Unit-3: Image Enhancement in Frequency Domain

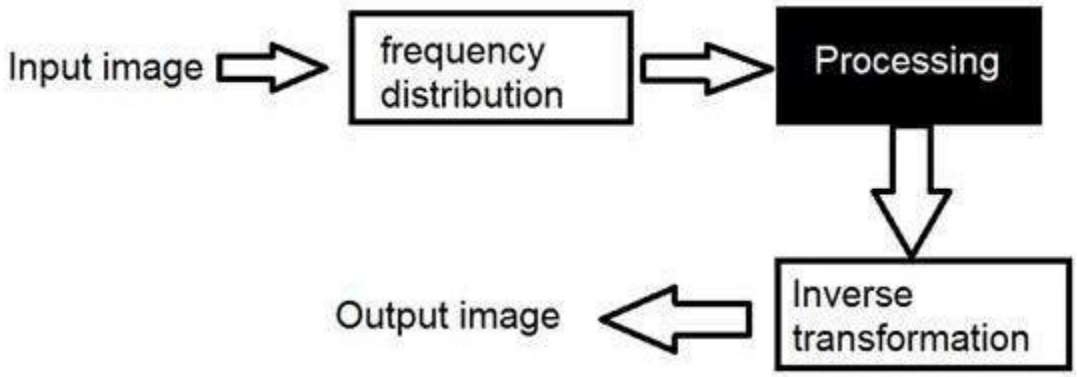
Frequency Domain Filters are used for smoothing and sharpening of image by removal of high or low frequency components. Sometimes it is possible of removal of very high and very low frequency. Frequency domain filters are different from spatial domain filters as it basically focuses on the frequency of the images. It is basically done for two basic operations i.e., Smoothing and Sharpening.

In spatial domain, we deal with images as it is. The values of the pixels of the image change with respect to scene. Whereas in frequency domain, we deal with the rate at which the pixel values are changing in spatial domain.

In spatial domain process is done with following steps

In simple spatial domain, we directly deal with the image matrix.

In frequency domain process is done with following steps



In frequency domain first transform the image to its frequency distribution. Then our black box system performs what-ever processing it has to performed, and the output of the black box in this case is not an image, but a transformation. After

performing inverse transformation, it is converted into an image which is then viewed in spatial domain.

Transformation

A signal can be converted from time domain into frequency domain using mathematical operators called transforms. There are many kind of transformation that does this. Some of them are given below.

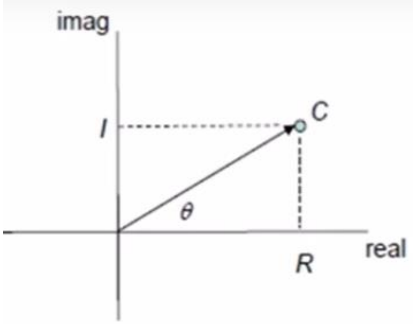
- Fourier Series
- Fourier transformation

Basic Terminologies:

Complex number C is defined as:

C = r + j I

Where, r and I are the real number and j is the imaginary number equals to square of -1. i.e. j = $\sqrt{-1}$.



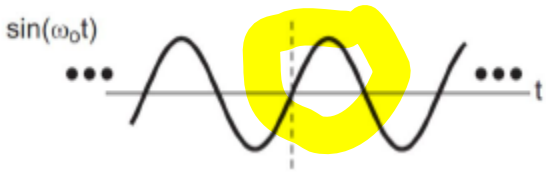
In polar coordinates:

$C = |C| (\cos\theta + j \sin\theta)$

$$e^{j\theta} = \cos\theta + j\sin\theta, \text{ so } C = |C|e^{j\theta}$$

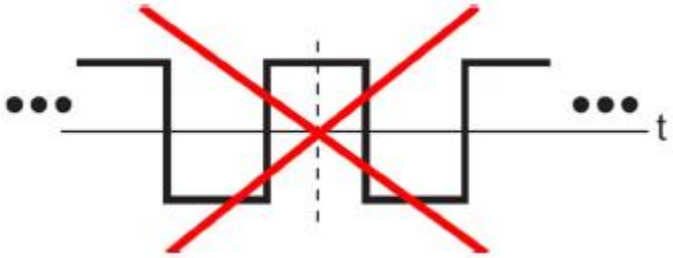
Some properties of sine and cosine frequencies

Sine is odd or anti-symmetric function



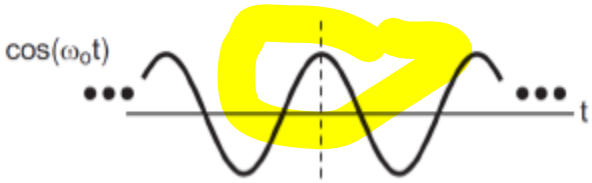
tala bata mathi airaxa

It can't create even function

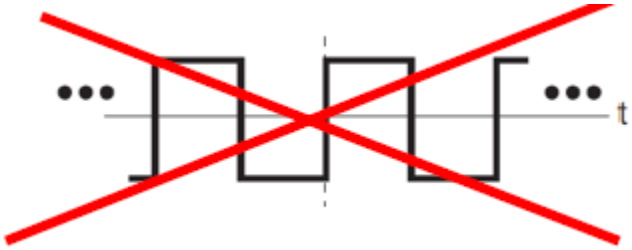


esma ni mathi bata tala airax even vayo esto chai sine le create garna skdenw re

Cosine is the even or symmetric function

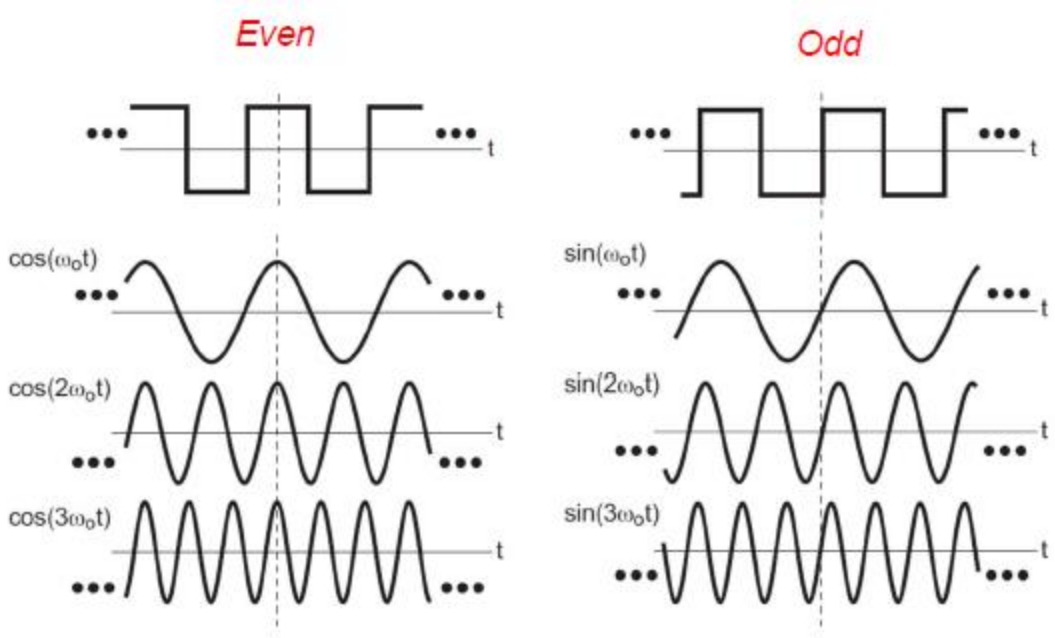


It can't create odd function



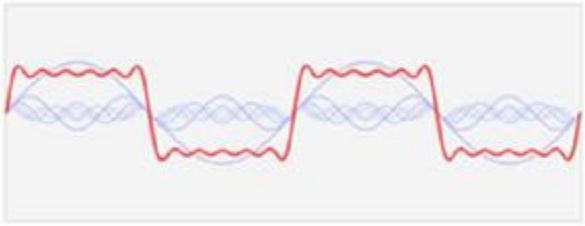
By combining both, we can handle odd and even function of sine and cosine waves. This is also called sinocos function.

frequency domain vnekae sine cosine waves ho



In 1807 Jean Baptiste Joseph Fourier proposed a Fourier function which is the combination of both sine and cosine functions.

Fourier series: It represents any periodic function as weighted combination of sine and cosine function of different frequencies.



A function ***f(t)*** of a continuous variable ***t***, that is periodic with period ***T***, can be express as the sum of sines and cosines multiplied by appropriate coefficients.

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{T}t}$$

fourrieer series vnekae periodic fxn ho so T involve hunxa ani sinecosine wave vnesi tyo e^j2pi tw hunxa nae

euta kura chai esam suruma power ma - xanew

Where,

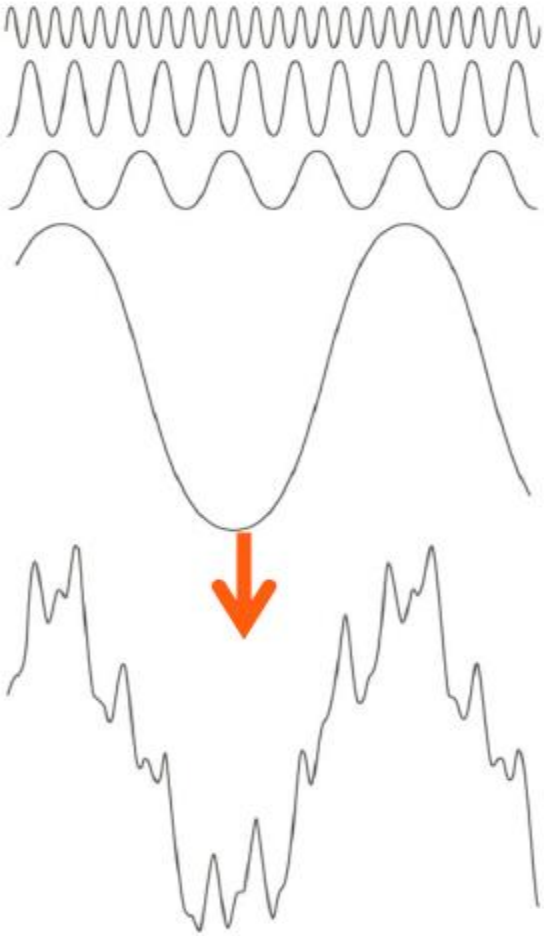
fun fact: f(t) aafae cn calculate garna use vairaxa
mathi feri cn f(t) calculate garda use viaraxa

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\, e^{-j\frac{2\pi n}{T}t}\, dt \qquad \text{for } n = 0, \pm 1, \pm 2, \ldots$$

Form calculus

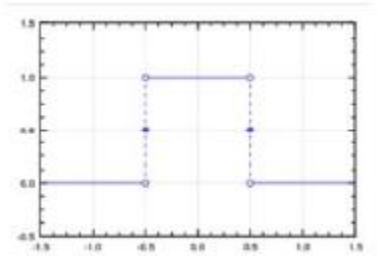
$$e^{jx} = \cos x + j \sin x$$

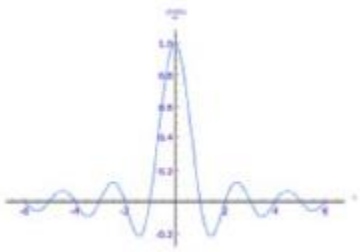
Following graph shows the combination of periodic function of sine and cosine. This graph at the bottom is sum of above four graphs.



aba yo 4 wota ma
kunai wave sine xa kunai
cosine xa ani 4 wotae combine
garesi fourrieer series banxa

Fourier Transform: Fourier transform is the input tool that is used to decompose an image into its sine and cosine components. It deals with non-periodic function with finite area. This is the integral of weighted sin and cosine functions.





only definition

Functions (signals) can be completely reconstructed from the Fourier domain without losing any information. For this Fourier proposed two methods

1. Fourier Transform
2. Inverse Fourier Transform

Fourier Transform:

1-D Fourier transform of continuous function is defined by the expression

$$F(s) = \int_{-\infty}^{\infty} f(t) e^{-j2\pi st} dt$$

continuous ma / wala pardewn
discrete ma prxa
esma - inifinity to infinity uta 0 to N-1

1-D Inverse Fourier transform continuous function is defined by the expression

$$f(t) = \int_{-\infty}^{\infty} F(s) e^{\ j2\pi st} ds$$

discrete wala ko numerical nae xa vnesi - infinity to infinity yve solve garnae mildenwni

1-D Discrete Fourier Transform function is defined by the expression

$$F(k) = \sum_{x=0}^{N-1} f(x) \cdot e^{-\frac{i2\pi}{N} kx}$$

Where, N is number of values, k = 0,1,2…..N-1

Example: Find the DFT of *f(x) = {0,1,2,1}*

Solution: here, N = 4, k = 0,1,2,3

$$F(k) = \sum_{x=0}^{N-1} f(x) \cdot e^{-\frac{i2\pi}{N} kx}$$

$F(k) = f(x)\ e^{-(i2pkx)/4} + f(x)\ e^{-(i2pkx)/4} + f(x)\ e^{-(i2pkx)/4} + f(x)\ e^{-(i2pkx)/4}$

Where, x = 0,1,2,3 p= pie e^{-ax} = (cos x – a sin x)

For k = 0

$F(0) = f(0)\ e^{-0} + f(1)\ e^{-0} + f(2)\ e^{-0} + f(3)\ e^{-0}$

$F(0)= 0*1 + 1*1 + 2*1 + 1*1$

$F(0) = 4$

For k = 1

$F(1) = f(0)\ e^{-0} + f(1)\ e^{-(i2p1*1)/4} + f(2)\ e^{-(i2p1*2)/4} + f(3)\ e^{-(i2p1*3)/4}$

$F(1) = f(0)\ e^{-0} + f(1)\ e^{-ip/2} + f(2)\ e^{-ip} + f(3)\ e^{-i3p/2}$

$F(1) = 0 + 1[\cos(p/2)-i \sin(p/2)] + 2[\cos(p)-i \sin(p)] + 1[\cos(3p/2)-i \sin(3p/2)]$

$F(1) = 0+1[0-i] + 2[-1-0] + 1[0-i(-1)]$

$F(1) = 0-i-2+i$

F(1) = -2

For k = 2

$F(2) = f(0)\, e^{-0} + f(1)\, e^{-(i2p2*1)/4} + f(2)\, e^{-(i2p2*2)/4} + f(3)\, e^{-(i2p2*3)/4}$

$F(2) = f(0)\, e^{-0} + f(1)\, e^{-ip} + f(2)\, e^{-i2p} + f(3)\, e^{-i3p}$

F(2) = 0 + 1[cos(p)-i sin(p)] + 2 [cos(2p)-i sin(2p)] + 1 [cos(3p)-i sin(3p)]

F(2) = 0+1[-1-0] + 2[1-0] + 1[-1-0]

F(2) = 0-1+2-1

F(2) = 0

For k = 3

$F(3) = f(0)\, e^{-0} + f(1)\, e^{-(i2p3*1)/4} + f(2)\, e^{-(i2p3*2)/4} + f(3)\, e^{-(i2p3*3)/4}$

$F(3) = f(0)\, e^{-0} + f(1)\, e^{-i3p/2} + f(2)\, e^{-i3p} + f(3)\, e^{-i9p/2}$

F(3) = 0 + 1 [cos(3p/2)-i sin(3p/2)] + 2 [cos(3p)-i sin(3p)] + 1 [cos(9p/2)-i sin(9p/2)]

F(3) = 0+1[0-(-i)] + 2[-1-0] + 1[0-i]

F(3) = 0+ i -2 -i

F(3) = -2

Hence DFT of f(x) = {0,1,2,1} is

F(k) = {4,-2,0,-2}

1-D Discrete Inverse Fourier Transform function is defined by the expression

$$f(x) = \sum_{x=0}^{N-1} F(\mathrm{k}) e^{2\pi i x \mathrm{k}/N}.$$

2-D Fourier Transform of continuous function is defined by the expression

$$F(u,v) = \iint_{-\infty}^{\infty} f(x,y) e^{-i2\pi(ux+vy)} dxdy$$

Where, u and v are frequencies along x and y axis respectively.

