**Introduction to Digital Signals**

**&**

**Information Processing**

19 | 07 | 2023

**Ben Cohen**

**Dolev Mishali**

**Table Of Content**

[Q1 **3**](#_Toc140687068)

[Part 1 - Code **3**](#_Toc140687069)

[Part 1 - Images **4**](#_Toc140687070)

[Part 2 - Code **4**](#_Toc140687071)

[Part 2 - Images **5**](#_Toc140687072)

[Q2 **6**](#_Toc140687073)

[Part 1 - Code **6**](#_Toc140687074)

[Part 1 - Images **7**](#_Toc140687075)

[Part 2 - Code **7**](#_Toc140687076)

[Part 2 - Images **8**](#_Toc140687077)

[Q3 **9**](#_Toc140687078)

[A - Code **9**](#_Toc140687079)

[A - Images **10**](#_Toc140687080)

[B - Code **10**](#_Toc140687081)

[B - Images **11**](#_Toc140687082)

[C - Code **12**](#_Toc140687083)

[C - Images **13**](#_Toc140687084)

[D - Code **14**](#_Toc140687085)

[D - Images **15**](#_Toc140687086)

[E - Code **15**](#_Toc140687087)

[E - Images **16**](#_Toc140687088)

[Q4 **17**](#_Toc140687089)

[Part 1 - A - Code **17**](#_Toc140687090)

[Part 1 - A - Image **18**](#_Toc140687091)

[Part 1 - B - Calculation Image **19**](#_Toc140687092)

[Part 1 - C + D - Code **19**](#_Toc140687093)

[Part 1 - C + D – Images **21**](#_Toc140687094)

[**Part 2 - A - Code 21**](#_Toc140687095)

[Part 2 - B - Code **22**](#_Toc140687096)

[Part 2 - B – Image **23**](#_Toc140687097)

[Q5 **24**](#_Toc140687098)

[Part 1 - Code **24**](#_Toc140687099)

[Part 1 – Images **25**](#_Toc140687100)

[Videos Links **26**](#_Toc140687101)

[Q1 **26**](#_Toc140687102)

[Q2 **26**](#_Toc140687103)

[Q3 **26**](#_Toc140687104)

[Q4 **26**](#_Toc140687105)

[Q5 **26**](#_Toc140687106)

# **Q1**

### **Part 1 - Code**

%% Impulse Response

clear;

clc;

n = 0:9;

r=0.98;

fs=300;

b = 1-r^10; % numerator coefficient of the filter function

a = [1,0,0,0,0,0,0,0,0,0,r^10]; % denominator coefficient of the filter function

figure;

impulse = [1, zeros(1, fs)]; % create an impulse signal with 1 followed by fs zeros

h = filter(b, a, impulse); % apply the filter to the impulse signal and get the impulse response

subplot(1,2,1);

stem(0:fs, h, 'filled', 'r');

title('Impulse Response');

xlabel('n');

ylabel('h[n]');

grid on;

%% Magnitude

%Drawing for magnitude of the frequency response:

[h,f] = freqz(b,a,fs,fs); % calculate frequency response at fs points

subplot(1,2,2);

plot(f,abs(h)); % plot magnitude as a function of frequency in Hz

title('Magnitude Response');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

grid on;

**הסבר:**

* **מסנן מסוג FIR, מסנן בעל תגובת ההלם סופית.**
* **במסנן המוצא הוא קונבולוציה של הכניסה עם הלם במישור התדר.**
* **המסנן נקרא “Comb filter” , מאופיין בכמות שפיצים גבוהה בגרף תגובת התדר שלו והוא עבור הגברה/הנמכה של רכיבי אות מסוימים, או עבור הנמכת רעשים מאותות מחזוריים.**

### **Part 1 - Images**



### **Part 2 - Code**

%% Impulse Response

clear;

clc;

n = 0:9;

r=0.96;

fs=200;

b = (1+r^10).\*[1,0,0,0,0,0,0,0,0,0,-1]; % numerator coefficient of the filter function

a = [1 r.^n.\*[0 0 0 0 0 0 0 0 0 -1]].\*2; % denominator coefficient of the filter function

figure;

impulse = [1, zeros(1, fs)]; % create an impulse signal with 1 followed by fs zeros

h = filter(b, a, impulse); % apply the filter to the impulse signal and get the impulse response

subplot(1,2,1);

stem(0:fs, h, 'filled', 'r');

title('Impulse Response (filter)');

xlabel('n');

ylabel('h[n]');

grid on;

