CR-114914

R71-014

COMPUTER PROGRAM DOCUMENTATION

SYSTID, System Time-Domain Simulation Program

Contract NAS9-10831

February 1971

Prepared for:

Space Electronics System Division
National Aeronautics Space Administration
Manned Spacecraft Center
Houston, Texas 77058

By:

Ernest Freeman
Michael Fashano

SYSTEMS ASSOCIATES

444 West Ocean Boulevard
Long Beach, California 90802
Telephone 213/435-8282

MSC PROGRAM, SYSTID

System Time Domain Simulation Program

(Apollo Telecommunications link Simulation)

Prepared By:

Ernest Freeman
Programming Staff
Systems Associates, Inc.

Approved By:

Michael Fashano
Software Design Manager
Systems Associates, Inc.

Robert J. Rechter
Program Manager
Systems Associates, Inc.

C. T. Dawson
Space Electronics System Division
Code EE8, NASA/MSC

Scientific Computing Branch
Completion and Analysis Division, NASA/MSC
Houston, Texas 77058

COMPUTER PROGRAM ABSTRACT

Date: February 1971

Facility: MSC

Title: SYSTID, Time Domain Simulation Program

Language: FORTRAN V

Key Words: Time Domain, Sampled Data, Link Simulation, Digital Filter, z-Transform

Computer Type: 1108

Total Cost: Overall Study, Plus Program: $23,881.

ABSTRACT

The SYSTID Computer Program allows the user to achieve time domain simulations of telecommunications links, such as the Apollo links. The program is written in FORTRAN V, and is operational on the NASA/MSC UNIVAC 1108 system. The SYSTID program has been designed such that the input data set required of the user is minimal. This program characteristic is achieved through use of a language processor which translates simple English-language user commands and link element descriptions and topology into the FORTRAN code necessary to establish a digital filter equivalent of each link. The program's time domain simulation technique employs the bilinear z-transform, to reduce run time, and minimize the errors commonly associated with sampled data representations. In addition a model library has been implemented so that commonly encountered telecommunications link elements can be called up by the user directly, without detailed user coding. These library models include link elements such as modulators, demodulators, filters, limiters, and other elements. The system simulation can be excited by a number of time functions, including those in its library, and the system's time response can be computed in listed or graphical output format, at any point. In addition, postprocessing routines can be applied to SYSTID output to convert the computed time response into frequency domain responses, and perform other output analysis.

TABLE OF CONTENTS

Section			Page
5 (con't)	5.3.8	EINES	5-5
	5.3.9	EQUNIT	5-5
	5.3.10	ERECTS	5-5
	5.3.11	ERRMSG	5-5
	5.3.12	EDIT1	5-5
	5.3.13	GREBE	5-5
	5.3.14	GWIN	5-5
	5.3.15	INCLUD	5-6
	5.3.16	LIB 003	5-6
	5.3.17	LIB 004	5-6
	5.3.18	LISTIO	5-6
	5.3.19	LISTIT	5-6
	5.3.20	LIT	5-6
	5.3.21	SYSTID	5-6
	5.3.22	MORGUE	5-6
	5.3.23	NADINE	5-7
	5.3.24	NAMLST	5-7
	5.3.25	NTAB S	5-7
	5.3.26	PGF/ISD	5-7
	5.3.27	PCF/MSC	5-7
	5.3.28	PREAMB	5-7
	5.3.29	PUTOUT	5-7
	5.3.30	QUEUE	5-7
	5.3.31	RECTUS	5-8
	5.3.32	SETUP	5-8
	5.3.33	SPORU	5-8
	5.3.34	TAPONE	5-8
	5.3.35	TAPTWO	5-8
	5.3.36	TAP 3	5-8
	5.3.37	THOR	5-8
	5.3.38	TITLE	5-8
	5.3.39	USRELT	5-9
	5.3.40	V	5-9
	5.3.41	ZWEI	5-9
	5.3.42	PARAM	5-9
	5.3.43	PROCO5	5-9
	5.3.44	QUARTR	5-9
	5.3.45	SUB 1	5-9
	5.4	Program Listing	5-9
	5.5	Sample Input/Output	5-9
	5.6	Bibliography and References	5-36
			A-1
Appendix			
A		THEORETICAL BASIS FOR SYSTID	A-1
	A.1	Theoretical Introduction	A-1
	A.2	Applications of Sampled Data Techniques in the SYSTID Program	A-5

Appendix			Page
	SYSTID MODEL DESCRIPTION AND THEORY		B-1
	B.1.0	Model Description	B-1
	B.2.0	Model Node Definitions	B-3
	B.3.0	Listing of Internal Variables and Systid Library Models	B-4
	B.3.1	Signal Generators	B-4
	B.3.1.1	Transcendental Functions	B-4
	B.3.1.2	Square Wave	B-4
	B.3.1.3	Pulse Generator	B-4
	B.3.1.4	Arbitrary Function or Non-Linearity	B-5
	B.3.1.5	Periodic Function Generator	B-5
	B.3.1.6	Gaussian Noise Generator	B-5
	B.3.2	Modulators	
	B.3.2.1	Amplitude Modualtor (AMMOD and RAMMOD)	B-9
	B.3.2.2	Linear Frequency Modulator (FMMOD and REMMOD)	B-10
	B.3.2.3	Linear Phase Modulator (PMMOD) and RPMMOD)	B-13
	B.3.2.4	Delta Modulation (DELMOD)	B-14
	B.3.2.5	Multi-Level Coding — M-ary Codes (MLTPCM)	B-16
	B.3.2.6	Analog/Digital Converter (ATOD)	B-19
	B.3.3	Demodulators and Decoders	B-21
	B.3.3.1	Amplitude Demodulator (AMDEMOD) and RAMDEMOD)	B-22
	B.3.3.2	Frequency Demodulator(FMDEMOD) and RFMDEMOD)	B-24
	B.3.3.3	Phase Demodulator (PMDEMM and RFPDEM)	B-26
	B.3.3.4	Matched Filter (MFILTR)	B-
	B.3.3.5	Frequency Demodulator with Feedback (FMFB)	B-30
	B.3.4	Filters	B-33
	B.3.5.	Miscellaneous Models	B-37
	B.3.5.1	Hard Limiters	B-37
	B.3.5.2	Soft Limiters	B-37
C	THEORETICAL BASIS FOR FILTER MODELS		C-1
	C.1	Transfer Function Response from Pole-Zero Models	C-1
	C.2	Generation of Filter Prototype Transfer Functions	C-2
	C.2.1	Ideal Filter	C-2
	C.2.2	Butterworth Filters	C-4
	C.2.3	Chebyshev Filters	C-6

SECTION 1.0

INTRODUCTION

The computer-aided analysis of various systems using time domain and other digital simulation techniques is well-trodden ground from a theorectical point of view. However, the practical aspects of such machine simulation have not been as highly developed. In particular, the fairly recent availability of high speed digital computers has increased the cost effectiveness of employing such digital time domain simulation, a technique whose analytical advantages have long been recognized. An advanced digital simulation program is described in this documentation which can be applied to telecommunications system simulation.

SYSTID, the SAI time domain simulation language, allows the us er to describe a telecommunication system's topology and element characteristics in simple English style text. Various system elements are then generated on a one-time basis and stored for subsequent use. The nature of the exciting signals, each transfer element in the system, and the desired locations of output response are all simply described in the SYSTID language.

Input descriptors to the program, defining arbitrary link elements, include a verbal and numeric parameter set and appropriate nodal connection data. The SYSTID program then generates appropriate FORTRAN routines to implement the sampled data equivalence to the desired continuous system transfer functions.

The SYSTID input is presently via punched card format, with tabulated and/or line printer and/or CalComp plotted output.

A number of typical Apollo-related communications system elements can be modeled, and stored in a program library, for subsequent simulation and analysis.

SECTION 2.0

PROGRAM DESCRIPTION

2.1 GENERAL DESCRIPTION

The SYSTID time domain simulation program can more accurately be described as a language processor, which in turn employs the sampled data, i.e. z-transform techniques of the SAI SAMDAT program, to allow user-oriented simulation of telecommunications systems. The SYSTID program can be applied more generally than to merely the set of telecommunications links; any continuous system representation can be achieved. The main thrust of the SYSTID Software Design has taken place in optimizing the user-program interface. The major program characteristics can be summarized by the following items:

Simple user-oriented input language, which allows user-selectable degrees of complexity. These complexity levels range from the simplest, where internally stored telecommunications link element models are used, i.e., the user essentially writes down "phase modulator", and the program employs a previously described internally stored model, appropriately interconnected to other link elements as defined by the user. Alternately, the user may choose to redefine certain of each link element's describing parameter set. Thus, in the limit, the user may externally specify all of the describing parameters for each link element, for each system element.

Flexibility in system topological description, allowing this topology information to be initialized or modified in successive program runs.

A wide range of input signal excitation can be specified, and output data, in listed and/or CalComp-plotted format, computed at any point in the system.

A faithful representation of system non-linearities, such as a limiter, can be achieved. This contrasts with alternate system representations often encountered, such as polonomial expansions, etc.

Utilization of the bi-linear z-transform and translation of high carrier frequencies to baseband frequencies to minimize aliasing error and yet achieve reasonable run-times.

Thus the SYSTID development has emphasized flexibility with respect to simulated system characteristics, yet minimizes input data requirements by means of appropriate stored telecommunications system element models. These models typically include common system elements such as angle modulators and demodulators, bandpass and lowpass filters, etc.

An important program feature is an initialization capability to minimize repetative user input data requirements. For example, a phase modulator with a linear phase versus frequency range of plus or minus 3 radians and a sensitivity of 1 radian/volt, or a predetection filter transfer function given by a 4th order Butterworth-Thomson characteristic are examples of such initialization. These "standard" characteristics would be modified only upon user command, for those analysis where they constitute a necessary parametric variation.

It is expected that with the supplied SYSTID library models and functions and a few user constructed models that most systems will involve little more than defining the topological structure, the inputs and the desired outputs. References to models and functions produce FORTRAN subroutine or function calls in the generated symbolic program. The use of subroutines and functions vastly reduces the amount of work the processor must perform. The subroutines are manipulated by the utility routines of the computer operating system (for example CUR). The technique saves a great deal of time and considerably reduces the complexity of the overall system. Certain types of devices are not easily constructed in the SYSTID language, for example the SYSTID library routines. Such devices can be added to the SYSTID system by modeling them directly in FORTRAN or BAL and informing the SYSTID "dictionary" about their existence, a simple operation not requiring use of the SYSTID processor. The generation of FORTRAN subroutines for models will result in an efficient simulation program. A simulation run will progress through several distinct stages; input-setup, simulation, output, and post processing. Since the simulation will generally consume the greatest amount of computer time, the system is designed to produce an efficient simulation stage. The modular structure of the program (subroutines) and the breakdown of the simulation program into distinct stages will allow parts of the program to be overlayed in core in order to minimize the use of core, thus maximizing the size of the largest permitted simulation. However, no overlays in the simulation stage are permitted since this would increase computer run time considerably.

2.2 TECHNICAL DESCRIPTION

2.2.1 Introduction

Time domain simulation of systems has classically employed analog computation, mainly in the area of control systems analysis. With the advent of second generation digital computers (IBM 7094) for example, time domain simulation using digital computers advanced rapidly with the development of programs such as MIDAS, MIMIC, DSL/90, CSMP, ECAP, SCEPTRE, etc. These programs however, were designed with control systems or circuit analysis in mind. Thus, the simulation programs which have been available in the past are not optimized for analysis of telecommunications systems, systems which possess characteristics not encountered in control systems or circuits.

It is only with the availability of large scale digital machines, such as the UNIVAC 1108, that economic considerations favor digital computer simulation as opposed to analog simulation. It will be recognized that the costs of analog versus digital simulation cannot be weighed on a one to one basis. For instance, analog simulation requires significant set-up and check-out time for problem initialization, with additional time for modifications to the original situation, but features extremely low unit run costs even for wide-bandwidth systems. In this respect, a digital simulation, using a well designed program, requires minimum set-up time, very limited initial checking, and negligible additional time penalties for parametric or topological variations but at higher hourly costs. Further, degradation of the electronic elements of the analog computer may create large solution error, further limiting inherent analog equipment fidelity. However, for certain situations use of an analog or hybrid computer simulation may be appropriate. In general, it would appear that use of digital computer simulation techniques offers the most attractive cost-performance characteristics.

Simple acceptance of the worth of the digital computer for system simulation does not lead directly to adoption of an optimum digital simulation technique. Section 2.2.2 will touch on some of the several digital simulation techniques which are available to the user.

In particular, one unique characteristic of telecommunications systems analysis impacts a time domain simulation badly. This characteristic is the generally large ratio between the RF carrier and the baseband frequencies; i.e., one must compute the system's response for a large number of carrier cycles while allowing propagation of much slower baseband excitation. Appendix A will illustrate the techniques employed in SYSTID to minimize this difficulty.

2.2.2 Comparative Analysis Techniques

There are several digital computer-aided techniques which can be implemented for telecommunications systems analysis [1] [2] [8] [9] [82] [84] [85] [97]. These include signal power/noise spectral density frequency domain and autocorrelation techniques as well as the time domain approach. The use of the spectral density approach is often of interest for certain limited applications, where the phase characteristic of the telecommunications link elements is not considered, i.e., any spectral density approach must necessarily be insensitive to phase transfer functions. This is a major weakness in systems analysis of high fidelity . . . The phase properties of real telecommunications links do contribute to performance degradation, and must be considered. The time domain approach does not have this limitation, and in addition, will inherently treat non-linear system element responses in a accurate, straight-forward fashion. The common time domain simulation failing of excessive computer run time is offset by the use of a high speed digital computer and the particular transform and carrier-to-baseband frequency translation approach which is employed in the SYSTID program.

2.2.3 Analysis

The theoretical basis for the SYSTID program is discussed in detail in appendix A. In addition, the extensive bibliography given in this document will illustrate some of the extensive work done in this area of time domain simulation. The techniques employed in the SYSTID program are based in part on earlier work done at Hughes Aircraft Company Space Systems Division by M. Fashano, W. Mayfield, and N. Wagner, and others [1] [2]. It will be recognized that SYSTID represents a significant program improvement due to its user-machine interface design, rather than to any fundamental algoritam developments. The sampled-data program aspects of SYSTID have been validated by many applications to various telecommunications links, such as Surveyor, Mariner Mars 1971, and Apollo.

2.2.4 Method of Solution

The SYSTID program is a two-phase processor consisting of a language processor (translator) and a library. When coupled with the UNIVAC EXEC II system, SYSTID becomes an easily used system simulation program entirely user-oriented.

The SYSTID processor accepts input decks written in the SYSTID language representative of systems or models to be used in systems. SYSTID then generates a symbolic FORTRAN V program (for systems) or subroutine (for models) which is then compiled by the FORTRAN V compiler. If the input describes a model, a temporary entry is made in the library dictionary for subsequent use. In either case, the symbolic routines are written into the Program Complex File (PCF) for subsequent use and interfacing with EXEC II.

A SYSTEM is a complete program which is to be executed to simulate a specific telecommunications link; i.e., is a model prefixed with system parameter definitions, input/output specifications, and post-processing declarations.

A MODEL is a subroutine which simulates the properties of a device in a telecommunications link. A MODEL is characterized by defining its topology and components.

The following will describe both the language processor and the sample-data techniques as applied to telecommunications analysis and simulation.

2.2.4.1 SYSTID Language Processor

The SYSTID Language Processor translates topological descriptions into a procedural method of solution based upon a fixed algorithm. This algorithm is best depicted by figure 2-1, the logical structure of input decomposition.

The input data for a model or system is decomposed into a set of linked tables which define the topological characteristic of the input. Once the tables are constructed, they are systematically scanned, starting at the left node name table entry "INPUT," until all expressions are solved up to and including the right node table entry "OUTPUT." The signal progress convention is from left node to right node; left being the input. Taps, depending on content, can be either inputs or outputs. The output of the device is passed to the next device through their common node.

The linked tables of figure 2-1 are built and searched by the routine depicted in figure 2-2. Whenever a tap is referenced by an expression, the "tap table" provides the necessary information for the reference. All addressing is relative to the input node of the model or system, at all levels. That is, all models generated by the SYSTID language processor become re-entrant, and each reference is independent of any other reference to the same model. With models referencing other models several times, this factor is of prime importance. The relative addressing is performed by a floating index which is set to the next available location by each model routine as it is entered so as to provide for its own storage requirements.

2.2.4.2 Sampled Data Modeling

The technique utilized in the SYSTID library for simulating continuous systems is the bi-linear z-transformation. The major advantage over the standard z-transform is that aliasing errors are eliminated, making possible the realization of commonly encountered functions whose response does not approach zero for high frequencies (e.g. high pass, bandstop) and allowing the reduction of required sampling frequency. Note that aliasing of the signals, however, is possible if the sampling rate is too low.

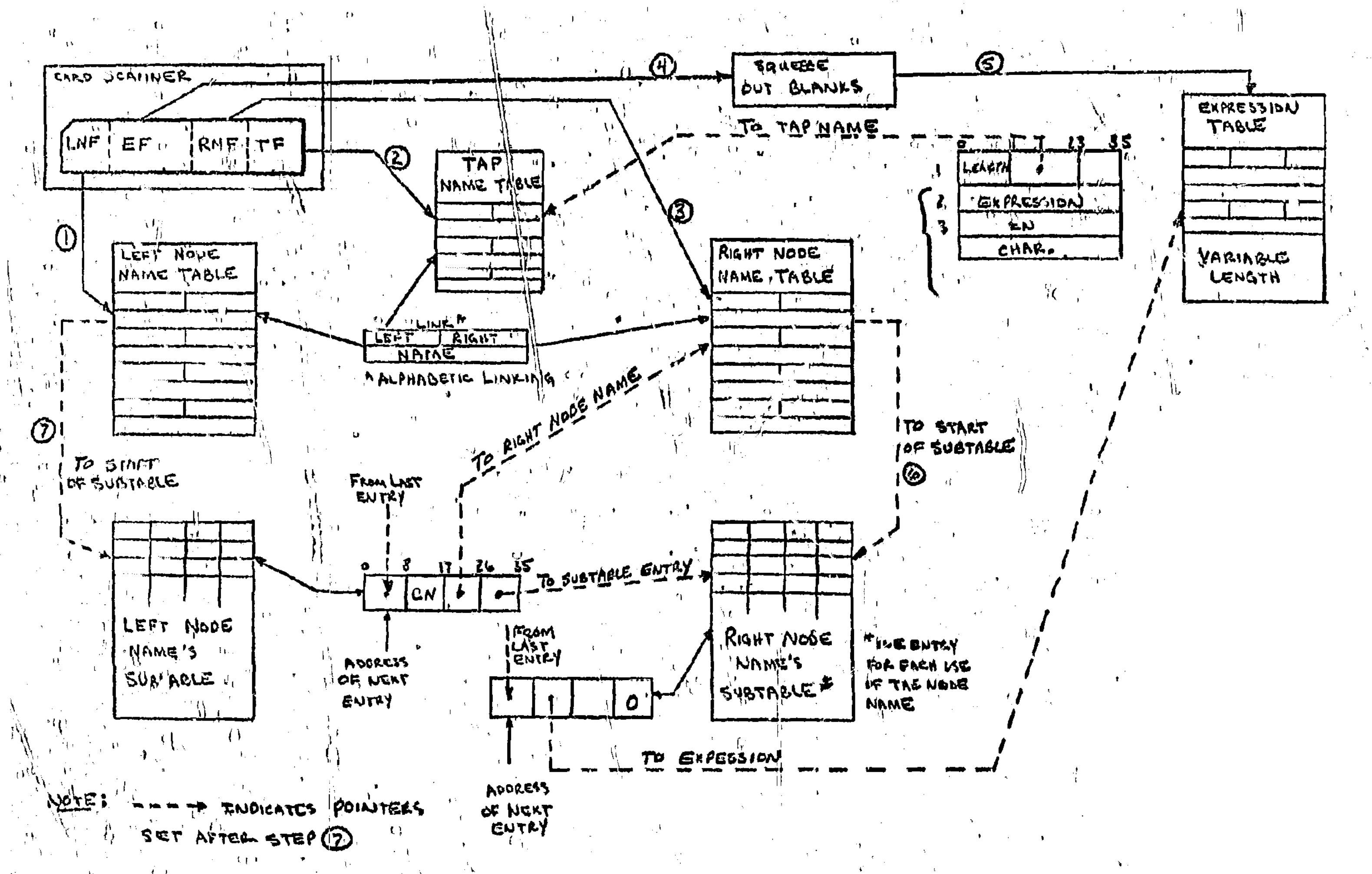


Figure 2-1. SYSTID Logical Structure of Input Decomposition.

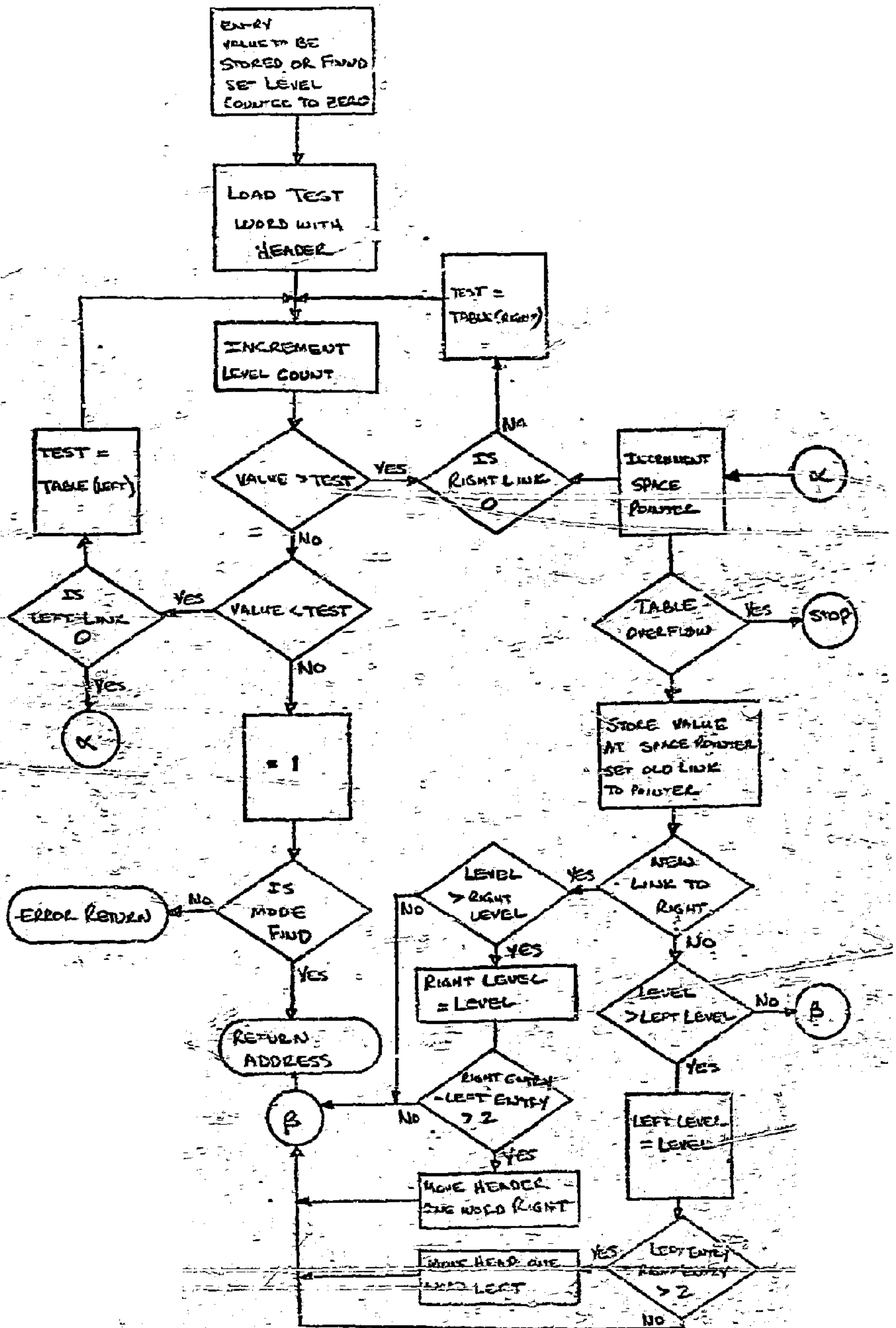


Figure 2-2. SYSTID Linked Table Search Routines.

