# DFT Applications – Final Exam

## Tamoghna Chattopadhyay

## Section 1: Sampling and Reconstruction with DFT filter.

## <u>Source Code:</u>

```
% -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -
% TITLE: Sampling and Reconstruction with DFT filter.
%
% Purpose: To  applying the Discrete Fourier Transform (DFT) to a
% variety of signal processing problems.
%
% Date created: 07/30/2016 Author: Tamoghna Chattopadhyay
% Date modified: rev1 - 08/02/2016
% -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- --

% Generate 8192 samples of x(t) at a sampling rate of 100K (10 u seconds).
n = [0:8191];
x = cos( 2*pi*(160/100)*n ) + 0.5 * sin( 2*pi*(180/100)*n );

% Compute the DFT and plot its magnitude for the sampled signal
X = fft(x);
figure(1);
subplot(211),plot( abs(X) );
title( 'Fourier Transform of Original' );
xlabel( 'Frequency (Hertz)');
ylabel( 'Magnitude (unknown) ');

% Insert 3 zeros between each sample
x1 = ins_zeros( x, 4 );

% The spectrum of the up sampled signal
X1 = fft(x1);
subplot(212),plot( abs(X1) );
title( 'Fourier Transform of Zero Inserted' );
xlabel( 'Frequency (Hertz)');
ylabel( 'Magnitude (unknown) ');

% Construct a normalized frequency array ( 0 to 1 matches 0 to fs/2 )
f = [0:length(X1)-1]/length(X1);

% Apply Highpass Filter to the transform
```

```
HP_FM  = X1 .* HighPass_dft( length( X1 ) )';

% Display the filtered transform
figure(2);
plot( f(1:end/2),  abs( HP_FM(1:end/2)));
title('Fourier Transform of the filtered version');
xlabel( 'Frequency (KHz)');
ylabel( 'Magnitude (unknown) ');

% Generate 8192 samples of x(t) at a sampling rate of 400K (2.5 u
seconds).
x2 = cos( 2*pi*(160/400)*n ) + 0.5 * sin( 2*pi*(180/400)*n );

% Reconstruct the Original Signal
Xr = 4*ifft(HP_FM);

% Compare the reconstructed signal and the original signal sampled at 400K
figure(3);
plot( x2, 'r:');
hold on;
plot( Xr, 'b-');
title('Comparison of the reconstructed signal and the original signal
sampled at 400K');
xlabel( 'Frequency (Hertz)');
ylabel( 'Magnitude (unknown) ');
```

## Function for Inserting Zeroes:

```
function xe = ins_zeros( x, I )
%
% xe = ins_zeros( x, I );
%
% This function will create an extended version of the signal x
% by simply inserting (I-1) zeros between each sample.

% Create extended set.
xe = zeros( length(x)*I, 1 );
xe(1:I:end) = x;
return;
```

