EE-3220 LABORATORY

Week 3

Difference Equations, Impulse Response Functions, and Discrete Time Filters

Goal – Investigate discrete filtering in MATLAB using difference equations. Additionally, use the Discrete Time Fourier Transform to investigate the effects of filtering in the frequency domain.

Materials - Laptop computer, speakers or headphones (recommended)

Background – The general form of a linear system represented by an Nth order difference equation is given by the following:

$$\boldsymbol{a_0 y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M) - a_1 y(n-1) - \cdots - a_N y(n-N)}$$

The system response can be found in MATLAB for a given set of a and b vectors with the following:

```
y(n) = (1/a(1)) (b(1)*x(n) + b(2)*x(n-1) + ... + b(M+1)*x(n-M)
                             - a(2)*y(n-1) - ... - a(N+1)*y(n-N));
```

1. *Explain why the a_0 term in the first equation becomes a(1) (higher index) in the second equation.*

A 6-point moving average FIR filter can be represented by the equation:

$$y(n) = \frac{1}{6}\big(x(n) + x(n-1) + \cdots + x(n-5)\big)$$

Let's apply this filter to the noisy audio clip we created in Lab 2, the 4-bit (lowres) version of the Nutcracker clip. Download the lowres_clip_file.mat file from https://faculty-web.msoe.edu/prust/EE3220/

```
load lowres_clip_file
```

and the variable **lowres_clip** will be loaded into the workspace. This is a 4 s cut of the original music clip. To remind yourself of the beauty of this noisy clip, turn down the volume to a comfortable level and play the clip:

```
sound(lowres_clip, 44100)
```

Now, pass the sound clip through the moving average filter. Use a "for" loop to do this. There is an immediate issue: how do we handle the input samples for n = 0,1,2,3, and 4? Notice that we don't have enough previous samples to fully implement the filter for these values of n! Let's take the simplest of approaches – ignore these samples and begin filtering at n = 5.

```
x = lowres_clip;        % x is filter input
y = zeros(size(x));     % y is filter output – initialize all values to zero
for ii=6:length(x)
     y(ii) = 1/6*(x(ii) + x(ii-1) + x(ii-2) + x(ii-3) + x(ii-4) + x(ii-5));
end
```

Listen to the filtered version of the clip:

```
sound(y, 44100)
```

2. *Describe the difference you hear between the lowres_clip and the filtered version of lowres_clip.*

MATLAB has a built-in function to do filtering – filter(). To use it, you must specify the “a” and “b” coefficients of your filter. The 6-point moving average FIR filter can be implemented as follows:

```
a = 1;
b = (1/6)*[1 1 1 1 1 1];
```

An easier way to create the vector b in MATLAB is

```
b = (1/6)*ones(1,6);
```

The ones(m,n) function creates a matrix of all ones, with m rows and n columns.

```
filtered_clip = filter(b, a, lowres_clip);
```

Listen to the filtered version of the clip:

```
sound(filtered_clip, 44100)
```

Examine the difference between the two methods of filtering, paying careful attention to the scale of the y axis:

```
plot(0:length(x)-1, filtered_clip-y)
```

3. *What can you conclude from the above graph of the differences?*
4. *Implement an 11-point moving average filter. Pass lowres_clip through this filter. How does the filter output compare to the 6-point moving average filter? Describe any differences you hear.*

Now, we will take the Discrete Time Fourier Transform (DTFT) of **lowres_clip** and the 11-point moving averaged **filtered_clip** to see the difference in frequency content of these sequences. Don’t use the matrix multiplication method described in the book since the sound clip is quite long and the resulting matrix is too large. MATLAB will reject operations that exceed memory limitations, or worse, try to perform the work using virtual memory which may critically degrade performance.

Instead, you can use a for loop and create a 4000-element vector with elements representing the specific frequencies:

```
f = linspace(0,22050,4000); % 22050 is half of the sampling frequency
```

5. *What is the slight difference between creating an f index with linspace(0,22050,4000) and [0:22050/4000:22050]?*

Now, create a vector of corresponding digital frequencies. The digital frequency has units of radians per sample and is related to the continuous time frequency by the sampling rate of the original signal.

```
w = f/44100*2*pi;
```

6. *What digital frequency, w, does the sample rate, 44.1 kHz map to? What digital frequency, w, does 22.05 kHz map to? The mapped values are important since they tell us which signals will alias.*

Create a vector of sequence indices for the audio clip.

```
n_clip = 0:length(lowres_clip)-1;
```

