Nicholas McClellan

November 1st, 2019

EENG310

Matlab Assignment #4

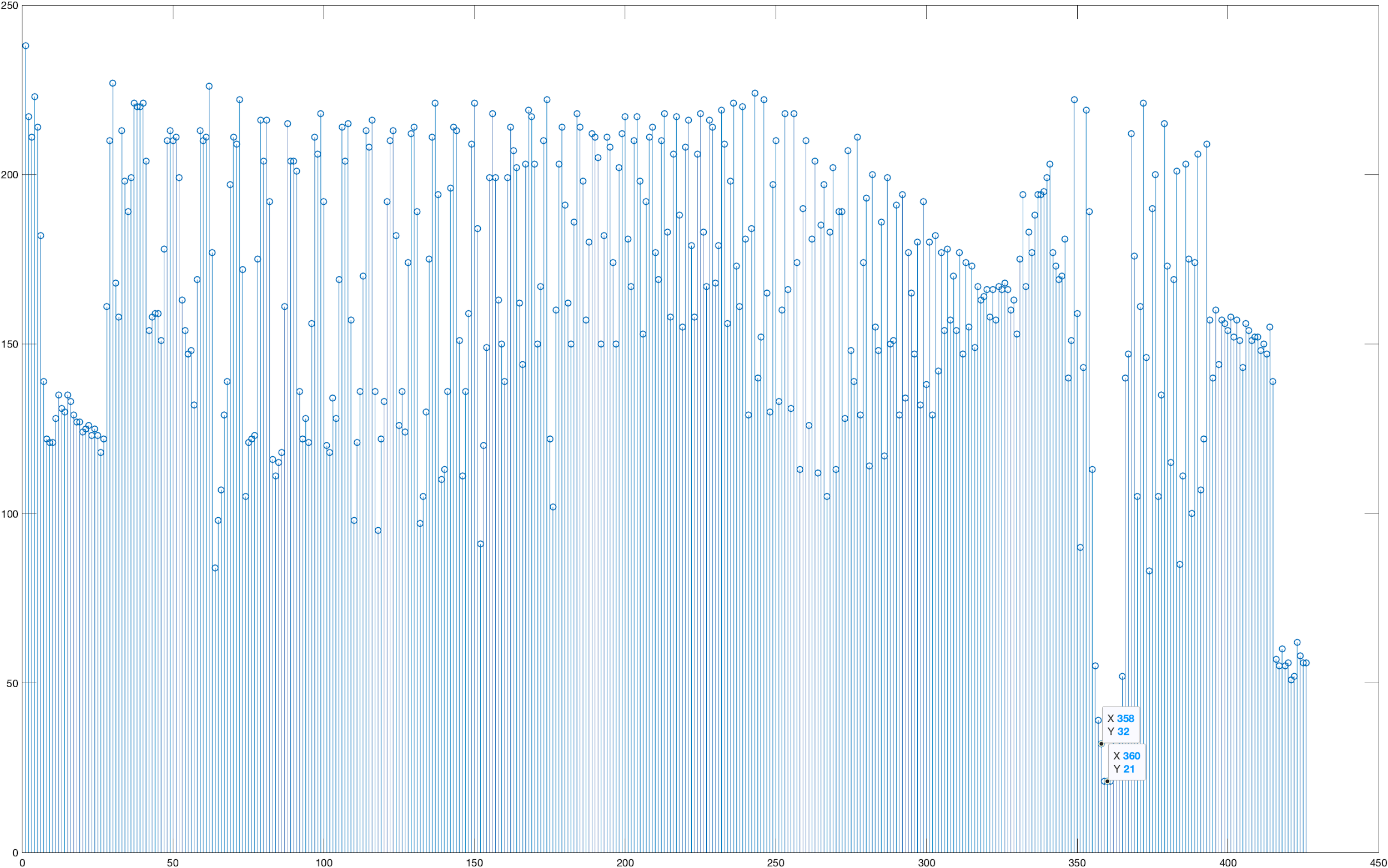
\*\*I tried to not resize any images in this document, as I imagine google docs does some kind of interpolation on its own. The formatting could probably be better, but I was not sure if we could resize. \*\*

**Problem 1.**

**(a)** **Figure 1. lighthouse.mat via imshow(xx, [0 255])**

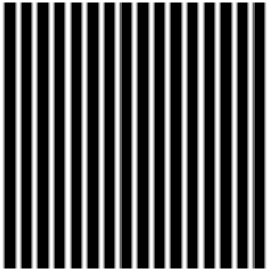
****

**(b) Figure 2. 200th Row via plot(xx(200,:))**

****When looking at what I believe to be a stem plot of the columns and their respective grayscale values, 0 represents the absolute black and 255 is absolute white. The colors still look relatively the same within about 30 values from their absolute color. This means that the colors have a range between (220-255) for white and (0-30) for black. The column where it crosses the binocular stand is about 370, where we see the very sudden drop in the plot.

**Problem 2.**

**(a) Figure 3. Synthetic Image via xpix = ones(256,1)\*cos(2\*pi\*(0:255)/16);**

****

The greyscale pattern that we see is due to the periodic behavior of the cosine function, that is that the values alternate between . Further, the resulting image has black bands when the result of the xpix expression is positive, and white bands when the result is negative. The number of bands can be determined by the division of 256 (the number of elements in the matrix) and the divisor of the cosine function (16).

This observation was confirmed by plotting the same function with different divisors and counting the number of bands in the new image. Further, the size of the black segment of the band is 12 pixels, from 0-11, and the white segment is 4 pixels, from 12-15. This pattern is then repeated over the 256 pixels to create the above image.

