ECE 2200 Fall 2014
Lab 4: IIR Filters and Time-Varying Systems

Professor Lang Tong

November 3–14, 2014 (two weeks)

1 Introduction

In this lab, you will experiment with an infinite impulse response (IIR) filter. These filters use both feedback and feedforward to compute the output to an arbitrary input. Because of the feedback, the response in general to any input, even a single impulse, has infinite duration. Additionally, in the z-domain, the system function of an IIR system has both poles and zeros. By looking at the pole-zero map of an IIR system, we can gain intuition about the frequency response of the system by looking at the pole and zero locations relative to the unit circle. In particular, you will look at the second-order form of a notch filter which is used to selectively filter out particular frequencies.

You will also experiment with a type of filter that is not time invariant; that is, its filter coefficients are changing in time. These types of filters are important for prediction, smoothing, and estimation application. In particular, you will experiment with Kálmán filters (henceforth called the Kalman filter), also known as linear quadratic estimation. This filter tries to estimate and predict the values of a signal that is embedded in perturbations and noise.

2 Preparation for the lab

Read the lab description carefully. Consider downloading the necessary files in advance if using your personal computer. **Bring headphones since we will be processing audio again.**

3 Pre-lab

We are not going to collect and grade the pre-lab. However, if you do not do anything, the lab will take a very long time. In particular, make sure that you bring the MATLAB code that you write to the lab.

3.1 Second order notch filter.

A second order IIR system has a difference equation of the form

$$y[n] + a_1 y[n-1] + a_2 y[n-2] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] \tag{1}$$

where the a_i are the feedback coefficients and the b_i are the feedforward coefficients.
A **notch filter** is a type of second order IIR system that has the following system function:

$$H(z) = \frac{1 - 2\cos(\omega)z^{-1} + z^{-2}}{1 - 2r\cos(\omega)z^{-1} + r^2 z^{-2}} = \frac{(1 - \mathrm{e}^{j\omega}z^{-1})(1 - \mathrm{e}^{-j\omega}z^{-1})}{(1 - r\mathrm{e}^{j\omega}z^{-1})(1 - r\mathrm{e}^{-j\omega}z^{-1})} \tag{2}$$

From the factored form of the system function, it is easy to see that the system function has zeros at $z = \mathrm{e}^{\pm j\omega}$ (i.e.: on the unit circle) and poles at $z = r\mathrm{e}^{\pm j\omega}$. (Typically, for stability purposes, we limit $|r| < 1$.) The notch filter can be used to selectively filter out narrow ranges of frequency; this can be seen from its frequency response.

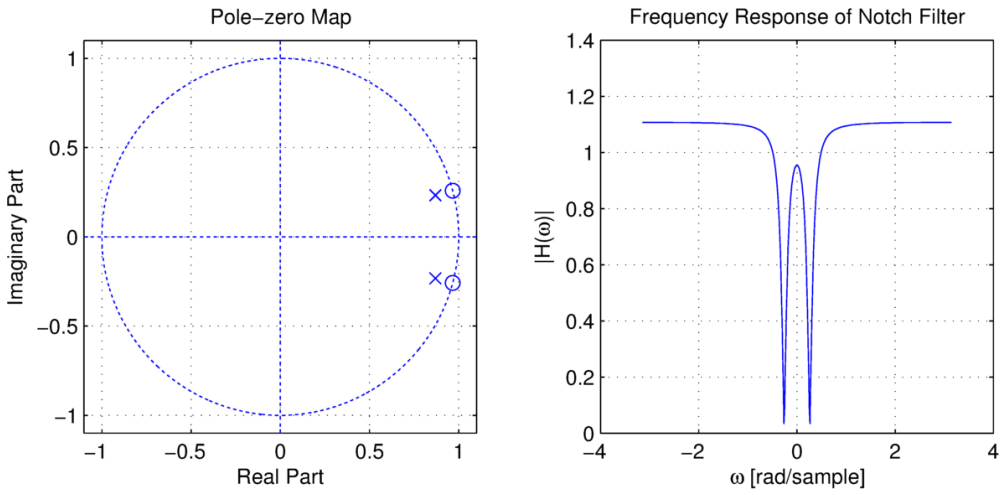


Figure 1: Pole-zero map and frequency response of a particular notch filter.

