עבודה מסכמת DSP – חלק 1

<u>מגישים:</u>

יובל פרץ 315053421

דניאל ברוקר 315015594

אור שחר 206582017

חלק א:

$$d_1 = 24\ d_2 = 31\ d_3 = 33\ d_t = 88$$

כלומר d הוא בעל ערך זוגי ולכן נבצע את השגרה ללא שימוש ברקורסיה, נוסיף בדיקות בקוד המשוות את התוצאות בין הקוד שלנו לבין ההתמרה המובנית של מטלב.

את הקוד בנינו בגרסה של radix-2 FFT בעזרת אלגוריתם cooley-tukey. האלגוריתם מותאם לאורכי קלט שהם חזקות 2 בלבד. האלגוריתם מחלק באופן רקורסיבי את רצף הקלט לחלקים של רצפים קטנים יותר לפי אינדקסים זוגיים ואי זוגיים (שזה יותר יעיל כאשר אורך הקלט הוא חזקה של 2).

סיבוכיות האלגוריתם היא O(NlogN) כאשר N זה האורך של וקטור הקלט המקורי.

כאשר הקלט הוא אינו חזקה של 2 האלגוריתם אינו אפקטיבי, לכן צריך לטפל בקלט בעזרת ריפוד הווקטור באפסים עד לאורך של חזקת 2 הקרובה ביותר, ובכך נוכל להשתמש באלגוריתם שלנו. חשוב להדגיש שכאשר מרפדים באפסים רכיבי התדר של הנתונים המקוריים אינם משתנים.

FFT וIFFT הם פעולות שקשורות, ניתן לחשב את הIFFT בעזרת האלגוריתם של FFT ובעזרת שימוש ב"צמוד מרוכב".

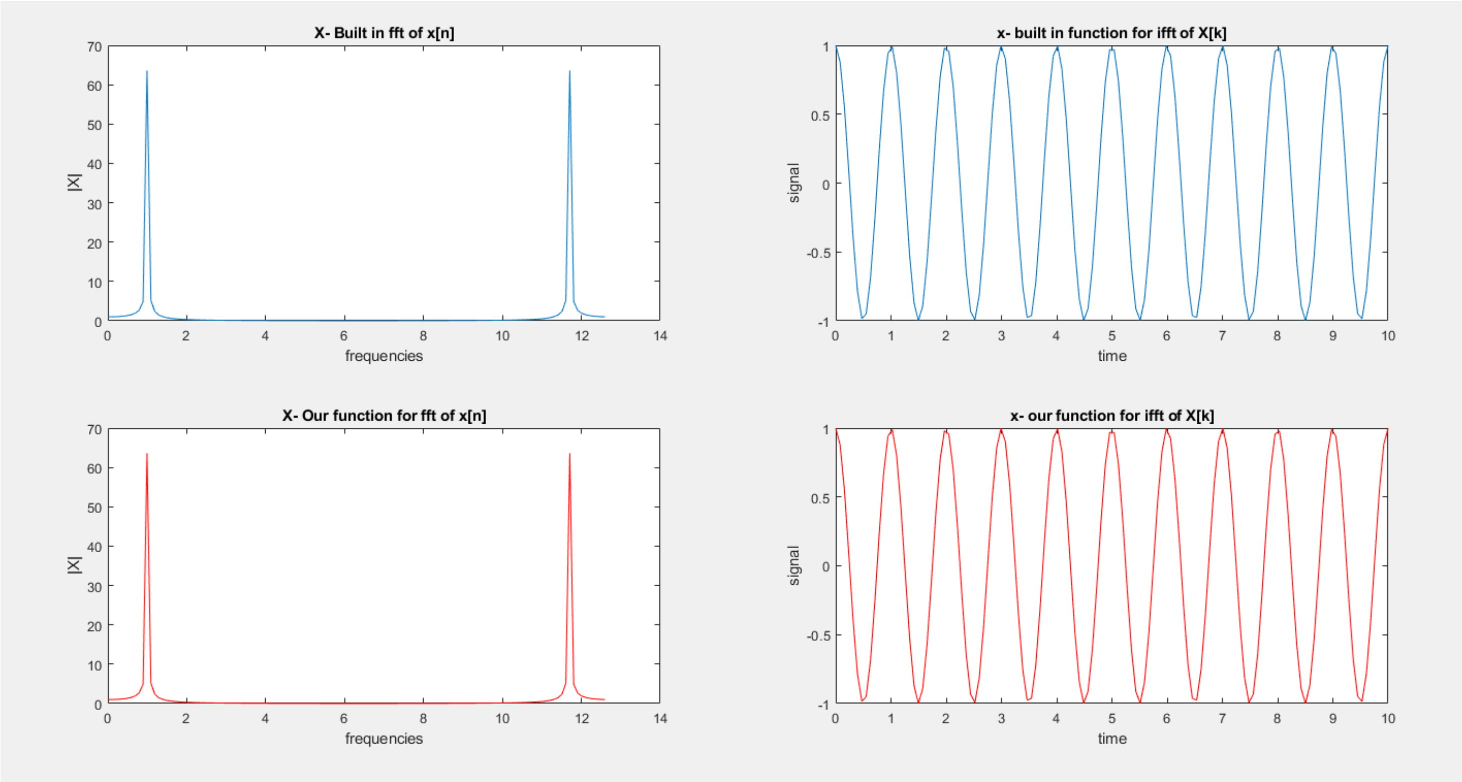
האלגוריתם שלנו בנוי על השלבים הבאים:

1. חישוב FFT עבור x[n] נתון באורך N על ידי הנוסחה הבאה:

$$X[k] = \sum_{0}^{N-1} x[n] e^{-\frac{i2\pi kn}{N}}$$

2. לקיחת הצמוד המרוכב של התוצאה- כלומר את $\overline{X[k]}$.
3. ביצוע FFT על הצמוד המרוכב , והתוצאה תוגדר להיות $\overline{Y[k]}$
4. שוב ניקח את הצמוד המרוכב של התוצאה- כלומר $Y[k]$
5. נחלק את התוצאה בN על מנת לקבל את הIFFT:

$$x[n] = \frac{1}{N} \overline{FFT(\overline{X[K]})}$$



הכנסנו אות של $\cos(2\pi t)$, מצד שמאל של התמונה ניתן לראות שההתמרה נותנת לנו שתי דלתאות כצפוי, ומצד ימין ניתן לראות שהתמרה הפובה נותנת אות קוסינוס כנדרש.

