Taylor Rembos
EEL 3135
Lab 4 Part 1

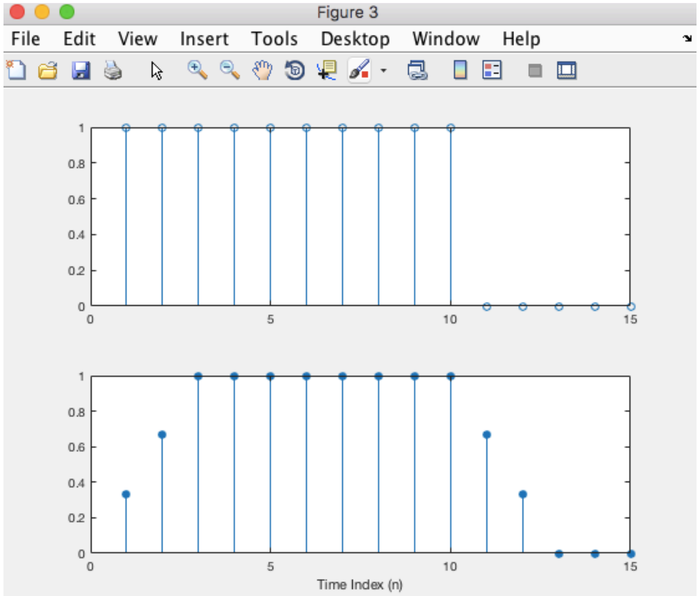
Lab 4: Echoes and Images

Lab Part One

1.1 Implementing the Three Point Averager

To illustrate the filtering action of the 3-point averager, run the above code and then make a stem plot of the input signal and output signals in the same figure using the following code:

```
>> xx = [ones(1,10), zeros(1,5)];

>> nn = 1:length(xx);

>> bk = [1/3 1/3 1/3];

>> yy = firfilt(bk, xx);

>> yy = conv(bk, xx);

>> figure(3); clf

>> subplot(2,1,1);

>> stem(nn, xx(nn))

>> subplot(2,1,2);

>> stem(nn, yy(nn), 'filled')

>> xlabel('Time Index (n)')
```



What characteristics of the input signal are most affected by the averager? What characteristics of the input signal are least affected by the averager? (in other words, how is the output yy the most and least different than xx?)

The characteristics of the input signal that are most affected by the average are the lowest and highest values. The average brings the lowest x values at a high initial value down and the highest x values at a value of 0 up. Specifically, the first two values, initially at a y value of 1, move down to be around 0.35 and 0.7. The last 5 values, initially at a y value of 0, move up to be at a value of around 0.7, 0.35, 0, 0, and 0.

The characteristics of the input signal that are least affected by the averager are the middle values. Specifically, the middle 8 values stay at a value of 1.

The following is the relation between the lengths of the input and output signals and the length of the coefficients vector:

```
length(yy) = length(xx) + length(bk) - 1
```

Did your input, output, and impulse response fulfill this relation? If so, explain why; if not, explain why not.

Length(xx) = 15, length(bk) = 3; length(yy) = 17. length(yy) = 15 + 3 -1 = 17. Therefore, yes the relation was fulfilled (done out below).

```
>> length(xx)
ans =
    15
>> length(bk)
ans =
     3
>> length(yy)
ans =
    17
```

1.2 An Echo Filter

>> load labdat.mat

. (a) Suppose you have an audio signal sampled at *fs*= 8000 and you want to simulate an echo. What values of *r* and *P* will give an echo with strength 85% of the original, with time delay 0.22 s? Implement this echo filter and use it on the signal in vector `x2` (Hint: see lab part 1.1).

r = 0.85, and 8000*0.22

```
>> r = 0.85

r =

   0.8500

>> bk = zeros(1,8000*0.22);

>> bk(1) = 1;

>> r = bk(8000*0.22);

>> yy = firfilt(bk, x2);

>> soundsc(yy, 8000)
```