When representing an RF link, the ability to model in the baseband region is significant when considering computer run times. Use of the sampled-data technique coupled with a translation process for all RF components provides a reduction in compute time grossly given by the ratio of baseband frequency to RF frequency. The complication, of course, is that the translation results in complex representations of all signals and components; the key system component being continuous functions or filters. Appendix A presents the mathematical development of such a process. When given an RF system, the SYSTID library will translate and maintain the integrity of any continuous transfer function. When translating from a carrier frequency (ω_c) to baseband, a bandpass function will have bandpass regions about the frequency origin and $-2\omega_c$. The response at $-2\omega_c$ should not be of importance since the baseband signal should be analytic (i. e., the imaginary part must equal the Hilbert transform of the real part). This means the spectrum of the baseband signal should be zero at $\pm\omega_c$.

It is noted that aliasing of the baseband signal will occur if the signal has frequency components greater than one-half the sample (Nyquist) frequency. Also, if the signal is not analytic, a ripple in the output amplitude and phase at $2\omega_1$ ($\omega_1 = 2/T \tan \omega_c T/2$) will occur.

The techniques discussed here and in the appendices have been utilized in the past with great success, resulting in the SAI SAMDAT library, upon which SYSTID depends.

SECTION 3.0

PROGRAM USAGE

3.1 INPUT DESCRIPTION

The SYSTID Processor accepts input decks written in the SYSTID input language which represent either systems, or models to be used in systems, and generates a symbolic FORTRAN V program (system) or subroutine (model) which is then compiled by the FORTRAN V compiler. If the input deck is a model, it may, depending upon the user's wishes, be entered into the user's library of SYSTID models. If the model generated is compiled, both the symbolic and relocatable are entered into the user Program Complex File (PCF), otherwise only the symbolic is entered.

A SYSTID system is a complete program which is to be executed on the computer in order to obtain specific information about the telecommunications link being simulated. A system is a model prefixed with system-parameter definitions, input data specifications, output specifications, and possibly post simulation analysis declarations (table 3-1). It is a stand alone program, i.e., it cannot be incorporated as such into another system or model. The functional make-up of a system can be described as follows:

- System parameters: start time, run time, sample time, specifications, etc.
- Input data specification statement: parameters to be read in at execution time.
- Output specifications (data to be printed or plotted).
- Post-processing (optional).
- Statements defining a model (Topological Description).

A model is intended to be a subroutine which simulates the properties of a physical device in a telecommunications link. A model is defined by specifying its topology and its components. The topology is defined in terms of the nodes at which its components connect.

TABLE 3-1

SYSTID IDENTIFIERS

Identifier	Use
SYSTEM -	Indicates that the deck following defines a SYSTID system
MODEL -	Indicates that the deck which follows defines a SYSTID model
END -	Used to indicate the end of a model or system deck
DATA - variable list	Indicates that the following values are to be read in at execution time by namelist (used only in a system) name SYSTID
DEFAUL(T) - variable list	Indicates the default values for DATA parameters not input through namelist
PRINT - node or tap names	Indicates a list follows specifying output to the printer
PLOT - node or tap names	Indicates a list follows specifying output to be plotted on the CALCOMP ~~plotter~~
PPLOT - node or tap names	Indicates a list follows specifying output to be plotted on the printer
POST - routine name, arg_1, arg_2, ...	Indicates that the following post-processing routine is to be called
PAGE -	When present causes 8 1/2" x 11" compatible output

A model component is defined as the following:

- A SYSTID library model
- A user written model
- A FORTRAN V arithmetic expression involving any intrinsic SYSTID system parameters (table 3-2), constants, FORTRAN library functions, SYSTID library functions, and model output nodes (taps).

3.1.1 Input Data Forms and Types

The SYSTID identifiers must precede any topological description; otherwise, any card order is permissible. All input to SYSTID is completely free field. Except where otherwise noted, blanks are totally ignored in all fields of SYSTID input data decks. Figure 3-1 depicts the card data fields

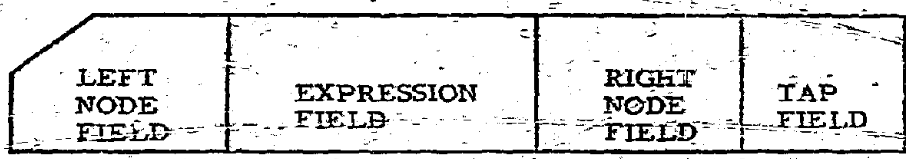


Figure 3-1.

The SYSTID field separator may be any nonalphanumeric, with two exceptions. The UNIVAC control character in column 1; and the 029 numeric shift T (0-8-2).

It is the responsibility of the user to be consistent in using real and integer numbers in any expression. Generally in this manual, all variables are floating point unless the variable name begins with I thru N (i.e. standard FORTRAN conventions are observed).

3.1.2 Data Specifications and Definitions

There are two basic SYSTID data cards - one being an identifier, the other a topological descriptor. An example for each identifier and topological descriptor is given below.

3.1.2.1 SYSTID Identifier Data

Table 3-1 lists the valid SYSTID identifiers. The standard format for any SYSTID identifier is:

identifier ϕ data list or comment

where: ϕ is any non-alphanumeric character

(a) SYSTEM ϕ comment

e.g. SYSTEM = TEST SYSTID

(b) MODEL = model name, $param_1$, $param_2$, ...

e.g. MODEL = FM MODULATOR, DF, FC

where: model name is ≤36 characters

$param_i$ = arguments to the model

(c) END ϕ comment

e.g. END = THATS ALL FOLKS

(d) DATA ϕ var_1, var_2, . . . var_n

e.g. DATA = TSTOP, DT, SETTLE, JOE

where: var_n is a FORTRAN or SYSTID variable name

(e) DEFAUL ϕ var_1, var_2, . . . var_n

e.g. DEFAUL, TSTOP = 100., DT = 1.0E-8, SAM = 10.

where: var_n is a FORTRAN or SYSTID variable name

(f) PRINT ϕ $name_1$, $name_2$, ... $name_n$

e.g. PRINT = INPUT, OUTPUT, TAP 96, NODE 6

where: $name_n$ is any valid node name or tap in the system

(g) PLOT / PPLOT } ϕ $name_1$, $name_2$, ... $name_n$

e.g. PLOT = OUTPUT, NODE 4, TAP 3

where: $name_n$ is any valid node name or tap in the system

(h) POST ϕ name, arg_1, arg_2, ...

e.g. POST = SPECTM, FL, FU, NPTS

where: name is the name of the post processing arg_n are its required arguments.

TABLE 3-2

INTRINSIC SYSTID PARAMETERS

Variable	Usage
TIME (or T)	The time as kept by the Simulation Clock (unrelated to actual computer run time)
TSART	The simulation start time (i.e. a time bias for output labeling)
TSTOP	Simulation stop time
SETTLE	Setting time before outputting
DT	Sample time
$	Used to denote the current signal
Z + 1	Absolute address of the first data cell available to the model (V(Z+1))
ZZ	Absolute address of the last data cell used by the model (V(ZZ))
V ( ) or VV ( )	Dynamic storage array
VCIN	Address of the current complex value
VIN	Address of the current real input
VOUT	Address of the current output
PI	3.14159

3.1.2.2 Topology Descriptors

3.1.2.2.1 Model Description Format[1]. Referring to the standard card format given in Section 3.1.1 depicting the Left Node Field (LNF), Expression Field (EF), Right Node Field (RNF), and Tap Field (TF), a model or system is easily defined.

Using figure 3-2 for example shows the basic format for a model description. A system is simply a model prefixed with the necessary SYSTID identifiers.

LNF	EF	RNF	TF
MODEL ϕ	NAME		
NODE 1 ϕ	EXPRESSION ϕ	NODE 2 ϕ	TAP 1
INPUT ϕ	DEVICE 2 ϕ	NODE M	
NODE M ϕ	DEVICE K ϕ	OUTPUT	
END			

Figure 3-2

There must be at least one reference to node INPUT and node OUTPUT. SYSTID assumes that the system or model signal begins at node INPUT and progresses to node OUTPUT. The use of taps provides the user with the flexability necessary for multiple input and outputs, which are covered below.

The structure cards, which define the topology and components, may be in any order. Statements may be continued onto more than one card (maximum of 4 continuations) by punching a ϕ into card column 1 of the continued cards (2nd, 3rd, etc.).

Node names may be any combination of up to six alphanumeric characteristics the first of which may be numeric, i.e., numbers may be used as node names. Tap names are 1-3 numeric characters preceeded by the characters TAP. Sequential numbering of taps is recommended. If the tap numbers are not sequential the SYSTID processor will renumber them internally. The significance of the renumbering will be discussed later. The tap field is not required, nor is the ϕ following the END card unless it is commented. Model decks may be stacked one after another and they must preceed a system. The end card may be omitted if another model or a system follows immediately. The model deck terminates when the words END, MODEL or SYSTEM are encountered in the left node field.

The device field may not reference a model which has not been processed and saved. If model A and model B are processed in the same run, with deck A preceeding deck B, the device fields of B may reference A but not vice versa.

---

[1] Note that Appendix B contains the development of the major SYSTID models.

3.1.2.2.2 Expression Field. The expression field (EF) may be one of several expressions as given in section 2.2.

The expression content is very flexible, in that FORTRAN expressions or simple model references are acceptable. Table 3-3 lists the SYSTID library models and their reference. It is the responsibility of the user to ensure that all the arguments to the models are correct. The only checking by SYSTID is that the total number of arguments is correct.

Section 3.1.2.2.6 contains the procedures necessary to update the permanent SYSTID library.

3.1.2.2.3 Error Checking. The SYSTID processor will check for syntactical errors such as missing ϕ's and mangled expressions in the device field and for logical errors in the topology. The topology is considered correct if

(1) No node connects directly to itself

(2) There exists at least one path from INPUT node to the OUTPUT node

(3) Every dangling branch terminates at a tap

```
MODEL ϕ EXAMPLE
INPUT ϕ $*$ ϕ OUTPUT
INPUT ϕ SIN ($) ϕ NODE 1
END ϕ this model sequare the signal
```

is incorrect since node 1 is dangling and does not have a tap specification.

Messages explaining the nature of any error found by the SYSTID processor will immediately precede the card image containing the error on the output listing. The processor may issue warning and give advice where it feels they are needed and, occasionally, make smart remarks.

3.1.2.2.4 Model Taps. As stated earlier a tap is an auxiliary input or output. When a tap appears only in the tap field it is an output which may be referenced by other models. A tap is probably most actually described as an intermodel node. When used in the tap field a tap has two properties which distinguish it from the node immediately preceeding it in the statement:

- A tap may be referenced outside of the model as well as inside
- A tap refers to the output of the device to which it is attached, not to the signal at the preceding node

An example will clarify this difference.

```
MODEL ɸ Model with taps.
INPUT ɸ SIN($)/2 ɸ node 1
INPUT ɸ COS ($-DELAY (T/2)) ɸ NODE 1 ɸ TAP1
NODE 1 ɸ hard limiter ɸ OUTPUT
END
```

The amplitude of the signal at node 1 is SIN ($)/2+COS ($-Delay (T/2)) but the output signal at the tap is cos($-Delay (T/2)). This particular property of taps will be very useful when specifying output data from system runs and will in general specify model topology.

A tap may be used in the device field. It has the value of the output signal of device to which it is attached. A tap which appears only in the device field and not in the tap field in a particular model is a tap input. If, when the model is used in another model, there is an input to this tap the reference to the tap is replaced by the value of the tap input signal. Any input taps which are not used will have a value of zero. Consider the following model:

```
MODEL ɸ MULTIPLIER
INPUT ɸ $*TAP1 ɸ OUTPUT
END
```

If no connection is made to TAP1 when the multiplier is used the output signal will always be zero.

A tap may appear both in the tap field and in the device field, which makes it an output signal used internally as well as externally. Logically, the value of the tap in the device field is just the output signal to the device to which the tap is attached.

```
MODEL ɸ this model doubles the signal
INPUT ɸ $ ɸ NODE 1 ɸ TAP01
NODE 1 ɸ $ + TAP01 ɸ OUTPUT
END
```

In the above example the output signal of the model is twice the input signal and the output signal at TAP01 is just the input signal.

A tap is referenced externally only from the device field of another model. A tap is specified by stating the name of the model and the tap number. The Tap number is positive for an input tap, and negative for an output tap. The Right Node Field of the tap connection is the LNF of the model reference. For example, consider the use of the multiple model given above:

MODEL ϕ tap use example
INPUT ϕ COS ($) ϕ NODE 1
INPUT ϕ SIN ($) ϕ NODE 2
NODE 1 ϕ MULTIPLIER ϕ OUTPUT
NODE 2 ϕ MULTIPLIER + 1 ϕ NODE 1
END

When the models are processed, a check is made to ensure all INPUT taps have been connected. If not, a warning message will be issued. Output taps are not necessarily connected.

The reason for using the model input node name in the RNF of any tap connection is to relieve any ambiguities created when the same model is referenced several times. For example, a large system could easily contain several multipliers. The RNF of the tap connections identify the particular multiplier being referenced.

Taps should be sequentially ordered from 1 to the largest tap number. The processor will not reorder the numbers but will shift them into proper position. The processor output, however, will output the tap order table in all cases. Note that TAP1, TAP01 and TAP001 are identical and all tap numbers must be unique.

3.1.2.2.5 Arguments to Models. It is often desirable to construct general models which require a small number of parameters when actually used. A good example of this type of model is FILTER in the SYSTID library. When using FILTER, the proper parameters defining the order and type of function, the bandwidth, etc., of the filter must be supplied. In general a reference of a model requiring parameters will be of the form:

NODE 1 ϕ MODEL 2 (A, B, C, D, E,) ϕ NODE 2

The parameters which will be required for actual use of the model must be supplied on the model card when constructing a model, as follows:

MODEL ϕ MODEL 2 ϕ A ϕ B ϕ C ϕ D ϕ E

Parameters defined on the MODEL card may be used in any expression in the device field. When used the model, parameters must be supplied (in proper order). The supplied values may be constant (1, 3.14159, ...), parameters which are read in at run time, or internally definedtaps.

The multiplier example can also be constructed in the following form:

```
MODEL ϕ MULTIPLIER ϕ A
INPUT ϕ $*A ϕ OUTPUT
END
```

and referenced as follows:

```
MODEL ϕ MULTIUSE
INPUT ϕ SIN ($) ϕ N1 ϕ TAP1
INPUT ϕ COS ($-DELAY (T/2)) ϕ N2
N2 ϕ MULTIPLIER (TAP 1) ϕ OUTPUT
END
```

The technique described above could also be used to scale the signal by any value.

```
MODEL ϕ MULTIUSE
INPUT ϕ MULTIPLIER (1/3.14154) ϕ OUTPUT
END
```

3.1.2.2.6 SYSTID Library Procedures. The SYSTID library dictionary is the element named LIBARY, which must be present for the first phase execution. The element provides:

- Cross referencing of model name references and entry points
- Flags the entry point as a function or subroutine
- Provides the number of required arguments for error checks
- Provides the number of taps and whether each is an input or an output
- Provides an indicator (for future use) as to the phase the model is used. (Presently this flag is ignored).

SYSTID, when encountering a model reference in the expression field, will scan the dictionary table to determine the proper entry point, arguments, etc. If not found, SYSTID assumes the reference is a FORTRAN function.

A partial listing of the LIBARY element is given in table 3-4, the header is self-explanatory.

TABLE 3-3

MODEL REFERENCES

SIGNAL GENERATORS

Transcendental Functions

SIN (x), COS (x), TAN (x) — x in radians

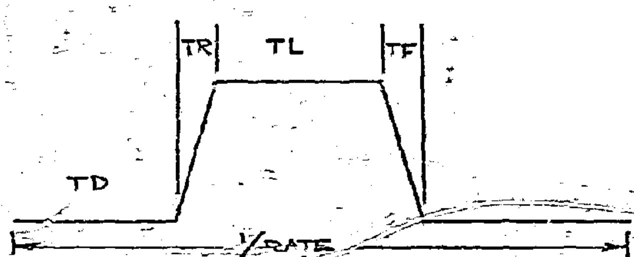
SINE (y), COSINE (y), TANGNT (y) — y in degrees

Square Wave

SQ (R) where R = frequency or rate

Pulse Generator (Periodic)

PULSE (RATE, TD, TR, TL, TF)



Arbitrary or Non-Linear Functions

TABLE (XIN, X1, Y1, X2, Y2, X3, Y3, X4, Y4, X5, Y5)

where:

XIN = Independent variable

X1, Y1 ... X5, Y5 — Five point pairs describing the function

TABLE 3-3 (Continued)

MODEL REFERENCES

Periodic Functions PTABLE (T1, Y1, T2, Y2, T3, Y3, T4, Y4, T5, Y5) where: T1, Y1 . . . T5, Y5 } Point pairs describing the function Period = T5 **Gaussian Noise Generator** GNOISE (SNR, ENB, ISTART) or GNOIS2 (ETA, ISTART) where: SNR = Signal-to-noise ratio desired in bandwidth ENB ETA = Desired noise spectral density ISTART = A positive, non-zero integer for initializing the random number generator
MODULATORS AND DEMODULATORS **Amplitude Modulators** AM MODULATOR (BETA, FC) or RF AM MODULATOR (BETA, FC) where: BETA = Modulation index ratio FC = Carrier frequency

TABLE 3-3 (Continued)

MODEL REFERENCES

Frequency Modulators

FM MODULATOR (DF, FC)

or

RF FM MODULATOR (DF, FC)

or

SQ FM MODULATOR (DF, FC)

where: DF = Carrier frequency deviation per unit input

FC = Carrier frequency

Phase Modulators

PHASE MODULATOR (BETA, FC)

or

RF PHASE MODULATOR (BETA, FC)

or

SQ PHASE MODULATOR (BETA, FC)

where: BETA = Modulation index (Radians)

FC = Carrier frequency

Delta Modulators

DELTA MODULATOR (PW, PPS)

where: PW = Pulse width

PPS = Pulse repetition rate

TABLE 3-3 (Continued)

MODEL REFERENCES

Multi-Level Coders (M-ary)

MLTPCM (BT, M)

where: BT = Bit time

N = Number of levels (symbols)

Amplitude Demodulators

AM DEMOD

or

RF AM DRMOD

Note: No arguments required

Frequency Demodulators

FM DEMOD (DV, FC)

or

RF FM DEMOD (DV, FC)

where: DV = Output magnitude per unit frequency deviation

FC - Carrier frequency

Phase Demodulators

PHASE DEMOD (DV, FC)

or

RF PHASE DEMOD (DV, FC)

where: DV = Output magnitude per unit phase deviation

FC = Carrier frequency

TABLE 3-3 (Continued)

MODEL REFERENCES

FM with Feedback (FMFB)

FMFB (NIF, NTYPE, AR, EM, BIF, GAIN, FIF, FC, DV, DF)

where:

NIF	= IF filter order	
NTYPE	= Type of filter	
	= 1 for Butterworth	
	= 2 for Chebyshev	
	= 3 for Bessel	
	= 4 for Butterworth-Thomson	
	= 5 for Elliptic	
AR	= Amplitude Ripple (dB)	
EM	= M-factor for BT, stopband ratio (>0) or modular angle (<0) for Elliptic	
BIF	= IF Bandwidth	
FIF	= IF frequency	
FC	= Carrier frequency	
DV	= FM Discriminator constant	
DF	= VCO Deviation constant	

Matched Filter

= MATCHED FILTER (BT)

where:

BT = Bit time

TABLE 3-3 (Continued)

MODEL REFERENCES

FILTERS
FILTER (NP, IF, IG, FX, BW, FC, AMP, AR, EM)
BUTTERWORTH (NP, IG, FX, BW, FC, AMP)
CHEBYSHEV (NP, IG, FX, BW, FC, AMP, AR)
BUTTERWORTH-THOMSON (NP, IG, FX, BW, FC, AMP, EM)
ELLIPTIC (NP, IG, FX, BW, FC, AMP, AR, EM)
BESSEL (NP, IG, FX, BW, FC, AMP)

where:

NP	= Filter order	
IF	= Filter function	
	= 1 for Butterworth	
	= 2 for Chebyshev	
	= 3 for Bessel	
	= 4 for Butterworth-Thomson	
	= 5 for Elliptic	
IG	= Filter geometry	
	= 1 for lowpass	
	= 2 for highpass	
	= 3 for bandpass	
	= 4 for bandstop	
AR	= Amplitude ripple (dB)	
EM	= M-factor for Butterworth-Thomson; stopband ratio (>0) for modular angle (>0) for elliptic functions	
FX	= Arithmetic Center Frequency	
BW	= Bandwidth	
FC	= Translation frequency	
AMP	= Voltage gain (ratio) at FX	

TABLE 3-3 (Continued)

MODEL REFERENCES

FC	FX	IG	Result
0	-	-	Baseband filter
>0	>0	3	RF translated filter
>0	<0	3	Symmetric RF translated filter
>0	0	1	Equivalent LP function

NOTE: Utilizing the symmetric RF translated filter will reduce the run time on the order of one-half that of the actual filter.

Utilizing the equivalent low pass filter will reduce run time on the order of one-fourth that of the actual filter.

TABLE 3-3 (Continued)

MODEL REFERENCES

Quadratic Factors

QFACTOR (AMP, A1, A2, A3, A4, A5, A6)

where:

$$G(s) = AMP * \frac{A1s^2 + A2s + A3}{A4s^2 + A5s + A6}$$

Leadlag Functions

LEADLAG (AMP, F1, F2, F3, F4)

where:

$$G(s) = AMP * \frac{\left(\frac{s}{2\pi F1} + 1\right)\left(\frac{s}{2\pi F2} + 1\right)}{\left(\frac{s}{2\pi F3} + 1\right)\left(\frac{s}{2\pi F4} + 1\right)}$$

LEAD FUNCTION (AMP, F1, F2, F3)

where

$$G(s) = AMP * \frac{\left(\frac{s}{2\pi F1} + 1\right)\left(\frac{s}{2\pi F2} + 1\right)}{s\left(\frac{s}{2\pi F3} + 1\right)}$$

TABLE 3-3 (Continued)

MODEL REFERENCES

LIMITERS

Hard Limiter

RF HARD LIMITER

or

HARD LIMITER

where:

Output level = 1

Soft Limiters

SOFT LIMITER (A, SLOPE)

or

RF SOFT LIMITER (A, SLOPE)

```
@ ELT LIBARY,1,710113, 54350

.
1
2 THIS ELEMENT IS THE MODEL LIBARY DIRECTORY FOR THE SYSTID PROCESSOR.
3 ADDITIONS AND DELETIONS CAN BE MADE SIMPLY BY REMOVING OR ENTERING THE
4 DESCRIPTOR CARD,  DO NOT EMBED BLANK CARDS IN THIS ELEMENT
.
5           FORMAT
.
6    A    CC  1-36       MODEL NAME, ALPHANUMERIC, LEFT ADJUST AND NO EMBEDDED
7                        BLANKS
8    B    CC 41-46       THE ENTRY POINT NAME CORRESPONDING TO (A)
9    C    CC    50       = 1 TO INDICATE SUBROUTINE, 0 FOR A FUNCTION.
10   D    CC    53       = 0-9 IS THE NUMBER OF ARGUMENTS REQUIRED FOR (B)
11   E    CC    56       = 0-9 IS THE NUMBER OF TAPS ON THE MODEL
12   F    CC 61-69       INDICATES TYPE OF EACH TAP, I. E. INPUT OR
13                       OUTPUT. THE RIGHTMOST DIGIT REPRESENTS THE
14                       FRIST TAP, ETC. 1 INDICATES INPUT, 0 INDICATES OUTPUT
14                       CC 68 FOR ARG NUMBER 2, ETC.
15   G    CC    73       PHASE INDICATOR = 1 FOR SETUP  = 2 FOR RUN  =3 FOR POST
.
.
**    THIS CARD STARTS THE LIBRARY. DO NOT REMOVE.  COMMENTS ABOVE IT ONLY.
MODEL MODEL      DO NOT REMOVE THIS IS THE TLEMENT NAME COUNTER
RFAMMODULATOR                        RAMMOD   1  2  0            2
RFLIMT                               RFLIMT   1  0  0            2
.
.
.
HARDLIMITER                          HARD     1  0  0            2
RFSOFTLIMITER                        RFSOFT   1  2  0            2
PMDEMOD                              PMDEMM   1  2  0            2



$$$$$$                                   DO NOT REMOVE MARKS END OF LIB.
.
```

Table 3-4

SYSTID, when processing new temporary models assigns entry points beginning with MODEL A and progressing thru MODEL Z as each model is encountered in any given run. A temporary entry is then made in the dictionary and will be displayed on phase one output. To incorporate any model into the SYSTID permanent library, a permanent entry name must be assigned and the library card inserted into the LIBARY element. Note that an entry point may be given several model names.