Now, for each of the 4000 frequencies, compute the DTFT value. This will take a couple of minutes. To assure yourself that the computation is progressing, you can print out some values of k in the for loop.

```
dtft_lowres = zeros(size(f)); % allocate space so MATLAB doesn’t need to
                              % resize each time through the loop
for k=1:length(f)
      if mod(k,100)==0, disp(k), end
      dtft_lowres(k) = exp(-n_clip*1i*w(k)) * lowres_clip;
end
```

The DTFT is a complex valued function consisting of a magnitude and phase component. Often the magnitude is plotted on a logarithmic scale with dB as the units (remember Bode Plots?).

```
plot(f, 20*log10(abs(dtft_lowres)))
```

Do the same with filtered_clip obtained from your 11-point moving average filter. Make a plot containing the DTFTs of lowres_clip and filtered_clip, again plot with 20*$\log_{10}$(| |) on the vertical axis. Do this with a ***single*** plot command, which has the benefit of choosing different colors for the traces. You can vertically concatenate the two DTFT row vectors in a matrix as the argument to abs():*[dtft_lowres; dtft_filtered]*.

7. *Provide a plot illustrating both DTFTs with appropriate labels in your submittal. Use legend() to appropriately label the traces. What is the difference between the DTFTs of the lowres_clip and the filtered_clip? How is this difference exemplified by what you heard in the audio?*

Now, let’s look more carefully at the filter itself. Recall that the impulse function response function is the output when the input is an impulse. Use an impulse function as the input to verify the impulse response of the 11-point moving average filter.

```
[d, n] = impseq(0, -20, 20); % OR: n=-20:20; d = n == 0;
h = filter(b, a, d);
```

8. *Provide a **stem** plot of the impulse response of the 11-point moving average filter. Completely label the graph. The vertical axis should have a label of “Impulse response: h(n)”, the horizontal axis should have a label of “Sample number: n”. Provide an appropriate title.*

9. *Is this an FIR or an IIR filter? Explain your answer based on the plot.*

The impulse response signal is special, because we can use it for convolution and we can obtain the frequency response of the filter. The spectrum (i.e., DTFT) of the impulse response is actually the frequency response of the filter. To illustrate this take the DTFT of the impulse response h, following the steps you used to take the DTFT for the audio clips. We will need to make some modifications, such as:

```
H = zeros(1,length(n)); % allocate space
f2 = linspace(0, 22050, length(n));
w = f2/44100*2*pi;
for k = 1:length(H)
      H(k) = exp(n*1i*w(k)) * h';
end
```

10. *Why must we use **h’** and not **h** to find **H(k)** in the above example?*

11. *Plot* ***20*log$_{10}$(|H|)*** *vs.* ***f2*** *and print the resulting graph. Use the linestyle 'b-o' for clarity. Roughly speaking, what type of filter is this (lowpass, highpass)?*

12. *Plot 20*log$_{10}$(| |) for dtft_lowres, dtft_filtered, and H on the same graph with appropriate labels. You can use different horizontal series in a single call to plot, e.g.,* `plot(f1,y,f2,h)`. *You can even set a different line style and color for each series, e.g.,* `plot(f1,y(1,:),'b-', f1,y(2,:),'g--', f2,h,'r.-')` *What observations and conclusions can you make from these 3 series?*

Consider a simple 1st order system characterized by the difference equation,

$$y(n) = x(n) + 0.5y(n-1)$$

13. *Write MATLAB code that implements the difference equation to calculate the impulse response of the filter. Do not use the "filter()" function, but rather implement the filter using a for loop. Include your MATLAB code. Provide a plot of the impulse response. Completely label the graph. The vertical axis should have a label of h(n), the horizontal axis should have a label of n. Provide an appropriate title. Include this with your submittal.*

14. *Is this and FIR or an IIR filter? Explain your answer based on the plot stem(n,h).*

Finally, we illustrate that the output of a discrete system is not always stable. As an example, compare the **step response** between two slightly different 1st order systems.

$$y1(n) = x(n) - 0.95y1(n-1)$$

and

$$y2(n) = x(n) - 1.05y2(n-1)$$

Use the "stepseq" function from the textbook to generate the input signal

```
[u,n] = stepseq(0, -20, 50); % OR n=-20:50; u = n >= 0;
```

Create the **a** and **b** vectors for these systems and use "filter()" to determine their step response.

15. *Provide stem plots of the unit step response for both systems. Use the subplot function to illustrate these responses. What can you conclude?*

Submittal

The submittal should follow all instructions on the provided grading checklist and cover sheet.

The standard summary section is not required this week.

Additional Problems

Submit answers to the following additional problems with your lab. Be sure to include any MATLAB code.

- Take a break from additional problems this week.