# Adaptive Project
# Adaptive filter design using NLMS

Balasubramaniam M C
Roll- EE18B155

November 24th 2020

## Abstract:

This project is about using the NLMS algorithm to train an adaptive FIR filter for replicating different filter responses-FIR, IIR and other analog filters having orders lower or more than the length of the adaptive FIR filter used. We analyze the responses from training different filters, having different lengths and see if we can improve it in anyway or make observations on comparing the estimate and original frequency responses and give intuitive explanations for this. The use of having an adaptive filter is that traditionally, it is hard to change parameters of filters designed on FPGAs or ASICs. Hence, we can use a DSP/ filter prototype that uses the same input signals as the adaptive filter, generating desired outputs which the Adaptive filter tries to produce by changing its weights.
We try to minimize the following:

$$E|\mathbf{d} - \mathbf{u}w|^2 \tag{1}$$

w is estimated iteratively using the NLMS algorithm as shown below:

$$w_i = w_{i-1} + \frac{\mu}{\epsilon + ||u_i||^2} u_i{}^*[d(i) - u_i w_{i-1}] \qquad i >= 0 \tag{2}$$

where,
d(i) is the desired output at time i
mew is the step size
epsilon is the regularization parameter
wi is the weights of the adaptive FIR filter at time i
ui represents the i'th input regressor generated using a sliding window on input white noise data

Fig.1 shows how the adaptive FIR filter is implemented. Here, intuitively after a long time when the error is low, the weights estimated can act as the weights of the FIR filter predicted since it almost produces d(n) for any x(n). Hence, we can use the filter weights at the end of training to replicate the frequency response of the filter we desire.

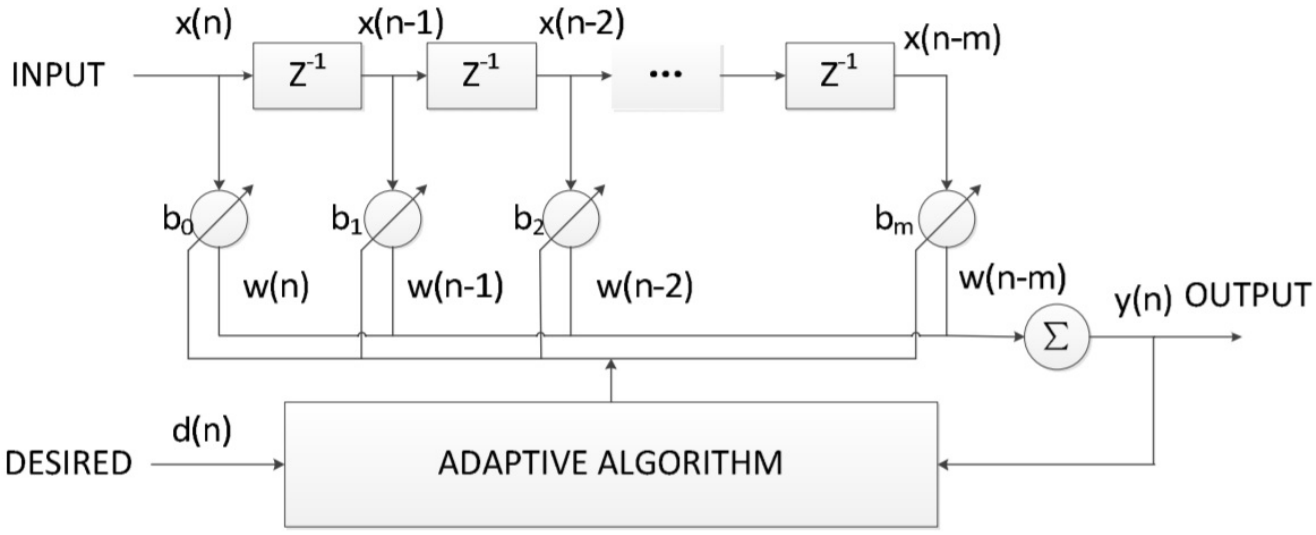


Figure 1: Direct form NLMS adaptive filter structure

## Introduction:

As shown in figure 2, we use the adaptive filter in parallel with the target filter having same inputs. With time, the error e(n) goes to zero and the adaptive filter imitates the target filter. We see that there is convergence in the time domain with signals d(n) and y(n). In order to ensure convergence in frequency domain, where the adaptive process tries to converge response at each frequency to the desired value, we use a 1 PSD (power spectral density) uniform white noise generator for the input signals. This is done so that we have non-zero equal PSD on all frequencies giving a better estimation. This is shown in fig.3. The input signal used is 1 PSD white noise sampled at 8000Hz generated using the wgn function for 20s (normally).

```
time = 20;
N=8000*time; %  @ 8kHz sampling rate
power =1;
noise = wgn(N,1,power);
```

## The eps-NLMS algorithm:

The epsilon-NLMS algorithm used and implemented in code is shown below:

```
function [w_out,A_out] = nlms(x,d,order)
```

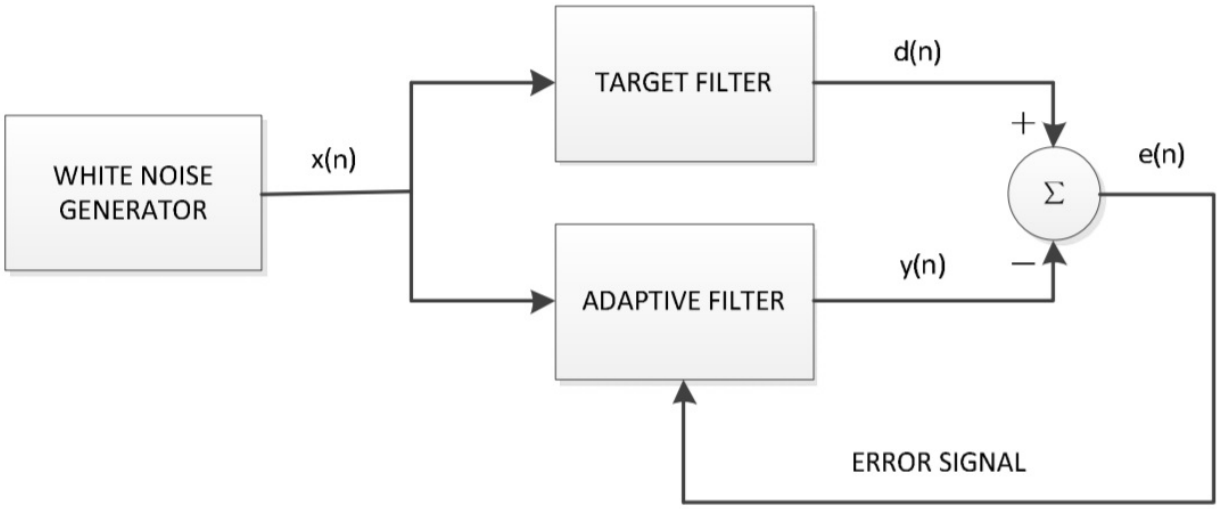


Figure 2: Target filter modelled by adaptive filter

```
wi = (zeros(order+1,1));  %weight vector initially zero
mew=0.01;            %step size
eps = 0.0001;        % epsilon chosen as a small positive parameter
A=[];
for i= 1 :length(d)
    di = d(i); %at time i
    c = i+order; %index for x
    %generating input regressors using sliding window.
    ui = flip(x(c-order:c)); % extracting inputs of size = filter order +1

    ei = di - ui*wi; %error
    wi = wi + (mew/(eps + ui*ui'))* ui'* ei; %estimating weights

    A= [A wi'*wi]; %storing progress of weight vector
end
w_out = wi;
A_out = A;
end
```

In most of the following sections we use a sliding window of the same length as the filter length on the white noise data to generate input regressors ui. The rest of the algorithm is implemented as explained in the abstract. Additionally, we generate a "progress" scalar- A = $wi^H * wi$. As the weight vectors start from 0, if A goes to a constant value (equal to the norm of the steady state weight vector) we say the filter converges. The step size and epsilon used normally are 0.01

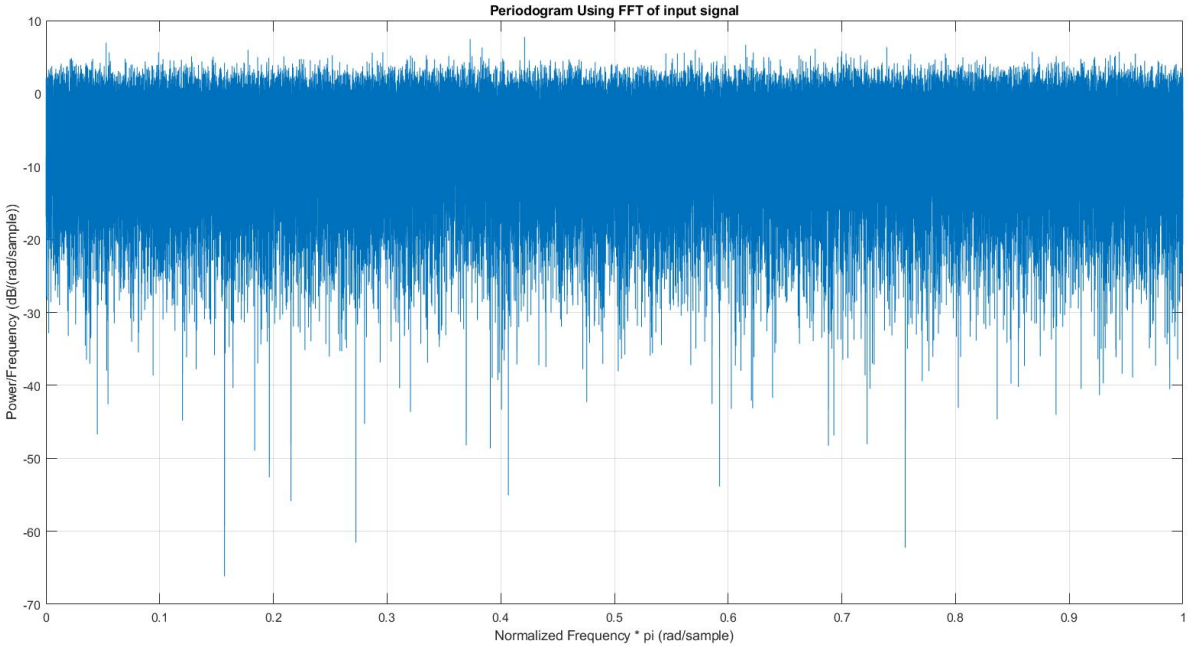


Figure 3: 1 PSD input with spectral density 0dBW/(rad/sample)

and 0.0001 for precision. The algorithm is derived from the Newton's method of gradient descent which gives a sufficient condition of step size being less than 2 for convergence and it's not dependent on the covariance of ui vector or any observation. However, here we are using the instantaneous approximation, so we have to verify with simulations.