3.2 PROGRAM RUN PREPARATIONS

When executing the first phase of SYSTID, the procedure PROCS/SYSTID and the element LIBARY are required along with the absolute element for the first phase (SYSTID). The LIBARY element is the model library dictionary.

The second phase requires only the library file containing all the SYSTID library relocatable elements and the elements output to the PCF by the first phase (MAINƀƀ/SYSTID).

3.2.1 Deck Setup

The technique used by SYSTID is to output the processed models to the PCF as elements named MODELA/SYSTID, MODELB/SYSTID, etc., in the same order as processed. When a system is processed, its element name is MAINƀƀ/SYSTID. Therefore, the user at MSC must provide the FORTRAN control card for each of the elements. The models generated in the following example are "temporary" models for this run only. That is, any system description can reference model EXAMPLE 1 during this run only. Models are permanent when the relocatable is available and an entry is made in the LIBARY element used by the first pass.

Figure 3-3 is the run deck set-up for the MSC Univac 1108 system.

```
" XQT SYSTID
MODEL= EXAMPLE ONE         }
      .                    }  MODELA/SYSTID
      .                    }  is generated in PCF
      .                    }
      .                    }
END                        }
MODEL= EXAMPLE TWO         }
      .                    }
      .                    }  MODELB/SYSTID
      .                    }
      .                    }
END                        }
MODEL= EXAMPLE THREE       }
      .                    }
      .                    }  MODELC/SYSTID
      .                    }
      .                    }
```

```
" XQT CUR
---- LOAD THE PCF WITH SYSTID ABSOLUTE,PROCS/SYSTID, AND LIBARY
" XQT SYSTID
        .
        .
        . MODEL DATA DECKS (IF ANY)
        .
        .
        . SYSTEM DATA DECK
        .
        .
        .
"I FOR,* MODELA/SYSTID          }  One for each model
        .                       }  input, if any
        .                       }
"I FOR,* MAIN  /SYSTID
" XQT CUR
        .
        . DELETE ABSOLUTE AND PROCEDURE ELEMENTS (if desirable)
        .
        . LOAD SYSTID LIBRARY
        .
        . SAVE ANY ELEMENTS IF DESIRED
        .
        .
" XQT MAIN  /SYSTID
 $SYSTID  .....  $  (NAMELIST I/O IF SPECIFIED IN SIMULATION
        .
```

Figure 3-3

```
END
SYSTEM. USE EXAMPLES ONE,TWO,THREE  }
       .                            }
       .                            }  MAINbb/SYSTID
       .                            }
       .                            }
END                                 }
"I FOR,* MODELA/SYSTID              }
"I FOR,* MODELB/SYSTID              }  User supplied
"I FOR,* MODELC/SYSTID              }  compiler cards
"I FOR,* MAIN  /SYSTID              }
" XQT MAIN  /SYSTID                 }  Execute the second pass
```

3.2.2 Required I/O Devices

The SYSTID program contains two phases. The first phase requires the Logical Unit assignments as follows:

Logical Unit	Device	Size
5	Card Reader	—
6	Line Printer	—
	Scratch Drum	—

Phase two requires the following I/O devices

Logical Unit	Device	Size
5	Card Reader	
6	Line Printer	
13	Drum	$250,000_{10}$
14	Drum	$250,000_{10}$

3.3 OUTPUT DESCRIPTION

3.3.1 Data Output

SYSTID output consists of the first phase processing and the simulation or second phase output. The first phase provides output similar to the FORTRAN V compiler. Figure 3-4 is an example of the first phase output for a particular model. Anotations on the output fully explain their significance.

```
FIRSTIFAMP                                    MODELC   1  3  2           1   2 } LIBARY card for
SYSTID PROCESSOR LEVEL 1                                                       } permanent entry
VERSION OF JANUARY 1, 1971 FOR THE UNIVAC 1108,
THIS DECK WAS PROCESSED ON 17 FEB 71 AT 16:43:31,



SYSTID MODELS REFERENCED                          ENTRY POINT

     COMPLEXMULTIPLY                                CMULT
     FILTER                                         FILTER


THIS MODEL ASSIGNED THE ENTRY POINT NAME MODELC


TAP ORDER:    1 TAP001    2 TAP002



000013          MODEL=FIRST IF AMP,GAIN,SLOPE,IFUNC
000014               INPUT < COMPLEX MULTIPLY(TAP2) > N1
000015               N1 < FILTER(2,IFUNC,3,0,    ,6,E6, 60,E6,1.,0.,0.) > N2
000016               N2 < FILTER(2,IFUNC,3,0,    ,6,E6, 60,E6,1.,0.,0.) > N3
000017               N3 < FILTER(2,IFUNC,3,0,    ,6,E6, 60,E6,1.,0.,0.) > OUTPUT
000018               INPUT < 10.**((GAIN+TAP1*SLOPE)/20.) > D1 'TAP2
000019          END
```

Figure 3-4

3.3.2 Optional Output

The second phase output is completely optional under user control. The two forms of output are printed and graphical presentations, whose size is selectable as 8-1/2 by 11 in. or 11 by 14 in. (see Section 3.1). The current version provides printer plots (PTPLT), with the entry point TMPLT reserved for calcomp or SC4020 graphical routines. Examples of the output are contained in Section 5.5.

SECTION 4.0

EXECUTION CHARACTERISTICS

4.1 RESTRICTIONS

The first phase of SYSTID, is a stand-alone program requiring approximately 41,500 decimal words of core. All variables affecting this size are parameters contained in the procedure named "PARAM". The capabilities are thus controlled by this element.

The storage requirement for the second phase, the actual simulation, is totally dependent upon the system being simulated.

The second phase requires the sampling time (DT) as a parameter. This sampling time must be carefully shosen such that accuracy is maintained while minimizing run time. The rule of thumb is to sample approximately 20 times the highest frequency of interest, although in several simulations, relaxation of this rule has been beneficial.

The output, when plotting or post processing, is limited to 50,000 decimal words due to drum storage. In addition, a maximum of 10 variables (including time) can be stored. This includes all plotted variables. A process in the plotting routines will edit the saved data and limit the number of points actually plotted to 1,000.

4.2 RUNNING TIME/LINES OF OUTPUT

Execution time of the first phase depends on the model or system complexity, but is insignificant when compared to phase two. One to two pages of output can be expected for each model.

Execution time of phase two is dependent upon the user's system and his selection of stop time and sampling rate. Section 5.5 provides some examples. Output, again, is directly controlled by the user.

4.3 ACCURACY/VALIDITY

Accuracy of the simulation process depends on both the selection of sampling rate and the ability to model various functions. The techniques employed have been verified on several programs such as SURVEYOR and Mariner-Mars 1971; and have compared favorable with theoretical results.

SECTION 5.0

REFERENCE INFORMATION

The data to follow will further define the characteristic properties of the SYSTID program in terms of:

5.1 Program and Subroutine Flowcharting[1]

5.2 Symbol Definition

5.3 Subroutine Documentation

5.4 Program Listing[1]

5.5 Sample Input/Output

[1] These elements consist of hundreds of sheets of computer listing sheets, far too bulky to include directly in this documentation. They have been provided to MSC separately.

5.1 DETAILED FLOW CHART[1]

Figure 5-1 presents the functional diagram of the processes involved when utilizing SYSTID. The basic functions are described below:

- Data Decomposition — the input data is scanned for topological or system data and the input is stored on scratch drum.
- Table Building — the various linked tables are built based upon the input data.
- Program Generator — based upon the linked tables and the library, the simulation program is written in the PCF. The listed output, with diagnostics is also printed.
- EXEC II processing — is necessary to compile Phase I output and to load the SYSTID Library.
- Execution of MAIN/SYSTID — actual simulation of the system utilizing scratch drum for saving output.
- Plotting — Printer plots of selectable output.
- Post Processing — linkage to post processing saved data.

5.2 SYSTEM DEFINITIONS

The key symbol definitions for the first phase of SYSTID are contained in the procedure "PARAM".

In addition the symbols given in tables 3-1, 3-2, and 3-3, as well as Appendix B comprise the definitions necessary for the user interface.

Internal symbol definitions are annotated in the program listing, which serve as the major subprogram documentation.

[1] The detailed flow charts are separate from this document.

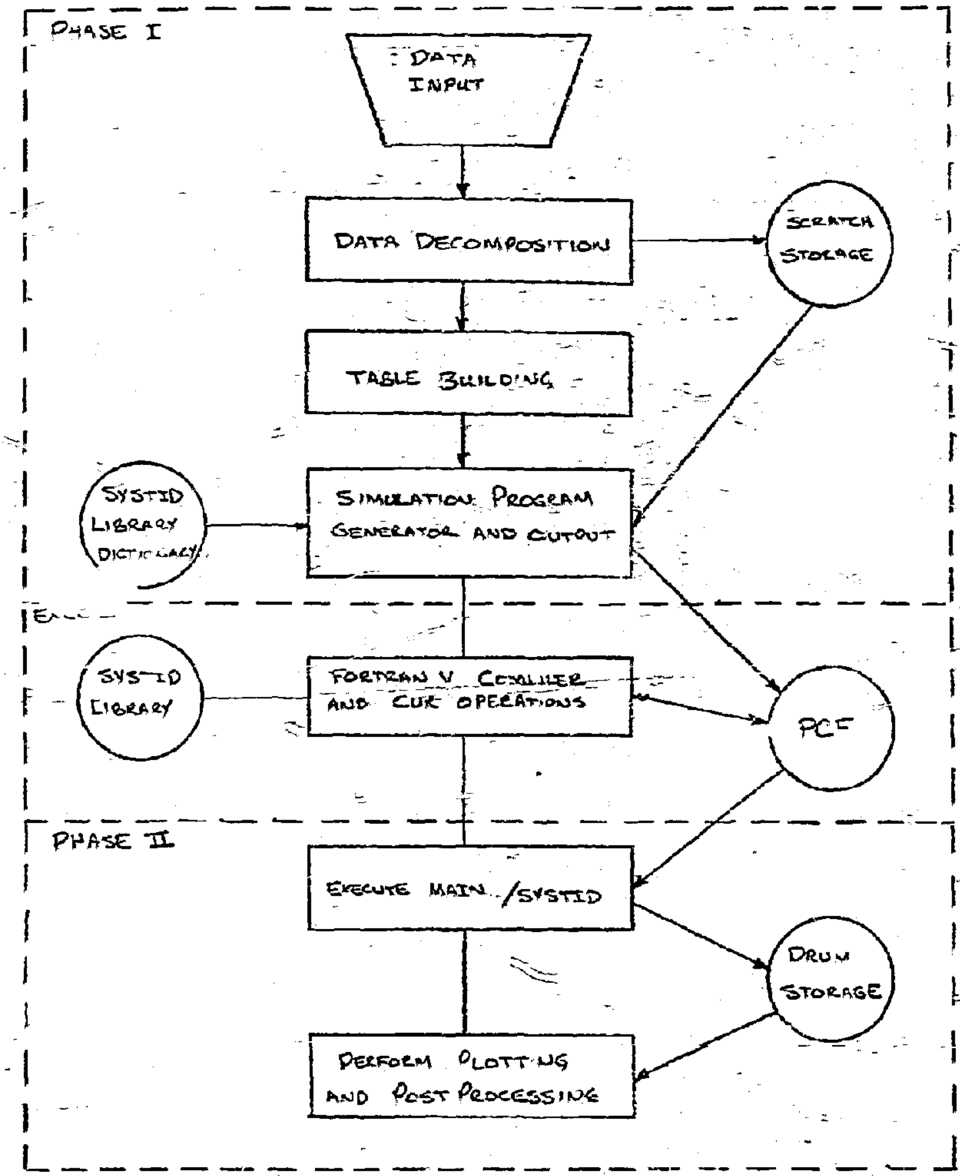


Figure 5-1. SYSTID Functional Flow

5.3 SUBROUTINE DOCUMENTATION

The material to follow will list in alphabetical order, all of the subroutines utilized by SYSTID. A brief synopsis of each subroutine's function is also given. A complete cross reference is contained in the flowchart documentation provided separately to MSC.

5.3.1 ASGTAP

This routine keeps track of the tap subscripts used in the model library.

5.3.2 ASSIGN

This routine assigns the V-array subscripts to nodes. If the node was previously assigned a subscript, ASSIGN merely returns the value. If the node is in both right-hand and left-hand node tables, the subscript increment is stored in both tables (in H2 of the first subword under a node name).

5.3.3 COMPIL

This routine controls generation of the body of the model in terms of its topology and appropriate generation of describing code of the proper sequence.

5.3.4 CREATE

This routine creates the model and main elements in the program complex file (PCF), and also the preamble code for models and systems. (The SYSTID processor presently can create one FORTRAN file at a time ... future up-dates may include multiple file capability).

5.3.5 DRUM

This routine generates the dimension statement required for the Drum I/0 buffer.

5.3.6 DRUMBF

This routine generates the FORTRAN I/0 statements required to save data or drum files for subsequent plotting or post processing.

5.3.7 EDIT

This routine edits the input expression and replaces the $ characteristics with the appropriate V array subscripted variable. It also edits the delimiter set into a FORTRAN-acceptable set.

5.3.8 EINES

This routine maintains a linked subtable under each entry in the left node table, by means of linear linking.

5.3.9 EQUNIT

This routine establishes correspondence between a FORTRAN unit and an opened element in the PCF.

5.3.10 ERECTS

This routine generates the various FORTRAN statements necessary to reference a model or function.

5.3.11 ERRMSG

This routine returns the error message and its length in words, given an error number.

5.3.12 EDIT1

This routine scans the expression to find model, function, and variable names. When a name is found the various tables are examined to determine the type of expression.

5.3.13 GREBE

This routine maintains a non-linked table of the expressions found between node names on the input deck.

5.3.14 GWIN

This routine processes the SYSTID input language. It constructs the tables and lists necessary for the generation of the output FORTRAN program.

5.3.15 INCLUD

This routine generates "include main list" to include the canned programs for use in the main program.

5.3.16 LIB 003

This routine reads in the library element and makes its entries into the library search program.

5.3.17 LIB 004

This routine is the library, consisting of a linked table of model names and associated descriptors. It is initially loaded with entries in the "library" element.

5.3.18 LISTIO

This routine generates the read and write NAMELIST statements.

5.3.19 LISTIT

This routine lists the input deck and diagnostic edited into proper position before the error line.

5.3.20 LIT

This routine converts an integer to its BCD form.

5.3.21 SYSTIL

This routine is the main program.

5.3.22 LUCHT

This routine maintains the tree of the left node names.

5.3.22 MOGUE

This routine generates a data statement for the "DEFAUL" values.

5.3.23 NADINE

This routine reads in the input deck and copies it on storage drum.

5.3.24 NAMLST

This routine generates a namelist statement to read in the variables in the data statement.

5.3.25 NTAB $

Used only for SAI 1108 system.

5.3.26 PCF/ISD

This routine performs all PCF manipulations on the SAI 1108 system.

5.3.27 PCF/MSC

This routine performs all PCF manipulations on the MSC 1108 system.

5.3.28 PREAMB

This routine generates the preamble or subroutine entry point for the "Model" programs.

5.3.29 PUTOUT

This routine generates the printer output statements buffered to the amount required for a readable output format.

5.3.30 QUEUE

This routine maintains a list of "Active" nodes, i.e. nodes which have appeared in the right node field, but which have not been processed yet.

5.3.31 RECTUS

This routine maintains the tree of the right node names.

5.3.32 SETUP

This routine is called initially and after the completion of processing on each model or system, it initializes all the necessary pointers and zeroes the appropriate tables, and loads the work area with the next batch of cards from the input buffer.

5.3.33 SPORU

This routine maintains the linked TAP tables.

5.3.34 TAPONE

This routine assigns each tap a number, beginning with 1 and increasing sequentially.

5.3.35 TAPTWO

This routine assigns output tap V-array subscript locations and generates the necessary FORTRAN statements.

5.3.36 TAP 3

This routine generates the V-array subscripted variable literals for taps in expression.

5.3.37 THOR

This routine tests for legitimate FORTRAN name, and if not, returns an error message.

5.3.38 TITLE

This routine loads the SYSTEM comment into an array for use by the output routines.

5.3.39 USRELT

This routine is an SAI 1108 system routine used for the creation of PCF element by users. It is used by the SAI version of SYSTID.

5.3.40 V

This routine generates the literal V(z + nn) where z is the input bias.

5.3.41 ZWEI

This routine maintains a linked subtable below each mode entry in the right node table (RECTUS), with linear linking.

5.3.42 PARAM

This procedure defines the parameters used in SYSTID for fixing program size.

5.3.43 PROCO5

This procedure defines several procedures utilized throughout the SYSTID processor for listing processing and linked table manipulation.

5.3.44 QUARTR

This procedure defines the quarter-word functions used in SYSTID.

5.3.45 SUB 1

This procedure defines some functions necessary in the linking routines.

5.4 PROGRAM LISTING

The sheer bulk of the SYSTID program listing (800 pages of output) dictates its exclusion from the manual proper. A copy of the listing is available from NASA/MSC or Systems Associates.

5.5 SAMPLE INPUT/OUTPUT

Two example sets are presented in this section, namely:

(1) SYSTID time domain simulation of a 5th order Butterworth lowpass filter, excited by a step change in input level. The listed and line-printed plots show the filter's step response. The filter parameters are: 5th order, lowpass, FX = 0, 128 Hz -3 dB bandwidth, FC = 0, GAIN = 1.

(2) SYSTID simulation of an Apollo PCM/PM/PM link, whose characteristics are given in figure 5-2.

```
FILTER RESPONSE
SYSTID PROCESSOR LEVEL I
VERSION OF JANUARY 1, 1971 FOR THE UNIVAC 1108,
THIS DECK WAS PROCESSED ON 15 JAN 71 AT 17:48:21,


SYSTID MODELS REFERENCED                    ENTRY POINT

    BUTTERWORTH                                  BUTWTH


000001          SYSTEM , FILTER RESPONSE
000002          PAGE,  MAKE IT 8 1/2
000003          PPLOT, OUTPUT
000004          DEFAUL,  TSTART=0.,TSTOP=.015,DT=.00005,NPRINT=2
000005          PRINT.TIME,N1,OUTPUT
000006            INPUT.1.0.N1
000007                N1 < BUTTERWORTH (5,1,0.,128.,0.,1.) > OUTPUT
000008          END.

 ADD ADDFIL
```

Figure 5-2. Step Response of a Butterworth Lowpass Filter, using SYSTID (21.74 secs CPU time)

```
@1 FOR,* MAIN  /SYSTID,MAIN  /SYSTID
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F5018S
THIS COMPILATION WAS DONE ON 15 JAN 71 AT 17:48:22

  MAIN PROGRAM

  STORAGE USED (BLOCK, NAME, LENGTH)

          0001   *CODE    000270
          0000   *DATA    000527
          0002   *BLANK   000000
          0003   COGENT   000007
          0004   VSPACE   023421
          0005   FLR      004785
          0006   NAME     000006
          0007   INT      000001
          0010   DIF      000001

  EXTERNAL REFERENCES (BLOCK, NAME)

          0011   BUTWTH
          0012   DRUMIT
          0013   PTPLT
          0014   NWDU$
          0015   NI02$
          0016   NSTOP$
          0017   NI01$

  STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

   0001     000057 1L           0001     000244 100L         0000     000467 1000F
   0001     000022 147G         0001     000135 15L          0001     000026 154G
   0001     000262 2L           0001     000212 20L          0001     000166 235G
   0001     000225 50L          0000 R 000125 DRUM           0003 R 000002 DT
   0005 R 000002 F              0000 I 000001 FIRST          0005 I 000000 FIRSTF
   0000 I 000435 I$             0000 I 000441 J              0000 I 000122 KOUNT
   0000 I 000124 NPRINT         0005 I 000001 NUMBF          0003 R 000005 PI
   0010 R 000000 TDT            0003 R 000003 TIME           0006 I 000000 TITLE
   0004 R 000001 V              0003 I 000006 VCIN           0003 I 000000 VIN
   0003 I 000004 ZZ

00101      1*         INCLUDE MAIN1,LIST
00103      1*         PARAMETER VSIZE=10000
00104      1*         PARAMETER FSIZE=500                            @
00105      1*         INTEGER FIRSTF,VOUT,VIN,Z,ZZ,VCIN,FIRST,TITLE(6)
00106      1*         DIMENSION PRINT(8,10)                          @ THE PRINT BUF
00107      1*         COMMON /COGENT/ VIN,VOUT,DT,TIME,ZZ,PI,VCIN
00110      1*         COMMON /VSPACE/DUMMY,V(VSIZE)                  @ SIZE IS FIXED
00111      1*         COMMON /FLR   / FIRSTF,NUMBF,F(5,FSIZE)        @ SIZE IS FIXED
00112      1*         COMMON /NAME/ TITLE
00113      1*         COMMON /INT/ DT2
```

```
00114     1*          COMMON /DIF/ TDT
00115     1*          DATA PI/3.1415927/,FIRST/0/,KOUNT/1/,SETTLE/0.0/,NPRINT/1/
00123     1*          END
00124     2*          DIMENSION DRUM(100, 2)
 .)125    3*          DATA TITLE/'FILTER RESPONSE                        '/
 00127    4*          DATA  TSTART/0./,TSTOP/.015/,DT/.00005/,NPRINT/2/
 00134    5*          INCLUDE MAIN2,LIST
 00135    5*          IF((TSTOP-TSTART)/DT.GT.0) GO TO 11          @ ARE THEY REASO
 00137    5*          WRITE(6  ,1999) TSTART,TSTOP,DT              @ NO, TELL THEM
 00144    5*     1999 FORMAT(1X,'** ERROR ** THE VALUES TSTART=',E12.6,' TSTOP=',
 00144    5*        . ,' DT=',E12.6,' ARE UNREASONABLE.')          @
 00145    5*          STOP HI
 00146    5*       11 DO 6 I=1,VSIZE
 00151    5*        6 V(I)=0.0                                      @ JUST TO MAKE S
 00153    5*          DO 7 I=1,FSIZE                                @ IN CASE CORE H
 0        5*          DO 7 J=1,5
 00161    5*        7 F(J,I)=0.0                                    @ NOT BEEN ZEROE
 00164    5*          DUMMY=0.0                                     @ FOR FIRST TIME
 00165    5*          DT2=DT/2.0
 00166    5*          TDT=2.0/DT
 00167    5*          TIME=TSTART-DT                                @ TO GET STARTED
 00170    5*          SETTLE=TIME+SETTLE                            @ SETTLING TIME
 00171    5*          @.............................................................
 00171    5*          @ THE SIMULATION LOOP BEGINS HERE.  GOOD LUCK SIR.
 00171    5*          @.............................................................
 00171    5*        4 TIME=TIME+DT
 00172    5*          ZZ=2
 00173    5*          FIRSTF=1                                      @ INITIALIZE F AF
 00174    5*          VIN=1                                         @ FOR MY SAKE
 00175    5*          VOUT=2                                        @ FOR MIKE'S SAKE
  0176    5*          Z=ZZ                                          @ FOR SAFETY'S SA
 00177    5*          GO TO 2
 00200    5*        1 CONTINUE
 00201    5*          IF(TIME.GT.TSTOP) GO TO 100                   @ IF FINISHED LEA
 00203    5*          END
 00204    6*          V(Z+3)=1.0
 00204    7*    C     BUTTERWORTH
 00205    8*          VIN=Z+3
 00206    9*          VOUT=Z+4
 00207   10*          CALL  BUTWTH(5,1.0,.128,.0,.1,)
 00210   11*          INCLUDE MAIN5,LIST
 00       11*          IF(TIME.LT.SETTLE) GO TO 4                    @ SKIP IT ??
 00       11*          IF(MOD(FIRST,NPRINT).NE.0) GO TO 20           @ PRINT IT ??
 00215   11*          END
 00215   12*    C     PRINTED OUTPUT
 00216   13*          IF(FIRST.NE.0) GO TO 15
 00220   14*          PRINT(1,1)='TIME'
 00221   15*          PRINT(1,2)='N1'
 00222   16*          PRINT(1,3)='OUTPUT'
 00223   17*          NODES=3
 00224   18*          NPAGE=5
 00225   19*       15 KOUNT =KOUNT+1
 00226   20*          PRINT(KOUNT,1)=TIME
 00227   21*          PRINT(KOUNT,2)=V(Z+3)
 00230   22*          PRINT(KOUNT,3)=V(Z+4)
  0231   23*          IF(KOUNT.LT.NPAGE) GO TO 20
 00233   24*          INCLUDE MAIN3,LIST
 00234   24*          DO 16 J=1,NODES
 00237   24*          @.............................................................
```

```
00237     24*        16 WRITE(6,1000) (PRINT(I,J),I=1,NPAGE)
00246     24*      1000 FORMAT(6X,A6, 7(4X,E12.6))
00247     24*           WRITE(6,1001)
00251     24*      1001 FORMAT(//)                                         @ FOR NICE LOOK:
00252     24*           END
00253     25*           KOUNT=1
00254     26*        20 CONTINUE
00255     27*           IF(MOD(FIRST+1,100).NE.0) GO TO 50
00257     28*           CALL DRUMIT(2,DRUM,13)
00260     29*        50 INDEX=MOD(FIRST,100)+1
00261     30*           DRUM(INDEX,1)=TIME
00262     31*           DRUM(INDEX,2)=V(Z+4)
00263     32*           INCLUDE MAIN4,LIST
00264     32*           FIRST=FIRST+1                                       @ LOOP COUNTER
00265     32*           GO TO 4                                             @ LOOP
00266     32*           @..................................................................
00266     32*       100 CONTINUE                                            @ FINISHED
00267     32*           END
00270     33*           CALL DRUMIT(2,DRUM,13)
00271     34*           CALL  PTPLT(13,2,'OUTPUT',FIRST,NPAGE)
00272     35*           STOP
00273     36*         2 ZZ=ZZ+       4
00274     37*           GO TO 1
00275     38*           END

        END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
```

