UNIVERSITY OF TEXAS AT SAN ANTONIO

Project : Linear Filtering of Signals

EE-5163 DSP

Mohammadhafez Bazrafshan @01347682

3/19/2013

Contents

Preface:

In this project I have created 4 functions for different types of convolution, namely dirConv, fftConv, oaConv and osConv. Typing in help [function name] in MATLAB command window will explain how the functions work:

help dirConv

```
 DIRCONV  convolution using direct method
 C=DIRCONV(A,B) convolves row vectors A and B. The resulting vector is
 length LENGTH(A)+LENGTH(B)-1
 This function uses the direct method as the basis for convolution.
```

help fftConv

```
 FFTCONV  convolution using fast fourier transform
 C=FFTCONV(A,B) convolves row vectors A and B. The resulting vector is
 length LENGTH(A)+LENGTH(B)-1
 This function uses the FFT as the basis for convolution.
```

help oaConv

```
 OACONV  convolution using overlap and add method
 C=OACONV(A,B,BLEN,MOD) performs convolution on A and B using overlap and
 and add method.  BLEN is the block length designated for convolution. BLEN
 should not be greater than the length of both signals.
 MOD is a string that designates which method of convolution should be performed
 in a block convolution.  MOD='direct' uses the direct method and MOD='fft' uses
 the fft.
The resulting vector is length LENGTH(A)+LENGTH(B)-1
```

help osConv

```
 OSCONV  convolution using overlap and save method
 C=OSCONV(A,B,BLEN,MOD) performs convolution on A and B using overlap and
 and save method.  BLEN is the block length designated for convolution. BLEN
 should not be greater than the length of both signals. In addition BLEN
 should at least be equal to length of the shorter signal - 1.
 (BLEN>=FLEN-1 and FLEN=min(length(A),length(B)).
 MOD is a string that designates which method of convolution should be performed
 in a block convolution.  MOD='direct' uses the direct method and MOD='fft' uses
 the fft.
 The resulting vector is length LENGTH(A)+LENGTH(B)-1
```

The details of these codes are provided in the appendix alongside with the m-file script for the different parts of the project. In order to test the performance of these functions I will show here some examples and will compare them to matlab built in convolution i.e **conv.** I will consider two signals as a simple example:

signal 1=[1 2 3 8 3 2 5 1]; signal 2=[6 3 4 7 9];

```
 conv(signal1,signal2)  %matlab answer
ans = [ 6    15    28    72    77    92   131   122    64    57    52     9];
```

dirConv(signal1,signal2)
ans=[6 15 28 72 77 92 131 122 64 57 52 9];

fftConv(signal1,signal2)
ans=[6.0000 15.0000 28.0000 72.0000 77.0000 92.0000 131.0000 122.0000 64.000 0 57.0000 52.0000 9.0000]

oaConv(signal1,signal2,**3**,**'fft'**)
ans
=[6.0000 15.0000 28.0000 72.0000 77.0000 92.0000 131.0000 122.0000 64.0000 57.0000 52.0000 9.0000];

oaConv(signal1,signal2,**4**,**'direct'**)
ans =[6 15 28 72 77 92 131 122 64 57 52 9];

osConv(signal1,signal2,**5**,**'fft'**)
ans=[6.0000 15.0000 28.0000 72.0000 77.0000 92.0000 131.0000 122.0000 64.000 0 57.0000 52.0000 9.0000];

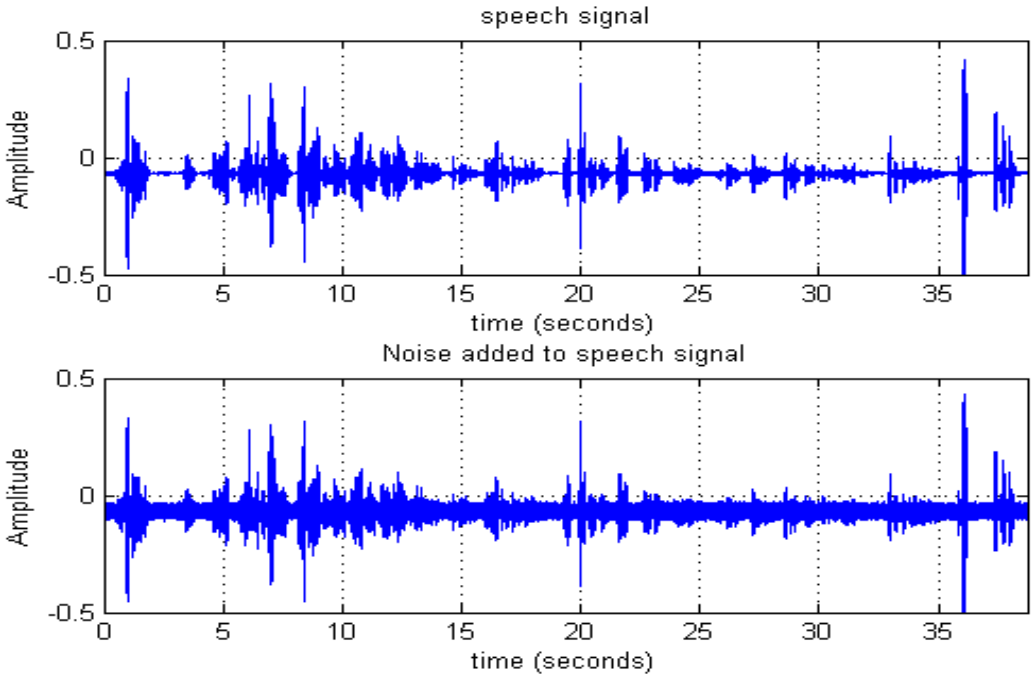
osConv(signal1,signal2,**5**,**'direct'**)
ans = [6 15 28 72 77 92 131 122 64 57 52 9];