# Performance using different Target Filters:

It has been shown that the Direct-Form FIR Filter structure (similar to what is in fig.1 without the adaptive algorithm part) can implement almost all types of filters- lowpass, bandpass, highpass, bandstop, multiband, etc. Hence, we can say that the eps-NLMS adaptive filter which has a direct form FIR structure may be able to do so as well.

## PART 1: An FIR Filter Target with a not larger length

### Equiripple FIR design method:

The following are the specifications of the multiband filter obtained via the Equiripple FIR design method.

```
%Multiband filter designed by equiripple FIR.
order = 98; % filter length = 99
%freq vector
f = [0, 0.28, 0.3, 0.48, 0.5, 0.69, 0.7, 0.8 ,0.81, 1];
%mag vector of mag. values for the respective frequencies.
a = [0, 0, 1, 1, 0, 0, 1, 1, 0, 0];
%weight vector for the weight of corresponding ripples in the frequency
%bands. Assume its the same ripple everywhere here
w = [1,1,1,1,1];
```

The filter coefficients are designed using the 'firpm' (Parks-McClellan Optimal FIR filter design) function

```
b =firpm(order,f,a,w);
```

We generate the desired output vector using the filter function giving the white noise vector as input.

```
% assuming a white noise input
x1 = transpose(noise);
%d is the desired output
d1 = filter(b,1,x1);
```

Next, we estimate an adaptive filter of same tap length( = 99) using the nlms function we declared earlier for 20s and plot the response using fvtool.

```
%estimation
%x1 is the input white noise vector
[wi A] = nlms(x1, d1, order);
%d1 is the desired output vector obtained from passing x1
%through the target filter
b_est = wi
%plotting
h = fvtool(b_est,1,b,1);
%b is the target filter coefficients vector
```

The resulting frequency responses are shown in fig.4. The norm of the weight vector is plotted in fig.5. Now, we compare the responses obtained when we train for 5s as shown in fig.6

**Hamming window design:**

The specifications of the multiband filter designed via a hamming window is shown below:

```
%hamming window designed multiband filter of same order
order2 = 98;
```

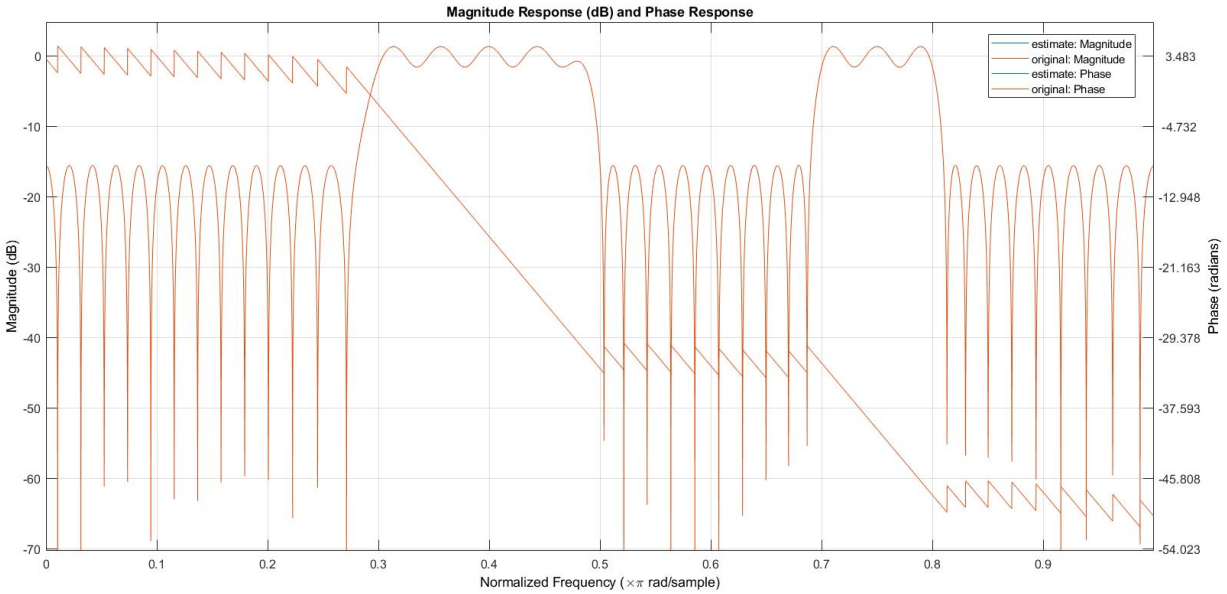


Figure 4: Plot of multiband equiripple fir filter estimation training for 20s.

```
%attenuates frequencies below 0.4 (normalized *pi)
low = 0.4;
%attenuates frequencies between 0.6 and 0.9(normalized *pi) as well
bnd = [0.6, 0.9];
%fir1 uses a hamming window to give coefficients for the multiband filter
b2 = fir1(order2,[low,bnd]);
```

Now, we estimate using an adaptive filter of length 99 (training for 20s). The responses are shown in fig.7. Additionally we plot the magnitude response for a 5s training period as well (fig.8).

**Kaiser window design:**

The specifications of the lowpass filter designed via a kaiser window is shown below:

```
%making a low pass filter of order 108 via kaiser window

%the end frequency of the passband and start frequency of the stopband
fcuts = [1000 1500];
%weight of magnitude on the bands as per fcuts
mags = [1 0];
%passband ripple of 5%, stopband ripple of 1%
```

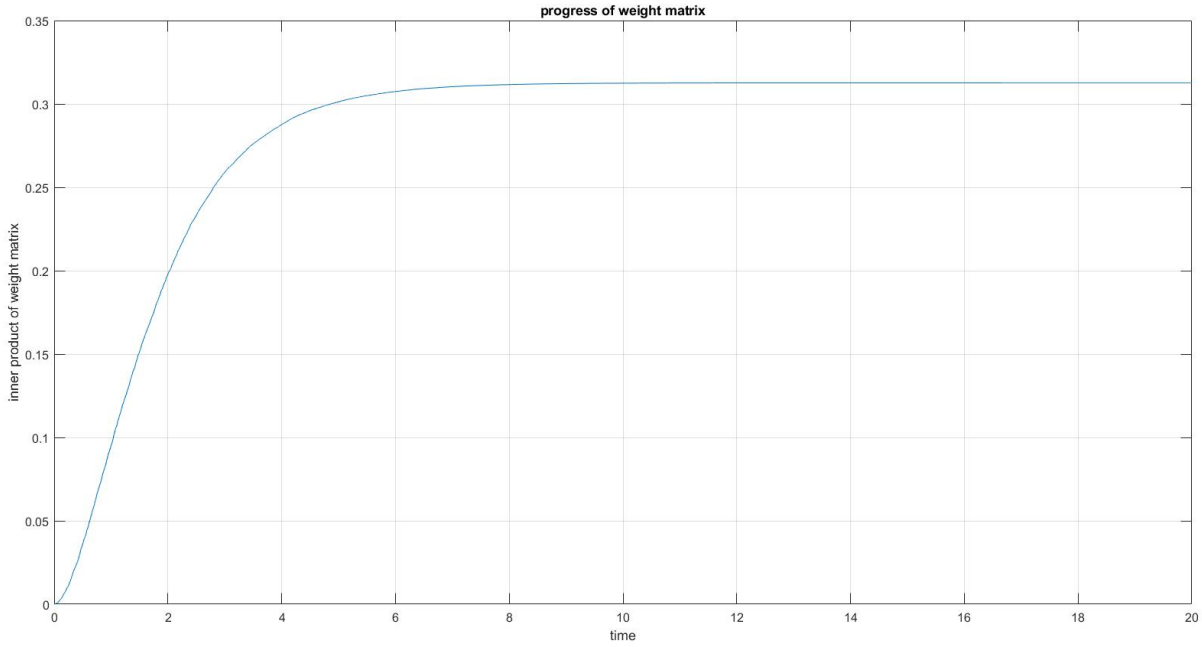


Figure 5: Plot of square of norm of the weight vector for estimating the multi-band equiripple fir filter (training for 20s).

```
devs = [0.05 0.01];
%3*Fs is done to get a filter order of 108, we anyway deal with normalized
%frequencies in the digital domain.
%generating order, normalized frequencies, beta-shape factor for kaiser window
[n,Wn,beta,ftype] = kaiserord(fcuts,mags,devs,3*Fs);
%using the order, shape factor,etc.
%with a kaiser window to generate FIR filter coefficients
hh = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
%adaptive filter order = 108 as well
order3 = n;
b3 = hh;
```

Now, we estimate using an adaptive filter of length 109 (training for 20s). The responses are shown in fig.9. Additionally we plot the magnitude response for a 5s training period as well (fig.10).

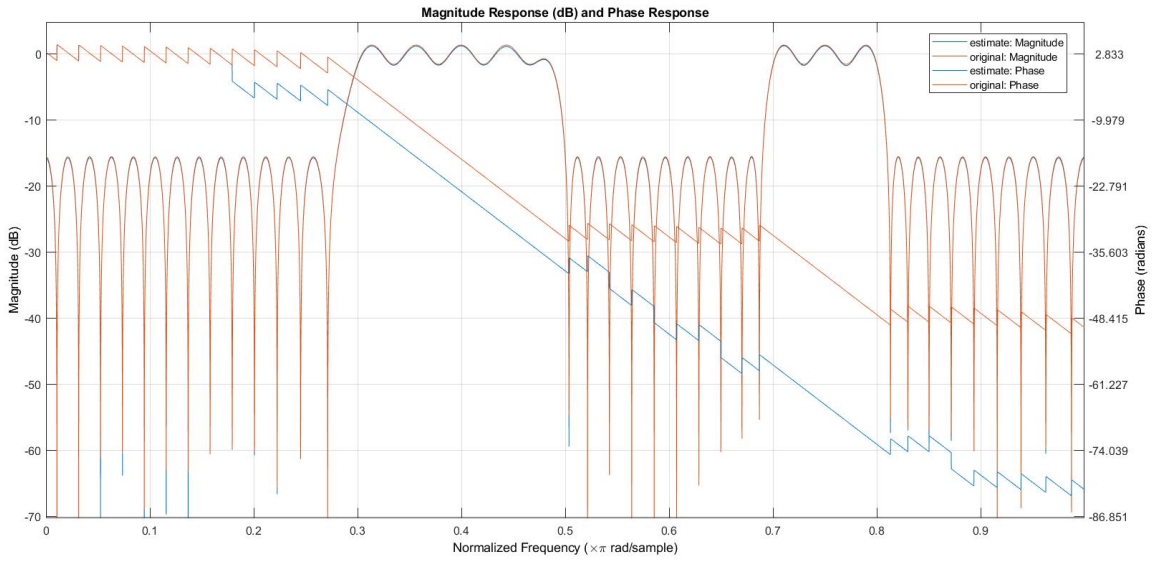


Figure 6: Plot of multiband equiripple fir filter estimation (training for 5s).

