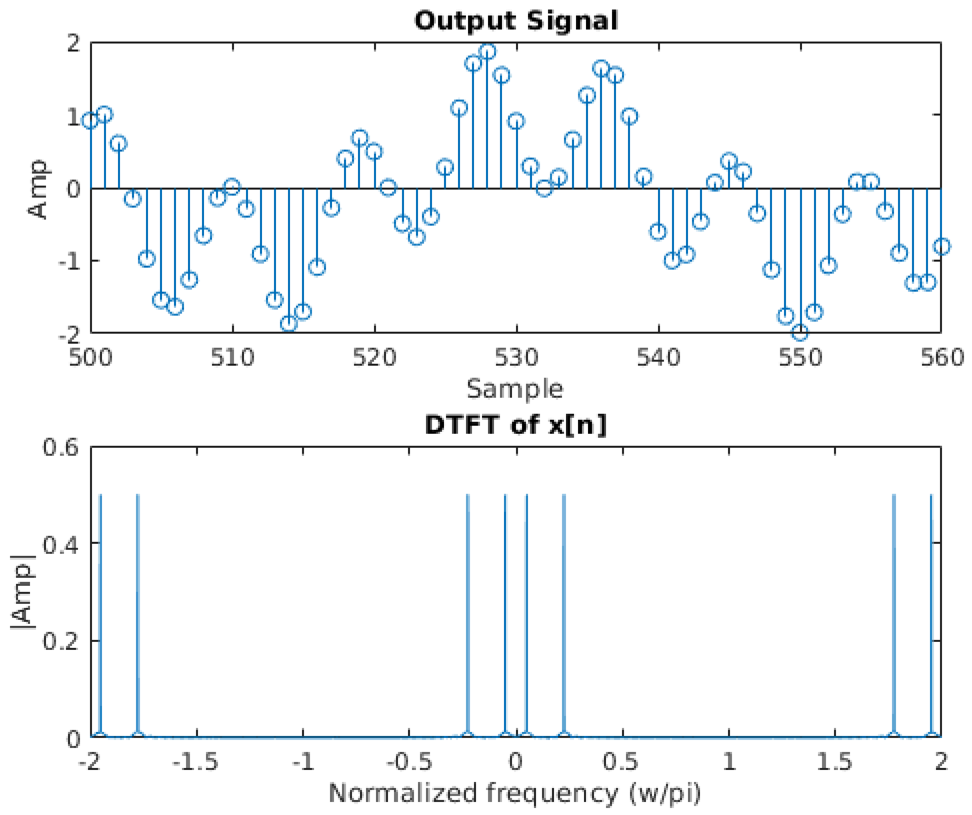
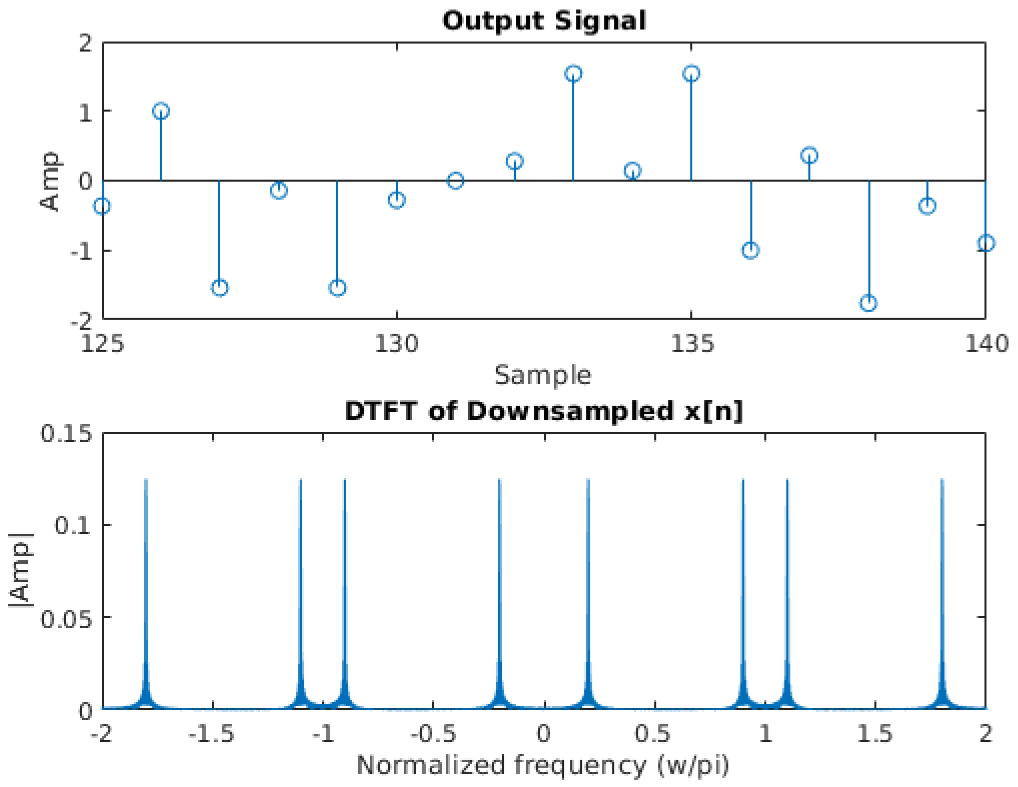
```
%!------------------------------------------------------------------------
%! DSP HW11 #1
%!  - Create sampled signal x[n] = sin[2*pi*f1*n*Ts] + sin[2*pi*f2*n*Ts]
%!  - Plot samples 500:560 and plot the DTFT
%!  - Upsample by 4 and display results
%!------------------------------------------------------------------------

%! Enviorment
clear; close all;
addpath([fileparts(mfilename('fullpath')), '/../../functions']);

%! Variables
Fs       = 4000;
Ts       = 1/Fs;
f1       = 100;
f2       = 450;
N        = 2048;
n        = 0:N-1;
n_plot   = 500:560;
w        = (-2000:2000)*pi/1000;
d_samp   = 4;

%! Create Signals
x_n = sin(2*pi*f1*n*Ts) + sin(2*pi*f2*n*Ts);
x_downsample = downsample(x_n, d_samp);

%! Downsample variables
m = n(1)/4 : n(end)/4;
m_plot = 125:140;
w_downsample = w(1:4:end);

%! Take DTFT of singals
x_f         = dtft(x_n, n, w) / N;
x_f_down    = dtft(x_downsample, m, w) / N;

%! Plot
figure()
subplot(2,1,1)
stem(n_plot, x_n(n_plot))
title('Output Signal')
xlabel('Sample')
ylabel('Amp')
subplot(2,1,2)
plot(w/pi, fftshift(abs(x_f)))
title('DTFT of x[n]')
xlabel('Normalized frequency (w/pi)')
ylabel('|Amp|')

figure()
subplot(2,1,1)
stem(m_plot, x_downsample(m_plot))
xlim([m_plot(1),m_plot(end)])
```

```
title('Output Signal')
xlabel('Sample')
ylabel('Amp')
subplot(2,1,2)
plot(w/pi, fftshift(abs(x_f_down)))
title('DTFT of Downsampled x[n]')
xlabel('Normalized frequency (w/pi)')
ylabel('|Amp|')
```

