MATHEMATICS OF COMPUTATION
VOLUME 36, NUMBER 153
JANUARY 1981

A Vector Implementation of the Fast Fourier Transform Algorithm*

By Bengt Fornberg

Abstract. A recent article in this journal by D. G. Korn and J. J. Lambiotte, Jr. discusses implementations of the FFT algorithm on the CDC STAR-100 vector computer. The 'Pease'-algorithm is recommended in cases when only a few transforms can be performed simultaneously. We show how the use of a different algorithm and of trigonometric tables will lead to more than three times faster execution times. The times for large transforms increase only about 39% if the tables are eliminated in order to save storage.

The recent article in this journal: "Computing the Fast Fourier Transform on a vector computer" [1] discussed how to code FFTs efficiently for the CDC STAR-100 computer. It was observed that different codes should be used dependent on the number of transforms that can be performed simultaneously. Two algorithms, referred to as the 'Pease' and the 'Stockham' algorithms, were recommended, dependent upon whether this number was less than or greater than about ten. The purpose of this note is to demonstrate an alternative to the 'Pease'-algorithm which significantly improves the computational efficiency.

To obtain efficient codes on a vector machine like the CDC STAR-100, the relative hardware speeds between different machine instructions must be taken into account. Operating on long vectors, an addition takes half a machine cycle; a multiply and a vector compress (which forms a shorter vector by removing selected elements from a longer vector) each take one cycle per operation. However, a merge (which forms one long vector by alternatively selecting consecutive elements from two shorter vectors) requires three cycles for each operation. A machine cycle takes 40 nanoseconds.

A variety of methods is available for coding the FFT algorithm. References [2]–[6] describe some different ways. The algorithm by Glassman [5] appears unique in that no data rearrangement by costly 'merges' is required at any step (nor any initial or final data permutation). For a scalar code, this advantage is often outweighed by the fact that an extra storage vector is required for temporary data. However, on the CDC STAR-100, the elimination of the expensive 'merge'-operations with no increase in operation count gives superior efficiency.

The Glassman algorithm is described in [5] in terms of a matrix factorization. We have implemented it in a very straightforward way in CDC STAR FORTRAN.

Received August 30, 1979; revised April 21, 1980.
AMS (MOS) subject classifications (1970). Primary 68A10; Secondary 42A68.
*This work was supported by Control Data Corporation and by D.O.E. (Office of Basic Energy Sciences).

0025-5718/81/0000-0014/$01.75

The first two matrix-vector multiplications (involving only matrix elements $1, -1, i$, and, $-i$) are performed separately using only additions and subtractions.

A transform over one set of $N = 2^M$ complex data points (with the real and imaginary parts stored in two consecutive vectors) is easily seen to require

First factor:

4 add/subtract length $N/2$.

Second factor:

8 add/subtract length $N/4$.

Each following factor:

2 compress length N
2 multiply length N
6 add/subtract length $N/2$.

The total cost for each factor (after the first two) is therefore ten vector startups and $5\frac{1}{2}N$ machine cycles.

We assume that all trigonometric constants have been tabulated in advance. This gives the highest possible speed. Actual execution times are shown in Table 1 and compared in Tables 2 and 3 with those for the 'Pease' and 'Stockham' algorithms as implemented by Korn and Lambiotte, Jr.

TABLE 1
Time in milliseconds on the CDC STAR-100

Number of simultaneous transforms	*Transform size*										
	16	32	64	128	256	512	1024	2048	4096	8192	16384
1	.17	.23	.35	.52	.91	1.68	3.34	7.03	14.94	32.04	68.97
2	.18	.28	.43	.77	1.46	2.95	6.29	13.50	29.21	63.33	–
5	.22	.39	.75	1.49	3.11	6.83	14.98	32.90	72.20	–	–
10	.29	.61	1.24	2.71	5.97	13.32	29.64	65.33	–	–	–
20	.44	.96	2.22	5.03	11.46	25.94	58.34	–	–	–	–
50	.82	2.14	5.10	12.14	28.16	64.82	–	–	–	–	–
100	1.56	4.01	10.00	23.92	56.29	–	–	–	–	–	–

TABLE 2
Execution time for the 'Pease' algorithm divided by the time for the present algorithm

Number of simultaneous transforms	*Transform size*					
	64	128	256	512	1024	2048
1	3.1	3.1	3.0	3.0	2.8	2.6
5	3.9	3.6	3.5	3.3	3.3	3.2
10	3.9	3.5	3.2	3.3	3.2	3.2
20	3.8	3.5	3.3	3.3	3.3	–

TABLE 3
Execution time for the 'Stockham' algorithm divided by the time for the present algorithm

Number of simultaneous transforms	*Transform size*					
	64	128	256	512	1024	2048
1	8.9	11.0	12.5	12.8	13.6	12.9
5	4.4	4.2	4.1	3.7	3.4	3.1
10	2.8	2.6	2.3	2.1	1.9	1.8
50	1.1	1.0	.9	.8	–	–
100	.8	.7	.7	–	–	–

The table size for one transform is $2N(\log_2 N - 2)$ words (plus another $1/32$ of this for control vectors). If tables of this size are not acceptable, computational speed can be traded against reductions in the table size. With a penalty only in the number of vector startups, but with none in the operation count, the size can be reduced by a factor of four. An additional factor of two can be gained at the cost of one 'vector reverse' operation of length $N/4$ costing N machine cycles for each matrix factor. If N is large, this represents an 18% increase in execution time. It is also possible to eliminate tables entirely and recalculate the trigonometric data for each transform. Repeated use of the relations

$$\cos\frac{x}{2} = \sqrt{\frac{1+\cos x}{2}} \quad \text{and} \quad \sin\frac{x}{2} = \sqrt{\frac{1-\cos x}{2}}$$

adds another $\frac{9}{8}N$ machine cycles for a total increase in cost of about 39%.

Department of Applied Mathematics
California Institute of Technology
Pasadena, California 91125

1. D. G. KORN & J. J. LAMBIOTTE, JR., "Computing the Fast Fourier Transform on a vector computer," *Math. Comp.*, v. 33, 1979, pp. 977–992.

2. J. W. COOLEY & J. W. TUKEY, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, v. 19, 1965, pp. 297–301.

3. W. T. COCHRAN ET AL., "What is the Fast Fourier Transform?," *IEEE Trans. Audio Electroacoust.*, v. Au-15, 1967, pp. 45–55.

4. M. C. PEASE, "An adaption of the Fast Fourier Transform for parallel processing," *J. Assoc. Comput. Mach.*, v. 15, 1968, pp. 253–264.

5. J. A. GLASSMAN, "A generalization of the Fast Fourier Transform," *IEEE Trans. Comput.*, v. C-19, 1970, pp. 106–116.

6. S. WINOGRAD, "On computing the discrete Fourier transform," *Proc. Nat. Acad. Sci. U.S.A.*, v. 74, 1976, pp. 1005–1006.