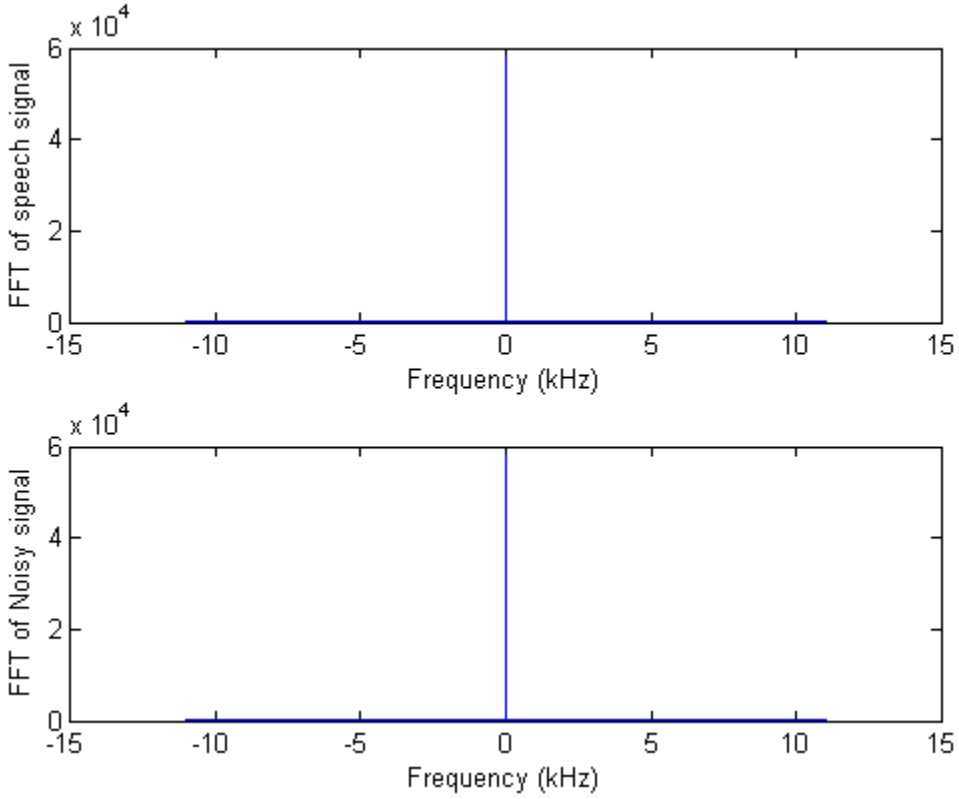
The functions have been evaluated and verified by other signal examples as well. They appear to be working. In oaConv and osConv functions, **If the signal length is not divisible by the block length the function automatically zero pads the signal.**

A second key note I would like to mention is about the zero index of a signal (origin of a signal). Consider the two signals below:
signal 1=[1 2 3 **8** 3 2 5 1]; signal 2=[6 **3** 4 7 9];
By convolving the two signals we will have an array of 11 elements (as shown above):
signal1*signal2=[6 15 28 72 **77** 92 131 122 64 57 52 9];
However it is important to be able to know where exactly is the zero index. It's relatively straightforward that the zero index of the convolution can be calculated using the following:
zeroIndexConv=zeroIndex1+zeroIndex2-1;

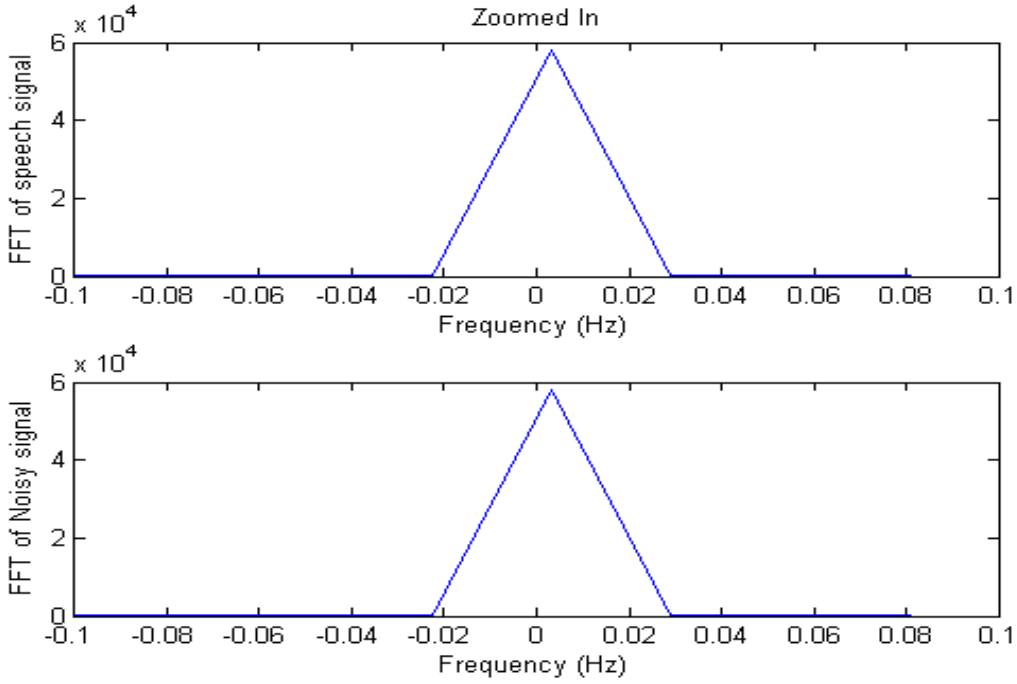
As the final note, I tried using row vectors and that's only because I'm more comfortable. My codes is not intended to work for column signals and also 2-D. Therefore after wavread command I have transposed the signal from the beginning to work with row vectors.

