MATHEMATICS FOR COMPUTING – 2

GROUP 14

GANESH SUNDHAR S
(CB.EN.U4AIE22017)

KARTHIGAI SELVAM
(CB.EN.U4AIE22025)

SHRUTHIKAA V
(CB.EN.U4AIE22047)

BHAVYA SAINATH U
(CB.EN.U4AIE22055)

IMAGE AND AUDIO CODECS USING DCT

DCT

Fourier Transform (FT) is a process that represents a aperiodic continuous time domain signal in its frequency domain.

But, Fourier Transform(FT) produces complex notations even for real functions. This is often seen as an disadvantage.

Discrete Cosine Transform (DCT) uses real cosine bases to calculate the DCT coefficients. This is efficient because representation and computation using real bases are easier.

IMAGE COMPRESSION

```
function ImgCompress

[file,path] = uigetfile('*.*','Select the image to compress');
name = strcat(path,file);
img = imread(name);
```

```
if(size(img,3) == 3)
    [R,G,B] = imsplit(img);

    R = dct2(double(R));

    R(abs(R) < 15) = 0;

    G = dct2(double(G));

    G(abs(G) < 15) = 0;

    B = dct2(double(B));

    B(abs(B) < 15) = 0;

    R = uint8(idct2(R));
    G = uint8(idct2(G));
    B = uint8(idct2(B));

    New_img = cat(3,R,G,B);
```

IMAGE COMPRESSION

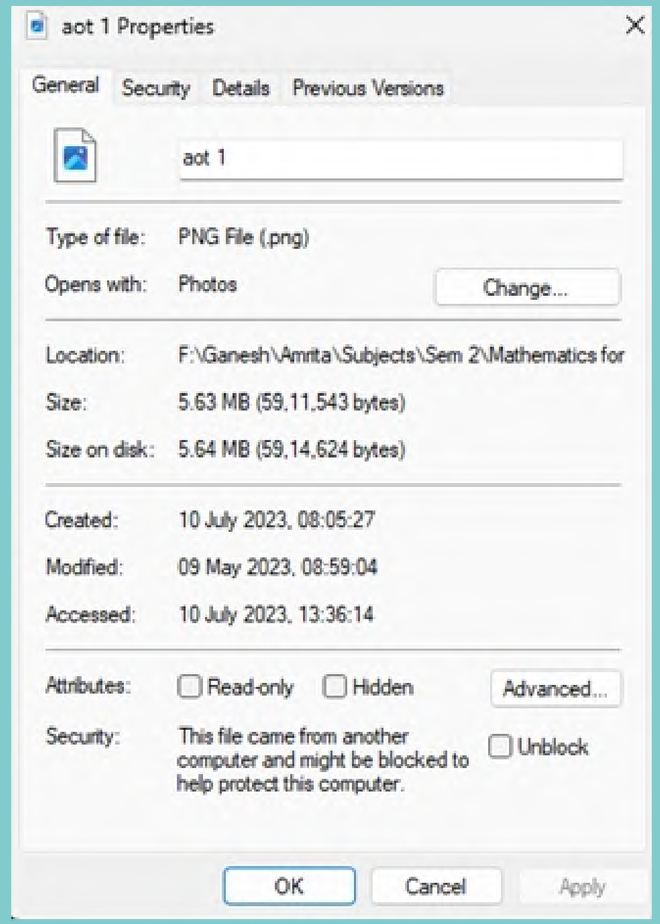
```
else
    Mat = dct2(double(img));
    Mat(abs(Mat) < 15) = 0;

    New_img = uint8(idct2(Mat));
end

imwrite(New_img,strcat(path,extractBefore(file,'.'),' red','.jpeg'));

end
```

AUDIO COMPRESSION

```
function AudCompress

[file,path] = uigetfile('*.*','Select the audio file to be compressed');
name = strcat(path,file);
[Aud,rate] = audioread(name);

Mat = dct2(Aud);

Mat(abs(Mat) < 0.02) = 0;

New_Aud = idct2(Mat);

audiowrite(strcat(path,extractBefore(file,'.'),' red','.mp3'),New_Aud,rate);

end
```

