[Connor Dupuis]

Table of Contents

[Friday 1:55pm] - [28944] - [Naoki Sawahashi]

DO NOT REMOVE THE LINE BELOW

```
clear; close all; clc;
```

QUESTION 2: IIR Filters with Difference Equations

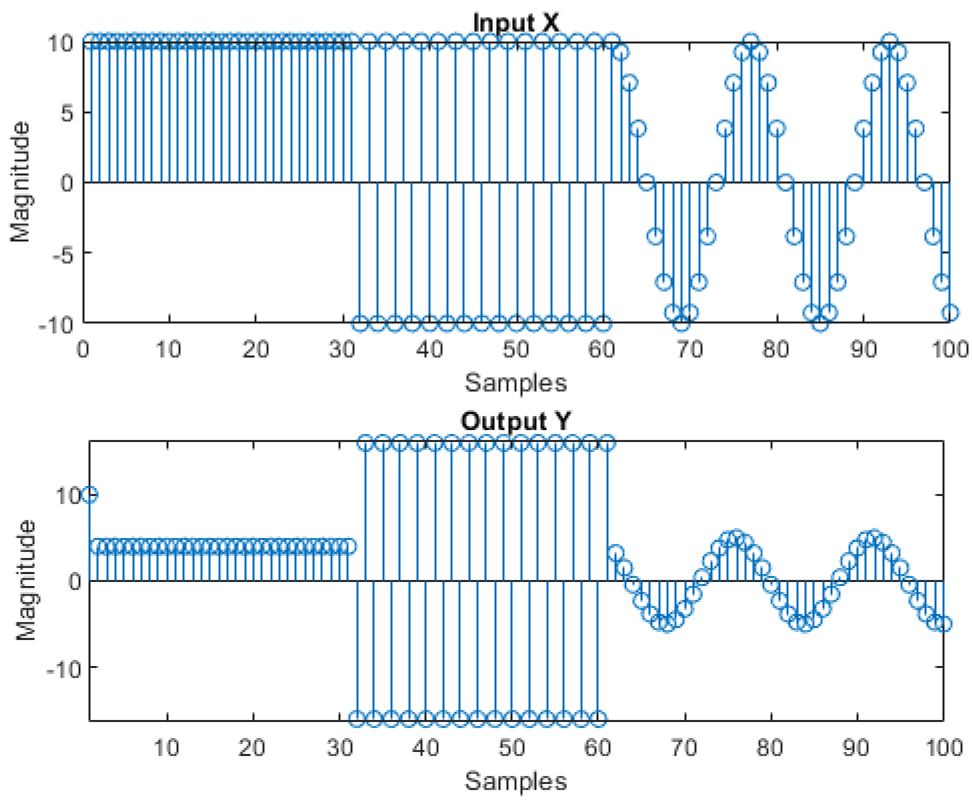
2 (a) Express as a difference equation

```
%   >>> y[n] = x[n] - 0.6x[n-1] <<<   %
```

2 (b) Plot x_1[n] and y_1[n]

```
x1 = [10*ones(1,30) 10*cos(pi.*(0:29)) 10*cos(pi/8.*(0:39))];
y1 = zeros(1, length(x1));

for n = 1:length(x1)
    if (n == 1)
         y1(n) = x1(n);
    else
        y1(n) = x1(n) - 0.6*x1(n-1);
    end
```

```
end

n = 1:1:length(x1);

figure
subplot(2,1,1)
stem(n, x1)
title('Input X')
xlabel('Samples')
ylabel('Magnitude')
subplot(2,1,2)
stem(n,y1)
title('Output Y')
xlabel('Samples')
ylabel('Magnitude')
axis equal
```



