On Cooley-Tukey FFT Method for Zero Padded Signals

Khalid Mahmood Aamir[1], Mohammad Ali Maud[2], Asim Loan[3]

[1,3]*Lahore University of Management Sciences, Lahore, Pakistan*
Email: {kmaamir, aloan}@lums.edu.pk
[2]*University of Management and Technology, Lahore, Pakistan*
Email: mamaud@umt.edu.pk

Abstract

The classical Cooley-Tukey fast Fourier transform (FFT) algorithm has the computational cost of $O(N\log_2 N)$ *where* N *is the length of the discrete signal. Spectrum resolution is improved through padding zeros at the tail of the discrete signal. If* $(p-1)N$ *zeros are padded (where* p *is an integer) at the tail of the data sequence, the computational cost through FFT becomes* $O(pN\log_2 pN)$*. This paper proposes an alternate instance of padding zeros to the data sequence that results in computational cost reduction to* $O(pN\log_2 N)$*. It has been noted that this modification can be used to achieve non-uniform upsampling that would zoom-in or zoom-out a particular frequency band. In addition, it may be used for pruning the spectrum, which would reduce resolution of an unimportant frequency band.*

1. Introduction

Suppose we have a discrete time signal $x[n]$ for $0 \le n < N$ and $N = 2^{\alpha}$ where α is an integer. The discrete Fourier transform (DFT) is used to transform the discrete time signal into discrete frequency domain. The classical Cooley-Tukey fast Fourier transform algorithm [1]-[3] for the computation of DFT has the computational cost of $O(N\log_2 N)$. Many techniques that reduce its complexity like higher-radix and split-radix algorithms [4, 5], pruning [6]-[13], block transforms [14], and slide transforms [15] can be employed. Zero padding at the tail of the discrete signal is carried out to improve the frequency resolution. This process is called *spectral interpolation*. Padding $(p-1)N$ zeros when $p>1$ increases the computational cost of the FFT algorithm to $O(pN\log_2 pN)$. The classical FFT algorithm's application to pruning and zooming are limited as non-uniform upsampling is not conveniently applicable.

This paper presents an alternate zero padding scheme to the standard practice of padding zeros at the tail of the data sequence, causing a reduction in the computational cost to $pN\log_2(N)$ of the zero padded signal. The proposed zero padding scheme, can be used for non-uniform upsampling in the frequency domain. This facilitates zooming-in of frequency bands of interest. Similarly, the same algorithm can be adapted for the purpose of pruning.

This paper is organized as follows. The modified zero padding scheme and its application to the FFT algorithm is presented in Section 2. Section 3 describes ways in which the proposed scheme can be used to achieve pruning, zooming and non-uniform upsampling in the frequency domain.

2. Modified FFT Algorithm

2.1 Proposed Modification

The discrete Fourier transform for a complex data sequence of length N is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N} \qquad (1)$$

for $k = 0, \cdots, N-1$

There are many fast Fourier transform algorithms but we will discuss only the so called radix-2 decimation in time (DIT) variant of the FFT. Consider a signal with eight data points with eight zeros padded. Figure 1 shows levels in FFT.

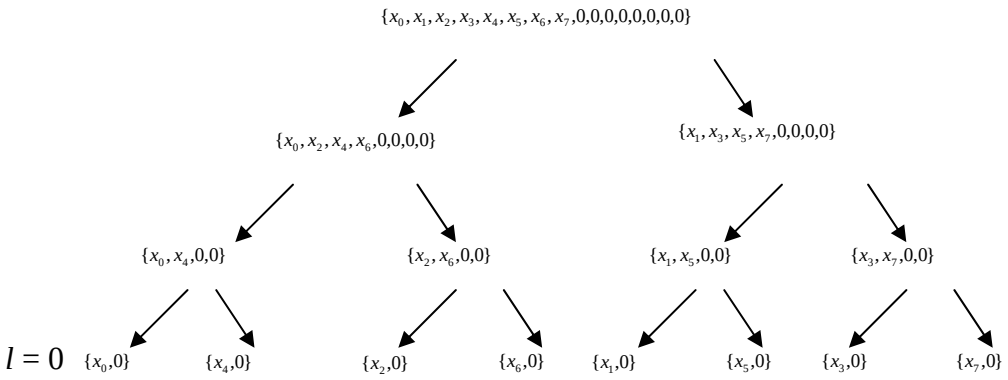

Figure-1: Levels in FFT of an eight point signal.

