עבודה 3 עיבוד תמונה

נתן דוידוב 211685300

ניקולאי קרושחמל 320717184

1. 2D-Fourier Transform

1.1 Writing your own functions

1.1.1

בסעיף זה נכתוב פונקציה שמממשת את ה-FFT ואת ה-IFFT על פי המשוואות הבאות:

$$F(u+1,v+1)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(m+1,n+1)\cdot e^{-2\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

$$I(m+1,n+1)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u+1,v+1)\cdot e^{2\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

ראשית ניסינו לממש את המשוואות הנ"ל באמצעות לולאה מקוננת אך נתקלנו בזמן ריצה ארוך מאוד ולכן עברנו לדרך אחרת שהיא לבנות בצורה וקטורית את מטריצות ה-DFT ולאחר מכפלה של התמונה במטריצות ה-DFT קיבלנו את ה-FFT כפי שניתן לראות בקוד להלן:

פונקצית FFT:

```
function fft_result = dip_fft2(I)
    [M, N] = size(I);

    u = 0:M-1;
    v = 0:N-1;

    Wn = exp(-2 * pi * 1i * (v.' * v) / N);
    Wm = exp(-2 * pi * 1i * (u.' * u) / M);

    fft_result = Wm * (I * Wn);
end
```

פונקצית IFFT:

```
function ifft_result = dip_ifft2(FFT)
    [M, N] = size(FFT);

    u = 0:M-1;
    v = 0:N-1;

    Wn = exp(2 * pi * 1i * (v.' * v) / N);
    Wm = exp(2 * pi * 1i * (u.' * u) / M);

    ifft_result = Wm * (FFT * Wn) / (M * N);
end
```

1.1.2

בסעיף זה נדרשנו לממש fft shifter, בדיוק כמו שנדרשנו בסרטוט החלפנו בין הבלוקים המתאימים במטריצה כך שנקבל שהתדרים הנמוכים נמצאים במרכז המטריצה החדשה

Figure 1: *dip_fftshift(FFT)*

עשינו זאת ישירות ע"י slicing:

```
function fftshift = dip_fftshift(FFT)
    [M, N] = size(FFT);
    fftshift = FFT;
    fftshift(1:floor(M/2),1:floor(N/2)) = FFT(floor(M/2)+1:M,floor(N/2)+1:N); %switch block 3 to 1
    fftshift(floor(M/2)+1:M,floor(N/2)+1:N) = FFT(1:floor(M/2),1:floor(N/2)); %switch block 1 to 3
    fftshift(floor(M/2)+1:M,1:floor(N/2)) = FFT(1:floor(M/2),floor(N/2)+1:N); %switch block 2 to 4
    fftshift(1:floor(M/2),floor(N/2)+1:N) = FFT(floor(M/2)+1:M,1:floor(N/2)); %switch block 4 to 2
end
```

1.1.3

בסעיף זה קראנו את התמונה beatles.png המרנו אותה ל-grayscale ונרמלנו וגם נציג אותה

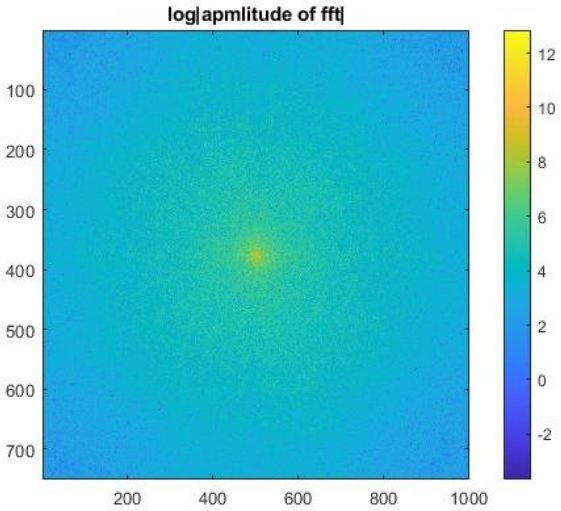
```
%% 1.1.3
img = imread("beatles.png");
img_gray1 = rgb2gray(img);
img_gray_double = im2double(img_gray1);
minImg = min(img_gray_double(:));
maxImg = max(img_gray_double(:));
normalized_gray_img = ((img_gray_double-minImg)/(maxImg-minImg));

figure();
imshow(normalized_gray_img);
title("Normalized grayscaled original image");
```

