HW4

Question 4 is a bonus question.

Questions 1 and 2 ask for computing some coefficients in frequency or time/sample domain. You can either type the values of these coefficients or make stem plots that show those values at each frequency or sample number. Note that plotting complex values includes separate magnitude and phase plots.

You may find the Matlab demo posted on Module 4 helpful.

Each question has 25 points.

1

Write a function X = dfs(x, N) to compute the Discrete Fourier Series (DFS) coefficients of a periodic 1-D signal, where:

x is one period of the signal

N is the fundamental period.

Let x(n) = [0 1 2 3], N = 4.

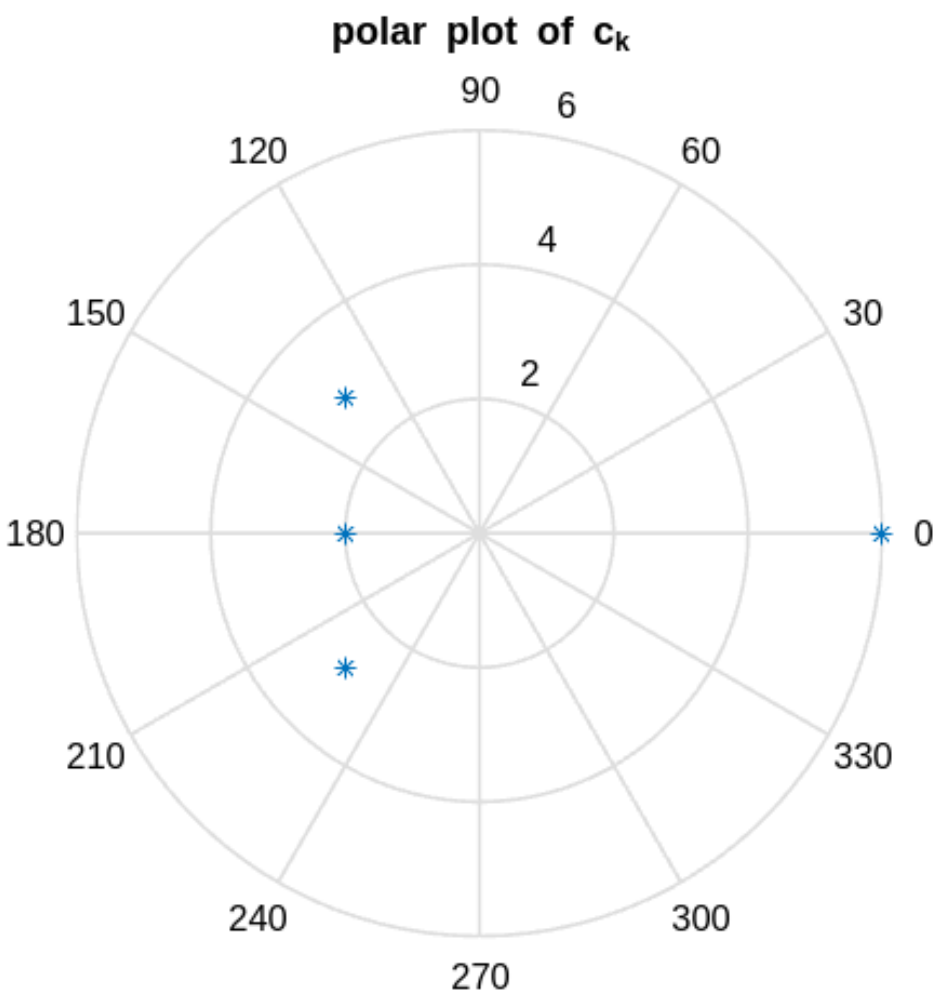
Find the DFS coefficients using your function.

function is called here but defined at the bottom.

