WAVELET TRANSFORM BASED FAST APPROXIMATE FOURIER TRANSFORM

Haitao Guo and C. Sidney Burrus

Electrical and Computer Engineering Department
Rice University
Houston, TX 77005

ABSTRACT

We propose an algorithm that uses the discrete wavelet transform (DWT) as a tool to compute the discrete Fourier transform (DFT). The Cooley-Tukey FFT is shown to be a special case of the proposed algorithm when the wavelets in use are trivial. If no intermediate coefficients are dropped and no approximations are made, the proposed algorithm computes the exact result, and its computational complexity is on the same order of the FFT, i.e. $O(N \log_2 N)$. The main advantage of the proposed algorithm is that the good time and frequency localization of wavelets can be exploited to approximate the Fourier transform for many classes of signals resulting in much less computation. Thus the new algorithm provides an efficient complexity v.s. accuracy tradeoff. When approximations are allowed, under certain sparsity conditions, the algorithm can achieve linear complexity, i.e. $O(N)$. The proposed algorithm also has built-in noise reduction capability.

1. INTRODUCTION

The discrete Fourier transform (DFT) is probably the most important computational tool in signal processing. Because of the characteristics of the basis functions, the DFT has enormous capacity for the improvement of its arithmetic efficiency [1]. The classical Cooley-Tukey fast Fourier transform (FFT) algorithm has the complexity of $O(N \log_2 N)$. The Fourier transform is related to many physical problems, and can not simply be replaced by other transforms, e.g. in seismic wave propagation and speech production. Thus the Fourier transform and its fast algorithm – the FFT are widely used in many areas including signal processing and numerical analysis. Any scheme to speed up the FFT would be very desirable.

Although the FFT has been studied extensively, there are still some desired properties that are not provided by the classical FFT. First of all, pruning is not easy. When the number of input points or output points are small comparing to the length of the DFT, a special technique called *pruning* [2] is often used. However, it is often required that those non-zero input data are grouped together. Classical FFT pruning algorithms does not work well when the few non-zero inputs are randomly located. In other words, a sparse signal does not give rise to a faster algorithm. Also, there is no speed v.s. accuracy tradeoff. In not so rare situations, it is desirable to allow some error in order to gain the speed. However, this is not so easy in the classical FFT algorithm. One of the main reasons is that the twiddle factors in the butterfly operations are unit magnitude complex numbers. So all parts of the FFT structure are of equal importance. It is hard to decide which part of the FFT structure to omit when error is allowed and the speed is crucial. In other words, the FFT is a single speed and single accuracy algorithm. Finally, The classical FFT does not have built-in noise reduction capability. Many real world signals are noisy. What people are really interested in are the DFT of the signals without the noise. Even if other denoising algorithms are used, the FFT requires the same computational complexity on the denoised signal. The wavelet transform is a powerful new mathematical tool. The discrete wavelets transform (DWT) is fast — linear complexity ($O(N)$). Wavelets are unconditional basis for many signal spaces [3], so the wavelet coefficients are maximally sparse. In the frequency domain, wavelets are related to the constant Q filter banks. The good time and frequency localization of wavelets could be exploited to approximate the Fourier transform.

The rest of the paper is organized as follows. Section 2 presents the preliminary results about the FFT and DWT. The fundamental structures and the necessary notations are introduced. The main result is presented in Section 3, where we develop the algorithm that uses the DWT to compute the discrete Fourier transform. The computational complexity is also investigated. While Section 3 deals with exact computation, Section 4 studies various ways to speed up the algorithm using approximations. In Section 5, we briefly discuss the built-in denoising capacity of our algorithm. Finally, we summarize our findings in Section 6.