%% Magnitude

[h,f] = freqz(b,a,fs,fs); % calculate frequency response at fs points

subplot(1,2,2);

plot(f,abs(h)); % plot magnitude as a function of frequency in Hz

title('Magnitude Response');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

grid on;

**הסבר:**

* **מסנן מסוג FIR, מסנן בעל תגובת ההלם סופית.**
* **בשונה מהמסנן הקודם, זה הוא מסנן מסוג Notch Filter, מנחית טווח צר של תדרים סביב נקודה מסוימת ולא נוגע בשאר התדרים.**
* **עבור הגברה/הנמכה של רכיבי אות מסוימים, או עבור הנמכת רעשים מאותות מחזוריים ועבור הנמכת רעשים מאותות מחזוריים.**

### **Part 2 - Images**



# **Q2**

### **Part 1 - Code**

%% The magnitude of the DFT:

clear;

clc;

% Define the function y

y = @(x,a) (0 <= abs(x) & abs(x) <= a).\*0 + (a < abs(x)).\*x;

% Assign the values of the given parameters

F\_0 = 20;

N = 100;

fs = 2000;

T = 1/fs;

a = 0.25; % Dead-zone parameter

% Create the input signal x[n]

n = 0:N-1;

x = cos(2.\*pi.\*F\_0.\*n.\*T);

% Evaluate the output signal y[n] using the function y

y\_out = y(x,a); % Calculate output signal

% Plot the DFT of y[n] as a function of frequency f

Y = fft(y\_out); % Compute DFT of output signal

f = linspace(0,fs/2,N/2); % Create frequency vector

% Plot DFT magnitude spectrum vs frequency

stem(f,abs(Y(1:N/2)), "filled", 'red')

xlabel('Frequency (Hz)')

ylabel('Magnitude')

title('DFT of y[n]')

grid on

%% Calculate the THD:

Y = fft(y\_out);

d = abs((Y)/(N/2));

% Calculate power of output signal

Py = (d(1).^2)/4 + (1/2)\*sum(d(2:(N/2)-1).^2);

% Calculate THD of output signal

Y\_thd = 100.\*(Py-d(2).^2/2)./Py;

disp(['The THD of y[n] is ', num2str(Y\_thd), '%'])

**הסבר:**

* **מסנן DEADZONE, לא לינארי שמגדיר את המוצא כ0 במידה ואות הכניסה נמצא בתחום ערכים מסוים, התלוי בפרמטר אלפא.**
* **בעל עיוות באותר המוצא, ונדרש לחשב עיוות זה על מנת לדעת מה היינו אמורים לקבל. מדד העיוות נקרא THD והוא נמדד באחוזים.**

### **Part 1 - Images**

The THD of y[n] is 0.93193%



### **Part 2 - Code**

%% The magnitude of the DFT:

clear;

clc;

% Define the function y

y = @(x) (x.^3);

% Assign the values of the given parameters

F\_0 = 20;

N = 50;

fs = 1000;

T = 1/fs;

a = 0.25; % Dead-zone parameter

% Create the input signal x[n]

n = 0:N-1;

x = cos(2.\*pi.\*F\_0.\*n.\*T);

% Evaluate the output signal y[n] using the function y

y\_out = y(x); % Calculate output signal

% Plot the DFT of y[n] as a function of frequency f

Y = fft(y\_out); % Compute DFT

f = linspace(0,fs/2,N/2); % Create frequency vector

% Plot DFT magnitude

stem(f,abs(Y(1:N/2)), "filled", 'red')

xlabel('Frequency (Hz)') % Label x-axis

ylabel('Magnitude') % Label y-axis

title('DFT of y[n]') % Add title

grid on % Add grid lines

%% Calculate the THD:

Y = fft(y\_out);

d = abs((Y)/(N/2));

% Calculate power of output signal

Py = (d(1).^2)/4 + (1/2)\*sum(d(2:(N/2)-1).^2);

% Calculate THD of output signal

Y\_thd = 100.\*(Py-d(2).^2/2)./Py;

disp(['The THD of y[n] is ', num2str(Y\_thd), '%'])