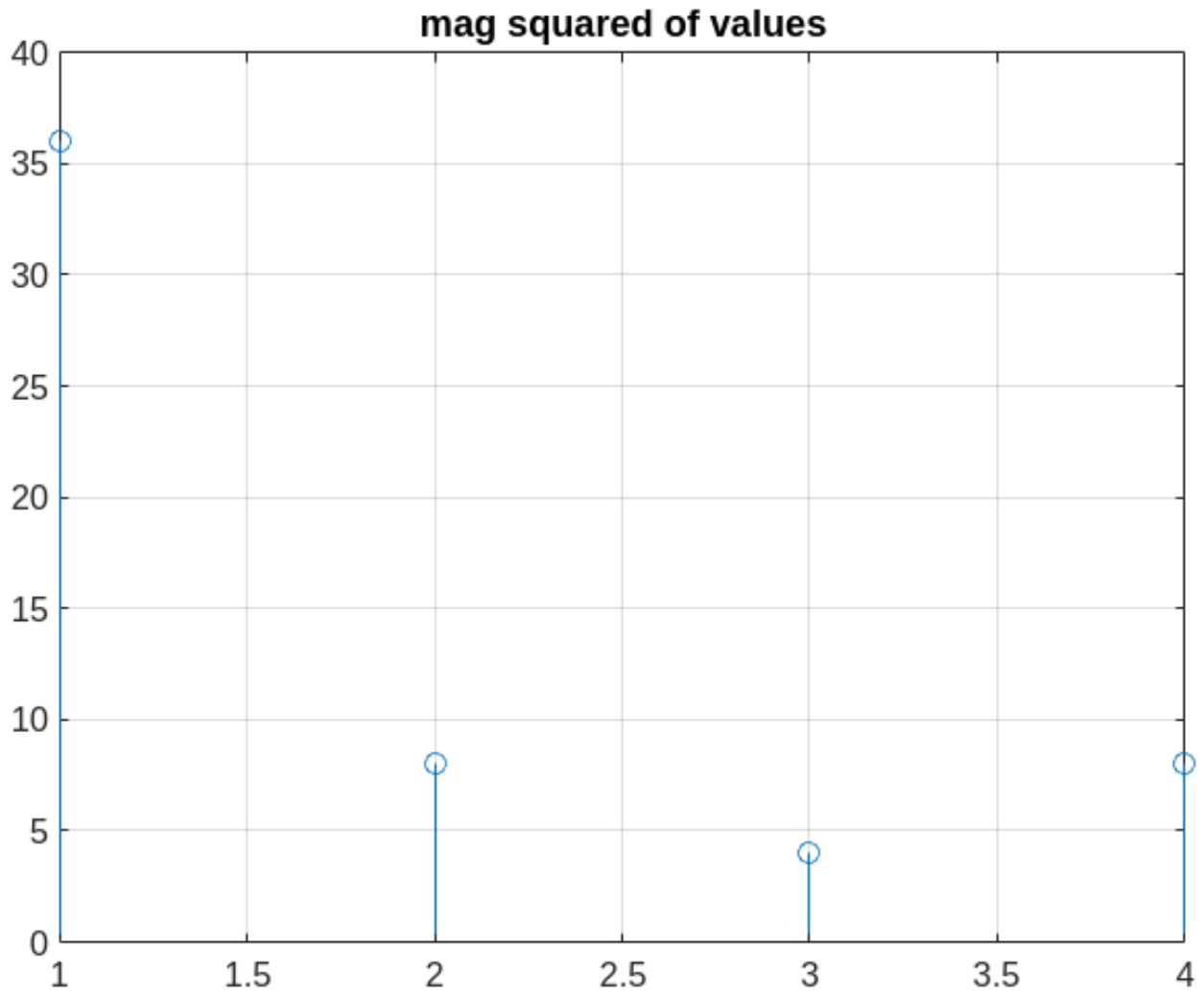
```
clear all
clc
figure
x = [0, 1, 2, 3];
N = 4;
coeffs = dfs(x,N);
disp(coeffs)
```

```
   6.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 - 0.0000i  -2.0000 - 2.0000i
```

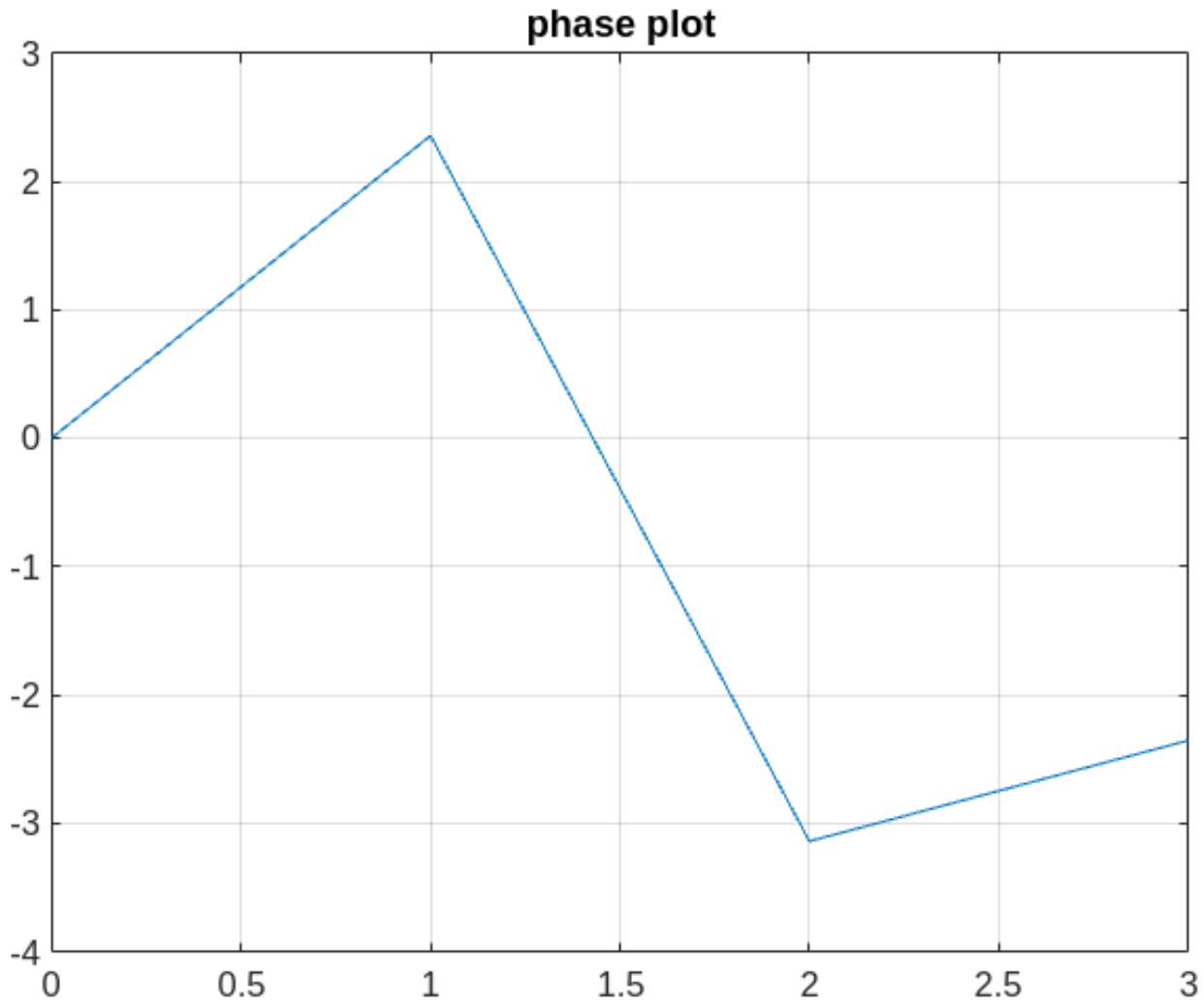
```
polar(angle(coeffs),abs(coeffs),'*')
title("polar plot of c_k")
```



