Contents


```
format compact
%Alexander Adams
%3769-0517
```

Lab 9

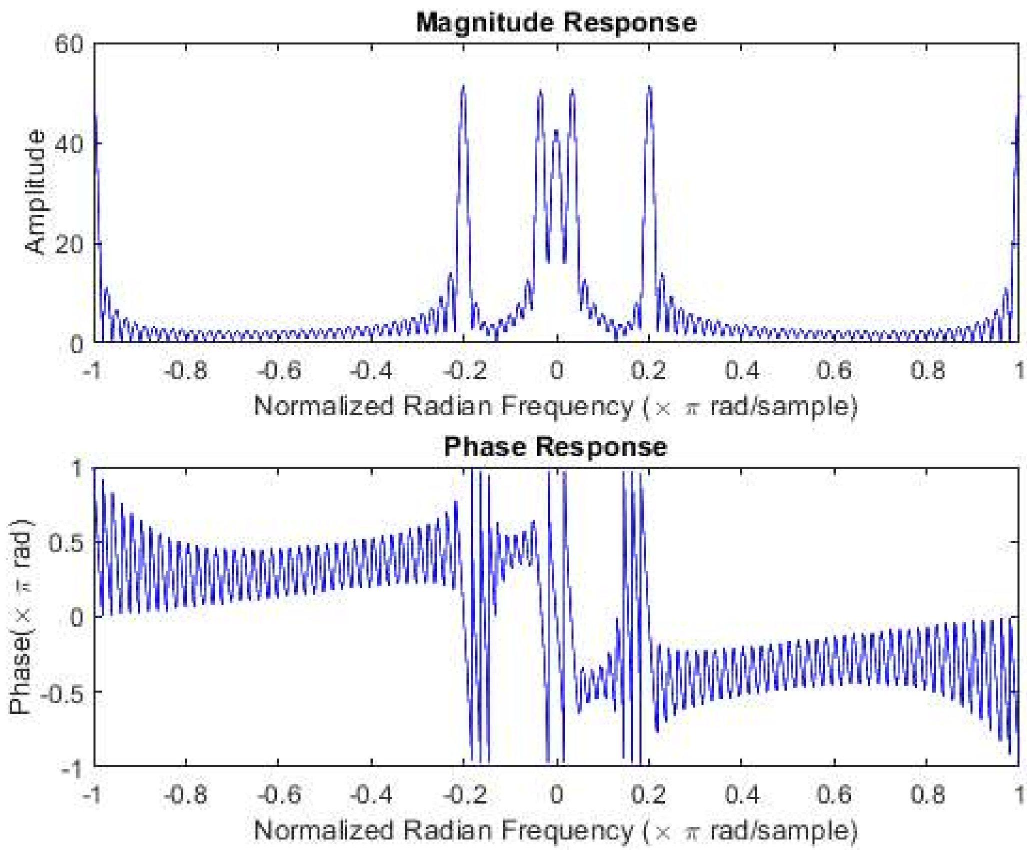
Exercise 9.1: (Effects of DFT size)

9.1a

```
x = [];
for n = 1:100
x(n) = 0.5+cos(pi*(n-1)/30)+cos(pi*(n-1)/5)+cos(pi*(n-1)+(2*pi/3));
end

type dtft
w = -pi:pi/2000:pi;
H = dtft(x,w);

figure(1);
subplot(2,1,1)
plot(w/pi,abs(H),'b-')
hold on;
title('Magnitude Response')
xlabel('Normalized Radian Frequency (\times \pi rad/sample)');
ylabel ('Amplitude');

subplot(2,1,2)
plot(w/pi,angle(H)/pi,'b-')
hold on;
title('Phase Response')
xlabel('Normalized Radian Frequency (\times \pi rad/sample)');
ylabel('Phase(\times \pi rad)');
```

```
function X = dtft(x, w)
```