Delay P = (100%-85%)*fs = 15%*fs = 0.15*8000 = 1200

1.3 Image Processing

1.3.1 Show a Test Image

```
>> load echart.mat
>> whos
 Name        Size          Bytes  Class    Attributes

 ans         1x1             8  double
 bk          1x1760      14080  double
 echart    257x256      526336  double
 h1          1x45          360  double
 h2          1x45          360  double
 nn          1x15          120  double
 r           1x1             8  double
 x1          1x100         800  double
 x2      24576x1        196608  double
 xtv       256x1          2048  double
 xx          1x15          120  double
 yy          1x1859      14872  double
```

```
function [ph] = show_img(img, figno, scaled, map) %SHOW_IMG
display an image with possible scaling

% usage: ph = show_img(img, figno, scaled, map) % img =
input image % figno = figure number to use for the plot %
if 0, re-use the same figure

% if omitted a new figure will be opened % optional args:

% scaled = 1 (TRUE) to do auto-scale (DEFAULT) % not equal
to 1 (FALSE) to inhibit scaling % map = user-specified
color map % ph = figure handle returned to caller

%----

end
```

>> show_img(echart, 257 ,1,colormap(gray(250)))

Image being scaled so that min value is 0 and max value is 255

ans =

Axes with properties:

XLim: [0.5000 256.5000]

YLim: [0.5000 257.5000]

XScale: 'linear'

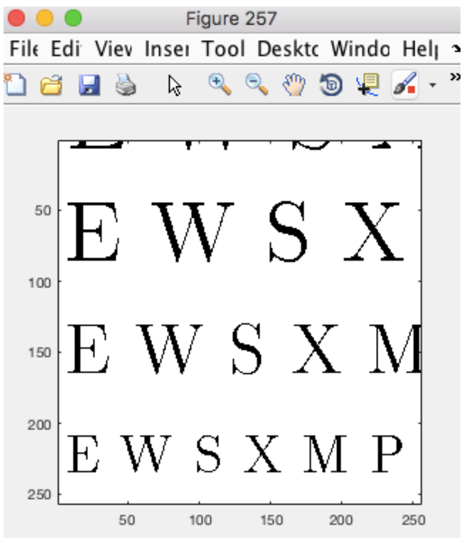
YScale: 'linear'

GridLineStyle: '-'

Position: [0.1308 0.1109 0.7734 0.8133]

Units: 'normalized'

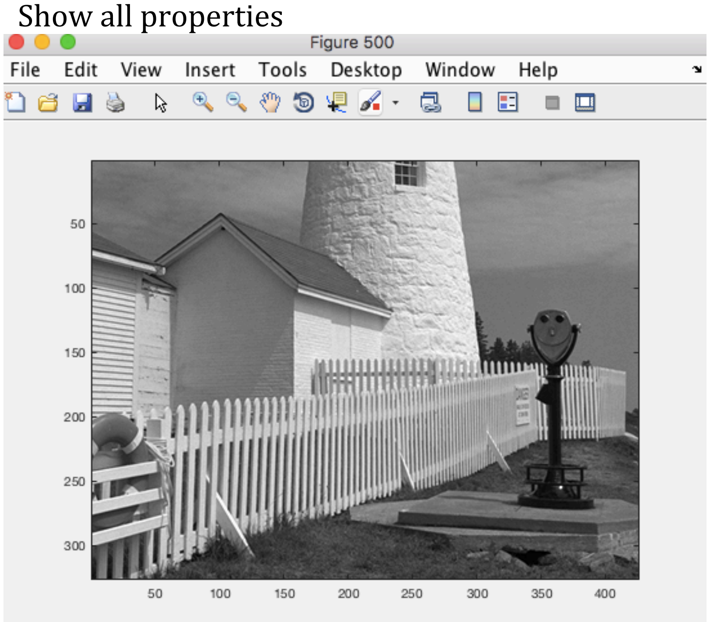
Show all properties



1.3.2 The Lighthouse Image

Load and display the 326 × 426 "lighthouse" image from `lighthouse.mat`.

```
>> load lighthouse.mat
>> show_img(xx, 500 ,1,colormap(gray(256)))
Image being scaled so that min value is 0 and max value is 255

ans =

  Axes with properties:

             XLim: [0.5000 426.5000]
             YLim: [0.5000 326.5000]
           XScale: 'linear'
           YScale: 'linear'
    GridLineStyle: '-'
         Position: [0.1302 0.1100 0.7745 0.8150]
            Units: 'normalized'

  Show all properties
```



Use the colon operator to extract the 225th row of the "lighthouse" image, and plot it as a discrete-time one-dimensional signal using the `plot()` function;

```
xx225 = xx(225,:);
```