There are $\log_2(N)$ levels of the butterfly algorithm. Let l be an integer that represents the stage number of FFT algorithm such that $0 \le l < \log_2(N)$. We call a stage as the lowest level stage ($l = 0$) when the dimension of the matrix $\Omega(0)$, representing constants and twiddle factors relating input sequence to output sequence, is minimum (2 × 2 in radix-2 DIT FFT) and the highest level $(l = \log_2[N]-1)$ when the dimensions of the matrix $\Omega(\log_2[N]-1)$ is maximum. The lowest level stage matrix Ω for DIT FFT algorithm is written as:

$$\Omega(0) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (2)$$

The proposal is to pad zeros at the lowest level rather than at the tail of the overall data sequence to achieve upsampling in the frequency domain. Suppose that after zero padding at the lowest level, the number of data points is $2p-1$. This fact is clear from Figure 2.

Equation (1) becomes:

$$X[k] = \sum_{n=0}^{2p-1} x[n] e^{-j\pi nk/p} = \sum_{n=0}^{1} x[n] e^{-j\pi nk/p} \qquad (3)$$
$$k = 0, \cdots, 2p-1$$

The lowest level $\Omega(0)$ in this case is obtained from equation (3). The first column of $\Omega(0)$ is all ones ($n = 0$) and the second column of $\Omega(0)$ is generated by $e^{-j\pi k/p}$ for $0 \le k < 2p$ ($n = 1$). The remaining data points for $n = 2$ to $2p-1$ are all padded zeros. So, $\Omega(0)$ can be written as:

$$\Omega(0) = \begin{bmatrix} 1 & 1 \\ 1 & e^{-j\pi/p} \\ 1 & e^{-j2\pi/p} \\ \vdots & \vdots \\ 1 & e^{-j\pi(2p-1)/p} \end{bmatrix} \qquad (4)$$

As a specific case, if the lowest level sequence is padded with two zeros ($p = 2$), then $\Omega(0)$ will be:

$$\Omega(0) = \begin{bmatrix} 1 & 1 \\ 1 & -j \\ 1 & -1 \\ 1 & j \end{bmatrix} \qquad (5)$$

$\Omega(0)$ is a $2p \times 2$ matrix. Here the lowest level had two data points in the subset. If there are q data points in the lowest level matrix, the dimensions of $\Omega(0)$ will be $qp \times q$ - the generalization is easy and straightforward. $\Omega(0)$ is generated by $e^{-j\pi kq/p}$ for $0 \le k < 2p$ where *k* is an integer.

The generalized mathematical derivation for the suggested modification is shown in Appendix A.

Example-1: Consider the function $x(t)$ as:

$$x(t) = \cos(6\pi t) + \cos(2\pi t) \qquad (6)$$

Signal is sampled at 10 Hz. The signal was 6.3 seconds long. Figure-3 shows the power spectral density (PSD) fo the signal obtained with $\Omega(0)$ of equation (5) and compared with the PSD of the same signal padded with equal number of zeros at the tail of the original signal.

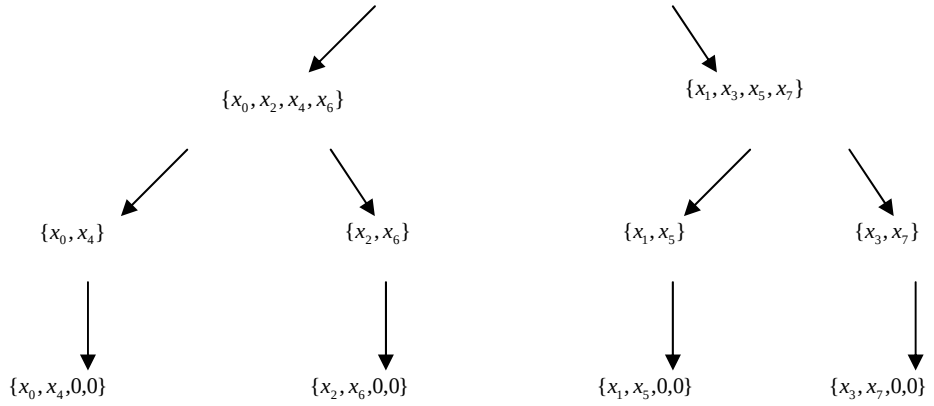

Figure-2: Modified suggested FFT levels.

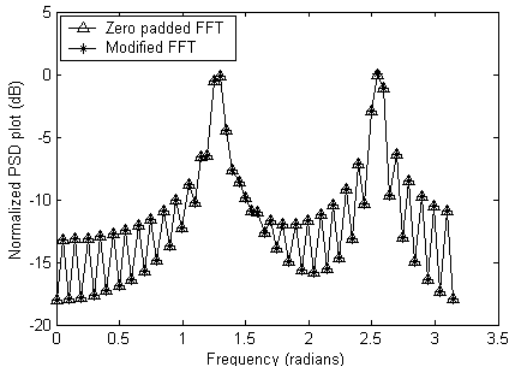

Figure-3: Comparison between conventional and proposed scheme. $\Omega(0)$ is as in (5).

