The Fast Fourier Transform: A Brief Overview with Applications

Petros Kondylis

December 4, 2014

Timeline

Researcher	Date	Length of Sequence	Application
C.F. Gauss	1805	Any Composite Integer	Interpolation of orbits of celestial bodies
F. Carlini	1828	12	Harmonic Analysis of Barometric Pressure
A. Smith	1846	4, 8, 16, 32	Correcting deviations in compasses on ships
J. D. Everett	1860	12	Modelling underground temperature variations
C. Runge	1903	$2^n K$	Harmonic Analysis of Functions
K. Stumpff	1939	$2^n K$ & 3^n	Harmonic Analysis of Functions
Danielson & Lanczos	1942	2^n	X-ray diffraction in crystals
L.H. Thomas	1948	Integer /w relatively prime factors	Harmonic Analysis of Functions
I. J. Good	1958	Integer /w relatively prime factors	Harmonic Analysis of Functions
Cooley & Tukey	1965	Composite Integer	Harmonic Analysis of Functions
S. Winograd	1976	Integer /w relatively prime factors	Complexity theory for Harmonic Analysis

Discrete Fourier Transform

Definition (Discrete Fourier Transform)

Matrix-vector-multiplication $y = Mx$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \omega_n^{kj} & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

where $x = (a_0, \dots, a_{n-1}), y = (y_0, \dots, y_{n-1}) \in \mathbb{C}^n, \omega_n = e^{2\pi i/n}$ (primitive nth root of unity) and

The direct evaluation requires $O(n^2)$ complex multiplications and additions.

To Divide is to Conquer

- Map the original problem in such a way that the following inequality is satisfied:

$$\sum cost(subproblems) + cost(mapping) < cost(originalproblem)$$

- The division can be applied recursively to the subproblems leading to a reduction of the order of complexity.
- **Question:** But How ?
- **Answer:** Consider Subsets of the initial sequence
  Take the DFT of these subsequences
  Reconstruct the DFT of the Initial subsequence

To Divide...

The original problem:

$$y_k = \sum_{j=0}^{n-1} a_j \, (\omega_n^k)^j, \qquad k = 0, \ldots, n-1$$

Let $P_l, l = 0, \ldots, r-1$ be the partition of $\{0, 1, \ldots, n-1\}$ defining r different subsets of the input sequence.

The partition gives:

$$y_k = \sum_{l=0}^{r-1} \sum_{j \in P_j} a_j \, (\omega_n^k)^j, \qquad k = 0, \ldots, n-1$$

$$y_k = \sum_{l=0}^{r-1} \omega_n^{j_l} \sum_{j \in P_j} a_j \, (\omega_n^k)^{j-j_l}$$

The chosen partition depends on the nature of the problem (input sequence).

To Divide...

Example: For input lengths equal to powers of 2 i.e $n = 2^s, s \in \mathbb{N}$ we can have a partition $P_l, l = 0, 1$ where $P_0 = \{0 : 2 : n-1\}, P_1 = \{1 : 2 : n-1\}$.

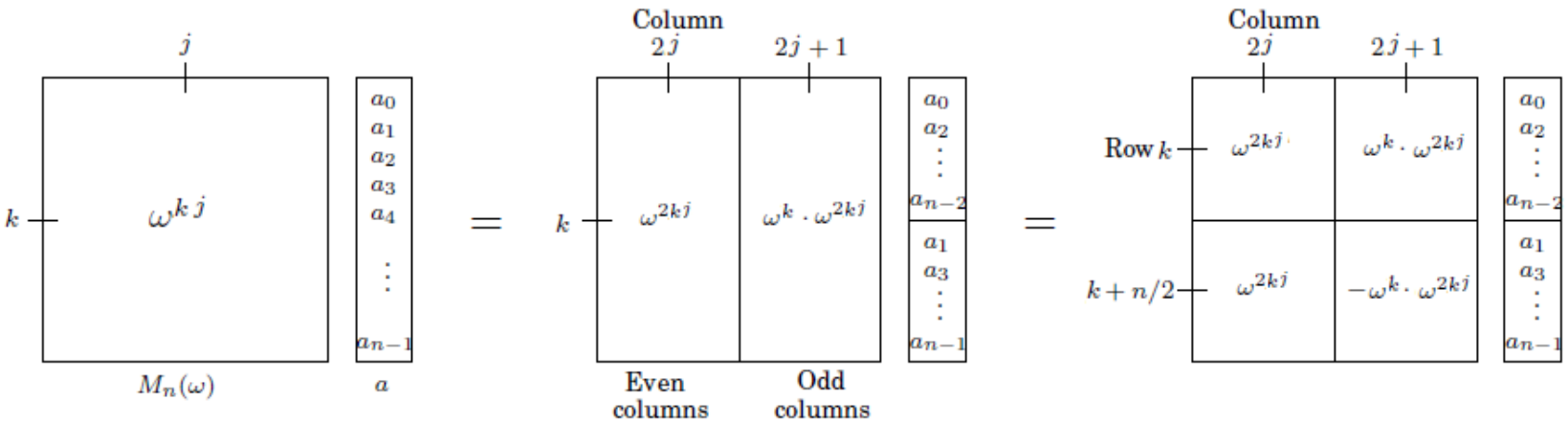
$$y_k = \sum_{j=0:2}^{n-1} a_j \left(\omega_n^k\right)^j + \omega_n^k \sum_{j=1:2}^{n-1} a_j \left(\omega_n^k\right)^{j-1}$$

