Reference Matlab codes

PART 1: FIR filter target with not a larger length

```
time = 20;
N=8000*time; %  @ 8kHz sampling rate
power =1;
noise = wgn(N,1,power);
Fs = 8000;
periodogram(noise)
grid on
title('Periodogram of input signal')
xlabel('Frequency(normalized *pi)')
ylabel('Power/Frequency (dB/Hz)')

%FIR target with not a larger length
%Multiband filter designed by equiripple FIR.

order = 98;

%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];

b =firpm(order,f,a,w);
'coeffs'
transpose(b)


%fir filter
```

```
%feeding input signal to Target filter and getting output d(n)

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

zer1 = zeros(1,order);
x1 = [zer1, x1];

%'A check'

%for k = 1:length(d)

   % c= k+order; %index for x
   % d(k)-(flip(x(c-order:c)))*transpose(b)
%end

% found to be approximately same.

%NLMS algorithm

[wi A] = nlms(x1, d1, order);

%now we have the estimated weights



%estimate
b_est = wi
h = fvtool(b_est,1,b,1);
h1 = fvtool(b_est,1,b,1);
legend (h,'estimate', 'original');
legend (h1,'estimate', 'original');
h1.Analysis='phase'

%plotting wi progress
t1 = linspace(0,time,length(A)) ;
figure;
plot(t1,A)
grid
title('progress of weight matrix')
xlabel('time')
ylabel('inner product of weight matrix')

%similarly we can test it out for other kinds of design methods, like the
%hamming window based multiband fir filter, or a kaiser window based low pass
```

```
%hamming window designed multiband filter of same order

order2 = 98;
low = 0.4;
bnd = [0.6, 0.9];
b2 = fir1(order2,[low,bnd]);

x2 = transpose(noise); % assuming a white noise input

d2 = filter(b2,1,x2);
%d is the desired output

zer2 = zeros(1,order2);
x2 = [zer2, x2];

[wi2 A2] = nlms(x2, d2, order2);
%estimate

b_est2 = wi2
h2 = fvtool(b_est2,1,b2,1);
h3 = fvtool(b_est2,1,b2,1);

legend (h2,'estimate', 'original');
legend (h3,'estimate', 'original');
h2.Analysis='Magnitude'

h3.Analysis='phase'
title('Phase via Hamming window')


%kaiser
%making a low pass filter of order 108 via kaiser window
fcuts = [1000 1500];
mags = [1 0];
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.
[n,Wn,beta,ftype] = kaiserord(fcuts,mags,devs,3*Fs);
hh = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
order3 = n;
b3 = hh;
%feeding input signal to Target filter and getting output d(n)

x3= transpose(noise); % assuming a white noise input
```

```
d3 = filter(b3,1,x3);
%d is the desired output

zer3 = zeros(1,order3);
x3 = [zer3, x3];

[wi3 A3] = nlms(x3, d3, order3);

%now we have the estimated weights

%estimate
b_est3 = wi3
h4 = fvtool(b_est3,1,b3,1);
h5 = fvtool(b_est3,1,b3,1);
legend (h4,'estimate', 'original');
legend (h5,'estimate', 'original');
h4.Analysis='Magnitude'
h5.Analysis='phase'
title('Phase via Kaiser window method')



%NLMS algorithm
function [w_out,A_out] = nlms(x,d,order)

size(x)
size(d)
wi = (zeros(order+1,1));  %weight vector initially zero
size(wi)
mew=0.01;            %step size
eps = 0.0001;        % epsilon chosen as a small positive parameter
%ei = 1;
A=[];
for i= 1 :length(d)
    di = d(i); %at time i
    c = i+order; %index for x
    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
    ei;
    A= [A wi'*wi];
end
w_out = wi;
A_out = A;
```

```
end
```

