1. IIR Digital Lattice Filter

   (a) Using `tf2sos` subroutine, obtaining coefficients to second-order-system cascaded filter for $H(z)$

```
>> sos =

    1.0000   -1.8810    1.0000    1.0000    0.7493    0.3790
    1.0000   -1.3129    1.0000    1.0000   -0.2219    0.7562
    1.0000   -1.0724    1.0000    1.0000   -0.6471    0.9241
    1.0000   -0.9177    1.0000    1.0000   -0.7763    0.9824

>> gain =

    0.0609
```

   where the transfer function is in the form

$$\begin{aligned} &0.0609\frac{1-1.881z^{-1}+z^{-2}}{1+0.7493z^{-1}+0.3790z^{-2}}\frac{1-1.3129z^{-1}+z^{-2}}{1-0.2219z^{-1}+0.7562z^{-2}} \\ &\qquad\frac{1-1.0724z^{-1}+z^{-2}}{1-0.6471z^{-1}+0.9241z^{-2}}\frac{1-0.9177z^{-1}+z^{-2}}{1-0.7763z^{-1}+0.9824z^{-2}} \end{aligned} \tag{1}$$

   (b) Calculating the first SOS section by hand:

$$K_1 = \frac{a(1)}{1+a(2)} = \frac{0.7493}{1+0.379} = 0.5434 \tag{2}$$

$$K_2 = 0.379 \tag{3}$$

$$V_3 = 1 \tag{4}$$

$$V_2 = b(1) - (K_1K_2 + K_1)V_3 = -2.6303 \tag{5}$$

$$V_1 = b(0) - K_1V_2 - K_2V_3 = 2.0503 \tag{6}$$

   (c) Calculate all the SOS lattice coefficients using MATLAB

```matlab
%% 1

% transfer function
b = [0.0609 -0.3157 0.8410 -1.4378 1.7082 -1.4378 0.8410 -0.3157 ...
    0.0609];
a = [1.0 -0.8961 2.6272 -0.9796 2.1282 -0.0781 0.9172 0.0502 0.2602];
% second order system + gain
[sos, gain] = tf2sos(b, a);

% allocate space for individual sos'
[rows, ¬] = size(sos);
b_new = zeros(rows, 3);
a_new = zeros(rows, 3);

% allocate space for new lattice and ladder coefficients
k_new = zeros(2, rows);
v_new = zeros(3, rows);
```

```
% loop through all and calculate all coefficients
for i = 1:rows
    b_new(i, :) = sos(i, 1:3);
    a_new(i, :) = sos(i, 4:end);

    [k_new(:, i), v_new(:, i)] = tf2latc(b_new(i, :), a_new(i, :));
end

% display
disp(k_new');
disp(v_new');
```

```
>> k_new' =
    0.5433    0.3790
   -0.1264    0.7562
   -0.3363    0.9241
   -0.3916    0.9824

>> v_new' =
    2.0501   -2.6302    1.0000
    0.1059   -1.0909    1.0000
   -0.0671   -0.4253    1.0000
   -0.0378   -0.1413    1.0000
```

where each section is represented by each row of the displayed output, and each coefficient index is represented by the column. We can see that our hand calculation in part (b) was the same as the MATLAB result:

```
>> k_new(:, 1)' =
    0.5433    0.3790

>> v_new(:, 1)' =
    2.0501   -2.6302    1.0000
```