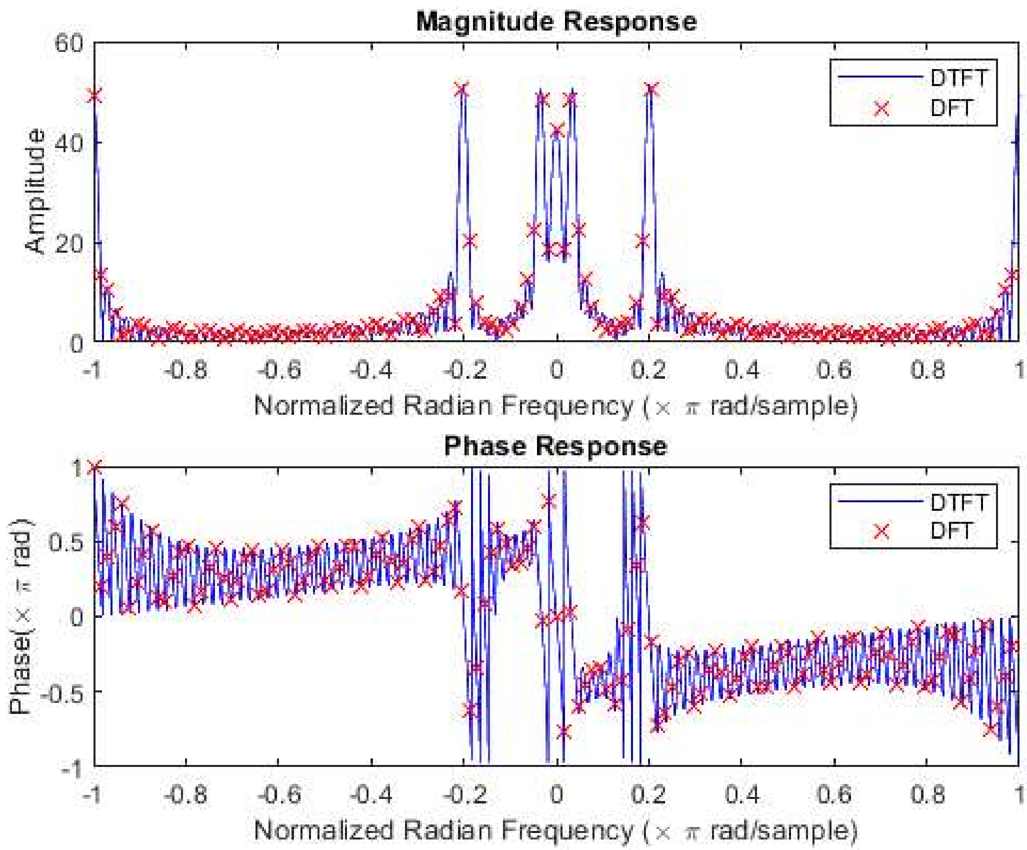
```
%dtft  Calculates X(e^jw) using inputs impulse response vector and
%frequency vector
X = zeros(1,length(w));
for k = 1:length(w)
    for r = 1:length(x)
        X(k)= X(k)+(x(r)*exp(-1j*w(k)*(r-1)));
    end
end
end
```



9.1b

```
wfft = -pi:2*pi/128:pi-2*pi/128;
xfft = fft(x,128);

subplot(2,1,1)
plot(wfft/pi,fftshift(abs(xfft)),'rx');
hold off;
legend ('DTFT','DFT');

subplot(2,1,2)
plot(wfft/pi,fftshift(angle(xfft)/pi),'rx');
hold off;
legend ('DTFT','DFT');

% The DFT coeffecients are indeed frequency samples of the DTFT.
% The sample normalized radian frequency is 2*k/128 for the kth term.
```

