# A Fast Fourier Transform Algorithm Using Base 8 Iterations

By G. D. Bergland

**1. Introduction.** Cooley and Tukey stated in their original paper [1] that the Fast Fourier Transform algorithm is formally most efficient when the number of samples in a record can be expressed as a power of 3 (i.e., $N = 3^m$), and further that there is little efficiency lost by using $N = 2^m$ or $N = 4^m$.

Later, however, it was recognized that the symmetries of the sine and cosine weighting functions made the base 4 algorithms more efficient than either the base 2 or the base 3 algorithms [2], [3]. Making use of this observation, Gentleman and Sande have constructed an algorithm which performs as many iterations of the transform as possible in a base 4 mode, and then, if required, performs the last iteration in a base 2 mode.

Although this "4 + 2" algorithm is more efficient than base 2 algorithms, it is now apparent that the techniques used by Gentleman and Sande can be profitably carried one step further to an even more efficient, base 8 algorithm. The base 8 algorithms described in this paper allow one to perform as many base 8 iterations as possible and then finish the computation by performing a base 4 or a base 2 iteration if one is required. This combination preserves the versatility of the base 2 algorithm while attaining the computational advantage of the base 8 algorithm.

**2. The Basic Algorithms.** Fast Fourier Transform recursive equations for $N = r_1 r_2 \cdots r_n$ can be derived in either of two different forms [4]. The second of these forms is

$$\begin{aligned}(1)\qquad &\hat{A}_p(j_0, j_1, \cdots, j_{p-1}, k_{n-p-1}, \cdots, k_0)\\ &= \sum_{k_{n-p}=0}^{r_p-1} \hat{A}_{p-1}(j_0, j_1, \cdots, j_{p-2}, k_{n-p}, \cdots, k_0)\\ &\qquad \cdot W_N^{j_{p-1}(k_{n-p}(r_{p+1}\cdots r_n)+\cdots+k_0)(r_1\cdots r_{p-1})},\\ &\qquad\qquad p = 1, 2, \cdots, n, \qquad W_N = \exp(2\pi i/N),\end{aligned}$$

where

$$\begin{aligned}(2)\qquad j &= j_{n-1}(r_1 r_2 \cdots r_{n-1}) + j_{n-2}(r_1 r_2 \cdots r_{n-2}) + \cdots + j_1 r_1 + j_0\\ k &= k_{n-1}(r_2 r_3 \cdots r_n) + k_{n-2}(r_3 r_4 \cdots r_n) + \cdots + k_1 r_n + k_0\end{aligned}$$

subject to the constraints

$$\begin{aligned}(3)\qquad j_{v-1} &= 0, 1, 2, \cdots, r_v - 1, \qquad 1 \leqq v \leqq n\\ k_v &= 0, 1, 2, \cdots, r_{n-v} - 1, \qquad 0 \leqq v \leqq n-1.\end{aligned}$$

The last array calculated gives the Fourier sums as

Received June 16, 1967.

$$X(j_{n-1}, \cdots, j_0) = \hat{A}_n(j_0, \cdots, j_{n-1}) . \tag{4}$$

In some cases the total computation required to evaluate these equations can be reduced by grouping these equations in a slightly different manner. For $N = r_1 r_2 \cdots r_n$, this regrouping takes the form

$$\begin{aligned} &\hat{A}_p(j_0, \cdots, j_{p-1}, k_{n-p-1}, \cdots, k_0) \\ &\quad = \left[ \sum_{k_{n-p}=0}^{r_p-1} \hat{A}_{p-1}(j_0, \cdots, j_{p-2}, k_{n-p}, \cdots, k_0) W_{r_p}^{j_{p-1}k_{n-p}} \right] \\ &\quad \cdot W_N^{j_{p-1}(k_{n-p-1}(r_{p+2}\cdots r_n)+\cdots+k_1 r_n+k_0)(r_1 r_2 \cdots r_{p-1})} \cdot \end{aligned} \tag{5}$$

Note that the bracketed term in (5) represents a set of $r_p$-point Fourier transforms and that the complex exponential weights outside the brackets simply rereference each set of results to a common time origin. (In Gentleman and Sande's paper this rereferencing is termed twiddling.) The term

$$W_{r_p} = W_N^{N/r_p} = e^{2\pi i/r_p}$$

forms the basis for the complex exponential weights required in evaluating each $r_p$ point transform, and $j_{p-1}$ and $k_{n-p}$ are the two indices of the transform.