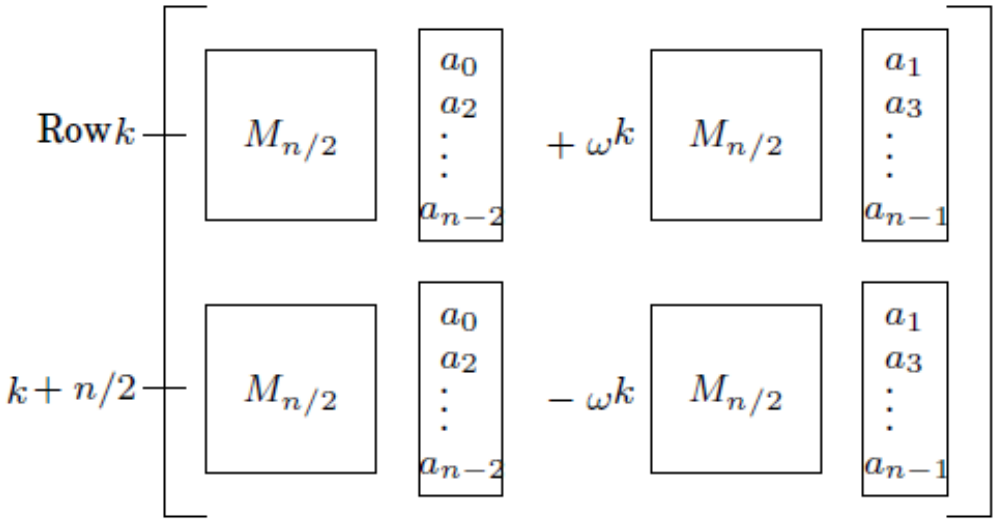
$$y_k = \sum_{j=0:2}^{n-1} a_j \left(\omega_n^{2k}\right)^{j/2} + \omega_n^k \sum_{j=0:2}^{n-1} a_{j+1} \left(\omega_n^{2k}\right)^{j/2}$$

$$y_k = \sum_{j=0}^{n/2-1} a_{2j} \left(\omega_{n/2}^k\right)^j \pm \omega_n^k \sum_{j=0}^{n/2-1} a_{2j+1} \left(\omega_{n/2}^k\right)^j$$

$$k = 0, \ldots, n/2 - 1$$

To Divide (again)...





To Conquer...