2-D Inverse Fourier Transform of continuous function is defined by the expression

$$f(x,y) = \iint_{-\infty}^{\infty} F(u,v) e^{i2\pi(xu+yv)} dudv$$

2-D Discrete Fourier Transform (DFT) function is defined by the expression

$$F[p,q] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n] e^{-i2\pi pm/M} e^{-i2\pi qn/N}$$

p = 0….. (M-1), q = 0….. (N-1)

Where, p and q are the frequencies along with m and n respectively

2-D Discrete Inverse Fourier Transform (DFT) function is defined by the expression

$$f[m,n] = \frac{1}{MN} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} F[p,q] e^{i2\pi pm/M} e^{i2\pi qn/N}$$

m = 0…. (M-1), n = 0….. (N-1)

The Mask or Kernel of 2-D DFT:

The kernel or mask of 2-point DFT is:

1	1
1	-1

The kernel or mask of 4-point DFT is:

1	1	1	1
1	-j	-1	J
1	-1	1	-1
1	J	-1	-j

Formulas of Calculation of DFT:

For 1-D DFT

$F(k) = mask * f(x)$

For 2-D DFT

$F(k,l) = mask * f(x,y) * mask^T$

Where, $mask^T$ is transpose of mask

Here, $mask = mask^T$, So we can also write,

$F(k,l) = mask * f(x,y) * mask$

Note: *square of imaginary number is equals to -1. i.e. $j^2 = -1$*

Example1: Calculate 4-point DFT for 1-D sequence f(x) = {0,1,2,3} using matrix method.

F(k) = mask*f(x)

1	1	1	1
1	-j	-1	j
1	-1	1	-1
1	j	-1	-j

*

0
1
2
3

=

0+1+2+3
0-j-2+3j
0-1+2-3
0+j-2-3j

=

6
-2+2j
-2
-2-2j

Hence, F(k) for 1-D DFT = {6, -2+2j, -2, -2-2j} esko inverse nikalne ni practice grne hae

Example2: Compute 2-D DFT of the grayscale image given bellow:

F(m,n) =

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Solution:

Mask for 4-point DFT is

1	1	1	1
1	-j	-1	J
1	-1	1	-1
1	j	-1	-j

Transpose to mask is

1	1	1	1
1	-j	-1	J
1	-1	1	-1
1	j	-1	-j

New, F(k) for 2D DFT is

$F(k,l) = mask * f(x,y) * mask^T$

1	1	1	1
1	-j	-1	j
1	-1	1	-1
1	j	-1	-j

*

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

*

1	1	1	1
1	-j	-1	j
1	-1	1	-1
1	j	-1	-j

4	4	4	4
0	0	0	0
0	0	0	0
0	0	0	0

*

1	1	1	1
1	-j	-1	j
1	-1	1	-1
1	j	-1	-j

=

16	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Hence, 2-D DFT of given gryscale image is:

16	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

For 2-D 4-point Inverse Discrete Fourier Transform (IDFT)

$F(k,l) = (1/m*n)\ (mask * f(x,y) * mask^T)$

Where, m = number of rows, n = number of column

Here, mask = $mask^T$, So we can also write,

*F(k,l) = (1/m*n)(mask*f(x,y)*mask)*

Mask =

1	1	1	1
1	J	-1	-j
1	-1	1	-1
1	-j	-1	J

Example: Find 2D – IDFT of image

32	-8	0	-8
-8	0	0	0
0	0	0	0
-8	0	0	0

Now, m = 4, n = 4

F(k,l) = (1/16) (mask*f(x,y)*mask)

= (1/16)

1	1	1	1
1	j	-1	-j
1	-1	1	-1
1	-j	-1	j

*

32	-8	0	-8
-8	0	0	0
0	0	0	0
-8	0	0	0

*

1	1	1	1
1	j	-1	-j
1	-1	1	-1
1	-j	-1	j

= (1/16)

16	-8	0	-8
32	-8	0	-8
48	-8	0	-8
32	-8	0	-8

*

1	1	1	1
1	j	-1	-j
1	-1	1	-1
1	-j	-1	j

= (1/16)

0	16	32	16
32	32	48	32
32	48	64	48
16	32	48	32

=

0	1	2	1
1	2	3	2
2	3	4	3
1	2	3	2

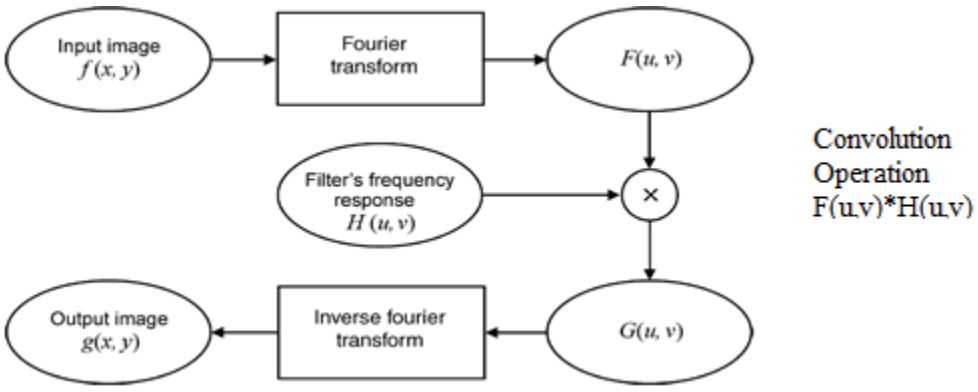
Hence, 2-D IDFT of given image is

0	1	2	1
1	2	3	2
2	3	4	3
1	2	3	2

Frequency Domain Filter:

Frequency Domain Filters are used for smoothing and sharpening of image by removal of high or low frequency components. Sometimes it is possible of removal of very high and very low frequency. Frequency domain filters are different from spatial domain filters as it basically focuses on the frequency of the images. It is basically done for two basic operations i.e., Smoothing and Sharpening.

The convolution is used to evaluate filters in the spatial domain, filtering is implemented in the frequency domain by multiplication. The image in the frequency domain is multiplied by the filter's frequency response in the frequency domain. Thus, the filter transfer function is simply a matrix of the same size as the image. The complex values of the image in the frequency domain are simply multiplied element by element with the filter transfer function to amplify or attenuate specific frequencies of the image. The inverse Fourier transform is then used to generate the output image.



Filter Parameters:

Distance Function

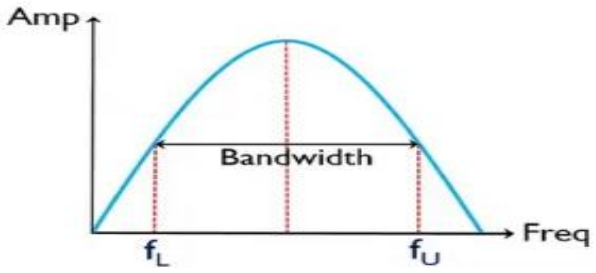
To generate filter transfer functions, it is necessary to know the distance of each element of the transfer function to the origin (0,0). In the filter transfer function equations, this distance to the origin is denoted as $D(u,v)$, or simply as D. Since the frequency domain data ranges from 0 to 2π in both the horizontal and vertical direction, rather than between ($-\pi$ and π), it is convenient to use a special function to calculate the distances.

Cutoff Frequency

The transition point between the pass and stop bands of the filter is known as cutoff frequency and denoted as D_0.

Band Width

For band-reject and band-pass filters, where the data between lower and upper cutoff frequencies is either rejected or passed, is called *Band Width*. The width is denoted as W.



Smoothing Frequency Domain Filter (Lowpass filter):

This filter technique is used to remove high frequency components of the image and keep low frequency components. Hence it smooth or blur the original image. It is also used to remove the noise from the image.

Following techniques are used for this filter process:

- Ideal Low Pass Filter
- Butterworth Low Pass Filter
- Gaussian Low Pass Filter

Ideal Low Pass Filter:

ILPF is the simplest low pass filter technique that "cuts off" all high frequency components of the DFT that are at a distance greater than a specified distance (D_0) from the origin (centered) of the transform.

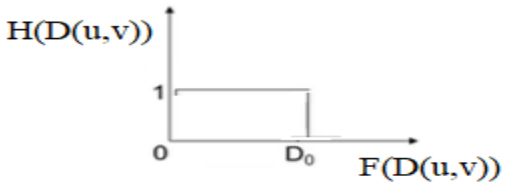
The transform function of this filter is:

$$H(u,v) = \begin{cases} 1 & if \;\; D(u,v) \le D_0 \\ 0 & if \;\; D(u,v) > D_0 \end{cases}$$

where D_0 is the cutoff frequency, and $D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$

M = no. of rows, N= no. of columns of the matrix

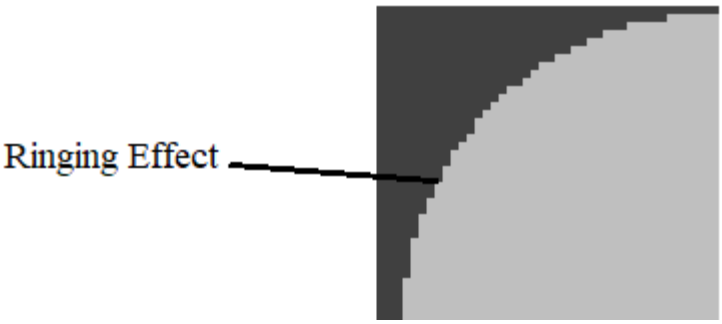
D(u,v) is the Euclidean distance between center pixel and neighbor pixels



Effect of ILPF

- Blurring effect is decrease with increasing cutoff of high frequency
- Ringing effect is decrease (become finer) with increase cutoff of high frequency

Note: *Ringing effect is the rippling (rough), artifact effect in sharp edges.*



Procedure for ILPF suruma center ma ani image lai fourrier transform garerw balla filter operation apply garinxa

Step1: Read an image

Step2: Multiply (*pixel by pixel multiplication*) the input image by $(-1)^{x+y}$ to move origin to the center, where x,y are the coordinate of pixel.

Step3: Compute F(u,v), the DFT of the image from step 2 matrix multiplication hunxa hae dft nikalda

Step4: Calculate G(u,v) by multiply F(u,v) by a filter function H(u,v) (i.e. pixel by pixel multiplication)

G(u,v) = F(u,v)* H(u,v) (*pixel by pixel multiplication*)

- Find filter function H(u,v) using Ideal Low Pass Filter with given cutoff frequency

$$H(u,v) = \begin{cases} 1 & if \;\; D(u,v) \leq D_0 \\ 0 & if \;\; D(u,v) > D_0 \end{cases}$$

where D_0 is the cutoff frequency, and $D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$

Step5: Compute the IDFT of the result from step 4

Step6: Obtain the real part of the result from step 5

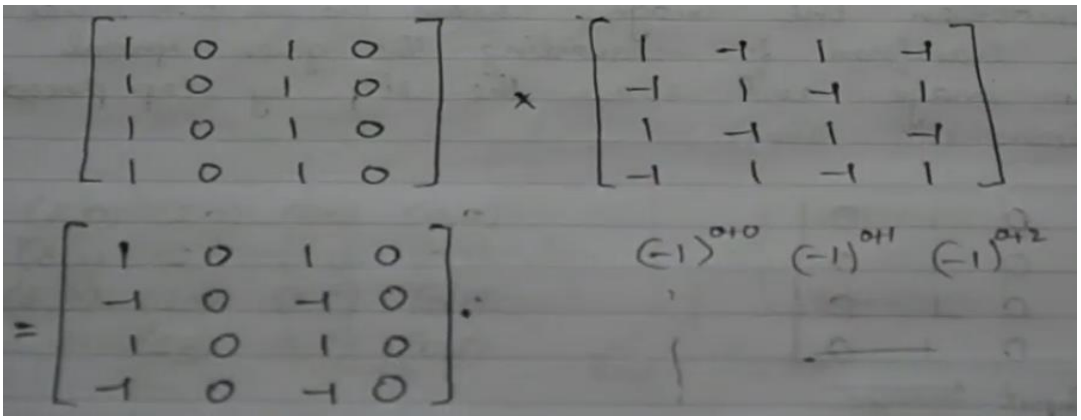
Step7: Multiply the result from step 6 by $(-1)^{x+y}$ to get origin back to original position.

Example: Convert the given spatial domain image using Fourier transform and perform Ideal Low Pass filter to smoothen the image. Choose D_0 as 0.5. Show the step by step procedure.

1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

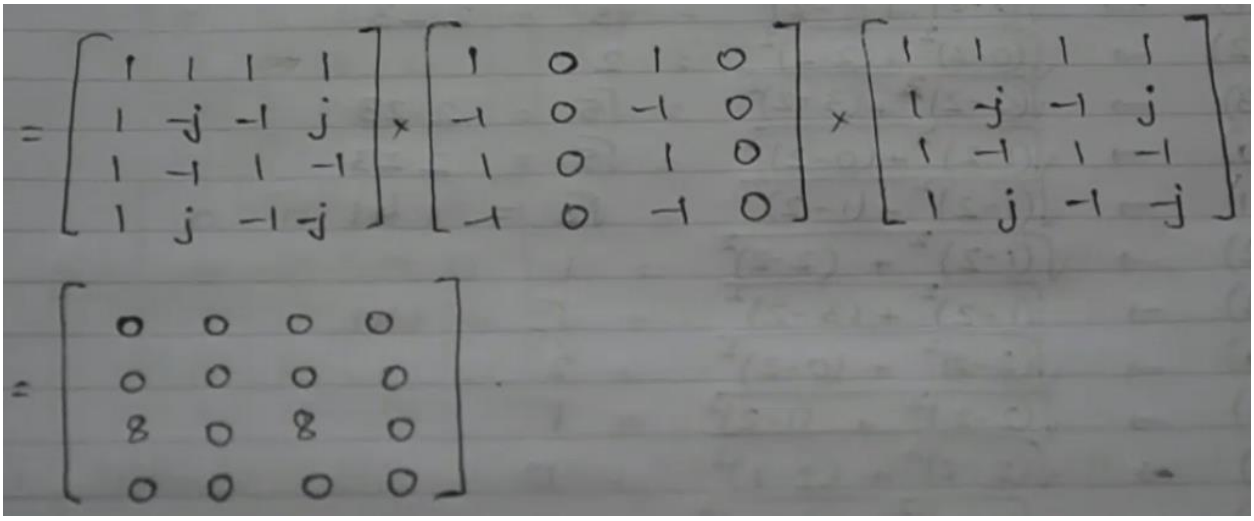
Solution:

Step 1: Multiply (pixel by pixel) the input image by $(-1)^{x+y}$ to move origin to the center



Step 2: Compute the DFT of the image from step 1

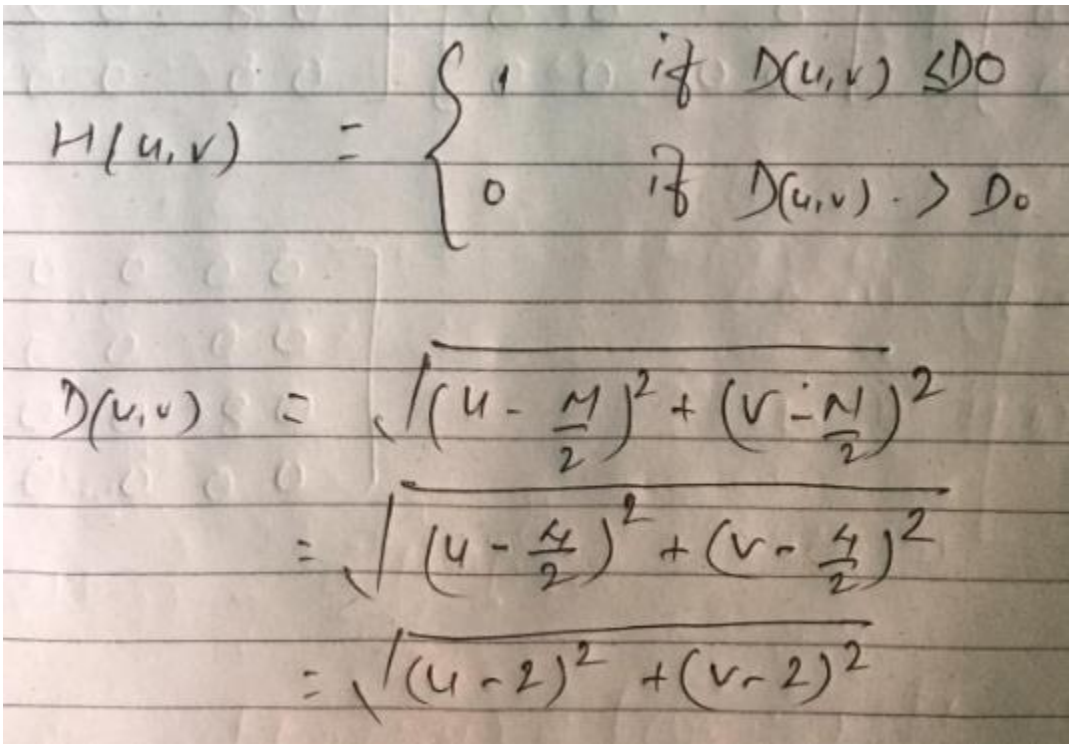
$F(u,v) = Mask*F(x,y)*Mask^T$ (Matrix multiplication)



