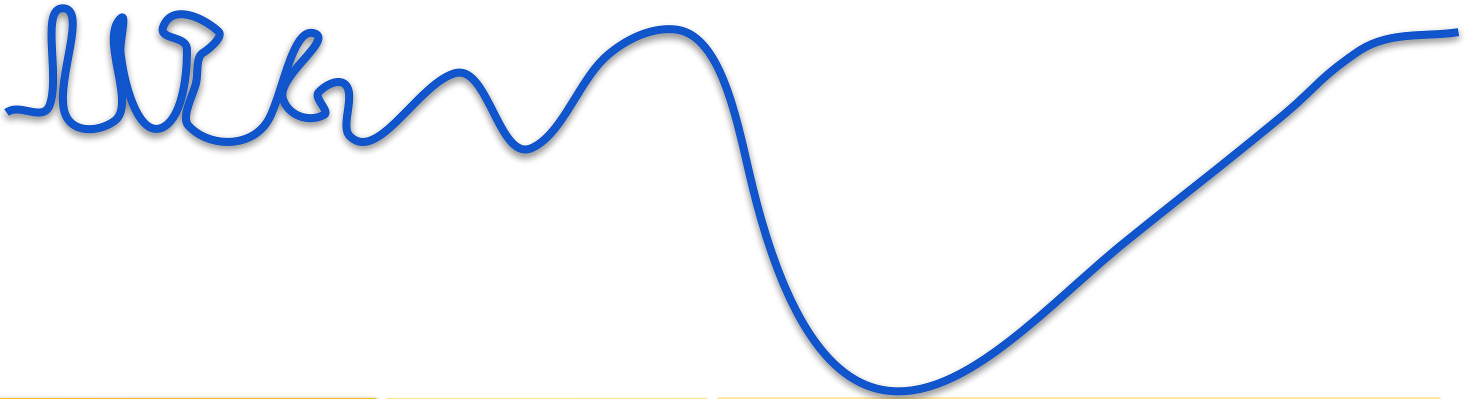
Computing with Signals



DA 623

Jan - May 2023

IIT Guwahati

Instructors: Neeraj Sharma

Lecture-23-24[17-20-Mar]

Recap

Fourier Transform

$$\hat{f}(s) = \int_{-\infty}^{\infty} e^{-2\pi i s t} f(t)\, dt$$

Discrete Fourier Transform

- How do we proceed?

$$\hat{f}(s) = \int_{-\infty}^{\infty} e^{-2\pi i s t} f(t)\, dt$$

(1) This will be discretized?

(2) Use Finite samples?

(3) Integral to summation?

(4) Can this be discretized?

Discrete Fourier Transform

- How do we proceed?

$$\hat{f}(s) = \int_{-\infty}^{\infty} e^{-2\pi i s t} f(t)\, dt$$

(1) This will be discretized?

(2) Use Finite samples?

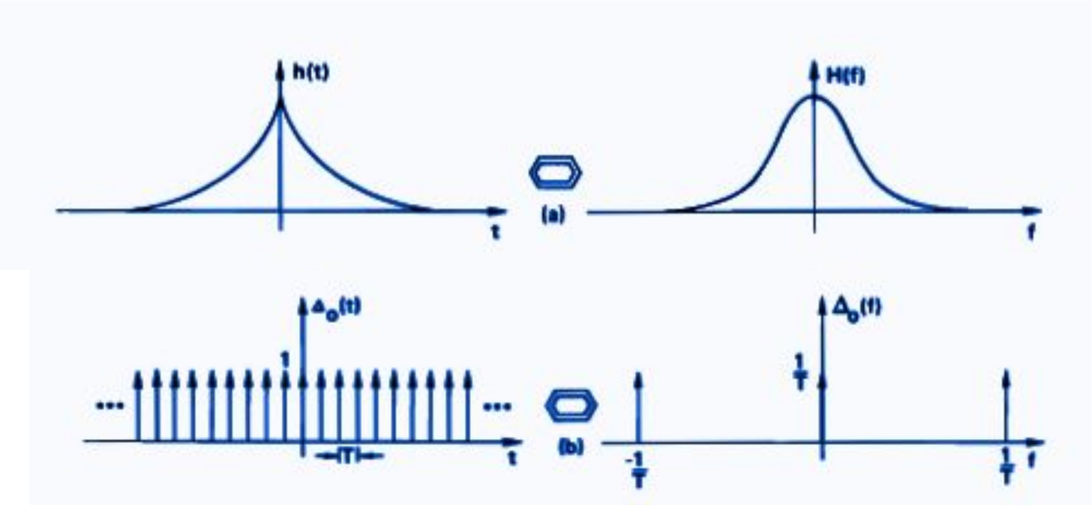
(3) Integral to summation?

(4) Can this be discretized?

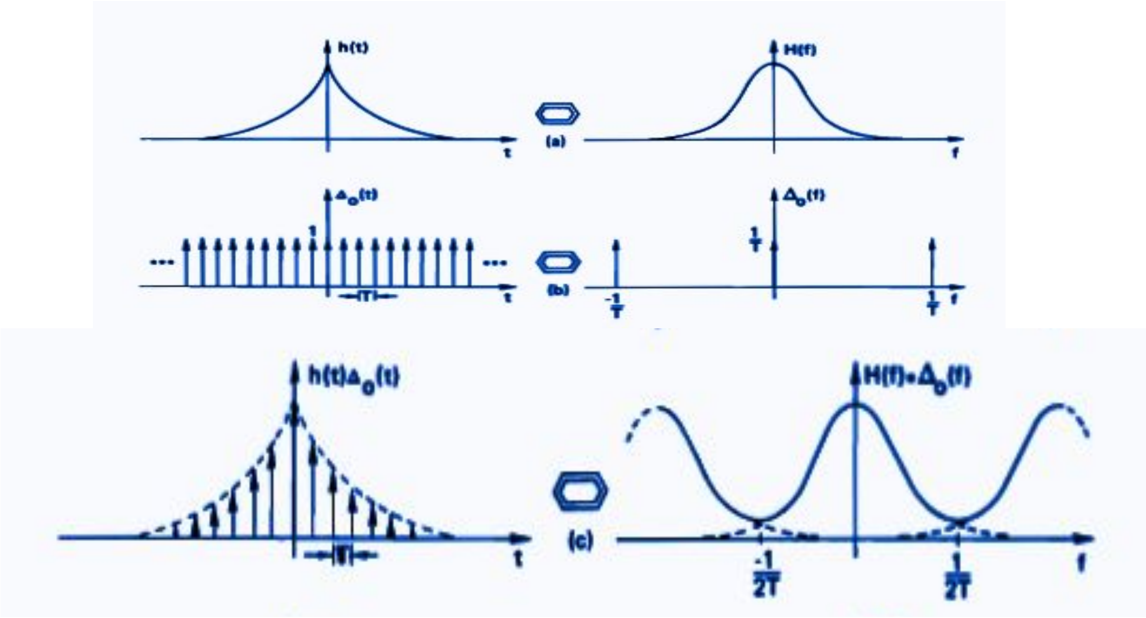
- Let's proceed through visualization.

