**מטלה מסכמת- חלק 1**

# מגישים:

סמי נחמד

נועם רהט

צחי טחן

# הנחיות

* התרגיל יבוצע ביחידים/זוגות/שלשות.**על כל אחד מהסטודנטים להגיש את התרגיל במודל!**
* גם אם לא מצוין, יש לצרף לכל סעיף את קוד המטלב שכתבתם בתוספת הערות (בגוף הקוד ומעבר).
* בתרגיל זה נשתמש במספר תעודות הזהות של הסטודנטים המבצעים להגדרות שונות.  
    
  נסמן:

*d*1=של סטודנט א'סכום הספרות של מספר תעודת זהות

*d*2=של סטודנט ב'סכום הספרות של מספר תעודת זהות

*d*3=של סטודנט ג'סכום הספרות של מספר תעודת זהות

ונגדיר בנוסף

**בתרגיל זה אסור להשתמש בפקודות המטלב: conv, filter, fft2, ifft2, fftn, ifftn, xcorr, conv2, xcorr2, fft2, fftn, fftfilt.**

**בחלק א' אין להשתמש גם בפקודות fft, ifft.**

# חלק א' - מימוש FFT

בחלק זה תכתבו שיגרה המממשתFFT ושיגרה נוספת המממשת IFFT על ידי שימוש בשגרה הראשונה.

יש לממש את השיגרות באחד מ-2 אופנים: א. ללא שימוש ברקורסיה. ב. באופן רקורסיבי.

שימו לב – אופן המימוש תלוי בזוג המבצע את התרגיל.

אם זוגי המימוש צריך להתבצעללא שימוש ברקורסיה.

אם אי זוגי המימוש יתבצע באופן רקורסיבי.

בכדי לבדוק את עבודתכם, השוו לפונקציות FFT ו IFFT של מטלב.צרפו את קוד המטלב שפיתחתם בתוספת הערות (בגוף הקוד).

קוד מטלב:

functionr=bitrev(k,bits)

    % Compute the bit-reversed value of an integer

    % Input:

    %   k - integer to be bit-reversed

    %   bits - number of bits in the binary representation

    % Output:

    %   r - bit-reversed value of k

    r=0;

    fori=0:bits-1

        r=bitor(bitshift(r,1),bitand(bitshift(k,-i),1));

    end

    return;  % Explicit return statement

end

functionx=bitrevorder(x)

    % Reorder the input array according to bit-reversed order

    % Input:

    %   x - input signal (vector of complex numbers)

    % Output:

    %   x - bit-reversed ordered signal

    N=length(x);

    bits=log2(N);

    fori=0:N-1

        rev\_i=bitrev(i,bits);

        ifrev\_i>i

            % Swap elements to achieve bit-reversed order

            temp=x(i+1);

            x(i+1)=x(rev\_i+1);

            x(rev\_i+1)=temp;

        end

    end

    return;  % Explicit return statement

end

functionX=iterativeFFT(x)

    % Iterative FFT implementation using the Cooley-Tukey algorithm

    % Input:

    %   x - input signal (vector of complex numbers)

    % Output:

    %   X - FFT of the input signal

    N=length(x);

    % Check if N is a power of 2

    ifbitand(N,N-1)~=0

        error('Length of input signal must be a power of 2.');

    end

    % Bit-reversed order permutation

    X=bitrevorder(x);

    % Perform the FFT using the iterative approach

    forlen=2:2:N

        halfLen=len/2;

        W=exp(-2i\*pi\*(0:halfLen-1)/len);  % Twiddle factors for current stage

        forstart=1:len:N

            fork=0:halfLen-1

                index1=start+k-1;  % Adjusted for MATLAB's 1-based indexing

                index2=start+k+halfLen-1;  % Adjusted for MATLAB's 1-based indexing

                % Combine the results of smaller transforms

                t=W(k+1)\*X(index2+1);  % Adjusted for MATLAB's 1-based indexing

                X(index2+1)=X(index1+1)-t;  % Adjusted for MATLAB's 1-based indexing

                X(index1+1)=X(index1+1)+t;  % Adjusted for MATLAB's 1-based indexing

            end

        end

    end

end

functionX=inverseFFT(X)

    % Inverse FFT implementation using the iterative FFT function

    % Input:

    %   X - input signal (vector of complex numbers)

    % Output:

    %   x - inverse FFT of the input signal

    N=length(X);

    X=conj(X);  % Conjugate the input signal

    X=iterativeFFT(X);  % Apply the FFT

    X=conj(X);  % Conjugate the result

    X=X/N;  % Normalize by dividing by the length

    return;  % Explicit return statement

end

% Main function to run the test

functioniterativeFFT\_test

    % Test the function

    x=[1,2,3,4];  % Sample input signal

    X\_iterative=iterativeFFT(x);  % Call the iterative FFT function

    % Compare with MATLAB's built-in FFT function

    X\_builtin=fft(x);

    % Display the results

    disp('Iterative FFT:');

    disp(X\_iterative);

    disp('Built-in FFT:');

    disp(X\_builtin);

    % Test the inverse FFT function

    x\_inverse=inverseFFT(X\_iterative);

    % Compare with MATLAB's built-in IFFT function

    x\_builtin\_inverse=ifft(X\_builtin);

    % Display the results

    disp('Inverse FFT (custom):');

    disp(x\_inverse);

    disp('Inverse FFT (builtin):');

    disp(x\_builtin\_inverse);

end

כמובן שתוצאה יצאה נכונה כמצופה

# חלק ב' – סינון דיגיטלי

נתון האות

כמו כן, נתון בקובץ filter\_0.25\_101.mat מסנן ספרתי, המסנן הינו בעל אורך סופי של 102 דגימות.

ניתן לקרב את תגובת התדר של המסנן באופן הבא

כאשר הינו קבוע כלשהו המקיים .

ברצוננו לדגום את בקצב , כאשר לסיגנל הדגום נקרא , ולאחר מכן נסנן את באמצעות .

1. קבעו את תדר הדגימה כך שבמוצא המסנן נקבל אתוננחית את פי A.