Step 3: Multiply F(u,v) with the filter function H(u,v) to calculate G(u,v)

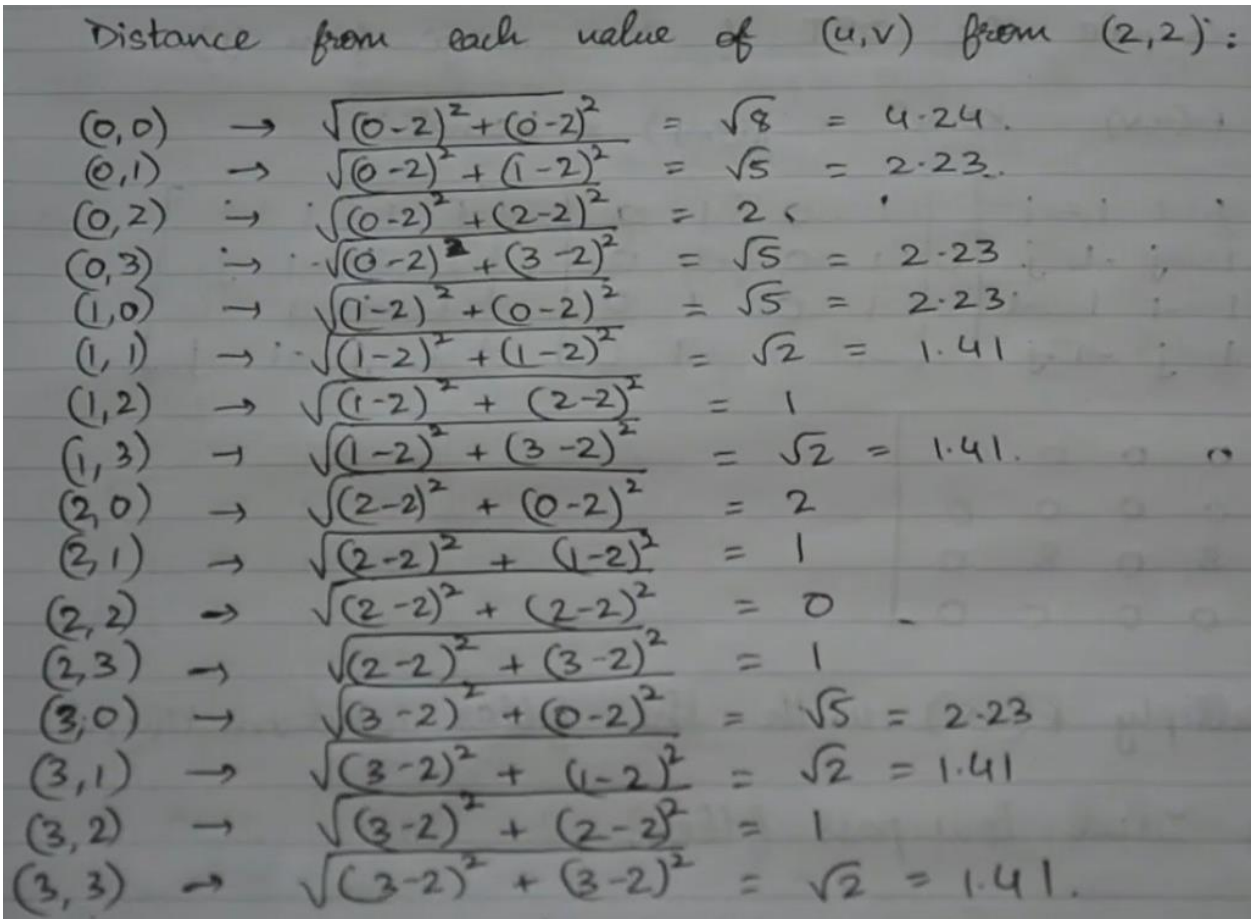
i.e. G(u,v) = F(u,v)*H(u,v) (pixel by pixel multiplication)

First find H(u,v) for Ideal Low Pass filter

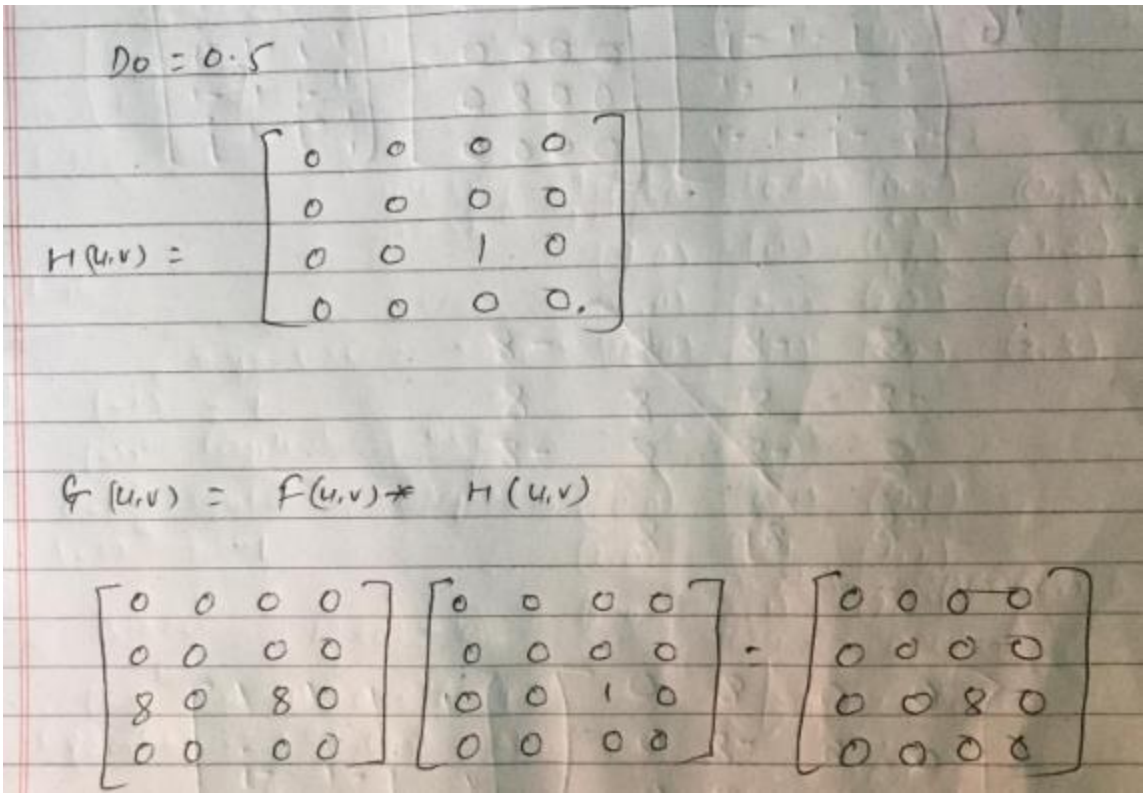


Given matrix is 4x4 so M/2 = 4/2 = 2 and N/2 = 4/2 = 2.

Here we are calculating the distance between (u,v) from the center (2,2) of the mask



$$D(u,v) = \begin{bmatrix} 4.24 & 2.23 & 2 & 2.23 \\ 2.23 & 1.41 & 1 & 1.41 \\ 2 & 1 & 0 & 1 \\ 2.23 & 1.41 & 1 & 1.41 \end{bmatrix}.$$



*Here, F(u,v)*H(u,v) is the direct pixel multiplication*

Step 4: Compute the IDFT of the result from step 3

F(u,v) = (1/m*n) (mask*f(x,y)*mask) (Matrix multiplication)

(1/16)*

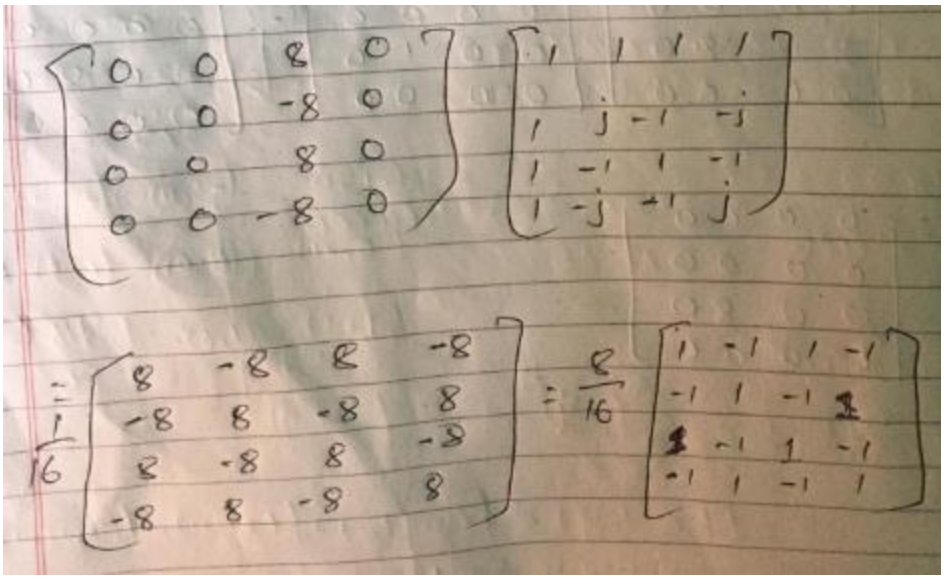
1	1	1	1
1	j	-1	-j
1	-1	1	-1
1	-j	-1	j

*

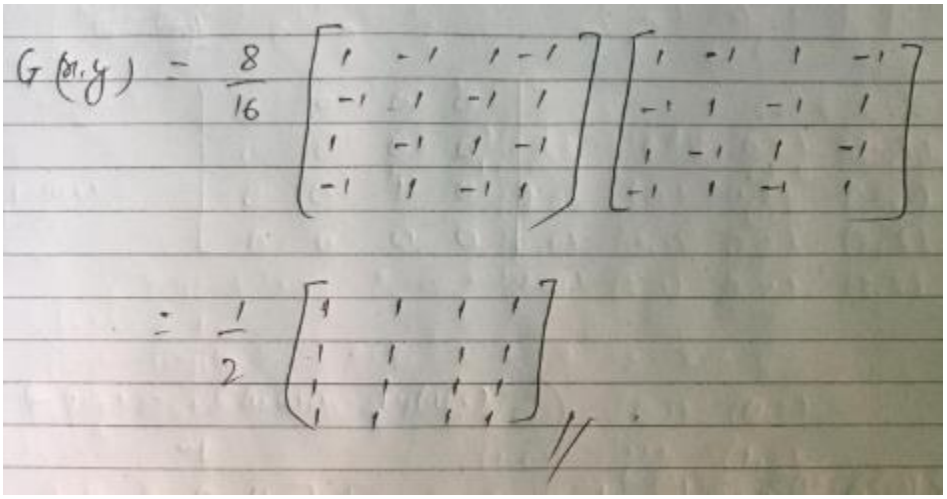
0	0	0	0
0	0	0	0
0	0	8	0
0	0	0	0

*

1	1	1	1
1	j	-1	-j
1	-1	1	-1
1	-j	-1	j



Step 5: Multiply (pixel by pixel multiplication) the result by $(-1)^{x+y}$ to move the origin to its original position



Butterworth Low Pass Filter

tyo h(u,v) aaune wala hunxa frequency response of pass band ma aune data ustae ustae hunxa hola so flat

A Butterworth Filter is a type of Active Filter, where the frequency response of the across its pass band is relatively flat. Because of this frequency response, Butterworth Filters are also known as Maximally Flat Filters or Flat-Flat Filters.

Unlike the ILPF, the BLPF transfer function does not have a sharp discontinuity that gives a clear cutoff between passed and filter frequencies.

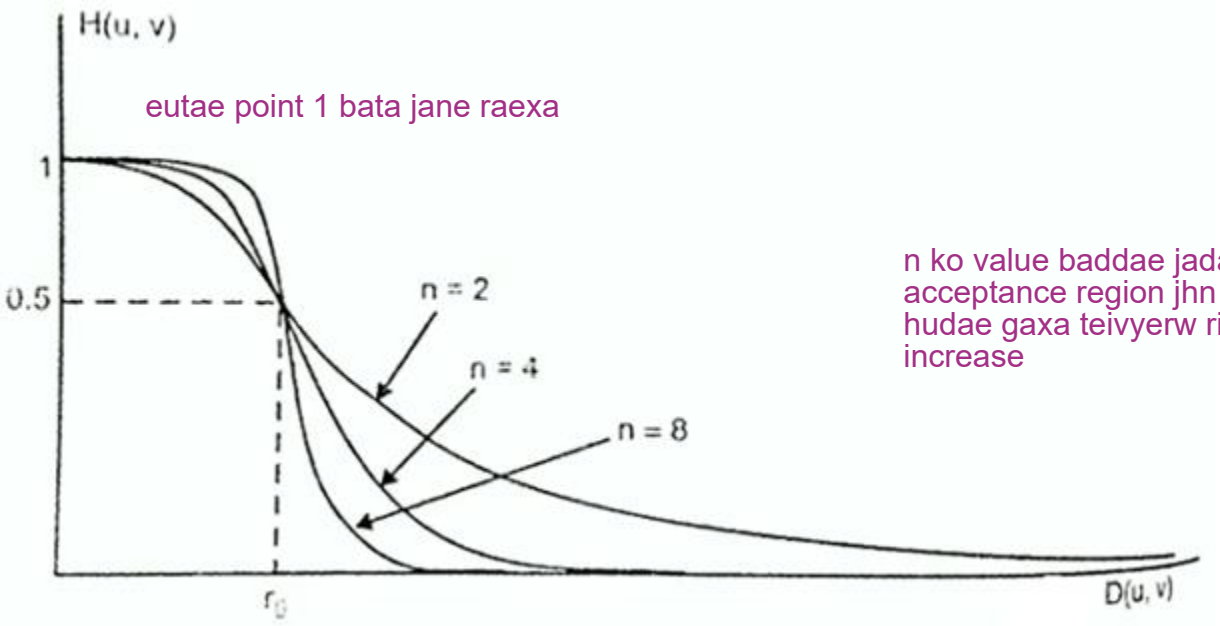
BLPF with frequency D_0 is defined by

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$

Where, n is order of filter

Effect of BLPF

- If n increases, increase ringing effect
- If n is 1, it produce no ringing effect



n ko value baddae jada acceptance region jhn kam kam hudae gaxa teivyerw ringing factor increase

- The magnitude response is nearly constant (equals to 1) at lower frequencies.
- There are no ringing effect in pass-band and stop-band

Procedure for BLPF

Step1: Read an image

Step2: Multiply (pixel by pixel multiplication) the input image by $(-1)^{x+y}$ to move origin to the center, where x,y are the coordinates of image matrix.

Step3: Compute F(u,v), the DFT of the image from step 2

Step4: Calculate G(u,v) by multiply F(u,v) by a filter function H(u,v) (i.e. pixel by pixel multiplication)

G(u,v) = F(u,v)* H(u,v) (pixel by pixel multiplication)

- Find filter function H(u,v) using Butterworth Low Pass Filter with given cutoff frequency

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$

Where, n is the order of filter, i.e. 1, 2, 4, 8,… Take any one of these values

Step5: Compute the IDFT of the result from step 4

Step6: Obtain the real part of the result from step 5

Step7: Multiply (pixel by pixel multiplication) the result from step 6 by $(-1)^{x+y}$ to get origin back to original position.

Gaussian Low Pass Filter: ILPF ma tw 0 ki 1 hunxa so clear cutoff between passed and filtered frequencies

Unlike ILPF, the GLPF transfer function does not have a sharp transition that establishes a clear cutoff between passed and filtered frequencies. Instead, GLPF has a smooth transition between low and high frequencies.

