**5a)** Since the noise that corrupts the signal is frequencies at $0.125\frac{p}{x}$ and above then an appropriate cutoff frequency for a brick wall filter would just be $0.125\frac{p}{x}$.

Since it's a brick wall filter so the transition will be instant therefore, the transfer function is:

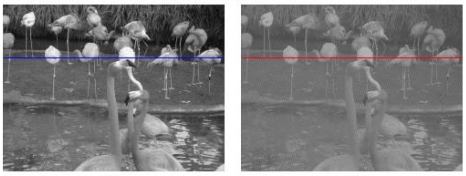
$$H(f) = \begin{cases} 1, & if\ |f| \leq 0.125\frac{p}{x} \\ 0, & if\ |f| > 0.125\frac{p}{x} \end{cases}$$

Impulse response/time domain representation is given by:

$$h_d[n] = \frac{\frac{w_c}{\pi}\sin(w_c n)}{w_c n}$$

$$h_d[n] = 2f_c sinc\left(\frac{2f_c\pi}{\pi}n\right)$$

$$h_d[n] = 2 * 0.125 * sinc(2 * 0.125 * n)$$



(a) (b)

Figure 1: The original image (a), and the image corrupted with high frequency noise (b). The coloured horizontal lines indicate the line of pixels corresponding to the intensity plots in Figure 2.

**Question 5. Image Denoising** **(30 marks)**

The Commonwealth Society of Ornithological Research and Investigation (CSORI) have set up a live camera feed in the remote wetlands of Far North Queensland to monitor a newly discovered native bird species (Phoenicopterus oculorum, see Figure 1a). Unfortunately, the live footage is corrupted with high frequency noise during transmission, such that only the blurry image shown in Figure 1b is received. Use your knowledge of digital filtering and convolution to help the researchers at CSORI denoise the signal and recover the original image.

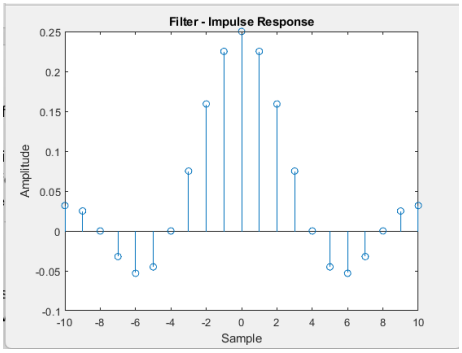
In `MATLAB`, load the `.mat` file attached to Problem Set 2. The file contains two variables: `imdata`, which is the original greyscale image, and `noisy_image_vec` which is the corrupted image transmitted to the CSORI. Note that `noisy_image_vec` is a vectorised image. i.e. if the image is of size $[m, n]$, then the vectorised image is of size $[1, mn]$. The first row of pixels in the image corresponds to the first $1 : n$ elements of the vector, the second row corresponds to the next $n+1 : 2n$ elements, and so on. Pixel intensities are all normalised to lie in $[0, 1]$.

The following steps will guide you through the design and analysis of digital finite impulse response filters that operate in the pixel domain. Attach your code in your submission.

**(a)** You have been told that the noise corrupting the broadcast is exhibiting frequencies at and above 0.125 $px^{-1}$. What is an appropriate cut-off frequency for a low-pass brick wall filter to remove this noise? Write its transfer function in the pixel domain. (3 marks)

**b)** Code for both b and c on the right

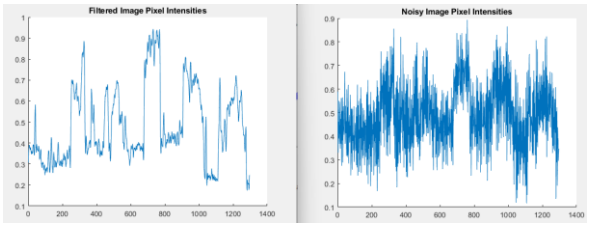
digital finite impulse response filter:



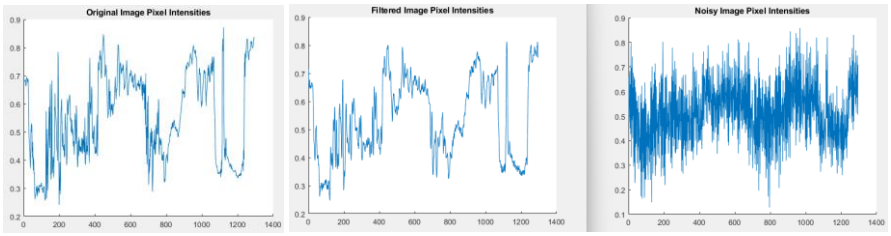

```
% Load the data
load('Q5_data.mat');

% Reshape the noisy image vector into a 2D matrix
noisy_image = fliplr(rot90(reshape(noisy_image_vec, 1296, 972), -1));

% Define the FIR filter parameters
cutoff_frequency = 0.125;
window_size = -10:10;
filter = 2 * cutoff_frequency * sinc(2 * cutoff_frequency * window_size);

window = numel(window_size);
fir_filter = filter .* window;

% Convolve noisy image with FIR filter
filtered_img = conv(noisy_image_vec, fir_filter, 'same');
filtered_img_rescaled = rescale(filtered_img);
filtered_img_corrected = fliplr(rot90(reshape(filtered_img_rescaled, 1296, 972), -1));

% Plot the FIR filter's impulse response
figure;
stem(window_size, filter);
title('Filter - Impulse Response');
xlabel('Sample');
ylabel('Amplitude');

% Plot the filtered image
figure;
imshow(filtered_img_corrected, []);
title('Filtered Image');

% Plot the filtered image
%figure;
%imshow(noisy_image, []);
%title('Noisy Image');
```

**c)** Convolved filter with the noisy image, result:

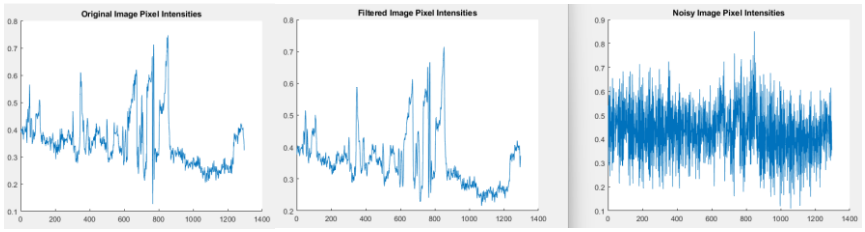
**d)** Row 310:



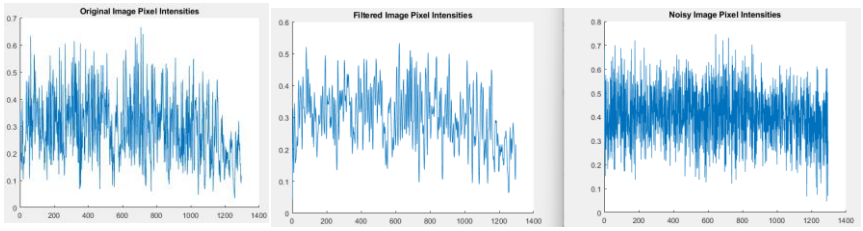

Row 970:




Row 640:




Row 1:




The filtering has effectively removed/attenuated the high-frequency components present in the noisy image while preserving the lower-frequency components from the noisy image and looks exactly like the one provided.

Comparing the Row 310 that was plotted to the one provided there seems to be a delay causing a shift. Other than that, The filtered image pixel intensities in retain the general shape and pattern of the original image, with distinct peaks and valleys corresponding to the underlying signals in the original image. However, some of the finer details and sharper transitions present in the original image have been smoothed out due to the filtering.

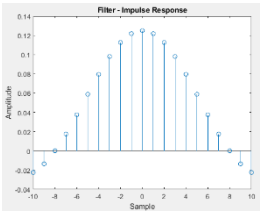
Although hard to see in Row 310 it is scaled down a bit where the original had intensity going to 1 whereas the one that was plotted goes to around 0.9 to 0.95. The Significant changes can be seen on row 1 where most intensities are between 0.2 to 0.7 whereas the filtered is around 0.2 to 0.45.

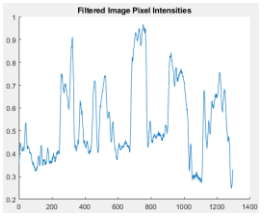
Analysis of the other rows do show the pixel intensities are attenuated, which correlates to the decrease in image sharpness as higher frequencies are being cut. Row 1 seems to be one of the worst not only differing in magnitude of intensity but also looking like it has less frequencies.