קודים לחלק א:

```
%% Q1- writing FFT and IFFT and comparing to the built in function.

t=linspace(0,10,128);
fs=1/t(2)-t(1);
exmp = cos(2*pi*t);
% Creating the freq axis, as we learned in class we have a change in the
% freq axis in the grapic represantaion, so we create it.
freq=(0:length(exmp)-1)*fs/length(exmp);
% Perform FFT using our function
X_FFT = custom_fft(exmp);
figure;
subplot(2,2,1)
% Display the result
% Perform FFT using built-in function
plot(freq, abs(fft(exmp)))
title('X- Built in fft of x[n]')
xlabel('frequencies')
ylabel('|X|')
subplot(2,2,3)
plot(freq,abs(X_FFT),'r')
title('X- Our function for fft of x[n]')
xlabel('frequencies')
ylabel('|X|')
% Perform IFFT using our function
x_new = custom_ifft(X_FFT);
subplot(2,2,2)
% Perform IFFT using built-in function
plot(t,ifft(X_FFT))
title('x- built in function for ifft of X[k]')
xlabel('time')
ylabel('signal')
subplot(2,2,4)
plot(t,x_new,'r')
title('x- our function for ifft of X[k]')
xlabel('time')
ylabel('signal')
%done


function x = custom_ifft(X)
    % Ensure X is a column vector
    X = X(:);
    % Get the number of points
    N = length(X);
    % Check if N is a power of 2, for the optimization of the Cooley-Tukey
    % FFT algorithm.
    if mod(log2(N), 1) ~= 0
        n = nextpow2(N);
        closest_pow_of_2 = 2^n;
        X = [X; zeros( closest_pow_of_2 -N,1)];
    end
    N = length(X);
    % Conjugate the input
    X = conj(X);
    % Perform the FFT on the conjugated input
    X = custom_fft(X);
    % Conjugate the result and scale by 1/N, the full mathematical
```

```
    % explanation for why it works is in the PDF.
    x = conj(X) / N;
end

function X = custom_fft(x)
    % Ensure x is a column vector
    x = x(:);
    % Get the number of points
    n=length(x);
    N = length(x);
    % Check if N is a power of 2
    if mod(log2(N), 1) ~= 0
       n_upper = nextpow2(N);
        closest_power_of_2 = 2^n_upper;
        x = [x; zeros( closest_power_of_2 -N,1)];
    end
    N = length(x);
    % Bit-reversal permutation
    n = 0:N-1;
    j = bitrevorder(n);
    % Reorder the input array
    x=x(j+1);
    % Initialize the FFT output
    X=x;
    % Iterative Cooley-Tukey FFT
    for s=1:log2(N)
        m=2^s;
        m2=m/2;
        w = exp(-2i * pi / m);
        for k = 0:m:(N-1)
            for j = 0:(m2-1)
                t = w^j*X(k+j+m2+1);
                u = X(k+j+1);
                X(k+j+1)= u+t;
                X(k+j+m2+1) = u - t;
            end
        end
    end
end
```

<u>חלק ב:</u>

נתון האות

$$r(t)=\underbrace{\cos(2\pi 5t)}_{s(t)}+\underbrace{\cos(2\pi 10t)}_{v(t)}=s(t)+v(t)$$

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

$$H\left(e^{j\omega}\right)=\begin{cases}1 & |\omega|\le \pi/4\\ A & \pi/4\le|\omega|\le\pi\end{cases}$$

כאשר A הינו קבוע כלשהו המקיים $A \ll 1$.

ברצוננו לדגום את $r(t)$ בקצב F_s, כאשר לסיגנל הדגום נקרא $r[n]$, ולאחר מכן נסנן את $r[n]$ באמצעות $H\left(e^{j\omega}\right)$.

א. קבעו את תדר הדגימה כך שבמוצא המסנן נקבל את $s[n]$ וננחית את $v[n]$ פי A.

ניזכר במשפט הדגימה שלמדנו בהרצאות ובתרגולים ונבין בדיוק מה אנחנו רוצים לבצע:

$$X\left(e^{jw}\right)=\frac{1}{T}*\sum_{k=-\infty}^{\infty}X_c\left(\frac{j(w-2\pi k)}{T}\right)$$

כלומר אנחנו מקבלים במישור התדר שכפולים כל 2π כאשר נרמלנו את ציר התדר פי תדר הדגימה שהוא כמובן T.

נסתכל על הדרישות של המסנן ונתאים אותם לדרישות של הסעיף:

עבור $s(t)$ שנרצה להשאיר אותו זהה למקור, נדרוש: $|w|\le\frac{\pi}{4}$ כיוון שבתחום הזה המסנן אינו מבצע כלום ומעביר את האות כמו שהוא ונקבל:

$$2\pi 5T\le\frac{\pi}{4}$$

$$T<\frac{1}{40}\rightarrow F_s=40\ [Hz]$$

עבור *v(t)* נרצה להנחית אותו פי *A* נדרוש *π/4≤|w|≤π* כיוון שבתחום הזה המסנן מנחית את האות פי *A*, כמובן כי נתון כי *A≪1* ולכן נקבל:

$$\frac{\pi}{4}\le 2\pi 10T\le\pi$$

$$\frac{1}{80}\le T\le\frac{1}{20}\rightarrow 20\le F_s\le 80$$

ובמצטבר נקבל: $40 \le F_s \le 80$.

ב. כמה דגימות יש ליטול מ- $r(t)$ כדי לקבל במוצא המסנן 2048 דגימות מסוננות?