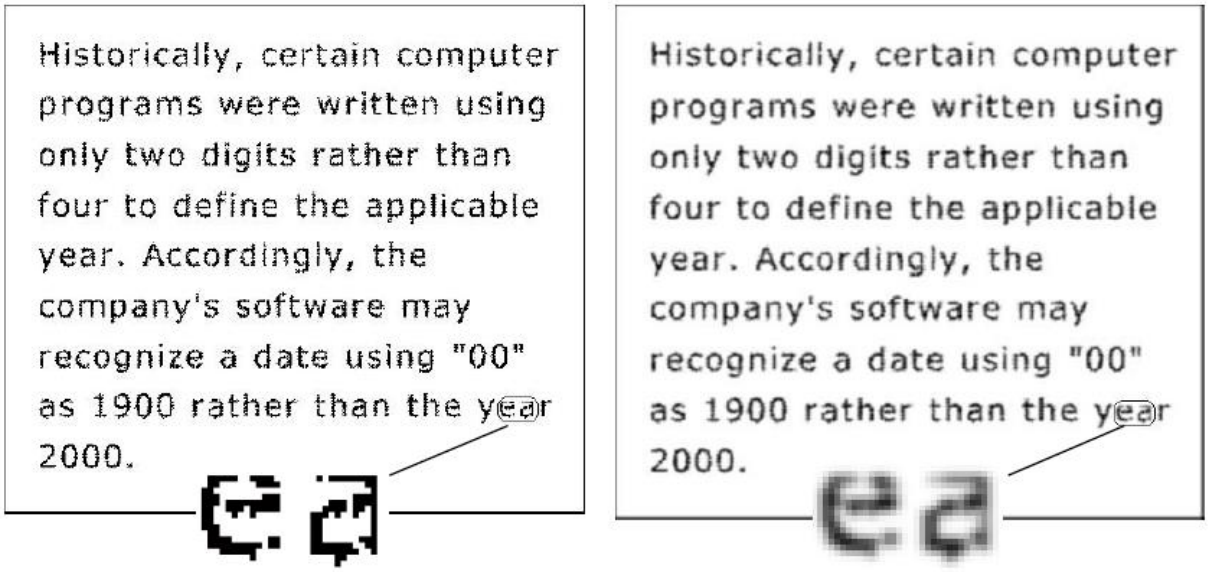
GLPF with frequency D_0 is defined as:

$$H(u,v) = e^{-D^2(u,v)/2D_0{}^2}$$

Effect of GLPF

- Smooth transition in blurring with increasing cutoff frequency
- No ringing effect

Smoothing (low pass) filter is useful in many applications. For example, it can be used to bridge small gaps in broken characters by blurring it.



Procedure for GLPF

Step1: Read an image

Step2: Multiply (pixel by pixel multiplication) the input image by $(-1)^{x+y}$ to move origin to the center, where x,y are the coordinates of image matrix.

Step3: Compute F(u,v), the DFT of the image from step 2

Step4: Calculate G(u,v) by multiply F(u,v) by a filter function H(u,v) (i.e. pixel by pixel multiplication)

G(u,v) = F(u,v)* H(u,v) (pixel by pixel multiplication)

- Find filter function H(u,v) using Gaussian Low Pass Filter with given cutoff frequency

$$H(u,v) = e^{-D^2(u,v)/2D_0{}^2}$$

Where, $D^2(u,v) = (u-M/2)^2 + (v-N/2)^2$ and M and N are row and column of input image matrix.

Step5: Compute the IDFT of the result from step 4

Step6: Obtain the real part of the result from step 5

Step7: Multiply (pixel by pixel multiplication) the result from step 6 by $(-1)^{x+y}$ to get origin back to original position.

Sharpening Frequency Domain Filter (Highpass filter):

This filter technique is used to remove low frequency components of the image and keep high frequency components. Hence, it is sharpening the original image.

We can get high pass by subtracting low pass by 1.

$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

Following techniques are used for this filter process:

- Ideal High Pass Filter
- Butterworth High Pass Filter
- Gaussian High Pass Filter

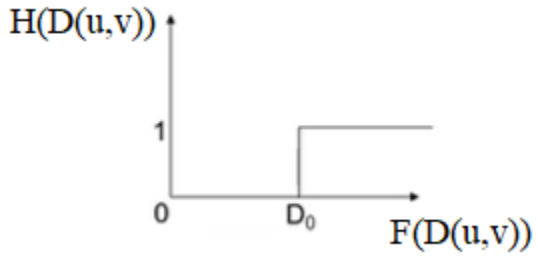
Ideal High Pass Filter:

ILPF is the simplest low pass filter technique that "cuts off" all low frequency components of the DFT that are at a distance less than or equals to a specified distance (D_0) from the origin (centered) of the transform.

The transform function of this filter is:

$$H(u,v) = \begin{cases} 1 & if \;\; D(u,v) > D_0 \\ 0 & if \;\; D(u,v) \leq D_0 \end{cases}$$

where D_0 is the cutoff frequency, and $D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$



Effect of IHPF

- Ringing effect is decrease with cutoff frequency increases
- Edge distortion (Thickness of object boundary) is decrease with cutoff frequency increases

Procedure for IHPF

Step1: Read an image

Step2: Multiply (pixel by pixel multiplication) the input image by $(-1)^{x+y}$ to move origin to the center

Step3: Compute F(u,v), the DFT of the image from step 2

Step4: Calculate G(u,v) by multiply F(u,v) by a filter function H(u,v) (i.e. pixel by pixel multiplication)

G(u,v) = F(u,v)* H(u,v) (pixel by pixel multiplication)

- Find filter function H(u,v) using Ideal High Pass Filter with given cutoff frequency

$$H(u,v) = \begin{cases} 1 & if\ \ D(u,v) > D_0 \\ 0 & if\ \ D(u,v) \le D_0 \end{cases}$$

where D_0 is the cutoff frequency, and $D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$

Step5: Compute the IDFT of the result from step 4

Step6: Obtain the real part of the result from step 5

Step7: Multiply (pixel by pixel multiplication) the result from step 6 by $(-1)^{x+y}$ to get origin back to original position.

Butterworth High Pass Filter:

BHPF with frequency D_0 is defined by

$$H(u,v) = \frac{1}{1 + \frac{1}{[D(u,v)/D_0]^{2n}}}$$

Where, n is order of filter

Effect of BHPF

- If n increases, the filter become shaper and increase ringing effect
- If n is 1, it produce no ringing effect

Procedure for BHPF

Step1: Read an image

Step2: Multiply (pixel by pixel multiplication) the input image by $(-1)^{x+y}$ to move origin to the center, where x,y are the coordinates of input image matrix.

Step3: Compute F(u,v), the DFT of the image from step 2

Step4: Calculate G(u,v) by multiply F(u,v) by a filter function H(u,v) (i.e. pixel by pixel multiplication)

G(u,v) = F(u,v)* H(u,v) (pixel by pixel multiplication)

- Find filter function H(u,v) using Butterworth High Pass Filter with given cutoff frequency

$$H(u,v) = \frac{1}{1 + \frac{1}{[D(u,v)/D_0]^{2n}}}$$

Where, n is the order of filter, i.e. 1,2,4,8,… chose one of these values

Step5: Compute the IDFT of the result from step 4

Step6: Obtain the real part of the result from step 5

Step7: Multiply (pixel by pixel multiplication) the result from step 6 by $(-1)^{x+y}$ to get origin back to original position.

Gaussian High Pass Filter:

The transfer function of the G HPF with cutoff frequency at distance D_0 from the center of the frequency rectangle defined as:

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0{}^2}$$

Effect of GHPF

- No ringing effect
- Less edge distortion
- Result is smoother then IHPF

Procedure for GHPF

Step1: Read an image

Step2: Multiply (pixel by pixel multiplication) the input image by $(-1)^{x+y}$ to move origin to the center, where x,y are the coordinates of input image matrix

Step3: Compute F(u,v), the DFT of the image from step 2

Step4: Calculate G(u,v) by multiply F(u,v) by a filter function H(u,v) (i.e. pixel by pixel multiplication)

G(u,v) = F(u,v)* H(u,v) (pixel by pixel multiplication)

- Find filter function H(u,v) using Gaussian High Pass Filter with given cutoff frequency

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0{}^2}$$

Where, $D^2(u,v) = (u-M/2)^2 + (v-N/2)^2$, M and N are the row and column of input image matrix.

Step5: Compute the IDFT of the result from step 4

Step6: Obtain the real part of the result from step 5

Step7: Multiply (pixel by pixel multiplication) the result from step 6 by $(-1)^{x+y}$ to get origin back to original position.

Laplacian Filter in Frequency Domain

The Laplacian can be implemented in frequency domain by using the filter shift to the center.

$H(u,v) = -4\lambda^2 [u^2 + v^2]$

If center is shifted at P/2 and Q/2

$H(u,v) = -4\lambda^2 [(u-P/2)^2 + (v-Q/2)^2]$

Where, P and Q are order of matrix

$H(u,v) = -4\lambda^2 D^2(u,v)$

Were, pie = 3.14

Procedure for GHPF

Step1: Read an image

Step2: Multiply (pixel by pixel multiplication) the input image by $(-1)^{x+y}$ to move origin to the center, where x,y are the coordinates of input image matrix.

Step3: Compute F(u,v), the DFT of the image from step 2

Step4: Calculate G(u,v) by multiply F(u,v) by a filter function H(u,v) (i.e. pixel by pixel multiplication)

G(u,v) = F(u,v)* H(u,v) (pixel by pixel multiplication)

- Find filter function H(u,v) using Laplacian Filter in frequency domain
  $H(u,v) = -4\lambda^2 D^2(u,v)$
  Where, pie = 3.14

Step5: Compute the IDFT of the result from step 4

Step6: Obtain the real part of the result from step 5

Step7: Multiply (pixel by pixel multiplication) the result from step 6 by $(-1)^{x+y}$ to get origin back to original position.

Fast Fourier Transform:

For the implementation of DFT it requires on the order of $(MN)^2$ multiplications and additions where as Fast Fourier Transform (FFT) reduces computation to the order of $MN*\log_2(MN)$ multiplication and additions. FFT is an algorithm to reduce the computation time for DFT.

For image of size (1024x1024) pixel requires trillion multiplications and additions for just one DFT. This would be challenge even for super computers. In another hand, FFT requires order of 20 million multiplication and addition for same size of image. This is the significant reduction of computation process.

The separability feature of DFT states that 2-D DFT of f(x,y) can be obtained by computing the 1-D transform of each row of f(x,y) and then computing 1-D transform of each column of the result. This is an important simplification because we have to deal only with one variable at a time. On the basis of this feature we need to focus only on the FFT of one variable.

Derivation of 1-D FFT

1-D FFT equation is defined as

$$\mathbf{F(u)} = \sum_{x=0}^{M-1} f(x)\, W_M^{ux}$$

Where, $\boldsymbol{u}$ *= 0,1,2……., M-1*

$\boldsymbol{W}$ *is the twiddle factor used to speed-up the computation of DFT and IDFT*

$$W_M = e^{-j2\lambda/M}$$

And M is assume to be the form of

$$\boldsymbol{M = 2^n}$$

With **n** being a positive integer, M can be expressed as

$$\boldsymbol{M=2K}$$

With **K** being a positive integer, by substitution the values above equation becomes

$$\mathbf{F(u)}= \sum_{x=0}^{2K-1} f(x)\, W_{2K}^{ux}$$

We can divides this equation in to two parts even function and odd function

$$\mathbf{F(u)} = \sum_{x=0}^{K-1} f(2x)\, W_{2K}^{u(2x)} + \sum_{x=0}^{K-1} f(2x+1)\, W_{2K}^{u(2x+1)}$$

However, u and K are the positive integer so, we can write

$$W_{2K}^{2ux} = W_K^{ux}$$

The the above equation becomes

$$\mathbf{F(u)}= \sum_{x=0}^{K-1} f(2x)\, W_K^{ux} + \sum_{x=0}^{K-1} f(2x+1)\, W_K^{ux} W_{2K}^{u}$$

Now, the even and odd function can be define

$$\mathbf{F_{even}\,(u)} = \sum_{x=0}^{K-1} f(2x)\, W_K^{ux}$$

For u = 0,1,2,3……..,K-1

$$\mathbf{F_{odd}\,(u)} = \sum_{x=0}^{K-1} f(2x+1)\, W_K^{ux}$$

For u = 0,1,2,3……..,K-1

Now the 1-D FFT equation is:

$$\mathbf{F(u)} = F_{even}\,(u) + F_{odd}\,(u)\, W_{2K}^{u}$$

The Fast Fourier Transform (FFT) is a way to reduce the complexity of the Fourier transform computation from $O(n^2)$ *to* $O(n*\log(n))$, which is a dramatic improvement. Hence, the time complexity of FFT is ***O(n*log(n))***

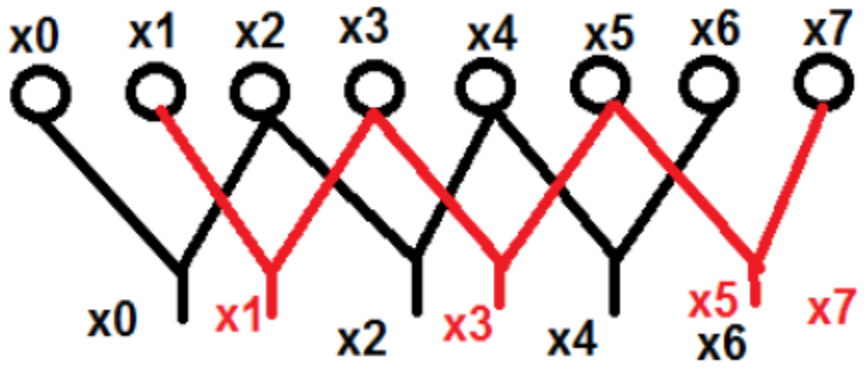
Complexity analysis:

Time complexity of DFT = $O(n^2)$

Time complexity of FFT = O(n*log(n))

Let us take an example to understand it better. We have considered eight points named from ***x0*** to ***x7***. We will choose the even terms in one group and the odd terms in the other.

Diagrammatic view of the above said has been shown below



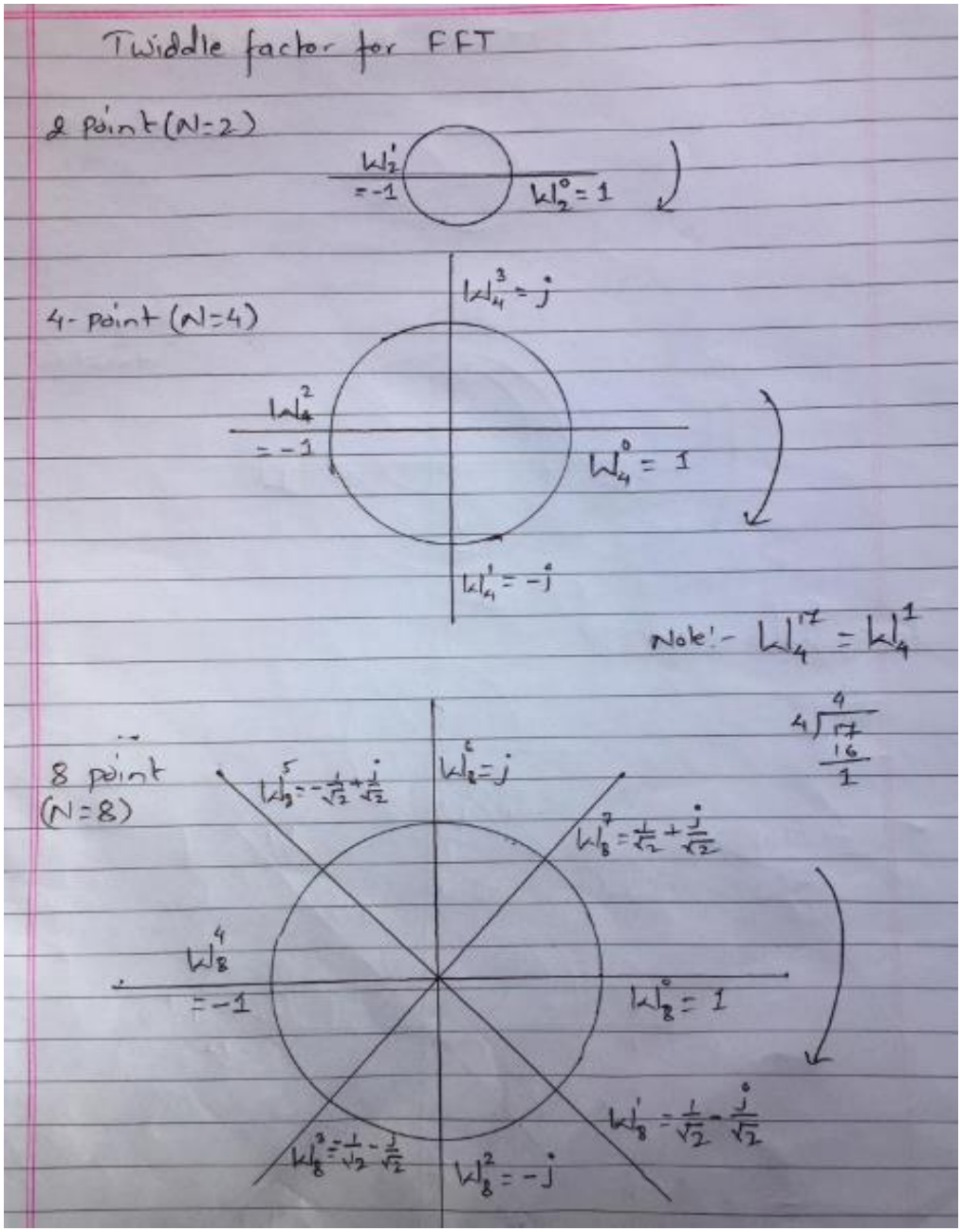
Here, points x_0, x_2, x_4 and x_6 have been grouped into one category and similarly, points x_1, x_3, x_5 and x_7 has been put into another category.