**הסבר:**

* **מסנן DEADZONE, לא לינארי שמגדיר את המוצא כ0 במידה ואות הכניסה נמצא בתחום ערכים מסוים, התלוי בפרמטר אלפא.**
* **בשונה מהמסנן הקודם, זהו מסנן לא לינארי, מסנן אי זוגי ובעל התכונה של חוסר הזיכרון.**
* **המסנן הזה עבור תדרים מסויימים ערך המגניטודה יגדל, ועבור ערכים מסויימים ערך המגניטודה יקטן.**

### **Part 2 - Images**

The THD of y[n] is 10%



# **Q3**

### **A - Code**

%% A

clear;

clc;

% Define the data

r = 0.9;

n = 0:127;

% Define delta function

x = [1 zeros(1,127)];

y = zeros(1,128);

% Set the first value of y[n] to the first value of x[n]

y(1) = x(1);

% Set the second value of y[n] using the formula given in the problem

y(2) = -r^2\*y(1) + x(2);

% Use a for loop to calculate the remaining values of y[n]

for i=3:length(n)

y(i) = -r^2\*y(i-2) + x(i);

end

% Calculate the DFT of y[n]

Y = fft(y);

% Plot the magnitude of Y[k]

figure;

k = (1:1:128);

stem(k,abs(Y), "filled", 'red');

title('Magnitude of Y[k]');

grid on;

xlabel('k');

ylabel('|Y[k]|');

xlim([1,128]);

**הסבר:**

* **בשאלה זאת קיבלנו מימוש של מסנן בעזרת משוואת הפרשים.**
* **משוואת הפרשים זאת מממשת מסנן שמחליק את הרכיבים באות בעלי תדר גבוה או מחליק רכיב תדר מסוים בעל אופי התנהגות מחזורי אותו אנחנו לא רוצים.**
* **משוואת ההפרשים שלנו היא משוואת הפרשים(משוואה רקורסיבית) מסדר שני, ומסננים מסוג זה הםIIR ולכן קיימת תגובת הלם אין סופית ולא נוכל לממש להם פאזה לינארית.**
* **במסנן כזה w(0) יהיה תדר תהודה וr- יהיה הרדיוס של הקטבים במפת הקטבים/אפסים.**

**כמה ש-r יהיה יותר קרוב ל1, כך תגובת המסנן תהיה חדה יותר בקרבה לתדר בו נקבל ערך מקסימלי במגניטודה של תגובת התדר.**

* **ניתן גם לראות שבקוד הגדרנו את w(0) להיות 0 ולכן המסנן הזה יעביר רכיבי תדר נמוכים, כלומר יתנהג כLOW PASS FILTER.**

### **A - Images**



### **B - Code**

%% B

clear;

clc;

% Define the data

r = 0.9;

n = 0:127;

% Define delta function

x = [1 zeros(1,127)];

y = zeros(1,128);

% Set the first value of y[n] to the first value of x[n]

y(1) = x(1);

% Set the second value of y[n] using the formula given in the problem

y(2) = -r^2\*y(1) + x(2);

% Use a for loop to calculate the remaining values of y[n]

for i=3:length(n)

y(i) = -r^2\*y(i-2) + x(i);

end

% Calculate w[n]

w = (0.92.^-n).\*y;

% Calculate the DFT of w[n]

W = fft(w);

% Plot the magnitude of W[k]

figure;

k = (1:1:128);

stem(k,abs(W), "filled", 'red');

title('Magnitude of W[k]');

grid on;

xlabel('k');

ylabel('|W[k]|');

xlim([1,128]);

**הסבר:**

* **בחלק 2 ההגדרות מאוד דומות והמימוש כמעט זהה. השוני היחיד הוא ההגדרה של w, ההשפעה של הw הזה באה לידי ביטוי בכך שהיא נותנת פקטור שינוי גודל לכל אלמנט של y שתלוי באינדקס n.**
* **הפקטור הזה דועך ככל שn גדל, כאשר y בעל גבולות רחבים יותר יונמך מy בעל גבולות נמוכים יותר.**
* **נראית כפונקציית חלון שמשנה את הגודל של y בעזרת הצרה של הגבולות של החלון לפי הn- .**

### **B - Images**



### **C - Code**

%% C

clear;

clc;

% Define the data

r = 0.5;

n = 0:127;

% Define delta function

x = [1 zeros(1,127)];

y = zeros(1,128);