3.1.1 MATLAB implementation.

Write a function that returns the output signal $y[n]$ of a notch filter given an input signal $x[n]$ and filter parameters r and ω as described in (2). This function should also return a, the feedback coefficients and b, the feedforward coefficients. Hint: use the MATLAB function `filter` to filter signals through IIR systems.

```
function [a, b, y] = my_notch(r, w, x)
% Returns the notch filter coefficients with poles located
% at r*exp(jw) and r*exp(-jw). Also returns the output signal y
% given an input signal x filtered by this notch filter.
%
% input:
% r - radius of poles
% w - angle of poles (in radians)
% x - input signal
%
% output:
% a - vector of feedback coefficients
% b - vector of feedforward coefficients
% y - filtered signal
```

3.1.2 Pole-zero map and frequency response.

Use the MATLAB function `zplane` to plot the zeros and the poles of the notch filter for all combinations of $r = 0.3, 0.6, 0.9$ and $\omega = \pi, \pi/2, \pi/4$. Additionally, plot the frequency response for these values of r and ω for the range $-\pi$ to π. How does the frequency response change with r and ω?

Hint: use the MATLAB function `freqz` to plot frequency response. Plot the pole-zero map and the frequency response for each r and ω combination as two subplots in the same figure.

Adapted from ECE 2200 Lab 5, Fall 2011/2012 (Prof. Bojanczyk)

3.1.3 SUMMARY: Do the following in the pre-lab for this part

1. Write your MATLAB function `my_notch`.
2. Make the pole-zero maps and frequency responses for all of the requested combinations of r and ω (nine total) which you will show to the TA!
3. Be prepared to tell the TA what happens to the frequency response as r is fixed and ω varies. Likewise, what happens when ω is fixed and r varies?

3.2 Scalar Kalman filter.

3.2.1 A brief primer on random processes.

In many cases, the signals that we encounter are not deterministic; that is, we do not know ahead of time what exact values a signal will take. Sometimes, what we do know is that the signal will take on values randomly as described by a probability distribution. For example, maybe the signal at some time n is equally likely to have a value of 0 or 1. When a signal takes on values according to some probability distribution, we can call it a **random process.** A particular sequence of samples from this random process is called a **realization** of the random process.

A deeper understanding of this part of the lab requires knowing probability (ECE 3100) and random processes (ECE 4110). However, we can explore some of the powerful tools used in signal processing without having to know very much.

Gaussian random processes. We will be dealing with Gaussian random processes in this lab. For our purposes, this means that each value of the signal at each time is drawn from a Gaussian distribution. A Gaussian distribution is completely described its mean and variance. The variance, the square of the standard deviation, intuitively describes the "spread" of the distribution. Thinking about our favorite Gaussian distribution, the bell curve, the variance describes the width of the bell: larger variance means a wider bell. It also means that samples drawn from the distribution are more likely to take on values from a larger range. Typically, random noise in systems are modeled as Gaussian random processes which is often justified using the central limit theorem. (And motivated by mathematical simplicity.)

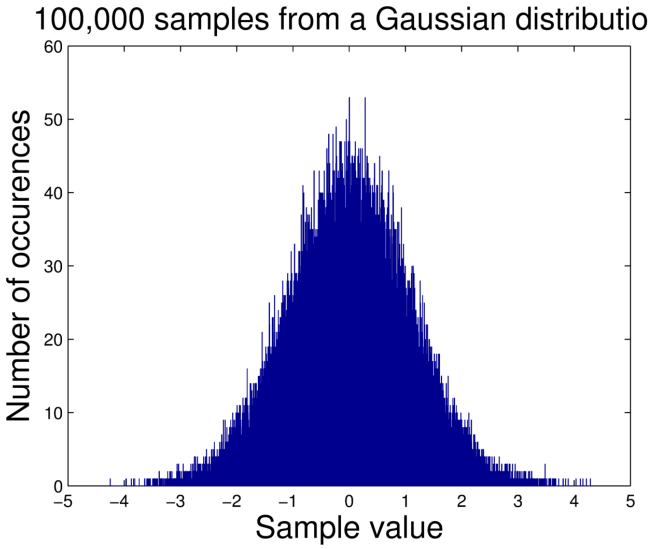


Figure 2: Illustrating a Gaussian distribution with mean 0 and variance 1. Notice that the curve is centered at 0 and that the width of the bell at half the maximum height is twice the variance.

