Contributions From RR-28

THE INDIAN INSTITUTE OF TROPICAL METEOROLOGY

ON ACCELERATING THE FFT OF COOLEY AND TUKEY

BY

S. K. MISHRA

Ramdurg House, University Road,
Pune 411 005 India

February 1981

RR-28

ON ACCELERATING THE FFT OF COOLEY AND TUKEY

S. K. Mishra
Indian Institute of Tropical Meteorology, Pune-411005.

ABSTRACT

The efficient Fourier transform (EFT) and FFT algorithms are described and their computational efficiencies with respect to the direct method are discussed. An efficient procedure is proposed for the reordering of data set; the use of EFT algorithm for the initial Fourier transforms and restricting the size of final subsets to not less than 4 are also suggested for saving computation time in the FFT. It is found that on average the FFT with the proposed modifications is more than twice as fast as the original FFT. The amount of overhead operations involved in computer routine, based on the modified FFT is estimated.

1. Introduction

In recent years various spectral models of the atmosphere have been developed by Bourke (1972, 74) and others, and used extensively with success in many dynamical studies and numerical weather prediction. The current revival of the interest in spectral models, is apparent from the sudden increase in the volume of the reported studies using spectral techniques, particularly in the last decade. It is linked to the fact that presently spectral models are as efficient as grid point models with respect to computation time which has been made possible by the advent of the transform method developed independently by Orszag (1970) and

Eliasen et al (1970) for spectral multiplications. In the transform method the computation of the non-linear terms are performed in the following two stages. In the first stage the variables are transformed from the spectral space to the physical space and the grid point values of the non-linear terms are obtained. For this purpose the grid point values of the variables involved in the non-linear terms at Gaussian latitudes and equally spaced longitudes are obtained from their spherical coefficients by the use of Legendre inverse transform along latitudinal direction and the Fourier inverse transform along zonal direction. Finally, the grid point values of the non-linear terms are obtained by multiplications of the required values. In the second and final stage the spectral representation of the non-linear terms are obtained by computing the spherical coefficients from their grid point values with the help of the Legendre and Fourier transform along latitude and longitude respectively. The overall performance of the transform method is determined to a large extent by the saving achieved in computation time in performing the Fourier transforms.

The purpose of the FFT (Fast Fourier Transform) is to develop an efficient algorithm for computing the sums occuring in the expressions for Fourier transform. The FFT algorithm, generally involves recursive relationships, when used reduces considerably the number of arithmatic operations involved in computing the Fourier transform. The first FFT

algorithm was introduced by Cooley and Tukey (1965) and subsequently its other versions were developed. The original FFT algorithm of Cooley and Tukey requires that the length N of data set should be even and preferably its major part be expressible as the products of factor 2 in order to exploit the full potential of the algorithm. Some other versions of the FFT as discussed by Temperton (1977) seem to be more efficient than the Cooley and Tukey algorithm and also applicable to more general cases where the Cooley and Tukey algorithm fails. Even in cases when N can not be factorized as a power of 2, but can be factorized as a power of some odd number, these methods are applicable. However, all these FFT versions require more computation time for the N which is factorisable as a power of some odd number compared to the case of a nearest higher N, which can be factorized as a power of 2. Therefore, it is still advantageous to improve the efficiency of the FFT algorithm of Cooley and Tukey in order to make it comparable to the other fast versions of the FFT.

The saving in computation time realised in actual practice by use of the FFT algorithm is much less than the theoretical expectation based on its computational efficiency. The reason for this discrepancy between actual and theoretical efficiency is due to the fact that considerable amount of extra computation time is required for generating the various indices for transfering information from one memory location

to another and the reordering of data set required in the implementation of the FFT algorithm. The operations associated with these factors may be termed as over-head operations and they are not taken into consideration in the theoretical efficiency. Another factor responsible for slowing down the performance of the FFT algorithm is the use of inefficient method for computing the initial Fourier transforms. In this study the main objective is to suggest the procedures for reducing the computation times in the overhead operations and in the evaluation of the initial Fourier transforms in order to increase the computational efficiency of the Cooley and Tukey algorithm.

