Lab Four: Realization of FIR and IIR Filters

Digital Signal Processing Laboratory

Matthew Bohr

Dr. Pearlstein

11 November 2022

Student	Analysis	Development	Coding	Results	Writing
Matthew Bohr	100%	100%	100%	100%	100%

Introduction

The purpose of this lab was to expose students to digital signal filtering techniques. In this lab in particular, the students implemented the Direct Form II of a recursive IIR filter. The coefficients were selected such that the IIR filter became a bandpass filter. A LabView and MATLAB implementation were designed and are presented throughout this lab report.

Procedure

An IIR filter has an infinite impulse response… obviously. The difference equation of an IIR filter can be written as follows:

$$y[n] - \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} a_k y[n-k]$$

Moreover, the z-domain transfer function takes the following form:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

Following the natural flow of the difference equation, the Direct Form II realization of the IIR filter can be found:

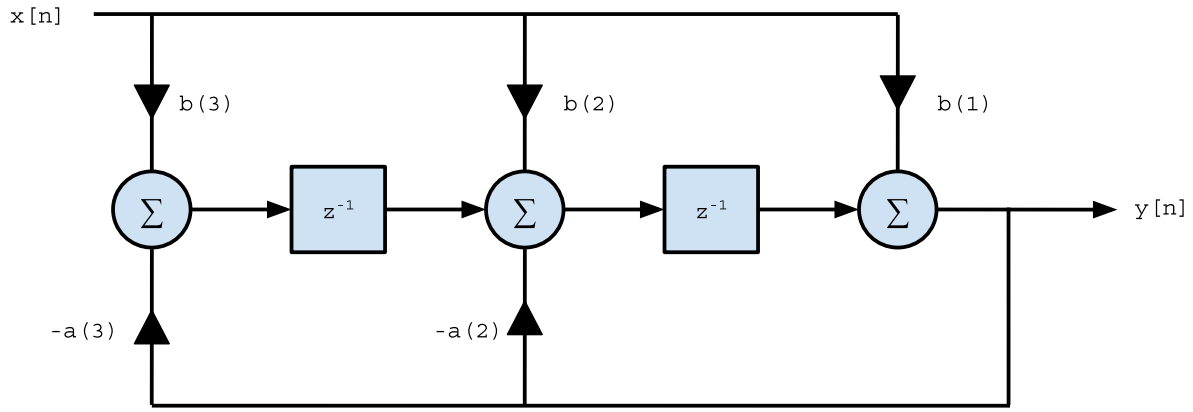


Figure 1: Direct Form II Realization of Recursive Filter

One may ask, "why use Direct Form II over Direct Form I", and that is because the Direct Form II is a *canonical* representation whereas Direct Form I is not. For a block diagram representation of a digital system to be canonical the number of delays needs to match the order of the system. So, if we have a second order system, then a canonical block diagram would have two delay blocks. In a digital system, a delay block typically indicates the necessity of memory since previous input or output sequence values need to be accessed, so minimizing the number of delays corresponds to a minimization of the memory used as well.

The filter was to be designed with an operating frequency of $\omega_0 = \pi/20$. The professor provided the students with a table detailing how we should select the weights of the coefficients ($0.8 \leq \alpha < 1.0$):

Coefficient	$\alpha = 0.95$	$\alpha = 0.98$
$a[1] = 2\alpha\cos(\omega_0)$	1.8766	1.9359
$a[2] = -\alpha^2$	-0.9025	-0.9604
$b[0] = 1$	1	1
$b[0] = 0$	0	0
$b[0] = -1$	-1	-1

Table 1: Filter Coefficients

In the first part of the lab, the filter was implemented on LabView and tested using a chirp signal. A chirp signal looks like a sine wave with an increasing frequency. This is perfect for testing our bandpass filter since we would expect the output signal to have the greatest magnitude somewhere during the middle of the signal when the frequency is somewhere in the middle between the maximum and minimum frequency. Appendix A and B shows the LabView design environment of our IIR filters.