3.2.2 A specific signal model.

Consider the class of discrete-time random processes that can be written as this recursion:

$$Z_n = aZ_{n-1} + W_n, \quad n = 1, 2, \ldots \tag{3}$$

(where the subscripts are the time index.) To illustrate this signal model, consider an example problem of tracking a moving object. Here Z_n is the position of the object at time n. Its current position is related to its previous position through some coefficient, a. Additionally, its current position may be affected by some outside perturbing force such as wind, and this is reflected by the addition of the process W_n.

Now suppose we cannot directly observe Z_n but instead observe it through the process X_n:

$$X_n = Z_n + N_n, \quad n = 0, 1, 2, \ldots \tag{4}$$

A realization of X_n contains our observations at each time n and consists of the actual signal that we want to know, Z_n, plus some observation noise, N_n. In our example, it may be that we are using radar to track the moving object but these measurements are corrupted by noise .

Our goal is to somehow process whatever observations we have in X_n and find the best estimate of Z_n. Preferably, we would like to do this in real time and do it efficiently. This implies the implementation of some kind of causal filter:

$$Y_n = \sum_{j=1}^{n} h_j^{(n-1)} X_{n-j} \tag{5}$$

where Y_n is our signal of estimates of Z_n and $h_j^{(n-1)}$ is a set of time-varying filter coefficients. How do we choose these filter coefficients?

3.2.3 Enter the Kalman filter.

It turns out that some very clever people developed an efficient, recursive algorithm to compute the "best" estimate of Z_n given all of the current and previous observations X_n. It is the best in the sense that the mean square error (MSE), or the average value of $(Z_n - Y_n)^2$ is minimized. A derivation of the results is well beyond the scope of this course but here is the basic idea:

We have the following models for a signal of interest and our noisy observations of it:

$$\begin{cases} Z_n = aZ_{n-1} + W_n, & n = 1, 2, \ldots \\ X_n = Z_n + N_n, & n = 0, 1, 2, \ldots \end{cases} \tag{6}$$

where Z_n is the hidden signal of interest, X_n is our noisy observations of Z_n, a is a scalar that describes the signal model, W_n is a Gaussian random process with zero mean and variance σ_W^2, and N_n is a Gaussian random process with zero mean and variance σ_N^2. The initial value of Z, Z_0, is drawn from a Gaussian distribution with mean μ_Z and variance σ_Z^2.

1. Start at time $n = 0$. First, set Y_0, our guess of Z_0, equal to μ_Z. In the absence of any other information, this is the best guess we can make. Additionally, set the MSE of our estimator, $\sigma^2_{e_0}$, equal to σ^2_Z.

2. Compute the Kalman gain, k_n. This factor helps us compute our next prediction of Z_n:

$$k_n = \frac{a\sigma^2_{e_n}}{\sigma^2_{e_n} + \sigma^2_N} \tag{7}$$

3. Compute the next prediction, Y_{n+1} by making corrections to our previous prediction Y_n:

$$Y_{n+1} = aY_n + k_n(X_n - Y_n) \tag{8}$$

   Intuitively, we are updating our previous prediction by a weighted amount of the difference between our observation and the prediction.

4. Predict the next MSE of our estimator:

$$\sigma^2_{e_{n+1}} = a(a - k_n)\sigma^2_{e_n} + \sigma^2_W \tag{9}$$

5. Increment n by 1 and repeat from step 2 for each successive observation.

The gist of the algorithm is to make successive corrections to previous predictions by using new data as it becomes available. Here is the pseudo-code version of the algorithm:

Algorithm: Kalman filter
Initialize: $Y_0 = \mu_Z$, $\sigma^2_{e_0} = \sigma^2_Z$ (These initializations can be based on fact or belief.)