IMAGE – RESULT

CASE 1



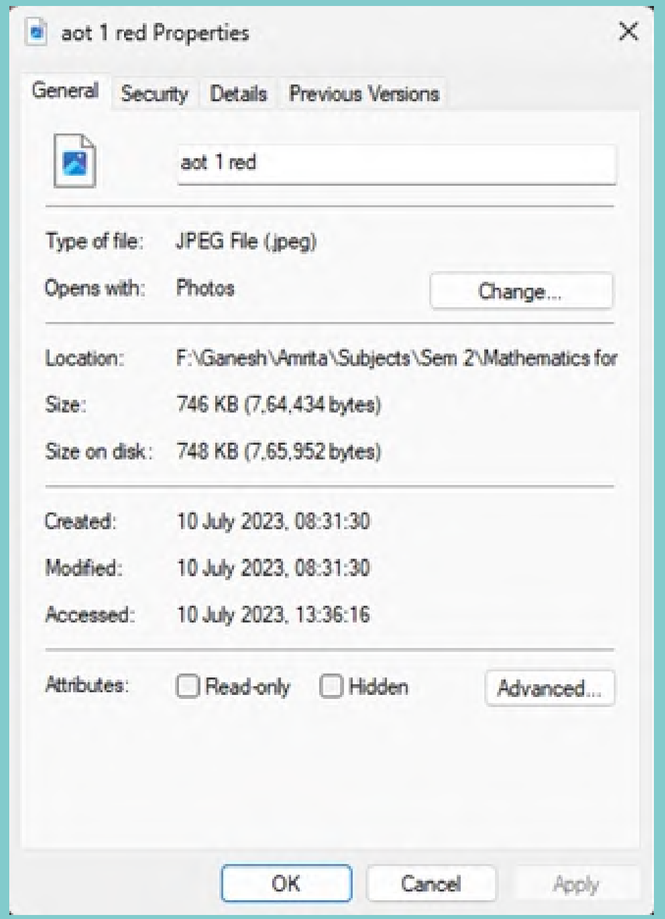
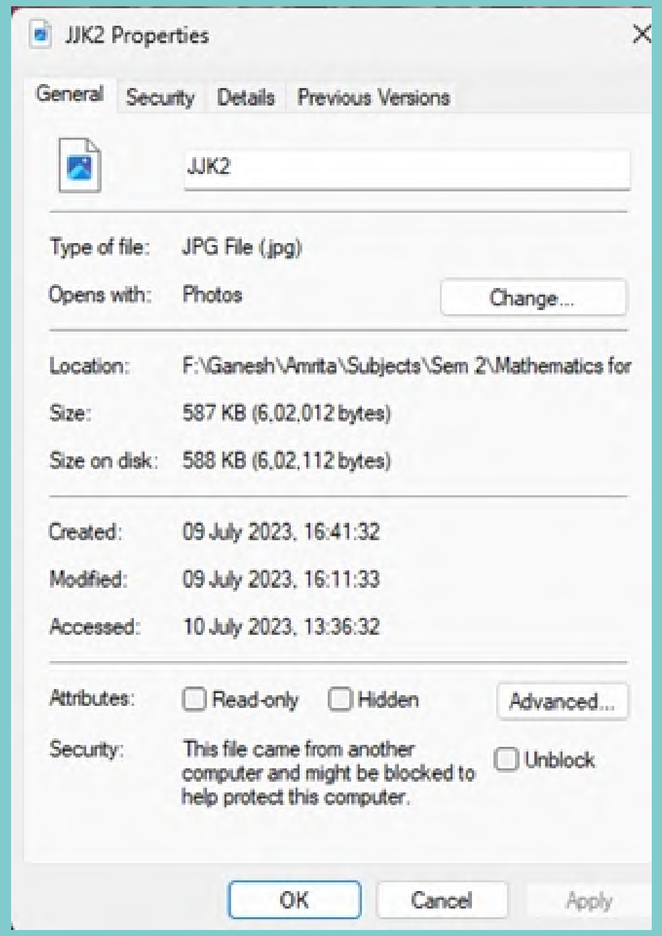
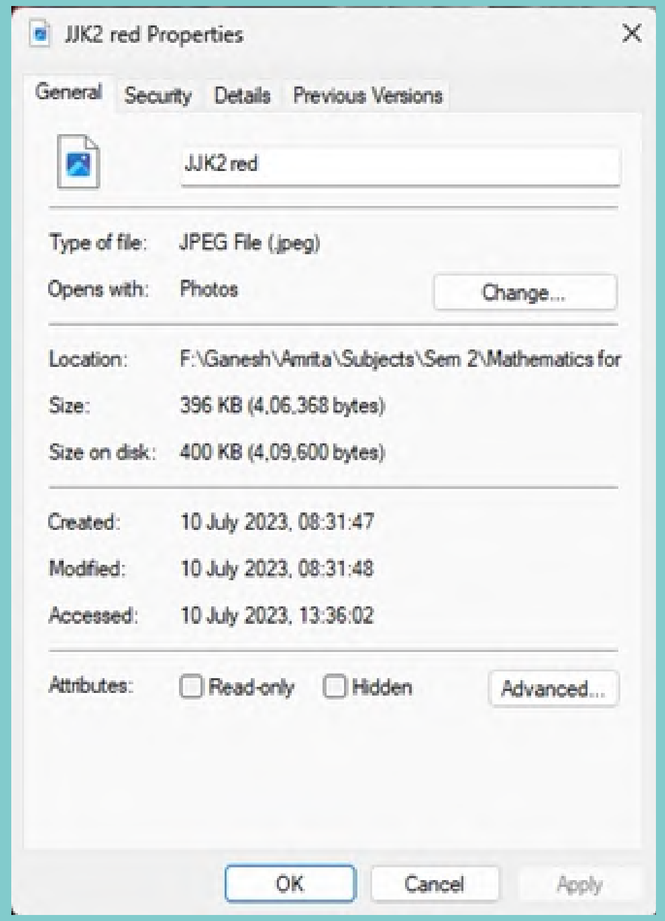


