# SDLC Documentation – FFT Calculator

## Requirements

Implement an N-point Fast Fourier Transform Calculator in C programming language.

### State of the Art

It is a computer algorithm which changes the time domain signal into the frequency domain by quickly performing a discrete Fourier transform (DFT) on the signal.

The common FFT algorithms are Cooley–Tukey FFT algorithm, Prime-factor FFT algorithm, Bruun's FFT algorithm, Rader's FFT algorithm, Bluestein's FFT algorithm, Hexagonal fast Fourier transform etc.

There are myriad uses of FFT namely in engineering, music, science, and mathematics. It is used in digital recording, sampling, additive synthesis and pitch correction software. It is used in fast large-integer and polynomial multiplication. It is used for efficient matrix-vector multiplication for Toeplitz, circulant and other structured matrices. It is used in Filtering algorithms (overlap-add and overlap-save methods), Fast algorithms for discrete cosine or sine transforms (e.g. fast DCT used for JPEG and MPEG/MP3 encoding and decoding). Fast Chebyshev approximation, differential equations and computation of isotopic distributions uses FFT.

#### Aging

The development of fast algorithms for DFT can be traced to Carl Friedrich Gauss's unpublished work in 1805 when he needed it to interpolate the orbit of asteroids Pallas and Juno from sample observations. His method was very like the one published in 1965 by James Cooley and John Tukey, who are generally credited for the invention of the modern generic FFT algorithm. Between 1805 and 1965, some versions of FFT were published by other authors. James Cooley and John Tukey published a more general version of FFT in 1965 that is applicable when *N* is composite and not necessarily a power of 2.

In the modern days FFT is observed to be widely used in the engineering industry. For instance, it is used in digital signal processing (DSP) to modify, filter and decode digital audio, video and images. It is the foundation for voice recognition and myriad other pattern recognition and image compression applications. A live example would be, noise-cancelling headphones using FFT to turn unwanted sounds into simple waves so that inverse signals can be generated to cancel them or FFTs being used to sharpen edges and create effects in static images.

#### Costing

There are online FFT calculators available for generic purpose in the internet free of cost. Their main disadvantage is the lack of accuracy. FFTs used in mathematical perspective are available in math tools/ software. These tools are not free of cost and they normally drain the budget. Small industry and consumer products like headphones have in-built FFT calculators in them made available at a lower cost.

### WWWWH

**What is FFT:** A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

**Why is it used:** DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT. {\displaystyle N}The difference in speed can be enormous, especially for long data sets where *N* may be in the thousands or millions.

**Where and When is FFT used:** Some of the important applications of the FFT include:

* Fast large-integer and polynomial multiplication
* Efficient matrix-vector multiplication for Toeplitz, circulant and other structured matrices
* Filtering algorithms (see overlap-add and overlap-save methods)
* Fast algorithms for discrete cosine or sine transforms (e.g. fast DCT used for JPEG and MPEG/MP3 encoding and decoding)
* Fast Chebyshev approximation
* Solving difference equations
* Computation of isotopic distributions

**How is FFT implemented:** The best-known use of the Cooley–Tukey algorithm is to divide the transform into two pieces of size *N*/2 at each step, and is therefore limited to power-of-two sizes, but any factorization can be used in general. These are called the *radix-2* and *mixed-radix* cases, respectively. Although the basic idea is recursive, most traditional implementations rearrange the algorithm to avoid explicit recursion. Also, because the Cooley–Tukey algorithm breaks the DFT into smaller DFTs, it can be combined arbitrarily with any other algorithm for the DFT.

### SWOT Analysis

**Strength:** FFT reduce the complexity of computing the DFT from {\displaystyle O\left(N^{2}\right)}O(N2), which arises if one simply applies the definition of DFT, to {\displaystyle O(N\log N)}O(N log N){\displaystyle N}. The difference in speed can be enormous, especially for long data sets where N may be in the thousands or millions. In the presence of round-off error, many FFT algorithms are much more accurate than evaluating the DFT definition directly or indirectly.

**Weakness:** The drawback is that once you choose a particular size for the time window, that window is the same for all frequencies.

**Opportunities: It could be used in consumer applications.**

**Threat:**

## Design

### Structural Diagram

### Behavioral Diagram

## Test Plan