For each time $n = 0, 1, 2, \ldots$:

1. Compute the Kalman gain:

$$k_n = \frac{a\sigma^2_{e_n}}{\sigma^2_{e_n} + \sigma^2_N} \tag{10}$$

2. **Correction:** Refine the previous prediction for the next estimate, Y_{n+1}:

$$Y_{n+1} = aY_n + k_n(X_n - Y_n) \tag{11}$$

3. **Prediction:** Predict the next value of the MSE, $\sigma^2_{e_{n+1}}$:

$$\sigma^2_{e_{n+1}} = a(a - k_n)\sigma^2_{e_n} + \sigma^2_W \tag{12}$$

3.2.4 MATLAB implementation.

Write a function that computes Y_n, given all of the signal model parameters and a set of observations X_n.

```
function [y, var_err, k] = my_kalman(x, z0, var_z0, var_W, var_N, a)
% Returns the output of the Kalman filter based on the following signal
% and observation model:
%
% Signal model:
% Z[n] = a*Z[n-1] + W[n]
% Observation model:
% X[n] = Z[n] + N[n]
%
% y : Kalman filter output
% var_err : computed MSE at each iteration
% k : computed Kalman gain at each iteration
```

Download the `test_kalman.m` script and `kalman_maybe.m` function from Blackboard. The `kalman_maybe.m` function generates observations and the `test_kalman.m` script is used to graphically visualize filter performance. You can test your filter's performance by generating observations and running them through the Kalman filter.

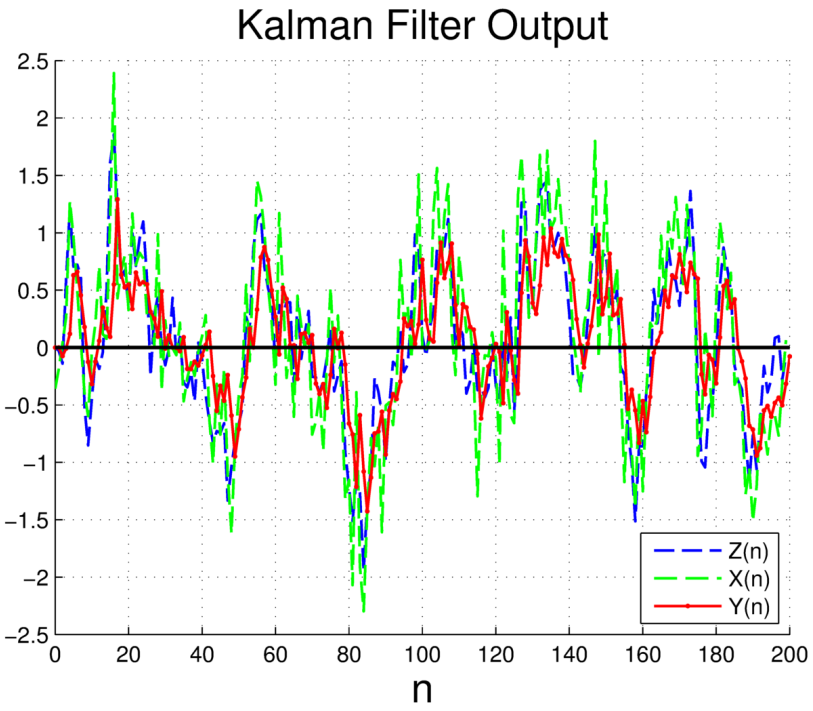


Figure 3: Example of Kalman filter output.

3.2.5 Steady state performance.

Here is a way to test if your Kalman filter is implemented properly.