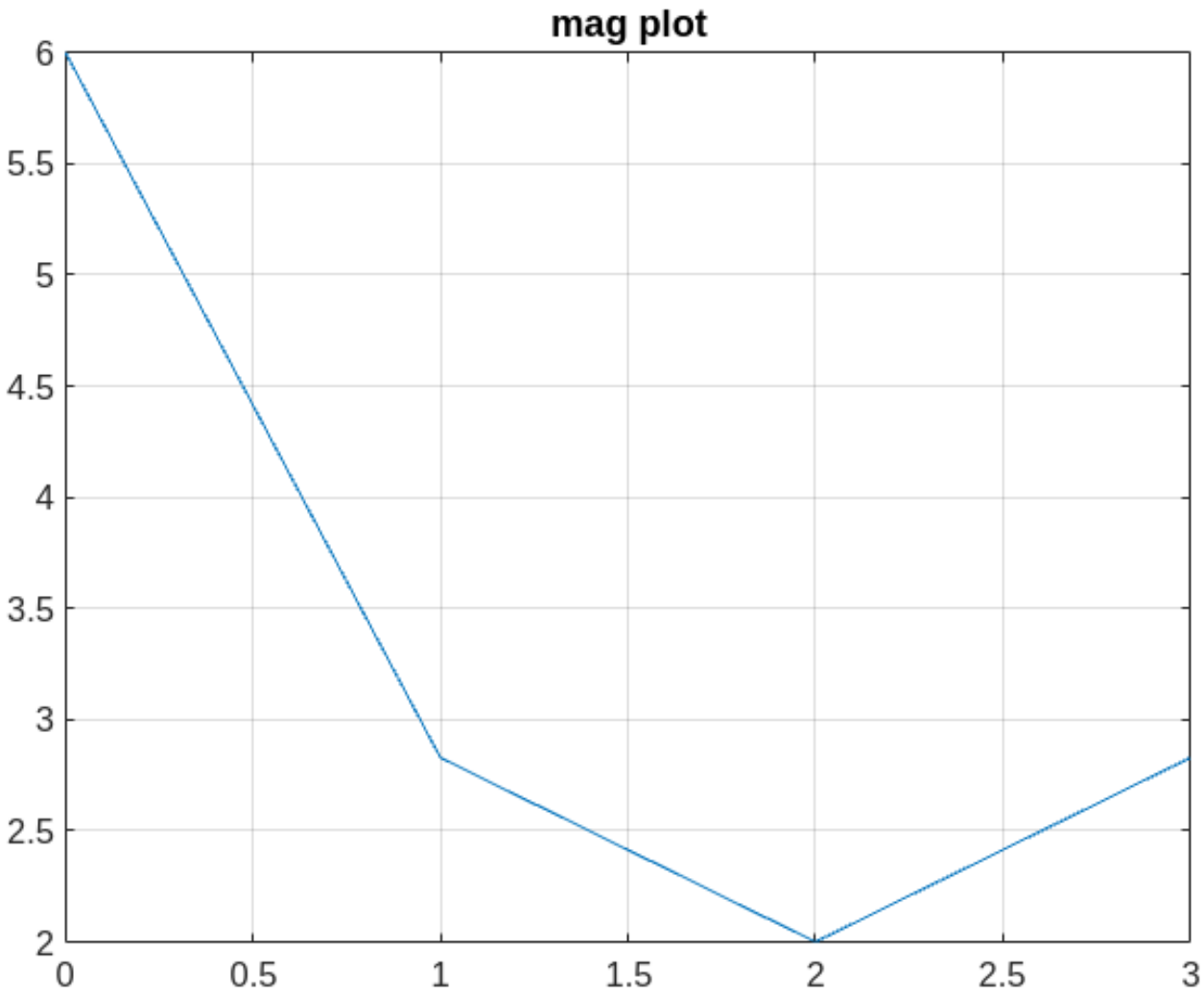

```
stem(coeffs.*conj(coeffs))
grid on
title("mag squared of values")
```




```
plot(x,angle(coeffs))
grid on
title("phase plot")
```




```
plot(x,abs(coeffs))
grid on
title("mag plot")
```



2

Implement another function x = idfs(X, N) to compute the Inverse Discrete Fourier Series (IDFS), where :

X is the DFS coefficient array

N is the fundamental period.

