Final Exam

Name: Zhankun Luo

Problem 1

Given the sequence $x(n)$ having 4 data points as $x(0) = -1, x(1) = 2, x(2) = 1, x(3) = 1$, and the sampling period of $T = 0.5$ seconds, no window function is used.

a. Sketch the 4-points fast Fourier transform algorithm (Decimation in frequency FFT) to compute the DFT coefficients: X(0), X(1), X(2), X(3)

b. Compute the amplitude spectrum A_0, A_1, A_2, A_3

c. Determine the corresponding frequency f for the amplitude spectrum A_1 .

d. Determine the frequency resolution.

e. Use the DFT formula to determine $X(1)$

(10 points)

solution

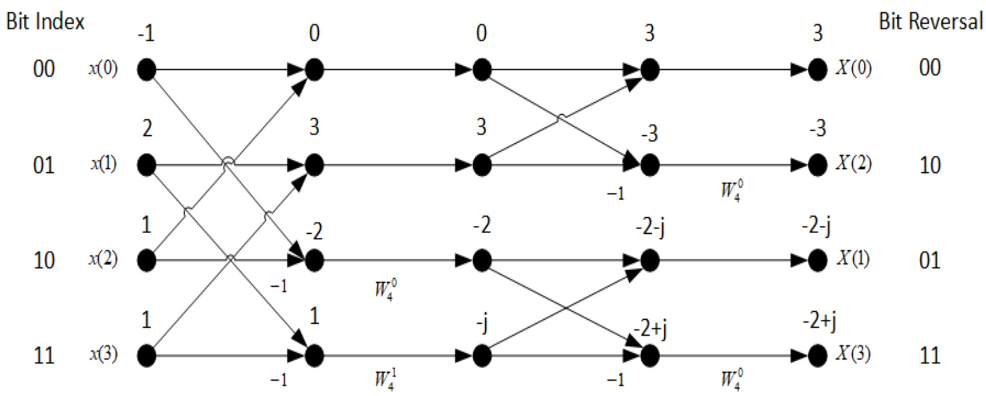
$$X(k) = \sum_{n=0}^{(N/2)-1} \left(x(n) + (-1)^k x\left(n + \frac{N}{2}\right)\right) W_N^{kn}$$

Thus

$$X(2m) = \sum_{n=0}^{(N/2)-1} \left(x(n) + x\left(n + \frac{N}{2}\right)\right) W_N^{2mn} = \text{DFT}[\left(x(n) + x\left(n + \frac{N}{2}\right)\right)]$$

$$X(2m+1) = \sum_{n=0}^{(N/2)-1} \left(x(n) - x\left(n + \frac{N}{2}\right)\right) W_N^n W_N^{2mn} = \text{DFT}[\left(x(n) - x\left(n + \frac{N}{2}\right)\right) W_N^n]$$

a. Sketch the 4-points fast Fourier transform algorithm (Decimation in frequency FFT) to compute the DFT coefficients: X(0), X(1), X(2), X(3)



We compute X(k)

$$X(0), X(1), X(2), X(3) = [3, -2-j, -3, -2+j]$$

b. Compute the amplitude spectrum A_0, A_1, A_2, A_3

The amplitude spectrum to a two-sided amplitude spectrum A_k is

$$A_k = \frac{1}{N}|X(k)|, k = 0, 1, 2, \cdots, N-1$$

We compute A_k

$$A_0, A_1, A_2, A_3 = [\frac{3}{4}, \frac{\sqrt{5}}{4}, \frac{3}{4}, \frac{\sqrt{5}}{4}]$$

c. Determine the corresponding frequency f for the amplitude spectrum A_1 .

For k, the corresponding frequency is

$$f = k\frac{f_s}{N} = \frac{k}{NT}$$

When k = 1, for the amplitude spectrum A_1 .

$$f = \frac{1}{NT} = \frac{1}{4 \times 0.5} = 0.5Hz$$

d. Determine the frequency resolution.

The frequency resolution is

$$\Delta f = \frac{f_s}{N} = \frac{1}{NT} = 0.5Hz$$

e. Use the DFT formula to determine $X(1)$

$$X(k) = \sum_{k=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn}$$

When k=1

$$\begin{aligned} X(1) &= (-1) \times e^{-j\frac{2\pi}{4}1\cdot 0} + 2 \times e^{-j\frac{2\pi}{4}1\cdot 1} + 1 \times e^{-j\frac{2\pi}{4}1\cdot 2} + 1 \times e^{-j\frac{2\pi}{4}1\cdot 3} \\ &= (-1) + 2 \times (-j) + 1 \times (-1) + 1 \times j = -2 - j \end{aligned}$$

Problem 2

Given the following DSP system with a sampling rate of 8000 Hz

$$y(n) = 0.2x(n) - 0.8y(n-2)$$

a. Obtain transfer function $H(z)$

b. Make a pole-zero plot and determine the stability.

c. Obtain the frequency response $H(e^{j\Omega})$ and then the magnitude response $|H(e^{j\Omega})|$

d. Compute the filter gain at the frequency of 0 Hz, 1000 Hz, 2000 Hz, 3000Hz, 4000 Hz, respectively. Make a plot of the magnitude frequency response.

e. Determine the filter type, that is, the lowpass filter, or high-pass filter, or bandpass filter, or band-stop filter. (10 points)