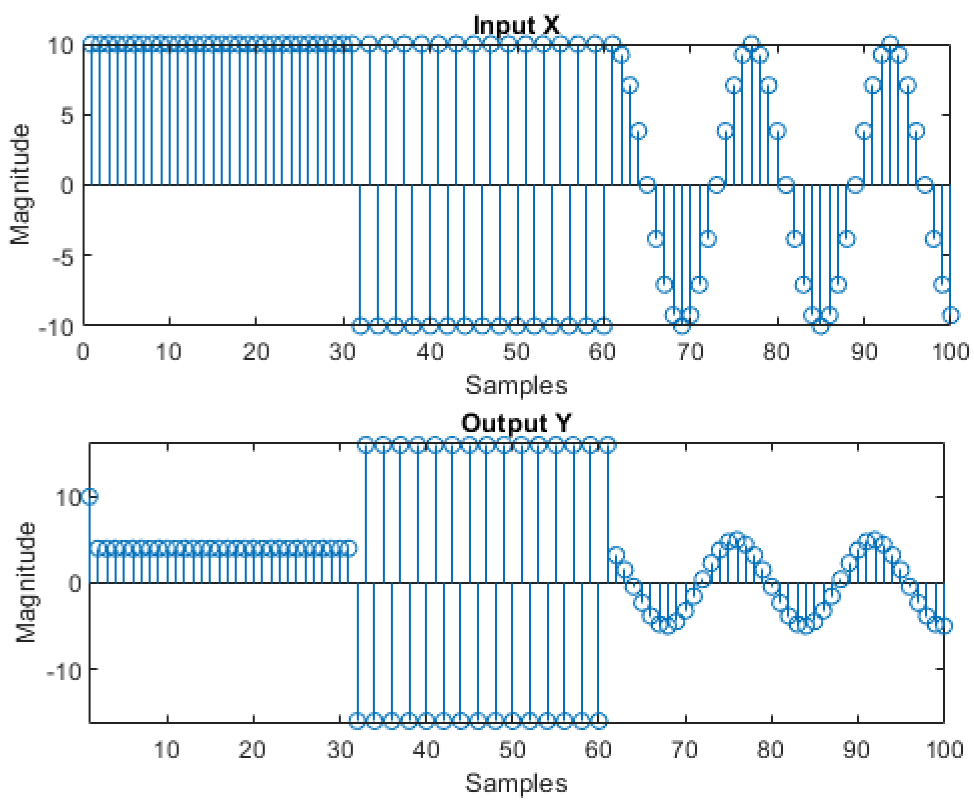
2 (c) Express as a difference equation

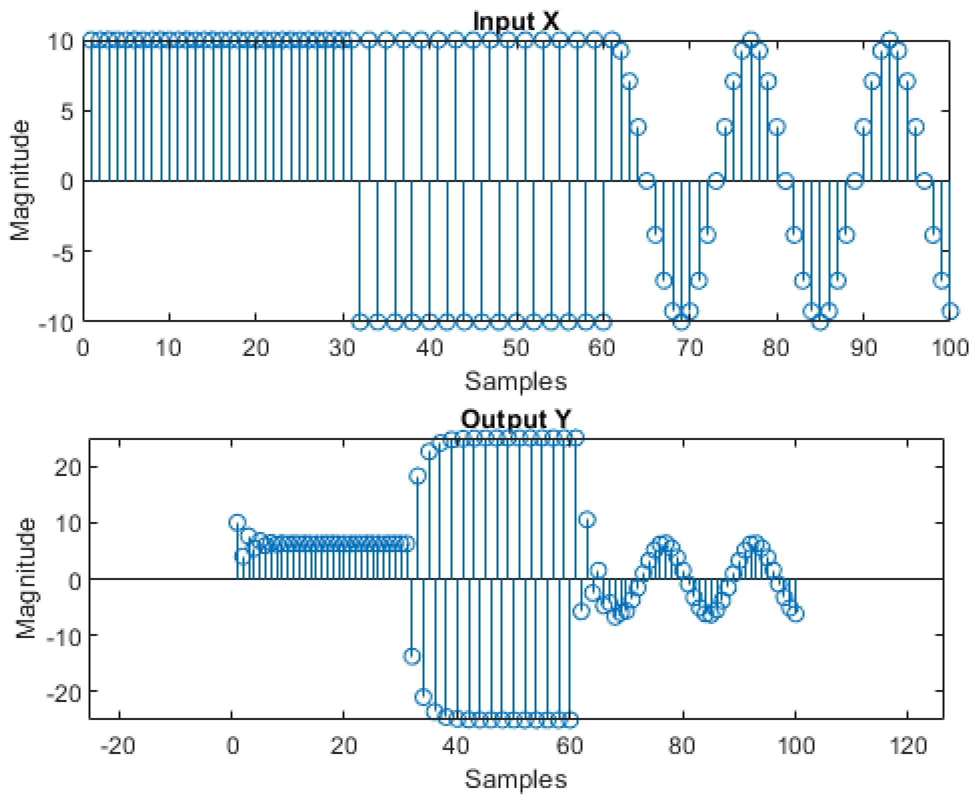
```
%   >>> y[n] = x[n] - 0.6y[n-1] <<<   %
```

2 (d) Plot x_2[n] and y_2[n]

```
x2 = [10*ones(1,30) 10*cos(pi.*(0:29)) 10*cos(pi/8.*(0:39))];
```

```
y2 = zeros(1, length(x2));

for n = 1:length(x2)
    if (n == 1)
        y2(n) = x2(n);
    else
        y2(n) = x2(n) - 0.6*y2(n-1);
    end
end

n = 1:1:length(x2);

figure
subplot(2,1,1)
stem(n, x2)
title('Input X')
xlabel('Samples')
ylabel('Magnitude')
subplot(2,1,2)
stem(n,y2)
title('Output Y')
xlabel('Samples')
ylabel('Magnitude')
axis equal
```