**Observations and Conclusions on Graphs:**

- From fig. 9,7 and 4, we can see that the NLMS algorithm perfectly estimates the magnitude and phase responses for a Multiband equiripple FIR filter , Multiband Hamming FIR filter as well as the lowpass Kaiser window FIR filter having the same length as the adaptive filter length for a training period of 20s. This is right intuitively because as we saw earlier, the direct form NLMS FIR adaptive filter structure after attaining steady state acts as the direct form FIR structure similar to the target filter of the same length. Hence, it replicates the responses perfectly using 20s of training period at 8kHz sampling rate and the small step size (0.01) resulting in convergence as we can see in the progress of the weight matrix( or vector) in fig. 5. It attains a constant value after a while showing convergence. This also means that the eps-NLMS algorithm works for estimating different FIR filters (multiband, lowpass) from different design methods, like the Hamming window,kaiser window and the equiripple design when the filter length is the same or lower than the adaptive FIR filter length. If it's lower in length then the rest of the coefficients in the estimated weight vector tend to zero.

- When the training time is changed to 5s, from fig. 6, we see that for the multiband equiripple fir filter the magnitude is almost converging, but the phase is not. A possible reason is that when the magnitude is very small, the real and imaginary part of the value of response at that frequency

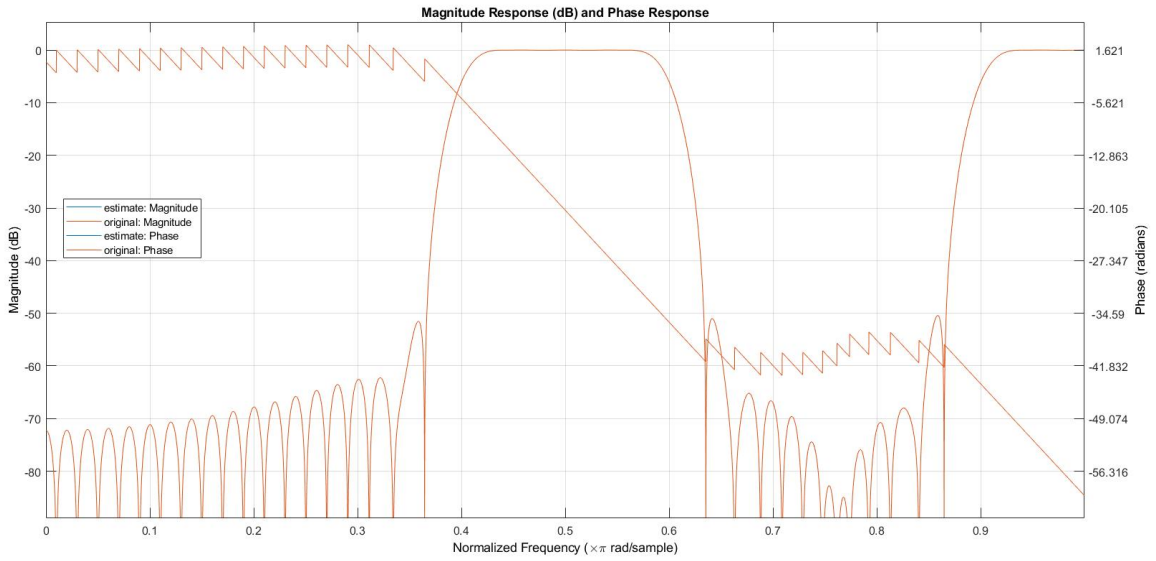


Figure 7: Plot of multiband hamming window fir filter estimation (training for 20s).

is also relatively small, which means the error in coefficients is relatively higher than this value giving huge variations in phase (because its roughly of the form $tan^{-1}(y/x)$. For ex: if the value is close to zero and error in coefficients can make it go from +ve to -ve the phase change would be 180 degrees. The following is the complex frequency response at w for an FIR filter:

$$h(jw) = \sum_{n=0}^{L} a_n e^{-jwn} \tag{3}$$

- From the magnitude responses in fig. 8,6 and 10 we can see that for the same training time, using the filter length equal to target filter length, we obtain the most converging result in multiband equiripple filter , followed by lowpass kaiser window fir and finally the multiband hamming window filter. This is because the variation of magnitude responses across frequency is different in them. For the equiripple filter, we have an equal ripple across the spectrum in both the bands. For the hamming window filter, we have a highly varying ripples and finally for the kaiser window filter, the ripple varies gradually. This variation affects the variance of **d** (desired output) and when there are high variations in d(i), the instantaneous approximation may be less efficient and the update parameter p in eqn. 2, varies a lot causing the weight vector to change a lot. Hence, not all three magnitude responses converge at 5s.

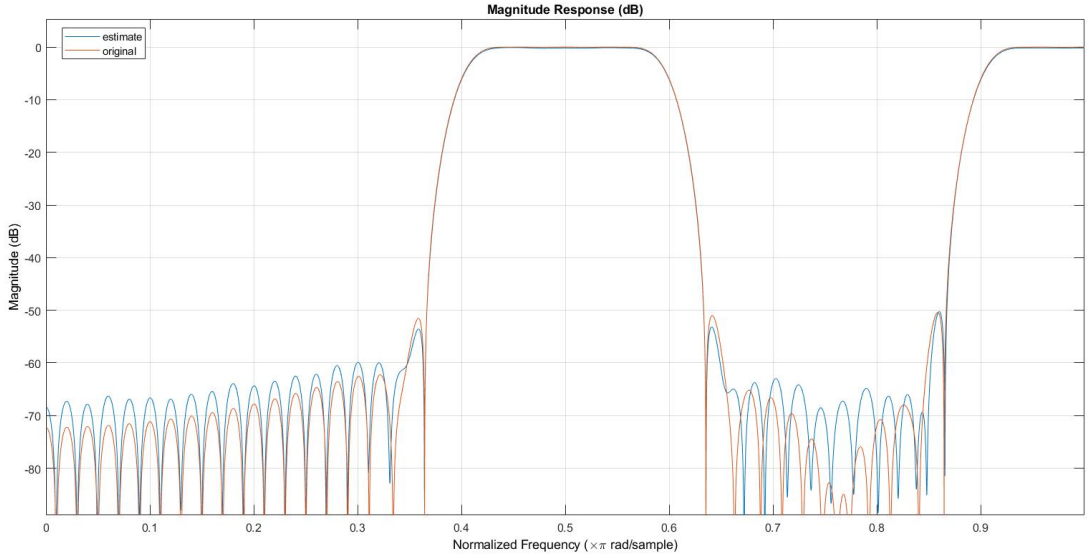


Figure 8: Magnitude response plot of multiband hamming window fir filter estimation (training for 5s).

## PART 2: A low order IIR Filter Target

The specifications of the target IIR filter are:

```
%Bandpass using Butterworth IIR method.
%Filter order 10
%Fc1 = 0.3 rad/s (normalized)
%Fc2 = 0.7 rad/s (normalized)
```

## Using the Adaptive FIR model:

We try to approximate a lower order IIR filter with an equivalent FIR filter of higher order. The step-size used is 0.01 and the length of the adaptive FIR filter used is 99. As done before, we use the same nlms function(training for 20s) , using input regressors generated via a sliding window on white noise data. The frequency response is shown in fig. 11. Additionally, because the filter is IIR, we can't exactly replicate the magnitude response. There are some slight differences at around -100dB as shown in fig. 12. The pole zero plot is shown in fig. 13.

After this, we see the variation of the response with order of the adaptive filter. So, we use an length 101 adaptive FIR filter. The response is shown in fig. 14.

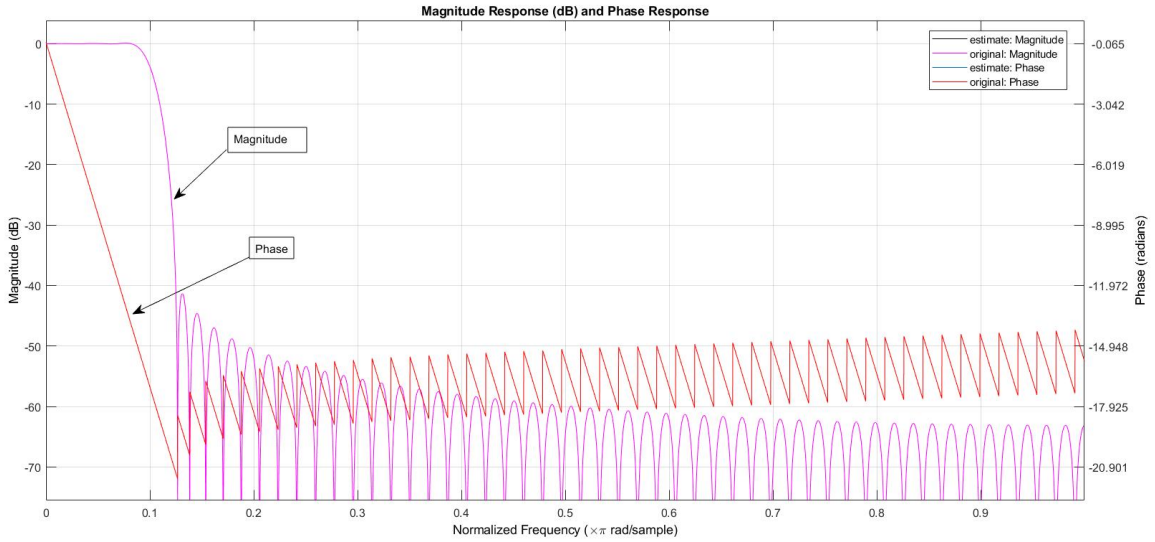


Figure 9: Plot of lowpass kaiser window fir filter estimation (training for 20s).

Additionally to see the variation with step size, we decrease the step size to 0.001 and take ten times the input data size. The results are in fig. 15.