PART 2:A lower order IIR filter Target:

```
time = 20;
N=8000*time; % 20s duration @ 8kHz sampling rate
power =1;
%training input data
noise = wgn(N,1,power);
Fs = 8000;

%Lower order IIR Filter target.
%Bandpass using butterworth iir method.
%Filter order 10
%Fc1 = 0.3 rad/s (normalized)
%Fc2 = 0.7 rad/s (normalized)


%NLMS Fir adaptive filter length 99
order =  98;
order2 =  100;
%making the filter
[b,a] = butter(5,[0.3 0.7],'bandpass');

wi_f = zeros(order+1,1);
%for changed mew
wi_fnew = zeros(order+1,1);
wi_f2 = zeros(order2+1,1);

%step size
mew = 0.01;

'coeffs orig.'
'num'
b
'den'
a

% 1 experiment

% assuming a white noise input
noise = wgn(N,1,power);

x1 = transpose(noise);
```

```
%d is the desired output
d1 = filter(b,a,x1);

zer1 = zeros(1,order);
zer2 = zeros(1,order2);

%zero padding both inputs
x2 = [zer2, x1];
x1 = [zer1, x1];

%esimtating
[wi A] = nlms(x1, d1, order,mew);
wi_f= wi+wi_f;

[wi2 A2] = nlms(x2, d1, order2,mew);
wi_f2= wi2+wi_f2;

%changed mew
mew2 = 0.001;
%1 experiment
exps=1;
for L=1:exps
    %Using 10 times the training data size.
noise2 = wgn(10*N,1,1*power);
x_new = transpose(noise2);
d_new = filter(b,a,x_new);
x_new = [zer1, x_new];
[wi_new A_new ]= nlms(x_new,d_new,order,mew2);
wi_fnew = wi_new+wi_fnew;
end



%estimate
b_est = wi_f;
b_est2 = wi_f2;
b_est_new = wi_fnew/exps;

'est coeffs fir'
wi_f
%plotting
h = fvtool(b,a,b_est,1);
title('Magnitude Resp. for Fir filter length 99')
h1 = fvtool(b,a,b_est,1);
legend (h,'original', 'estimate')
legend (h1,'original', 'estimate')
h1.Analysis='phase'
title('Phase Resp. for Fir filter length 99')

%phase






%The group delay seems to be the same, hence the phase is just off by some
%constant (approximately 2*pi) which means the phase is having a very low
%error actually.
%The graphs are different because of phase errors at dc and at normalized
%freq of 1 as the phase is more sensitive at these frequencies due to error
%in coeffs.

%Now we can see how to get the phase closer to the desired phase response
%by increasing the order

h2 = fvtool(b,a,b_est2,1);
title('Magnitude Resp. for Fir filter length 101')

h3 = fvtool(b,a,b_est2,1);

legend (h2,'original', 'estimate')
legend (h3,'original', 'estimate')
h3.Analysis='phase'
title('Phase Resp. for Fir filter length 101')

%for order = 100 phase fits almost perfectly for most experiments. Hence,
%changing order is a good way to improvise the performance at a low cost.
%Here, we just tweaked the order by 1.

%Changed mew
%plotting
h6 = fvtool(b,a,b_est_new,1);
title('Magnitude Resp. for Fir filter length 99(changed mew)')
h7 = fvtool(b,a,b_est_new,1);
legend (h6,'original', 'estimate')
legend (h7,'original', 'estimate')
h7.Analysis='phase'
title('Phase Resp. for Fir filter length 99(changed mew)')
%we can see for this, even with 10 times the training data and decreasing step
%size, assuming it has reached steady state the mag. and phase is diverging
%in the ends of the spectrum. Hence, tweaking step size isn't a good way to
```

```
%improvise the response.



%An alternative way of predicting lower order iir response is
%to model the adaptive filter as iir itself. This would require length of
%numerator and denominator weight vectors( b and a).
%we compare this and the fir case.

noise = wgn(N,1,power);
x3 = transpose(noise);

%d is the desired output
%d is same
d3 = filter(b,a,x3);

%order to be sent: order = length(b) + length(a)-1
filt_len3 = length(b) + length(a) -1;

%zero padding for initial samples
x3 = [ zeros(1,length(b)) x3];
d3 = [ zeros(1,length(a)) d3];

%Here the input regressors would be different, so we define the nlms
%algorithm separately here.
wi_3 = (zeros(filt_len3,1));  %weight vector initially zero
%size(wi);
mew3=0.01;            %step size
eps3 = 0.0001;        % epsilon chosen as a small positive parameter
A3=[];

for i= 1 :length(d3)-length(a)


    %extracting ui of length b
    c = i+length(b)-1; %index for x
    ui = flip(x3(c-(length(b)-1):c));

    %extracting 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);

    %concatenating di_block ( length = length(den.) -1) and ui till len(num.)
    ui_net = [ui di_block];


    %a check
    %'check'
    %d_latest - ui_net* transpose([ (b) (-a(2:end))]) %error


    % so the resulting weight vector would be
    % [b1 b2 .....bm -a2 -a3....-an]
    %where m is num. length , n is den. length
    %a1 is 1 and we include d(n) corresponding to a1 in the nlms recursion.

    ei3 = d_latest - ui_net*wi_3; %error
    wi_3 = wi_3 + (mew3/(eps3 + ui_net*ui_net'))* ui_net'* ei3; %estimating weights
    %ei3
    A3= [A3 wi_3'*wi_3];
end
w_out = wi_3;
A_out = A3;

'final est coeffs est.'

'numerator'
numf= transpose(wi_3(1:length(b)))
'actual num'
b

'denominator'
denf = [ 1 -transpose(wi_3(length(b)+1:end))]

'actual den'
a

h4 = fvtool(b,a,numf,denf);
title('Magnitude Resp. for adaptive iir model')

h5 = fvtool(b,a,numf,denf);

legend (h4,'original', 'estimate')
legend (h5,'original', 'estimate')
h5.Analysis='phase'

figure
t1 = linspace(0,time,length(A3)) ;
plot(t1,A3)
```

```
grid on
title('Progress of weight vector in Adaptive IIR Model')
xlabel('Time')
ylabel('Squared norm of weight vector')
%Hence, we see that even with modelling the iir filter as exactly with
%separate 'a', 'b' coeffs. We dont get the performance of the fir case with
% order 98 or 100
%intuitively this is because we involve di_block in the input regressors which
%isn't in our hands, while earlier we could generate ui and use a sliding
%window to give an estimate. Hence, the instantaneous approximation may not
%be so accurate in this case. Although it may converge better in the RLS
%algorithm.
```

