יונתן תורג'מן 315468363
יהל אורגד 325010809

בס"ד

עיבוד ספרתי של אותות 1 – מטלת מטלב – יונתן תורגמן ויהל אורגד

שאלה 1:

התבקשנו לממש שגרה המבצעת פעולת FFT, ראשית ללא רקורסיה וכן באמצעות רקורסיה.

$$d1 + d2 = 67$$

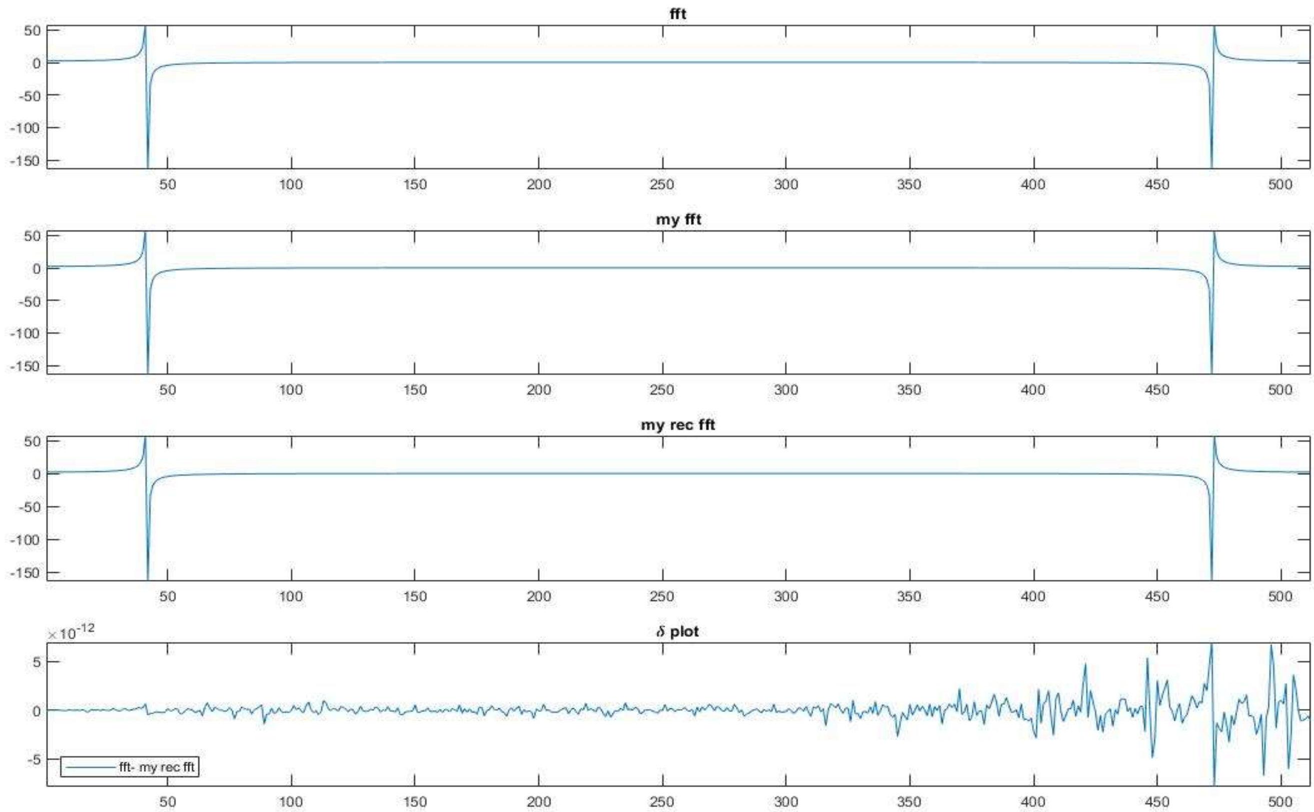
$$67\ \%\ 2 = 1$$

לכן נבצע FFT ע"י דצימציה בזמן. נציג את האלגוריתמים שכתבנו ב$matlab$:

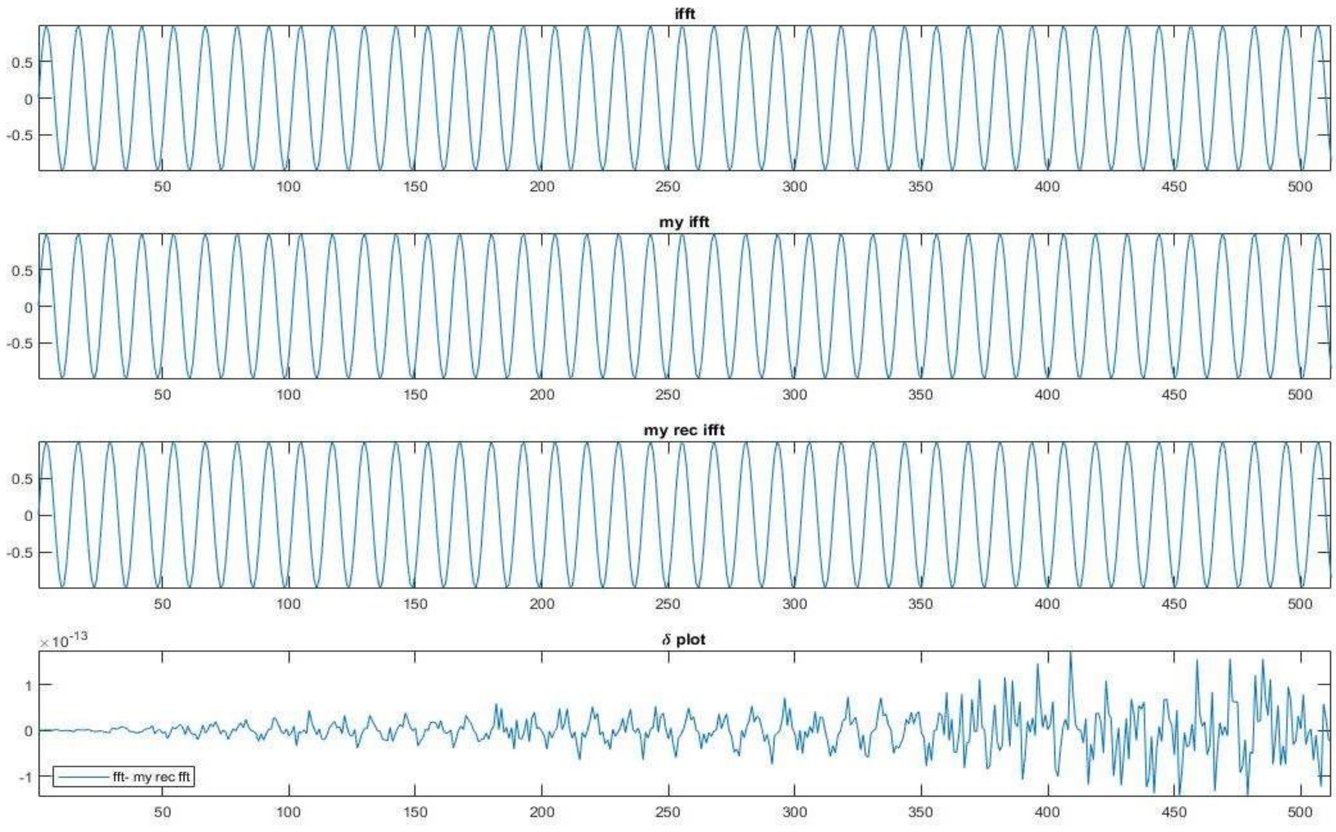
```
function [X] = ex1_FFT(x)
%ex1_FFT algorithm
    g = x(1:2:end);
    h = x(2:2:end);
    N = length(x);
    k = 0:N-1;
    L = N/2;
    X = 0:1:N-1;
    l = 0:1:L-1;
    W_lk_L = exp(-1j*2*pi* l'*k/L);
    W_k_N = exp(-1j*2*pi*k/N);
    %   we used matrix multiplication to compute the algorithm
    for k = 1:1:N
        X(:,k)  = g*W_lk_L(:,k) + W_k_N(:,k) * h * W_lk_L(:,k) ;
    end
end


function F = ex1_FFT_recursive(f)
%just to be in sync with the ex1_IFFT_recursive function
F = fft_rec(f) ;
end

function F = fft_rec(f)
n = length(f);
if (n == 1) %if the length is 1 return the function
  F = f;
else
    %divide into two vectors, evens and odds
  f_even = f(1:2:n);
  f_odd = f(2:2:n);
  %recursive calling, in a tree style
  X1 = fft_rec(f_even);
  X2 = fft_rec(f_odd).*Wn(n);
  F1 = X1 + X2;
  F2 = X1 - X2;
  F = [F1 F2];
end
end

function W = Wn(n)
%just to be more neat
m = n/2;
W = exp(-2*pi*1i.*(0:1:m-1)/n);
end

function [x] = ex1_IFFT(X)
```

```
%we used our fft function and the fact that it is a near identical
%algorithem (just need to divide by length(X))
    x = conj(ex1_FFT(conj(X)))/length(X);
end
function F = ex1_IFFT_recursive(f)
%we need to divide by length(f) after the recursive algorithm
F = ifft_rec(f)/length(f);
end

function F = ifft_rec(f)
n = length(f);
if (n == 1)%if the length is 1 return the function
  F = f;
else
  %divide into two vectors, evens and odds
  f_even = f(1:2:n);
  f_odd = f(2:2:n);
  %recursive calling, in a tree style
  X1 = ifft_rec(f_even);
  X2 = ifft_rec(f_odd).*Wn(n);
  F1 = X1 + X2;
  F2 = X1 - X2;
  F = [F1 F2];
end

end
```

על מנת לבחון את האלגוריתמים שבנינו הכנסנו אות סינוסי (הקפדנו שמספר הדגימות יהיה חזקה של 2 על מנת שהאלגוריתם יעבוד). יצרנו גרף משולב עם כל סוגי הFFT שיצרנו :



יונתן תורג'מן 315468363
יהל אורגד 325010809



הוספנו את גרף השגיאה בתחתית. ניתן לראות שאנחנו מקבלים שגיאה מצוינת בסדר גודל של 10^{-12}.