2.2 Computational Cost

The Cooley-Tukey fast Fourier transform algorithm has a complexity/cost of $O(N \log_2 N)$. In fact, there are $\log_2 N$ stages of the FFT algorithm each with $O(N)$ computations. Consider a discrete time signal $x[n]$ where $0 \le n < N = 2^{\alpha}$ and α is a positive integer. For improving the spectral resolution, zeros are appended to the signal. Suppose we pad $(p-1)N$ zeros where p is a positive integer, greater than one. The number of stages in the FFT algorithm is now $\log_2(pN)$. Therefore, the computational cost of the algorithm is $pN \log_2(pN)$ each with $O(pN)$ computations. If the modification suggested in Section 2.1 is implemented where each subset has least possible number of samples then the number of stages would remain unchanged, i.e., $\log_2(N)$ stages with $O(pN)$ computations due to zero padding at this stage. Consequently, zero padding at the lowest stage reduces the computational cost by $O(pN \log[p])$.

3. Conclusions

When we are performing frequency-domain analysis, zoom FFT is useful for zooming in on a narrow frequency band. We can use zoom FFT to focus on a narrowband channel by performing only the calculations needed to obtain the FFT data in the frequency range of interest. For a large data size, we may use pruning to reduce the computational load.

As shown in Section 2, we can expand $\Omega(0)$ for a better spectral estimate. We can expand any $\Omega(l)$ at any stage of the algorithm. Expanding $\Omega(l)$ in a required particular fashion will yield non-uniform

upsampling. This is done when we are interested in spectrum of the signal in a particular frequency band. $\Omega(l)$ is squeezed when we are not interested in some frequency band.

Suppose some of the twiddle factors had very small magnitude, then the corresponding branches of the butterfly operations could be dropped (pruned) to reduce complexity. Thus, we can squeeze $\Omega(l)$ for pruning. Various pruning techniques can still be applied after this modification.

It is obvious that computational cost has been reduced by a factor of $O(pN\log_2[p])$. Moreover, the proposed method can be used for pruning and non-uniform upsampling in the frequency domain.

Acknowledgements

This research work was funded by Higher Education Commission (HEC), Islamabad, Pakistan. Their support is gratefully acknowledged.

References

[1] Z. J. Mou, P Duhamel, "In-place Butterfly-Style FFT of 2D Real Sequences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no.10, Oct. 1988.

[2] A. Fertner, "Computationally efficient methods for analysis and synthesis of real signals using FFT and IFFT," *IEEE Trans. Signal Processing*, vol. 47, pp. 1061-1064, April 1999.

[3] J. G. Proakis and D. G. Manolakis, "Digital Signal Processing, Principles, Algorithms and Applications," 3rd Edition, Prentice-Hall Inc., 1996.

[4] P. Duhamel, "Implementation of "Split-radix" FFT algorithms for complex, real, and real-symmetric data," *IEEE Trans. on ASSP*, vol. 34, pp. 285-295, April 1986.

[5] H. V. Sorensen, M. T. Heideman, C. S. Burrus, "On Computing the Split radix FFT," *ICASSP-85 Proceedings*, March 1985.

[6] H. V. Sorenson and C. S. Burrus, "Efficient computation of the DFT with only a subset of input or output points," *IEEE Trans. Signal Processing*, vol. 41, no. 3, pp. 1184–1199, Mar. 1993.

[7] D. P. Skinner, "Pruning the decimation-in-time FFT algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, no. 4, pp. 193–194, Apr. 1976.

[8] S. B. Narayanan and K. M. M. Prabhu, "Fast Hartley transform pruning," *IEEE Trans. Signal Processing*, vol. 39, no. 1, pp. 230–233, Jan. 1991.

[9] S. Barash, Y. Ritov, "Logarithmic pruning of FFT frequencies," *IEEE Trans. Signal Process.*, vol. 41, no. 3, 1398-1400, 1993.

[10] S.R. Rangarajan, S. Srinivasan, "Generalized Method for Pruning an FFT Type of Transform," *VISP(144)*, No. 4, pp. 189-192, August 1997.

[11] H. Huang, Y. Lee, P. Lo, "A novel algorithm for computing the 2D split-vector-radix FFT," *Signal Processing*, vol. 84, no. 3, March 2004.

[12] S. Franz, S. K. Mitra, "Gerhard Doblinger, Frequency estimation using warped discrete Fourier transform," *Signal Processing*, vol. 83, no. 8, August 2003.

[13] S. Barasch and Y. Ritov, "Pruning FFT frequencies," *IEEE transactions on Signal Processing*, vol. 41, pp. 1398-1400, 1993.

