Dela Rosa, Joshua Gabriel
2019142208

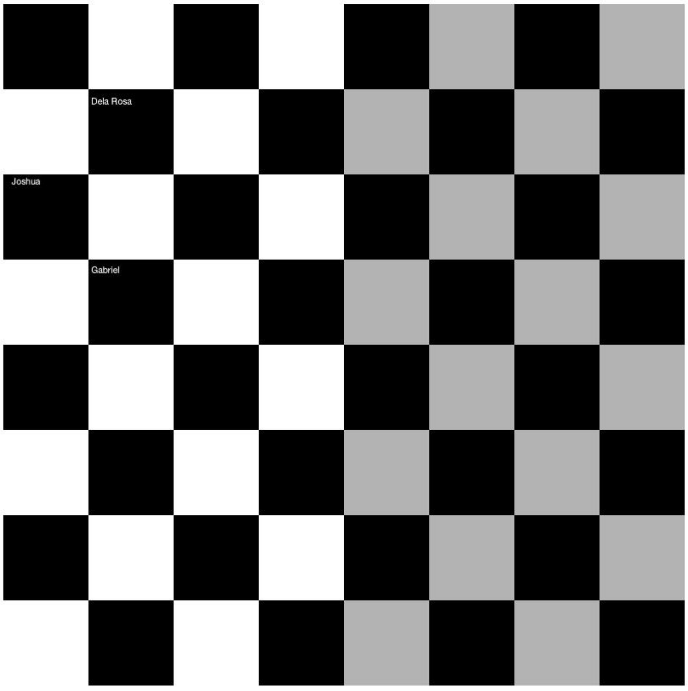
Signals and Systems Homework 2

Objective

In this homework, we wish to investigate the application of convolution in image processing. We will filter out an image with a given impulse response for problem 1 and further investigate filtering using convolution for problem 2.

Problem 1

The original image is as follow.



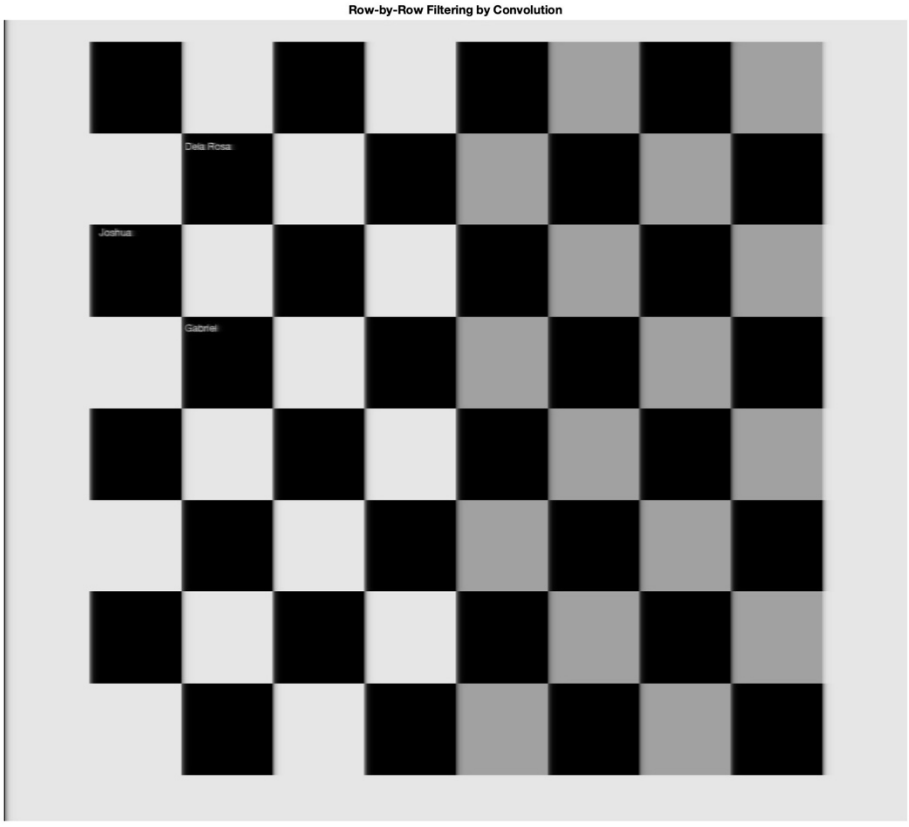
To solve problem 1, we start by creating the checkerboard and extracting the image X from the given code. Then, we define the impulse response of the filter. We also extract the image dimensions for the convolution process. The code for all these processes is show below.

```
%% Setup the Filter Info and Image Info

% Filter
n = -500:1:500;
h = 0.2*(0.8).^n.*us(n); % Custom Step Function from MJ Roberts Book

% Image
dims = size(X); % m x n
row_indx = dims(1); % m
col_indx = dims(2); % n
```