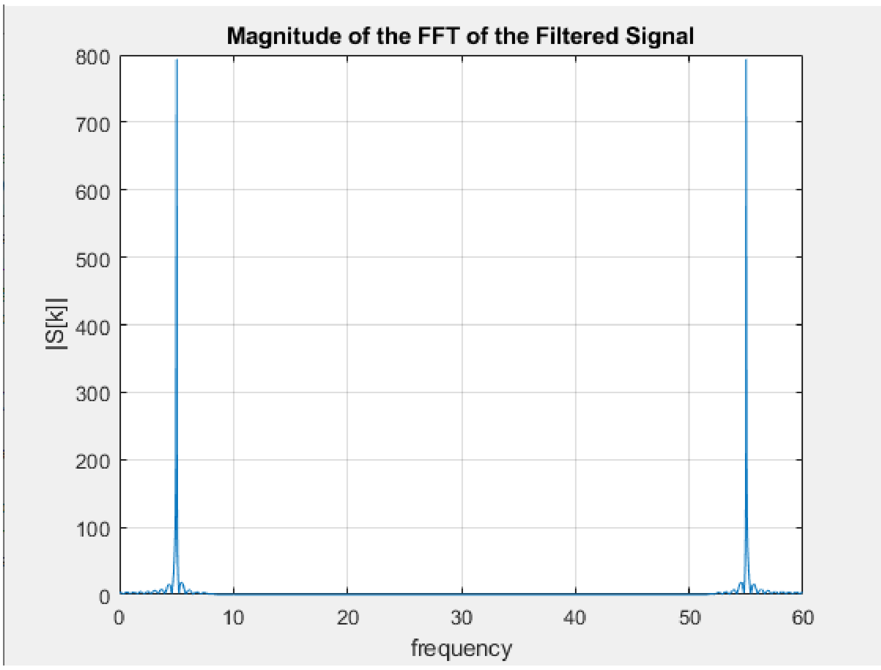
אורך המסנן שלנו הוא 102. כאשר מעבירים את האות *r[n]* במסנן בעצם מתבצעת קונבלוציה – *r[n]*h[n]*, כאשר *h[n]* הוא המסנן ו*r[n]* הוא אות הכניסה. ולכן, לפי תכונות של קונבלוציה שלמדנו בקורס, אורך הוקטור שיתקבל במוצא הוא *n+m-1*.

כאשר *n=102* ו*m* זה הנעלם שלנו (מספר הדגימות של אות הכניסה), והתוצאה תיהיה 2048 כי זה מספר הדגימות במוצא המסנן. נחשב את *m*:

$$n + m - 1 = 2048$$
$$102 + m - 1 = 2048$$
$$m = 1947$$

ג. נסמן ב- $S[n]$ את מוצא המסנן וב- $S[k]$ את התמרת ה-DFT של 2048 הדגימות של $S[n]$.

שרטטו בעזרת מטלב את $|S[k]|$ כאשר ציר האיקס הינו ציר תדר אנלוגי בתחום $[0, F_s]$.



לפי טבלאות ההתמרה אנחנו יודעת שהתמרה של קוסינוס אנחנו צריכים לקבל שתי דלתאות בשתי נקודות מסוימות. (במקרה שלנו האות לא מרוכז סביב האפס ולכן זה לא סימטרי). בתרגול למדנו שבגלל שהמסנן הוא חלון סופי נקבל שני גרעיני דיריכלה במקום שתי דלתאות. וזה מה שאנחנו רואים בגרף. ניתן לראות שהפיקים נמצאים סביב f=5,55 כיוון שרצינו להעביר את $s(t) = \cos(2\pi 5t)$.

ד. לסעיף זה בלבד, נתון $F_s = 25\ \mathrm{Hz}$. האם ניתן להשתמש במסנן הנ"ל לצורך סינון $v[n]$? אם בן, הסבירו כיצד.

לאחר דגימה של אות הכניסה r[n], נקבל:

V[n] ישוכפל בקפיצות של:

$$\frac{2\pi 10}{F_s} - 2\pi k = \frac{2\pi 10}{25} - 2\pi k = \frac{4}{5}\pi - 2\pi k$$

S[n] ישוכפל בקפיצות של :

$$\frac{2\pi 5}{F_s} - 2\pi k = \frac{2\pi 5}{25} - 2\pi k = \frac{2}{5}\pi - 2\pi k$$

על מנת לקיים שנוכל להשתמש ב $F_S = 25$ ולהשתמש במן הנתון ולסנן את *v[n]*, נשתמש באינטרפולציה ודצימציה.

נדרוש את התנאים הבאים:

עובר מיקום *v[n]*:

$$\pm\frac{4\pi}{5} * \frac{A}{B}$$

ועבור *s[n]*:

$$\pm\frac{2\pi}{5} * \frac{A}{B}$$

נמצא את *A,B* שמתקיימים את הדרוש כדי לעבור במסנן, כלומר שהמיקומים קטנים מ:

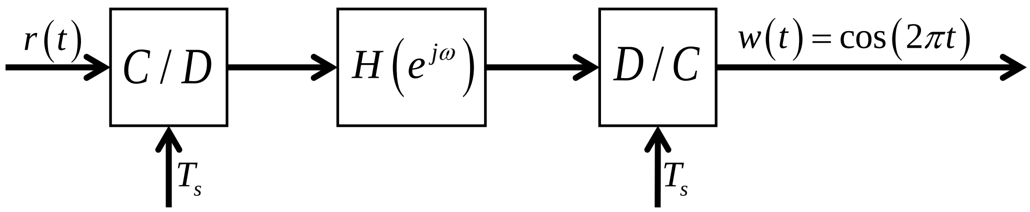
$$\frac{\pi}{4}$$

$$\begin{cases} \frac{2\pi}{5} * \frac{A}{B} < \frac{\pi}{4} \\ \frac{\pi}{4} \le \frac{4\pi}{5} * \frac{A}{B} \le \pi \end{cases} \rightarrow \begin{cases} \frac{A}{B} < \frac{5}{8} \\ \frac{5}{16} \le \frac{A}{B} \le \frac{5}{4} \end{cases} \rightarrow \frac{5}{16} \le \frac{A}{B} \le \frac{5}{8}$$

נבחר *A=1, B=2* כדי להיות בתחום הדרוש.

כלומר, ניתנת האפשרות לקיים סינון של *v[n]* בעזרת המסנן הנתון גם עם קצב דגימה של $F_S =$ 25.