2. PRELIMINARIES

Review of the Discrete Fourier Transform and FFT

The discrete Fourier transform (DFT) is defined for a length-N complex data sequence by

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi nk/N}, \quad k = 0, \ldots, N-1. \qquad (1)$$

There are several ways to derive the different fast Fourier transform (FFT) algorithms. It can be done using index

This work was supported in part by ARPA, BNR and TI.

mapping [1], by matrix factorization, or by polynomial factorization. In this paper, we only discuss the matrix factorization approach, and only discuss the so called *radix-2 decimation in time* (DIT) variant of the FFT.

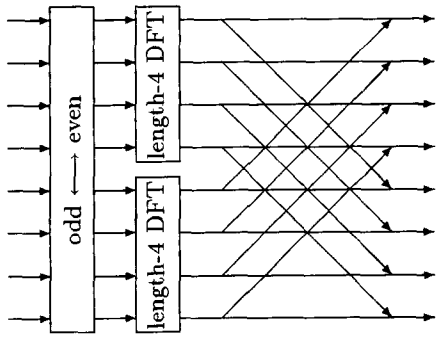


Figure 1: Last stage of a length-8 radix-2 DIT FFT.

In stead of repeating the derivation of the FFT algorithm. We show the block diagram and matrix factorization, in an effort to highlight the basic idea and gain some insight. The block diagram of the last stage of a length-8 radix-2 DIT FFT is shown in Figure 1. First, the input data are separated into even and odd groups. Then, each group goes through a length-4 DFT block. Finally, *butterfly operations* are used to combine the shorter DFTs into longer DFT.

The detail of the *butterfly operations* is shown in Figure 4(a), where $W_N^i = e^{-j2\pi i/N}$ is called the *twiddle factor*. All the twiddle factors are of magnitude one, i.e. on the unit circle. This is one of the main reasons that there is no complexity v.s. accuracy tradeoff for the classical FFT. 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 while minimizing the error to be introduced. Of course the error depends on the value of the data to be multiplied with the twiddle factors. When the value of the data is unknown, the best way is nevertheless to cutoff the branches with small twiddle factors.

The computational complexity of the FFT algorithm can be easily established. Let $C_{FFT}(N)$ be the complexity for a length-N FFT, we can show

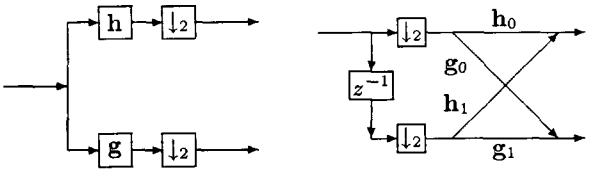
$$C_{FFT}(N) = O(N) + 2C_{FFT}(N/2), \tag{2}$$

where $O(N)$ denotes linear complexity. The solution to Equation 2 is well known,

$$C_{FFT}(N) = O(N \log_2 N). \tag{3}$$

This is a classical case where the *divide and conquer* approach results in very effective solution.

The matrix point of view gives us additional insight. Let $\mathbf{F}_N$ be the $N \times N$ DFT matrix, i.e. $\mathbf{F}_N(m,n) = e^{-j2\pi mn/N}$,



(a) Direct form. (b) Polyphase form.

Figure 2: Building block for the discrete wavelet transform.

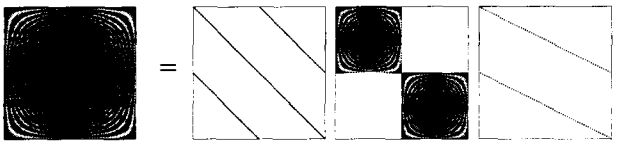
where $m,n \in \{0,1,\ldots,N-1\}$. Let $\mathbf{S}_N$ be the $N \times N$ even-odd separation matrix, e.g.

$$\mathbf{S}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

Clearly $\mathbf{S}_N'\mathbf{S}_N = \mathbf{I}_N$, where $\mathbf{I}_N$ is the $N \times N$ identity matrix. Then the DIT FFT is based on the following matrix factorization,

$$\mathbf{F}_N = \begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{T}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{T}_{N/2} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{N/2} & 0 \\ 0 & \mathbf{F}_{N/2} \end{bmatrix} \mathbf{S}_N, \tag{5}$$

where $\mathbf{T}_{N/2}$ is diagonal matrix with W_N^i, $i \in \{0,1,\ldots,N/2-1\}$ on the diagonal. We can visualize the above factorization as



where we image the real part of DFT matrices, and the magnitude of the matrices for butterfly operations and even-odd separations. N is taken to be 128 here.