% Set the first value of y[n] to the first value of x[n]

y(1) = x(1);

% Set the second value of y[n] using the formula given in the problem

y(2) = -r^2\*y(1) + x(2);

% Use a for loop to calculate the remaining values of y[n]

for i=3:length(n)

y(i) = -r^2\*y(i-2) + x(i);

end

% Calculate the DFT of y[n]

Y = fft(y);

% Plot the magnitude of Y[k]

figure;

k = (1:1:128);

stem(k, abs(Y), "filled", 'red');

title('Magnitude of Y[k]');

xlabel('k');

ylabel('|Y[k]|');

grid on;

xlim([1,128]);

**הסבר על ג' מופיע בהסבר של סעיף ד'.**

### **C - Images**



### **D - Code**

%% D

clear;

clc;

% Define the data

r = 0.5;

n = 0:127;

% Define delta function

x = [1 zeros(1,127)];

y = zeros(1,128);

% Set the first value of y[n] to the first value of x[n]

y(1) = x(1);

% Set the second value of y[n] using the formula given in the problem

y(2) = -r^2\*y(1) + x(2);

% Use a for loop to calculate the remaining values of y[n]

for i=3:length(n)

y(i) = -r^2\*y(i-2) + x(i);

end

% Calculate w[n]

w = (0.55.^-n).\*y;

% Calculate the DFT of w[n]

W = fft(w);

% Plot the magnitude of W[k]

figure;

k = (1:1:128);

stem(k, abs(W), "filled", 'red');

title('Magnitude of W[k]');

xlabel('k');

ylabel('|W[k]|');

grid on;

xlim([1,128]);

**הסבר:**

* **בסעיף ג' שינינו את הr מ0.9 ל 0.5 – המשמעות של זה היא שהזזנו את הקטבים של המסנן קרוב יותר למקור, מה שאומר שלמסנן הזה תהיה פחות חדות ופחות גובה בערך המקסימלי בגרף המגניטודה.**
* **בסעיף ד' גודל החלון קטן, בעזרת שינוי של w מ0.92 ל0.55 ו- המשמעות של זה היא**

**שערכי y בעלי גבולות רחבים יותר יונמכו אפילו יותר. זליגה ספקטרלית מונמכת וגם הנמכה את האמפליטודה של האונה המרכזית.**

### **D - Images**



### **E - Code**

%% E

clear;

clc;

% Define the data

r = 0.5;

n = 0:127;

% Define delta function

x = [1 zeros(1,127)];

y = zeros(1,128);

% Set the first value of y[n] to the first value of x[n]

y(1) = x(1);

% Set the second value of y[n] using the formula given in the problem

y(2) = -r^2\*y(1) + x(2);

% Use a for loop to calculate the remaining values of y[n]

for i=3:length(n)

y(i) = -r^2\*y(i-2) + x(i);

end

% Add noise to y[n]

y = y + sqrt(0.1)\*randn(1,128);

% Calculate w[n]

w = (0.55.^-n).\*y;

% Calculate the DFT of w[n]

W = fft(w);

% Plot the magnitude of W[k]

figure;

k = (1:1:128);

stem(abs(W), "filled", 'red');

title('Magnitude of W[k]');

xlabel('k');

ylabel('|W[k]|');

grid on;

xlim([1,128]);

**הסבר:**

* **השוני הוא הוספת רעש אקראי.**
* **לאחר הוספת הרעש האקראי, מתבצע סינון על הרעש הנתון בעזרת סינון שהגדרנו קודם כאגרסיבי.**

### **E - Images**

מכיוון שהקלט משתנה בכל הרצה, צירפנו תמונות של הרצה לדוגמא.