נרצה לדגום את האות הרציף r(t) בקצב Fs:כפי שלמדו עבור אות שנראה כמו אם נדגום בקצב Fs האות הדגום -הבדיד,ייראה כמו: (מתיחת הציר פי והכפלה כל ואצלנו:  *כעת, אם נרצה שהאות שיעבור s[n] דרך הסנן יעבור בטווח של ואילו שהאות v[n] יונחת פי A יעבור בטווח של   
נזכור תחילה ש מקיים בתדר ונקבל:*

*ומכיוון שדגמנו את האות-התדר בבדיד הוא הכפלת הציר פי ושכפול כל -סכ"ה נקבל ש:*

*ועל מנת לקבל את הטווחים שרצינו נדרוש: (נשים לב שזה גם יעמוד בתנאי ניקוייסט כי כלומר גם במקרה הכי גרוע של בחירת F לא נחרוג מהתנאי...)*

1. כמה דגימות יש ליטול מ- כדי לקבל במוצא המסנן 2048 דגימות מסוננות?

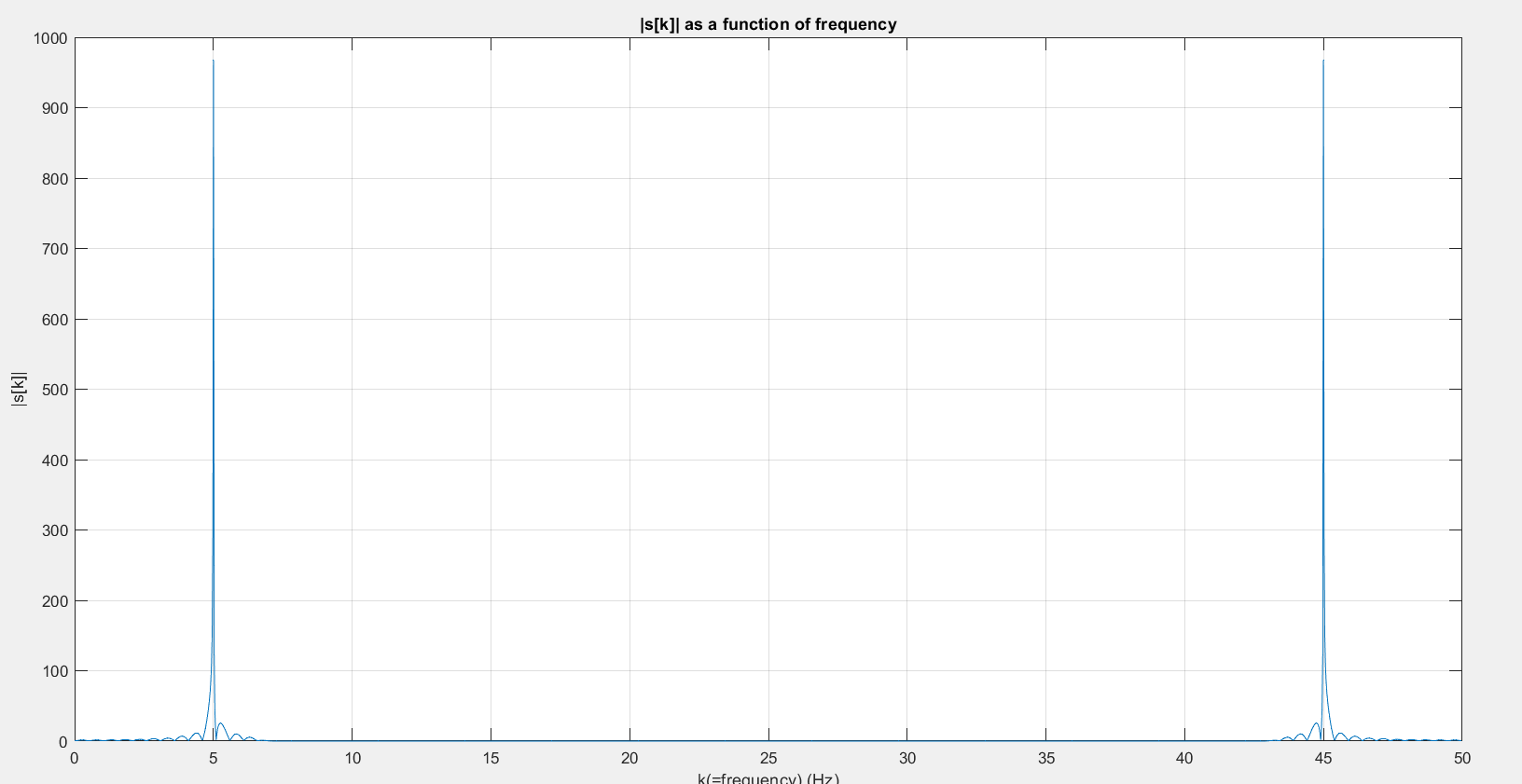
*כפי שלמדנו, בקונבלוציה לינארית בין אות באורך N לאות באורך M סך האיברים במוצא יהיו L=N+M-1 ואצלנו נבצע קונבלוציה בין אות r באורך N לבין המסנן שנתון שאורכו הוא M=102 ולכן אם נרצה שאורך האות במוצא יהיה 2048:*

1. נסמן ב- את מוצא המסנן וב- את התמרת ה-DFT של 2048 הדגימות של . שרטטו בעזרת מטלב את כאשר ציר האיקס הינו ציר תדר אנלוגי בתחום .