2. The DFT and FFT algorithms

Let us consider a one dimensional periodic field $f(x)$ of period L, observed over a set of equally spaced points $\{x_j\}$ at the interval of Δx, such that $x_j = j\Delta x$. The discrete field is represented by the set $\{f_j\}$, where $f_j = f(x_j)$ denotes the observed value of the field at the point x_j. The periodicity of the discrete field implies that $f_{N+j} = f_j$ where N is the number of intervals in the period L and assumed to be even. The discrete field $\{f_j\}$ is represented as a discrete Fourier series

$$f_j = \frac{A_o}{2} + \sum_{m=1}^{M-1} \left(A_m \cos \frac{2\pi mj}{N} + B_m \sin \frac{2\pi mj}{N} \right) + \frac{A_M}{2} \cos \frac{2\pi Mj}{N} \qquad .. (1)$$

where $M = N/2$ and A_m and B_m are the Fourier coefficients given by

$$A_m = \frac{2}{N} \sum_{j=1}^{N} f_j \text{ Cos } \frac{2\pi mj}{N} \quad \text{for } m = 0, \ldots, N/2, \qquad \ldots (2)$$

$$B_m = \frac{2}{N} \sum_{j=1}^{N} f_j \text{ Sin } \frac{2\pi mj}{N} \quad \text{for } m = 1, \ldots, N/2-1$$

(A_m, B_m) is the Fourier transform of $\{f_j\}$, and $\{f_j\}$ is the Fourier inverse transform of (A_m, B_m). Eqs (1) and (2) together constitute a discrete Fourier transform pair.

The numerical algorithm based on (2) for computing the Fourier transform will be called Direct Fourier Transform (DFT). Let us count one multiplication and one addition as one machine operation. The total number of the Fourier Coefficients are N and for each coefficient, N machine operations are required as evident from Eq (2). Therefore, N^2 number of machine operations are involved in computing all the Fourier coefficients. The most time consuming part of this algorithm is the direct computation of the N^2 trigonometric functions. However, this computation time can be considerably reduced by using the following trigonometric relations :

$$\text{Cos } (n+1)\theta = \text{Cos } n\theta \cdot \text{Cos}\theta - \text{Sin } n\theta \cdot \text{Sin}\theta$$

and

$$\text{Sin } (n+1)\theta = \text{Sin } n\theta \cdot \text{Cos}\theta + \text{Cos } n\theta \cdot \text{Sin}\theta \qquad \ldots (3)$$

$2N^2$ number of machine operations are required for computing N^2 number of trigonometric functions from Eq (3), under the approximation that two multiplications and one addition are equal to two machine operations, which is very nearly true. The final estimate of the total number of machine operations required in computing the Fourier transform by using the DFT algorithm is $3N^2$.

An efficient Fourier transform (EFT) method as discussed by Ralston (1965) is based on the following algorithm for computing the sums in Eq (2).

$$A_m = \frac{2}{N} \left(f_N + U_{1,m} \cos \frac{2\pi m}{N} - U_{2,m} \right) \quad .. (4a)$$

and

$$B_m = \frac{2}{N} U_{1,m} \sin \frac{2\pi m}{N} , \quad .. (4b)$$

where $U_{1,m}$ and $U_{2,m}$ are computed from the following recurrence relation.

$$U_{k,m} = f_k + 2 \cos \frac{2\pi m}{N} U_{k+1,m} - U_{k+2,m}, \quad .. (5)$$

for $k = N-1, N-2, \ldots, 1$ and $U_{N,m} = U_{N+1,m} = 0$. The EFT algorithm lies somewhere in between the DFT and FFT algorithm in computational efficiency. The computation of $U_{k,m}$ by Eq (5) requires one multiplication and two additions, assuming that the cosine function is stored. We can approximate this to 4/3 machine operations. Thus the total number of machine

operations required are $2N^2/3$. Further, it is also assumed that all the required trigonometric functions are stored which is a practical assumption in this case. It is evident from above, that theoretically the EFT algorithm is 4.5 times faster than the DFT algorithm.

3. The FFT algorithm of Cooley and Tukey

Let us introduce a superscript in the Fourier coefficients to denote the length of data set. For saving a few multiplications, we dropped the factor 2/N from Eq (2) for further discussion here but the coefficients so obtained will be multiplied by this factor. The required recursive relations for the Fourier coefficients for the Cooley and Tukey algorithm are obtained from Eq (2) as follows :

$$A_m^N = \sum_{j=1}^{N/2} f_{2j-1} \cos \frac{2\pi m (2j-1)}{N} + \sum_{j=1}^{N/2} f_{2j} \cos \frac{2\pi m 2j}{N}$$

or

$$A_m^N = \cos \frac{2\pi m}{N} \sum_{j=1}^{N/2} f_{2j-1} \cos \frac{2\pi mj}{N/2} + \sin \frac{2\pi m}{N} \sum_{j=1}^{N/2} f_{2j-1}$$

$$\sin \frac{2\pi mj}{N/2} + \sum_{j=1}^{N/2} f_{2j} \cos \frac{2\pi mj}{N/2}$$

or

$$A_m^N = \cos \frac{2\pi m}{N} C_m^{N/2} + \sin \frac{2\pi m}{N} D_m^{N/2} + E_m^{N/2} \quad ; \ldots \quad (6a)$$