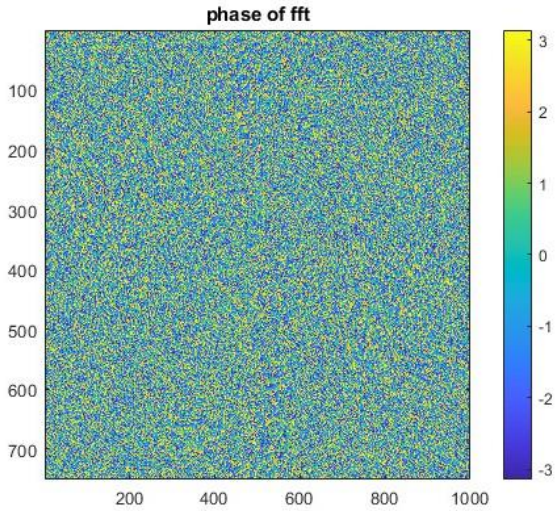
Normalized grayscaled original image

1.1.4

בסעיף זה חישבנו את ה-FFT של התמונה באמצעות הפונקציה שכתבנו בסעיף הראשון ונציג את הלוג של האמפליטודה ואת הפאזה של ה-FFT

```
%% 1.1.4
fft = dip_fft2(normalized_gray_img);
shifted_fft = dip_fftshift(fft);
log_amp_fft = log(abs(shifted_fft));
phase = angle(shifted_fft);
figure();
subplot(1,2,1);
imagesc(log_amp_fft);
title("log|apmlitude of fft|")
colorbar();
subplot(1,2,2);
imagesc(phase);
title("phase of fft")
colorbar();
```





1.1.5

כעת שחזרנו את התמונה המקורית באמצעות שליחת ה-FFT אל פונקציית ה-IFFT שבנינו ונציג אותה באמצעות כך שנציג את החלק הממשי שלה:

```
%% 1.1.5
reconstructed_img = dip_ifft2(fft);
figure();
imshow(real(reconstructed_img));
title("Reconstructed image after ifft")
colorbar();
```

ניתן לראות שהשחזור ממש טוב, התמונה המשוחזרת זהה לתמונה המקורית למראית העין.

1.2 Transformation Properties

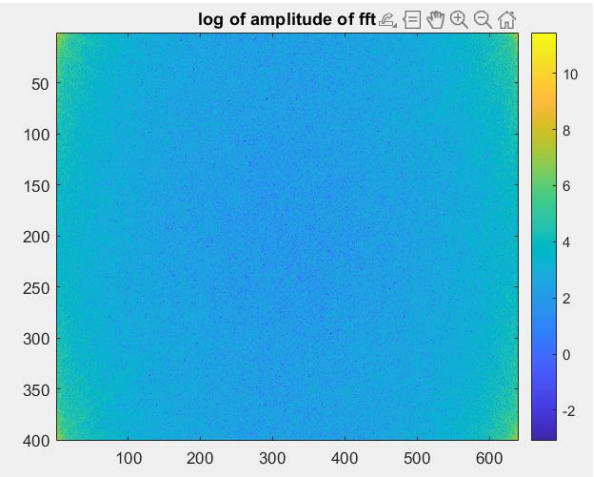
1.2.1 Linearity(Free Willy)

(a) טענו את הקובץ freewilly.mat והצגנו אותו

```
%% (a)
load("freewilly.mat");
figure()
imshow(freewilly);
title("free willy original image")
```