## Using the IIR Adaptive model:

We now attempt to use an IIR Adaptive model filter with the same NLMS algorithm to make an IIR filter whose parameters are close to the original using the same training period of 20s. The idea is to obtain d(i) (current desired output) as a product of a weight vector and a modified input regressor (which is what we want for NLMS) as shown below:

$$d(n) = b1*u(n)+b2*u(n-1)....+bm*u(n-m+1)-a2*d(n-1)-a3*d(n-2)...-ap*d(n-p+1) \quad (4)$$

where,
[b1 b2 b3...bm] is the IIR numerator coefficient vector in decreasing degrees
[a1 a2 a3...ap] is the IIR denominator coefficient vector in decreasing degrees (a1 is 1)
d(n) is the current desired output
u(n) is the current white noise input
The above equation can be derived from the digital IIR transfer function. Using this form, the modified weight vector and input regressor would be:
Weight vector:[b1 b2 b3...bm -a1 -a2 -a3 ...-ap ]
where, a1 is 1
Input regressor:[ u(n) u(n-1) ... u(n-m+1) d(n-1) d(n-2) ...d(n-p+1)]

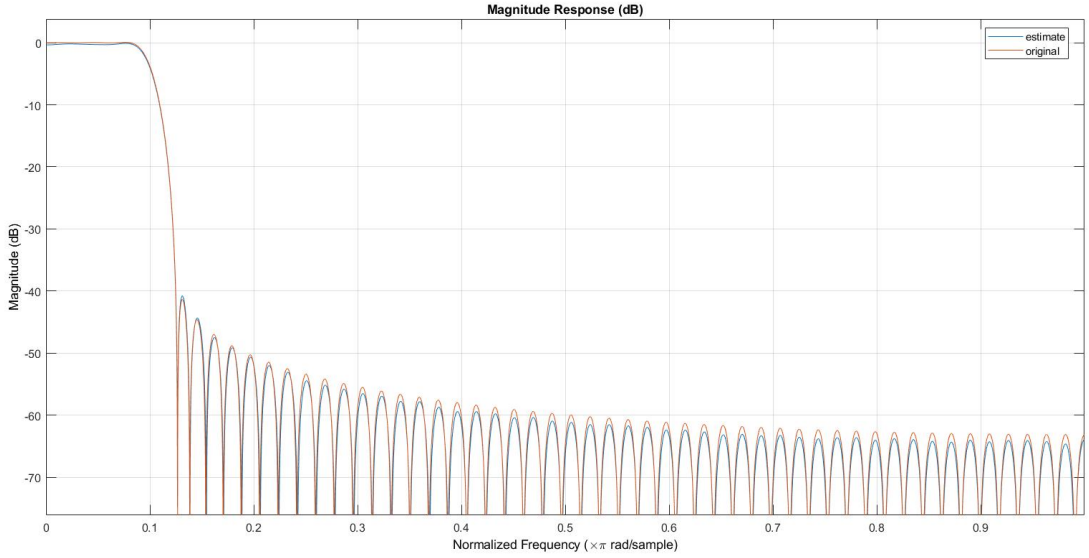


Figure 10: Magnitude response of lowpass kaiser window fir filter( original and estimated with training for 5s).

The following code shows the core of the modified algorithm :

```
%step size
mew3=0.01;
% epsilon chosen as a small positive parameter
eps3 = 0.0001;
%Modified weight vector
filt_len3 = length(b) + length(a) -1;
wi_3 = (zeros(filt_len3,1));
%following is inside the loop at ith iteration
%a is the denominator weight vector
%b is the numerator weight vector
%extracting ui of length(b)- the first part of the inp. regressor
    c = i+length(b)-1;
    %x3 contains white noise inp.
    ui = flip(x3(c-(length(b)-1):c));
    %2nd part of inp. regressor- di_block
    %extracting the di_block of length(a) not including latest sample
    c2 = i+length(a)-2;
    di_block = flip(d3(c2 - (length(a)-2):c2));
    %latest sample extracted separately or d(n)
    d_latest = d3(c2+1);
```

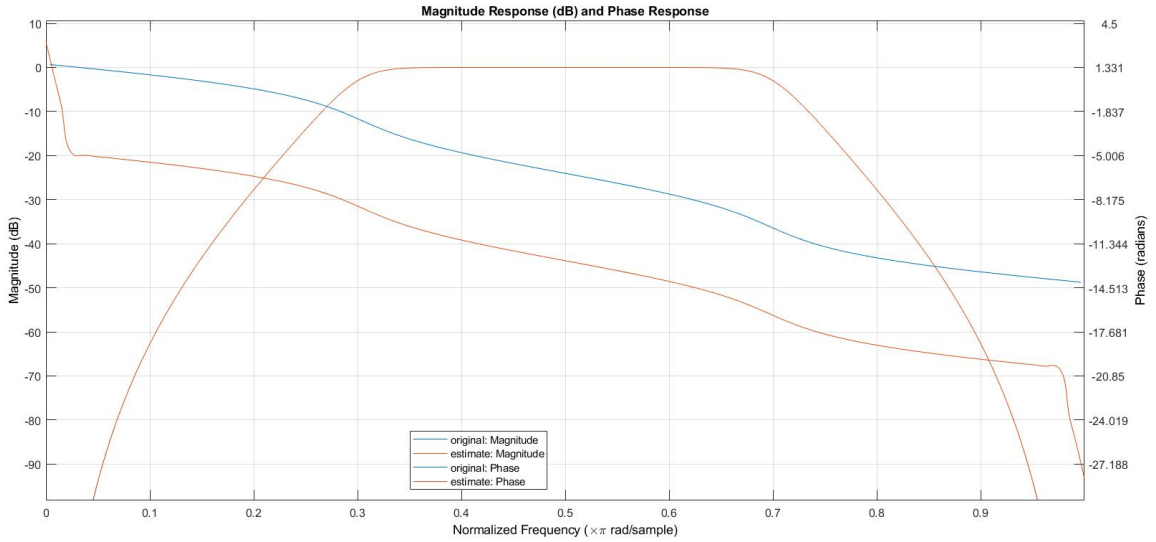


Figure 11: Frequency response of length 99 Adaptive fir filter estimating the 10th order target IIR filter with step size of 0.01.

```
%concatenating di_block and ui to give final input regressor
ui_net = [ui di_block];
```

After this would follow the same nlms update relation using the modified weight vector and input regressor. The plots for this method are shown in fig. 16.

**Observations and Conclusions on Graphs:**

- As we can see from fig. 11,12 and 13, the Adaptive FIR filter's characteristics doesn't exactly converge to that of the target IIR butterworth bandpass filter. This is obvious since, the FIR and IIR filters have different structures and their responses can never exactly match. In fig. 11, we can see that although the magnitude is almost the same, the phase differs. Due to the same reason in fig. 12, we can see small differences in magnitude at around -100dB. Finally in fig.13, the pole zero plot takes that interesting shape because the poles are known to boost the frequency at the pole and zeroes are known to zero out frequency at the zero. So, the best way an adaptive FIR filter replicates an IIR one is to have as many zeroes as possible in a rough circle except at the poles of the original filter however, an additional 98 poles are also added at origin (because its FIR). Now, since the zeroes do this only for the outer poles, the responses of the FIR and IIR filters can't be exactly the same.

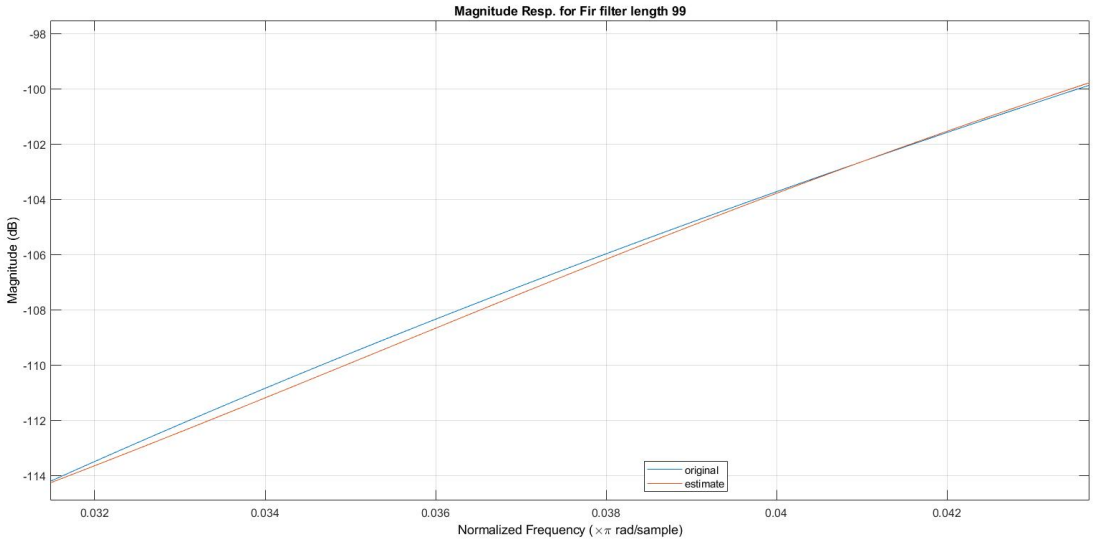


Figure 12: Differences in magnitude response of fir filter( length =99) and target iir filter.

- The phase responses for the Adaptive FIR and target IIR filter in fig.11 don't match mainly because of the deviations at the edges of the spectrum. Otherwise, they differ roughly by 2pi in between. In order to remove the deviation at the edges, we increase the Adaptive Filter's length to 101 using the same step size. We observe that the magnitude and phase almost completely match, except for minor deviations at the start and end frequencies( as shown in fig. 14).
  Explanation: An IIR transfer function with a polynomial in the denominator can be written as an infinite order polynomial after taking the expansion of the inverse of the denominator polynomial. So, if we use an FIR filter for approximation, the more it's order, the better the approximation. At the edge frequencies (0 and pi) the magnitude is very small. Hence, due to the error in coefficients, the phase can vary a lot (as seen earlier). Therefore, on using a length 101 FIR filter, we obtain a better approximation ,decreasing error in coefficients and decreasing the error in phase.

- As shown in fig. 15, upon decreasing step size to 0.001 and taking 10 times the input data, we still don't get a better response for the FIR adaptive filter (length =99). This is because the IIR response can't be replicated using the same order. More zeroes are needed for a better pole-zero response( as discussed earlier) and for a better approximation of the IIR polynomial( from the previous point).

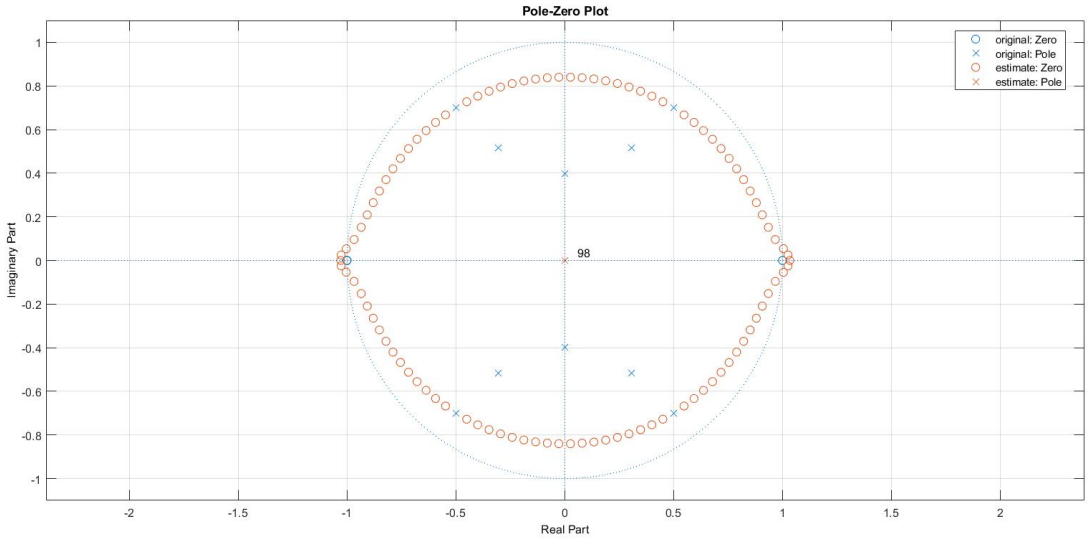


Figure 13: Pole-zero plot of fir filter(length =99) and target iir filter( step size =0.01).