Review of the Discrete Wavelet Transform

At the heart of the discrete wavelet transform are a pair of filters **h** and **g** – lowpass and highpass respectively. They have to satisfy a set of constrains [4, 5]. The building block of the DWT is shown in Figure 2(a). The input data are first filtered by **h** and **g** then downsampled. The same building block is further iterated on the lowpass outputs.

The computational complexity of the DWT algorithm can also be easily established. Let $C_{DWT}(N)$ be the complexity for a length-N DWT. Since after each scale, we only further operate on half of the output data, we can show

$$C_{DWT}(N) = O(N) + C_{DWT}(N/2), \tag{6}$$

which give rise to the solution

$$C_{DWT}(N) = O(N). \tag{7}$$

The operation in Figure 2 can also be expressed in matrix form $\mathbf{W}_N$, e.g. for Haar wavelet,

$$\mathbf{W}_4^{Haar} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}. \tag{8}$$

The orthogonality conditions on h and g ensure $\mathbf{W}_N'\mathbf{W}_N = \mathbf{I}_N$. The matrix for multiscale DWT is formed by $\mathbf{W}_N$ for different N. We could further iterate the building block on some of the highpass outputs. This generalization is called the wavelet packets [6].

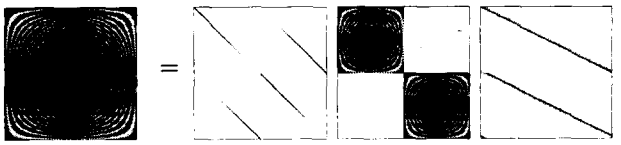
3. FAST FOURIER TRANSFORM VIA DISCRETE WAVELET TRANSFORM

The Algorithm Development

The key to the fast Fourier transform is the factorization of $\mathbf{F}_N$ into several sparse matrices, and one of the sparse matrices represents two DFTs of half the length. Similarl to the DIT FFT, the following matrix factorization has been found,

$$\mathbf{F}_N = \begin{bmatrix} \mathbf{A}_{N/2} & \mathbf{B}_{N/2} \\ \mathbf{C}_{N/2} & \mathbf{D}_{N/2} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{N/2} & 0 \\ 0 & \mathbf{F}_{N/2} \end{bmatrix} \mathbf{W}_N, \quad (9)$$

where $\mathbf{A}_{N/2}, \mathbf{B}_{N/2}, \mathbf{C}_{N/2}$, and $\mathbf{D}_{N/2}$ are all diagonal matrices. The values on the diagonal of $\mathbf{A}_{N/2}$ and $\mathbf{C}_{N/2}$ are the length-N DFT of h, and the values on the diagonal of $\mathbf{B}_{N/2}$ and $\mathbf{D}_{N/2}$ are the length-N DFT of g. We can visualize the above factorization as



where we image the real part of DFT matrices, and the magnitude of the matrices for butterfly operations and the one-scale DWT using length-16 Daubechies' wavelets [3]. Clearly we can see that the twiddle factors have non-unit magnitudes.

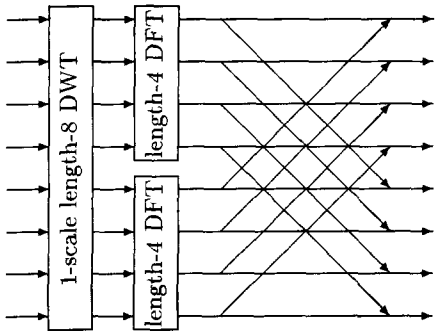


Figure 3: Last stage of a length-8 DWT based FFT.