## Function for HighPass DFT Filter:

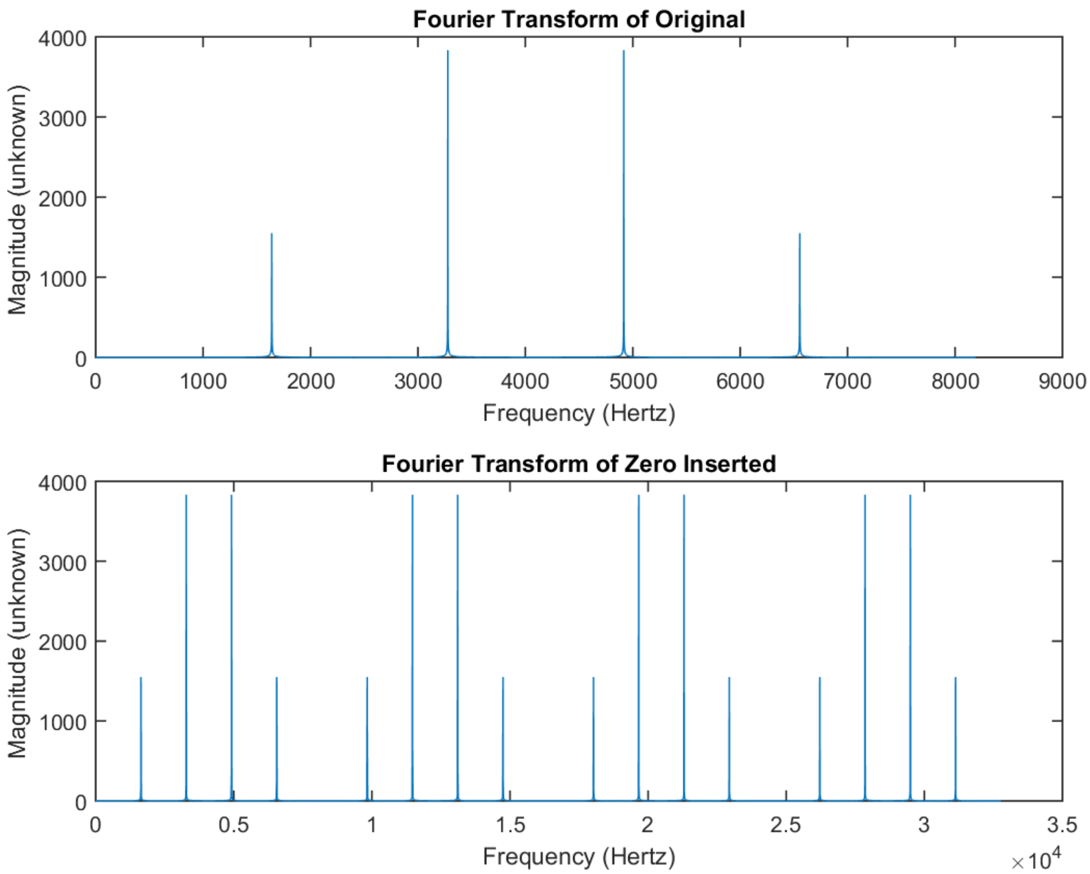
```
function H = HighPass_dft( N );
%
% H = HighPass_dft( N );
%
% This program generates a High Pass filter for application to the DFT of
a signal.
% The output is an array that is point by point multiplied
% with the dft of the signal.
```

```
% The input parameter N is then length of the signal.

n1 = floor( 0.35 * N );
n2 = floor( 0.3875 * N );
n3 = floor( 0.5 * N + 1 );

H = zeros( 1, N );
H( n1:n2 ) = [0:1/(n2-n1):1]; % Up to one
H( n2+1:n3 ) = H( n2+1:n3 ) + 1; % Set At one
H( (N/2+2):N ) = H(N/2:-1:2); % Conjugate Symmetry
return;
```