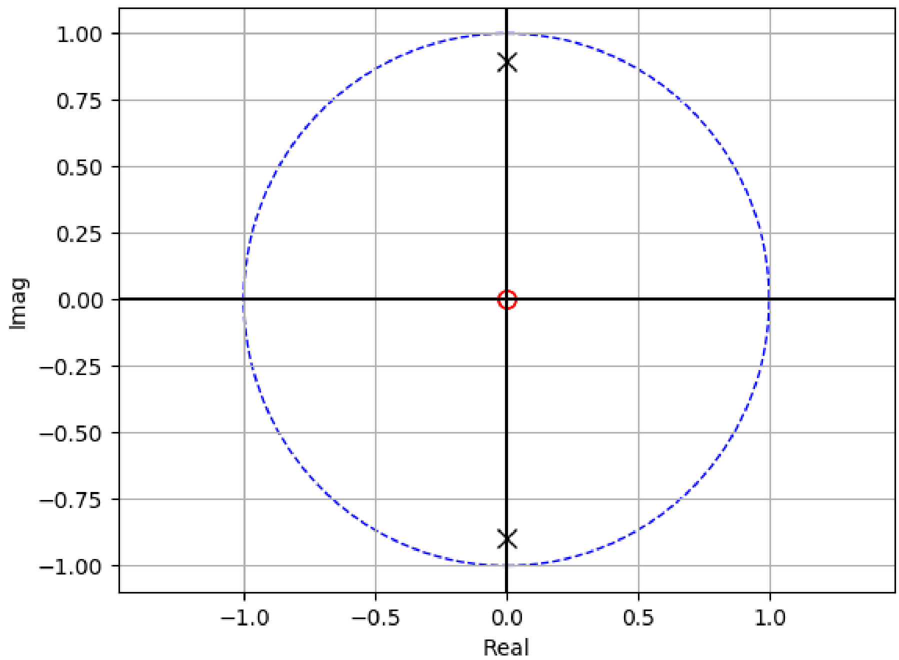
solution

a. Obtain transfer function $H(z)$

Do $\mathbb{Z}$ transform to the DSP equation

$$Y(z) = 0.2X(z) - 0.8z^{-2}Y(z)$$
$$H(z) \equiv \frac{Y(z)}{X(z)} = \frac{0.2}{1 + 0.8z^{-2}}$$

b. Make a pole-zero plot and determine the stability.



zeros: $0, 0$

poles: $+j\sqrt{0.8} = +j0.8944, -j\sqrt{0.8} = -j0.8944$

Because all poles: $|+j0.8944| < 1, |-j0.8944| < 1$

The DSP system is **stable**.

c. Obtain the frequency response $H(e^{j\Omega})$ and then the magnitude response $|H(e^{j\Omega})|$

$$H(e^{j\Omega}) = H(z)|_{z=e^{j\Omega}} = \frac{0.2}{1 + 0.8z^{-2}}|_{z=e^{j\Omega}} = \frac{0.2}{1 + 0.8\cos(2\Omega) - j0.8\sin(2\Omega)}$$

Then the magnitude response

$$\begin{aligned} |H(e^{j\Omega})| &= \frac{0.2}{\sqrt{(1 + 0.8\cos(2\Omega))^2 + 0.8^2\sin^2(2\Omega)}} \\ &= \frac{0.2}{\sqrt{1.64 + 1.6\cos(2\Omega)}} \end{aligned}$$

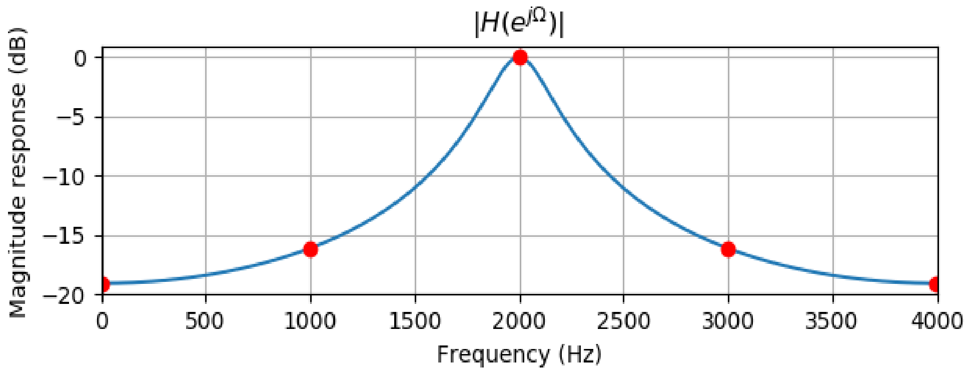
d. Compute the filter gain at the frequency of 0 Hz, 1000 Hz, 2000 Hz, 3000Hz, 4000 Hz, respectively. Make a plot of the magnitude frequency response.

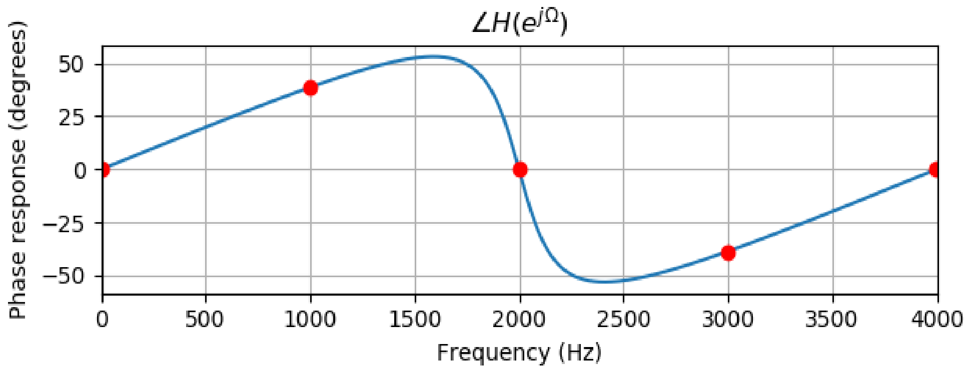
$$\Omega = 2\pi\frac{f}{f_s}$$

$$[0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi] \Leftarrow 0, 1000, 2000, 3000, 4000Hz$$

So, the filter gain at 0 Hz, 1000 Hz, 2000 Hz, 3000Hz, 4000 Hz are

$$\begin{aligned} |H(e^{j\Omega})| &= \frac{0.2}{\sqrt{1.64 + 1.6\cos(2\Omega)}} = [0.1111, 0.1562, 1, 0.1562, 0.1111] \\ &= [-19.08, -16.13, 0, -16.13, -19.08]\text{dB} \end{aligned}$$





e. Determine the filter type, that is, the lowpass filter, or high-pass filter, or bandpass filter, or band-stop filter.

filter type: **band-pass**

Problem 3

Design a 5-tap **bandpass FIR** filter whose lower and upper cutoff frequencies are 800 Hz, and 1000, respectively using the Hamming window method.

Assume the sampling frequency is 4000 Hz.

a. List the FIR filter coefficients

b. Determine the transfer function

c. Determine the DSP equation

d. Set up MATLAB routine "freqz()" to obtain the frequency response plot.

