### 409

### Name: Md. Shifat E Arman Bhuiyan

### Roll: 008

# Problem 1

# Code

### problem\_1.m

#### clear; clc;

#### %% Input

#### L = 200;

#### K = 50;

#### x = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

#### %x = [1 2 1 -1];

#### %% b\_1

#### h = [0.1];

#### yb\_1 = myconv(x,h)

#### subplot(121)

#### stem(yb\_1)

#### title('Implemented function')

#### %% b\_2

#### yb\_2 = conv(x,h)

#### subplot(122)

#### stem(yb\_2)

#### title('Built-in function')

#### %% c

#### for n = 0:14

#### h(n+1) = 0.25\*(.75)^n

#### end

#### yc = myconv(x,h)

#### stem(yc)

### myconv.m

#### function [YY, NN] = myconv(x,h)

#### for n=1:length(x)+length(h)-1

#### y1(n)=0;

#### for k=1:length(x)

#### if (n-k+1>0) & (n-k+1<=length(h))

#### y1(n)=y1(n)+x(k)\*h(n-k+1);

#### else

#### end

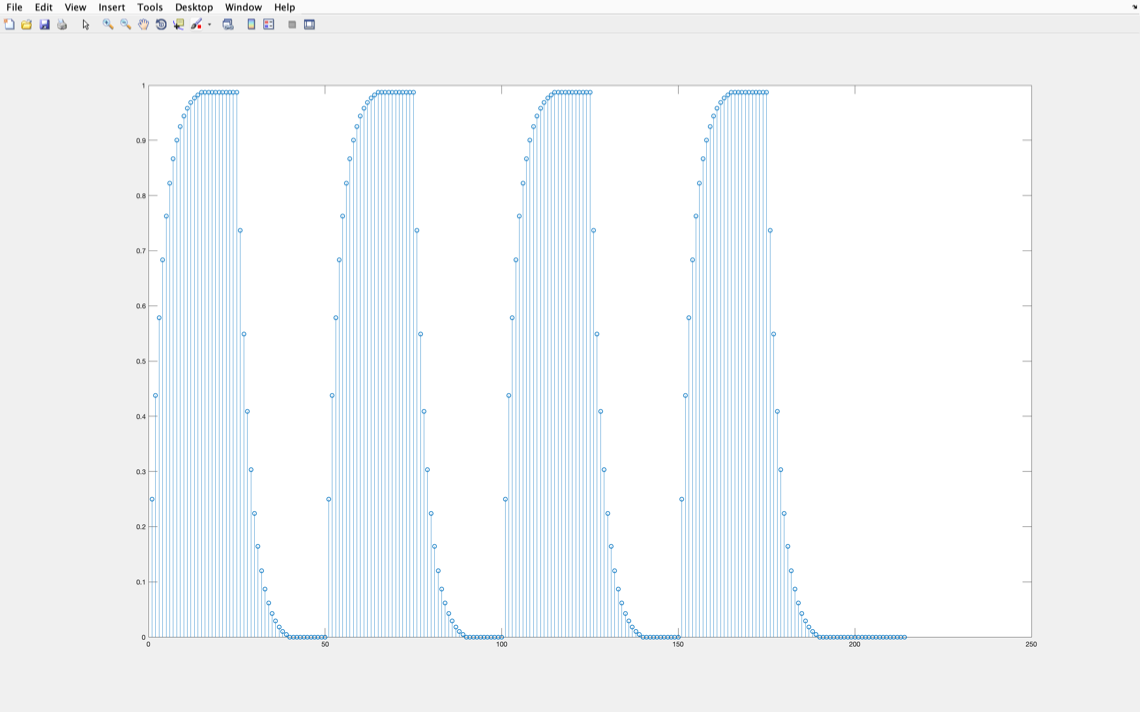
#### end

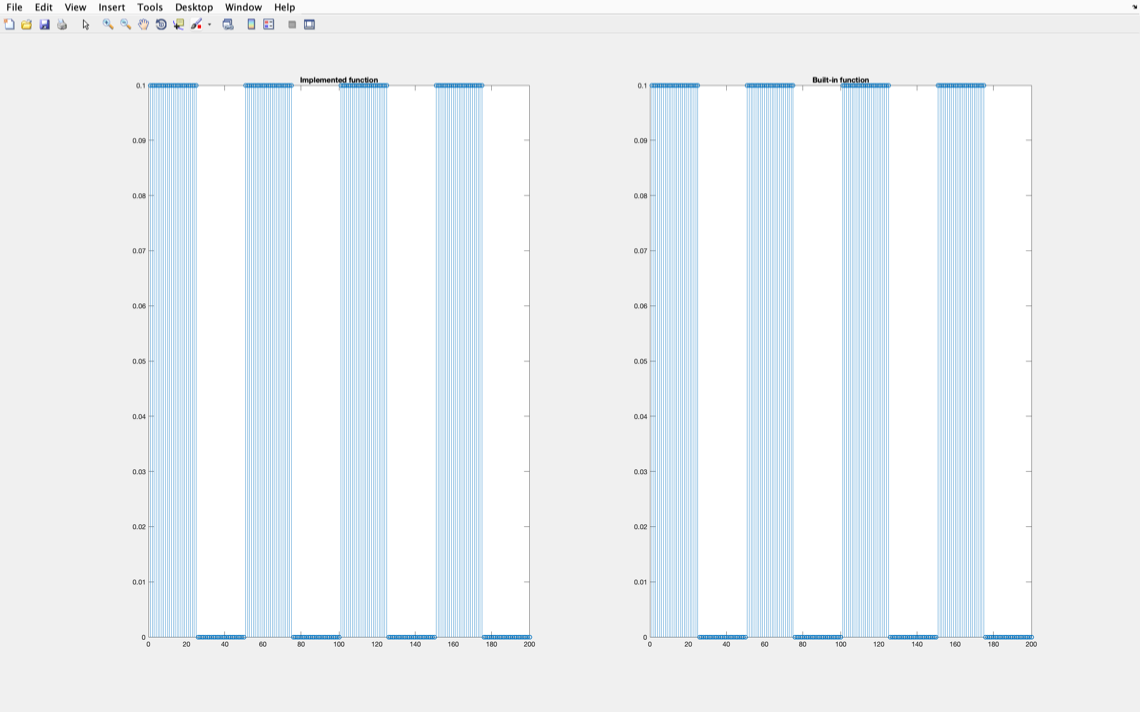
#### end

#### YY = y1;

#### end

# Output





### ————————

# Problem 2

# Code

#### clear;

#### clc;

#### %% Steps of Filtering in frequency domain

#### % 1. Input image.

#### % 2. Pre-process image.

#### % 1. Gray image

#### % 2. Double image

#### % 3. Alternate positive-negative in image pixels

#### % 3. Fourier transform image.

#### % 4. Create filter.

#### % 5. Filter image.

#### % 6. Inverse fourier transform image.

#### % 7. Post-process image.

#### % 8. Output image.

#### %% 1. Input image.

#### A = imread('problem\_two.tif');

#### %% 2. Pre-process image.

#### % Alternate positive-negative in image pixels

#### Ag = A;

#### Agd = double(Ag);

#### [row,col] = size(Agd);

#### for i=1:row

#### for j=1:col

#### Agda(i,j) = Agd(i,j) \* (-1)^(i+j);

#### end

#### end

#### %% 3. Fourier transform image. DFT

#### % time-domain to frequency-domain. Introduces a complex part.

#### Agda\_dft = fft2(Agda);

#### %% 4. Create filter.

#### filter\_high = zeros(row,col); % All zero

#### cut\_freq = 10; % Cut-off frequency

#### for i=1:row

#### for j=1:col

#### distance = ((row/2-i)^2+(col/2-j)^2)^0.5;

#### if(distance>cut\_freq)

#### filter\_high(i,j) = 1;

#### end

#### end

#### end

#### imshow(filter\_high)

#### title('High Pass Filter')

#### %% 5. Filter image.

#### filtered\_high = Agda\_dft.\*filter\_high;

#### %% 6. Inverse fourier transform image.

#### filtered\_high\_idft = ifft2(filtered\_high);

#### %% 7. Post-process image.

#### final\_high = uint8(filtered\_high\_idft);

#### %% 8. Output sharpened high pass filtered image.

#### subplot(1,4,1)

#### imshow(A)

#### title('Original image')

#### subplot(1,4,2)

#### imshow(Agda\_dft)

#### title('Spectrum image in fourier domain')

#### subplot(1,4,3)

#### imshow(filtered\_high)

#### title('Filter image in fourier domain')

#### subplot(1,4,4)

#### imshow(final\_high)

#### title('Filtered image in spatial domain')

### Result