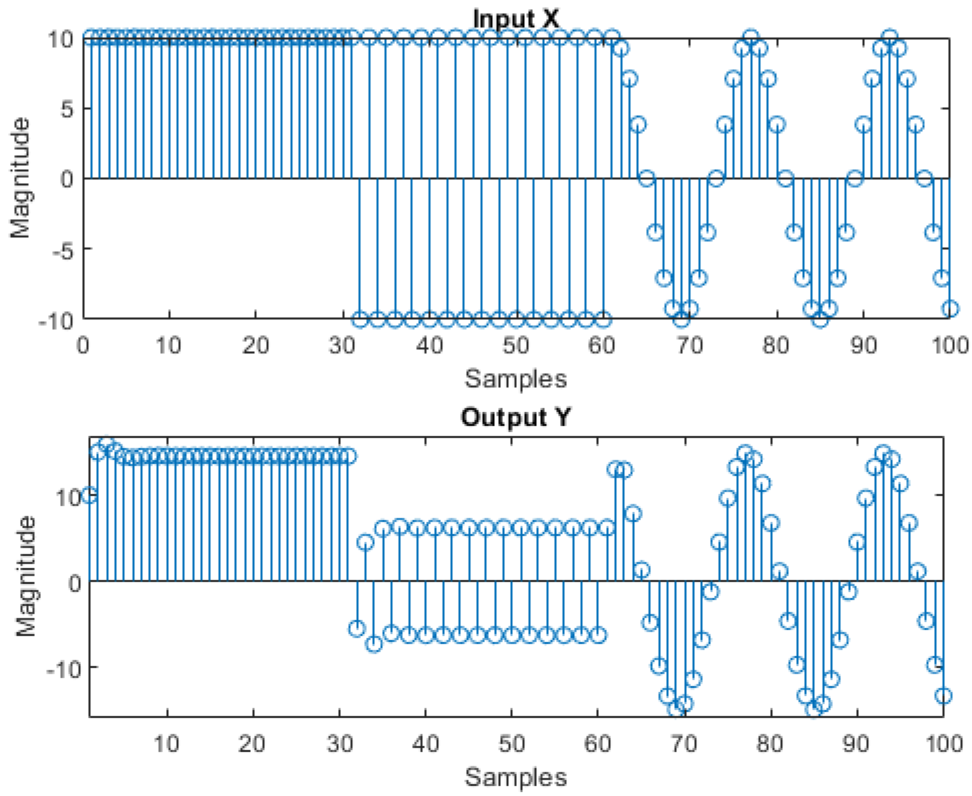
2 (e) Express as a difference equation

```
%   >>> y[n] = x[n] + 0.09x[n-2] + 0.5y[n-1] - 0.25y[n-2] <<<   %
```

2 (f) Plot x_3[n] and y_3[n]

```
x3 = [10*ones(1,30) 10*cos(pi.*(0:29)) 10*cos(pi/8.*(0:39))];

y3 = zeros(1, length(x3));

for n = 1:length(x3)
    if (n == 1)
        y3(n) = x3(n);
    elseif (n == 2)
        y3(n) = x3(n) + 0.5*y3(n-1);
    else
        y3(n) = x3(n) + 0.09*x3(n-2) + 0.5*y3(n-1) - 0.25*y3(n-2);
    end
end

n = 1:1:length(x3);

figure
subplot(2,1,1)
stem(n, x3)
```

```
title('Input X')
xlabel('Samples')
ylabel('Magnitude')
subplot(2,1,2)
stem(n,y3)
title('Output Y')
xlabel('Samples')
ylabel('Magnitude')
axis equal
```