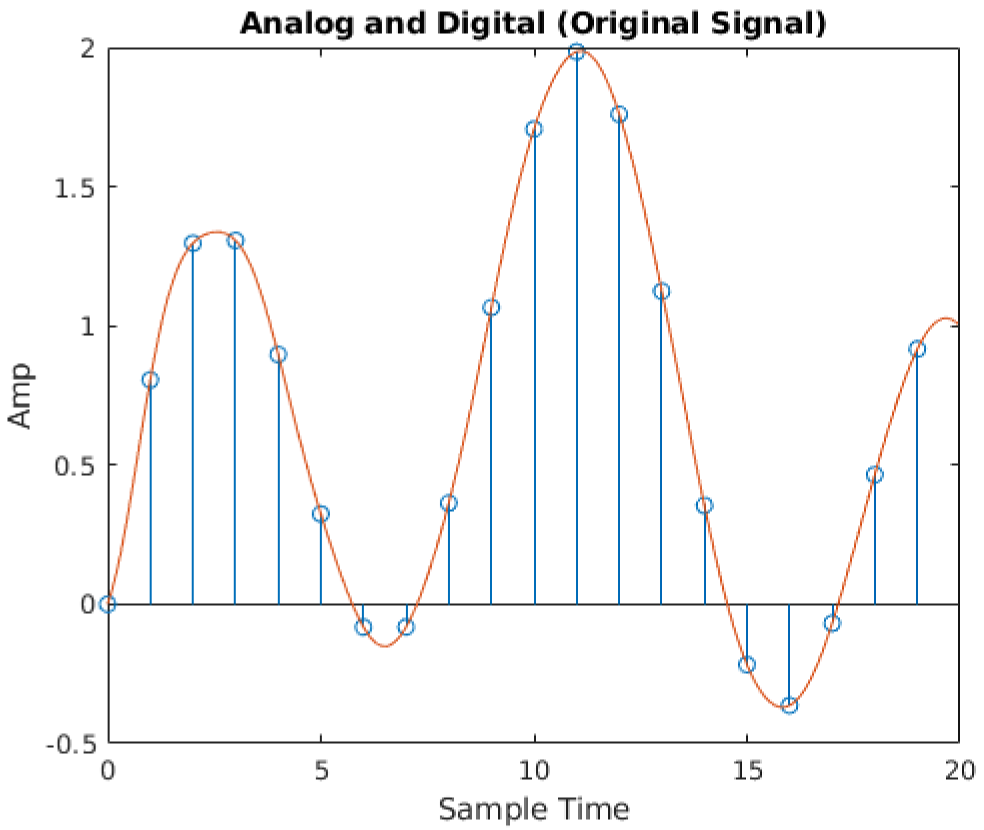


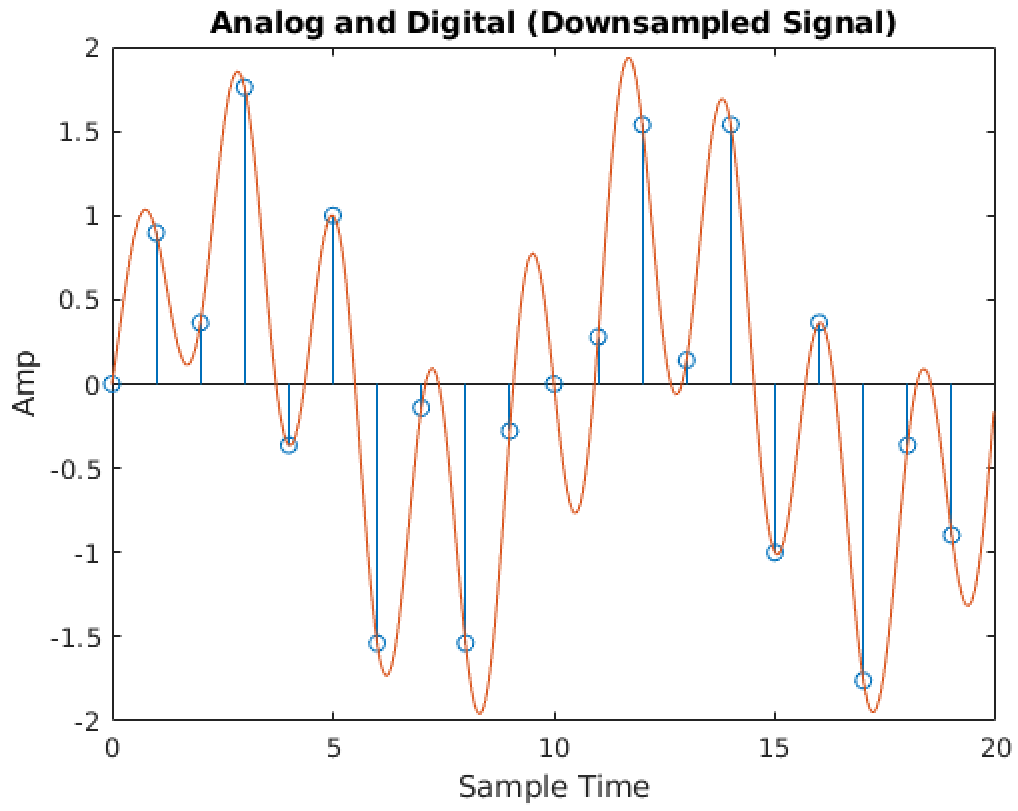


Published with MATLAB® R2023a

```
%!-------------------------------------------------------------------------
%! DSP HW11 #2
%!  - Reconstruct x(t) from x[n]
%!  - Reconstruct x(t) from downsampled x[n]
%!  - Upsample by 4 and display results
%!-------------------------------------------------------------------------

%! Enviorment
clear; close all;
addpath([fileparts(mfilename('fullpath')), '/../../functions']);

%! Variables
Fs      = 4000;
Ts      = 1/Fs;
f1      = 100;
f2      = 450;
N       = 2048;
n       = 0:N-1;
d_samp  = 4;
t_delta = .05;
t       = 0:t_delta:N-1;
t_down  = 0:t_delta:N/4-1;

%! Create Signals
x_n = sin(2*pi*f1*n*Ts) + sin(2*pi*f2*n*Ts);
x_downsample = downsample(x_n, d_samp);

%! Reconstruct orignial signal using sinc method
x = zeros(N, length(t));
for i=1:N
    x(i, :) = x_n(i)*sinc(t-i+1);
end
xa_orig = sum(x);

%! Reconstruct downsampled signal using sinc method
x = zeros(N/4, length(t_down));
for i=1:N/4
    x(i, :) = x_downsample(i)*sinc(t_down-i+1);
end
xa_down = sum(x);

%! Plot
%! Plot 20 digital samples
figure()
stem(n(1:20), x_n(1:20))
title('Analog and Digital (Original Signal)')
xlabel('Sample Time')
ylabel('Amp')
hold on
plot(t(1:20/.05), xa_orig(1:20/.05))
hold off
```

```
%! Plot 20 Digital Samples
figure()
stem(n(1:20), x_downsample(1:20))
title('Analog and Digital (Downsampled Signal)')
xlabel('Sample Time')
ylabel('Amp')
hold on
plot(t(1:20/.05), xa_down(1:20/.05))
hold off
```