Next, we convolve each row of the image X with the filter response. To do this, a for loop is used to extract each row of the image matrix then it is convolved with the filter response. After convolving, the resulting row convolution is added to a blank matrix holder that would be the new image. The code for the row convolution as well as the resulting image (next page) is show below.

```
%% Extract each row and convolve
convRow = []; % New matrix holder for row convolutions
for i=1:row_indx
    current = X(i,:);
    res = conv(current,h,'same');
    convRow = [convRow;res];
end
figure; imshow(convRow); title('Row-by-Row Filtering by Convolution');
```

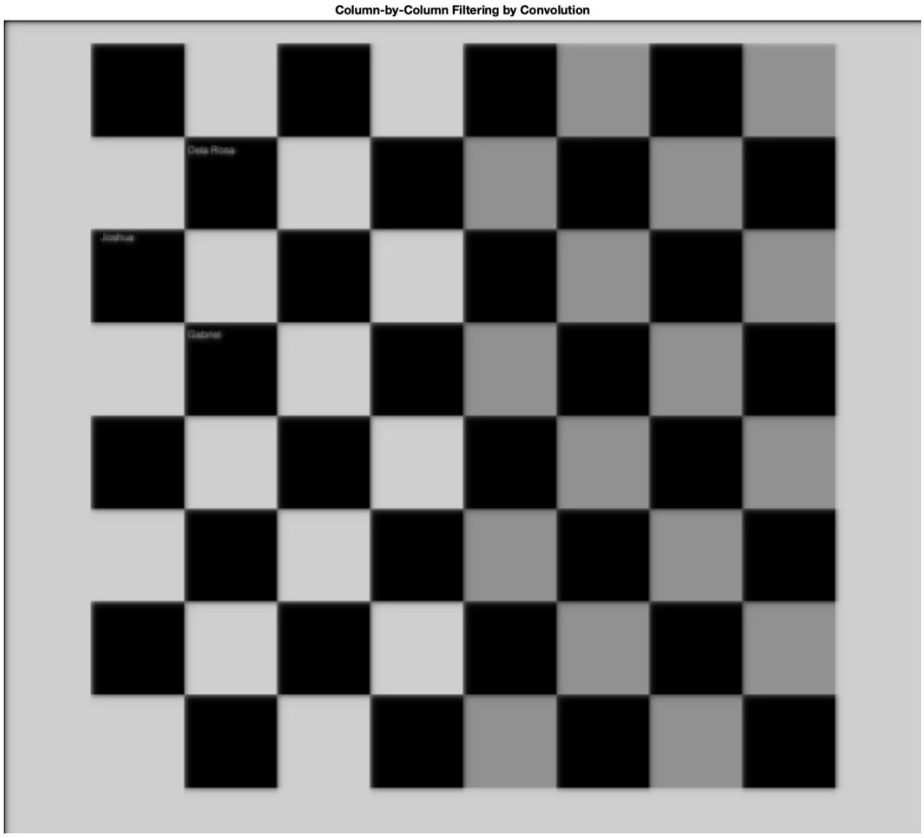
To further explain the code above, convRow will hold the new image matrix for the filtered row-per-row convolution. On the conv() function, we set the convolution to be the *same* which will set the resulting size to be the same with *current* thus keeping the size of convRow to be the same with the original image. Once a single row convolution is done, the resulting convolution is appended to the end of convRow.