(10 points)

solution

a. List the FIR filter coefficients

Here $M = \frac{5-1}{2} = 2$, then $\Omega_L = 2\pi \frac{f_L}{f_s} = \frac{2\pi}{5}$, $f_H = 2\pi \frac{f_H}{f_s} = \frac{\pi}{2}$

$$h(n) = \begin{cases} \frac{\Omega_H - \Omega_L}{\pi} & \text{for } n = 0 \\ \frac{\sin(\Omega_H n)}{n\pi} - \frac{\sin(\Omega_L n)}{n\pi} & \text{for } n \neq 0 \quad -M \leq n \leq M \end{cases}$$

So, we compute

$$h(n) = [-0.09355, 0.01558, 0.1, 0.01558, -0.09355]$$

Using the Hamming window method.

$$w_{\text{ham}}(n) = 0.54 + 0.46 \cos\left(\frac{n\pi}{M}\right), -M \leq n \leq M$$

Then,

$$h_w(n) = h(n) \cdot w_{\text{ham}}(n) = [-0.007484, 0.008413, 0.1, 0.008413, -0.007484]$$

b. Determine the transfer function

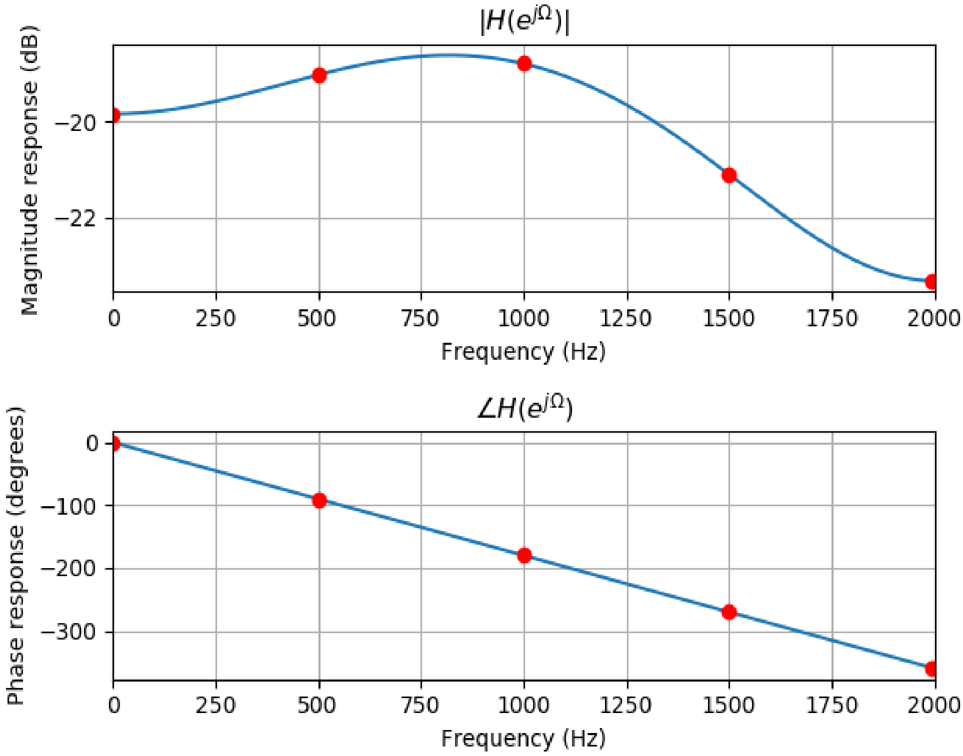
$$H(z) = -0.007484 + 0.008413z^{-1} + 0.1z^{-2} + 0.008413z^{-3} - 0.007484z^{-4}$$

c. Determine the DSP equation

$$y(n) = -0.007484x(n) + 0.008413x(n-1) + 0.1x(n-2) + 0.008413x(n-3) - 0.007484x(n-4)$$

d. Set up MATLAB routine "freqz()" to obtain the frequency response plot.

```
freqz([-0.007484, 0.008413, 0.1, 0.008413, -0.007484], [1], 4096, fs)
% B(z) = [-0.007484, 0.008413, 0.1, 0.008413, -0.007484]
% A(z) = [1]
% 4096 points for plot
% fs: 4000 Hz sampling rate
```



$|H(e^{j\Omega})|$
Magnitude response (dB)
−20
−22
0
250
500
750
1000
1250
1500
1750
2000
Frequency (Hz)
$\angle H(e^{j\Omega})$
Phase response (degrees)
0
−100
−200
−300
0
250
500
750
1000
1250
1500
1750
2000
Frequency (Hz)

Problem 4

A DSP design engineer used the following MATLAB code to design FIR filter.

```
fs=8000;
f=[ 0 0.15 0.25  0.4  0.5 1]; % edge frequencies
m=[ 1  1   0    0    1 1]; % ideal magnitudes
w=[  10 15 10 ]; % error weight factors
format long
b=remez(24,f,m,w) % Parks-McClellen algorithm and Remez exchange
```

(1) Determine the edge frequencies in Hz for passband and stopband

(2) Determine the filter type and number of taps.

(3) Weights for optimization, that Wp and Ws

(5 points)

solution

(1) Determine the edge frequencies in Hz for passband and stopband

$$f = [0, 0.15, 0.25, 0.4, 0.5, 1] \times \frac{f_s}{2} = [0, 600, 1000, 1600, 2000, 4000]Hz$$

passband: **0 - 600**Hz and **2000 - 4000** Hz

stopband: **1000 - 1600** Hz

(2) Determine the filter type and number of taps.

filter type: **band-stop**

number of taps: **24+1=25**

(3) Weights for optimization, that Wp and Ws

$$W_p = 10, W_s = 15$$

Problem 5

Design a **second-order bandpass IIR** digital Butterworth filter with the lower cut-off frequency of 100 Hz and upper cut-off frequency of 120 Hz at a sampling frequency of 1000 Hz using the bilinear transformation method.

a. Determine the transfer function $H(z)$

b. Make a pole zero plot and determine the stability

c. Set up MATLAB routine "freqz()" to obtain the frequency response plot.

d. Determine difference equation in the direct-form I

e. Draw the realization block diagram using the direct-form II.

