DIGITAL FILTERING USING THE FAST FOURIER TRANSFORM SUBROUTINE

A THESIS

SUBMITTED TO THE FACULTY OF ATLANTA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

BY

RAMONA MARIE CALVEY

DEPARTMENT OF CHEMISTRY

ATLANTA, GEORGIA

JULY 1981

ABSTRACT

CHEMISTRY

CALVEY, RAMONA M. B.S. XAVIER UNIVERSITY, 1978

<u>DIGITAL FILTERING USING THE FAST FOURIER TRANSFORM SUBROUTINE</u>

Advisor: Professor G. Scott Owen

Thesis date July 1981

When dealing with data generated in the chemical laboratory it is important that the frequency spectrum of the data be known, for it is the frequency spectrum that provides insight into the signal's noise content. Capitalizing on the frequency differences between noise and the original signal, digital filtering techniques make possible the partial removal of noise and maximize the possibility of accurate and sensitive measurements.

A FORTRAN program which performs digital filtering employing the fast Fourier transform (FFT) method was developed for a PDP 11/34 minicomputer. This program provides the user with the option of attaining a graphical representation of the data as each step- 1) reading in a previously stored file of data, 2) performing a forward transformation, and 3) zero filling the arrays to remove unwanted frequency components (noise) and performing an inverse transform-is performed. Also, the option of writing the results of either the forward or inverse transform out to a file is given.

In addition, other investigators have reported the application of the FFT method to a number of chemical techniques in the analytical laboratory. These include multiplex gas chromatography, linear least squares parameter estimation of fused peak systems, and the interpolation of chromatographic, spectroscopic, and electrochemical data.

ACKNOWLEDGEMENTS

First of all, I would like to express boundless gratitude to my parents, whose tireless efforts, encouragement, and understanding provided inspiration and support that laid a foundation for all my endeavors.

In expressing appreciation to associates and staff who contributed to the success of this project, I extend my deepest gratitude to Terry J. Green, who through patience and insight, facilitated the technical resolution of a number of problems associated with the program. I would like to thank Dr. G. Scott Owen for the definition of the project and for his support during the oral defense of my thesis. I would also like to express my appreciation to Dr. Thomas W. Cole and the staff of the Resource Center for Science and Engineering for the support that they provided. Finally, I would like to thank Ms. Carol E. Johnson for her encouragement and indulgence in the typing and revising of my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure		Page

INTRODUCTION

During the decade of the sixties, an important new approach to waveform manipulation, or signal processing, came into prominence. The practicality of representing information-bearing waveforms digitally so that signal processing could be done on the digital representation of the waveform was realized. In addition, the sixties and seventies were marked by phenomenal progress in computer technology, thus enabling computers to be more affordable and therefore more accessible to an ever increasing user community generating new demands for even more sophisticated technology. Several efficient algorithms for filtering and spectral analysis were developed, making the computer a useful tool in the analytical laboratory.

When dealing with data generated in the laboratory it is important that the frequency spectrum of the data be known. This is significant for it is the frequency spectrum of the original signal that provides insight into the signal's noise content, where noise may be defined as any component of a signal which impedes observation, detection, or utilization of the information the signal is carrying.[1] Filtering techniques capitalizing on the frequency differences between noise and the original signal make possible the partial removal of noise and maximize the possibility of accurate and sensitive measurement. Although the bulk of the data recorded in the laboratory is in the form of amplitude versus time, these signals may also be characterized in the form of amplitude versus frequency by the application of a fast Fourier transformation (FFT) to the digitized data.

The purpose of this research project was to develop a FORTRAN program which would perform digital filtering employing the FFT method. This program was developed for a PDP 11/34 minicomputer. The program reads in a previously stored file of data, graphs the data, and gives the option of writing the results of the Fourier transformation out to a file.

Theory

In order that the theory of the FFT may be understood it is appropriate to briefly survey the general concepts of both the classical continuous Fourier transform (CFT) and the discrete Fourier transform(DFT).

The CFT takes a function defined for all values of time and seperates it into a set of sinusoids that are represented by a complex-valued frequency function. Each value of the frequency function that is non-zero indicates that a sinusoid at that frequency is a component of the original time function and is equal to the amplitude of the associated sinusoid. The real components of the complex-valued frequency function specify sinusoids that are even functions, functions having the property

$$f(t) = f(-t)$$

while the imaginary components specify sinusoids that are odd functions, or functions having the property

$$f(t) = -f(-t)$$

The process of Fourier transformation is represented analytically by the function[2,3]

$$H(f) = \int_{-\infty}^{\infty} h(t)\cdot e^{-j\cdot 2\cdot \pi \cdot f\cdot t} dt \qquad (1)$$

where: j is $\sqrt{-1}$,

h(t) is the time function to be transformed,

H(f) is the Fourier transform of h(t),

t is time,

f is frequency.

If the Fourier transform of a spectrum is known, the time function may be determined from the inverse transformation which is given by[2,3]

$$h(t) = \int_{-\infty}^{\infty} H(f) \cdot e^{j \cdot 2 \cdot \pi \cdot t \cdot f} df \qquad (2)$$

The general form of the inverse function is essentially the same as the direct or forward transform with the exception of the sign of the exponential argument.

Since a digital computer can only deal with discrete data points, the integration indicated by the expression of the CFT can not be accomplished. A method known as the DFT was developed to approximate the CFT at discrete frequencies. The DFT is represented mathematically as[2-4]

$$H\left(\frac{n}{N \cdot dt}\right) = \sum_{k=0}^{N-1} h(k \cdot dt) \cdot e^{-j \cdot 2 \cdot \pi \cdot k \cdot n/N} \quad \text{for } n = 0,1,\ldots,N-1 \qquad (3)$$

The mathematical expression of the inverse DFT is[2-4]

$$h(k \cdot dt) = \frac{1}{N} \sum_{n=0}^{N-1} H\left(\frac{n}{N \cdot dt}\right) \cdot e^{j \cdot 2 \cdot \pi \cdot n \cdot k/N} \quad \text{for } k = 0,1,\ldots N-1 \qquad (4)$$

where in addition to those terms explained for equation (1): N is the number of sample input values from h(t) or H(f), n is a function of frequency, and k is a function of time.

Observation of the definition of the DFT reveals that there are approximately N complex multiplications and about the same number of complex additions required to compute the spectrum at a particular value. The functions are periodic with period N, however, only N/2 of the spectral components are unique. The total number of computations required to generate a complete spectrum is of the order of N^2.

An algorithm for efficiently computing the DFT of a time series was reported by J. W. Cooley and J. W. Tukey[5] in April of 1965. This method demonstrated that there was an algebraic structure in the computation of discrete Fourier transforms that could be exploited to speed up the computation of such transforms by orders of magnitude. The finite Fourier transform of a series of N (complex) data points could be computed in approximately $N \log_2 N$ operations as opposed to the N^2 operation method (Table 1). Because of the computational time reduction this method became known as the FFT.

The FFT algorithm requires that the number of elements to be transformed is an integer power of 2, i.e.: $N = 2^k$, where k is an integer in the range 2-10 inclusive. A derivation of the FFT algorithm for evaluating the Fourier coefficients for the case of N = 8 is illustrated in Fig. 1. Although N = 8 is quite limited in application, the signal flow graph may be employed to exemplify the form of the general computational algorithm.

In the signal flow graph, merging paths in a given column represent a combination of two quantities in the preceding column. For example, the first quantity in the second column, $X_1(0)$, is obtained by the combination $X(0) + W^o \cdot X(4)$ where the first term is indicated by the dashed line, the second term is indicated by the solid line, and the integer in the circle gives the power of W.[3]

Table 1. Comparison of Required Multiplication Operations Using the FFT and the Direct DFT.[4]

N	N^2 (direct DFT)	$N \log_2 N$ (FFT)
64	4,096	768
128	16,384	1,792
256	65,536	4,096
512	262,144	9,216
1,024	1,048,576	20,480

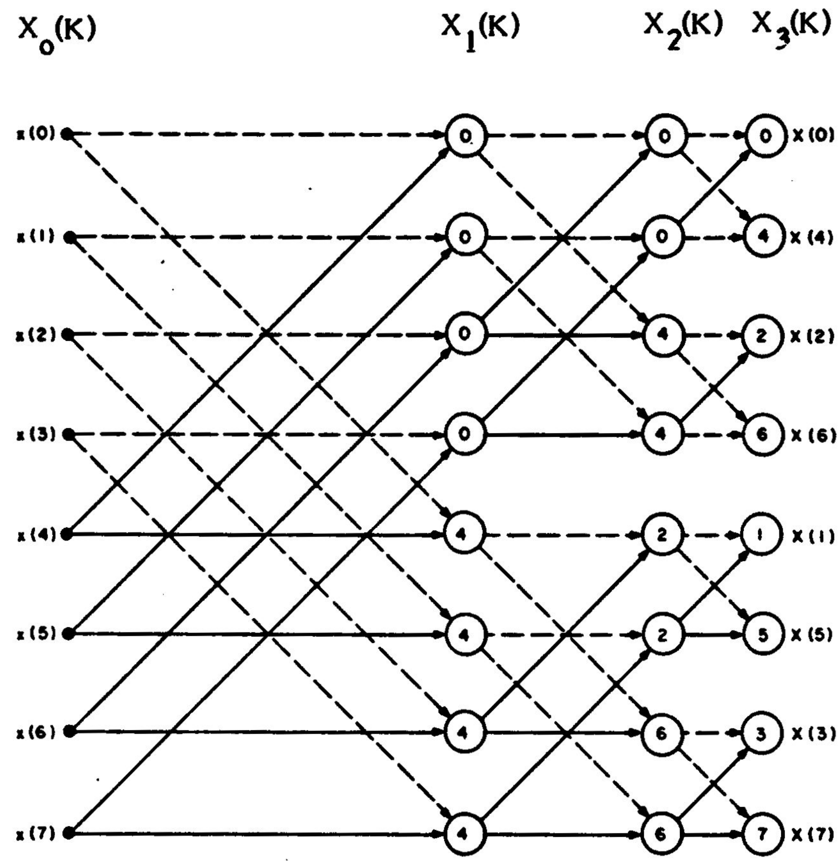


Fig. 1. Flow diagram indicating the computations involved in a 8 point FFT implementation of the DFT function. Merging paths represent a combination of two quantities in a preceding column. For example, $X_1(0) = X(0) + W^0 * X(4)$.[3]

If k•dt is replaced by k and n/N dt by n, for convenience of notation, equation 3 may be rewritten as[6]

$$X(n) = \sum_{k=0}^{N-1} X_o(k) W^{n \cdot k} \qquad (5)$$

where $W = e^{-j \cdot 2 \cdot \pi / N}$

The process continues until finally the spectral coefficients are obtained, the last column of X's, in scrambled order, X(0), X(4), X(2), X(6), X(1), X(5), X(3), X(7). The generality of this order may not be readily apparent but can best be described by looking at the subscripts as binary numbers:

0	000	4	100	2	010	6	110
1	001	5	101	3	011	7	111

If these subscripts are bit-reversed in binary, or read from right to left, the numbers from 0 to 7 are obtained in their natural order:[7]

000	0	001	1	010	2	011	3
100	4	101	5	110	6	111	7

The results of the FFT algorithm are identical to the results obtained by the DFT algorithm differing only by a known scale factor. The results of the FFT and DFT only resemble the expected results of the CFT, largely because of different inputs. Only under certain special conditions do the FFT and the DFT produce the same results as the CFT for corresponding frequencies. In order to get agreement between the results of the FFT and the results of the CFT three conditions must be satisfied.

1. The function to be transformed must be periodic and band-limited, the function's highest frequency component must be finite.
2. The function to be transformed must be truncated at exactly one non-zero integer multiple of the functions period.
3. The function to be transformed must be sampled at a rate greater than twice the functions largest frequency component.

If the third condition is not met, if the sampling rate is too low, the period of the function will be too short and aliasing (the overlapping of representations of the expected transform resulting in false frequency contributions) results. If either T, the sampling time, or N, the number of points, is not chosen such that the original function is truncated at an integer multiple of the period, the second requirement is not met, the resulting spectrum will be distorted as a result of the leakage effect (a spreading of the spectral components away from the current frequency). If the first requirement is not met either the second or the third condition can not be met.[8,9]

Chemical Applications of the FFT

The FFT method has been applied in the analysis of a number of analytical techniques in the chemical laboratory. A few of these techniques include multiplex gas chromatography, linear parameter estimation of fused peak systems, and the interpolation of sampled data obtained by chromatographic, spectroscopic, and electrochemical techniques.

Multiplex gas chromatography.--In multiplex gas chromatography instead of injecting a single impulse of the sampled mixture into the carrier gas stream, as in regular chromatographic techniques, multiple sample injections are made. The chromatogram obtained contains peaks which are severely overlapped resulting in an output not directly interpretable by the chemist. The information may be recovered and presented in the form of a chromatogram by applying fairly simple computational techniques; a Fourier transformation followed by cross-correlation. Although this technique does not help reduce the minimum time for an experiment or allow a time/resolution trade off, for applications where repetitive chromatograms are needed either to enhance the signal to noise ratio or to continuously monitor a sample stream the advantage attainable with this technique becomes important.[10]

Linear parameter estimation of fused peak systems.--Because Fourier transforms are sensitive to changes in the normally used peak parameters- peak height, peak position, and general peak shape- a change in any of these parameters will be reflected in a change in the real and imaginary spatial frequencies. By performing the linear least squares parameter estimation calculation in the Fourier domain a significant improvement in the resolution of fused peaks in a fused peak system was observed with a substantial savings in computation time relative to the traditional approach.[11] This technique is not discipline specific and therefore can easily be applied to other areas of analytical chemistry which suffer from fused peak systems.

Interpolation of sampled data.--Precise identification of the magnitude and position of peak-type responses, as well as the recognition of peak-shoulder combinations, in chromatography, spectroscopy, and electrochemistry may be hindered by inadequate resolution of sampled data. An interpolation technique for drawing a smooth curve through the sampled data is needed to overcome these difficulties. One such approach, which has been successfully implemented by Griffiths[12] in processing spectroscopic data, involves the use of Fourier domain (FD) interpolation. With this technique the FD spectrum of the data array to be interpolated is computed, extended by a factor of 2^n, where n is a positive integer, by zero filling,[12] and inverse Fourier transformed. The resulting array contains 2^n times the number of points in the original array. The validity of this procedure rests upon the fact that although the original data array may not provide sufficient resolution to satisfactorily serve its intended purpose, it is adequate to define the FD spectrum.[13] A sufficient reduction in the total measurement time and computer memory requirements are attained by acquiring fewer than normal points along the time, frequency, or dc potential axis.