Parts 1 & 2: Reading Speech Signal and Adding Noise:

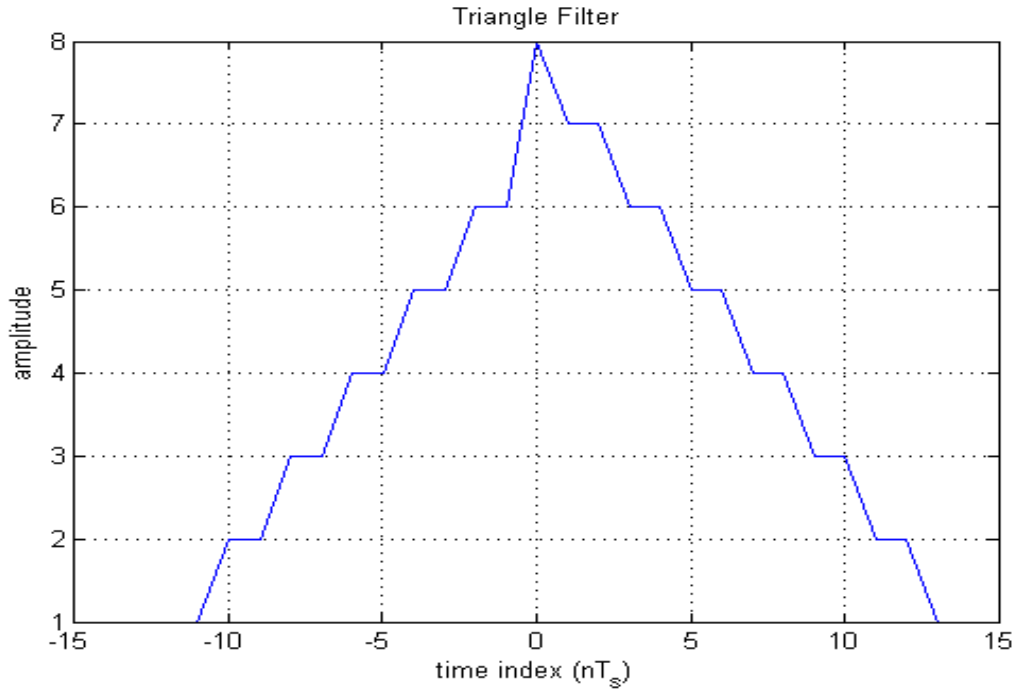


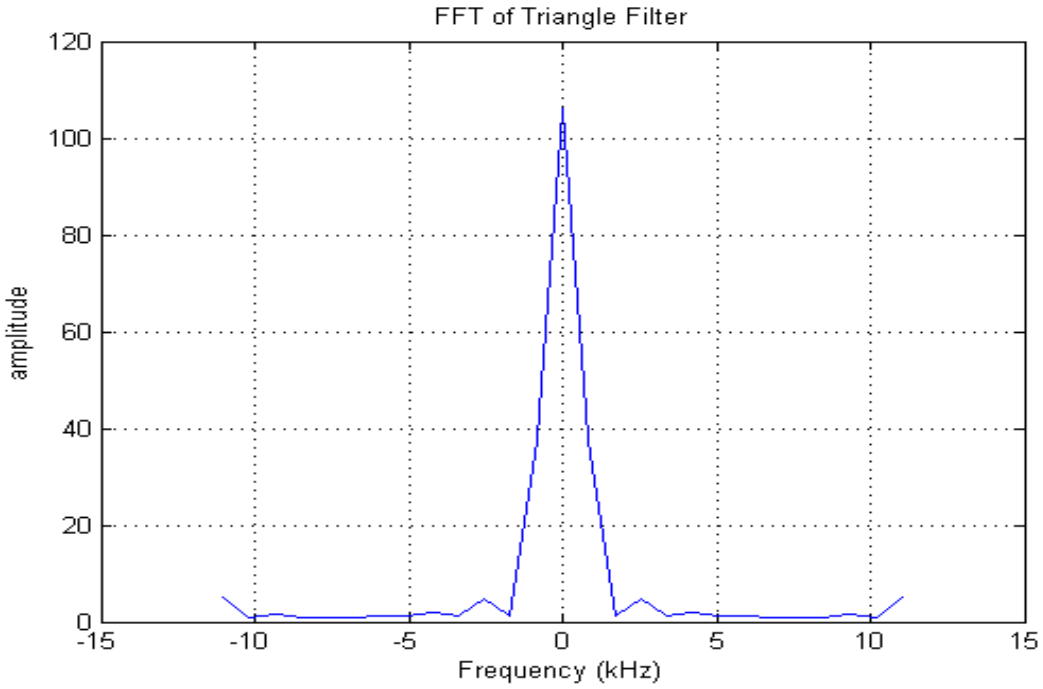


For a more zoomed in picture of the FFT of speech signal:

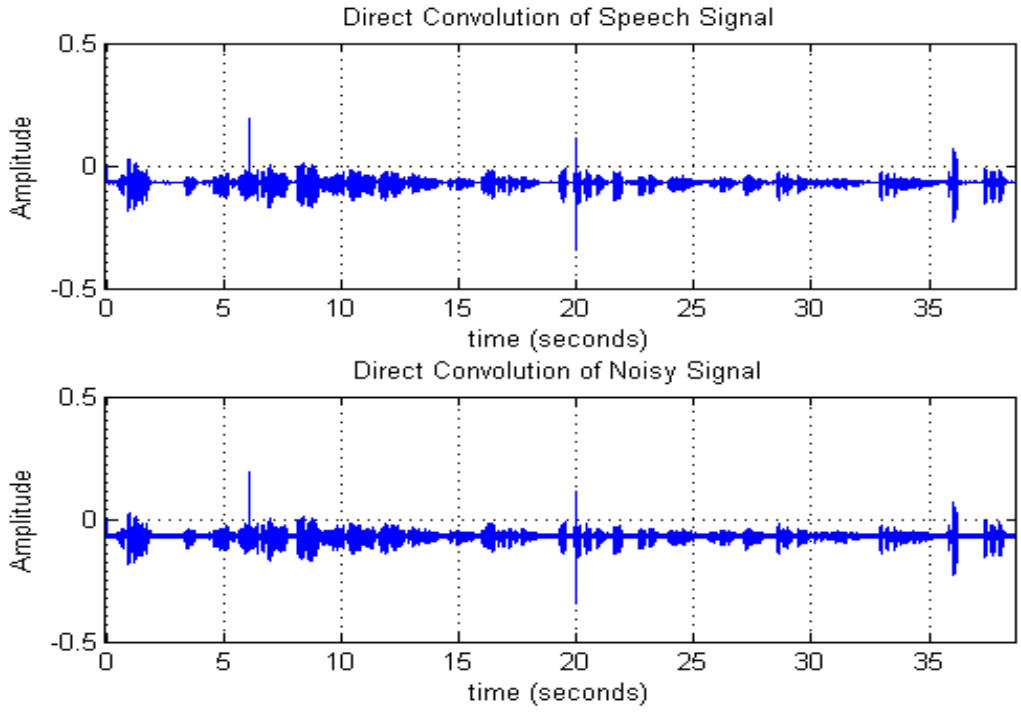


Part 3: The Triangle Filter (not normalized):