## <u>Plots:</u>




<u>Figure 1</u>

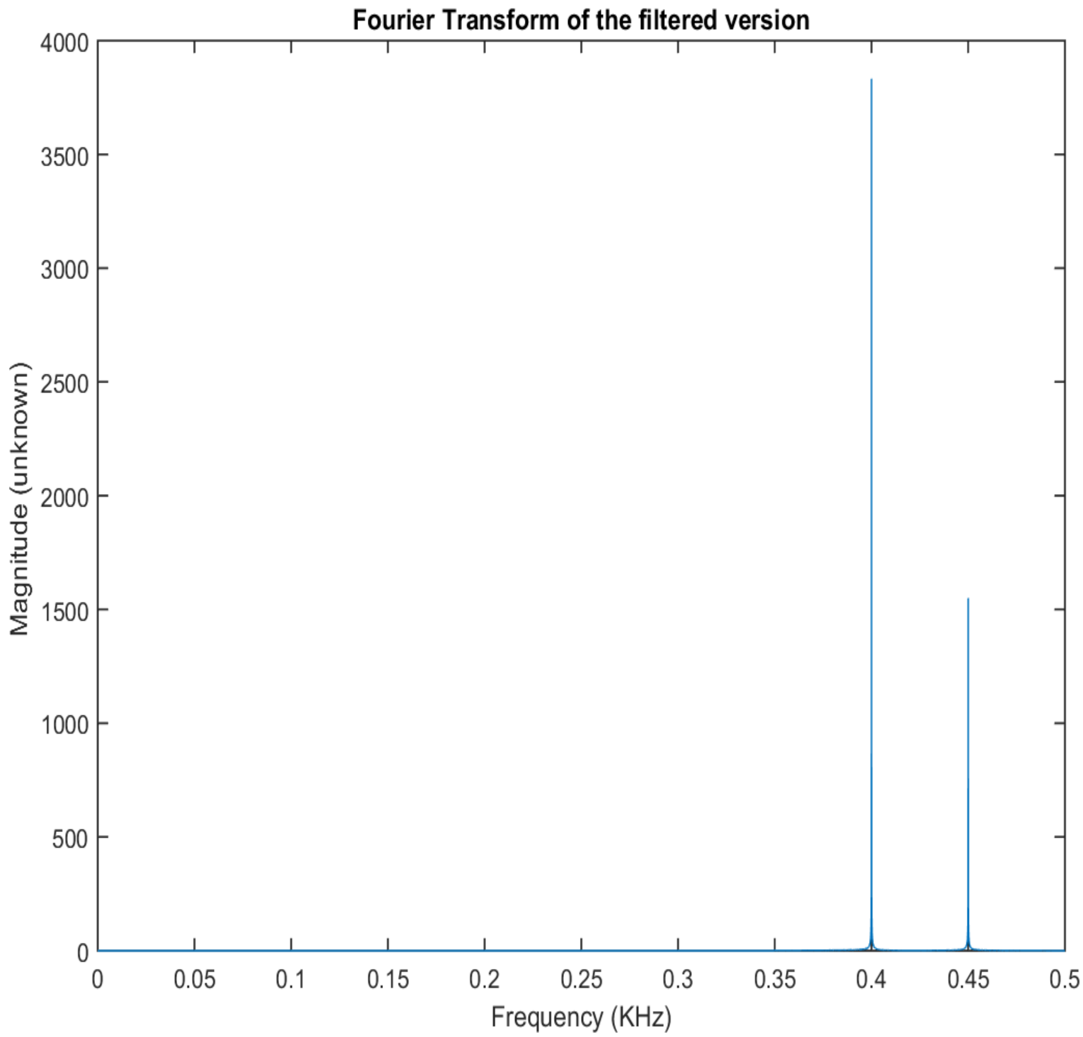
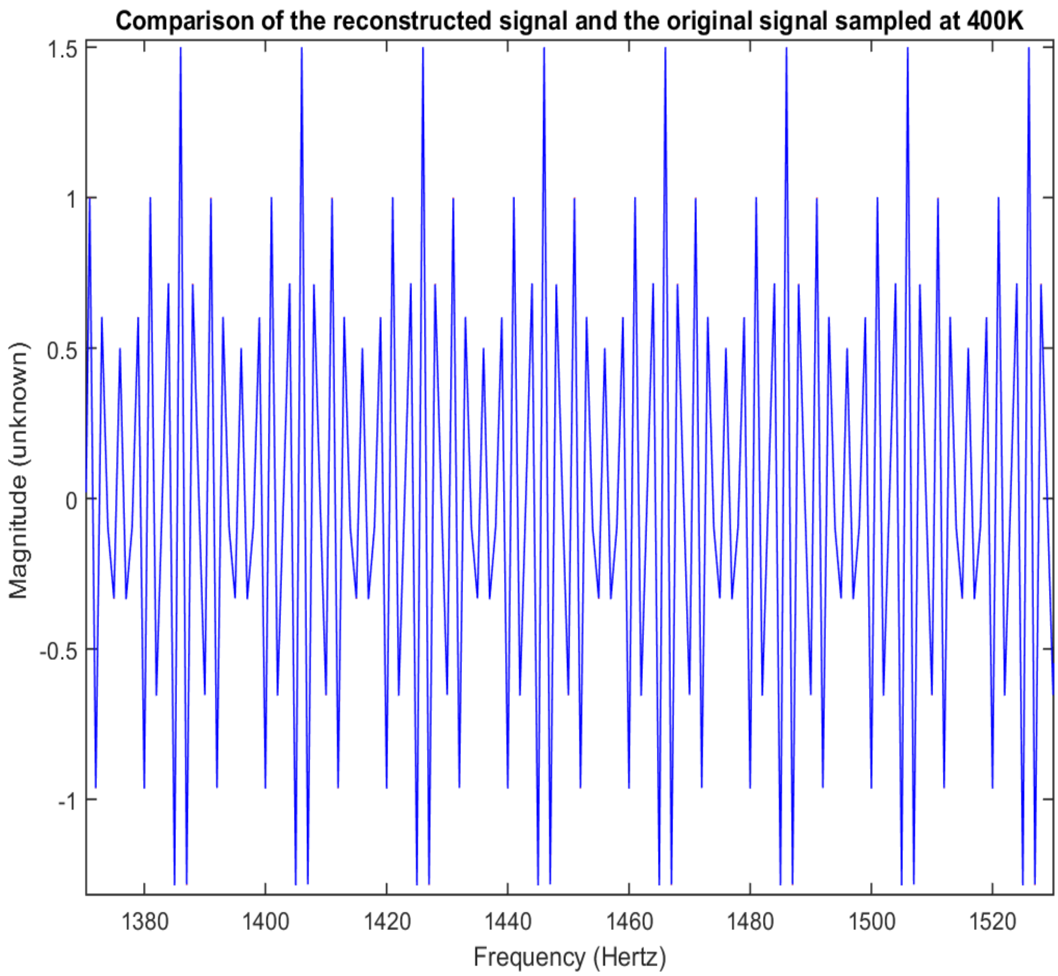

