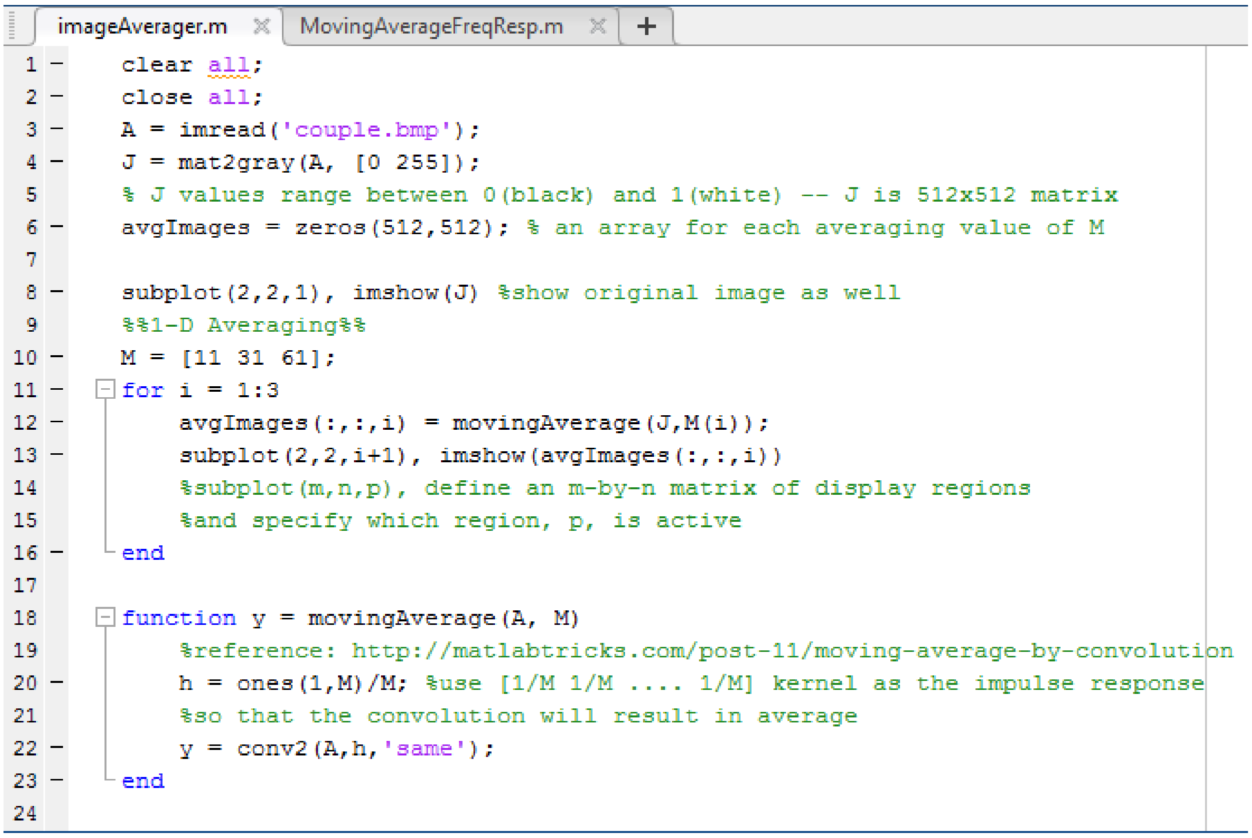
EEE391 – Computer Assignment 3
Basics of Signals and Systems
Spring 2016-2017

Ecem İlgün – 21502157

1)



```
clear all;
close all;
A = imread('couple.bmp');
J = mat2gray(A, [0 255]);
% J values range between 0(black) and 1(white) -- J is 512x512 matrix
avgImages = zeros(512,512); % an array for each averaging value of M

subplot(2,2,1), imshow(J) %show original image as well
%%1-D Averaging%%
M = [11 31 61];
for i = 1:3
    avgImages(:,:,i) = movingAverage(J,M(i));
    subplot(2,2,i+1), imshow(avgImages(:,:,i))
    %subplot(m,n,p), define an m-by-n matrix of display regions
    %and specify which region, p, is active
end

function y = movingAverage(A, M)
    %reference: http://matlabtricks.com/post-11/moving-average-by-convolution
    h = ones(1,M)/M; %use [1/M 1/M .... 1/M] kernel as the impulse response
    %so that the convolution will result in average
    y = conv2(A,h,'same');
end

```

Figure 1: Code Script of 1 Dimensional M point Averager Filter

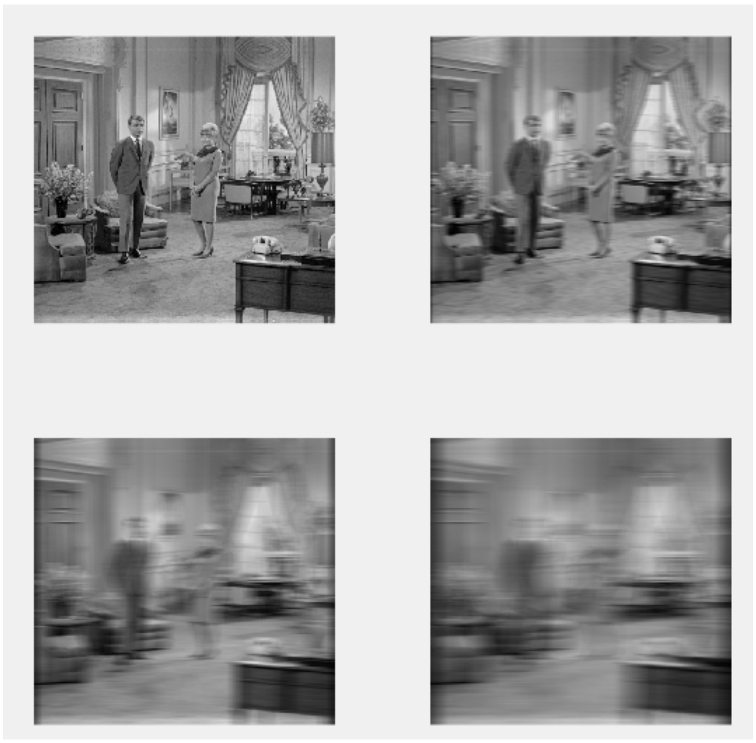
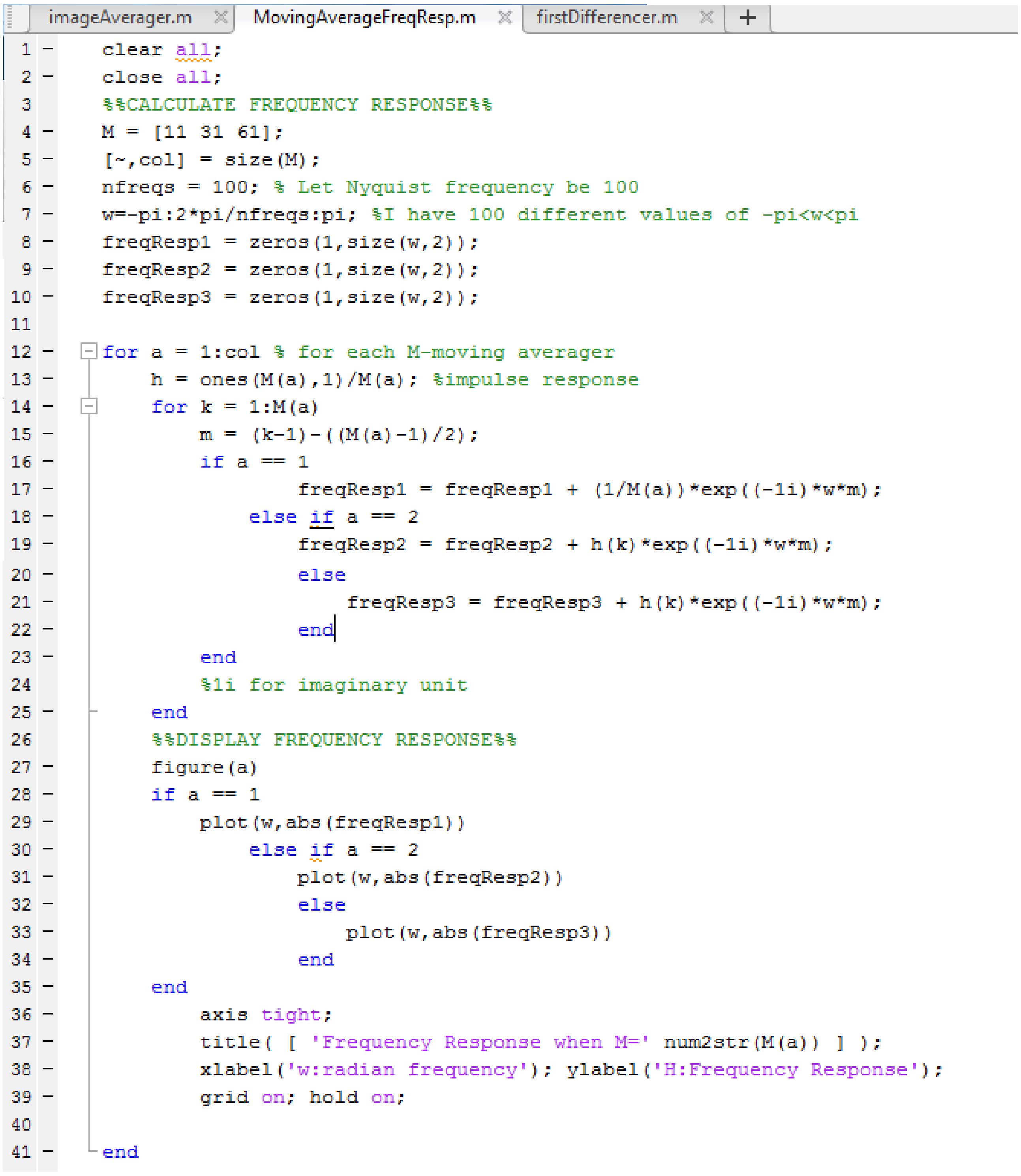


Figure 2: The original image(top-left), Averaged Image for M=11(top-right), M=31 (bottom-left), M=61 (bottom right)



```
clear all;
close all;
%%CALCULATE FREQUENCY RESPONSE%%
M = [11 31 61];
[~,col] = size(M);
nfreqs = 100; % Let Nyquist frequency be 100
w=-pi:2*pi/nfreqs:pi; %I have 100 different values of -pi<w<pi
freqResp1 = zeros(1,size(w,2));
freqResp2 = zeros(1,size(w,2));
freqResp3 = zeros(1,size(w,2));

for a = 1:col % for each M-moving averager
    h = ones(M(a),1)/M(a); %impulse response
    for k = 1:M(a)
        m = (k-1)-((M(a)-1)/2);
        if a == 1
                freqResp1 = freqResp1 + (1/M(a))*exp((-1i)*w*m);
            else if a == 2
                freqResp2 = freqResp2 + h(k)*exp((-1i)*w*m);
                else
                    freqResp3 = freqResp3 + h(k)*exp((-1i)*w*m);
                end
        end
        %1i for imaginary unit
    end
    %%DISPLAY FREQUENCY RESPONSE%%
    figure(a)
    if a == 1
        plot(w,abs(freqResp1))
            else if a == 2
                plot(w,abs(freqResp2))
                else
                    plot(w,abs(freqResp3))
                end
    end
        axis tight;
        title( [ 'Frequency Response when M=' num2str(M(a)) ] );
        xlabel('w:radian frequency'); ylabel('H:Frequency Response');
        grid on; hold on;

end
```

Figure 3: Code Script for Displaying Frequency Response

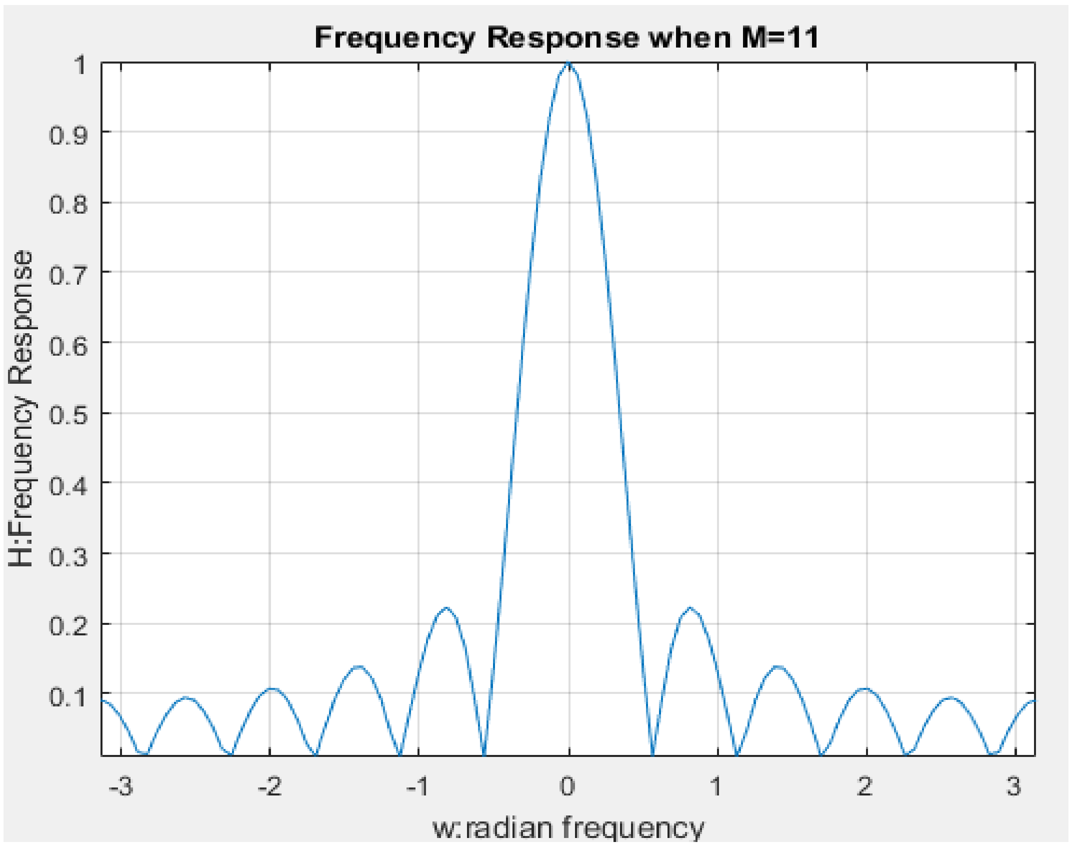


Figure 4: Plot of Frequency Response Function when M=11

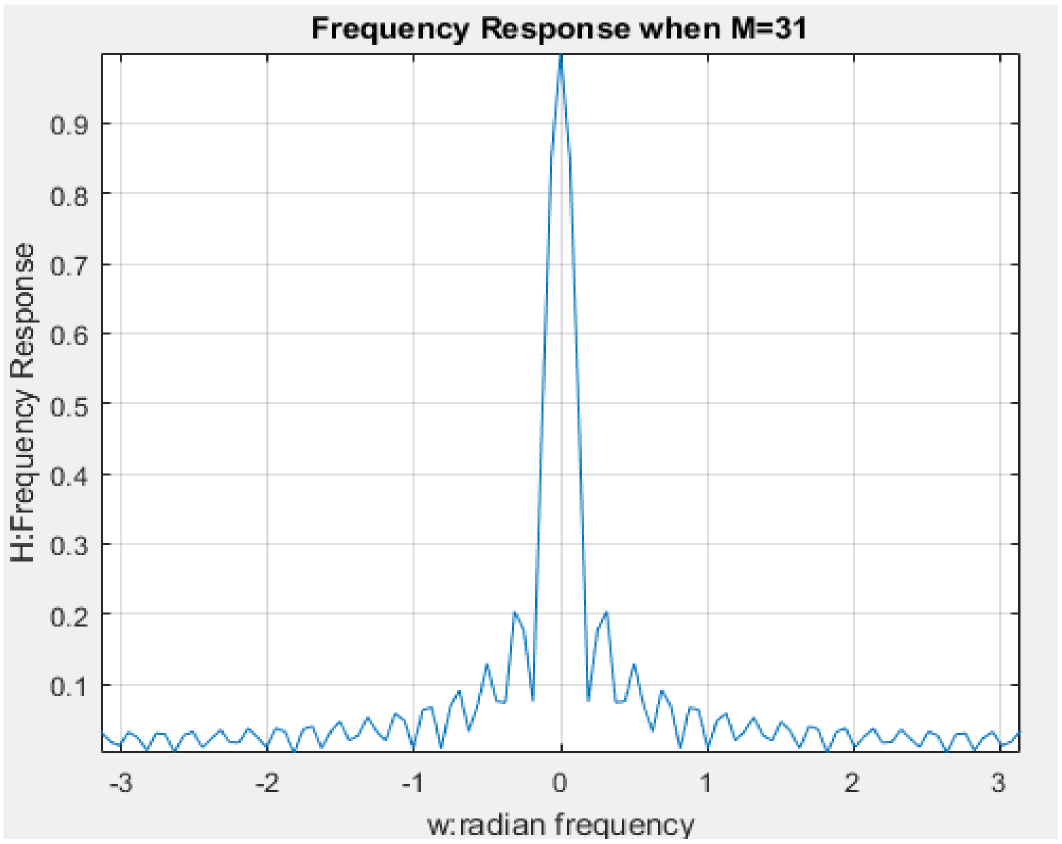


Figure 5: Plot of Frequency Response Function when M=31

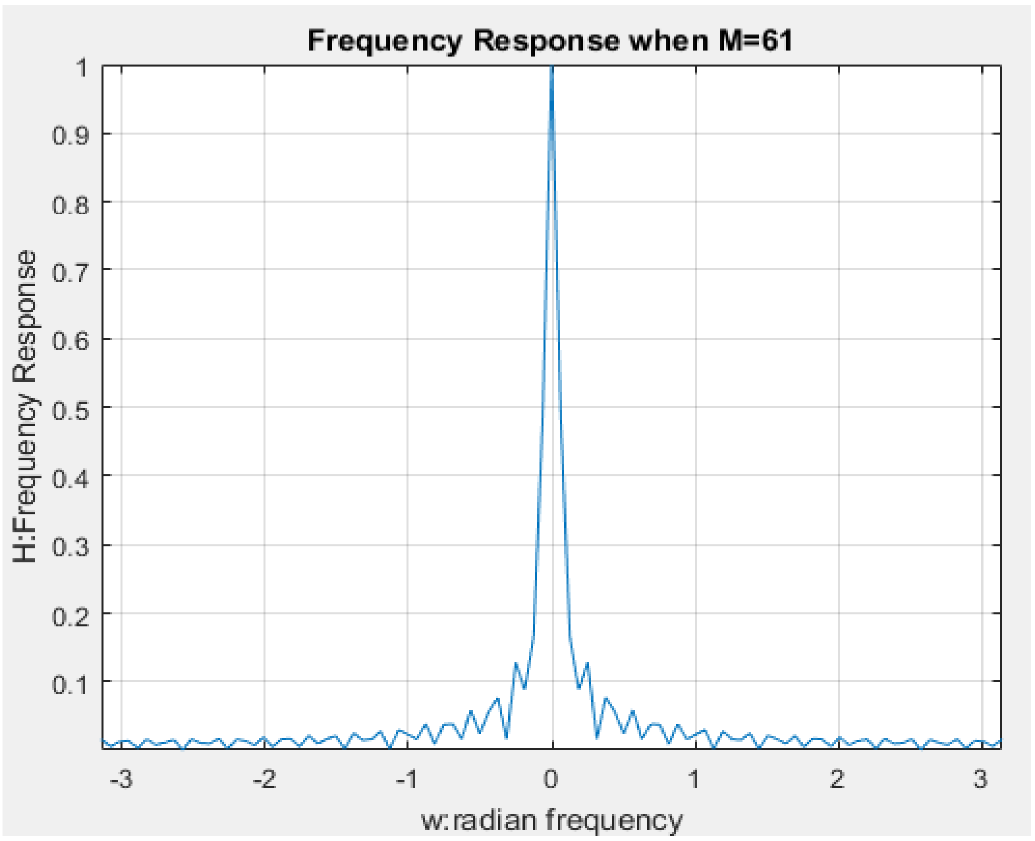


Figure 6: Plot of Frequency Response Function when M=61

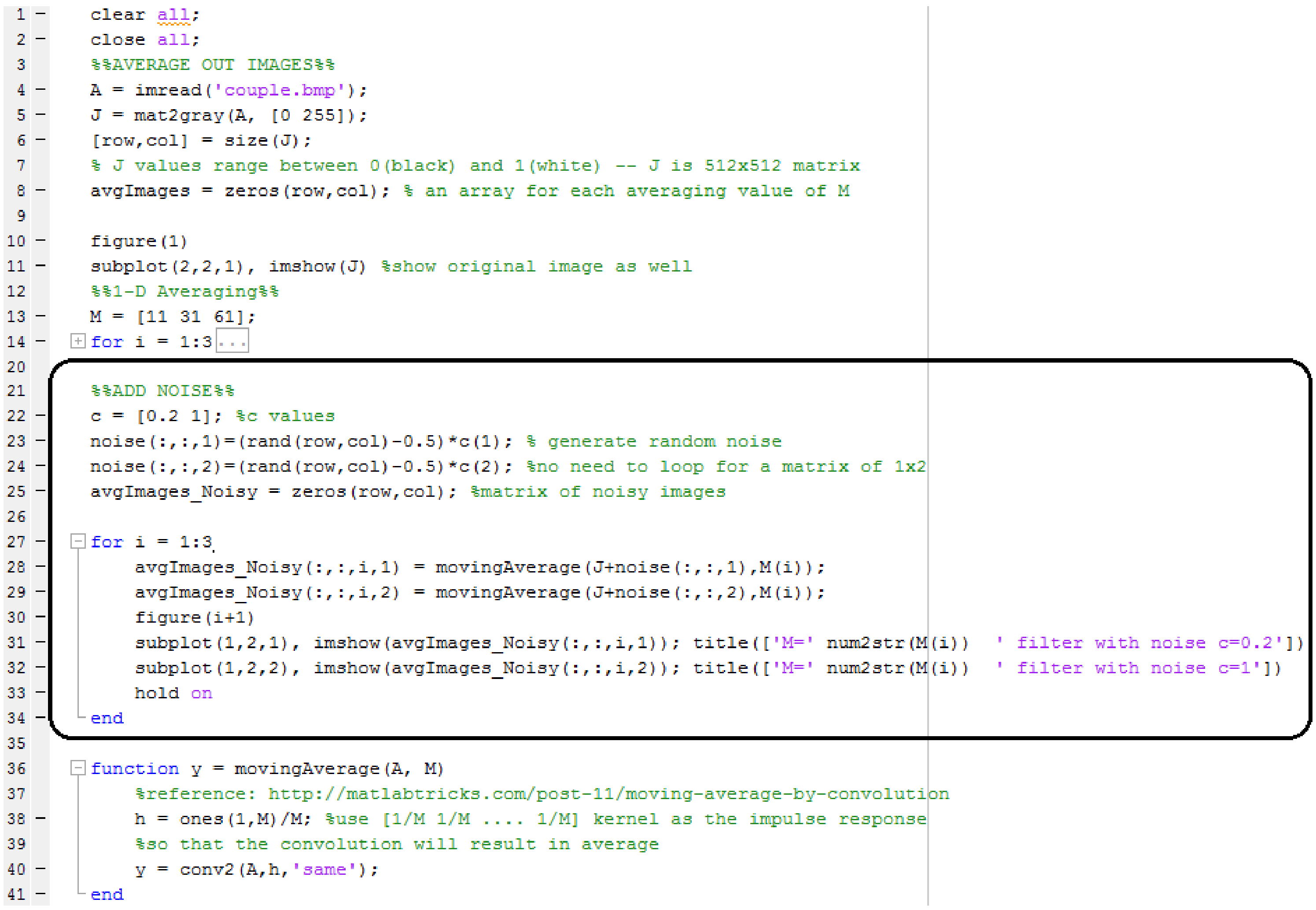
(i) The visual effect of averaging filter looks like somewhat strecthing the image. It makes the image blurry in horizontal dimension by averaging each point with its neighbors, which causes the strecthing effect. The strecthing effect I mentioned is also called smoothing.

(ii) The details of the image decrease as M increases, since averaging causes blurriness as mentioned in question i.

(iii) The averager is one dimensional therefore it makes no change to dimension n(vertical), however it averages, smooths out the horizontal points.

(iv) Low frequencies go through little attenuation when they are passed through the filter, wherease high frequencies are greatly attenuated. Therefore the filter is *low-pass* filter.

(v) The edges start to look darker. That is because the edges receive less input than the middle points. The larger M gets, the bigger the denominator of averaging gets. Since edges receive less input and M keeps increasing, the data point representing an edge point gets closer to 0.



```
clear all;
close all;
%%AVERAGE OUT IMAGES%%
A = imread('couple.bmp');
J = mat2gray(A, [0 255]);
[row,col] = size(J);
% J values range between 0(black) and 1(white) -- J is 512x512 matrix
avgImages = zeros(row,col); % an array for each averaging value of M

figure(1)
subplot(2,2,1), imshow(J) %show original image as well
%%1-D Averaging%%
M = [11 31 61];
for i = 1:3...

%%ADD NOISE%%
c = [0.2 1]; %c values
noise(:,:,1)=(rand(row,col)-0.5)*c(1); % generate random noise
noise(:,:,2)=(rand(row,col)-0.5)*c(2); %no need to loop for a matrix of 1x2
avgImages_Noisy = zeros(row,col); %matrix of noisy images

for i = 1:3
    avgImages_Noisy(:,:,i,1) = movingAverage(J+noise(:,:,1),M(i));
    avgImages_Noisy(:,:,i,2) = movingAverage(J+noise(:,:,2),M(i));
    figure(i+1)
    subplot(1,2,1), imshow(avgImages_Noisy(:,:,i,1)); title(['M=' num2str(M(i))  ' filter with noise c=0.2'])
    subplot(1,2,2), imshow(avgImages_Noisy(:,:,i,2)); title(['M=' num2str(M(i))  ' filter with noise c=1'])
    hold on
end

function y = movingAverage(A, M)
    %reference: http://matlabtricks.com/post-11/moving-average-by-convolution
    h = ones(1,M)/M; %use [1/M 1/M .... 1/M] kernel as the impulse response
    %so that the convolution will result in average
    y = conv2(A,h,'same');
end
```

Figure 7 : Code Script of 1 Dimensional M point Averager Filter, with the parts added for noise shown in black rectangular shape

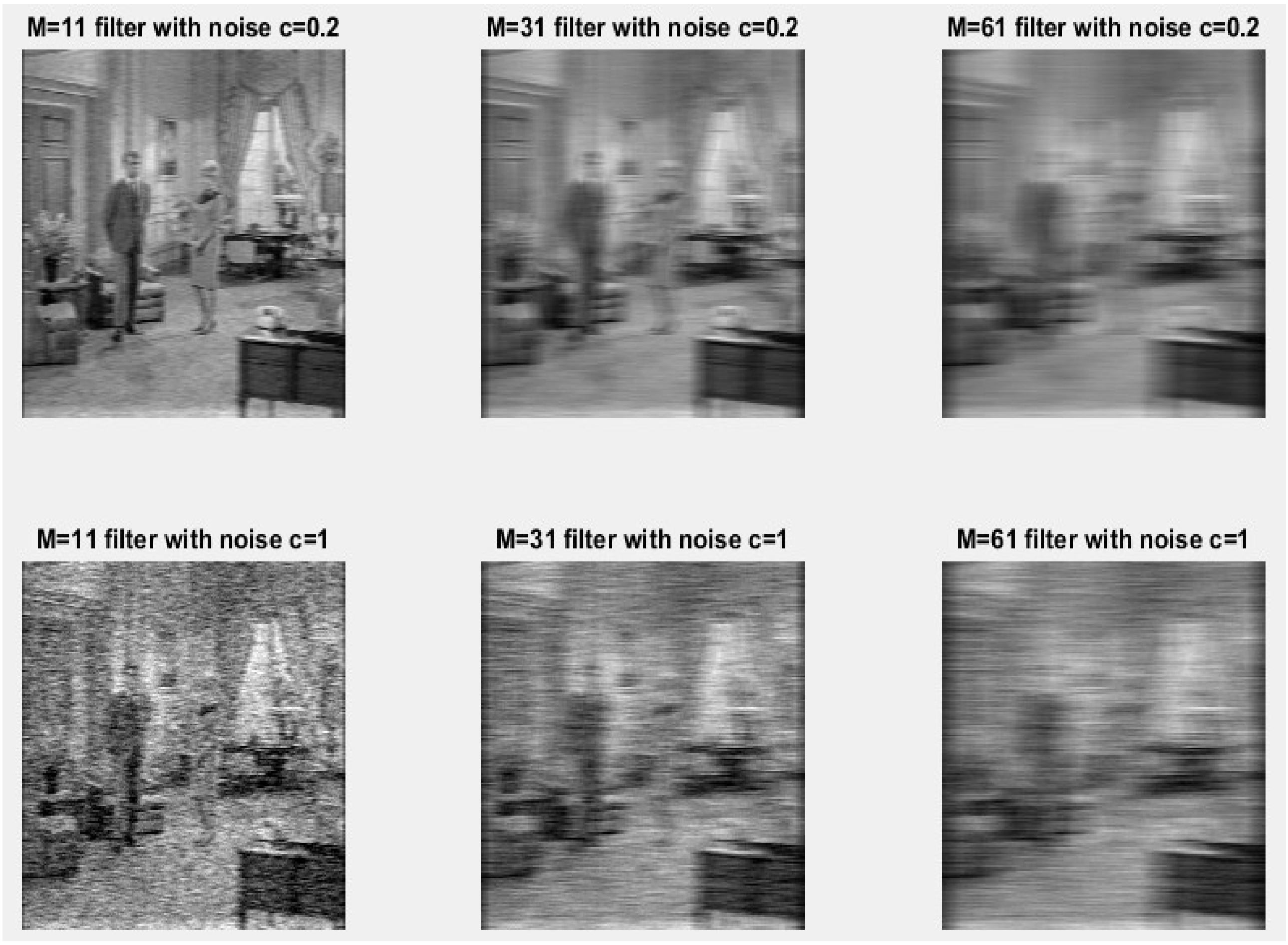

M=11 filter with noise c=0.2
M=31 filter with noise c=0.2
M=61 filter with noise c=0.2
M=11 filter with noise c=1
M=31 filter with noise c=1
M=61 filter with noise c=1

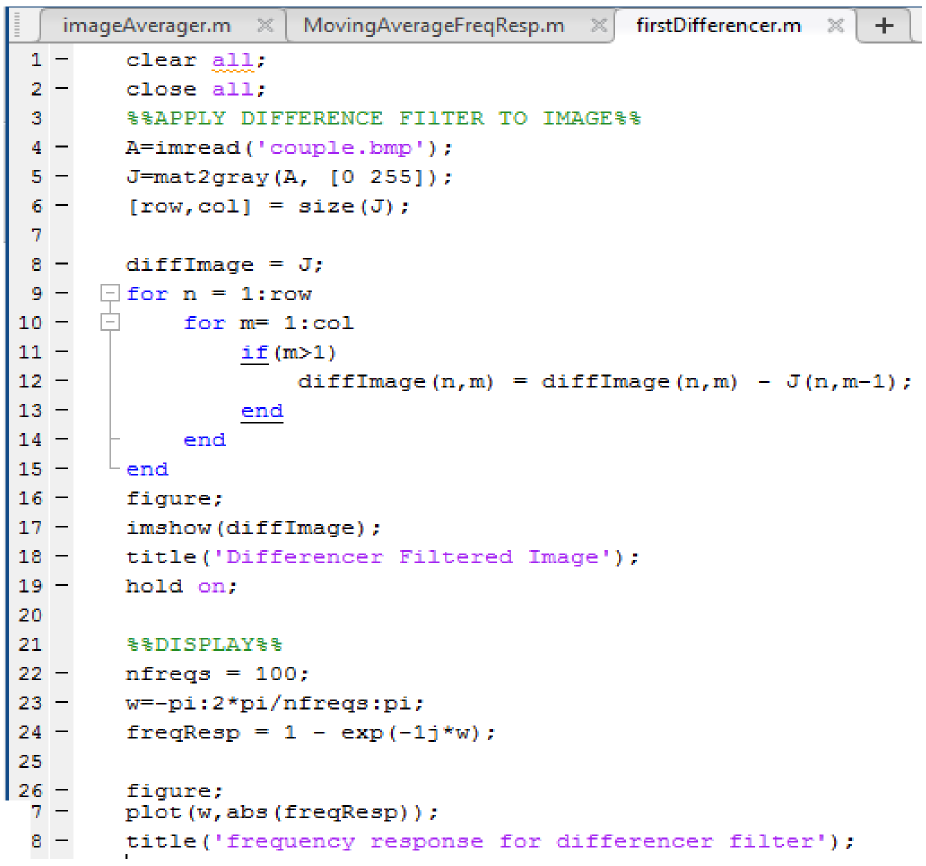
Figure 8 : Plots of each averaging filter with the amount of noise added written in title on top of each image

(vi) Yes, it helps to reduce the noise. This happens because by averaging out the filter makes the amplitude of noise smaller. As M gets larger, the amplitude of the noise added gets smaller and smaller, to the point of images with or without the noise looking almost exact when M=61. Therefore, the smoothing effect eliminates the noise for large M values.

(vii) The undesirable effect of smoothing is that not only the noise but also the signal itself gets smoothed, making the image blurry. In this way, we can obtain less information from the actual signal if we choose to smooth out with large values of M.

(viii) I believe when M is 31 and 61 the image gets too blurry to grasp the couple, therefore, choosing M=11 would be the best case for this particular image.

2)



```
clear all;
close all;
%%APPLY DIFFERENCE FILTER TO IMAGE%%
A=imread('couple.bmp');
J=mat2gray(A, [0 255]);
[row,col] = size(J);

diffImage = J;
for n = 1:row
    for m= 1:col
        if(m>1)
            diffImage(n,m) = diffImage(n,m) - J(n,m-1);
        end
    end
end
figure;
imshow(diffImage);
title('Differencer Filtered Image');
hold on;

%%DISPLAY%%
nfreqs = 100;
w=-pi:2*pi/nfreqs:pi;
freqResp = 1 - exp(-1j*w);

figure;
plot(w,abs(freqResp));
title('frequency response for differencer filter');
```

Figure 9: Code Script of First Differencer Filter and its Freqeuncy Response Question

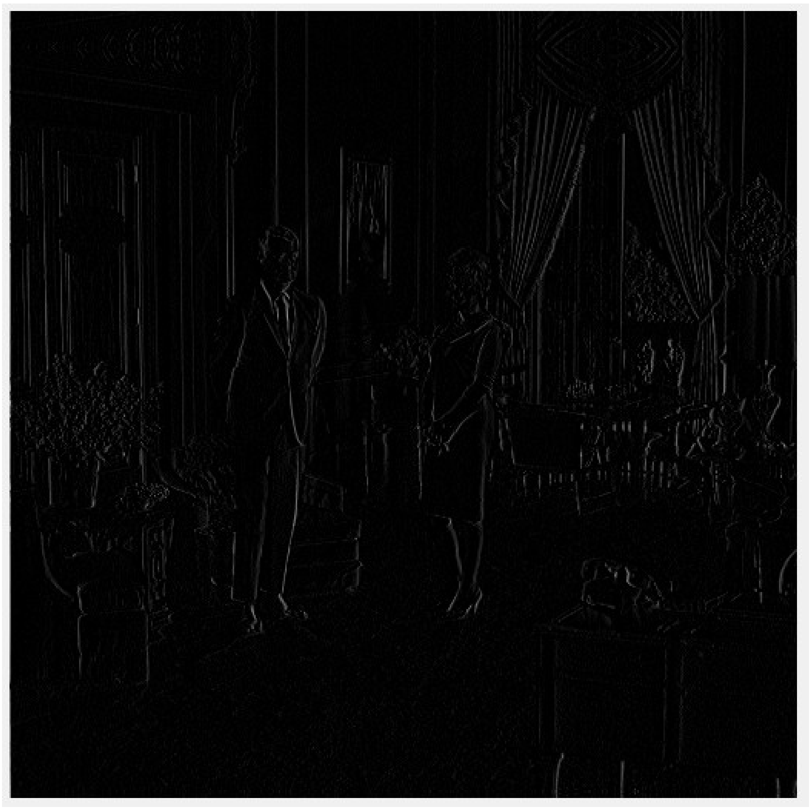


Figure 10 : Difference Filtered Image

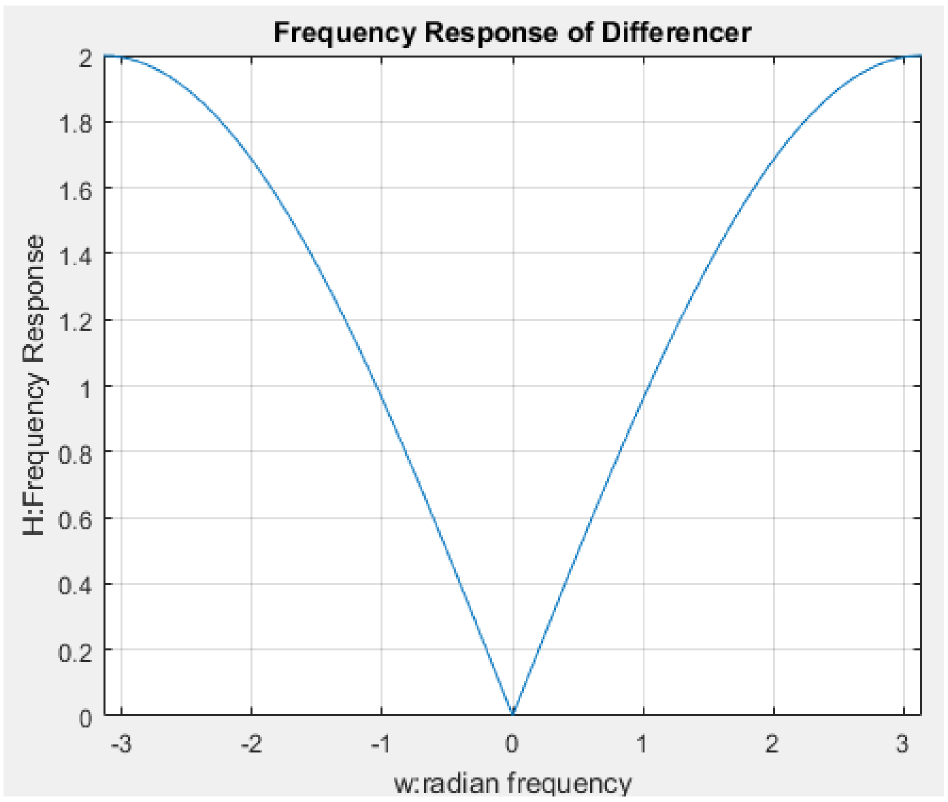


Figure 11 : Plot of Differencer Filter's Frequency Response

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

(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.

(i) The visual effect of differencing filter looks like its emphasizing the edges of the photo. It seems to be quite the opposite of averaging filter, since averaging filter smoothed and differencing filter made edgy.

(ii) The edgying effect can be explained by the theoretical aspect of differencing filter. In differencing filter, the overall trend of data (be black or white or gray, any shade) is gone, only local irregularities remain. These irregularities happen to be the edges of couple or other objects.

(iii) The differencer is again one dimensional therefore it makes no change to dimension n(vertical), however it displays the irregularities in horizontal axis.

Below are horizontal and vertical difference filtered images, from the difference between images it can be seen that the horizontal differencer filter emphasized horizontal edges, whereas vertical differencer filter did the opposite.

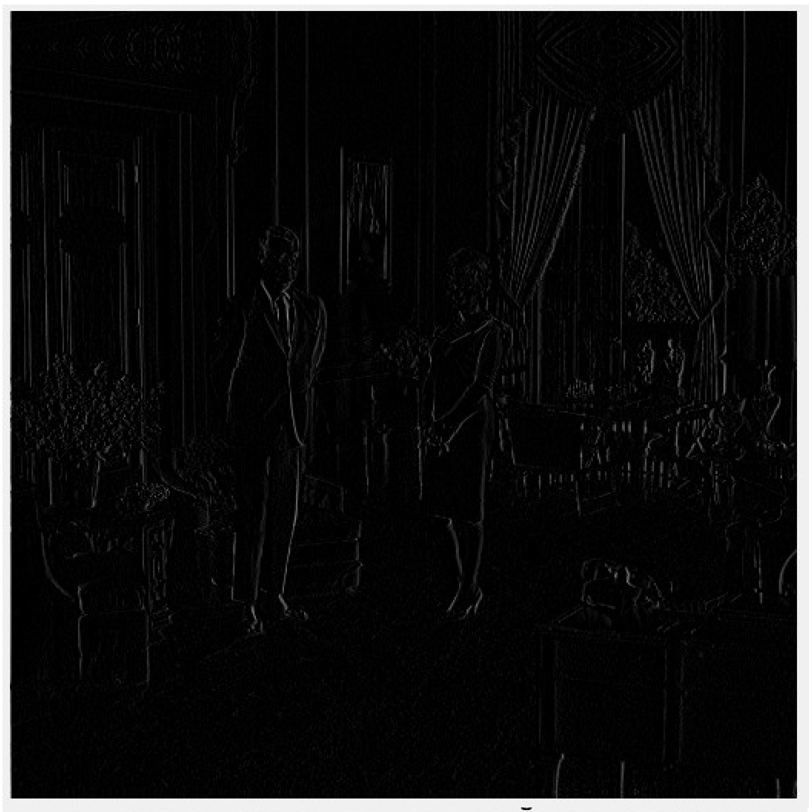
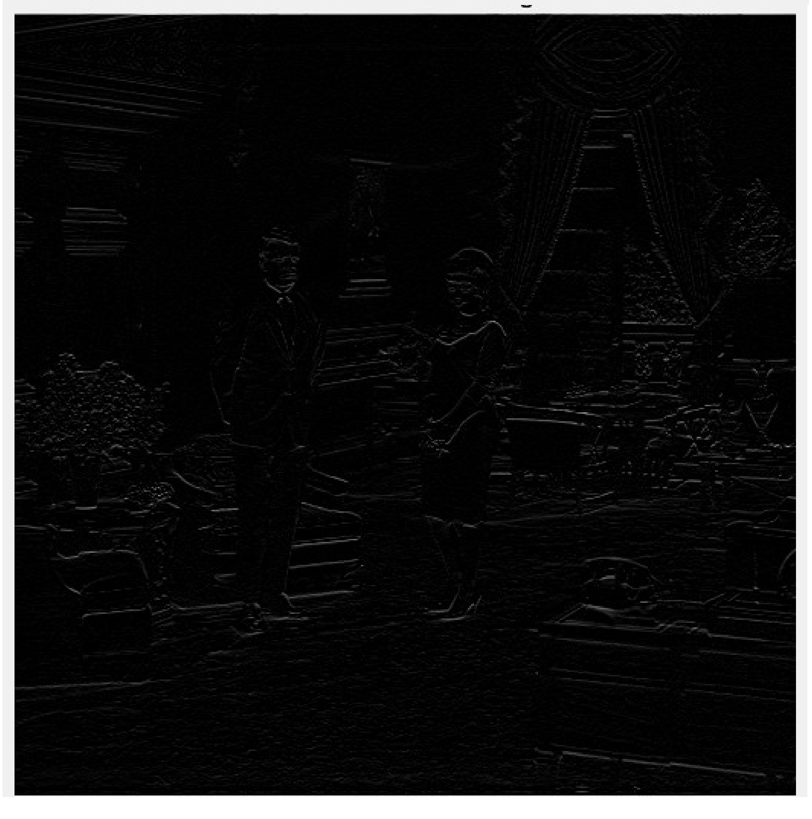



Figure 11: Difference Filtered Image by horizontal axis(up) and by vertical axis(down)

(iv) It is seen from frequency response that the system emphasizes higher frequencies, therefore the filter is *high-pass* filter. It is opposite of averaging filter, low frequencies are highly attenuatied when they are passed through the filter, whereas high frequencies are not much attenuated.

Appendix of Codes:

```
clear all;
close all;
%%AVERAGE OUT IMAGES%%
A = imread('couple.bmp');
J = mat2gray(A, [0 255]);
[row,col] = size(J);
% J values range between 0(black) and 1(white) -- J is 512x512 matrix
avgImages = zeros(row,col); % an array for each averaging value of M

figure(1)
subplot(2,2,1), imshow(J) %show original image as well
%%1-D Averaging%%
M = [11 31 61];
for i = 1:3
   avgImages(:,:,i) = movingAverage(J,M(i));
   subplot(2,2,i+1), imshow(avgImages(:,:,i))
   %subplot(m,n,p), define an m-by-n matrix of display regions
   %and specify which region, p, is active
end

%%ADD NOISE%%
c = [0.2 1]; %c values
noise(:,:,1)=(rand(row,col)-0.5)*c(1); % generate random noise
noise(:,:,2)=(rand(row,col)-0.5)*c(2); %no need to loop for a matrix of 1x2
avgImages_Noisy = zeros(row,col); %matrix of noisy images

for i = 1:3
   avgImages_Noisy(:,:,i,1) = movingAverage(J+noise(:,:,1),M(i));
   avgImages_Noisy(:,:,i,2) = movingAverage(J+noise(:,:,2),M(i));
   figure(2)
   subplot(2,3,i), imshow(avgImages_Noisy(:,:,i,1)); title(['M=' num2str(M(i))  ' filter with noise
c=0.2'])
   subplot(2,3,i+3), imshow(avgImages_Noisy(:,:,i,2)); title(['M=' num2str(M(i))  ' filter with noise
c=1'])
   hold on
end

function y = movingAverage(A, M)
   %reference: http://matlabtricks.com/post-11/moving-average-by-convolution
   h = ones(1,M)/M; %use [1/M 1/M .... 1/M] kernel as the impulse response
   %so that the convolution will result in average
   y = conv2(A,h,'same');
end
```

```
clear all;
close all;
%%CALCULATE FREQUENCY RESPONSE%%
M = [11 31 61];
[~,col] = size(M);
nfreqs = 100; % Let Nyquist frequency be 100
w=-pi:2*pi/nfreqs:pi; %I have 100 different values of -pi<w<pi
freqResp1 = zeros(1,size(w,2));
freqResp2 = zeros(1,size(w,2));
freqResp3 = zeros(1,size(w,2));

for a = 1:col % for each M-moving averager
   h = ones(M(a),1)/M(a); %impulse response
   for k = 1:M(a)
      m = (k-1)-((M(a)-1)/2);
      if a == 1
            freqResp1 = freqResp1 + (1/M(a))*exp((-1i)*w*m);
         else if a == 2
            freqResp2 = freqResp2 + h(k)*exp((-1i)*w*m);
            else
               freqResp3 = freqResp3 + h(k)*exp((-1i)*w*m);
            end
      end
      %1i for imaginary unit
   end
   %%DISPLAY FREQUENCY RESPONSE%%
   figure(a)
   if a == 1
      plot(w,abs(freqResp1))
         else if a == 2
            plot(w,abs(freqResp2))
            else
               plot(w,abs(freqResp3))
            end
   end
      axis tight;
      title( [ 'Frequency Response when M=' num2str(M(a)) ] );
      xlabel('w:radian frequency'); ylabel('H:Frequency Response');
      grid on; hold on;

end
```

```
clear all;
close all;
%%APPLY DIFFERENCE FIlTER TO IMAGE%%
A=imread('couple.bmp');
J=mat2gray(A, [0 255]);
[row,col] = size(J);

diffImage = J;
for n = 1:row
   for m= 1:col
      if(n>1)
         diffImage(n,m) = diffImage(n,m) - J(n-1,m);
      end
   end
end
figure;
imshow(diffImage);
title('Differencer Filtered Image');
hold on;

%%DISPLAY FREQ RESPONSE%%
nfreqs = 100;
w=-pi:2*pi/nfreqs:pi;
freqResp = 1 - exp(-1j*w);

figure;
plot(w,abs(freqResp));
axis tight;
title( 'Frequency Response of Differencer' );
xlabel('w:radian frequency'); ylabel('H:Frequency Response');
grid on;
```