- Continuous-time signal

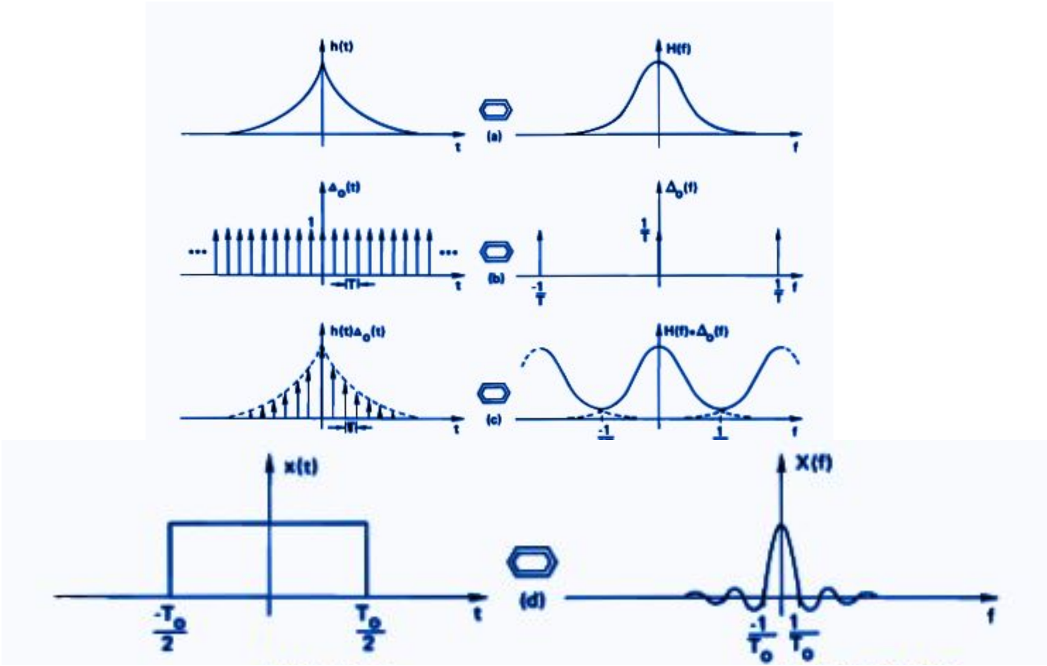
- sampling



- sampling

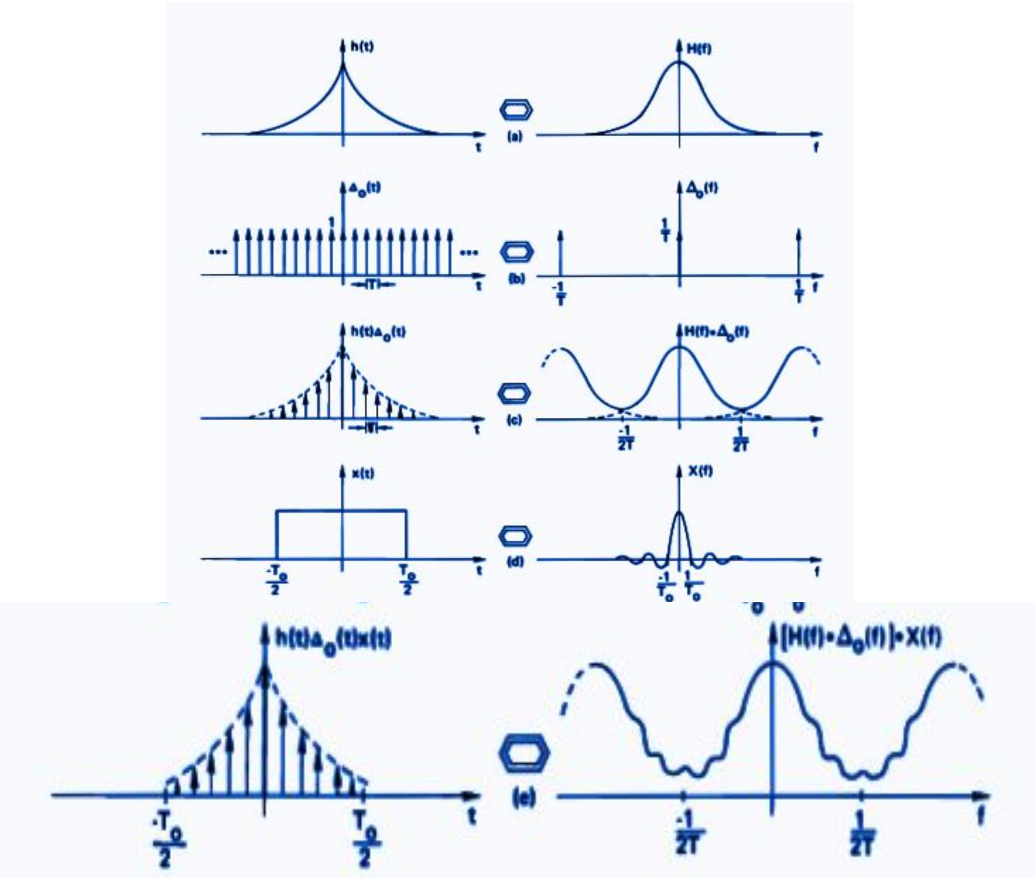


- sampling
- truncation

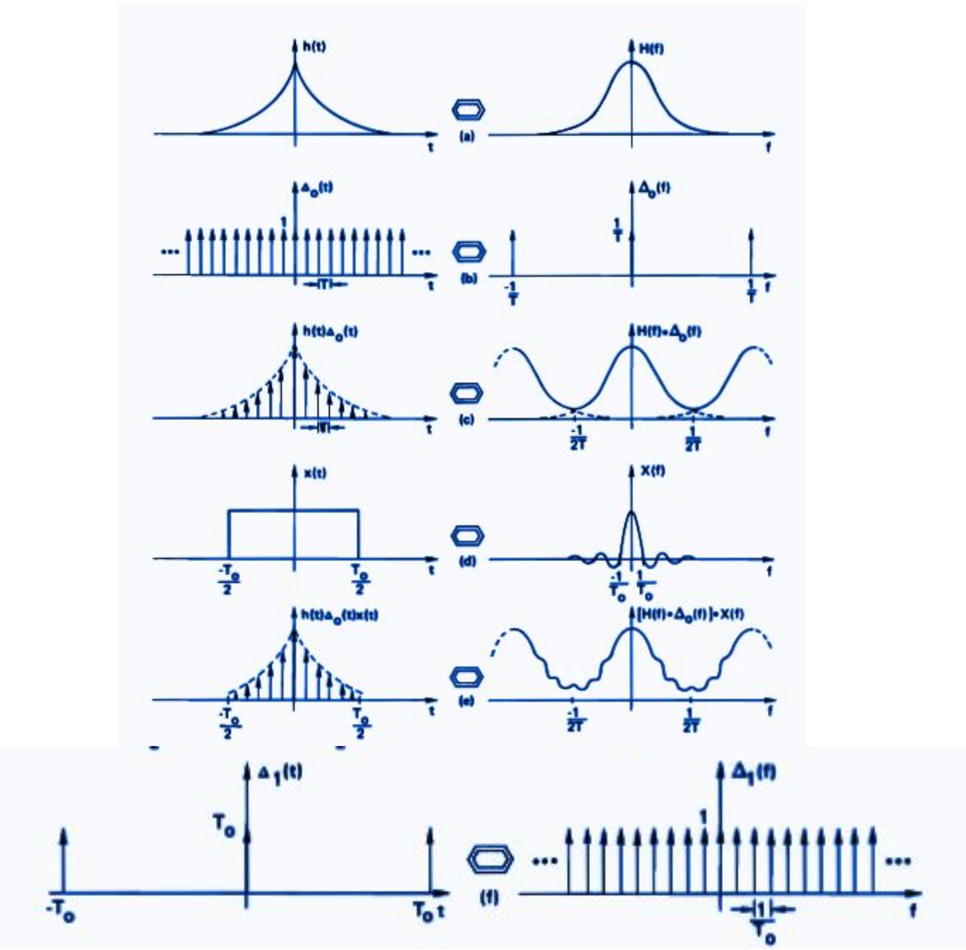


- sampling

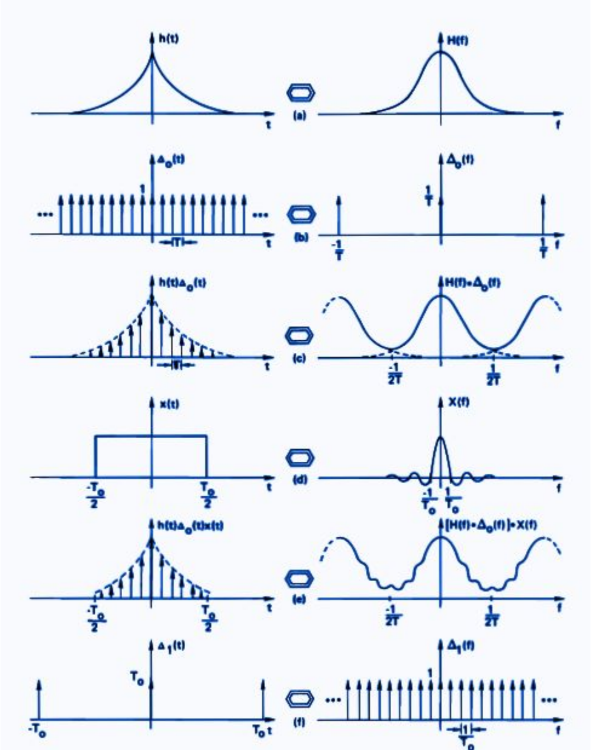
- truncation