Fourier Transform of the filtered version
4000
3500
3000
2500
2000
1500
1000
500
0
Magnitude (unknown)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
Frequency (KHz)

Figure 2



Comparison of the reconstructed signal and the original signal sampled at 400K
1.5
1
0.5
0
-0.5
-1
Magnitude (unknown)
1380
1400
1420
1440
1460
1480
1500
1520
Frequency (Hertz)


Figure 3

# Section 2: Overlap Add.

## Source Code:

```
% -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -
% TITLE: Overlap Add
%
% Purpose: To employ Overlap Add in order to process a long signal. A FIR
% bandpass filter with cosine times a hamming window is utilised.
%
%
% Date created: 08/01/2016 Author: Tamoghna Chattopadhyay
% Date modified: rev1 - 08/02/2016
% -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- --

% Load the given file
load('DFT_Final.mat');

% Plot the original signal
figure(1);
plot( signal ); % Plot the spectrum
title('Original signal in given MatLab file');
xlabel (' Frequency in Hertz ' );
ylabel (' Magnitude (unknown) ');

% Define the lengths of filter and transform, i.e. 1024 point data
H_Length = length(h);
F_Length = 1024

% Find the fourier transform
H = fft( h, 1024 );

% Filter the data using Overlap-Add
n = 0;
m = 1;
figure(2);
for k = 1:4
    SIG = fft( signal( m:m+F_Length-H_Length-1 ),F_Length );
    y = ifft( SIG.*H );
    if( k==1 )
        z = y;
        n = F_Length-H_Length;
    else
        z = [z(1:n)  z(n+1:n+H_Length)+y(1:H_Length)
y(H_Length+1:F_Length)];
        n = n + F_Length-H_Length;
        end;
    m = m + F_Length-H_Length;
end;

% Plot the result of filtering
```

```
figure(2);
plot( real( z ) );
title('Overlap-Add Method of Filter Data');
xlabel( 'Frequency (Hertz)');
ylabel( 'Magnitude (unknown) ');
```