- Finally, we try to directly estimate the filter using an Adaptive IIR Model (shown in fig. 16). For a 20s training time, we don't get as good a result compared to the FIR case . This is mainly because the filter hasn't converged yet at 20s. An explanation for this slow convergence is that the instantaneous approximation isn't that efficient here since the modified input regressor has previous desired outputs( d(i-1) ..etc). Since the desired outputs are dependent on previous inputs, the contribution of these inputs( u(i-2)...) are approximated into one value (d(i-1)) due to instantaneous approximation. Hence the slow convergence.

## PART 3: FIR filter target with a larger length

In this section, we use a Hann window FIR filter with order higher than that of the adaptive FIR. It's specifications are as shown below:

```
Response Type: Lowpass
Design Method: Hann window
Filter order: 20 or 70
Cutoff freq. (normalized): 0.5
```

We can design a Hann window digital FIR filter using the following function:

```
Fc = 0.5;
Fs = 8000;
```

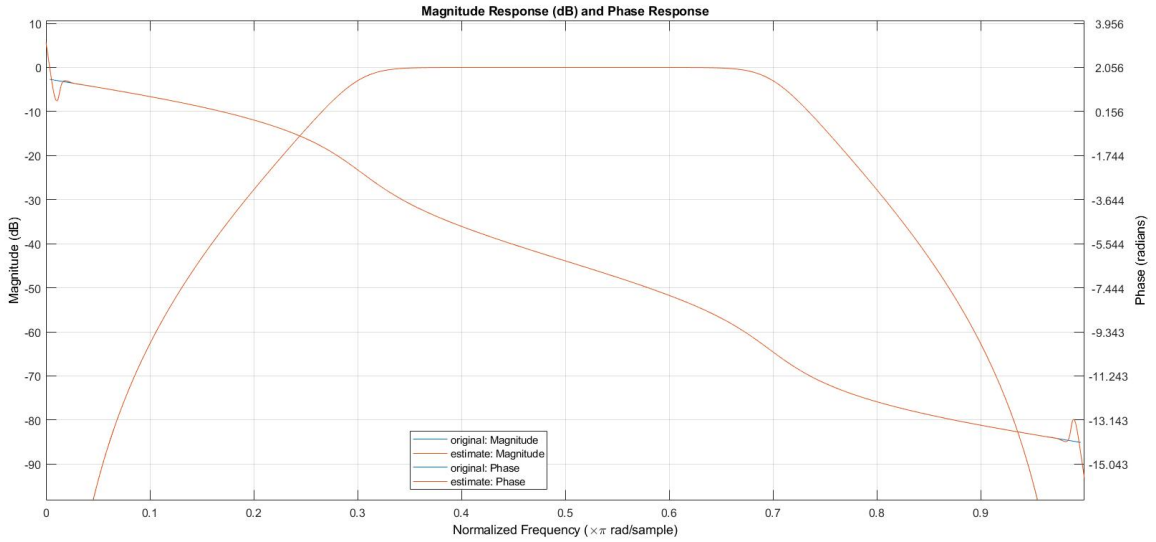


Figure 14: Frequency response of Adaptive fir filter( length =101) and target iir filter with step size 0.01.

```
%N is the order
lowpass_hann = designfilt('lowpassfir','FilterOrder',N...
...,'CutoffFrequency',Fc*Fs/2, 'Window',hann(N+1),'SampleRate',Fs);
```

We experiment on two cases: Target filter order being 20 (low order) and 70 (high order).

**Using a Lower order adaptive filter:**

The comparing filter of order 18 (other specifications remaining same) can also be generated using the designfilt function and a Hann window. This is done to see if the target filter is not far from the describing abilities of the lower order comparing filter.If this is true, the adaptive filter of order 18 gives a better and closer response to that of the target filter's compared to the comparing filter. This can be seen in fig.17.
After the weights of the adaptive filter are calculated, it's converted into a 'dfilt' object to plot via 'fvtool' as shown below:

```
%estimating weights
[w2 A2] = nlms(x2, d2, N_adap2,mew1);
%creating filter object
Hd_adap2 = dfilt.df1(w2,1);
%plotting
hfvt2 = fvtool(Hd_adap2,'Color')
```

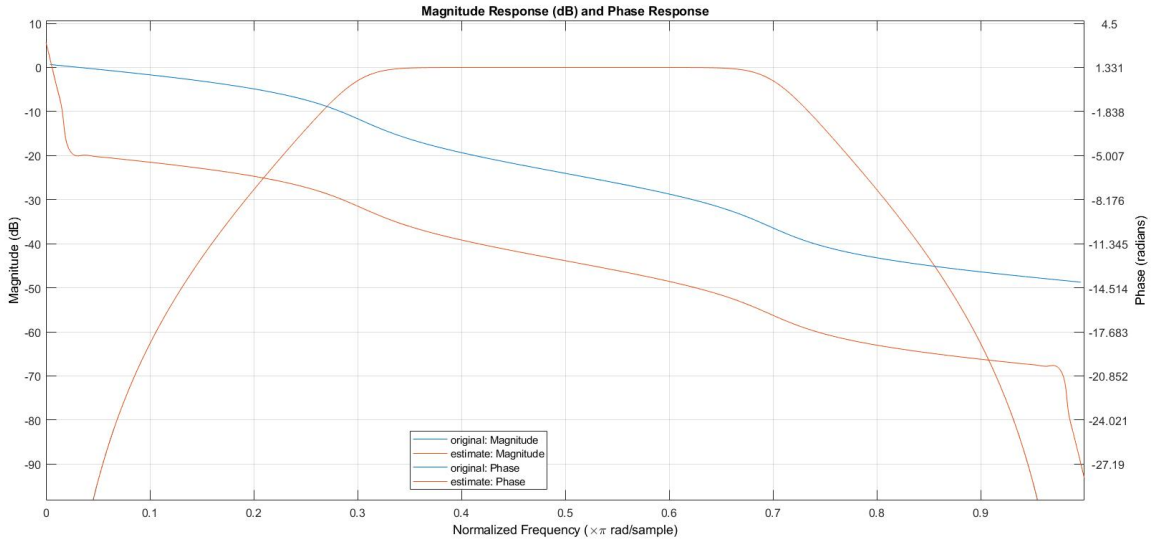


Figure 15: Frequency response of fir filter( length =99) and target iir filter with step size 0.001 and 10 times size of input training data.

Following this we plot the phase response in fig. 18, where we can see the adaptive filter's phase diverges after a frequency.

Now, we reduce the order of the Adaptive filter as much as we can to describe the higher order target. We see that we can't go below an order of 17 as shown in fig. 19, where the response diverges for order 16.

**Target filter with higher order:**

We now use a target lowpass Hann FIR filter of order 70 (with other specifications being same) and adaptive filters of orders 68 and 67, in order to see how the responses vary with order of the FIR adaptive filter for a higher order target compared to the earlier lower order one (20). We can see that the response diverges for order 67 as shown in fig.20. The code for this is the same as earlier except for a different order.

**Observations and Conclusions on Graphs:**

- As we can see from fig. 17, the frequency response of the target filter(order 20) is not far from the describing abilities of the minor length comparing filter(order 18). Additionally, the adaptive filter of the same order (18) gives a better and closer response to the target filter's response. This is because the Hann window FIR filter design method of making the comparing

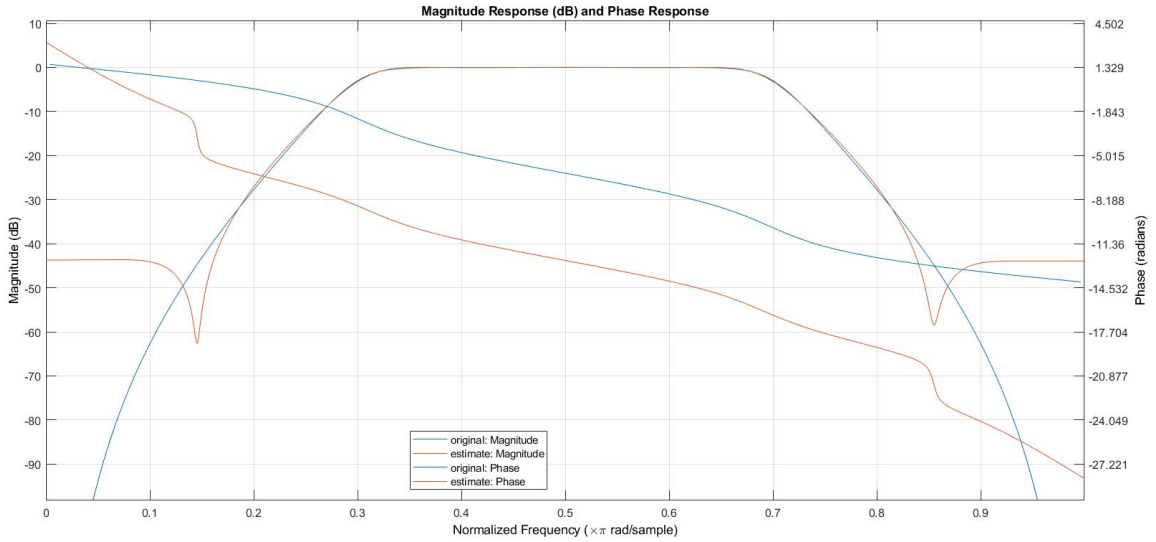


Figure 16: Frequency response of Adaptive IIR Model filter(trained for 20s with step size 0.01) and original Target IIR filter .

filter of a given order ensures that the frequency response specifications (here- cutoff freq.) are met using a Hann window(length = order +1) by setting certain parameters like stopband attenuation, ripple etc. accordingly. On the other hand, an adaptive filter tries to mimic the original response as much as it can which is why we can see it almost outlines the original response in fig.17. However, the stopband attenuation isn't as good. Therefore, the two methods aim to do different things and an adaptive filter will always try to produce a closer response to the original as compared to a comparing filter( of same order lower than the target filter's order).

- From fig. 18, we can see that the phase of the adaptive filter of order 18 diverges around a normalized frequency of 0.7. The cutoff frequency of the original filter is 0.6 and the stopband also starts roughly from 0.7. Considering that the adaptive filter's coefficients are padded with extra zeroes to attain the same order as the original filter, we can compare coefficients more easily.We see that as the magnitude is very low in the stopband, error in coefficients of the adaptive FIR compared to the original filter would be relatively higher than the original complex response values in these frequencies(due to attenuation in stopband). This affects the phase response a lot(as seen earlier). Hence the divergence in phase.