# **Q4**

### **Part 1 - A - Code**

%% A

% Define the transfer function H(Z)

z = tf('z');

H = (0.2\*((z+0.5)^2+1.5^2))/(z^2-0.64);

% Get the numerator and denominator coefficients of H(Z)

[num,den] = tfdata(H,'v');

% Compute the frequency response of H(Z)

[h,f] = freqz(num,den);

% Plot the magnitude of the frequency response

figure;

subplot(2,1,1)

plot(f,abs(h))

xlabel('Frequency (Hz)')

ylabel('Magnitude')

title('Magnitude of frequency response of H(Z)')

grid on;

xlim([0 3.14]);

% Compute the poles and zeros of H(Z)

p = roots(den);

z = roots(num);

% Plot the pole-zero map of the filter

subplot(2,1,2)

zplane(z,p)

xlabel('Real part')

ylabel('Imaginary part')

legend('Zeros','Poles')

title('Pole-Zero map of H(Z)')

grid on

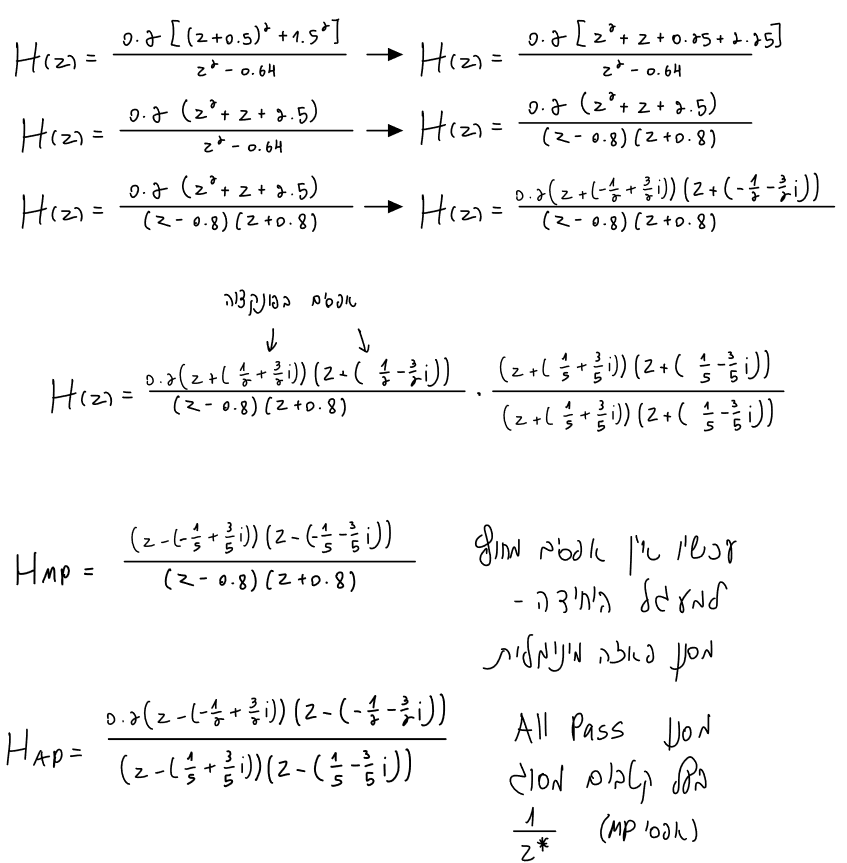
**הסבר על כל שאלה 4 חלק 1:**

* **המסנן מסוג BAND STOP FILTER.**
* **בסעיף ב' התבקשנו לחשב אנליטית את הייצוג של המסנן בצורה ו- .**
* **בהתבסס על הניתוח המתמטי של האות, הבנו שעל מנת להשיג מסנן עם פאזה לינארית קטנה ביותר אנחנו צריכים “להחליף” את הקטבים שיוצאים מחוץ לעיגול היחידה במסנן המקורי, ו"להעביר" אותם למסנן הALL PASS על ידי יצירת אפסים במסנן הALL PASS ש"מבטלים" את הקטבים שיוצאים מחוץ למעגל היחידה עבור המסנן הנתון, שאנחנו בונים.**
* **בסעיף ג' השתמשנו בחישוב מסעיף ב', על מנת לצייר את המגניטודה של תגובה התדר של כל אחד מהצורות שהתבקשנו לחשב בסעיף הקודם.**
* **בסעיף ד' התבקשנו לצייר את מפות הקטבים והאפסים של כל אחד מהרכיבים במסנן שחושבו בסעיף ב'.**

### **Part 1 - A - Image**



### **Part 1 - B - Calculation Image**



### **Part 1 - C + D - Code**

%Transfer function model

z = tf('z');

%Minimal Phase Filter

H\_MP = (z-(-1/5+3i/5))\*(z-(-1/5-3i/5))/((z-0.8)\*(z+0.8));