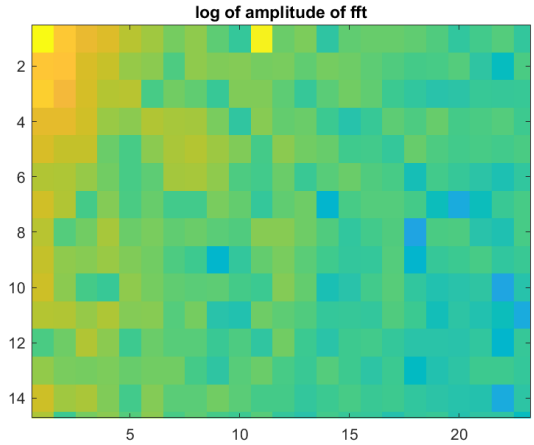
(b) על מנת למצוא את התדר של ה'סורגים' נעשה FFT ונחפש את התדר הכי דומיננטי

```
%% (b)
fft_willy = fft2(freewilly);
figure;
imagesc(log(abs(fft_willy)));
title("log of amplitude of fft")
```



ניתן לראות כי התדרים הדומיננטיים נמצאים בפינות הfft.

נעשה זום-אין לתדרים הנמוכים(לפינה השמאלית העליונה)

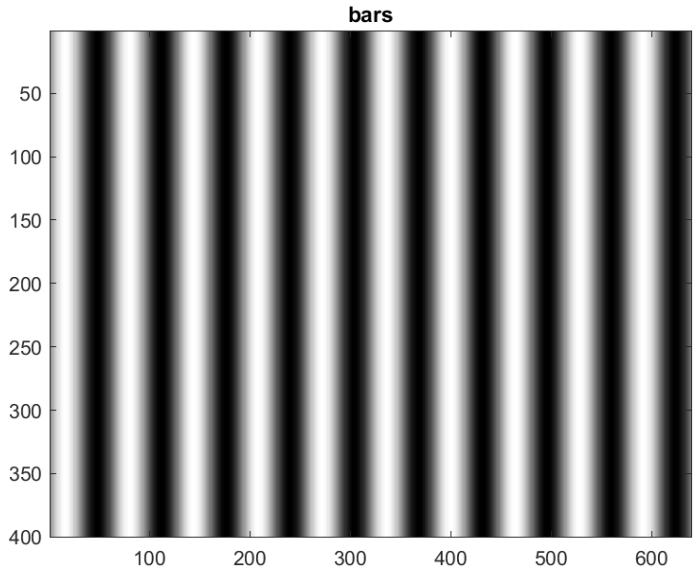


ונבחין כי יש נקודה בהירה בתדר 10 ונבין שזהו התדר של הסורגים מכיוון שהם מאוד בולטים

(c) ניצור וקטור סינוס $0.5 \cdot sin\left(\frac{2\pi f_x}{N}x\right)$ עם התדירות המתאימה $f_x = 10$ ולאחר מכן נכפיל אותו בווקטור עמודה של אחדות על מנת לקבל את המימדים של התמונה המקורית.

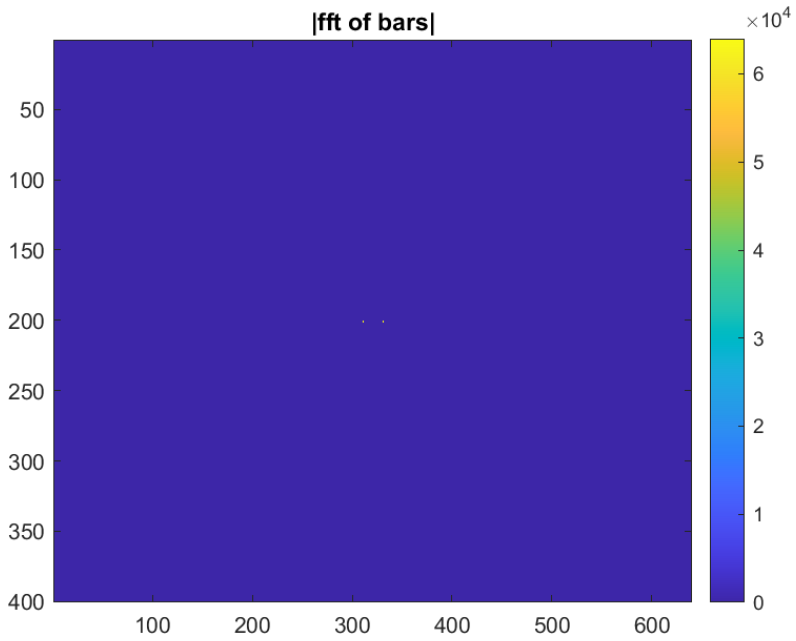
```
%% (c)
x = 1:640;
f_x = 10;
sinus = sin(2*pi*f_x*x/640);
one = ones(400,1);
bars = 0.5*one*sinus;
figure;
imagesc(bars);
colormap("gray");
```