Hand drawn block diagram

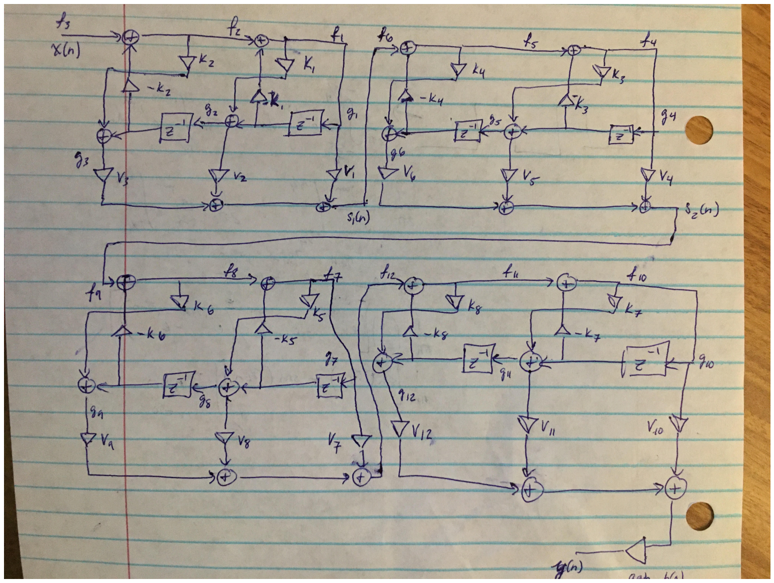

Figure 1: Handdrawn Block Diagram of the IIR Lattice implementation

At the bottom, it got cut off, but the gain term is taken into account right before $y(n)$

(d) Difference equations from block diagram as well as the coefficients from (c) are given by

$$f_3(n) = x(n) \tag{7}$$

$$g_3(n) = 0.3790 f_2(n) + g_2(n-1) \tag{8}$$

$$f_2(n) = -0.3790 g_2(n-1) + f_3(n) \tag{9}$$

$$g_2(n) = 0.5433 f_1(n) + g_1(n-1) \tag{10}$$

$$f_1(n) = -0.5433 g_1(n-1) + f_2(n) \tag{11}$$

$$g_1(n) = f_1(n) \tag{12}$$

$$s_1(n) = 2.0501 g_1(n) - 2.6302 g_2(n) + g_3(n) \tag{13}$$

$$f_6(n) = s_1(n) \tag{14}$$

$$g_6(n) = 0.7562 f_5(n) + g_5(n-1) \tag{15}$$

$$f_5(n) = -0.7562 g_5(n-1) + f_6(n) \tag{16}$$

$$g_5(n) = -0.1264 f_4(n) + g_4(n-1) \tag{17}$$

$$f_4(n) = 0.1264 g_4(n-1) + f_5(n) \tag{18}$$

$$g_4(n) = f_4(n) \tag{19}$$

$$s_2(n) = 0.1059g_4(n) - 1.0909g_5(n) + g_6(n) \tag{20}$$

$$f_9(n) = s_2(n) \tag{21}$$
$$g_9(n) = 0.9241f_8(n) + g_8(n-1) \tag{22}$$
$$f_8(n) = -0.9241g_8(n-1) + f_9(n) \tag{23}$$
$$g_8(n) = -0.3363f_7(n) + g_7(n-1) \tag{24}$$
$$f_7(n) = 0.3363g_7(n-1) + f_8(n) \tag{25}$$
$$g_7(n) = f_7(n) \tag{26}$$
$$s_3(n) = -0.0671g_7(n) - 0.4253g_8(n) + g_9(n) \tag{27}$$

$$f_{12}(n) = s_3(n) \tag{28}$$
$$g_{12}(n) = 0.9824f_{11}(n) + g_{11}(n-1) \tag{29}$$
$$f_{11}(n) = -0.9824g_{11}(n-1) + f_{12}(n) \tag{30}$$
$$g_{11}(n) = -0.3916f_{10}(n) + g_{10}(n-1) \tag{31}$$
$$f_{10}(n) = 0.3916g_{10}(n-1) + f_{11}(n) \tag{32}$$
$$g_{10}(n) = f_{10}(n) \tag{33}$$
$$s_4(n) = -0.0378g_{10}(n) - 0.1413g_{11}(n) + g_{12}(n) \tag{34}$$

$$y(n) = 0.0609s_4(n) \tag{35}$$

2. Lattice coefficients for various SOS are given

   (a) MATLAB function to implement the second order IIR lattice filter

```matlab
function y = my_sos(k, v, x)
%my_sos is a function for an individual SOS to implement a digital ...
    IIR filter
%   INPUTS:     k - k coefficients
%               v - v coefficients
%               x - input data

% allocating memory for coefficients
a = ones(1, 3);
b = ones(1, 3);

% calculating denominator coefficients
a(3) = k(2);
a(2) = k(1) * (1 + a(3));

% calculating numerator coefficients
b(3) = v(3);
```

```
b(2) = v(2) + (k(1)*k(2) + k(1))*b(3);
b(1) = v(1) + v(2)*k(1) + v(3)*k(2);

% appropriately filtering
y = filter(b, a, x);
end
```