- sampling
- truncation
- periodization

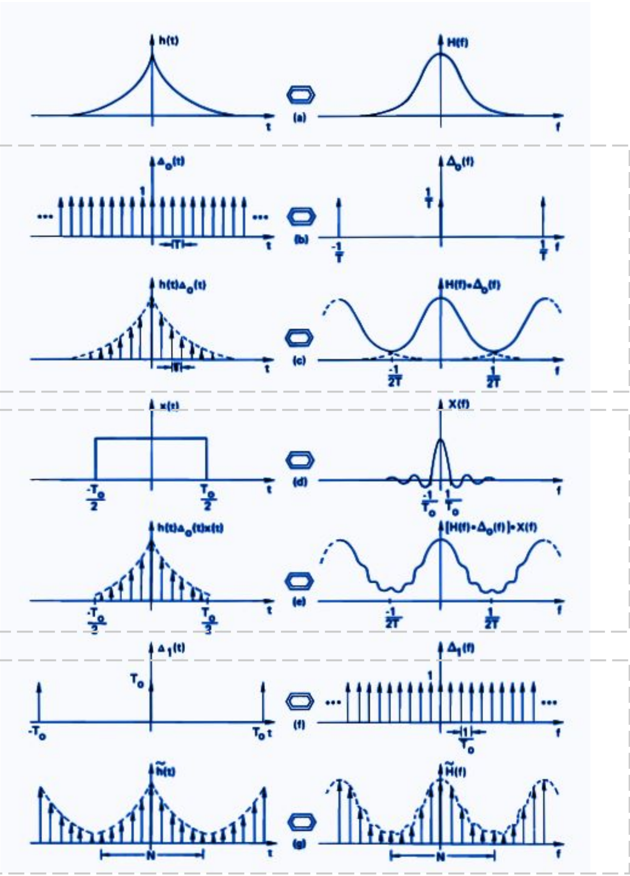


- sampling
- truncation
- periodization



Steps

- sampling
- truncation
- periodization



Result

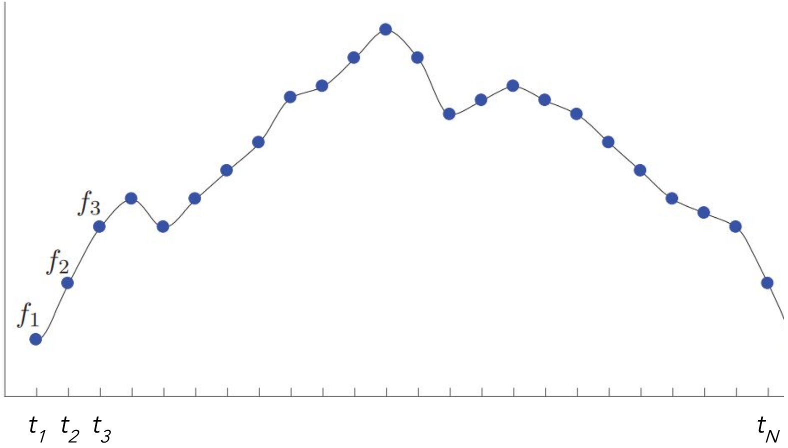
- Discrete Fourier Transform (DFT)