*נרצה לשרטט במטלב את התמרת הDTF של מוצא המסנן שלנו-לא מוגדר עבורנו מה הערך של Fs שאיתו נעבוד ולכן נחליט לעבוד עם תדר דגימה של ;   
כפי שהסברנו לעייל וכפי שהראנו בסעיף קודם-באורך של 2048 איברים, לאחר מכן נמצא את s[k] דרך ה DFT של s[n] ולביטוי הזה את הערך המוחלט:*

*נבחר Fs=50 -נצפה שהרכיב v ייסונן והרכיב s יישאר-נצפה לקבל:*

*(מכיוון שזו התמרת DFT נקבל את הדלתאות שלנו ב5,N-5 ולא ב )כלומר נצפה לקבל שתי דלתאות סביב , אבל מכיוון שאנחנו עובדים בציר תדר אנלוגי-הפתרון יהיה סביב כי כפי שלמדנו מתקיים כאשר w הוא התדר בבדיד וf הוא התדר בציר אנלוגי(הרצים) ואנחנו עובדים בהרצים ולכן ונקבל את*

**

*ואכן קיבלנו 2 דלתאות סביב5,45*

*קוד מטלב:*

% Function to perform linear convolution

function [y, num\_mult, num\_add] = linear\_conv(x, h)

    % Linear convolution implementation

    Nx = length(x);

    Nh = length(h);

    Ny = Nx + Nh - 1;

    y = zeros(1, Ny);

    num\_mult = 0;

    num\_add = 0;

    % Perform convolution

    for i = 1:Nx

        for j = 1:Nh

            y(i + j - 1) = y(i + j - 1) + x(i) \* h(j);

            num\_mult = num\_mult + 1;  % Count multiplications

            if j > 1

                num\_add = num\_add + 1;  % Count additions (excluding first in each row)

            end

        end

        %fprintf('Number of multiplications : %d\n', num\_mult);

    end

end

close all;

clc;

clear;

% Load the filter

data = load('filter\_0.25\_101.mat'); % This loads a struct with field names matching the variables in the .mat file

% Correctly access the filter coefficients using the actual field name 'h'

h = data.h; % Use struct field access to get the correct filter variable

Fs=50;

t=(0:1946)/Fs;% in order to creat a 1947 lenght vector

% c/d transform-will mention that we allready included the /Fs in the

% t vector

r\_n=cos(2\*pi\*5\*t)+cos(2\*pi\*10\*t);

bpf\_length=length(h);

s\_n=linear\_conv(r\_n,h);% linear convelution-will result a 2048 output lenght signal

S\_k=abs(fft(s\_n));%s[k] is the dft of s[n]-finding its abs:

f=(0:2047)\*(Fs/2048);% Frequency,after resolutiong like Fs/N

plot(f,S\_k);

xlabel('k(=frequency) (Hz)');

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

title('|s[k]| as a function of frequency');

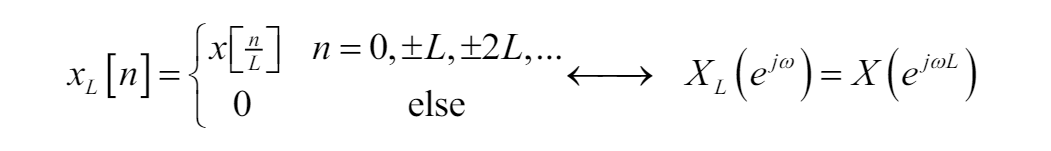
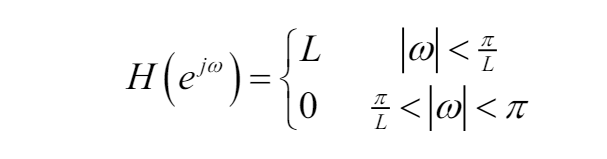
grid on;

1. לסעיף זה בלבד, נתון . האם ניתן להשתמש במסנן הנ"ל לצורך סינון ? אם כן, הסבירו כיצד.

*נתון לנו כי התדר הוא Fs=25 ולכן:   
 ולכן שתי האותות יעברו במסנן דרך המכפלה ב1:ע"מ לסנן את v שוב-נרצה ש:*

*נעשה העלאת קצב:  
נבחר Fs=50 כלומר נרצה שהאות-שנדגם בתדר Fs ייראה כאילו נדגם בתדר כאשר L=2*

*בעצם נרצה להגיע לאות שייראה כמו:*

*זה אומר שנרפד באפס את האות בזמן כך:  
 כתוצאה מכך האות יכווץ פי L בתדר ונקבל שכפולים של האות הבדיד הנ"ל-במקום כל כל ולאחר מכן פשוט נסנן שכפולים מיותרים על ידי:  
*

*לאחר מכן נבצע דמנציה כלומר הורדת קצב כפול 2 על מנת לקבל את האות המקורי בחזרה ונקבל את הדרוש...*

