**COMMUNICATION SYSTEMS LAB REPORT**

****

**Project Title:**

**Submitted By**

**Name (Student ID)**

**Name (Student ID)**

**Submitted To**

**Engr. Omer Mujahid**

**Department of Electrical Engineering**

**CECOS University of IT & Emerging Sciences**

**Date:**

Contents

[List of Figure 3](#_Toc46616296)

[Introduction 4](#_Toc46616297)

[Concept and Frequency domain 4](#_Toc46616298)

[Importance of frequency domain 4](#_Toc46616299)

[Application 4](#_Toc46616300)

[Methodology 4](#_Toc46616301)

[Implementation 5](#_Toc46616302)

[Code 7](#_Toc46616303)

[Result Analysis 7](#_Toc46616304)

[Conclusion 9](#_Toc46616305)

# List of Figure

[Figure 1: Methodology 5](#_Toc46616308)

[Figure 2: FFT in 1D 5](#_Toc46616309)

[Figure 3: formula of fft in 2D 6](#_Toc46616310)

[Figure 4: Original Image 7](#_Toc46616311)

[Figure 5: Magnitude response 8](#_Toc46616312)

[Figure 6: Phase Response 8](#_Toc46616313)

# Introduction

The Fast Fourier Transform is an algorithm optimization of the DFT (Discrete Fourier Transform). The “discrete” part just means that it’s an adaptation of the Fourier Transform, a continuous process for the analog world, to make it suitable for the sampled digital world. Most of the discussion here addresses the Fourier Transform and its adaptation to the DFT. When it’s time for you to implement the transform in a program, you’ll use the FFT for efficiency. The results of the FFT are the same as with the DFT; the only difference is that the algorithm is optimized to remove redundant calculations. In general, the FFT can make these optimizations when the number of samples to be transformed is an exact power of two, for which it can eliminate many unnecessary operations.

## Concept and Frequency domain

A signal can be viewed from two different standpoints:

1. The frequency domain
2. The time domain

In astronomy the frequency domain is perhaps the most familiar because a spectrometer, e.g. a prism or a diffraction grating, splits light into its component colour or frequencies and permits us to record its spectral content. This is like the trace on a spectrum analyser, where the horizontal deflection is the frequency variable and the vertical deflection is the signals amplitude at that frequency.

In the lab we are also familiar with the time domain. This is like the trace on an oscilloscope where the vertical deflection is the signals amplitude, and the horizontal deflection is the time variable.

Any signal can be fully described in either of these domains. We can go between the two by using a tool called the Fourier transform

## Importance of frequency domain

Depending on what we want to do with the signal, one domain tends to be more useful than the other, so rather than getting tied up in mathematics with a time domain signal we might convert it to the frequency domain where the mathematics are simpler.

## Application

The Fourier transform has many applications, in fact any field of physical science that uses sinusoidal signals, such as engineering, physics, applied mathematics, and chemistry, will make use of Fourier series and Fourier transforms. It would be impossible to give examples of all the areas where the Fourier transform is involved, but here are some examples from physics, engineering, and signal processing.

* Communications
* Astronomy
* Geology
* Optics

# Methodology

Flow diagram of the FFT. This is based on three steps:

* Decompose an N point time domain signal into N signals each containing a single point.
* Find the spectrum of each of the N point signals
* Synthesize the N frequency spectra into a signal frequency spectrum

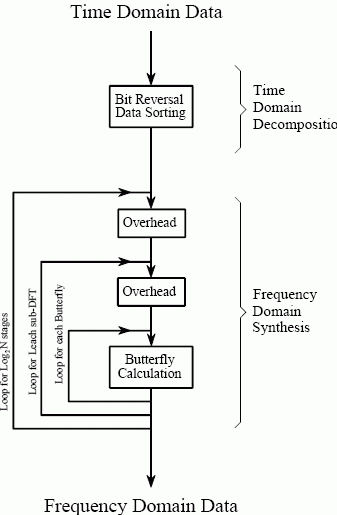


Figure : Methodology

# Implementation

FFT in one dimension

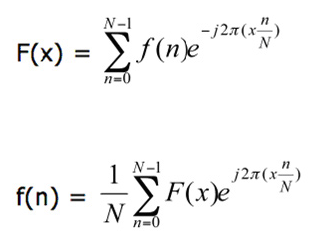


Figure : FFT in 1D

FFT in 2D case is;

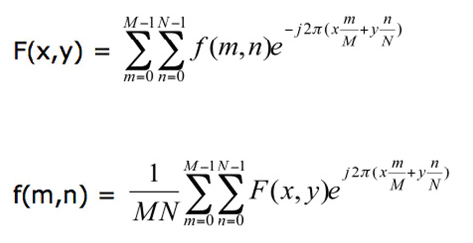


Figure : formula of fft in 2D

Because the FFT is a divide-and-conquer algorithm, the various steps can be implemented in multiple passes in a shader by rendering the result of each pass to a texture. These steps are called butterflies, because of the shape of the data-flow diagram.

