Lab 7 Information

- The material in this section is informational. Please read through the section as it helps you work on the lab exercises in the next section. There may be code examples in this informational section. You are welcome to copy-and-paste them to MATLAB to run the code, but no submission is needed on any test run.

Implementing IIR Filters in MATLAB

We have been using the MATLAB function `filter` to implement FIR filters. It turns out that the same function can also be employed to implement IIR filters. Consider the general IIR filter specified by the difference equation:

$$\sum_{l=0}^{L} a_k y[n-l] = \sum_{k=0}^{M} b_k x[n-k]., \tag{1}$$

Put the feedback coefficients $a_0, a_1, \ldots, a_L$ into the $(L+1)$-dimensional vector `a` and the feedforward coefficients $b_0, b_1, \ldots, b_M$ into the $(M+1)$-dimensional vector `b` (note that neither `a` nor `b` gives the impulse response of the IIR filter). Then

```
y = filter(b,a,x);
```

implements the IIR filter on the input signal vector `x` to give the output vector `y`, assuming the initial rest condition. For example, the simple first-order IIR filter:

$$y[n] = 0.9y[n-1] + 2x[n] - x[n-1]$$

may be implemented using the following lines of MATLAB code:

```
a = [1  -0.9];
b = [2  -1];
y = filter(b,a,x);
```

Do `help filter` in MATLAB to learn more about implementing IIR filters using the function `filter`.

Lab 7 Exercises

- Unless stated otherwise, you must submit your solutions to all the lab exercises in this section.
- Your laboratory solutions should be submitted on Canvas as a single PDF. The simplest way is to put your codes and answers for all the lab exercises in a single MATLAB Publisher script and use `%%` to separate the codes and answers for different exercises into different sections as described in the information section of Lab 1.
- **Plot all magnitude spectrums in this lab in dB scale.**

Exercise 7.1:

This exercise shows you how to generate the impulse response of an IIR filter using the MATLAB function `filter`. Recall the impulse response of any LTI system is the system's output when the input is the unit impulse signal. Hence, given the feedback and feedforward tap vectors `a` and `b` of an IIR filter, we may simply do

```
h = filter(b,a,impulse);
```

to obtain the impulse response of the IIR filter in the vector `h` with the input vector `impulse` representing the unit impulse signal. Nevertheless, one must pay attention to the fact that the `filter` function generates an output vector of the same length as that of the input vector. For an IIR filter, the impulse response is of infinite length. Thus, the `filter` function may only give a truncated version of the true impulse response of the IIR filter. Intuitively, the longer the truncation, the better is the approximation. Thus, your input vector `impulse` must be long enough so that `filter` outputs a "good" enough approximation to the impulse response of an IIR filter. I'll leave it to you to determine how long the input vector `impulse` needs to be and how to generate it for each case below.

(a) For each of the IIR filters below, analytically determine an expression for the impulse response, find the transfer function and its ROC, and draw the pole-zero plot. Use the MATLAB function `filter` to generate and plot (use `stem`) the impulse response. Compare your plot with the expression that you obtain analytically. Try to correlate the location(s) of the pole(s) to the plot of your impulse response. In particular, correlate the pole location(s) with the speed of decay/growth in magnitude of the impulse response. Determine also whether the filter is stable or not.

i) $y[n] = 0.8y[n-1] + x[n]$
ii) $y[n] = -0.8y[n-1] + x[n]$
iii) $y[n] = -0.8y[n-1] + x[n-10]$
iv) $y[n] = 2y[n-1] + 2x[n]$

(b) For each of the IIR filters below, analytically find the transfer function and its ROC, and draw the pole-zero plot. Use the MATLAB function `filter` to generate and plot (use `stem`) the impulse response. Try to correlate the location(s) of the pole(s) to the plot of your impulse response. In particular, correlate the pole location(s) with the speed of decay/growth in magnitude and any oscillatory characteristics of the impulse response. Determine also whether the filter is stable or not. If the filter is stable, determine analytically the frequency response of the filter and plot its magnitude response in MATLAB. Correlate the shape of the magnitude response with the pole location(s).

i) $y[n] = 1.0\cos(0.2\pi)y[n-1] - 0.25y[n-2] + x[n]$

ii) $y[n] = 1.8\cos(0.2\pi)y[n-1] - 0.81y[n-2] + x[n]$

iii) $y[n] = 1.98\cos(0.2\pi)y[n-1] - 0.9801y[n-2] + x[n]$

Exercise 7.2: *(Notch Filter)*

This exercise shows you how to use a more selective version of a nulling filter to remove a sinusoidal interfering signal that corrupts the same audio signal in Exercise 6.2. The corrupted audio signal is provided in the WAV file `bad_wannabe.wav`. Note that the frequency of the sinusoidal interference may not be the same as that in Exercise 6.2.

(a) Load `bad_wannabe.wav` into MATLAB. Repeat Exercise 6.2 by using the same nulling FIR filter prototype discussed there. In particular, save the filtered audio signal to the WAV file `fired_wannabe.wav`. Submit this WAV file. Listen to the filtered audio, and comment on its quality.

(b) Now consider the IIR notch filter specified by the following transfer function:

$$H(z) = \frac{\left(1 - e^{j\hat{\omega}_0} z^{-1}\right)\left(1 - e^{-j\hat{\omega}_0} z^{-1}\right)}{\left(1 - \alpha e^{j\hat{\omega}_0} z^{-1}\right)\left(1 - \alpha e^{-j\hat{\omega}_0} z^{-1}\right)} \tag{2}$$

where $\hat{\omega}_0$ determines the normalized radian frequency of the notch and α is a positive real number that determines the "sharpness" of the notch. State the range of α for which the IIR filter above is stable. Choose $\hat{\omega}_0$ to be the frequency of the sinusoidal interference that you determined in (a). Draw the pole-zero plot of the IIR filter for a choice of α that makes the filter stable and for a choice of α that makes the filter unstable. Select a few values of α that make the filter stable and plot the filter's magnitude response for each choice of α. Deduce a general rule to describe the "sharpness" of the notch as a function of α.

(c) Analytically determine the difference equation for the IIR notch filter whose transfer function is given by (2). Using the difference equation, write a MATLAB function to generate the feedback vector `a` and feedforward vector `b` that allow us to implement the IIR notch filter. Your function should be in the following form:

```
[b a] = notch(w0, alpha);
```

where `w0` and `alpha` specify the values of $\hat{\omega}_0$ and α in (2), respectively.

(d) Use your `notch` function to generate `b` and `a` by choosing `alpha=0.99` and `w0` to be the normalized radian frequency of the sinusoidal interference. Apply this IIR notch filter to remove the interference in `bad_wannabe.wav`. Save the filtered signal to the WAV file `iired_wannabe.wav`. Submit this WAV file. Listen to this filtered audio, and comment on its quality w.r.t. that of the filtered audio using the FIR nulling filter in (a). Plot the magnitude spectrum of this filtered signal and compare it with that of the filtered signal obtained in (a). Use the differences in the magnitude spectrums to explain the different audio quality achieved by using the FIR nulling filter in (a) and the IIR notch filter here in removing the sinusoidal interference.