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

%size(x)
%size(d)
wi = (zeros(order+1,1));  %weight vector initially zero
%size(wi);
%mew=0.01;            %step size
eps = 0.0001;        % epsilon chosen as a small positive parameter
ei = 1;
A=[];
for i= 1 :length(d)
    di = d(i); %at time i
    c = i+order; %index for x
    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
    ei;
    A= [A wi'*wi];
end
w_out = wi;
A_out = A;
end
```

PART 3:An FIR target with larger length:

```
time = 20;
N_samp=8000*time; % 20s duration @ 8kHz sampling rate
power =1;
%training input data
noise = wgn(N_samp,1,power);
Fs = 8000;

%Hann window lowpass filter
Fc = 0.5;
N = 20;
lowpass_hann = designfilt('lowpassfir','FilterOrder',N,'CutoffFrequency',Fc*Fs/2, ...
..'Window',hann(N+1),'SampleRate',Fs);
coeffs = lowpass_hann.Coefficients;
%parameters of comparing fir filter is same, order = 18;
%two different orders of adaptive and comparing filters
N_comp=18;
N_adap = 18;
lowpass_hann_comp = designfilt('lowpassfir','FilterOrder',N_comp,..
..'CutoffFrequency',Fc*Fs/2, 'Window',hann(N_comp+1),'SampleRate',Fs);
x1 = transpose(noise); % assuming a white noise input
%d is the desired output
d1 = filter(coeffs,1,x1);
zer1 = zeros(1,N_adap);
x1 = [zer1, x1];
%NLMS algorithm
mew1= 0.01;
[w1 A] = nlms(x1, d1, N_adap,mew1);
Hd_adap = dfilt.df1(w1,1);
%Plotting
hfvt = fvtool(lowpass_hann,lowpass_hann_comp,Hd_adap,'Color','white')
legend('Target FIR(order 20)','Comparing FIR(order 18)','Adaptive FIR(order 18)')
title('Hanning window design Adaptive FIR')