1. נתונה המערכת הבאה

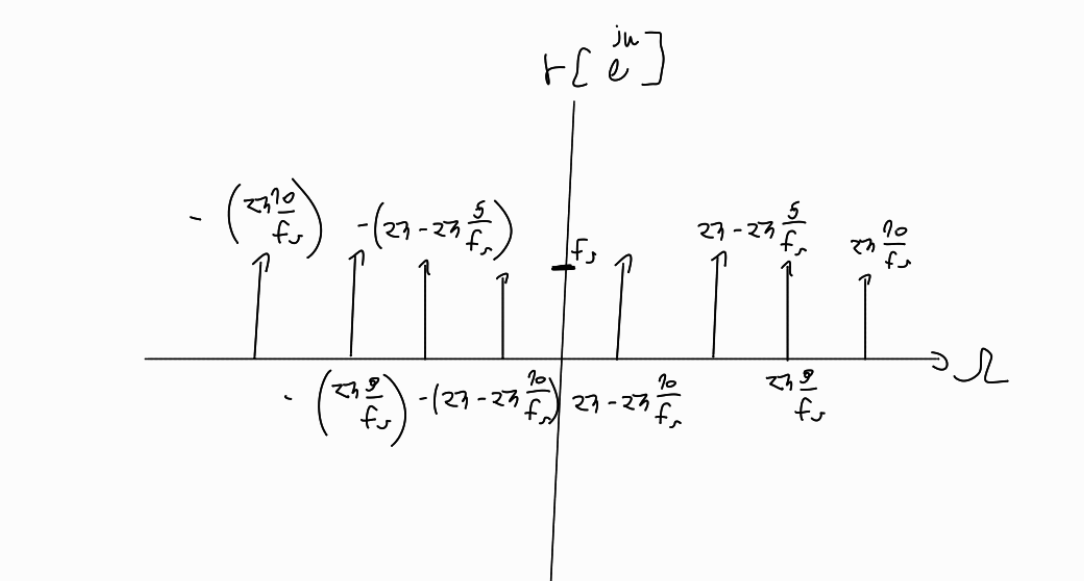
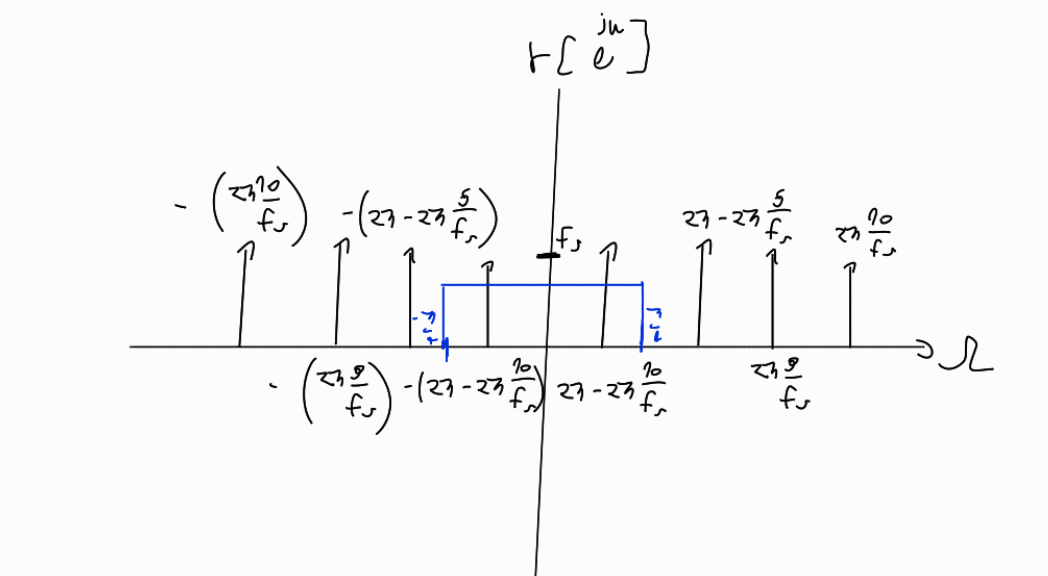


הסבירו בפירוט כיצד ניתן לממש מערכת זו בהינתן המסנן ואות הכניסה הנ"ל? (מה צריך להיות קצב הדגימה ומדוע?)

*נזכור שבנקודה Y נקבל שהאות הבדיד יהיה:*

כעת,במישור התדר-

כאשר לאחר כיווץ הציר והכפלת האיברים פי Fs וביצוע שכפולים כל נקבל:

  
את זה נכפיל במסנן שלנו ונקבל את X  
  
כעת,נזכור שבמוצא נצטרך לקבל  *שבמישור התדר-בשלב הX נצטרך לקבל:  
סכ"ה במישור התדר נדרוש   
נזכור כעת שנוכל להגיע לכך אם נסנן את כל האיברים חוץ מ2 האיברים הכי קטנים-  
  
על מנת להגיע לכך נדרוש שיתקיים:על מנת שהדלתאות האילו יהיו שוות לדלתאות שנרצה-נציב ונקבל בנוסף נשים לב שאכן רק הרכיב הזה ייעבור בטווח המסנן של*

ושהאיברים האחרים לא ייעברו:ואכן נקבל את הדרוש.

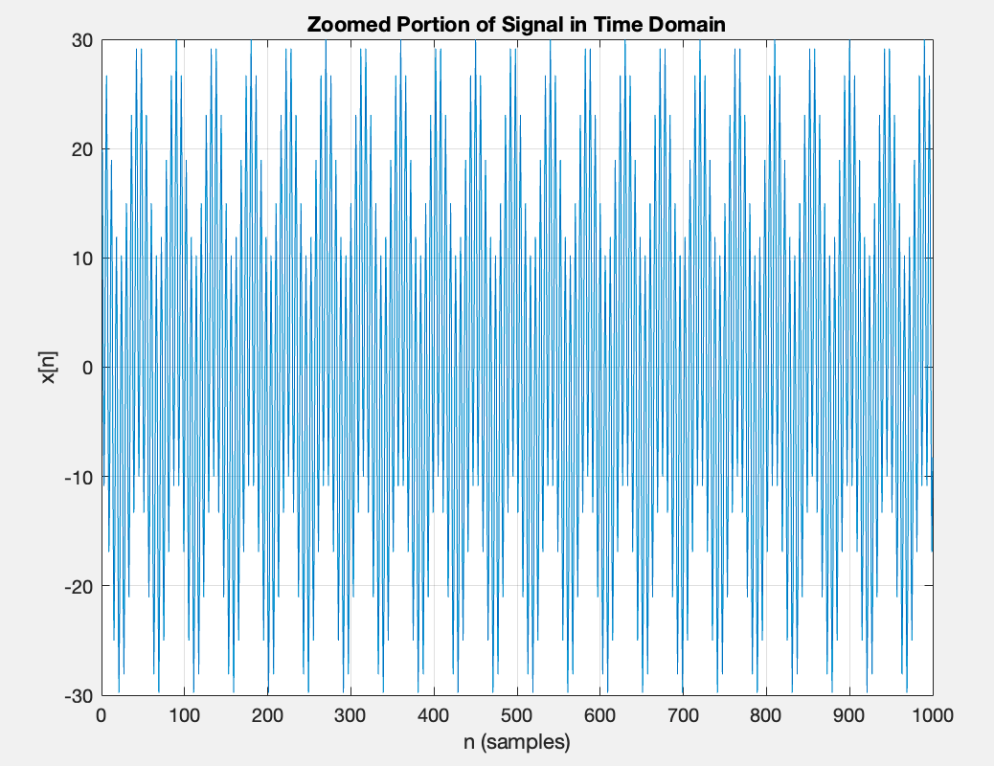
# חלק ג' –OVA

בחלק זה עליכם לממש את שיטת OVA עליה דיברנו בכיתה.