ה. נתונה המערכת הבאה



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

מה שהמערכת דורשת זה שהמוצא יהיה $w(t) = \cos(2\pi t)$ כלומר-בציר התדר ההלם צריך להיות ב*f=1[Hz]* .
לכן- נדרש לשנות את המיקום של ההלם ל$2\pi T$ לאחר הדגימה.
נדרוש עבור *v(t)*:

$$2\pi 10T - 2\pi k = 2\pi T \rightarrow T = \frac{k}{9}$$

נבדוק איזה ערכים של *k* יעברו במסנן:

$$\frac{2\pi 10k}{9} - 2\pi k < \frac{\pi}{4} \rightarrow k < \frac{9}{8}$$

ולכן הערך היחיד שמאים עם שתי הדרישות הוא *k=1*:

כלומר נקבל קצב דגימה:

$$F_S = \frac{1}{T_S} = 9[Hz]$$

קודים לחלק ב:

```
%%Q2-C
fs = 60; % Sampling frequency in Hz, which fits the critiria.
N_in = 1947; % Number of input samples, we calculated the needed size.
N_out = 2048; % Number of output samples
t = (0:N_in-1)*(1/fs);% Creating the time values
t2=linspace(0,32.45,1947);
r = cos(2*pi*5*t)+cos(2*pi*10*t); % Creating the signal r(t)
load('filter_0.25_101.mat','h'); % Load the filter
% The FFT on r will give us a signal with 2048 samples which is the size
% requested for the output, we than want to mult it by the FFT of h, in
order
% to do that we need the leangth of the filter and the signal to be the
same,
% so we will pad h.
h_padded = [h, zeros(1, N_out - length(h))];
R=custom_fft(r);
H_Pad=custom_fft(h_padded);
S=R.*H_Pad;
% Frequency vector for the x-axis, as we learned in class we have a change
in the
% freq axis in the grapic represantaion, so we create it.
f = (0:N_out-1) * (fs / N_out);
figure;
plot(f, abs(S));
xlabel('frequency');
ylabel('|S[k]|');
title('Magnitude of the FFT of the Filtered Signal');
grid on;
%done
```

<u>חלק ג:</u>

בחלק זה עליכם לממש את שיטת OVA עליה דיברנו בכיתה.

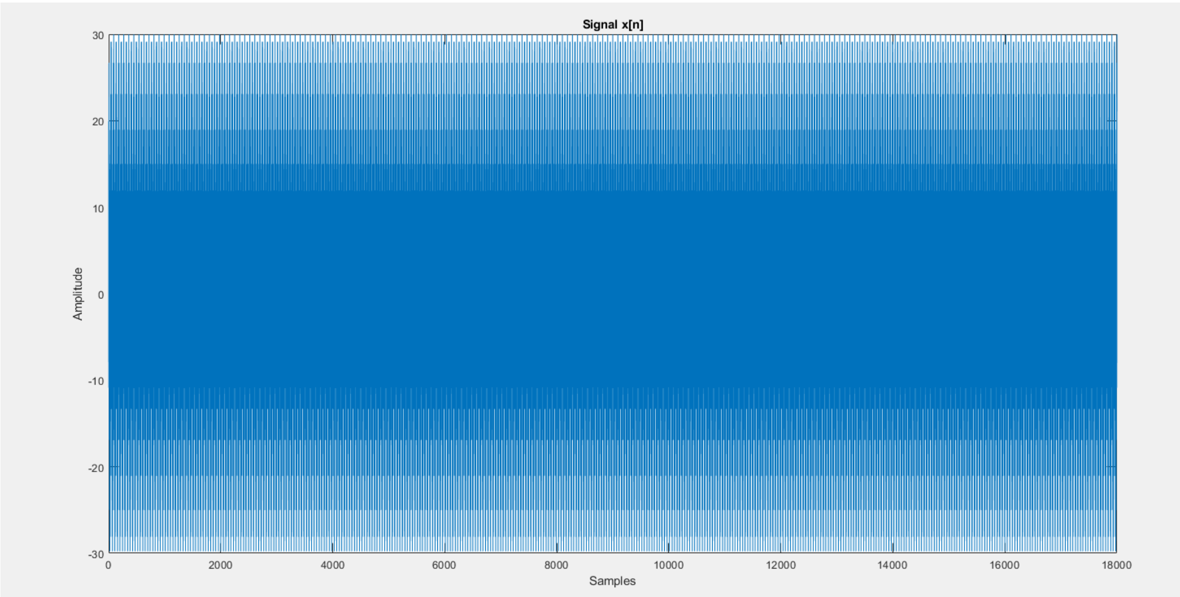
נתון אות ממשי $\{x[n] \in \mathbb{R}: n = 0, \dots, N\}$ ושני מסננים $h_1[n], h_2[n]$. קצב הדגימה הינו 18000 לשנייה. ברצוננו לממש את הקונבולוציה הלינארית:

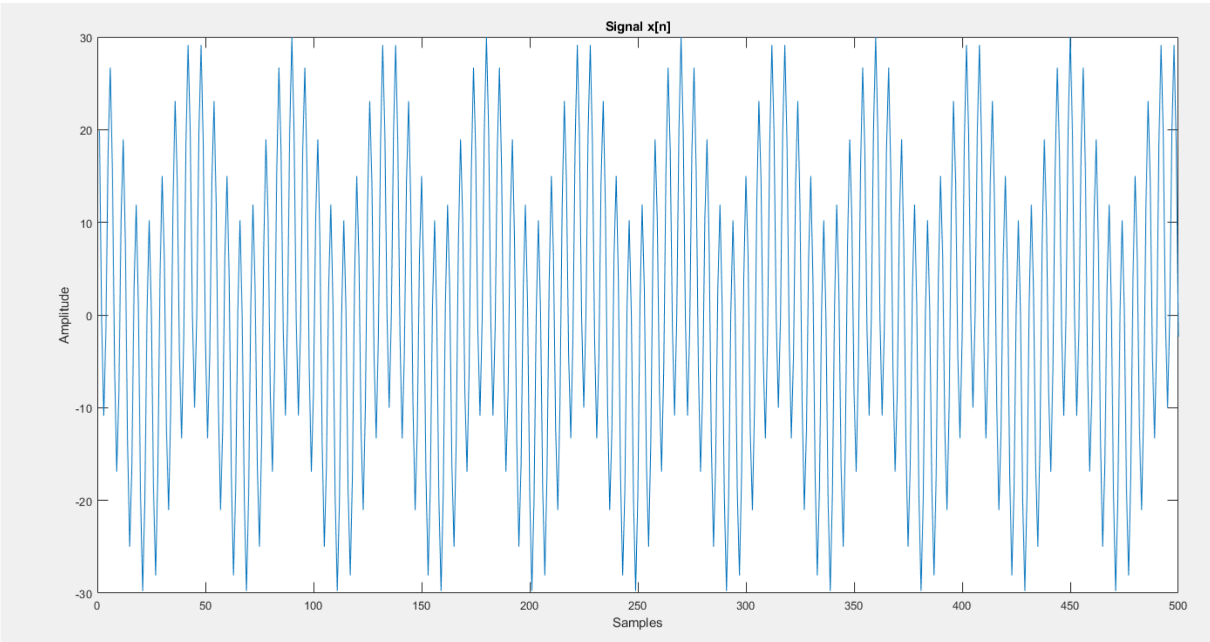
$$y[n] = x[n] * h[n]$$

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

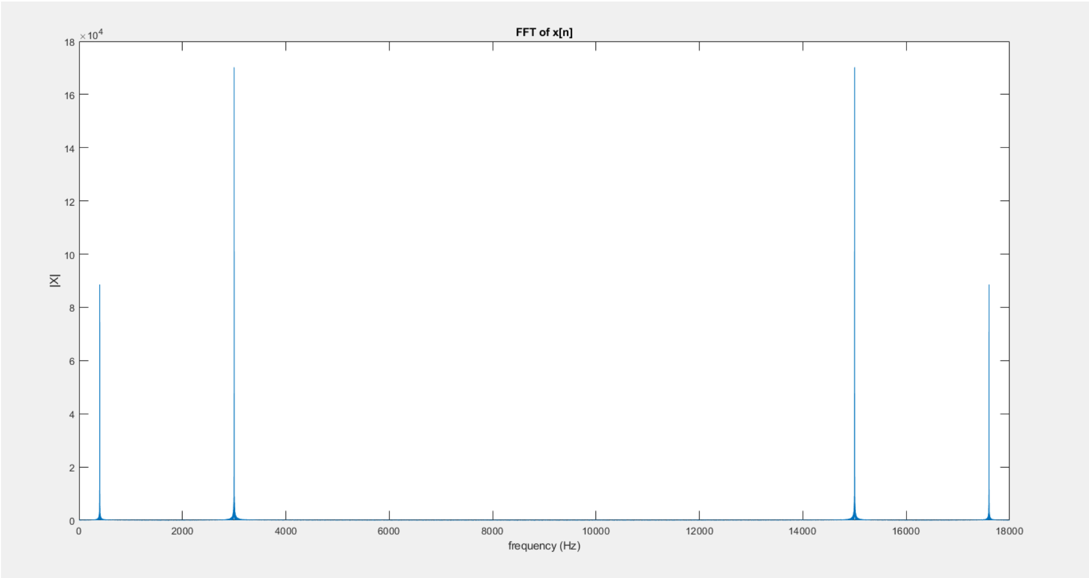
א. על מנת לטעון את האות $x[n]$, הורידו את הקובץ *sig_x.mat* מאתר הקורס. כיצד נראה הסיגנל? מהם התדרים הפעילים?

טענו את הקבצים, נריץ *plot* <u>בכדי לראות כיצד נראה האות</u>:



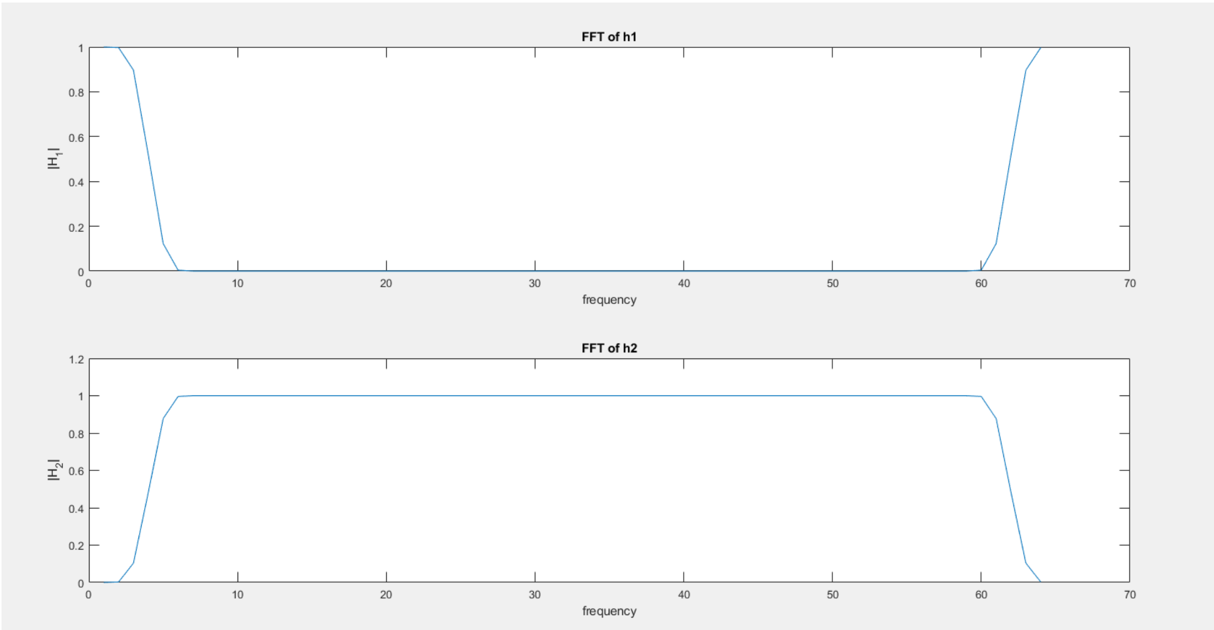


<u>והתדרים הפעילים הם:</u>



ב. על מנת ליצור את המסננים הורידו את הקבצים *filter_1.mat, filter_2.mat* מאתר הקורס. מהם סוגי המסננים?