Similarly it can be shown that

$$B_m^N = -\text{Sin}\ \frac{2\pi m}{N}\ C_m^{N/2} + \text{Cos}\ \frac{2\pi m}{N}\ D_m^{N/2} + F_m^{N/2}, \qquad .. (6b)$$

where

$$C_m^{N/2} = \sum_{j=1}^{N/2} f_{2j-1}\,\text{Cos}\ \frac{2\pi mj}{N/2}\ , \qquad .. (7a)$$

$$D_m^{N/2} = \sum_{j=1}^{N/2} f_{2j-1}\ \text{Sin}\ \frac{2\pi mj}{N/2}\ , \qquad .. (7b)$$

$$E_m^{N/2} = \sum_{j=1}^{N/2} f_{2j}\ \text{Cos}\ \frac{2\pi mj}{N/2}\ , \qquad .. (8a)$$

$$F_m^{N/2} = \sum_{j=1}^{N/2} f_{2j}\ \text{Sin}\ \frac{2\pi mj}{N/2}\ , \qquad .. (8b)$$

and $m = 0, \ldots\ldots, M\ 1$

$(C_m^{N/2}, D_m^{N/2})$ is the Fourier transform of the data set $\left\{ f_{2j-1} \right\}$, $j = 1, \ldots\ldots N/2$, constituted by the odd data points and $(E_m^{N/2}, D_m^{N/2})$ is the Fourier transform of the data set $\left\{ f_{2j} \right\}$, $j = 1, \ldots\ldots N/2$, constituted by the even data points, and the length of each field is $N/2$. In the first stage of the FFT method, the data set is split

into two subsets. The Fourier coefficients of the data set are obtained from Eqs (6a) and (6b). M1 = N/4, when N/2 is even and M1 = (N/4 - 1/2) when N/2 is odd. It is clear that from Eqs (6a) and (6b) Fourier coefficients of the original data set can be determined only up to M1. The recursion equations for determining the remaining Fourier coefficients, for which m lies in the closed interval [M1+1, M], are obtained after taking into account the following symmetric relations :

$$C^{N/2}_{N/2-m} = C^{N/2}_{m} \quad ,$$

$$D^{N/2}_{N/2-m} = -D^{N/2}_{m} \quad , \qquad \text{.. (9)}$$

$$E^{N/2}_{N/2-m} = E^{N/2}_{m} \quad ,$$

and

$$F^{N/2}_{N/2-m} = -F^{N/2}_{m} \quad .$$

Replacing m by N/2-m in (6) and using (9) we get

$$A^{N}_{N/2-m} = -\text{Cos}\ \frac{2\pi m}{N}\ C^{N/2}_{m} - \text{Sin}\ \frac{2\pi m}{N}\ D^{N/2}_{m} + E^{N/2}_{m}, \quad \text{..(10a)}$$

$$B^{N}_{N/2-m} = -\text{Sin}\ \frac{2\pi m}{N}\ C^{N/2}_{mm} + \text{Cos}\ \frac{2\pi m}{N}\ D^{N/2}_{m} - F^{N/2}_{m} \quad \text{.. (10b)}$$

These are the required recursion equations for computing the remaining Fourier coefficients.

In the second stage each subset is further divided

into two subsets if $N/2$ is still even and this process of splitting will be continued till each subset contains only an odd number of data points, We can express the length of data set as

$$N = N_o 2^P$$

where N_o and 2^P are the length and number of the final subsets respectively and p represents the number of times the original data set is divided into subsets which is a positive integer.

To understand how the FFT works, let us consider an example of the data set of length $N = 12$. In this case $N_o = 3$, $p = 2$ and the number of subsets are 4. The complete reordering of the original data set in the present example is shown stage by stage in the Fig. 1. At the top of the figure is shown the data set, which we call zeroth stage of reordering process. The symbols inside the boxes denote the values of data and their positions with reference to the zeroth stage. In the figure, a group of boxes represents the data set or a subset. The reordering of the original data set is essential, for the working of the Cooley and Tukey algorithm. The Fourier coefficients of each of the four subsets can be obtained by the direct or the efficient method. From these initial Fourier coefficients, the final Fourier coefficients of the data set are obtained in two stages as indicated in the figure, with the help of Eqs (6)

and (10).