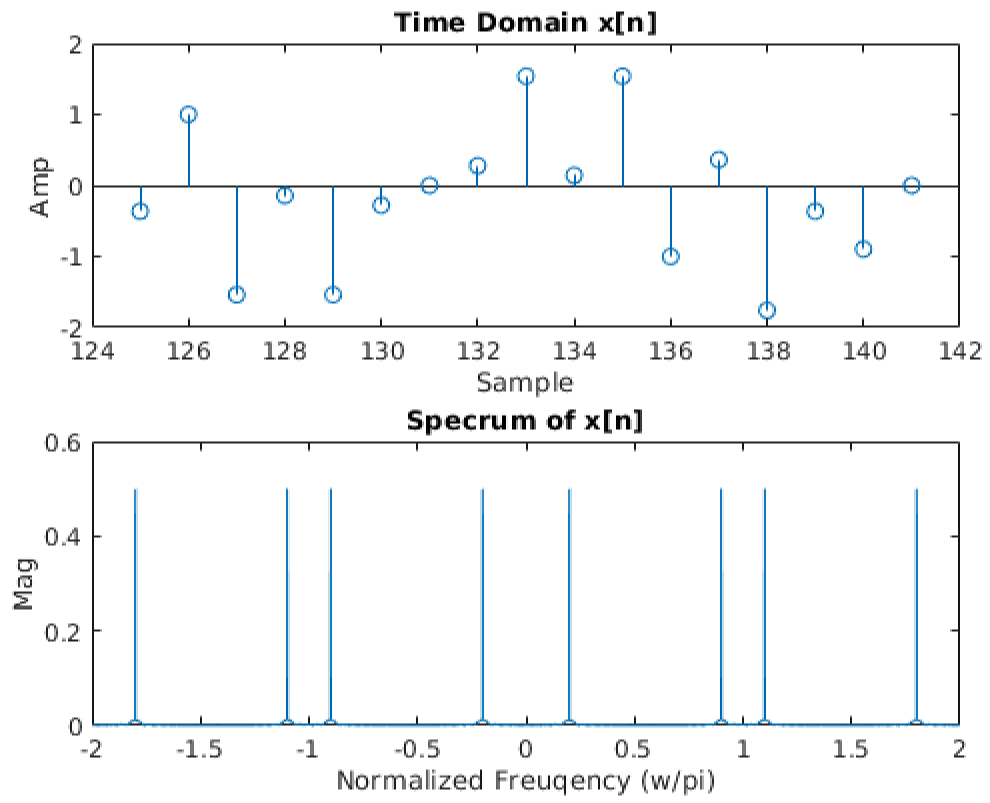


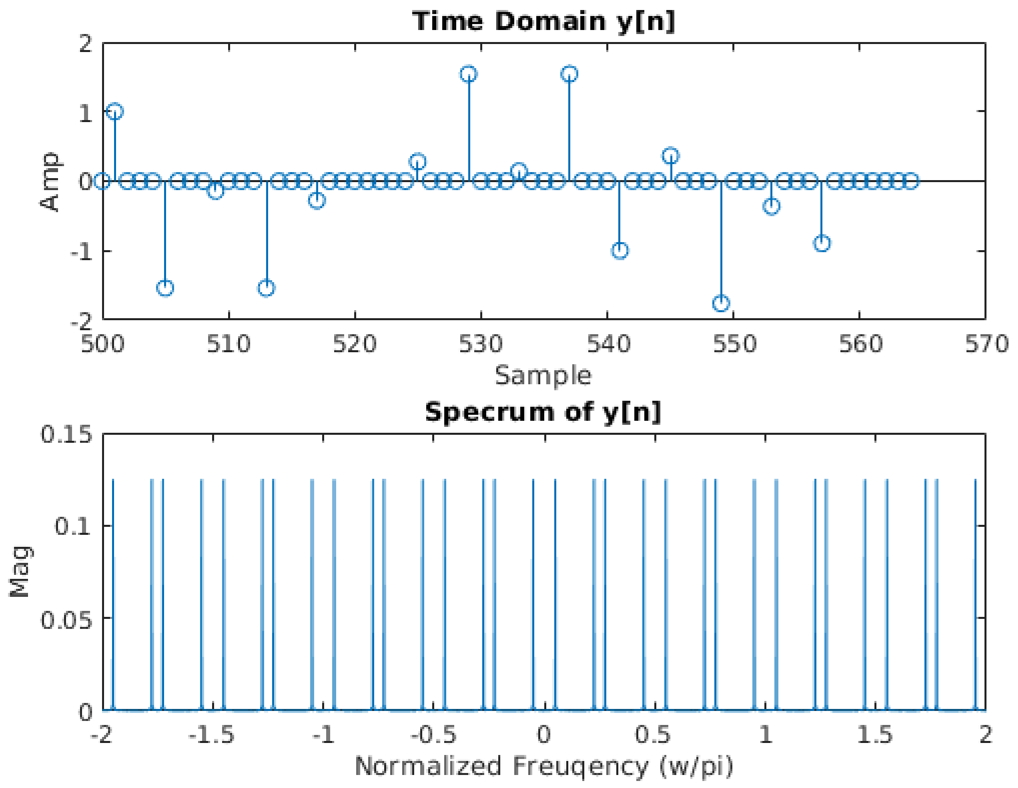
Published with MATLAB® R2023a

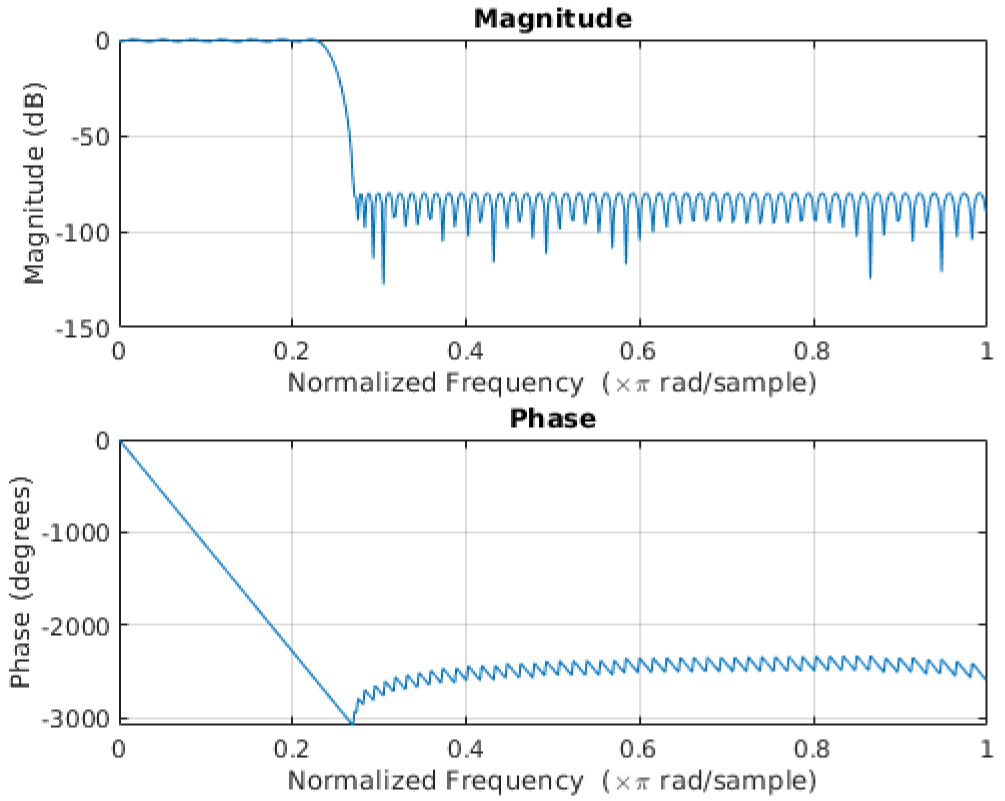
```
%!---------------------------------------------------------------------------
%! DSP HW11 #3
%!  - Create x[n] and upsample
%!  - Design a filter for the upsampled signal
%!  - Display and compare the two singals
%!---------------------------------------------------------------------------

%! Enviorment
clear; close all; clc;
addpath([fileparts(mfilename('fullpath')), '/../../functions']);
coeff = load("filters/coeff_3.mat");

%! Variables
Fs      = 1000;
Ts      = 1/Fs;
f1      = 100;
f2      = 450;
N       = 2048;
n       = 0:N-1;
up_samp = 4;
n_plot  = 125:141;
w       = (-2000:2000)*pi/1000;

%! Create Signals
x_n = sin(2*pi*f1*n*Ts) + sin(2*pi*f2*n*Ts);
y_n = upsample(x_n, up_samp);

%! LFP
y_m = filter(coeff.Num, 1, y_n);

%! Take DTFT
x_f = dtft(x_n, n, w) / N;
y_f = dtft(y_n, 0:N*up_samp-1, w) / (N*up_samp);
ym_f = dtft(y_m, 0:N*up_samp-1, w) / (N*up_samp);

%! Plot
figure()
subplot(2,1,1)
stem(n_plot, x_n(n_plot))
title('Time Domain x[n]')
xlabel('Sample')
ylabel('Amp')
subplot(2,1,2)
plot(w/pi, abs(x_f))
title('Specrum of x[n]')
ylabel('Mag')
xlabel('Normalized Freuqency (w/pi)')

figure()
y_plot = n_plot(1)*4:n_plot(end)*4;