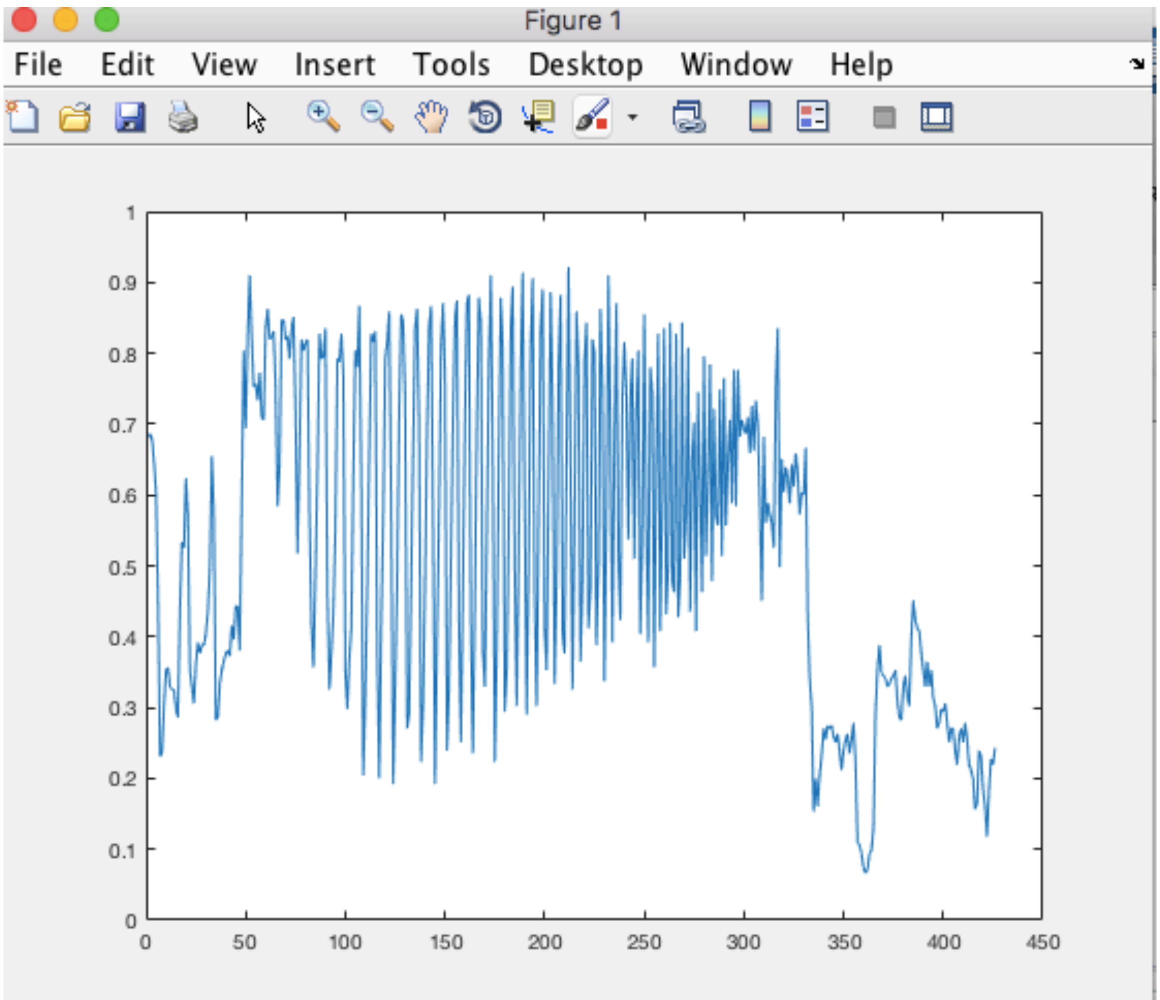
Observe that the range of signal values is between 0 and 1. Which values represent

white? Which values represent black? Where does the 225th row cross the fence? What features of the image correlate with the periodic-like portion of the plot?

Please annotate your plot to support your answer.

```
>> xx225 = xx(225,:);
>> plot(xx225)
```

The 1 values represent white and 0 represents black. The row crosses the fence at around 50 , as that is where the graph starts to oscillate rapidly. The periodic-like portion of the plot is an imitation of the fences white planks and black spaces.