The above factorization suggests a DWT based FFT algorithm. The block diagram of the last stage of a length-8 algorithm is shown in Figure 3. Following the length-8 DWT, the highpass and the lowpass DWT outputs go through separate length-4 DFT, then they are combined with butterfly operations. Same scheme in Figure 3 are iteratively applied to shorter length DFTs to get the full

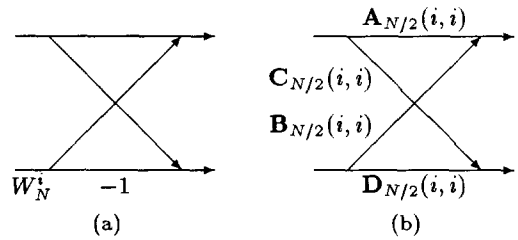


Figure 4: Butterfly operations (a) in a radix-2 DIT FFT; (b) in a wavelet transform based FFT.

DWT based FFT algorithm. The final system is equivalent to a full binary tree wavelet packets transform [6] followed by modified FFT butterfly operations, where the twiddle factors are the frequency response of the wavelet filters. The detail of the butterfly operation is shown in Figure 4(b), where $i \in \{0, 1, \ldots, N/2-1\}$. Now the twiddle factors are length-N DFT of h and g.

The classical radix-2 DIT FFT is a special case of the above algorithm when h = [1, 0] and g = [0, 1]. This can be seen using the polyphase form as in Figure 2(b). Although they do not satisfy some of the conditions required for wavelets, they do constitute a legitimate (and trivial) orthogonal filter bank.

Computational Complexity

For the DWT based FFT algorithm, the computational complexity is also $O(N \log_2 N)$, since the recursive relation in Equation 2 is again satisfied. However, the constant appears before $N \log_2 N$ depends on the wavelet filters used.

4. FAST APPROXIMATE FOURIER TRANSFORM

Basic Idea

The basic idea of the fast approximate Fourier transform (FAFT) is *pruning*, i.e. cut off part of the diagram. Traditionally, when only part of the inputs are non-zero, or only part of the outputs are required, the part of the FFT diagram where either the inputs are zero or the outputs are undesired is pruned [2], so that the computational complexity is reduced. However, the classical pruning algorithm is quite restrictive, since for majority of the applications, both the inputs and the outputs are of full length.

The structure of the DWT based FFT algorithm can be exploited to generalize the classical pruning idea for arbitrary signals. From the input data side, the signals are made sparse by the wavelet transform [3], thus approximation can be made to speed up the algorithm by *dropping* the insignificant data. In other words, although the input signal are normally not sparse, DWT creates the sparse inputs for the butterfly stages of the FFT. So any scheme to prune the butterfly stages for the classical FFT can be used here. Of course, the price we have to pay here is the computational complexity of the DWT operations. In actual implementation, the wavelets in use have to be carefully chosen to balance the benefit of the pruning and the price

of the transform. Clearly, the optimal choice depends on the class of the data we would encounter.

From the transform side, since the twiddle factors of the new algorithm have decreasing magnitudes, approximation can be made to speed up the algorithm by *pruning* the sections of the algorithm which correspond to the insignificant twiddle factors. For well defined wavelet filters, they have well known properties, e.g for Daubechies' family of wavelets, their frequency responses are monotone, and nearly half of which have magnitude close to zero. It should be noted that those filters are not designed for frequency responses. They are designed for flatness at 0 and π. Various methods can be used to design wavelets or orthogonal filter banks [7, 8, 5, 4] to achieve better frequency responses. Again, there is a tradeoff between the good frequency response of the longer filters and the higher complexity required by the longer filters.