%All Pass Filter

H\_AP = 0.2\*(z-(-1/2+3i/2))\*(z-(-1/2-3i/2))/((z-(-1/5+3i/5))\*(z-(-1/5-3i/5)));

% Get the numerator and denominator coefficients of H\_AP(Z)

[num\_ap,den\_ap] = tfdata(H\_AP,'v');

[h\_ap,f\_ap] = freqz(num\_ap,den\_ap);

% Get the numerator and denominator coefficients of H\_MP(Z)

[num\_mp,den\_mp] = tfdata(H\_MP,'v');

[h\_mp,f\_mp] = freqz(num\_mp,den\_mp);

% Compute the poles and zeros of H\_AP(Z)

p\_ap = roots(den\_ap);

z\_ap = roots(num\_ap);

% Compute the poles and zeros of H\_MP(Z)

p\_mp = roots(den\_mp);

z\_mp = roots(num\_mp);

% Plot the magnitude of the H\_AP

figure;

subplot(4,1,1)

plot(f\_ap,abs(h\_ap))

xlabel('Frequency (Hz)')

ylabel('Magnitude')

title('Magnitude of frequency response of H\_A\_P(Z)')

grid on;

xlim([0 3.14]);

% Plot the pole-zero map of the H\_AP

subplot(4,1,2)

zplane(z\_ap,p\_ap)

xlabel('Real part')

ylabel('Imaginary part')

legend('Zeros','Poles')

title('Pole-Zero map of H\_A\_P(Z)')

grid on

% Plot the magnitude of the H\_MP

subplot(4,1,3)

plot(f\_mp,abs(h\_mp))

xlabel('Frequency (Hz)')

ylabel('Magnitude')

title('Magnitude of frequency response of H\_M\_P(Z)')

grid on;

xlim([0 3.14]);

% Plot the pole-zero map of the H\_MP

subplot(4,1,4)

zplane(z\_mp,p\_mp)

xlabel('Real part')

ylabel('Imaginary part')

legend('Zeros','Poles')

title('Pole-Zero map of H\_M\_P(Z)')

grid on

### **Part 1 - C + D – Images**



### Part 2 - A - Code

function X = dft\_filter(k,x)

% A

% This function calculates the DFT values for a signal x using equations

% (1),(2) and filter()

% k is a vector of the k values to calculate the DFT for

% x is a vector of the input signal

% X is a vector of the DFT values

% Get the length of the signal

N = length(x);

% Define the filter coefficients b and a

W = exp(-1j\*2\*pi/N); %Defining W

b = [1, 0]; % Numerator coefficient

a = [1 W]; % Denominator coefficient

% Initialize the output vector X

X = zeros (1, length(k));

% Loop over the k values

for i = 1:length(k)

% Calculate y\_k using the filter function

%Instead of implementing W^(-k(i)) we used the line below:

%since a(0) = 1 it will not affect it and do what we want for the

%rest of of the W's.

y\_k = filter(b,-(a.^-k(i)),x); %(1)

% Calculate X[k] using equation (2)

X(i) = y\_k(N);

end

end

### **Part 2 - B - Code**

%% B

clear; clc;

% Defining data

N = 100;

n = 0:N-1;

fs = 8000;

T = 1/fs;

f\_d = [490 1280 2730 3120]; % Vector of f\_d values

x = zeros (4,length (n)); % Matrix of x values

figure;

% We figured that the only possible k's that could obtain those f\_d.

% frequencies are k between 0-100 (periodical) and the ks that match to the

% formula: f/d/(fs/N) because those are the ks that represent the

% frequencies 490(k=6.125), 1280(k=16) ,2730(k=34.125), 3120(k=39).

% using dft properties: f\_k = K/NT = Kfs/N (resolution in frequency):

% for each k which is not an integer, we will get spectral leakage.

% we can see the spectral leakage for the said k values as X(k) values that are

% not equal to 0 in places that are different from the impulses.

