Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 12, Number 2 (2016), pp. 1671-1676

http://www.ripublication.com/gjpam.htm

The intrinsic structure of FFT and a synopsis of SFT

Hwajoon Kim[1]

Kyungdong University,
School of IT Engineering,
11458, Kyungdong Univ. Rd. 27,
Yangju, Gyeonggi, S. Korea.

Abstract

We have checked the intrinsic structure of fast Fourier transform, centering on Danielson-Lanczos lemma, and investigated a synopsis of sparse Fourier transform.

AMS subject classification: 34A25, 28A05.
Keywords: Fast Fourier transform.

1. Introduction

If a given function $f(x)$ has only values at finite points, we think over the discrete Fourier transform (DFT). This DFT concerns large amounts of equal speed data, as it occur in compression, signal processing, analysis images, time series analysis, and various simulation problems. The fast Fourier transform (FFT) is a practical method of computing DFT that needs only $O(N)\log_2 N$ operations instead of $O(N^2)$. Although this algorithm is not novel, the importance is still valid because the existing commercialized system is based on FFT. Additionally, this FFT has an important application to Radon transform which is widely applicable to tomography, and the short-term fast Fourier transform weightily is applied to the analysis of brain waves. It is well-known fact that the Fourier transform of f has the form of

$$\hat{f}(w) = \int_{-\infty}^{\infty} f(x)e^{-iwx}\,dw,$$

[1]Corresponding author.

and we can interpret that a function f is the given signal and $\hat{f}$ is the frequency spectrum of f. Let us define the DFT of the given signal $f = [f_0, \cdots, f_{N-1}]^T$ to be the vector $\hat{f} = [\hat{f_0}, \cdots, \hat{f_{N-1}}]$ with the components

$$\hat{f_n} = Nc_n = \sum_{k=0}^{N-1} f_k e^{-inx_k},$$

where $f_k = f(x_k)$ and $x_k = 2\pi k/N$ [7]. This is the frequency spectrum of the signal f, and we can write $\hat{f} = F_N f$, where the $N \times N$ Fourier matrix $F_N = [e_{nk}]$ has the entries

$$e_{nk} = e^{-inx_k} = e^{-2\pi ink/N} = w^{nk}, \ w = w_N = e^{-2\pi i/N}$$

for $n, k = 0, \cdots, N-1$. Here, since N is normally large, we need quite a number of operations. To avoid this difficulty, we use FFT, a computational method for the DFT, which needs only $O(N \log N)$ instead of $O(N)$. This FFT can get possible the Fourier matrix to break down the given problem into smaller problems by using

$$w_{2M}^2 = (e^{-2\pi i/N})^2 = e^{-2\pi i/M} = w_M$$

for $N = 2M$, sometimes called the Danielson-Lanczos lemma.

With relation to this Cooley-Tukey FFT, several researches have been pursued and progressed [1-6, 9-11]. The Hartley transform [2], an efficient real Fourier transform algorithm, gave a further increase in speed, Winograd transform algorithm [10] and Sande-Tukey algorithm [9] gave big influence as well. The sparse Fourier transform(SFT) is the most advanced algorithm recently presented by Katabi, and it is a kind of improved FFT which is increased in speed as ten times as FFT. Signals whose Fourier transforms include a relatively small number of heavily weighted frequencies are called sparse. This SFT determines the weights of a signals having heavily weighted frequencies; the sparser the signal, the greater the speedup which the algorithm provides[8]. The researches that identify the most heavily weighted frequency is being proceeded. For k is the sparsity of the signal spectrum and for the typical case of n is a power of 2, the SFT is known to require $O(\log n\sqrt{nk \log n})$ only as the number of needed operations for the l_∞/l_2 guarantee, and the importance of speed-up problem is increased in connection with big data ones. In a word, SFT is a compressed version of DFT and it is using the sparsity of the spectrum of the signal. Of course, this SFT is closely related to the concept of streaming which save run time which is required to read the entire original data set.

In this article, we have checked the intrinsic structure of FFT and a synopsis of the FFT and SFT.

2. The intrinsic structure of fast Fourier transform

To begin with, let us start by an example.