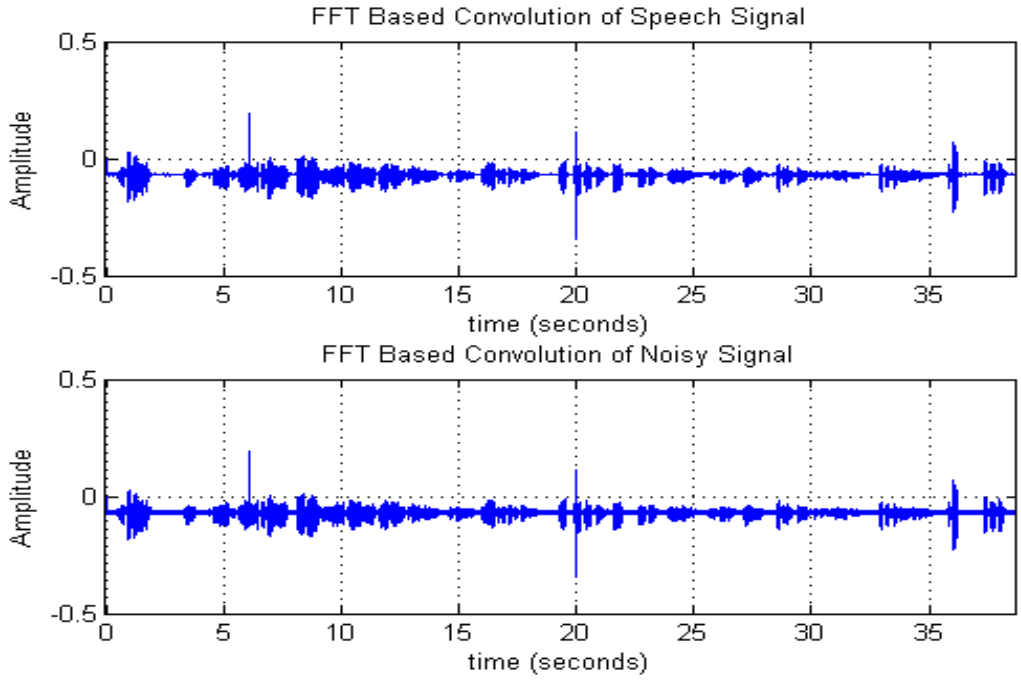



FFT of Triangle Filter
120
100
80
60
40
20
0
amplitude
-15
-10
-5
0
5
10
15
Frequency (kHz)

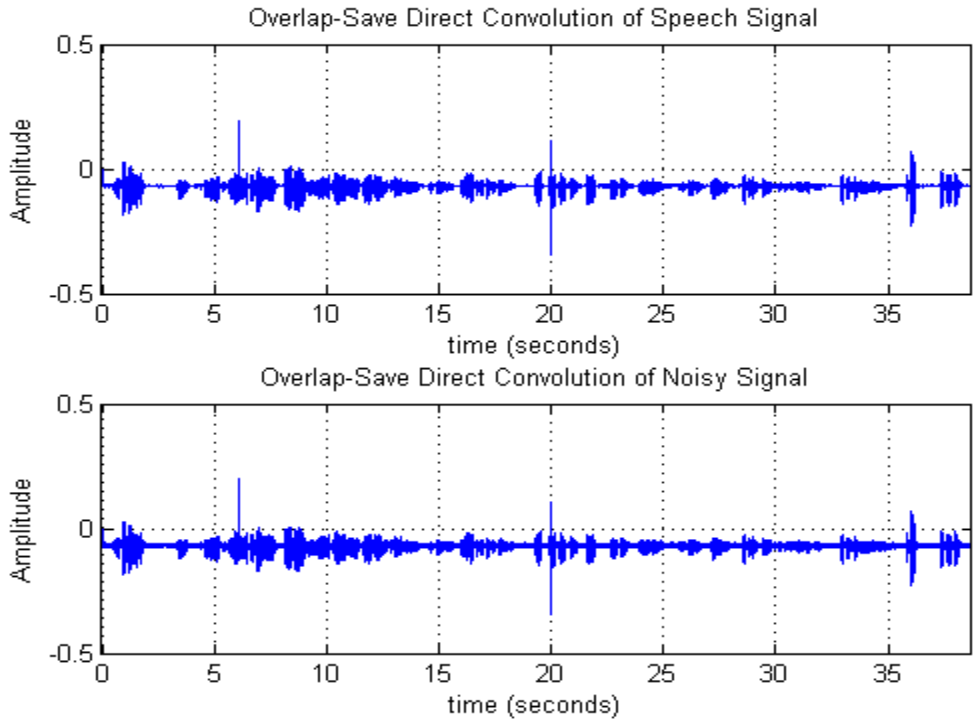
Part 4: Filtering of the Noisy Signal Using Direct Convolution:



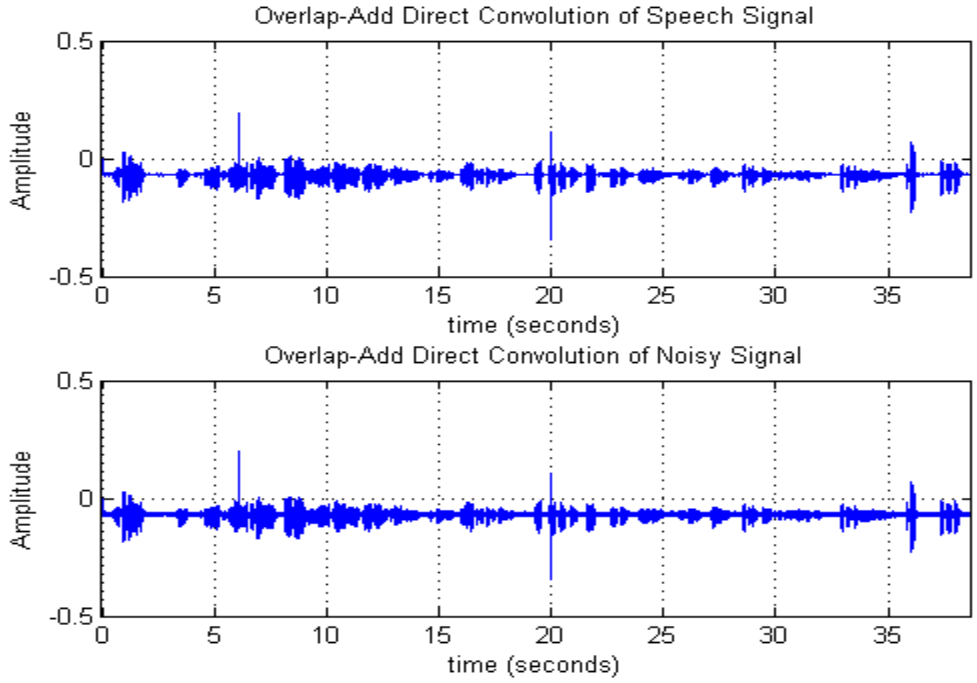
Part 5: Filtering of the Noisy Signal Using FFT:



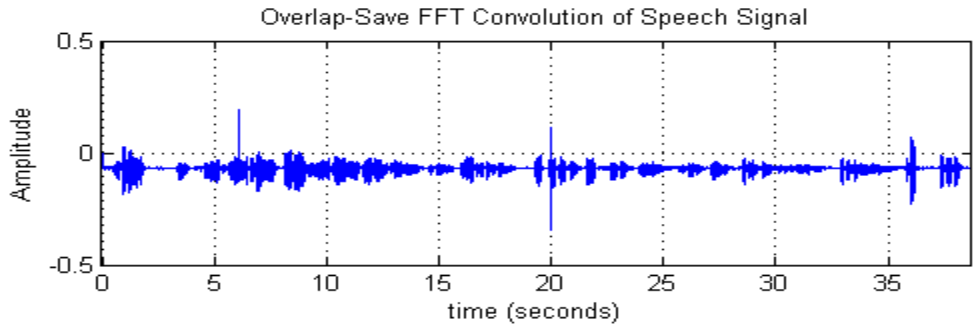
Part 6.1: Filtering Overlap and Save method Direct:

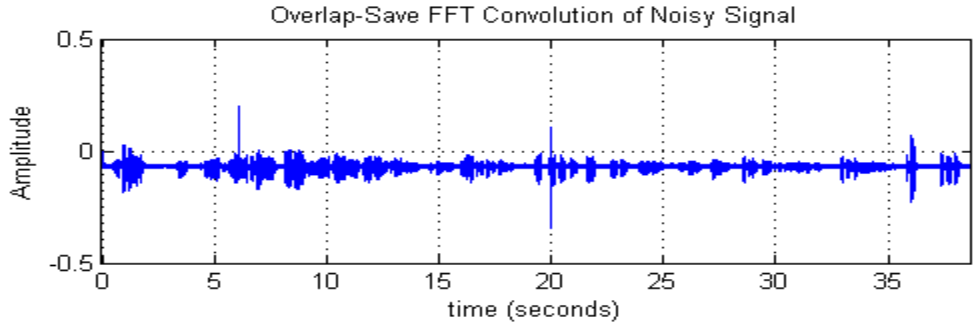


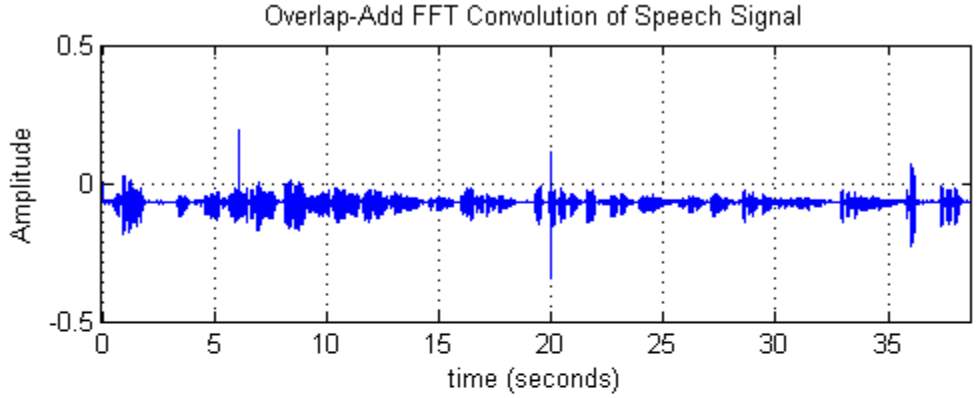
Part 6.2: Filtering Overlap and Add method Direct:

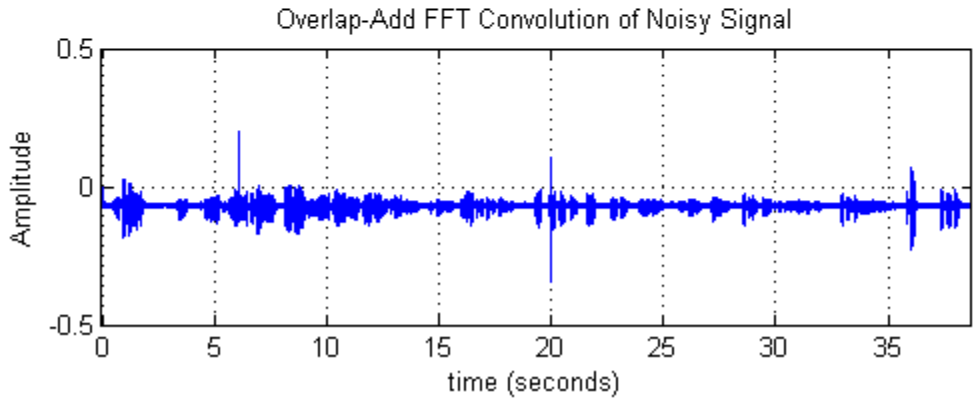


Part 6.3: Filtering Overlap and Save – Overlap and Add with FFT:









Errors and Time of Calculation:

The errors of filteration were calculated as the MRSE :

$$\frac{1}{N} \times \sqrt{\sum_{n} (fn - gn * hn)^2}$$

Remarking the second note in preface section, the zero index of $gn * hn$ starts from is the 13th element (because 13+1-1=13) and thus gn*hn(13:13+length(fn)-1) is cropped from the result of convolution to calculate the MRSE.

Using Matlab 2011 this error was calculated as 1.4102e-005 for all functions (dirConv, fftConv, osConv, oaConv). The built in Matlab function "conv" was also investigated and it had the same calculated error of 1.4102e-005.

```
% Using Matlab 2008 on My own Laptop:
% Direct Convolution: Elapsed time is 1847.949951 seconds.
% FFT Convolution: Elapsed time is 0.282791 seconds.
% Overlap-Save Convolution: Elapsed time is 131.300173 seconds.
% Overlap-Add Convolution: Elapsed time is 125.526493 seconds.
% Overlap-Save FFT Convolution: Elapsed time is 0.250491 seconds.
% Overlap-Add FFT Convolution: Elapsed time is 0.246442 seconds.
```

```
% Using Matlab 2011b on the lab computer:
% Direct Convolution: Elapsed time is 4.455936 seconds.
% FFT Convolution: Elapsed time is 0.326912 seconds.
% Overlap-Save Convolution: Elapsed time is 5.017626 seconds.
% Overlap-Add Convolution: Elapsed time is 4.974654 seconds.
% Overlap-Save FFT Convolution: Elapsed time is 0.370859 seconds.
% Overlap-Add FFT Convolution: Elapsed time is 0.357395 seconds.
% all errors for Matlab 2011:   1.4102e-005
```

The reason for the long time in my laptop is for the registry problems. I've also tried this code on a friend's laptop (beside my own laptop and the lab computer) and the elapsed time was as low as 5 seconds at most.