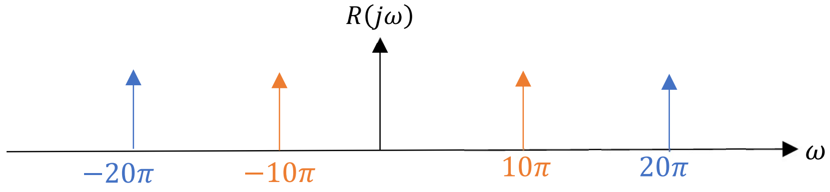
יונתן תורג'מן 315468363
יהל אורגד 325010809

שאלה 2:

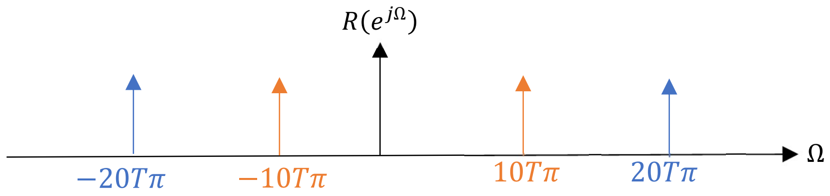
סעיף א

$$r(t)=\cos(2\pi 5t)+\cos(2\pi 10t)=s(t)+v(t)$$

התמרת פורייה של אות זה תיראה כך:



וכעת לפי משפט הדגימה, נניח שדגמנו בתדר $F_s=\frac{1}{T}$, האות הדגום בתדר ייראה כך (שכפול בודד שלו):



כעת אנחנו רוצים להנחית את $v(t)$ פי A (מכיוון שA קטן מ1 הכוונה היא פשוט להכפיל בו) ובנוסף נרצה להשאיר את $s(t)$ כמו שהוא.

זה בדיוק מתאים לנו למסנן הנתון ורק נצטרך לדרוש שכל דלתא תהיה בתחום ההגבר המתאים לה:

$$10T\pi<\frac{\pi}{4}$$

$$\frac{\pi}{4}\le 20T\pi\le \pi$$

יחד שני תנאים אלו נותנים דרישה $\frac{1}{80}\le T\le\frac{1}{40}$

כלומר $40\le F_s\le 80\ [Hz]$

סעיף ב

נתון אורך המסנן $N=102$ דגימות

נגדיר אורך האות לאחר שדגמנו אותו M

במוצא המסנן אנחנו מקבלים כמובן $r[n]*h[n]$

וכפי שלמדנו האורך של אות זה הוא $M+N-1$ ונותר לדרוש

$$M+N-1=2048$$

יונתן תורג'מן 315468363
יהל אורגד 325010809

ונקבל $M=1947$ מספר הדגימות שיש ליטול כדי לקבל במוצא המסנן 2048 דגימות.

סעיף ג

$$s[n] \triangleq r[n] * h[n]$$

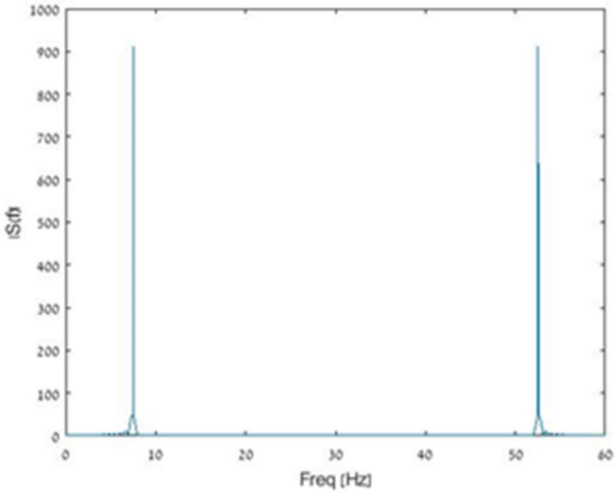
כעת מכיוון שאין לנו את המסנן בזמן נרצה לעבור לפי הנוסחה בתדר:

$$S[k] \triangleq R[k] \cdot H[k]$$

נכניס את האות שלנו למטלב עם 1947 דגימות, ונעשה לו התמרה בעזרת הפונקציה שבנינו בחלק א, לאחר ריפודים מתאימים באפסים, נחשב את על פי הנוסחה לעיל ונצייר את הערך המוחלט שלו לפי התדר כמבוקש: הקוד:

```
h=load('filter_0.25_101.mat');

h=struct2cell(h);

h=cell2mat(h);

r_t=@(t) cos(2*pi*5.*t)+cos(2*pi*10.*t);

%until here it was just definitions

%lets sample r_t to r_n

%we will choose Fs to be 60

r_n=zeros(1,1947);

for i=1:1947

    j=0:(1/60):((1947/60)-(1/60));%sample vector

    r_n(i)=r_t(j(i)); %sampling

end

%now lets pad in zeros where it is needed

r_n_pad=[r_n zeros(1,101)];

h_pad=[h zeros(1,1946)];

R_k=fft(r_n_pad);

H_k=fft(h_pad);

%now we can finally calculate S_k

S_k=R_k.*H_k;

%now lets plot it

k=0:(60/2047):60;

plot(k,abs(S_k))

xlabel('Freq [Hz]')

ylabel('|S(f)|')
```

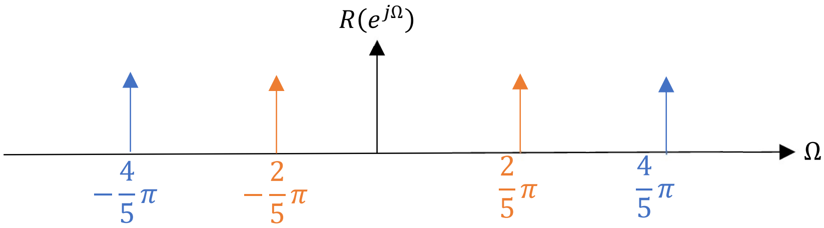
הציור:



<u>סעיף ד</u>

נתון $F_s=\frac{1}{T}=25\ [Hz]$ כלומר $T=\frac{1}{25}\ [sec]$

כעת ממש כמו בסעיף א נקבל שהאות הדגום בתדר (שכפול בודד שלו) ייראה כך:



(הערה: בזכור מאותות ומערכות, כאשר מדובר באותות סינוסוידאלים כל עוד השכפול הבודד הראשון אינו חוצה את גבולות $[-\pi,\pi]$ לא יהיו דריכות, כיוון שהשכפולים הם כל 2π, ולכן התמונה המוצגת כאן היא נכונה)

כאן על מנת לסנן את $v[n]$ מספיק פשוט להכפיל את הדלתאות שהיא תורמת בתדר בA כיוון שנתון כי $1 \ll A$ אזי הרכיבים שהיא תורמת בתדר כבר לא יהיו משמעותיים לעומת הרכיבים של $s[n]$.

ממש כמו בסעיף א נבדוק האם הדלתאות עומדות במקומות המתאימים כדי לקבל את ההנחתה הרצויה:

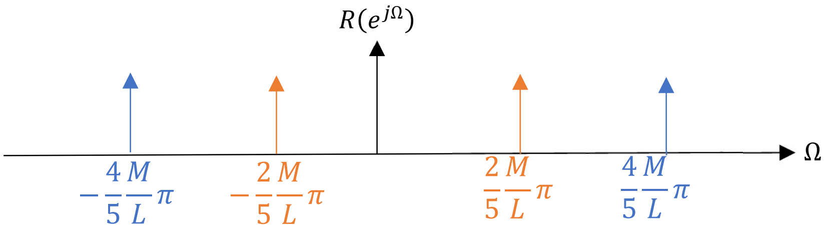
$$\frac{2}{5}\pi \cancel{<} \frac{\pi}{4}$$

$$\frac{\pi}{4} \cancel{\le} \frac{4}{5}\pi \le \pi$$

רואים שהתנאי הראשון לא מתקיים וגם הדלתאות של $s[n]$ יונחתו, לכן לא ניתן לסנן את $v[n]$ <u>**רק**</u> על ידי שימוש במסנן הנתון.

