DSP Project

Or Ben Daniel 213140361

Ido Ben David 325213577

Alon Kenan 323132571

0. Introduction

Calculating the value of d using ido's and alon's id numbers we get:

d= 0.148346148a

d%0.1<0.5 => time decimation

1. FFT – in code: (by doing decimation in time)

FFT ALG

```
function [fout] = FFT_ALG(fin)
    N = length(fin);
    if N == 1
        fout = fin;
    else
        fout = zeros(1, N);
        feven = FFT_ALG(fin(1:2:end));
        fodd = FFT_ALG(fin(2:2:end));
        for k = 1:N/2
            t = fodd(k) * exp(-1i * 2 * pi * (k-1) / N);
            fout(k) = feven(k) + t;
            fout(k + N/2) = feven(k) - t;
        end
    end
end

```

IFFT ALG

```
function [fout] = IFFT_ALG(fin)
    % assume length(F)=2^n
    N = length(fin);
    fout = conj( FFT_ALG(conj(fin)) )/N;
    %N = length(fin);
    %y =FFT_ALG(fin) /N;
    %fout = [y(1), y(end:-1:2)];
end
```

2. Image analysis and processing using 2D DTFT:

a.

$$DTFT_m(x[n,m]) = X\left(e^{j\omega_1}, n\right) = \sum_{m=-\infty}^{\infty} x[n,m]e^{-j\omega_1 m} \rightarrow DTFT_n\left(X\left(e^{j\omega_1}, n\right)\right)$$

$$= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} x[n,m]e^{-j\omega_1 m} * e^{-j\omega_2 n}$$

$so\ we\ get\ what\ we\ need$:

$$X(e^{j\omega_1}, e^{j\omega_2}) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} x[n,m]e^{-j(\omega_1 n+\omega_2 m)}$$

b.

$$X\left(e^{j\omega_1}, e^{j\omega_2}\right) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} x[n,m]e^{-j(\omega_1 n+\omega_2 m)}$$

$$\xRightarrow{*\ 0\le n<N\ ,0\le m<M} \left[X\left(e^{j\omega_1}, e^{j\omega_2}\right)\right]\Big|_{\omega_1=\frac{2\pi k_1}{N},\ \omega_2=\frac{2\pi k_2}{M}} = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x[n,m]e^{-j\left(\frac{2\pi n}{N}k_1+\frac{2\pi m}{M}k_2\right)} = X[k_1,k_2]$$

$*$ — because DFT's definition applies to finite series only.

c.

$$x[n,m] = y[n] * z[m]$$

$$\mathrm{DTFT_{n,m}(x[n,m]) = X\left(e^{j\omega 1}, e^{j\omega_2}\right) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} x[n,m]e^{-j(\omega_1 n+\omega_2 m)} =}$$

$$\mathrm{\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} y[n] * z[m]e^{-j(\omega_1 n+\omega_2 m)} = \sum_{n=-\infty}^{\infty} y[n]e^{-j\omega_1 n}\sum_{m=-\infty}^{\infty} z[m]e^{-j\omega_2 m} =}$$

$$\mathrm{Y\left(e^{j\omega 1}\right) * Z\left(e^{j\omega 1}\right)}$$

d.

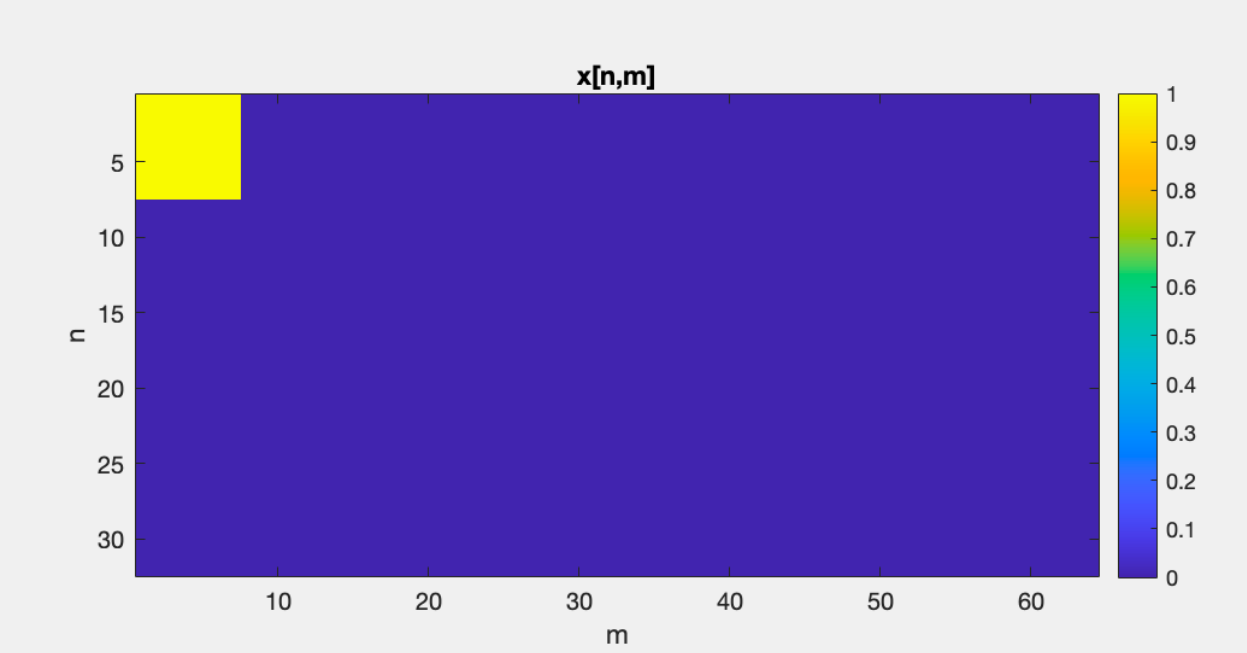
$$X[k_1,k_2]=\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}x[n,m]e^{-j\left(\frac{2\pi n}{N}k_1+\frac{2\pi m}{M}k_2\right)}=$$