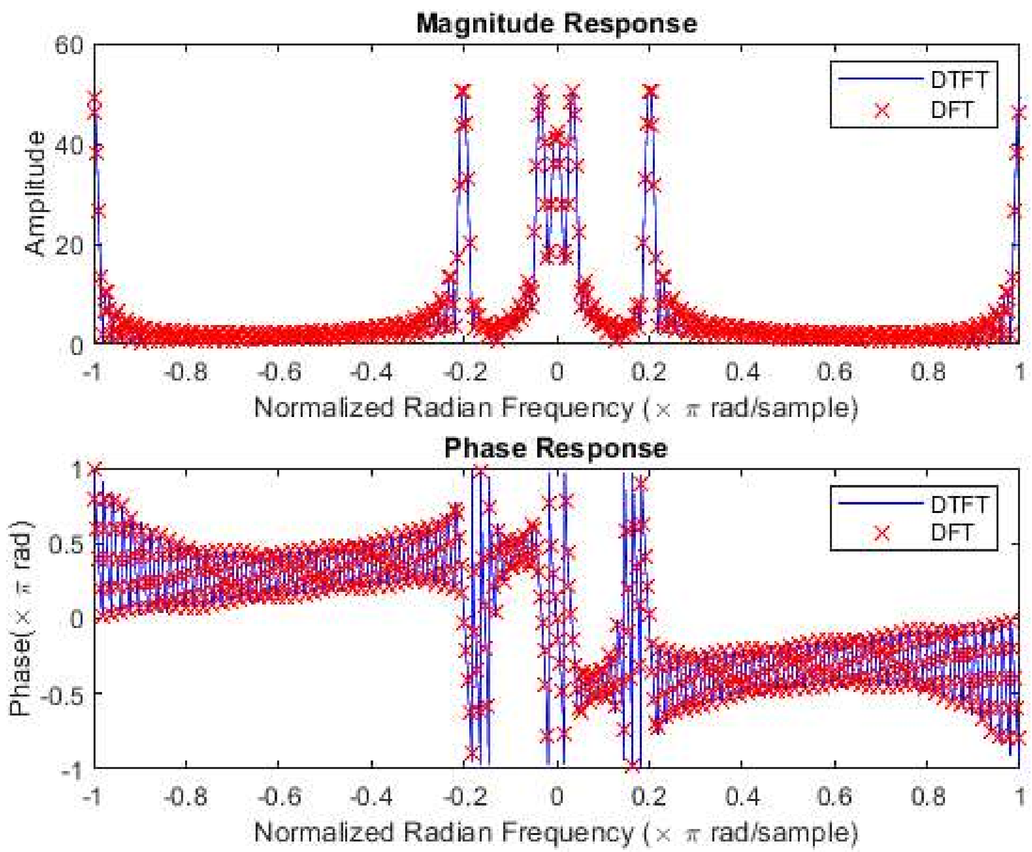
```
% This can be seen as k0 is 0 while k1 is 0.01563 and k(-1) is -0.01562.
% If you plug in k = 0, you get k0 = 0.
% If you plug in k = 1, you get k1 = 2/128 = 0.015625
% If you plug in k = -1, you get k(-1) = -2/128 = -0.015625
```



9.1c

```
wdft = -pi:2*pi/512:pi-2*pi/512;
xdft = fft(x,512);

figure(2);
subplot(2,1,1)
plot(w/pi,abs(H),'b-')
hold on;
plot(wdft/pi,fftshift(abs(xdft)),'rx');
hold off;
legend ('DTFT','DFT');
title('Magnitude Response')
xlabel('Normalized Radian Frequency (\times \pi rad/sample)');
ylabel ('Amplitude');

subplot(2,1,2)
plot(w/pi,angle(H)/pi,'b-')
hold on;
plot(wdft/pi,fftshift(angle(xdft)/pi),'rx');
hold off;
legend ('DTFT','DFT');
title('Phase Response')
```