@ XQT MAIN /SYSTID

TIME	.000000	.100000-03	.200000-03	.300000-03
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.306093-08	.183152-06	.197848-05	.102205-04
TIME	.400000-03	.500000-03	.600000-03	.700000-03
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.351265-04	.936974-04	.210531-03	.416206-03
TIME	.800000-03	.900000-03	.100000-02	.110000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.757232-03	.127570-02	.202863-02	.307718-02
TIME	.120000-02	.130000-02	.140000-02	.150000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.448759-02	.633014-02	.867789-02	.116055-01
TIME	.160000-02	.170000-02	.180000-02	.190000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.151681-01	.194997-01	.246125-01	.305954-01
TIME	.200000-02	.210000-02	.220000-02	.230000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.375133-01	.454255-01	.543858-01	.644407-01
TIME	.240000-02	.250000-02	.260000-02	.270000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.756296-01	.879837-01	.101526+00	.116270+00
TIME	.280000-02	.290000-02	.300000-02	.310000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.132221+00	.149375+00	.167719+00	.187232+00
TIME	.320000-02	.330000-02	.340000-02	.350000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.207561+00	.229629+00	.252427+00	.276220+00
TIME	.360000-02	.370000-02	.380000-02	.390000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	[illegible]	.326537+00	.352916+00	.380004+00

TIME	.400000-02	.410000-02	.420000-02	.430000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.407714+00	.435958+00	.464643+00	.493674+00
TIME	.440000-02	.450000-02	.460000-02	.470000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.522953+00	.552381+00	.581860+00	.611291+00
TIME	.480000-02	.490000-02	.500000-02	.510000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.640575+00	.669616+00	.698321+00	.726596+00
TIME	.520000-02	.530000-02	.540000-02	.550000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.754355+00	.781512+00	.807986+00	.833703+00
TIME	.560000-02	.570000-02	.580000-02	.590000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.668591+00	.882586+00	.905626+00	.927660+00
TIME	.600000-02	.610000-02	.620000-02	.630000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.948637+00	.968518+00	.987266+00	.100485+01
TIME	.640000-02	.650000-02	.660000-02	.670000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.102126+01	.103646+01	.105045+01	.106322+01
TIME	.680000-02	.690000-02	.700000-02	.710000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.107479+01	.108514+01	.109430+01	.110229+01
TIME	.720000-02	.730000-02	.740000-02	.750000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.110913+01	.111484+01	.111946+01	.112303+01
TIME	.760000-02	.770000-02	.780000-02	.790000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.112556+01	.112716+01	.112782+01	.112759+01

TIME	.800000-02	.810000-02	.820000-02	.830000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.112655+01	.112473+01	.112220+01	.111901+01

TIME	.840000-02	.850000-02	.860000-02	.870000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.111521+01	.111087+01	.110604+01	.110078+01

TIME	.879999-02	.889999-02	.899999-02	.909999-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.109514+01	.108918+01	.108297+01	.107654+01

TIME	.919999-02	.929999-02	.939999-02	.949999-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.106995+01	.106326+01	.105650+01	.104974+01

TIME	.959999-02	.969999-02	.979999-02	.989999-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.104300+01	.103633+01	.102978+01	.102337+01

TIME	.999999-02	.101000-01	.102000-01	.103000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.101714+01	.101112+01	.100534+01	.999820+00

TIME	.104000-01	.105000-01	.106000-01	.107000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.994584+00	.989650+00	.985034+00	.980750+00

TIME	.108000-01	.109000-01	.110000-01	.111000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.976807+00	.973215+00	.969977+00	.967096+00

TIME	.112000-01	.113000-01	.114000-01	.115000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.964573+00	.962405+00	.960589+00	.959116+00

TIME	.116000-01	.117000-01	.118000-01	.119000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.957985+00	.957179+00	.956689+00	.956503+00

TIME	.120000-01	.121000-01	.122000-01	.123000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.956606+00	.956983+00	.957619+00	.958496+00

TIME	.124000-01	.125000-01	.126000-01	.127000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.959596+00	.960902+00	.962395+00	.964056+00

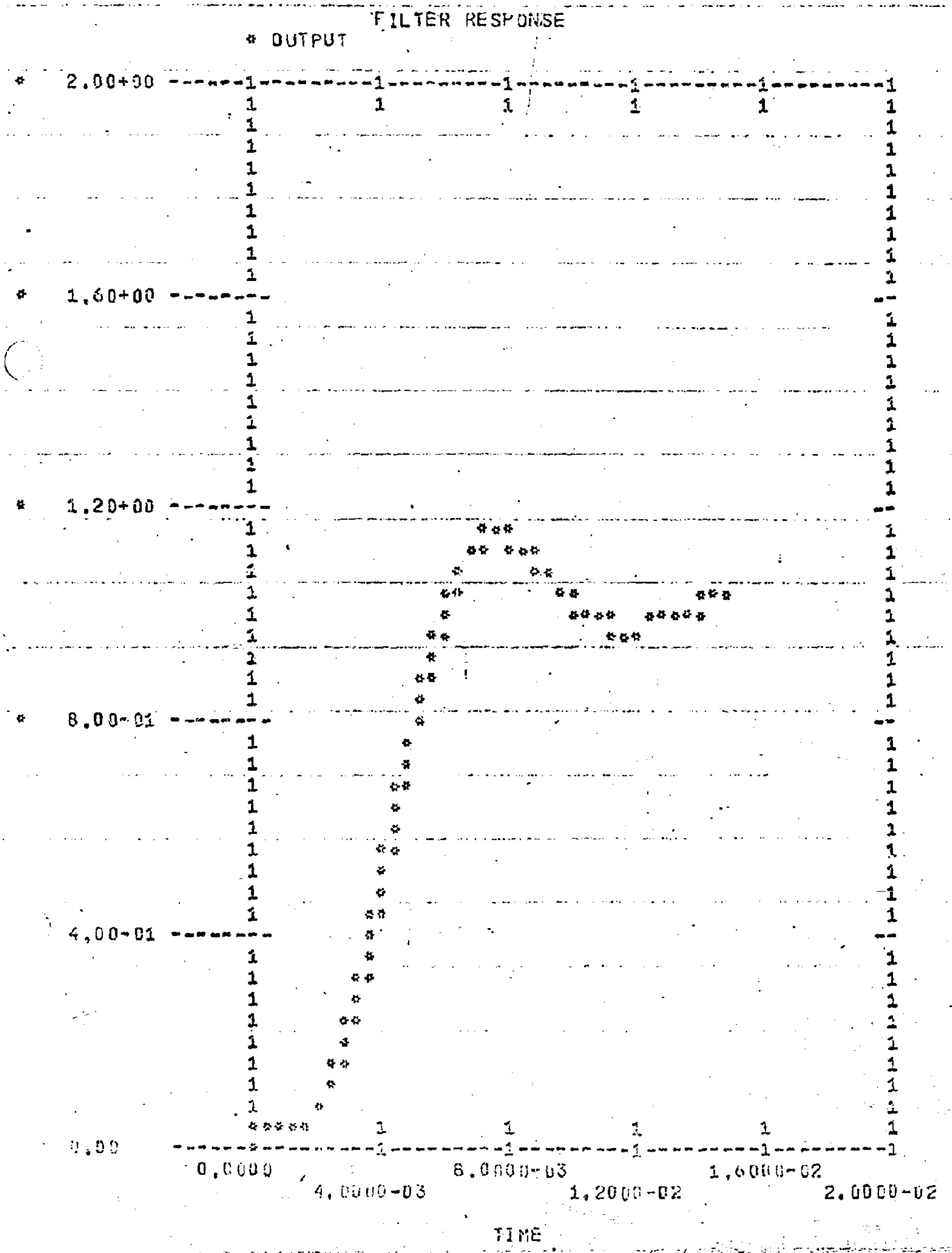
TIME	.128000-01	.129000-01	.130000-01	.131000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.965866+00	.967806+00	.969858+00	.972002+00

TIME	.132000-01	.133000-01	.134000-01	.135000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.974221+00	.976497+00	.978812+00	.981150+00

TIME	.136000-01	.137000-01	.138000-01	.139000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.983494+00	.985830+00	.988143+00	.990419+00

TIME	.140000-01	.141000-01	.142000-01	.143000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.992645+00	.994811+00	.996904+00	.998916+00

TIME	.144000-01	.145000-01	.146000-01	.147000-01
N1	.100000+01	.100000+01	.100000+01	.100000+01
OUTPUT	.100084+01	.100266+01	.100438+01	.100598+01



FILTER RESPONSE
* OUTPUT
2.00+00
1.60+00
1.20+00
8.00-01
4.00-01
0.00
0.0000
4.0000-03
8.0000-03
1.2000-02
1.6000-02
2.0000-02
TIME

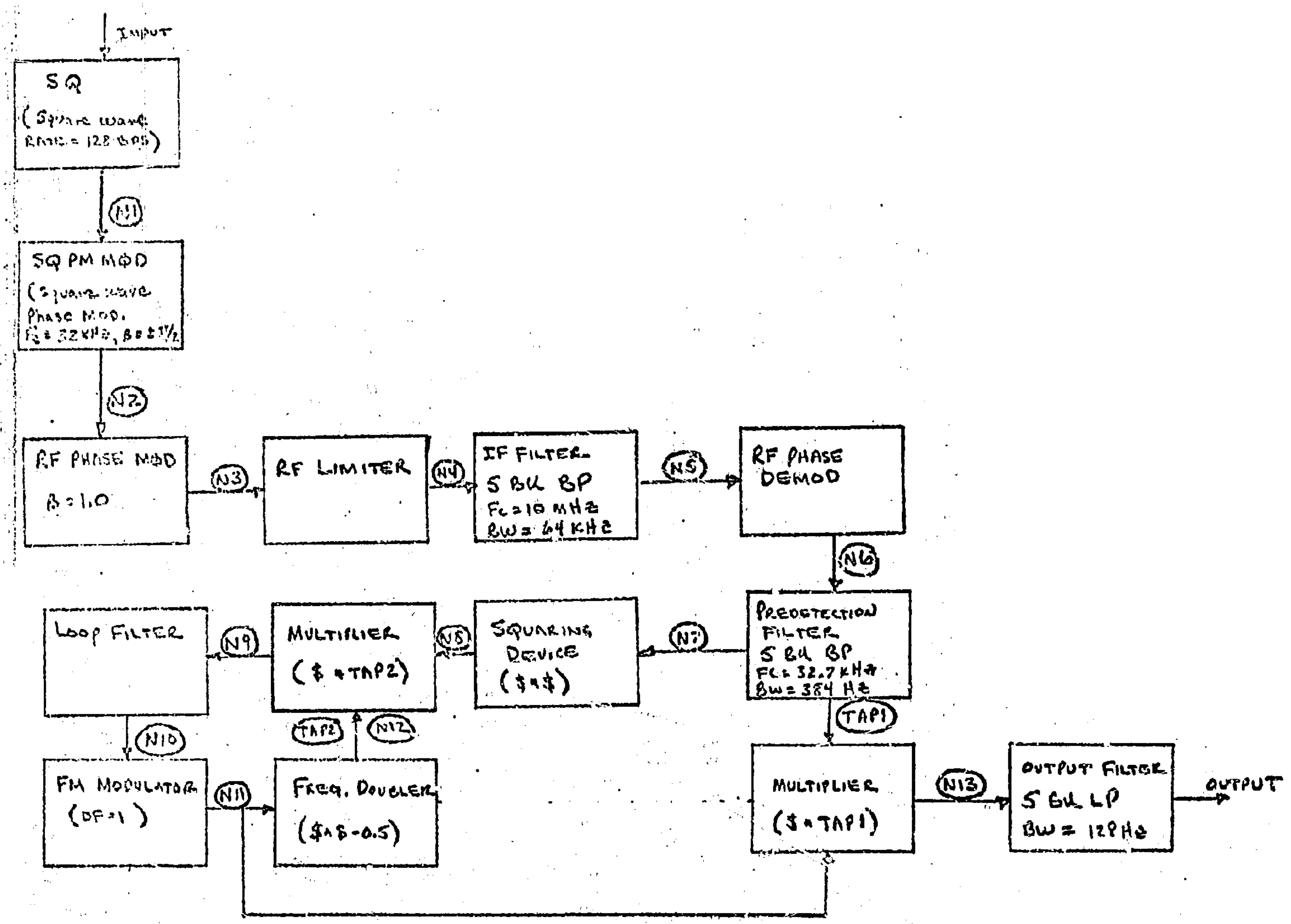


Figure 5-2. Apollo PCM/FM/PM Link Block Diagram

```
                                          MODELA    1  1  0              0   2
D PROCESSOR LEVEL I
ON OF JANUARY 1, 1971 FOR THE UNIVAC 1108.
S DECK WAS PROCESSED ON 22 FEB 71 AT 11:35:31.


TID MODELS REFERENCED                          ENTRY POINT

  SO                                             SO

MODEL ASSIGNED THE ENTRY POINT NAME MODELA.


             MODEL=NRZ.BR
                  INPUT < SQ(BR/2.) > N1
                  N1 < $*.5+.5 > OUTPUT
             END
                                                                    TAP ORDER-TAP001
                                                                    TAP ORDER-ZAP002
```

Figure 5-2. Listing of SYSTID Simulation of Apollo PCM/PM/PM Link

```
APOLLO PCM/PM/PM LINK
SYSTID PROCESSOR LEVEL I
VERSION OF JANUARY 1, 1971 FOR THE UNIVAC 1108.
THIS DECK WAS PROCESSED ON 22 FEB 71 AT 11:35:32.


SYSTID MODELS REFERENCED                               ENTRY POINT

      NRZ                                                MODELA
      SOPMMOD                                            SOPMOD
      RFPHASEMODULATOR                                   RPMMOD
      RFLIMITER                                          RFLIMT
      BUTTERWORTH                                        BUTWTH
      RFPHASEDEMOD                                       RFPDEM
      LOOPFILTER                                         LEDLAG
      FMMODULATOR                                        FMMOD

TAP ORDER:   1 TAP001   2 TAP002


              SYSTEM .  APOLLO PCM/PM/PM LINK
              PAGE,  SMALL
              DEFAUL, TSTART=0.,TSTOP=.03,DT=1.5E-6,NPRINT=200
              DATA = DT,TSTOP,SETTLE
              PRINT, TIME,N1,N10,N13,OUTPUT
              PPLOT, OUTPUT,N1,N10,N13
                   INPUT < NRZ(128.) > N1
                   N1 < SOPMMOD(PI,32.768E3)> N2
                   N2 <RF PHASE MODULATOR (1.0,10.E6) > N3
                   N3 < RF LIMITER > N4
                   N4 < BUTTERWORTH (5,3,10.E6,64.E3,10.E6,1.) > N5
                   N5 < RF PHASE DEMOD (1.0) > N6
                   N6 < BUTTERWORTH (5,3,32.768E3,384.,0.,1.) > N7 'TAP1
                   N7 < $*$ > N8
                   N8 < $*TAP2 > N9
                   N9 < LOOP FILTER (200.,1.8775994,0.,.19751719,0.) > N10
                   N10 < FM MODULATOR (2.*PI,32.768E3) > N11
                   N11 < $*$-0.5 > N12 'TAP2
                   N11 < $*TAP1 > N13
                   N13 < BUTTERWORTH (2,1,0.,128.,0.,1.) > OUTPUT
              END

ADD ADDFIL
```

```
FOR,* MODELA/SYSTID,MODELA/SYSTID
UNIVAC 1108 FORTRAN V LEVEL   2206 0018 F50185
THIS COMPILATION WAS DONE ON 22 FEB 71 AT 11:35:33

SUBROUTINE MODELA     ENTRY POINT 000056

STORAGE USED (BLOCK, NAME, LENGTH)

      0001   *CODE    000067
      0000   *DATA    000022
      0002   *BLANK   000000
      0003   COGENT   000007
      0004   VSPACE   000003
      0005   FLR      000014


EXTERNAL REFERENCES (BLOCK, NAME)

      0006   SQ
      0007   NERR3$


STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

 0001   000003 1L          0001   000036 2L          0003 R 000002 DT         000
 0005 I 000000 FIRSTF      0005 I 000001 NUMF        0003 R 000005 PI         0003
 0004 R 000001 V           0003 I 000006 VCIN        0003 I 000000 VIN        0003
 0000 I 000000 Z           0003 I 000004 ZZ


00101      1*            SUBROUTINE  MODELA(BR)
00101      2*      CMODEL       NRZ
00103      3*            INTEGER FIRSTF,VOUT,VIN,Z,ZZ,VSAVE,VCIN
00104      4*            COMMON /COGENT/ VIN,VOUT,DT,TIME,ZZ,PI,VCIN
00105      5*            COMMON /VSPACE/ DUMMY,V(2)                    @ SIZE FIXED IN MAI
0010       6*            COMMON /FLR   / FIRSTF,NUMF,F(5,2)            @ SIZE FIXED IN MAI
0010       7*            EQUIVALENCE(TIME,T)                           @ ALLOW BOTH
00110      8*            Z=ZZ
00111      9*            GO TO 2
00112     10*          1 CONTINUE
00113     11*            VSAVE=VOUT                                    @ SAVE THE OUTPUT A
00114     12*            V(Z+2)=V(VIN)
00114     13*      C     SQ
00115     14*            VIN=Z+2
00116     15*            VOUT=Z+3
00117     16*            CALL  SQ(BR/2.)
00120     17*            V(Z+4)=V(Z+3)*.5+.5
00121     18*            V(VSAVE)=V(Z+4)
00122     19*            RETURN
00123     20*          2 ZZ=ZZ+     4
  124     21*            GO TO 1
00125     22*            END

    END OF UNIVAC 1108 FORTRAN V COMPILATION,     0 *DIAGNOSTIC* MESSAGE(S)
```

```
FOR,* MAIN  /SYSTID,MAIN  /SYSTID
VAC 1108 FORTRAN V LEVEL    2206 0018 F5018S
S COMPILATION WAS DONE ON 22 FEB 71 AT 11:35:35

MAIN PROGRAM

STORAGE USED (BLOCK, NAME, LENGTH)

     0001    *CODE     000570
     0000    *DATA     001241
     0002    *BLANK    000000
     0003    COGENT    000007
     0004    VSPACE    023421
     0005    FLR       004706
     0006    NAME      000006
     0007    INT       000001
     0010    DIF       000001


EXTERNAL REFERENCES (BLOCK, NAME)

     0011    MODELA
     0012    SQPMOD
     0013    RPMMOD
     0014    RFLIMT
     0015    BUTWTH
     0016    RFPDEM
     0017    LEDLAG
     0020    FMMOD
     0021    DRUMIT
     0022    PTPLT
     0023    NRNL$
     0024    NWNL$
     0025    NWDU$
     0026    NIO2$
     0027    NSTOP$
     0030    NIO1$


STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

 0001   000067 1L          0001   000463 100L        0000   001160 1000F       000
 0001   000030 11L         0001   000341 15L         0001   000032 157G        000
 0000   001136 1999F       0001   000562 2L          0001   000422 20L         000
 0001   000502 355G        0001   000507 361G        0001   000054 4L          000
 0003 R 000002 DT          0007 R 000000 DT2         0004 R 000000 DUMMY       000
 0005 I 000000 FIRSTF      0000 I 001114 I           0000 I 001122 INDEX       000
 0000 I 001112 IS          0000 I 001115 J           0000 I 000122 KOUNT       000
 0000 I 000124 NPRINT      0005 I 000001 NUMBF       0003 R 000005 PI          000
 0003   000003 T           0010 R 000000 TDT         0003 R 000003 TIME        000
 0000 R 001111 TSTOP       0004 R 000001 V           0003 I 000006 VEIN        000
 0000 I 000000 Z           0003 I 000004 ZZ
```

```
      INCLUDE MAIN1,LIST
      PARAMETER VSIZE=10000
      PARAMETER FSIZE=500                                  @
      INTEGER FIRSTF,VOUT,VIN,Z,ZZ,VCIN,FIRST,TITLE(6)
      DIMENSION PRINT(8,10)                                @ THE PRINT BUFF
      COMMON /COGENT/ VIN,VOUT,DT,TIME,ZZ,FI,VCIN
      COMMON /VSPACE/DUMMY,V(VSIZE)                        @ SIZE IS FIXED
      COMMON /FLR   / FIRSTF,NUMBF,F(5,FSIZE)              @ SIZE IS FIXED
      COMMON /NAME/ TITLE
      COMMON /INT/ DT2
      COMMON /DIF/ TDT
      DATA PI/3.1415927/,FIRST/0/,KOUNT/1/,SETTLE/0.0/,NPRINT/1/
      EQUIVALENCE (T,TIME)
      END
      DIMENSION DRUM(100, 5)
      NAMELIST/SYSTID/DT,TSTOP,SETTLE
      DATA TITLE/'APOLLO PCM/PM/PM LINK                   '/
      DATA  TSTART/0./,TSTOP/.03/,DT/1.5E-6/,NPRINT/200/
      READ (5,SYSTID)
      WRITE(6,SYSTID)
      INCLUDE MAIN2,LIST
      IF((TSTOP-TSTART)/DT,GT,0) GO TO 11                  @ ARE THEY REASO
      WRITE(6 ,1999) TSTART,TSTOP,DT                       @ NO, TELL THEM
 1999 FORMAT(1X,'** ERROR **  THE VALUES TSTART=',E12.6,' TSTOP=',
     ,' DT=',E12.6,' ARE UNREASONABLE.')                   @
      STOP HI
   11 DO 6 I=1,VSIZE
    6 V(I)=0,0                                             @ JUST TO MAKE S
      DO 7 I=1,FSIZE                                       @ IN CASE CORE H
      DO 7 J=1,5
    7 F(J,I)=0,0                                           @ NOT BEEN ZEROE
      DUMMY=0,0                                            @ FOR FIRST TIME
      DT2=DT/2,0
      TDT=2,0/DT
      TIME=TSTART-DT                                       @ TO GET STARTED
      SETTLE=TIME+SETTLE                                   @ SETTLING TIME
      @.............................................................
      @ THE SIMULATION LOOP BEGINS HERE.  GOOD LUCK SIR.
      @.............................................................
    4 TIME=TIME+DT
      ZZ=2
      FIRSTF=1                                             @ INITIALIZE F A
      VIN=1                                                @ FOR MY SAKE
      VOUT=2                                               @ FOR MIKE'S SAK
      Z=ZZ                                                 @ FOR SAFETY'S S
      GO 10 2
    1 CONTINUE
      IF(TIME,GT,TSTOP) GO TO 100                          @ IF FINISHED LE
      END
      IOT1=Z+2
      IOT2=Z+3
C     NRZ
      VIN=Z+4
      VOUT=Z+5
      CALL  MODELA(128,)
C     SQPMMOD
      VIN=Z+5
      VOUT=Z+6
      CALL  SQPMOD(PI,32.768E3)
```

```
C      RFPHASEMODULATOR
       VIN=Z+6
       VOUT=Z+7
       CALL  RPMMOD(1,0,10,E6)
C      RFLIMITER
       VIN=Z+7
       VOUT=Z+8
       CALL  RFLIMT
C      BUTTERWORTH
       VIN=Z+8
       VOUT=Z+9
       CALL  BUTWTH(5,3,10.E6,64.E3,10.E6,1.)
C      RFPHASEDEMOD
       VIN=Z+9
       VOUT=Z+10
       CALL  RFPDEM(1,0)
C      BUTTERWORTH
       VIN=Z+10
       VOUT=Z+11
       CALL  BUTWTH(5,3,32.768E3,384.,0.,1.)
       V(IOT1)=V(Z+11)
       V(Z+12)=V(Z+11)*V(Z+11)
       V(Z+13)=V(Z+12)*V(IOT2)
C      LOOPFILTER
       VIN=Z+13
       VOUT=Z+14
       CALL  LEDLAG(200.,1.5775994,0.,.19751719,0.)
C      FMMODULATOR
       VIN=Z+14
       VOUT=Z+15
       CALL  FMMOD(2.*PI,32.768E3)
       V(Z+16)=V(Z+15)*V(IOT1)
       V(Z+17)=V(Z+15)*V(Z+15)-0.5
       V(IOT2)=V(Z+17)
C      BUTTERWORTH
       VIN=Z+16
       VOUT=Z+18
       CALL  BUTWTH(2,1,0.,128.,0.,1.)
       INCLUDE MAIN5,LIST
       IF(TIME.LT.SETTLE) GO TO 4                        @ SKIP IT ??
       IF(MOD(FIRST,NPRINT).NE.0) GO TO 20               @ PRINT IT ??
       END
C      PRINTED OUTPUT
       IF(FIRST.NE.0) GO TO 15
       PRINT(1,1)='TIME'
       PRINT(1,2)='N1'
       PRINT(1,3)='N10'
       PRINT(1,4)='N13'
       PRINT(1,5)='OUTPUT'
       NODES=5
       NPAGE=5
    15 KOUNT =KOUNT+1
       PRINT(KOUNT,1)=TIME
       PRINT(KOUNT,2)=V(Z+5)
       PRINT(KOUNT,3)=V(Z+14)
       PRINT(KOUNT,4)=V(Z+16)
       PRINT(KOUNT,5)=V(Z+18)
       IF(KOUNT.LT.NPAGE) GO TO 20
       INCLUDE MAIN3,LIST
```

```
00312     74*           DO 16 J=1,NODES
00315     74*           @..........................................................................
00315     74*        16 WRITE(6,1000) (PRINT(I,J),I=1,NPAGE)
00324     74*      1000 FORMAT(6X,A6, 7(4X,E12.6))
00325     74*           WRITE(6,1001)
00327     74*      1001 FORMAT(//)                                                   @ FOR NICE LOOKING
00330     74*           END
00331     75*           KOUNT=1
00332     76*        20 CONTINUE
00333     77*           IF(MOD(FIRST,100),NE,0) GO TO 50
00335     78*           IF(FIRST,EQ,0) GO TO 50
00337     79*           CALL DRUMIT(5,DRUM,13)
00340     80*        50 INDEX=MOD(FIRST,100)+1
00341     81*           DRUM(INDEX,1)=TIME
00342     82*           DRUM(INDEX,2)=V(Z+18)
00343     83*           DRUM(INDEX,3)=V(Z+5)
00344     84*           DRUM(INDEX,4)=V(Z+14)
00345     85*           DRUM(INDEX,5)=V(Z+16)
00346     86*           INCLUDE MAIN4,LIST
00347     86*           FIRST=FIRST+1                                                @ LOOP COUNTER
00350     86*           GO TO 4                                                      @ LOOP
00351     86*           @..........................................................................
00351     86*       100 CONTINUE                                                     @ FINISHED
00352     86*           IF(KOUNT,LE,1) GO TO 102
00354     86*           DO 101 J=1,NODES
00357     86*       101 WRITE(6,1000)(PRINT(I,J),I=1,KOUNT)
00366     86*       102 CONTINUE
00367     86*           END
00370     87*           CALL DRUMIT(5,DRUM,13)
00371     88*           CALL  PTPLT(13,2,'OUTPUT',FIRST,NPAGE)
00372     89*           CALL  PTPLT(13,3,'N1',FIRST,NPAGE)
00373     90*           CALL  PTPLT(13,4,'N10',FIRST,NPAGE)
00374     91*           CALL  PTPLT(13,5,'N13',FIRST,NPAGE)
00375     92*           STOP
00376     93*         2 ZZ=ZZ+     18
00377     94*           GO TO 1
00400     95*           END

      END OF UNIVAC 1108 FORTRAN V COMPILATION,      0 *DIAGNOSTIC* MESSAGE(S)
```

```
$SYSTID
 DT      =     ,15000000E-05,
 TSTOP   =     ,30000000E-01,
 SETTLE  =     ,00000000E+00,
 $END
      TIME       ,000000         ,300000-03      ,599999-03      ,899998-03
      N1         ,500000+00      ,100000+01      ,100000+01      ,100000+01
      N10        ,000000         ,388039-09     -,323883-07     -,134174-03
      N13        ,000000         ,552889-05      ,114473-03     -,229617-03
      OUTPUT     ,000000         ,170290-07      ,501901-07     -,843802-05


      TIME       ,120000-02      ,150000-02      ,180000-02      ,209999-02
      N1         ,100000+01      ,100000+01      ,100000+01      ,100000+01
      N10        ,222456-02     -,270604-05     -,492722-01      ,212163+00
      N13       -,133625-01      ,993476-03     -,930145-02     -,132088+00
      OUTPUT    -,808462-04     -,383416-03     -,124927-02     -,319650-02


      TIME       ,239999-02      ,269999-02      ,299998-02      ,329998-02
      N1         ,100000+01      ,100000+01      ,100000+01      ,100000+01
      N10       -,332601-03     -,698280+00      ,185614-01     -,117060-01
      N13        ,867057-02     -,561403-01     -,396341+00      ,389958-01
      OUTPUT    -,689649-02     -,130842-01     -,224434-01     -,354539-01


      TIME       ,359998-02      ,389997-02      ,419996-02      ,449995-02
      N1         ,100000+01      ,100000+01      ,100000+01      ,100000+01
      N10       -,253757+01      ,520419+01     -,817281-01     -,430237+01
      N13       -,147199+00     -,672339+00      ,903282-01     -,237639+00
      OUTPUT    -,524182-01     -,731061-01     -,970330-01     -,123345+00


      TIME       ,479995-02      ,509994-02      ,539993-02      ,569992-02
      N1         ,100000+01      ,100000+01      ,100000+01      ,100000+01
      N10        ,737315+01     -,233798+00     -,447370+01      ,653365+01
      N13       -,809153+00      ,140218+00     -,289963+00     -,727646+00
      OUTPUT    -,150957+00     -,176641+00     -,205178+00     -,229465+00


      TIME       ,599991-02      ,629990-02      ,659989-02      ,689988-02
      N1         ,100000+01      ,100000+01      ,100000+01      ,100000+01
      N10       -,410243+00     -,339214+01      ,515596+01     -,590456+00
      N13        ,171314+00     -,311799+00     -,694064+00      ,167309+00
      OUTPUT    -,250683+00     -,268139+00     -,281538+00     -,290863+00


      TIME       ,719987-02      ,749986-02      ,779985-02      ,809983-02
      N1         ,100000+01      ,100000+01      ,100000+01      ,000000
      N10       -,220831+01      ,386854+01     -,847661+00     -,125772+01
      N13       -,330140+00     -,612469+00      ,202469+00     -,367442+00
      OUTPUT    -,296353+00     -,298468+00     -,297788+00     -,294949+00
```