4. Efficiency of the FFT Algorithm

Let us consider a data set of length N, which is splitted p times into the 2^P subsets, each of length N_o. The number of machine operations required for obtaining the Fourier coefficients of all the 2^P subsets, at p^{th} stage by the direct method is $2^P N_o (N_o - 1)$, the few additions required for the zeroth coefficients are neglected. The number of operations required for obtaining the Fourier coefficients of all the subsets at each stage preceding the p^{th} stage is the same and equal to N. Here, we have not counted the number of multiplications involved in the computations from Eq (10) because they are same with those of Eq (6), already taken into account. Further, the additions of Eq (10) are neglected because nearly an equivalent number of the machine operations are saved in the computations by using the simplified form of the Eqs (6) and (10) for $m = 0$ and M_1. The recursion relations have to be used p times to get the final results. Finally, the N Fourier coefficients so obtained are multiplied by 2/N and this means an additional N machine operations. Therefore the total number of operations (S) required in the FFT may be written as

$$S = N (N_o + p).$$

It is assumed in the above computation that the trigonometric functions are readily available. Thus the comparative effi-

ciency (E) of the FFT over DFT is given by

$$E = \frac{3N_o\, 2^p}{(N_o + p)} \qquad .. (11)$$

It is evident from Eq (11) that E is large for the higher values of p. In the direct method it is not practical to store all the required trigonometric functions, since the number of storage locations required for this purpose are $(N^2 + 1)$, which is quite high for large N compared to the number of locations $(N/2 + N_o^2)$ required in the FFT. The first part in the above formula represents the number of distinct trigonometric functions required in (6) and (10), and the second part for obtaining the initial Fourier coefficients of the final subsets by the direct method, which is quite small, and hence a practical proposition to store them. This feature of the FFT is one of the major factors for the large gain in computational time. The actual gain in the FFT over DFT given by Eq (11) is an over estimate, since the requirement of reordering of the data set and generation of a number of indices required for the effeetive use of the recursion relations without using extra memory storage means extra operations which are not considered in the derivation of Eq (11). It may be added here that the computational efficiency of the FFT with respect to the EFT is 2/9 times of E.

5. The proposed reordering procedure for the FFT

In this and the next section we shall discuss the

main objectives of this paper namely : (a) accelerating the process of reordering of the data set into the subsets and (b) increasing the efficiency in computing the initial Fourier transforms. The reordering of the data set can be achieved by a many steps direct splitting procedure where odd and even positioned values are group together at each step. This procedure for the reordering is inefficient in performance as will be shown subsequently.

From the example illustrated by Fig. 1 and from other examples, the following rule regarding the reordering process of the original data set can easily be deduced by inspection. It may be noted in this connection that the index of data which occupies the first position in the first subset is 1, the remaining $(N_o - 1)$ data in this subset have indices which are higher by 2^P from the preceding one. Further, the indices of the data in the next $2^o (= 1)$ subsets are larger by 2^{P-1} from the corresponding indices of the first $2^o (= 1)$ subsets, the indices in the next $2^1 (= 2)$ subsets are larger by 2^{P-2} from the corresponding index of the first $2^1 (= 2)$ subsets; and so on, till finally the indices of the data in the last 2^{P-1} subsets are larger by $2^o (= 1)$ from the corresponding indices in the first 2^{P-1} subsets. Let q_m denote index of the data point in the original set, which now occupies the position m, in the reordered set.

The rule for correspondence between the positions of data points in the original and reordered data set can be represented mathematically as

$$q_m = q_{m-k} + 2^{P-j} \qquad .. (12)$$

where $j = 0, 1, 2, \ldots\ldots, p$;

$m = 2, \ldots\ldots\ldots\ldots, N_o$ for $j = 0$;

$m = N_o\ 2^{j-1} + 1, \ldots, N_o\ 2^j$ for $j = 1, 2, \ldots, p$;

$k = 1$ for $j = 0$;

$k = N_o\ 2^{j-1}$ for $j = 1, 2, \ldots\ldots, p$;

and $q_1 = 1$.

Using these indices q_m the reordering of data set is accomplished in a single step. This procedure of the reordering in FFT is easy to program and is implemented without any extra memory requirements. We can call the FFT with this reordering procedure as the modified FFT.