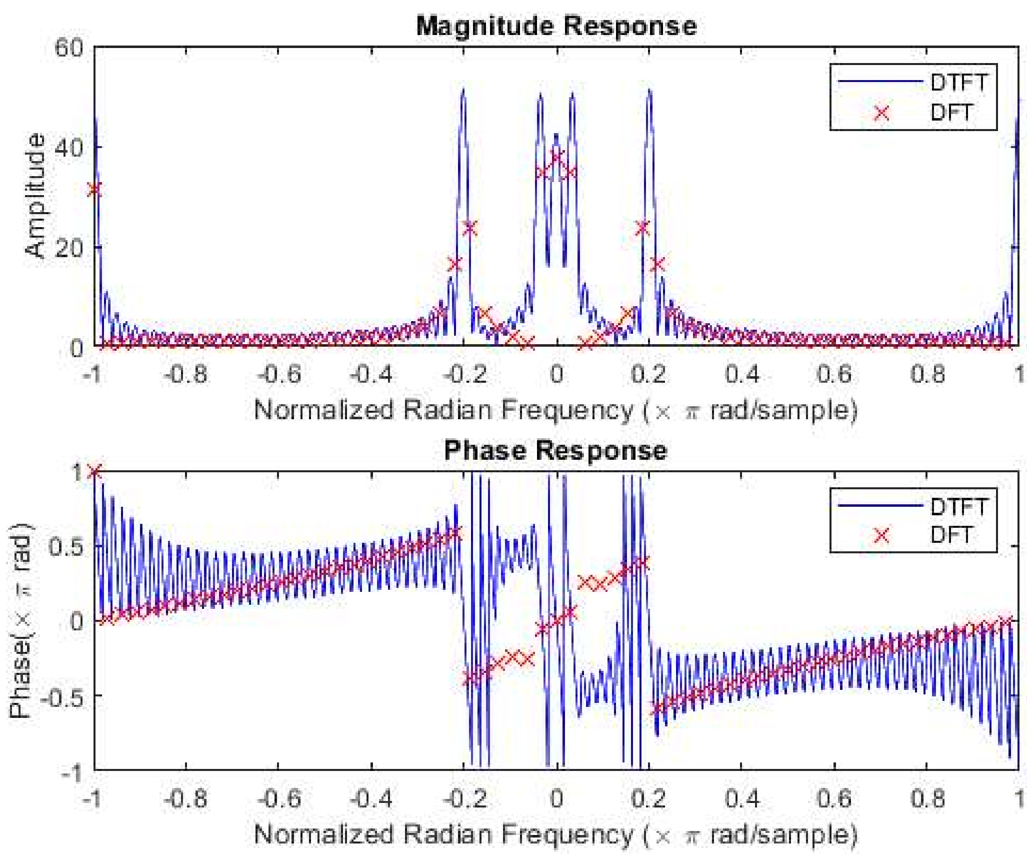
```
xlabel('Normalized Radian Frequency (\times \pi rad/sample)');
ylabel('Phase(\times \pi rad)');

% The sample frequency from (b) is decreased by 4x for every point meaning
% that the frequency is now 2*k/512 for every kth term.  The normalized radian frequency step
 is 0.03906.  As 512/128 = 4 this is how the sample
% frequency factor changes.  If you use an even larger-size DFT, the
% sample frequency will become even smaller.
```



9.1d

```
wdt = -pi:2*pi/64:pi-2*pi/64;
xdt = fft(x,64);

figure(3);
subplot(2,1,1)
plot(w/pi,abs(H),'b-')
hold on;
plot(wdt/pi,fftshift(abs(xdt)),'rx');
hold off;
legend ('DTFT','DFT');
title('Magnitude Response')
xlabel('Normalized Radian Frequency (\times \pi rad/sample)');
ylabel ('Amplitude');

subplot(2,1,2)
plot(w/pi,angle(H)/pi,'b-')
hold on;
```

```
plot(wdt/pi,fftshift(angle(xdt)/pi),'rx');
hold off;
legend ('DTFT','DFT');
title('Phase Response')
xlabel('Normalized Radian Frequency (\times \pi rad/sample)');
ylabel('Phase(\times \pi rad)');

% As shown in the figure, this DFT has a sample frequency that is twice as
% much as the 128 point DFT.  The normalized radian frequency step for is
% 2/64 = 0.03125.  However, many points for the DFT do not match the output
% of the DTFT.  This is due to the size, which is 64 for the DFT,
% being smaller than the size of the signal, which is 100.
```