וכמצופה נקבל:

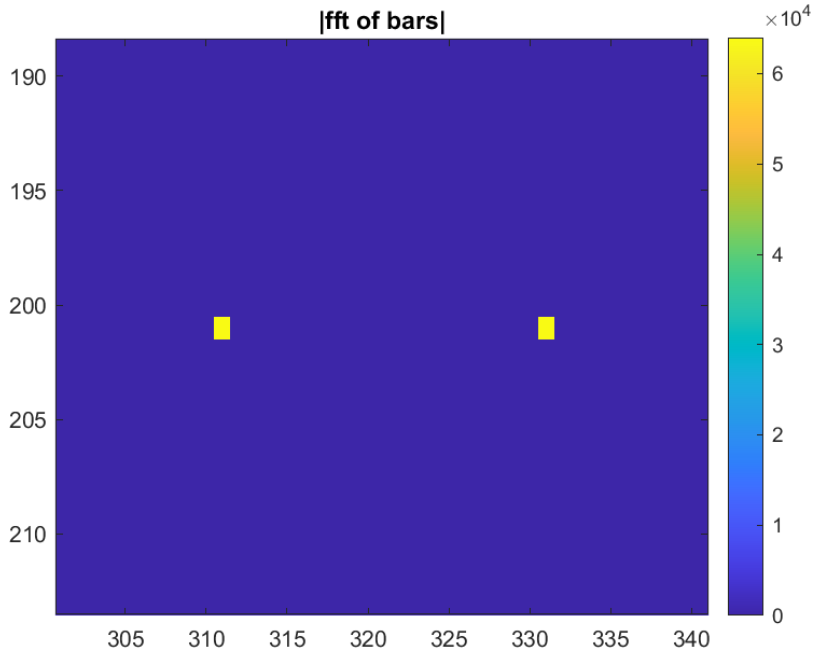


(d) נחשב את הטרנספורם פורייה הדו מימדי של התמונה הנ"ל ונציג את האמפליטודה שלה

```
%% (d)
bars_fft = fft2(bars);
bars_fft_shifted= fftshift(bars_fft);
figure;
imagesc(abs(bars_fft_shifted));
title("|fft of bars|");
colorbar();
```



נעשה זום-אין



וכפי שציפינו, קיבלנו תדר ה-dc אפס ומשני צדדיו במרחק 10 ממנו קיבלנו את התדר של הסינוס שלנו.

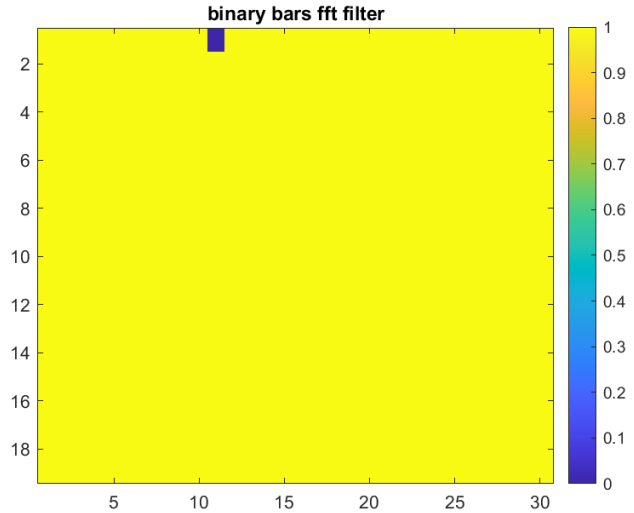
נחשב באופן ידני את הfft של $0.5\sin(\frac{20\pi}{640}x)$ כדי לאמת את התוצאה שקיבלנו:

$$F(0.5\sin(\frac{2*10\pi}{640}x)) = \sum_{y=0}^{399}\sum_{x=0}^{639}\frac{1}{2}\sin(\frac{2*10\pi}{640}x)e^{-2\pi j\left(\frac{y*u}{400}\right)}e^{-2\pi j(\frac{x*v}{640})} =$$

$$= \frac{1}{2}(\sum_{y=0}^{399}e^{-2\pi j\left(\frac{y*u}{400}\right)})(\sum_{x=0}^{639}\sin(\frac{2*10\pi}{640}x)e^{-2\pi j(\frac{x*v}{640})}) = \frac{400*640}{4j}\delta(u)(\delta(v-10)+\delta(v+10))$$

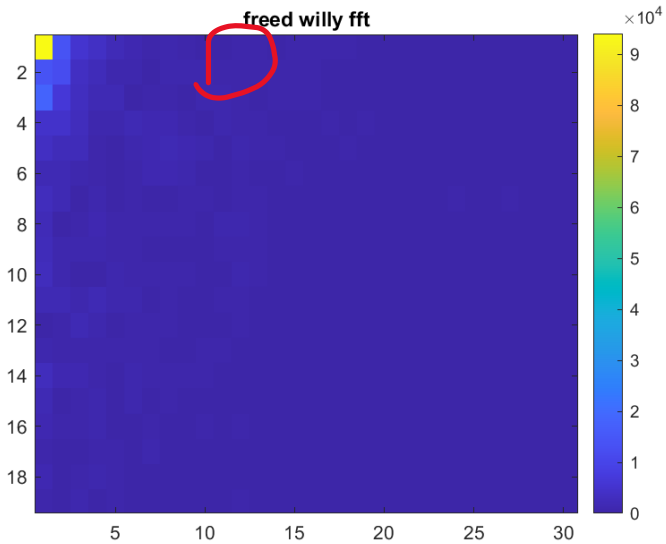
כאשר השתמשנו בתכונות של הספרביליות וחישבנו בנפרד את הfft של ציר y ושל ציר x. בציר y כצפוי קיבלנו דלתא בודדת בראשית (כי y=1) ובציר x קיבלנו שתי דלתאות במרחק של $f_x = 10$, כצפוי מfft של סינוס. בדיוק בהתאם לתמונה למעלה.

(e) ראשית הפכנו את ה-FFT של הסורגים למטריצה בינארית שכולה אחדות, ואפסים בדיוק בתדר של הסינוס, להלן תמונת זום אין שלה:

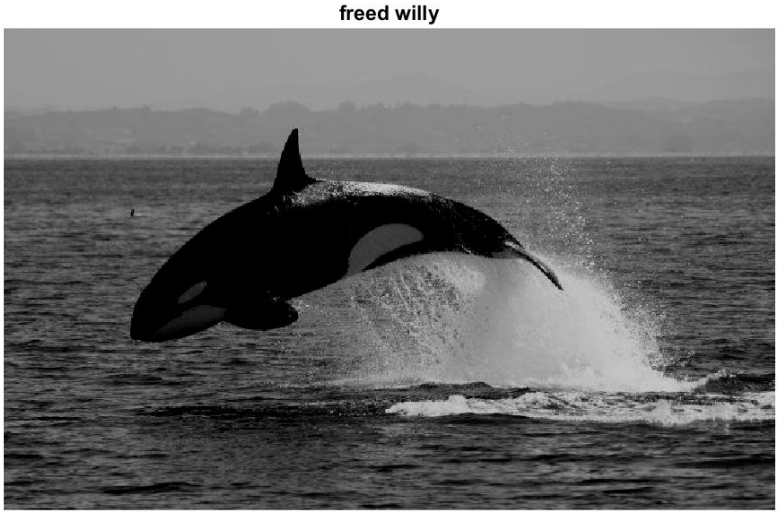


לאחר מכן כפלנו איבר איבר בין מטריצה זו למטריצה ה-FFT של התמונה המקורית של free willy ובכך בעצם איפסנו את התדר של הסורגים.