TIME	.339981-02	.869979-02	.899977-02	.929975-02
N1	.000000	.000000	.000000	.000000
N10	.314542+01	-.125558+01	-.238227+00	.212848+01
N13	-.566368+00	.218275+00	-.408989+00	-.479956+00
OUTPUT	-.290640+00	-.285458+00	-.279810+00	-.273804+00

TIME	.959973-02	.989971-02	.101997-01	.104997-01
N1	.000000	.000000	.000000	.000000
N10	-.119569+01	.408838+00	.427024+00	-.984453-01
N13	.188817+00	-.315235+00	-.215864+00	.546036-01
OUTPUT	-.267111+00	-.256999+00	-.248410+00	-.234134+00

TIME	.107996-01	.110996-01	.113996-01	.116996-01
N1	.000000	.000000	.000000	.000000
N10	.420319-01	.193243+00	-.926798+00	.217796+01
N13	.229749-01	.144007+00	-.124337+00	.468425+00
OUTPUT	-.215011+00	-.190172+00	-.159145+00	-.122044+00

TIME	.119996-01	.122995-01	.125995-01	.128995-01
N1	.000000	.000000	.000000	.000000
N10	.986680+00	-.478685+01	.639312+01	.963816+00
N13	.378218+00	-.240334+00	.759499+00	.397829+00
OUTPUT	-.796096-01	-.331894-01	.154142-01	.641674-01

TIME	.131995-01	.134995-01	.137994-01	.140994-01
N1	.000000	.000000	.000000	.000000
N10	-.728118+01	.776651+01	.451468+00	-.671892+01
N13	-.254643+00	.807030+00	.287792+00	-.201290+00
OUTPUT	.110996+00	.154014+00	.191722+00	.223073+00

TIME	.143994-01	.146994-01	.149993-01	.152993-01
N1	.000000	.000000	.000000	.000000
N10	.652872+01	.154423+00	-.549227+01	.546664+01
N13	.721485+00	.168382+00	-.138980+00	.650840+00
OUTPUT	.247535+00	.265061+00	.276062+00	.281308+00

TIME	.155993-01	.158993-01	.161993-01	.164992-01
N1	.000000	.100000+01	.100000+01	.100000+01
N10	.681337-01	-.526013+01	.561137+01	.502236-01
N13	.898754-01	-.915318-01	.656469+00	.367849-01
OUTPUT	.281824+00	.278752+00	.273236+00	.266352+00

TIME	.167992-01	.170992-01	.173992-01	.176992-01

N1	,100000+01	,100000+01	,100000+01	,100000+01
N10	-,544594+01	,523739+01	,493544+01	-,275585+01
N13	-,460515-01	,635356+00	-,113257-01	,294926-02
OUTPUT	,258911+00	,251353+00	,243646+00	,235277+00

TIME	,179991-01	,182991-01	,185991-01	,188991-01
N1	,100000+01	,100000+01	,100000+01	,100000+01
N10	,152987+01	,489287-01	,486606-01	,551391+00
N13	,341921+00	-,164417-01	-,827620-03	-,201461+00
OUTPUT	,225291+00	,212515+00	,195757+00	,174076+00

TIME	,191991-01	,194990-01	,197990-01	,200990-01
N1	,100000+01	,100000+01	,100000+01	,100000+01
N10	,312217-01	-,330063+01	,551035+01	-,186161+00
N13	,595381-01	-,970037-01	-,675041+00	,172868+00
OUTPUT	,146914+00	,114215+00	,764991-01	,348498-01

TIME	,203990-01	,206989-01	,209989-01	,212989-01
N1	,100000+01	,100000+01	,100000+01	,100000+01
N10	-,717375+01	,824332+01	-,631051+00	-,614618+01
N13	-,226001+00	-,842428+00	,248085+00	-,309430+00
OUTPUT	-,919964-02	-,538516-01	-,972317-01	-,137566+00

TIME	,215989-01	,218989-01	,221988-01	,224988-01
N1	,100000+01	,100000+01	,100000+01	,100000+01
N10	,622567+01	-,983193+00	-,324660+01	,358893+01
N13	-,748305+00	,261738+00	-,336378+00	-,582688+00
OUTPUT	-,173383+00	-,203698+00	-,227612+00	-,245258+00

TIME	,227988-01	,230988-01	,233988-01	,236987-01
N1	,100000+01	,100000+01	,100000+01	,000000
N10	-,120617+01	-,132189+01	,226564+01	-,159886+01
N13	,248557+00	-,354113+00	-,477097+00	,245976+00
OUTPUT	-,256835+00	-,262941+00	-,264476+00	-,262493+00

TIME	,239987-01	,242987-01	,245987-01	,248987-01
N1	,000000	,000000	,000000	,000000
N10	-,115214+00	,174375+01	-,218506+01	,967079+00
N13	-,405657+00	-,434526+00	,247367+00	-,440506+00
OUTPUT	-,258077+00	-,252238+00	-,245791+00	-,239193+00

TIME	,251986-01	,254986-01	,257986-01	,260986-01
N1	,000000	,000000	,000000	,000000
N10	,955574+00	-,143681+01	,627737+00	,110575+00
N13	-,333498+00	,173605+00	-,266392+00	-,910112-01

OUTPUT	-.232441+00	-.225029+00	-.216052+00	-.204380+00

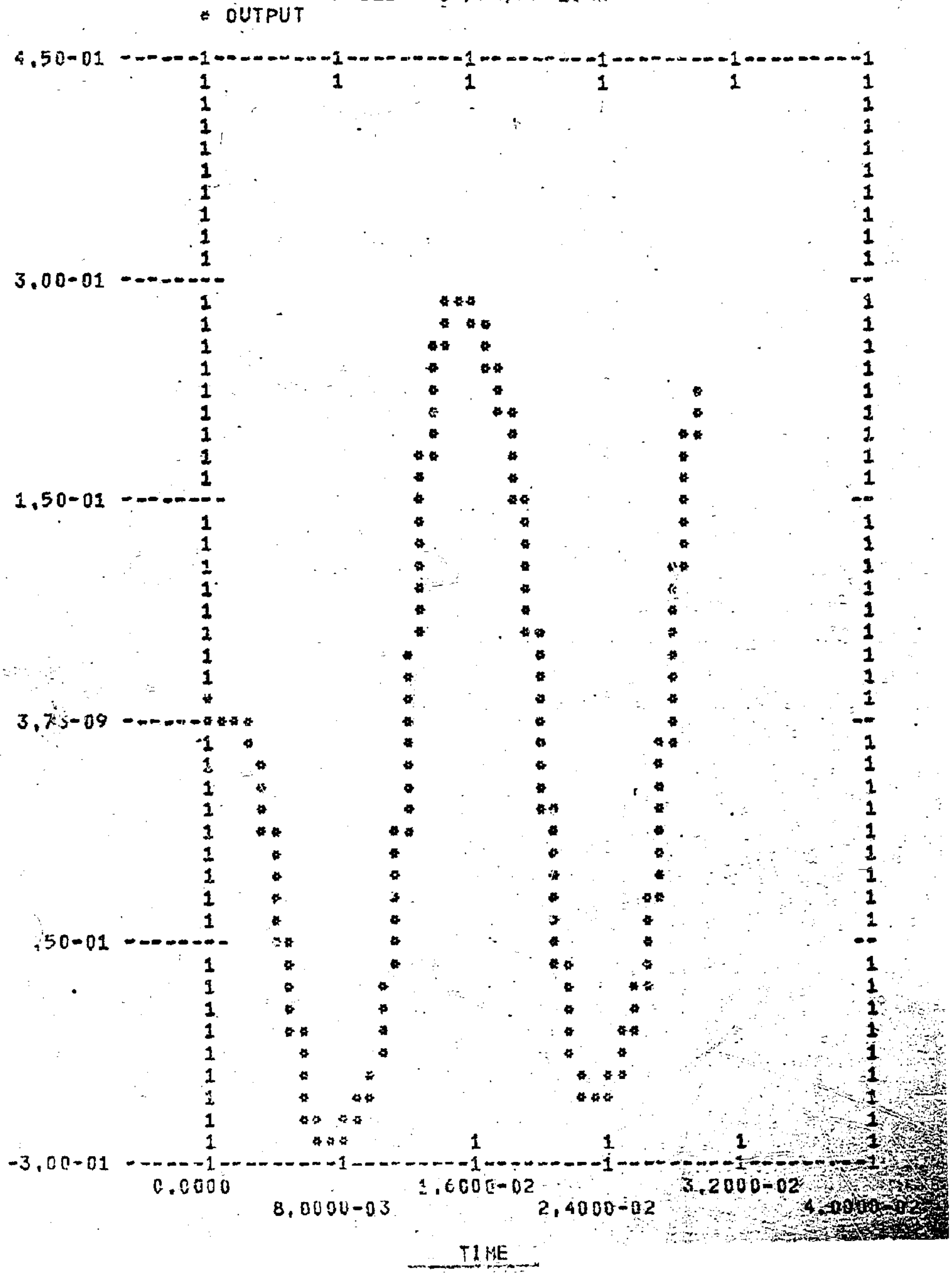
TIME	.263986-01	.266985-01	.269985-01	.272985-01
N1	.000000	.000000	.000000	.000000
N10	.493863-01	.316746+00	.205567+00	-.274617+01
N13	-.578252-03	.161435+00	.157804+00	-.169191+00
OUTPUT	-.188863+00	-.168553+00	-.142883+00	-.111772+00

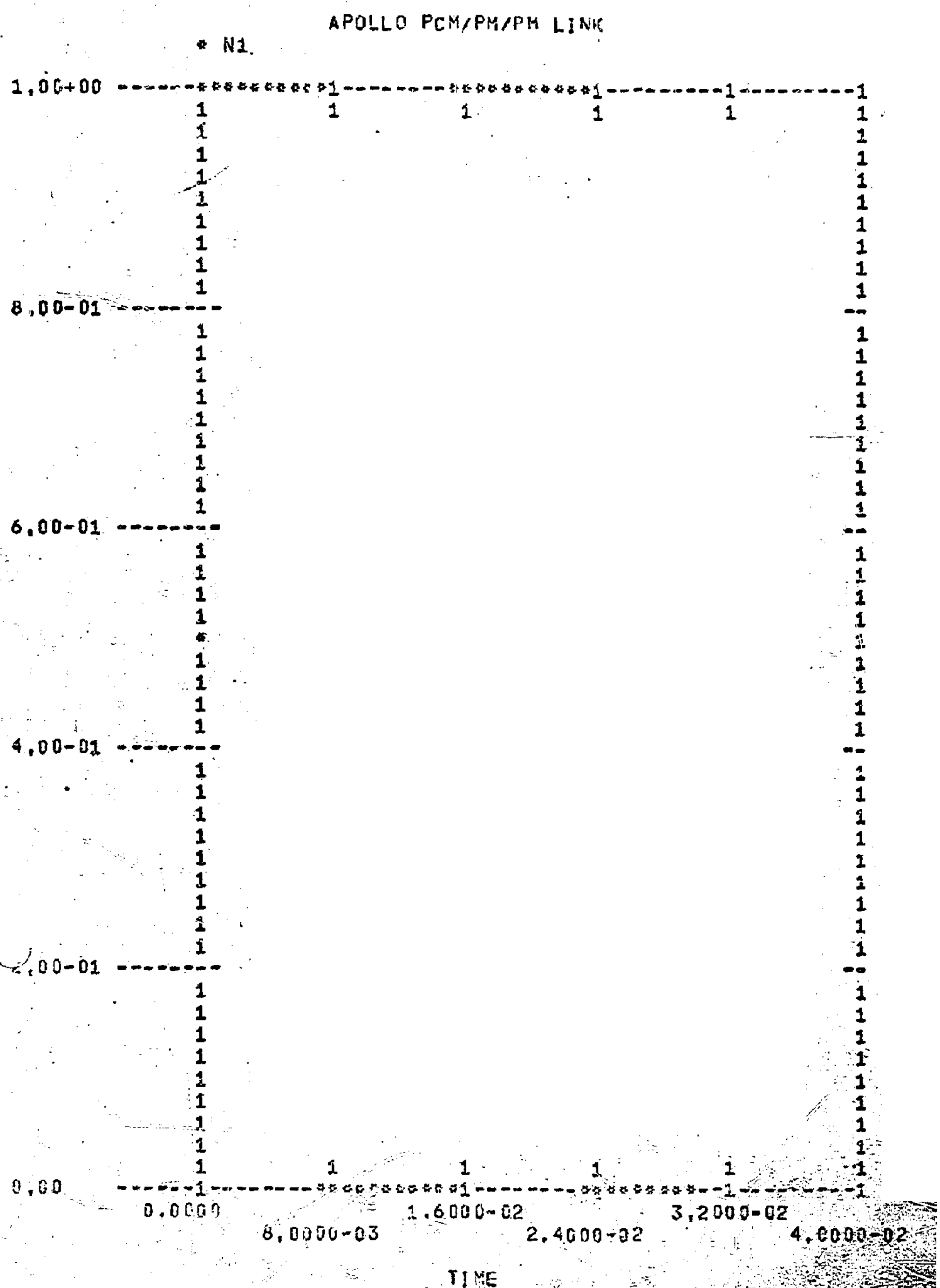
TIME	.275985-01	.278984-01	.281984-01	.284984-01
N1	.000000	.000000	.000000	.000000
N10	.439276+01	.396918+00	-.793757+01	.881933+01
N13	.612796+00	.258838+00	-.231202+00	.842936+00
OUTPUT	-.756977-01	-.356720-01	.685484-02	.501536-01

TIME	.287984-01	.290984-01	.293983-01	.296983-01
N1	.000000	.000000	.000000	.000000
N10	.220886+00	-.932974+01	.875390+01	.759254-01
N13	.207667+00	-.190994+00	.826095+00	.102257+00
OUTPUT	.924148-01	.131909+00	.167163+00	.197191+00

