk,fsample,
                            num_of_f_points,num_of_n_points,figure_num);
```

where the arguments are:

*Ak* = a list of the $A_k$ coefficients of the filter difference equation (coefficients of the "y" terms)
*Bk* = a list of the $B_k$ coefficients of the filter difference equation (coefficients of the "x" terms)
*fsample* = sampling frequency (samples / second)
*num_of_f_points* = the # of points for the freq. response plot
*num_of_n_points* = the # of points for the unit sample response plot
*figure_num* = number of the $1^{st}$ figure to use for plots

and the function returns the following information:

poles = a list of the complex pole locations (z values) for the Transfer Function (the roots of the H(z) denominator polynomial)
zeros = a list of the complex zero locations (z values) for the Transfer Function (the roots of the H(z) numerator polynomial)
HF = the complex DTFT frequency response values (linear scale)
Fd = digital frequencies that match the freq response values
hn – has the unit sample response sequence values
n – has the corresponding sample indices (0 to [*num_of_n_points* – 1]);

The user of this function would set up the Ak and Bk arrays for their filter design, and then use this function to analyze the design. This function should compute and **print the values determined for the filter's poles and zeros locations to the screen** (Matlab Command Window).

This function should then create the following system response plots:

- A "Pole/Zero" diagram

    - Plot in figure number $=$ *figure_num* .
    - Using command: **zplane(Bk, Ak)**;
    - With the grid turned on.
    - Provide a title for the plot.
- Two Frequency Response plots of H(F) vs F (digital frequency)
    - Using your **plot_freq_responses( )** function from last week.
    - Using the "num(ber) of f points" specified in the original **show_filter_responses()** function call.

  1) H(F) vs F (digital frequency): Using a <u>digital frequency</u> axis.
    - Showing only the frequency range from F=0 to F=0.5 (the Nyquist frequency), since the spectrum elsewhere is simply repeated or mirror-image versions of the spectrum in this range.
    - Plot in figure number $=$ *figure_num* + 1 .

  2) H(F) in dB vs $f_{\text{analog}}$: Using an equivalent analog frequency axis (assuming ideal sampling), from f = 0 Hz to f = $f_{\text{sample}}/2$.
    - Plot in figure number $=$ *figure_num* + 2 .
- A Unit Sample Response Plot h[n] vs n using your function from above.
    - Using the number of "n" points specified in the original **show_filter_responses( )** function call.
    - Plot in figure number = (*figure_num* + 3).

The Matlab Help information provided for the following functions may prove useful:

```
zplane, roots, freqz,
```

To help you to use this program in the future, you should document it with the following:

1) An exact duplicate of the `function` statement (at the top of the m-file, after the function declaration) as a comment.
   Example:
   ```
   function [ out1, out2] = my_function (in1, in2)
   % function [ out1, out2] = my_function (in1, in2)
   ```
2) Header comments (comments right after the **function** comment at the top of the file) that explain what the program does, and explains what each input and output arguments are. (These comments, and the function information from #1 above will be printed anytime someone types "help" followed by the name of your function.)
3) Comments (preceded by "%") explaining what you are doing in various blocks of code in the body of your m-file (or at the end of particular statement lines).

Test your program by finding the first 20 points of the impulse response, and 500 points of the frequency responses for a filter with the following diff. equation:

```
y[n] + 0.1 y[n-1] = x[n]+ x[n-2]
```

Assume a sampling rate of 1 KHz.

<u>**Turn in the following:**</u>

- What were your Ak and Bk values?
- What is H(z) for this filter?
- What were the values you found for your poles and zeros?
    - Ones you computed by hand (show how you derived them)
    - Ones your program returned
- A printout of the m-file for your function
- A printout of the figures created by your function for the test filter given above.

## 3) [Lab / Matlab] Analyzing and Plotting the Frequency Response of a Filter

Using Matlab, create a second m-file FUNCTION to analyze and provide frequency response plots for an arbitrary digital filter (FIR or IIR, any length), to aid in your designs of digital filters **using Pole/Zero Placement** (coming this week!).

Your Matlab m-file should use the following function call format:

```
[Ak,Bk,HF,Fd,hn,n]=show_filter_responses_pz(poles,zeros,K,fsample,num_of_f_points,
                                            num_of_n_points,figure_num);
```

where the arguments are:

*poles* = a list of the Z plane locations (complex values) for all the POLES of the filter (a row vector)
*zeros* = a list of the Z plane locations (complex values) for all the ZEROS of the filter (a row vector)
*K* = Multiplier constant for the transfer function (which you should multiply H(z) by)
*fsample* = sampling frequency (samples / second)
*num_of_f_points* = the # of points for the freq. response plot
*num_of_n_points* = the # of points for the unit sample response plot
*figure_num* = number of the 1st figure to use for plots

and the function returns the following information:

Ak = a list of the $A_k$ coefficients of the filter difference equation (coefficients of the "y" terms)
Bk = a list of the $B_k$ coefficients of the filter difference equation (coefficients of the "x" terms)
HF = the DTFT frequency response values (linear scale)
Fd = digital frequencies that match the freq response values
hn – has the unit sample response sequence values
n – has the corresponding sample indices (0 to [*num_of_n_points* – 1]);

The user of this function would set up the "poles" and "zeros" arrays for their design, and then use this function to analyze the design. This function should perform the following:

1. Convert the poles and zeros into the numerator & denominator polynomials of the transfer func. H(z).
2. Scale the transfer function by the multiplier "K".
3. Print the determined values of the filter coefficients (Ak's and Bk's) to the screen.

This function should then create the same plots as the `show_filter_responses()` function (above).

The Matlab Help information provided for the following functions may prove useful:
`zplane, poly, freqz, transpose, figure`

Some hints for helping you to use this program:

- Your "poles" and "zeros" arrays will contain complex numbers. You can enter these in Matlab using either: (real) +j*(imaginary) format:
  Example: zeros = [ 0.5 + j*0.5 , 0.5 – j*0.5];
  OR using a "phasor" type form: magnitude*exp(j*angle) where angle is in radians
  Example: zeros = [ 0.707*exp(j*pi/4), 0.707*exp(-j*pi/4) ];

Test your program by finding the frequency response (500 points) using a filter with the difference equation:

$$y[n] = x[n] + x[n-2];$$

Assume a sampling rate of 1 KHz.

**Turn in the following:**

- What were your Ak and Bk values?
  - Ones you determined from the difference equation
  - Ones your program returned
- What is H(z) for this filter?
- What is your 'K' value? (Show how you determined it).

- What were the values you used for your poles and zeros?
    - Show how you derived them
- A printout of the m-file for your function
- A printout of the figures produced for the test filter design given above.

## 4) [Matlab] Additional Analysis of the Responses of a Digital Filter

Using Matlab, revise both of your two filter analysis programs (from the previous sections) to include the following new features:

- In addition to all the frequency response plots already created, your functions should also perform several numeric evaluations of the DTFT frequency response, including:
    - Find the **Peak magnitude response value**, and the **digital frequency** at which it occurs
    - Find the **Minimum magnitude response value,** and its **digital frequency**
    - **Maximum attenuation** of the filter (dB difference between max and min responses)
    - **Magnitude response at the 3 dB cutoff frequency** (linear magnitude value that is 3 dB below the peak magnitude value).

- Based on the frequency response magnitude shape (and particularly the magnitudes at DC, the Nyquist Frequency, the peak magnitude response, and the minimum magnitude response values), determine the following, and report them in the Matlab Command Window:
    - **Determine the type of filter**: *i.e.,* low-pass, high-pass, band-pass or band-stop (notch) filter. [You will need to learn to use Matlab's version of an "if-then-elseif-else" statement, and logical operations such as and() and or() for creating multiple factor conditional statements.]

    - **Determine the 3 dB cutoff frequency** (or frequencies) $F_{3dB}$ – approximate digital frequencies where the magnitude response crosses through the level that is 3 dB below the peak magnitude response value. Determine the value that is appropriate for the type of filter that was determined above. For example, for a low-pass filter, the proper 3dB frequency to report is the frequency at which the magnitude response *first* drops more than 3 dB below the peak response. For a high-pass filter, you want to report the last frequency that has a magnitude response that is more than 3 dB below the peak value (or the 1st frequency with a response that is less than 3 dB below the peak value). With Matlab, these can most easily be found using the "find()" function. This function returns all of the indices for the elements of an array that meet a certain criteria. For example, if you have an array of values called "A", and you want to find which elements in A have values greater than 1, you could use:

```
Agt1_indices = find( A > 1 ); % find the array indices for values in A that are >1
Big_A_values = A(Agt1_indices); % get all the actual values of A that are >1
First_Big_A_Value = A(Agt1_indices(1)); % get just the 1st value of A that is >1
Last_Big_A_Value = A(Agt1_indices(length(Agt1_indices)));
    % get only the last value of A that is >1
            % "length(x)" returns the number of element in the array "x"
            % "x(length(x))" is the last value in the array "x"
```

    - Determine the (one-sided) **3dB bandwidth** of the filter (as appropriate for the filter type).
      For example, for the low-pass filter, the 3 dB bandwidth would be from F=0 to $F_{3dB}$.

The results of each of these computations should be displayed in the Matlab command window when you run your analysis programs. You can refer to the instructor-guided frequency response plotting m-file created in Lab #1 for help with some of these computations.

**Test your program** by analyzing filters with the following difference equations:

**a) y[n] = x[n] - 0.9 y[n-1]**
**b) y[n] = 0.5 x[n] + 0.5 x[n-2]**
**c) y[n] = 0.5 x[n] + 0.5 x[n-1]**

and one with the following pole/zero locations:

**d) Poles at z=[0.7*j -0.7*j], Zeros at z=[-1 1], K=1**

- Test your program using 50 points for h[n], and 500 points for the frequency responses; and assuming a sampling frequency $f_{sample}$=1 KHz.
- Verify that the filter response metrics determined by your program were correct, by checking the computed values against the frequency response plots created by your program. You may find the zooming and Data Cursor tool features on the Matlab figures helpful for this. For example, verify the peak magnitude response value using the Data Cursor to follow the plot to its peak value. Confirm that the magnitude response is at the correct level to be 3 dB below the peak response at the determined 3dB frequency. *Etc.*

**Turn in the following:**

- A printout of the m-files for your functions.
- The results of the peak response, minimum response, filter type, and all the other new metric values that were computed by your program for each of the 4 test cases.
- A printout of the DTFT digital frequency response plot figures created by your function for each of the test filter cases given above. Use the Matlab figure tools (such as zooming and the Data Cursor) to make customized versions of your plots that show in detail the parts of the frequency response plot that confirm your computed metric results. Place multiple plots on each page of your report to save paper.