3.1 (1A) Given: $\mathbf{x_1[n] = \delta[n] - 2\delta[n-1]}$

$$\mathbf{x_2[n] = 2\delta[n] + \delta[n-1] - \delta[n-2]}$$

3.1.1 Compute the linear convolution $\mathbf{x_1[n] * x_2[n]}$

$x_1[n] = [1; n = 0], [-2; n = 1]$ where $L_1 = 2$
$x_2[n] = [2; n = 0], [1; n = 1], [-1; n = 2]$ where $L_2 = 3$
$y[k] = h[n] * x[n] = \sum_{i=-k}^{L} x[i]h[k-i]$

$L = [2-1] + [3-1] = 3$

$y[0] = x_2[n] \bullet x_1[4-n]$:
$(x_2[0] \bullet x_1[0]) + (x_2[1] \bullet x_1[3]) + (x_2[2] \bullet x_1[2]) + (x_2[3] \bullet x_1[1])$=$2 \bullet 1 + 1 \bullet 0 + -1 \bullet 0 + 0 \bullet -2$
= 2

$y[1] = x_2[n] \bullet x_1[1-n]$:
$(x_2[0] \bullet x_1[1]) + (x_2[1] \bullet x_1[0]) + (x_2[2] \bullet x_1[3]) + (x_2[3] \bullet x_1[2])$=$2 \bullet -2 + 1 \bullet 1 + -1 \bullet 0 + 0 \bullet 0$ = -3

$y[2] = x_2[n] \bullet x_1[2-n]$:
$(x_2[0] \bullet x_1[2]) + (x_2[1] \bullet x_1[1]) + (x_2[2] \bullet x_1[0]) + (x_2[3] \bullet x_1[3])$=$2 \bullet 0 + 1 \bullet -2 + -1 \bullet 1 + 0 \bullet 0$
= -3

$y[3] = x_2[n] \bullet x_1[3-n]$:
$(x_2[0] \bullet x_1[3]) + (x_2[1] \bullet x_1[2]) + (x_2[2] \bullet x_1[1]) + (x_2[3] \bullet x_1[0])$=$2 \bullet 0 + 1 \bullet 0 + -1 \bullet -2 + 0 \bullet 1$
= 2

$$y[n] = \{2, -3, -3, 2\}$$

3.1.2 Compute 4-point DFT: **$X_1[k]$=DFT{$x_1[n]$}** and **$X_2[k]$=DFT{$x_2[n]$}**.

Note:

a. DFT should be computed in numbers, following the DFT example provided
b. You can use Matlab to help in calculation, without using Matlab DFT function
c. Show all your steps.

$x_1[k] = \sum_{n=0}^{N-1} x_1[n] \bullet W_N^{k_n}$ where $N = 4$ and $W_N^{k_n} = e^{-j\frac{2\pi k_n}{N}}$

$x_1[0] = x_1[0] + x_1[1] + x_1[2] + x_1[3]$

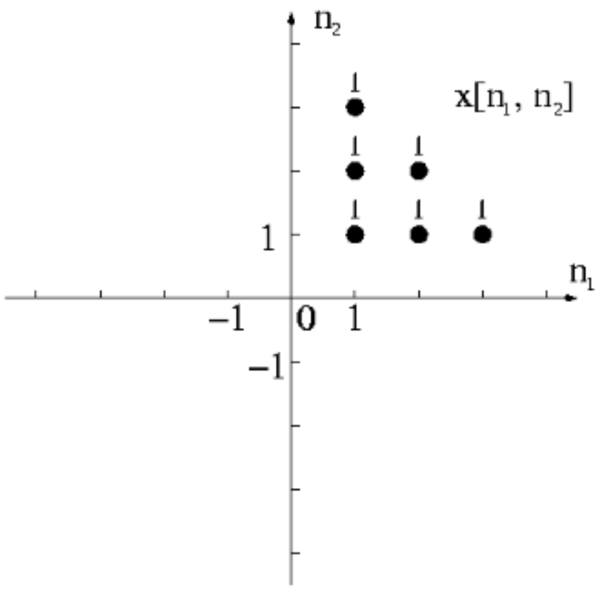
$x_1[0] = 1 - 2 = -1$
$x_1[1] = 1 + (-2)W_4^1 = 1 + (-2)(-j) = 1 + 2j$
$x_1[2] = 1 + (-2)W_4^2 = 1 + (-2)(-1) = 3$
$x_1[3] = 1 + (-2)W_4^3 = 1 + (-2)(j) = 1 - 2j$