After implementing the filter on LabView, the students also needed to implement the same filter on MATLAB. The first step was loading the filter_test_segments.mat file that contains the chirp signal. Then, the filter coefficients were computed so that they can be used when implementing the filter. The overall structure of the MATLAB filter is mainly derived from the difference equation describing the IIR filter, however, it also uses a shift register mechanism and a for-loop in order to compute the output sequence at each index while using the previous members from the sequences of x[n] and y[n].

On top of computing the output sequence of the FIR filter given a particular input sequence, the frequency response of our filter, the input chirp signal, and the output band passed chirp signal were plotted.

Results

After designing the IIR filter on LabView, it was finally time to perform some experiments. A chirp signal was the input to an IIR filter which was designed to operate as a band pass filter. Figure 2 below shows both the input chirp signal as well as the band passed output with $\alpha = 0.95$. The output signal very obviously demonstrates that our IIR filter is in fact only passing a small frequency band while attenuating frequencies outside this band. Figure 3 shows the same thing with $\alpha = 0.98$.

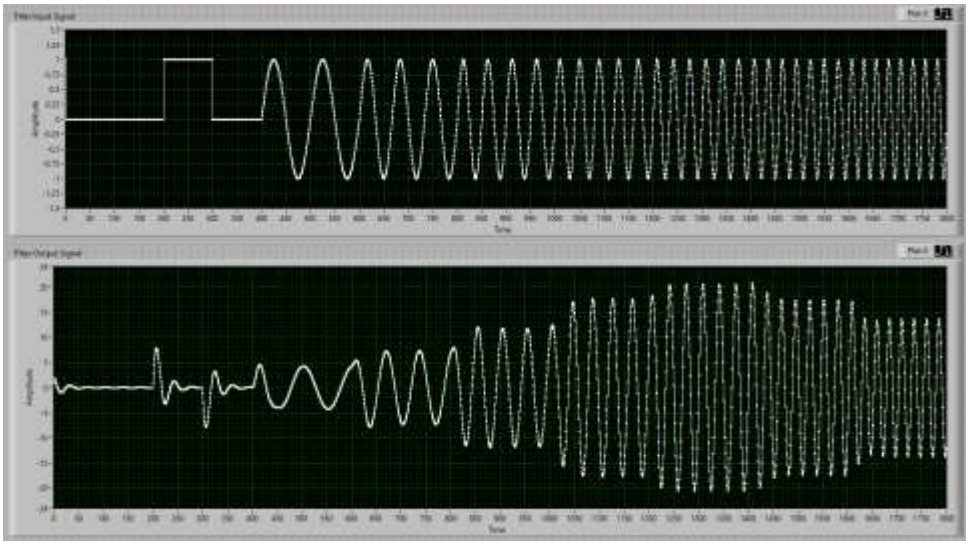


Figure 2: LabView Bandpass Filter Response (α = 0.95)