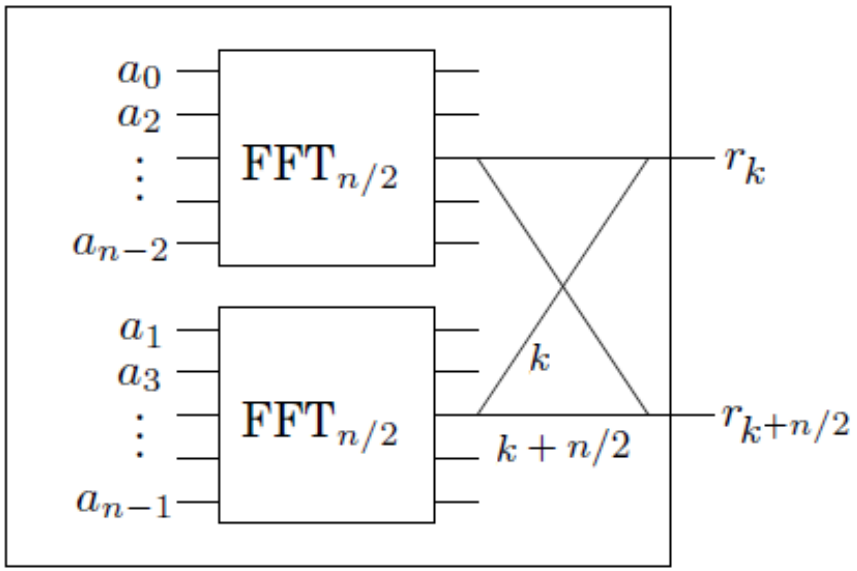
Algorithm 1 Recursive-FFT(a)

1: $n = a.lenght$
2: **if** $n == 1$ **then**
3: **return** a
4: $\omega_n = e^{2\pi i/n}$
5: $\omega = 1$
6: $a^{[0]} = (a_0, a_2, ..., a_{n-2})$
7: $a^{[1]} = (a_1, a_3, ..., a_{n-1})$
8: $y^{[0]} = Recursive - FFT(a^{[0]})$
9: $y^{[1]} = Recursive - FFT(a^{[1]})$
10: **for** $k = 0$ **to** $n/2 - 1$ **do**
11: $t = \omega\, y_k^{[1]}$
12: $y_k = y_k^{[0]} + t$
13: $y_{k+n/2} = y_k^{[0]} - t$
14: $\omega = \omega\omega_n$
15: **return** y

Complexity: $T(n) = 2T(n/2) + \Theta(n) = \Theta(nlgn)$.

The Butterfly Operation

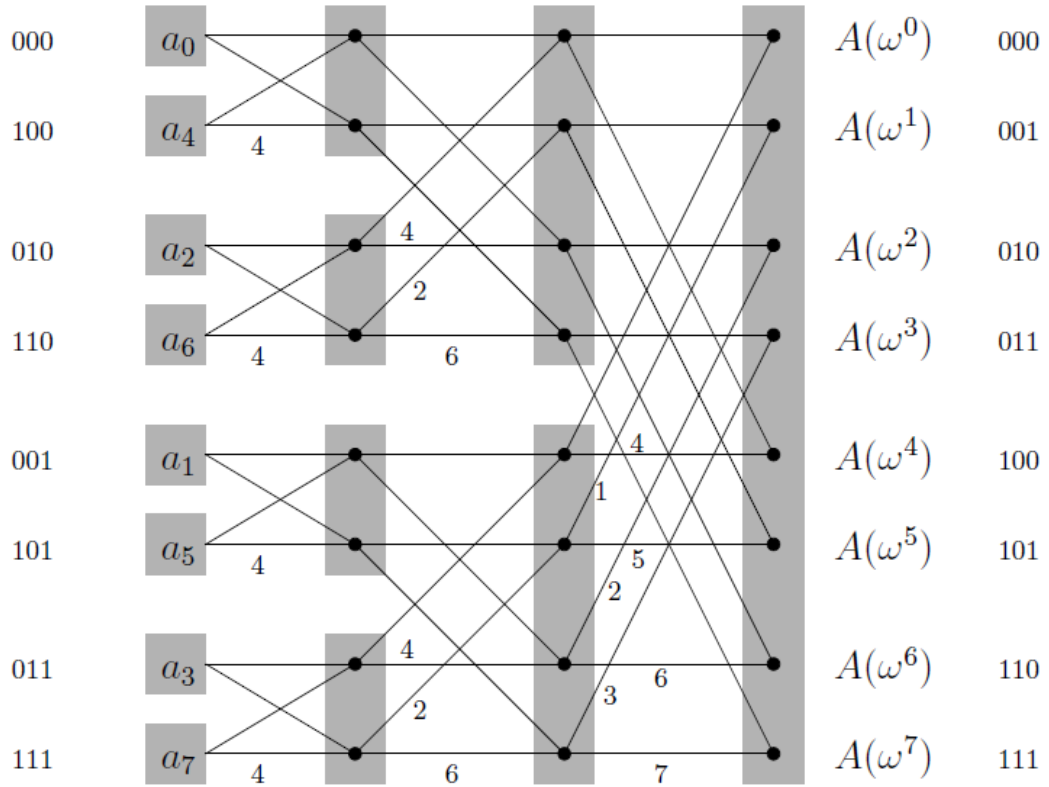
FFT_n (input: $a_0, \ldots, a_{n-1}$, output: $r_0, \ldots, r_{n-1}$)