$$\sum_{n=0}^{B_1-1}\sum_{m=0}^{B_2-1}1*1*e^{-j\left(\frac{2\pi n}{N}k_1+\frac{2\pi m}{M}k_2\right)}+\sum_{n=B_1}^{N-1}\sum_{m=B_2}^{M-1}0*0*e^{-j\left(\frac{2\pi n}{N}k_1+\frac{2\pi m}{M}k_2\right)}=$$

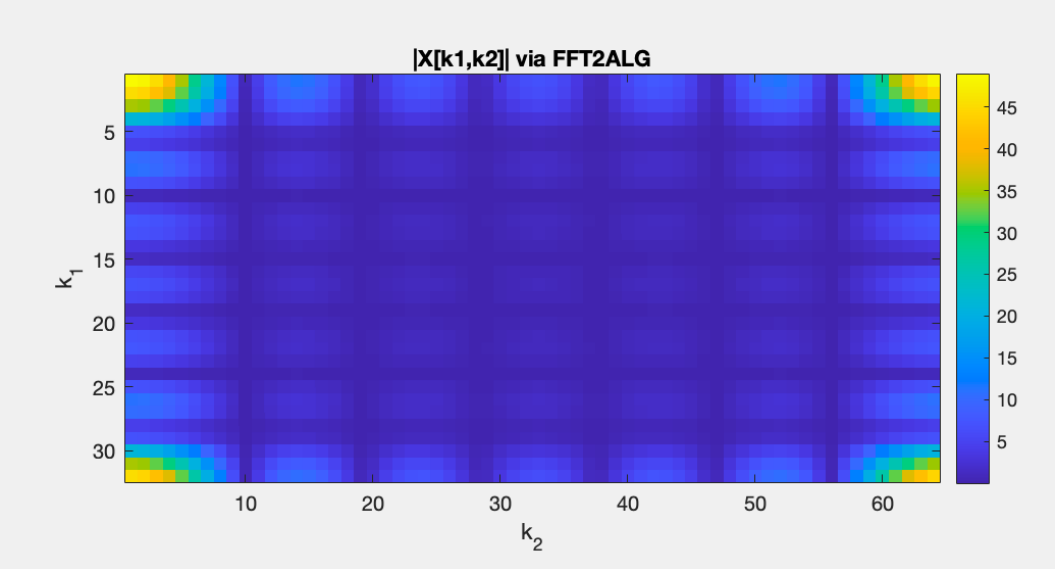
$$\sum_{\mathrm{n}=0}^{\mathrm{B}_1-1}e^{-j\left(\frac{2\pi n}{N}k_1\right)}\sum_{\mathrm{m}=0}^{B_2-2}e^{-j\left(\frac{2\pi m}{M}k_2\right)}=\frac{e^{-j\left(\frac{2\pi B_1}{N}k_1\right)}-1}{e^{-j\left(\frac{2\pi}{N}k_1\right)}-1}*\frac{e^{-j\left(\frac{2\pi B_2}{M}k_2\right)}-1}{e^{-j\left(\frac{2\pi}{M}k_2\right)}-1}$$

$$\xrightarrow{*B_1,B_2=7}X[k_1,k_2]=\begin{cases}\dfrac{e^{-j\left(\frac{2\pi}{N}7k_1\right)}-1}{e^{-j\left(\frac{2\pi}{N}k_1\right)}-1}*\dfrac{e^{-j\left(\frac{2\pi}{M}7k_2\right)}-1}{e^{-j\left(\frac{2\pi n}{M}k_2\right)}-1}, & k_{1,2}\neq 0\\ 7*\dfrac{e^{-j\left(\frac{2\pi}{M}7k_1\right)}-1}{e^{-j\left(\frac{2\pi}{M}k_1\right)}-1} & ,k_2=0,k_1\neq 0\\ 7*\dfrac{e^{-j\left(\frac{2\pi}{N}7k_2\right)}-1}{e^{-j\left(\frac{2\pi}{N}k_2\right)}-1} & ,k_1=0,k_2\neq 0\\ 49 & ,k_1=k_2=0\end{cases}$$

