



In this lab, you will get a preview of various filtering commands in matlab.

**PRELAB:**

Recall that any discrete system can be represented in time domain either through its impulse response, or through its constant coefficient linear difference equation (CCLDE), both of which are identical representations. In Matlab, most filtering functions use the CCLDE representation. Note that CCLDE represent the system as weighted sums of current and former inputs and outputs.

$$\sum_{i=0}^N a_i y[n-i] = \sum_{j=0}^M b_j x[n-j], \quad a_0 = 1$$

or equivalently, by pulling  $y[n]$  out of the equation,

$$y[n] = \frac{1}{a_0} \left\{ \sum_{j=0}^M b_j x[n-j] - \sum_{i=1}^N a_i y[n-i] \right\}$$

where  $a_i$  ( $i \neq 0$ ) are the coefficients of the previous output,  $a_0 = 1$  is the coefficient of the current output, and  $b_j$  are the coefficients of the inputs. Note that the above description makes it obvious why  $a_0 = 1$ : the entire equation can be divided by  $a_0$  to effectively make it “1”. Also remember that if all  $a_i = 0$  (except  $a_0$ , of course), then the system becomes an FIR system, otherwise it is an IIR. For FIR systems, the  $b_j$  coefficients represent the impulse response of the system as well.

For reasons that will become obvious when we cover transfer functions,  $a_i$  coefficients are also known as denominator coefficients, whereas the  $b_j$  are the numerator coefficients.

In Matlab, we will use the following filtering commands quite often. A summary statement is given below, but check the Matlab documentation by typing **doc** <command> to find all options available to you with each command

**Y=filter(b,a,x)**; Filters the signal in **x** using the discrete filter characterized by the CCLDE coefficients **a** and **b**

**h=impz(b,a,N)**; Computes **N** uniformly sampled samples of the impulse response of the system that is characterized by the CCLDE coefficients **a** and **b**

**[H,f]=freqz(b,a,N,Fs)**; Plots the frequency response of the discrete filter that is characterized by the CCLDE coefficients **a** and **b**. **Fs** is the sampling frequency of the signal to be filtered, **H** is the complex

frequency response computed over  $N$  points, and  $f$  is a frequency base computed based on the sampling frequency. If  $F_s$  is not used, the output frequency base will be in angular frequency, not in Hz.

### LAB & QUESTIONS

A digital filter is described by

$$y[n] - 2.56y[n-1] + 2.22y[n-2] - 0.65y[n-3] = x[n] + x[n-3]$$

Assume all zero initial conditions, that is  $y[n]=0$  for  $n \leq 0$

1. Generate an input signal  $x(n)$ , as a sinusoid of frequency 500 Hz sampled at 6kHz.
2. Compute the first four cycles of the output by directly implementing the above difference equation. Plot the input and output on the same graph, using the **hold** command.
3. Implement this filter by using Matlab's **filter** function. How does your response compare to what you obtained in part (2).
4. Plot the impulse response of the filter by using Matlab's **impz** function.
5. Based on its impulse response, what kind of a filter is this? Why?
6. What happens if we truncate this impulse response down to 50 points, that is, what type of filter does it then become? Note that the truncated filter is now just an approximation of the original filter. Why would we want to use such an approximation? How about down to 32 points?
7. Let the truncated filter's impulse response be represented by  $h_{50}[n]$  and  $h_{32}[n]$ . Find the output of these filters to the same input  $x[n]$  you generated in part (1). Compare your result with (2) or (3). Are they similar? Why/ why not / how?
8. Generate a new input signal  $x[n]$ , as a summation of two sinusoids with frequencies 500 Hz, and 1500 Hz sampled at 6 kHz. Repeat (3). Discuss your results.
9. Obtain and plot the magnitude frequency response of the original as well as the truncated filters.
10. Comment on the results of (2) and (6), based on the frequency response of the filter.
11. Generate an audio signal and/or import one into Matlab using **wavread** command. Apply your filter to this signal and save the output to another wave file using the **wavwrite** command. Listen to your filtered signal. What – if anything – has changed? Comment on your results.
12. Implement the CCLDE system in Simulink. Add noise to the system and see if the filter removes the noise.