QUESTION 3: High-Order IIR Filters

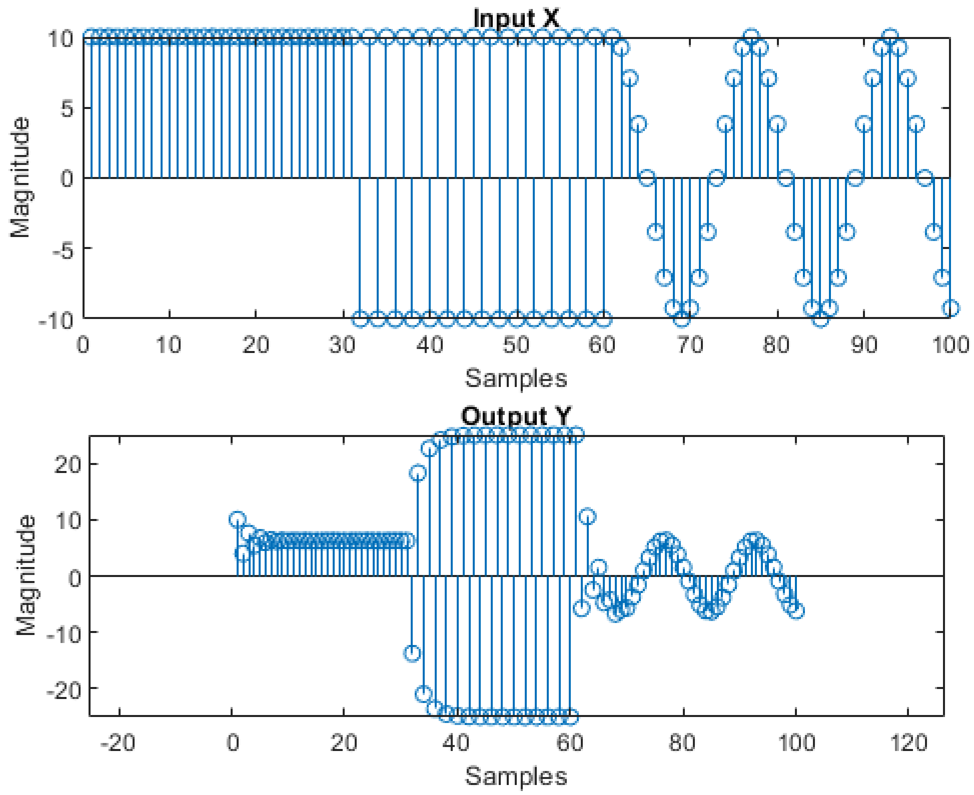
3 (a) Describe input

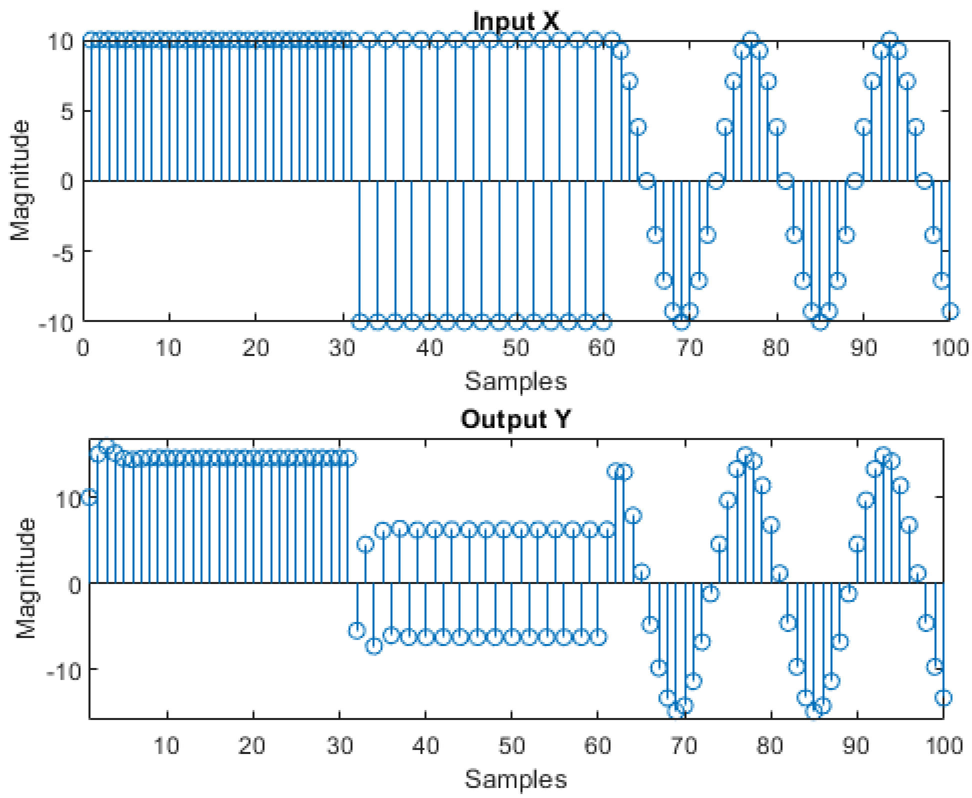
```
n = 1:1:9999;
x = cos((pi/30000) * n.^2);
fs = 4000;

soundsc(x, fs);
%   >>> The signal moves across the frequency spectrum starting from
 low frequency towards the higher ones <<<   %
```

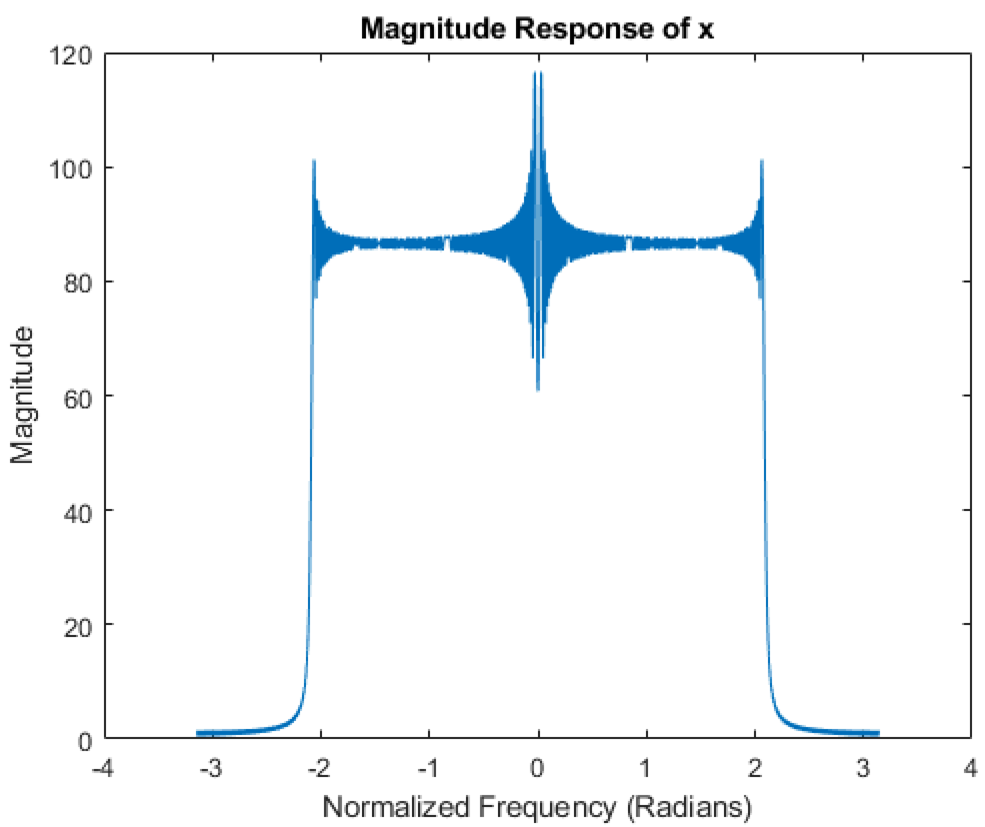
3 (b) Plot the chirp signal in freq-domain

```
w = -pi:pi/2000:pi-pi/2000;
```

```
H = DTFT(x,w);

figure
plot(w,abs(H))
title('Magnitude Response of x')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
```