(b) Downloaded `sampdata` from Moodle

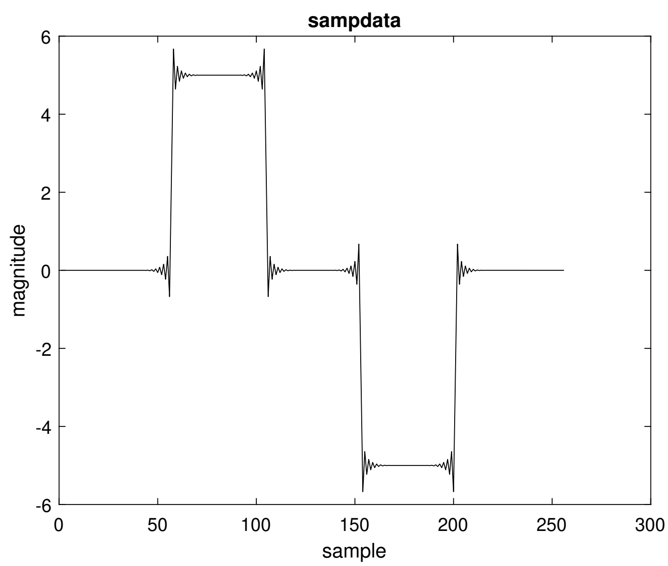

Figure 2: `sampdata`

(c) Filtering using `my_sos` function from part (a)

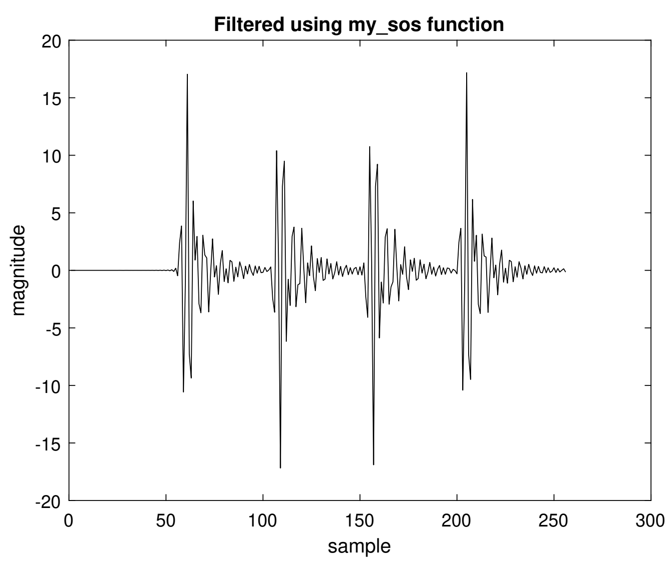

Figure 3: Using my function to filter `sampdata`

```
% get data
data = sampdata;

% coefficients
k1 = [0.4545, 0.5703];
v1 = [1.2459, 0.7137, -1];

k2 = [-0.1963, 0.8052];
v2 = [-0.0475, -1.2347, 1];

k3 = [0.8273, 0.8835];
v3 = [-0.1965, 0.3783, 1];

% cascade through my function
y_cascade = my_sos(k1, v1, data);
y_cascade = my_sos(k2, v2, y_cascade);
y_cascade = my_sos(k3, v3, y_cascade);

% plot and save
figure(1);
plot(y_cascade);
xlabel("sample");
ylabel("magnitude");
title("Filtered using my\_sos function");
print -deps 2c_mysos
```

