CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r. 82/08)

Canada

ON-LINE WAVE DATA ANALYSIS ON HP1000 COMPUTER

by

Kishor C. Kelkar

A thesis presented to the
School of Graduate Studies and Research
of the University of Ottawa
in partial fulfillment of the requirements for the degree of
M.Sc. (Systems Science)

Ottawa, Ontario 1982

CONTENTS

PREFACE

This thesis work attempts to put together theories, techniques and procedures which can be used for ON-LINE (REAL-TIME) analysis of digital data signals, acquired by data acquisition system which may form an integral part of the HP1000 computer dedicated to experiment control, data acquisition, data storage and analysis.

In the first chapter concepts of ON-LINE and OFF-LINE analysis and their desirable features and needs are explained. Then a number of examples of ON-LINE data analysis in physical model studies in the field of hydrodynamics conducted at the Hydraulics Laboratory of the National Research Council, are described. Chapter 1 concludes with describing various real-time tasks required to be performed in ON-LINE data analysis.

Chapter 2 describes features of the Hewlett-Packard RTE IVB Real Time Executive operating system on an HP1000 computer which is used for experiment control, data acquisition, data storage and analysis at Hydraulics Laboratory N.R.C. Those hardware and software features which are of relevance from the point of view of ON-LINE data analysis, are discussed in order to indicate the potential with which the task of ON-LINE analysis can be handled.

Chapter 3 describes various system services which are available to the user's program through Executive calls. De-

tailed description of various calls which are particularly useful to write real-time programs is given. It is felt necessary to include these two chapters (2 & 3) , since all the programs written to implement ON-LINE data analysis tasks make use of system features which are characteristics of the HP1000 RTE IVB operating system. It must be borne in mind that these features simplify considerably REAL-TIME program development efforts.

Chapter 4 is devoted to ON-LINE digital filtering techniques, their implementation and related problems. A digital filter is implemented by means of software and this is because most of the signals do not contain frequency components of significance in excess of 5 HZ. Both recursive and non-recursive filter design programs are developed, which enable the user to evaluate filter coefficients for given pass-band/stop-band specifications. Programs which implement these filters are also given. Even though it is well known that recursive filter implementations undoubtedly execute faster than nonrecursive filters with comparable specifications , nonrecursive filter design programs and their implementations are included because their program implementation can take advantage of the microcoded Vector Instruction Set available on the F-series machine, which drastically reduces their execution time. Another objectionable characteristics of nonrecursive digital filters is that they require large memory storage to retain previous data samples, but with de-

creasing cost of memory and availability of hardware features like Dynamic Mapping System (DMS) this problem should no longer be of overriding concern.

Chapter 5 describes in detail push down stack and ring buffer data structures which are the basic tools in the implementation of digital filters by software and the ON-LINE computation of auto-correlation functions and the nonrecursive methods of finding running RMS and Average values. Comparison in terms of their performance on the basis of execution time, type of machine and use of microcoded instruction like the VIS library is given.

Chapter 6 summarizes theoretical considerations for ON-LINE analysis and gives recursive and nonrecursive methods for running average and RMS analysis of the data. It also gives a recursive method for finding the auto-correlation and the running estimation of the variance spectral density.

Chapters 7, 8, 9 describe program Implementations for ON-LINE RMS analysis, ON-LINE average analysis and ON-LINE variance spectral density estimation respectively. They describe related problems and programming considerations using RTE IVB system features. This is an effort to achieve one of the goals of this thesis namely implementation of ON-LINE analysis tasks for user's interaction with experimental conditions.

Chapter 11 makes concluding remarks about the work done to meet these goals of the thesis and makes recommendations regarding methods best suited for ON-LINE data analysis based on experimental observations.

Ackowledgements

I wish to thank Mr. E. R. Funke, Senior Research Officer of the Hydraulics Laboratory, Division of Mechanical Engineering N.R.C., without whose valuable guidance , this thesis work would not have been possible. In fact the subject of the thesis was suggested by him, and a series of discussions with him had been of great motivation to orient my thinking and work to achieve the goals of this thesis. I wish to thank Dr. G. M. White of University of Ottawa, for his kind cooperation and time to time reviewing of the work which I used to discuss with him during its progress. I also give my sincere thanks to Mr. N. L. Crookshank (Systems Manager Hydraulics Laboratory N.R.C.) and Mr. D. Pelletier for their extremely valuable suggestions in developing programs under HP RTE IVB operating system on HP1000 computer, for ON-LINE data analysis . I also wish to thank the National Research Council of Canada and Mr. J. Ploeg the head of the Hydraulics Laboratory for supporting the practical work required for this thesis. Last but not least, I wish to thank Dr. E.P.D. Mansard (Associate Research Officer) who helped me to evaluate the performance of various programs

during practical experiments and who made timely and valuable suggestions during preparation of the final report.

Kishor C. Kelkar

CHAPTER 1

REAL TIME DATA ANALYSIS BY DIGITAL COMPUTER

a) Introduction

The use of digital computers in support of scientific or engineering research is now a well established practice. However, in many situations these computers are still far removed from the laboratory. Experimental data may be collected by various recording systems and these recordings are then shipped to a computation centre for analysis. This mode of operation describes what may be called OFF-LINE data analysis.

Another approach, which is becoming increasingly more practical with increasing power and decreasing cost of mini or microcomputers, is their direct connection to the instrumentation of an experiment via suitable analog to digital converters. In most cases the computer serves in the first instance, the purpose of data acquisition and data storage. This function may be consideredON-LINE with the experiment. The subsequent data analysis is then carried out after the experiment is completed and, in this sense the data analysis must be considered OFF-LINE.

Modern computer systems with a Real-time operating system have the capability of executing function in the Foreground as well as in Background. A Foreground function is always

slaved to a Real-time clock and the computer is forced to respond with high priority to such Real-time interrupts. A typical Foreground activity may be a data acquisition function and the storage of such data into a Real-time buffer. Any unused processor time is then utilised by the Background phase and this may include analysis of data contained in the Real-time buffer. For all intents and purposes this mode may be considered a Real-time data analysis. However, if one wants to be a purist one may restrict the use of the term Real-time data analysis only to those cases where the analysis is also done in the Foreground.

In either case data are being analysed as they are acquired and it is this particular requirement which creates special demands and constraints on the implementation.

It may well be asked : Why go to the trouble of Real-time analysis if data could be analysed more conveniently Off-line. In order to answer this it may be best to give a number of examples. However, first it should be emphasised that Real-time data processing is not new. Most sophisticated instrumentation incorporates some degree of data processing usually as a part of the electronic subsystem. Such processing is mostly done by analog techniques, that is by means of resistors, capacitors, inductors, diodes and so on. The type of processing which takes place, includes averaging, filtering, peak detection, RMS calculation and various non-linear conversions.

Analog electronic techniques offer almost unlimited processing speeds but their power, particularly for nonlinear functions is rather limited. Digital data processors, on the other hand have in principle, almost unlimited computational power however, their processing speeds are still too slow for most applications. However, as long as the frequency content of the data is in the sub-audio range(that is less than 100 HZ) then modern on-line digital processors can prove to be most useful.

One area of scientific and engineering research which deals extensively with such sub-audio range measurement signals is directed to studies of the coastal zone and deals with problems in hydrodynamics. Most of this research is carried out by means of scaled models for the purpose of simulating a situation which occurs in Nature. Measurements are made for varying water surface elavations, currents, pressures, forces, displacements, velocities or accelerations of floating objects and so on. Most of these signals do not contain significant frequency components in excess of 5 HZ. Some shock pressure signals may exceed 100 HZ, but those are special cases. Because of the suitability of the frequency range of these signals the following examples are drawn from experience in a coastal engineering laboratory.

Applications

Example 1

b) Pretension Adjustment of mooring forces

A model ship is tied to its mooring by six lines which pull the ship against elastic fenders. All lines must be pretensioned to specified levels.

Pretension adjustment on one line affects the setting on all other remaining five lines. In order to bring the total system to the required setting it is necessary to monitor the forces on all lines at the same time and then, by trial and error, adjust one at a time until all lines have the desired setting.

Force transducers generate voltages which must be converted to force by means of the formula :

$$Fi = (Vi - Vo)*SC$$

where

Vi is the voltage measured at the instant i

Vo is an offset voltage and

SC is a scale factor in force units/volt

The converted force signal is typically quite noisy due to the highly resonant nature of the model set up. In order to get an accurate steady force reading, smoothing operations must be undertaken. Conversion and smoothing on six simultaneous transducers is therefore the Real-time task.

Example 2

c) Calibration of scaled model of a tidal Estuary

Scaled models of tidal estuaries give the appearance of being miniature versions of an estuary which is to be studied. Because of scaling laws, a tidal cycle of 12 hours may be re-enacted in periods from 1 to 5 minutes depending on the scale factor used. When calibrating a model of this type, artificial roughness must be introduced into the model by means of rocks, wire mesh or metal strips which must be placed and distributed by trial and error until the tidal highs and lows within the estuary correspond in magnitude to what is normally observed in Nature.

The ON-LINE computer must therefore monitor all tidal gauges, convert the results into physical units and then apply a running crest and trough detection algorithm which keeps track of the last two troughs and the last two crests. Meanwhile, the scientist can experiment with the placement of roughness and observe the consequence of his actions within one tidal period without having to halt the running of the model.

Example 3

d) Measurement of three dimensional flow

A technique for measuring the magnitude and direction of low velocity flow in three dimensional space has been proposed. A probe with a short heating element is inserted into the medium. A heating current is pulsed through the element with an intensity and frequency which makes it possible to generate measurable packets of heated fluid after the fluid passes the heating element.

A fast response temperature sensor is placed in proximity to the heater but at a distance far enough so that a measurable time elapses before a heated elemental volume of fluid travels the distance from the heater to the temperature sensor.

Data processing requires the calculation of the cross-covariance function for a range of time shifts. If I(t) is the heating current and R(t) is the temperature measured by the sensor then the cross-covariance is given by

$$CCVR(T) = (1/Tn)* \int_{0}^{Tn} R(t)*I(t+T)*dt$$

For some particular value of T this cross covariance will be a maximum.

If the cross-covariance is continuously evaluated in Real-time, then the experimenter may move the thermal sensor in a constant orbit about the heater until he finds the lo-

cation at which the maximum covariance occurs.This position of the sensor relative to the heater will then give direction of the flow and the probe spacing , and the time shift computed from the cross-covariance gives the velocity average over the interval Tn.

Example 4

e) Tuning the speed of a D.C. motor driven Wave-machine

Some irregular wave generators are driven by D.C. motors whose speed is modulated in some controlled fashion so that the wave train, as observed some distance down stream from the wave board, has the desired spectral shape and, in particular, has a spectral peak at a specified frequency. Since non-linear wave-wave interaction leads to very complicated shifts of energy in the frequency domain, it is very difficult to predict the outcome of a particular generation process.

A wave probe at the desired test site can monitor the generated wave train. The ON-LINE computer program must be programmed to compute in Real-time a variance spectral density which is able to reveal to the experimenter how the energy is distributed in the frequency domain. He may then adjust the speed of the motor, note the resultant effect on the spectrum and then use this information by trial and error until the desired spectral distribution has been matched as well as it is practical.

Example 5

f) Determination of the location of nodes and antinodes within a model harbour basin.

For the purpose of improving the behaviour of marine harbours, scaled model studies are being conducted wherein the model harbour is being agitated by waves propagating into the harbour from the outside. Wave probes are then placed inside the harbour basin which are then used to assess the behaviour of the harbour for any one of several experimental conditions which must be investigated.

The placement of probes inside the harbour is rather critical because the harbour walls generate reflections and the pattern of the wave activity within the basin is complex. There are not only the wave frequencies which were contributed by the external wave field but there are, in addition several harmonics of this external wave as well as resonances related to the physical lay-out and dimension of the harbour.

The optimum location for wave probes are those which yield the greatest wave activity as a function of the agitation frequency. The REAL-TIME task is then band-pass filtering of the wave probe output and subsequent computation of the standard deviation or variance with appropriate averaging. As a consequence, the experimenter can scan the basin

with the wave probe and seek out the location which may be considered nodes of the basin.

Example 6

g) Testing the strength of model ice.

Model ice is made from a chemically doped water under controlled freezing conditions which favour the growth of vertical crystal formations. The strength properties of this model ice are important if interaction between certain model structure and ice are to be simulated correctly.

It is quite difficult to test the physical properties of this model ice. One of the tests which is now being conducted is as follows:

A cantilever is generated by cutting two parallel slots of finite length into a floating sheet of ice. The one end of the ice beam which was so created is also cut at right angles to the slots. A spring loaded pusher suitably instrumented to measure force is then used to push the cantilever down with ever increasing force until this ice specimen fails. The largest force so generated and the largest deflection prior to failure are the two quantities which must be recorded. Since these quantities depend somewhat on the speed with which the force is applied it would be most desirable to have an instrument which gives instantaneous read-out of applied force and corresponding deflection but will

hold the largest force value and corresponding deflection which occurred just before failure. The experimenter can then modify his experimental technique to achieve the results which he considers to be most applicable.

The above six examples, although having different requirements all have one thing in common:

They demand a direct interaction between the experiment and the experimenter who assesses the consequence of his experimental actions in terms of some more or less sophisticated transformation of one or several measured variables. The time delay introduced by an Off-line analysis would destroy the trial and error basis of experimental technique.

There is no doubt that many of the requirements described above could be fulfilled through the development of special purpose analog instrumentation although the nonlinear requirements might be quite difficult to realize.

The advantage of using an ON-LINE digital computer for these Real-time tasks is then that :

-Complicated functions can be implemented relatively easily .

-Any one implementation can be shared readily between several simultaneous channels.

-Long time-constants are more easily realizable.

-Because of the time sharing nature of digital computers, the processing equipment is much simpler as each additional function is only an other logical use of the same piece of hardware. This must be compared to maintenance requirements of several parallel analog processors associated with their respective instruments. Quite clearly the use of shared digital processor is preferable providing , ofcourse that it has the speed to execute the given task.

-Storage and documentation of data and calibration constants are superior.

The computer can thus be seen to form an integral part of the instrumentation and replaces in a sense, what was formerly realized by special purpose electronics which was part of an instrument.

On the other side there is the overriding disadvantage of speed which is related to a limited frequency range of use-

fulness. However with advancing digital technology great improvements can still be expected and the limitations of today should not restrict the visions of the future.

Real-time Processing Tasks

The examples given in section 1.0 can be classified into the following categories.

h) Conversions from voltage to physical variables.

These are typically linear operations of the form:

$$Yi = (Vi - Vo)*SC .$$

However, various non-linear conversions including table look-up procedures or multi-variable conversions are also possible. In any case these operations are carried out on one sample at a time as these become available through sampling.

i) Smoothing or Filtering of converted variables

In its simplest form, smoothing may be a simple integration but in general one may consider this either as a solution of the convolution integral with judicious use of a suitable smoothing or filtering window or it may be considered a solution of a differential equation by means of a numerical approximation which leads to a recursive formula.

Auto or Cross-covariance should also be considered part of this class because the covariance function is hardly distinguishable from the convolution integral.

All operations in this category have in common that past values of the data participate in the calculation and therefore memory for previously sampled data is an essential part of this operation.

Differentiations and integrations should also be considered as a special case of filtering.

j) Non-linear or complex Transformations

In this category one may consider such operations as squaring, limiting, holding, crest or trough detection and transformations such as Fourier Transforms. It is not possible to identify unique features common to these operations. Some require memory of past data and others do not. However, whereas the first two categories appear to offer some choice in methodology which may indicate that some algorithms are more suitable executed in Real-time, this latter category does not offer any choice.

From the examples given in section 1.0 it may be seen that some require operations derived from more than one category. For example, the computation of the variance spectral density may require the calculation of the auto-covariance with associated integration or smoothing and susequent Fourier transformation.

k) The objectives of the research work described in this thesis are :

- To identify fundamental procedures for Real-time data analysis and select those most suited for the task.
- To implement a number of specific Real-time analysis tools and demonstrate their proper functioning.
- To implement enhanced control and display features for Real-time analysed data.

CHAPTER 2

FEATURES OF THE REAL TIME EXECUTIVE OPERATING SYSTEM HP - RTE IVB

RTE IVB is a Real-time executive system. It processes all decisions and scheduling tasks internally unless overriden by user intervention. User requests for system action can be made by a call from within a program or interactively via an operator command.

As the major control element within the operating environment , RTE IVB provides the user with various services and automatically handles the machine-related functions associated with each service.

a) Some of the major services are :

*Executive communication scheme that provides a communication link between user-written programs and system services.

*Input/Output scheme which allows a program to continue executing while its own input/output requests are being processed.

*Program execution control that features multiprogramming(allows several programs to be active concurrently).

*Partitioned memory technique that takes advantage of the hardware Dynamic Mapping System (DMS) to provide access to 1024 K words of physical memory.

*Operator Interface that provides the user with the ability to control system action via operator commands.

MULTIPROGRAMMING

b) RTE IVB is a multiprogramming system that allows several Real-time programs to be active concurrently. Each program executes during the unused central processor time of the others and in order of their priority. Priority is based on the importance of the task they are performing. Scheduling/Dispatching modules in RTE IVB decide when to execute programs that are simultaneously requesting system services and/or resources. The Scheduling module places the program into a scheduled list in order of their priority(the highest priority program at the head of the list) and the dispatching module initiates the execution of the highest priority program. When the executing program is completed , is terminated ,or is suspended , it is removed from the scheduled list and the dispatching module transfers control to the next program with the highest priority.

Programs within the RTE IVB operating system are categorized according to where they reside., what type of memory partition they execute in , and which common areas they have access to. ON-LINE analysis programs reside in Real-time partitions and they have access to unlabled Real-time common where the data acquisition

program stores the acquired data for all analog channels.

MEMORY MANAGEMENT

c) The RTE IVB operating system is written to take advantage of the hardware Dynamic Mapping System (DMS) available with HP1000 computers. The co-operation between the software operating system and the hardware mapping system allows access to 1024 K words of physical memory.

The basic addressing space of the HP1000 computer is 32,768 words(32K) as defined by the 15-bit address length used by the CPU. This is referred to as logical memory. The amount of memory actually installed in the computer system is referred to as physical memory. The DMS maps the 32K words of physical memory into logical memory by translating the 15-bit address through memory maps.

Memory maps :- The page pointers contained in the map registers that comprise the memory maps are loaded by software modules within the operating system. Each map is configured to represent the 32 pages of physical memory (not necessarily continuous) that contain the tables, data buffers, program code etc, necessary to perform specific processing tasks. Since there are four maps , four different 32 page sections of the phy-

sical memory can be described simultaneously , with only one map being enabled at a time to indicate the section of the memory currently in use. The maps are altered by the operating system to reflect dynamic changes in the operating environment i.e. when a program is scheduled to execute a map has to be configured to describe the physical memory pages it requires (known as programs's logical address space).

System map :- The system map describes the logical address space associated with RTE IVB operating system and its base page common, subsystem Global area, Table area I and II, driver partition system, driver area and system available memory. The system map is loaded during system initialisation and is changed only to map in different partitions. Since the RTE IVB operating system handles all interrupt processing, the system map is automatically enabled whenever an interrupt occurs.

User map :- The user map is associated with each disc resident program. It is a unique set of pages that describe the logical address space containing the program code.

d) Physical memory :- Because of DMS feature, a number of Real-time programscan simultaneously reside in large physical memory and the system will automatically map corresponding physical pages in map registers whenever it

is time to schedule that particular program . Since loading of map registers is much faster, switching between two programs is fast and there is no time delay involved due to swapping of programs between memory and disc as used to be the case in absence of large physical memory and DMS feature.

Another way to meet Real-time demands is use of special purpose mathematical function processors in addition to main central processor. This not only makes computational operations faster but relieves CPU of some load. With advent of Microprocessor technology it has been further possible to distribute the tasks under the control of Main processor , such as experiment control or data acquisition and it is possible to add sophistication to control function activity. This would enable CPU to coordinate control of number of simultaneous experiments.

ADDITION OF POWERFUL LIBRARY ROUTINES

e) The HP1000 signal processing library (SIGNAL/1000) is a comprehensive subroutine library that forms the foundation for digital signal processing applications. SIGNAL/1000 is based on the IEEE Digital Signal Processing Committee software package and thus represents a compilation of the most current and effi-

cient algorithms in this field. These routines have been optimized to exploit the full computational capability of HP1000 system.

The software is accelerated via a set of signal processing Instructions that have been microcoded for the HP1000 F-series processor. These include :

1) Complex array bit-reversal

2) Complex FFT butterfly

3) Real FFT phasor multiplication

4) Complex add, subtract, multiply and divide

5) Complex conjugate, complement, AIMAG and CMPLX operations

Use of these Instructions as well as the HP1000 Vector Instruction Set enables the signal processing Library to achieve extremely high processing rates. For example a 1024 point FFT can be computed in just 270 milliseconds.

SIGNAL/1000 contains a set of FFT routines that have been optimised for HP1000 F-series processor . These routines have been designed for utmost performance in time-critical applications . Accuracy is maintained by performing all computations in floating point represen-

tation utilizing 23-bit mantissa. This format will accomodate the large dynamic range used by modern data acquisition hardware to represent input waveforms.

VECTOR INSTRUCTION SET (VIS) for HP1000 F-series computers

f) These routines perform arithmetic operations on arrays, particularly for vector processing. The Vector Instruction Set provides the following operations:

1) The sum, difference, product or quotient of corresponding elements of two arrays
2) The sum, difference, product and quotient of a scalar and an array.
3) Dot product of two arrays
4) The absolute value of an array, element by element
5) The sum of elements of an array or the sum of their absolute values
6) The sum of an array and the product of a scalar and another array knwon as a pivot operation
7) The identification by index of the maximum or minimum value of an array by actual or absolute value
8) Copying of an array into another
9) Copying an array to and from EMA(Extended Memory Access)

Advantages

VIS allows easier array manipulation and faster execution time of vector operations. VIS can replace FORTRAN DO LOOPS performing repetitive operations such as calculating the average value of an array (i.e. the sum of the elements of the vector divided by the number of elements , scaling etc). Execution is functionally equivalent yet much faster due to micro-code and the elimination of loop overload with a transparent interface to EMA , VIS allows handling of large data arrays with just few simple FORTRAN statements.

The ability to perform vector arithmetic easily at high speed is desirable for many applications some of which include:

1) Statistical analysis

2) Digital filtering

3) On-line digital data analysis

CHAPTER 3

SYSTEM PROGRAMS UNDER HEWLETT-PACKARD RTE IVB

An executing program may request various system services via a call to the EXEC processor, specifying the request in the parameter string. Initiation of the call causes a memory protect violation interrupt (enables the System map) and transfers control to the EXEC module.

We will discuss very briefly only those EXEC calls which are of relevance to Real-Time programming and some other calls which indirectly help Real-time programming.

a) Program scheduling EXEC 9,10

This schedules a program for execution and optionally passes up to five parameters and/or a buffer to the program.

CALL EXEC(ICODE,INAME,{,IOP1{,IOP2{,IOP3{,IOP4,
{,IOP5{,IBUFF}}}}}}<LEN>)

ICODE -Request code 9 = immediate schedule with wait

10 = immediate schedule without wait

INAME -3 word array containing an ASCII name of the program to be scheduled

IOP1 thru IOP5 -Optional parameters,passed to the program specified in INAME

IBUFF -Optional buffer,passed to the program specified in INAME

ILEN -Data length in number of words

FATHER AND SON -When one program schedules another,the program that did the scheduling is known as the FATHER and the program that was scheduled is known as the SON.

ICODE=9 or 10 IMMEDIATE SCHEDULE(WITH OR WITHOUT WAIT) When a father schedules a son without wait(ICODE=10),the son is scheduled for execution according to its priority , the father continues at its priority without waiting for the son to complete.(This feature is important when the father has to execute at a higher repetition rate than the son so that it cannot wait for son to complete , for example the display updating function required to be done at a slow rate can be handed over to the son program which is scheduled by the father which is performing ON-LINE data analysis at a data sampling rate.) But in some cases it may be necessary that the son should complete execution before the father can take over the control. In that case the father sche-

dules the son with wait(ICODE=9). The father is then placed in the general wait list(state 3). When the son completes or is terminated , the father resumes execution at the point immediately following the scheduling EXEC call.

In order for the father program to immediately schedule a son (ICODE=9 or 10), the son must be dormant(state 0). The father can determine whether the son was scheduled or not by examining the contents of the A-register after execution of the scheduling call.

Optional parameter passage

The optional parameters(IOP1 - IOP5) can be used to pass values from father to the son . The father places the values to be passed into the optional parameters and then makes the scheduling call. When the son begins executing it can obtain these values by calling the library subroutine RMPAR. It should be noted that the RMPAR call should be the first executable statement into the son program.

Optional buffer passage

In addition to five optional parameters the father can pass an optional buffer to the son. The father places the data to be passed into IBUFF and specifies

the length of data in ILEN. When the father makes the scheduling call the buffer is moved into system available memory(SAM). The son can obtain the buffer by making an EXEC call 14(STRING PASSAGE-EXEC 14). If there is not enough SAM currently available to hold the buffer, the father is memory suspended(state 4). If there will never be enough SAM available, the father is aborted or if the 'no abort' bit is set , the error return is taken. b) String passage EXEC 14

This allows a program to retrieve a buffer from the program that scheduled it (i.e. the father) or retrieve the command string if it was scheduled by an operator command. It allows the son to pass a buffer back to its father.

CALL EXEC(ICODE,IRWCD,IBUFF,ILEN)

ICODE -Request code 14

IRWCD -Retrieve/Write code 1=retrieve buffer or
command string
2=write buffer to father

IBUFF -Buffer into which retrieved data or command string is placed(IRWCD=1) or a buffer into which son places data to be returned to the father IRWCD=2

ILEN -Positive number of words or negative number of characters(bytes) to be transferred.

- FATHER/SON BUFFER PASSAGE

If the father includes the optional buffer as a parameter in the EXEC call used to schedule a son the contents of the buffer are placed in SAM when the call is executed. If the son performs an EXEC-14(IRWCD=1), the buffer in SAM is retrieved. The contents are placed in IBUFF and the SAM is deallocated. The son program should perform the EXEC-14 prior to any other executable statement which may cause the SAM buffer to be overwritten or deallocated.

If the son was scheduled with wait the son can pass a buffer back to the father. The son would place the data to be returned into IBUFF, specify the length in ILEN and perform an EXEC-14 call with IRWCD=2. The son should then terminate itself (EXEC-6), allowing the father to be rescheduled. The father could obtain the returned buffer from SAM by performing an EXEC-14 call with IRWCD=1.

A and B registers contain information which tells whether the call was successful or an error code in case of unsuccessful call.

c)Program time scheduling EXEC 12

This places the calling or any other program into the time scheduling list. The EXEC-12 call has two modes of operation.

1) Absolute Time Mode - A program can be scheduled to run once at an absolute time or run repeatedly at specified intervals with the first run to be at an absolute time.

2) Initial Offset Mode - A program can be scheduled to run once at a time offset from the current time or to run repeatedly at specified intervals with the first run to be at a time offset from the current time.

CALL EXEC(ICODE,INAME,IRESL,IMULT,IHRS,IMIN,ISEC,IMSEC)

ICODE -Request code 12 = time scheduling

INAME -Program name , set to 0 if the calling program is to be scheduled.

IRESL -Resolution code 1-10's of milliseconds
2-Seconds
3-Minutes
4-Hours

IMULT -Execution multiple that specifies the time interval between runs for programs that run repeated-

ly . It is used in conjunction with IRESL. IMULT = 0 indicates that the program is to run once.

IOFST -Initial offset, negative integer that specifies an offset from. current time that a program will first run. It is used in conjunction with IRESL.

IHRS,IMIN,ISEC,IMSEC -specify hours,minutes,seconds and milliseconds respectively when a program will first run.

It should be noted that the program must be dormant at that time when it is to be scheduled or the call is ignored.

Run repeatedly- IMULT is set in conjunction with IRESL to specify the interval between run times. The program to be scheduled must be dormant at the indicated times or the call is ignored for that time period. The future scheduling times are not affected.

IOFST must be less than 24 hours.

For unsuccessful calls the A and B registers will contain error information.

d) Program swapping control EXEC 22

This allows a program to lock itself into memory i.e. the program performing a call will not be swap-

ped out if a higher priority program needs its partition. It also allows the program to release the lock.

CALL EXEC(ICODE,ILOCK)

ICODE -Request code 22 = swapping control

ILOCK -Lock control 1 = program cannot be swapped.
0 = program can be swapped.

Memory lock considerations - In order for a program to be allowed to lock itself into a memory partition, the memory lock feature must be enabled at generation time. The memory-lock bit in program's ID (Identification) segment is cleared if the program aborts or terminates except if the program terminated because of an EXEC call with saving of resources. This feature enables Real-time programs to be always dormant by locking them to the memory so that they can be scheduled in time correctly.

e) Program completion EXEC 6

It notifies the operating system that the calling program wishes to terminate itself or a subordinate program.

CALL EXEC(ICODE,INAME,ICMCD,IOP1,IOP2,IOP3,IOP4,IOP5)

ICODE -Request code 6 = program termination

INAME -Program name , set to 0 if calling program wishes to terminate itself

ICMCD -Completion code

IOP1 - IOP5 - Optional parameters , when INAME = 0 parameter values are saved in terminating program's ID segment and they can be recovered when the program next executes.

ICMCD = 1, saves resources , makes program dormant , saves suspension point and all resources allocated to the program . When the program is rescheduled it will continue executing from the point of suspension and have access to previously allocated resources.

ICMCD = 2 terminates program and removes it from the time list. If the program is input/output suspended, the system waits until the input/output operation completes before setting the program dormant.

ICMCD = 3 immediately terminates program . It removes it from time list and releases all disk tracks allocated to it.

STATUS EXEC CALL

The status EXEC calls are used to obtain information about the operating environment.

f) System status time request EXEC 11.

It requests the current time indicated by the 24 hour real time clock.

CALL EXEC(ICODE,ITIME,IYEAR)

ICODE - Request code 11 = Time request

ITIME - 5 word array Time indicated by a Real-time clock is returned by the system as follows:

ITIME(1) = 10's of milliseconds

ITIME(2) = Seconds

ITIME(3) = Minutes

ITIME(4) = Hours

ITIME(5) = Day of the year

This call was particularly useful in the measurement of execution time of various real time programs for which the execution time requirement was critical and to test the efficacy of various algorithms to achieve the best execution speed.

CHAPTER 4

ON-LINE DIGITAL FILTERING TECHNIQUES IMPLEMENTATION AND RELATED PROBLEMS

a) What is digital filtering ?

Digital filtering is a computational process or algorithm by which a sampled signal or sequence of numbers, acting as an input, is transformed into a second sequence of numbers called the output. Digital filters are of crucial importance in the processing of signals. The computational process may correspond to high-pass, low-pass, band-pass or band-stop filtering, integration, differentiation or something else(Ref 10). Thus the aims of digital filtering are the same as those of continuous(i.e. analog) filtering but the physical realization is different. Linear continuous filter theory is based on the mathematics of linear differential equations. Linear digital filter theory is based on the mathematics of linear difference equations.

Increasing speeds and decreasing costs of mini and microcomputers now make it possible to consider Real-time digital systems which perform filtering operations usually performed by analog hardware. However, the constraints on the maximum frequency which can be processed by such REAL-TIME digital filters

are almost entirely a consequence of the computation time needed to execute the difference equation within one sample interval.

b) Advantages of digital filtering over analog filtering

Real-time digital filters have several advantages over continuous filters. A greater degree of accuracy can be attained in the digital filter realization. A greater variety of digital filters can be built, since certain realization problems (akin to negative elements) do not arise(Ref 11). No special components are needed to realize filters with time varying coefficients. Aggregates of digital filters should be economic in the very low frequency band (0.01 to 1 HZ) where size of analog components becomes appreciable.

The digital filter approach offers many additional advantages over analog approaches(Ref 10).

*Changes resulting from variance in component values normally associated with filter capacitors and resistors as a result of temperature or aging - are non existent.

*Periodic calibration is eliminated.

*The performance from unit to unit is stable and repeatable.

*Great flexibility is available since filter response can be altered by changing arithmetic coefficients.

*Arbitrarily high precision can be achieved, limited only by the number of bits involved in the numerical computation.

The principal use of digital filter or numerical filter is to smooth a data series in the time or frequency domain, same as its analog counterpart i.e. low-pass, high-pass or band-pass etc.

Difference Equations of digital filters :

An mth order linear difference equation may be written as

$$Y(nT) = \sum_{i=0}^{r} Li*X(nT-iT) - \sum_{i=1}^{m} Ki*Y(nT-iT) \qquad (1)$$

Equation (1) emphasises the iterative nature of the difference equation; given the m previous values of the output Y and the (r+1) most recent values of the input X, the new output may be computed from (1).

Physically, the input numbers are samples of a continuous waveform and real time digital filtering consists of performing the iteration (1) for each arrival of a new input sample. Design of the filter consists of finding the constants Ki's and Li's to fulfil the given filter requirement. Real time filtering implies that the execution time of the computer program for computing the right side of (1) is less than T, the sampling interval. Simulation programs in which the computations are not carried out in real-time are understood to be simulations of Real-time digital filters in which the original sampling interval T of the continuous input remains a referrence parameter only. It should be realized that quantization as well as sampling is performed on the continuous data before entry into the program and also, that the multiplication indicated in (1) contains an inherent round-off error caused by the finite register length.

As shown in Ref 11 a useful pictorial representation of (1) may be given by elements of unit delays (of time T), adders and multipliers as shown in Fig 1. Such arrangements can be serial or parallel or combination of both as shown in Fig 2 and 3 respectively. Some arrangements may be preferrable to others from the computation and implementation point of view.