(10 points)

solution

a. Determine the transfer function $H(z)$

$\omega_{zp} = 2\pi \times [100, 120]$ rad/s, $f_s = 1000$ Hz

$$\omega_{sp} = 2f_s \tan(\frac{\omega_{zp}}{2f_s}) = [649.8394, 791.8560]$$

order of Butterworth: 2 / 2 = 1

The 1st-order Butterworth filter $H(s') = \frac{1}{1+s'}$

Then substitute $s' = \frac{s^2+\omega_{sp}[0]\omega_{sp}[1]}{s(\omega_{sp}[1]-\omega_{sp}[0])}$, band-pass filter

$$H(s) = \frac{142.0166s}{s^2 + 142.0166s + 514579.2334}$$

Then with BLT, transfer function

$$H(z) = H(s)|_{z=2f_s\frac{1-z^{-1}}{1+z^{-1}}} = \frac{0.0592 - 0.0592z^{-2}}{1 - 1.4527z^{-1} + 0.8816z^{-2}}$$

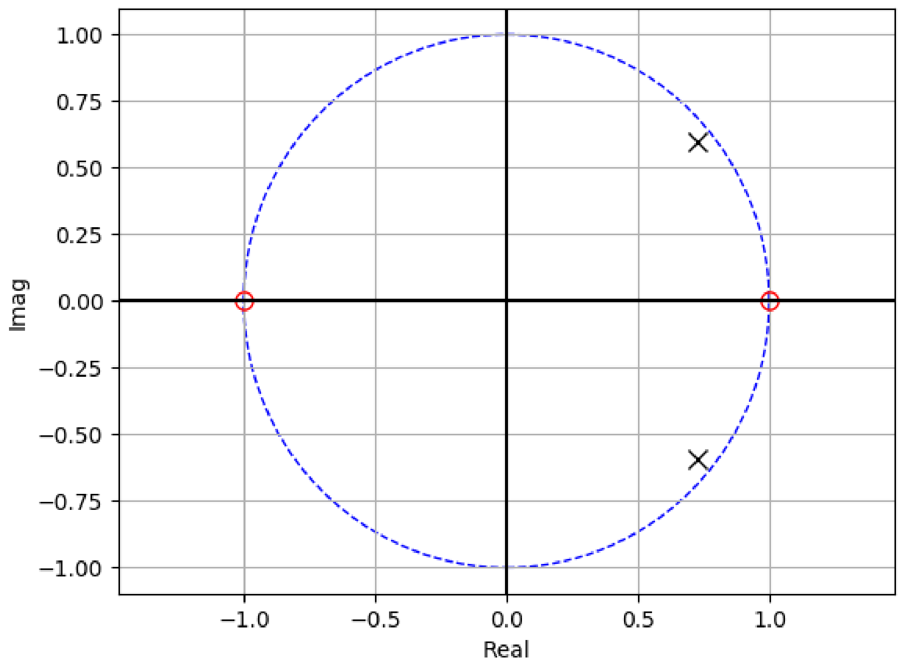
b. Make a pole zero plot and determine the stability

poles: 0.7263 -j 0.5950, 0.7263 +j 0.5950

zeros: -1, +1

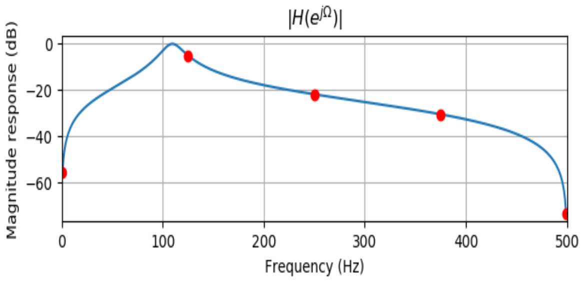
Because all poles, $|0.7263 - j0.5950| < 1, |0.7263 + j0.5950| < 1$

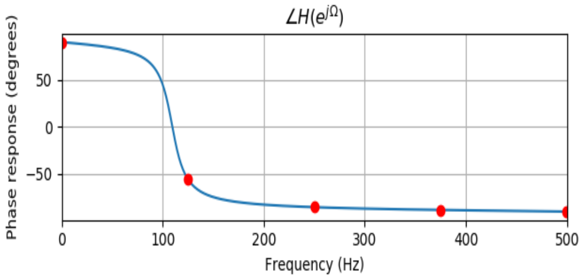
The DSP system is **stable**

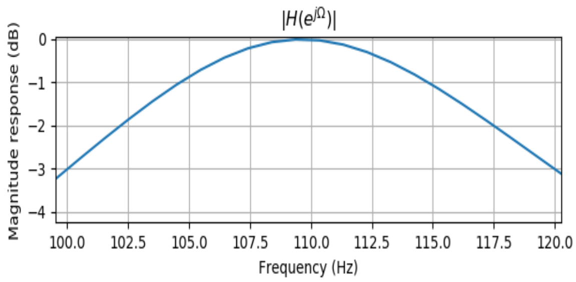


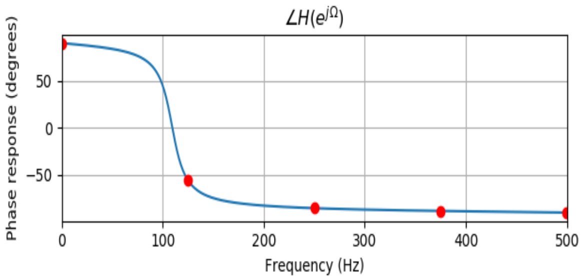
c. Set up MATLAB routine "freqz()" to obtain the frequency response plot.

```
freqz([0.0592, 0, -0.0592], [1, -1.4527, 0.8816], 4096, fs)
% B(z) = [0.0592, 0, -0.0592]
% A(z) = [1, -1.4527, 0.8816]
% 4096 points for plot
% fs: 1000 Hz sampling rate
```









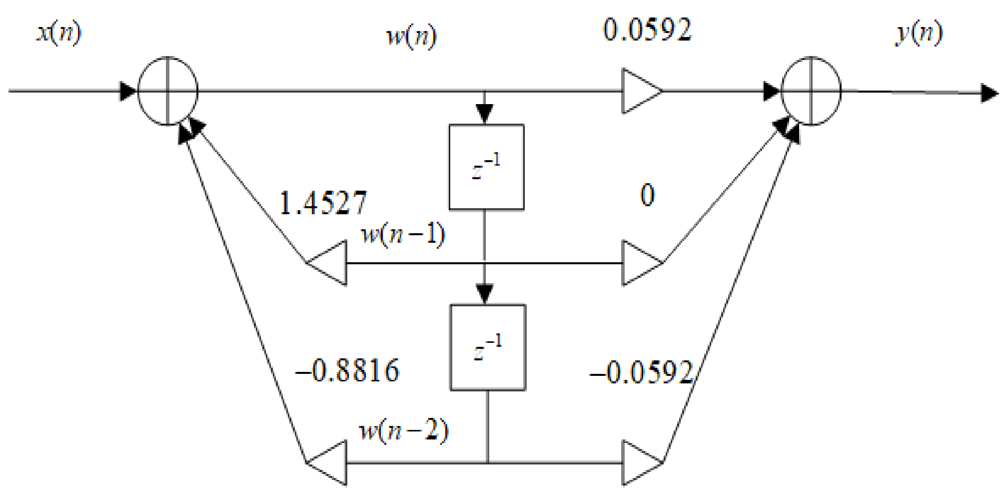
d. Determine difference equation in the direct-form I

$$y(n) = 0.0592x(n) - 0.0592x(n-2) + 1.4527y(n-1) - 0.8816y(n-2)$$

e. Draw the realization block diagram using the direct-form II.