Discrete Fourier Transform (DFT)

$$F(s_m) = \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i s_m t_n}$$

$$= \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i n m / 2BL}$$

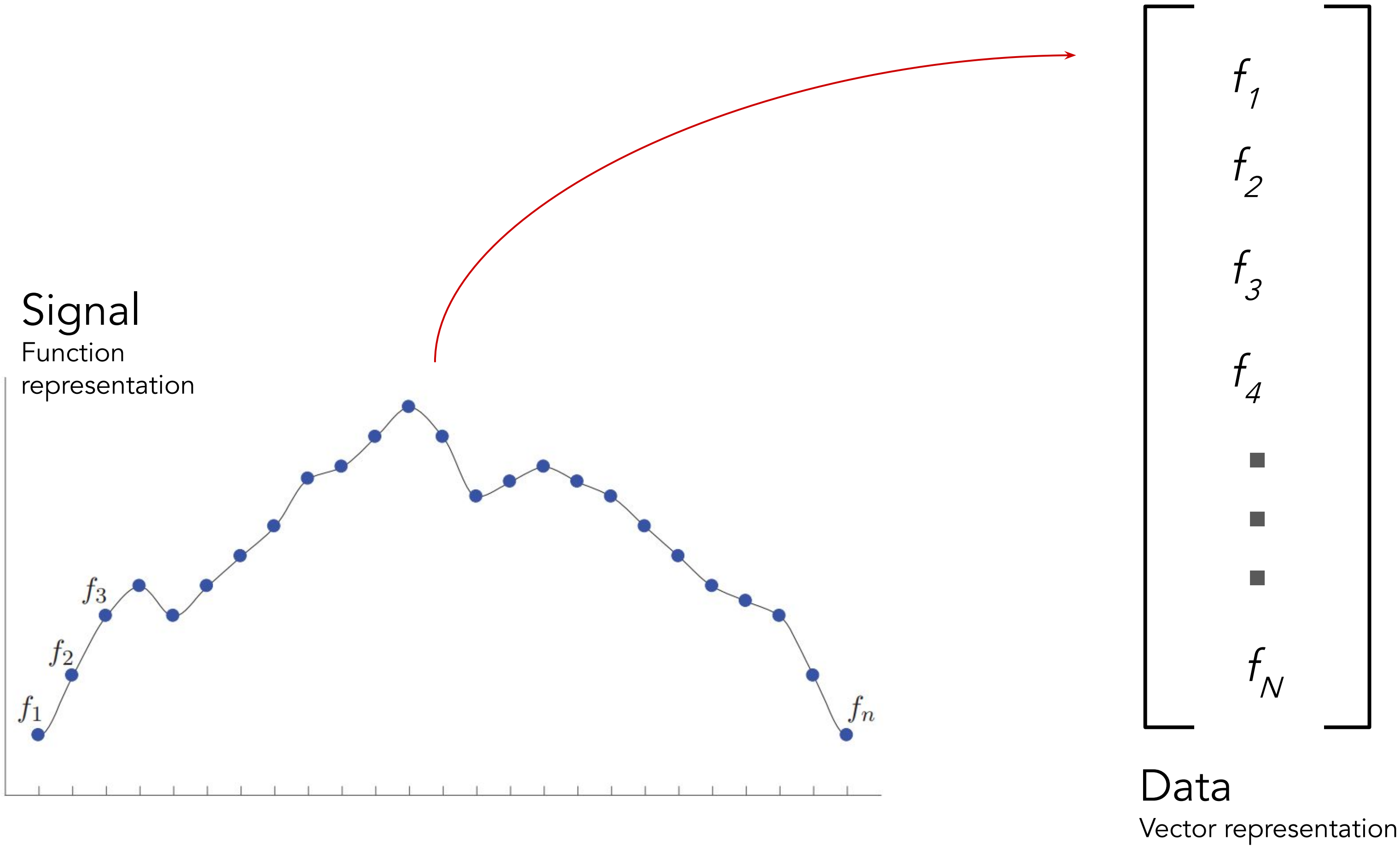
$$= \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i n m / N}$$



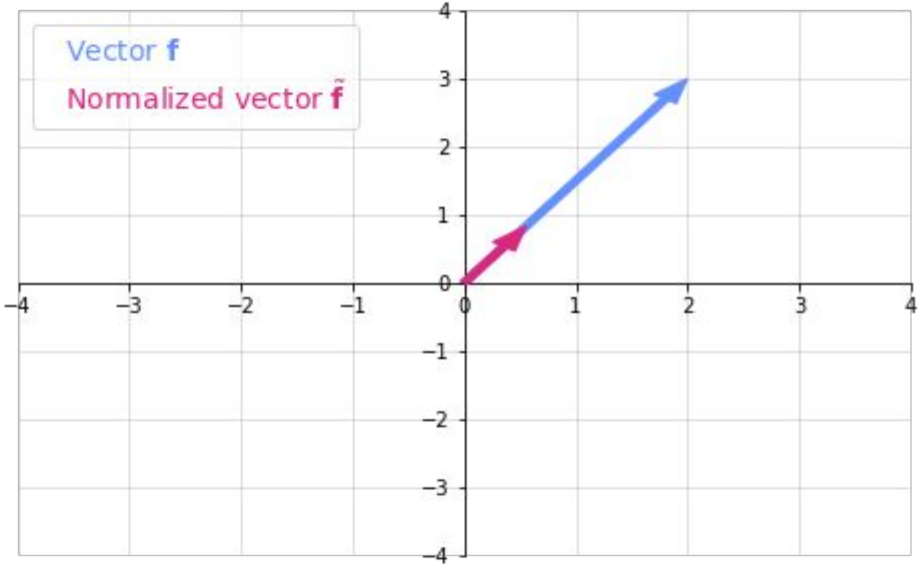
Assumptions:

- *f(t)* is (effectively) finite length in time and frequency
- Duration (length): *L*
- Bandwidth = *2B* (*-B* to *+B*)

  $t_n = nT_s$, $T_s = L/N = 1/2B$


Signal
Function
representation
f1
f2
f3
fn
f1
f2
f3
f4
fN
Data
Vector representation