Now, we can further make them in a group of two and can proceed with the computation.

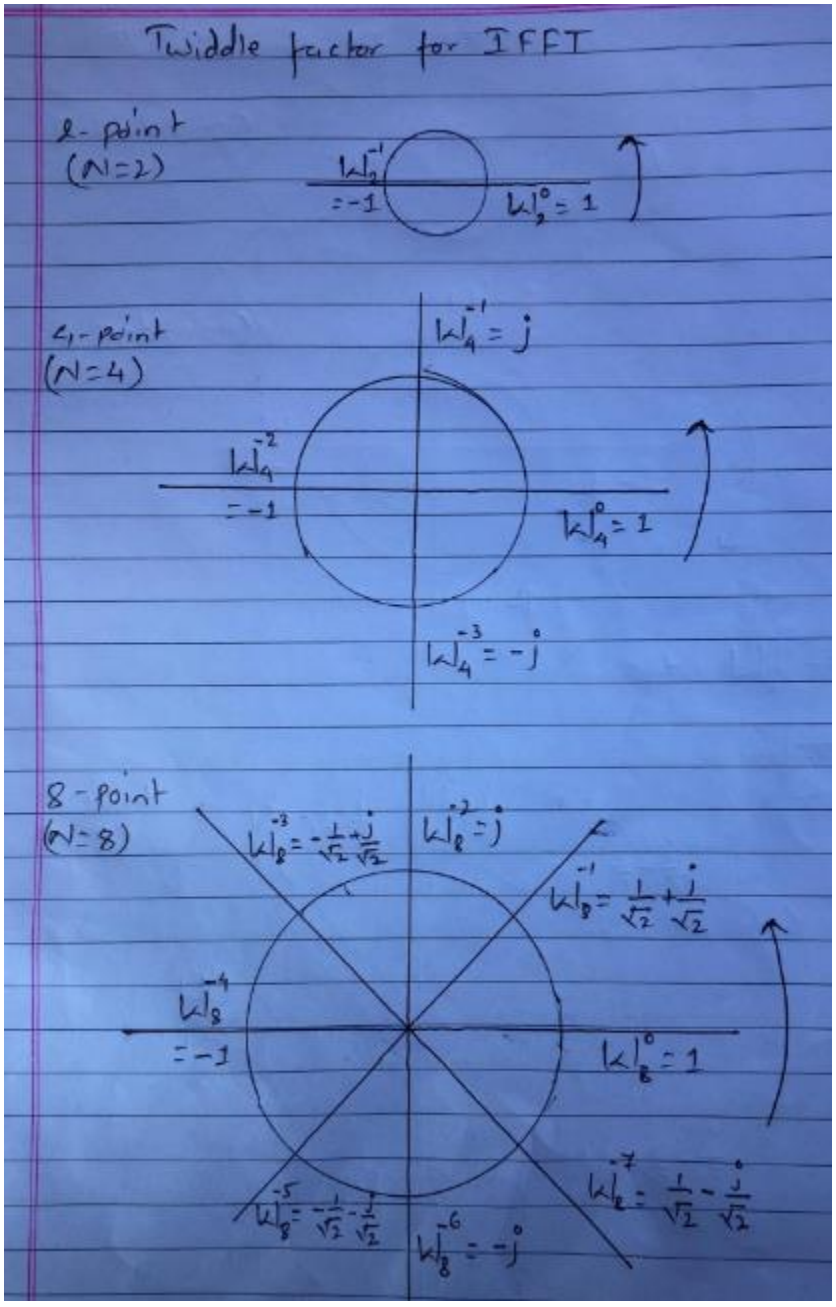
Twiddle Factor:

A **twiddle factor**, in fast Fourier transform (FFT) algorithms, is any of the trigonometric constant coefficients that are multiplied by the data in the course of the algorithm. This remains the term's most common meaning, but it may also be used for any data-independent multiplicative constant in an FFT. It is used to speed up the operation of DFT and IDFT.

Twiddle factor for FFT:



Twiddle factor for IFFT:



FFT:

FFT is used to change data from spatial domain to frequency domain

Spatial domain (X_n) -------→ Frequency Domain (X_k)

FFT can be divided in to two parts

- Decimation in Time (DIT)
- Decimation in Frequency (DIF)

Decimation in Time (DIT-FFT)

Decimation means signal flow variation. In DIT-FFT sequence of input data in spatial domain (X_n) have order variation.

X_n ----------→ X_k change gaererw even even lagatar odd odd lagatar testo grxa (Change)

This change means change in occurrence of sequence in spatial domain data.

First divide the given spatial domain data (X_n) into two parts according to Odd and Even sequence numbers.

For 4 point (N=4):

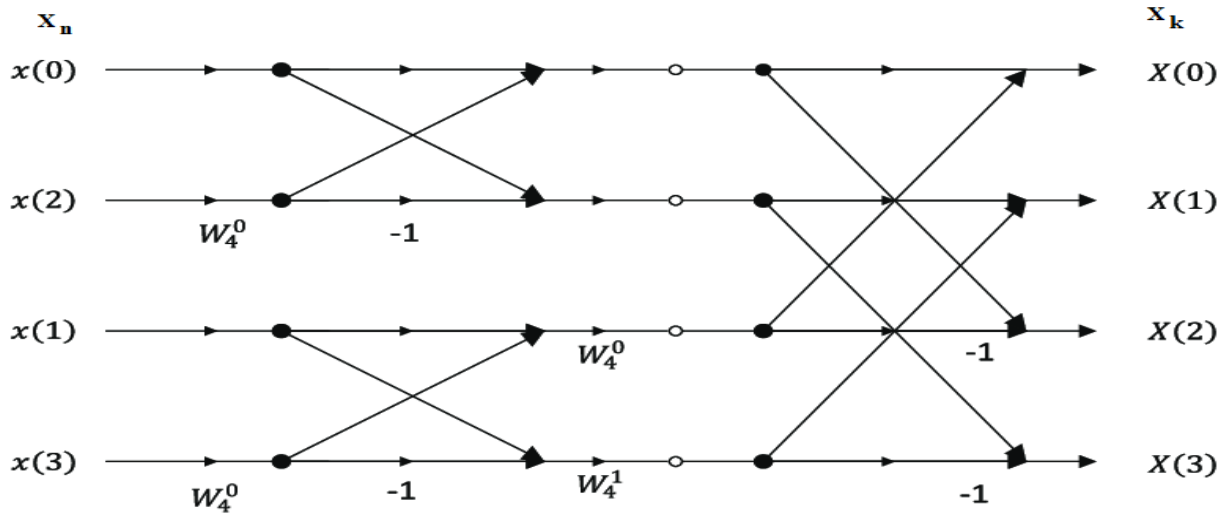
Even = (0,2) Odd = (1,3)

For 8 point (N=8):

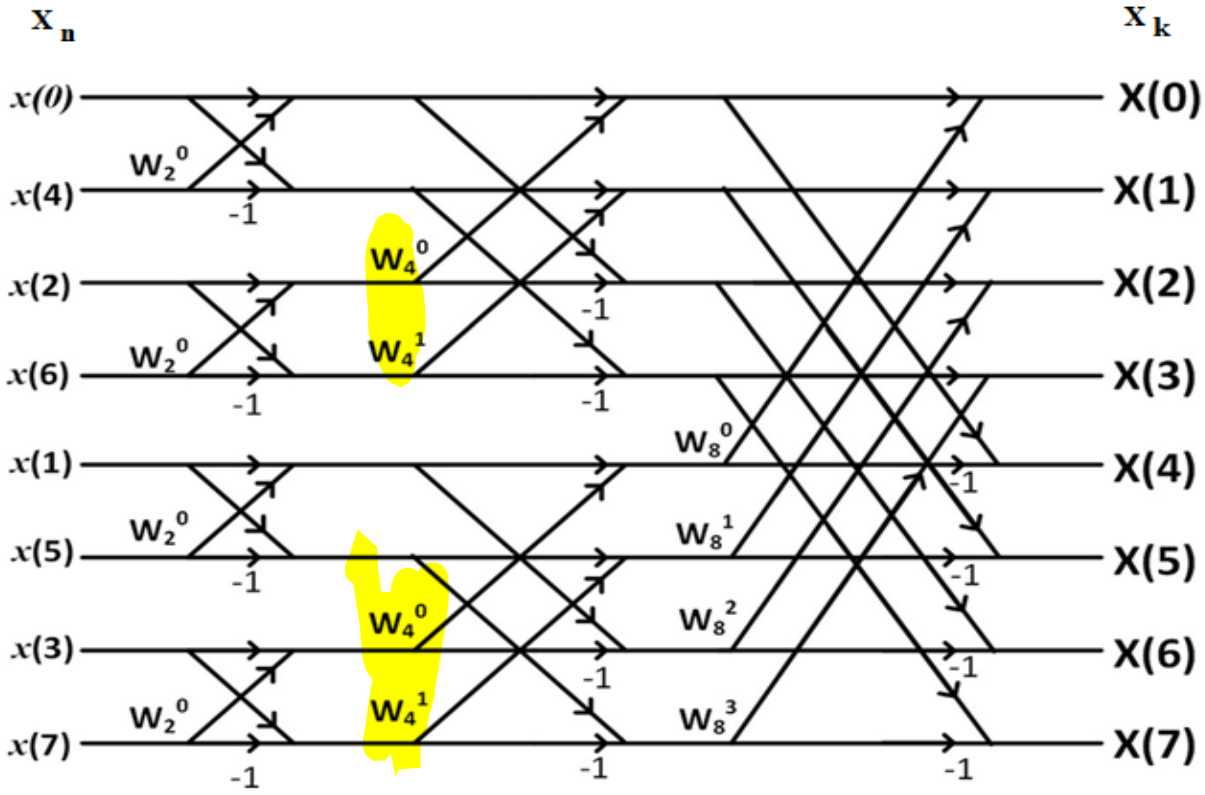
Even = (0,4,2,6) Odd = (1,5,3,7)

Butterfly diagram for DIT-FFT:

For 4-point (N=4):



For 8-point (N=8):



Decimation in Frequency (DIF-FFT)

In DIF-FFT sequence of output data in frequency domain (X_k) have order variation.

X_n ----------→ X_k
(Change)

This change means change in occurrence of sequence in frequency domain data.

For 4 point (N=4):

Even = (0,2) Odd = (1,3)

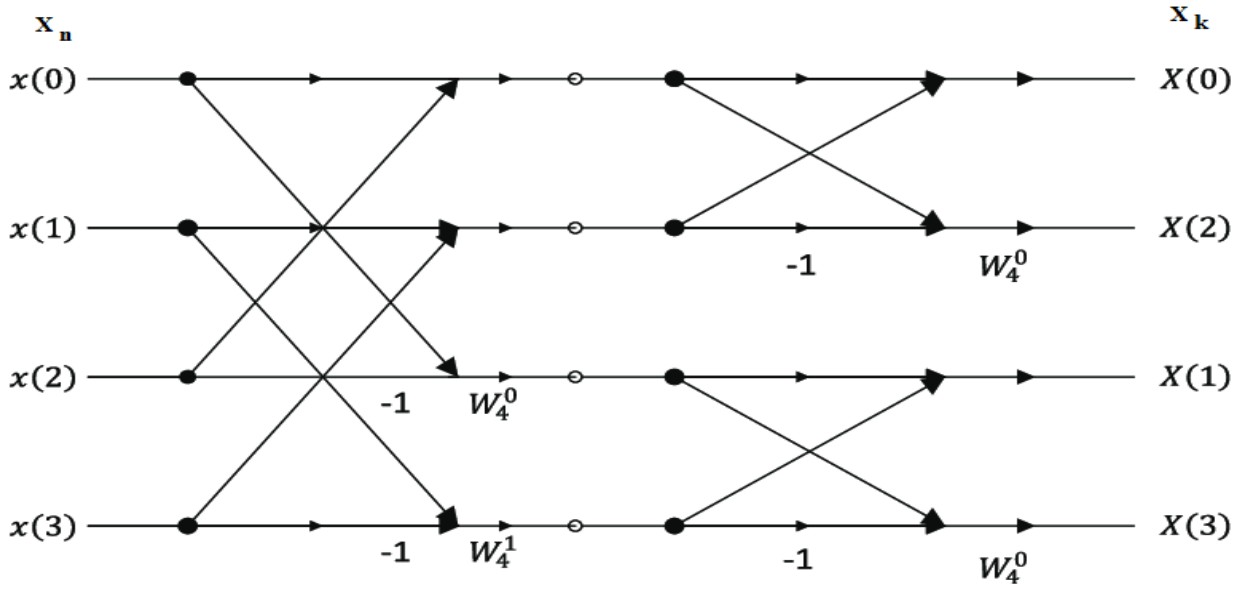
For 8 point (N=8):

Even = (0,4,2,6) Odd = (1,5,3,7)

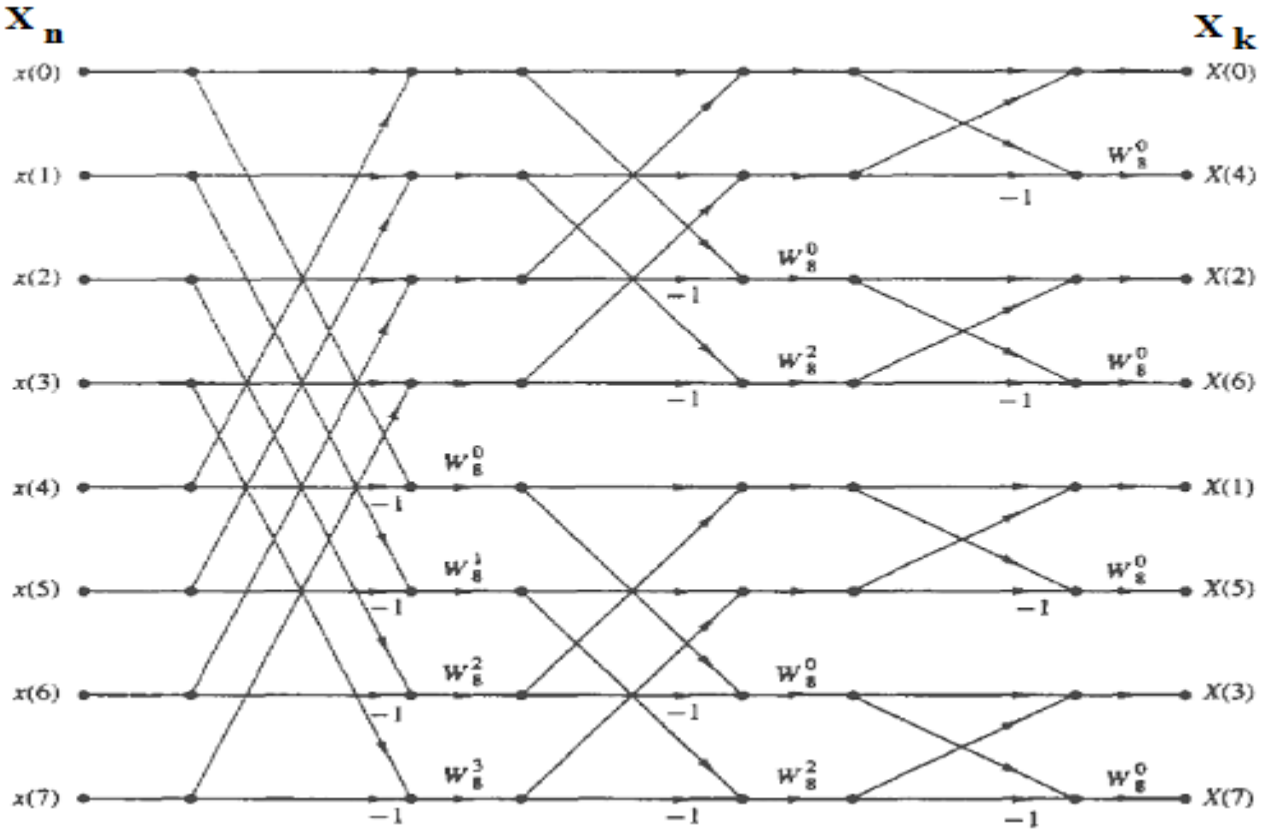
Here the data occurrence in X_n is changed.