Use the function idfs to retrieve back the original signal x(n) from the results of the previous part.

```
icoeffs = idfs(coeffs,N);
disp(real(icoeffs))
```

```
    0.0000    1.0000    2.0000    3.0000
```

3

The difference equation of a filter is given as $y(n) = -0.8y(n-1) + 0.2x(n)$

Find its frequency response,

$y(n) = -0.8y(n-1) + 0.2x(n)$

$y(n) + 0.8y(n-1) = 0.2x(n)$

$Y(z) + 0.8z^{-1}Y(z) = 0.2X(z)$

$Y(z)(1 + .8z^{-1}) = 0.2X(z)$

$\frac{Y(z)}{X(z)} = H(z) = \frac{.2}{1 + .8z^{-1}}$

evaluate at $z = e^{j\omega}$

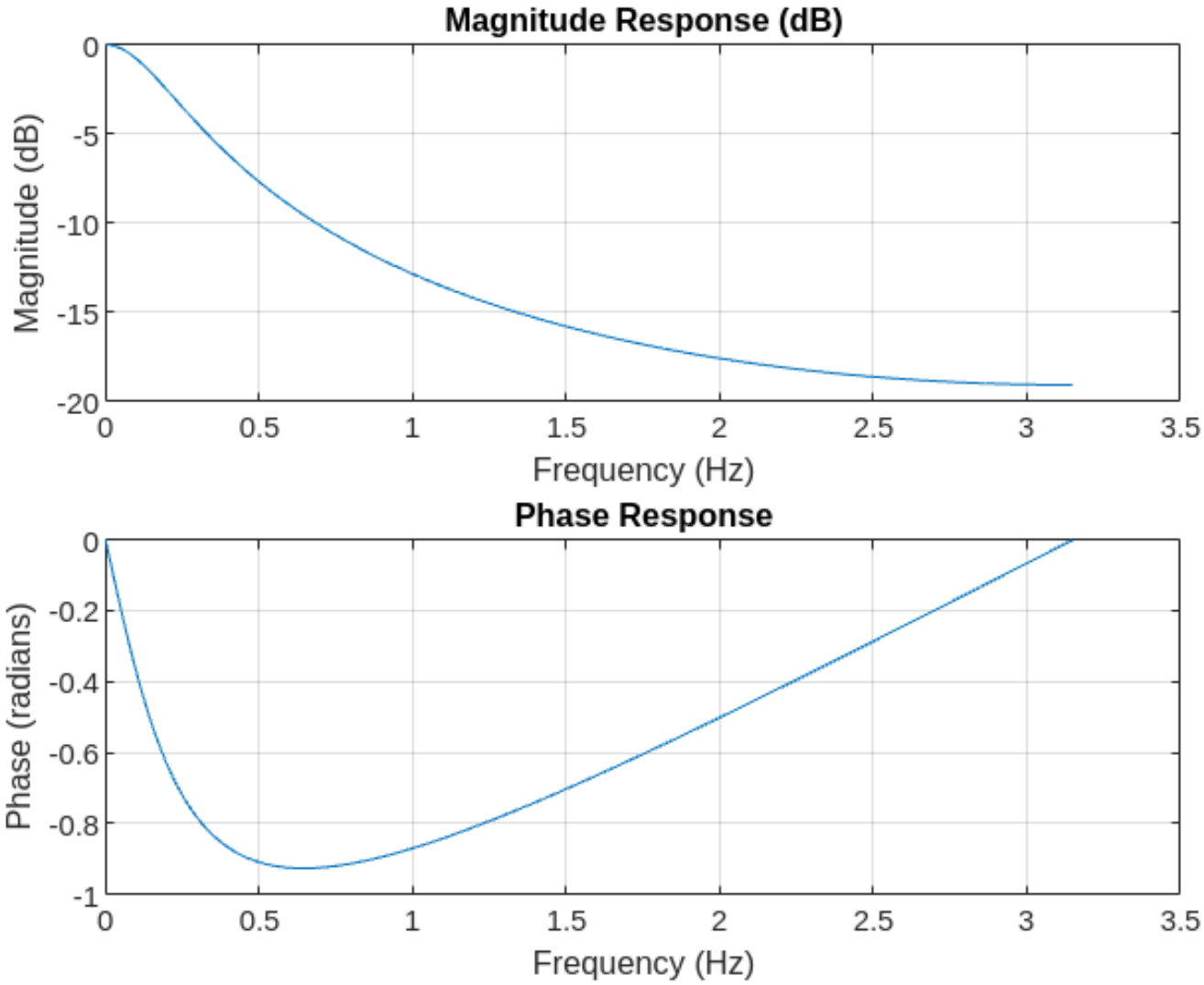
plot the magnitude and phase responses,

```
% Define the transfer function coefficients
b = 0.2;
a = [1, -0.8];  % Coefficients in the denominator (corrected)
k = 6.3
```

```
k = 6.3000
```

```
% Frequency vector
fs = k;  % Sampling frequency (you can adjust this)
num_points = 1024;  % Number of frequency points
frequencies = linspace(0, fs/2, num_points);  % Frequencies in the range [0, fs/2]
```

```
% Calculate the frequency responses at discrete frequencies on the unit circle
z = exp(1i * 2 * pi * frequencies / fs);  % Calculate z values
H = b ./ (1 +  a(2) * z.^(-1));  % Calculate H(z) for all frequencies (note the negative sign)

% Compute magnitude and phase responses
magnitude_response = abs(H);
phase_response = angle(H);

% Plot magnitude response in dB
subplot(2, 1, 1);
plot(frequencies,20*log10(magnitude_response));
title('Magnitude Response (dB)');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

% Plot phase response
subplot(2, 1, 2);
plot(frequencies, phase_response);
title('Phase Response');
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
grid on;
```