TIME	.299983-01
N1	.000000
N10	-.745170+01
N13	-.115288+00
OUTPUT	.221060+00

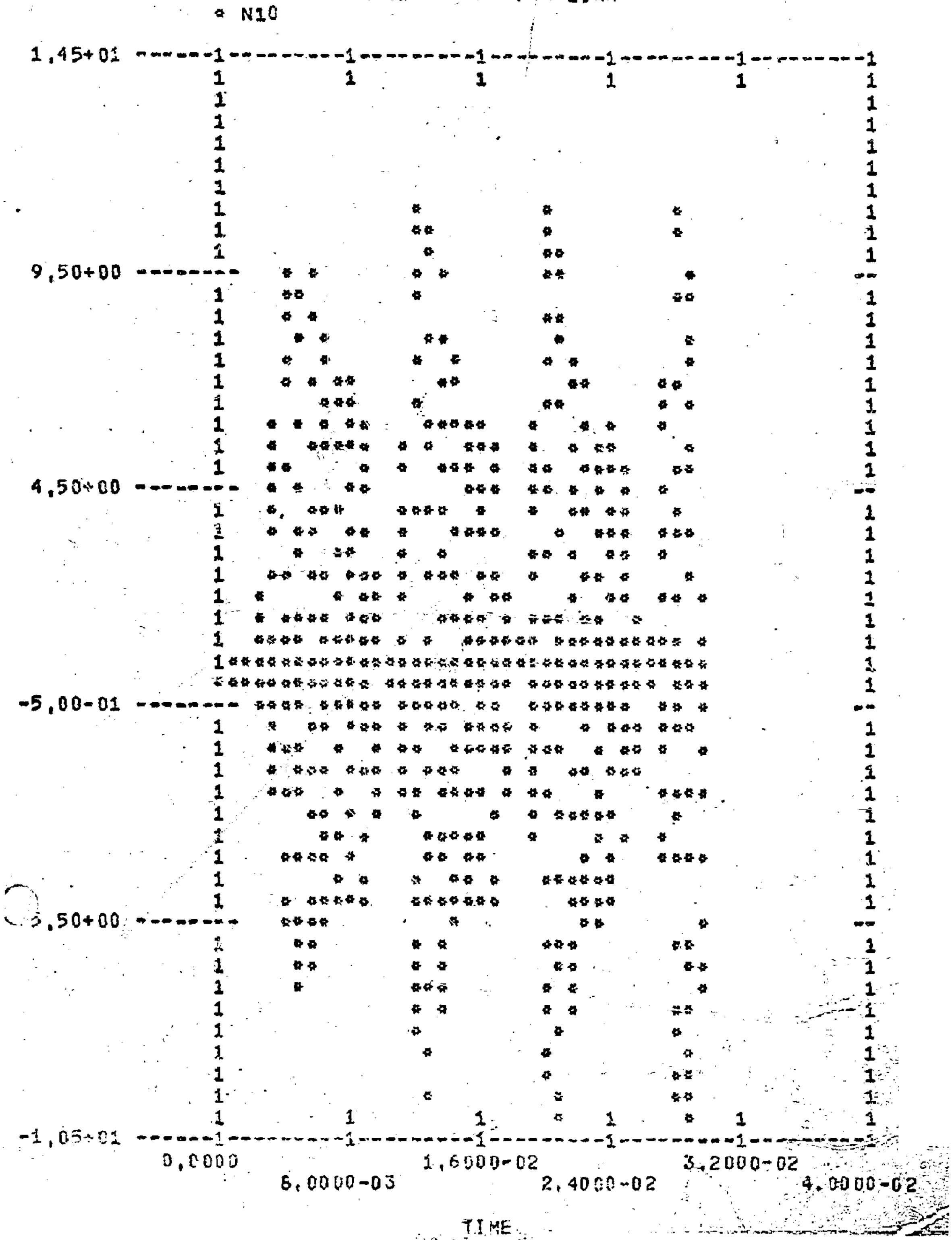
APOLLO PCM/PM/PM LINK



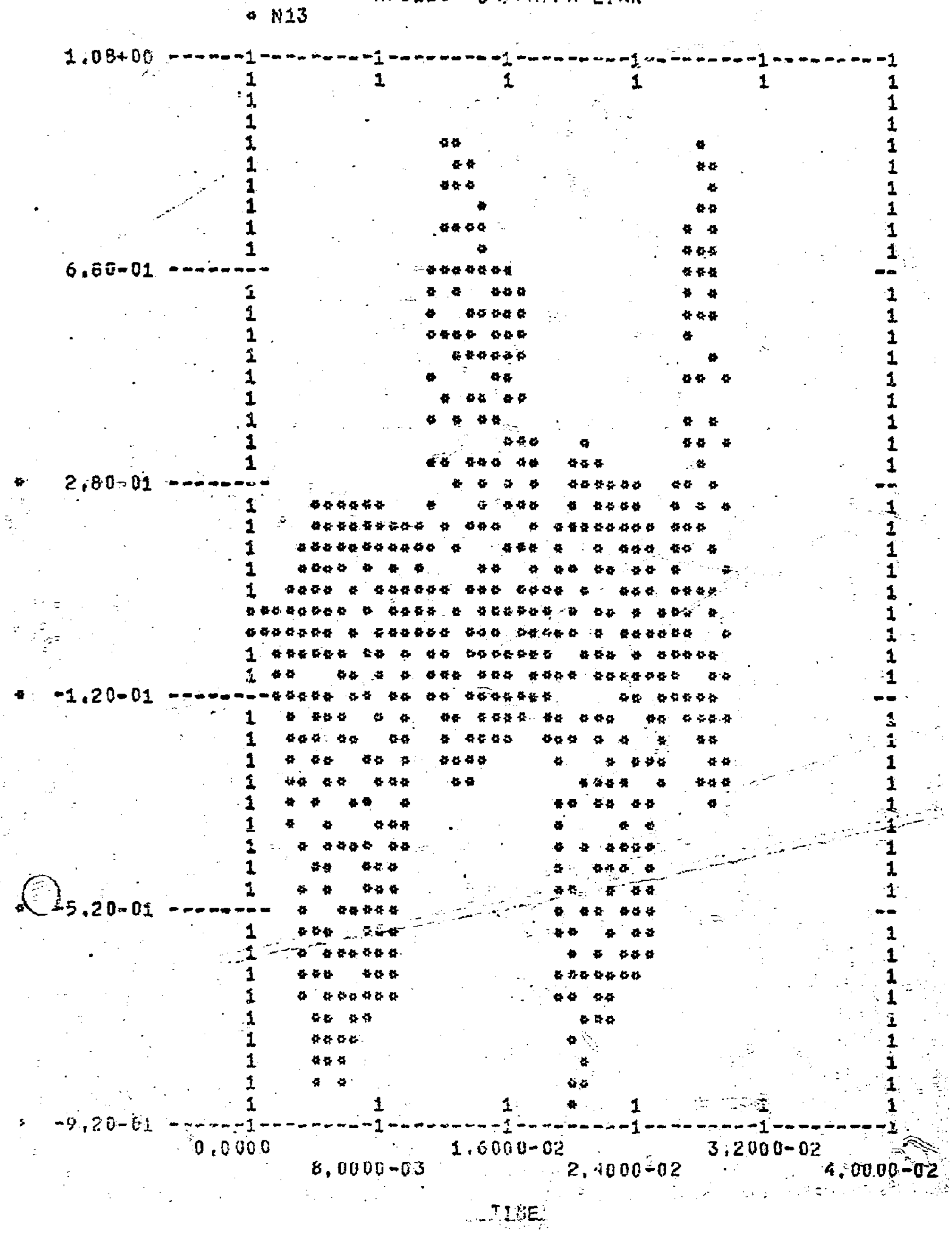


APOLLO PCM/PM/PM LINK
* N1
1,00+00
8,00-01
6,00-01
4,00-01
2,00-01
0,00
0,0000
8,0000-03
1,6000-02
2,4000-02
3,2000-02
4,0000-02
TIME

APOLLO PCM/PM/PM LINK



APOLLO PCM/PM/PM LINK



5.6 BIBLIOGRAPHY AND REFERENCES

(1) Golden, R. M. and Kaiser, J. F., "Design of Wideband Sampled Data Filters", B. S. T. J., pp. 1533-1545, vol. 43, part 2, July 1964.

(2) Kaiser, J. F., "Design Methods for Sampled-Data Filters", Proc. First Allerton Conference on Circuit and System Theory, Nov., 1963, Monticello, Illinois, pp. 221-236.

(3) Golden, R. M., "Digital Computer Simulation of a Sampled-Data Voice-Excited Vocoder", J. Acoust. Soc. Am., 35, Sept., 1963, pp. 1358-1366.

(4) Wilts, C. H., Principles of Feedback Control, Addison-Wesley, 1960, pp. 197-207.

(5) Hamming, R. W., Numerical Methods for Scientists and Engineers, McGraw-Hill, New York, 1962, pp. 277-280.

(6) Kelly, J. L., Jr., Lochbaum, C. and Vyssotsky, V. A. "a Block Diagram Compiler", B. S. T. J., 40, May, 1961, pp. 669-676.

(7) Storer, J. E., Passive Network Synthesis, McGraw-Hill, New York, 1957, pp. 293-296.

(8) Kuo, F. F. and Kaiser, J. F., System Analysis By Digital Computer, J. Wiley & Sons, Inc., New York, 1966.

(9) D. C. Baxter, Digital Simulation Using Approximate Methods, National Research Council, Ottawa, Canada, Report MK-15, Division of Mechanical Engineering, July 1965.

(10) J. E. Gibson, Nonlinear Automatic Control, McGraw Hill, New York, 1963, pp. 147-159.

(11) K. Steiglitz, The General Theory of Digital Filters with Applications to Spectral Analysis, AFOSR Report No. 64-1664, New York University, New York, May 1963.

(12) K. Steglitz, "The Equivalence of a Digital and Analog Signal Processing", Information and Control, Vol. 8, No. 5, October 1965, pp. 455-467.

(13) J. F. Kaiser, "Some Practical Considerations in the Realization of Linear Digital Filters", Proceedings Third Allerton Conference on Circuit and System Theory, Monticello, Illinois, October 1965, pp. 621-633.

(14) G. E. Heyliger, The Scanning Function Approach to the Design of Numerical Filters, Report R-63-2, Martin Co., Denver, Colorado, April 1963.

(15) R. J. Graham, Determination and Analysis of Numerical Smoothing Weights, NASA Technical Report No. TR-R-179, December 1963.

(16) E. B. Anders et al., Digital Filters, NASA Contractor Report CR-136, December 1964.

(17) D. G. Watts, Optimal Windows for Power Spectra Estimation, Mathematics Research Center, University of Wisconsin, MRC-TSR-506, September 1964.

(18) E. Parzen, Notes on Fourier Analysis and Spectral Windows, Applied Mathematics and Statistical Laboratories, Technical Report No. 48, May 15, 1963, Stanford University, California.

(19) G. A. Campbell and R. M. Foster, Fourier Integrals for Practical Applications, D. Van Nostrand, 1948, p. 113 pair 872.1.

(20) J. F. Kaiser, "A family of window functions having nearly ideal properties", November 1964, unpublished memorandum.

(21) D. Slepian and H. O. Pollak, "Prolate spheroidal wave functions, Fourier analysis and uncertainty - I and II", B.S.T.J. Vol. 40, No. 1, January 1961, pp. 43-84.

(22) P. E. Fleischer, "Digital realization of complex transfer functions", Simulation, Vol. 6, No. 3, March 1966, pp. 171-180.

(23) M. A. Martin, Digital Filters for Data Processing, General Electric Co., Missile and Space Division, Tech. Info. Series Report No. 62-SD484, 1962.

(24) S. A. Schelkunoff, "A mathematical theory of linear arrays", B.S.T.J. Vol. 22, January 1943, pp. 80-107. Mark Tso-Han Ma, A New Mathematical Approach for Linear Array Analysis and Synthesis, Ph.D. thesis, Syracuse University, 1961, University Microfilms No. 62-3050.

(25) G. M. Jenkins, "A survey of spectral analysis", Applied Statistics, Vol. XIV, No. 1, 1965, pp. 2-32.

(26) H. H. Robertson, "Approximate design of digital filters," Technometrics, Vol. 7, No. 3, August 1965, pp. 387-403.

(27) W. K. Linvill, "Sampled-data control systems studied through comparison of sampling with amplitude modulation, "Trans. AIEE, Vol. 70, Part II, 1951, pp. 1779-1788.

(28) A. Susskind, Notes on Analog-Digital Conversion Techniques, John Wiley, New York, 1957. See especially Chapter II, Sampling and Quantization by D. Ross.

(29) D. T. Ross, Improved Computational Techniques for Fourier Transformation, Servomechanisms Laboratory Report No. 7138-R-5, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 25, 1954.

(30) C. Jordan, Calculus of Finite Differences, Chelsea, 1960, (Reprint of 1939 Edition).

(31) H. M. James, N. B. Nichols, R. S. Phillips, Theory of Servomechanisms, McGraw-Hill, New York, 1947, pp. 231-261.

(32) J. R. Ragazzini and G. F. Franklin, Sampled-Data Control Systems, McGraw Hill, 1958.

(33) E. I. Jury, Sampled-Data Control Systems, John Wiley and Sons, Inc. 1958.

(34) J. T. Tou, Digital and Sampled-Data Control Systems, McGraw-Hill, 1959.

(35) E. Mishkin and L. Braun, Jr., Adaptive Control Systems, McGraw Hill, New York, 1961, pp. 119-183.

(36) E. I. Jury, Theory and Application of the z-Transform Method, John Wiley, New York, 1964.

(37) H. Freeman, Discrete-Time Systems, John Wiley, 1965.

(38) P. M. DeRusso, R. J. Roy, C. M. Close, State Variables for Engineers, John Wiley, 1965, pp. 158-186.

(39) R. B. Blackman, Linear Data-Smoothing and Prediction in Theory and Practice, Addison-Wesley, Reading, Massachusetts, 1965.

(40) C. Lanczos, Applied Analysis, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1956.

(41) R. B. Blackman, J. W. Tukey, The Measurement of Power Spectra from the Point of View of Communication Engineering, Dover, 1959.

(42) M. A. Martin, "Frequency Domain Applications to date Processing," IRE Transactions on Space Electronics and Telemetry, Vol SET-5, No. 1, March 1959, pp. 33-41.

(43) J. F. A. Ormsby, "Design of Numerical Filters with Applications to Missile Data Processing," Jour. ACM, Vol. 8, No. 3, July 1961, pp. 440-466.

(44) A. J. Monroe, Digital Processes for Sampled Data Systems, John Wiley and Sons, Inc., New York, 1962.

(45) R. M. Golden, "Digital Computer Simulation of Communication Systems Using the Block Diagram Compiler; BLODIB," Third Annual Allerton Conference on Circuit and System Theory, Monticello, Illinois, October 1965, pp. 690-707.

(46) A. Tustin, "A Method of Analyzing the Behavior of Linear Systems In Terms of Time Series," Jour. IEE, Vol. 94, Part II A, May 1947, pp. 130-142.

(47) J. M. Salzer, "Frequency Analysis of Digital Computers Operating In Real Time," Proc, IRE, Vol. 42, February 1954, pp. 457-466.

(48) R. Boxer, S. Thaler, "A Simplified Method of Solving Linear and Non-Linear Systems," Proc. IRE, Vol. 44, January 1956, pp. 89-101.

(49) R. Boxer, "A Note on Numerical Transform Calculus," Proc. IRE, Vol. 45, No. 10, October 1957, pp. 1401-1406.

(50) D. C. Baxter, The Digital Simulation of Transfer Functions, National Research Laboratories, Ottawa, Canada, DME Report No. MK-13, April 1964.

(51) C. J. Drane, Directivity and Beamwidth Approximations for Large Scanning Dolph-Chebyshev Arrays, AFCRL Physical Science Research Papers No. 117, AFCRL-65-472, June 1965.

(52) H. L. Garabedian (Ed.), Approximation of Functions, Elsevier Publishing Co., 1965, see especially Walsh pp. 1-16 and Cheney pp. 101-110.

(53) E. W. Cheney and H. L. Loeb, "Generalized rational approximation, "J. SIAM, Numerical Analysis, B, Vol. 1, 1964, pp. 11-25.

(54) Josef Stoer, "A direct method for Chebyshev approximation by rational functions," JACM, January 1964, Vol. 11, No. 1, pp. 59-69.

(55) R. M. Golden and J. F. Kaiser, "A computer program for the design of continuous and sampled-data filters", to be published.

(56) C. M. Rader and B. Gold, Digital Filter Design Techniques, Lincoln Laboratory Report, M.I.T., Preprint JA2612, September 1965.

(57) H. Holtz and C. T. Leondes, "The synthesis of recursive filters," Jour. ACM, Vol. 13, No. 2, April 1966, pp. 262-280.

(58) L. Weinberg, Network Analysis and Synthesis, McGraw Hill, 1962.

(59) D. A. Calahan, Modern Network Synthesis, Hayden, New York, 1964.

(60) J. E. Storer, Passive Network Synthesis, McGraw Hill, New York, 1957, pp. 287-302.

(61) P. Broome, "A frequency transformation for numerical filters," Proc. IEEE, Vol. 52, No. 2, February 1966, pp. 326-7.

(62) P. E. Mantey, Convergent Automatic-Synthesis Procedures for Sampled-Data Networks with Feedback, Stanford Electronics Laboratories, Technical Report No. 6773-1, SU-SEL-112, Stanford University, October 1964.

(63) J. R. B. Whittlesey, "A rapid method for digital filtering, Comm."ACM, Vol. 7, No. 9, September 1964, pp. 552-556.

(64) T. Y. Young, Representation and Analysis of Signals, Part X. Signal Theory and Electrocardiography, Department of Electrical Engineering, Johns Hopkins University, May 1962.

(65) P. W. Broome, "Discrete orthonormal sequences,"Jour. ACM, Vol. 12, No. 2, April 1965, pp. 151-168. Archambeau et al., "Data processing techniques for the detection and interpretation of teleseismic signals,"Proc. IEEE, Vol 53, No. 12, December 1965, pp. 1860-1994, see especially p. 1878.

(66) H. W. Bode, Network Analysis and Feedback Amplifier Design, Van Nostrand, 1945, pp. 47-49.

(67) M. Mansour, "Instability criteria of linear discrete systems, "Automatica, Vol. 2, No. 3, January 1965. pp. 167-178.

(68) C. E. Maley, "The effect of parameters on the roots of an equation system,"Computer Journal, Vol. 4, 1961-2. pp. 62-63.

(69) J. G. Truxal, Automatic Feedback Control System Synthesis, McGraw Hill Book Co., Inc., New York, 1955, pp. 223-250.

(70) F. F. Kuo, Network Analysis and Synthesis, John Wiley, New York, First Edition, 1962, pp. 136-137, (Second Edition, pp. 148-155).

(71) C. Pottle, "On the partial-fraction expansion of a rational function with multiple poles by a digital computer,"IEEE Trans. Circuit Theory, Vol. CT-11, March 1964, pp. 161-162.

(72) W. R. Bennett, "Spectra of quantized signals", B. S. T. J. Vol. 27, July 1948, pp. 446-472.

(73) B. Widrow, "A study of rough amplitude quantization by means of Nyquist sampling theory,"Trans, IRE on Circuit Theory, Vol. CT-3, No. 4, December 1956, pp. 266-276.

(74) B. Widrow, "Statistical analysis of amplitude quantized sampled-data systems,"Trans. AIEE Applications and Industry, No. 52, January 1961, pp. 555-568.

(75) F. E. Hills, A Study of Incremental Computation by Difference Equations, Servomechanisms Laboratory Report No. 7849-R-1, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1958.

(76) J. B. Knowles and R. Edwards,"Effect of a finite-word-length computer in a sampled-data feedback system,"Proc. IEE Vol. 112, No. 6, June 1965, pp. 1197-1207.

(77) J. B. Knowles and R. Edwards,"Simplified analysis of computational errors in a feedback system incorporating a digital computer,"S. I. T. Symposium on Direct Digital Control, April 22, 1965, London.

(78) J. B. Knowles and R. Edwards, "Complex cascade programming and associated computational errors,"Electronics Letters, Vol. 1, No. 6, August 1965, pp. 160-161.

(79) J. B. Knowles and R. Edwards, "Finite word-length effects in multirate direct digital-control systems,"Proc. IEE, Vol. 112, No. 12, December 1965, pp. 2376-2384.

(80) B. Gold and C. Rader,"Effects of quantization noise in digital filters,"Proceedings Spring Joint Computer Conference, 1966. Vol. 28, pp. 213-219.

(81) J. H. Wilkinson, Rounding Errors in Algebraic Processes, Prentice Hall, Englewood Cliffs, New Jersey, 1963.

(82) J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series,"Mathematics of Computation, Vol. 19, No. 90, April 1965, pp. 297-301.

(83) R. A. Gaskill, "A versatile problem-oriented language for engineers, "IEEE Transactions on Electronic Computers, Vol. EC-13, No. 4 (August, 1964), pp. 415-421.

(84) R. D. Brennan and R. N. Linebarger, "A survey of digital simulation: digital analog simulator programs,"Simulation Vol. 3, No. 6 (December, 1964), pp. 22-36.

(85) J. J. Clancy and M. S. Dineberg, "Digital simulation languages: a critique and a guide," AFIPS Conference Proceedings, Vol. 27, Part I (1965), Spartan Books, Washingron, D.C., pp. 23-36.

(86) J. C. Strauss and W. L. Gilbert, SCADS: a programming system for the simulation of combined analog digital systems, Carnegie Institute of Technology, March, 1964.

(87) W. M. Syn and D. G. Wyman, DSL/90 Digital Simulation Language User's Guide, IBM Corporation, San Jose, California, July, 1965.

(88) R. W. Burt and A. P. Sage, "Optimum design and error analysis of digital integrators for discrete system simulation," AFIPS Conference Proceedings, Vol. 27, Part I (1965), Spartan Books, Washington, C.D., pp. 903-914.

(89) M. E. Fowler, "A new numerical method for simulation," Simulation, Vol. 4, No. 5 (May, 1965), pp. 324-330.

(90) C. C. Cutler, "Transmission Systems Employing Quantization," U.S. Patent No. 2, 927, 962, March 8, 1960 (filed April 26, 1954).

(91) T. G. Stockham, Jr., "High speed convolution and correlation," Proceedings Spring Joint Computer Conference, 1966, Vol. 28, pp. 229-233.

(92) H. D. Helms, "Fast Fourier transforms methods of computing difference equations arising from z-transforms and autoregressions," (to appear).

(93) J. L. Kelly, Jr., C. Lochbaum, and V. A. Vyssotsky, "A block diagram compiler", B.S.T.J., Vol. 40, No. 3 (May, 1961) pp. 669-676.

(94) M. R. Schroeder et al., New methods for speech analysis-synthesis and bandwidth compression. Congress Report of the Fourth International Congress on Acoustics, Copenhagen, 1962.

(95) L. S. Frishkopf and L. D. Harmon, "Machine recognition of cursive script," Symposium on Information Theory, London (1960), C. Cherry, Editor, Butterworth and Co. Ltd, London (1961), pp. 300-316.

(96) B. J. Karafin, "The new block diagram compiler for simulation of sampled-data systems," AFIPS Conference Proceedings, Vol. 27, Part I (1965), Spartan Books, Washington, D. C. pp. 53-62.

(97) A study of Math Model Input/Output Parameters for the Computer Aided Analysis Program HASD 6420-821429, Lockheed Electronics Company.

(98) G. Franklin, "Linear Filtering of Sampled Data", IRE Conv. Rec., Vol. 3, Pt IV, pp 119-128, 1955.

(99) R. P. Brennan and R. N. Linebarger, "An Evaluation of Digital Analog Simulator Languages", I. F. I. P. 1965 Proceedings, Vol. 2.

(100) J. R. Hurley and J. J. Skiles, "DYSAC", 1963 SJCC, Vol. 23, Spartan Books, Washington, D.C.

(101) V. C. Rideout and L. Tavernini, "MAD BLOC", Simulation, Vol. 4, No. 1, January 1965.

(102) M. L. Stein, J. Rose and D. B. Parker, "A Compiler with An Analog Oriented Input Language (ASTRAC)", Proc. 1959 WJCC.

(103) F. J. Sansom and H. E. Peterson, "MIMIC - Digital Simulator Program", SESCA Internal Memo 65-12, WPAFB, May 1965.

(104) R. T. Harnett and F. T. Sansom, "MIDAS Programming Guide", Report No. 5EG-TDR-64-1, WPAFB, Ohio, January 1964.

(105) M. Palevsky and J. V. Howell, "DES-1", Fall JCC, Vol. 24, Spartan Books, Inc., Washington, D.C., 1963.

(106) R. D. Brennan and H. Sano, "PACROLUS", Fall JCC, Vol. 26, Spartan Books, Inc., Washington, D.C., 1964.

(107) R. N. Linebarger, "DSL/90", Paper at Joint Meeting Midwestern and Central States Simulation Councils, May 1965.

(108) R. A. Gaskill, J. W. Harris, and A. L. McKnight, "DAS-A Digital Analog Simulator", Proc. 1963 Spring JCC, AFIPS Conference Proc., Vol. 23, p. 83.

(109) G. E. Blechman, "An Enlarged Version of MIDAS", S&I Div. NAR, June 1964; Simulation, Vol. 3, No. 4, October 1964.

(110) M. E. Fowler, "A New Numerical Method for Simulation", Simulation, pp. 324-330, May 1965.

(111) M. E. Fowler, "An Example Showing Use of Root Locus Techniques to Study Nonlinear Systems", TR, August 12, 1964, IBM R&D Ctr., Palo Alto, California.

(112) SAI Proposal No. 69-045, dated October 1969, "Time Domain Simulation of Apollo Telecommunications Lines".

(113) W. E. Thompson, "Network with Maximally Flat Delay," Wireless Engineer, Vol. 29, pg 255, October 1952.

(114) "On the Design of Filters by Synthesis," R. Saul and E. Ulbrich, IRE Transactions on Circuit Theory, December 1958.

(115) "The Design of Filters Using the Catalog of Normalized Low-Pass Filters," R. Saal, Telefunken, 1963.

(116) "Passive Network Synthesis," James E. Storer, McGraw-Hill, 1957.

APPENDIX A

THEORETICAL BASIS FOR SYSTID

A.1 THEORETICAL INTRODUCTION

Direct representation of systems on the digital computer by sample data simulation is a powerful systems analysis technique. Such simulation requires transformation by the computer of continuous system input functions in a manner which characterizes system behavior. The digital computation/process by which this transformation is accomplished is known as a digital filter. This is an algorithm by which sample values of a continuous input function are transformed into sample values of the continuous output function which would result from operating on the input with a given continuous transfer characteristic. The central problem in sample data simulation is obtaining the digital filter algorithm which effects this transformation in the most accurate and efficient manner.

Digital filters may be classified into two major categories as recursive or non-recursive. Non-recursive digital filter outputs depend only on present and previous input samples; recursive filter outputs depend on previous output values as well. The design methods for these two filter types are distinctly different as are their properties. The non-recursive filter has finite memory and excellent phase response characteristics but may require a large number of terms to obtain sharp cutoff properties. The recursive filter has infinite memory but rather poor characteristics. Recursive filters have fewer terms and lend themselves more efficiently to applications requiring sharp cutoff properties. The recursive filter is the digital counterpart of the linear lumped parameter continuous filter. For these reasons, recursive filters are of greater interest in systems analysis by sample data simulation and will be summarized briefly.

If it is assumed that the linear system for which a digital approximation is sought has a transfer characteristic of the form

$$H(s) = \frac{\sum_{m=0}^{M} C_m s^m}{\sum_{n=0}^{N} d_n s^n} \qquad \text{(A-1)}$$

where $s = j\omega$, then the corresponding digital transfer characteristics has the form

$$H^*(Z) = \frac{\sum_{j=0}^{N-1} a_j z^{-j}}{1 + \sum_{j=1}^{N} b_j z^{-j}} \qquad \text{(A-2)}$$

where z^{-1} is the unit delay operator. It is assumed that the continuous function H(s) is known or can be determined by established design procedures. The digital filter design problem is thus reduced to determining the coefficients a_j and b_j in $H^*(z)$ such that the continuous filter characteristic H(s) is best approximated for a given number of terms.