Butterfly Diagram for DIF-FFT:

For 4-point (N=4):



For 8-point (N=8):



IFFT:

IFFT is used to change data from frequency domain to spatial domain

Frequency domain (X_k) -------→ Spatial Domain (X_n)

IFFT can be divided in to two parts

- Decimation in Time (DIT)
- Decimation in Frequency (DIF)

Decimation in Time (DIT-IFFT):

In DIT-IFFT sequence of output data in spatial domain (X_n) have order variation.

X_k ----------→ X_n
(Change)

This change means change in occurrence of sequence in spatial domain data.

For 4 point (N=4):

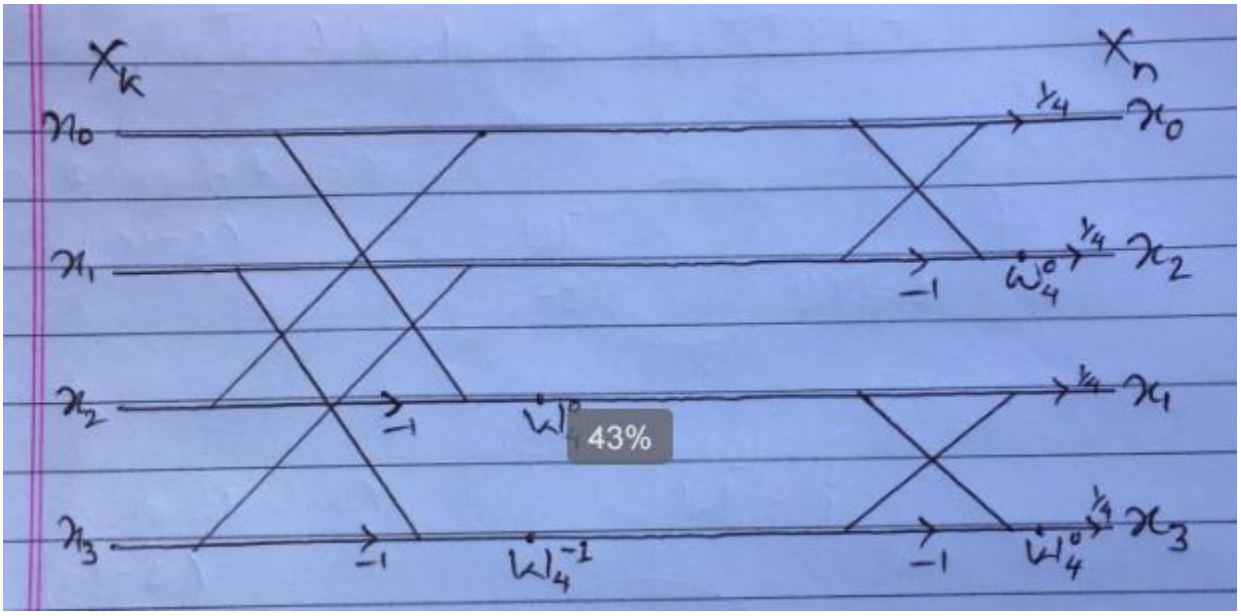
Even = (0,2) Odd = (1,3)

For 8 point (N=8):

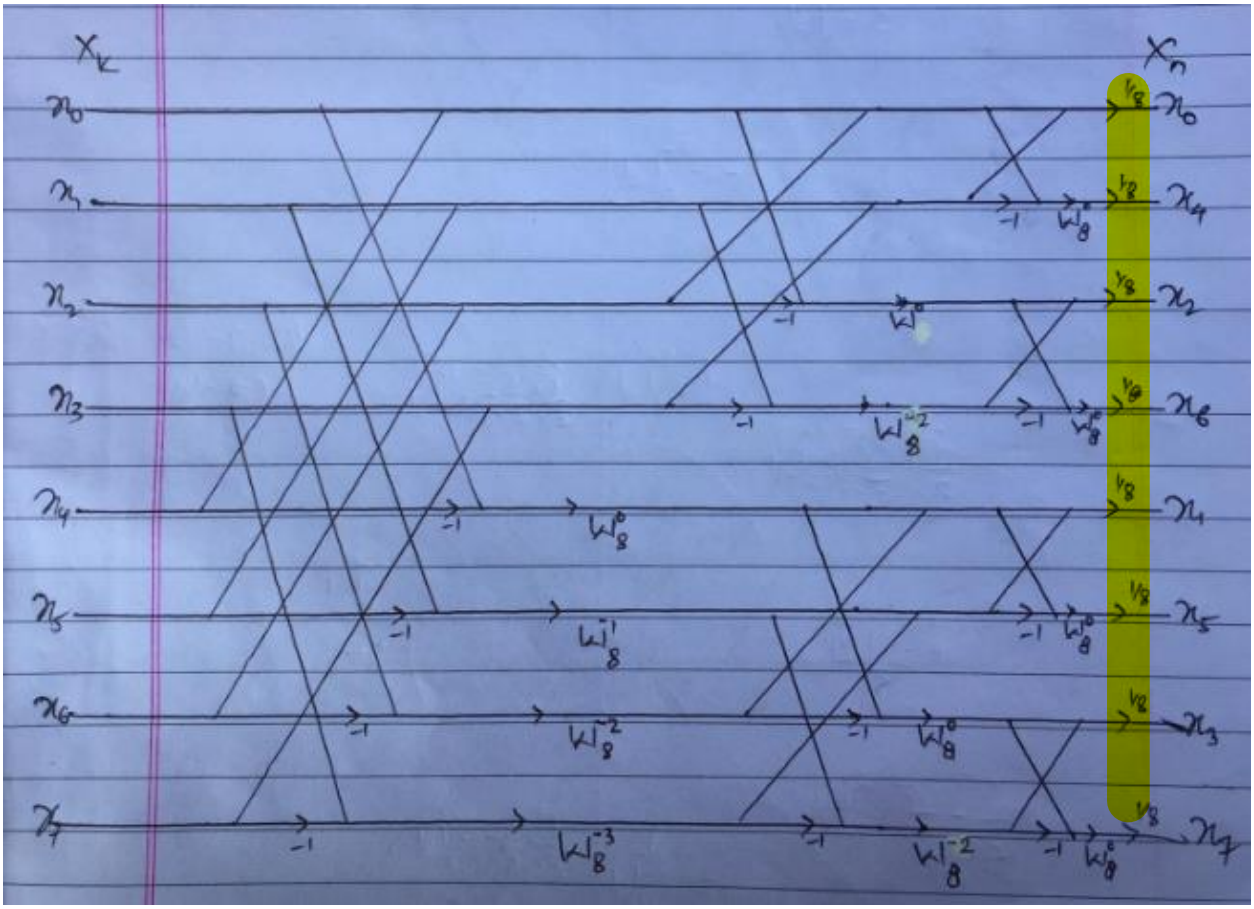
Even = (0,4,2,6) Odd = (1,5,3,7)

Butterfly Diagram for DIT-IFFT:

For 4 point (N=4):

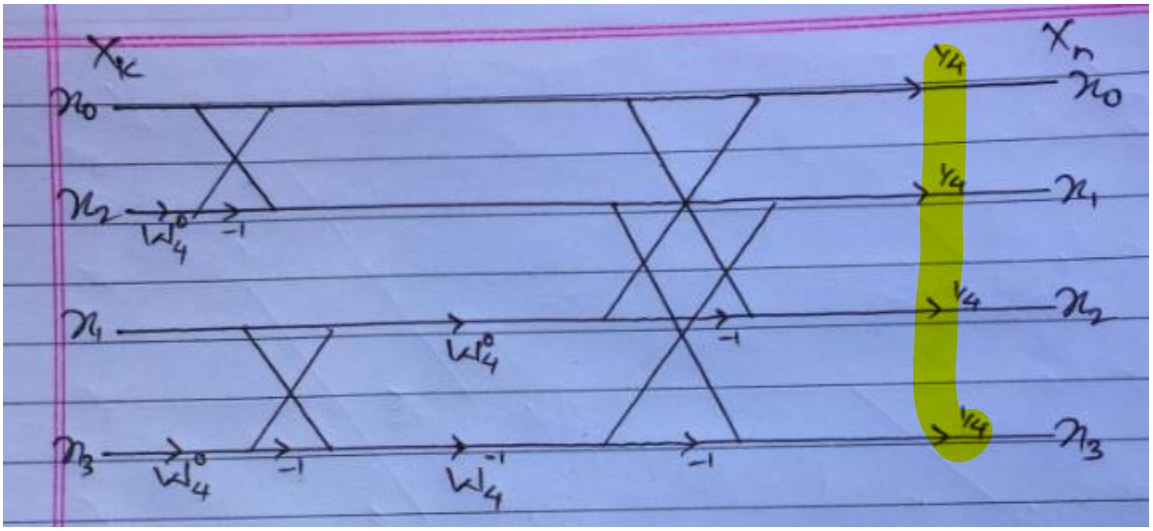


For 8 point (N=8):

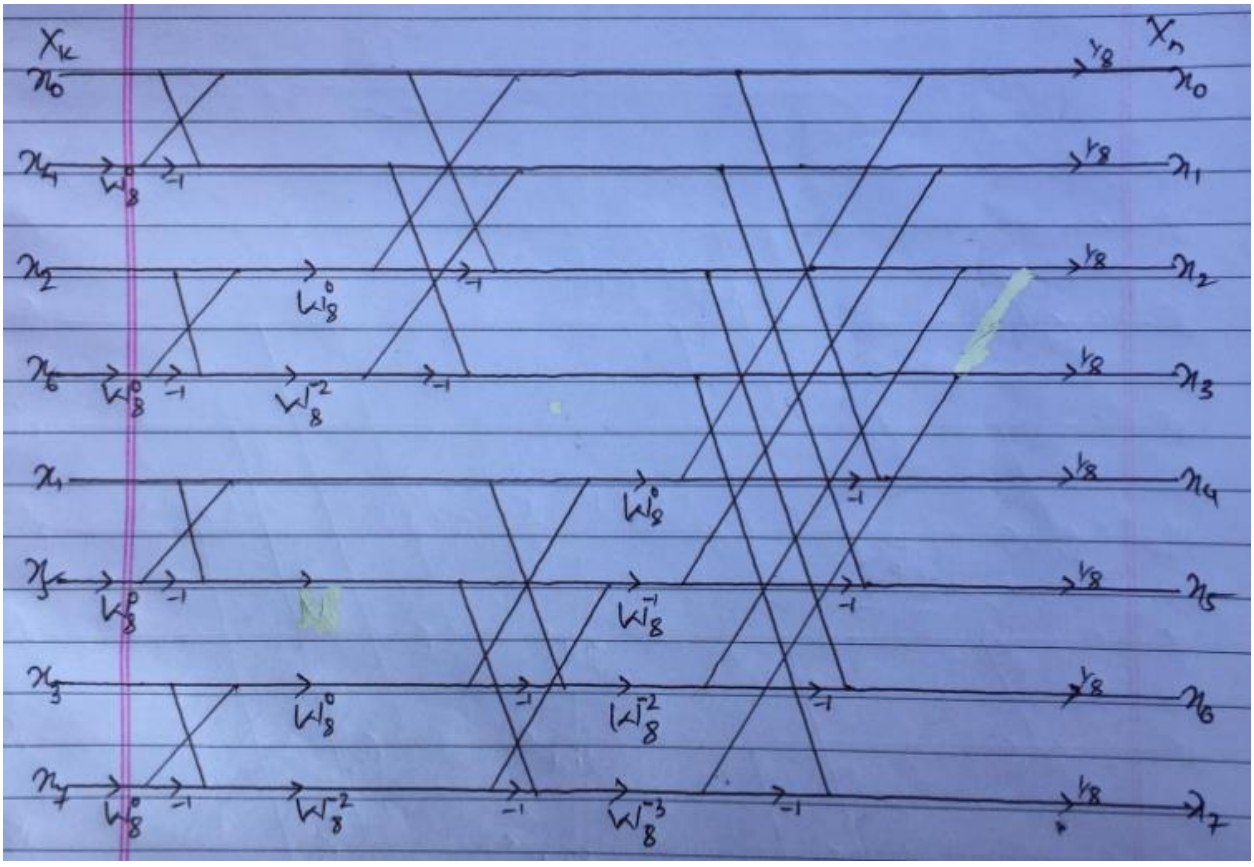


Butterfly Diagram for DIF-IFFT:

For 4 point (N=4):



For 8 point (N=8):

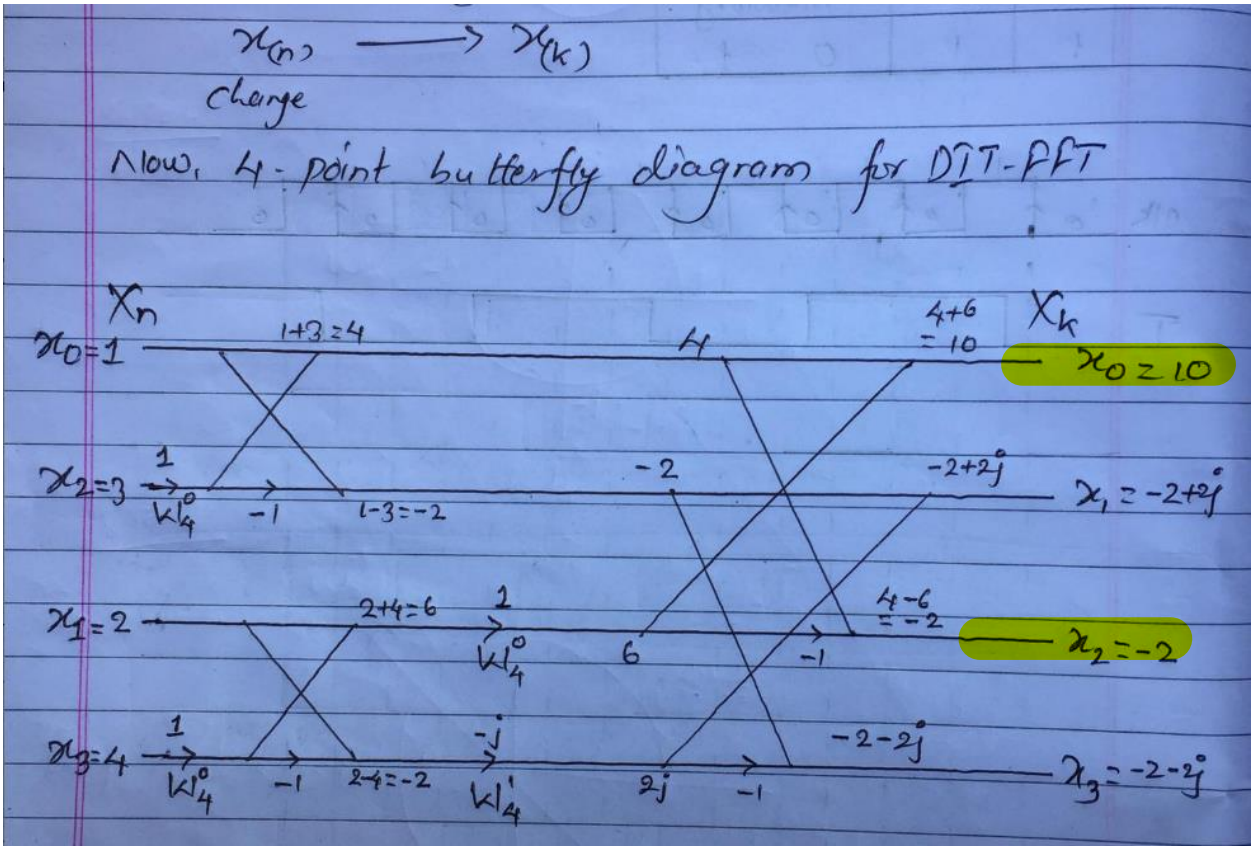


Example 1: Find the DFT of given function using DIT-FFT

f(x) = {1,2,3,4}.

Solution:

Here, we have to transfer data from spatial domain to frequency domain.



Hence, X_k = {10, -2+2j, -2, -2-2j} this is the DFT of given f(x).

3 step verification of result:

1. $(N/2)^{th}$ term of the result should be real number.
   Here, N/2 = 4/2 = 2, so 2^{th} position number is real i.e. -2, verified.
2. 1^{st} term of result should be equals to sum of f(x).
   Here, sum of f(x) = 10, and 1^{st} term of result is also 10, verified.
3. Result should be Complex Conjugate.
   Here, Result is complex conjugate, verified.