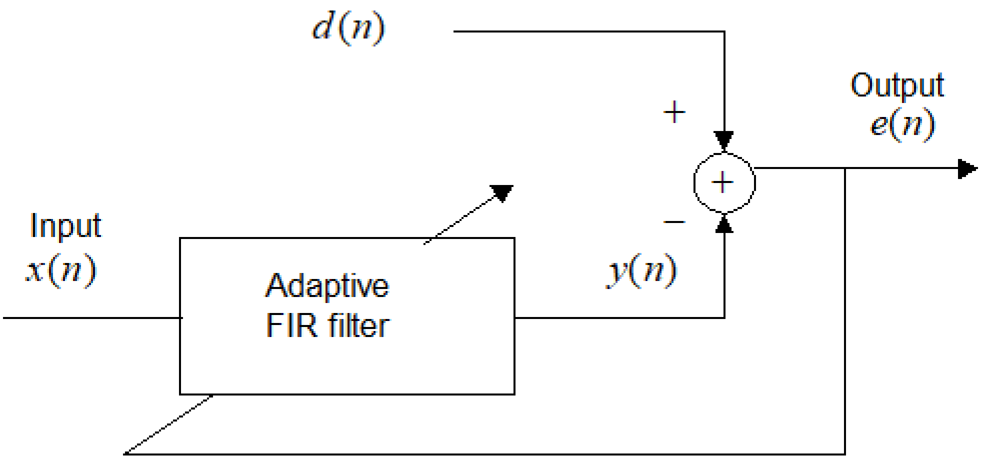
$$w(n) = x(n) + 1.4527w(n-1) - 0.8816w(n-2)$$
$$y(n) = 0.0592w(n) - 0.0592w(n-2)$$

the realization block diagram



Problem 6

For the following adaptive filter used for noise cancellation application,



$d(0) = -1, d(1) = 2, d(2) = 1, x(0) = -0.5, x(1) = 1.2, x(2) = 0.5$

and an adaptive filter with two taps: $y(n) = w_0 x(n) + w_1 x(n-1)$

with initial values $w_0 = 0.5, w_1 = -0.5$, and $\mu = 0.1$

(a) determine the **LMS** algorithm equations

$$y(n) =$$
$$e(n) =$$
$$w_0 =$$
$$w_1 =$$

(b) perform adaptive filtering for each $n = 0, 1, 2$

(10 points)

solution

(a) Determine the DSP equations using the LMS algorithm

$$y(n) = \sum_{k=0}^{1} w_k x(n-k) = w_0 x(n) + w_1 x(n-1)$$
$$e(n) = d(n) - y(n)$$
$$w_k \Leftarrow w_k + 2\mu e(n) x(n-k)$$

for k= 0, 1; that is, write the equations for all adaptive coefficients:

$$w_0 = w_0 + 2\mu e(n) x(n)$$
$$w_1 = w_1 + 2\mu e(n) x(n-1)$$

(b) perform adaptive filtering for each $n = 0, 1, 2$

Python script is below:

```
from rls.lms import Lms
from rls.rls import Rls
```

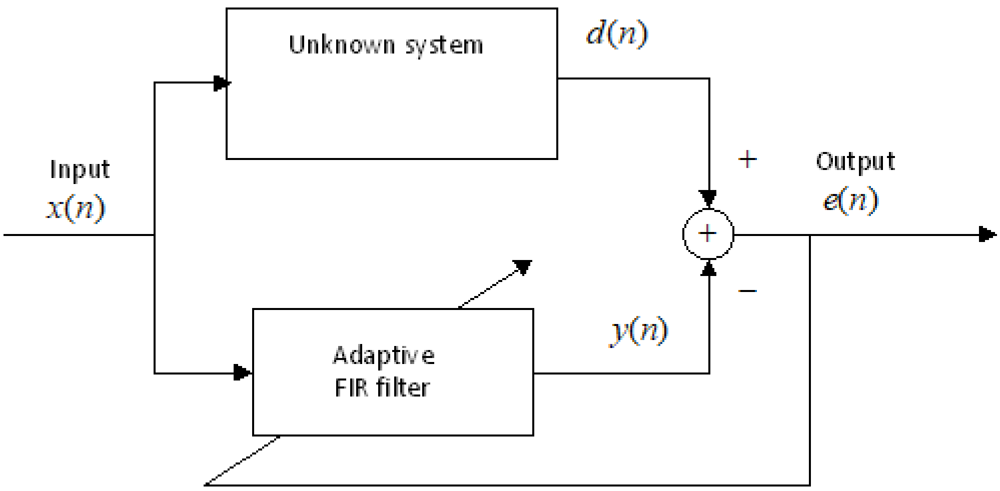
```python
list_x = [-0.5, 1.2, 0.5]
list_d = [-1, 2, 1]
#####################
# LMS adaptive filter
#####################
N, mu, w_0 = 2, 0.1, [0.5, -0.5]
lms = Lms(mu, N, w_0)
list_y, list_e, list_w = [], [], [w_0]
for x, d in list(zip(list_x, list_d)):
    lms.train(x, d)
    list_y.append(lms.y)
    list_e.append(lms.e)
    list_w.append(lms.w)
print(list_y)
print(list_e)
print(list_w)
print('\n')
```

Then

$$y(n) = [-0.25, 0.94, -0.3125]$$
$$e(n) = [-0.75, 1.06, 1.3125]$$
$$[w_0, w_1] = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} \begin{bmatrix} 0.575 \\ -0.5 \end{bmatrix} \begin{bmatrix} 0.8294 \\ -0.606 \end{bmatrix} \begin{bmatrix} 0.96065 \\ -0.291 \end{bmatrix}$$

Problem 7

Given a DSP system with a sampling rate set up to be 8,000 samples per second, implement adaptive filter with 5 taps for system modeling.



Assume that the system as the following input and output:

$x(0) = 2, x(1) = -4, x(2) = 4, x(3) = -2$

$d(0) = 1, d(1) = -1, d(2) = 0, d(3) = 1$

tow taps: $y(n) = w_0 x(n) + w_1 x(n-1)$

a. Determine the DSP equations using **the RLS algorithm** with $\delta = 1$ and $\lambda = 0.96$.

b. Perform adaptive filtering for n=0, 1, 2.