אבל, ניתן לעשות אינטרפולציה בL ודצימציה בM על מנת לקבל את הדלתאות במקומות הבאים:

יונתן תורג'מן 315468363
יהל אורגד 325010809



וכעת על פי אותם תנאים נקבל:

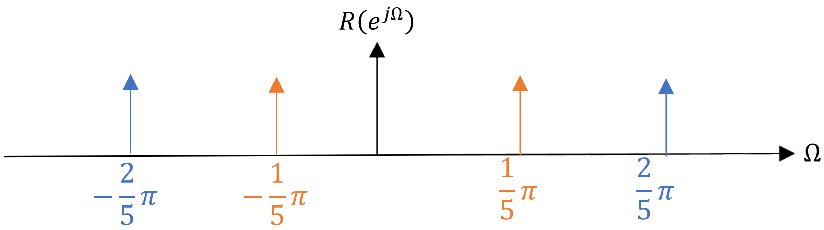
$$\frac{2}{5}\frac{M}{L}\pi < \frac{\pi}{4}$$

$$\frac{\pi}{4} \le \frac{4}{5}\frac{M}{L}\pi \le \pi$$

וקצת אלגברה תביא אותנו לתנאי הבא על M, L :

$$\frac{5}{16} \le \frac{M}{L} < \frac{5}{8}$$

ניתן לראות שחצי נמצא בתחום לכן אם נבחר למשל $L = 4$, $M = 2$ נקבל כי (לא מתחייב על הגבהים של הדלתאות זה לא מה שקריטי כאן)



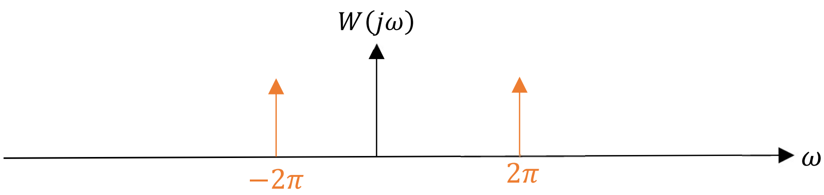
וכעת ברור שזה עומד בתנאים שלנו והמסנן הנתון ייתן את ההנחתה הרצויה וסך הכל יסנן את $v[n]$:

$$\frac{1}{5}\pi < \frac{\pi}{4}$$

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

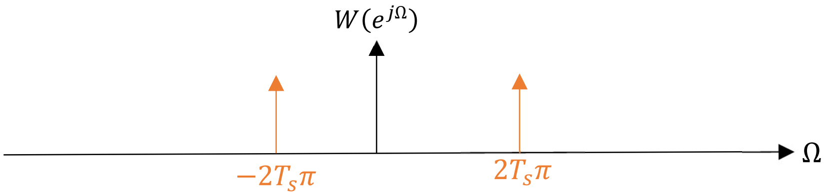
<u>סעיף ה</u>

בוא נסתכל על אות המוצא הרצוי, בתדר הוא ייראה כך:

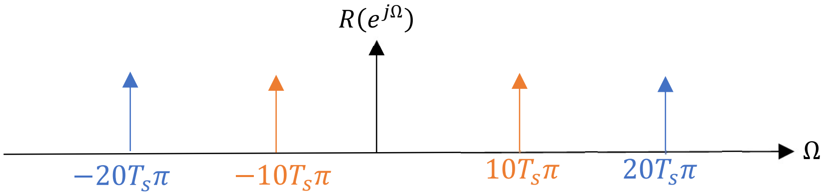


ולפני החזרה לאות רציף הוא היה נראה בתדר כך (שכפול בודד שלו):

יונתן תורג'מן 315468363
יהל אורגד 325010809



כזכור האות שלנו לאחר הדגימה נראה כך:



אבל בציור זה נמצא רק שכפול בודד וישנם עוד שכפולים כל 2π, כך למשל הדלתא החיובית של $v[n]$ נמצאת במיקומים הבאים עבור כל k שלם: $20T_s\pi - 2\pi k$

כעת נדרוש ששכפול k כלשהו של הדלתא הזו ייפול בדיוק איפה שנמצאת הדלתא החיובית של אות המוצא הרצוי בתדר לפני המעבר חזרה לרציף כלומר ב: $2T_s\pi$

$$20T_s\pi - 2\pi k = 2T_s\pi$$

ומכאן נקבל כי $T_s = \frac{k}{9}$.

כעת נבחר את הk כך שהדלתא $20T_s\pi - 2\pi k$ תהיה באזור שבו המסנן שווה 1 כלומר:

$$20T_s\pi - 2\pi k = 20\pi\frac{k}{9} - 2\pi k < \frac{\pi}{4}$$

$$k < \frac{9}{8}$$

k חייב להיות מספר שלם ולכן נבחר $k = 1$ כי הוא היחיד שמתאים, וסך הכל קיבלנו:

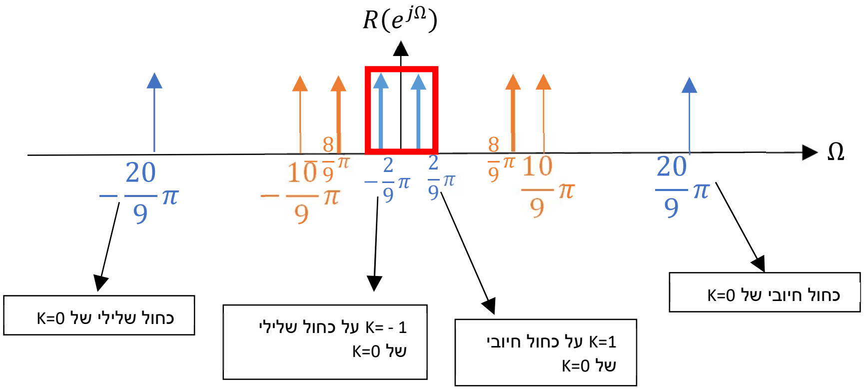
$T_s = \frac{1}{9}$, $F_s = 9\ [Hz]$ וזהו קצב הדגימה בוא ניתן לקבל את המוצא הרצוי.

בוא נבדוק איפה הדלתא החיובית של $s[n]$ תימצא במקרה שכזה,

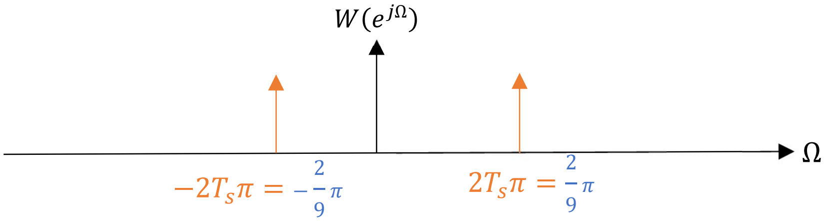
$$10T_s\pi - 2\pi k = \frac{10}{9}\pi - 2\pi = -\frac{8}{9}\pi$$

ובעת אם נכניס גם את השכפולים של $k = -1$ והשפעתם על הדלתאות השליליות נקבל את התמונה הבאה:

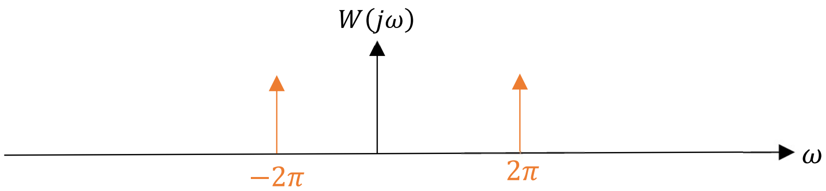
יונתן תורג'מן 315468363
יהל אורגד 325010809



ואכן רואים שהמסנן באדום לוקח רק את 2 הדלתאות הרצויות שיוצרות לנו את



שלאחר השחזור לרציף שווה לאות המוצא הרצוי, ובכך הגענו למטרה הרצויה:



יונתן תורג'מן 315468363
יהל אורגד 325010809

בס"ד

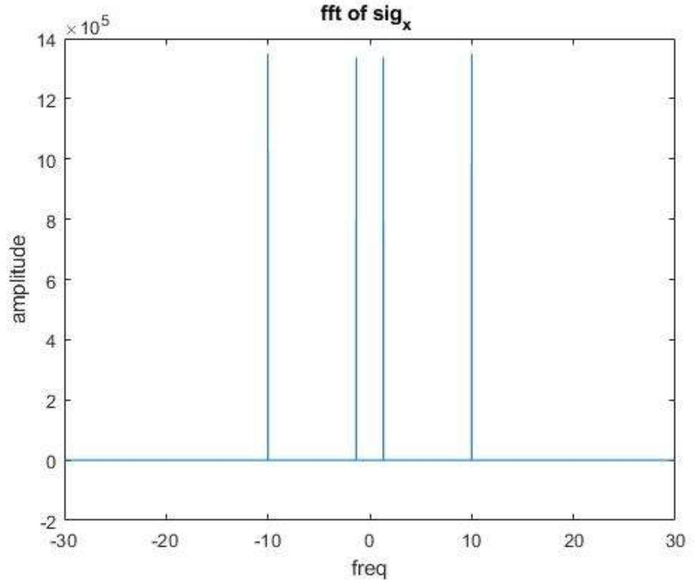
שאלה 3:

סעיף א

נטען ונציג את האות $x[n]$ על ידי הרצת הקוד הבא:

```
load('sig_x.mat');
plot(-30:61/274499:30,fftshift(fft(x)))
```

סיגנל המוצא נראה כך:



ניתן בקלות להבחין כי התדרים הפעילים הם $f=\pm10,\pm1.333[Hz]$.