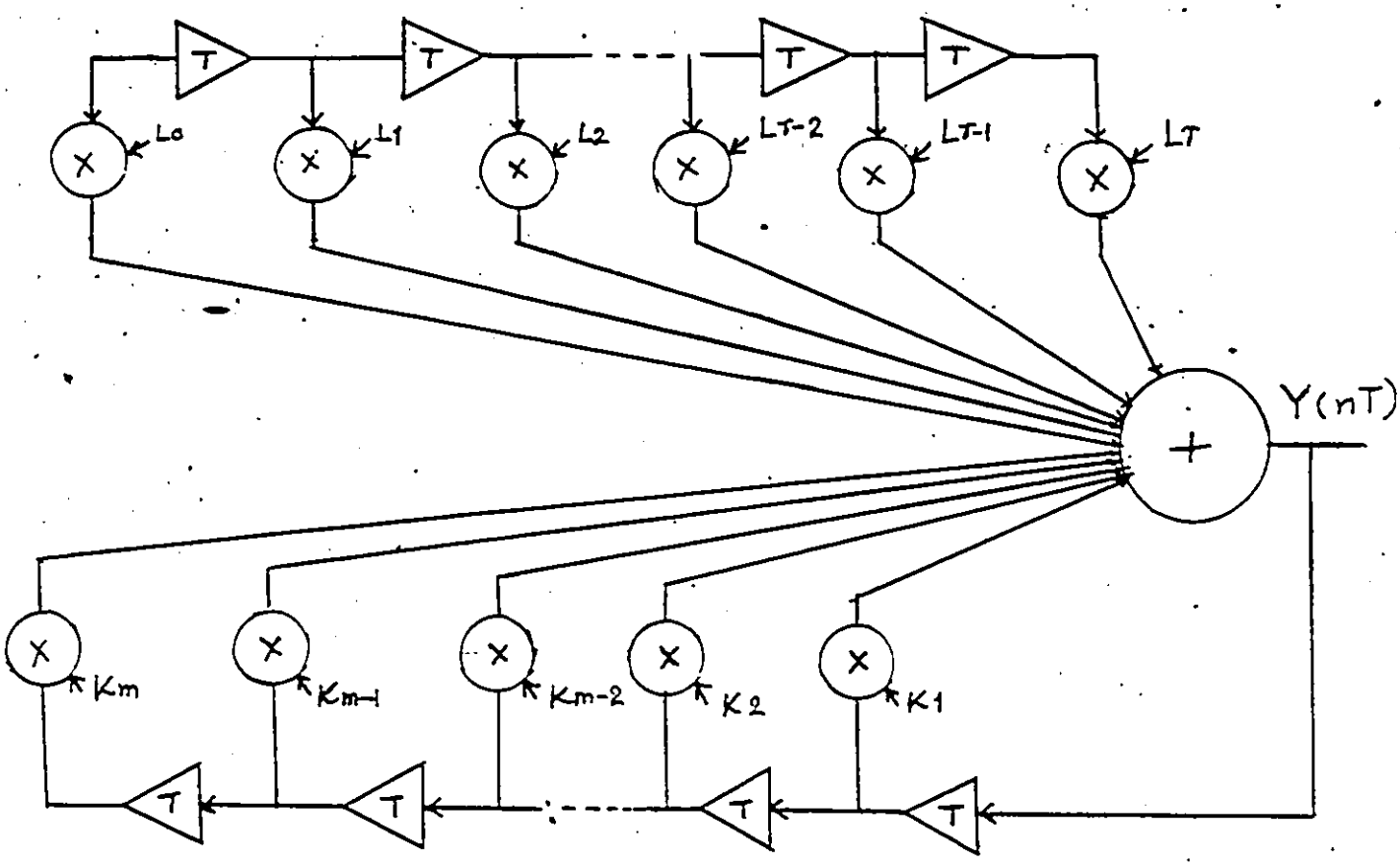


Fig 1 Pictorial representation of m^{th} order difference equation

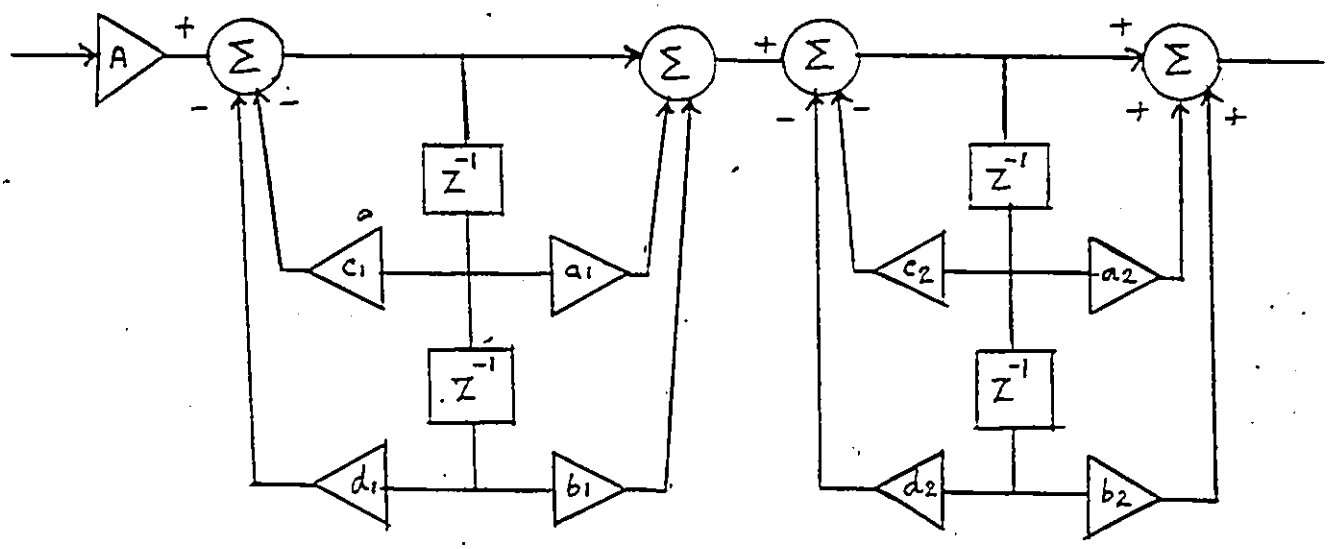


Fig 2 cascade realization corresponding to

$$A \prod_{K=1}^{K} \frac{1 + a_K \bar{z}^1 + b_K \bar{z}^2}{1 + c_K \bar{z}^1 + d_K \bar{z}^2}$$

Alternatively, the transfer function can be expanded into partial fractions as

$$H(z) = \sum_{i=1}^{m} Hi(z)$$

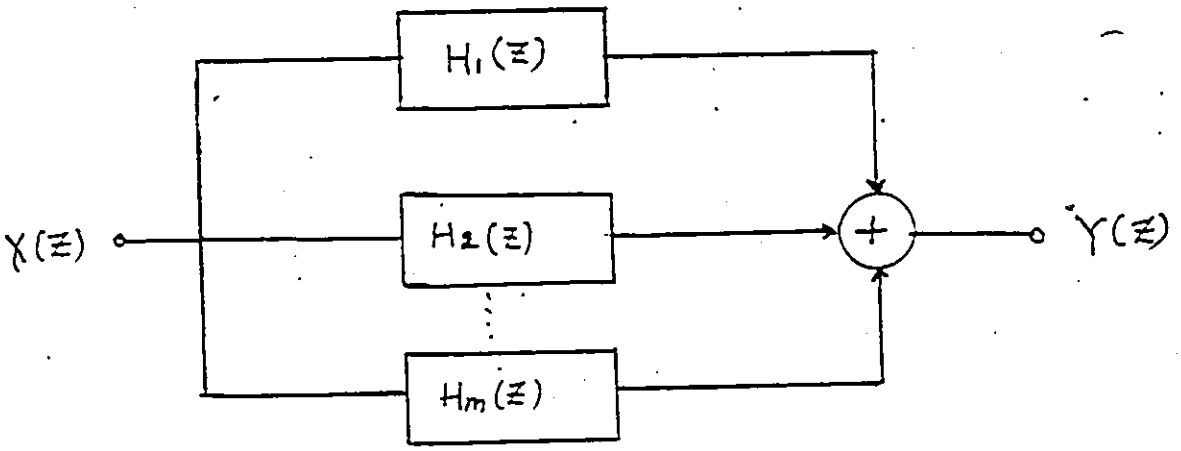
where

$$Hi(z) = \frac{a_{0i} + a_{1i} z^{-1}}{1 + b_{1i} z^{-1} + b_{2i} z^{-2}}$$

In this case

$$Y(z) = \sum_{i=1}^{m} Hi(z) X(z)$$

So parallel configyration is



Fig(3) where each $Hi(z)$ is similar to second order canonic form as shown in Fig 2

There are several methods of obtaining digital representations of analog systems. One of these, the Bilinear Transform method, is the natural method for solving the frequency domain approximation problem for digital filters. The method consists of transforming the open left half-plane into the interior of the unit circle, in a one-to-one fashion. Rational functions transform to rational functions and because the transformation is into and one-to one, for every digital filter represented by a rational function, there exists a unique analog filter represented by a rational function and vice versa.

The frequency-domain distortion is well defined. To every well-defined frequency domain specification for a digital filter, there corresponds a well defined specification for an analog filter. Thus many of the frequency-domain methods for designing analog filters are directly applicable to the design of digital filters.

The response of a non-recursive filter at instant nT is of the form :

$$Y(nT) = \sum_{i=0}^{N} Li*X(nT - iT) \qquad (1A)$$

Therefore, a linear, time-invariant, causal, non-recursive filter can be represented by Nth-order linear difference equation. N is the order of the filter.

The response of a recursive filter is a function of elements in the exitation as well as the response sequence. In the case of a linear, time-invariant, causal filter i.e., if instant nT is taken to be the present, the present response is a function of the present and past N values of the excitation as well as the past N values of the response.

$$Y(nT) = \sum_{i=0}^{N} Li*X(nT - iT) - \sum_{i=1}^{N} Ki*Y(nT - iT) \quad (1B)$$

Note that equation(1B)simplifies to equation(1A)if Ki=0, and essentially the non-recursive filter is a special case of recursive one.

The possibility of achieving physically realizable linear phase filters is an interesting facet of non-recursive digital filters. However in order to achieve a sharp cutoff characteristics, a non recursive filter of high complexity(high order) is required which is prohibitive in many Real-time applications. Thus in these cases recursive digital filters must be used. It is known that recursive fil-

ters, of necessity, introduce phase distortion and thus to achieve a linear phase characteristics phase equalization must be used.

d) The Bilinear Transform

This transformation has many significant properties :

a) It is a one-to-one mapping of the open left-half complex plane into the open unit disc.

b) It is a one-to-one mapping of the imaginary axis including the point at infinity, onto the unit circle.

c) It is a one-to-one mapping of the open right-half complex plane onto the complement of the closed unit disc.

d) The inverse transformation $Z = \frac{(2/T + S)}{(2/T - S)}$ has analogous reciprocal properties.

The use of the Bilinear transform allows the digital approximation problem to be solved by solving a corresponding analog problem and well established techniques can be brought to bear on the digital problem.

Suppose we have a stable analog filter described by H(S), its frequency response is found by evaluating H(S) at points on the imaginary axis of the S plane namely at S = jW . If the function H(S) is replaced by a rational function of Z which maps the imaginary axis of the S plane into the unit circle of the Z plane, then the resulting function H'(Z) evaluated along the unit circle will take on the same set of values as H(S) evaluated along the Imaginary axis.

Of course this does not mean that the functions are the same because the frequency scales are distorted relative to one another. This is illustrated by the simplest rational function which maps the jW axis onto the unit circle.

$$S = (Z-1)/(Z+1)*\frac{2}{T} \qquad (2)$$

Let the analog frequency variable be WA and let the digital frequency variable be WDT. Then the functions H(WA), H'(WDT) take on the same values for

$$WA = TAN(WDT/2)*2/T \qquad (3)$$

Note that the transformation (2) leads to a ratio of polynomials in Z. Since it maps the left half of the S plane onto the inside of the unit circle, we can be sure that it will always yield H'(Z) a realizable stable digital filter. Equations (2) and (3) yield a technique for designing a digital filter by analog techniques.

The procedure is as follows :

1) Note the critical frequencies and ranges (pass-band and stop-band maximum attenuation points etc) of the desired digital filter and call them WDiT

Compute a new set of frequencies WAi by

$$WAi = TAN(WDiT/2)*2/T$$

2) Design a transfer function H(S) with the properties of the digital filter at the new frequencies and ranges. There is of course no need to synthesize H(S).

3) Replace S by $(Z-1)/(Z+1)*\frac{2}{T}$ in H(S) and perform the algebra necessary to express the resulting H'(Z) as a ratio of polynomials. This yields the desired digital filter.

The Z transform calculus is the mathematical basis for the analysis and design of digital filters. Such filters are best understood by emphasizing the relations between the difference equations, the pictorial representation or block diagram, the Z plane geometry and the frequency response function. For the cases of design techniques, using either the standard Z transform or the Bilinear transformation, advantage is taken of available material in the field of analog filter design. In these cases it must be kept in mind

that the digital filter can never be exactly the same as the corresponding analog filter although the two may be made sufficiently equivalent in many practical situations by choosing appropriate sampling rates.

A digital filter acts on a sampled version of the signal to be filtered and may be realized through discrete logical hardware elements or by suitable programming of a digital computer which is fed with sampled version of the input signal. This chapter will be primarily concerned with the computer realization of digital filters.

The impulse-response function, in its discrete form as a number sequence, can be used to specify a filter in the time-domain. The sequence may be used directly to form a convolution of the signal with the response function or indirectly via a transformation in the frequency domain.

The advent of the fast Fourier transform algorithm has played an important role in the indirect filtering operation of the time series because this often proved more economical than the direct method. For fast convolution a block of output samples is obtained from a block of input samples by Fourier transforming the input, multiplying the transform by the Fourier transform of the impulse response, and then inverse Fourier transforming the product.

Where the same filtering requirements can be adequately met by several different digital filters, the choice between them depends on the speed of execution of a computer program which solves the difference equation. An important factor in this speed is the number of multiplications. Some digital filters are able to meet essentially the same requirements as others with significantly fewer multiplications per output sample, and these are therefore to be preferred. It is important to stress that speed of execution is the main limiting factor in the utilization of digital filtering methods in the real-time environment.

Frequency specifications in terms of amplitude and phase characteristics express the classical method of continuous analog filters and may also be used with discrete filters. Since the wealth of design information on the transfer characteristics of continuous filters is available, synthesis methods have been developed to permit conversion of these characteristics into discrete forms so that the design of a digital filter can proceed.

A linear system, such as a filter can be characterized by a complex transfer function expressed in Laplace form as :

$$H(S) = Y(S)/X(S) = (A0*S**n + A1*S**(n-1) + \\ + An+1)/(B0*S**m + B1*S**(m-1) + \quad + Bm) \qquad (4)$$

Equation (4) is a form of differential equation applicable to a representation of a continuous filters. In the discrete case we use the linear difference equation.

$$Yi = \sum_{K=1}^{P} Bk*Y(i-k) + \sum_{K=0}^{M} Ak*X(i-k) \qquad (4A)$$

For p = 0 and finite M this is known as FIR (finite impulse response) or non-recursive filter.

The two classes of filters have quite different properties. The FIR filter represents the summation of a limited number of input terms and thus has a finite memory. It has excellent phase characteristics but requires a large number of terms to obtain relatively sharp attenuation characteristics in the frequency domain.

The IIR(Infinite Impulse Response) filter represents the summation of both input and output terms so that it can be considered as having infinite memory. It requires relatively fewer terms for similar attenuation characteristics but will posses poorer phase performance than the equivalent FIR filter.

The appropriate transform for the difference equation is the Z-transform which performs the same role with difference equations as the Laplace transform carries out for differential equations. The use of the Z transform permits the specification of a digital filter from the continuous transfer function directly in terms of delays, multipliers and adders which is the correct form for hardware implementation. Its use for a 'software' filter(i.e. one realized by a computer program) permits a convenient form of expression particularly when recursive filters are considered.

e) Different approaches to realize digital filters

In summary we can recognize three fundamental techniques used in digital filtering.

1) Convolution (direct method)

2) Fourier transformation (indirect method)

3) Recursion (use of difference equation)

The first two techniques are generally confined to FIR filter and the third to IIR filter.

Any FIR filter can be realized nonrecursively by direct convolution or fast convolution. Nonrecursive realizations require the coefficients of the filter

response h(nT). For direct convolution the output Y(nT) is determined explicitly from the inputs X(nT) as

$$Y(nT) = \sum_{m=0}^{N} h(mT)*X(nT-mT) \qquad (4B)$$

Essentially, the synthesis of a digital filter follows the form familiar with continuous (analog) filter. That is we need to choose the desired characteristics in the frequency domain , determine an acceptable approximation for this and finally synthesize the filter by suitable transformation and calculation of filter weights (or coefficients).

f) Analog-filter Approximation

The approximation problem in the design of recursive digital filters is usually solved by using the following analog-filter approximations :

1) Butterworth

2) Tschebyscheff

3) Elliptic

4) Bessel

The elliptic approximation is the most efficient of the aboye four, if prescribed loss specifications

are to be met. In the elliptic filter approximation, the pass-band loss oscillates between zero and a prescribed maximum Ap,(as in the Tschebyscheff approximation) and the stop-band loss oscillates between infinity and a prescribed minimum Aa.

The elliptic approximation is more efficient than the Bessel or Tschebyscheff, in that the transition between the pass-band and the stop-band is steeper for a given approximation order. This is the reason why this filter approximation was selected. Since computations involved to realize a digital filter by computer program are directly proportional to the order of the filter, the criterion of selection must be to get the best performance for the least computational effort.

g) Elliptic filter design approximation

An elliptic, normalised, low-pass filter with a selectivity factor k, a maximum pass-band loss of Ap dbs and a minimum stop-band loss equal to or in excess of Aa dbs has the transfer function of the form :

$$Hn(S) = Ho/Do(S) * \prod_{i=1}^{r} (S^{**}2 + Aoi)/(S^{**}2 + B1i*S + Boi)$$

where

$$r = (n-1)/2 \quad \text{for odd } n$$

= n/2 for even n

and

Do(S) = S + sigma for odd n

= 1 for even n

Ho = scale factor

The transfer function coefficients and multiplier constants Ho can be computed by using programs RLPFD, RHPFD (for low-pass and high-pass filter designs respectively) which make use of formulae listed in Ref 5

RLPFD - This is a program to design a Recursive Low-pass digital filter. Input parameters to the program are entered from the user terminal interactively.These are the following:

1) Ap - Maximum permissible ripple in the pass-band in dbs

2) Aa - Minimum attenuation in the stop-band in dbs

3) FP - Pass-band edge in HZ

4) FA - Stop-band edge in HZ

5) FS - Sampling frequency in HZ

The transfer function coefficients and multiplier constants Ho can be computed using the formulae in sequence :

K = FP/FA

K' = SQRT(1 - K*K)

Qo = 0.5*(1 - K**0.5)/(1 + K**0.5)

Q = Qo + 2.*Qo**5 + 15.*Qo**9 + 150.*Qo**13

D = (10.**0.1*AA - 1)/(10.**0.1*AP - 1)

N > = log(16*D)/log(1/Q)

LAMDA = (1/2*N) ln(10**0.05AP + 1)/(10.**0.05*AP - 1)

$$A = 2.*Q**0.25* \sum_{m=0}^{\infty} (-1)**m*Q**(m*(m+1)\sinh((2*m+1)*LAMDA)$$

$$B = 1 + 2.* \sum_{m=1}^{\infty} (-1)**m*Q**(m*m)*\cosh(2.*m*LAMDA)$$

SIGMA = A/B

W = SQRT((1+K*SIGMA**2)(1+SIGMA**2/K))

E = (2.*m + 1)* ΠI*MU/n $\pi = 4.ATAN(1.0)$

$$C = 2.*Q**0.25* \sum_{m=0}^{\infty} (-1)**m*Q**m*(m+1)*\sin(E)$$

F = 2.*m* ΠI*MU/n

$$G = 1 + 2.* \sum_{m=1}^{\infty} (-1)**m*Q**(m*m)*\cos(F)$$

OHMi = C/G

where MU = i for odd n

= i - 0.5 for even N and i = 1,2, r

r = (N - 1)/2 for odd N

= N/2 for even N

Vi = SQRT((1 - K*OHMi**2)(1 - OHMi**2/K)

AOi = 1/OHM**2

P = (SIGMA *Vi)**2 + (OHMi*W)**2

R = (1 + (SIGMA *OHMi)**2)**2

BOi = P/R

S = 2.*SIGMA *Vi

T = 1 + (SIGMA*OHMi)**2

Bli = S/T

HO = SIGMAO* $\prod_{i=1}^{\gamma}$ (BOi/AOi) for odd N

= 10.**(-0.05*AP)* $\prod_{i=1}^{\gamma}$ (BOi/AOi) for even N

After the design of the normalized low-pass filter follows a low-pass to low-pass frequency transformation

S = lamda*Sbar for LP to LP

and

S = lamda/Sbar for LP to HP

where

lamda = WN/WAI

WN = normalised cutoff frequency of 1 radian/sec

WAI = 2./T*TAN(WDI*T/2)

to take into account the warping effect due to the Bilinear transformation

After frequency scaling the coefficients AOi, BOi, Bli, HO and SIGMA these are changed to :

AATOi = AOi/(lamda**2)

BATOi = BOi/(lamda**2)

BATli = Bli/lamda

HTO = HO/lamda

SIGMATO = SIGMA /lamda

Then follows the usual Bilinear transformation to convert the S domain transfer function into the Z domain transfer function for digital filter realization

```
CCi = BATOi + 2.*BATli/T + 4./(T*T)
AOi = (AATOi + 4./(T*T))/CCi
Ali = 2.*(AATOi - 4./(T*T))/CCi
BOi = (BATOi - 2.*BATli/T + 4./(T*T))/CCi
Bli = 2.*(BATOi - 4./(T*T))/CCi
```

for i = 1 to n where n = number of second order stages in the filter.

```
AAO = T/(2. + SIGMATO*T)
BBO = AAO
CCO = (2. - SIGMATO*T)/(2. + SIGMATO*T)
```

This completes the design of the low-pass digital filter.

Implementation of the digital filter by a program :

Transfer function of the digital filter can be written as follows :

$$HD(Z) = H'(Z) * \prod_{i=1}^{r} (Aoi+Ali*Z+Aoi*Z*Z)/(Boi+Bli*Z+Z*Z)$$

where r = (N-1)/2 for odd

= N/2 for even N

and H'(Z) = (BBO+AAO*Z)/(CCO+Z) for odd N

= 1 for even N

We now consider realization of a single second order digital filter. The equation for a single second order filter can be rewritten as follows :

H2(Z) = N(Z)/D(Z)

where

N(Z) = AA0*Z**(-2)+AA1*Z**(-1)+AA0

and

D(Z) = BB0*Z**(-2)+BB1*Z**(-1)+1

Therefore the output Y(nT) of a second order filter can be written as :

Y(nT)=AA0*X(nT)+AA1*X(nT-T)+AA0*X(nT-2T)-BB0*Y(nT-2T)
-BB1*Y(nT-T)

f) Implementation of recursive digital filter by a computer program

Therefore if there are R second order stages in cascade then defining DI(I,J) and DO(I,J) as buffers for holding previous input and output samples of the filter we can write a program to realize the above filter as follows:

```
  DATA = X
  DO 50 J = 1,R      R NO. OF SECOND ORDER STAGES IN CASCADE
     DO 60 I = 3,2,-1
        DI(I,J) = DI((I-1),J)
60   CONTINUE
     DI(1,J) = DATA
     DATA = AA0(J)*DI(1,J)+AA1(J)*DI(2,J)+AA0(I)*DI(3,J)
```

```
      DATA = DATA-BB0(J)*D0(2,J)-BB1(J)*D0(1,J)
      D0(2,J) = D0(1,J)
      D0(1,J) = DATA
50    CONTINUE
      Y = DATA     OUTPUT OF THE FILTER
```

It is necessary to measure execution time of all the programs which run in Real-time and this exercise was carried out for two types of filter design approaches namely the recursive filter and the non-recursive filter. Comparison between the two filter realizations were done on the basis of :

1) Execution time

2) Memory requirement

3) Type of machine (E-series or F-series)

Execution times for these filter programs are made up of two contributions

1) Manipulation of the fixed length buffer which keeps the last N data samples in time consecutive order.

2) N multiplications and N-1 additions and the DO LOOP overhead.

m) Comparison between recursive and non-recursive filter implementations by a computer program : -

The following table gives comparison between recursive and non-recursive implementations of digital low-pass filters on the basis of the number of coefficients, execution time and memory storage required to hold previous data samples.

A low-pass filter with following specification was considered :

1) Pass-band edge 2 HZ
2) Stop-band edge 3 HZ
3) Pass-band ripple $\leq$ 0.5 db
4) Stop-band attenuation $\geq$ 40 dbs

For recursive filter design, Elliptic filter approximation, and subsequent Bilinear transformation, were used. For Non-recursive filter design, Fourier series analysis method was used. In order to reduce the Gibbs' oscillations h(nT) was preconditioned by Kaiser window function. (Design method described in Ref 5)

	Recursive filter	Non-recursive filter
No. of coeff	12	25
Memory storage in words(16-bit)	26	50
Execution time in millisec on E-series machine	1	2.73
Execution time in		

millisec on F-series machine. 0.478 1.92

The F-series machine is faster than the E-series machine because the former has a special Floating point processor. It is observed that a recursive filter realization is approximately two and a half times faster than a comparable non-recursive filter realization, and this becomes important especially when data from several channels is to be filtered in Real-time simultaneously. For example, in program AVEA1 the subroutine FILTR is used in sequential fashion to filter data coming from 8 different channels. To improve execution speed, this subroutine was modified to replace DO LOOP by simple assignment statements. This improved the execution speed almost by a factor of two. Of course this was at the cost of extra program memory. This tradeoff between memory requirement and execution time is always a factor to be considered. One should note that it is a distinct advantage of digital filters that they can be shared as a common resource provided that execution time requirements can be met. Naturally non-recursive filters, which require a large amount of memory and a larger number of computations than recursive filters with comparable performance, were not used in ON-LINE data analysis programs.

1) Implementation of non-recursive digital filter by

a computer program

Programs NRRWV, NRFPV, NRRWV and NRFRV are written to implement non-recursive digital filters by the direct convolution method. They are used for the purpose of measuring their execution time on either the E or the F-series machine. Programs NRFPV and NRFRV use the Vector Instruction Set library available on the F-series machine. This yields a significant improvement in execution speed as can be seen from the following Table. Vector Instruction Set is particularly useful in the realization of non-recursive filters (i.e. direct convolution type). This operation requires the multiplication of two vectors (called as DOT product), one vector representing the filter coefficients (Impulse response function in the time-domain) and the other representing the last N data samples. The latter data vector is carried in a fixed length push down stack with the most recent data sample always residing on the top of the stack. Alternative to the push down stack a ring buffer data structure may be used. Comparison based on type of data structure(push down stack or ring buffer) and whether the VIS library is used or not, for implementation of non-recursive filter, is given in the following table :

Non-recursive filter implementation with push down stack data structure and without use of the VIS library on F-series machine

Order of the filter	Execution time in millisec
25	1.92
50	3.83
100	7.83

Non-recursive filter implementation with push down stack data structure and use of the VIS library on F-series machine :

Order of the filter	Execution time in millisec
25	0.97
50	1.94
100	3.80

Non-recursive filter implementation with ring buffer data structure without use of the VIS library on F-series machine :

Order of the filter	Execution time in millisec
25	1.02
50	1.97
100	3.89

Non-recursive filter implementation with Ring buffer data structure using the VIS library on F-series machine :

Order of the filter	Execution time in millisec
25	0.55

50	0.77
100	1.40

From this table we can say that the non-recursive filter design can be given a second thought provided they are implemented on the F-series machine. This will enable one to take advantage of their desirable characteristics.

g) Design of recursive high-pass digital filter

On similar lines to the program RLPFD, the program RHPFD will accept specifications for a given high-pass filter and output the order of the filter and the filter coefficients. Detailed program listings are given in Appendix II. The program RLPFD is in fact used to design a recursive low-pass digital filter which is realised by the computer program RLPDF and it is used in the ON-LINE average analysis program in order to remove high frequency noise in the sampled data.

h) Design of recursive band-pass digital filter

A recursive band-pass digital filter is realized as a cascade of a high-pass and a low-pass digital filterswith appropriate specifications. Such a filter realisation could be used in an ON-LINE data analysis program which is to be used to locate positions of nodes and antinodes in a harbour model. This may be necessary because the wave data contain unwanted frequencies because of the resonant nature of a harbour basin.

The filtered output is then processed to find the running RMS value in Real time.

Programs RLPDF and RHPDF which are realizations of above described low-pass and high-pass filters, were written to GEDAP(Generalized Experiment control and Data aquisition and Analysis Package developed by the Hydraulics Laboratory, N.R.C Ref 9) compatible form to take advantage of GEDAP features. This supports the creation of GEDAP compatible disc files for the storage of data vectors which may be the input to and output from the filter. Subsequently the contents of these files can be listed on the terminal screen for quick checking of the results or they may be plotted on a variety of graphic plotting devices such as the VERSATEC plotter or an HP 2648 graphics CRT terminal using the GEDAP-GPLOT package. These plots can be either a gain versus frequency characterists of the designed filter or the filter time functions.

Detailed program listings for RLPFD & RHPFD and frequency versus gain response are given in Appendix I.

i),j),k) Design of non-recursive low-pass, high-pass and band-pass digital filters

Programs NRFLP & NRFHP are written to find filter coefficients for nonrecursive low-pass & high-

pass [redacted] filters for given specifications and program NRFIL similar to RLPDF is written to realize non-recursive filters by computer program in Real time. It was also used to find frequency versus gain response and execution time of these realizations. These prgrams were used to gather information given in Table II. Detailed program listings and frequency versus gain characteristics for nonrecursive filters is given in Appendix I.

CHAPTER 5

PUSH DOWN STACK AND RING BUFFER

a) Description of these data structures and their implementation by program

These are data structures often required to implement nonrecursive or recursive digital filters and, in ON-LINE computation (running estimation) of auto-correlation functions in the time domain , where it is necessary to hold the last N data samples in a memory buffer. One way to accomplish this is by means of the push down stack where the most recent data sample is put on the top of the stack while the oldest data sample is pushed out of the stack and is lost. An alternative way is to maintain a ring buffer in which data are not shifted but instead the index pointer to the most recent sample is updated every time a new data sample is available. In a ring buffer, the most recent data sample overwrites the oldest data sample.

Neither the push down stack nor the ring buffer structure is supported by hardware architecture of the Hewlett-Packard machines , and they must therefore be simulated by software. In case of the ring buffer, a pointer pointing to the next vacant location(i.e. pointing to the oldest data sample) must be com-

puted and its value must be compared to the buffer start index. Whenever its value is less than the start index, it is reset to the end index value. It is necessary to keep track of this pointer value at all times. In case of a push down stack it is necessary to shift N-1 data samples every time a new data sample becomes available(N is the number of previous samples retained) . This is a time consuming operation, however, its programatic realization is straight forward.

Both these structures were simulated and their performance was compared on the basis of execution time versus the number of data samples retained in the buffer.

By making use of the time request feature (CALL EXEC 11) of the RTE IVB operating system system and increasing the priority of the program (by operator's command) to ensure that this program alone executes and gets almost 100% CPU share (except for system overheads) one can measure execution time fairly precisely. Fig 4 shows the graph of execution time of two methods against number of data samples (type of machine used and with and without VIS library on the F-machine).

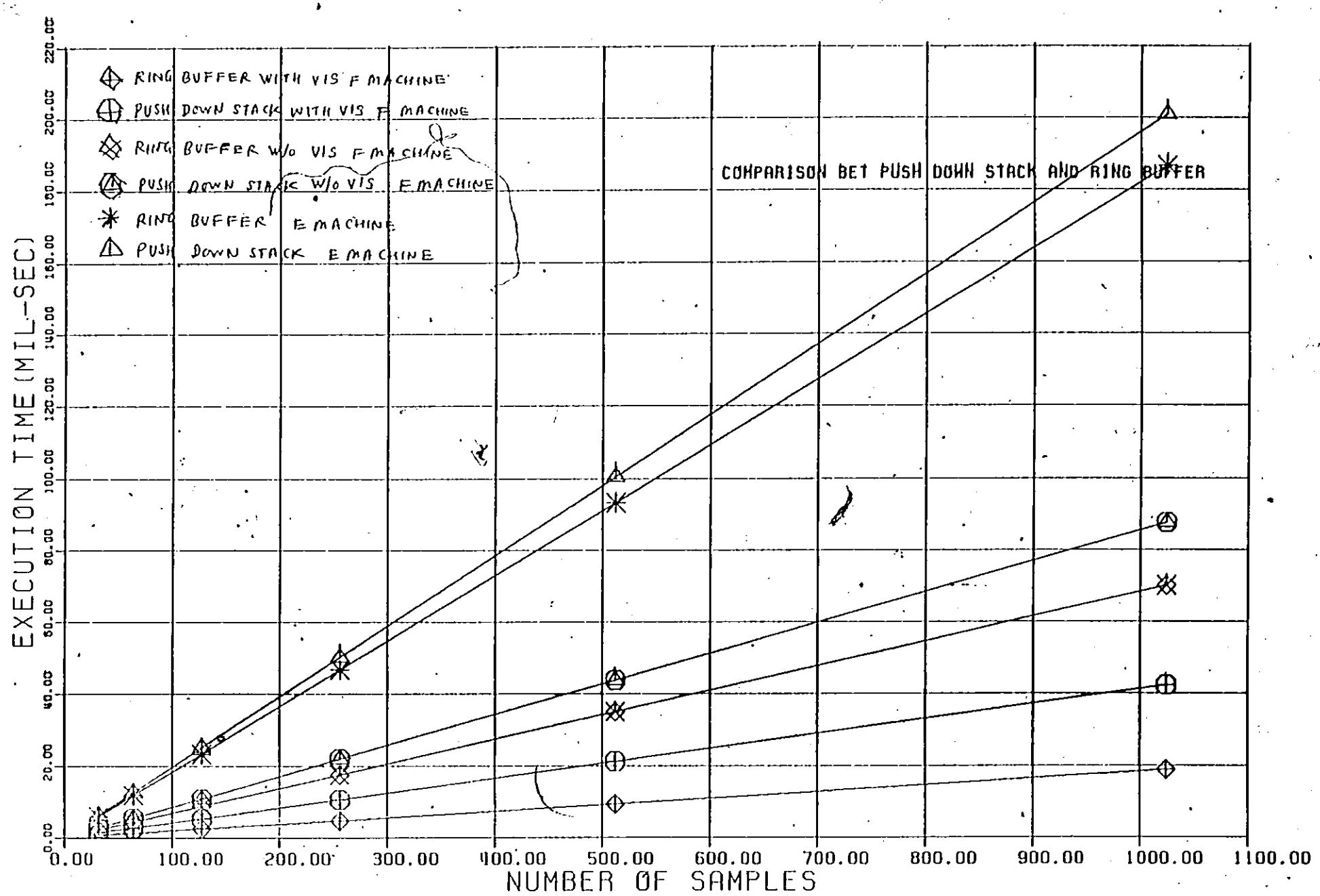

RING BUFFER WITH VIS E MACHINE
PUSH DOWN STACK WITH VIS E MACHINE
RING BUFFER W/o VIS E MACHINE
PUSH DOWN STACK W/o VIS E MACHINE
RING BUFFER E MACHINE
PUSH DOWN STACK E MACHINE
COMPARISON BET PUSH DOWN STACK AND RING BUFFER
EXECUTION TIME (MIL-SEC)
NUMBER OF SAMPLES
0.00
100.00
200.00
300.00
400.00
500.00
600.00
700.00
800.00
900.00
1000.00
1100.00
0.00
20.00
40.00
60.00
80.00
100.00
120.00
140.00
160.00
180.00
200.00
220.00

FIG 4

It is observed that the Ring buffer structure performs better than the Push down stack as can be anticipated from the fact that there are many more operations involved in the shifting of N-1 samples than there are in the simple pointer arithmetic of the ring buffer.

Programs to realize the push down stack and the ring buffer are given in PSHDS and RINGB listings. Table I shows their comparative performance. Program listings are given in Appendix III

We can make some intersting observations from the comparison between the push down stack and the ring buffer implementations on two different machines. In the E - series machine, the Ring buffer is faster than the Push down stack by about 6.5%. In the F - series machine, the Ring buffer is faster by about 25%. In comparison with the E - series machine, the implementation on the F - series machine is about 2.22 times and 2.72 times faster for push down stack and ring buffer respectively. In the F - series there is further improvement by a factor of 2 and 3.5 for the push down stack and the ring buffer respectively due to the use of the microcoded vector instruction set available on the F - machine. In conclusion we can state that the ring buffer implementation is recommended for time critical Real-time programs.

Table I

COMPARISON BETWEEN PUSH DOWN STACKS AND RING BUFFERS

IMPLEMENTATIONS ON E - SERIES MACHINE

Execution time in milliseconds

No. of samples	Push down stack	Ring buffer
32	6.25	5.93
64	12.65	11.88
128	25.15	23.35
256	50.23	46.75
512	100.48	93.37
1024	200.91	186.70

COMPARISON BETWEEN PUSH DOWN STACKS AND RING BUFFERS

IMPLEMENTATION ON F - SERIES MACHINE

Execution time in milliseconds

No. of samples	Push down stack	Ring buffer
32	2.81	2.18
64	5.47	4.38
128	10.90	8.75
256	21.91	17.53
512	43.82	35.01
1024	87.61	70.00

Talbe I (continued)

COMPARISON BETWEEN PUSH DOWN STACKS AND RING BUFFERS

IMPLEMENTATION ON F - SERIES MACHINE USING VIS

Execution time in milli-seconds

No. of samples	Push down stack	Ring buffer
32	1.4	0.7
64	2.7	1.3
128	5.4	2.5
256	10.7	4.8
512	21.3	9.3
1024	42.4	18.8

CHAPTER 6

A SUMMARY OF THEORETICAL CONSIDERATIONS FOR ON LINE ANALYSIS

Statistical properties

Mean value - The average or mean value of continuous data is

$$Xave = 1/Tn * \int_0^{Tn} (X(t)dt)$$

The corresponding value for a discrete set of N measurements Xi, is

$$Xave = 1/N * \sum_{i=1}^{N} Xi$$

For ON-LINE analysis the value of N is unpredictable and one cannot go on accumulating the sum and divide it by the total number of samples, because both the sum as well as the value of N will overflow as N becomes large. In addition one is also more interested in the estimation of the most recent value of the average. Therefore a running average is defined by a first order recursive smoothing relation (Ref 6).

$$AVE(I+1) = ALPHA*AVE(I) + (1. - ALPHA)*X(I) \qquad (5)$$

where a typical value of ALPHA is taken as 0.95 and it is always less than and close to 1. As ALPHA approaches unity it takes more time to reach a final value and the response time increases. On the other

hand, if ALPHA is smaller then the response is faster but fluctuations in the estimated value are higher. This recursive relation represents a first order low pass RC filter. Using Z transform analysis we can find the unit step response of this recursive low-pass filter as shown below :

$Y(nT) = 0.95*Y(nT-T) + 0.05*X(nT)$

Taking Z transform of both the sides

$Y(Z) = 0.95*Y(Z)/Z + 0.05*X(Z)$

Hence filter transfer function is

$H(Z) = Y(Z)/X(Z) = 0.05*Z/(Z-0.95)$

If $X(nT) = U(nT)$ then $X(Z) = Z/(Z-1)$

therefore unit step response is

$Y(Z) = H(Z)*X(Z) = 0.05*Z*Z/(Z-0.95)*(Z-1)$

Taking inverse Z transform we have

$Y(nT) = 0.05*\delta(nT) + U(nT-T) - 0.9025*U(nT-T)*(0.95)**(n-1)$

n	Y(nT)
10	0.40
20	0.64
30	0.79
50	0.92
70	0.97
100	0.99

As an example, suppose T = 0.1 second, then the filter will have reached the final value within 1% in 100 steps = 10 seconds.