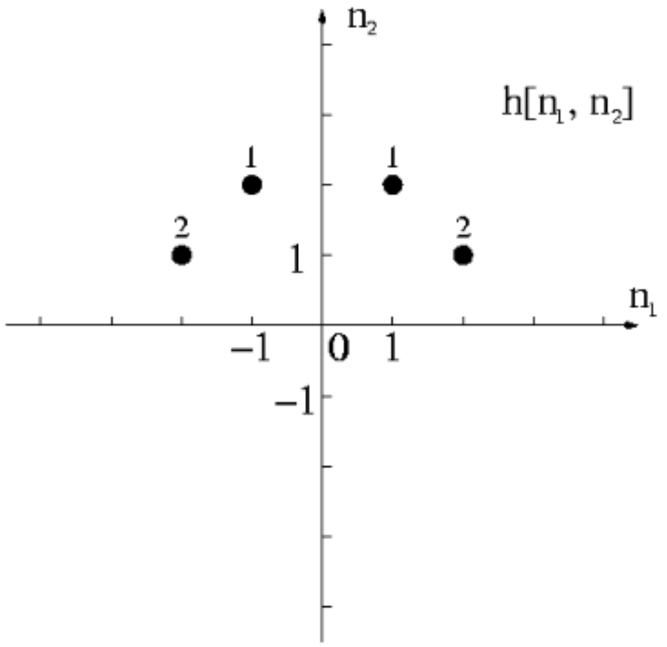
$x_2[k] = \sum_{n=0}^{3} x_2[n] \bullet W_4^{k_n}$

$x_2[0] = 2 + 1 - 1 = 2$
$x_2[1] = 2 + W_4^1 - W_4^2 + 0 = 2 - j + 1 = 3 - j$
$x_2[2] = 2 + W_4^2 - W_4^4 + 0 = 2 - 1 - 1 = 0$
$x_2[3] = 2 + W_4^3 - W_4^6 = 2 + j + 1 = 3 + j$

$$x_1[k] = \{-1, 1 + 2j, 3, 1 - 2j\}$$
$$x_2[k] = \{2, 3 - j, 0, 3 + j\}$$

3.2 (1A) Calculate the 2-D convolution **x[n,n]****h[n,n]**, show all the necessary intermediate steps.





$y[n,n] = x[n,n] * h[n,n]$
$x[1,1] = x[1,2] = x[1,3] = 1$
$x[2,1] = x[2,2] = 1$
$x[3,1] = 1$

$h[-2,1] = h[2,1] = 2$
$h[-1,2] = h[1,2] = 1$

$y[-1,2] = x[1,1]h[-2,1] = 2$
$y[-1,3] = x[1,2]h[-2,1] = 2$
$y[-1,4] = x[1,3]h[-2,1] = 2$

$y[0,2] = x[2,1]h[-2,1] = 2$
$y[0,3] = x[1,1]h[-1,2] + x[2,2]h[-2,1] = 3$
$y[0,4] = x[1,2]h[-1,2] = 1$
$y[0,5] = x[1,3]h[-2,1] = 2$

$y[1,2] = x[3,1]h[-2,1] = 2$
$y[1,3] = x[2,1]h[-1,2] = 1$
$y[1,4] = x[2,2]h[-1,2] = 1$

$y[2,3] = x[1,1]h[1,2] + x[3,1]h[-1,2] = 2$
$y[2,4] = x[1,2]h[1,2] = 1$

$y[2,5] = x[1,3]h[1,2] = 1$

$y[3,2] = x[1,1]h[2,1] = 2$
$y[3,3] = x[1,2]h[2,1] + x[2,1]h[1,2] = 3$
$y[3,4] = x[1,3]h[2,1] + x[2,2]h[1,2] = 3$

$y[4,2] = x[2,1]h[2,1] = 2$
$y[4,3] = x[2,2]h[2,1] + x[3,1]h[1,2] = 3$

$y[5,2] = x[3,1]h[2,1] = 2$

$$y[n,n] = \{2, 2, 2, 2, 3, 1, 2, 2, 1, 1, 2, 1, 1, 2, 3, 3, 2, 3, 2\}$$

CPE462 Image Processing Project Proposal

Image Sharpening to Treat Blurred Visuals

A common issue with many self-proclaimed photographers and average iPhone users, is their tendency to capture blurry images. While their hearts are in the right place, taking a still photo can require a lot of patience, and patience among individuals is often hard to find in today's world.

In an effort to remedy this problem, our group: Nikola Ciric [CPE] and Alexander Gaskins [CPE], plan to test various different methods that can sharpen blurry images to increase their visibility. Two possible image-sharpening methods include edge detection and using a high-pass filter.

Using OpenCV in C++, the canny edge detector and Sobel() are both possible edge detection methods that can be applied. This allows a blurred image to be sharpened through the detection and segmentation of discontinuities that can interrupt the content being displayed.

On the other hand, a high-pass filter can be tested on an image to prevent certain data from passing through, with the goal of being able to identify and implement a system that can filter discontinuities that lead to blurriness. This can be done in MATLAB and Simulink, where a high pass filter can be modeled and tested on different blurred images.

Team Member roles

Nikola plans on tackling the edge detection in OpenCV, but may also test edge detection in other software, such as MATLAB and Python. As we explore more into the concept of image sharpening, we may even find another method that can perform better in OpenCV than edge detection.

Alexander plans on working with filters in Simulink, experimenting with different designs of high pass filters that can target the disruptions in different blurred images.