## Plots:

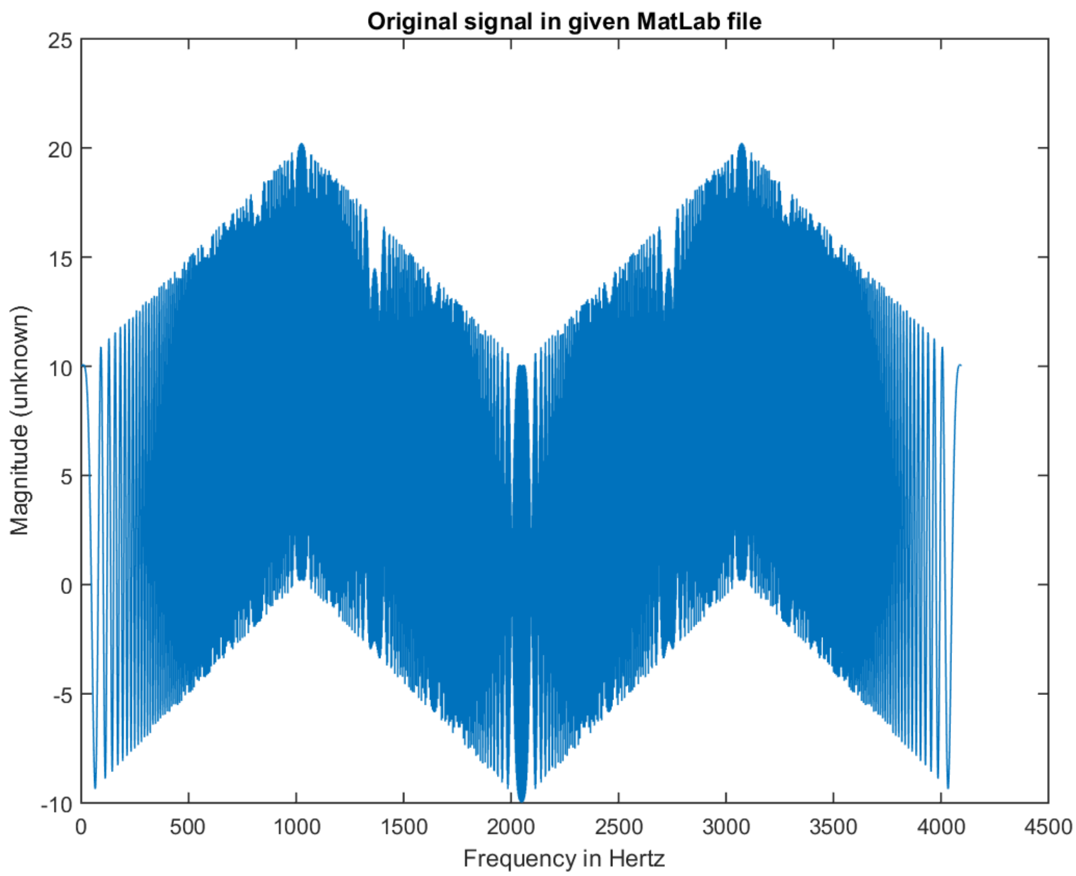



Figure 1

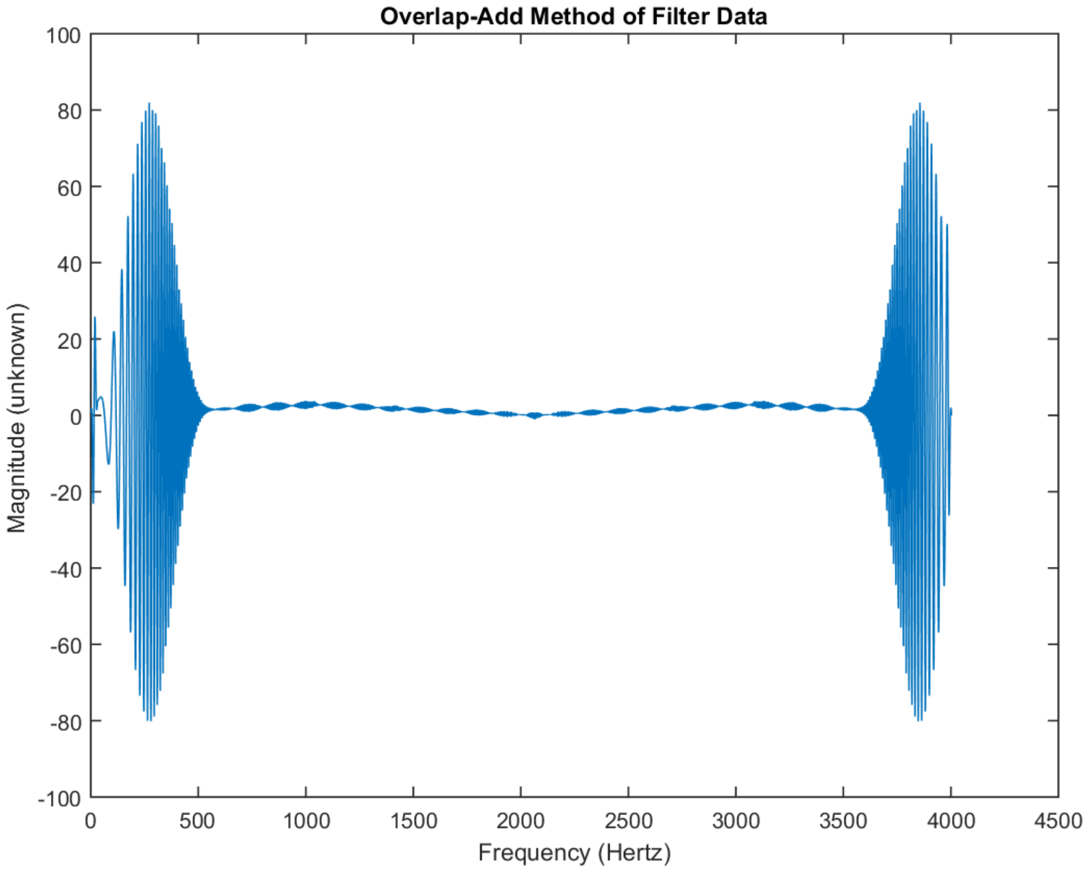

Overlap-Add Method of Filter Data
Magnitude (unknown)
Frequency (Hertz)
100
80
60
40
20
0
-20
-40
-60
-80
-100
0
500
1000
1500
2000
2500
3000
3500
4000
4500

Figure 2

Section 3: Spectral Analysis.

