Computers ind. Engng Vol. 21, Nos 1–4, pp. 653–657, 1991
Printed in Great Britain.

0360-8352/91 $3.00 + 0.00

PREPROCESSING METHODS IN THE COMPUTATION OF THE FAST FOURIER TRANSFORM

Chao-Yen Wu and A. Terry Bahill

Division of Industrial Engineering
Alfred University
Alfred, NY 14802

ABSTRACT

In this paper, two preprocessing methods are presented: the Hamming window and the flip and reverse in time methods, to process the incoming signals before the FFT routine. The reason for doing this preprocessing adjustments is that the FFT routine requires the input signals start and end at the same level. With the Hamming window method, the original signal is multiplied it by the Hamming window function. With the flip and reverse in time method, simply flip and reverse the original signal. The saccadic eye movements were used in this research. These signals do not start and end at the same level. The results show that the flip and reverse technique is better than the Hamming window for saccadic eye movement data. The Hamming window technique multiplies the original signal with a cosine wave. Multiplication in the time domain is equivalent to convolution in the frequency domain, which would be the same as filtering or averaging in the time domain. Therefore, FFT produced with the Hamming window is smoother than the one produced with the flip and reverse method.

1 Introduction

The fast Fourier transform (FFT) is an algorithm for the computation of Fourier coefficients that reduces the computational complexity substantially from the conventional method, was first reported by Cooley and Tukey [1] in 1965. This method is now widely known as the "fast Fourier transform", and has produced dramatic change in computational techniques used in digital computing. The history of this technique has been summarized by Cooley, Lewis, and Welch in [2].

The fast Fourier transform is a computational tool, which facilitates signal analysis such as power spectrum analysis and filter simulation by digital computers. It is a technique for efficiently computing the discrete Fourier transform (DFT) of a series of data samples (referred to as a time series) [3]. In this paper, the discrete Fourier transform of a time series is defined and the associated fast method (fast Fourier transform) for computing this transform is derived. A general purpose software program, which computes the FFT for variety of signals such as the saccadic eye movements, ECG, sinusoidal, cubic, parabolic, and triangular signals, is also developed. Two preprocessing methods: the Hamming window and the flip and reverse in time methods, are used to process the incoming signals before the FFT routine if these incoming signals do not start and end at the same level.

2 Definition Of The DFT And Its Inverse

Since the FFT is an efficient technique for computing the DFT it is appropriate to begin by discussing the DFT and its inverse. The DFT is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j(2\pi/N)kn}\ ,\ k = 0, 1, \ldots, N-1, \tag{1}$$

where $X(k)$ is the k-th coefficient of the DFT, $x(n)$ denotes the n-th sample of the time series that consists of N samples, and j is the imaginary unit. The $x(n)$'s can be complex numbers and the $X(k)$'s are almost always complex. For notational convenience (1) is often written as

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} \ , \ k = 0, 1, ..., N-1, \tag{2}$$

where

$$W_N^{nk} = e^{-j(2\pi/N)kn}. \tag{3}$$

Since the $x(n)$'s are often values of a function at discrete time points (in time-domain), the index k is sometimes called the "frequency" of the DFT (in frequency-domain). The inverse discrete Fourier transform (IDFT) is

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, W_N^{-nk} \ , \ n = 0, 1, ..., N-1. \tag{4}$$

To show the importance of efficient computation techniques, consider the direct computation of the DFT equations. Since $x(n)$ may be complex equation (2) can be rewritten as

$$\begin{aligned} X(k) = \sum_{n=0}^{N-1} [(& Re[x(n)]\, Re[W_N^{nk}] - Im[x(n)]\, Im[W_N^{nk}]) \\ & + j\,(Re[x(n)]\, Im[W_N^{nk}] + Im[x(n)]\, Re[W_N^{nk}])] \\ & k = 0, 1, ..., N-1. \end{aligned} \tag{5}$$

Clearly, for each value of k from equation (5), the direct computation of $X(k)$ requires $4N$ real multiplications and $(4N-2)$ real additions. Since $X(k)$ must be computed for N different values of k, the direct computation of the discrete Fourier transform of a sequence $x(n)$ requires $4N^2$ real multiplications and $N(4N-2)$ real additions or, alternatively, N^2 complex multiplications and $N(N-1)$ complex additions. In addition to the multiplications and additions needed for computing equation (5), the implementation of the computation of the DFT on a general-purpose digital computer requires provision for storing and accessing the input sequence values $x(n)$ and values of the coefficients W_N^{nk}. It is generally accepted that a significant measure of the time required to implement a computational algorithm, is the number of multiplications and additions required. Thus, for the direct computation of the discrete Fourier transform, an acceptable measure of the efficiency of the computations is that $4N^2$ real multiplications and $N(4N-2)$ real additions are required. Since the amount of computation time is about proportional to N^2, it is obvious that the number of arithmetic operations needed to compute the DFT by the direct methods becomes huge for large values of N. So, computational procedures such as the FFT that reduce the number of multiplications and additions are of considerable interest [4].