(10 points)

solution

a. Determine equations using the RLS algorithm. [initialization for $w_k = 0, Q(-1) = \delta \times I$]

$$X(n) \Leftarrow \begin{bmatrix} x(n) \\ x(n-1) \end{bmatrix}, w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$\alpha(n) = d(n) - w^T X(n) = d(n) - \sum_{k=0}^{1} w_k x(n-k)$$

$$\vec{k}(n) = [\frac{1}{\lambda + X^T(n)Q(n-1)X(n)}]Q(n-1)X(n)$$

$$Q(n) \Leftarrow \frac{1}{\lambda}[Q(n-1) - \vec{k}(n)X^T(n)Q(n-1)]$$

$$w \Leftarrow w + \alpha(n)\vec{k}(n)$$

$$y(n) = w^T X(n) = \sum_{k=0}^{4} w_k x(n-k)$$

$$e(n) = d(n) - y(n)$$

b. Repeat (b) using the RLS algorithm with δ=1 and λ=0.96.

Python script is below:

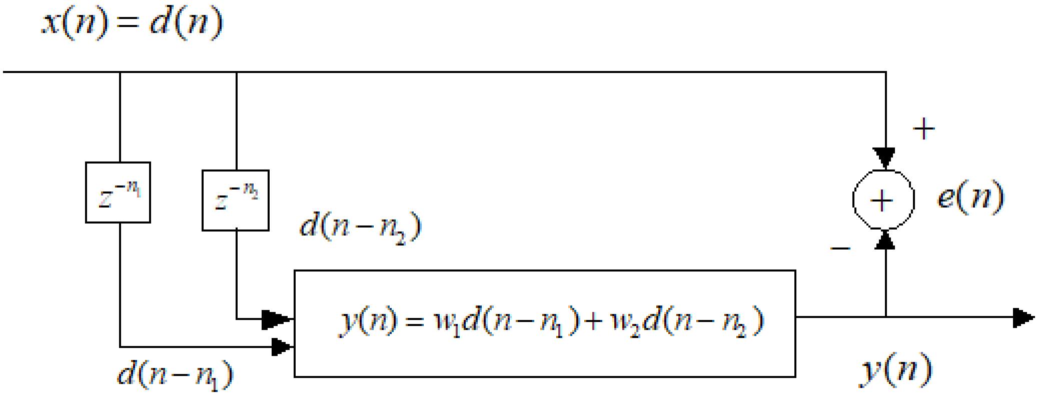
```
from rls.lms import Lms
from rls.rls import Rls
def convert(L, n):
    if isinstance(L, (float, int)):
        return round(L, n)
    list_new = []
    for elem in L:
        list_new.append(convert(elem, n))
    return list_new

def print_approx(L, n=6):
    L_new = convert(L, n)
    print(L_new)

list_x = [2, -4, 4, -2]
list_d = [1, -1, 0, 1]
#####################
# RLS adaptive filter
#####################
delta, lambda_, N = 1, 0.96, 2
rls = Rls(delta, lambda_, N) # initial w = [0, ..., 0]
list_y, list_e, list_alpha, list_w,  = [], [], [], [[0] * N]
for x, d in list(zip(list_x, list_d)):
    rls.train(x, d)
    list_y.append(rls.y)
    list_e.append(rls.e)
    list_alpha.append(rls.alpha)
    list_w.append(rls.w)
    print_approx(rls.Q)
    print_approx(rls.k)
print('\n')
print_approx(list_y)
print_approx(list_e)
print_approx(list_alpha)
print_approx(list_w)
print('\n')
```

Then

$$y(n) = [0.806452, -1.070445, 0.146298]$$
$$e(n) = [0.193548, 0.070445, -0.146298]$$
$$\alpha(n) = [1, 0.612903, -0.764695]$$
$$[w_0, w_1] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.403226 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.344049 \\ 0.152875 \end{bmatrix}, \begin{bmatrix} 0.393197 \\ 0.356623 \end{bmatrix}$$

Problem 8

The following Wiener filter is used to predict the sinusoid



Assume that $d(n) = \sin(n\Omega_1) + \cos(n\Omega_2), \Omega_1 \neq \Omega_2$ where the **Wiener** filter predictor with delays of n_1 , and n_2 is given as

$$y(n) = w_1 d(n - n_1) + w_2 d(n - n_2)$$

Find the Wiener filter coefficients: w_1, and w_2, and the minimized the cost function of

$J_{\min} = E\{[d(n) - y(n)]^2\}$

(15 points)

solution

$$\begin{aligned} J &= E\left\{e^2(n)\right\} = E\left\{\left(d(n) - w^T X(n)\right)^2\right\} \\ &= w^T E\left\{X^T(n)X(n)\right\} w - 2E\{d(n)X(n)\}^T w + E\left\{d^2(n)\right\} \\ &= w^T R w - 2P^T w + \sigma^2 \end{aligned}$$

Here we define

$$\begin{aligned} w &\equiv [w_1 \ w_2]^T \\ X(n) &\equiv \begin{bmatrix} d(n - n_1) \\ d(n - n_2)) \end{bmatrix} \end{aligned}$$

Moreover, forR, P, σ^2, we conclude that (n_1, n_2, n_3 are not equal each other)

$$\begin{aligned} R &\equiv E\left\{X^T(n)X(n)\right\} \\ &= \begin{bmatrix} E\left\{d(n-n_1)d(n-n_1)\right\} & E\left\{d(n-n_1)d(n-n_2)\right\} \\ E\left\{d(n-n_2)d(n-n_1)\right\} & E\left\{d(n-n_2)d(n-n_2)\right\} \end{bmatrix} \\ P &\equiv E\{d(n)X(n)\} \\ &= \begin{bmatrix} E\left\{d(n)d(n-n_1)\right\} \\ E\left\{d(n)d(n-n_2)\right\} \end{bmatrix} \\ \sigma^2 &\equiv E\left\{d^2(n)\right\} \end{aligned}$$