- For a target order of 20, we can see that the magnitude response of the

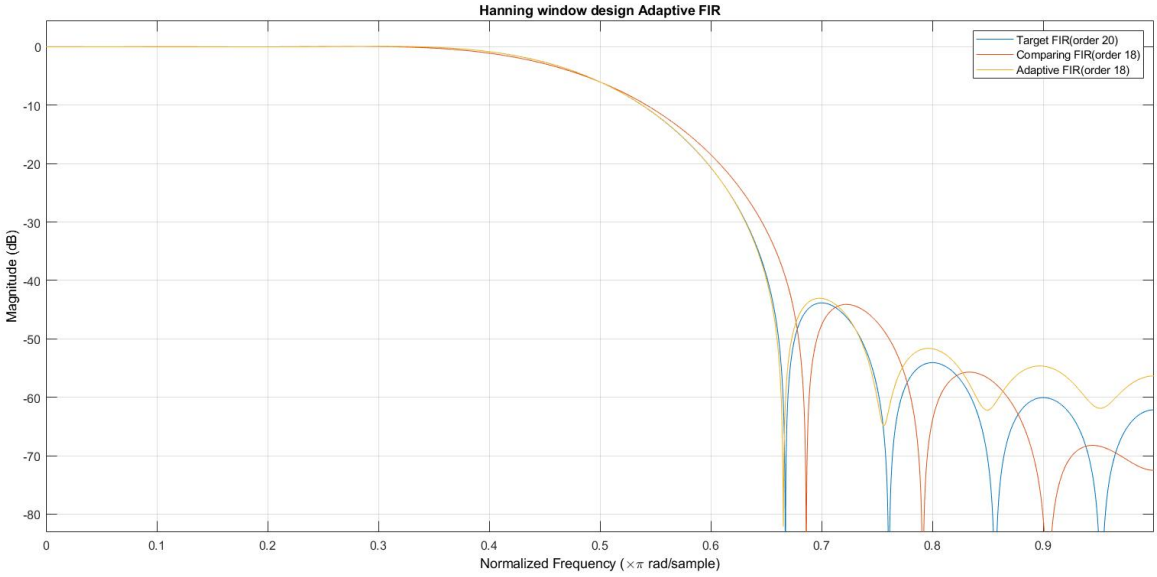


Figure 17: Magnitude responses of a Target lowpass Hann FIR filter of order 20 ,a comparing Hann and adaptive FIR filter of order 18 using step size 0.01.

Adaptive FIR filter diverges when its order is 16 (as shown in fig.19). This is probably because the comparing filter of order 16 wouldn't have a magnitude response close to the original magnitude response. Now, on using a target filter of order 70, we see that the magnitude response of an adaptive FIR filter of order 68 almost perfectly converges to the original while the Adaptive FIR of order 67 suddenly diverges after a certain frequency as seen in fig. 20. This sudden divergence on changing the order for a higher order target as compared to the more gradual change seen in the lower order case can be explained based on the ripples in the stopband. As the order of the original filter increases for the given specification of cutoff frequency, its seen the number of stopband ripples increases. This variation causes more variation in d(i)(desired output) and it becomes harder for the filter to track and adapt the weight vector to give minimum error. The error in steady state can be approximated to the form:

$$MSE(approx) = E||\mathbf{d} - \mathbf{u}w_{ss}||^2 \tag{5}$$

where,
$\mathbf{d}$ = desired signal random variable
$\mathbf{u}$ = input regressor random vector
$w_{ss}$ = steady state weight vector

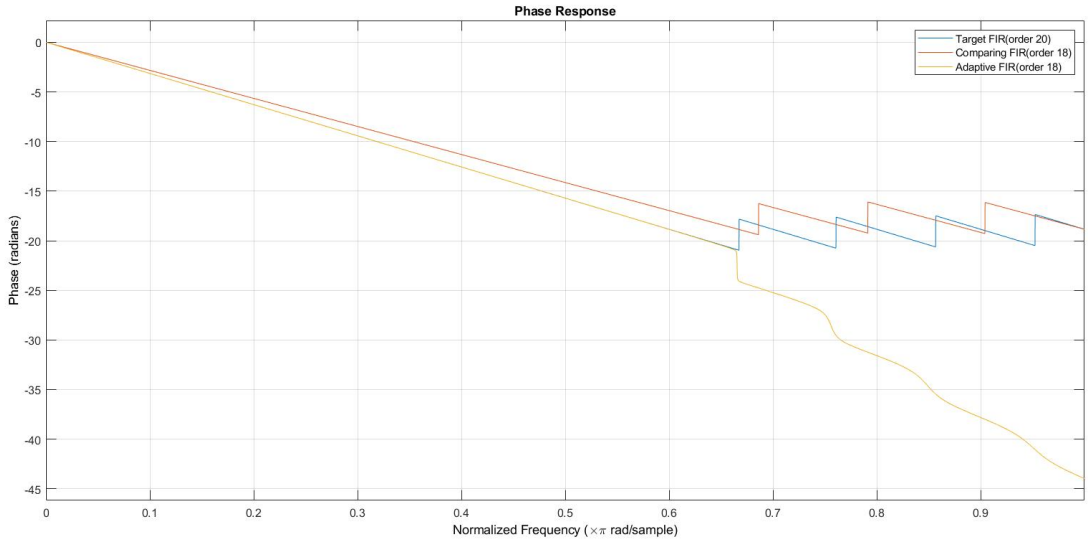


Figure 18: Phase responses of a Target lowpass Hann FIR filter of order 20 ,a comparing and adaptive FIR filter of order 18 using step size 0.01.

Now, if the variance of $\mathbf{d}$ is a lot, the steady state approximate MSE would also increase which is what we see in fig. 20.

## PART 4:An IIR target far exceeding the edge:

In this section, we deal with a high order IIR target filter using the same length adaptive FIR filter. We are going to be discussing two types of such filters:

### Butterworth IIR Target Filter

We use two different order butterworth IIR filters: 50 and 14. The target butterworth IIR filters differ only in order and have the following specifications:

```
%Bandpass filter using butterworth iir method.
%Filter order 50 or 14
%Fc1 = 0.3 rad/s (normalized)
%Fc2 = 0.7 rad/s (normalized)
```

This is generated using the "butter" function:

```
%making the filter
% b and a are the numerator and denominator coefficients vector.
[b,a] = butter(25,[0.3 0.7],'bandpass');
%lower order bandapss butterworth iir ( order 14)
[b2,a2] = butter(7,[0.3 0.7],'bandpass');
```

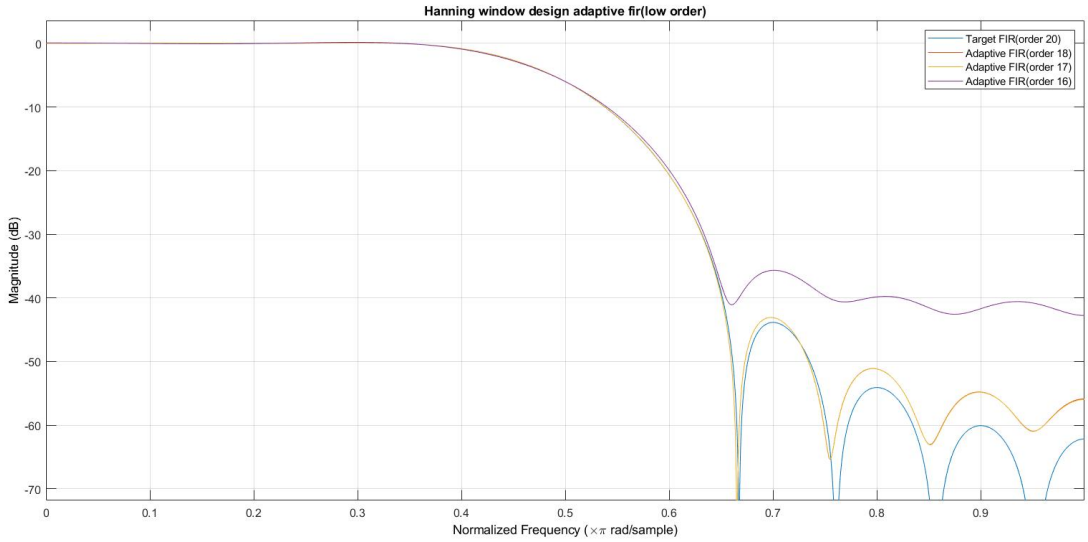


Figure 19: Comparison in magnitude responses of Target Hann FIR (order 20) with Adaptive FIR filters of orders-18,17,16 (using step size 0.01).

We now use the NLMS algorithm (step size 0.01 and trained for 20s) to produce the coefficients of the Adaptive FIR filter (order 98) separately for order 50 and 14. The magnitude responses are plotted using "fvtool" as shown in fig. 21 and 22. The following code illustrates this:

```
%h = fvtool(b,a,b_est,1);
%b,a are numerator and denominator coefficients
%of the original butterworth 50th order IIR filter.
%b_est are the coefficients of the adaptive FIR filter.
%similarly for the 14th order filter
h2 = fvtool(b2,a2,b_est2,1);
```

We use two orders in an attempt to find the order of the original IIR filter at which an Adaptive FIR filter of order 98 can almost converge to the original filter magnitude response. As the order increases from 14, the results keep diverging. To prove this, we find the adaptive FIR filter for an 18th order Butterworth IIR target as well. This is shown in fig.23.

### Chebyshev Type-1 IIR Target Filter

Now, we want to see how the results vary with a different type of filter. So, we use the Bandpass Chebyshev Type-1 IIR target filter which is known for its high roll-off.