Let us consider the case of the sample values $N = 8$. Then

$$w = e^{-2\pi i/N} = e^{-\pi i/4} = \frac{1}{\sqrt{2}}(1 - i)$$

and so $w^{nk} = \left(\frac{1}{\sqrt{2}}(1-i)\right)^{nk}$. Since the frequency spectrum of the signal f is wrote as $\hat{f} = F_N f$, we have $\hat{f} = F_8 f =$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 \\ 1 & w^2 & w^4 & w^6 & w^8 & w^{10} & w^{12} & w^{14} \\ 1 & w^3 & w^6 & w^9 & w^{12} & w^{15} & w^{18} & w^{21} \\ 1 & w^4 & w^8 & w^{12} & w^{16} & w^{20} & w^{24} & w^{28} \\ 1 & w^5 & w^{10} & w^{15} & w^{20} & w^{25} & w^{30} & w^{35} \\ 1 & w^6 & w^{12} & w^{18} & w^{24} & w^{30} & w^{36} & w^{42} \\ 1 & w^7 & w^{14} & w^{21} & w^{28} & w^{35} & w^{42} & w^{49} \end{pmatrix} f$$

$$= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & -i & -iw & -1 & -w & i & iw \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & -iw & i & w & -1 & iw & -i & -w \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -w & -i & iw & -1 & w & i & -iw \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & iw & i & -w & -1 & -iw & -i & w \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{pmatrix} \qquad (1)$$

It is well-known fact that the given vector $f = [f_0 \cdots f_{N-1}]^T$ is split into two vectors with M components each, namely, f_{even} and f_{odd}. We determine the DFTs

$$\hat{f}_{ev} = [\hat{f}_{ev,0}\ \hat{f}_{ev,2} \cdots \hat{f}_{ev,N-2}]^T = F_M f_{ev}$$

and

$$\hat{f}_{od} = [\hat{f}_{od,1}\ \hat{f}_{od,3} \cdots \hat{f}_{od,N-1}]^T = F_M f_{od}$$

involving the same $M \times M$ matrix F_M[7]. From these things we have the components of the DFT of f by

$$\hat{f}_n = \hat{f}_{ev,n} + {w_N}^n \hat{f}_{od,n} \qquad (2)$$

and

$$\hat{f}_{n+M} = \hat{f}_{ev,n} - {w_N}^n \hat{f}_{od,n} \qquad (3)$$

for $n = 0, \cdots, M-1$.

Example 2.1. From the case of $N = 8$, let us check the validity of the above formulas (2) and (3).

Solution. When $N = 8$, we have $w = w_N = \frac{1}{\sqrt{2}}(1-i)$ and $M = N/2 = 4$. Thus

$w = w_M = e^{-2\pi i/4} = e^{-\pi i/2} = -i$. From the above statements,

$$\hat{f}_{ev} = \begin{pmatrix} \hat{f}_0 \\ \hat{f}_2 \\ \hat{f}_4 \\ \hat{f}_6 \end{pmatrix} = F_4 f_{ev} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \begin{pmatrix} f_0 \\ f_2 \\ f_4 \\ f_6 \end{pmatrix} = \begin{pmatrix} f_0 + f_2 + f_4 + f_6 \\ f_0 - if_2 - f_4 + if_6 \\ f_0 - f_2 + f_4 - f_6 \\ f_0 + if_2 - f_4 - if_6 \end{pmatrix}$$

and similarly, we have

$$\hat{f}_{od} = \begin{pmatrix} \hat{f}_1 \\ \hat{f}_3 \\ \hat{f}_5 \\ \hat{f}_7 \end{pmatrix} = F_4 f_{od} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \begin{pmatrix} f_1 \\ f_3 \\ f_5 \\ f_7 \end{pmatrix} = \begin{pmatrix} f_1 + f_3 + f_5 + f_7 \\ f_1 - if_3 - f_5 + if_7 \\ f_1 - f_3 + f_5 - f_7 \\ f_1 + if_3 - f_5 - if_7 \end{pmatrix}.$$

Using the formulas (2) and (3), we have

$$\hat{f}_0 = \hat{f}_{ev,0} + {w_N}^0 \hat{f}_{od,0} = (f_0 + f_2 + f_4 + f_6) + (f_1 + f_3 + f_5 + f_7)$$

$$\hat{f}_1 = \hat{f}_{ev,1} + {w_N}^1 \hat{f}_{od,1} = (f_0 - if_2 - f_4 + if_6) + w(f_1 - if_3 - f_5 + if_7)$$

$$\ddots \quad \ddots \quad \ddots$$

$$\hat{f}_6 = \hat{f}_{2+4} = \hat{f}_{ev,2} - {w_N}^2 \hat{f}_{od,2} = (f_0 - f_2 + f_4 - f_6) - (-i)(f_1 - f_3 + f_5 - f_7)$$

$$\hat{f}_7 = \hat{f}_{3+4} = \hat{f}_{ev,3} - {w_N}^3 \hat{f}_{od,3} = (f_0 + if_2 - f_4 - if_6) - (-iw)(f_1 + if_3 - f_5 - if_7).$$

This agrees with the equation (1).