subplot(2,1,1)
stem(y_plot, y_n(y_plot))
```

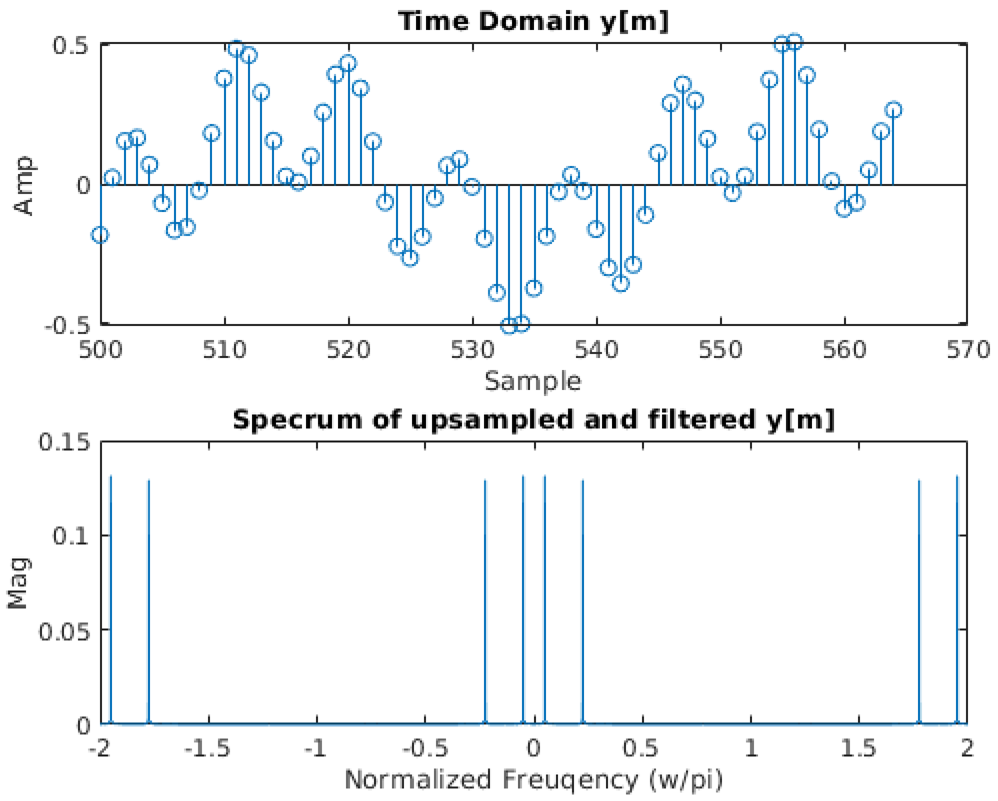
```
title('Time Domain y[n]')
xlabel('Sample')
ylabel('Amp')
subplot(2,1,2)
plot(w/pi, abs(y_f))
title('Specrum of y[n]')
ylabel('Mag')
xlabel('Normalized Freuqency (w/pi)')

figure()
freqz(coeff.Num)

figure()
y_plot = n_plot(1)*4:n_plot(end)*4;
subplot(2,1,1)
stem(y_plot, y_m(y_plot))
title('Time Domain y[m]')
xlabel('Sample')
ylabel('Amp')
subplot(2,1,2)
plot(w/pi, abs(ym_f))
title('Specrum of upsampled and filtered y[m]')
ylabel('Mag')
xlabel('Normalized Freuqency (w/pi)')

disp(["The new upsampled signal y[m]'s spectrum is the same as the origianl
 signal in",
    "problem 1. The only differences I have noticed is that a slight phase
 shift in the",
    "time damain signal."])
```

```
    "The new upsampled signal y[m]'s spectrum is the same as the origianl
 signal in"
    "problem 1. The only differences I have noticed is that a slight phase
 shift in the"
    "time damain signal."
```


Time Domain x[n]
Amp
Sample
Specrum of x[n]
Mag
Normalized Freuqency (w/pi)


Time Domain y[n]
Amp
Sample
Specrum of y[n]
Mag
Normalized Freuqency (w/pi)


Magnitude
Magnitude (dB)
0
-50
-100
-150
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
0
-1000
-2000
-3000
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