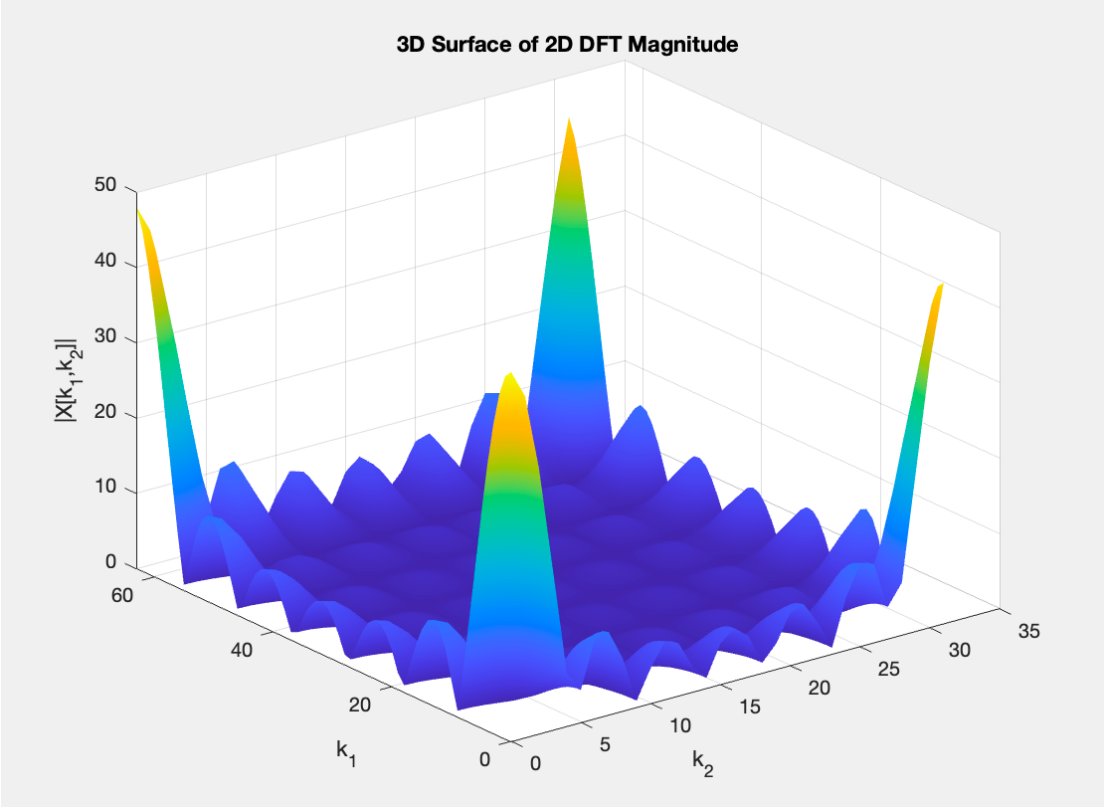
Photo of original signal



Plog of the magnitude of the transformed signal

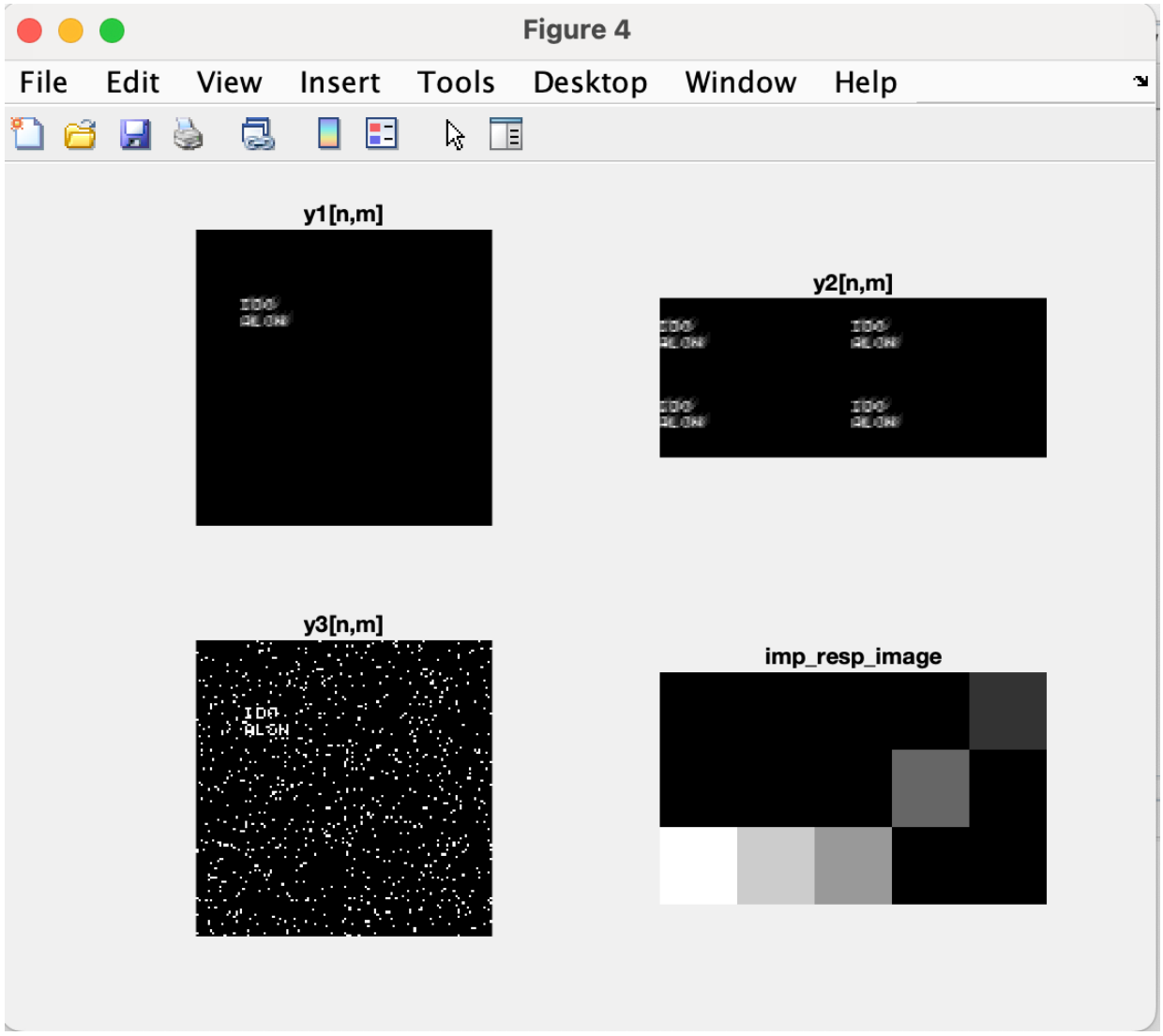


3D plot of the transformed signal



e.

All 4 images generated in one figure



f.

The code for this part

```
%part 1
pad_h0 = zeros(6,1);
for i = 1:6
    if i< 4
        pad_h0(i) = h0(i);
    else