## Source Code:

```
% -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -
% TITLE: Spectral Analysis
%
% Purpose: To do spectral analysis on the signal test given in the MatLab
% file provided.
%
%
% Date created: 08/01/2016 Author: Tamoghna Chattopadhyay
% Date modified: rev1 - 08/02/2016
% -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- --

% Load the given file
load('DFT_Final.mat');

% Find the frequency of the large main component by iteration
N = 256;
TEST = fft(test,N);
[TMax,NMax] = max(abs(TEST(1:end/2)));
freq = 100e3*(NMax-1)/N;

N = 2*N;
TEST = fft(test,N);
[TMax,NMax] = max(abs(TEST(1:end/2)));
freq = 100e3*(NMax-1)/N;

N = 2*N;
TEST = fft(test,N);
[TMax,NMax] = max(abs(TEST(1:end/2)));
freq = 100e3*(NMax-1)/N;

N = 2*N;
TEST = fft(test,N);
[TMax,NMax] = max(abs(TEST(1:end/2)));
freq = 100e3*(NMax-1)/N;

N = 2*N;
TEST = fft(test,N);
[TMax,NMax] = max(abs(TEST(1:end/2)));
freq = 100e3*(NMax-1)/N

% Plot the fourier transform of the signal
figure(1);
plot( abs(TEST));
title('Fourier Transform of original Signal');
xlabel( 'Frequency (Hertz)');
ylabel( 'Magnitude (unknown) ');
```

```
% Subtract out the maximum component
n=0:length(test)-1;
Sub = test - real((1/length(test))*(TMax*exp(i*2*pi*(NMax-1)/N*n))) + ...
conj(TMax)*exp(-i*2*pi*(NMax-1)/N*n);

% Compute its fourier transform
SUB = fft(Sub,N);

% Plot the output of the fourier transform of the subtraction
figure(2);
plot( abs(SUB));
title('Fourier Transform of Signal with subtracted component ');
xlabel( 'Frequency (Hertz)');
ylabel( 'Magnitude (unknown) ');

% Construct a normalized frequency array ( 0 to 1 matches 0 to fs/2 )
f = [0:length(TEST)-1]/length(TEST);

% Apply Highpass Filter to the transform
BP_FM  = TEST .* BandPass_dft( length( TEST ) );

% Display the filtered transform
figure(3);
plot( f(1:end/2),  abs( BP_FM(1:end/2)));
title('Fourier Transform of the filtered version');
xlabel( 'Frequency (KHz)');
ylabel( 'Magnitude (unknown) ');
```

## Function for BandPass DFT Filter:

```
function H = BandPass_dft( N );
%
% H = BandPass_dft( N );
%
% This program generates a Band Pass filter for application to the DFT of
a signal.
% The output is an array that is point by point multiplied
% with the dft of the signal.

% The input parameter N is then length of the signal.

n1 = floor( 0.1 * N );
n2 = floor( 0.15 * N );
n3 = floor( 0.25 * N );
n4 = floor( 0.3 * N );
H = zeros( 1, N );
H( n1:n2 ) = [0:1/(n2-n1):1]; % Up to one
H( n2+1:n3-1 ) = H( n2+1:n3-1 ) + 1; % Set At one
H( n3:n4 ) = [1:-1/(n4-n3):0]; % Down to zero
H( (N/2+2):N ) = H(N/2:-1:2); % Conjugate Symmetry
return;
```

Plots:

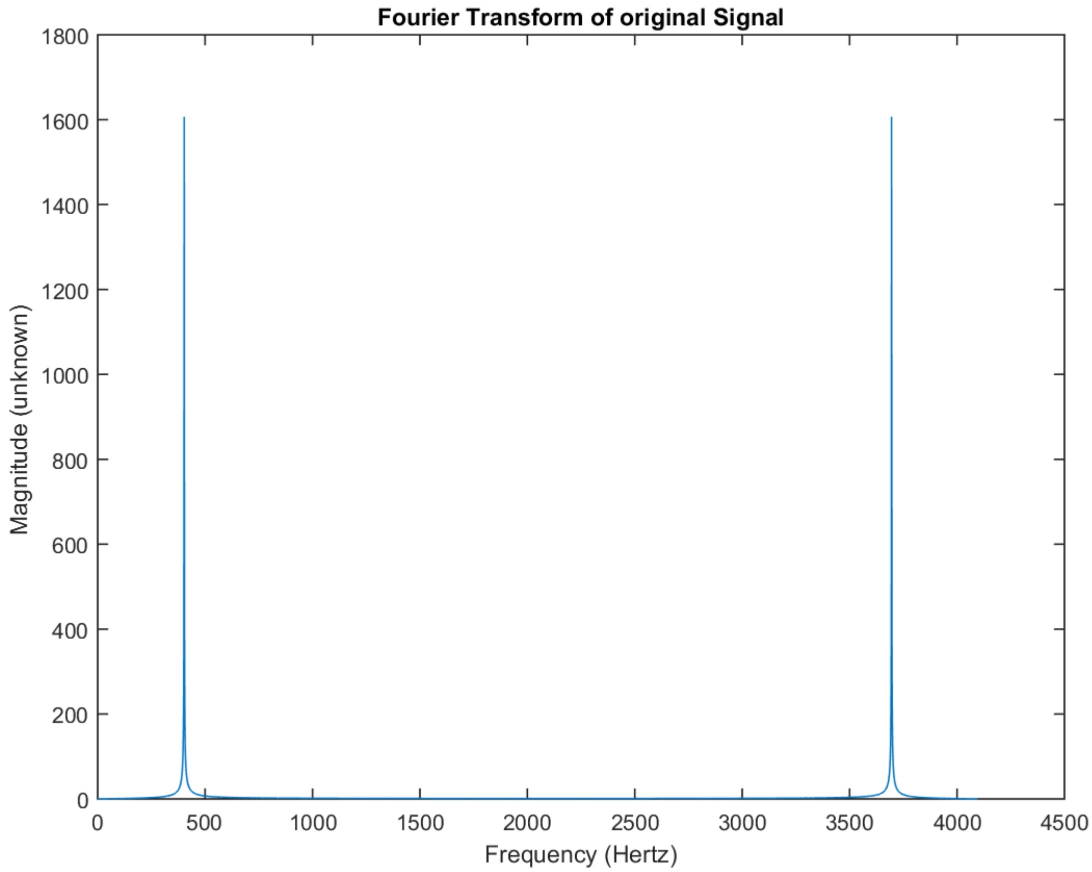
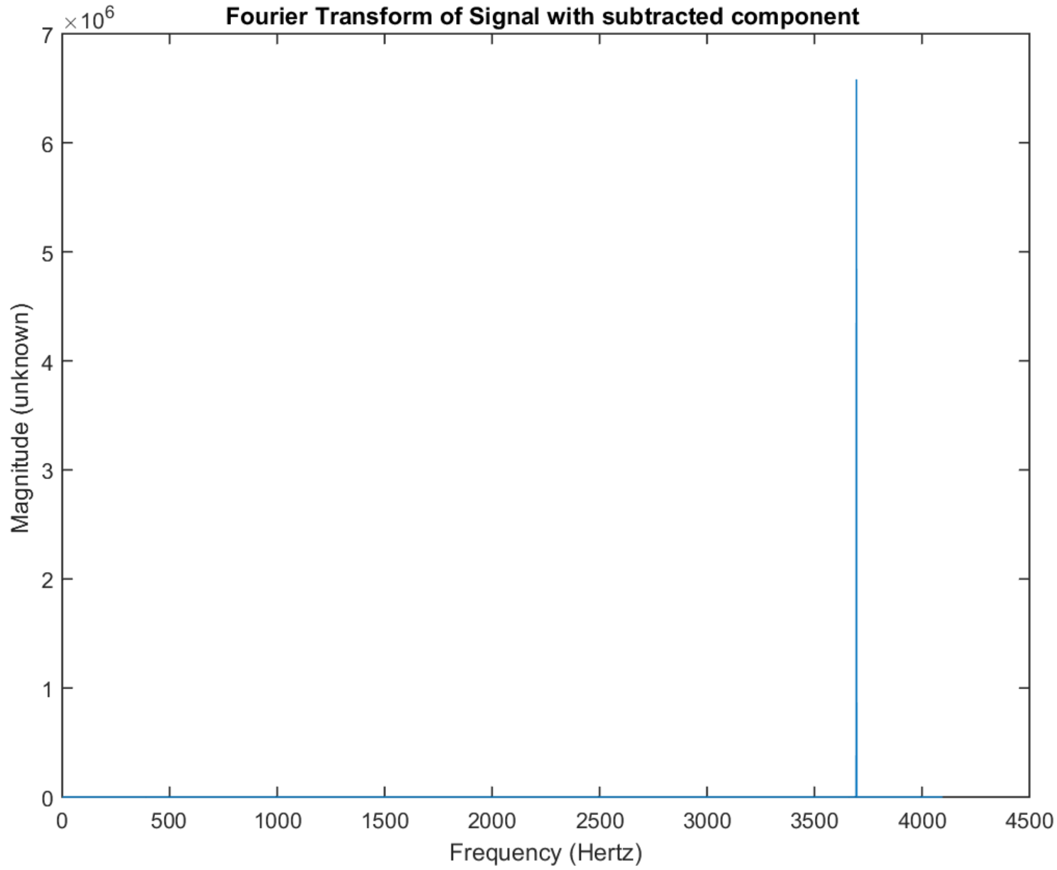