Because we know that (for $i, j \in \{1, 2\}$)

$$
\begin{aligned}
E\{d(n-n_i)d(n-n_j)\} &= E\{[\sin((n-n_i)\Omega_1)+\cos((n-n_i)\Omega_2)][\sin((n-n_j)\Omega_1)+\cos((n-n_j)\Omega_2)]\} \\
&= E\{[0.5\cos((n_j-n_i)\Omega_1)+0.5\cos((n_j-n_i)\Omega_2)]\} \\
&= 0.5\cos(|n_j-n_i|\Omega_1)+0.5\cos(|n_j-n_i|\Omega_2) \\
&= \begin{cases} 1 & i=j \\ 0.5\cos(|n_j-n_i|\Omega_1)+0.5\cos(|n_j-n_i|\Omega_2) & i\neq j \end{cases} \\
E\{d(n)d(n-n_i)\} &= 0.5\cos(|0-n_i|\Omega_1)+0.5\cos(|0-n_i|\Omega_2) \\
&= 0.5\cos(n_i\Omega_1)+0.5\cos(n_i\Omega_2) \\
E\left\{d^2(n)\right\} &= 0.5\cos(|0-0|\Omega_1)+0.5\cos(|0-0|\Omega_2) \\
&= 1
\end{aligned}
$$

Thus forR, P, σ^2, we conclude

$$
\begin{aligned}
R &= \begin{bmatrix} E\{d(n-n_1)d(n-n_1)\} & E\{d(n-n_1)d(n-n_2)\} \\ E\{d(n-n_2)d(n-n_1)\} & E\{d(n-n_2)d(n-n_2)\} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0.5\cos(|n_1-n_2|\Omega_1)+0.5\cos(|n_1-n_2|\Omega_2) \\ 0.5\cos(|n_1-n_2|\Omega_1)+0.5\cos(|n_1-n_2|\Omega_2) & 1 \end{bmatrix} \\
P &= \begin{bmatrix} E\{d(n)d(n-n_1)\} \\ E\{d(n)d(n-n_2)\} \end{bmatrix} = \begin{bmatrix} 0.5\cos(n_1\Omega_1)+0.5\cos(n_1\Omega_2) \\ 0.5\cos(n_2\Omega_1)+0.5\cos(n_2\Omega_2) \end{bmatrix} \\
\sigma^2 &= E\left\{d^2(n)\right\} = 1
\end{aligned}
$$

Find the Wiener filter coefficients w_* , here **if R is invertible**

$$
w_* = R^{-1}P = \begin{bmatrix} 1 & \cos(|n_1-n_2|\Omega) \\ \cos(|n_1-n_2|\Omega) & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \cos(n_1\Omega) \\ \cos(n_2\Omega) \end{bmatrix}
$$

Here we have

$$
\begin{aligned}
R^{-1} &= \frac{\operatorname{adj}(R)}{\det(R)} = \frac{1}{\det(R)}\begin{bmatrix} 1 & -\cos(|n_1-n_2|\Omega) \\ -\cos(|n_1-n_2|\Omega) & 1 \end{bmatrix} \\
w_* &= R^{-1}P \\
&= \frac{1}{\det(R)}\begin{bmatrix} 1 & -0.5\cos(|n_1-n_2|\Omega_1)-0.5\cos(|n_1-n_2|\Omega_2) \\ -0.5\cos(|n_1-n_2|\Omega_1)-0.5\cos(|n_1-n_2|\Omega_2) & 1 \end{bmatrix} \\
&\cdot \begin{bmatrix} 0.5\cos(n_1\Omega_1)+0.5\cos(n_1\Omega_2) \\ 0.5\cos(n_2\Omega_1)+0.5\cos(n_2\Omega_2) \end{bmatrix}
\end{aligned}
$$

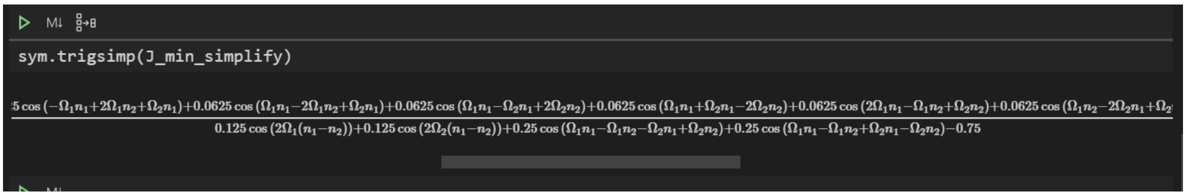
Finally, we compute with python **Sympy** library, w_* is

$$
\begin{bmatrix}
\frac{1.0(-0.375\cos(\Omega_1 n_1)+0.125\cos(\Omega_1(n_1-2n_2))-0.375\cos(\Omega_2 n_1)+0.125\cos(\Omega_2(n_1-2n_2))+0.125\cos(-\Omega_1 n_1+\Omega_1 n_2+\Omega_2 n_2)+0.125\cos(\Omega_1 n_1-\Omega_1 n_2+\Omega_2 n_2)+0.125\cos(\Omega_1 n_2-\Omega_2 n_1+\Omega_2 n_2)+0.125\cos(\Omega_1 n_2+\Omega_2 n_1-\Omega_2 n_2))}{0.125\cos(2\Omega_1(n_1-n_2))+0.125\cos(2\Omega_2(n_1-n_2))+0.25\cos(\Omega_1 n_1-\Omega_1 n_2-\Omega_2 n_1+\Omega_2 n_2)+0.25\cos(\Omega_1 n_1-\Omega_1 n_2+\Omega_2 n_1-\Omega_2 n_2)-0.75} \\
\frac{1.0(-0.375\cos(\Omega_1 n_2)+0.125\cos(\Omega_1(2n_1-n_2))-0.375\cos(\Omega_2 n_2)+0.125\cos(\Omega_2(2n_1-n_2))+0.125\cos(-\Omega_1 n_1+\Omega_1 n_2+\Omega_2 n_1)+0.125\cos(\Omega_1 n_1-\Omega_1 n_2+\Omega_2 n_1)+0.125\cos(\Omega_1 n_1-\Omega_2 n_1+\Omega_2 n_2)+0.125\cos(\Omega_1 n_1+\Omega_2 n_1-\Omega_2 n_2))}{0.125\cos(2\Omega_1(n_1-n_2))+0.125\cos(2\Omega_2(n_1-n_2))+0.25\cos(\Omega_1 n_1-\Omega_1 n_2-\Omega_2 n_1+\Omega_2 n_2)+0.25\cos(\Omega_1 n_1-\Omega_1 n_2+\Omega_2 n_1-\Omega_2 n_2)-0.75}
\end{bmatrix}
$$