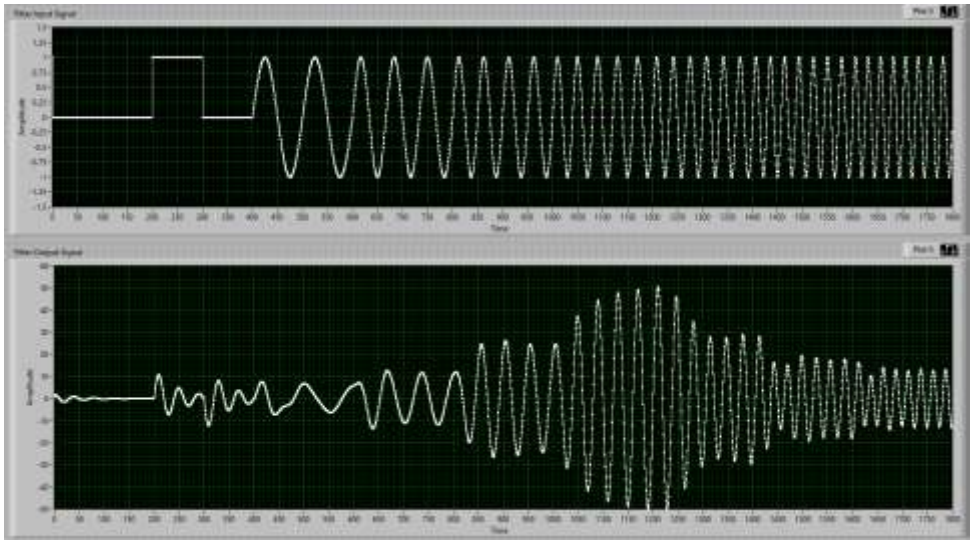


Figure 3: LabView Bandpass Filter Response (α = 0.98)

The IIR filter was also implemented on MATLAB. The frequency response (both magnitude and phase) is plotted in Figure 4. The chirp signal had discrete jumps in frequency. This is reflected in the graph of the output signal, in which the discrete jumps in amplitude are observed by the sharp jumps in attenuation intensity for each frequency.

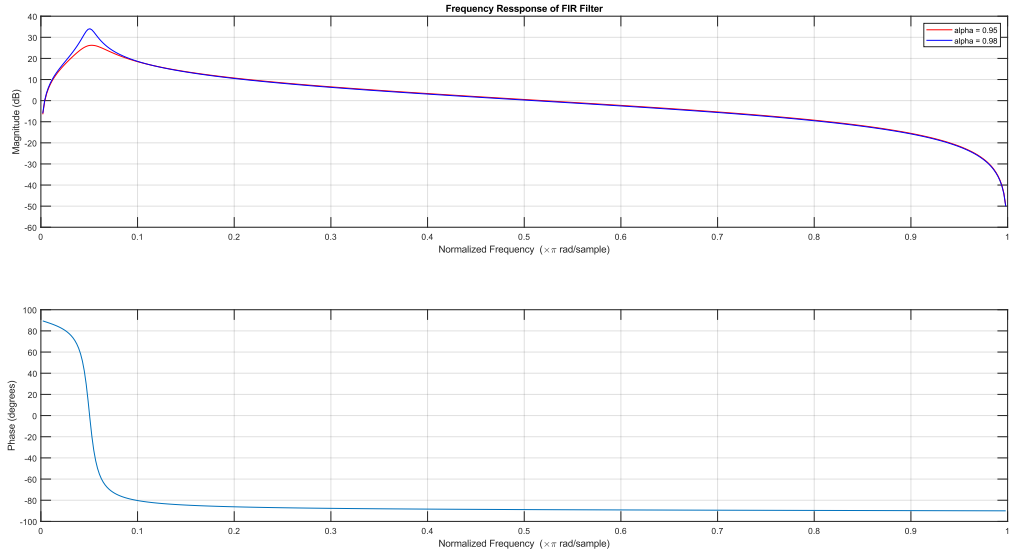


Figure 4: Frequency Response of IIR Filter on MATLAB

While the frequency response reveals a lot about the filter, an equally exciting way to showcase the functionality of the filter is simply by showing the output and input signals in the time domain, just as we did with LabView. Figure 5 shows both the input chirp signal and the output signal after the chirp has passed through our digital band pass filter. The results produced in MATLAB match those produced in LabView very closely.