One digital filter design technique is based on the standard z-transform, defined so that the impulse response of the digital filter is identical to the sampled impulse response of the corresponding continuous filter. The standard z-transform of H(s) is given by

$$H^*(s) = \sum_{m=-\infty}^{\infty} H(s + jm\,\omega s) \qquad \text{(A-3)}$$

or in terms of the filter impulse response h(t)

$$H^*(z) = T \sum_{\ell=0}^{\infty} h(\ell T) z^{-1} \qquad \text{(A-4)}$$

where

$s \quad = \sigma + j\omega$

$H(s) \quad =$ Laplace transform of h(t)

$\omega_s \quad = \frac{2\pi}{T}$, radian sampling frequency

$H^*(s) \quad =$ Laplace transform of sampled filter impulse response

$z^{-1} \quad = e^{-st}$, unit delay operator

$H(z) \quad = H^*(s)|_{s = \ln z/T}$, z transform of h(t)

For s greater than some critical frequency ω_c, H(s) is assumed to have the form

$$H(s)|_{s>j\omega_c} = K/s^n \tag{A-5}$$

where n>0 and K is a constant.

Equations (A-3) and (A-4) are the digital filter transfer functions which approximate that of the continuous filter.

The disagreement between the digital filter characteristics provided by the standard z-transform and the continuous filter characteristic in the baseband ($-\omega_s/2 \le \omega \le \omega_s/2$) is known as frequency aliasing error and results from terms of the form $H(s+jm\omega_s)$, $m\neq 0$. This disagreement is present whenever the continuous filter characteristic is not bandlimited to the baseband. Unfortunately this is the case for most lumped parameter systems, for which H(s) is a rational function of s. Thus, for physical systems of interest, the standard z-transform yields $H^*(s) \neq H(s)$ in the baseband and aliasing error is present to some degree.

For higher order continuous filter transfer functions (n in equation (5) is large) having a critical frequency ω_c much less than the sample frequency ω_s, aliasing error is sufficiently small that the standard z-transform yields useful results. In many practical situations, however, neither of these conditions are met. In these cases, the standard z-transform results in prohibitive aliasing errors in the digital filter frequency characteristic.

Frequency aliasing error may be avoided if digital filters are designed by means of an artifice known as the bilinear z-transform. The bilinear z-transform maps the entire complex s plane into an s_1 plane bounded by the lines $s_1 = j\omega_s/2$ and $s_1 = j\omega_s/2$. The bilinear z-transform is defined by

$$s = \frac{2}{T}\tanh\frac{s_1 T}{2} \tag{A-6}$$

where $s_1 = j\omega_s/2$ and T is the sample interval. This becomes upon substitution of the unit delay operator $z^{-1} = e^{-s_1 T}$,

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \tag{A-7}$$

The digital filter transfer function, $H^*(z)$ is determined by substituting the bilinear z-transform into the continuous filter transfer function $H(s)$,

$$H^*(z) = H(s)\Big|_{s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)} \qquad \text{(A-8)}$$

One aspect of the digital filter so obtained is that a non-linear warping is imparted to its frequency scale in accord with the transformation

$$\frac{\omega T}{2} = \tan \frac{\omega_1 T}{2} \qquad \text{(A-9)}$$

This transformation is depicted in Figure A-1 which plots normalized warped frequency ω, vs. normalized unwarped frequency ω. Frequency warping is not a significant constraint on the versatility of the biliner z-transform. The warping may be arbitrarily reduced by making the sample frequency ω_s high compared to the critical frequency ω_c of the continuous filter. Furthermore, frequency warping may be compensated for by prewarping the critical frequencies of the continuous filter so that transformed frequencies will be shifted back to the desired ones.

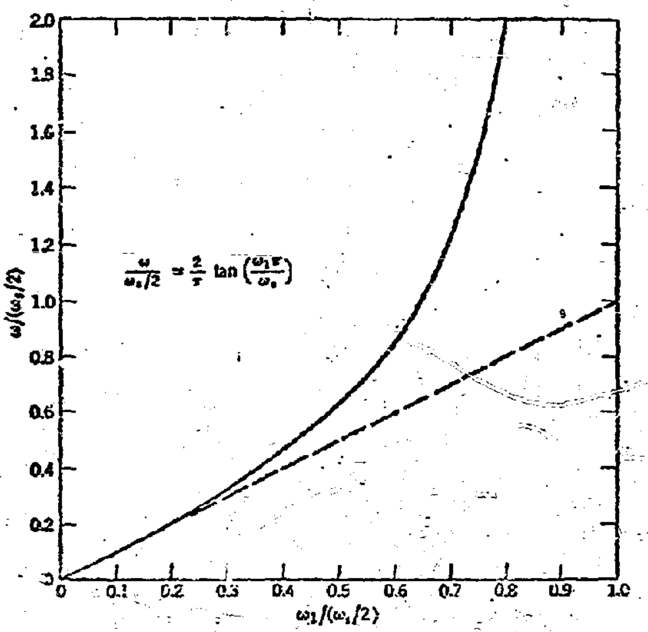


Figure A-1. Nonlinear Warping of the Frequency Scale in the Bilinear z-transformation

Because it obviates inaccuracies due to aliasing error, the bilinear z-transform is a most appealing digital filter design technique. It is applicable to low-pass band-pass, band-stop, and other continuous filters whose magnitude characteristics are essentially constant within successive pass and stop bands.

A.2 APPLICATIONS OF SAMPLED DATA TECHNIQUES IN THE SYSTID PROGRAM

A great diversity of systems are amenable to analysis by sample data simulation. Telemetry links may be modeled with any combination of amplitude or angle modulation, phase-locked loops may be simulated either separately or as part of a more complex system, transient response of filters to various input waveforms can be accurately determined, or recorded data may be processed by digital filtering.

The existing extensive SAI library of sample data simulation programs[1] is designed for maximum flexibility. As an example of sample data simulation, consider the following analysis of a radio frequency communications link.

A sample data simulation of an RF link can be implemented by a sampled carrier and any of several modulation and demodulation schemes. In this case, however, the sample frequency ω_s must be high enough that frequency warping due to the bilinear z-transform remains within acceptable limits. Since the carrier frequency is typically many orders of magnitude higher than the baseband frequency, a correspondingly higher sample rate is indicated by Figure 1, resulting in excessive computer runtime.

A technique has been devised whereby carrier frequency can be eliminated from the system representation. This permits greatly reduced sample rate and correspondingly shorter computer runtime without degrading the carrier frequency bandpass filter characteristics.

Consider the following system:

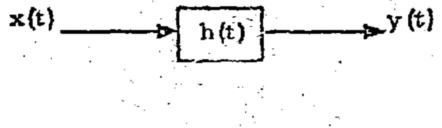


Figure A-1.

[1] The SAMDAT program, for instance.

where

$$x(t) = A(t)e^{j[\omega_c t + \varphi(t)]} + n(t)$$

$$y(t) = A_0(t)e^{j[\omega_c t + \varphi_0(t)]}$$

$n(t)$ = additive noise

$h(t)$ = impulse response of the filter

The input is a general modulated signal with additive noise. The output is a complex signal represented by an amplitude and a phase.

The actual signal inputs and outputs are the real parts of those given. In accordance with Reference [ ], for amplitudes and phases to be determined by the magnitudes and phases of the complex signals, all inputs must be analytic signals, i.e., the imaginary parts must be the Hilbert transform of the real parts. This condition is satisfied on the signal term in $x(t)$ if the frequency spectrum of $A(t)e^{j\phi(t)}$ is essentially zero at the frequence ω_c; however, the noise term $n(t)$ must also be an analytic signal.

Assume $Re[n(t)]$, the actual noise term, is Gaussian with a frequency spectrum which is symmetric about ω_c (This does not imply that the bandpass filter has symmetric response about ω_c). The real part of the noise term can be written as

$$Re[n(t)] = n_1(t)\cos\omega_c t - n_2(t)\sin\omega_c t, \qquad \text{(A-10)}$$

The two quantities $n_1(t)$ and $n_2(t)$ will be independent and Gaussian, will have identical frequency spectra equal to the original spectrum translated to dc, and will each have variance equal to that of $Re(n(t))$. The imaginary part of $n(t)$ must be the Hilbert transform of this quantity. As long as the frequency spectrum of $n_1(t)$ or $n_2(t)$ is essentially zero at ω_c, $n(t)$ is given by

$$n(t) = [n_1(t) + jn_2(t)]e^{j\omega_c t}. \qquad \text{(A-11)}$$

The system now is represented by the following convolution equation.

$$A_0(t)e^{j\varphi_0(t)}e^{j\omega_c t} = \int_0^\infty h(\mu)\Big[A(t-\mu)e^{j\varphi(t-\mu)} + n_1(t-\mu) + jn_2(t-\mu)\Big]e^{j\omega_c(t-\mu)}d\mu \qquad \text{(A-12)}$$

After rearranging factors and removing the carrier term from the integral, equation (3) becomes

$$A_0(t)e^{j\varphi_0(t)} = \int_0^\infty k(\mu)\Big[A(t-\mu)e^{j\varphi(t-\mu)} + n_1(t-\mu) + jn_2(t-\mu)\ d\mu\Big] \qquad \text{(A-13)}$$

where

$$k(t) \equiv h(t)e^{-j\omega_c t}$$

Equation (A-13) implies that our system can be simulated by the following one having the same output and input except that the carrier frequency has been eliminated

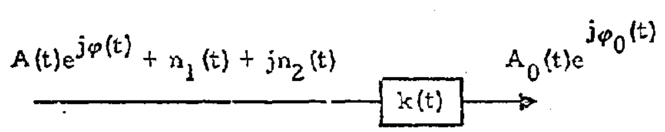


Figure A-2.

In order to simulate the new system it will be convenient to determine $K_r(s)$ and $K_i(s)$ where these are the Laplace transforms of $k_r(t)$ and $k_i(t)$, the real and imaginary parts of $k(t)$. Also, because the subroutine used to set up the simulation first removes the center frequency phase from $H(s)$, it will be convenient to define

$$k(t) = h(t)e^{-j(\omega_c t + \psi)} \tag{A-14}$$

where ψ is a constant phase.

The two systems will now be the same except for the constant phase ψ.

It may be seen that

$$k_r(t) = h(t)\cos(\omega_c t + \psi)$$

$$k_i(t) = -h(t)\sin(\omega_c t + \psi). \tag{A-15}$$

Taking Laplace transforms and applying Euler's formulas to the sin and cos yields:

$$K_r(s) = \frac{1}{2}\int_0^\infty h(t)\left[e^{j(\omega_c t+\psi)} + e^{-j(\omega_c t+\psi)}\right]e^{-st}\,dt$$

$$K_i(s) = \frac{-1}{2j}\int_0^\infty h(t)\left[e^{j(\omega_c t+\psi)} - e^{-j(\omega_c t+\psi)}\right]e^{-st}\,dt \tag{A-16}$$

Equation (A-15) becomes

$$K_r(s) = \frac{1}{2}\left[e^{-j\psi}H(s + j\omega_c) + e^{j\psi}H(s - j\omega_c)\right]$$

$$K_i(s) = \frac{1}{2j}\left[e^{-j\psi}H(s + j\omega_c) - e^{j\psi}H(s - j\omega_c)\right] \tag{A-17}$$

H(s) is a rational fraction in s. If the real or imaginary parts of $e^{-j\psi}$ $H(s + j\omega_c)$ are taken with respect to the coefficients of s, not s itself, equation (A-17) can be rewritten as

$$K_r(s) = \text{Re}\left[e^{-j\psi}H(s + j\omega_c)\right]$$

$$K_i(s) = \text{Im}\left[e^{-j\psi}H(s + j\omega_c)\right] \tag{A-18}$$

or the entire function

$$K(s) = e^{-j\psi}H(s + j\omega_c)$$

If H(s) represents a bandpass filter, K(s) will have bandpass regions about the origin and about the frequency $-2\omega_c$. The response at $-2\omega_c$ is not important since the input signals should not have spectral components there. As indicated in the discussion of digital filter designs, aliasing error is eliminated by the bilinear z-transform. Thus, the sample frequency can be selected in accordance with the bandwidth of the filter, not its center frequency.

If $A(t)\, e^{j\varphi(t)}$ has frequency components greater than one-half the sample frequency, aliasing error in the signal will occur although the filter will be represented correctly. Also if $A(t)\, e^{j\varphi(t)}$ has a nonzero frequency spectrum at $\omega_1 = 2/T \tan \omega_c T/2$, which corresponds to the center frequency shifted by the bilinear z-transform, a ripple in the output amplitude and phase at frequencies about $2\omega_1$ may occur. This is because the assumption that $A(t)e^{j\varphi(t)}\, e^{j\omega_c t}$ is an analytic will not be true. Some ripple for instance may be detected if A(t) is a step which has a frequency response which rolls off slowly at higher frequencies.

A sample data demodulation can now be implemented quite simply, following the bandpass filter. Denote the output of the second system as

$$A_0(t)e^{j\varphi_c(t)} = a(t) + jb(t). \tag{A-19}$$

Then

$$A_o^2(t) = a^2(t) + b^2(t) \tag{A-20}$$

and the output phase is given by

$$\varphi_o(t) = \tan^{-1} \frac{b(t)}{a(t)} \tag{A-21}$$

In an FM system, $\mu_o(t)$, the instantaneous frequency, is of interest, rather than the phase. Whenever $A_o(t)$ does not pass through zero, $\mu_o(t)$ is given by

$$\mu_o(t) = \frac{d\varphi_o(t)}{dt} = \frac{a(t)\frac{db(t)}{dt} - b(t)\frac{da(t)}{dt}}{A_o^2(t)} \tag{A-22}$$

$A_o(t)$ by definition is always positive. The sign information which would normally be associated with $A_o(t)$ is implied by the phase. This means that a sign reversal in $A_o(t)$ would be manifested by $A_o(t)$ going to zero between two successive sample points and the phase undergoing a step change of π radians. This sign reversal can be detected by sensing when both $a(t)$ and $b(t)$ change signs from one sample point to the next. The step change in phase can be implemented by causing $\mu_o(t)$ to contain an impulse function of magnitude $\pm\pi$, which in a sample data system is simply one point of magnitude $\pm\pi/T$. The sign of the impulse is selected alternately as plus and minus. This is correct because if $A_o(t)$ had passed from positive to negative at some point in time it must next pass from negative to positive.

Equation (A-11) suggests that Gaussian noise may be inserted, less the carrier component, at the input to the RF frequency filter in the following form.

$$n(t) = n_1(t) + jn_2(t) \tag{A-23}$$

The terms $n_1(t)$ and $n_2(t)$ were assumed to have identical power spectra and were assumed to be statistically independent. They were also assumed to have a power spectrum which is essentially zero at the carrier frequency ω_c.

The last assumption requires that n(t) be bandlimited by a filter prior to its insertion into the link. The possibility of relaxing this requirement for certain simulations avoids the use of an extra filter.

Let $n_1(t)$ and $n_2(t)$ be generated as pseudo-random numbers, independent, and with Gaussian statistics by a computer random number subroutine. Let these numbers be of zero mean and standard deviation σ. Because they are independent they can be assumed to be samples of a noise process whose spectrum is uniformly distributed between $-f_s/2$ to $f_s/2$ so that the two sided power spectral density η is given by

$$\eta = \frac{\sigma^2}{f_s} \qquad \text{(A-24)}$$

where f_s = sample frequency. Now assume this signal is passed through a sample data bandlimiting filter so that its power spectrum in the interval $-\omega_s/2 \le \omega \le \omega_s/2$ becomes

$$S_1(\omega) = \frac{\sigma^2}{f_s} \left| H_1\left(j\frac{2}{T} \tan \frac{\omega T}{2}\right)\right|^2 \qquad \text{(A-25)}$$

The spectrum $S_1(\omega)$ is now passed through the bandpass filter transfer function $H(j\omega)$ translated to zero and distorted in frequency response to produce the output spectrum

$$S(\omega) = \frac{\sigma^2}{f_s} \left| H_1\left(j\frac{2}{T} \tan \frac{\omega T}{2}\right) H\left(j\frac{2}{T} \tan \frac{\omega T}{2} + j\omega_c\right)\right|^2 \qquad \text{(A-26)}$$

Without bandlimiting by $H_1(j\omega)$, $S(\omega)$ will have the same shape as $|H(j\,2/T \tan \omega T/2 + j\omega_c)|^2$, i.e., it will have the desired bandpass shape about dc and in addition will have a more narrow bandpass shape about the frequency $-\omega_{c2} = -2/T \tan^{-1} \omega_c T$. The analytic signal assumption, because of the frequency warping, is equivalent to setting $S(\omega) = 0$ for $\omega < -\omega_{c1} = -2/T \tan^{-1} \omega_c T/2$ so that any noise power below $-\omega_{c1}$ represents an error in the simulation.

Let N denote the total noise power above $-\omega_{c1}$ and N_e denote that below. N will then be the noise power which should normally be used in computing carrier to noise ratio. N_e, the extra noise added by the simulation, will normally affect the outputs at frequencies which are not of interest and may later be filtered out. However, it is important to realize that N_e will affect threshold in the system and must be kept small whenever the simulation is to function near or below threshold.

The quantities N and N_e can be derived by integrating equation (A-26) over the proper regions to obtain

$$N = \sigma^2 \frac{f_b}{f_s}$$

$$N_e = \sigma^2 \frac{f_{be}}{f_s} \qquad \text{(A-27)}$$

where

$$f_b = \frac{1}{2\pi} \int_{-\frac{2}{T}\tan^{-1}\frac{\omega_c T}{2}}^{\omega_s/2} \left| H_1\left(j\frac{2}{T}\tan\frac{\omega T}{2}\right) H\left(j\frac{2}{T}\tan\frac{\omega T}{2} + j\omega_c\right)\right|^2 d\omega \qquad \text{(A-28)}$$

$$f_{be} = \frac{1}{2\pi} \int_{-\omega_s/2}^{-\frac{2}{T}\tan^{-1}\frac{\omega_c T}{2}} \left| H_1\left(j\frac{2}{T}\tan\frac{\omega T}{2}\right) H\left(j\frac{2}{T}\tan\frac{\omega T}{2} + j\omega_c\right)\right|^2 d\omega \qquad \text{(A-29)}$$

Equations (A-28) and (A-29) can be modified by a change in variables to the following equations.

$$f_b = \frac{1}{2\pi} \int_0^{\infty} \left| H_1(j\omega - j\omega_c) H(j\omega) \right|^2 \frac{d\omega}{1 + \left(\frac{(\omega - \omega_c)T}{2}\right)^2} \qquad \text{(A-30)}$$

$$f_{be} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left| H_1(j\omega - j\omega_c)\, H(j\omega) \right|^2 \frac{d\omega}{1 + \left(\frac{(\omega - \omega_c)T}{2} \right)^2} \qquad \text{(A-31)}$$

From Equation (A-30) it can be seen that if $H_1(j\omega) \approx 1$ over a range equal to the passband of $H(j\omega)$ and if $\omega - \omega_c \ll \omega_s/2$, f_b is the equivalent noise bandwidth of the modulation link filter. An estimate of the ratio of N_e to N can be obtained by assuming $H(j\omega)$ is narrowband. This estimate is given below.

$$\frac{N_e}{N} \approx \left| \frac{H_1(j2\omega_c)}{H_1(j0)} \right|^2 \frac{1}{1 + \left(2\pi \frac{\omega_c}{\omega_s} \right)^2} \qquad \text{(A-32)}$$

If the system requires that the ratio of N_e/N be small this may be achieved if the carrier frequency is much higher than the sample frequency, or it can be assured by bandlimiting the noise by a lowpass filter $H_1(j\omega)$.

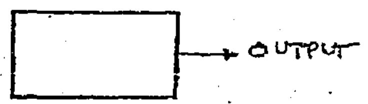
APPENDIX B

SYSTID MODEL DESCRIPTIONS AND THEORY

B.1.0 MODEL DESCRIPTION

There are three basic types of models used in the SYSTID system:

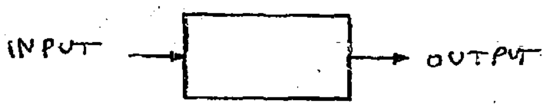
(a) Mononode devices,



Example: noise generator

Figure B-1. Mononode Device

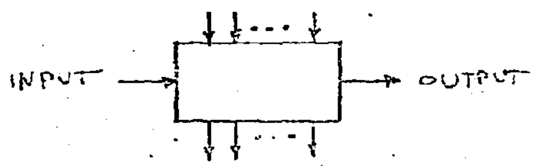
(b) Binode devices, and



Example: limiter

Figure B-2. Binode Device

(c) Multinode devices



Example: multiplier (3 nodes)

Figure B-3. Multinode Device

Devices of type (a) will be referred to as SYSTID library functions. They cannot be directly constructed in this form in the SYSTID language. Most models will be of type (b). They are the simplest to construct and use and result in the most efficient simulation program. Many models of type (c) can be constructed as models of type (b) by an appropriate choice of the model boundaries. For example incorporating two models of type (c) having a common tap into a single model of type (b) is such a construction.

The difference between a model and a system is not particularly significant from the user's point of view. A system is the minimum configuration necessary to run a meaningful simulation on the computer. The user will find it easy to construct a model of a link such as the link from A-to-B, simulate it and later incorporate the A-to-B link into an A-to-C link without altering the A-to-B model. The distinction between a model and a system is a matter of fixing all system parameters and input and output specifications. The philosophy of the program is that IO statements should not be embedded into a model, as they will require alteration of the model as it is used in various systems for which different input and output is desired.

In the SYSTID system the following convention is used: The signal, as a function of time, progresses from the left to the right, left node to right node. The left node will always be an input node and the right node will always be an output node. Taps, depending upon context will be either inputs or outputs.

Models of type (c) are not easily handled on a digital computer since the machine effectively executes only one instruction at a time. The input and output nodes are handled automatically by the processor. The output of device is passed to the devices down-link through their common node. Taps, on the other hand require more information to be properly connected, however. A detailed discussion of the use of taps will be deferred to a later section. It should be clear that the distinction between a tap and the input and output nodes is a functional requirement of the SYSTID processor and not a property of the device being modeled.

B.2.0 MODEL NODE DEFINITIONS

The topology of a model is defined by its nodes. Every user-constructed model must have an input node and an output node. The signal enters the model at the input node and progresses through the model along the paths defined by the nodes and the devices connecting nodes. At least one such path must reach the output node and all paths must terminate at the output node or at a tap. At each node the signal enters the input node of all devices common to the node. The signal may also leave a device through a tap or enter a device through a tap, but the connection node to tap is never explicit. A node is a collection point for the signal. The value of the signal at a node is the instantaneous sum of the output of all devices immediately up-link of the node; this value is passed to the input nodes of all devices immediately down-link of the node. Consider Figure B-4, if the device DO NOTHING is a short circuit, output equals input, the value of the signal at node 2 is twice the value of the signal at node 1. Normally, the summing effect of a node will not lead to difficulties, however, the user should be aware of its effect on his model, particularly if he is using identical parallel branches.

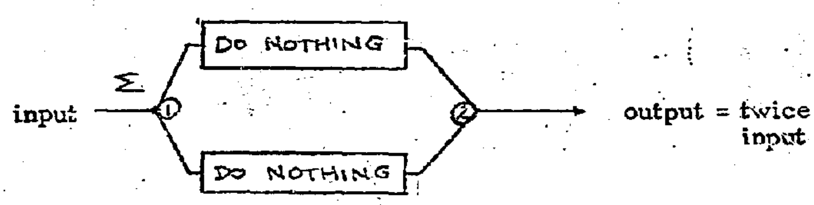


Figure B-4. Parallel Devices

B.3.0 LISTING OF INTERNAL VARIABLES AND SYSTID LIBRARY MODELS

B.3.1 SIGNAL GENERATORS

The set of internal generators comprises all functions provided by FORTRAN and those written into the SYSTID Library. These elements are functions and may be utilized in expressions. NOTE that all output peak levels are unity, unless otherwise noted.

B.3.1.1 Transcendental Functions

SIN (x)		SINE (y)	
COS (x)	x in radians	COSINE (y)	y in degrees
TAN (x)		TANGNT (y)	

B.3.1.2 Square Wave

SQ(P) where p = frequency or rate

B.3.1.3 Pulse Generator

PULSE (RATE, TD, TR, TL, TF)

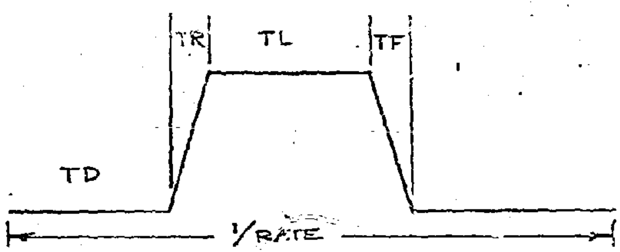


Figure B-5.