The advantage of this recursive relation is that it is necessary to store only one previous value. It is very simple to evaluate and gives fairly good estimates of a running average, but that depends surely on the frequency content of the fluctuations. That is really why a good low-pass filter may be better in some situations.

An alternate method for the computation of the running average is based on the definition of the mean value for N discrete values :

$$X\ ave = 1/N * \sum_{i=1}^{N} Xi$$

This formula is then applied to a record of finite length of the last N data samples. Every time a new data sample arrives, the oldest data sample is discarded and the new data value is stored in its place.

The Ring buffer implemetation is particularly efficient for this application. Since the pointer always points to the oldest data value which is being discarded at the same time a new data value arrives, the sum may be updated by the formula :

```
      J  = Pointer value
    SUM  = SUM - DATA(J) + X
DATA(J)  = X
      J  = J - 1  IF J.EQ.0  THEN J = N
   Xave  = SUM/N
```

The greatest disadvantage of this method is that it requires larger memory as do all non-recursive filters except this one is more efficient because the filter response function is uniform and therefore need not be stored. The estimated value strictly represents the average over last N data samples because of the nonrecursive nature of the formula.

The larger the number of samples retained in the buffer the better is the estimated value of the mean. This is, however, at the cost of a larger memory requirement . It should be noted that, in case of ring buffer implementation, the time overhead to maintain the storage is constant irrespective of the size of the buffer, whereas it increases in case of a push down stack structure.

The disadvantage of the above method namely the large memory storage requirement can be partially eliminated by some programming gimmick. This results into a method which can be called a variable order Non-recursive method of finding a running average.

The following algorithm accumulates the sum of data samples in SUM, and keeps track of how many data samples have been summed up in a scalar N. As N increases and equals 100,200,300,400 and 500, partial sums of samples from 1 to 100, 101 to 200, 201 to 300

etc are removed from the scalar SUM and are stored in locations SUM01, SUM12, SUM23 etc respectively. When N equals 500 the scalar SUM is decreased by an amount SUM01 and N is also decreased by 100 . The partial sums SUM01,SUM12,SUM23 etc are pushed down on the stack by one location (hence old SUM01 is lost and replaced by SUM12) . At any instant the running average is AVE=SUM/N. In the long run N will vary in the range from 400 to 500.

This effectively represents a variable order nonrecursive filter with the special property of having an impulse response function which is 1 over the entire duration of its response. Because of this special property it is possible to eliminate the need for large memory requirement for individual past samples and for impulse response function itself, both requirements associated with nonrecursive filters. Because the scalars SUM and N are decremented everytime they exceed a certain value, it can be ensured that they will never overflow.

This approach to calculating running averages can be easily extended to the computation of running RMS values. It was observed that for simulated as well as real random wave data from the model studies, results were considerably better than those obtained with the simple recursive filter.

The following program shows an implementation of above approach. Values of N and n (corresponding to the scalar SUM and partial sum) can be changed easily in order to suit particular situation.

```
X - data value read from real-time buffer
Initialise N.
   N = 0
Accumulate sum
 SUM = SUM + X
   N = N + 1
 IF(N.GT.400.AND.N.LT.500) 1090,1050
1050 IF(N.EQ.400) 1060,1070
1060 SUM34 = SUM - SUM01 - SUM12 - SUM23
     GO TO 1090
1070 IF(N.EQ.500) 1080,900
1080 SUM45 = SUM - SUM01 - SUM12 - SUM23 - SUM34
     SUM = SUM - SUM01
     SUM01 = SUM12
     SUM12 = SUM23
     SUM23 = SUM34
     SUM34 = SUM45
         N = N - 100
     GO TO 1090
900  IF(N.EQ.100) 1000, 1010
1000 SUM01 = SUM
     GO TO 1090
1010 IF(N.EQ.200) 1020,1030
```

```
1020 SUM12 = SUM - SUM01
     GO TO 1090
1030 IF(N.EQ.300) 1040,1090
1040 SUM23 = SUM - SUM01 - SUM12
1090 CONTINUE
* Now  calculation of AVE = SUM/N and its
*    display  can follow
* Program code is arranged in such a way that redundant
* checks on the value of N are avoided in the long run
* once N exceeds 400.
```

MEAN SQUARE VALUE

A single figure describing the intensity of random data is the mean-squared value. This is the average of the squared values of time history record which for continuous data is

$$X\ msq = 1/Tn * \int_0^{Tn} (X(t)**2*dt)$$

and for discrete data

$$X\ msq = 1/N * \sum_{i=1}^{N} Xi*Xi$$

Variance - A signal can consist of a slowly varying trend or low frequency component upon which are superimposed rapidly changing higher frequency

components. In such a case the mean value will represent the average base-level of the signal about which it is fluctuating at the higher rate.

A measure of the scatter of a signal about its average mean value is given by the variance of the signal which is frequently of interest in scientific measurements. This is given as as the mean-square value about the mean, namely for a continuous signal :

$$V(X) = 1/Tn * \int_0^{Tn} (X(t) - Xave)^{**}2*dt$$

and for discrete data

$$V(X) = 1/N * \sum_{i=1}^{N} (Xi-Xave)^{**}2$$

Standard deviation

The instantaneous deviation of a signal from its mean value, is given as

$$DEL\ X = Xi - Xave$$

Standard deviation is the square-root of the variance. Many engineers often refer to it loosely as the RMS value, which is true if the average is equal to 0, but wrong otherwise.

ROOT MEAN-SQUARE (RMS) VALUE

The positive root of the mean-square value is called the RMS value of the signal i.e. SQRT(X msq).

From probability theory

$$V(X) = E(X*X) - E(X)*E(X)$$

$$E(X*X) = 1/N* \sum_{i=1}^{N} (Xi*Xi)$$

$$E(X) = 1/N* \sum_{i=1}^{N} (Xi)$$

In practice to estimate the running V(X) one must compute both the running mean square value as well as the running mean by first order recursive smoothing relation :

```
SQR = ALPHA*SQR + (1. - ALPHA)*X*X
AVE = ALPHA*AVE + (1. - ALPHA)*X*X
```

and

```
RMS = SQRT( AMAX1(0.,(SQR - AVE*AVE)))
```

An alternative method similar to the one described in estimation of the averages, namely the use of the rectangular, non-recursive low-pass filter of engtn N was tried. Its performance for a buffer size of 128 samples was relatively better than the previous method. The larger the buffer size better the performance. Theoretical reason for the difference in per-

formace in estimating the running RMS can be found by computing gain versus frequency response of two filters.

Frequency response of first order recursive relation is given by

G(w) = 0.05/SQRT(1.9025 - 1.9*COS(wT))

Fig 5 shows frequency response for first order low-pass recursive filter for ALPHA = 0.95 and 0.98 and T = 0.1 and 0.05 second. It is observerd that for larger value of ALPHA smoothing is improved at the expense of response time.

Similarly frequency response of nonrecursive low-pass filter with rectangular window characteristics is given by

G(w) = (1/N)*SIN(0.5*N*w*T)/SIN(0.5*w*T)

Fig 6 shows the comparison of gain versus frequency response for the recursive and the non-recursive filter(128 stage), and it is evident from the figure why a 128 stage non-recursive filter gives better estimation of the running RMS, than the simple first order recursive filter. However, which method is to be used, depends upon the purpose of the measurement. For example if one is making adjustment of a potentiometer which controls a driving signal to a wave machine, then the operator should be able to see im-

EQUATION OF FIRST ORDER LOW-PASS RECURSIVE FILTER

$$Y(nT) = \alpha * Y(nT-T) + (1.-\alpha) * X(nT)$$

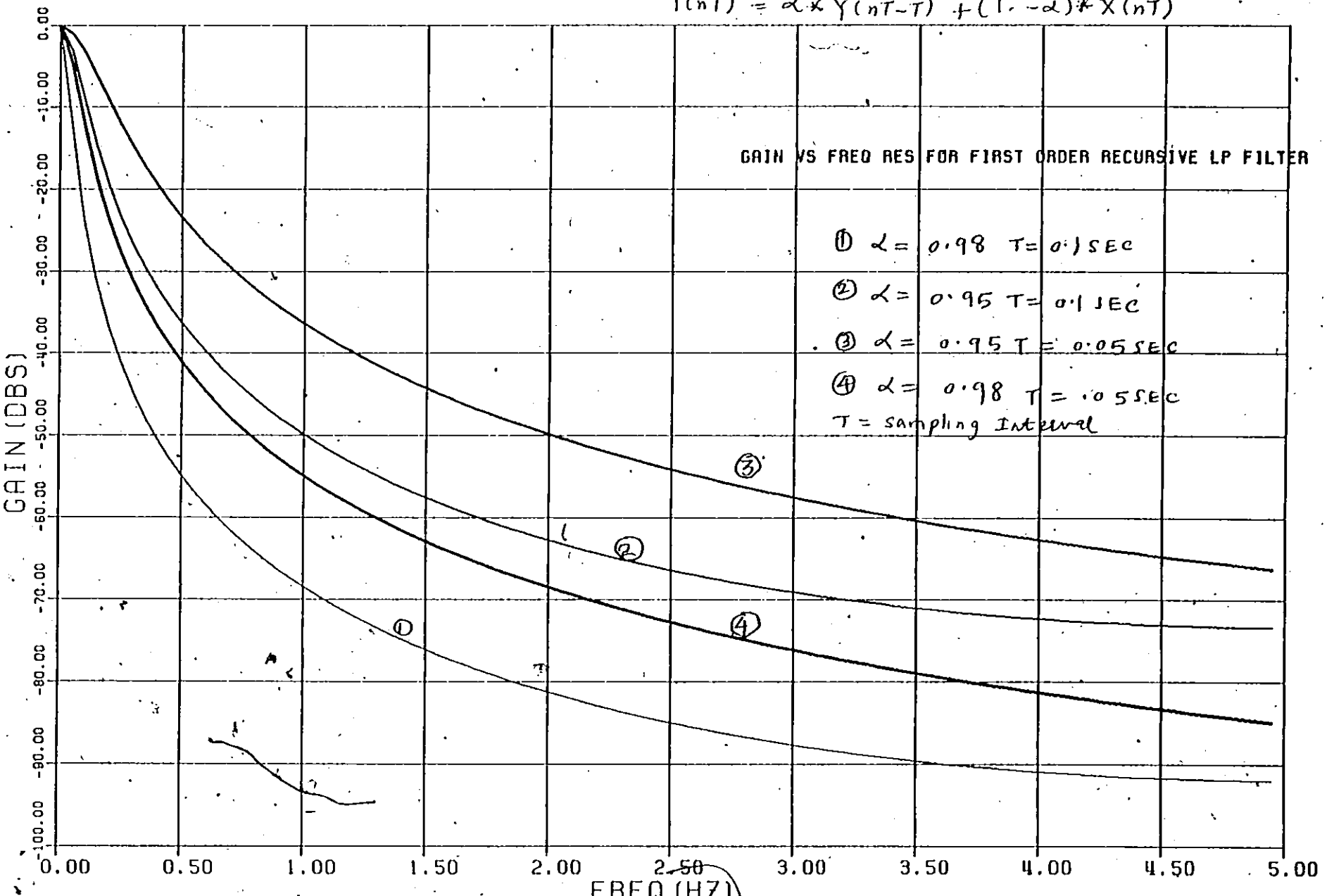


FIG 5

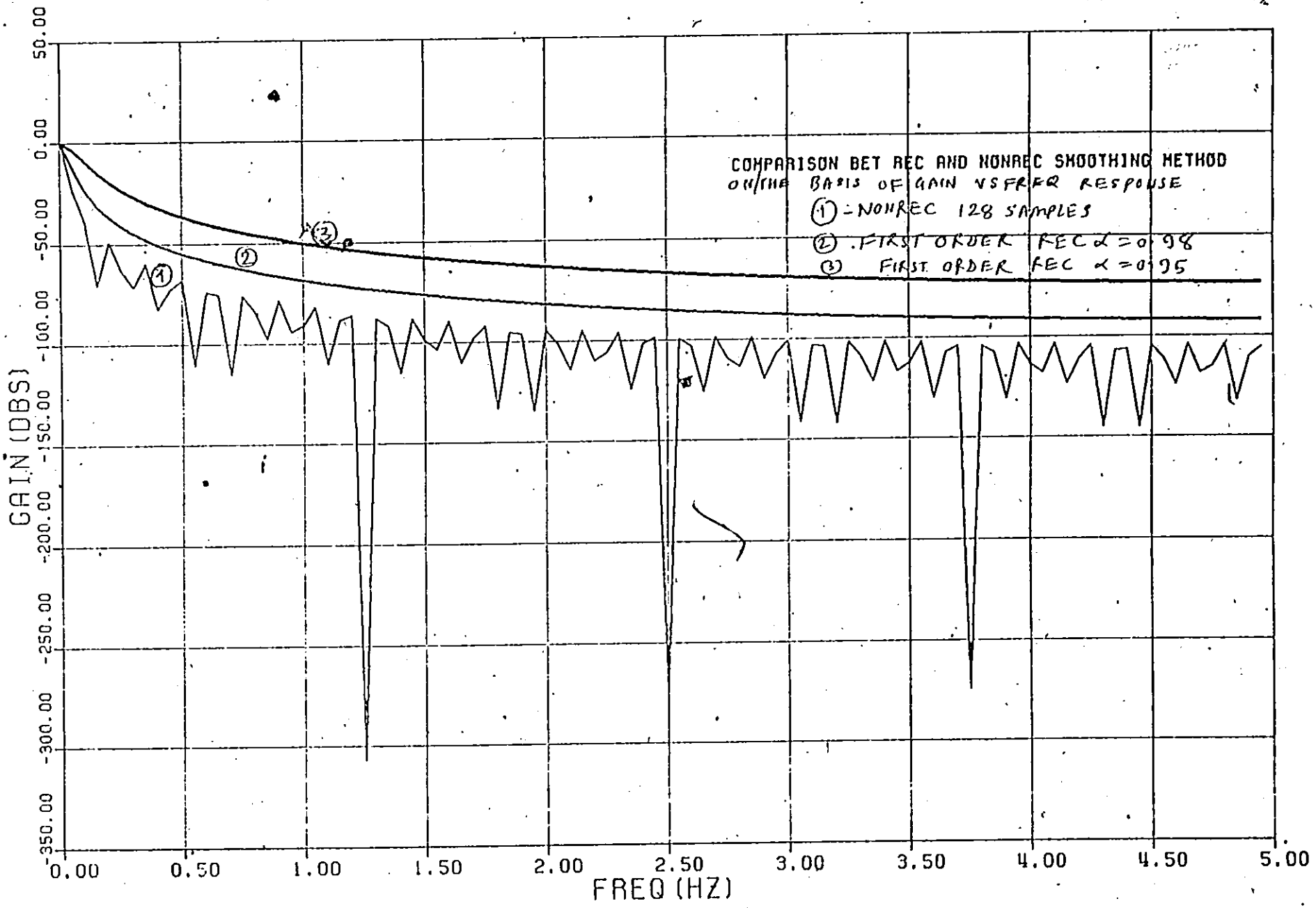

COMPARISON BET REC AND NONREC SMOOTHING METHOD
ON THE BASIS OF GAIN VS FREQ RESPONSE
① = NONREC 128 SAMPLES
② FIRST ORDER REC α = 0.98
③ FIRST ORDER REC α = 0.95
GAIN (DBS)
50.00
0.00
-50.00
-100.00
-150.00
-200.00
-250.00
-300.00
-350.00
0.00
0.50
1.00
1.50
2.00
2.50
3.00
3.50
4.00
4.50
5.00
FREQ (HZ)

FIG 6

mediately the effect of his adjustment, in that case a recursive method with a smaller value of ALPHA(e.g 0.95) would be appropriate. After the adjustment has been made, he may change ALPHA to a higher value (e.g.0.97) for more precise estimation of the RMS value.

The following formula was used with a ring buffer implementation :

```
SUM = SUM + X - DATA(J)
SQR = SQR + X*X - DATA(J)*DATA(J)
DATA(J) = X
J = J - 1  IF NEW J.EQ.0 THEN J = N
AVE = SUM/N
RMS = SQRT( AMAX1(0.,(SQR/N - AVE*AVE)))
```

It must be noted that this improvement in performance is at the cost of extra memory required to retain the last N data samples. In some cases it may be difficult to spare this additional memory. This estimated variance is applicable to history of last N data values because of nonrecursive nature of the formulation.

ESTIMATION OF SMOOTHED POWER SPECTRUM

Given a random function of time X(t) and of length Tn, then the estimation of the auto-correlation function is defined as (Ref 8)

$$ACF(T') = 1/(Tn-T') * \int_0^{Tn-T'} (X(t)*X(t+T'))*dt$$

which is usually evaluated over the range $0 \leq T' \leq Tm$

The estimation of the mean value of X(t) is

$$Xave = 1/Tn* \int_0^{Tn} (X(t)*dt)$$

The estimation of the auto-covariance function is given by :

$$Cx(T') = 1/(Tn-T')* \int_0^{Tn-T'} (X(t)-Xave)*(X(t+T')-Xave)*dt$$

$$= ACF(T') - (Xave)**2$$

The estimation of the standard deviation of X(t) is readily available from

$$X = SQRT(Cx(0))$$

and the estimation of the normalized covariance function, which is generally required, is given by :

$$Rx(T) = Cx(T)/Cx(0)$$

The single-sided power spectral density function may be derived from the auto-covariance function by means of a Fourier transform of an appropriately weighted auto-covariance function, i.e.

$$Sx(w) = 4.* \int_0^{\infty} (Cx(T')*D(T')*COS(wT')*dT'$$

where D(T') is a weighting function that is called a lag-window. The Fourier transform of D(T') is the spectral window, which may be chosen according to one's preference. The lag-window is finite over the interval -Tm <= T' <= Tm and zero everywhere else, so that the integral in the equation needs to be evaluated over this interval only.

For the purpose of numerical evaluation, the random function of time is sampled at regular intervals T. There are, therefore

$$N = Tn/T$$

samples in one record.

The auto-correlation function ACF(T') is given for discrete values of T' as:

$$ACF((J-1)*T) = 1/(N-J+1)* \sum_{I=1}^{N-J+1} (X(I*T)*X((I-J+1)*T)$$

and J = 1,2, M+1

and where M = Tm/T

Simultaneously one may obtain the average

$$Xave = 1/N* \sum_{I=1}^{N} (X(I))$$

and then the variance

$$C(0) = ACF(0) - Xave**2$$

It is evident that we have to retain last M data samples if auto- correlation function is to be evaluated over M*T time interval. The value M depends upon two factors :

1) This is decided on the basis of desired resolution for

Wc = 1/M*T

2) It must meet the requirement that the above computations of the cross products and summations can be completed for M shifts in a time less than the sampling interval T.

As usual we use a first order recursive relation to estimate the smoothed value of various cross products.

ACF(I,0) = ALPHA*ACF(I-1,0) + (1.-ALPHA)*X(I)*X(I)

ACF(I,1) = ALPHA*ACF(I-1,1) + (1.-ALPHA)*X(I)*X(I-1)

ACF(I,M) = ALPHA*ACF(I-1,M) + (1.-ALPHA)*X(I)*X(I-M)

GAMA(J) = D(J)(ACF(I,J) - Xave**2)

for J = 0 to M

M $\leq$ = 128

D(J) = 0.5*(1. - (J-1)/M)

for J = 1 to M

Xave = ALPHA*Xave + (1. - ALPHA)*X

Sx(w) = FT(GAMA(T')) T' = 0 to M*T

FT stands for Fourier transform

For ON-LINE computation of the auto-correlation function, it is necessary to find cross products between the most recent data sample and previous m-1 data samples, and add them to previous values.This is nothing but a scalar and vector multiplication and addition of two vectors. This can be done efficiently by using the powerful vector instruction set in F-series machine so that execution becomes faster due to the elimination of the DO LOOP overhead.

CHAPTER 7

A PROGRAM FOR ON-LINE RMS ANALYSIS

This program finds the running standard deviation of incoming wave data by means of a first order recursive low-pass filter and displays the results on the user's terrminal screen. Displayed values are updated at a rate of 1 or 2 per second. From this, the experimenter can estimate the significant wave height which is four times the RMS value and for irregular waves he can check for correct running conditions of the model in real time . If he wishes, he can exercise some control over the experiment on the basis of the ON-LINE computed information.

PROGRAM INPUT

When the program is invoked, it expects the following information from the user.

1) Data sampling rate

2) Display update rate

3) Name of calibration file for the analog I/O port

4) Analog I/O port number to which his transducers are connected

5) Analog channel number for which the RMS value is to be computed and displayed

Programming consideration

The program can be divided into two parts.

1) Initialization part which initializes the program. It runs only once when the program is invoked for the first time.

2) ON-LINE analysis part which computes the running standard deviation of wave data comming from the assigned port and channel. Only this part is repeatedly scheduled and executed. It displays the results on the user's terminal screen at a rate depending on the value the user has entered during the initialization phase.

Initialization part- It accepts interactively the input from the user and makes use of the GEDAP library subroutine to read the user specified calibration file for the port. This part also calculates parameter values for a self-time schedule EXEC call.

ON-LINE analysis part- This part makes use of an RTE IVB operating system feature through the use of an EXEC call. This enables the program to schedule itself repeatedly in Real-time and save its resources when it is running continuously .It also locks itself

to physical memory so that it cannot be swapped out of the memory by other high priority programs contending for the memory resource. In addition this will ensure that program is always dormant at times it is to be scheduled.

If the program is not available at the time when it is to be scheduled, the system reports an error and ignores the call. Therefore it is necessary to lock all real-time programs to the physical memory for their correct operation.

The display part of the program makes use of a special control character string which, when received by the microprocessor based CRT terminal, positions the alphanumeric cursor at the desired location. This is how a computational result which is to be displayed on the screen can be locked at fixed position relative to the screen. As a consequence the user will see most of the same values displayed at all time with only a few digits varying from update to update.

During the development phase of this program, its performance was checked by inputing simulated sinusoidal data of known RMS value and frequency. The performance of ON-LINE RMS calculation algorithm was checked at various frequencies in the range 0 to 3 HZ which is the frequency range of interest normally en-

countered during experiments in hydrodynamics. Performance was found to be satisfactory in the range 0.3 to 3 HZ . Later on smoothing parameter ALPHA was changed in the range 0.95 to 0.98. It was observed that the performance at low frequencies improved as ALPHA increased but it also increased the response time of the first order filter to an unacceptable degree. Therefore a best compromise between the response time and the frequency response was achieved by setting ALPHA = 0.95.

Subsequent to these simulation tests, actual measurement data were picked up from the Real-time unlabled common buffer. Real-time common is defined at system generation time and it is shared by many cooperating programs such as ON-LINE data aquisition and ON-LINE analysis programs as well as ON-LINE data storage programs for mass storage. Thereafter the reading of calibration file for the selected analog I/O port was also included in order to convert millivolt data into engineering units.

Detailed listing of this program is given in Appendix IV. The program has been tested with real data acquired during the experiment and results of this experiment were within reasonably close agreement with the results calculated by separate off-line analysis procedures.

CHAPTER 8

A PROGRAM FOR ON-LINE AVERAGE ANALYSIS

This program is written for Example 1 mentioned in chapter 1. A model ship is tied to its mooring by six lines which pull the ship against elastic fenders. All lines must be pretensioned to a specified levels.

Pretension adjustment on one line affects the setting on all other lines. In order to bring the total system to the required setting, it is necessary to monito. the forces on all lines at the same time until all lines have the desired setting. Therefore monitoring of six forces simultaneously represents a Real-time task.

Also here the program is divided into two parts :

1) Initialisation part- This part interactively accepts the information from the user terminal, such as the name of the calibration file for the port, the data sampling rate, the display update rate, the analog I/O port number to which transducers are connected and the channel number of first channel to which the first transducer is connected. This part executes only once. After that the program locks itself to the memory and schedules itself in a repetitive mode at a rate which corresponds to the data sampling rate.

Every time the ON-LINE analysis operation is over, the program terminates itself with saving of resources.

2) ON-LINE analysis and Display part- In this ON-LINE analysis operation, both the low-pass digital filtering and the display of the average is executed. Whereas the filtering operation must take place at the data sampling rate, the display function whose update rate is usually as low as 1 or 2 updates/seconds, can run at a low priority. It usually takes in the order of 100 milliseconds to display the readings on the terminal screen but the display function may complete its operation in as much as 500 milliseconds. To deal with this wide range of display times a separate display program DISP1 is written which is scheduled by the ON-LINE analysis program AVEA1 along with the passage of the buffer which contains the values to be displayed. This scheduling operation is without wait so that the father program AVEA1 need not wait for the display program DISP1 to complete and the latter can take its own time to complete. This is achieved by the RTE IVB system's EXEC call feature which enables this operation. After being scheduled, the SON program DISP1 retrieves the buffer passed by the FATHER program AVEA1 and displays the data contained in the buffer. It is only because of

these RTE IVB features that it has been possible to achieve this multiprogramming capability.

Detailed listings for AVEA1(FATHER) and DISP1 (SON) programs are given in Appendix V

CHAPTER 9

A PROGRAM FOR DETERMINATION OF NODES AND ANTINODES WITHIN MODEL HARBOUR BASIN

This program is nothing but an extension of RMS analysis program. It is written for example 4 mentioned in chapter 1. In this particular problem it is necessary to filter the data through band-pass digital filter in REAL-TIME and then find standard deviation of the filtered output.

Also here the program is divided into two parts :

1) Initialization part : This part interactively accepts the information from the user terminal, such as the name of the calibration file for the port, the data sampling rate, the display update rate, the analog I/O port number to which transducers are connected and the channel number of the first channel to which the first transducer is connected and specifications for the band-pass filter. part executes only once and it computes filter coefficients in double precision arithmetic and then rounds them to real number representation. is program then schedules ON-LINE analysis program RMSA9, along with passage of buffer which contains the filter coefficients. It then terminates itself.

c) ON-LINE analysis and display part : This part executes repetitively in real-time. In this ON-LINE analysis operation, both the band-pass filtering and the display of the RMS value is executed. Band-pass filter is realized as a cascade of low-pass and high-pass filter with following specifications :

Low-pass filter

i) pass-band ripple +/- 0.05 db

ii) stop-band attenuation > = 40 dbs

iii) pass-band edge 2 HZ

iv) stop-band edge 3 HZ

High-pass filter

i) pass-band ripple +/- 0.05 db

ii) stop-band attenuation > = 40 dbs

iii) pass-band edge 0.6 HZ

iv) stop-band edge 0.4 HZ

Sampling frequency which equals data sampling rate is 10 HZ

Gain versus frequency response of this recursive band-pass filter is shown in Fig 9 . RMS computation

procedure is similar to the one used in RMS analysis program in chapter 8.

CHAPTER 10

A PROGRAM FOR ON-LINE ESTIMATION OF VARIANCE SPECTRAL DENSITY

This program was developed to estimate the running variance spectral density of random wave data. It is useful for any experiment where it is necessary to check whether the generated waves have desired spectral shape and desired peak frequency.

Programming considerations

The program computes in real time the auto-covariance function by the recursive formula :

Y(I,J) = ALPHA*Y(I-1,J) + (1.-ALPHA)*X(I)*X(I-J)

for J = 0 to M

I = Ith data sample

M = number of time shifts for which auto-covariance function is to be evaluated.

The length M depends upon two factors

1) This is decided on the basis of desired resolution for

Wc = 1/M*T

2) It must meet requirement that the above computations of cross products and summations can be completed for M shifts in a time less than the sampling interval T.

It is evident that in order to compute the auto-covariance function for M lags it is necessary to store the last M data samples in a memory buffer. This data structure may be a push down stack or a ring buffer.The ring buffer data structure was selected because it is more efficient in execution time than the push down stack. The cross products involved in the M lags may be executed either by a DO LOOP or by making judicious use of the Vector Instruction Set (provided on the F-series machine) for additional improvements in execution time.

Program input

The program expects the following input parameters from the user terminal in an interactive mode.

1) Data sampling rate

2) Display update rate

3) Analog I/O port number to which the water level transducers are connected.

4) Analog channel number of the transducer for which the estimation of the variance spectral density is to be computed

5) Name of calibration file for the analog I/O port

The program is divided into two parts :

A) INITIALIZATION - This runs only once when the program is invoked for the first time and accepts input for the above parameters. It also schedules itself repetitively depending on data sampling rate.

B) ON-LINE ANALYSIS PART - Depending on data sampling rate this part is scheduled by the system repetitively. It computes auto-covariance function in REAL-TIME. When it is time to display the power spectrum, it schedules the SON program DISP4 without wait , along with the passage of the auto-covariance buffer and other parameters. The Son program, after getting scheduled by the father program, retrieves the parameters and the auto-covariance buffer by the string passage EXEC call and then multiplies the auto-covariance function by a suitable smoothing window. It then executes a Fourier transform of the windowed auto-covariance and displays the spectrum on the graphic screen of the CRT terminal HP 2648A. It should be noted that the DISP4 program runs at a lower priority than the father program and runs during the unused CPU time when other more time critical programs are not running. It is necessary to lock both programs to memory so that they can be scheduled at the right time. The self-termination feature

with saving of resources retains previous computations intact.

This program was tested during an experiment when regular waves were being generated. Results from the program were encouraging, however still more exhaustive testing with random wave data is necessary in order to evaluate the performance of the proposed procedure for the estimation of running spectral densities. Because of inherent delay involved in estimating the value of auto-covariance function it is to be expected that to get correct estimate of power spectrum certain time interval, which depends upon recursive filter characteristic must be allowed to elapse after starting the program.

Detailed listings of the programs PDSA1 and DISP4 and graphical representation of the estimated power spectrum with regular waves are given in Appendix VI.

CHAPTER 11

CONCLUSIONS

The conclusions are based on the results of several programs written for implementation of ON-LINE analysis procedures, including ON-LINE digital filtering.

1) Recursive filter program implementations execute faster than non-recursive filter implementations almost by a factor of two. They also require less memory storage for retention of previous data samples and filter coefficients. They are therefore recommended for ON-LINE digital filtering.

2) Because of the availability of the microcoded VIS library routines available on Hewlett-Packard F-series computers and because of the availability of large physical memory and the dynamic memory mapping system nonrecursive filter implementations can be given second thought because they offer some specific advantage over recursive designs because of their linear phase characheristics and inherent stability.

3) The Ring buffer data structure is superior to the push down stack data structure for any number of data samples in the buffer. Therefore the Ring buffer is recommended for any ON-LINE analysis implementa-

tion where it is necessary to retain a number of previous data samples.

4) First order recursive filtering for estimating the running average and the RMS value of data has the definite advantage of simplicity, fast execution memory economy. However when a signal is a mixture of number of low and high frequency components, then the estimated value fluctuates to a certain extent and it is somewhat difficult for the experimenter to make use of displayed value. In short, selection of this method depends upon spectral properties of input signal.

5) To overcome the problem of excessive variability of the estimate, a non-recursive method of estimating the running average and the RMS value was tried. Its performance for running RMS estimation was observed both during an actual experiment as well as for a simulated input signal. In case of simulated signal the displayed value was observed to fluctuate by +/- 5% whereas it was observed to be +/- 12 to 15 % for the recursive method. The requirement for large memory storage has been partly reduced by an alternative method whose analysis and program implementation is given in chapter 6. Depending upon one's need either of the methods can be selected.

6) While computing variance spectral density estimates for tests carried out with regular waves, the results were encouraging. However for random waves more experimental trials are needed to evaluate the performance of the procedure.

7) HP-1000 F-series machines can better meet the more complex ON-LINE data analysis tasks than the E-series machines because of the latter's fast specially microcoded VIS instruction set and its hardware floating point processor.

REFERENCES

1)'The measurement of power spectra from point of view of communications engineering'

By R. B. Blackman and J. W. Tu key

Dover New York 1958

2) 'Measurement and analysis of random data

By Julius S. Bendat and Allan G. Piersol

John Wi ley and Sons Inc New York 1966

3) 'Spectral analysis and its applications'

By Gwilym M. Jenkins and Donald G Watts

Holden-Day series in time series analysis

San-Francisco 1969

4) ' Digital signal processing and time series analysis

By Enders A. Robinson and Manuel T. Silvia

Holden-Day Inc San-Francisco 1978

5) 'Digital filters analysis and design'

By Andreas Antiniou

McGraw-Hill Book Company New York 1979

6)'Digital to Analog converter with straight line smoothing'

By E. R. Funke, N. L. Crookshank

and A. W. Wiegert

LTR-HY-80 Feb 1981 NRC Ottawa

7) 'An introduction to the analysis of random processes'

By E. R. Funke

Division of Mechanical engineering DME/NAE Quarterly bulletin No. 1966(4) NRC Ottawa

8) 'A digital program for on-line correlation and power spectral density analysis'

By E. R. Funke, F. T. Stock and L. Groves

Division of Mechanical engineering NRC Jan 1968 MK-23

9) 'An Introduction to GEDAP ' Hydraulics Laboratory.