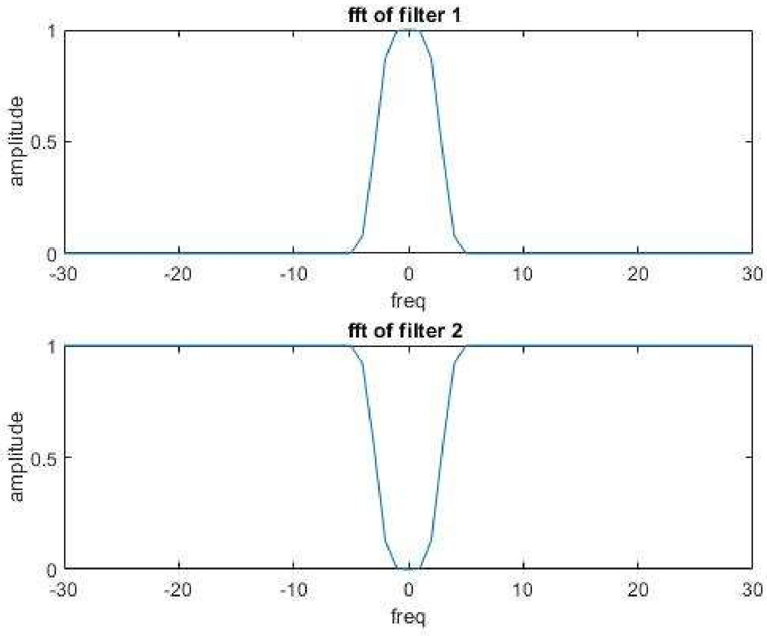
סעיף ב

נטען את הסיגנלים ונציג את התמרותיהן של שני המסננים על ידי הרצת הקוד הבא:

```
load('filter_1.mat');
F1 = xx;
load('filter_2.mat');
F2 = xx;
figure
subplot(2,1,1)
plot(-30:1:30,abs(fftshift(fft(F1))));
title('fft of filter 1');
xlabel('freq');
ylabel('amplitude');
axis tight;
subplot(2,1,2)
plot(-30:1:30,abs(fftshift(fft(F2))));
title('fft of filter 2');
xlabel('freq');
ylabel('amplitude');
axis tight;
```

יונתן תורג'מן 315468363
יהל אורגד 325010809

ההתמרות נראות כך :



וניתן בקלות להבחין כי מסנן 1 הינו מעביר תדרים נמוכים כלומרLPF

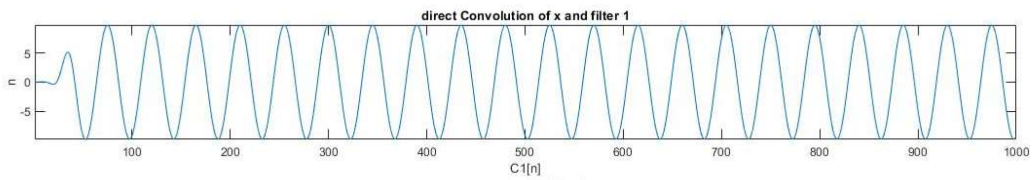
בעוד שמסנן 2 הינו מעביר תדרים גבוהים כלומר.HPF

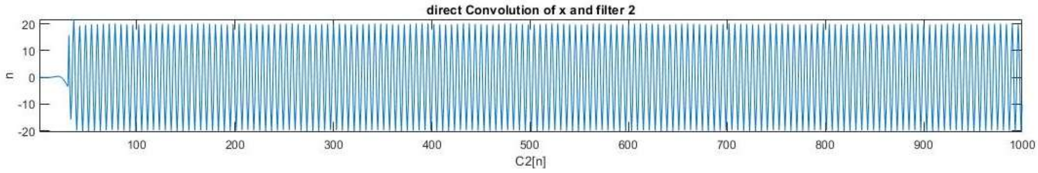
<u>סעיף ג</u>

מימשנו קונבולוציה ישירה על ידי הקוד של הפונקציה הבאה:

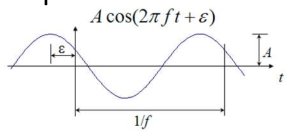
```
function y = direct_Convolution(x, h)
 %just the straightforward convolution algorithm
    N = length(x);
    p = length(h);
    Nk = N + p -1;
    y = zeros(1,Nk);
    for i = 1:N
          for k = 1:p
           y(i+k-1) = y(i+k-1) + h(k)*x(i);
          end
    end

end
```

את ההרצה של פונקציה זו על שני המסננים ניתן למצוא בנספחים יחד עם שאר ההרצות (הנדרשות לסעיף ו') בצורה מסודרת.
להלן התוצאות רק של הקונבולוציה הישירה:





ניתן לראות בבירור כי מסנן 2 אכן העביר את התדרים הגבוהים ולכן האוסילציות בגרף שלו 'חזקות' יותר, בעוד שמסנן 1 העביר את התדרים הנמוכים ולכן האוסילציות בגרף שלו 'חלשות' יותר. על מנת להבהיר את הנקודה ניזכר:



ובעת ברור כי ככל שהתדר גבוהה יותר, האוסילציה קצרה ('חזקה') יותר.

(**הערה:** הקוד של חישוב זמן הריצה של הקונבולוציה הישירה **מפורט בסעיף הבא (סעיף ד')**, יחד עם חישובי שאר זמני הריצה הדרושים)

קיבלנו כי **זמן הריצה של הקונבולוציה הישירה** עם מסנן 1 הוא $0.0578918\ [sec]$.

וזמן הריצה של הקונבולוציה הישירה עם מסנן 2 הוא $0.0579744\ [sec]$.

<u>סעיף ד</u>

מימשנו קונבולוציה ליניארית לפי אלגוריתם OVA על ידי הקוד של הפונקציה הבאה:

```
function y = OVA(x, h, K)
     N = length(x);
     P = length(h);
     L = K - P + 1;
     h = [h zeros(1,(K - P))]; %we need to add K - P zeros after the
filter
     H = fft(h);
     t = 1;
     y = zeros (1,(P+N-1)); %y has P+N samples (let matlab know
length, not necessary)
     while t < N - L %while loop to calculate each frame
        X = [x(t:(t+L-1)) zeros(1,P-1)];
        wt = ifft((fft(X)).* H);
        yt = y(t:t+K-1) + wt;
        y (t:t+K-1) = yt; %add the frame to the finel y
        t = t + L; %move to next frame
     end

end
```

כאשר את פרמטרי האלגוריתם קבענו לפי מה שלמדנו בתרגול 7, להלן תיאור השיטה שנלמד שלפיו קבענו את פרמטרי האלגוריתם:

ראשית נגדיר P אורך המסנן.

בשיטה זו אנו לוקחים מסגרות של אות הכניסה, כל אחת באורך $K-P+1$

מרפדים כל מסגרת כזו ב $P-1$ אפסים, כך שתהיה באורךK

מבצעים התמרת DFT באורך K למסגרת

מכפילים במישור הDFT בין התמרת המסנן להתמרת המסגרת (המרופדת כמובן)

על תוצאת המכפלה מבצעים התמרה הפובה $IDFT$ שנותנת לנו מסגרת (בודדת) של המוצא הרצוי.

כעת עוברים למסגרת הבאה של אות הכניסה, שנמצאת מיד אחרי המסגרת הקודמת לה, שלאחר חזרה על אותו תהליך תיתן מסגרת מוצא בעלת $P-1$ חפיפות עם קודמתה, וביניהן נבצע חיבור.

יונתן תורג'מן 315468363
יהל אורגד 325010809

נחזור על תהליך זה עבור כל המסגרות עד לסוף אות הכניסה ובכך נקבל את אות המוצא הסופי והרצוי.