An analogous regrouping can be performed on the original Cooley-Tukey recursive equations. For $N = r_1 r_2 \cdots r_n$, these take the form [4]

$$\begin{aligned} &A_p(j_0, j_1, \cdots, j_{p-1}, k_{n-p-1}, \cdots, k_0) \\ &\quad = \sum_{k_{n-p}=0}^{r_p-1} A_{p-1}(j_0, j_1, \cdots, j_{p-2}, k_{n-p}, \cdots, k_0) \\ &\quad \cdot W_N^{(j_{p-1}(r_1 r_2 \cdots r_{p-1})+\cdots+j_0)k_{n-p}(r_{p+1}\cdots r_n)}, \qquad p = 1, 2, \cdots, n . \end{aligned} \tag{6}$$

When these equations are regrouped, the rereferencing of iteration $p + 1$ must be performed before the $r_{p+1}$ point transform instead of after it. This can be done by performing the $p + 1$ iteration rereferencing at the end of the $p$th iteration. If these rereferenced $A_p$ terms are labeled $\overline{A}_p$, we have

$$\begin{aligned} &\overline{A}_p(j_0, \cdots, j_{p-1}, k_{n-p-1}, \cdots, k_0) \\ &\quad = \left[ \sum_{k_{n-p}=0}^{r_p-1} \overline{A}_{p-1}(j_0, \cdots, j_{p-2}, k_{n-p}, \cdots, k_0) W_{r_p}^{j_{p-1}k_{n-p}} \right] \\ &\quad \cdot W_N^{(j_{p-1}(r_1 r_2 \cdots r_{p-1})+\cdots+j_1 r_1+j_0)k_{n-p-1}(r_{p+2}\cdots r_n)} . \end{aligned} \tag{7}$$

This expression is valid for $p = 1, 2, \cdots, n$ provided that we define $(r_{p+2} \cdots r_n) = 1$ for $p > n - 2$ ,and define $k_{-1} = 0$. Note that the bracketed term of (7) is identical to the bracketed term of (5).

Each iteration of both (5) and (7) is thus conveniently divided into two steps. The first step involves performing a set of $r_p$-point Fourier transforms, and the second step involves use of the Fourier transform shifting theorem to rereference the resulting spectral estimates to the correct time origins. These two operations are performed during each iteration.

Of particular interest in this paper are the algorithms which result when as many of the $r_p$ terms as possible are set to 8. When this is done, the bracketed terms represent a large number of 8-point Fourier transforms. The complex exponential weights

TABLE I
Comparison of arithmetic operations required for base 2, base 4, base 8, and base 16 algorithms.

| *Algorithm* | *Required computation for* | *Real Multiplications* | *Real Additions* |
|---|---|---|---|
| Base 2 algorithm for $N = 2^m$, $m = 0, 1, 2, \cdots$ | Evaluating $(N/2)m$, 2 term Fourier transforms. | 0 | $2Nm$ |
| | Referencing | $((m/2 - 1)N + 1)(4)$ | $((m/2 - 1)N + 1)(2)$ |
| | Complete analysis | $(2m - 4)N + 4$ | $(3m - 2)N + 2$ |
| Base 4 algorithm for $N = (2^2)^{m/2}$, $m/2 = 0, 1, 2, \cdots$ | Evaluating $(N/4)(m/2)$, 4 term Fourier transforms. | 0 | $2Nm$ |
| | Referencing | $((3m/8 - 1)N + 1)(4)$ | $((3m/8 - 1)N + 1)(2)$ |
| | Complete analysis | $(1.5m - 4)N + 4$ | $(2.75m - 2)N + 2$ |
| Base 8 algorithm for $N = (2^3)^{m/3}$, $m/3 = 0, 1, 2, \cdots$ | Evaluating $(N/8)(m/3)$, 8 term Fourier transforms. | $Nm/6$ | $13Nm/6$ |
| | Referencing | $((7m/24 - 1)N + 1)(4)$ | $((7m/24 - 1)N + 1)(2)$ |
| | Complete analysis | $(1.333m - 4)N + 4$ | $(2.75m - 2)N + 2$ |
| Base 16 algorithm for $N = (2^4)^{m/4}$, $m/4 = 0, 1, 2, \cdots$ | Evaluating $(N/16)(m/4)$, 16 term Fourier transforms. | $3Nm/8$ | $9Nm/4$ |
| | Referencing | $((15m/64 - 1)N + 1)(4)$ | $((15m/64 - 1)N + 1)(2)$ |
| | Complete analysis | $(1.3125m - 4)N + 4$ | $(2.71875m - 2)N + 2$ |

required in performing these transforms are: $\pm 1$, $\pm i$, $\pm \exp(+i\pi/4)$, and $\pm \exp(-i\pi/4)$. Since use of the first 2 weights requires no multiplications and the product of a complex number and either of the last 2 weights requires only 2 real multiplications, a total of only 4 real multiplications is required in evaluating each 8-point Fourier transform.