By E. R. Funke, N. L. Crookshank and M. Wingham

NRC Ottawa 1980

10) Hewlett-Packard Journal Sep 1978

11) 'Digital Signal Processing'

Edited by Lawrence R. Rabiner and Charles M. Rader

IEEE Press New York 1972

12) 'VIS HP 12824A Vector Instruction set User's manual'

Hewlett-Packard Company Data Systems Division

Cupertino California 95014 June 1979

13)' HP 92840A Graphics plotting software user's manual April 1980

14)'Graphics/1000 -II 92841A Device independent Graphics Library

Programmer's reference manual June 1981

15) 'HP 92835A SIGNAL/1000 Digital Signal Processing

Library

User's manual Nov 1980

16) ' HP 92068A RTE IVB Programmer's reference manual

Jan 1980

17) 'HP 92068A RTE IVB Terminal User's Reference Manual Feb 1980

18)'NRC Hydraulics Lab GPLOT
General plotting package user guide

19)'HP 2648A CRT Graphics terminal user's manual

20) 'Programs for Digital signal processing

Edited by Digital Signal Processing Committee

IEEE Acoustic,Speech, and Signal processing society

IEEE Press New York 1979

21) 'Signal processing theories and applications'

Edited by M. Kunt and F. de Coulon

North-Holland publishing company Amsterdam New york 1980

APPENDIX I

```
C
FTN4
      PROGRAM RLPDF(),VERSION 1.0 - 25 MAY  1982
C
C************************************************************************
C**************+++++++++++++++++++++++++++++++****************************
C**************+                              +***************************
C**************+       PROGRAM RLPDF          +***************************
C**************+                              +***************************
C**************+++++++++++++++++++++++++++++++****************************
C************************************************************************
C                                                                       *
C      PROGRAMMED IN MAY 1982 BY                                        *
C      K.C.Kelkar                                                       *
C      Hydraulics Laboratory                                            *
C      National Research Council                                        *
C      Ottawa  K1A 0R6                                                  *
C                                                                       *
C*                                                                      *
C*----------------------------------------------------------------------*
C*     PROGRAM SUMMARY                                                  *
C*----------------------------------------------------------------------*
C*                                                                      *
C*     Gain versus frequency response of recursive low-pass digital     *
C*     filter.                                                          *
C*                                                                      *
C*----------------------------------------------------------------------*
C*                                                                      *
C*                                                                      *
C*----------------------------------------------------------------------*
C*     PROGRAM DESCRIPTION                                              *
C*----------------------------------------------------------------------*
C*                                                                      *
C*    1 - Generate simulated signal for filter input.                   *
C*    2 - Filter the signal.                                            *
C*    3 - When number of data samples equals NDATA compute RMS of       *
C*        input and output signal of the filter and compute GAIN        *
C*    4 - Increment frequency of simulated data signal by DELF          *
C*      - continue with step 1 until all frequency band is covered      *
C*    5 - Write GAIN versus FREQ response in output data GEDAP file     *
C*    6 - End.                                                          *
C*                                                                      *
C*----------------------------------------------------------------------*
C*     CONVERSATIONAL INPUT(INPUT UNIT=LUC):                            *
```

```
C*-------------------------------------------------------------------*
C*                                                                   *
C*     NDATA - Number of data samples to be generated                *
C*     NOFRE - Number of frequency components for response computation*
C*     DELF  - Incremental change in frequeny in response computation*
C*     NOCYC - The number of program cycles                          *
C*     IN1   - The name of the input GEDAP data file                 *
C*     IOUT1 - The name of the output GEDAP data file                *
C*                                                                   *
C*-------------------------------------------------------------------*
C*     GEDAP TRANSFER FILE EXAMPLE                                   *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*     FORMATTED OUTPUT(OUTPUT UNIT=LUL):                            *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*     LOG FILE OUTPUT(OUTPUT UNIT=LOG):                             *
C*-------------------------------------------------------------------*
C*                                                                   *
C*     IPSID  -The GEDAP program name                                *
C*     IN1    -The name of the input GEDAP data files                *
C*     IOUT1  -The name of the output GEDAP data files               *
C*                                                                   *
C*-------------------------------------------------------------------*
C*     GEDAP HEADER INPUT:                                           *
C*-------------------------------------------------------------------*
C*                                                                   *
C*                                                                   *
C*-------------------------------------------------------------------*
C*     GEDAP HEADER OUTPUT:                                          *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*     SUBROUTINES AND FUNCTIONS CALLED:                             *
C*-------------------------------------------------------------------*
C*                                                                   *
C*     Nil                                                           *
C*                                                                   *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*     PROGRAM LIMITATIONS:                                          *
```

```
C*-----------------------------------------------------------------------*
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    ERROR CODES:                                                       *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    LOADING INSTRUCTIONS                                               *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*    TR,LXXXXX                                                          *
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    REPORT ANY BUGS IN THIS SPACE                                      *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*************************************************************************
C
C
C
C
C
C*************************************************************************
C***                    Module #1                              **********
C***  System required GEDAP DIMENSION and COMMON statements     **********
C***                                                           **********
C*************************************************************************
C
```

```
      INTEGER JOBID(3),IPSID(3),ISMP(3)
      REAL DLI(10,3),DLO(10,2),DHI(10,3),DHO(10,2),DATA
      DOUBLE PRECISION AA0(10),BA0(10),BA1(10),AAT0(10),BAT0(10)
      DOUBLE PRECISION LAMDA,LAMDA1,CC(10),BAT1(10)
      DOUBLE PRECISION WDC,WD,LAMDA
      DOUBLE PRECISION K,K1,Q0,Q,D,AA,AP,XN,S0,S1,SIGMA,W,MU
      DOUBLE PRECISION T,WC,WA,WP,WI
      DOUBLE PRECISION A,B,C,E,OHM,V,AL0(10),AL1(10),BL0(10),BL1(10),HL0
      REAL AD0(10),AD1(10),BD0(10),BD1(10),AH1(2),AH0(2),BH0(2),BH1(2)
      REAL RESPNS(512),X(1024),Y(1024)
      COMMON/CFIX/IFIX(484),IN1(3),IOUT1(3),IOUTL(3)
      COMMON/IO/LUC,LUL,IPRINT,ISCI,ICRI,ISCO,ICRO
      COMMON/SYSIO/LOG,LUS,LU1,LU2,LU3
      COMMON/GDFMP/IDCBI(144),IDCBO(144),IBUF1(40),IPRAM(5),IRBUF(33)
C
C
C*****************************************************************************
C***              Module #2                                       **************
C***  System required GEDAP EQUIVALENCE statements                **************
C***                                                              **************
C*****************************************************************************
C
      EQUIVALENCE(IFIX(2),JOBID(1))
      EQUIVALENCE(IFIX(6),VERNO)
      EQUIVALENCE(IFIX(12),IFORM)
      EQUIVALENCE(IFIX(13),ITYPE1)
      EQUIVALENCE(IFIX(14),ITYPE2)
      EQUIVALENCE(IFIX(20),N)
      EQUIVALENCE(IFIX(25),DEL)
      EQUIVALENCE(IFIX(60),SCFI)
      EQUIVALENCE(IFIX(260),TR)
C
C
C*****************************************************************************
C***                    Module #3                                 **************
C***  System required GEDAP initialization statements             **************
C***                                                              **************
C*****************************************************************************
C
C*---------------------------------------------------------------------------*
C*    Insert program name and version number                                 *
C*---------------------------------------------------------------------------*
      DATA IPSID/2HFI,2HLT,2HR /
      DATA INDI,INDO,ICNTR,NOCYC/1,1,1,1/
```

```
C*-------------------------------------------------------------------*
C                                                                    *
C     Coefficients for recursive low pass filter with following specs: *
C     WP - 2HZ                                                       *
C     WA - 3HZ                                                       *
C     Maximum permissible ripple in the pass-band < = 0.5 db         *
C     Minimum attenuation in the stop-band > = 40 dbs                *
C     number of stages required 2 second order stages and 1 first    *
C     order stage                                                    *
C                                                                    *
C*-------------------------------------------------------------------*
C                                                                    *
C     DATA AL0/1.89072,1.26947/
C     DATA AL1/1.18568,-0.20059/
C     DATA BL0/0.56096,0.86717/
C     DATA BL1/-1.03479,-0.87236/
C     DATA HL0,AAL0,BBL0,CCL0/0.2277563,0.039914,0.039914,0.5965589/
C                                                                    *
C*-------------------------------------------------------------------*
C*    Coefficients for low-pass recursive filter with FC = 2.4494HZ  *
C*-------------------------------------------------------------------*
C     DATA AL0/2.43018,1.41841/
C     DATA AL1/2.65217,0.58318/
C     DATA BL0/0.50200,0.85431/
C     DATA BL1/-0.70619,-0.39933/
C     DATA HL0,AAL0,BBL0,CCL0/0.3036901,0.0373989,0.0373989,0.4959562/
C*-------------------------------------------------------------------*
C                                                                    *
C     Coefficients for recursive high-pass digital filter with       *
C     following specs :                                              *
C     FA - 0.4 HZ stop-band edge                                     *
C     FP - 0.6 HZ pass-band edge                                     *
C     FS - 10.0 HZ sampling frequency                                *
C     Pass-band ripple < = 0.5 db                                    *
C     Stop-band attenuation > = 40 dbs                               *
C     number of stages 2 second order stages and 1 first order stage *
C                                                                    *
C*-------------------------------------------------------------------*
      DATA AH0/0.8048190382,0.9528693144/
      DATA AH1/-1.5884070075,-1.8498437933/
      DATA BH0/0.6992349019,0.9434829226/
      DATA BH1/-1.4988101819,-1.8120994995/
      DATA HH0,AAH0,BBH0,CCH0/0.999999,0.6915462835,-0.6915462835,
     1 0.3030925671/
```

```
      CALL GETLU(IER)
      IF(IER.LT.0)GO TO 999
      VERNO=1.0
 1000 CONTINUE
      WRITE(LOG,994)IPSID,VERNO
  994 FORMAT(" GEDAP PROGRAM: ",3A2," VERSION:",F7.2)
C
C*-------------------------------------------------------------------*
C*    Obtain the number of program cycles                            *
C*-------------------------------------------------------------------*
      IF (IPRINT .EQ. 1) WRITE(LUC,995)
  995 FORMAT(" ENTER NUMBER OF PROGRAM CYCLES  _")
      READ(LUC,*) NOCYC
C
C*******************************************************************
C*****************++++++++++++++++++++++++++++*************************
C**************+                             +************************
C**************+     PROGRAM RLPFD           +************************
C**************+                             +************************
C*****************++++++++++++++++++++++++++++*************************
C*******************************************************************
C*                                                                 *
C*    PROGRAMMED IN MAY 1982 BY                                    *
C*    K.C.Kelkar                                                   *
C*    Hydraulics Laboratory                                        *
C*    National Research Council                                    *
C*    Ottawa  K1A OR6                                              *
C*                                                                 *
C*-------------------------------------------------------------------*
C*    PROGRAM SUMMARY                                              *
C*-------------------------------------------------------------------*
C*                                                                 *
C*    Design precedure for recursive low-pass digital filter       *
C*                                                                 *
C*-------------------------------------------------------------------*
C*    PROGRAM DESCRIPTION                                          *
C*-------------------------------------------------------------------*
C*                                                                 *
C*  1 - Design low-pass elliptic filter with normalized cutoff     *
C*      frequency of 1 radian/sec and pass-band edge FNP and stop-band *
C*      edge FNA  such that FNP =sqrt(RATIO) and FNA=1./ sqrt(RATIO)  *
C*      RATIO = FP/FA                                              *
C*  2 - Find WI taking into account warping effect due to Bilinear *
C*      transformation in order to realize desired FC=sqrt(FA*FP)  *
```

```
C*  3 - Find frequency scaling factor LAMDA1 and compute new
C*      filter coeffients in S domain
C*  4 - Transform S domain filter transfer function to Z domain
C*      using Bilinear transformation
C*  5 - End
C*
C*-------------------------------------------------------------
C*     PROGRAM INPUT
C*-------------------------------------------------------------
C*
C*   AP - Pass-band ripple in  dbs
C*   AA - Stop-band attenuation in dbs
C*   FP - Pass-band edge in HZ
C*   FA - Stop-band edge in HZ
C*   FS - Sampling frequency in HZ
C*
C*-------------------------------------------------------------
C*     PROGRAM OUTPUT
C*-------------------------------------------------------------
C*
C*    Second order filter coefficients AL0(i),AL1(i),BL0(i),
C*    BL1(i)  where i = 1 to N and N = number of second order stages
C*    and H0,AAL0,BBL0,CCL0 scaling factor and first order stage
C*    coefficients if applicable
C*
C*-------------------------------------------------------------
C*    REFERENCE
C*-------------------------------------------------------------
C*
C*     ANALYSIS AND DESIGN OF DIGITAL FILTERS
C*     AUTHOR ANDREAS ANTONIOU
C*     PUBLISHER McGraw-Hill
C*
C*-------------------------------------------------------------
C*-------------------------------------------------------------
C      Design formulae
C*-------------------------------------------------------------
C      WP = SQRT(K)
C      WA = 1/SQRT(K)
C      WC = SQRT(WA*WP)
C      Hn(S) = H0/D0(S)*     (S*S + AA0(I))/(S*S + BA1(I)*S + BA0(I))
C       r = (n-1)/2     FOR ODD n
C         = n/2         FOR EVEN n
C      D0(S) = S + SIGMA   FOR ODD n
```

```
C            = 1          FOR EVEN n
C       K1 = SQRT(1 - K*K)
C       Q0 = 0.5*(1 - SQRT(K1))/(1 + SQRT(K1))
C       Q = Q0  +  2*Q0**5  +  15*Q0**9  +  150*Q0**13
C       D = (10.0**(0.1*Aa) - 1)/(10.0**(0.1*Ap) - 1)
C       n ) = LOG(16*D)/LOG(1/Q)
C       LAMDA = (1/2*n)ln(10.0**(0.05*Ap$ + 1)/(10.0**(0.05*Ap) - 1)
C
C       SIGMA = 2*Q**0.25*      (-1)**m*Q**(m*(m + 1)*SINH((2*m + 1)*LAMDA)
C
C             ------------------------------------------------------
C
C              (1 + 2*       (-1)**m*Q**(m*m)*COSH(2*m*LAMDA)
C
C       W = SQRT((1 + KSIGMA**2)*(1 + SIGMA**2/K))
C
C       A = 2*Q**0.25*         (-1)**m*Q**(m*(m + 1))*SIN((2*m + 1)*PI*MU)/n
C
C       B = 1 + 2*          (-1)**m*Q**(m*m)*COS(2*m*PI*MU)/n
C
C       OHM = A/B
C       MU = i       FOR ODD n
C          = i - 0.5  FOR EVEN n      i = 1,2,.....r
C       Vi = SQRT((1 - K*OHM**2)*(1 - OHM**2/K)
C       AA0(i) = 1/OHM**2
C       BA0(i) = ((SIGMA*V(i))**2+(OHM(i)*W)**2)/(1+SIGMA**2*OHM(i)**2)**2
C       BA1(i) = 2*SIGMA*V(i)/(1 + (SIGMA*OHM(i))**2)
C       H0 = SIGMA*         BA0(i)/AA0(i)       FOR ODD n
C
C          = 10.0**(-0.05*Ap)*          BA0(i)/AA0(i)    FOR EVEN n
C
      PI = 3.14159
      PI2 = 8.*ATAN(1.)
C*-------------------------------------------------------------------*
C*                                                                   *
C*     Get specifications for the desired low pass digital filter    *
C*                                                                   *
C*-------------------------------------------------------------------*
      WRITE(1,351)
  351 FORMAT(5X,"ENTER PASS-BAND RIPPLE IN DBS ")
      READ(1,*) AP
      WRITE(1,352)
  352 FORMAT(5X,"ENTER ATTENUATION IN STOP-BAND IN DBS")
      READ(1,*) AA
```

```
      WRITE(1,302)
 302  FORMAT(5X,"ENTER PASS-BAND EDGE FREQUENCY IN HERTZ")
      READ(1,*) FP
      WP=PI2*FP
      WRITE(1,303)
 303  FORMAT(5X,"ENTER STOP-BAND EDGE FREQUENCY IN HERTZ")
      READ(1,*) FA
      WRITE(1,304)
 304  FORMAT(1X,"ENTER SAMPLING FREQUENCY IN HERTZ")
      READ(1,*)FS
      T = 1./FS
      WA = PI2*FA
      RATIO = DTAN(PI*FP*T)/DTAN(PI*FA*T)
      WNP = SQRT(RATIO)
      WNA = 1./WNP
C-----------------------------------------------------------------------*
C     Program to design low-pass elliptic filter in S domain            *
C-----------------------------------------------------------------------*
      K = RATIO
      K1 = DSQRT(1 - K*K)
      Q0 = 0.5*(1 - DSQRT(K1))/(1 + DSQRT(K1))
      Q = Q0 + 2*Q0**5 + 15*Q0**9  + 150*Q0**13
      D = ((10.0)**(0.1*AA) - 1)/((10.0)**(0.1*AP) - 1)
      XN = DLOGT(16*D)/DLOGT(1/Q)
      N = 0
 10   XN = XN - 1
      N = N + 1
      IF(XN.GT.0) GO TO 10
      IN = N
C-----------------------------------------------------------------------*
C     Calculation of LAMDA                                              *
C-----------------------------------------------------------------------*
      A = (10.0)**(0.05*AP) + 1
      B = (10.0)**(0.05*AP) - 1
      C = DLOG(A/B)
      XN = FLOAT(IN)
      LAMDA = C/(2*XN)
C-----------------------------------------------------------------------*
C    Calculation of SIGMA                                               *
C-----------------------------------------------------------------------*
      S = 0
      S1 = 0
      DO 20 M = 1,5
      A = (-1.0)**(M - 1)
```

```
      B = Q**((M - 1)*M)
      C = DEXP((2*M - 1)*LAMDA)
      D = DEXP(-1.0*(2*M - 1)*LAMDA)
      E = (C - D)/2.
      S = S + A*B*E
      A = (-1.0)**M
      B = Q**(M*M)
      C = DEXP(2*M*LAMDA)
      D = DEXP(-2.0*M*LAMDA)
      E = (C + D)/2.0
      S1 = S1 + A*B*E
   20 CONTINUE
      S = 2*Q**(0.25)*S
      S1 = 1 + 2*S1
      SIGMA = S/S1
      W = DSQRT((1 + K*SIGMA**2)*(1 + SIGMA**2/K))
      IFLAG = 0
      XN = IN
   30 IN = IN - 2
      IF(IN.GT.0) GO TO 30
      IF(IN.EQ.0) GO TO 50
      IFLAG = 1
   50 CONTINUE
      IN = N
      IP = MOD(IN,2)
      IF(IP.EQ.0) 700,710
  700 IK = IN/2
      GO TO 720
  710 IK = (IN - 1)/2
  720 DO 90 I = 1,IK
      S = 0
      S1 = 1
      IF(IP.EQ.0)740,730
  730 MU = I
      GO TO 750
  740 MU = I - 0.5
  750 DO 60 M = 1,5
      S = S+(-1)**(M - 1)*Q**(M*(M - 1))*DSIN((2*M - 1)*PI*MU/XN)
      S1 = S1+(-1)**M*Q**(M*M)*DCOS(2*M*PI*MU/XN)
   60 CONTINUE
      S = 2*(Q**0.25)*S
      S1 = 1 + 2*S1
      OHM = S/S1
      V = DSQRT((1 - K*OHM**2)*(1 - OHM*OHM/K))
```

```
      AA0(I) = 1/(OHM*OHM)
      BA0(I) = ((SIGMA*V)**2 + (OHM*W)**2)/(1 + (SIGMA*OHM)**2)**2
      BA1(I) = (2*SIGMA*V)/(1 + (SIGMA*OHM)**2)
 90   CONTINUE
      S = 1
      DO 100 I = 1,IK
      S = S*BA0(I)/AA0(I)
 100  CONTINUE
      IF(IP.EQ.0)800,810
 800  H0 = S*(10.0)**(-0.05*AP)
      GO TO 820
 810  H0 = SIGMA*S
C-----------------------------------------------------------------------*
C     To account for warping effect due to bilinear transformation      *
C     S domain frequency is prewarped by the formula                    *
C                          WI=2 T*TAN(WD*T 2)                           *
C-----------------------------------------------------------------------*
 820  WD = DSQRT(WP*WA)
      WI = (2.0/T)*TAN(WD*T/2.0)
      LAMDA1 = 1./WI
C-----------------------------------------------------------------------*
C       Reference page 131 low-pass to low-pass transformation          *
C-----------------------------------------------------------------------*
C     WNC = cutoff frequency of normalised filter .                     *
C     WC = 1.0
C-----------------------------------------------------------------------*
C      Effect on filter coeff due to frequency scaling                  *
C-----------------------------------------------------------------------*
      DO 11 I = 1,IK
      AAT0(I) = AA0(I)/(LAMDA1**2)
      BAT0(I) = BA0(I)/(LAMDA1**2)
      BAT1(I) = BA1(I)/LAMDA1
 11   CONTINUE
      HL0 = H0/LAMDA1
      SIGMA = SIGMA/LAMDA1
C-----------------------------------------------------------------------*
C      S domain to Z domain transformation using bilinear               *
C       transformation                                                  *
C       Formulae used from page 185 ANDREAS ANTONIOU                    *
C-----------------------------------------------------------------------*
      DO 21 I = 1,IK
      CC(I) = BAT0(I) + (2.0*BAT1(I))/T + 4.0/(T*T)
      AL0(I) = (AAT0(I) + 4.0/(T*T))/CC(I)
      AL1(I) = 2.0*(AAT0(I) - 4.0/(T*T))/CC(I)
```

```
      BL0(I) = (BAT0(I) - 2*BAT1(I)/T + 4/(T*T))/CC(I)
      BL1(I) = 2.0*(BAT0(I) - 4/(T*T))/CC(I)
   21 CONTINUE
      IF(IP.EQ.0)1001,1010
 1010 AAL0 = T/(2. + SIGMA*T)
      BBL0 = AAL0
      CCL0 = (2. - SIGMA*T)/(2. + SIGMA*T)
 1001 WRITE(1,39)  (AL0(I),AL1(I),BL0(I),BL1(I),I = 1,IK)
   39 FORMAT(1X,4(F10.7,X))
      WRITE(1,36)HL0
   36 FORMAT(1X,"HL0 = ",F10.7)
      IF(IP.EQ.0)1020,1030
 1030 WRITE(1,38)AAL0,BBL0,CCL0
   38 FORMAT(1X,"AAL0 BBL0 CCL0 = ",3(F10.7,X))
 1020 CONTINUE
      DO 3000 I = 1,IK
      AD0(I) = AL0(I)
      AD1(I) = AL1(I)
      BD0(I) = BL0(I)
      BD1(I) = BL1(I)
 3000 CONTINUE
      HD0 = HL0
      AAD0 = AAL0
      BBD0 = BBL0
      CCD0 = CCL0
C     Get pameter values for NDATA , NOFRE , DELF
C
      WRITE(1,2000)
 2000  FORMAT(5X,"ENTER NDATA")
      READ(1,*) NDATA
      WRITE(1,2010)
 2010 FORMAT(5X,"ENTER NUMBER OF FREQUENCY COMPONENTS FOR RESPONSE")
      READ(1,*)NOFRE
      WRITE(1,2020)
 2020 FORMAT(1X,"ENTER FREQUENCY STEP FOR RESPONSE COMPUTATION")
      READ(1,*) DELF
C      GET SAMPLING INTERVAL
      WRITE(1,2030)
 2030  FORMAT(1X,"ENTER SAMPLING INTERVAL IN SEC")
      READ(1,*)TP
C*    Initialize frequency counter to 1
C
      IK2 = 0
      PI2 = 8.*ATAN(1.)
```

```
C*     Initialize DI & DO buffers
C
 101   DO 110 I = 1,IK
          DO 120 J = 1,3
             DLI(I,J) = 0.0
 120         CONTINUE
 110         CONTINUE
       DO 130 I = 1,IK
          DO 140 J = 1,2
             DLO(I,J) = 0.0
 140         CONTINUE
 130         CONTINUE
       DO 111 I = 1,5
          DO 121 J = 1,3
             DHI(I,J) = 0.0
 121         CONTINUE
 111      CONTINUE
       DO 131 I = 1,5
          DO 141 J = 1,2
             DHO(I,J) = 0.0
 141         CONTINUE
 131      CONTINUE
C
C      Generate input signal sampled at 10 HZ
C
C      Initialize simulated time interval counter to 0
C
          IN = 0
 145      XH = DELF*FLOAT(IK2)
          IF(XH.EQ.0) 146,147
 147      XXN = IN
          IF(IK2.GE.90) 149,153
 149      DATA = SIN(PI2*XH*XXN*TP + PI2/4.)
          GO TO 148
 153      XN = IN
          DATA = SIN(PI2*XH*XXN*TP)
          GO TO 148
 146      DATA = 1.0
 148      X(IN + 1) = DATA
C
C       Filter the I/P signal through 2 and 1/2 stage LP filter
C       stage means second order
C
         DO 150 J = 1,IK
```

```
          DO 160 I = 3,2,-1
             DLI(J,I) = DLI(J,(I-1))
 160         CONTINUE
          DLI(J,1) = DATA
          DATA = AD0(J)*DLI(J,1) + AD1(J)*DLI(J,2) + AD0(J)*DLI(J,3)
          DATA = DATA - BD1(J)*DLO(J,1) - BD0(J)*DLO(J,2)
          DLO(J,2) = DLO(J,1)
          DLO(J,1) = DATA
 150      CONTINUE
        IF(IP.EQ.0) 5050,5060
5060   TL2 = TL1
       TL1 = DATA
       DATA = AAD0*TL1 + BBD0*TL2 + CCD0*TL3
       TL3 = DATA
5050   XY = HD0*DATA
      DO 151 IJ = 1,2
         DO 161 II = 3,2,-1
         DHI(IJ,II) = DHI(IJ,(II-1))
161      CONTINUE
      DHI(IJ,1) = XY
      DATA = AH0(IJ)*DHI(IJ,1) + AH1(IJ)*DHI(IJ,2) + AH0(IJ)*DHI(IJ,3)
      DATA = DATA - BH1(IJ)*DHO(IJ,1) - BH0(IJ)*DHO(IJ,2)
      DHO(IJ,2) = DHO(IJ,1)
      DHO(IJ,1) = DATA
151   CONTINUE
      TH2 = TH1
      TH1 = DATA
      DATA = AAH0*TH1 + BBH0*TH2 + CCH0*TH3
      TH3 = DATA
      IN = IN + 1
       Y(IN) = DATA
C      Y(IN) = XY
       IF(IN.NE.NDATA) 145,300
C                                                                      *
C*    Find RMS of input and output signal ,then compute frequency      *
C*    versus amplitude response for the filter.                        *
C                                                                      *
 300  SUM1 = 0.0
      SUM2 = 0.0
      DO 200 I = 1,NDATA
         SUM1 = SUM1 + X(I)*X(I)
         SUM2 = SUM2 + Y(I)*Y(I)
 200     CONTINUE
         WRITE(1,331)SUM1,SUM2
```

```
 331      FORMAT(1X,2(F10.5,X))
       SUM1 = SUM1/FLOAT(NDATA)
       SUM2 = SUM2/FLOAT(NDATA)
       AINPUT = SQRT(SUM1)
       AOUTPT = SQRT(SUM2)
       IF(IK2.EQ.100) 301,310
 301   WRITE(1,3001) AOUTPT,AINPUT
 3001  FORMAT(1X,"AOUT AINPUT ",2(F20.15,X))
 310   RESPNS(IK2 + 1) = 20.*ALOGT(AOUTPT/AINPUT)
       IK2 = IK2 + 1
       IF(IK2.NE.NOFRE) 101,210
C
C*************************************************************************
C***                        Module #8                                  ***
C***  Obtain GEDAP.output file name and create data file               ***
C***                                                                   ***
C*************************************************************************
C*-----------------------------------------------------------------------*
C*     Set IFORM,ITYPE1,ITYPE2,N                                         *
C*-----------------------------------------------------------------------*
C210   PI2 = 8.*ATAN(1.)
C        WRITE(1,7010)
C7010   FORMAT(1X,"ENTER ALPHA")
C        READ(1,*)ALPHA
C        BETA = 1. - ALPHA
C        WRITE(1,7011)
C7011   FORMAT(1X,"ENTER NO. OF SAMPLES IN THE BUFFER")
C        READ(1,*) NSAMP
C        XS = FLOAT(NSAMP)
C        WRITE(1,7012)
C7012   FORMAT(1X,"ENTER SAMPLING INTERVAL")
C        READ(1,*)TP
C        DO 7020 I = 2,100
C        SUM = 0.0
C        F =.0.05*FLOAT(I-1)
C        DO 7030 J = 1,NSAMP
C        X(J) = SIN(PI2*F*FLOAT(J)*TP)
C        SUM = SUM + X(J)
C7030   CONTINUE
C       J = 1
C        DO 7040 K10 = (NSAMP+1),(2*NSAMP)
C        XY = SIN(PI2*F*FLOAT(K10)*TP)
C        SUM = SUM + XY - X(J)
C        J = J + 1
```

```
C       Y(K10-NSAMP) = SUM/XS.
C7040   CONTINUE
C       SQR = 0.0
C       DO 7050 L = 1,NSAMP
C       SQR = SQR + Y(L)*Y(L)
C7050   CONTINUE
C       RESPNS(I) = 20.*ALOG(SQRT(SQR/XS))
C7020   CONTINUE
C       A1= 1. + ALPHA*ALPHA -2.*ALPHA*COS(PI2*F*0.05)
C       A1= SQRT((1.-COS(128.*PI2*F))**2+(SIN(128.*PI2*F))**2)
C       B1 = SQRT((1.-COS(PI2*F))**2+(SIN(PI2*F))**2)
C       RATIO=(1./XS)*SIN(0.5*XS*PI2*TP*F)/SIN(0.5*PI2*TP*F)
C       RESPNS(I) = 20.*ALOG(RATIO)
C7020   CONTINUE
C       RESPNS(1) = 1.0
 210   IFORM = 2
       ITYPE1 = 11
       ITYPE2 = 0
       SCFI = 1.0
       N = 100
       DEL = DELF
       CALL GDOUT(INDO,IER)
       IF(IER.LT.0)GO TO 999
C
C
C*************************************************************************
C***              Module #9                    ********************************
C***  Create and output GEDAP header           ********************************
C***     file then open data file              ********************************
C***                                           ********************************
C*************************************************************************
C
       DO 997 IPRGNM = 1,3
  997  JOBID(IPRGNM) = IPSID(IPRGNM)
       CALL HDROT(IFORM,ITYPE1,ITYPE2,IER)
       IF(IER.LT.0)GO TO 999
C
C
C*************************************************************************
C***         Module #10                        ********************************
C***  Output data to GEDAP data file           ********************************
C***                                           ********************************
C*************************************************************************
C-------------------------------------------------------------------------
```

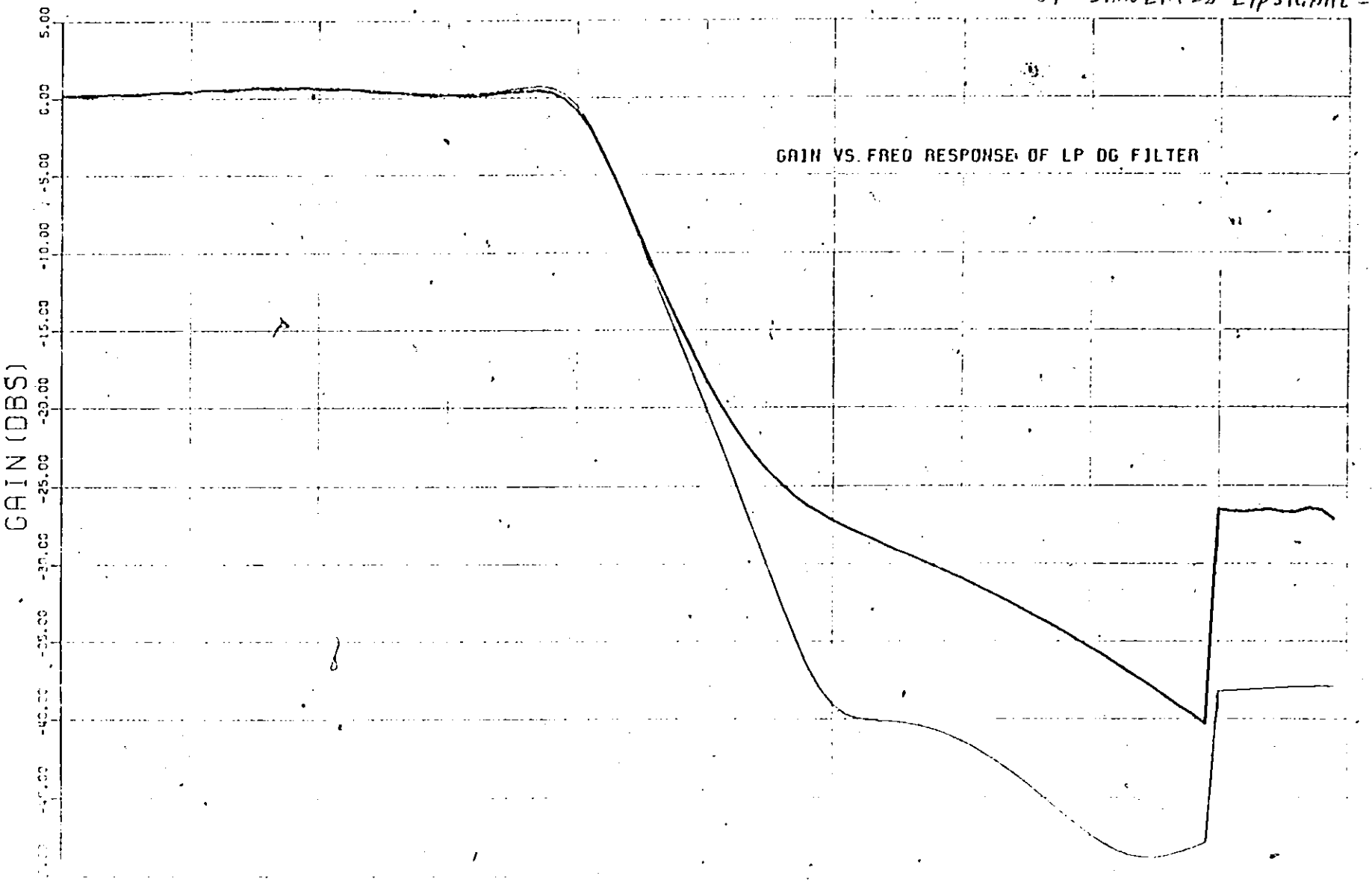
```
C
C     Write response into output GEDAP file                          *
C
      CALL WFOUT(RESPNS,200,IER)
      IF(IER.LT.0)GO TO 999
C
      WRITE(LOG,998)-IOUT1,ISCO,ICRO
  998 FORMAT(" OUTPUT FILE=",3A2,":",A2,":",I5)
C
      INDO = INDO+1
C*****************************************************
C****      Module #10                  ****************
C****    Output data to GEDAP data file ****************
C*****************************************************
C*-----------------------------------------------------------------*
C*   Check if the program is to be recycled                        *
C*-----------------------------------------------------------------*
      ICNTR = ICNTR + 1
      IF(ICNTR.LE.NOCYC) GO TO 1000
C
C
C*****************************************************************
C***              Module #11                  **********************
C***  System required GEDAP completion section **********************
C***                                          **********************
C*****************************************************************
C
  999 CALL GDCLL(IER)
      END
      END$
```

SPECIFICATIONS FOR THE FILTER

PASS BAND RIPPLE <= 0.5dB
STOP-BAND ATTENUATION >= 40dB
PASS-BAND EDGE 2 Hz
STOP BAND EDGE 3 Hz

THIN GRAPH - NO OF SAMPLES OF SIMULATED I/P SIGNAL = 2048

THICK SIGNAL - NO OF SAMPLES OF SIMULATED I/P SIGNAL = 128



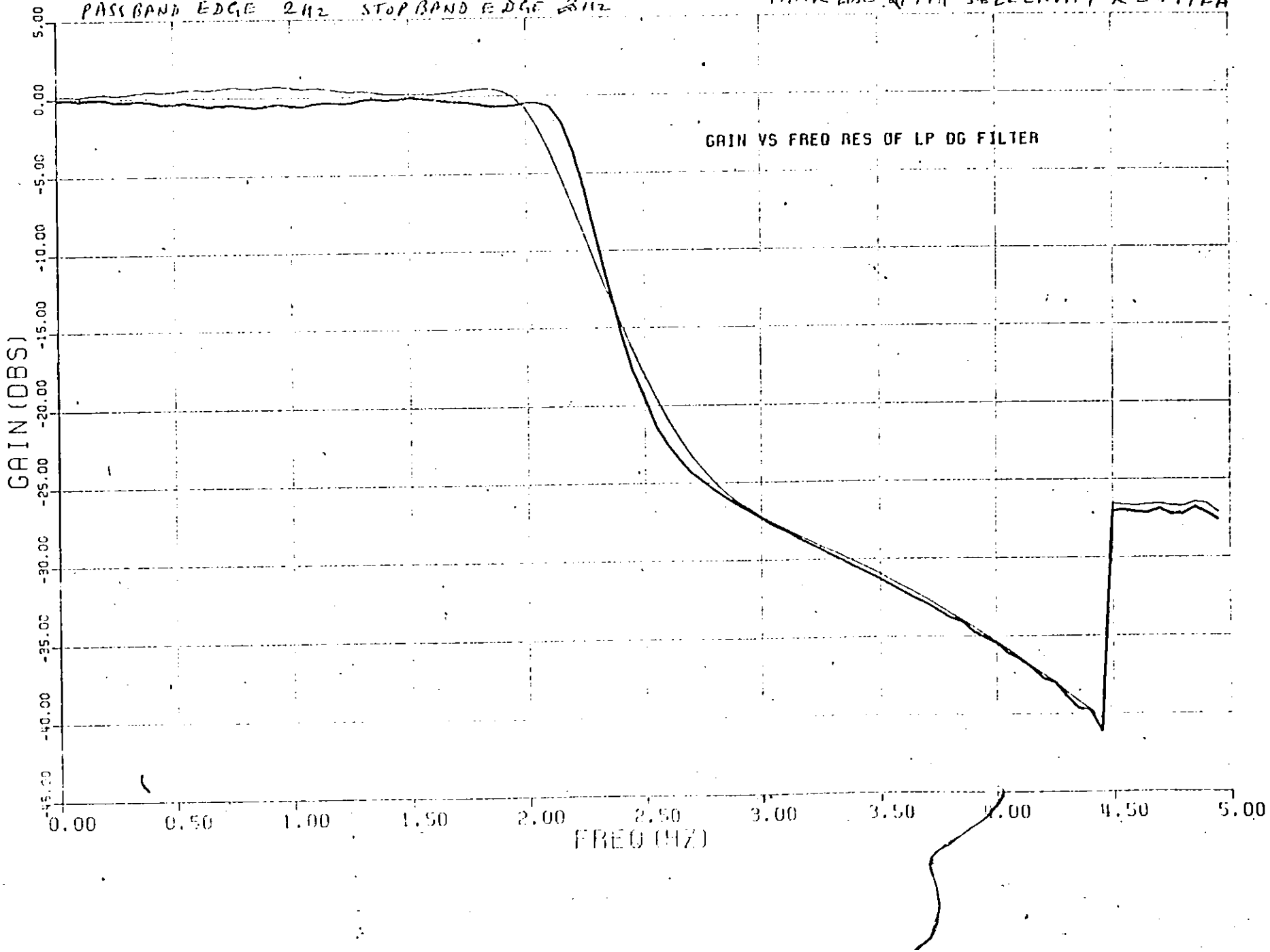
SPECIFICATIONS FOR THE FILTER

PASS BAND RIPPLE $\leq$ 0.5db STOP BAND ATTN > 40db

PASS BAND EDGE 2Hz STOP BAND EDGE 3Hz

THIN LINE GRAPH SELECTIVITY RATIO, $K = \frac{\text{DTAN}(\pi FP \cdot T)}{\text{DTAN}(\pi \cdot FA \cdot T)}$

