EE 419 - Project 2

**Developing a Filter Analysis Program in Matlab**

|  |  |
| --- | --- |
| **Names: Chris Adams, Aiku Shintani** | **Lab Date: 1/15/19** |
| **Bench #: 9** | **Section: 2** |

**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**

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

**[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);

**Function Specifications:** Your Matlab function should

* Create a unit sample sequence using the function unit\_sample( ) 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 Case:** y[n] + 0.8 y[n-1] = x[n]

**Test Results:**

****

**Figure 1: Unit sample response using MATLAB code below and test case in part 1**

**Matlab Code:**

% This function takes 4 inputs: the Bk and Ak coefficients for a

% difference equation, the number\_of\_samples (n), and figure #.

% The function returns the unit sample response as a figure.

function [hn, n] = unit\_sample\_response(Bk, Ak, number\_of\_samples, figure\_number)

[dn, n] = unit\_sample(number\_of\_samples); %get arrays of sample index

hn = filter(Bk, Ak, dn); %filter the Z domain tf to the time domain

figure(figure\_number)

%make stem plot of unit sample response

stem(n, hn, '.', 'MarkerSize', 20, 'Linewidth', 2);

xlabel('Sample index')

ylabel('Unit Sample Response')

title('Unit Sample Response')

end

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

**[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 Ak coefficients of the filter difference equation (coefficients of the “y” terms)

*Bk* = a list of the Bk 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 1st 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]);

This function should create the following system response plots:

* A “Pole/Zero” diagram
* Two Frequency Response plots of H(F) vs F

1. H(F) vs F (digital frequency): Using a digital frequency axis.
2. H(F) in dB vs *f*analog

* A Unit Sample Response Plot h[*n*] vs *n*

**Test Case:** y[n] + 0.1 y[n-1] = x[n]+ x[n-2] Assume a sampling rate of 1 KHz.

**Analysis:**

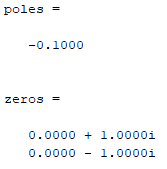
* What are the Filter Difference Equation Coefficient Ak and Bk values?

**Ak = [1, 0.1 ] Bk = [ 1, 0, 1 ]**

* What is the Transfer Function *H*(*z*) for this filter?



* What are the values of the Poles and Zeros of this filter?
  + Computed by hand, factoring H(z): **Poles @ z= [ 0, -0.1 ] Zeros @ z= [ j, -j ]**
  + Computed by your Matlab function:



**Test Results:**

****

**Figure 2a: Pole zero plot using test case in part 2**

****

**Figure 2b: Frequency response (decimal magnitude) plot using test case in part 2**

****

**Figure 2c: Frequency response (dB magnitude) plot using test case in part 2**

****

**Figure 2d: Unit sample response (first 20 points) using test case in part 2**

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

**[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 Ak coefficients of the filter difference equation (coefficients of the “y” terms)

Bk = a list of the Bk 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

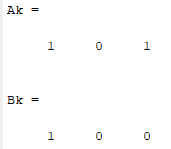
hn – has the unit sample response sequence values

n – has the corresponding sample indices (0 to [*num\_of\_n\_points* – 1]);

**Test Case:** y[n] = x[n] + x[n-2] Assume a sampling rate of 1 KHz.

**Analysis:**

* What are the Filter Difference Equation Coefficient Ak and Bk values?
  + From the difference equation: **Ak = [ 1, 0, 0 ] Bk = [ 1, 0, 1 ]**
  + Computed by your Matlab function:



* What is the Transfer Function *H*(*z*) for this filter?



* What is the overall Transfer Function Gain (‘K’) value? **K =** 1
* What are the values of the Poles and Zeros of this filter?
  + Computed by hand, factoring H(z): **Poles @ z= [ 0, 0 ] Zeros @ z= [j, -j ]**

**Test Results:**

****

**Figure 3a: Pole zero plot using test case in part 3**

****

**Figure 3b: Frequency response (decimal magnitude) plot using test case in part 3**



**Figure 3c: Frequency response (dB magnitude) plot using test case in part 3**



**Figure 3d: Unit sample response (first 20 points) using test case in part 3**

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

**Filter Analysis Features**

* + 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).
  + **Determine the type of filter**: *i.e.,* low-pass, high-pass, band-pass or band-stop (notch) filter.
  + **Determine the 3 dB cutoff frequency** (or frequencies) F3dB
  + Determine the (one-sided) **3dB bandwidth** of the filter (as appropriate for the filter type).