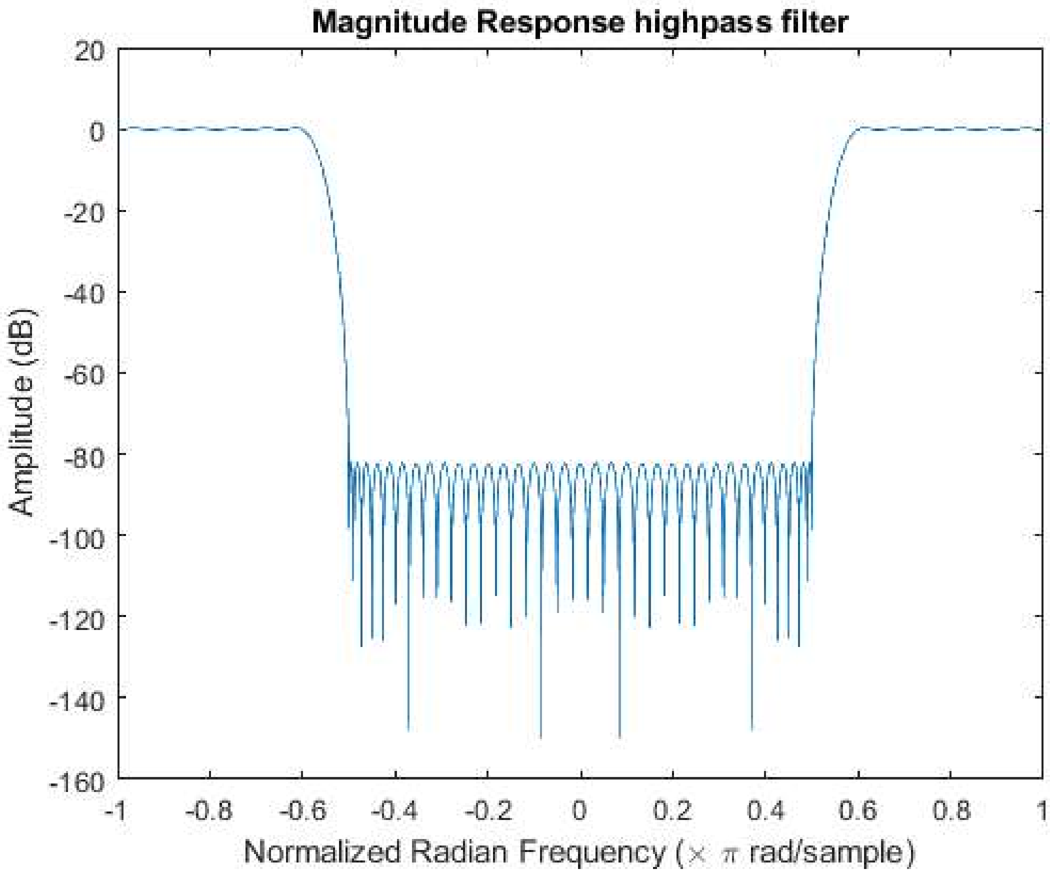
Exercise 9.2: (Frequency-domain analysis using FFT)