**(b)** I believe we could generate an image with horizontal bands in the same fashion, but perhaps transposing the resulting matrix. Similarly, we could scale the divisor such that the resulting expression is separated into eight bands.

**Figure 4. Horizontal bands via xpix = ones(400,1)\*cos(2\*pi\*(0:399)/50) & imshow((xpix)’)**

****

**Problem 3.**

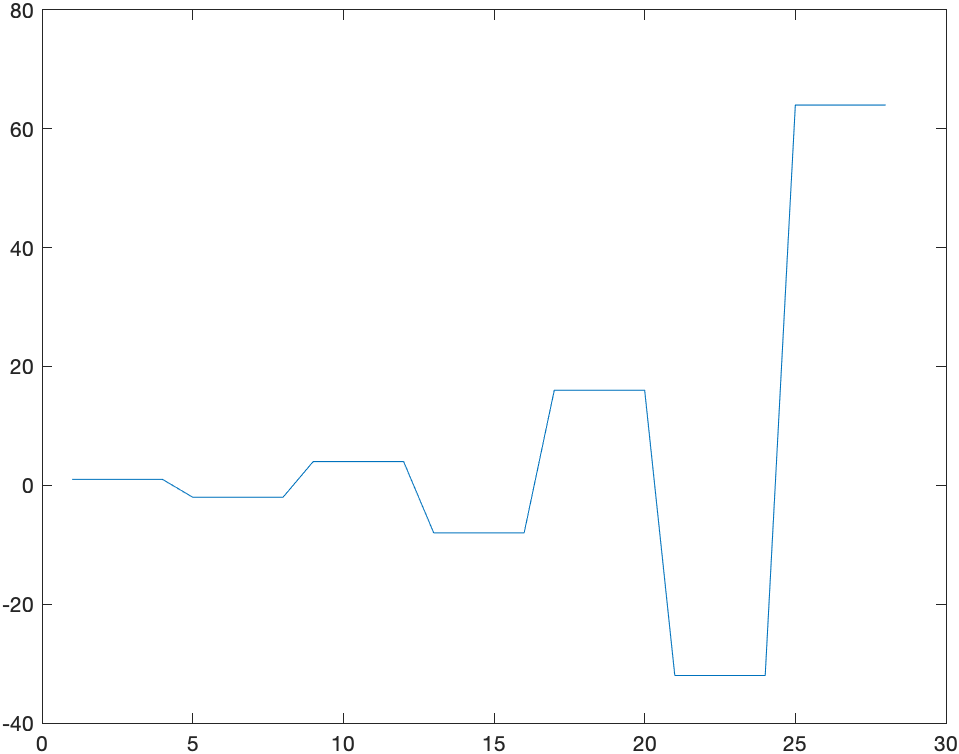
**Figure 5. Downsampled lighthouse by a factor of 2**

****

In the original image, it is apparent that the spatial frequency of the pickets in the fence increases from left to right, as they appear to be closer together. Similarly, the fence in the downsampled image appears less clear as the spatial frequency increases from left to right due to aliasing. Further, the aliasing is caused due to a lower sampling rate (downsampling by a factor of 2). According to the Nyquist Sampling Theorem, if the sampling rate is less than where is the spatial frequency, aliasing will be seen. This explains the indistinguishable pickets a bit right of center, and again at the furthest right side of the fence.

**Problem 4**

**Figure 6. Zero-Order hold of test vector “xr1”**

****

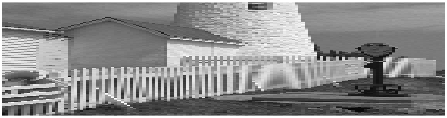
The ceil() function rounds the result up to the nearest integer and is used such that the operation effectively creates four copies of the integers from 1 to L, the length of our signal vector. More specifically, the vector “nn” contains [1 1 1 1, 2 2 2 2, . . . 7 7 7 7]. If we treat the vector “xr1hold” as the interpolated version of xr1, the interpolation factor would be 4 as it contains four times as many elements.

**Problem 5**

**Figure 7. Downsampled (3x) lighthouse**

****

**Figure 8. The interpolation (3x) of the rows of the downsampled image “xx3”**

****

Comparing the images, this new image is now 109x426, as opposed to the original dimensions 109x142. Moreover, this new size is the same dimension as the original lighthouse image, after downsampling by a factor of 3. The content of the image at this point is still highly distorted, but it is apparent that many of the key features of the image have started to take form.

**Problem 6**

**Figure 9. The interpolation (3x) of the columns of the downsampled image “xholdrows”**

****

To compare this image with the original lighthouse image, first I will compare the sizes. The original image was 326x426 and the final interpolated image had dimension 327x426. Although the sizes of the image are not exactly the same, it was almost regenerated to nearly the same size image. Further, looking at the contents of the image, it is clear that there is a heavy blur to the interpolated image. This is due to aliasing, although this is a zero-order hold, aliasing cannot be eliminated. Further, the sharpness or clarity of the image is lost, but most of the distinguishing qualities are present, such as the dark-colored viewer and the bigger objects in the background.

**Code Segments for Problems 5 & 6:**

|  | I did not realize the code needed to be submitted so I had cleared much of my workbench, this is the reconstruction. |
| --- | --- |

**Problem 7**

**Figure 10. Linearly interpolated reconstruction of original lighthouse image**

****

Similar to the previous reconstruction, this image fails to capture the sharp edges and clarity of the first. Again, the size of this image is not exactly the same as the original, but it is within a few pixels. Further, the zooming process cannot remove aliasing effects from the downsampled image. This is similar to when you zoom in far enough on the original image, the same sort of “blur” can be seen due to the aliasing effect. The areas of low frequency are a bit more clear than the areas of high frequency, like the center and right sections of the fencing.