Consider a signal defined by

$$Z_n = aZ_{n-1} + W_n, \quad n = 1, 2, \ldots, \quad \mu_Z = 0, \quad \sigma_Z^2 = 0 \tag{13}$$

where $\sigma_W^2 = 0.36$ and $a = 0.8$. The observations are given by

$$X_n = Z_n + N_n, \quad n = 0, 1, 2, \ldots \tag{14}$$

where $\sigma_N^2 = 1$. The Kalman gain at step n is given by

$$k_n = \frac{a\sigma_{e_n}^2}{\sigma_{e_n}^2 + \sigma_N^2} \tag{15}$$

The MSE at step n is given by

$$\sigma_{e_0}^2 = \sigma_Z^2 = 0 \tag{16}$$

$$\sigma_{e_{n+1}}^2 = a(a - k_n)\sigma_{e_n}^2 + \sigma_W^2 \tag{17}$$

$$= a\left(\frac{a}{\sigma_N^2 + \sigma_{e_n}^2}\right)\sigma_{e_n}^2 + \sigma_W^2 \tag{18}$$

In steady state, $\sigma_{e_n}^2 = \sigma_{e_{n+1}}^2 = e_\infty$ for all n. This produces the following equation:

$$e_\infty = \frac{a^2}{\sigma_N^2 + e_\infty} e_\infty + \sigma_W^2 \tag{19}$$

This is a quadratic equation in e_∞. Solving it for the positive root and substituting the result into:

$$k_\infty = \frac{ae_\infty}{e_\infty + \sigma_N^2} \tag{20}$$

gives the steady state Kalman gain and MSE. This means that in steady state, the predictor update is given by

$$Y_{n+1} = aY_n + k_\infty(X_n - Y_n) \tag{21}$$

and that the MSE remains constant.

Show (verify) that $k_\infty = 0.3$ and $e_\infty = 0.6$ for this example.

One of the output plots of the test script plots the MSE and Kalman gain at each time step. If your Kalman filter is implemented correctly, these two curves should converge to e_∞ and k_∞, respectively.

3.2.6 Some comments.

Here we considered a very specific implementation of the Kalman filter where we had all scalar quantities. The theory can be expanded to more complex dynamic systems described by matrix-vector equations; control theory and economics are some of the fields that utilize Kalman filters. Furthermore, the theory can handle all types of random processes, not just Gaussian processes.

The fact that this filter can be implemented recursively is crucial for real-time applications. In general, the filter coefficients would have to be determined by solving linear systems that grow with the number of observations, requiring significantly more memory and computation.

3.2.7 SUMMARY: Do the following as part of the pre-lab for this part.

1. Write your MATLAB function `my_kalman`.
2. Be prepared to show the TA that $k_\infty = 0.3$ and $e_\infty = 0.6$ for the given example.

3.3 Summary of the summaries: What you need to do before you come to lab.

1. Write your MATLAB function `my_notch`.
2. Make the pole-zero maps and frequency responses for all of the requested combinations of r and ω (nine total).
3. Be prepared to describe what happens to the frequency response of the notch filter as r is fixed and ω varies. Likewise, what happens when ω is fixed and r varies?
4. Write your MATLAB function `my_kalman`.
5. Be prepared to show that $k_\infty = 0.3$ and $e_\infty = 0.6$ for the given example.

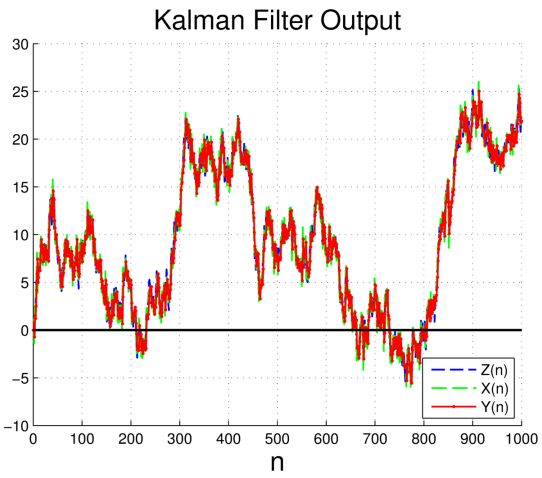


Figure 4: Tracking Brownian motion. Predicting the prices of financial assets perhaps?

