Rüzgar Ayan 21801984

2020-2021 SPRING EEE391 MATLAB Assignment 2 Report

I have chosen the image "couple.bmp". The original picture and the results with M=11,31,61 are shown below:

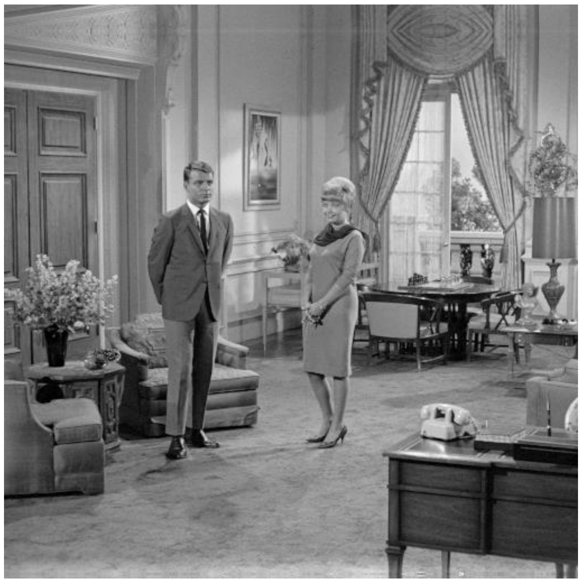


Figure 1. Original Picture

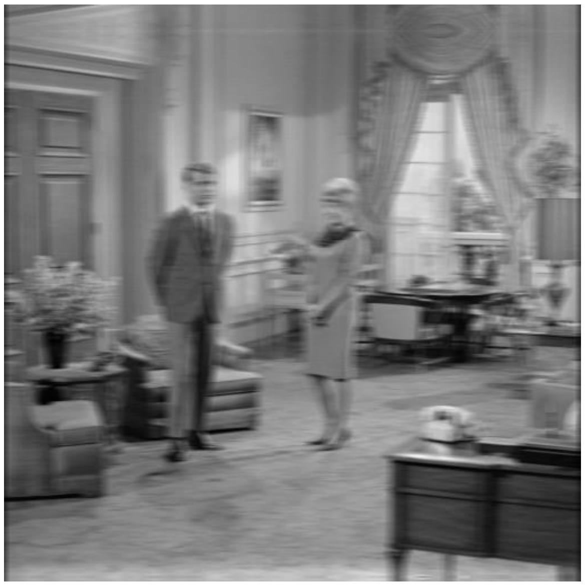


Figure 2. Result with M=11

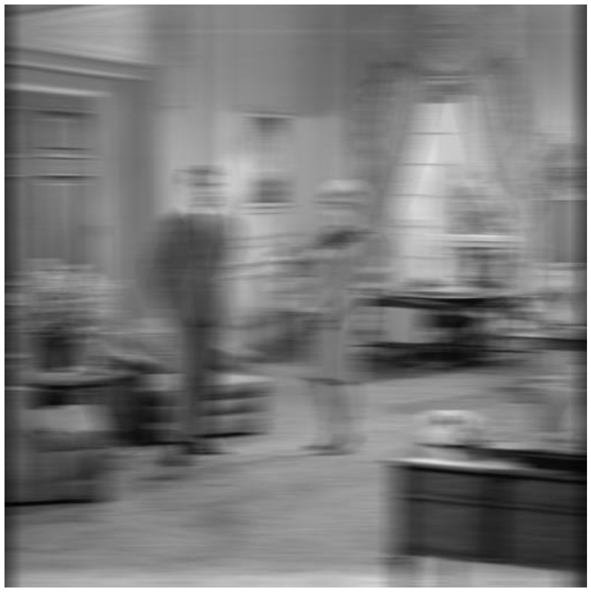


Figure 3. Result with M=31

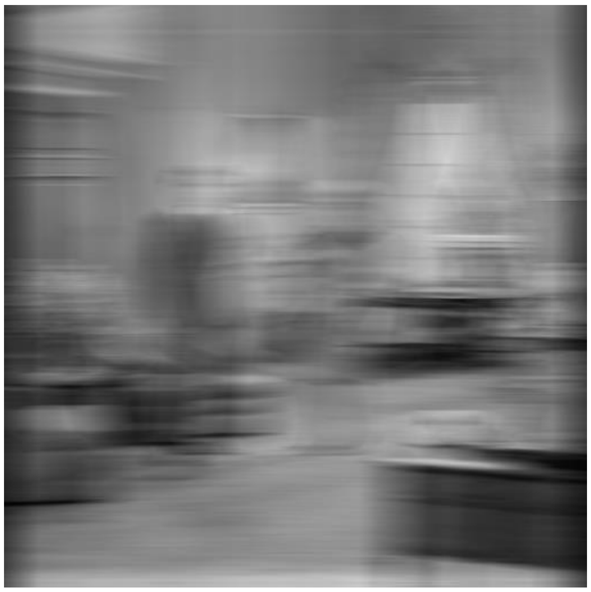
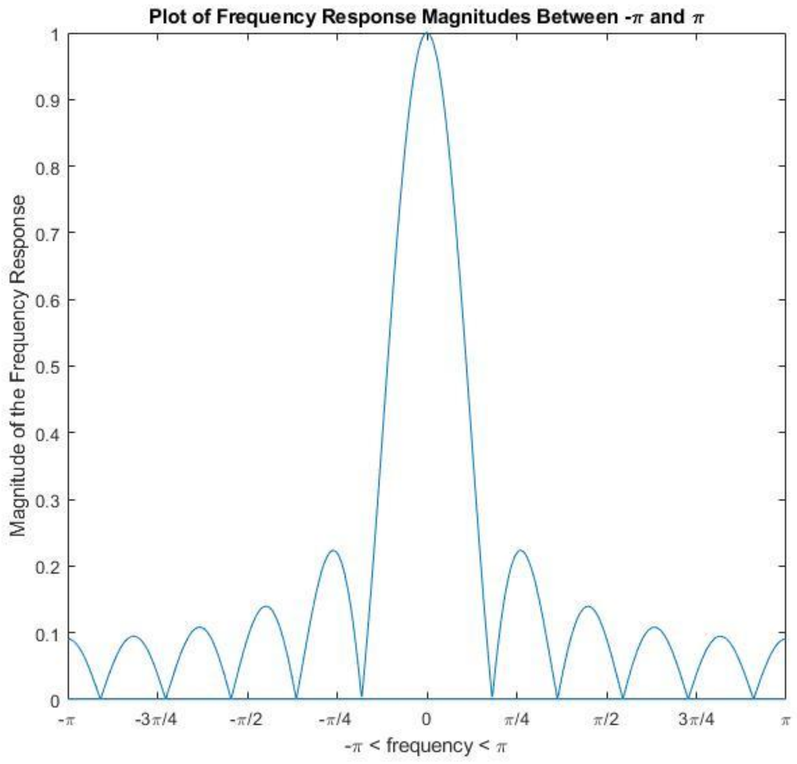


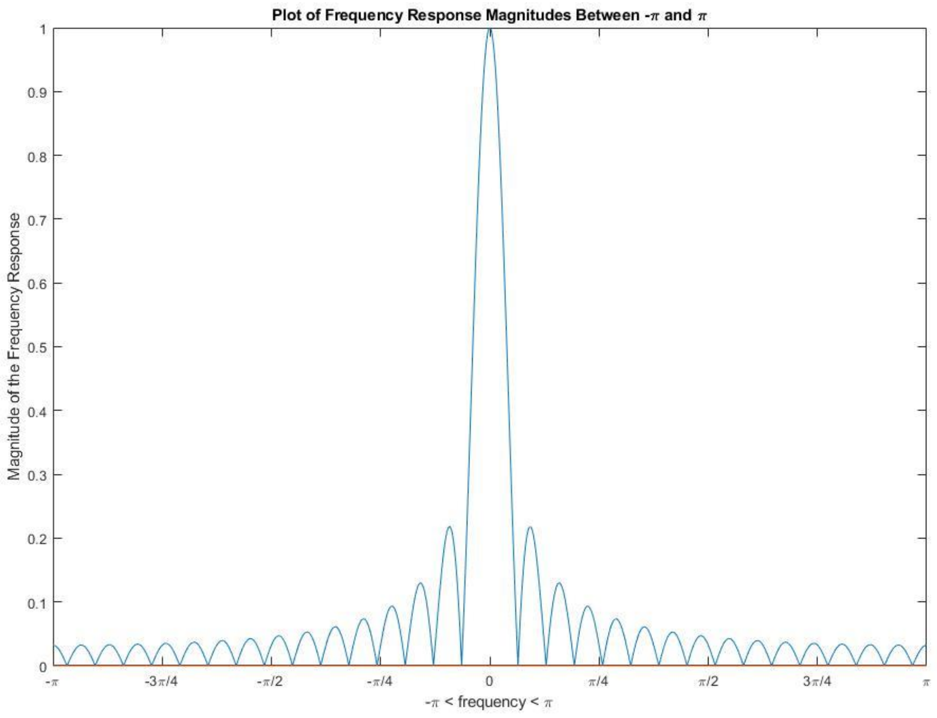
Figure 4. Result with M=61

Magnitude of The Frequency Response Function

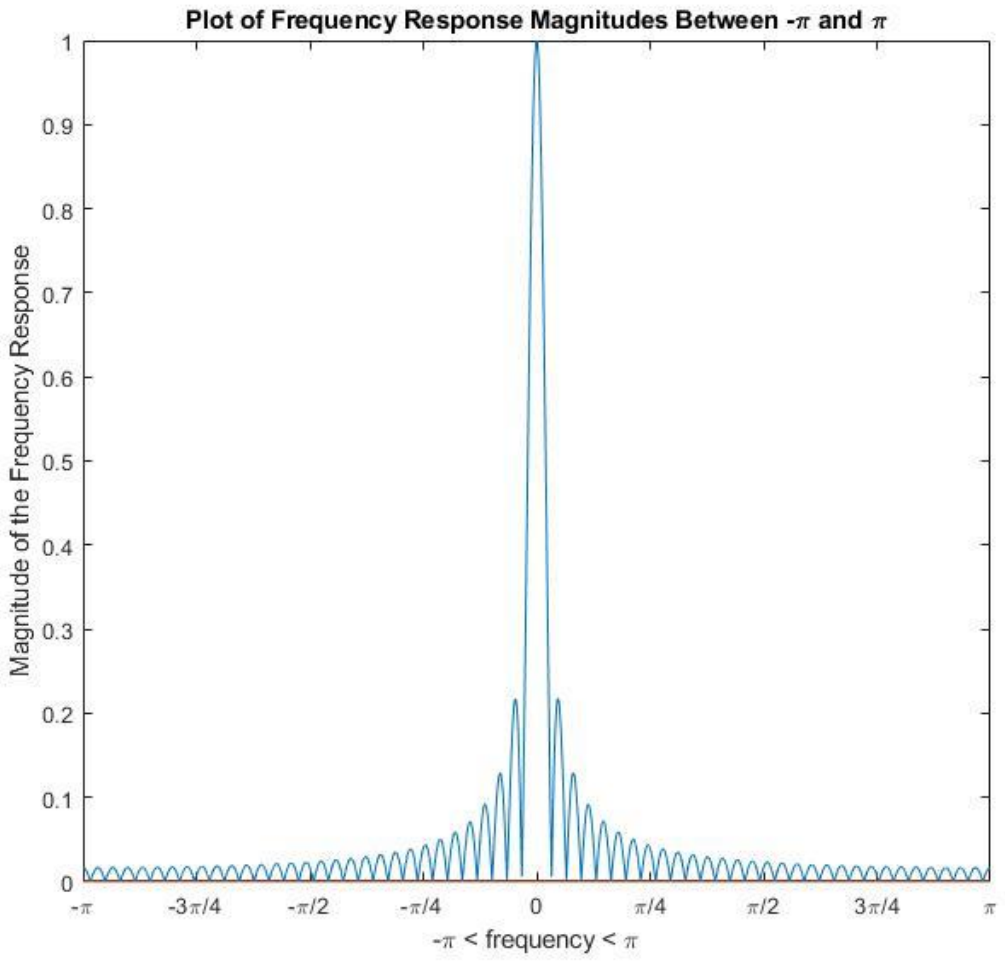
With M = 11



With M = 31



With M = 61



Plot of Frequency Response Magnitudes Between -π and π
Magnitude of the Frequency Response
-π < frequency < π
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-π
-3π/4
-π/2
-π/4
0
π/4
π/2
3π/4
π

Implementation Details

To account for the values lying outside of the image, I have added the right amount of zeroes.

```
m = 31;
mOver2 = (m - 1)/2;

J = [zeros(512, mOver2) , J, zeros(512, mOver2)];
```

After that, I easily apply the filter in the following loop by summing over the M pixel values and the dividing all the result by M.

```
res = zeros(512, 512);
for i = 1:512
    for j = 1:512
        res(i,j) = 0;
        for k = 0:(m - 1)
            res(i,j) = res(i,j) + J(i, j + k);
        end
    end
end
res = res / m;
```

For finding the frequency response, I have divided the range [-pi, pi] into 1000 for smooth plot results. Then, I have used the formula for the frequency response where the coefficients are all 1/M.

```
omegaHat = linspace(-pi, pi, 1000);
freqResponse = zeros(1000);

for a = 1:1000
    freqResponse(a) = 0;
    for b = -mOver2:mOver2
        freqResponse(a) = freqResponse(a) + exp(-1.0i *
omegaHat(a) * b);
    end
    freqResponse(a) = freqResponse(a) / m;
end
```

(i) How can you describe the visual effect created by the filter?

As it is expected from the filter equation, this filter averages the pixel values. Visually, this looks like a blurred image. And also because of the averaging, the edges in the image gets softer.

(ii) What happens to the details in the image? How is this affected by changing the value of *M*?

Details get harder to see, or we can also say that the details are lost as we do this averaging. As M increases, details get even harder to see. As an example to this detail loss, images become like when someone normally wearing glasses takes off their glasses.

(iii) Compare the visual effect in the dimension *n* versus the dimension *m* (vertical versus horizontal)?

This distinction is hard to see for small M. But for example when M = 61 in the images before, this distinction is clearly visible. Images become as if someone **stretches** in the left-right direction.

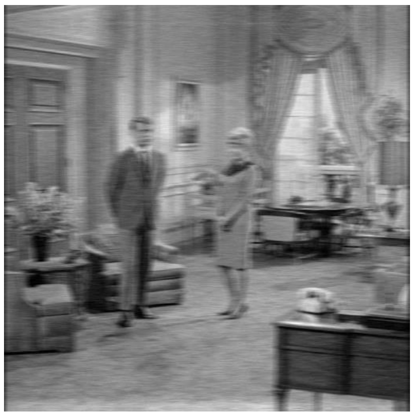
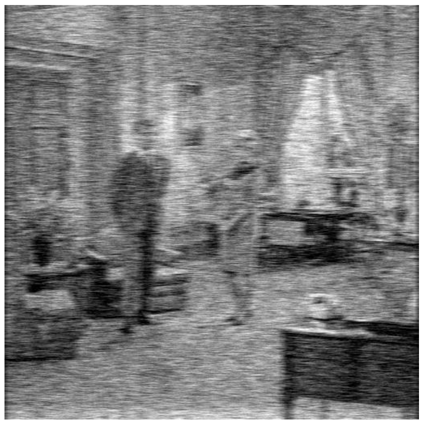
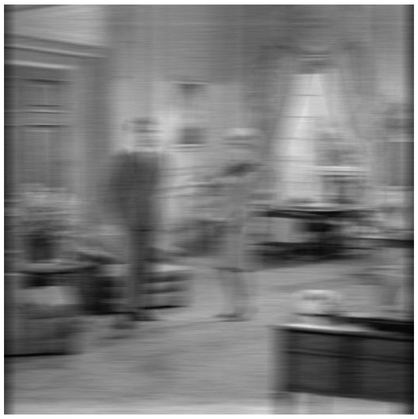
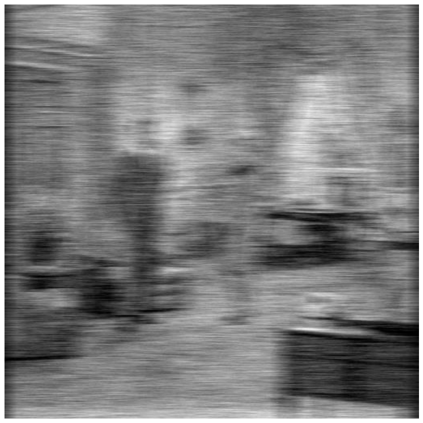
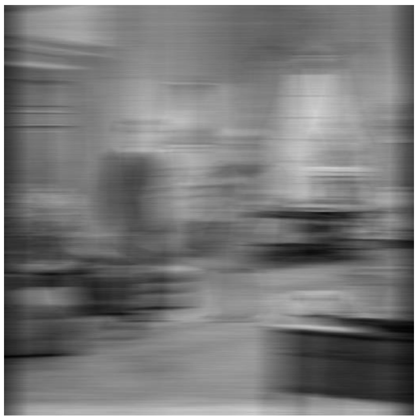
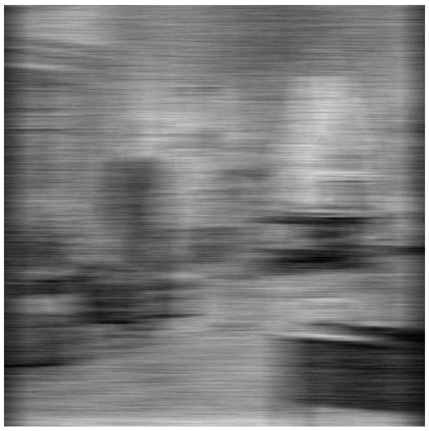
(iv) Recall that a signal can be written as the weighted sum of cosine/sines of different frequencies. Referring to the magnitude of the frequency response, describe the effect of the filter in terms of what happens to the different frequency components in the image. How is this affected by changing the value of *M*?

For all the M values, the magnitude of the frequency response is the maximum at the frequency 0. So, the constant components in the image are exactly passed from the filter. Then, frequency response becomes 0 in many points. The number of these 0 points increase as M increases.

(v) Comment on what happens close to the edges of the image? How is this affected by changing the value of *M*?

Since we have assumed that the pixels out of the image to have value 0, the pixels at the edge of the image will take into account these 0 values. So when averaging with these 0's too, the result will become smaller and that will make the pixels darker in the left and right edges. As M increases, the amount of affected pixels in these edges also increases.

Images with Noise Added

	c=0.2	**C=1**
M=11		
M=31		
M=61		

(vi) Does averaging help to reduce/eliminate the noise? How is this affected by changing the value of *M*?

Yes, averaging definitely reduces the noise and even eliminates it if the M value is high enough. As it can be seen from the images above, for the case c = 0.2, we can still see the noise when M = 11. But when M = 31 or 61, the noise is undetectable.

When there is more noise, as in the case c = 1, these values of M is not enough since we can still slighty notice the noise in M = 61.

(vii) Is there an undesirable side-effect of eliminating the noise in this manner? How is this affected by changing the value of *M*?

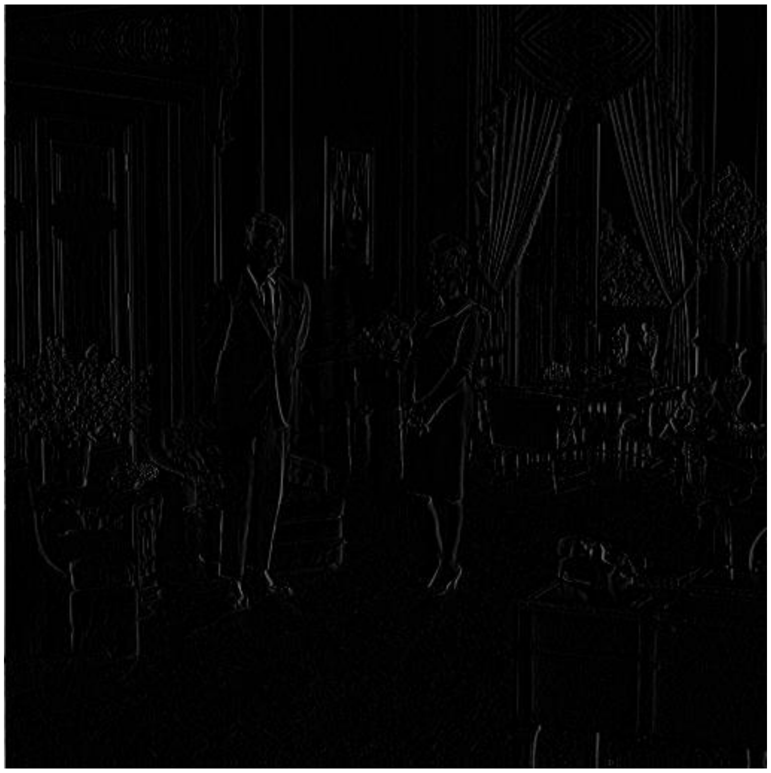
As discussed in **(ii)**, averaging causes the details to be lost. So, as we increase the value of M to eliminate the noise, the more details will be lost.

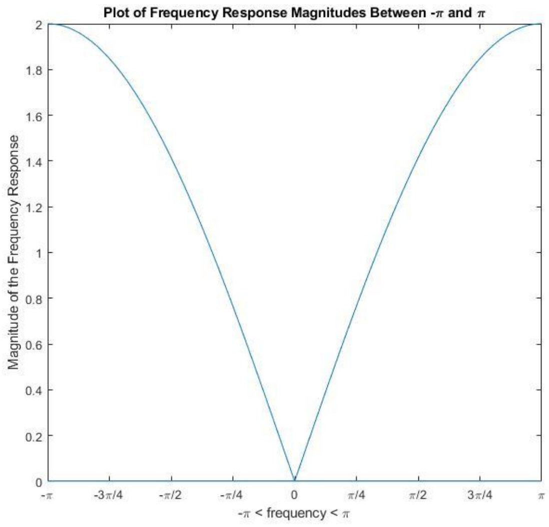
(viii) What do you think is the best choice of *M* here?

It depends on the value of c and and also how much detail loss can be tolerated. According to the images in the previous page, I would **personally** choose M = 11 for c = 0.2 because it eliminates almost all the noise without losing so many details. And I would choose M = 31 for c = 1 because even though noise is noticable there, it is not worth losing so much details with M = 61.

PART 2)

The following mostly-black image is the result of the first-different filter. The plot next to it is the magnitude of the frequency response for this filter.




Plot of Frequency Response Magnitudes Between -π and π
Magnitude of the Frequency Response
-π < frequency < π
-π
-3π/4
-π/2
-π/4
0
π/4
π/2
3π/4
π
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Implementation of this filter was fairly easy. The following two lines of code were enough.

```
J = [zeros(512, 1) , J];
res = J(:,2:513) - J(:,1:512);
```

In the first line, I add a zero column to the matrix to account to represent the zeroes lying outside the actual matrix. Then, I shift the image matrix by 1 unit and subtract it from the itself to get the result.

Computing the frequency response was almost the same as in Part 1) but just using different coefficients.

(i) How can you describe the visual effect created by the filter?

The resulting image was mostly black with some small lighter zones. These lighter zones appear at the boundaries of objects or bodies. In this sense the output was similar to the output of an edge detection algorithm.

(ii) Compare this to what you would theoretically expect it to be.

Since the first-differencer filter is intuitionally equivalent of finding the derivative (or change) in some point, I would theoretically expect it to highlight the pixels where there is a large change in the pixel values. And these points where there is a large change are exactly the boundaries of the bodies in the image as in **(i)**.

(iii) Compare the visual effect in the dimension *n* versus the dimension *m* (vertical versus horizontal)?

Assume that there was a rectangle box in the image, since we are applying the filter in horizontal axis, only the left and right edges of this box would be highlighted. And if we would be working in vertical, then the top and bottom edges of the box would be highlighted.

Correspondingly in the example image, the left and right (approximately) edges of the bodies are highlighted. This produces light stripes that are going in bottom-top direction.

(iv) Referring to the magnitude of the frequency response, describe the effect of the filter in terms of what happens to different frequencies in the image.

When the frequency is 0 in the plot, the frequency response has the magnitude 0. This is expected since when the frequency is 0, there is no change and thus the filter will give 0. As the frequency increases in the plot, we see that the magnitude of the frequency response also increases. This corresponds to fact that higher frequency means higher (faster) change and this filter finds the change.

Code for Part 1)

```
A=imread('couple.bmp');
J=mat2gray(A, [0 255]);
imshow(J);

m = 11;
mOver2 = (m - 1)/2;

c = 0.2;
noise = rand(512,512);
noise = noise - 0.5;
noise = noise * c;
J = J + noise;

J = [zeros(512, mOver2) , J, zeros(512, mOver2)];
res = zeros(512, 512);
for i = 1:512
    for j = 1:512
        res(i,j) = 0;
        for k = 0:(m - 1)
            res(i,j) = res(i,j) + J(i, j + k);
        end
    end
end
res = res / m;

omegaHat = linspace(-pi, pi, 1000);
freqResponse = zeros(1000);

for a = 1:1000
    freqResponse(a) = 0;
    for b = -mOver2:mOver2
        freqResponse(a) = freqResponse(a) + exp(-1.0i *
omegaHat(a) * b);
    end
    freqResponse(a) = freqResponse(a) / m;
end

freqResponseMagnitude = abs(freqResponse);
plot(omegaHat, freqResponseMagnitude);
title('Plot of Frequency Response Magnitudes Between -\pi and
\pi');
xlabel('-\pi < frequency < \pi');
ylabel('Magnitude of the Frequency Response');
xlim([-pi pi]);
set(gca,'XTick',[-pi -3*pi/4 -pi/2 -pi/4 0 pi/4 pi/2 3*pi/4
pi]);
```

```
set(gca,'XTickLabel',{'-\pi', '-3\pi/4', '-\pi/2', '-\pi/4',
'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'});


input('ENTER for the resulting image.');

imshow(res);
imwrite(res, 'sonuc.bmp');
```

Code for Part 2)

```
A=imread('couple.bmp');
J=mat2gray(A, [0 255]);
imshow(J);

J = [zeros(512, 1) , J];
res = J(:,2:513) - J(:,1:512);

omegaHat = linspace(-pi, pi, 1000);
freqResponse = zeros(1000);

for a = 1:1000
    freqResponse(a) = 1 - exp(-1.0i * omegaHat(a) * 1);
end

freqResponseMagnitude = abs(freqResponse);
plot(omegaHat, freqResponseMagnitude);
title('Plot of Frequency Response Magnitudes Between -\pi and
\pi');
xlabel('-\pi < frequency < \pi');
ylabel('Magnitude of the Frequency Response');
xlim([-pi pi]);
set(gca,'XTick',[-pi -3*pi/4 -pi/2 -pi/4 0 pi/4 pi/2 3*pi/4
pi]);
set(gca,'XTickLabel',{'-\pi', '-3\pi/4', '-\pi/2', '-\pi/4',
'0', '\pi/4', '\pi/2', '3\pi/4', '\pi'});

input('ENTER for the resulting image.');

imshow(res);
imwrite(res, 'sonuc.bmp');
```