Next, column-by-column convolution is done using the following code. We know from Linear Algebra that the transpose of a matrix changes rows to columns. Hence, we get the transpose of the image matrix above then repeat the procedure for convolution. The resulting image is presented below (next page).

```
%% Extract each column and convolve
convCol = []; % matrix holder for column convolutions
convRowT = convRow';
for i=1:col_indx
    current = convRowT(i,:);
    res = conv(current,h,'same');
    convCol = [convCol;res];
end
filtered = convCol';
figure; imshow(filtered); title('Column-by-Column Filtering by
Convolution')
```

Similar with the row-by-row convolution, the same working principle is applied in here; except that we convolve the transpose of the row-filtered matrix to obtain the columns as rows. Once the whole image is convolved, we return to the original orientation by transposing convCol.

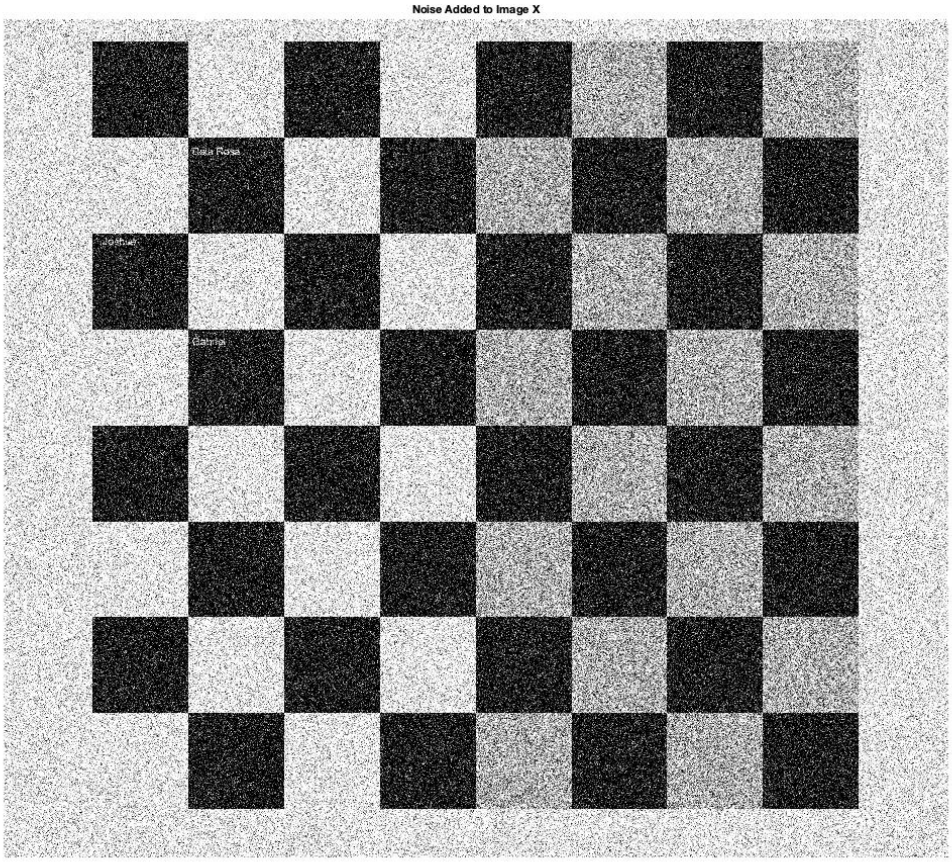


Problem 2

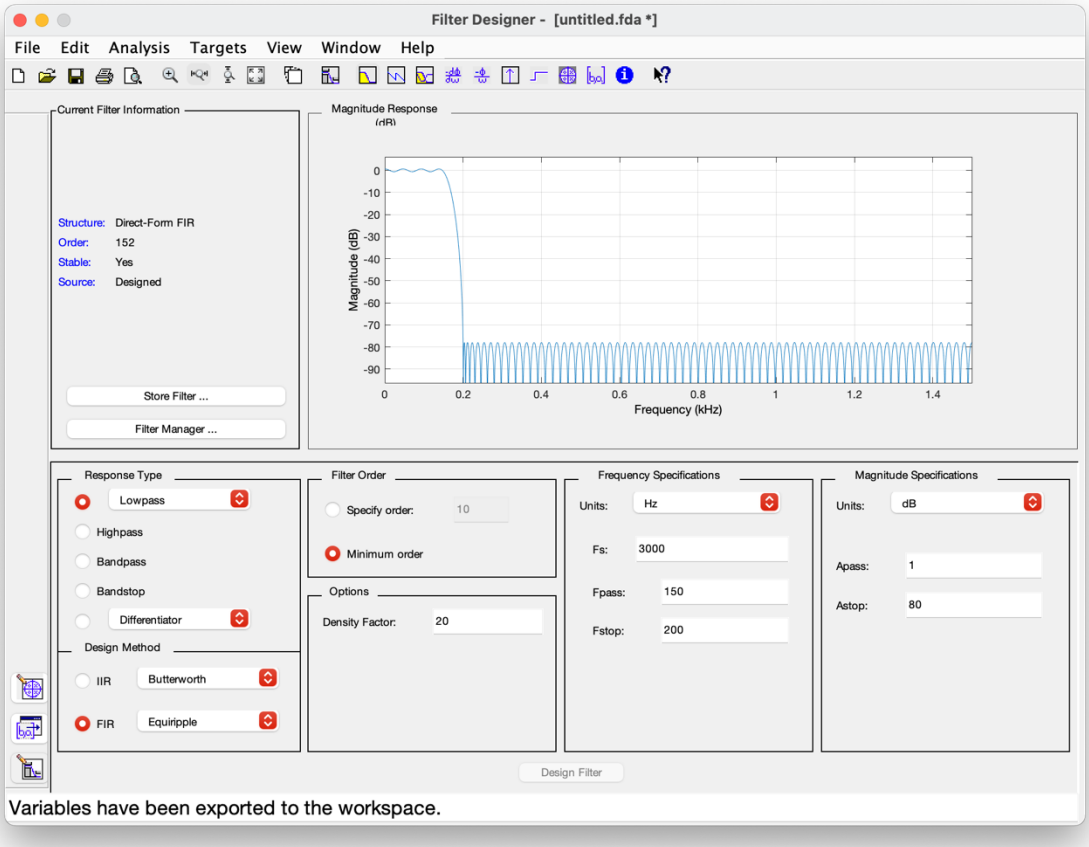