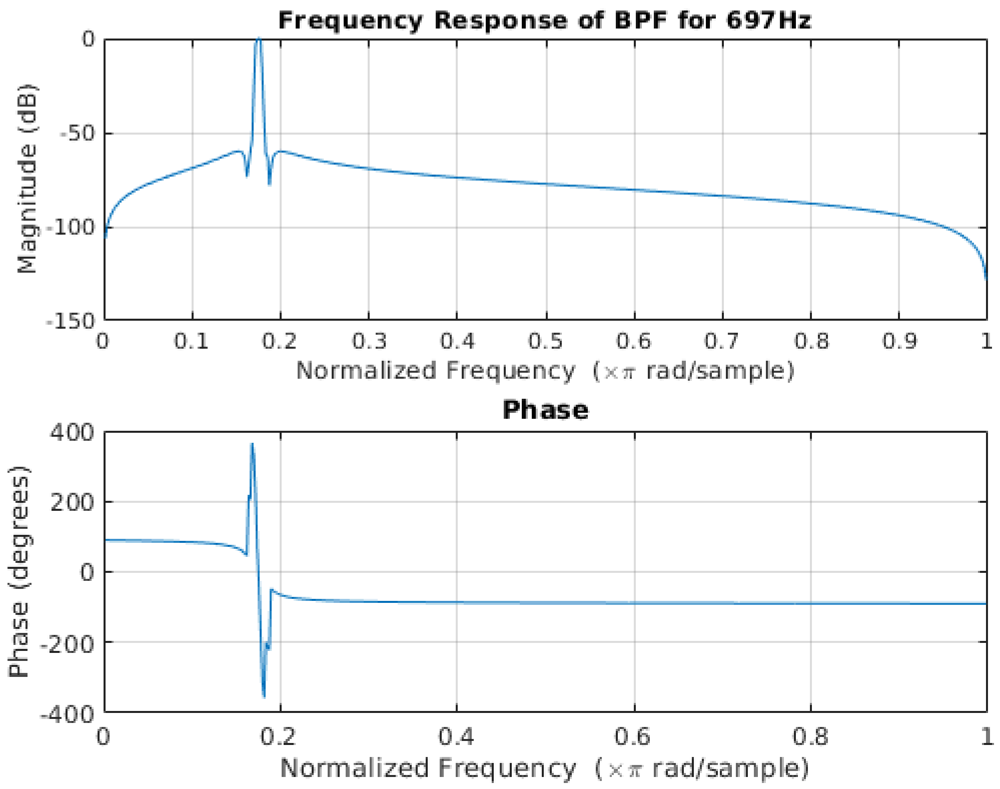

Time Domain y[m]
Amp
0.5
0
-0.5
500
510
520
530
540
550
560
570
Sample
Specrum of upsampled and filtered y[m]
Mag
0.15
0.1
0.05
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normalized Freuqency (w/pi)

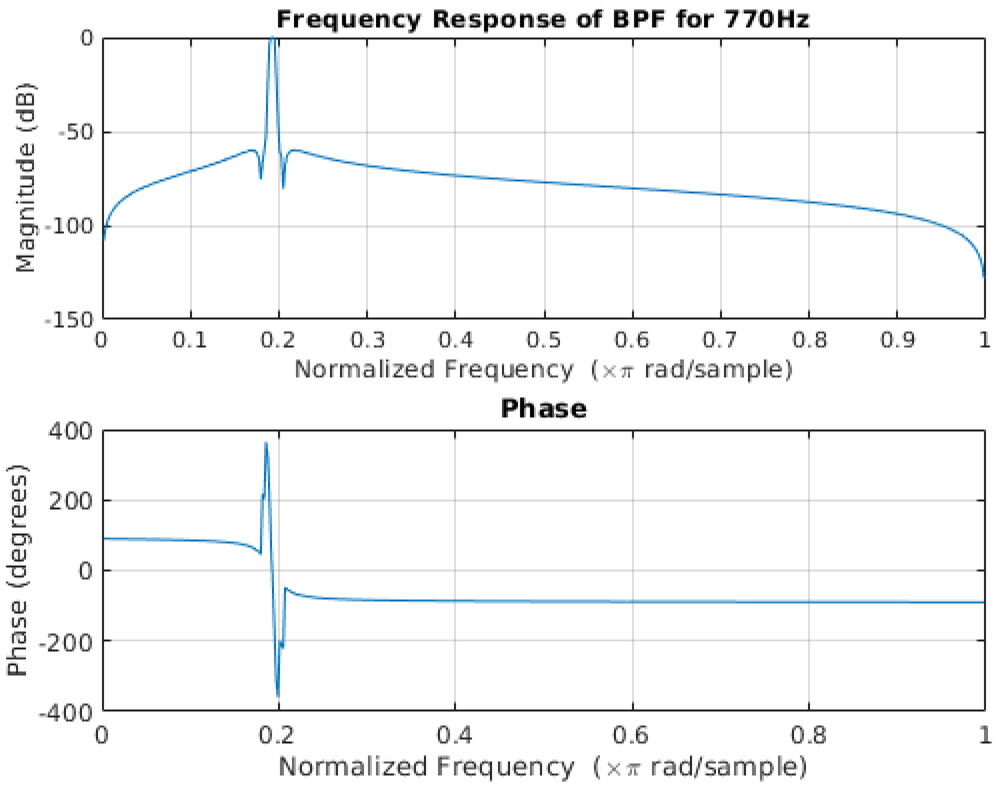
```matlab
%!-----------------------------------------------------------------------------
%! DSP HW11 #4
%!  - Create filters to match dtmf frequencies
%!  - Filter the signal
%!  - Decode the phone number
%!-----------------------------------------------------------------------------

%! Enviorment
clear; close all; clc;
addpath([fileparts(mfilename('fullpath')), '/../../functions']);
f697    = load("filters/filter697.mat");
f770    = load("filters/filter770.mat");
f852    = load("filters/filter852.mat");
f941    = load("filters/filter941.mat");
f1209   = load("filters/filter1209.mat");
f1336   = load("filters/filter1336.mat");
f1477   = load("filters/filter1477.mat");
filter_LP = load("filters/filter_LP.mat");

%! Phone Number
phone_num = [1 2 3 4 5 6 7 8 9 0];

%! Call dtmf
[x, Fs] = dtmf(phone_num);

%! Filter the phone signal with each of the filters
%   This creates 7 signals each with a single frequency range
y697    = filter(f697.Num, f697.Den, x);
y770    = filter(f770.Num, f770.Den, x);
y852    = filter(f852.Num, f852.Den, x);
y941    = filter(f941.Num, f941.Den, x);
y1209   = filter(f1209.Num, f1209.Den, x);
y1336   = filter(f1336.Num, f1336.Den, x);
y1477   = filter(f1477.Num, f1477.Den, x);

%! Take power and find if power is above threshold
%   If the power in a range of the signal is greater than the threshold
%   set that range to 1.
y = zeros(7, length(x));
y697out = filter(filter_LP.Num, filter_LP.Den, y697.^2);
y(1,:) = double(y697out > .3);
y770out = filter(filter_LP.Num, filter_LP.Den, y770.^2);
y(2,:) = double(y770out > .3);
y852out = filter(filter_LP.Num, filter_LP.Den, y852.^2);
y(3,:) = double(y852out > .3);
y941out = filter(filter_LP.Num, filter_LP.Den, y941.^2);
y(4,:) = double(y941out > .3);
y1209out = filter(filter_LP.Num, filter_LP.Den, y1209.^2);
y(5,:) = double(y1209out > .3);
y1336out = filter(filter_LP.Num, filter_LP.Den, y1336.^2);
y(6,:) = double(y1336out > .3);
y1477out = filter(filter_LP.Num, filter_LP.Den, y1477.^2);
```