%k = fd/Δ(f), Δ(f) = fs/N = those values are how we find the correct k's

%per f\_d

%We have put more k values so you can see the spectral leakage.

k = [0:99];

for i = 1:4

% Calculate x for each f\_d value

x = cos(2\*pi\*f\_d(i)\*n\*T);

% Calculate X using dft\_filter function

X = dft\_filter(k,x);

% Plot the magnitude of X for each f\_d value

subplot (4,1,i)

stem(k,abs(X),"filled",'red')

xlabel('k')

ylabel('|X[k]|')

title (['Magnitude of DFT for f\_d = ' num2str(f\_d(i)) ' Hz'])

grid on

end

**הסבר על כל שאלה 4 חלק 2:**

* **בחלק זה התבקשנו לבנות פונקציה המחשבת DFT עבור איקס ו K נתונים.**
* **השתמשנו בפונקציית filter כנדרש, על מנת לחשב את ערך y(k).**
* **בסעיף ב' התבקשנו להשתמש בפונקציה שבנינו בסעיף הקודם על מנת לגלות מה תדר fd עבור אות קוסינוסי כאשר fd יכול להיות אחת מ4 אופציות נתונות.**
* **בחלק זה הבנו שהערכים היחידים שיכולים להתקבל ללא זליגה ספקטרלית הן ערכים של k שלמים.**
* **על מנת לגלות את ערכי הk האפשריים בהתמרה עבורם יכול להתקבל האות אותו רצינו, נשתמש בנוסחה:**
* **k = fd/Δ(f), Δ(f) = fs/N**
* **כאשר תדר מסויים יוצא K לא שלם נוכל לראות שמלבד ההלם בK זה שאמור להיות הערך השונה מ0 היחיד, נראה עוד כמה ערכים של K שהם מה שמייצג את הזליגה.**
* **על מנת להראות כי הקוד עובד הדפסנו את גרף המגניטודה של ההתמרה של x[n] , X(K) ובשם ניתן לראות את ערכי K המתאימים לכל אחד מהתדרים, וגם עבור כל תדר האם יש לו זליגה ספקטרלית או לא(הדפסנו בגרף ערכי 100 K-ים).**

### **Part 2 - B – Image**



# **Q5**

### **Part 1 - Code**

% Load the input signal from the file "x.mat"

load x.mat

% Define the coefficients of the given IIR filter

b = [2 -3 -1 0 4 5 -8]; % Numerator coefficients

a = [1 -1.6 1.75 -1.436 0.6814 -0.1134 -0.0648]; % Denominator coefficients

% Filter the input signal with the IIR filter

d = filter(b,a,x);

% Define the order and step size of the FIR filter

m = 50; % Order of the filter

mu = 0.01; % Step size

% Initialize the FIR filter coefficients as zeros

w = zeros(m+1,1);

% Iterating over the input signal:

% Update the FIR filter output, error, and coefficients:

N = length(x);

y = zeros(N,1); % Initiallizng y

e = zeros(N,1); % Initiallizing e

for k = m+1:N

% Input vector of length m+1

u = x(k:-1:k-m);

% Reshape u into a column vector of size m+1 by 1

u = reshape(u,m+1,1);

% Output as product of coefficients and input vector

y(k) = w'\*u;

% Error = difference between wanted output and actual output

e(k) = d(k)-y(k);

% Updating coefficients

w = w + 2\*mu\*e(k)\*u;

end

% Calculate the squared error e^2(k) for each iteration k

e2 = e.^2;

% Plot e^2(k) as a function of k

figure(1)

plot(0:N-1,e2)

xlabel('k')

ylabel('e^2(k)')

title('Squared Error')

grid on

%Plotting the magnitude of the frequency response of the IIR filter and the FIR filter

figure(2)

[h\_iir,w\_iir] = freqz(b,a); % Frequency response of the IIR filter

%for FIR filters denominator coefficient is 1

[h\_fir,w\_fir] = freqz(w,1); % Frequency response of the FIR filter

plot(w\_iir/pi,abs(h\_iir),w\_fir/pi,abs(h\_fir))

xlabel('Frequency')

ylabel('Magnitude')

legend('IIR Filter','FIR Filter')

title('Frequency Response')

grid on

**הסבר:**

* **נתון מסנן IIR - מסנן מסוג** BAND PASS **- שעבורו אנו רוצים למצוא מסנן FIR בעל התנהגות מגניטודה דומה, בעזרת האלגוריתם הנתון ברקע התיאורטי.**
* **התבקשנו לצייר את הגרף של מסנן הIIR הנתון ומסנן הFIR שמצאנו על אותו הגרף.**
* **בגרף מופיעות התנהגויות שונות, ושגיאות שונות עבור K-ים שונים עבור תדרים שונים.**

### **Part 1 – Images**