For this problem, we wish to add noise to the image X and filter out the noise using a filter. To do this without using any complex function from the Image Processing Toolbox, we must make our own filtering tool without using *imfilter* as well as the noise generator without using *imnoise* function by using other tools available on the software.

For the noise adding part, we just add some random noise with a small magnitude to the image matrix by using randn() function. The code for adding the noise is shown below as well as the resulting noisy image.

```
%% Add noise to Image
noisy = (1/3).*randn(size(X)) + X; % low magnitude noise
figure; imshow(noisy); title('Noise Added to Image X');
```

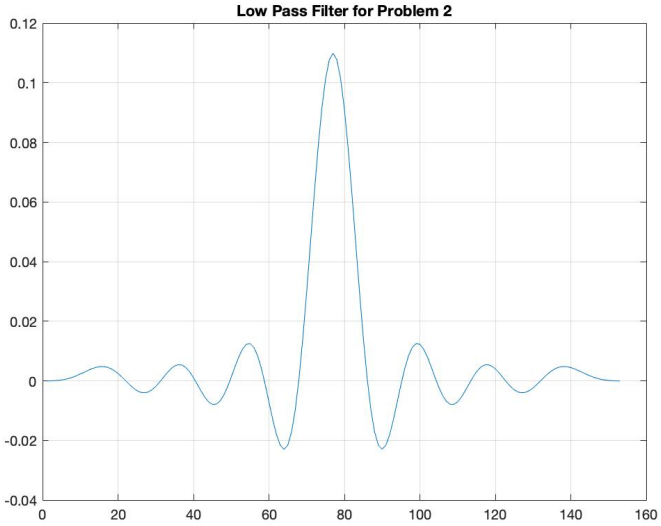


To remove the added noise, we just choose to use a low pass filter to smoothen the image. Since we only added simple noise a simple low pass filter would suffice. That is, any low pass filter should do. To obtain a simple low pass filter, we used MATLAB's filter designer program. The characteristics and creation of this simple low pass filter is shown below.