Thus the weights used in evaluating an 8-point Fourier transform all have symmetries which can be profitably exploited. Considerable use of these symmetries is being made since the base 8 algorithm forces us to compute $N/8$, 8-point transforms during each iteration.

**3. Computation Required by Base 2, Base 4, Base 8, and Base 16 Algorithms.** When $N$ can be expressed as a power of 2, the recursive equations (5) and (7) can be specialized such that $r_1 = r_2 = \cdots = r_{n-1} = 2^q$ and $r_n = 2^r$, where $N = 2^m$ and $r = m - q(n-1) \leqq q$. That is $N = 2^q 2^q \cdots 2^q 2^r$. If $2^q$ is referred to as the base of the algorithm, we can compare the computation required by base 2, base 4, base 8, and base 16 algorithms for $N$ being any power of 2. The number of real multiplications and real additions required for various values of $m$ is expressed in Table I. Although these expressions are only exact for values of $N$ which are integral powers of 2, 4, 8, and 16, respectively, they are good approximations for any integral value of $m$.

The real multiplications and additions required for $m = 12$ are given exactly by these expressions and are expressed in Table II.

TABLE II

Real multiplications and additions required in performing base 2, base 4, base 8 and base 16 Fast Fourier Transform algorithms for $N = 4096$.

| *Algorithm* | *Number of real Multiplications* | *Number of real Additions* |
|---|---|---|
| Base 2 $(N = (2)^{12})$ | 81,924 | 139,266 |
| Base 4 $(N = (2^2)^6)$ | 57,348 | 126,978 |
| Base 8 $(N = (2^3)^4)$ | 49,156 | 126,978 |
| Base 16 $(N = (2^4)^3)$ | 48,132 | 125,442 |

In counting the number of multiplications and additions required by each of these algorithms, it is assumed that each rereferencing operation requires one complex multiplication except when the multiplier is $W^0$.* It is also assumed that the basic transform of each algorithm is performed in the most efficient manner possible. Thus as the number base gets progressively higher, the $2^q$ point transform becomes more and more specialized but the total computation required decreases.

* Rereferencing operations involving a multiplication by $\pm i$ or $\pm \exp(\pm i\pi/4)$ could be performed with less arithmetic operations at the expense of having to locate these special products. Since only the rereferencing products involving $W^0$ are easily located, present implementations of these algorithms only treat rereferencing products involving $W^0$ as special cases.

**4. Conclusions.** It is apparent from Table I that performing as many high base iterations as possible, reduces the total computation. As the base of the algorithm increases, however, the basic $2^q$ point transform becomes more involved, and it becomes increasingly difficult to let $N$ be an arbitrary power of 2. A reasonable compromise appears to be the base 8 algorithm. The computation required is very nearly minimized while the 8-point transforms required are still relatively easy to compute.

A Fortran II subroutine, using the base 8 algorithm, has been written for use on the G.E. 635 computer. The execution time is approximately $60Nm$ microseconds (where $N = 2^m$), which is more than 40% faster than the base 2 programs previously used and 20% faster than Gentleman and Sande's "4 + 2" subroutine.

**5. Acknowledgments.** The author wishes to thank W. M. Gentleman, G. Sande, and W. L. Zweig for their suggestions which led to what appears to be the minimum number of computations required in executing the base 16 algorithm.

Bell Telephone Laboratories, Incorporated
Whippany, New Jersey

1. J. W. Cooley & J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, v. 19, 1965, pp. 297–301. MR **31** #2843.
2. W. M. Gentleman & G. Sande, "Fast Fourier transforms for fun and profit," *Fall Joint Computer Conference Proceedings*, Vol. 29, 1966, pp. 563–578.
3. R. E. Miller & S. Winograd, Private Communication.
4. G. D. Bergland, "The fast Fourier transform recursive equations for arbitrary length records," *Math. Comp.*, v. 21, 1967, pp. 236–238.