Figure 1


Fourier Transform of Signal with subtracted component
×10^6
7
6
5
4
3
2
1
0
Magnitude (unknown)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Frequency (Hertz)

<u>Figure 2</u>

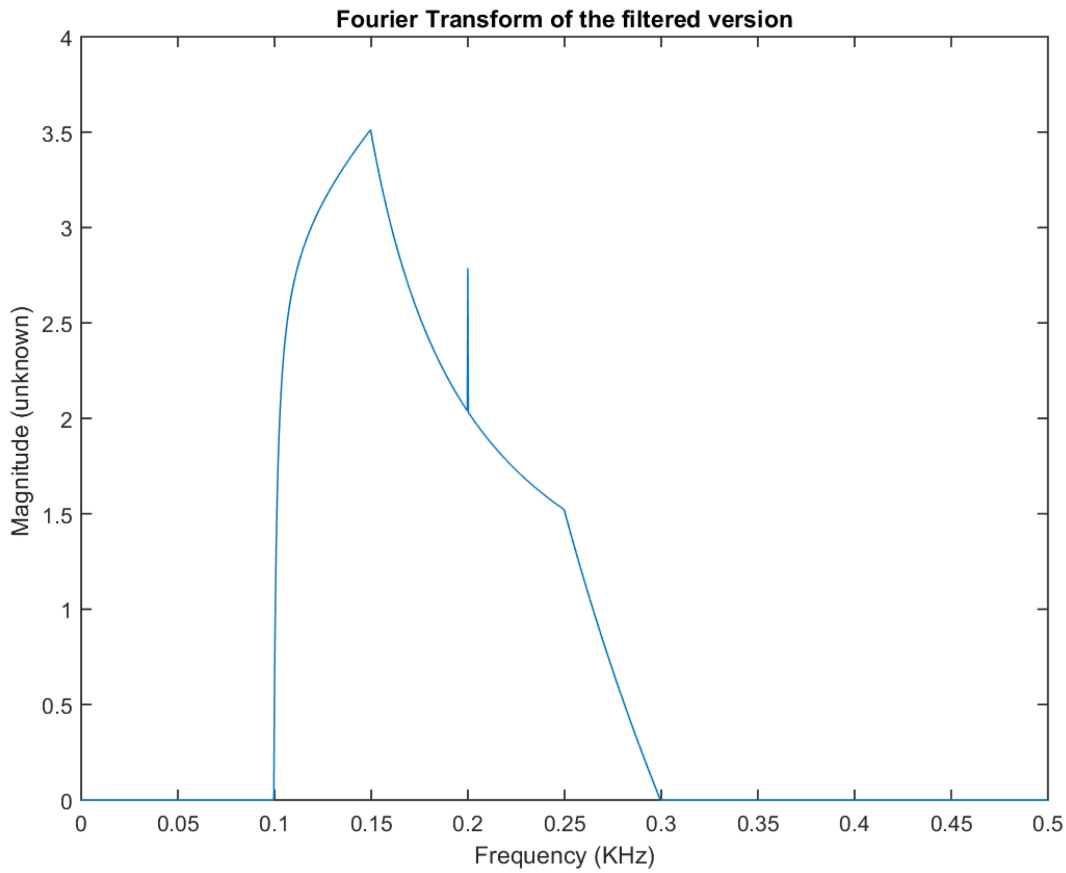

Fourier Transform of the filtered version
Magnitude (unknown)
Frequency (KHz)
0
0.5
1
1.5
2
2.5
3
3.5
4
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Figure 3