IMAGE – RESULT

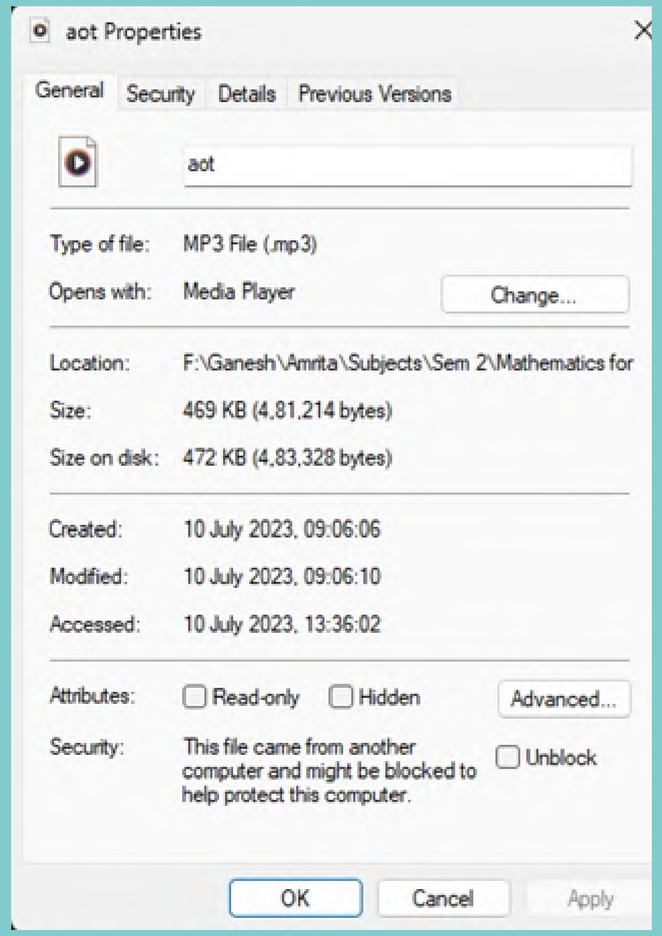
CASE 2

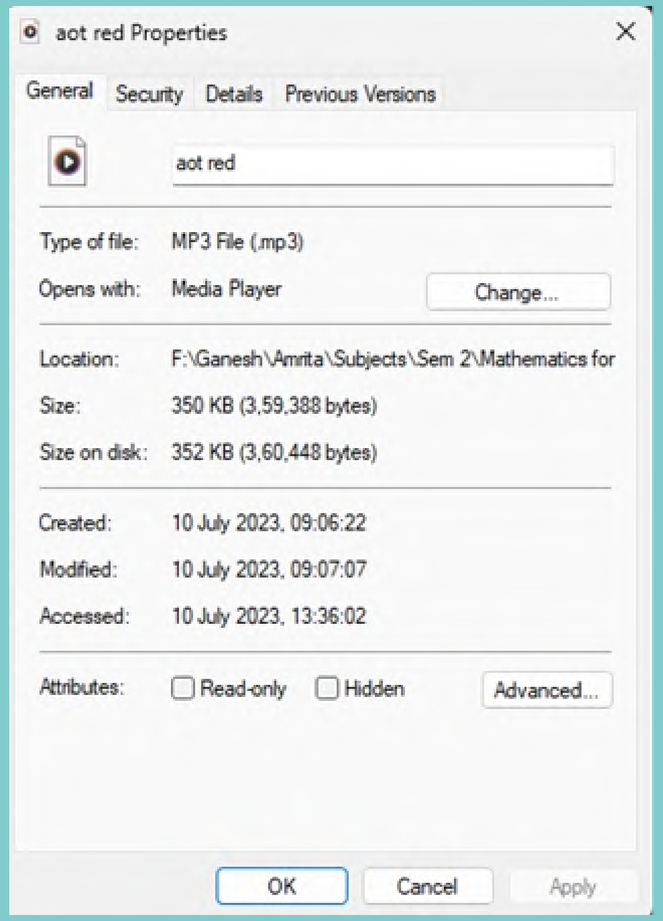




AUDIO – RESULT

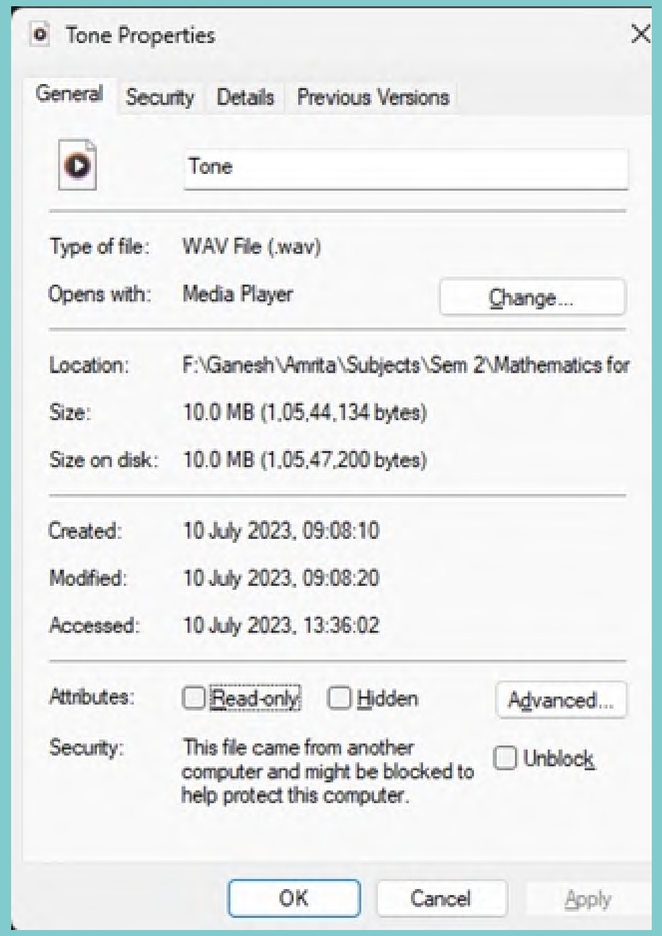
CASE 1

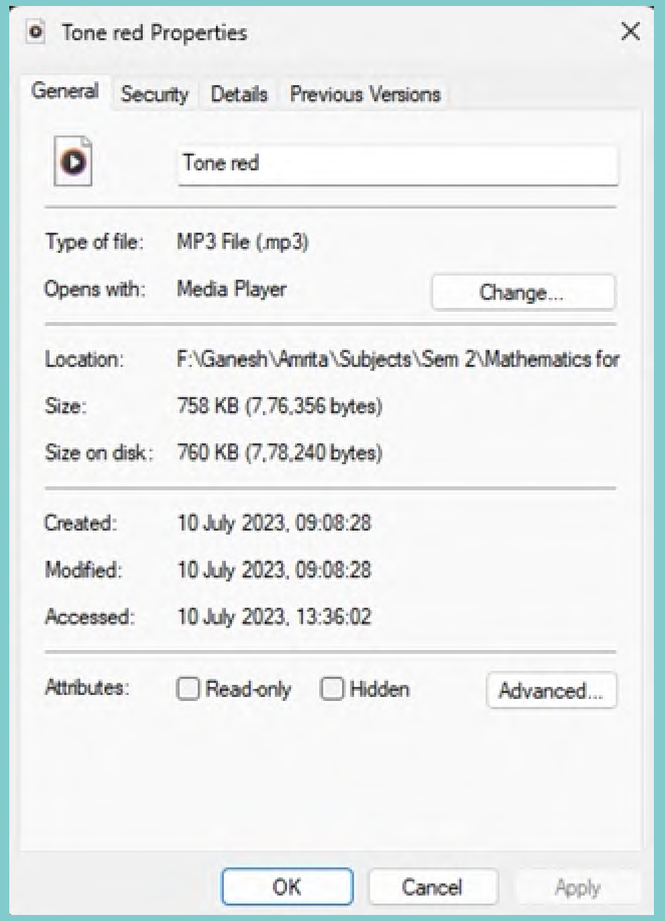




AUDIO – RESULT

CASE 2





BAND PASS FILTERING: CONVOLUION AND DECONVOLUTION (FFT)

THEORY:

As The Fourier Transform allows us to decompose a signal into its constituent frequencies, and analyze the spectral properties as the Fourier transforms of the original white noise and the filtered noise signals are computed.

By examining the spectrum characteristics of the filtered white noise and separating the IR from it, we can characterise the response of a system.

Fast Fourier transform (FFT) methods are used to implement the deconvolution process.

THEORY:

After being zeropadded to the same length, the FFTs of the filtered noise signal and the original white noise signal are calculated.

The frequency responsiveness of the system iscalculated by dividing the FFT of the filtered noise by the FFT of the original noise.
The estimated IR is obtained using the frequency response's inverse FFT.