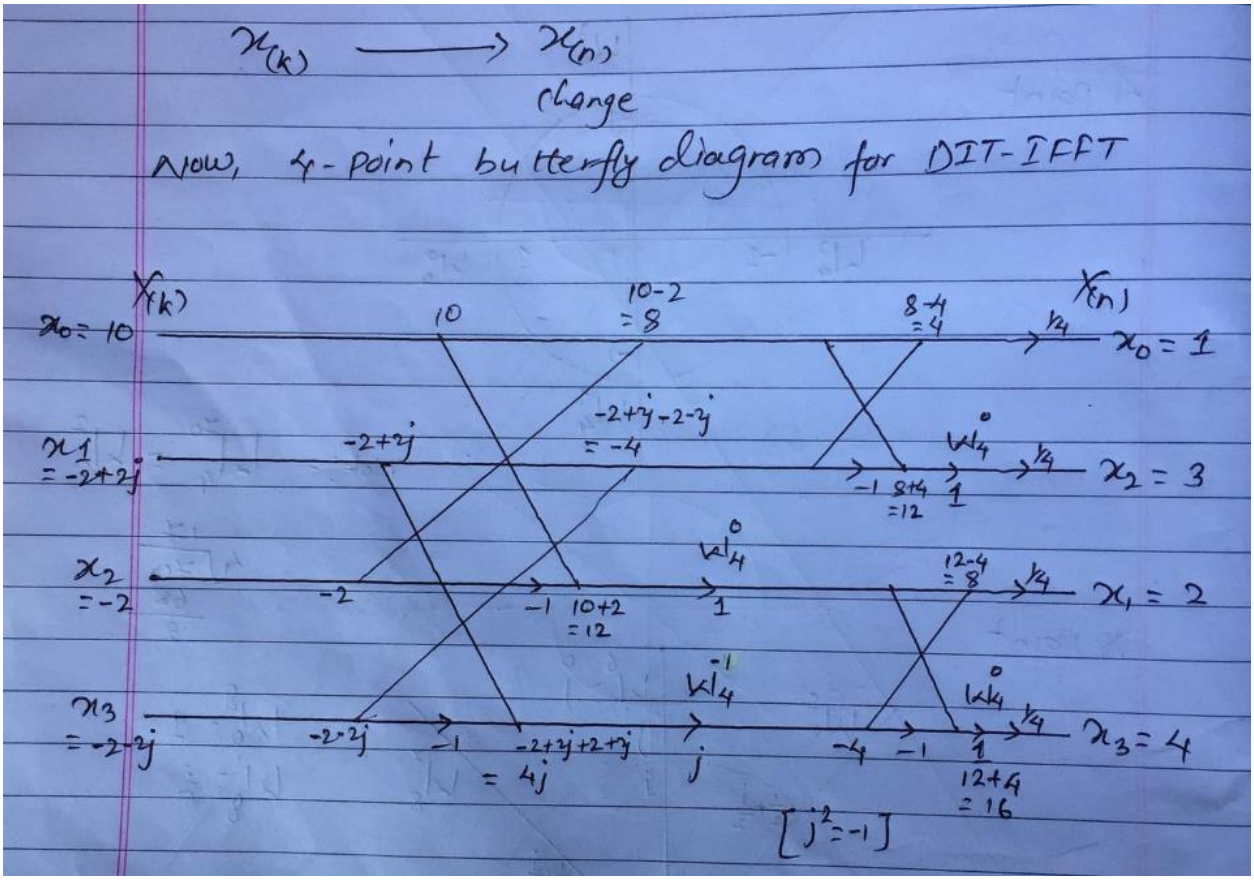
Note:

Complex Conjugate: *A conjugate of a complex number is another complex number which has the same real part as the original complex number and the imaginary part has the same magnitude but opposite sign. If we multiply a complex number with its conjugate, we get a real number.*

Example 2: Find the Inverse DFT (IDFT) of given function using DIT-IFFT

f(x) = {10, -2+2j, -2, -2-2j}.

Solution: Here, we have to convert frequency domain data to spatial domain.



Here, $X_0 = 1$, $X_1 = 2$, $X_2 = 3$, $X_3 = 4$

Hence, $X_n = \{1,2,3,4\}$ this is the IDFT of given function.

Verification of Result:

1. Sum of result should be equal to 1st term of given function.

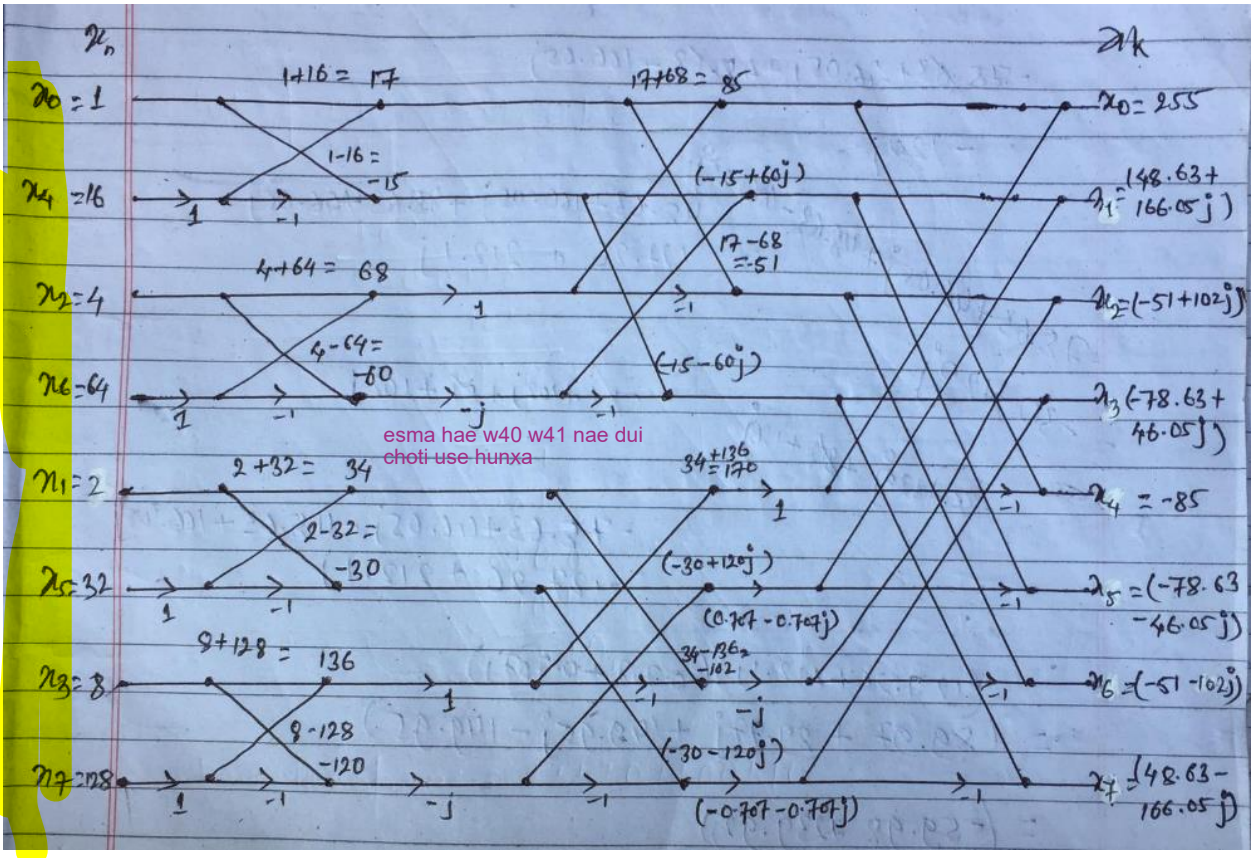
   Here, sum of result is 10, and 1st term of given function is also 10, verified.

Example 3: Find the DFT of given function using DIT-FFT

F(x) = {1, 2, 4, 8, 16, 32, 64, 128}.

Solution: here we have to transfer spatial domain data to frequency domain

Here, N=8,



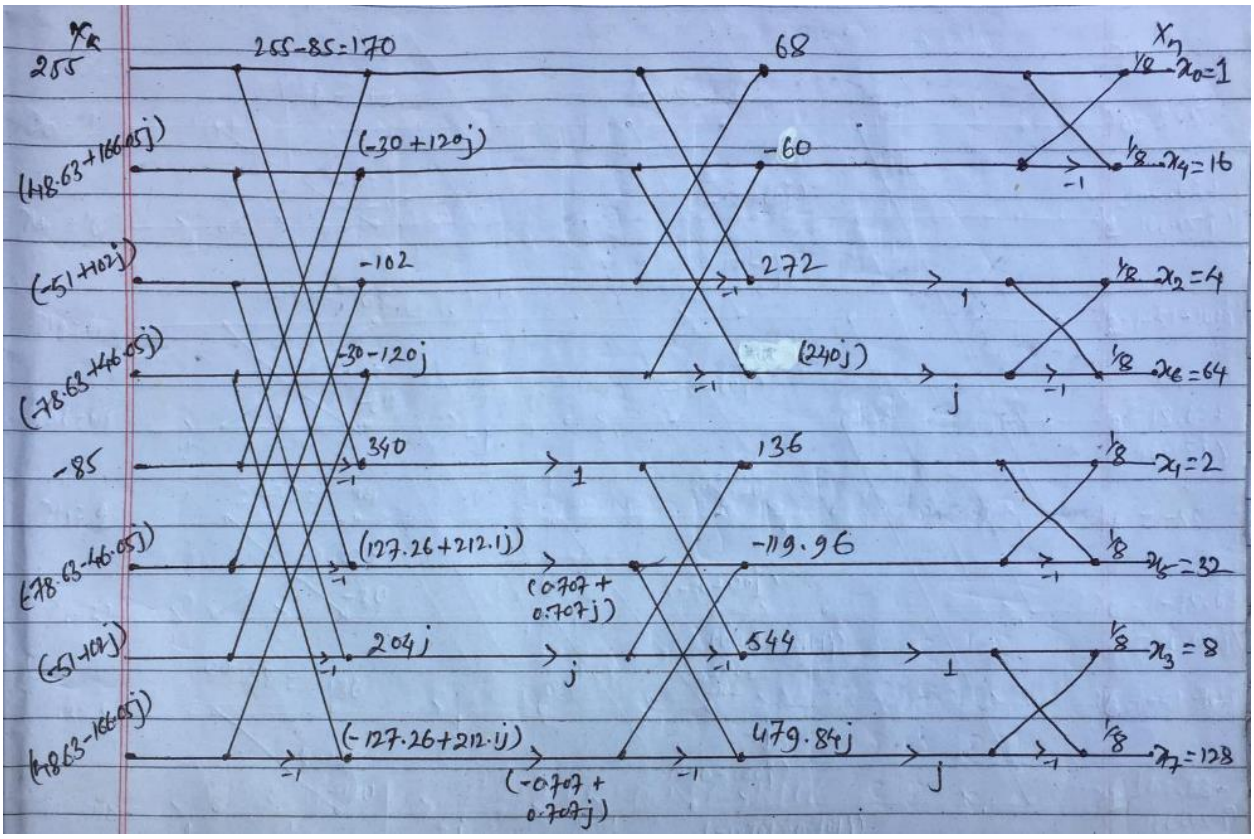
Hence, $X_{,k}$ = {255, 48.63+166.05j, -51+102j, -78.63+46.05j, -85, -78.63-46.05j, -51-102j, 148.63-166.05j} this is the DFT of given function.

Example 4: Find the IDFT of given function using DIT-IFFT

f(x) = {255, 48.63+166.05j, -51+102j, -78.63+46.05j, -85, -78.63-46.05j, -51-102j, 148.63-166.05j}

Solution: here, we have to transfer frequency domain data to spatial domain

Here, N=8



Here, $X_0 = 1$, $X_1 = 2$, $X_2 = 4$, $X_3 = 8$, $X_4 = 16$, $X_5 = 32$, $X_6 = 64$, $X_7 = 128$

Hence, $X_n = \{1, 2, 4, 8, 16, 32, 64, 128\}$ this is the IDFT of given function.

Discrete Cosine Transform:

Image Compression: Image is stored or transmitted with having pixel value. It can be compressed by reducing the value its every pixel contains. Image compression is basically of two types :

2. **Lossless compression:** In this type of compression, after recovering image is exactly become same as that was before applying compression techniques and so, its quality didn't gets reduced.
3. **Lossy compression:** In this type of compression, after recovering we can't get exactly as older data and that's why the quality of image gets significantly reduced. But this type of compression results in very high compression of image data and is very useful in transmitting image over network.

Discrete Cosine Transform is used in lossy image compression because it has very strong energy compaction, i.e., its large amount of information is stored in very low frequency component of a signal and rest other frequency having very small data which can be stored by using very less number of bits (usually, at most 2 or 3 bit).

To perform DCT Transformation on an image, first we have to fetch image file information (pixel value in term of integer having range 0 – 255) which we divides in block of 8 X 8 matrix and then we apply discrete cosine transform on that block of data.

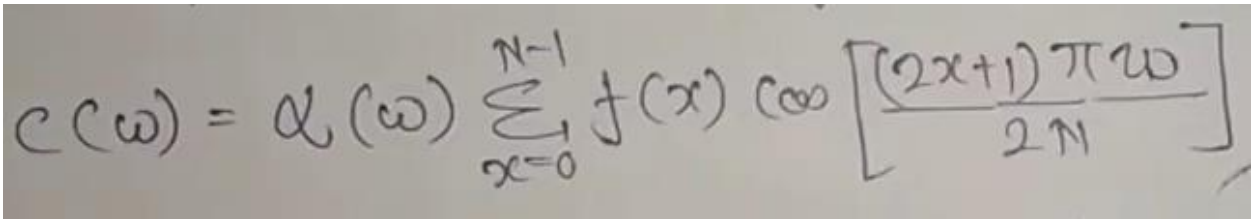
It is basically used for JPEG image compression.

Complexity of DCT:

Time Complexity: $O((nm)^2)$

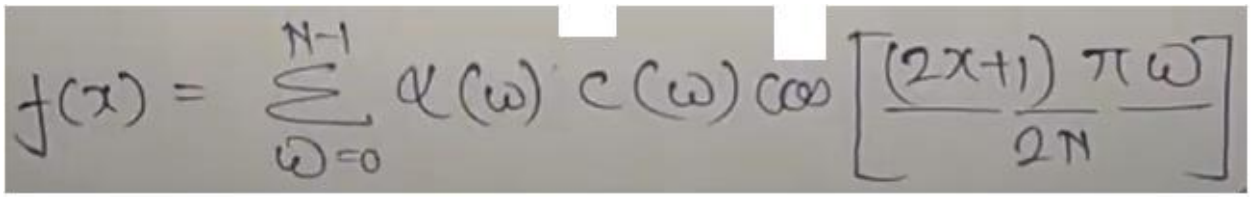
Auxiliary Space: O(nm)