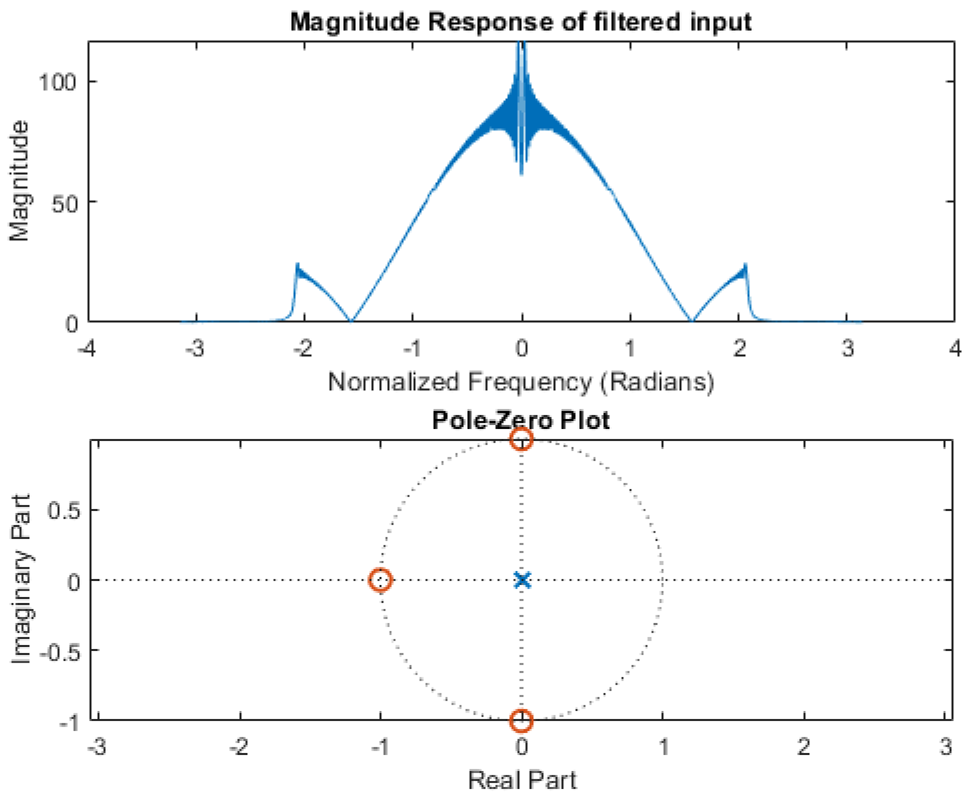



Input X
Magnitude
Samples
Output Y
Magnitude
Samples


Magnitude Response of x
Magnitude
Normalized Frequency (Radians)

3 (c) First-order filter

```
b = 1/4*[1 1 1 1];
a = 1;
y1 = filter(b,a,x);
H1 = DTFT(y1,w);

figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of filtered input')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b,a)
```