**Test Cases**:

**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]**

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

**Summarized Test Results:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Filter** | **Type** | **Peak**  **Response** | **Peak**  **Freq**  **(cyc/samp)** | **Min**  **Response** | **Min**  **Freq**  **(cyc/samp)** | **Max**  **Atten.**  **(dB)** | **3dB**  **Mag** | **3dB**  **Freq**  **(cyc/samp)** | **3dB**  **BW**  **(cyc/samp)** |
| **a** | **HPF** | **9.982** | **0.5** | **0.5263** | **0** | **25.56** | **7.026** | **0.484** | **0.016** |
| **b** | **Notch** | **1** | **0** | **0** | **0.2505** | **Inf** | **0.7079** | **0.1253, 0.3758** | **0.2505** |
| **c** | **LPF** | **1** | **0** | **0.0031** | **0.5** | **50.06** | **0.7079** | **0.2495** | **0.2505** |
| **d** | **BPF** | **3.922** | **0.2505** | **0** | **0** | **Inf** | **2.776** | **0.1984, 0.3026** | **0.1042** |

**Individual Test Case Results:**

**Test Case** **a: y[n] = x[n] - 0.9 y[n-1]**

**

**Figure 4a: Frequency response plot (decimal magnitude) showing min and max of test case a**

**Max attenuation:** 20log(9.82) - 20log(0.5263) = 25.56 dB

****

**Figure 4b: Frequency response plot (decimal magnitude) showing 3dB cutoff and BW of test case a**

**BW:** 0.5-0.484 = 0.016 cycles/samples

**Test Case** **b: y[n] = 0.5 x[n] + 0.5 x[n-2]**

**

**Figure 4c: Frequency response plot (decimal magnitude) showing max/min, 3dB cutoff and BW of test case b**

**Maximum attenuation:** infinite

**BW:** 0.3758-0.1253 = 0.2505 cycles/samples

**Test Case** **c: y[n] = 0.5 x[n] + 0.5 x[n-1]**

**

**Figure 4d: Frequency response plot (decimal magnitude) showing max/min, 3dB cutoff and BW of test case c**

**Maximum attenuation:** 20log(1/0.003142) = 50.06 dB

**BW:** 0.2505 – 0 = 0.2505 cycles/samples

**Test Case** **d: Poles at z=[0.7\*j -0.7\*j], Zeros at z=[-1 1], K=1**

**

**Figure 4e: Frequency response plot (decimal magnitude) showing max/min, 3dB cutoff and BW of test case d**

**Maximum attenuation:** 20log(3.922/0) = infinite

**BW:** 0.3036 – 0.1974 = 0.1042 cycles/samples

**Final (Complete) Matlab Code for Each Function:**

function [poles, zeros, HF, Fd, hn, n] = show\_filter\_response(Ak, Bk, fsample, num\_of\_f\_points, num\_of\_n\_points, figure\_num)

% This function takes multiple arguments: the Ak and Bk coefficients of the

% difference equation, the sampling frequency, the number of f and n

% points for which to plot, and the figure number. The function returns

% a pole zero plot, digital and analog frequency responses, and a unit

% sample response.

%create pole/zero diagram

figure(figure\_num)

poles = roots(Ak);

zeros = roots(Bk);

zplane(Bk, Ak);

grid

title('Pole Zero Diagram')

%plot the frequency response

HF = freqz(Bk, Ak, num\_of\_f\_points);

Fd = linspace(0, 0.5, num\_of\_f\_points);

plot\_freq\_responses(Fd, HF, fsample, figure\_num + 1);

%obtain critical values from the frequency response

HF\_mag = abs(HF);

Fd\_max\_index = find(HF\_mag == max(HF\_mag), 1, 'first');

Fd\_max = Fd(Fd\_max\_index)

HF\_max = HF\_mag(Fd\_max\_index)

Fd\_min\_index = find(HF\_mag == min(HF\_mag), 1, 'first');

Fd\_min = Fd(Fd\_min\_index)

HF\_min = HF\_mag(Fd\_min\_index)

%Find maximum attenuation (in dB)

max\_atten = 20\*log10(HF\_max/HF\_min)

%Determine the 3dB cut off frequency

cut\_off\_dB = 20\*log10(HF\_max) - 3

cut\_off\_magn = 10^(cut\_off\_dB/20)

%Determine filter type

if (HF\_mag(1) < (cut\_off\_magn) && HF\_mag(500) > (cut\_off\_magn))

    disp('Filter type = HPF')

    Filter = 1;

elseif (HF\_mag(1) > (cut\_off\_magn) && HF\_mag(500) < (cut\_off\_magn))

    disp('Filter type = LPF')

    Filter = 2;

elseif (HF\_mag(1) > (cut\_off\_magn) && HF\_mag(500) > (cut\_off\_magn))

    disp('Filter type = Notch Filter')

    Filter = 3;

elseif (HF\_mag(1) < (cut\_off\_magn) && HF\_mag(500) < (cut\_off\_magn))

    disp('Filter type = BPF')

    Filter = 4;

end

if (Filter == 1)

    hp\_cutoff = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'first'));

    disp('Cut off frequency:')

    disp(hp\_cutoff)

    disp('BW:')

    disp(0.5-hp\_cutoff)

elseif (Filter == 2)

    lp\_cutoff = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'last'));

    disp('Cut off frequency:')

    disp(lp\_cutoff)

    disp('BW:')

    disp(lp\_cutoff)

elseif (Filter == 3)

    notch\_cutoff\_low = Fd(find( HF\_mag < (cut\_off\_magn) , 1 , 'first'));

    notch\_cutoff\_high = Fd(find( HF\_mag < (cut\_off\_magn) , 1 , 'last'));

    disp('Cut off frequencies:')

    disp(notch\_cutoff\_low)