THICK LINE GRAPH SELECTIVITY $K = FP/FA$



```
FTN4
      PROGRAM RHPFD(),VERSION 1.0 31 MAY 1982
C*****************************************************************
C*************++++++++++++++++++++++++++++++*************************
C**************+                            +************************
C**************+   PROGRAM RHPFD            +************************
C**************+                            +************************
C**************+                            +************************
C*************++++++++++++++++++++++++++++++++*************************
C*****************************************************************
C*---------------------------------------------------------------*
C*                                                               *
C*     PROGRAMMED IN MAY 1982 BY                                 *
C*     K.C.Kelkar                                                *
C*     Hydraulics Laboratory                                     *
C*     National Research Council                                 *
C*     Ottawa K1A 0R6                                            *
C*                                                               *
C*---------------------------------------------------------------*
C*---------------------------------------------------------------*
C                                                                 *
C     PROGRAM SUMMARY                                           *
C                                                               *
C*---------------------------------------------------------------*
C                                                                 *
C     Design  procedure for recursive high-pass digital filter  *
C                                                               *
C*---------------------------------------------------------------*
C                                                                 *
C     PROGRAM DESCRIPTION                                       *
C                                                               *
C*---------------------------------------------------------------*
C                                                                 *
C     1 - Design an eliptic low-pass filter with normalized cutoff *
C         frequency of 1 radian/sec and pass-band edge FNP and  *
C         stop-band edge FNA such that FNP = sqrt(RATIO) and    *
C         FNA = 1./sqrt(RATIO) and RATIO = FA/FP                *
C       --other specifications for the filter are :             *
C       - Pass-band ripple < = AP  db                           *
C       - Stop-band ripple > = AA dbs                           *
C       - Sampling frequency FS HZ                              *
C     2 - Find frequecy in time domain WI to account for warping *
C         effect due to Bilinear transformation in order to realize *
C         desired cutoff frequeny in digital filter FC = sqrt(FA*FP) *
```

```
C      3 - Do frequency scaling                                          *
C      4 - Fing Z-transform of filter transfer function using            *
C        - bilinear transformation.                                      *
C      5 - End                                                           *
C                                                                        *
C*-------------------------------------------------------------------*   *
C*      PROGRAM INPUT                                                *   *
C*-------------------------------------------------------------------*   *
C*                                                                   *   *
C*     AP - Pass-band ripple in dbs                                  *   *
C*     AA - Stop-band attenuation in dbs                             *   *
C*     FP - Pass-band edge in HZ                                     *   *
C*     FA - Stop-band edge in HZ                                     *   *
C*     FS - Sampling frequency in HZ                                 *   *
C*                                                                   *   *
C*-------------------------------------------------------------------*   *
C*      PROGRAM OUTPUT                                               *   *
C*-------------------------------------------------------------------*   *
C*                                                                   *   *
C*      Second order filter coefficients AH0(i),AH1(i),BH0(i),BH1(i) *   *
C*      where i = 1 to N and N = number of second order stages       *   *
C*      and HH0,AAH0,BBH0,CCH0 scaling factor and first order        *   *
C*      stage coefficients if applicable                             *   *
C*                                                                   *   *
C*-------------------------------------------------------------------*   *
C       REFFRENCE                                                    *   *
C*-------------------------------------------------------------------*   *
C*                                                                   *   *
C       Analysis and design of digital filters                       *   *
C       Author Andreas Antoniou                                      *   *
C       Publisher McGraw-Hill                                        *   *
C                                                                    *   *
C*-------------------------------------------------------------------*   *
      DOUBLE PRECISION K,K1,Q0,Q,D,AA,AP,XN,LAMDA,S0,S1,SIGMA,W,MU
      DOUBLE PRECISION AH0(2),BH0(2),BH1(2),AD0(2),AD1(2),BD0(2),BD1(2)
      DOUBLE PRECISION C(2),T,WC,WA,WP,WI
      DOUBLE PRECISION SIGH0,HH0,A,B,C,CF1,CF2,CF3,H0,AH1(2)
      DOUBLE PRECISION AA0(20),BA0(20),BA1(20),CHM,A,B,C,D,E,XN,XM
      DOUBLE PRECISION WA,WP,WC,A0(10),B0(10),B1(10),WI
      PI=3.14159
C*-------------------------------------------------------------------*   *
C*                                                                   *   *
C*      Get specifications for desired high-pass digital filter      *   *
C*                                                                   *   *
```

```
C*-------------------------------------------------------------------*
      WRITE(1,299)
 299  FORMAT(1X,"DESIGN OF HIGH-PASS RECURSIVE DIGITAL FILTER")
      WRITE(1,300)
 300  FORMAT(5X,"ENTER PASS-BAND RIPPLE IN DBS ")
      READ(1,*) AP
      WRITE(1,301)
 301  FORMAT(5X,"ENTER ATTENUATION IN STOP-BAND IN DBS")
      READ(1,*) AA
      WRITE(1,302)
 302  FORMAT(5X,"ENTER PASS-BAND EDGE FREQUENCY IN HZ")
      READ(1,*) FP
      WP=PI2*FP
      WRITE(1,303)
 303  FORMAT(5X,"ENTER STOP-BAND EDGE FREQUENCY IN HZ")
      READ(1,*) FA
      WA = PI2*FA
      WC = DSQRT(WA*WP)
      WRITE(1,304)
 304  FORMAT(5X,"ENTER SAMPLING FREQUENCY IN HZ")
      READ(1,*) FS
      T = 1./FS
C-------------------------------------------------------------------*
C     Program to design low-pass elliptic filter                    *
C-------------------------------------------------------------------*
      RATIO = DTAN(PI*FA*T)/DTAN(PI*FP*T)
      K = RATIO
      K1 = DSQRT(1 - K*K)
      Q0=0.5*(1 - DSQRT(K1))/(1 + DSQRT(K1))
      Q = Q0 + 2*Q0**5 + 15*Q0**9  + 150*Q0**13
      D = ((10.0)**(0.1*AA) - 1)/((10.0)**(0.1*AP) - 1)
      XN=DLOGT(16*D)/DLOGT(1/Q)
      N = 0
 10   XN = XN - 1
      N = N + 1
      IF(XN.GT.0) GO TO 10
      IN = N
C*-------------------------------------------------------------------*
C     Calculation of LAMDA                                          *
C*-------------------------------------------------------------------*
      A = (10.0)**(0.05*AP) + 1
      B = (10.0)**(0.05*AP) - 1
      C = DLOG(A/B)
      XN = FLOAT(IN)
```

```
      LAMDA = C/(2*XN)
C*-------------------------------------------------------------------*
C     Calculation of SIGMA                                           *
C*-------------------------------------------------------------------*
      S = 0
      S1 = 0
      DO 20 M = 1,5
      A = (-1.0)**(M - 1)
      B = Q**((M - 1)*M)
      C = DEXP((2*M - 1)*LAMDA)
      D = DEXP(-1.0*(2*M - 1)*LAMDA)
      E = (C - D)/2.
      S = S + A*B*E
      A = (-1.0)**M
      B = Q**(M*M)
      C = DEXP(2*M*LAMDA)
      D = DEXP(-2.0*M*LAMDA)
      E = (C + D)/2.0
      S1 = S1 + A*B*E
   20 CONTINUE
      S = 2*Q**(0.25)*S
      S1 = 1 + 2*S1
      SIGMA = S/S1
      W = DSQRT((1 + K*SIGMA**2)*(1 + SIGMA**2/K))
      IFLAG = 0
      XN = IN
   30 IN = IN - 2
      IF(IN.GT.0) GO TO 30
      IF(IN.EQ.0) GO TO 50
      IFLAG = 1
   50 CONTINUE
      IN = N
      IP = MOD(IN,2)
      IF(IP.EQ.0) 700,710
  700 IK = IN/2
      GO TO 720
  710 IK = (IN - 1)/2
  720 DO 90 I = 1,IK
      S = 0
      S1 = 0
      IF(IP.EQ.0)740,730
  730 MU = I
      GO TO 750
  740 MU = I - 0.5
```

```
 750  DO 60 M = 1,5
      S = S+(-1)**(M - 1)*Q**(M*(M - 1))*DSIN((2*M - 1)*PI*MU/XN)
      S1 = S1 + (-1)**M*Q**(M*M)*DCOS(2*M*PI*MU/XN)
  60  CONTINUE
      S = 2*(Q**0.25)*S
      S1 = 1 + 2*S1
      OHM = S/S1
      V = DSQRT((1 - K*OHM**2)*(1 - OHM*OHM/K))
      A0(I) = 1/(OHM*OHM)
      B0(I) = ((SIGMA*V)**2 + (OHM*W)**2)/(1 + (SIGMA*OHM)**2)**2
      B1(I) = (2*SIGMA*V)/(1 + (SIGMA*OHM)**2)
  90  CONTINUE
      S = 1
      DO 100 I = 1,IK
      S = S*B0(I)/A0(I)
 100  CONTINUE
      IF(IP.EQ.0)800,810
 800  H0 = S*(10.0)**(-0.05*AP)
      GO TO 820
 810  H0 = SIGMA*S
C*-------------------------------------------------------------------*
C                                                                    *
C     Find WI taking into consideration warping effect due to bilinear *
C     transformation                                                 *
C*-------------------------------------------------------------------*
 820  PI2 = 8.*ATAN(1.)
      WP =  FP*PI2
      WA =  FA*PI2
      WC = DSQRT(WA*WP)
      T = 1./FS
      WI = (2.0/T)*DTAN(WC*T/2.)
C*-------------------------------------------------------------------*
C                                                                    *
C     Frequency scaling                                              *
C                                                                    *
C*-------------------------------------------------------------------*
      LAMDA = 1.*WI
      DO 511 I = 1,2
      AH0(I) = LAMDA**2/A0(I)
      BH1(I) = B1(I)*LAMDA/B0(I)
      BH0(I) = LAMDA**2/B0(I)
 511  CONTINUE
      SIGH0 = LAMDA/SIGMA
      HH0 = (H0*A0(1)*A0(2))/(B0(1)*B0(2)*SIGMA)
```

```
C*-------------------------------------------------------------------------*
C                                                                         *
C     Bilinear transformation                                             *
C                                                                         *
C*-------------------------------------------------------------------------*
      DO 600 I = 1,2
         C(I) = BH0(I) + (2.*BH1(I))/T + 4./(T*T)
         AD0(I) = (AH0(I) + 4./(T*T))/C(I)
         AD1(I) = 2.*(AH0(I) - 4./(T*T))/C(I)
         BD0(I) = (BH0(I) - 2.*BH1(I)/T + 4./(T*T))/C(I)
         BD1(I) = 2.*(BH0(I) - 4./(T*T))/C(I)
 600     CONTINUE
      A = SIGH0*T - 2.0
      B = SIGH0*T + 2.0
      CF1 = 2./B
      CF2 = -CF1
      CF3 = -(A/B)
      DO 621 I = 1,2
      AH0(I) = AD0(I)
      AH1(I) = AD1(I)
      BH0(I) = BD0(I)
      BH1(I) = BD1(I)
 621  CONTINUE
      AAH0 = CF1
      BBH0 = CF2
      CCH0 = CF3
      DO 620 I = 1,2
      WRITE(1,4000) AH0(I),AH1(I),BH0(I),BH1(I)
4000  FORMAT(1X,4(F15.10,X))
 620  CONTINUE
      WRITE(1,4000) HH0,AAH0,BBH0,CCH0
      END
      END$
```

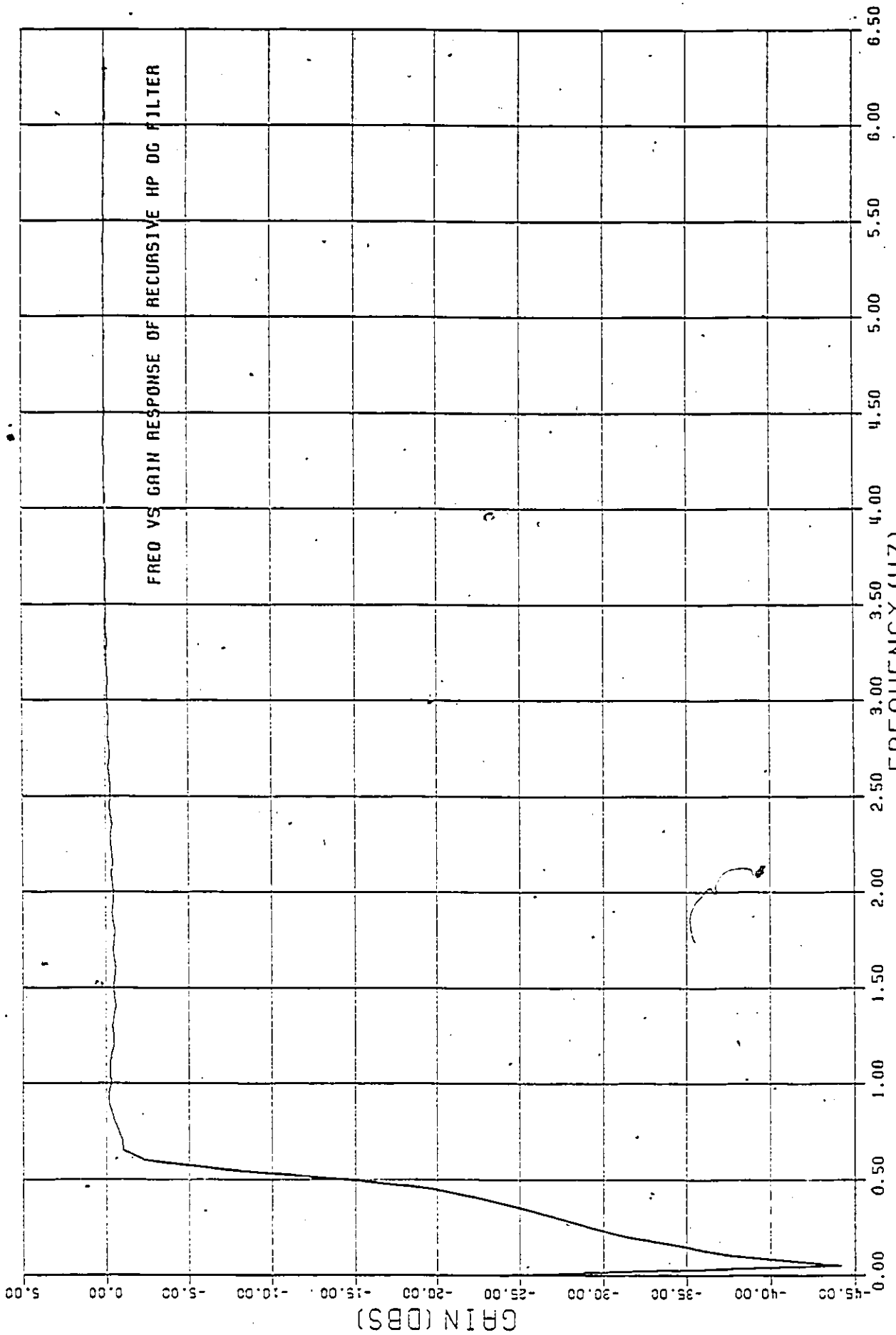

FREQ VS GAIN RESPONSE OF RECURSIVE HP DC FILTER
GAIN (DBS)
5.00
0.00
-5.00
-10.00
-15.00
-20.00
-25.00
-30.00
-35.00
-40.00
-45.00
FREQUENCY (HZ)
0.00
0.50
1.00
1.50
2.00
2.50
3.00
3.50
4.00
4.50
5.00
5.50
6.00
6.50

FIG 8

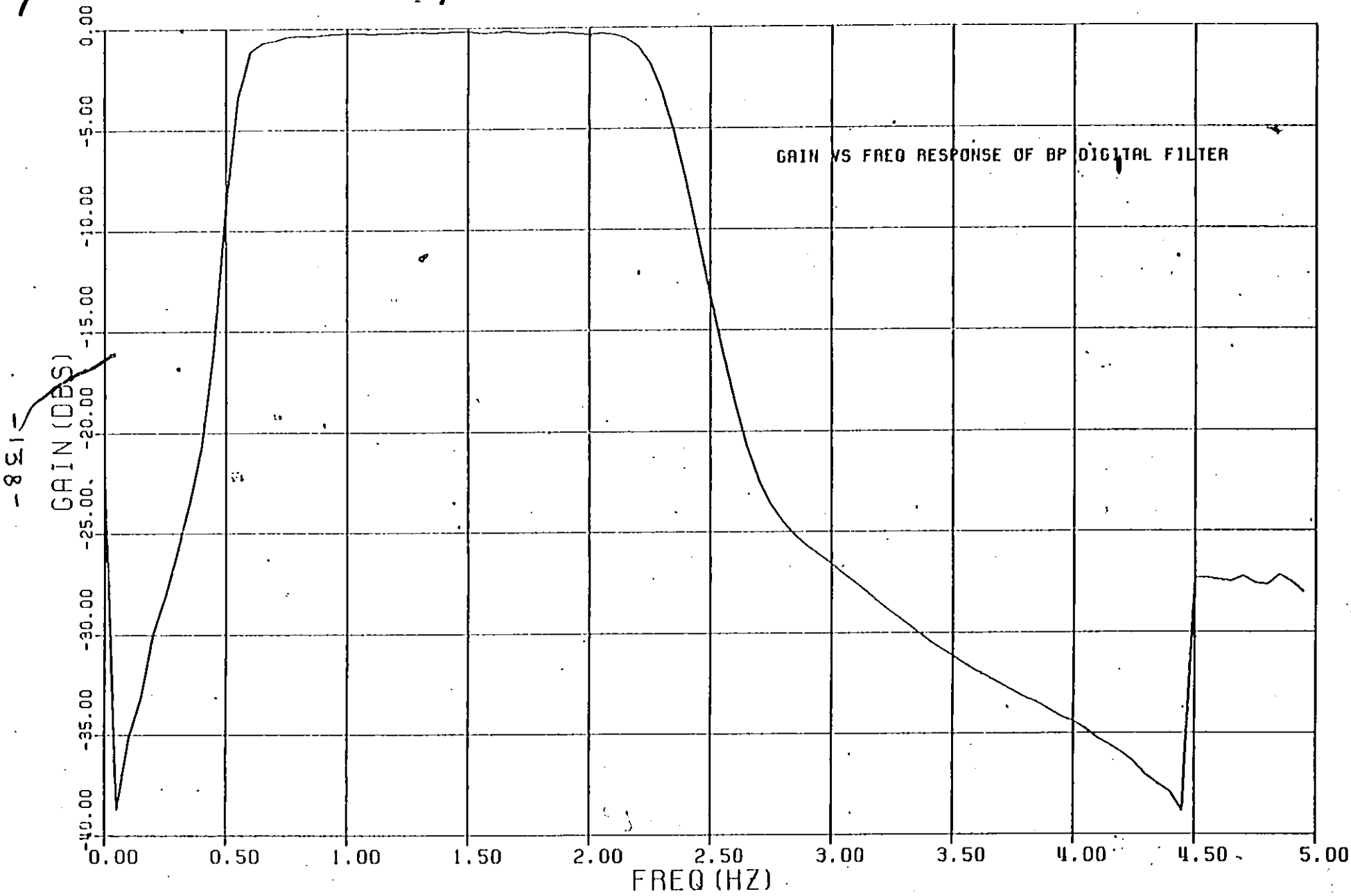

GAIN VS FREQ RESPONSE OF BP DIGITAL FILTER
GAIN (DBS)
0.00
-5.00
-10.00
-15.00
-20.00
-25.00
-30.00
-35.00
-40.00
0.00
0.50
1.00
1.50
2.00
2.50
3.00
3.50
4.00
4.50
5.00
FREQ (HZ)

FIG 9

```
FTN4
      PROGRAM NRFLP(),VERSION 1.0 - 22 MAY 1982
C
C*****************************************************************************
C*****************+++++++++++++++++++++++++++++++*****************************
C***************+                                :*****************************
C***************+    PROGRAM NRFLP               +*****************************
C***************+                                :*****************************
C*****************+++++++++++++++++++++++++++++++*****************************
C*****************************************************************************
C                                                                             *
C     PROGRAMMED IN MAY 1982 BY                                               *
C        K. C. Kelkar                                                         *
C     Hydraulics Laboratory                                                   *
C     National Research Council                                               *
C     Ottawa  K1A  0R6                                                        *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C     PROGRAM DESCRIPTION                                                     *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C    Computes coefficients for nonrecursive hlow-pass digital filter          *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C     PROGRAM DESCRIPTION                                                     *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C    1) Determine h(nT) using the fourier-series approach assuming            *
C       on idealized frequency response.                                      *
C       H(exp(j*w*T)  = 1  for mod(w) ( = WC                                  *
C                     = 0  for WC (   mod(w) ( = WS/2                         *
C       WC = 0.5*(WP + WA)                                                    *
C    2) Choose DELTA in eqation 1 and 2 such that AP ( = AP',                 *
C       and AA ) = AA', A suitable value is                                   *
C       DELTA = min(DELTA1 , DELTA2)                                          *
C       where DELTA1 = 10**(-0.05AA') and DELTA2 = A/B                        *
C       A = 10**(0.05*AP')-1 and B = 10**(0.05*AP') + 1                       *
C       AP = 20*log((1 + DELTA)/(1 - DELTA))          equation 1              *
C       AA = -20*log(DELTA)                           equation 2              *
C    3) Calculate AA using equation 2                                         *
```

```
C     4) Choose parameter ALPHA as follows:                                  *
C        ALPHA = 0                                          for AA ( 21    *
C              = 0.5842*(AA - 21)**0.4 + 0.07886*(AA - 21)  for 21(AA(=50*
C              = 0.1102*(AA - 8.7)                          for AA ) 50    *
C     5) Choose parameter D as follows :                                     *
C        D = 0.9222                                         for AA(= 21    *
C          = (AA - 7.95)/14.36                              for AA ) 21    *
C        Then select the lowest odd value of N satisfying the inequality*
C            N ) = WS*D/BT + 1                                             *
C        where transition bandwidth BT = WP - WA                           *
C     6) Form Wk(nT) using equation 3                                        *
C        Wk(nT) = I0(BETA)/I0(ALPHA) for mod(n) ( = (N - 1)/2 equation 3*
C               = 0   otherwise                                            *
C        where ALPHA is an independent parameter and                       *
C        BETA = ALPHA*sqrt(1 - (2*n/(N - 1))**2))                          *
C        Where I0(x) is zeroth-order Bessel function of the first kind     *
C        This can be evaluated to any desired degree of accuracy by        *
C        using the rapidly converging series                               *
C        I0(x) = 1 + SIGMA((1/k!)*(x/2)**k)**2                             *
C     7) Form H'w(z) = z**(-(N - 1)/2*Hw(z) where                           *
C        Hw(z) = Z(Wk(nT)*h(nT))                                           *
C        The above expressions for ALPHA and D are emperical relations     *
C        developed by Kaiser (REF 1)                                       *
C        REF 1  J.F. Kaiser ,Nonrecursive Digital Filter Design Using      *
C        the I0-sinh Window Functions, Proc 1974 IEEE Int Symp.            *
C        Circuit Theory, pp 20-23                                          *
C                                                                          *
C*-------------------------------------------------------------------------*
C                                                                          *
C     CONVERSATIONAL INPUT                                                 *
C                                                                          *
C*-------------------------------------------------------------------------*
C     FS - Sampling frequency in hertz                                     *
C     FP - Pass-band edge in  low-pass filter in hertz                     *
C     FA - Attenuation band edge in  low-pass filter in hertz              *
C     AP - Maximum permissible ripple in the pass-band in dbs              *
C     AA - Minimum attenuation in the stop-band in dbs                     *
C                                                                          *
C*-------------------------------------------------------------------------*
C                                                                          *
C     DIAGRAM INDICATING FILTER CHARACTERISTICS                            *
C                                                                          *
C*-------------------------------------------------------------------------*
C                                                                          *
```

```
C
C
C
C
C
C
C
C
C
C
C
C
C*-------------------------------------------------------------------*
C
C     PROGRAM OUTPUT
C
C*-------------------------------------------------------------------*
C
C     Minimum number of stages required to realize filter with given
C     specifications:
C     Coefficients for the filter
C
C*-------------------------------------------------------------------*
      REAL WP,WC,WA,WS,H(30),W(30),WH(100)
C     WP - Passband edge
C     WC - Mid-frequency between WP and WA
C     WA - Attenuation band edge
C     WS - Sampling frequency
C-------------------------------------------------------------------*
C
C     Get specifications for the filter
C
C-------------------------------------------------------------------*
      WRITE(1,7090)
 7090 FORMAT(1X,"ENTER SAMPLING FREQUENCY IN HZ")
      READ(1,*) FS
      WRITE(1,8000)
 8000 FORMAT(1X,"ENTER PASS-BAND EDGE IN HZ")
      READ(1,*) FP
      WRITE(1,8010)
 8010 FORMAT(1X,"ENTER STOP-BAND EDGE IN HZ")
      READ(1,*) FA
      WRITE(1,8020)
 8020 FORMAT(1X,"ENTER RIPPLE IN DBS IN THE PASS-BAND")
```

```
      READ(1,*) AP
      WRITE(1,8030)
 8030 FORMAT(1X,"ENTER MINIMUM ATTENUATION IN THE STOP-BAND IN DBS")
      READ(1,*) AA
      PII = 4.*ATAN(1.)
      PI2 = 2.*PII
      WS  = PI2*FS
      WP  = PI2*FP
      WA  = PI2*FA
      WC  = 0.5*(WP + WA)
      WRITE(1,8050) WS,WP,WA,WC
 8050 FORMAT(1X,"WS WP WA WC",4F5.2)
C                                                                      *
C*    DELTA = MIN(DELTA1,DELTA2)                                       *
C                                                                      *
      DELTA1 = 10.**(-0.05*AA)
      DELTA2 = (10.**(0.05*AP) - 1)/(10.**(0.05*AP) + 1)
      WRITE(1,8060) DELTA1,DELTA2
 8060 FORMAT(1X,"DELTA1 DELTA2",2F8.4)
      IF(DELTA1.LE.DELTA2) 200,210
  200 DELTA = DELTA1
      GO TO 220
  210 DELTA = DELTA2
  220 CONTINUE
      WRITE(1,8070) DELTA
 8070 FORMAT(1X,"DELTA ",F8.4)
C                                                                      *
C*    Calculate AA using equation AA = -20*log(DELTA)                  *
C                                                                      *
      AA = -20.*ALOGT(DELTA)
      WRITE(1,9000) AA
 9000 FORMAT(1X,"AA ",F8.4)
C                                                                      *
C*    Choose parameter ALPHA, as follows:                              *
C*    ALPHA = 0 for AA < = 21                                          *
C*          = 0.5842*(AA-21)**0.4 + 0.07886*(AA-21) for 21< AA < = 50  *
C*          = 0.1102*(AA - 8.7)   for AA > 50                          *
C                                                                      *
      IF(AA.LE.21) 300,310
  300 ALPHA = 0.0
      GO TO 350
  310 IF(AA.LE.50.AND.AA.GT.21) 100,110
  100 ALPHA = 0.5842*(AA-21.)**0.4 + 0.07886*(AA - 21.)
      GO TO 350
```

```
 110  ALPHA = 0.1102*(AA - 8.7)
 350  WRITE(1,9010) ALPHA
 9010 FORMAT(1X,"ALPHA ",F8.4)
C                                                                      *
C*    Choose parameter D as follows:                                   *
C*    D = 0.9222   for AA ( = 21                                       *
C*      = (AA - 7.95)/14.36  for AA ) 21                               *
C                                                                      *
      IF(AA.LE.21) 400,410
 400  D = 0.9222
      GO TO 450
 410  D = (AA - 7.95)/14.36
 450  WRITE(1,9020) D
 9020 FORMAT(1X,"D ",F8.4)
C                                                                      *
C*    Select the lowest odd value of N satisfying the inequality       *
C*    N ) = WS*D/BT + 1                                                *
C*    where BT = WA - WP                                               *
C                                                                      *
      BT = WA - WP
      XN = WS*D/BT + 1
      WRITE(1,9005) XN
 9005 FORMAT(1X,"XN ",F8.4)
      N  = INT(XN)
      IF(N.LT.XN) N = N + 1
      WRITE(1,9030) N
 9030 FORMAT(1X,"N ",I3)
C     WRITE(1,9011)
      IP = MOD(N,2)
      IF(IP.EQ.0) 510,500
 510  N = N + 1
C     WRITE(1,9031) IP
C9031 FORMAT(1X,"IP = ",I2)
C9011 FORMAT(1X,"ENTER NEAREST ODD N")
C     READ(1,*)N
 500  WRITE(1,9032) N
 9032 FORMAT(1X,"NEAREST ODD N = ", I3)
C                                                                      *
C*    COMPUTE h(nT) by the formula                                     *
C*    h(nT) = 1./n*PII*sin(WC*n*T)                                     *
C*    where WC = 0.5(WP + WA)                                          *
C*          T  = PI2/WS                                                *
C                                                                      *
      T = PI2/WS
```

```
      IN = N/2 + 1
      WRITE(1,9040) IN
 9040 FORMAT(1X,"IN ",I3)
      H(1) = (1./PII)*(WC*T)
      DO 130 I = 2,IN
         XI = I
         XI = XI - 1.0
         H(I) = (1./(XI*PII))*(SIN(WC*XI*T))
  130    CONTINUE
         WRITE(1,9080) (H(I),I=1,IN)
  9080   FORMAT(F10.8)
C-------------------------------------------------------------------------*
C                                                                         *
C     Compute window function by following formula                        *
C     Wk(nT) = I0(B)/I0(ALPHA)   for mod(n) < = (N-1)/2                   *
C            = 0    otherwise                                             *
C     ALPHA is an independent parameter                                   *
C     B  = ALPHA*SQRT(1 - (2.n/(N-1))**2)                                 *
C     and  I0(X) = 1 + SIGMA((1/k!)*(X/2)**k)**2   k= 1,infinity          *
C     in practical case series converges rapidly after k >10              *
C                                                                         *
C     Compute I0(ALPHA)                                                   *
C                                                                         *
      SUM = 0.0
      DO 150 K = 1, 10
         IJ = K
         XK = K
         FACT = 1.
         DO 160 J = 1,IJ
            XJ = J
            FACT = FACT*XJ
  160       CONTINUE
         PRODCT = ((1./FACT)*(ALPHA/2.)**XK)**2
         SUM = SUM + PRODCT
  150    CONTINUE
      ALF0  = SUM +1.0
C                                                                         *
C     Compute I0(B) and then Wk(nT)                                       *
C                                                                         *
      XN = N - 1
      DO 170 I = 1,IN
         IP = I - 1
         XP = IP
         B = ALPHA*SQRT(1- (2.*XP/XN)**2)
```

```
          SUM = 0.0
          DO 180 K = 1,10
             IJ = K
             FACT = 1.0
             XK = K
             DO 190 J = 1,IJ
                XJ = J
                FACT = FACT*XJ
 190            CONTINUE
             PRODCT = ((1./FACT)*(B/2.)**XK)**2
             SUM = SUM + PRODCT
 180         CONTINUE
          BOI = SUM +1.
          W(I) = BOI/ALF0
          H(I) = H(I)*W(I)
 170      CONTINUE
C*-------------------------------------------------------------------*
C                                                                     *
C     Compute final Wk(nT)*h(nT) impulse response of the filter       *
C     by using symmetrical response characteristics                   *
C                                                                     *
C*-------------------------------------------------------------------*
       DO 520 I = IN,1,-1
          WH(IN+I-1) = H(I)
 520      CONTINUE
       DO 530 I = 1,(IN-1)
          WH(I) = WH(2*IN - I)
 530      CONTINUE
       DO 540 I = 1,N
          WRITE(1,5500) I,WH(I)
 540      CONTINUE
 5500     FORMAT(1X,"WH(",I2,")= ",F15.13)
       END
       END$
```