Here is the python script to compute w_*

```
import sympy as sym
from sympy import cos, sin
n1, n2, n3, omega1, omega2 = sym.symbols('n_1, n_2, n_3, Omega_1, Omega_2')
R = sym.Matrix([[1, 0.5*cos((n1-n2)*omega1) + 0.5*cos((n1-n2)*omega2)],
                [0.5*cos((n1-n2)*omega1) + 0.5*cos((n1-n2)*omega2), 1]])
P = sym.Matrix([[0.5*cos(n1*omega1) + 0.5*cos(n1*omega2)],
                [0.5*cos(n2*omega1) + 0.5*cos(n2*omega2)]])
det = R.det()
sym.trigsimp(sym.cancel(det)) # simplity det(R)
eigen = R.eigenvals()
sym.trigsimp(sym.cancel(eigen)) # find eigen value of R
w = R.inv()* P                  # find Wiener filter w_*
w = sym.trigsimp(sym.cancel(w)) # simplifty w_*
sym.cancel(w)
```

```python
J_min = -P.transpose() * w + sig_sq # find J_min
J_min_simplify = sym.cancel(sym.trigsimp(J_min))
print(sym.latex(J_min_simplify))
```



As we can see, the minimized the cost function of $J_{\text{min}} = -P^T w_* + \sigma^2$ is a very complex and long expression

Actually, we need **4th-order** difference equation to describe the $d(n)$,

2nd-order is not enough

Problem 9

For the sampling conversion from 7 kHz to 3 kHz with the following specifications:

- Passband frequency range = 0 – 500 Hz
- Passband ripple = 0.02 dB
- Stopband attenuation = 46 dB,

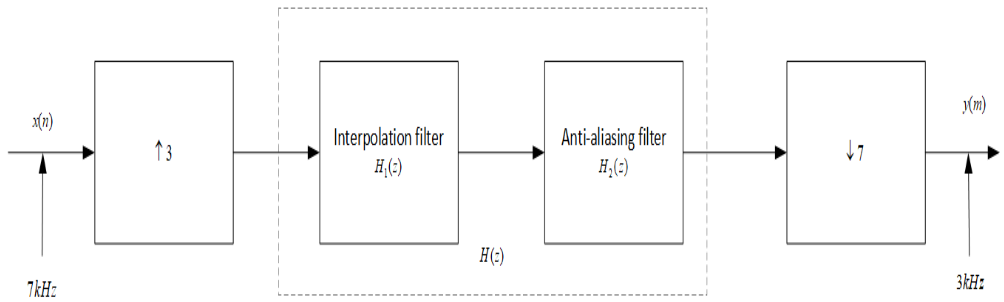
a. draw the block diagram for the interpolator;

b. determine the window type, filter length, and cutoff frequency if the window method is used for the combined FIR filter H(z).

(10 points)

solution

a. draw the block diagram for the interpolator;



b. determine the window type, filter length, and cutoff frequency if the window method is used for the combined FIR filter H(z).

For interpolation filter:

$$f_{stop} = \frac{f_s}{2} = 3.5kHz$$

For Anti-aliasing filter:

$$f_{stop} = \frac{(f_s \times L)/M}{2} = 1.5kHz$$

Because 3 kHz < 4 kHz, we choose $f_{stop} = \min(3.5, 1.5) = 1.5$ kHz

From table, Passband ripple<0.02 dB Stopband attenuation>46 dB,

window type: **Hamming**

$$f_{pass} = 500Hz, f_{stop} = 1.5kHz$$
$$\Delta f = \frac{f_{stop} - f_{pass}}{f_s \times L} = 1/21 = 0.04762$$
$$N = \frac{3.3}{\Delta f} = 69.3$$

select the closest odd number $N = 71$

cutoff frequency

$$f_c = \frac{f_{pass} + f_{stop}}{2} = 1kHz$$

Problem 10

For the design of a two-stage decimator (M1xM2=5x3) with the following specifications:

- Original sampling rate = 15 kHz
- Frequency of interest = 0 - 250 Hz
- Passband ripple = 0.05 (absolute)
- Stopband attenuation = 0.005 (absolute)
- Final sampling rate = 1,000 Hz,

a. Draw the decimation block diagram;

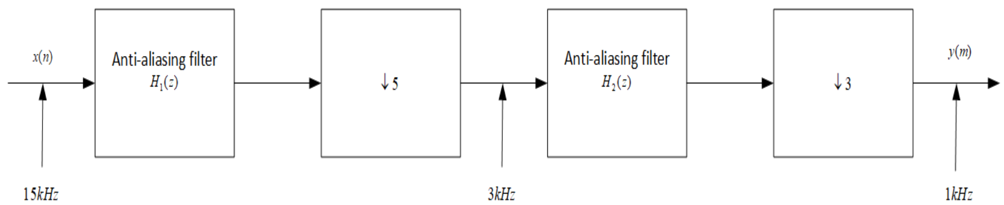
b. Specify the sampling rate for each stage;

c. determine the window type, filter length, and cutoff frequency for each stage if the window method is used for anti-aliasing FIR filter design

(10 points)

solution

a. Draw the decimation block diagram;



b. Specify the sampling rate for each stage;

$$\frac{15kHz}{1kHz} = 15 = 5 \times 3 = M_1 \times M_2$$

Here we select the sampling rate $M_1 = 5$ for stage 1

the sampling rate $M_2 = 3$ for stage 2.

c.

A. determine the window type, filter length, and cutoff frequency for the **first stage** H1(z);

$$20\log_{10}(1/0.005) = 46.02dB$$

From table, Passband ripple<0.05 dB Stopband attenuation>46.02 dB,

window type: **Hamming**

filter length,

$$f_{pass} = 0.25kHz, f_{stop} = \frac{f_s/M_1}{2} = 1.5kHz$$
$$\Delta f = \frac{(f_{stop} - f_{pass})}{f_s} = 1.25/15 = 0.08333$$
$$N = \frac{3.3}{\Delta f} = 39.6$$

select the closest odd number $N = 41$

cutoff frequency

$$f_c = \frac{f_{pass} + f_{stop}}{2} = 0.875kHz = 875Hz$$

B. determine the window type, filter length, and cutoff frequency for the **second stage** H2(z)

From table, Passband ripple<0.05 dB Stopband attenuation>46.02 dB,

window type: **Hamming**

filter length,

$$f_{pass} = 0.25kHz, f_{stop} = \frac{f_s/(M_1 M_2)}{2} = 0.5kHz$$
$$\Delta f = \frac{(f_{stop} - f_{pass})}{f_s/M_1} = 0.25/3 = 0.08333$$
$$N = \frac{3.3}{\Delta f} = 39.6$$

select the closest odd number $N = 41$

cutoff frequency

$$f_c = \frac{f_{pass} + f_{stop}}{2} = 0.375kHz = 375Hz$$