A random number generator is used to create one second of white noise to start the experiment.

Finally, the extracted IR is compared with the original IR used in the bandpass filter design. The time-domain plots of both IRs are displayed for visual comparison.

This step allows for evaluating the accuracy of the deconvolution process and the effectiveness of the bandpass filter in capturing the system's behavior.

CODE:

```
clc; clear;close all;
Fs=44100;

% Generate 1 second of white noise
rand('state',sum(100 * clock));
noise = randn(1, Fs);
noise = noise / max(abs(noise));

% BAND PASS FILTER
fc=1000;                    % Central Frequency
f1=fc/2^0.5;
f2=fc*2^0.5;
w1=2*pi*f1/Fs;
w2=2*pi*f2/Fs;
wc=2*pi*fc/Fs;
```

```
ncof=200;                   % number of coeficients,
if rem(ncof, 2)             % must be even
    ncof=ncof+1;
end
n=0:ncof;
M=length(n)-1;              % filter order
% Filter
h=sin(w2*(n-(M/2))./((n-(M/2))*pi)-sin(w1*(n-(M/2)))./((n-(M/2))*pi);
h(M/2+1)=(w2-w1)/pi;

% Filter white noise
filternoise = fftconv(noise,h);
% Extract IR from filtered noise
hnew = fftdec(filternoise,noise);
```

CODE:

```
% White noise spectrum
nf=4096; %number of point in DTFT
Yw = fft(noise,nf);
fw = Fs/2*linspace(0,1,nf/2+1);
% Filtered white noise spectrum
Yf = fft(filternoise,nf);
```

```
figure
plot(fw, abs(Yw(1:nf/2+1)));
title('White noise spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([500 10000]);

figure
plot(fw, abs(Yf(1:nf/2+1)));
title('Filtered white noise spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([100 10000]);

figure
plot(0:length(h)-1, h);
title('Original IR - BandPass Filter');
xlabel('Samples');
ylabel('Amplitude');
xlim([0 length(h)]);

figure
plot(0:length(hnew)-1, hnew);
title('Extracted IR');
xlabel('Samples');
ylabel('Amplitude');
xlim([0 length(h)]);
```

CODE:

```
function [ y ] = fftconv( x, h )

nx = length(x);
nh = length(h);
nfft = 2^nextpow2(nx+nh-1);
xzp = [x, zeros(1,nfft-nx)];
hzp = [h, zeros(1,nfft-nh)];
X = fft(xzp);
H = fft(hzp);

Y = H .* X;
y = real(ifft(Y));
end
```

```
function [ h ] = fftdec( out,in )
no = length(out);
ni = length(in);
nfft = 2^nextpow2(no+ni-1);
ozp = [out, zeros(1,nfft-no)];
izp = [in, zeros(1,nfft-ni)];
O = fft(ozp);
I = fft(izp);

Y = O ./ I;
h = real(ifft(Y));
end
```

signal filtering in fourier transform

signal filtering :

signal filtering is the type of process which remove unnecessary features from a signal. Signal filtering is to reduce high frequencies noise associated with measurement temperature and reduce the back ground noise. signal filtering tells the relative amplitude of frequencies present in the signal.

in the code part we are giving input signals consists of the sum of two sin waves with the frequencies of 50Hz and 120Hz and amplitude of 0.7 and 1.2 to the first and second sin wave. after that that the signal will plot the time domain.

```
%Sampling frequency (Hz)
Fs = 1000;
% Sampling period (s)
T = 1/Fs;
% Signal length
L = 1000;
% Time vector
t = (0:L-1)*T;
% Create input signal (we are generating two sin waves and adding together to form the input signal)
%f1 and f2 represents the frequency of first and second sin waves in (Hz)
f1 = 50;
f2 = 120;
% A1 and A2 represents the amplitude of first and second sin waves
A1 = 0.7;
A2 = 1.2;
x = A1*sin(2*pi*f1*t) + A2*sin(2*pi*f2*t);
% here we are plotting the input signal
subplot(2,1,1);
plot(t, x);
title('Input Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

in This code calculates the Fourier series coefficients of an input signal using the Fast Fourier Transform (FFT) algorithm. The code first computes the length of the input signal and stores it in the variable `N`. Then, it applies the FFT to the signal by dividing by `N` to normalize the coefficients.

a frequency vector `f` is created using the sampling frequency (`Fs`) and the number of coefficients (`N`). This vector represents the range of frequencies for which the Fourier series coefficients were calculated. It then plots the amplitude spectrum of the signal.

