Name: Brihat Ratna Bajracharya

Roll No.: 19/075

Solution to Exercise 3 (Image Processing and Pattern Recognition)

Q1. By hand, compute the DFT of each of the following sequences:

a. [2, 3, 4, 5] **b. [2, -3, 4, -5]** **c. [-9, -8, -7, -6]**

Compare your answers with those given by Matlab's `fft` function.

Solution

Matlab Script

```
a = [2, 3, 4, 5]
b = [2, -3, 4, -5]
c = [-9, -8, -7, -6]

ffta = fft(a)
fftb = fft(b)
fftc = fft(c)
```

Matlab Output

```
a =
     2     3     4     5

b =
     2    -3     4    -5

c =
    -9    -8    -7    -6

ffta =
  14.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i

fftb =
  -2.0000 + 0.0000i  -2.0000 - 2.0000i  14.0000 + 0.0000i  -2.0000 + 2.0000i

fftc =
 -30.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i
```

Fourier Transform for above sequence is also calculated manually which is attached in Annex Section. From manual calculation and using `fft` function of Matlab, we found the same answers.

Q2. For each of the transforms you computed in the previous question, compute the inverse transform by hand. Verify by The Matlab function `ifft`

Solution

Matlab Script

```
ffta = [14.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i]
fftb = [-2.0000 + 0.0000i  -2.0000 - 2.0000i  14.0000 + 0.0000i  -2.0000 + 2.0000i]
fftc = [-30.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000- 2.0000i]

iffta = ifft(ffta)
ifftb = ifft(fftb)
ifftc = ifft(fftc)
```

Matlab Output

```
ffta =
  14.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i

fftb =
  -2.0000 + 0.0000i  -2.0000 - 2.0000i  14.0000 + 0.0000i  -2.0000 + 2.0000i

fftc =
 -30.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i

iffta =
     2     3     4     5

ifftb =
     2    -3     4    -5

ifftc =
    -9    -8    -7    -6
```

Analysis

Similarly as in Q1, the inverse fourier transform is also computed manually which is given in Annex Section. We successfully verified the answers using Matlab's `ifft` method.

Q3. Consider the following matrix:

$$\begin{bmatrix} 4 & 5 & -9 & -5 \\ 3 & -7 & 1 & 2 \\ 6 & -1 & -6 & 1 \\ 3 & -1 & 7 & -5 \end{bmatrix}$$

Using Matlab, calculate the DFT of each row.

Solution

Matlab Script

```
mat_44 = [4, 5, -9, -5;
          3, -7, 1, 2;
          6, -1, -6, 1;
          3, -1, 7, -5;]
[row column] = size(mat_44);
fft_array = zeros(4, 4);

for iter = [1:row]
    disp(['DFT of row ', num2str(iter), ': ']);
    fft_array(iter, :) = fft(mat_44(iter,:));
    disp(fft_array(iter, :))
end
```

Matlab Output

```
mat_44 =
     4     5    -9    -5
     3    -7     1     2
     6    -1    -6     1
     3    -1     7    -5

DFT of row 1:
  -5.0000 + 0.0000i  13.0000 -10.0000i  -5.0000 + 0.0000i  13.0000 +10.0000i

DFT of row 2:
  -1.0000 + 0.0000i   2.0000 + 9.0000i   9.0000 + 0.0000i   2.0000 - 9.0000i

DFT of row 3:
   0.0000 + 0.0000i  12.0000 + 2.0000i   0.0000 + 0.0000i  12.0000 - 2.0000i

DFT of row 4:
   4.0000 + 0.0000i  -4.0000 - 4.0000i  16.0000 + 0.0000i  -4.0000 + 4.0000i
```

Q4. Open up the image `engineer.tiff`:

```
>> en=imread('engineer.tiff');
```

Experiment with applying the Fourier transform to this image and the following filters:

(a) `ideal_filters` (both low and high pass),

(b) `Butterworth_filters`,

(c) `Gaussian_filters`.

What is the smallest radius of a low pass `ideal_filters` for which the face is still recognizable?

Matlab Script

1. fftshow.m

```
function fftshow(f,type)
    if nargin < 2,
        type = 'log';
    end

    if (type=='log')
        fl = log(1 + abs(f));
        fm = max(fl(:));
        imshow(im2uint8(fl / fm))
    elseif (type=='abs')
        fa = abs(f);
        fm = max(fa(:));
        imshow(fa / fm)
    else
        error('TYPE must be abs or log.');
end;
```

2. lowpass.m

```
function out = lowpass(im, d)
    height = size(im, 1);
    width = size(im, 2);
    [x, y] = meshgrid(-floor(width / 2) : floor((width - 1) / 2), -floor(height...
             / 2): floor((height - 1) / 2));
    z = sqrt(x .^ 2 + y .^ 2);
    out = (z < d);
end
```

3. highpass.m

```
function out = highpass(im, d)
    height = size(im, 1);
    width = size(im, 2);
   [x, y] = meshgrid(-floor(width / 2) : floor((width - 1) / 2), -floor(height...
             / 2): floor((height - 1) / 2));
    z = sqrt(x .^ 2 + y .^ 2);
    out = (z > d);
end
```

4. lbutter.m

```
function out=lbutter(im,d,n)
    height = size(im, 1);
    width = size(im, 2);
    [x, y] = meshgrid(-floor(width / 2) : floor((width - 1) / 2), -floor(height...
             / 2): floor((height - 1) / 2));
    out = 1 ./ (1 + (sqrt(2) - 1) * ((x .^ 2 + y .^ 2) / d ^ 2) .^ n);
end
```

5. hbutter.m

```
function out=hbutter(im,d,n)
    out = 1 - lbutter(im, d, n);
end
```

6. e3q4idealfilters.m

```
radius = 30;
cm = rgb2gray(imread('engineer.tiff'));

% IDEAL LOW PASS FILTERS
cf = fftshift(fft2(cm));
figure, fftshow(cf, 'log');
title('Image after DFT');

low = lowpass(cm, radius);
cfl = cf .* low;
figure, fftshow(cfl, 'log');
title(['Low Pass Ideal Filtering, radius=' num2str(radius)]);
```

```
cfli = ifft2(cfl);
figure, fftshow(cfli, 'abs');
title('Ideal Low Pass Resulting Image');

% IDEAL HIGH PASS FILTERS
cf = fftshift(fft2(cm));
figure, fftshow(cf, 'log');
title('Image after DFT');

high = highpass(cm, radius);
cfh = cf .* high;
figure, fftshow(cfh, 'log');
title(['High Pass Ideal Filtering, radius=' num2str(radius)]);

cfhi = ifft2(cfh);
figure, fftshow(cfhi, 'abs');
title('Ideal High Pass Resulting Image');
```

7. e3q4butterworth.m

```
cutoff = 15;
order = 1;

cm = rgb2gray(imread('engineer.tiff'));
cf = fftshift(fft2(cm));
figure, fftshow(cf, 'log')
title('Image after DFT');

% BUTTERWORTH LOW PASS FILTER
bl = lbutter(cm, cutoff, order);
cfbl = cf .* bl;
figure, fftshow(cfbl, 'log')
title(['Low Pass ButterWorth Filtering, cutoff=' num2str(cutoff) ', order='
num2str(order)]);

cfbli = ifft2(cfbl);
figure, fftshow(cfbli, 'abs')
title('ButterWorth Low Pass Resulting Image');

% BUTTERWORTH HIGH PASS FILTER
bh = hbutter(cm, cutoff, order);
```

```
cfbh = cf .* bh;
figure, fftshow(cfbh, 'log')
title(['High Pass ButterWorth Filtering, cutoff=' num2str(cutoff) ', order='
num2str(order)]);

cfbhi = ifft2(cfbh);
figure, fftshow(cfbhi, 'abs')
title('ButterWorth High Pass Resulting Image');
```

8. e3q4gaussianfilter.m

```
cm = rgb2gray(imread('engineer.tiff'));
cf = fftshift(fft2(cm));
figure, fftshow(cf, 'log')
title('Image after DFT');

% [row column] = size(cm);
cm_size = size(cm);
sigma = 10;

% GAUSSIAN LOW PASS FILTER USING fspecial
gl = mat2gray(fspecial('gaussian', cm_size, sigma));
cfgl = cf .* gl;
figure, fftshow(cfgl, 'log')
title(['Low Pass Gaussian Filtering, \sigma=' num2str(sigma)]);

cfgli = ifft2(cfgl);
figure, fftshow(cfgli, 'abs')
title('Gaussian Low Pass Resulting Image');

% GAUSSIAN HIGH PASS FILTER USING fspecial
gh = 1 - gl;
cfgh = cf .* gh;
figure, fftshow(cfgh, 'log')
title(['High Pass Gaussian Filtering, \sigma=' num2str(sigma)]);

cfghi = ifft2(cfgh);
figure, fftshow(cfghi, 'abs')
title('Gaussian High Pass Resulting Image');
```

Matlab Output

Figure 1: Original (engineer.tiff)

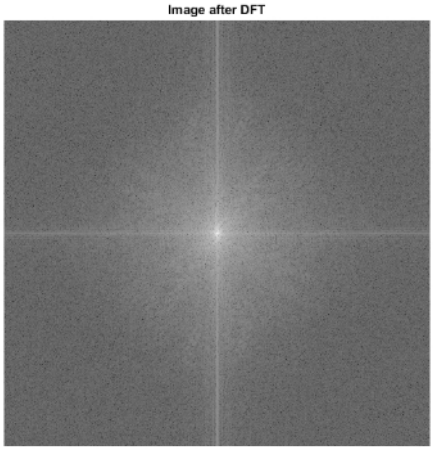


Figure 2: Image after DFT

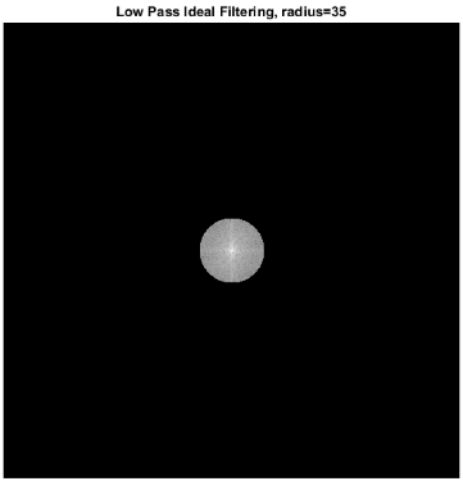


Figure 3: Low Pass Ideal Filter

Figure 4: Image through Low Pass Ideal Filter

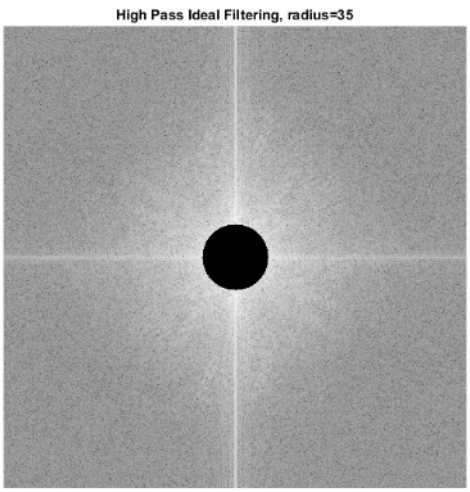


Figure 6: HIgh Pass Ideal Filter

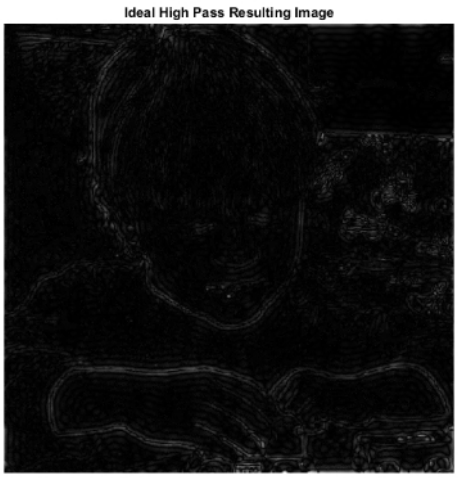


Figure 5: Image through High Pass Ideal Filter

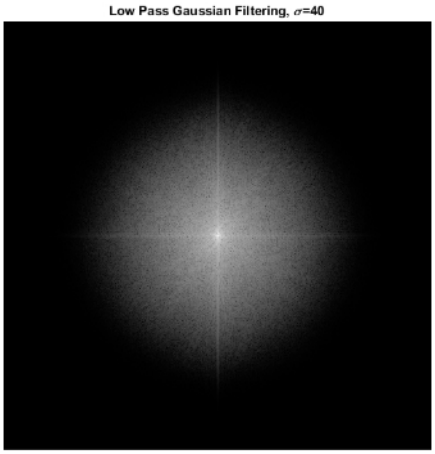


Figure 8: Low Pass Gaussian Filter

Figure 7: Image through Low Pass Gaussian Filter

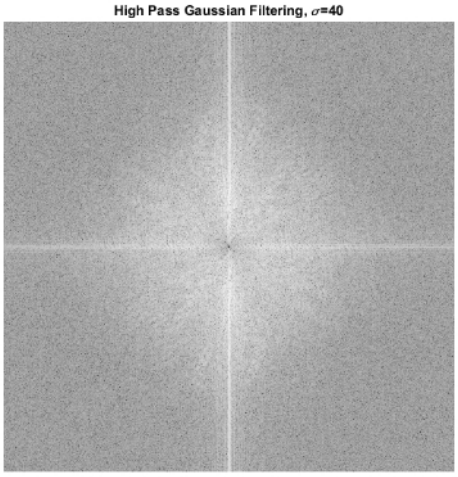


Figure 9: High Pass Gaussian Filter

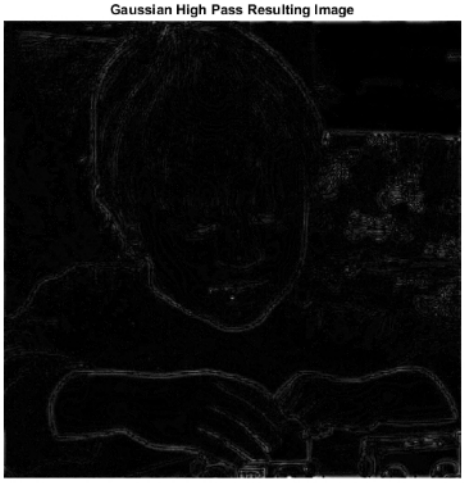


Figure 10: Image through High Pass Gaussian Filter

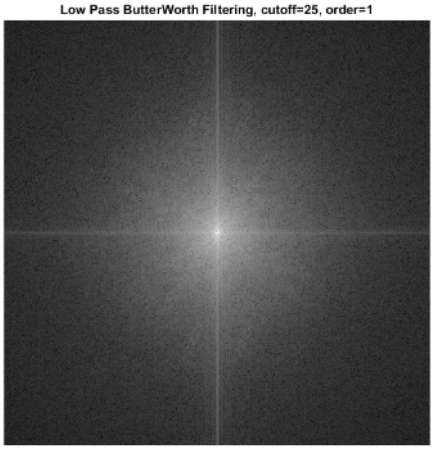


Figure 12: Low Pass ButterWorth Filter

Figure 11: Image through Low Pass Butterworth Filter

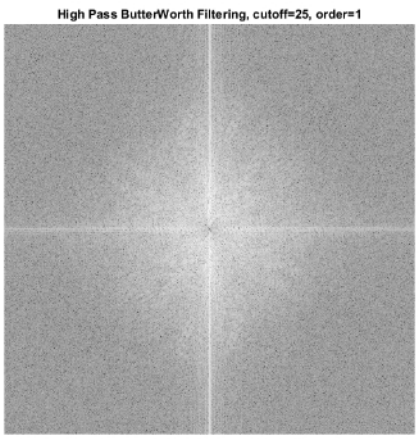


Figure 14: High Pass ButterWorth Filter

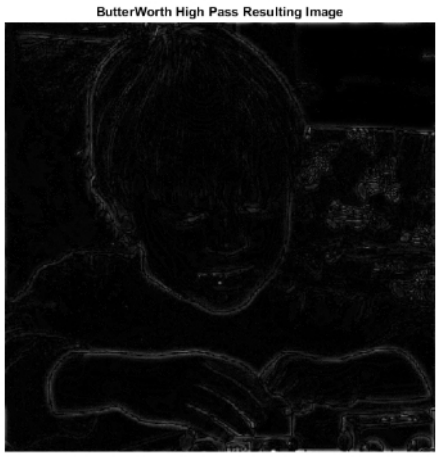


Figure 13: Image through High Pass ButterWorth Filter

Analysis

For this activity, the low and high pass filters for ideal filter and butterworth filter was implemented from scratch and low pass and high pass gaussian filter was implemented using `fspecial` method in Matlab. Also, using trial and error, the minimum radius for which face is able to be recognized was found out to be 30. In above group of figures, Figure 1 is the original image and Figure 2 is the image after DFT operation using `fft2` and `fftshift` method of Matlab. Similarly, Figure 3-6 corresponds to output for Ideal Low and High Pass Filters. Figure 7-10 corresponds to output for Gaussian Filters and Figure 11-14 corresponds to output for ButterWorth Filter.

Q5. Using your digital camera, produce a digital image of the face of somebody you know, and perform the same calculations as in the previous question.

Matlab Script

```
b_original = imread('brihat.png');
brihat = rgb2gray(b_original);

figure

subplot(3, 3, 1);
imshow(b_original);
title('Original');
```

```
subplot(3, 3, 4);
imshow(brihat);
title('Original (Grayscale)');


% IDEAL FILTERS
% -------------
radius = 25;
cf = fftshift(fft2(brihat));

%  IDEAL LOW PASS
low = lowpass(brihat, radius);
cfl = cf .* low;

cfli = ifft2(cfl);
subplot(3, 3, 2);
fftshow(cfli, 'abs');
title('Ideal Low Pass');

%  IDEAL HIGH PASS
cf = fftshift(fft2(brihat));

high = highpass(brihat, radius);
cfh = cf .* high;

cfhi = ifft2(cfh);
subplot(3, 3, 3);
fftshow(cfhi, 'abs');
title('Ideal High Pass');


% BUTTERWORTH FILTERS
% -------------------
cutoff = 15;
order = 1;

% BUTTERWORTH LOW PASS
bl = lbutter(brihat, cutoff, order);
cfbl = cf .* bl;
```

```
cfbli = ifft2(cfbl);
subplot(3, 3, 5);
fftshow(cfbli, 'abs');
title('ButterWorth Low Pass');

% BUTTERWORTH HIGH PASS
bh = hbutter(brihat, cutoff, order);
cfbh = cf .* bh;

cfbhi = ifft2(cfbh);
subplot(3, 3, 6);
fftshow(cfbhi, 'abs')
title('ButterWorth High Pass');


% GAUSSIAN FILTERS
% ----------------
cm_size = size(brihat);
sigma = 20;

% GAUSSIAN LOW PASS
gl = mat2gray(fspecial('gaussian', cm_size, sigma));
cfgl = cf .* gl;

cfgli = ifft2(cfgl);
subplot(3, 3, 8);
fftshow(cfgli, 'abs')
title('Gaussian Low Pass');

% GAUSSIAN HIGH PASS
gh = 1 - gl;
cfgh = cf .* gh;

cfghi = ifft2(cfgh);
subplot(3, 3, 9);
fftshow(cfghi, 'abs')
title('Gaussian High Pass');
```

Matlab Output

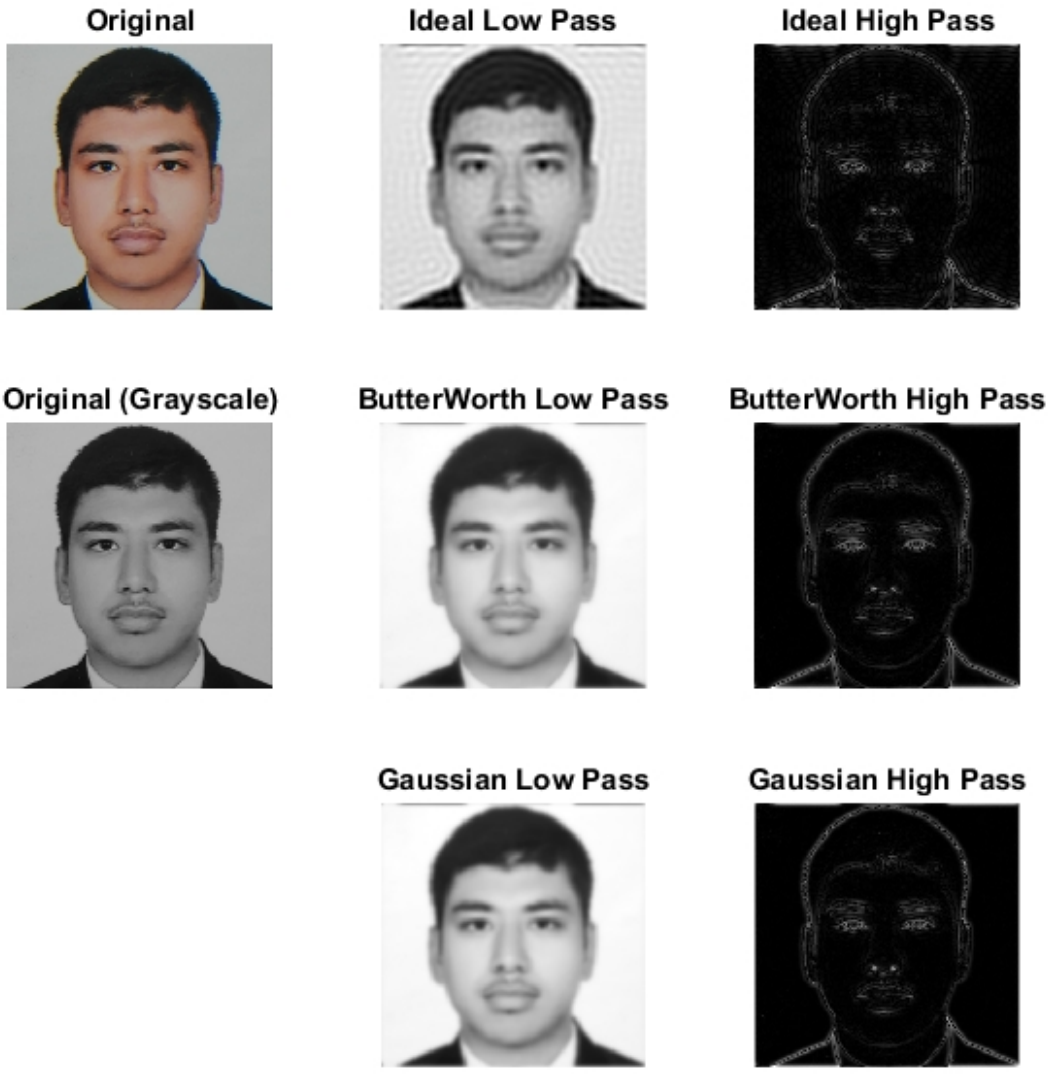


Figure 15: My Image through various filters

Parameters Used

Following values were used in this activity

Image Size: `256 x 256`

Ideal Filter: *Radius* = `25`

ButterWorth Filter: *Cutoff* = `15`, *Order* = `1`

Gaussian Filter: *Sigma* = `20`

ANNEX

Manual Calculations of DFT and IDFT for Q1 and Q2

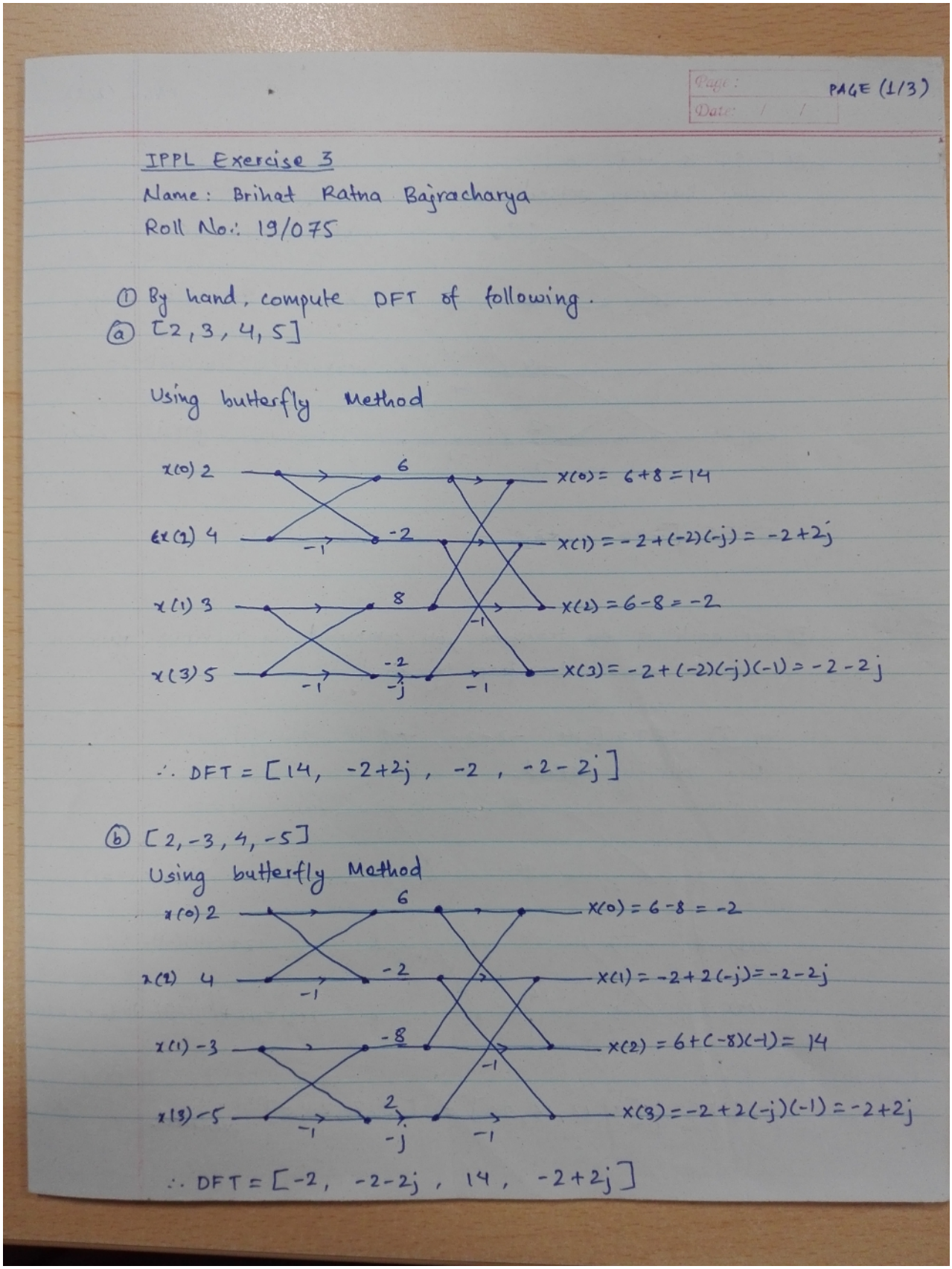
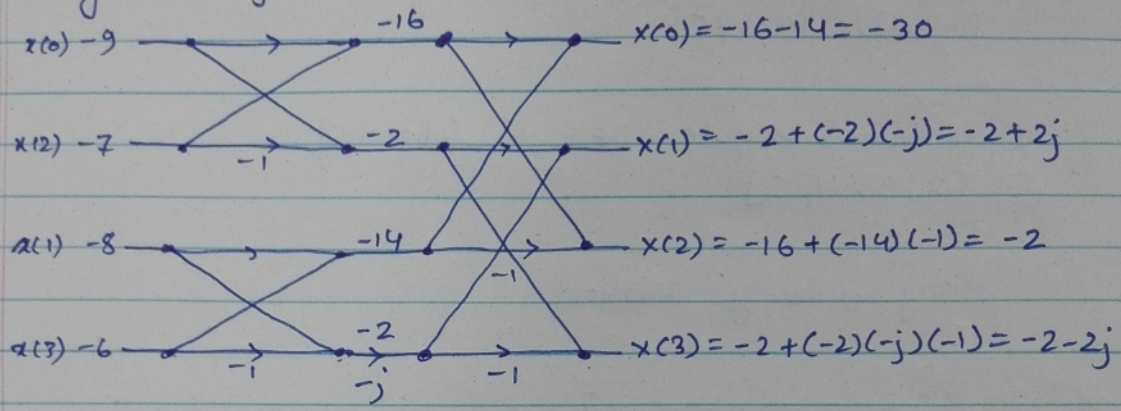


Illustration 1: Manual Calculation for Q1 and Q2 (1/3)

② [-9, -8, -7, -6]

Using butterfly Method

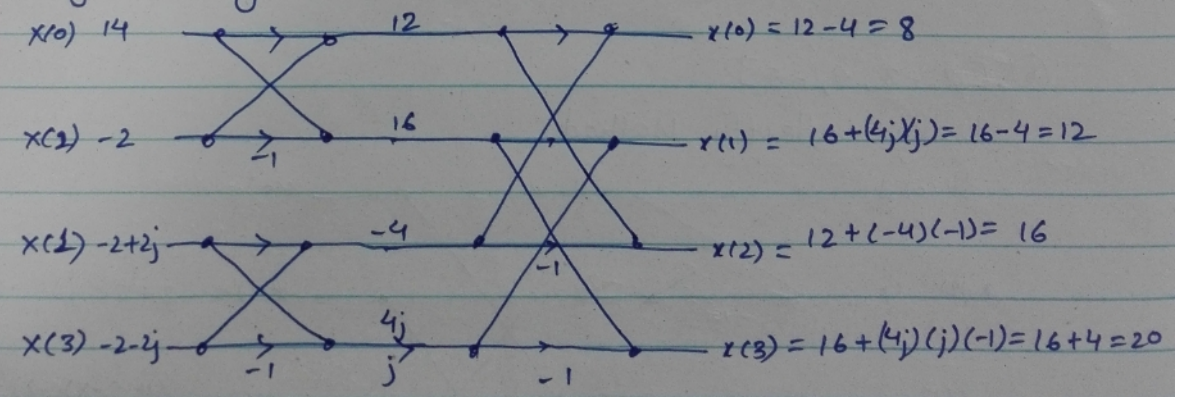


∴ DFT = [-30, -2+2j, -2, -2-2j]

③ For each of the transform you computed in previous question compute inverse transform by hand.

For ④, DFT = [14, -2+2j, -2, -2-2j]

Using butterfly method,

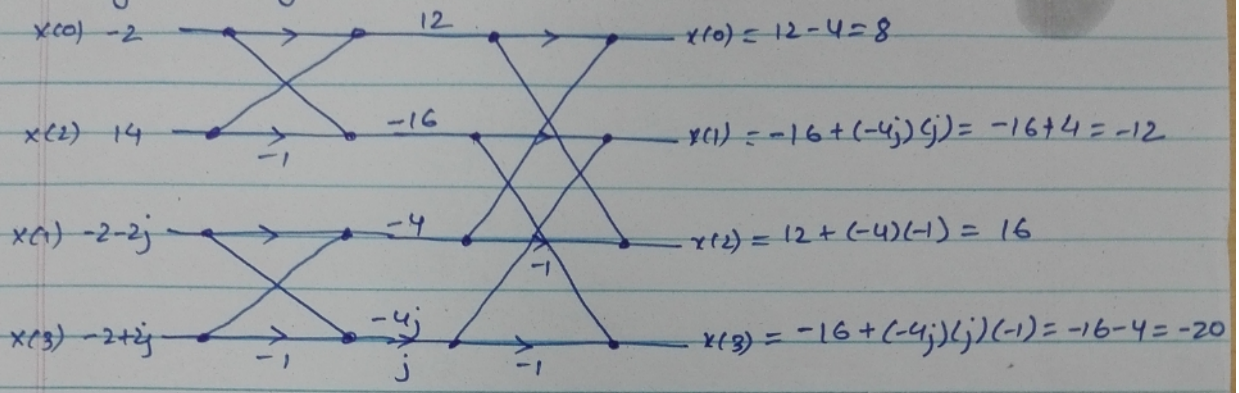


∴ IDFT = $\frac{1}{4}$[8, 12, 16, 20] = [2, 3, 4, 5] verified.

Illustration 2: Manual Calculation for Q1 and Q2 (2/3)

For (b) DFT = [-2, -2-2j, 14, -2+2j]

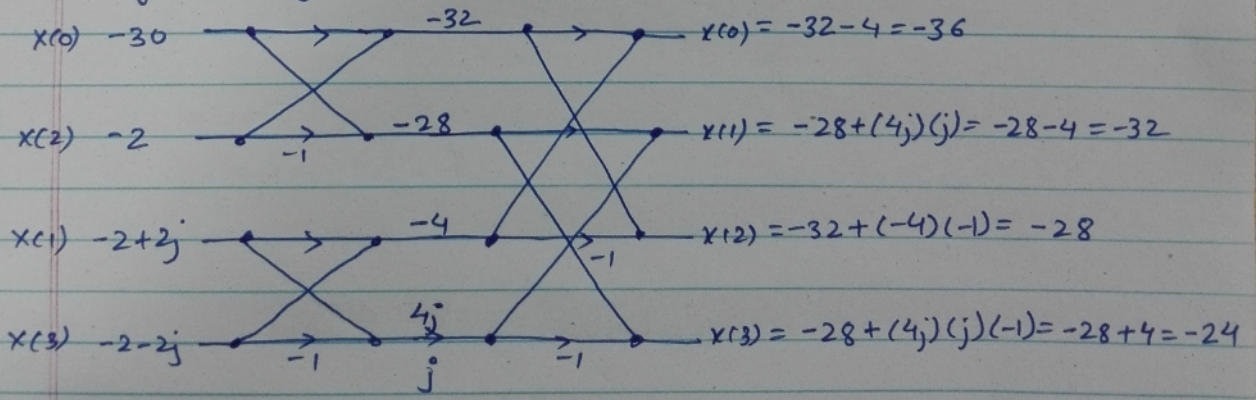
Using Butterfly method,



$\therefore IDFT = \frac{1}{4}[8, -12, 16, -20] = [2, -3, 4, -5]$ Verified

For (c), DFT = [-30, -2+2j, -2, -2-2j]

Using Butterfly Method,



$\therefore IDFT = \frac{1}{4}[-36, -32, -28, -24] = [-9, -8, -7, -6]$

Verified.

Illustration 3: Manual Calculation for Q1 and Q2 (3/3)