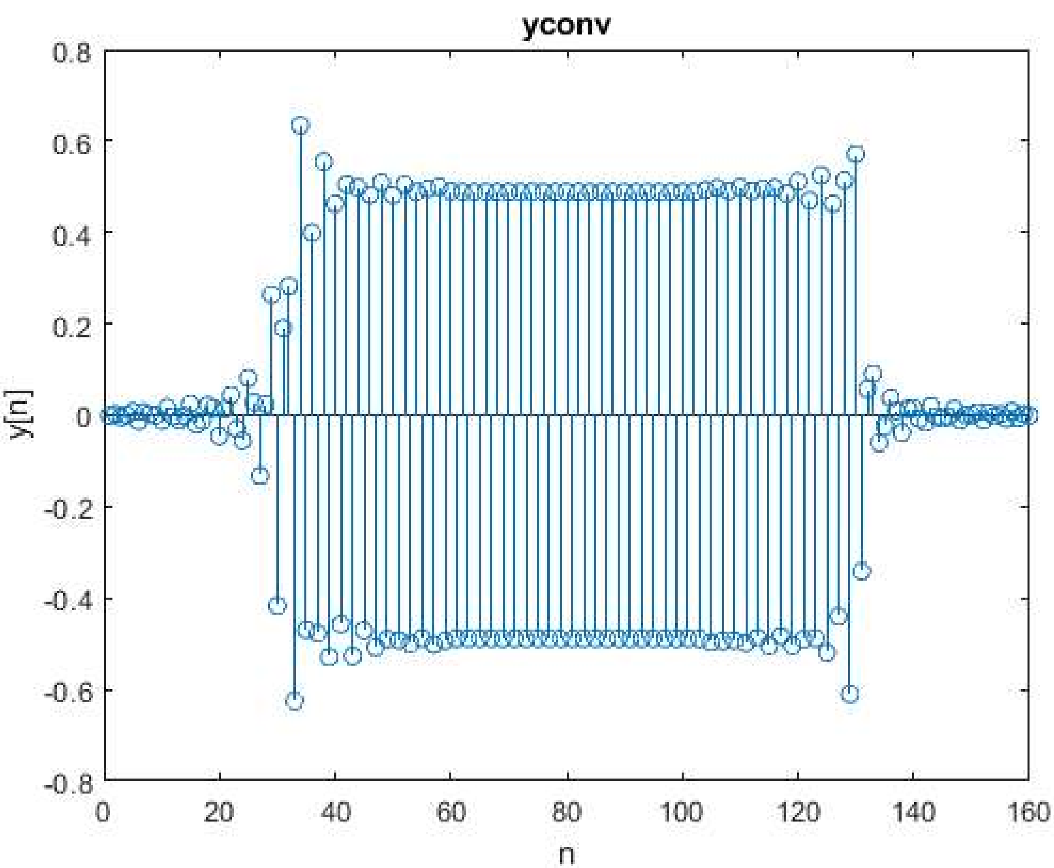
9.2a

```
h
Hfilt = freqz(h,1,w);
figure(4);
plot(w/pi, 20*log10(abs(Hfilt)));
title('Magnitude Response highpass filter');
xlabel('Normalized Radian Frequency (\times \pi rad/sample)');
ylabel('Amplitude (dB)');

yconv = conv(x,h);
figure(5);
stem(1:length(yconv),yconv)
```

```
title('yconv');
xlabel('n');
ylabel('y[n]');

% yconv has length 160, which is 100+61-1 (L+N-1).  The magnitude response
% of the filter shows that the filter oscillates at a low output for normalized
% frequency 0 to 0.5. At around n = 30 in yconv, the output rises and this
% corresponds to half the length of the filter which has length 61.  For
% the next 100 samples (length of the signal), the magnitude of the output
% for yconv is higher than the previous lower frequencies.  This
% demonstrates how the highpass filter affects the output.  The remaining
% 30 points go back to oscillating at low output as it is half the length
% of the filter.
```