<u>נדפיס בגרף את המסננים ונקבל:</u>

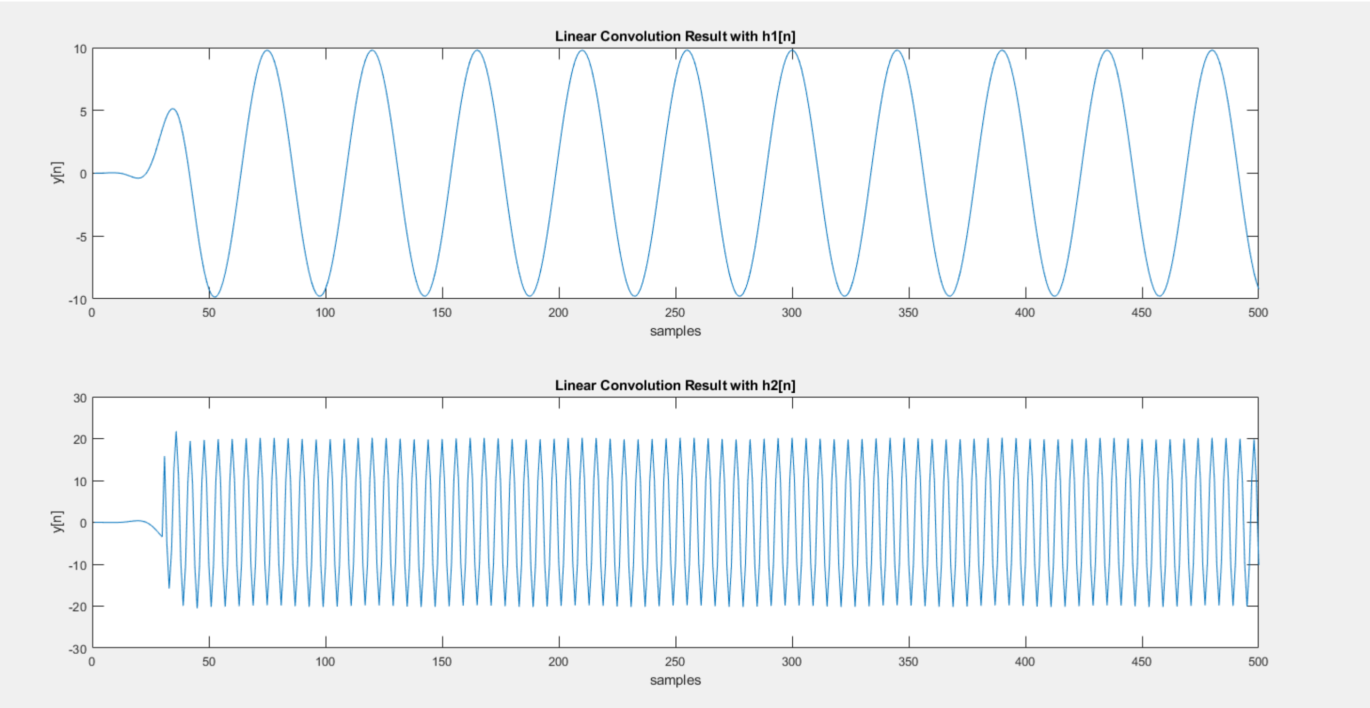


לפי הגרפים ניתן לראות ש*h1* הוא מסנן מסוג: *Band stop filter* .

ו*h2* הוא מסנן מסוג:*Band pass filter* .

ג. ממשו באופן ישיר קונבולוציה לינארית בין לסיגנל $x[n]$ לכל אחד מהמסננים. השוו בין התוצאות והסבירו אותם. מהו זמן הריצה?

<u>תוצאות הקונבלוציה:</u>



נשים לב שמסנן $h1$ (BSF) הוא בעל קוסינוס רחב יותר שזה הגיוני כי יש תחום רחב יחסית של תדרים שנחסם והוא מעביר רק את התדרים שמחוץ לחסימה(במישור התדר). במישור הזמן נקבל שהמסנן דורש זמן תגובה קצר יותר כיוון שמעביר פחות תדרים (בכמות).

מצד שני מסנן $h2$ (BPF) הוא בעל קוסינוס צפוף יותר, כך שהוא מעביר תחום רחב יותר של תדרים ולכן נקבל גרף צפוף ומלא יותר. במישור הזמן נקבל זמן תגובה ארוך יותר כיוון שיש כמות תדרים גדולה יותר שצריכה לעבור.

זמן הריצה עבור שיטת קונבלוציה לינארית הוא $O(N^2)$. אנחנו יודעים כי על כל ערך במסנן נדרש לעשות סכימה לכל ערך של *x*. כלומר יש לנו מעבר כפול על המערכים בסכימתם ולכן נקבל *N*N* ולכן זו הסיבוכיות שלנו.

ד. 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.

קביעת הפרמטרים מתחשבת באורך המסנן *N*– אצלנו זה וקטור שורה באורך 61.