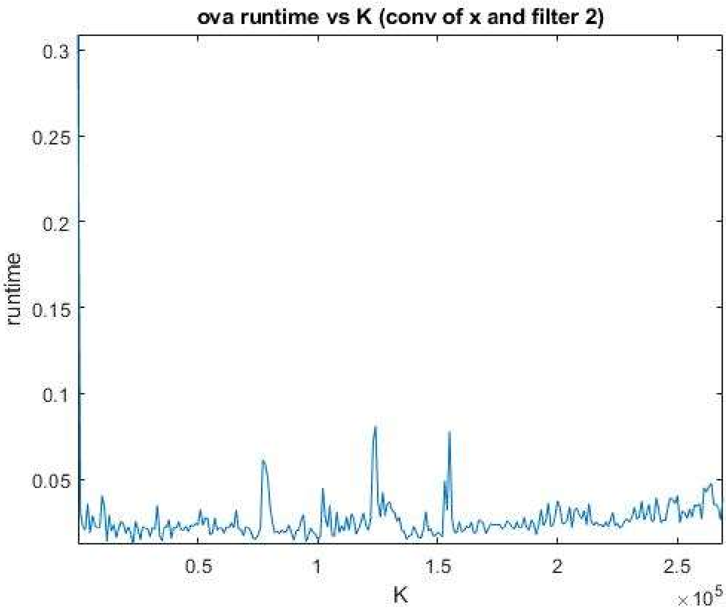
את **הגרף של זמן הריצה** של מימוש הקונבולוציה הלינארית על ידי אלגוריתם OVA, על שני המסננים הנתונים, כפונקציה של גודל המסגרת K , חישבנו והצגנו על ידי הקוד הבא
(**הערה חשובה:** כמו כן חישבנו את זמני הריצה של OVS ושל החישוב הישיר גם בקוד הנ"ל תחת אותה לולאה לשם נוחות ולכן **שאר חישובי הזמנים בסעיפים ג' ה' ו ו' מופנים לקוד זה**):

```
function [] = ex3_runtime()
load('sig_x.mat');
load('filter_1.mat');
F1 = xx;
load('filter_2.mat');
F2 = xx;
j=1;
for K = 100:1000:270000
    %ova runtime calculation
    tic
    ova_conv1 = OVA(x, F1, K);
    T = toc;
    time_ova_f1(j) = T ;

    tic
    ova_conv1 = OVA(x, F2, K);
    T = toc;
    time_ova_f2(j) = T ;
    %ovs runtime calculation
    tic
    ovs_conv1 = OVS(x, F1, K);
    T = toc;
    time_ovs_f1(j) = T ;

    tic
    ovs_conv1 = OVS(x, F2, K);
    T = toc;
    time_ovs_f2(j) = T ;

    %direct runtime calculation
    tic
    direct_conv2 = direct_Convolution(x, F2);
    T = toc;
    time_direct_f2(j) = T;

    tic
    direct_conv1 = direct_Convolution(x, F1);
    T = toc;
    time_direct_f1(j) = T;
    j=j+1;
end


fprintf("min runtime direct f1: %d\n",min(time_direct_f1) )

fprintf("min runtime direct f2: %d\n",min(time_direct_f2) )


K = 100:1000:270000;
```

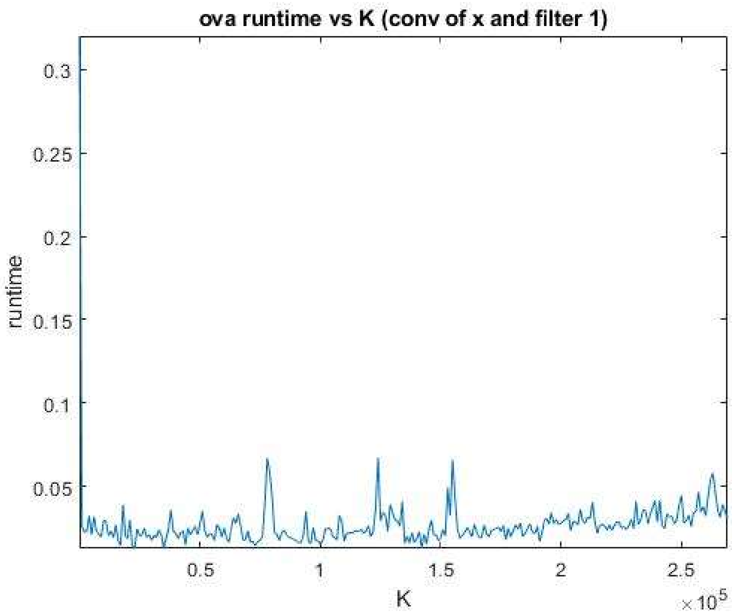
```
%ova plots
[opt_runtime_ova_f1,index_ova_f1] = min(time_ova_f1);
fprintf('optimal runtime ova f1: %d .  K = %d\n',opt_runtime_ova_f1,
K(index_ova_f1))

figure
plot(K, time_ova_f1)
title('ova runtime vs K (conv of x and filter 1)');
xlabel('K');
ylabel('runtime');
axis tight;

[opt_runtime_ova_f2,index_ova_f2] = min(time_ova_f2);
fprintf('optimal runtime ova f2: %d .  K = %d\n',opt_runtime_ova_f2,
K(index_ova_f2))

figure
plot(K, time_ova_f2)
title('ova runtime vs K (conv of x and filter 2)');
xlabel('K');
ylabel('runtime');
axis tight;


%ovs plots
[opt_runtime_ovs_f1,index_ovs_f1] = min(time_ovs_f1);
fprintf('optimal runtime ovs f1: %d .  K = %d\n',opt_runtime_ovs_f1,
K(index_ovs_f1))

figure
plot(K, time_ovs_f1)
title('ovs runtime vs K (conv of x and filter 1)');
xlabel('K');
ylabel('runtime');
axis tight;

[opt_runtime_ovs_f2,index_ovs_f2] = min(time_ovs_f2);
fprintf('optimal runtime ovs f2: %d .  K = %d\n',opt_runtime_ovs_f2,
K(index_ovs_f2))

figure
plot(K, time_ovs_f2)
title('ovs runtime vs K (conv of x and filter 2)');
xlabel('K');
ylabel('runtime');
axis tight;

%all together plot
figure
subplot(2,1,1)
plot(K, time_ova_f1, K, time_ovs_f1)
title('ova and ovs runtime vs K (conv of x and filter 1)');
xlabel('K');
ylabel('runtime');
legend({'ova','ovs'},'Location','southwest')
axis tight;

subplot(2,1,2)
plot(K, time_ova_f2, K, time_ovs_f2)
```

```
title('ova and ovs runtime vs K (conv of x and filter 2)');
xlabel('K');
ylabel('runtime');
legend({'ova','ovs'},'Location','southwest')
axis tight;

end
```

גרף זמני הריצה על מסנן 1:



גרף זמני הריצה על מסנן 2:



מתוך הקוד גם חילצנו את הזמן המינימלי מבין כל ההרצות עבור הKים השונים וקיבלנו כי **זמן הריצה האופטימלי** (הכי קצר) מתקבל עבור $K = 142100$ והוא:

$$t_{OVS,opt} = 0.01128730[sec]$$

<u>סעיף ה</u>

מימשנו קונבולוציה לינארית לפי אלגוריתם OVS על ידי הקוד של הפונקציה הבאה:

```
function y = OVS(x, h, K)
```

```
    N = length(x);
    x = [x zeros(1,50)];
    P = length(h);
    L = K - P + 1;
    h = [h zeros(1,(K - P))];%we need to add K - P zeros after the
filter
    H = fft(h);
    t = 1;
    y = zeros (1,(P+N-1));%y has P+N samples (let matlab know
length, not necessary)
    while t < N - L %while loop to calculate each frame
       X = x(t:(t + K - 1));
       wt = ifft((fft(X)).* H);
       y(t:t + L - 1) = wt((P -1):(K - 1));%add the frame to the
finel y
       t = t + L; %move to next frame
    end

end
```

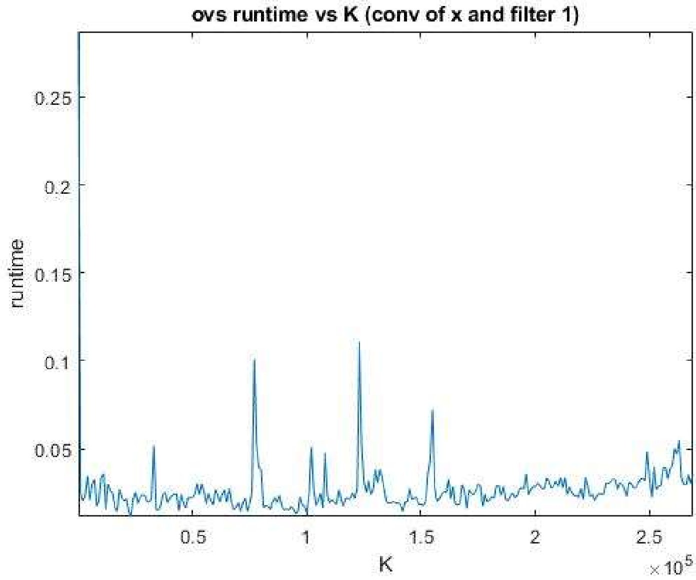
כאשר את פרמטרי האלגוריתם קבענו לפי מה שלמדנו בתרגול 7, להלן תיאור השיטה שנלמד שלפיו קבענו את פרמטרי האלגוריתם:

ראשית נגדיר P אורך המסנן.
בשיטה זו אנו לוקחים מסגרות של אות הכניסה, כל אחת באורך $K > P$
מרפדים את המסנן במישור הזמן ב $P-1$ אפסים, כך שיהיה באורך K
מבצעים התמרת DFT באורך K למסגרת
מכפילים במישור הDFT בין התמרת המסנן להתמרת המסגרת (המרופדת כמובן)
על תוצאת המכפלה מבצעים התמרה הפוכה $IDFT$ שנותנת לנו מסגרת (בודדת) של המוצא הרצוי.
כל מסגרת של המוצא כוללת $P-1$ דגימות (הראשונות) שאינן שוות לקונבולציה הלינארית ולכן נמחק אותן ונשתמש רק ב $K-P+1$ הדגימות האחרונות עבור כל מסגרת.

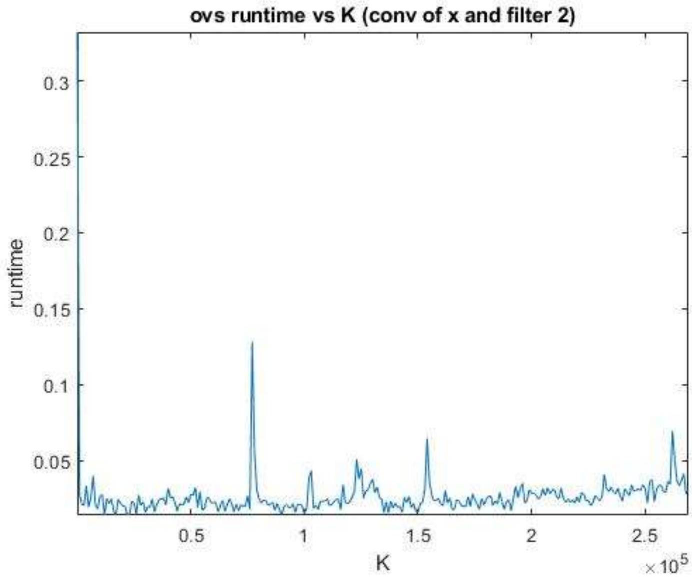
כעת עוברים למסגרת הבאה, חשוב! המסגרת הבאה נמצאת במרחק L מתחילת המסגרת הקודמת, בתרגול הוכחנו כי אורך L האופטימלי כך ששיטת OVS תיתן את כל דגימות המוצא, מבלי לחשב אותה דגימה פעמיים הוא $L = K - P + 1$, וכך יוצא שעבור L זה, לאחר חיבור כל מסגרות המוצא (שחושבו בדרך המפורטת לעיל) יחדיו אחת אחרי השנייה, נקבל את המוצא הרצוי.

את **הגרף של זמן הריצה** של מימוש הקונבולוציה הלינארית על ידי אלגוריתם OVS, על שני המסננים הנתונים, כפונקציה של גודל המסגרת K , חישבנו והצגנו על ידי **הקוד שהוצג בסעיף הקודם (סעיף ד) .**

גרף זמני הריצה על מסנן 1:



גרף זמני הריצה על מסנן 2:



מתוך הקוד גם חילצנו את הזמן המינימלי מבין כל ההרצות עבור הKים השונים וקיבלנו כי **זמן הריצה האופטימלי** (הכי קצר) מתקבל עבור $K_{opt} = 93100$ והוא:

$$t_{OVS,opt} = 0.01178690[sec]$$

<u>סעיף ו</u>

כעת נציג את שני זמני הריצה עבור OVA & OVS על אותו הגרף, פעמיים פעם אחת עבור כל מסנן.

עבור הקונבולציה הישירה, ביצענו מספר זהה של הרצות כמו ova, ovs ולבסוף חישבנו את המינימום של כל ההרצות, שאלו כמובן הזמנים האופטימליים .

להלן הקוד:

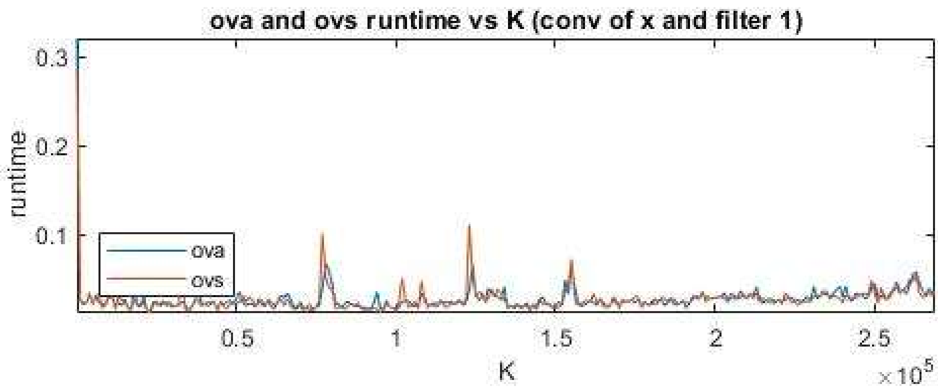
(**הערה:** הקוד כולל רק את הצגת וקטורי זמני הריצה כפונקציה של גודל המסגרת, החישוב עצמו של ווקטורים אלו **כבר מוצג בסעיף ד'**):

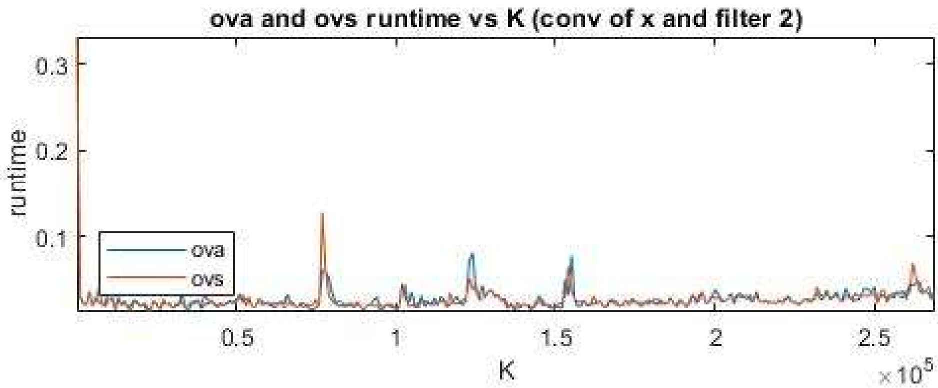
```
figure
subplot(2,1,1)
plot(K, time_ova_f1, K, time_ovs_f1)
title('ova and ovs runtime vs K (conv of x and filter 1)');
```

יונתן תורג'מן 315468363
יהל אורגד 325010809

```
xlabel('K');
ylabel('runtime');
legend({'ova','ovs'},'Location','southwest')
axis tight;

subplot(2,1,2)
plot(K, time_ova_f2, K, time_ovs_f2)
title('ova and ovs runtime vs K (conv of x and filter 2)');
xlabel('K');
ylabel('runtime');
legend({'ova','ovs'},'Location','southwest')
axis tight;
```

הגרפים:





בנוסף נשווה את הזמן המינימלי של שלושת השיטות :