נתון אות ממשי ושני מסננים *. קצב הדגימה הינו 18000 לשנייה. ברצוננו לממש את הקונבולוציה הלינארית:*

*בכדי לבצע את הקונבולוציה הלינארית בצורה יעילה אנו נשתמש בשיטת OVA.*

1. *על מנת לטעון את האות , הורידו את הקובץ sig\_x.mat מאתר הקורס. כיצד נראה הסיגנל? מהם התדרים הפעילים?*

**

* *תדרים פעילים באות (מעל לסף):*

*400 3000*

* *הייצוג המתמטי של האות:*

*קוד מטלב:*

%%Q1:

%%What does the signal "sig\_x.mat" look like? What are the active frequencies?

% Load the signal from the .mat file and define initial parameters

load('sig\_x.mat');  % Loads the variable 'x'

fs=18000;  % Sampling rate in Hz

N=length(x);  % Length of the signal

% Define the time vector for the entire signal and for the zoomed portion

n=0:N-1;  % Time vector for the entire signal

zoomRange=1:min(N,300);  % Zoomed range

% Plot the zoomed portion of the signal in the time domain

figure;

plot(n(zoomRange),x(zoomRange));

title('Zoomed Portion of Signal in Time Domain');

xlabel('n (samples)');

ylabel('x[n]');

gridon;

% Compute the FFT of the signal and define the frequency axis

X=fft(x);

f=(-N/2:N/2-1)\*(fs/N);  % Frequency axis centered around 0

% Compute and plot the magnitude spectrum centered around 0

magnitude=abs(fftshift(X));

figure;

plot(f,magnitude);

title('Magnitude Spectrum of the Signal (Centered)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

gridon;

% Detect active frequencies using a threshold

threshold=max(magnitude)\*0.1;

active\_indices=find(magnitude>threshold);

active\_frequencies=f(active\_indices);

disp(active\_frequencies);

fprintf('Mathematical representation of the signal:\n');

phases=angle(fftshift(X));  % Assign to a variable first

fori=1:length(active\_frequencies)

    fprintf('A%d \* cos(2 \* pi \* %.0f \* n / %.0f + %.2f)\n',i,abs(active\_frequencies(i)),fs,phases(active\_indices(i)));

end

1. *על מנת ליצור את המסננים הורידו את הקבצים filter\_1.mat, filter\_2.mat מאתר הקורס. מהם סוגי המסננים?*

*המסנן הראשון "filter\_1.mat" הוא LPF, ואילו המסנן השני "filter\_2.mat" הוא HPF:A close-up of a graph

Description automatically generated*

*קוד מטלב:*

%%Q2:

%%What are the types of filters filter\_1.mat and filter\_2.mat?

filter\_1=load('filter\_1.mat');

filter\_2=load('filter\_2.mat');

filter\_1\_varname=fieldnames(filter\_1);

filter\_2\_varname=fieldnames(filter\_2);

h1=filter\_1.(filter\_1\_varname{1});% Access the first variable in filter\_1.mat

h2=filter\_2.(filter\_2\_varname{1});% Access the first variable in filter\_2.mat

% Compute and plot the frequency response of each filter

h1\_fft=fft(h1);

h2\_fft=fft(h2);

freq=(0:length(h1\_fft)-1)\*(18000/length(h1\_fft));

figure;

subplot(2,1,1);

plot(freq,abs(h1\_fft));

title('Frequency Response of Filter 1');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

subplot(2,1,2);

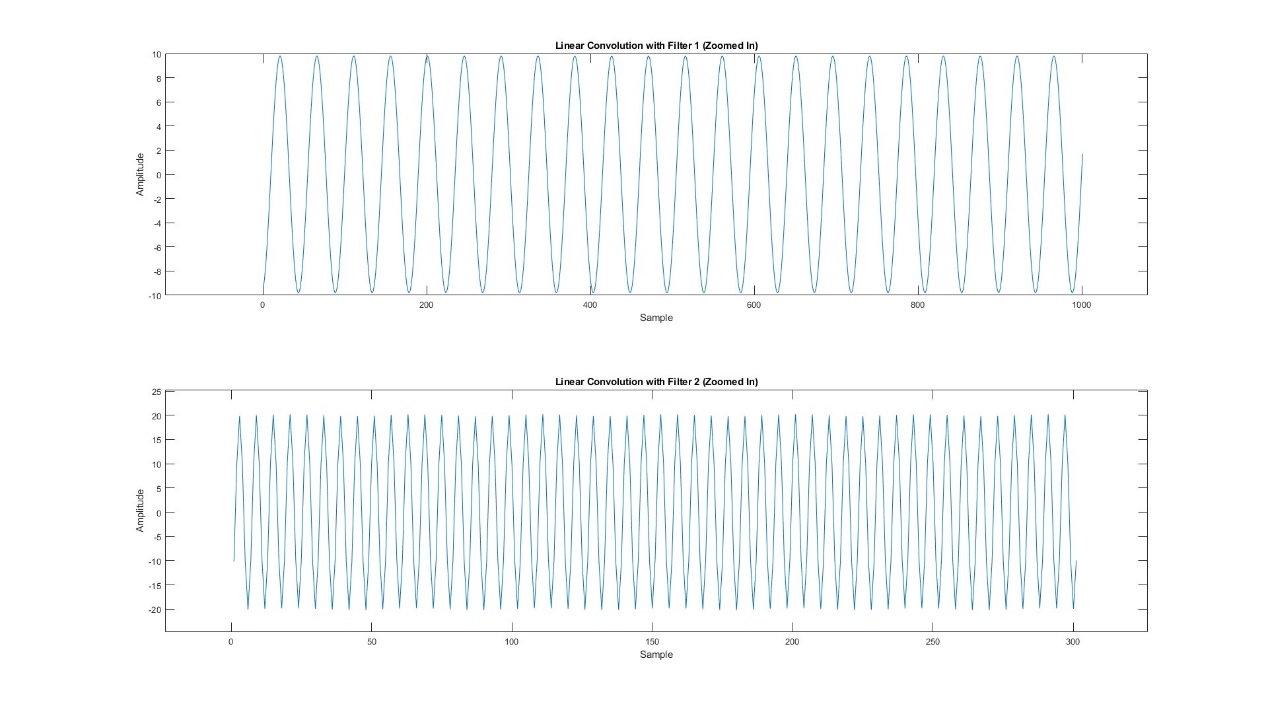
plot(freq,abs(h2\_fft));

title('Frequency Response of Filter 2');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