SPECIFICATIONS FOR THE FILTER

PASS-BAND RIPPLE <= 0.5db PASS-BAND EDGE 2KZ

STOP-BAND ATTN @ > 30dbs STOP-BAND EDGE 3Hz

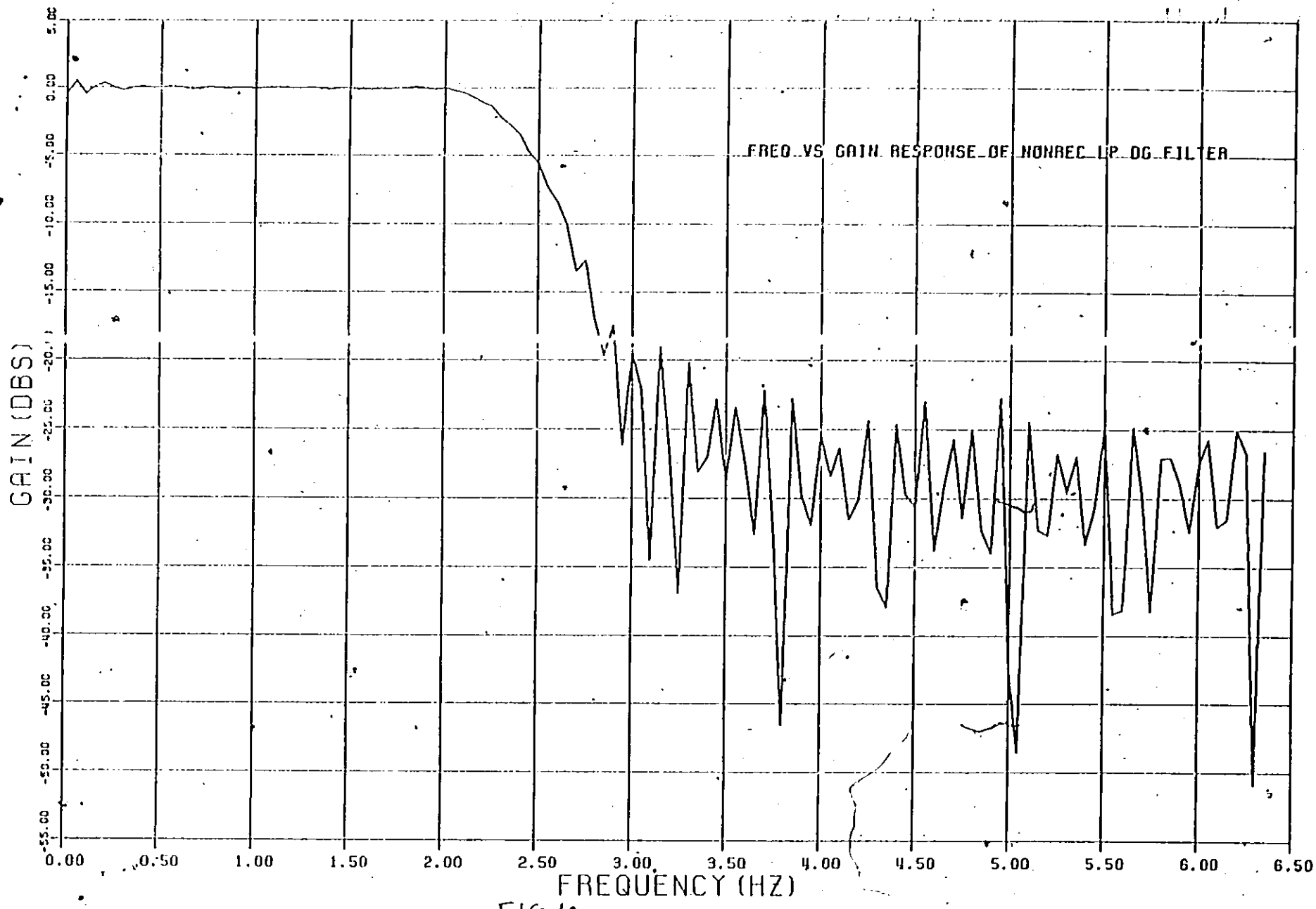


FIG 10

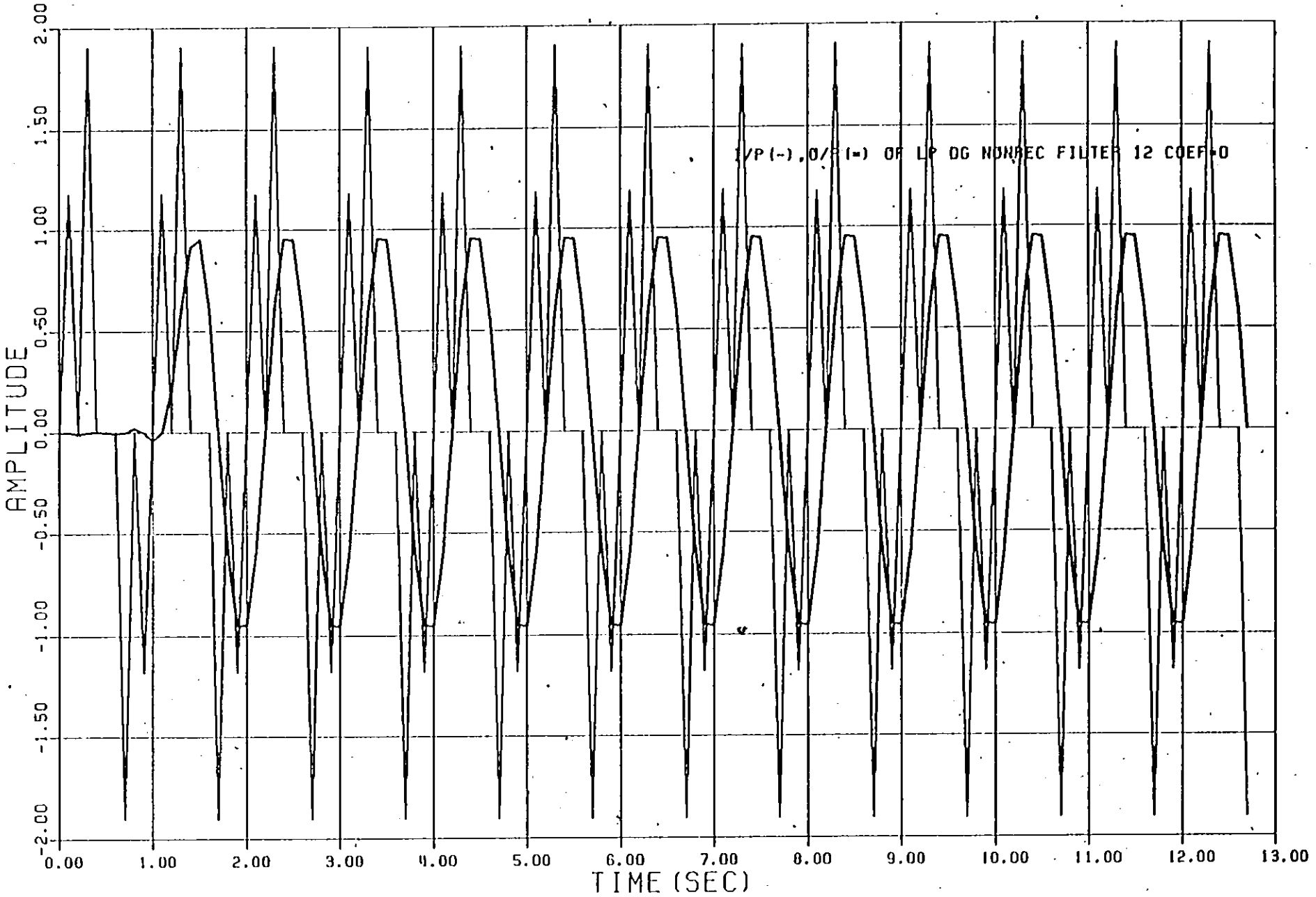


FIG 11 THIN LINE GRAPH - I/P SIGNAL TO THE FILTER 1HZ + 3HZ SUM OF TWO SINUSOIDAL SIGNALS

THICK LINE GRAPH - O/P SIGNAL OF THE FILTER 1HZ PASSES. 3HZ IS ATTENUATED

```
FTN4
      PROGRAM NRFHP(),VERSION 1.0 - 22 MAY 1982
C
C*****************************************************************************
C*****************************************************************************
C***************+                                 +*************************
C***************+     PROGRAM NRFHP               +*************************
C***************+                                 +*************************
C*****************************************************************************
C*****************************************************************************
C                                                                             *
C     PROGRAMMED IN MAY 1982 BY                                               *
C        K. C. Kelkar                                                         *
C     Hydraulics Laboratory                                                   *
C     National Research Council                                               *
C     Ottawa  K1A  0R6                                                        *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C     PROGRAM DESCRIPTION                                                     *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C  Computes coefficients for nonrecursive high-pass digital filter            *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C     PROGRAM DESCRIPTION                                                     *
C                                                                             *
C*-----------------------------------------------------------------------------*
C                                                                             *
C    1) Determine h(nT) using the fourier-series approach assuming            *
C       an idealized frequency response.                                      *
C       H(exp(j*w*T)  = 0  for mod(w) <  WC                                   *
C                     = 1  for WC < = mod(w) < = WS/2                         *
C       WC = 0.5*(WP + WA)
C    2) Choose DELTA in eqation 1 and 2 such that AP < = AP'                  *
C       and AA > = AA'. A suitable value is                                   *
C       DELTA = min(DELTA1 , DELTA2)                                          *
C       where DELTA1 = 10**(-0.05AA') and DELTA2 = A/B                        *
C       A = 10**(0.05*AP')-1 and B = 10**(0.05*AP') + 1                       *
C       AP = 20*log((1 + DELTA)/(1 - DELTA))          equation 1              *
C       AA = -20*log(DELTA)                           equation 2              *
C    3) Calculate AA using equation 2                                         *
```

```
C     4) Choose parameter ALPHA as follows:                                  *
C        ALPHA = 0                                          for AA ( 21    *
C              = 0.5842*(AA - 21)**0.4 + 0.07886*(AA - 21)  for 21(AA(=50*
C              = 0.1102*(AA - 8.7)                          for AA ) 50    *
C     5) Choose parameter D as follows :                                     *
C        D = 0.9222                                         for AA(= 21    *
C          = (AA - 7.95)/14.36                              for AA ) 21    *
C        Then select the lowest odd value of N satisfying the inequality*
C            N ) = WS*D/BT + 1                                             *
C        where transition bandwidth BT = WP - WA                           *
C     6) Form Wk(nT) using equation 3                                        *
C        Wk(nT) = I0(BETA)/I0(ALPHA) for mod(n) ( = (N - 1)/2 equation 3*
C               = 0    otherwise                                           *
C        where ALPHA is an independent parameter and                       *
C        BETA = ALPHA*sqrt(1 - (2*n/(N - 1))**2)                           *
C        Where I0(x) is zeroth-order Bessel function of the first kind     *
C        This can be evaluated to any desired degree of accuracy by        *
C        using the rapidly converging series                               *
C        I0(x) = 1 + SIGMA((1/k!)*(x/2)**k)**2                             *
C     7) Form H'w(z) = z**(-(N - 1)/2*Hw(z) where                           *
C        Hw(z) = Z(Wk(nT)*h(nT))                                           *
C        The above expressions for ALPHA and D are emperical relations     *
C        developed by Kaiser (REF 1)                                       *
C        REF 1  J.F. Kaiser ,Nonrecursive Digital Filter Design Using      *
C        the I0-sinh Window Functions, Proc 1974 IEEE Int Symp.            *
C        Circuit Theory, pp 20-23                                          *
C                                                                          *
C*------------------------------------------------------------------------*
C                                                                          *
C     CONVERSATIONAL INPUT                                                 *
C                                                                          *
C*------------------------------------------------------------------------*
C     FS - Sampling frequency in hertz                                     *
C     FP - Pass-band edge in high-pass filter in hertz                     *
C     FA - Attenuation band edge in high-pass filter in hertz              *
C     AP - Maximum permissible ripple in the pass-band in dbs              *
C     AA - Minimum attenuation in the stop-band in dbs                     *
C                                                                          *
C*------------------------------------------------------------------------*
C                                                                          *
C     DIAGRAM INDICATING FILTER CHARACTERISTICS                            *
C                                                                          *
C*------------------------------------------------------------------------*
C                                                                          *
```

```
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C                                                                      *
C*-----------------------------------------------------------------------*
C                                                                      *
C     PROGRAM OUTPUT                                                   *
C                                                                      *
C*-----------------------------------------------------------------------*
C                                                                      *
C     Minimum number of stages required to realize filter with given   *
C     specifications:                                                  *
C     Coefficients for the filter                                      *
C                                                                      *
C*-----------------------------------------------------------------------*
      REAL WP,WC,WA,WS,H(30),W(30),WH(100)
C     WP - Passband edge
C     WC - Mid-frequency between WP and WA
C     WA - Attenuation band edge
C     WS - Sampling frequency
C-----------------------------------------------------------------------*
C                                                                      *
C     Get specifications for the filter                                *
C                                                                      *
C-----------------------------------------------------------------------*
      WRITE(1,7090)
 7090 FORMAT(1X,"ENTER SAMPLING FREQUENCY IN HZ")
      READ(1,*) FS
      WRITE(1,8000)
 8000 FORMAT(1X,"ENTER PASS-BAND EDGE IN HZ")
      READ(1,*) FP
      WRITE(1,8010)
 8010 FORMAT(1X,"ENTER STOP-BAND EDGE IN HZ")
      READ(1,*) FA
      WRITE(1,8020)
 8020 FORMAT(1X,"ENTER RIPPLE IN DBS IN THE PASS-BAND")
```

```
      READ(1,*) AP
      WRITE(1,8030)
 8030 FORMAT(1X,"ENTER MINIMUM ATTENUATION IN THE STOP-BAND IN DRS")
      READ(1,*) AA
      PII = 4.*ATAN(1.)
      PI2 = 2.*PII
      WS  = PI2*FS
      WP  = PI2*FP
      WA  = PI2*FA
      WC  = 0.5*(WP + WA)
      WRITE(1,8050) WS,WP,WA,WC
 8050 FORMAT(1X,"WS WP WA WC",4F5.2)
C                                                                      *
C*    DELTA = MIN(DELTA1,DELTA2)                                       *
C                                                                      *
      DELTA1 = 10.**(-0.05*AA)
      DELTA2 = (10.**(0.05*AP) - 1)/(10.**(0.05*AP) + 1)
      WRITE(1,8060) DELTA1,DELTA2
 8060 FORMAT(1X,"DELTA1 DELTA2",2F8.4)
      IF(DELTA1.LE.DELTA2) 200,210
  200 DELTA = DELTA1
      GO TO 220
  210 DELTA = DELTA2
  220 CONTINUE
      WRITE(1,8070) DELTA
 8070 FORMAT(1X,"DELTA ",F8.4)
C                                                                      *
C*    Calculate AA using equation AA = -20*log(DELTA)                  *
C                                                                      *
      AA = -20.*ALOGT(DELTA)
      WRITE(1,9000) AA
 9000 FORMAT(1X,"AA ",F8.4)
C                                                                      *
C*    Choose parameter ALPHA as follows:                               *
C*    ALPHA = 0 for AA < = 21                                          *
C*          = 0.5842*(AA-21)**0.4 + 0.07886*(AA-21) for 21< AA < = 50  *
C*          = 0.1102*(AA - 8.7)   for AA > 50                          *
C                                                                      *
      IF(AA.LE.21) 300,310
  300 ALPHA = 0.0
      GO TO 350
  310 IF(AA.LE.50.AND.AA.GT.21) 100,110
  100 ALPHA = 0.5842*(AA-21.)**0.4 + 0.07886*(AA - 21.)
      GO TO 350
```

```
 110  ALPHA = 0.1102*(AA - 8.7)
 350  WRITE(1,9010) ALPHA
 9010 FORMAT(1X,"ALPHA ",F8.4)
C                                                                      *
C*    Choose parameter D as follows:                                   *
C*    D = 0.9222    for AA < = 21                                      *
C*      = (AA - 7.95)/14.36  for AA > 21                               *
C                                                                      *
      IF(AA.LE.21) 400,410
 400  D = 0.9222
      GO TO 450
 410  D = (AA - 7.95)/14.36
 450  WRITE(1,9020) D
 9020 FORMAT(1X,"D ",F8.4)
C                                                                      *
C*    Select the lowest odd value of N satisfying the inequality       *
C*    N > = WS*D/BT + 1                                                *
C*    where BT = WA - WP                                               *
C                                                                      *
      BT = WP - WA
      XN = WS*D/BT + 1
      WRITE(1,9005) XN
 9005 FORMAT(1X,"XN ",F8.4)
      N  = INT(XN)
      IF(N.LT.XN) N = N + 1
      WRITE(1,9030) N
 9030 FORMAT(1X,"N=",I3)
C     WRITE(1,9011)
      IP = MOD(N,2)
      IF(IP.EQ.0) 510,500
 510  N = N + 1
C     WRITE(1,9031) IP
C9031 FORMAT(1X,"IP = ",I2)
C9011 FORMAT(1X,"ENTER NEAREST ODD N")
C     READ(1,*)N
 500  WRITE(1,9032) N
 9032 FORMAT(1X,"NEAREST ODD N = ", I3)
C                                                                      *
C*    COMPUTE h(nT) by the formula                                     *
C*    h(nT) = 1./n*PII*sin(WC*n*T)                                     *
C*    where WC = 0.5(WP + WA)                                          *
C*          T  = PI2/WS                                                *
C                                                                      *
      T = PI2/WS
```

```
      IN = N/2 + 1
      WRITE(1,9040) IN
 9040 FORMAT(1X,"IN ",I3)
      H(1) = (1./PII)*(0.5*WS*T - WC*T)
      DO 130 I = 2,IN
         XI = I
         XI = XI - 1.0
         H(I) = (1./(XI*PII))*(SIN(0.5*WS*XI*T) - SIN(WC*XI*T))
  130    CONTINUE
         WRITE(1,9080) (H(I),I=1,IN)
 9080    FORMAT(F10.8)
C-----------------------------------------------------------------------*
C                                                                       *
C     Compute window function by following formula                      *
C     Wk(nT) = I0(B)/I0(ALPHA)   for mod(n) < = (N-1)/2                 *
C            = 0     otherwise                                          *
C     ALPHA is an independent parameter                                 *
C     B  = ALPHA*SQRT(1 - (2.n/(N-1))**2)                               *
C     and  I0(X) = 1 + SIGMA((1/k!)*(X/2)**k)**2   k= 1,infinity        *
C     in practical case series converges rapidly after k >10            *
C                                                                       *
C     Compute I0(ALPHA)                                                 *
C                                                                       *
      SUM = 0.0
      DO 150 K = 1, 10
         IJ = K
         XK = K
         FACT = 1.
         DO 160 J = 1,IJ
            XJ = J
            FACT = FACT*XJ
  160       CONTINUE
         PRODCT = ((1./FACT)*(ALPHA/2.)**XK)**2
         SUM = SUM + PRODCT
  150    CONTINUE
      ALF0  = SUM +1.0
C                                                                       *
C     Compute I0(B) and then Wk(nT)                                     *
C                                                                       *
      XN = N - 1
      DO 170 I = 1,IN
         IP = I - 1
         XP = IP
         B = ALPHA*SQRT(1- (2.*XP/XN)**2)
```

```
          SUM = 0.0
          DO 180 K = 1,10
             IJ = K
             FACT = 1.0
             XK = K
             DO 190 J = 1,IJ
                XJ = J
                FACT = FACT*XJ
 190         CONTINUE
             PRODCT = ((1./FACT)*(B/2.)**XK)**2
             SUM = SUM + PRODCT
 180         CONTINUE
          BOI = SUM +1.
          W(I) = BOI/ALF0
          H(I) = H(I)*W(I)
 170      CONTINUE
C*-------------------------------------------------------------------*
C                                                                    *
C     Compute final Wk(nT)*h(nT) impulse response of the filter      *
C     by using symmetrical response characteristics                  *
C                                                                    *
C*-------------------------------------------------------------------*
       DO 520 I = IN,1,-1
          WH(IN+I-1) = H(I)
 520      CONTINUE
       DO 530 I = 1,(IN-1)
          WH(I) = WH(2*IN - I)
 530      CONTINUE
       DO 540 I = 1,N
          WRITE(1,5500) I,WH(I)
 540      CONTINUE
 5500     FORMAT(1X,"WH(",I2,")= ",F15.13)
       END
       END$
```

SPECIFICATIONS FOR THE HP FILTER

PASS-BAND RIPPLE ≤ 0.5db STOP-BAND EDGE 2HZ

STOP-BAND ATTN > 30dbs PASS-BAND EDGE 3HZ

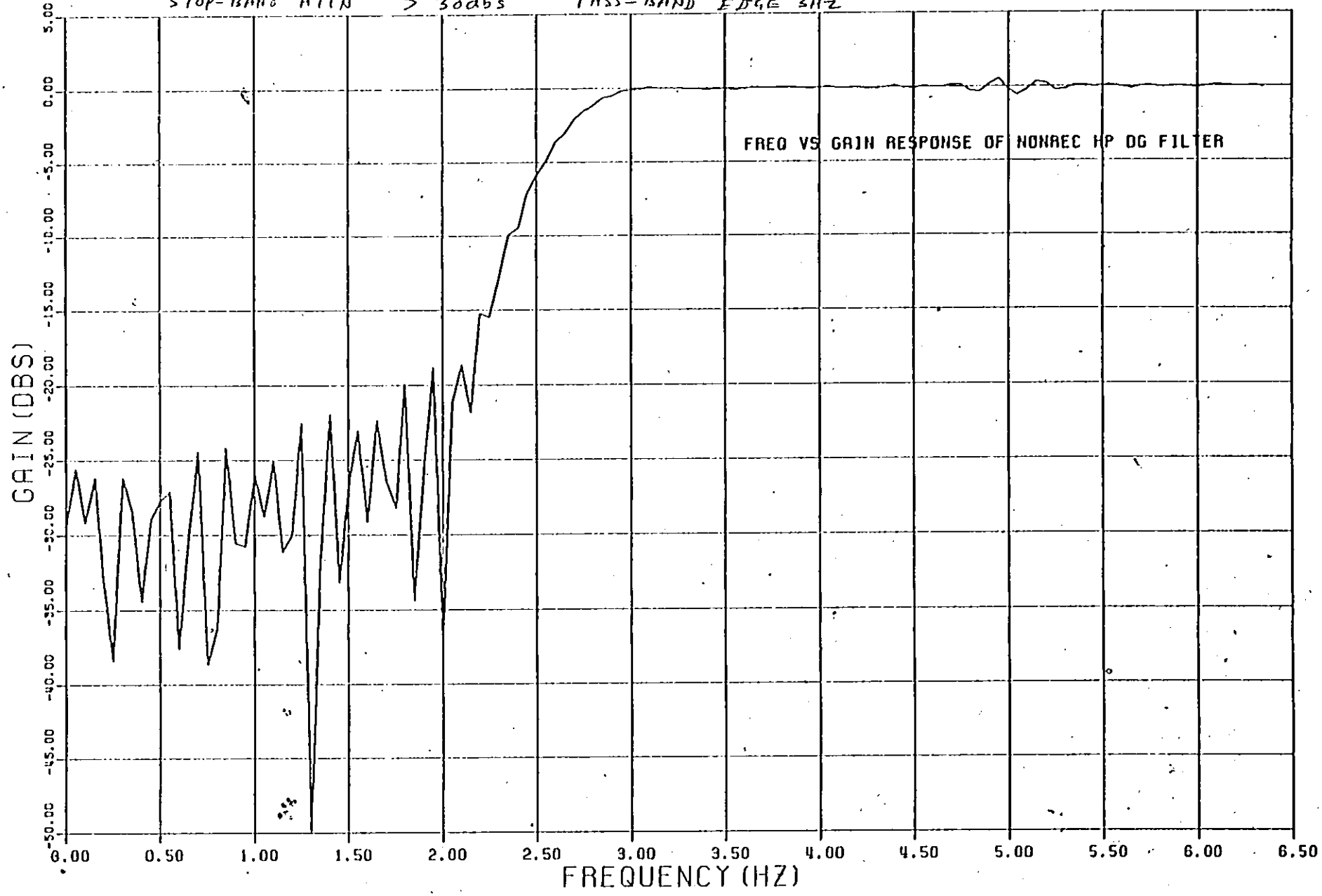


FIG 12

SPECIFICATIONS FOR THE BP FILTER

PASS-BAND RIPPLE ≤ 0.5db PASS-BAND EDGES 0.6Hz, 2Hz

STOP-BAND ATTN > 30dbs STOP-BAND EDGES 0.4Hz, 3Hz

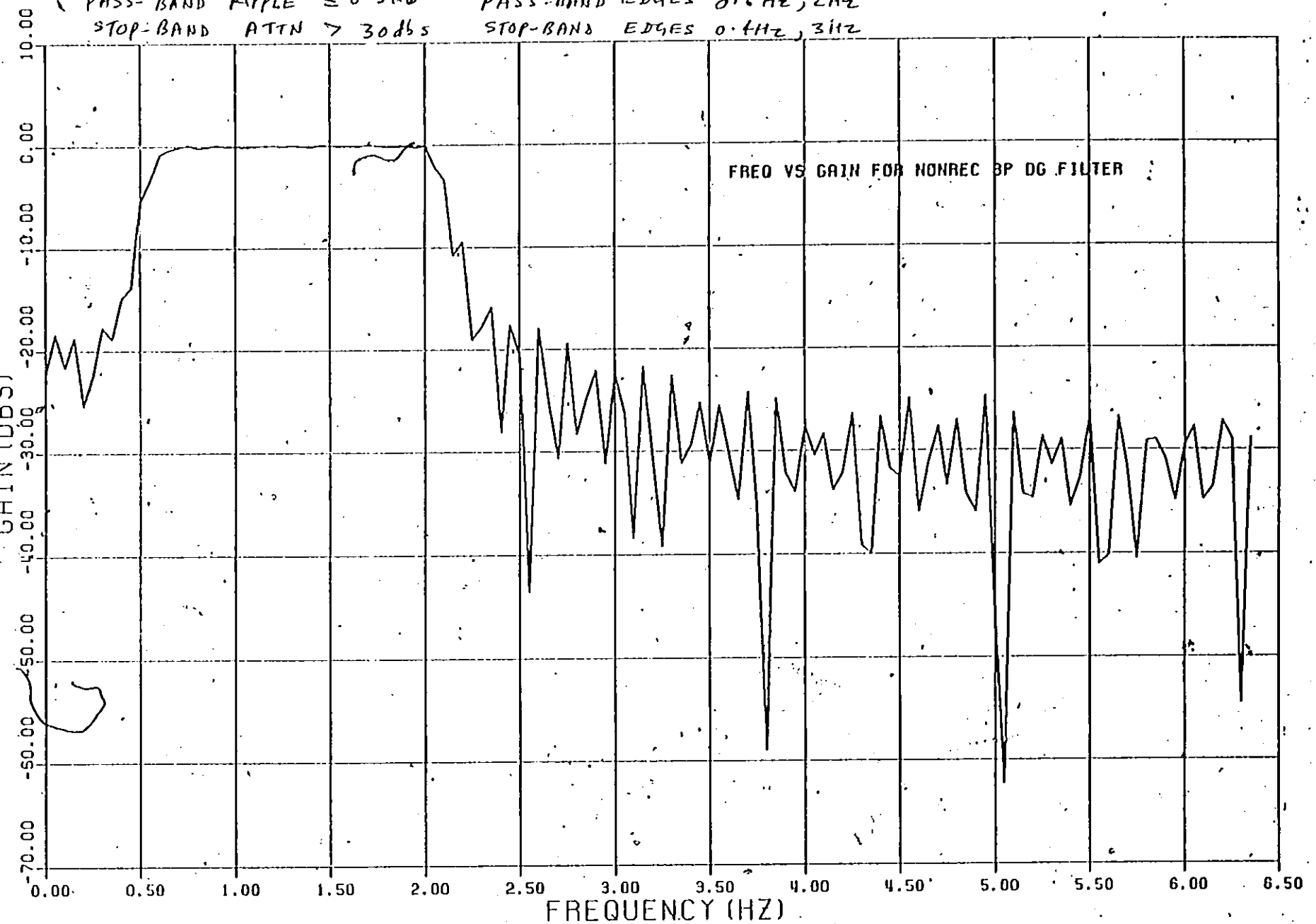


FIG 13

APPENDIX II

```
FTN4
      PROGRAM NRFPV
      REAL X(128),XI(128),Y(128),C(128)
      INTEGER ISTART(5),IEND(5)
C
C     Program to measure execution time of non-recursive
C     digital filter of order N with push down stack data
C     structure and using VIS on F-series machine.
C
C     Get value of order of the filter
C
      WRITE(1,1000)
 1000 FORMAT(1X,"ENTER ORDER OF THE FILTER")
      READ(1,*) NTIME
C
C     Generate simulated data and coefficient array
C
      DO 100 I = 1,NTIME
         XI(I) = URAN(1)
          X(I) = XI(I)
          C(I) = XI(I)
  100     CONTINUE
C
C     Note system time at the start of the program
C
      ICODE = 11
      CALL EXEC(ICODE,ISTART)
      DO 110 K = 4
         DO 120 L = 1,25
C
C     Push down NTIME - 1 data samples by one location
C
            DO 130 I = NTIME,2,-1
  130          X(I) = X(I-1)
C
C     Put recent sample on the top of the stack
C
            X(1) = XI(L)
C
C     Convolve data with finite Impulse response of the filter
C
  120    CALL VDOT(Y(L),X,1,C,1,NTIME)
  110    CONTINUE
C
```

```
C     Note system time at the end of execution of the program.
C
      CALL EXEC(ICODE,IEND)
      WRITE(1,1010) (IEND(I),I = 5,1,-1)
      WRITE(1,1010) (ISTART(I), I = 5,1,-1)
 1010 FORMAT(1X,5(I3,X))
      END
      END$
```

```
FTN4
      PROGRAM NRFRV
      REAL X(128),XI(128),Y(128),C(128)
      INTEGER ISTART(5),IEND(5)
C
C     Program to measure execution time of nonrecursive filter
C     implementation using Ring buffer data structure and VIS
C     instruction on F-series machine.
C
C
C     Get order of the filter
C
      WRITE(1,1000)
 1000 FORMAT(1X,"ENTER ORDER OF THE FILTER")
      READ(1,*) NTIME
C
C     Initialise pointer J to bottom of the ring buffer
C
      J = NTIME
C
C     Generate simulated data and filter coefficients
C
      DO 100 I = 1,NTIME
         XI(I) = URAN(1)
          X(I) = XI(I)
  100     C(I) = XI(I)
C
C     Note the system time at the start of program execution
C
      ICODE = 11
      CALL EXEC(ICODE,ISTART)
      DO 110 K = 1,4
         DO 120 L = 1,25
            X(J) = XI(L)
            IN1 = NTIME - J + 1
            CALL VDOT(SCAL1,X(J),1,C(1),1,IN1)
            IN2 = NTIME - IN1
            CALL VDOT(SCAL2,X(1),1,C(IN1+1),1,IN2)
            Y(L) = SCAL1 + SCAL2
            J = J - 1
            IF(J.EQ.0) J = NTIME
  120       CONTINUE
  110    CONTINUE
C
```

```
C     Note system time at the end of execution of the program
C
      CALL EXEC(ICODE,IEND)
      WRITE(1,1010) (IEND(I),I = 5,1,-1)
      WRITE(1,1010) (ISTART(I), I = 5,1,-1)
 1010 FORMAT(1X,5(I3,X))
      END
      END$
```

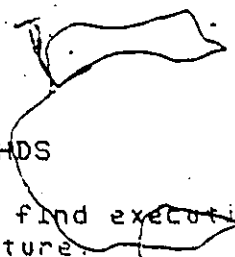
```
FTN4
      PROGRAM NRFPV
      REAL X(128),XI(128),Y(128),C(128)
      INTEGER ISTART(5),IEND(5)
C
C     Program to measure execution time of non-recursive
C     digital filter of order N with push down stack data
C     structure and using VIS on F-series machine.
C
C     Get value of order of the filter
C
      WRITE(1,1000)
 1000 FORMAT(1X,"ENTER ORDER OF THE FILTER")
      READ(1,*) NTIME
C
C     Generate simulated data and coefficient array
C
      DO 100 I = 1,NTIME
         XI(I) = URAN(1)
          X(I) = XI(I)
          C(I) = XI(I)
  100     CONTINUE
C
C     Note system time at the start of the program
C
      ICODE = 11
      CALL EXEC(ICODE,ISTART)
      DO 110  K = 1,40
         DO 120 L = 1,25
C
C     Push down NTIME - 1 data samples by one location
C
            DO 130 I = NTIME,2,-1
  130           X(I) = X(I-1)
C
C     Put recent sample on the top of the stack
C
            X(1) = XI(L)
C
C     Convolve data with finite Impulse response of the filter
C
         Y(L) = 0
         DO 160 I = 1,NTIME
  160    Y(L) = Y(L) + C(I)*X(I)
```

```
 120      CONTINUE
 110    ' CONTINUE
C
C     . Note system time at the end of execution of the program '
C
       CALL EXEC(ICODE,IEND)
       WRITE(1,1010) (IEND(I),I = 5,1,-1)
       WRITE(1,1010) (ISTART(I), I = 5,1,-1)
 1010 FORMAT(1X,5(I3,X))
       END
       END$
```

```
FTN4
      PROGRAM NRRWV
      REAL XI(128),X(128),Y(128),C(128)
      INTEGER ISTART(5),IEND(5)
C
C     Program to measure execution time of program which implements
C     nonrecursive filter of N stages with Ring buffer data
C     structure without VIS on F - series machine
C
C     Get order of the filter
C
      WRITE(1,1000)
 1000 FORMAT(1X,"ENTER ORDER OF THE FILTER")
      READ(1,*) NTIME
C
C     Initialise poiter of Ring buffer at the bottom of the buffer
C
      J = NTIME
C
C     Generate simulated data and coeffients for the filter
C
      DO 100 I = 1,NTIME
         XI(I) = URAN(1)
          X(I) = XI(I)
  100     C(I) = XI(I)
C
C     Note system time at the start of execution of the program
C
      ICODE = 11
      CALL EXEC(ICODE,ISTART)
      DO 110 K = 1,4
      DO 120 L = 1,25
      X(J) = XI(L)
      IN1 = NTIME - J + 1
      SUM = 0
      DO 130 I = 1, IN1
  130 SUM = SUM + C(I)*X(J+I-1)
      IN2 = NTIME - IN1
      DO 140 I = 1,IN2
  140 SUM = SUM + X(I)*C(IN1+I)
      Y(L) = SUM
  120 CONTINUE
  110 CONTINUE
C
```

```
C.      Note system time at the end of execution of the program
C
        CALL EXEC(ICODE,IEND)
        WRITE(1,1010)(IEND(I),I = 5,1,-1)
        WRITE(1,1010)(ISTART(I),I = 5,1,-1)
 1010   FORMAT(1X,5(I3,X))
        END
        END$
```

APPENDIX III



```
FTN4
      PROGRAM PSHDS
C
C     Program to find execution time to manupulate push down stack
C     data structure.
C
      REAL W(1024),X(1024),Y(1024),A(1024)
      INTEGER ISTART(5),IEND(5)
C
C     Get no. of samples in the stack
C
      WRITE(1,1000)
 1000 FORMAT(1X,"ENTER NO. OF SAMPLES")
      READ(1,*) NTIME
C
C     Generate simulated signal
C
      DO 100 I = 1,NTIME
  100    X(I) = 10.*(URAN(1) - 0.5)
C
C     Compute window function
C
      XN = FLOAT(NTIME)
      DO 110 I = 1,NTIME
         XI = FLOAT(I-1)
  110    W(I) = 0.5*(1.0 - XI/XN)
C
C     Note time when program starts execution
C
      ICODE = 11
      CALL EXEC(ICODE,ISTART)
      DO 120 K = 1,1
         DO 125 L = 1,NTIME
C
C     Push down all samples by one location
C
      DO 130 I = NTIME,2,-1
  130    X(I) = X(I-1)
      Y(1) = X(L)
      DO 140 I = 1,NTIME
  140    A(I) = 0.95*A(I) + 0.05*Y(1)*Y(I)
  125  CONTINUE
       DO 150 I = 1,NTIME
  150  A(I) = W(I)*A(I)
```

```
 120    CONTINUE
C
C       Note time at the end of execution
C
        ICODE = 11
        CALL EXEC(ICODE,IEND)
        WRITE(1,1010)(IEND(I),I=5,1,-1)
        WRITE(1,1010)(ISTART(I),I=5,1,-1)
 1010   FORMAT(1X,5(I3,X))
        END
        END$
```

```
FTN4
      PROGRAM RINGB
C
C     Program to measure execution time to manupulate Ring buffer
C
      INTEGER IEND(5),ISTART(5)
      REAL X(1024),Y(1024),A(1024),W(1024)
C
C     Get no. of samples in Ring buffer
C
      WRITE(1,1000)
 1000 FORMAT(1X,"ENTER NTIME")
      READ(1,*) NTIME
C
C     Generate simulated data
C
      DO 100 I=1,NTIME
      X(I) = 10.0*(URAN(1) - 0.5)
  100 CONTINUE
C
C     Compute smoothing window function
C
      XN = FLOAT(NTIME)
      DO 110 I = 1,NTIME
      XI = FLOAT(I-1)
      W(I) = 0.5*(1.0-XI/XN)
  110 CONTINUE
C
C     Initialise pointer to the bottom of the buffer
C
      J = NTIME
C
C     Note system time at the start of program execution
C
      CALL EXEC(11,ISTART)
      DO 120 K = 1,NTIME
C
C     Comutation from pointer upto end of buffer
C
      Y(J) = X(K)
      IP = J-1
      DO 130 IK = J,NTIME
      L = IK - IP
      A(L) = 0.95*A(L) + 0.05*Y(J)*Y(IK)
```

```
 130  CONTINUE
C
C     Computation from top of the buffer upto pointer
C
      IP2 = NTIME - J + 1
      DO 140 IK = 1,(J-1)
      L = IK+IP2
      A(L) = 0.95*A(L) + 0.05*Y(J)*Y(IK)
 140  CONTINUE
      J = J-1
C
C     If pointer has reached the top initialise it to bottom of
C     the buffer
C
      IF(J.EQ.0) 60,70
 60   J = NTIME
 70   CONTINUE
 120  CONTINUE
C
C     Note system time at the end of execution of the program
C
      CALL EXEC(11,IEND)
      WRITE(1,1010)(IEND(I),I=5,1,-1)
      WRITE(1,1010)(ISTART(I),I=5,1,-1)
 1010 FORMAT(1X,5(I3,X))
      END
      END$
```