%An observation noted is that with increasing order, the affect on
% stopband performance with slight changes in filter order is high.
%So this is experimented comparing for two cases
% original: 20,18 and new: 20 17
%original: 70 68 and new: 70 67

%order 20 target fir
lowpass_hann1 = lowpass_hann;
%order 18 adaptive fir
Hd_adap1 = Hd_adap;
```

```
%for order 17 adaptive fir
N_adap2 = 17;
N_adap5 = 16;
x2 = transpose(noise); % assuming a white noise input
x5 = transpose(noise); % assuming a white noise input
%coeffs are target fir (order 20) coeffs
d2 = filter(coeffs,1,x2);
d5 = filter(coeffs,1,x5);
x2 = [zeros(1,N_adap2) , x2];
x5 = [zeros(1,N_adap5) , x5];
%NLMS algorithm
mew1= 0.01;
[w2 A2] = nlms(x2, d2, N_adap2,mew1);
Hd_adap2 = dfilt.df1(w2,1);
[w5 A5] = nlms(x5, d5, N_adap5,mew1);
Hd_adap5 = dfilt.df1(w5,1);
%plotting
hfvt2 = fvtool(lowpass_hann1,Hd_adap1,Hd_adap2,Hd_adap5,'Color','white')
title('Hanning window design adaptive fir(low order)')
legend('Target FIR(order 20)','Adaptive FIR(order 18)','Adaptive FIR(order 17)',...
...'Adaptive FIR(order 16)')

%order 70 target fir
N2 = 70;
lowpass_hann2 = designfilt('lowpassfir','FilterOrder',N2...
..,'CutoffFrequency',Fc*Fs/2, 'Window',hann(N2+1),'SampleRate',Fs);
coeffs2 = lowpass_hann2.Coefficients;
%order 68 and 67 adaptive fir
N_adap3 = 68;
N_adap4 = 67;
x3 = transpose(noise); % assuming a white noise input
%coeffs are target fir (order 70) coeffs
d_70 = filter(coeffs2,1,x3);
x3_1 = [zeros(1,N_adap3) , x3];
x3_2 = [zeros(1,N_adap4) , x3];
%NLMS algorithm
mew1= 0.01;
[w3_1 A3_1] = nlms(x3_1, d_70, N_adap3,mew1);
Hd_adap3 = dfilt.df1(w3_1,1); %order 68

[w3_2 A3_2] = nlms(x3_2, d_70, N_adap4,mew1);
Hd_adap4 =dfilt.df1(w3_2,1); %order 67
%plotting
hfvt3 = fvtool(lowpass_hann2,Hd_adap3,Hd_adap4,'Color','white')
title('Hanning window design adaptive fir(high order)')
legend('Target FIR(order 70)','Adaptive FIR(order 68)','Adaptive FIR(order 67)')
```

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

wi = (zeros(order+1,1));  %weight vector initially zero
eps = 0.0001;        % epsilon chosen as a small positive parameter
ei = 1;
A=[];
for i= 1 :length(d)
    di = d(i); %at time i
    c = i+order; %index for x
    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
    ei;
    A= [A wi'*wi];
end
w_out = wi;
A_out = A;
end
```