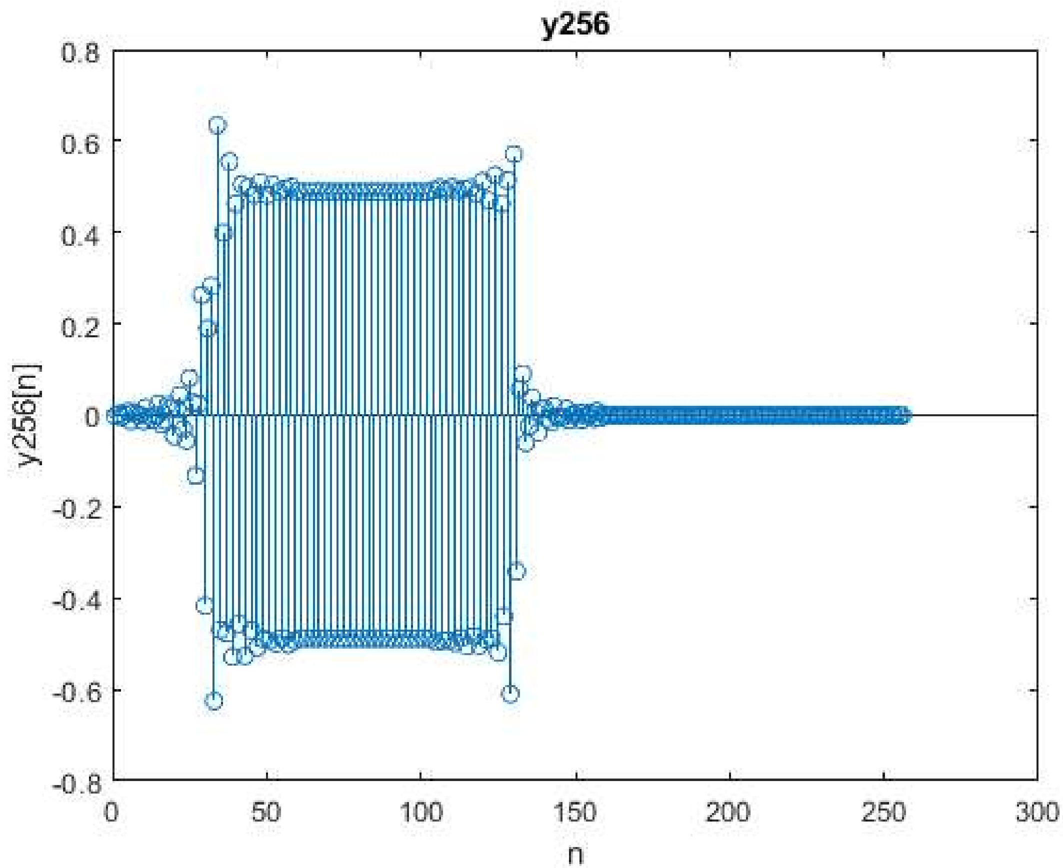
```
h =
  Columns 1 through 7
   -0.0003    0.0019   -0.0045    0.0055   -0.0024   -0.0029    0.0040
  Columns 8 through 14
    0.0012   -0.0055    0.0013    0.0064   -0.0050   -0.0057    0.0094
  Columns 15 through 21
    0.0027   -0.0135    0.0031    0.0159   -0.0118   -0.0150    0.0228
  Columns 22 through 28
    0.0089   -0.0349    0.0049    0.0468   -0.0310   -0.0568    0.0848
  Columns 29 through 35
    0.0634   -0.3109    0.4342   -0.3109    0.0634    0.0848   -0.0568
  Columns 36 through 42
   -0.0310    0.0468    0.0049   -0.0349    0.0089    0.0228   -0.0150
  Columns 43 through 49
   -0.0118    0.0159    0.0031   -0.0135    0.0027    0.0094   -0.0057
  Columns 50 through 56
   -0.0050    0.0064    0.0013   -0.0055    0.0012    0.0040   -0.0029
  Columns 57 through 61
   -0.0024    0.0055   -0.0045    0.0019   -0.0003
```


Magnitude Response highpass filter
Amplitude (dB)
Normalized Radian Frequency (× π rad/sample)
20
0
-20
-40
-60
-80
-100
-120
-140
-160
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


yconv
y[n]
n
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
20
40
60
80
100
120
140
160