(d) Filtering using `latc2tf` function from MATLAB

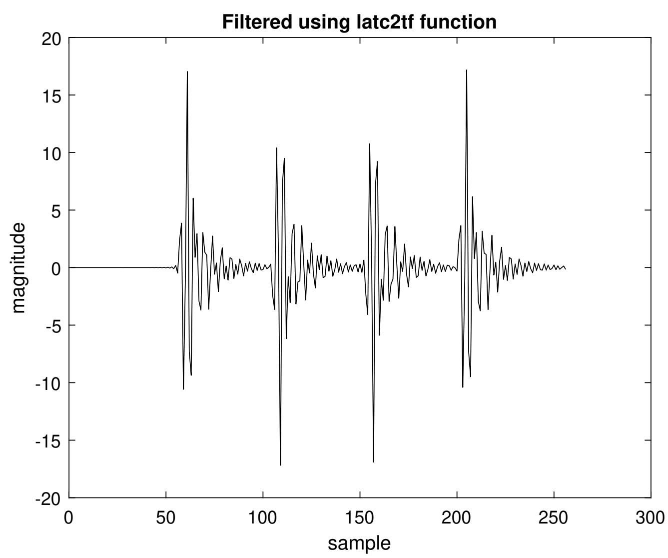

Figure 4: Using `latc2tf` to filter `sampdata`

```
% get data
data = sampdata;

% coefficients
k1 = [0.4545, 0.5703];
```

```
v1 = [1.2459, 0.7137, -1];

k2 = [-0.1963, 0.8052];
v2 = [-0.0475, -1.2347, 1];

k3 = [0.8273, 0.8835];
v3 = [-0.1965, 0.3783, 1];

% find numerator and denominators
[b1, a1] = latc2tf(k1, v1);
[b2, a2] = latc2tf(k2, v2);
[b3, a3] = latc2tf(k3, v3);

% cascade through the filter each time
y_cascade_latc = filter(b1, a1, data);
y_cascade_latc = filter(b2, a2, y_cascade_latc);
y_cascade_latc = filter(b3, a3, y_cascade_latc);

% plot and save
figure(2);
plot(y_cascade_latc);
xlabel("sample");
ylabel("magnitude");
title("Filtered using latc2tf function");
print -deps 2d_latc
```

(e) The two plots from (c) and (d) are identical (with a minute absolute error). This means that the MATLAB function `latc2tf` is a simple routine that can be implemented in a couple lines, like shown with my routine of `my_sos`. I have calculated the absolute difference, which has a maximum error on the order of 10^{-14}