Appendix 1: Matlab m-file for running the code

```
%EE-5163, Digital Signal Processing, ART GRIGORYAN, UTSA-2013
%Project: Linear Filtering of Signals
%Student name: Mohammadhafez Bazrafshan
%Banner ID: @01347682
%Due March 7th, 2013. Extended to March 19th, 2013



clear all;
clear;
clc;
close all;



%***********************PART 1 reading the signal************************%
[fn,Fs,nbits]=wavread('mike.wav');
fn=fn'; %column vector to row vector (I'm more comfortable);
% Fs - sampling rate  22050 Hz
N=length(fn);        % 854128
N_insec = N/Fs;     % 38.7360 sec
% sound(X,Fs);
FN=abs(fft(fn));
% ************************************************************************%


% ***********************PART 2 NORMAL DISTRIBUTION**********************%
std=0.01; %standard deviation of noise
meaN=0;   %mean of Noise
nN=meaN+std*randn(1,N);   %generate random noise with normal distribution
gn=fn+nN;  %add noise to the original signal
GN=abs(fft(gn));
% ************************************************************************%
B=5;  %5 seconds partitioning
or1=1; %position of zero (origin) in gn

figure(1)
subplot(2,1,1);
plot(linspace(0,N_insec,N),fn)
grid on
xlabel('time (seconds)');
ylabel('Amplitude');
title('speech signal');
axis([0 N_insec -0.5 0.5]);
subplot(2,1,2);
plot(linspace(0,N_insec,N),gn);
grid on;
xlabel('time (seconds)');
ylabel('Amplitude');
title('Noise added to speech signal');
```

```
axis([0 N_insec -0.5 0.5]);

figure(2)
subplot(2,1,1);
fvec=linspace(-Fs/2,Fs/2,N)./1000;
plot(fvec,fftshift(FN));
xlabel('Frequency (kHz)');
ylabel('FFT of speech signal');

subplot(2,1,2);
fvec=linspace(-Fs/2,Fs/2,N)./1000;
plot(fvec,fftshift(GN));
xlabel('Frequency (kHz)');
ylabel('FFT of Noisy signal');

figure(3)
subplot(2,1,1);
fvec=[-0.1:Fs./N:0.1];
plot(fvec,fftshift(FN(1:length(fvec))));
xlabel('Frequency (Hz)');
ylabel('FFT of speech signal');
title('Zoomed In');
subplot(2,1,2);
fvec=[-0.1:Fs./N:0.1];
plot(fvec,fftshift(GN(1:length(fvec))));
xlabel('Frequency (Hz)');
ylabel('FFT of Noisy signal');

% ************************PART 3 TRIANGLE FILTER**************************%
hn=[1 1 2 2 3 3 4 4 5 5 6 6 8 7 7 6 6 5 5 4 4 3 3 2 2 1 1]; %triangle
NH=length(hn); % length of the filter
or2=13;  %position of zero(origin) in hn
NH_insec=NH./Fs;
figure(4);
plot([-12:14],hn,'-');
xlabel('time index (nT_s)');
ylabel('amplitude');
title('Triangle Filter');
grid on;

figure(5);
plot(linspace(-Fs./2,Fs./2,NH)./1000,abs(fftshift(fft(hn))));
title('FFT of Triangle Filter');
xlabel('Frequency (kHz)');
ylabel('amplitude');
grid on;
hn=hn./sum(hn); %normalizing hn; if we want to listen to sound
% ************************************************************************%

% ************************PART 4 Direct Convolution***********************%
yn4=zeros(1,length(fn)+length(hn)-1);
yn4=dirConv(fn,hn);
tic
ygn4=dirConv(gn,hn);
toc
%Elapsed time is 4.476943 seconds.
```

```
figure(6);
subplot(2,1,1);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(yn4)),yn4);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Direct Convolution of Speech Signal');
grid on;
subplot(2,1,2);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(ygn4)),ygn4);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Direct Convolution of Noisy Signal');
grid on;
dirErr=sqrt(sum((ygn4(or2:or2+N-1)-fn).^2))./N;


%**********************PART 5 FFT Based Convolution*********************%
yn5=fftConv(fn,hn);
tic
ygn5=fftConv(gn,hn);
toc
%Elapsed time is 0.323113 seconds.
figure(7);
subplot(2,1,1);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(yn5)),yn5);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('FFT Based Convolution of Speech Signal');
grid on;
subplot(2,1,2);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(ygn5)),ygn5);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('FFT Based Convolution of Noisy Signal');
grid on;
fftErr=sqrt(sum((ygn5(or2:or2+N-1)-fn).^2))./N;

%**********************PART 6.1 Overlap-Save Convolution****************%
blen=5*Fs;  %block length
yn61=osConv(fn,hn,blen,'direct');
tic
ygn61=osConv(gn,hn,blen,'direct');
toc
%elapsed time;
figure(8);
subplot(2,1,1);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(yn61)),yn61);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Save Direct Convolution of Speech Signal');
grid on;
```

```
subplot(2,1,2);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(ygn61)),ygn61);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Save Direct Convolution of Noisy Signal');
grid on;
osErr=sqrt(sum((ygn61(or2:or2+N-1)-fn).^2))./N;
```

```
%***********************PART 6.2 Overlap-Save Convolution*****************%
blen=5*Fs;  %block length
yn62=oaConv(fn,hn,blen,'direct');
tic
ygn62=oaConv(gn,hn,blen,'direct');
toc
%elapsed time;
figure(9);
subplot(2,1,1);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(yn62)),yn62);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Add Direct Convolution of Speech Signal');
grid on;
subplot(2,1,2);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(ygn62)),ygn62);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Add Direct Convolution of Noisy Signal');
grid on;
oaErr=sqrt(sum((ygn62(or2:or2+N-1)-fn).^2))./N;
```

```
%************PART 6.3 Overlap-Save and Overlap-Add FFT Convolution********%
yn63a=osConv(fn,hn,blen,'fft');
tic
ygn63a=osConv(gn,hn,blen,'fft');
toc
%elapsed time;
figure(10);
subplot(2,1,1);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(yn63a)),yn63a);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Save FFT Convolution of Speech Signal');
grid on;
subplot(2,1,2);
```

```
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(ygn63a)),ygn63a);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Save FFT Convolution of Noisy Signal');
grid on;
osfftErr=sqrt(sum((ygn63a(or2:or2+N-1)-fn).^2))./N;


yn63b=oaConv(fn,hn,blen,'fft');
tic
ygn63b=oaConv(gn,hn,blen,'fft');
toc
%elapsed time;
figure(11);
subplot(2,1,1);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(yn63b)),yn63b);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Add FFT Convolution of Speech Signal');
grid on;
subplot(2,1,2);
plot(linspace(-NH_insec/2,N_insec+NH_insec/2,length(ygn63b)),ygn63b);
axis([-NH_insec N_insec+NH_insec/2 -0.5 0.5]);
xlabel('time (seconds)');
ylabel('Amplitude');
title('Overlap-Add FFT Convolution of Noisy Signal');
grid on;
oafftErr=sqrt(sum((ygn63b(or2:or2+N-1)-fn).^2))./N;


%
% Elapsed time is 4.618433 seconds.
% Elapsed time is 0.346697 seconds.
% Elapsed time is 4.722118 seconds.
% Elapsed time is 5.087066 seconds.
% Elapsed time is 0.293950 seconds.
% Elapsed time is 0.376626 seconds.
```