[14] G. P. M. Egelmeers and P. C. W. Sommen, "Recursive computation of block-FFT's," *Proc. ProRISC/IEEE Symp. Circuits, Systems Signal Processing*, Netherlands, pp. 59–66, Mar. 1995.

[15] T. Springer, "Sliding FFT computes frequency spectra in real time," *EDN*, pp. 161–170, Sep. 1988.

Appendix – A

Consider $a_n = \{x_0, x_1, x_2, \cdots, x_7\}$

$$\sum_{n=0}^{7} a_n e^{-j\frac{2\pi}{8}nk} = \sum_{n=0}^{3} a_{2n} e^{-j\frac{2\pi}{8}2nk} + e^{-j\frac{\pi}{4}k} \sum_{n=0}^{3} a_{2n+1} e^{-j\frac{2\pi}{8}2nk}$$

$$= \sum_{n=0}^{3} a_{2n} e^{-j\frac{2\pi}{4}nk} + e^{-j\frac{\pi}{4}k} \sum_{n=0}^{3} a_{2n+1} e^{-j\frac{2\pi}{4}nk}$$

Let $a_{2n} = b_n$ and $a_{2n+1} = c_n$

$$= \sum_{n=0}^{3} b_n e^{-j\frac{2\pi}{4}nk} + e^{-j\frac{\pi}{4}k} \sum_{n=0}^{3} c_n e^{-j\frac{2\pi}{4}nk}$$

$$= \left[\sum_{n=0}^{1} b_{2n} e^{-j\frac{2\pi}{4}2nk} + e^{-j\frac{\pi}{2}k} \sum_{n=0}^{1} b_{2n+1} e^{-j\frac{2\pi}{4}2nk} \right] + e^{-j\frac{\pi}{4}k} \left[\sum_{n=0}^{1} c_{2n} e^{-j\frac{2\pi}{4}2nk} + e^{-j\frac{\pi}{2}k} \sum_{n=0}^{1} c_{2n+1} e^{-j\frac{2\pi}{4}2nk} \right]$$

$$= \left[\sum_{n=0}^{1} b_{2n} e^{-j\frac{2\pi}{2}nk} + e^{-j\frac{\pi}{2}k} \sum_{n=0}^{1} b_{2n+1} e^{-j\frac{2\pi}{2}nk} \right] + e^{-j\frac{\pi}{4}k} \left[\sum_{n=0}^{1} c_{2n} e^{-j\frac{2\pi}{2}nk} + e^{-j\frac{\pi}{2}k} \sum_{n=0}^{1} c_{2n+1} e^{-j\frac{2\pi}{2}nk} \right] \quad \text{(A-1)}$$

Now we suppose that two zeros are padded. So changing the summands with four point DFT and also changing the twiddle factors in equation (A-1),

$$\left[\sum_{n=0}^{1} b_{2n} e^{-j\frac{2\pi}{4}nk} + e^{-j\frac{\pi}{4}k} \sum_{n=0}^{1} b_{2n+1} e^{-j\frac{2\pi}{4}nk} \right] + e^{-j\frac{\pi}{8}k} \left[\sum_{n=0}^{1} c_{2n} e^{-j\frac{2\pi}{4}nk} + e^{-j\frac{\pi}{4}k} \sum_{n=0}^{1} c_{2n+1} e^{-j\frac{2\pi}{4}nk} \right]$$

$$= \sum_{n=0}^{3} b_n e^{-j\frac{2\pi}{8}nk} + e^{-j\frac{\pi}{8}k} \sum_{n=0}^{3} c_n e^{-j\frac{2\pi}{8}nk}$$

$$= \sum_{n=0}^{3} a_{2n} e^{-j\frac{2\pi}{8}nk} + e^{-j\frac{\pi}{8}k} \sum_{n=0}^{3} a_{2n+1} e^{-j\frac{2\pi}{8}nk} \quad \text{(A-2)}$$

$$= \sum_{n=0}^{7} a_n e^{-j\frac{2\pi}{16}nk}$$

This is DFT of eight data point DFT with eight zeros padded at the end. We generalize the results of equation (A-2) as:

$$\sum_{n=0}^{N-1} a_n e^{-j\frac{2\pi}{Np}nk} = \sum_{n=0}^{N/2-1} a_{even} e^{-j\frac{2\pi}{Np/2}nk} + e^{-j\frac{2\pi}{Np}k} \sum_{n=0}^{N/2-1} a_{odd} e^{-j\frac{2\pi}{Np/2}nk}$$

Where $a_n = \{x_0, x_1, x_2, \cdots, x_{N-1}\}$. This relationship is used recursively.