3 Decimation-In-Time FFT Algorithms

The fundamental principle that these algorithms are based on is that of decomposing the computation of the discrete Fourier transform of a sequence of length N into successively smaller discrete Fourier transforms. Algorithms that the decomposition is based on decomposing the sequence $x(n)$ (the index n is usually associated with time) into successively smaller subsequences, are called decimation-in-time algorithms. For convenience, the algorithms are often illustrated by considering the special case of N being an integer power of 2; that is,

$$N = 2^v. \tag{6}$$

Since N is an even integer, $X(k)$ can be computed by separating $x(n)$ into two $N/2$-point sequences consisting of the even-numbered points in $x(n)$ and the odd-numbered points in $x(n)$. By separating $x(n)$ into its even- and odd-numbered points, equation (2) can be rewritten as

$$X(k) = \sum_{n\ even} x(n)\, W_N^{nk} + \sum_{n\ odd} x(n)\, W_N^{nk}. \tag{7}$$

Let $n = 2r$ for n even and $n = 2r+1$ for n odd, here r is an integer. Thus, equation (7) becomes

$$\begin{aligned} X(k) &= \sum_{r=0}^{(N/2)-1} x(2r)\, W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x(2r+1)\, W_N^{(2r+1)k} \\ &= \sum_{r=0}^{(N/2)-1} x(2r)\, (W_N^2)^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x(2r+1)\, (W_N^2)^{rk}. \end{aligned} \tag{8}$$

We know

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi(N/2)} = W_{N/2}. \tag{9}$$

Therefore, equation (8) can be reduced to

$$X(k) = \sum_{r=0}^{(N/2)-1} x(2r)\, W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x(2r+1)\, W_{N/2}^{rk}$$
$$= G(k) + W_N^k\, H(k). \tag{10}$$

$G(k)$ and $H(k)$ in equation (10) are recognized as an $N/2$-point DFT, the first sum being the $N/2$-point DFT of the even-numbered points of the original sequence and the second being the $N/2$-point DFT of the odd-numbered points of the original sequence. Although the index k ranges over N values, $k = 0, 1, ..., N-1$, each of the sums need only be computed for k between 0 and $N/2 - 1$, since $G(k)$ and $H(k)$ are each periodic in k with period $N/2$. After the two DFTs corresponding to the two sums in equation (10) are computed, they are then combined to yield the N-point DFT, $X(k)$.

In general, with N a power of 2 greater than 3, it would be proceed by decomposing the $N/2$-point transforms in equation (10) into $N/4$-point transforms, decomposing the $N/4$-point transforms into $N/8$-point transforms, and continue until left with only two-point transforms. This requires v stages of computation, where $v = log_2 N$. The number of complex multiplications and additions required is $N + 2(N/2)^2$ in the decomposition of an N-point transform into two $N/2$-point transforms. When the $N/2$-point transforms are decomposed into $N/4$-point transforms, then the factor of $(N/2)^2$ is replaced by $N/2 + 2(N/4)^2$, so the overall computation then requires $N + N + 4(N/4)^2$ complex multiplications and additions. If $N = 2^v$, this can be done at most $v = log_2 N$ times, so that after carrying out this decompositions as many times as possible the number of complex multiplications and additions is equal to $N log_2 N$. This is substantial computational savings over the direct computation of the DFT equations that required $2N^2$ complex operations [4].

4 Computer Simulations

A general purpose software program was developed to compute the FFT for variety of signals such as the saccadic eye movements, ECG, sinusoidal, cubic, parabolic, and triangular signals. Three steps are involved in running this program. First, the user has to specify the input data. That is, the input data can be either biological data or the output of an equation. Two different categories of input data are processed through the FFT routine. The first category of input data are binary signals that are collected in the laboratory and recorded in the computer. The saccadic eye movements and ECG signals belong to this group. The second category of input data are numerical points that are generated from mathematical equations. The sinusoidal, cubic, parabolic, and triangular signals all fall into this group.