1. *ממשו באופן ישיר קונבולוציה לינארית בין הסיגנל לכל אחד מהמסננים. השוו בין התוצאות והסבירו אותם. מהו זמן הריצה?*

*תוצאות הקונבולוציה מראות את ההשפעה של כל מסנן על הסיגנלהמסננים מעבירים תדרים שונים ומנחיתים אחרים, כפי שניתן לראות מהגרפים של הקונבולוציה הלינארית: זמני הריצה הקצרים מראים יעילות בביצוע החישוב, עם זמני ריצה כמעט זהים לשני המסננים.*

*Linear convolution with Filter 1 (BSF) took 0.115626 seconds.*

*Number of multiplications with Filter 1: 16470000*

*Number of additions with Filter 1: 16200000*

*Linear convolution with Filter 2 (BPF) took 0.065639 seconds.*

*Number of multiplications with Filter 2: 16470000*

*Number of additions with Filter 2: 16200000*

*קוד מטלב:*

%%Q3:

%%Implement a Directly Linear Convolution between the signal x[n] and each of the filters.

%%Compare the results and explain them. What is the running time?

% Compute linear convolution with each filter and count operations

tic;

[y1\_linear,mult1,add1]=linear\_conv(x,h1);

time\_y1\_linear=toc;

tic;

[y2\_linear,mult2,add2]=linear\_conv(x,h2);

time\_y2\_linear=toc;

% Display results

fprintf('Linear convolution with Filter 1 (LPF) took %.6f seconds.\n',time\_y1\_linear);

fprintf('Number of multiplications with Filter 1: %d\n',mult1);

fprintf('Number of additions with Filter 1: %d\n',add1);

fprintf('Linear convolution with Filter 2 (HPF) took %.6f seconds.\n',time\_y2\_linear);

fprintf('Number of multiplications with Filter 2: %d\n',mult2);

fprintf('Number of additions with Filter 2: %d\n',add2);

% Plot the results of linear convolution

figure;

subplot(2,1,1);

plot(y1\_linear(1200:1500));

title('Linear Convolution with Filter 1 (Zoomed In)');

xlabel('Sample');

ylabel('Amplitude');

subplot(2,1,2);

plot(y2\_linear(1200:1500));

title('Linear Convolution with Filter 2 (Zoomed In)');

xlabel('Sample');

ylabel('Amplitude');

1. *ממשו קונבולוציה לינארית על ידי OVA. הסבירו כיצד קבעתם את פרמטרי האלגוריתם. מהו זמן הריצה האופטימלי? הציגו זאת בגרף כתלות בגודל המסגרת. הסבירו באופן מפורט כיצד עובדת השיטה.*

*Filter1*

*Number of multiplications : 16428032*

*Number of additions: 16158720*

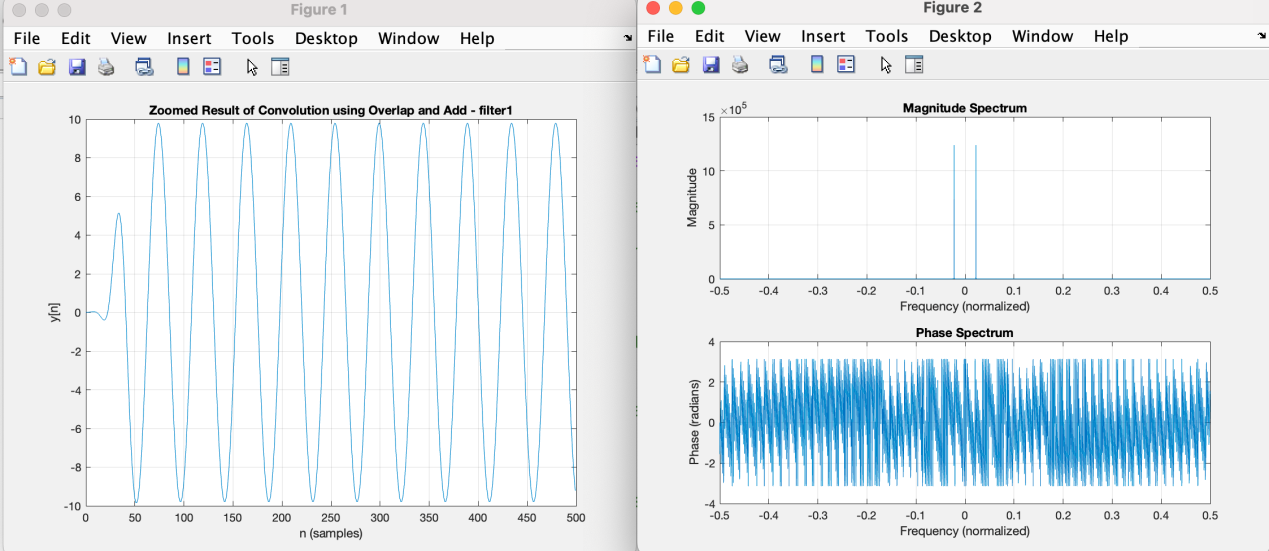
*Linear convolution with Filter 1 took 0.042599 seconds.*