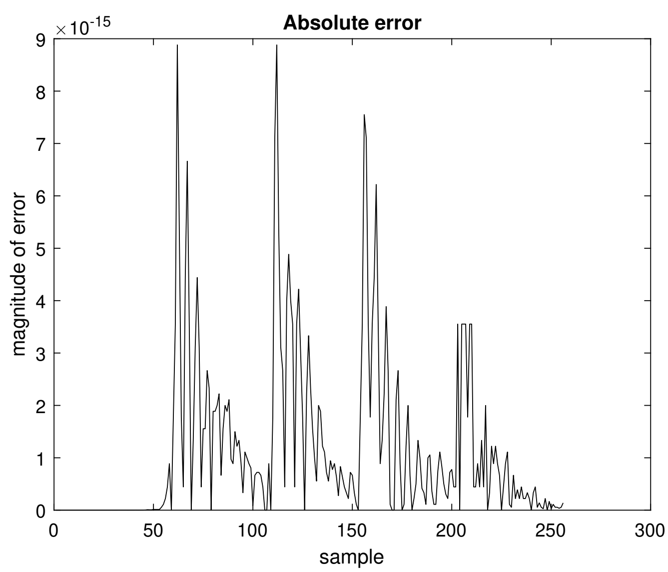

Figure 5: Absolute error between `my_sos` and `latc2tf`

(f) The plots are seen above, and can also be reproduced using the MATLAB script below

```
%% PROBLEM 2

data = sampdata;

figure(1)
plot(sampdata);
xlabel("sample");
ylabel("magnitude");
title("sampdata");
print -deps sampdata

k1 = [0.4545, 0.5703];
v1 = [1.2459, 0.7137, -1];

k2 = [-0.1963, 0.8052];
v2 = [-0.0475, -1.2347, 1];

k3 = [0.8273, 0.8835];
v3 = [-0.1965, 0.3783, 1];

y_cascade = my_sos(k1, v1, data);
y_cascade = my_sos(k2, v2, y_cascade);
y_cascade = my_sos(k3, v3, y_cascade);

figure(2);
plot(y_cascade);
xlabel("sample");
ylabel("magnitude");
title("Filtered using my\_sos function");
print -deps 2c_mysos

[b1, a1] = latc2tf(k1, v1);
[b2, a2] = latc2tf(k2, v2);
[b3, a3] = latc2tf(k3, v3);

y_cascade_latc = filter(b1, a1, data);
y_cascade_latc = filter(b2, a2, y_cascade_latc);
y_cascade_latc = filter(b3, a3, y_cascade_latc);

figure(3);
plot(y_cascade_latc);
xlabel("sample");
ylabel("magnitude");
title("Filtered using latc2tf function");
print -deps 2d_latc

figure(4);
plot(abs(y_cascade - y_cascade_latc));
xlabel("sample");
ylabel("magnitude of error");
title("Absolute error");
print -deps 2e_diff

function y = my_sos(k, v, x)
%my_sos is a function for an individual SOS to implement a digital ...
    IIR filter
%   INPUTS:     k - k coefficients
%               v - v coefficients
%               x - input data

% allocating memory for coefficients
```

```
a = ones(1, 3);
b = ones(1, 3);

% calculating denominator coefficients
a(3) = k(2);
a(2) = k(1) * (1 + a(3));

% calculating numerator coefficients
b(3) = v(3);
b(2) = v(2) + (k(1)*k(2) + k(1))*b(3);
b(1) = v(1) + v(2)*k(1) + v(3)*k(2);

% appropriately filtering
y = filter(b, a, x);
end
```

3. The ideal frequency response is given by

$$H_d(\omega) = 1, \quad \frac{\pi}{4} < |\omega| \leq \frac{3\pi}{4} \tag{36}$$

and 0 elsewhere between $-\pi \leq \omega \leq \pi$

The Blackman window is given by

$$w(n) = 0.42 + 0.5\cos\left(\frac{2\pi n}{2M}\right) + 0.08\cos\left(\frac{4\pi n}{2M}\right), \quad -M \leq n \leq M \tag{37}$$

(a) Determine the coefficients for the linear phase filter with 61 filter coefficients

i. The required samples, M, of the impulse response is given by

$$M = \frac{61-1}{2} = 30 \tag{38}$$

In addition, the filter coefficients are determined by taking the DTFT of the window $H_d(\omega)$

$$\text{DTFT}\{H_d(\omega)\} = \frac{1}{2\pi}\int_{\omega=-\pi}^{\pi} H_d(\omega)e^{j\omega n}d\omega \tag{39}$$

$$\frac{1}{2\pi}\int_{\omega=-\frac{3\pi}{4}}^{-\frac{\pi}{4}} e^{j\omega n}d\omega + \frac{1}{2\pi}\int_{\omega=\frac{\pi}{4}}^{\frac{3\pi}{4}} e^{j\omega n}d\omega \tag{40}$$

Which boils down to the following after applying Euler's identities

$$\frac{1}{\pi n}\left[\sin\left(\frac{3\pi}{4}n\right) - \sin\left(\frac{\pi}{4}n\right)\right], \quad n \neq 0 \tag{41}$$

$$\frac{1}{2}, \quad n = 0 \tag{42}$$

ii. Write MATLAB scripts for

A. Computing the required 61 samples for the desired impulse response

```
%% 3

% length
len = 61;
range = (len - 1) / 2;

% two angular frequencies
w1 = pi / 4;
w2 = 3 * pi / 4;

% get negative values of impulse response
n1 = -range:-1;
bneg = (1 ./ (pi * n1)).*(sin(w1 * n1) - sin(w2 * n1));

% get positive values of impulse response
n2 = 1:range;
bpos = (1 ./ (pi * n2)).*(sin(w1 * n2) - sin(w2 * n2));

% value of impulse response at n = 0
bmid = 0.5;
b = [bneg bmid bpos];
```