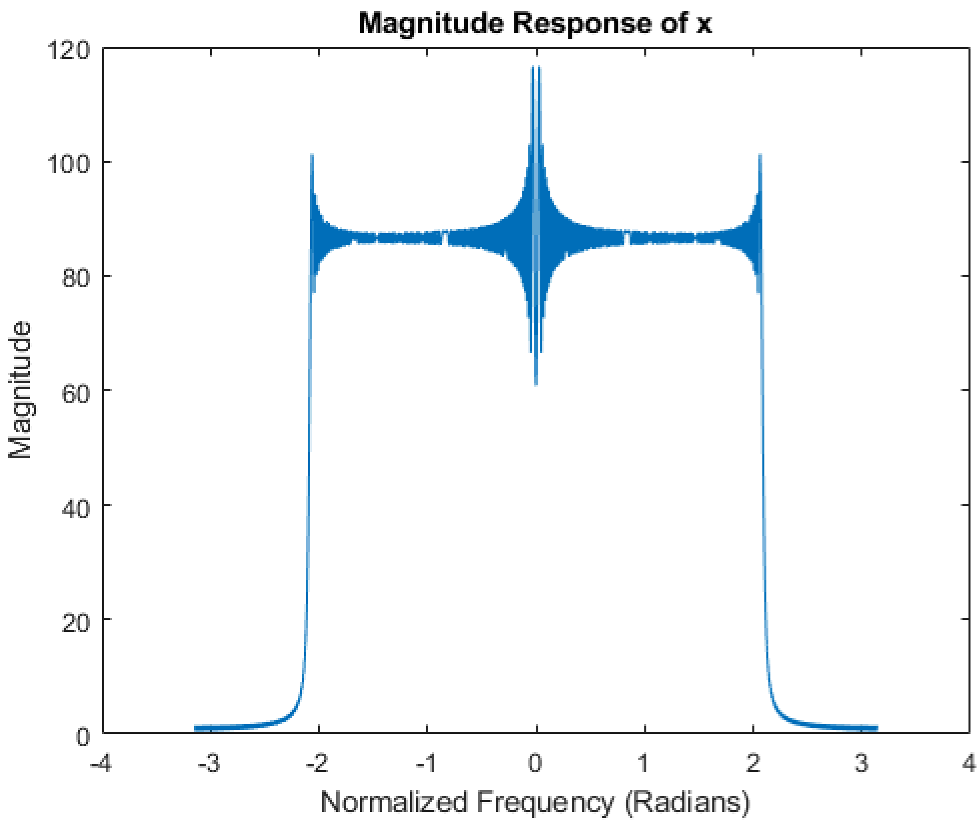


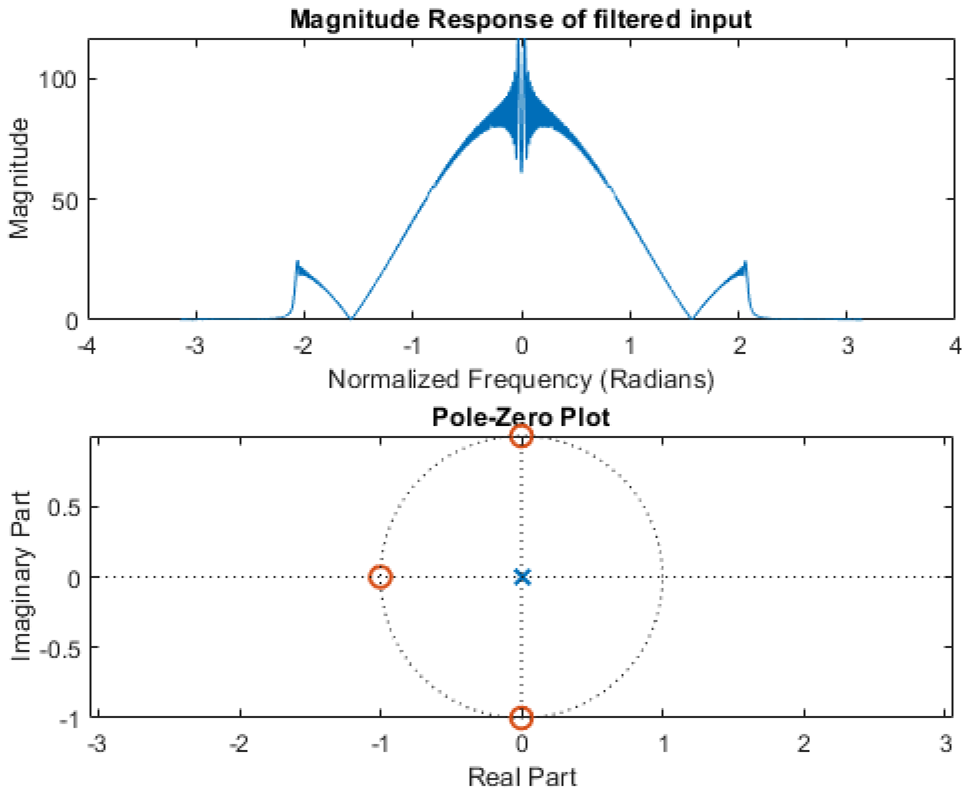
3 (d) Describe output

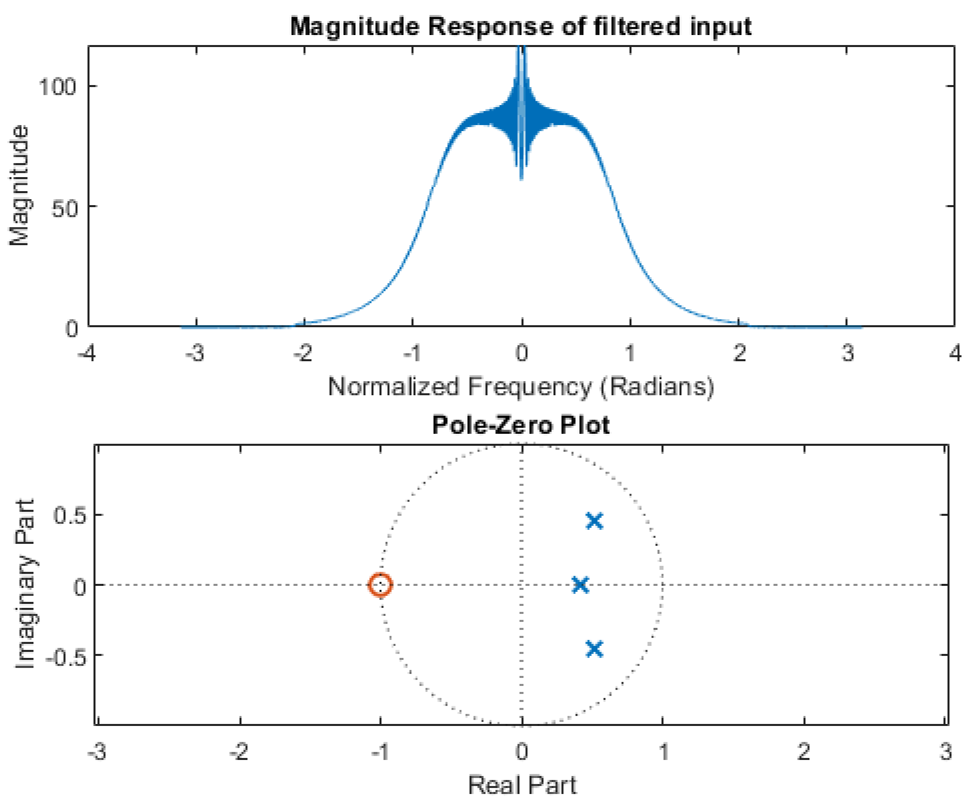
```
soundsc(y1, fs);

% The signal again sweeps across the frequency spectrum similar to the
 last one, but
% this time it decreases in volume, then near the end it spikes up a
 little
% again.
```