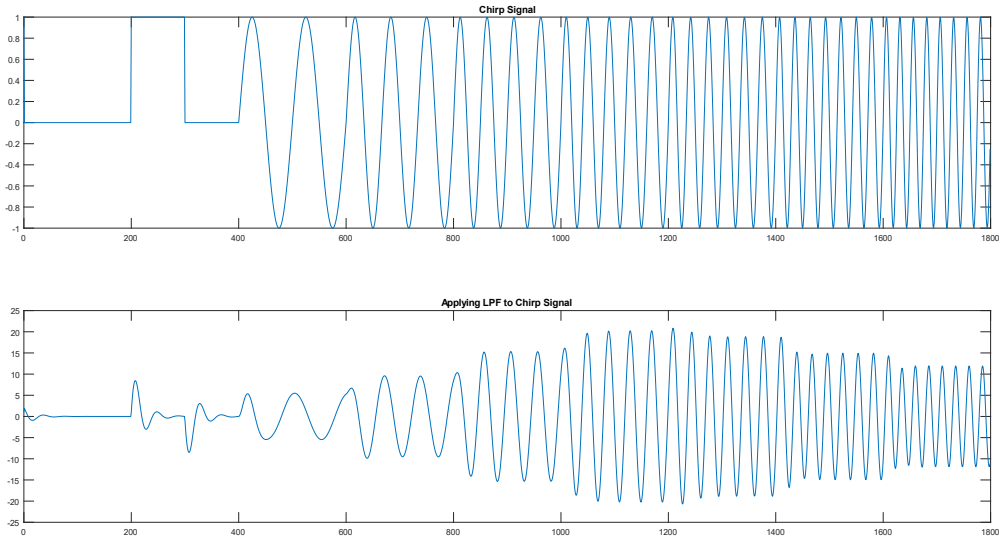


Figure 5: Chirp Signal Before and After Being Passed Through Filter

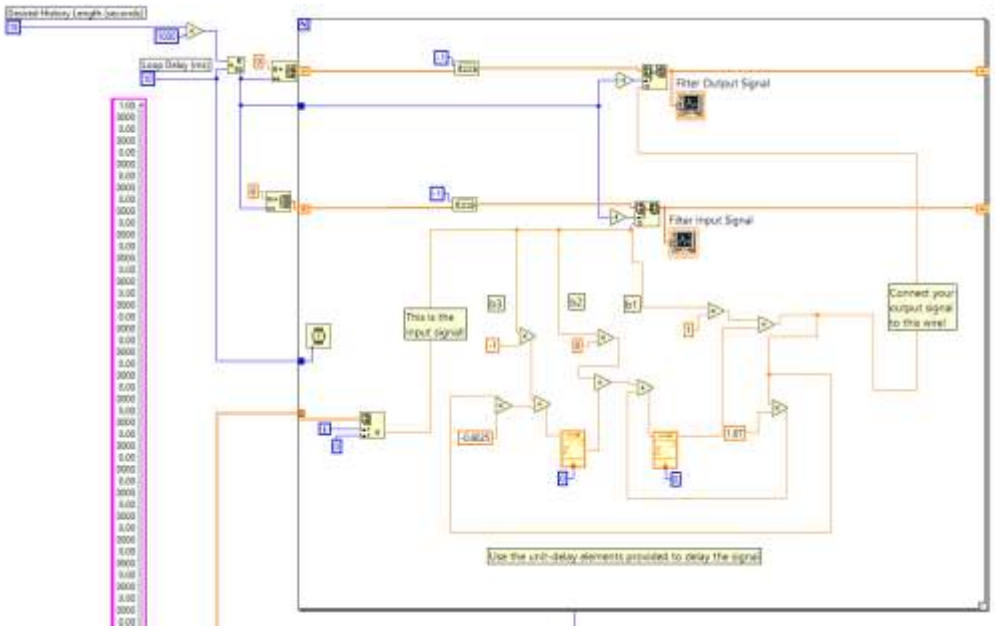
Bandpass filters allow signals at a particular frequency band to pass through relatively unaffected while attenuating frequencies below or above this band. Our chirp signal has discrete increases in frequency, hence the discrete jumps in amplitude of the output signal.