Filter Designer window for our simple LPF

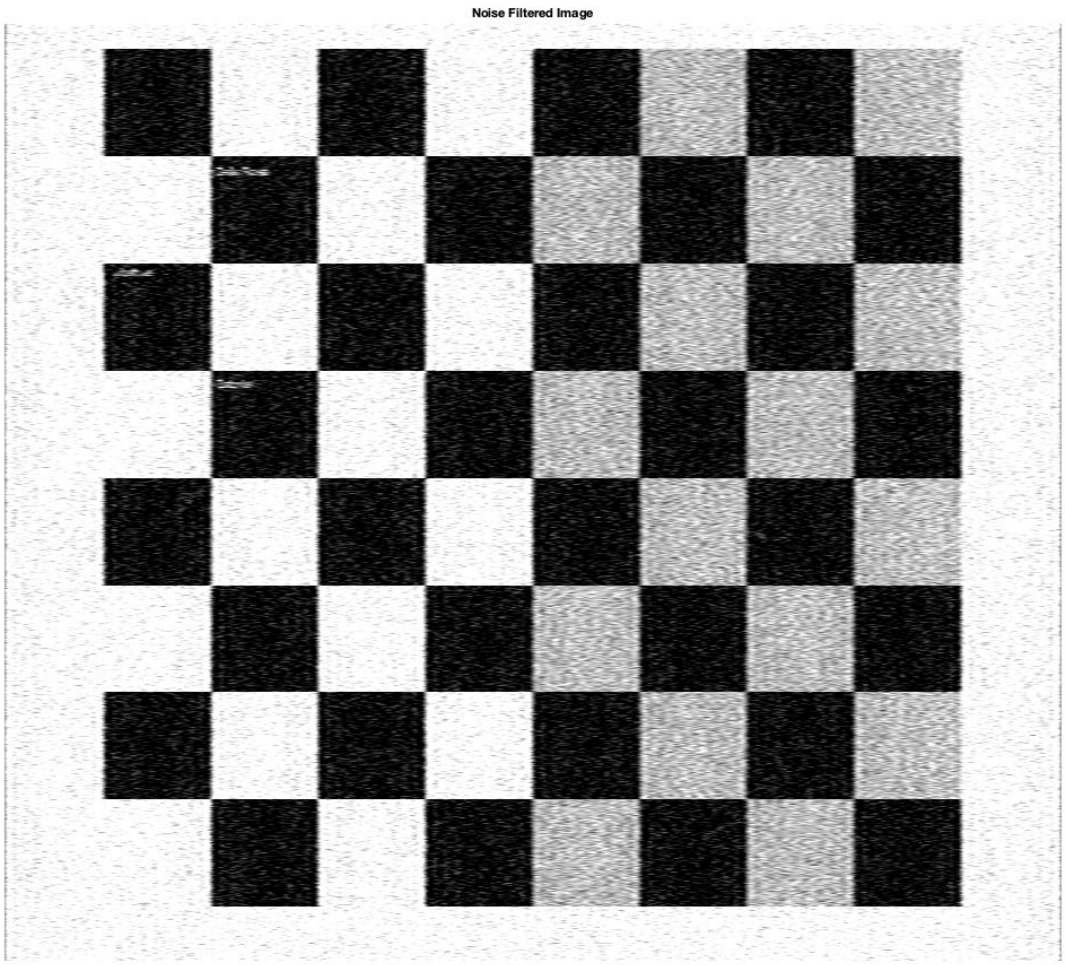
The resulting filter characteristic in time domain shown below. As expected, a low pass filter is a sinc function in time domain.



Now that we have our simple filter, we can now use this to filter the noisy image using convolution. We use the following code below. The resulting filtered image is shown below.