```
% Computing Fourier series coefficients of the input signal
N = length(x);
% by applying the 'fft' function we will obtain the frequency domain
%representaton of the signal
X = fft(x)/N;
f = Fs*(0:(N/2))/N;

% here we are plotting the amplitude spectrum of the input signal
subplot(2,1,2);
stem(f, 2*abs(X(1:N/2+1)));
title('Amplitude Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

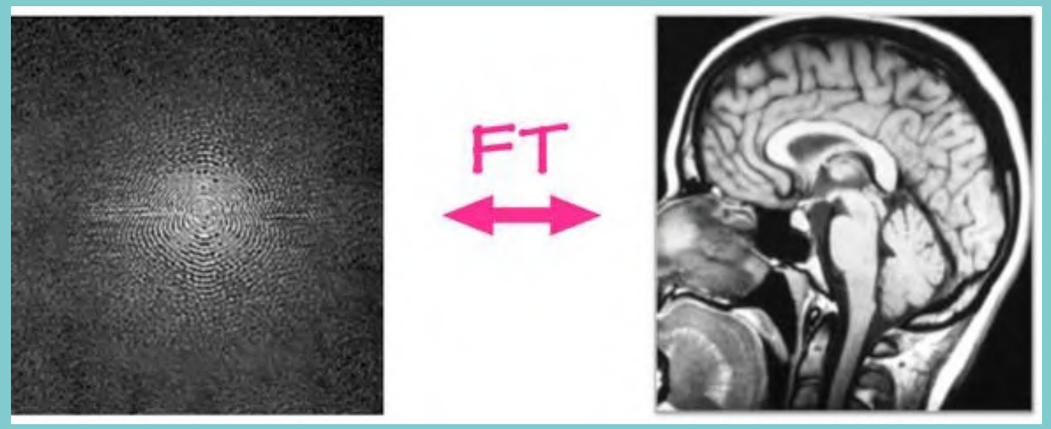
now the input signal is filtered by removing frequencies below the cutoff frequency. The filtered signal is then reconstructed using the inverse Fourier transform. Frequencies below the cutoff frequency are set to zero in X_filtered. The filtered signal is reconstructed by applying the inverse Fourier transform to X_filtered using the ifft function.the filtered signal is plotted in the time domain using the `plot` function.

```
% Filter the signal by removing unwanted frequencies
% Remove the frequencies below 80 Hz
cutoff_frequency = 80;
X_filtered = X;
X_filtered(f < cutoff_frequency) = 0;

% Reconstruct the filtered signal using inverse Fourier transform
x_filtered = ifft(X_filtered)*N;

% Plot the filtered signal
figure;
plot(t, x_filtered);
title('Filtered Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

MEDICAL IMAGING USING FAST FOURIER TRANSFORM

K-Space:

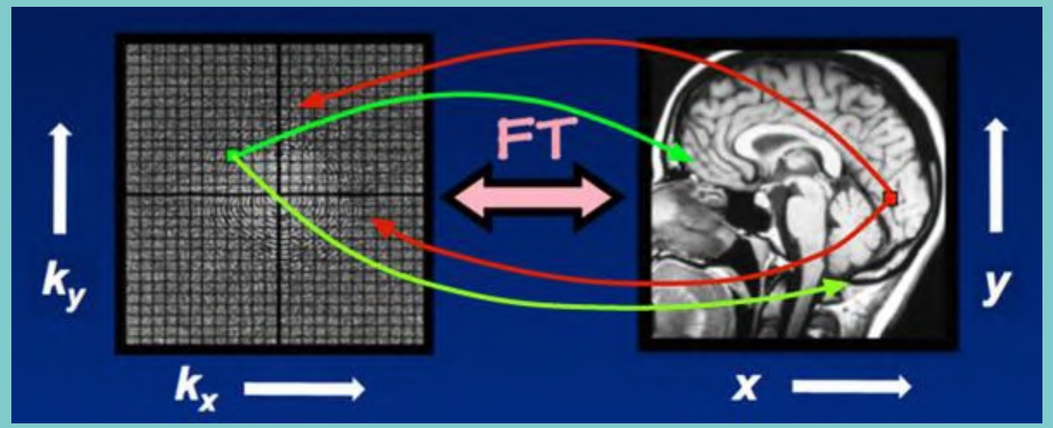
The popular graphic representation of k-space as a 'galaxy' adds to the mystery. Each 'star' in k-space is simply a data point extracted from the MR signal. The brightness of each star symbolizes its unique spatial frequency's relative contribution to the final image.



Although the k-space “galaxy” and MRI image appear quite different, they contain identical information about the scanned object.The two representation may be converte to one another using an advanced mathematical procedure the Fourier Transform.

The cells of k-space are often represented on a rectangular grid with the primary axes kx and ky. The kx and ky axes of k-space correspond to the image's horizontal (x-) and vertical (y-) axes. The k-axes, on the other hand, represent spatial frequencies rather than places in the x and y directions. Individual points (kx,ky) in k-space do not correlate to individual pixels (x,y) in the image one-to-one.

Every pixel in the final image is represented by a k-space point, which comprises spatial frequency and phase information. In contrast, each pixel in the image corresponds to every point in k-space. As a result, the k-space representation of the MR picture is similar to the diffraction patterns produced by x-ray crystallography, optics, or holography.



Fast Fourier Transform (FFT):