APPENDIX IV

```
C
FTN4
      PROGRAM RMSA1(),VERSION 1.0 -  4 JUNE  1982
C
C*****************************************************************************
C*****************++++++++++++++++++++++++++++++*****************************
C****************+                              +*****************************
C****************+      PROGRAM RMSA1           +*****************************
C****************+                              +*****************************
C*****************++++++++++++++++++++++++++++++*****************************
C*****************************************************************************
C*                                                                           *
C*      PROGRAMMED IN MAY 1982 BY                                            *
C*      K.C.Kelkar                                                           *
C*      Hydraulics Laboratory                                                *
C*      National Research Council                                            *
C*      Ottawa , K1A 0R6                                                     *
C*                                                                           *
C*---------------------------------------------------------------------------*
C*    PROGRAM SUMMARY                                                        *
C*---------------------------------------------------------------------------*
C*                                                                           *
C*    On-line Mean and Std.Deviation Analysis                                *
C*                                                                           *
C*---------------------------------------------------------------------------*
C*    PROGRAM DESCRIPTION                                                    *
C*---------------------------------------------------------------------------*
C*                                                                           *
C*    -Get values for the parameters ISRAT,IDISCN,IPORT,ICHAN                *
C*    -Read calibration file for the port                                    *
C*                                                                           *
C*     Procedure for first order smoothing                                   *
C*                                                                           *
C*    -Pick up X                                                             *
C*    -Compute AV = ALF*AV + (1. - ALF)*X                                    *
C*    -Compute SQR = ALF*SQR + (1. - ALF)*X*X                                *
C*    -Compute RMS = SQRT(AMAX1(0. , SQR-AV*AV))                             *
C*    -Decrement display counter.                                            *
C*    -IF display counter is zero                                            *
C*     THEN :                                                                *
C*          - Display RMS                                                    *
C*          - RESET Display counter                                          *
C*          - Wait for next Real-time request                                *
C*     ELSE :                                                                *
```

```
C*          - Wait for next real time request                          *
C*                                                                     *
C*-------------------------------------------------------------------*
C*    CONVERSATIONAL INPUT(INPUT UNIT=LUC):                            *
C*-------------------------------------------------------------------*
C*                                                                     *
C*    IREST -Response time in tens of milli-seconds i.e. rate at which *
C*           data are sampled.                                         *
C*    IDISC -Display update rate number of times reading is to be      *
C*           updated per second.                                       *
C*    IPORT -Port number to which channel is connected                 *
C*    ICHAN -Channel number whose RMS and AVE readings are desired     *
C*                                                                     *
C*    Remaining parameters are only for test purpose in final version  *
C*    these will be omitted.                                           *
C*    IK1   -Number of frequency components in the input signal        *
C*    FR(i) - ith frequency component in HZ in the input signal        *
C*    AM(i) - Amplitude of ith frequency component in the input signal *
C*    PH(i) - Phase angle in degrees of ith frequency component        *
C*    NOCYC - The number of program cycles                             *
C*    IN1   - The name of the input GEDAP data file                    *
C*    IOUT1 - The name of the output GEDAP data file                   *
C*                                                                     *
C*-------------------------------------------------------------------*
C*    GEDAP TRANSFER FILE EXAMPLE                                      *
C*-------------------------------------------------------------------*
C*                                                                     *
C*-------------------------------------------------------------------*
C*    FORMATTED OUTPUT(OUTPUT UNIT=LUL):                               *
C*-------------------------------------------------------------------*
C*                                                                     *
C*-------------------------------------------------------------------*
C*    LOG FILE OUTPUT(OUTPUT UNIT=LOG):                                *
C*-------------------------------------------------------------------*
C*                                                                     *
C*    IPSID  -The GEDAP program name                                   *
C*    IN1    -The name of the input GEDAP data files                   *
C*    IOUT1  -The name of the output GEDAP data files                  *
C*                                                                     *
C*-------------------------------------------------------------------*
C*    GEDAP HEADER INPUT:                                              *
C*-------------------------------------------------------------------*
C*                                                                     *
C*                                                                     *
```

```
C*-------------------------------------------------------------------*
C*    GEDAP HEADER OUTPUT:                                           *
C*-------------------------------------------------------------------*
C*                                                                   *
C*    None                                                           *
C*                                                                   *
C*-------------------------------------------------------------------*
C*    SUBROUTINES AND FUNCTIONS CALLED:                              *
C*-------------------------------------------------------------------*
C*                                                                   *
C*    Nil                                                            *
C*                                                                   *
C*-------------------------------------------------------------------*
C*    PROGRAM LIMITATIONS:                                           *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*    ERROR CODES:                                                   *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*    LOADING INSTRUCTIONS                                           *
C*-------------------------------------------------------------------*
C*                                                                   *
C*    TR,LXXXXX                                                      *
C*                                                                   *
C*-------------------------------------------------------------------*
C*    REPORT ANY BUGS IN THIS SPACE                                  *
C*-------------------------------------------------------------------*
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*                                                                   *
```

```
C*                                                                      *
C*                                                                      *
C************************************************************************
C
C
C
C
C
C*************************************************************************
C***                      Module #1                                    ***********
C***  System required GEDAP DIMENSION and COMMON statements            ***********
C***                                                                   ***********
C*************************************************************************
C
      INTEGER JOBID(3),IPSID(3),ISMP(3)
      INTEGER IREST,IDSCN,VALUE,ICRSOR(4),IAV(16)
      REAL SCFIT(8),ALFA(16),ASCFA(16),ALFSC(16)
      REAL X,AV,SQR,RMS,ALF,BETA,DISCN,FR(10),AM(10),PHI(10)
      COMMON/CFIX/IFIX(484),IN1(3),IOUT1(3),IOUTL(3)
      COMMON/IO/LUC,LUL,IPRINT,ISCI,ICRI,ISCO,ICRO
      COMMON/SYSIO/LOG,LUS,LU1,LU2,LU3
      COMMON/GDFMP/IDCBI(144),IDCBO(144),IBUF1(40),IPRAM(5),IRBUF(33)
      COMMON IDABU(128)
C
C
C*************************************************************************
C***                Module #2                                  ***********
C***  System required GEDAP EQUIVALENCE statements             ***********
C***                                                           ***********
C*************************************************************************
C
      EQUIVALENCE(IFIX(2),JOBID(1))
      EQUIVALENCE(IFIX(6),VERNO)
      EQUIVALENCE(IFIX(12),IFORM)
      EQUIVALENCE(IFIX(13),ITYPE1)
      EQUIVALENCE(IFIX(14),ITYPE2)
      EQUIVALENCE(IFIX(20),N)
      EQUIVALENCE(IFIX(25),DFL)
      EQUIVALENCE(IFIX(60),SCFI)
      EQUIVALENCE(IFIX(260),TR)
      EQUIVALENCE(IFIX(96),ALFSC(1))
      EQUIVALENCE(IFIX(190),IAV(1))
      EQUIVALENCE(IFIX(207),ASCFA(1))
      EQUIVALENCE(IFIX(262),ALFA(1))
```

```
      EQUIVALENCE(IFIX(300),HSCL)
C
C
C*********************************************************************
C***                    Module #3                                  ***
C***  System required GEDAP initialization statements             ***
C***                                                               ***
C*********************************************************************
C
C*-------------------------------------------------------------------*
C*    Insert program name and version number                         *
C*-------------------------------------------------------------------*
      DATA IPSID/2HRM,2HSA,2HN /
      DATA INDI,INDO,ICNTR,NOCYC/1,1,1,1/
      DATA ICRSOR/015446B,060465B,074461B,030103B/
      DATA VALUE/0/
      DATA SQR,AV,X/0.0,0.0,0.0/
      IF(VALUE.EQ.0)100,160
  100 CALL GETLU(IER)
      IF(IER.LT.0)GO TO 999
      VERNO=1.1
      WRITE(LOG,994)IPSID,VERNO
  994 FORMAT(" GEDAP PROGRAM: ",3A2," VERSION:",F7.2)
C
C*-------------------------------------------------------------------*
C*    Obtain the number of program cycles                            *
C*-------------------------------------------------------------------*
      IF (IPRINT .EQ. 1) WRITE(LUC,995)
  995 FORMAT(" ENTER NUMBER OF PROGRAM CYCLES  _")
      READ(LUC,*) NOCYC
C
C
 1000 CONTINUE
C*********************************************************************
C****       MODULE #4                                            ****
C****  OBTAIN GEDAP INPUT FILE NAME                              ****
C****                                                            ****
C*********************************************************************
      CALL GDIN(1,IER)
      IF(IER.LT.0) GO TO 999
      WRITE(LOG,991) IN1,ISCI,ICRI
  991 FORMAT("INPUT FILE =",3A2,":",A2,":",I5)
C
C*********************************************************************
```

```
C****              MODULE #5                          **********************
C**** INPUT GEDAP HEADER FILE AND OPEN DATA FILE ***********************
C****                                                 **********************
C*********************************************************************
      IFORMB = 1
      IFORMB = 0
      ITYPE1 = 0
      ITYPE2 = 0
      CALL HDRIN(IFORMA,ITYPE1,ITYPE2,IFORMB,IER)
      IF(IER.LT.0) GO TO 999
      DO 700 I = 1,16
  700 SCFIT(I) = (ASCFA(I)/1000.)*HSCL
C-------------------------------------------------------------------*
C                                                                   *
C     Initialize simulated time counter to zero                     *
C                                                                   *
C-------------------------------------------------------------------*
      N = 0
C-------------------------------------------------------------------*
C                                                                   *
C     Input parameters for Scan and Display                         *
C                                                                   *
C-------------------------------------------------------------------*
      WRITE(1,8000)
 8000 FORMAT(5X,"ENTER DATA SCAN RATE(IN SCANS PER SECOND)")
      READ(1,*)ISRATE
      ITRESP = 100/ISRATE
      WRITE(1,8010)
 8010 FORMAT(5X,"ENTER DISPLAY UPDATE RATE (IN DISPLAYS PER SECOND)")
      READ(1,*) DISCN
      TDSPL = 100/DISCN
      IDSCN = TDSPL/ITRESP
      VALUE = IDSCN
C-------------------------------------------------------------------*
C                                                                   *
C     Get parameters for first order smoothing filter               *
C                                                                   *
C-------------------------------------------------------------------*
      WRITE(1,8016)
 8016 FORMAT(5X,"ENTER PARAMETER ALPHA ( TYPICAL VALUE IN THE RANGE
     1 0.93 TO 0.97)")
      READ(1,*) ALF
      BETA = 1. - ALF
```

```
C-------------------------------------------------------------------*
C-------------------------------------------------------------------*
C                                                                   *
C     Get Analog Input Port and Channel Number                      *
C     and compute corresponding Real Time Buffer Pointer            *
C                                                                   *
C-------------------------------------------------------------------*
 131  WRITE(1,8064)
 8064 FORMAT(1X,"ENTER PORT NUMBER")
      READ(1,*) IPORT
      WRITE(1,8065)
 8065 FORMAT(1X,"ENTER CHANNEL NUMBER")
      READ(1,*) ICHAN
      IPTR = (IPORT-1)*16 + ICHAN
C-------------------------------------------------------------------*
C                                                                   *
C    Clear CRT screen                                               *
C                                                                   *
C-------------------------------------------------------------------*
 132  DO 140 I = 1,25
         WRITE(1,8080)
 140     CONTINUE
 8080 FORMAT(" ")
C-------------------------------------------------------------------*
C                                                                   *
C      Lock the program to the memory                               *
C                                                                   *
C-------------------------------------------------------------------*
      ICODE = 22
      ILOCK = 1
      CALL EXEC(ICODE,ILOCK)
C-------------------------------------------------------------------*
C                                                                   *
C     Self-time-schedule                                            *
C                                                                   *
C-------------------------------------------------------------------*
      ICODE = 12
      INAME = 0
      IRESL = 1
      IMULT = ITRESP
      IOFST = -1.
      CALL EXEC(ICODE,INAME,IRESL,IMULT,IOFST)
C-------------------------------------------------------------------*
C                                                                   *
```

```
C     Self-terminate with saving of resources                    *
C                                                                *
C----------------------------------------------------------------*
  150 ICODE = 6
      INAME = 0
      INUMB = 1
      CALL EXEC(ICODE,INAME,INUMB)
C----------------------------------------------------------------*
C****************************************************************
C****                                                        *****
C****    START OF ON-LINE PROCEDURE                          *****
C****                                                        *****
C****************************************************************
C----------------------------------------------------------------*
C                                                                *
C     Compute Average and RMS value of the data by first order   *
C     recursive relation                                         *
C                                                                *
C----------------------------------------------------------------*
  160 DATA = FLOAT(IDABU(IPTR))
      B = DATA - FLOAT(IAV(ICHAN))/3.2767
      FD = B*(1. + ALFA(ICHAN)*B/1000.)
      X = SCFIT(ICHAN)*FD
      AVE = ALF*AVE + BETA*X
      SQR = ALF*SQR + BETA*X*X
      RMS = SQRT(AMAX1(0.,(SQR - AVE*AVE)))
C     Decrement Display counter and branch if not zero
      IDSCN = IDSCN - 1
      IF(IDSCN.EQ.0) 170,150
C----------------------------------------------------------------*
C                                                                *
C    IF Display counter is zero , Reset counter and write results*
C     Position  cursor at row number 5 and column number 10      *
C     Data value for ICRSOR corresponds to ASCII string of       *
C     characters  ESC&a5y10C which when received by CRT terminal HP2648*
C     positions the cursor at row number 5 and column number 10  *
C                                                                *
C----------------------------------------------------------------*
  170 WRITE(1,8090) ICRSOR
 8090 FORMAT(4A2)
      WRITE(1,9000) RMS,AVE
 9000 FORMAT("RMS VALUE = ",F6.4,"     AVERAGE VALUE = ",F6.4)
      IDSCN = VALUE
      GO TO 150
```

```
C-------------------------------------------------------------------------*
C*************************************************************************
C*****                                                              ******
C*****   END OF ON-LINE PROCEDURE                                   ******
C*****                                                              ******
C*************************************************************************
C-------------------------------------------------------------------------*
C     CALL GDOUT(INDO,IER)
C     IF(IER.LT.0)GO TO 999
C
C
C*************************************************************************
C***              Module #9                 ***********************************
C***  Create and output GEDAP header        ***********************************
C***     file then open data file           ***********************************
C***                                        ***********************************
C*************************************************************************
C
C     DO 997 IPRGNM=1,3
C 997 JOBID(IPRGNM)=IPSID(IPRGNM)
C     CALL HDROT(IFORM,ITYPE1,ITYPE2,IER)
C     IF(IER.LT.0)GO TO 999
C
C
C*************************************************************************
C***         Module #10                 ***********************************
C***  Output data to GEDAP data file    ***********************************
C***                                    ***********************************
C*************************************************************************
C
C*-----------------------------------------------------------------------*
C*     "buffer"=user output buffer                                       *
C*      "size" =maximum output buffer size(integer words) as determined  *
C*              from specification statements                            *
C*-----------------------------------------------------------------------*
C     CALL WFOUT("buffer","size",IER)
C     IF(IER.LT.0)GO TO 999
C
C     WRITE(LOG,998) IOUT1,ISCO,ICRD
C 998 FORMAT(" OUTPUT FILE=",3A2,".",A2,":",I5)
C
C     INDO=INDO+1
C
C
```

```
C*-------------------------------------------------------------------*
C*    Check if the program is to be recycled                         *
C*-------------------------------------------------------------------*
C     ICNTR=ICNTR+1
C     IF(ICNTR.LE.NOCYC) GO TO 1000
C
C
C*********************************************************************
C***              Module #11                    **********************
C***  System required GEDAP completion section  **********************
C***                                            **********************
C*********************************************************************
C
  999 CALL GDCLL(IER)
      END
      END$
```

APPENDIX V

```
C
FTN4
      PROGRAM AVEA1(),VERSION 1.0 - 1 JUNE   1982
C
C*****************************************************************************
C*****************+++++++++++++++++++++++++++++++++*****************************
C***************+                                  +*****************************
C***************+       PROGRAM AVEA1              +*****************************
C***************+                                  +*****************************
C*****************+++++++++++++++++++++++++++++++++*****************************
C*****************************************************************************
C*                                                                           *
C*      PROGRAMMED IN MAY 1982 BY                                            *
C*      K.C.Kelkar                                                           *
C*      Hydraulics Laboratory                                                *
C*      National Research Council                                            *
C*      Ottawa  K1A 0R6                                                      *
C*                                                                           *
C*---------------------------------------------------------------------------*
C*    PROGRAM SUMMARY                                                        *
C*---------------------------------------------------------------------------*
C*                                                                           *
C*    On-line Mean analysis and display of reading for 8 simultaneous        *
C*    transducers                                                            *
C*                                                                           *
C*---------------------------------------------------------------------------*
C*    PROGRAM DESCRIPTION                                                    *
C*---------------------------------------------------------------------------*
C*                                                                           *
C*   1-Get values for the parameters IREST,IDISC,IPORT,ISTCH                 *
C*   2-Read calibration information from the port file                       *
C*   3-Pick up datum value X from Real-time buffer                           *
C*   4-Filter the datum through low-pass digital filter                      *
C*   5-Compute AV = ALF*AV + (1.5- ALF)*X                                    *
C*     Do same operation for all the channels                                *
C*   6-Decrement display counter.                                            *
C*     IF display counter is zero                                            *
C*     THEN :                                                                *
C*           - Schedule the program DISPL (immediate without wait)           *
C*             along with passing averaged data buffer                       *
C*           - RESET Display counter                                         *
C*           - Wait for next Real-time request                               *
C*     ELSE :                                                                *
C*           - Wait for next real time request                               *
```

```
C*                                                                      *
C*          - When rescheduled continue with step 3                     *
C*----------------------------------------------------------------------*
C*    CONVERSATIONAL INPUT(INPUT UNIT=LUC):                             *
C*----------------------------------------------------------------------*
C*                                                                      *
C*    IREST -Response time that is rate at which data are sampled       *
C*           in hertz                                                   *
C*    IDISC -Display update rate number of times reading is to be       *
C*           updated per second.                                        *
C*    IPORT -Port number to which channels are connected                *
C*    ISTCH -Start channel address to which first transducer output     *
C*           is connected                                               *
C*    NTIME - Number of data samples to be recorded.                    *
C*    NOCYC - The number of program cycles                              *
C*    IN1   - The name of the input GEDAP data file                     *
C*    IOUT1 - The name of the output GEDAP data file                    *
C*                                                                      *
C*----------------------------------------------------------------------*
C*    GEDAP TRANSFER FILE EXAMPLE                                       *
C*----------------------------------------------------------------------*
C*                                                                      *
C*----------------------------------------------------------------------*
C*    FORMATTED OUTPUT(OUTPUT UNIT=LUL):                                *
C*----------------------------------------------------------------------*
C*                                                                      *
C*----------------------------------------------------------------------*
C*    LOG FILE OUTPUT(OUTPUT UNIT=LOG):                                 *
C*----------------------------------------------------------------------*
C*                                                                      *
C*    IPSID  -The GEDAP program name                                    *
C*    IN1    -The name of the input GEDAP data files                    *
C*    IOUT1  -The name of the output GEDAP data files                   *
C*                                                                      *
C*----------------------------------------------------------------------*
C*    GEDAP HEADER INPUT:                                               *
C*----------------------------------------------------------------------*
C*                                                                      *
C*                                                                      *
C*----------------------------------------------------------------------*
C*    GEDAP HEADER OUTPUT:                                              *
C*----------------------------------------------------------------------*
C*                                                                      *
C*    None                                                              *
```

```
C*                                                                        *
C*------------------------------------------------------------------------*
C*    SUBROUTINES AND FUNCTIONS CALLED:                                   *
C*------------------------------------------------------------------------*
C*                                                                        *
C*    Subroutine FILTR                                                    *
C*                                                                        *
C*------------------------------------------------------------------------*
C*    PROGRAM LIMITATIONS:                                                *
C*------------------------------------------------------------------------*
C*                                                                        *
C*------------------------------------------------------------------------*
C*    ERROR CODES:                                                        *
C*------------------------------------------------------------------------*
C*                                                                        *
C*------------------------------------------------------------------------*
C*    LOADING INSTRUCTIONS                                                *
C*------------------------------------------------------------------------*
C*                                                                        *
C*    TR,LXXXXX                                                           *
C*                                                                        *
C*------------------------------------------------------------------------*
C*    REPORT ANY BUGS IN THIS SPACE                                       *
C*------------------------------------------------------------------------*
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C*                                                                        *
C**************************************************************************
C
C
```

```
C
C
C
C*****************************************************************************
C***                        Module #1                                      ***********
C***  System required GEDAP DIMENSION and COMMON statements                ***********
C***                                                                       ***********
C*****************************************************************************
C
      INTEGER JOBID(3),IPSID(3),ISMP(3)
      INTEGER IREST,IDSCN,VALUE,ICRSR1(4),ICRSR2(4),ICRSR3(4),ICRSR4(4)
      INTEGER ICRSR5(1),IAVE(8),INAM2(3),IAV(16)
      REAL SCLFC(8),VOFST(8),EAVE(8),SCFIT(8),ALFA(16),ASCFA(16)
      REAL ALFSC(16)
      REAL AVES(8,512),AVESD(512)
      REAL X(64),Y(40),DATAI(8),AVE(8),ALF,BETA,DISCN,DATAO(8)
      COMMON/CFIX/IFIX(484),IN1(3),IOUT1(3),IOUTL(3)
      COMMON/IO/LUC,LUL,IPRINT,ISCI,ICRI,ISCO,ICRO
      COMMON/SYSIO/LOG,LUS,LU1,LU2,LU3
      COMMON/GDFMP/IDCBI(144),IDCBO(144),IBUF1(40),IPRAM(5),IRBUF(33)
      COMMON IDABU(128)
C
C
C*****************************************************************************
C***                Module #2                                      **********************
C***  System required GEDAP EQUIVALENCE statements                 **********************
C***                                                               **********************
C*****************************************************************************
C
      EQUIVALENCE(IFIX(2),JOBID(1))
      EQUIVALENCE(IFIX(6),VERNO)
      EQUIVALENCE(IFIX(12),IFORM)
      EQUIVALENCE(IFIX(13),ITYPE1)
      EQUIVALENCE(IFIX(14),ITYPE2)
      EQUIVALENCE(IFIX(20),N)
      EQUIVALENCE(IFIX(25),DEL)
      EQUIVALENCE(IFIX(60),SCFI)
      EQUIVALENCE(IFIX(260),TR)
      EQUIVALENCE(IFIX(96),ALFSC(1))
      EQUIVALENCE(IFIX(190),IAV(1))
      EQUIVALENCE(IFIX(207),ASCFA(1))
      EQUIVALENCE(IFIX(262),ALFA(1))
      EQUIVALENCE(IFIX(300),HSCL)
C
```

```
C
C*****************************************************************
C***                    Module #3                              ***
C***  System required GEDAP initialization statements          ***
C***                                                           ***
C*****************************************************************
C
C*---------------------------------------------------------------*
C*    Insert program name and version number                     *
C*---------------------------------------------------------------*
      DATA IPSID/2HAV,2HEA,2H1 /
      DATA INDI,INDO,ICNTR,NOCYC/1,1,1,1/
      DATA ICRSR1/015446B,060467B,074465B,041440B/
      DATA ICRSR2/015446B,060461B,030171B,032503B/
      DATA ICRSR3/015446B,060461B,032171B,032503B/
      DATA VOFST/0.5,0.6,0.8,0.9,0.8,1.1,1.2,1.3/
      DATA SCLFC/1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8/
      DATA ICRSR4/015446B,060461B,033571B,032503B/
      DATA INAM2/2HDI,2HSP,2H1 /
      DATA SQR,AV,X/0.0,0.0,0.0/
      CALL GETLU(IER)
      IF(IER.LT.0)GO TO 999
      VERNO=1.1
      WRITE(LOG,994)IPSID,VERNO
  994 FORMAT(" GEDAP PROGRAM: ",3A2," VERSION:",F7.2)
C
C*---------------------------------------------------------------*
C*    Obtain the number of program cycles                        *
C*---------------------------------------------------------------*
      IF (IPRINT .EQ. 1) WRITE(LUC,995)
  995 FORMAT(" ENTER NUMBER OF PROGRAM CYCLES  _")
      READ(LUC,*) NOCYC
C
C
 1000 CONTINUE
C
C*****************************************************************
C***       MODULE #4                                           ***
C*** OBTAIN GEDAP INPUT FILE NAME                              ***
C***                                                           ***
C*****************************************************************
C
      CALL GDIN(1,IER)
      IF(IER.LT.0)  GO TO 999
```

```
C
      WRITE(LOG,991) IN1,ISCI,ICRI
 991  FORMAT(" INPUT FILE=",3A2,":",A2,":",I5)
C
C*************************************************************************
C***            MODULE #5                                      ***********
C*** INPUT GEDAP HEADER FILE AND OPEN DATA FILE                ***********
C***                                                           ***********
C*************************************************************************
C
C*-----------------------------------------------------------------------
C* . SET IFORM,ITYPE1,ITYPE2,IFORMB
C*-----------------------------------------------------------------------
      IFORM = 1
      IFORMB = 0
      ITYPE1 = 0
      ITYPE2 = 0
      CALL HDRIN(IFORMA,ITYPE1,ITYPE2,IFORMB,IER)
      IF(IER.LT.0) GO TO 999
      DO 700 I = 1,8
 700  SCFIT(I) = ASCFA(I) / 1000. *HSCL
C-----------------------------------------------------------------------*
C                                                                       **
C    Input parameters for Scan and Display                              **
C                                                                       **
C-----------------------------------------------------------------------*
      IF(VALUE.EQ.0) 100,160
 100  WRITE(1,1001)
 1001 FORMAT(1X,"ENTER NUMBER OF DATA SAMPLES")
      READ(1,*) NTIME
C     WRITE(1,1002)
C1002 FORMAT(1X,"ENTER OFFSET VOLTAGE FOR EACH CHANNEL")
C     DO 101 I = 1,8
C     WRITE(1,1003) I
C1003 FORMAT(1X,"VOFST(",I2,")=?")
C     READ(1,*) VOFST(I)
C101  CONTINUE
C     WRITE(1,1004)
C1004 FORMAT(1X,"ENTER SCAL FACTOR FOR EACH CHANNEL")
C     DO 102 I = 1,8
C     WRITE(1,1005) I
C1005 FORMAT(1X,"SCLFC(",I2,")=?")
C     READ(1,*) SCLFC(I)
C102  CONTINUE
```

```
      WRITE(1,8000)
 8000 FORMAT(5X,"ENTER DATA SCAN RATE(IN SCANS PER SECOND)")
      READ(1,*)ISRATE
      ITRESP = 100/ISRATE
      WRITE(1,8010)
 8010 FORMAT(5X,"ENTER DISPLAY UPDATE RATE (IN DISPLAYS PER SECOND)")
      READ(1,*) DISCN
      TDSPL = 100/DISCN
      IDSCN = TDSPL/ITRESP
      VALUE = IDSCN
C-------------------------------------------------------------------*
C     Get parameters for first order smoothing filter               *
C                                                                   *
C-------------------------------------------------------------------*
      WRITE(1,8016)
 8016 FORMAT(5X,"ENTER PARAMETER ALPHA ( TYPICAL VALUE IN THE RANGE
     1 0.93 TO 0.97)")
      READ(1,*) ALF
      BETA = 1. - ALF

C-------------------------------------------------------------------*
C                                                                   *
C     Get Analog Input Port and Start channel number to which first *
C     transducer output is connected and compute corrsponding       *
C     Real-time buffer pointer                                      *
C                                                                   *
C-------------------------------------------------------------------*
  131 WRITE(1,8064)
 8064 FORMAT(1X,"ENTER PORT NUMBER")
      READ(1,*) IPORT
      WRITE(1,8065)
 8065 FORMAT(1X,"ENTER START CHANNEL NUMBER")
      READ(1,*) ISTCH
      IPTR = (IPORT-1)*16 + ISTCH
C-------------------------------------------------------------------*
C                                                                   *
C    Clear CRT screen                                               *
C                                                                   *
C-------------------------------------------------------------------*
  132 DO 140 I = 1,25
         WRITE(1,8080)
  140    CONTINUE
 8080 FORMAT(" ")
      WRITE(1,8081) ICRSR1
```

```
 8081 FORMAT(4A2)
      WRITE(1,8082)
 8082 FORMAT("  CHAN. NO          1          2          3
     14")
      WRITE(1,8081)ICRSR2
      WRITE(1,8083)
 8083 FORMAT("  AVE VOLTS  ",3(7X,F5.3),9X,F5.3)
      WRITE(1,8081)ICRSR3
      WRITE(1,8084)
 8084 FORMAT("  CHAN. NO          5          6          7
     13")
      WRITE(1,8081)ICRSR4
      WRITE(1,8083)
C-------------------------------------------------------------------*
C                                                                   *
C     Lock the program to the memory                                *
C                                                                   *
C-------------------------------------------------------------------*
      ICODE = 22
      ILOCK = 1
      CALL EXEC(ICODE,ILOCK)
C-------------------------------------------------------------------*
C                                                                   *
C     Self-time-schedule                                            *
C                                                                   *
C-------------------------------------------------------------------*
      ICODE = 12
      INAME = 0
      IRESL = 1
      IMULT = ITRESP
      IOFST = -1
      CALL EXEC(ICODE,INAME,IRESL,IMULT,IOFST)
C-------------------------------------------------------------------*
C                                                                   *
C     Self-terminate with saving of resources                       *
C                                                                   *
C-------------------------------------------------------------------*
  150 ICODE = 6
      INAME = 0
      INUMB = 1
      CALL EXEC(6,0,1)
C-------------------------------------------------------------------*
C*******************************************************************
C****                                                           ****
```

```
C****   START OF ON-LINE PROCEDURE                                      *******
C****                                                                   *******
C*************************************************************************
C-------------------------------------------------------------------------*
C                                                                         *
C     Filter the data and find average by first order smoothing           *
C     recursive relation                                                  *
C                                                                         *
C-------------------------------------------------------------------------*
 160  DO 170 I = 1,8
      DATAI(I) = FLOAT(IDABU(IPTR + I -1))/1000.
      CALL FILTR(DATAI(I),X((I-1)*8 + 1),Y((I-1)*5 + 1),DATAO(I))
      B = DATAO(I) - FLOAT(IAV(I))/3.2767
      FD = B*(1. + ALFA(I)*B/1000.)
      DATAO(I) = SCFIT(I)*FD
      AVE(I) = ALF*AVE(I) + BETA*DATAO(I)
 170     CONTINUE
C     Decrement Display counter and branch if not zero
      IDSCN = IDSCN - 1
      IF(IDSCN.EQ.0) 171,150
C                                                                         *
C     If it is time to update the display then schedule the               *
C     program DISPL without wait along with passage of buffer             *
C     IAVE                                                                *
C                                                                         *
 171  ICODE = 10
      ILEN = 8
      IN = IN +1
      IF(IN.EQ.NTIME) 300,310
 310  DO 173 I = 1,8
         IAVE(I) = INT(EAVE(I)*1000.)
         AVES(I,IN) = EAVE(I)
 173     CONTINUE
      CALL EXEC(ICODE,INAM2,IOP1,IOP2,IOP3,IOP4,IOP5,IAVE,ILEN)
      IDSCN = VALUE
      GO TO 150
C*************************************************************************
C*****                                                                   *****
C*****   END OF ON-LINE PROCEDURE                                        *****
C*****                                                                   *****
C*************************************************************************
C-------------------------------------------------------------------------*
C                                                                         *
 300  IFORM = 2
```

```
      ITYPE1 = 11
      ITYPE2 = 0
      N = NTIME
      DEL = 1./DISCN
      DO 500 I = 1,8
      DO 510 J = 1,NTIME
  510 AVESD(J) = AVES(I,J)
      CALL GDOUT(INDO,IER)
      IF(IER.LT.0)GO TO 999
C
C
C*************************************************************************
C***          Module #9                ***********************************
C*** Create and output GEDAP header    ***********************************
C***    file then open data file       ***********************************
C***                                   ***********************************
C*************************************************************************
C
      DO 997 IPRGNM=1,3
  997 JOBID(IPRGNM)=IPSID(IPRGNM)
      CALL HDROT(IFORM,ITYPE1,ITYPE2,IER)
      IF(IER.LT.0)GO TO 999
C
C
C*************************************************************************
C***          Module #10               ***********************************
C*** Output data to GEDAP data file    ***********************************
C***                                   ***********************************
C*************************************************************************
C
C*-----------------------------------------------------------------------*
C*     "buffer"=user output buffer                                       *
C*      "size" =maximum output buffer size(integer words) as determined  *
C*              from specification statements                            *
C*-----------------------------------------------------------------------*
      CALL WFOUT(AVESD,2*NTIME,IER)
      IF(IER.LT.0)GO TO 999
C
      WRITE(LOG,998) IOUT1,ISCO,ICRO
  998 FORMAT(" OUTPUT FILE=",3A2,":",A2,":",I5)
C
      INDO=INDO+1
C
  500 CONTINUE
```

```
C
C*-------------------------------------------------------------------*
C*    Check if the program is to be recycled                         *
C*-------------------------------------------------------------------*
      ICNTR=ICNTR+1
      IF(ICNTR.LE.NOCYC) GO TO 1000
C
C
C*********************************************************************
C***                Module #11                                     ***
C***  System required GEDAP completion section                     ***
C***                                                               ***
C*********************************************************************
C
  999 CALL GDCLL(IER)
      CALL EXEC(7)
      END
      SUBROUTINE FILTR(DI,XX,YY,DO)
      DIMENSION XX(1),YY(1),DI(1),DO(1)
      REAL AL0(2),AL1(2),BL0(2),BL1(2),AAL0,BBL0,CCL0,HL0
      DATA AL0/1.89072,1.26947/
      DATA AL1/1.18568,-0.20059/
      DATA BL0/0.56096,0.86717/
      DATA BL1/-1.03479,-0.87236/
      DATA HL0,AAL0,BBL0,CCL0/0.2277563,0.039914,0.039914,0.5965589/
      XX(3) = XX(2)
      XX(2) = XX(1)
      XX(1) = DI(1)
      OUT1 = AL0(1)*(XX(1) + XX(3)) + AL1(1)*XX(2) - BL1(1)*YY(1)
     1 -BL0(1)*YY(2)
       YY(2) = YY(1)
       YY(1) = OUT1
       XX(6) = XX(5)
       XX(5) = XX(4)
       XX(4) = OUT1
       OUT2 = AL0(2)*(XX(4) + XX(6)) + AL1(2)*XX(5) - BL1(2)*YY(3)
     1 -BL0(2)*YY(4)
      YY(4) = YY(3)
      YY(3) = OUT2
      XX(8) = XX(7)
      XX(7) = OUT2
      OUT3 = AAL0*XX(7) + BBL0*XX(8) + CCL0*YY(5)
      YY(5) = OUT3
      DO(1) = OUT3*HL0
```

```
      RETURN
      END
      END$
```

```
FTN4
      PROGRAM DISP1(),VERSION 1.0 29 MAY 1982
C
C*****************************************************************
C*****************+++++++++++++++++++++++++++++++*****************
C****************+                               *****************
C***************+    PROGRAM DISPL                +****************
C****************+                               *****************
C*****************+++++++++++++++++++++++++++++++*****************
C*****************************************************************
C*                                                               *
C*     PROGRAMMED IN MAY 1982 BY                                 *
C*     K.C.Kelkar                                                *
C*     Hydraulics Laboratory                                     *
C*     National Research Council                                 *
C*     Ottawa  K1A 0R6                                           *
C*                                                               *
C*---------------------------------------------------------------*
C*     PROGRAM SUMMARY                                           *
C*---------------------------------------------------------------*
C*                                                               *
C*     Displays the averaged value buffer passed by father program *
C*                                                               *
C*---------------------------------------------------------------*
C*     PROGRAM DESRIPTION                                        *
C*---------------------------------------------------------------*
C*                                                               *
C*     -Retrieve the buffer passed by father program which       *
C*      scedyled it                                              *
C*     -Position the cursor at proper location and display       *
C*      the value                                                *
C*     -Remain locked in the memory                              *
C*     -Self terminate                                           *
C*                                                               *
C*---------------------------------------------------------------*
C*     CONVERSATIONAL INPUT                                      *
C*---------------------------------------------------------------*
C*     Nil                                                       *
C*---------------------------------------------------------------*
      INTEGER ICRSR2(4),ICRSR4(4)
      REAL D(8)
      DIMENSION IAVF(8)
      DATA ICRSR2/015446B,060461B,030171B,032503B/
      DATA ICRSR4/015446B,060461B,033571B,032503B/
```

```
C
C*    Retrieve the buffer passed by father program AVEAN
C
      ICODE = 14
      IRWCD = 1
      ILEN = 8
      CALL EXEC(ICODE,IRWCD,IAVE,ILEN)
      DO 100 I = 1,8
  100 D(I) = FLOAT(IAVE(I))/1000.
      WRITE(1,2010) ICRSR2
 2010 FORMAT(4A2)
      WRITE(1,2030)D(1),D(2),D(3),D(4)
 2030 FORMAT("  AVE VOLTS ",3(7X,F5.3),9X,F5.3)
      WRITE(1,2010) ICRSR4
      WRITE(1,2030)D(5),D(6),D(7),D(8)
C
C     Lock to memory
C
      ICODE = 22
      ILOCK = 1
      CALL EXEC(ICODE,ILOCK)
C
C     Self-terminate
C
      ICODE = 6
      CALL EXEC(ICODE)
      END
      END$
```