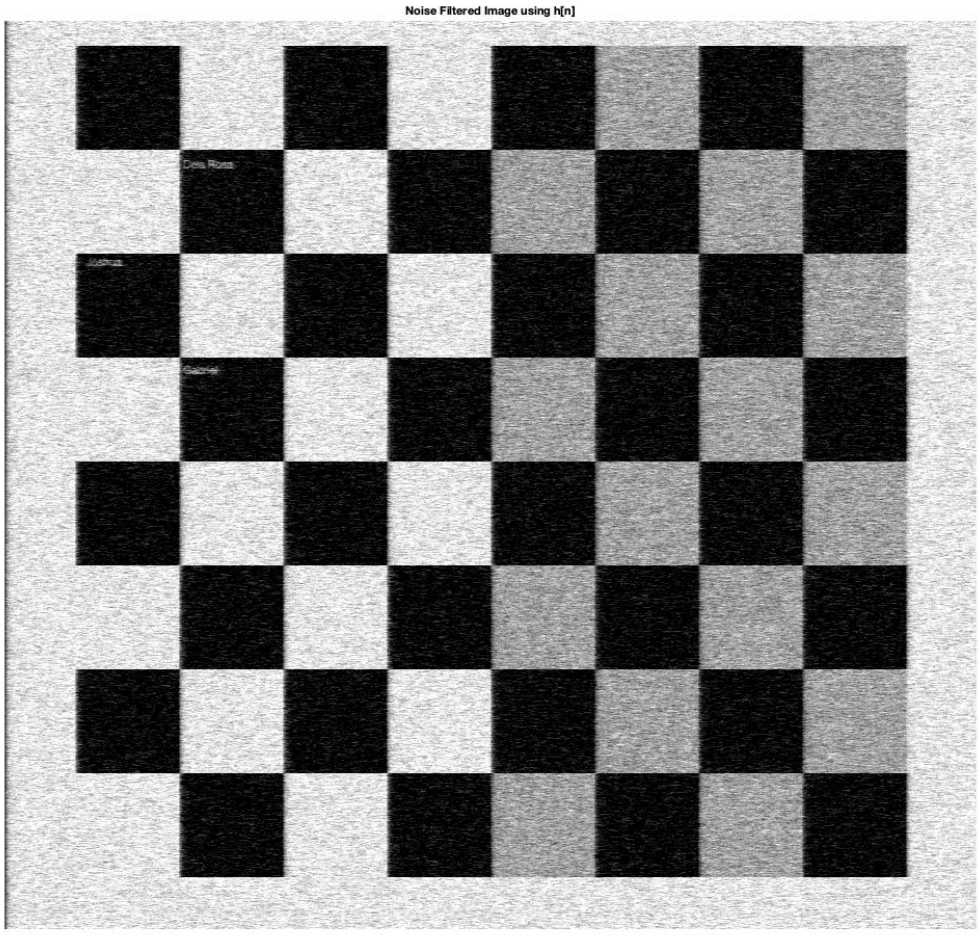
```
%% Filter noise
LPFilter; %custom made LPF,
% this filter is not in any way special. just a low pass filter
% as the noise is not in any way special
figure; plot(lpf); title('Low Pass Filter for Problem 2'); grid
F = convn(noisy,lpf,'same');
figure
imshow(F); title('Noise Filtered Image');
```

Based on the results, we see that the noise had been reduced by the simple low pass filter. However, since low pass filters blur the image, some information particularly the small details can be blurred as well for instance the name written on the checkerboard. However, the general picture of a checkerboard remains. This is a tradeoff that results from filtering using a low pass filter.



We can also look at the filter using the given filter impulse response on problem 1 as it too, is a low pass filter. Hence, the code for this second filter is shown below as well as the resulting image on the next page.

```
%% Filter noise with given h[n]
figure
plot(n,h); title('Given Low Pass Filter h[n] for Problem 2'); grid
F_h = convn(noisy,h,'same');
figure; imshow(F_h); title('Noise Filtered Image using h[n]');
```



As we can see, this did reduce some noise but not as good as the first one on the checker box. However, quite noticeably, there is some improvements on the written name on the figure.

Conclusion

For problem 1, we saw that a low pass filter blurs the image. That is, it smoothens the image. On problem 2, the addition of random noise added high frequency values on the image. That is, we observed that high frequency values are sharp edges or quick changes in between each value.

As observed in problem 2, we can design a filter depending on the need or the problem. For instance, if we wanted a clearer picture of the checkerboard without much importance on the written name, then we can proceed with the first filter. But, if we wanted more on the name, we could choose the second one. Either way, both filters filtered out the noise on the images.