        pad_h0(i) = 0;
    end
end
pad_H0 = fft(pad_h0);
H0 = zeros(4,1);
for i = 1:3
        H0(i) = pad_H0(i);
end
H0(4) = pad_H0(5);

disp(H0)
```

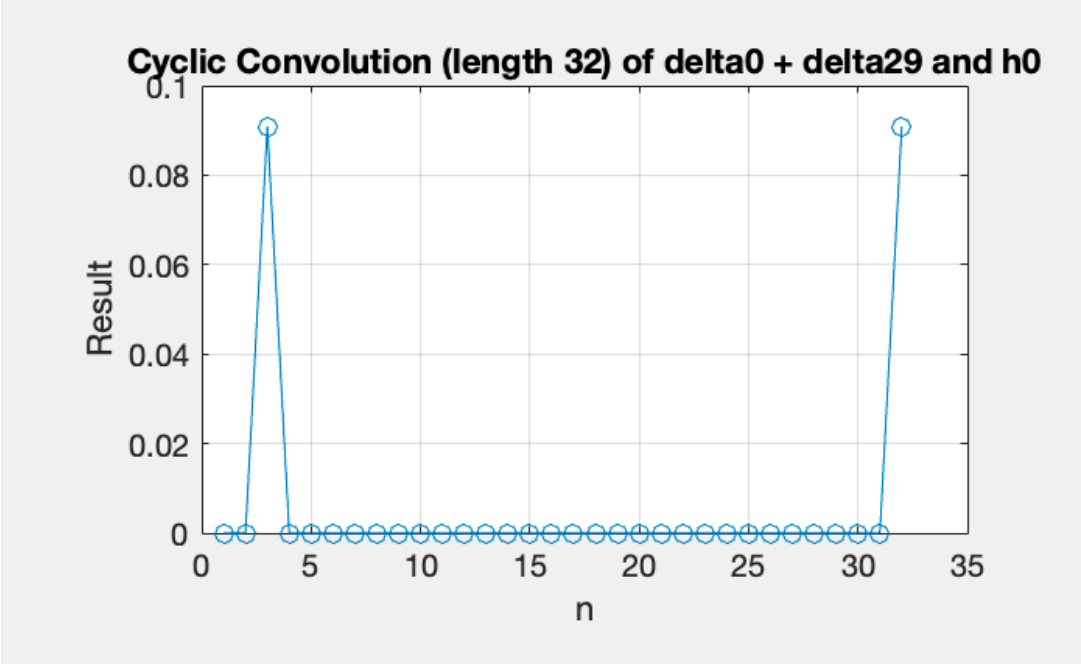
$H_0(e^{j\omega})$ evaluated at $0, \frac{2\pi}{6}, 2\frac{2\pi}{6}, 4\frac{2\pi}{6}$

```
  0.0909 + 0.0000i
 -0.0455 - 0.0787i
 -0.0455 + 0.0787i
 -0.0455 - 0.0787i
```

g.

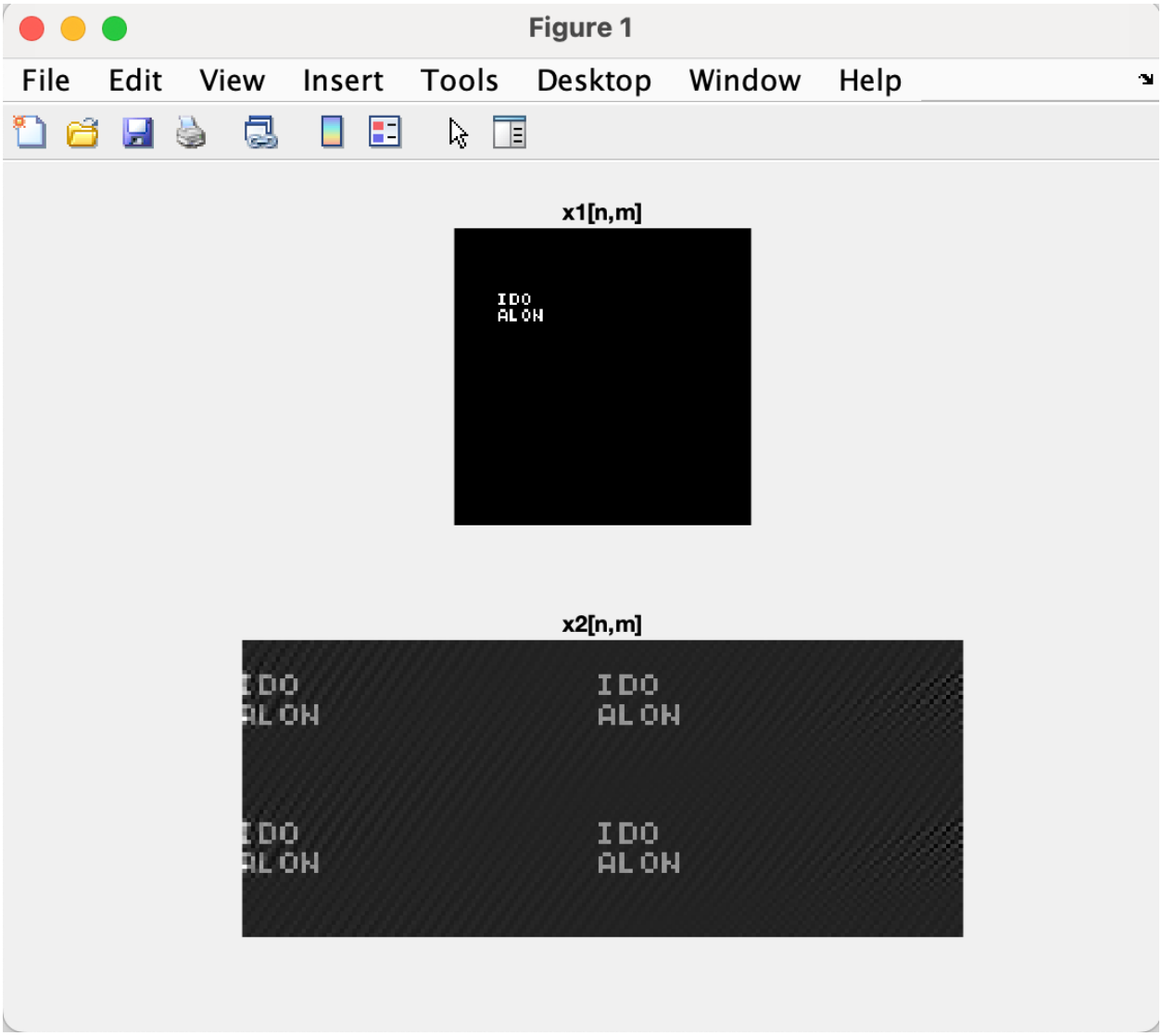
The code for this part

```
% %part 1
% cyclic convelution - by fft
pad1_h0 = zeros(32,1);
for i = 1:32
    if i< 4
        pad1_h0(i) = h0(i);
    else
        pad1_h0(i) = 0;
    end
end

w = zeros(32,1);
w(1) = 1;
w(30) = 1;
H0_pad1 = fft(pad1_h0);
W = fft(w);
Z = H0_pad1.*W;
z = ifft(Z);
n = 1:1:32;

plot(n, z);
title('cyclic convolution (32) of w and h0');
xlabel('n');
ylabel('w*h0');
grid on;
```



h.

Reconstructed photos



i.

(a) In the first case, zero-padding was added to the original image before applying linear convolution. Although the full convolution result was size 133 * 133 thus longer then the original size of the matrix, only the central part — matching the original image size — was kept. Because of the zero-padding, this cropped image still included all the important data, and nothing critical was lost. Also we implemented linear convolution using circlic convlution, by using circlic convlution on 2 zero padded matrixes h_zeropad[n,m] and y2[n,m]. in class we learned that to implement linear convolution using circlic convlution on two function sized N and L, one should zero pad each one to the size of N+L-1 and do circlic convolution between them.
in our case we padded both function only to size N = 128, the reason that it works is that y1 is already zero padded with at least 20 zeros on all sides, thus we are actualy imlpmenting a linear conv between two matrixes y1 sized 126 *124 and h sized 3*5. Because of the zero padding the data lost is not important, As a result, the reconstruction came out clean and accurate.

(b) In the second case,the way image y2 was created is: x was duplicated four times without any zero-padding to create x2. Then Linear convolution was again applied between x2 and h, and then cropped back to the original image size to create y2. But since there was no padding to protect the 2d signal's edges, some important parts of the convolution were likely cut off. This led to a less clean reconstruction, with visible artifacts and distortions in the result.

It's also important to note that in the reconstruction we performed — which used cyclic convolution — the result was almost equivalent to regular linear convolution. But because there is no zero padding to x2 in reference to the original image, the circlic convolution might not reasult in a perfect linear convolution.

j. (bonus)

We can suggest a way to resolve the variance in the image $x_2[m,n]$ (maybe this is not the intended way, but it will work) is a solution that is being used in communication. Basically, if we look at the original image and the distorted image $x_2[m,n]$ we can see that the variance is mainly in the color of each pixel, instead of being in the color white or black it varies between dark gray and light gray because of the distortion caused by the reconstruction. We can suggest some sort of a "MAP rule" that will change the color of each pixel to either black or white according to which color it is closest to.

To keep it simple, lets assume black is 0 and white is 1, and all the colors in between are those in picture $x_2[m,n]$. Then such function can be suggested:

$$\Phi_{\text{MAP}} = \begin{cases} 0 \; if \; pixel \; color < 0.5 \\ 1 \; if \; pixel \; color > 0.5 \end{cases}$$

And in the case that the pixel color is exactly 0.5 we can choose uniformly at random between 1 and 0. (pixel color $\sim U\{0,1\}$)

Thus, we can assure that the color of each pixel will be mapped to only black or white and the distortion in the image color will be fixed.

3. Create and analyze speech signals

a.

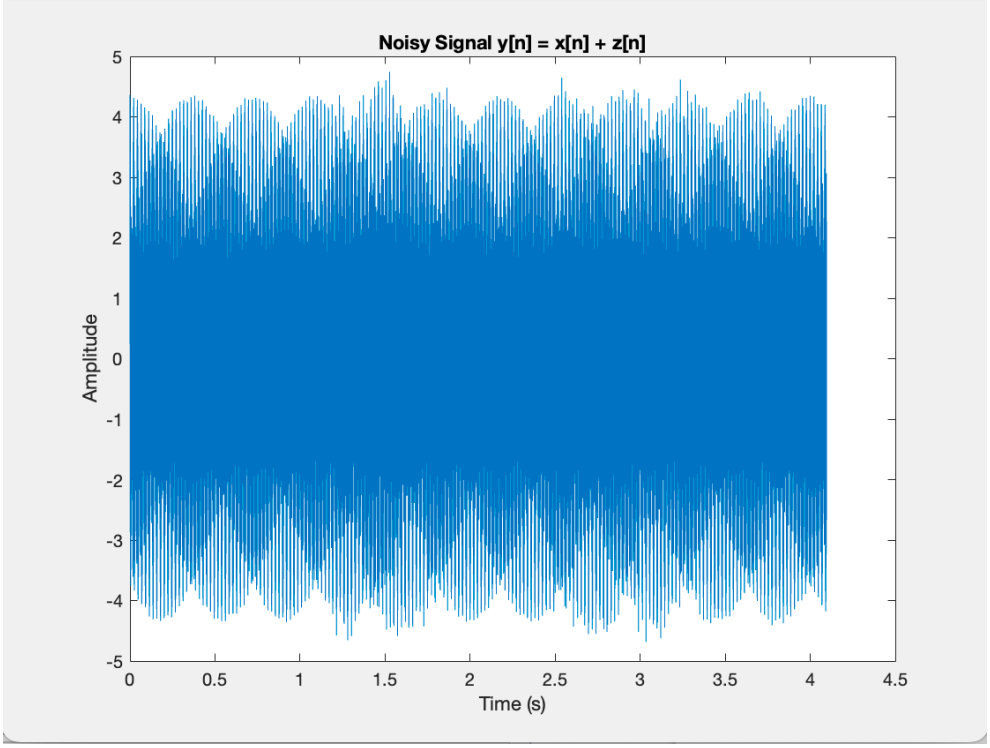
$$z_2[n] = z[2n] = 10\sqrt{P_x}[\cos(\omega_1 2n) + \cos(\omega_2 2n) + \cos(\omega_3 2n)], \quad n = 0, \dots, \frac{N}{2} - 1, else: 0$$

$$Z_2(e^{jw}) = Z\left(e^{\frac{jw}{2}}\right)$$

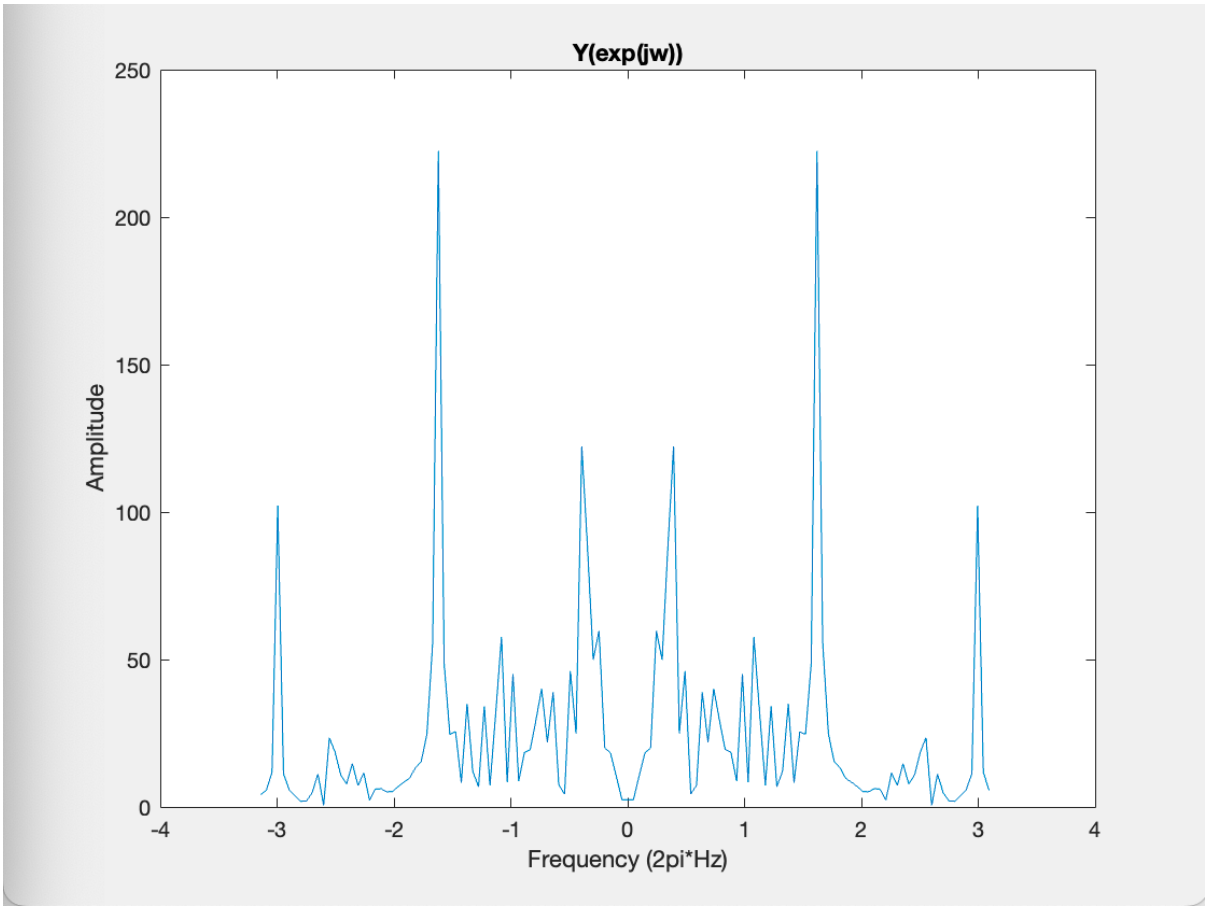
$$\sum_{n=-\infty}^{\infty} z_2[n] * e^{-jwn} = \sum_{n=-\infty}^{\infty} z[2n] * e^{-jwn} = \sum_{n=0}^{\frac{N}{2}-1} z[2n] * e^{-jwn} =$$

$$\sum_{n=0}^{\frac{N}{2}-1} 10\sqrt{P_x}[\cos(\omega_1 2n) + \cos(\omega_2 2n) + \cos(\omega_3 2n)] * e^{-jwn}$$

c.



d.



e.

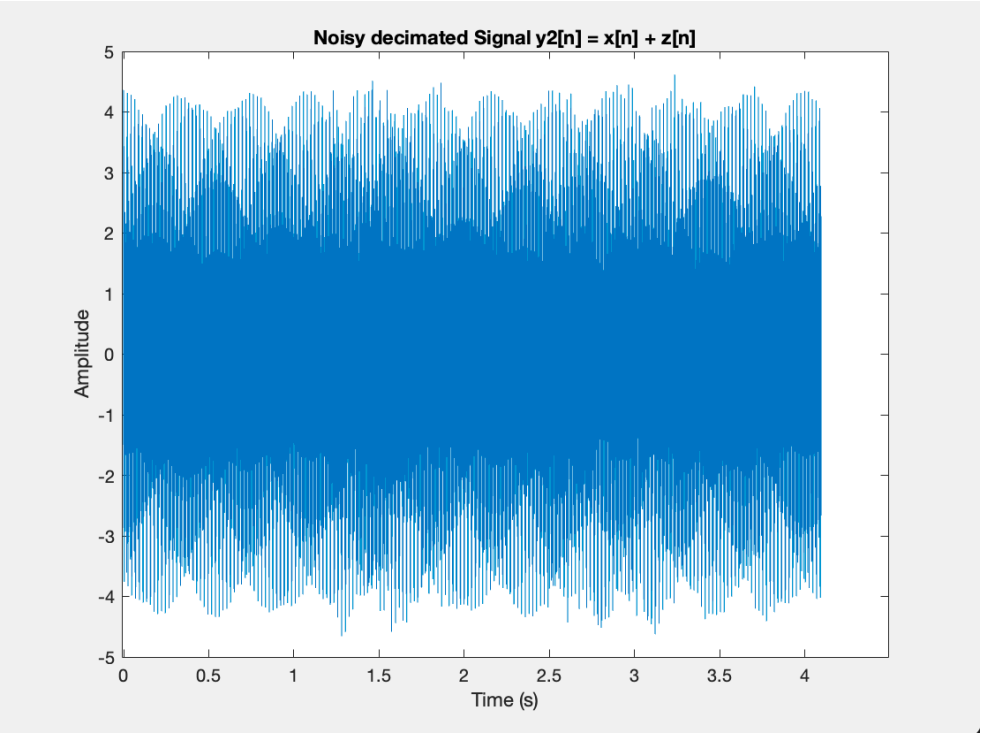
$$z_2[n] = z[2n] = 10\sqrt{P_x}[\cos(\omega_1 2n) + \cos(\omega_2 2n) + \cos(\omega_3 2n)], \quad n = 0, \dots, \frac{N}{2} - 1, else: 0$$