The 1-D DCT is defined by the equation

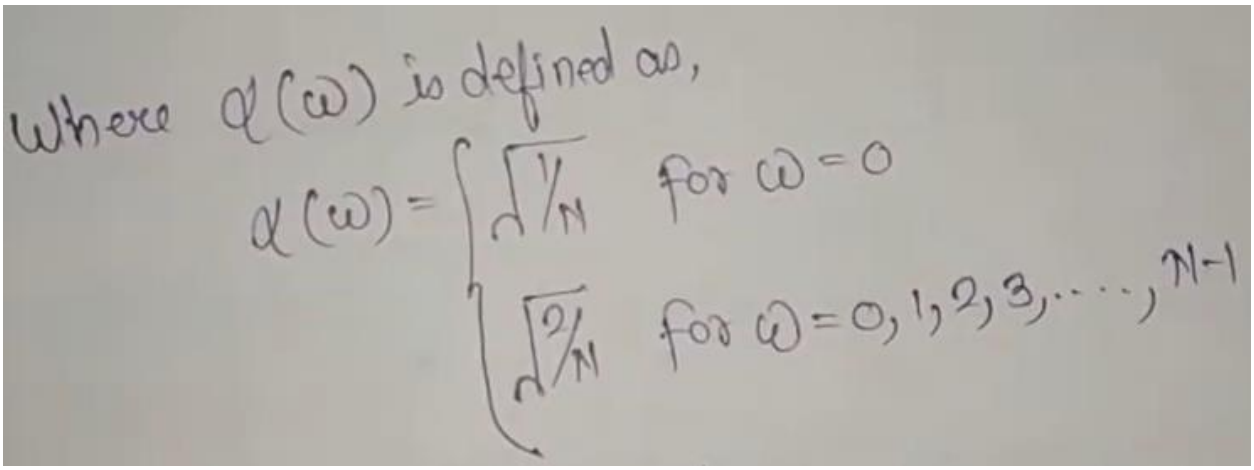


For ω = 0, 1, 2, 3, ……. N-1

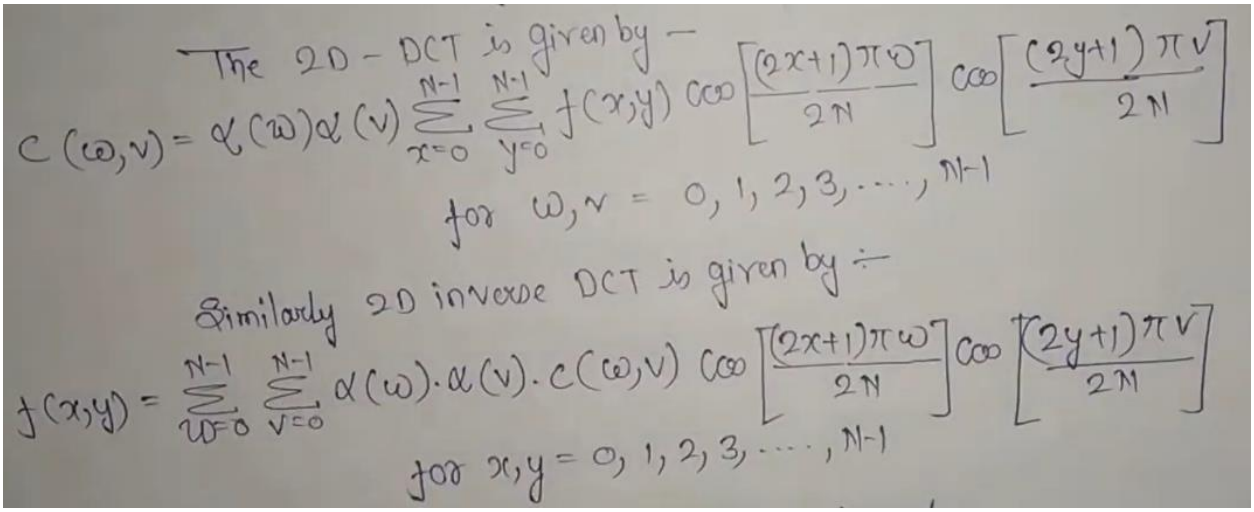
Similarly, the 1-D Inverse DCT is defined by the equation



For x = 0, 1, 2, 3, ……. N-1



The 2-D DCT is defined by the equation



We can perform DCT operation using matrix method.

For 1-D DCT:

F(k) = Mask*f(x)

For 2-D DCT:

$F(u,v) = Mask * f(x,y) * Mask^T$

The transpose mask or kernel matrix of DCT is given bellow.

4-point mask:

0.5	0.5	0.5	0.5
0.653	0.2705	-0.2705	-0.653
0.5	-0.5	-0.5	0.5
0.2705	-0.653	0.653	-0.2705

surumae third rw last - ani second rw third
ani second rw last

Example 1: Find the DCT of f(x) = {1, 2, 4, 7}.

Solution: here, given function is 1D

So, F(k) = Mask*f(x)

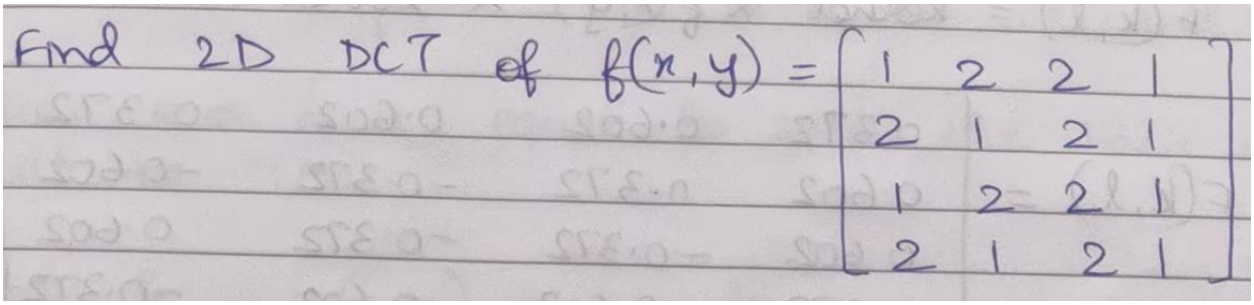
0.5	0.5	0.5	0.5	1
0.653	0.2705	-0.2705	-0.653	2
0.5	-0.5	-0.5	0.5	4
0.2705	-0.653	0.653	-0.2705	* 7

=

7
-4.459
1
-0.370

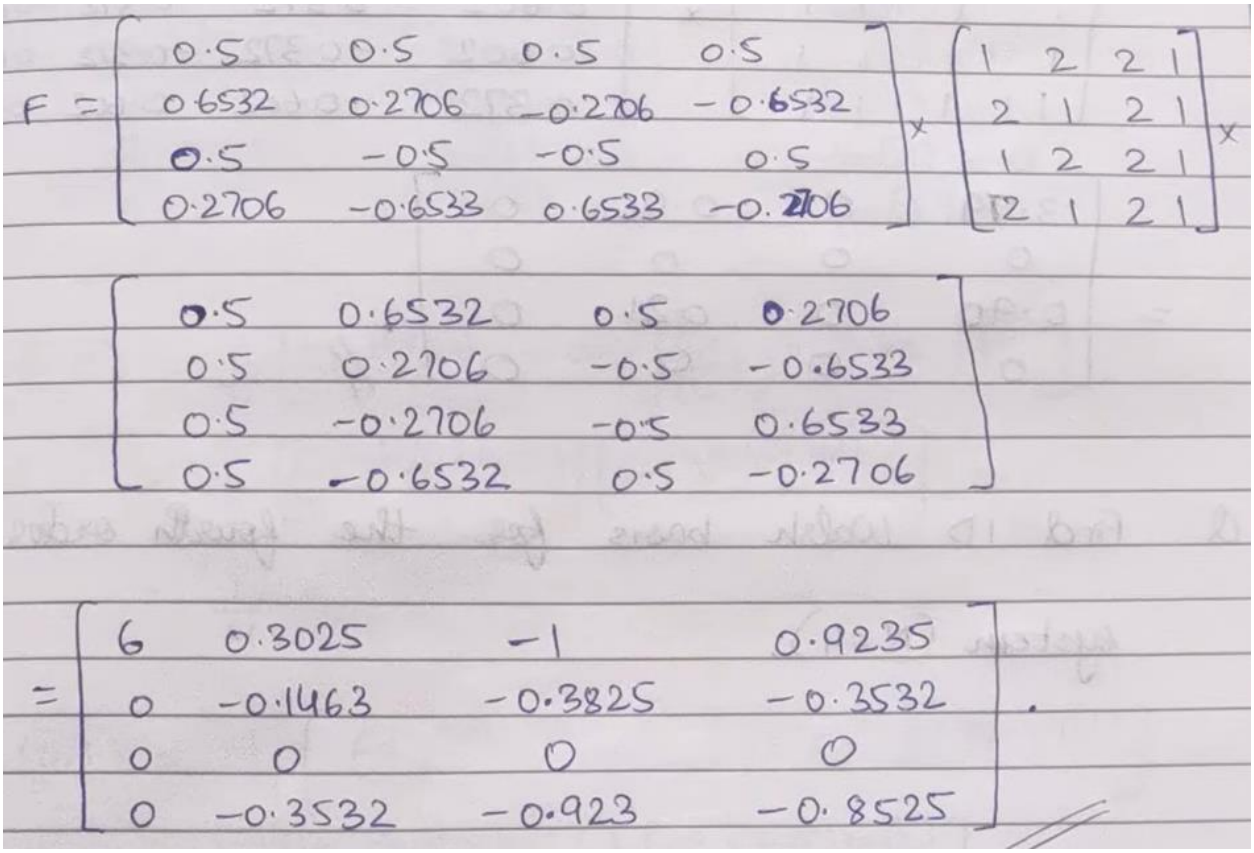
Hence, F(k) = {7, -4.459, 1, 0.370}

Example 2:



Solution:

$F(u,v) = Mask*f(x,y)*Mask^T$



Haar Transformation:

Haar transform is used in image compression. This is efficient to use in both lossy and lossless image compression techniques.

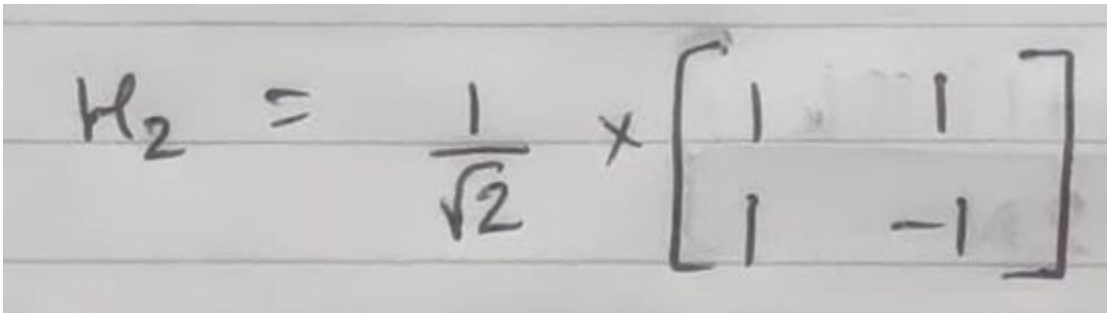
1-D haar transform:

F(k) = Mask*f(x)

2-D haar transform:

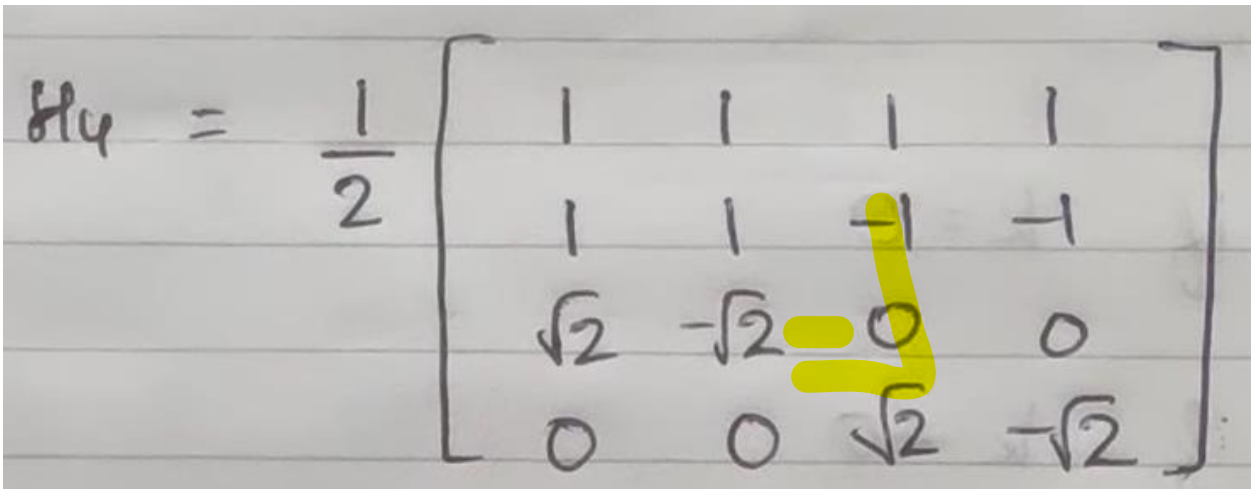
$F(k,l) = Mask*f(m,n)*Mask^T$

Mask for 2x2 matrix:

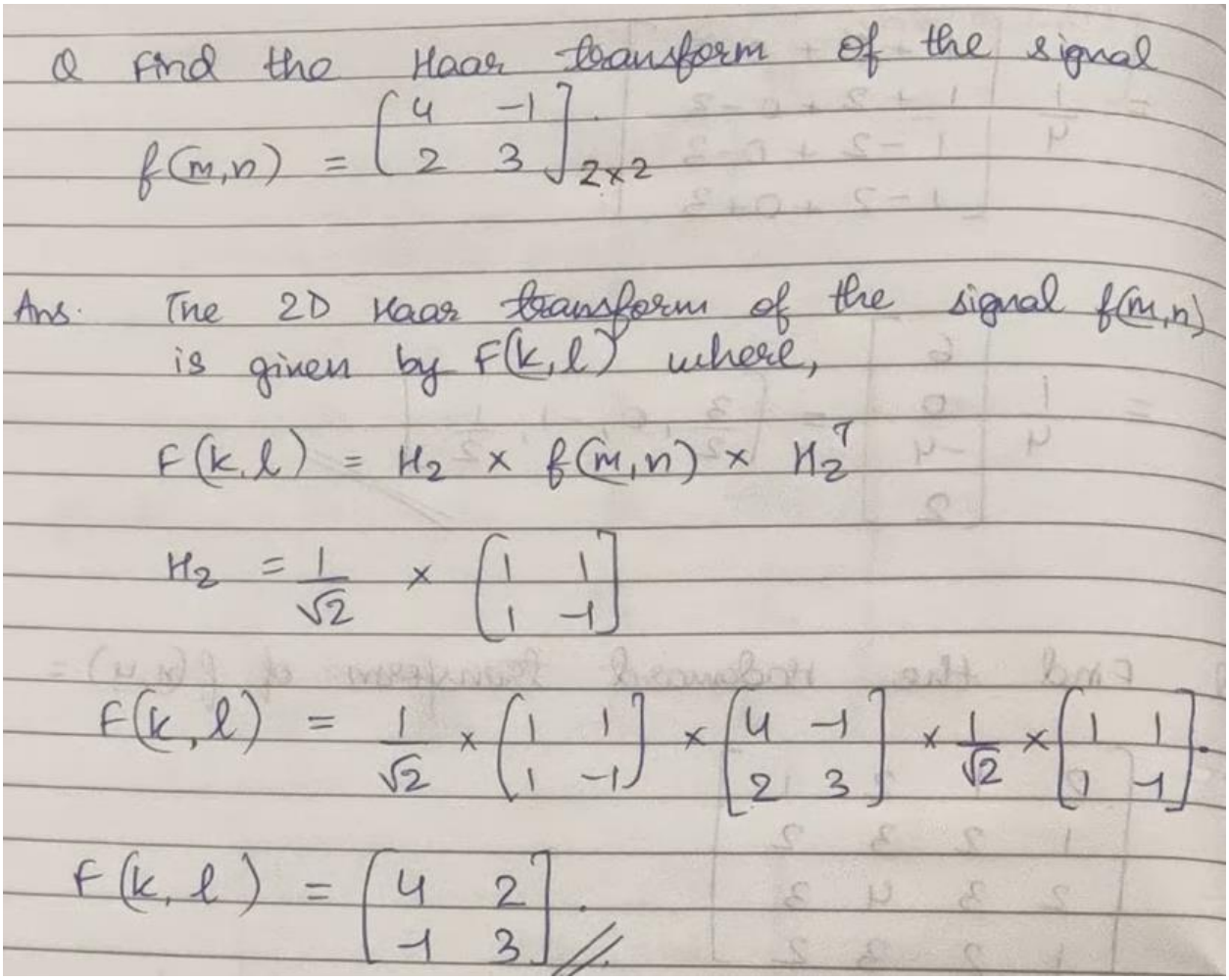


Mask for 4x4 matrix:

tyo last ko kuna
kuna wala part lai - xa sadhae



Example:



Hadamard Transformation: signals lai chai fxns ma dkehauna milne banaune hola

The Walsh-Hadamard transform is a non-sinusoidal, orthogonal transform widely used in signal and image processing. In this transform, the signal is decomposed into a set of basis functions.

The Walsh-Hamard has a wide application in the field of science and engineering. These applications include image processing, speech processing, signal and image compression, power spectrum analysis, spread spectrum analysis etc.

1-D hadamard transform:

F(k) = Mask*f(x)

2-D hadamard transform:

$F(k,l) = Mask*f(x,y)*Mask^T$

Hadamard mask for 2x2 matrix:

H_{2x2} =

1	1
1	-1

Hadamard mask for 4x4 matrix:

1	1	1	1
1	-1	1	-1
1	1	-1	-1
1	-1	-1	1

Example 1: find the DFT of given function using hadamard transform

F(x) = {1, 2, 0, 3}

Solution:

F(k) = Mask*f(x)

1	1	1	1
1	-1	1	-1
1	1	-1	-1
1	-1	-1	1

*

1
2
0
3

=

6
-4
0
2

Hence, F(k) = {6, -4, 0, 2}

End of Unit-3