APPENDIX VI

```
FTN4
      PROGRAM PDSAN(),VERSION 1.0 -  1 JUNE  1982
C
C*****************************************************************************
C*****************+++++++++++++++++++++++++++++++++*****************************
C*****************+                                 *****************************
C*****************+       PROGRAM PDSA1             +*****************************
C*****************+                                 *****************************
C*****************+++++++++++++++++++++++++++++++++*****************************
C*****************************************************************************
C*                                                                            *
C*       PROGRAMMED IN JUNE 1982 BY                                           *
C*       K.C.Kelkar                                                           *
C*       Hydraulics Laboratory                                                *
C*       National Research Council                                            *
C*       Ottawa  K1A OR6                                                      *
C*                                                                            *
C*----------------------------------------------------------------------------*
C*     PROGRAM SUMMARY                                                        *
C*----------------------------------------------------------------------------*
C*                                                                            *
C*     On-line Power density spetrum analysis and display of random           *
C*     wave data                                                              *
C*                                                                            *
C*----------------------------------------------------------------------------*
C*     PROGRAM DESCRIPTION                                                    *
C*----------------------------------------------------------------------------*
C*                                                                            *
C*     -Pick up datum value X from Real-time buffer                           *
C*     -Compute AV = ALF*AV + (1. - ALF)*X                                    *
C*     -Find auto-correlation function                                        *
C*     -Decrement display counter.                                            *
C*     -IF display counter is zero                                            *
C*      THEN :                                                                *
C*            - Schedule the program DISP2 (immediate without wait)           *
C*              along with passing autocorrelation buffer and smoothing       *
C*              window buffer and averaged datum value                        *
C*            - RESET Display counter                                         *
C*            - Wait for next Real-time request                               *
C*      ELSE :                                                                *
C*            - Wait for next real time request                               *
C*            - Repeat the whole procedure from step 1 on next real-time      *
C*              request                                                       *
C*                                                                            *
```

```
C*-------------------------------------------------------------------*
C*    CONVERSATIONAL INPUT(INPUT UNIT=LUC):                          *
C*-------------------------------------------------------------------*
C*                                                                   *
C*    NDATA -Number of data samples stored in the buffer             *
C*    IREST -Response time that is rate at which data are sampled    *
C*           in hertz                                                *
C*    IDISC -Display update rate number of times reading is to be    *
C*           updated per second.                                     *
C*    IPORT -Port number to which channels are connected             *
C*    ISTCH -Start channel address to which first transducer output  *
C*           is connected                                            *
C*    NOCYC - The number of program cycles                           *
C*    IN1   - The name of the input GEDAP data file                  *
C*    IOUT1 - The name of the output GEDAP data file                 *
C*                                                                   *
C*-------------------------------------------------------------------*
C*    GEDAP TRANSFER FILE EXAMPLE                                    *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*    FORMATTED OUTPUT(OUTPUT UNIT=LUL):                             *
C*-------------------------------------------------------------------*
C*                                                                   *
C*-------------------------------------------------------------------*
C*    LOG FILE OUTPUT(OUTPUT UNIT=LOG):                              *
C*-------------------------------------------------------------------*
C*                                                                   *
C*    IPSID  -The GEDAP program name                                 *
C*    IN1    -The name of the input GEDAP data files                 *
C*    IOUT1  -The name of the output GEDAP data files                *
C*                                                                   *
C*-------------------------------------------------------------------*
C*    GEDAP HEADER INPUT:                                            *
C*-------------------------------------------------------------------*
C*                                                                   *
C*                                                                   *
C*-------------------------------------------------------------------*
C*    GEDAP HEADER OUTPUT:                                           *
C*-------------------------------------------------------------------*
C*                                                                   *
C*    None                                                           *
C*                                                                   *
C*-------------------------------------------------------------------*
```

```
C*    SUBROUTINES AND FUNCTIONS CALLED:                                 *
C*----------------------------------------------------------------------*
C*                                                                      *
C*    Nil                                                               *
C*                                                                      *
C*----------------------------------------------------------------------*
C*    PROGRAM LIMITATIONS:                                              *
C*----------------------------------------------------------------------*
C*                                                                      *
C*----------------------------------------------------------------------*
C*    ERROR CODES:                                                      *
C*----------------------------------------------------------------------*
C*                                                                      *
C*----------------------------------------------------------------------*
C*    LOADING INSTRUCTIONS                                              *
C*----------------------------------------------------------------------*
C*                                                                      *
C*    TR,LXXXXX                                                         *
C*                                                                      *
C*----------------------------------------------------------------------*
C*    REPORT ANY BUGS IN THIS SPACE                                     *
C*----------------------------------------------------------------------*
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C***********************************************************************
C
C
C
C
```

```
C
C*************************************************************************
C***                      Module #1                                    ***********
C***  System required GEDAP DIMENSION and COMMON statements            ***********
C***                                                                   ***********
C*************************************************************************
C
      INTEGER JOBID(3),IPSID(3),ISMP(3)
      INTEGER IREST,IDSCN,VALUE,NDATA
      INTEGER INAM2(3)
      REAL X(512),Y(1028),XY(512)
      REAL ALF,BETA,DISCN
      COMMON/CFIX/IFIX(484),IN1(3),IOUT1(3),IOUTL(3)
      COMMON/IO/LUC,LUL,IPRINT,ISCI,ICRI,ISCO,ICRO
      COMMON/SYSIO/LOG,LUS,LU1,LU2,LU3
      COMMON/GDFMP/IDCBI(144),IDCBO(144),IBUF1(40),IPRAM(5),IRBUF(33)
      COMMON IDABU(128)
C
C
C*************************************************************************
C***                Module #2                                  **********
C***  System required GEDAP EQUIVALENCE statements            **********
C***                                                          **********
C*************************************************************************
C
      EQUIVALENCE(IFIX(2),JOBID(1))
      EQUIVALENCE(IFIX(6),VERNO)
      EQUIVALENCE(IFIX(12),IFORM)
      EQUIVALENCE(IFIX(13),ITYPE1)
      EQUIVALENCE(IFIX(14),ITYPE2)
      EQUIVALENCE(IFIX(20),N)
      EQUIVALENCE(IFIX(25),DEL)
      EQUIVALENCE(IFIX(60),SCFI)
      EQUIVALENCE(IFIX(260),TR)
C
C
C*************************************************************************
C***                    Module #3                                ***********
C***  System required GEDAP initialization statements           ***********
C***                                                            ***********
C*************************************************************************
C
C*-------------------------------------------------------------------------*
C*    Insert program name and version number                               *
```

```
C*-------------------------------------------------------------------*
      DATA IPSID/2HPD,2HSA,2H1 /
      DATA INDI,INDO,ICNTR,NOCYC/1,1,1,1/
      DATA INAM2/2HDI,2HSP,2H4 /
C     CALL GETLU(IER)
C     IF(IER.LT.0)GO TO 999
C     VERNO=1.1
C     WRITE(LOC,994)IPSID,VERNO
C 994 FORMAT(" GEDAP PROGRAM: ",3A2," VERSION:",F7.2)
C
C*-------------------------------------------------------------------*
C*    Obtain the number of program cycles                            *
C*-------------------------------------------------------------------*
C     IF (IPRINT .EQ. 1) WRITE(LUC,995)
C 995 FORMAT(" ENTER NUMBER OF PROGRAM CYCLES _")
C     READ(LUC,*) NOCYC
C
C
 1000 CONTINUE
C*-------------------------------------------------------------------*
C                                                                    *
C     Input parameters for Scan and Display                          *
C                                                                    *
C*-------------------------------------------------------------------*
      IF(VALUE.EQ.0) 100,160
  100 WRITE(1,7090)
 7090 FORMAT(1X,"ENTER NUMBER OF DATA SAMPLES IN THE BUFFER")
      READ(1,*) NDATA
C
C*    Find power of 2 corresponding to NDATA
C
      L = 0
  406 IF(2**L.EQ.NDATA) 404,405
  405 L = L + 1
      IF(L.EQ.15) 410,406
  410 WRITE(1,4100)
 4100 FORMAT(1X,"NDATA IS NOT A POWER OF TWO")
      GO TO 100
C
C*    Define parameters passed to SON program DISP4
C
  404 IOP2 = L
      IOP1 = NDATA
C
```

```
C*     Pointer to data average in the buffer
C
       IPTR2 = 2*NDATA + 1
C
       WRITE(1,9077)
 9077  FORMAT(1X,"ENTER LINE STYLE FOR THE GRID")
       READ(1,*) IOP4
C*     Find length of buffer passed to SON
C
       ILEN = 4*NDATA + 8
       WRITE(1,7060)
 7060  FORMAT(1X,"ENTER FREQUENCY OF SIMULATED SIGNAL")
       READ(1,*) FH
       WRITE(1,7092)
 7092  FORMAT(1X,"ENTER NUMBER OF FILES TO BE CREATED")
       READ(1,*) NOFIL
       WRITE(1,8000)
 8000  FORMAT(5X,"ENTER DATA SCAN RATE(IN SCANS PER SECOND)")
       READ(1,*)ISRATE
       XI = ISRATE
       TP = 1./XI
       ITRESP = 100/ISRATE
       WRITE(1,8010)
 8010  FORMAT(5X,"ENTER DISPLAY UPDATE RATE (IN DISPLAYS PER SECOND)")
       READ(1,*) DISCN
       TDSPL = 100/DISCN
       IDSCN = TDSPL/ITRESP
       VALUE = IDSCN
       IDSCN = NDATA
       VALUE = NDATA
C-------------------------------------------------------------------*
C      Get parameters for first order smoothing filter              *
C                                                                   *
C-------------------------------------------------------------------*
       WRITE(1,8016)
 8016  FORMAT(5X,"ENTER PARAMETER ALPHA ( TYPICAL VALUE IN THE RANGE
      1 0.93 TO 0.97)")
       READ(1,*) ALF
       BETA = 1. - ALF

C-------------------------------------------------------------------*
C                                                                   *
C      Get Analog Input Port and Start channel number to which first*
C      transducer output is connected and compute corrsponding      *
```

&PDSA1 1982/202 11: 5

```
C     Real-time buffer pointer                                          *
C                                                                       *
C-----------------------------------------------------------------------*
 131  WRITE(1,8064)
 8064 FORMAT(1X,"ENTER PORT NUMBER")
      READ(1,*) IPORT
      WRITE(1,8065)
 8065 FORMAT(1X,"ENTER START CHANNEL NUMBER")
      READ(1,*) ISTCH
      IPTR = (IPORT-1)*16 + ISTCH
C
      PI2 = 8.*ATAN(1.)
C*    Compute smoothing window function and store its value after       *
C*    ACF buffer that is from Y(NDATA + 1) onwards                      *
C
      XD = NDATA
      DO 149 I = 1,NDATA
         XI = FLOAT(I-1)
         Y(NDATA + I) = 0.5*(1. - XI/XD)
 149     CONTINUE
C-----------------------------------------------------------------------*
C                                                                       *
C    Clear CRT screen                                                   *
C                                                                       *
C-----------------------------------------------------------------------*
 132  DO 140 I = 1,25
         WRITE(1,8080)
 140     CONTINUE
 8080 FORMAT(" ")
C-----------------------------------------------------------------------*
C                                                                       *
C      Lock the program to the memory                                   *
C                                                                       *
C-----------------------------------------------------------------------*
      ICODE = 22
      ILOCK = 1
      CALL EXEC(ICODE,ILOCK)
C-----------------------------------------------------------------------*
C                                                                       *
C     Self-time-schedule                                                *
C                                                                       *
C-----------------------------------------------------------------------*
      ICODE = 12
      INAME = 0
```

```
      IRESL = 1
      IMULT = ITRESP
      IOFST = -1
      CALL EXEC(ICODE,INAME,IRESL,IMULT,IOFST)
C-----------------------------------------------------------------------*
C                                                                       *
C     Self-terminate with saving of resources                           *
C                                                                       *
C-----------------------------------------------------------------------*
  150 ICODE = 6
      INAME = 0
      INUMB = 1
      CALL EXEC(6,0,1)
C-----------------------------------------------------------------------*
C***********************************************************************
C****                                                          ********
C****    START OF ON-LINE PROCEDURE                            ********
C****                                                          ********
C***********************************************************************
C-----------------------------------------------------------------------*
C                                                                       *
C     Find average of the data by first order smoothing and also        *
C     compute auto-correlation function                                 *
C                                                                       *
C*----------------------------------------------------------------------*
C
C*    Read datum from real-time buffer and find average of the data
C
 160  DATA = FLOAT(IDABU(IPTR))/1000.
C160  XN = IN
C     DATA = 10.0*SIN(PI2*XN*TP*FH) + 2.*(URAN(1) - 0.5)
C     IN = IN + 1
C     IF(IN.EQ.32767) THEN IN = 0
      AVE = ALF*AVE + BETA*DATA
C
C*    Push all previous NDATA-1 samples one location down on the stack  *
C*    and put the recent sample on the top of the stack                 *
C
      DO 170 I = NDATA,2,-1
         X(I) = X(I-1)
 170     CONTINUE
      X(1) = DATA
C
C*    Compute auto-correlation function
```

```
C
      DO 180 I = 1,NDATA
         Y(I) = ALF*Y(I) + BETA*X(1)*X(I)
 180     CONTINUE
C
C     Decrement Display counter and branch if not zero
      IDSCN = IDSCN - 1
      IF(IDSCN.EQ.0) 171,150
C                                                                      *
C     If it is time to to compute FFT of auto-correlation function     *
C     then schedule the program DISP2 along with passage of ACF and    *
C     WINDOW buffer and average data value                             *
C                                                                      *
 171  ICODE = 10
      Y(IPTR2) = AVE
      IOP3 = INDO
      INDO = INDO + 2
      CALL EXEC(ICODE,INAM2,IOP1,IOP2,IOP3,IOP4,IOP5,Y,ILEN)
      IDSCN = VALUE
      NOFIL = NOFIL - 1
      IF(NOFIL.NE.0) 150,600
600   CALL EXEC(7)
      END
      END$
C*****   END OF ON-LINE PROCEDURE                                      *****
C*****                                                                 ******
C**********************************************************************
```

```
C
FTN4
      PROGRAM DISP4(),VERSION 1.0 -  4 JUNE  1982
C
C*****************************************************************************
C*****************+++++++++++++++++++++++++++++++++*****************************
C***************+                                   :*************************
C***************+       PROGRAM DISP4               +*************************
C***************+                                   :*************************
C*****************+++++++++++++++++++++++++++++++++*****************************
C*                                                                            *
C*----------------------------------------------------------------------------*
C*----------------------------------------------------------------------------*
C*    PROGRAM DESCRIPTION .                                                   *
C*----------------------------------------------------------------------------*
C*                                                                            *
C*    1 - Retrieve parameter values and ACF buffer passed by Father           *
C*        scheduling program                                                  *
C*    2 - Smooth the ACF                                                      *
C*    3 - Find Fourier transform of smoothed ACF                              *
C*    4 - Display power spectrum on the terminal screen                       *
C*                                                                            *
C*                                                                            *
C*----------------------------------------------------------------------------*
C*    CONVERSATIONAL INPUT(INPUT UNIT=LUC):                                   *
C*----------------------------------------------------------------------------*
C*    Nil                                                                     *
C*----------------------------------------------------------------------------*
C*    GEDAP TRANSFER FILE EXAMPLE                                             *
C*----------------------------------------------------------------------------*
C*    Display of power spectrum on the terminal screen and output             *
C*    GEDAP file of ACF and power spectrum                                    *
C*----------------------------------------------------------------------------*
C*    FORMATTED OUTPUT(OUTPUT UNIT=LUL):                                      *
C*----------------------------------------------------------------------------*
C*                                                                            *
C*----------------------------------------------------------------------------*
C*    LOG FILE OUTPUT(OUTPUT UNIT=LOG):                                       *
C*----------------------------------------------------------------------------*
C*                                                                            *
C*    IPSID  -The GEDAP program name                                          *
C*    IN1    -The name of the input GEDAP data files                          *
C*    IOUT1  -The name of the output GEDAP data files                         *
C*                                                                            *
```

```
C*-----------------------------------------------------------------------*
C*    GEDAP HEADER INPUT:                                                *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    GEDAP HEADER OUTPUT:                                               *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    SUBROUTINES AND FUNCTIONS CALLED:                                  *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    PROGRAM LIMITATIONS:                                               *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    ERROR CODES:                                                       *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    LOADING INSTRUCTIONS                                               *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*    TR,LXXXXX                                                          *
C*                                                                       *
C*-----------------------------------------------------------------------*
C*    REPORT ANY BUGS IN THIS SPACE                                      *
C*-----------------------------------------------------------------------*
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*                                                                       *
```

```
C*                                                                      *
C*                                                                      *
C*                                                                      *
C************************************************************************
C
C
C
C
C
C************************************************************************
C***                        Module #1                                 ***********
C***  System required GEDAP DIMENSION and COMMON statements           ***********
C***                                                                  ***********
C************************************************************************
C
      INTEGER JOBID(3),IPSID(3),ISMP(3),IPARM(5),ICRSR(17),ICRSR2(2)
      REAL ACF(258),A(129),XX(4),YY(4),X(129),B(129)
      COMPLEX CA(1)
      DIMENSION Y(516)
      COMMON/CFIX/IFIX(484),IN1(3),IOUT1(3),IOUTL(3)
      COMMON/IO/LUC,LUL,IPRINT,ISCI,ICRI,ISCO,ICRO
      COMMON/SYSIO/LOG,LUS,LU1,LU2,LU3
      COMMON/GDFMP/IDCBI(144),IDCBO(144),IBUF1(40),IPRAM(5),IRBUF(33)
C
C
C************************************************************************
C***                Module #2                                 ***********
C***  System required GEDAP EQUIVALENCE statements            ***********
C***                                                          ***********
C************************************************************************
C
      EQUIVALENCE(IFIX(2),JOBID(1))
      EQUIVALENCE(IFIX(6),VERNO)
      EQUIVALENCE(IFIX(12),IFORM)
      EQUIVALENCE(IFIX(13),ITYPE1)
      EQUIVALENCE(IFIX(14),ITYPE2)
      EQUIVALENCE(IFIX(20),N)
      EQUIVALENCE(IFIX(25),DEL)
      EQUIVALENCE(IFIX(60),SCFI)
      EQUIVALENCE(IFIX(260),TR)
      EQUIVALENCE(ACF(1),CA(1))
C
C
C************************************************************************
```

```
C***                    Module #3                                  **********
C***  System required GEDAP initialization statements              **********
C***                                                               **********
C*******************************************************************************
C
C*-----------------------------------------------------------------------------*
C*    Insert program name and version number                                   *
C*-----------------------------------------------------------------------------*
      DATA IPSID/2HDI,2HSP,2H4 /
      DATA INDI,INDO,ICNTR,NOCYC/1,1,1,1/
      DATA XX/30.0,30.0,0.0,0.0/
      DATA YY/0.0,15.0,15.0,0.0/
      DATA ICRSR/015452B,060465B,030551B,031152B,030553B,030154B,
     1 032555B,030156B,030465B,067461B,070061B,070461B,071061B,
     1 071564B,072461B,073061B,073440B/
       DATA ICRSR2/015452B,060503B/
      IF(ICP1.EQ.0)500,511
  500 LUC = 1
      LUL = 1
      LOG = 1
      GO TO 511
C
C*    Self terminate with saving of resources
C
  490 CALL EXEC(6,0,1)
C
C*    Retrive parameters passed by father program
C
  511 CALL RMPAR(IPARM)
      NDATA = IPARM(1)
      IL = IPARM(2)
      INDO = IPARM(3)
      IPL = IPARM(4)
      IN = NDATA/2 + 1
C
C*    Retrieve the ACF buffer passed by father program PSDAN
C
C*    Lock program to the memory
C
C     CALL EXEC(22,1)
C
C*    Retrieve ACF buffer
C
      ICODE = 14
```

```
      ILEN = (NDATA*2 + 4)*2
      IRWCD = 1
      CALL EXEC(ICODE,IRWCD,Y,ILEN)
C
C*    Multiply ACF function by smoothing window
C
      SQR = Y(NDATA*2 + 1)**2
      DO 100 I = 1,NDATA
         ACF(I) = Y(I + NDATA)*(Y(I) -SQR)
  100    CONTINUE
C
C*    Take FFT of smoothed ACF to find power spectrum
C
      CALL FFT(CA,(IL-1),-1)
      CALL FFTR(CA,NDATA)
C
C*    Find amplitude of each frequency component in power spectrum
C
      DO 120 I = 1,IN
         A(I) = 2.*(SQRT(ACF(2*I)**2 + ACF(2*I - 1)**2))
  120    CONTINUE
C
C*    Scale all the x and y coordinates of points to be plotted
C
C
C*    Find maximum of array A(I)
C
      AMAX = 0.0
      DO 130 I = 1,IN
      IF(A(I).GT.AMAX)  AMAX = A(I)
  130 CONTINUE
      AMAX = AMAX*1.1
C
C*    Do scaling
C
      XN = FLOAT(IN)
      DO 140 I = 1,IN
      B(I) = A(I)*15.0/AMAX
  140 X(I) =  FLOAT(I - 1)*30./XN
C
C*    Initialize the terminal
C
      CALL ZBEGN
      CALL ZDINT(1,0,IERR)
```

```
      CALL ZVIEW(0.0,30.0,0.0,15.0)
      CALL ZDLIM(0.0,30.0,0.0,15.0)
      CALL ZASPK(1.0,1.0)
C
C*    Draw the rectangle
C
      CALL ZMOVE(0.0,0.0)
C
C*    Draw the grid lines
C
      CALL ZLSTL(1)
      DO 155 I = 1,4
  155    CALL ZDRAW(XX(I),YY(I))
        CALL ZLSTL(IPL)
       DO 156 I = 1,4
            YI = 3.*FLOAT(I)
          CALL ZMOVE(0.0,YI)
  156      CALL ZDRAW(30.0,YI)
          DO 157 I = 1,9
              YI = 3.*I
             CALL ZMOVE(YI,0.0)
  157         CALL ZDRAW(YI,15.0)
      CALL ZMOVE(X(1),B(1))
      CALL ZLSTL(1)
      DO 150 I = 2,IN
  150    CALL ZDRAW(X(I),B(I))
         CALL ZEND
      IFORM=2
      ITYPE1=11
      ITYPE2=0
      N=NDATA
      DEL = 0.1
      SCFI = 1.0
      CALL GDOUT(INDO,IER)
      IF(IER.LT.0)GO TO 999
C
C
C*********************************************************************
C***            Module #9                  **************************
C***  Create and output GEDAP header       **************************
C***     file then open data file          **************************
C***                                       **************************
C*********************************************************************
C
```

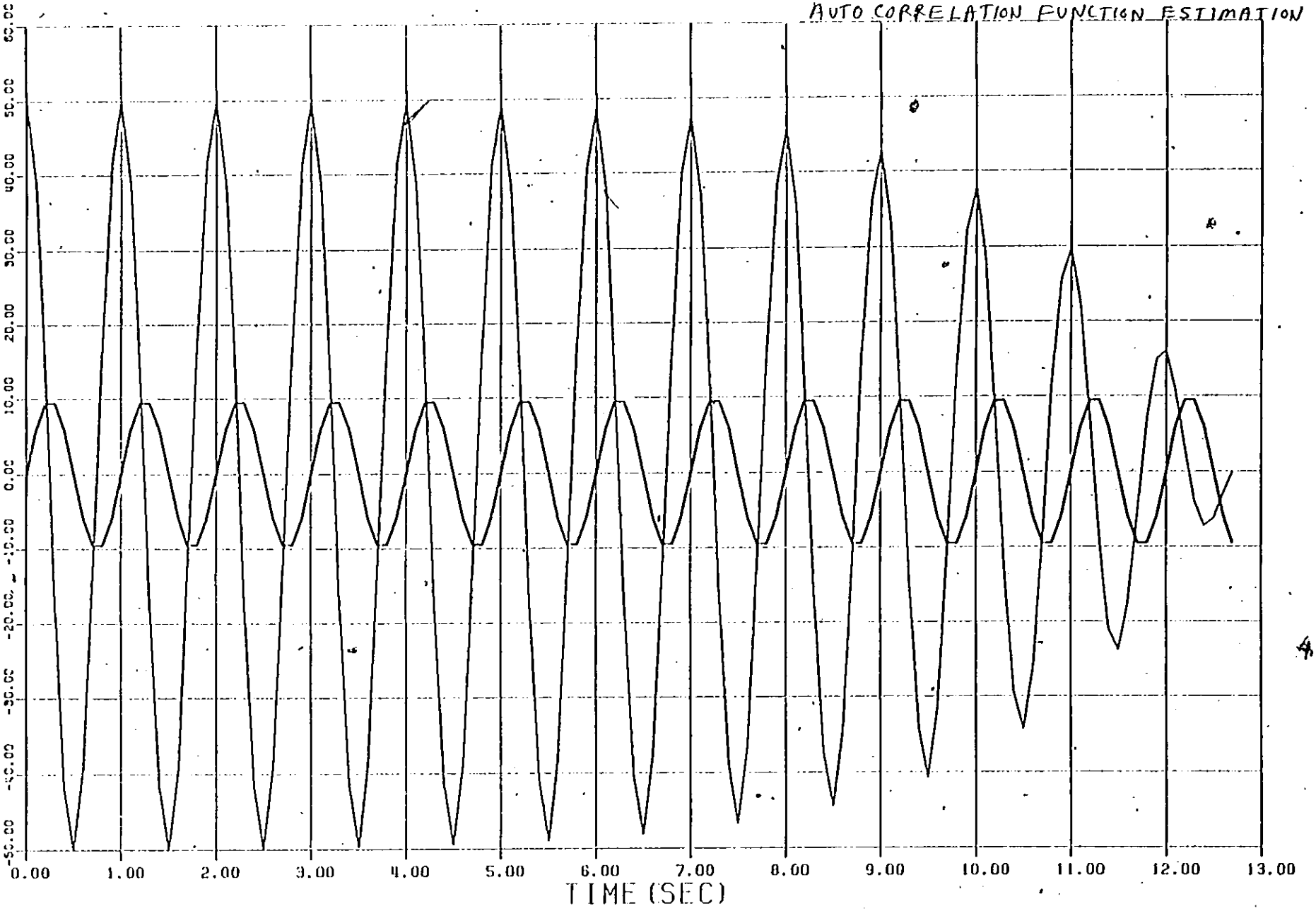
```
      DO 997 IPRGNM=1,3
  997 JOBID(IPRGNM)=IPSID(IPRGNM)
      CALL HDROT(IFORM,ITYPE1,ITYPE2,IER)
      IF(IER.LT.0)GO TO 999
C
C
C*************************************************************************
C***       Module #10                                  *********************
C***  Output data to GEDAP data file                   *********************
C***                                                   *********************
C*************************************************************************
C
C*-----------------------------------------------------------------------*
      CALL WFOUT(ACF,2*NDATA,IER)
      IF(IER.LT.0)GO TO 999
C
C     WRITE(LOG,998) IOUT1,ISCO,ICRO
C 998 FORMAT(" OUTPUT FILE=",3A2,":",A2,":",I5)
C
      INDO=INDO+1
C
      N = NDATA/2 + 1
      DEL = 10./NDATA
C
      SCFI = 1.0
      CALL GDOUT(INDO,IER)
      IF(IER.LT.0) GO TO 999
      DO 9971 I = 1,3
 9971 IOBID(I) = IPSID(I)
      CALL HDROT(IFORM,ITYPE1,ITYPE2,IER)
      IF(IER.LT.0) GO TO 999
C
C*    WRITE SPECTRUM INTO GEDAP FILE
C
      CALL WFOUT(A,2*N,IER)
      IF(IER.LT.0) GO TO 999
C
      GO TO 490
  999 CALL GDCLL(IER)
      END
C
      SUBROUTINE FNAME(IND2,IFILE,IER),VERSION 1.5 - 17 JUNE 1980
C
C*************************************************************************
```

```
C*************+++++++++++++++++++++++++++++++++++**************************
C*************+                                 +**************************
C*************+       SUBROUTINE FNAME          +**************************
C*************+                                 +**************************
C*************+++++++++++++++++++++++++++++++++++**************************
C**************************************************************************
C*                                                                        *
C*-----------------------------------------------------------------------*
C*    SUBROUTINE SUMMARY                                                  *
C*-----------------------------------------------------------------------*
C*                                                                        *
C*         This routine obtains an FMP file name                          *
C*                                                                        *
C*-----------------------------------------------------------------------*
C*    SUBROUTINE DESCRIPTION                                              *
C*-----------------------------------------------------------------------*
C*                                                                        *
C*         This routine obtains an FMP input or output file name          *
C*                                                                        *
C*-----------------------------------------------------------------------*
C     INPUT PARAMETERS:                                                   *
C*-----------------------------------------------------------------------*
C*         IND2  = 1  obtain input file name(IN1)                         *
C*               = 2  obtain output file name(IOUT1)                      *
C*             **= 3  obtain list file name(IOUTL)**no longer used        *
C*                                                                        *
C*-----------------------------------------------------------------------*
C*    OUTPUT PARAMETERS:                                                  *
C*-----------------------------------------------------------------------*
C*                                                                        *
C*         IFILE - GEDAP data file name                                   *
C*                                                                        *
C*-----------------------------------------------------------------------*
C*    SUBROUTINES AND FUNCTIONS CALLED:                                   *
C*-----------------------------------------------------------------------*
C*                                                                        *
C*         REIO                                                           *
C*         COLON                                                          *
C*         PARSE                                                          *
C*         SHFNM                                                          *
C*                                                                        *
C*-----------------------------------------------------------------------*
C*    ERROR CODES:                                                        *
C*-----------------------------------------------------------------------*
```

```
C*                                                                    *
C**********************************************************************
C
      COMMON/IO/LUC,LUL,IPRINT,ISCI,ICRI,ISCO,ICRO
      COMMON/GDFMP/IDCBI(144),IDCBO(144),IBUF1(40),IPRAM(5),IRBUF(33)
      INTEGER IFILE(3),IREG(2),OFILE(3)
      EQUIVALENCE(IREG,REG)
C*---------------------------------------------------------------------
C*    DEFINE HASH CHARACTER
C*---------------------------------------------------------------------
      DATA IHASH/021400B/
C*---------------------------------------------------------------------
C*    SET FILE NAME
C*---------------------------------------------------------------------
      OFILE(1)=2HDA
      OFILE(2)=2HT0
      OFILE(3)=2H1
C*---------------------------------------------------------------------
C*    SHIFT CHARACTERS RIGHT
C*---------------------------------------------------------------------
      CALL SHFNM(OFILE,IFILE)
C*---------------------------------------------------------------------
C*    ADD FILE IDICATOR CHARACTER
C*---------------------------------------------------------------------
      IFILE(1)=IFILE(1)+IHASH
      IER=0
      RETURN
      END
      END$
```

```
C*                                                                      *
C************************************************************************
C
      COMMON/IO/LUC,LUL,IPRINT,ISCI,ICRI,ISCO,ICRO
      COMMON/GDFMP/IDCBI(144),IDCBO(144),IBUF1(40),IPRAM(5),IRBUF(33)
      INTEGER IFILE(3),IREG(2),OFILE(3)
      EQUIVALENCE(IREG,REG)
C*----------------------------------------------------------------------
C*    DEFINE HASH CHARACTER
C*----------------------------------------------------------------------
      DATA IHASH/021400B/
C*----------------------------------------------------------------------
C*    SET FILE NAME
C*----------------------------------------------------------------------
      OFILE(1)=2HDA
      OFILE(2)=2HT0
      OFILE(3)=2H1
C*----------------------------------------------------------------------
C*    SHIFT CHARACTERS RIGHT
C*----------------------------------------------------------------------
      CALL SHFNM(OFILE,IFILE)
C*----------------------------------------------------------------------
C*    ADD FILE IDICATOR CHARACTER
C*----------------------------------------------------------------------
      IFILE(1)=IFILE(1)+IHASH
      IER=0
      RETURN
      END
      END$
```

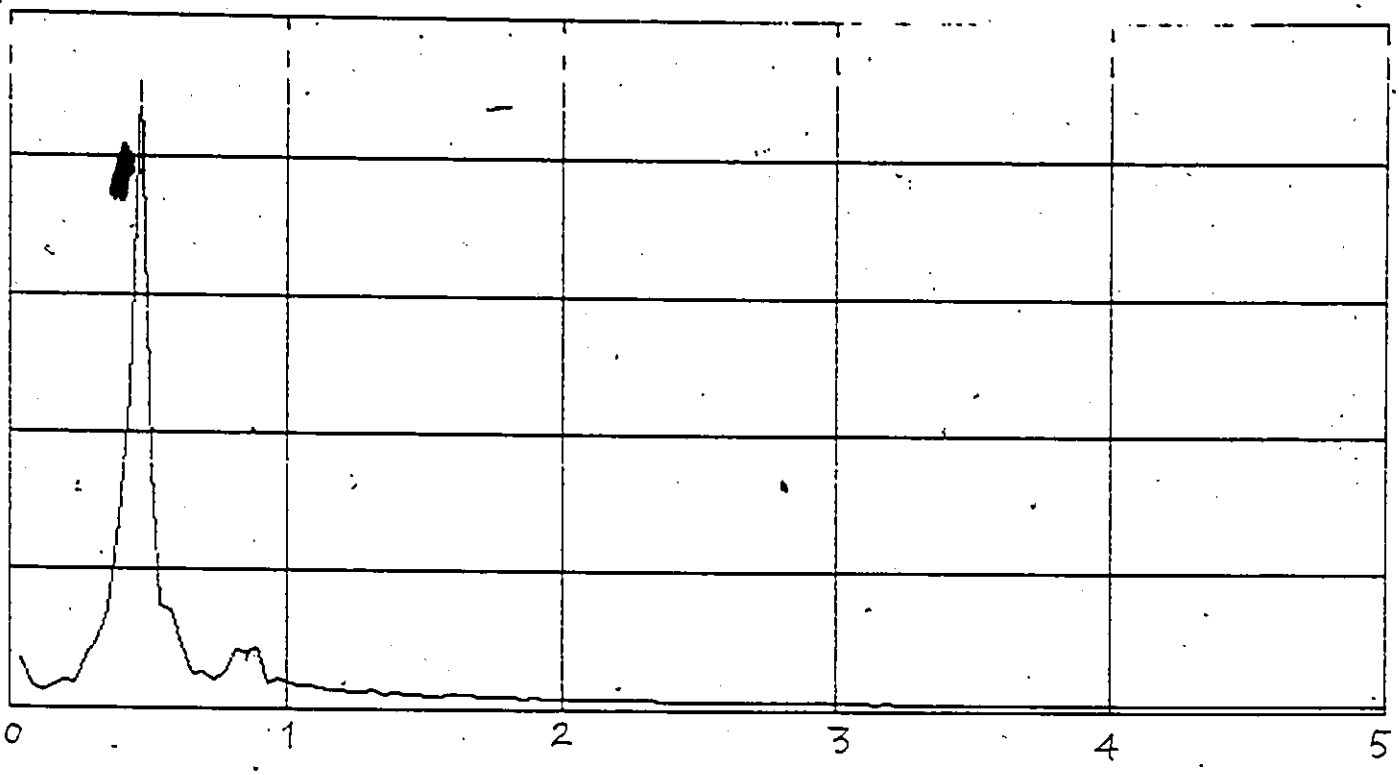
I/P SIGNAL ← ACF FIRST 128 samples



THICK LINE GRAPH I/P SIGNAL

THIN LINE - ITS AUTO-COVARIANCE FUNCTION

FIG 14



.4306

FIG 15

ESTIMATED POWER SPECTRUM FOR REGULAR WAVES

FREQUENCY OF GENERATED WAVE AND ITS MEASURED VALUE COINCIDE