FFT as a Digital Filter

One of the numerous applications of the FFT is digital filtering (Fig. 2) which in the broadest sense refers to the assessment of the frequency domain effects of any digital system or processing algorithm.[14,15] The initial step in the filtering process involves the computation of the DFT of the input signal. Since the output of the Fourier transformation gives an estimate of how much each frequency component contributes to the original signal it can be employed to analyze a complex waveform into a set of simple additive sinusoidal components, Fig. 3. Next, the DFT is modified by the desired frequency response directly in the frequency domain, allowing for the removal or filtering of unwanted frequency components. The inverse DFT is then computed; the resultant being the filtered signal, Fig. 4.[16] This concept represents a rather interesting approach to digital filtering in the sense that frequency response functions completely unattainable with rational transfer functions may be applied directly to the spectrum.

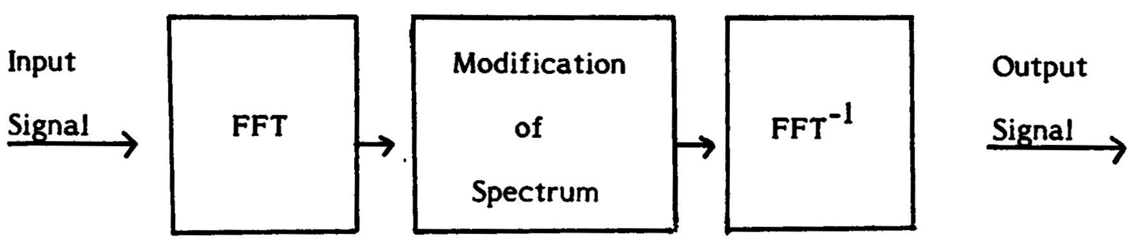
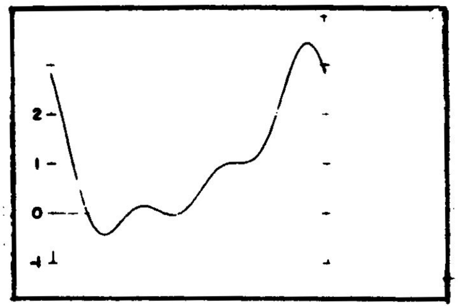


Fig. 2. Digital filtering using the FFT.

a)



b)

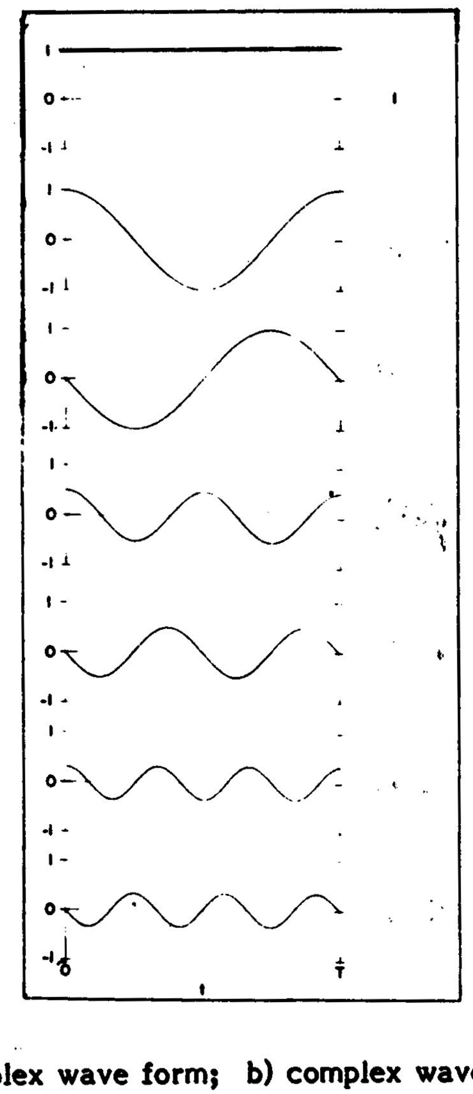


Fig. 3. a) The complex wave form; b) complex wave form analyzed into the seven simple sinusoidal components.[15]

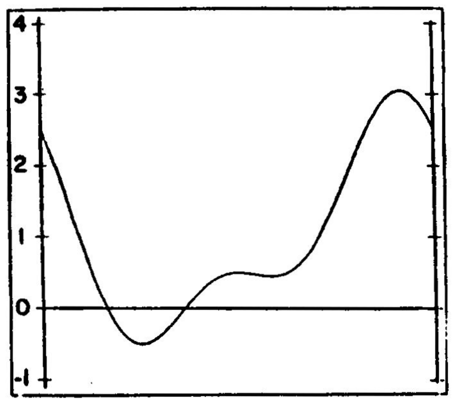


Fig. 4. A filtered version of the wave shown in Fig. 3a. This wave is reconstructed from only the first five of the seven components of Fig. 3b.[15]

EXPERIMENTAL

Materials and Methods

A FORTRAN-IV computer program, which reads in a data file and performs a foward and an inverse Fourier transform, was written and implemented on a Digital Equipment Corporation (DEC) PDP 11/34 minicomputer. The hardware and software which were used in the implementation are described below.

Hardware

PDP 11/34 System.--The DEC PDP 11/34 is a 16-bit minicomputer with an RT-11 (Real Time-11) operating system which supports either a single job (SJ), a foreground/background (FB) or an extended monitor (XM). The minicomputer has approximately 50 Kbytes of user (or program) memory space. For mass storage it utilizes three RK05 cartridge disks, each disk being able to hold over 2.4 Mbytes of information. The languages available on this system are FORTRAN IV, BASIC, PASCAL, and the PDP-11 assembly lanaguage, MACRO.

4025 computer display terminal.--The 4025 computer display terminal, a product of Tektronix, Inc., belongs to a class of machines popularly known as "smart terminals" because it has its own microprocessor and responds to its own set of commands, independent of the host computer. The 4025 has basic graphic capabilities and is equipped to draw several styles of vectors (line segments) and to intermix graphics with text and forms.[17] There are several commands designed for creating graphic displays on the 4025

1. WORKSPACE-This command initializes the screen so that an applications program may be run. The syntax of this statement is:
!WORKspace [<number>] [Host] [Keyboard] <CR> where <number> is an integer between 0 and 33, inclusive. If number is included, this command erases the entire display list, defines a workspace, and allots the top number lines of the screen for the workspace window. The remaining 34-<number> lines are used for the monitor window. If H (Host) is specified, text from the host computer is directed into the workspace. If K (Keyboard) is specified, text from the keyboard is directed into the workspace.

2. MONITOR-This command allows the user to specify which device (Host or keyboard) will send information to the monitor and to create text windows. The syntax of this statement is:
!MONitor [<number>] [Host] [Keyboard] <CR> where number is an integer between 1 and 34, inclusive. If <number> is included this command erases the entire display list. The terminal then defines a workspace and reserves the top 34-<number> lines of the screeen for the workspace window, using the remaining <number>lines for the monitor window. If H (Host) or K (Keyboard) is specified, then text from the computer or keyboard is directed into the monitor.

3. GRAPHIC-This command defines a graphic region in the 4025 workspace and erases all information currently stored in this region. The syntax of the statement is:

! GRAphic <beginning (beg) row> <end row> [<beg column (col)> [<end col>]] <CR> where all parameters are positive integers designating rows and columns in absolute workspace coordinates. The default values of <beg col> and <end col> are 1 and 80, respectively.

4. SHRINK-This command is used to "shrink" the coordinates of graphic information by a factor of approximately 5/8. This is done because the 4025 accepts 4010-style graphic commands from a host computer and, therefore, it is necessary to scale the incoming 4010 commands for display in the 4025 graphic region. The syntax of this command is: ! SHRink [Yes/Hardcopy/Both/No] <CR>. The default parameter is Yes.

5. ERASE G-This command erases the contents of the graphic region without destroying it such that new graphic information and text can be displayed there. The syntax of the statement is:
! ERAse G <CR>

4662 interactive digital plotter.--The 4662 interactive digital plotter, developed by Tektronix, Inc., provides permanent graphic recording capabilities for devices such as the Tektronix 4010-series terminal based system. Paper sizes of up to 11 inches (27.9 cm) in Y by 17 inches (43.2 cm) in X can be employed. The page scaling feature of the 4662 allows the plot size, which has a maximum value of 10 inches (25.4 cm) by 15 inches (38.1 cm), to be easily adjusted from the front panel to suit the paper size. The paper is held in position by electrostatic attraction generated by the platen.

There are three basic types of operations which may be performed with the 4662. The Hardware Alphanumerics feature of the plotter allows alphanumeric text to be printed on plots without requiring character generation software support by the host system. The plotter can also produce graphics by moving the pen across the plotting surface, lifting and lowering it to produce written vectors only when desired. In addition, the GIN (Graphic Input) operation allows the plotter to act as a digitizer, transmitting the coordinate position of the pen along with the pen status (up or down) upon command.[18]

Software

Plot 10-terminal control system.--The Terminal Control System software package, which was developed by Tektronix, consists of a comprehensive set of subroutines which allows terminal-independent graphic programming. It offers many graphing conveniences to the user; bright and dark vectors (line segments) as well as points may be displayed on the terminal screen. In addition, the software package also includes a choice of linear, logarithmic, or polar coordinate systems; automatic scaling of graphic data; and buffered input and output for faster, more efficient character handling.

The fast Fourier transform subroutine.--The laboratory subroutines package, developed by DEC, is a set of eight data-processing subroutines designed to be used in a laboratory environment. The FFT subroutine is one of these routines and provides an efficient means of numerically approximating analytical or continuous Fourier transforms. The subroutine is supplied in two forms: an object file (F4FFT.OBJ) which is ready to be linked with the application FORTRAN code and a source file (F4FFT.MAC) which may be used to produce an object file having features different from those of the distributed object file by defining certain conditional assembly parameters. The conditional assembly parameters that may be defined are the Extended Instruction Set (EIS), the Extended Arithmetic Set (EAS), and the maximum input/output (I/O) array size (F.MAXN).

The EIS and EAS are hardware packages which if available on your installation and so defined will increase the execution speed and decrease the memory requirements for the subroutine by 86 and 80 words, respectively. F.MAXN specifies the maximum number of complex elements that the FFT subroutine can process at one time. It is limited to those multiples of 1K (1024) that are powers of 2 in the range 1K to 8K (8192). F.MAXN is not defined on the distributed object file and if it is not defined by the user a default value of 1024 is assumed. Redefining F.MAXN increases the memory requirements for the FFT subroutine by increasing its size by 256 words, going from 1024 to 2048, by 768 words, going from 1024 to 4096, and by 1792 words, going from 1024 to 8192.[9]

The general format of the FORTRAN call to the FFT subroutine is

CALL FFT(IERROR,N,IREAL,IMAG,INVRS, ISCALE).

IERROR is an integer variable used to signal error conditions. The values IERROR can assume are:

0 = no error has occurred

1 = N is less than eight

2 = N is greater than 4096 or F.MAXN

3 = N is not a power of two

-n = The N^{th} argument is missing.

If IERROR is omitted, the program generates a fatal error when the FFT subroutine is called. N specifies the number of elements to be transformed and must be an integer variable that is a power of 2 between 8 and the maximum array size F.MAXN. IREAL is an integer array N elements long which contains the real portion of the input data to be transformed. The FFT returns the real results to this array replacing the input data. IMAG is an integer array N elements long which contains the imaginary portion of the input data to be transformed. The FFT returns the imaginary results to this array replacing the output data. INVRS is an integer variable that indicates whether a forward or inverse transform is to be performed. The two values that can be used for INVRS are:

0 - a forward transform is to be performed

1 - an inverse transform is to be performed.

ISCALE is an argument that the FFT subroutine sets. It is an integer variable that indicates the number of times the results of the FFT subroutine have been divided by two. The FFT subroutine sets the scaling factor as necessary to overflow. In order to obtain the unscaled results of the FFT subroutine each output element of arrays IREAL and IMAG should be multiplied by

$$(\text{real})\ \text{ISCALE}^2.$$

RESULTS AND DISCUSSION

The main program, RCFFT, as developed in this thesis (see Appendix), exemplifies how the FFT subroutine may be used as a digital filter to eliminate unwanted noise in spectroscopic data. In addition, it provides the user with the option of attaining a graphical representation of the data as each step: 1) reading the data in, 2) performing a forward transformation, 3) performing an inverse transform, and 4) scaling of data; is performed and of writing the results of either the forward or inverse transform out to a file. Listings of the main program and the subroutine subprograms appear in the appendix.

Main Program

In the RCFFT main routine the user is asked to input the number of files to be read, the maximum value of which is four, the file name for each file, and the number of points per file. The maximum number of points per file is then found.

The file or files are then read in, a XPNTS (magnitude) and a YPNTS (time) value. The elements of the imaginary array (IM) are set to zero and the elements of the real array (IR) are set to the integer conversion of the time array, in order to be called by the FFT subroutine.

The user is then given the option of a graphical representation of the data, to which either a Y (yes) or N (no) reply should be given. If yes is specified, subroutines RCGRSP, which creates a graphic workspace on the 4025, and RCGRAF, the graphics routine, are called.

RCFFT then calls the FFT subroutine to perform a forward transformation on arrays IR and IM. If an error is generated, IE, which is the same as IERROR in the argument list of subroutine F4FFT, does not equal zero, the error code is printed and execution is terminated. Similarly, if the scaling factor (ISCALE), ISF in the calling list, does not equal to zero, the scaling factor is printed and execution terminates. The user is then given the option of calling subroutine WRITFL, which will write the results of the forward transform, arrays IR and IM, out to a file. Arrays IR and IM are converted from integer to real and subroutines RCGRSP and RCGRAF are called to graph the forward transform of data.

The main routine then gives the option of an inverse transform. If no is specified, execution ceases. If a yes response is given, the user is given the option of performing the filtering process. The X value corresponding to the initial frequency value (IFMIN) and the final frequency value (IFMAX) are inputed and the elements of the array IR and IM within this range are set equal to zero. The FFT routine is called and an inverse transform is done. If an error occurs during the execution of the FFT routine or the value of the scaling factor does not equal zero, a message is generated. The option of writing the results of the inverse transform out to a file is given (WRITFL). Then, subroutines RCGRSP and RCGRAF are called and the results of the inverse transform are graphed.