In 2-D:



Goto:
jupyter-notebook

In N-D:

Identity basis

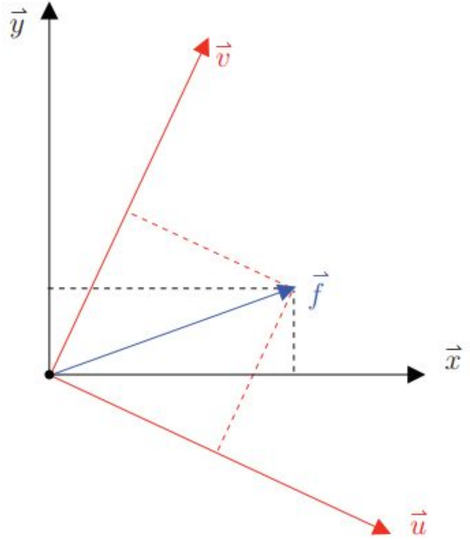
$$\vec{f} = \langle \vec{f}, \vec{x} \rangle \frac{\vec{x}}{\|\vec{x}\|^2} + \langle \vec{f}, \vec{y} \rangle \frac{\vec{y}}{\|\vec{y}\|^2}$$

In N-D:

$$\vec{f} = \langle \vec{f}, \vec{x} \rangle \frac{\vec{x}}{\|\vec{x}\|^2} + \langle \vec{f}, \vec{y} \rangle \frac{\vec{y}}{\|\vec{y}\|^2}$$

$$= \langle \vec{f}, \vec{u} \rangle \frac{\vec{u}}{\|\vec{u}\|^2} + \langle \vec{f}, \vec{v} \rangle \frac{\vec{v}}{\|\vec{v}\|^2}$$

Basis transformation



DFT

$$F(s_m) = \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i s_m t_n}$$

$$= \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i n m / N}$$

Notation,

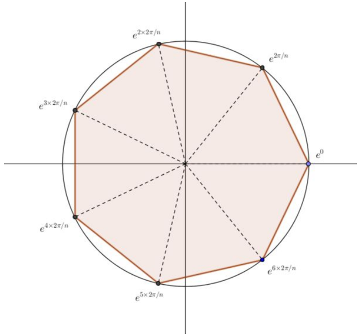
$$\omega = e^{2\pi i / N}$$

DFT

$$F(s_m) = \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i s_m t_n}$$

$$= \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i n m / N}$$

Notation,



$$\omega = e^{2\pi i/N}$$

Stacking all roots into a vector,

$$\boldsymbol{\omega}^k = (1, \omega^k, \omega^{2k}, \ldots, \omega^{(N-1)k})$$

- these are the N roots of unity in the complex plane

Additional notation,

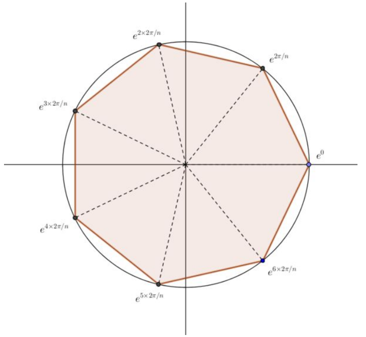
$$\boldsymbol{\omega}^{-k} = (1, \omega^{-k}, \omega^{-2k}, \ldots, \omega^{-(N-1)k})$$

DFT

$$F(s_m) = \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i s_m t_n}$$

$$= \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i n m / N}$$

Notation,



$$\omega = e^{2\pi i / N}$$

Stacking all roots into a vector,

$$\boldsymbol{\omega}^k = (1, \omega^k, \omega^{2k}, \ldots, \omega^{(N-1)k})$$

- these are the N roots of unity in the complex plane

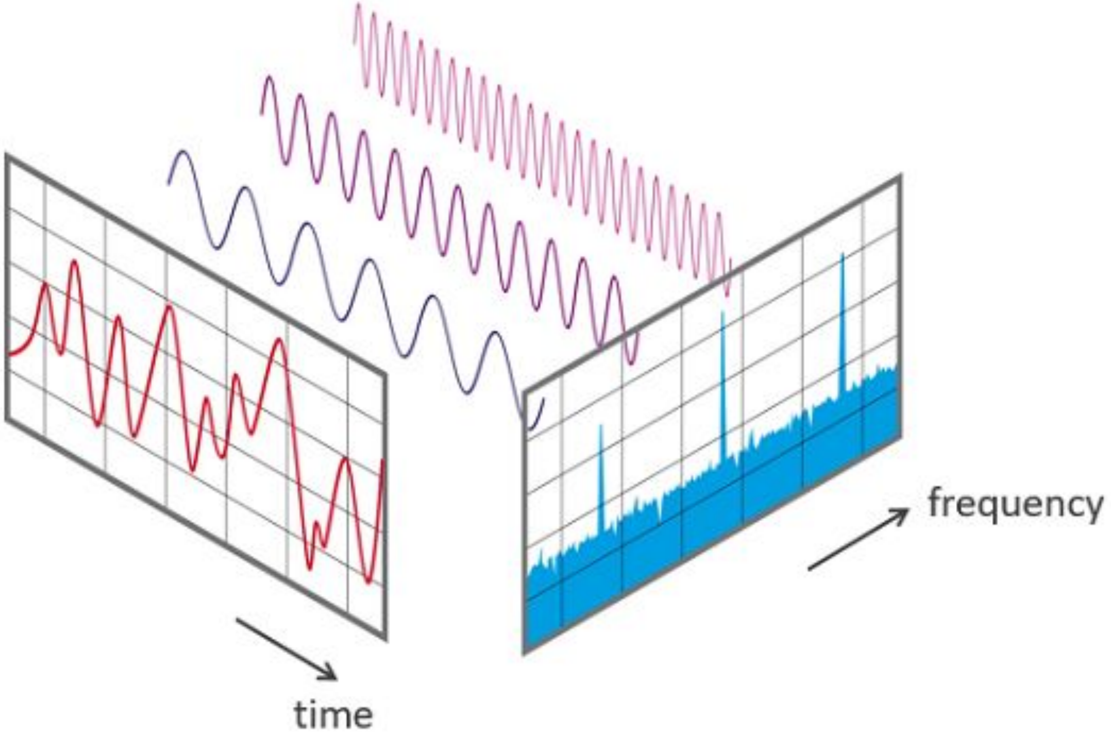
$$\mathbf{F}[m] = \sum_{k=0}^{N-1} \mathbf{f}[k] \omega^{-km} = \sum_{k=0}^{N-1} \mathbf{f}[k] e^{-2\pi i k m / N}$$