Conclusion

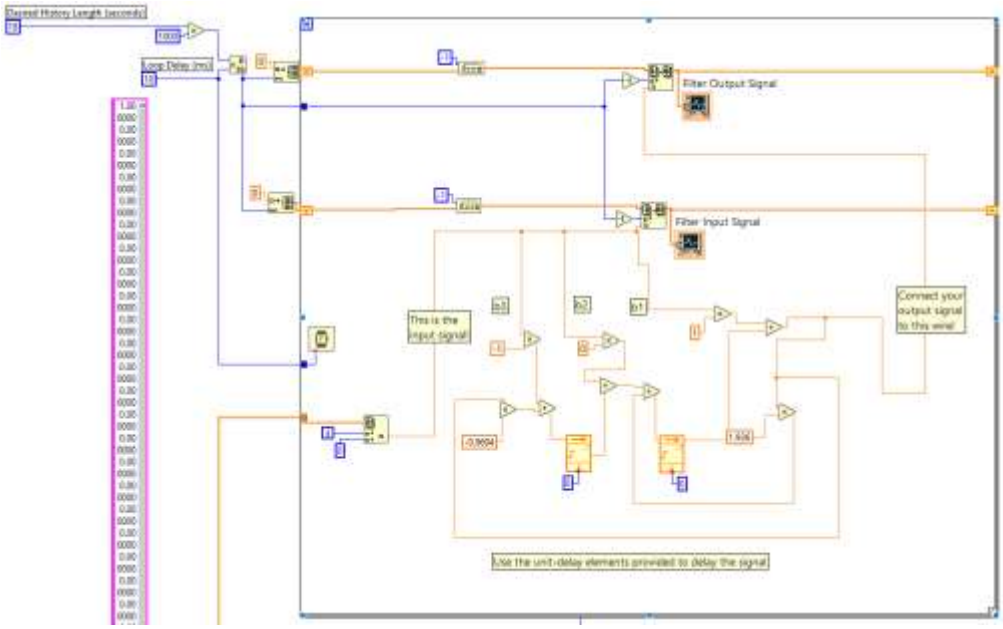
This lab provided students with insight into how an IIR filter may be implemented on various software platforms. We saw how you can choose the IIR filter coefficients to form a bandpass filter. Overall, the students learned useful skills about implementing digital filters and how they can be fine tune in order to process a signal in a particular way.

Appendix

Appendix A: LabView Diagram with $\alpha = 0.95$



Appendix B: LabView Diagram with $\alpha = 0.98$


Filter Output Signal
Filter Input Signal
This is the input signal
Connect your output signal to this wire
Use the unit-delay elements provided to delay the signal

Appendix B: MATLAB Code of IIR Filter-lab4_part1.m

```
clear
close all

%parameters for tuning IIR filter coeffs
alpha1 = 0.95;
alpha2 = 0.98;

%center frequency
w0 = pi / 20;

x_struct  = load('filter_test_segments.mat');     %filter_test_segments placed in struct x_struct
x_extracted = x_struct.x;                         %extracting member x from struct which holds chirp signal
coeffs

%matrix to hold filter coefficients
a1 = [1     -1*2*alpha1*cos(w0)     alpha1^2];    %using alpha1
a2 = [1     -1*2*alpha2*cos(w0)     alpha2^2];    %using alpha2
b =  [1           0                  -1    ];

%----------------implementing 2nd order difference equation-----------------
%variables to hold sequence values and their delays
%instantiated to zero since any member from x and y before n = 0 should be 0
y = 0;
y_n1 = 0;
y_n2 = 0;
x_n1 = 0;
x_n2 = 0;

%loop to populate output sequence using difference equation describing
%IIR filter
for n = 0:1799
    y(n+1)= -a1(2)*y_n1 - a1(3)*y_n2 + b(1)*x_extracted(n+1) + b(2)*x_n1 + b(3)*x_n2;

    y_n2 = y_n1;
    y_n1 = y(n+1);

    x_n2 = x_n1;
```

```
    x_n1 = x_extracted(n+1);
end

%plotting frequency response of filter using freqz function
figure(1);
freqz(b, a1);
hold on
freqz(b, a2);
legend('alpha = 0.95','alpha = 0.98')
hold off
lines = findall(gcf,'type','line');
set(lines(1),'Color','blue')
set(lines(2),'Color','red')
title('Frequency Ressponse of FIR Filter');

%plotting chirp signal and output chirp after filtered
k = 0:1799;
figure(2);
subplot(2,1,1);
plot(k,x_extracted);
title('Chirp Signal')
subplot(2,1,2);
plot(k, y);
title('Applying LPF to Chirp Signal')
```