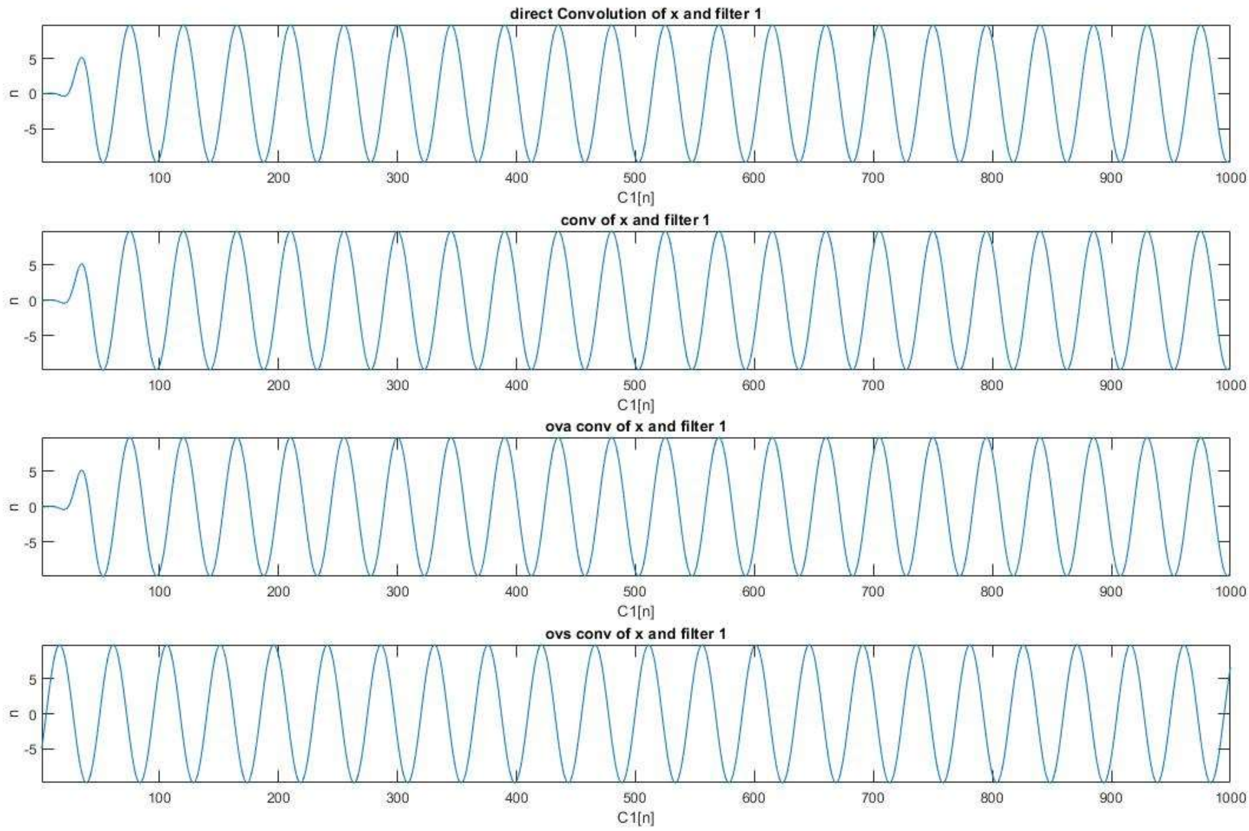
שיטה	זמן מינימלי עבור פילטר 1 $[sec]$	זמן מינימלי עבור פילטר 2 $[sec]$
OVA	0.012873	0.016009
OVS	0.017869	0.01926
$Direct\ Convolution$	0.0578918	0.0579744

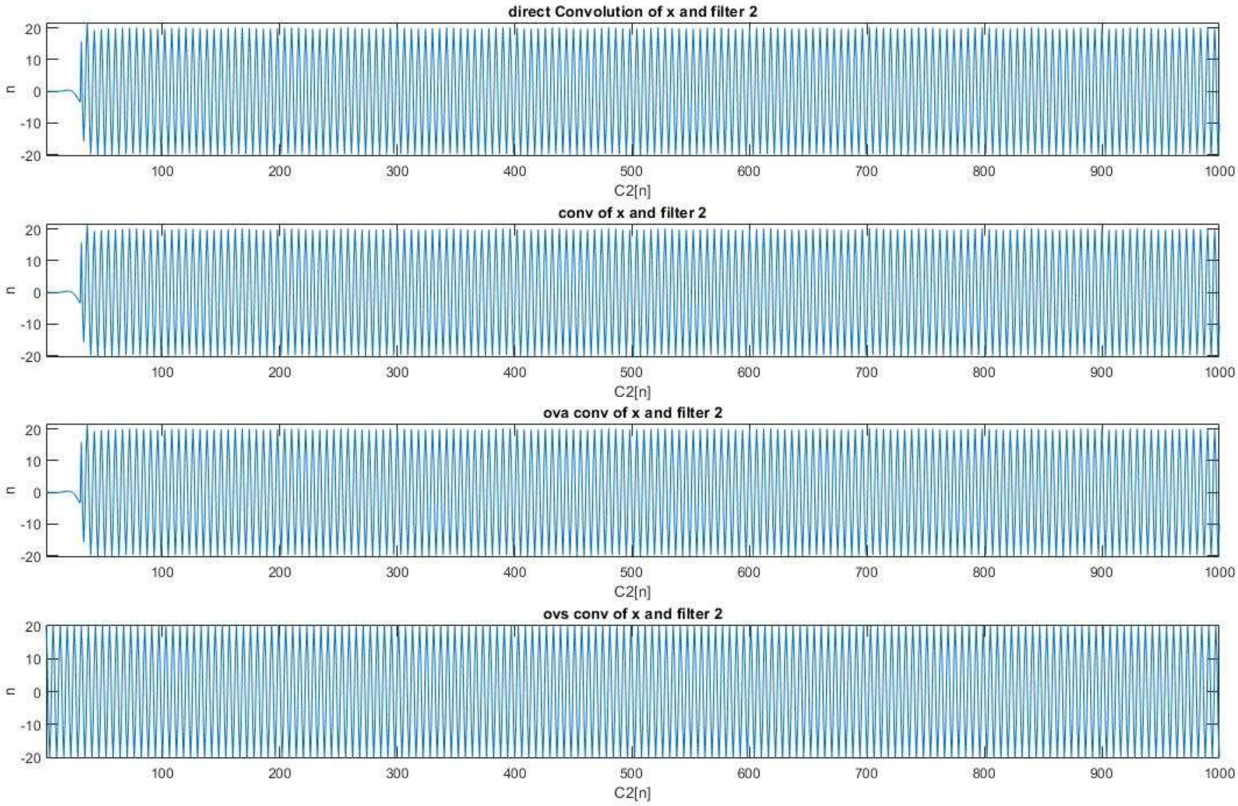
לכן לפי התוצאות שלנו קיבלנו שהאלגוריתמים $OVA\ \&\ OVS$ יעילים יותר במקרה שלנו, מה שעולה בקנה אחד עם מה שלמדנו בתרגול, כי בכל שיש יותר דגימות, השימוש בFFT ובשיטות אלו בפרט, יהיה יותר יעיל מקונבולוציה ישירה.

יונתן תורג'מן 315468363
יהל אורגד 325010809

<u>סעיף ז</u>

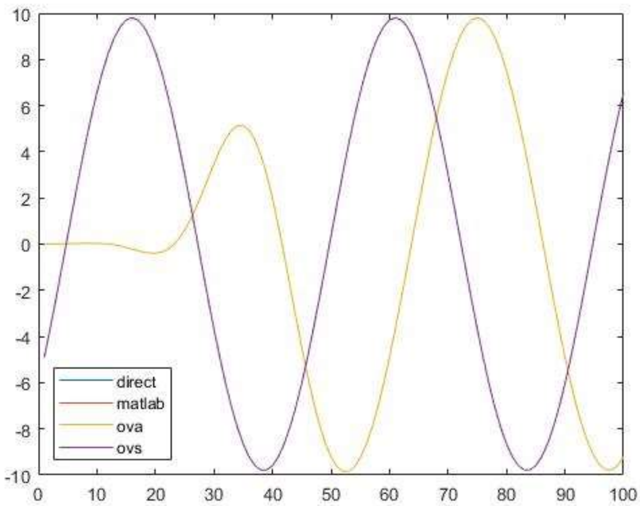
הקוד המלא של הרצת 8 הגרפים הבאים מפורט בנספח מספר 2 בסוף הקובץ.
ביצענו את ההרצות שנתנו את כל הגרפים הבאים עבור בחירת . $K_{opt} = 10062$
להלן הסבר על הגרפים:
תמונה עליונה – 4 סוגי קונבולוציה עם מסנן 1
תמונה עליונה – 4 סוגי קונבולוציה עם מסנן 2
בכל אחת מהתמונות:
הגרף העליון הוא תוצאת הקונבולוציה הישירה שאנו מימשנו,
מתחתיו זוהי תוצאת פונקציית הקונבולוציה המובנת של המטלב,
מתחתיו זוהי תוצאת אלגוריתם ה,OVA
ומתחתיו זוהי תוצאת אלגוריתם OVS .





ניתן לראות כי כל התוצאות מתואמות למעט סטיית הפאזה באלגוריתם OVS. (ניסינו להבין מה גורם לבעיה אבל לא הצלחנו למצוא).

אם נגדיל את תחילת הגרף ונשים את כל התוצאות על גרף אחד :



ניתן לראות שהגרפים זהים ורק לאלגוריתם OVS יש סטיית פאזה אבל צורת הגרף זהה. (לא רואים את שאר הגרפים כי הם זהים לOVA לכן הם מוסתרים).