Additional notation,

$$\boldsymbol{\omega}^{-k} = (1, \omega^{-k}, \omega^{-2k}, \ldots, \omega^{-(N-1)k})$$

DFT

$$\mathbf{F}[m] = \sum_{k=0}^{N-1} \mathbf{f}[k]\omega^{-km} = \sum_{k=0}^{N-1} \mathbf{f}[k]e^{-2\pi i k m/N}$$


frequency
time

DFT

$$\mathbf{F}[m] = \sum_{k=0}^{N-1} \mathbf{f}[k]\omega^{-km} = \sum_{k=0}^{N-1} \mathbf{f}[k]e^{-2\pi ikm/N}$$

$$\begin{pmatrix} F[0] \\ F[1] \\ F[2] \\ \vdots \\ F[N-1] \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1\cdot 1} & \omega^{-1\cdot 2} & \cdots & \omega^{-(N-1)} \\ 1 & \omega^{-2\cdot 1} & \omega^{-2\cdot 2} & \cdots & \omega^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(N-1)\cdot 1} & \omega^{-(N-1)\cdot 2} & \cdots & \omega^{-(N-1)^2} \end{pmatrix} \begin{pmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N-1] \end{pmatrix}.$$

Fourier Transform
vector

DFT
Square Matrix

Data
vector

DFT Computation

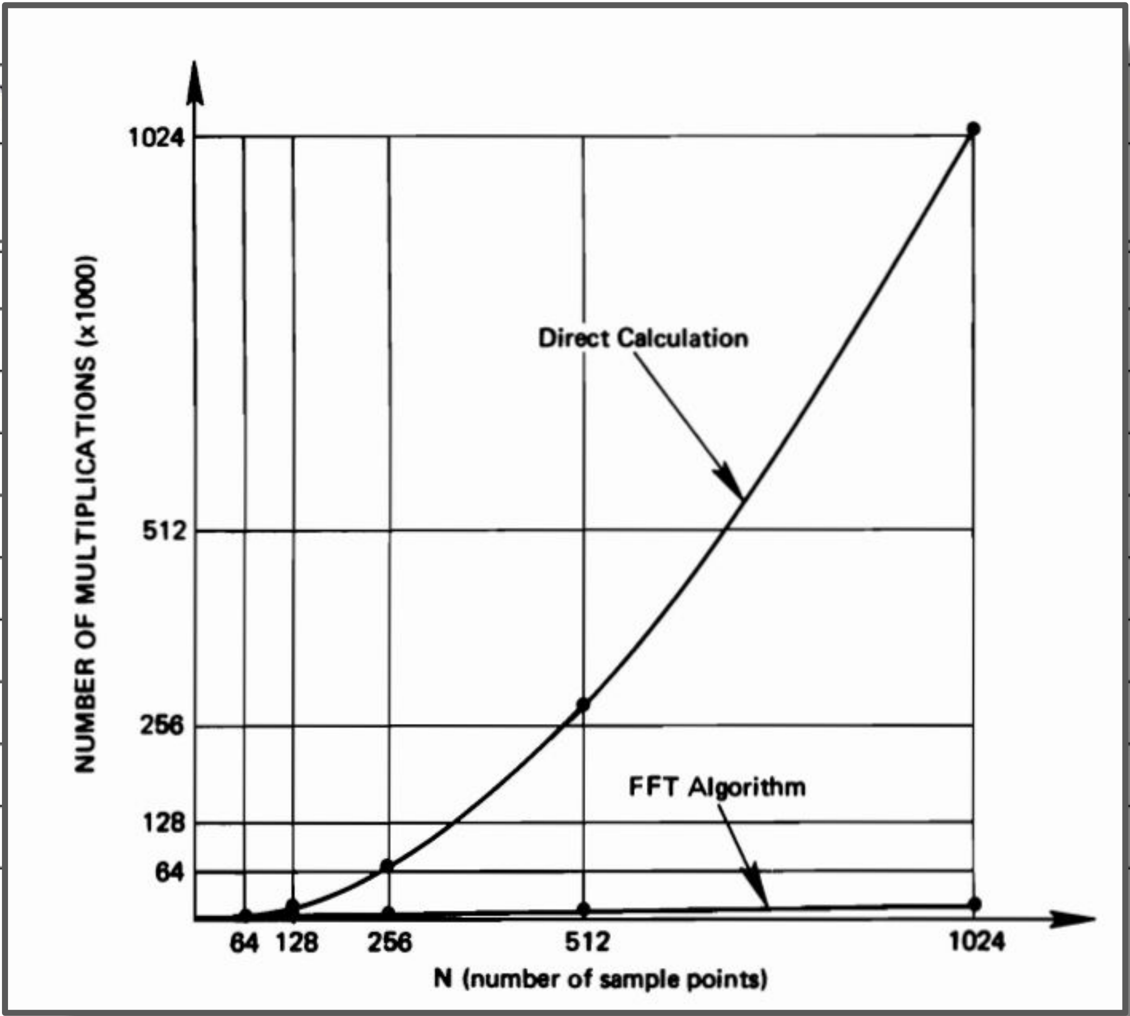
N	DFT (real multiplies) $4N^2$	DFT (real additions) $2N(2N-1)$
2	16	12
4	64	56
8	256	240
16	1,024	992
32	4,096	4,032
64	16,384	16,256
128	65,536	65,280
256	262,144	261,632
512	1,048,576	1,047,552
1,024	4,194,304	4,192,256

- A lot of multiplications and additions as the signal length increase.
- Is there a way to compute it efficiently?