The program gives the option of starting over so that a new frequency range may be input. If yes is specified, control is transferred to the read statement (statement 9999) at the beginning of the program. The initial file is read in and execution proceeds as outlined above. Otherwise, execution continues and the option of receiving a graphical representation of the data after scaling is given. A no response terminates program execution.

The scaling factor is calculated as

$$SCAL = 2.0**(ISF + ISI) \tag{6}$$

Arrays IR and IM are multiplied by the scaling factor, so that the data is of the same magnitude as when it was input, subroutines RCGRSP and RCGRAF are called and the results of arrays IR and IM after scaling are graphed.

Subroutines

Graphics Routine.--The general format of the FORTRAN call to the graphics routine, RCGRAF is

CALL RCGRAF(NFILES,NDATUM,MDATUM,IDUMMY,ITORF,JI).

NFILES is an integer variable specifying the number of data files to be read in.

NDATUM is an integer array containing the number of points per file for each file read in.

MDATUM is an integer variable which contains the maximum value of array NDATUM, or the maximum number of points per file.

IDUMMY is an integer constant which indicates whether array YPNTS, which is equivalenced to array YIRM in the main routine, is composed of one or more sub-array. The maximum number of sub-arrays comprising array YPNTS is eight (when NFILES is equal to four). The only two values IDUMMY can assume are:

1- YPNTS is composed of one sub-array (you are graphing the input data).

2- YPNTS is composed of more than one sub-array (you are graphing the forward transform, the inverse transform, or the data after scaling.)

ITORF is an integer array which indicates the domain of the data being graphed. This array contains the ASCII code for either the word time or frequency depending upon the data being graphed.

JI is an integer constant which is the dimension of the ITORF array. The values JI can assume are:

4- ITORF is the ASCII code for time.

9- ITORF is the ASCII code for frequency.

Before the data can be graphed it is necessary to obtain the minimum and maximum values of the data so that the range may be determined for both the X and Y values. This may be accomplished with subroutine MINMAX

CALL MINMAX(XPNTS,MDATUM,NDATUM,NFILES,XMIN,XMAX) ,

which was written by Terry J. Green.

Subroutine RCGRAF checks the value of the IDUMMY argument to determine the number of sub-arrays comprising array XPNTS. If the value of IDUMMY equals one, subroutine MINMAX is called and the minimum (YMIN) and maximum (YMAX) values are determined for the YPNTS array. Control is transferred to line 2000 and execution continues, with the value of YRANGE being calculated.

If the value of IDUMMY equals two, control is transferred to line 1000, the minimum value (YMIN) and the maximum value (YMAX) are found and the range (YRANGE) is calculated.

The Terminal and Terminal Status Area are initialized so that the Terminal Control System routines may be employed. This is accomplished by calling the initializing routine, INITT which:

1. Erases the screen and causes the cursor to be moved to the HOME position.
2. Sets the Terminal to alphanumeric mode.
3. Sets the margin values to the left and right screen extremes.
4. Defines the window so that the portion of virtual space will be displayed which is equivalent in coordinates with the screen.

The rate of character transmission from the computer to the Terminal, the baud rate, is required as an input parameter with the general format of the statement being

Call INITT (IBAUD)

with IBAUD = 120 in this case.

The screen window (display area) is defined by calling subroutine TWINDO

CALL TWINDO (300, 900, 200, 600).

The general format of this command is

CALL TWINDO (MINX,MAXX,MINY,MAXY).

where MINX is the minimum horizontal screen coordinate, MAXX is the maximum horizontal screen coordinate, MINY is the minimum vertical screen coordinate, and MAXY is the maximum vertical screen coordinate.

Subroutine MOVABS is called and a move to (300, 600) is generated so that drawing can proceed from there. The X and Y axes are drawn by issuing calls to subroutine DRWREL, which is used to generate a bright vector by indicating the number of horizontal and vertical screen units to move relative to the last beam position.

A virtual window is defined by calling subroutine DWINDO in order to fit the data to the screen window. Any graphic lines (vectors) or portions of vectors which lie outside of the area specified are clipped.

A move to (X, YMIN) is generated in virtual units (MOVEA), where X is equal to XRANGE initially and is incremented by X each time this sequence of steps reoccurs. Subroutine DTIC (down tic), which is contained in the PLOT 10 library, is called and a tic mark is generated at the point on the X axis specified by the MOVEA command. ASCODE, a routine written by Terry J. Green to determine the ASCII code of a number, is called. ASCODE requires as input the double precision conversion of the number, returning the ASCII code array (ITRANS) and the dimension of (or number of characters in) the array. A move relative to the current cursor position is generated and subroutine ANSTR, which is part of the PLOT 10 library, is called to print out the X value. Once the X axis tic marks and values have been printed, subroutines MOVABS and ANSTR are called and the X axis is labeled, either TIME or FREQUENCY, depending upon the domain of the data being graphed.

An analogous procedure is followed to generate the Y axis tic marks (subroutine LTIC) and to print out the y values on the axis. Subroutines MOVABS and ANSTR are called and the Y axis label (MAGNITUDE) is printed.

A FORTRAN call to subroutine MOVEA is generated and the cursur is moved to the origin, (XMIN, 0.0). Subroutine LTIC is called and a tic mark is generated at this position. MOVREL and ANSTR are called and a zero is printed at the origin.

The input parameter IDUMMY is checked to determine the number of sub-arrays comprising array YPNTS. If IDUMMY equals one, control is transferred to line 3000. Then the MOVEA routine is used to position the cursur at the initial X, Y value,

(XPNTS(1,1), YPNTS(1,L))

and subroutine DASHA is called to plot out the data

CALL DASHA (X,Y,L)

where L is the dash type specification and can assume the values:

1- a dotted line.

2- a dash-dot line.

3- a short-dashed line.

4- a long-dashed line.

The sequence is repeated until the file or files are graphed.

If IDUMMY equals two, the total number of files, NTFILE, is equal to twice the number of files input

NTFILE = 2·NFILES.

Therefore another method of graphing the data must be employed. Two counters N and IN which are used to manipulate the loop indexing are initialized to one. Subroutine MOVEA is called to move to the initial X,Y value (IREAL(1,1)). Subroutine DASHA is called and the array IREAL is plotted for the first file. The value of IN is incremented. A move is generated to the initial X,Y value of the imaginary array. Subroutine DASHA is utilized to plot out the imaginary array for the first file. IN is incremented by one so that its value is now three. The value of N is incremented and IN is set to one, indicating that a new file is being graphed, and execution continues as outlined above with first the real and then the imaginary arrays of each file being graphed. When the total number of files have been graphed a return to subroutine statement is executed.

Write File Routine.--Subroutine WRITFL may be utilized to write the results of either the forward or inverse transformation out to a file. The general format of the calling statement for the write file routine is

CALL WRITFL(IR,IM,NFILES,NDATUM,MTRANS,M)

where IR is an integer array N elements long that contains the real portion of the data after transformation, IM is an integer array N elements long that contains the imaginary portion of the data after transformation, NFILES is an integer variable that indicates the number of files read in, NDATUM is an integer array that contains the number of points per file for each file read in, and MTRANS is a variable which indicates whether a forward or inverse transform has been performed. It assumes the values FORWARD or INVERSE. M is an integer variable which specifies the logical unit which should be opened for output to a file.

Subroutine WRITFL initially checks the value of MTRANS to determine whether a forward or an inverse transform has been performed. If MTRANS equals INVERSE control is transferred to line 300 and the user is asked to input the file name for the inverse transform. Otherwise, a forward transform has been performed, the user specifies the file name for the results of the forward transform and control is transferred to line 500.

A FORTRAN call to subroutine ASSIGN, which is one of the routines contained in the PDP 11/34's library, is executed. This routine associates a device and a file name with a logical unit number (M) and must be executed before a logical unit is opened for input/output (I/O) operations.

If an inverse transform was performed, control is transferred to line 1000 and the results of the inverse transform of data are written out to the file; the real part, array IR, and the imaginary part, array IM. Otherwise, a forward transform occurred and the results (the real part, IR, and the imaginary part, IM) are written out to the file.

A call to subroutine CLOSE is executed thus freeing the logical unit so that it may be associated with another file.

Program Application

In order to illustrate the digital filtering capability of the RCFFT program a number of sets, or files, of data (each set containing thirty-two points) were generated via computer programs employing different sine wave functions.

A graphical representation of the sine wave function

$$\mathrm{SIN(DELX \cdot 1000.0 \cdot L)} \qquad (7)$$

is illustrated in Fig. 5 with DELX being equal to two times pi and L, with an initial value of zero, being incremented by

$$\mathrm{L} + 1.953125\mathrm{E}{-4}$$

as each value of the sine is calculated.

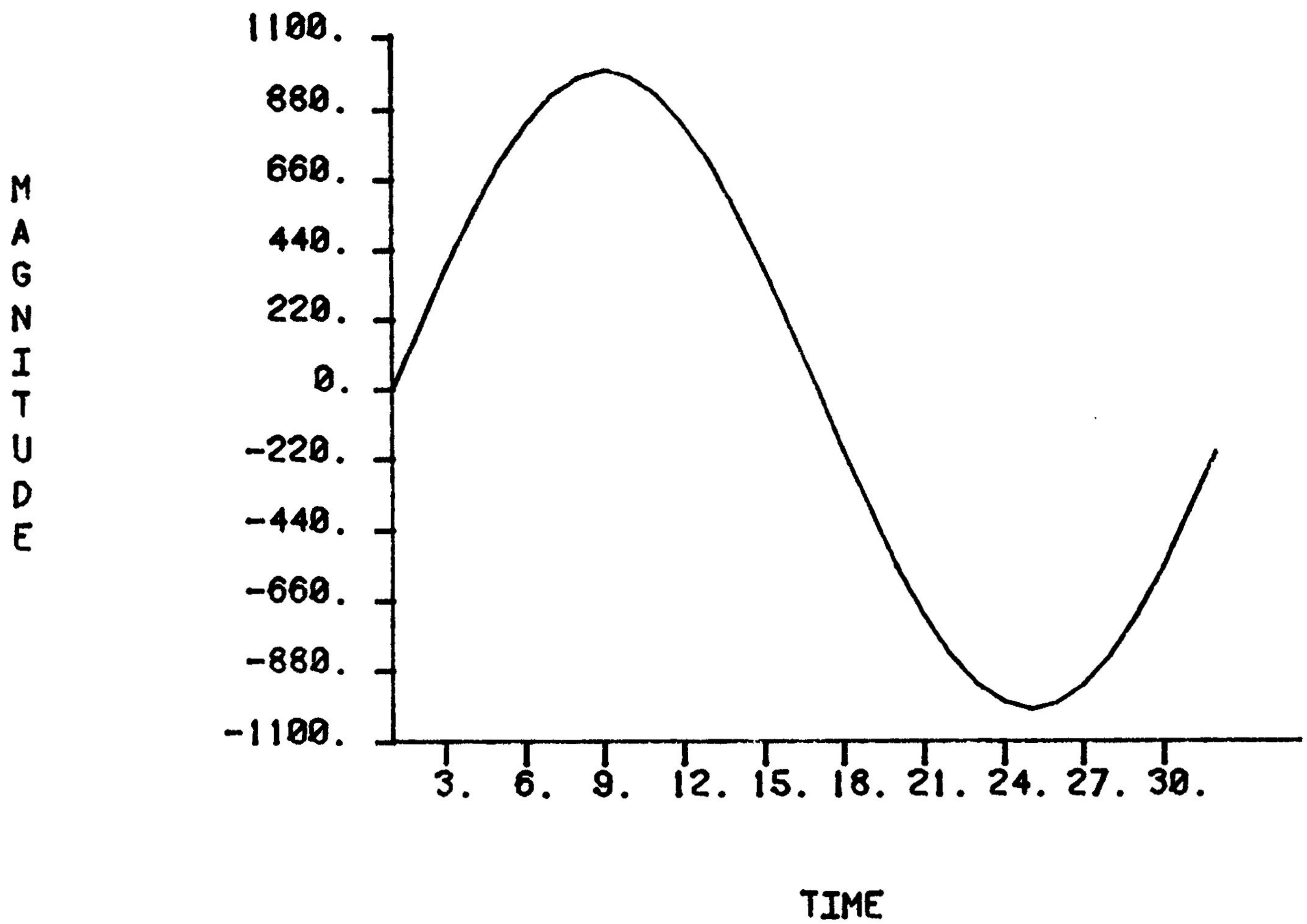


Fig. 5. Sine wave obtained by SIN(DELX*1000.0*L).

Fig. 6 depicts a sine wave with noise added to it

$$SIN(X) + RAN(L,M) \cdot SIN(X) \tag{8}$$

The FORTRAN library function RAN(L,M) which returns a random number of uniform distribution over the range 0 to 1 is used to generate the noise.

Figs. 7 and 8 depict the results of a forward transform on the data generated with equations 7 and 8 respectively. Comparing the frequency range 3-29 of the figures one notes that the results of both the real array (the solid line) and the imaginary array (the dashed line) of Fig. 7 are straight lines whereas with Fig. 8, the noisy spectrum, the lines are not. Based upon the results obtained in this step the zero filling range is chosen to eliminate the noise to attain a curve analogous to Fig. 5.

Figs. 9 through 11 illustrate the results of the inverse transform of the noisy spectrum, equation 8, with different zero filling ranges being employed. Fig. 9 depicts the result of the real array after the inverse transform with a zero filling range of 5-29, where the zero filling range is determined from Fig. 8. For illustrative purposes the result of the real arrays after the inverse transform are shown for the zero filling ranges 1-27, Fig. 10, and 21-29, Fig. 11, where both zero filling ranges are taken from Fig. 8. Because the spectrum obtained in Fig. 9 more closely resembles the spectrum in Fig. 5 the zero filling range 5-29 is the optimum.

Fig. 12 illustrates the results of the inverse transform of the noisy spectrum (same as Fig. 6) after scaling using the optimal zero filling range while Fig. 13 depicts the inverse transform of the spectrum without noise after scaling (same as Fig. 5).

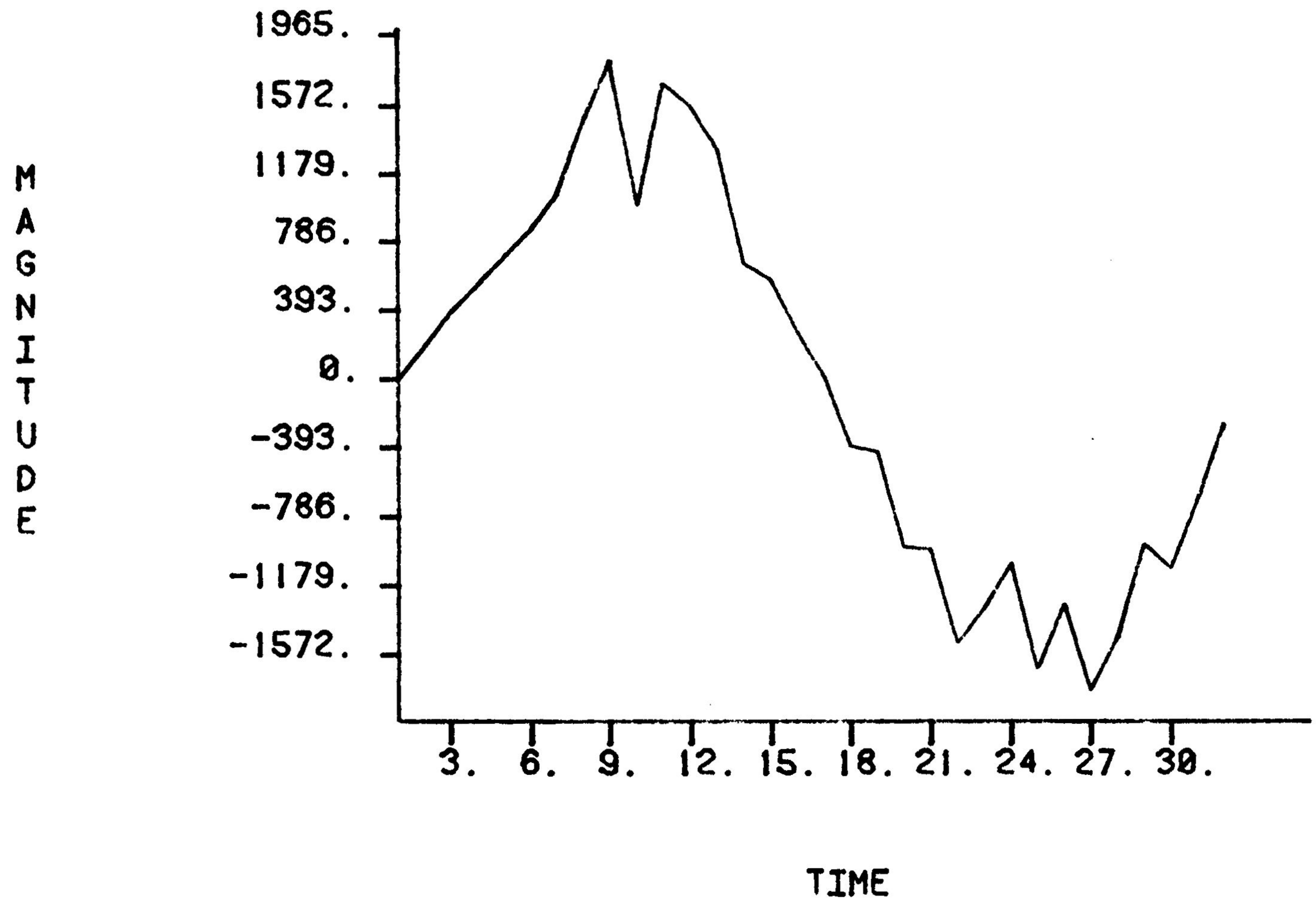


Fig. 6. Noisy sine wave obtained by SIN(X)+RAN(L,M)*SIN(X).

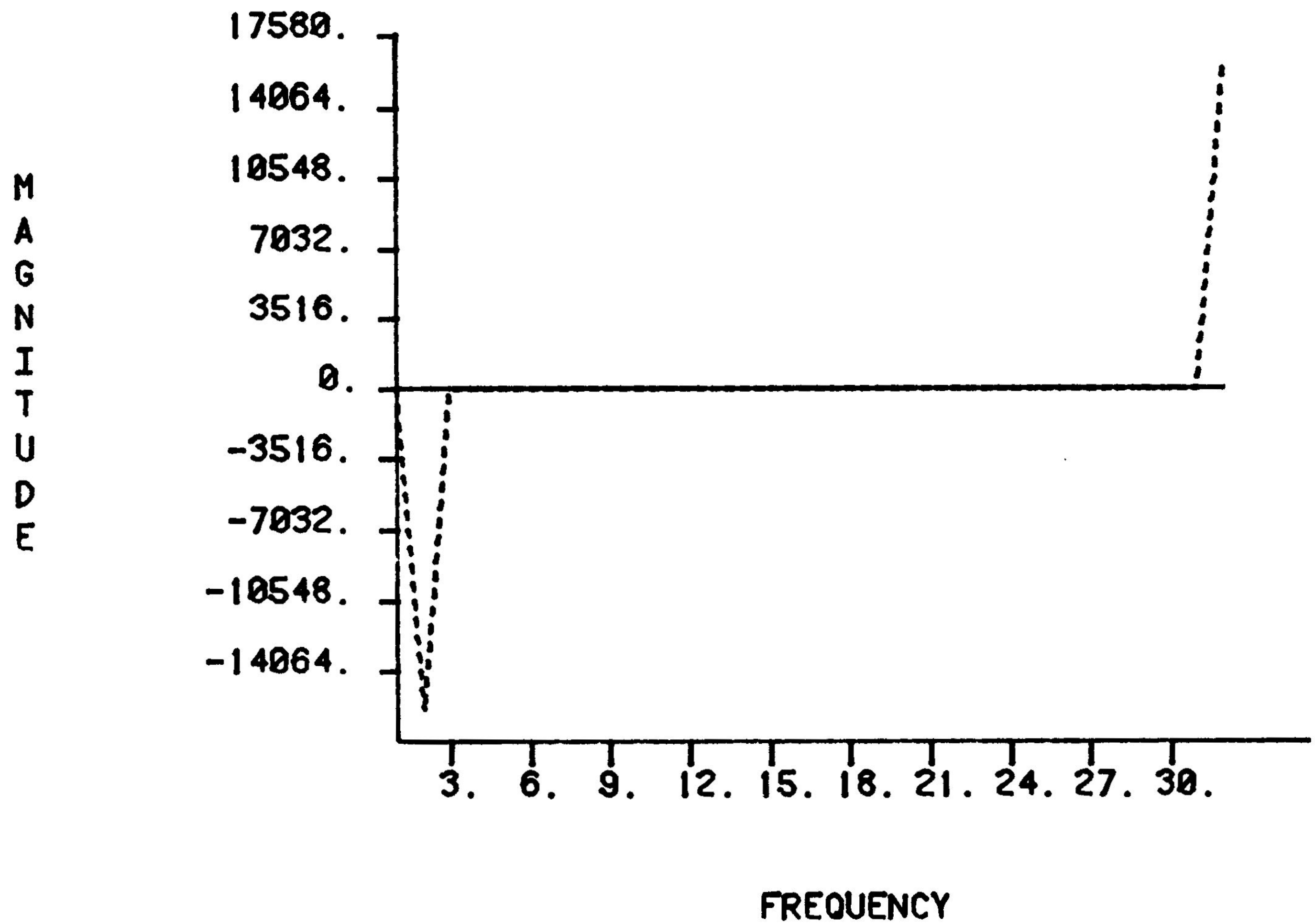


Fig. 7. Results of the forward transform on the data generated by SIN(DELX*1000.0*L), the data depicted in Fig. 5. The solid line indicates the real array and the dashed line indicates the imaginary array.

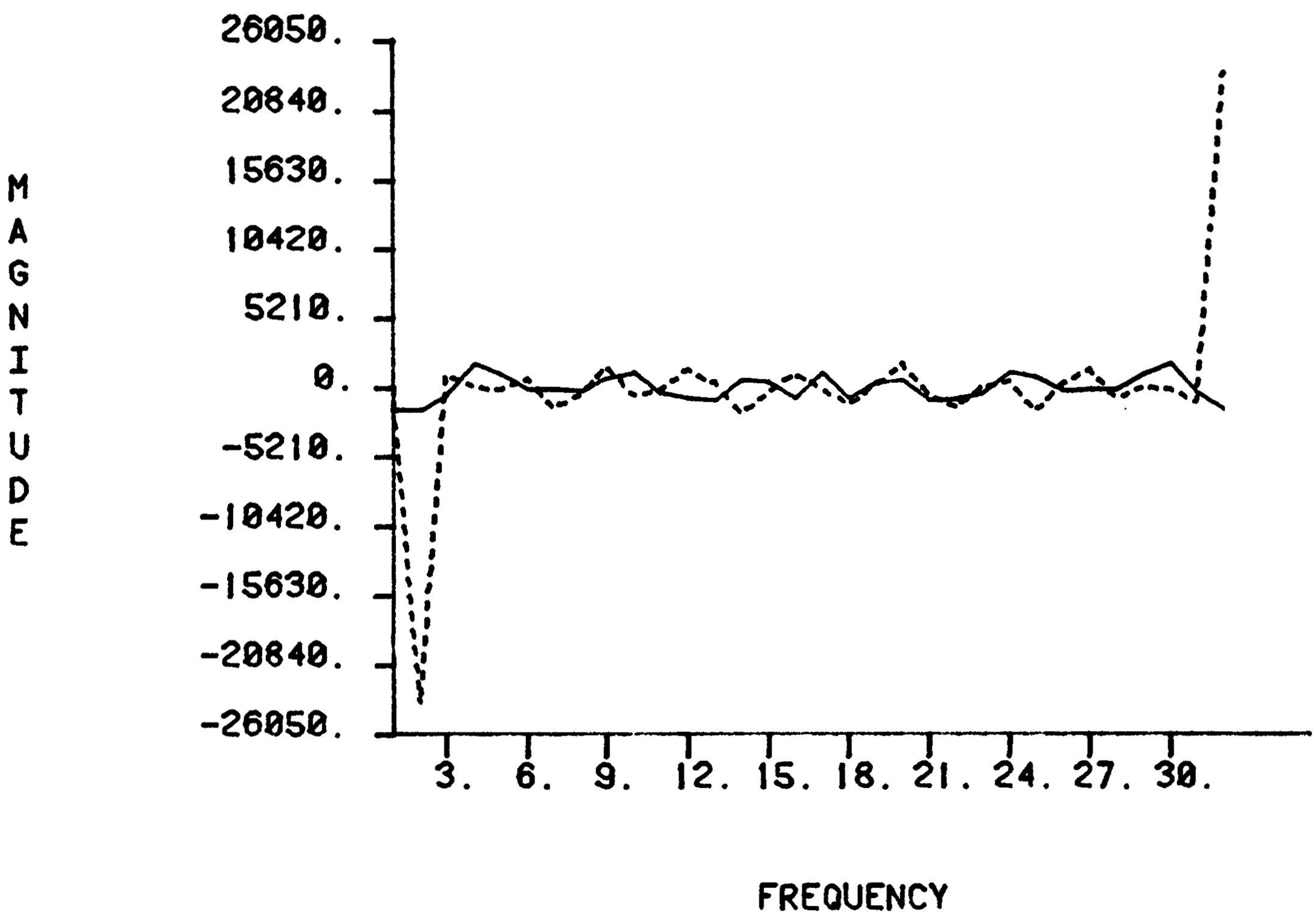


Fig. 8. Results of the forwarded transform on the data obtained by SIN(X)+RAN(L,M)*SIN(X), the data depicted in Fig. 6.

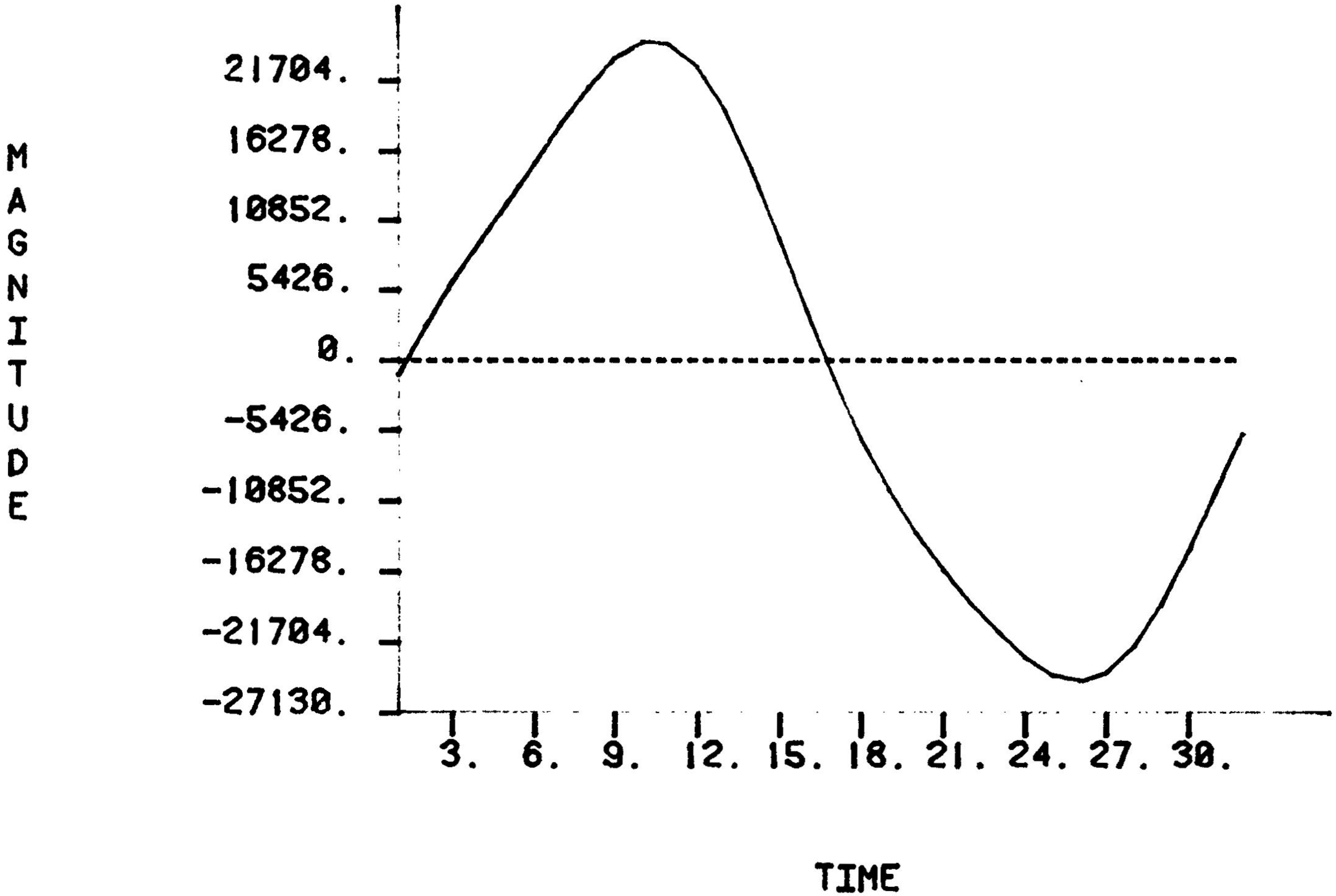


Fig. 9. Results of the inverse transform on the data obtained by SIN(X)+RAN(L,M)*SIN(X) employing the zero filling range 5-29.

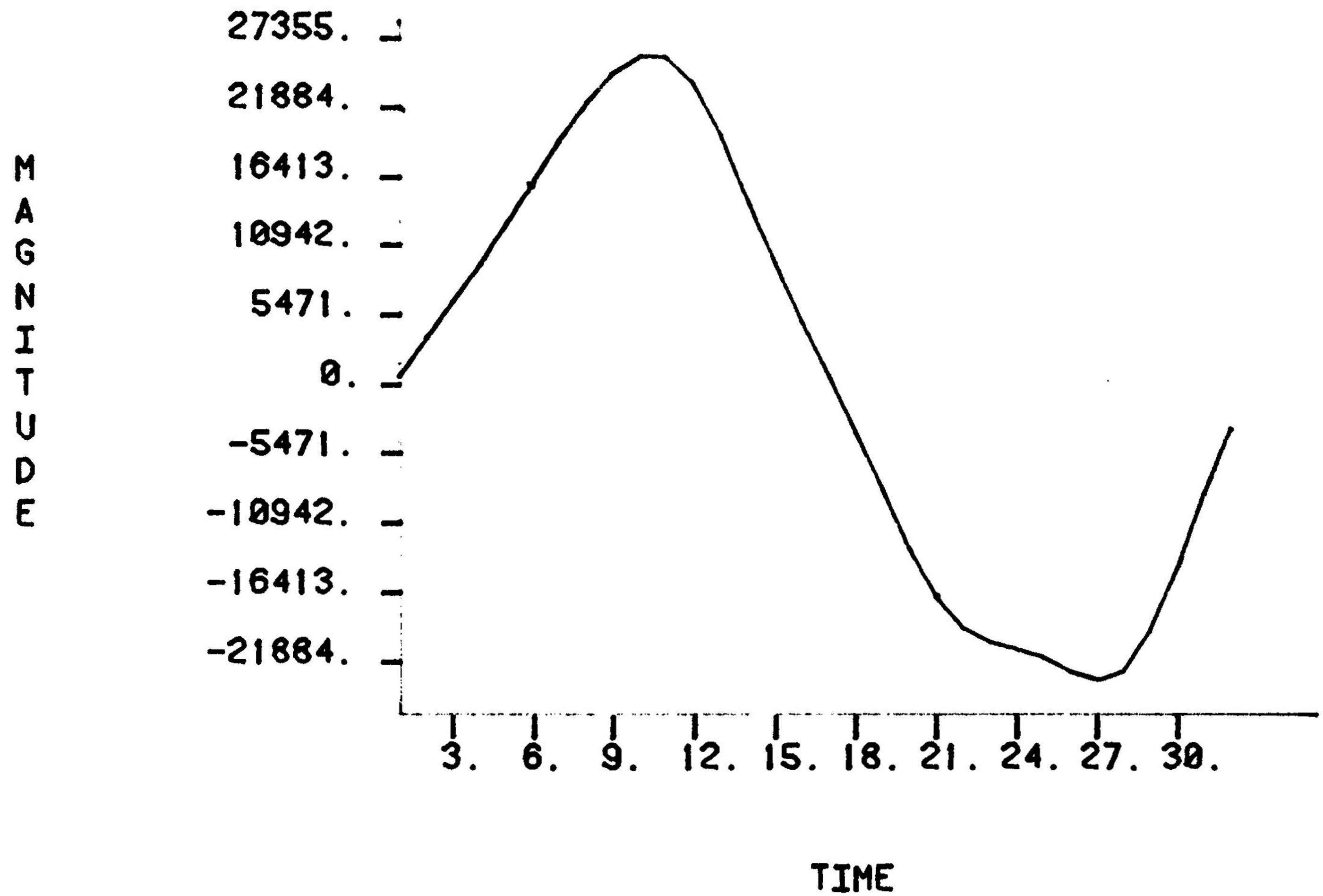


Fig. 10. Results of the inverse transform on the data obtained by SIN(X)+RAN(L,M)*SIN(X) employing the zero filling range 1-27.

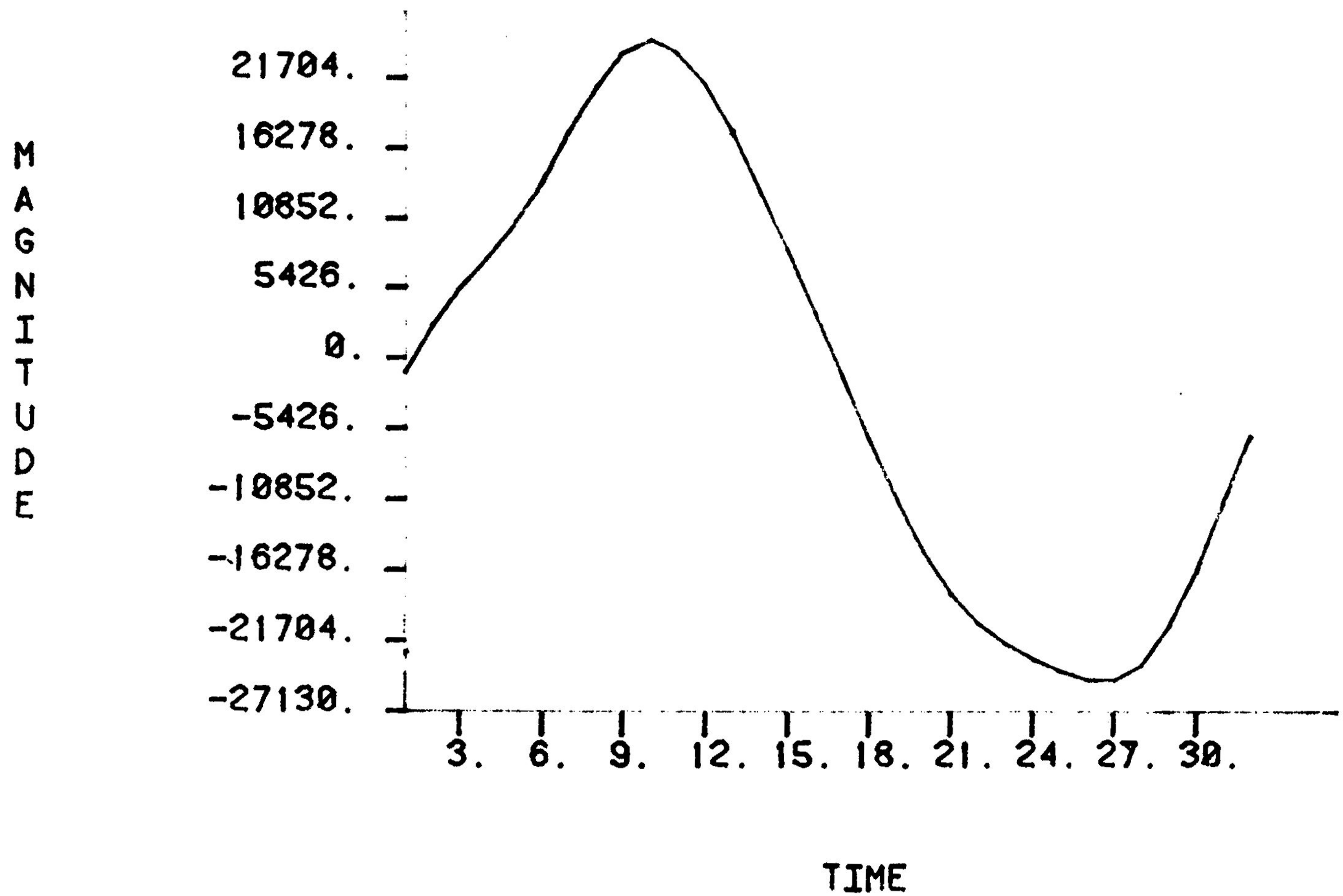


Fig. 11. The results of the inverse transform on the data obtained by SIN(X)+RAN(L,M)*SIX(X) employing the zero filling range 21-29.

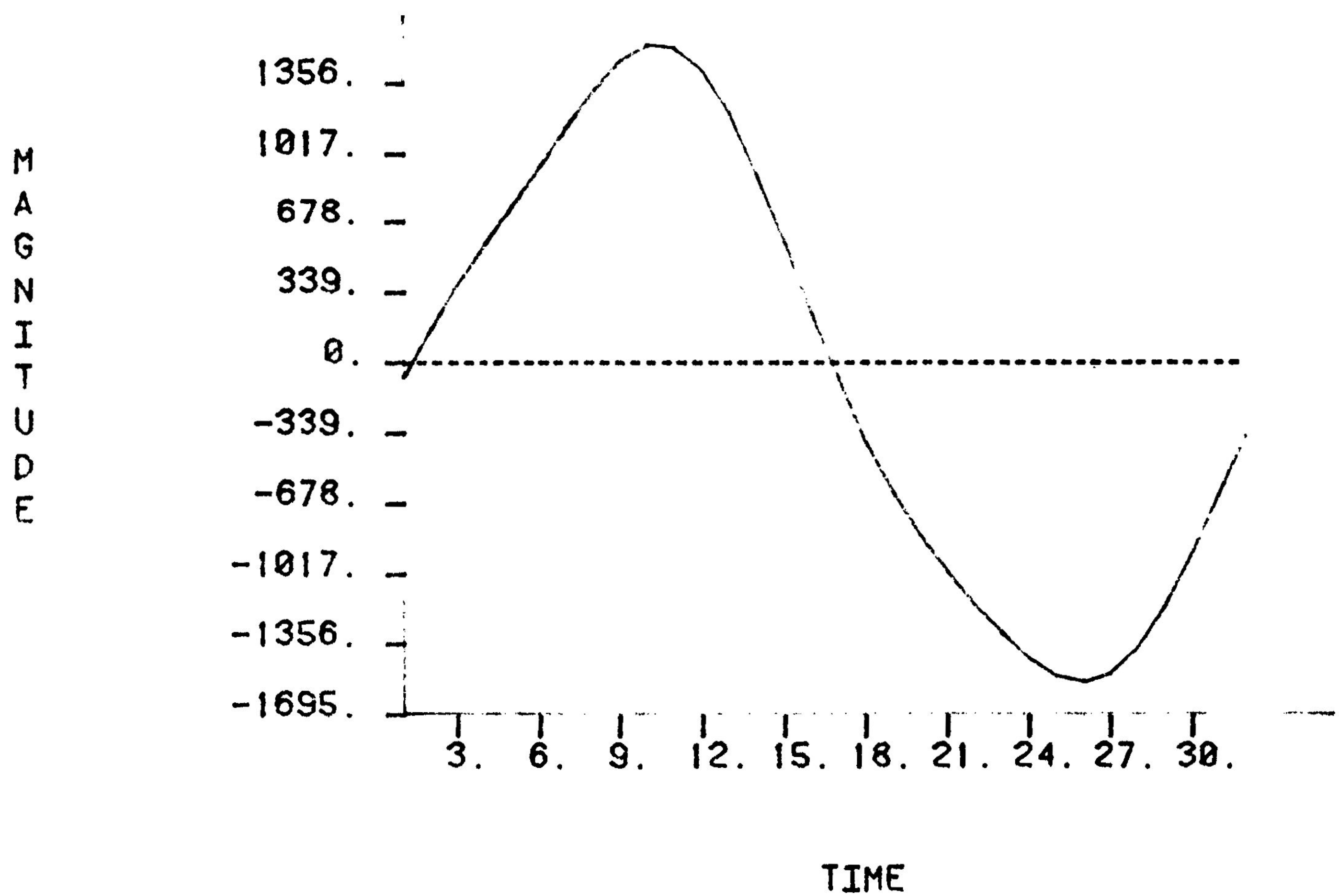


Fig. 12. The results of the inverse transform of the data obtained by SIN(X)+RAN(L,M)*SIN(X) after scaling employing the optimal zero filling range 5-29.

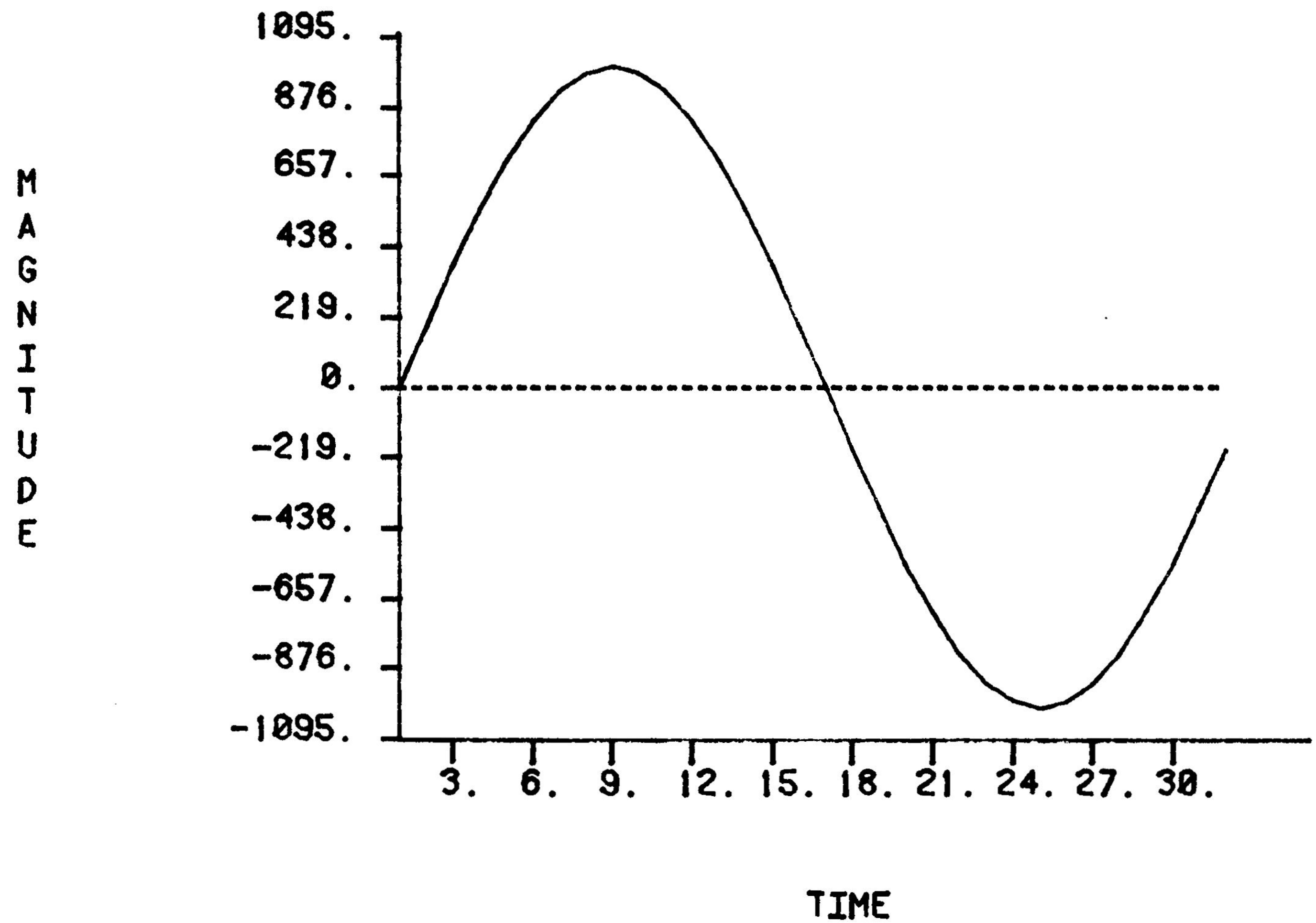


Fig. 13. The results of the inverse transform of the data obtained by SIN(DELX*1000.0*L).

To further illustrate the use of the RCFFT program several spectra were generated using the formula

$$\mathrm{AINT(ABS(X \cdot SIN(2 \cdot X)+NOISE \cdot RAN(L,M) \cdot ABS(X \cdot SIN(2 \cdot X)) \cdot 1000)} \qquad (9)$$

by altering the NOISE parameter. Spectra were generated with a zero NOISE factor (Fig. 14) and a 50% (or 0.5) NOISE factor (Fig. 15). A forward transformation was performed on both sets of data and various zero filling ranges were chosen so as to eliminate the noise. Fig. 16 depicts the result of the inverse transform of data (50% NOISE) with a zero filling range of 8-25. This was found to be the optimal range because this spectrum more closely resembled the spectrum in Fig. 14 with the exception of the fact that the spectrum was shifted above the zero mark.

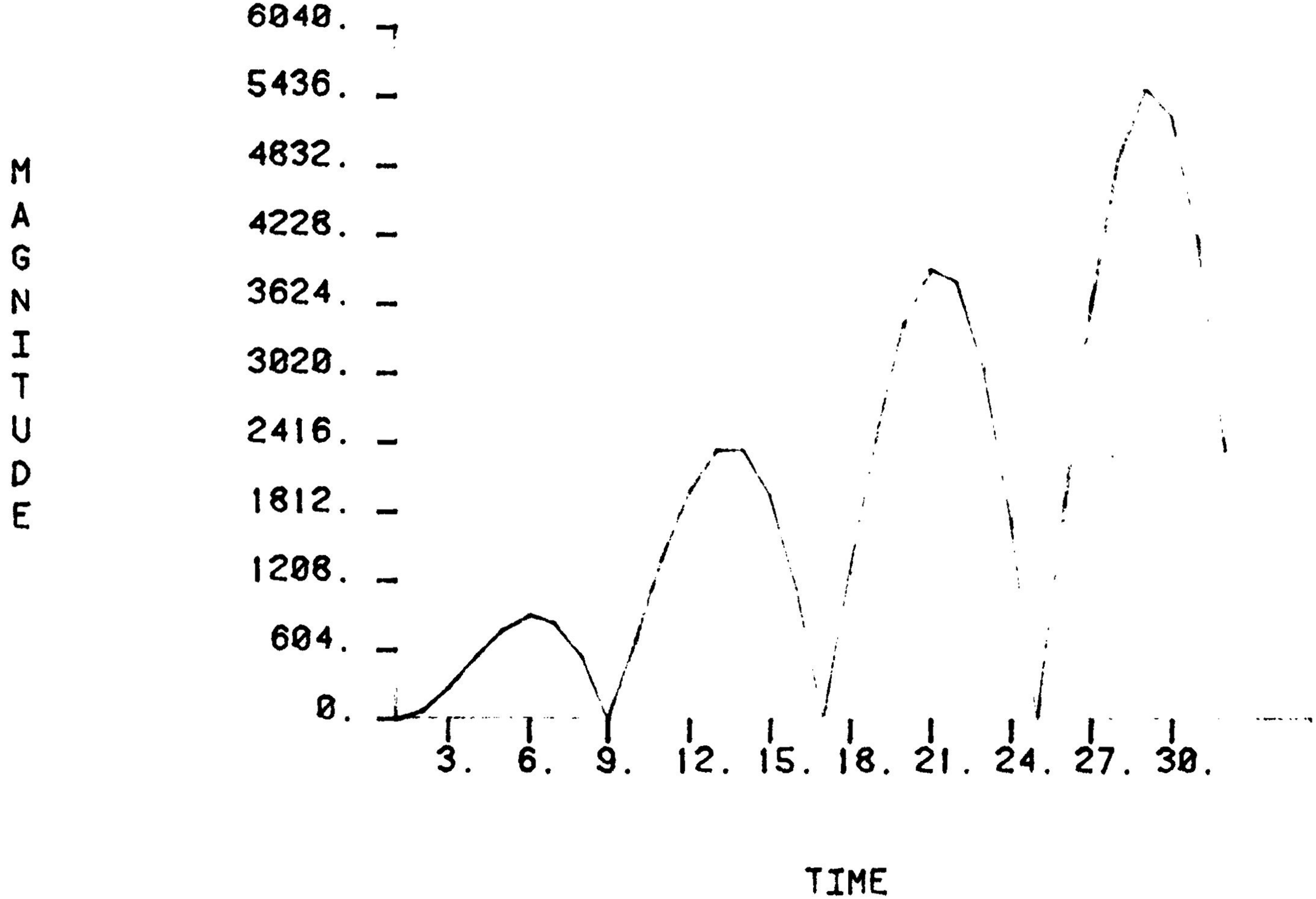
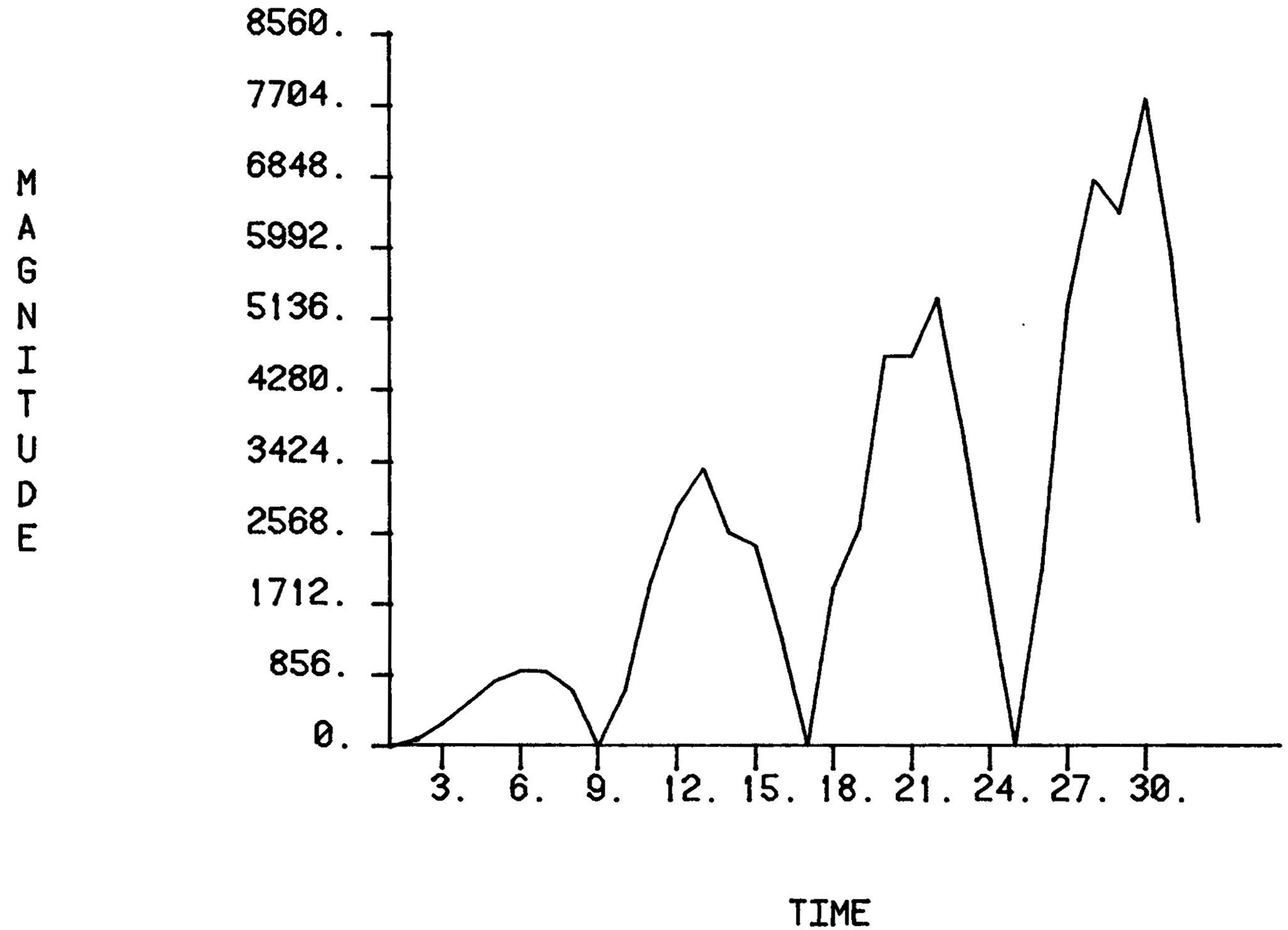


Fig. 14. Curve obtained by AINT(ABS(X*SIN(2*X))+NOISE*RAN(L,M)*ABS(X*SIN(2*X))*1000) with a zero NOISE factor.



<u>Fig. 15.</u> Curve obtained by AINT(ABS(X*SIN(2*X))+NOISE*RAN(L,M)*ABS(X*SIN(2*X))*1000) where NOISE equals 0.5, a 50% NOISE factor.

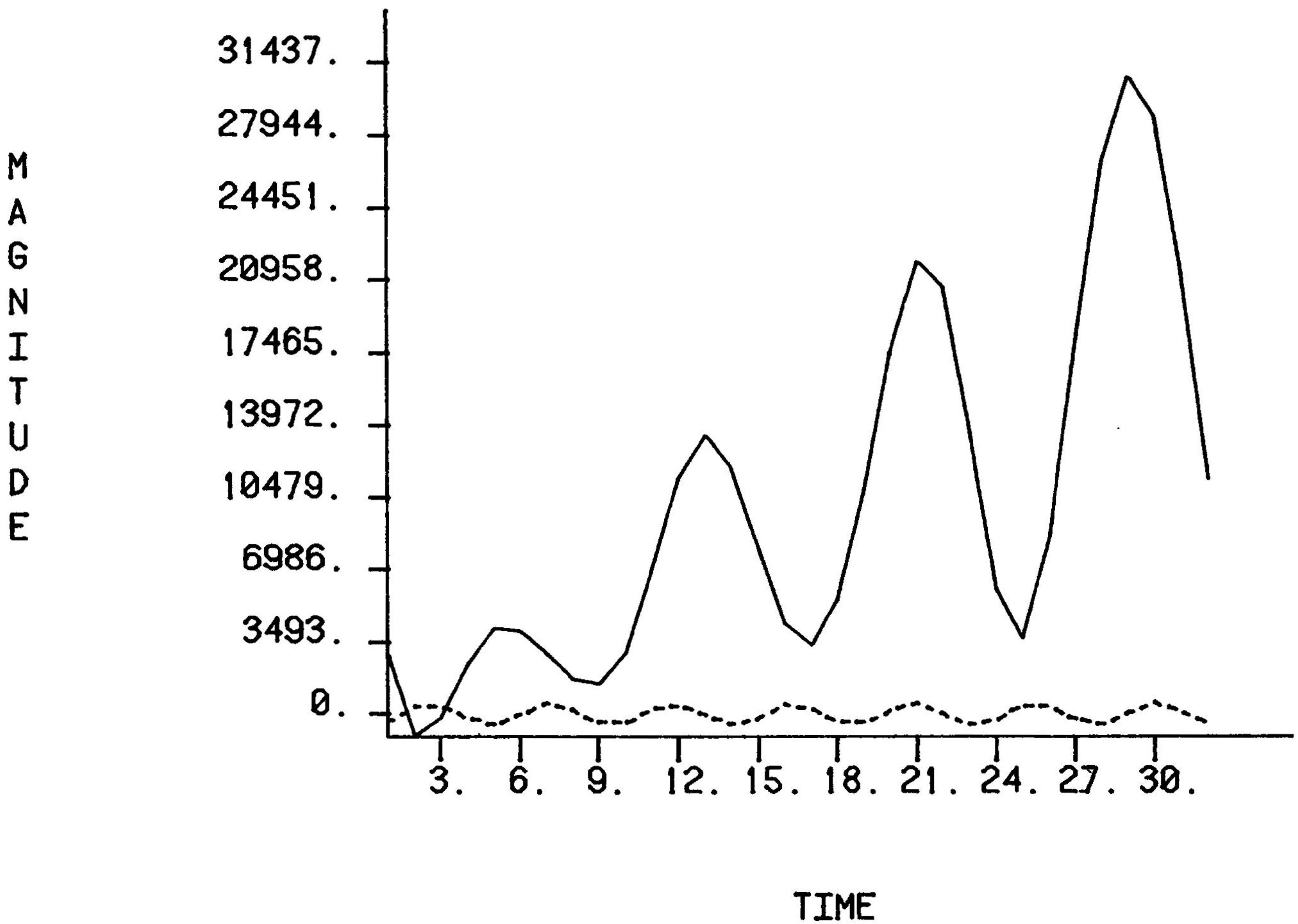


Fig. 16. The results of the inverse transform on the data obtained by AINT(ABS(X*SIN(2*X))+NOISE*RAN(L,M)*ABS(X*SIN(2*X))*1000) employing a zero filling range of 8-25.

CONCLUSION

A FORTRAN program was developed that performs digital filtering via the FFT method. In order to illustrate the effectiveness of the digital filtering technique a noisy spectrum was simulated and the random frequency components (noise) were removed. This method may be practically applied to the generation of more coherent spectra through signal enhancement of data obtained by analytical techniques in the chemical laboratory.

REFERENCES

1. D. P. Binkley and R. E. Dessy, J. Chem. Ed., 56, 148 (1979).

2. E. O. Brigham, "The Fast Fourier Transform," Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.

3. W. D. Stanley and S. J. Peterson, BYTE, 3, 14(1978).

4. H. J. Blinchikoff and A. I. Zverev, "Filtering in the Time and Frequency Domains," John Wiley and Sons, Inc., New York, NY, 1976.

5. J. W. Cooley and J. W. Tukey, Math. Comput., 19, 297(1965).

6. IEEE Group on Audio and Electroacoustics Subcommittee on Measurement Concepts, IEEE Trans. Audio Electroacoust., AU 15, 45(1967).

7. J. W. Cooper in "Transform Techniques in Chemistry," P. R. Griffiths, Ed., Plenum Press, New York, NY, 1978, Chapter 4.

8. G. D. Bergland, IEEE Spectrum, 6, 41(1969).

9. "Laboratory Subroutines Manual," Digital Equipment Co., Maynard, MA, 1978, Chapter 6.

10. J. B. Phillips, Anal. Chem., 52, 468(1980).

11. D. P. Binkley and R. E. Dessy, ibid., 52, 1335(1980).

12. P. R. Griffiths, Appl. Spectrosc., 29, 11(1975).

13. R. J. Halloran and D. E. Smith, Anal. Chem., 50, 1391 (1978).

14. S. D. Stearns, "Digital Signal Analysis," Hayden Book Co., Inc., Rochelle Park, NJ, 1975, p. 102.

15. P. L. Emerson, Creative Computing, 6, 58(1980).

16. W. D. Stanley, "Digital Signal Processing," Reston Publishing Co., Inc., Reston, VA, 1975, p. 285.

17. "4024/4025 Computer Display Terminal Programmer's Reference Manual," Tektronix, Inc., Beaverton, OR, 1978,p. 91.

18. "4662 Interactive Digital Plotter Users Instruction Manual," Tektronic, Inc., Beaverton, OR, 1978, p. 1-1.

APPENDIX

THE PROGRAM RCFFT

```
FORTRAN IV      V02.1-1                                          PAGE 001

      C
      C THIS PROGRAM WILL READ IN A SEQUENCE OF FILES (MAX=4), PERFORM
      C  A FORWARD AND/OR INVERSE FOURIER TRANSFORM VIA THE FFT
      C   SUBROUTINE, AND GIVE A GRAPHICAL REPRESENTATION OF THE DATA.
      C
      C
      C THIS PROGRAM WAS WRITTEN BY RAMONA M. CALVEY
      C  FOR THE CHEMOMETRICS LABORATORY
      C   AT ATLANTA UNIVERSITY
      C
      C
0001        DIMENSION NDATUM(4),YPNTS(100,4),IM(100,4),IR(100,4),
           1   RYPNT(100,4),ITIME(4),IFREQ(9)
0002        COMMON XPNTS(100,4),YIRM(100,4)
0003        EQUIVALENCE (YPNTS,YIRM)
0004        DATA IY/'Y'/
0005        DATA IN/'N'/
0006        DATA IFOR/'FORWARD'/
0007        DATA INVR/'INVERSE'/
0008        DATA ITIME/84,73,77,69/
0009        DATA IFREQ/70,82,69,81,85,69,78,67,89/
      C
0010        TYPE 55
0011        TYPE 56
      C
0012        TYPE 100
      C
      C INPUT THE NUMBER OF FILES TO BE READ IN.
      C
0013        ACCEPT *,NFILES
0014        MDATUM=0
      C
      C INPUT THE FILE NAMES.
      C
0015        DO 10 N=1,NFILES
0016        TYPE *,' INPUT THE FILE NAME FOR FILE ',N
0017        TYPE 500
0018        CALL ASSIGN(N,'XXXXXX.DAT',-1,'OLD','NC',1)
      C
      C INPUT THE NUMBER OF POINTS PER FILE.
      C
0019        TYPE *,' INPUT THE NUMBER OF DATA POINTS IN FILE ',N
0020        ACCEPT *,NDATUM(N)
0021        IF(NDATUM(N) .GT. MDATUM) MDATUM=NDATUM(N)
      C
      C READ IN THE FILES.
      C
0023   9999 READ(N,*)(XPNTS(I,N),YPNTS(I,N),I=1,NDATUM(N))
0024    10  CONTINUE
      C
      C
0025        DO 20 J=1,NFILES
0026        DO 30 I=1,NDATUM(J)
0027        IM(I,J)=0
```

```
            IR(I,J)=IFIX(YPNTS(I,J))
   30       CONTINUE
   20       CONTINUE
            TYPE 200
            ACCEPT 300,IYORN
            IF(IYORN .EQ. IN) GOTO 5000
            CALL RCGRSP
            TYPE 37
            PAUSE 'ERASE THE GRAPHICS AREA-HIT THE RETURN KEY TO CONTINUE'
      C
      C GRAPH THE DATA READ IN.
      C
            CALL RCGRAF(NFILES,NDATUM,MDATUM,1,ITIME,4)
   5000     CONTINUE
            PAUSE 'HIT THE RETURN KEY TO CONTINUE EXECUTION'
            TYPE 400
      C
      C CALL THE FFT SUBROUTINE TO PERFORM A FORWARD TRANSFORM
      C  ON THE VALUES STORED IN IR AND IM.
      C
            DO 70 I=1,NFILES
            NF=NDATUM(I)
            CALL FFT(IE,NF,IR,IM,0,ISF)
   70       CONTINUE
      C
      C IF IE(RROR)IS NOT EQUAL TO ZERO, PRINT AN ERROR MESSAGE;
      C  IF ISCAL IS NOT EQUAL TO ZERO, PRINT THE SCALING FACTOR.
      C
            IF(IE)99,2,99
   2        IF(ISF .NE. 0) TYPE 999,ISF
      C
            TYPE 250
            ACCEPT 300,IYORN
            IF(IYORN .EQ. IN) GOTO 6000
            CALL WRITFL(IR,IM,NFILES,NDATUM,IFOR,2)
      C
   6000     CONTINUE
            DO 80 J=1,NFILES
            DO 90 K=1,NDATUM(J)
            YIRM(K,J)=FLOAT(IR(K,J))
            YIRM(K,J+1)= FLOAT(IM(K,J))
   90       CONTINUE
   80       CONTINUE
            CALL RCGRSP
            TYPE 37
            PAUSE 'ERASE THE GRAPHICS AREA-HIT THE RETURN KEY TO CONTINUE'
      C
      C GRAPH THE FORWARD TRANSFORM.
      C
            CALL RCGRAF(NFILES,NDATUM,MDATUM,2,IFREQ,9)
            PAUSE 'HIT THE RETURN KEY TO CONTINUE EXECUTION'
            TYPE 400
            TYPE 260
            ACCEPT 300,IYORN
```

```
0069          IF(IYORN .EQ. IN) GOTO 1111
      C
      C INPUT THE CUT OFF FREQUENCY AND FILL THE FREQUENCY VALUES
      C  ABOVE FMAX WITH ZEROES.
      C
0071          TYPE 265
0072          ACCEPT 300,IYORN
0073          IF(IYORN .EQ. IN) GOTO 6100
0075          DO 110 J=1,NFILES
0076          TYPE *,' INPUT THE CUT OFF FREQUENCY RANGE FOR FILE',J
0077          TYPE 267
0078          ACCEPT *,IFMIN,IFMAX
0079          DO 110 I=IFMIN,IFMAX
0080          IR(I,J)=0.0
0081          IM(I,J)=0.0
0082     110  CONTINUE
0083     6100 CONTINUE
      C
      C CALL THE FFT SUBROUTINE TO PERFORM AN INVERSE TRANSFORM ON
      C  VALUES STORED IN IR AND IM.
      C
0084          DO 40 K=1,NFILES
0085          NF=NDATUM(K)
0086          CALL FFT(IE,NF,IR,IM,1,ISI)
0087     40   CONTINUE
      C
      C IF IE(RROR) IS NOT EQUAL TO ZERO, PRINT AN ERROR MESSAGE;
      C  IF ISCAL IS NOT EQUAL TO ZERO, PRINT THE SCALING FACTOR.
      C
0088          IF(IE)99,3,99
0089     3    IF(ISI .NE. 0) TYPE 999,ISI
      C
0091          TYPE 250
0092          ACCEPT 300,IYORN
0093          IF(IYORN .EQ. IN) GOTO 6500
0095          CALL WRITFL(IR,IM,NFILES,NDATUM,INVR,3)
      C
0096     6500 CONTINUE
0097          DO 50 L=1,NFILES
0098          DO 60 M=1,NDATUM(L)
0099          YIRM(M,L)=FLOAT(IR(M,L))
0100          YIRM(M,L+1)=FLOAT(IM(M,L))
0101     60   CONTINUE
0102     50   CONTINUE
0103          CALL RCGRSP
0104          TYPE 37
0105          PAUSE 'ERASE THE GRAPHICS AREA-HIT THE RETURN KEY TO CONTINUE'
      C
      C GRAPH THE INVERSE TRANSFORM.
      C
0106          CALL RCGRAF(NFILES,NDATUM,MDATUM,2,ITIME,4)
0107          PAUSE 'HIT THE RETURN KEY TO CONTINUE EXECUTION'
0108          TYPE 400
0109          TYPE 269
```

```
0110          ACCEPT 300,IYORN
0111          IF(IYORN .EQ. IY) GOTO 9999
0113          TYPE 270
0114          ACCEPT 300,IYORN
0115          IF(IYORN .EQ. IN) GO TO 1111
      C
      C COMPUTE THE SCALING FACTOR.
      C
0117          SCAL=2.0**(ISF+ISI)
      C
      C MULTIPLY THE REAL AND IMAGINARY PARTS OF THE TRANSFORM
      C  OF DATA BY THE SCALING FACTOR.
      C
0118          DO 5 L=1,NFILES
0119          DO 6 M=1,NDATUM(L)
0120          IR(M,L)=SCAL*(IR(M,L)/MDATUM)
0121          YIRM(M,L)=FLOAT(IR(M,L))
0122          IM(M,L)=SCAL*(IM(M,L)/MDATUM)
0123          YIRM(M,L+1)=FLOAT(IM(M,L))
0124     6    CONTINUE
0125     5    CONTINUE
0126          CALL RCGRSP
0127          TYPE 37
0128          PAUSE 'ERASE THE GRAPHICS AREA-HIT THE RETURN KEY TO CONTINUE'
      C
      C GRAPH THE DATA AFTER SCALING.
      C
0129          CALL RCGRAF(NFILES,NDATUM,MDATUM,2,ITIME,4)
0130     9000 CALL FINITT(0,760)
0131          STOP
      C
      C PRINT ERROR MESSAGES (ERROR CODE RETURNED).
      C
0132     99   TYPE 998,IE
0133          CALL FINITT(0,760)
0134          STOP
      C************************* FORMAT STATEMENTS *************************
0135     37   FORMAT(/,' IF YOU WANT A GRAPHICS HARDCOPY, TURN ON',
             1     ' THE 4662 DIGITAL PLOTTER AT THIS POINT',/,' -----',
             1     ' YOU MUST USE THE TEKTRONIX 4025 TERMINAL IN ORDER TO',
             1     ' GET A HARDCOPY-----',///)
0136      55  FORMAT(' THIS PROGRAM WILL READ IN A SEQUENCE OF FILES',
             1     ' (MAX=4), PERFORM A FORWARD AND',/,4X,'INVERSE FOURIER',
             1     ' TRANSFORM VIA THE FFT SUBROUTINE, AND GIVE A GRAPHICAL',/,
             1     26X,'REPRESENTATION OF THE DATA',//)
0137     56   FORMAT(' THE NUMBER OF POINTS PER FILE MUST BE AN INTEGER',
             1     ' VALUE THAT IS A POWER OF 2',/,27X,'BETWEEN 8 AND 1024',///)
0138     100  FORMAT(' INPUT THE NUMBER OF FILES TO BE READ; MAX=4')
0139     200  FORMAT(' WOULD YOU LIKE A GRAPHICAL REPRESENTATION OF',
             1     ' THE DATA INPUTED (Y OR N)?')
0140     250  FORMAT(' DO YOU WANT THE RESULTS OF THE TRANSFORM WRITTEN',
             1  ' OUT TO A FILE (Y OR N)?')
0141     260  FORMAT(' WOULD YOU LIKE TO PERFORM AN INVERSE TRANSFORM',
             1  ' (Y OR N)?')
```

```
FORTRAN IV      V02.1-1                                              PAGE 005

0142    265  FORMAT(' WOULD YOU LIKE TO ZERO ANY OF THE ELEMENTS IN THE',
           1     ' ARRAY(S)(Y OR N)?')
0143    267  FORMAT(' TO DO SO INPUT THE X VALUES CORRESPONDING TO THE',
           1     ' INITIAL FREQUENCY VALUE',/,' AND THE FINAL FREQUENCY VALUE',
           1  ' (XMIN-OF-RANGE  XMAX-OF-RANGE).')
0144    269  FORMAT(' WOULD YOU LIKE TO START OVER SO THAT A NEW CUT OFF',
           1     ' FREQUENCY RANGE MAY BE',/,' SPECIFIED (Y OR N)?')
0145    270  FORMAT(' WOULD YOU LIKE A GRAPH OF THE DATA AFTER SCALING',
           1     ' (Y OR N)?')
0146    300  FORMAT(A4)
0147    400  FORMAT(1X,'!WOR 0')
0148    500  FORMAT(/)
0149    998  FORMAT(//,' THE ERROR CODE RETURNED = ',I4)
0150    999  FORMAT(//,' THE SCALING FACTOR RETURNED = ',I4)
      C***********************************************************************
0151    1111 CALL FINITT(0,760)
0152         END
```

```
FORTRAN IV      V02.1-1                                          PAGE 001

      C
      C THIS SUBROUTINE CREATES A GRAPHIC WORKSPACE ON THE 4025.
      C
      C WRITTEN BY RAMONA M. CALVEY
      C
            SUBROUTINE RCGRSP
            TYPE 20
            TYPE 25
            TYPE 30
            TYPE 40
      20    FORMAT (1X,'!WOR 31 H')
      25    FORMAT (1X,'!MON K')
      30    FORMAT (1X,'!GRA 1,33')
      40    FORMAT (1X,'!SHR')
            RETURN
            END
```

```
FORTRAN IV      V02.1-1                                         PAGE 001

      C
      C THIS SUBROUTINE PERFORMS THE GRAPHICS FOR THE RCFFT MAIN ROUTINE.
      C
      C WRITTEN BY RAMONA M. CALVEY
      C
            SUBROUTINE RCGRAF(NFILES,NDATUM,MDATUM,IDUMMY,ITORF,JI)
            DOUBLE PRECISION XNUM,YNUM
            DIMENSION ITRANS(15),NDATUM(NFILES),IZERO(2),IDATUM(10),
           1     IMAGNI(9)
            COMMON XPNTS(100,4),YPNTS(100,4)
            DATA IZERO/48,46/
            DATA IMAGNI/77,65,71,78,73,84,85,68,69/
      C
            J=1
            NTFILE=2*NFILES
            DO 5 I=1,NTFILE
            DO 5 IX=1,2
            IDATUM(J)=NDATUM(I)
            J=J+1
        5   CONTINUE
      C
            CALL MINMAX(XPNTS,MDATUM,NDATUM,NFILES,XMIN,XMAX)
            XMAX=XMAX+.1*XMAX
            XRANGE=(ABS(XMAX-XMIN))/10.0
            XRANGE=AINT(XRANGE)
            IF(IDUMMY .EQ. 2) GOTO 1000
            CALL MINMAX(YPNTS,MDATUM,NDATUM,NFILES,YMIN,YMAX)
            GOTO 2000
       1000 CONTINUE
            YMIN=YPNTS(1,1)
            YMAX=YPNTS(1,1)
            DO 50 N=1,NTFILE
            DO 60 I=1,IDATUM(N)
            IF(YPNTS(I,N) .LT. YMIN)YMIN=YPNTS(I,N)
            IF(YPNTS(I,N) .GT. YMAX)YMAX=YPNTS(I,N)
       60   CONTINUE
       50   CONTINUE
       2000 CONTINUE
            IF(YMIN.GT.0.0)YMIN=YMIN-.1*YMIN
            IF(YMIN.LT.0.0)YMIN=YMIN+.1*YMIN
            IF(YMIN.EQ.0.0)YMIN=-1.0
            YMAX=YMAX+.1*YMAX
            YRANGE=(ABS(YMAX-YMIN))/10.0
            YRANGE=AINT(YRANGE)
      C
            CALL INITT(120)
            CALL TWINDO(300,900,200,600)
            CALL MOVABS(300,600)
            CALL DRWREL(0,-400)
            CALL DRWREL(600,0)
            CALL DWINDO(XMIN,XMAX,YMIN,YMAX)
      C
            X=0.0
            DO 10 K=1,10
```

```
FORTRAN IV      V02.1-1                                          PAGE 002

0051          X=X+XRANGE
0052          CALL MOVEA(X,YMIN)
0053          CALL DTIC
0054          XNUM=DBLE(X)
0055          CALL ASCODE(XNUM,ITRANS,J)
0056          J=J-3
0057          JMOVE=-3*J
0058          CALL MOVREL(JMOVE,-30)
0059          CALL ANSTR(J,ITRANS)
0060     10   CONTINUE
        C
0061          IMOVE=570
0062          IF(JI .EQ. 9)IMOVE=540
0064          CALL MOVABS(IMOVE,100)
0065          CALL ANSTR(JI,ITORF)
        C
0066          Y=0.0
0067          YN=0.0
0068          DO 20 J=1,10
0069          Y=Y+YRANGE
0070          IF (Y .GT. YMAX) GOTO 15
0072          CALL MOVEA(XMIN,Y)
0073          CALL LTIC
0074          YNUM=DBLE(Y)
0075          CALL ASCODE(YNUM,ITRANS,K)
0076          K=K-3
0077          KMOVE=-15*K-20
0078          CALL MOVREL(KMOVE,0)
0079          CALL ANSTR(K,ITRANS)
0080    15    YN=YN-YRANGE
0081          IF (YN .LT. YMIN) GOTO 20
0083          CALL MOVEA(XMIN,YN)
0084          CALL LTIC
0085          YNUM=DBLE(YN)
0086          CALL ASCODE(YNUM,ITRANS,K)
0087          K=K-3
0088          KMOVE=-15*K-20
0089          CALL MOVREL(KMOVE,0)
0090          CALL ANSTR(K,ITRANS)
0091     20   CONTINUE
        C
0092          I=1
0093          DO 90 K=510,310,-25
0094          CALL MOVABS(50,K)
0095          CALL ANSTR(1,IMAGNI(I))
0096          I=I+1
0097     90   CONTINUE
        C
0098          IF(YMIN.GE.0.0) GOTO 25
0100          CALL MOVEA(XMIN,0.0)
0101          CALL LTIC
0102          CALL MOVREL(-50,0)
0103          CALL ANSTR(2,IZERO)
0104     25   CONTINUE
```

```
      C
            IF(IDUMMY .EQ. 1)GOTO 3000
            N=1
            IN=1
            DO 30 L=1,NTFILE
            IF(IN .EQ. 3) N=N+1
            IF(IN .EQ. 3) IN=1
            CALL MOVEA(XPNTS(1,N),YPNTS(1,L))
            DO 40 M=1,IDATUM(L)
            CALL DASHA(XPNTS(M,N),YPNTS(M,L),L-1)
        40  CONTINUE
            IN=IN+1
        30  CONTINUE
            CALL MOVABS(0,50)
            CALL ANMODE
            RETURN
      C
        3000 CONTINUE
            DO 70 L=1,NFILES
            CALL MOVEA(XPNTS(1,1),YPNTS(1,L))
            DO 80 M=1,NDATUM(L)
            CALL DASHA(XPNTS(M,1),YPNTS(M,L),L-1)
        80  CONTINUE
        70  CONTINUE
            CALL MOVABS(0,50)
            CALL ANMODE
            RETURN
      C
            END
```

```
FORTRAN IV      V02.1-1                                        PAGE 001

      C                 SUBROUTINE MINMAX
      C
      C
      C THIS SUBROUTINE DETERMINES THE MINIMUM AND MAXIMUM VALUES----
      C -----WRITTEN BY T. J. GREEN.
      C
      C
              SUBROUTINE MINMAX (PTS,NUMBER,NUM,NFILE,TMIN,TMAX)
              DIMENSION PTS(NUMBER,NFILE), NUM(NFILE)
      C PLACE THE FIRST VALUE IN THE ARRAY IN TMIN AND TMAX
              TMIN=PTS(1,NFILE)
              TMAX=PTS(1,NFILE)
              DO 20 N=1, NFILE
              NUM1=NUM(N)
              DO 20 M=1, NUM1
      C FIND THE LOWEST VALUE IN THE ARRAY AND PLACE IT IN TMIN
              IF (PTS(M,N) .LT. TMIN) TMIN=PTS(M,N)
      C FIND THE LARGEST VALUE IN THE ARRAY AND PLACE IT IN TMAX
              IF (PTS(M,N) .GT. TMAX) TMAX=PTS(M,N)
      20      CONTINUE
              RETURN
              END
```

```
FORTRAN IV      V02.1-1                                          PAGE 001

C                   SUBROUTINE ASCODE
C
C
C THIS SUBROUTINE RETURNS THE ASCII CODE OF A NUMBER
C ----------WRITTEN BY T. J. GREEN (1/18/80)
C
C
      SUBROUTINE ASCODE (YNUM,ITRANS,J)
      DOUBLE PRECISION YNUM1, TENTH, ONE, TEN, ZERO
      DOUBLE PRECISION ADD, TMULTY, TRANGE, BRANGE, XN, YN, ZN
      DIMENSION ICHAR(12), ITRANS(15)
      DATA ICHAR/49,50,51,52,53,54,55,56,57,48,46,45/
      ZERO=0.0
      TENTH=0.10000
      ONE=1.00000
      TEN=10.00000
      ADD=0.00090
C THE MAXIMUM NUMBER IS 999,999,999 AND THE MINIMUM NO. IS 0.001
      TMULTY=1.0D-10
      TRANGE=1.0D+10
      BRANGE=1.0D-04
      N1=0
      N2=10
      NDEC=2
C DETERMINE WHETHER THE NUMBER IS OUT OF RANGE
      IF (DABS(YNUM) .GE. TRANGE) GOTO 5
      IF (DABS(YNUM) .LE. BRANGE .AND. YNUM .NE. ZERO) GOTO 5
      GOTO 15
5     TYPE 10
      STOP
C DETERMINE WHETHER THE ABSOLUTE VALUE OF A NUMBER IS
C LESS THAN ONE
15    IF (DABS(YNUM) .LT. ONE .OR. YNUM .EQ. ZERO) GOTO 40
C DETERMINE THE ASCII CODE OF A NUMBER GREATER OR EQUAL TO ONE
      L=2
      J=14
20    YNUM1=(DABS(YNUM))*TMULTY
      XN=(DMOD(YNUM1,ONE))*TEN
      N=IDINT(XN)
      IF (N .EQ. N1) GOTO 25
      GOTO 30
25    TMULTY=TMULTY*TEN
      J=J-1
      GOTO 20
30    ITRANS(L)=ICHAR(N)
      L=L+1
      ICOUNT=J-1
      IJ=J-NDEC
      DO 35 I=1, ICOUNT
      IF (L .EQ. IJ) L=IJ+1
      YN=(DMOD(XN,ONE))*TEN
      YN=YN+ADD
      ZN=(DMOD(YN,ONE))*TEN
      N3=IDINT(ZN)
```

```
      XN=YN-ADD
      IF (N3 .EQ. N1) YN=YN+(YN*TENTH)
      N=IDINT(YN)
      IF (N .EQ. N1 .OR. N .GE. N2) N=N2
      ITRANS(L)=ICHAR(N)
      L=L+1
35    CONTINUE
      ITRANS(IJ)=ICHAR(11)
      GOTO 50
C DETERMINE THE ASCII CODE OF A NUMBER LESS THAN ONE
40    L=2
      J=5
      ITRANS(L)=ICHAR(10)
      L=L+1
      ITRANS(L)=ICHAR(11)
      L=L+1
      ICOUNT=J-2
      XN=YNUM
      DO 45 I=1, ICOUNT
      YN=(DMOD(XN,ONE))*TEN
      YN=YN+ADD
      ZN=(DMOD(XN,ONE))*TEN
      N3=IDINT(ZN)
      XN=YN-ADD
      IF (N3 .EQ. N1) YN=YN+(YN*TENTH)
      N=IDINT(YN)
      IF (N3 .EQ. N1 .OR. N .GE. N2) N=N2
      ITRANS(L)=ICHAR(N)
      L=L+1
45    CONTINUE
50    CONTINUE
      IF (YNUM .GE. ZERO) GOTO 55
      ITRANS(1)=ICHAR(12)
      J=J+1
      GOTO 65
55    CONTINUE
      DO 60 I=1,J
      ITRANS(I)=ITRANS(I+1)
60    CONTINUE
65    CONTINUE
C
C************************* FORMAT STATEMENT *************************
10    FORMAT (1X,'OUT OF RANGE----------OUT OF RANGE')
C*******************************************************************
C
      RETURN
      END
```

```
FORTRAN IV      V02.1-1                                            PAGE 001

      C
      C THIS SUBROUTINE WILL WRITE OUT A FILE CONTAINING EITHER
      C THE RESULTS OF AN INVERSE OR FORWARD TRANSFORM OF DATA.
      C
      C WRITTEN BY RAMONA M. CALVEY
      C
            SUBROUTINE WRITFL(IR,IM,NFILES,NDATUM,MTRANS,M)
            DIMENSION IR(100,4),IM(100,4),NDATUM(4)
            IF(MTRANS .EQ. 'INVERSE') GOTO 300
            TYPE 100
            GO TO 500
       300  TYPE 200
       500  TYPE 50
            CALL ASSIGN(M,'XXXXXX.DAT',-1,'NEW','NC',1)
            IF(MTRANS .EQ. 'INVERSE') GOTO 1000
            WRITE(M,*)' RESULTS FROM THE FORWARD TRANSFORM OF DATA'
            GO TO 2000
       1000 WRITE(M,*)' RESULTS FROM THE INVERSE TRANSFORM OF DATA'
       2000 WRITE(M,*)' -REAL PART-'
            DO 10 I=1,NFILES
            DO 20 J=1,NDATUM(I)
            WRITE(M,*)IR(J,I)
       20   CONTINUE
       10   CONTINUE
            WRITE(M,*)' -IMAGINARY PART-'
            DO 30 K=1,NFILES
            DO 40 L=1,NDATUM(K)
            WRITE(M,*)IM(L,K)
       40   CONTINUE
       30   CONTINUE
       50   FORMAT(/)
       100  FORMAT(' INPUT THE FILE NAME FOR THE FORWARD TRANSFORM')
       200  FORMAT(' INPUT THE FILE NAME FOR THE INVERSE TRANSFORM')
      C
            CALL CLOSE(M)
            END
```