$$r_k = y_k^{[0]} + \omega^k \, y_k^{[1]}$$

$$r_{k+n/2} = y_k^{[0]} - \omega^k \, y_k^{[1]}$$

The FFT circuit: Example



- The input is arranged in bit reversed order
- Unique path between each input a_k and each output $A(\omega^j)$
- On the path between a_k and $A(\omega^j)$ the labels add up to $jk \ mod \ 8$

Multiplying Polynomial

The product of two degree-d polynomials is a polynomial of degree 2d:

$$A(x) = a_0 + a_1 x + \ldots + a_d x^d \text{ and } B(x) = b_0 + b_1 x + \ldots + b_d x^d$$

$$C(x) = A(x)B(x) = c_0 + c_1 x + \ldots + c_{2d} x^{2d}$$

$$c_k = a_0 b_k + a_1 b_{k-1} + \ldots + a_k b_0 = \sum_{i=0}^{k} a_i b_{k-1}$$

Complexity: $\Theta(d^2)$

Representing Polynomials

Two ways of representing polynomials

- Coefficient form $a_0, a_1, \ldots, a_d$
- Values $A(x_0), A(x_1), \ldots A(x_d)$ at $d+1$ distinct points

Multiplying polynomials in the value representation takes linear time

$$C(x_k) = A(x_k)B(x_k), \quad k = 0, \ldots, 2d$$

Are we there yet?

Polynomial multiplication

Input: Coefficients of two polynomials, $A(x)$ and $B(x)$, of degree d
Output: Their product $C = A \cdot B$

Selection
Pick some points $x_0, x_1, \ldots, x_{n-1}$, where $n \geq 2d+1$
Evaluation
Compute $A(x_0), A(x_1), \ldots, A(x_{n-1})$ and $B(x_0), B(x_1), \ldots, B(x_{n-1})$
Multiplication
Compute $C(x_k) = A(x_k)B(x_k)$ for all $k = 0, \ldots, n-1$
Interpolation
Recover $C(x) = c_0 + c_1 x + \cdots + c_{2d} x^{2d}$

- Evaluation takes $\Theta(n^2)$ time
- Multiplication takes $\Theta(n)$ time
- Interpolation takes $\Theta(n^2)$ time

This joke isn't funny (or is it)!

Evaluation... as in Linear Transformation... as in Fast Fourier Transformation

$$\begin{pmatrix} A(x_0) \\ A(x_1) \\ A(x_2) \\ \vdots \\ A(x_{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Is it OK if I... $x_k = \omega_n^k$? By all means!

$$\begin{pmatrix} A(\omega_n^0) \\ A(\omega_n^1) \\ A(\omega_n^2) \\ \vdots \\ A(\omega_n^{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

But this I can do in $\Theta(nlgn)$ time (you don't say...!)

Some kind of Magic (what ???)

Lemma

The columns of matrix M are orthogonal to each other.

$$M = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix}$$

Proof.

Take the inner product of any columns j and k of matrix M

$1 + \omega^{k-j} + \omega^{2(k-j))} + \ldots + \omega^{(n-1)(k-j)} = (1 - \omega^{n(k-j)})/(1 - \omega^{k-j})$

For $j \neq k$ it evaluates to 0.

For $j = k$ it evaluates to n. □

Can we go back now please

The **orthogonality property** can be summarized in the single equation

$$\begin{aligned} MM^* &= nI \\ M^* &= nM^{-1} \\ M^* &= M(\omega^{-1}) \end{aligned}$$

Inversion Formula: $M(\omega_n)^{-1} = \frac{1}{n}M(\omega^{-1})$

M^* is the **conjugate transpose** of M.
But ω^{-1} is also an *nth* **root of unity** and therefore **multiplication** by $M(\omega^{-1})$, i.e **interpolation**, is itself just an **FFT operation**, but with ω replaced by ω^{-1}.

Polynomial Multiplication in a Nutshell