Appendix 2: Functions

```
function sigConv=dirConv(sig1,sig2)
% DIRCONV  convolution using direct method
% C=DIRCONV(A,B) convolves row vectors A and B. The resulting vector is
% length LENGTH(A)+LENGTH(B)-1
% This function uses the direct method as the basis for convolution.
% Created by Mohammadhafez Bazrafshan
if nargin<2
    error('Input Arguments Not defined');
end
[A AA]=size(sig1);
[B BB]=size(sig2);
log1=(A~=1) & (AA~=1);
log2=(B~=1) & (BB~=1);
if  log1==1
    error('signal 1 should be one dimensional');
end
if  log2==1
    error('signal 2 should be one dimensional');

end

if length(sig1)>length(sig2)
    filter=sig2;
    signal=sig1;
else
    filter=sig1;
    signal=sig2;
end


flen=length(filter);
slen=length(signal);
clen=flen+slen-1;  %length of convolution

sigg=[zeros(1,flen-1) signal zeros(1,flen-1) zeros(1,flen-1)]; %zero pad
signal so that it
%doesn't run out of points while convolution
sigConv=zeros(1,clen);
filflip=fliplr(filter);
for i=flen:clen+flen-1
    sigConvv(i)=sum(sigg(i-flen+1:i).*filflip); %multiply by the flip filter
%     and shift
end
sigConv=sigConvv(flen:end);
```

```
function fConv=fftConv(sig1,sig2)
% FFTCONV  convolution using fast fourier transform
% C=FFTCONV(A,B) convolves row vectors A and B. The resulting vector is
% length LENGTH(A)+LENGTH(B)-1
% This function uses the FFT as the basis for convolution.
% Created by Mohammadhafez Bazrafshan
if nargin<2
    error('Input Arguments Not defined');
end
[A AA]=size(sig1);
[B BB]=size(sig2);
log1=(A~=1) & (AA~=1);
log2=(B~=1) & (BB~=1);
if  log1==1
    error('signal 1 should be one dimensional');
end
if  log2==1
    error('signal 2 should be one dimensional');

end

if length(sig1)>length(sig2)
    filter=sig2;
    signal=sig1;
else
    filter=sig1;
    signal=sig2;
end


flen=length(filter);
slen=length(signal);
clen=flen+slen-1;  %length of convolution

fConv=real(ifft(fft(signal,clen).*fft(filter,clen)));
```

```
function sConv=osConv(sig1,sig2,Blen,MOD)
% OSCONV  convolution using overlap and save method
% C=OSCONV(A,B,BLEN,MOD) performs convolution on A and B using overlap and
% and save method.  BLEN is the block length designated for convolution. BLEN
% should not be greater than the length of both signals. In addition BLEN
% should at least be equal to length of the shorter the signal - 1.
% (BLEN>=FLEN-1 and FLEN=min(length(A),length(B)).
% MOD is a string that designates which method of convolution should be
performed
% in a block convolution.  MOD='direct' uses the direct method and MOD='fft'
uses
% the fft.
%The resulting vector is length LENGTH(A)+LENGTH(B)-1
%Created by Mohammadhafez Bazrafshan
if nargin<2
    error('Signals Not defined');
end
if nargin<3
    error('Block Length Not defined');
end

if nargin<4
    error('Mode of conv not defined, ''direct'' for direct convolution and
''fft'' for fast');
end

[A AA]=size(sig1);
[B BB]=size(sig2);
log1=(A~=1) & (AA~=1);
log2=(B~=1) & (BB~=1);
if  log1==1
    error('signal 1 should be one dimensional');
end
if  log2==1
    error('signal 2 should be one dimensional');

end

if length(sig1)>length(sig2)
    filter=sig2;
    signal=sig1;
else
    filter=sig1;
    signal=sig2;
end


flen=length(filter);
slen=length(signal);
clen=flen+slen-1;  %length of convolution


if Blen>slen
    error('Block length should be less than signal length');
end
```

```
if Blen==0
    error('Block length Should not be zero');
end
if Blen<flen-1
    error('Block length should be at least filter length-1');
end




r=rem(slen,Blen);
if r~=0
sigg=[signal zeros(1,Blen-r)];
else
    sigg=signal;
end


sslen=length(sigg);
Nbins=sslen./Blen;



switch MOD
case 'direct'
sConvv=zeros(1,Nbins*Blen+flen-1);
sigg=[zeros(1,flen-1) sigg];
idx1=1;

for i=1:Nbins

    ss2=sigg(idx1:idx1+Blen-1+flen-1);  %taking out one bin = N1 points
    cnv=dirConv(ss2,filter);
    sConvv(idx1:idx1+Blen-1+flen-1)=cnv(flen:end);
    idx1=idx1+Blen;
end



sConv=sConvv(1:clen);


case 'fft'

sConvv=zeros(1,Nbins*Blen+flen-1);
sigg=[zeros(1,flen-1) sigg];
idx1=1;

for i=1:Nbins

    ss2=sigg(idx1:idx1+Blen-1+flen-1);  %taking out one bin = N1 points
    cnv=fftConv(ss2,filter);
    sConvv(idx1:idx1+Blen-1+flen-1)=cnv(flen:end);
```

```
    idx1=idx1+Blen;
end


sConv=sConvv(1:clen);
end
```

```
function aConv=oaConv(sig1,sig2,Blen,MOD)
% OACONV  convolution using overlap and add method
% C=OACONV(A,B,BLEN,MOD) performs convolution on A and B using overlap and
% and add method.  BLEN is the block length designated for convolution. BLEN
% should not be greater than the length of both signals.
% MOD is a string that designates which method of convolution should be
performed
% in a block convolution.  MOD='direct' uses the direct method and MOD='fft'
uses
% the fft.
%The resulting vector is length LENGTH(A)+LENGTH(B)-1
%Created by Mohammadhafez Bazrafshan

if nargin<2
    error('Input Arguments Not defined');
end

if nargin<3
    error('Block Length Not defined');
end

if nargin<4
    error('Mode of conv not defined, ''direct'' for direct convolution and
''fft'' for fast');
end

[A AA]=size(sig1);
[B BB]=size(sig2);
log1=(A~=1) & (AA~=1);
log2=(B~=1) & (BB~=1);
if  log1==1
    error('signal 1 should be one dimensional');
end
if  log2==1
    error('signal 2 should be one dimensional');

end

if length(sig1)>length(sig2)
    filter=sig2;
    signal=sig1;
else
    filter=sig1;
    signal=sig2;
end


flen=length(filter);
slen=length(signal);
clen=flen+slen-1;  %length of convolution


if Blen>slen
    error('Block length should be less than signal length');
end
```

```
if Blen==0
    error('Block length Should not be zero');
end



r=rem(slen,Blen);
if r~=0
sigg=[signal zeros(1,Blen-r)];
else
    sigg=signal;
end


switch MOD
    case 'direct'
sslen=length(sigg);
Nbins=sslen./Blen;
aConvv=zeros(1,Nbins*Blen+flen-1);

for i=1:Nbins
    idx1=(i-1)*Blen+1;
    idx2=i*Blen;
    Sig=sigg(idx1:idx2);
    aConvv(idx1:idx2+flen-1)=aConvv(idx1:idx2+flen-1)+dirConv(Sig,filter);
end
aConv=aConvv(1:clen);

    case 'fft'
        sslen=length(sigg);
Nbins=sslen./Blen;
aConvv=zeros(1,Nbins*Blen+flen-1);

for i=1:Nbins
    idx1=(i-1)*Blen+1;
    idx2=i*Blen;
        Sig=sigg(idx1:idx2);
    aConvv(idx1:idx2+flen-1)=aConvv(idx1:idx2+flen-1)+fftConv(Sig,filter);
end
aConv=aConvv(1:clen);

end
```