# AN ALGORITHM FOR EFFICIENT COMPUTATION OF THE FAST FOURIER TRANSFORM OVER ARBITRARY FREQUENCY INTERVALS

Steve DaBell
Motorola GSTG
Communications Division
Scottsdale, AZ
email: P26109@email.mot.com
June 25, 1992

## ABSTRACT

In many signal processing and telemetry applications only a portion of the Discrete Fourier Transform (DFT) of a data sequence is of interest. This paper develops an algorithm which enables computation of the FFT only over the frequency values of interest, reducing the computational complexity. As will be shown, the algorithm is also very modular which lends to efficient parallel processing implementation. This paper will begin by developing the frequency selective FFT algorithm, and conclude with a comparative analysis of the computational complexity of the algorithm with respect to the traditional FFT

## KEY WORDS

Fast Fourier Transform, Efficient Computation, Arbitrary Frequency Intervals.

## INTRODUCTION

Since the original FFT algorithm was developed in the late sixties, many alternate algorithms have been proposed. In practice; however, the most efficient FFT algorithm depends on the data type and application. This paper develops an algorithm which optimizes the FFT such that the transform only at the frequencies of interest is calculated. This algorithm may result in a reduction of computational complexity. Additionally, this approach has many applications in spectrum analysis and signal processing in general.

## DESCRIPTION OF ALGORITHM

Efficient frequency selective computation of the FFT can be accomplished through the use of the time shifting and superposition properties of the Discrete Fourier Transform (DFT). As a review, the equation for computing the DFT is,

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

Additionally , the time shifting property states that,

$$x(n-m) = e^{(-j2Bm/N)} x(k) \qquad (2)$$

where k is the frequency index , x(k) is the FFT of x(n), and N is the length of the data sequence x(n).

The superposition property states that ,

$$\mathrm{DFT}(x(n) + y(n) + z(n)) = \\ \mathrm{DFT}(x(n)) + \mathrm{DFT}(y(n)) + \mathrm{DFT}(z(n)) \qquad (3)$$

The basic concept of the algorithm is to divide the original data sequence x(n) of length N into c sub sequences each of length N/c as demonstrated below (the optimal value of c will be computed later in the paper):

Original_sequence: x(0), x(1), x( 2)....x(N - 1)

Divided_into_c_subsequences:

for_m = 0,

$x_0(n) = x(n\times c) = x(0), x(c), x(2c),....x(N-c)$

for_m = 1,

$x_1(n) = x(n\times c+1) = x(1),x(c+1),x(2c+1),....x(N-c+1)$

for_m = 2,

$x_2(n) = x(n\times c+2) = x(2), x(c+2), x(2c+2),....., x(N-c+2)$

.

.

.

for_m = c-1,

$x_{c-1}(n) = x(n\times c+m) = x(c-1),x(2c-1),x(3c-1),....,x(N-1)$

The N/c point FFT for each of the above sequences is calculated using the most efficient FFT algorithm for the given data type, sequence length and hardware architecture. The resulting FFT for each of the c sequences $X_0(k)$ to $X_{c-1}(k)$ is periodic with a period N/c.

To calculate the FFT of the original data sequence x(n) at a particular value of k, the time shifting which was utilized in creating $x_0(n)$ to $x_{c-1}(n)$ must be compensated for by multiplying each sequence $X_0(k)$ to $X_{c-1}(k)$ by a "twiddle factor" $e^{(-j2\pi km/N)}$ over the k values of interest. Finally, the value of the FFT at a particular index k is obtained by summing the c sequences (superposition property) at each value of k:

$$X(k) = \sum_{a=0}^{c-1} X_a(k \bmod (N/c)) \qquad (4)$$

In summary, the steps described in the previous paragraph are captured in the following equation:

$$X(k) = \sum_{a=0}^{c-1} e^{(-j2\pi ka/N)} X_a(k \bmod (N/c)) \qquad (5)$$

The optimal number of sub sequences c which the original sequence x(n) is divided into can be derived by differentiating the equation describing the computational complexity of the algorithm with respect to c. For simplicity in developing the relationship, only the number of complex mulitiplications will be considered. Thus, as can be seen from the above development, the complex multiplicative complexity of the algorithm is:

$$c(N/c)\log_2(N/c)+c\Delta k \qquad (6)$$

The first term in the above equation represents the complex multiplication required to compute c N/c point FFTs, and the second term represents the multiplication required to sum the c FFT results for the total number of frequency indexes Δk. Finally, to compute the optimal value of c which minimizes the computational complexity, the above equation can be differentiated with respect to c:

$$d(c(N/c)\log_2(N/c)+c\Delta k)/dc =$$
$$(N/\ln(2))(d(\ln(N/c))/dc)+\Delta k =$$
$$(N/\ln(2))(\frac{-cN}{c^2N})+\Delta k = 0$$
$$\textit{Thus,} \qquad (7)$$
$$c=\frac{N}{\ln(2)\Delta k}$$