להלן תמונת זום-אין של המכפלה:



ניתן לראות שאכן התדר התאפס,



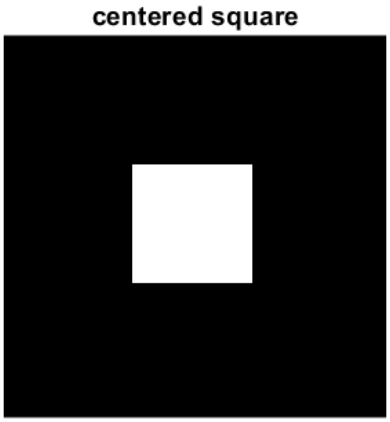
לאחר מכן עשינו IFFT למכפלה הנ"ל וקיבלנו את התמונה כפי שרצינו ללא הסורגים.

להלן הקוד של הסעיף:

```
%% (e)
binary_bars_fft = (bars_fft>1);
binary_bars_fft_filter = 1-binary_bars_fft;
figure;
imagesc(binary_bars_fft_filter);
colorbar();
freed_willy_fft = fft_willy.*binary_bars_fft_filter;
freed_willy = ifft2(freed_willy_fft);
figure;
imagesc(abs(freed_willy_fft));
colorbar();
figure;
imshow(freed_willy);
```

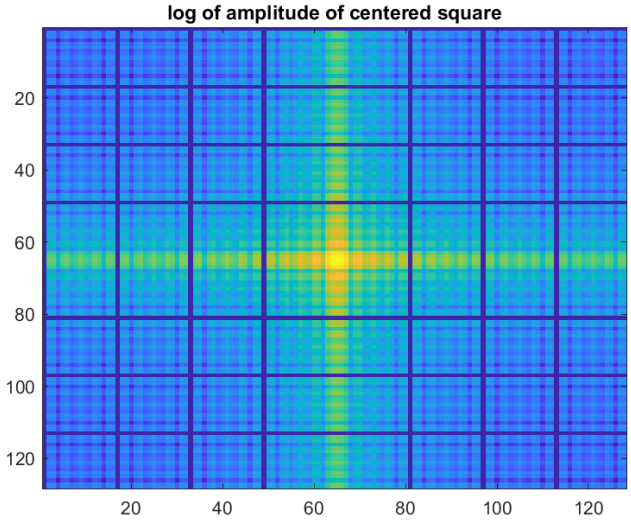
1.2.2 Scaling, translation and separability

(a) אתחלנו מטריצת אפסים בגודל 128x128 והכנסנו ריבוע של אחדות בגודל 40x40 במרכז מטריצת האפסים. הצגנו את התמונה וזה מה שקיבלנו:



זה מה שקיבלנו מכיוון ששחור מייצג את האפסים ולבן את האחדות.

לאחר מכן ביצענו 2D-FFT וקיבלנו:



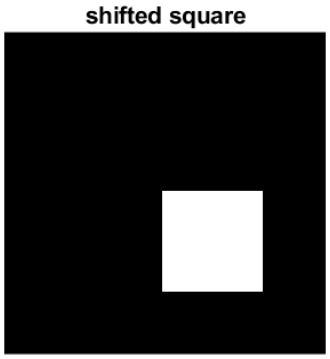
וזה הגיוני מכיוון שחלון במרחב הוא sinc במישור fft ובריבוע שלנו זה בעצם חלון בציר x וחלון בציר y ולכן ה-FFT שקיבלנו הוא sinc דו מימדי.