Computational times of the FFT for different lengths N are given in Table 1. Column 5 shows the time required for the modified FFT and column 4 is for the FFT with the reordering process based on the splitting of a set into subsets, stage by stage. In both the cases the DFT method was used for the initial Fourier transforms. It can be seen by comparing the time noted in the two columns that the present

reordering procedure resulted in the decrease of the computational time by about 40%, for the cases when $N_o = 1$. However, when $N_o = 3$, this decrease is only around 25%. This difference in the efficiency in the two cases is due to the fact that in the second case when $N_o = 3$, the relative contribution of the reordering time to the total time is less compared to the first case when $N_o = 1$.

6. An efficient FFT (EFFT) algorithm

From Table 2, it can be seen that the EFT algorithm is far superior to the modified FFT for $N \leqslant 8$. However, it is expected because even the theoretical efficiency of the FFT as given by (11) is considerably less than that of the EFT.

Therefore, in order to increase further the efficiency of the modified FFT, it is proposed that the each subset of the reordered set must not have less than four data points, i.e., $N_o \geqslant 4$, and the initial Fourier transforms are to be obtained by using the EFT algorithm. The FFT algorithm with this modification coupled with the modified reordering procedure for the data set, discussed in the last section, will be called as EFFT. It may also be noted that the storage requirement for the trigonometric functions, $(N/2 + N_o)$ in number is considerably less, particularly for large N_o, than the case when DFT is used for obtaining the initial Fourier coefficients.

A comparison between columns 4 and 6 of Table 1, has shown that the saving of computation time in the EFFT is around 55% of the FFT, for the cases when $N_0 = 1$, in other cases when $N_0 = 3$ it is about 40% . This difference in the saving of the computation times for the two different types of cases is as explained in the last section. Thus the EFFT is more than two times faster than the FFT. It is not possible to make precise comparison of the computation times between the EFFT, proposed here and the other fast versions of the FFT. A rough estimate of the efficiency of the EFFT can be obtained with the help of CPU times on CDC-6600, reported by Temperton (1977), for the routines based on his version of the FFT and that of Norman Bernner of MIT, based on Cooley and Tukey version of the FFT, for the power of 4. The computation times were reported for the Fourier transforms of different lengths N. After taking into consideration the difference in the speed of the two computer models, it can be said that the EFFT as proposed here is comparable to the other versions in the computational efficiency.

A self explainatory Fortran program based on the proposed EFFT for obtaining the real Fourier transform of length N is given in Appendix.

7. Estimation of the overhead operations in the EFFT

As indicated earlier, the theoretical efficiency

of the FFT is not achieved in its actual performance. This is due to the significant contribution of overhead operations in the total computation time. The increase in the efficiency of EFFT is contributed mostly by saving in the computation times of overhead calculations due to the suggested procedures for this purpose. It seems natural to know the amount of overhead operations (R) still required in the EFFT as a percentage of the main machine operations, counted for computing the theoretical efficiency. It is understandable that the number of overhead arithmatic operations is dependent upon the parameters p and N_o. The number of main machine operations required in the EFFT is also a function of these two parameters. A more simplified assumption and still realistic may be that the ratio of the overhead and the main operations is a constant. Under this assumption the theoretical (E_1) and actual (E_1^1) efficiencies of EFFT with respect to EFT are related as

$$E_1^1 = \frac{E_1}{(1 + R/100)} , \qquad .. (13)$$

where

$$E_1 = \frac{2}{9} E .$$

E_1^1 is computed as the ratio of computation times of EFT and EFFT presented in columns 3 and 6 respectively of Table 1 and E from Eq (11). R is computed for different data

lengths N of the table, and its average value is determined to be 113%. This means that even in the present form of EFFT, more computation time is used in performing overhead operations than the main operations. On the basis of above result, it can be said that still there exists a scope for achieving significant gain in computational efficiencies of the various FFT algorithms by further reduction in the number of overhead operations alone. Further calculations revealed that around 30% of the total overhead computation time is utilised in the reordering procedure.

8. Conclusions

It has been shown in this paper that the computational efficiency of routine based on the Cooley and Tukey algorithm can be considerably enhanced and made comparable to the other recent fast versions of FFT by reducing the overhead computation time involved and by using the efficient algorithm for computing the initial Fourier transforms. For this purpose an efficient procedure has been proposed for the reordering of data set. It has been demonstrated that by restricting the size of final subsets to be not less than 4, significant saving in computation time can be achieved. Further, it has been estimated that in the accelerated version of the FFT algorithm of Cooley and Tukey, the computation time associated with performing of overhead operations is greater than that of the main operations, even after in-

corporating the proposed modifications for this purpose.