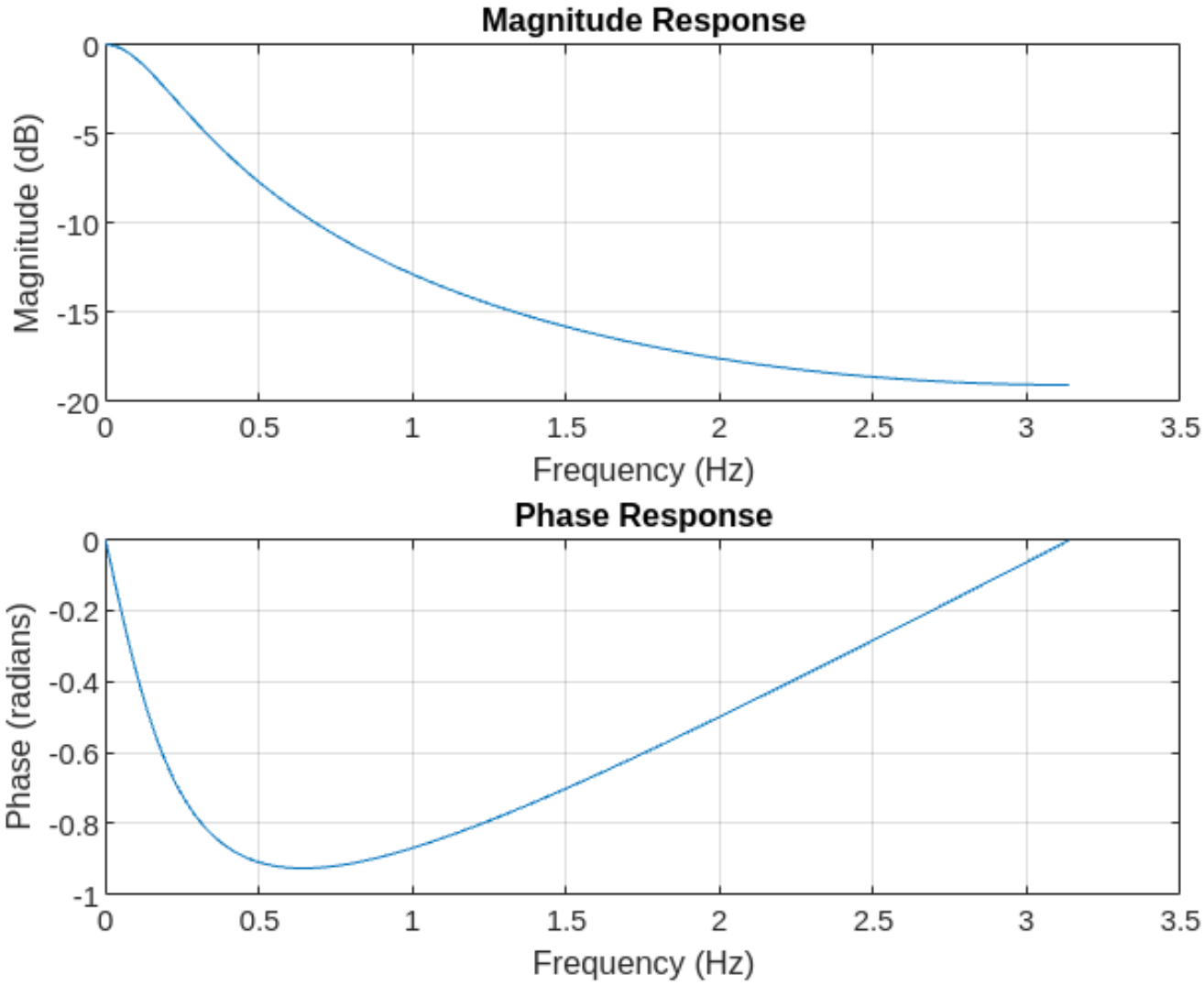


and state what type of a filter it is

It is a low pass filter. the magnitude drops opp as we increase the frequency.

```
% testing my version against freqz
% Define the coefficients of the difference equation
```

