Assignment 4: Working with FIR filters

TA: Stefania Serafin
CCRMA[*]
serafin@ccrma.stanford.edu

1 Filtering of Sampled Waveforms

(from DSP first)

A discrete-time linear time-invariant system is described by the formula

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{1}$$

This equation gives a formula from which to compute the nth value of the output sequence from values of the input sequence. This system is called an FIR filter. Eq. (1) is implemented by the following MATLAB statement:

```
yy = filter(b, 1,xx);
```

where it is assumed that `xx` is a vector of input samples and the vector `b` contains the b_k coefficients of Eq. (1) stored in the following way: `[b0, b1, b2, ... , bM]`.

In the experiments of this lab, you will use `filter( )` to implement filters and begin to understand how the filter's frequency response relates to the action of the filter for smoothing or sharpening. Thus we can characterize how a filter reacts to different frequency components in the input.

The frequency response of a general FIR linear time-invariant system is

$$H(\hat{\omega}) = \sum_{k=0}^{M} b_k e^{-j\hat{\omega}k} \tag{2}$$

We have seen that MATLAB has a built-in function for computing the frequency response of a discrete-time LTI system. It is called `freqz( )`. To reillustrate its use, the following MATLAB statements would compute and plot the magnitude (absolute value) of the frequency response of a first difference system as a function of $-\pi \le \hat{\omega} \le \pi$:

```
b = [1, -1];          %-- Filter Coefficients
omega = -pi:(pi/100):pi;
B = freqz(b, 1, omega);
plot(omega, abs(B))
```

For filters of the form of Eq. (1), the second argument of `freqz( , 1, )` must always be `1`.

2 Loading data in MATLAB

From the dspfirst CDROM, lab 6, find the file lab6dat.mat. Load it in MATLAB as follows:

```
load lab6mat.mat
```

In this way you will load in MATLAB the variables you will need for this assignment.

[*]http://www-ccrma.stanford.edu

3 Filtering a Stair-Step Signal

In this experiment we are going to investigate the two systems shown in Figs. 1 and 2. In these two systems, the system called "First Difference" is defined by the difference equation

$$y[n] = x[n] - x[n-1] \tag{3}$$

and the system called "5-Point Averager" is defined by the equation:

$$y[n] = \frac{1}{5}\sum_{k=0}^{4} x[n-k] \tag{4}$$

The first test signal used will be a stair-step signal in which the signal is flat for different intervals, but the value of the signal in each flat region is different.

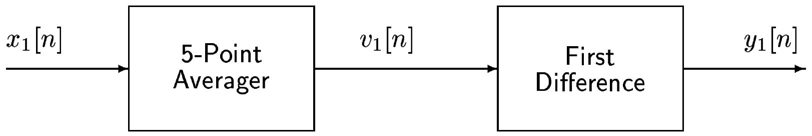

Figure 1: First cascade system: averaging operator followed by difference.

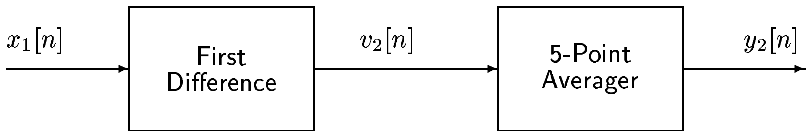

Figure 2: Second cascade system: differencing operator followed by average.

3.1 Implementation of 5-Point Averager

Use the MATLAB function `filter( )` to implement the 5-Point Averager, i.e., compute `v1`.

(a) Plot `x1` and `v1` in the same figure using a two-panel `subplot` . The signals have different lengths, but the plot should start at the same index, i.e., $n = 0$.

Give a qualitative description of how the 5-Point Averager system changed the input signal. Estimate the time shift between the input and output signals. Express this delay in number of samples.

(b) Use `freqz( )` to compute the frequency response of the 5-Point Averager and plot its magnitude as a function of frequency for $-\pi \le \hat{\omega} \le \pi$. From the shape of the frequency response curve, determine which frequency region is "passed" by the filter? Which frequencies are rejected? Relate the frequency response to the qualitative description of the time-domain response.

(c) Plot the phase response versus frequency. Measure the slope of the phase response and compare this slope to the time shift between the input and output signals.

3.2 Implementation of First Difference System

(a) Plot `x1` and `v2` in the same figure using a two-panel `subplot`.

(b) Describe qualitatively how the First Difference system changed the input signal? Where are the peaks of `v2`? What is the nature of the input `x1` at the places where the peaks of `v2` occur?