9.2b

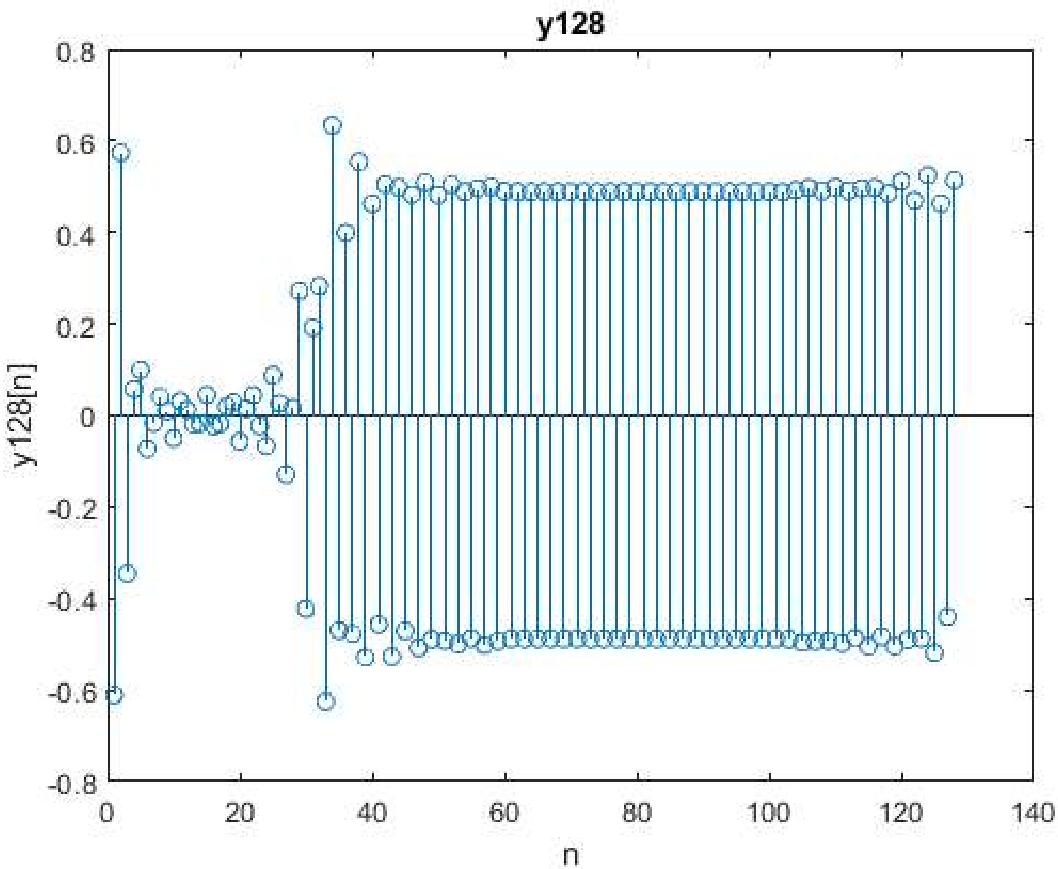
```
xd = fft(x,256);
hd = fft(h,256);
DFTy256 = xd.*hd;
y256 = ifft(DFTy256,256);

figure(6);
stem(1:length(y256),y256)
title('y256');
xlabel('n');
ylabel('y256[n]');

% The length of y256 is 256, hence its name while the length of yconv is
% 160.  The output of each is identical for the
% first 160 points, which is the shorter of the two lengths.  The points
% after this are virtually 0 for y256 while yconv does not have output,
% which shows that the output of each is the same.
```



9.2c

```
xx = fft(x,128);
hx = fft(h,128);
DFTy128 = xx.*hx;
y128 = ifft(DFTy128,128);

figure(7);
stem(1:length(y128),y128)
title('y128');
xlabel('n');
```

```
ylabel('y128[n]');

% y128 is not the same as yconv.  y128 only has length 128 while yconv has
% length 160.  Additionally, the first 30 inputs of y128 are not as low
% output as a highpass filtered equation should function, such as yconv.  The next ~100
% points are basically the same for the higher frequencies however.
```



Published with MATLAB® R2018b