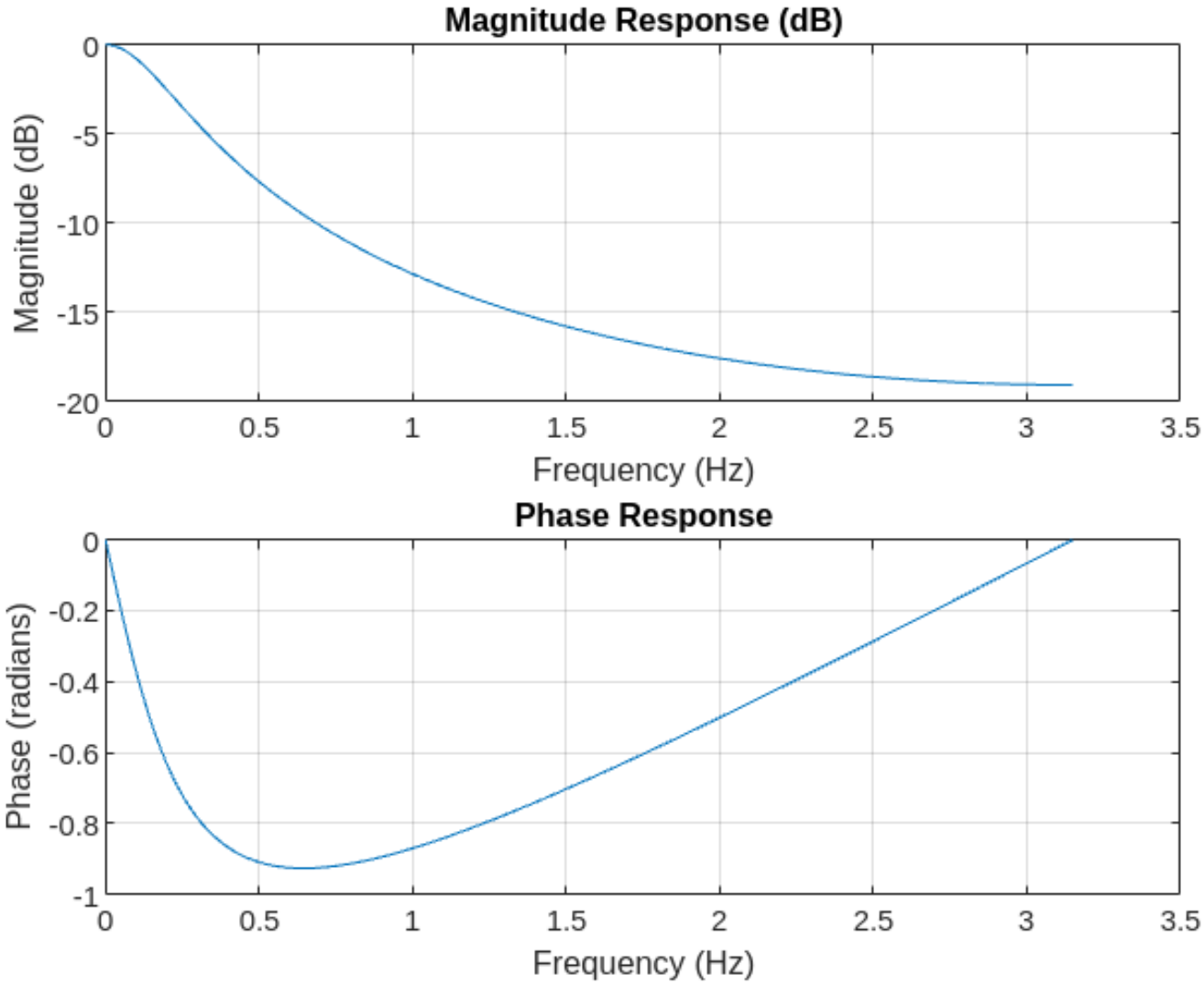
```
a = [1, -0.8];  % Coefficients of y(n)
b = 0.2;        % Coefficient of x(n)

% Compute the transfer function
[H w] = freqz(b, a, 1024);  % Frequency response

% Calculate magnitude and phase responses
magnitude = abs(H);
phase = angle(H);

% Frequency vector
fs = 50;  % Sampling frequency (you can adjust this)
frequencies = (0:1023) * fs / 1024;

% Plot magnitude response
subplot(2,1,1);
plot(w, 20*log10(magnitude));
title('Magnitude Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
% Plot phase response
subplot(2,1,2);
plot(w, unwrap(phase));  % Unwrap phase for better visualization
title('Phase Response');
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
grid on
```




```
% Define the transfer function coefficients
b = 0.2;
a = [1, -0.8];  % Coefficients in the denominator (corrected)
k = 6.3% for messing around with the value
```

```
k = 6.3000
```

```
% Frequency vector
fs = 1;
num_points = 1024;  % Number of frequency points
frequencies = linspace(0, k/2, num_points);

z = exp(1i * 2 * pi * frequencies / k);
H = b ./ (1 +  a(2) * z.^(-1));

magnitude_response = abs(H);
phase_response = angle(H);
subplot(2, 1, 1);
plot(frequencies,20*log10(magnitude_response));
title('Magnitude Response (dB)');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

subplot(2, 1, 2);
plot(frequencies, phase_response);
title('Phase Response');
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
grid on;
```



4

using the same kind of solving as before,

$$y(n) = 0.01x(n) + 0.05x(n-1) + 0.05x(n-2) + 0.01x(n-3) + 1.75y(n-1) - 1.18y(n-2) + 0.28y(n-3)$$
$$-1.75y(n-1) + 1.18y(n-2) - 0.28y(n-3)$$

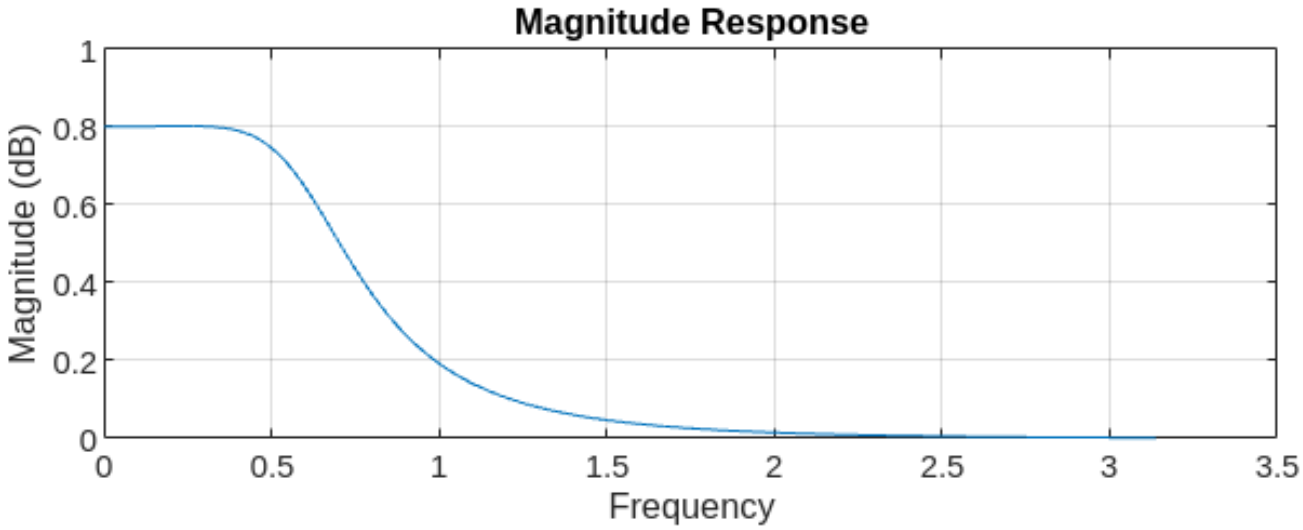
$$y(n) - 1.75y(n-1) + 1.18y(n-2) - 0.28y(n-3) = 0.01x(n) + 0.05x(n-1) + 0.05x(n-2) + 0.01x(n-3)$$