3 (e) 3rd order Butterworth filter

```
[b, a] = butter(3, 1/4);

y2 = filter(b,a,x);
H1 = DTFT(y2,w);

figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of filtered input')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b,a)
```




Magnitude Response of filtered input
Magnitude
100
50
0
-4
-3
-2
-1
0
1
2
3
4
Normalized Frequency (Radians)
Pole-Zero Plot
Imaginary Part
0.5
0
-0.5
-1
-3
-2
-1
0
1
2
3
Real Part

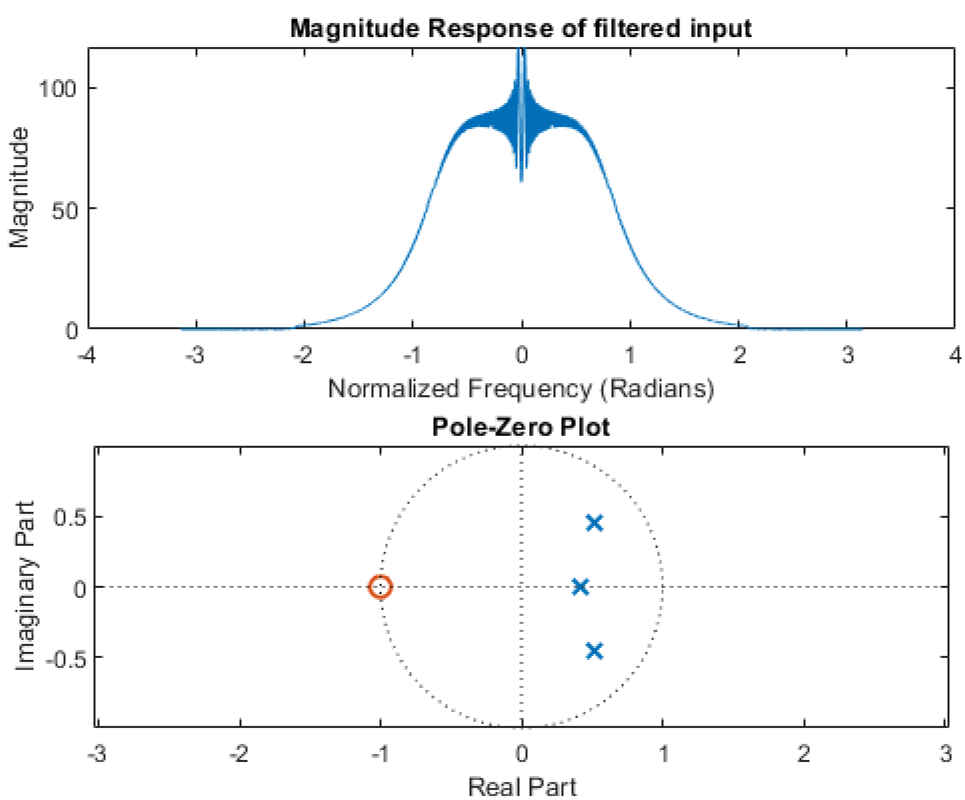

Magnitude Response of filtered input
Magnitude
100
50
0
-4
-3
-2
-1
0
1
2
3
4
Normalized Frequency (Radians)
Pole-Zero Plot
Imaginary Part
0.5
0
-0.5
-3
-2
-1
0
1
2
3
Real Part

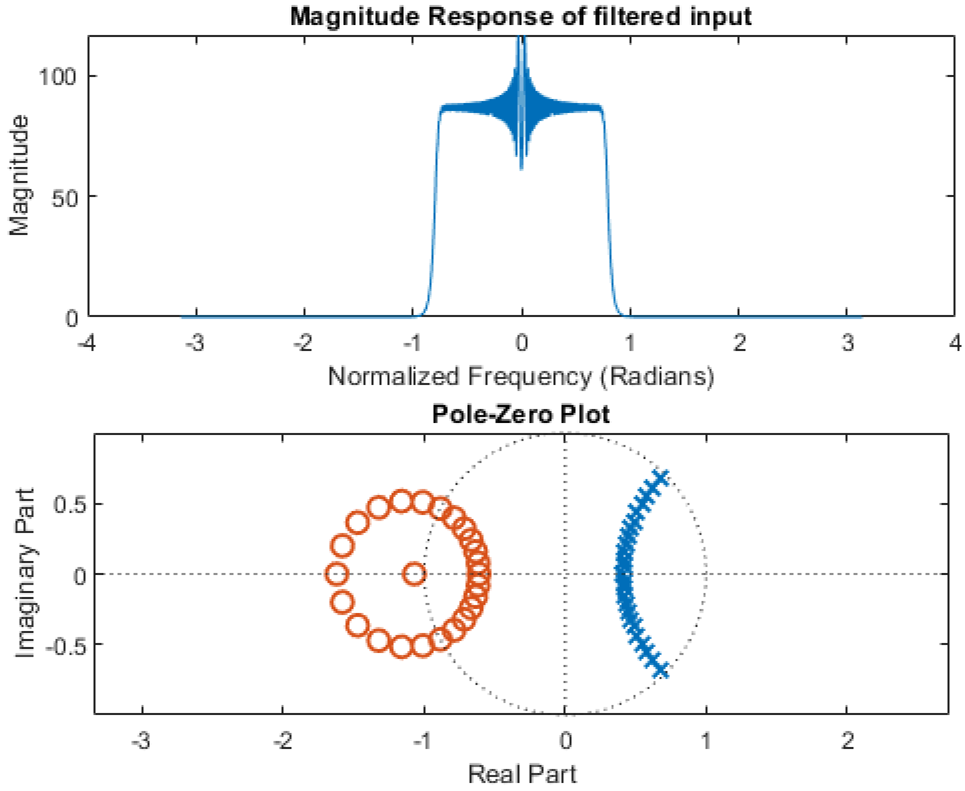
3 (f) Describe output

```
soundsc(y2, fs);

% The signal sounds different because the filter is a lowpass and is
 removing frequencies at the higher end.
% This filter has no spike at the end like the previous.
```