The FFT algorithm recursively breaks down a DFT of composite size N = n1n2 into n1 smaller transforms of size n2.

These smaller DFTs are then combined with size n1 butterflies, which themselves are DFTs of size n1 (performed n2 times on corresponding outputs of the sub-transforms) pre-multiplied by roots of unity (known as twiddle factors)5.

The application employs four 2D textures: two for the ping-pong operations, one for the source data (for each pass), and one for holding the indices and weights for performing the butterfly steps.

The textures used for ping-ponging are marked as either a Render Target texture or a source depending on whether they are used as destinations or sources in the current pass.

This enables the shader to use the output of the previous pass as input for the current pass. Here are the steps to implement the FFT algorithm.

1. Compute the indices and weights for performing the butterfly operations.
2. Compute the log2(Width) horizontal butterflies.
3. Compute the log2(Height) vertical butterflies.

If the height and width of the image are equal, only one texture can be used for the butterfly values, for both the horizontal and vertical passes. Also note that in the vertical butterfly pass the input is the result of the horizontal butterfly pass.

In each butterfly pass the current pixel is combined with another pixel using a complex multiply and add operation and written to the current location. In other words, if the current pixel is a, then

a = a + wb

where w is a complex number representing the weight and b is some other pixel. Each Texel of the butterfly texture contains the locations of a, b and the value of w and passed to the shader by the application.

Note that for simplicity, only gray scale images are considered in the current implementation. However, extending the algorithm to multiple color channels is straightforward and only requires more textures for additional channels.

# Code

clc

clear all

close all

% read image

image=imread('cameraman.tif');

% perform fft

% use fft2 for 2 diminsion

image\_fft=fft2(image);

% show image

figure

imshow(image)

title('original image')

% plot Magnitude responce

figure

plot(abs(image\_fft))

title('Magnitude Responce')

% plot phase responce

figure

plot(angle(image\_fft))

title('Phase Responce')

# Result Analysis



Figure : Original Image

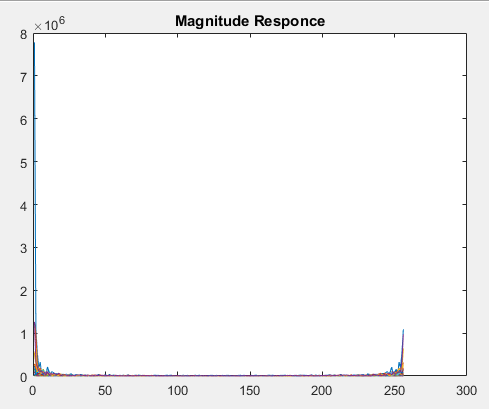


Figure : Magnitude response

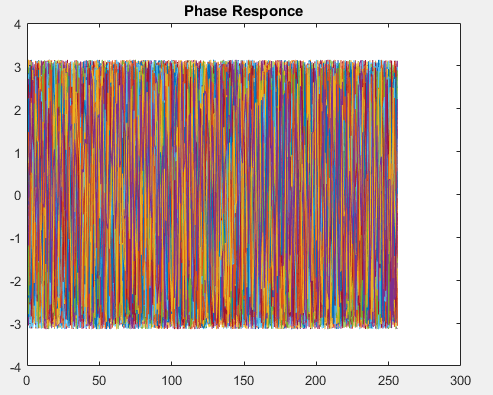


Figure : Phase Response

# Conclusion

the implementation of FFT and its inverse for transforming a 2D image from the spatial domain to the frequency domain and back. The advantage of representing an image in the frequency space is that performing some operations on the frequencies is much more efficient than doing the same in the image space. Many of the convolutions are just multiplications in the frequency domain (the computational cost in the image space is O(N2) vs. O (N log(N)) in the Fourier space for N points). This enables efficient implementations of very large convolutions in image processing and other algorithms/operations in many fields.