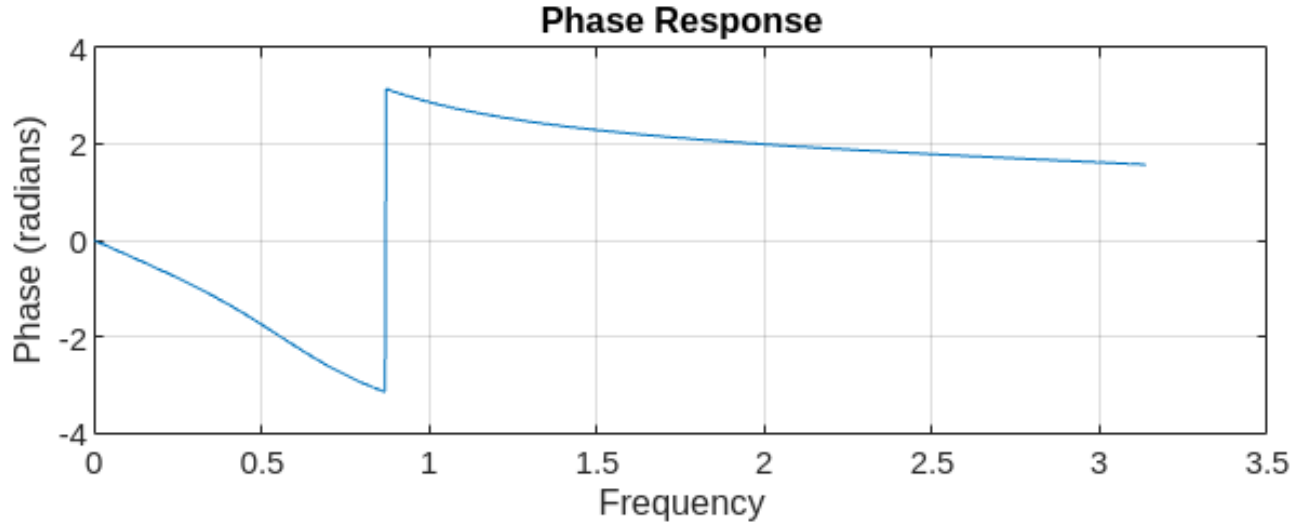
$$y(n) - 1.75y(n-1) + 1.18y(n-2) - 0.28y(n-3) = 0.01x(n) + 0.05x(n-1) + 0.05x(n-2) + 0.01x(n-3)$$

$$Y = [1, -1.75, 1.18, -0.28]$$

$$X = [.01, .05, .05, .01]$$

```
% Define the coefficients of the difference equation
b = [0.01, 0.05, 0.05, 0.01];  % Coefficients of x(n)
a = [1, -1.75, 1.18, -0.28];  % Coefficients of y(n)

% Frequency vector (normalized)
num_points = 1024;  % Number of frequency points
frequencies_normalized = linspace(0, pi, num_points);  % Frequencies in the range [-pi, pi]
z = exp(-1i * 2 * pi * frequencies_normalized);
[H1,w] = freqz(b, a, 1024);  % Calculate H(z)

magnitude_response = abs(H1);
phase_response = angle(H1);


subplot(2, 1, 1);
plot(w, (magnitude_response));
title('Magnitude Response');
xlabel('Frequency');
ylabel('Magnitude (dB)');
grid on;

subplot(2, 1, 2);
plot(w, phase_response);
title('Phase Response');
xlabel('Frequency');
ylabel('Phase (radians)');
grid on;
```