Second, the user will be asked to choose the appropriate preprocessing method. Three preprocessing methods are on the list: the Hamming window, the flip and reverse in time method, and nothing. The user can choose either the Hamming window method or the flip and reverse in time method if the input data are binary signals such as the saccadic eye movements and the ECG signals. Conversely, the user can choose doing nothing if the input data are numerical data such as the sinusoidal signal. The reason for doing this preprocessing adjustment is that the FFT routine requires the input signals start and end at the same level.

In saccadic eye movements, the signals do not start and end at the same level. Therefore, the signals can be adjusted to start and end at the same level by using the appropriate preprocessing methods such as the Hamming window or the flip and reverse in time. With the Hamming window method, the original signal is multiplied it by the Hamming window function. This produces a signal that starts and ends at the same level as show in Figure 1. With the flip and reverse in time method, simply flip and reverse the original signal. For convenience, suppose there is a time series that contains 256 points. Flip and reverse this time series to get a new time series that contains 512 points. The first 256 points of the new time series are exactly the same as the original time series and the last 256 points of the new time series are the image of the first 256 points. That is, the 257th point is the same as the 256th point, the 258th point is the same as the 255th point, the 512th point is the same as the first point, and so on. This is shown in Figure 2. In sinusoidal, cubic, parabolic, and triangular signals, the signals start and end at the same level. So it is not necessary to do any adjustment for these signals. Therefore, the user should choose doing nothing in this case.

Finally, the signal that starts and ends at the same level is processed through the FFT routine and the results are displayed on the HS100 graphic terminal. These results are discussed in next section.

5 Results And Discussions

The results are displayed on the HS100 graphic terminal, which emulates the Tetronix 4014, and hardcopy can be produced with the laser printer. The horizontal axis is frequency and the vertical axis is magnitude in logarithm scale. The FFT of a saccadic eye movement preprocessed with the Hamming window method is shown in Figure 3. The magnitude is above 1000 for low frequencies. The magnitude drops dramatically as the frequency goes higher. Figure 4 shows the FFT of the same saccadic eye movement with the flip and reverse in time preprocessing method. Here, the magnitude almost reached 1000 for low frequencies and it goes down faster in comparison to the Hamming window figure.

The FFT of the Hamming window all by its self would be a spike at 2 Hz (for a data length of 512 points with millisecond sampling). The flip and reverse in time technique also add an artificial spike at 2 Hz. Comparing these two figures suggests that the Hamming window is smoothing the data significantly; Figure 4 has larger oscillations than Figure 3. Also note that the Hamming window FFT has more high frequency energy. The flip and reverse method has two saccades in the window so it should represents more for the biological data. Therefore, it appears that the flip and reverse technique is better than the Hamming window for saccadic eye movement data.

The Hamming window technique multiplies the original signal with a cosine wave. Multiplication in the time domain is equivalent to convolution in the frequency domain, which would be the same as filtering or averaging in the time domain. Therefore, FFT produced with the Hamming window is smoother than the one produced with the flip and reverse method.

In summary, two preprocessing methods: the Hamming window and the flip and reverse in time methods, are used to adjust the incoming signals before the FFT routine if these signals do not start and end at the same level. The results show that the flip and reverse technique is better than the Hamming window for saccadic eye movements. Furthermore, the results also suggest that FFT produced with the Hamming window is smoother than the one produced with the flip and reverse method.

References

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. of Comput.,* Vol. 19, pp. 297-301, April 1965.

[2] J. W. Cooley, P. A. W. Lewis, and P. D. Welch, "Historical notes on the fast Fourier transform," *IEEE trans. on Audio and Electroacoustics,* Vol. AU-15, No. 2, June 1967.

[3] W. T. Cochran et al., "What is the fast Fourier transform," *IEEE trans. on Audio and Electroacoustics,* Vol. AU-15, No. 2, June 1967.

[4] Alan V. Oppenheim and Ronald W. Schafer, *Digital Signal Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1975.

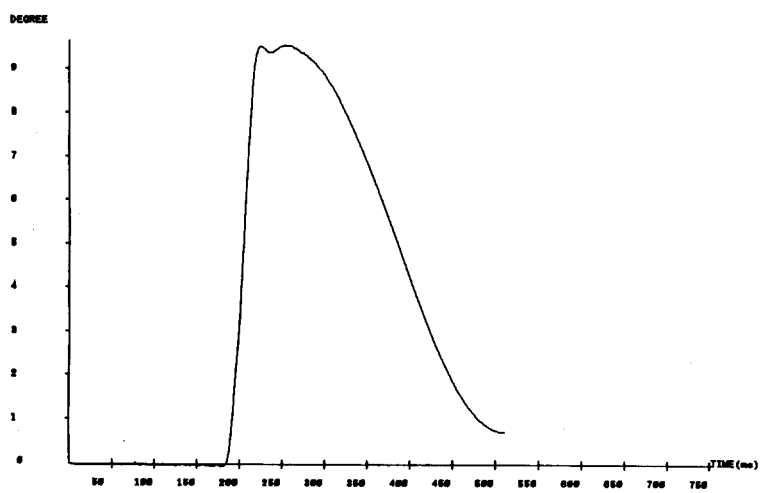

Figure 1 : A signal that starts and ends at the same level (a saccadic eye movement multiply it by the Hamming window function).