**e)** The filter moves across the image, and each value in the filtered image is computed by multiplying the filter with the corresponding pixels in the input image and summing them. Therefore, on the right side of graph there is zero padding to allow the filter to be centred over the input and help stop the edge effects or distortions. Due to the convolution a small shift has occurred to the filtered pixel intensities. The ‘same’ mode of convolution ensures that the output has the same dimensions as the input, making it easier to compare results and align outputs in sequential operations.

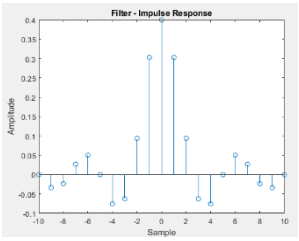
Convolution with the filter and the sinc function, will produce a filter with ringing, know as the Gibbs phenomenon, which could explain the differences between the original and the filtered response.

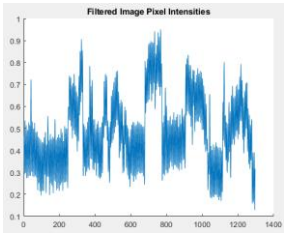
**f)** Cut-off frequency of 0.625


Cut-off frequency of 0.2

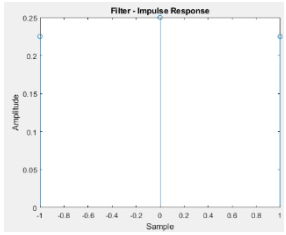



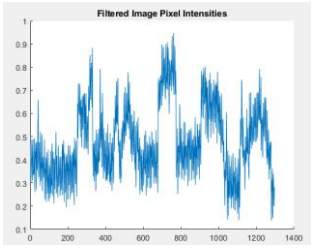

With a lower cut-off frequency, the pixel intensity for row 310 is much smoother than the one provided for the normal image meaning that not only is the noise being filtered out but also vital information for the image, therefore producing a very smooth image that it become blurry.

With a higher cut-off frequency, the pixel intensity still retains most of the high frequency noise, this means the image retains its detail but also the noise, which is still present, and causes diagonal lines to form.

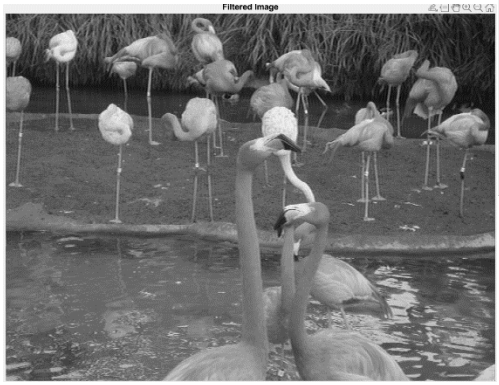
As the noise frequency starts from 0.125, my filtered image used that as the cut off. To get a balance of retaining the details and smoothness while not having the noise still present or getting the image blurry.

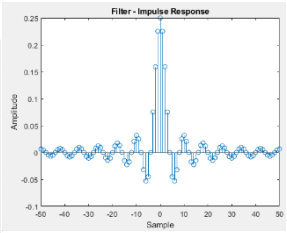
**g)** Window from -1:1

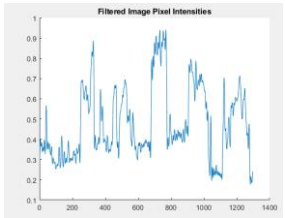




Window from -50:50




For frequency domain analysis, a larger window size means a narrower filter response in the frequency domain. This can lead to a steeper roll-off at the cutoff frequency. With the larger window from -50:50, the pixel intensity for row 310, on the peaks look a bit smother, which is slightly smother than the one provided for the original image. This is due to more data points that are taken into consideration for each computation. This leads to a bit of distortion and effects the sharpness of the image, where the colours don't blend in nicely.

With a smaller window size meaning fewer data points are involved in the computation. This results in narrower smoothing, retaining more high-frequency information, which preserves the details of the noisy image.

It shows with a smaller window size you preserve more detail and as you increase the window size you get more noise reduction, therefore having to find the balance in between them -10:10 was chosen, which preserved enough detail with sufficient noise reduction.