B. Computing the required 61 samples for the Blackman window

```
function wind = myblackman(n)
% wind = myblackman(n);
%
% This routine returns a Blackman window
% of length n

nmod = mod(n, 2);

c = 2 * pi/(n - 1);
if (nmod == 1)
    m = fix(0.5 * (n - 1));
    k = -m:1:m;
    wind = 0.42 + 0.5*cos(k * c) + 0.08*cos(2 * k * c);
elseif (nmod == 0)
    m = fix(0.5 * n);
    k = -m:1:(m - 1);
    k = k + 0.5;
    wind = 0.42 + 0.5*cos(k * c) + 0.08*cos(2 * k * c);
end

return
```

and then in the main script, the following is called

```
% get 61 samples of blackman window
myblack = myblackman(len);
```

C. Multiplying the two sequences together

```
% product of the impulse response and blackman window
% to get filter coefficients
b_prod = b .* myblack;

```

```
% plotting and saving
figure(1); plot(b_prod);
title("Filter designed using my MATLAB scripts");
print -deps 3aiic
```

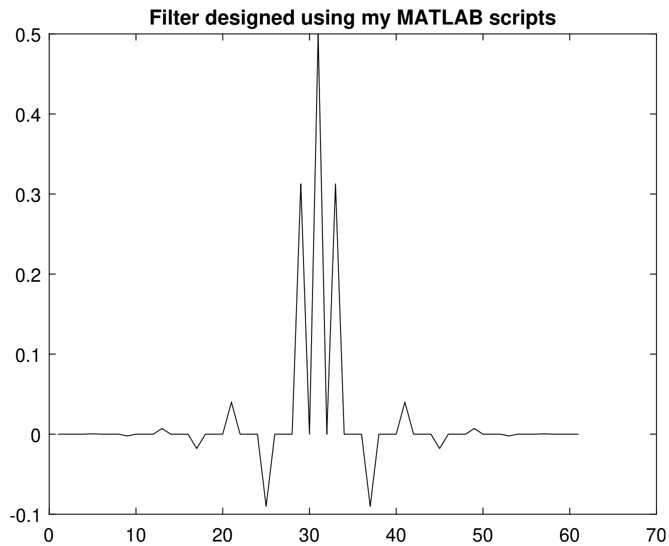

Figure 6: Plot of the filter coefficients

D. Magnitude and phase response of the designed filter

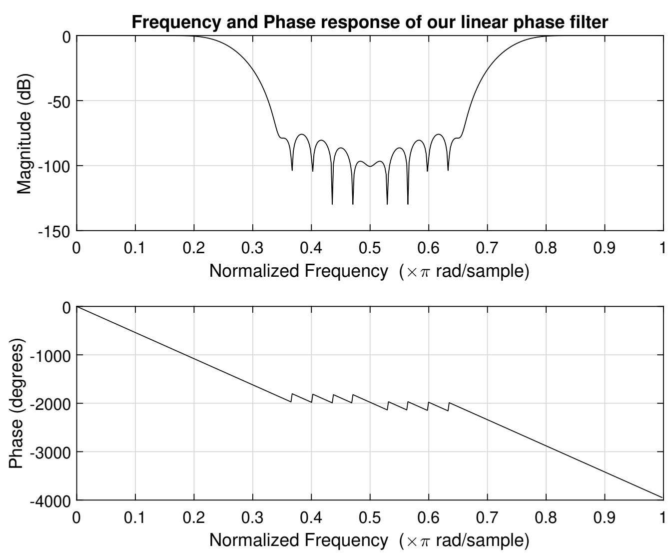

Figure 7: Magnitude and Phase response of filter designed above

iii. Design the same digital filter as above, but using built-in MATLAB routine `fir1` within the signal processing toolbox. Use Blackman window function as well

```
% order of 60
order = len - 1;
% pi/4 and 3pi/4
wn = [0.25 0.75];
% get the blackman window for 61 values
w = window(@blackman, order + 1);
% get the filter coefficients using fir1
b_black = fir1(order, wn, 'stop', w);
```