The FFT is a fast algorithm used to compute the Discrete Fourier Transform (DFT) of a sequence or signal. The DFT represents a discrete-time signal as a sum of complex sinusoidal functions, which allows us to analyze the signal's frequency content.

$X[k] = \Sigma [x[n] * e^{(-j * 2\pi * k * n / N)}]$

X[k]: This represents the frequency domain representation of the signal. It denotes the complex amplitude at frequency index k.

x[n]: This represents the discrete-time sequence or signal in the time domain. It is the input sequence to the FFT.

exp(-j * 2π * k * n / N): This term represents a complex sinusoidal function with a frequency index k. It is raised to the power of -j (the imaginary unit multiplied by π), and multiplied by 2π and k and n divided by N. This term is used to decompose the input signal into its frequency components.

where k is the frequency index (ranging from 0 to N-1) and j represents the imaginary unit. The FFT algorithm breaks down the computation of the DFT into smaller sub-problems, reducing the time complexity from O(N^2) to O(N log N). This makes it much more efficient for practical applications.

The key idea behind the FFT is the concept of 'decimation in time' or 'divide and conquer.' It recursively splits the sequence into even and odd indexed elements and applies a butterfly operation to combine the results. This process is repeated until the sequence is reduced to its individual components, resulting in the final DFT.

Inverse Fast Fourier Transform (IFFT):

The IFFT is the inverse operation of the FFT. It takes a frequency domain representation (obtained through the FFT) and reconstructs the original signal in the time domain.
x[n] = (1/N) * Σ [X[k] * e^(j * 2π * k * n / N)]

x[n]: This represents the reconstructed time domain signal obtained through the IFFT. It is the output signal.

X[k]: This represents the frequency domain representation of the signal. It denotes the complex amplitude at frequency index k.

exp(j * 2π * k * n / N): This term represents a complex sinusoidal function with a frequency index k. It is raised to the power of j (the imaginary unit multiplied by π), and multiplied by 2π and k and n divided by N. This term is used to reconstruct the time domain signal from its frequency components.

Similar to the FFT, the IFFT algorithm employs the 'decimation in time' approach to efficiently compute the inverse transformation. It reverses the order of operations compared to the FFT, enabling the reconstruction of the original signal.

By applying the IFFT to the frequency domain representation obtained through the FFT, you can convert back from the frequency domain to the time domain, allowing further analysis or processing in the spatial domain.

Both the FFT and IFFT are powerful tools in signal processing and have extensive applications in various fields, including medical imaging.

CODE