יונתן תורג'מן 315468363
יהל אורגד 325010809

נספח1: חלק א הקוד המלא:

```
function [F] = ex1(f)
%EX1 Summary of this function goes here
%   Detailed explanation goes here
F_rec = ex1_FFT_recursive(f);
F = ex1_FFT(f);
IF_rec = ex1_IFFT_recursive(F_rec);
IF = ex1_IFFT(F);
x = 1:1:length(F);
%FFT plot
figure
subplot(4,1,1)
plot(x,real(fft(f)));
title('fft')
axis tight;

subplot(4,1,2)
plot(x,F)
title('my fft')
axis tight;

subplot(4,1,3)
plot(x,real(F_rec))
title('my rec fft')
axis tight;

subplot(4,1,4)
plot(x,real(fft(f) - F))
legend({'fft- my rec fft '},'Location','southwest')
title('\delta plot')
axis tight;

x0=300;
y0=-100;
width=1300;
height=800;
set(gcf,'position',[x0,y0,width,height])

%IFFT plot
figure
subplot(4,1,1)
plot(x,ifft(F))
title('ifft')
axis tight;

subplot(4,1,2)
plot(x,IF)
title('my ifft')
axis tight;

subplot(4,1,3)
plot(x,IF_rec)
title('my rec ifft')
axis tight;

subplot(4,1,4)
plot(x,ifft(F) - IF)
legend({'fft- my rec fft '},'Location','southwest')
```

```
title('\delta plot')
axis tight;

x0=300;
y0=-100;
width=1300;
height=800;
set(gcf,'position',[x0,y0,width,height])
end
```

נספח 2: חלק ג הקוד המלא (הפונקציות נמצאות בסעיפים המתאימים עצמם):

```
function [] = ex3()
%EX3 Summary of this function goes here
%   Detailed explanation goes here
load('sig_x.mat');
figure
plot(-30:61/274499:30,fftshift(fft(x)))
load('filter_1.mat');
F1 = xx;
load('filter_2.mat');
F2 = xx;
figure
subplot(2,1,1)
plot(-30:1:30,abs(fftshift(fft(F1))));
title('fft of filter 1');
xlabel('freq');
ylabel('amplitude');
axis tight;
subplot(2,1,2)
plot(-30:1:30,abs(fftshift(fft(F2))));
title('fft of filter 2');
xlabel('freq');
ylabel('amplitude');
axis tight;

j=1;
for K = 62:1000:270000
    tic
    ova_conv1 = OVA(x, F1, K);
    time(j) = toc;
    j=j+1;
end

[M,I] = min(time);
plot(time)

K = 62 + (I-1)*1000;

d_conv1 = direct_Convolution(x,F1);
d_conv1 = d_conv1(1:1000);

mtlb_conv1 = conv(x,F1);
mtlb_conv1 = mtlb_conv1(1:1000);

ova_conv1 = OVA(x, F1, K);
ova_conv1 = ova_conv1(1:1000);
```

```
ovs_conv1 = OVS(x, F1, K);
ovs_conv1 = ovs_conv1(1:1000);

d_conv2 = direct_Convolution(x,F2);
d_conv2 = d_conv2(1:1000);

mtlb_conv2 = conv(x,F2);
mtlb_conv2 = mtlb_conv2(1:1000);

ova_conv2 = OVA(x, F2, K);
ova_conv2 = ova_conv2(1:1000);

ovs_conv2 = OVS(x, F2, K);
ovs_conv2 = ovs_conv2(1:1000);


figure
plot(1:100, real(d_conv1(1:100)),1:100, real(mtlb_conv1(1:100)),
1:100, real(ova_conv1(1:100)), 1:100, real(ovs_conv1(1:100)));
legend({'direct','matlab','ova','ovs'},'Location','southwest')


figure
subplot(4,1,1)
plot(real(d_conv1));
title('direct Convolution of x and filter 1');
xlabel('C1[n]');
ylabel('n');
axis tight;

subplot(4,1,2)
plot(real(mtlb_conv1));
title('conv of x and filter 1');
xlabel('C1[n]');
ylabel('n');
axis tight;


subplot(4,1,3)
plot(real(ova_conv1));
title('ova conv of x and filter 1');
xlabel('C1[n]');
ylabel('n');
axis tight;

subplot(4,1,4)
plot(real(ovs_conv1));
title('ovs conv of x and filter 1');
xlabel('C1[n]');
ylabel('n');
axis tight;

x0=300;
y0=-100;
width=1300;
height=800;
set(gcf,'position',[x0,y0,width,height])
```

```
figure
subplot(4,1,1)
plot(real(d_conv2));
title('direct Convolution of x and filter 2');
xlabel('C2[n]');
ylabel('n');
axis tight;

subplot(4,1,2)
plot(real(mtlb_conv2));
title('conv of x and filter 2');
xlabel('C2[n]');
ylabel('n');
axis tight;


subplot(4,1,3)
plot(real(ova_conv2));
title('ova conv of x and filter 2');
xlabel('C2[n]');
ylabel('n');
axis tight;

subplot(4,1,4)
plot(real(ovs_conv2));
title('ovs conv of x and filter 2');
xlabel('C2[n]');
ylabel('n');
axis tight;

x0=300;
y0=-100;
width=1300;
height=800;
set(gcf,'position',[x0,y0,width,height])
end




 function [] = ex3_runtime()
load('sig_x.mat');
load('filter_1.mat');
F1 = xx;
load('filter_2.mat');
F2 = xx;
j=1;
for K = 100:1000:270000
    %ova runtime calculation
    tic
    ova_conv1 = OVA(x, F1, K);
    T = toc;
    time_ova_f1(j) = T ;

    tic
    ova_conv1 = OVA(x, F2, K);
    T = toc;
    time_ova_f2(j) = T ;
    %ovs runtime calculation
```

```
    tic
    ovs_conv1 = OVS(x, F1, K);
    T = toc;
    time_ovs_f1(j) = T ;

    tic
    ovs_conv1 = OVS(x, F2, K);
    T = toc;
    time_ovs_f2(j) = T ;

    %direct runtime calculation
    tic
    direct_conv2 = direct_Convolution(x, F2);
    T = toc;
    time_direct_f2(j) = T;

    tic
    direct_conv1 = direct_Convolution(x, F1);
    T = toc;
    time_direct_f1(j) = T;
    j=j+1;
end


fprintf("min runtime direct f1: %d\n",min(time_direct_f1) )

fprintf("min runtime direct f2: %d\n",min(time_direct_f2) )


K = 100:1000:270000;
%ova plots
[opt_runtime_ova_f1,index_ova_f1] = min(time_ova_f1);
fprintf('optimal runtime ova f1: %d .  K = %d\n',opt_runtime_ova_f1,
K(index_ova_f1))

figure
plot(K, time_ova_f1)
title('ova runtime vs K (conv of x and filter 1)');
xlabel('K');
ylabel('runtime');
axis tight;

[opt_runtime_ova_f2,index_ova_f2] = min(time_ova_f2);
fprintf('optimal runtime ova f2: %d .  K = %d\n',opt_runtime_ova_f2,
K(index_ova_f2))

figure
plot(K, time_ova_f2)
title('ova runtime vs K (conv of x and filter 2)');
xlabel('K');
ylabel('runtime');
axis tight;


%ovs plots
[opt_runtime_ovs_f1,index_ovs_f1] = min(time_ovs_f1);
fprintf('optimal runtime ovs f1: %d .  K = %d\n',opt_runtime_ovs_f1,
K(index_ovs_f1))
```

```
figure
plot(K, time_ovs_f1)
title('ovs runtime vs K (conv of x and filter 1)');
xlabel('K');
ylabel('runtime');
axis tight;

[opt_runtime_ovs_f2,index_ovs_f2] = min(time_ovs_f2);
fprintf('optimal runtime ovs f2: %d .  K = %d\n',opt_runtime_ovs_f2,
K(index_ovs_f2))

figure
plot(K, time_ovs_f2)
title('ovs runtime vs K (conv of x and filter 2)');
xlabel('K');
ylabel('runtime');
axis tight;

%all together plot
figure
subplot(2,1,1)
plot(K, time_ova_f1, K, time_ovs_f1)
title('ova and ovs runtime vs K (conv of x and filter 1)');
xlabel('K');
ylabel('runtime');
legend({'ova','ovs'},'Location','southwest')
axis tight;

subplot(2,1,2)
plot(K, time_ova_f2, K, time_ovs_f2)
title('ova and ovs runtime vs K (conv of x and filter 2)');
xlabel('K');
ylabel('runtime');
legend({'ova','ovs'},'Location','southwest')
axis tight;

end
```

<u>נספח 3: חלק ג תוצאות קוד זמן ריצה:</u>

```
optimal runtime ova f1: 1.128730e-02 .  K = 142100
optimal runtime ova f2: 1.160090e-02 .  K = 142100
optimal runtime ovs f1: 1.178690e-02 .  K = 93100
optimal runtime ovs f2: 1.192600e-02 .  K = 93100
```

```
min runtime direct f1: 5.789180e-02
min runtime direct f2: 5.797440e-02
```