*L = 2048*

*Number of multiplications : 16365568*

*L = 4*

*Number of multiplications : 16469756*

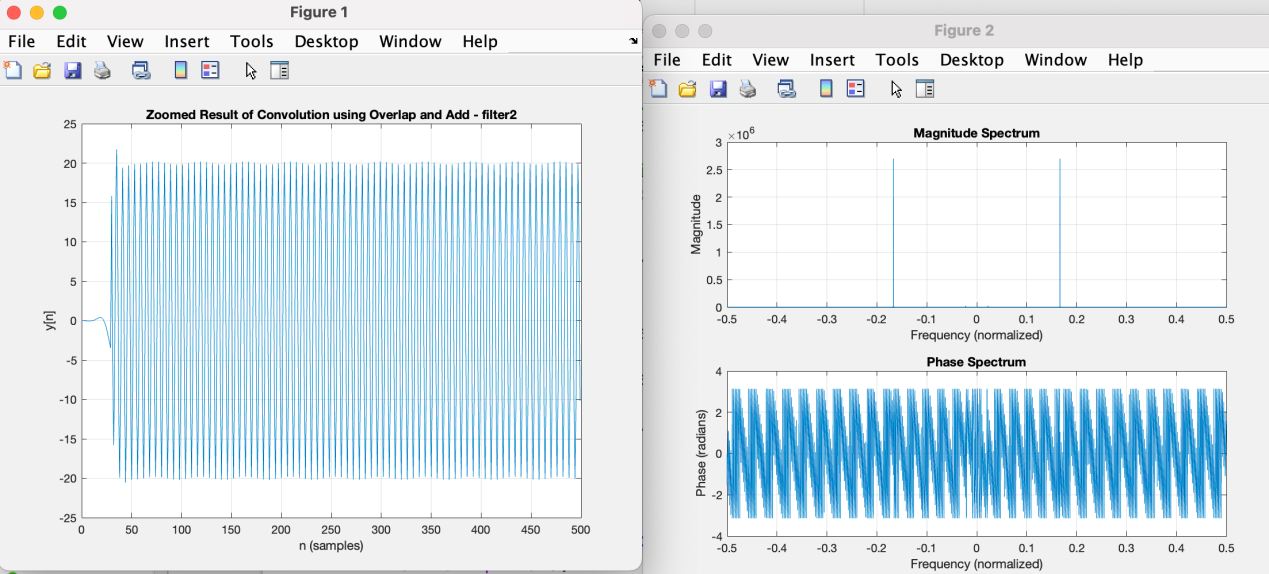
**

*Filter2*

*Number of multiplications : 16428032*

*Number of additions: 16158720*

*Linear convolution with Filter 2 took 0.043691 seconds.*

**

1. *השוו בין זמני הריצה של שתי השיטות על אותו הגרף כפונקציה של גודל המסגרת. לאיזו שיטה ישנה עדיפות מבחינת הביצועים?*

*קוד מטלב:*

%%Q5:

%%Compare the running times of the two methods on the same graph as a function of frame size. Which method has better in terms of performance?

%%Draw on the same graph the output of the 2 types of convolution for each of the filters and show that you performed OVA correctly.

% Linear Convolution using the provided linear\_conv function

[y1\_linear,unus1,unus2]=linear\_conv(x,h1);

[y2\_linear,unus1,unus2]=linear\_conv(x,h2);

% Overlap-Add Convolution using the provided overlap\_and\_add function

L=2048;% Block size

[y1\_ova,unus1,unus2]=overlap\_and\_add(x,h1,L);

[y2\_ova,unus1,unus2]=overlap\_and\_add(x,h2,L);

% Plotting the results

% Zoomed plot for better visibility

zoom\_start=1;

zoom\_end=500;% Adjust as needed

figure;

subplot(2,1,1);

holdon;

plot(zoom\_start:zoom\_end,y1\_linear(zoom\_start:zoom\_end),'r','DisplayName','Linear Convolution','LineWidth',1.5);

plot(zoom\_start:zoom\_end,y1\_ova(zoom\_start:zoom\_end),'b--','DisplayName','OVA Convolution','LineWidth',1.5);

title('Filter 1 (Zoomed)');

legend('show');

xlabel('n (samples)');

ylabel('y[n]');

gridon;

holdoff;

subplot(2,1,2);

holdon;

plot(zoom\_start:zoom\_end,y2\_linear(zoom\_start:zoom\_end),'r','DisplayName','Linear Convolution','LineWidth',1.5);

plot(zoom\_start:zoom\_end,y2\_ova(zoom\_start:zoom\_end),'b--','DisplayName','OVA Convolution','LineWidth',1.5);

title('Filter 2 (Zoomed)');

legend('show');

xlabel('n (samples)');

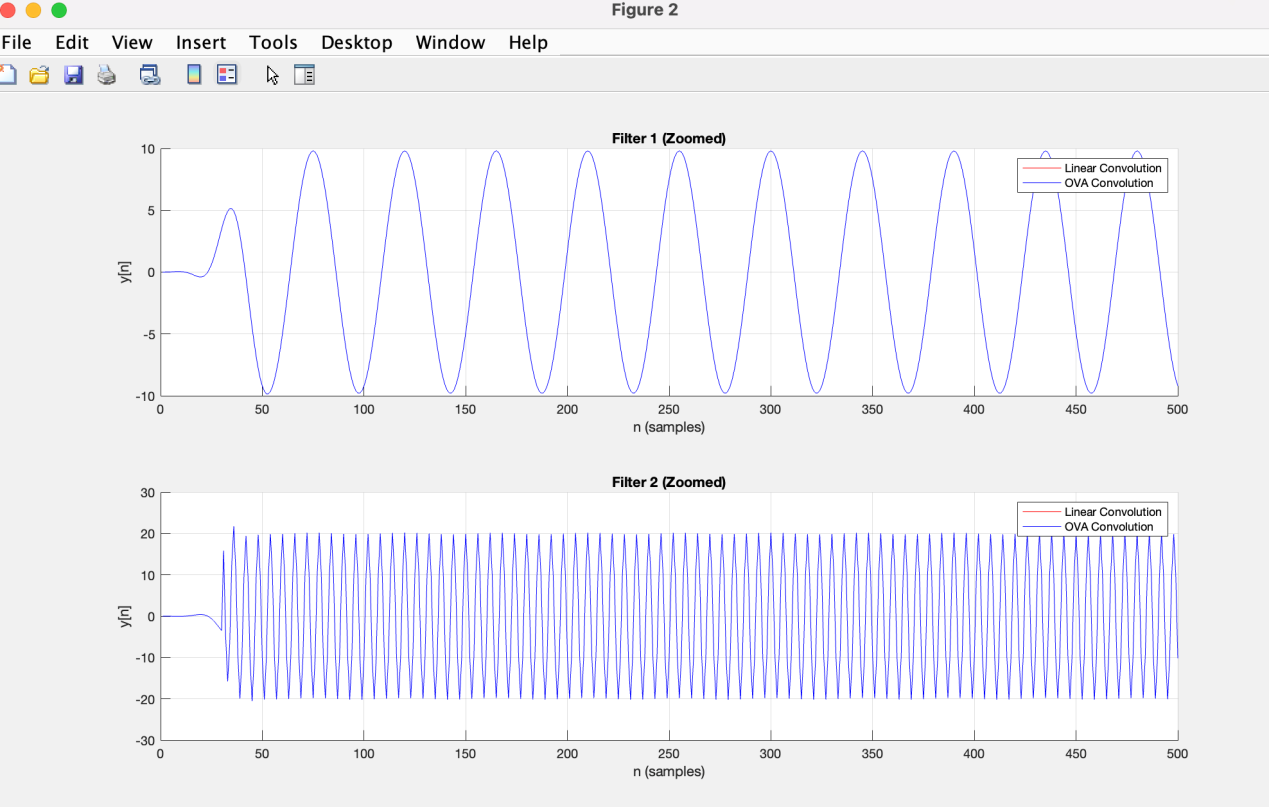
ylabel('y[n]');