3 (g) 25th order Butterworth filter

```
[b, a] = butter(25, 1/4);

y3 = filter(b,a,x);
H1 = DTFT(y3,w);

figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of filtered input')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b,a)
```





3 (h) Describe output

```
soundsc(y3, fs);

% This filter is a lowpass with a quicker cutoff than the last filter
% resulting in more frequencies being attenuated.
```

3 (i) Discussion

The buttersworth filter has the smoothest frequency response and a simple transfer function making it relatively easy to implement. The last filter would be the 'best' Higher-order filters are harder to produce and maintain in real world implementations.

ALL FUNCTIONS SUPPORTING THIS CODE %%

```
function pzplot(b,a)
% PZPLOT(B,A)  plots the pole-zero plot for the filter described by
% vectors A and B.  The filter is a "Direct Form II Transposed"
% implementation of the standard difference equation:
%
%    a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
%                          - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```

```
%

    % MODIFY THE POLYNOMIALS TO FIND THE ROOTS
    b1 = zeros(max(length(a),length(b)),1); % Need to add zeros to get
 the right roots
    a1 = zeros(max(length(a),length(b)),1); % Need to add zeros to get
 the right roots
    b1(1:length(b)) = b;    % New a with all values
    a1(1:length(a)) = a;    % New a with all values

    % FIND THE ROOTS OF EACH POLYNOMIAL AND PLOT THE LOCATIONS OF THE
 ROOTS
    h1 = plot(real(roots(a1)), imag(roots(a1)));
    hold on;
    h2 = plot(real(roots(b1)), imag(roots(b1)));
    hold off;

    % DRAW THE UNIT CIRCLE
    circle(0,0,1)

    % MAKE THE POLES AND ZEROS X's AND O's

 set(h1, 'LineStyle', 'none', 'Marker', 'x', 'MarkerFaceColor','none', 'linewidth'
 1.5, 'markersize', 8);

 set(h2, 'LineStyle', 'none', 'Marker', 'o', 'MarkerFaceColor','none', 'linewidth'
 1.5, 'markersize', 8);
    axis equal;

    % DRAW VERTICAL AND HORIZONTAL LINES
    xminmax = xlim();
    yminmax = ylim();
    line([xminmax(1) xminmax(2)],[0 0], 'linestyle', ':', 'linewidth',
 0.5, 'color', [1 1 1]*.1)
    line([0 0],[yminmax(1) yminmax(2)], 'linestyle', ':', 'linewidth',
 0.5, 'color', [1 1 1]*.1)

    % ADD LABELS AND TITLE
    xlabel('Real Part')
    ylabel('Imaginary Part')
    title('Pole-Zero Plot')

end


function circle(x,y,r)
% CIRCLE(X,Y,R)  draws a circle with horizontal center X, vertical
 center
% Y, and radius R.
%

    % ANGLES TO DRAW
    ang=0:0.01:2*pi;
```

```
    % DEFINE LOCATIONS OF CIRCLE
    xp=r*cos(ang);
    yp=r*sin(ang);

    % PLOT CIRCLE
    hold on;
    plot(x+xp,y+yp, ':', 'linewidth', 0.5, 'color', [1 1 1]*.1);
    hold off;

end



function H = DTFT(x,w)
% DTFT(X,W)  compute the Discrete-time Fourier Transform of signal X
% acroess frequencies defined by W.

    H = zeros(length(w),1);
    for nn = 1:length(x)
        H = H + x(nn).*exp(-1j*w.'*(nn-1));
    end

end


function xs = shift(x, s)
% SHIFT(x, s) shifts signal x by s such that the output can be defined
 by
% xs[n] = x[n - s]

    % INITIALIZE THE OUTPUT
    xs = zeros(length(x), 1);

    for n = 1:length(x)
        % CHECK IF THE SHIFT IS OUT OF BOUNDS FOR THIS SAMPLE
        if n-s > 0 && n-s < length(x)
            % SHIFT DATA
            xs(n) = x(n-s);
        end
    end

end
```

Published with MATLAB® R2020a