4 In-Lab

Complete the following tasks. You must have all of the TA checks completed by a TA in order to receive full credit for the lab. Label all figures.

4.1 Removing that annoying whistle...

Imagine that you're at a show and the DJ starts playing this amazing new track. [1] Like anyone else with a cell phone, you decide to take a video of the occasion so that you can share a low quality recording of the moment with everyone you know on social media. Unfortunately, someone with a whistle decides to ruin your recording by blowing their whistle without abandon while the track is playing. You decide to attempt to remove this contaminating noise from the audio using your signal processing knowledge.

Figure 5: It ruins everything.

- Fortunately, you were able to confiscate the whistle from that person before you went home. A recording of it is stored in the `whistle.mat` file. **Download it from Blackboard.**

[1] Gareth Emery, Bo Bruce - U (Bryan Kearney Remix) vs. Gareth Emery, Christina Novelli - Concrete Angel (John O'Callaghan Remix) (Armin van Buuren mashup) to be exact.

- The song recording is stored in the `recording.mat` file. **Download it from Blackboard.** Note that the signal has two channels so you will have to process each channel separately.

Your task is to make use of the notch filter that you created and reduce the presence of the whistle as much as possible while minimizing changes to the actual song.

Some hints:

- What are the main frequency components of the whistle? You should make use of frequency analysis tools from previous labs.
- From the pre-lab: how do we selectively reject specific frequencies without affecting other frequencies that much?

TA CHECK: Describe your approach to solving this problem, using figures such as signal spectra, filter frequency responses, etc. How did you use the notch filter? How many times did you filter the signal? How good were your results? What if the whistle had a higher frequency sound, say 10 kHz – could we do better?

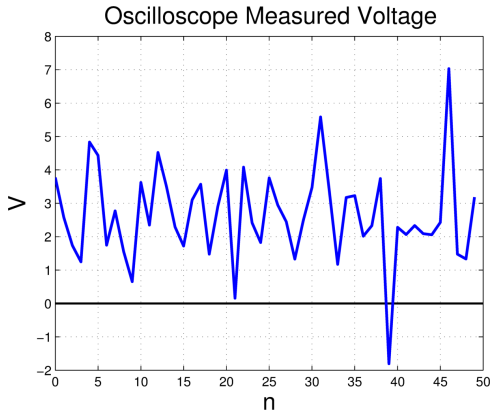
4.2 Finding the DC voltage level embedded in noise.

Suppose you're trying to power a circuit board with a DC power supply but the display that shows its voltage level is broken. The control knob for the voltage level is digital so there is no way of finding out what the voltage level of the power supply is; it could be anywhere between 0 and 8 V. You're also told that the power supply happens to be stuck at whatever unknown voltage level it is but that it is definitely DC. You're apprehensive about powering your circuit board with it because the board can only handle up to 3.3 V and you would rather not fry it.

You decide to power a circuit board that can handle up to 12 V with the power supply. You connect an oscilloscope probe to a component on the board that is at the power supply's voltage to measure the voltage. Lo and behold, the display of the oscilloscope shows an extremely noisy voltage signal; the oscilloscope probe must be damaged because you're certain that the actual voltage level is constant. How are you going to figure out the voltage level?

Your task is to estimate the actual voltage level using your Kalman filter. Translate the problem posed here to the signal and observation model that we based our Kalman filter on and determine the appropriate parameters for models and filter.

1. Set the measurement noise variance to equal 1.96.
2. **Download `voltage.mat` from Blackboard.** This is a vector of 50 noisy voltage samples from the oscilloscope that you will use to estimate the actual voltage level.
3. Feed it into your Kalman filter. For `z0`, your initial guess, guess anywhere in the stated range. For `var_z0`, the variance of your initial guess, set it equal to any positive number.



4. Try different values of `z0` and `var_z0` and try to determine the actual voltage level within 0.1 V. How do your choices of `z0` and `var_z0` affect the behavior of the filter output?

TA CHECK: Describe your approach to solving this problem, giving the signal model for this problem and its parameters. What is your guess of the voltage level? Is it safe to power your board with the power supply? Justify with your filter output.