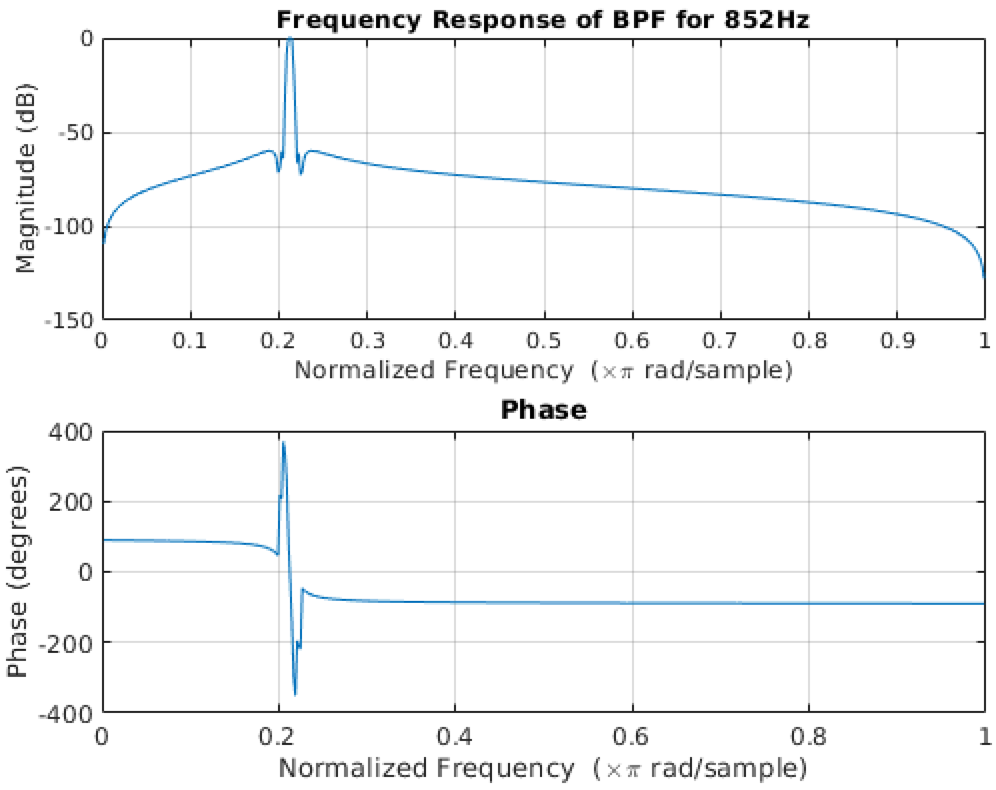
```
y(7,:) = double(y1477out > .3);

%! Downsample the y vectors
%   Get one sample per symbol
d = zeros(7, 10);
for i=1:7
    d(i,:) = downsample(y(i,:), 2500, 700);
end

% Array for the phone number pad
phone_num_arr = [1 2 3;
                 4 5 6;
                 7 8 9;
                -1 0 -2];
num=zeros(1,length(x)/2400);
for i=1:10
    idx = find(d(:,i) == 1);    % Find idx where the power is > thresh
    row = idx(1);
    col = idx(2)-4;
    num(1,i) = phone_num_arr(row, col);
end

disp(' ')
disp([' Transmitted Data = ',num2str(num)])

% Plots
figure()
freqz(f697.Num, f697.Den)
title('Frequency Response of BPF for 697Hz')
figure()
freqz(f770.Num, f770.Den)
title('Frequency Response of BPF for 770Hz')
figure()
freqz(f852.Num, f852.Den)
title('Frequency Response of BPF for 852Hz')
figure()
freqz(f941.Num, f941.Den)
title('Frequency Response of BPF for 941Hz')
figure()
freqz(f1209.Num, f1209.Den)
title('Frequency Response of BPF for 1209Hz')
figure()
freqz(f1336.Num, f1336.Den)
title('Frequency Response of BPF for 1336Hz')
figure()
freqz(f1477.Num, f1477.Den)
title('Frequency Response of BPF for 1477Hz')

figure()
subplot(7,1,1)
plot(y697)
subplot(7,1,2)
plot(y770)
subplot(7,1,3)
```

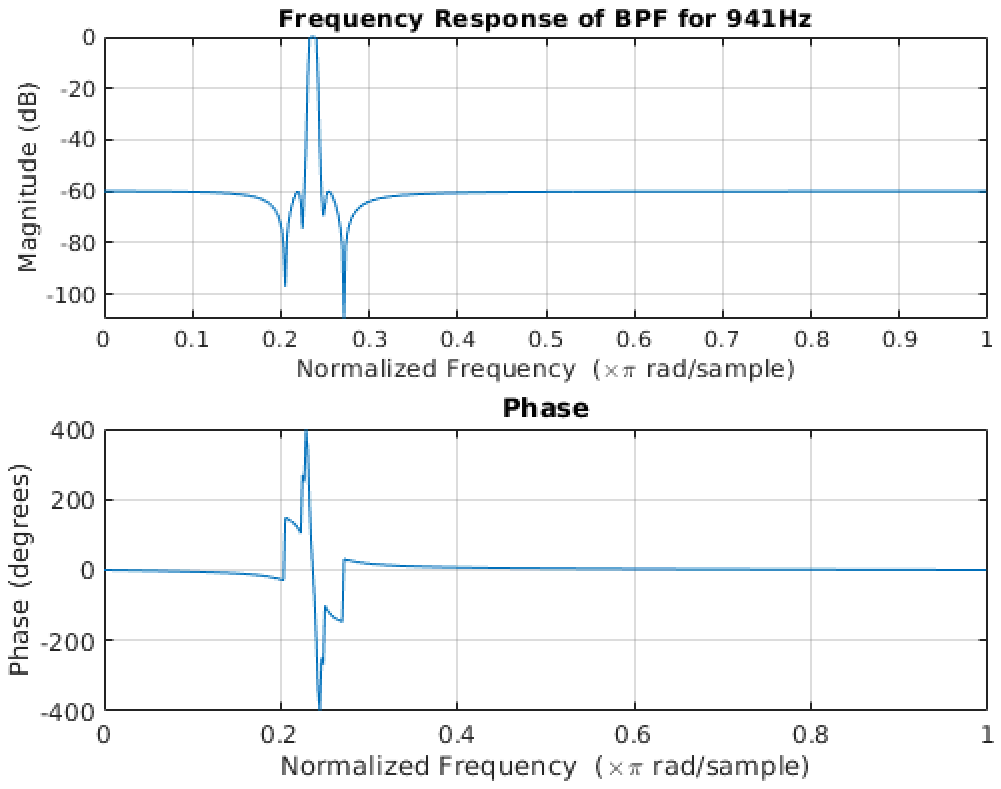
```
plot(y852)
subplot(7,1,4)
plot(y941)
subplot(7,1,5)
plot(y1209)
subplot(7,1,6)
plot(y1336)
subplot(7,1,7)
plot(y1477)
xlabel('Samples')

figure()
for i=1:7
    subplot(7,1,i)
    plot(y(i,:))
end
```