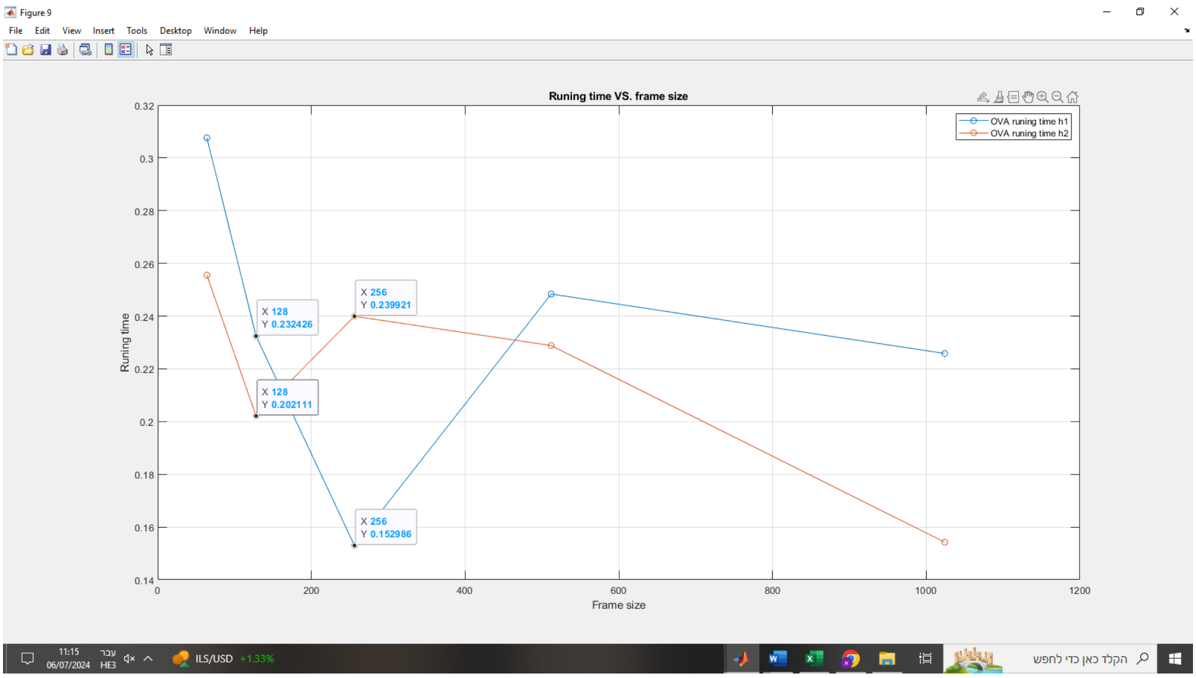
עבור *L* -משתנה עזר לחישוב המסגרת האידיאלית. הוא מוגדר להיות חזקת 2 הכי קרובה לאורך המסנן. אצלנו *L=64*.

אורך ה*FFT* (גודל המסגרת) מוגדר להיות *L+N-1* במקרה שלנו רצינו לבדוק אופציות נוספות לאורך ה*FFT* כדי לחפש את גודל המסגרת האופטימלי, ולכן ניתן כמה חזקות 2 שונות ועבורן נחשב את הקונבלוציה.

זמן הריצה של השיטה הוא $O(NlogN)$

שיטת *Overlp-Add* היא שיטה לשיפור התוצאות שאפשר לקבל מפעולת קונבולציה.
השיטה מתבוססת על עיקרון שהוא פיצול האות הכללי לחלקים קטנים, ביצוע קונבלוציה בנפרד על כל חלק וחיבור התוצאות של החלקים לתוצאה סופית.
שלב ראשון בשיטה זה לחלק את האות *x[n]* לחלקים שאורכם *L*, נשאף לחפיפה מינימלית בין החלקים, ולכן נשתמש בחפיפה נוספת באורך *M-1* כאשר *M* זה אורך המסנן *h[n]*.
שלב שני-לאחר מכן נוסיף לכל חלק אפסים בסוף כך שהאורך החדש יהיה *L+M-1*. שלב שלישי- ביצוע *FFT* לכל חלק ולמסנן *h[n]*. ביצוע כפל במישור התדר בין ה*FFT* של החלק ל*FFT*של המסנן. לאחר מכן מבצעים *IFFT* על התוצאה כדי לחזור למישור הזמן.
שלב רביעי-חיבור כל התוצאות של החלקים כך שכל חלק חוזר למיקומו המקורי ויכלול גם את החפיפות המתאימות מהחלקים השכנים אליו.
לשיטה יש מספר יתרונות- מבחינת סיבוכיות, שיטת *OVA* מצמצת את הסיבוכיות מ$O(N^2)$ ל $O(NlogN)$. (יעיל בעיקר עבור אותות ארוכים). בנוסף ניתן לטפל במסננים בעלי אורך גדול יותר בצורה יותר טובה.

<u>גרף של זמן ריצה כתלות בגודל המסגרת:</u>

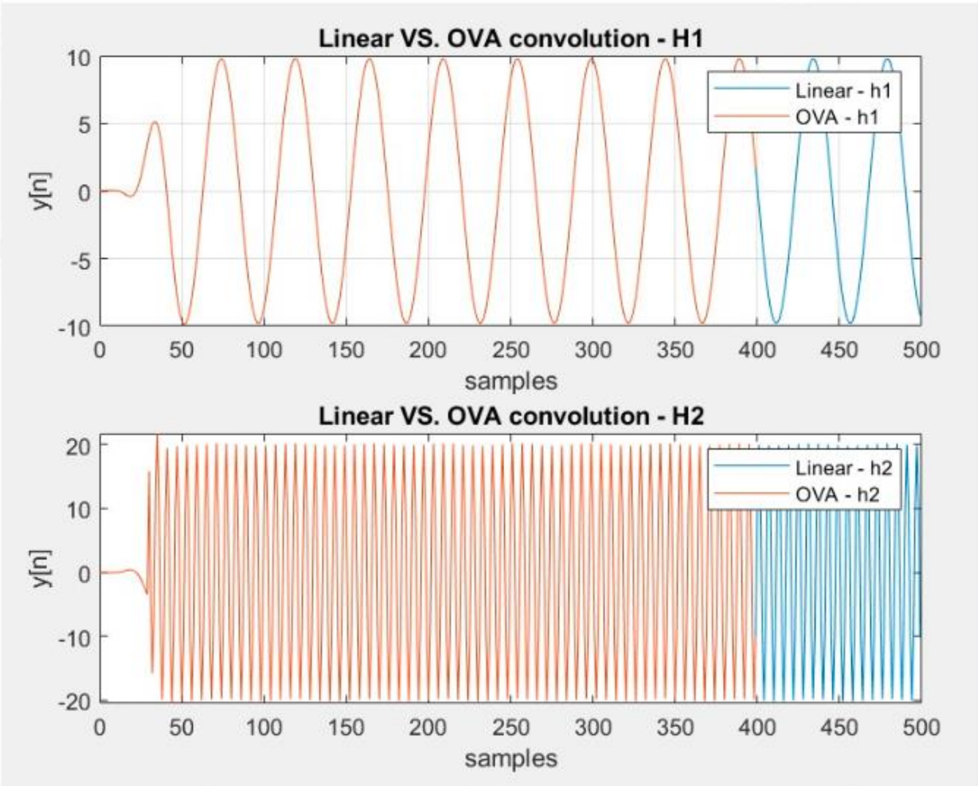


ה. השוו בין זמני הריצה של שתי השיטות על אותו הגרף כפונקציה של גודל המסגרת. לאיזו שיטה ישנה עדיפות מבחינת הביצועים?