1.3.3 Synthesize a Test Image

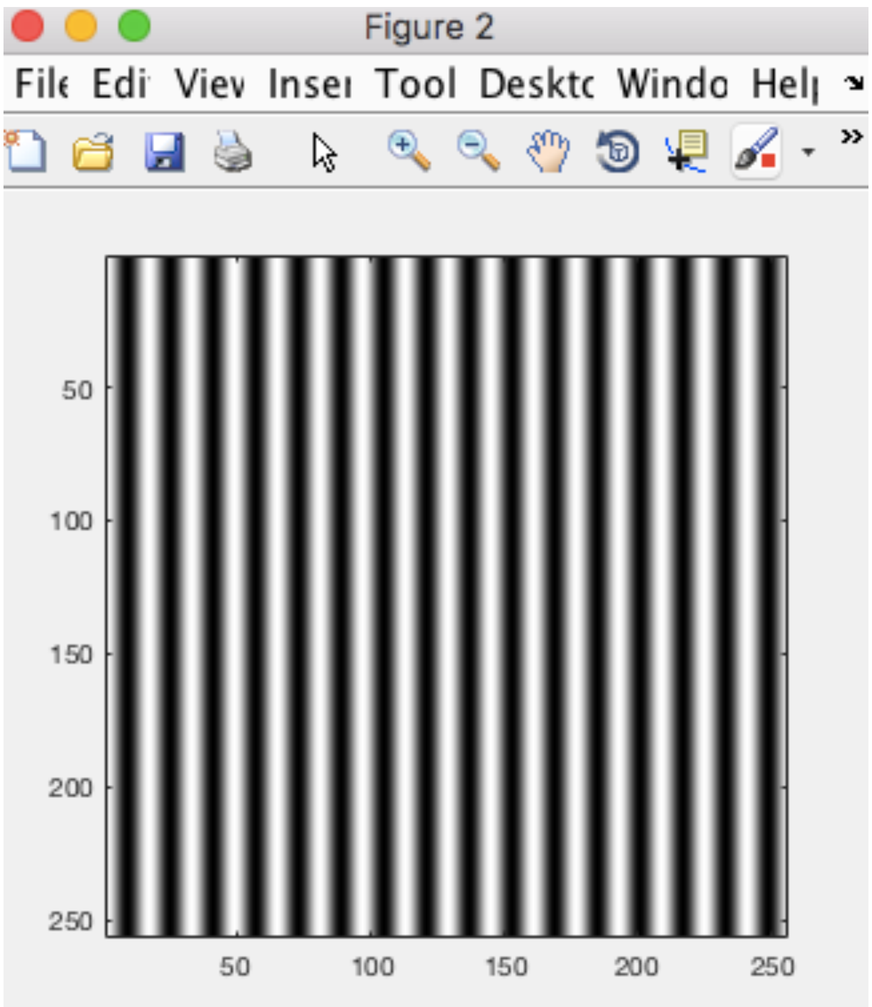
Display this synthetic image in which all of the columns are identical by using the following *outer product*:

xpix = ones(256,1)*cos(2*pi*(0:255)/16);

```
>> xpix = ones(256,1)*cos(2*pi*(0:255)/16);
```

```
>> show_img(xpix, 2 ,1,colormap(gray(256)));
```

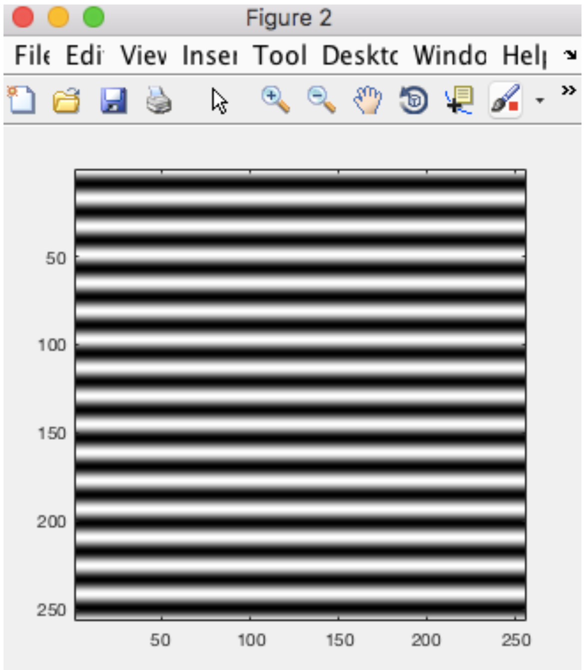
Image being scaled so that min value is 0 and max value is 255