The value of c obtained from the above equation should either be rounded up or down to the nearest power of 2. Now to verify the above equation is correct, range checking can be performed for the minimum and maximum cases: if all k values of the DFT are of interest, the value of c would be l/ln(2) which simply implies c equals 1, and the standard FFT would be the optimal approach. However, if the value of the DFT at only one point is of interest, then the optimal value of c would be N/ln(2)=N which implies it is most efficient to use the basic DFT equation (1) to compute the value. Obviously, in most applications Δk will result in a situation which is nontrivial and the algorithm presented in this paper may be used to determine the optimal approach for minimizing the computational complexity.

## COMPUTATIONAL COMPLEXITY ANALYSIS

To demonstrate the computational complexity relationship between the FFT and the frequency selective DFT algorithms, it is most convenient to graph the required number of multiplications and additions as a function of data sequence length and the number of frequencies indexes for which the spectrum is desired. Assuming that complex multiplications require 4 real multiplications and complex additions require 2 real additions, it can easily be shown that the FFT will require 4Nlog(N) real multiplications, and Nlog(N) additions. On most RISC and DSP devices multiplications and additions can require only one clock stroke to complete, thus a total of 5Nlog(N) real computations are required to compute the N point FFT over all frequency indexes.

The computational complexity of the frequency selective transform is calculated in a similar manner. For the c N/c point FFTs a total of 5Nlog(N/c) real computations are required to compute the c N/c point FFTs, and a total of 2cΔk real additions are required to compensate for time shifting and superposition of the individual sequences, over the frequencies of interest. The computational complexity of the FFT vs. the frequency selective FFT algorithm are shown in figure 1-1 where the horizontal represents the normalized frequency interval of interest and the vertical represents the computational complexity.

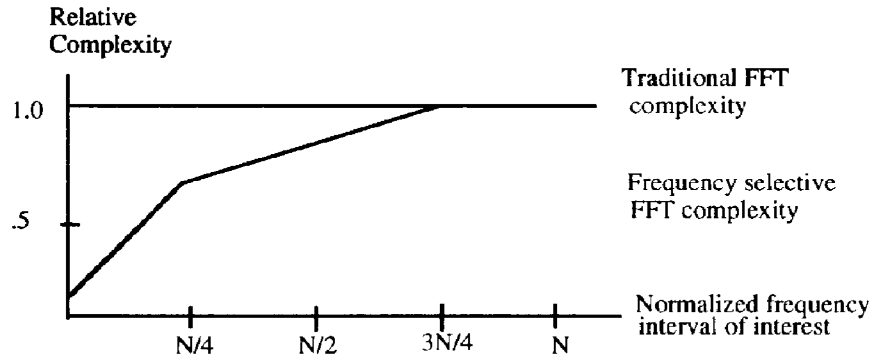

Figure 1-1, Comparison of computational complexity

As can be seen from the figure, the frequency selective FFT algorithm yields a complexity which is less than that of the traditional FFT when only a portion of the spectrum is of interest. Additionally, it can be seen that the complexity of the algorithm converges to that of the FFT as the number of frequency indexes increase. In summary, the frequency selective FFT could be viewed as a generalization of the FFT which optimizes the implementation for a given frequency interval of interest.

## CONCLUSION

In conclusion, an algorithm for computing the FFT of data sequences at arbitrary frequencies has been developed. The algorithm has a lower computational complexity that the FFT if only portions of the spectrum are of interest. Additionally, the algorithm is very modular which lends to efficient implementation in parallel processing architecture.

## ACKNOWLEDGMENTS

Thanks to Jim Rieger PE/PTBW, Naval Weapons Center, China Lake, CA. for many contributions.

## References

[1] Oran E. Brigham, "Fourier Transform Properties, "The Fast Fourier Transform and its Applications, First Ed., Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[2] Alan Oppenheim, "Computation of the Discrete Fourier Transform," Discrete-Time Signal Processing, First Ed., Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[3] J. W. Cooley, Lewis, and P. D. Welch, "The Fast Fourier transform and Its Applications," IEEE Trans. Educ., Vol. E-12, No.1, pp. 27-34, March 1969.

[4] Nagai K., "Pruning the Decimation-in-time FFT Algorithm with Frequency -shift," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol 34, No. 4, pp. 1008-10, 1986.