Acknowledgements

The author is grateful to Shri J.D. Murti for computer programming help at the initial stage of this study, Shri S.K. Kar and Dr. D. Subrahmanyam for their helpful comments, and Shri D.R. Chakraborty and Mrs. S.S. Desai for cooperation. He also wishes to thank Mrs. V.V. Savant for typing the manuscript.

REFERENCES

Bourke, W. 1972 An efficient, one-level, primitive equation spectral model. Mon. Wea. Rev., 100, 683-689.

Bourke, W. 1974 A multi-level spectral model I. Formulation and hemispheric integration. Mon. Wea. Rev., 102, 687-701.

Cooley, J.W. and J.W. Tukey 1965 An algorithm for the machine calculation of complex Fourier series. Math. Comput. 19, 297-301.

Eliasen, E., B. Machenhauer and E. Rasmussen 1970 On a numerical method for integration of the hydrodynamical equation with a spectral representation of the horizontal fields. Rept. No. 2, Inst. Teoret. Meteor., Copenhagen, 35 pp.

Orszag, S.A. 1970 Transform method for the Calculation of Vector coupled sums : Application to the spectral form of the vorticity equation. J. Atmos. Sci. 27, 890-895.

Ralston, A. 1965 A First Course in Numerical Analysis. McGrew-Hill Book Co., New York, 578 pp.

Temperton, C. 1977 Mixed-radix fast Fourier transforms without reordering. Tech. Rept. No. 3, Bracknell, European Centre for Medium Range Weather Forecasts, 35 pp.

...

BHEGADE /20.1.1981.

APPENDIX

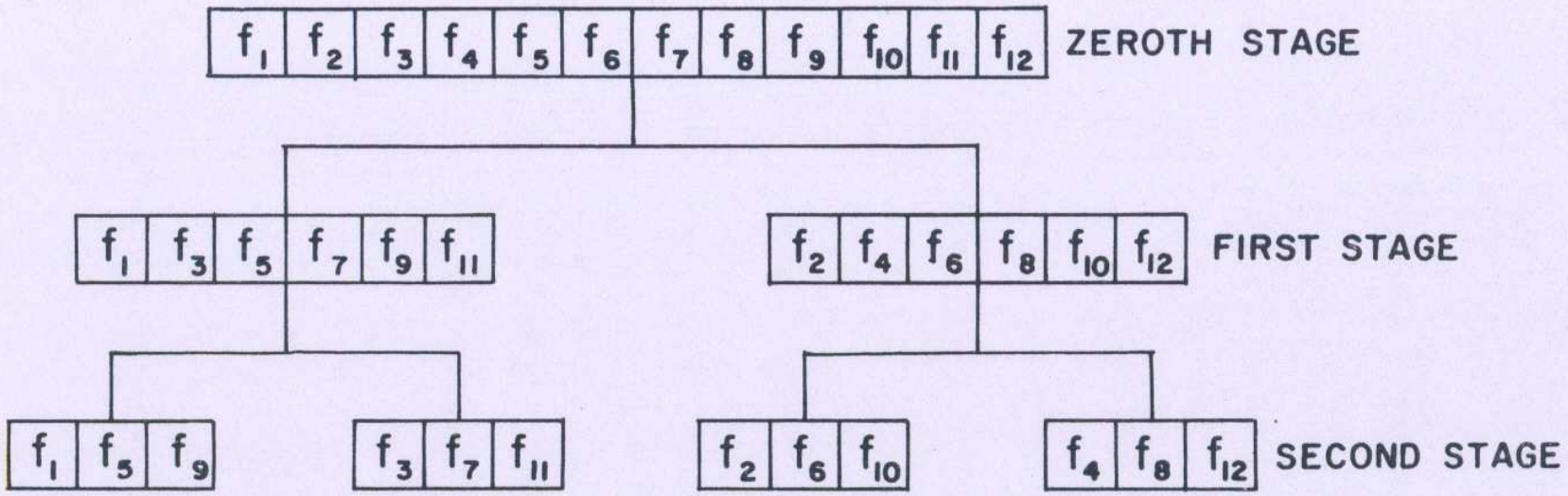
```
C       MAIN PROGRAM FOR FOURIER SERIES ANALYSIS
        COMMON/H1/IPT,NS,NF,NF1,NF2,NF3,IEOD,FSN,PIE,MMAX1,
        NDATA,IPQ(128)
        COMMON/H2/CF(35),SF(35), CFF(5),SFF(5)
        DIMENSION Y(128),FV(128)
      1 FORMAT(I3)
    100 FORMAT(1X,12E10.3)
    101 FORMAT (1X,20HFOURIER ANALYSIS FOR,1X,I3,11HDATA POINTS)
        PIE=3.14159265358
C       NDATA NO. OF DATA POINTS
        READ 1,NDATA
C       READ DATA  IN FV LOCATION
        CALL REARG
        CALL CSFUN
        PRINT 101,NDATA
        CALL EFFT(FV,Y)
C       COS FOURIER COEFFICIENTS IN FV AND SIN COEFFICIENTS IN Y
        PRINT 100,(FV(I),Y(I)), I=1,MMAX1)
        END
        SUBROUTINE REARG
        COMMON/H1/IPT,NS,NF,NF1,NF2,NF3,IEOD,FSN,PIE,MMAX1,
        NDATA,IPQ(128)
        N=NDATA
        AL=N
        FSN=2./AL
        IPT=0
      4 IF(N/2*2.NE.N)GO TO 5
        N=N/2
        IF(N.GE.4) GO TO 3
        N=N+N
        GO TO 5
      3 IPT=IPT+1
        GO TO 4
      5 NS=2**IPT
        NF=N
        IEOD=1
        IF(NF/2*2.NE.NF) IEOD=2
        NF1=NF-1
        NF2=NF-2
        NF3=NF-3
        K1=1
        K2=NF
        IK=NS
        IPQ(1)=1
        MM=IPT+1
        DO 7 I=1, MM
        KK1=K1+1
        DO 8 J=KK1,K2
        JJ=J-K1
      8 IPQ(J)=IPQ(JJ)+IK
        K1=K2
        K2=K2+K2
```

```
      7 IK=IK/2
        RETURN
        END
        SUBROUTINE CSFUN
        COMMON/H1/IPT,NS,NF,NF1,NF2,NF3,IEOD,FSN,PIE,MMAX1,
        NDATA,IPQ(128)
        COMMON/H2/CF(35),SF(35),CFF(5),SFF(5)
        IMAX=1
        IF(NF.NE.4) IMAX=2
        DO 10 I=1,IMAX
        IF(I.EQ.2) GO TO 5
        LL=NDATA/4+1
        T=PIE*FSN
        GO TO 6
      5 LL=NF/2+1
        ANF=NF
        T=(PIE+PIE)/ANF
      6 C=COS(T)
        S=SIN(T)
        C1=1.
        S1=0.
        DO 10 L=1,LL
        IF(I.EQ.2) GO TO 7
        CF(L)=C1
        SF(L)=S1
        GO TO 8
      7 CFF(L)=C1
        SFF(L)=S1
      8 A=C1*C-S1*S
        B=S1*C+C1*S
        C1=A
     10 S1=B
        RETURN
        END
        SUBROUTINE EFFT(X,Y)
C THIS SR COMPUTE FOURIER COEFFICIENTS BY THE EFFICIENT FFT
        ALGORITHM
        COMMON/H1/IPT,NS,NF,NF1,NF2,NF3,IEOD,FSN,PIE,MMAX1,
        NDATA,IPQ(128)
        COMMON/H2/CF(35),SF(35),CFF(5),SFF(5)
        DIMENSION X(128),Y(128),D(9)
C       INPUT DATA SET IN X
C       NDATA NO OF DATA IN GIVEN SET.NDATA=NF NS
C       NF NO OF DATA IN EACH FINAL SUBSET
C       NS=2**IPT NO OF SUBSETS
C       MMAX1 NO OF FOURIER COEFFICIENTS REQUIRED
C       MAXFC=(NDATA/2+1)MAXIMUM POSSIBLE FOURIER COEFFICIENTS
        FOR DATA SET
C       X(1) NOT AT ORIGIN,X(NDATA) AT 2 PIE
        MAXFC=NDATA/2+1
        IF(MMAX1.GT.MAXFC) MMAX1=MAXFC