להלן הקוד:

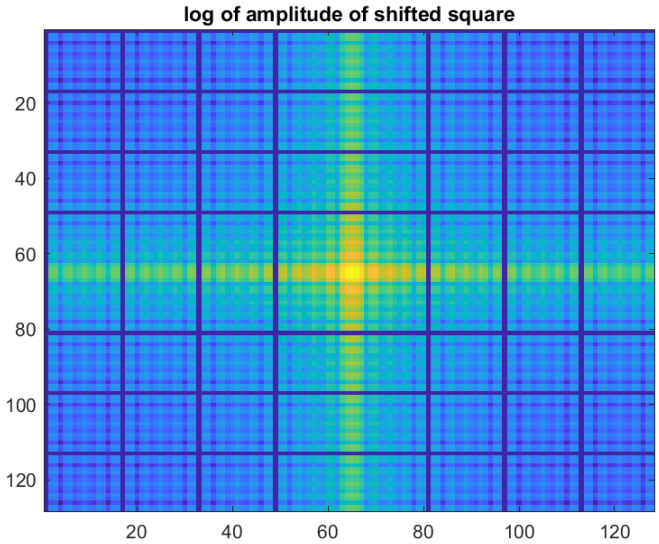
```
%% (a)
square = zeros(128,128);
square(44:83,44:83) = 1;
square_fft = fft2(square);
figure;
imshow(square);
title("centered square");
figure;
imagesc(log(abs(fftshift(square_fft))));
title("log of amplitude of centered square");
```

(b) בסעיף זה נעשה בדומה לסעיף הקודם רק שכאן הריבוע הוא לא במרכז התמונה אלה מוזז,

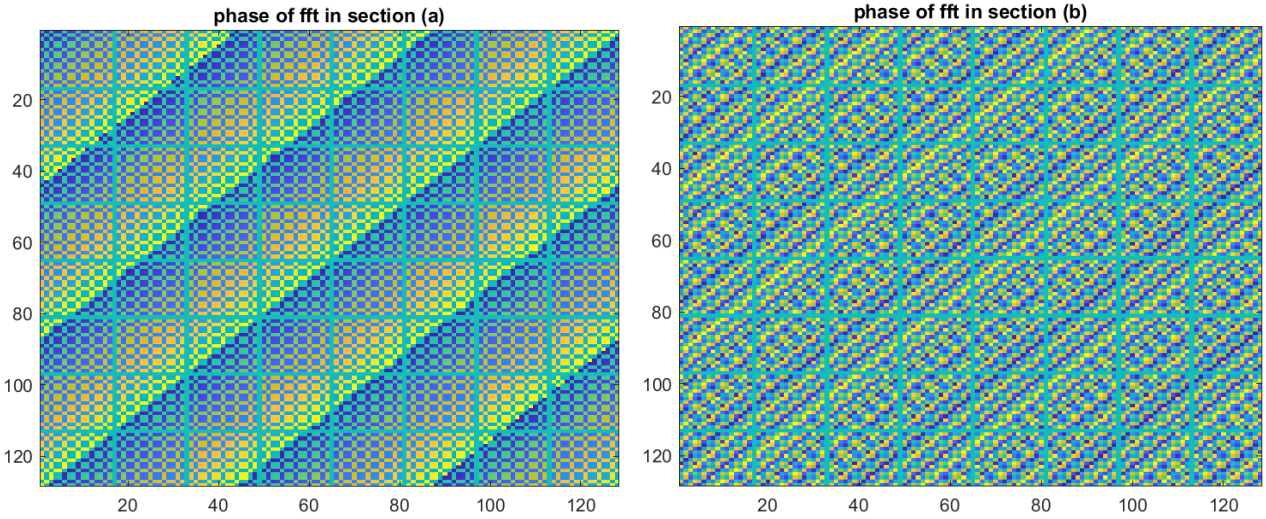
זהו הריבוע שקיבלנו:



נסתכל על ה-fft של התמונה הנ"ל:

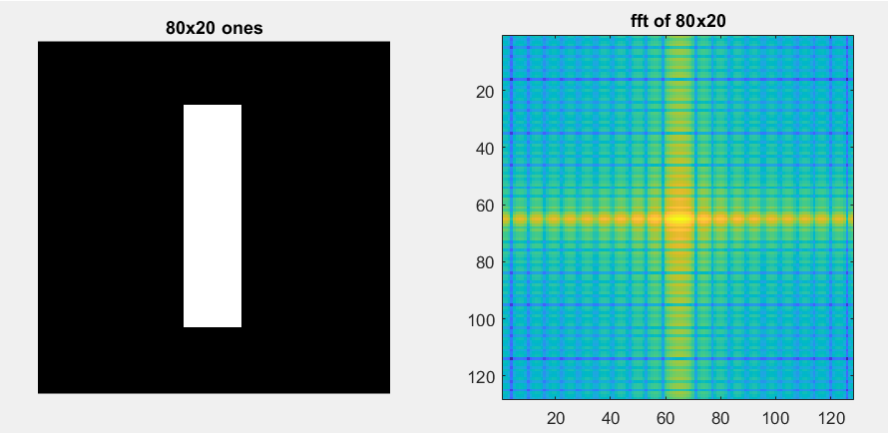


ניתן לראות שאין הבדל בין האמפליטודות של ה-fft של שתי התמונות וזאת מכיוון שהזזה מתורגמת לכפל באקספוננט בתדר ולכן נצפה לראות שינוי דווקא בפאזה, ואכן כך:



(c)

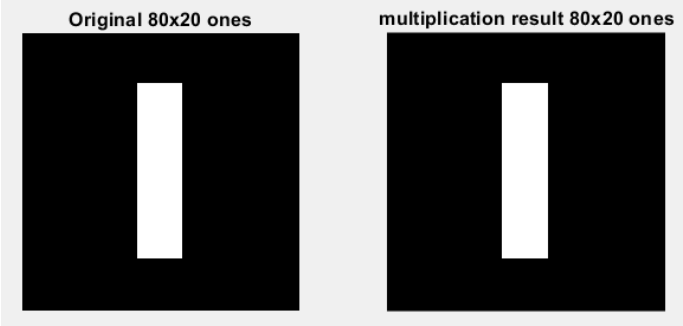
בסעיף זה הכנסנו מלבן בגודל 20x80 של אחדות למרכז מטירצת אפסים בגודל 128x128 ולאחר מכן חישבנו את הfft שלה. להלן התוצאות:



ניתן לראות שהfft של המלבן דומה לfft של המרובע ב(log(magnitude אך הם אינם זהים. כידוע ככל שהחלון במישור אחד רחב יותר כל האונה המרכזית המישור fft תיהיה צרה יותר. כאן ניתן לראות כי חלון רחב יותר בציר Y גורם לfft בתדר המרחבי להיות צר יותר מאשר בתמונה של הריבוע.

(d)

בהחלט ניתן לייצג את התמונה מסעיף (c) בעזרת שני וקטורים ע"י הכפלת וקטור עמודה בוקטור שורה. נבחר וקטור עמודה בגדול 128 שיכיל אחדות באידקסים (24:104) והשאר אפסים, ווקטור שורה בגודל 128 שיכיל אחדות באינדקסים (54:74) והשאר אפסים. הכפלה של וקטור העמודה בוקטור השורה מימין יתן מטריצה בגודל 128x128 שבמרכזה בדיוק אותו מלבן אחדות בגודל 20x80.



(e)

בסעיף זה ממישנו פונקציה פונקציה המחשבת את הfft של תמונה שניתן לייצג בעזרת 2 וקטורים. החישוב נעשה על ידי חישוב fft של כל וקטור בניפרד ואז הכפלת הוקטורים זה בזה באופן המתואר בסעיף קודם לקבלת תמונת הfft המלאה. ניתן לחשב את הfft הדו-מיימדי בשיטה זו מאחר ומטריצות הניתנות להפרדה למכפלה של שני וקטורים חד מיימדיים, כך של שכל וקטור מכיל בתוכו את האינפורמציה על מיימד אחר. כידוע פעולת הfft היא ספרבילית ולכן ניתן לחשב fft על כל ציר בנפרד ואז להכפיל. את הפונקציה מימשנו בעזרת חישוב fft של כל וקטור, שחלוף הוקטור השני שקבלת וקטור שורה והכפלה של שניהם זה בזה.

להלן הקוד:

```
%Natan Davidov 211685300, Nikolai Krokhmal 320717184

function matFFT = sep_fft2(v1,v2)
    v1 = fft(v1);
    v2 = fft(v2);
    v2 = transpose(v2);
    matFFT =v1*v2;
end
```

וניתן לראות שקיבלנו בדיוק את אותה תוצאה: