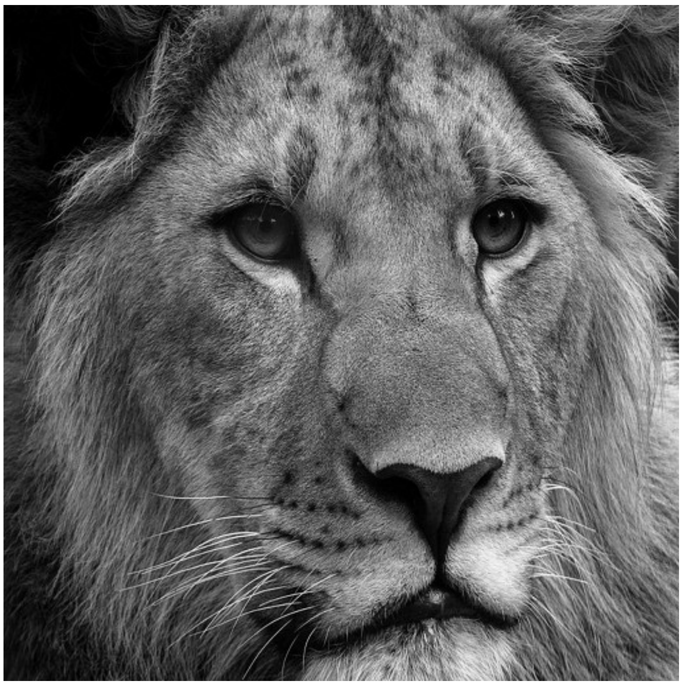
EEE 391 Basics of Signals and Systems

MATLAB Assignment 2

Barkın Saday
21902967
Section: 1

REPORT:

-I choose "the lion image" for this assignment. The original picture is provided below for better comparision.



1-D Filter:

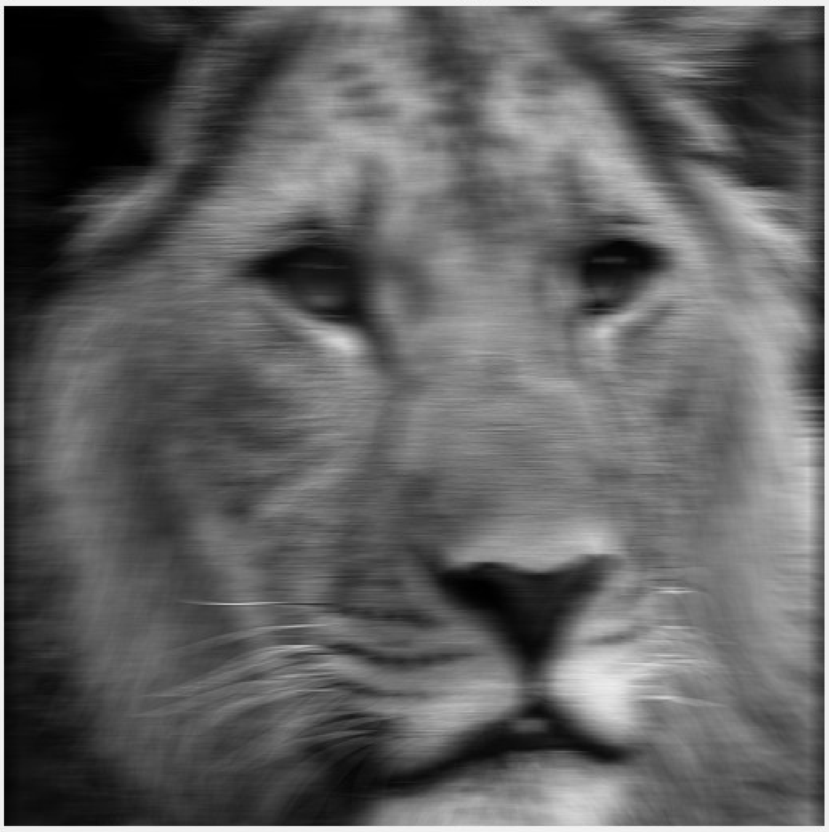
-The filter blurs the image in one dimension (hozirontally). As the value of M increases the smoothing is also increased.

-As for this part, since smoothing takes input for 1-D, there is no change in vertical dimension. At first this does not seem obvious when we look at the images however after "part 2" of the project it will be more obvious that for "part 1" only horizontal dimension is changed.

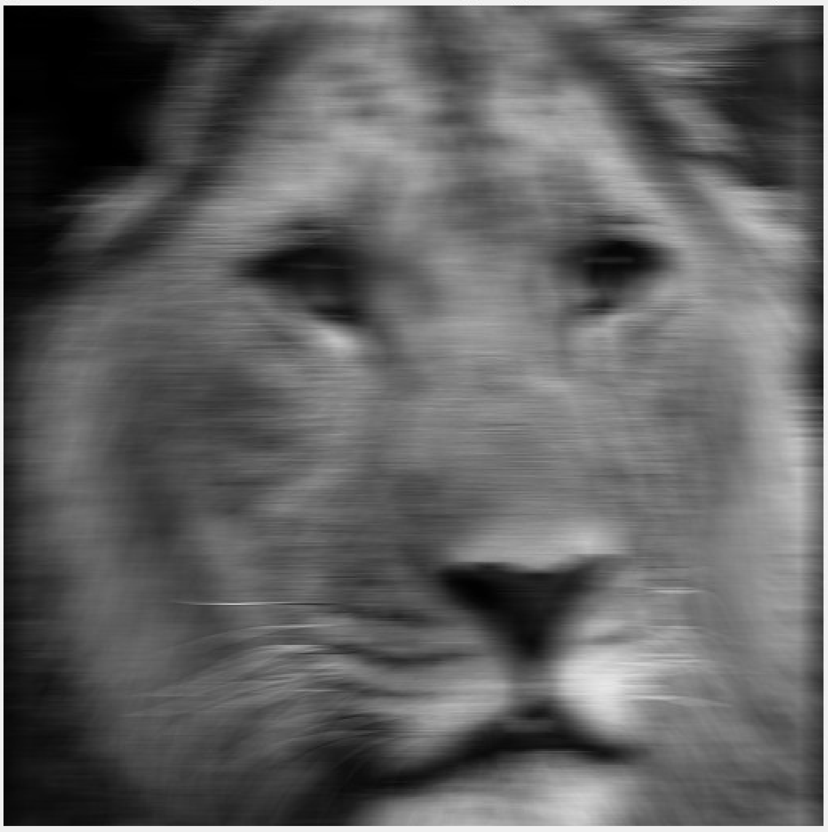
-With increase frequency magnitude is decreased. This is why the effect of filter decreases for larger M values.

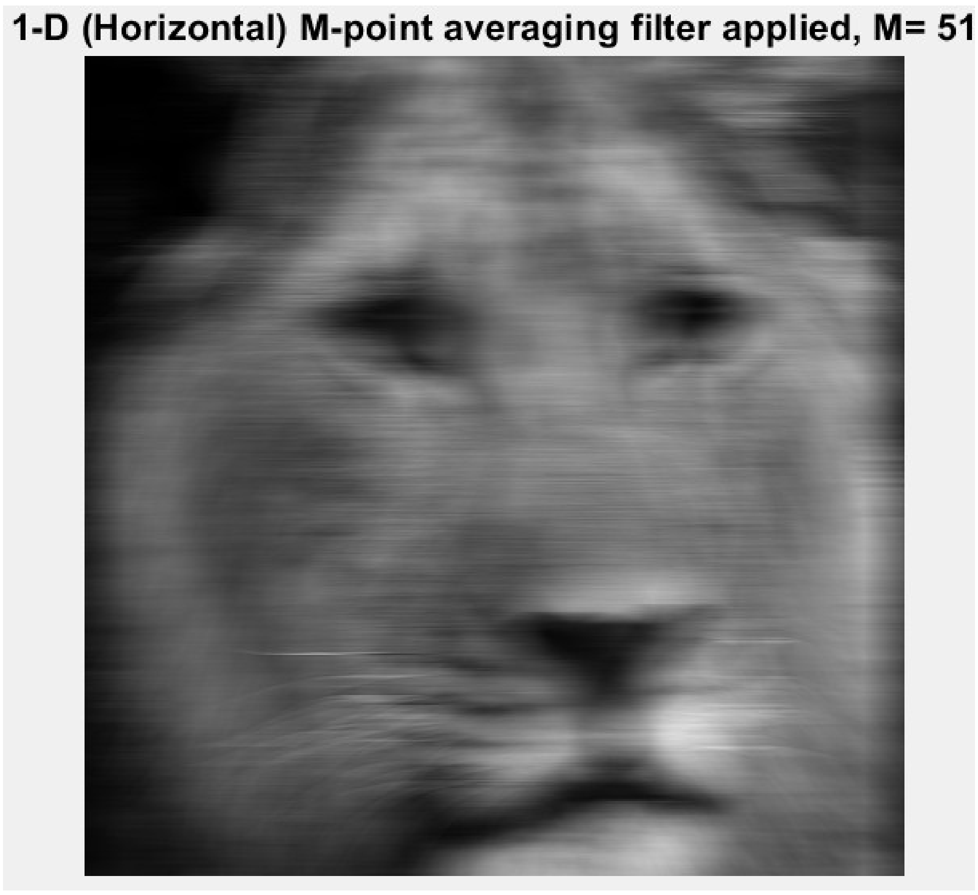
-The images with 1-D M-point averaging filter applied for the values M=21, M=31, M=51 can be found below. You can check the titles to match the values with the images.

1-D (Horizontal) M-point averaging filter applied, M= 21



1-D (Horizontal) M-point averaging filter applied, M= 31





Adding Random Variable “b”:

-Adding a random variable to each element of the the matrix is basically adding noise to our signal.

-By looking at the images after noise is added, we can make a visual interpretation as the following: Noise reduces the effect caused by the value of M. When noise is added the images does not get closer to the original image. However, The images look more alike to each other for the values M=21, M=31, M=51.

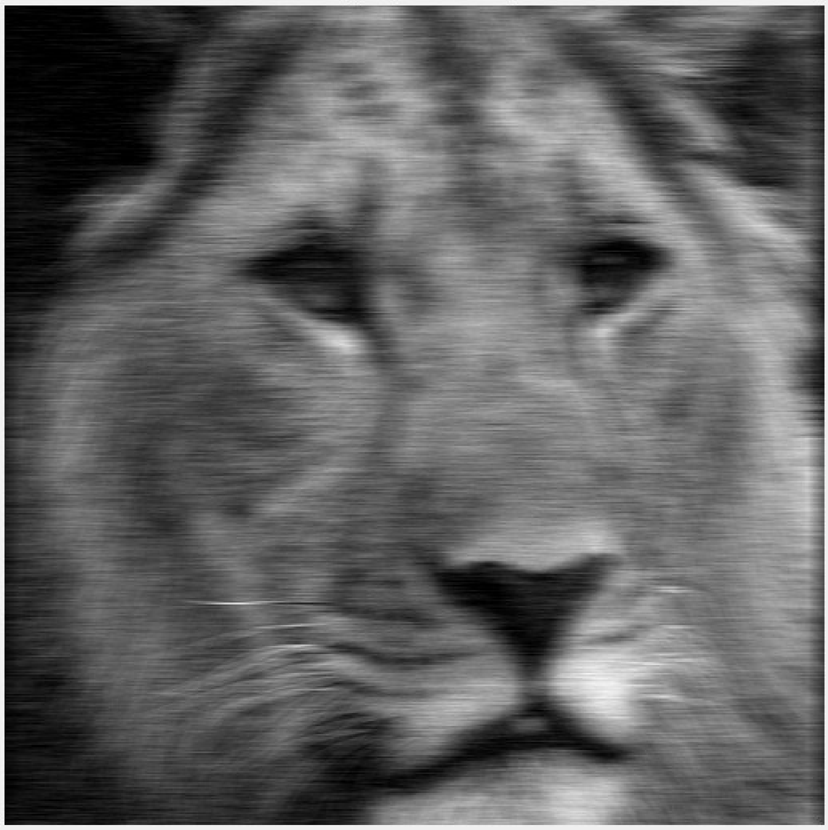
-We can also say that higher M reduces the effect of noise. But also, a high M value would mean that we are further bluring the image in the horizontal dimension.

-So, if we had a real life situation in which we have a noised version of the original image and we want to recover it, we would need to choose a fitting M value. If M is too low our image is highly affected by the noise. On the other hand, if the M value is too high, we can end up with an over-kill such that smoothing on horizontal dimension is way too much.

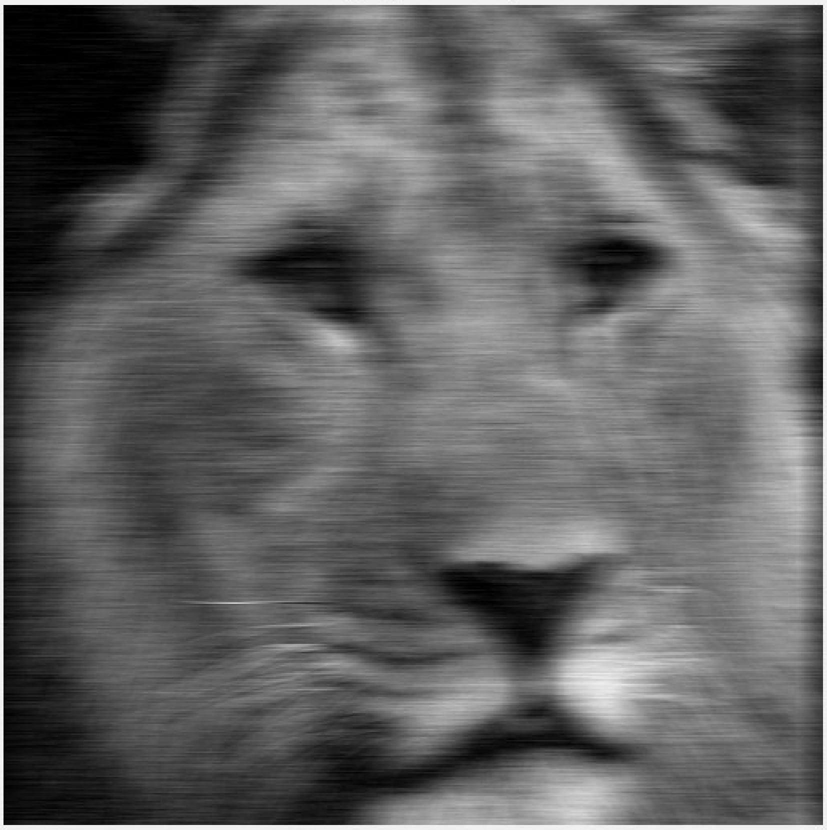
-Bellow figures for different M values (21, 31, 51) with noise can be seen. We have two different versions of the noise such that b=0.4 or b=0.9 (6 images in total).

-Note: b=4.000000e-1 and b=9.000000e-1 indicates b=0.4 and b=0.9 respectively.

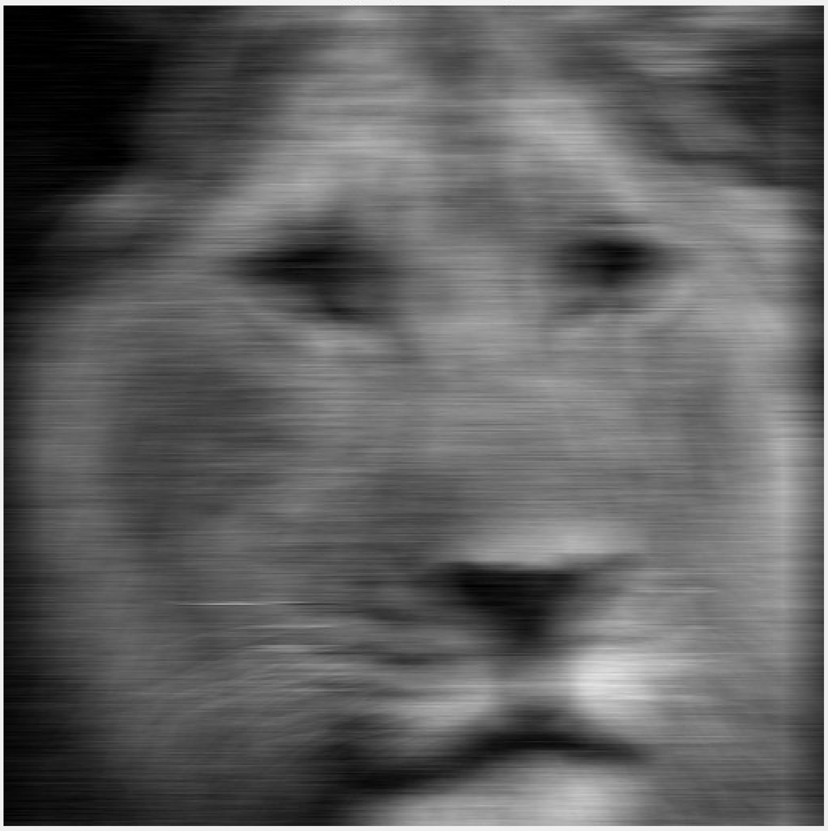
Added noise to image, M= 21, b= 4.000000e-01



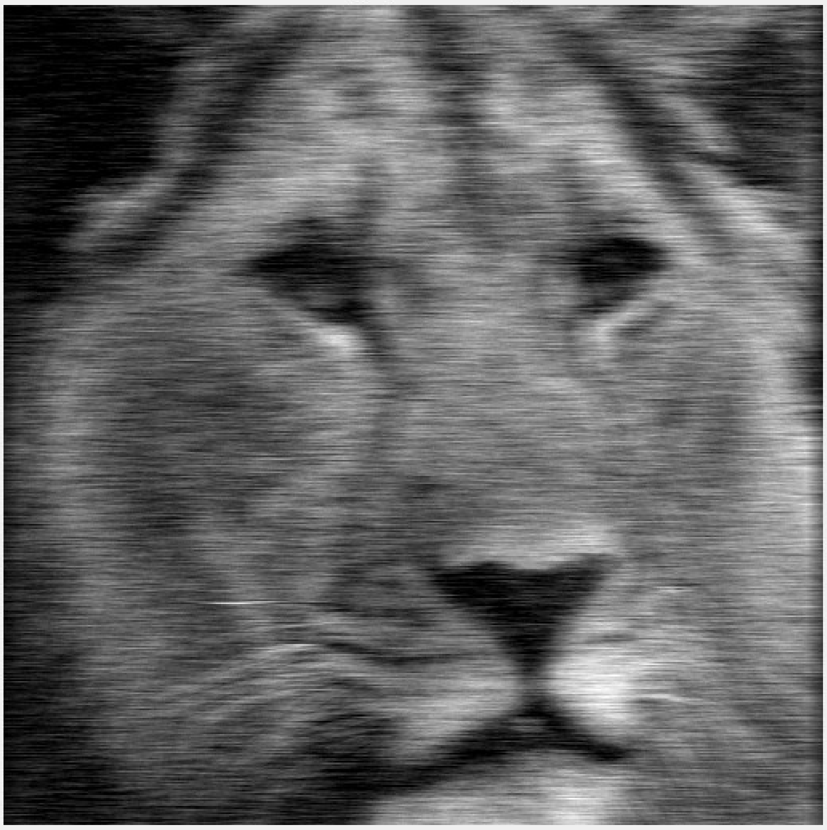
Added noise to image, M= 31, b= 4.000000e-01



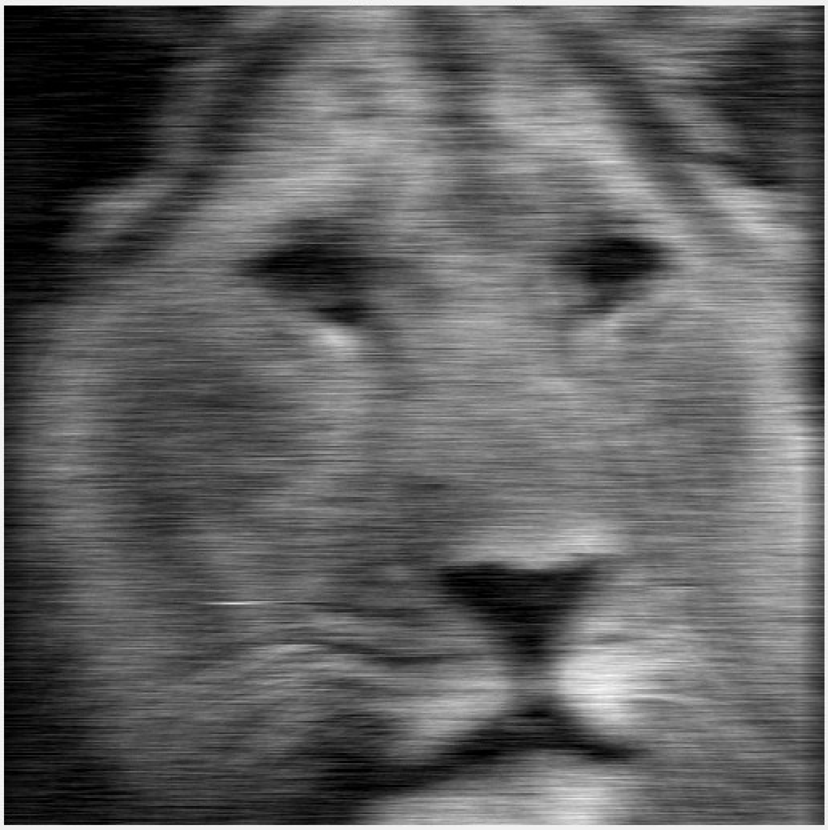
Added noise to image, M= 51, b= 4.000000e-01



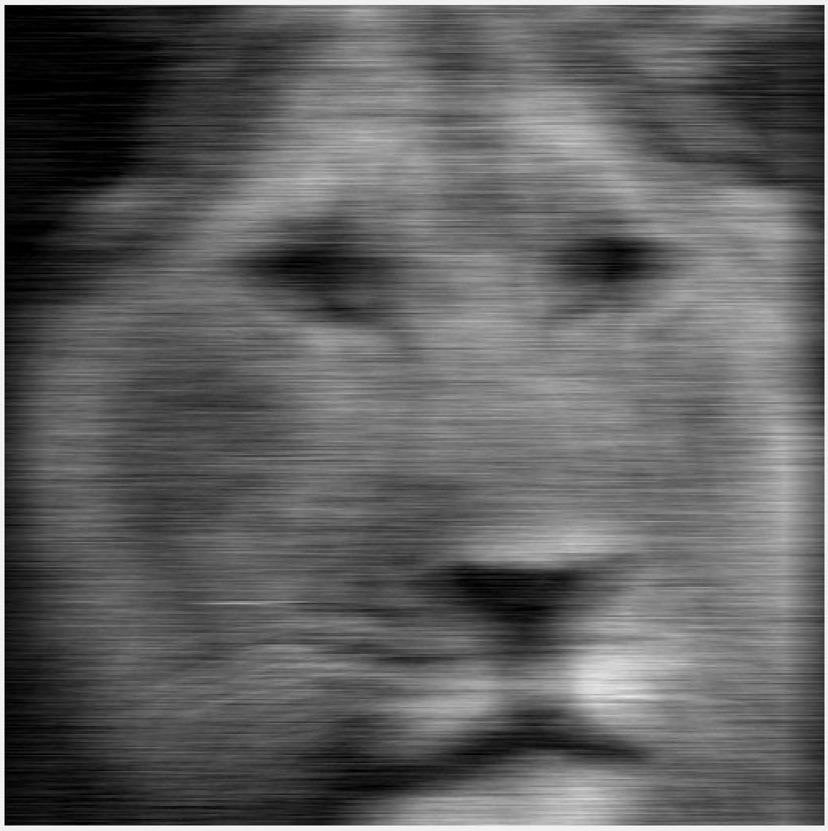
Added noise to image, M= 21, b= 9.000000e-01



Added noise to image, M= 31, b= 9.000000e-01



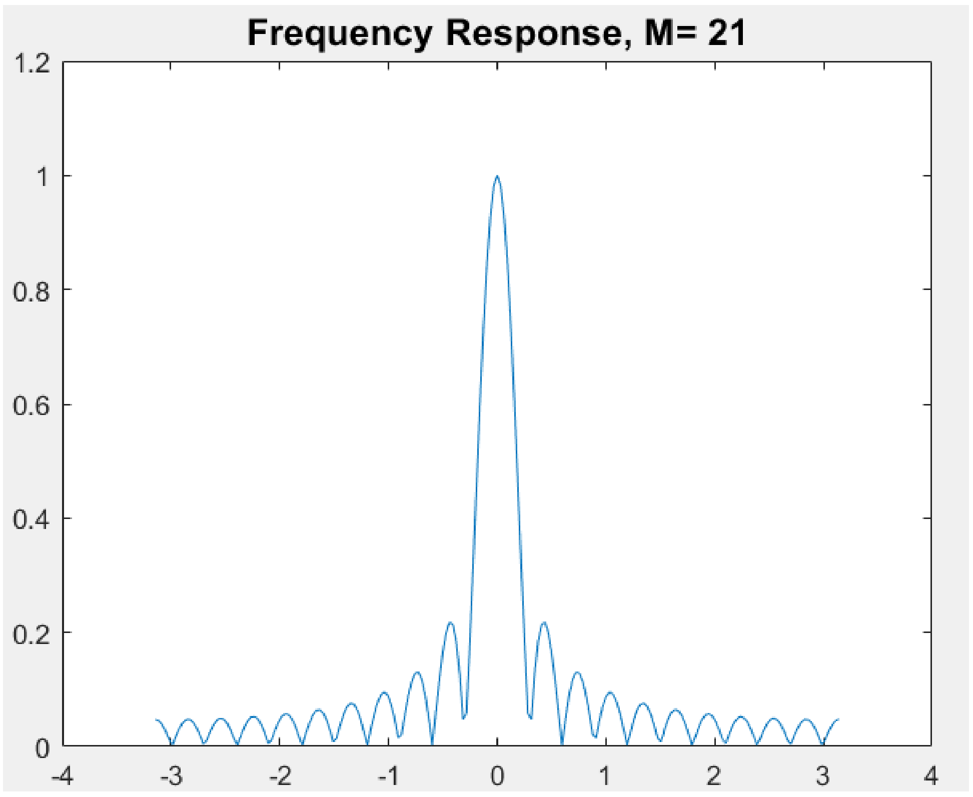
Added noise to image, M= 51, b= 9.000000e-01

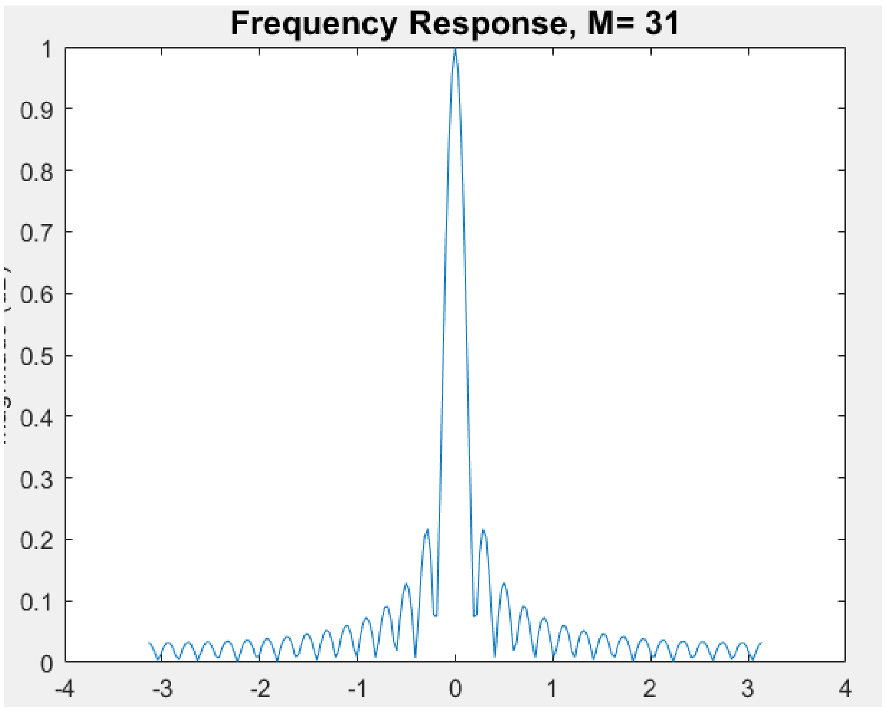


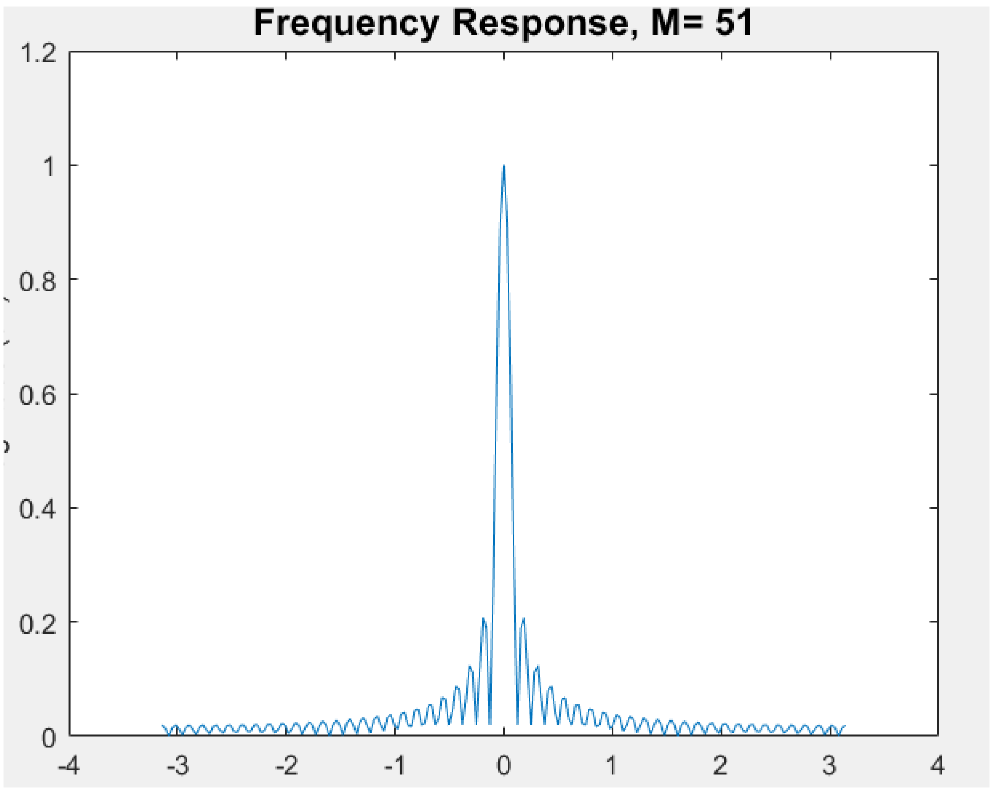
Frequency Responses and First Differencer:

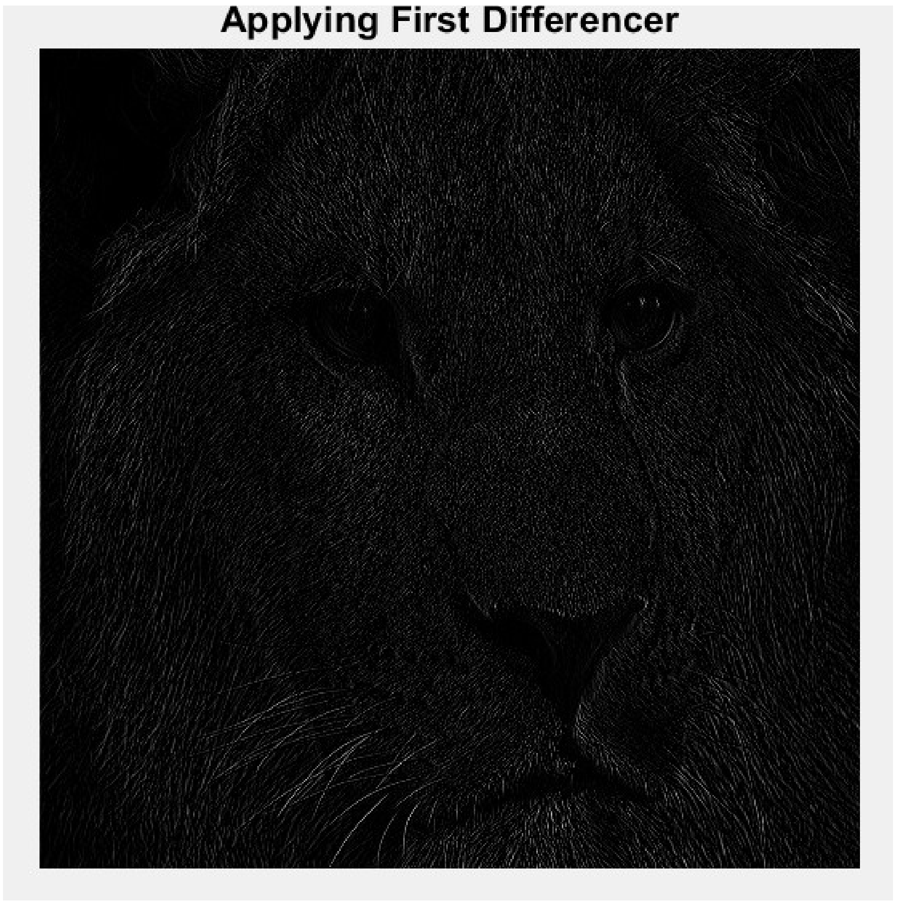
- Plots for frequency responses and first differencer can be found below.

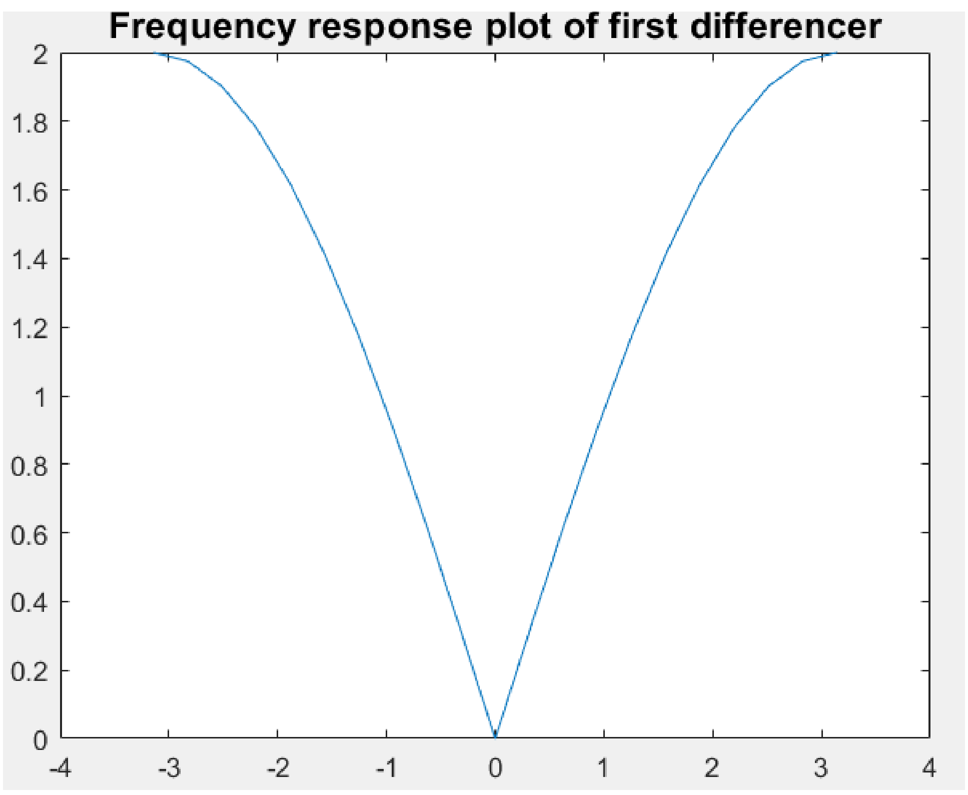
-Image after applying first differencer can be found below.






Frequency Response, M= 51
1.2
1
0.8
0.6
0.4
0.2
0
-4
-3
-2
-1
0
1
2
3
4


Applying First Differencer



MATLAB Code:

```
%Barkın Saday, EEE391-Project 2
%Part 1: Creating 1D M-point Smoothing Filter

% Image selection: "Lion.bmp"
B=imread("Lion.bmp");
K=mat2gray(B, [0 255]);
img_number = 1;

% Displaying original image
figure;
imshow(K);
title_text = sprintf('Initial Image Display');
title(title_text, 'FontSize', 14);
pause(0.5);

for M = [21, 31, 51]
    % column-wise smoothing operation
    Z = zeros(512,512);
    for column = 1:size(K,2)
        z_column = zeros(512,1);
        % setting window parameters
        middle = (M-1)./2;
        start = -1 * middle;
        end_window = middle;
        % going through each window element
        for j = start:end_window
            if (((j  + column) >= 1) && ((j + column) <= 512))
                z_column(:,1) = z_column(:,1)+ K(:,(j+column));
            end
        end
        % updating column with average values
        Z(:,column) = z_column./M;
    end
    % showing the processed image
    figure(img_number);
    imshow(Z);
    filter_size = M;
    title_text = sprintf('1-D Horizontal M-point Smoothing, M= %d', filter_size);
    title(title_text, 'FontSize', 14);
    img_number = img_number + 1;
end

%% Part 2: Adding Random Noise

for c = [0.4, 0.9]
    random_noise = rand(512,512) - (0.5 .* ones(512,512));
    random_noise = random_noise .* c;
    modified_K = K + random_noise;
    for M = [21, 31, 51]
        % column-wise averaging filter
        Z = zeros(512,512);
        for column = 1:size(K,2)
            z_column = zeros(512,1);
            % window calculation
            middle = (M-1)./2;
            start = -1 * middle;
            end_window = middle;
            for j = start:end_window
                if (((j  + column) >= 1) && ((j + column) <= 512))
                    z_column(:,1) = z_column(:,1) + modified_K(:,(j+column));
                end
            end
            Z(:,column) = z_column./M;
        end
        figure(img_number);
        imshow(Z);
        filter_size = M;
        title_text = sprintf('Noise Added, M= %d, c= %d ', filter_size, c);
        title(title_text, 'FontSize', 14);
        img_number = img_number + 1;
    end
end

%% Part 3: Frequency Response of Moving Average Filter
```

```
for M = [21, 31, 51]
    filter_length = M;
    omega = -pi:(pi/100):pi; % frequency range
    filter_coeff = [ones(1,filter_length)]/filter_length;
    old_filter = filter_coeff;
    [response,freq] = freqz(filter_coeff,1,omega);
    % phase shift adjustment
    response = response .* exp(((M-1)/2) .* 1j .* omega);
    new_filter = response;
    figure(img_number);
    plot(freq,abs(response));
    title_text = sprintf('Frequency Response, M= %d', M);
    title(title_text, 'FontSize', 14);
    xlabel('Normalised Frequency (x pi rad/sample)');
    ylabel('Magnitude (dB)');
    img_number = img_number + 1;
end

%% Part 4: First Difference Filter Implementation

Z = zeros(512,512);
for column = 1:size(K,2)
    z_column = zeros(512,1);
    for j = -1:0
        if (((j  + column) >= 1) && ((j + column) <= 512))
            if (j == 0)
                z_column(:,1) = z_column(:,1)+ K(:,(j+column));
            else
                z_column(:,1) = z_column(:,1) - K(:,(j+column));
            end
        end
    end
    Z(:,column) = z_column;
end

figure(img_number);
imshow(Z);
title_text = sprintf('First Difference Filter Applied');
title(title_text, 'FontSize', 14);
img_number = img_number + 1;

%% Part 5: Frequency Response of First Difference Filter

omega = -pi:(pi/10):pi; % frequency values
filter_b = [1 -1];
filter_a = [1];
figure(img_number);
[response,freq] = freqz(filter_b,filter_a,omega);
plot(freq,abs(response));
xlabel('Normalised Frequency (x pi rad/sample)');
ylabel('Magnitude (dB)');
title_text = sprintf('First Difference Filter Frequency Response');
```

```
title(title_text, 'FontSize', 14);
img_number = img_number + 1;
```