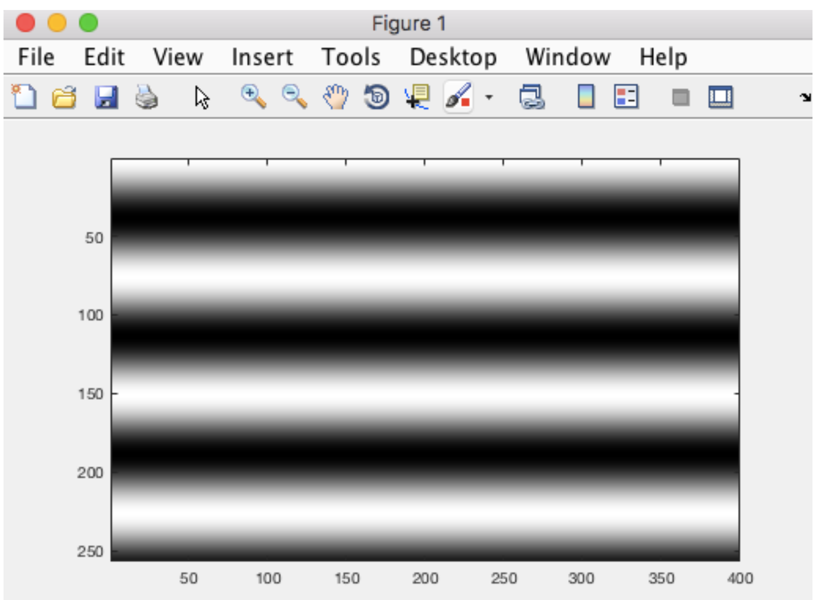


How wide are the bands in number of pixels? How is this width related to the formula for `xpix`? How would you produce an image with horizontal bands?

Create (and display/submit) a 450×450 image with 4 horizontal black bands separated by white bands.

From the formula for xpix, the length is 256, 256 / 16 = 16 which is how many pixels wide the bands are. To produce an image with horizontal bands, you could rotate the previous picture 90 degrees.





>> xpix = ones(400,1)*cos(2*pi*(0:400)/100);
>> show_img(xpix', 1 ,2,colormap(gray(256)));
Image being scaled so that min value is 0 and max value is 255

1.3.4 Sampling of Images

The following code down-samples by a factor of   :

wp = ww(1:p:end,1:p:end);

However, just as down-sampling a sinusoid can result in aliasing, down-sampling an image can also result in aliasing. Load the lighthouse.mat file (if it isn't loaded already).

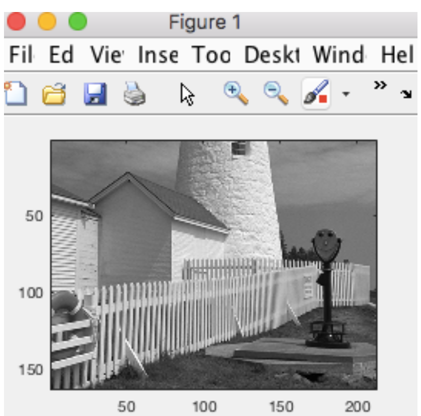
Down-sample the lighthouse image by a factor of 2, and show the image. Note that the image is not square. What is the size of the down-sampled image?

>> xx = ww(1:2:end,1:2:end);

>> show_img(xx, 1 ,2,colormap(gray(256)));

Image being scaled so that min value is 0 and max value is 255

The size of the down-sampled image is 163x213.



Notice the difference in appearance of the image, despite there not being any added points. Describe how the aliasing appears visually. Which parts of the image most dramatically show the effects of the aliasing? Why does the aliasing manifest itself in these places?

The fence and the bricks most dramatically show the effects of the aliasing, as they appear to be blurry or skewed slightly. Aliasing manifests itself in those places because in the rapid oscillation of the image from black to white, the down-sampling blurred the distinctions.

From your row plot and from zooming in on the image, estimate the frequency of the aliased features in cycles per pixel. When estimating spatial frequency, consider reoccurring features of the images as 'peaks'. From this you can obtain a period (how many pixels until the image 'peaks' again), and from there a frequency.

The frequency was down-sampled by 2 as well so it would go from 16 to 8Hz in cycles per pixel.

How does your estimation of aliased features fit into the Sampling Theorem? In other words, do the features that you expect to experience aliasing indeed do so, and why are those features aliased as opposed to others?

The sampling theorem frequency should be 16 but it is 8. In order to avoid aliasing, the frequency should be 2*16 = 32 Hz. The features that I expect to experience aliasing do so because the rapid change from black to white would cause a blur with the lack of sharper edges, which is why they are aliased as opposed to others.