FFT of Externally Stored Data

P. S. NAIDU

Abstract—A one-dimensional array is split into a pseudo two-dimensional array and stored into an external device, like a disk. The FPT of the stored array is computed using Eklundh's fast transposition algorithm.

This short note describes an efficient method of computing the discrete Fourier transform (DFT) of a large data array stored in an external device (disk). The present method requires a single device unlike Singleton's method which requires four to eight external devices [3], [2]. The method uses the fast matrix transposition procedure of Eklundh [1]. The method has been programmed for the IBM 360/44 with 32K word memory and 2314 disk drive. The computer program may be obtained from the author upon request.

Manuscript received September 1, 1977; revised June 1, 1978. This work was supported by the University Grants Commission, New Delhi, India.

The author is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India.

0096-3518/78/1000-0472$00.75

I. THEORY

Let N stand for length of data. We assume $N = 2^{\gamma}$. Let us express $N = N_1 \times N_2$ where $N_1 = 2^{\gamma_1}$ and $N_2 = 2^{\gamma_2}$. We segment the array, X_i, $i = 0, 1, 2, \cdots, N - 1$, into N_1 segments each of length N_2. A matrix is formed with these segments as rows. The X_i element of the array is mapped into $X_{m,n}$ element of the matrix, where

$$i = m + n N_1, \quad \begin{matrix} m = 0, 1, \cdots, N_1 - 1 \\ n = 0, 1, \cdots, N_2 - 1. \end{matrix} \tag{1}$$

As an illustration, we show in Fig. 1 an array of 16 elements and corresponding matrix obtained by applying the mapping rule (1).

Now we consider the DFT of the array

$$\bar{X}_k = \sum_{i=0}^{N-1} x_i \exp\left(j \frac{2\pi}{N} ik\right).$$

We shall use the mapping rule to map data array as well as its Fourier coefficients into a matrix. After some simplification we obtain

$$[\bar{X}_{p,q}] = \left[\sum_{m=0}^{N_1-1} \exp\left(\frac{2\pi j}{N} mp\right)\left\{\sum_{n=0}^{N_2-1} X_{m,n} \exp\left(\frac{2\pi j}{N_2} np\right)\right\} \cdot \exp\left(\frac{2\pi j}{N_1} mq\right)\right]^T \tag{2}$$

where $p = 0, 1, \cdots, N_2 - 1$ and $q = 0, 1, \cdots, N_1 - 1$, and $[\ ]^T$ stands for matrix transposition.

II. PROCEDURE

The first sum inside the curly brackets is a DFT of the mth column. It is then multiplied by a phase factor. The second sum (with respect to m) is also a DFT, of the pth row. A column (or row) is read from the external device and is Fourier transformed using a fast Fourier transform (FFT) routine. The resulting Fourier transform is stored in the same location.

In the above procedure the matrix needs to be transposed three times; the first transposition to compute the DFT of columns, the second transposition to compute the DFT of rows, and finally, the third transposition to obtain proper output format of the Fourier coefficients. The last transposition may be avoided if we agree to read Fourier coefficients columnwise. About 25 percent computation time can be saved by the above step.

III. RESULTS

The procedure described above, including the procedure for fast matrix transposition, has been programmed by the author for the **IBM** 360/44 computer. The minimum core storage required is equivalent to storage space for two rows in addition to space required to store the program.

However, time for transposition depends upon the number of rows which can be accommodated within the available core storage. **If** 2^m rows can be stored in the core memory, the number of passes required to transpose a matrix of size 2^{γ_1} is equal to $\gamma_{1/m}$ (rounded up to next integer, if not an integer already). In the procedure described above we need to transpose the matrix at least two times. Hence, we need a total number of $2^{\gamma_1/m}$ passes. Fraser [2], who generalized Singleton's method, claimed to do FFT of an externally stored array in the same number of passes. However, the present method is quite different from that of Fraser's and appears to be simpler.

Finally, the program described above is well-suited for implementation on any small to medium-sized computers. In Table I we present actual computer time for different array sizes.

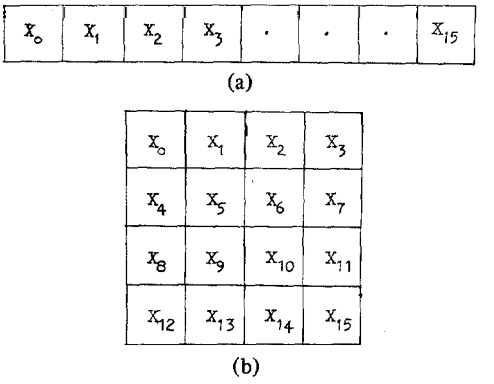

Fig. 1.

TABLE I

N	Time for DFT Data on Disk	Time for DFT Data in Core
512	24 s	1 s
2048	66 s	6 s
4096	**83 s**	16 s
16 **384**	220 **s**	–

REFERENCES

[1] J. **O.** Eklundh, "A fast computer method for matrix transposition," *IEEE Trans. Comput.*, vol. C-21, pp. **801–803,** July 1972.

[2] D. Fraser, "Array permutation by index digit permutation," *J. **Ass. Comput.** Mach.*, vol. ***23,*** no. 2, pp. 298–309, 1976.

[3] R. **C.** Singleton, "A method for computing the fast Fourier transform with auxiliary memory and limited high-speed storage," *IEEE Trans.* ***Audio*** *Electroacoust.*, vol. AU-15, pp. 91–98, June 1967.