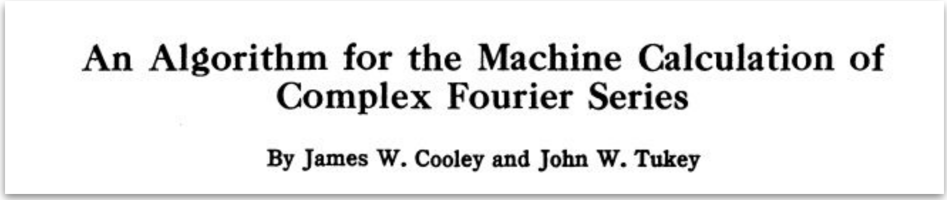
DFT Computation using FFT

N	*DFT (real multiplies)*	*DFT (real additions)*	*FFT (real multiplies)*	*FFT (real additions)*
	$4N^2$	$2N(2N-1)$	$2N\log_2 N$	$3N\log_2 N$
2	16	12	4	6
4	64	56	16	24
8	256	240	48	72
16	1,024	992	128	192
32	4,096	4,032	320	480
64	16,384	16,256	768	1,152
128	65,536	65,280	1,792	2,688
256	262,144	261,632	4,096	6,144
512	1,048,576	1,047,552	9,216	13,824
1,024	4,194,304	4,192,256	20,480	30,720

N	DFT	FFT (real additions)
		$3N \log_2 N$
2		6
4		24
8		72
16		192
32		480
64		1,152
128		2,688
256		6,144
512		13,824
1,024		30,720



FFT: Fast Fourier Transform

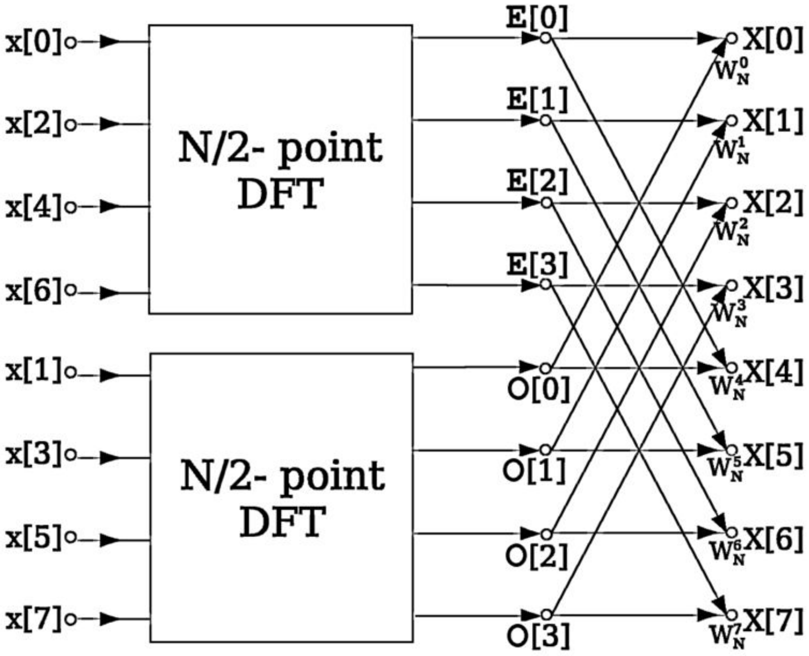


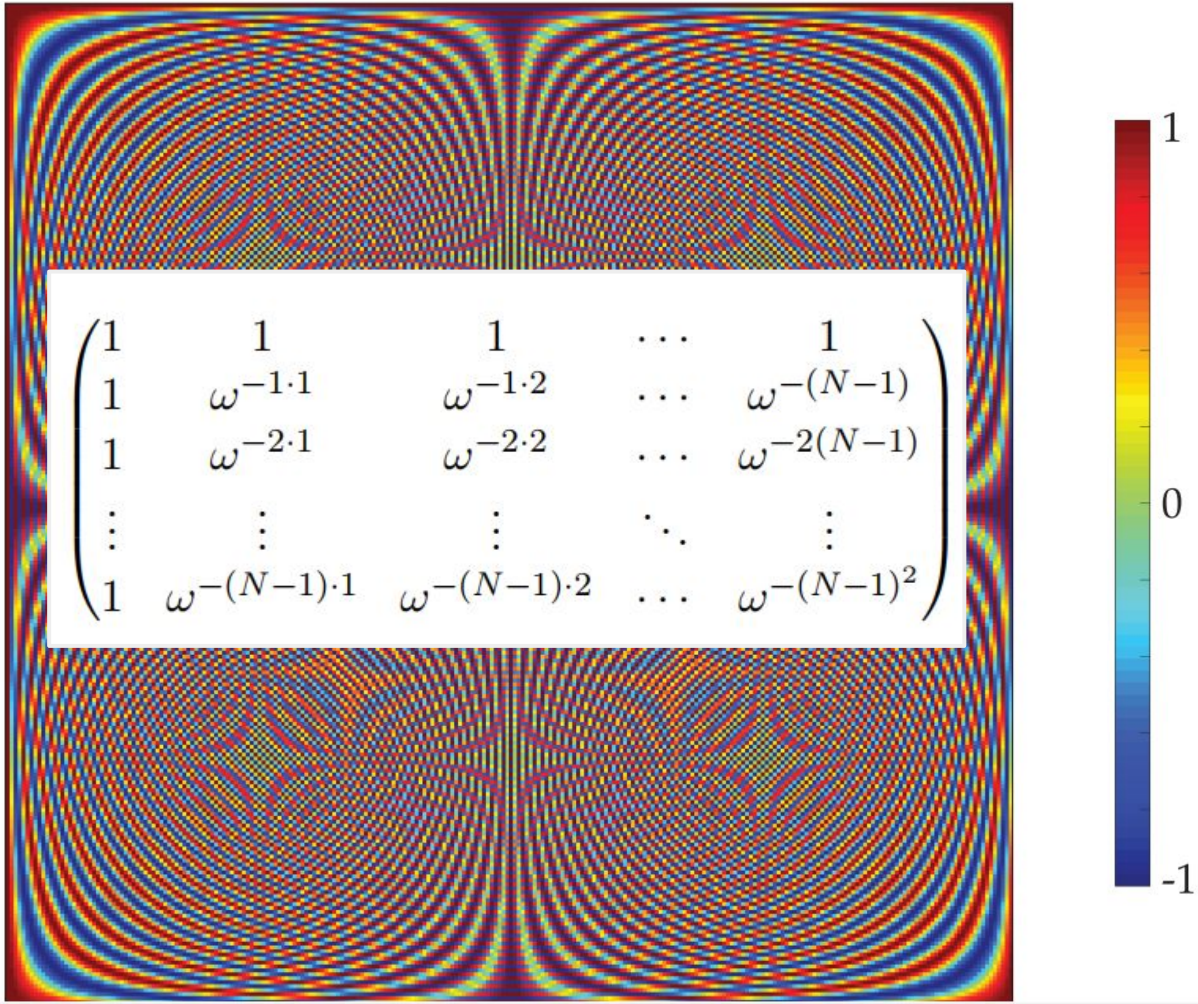
An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An algorithm for faster computation of DFT.