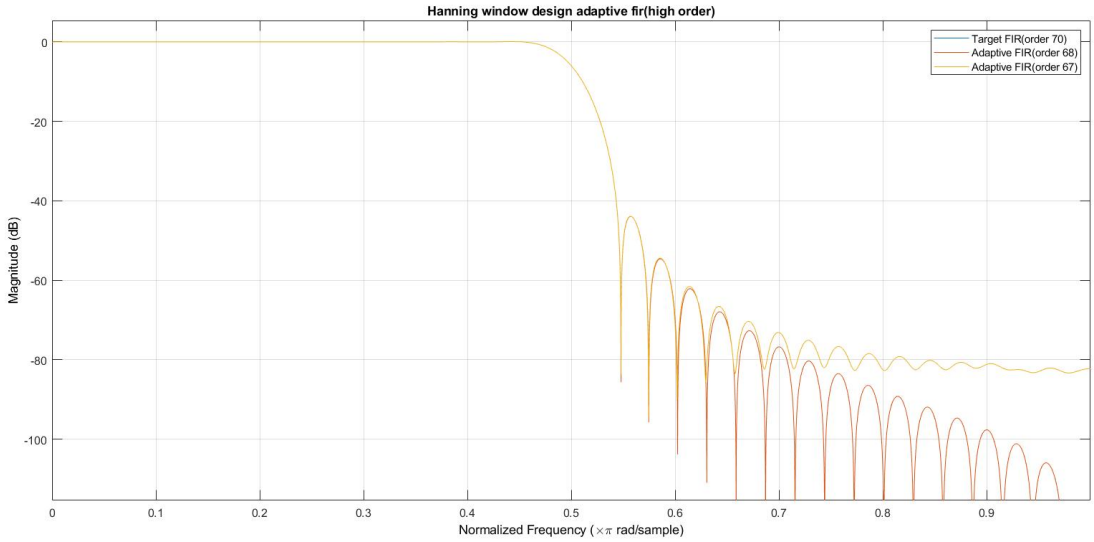


Figure 20: Magnitude responses of Target Hann FIR filter (order 70) and Adaptive FIR filters of orders 68 and 67 using step size 0.01.

The filter specifications are as follows:

```
% Bandpass IIR Chebyshev Type I
% Freq specs-(0.3 - 0.7) (normalized *pi freq).
%Filter order 10
%1dB passband ripple
```

This filter is generated using the "cheby1" function:

```
[b3,a3] = cheby1(5,1,[0.3 0.7]);
```

After this, as usual we find the adaptive FIR coefficients using the NLMS algorithm (step size 0.01 and trained for 20s) and plot the magnitude responses using "fvtool" as shown in fig. 24.

**Observations and Conclusions on Graphs:**

- From fig. 21,22 and 23, we can see that the magnitude responses of the adaptive FIR filter (length 99) diverges more as the order of the target IIR butterworth filter increases from 14 to 18 to 50. This is because the discrete IIR transfer function polynomial as seen earlier can be written as an infinite order polynomial after expanding the inverse of the denominator. So, as the order of the IIR filter increases, approximating it as

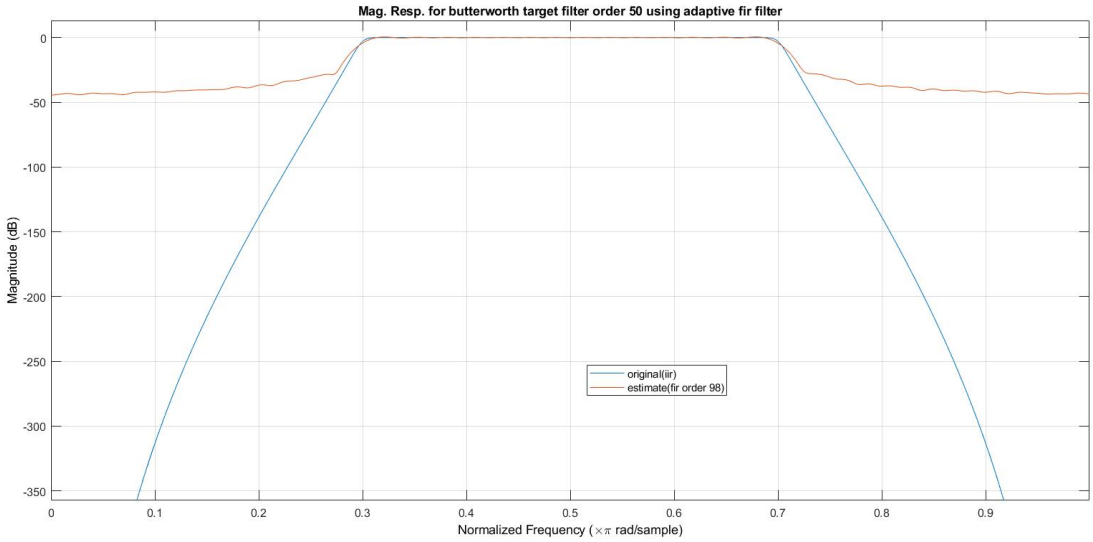


Figure 21: Magnitude responses of target butterworth IIR filter (order 50) and an FIR Adaptive Filter(length 99).

the same 98th order FIR filter would lead to more error. We saw earlier that for a 10th order IIR filter, we need an FIR length of 101 for convergence; additionally, its seen in fig. 22 that only for a 14th order IIR target, we get reasonable convergence in magnitude response. Hence, for a 50th order IIR target, the length of FIR filter needed would be much more or sometimes it would be impossible to form a stable response( not diverging). Therefore, we get a more diverging response as the target filter order increases.

- In fig. 24, we can see that the FIR adaptive Filter (length 99) diverges even for a target filter order of 10. This is mainly because it's a Chebyshev-Type 1 IIR filter with high roll-off (steepness in the transient band). This is because high roll-off/ increased steepness in the magnitude response with frequency means increased variance in the d(i) (desired output). From eqn. (5) we have seen that the approximate steady state error would be more if the variance in $\mathbf{d}$ is more. This is why there's divergence for a Chebyshev-Type 1 IIR target even at a small order of 10.

- On the practical side, whenever programming the FPGA for an IIR response using the FIR structure, we can use a lower order target IIR for the given specifications. Additionally, we can make educated guesses about the target order based on roll-off/steepness/ripple of the original frequency response.

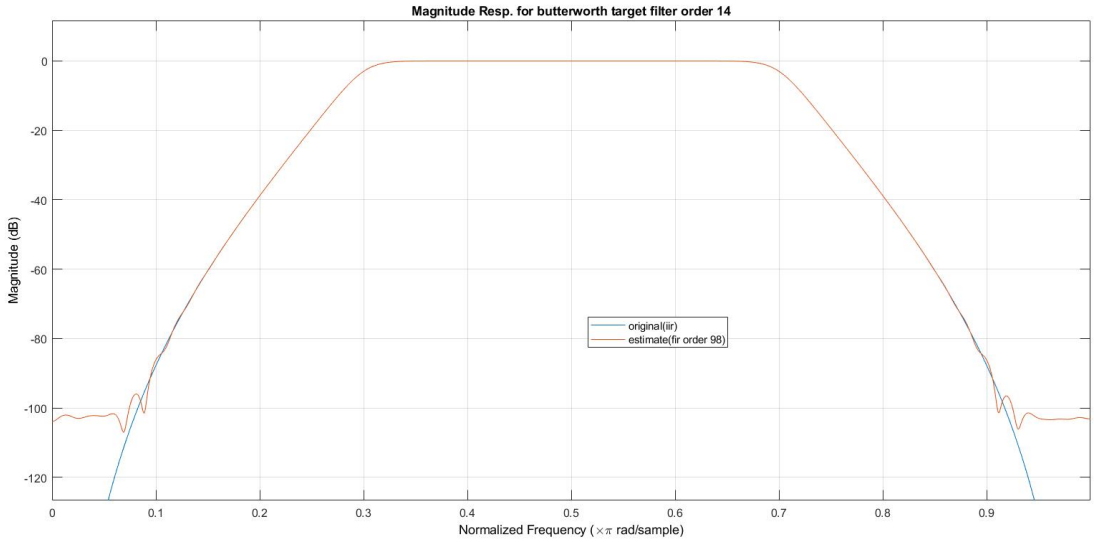


Figure 22: Magnitude responses of target butterworth IIR filter (order 14) and an FIR Adaptive Filter(length 99).

## PART 5:Analog Passive Filter Target

So far, we have seen adapting with different IIR and FIR target filters. Now, we try to estimate an Adaptive FIR filter using a lowpass butterworth analog filter in Simulink. The specifications are as follows:

```
Response: Lowpass
Design method: Butterworth
Filter Order: 8
Passband Edge Freq. (rad/s): 12560(=2*3.14*2000Hz)
```

The responses of an analog IIR filter and it's digital counterpart are similar and the transformation is usually done via Bilinear Transformation or Impulse invariance. Usually digital IIR filter would be built using the analog filter as a prototype.
To adapt a digital FIR Adaptive filter of length 99, we need to put an A/D convertor between the output of the analog filter and the desired signal input port of the NLMS adaptive filter. The experiment is done at a sampling rate of 8000Hz for 20s using a step size of 0.01.
The "Bandlimited White Noise" block is used to generate 1 PSD white noise (with sample rate 8000Hz) as an input to the butterworth filter. Simulink holds the white noise sample values for a sampling time period. The "Rate Transition" block is used as an A/D converter to sample continuous white noise (@ 8Khz) which is then fed to the input signal port of the NLMS adaptive filter.

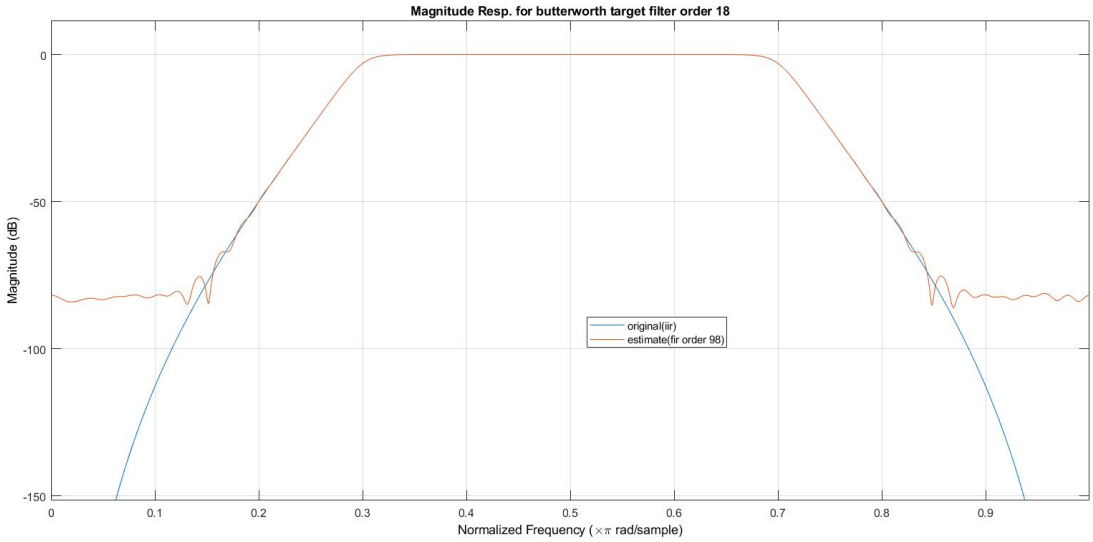


Figure 23: Magnitude responses of target butterworth IIR filter (order 18) and an FIR Adaptive Filter(length 99) .