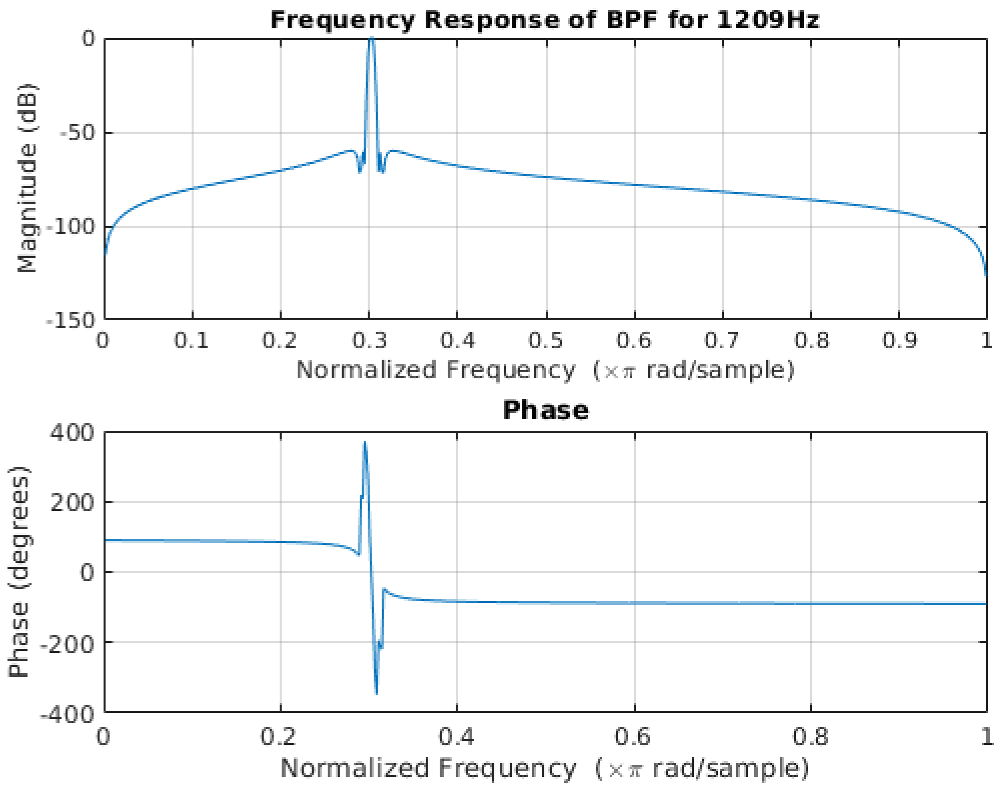
```
 Transmitted Data = 1   2   3   4   5   6   7   8   9   0
```