    disp(notch\_cutoff\_high)

    disp('BW:')

    disp(notch\_cutoff\_high - notch\_cutoff\_low)

else

    bp\_cutoff\_low = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'first'));

    bp\_cutoff\_high = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'last'));

    disp('Cut off frequencies:')

    disp(bp\_cutoff\_low)

    disp(bp\_cutoff\_high)

    disp('BW:')

    disp(bp\_cutoff\_high - bp\_cutoff\_low)

end

%plot the unit sample response

[hn, n] = unit\_sample\_response(Bk, Ak, num\_of\_n\_points, figure\_num + 3);

poles

zeros

end

function [Ak,Bk,HF,Fd,hn,n]= show\_filter\_responses\_pz(poles,zeros,K,fsample,num\_of\_f\_points, num\_of\_n\_points,figure\_num)

% This function takes multiple arguments: the poles and zeros of the

% system, the DC gain (K), the sampling frequency, the number of f and n

% points for which to plot, and the figure number. The function returns

% a pole zero plot, digital and analog frequency responses, and a unit

% sample response.

%create tf (in Z domain) from poles and zeros

numerator = poly(zeros);

denominator = poly(poles);

HZ = K\*filt(numerator, denominator)

Ak = denominator

Bk = K\*numerator

%create pole/zero diagram

figure(figure\_num)

zplane(Bk, Ak);

grid

title('Pole Zero Diagram')

%plot the frequency response

HF = freqz(Bk, Ak, num\_of\_f\_points);

Fd = linspace(0, 0.5, num\_of\_f\_points);

plot\_freq\_responses(Fd, HF, fsample, figure\_num + 1);

%obtain critical values from the frequency response

HF\_mag = abs(HF);

Fd\_max\_index = find(HF\_mag == max(HF\_mag), 1, 'first');

Fd\_max = Fd(Fd\_max\_index)

HF\_max = HF\_mag(Fd\_max\_index)

Fd\_min\_index = find(HF\_mag == min(HF\_mag), 1, 'first');

Fd\_min = Fd(Fd\_min\_index)

HF\_min = HF\_mag(Fd\_min\_index)

%Find maximum attenuation (in dB)

max\_atten = 20\*log10(HF\_max/HF\_min)

%Determine the 3dB cut off frequency

cut\_off\_dB = 20\*log10(HF\_max) - 3

cut\_off\_magn = 10^(cut\_off\_dB/20)

%Determine filter type

if (HF\_mag(1) < (cut\_off\_magn) && HF\_mag(500) > (cut\_off\_magn))

    disp('Filter type = HPF')

    Filter = 1;

elseif (HF\_mag(1) > (cut\_off\_magn) && HF\_mag(500) < (cut\_off\_magn))

    disp('Filter type = LPF')

    Filter = 2;

elseif (HF\_mag(1) > (cut\_off\_magn) && HF\_mag(500) > (cut\_off\_magn))

    disp('Filter type = Notch Filter')

    Filter = 3;

elseif (HF\_mag(1) < (cut\_off\_magn) && HF\_mag(500) < (cut\_off\_magn))

    disp('Filter type = BPF')

    Filter = 4;

end

if (Filter == 1)

    hp\_cutoff = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'first'));

    disp('Cut off frequency:')

    disp(hp\_cutoff)

    disp('BW:')

    disp(0.5-hp\_cutoff)

elseif (Filter == 2)

    lp\_cutoff = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'last'));

    disp('Cut off frequency:')

    disp(lp\_cutoff)

    disp('BW:')

    disp(lp\_cutoff)

elseif (Filter == 3)

    notch\_cutoff\_low = Fd(find( HF\_mag < (cut\_off\_magn) , 1 , 'first'));

    notch\_cutoff\_high = Fd(find( HF\_mag < (cut\_off\_magn) , 1 , 'last'));

    disp('Cut off frequencies:')

    disp(notch\_cutoff\_low)

    disp(notch\_cutoff\_high)

    disp('BW:')

    disp(notch\_cutoff\_high - notch\_cutoff\_low)

else

    bp\_cutoff\_low = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'first'));

    bp\_cutoff\_high = Fd(find( HF\_mag > (cut\_off\_magn) , 1 , 'last'));

    disp('Cut off frequencies:')

    disp(bp\_cutoff\_low)

    disp(bp\_cutoff\_high)

    disp('BW:')

    disp(bp\_cutoff\_high - bp\_cutoff\_low)

end

%plot the unit sample response

[hn, n] = unit\_sample\_response(Bk, Ak, num\_of\_n\_points, figure\_num + 3);

end

**5) [Matlab] Using Matlab’s Filter Analysis Tool**

fvtool(B,A) launches the Filter Visualization Tool and computes the

Magnitude Response for the filter defined by numerator and denominator

coefficients in vectors B and A.

**Test Case:**  **y[n] = ¼ x[n] - ¼ x[n-2] + ¼ y[n-1] – ½ y[n-2]**

**Test Results** (plots from the fvtool):

* Frequency Response Magnitude and Phase Responses (on single plot)

**

* Group Delay



* Step Response

**

* Pole/Zero Diagram

**