PART 4:An IIR target far exceeding the edge

```
time = 20;
N=8000*time; % 20s duration @ 8kHz sampling rate
power =1;
%training input data
noise = wgn(N,1,power);
Fs = 8000;

%Lower order IIR Filter target.
%Bandpass using butterworth iir method.
%Filter order 50
%Fc1 = 0.3 rad/s (normalized)
%Fc2 = 0.7 rad/s (normalized)

%making the filter
[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');
%order 18 bandapss butterworth iir
[b4,a4] = butter(9,[0.3 0.7],'bandpass');
```

```
% Bandpass IIR Chebyshev Type I (0.3 - 0.7) (normalized *pi freq)
%same freq specs.
%Filter order 10
%1dB passband ripple
[b3,a3] = cheby1(5,1,[0.3 0.7]);


'coeffs orig.'
'num'
b
'den'
a


%We use an adaptive filter of order 98
order =  98;
%wi_f;
%wi_f2;
%wi_f3;

%step size
mew = 0.01;

%white input data
x1 = transpose(noise);

%d is the desired output
d1 = filter(b,a,x1);
d2 = filter(b2,a2,x1);
d4 = filter(b4,a4,x1);
d3 = filter(b3,a3,x1);

x1 = [zeros(1,order), x1];

%esimtating for 50th order target
[wi_f  A] = nlms(x1, d1, order,mew);

%lower order target-14
[wi_f2  A2] = nlms(x1, d2, order,mew);

%estimating for  18th order target
[wi_f4  A4] = nlms(x1, d4, order,mew);

%chebyshev filter
[wi_f3  A3] = nlms(x1, d3, order,mew);
```

```
%estimate
b_est = wi_f;
b_est2 = wi_f2;
b_est4 = wi_f4;
b_est3 = wi_f3;


%plotting
h = fvtool(b,a,b_est,1);
title('Mag. Resp. for butterworth target filter order 50 using adaptive fir filter')
legend (h,'original(iir)', 'estimate(fir order 98)')

%As we can see for a higher order iir, it becomes hard to implement via an fir
%filter. For the given frequency specifications, all we can do is to reduce
%the order of the target filter to get it to converge. After hit and trial,
%for an adaptive filter of order 98 it's seen it slowly starts diverging at
%target filter order 14. We saw earlier that it converged for order = 10
h2 = fvtool(b2,a2,b_est2,1);
title('Magnitude Resp. for butterworth target filter order 14')
legend (h2,'original(iir)', 'estimate(fir order 98)')

%To verify that response diverges as we increase order from 14
h4 = fvtool(b4,a4,b_est4,1);
title('Magnitude Resp. for butterworth target filter order 18')
legend (h4,'original(iir)', 'estimate(fir order 98)')




%Moreover, it depends on the kind of iir filter, for sharp filters like
%chebyshev-type 1 it starts diverging earlier itself( around order 10 whereas
% for butterworth iir order 10 we saw previously that the mag resp. converged )
h3 = fvtool(b3,a3,b_est3,1);
title('Magnitude Resp. for chebyshev target filter order 10')
legend (h3,'original(iir)', 'estimate(fir order 98)')




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

%size(x)
%size(d)
wi = (zeros(order+1,1));  %weight vector initially zero
%wi = (10^-10)*randn(order+1,1)
```

```
%size(wi);
%mew=0.01;            %step size
eps = 0.0001;        % epsilon chosen as a small positive parameter
ei = 1;
A=[];
for i= 1 :length(d)
    di = d(i); %at time i
    c = i+order; %index for x
    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
    ei;
    A= [A wi'*wi];
end
w_out = wi;
A_out = A;
end
```

PART 5: Analog passive filter-Code for plotting Magnitude responses:

```
simOut = sim('analog_adapt2','ReturnWorkspaceOutputs','on')
'predicted weights of the filter'
wts_est = simOut.weights

%Comparing bode responses of original and est.
%original
%order of analog lowpass butterworth= 8
order = 8;
fc = 2000;
fs = 8000;
[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);

%plotting
%estimate
```

```
semilogy(wb2*fs/(2*pi),abs(hb2))
hold on
%original
semilogy(wb/(2*pi),abs(hb))
grid
hold off
title('Magnitude response of analog butterworth filter and estimate')
ylabel('Magnitude')
xlabel('Frequency in Hz')
legend('estimate(fir length 99)','original (analog)')
xlim([0,4000])
```