The output of the butterworth filter is also sampled using this block and given to the desired signal port of the adaptive filter. Finally the weights are fed to an "Array plot" and given back to workspace. The model is shown in fig. 25. Finally, in MATLAB, we generate the Target Analog transfer function using "butter" and "zp2tf" functions and find it's response using "freqs". The estimated weights of the Adaptive FIR received from the SIMULINK model are used in "freqz" to find it's frequency response. Both the responses are plotted as shown in fig.26. The following code illustrates all this.

```
%order= 8
%fc = 2000 Hz (cutoff freq.)
%fs = 8000Hz (sampling rate.)
[zb,pb,kb]= butter(order,fc*2*pi,'s');
[bb,ab] = zp2tf(zb,pb,kb);
%original
[hb,wb] = freqs(bb,ab,4096);
%estimate
[hb2,wb2] = freqz(wts_est,1,'whole',4096);
%estimate
semilogy(wb2*fs/(2*pi),abs(hb2))
hold on
%original
semilogy(wb/(2*pi),abs(hb))
```

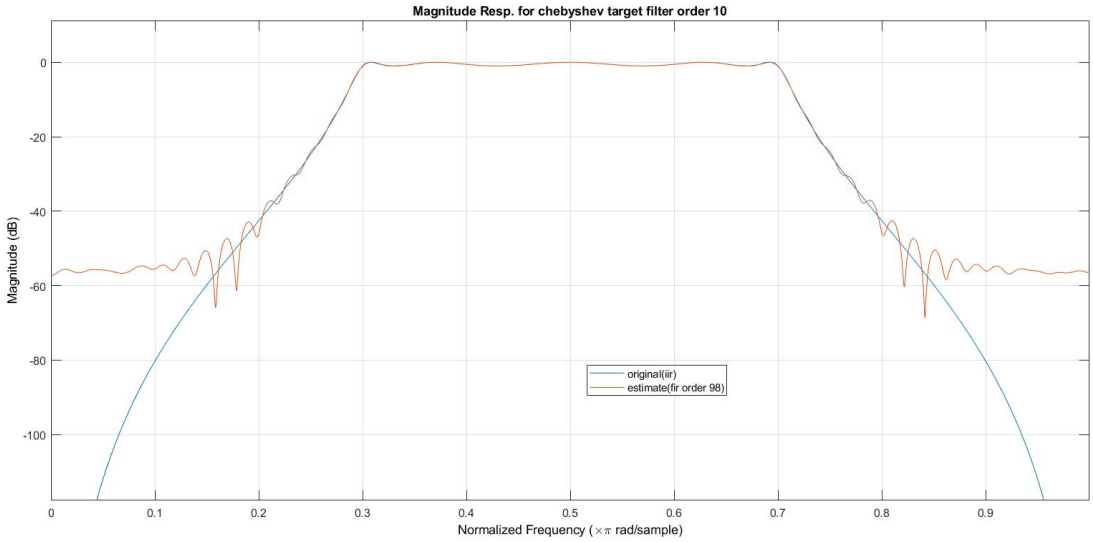


Figure 24: Magnitude responses of target Chebyshev-Type 1 IIR filter (order 10) and an FIR Adaptive Filter(length 99) .

**Observations and Conclusions on Graphs:**

- From fig. 26, we can see that the magnitude response of the Adaptive FIR estimated filter(length 99) is close to that of the Target analog lowpass IIR butterworth filter(order 8). We saw earlier that estimating a length 99 Adaptive FIR filter for a target digital IIR filter (order 10) gave almost full convergence in the magnitude response. However, a lower order analog IIR filter doesn't shown this. An intuitive explanation for this is that, fundamentally in a sense we are trying to estimate the Adaptive FIR filter for a target digital IIR filter whose prototype is the original analog IIR counterpart given. This makes sense since we can assume the analog butterworth filter and the A/D converter in the output mimics a digital IIR filter (referring to fig. 25) whose output is fed to the desired signal port of the NLMS filter. The discretized white noise would be inputs to the hypothetical digital IIR filter and the NLMS filter. Although there is no sure way of knowing this digital filter's transfer function, we can assume some transformation like the bilinear transform/ impulse invariance between the s-plane(analog) to z-plane (digital). Impulse invariance usually gives aliasing near nyquist frequencies causing magnitude to rise which isn't the case here(in fig. 26 the analog magnitude response is above the Adaptive FIR's/ hypothetical digital IIR). With bilinear transform however we have high roll off in the digital counterparts and deviations close to nyquist frequencies(as can be seen in fig. 26). Hence, the adaptive

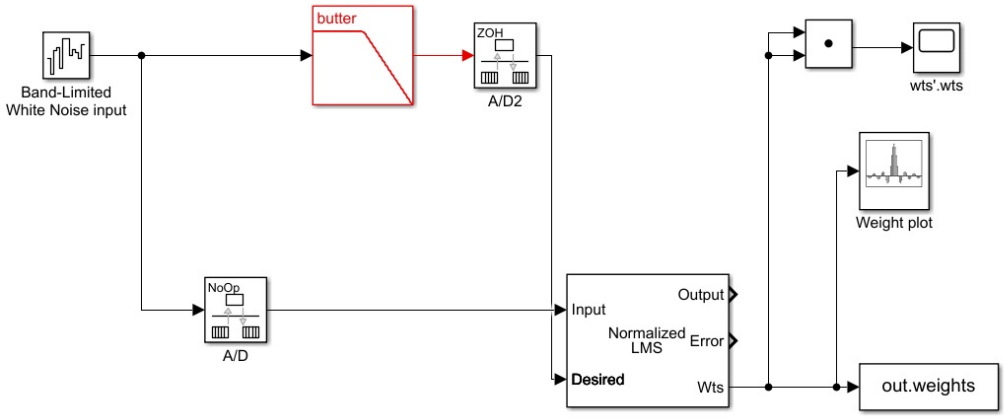


Figure 25: SIMULINK Model for estimating an Adaptive FIR using an analog Butterworth IIR lowpass filter(order 8)

FIR filter tries to make it's response close to the hypothetical digital IIR response which differs from the original analog IIR response if we design it via Bilinear transformation. Therefore,analog filters fundamentally can't be exactly replicated via Adaptive FIR filters although we get a close enough response.

- The dot product of the weight vector converges as shown in fig. 27 showing convergence of the NLMS adaptive filter.
- On the practical side, this shows that we can replicate magnitude responses even for analog IIR filters with low order to a fairly resasonable extent.

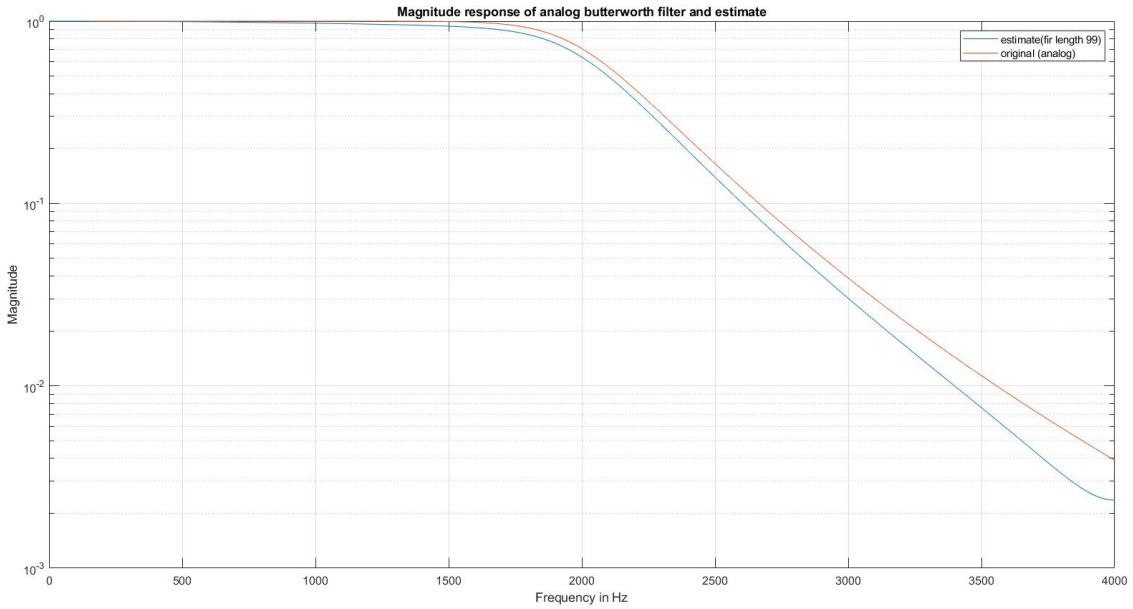


Figure 26: Magnitude responses of Target IIR analog butterworth filter(order 8) and Adaptive FIR filter of length 99(using step size 0.01) sampled at 8Khz.

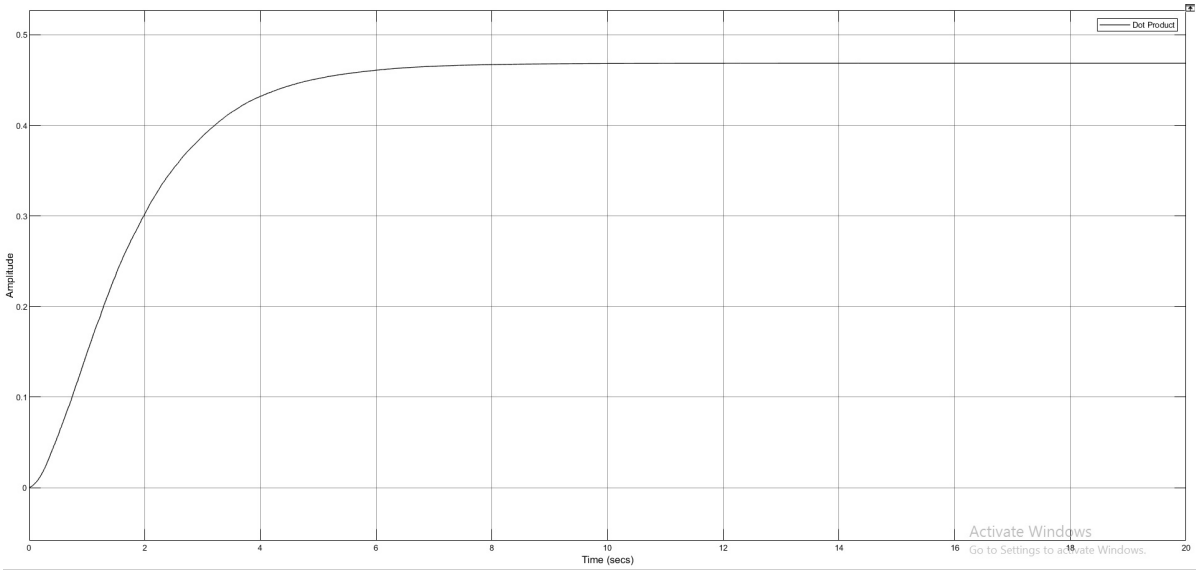


Figure 27: Plot of dot product of the estimated FIR weight vector with time.