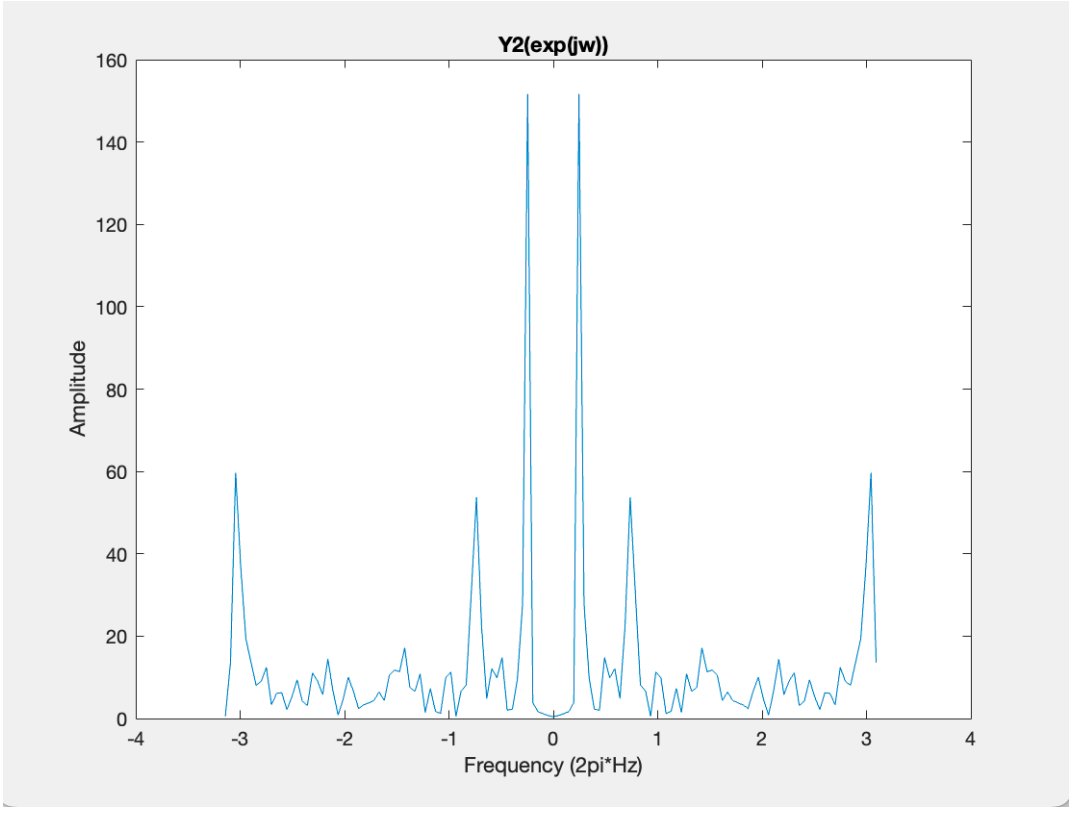
$$Z_2(e^{jw}) = Z\left(e^{\frac{jw}{2}}\right)$$

$$\sum_{n=-\infty}^{\infty} z_2[n] * e^{-jwn} = \sum_{n=-\infty}^{\infty} z[2n] * e^{-jwn} = \sum_{n=0}^{\frac{N}{2}-1} z[2n] * e^{-jwn} =$$

$$\sum_{n=0}^{\frac{N}{2}-1} 10\sqrt{P_x}[\cos(\omega_1 2n) + \cos(\omega_2 2n) + \cos(\omega_3 2n)] * e^{-jwn}$$

f.


Noisy decimated Signal y2[n] = x[n] + z[n]
Amplitude
Time (s)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5
4


Y2(exp(jw))
Amplitude
Frequency (2pi*Hz)
160
140
120
100
80
60
40
20
0
-4
-3
-2
-1
0
1
2
3
4