ראינו שזמן הריצה של קונבלוציה לינארית הוא $O(N^2)$ ושעבור שיטת *OVA* $O(NlogN)$

כלומר השיטה העדיפה מבחינת סיבוכיות היא *OVA*.

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



לקחנו מספר דגימות שונה עבור כל אחת מהקונבלוציות על מנת שנוכל לראות את השיוון בין שתי השיטות. כאשר הגרפים האדומים מתארים את שיטת *OVA* והכחולים את הקונבלוציה הלינארית.

קודים לחלק ג:

```
%% Q3-A
% Loading the signal and filters from the files given
sig = load('sig_x.mat');
x_FULL = sig.x;
x=x_FULL(1:18000);
filter1 = load('filter_1.mat');
h1 = filter1.xx;
filter2 = load('filter_2.mat');
h2 = filter2.xx;
% Plotting the signal
figure;
plot(x);
title('Signal x[n]');
xlabel('Samples');
ylabel('Amplitude');
% Sampling rate
fs = 18000;  % Hz
X = custom_fft(x);
% Creating the freq axis, as we learned in class we have a change in the
% freq axis in the grapic represantaion, so we create it.
frequencies = (0:length(X)-1)*(fs/length(X));
% Plot the mag of the FFT
figure;
plot(frequencies(1:floor(length(frequencies))),
abs(X(1:floor(length(X)))));
title('FFT of x[n]');
xlabel('frequency (Hz)');
ylabel('|X|');
%done


%% Q3-B presenting the FFT of the filters
%FFT on the filters
H1=fft(h1);
H2=fft(h2);
%Plot the mag of the filters
figure;
subplot(2,1,1);
plot(abs(H1));
title('FFT of h1');
xlabel('frequency');
ylabel('|H_1|');
subplot(2,1,2);
plot(abs(H2));
title('FFT of h2');
xlabel('frequency');
ylabel('|H_2|');
%done
%% Q3-C direct linear convolution between the filters and the signal
y1_direct = manual_conv(x, h1);
y2_direct = manual_conv(x, h2);
% plotting the covolution between the signals
figure;
subplot(2,1,1);
plot(y1_direct(1:500));
title('Linear Convolution Result with h1[n]');
```

```
xlabel('samples');
ylabel('y[n]');
subplot(2,1,2);
plot(y2_direct(1:500));
title('Linear Convolution Result with h2[n]');
xlabel('samples');
ylabel('y[n]');
%done


%% Q3-D
% Creating diffreant frame sizes.
L=[64,128,256,512,1024];
timeOVAconv1=zeros(1,length(L));
timeOVAconv2=zeros(1,length(L));

% Running on each frame size, checking the ova conv for both of the filters
% and saving the results.
for i = 1:length(L)
    tic;
    y1=ova_conv(x,h1,L(i));
    timeOVAconv1(i)=toc;
    tic;
    y2=ova_conv(x,h2,L(i));
    timeOVAconv2(i)=toc;
    % We're also saving the time for each frame size.
end
% Plotting the results.
t=0:1:18060-1;
figure;
subplot(2,1,1)
plot(t(1:500),y1(1:500),t(1:400),y1_direct(1:400))
title('Linear VS. OVA convolution - H1');
xlabel('samples');
ylabel('y[n]');
legend('Linear - h1', 'OVA - h1');
grid on;
subplot(2,1,2)
plot(t(1:500),y2(1:500),t(1:400),y2_direct(1:400))
title('Linear VS. OVA convolution - H2');
xlabel('samples');
ylabel('y[n]');
legend('Linear - h2', 'OVA - h2');
grid on;
figure;
plot(L,timeOVAconv1,'o-',L,timeOVAconv2,'o-');
title('Runing time VS. frame size')
xlabel('Frame size')
ylabel('Runing time')
legend('OVA runing time h1','OVA runing time h2')
grid on;
%done

%conv function- Q3-C
function y = manual_conv(x, h)
    x_l=length(x);
    h_l=length(h);
    % As we learned the size of the conv is the sum of the 2 signals -1.
    y = zeros(1, h_l+x_l-1);
```

```
    for n = 1:length(y)
        for k = 1:h_l
            % For each itr we need to check if the value is within the
            % range for x, normally we would do n-k, because the sum would
            % start from 0, but it is MATLAB so +1 it is.
            if (n-k+1>0) && (n-k+1<=x_l)
                % Summing the products and adding to y in the correct ind.
                y(n) = y(n)+x(n-k+1)*h(k);
            end
        end
    end
end
% done

% OVA convolution function-Q3-D
% the general frame suze will be: Nh + L -1.
% we want to check diffrence sizes of L to check witch frame will give us
% the best results, regarding time.
function y= ova_conv(x,h,L)
    frame_size=2^nextpow2(L+length(h)+1);
    % In order to mult H*X we want to make sure theyre the same frame size
so we
    % add padding to H.
    h_new=[h,zeros(1,frame_size-length(h))];
    H=custom_fft(h_new);
    % Checking how many frames we have.
    nof=ceil(length(x)/L);
    y=zeros(1,length(x)+length(h)-1);
    for i=1:nof
        curr_seg=x((i-1)*L + 1 : min(length(x),i*L));%extract the curr
segment from x
        Curr_Seg=custom_fft([curr_seg,zeros(1,frame_size-
length(curr_seg))]);%fft on the seg
        Y_seg=Curr_Seg.*H;%convolution
        y_seg=custom_ifft(Y_seg);%ifft on y
        % Checking edge case for the end of y.
        str_ind=(i-1)*L+1;
        min_y=min(str_ind+frame_size-1,length(y));
        % We do overlap in order to avoid circ conv
        y(str_ind:min_y)=y(str_ind:min_y)+y_seg(1:min(frame_size,min_y-
str_ind+1))';%coneecting the segments
    end
end
%done
```