C       REORDERING OF DATA SET
```

```
      DO 5 I=1,NDATA
      J=IPQ(I)
    5 Y(I)=X(J)
      MM=NF/2+1
      HMM=MM
      IN=+1
      ZK=.5*(1.+HMM)
      ZKI=ZK
      K=-NF
      KK=-MM
C     COMPUTATION OF INITIAL FOURIER COEFFICIENTS OF
      SUBSETS IN D
C     COSINE COEFFICIENTS IN X AND SINE ONE IN Y
      DO 10 I=1,NS
      ZK=-ZK
      K=K+NF
      IZON=ZKI+ZK
      KK=KK+MM
      II=KK+IZON
      IN=-IN
      SUM=0.
      DO 9 J=1,NF
      JJ=K+J
      D(J)=Y(JJ)
    9 SUM=SUM+D(J)
      II0=II+IN
      X(II0)=SUM
      Y(II0)=0.
C     CODING FOR N=4, FOURIER TRANSFORM
      IF(NF.NE.4)GO TO 8
      III=II0+IN
      X(III)=D(4)-D(2)
      Y(III)=D(1)-D(3)
      III=III+IN
      X(III)=D(2)+D(4)-D(1)-D(3)
      Y(III)=0.
      GO TO 10
    8 MI=II0
      DO 7 M=2,MM
      MI=MI+IN
      CE=CFF(M)
      TCE=CE+CE
      SE=SFF(M)
      UP2=D(NF1)
      UP1=D(NF2)+TCE*UP2
      L=NF2
      DO 6 J=1,NF3
      L=L-1
      U=D(L)+TCE*UP1-UP2
      UP2=UP1
    6 UP1=U
      Y(MI)=U*SE
```

```
  7 X(MI)=D(NF)+UP1*CE-UP2
 10 CONTINUE
    K2=NS
    DO 20 I=1,IPT
    K2=K2/2
    JJ=-MM
    IN=MM+MM
    MM2=MM-1
    IF(I.GE.IEOD) GO TO 19
    MINW=1
    IS=0
    IM=0
    GO TO 18
 19 MINW=2
    IS=1
    IM=-K2
    I1=MM
    DO 17 J=1,K2
    I2=I1+1
    X2=X(I2)
    SY1=Y(I1)
    Y2=Y(I2)
    SX1=X(I1)
    X(I1)=X2+SY1
    Y(I1)=Y2-SX1
 17 I1=I1+IN
 18 KMM=K2*MM+IM+1
    DO 16 J=1,K2
    JJ=JJ+IN
    JJ1=JJ+1
    IML=KMM
    DO 15 M=MINW,MM2
    J1=JJ1-M
    J2=JJ+M
    J2S=J2-IS
    IML=IML-K2
    CE=CF (IML)
    SE=SF (IML)
    X1=X(J1)
    X2=X(J2)
    Y1=Y(J1)
    Y2=Y(J2)
    CCO=CE*X1+SE*Y1
    SCO=-SE*X1+CE*Y1
    X(J1)=X2+CCO
    X(J2S)=X2-CCO
    Y(J1)=Y2+SCO
 15 Y(J2S)=-Y2+SCO
    J1=J1-1
    J2=J2+1
    J2S=J2-IS
    X1=X(J1)
```

```
      X2=X(J2)
      X(J1)=X1+X2
      X(J2S)=-X1+X2
      Y(J1)=0
   16 Y(J2S)=0.
      NM=IN-IS
      IF(K2.EQ.1)GO TO 20
      IF(IS.EQ.0)GO TO 14
      J1=0
      JJ=0
      DO 13 J=2,K2
      JJ=JJ+IN
      J1=J1+IS
      DO 13 M=1,NM
      J2=JJ+M
      ML=J2-J1
      X(ML)=X(J2)
   13 Y(ML)=Y(J2)
   14 II=1
      JJ=-NM
      IN=NM+NM
      DO 12 J=2,K2,2
      JJ=JJ+IN
      II=II+IN
      DO 12 M=1,MM
      J1=JJ+M
      J2S=II-M
      X1=X(J1)
      X2=X(J2S)
      X(J1)=X2
      X(J2S)=X1
      Y1=Y(JL)
      Y2=Y(J2S)
      Y(J1)=Y2
   12 Y(J2S)=Y1
      MM=NM
   20 CONTINUE
C     COSINE COEFFICIENTS IN X,AND SINE ONE IN Y
      DO 25 I=1, MMAX1
      X(I)=X(I)*FSN
   25 Y(I)=Y(I)*FSN
      X(1)=.5*X(1)
      RETURN
      END
```



Legend of diagram

Fig.1. Flow diagram showing stage by stage reordering of data set from top to bottom and in the opposite direction indicating the process of obtaining Fourier transform for $N = 12$.