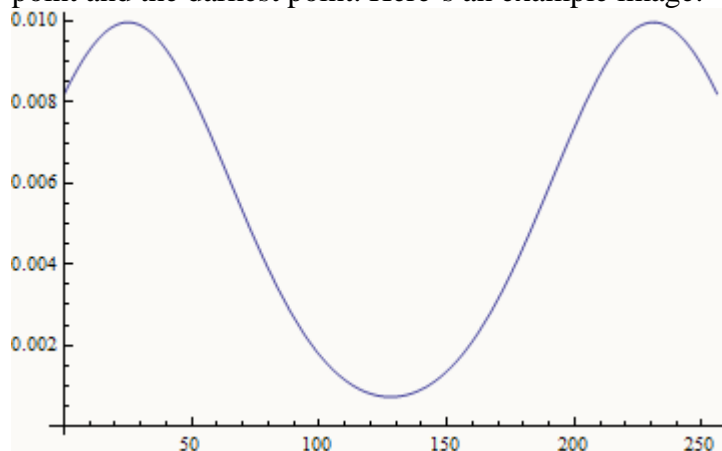


1. Convolution of  $f = [1, 3, -1, 2, 0, -3]$  and  $h = [-1, 3, -2]$ .
  - a. The number of samples in the output will be  $6+3-1 = 8$ . Shifting  $h$  and dot product with  $f$  yields an output  $[-2, -3, 10, -10, 7, 4, -9, 3]$ .
  - b. Use the synthesis and decomposition formulae to generate the functions  $\text{ReX}$  and  $\text{ImX}$ . Here  $N=6$ . Hence  $\text{ReX}$  and  $\text{ImX}$  will be defined for  $0..3$ .  

$$\text{ReX}[0] = \sum_{i=0..N-1} -f[i] = -2$$

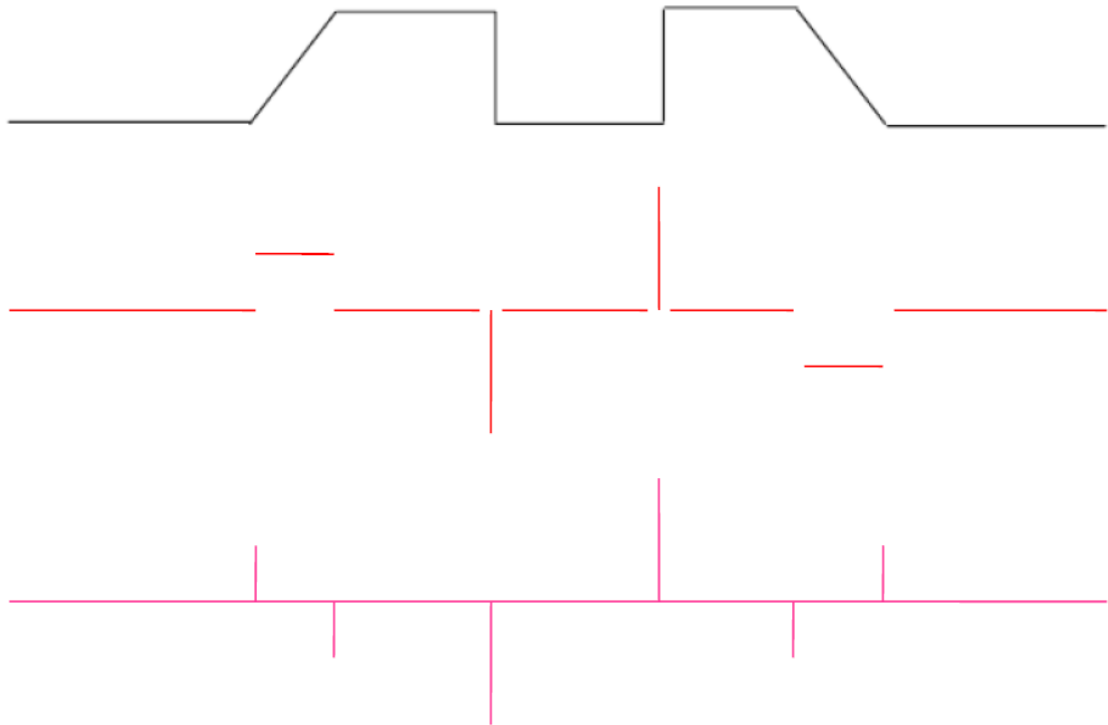
$$\text{ReX}[1] = \sum_{i=0..N-1} f[i]\cos(i*\pi/3)$$
 Similarly find  $\text{ImX}$ .  
 Convert to  $\text{MagX}$  and  $\text{PhX}$ . Multiply by multiplying  $\text{MagX}$  and adding  $\text{PhX}$ . Invert using synthesis equations.
2. 4 x 6 image at 300 dpi.
  - a. As per Nyquist sampling criteria, to sample frequencies of 300 dpi, we must sample at 600 dpi.
  - b. The spatial domain signal will then have  $N=600$ . The DFT will have values till  $N/2 = 300$ .
  - c. If we assume that the image has 600 dpi as its maximum frequency instead of 300 dpi, then again by the Nyquist rate we need to sample at 1200 dpi. This means sampling with a denser comb in the spatial domain. Denser comb in spatial domain indicates a sparser comb in the frequency domain. Sampling in spatial domain is a multiplication. Hence, it is a convolution of a sparser comb with a signal of same bandwidth as 300 dpi (since the frequency content is still 300 dpi). Hence, we will not see any aliasing artifacts. Further, we can use a reconstruction kernel of half the width, resulting in sharper pixels.
3. Segmentation of two “blob” images
  - a. To segment blobz1.png, we can use histogram-based segmentation. Imagine the histogram of the first image – it will have a lot of values at the low end and high end of the brightness range, and a low point in the middle, as there are comparatively few gray values between the brightest point and the darkest point. Here’s an example image:



So, to effectively segment the image, we can define the threshold to be the minimal point in the histogram. With the above sample histogram, it would fall around 127. Once this value is chosen, we simply iterate through each pixel of the image and color it black if its value is less than or equal to 127, and color it white if its value is greater than 127.

- b. Segmentation of the second image is slightly harder, since there is an illumination gradient. If the first image is multiplied by a smooth global gradient from dark to bright, the second image will result. Thus, the histogram in a local region over which the illumination is more or less constant will show the same property of two peaks with a minima, but this will not be true for the global histogram. Hence, this problem can be solved by applying a local histogram equalization algorithm first, with an appropriate neighborhood size. This will essentially expand the histogram of each region of the image, which will cause the entire second image to resemble the first image. After this is done, we can examine the entire image's histogram, which should look quite similar to the first image's histogram, and we can perform the segmentation in the same way as the first image.

4. Gradient and Laplacian of a 1D signal.



5. Low quality image correction

- a. Low contrast – image is too bright
- b. Image is blurred, edges are not very sharp
- c. Shot noise, specifically pepper noise, is present in the image

First, let's think about the basic things we need to do in order to reverse each problem with the image. To fix a low contrast image that is too bright, we need to perform histogram equalization on the image. Because the entire image suffers from the same symptoms, global histogram equalization should suffice. To fix a blurry image, we need to improve the amplitude of the high frequency signals so that they become more apparent, but we don't want to disturb the low frequency data that exists. To accomplish this, we can use high boost filtering, where the original image is scaled by a constant and a high-pass filtered version of the original is added back in. Finally, correction of the pepper noise requires a median filter, and we need to know the largest size pepper noise spot in order to determine the median filter's neighborhood size.

It is also worth looking at which operations may be destructive to others. For example, if we apply histogram equalization to the image first, the dark black of the pepper noise will adversely impact the cumulative histogram curve construction, and therefore impact the quality of the equalization. Similarly, if we high pass filter the image, the pepper noise will present high contrast information that is not supposed to be in the image, and it will impact the output adversely.

In consideration of these facts, we should first remove the pepper noise, then equalize the image's histogram, and then perform the high boost filtering.

6. JPEG quantization tables

- a. To quantize the DCT block by the given quantization table, each cell of the DCT block is divided by the corresponding cell of the quantization table. The output of the quantization operation, with normal floating point rounding applied, is:

28	2	1	0
2	1	0	0
1	1	0	0
0	0	0	0

- b. The table will be transmitted on a diagonal space-filling curve, similar to z-order or Hilbert curves. The table cells are transmitted in the following order, with the coordinates being (row, column):

(0, 0), (0, 1), (1, 0), (2, 0), (1, 1), (0, 2), (0, 3), (1, 2), (2, 1), (3, 0), (3, 1), (2, 2), (1, 3), (2, 3), (3, 2), (3, 3).

So the data transmitted from the above table would be:

28, 2, 2, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0.

- c. The sequence could be further compressed by applying run-length encoding, since there are values that repeat. For example, the block "1, 1,

1” might be condensed down to “3, 1” instead, and the long string of zeros could be converted to “7, 0”. This compression scheme works well because many values in quantized DCT tables are zeros, and the diagonal order linearization of the table helps put most of the zeros together at the end of the sequence.