Computational Complexity

The wavelet coefficients are mostly sparse, so the input of the shorter DFTs are sparse. If the implementation scales well with respect to the percentage of the significant input, e.g. it uses half of the time if only half of the inputs are significant, then we can further lower the complexity. Assume for N inputs, αN of them are significant ($\alpha \leq 1$), we have

$$C_{FAFT}(N) = O(N) + 2\alpha C_{FAFT}(N/2). \qquad (10)$$

For example if $\alpha = \frac{1}{2}$, Equation (10) simplifies to

$$C_{FAFT}(N) = O(N) + C_{FAFT}(N/2), \qquad (11)$$

which leads to

$$C_{FAFT}(N) = O(N). \qquad (12)$$

So under above conditions, we have a linear complexity approximate FFT. Of course, the complexity depends on the input data, the wavelets we use, the threshold value used to drop insignificant data, and the threshold value used to prune the butterfly operations. Good tradeoff need to be found. Also the implementation would be more complicated than the classical FFT.

5. NOISE REDUCTION CAPACITY

It has been shown that the thresholding of wavelet coefficient has near optimal noise reduction property for many classes of signals [9]. The thresholding scheme used in the approximation in the proposed FAFT algorithm is the exactly the hard thresholding scheme used to denoise the data. Soft thresholding can also be easily embedded in the FAFT. Thus the proposed algorithm also reduces the noise while doing approximation. If we need to compute the DFT of noisy signals, the proposed algorithm not only can reduce the numerical complexity, but also can produce cleaner results.

6. SUMMARY

In the past, FFT has been used to calculate DWT [4, 5], which leads to efficient algorithm when filters are infinite impulse response (IIR). In this paper, we did just the opposite – using DWT to calculate FFT. We have shown that when no intermediate coefficients are dropped and no approximations are made, the proposed algorithm computes the exact result, and its computational complexity is on the same order of the FFT, i.e. $O(N \log_2 N)$. The advantage of our algorithm is two fold. From the input data side, the signals are made sparse by the wavelet transform, thus approximation can be made to speed up the algorithm by *dropping* the insignificant data. From the transform side, since the twiddle factors of the new algorithm have decreasing magnitudes, approximation can be made to speed up the algorithm by *pruning* the section of the algorithm which corresponds to the insignificant twiddle factors.

In summary, we proposed a fast approximate Fourier transform algorithm using the wavelet transform. Since wavelets are the conditional basis of many classes of signals [3] the algorithm is very efficient and has built-in denoising capacity.

Some of our preliminary results have been presented in [10]. A more detailed and accessible treatment of this subject and other aspects of the theory and application of wavelet and wavelet transform will appear in [11].

7. REFERENCES

[1] C. S. Burrus and T. W. Parks. *DFT/FFT and Convolution Algorithms.* John Wiley & Sons, New York, 1985.

[2] H. V. Sorensen and C. S. Burrus. Efficient computation of the DFT with only a subset of input or output points. *IEEE Transactions on Signal Processing*, 41(3):1184–1200, March 1993.

[3] I. Daubechies. *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA, 1992. Notes from the 1990 CBMS-NSF Conference on Wavelets and Applications at Lowell, MA.

[4] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding.* Prentice Hall, Englewood Cliffs, NJ, 1995.

[5] P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[6] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Trans. Inform. Theory*, 38(2):1713–1716, 1992.

[7] J. E. Odegard. *Moments, smoothness and optimization of wavelet systems.* PhD thesis, Rice University, Houston, TX 77251, USA, May 1996.

[8] I. W. Selesnick. *New Techniques for Digital Filter Design.* PhD thesis, Rice University, 1996.

[9] D. L. Donoho. De-noising by soft-thresholding. *IEEE Trans. Inform. Theory*, 41(3):613–627, May 1995.

[10] H. Guo and C. S. Burrus. Fast approximate Fourier transform via wavelets transform. In *SPIE Math. Imaging: Wavelet Applications in Signal and Image Processing*, volume 2825, Denver, CO, August 1996.

[11] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and the Wavelet Transform.* Prentice Hall, Englewood Cliffs, NJ, 1997.