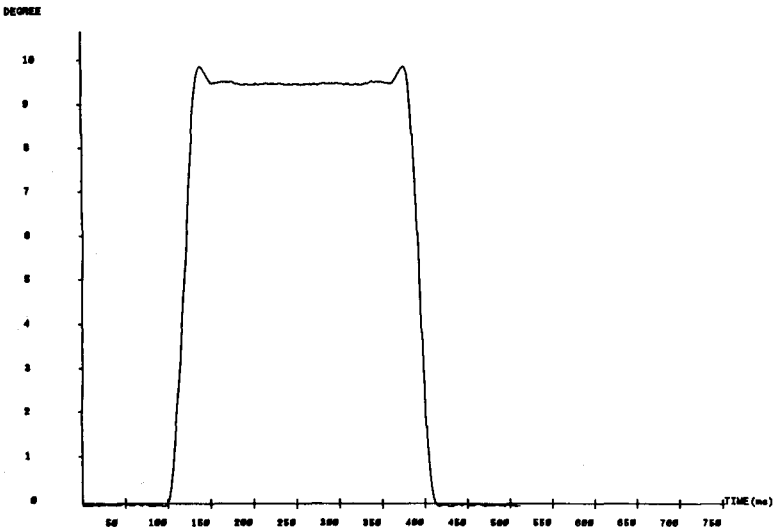

Figure 2: A signal that starts and ends at the same level (flip and reverse a saccadic eye movement in time and connect with the original signal).

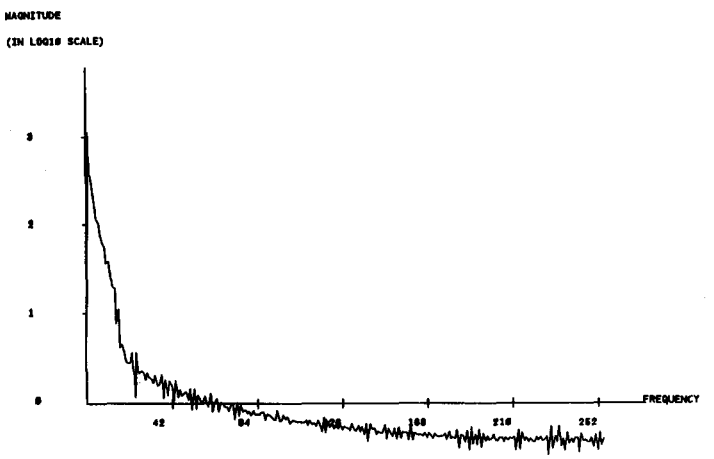

Figure 3 : The FFT of a saccadic eye movement preprocessed with the Hamming window function.

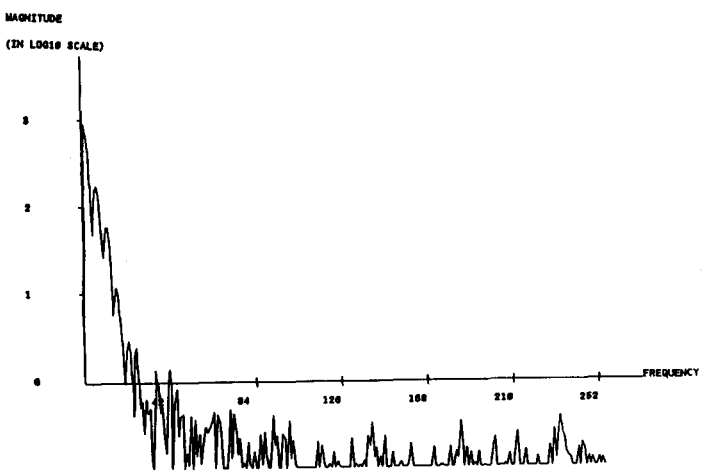

Figure 4 : The FFT of a saccadic eye movement preprocessed with the flip and reverse in time method.