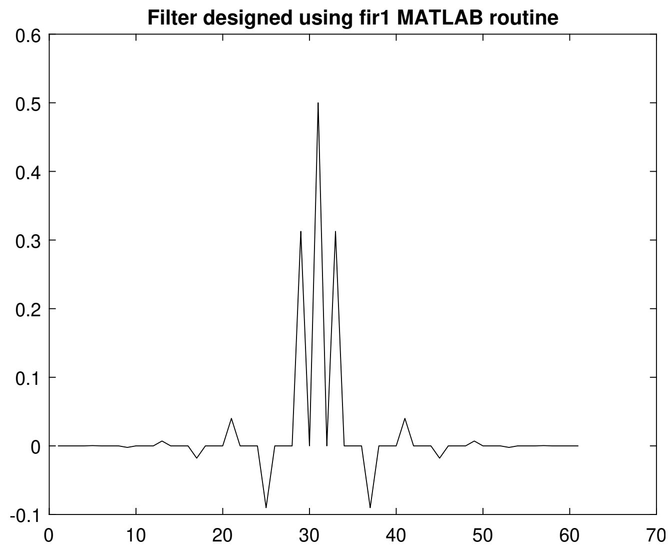

Figure 8: Filter coefficients from MATLAB `fir1` routine plotted

iv. Magnitude and phase response of the newly designed filter

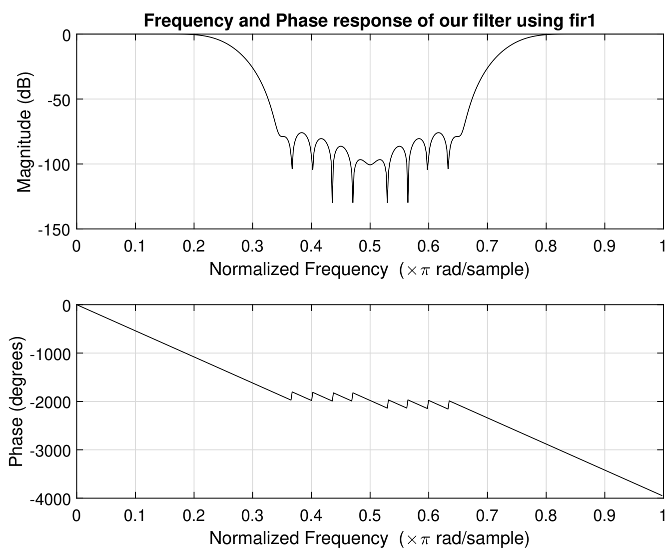

Figure 9: Magnitude and Phase response of filter designed above

v. Compare the coefficients from the two calculations

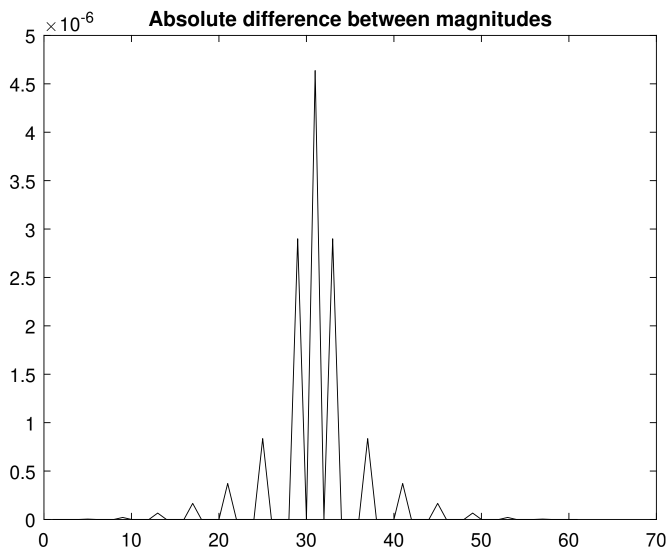

Figure 10: Absolute difference between coefficients

As one can see, the coefficients of the two methods are near identical, with a maximum error on the magnitude of 10^{-6}

4. Designing a linear phase FIR filter with length $N = 61$, which satisfies the conditions given in the HW7 document

(a) The equation to determine the causal frequency response $H(k)$ is given by

$$H(k) = \begin{cases} e^{\frac{-j2\pi k(N-1)}{2N}}, & 0 \le k \le 12 \\ 0.9e^{\frac{-j2\pi k(N-1)}{2N}}, & k = 13 \\ 0, & 14 \le k \le 47 \\ 0.9e^{\frac{j2\pi k(N-1)}{2N}}, & k = 48 \\ e^{\frac{j2\pi k(N-1)}{2N}}, & 49 \le k \le 60 \end{cases} \tag{43}$$