3. A synopsis of Sparse Fourier transform

The concept of compress sensing (CS) has aroused many researcher's interest in signal processing field. We are using sample signals in their entire ones either to save space while storing them or to save time while sending them. In the compressing process, we sample only the qualifying parts of the signals, and we call it the signal sparsity. In a word, we can compress data by using the signal sparsity. Even if the given signal is not sparse, we can use the best k-sparse approximation of its Fourier transform. The SFT applies almost same principle as CS to Fourier transform, and computes the significant coefficients in the frequency domain.

Let us see the needed definitions. The frequency spectrum of the signal f is written as $\hat{f} = F_N f$ in section 2, and equivalently, we can write

$$\hat{f}_n = Nc_n = \sum_{k=0}^{N-1} f_k e^{-inx_k},$$

where $f_k = f(x_k)$ and $x_k = 2\pi k/N$. Thus we can write $F_N{}^{-1} = e^{inx_k}$. Since $f = F_N{}^{-1}\hat{f}$, we have

$$f_k = \sum_{k=0}^{N-1} \hat{f}_n e^{inx_k}$$

for $x_k = 2\pi k/N$. In case that a vector f contains a single frequency, we can write $f_k = \hat{f}_n e^{inx_k} = \hat{f}_n e^{in2\pi k/N}$. In this case, let us compute $f_{k+1}/f_k = e^{2\pi nki/N} = \cos\ (2\pi k/N) + i \sin\ (2\pi k/N)$. This requires only 2 entries, and so, faster than the existing FFT [5].

[4] insists that this SFT does not estimating large coefficients, but subtracting them and recursing on the reminder, it identifies and estimates the k largest coefficients in "one shot", in a manner akin to sketching/streaming algorithms. The approximation $\hat{f}'$ of $\hat{f}$ satisfies the following l_2/l_2 guarantee for a function f whose Fourier transform $\hat{f}$:

$$||\hat{f} - \hat{f}'||_2 \le C \min_{k-sparse\ g} ||\hat{f} - g||_2,$$

where C is some approximation factor and the minimization is over k-sparse signals[11]. Recently, the approximation is performed for l_∞/l_2 guarantee:

$$||\hat{f} - \hat{f}'||_\infty^2 \le \epsilon||\hat{f} - g||_2^2/k + \delta||f||_1^2,$$

with probability $1 - 1/n$. It is known that this l_∞/l_2 guarantee is better than l_2/l_2 one in speed-up and efficiency. The detailed techniques are found in [1], [4] and [5].

Next, let us check the computational side. Normally, the SFT hashes the Fourier coefficients of the input signal into a small number of bins, and the vector $\hat{f}'$ computed by the algorithm satisfies $l_2 l_2$ or l_∞/l_2 guarantee: The sum and the weighted sum of the Fourier coefficients stored inside i-th bin are defined as

$$\hat{u_i} = \Sigma_n\ \hat{f}_n,\ \hat{u_i'} = \Sigma_n\ n \cdot \hat{f}_n$$

respectively, for n in i-th bin. If the given signal is sparse, we can make each bin contains just one coefficient (or frequency). Otherwise, if the given signal is not sparse or sparse enough, that is, if each bin contains more than one coefficient, then we have to approximate it to sparse. If $k = O(N)$ and the number of unidentified coefficients is k', then we can remove $k - k'$ from the given signal and we say k'-sparse. Consequently, the identified coefficients $k - k'$ is removed from bins, and the process is repeated to each coefficients are identified.

References

[1] A. Akavia, Deterministic sparse Fourier approximation via fooling arithmetic progressions, *COLT.* (2010), 381–393.

[2] R. Bracewell, The fast Hartley transform, *Proc. IEEE* **72** (1984), 1010–1018.

[3] S. K. Q. Al-Omari and A. Kilicman, On the generalized Hartley and Hartley-Hilbert transformations, *Adv. Diff. Eqs.*, **222** (2013), 1–14.

[4] A. C. Gilbert, P. Indyk and M. Iwen, Recent Ddvelopments in the sparse Fourier transform, *Sig. Proce. Mag., IEEE* **31** (2014), 91–100.

[5] H. Hassanieh, P. Indyk, D. Katabi and E. Price, Simple and practical algorithm for sparse Fourier transform, *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms* (2012).

[6] Hj. Kim and S. Lekcharoen, A cooley-Tukey modified algorithm in fast Fourier transform, *Kor. J. Math.* **19** (2011), 243–253.

[7] E. Kreyszig, *Advanced Engineering Mathematics*, Wiley, Singapore, 2013.

[8] MIT news, *The faster-than-fast Fourier transform*, Jan. 18, 2012, http://news.mit.edu/2012/faster-fourier-transforms-0118.

[9] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer Verlag, 1980.

[10] S. Winograd, On computing the discrete Fourier transform, *Math. Compu.* **32** (1978), 175–199.

[11] L. Ying, Sparse Fourier transform via butterfly algorithm, *SIAM J. Sci. Compu.* **31** (2009), 1678–1694.