(c) Use `freqz( )` to compute the frequency response of the First Difference and plot its magnitude as a function of frequency for $-\pi \leq \hat{\omega} \leq \pi$. Is the shape of the frequency response curve consistent with your interpretation of what the system did to the input signal? Is the first difference a high-pass or low-pass filter?

3.3 Implementation of Figure 1

(a) Use the MATLAB function `filter( )` to implement the overall system of Fig. 1 by first computing the output of the 5-Point Averager, `v1`, and then using `v1` as input to the First Difference system in Fig. 1 to compute the output `y1`.

(b) Use `freqz( )` to compute the frequency response of the cascaded system and plot its magnitude and phase as a function of frequency for $-\pi \leq \hat{\omega} \leq \pi$.

3.4 Implementation of Figure 2

(a) Use the MATLAB function `filter( )` to implement the overall system of Fig. 2 by first computing the output of the First Difference, `v2`, and then using `v2` as input to the 5-Point Averager system in Fig. 2 to compute the output `y2`.

(b) Use `freqz( )` to compute the frequency response of the cascaded system and plot its magnitude and phase as a function of frequency for $-\pi \leq \hat{\omega} \leq \pi$. Compare to the overall frequency response computed for Fig. 1.

3.5 Comparison of Systems of Figures 1 and 2

Execute the MATLAB statement `sum( (y1-y2).*(y1-y2) )`. Discuss the implications of the result. Hint: write the mathematical expression that is being evaluated by MATLAB.

4 Filtering the Speech Waveform

The sampled speech waveform is stored in the variable `x2`. Two sets of filter coefficients are stored in `h1` and `h2` (i.e., these are the "b_ks" for two different filters). Use `length` to find out how many filter coefficients are contained in `h1` and `h2`. In this experiment we will test these filters on the speech signal.

(a) Filter the speech signal with filter `h1` using the statements

```
y1 = filter(h1,1, x2);
inout(x2, y1, 3000, 1000, 3)
```

The M-file `inout( )` will plot two very long signals together on the same plot. You can find it in your dspfirst CD-ROM. If you have troubles finding it let me know. It formats the plot so that the input signal occupies the first, third, and fifth lines, etc. while the output is on the second, fourth, and sixth lines etc. Type `help inout` to find out more.

Compare the input and output signals. Is the output "rougher" or "smoother" than the input signal?

(b) Use `freqz( )` to plot the frequency response of the system defined by the coefficients `h1` as a function of frequency for $-\pi \leq \hat{\omega} \leq \pi$. What kind of filter is `h1` ?

(c) Since the vector of filter coefficients is rather long, a `stem` plot of `h1` can be informative to show the nature of the filter. Do you notice a symmetry in the coefficients?

(d) Filter the speech signal with filter h2 and plot the input and output using the statements

```
y2 = filter(h2,1,x2);
inout(x2, y2, 3000, 1000, 3)
```

Compare the input and output and state whether the output is "rougher" or "smoother" than the input signal.

(e) Use freqz( ) to plot the frequency response of the system defined by the coefficients h2 as a function of frequency for $-\pi \leq \hat{\omega} \leq \pi$. What kind of filter is h2 ?

(f) Make a stem plot of h2 and look for a symmetry in the coefficients.

(g) Make a listening comparison by executing the following statements

```
sound([x2; y1; y2], 8000)
```

Comment on your perception of the filtered outputs versus the original

5 Audio effects in MATLAB

1. Write a matlab function that creates a flanging effect. You function should look like:

```
[y]=flanger(input,delay);
%
% input = input soundfile
% y     = output with flanger effect applied
% delay = length of the delay
```

2. Choose a snippet of a song you like.

3. Apply to it the flanging effect.

3. Provide the Matlab code you implemented together with the snippet before and after applying the effects.

   The code should contain an instruction to play the modified soundfile.

4. Now build an equalizer to adjust the levels of your snippet.

   Create a function in MATLAB that does that.

```
[y]=equalizer(input,f1,f2,g1,g2,g3);

% input = soundfile output of the flanger function.
% y     = resulting equalized soundfile.
% f1    = cutoff frequency for the lowpass filter
% [f1,f2] = frequency band for the bandpass filter.
% f2      = lowest frequency for the highpass filter
% g1     = gain for the lowpass filter.
% g2     = gain for the bandpass filter.
% g3     = gain for the highpass filter.
%
% Uses FIR1 to implement an equalizer
```

You can choose arbitrary values of g1, g2 and g3 and of f1 and f2.