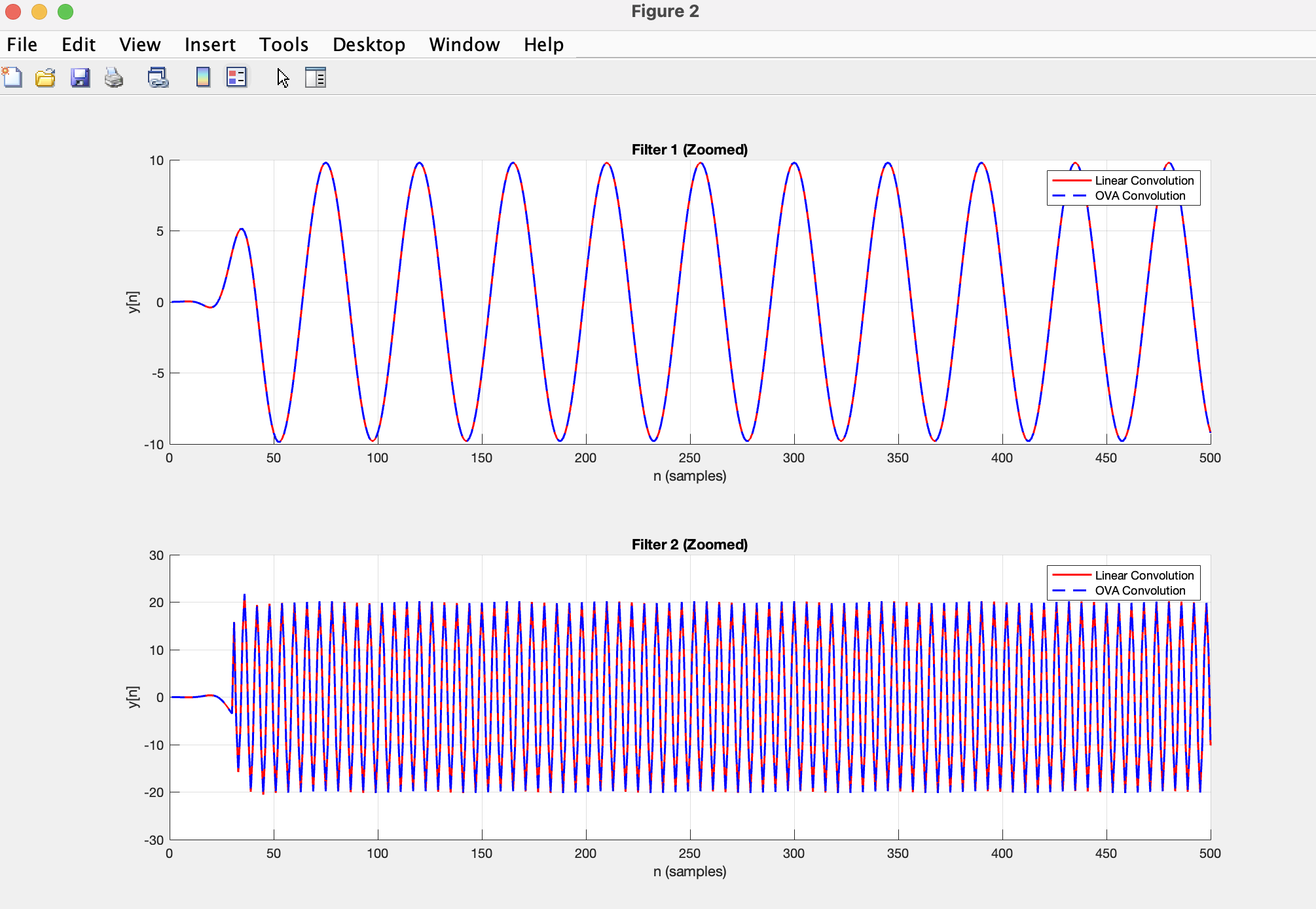
gridon;

holdoff;

*אין שיפור משמעותי בכמות הפעולות בשימוש בשיטה OVA, אך החשיבות של השיטה היא שניתן לחלק קונבולוציה אחד גדולה מאוד (שיכולה אפילו לגלוש מהזיכרון הזמין) למספר קונבולוציות קטנות יותר, ככה שניתן:  
- למקבל את הקונבולוציה (אם יש יותר מליבה אחד)   
-לחשב את הקונבולוציה גם אם לכאורה לא היה מספיק זכרון למחשב*

1. *ציירו על אותו הגרף את המוצא של 2 סוגי הקונבולוציה עבור כל אחד מהמסננים והראו כי ביצעתם OVA כראוי.*

**

**