```
i = imread('MRI.jpeg');
subplot(1,3,1)
imshow(i);
title('Original Image');

grayi = rgb2gray(i);
fftimage = fftshift(fft2(grayi));
subplot(1,3,2)
fftshow = mat2gray(log(1+abs(fftimage)));
imshow(fftshow)
title('FFT of Image');

inverse = abs(ifft2(fftimage));
inverse = mat2gray(inverse);
subplot(1,3,3)
imshow(inverse);
title('IFFT of Image');
```

Reads image using 'imread'

converting the image to gray scale using 'rgb2gray' function.

using fftshift shifting the zero frequency component to center to create a collection of points (star).

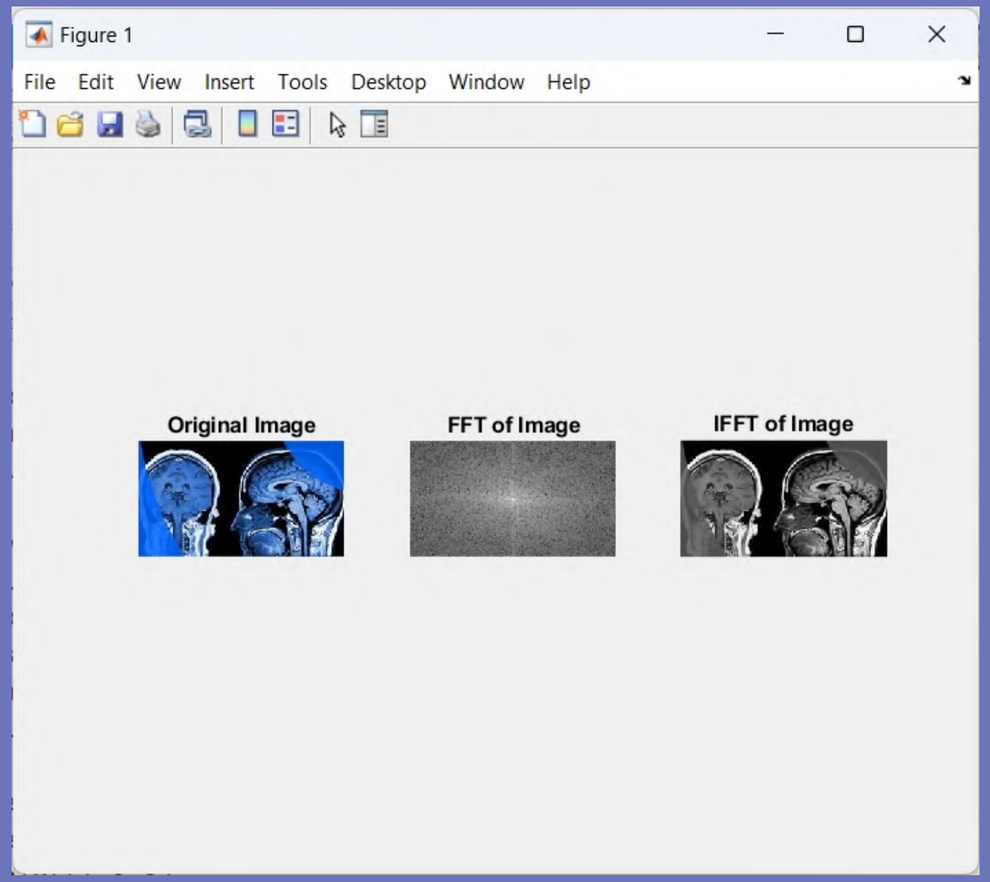
Taking log 1 and absolute of fftimage to improve visiblity and to get absolute value

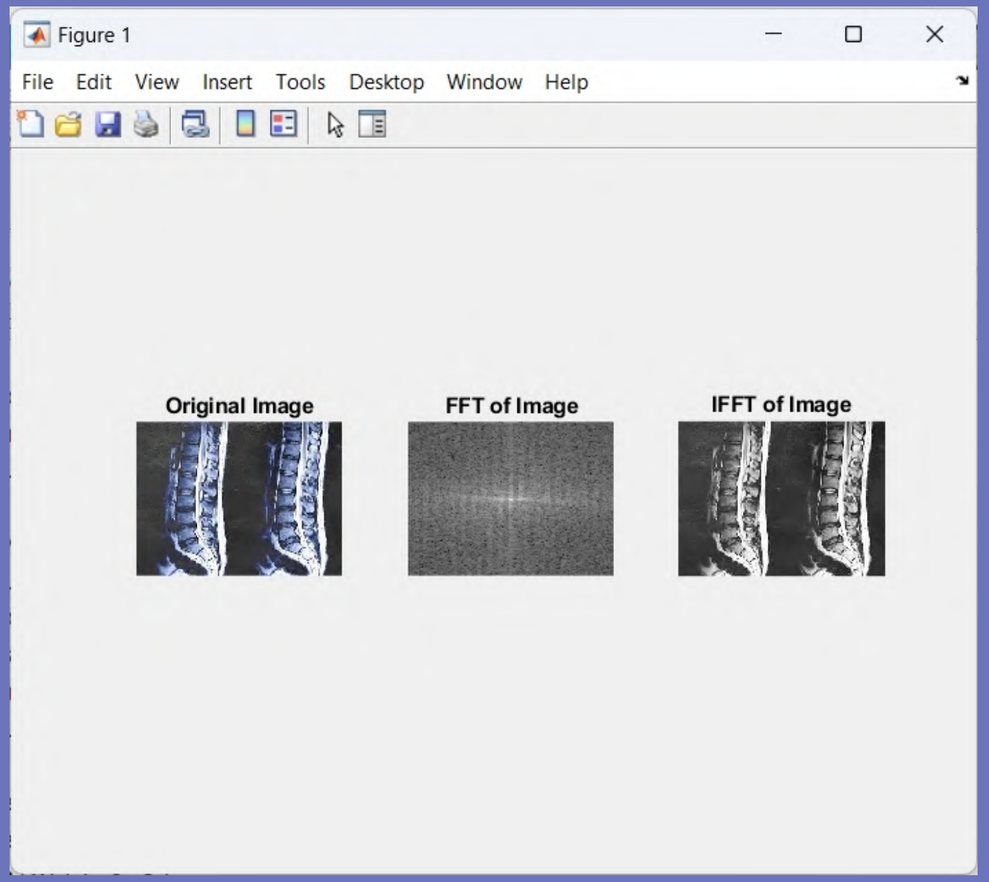
The mat2gray function scales the values of the resulting matrix to the range [0, 1], and the resulting matrix is stored in fftshow

Then using ifft2 the inverse operation starts and absolute value of the ifft image is taken

The mat2gray function is used to scale the values of inverse to the range [0, 1].

OUTPUT





THANK YOU