Due to the circular property in the frequency domain

(b) MATLAB script to determine the 61 filter coefficients for this filter

```
%% 4
% number of coefficients
N = 61;
M = 0.5 * (N - 1);
% indicies 0->12 are constant magnitude of 1
k1 = 0:12;

th = -1i * 2 * pi * k1 * M / N;
h = exp(th);

% at k = 13, 0.9 in magnitude
h(14) = 0.9 * exp(-1i * 2 * pi * 13 * M / N);
```

```

% at 14 ≤ k ≤ 30, 0 in magnitude
h(15:31) = 0;

% flip and complex conjugate the remaining ones
h(32:61) = fliplr(conj(h(2:31)));

% determine the coefficients
b = real(ifft(h));
a = 1;
```

(c) Plotting magnitude and phase response of the filter

```
% plotting mag and phase response with freqz
freqz(b, a);
title("Magnitude and Phase response of the FIR filter");
print -deps 4c
```

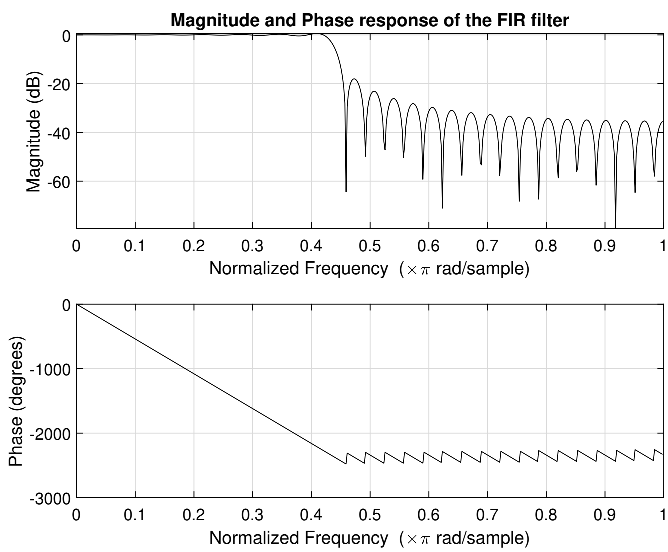

Figure 11: Magnitude and phase response of the filter we designed

(d) At $k = 13$, the frequency $\omega = \frac{2\pi 13}{61} \approx 1.339$. If instead of plotting the magnitude in terms of dB, but we plot the magnitude with respect to the phase, at $\omega \approx 1.339$ we find that

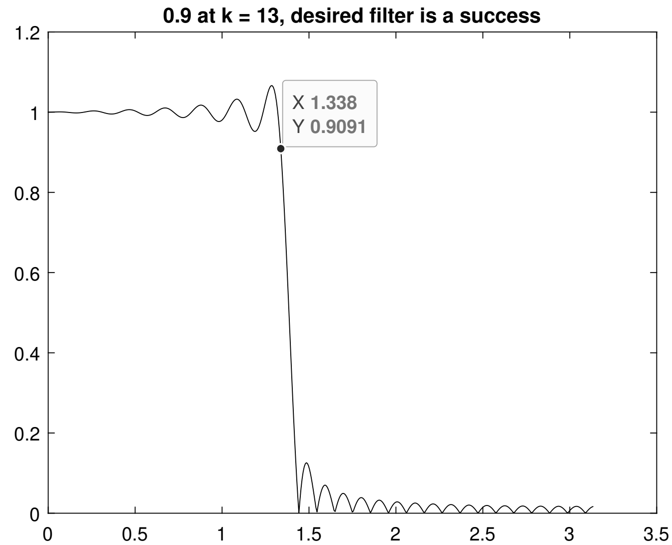

Figure 12: Magnitude of filter plotted against angular frequency

And you can observe that for lower values of k, the magnitude equals roughly 1, and for higher values of k, the magnitude equals roughly zero. The filter is a success!