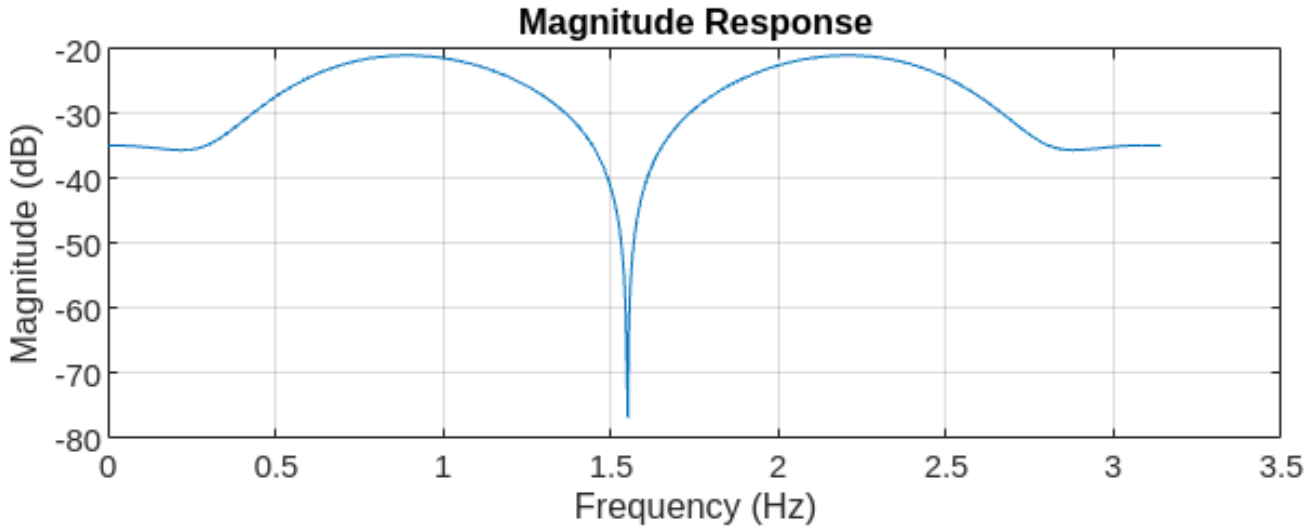


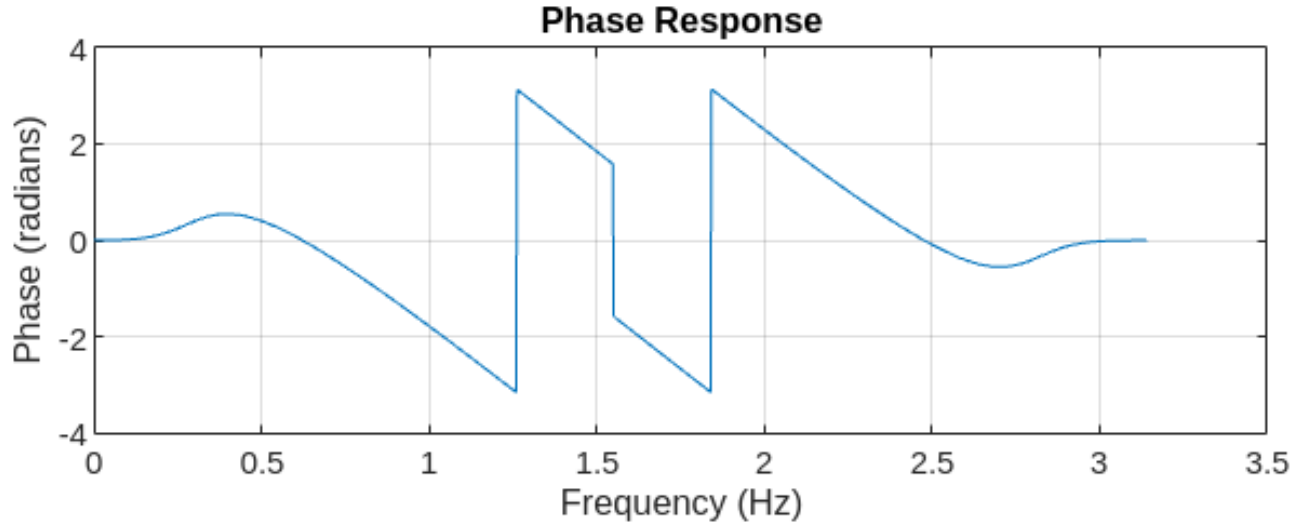
4 Interesting

my version does not agree and I am not sure why. I will look into it and see where I went wrong

```
for k = 1:.1:pi
% Frequency vector
```

```
fs = 2*pi;
frequencies = linspace(0, fs/2, num_points);

z = exp(1i * 2 * pi * frequencies / k);
ones_for_a = ones(size(frequencies))*a(1);
H_num = (a(1)*ones(size(frequencies)) + a(2)*z.^(-1) + a(3)*z.^(-2) + a(4)*z.^(-3));
H_den = (b(1)*ones(size(frequencies)) + b(2)*z.^(-1) + b(3)*z.^(-2) + b(4)*z.^(-3)).^(-1);
H = H_num./H_den;
magnitude_response = abs(H);

phase_response = angle(H);
subplot(2, 1, 1);
plot(frequencies,20*log10(magnitude_response));
title('Magnitude Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

subplot(2, 1, 2);
plot(frequencies, phase_response);
title('Phase Response');
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
grid on;
pause(.1)
end
```






```
function coeffs = dfs(x, N)
    % computes the Discrete Fourier Series (DFS) coefficients of a periodic 1-D signal
    % x is one period of the signal
    % N is the fundamental period.
    fund_period = N;
```

```
    signal = x;


    n = 0:fund_period-1;
    k = n;

    % Compute the DFS coefficients using vectorized operations
    coeffs = signal * exp(-1i * 2 * pi * (n.') * k / fund_period);

end


function icoeffs = idfs(X, N)
    % computes the Inverse Discrete Fourier Series coefficients
    % x is one period of the signal
    % N is the fundamental period.
    fund_period = N;
    signal = X;

    n = 0:fund_period-1;
    k = n;
    icoeffs = (signal * exp(1i * 2 * pi * (n.') * k / fund_period)) / fund_period;

end
```