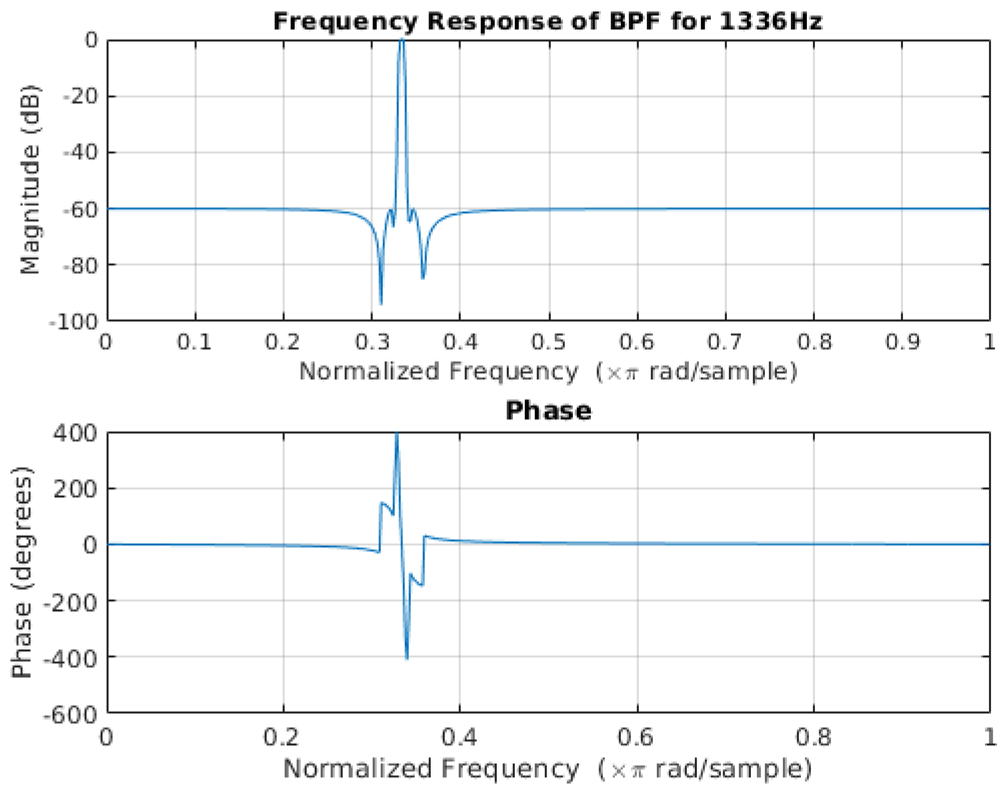


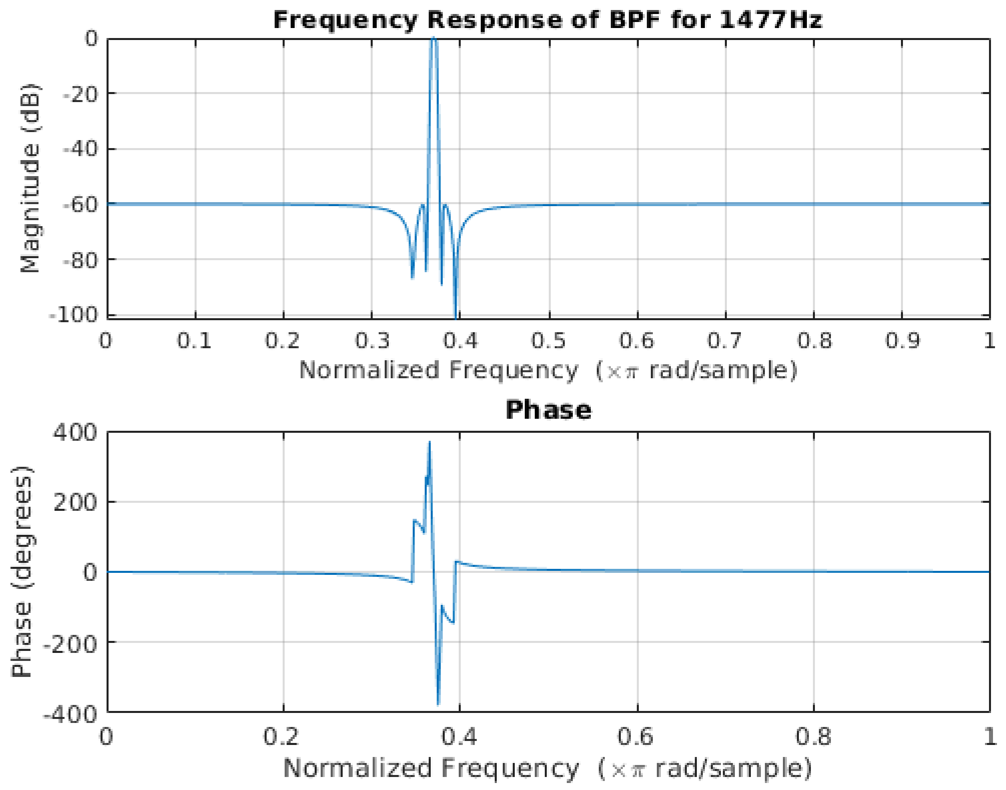

Frequency Response of BPF for 770Hz
Magnitude (dB)
0
-50
-100
-150
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
400
200
0
-200
-400
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)

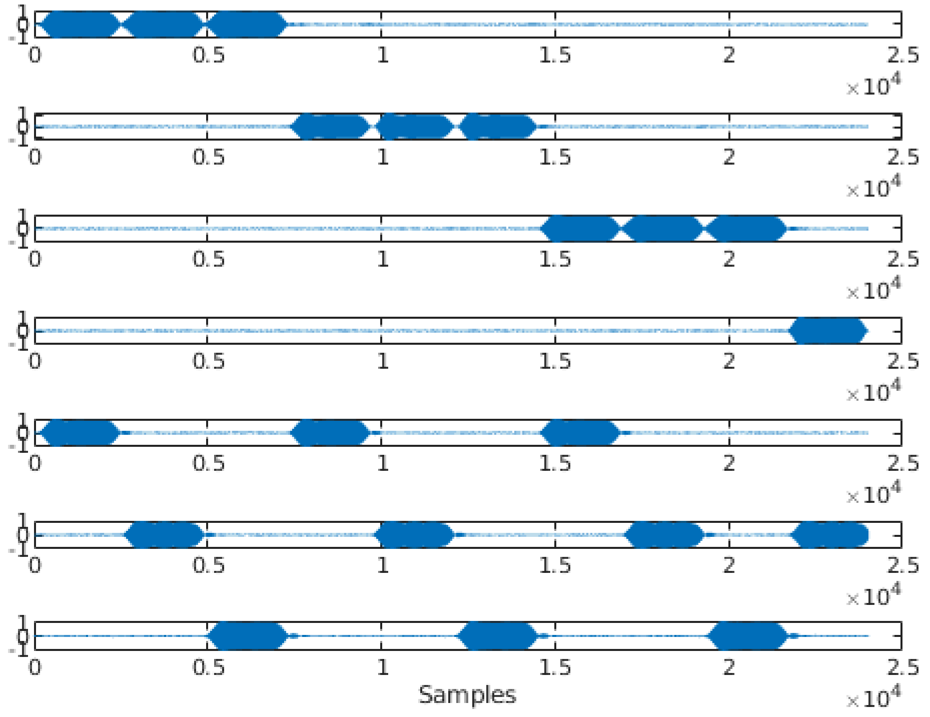

Frequency Response of BPF for 852Hz
Magnitude (dB)
0
-50
-100
-150
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
400
200
0
-200
-400
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)

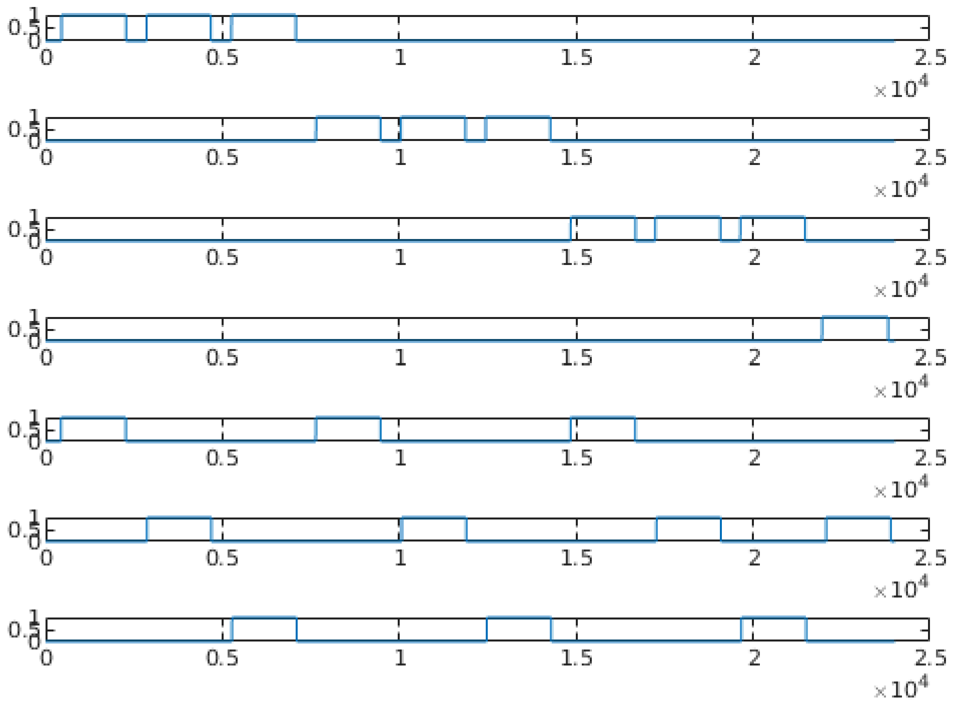

Frequency Response of BPF for 941Hz
Magnitude (dB)
0
-20
-40
-60
-80
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
400
200
0
-200
-400
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)


Frequency Response of BPF for 1209Hz
Magnitude (dB)
0
-50
-100
-150
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
400
200
0
-200
-400
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)


Frequency Response of BPF for 1336Hz
Magnitude (dB)
0
-20
-40
-60
-80
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
400
200
0
-200
-400
-600
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)


Frequency Response of BPF for 1477Hz
Magnitude (dB)
0
-20
-40
-60
-80
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
400
200
0
-200
-400
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)


1
0
-1
0
0.5
1
1.5
2
2.5
×10^4
Samples


1
0.5
0
0
0.5
1
1.5
2
2.5
×10^4