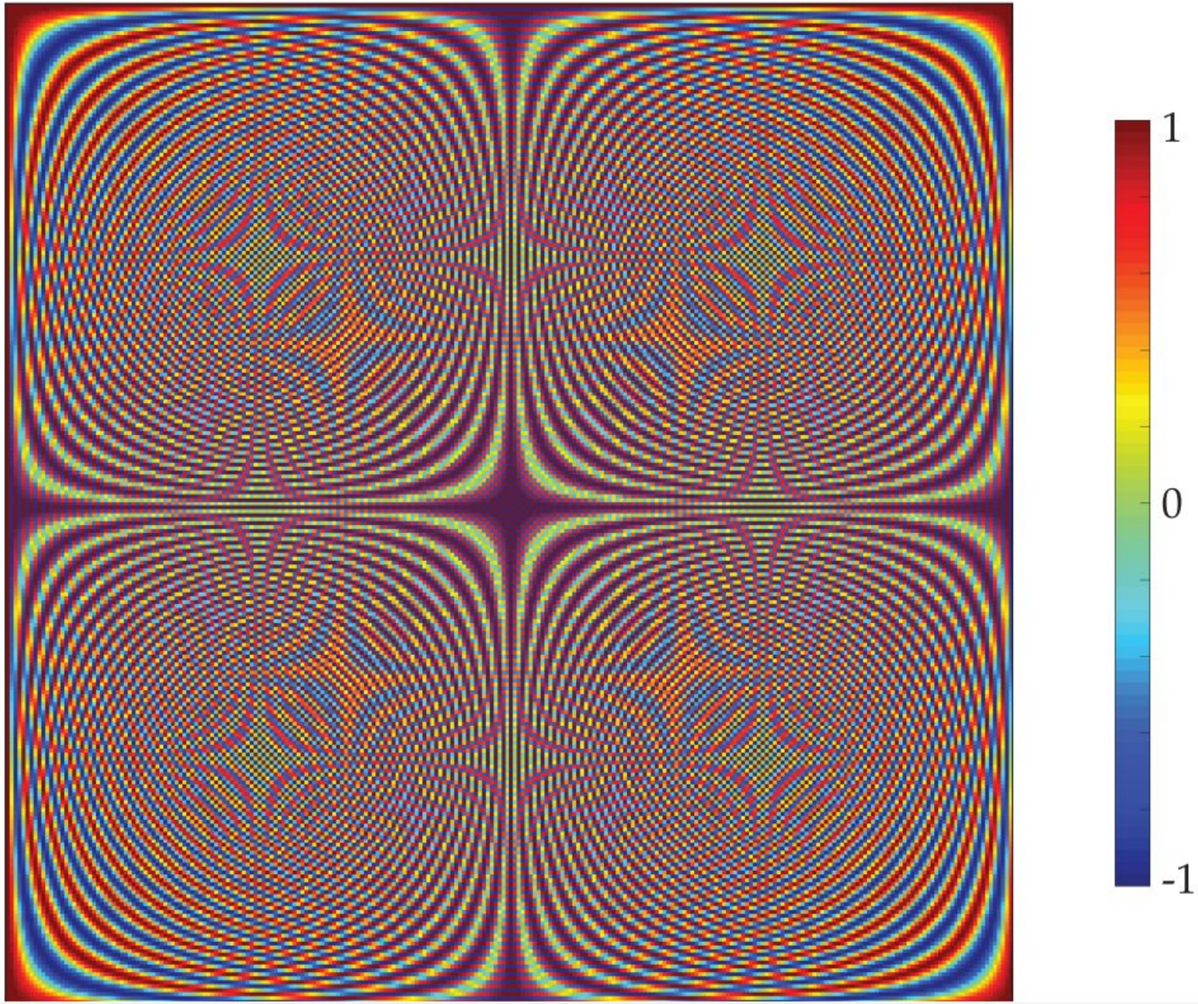
Proceeds by making group of even and odd indices in the input





$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1\cdot 1} & \omega^{-1\cdot 2} & \cdots & \omega^{-(N-1)} \\ 1 & \omega^{-2\cdot 1} & \omega^{-2\cdot 2} & \cdots & \omega^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(N-1)\cdot 1} & \omega^{-(N-1)\cdot 2} & \cdots & \omega^{-(N-1)^2} \end{pmatrix}$$

- Visualizing the Real [DFT]



- Cooley-Tukey algorithm calculates the DFT directly with fewer summations
- The trick to the Cooley-Tukey algorithm is recursion.
- Split the matrix we wish to perform the FFT on into two parts: one for all elements with even indices and another for all odd indices.
- We then proceed to split the array again and again until we have a manageable array size to perform a DFT (or similar FFT) on.
- We can also perform a similar re-ordering by using a bit reversal scheme, where we output each array index's integer value in binary and flip it to find the new location of that element.
- Complexity to ~O(N logN)

https://vanhunteradams.com/FFT/FFT.html

A fast means for Fourier transform implies lots of applications! Let’s look at some funs ones!

- https://dazzling-jang-471a34.netlify.app/
- https://jojo.ninja/fluctus/
- More here: https://github.com/willianjusten/awesome-audio-visualization

Summary

$$\begin{pmatrix} F[0] \\ F[1] \\ F[2] \\ \vdots \\ F[N-1] \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1\cdot 1} & \omega^{-1\cdot 2} & \cdots & \omega^{-(N-1)} \\ 1 & \omega^{-2\cdot 1} & \omega^{-2\cdot 2} & \cdots & \omega^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(N-1)\cdot 1} & \omega^{-(N-1)\cdot 2} & \cdots & \omega^{-(N-1)^2} \end{pmatrix} \begin{pmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N-1] \end{pmatrix}.$$

Fourier Transform
vector

DFT
Square Matrix

Data
vector

- Matrix multiplication is $O(N^2)$ computations
- This is a lot of computation for N>>1, as usual case
- Linear scaling is desired in most applications
- Fast Fourier Transform (FFT) algorithm enables computing DFT in O(N logN)

Resources

Gauss and the History of the Fast Fourier Transform

Michael T. Heideman
Don H. Johnson
C. Sidney Burrus

INTRODUCTION

THE fast Fourier transform (FFT) has become well known as a very efficient algorithm for calculating the discrete Fourier Transform (DFT) of a sequence of *N* numbers. The

coefficients of Fourier series were developed at least a century earlier than had been thought previously. If this year is accurate, it predates Fourier's 1807 work on harmonic analysis. A second reference to Gauss' algorithm

How the FFT Gained Acceptance

James W. Cooley

1. Introduction

The fast Fourier transform (FFT) has had a fascinating history, filled with ironies and enigmas. Even more appropriate for this meeting and its sponsoring professional society, it speaks not only of numerical analysis but also of the importance of the functions performed by professional societies.