B.3.1.4 Arbitrary Function or Non-Linearity

TABLE (XIN, X1, Y1, X2, Y2, X3, Y3, X4, Y4, X5, Y5)

This function provides a piece-wise linear function for modeling both driving functions and non-linearities.

where

XIN = independent variable

X1, Y2
⋮ } five point pairs describing the function
X5, Y5

B.3.1.5 Periodic Function Generator

PTABLE (T1, Y1, T2, Y2, T3, Y3, T4, Y4, T5, Y5)

This model provides the periodic function capability. The output is periodic with period T5.

B.3.1.6 Gaussian Noise Generator

This function provides noise modeling capability and has two uses: One by providing the spectral density desired, the other providing the SNR and ENB of the generator.

GNØISE (SNR, ENB, ISTART)

GNØIS2 (ETA, ISTART)

where

SNR is the signal-to-noise ratio desired in ENB (equivalent noise bandwidth) assuming a unity signal level

ISTART is a positive integer (>0) for initializing the random number generator

ETA is the desired spectral density

White Gaussian:

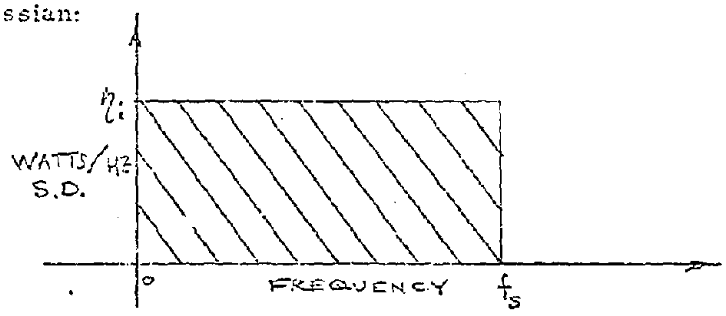


Figure B-6.

where

$$f_s = \frac{1}{DT} = \frac{1}{\text{sampling rate}}$$

$$\sigma_i^2 = \frac{\eta_i f_s}{2} = \frac{\eta_i}{2DT}$$

or

$$\boxed{\eta_i = 2\sigma_i^2 DT} \qquad \text{(B-1)}$$

$$N_o = \eta_i * ENB = 2\sigma^2 DT * ENB$$

where ENB = equivalent noise bandwidth under consideration.

For a given SNR in bandwidth, BW:

$$\frac{S}{N_o} = 10^{SNR/10}$$

where

S = signal power in BW

or

$$N_o = S * 10^{-SNR/10} = 2\sigma_i^2 DT * ENB$$

or

$$\sigma_i = \sqrt{S} / \sqrt{10^{SNR/10} * 2DT * ENB} \quad \text{(B-2)}$$

B.3.2 MODULATORS

This section provides the definition of the modulators available to the SYSTID user. As described in section 2.0 of the main text, the capability for efficiently simulating RF communications systems relies on the ability to model such systems at baseband. This is accomplished by translating the RF frequency components to the baseband region with a parameter normally set to the highest carrier frequency.

It is the responsibility of the user to consistently define this translation to all RF components. To illustrate, the following example is posed:

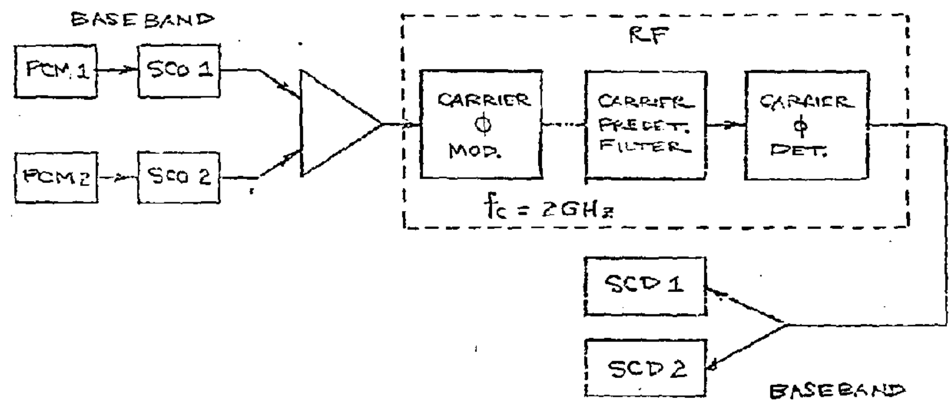


Figure B-7.

The RF components can be simulated in SYSTID by translating them (automatically) by f_c (2GHz), thereby allowing the simulation to be performed in the kHz region. The translation, however, must be consistent and therefore requires two modulator models for each type of modulation: baseband and RF. The RF modulators are prefixed with an "R".

The interface when transcending the baseband to RF domain is a model named SPLIT, which simply convert the signal into its real and imaginary components for use internally.

B.3.2.1 Amplitude Modulator - (AMMOD and RAMMOD)

(a) Functional Description - The linear Amplitude Modulator (AMMOD) model provides classical modulation capability to the SYSTID user. This model is written in the SYSTID language. The two forms of the modulator are functionally identical; the difference lies in their use (section B.3.2).

(b) Parameters - Two parameters are required:

BETA = Modulation Index (ratio)

FC = Carrier Frequency

(c) Detailed Description - The AM modulation process is described as follows:

$$e_o(t) = \left(1.0 + BETA \cdot INPUT\ (t)\right) \cdot \cos\ (2\pi\, FC\ Time)$$

where INPUT (t) = modulating time function (the model input)

NOTE: |BETA * INPUT (t)| ≤ 1 for no over-modulation. For RAMMOD, FC = 0 in the above and the output consists of the complex baseband signal.

(d) Block Diagram

AMMOD

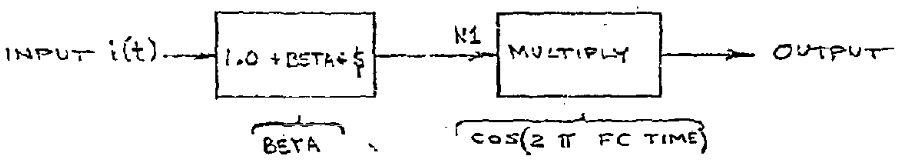


Figure B-8.

RAMMOD

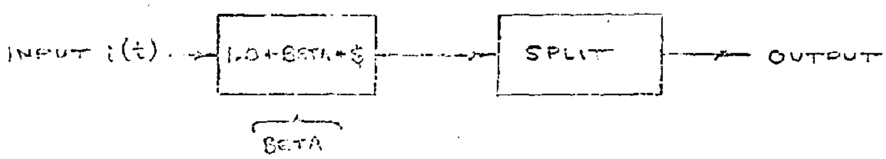


Figure B-9.

(e) Listing

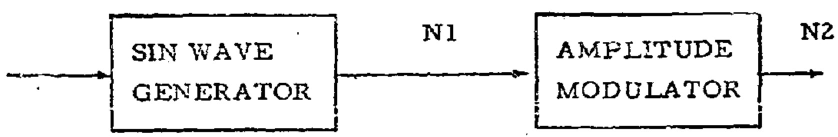
AMMOD

```
MODEL= AMMOD,BETA,FC
       INPUT<1.0+S*BETA>N1
       N1<S*COSINE(FC*TIME)>OUTPUT
END
```

RAMMOD

```
MODEL=RAMMOD,BETA,FC
       INPUT<1.0+S*BETA>N1
       N1<SPLIT>OUTPUT
END
```

(f) Application — An example of using the model in a system is as follows:



INPUT < SINE (1.0) > N1

N1 < AMMOD (1.0, 100 E3) > N2

Figure B-10.

B.3.2.2 Linear Frequency Modulator (FMMOD and RFMMOD)

(a) Function Description - The linear Frequency Modulator Model provides a classical model for this type of angle modulation. This model is written in the SYSTID language. The carrier output magnitude is defined as unity. The two forms of the modulator are functionally identical; the difference lies in their use (section B.3.2).

(b) Parameters

DF = frequency deviation of the carrier per unit input

FC = carrier frequency

(c) Detailed Description – The FM process is described as follows:

$$w_o(t) = 2\pi FC + 2\pi * DF * INPUT\ (t)$$

$$e_o(t) = COS\ (2\pi FC\ t + 2\pi * DF \int INPUT\ (t)\ dt)$$

NOTE: For RFMMOD, FC = O in the above and the output consists of the complex baseband signal.

(d) Block Diagram

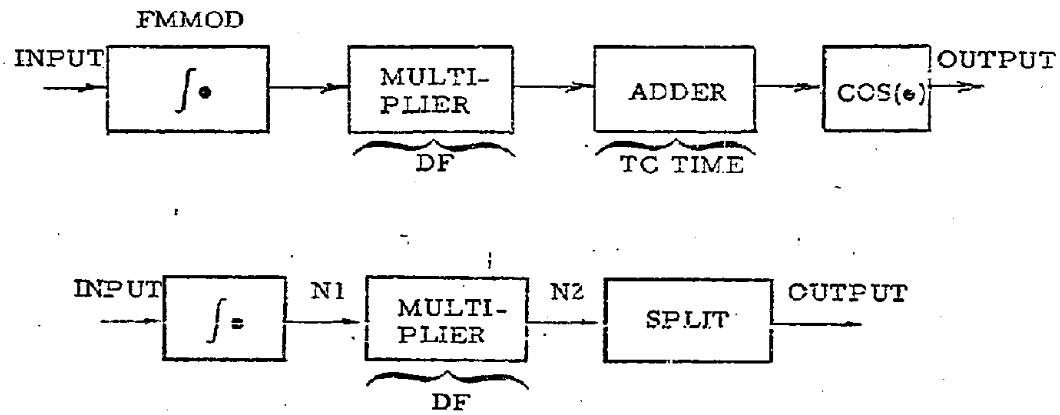


Figure B-12.

(e) Listing

FMMOD

```
MODEL, FMMOD,DF,FC
       INPUT<INTGRT>N1
       N1<COSINE(DF*$+FC*TIME)>OUTPUT
END
```

REMMOD

```
MODEL=RFMMOD,DF,FC
       INPUT<INTGRT>N1
       N1<$*DF>N2
       N2<SPLIT>OUTPUT
END
```

(e) Listing

PMMOD

```
MODEL=PMMOD,BETA,FC
        INPUT<COS(2.*PI*FC+$*BETA)>OUTPUT
END
```

RPMMOD

```
MODEL=RPMMOD,BETA,FC
        INPUT<$*BETA>N1
        N1<SPLIT>OUTPUT
END
```

(f) Application - These two linear PM modulators are used as block elements in two distinct cases. PMMOD being used when no carrier translation is desired; RPMMOD when carrier translation is required. Section 2.0 describes in detail the distinction in the use of RF section simulation. An example for referencing the model is as follows:

NX < PMMOD (1.0, 10.E3) > NY

(f) Application — These two linear FM modulators are used as block elements in two distinct cases. FMMOD is used when no carrier translation is desired; RFMMOD when carrier translation is required. Section 2.0 describes in detail the distinction in the use of RF section simulation. An example for referencing the model is as follows:

NX < FMMOD (1.0, 10.E3) > NY

B.3.2.3 Linear Phase Modulator (PMMOD and RPMMOD)

(a) Functional Description - The linear phase modulator provides a classical model for this type of angle modulation. The model is written in the SYSTID language. The carrier output magnitude is defined as unity. The two forms of the modulator are functionally identical, the difference being in their use (section 3.0).

(b) Parameters

BETA = Phase (Radians) deviation per unit input

FC = Carrier frequency

(c) Detailed Description - The PM process is described as follows:

$e_o(t)$ = Cos (2 *FC * t + BETA * INPUT (t))

NOTE: Modulation Index - BETA * $Input_{max}$. For RPMMOD, FC=0 in the above and the output consists of the complex baseband signal.

(d) Block Diagram

PMMOD

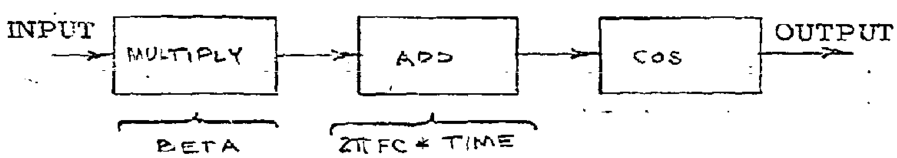


Figure B-13.

RPMMOD

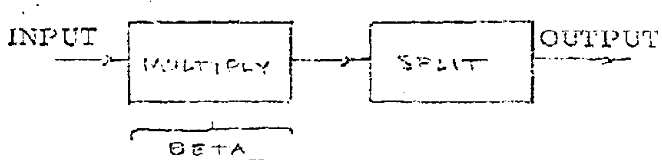


Figure B-14.

B.3.2.4 Delta Modulation (DELMOD)

(a) Functional Description - Delta modulation is a coded modulation system nearly as efficient as PCM, requires more bandwidth than PCM, but has much simpler circuitry. These advantages make delta modulation quite attractive as a standard model. The delta modulator model is written in the SYSTID language. The output waveform magnitude is defined as ±1.

(b) Parameters

PW = Pulse width (unit time)

PPS = Pulse repetition rate (pulses/unit time)

(c) Detailed Description - In a delta modulation system, only the changes in signal amplitude from sample to sample are output. The process consists of utilizing a pulse generator, pulse modulator, an integrator, and a difference circuit.

let $e_{(t)} = \text{input}_{(t)} - \text{Output}_{(t)}\ dt$

where $\text{output}_{(t)} = \text{Sign}\ (e_{(t)}) * \delta(t)$

where $\delta(t)$ is a finite pulse of width PW

The above process is clocked at a repetition rate of PPS

(d) Block Diagram

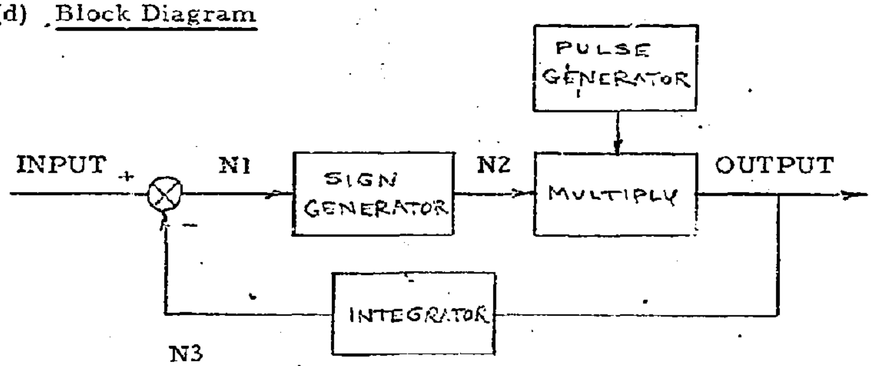


Figure B-15.

(e) Listing

DELMOD

```
MODEL=DELMOD,PW,PPS
        INPUT<S-TAP1>N1
        N1<S/ABS(S)>N2
        N2 < S*PULSE(PPS,0.,DT,PW,DT) > OUTPUT
        OUTPUT<INTGRT>N3 'TAP 1
END
```

(f) Application - The delta modulator is normally used in pulse coding an audio signal for subsequent transmission via a modulated carrier. As such, this model will be mainly used in generating a baseband signal.

B.3.2.5 Multi-Level Coding — M-ary Codes (MLTPCM)

(a) Functional Description - This code modulator produces an m-level signal based upon a serial input bit stream (polar or binary). A serial-to-parallel conversion is made and a gray code level selection is used in generating an output bit stream at a rate of M * BT. Output levels are normalized to a peak value of unity (+1), represented by equal levels separated by 1/M. The minimum level (0) corresponds to the null symbol.

The multi-level coder may be used to drive the FM and PM modulators to produce m-ary PM carrier signals. Attention should be made to appropriate scaling of the MLTPCM output to produce the correct modulating signal magnitudes.

(b) Parameters

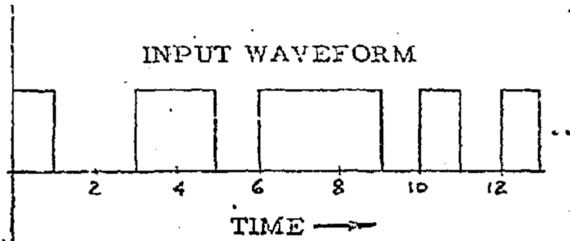
BT = Bit Time

M = Number of levels (symbols)

(c) Detailed Description - The serial bit stream is loaded into an N-bit register. At time M*BT, the register is sampled and the output waveform value (0 to +1) is determined from the reflected (gray) code in the register. The output is held at this level for M*BT, at which time the register is sampled for the next bit. The following diagram depicts the time sequence for an arbitrary input bit stream. The parameters for this example are:

M = 16 (4 Bits)

BT = 1



1 0 0 1 1 0 1 1 1 0 1 0 1 . . .

BIT STREAM

Register History

Time	Contents 1234	Output
0	1---	0
1	10--	0
2	100-	0
3	1001	0
4	1---	14/15
5	10--	14/15
6	101-	14/15
7	1011	14/15
8	1---	13/15
9	10--	13/15
10	101-	13/15
11	1010	13/15
12	1---	12/15
13	10--	12/15
14	100-	12/15
15	1001	14/15
16	0---	14/15
17	00--	14/15
18	001-	14/15
19	0011	14/15
20		2/15

Figure B-16.

(d) Block Diagram - Functionally, the model is represented as follows:

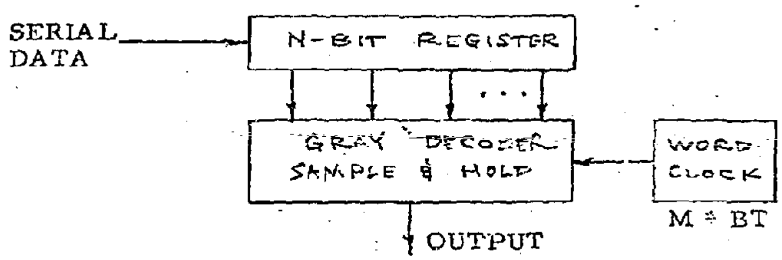


Figure B-17.

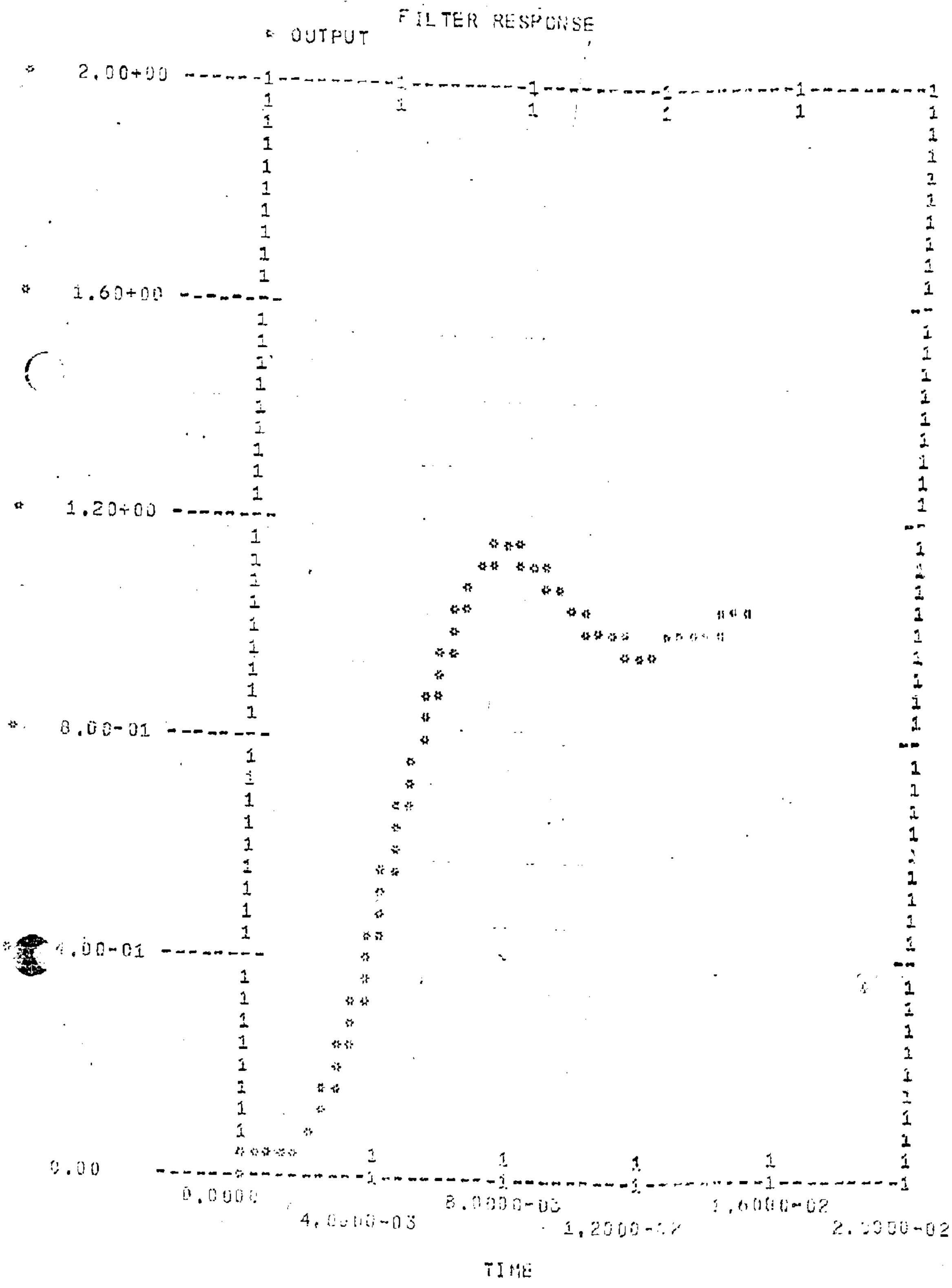
(e) Listing - The model is written in FORTRAN IV.

```
"I FOR MLTPCM,MLTPCM
      SUBROUTINE MLTPCM(BT,M)
      INCLUDE HEDFOR,LIST
      XV=V(ZZ+1)
      INVAL=0
      IF(V(INV).GT.0.) INVAL=1
      N=ALOG10(FLOAT(M))/.30102
      IB=T/BT+.01*DT
      IB=MOD(IB,N)
      IF(V(ZZ+2)),,100
C *** PURGE THE REGISTER
      XV=V(ZZ+3)*FLOAT(2**(N-1))
      DO 10 I=2,N
      IZ=ZZ+I+2
      V(IZ)=AMOD(V(IZ-1)+V(IZ),2.)
   10 XV=XV+V(IZ)*FLOAT(2**(N-I))
      V(ZZ+2)=1.
C *** CLOCK IN THE NEW BIT
  100 CONTINUE
      V(ZZ+3+IB)=INVAL
      V(ZZ+1)=XV
      V(OUTV)=XV/FLOAT(M-1)
      IB1=(T+DT)/BT +.01*DT
      IB1=MOD(IB1,N)
      IF(IB1.EQ.0.AND.IB.NE.0) V(ZZ+2)=0.
      ZZ=ZZ+N+2
      RETURN
      END
```

(f) Application - The m-ary coder is referenced in the following way.

N1 < MLTPCM (BT, M) > N2

The signal at the input node (e.g., N1) is assumed to be a polar or binary bit stream.


FILTER RESPONSE
* OUTPUT
2.00+00
1.60+00
1.20+00
8.00-01
4.00-01
0.00
0.0000
4.0000-03
8.0000-03
1.2000-02
1.6000-02
2.0000-02
TIME

(e) Listing — The model is written in FORTRAN IV.

```
@I FOR ATOD,ATOD
      SUBROUTINE ATOD(NBIT,PEAK,BT)
      INCLUDE HEDFOR,LIST
      XVAL=V(INV)
      IB=T/BT+.01*DT
      IB=MOD(IB,NBIT)
      XT=PEAK/2.
      IF(IB)5,5,
C ***  FIND THE CURRENT THRESHOLD LEVEL
      DO 10 I=1,IB
      Y=V(ZZ+I)
   10 XT=XT+Y*PEAK/FLOAT(2**(I+1))
C ***  SET BIT IB
    5 CONTINUE
      IZ=ZZ+IB+1
      V(IZ)=-1.
      IF(XVAL-XT)100,,
      V(IZ)=1.
  100 V(OUTV)=AMAX0(V(IZ),0.)
      ZZ=ZZ+NBIT
      RETURN
      END
```

(f) Application — The A/D converter is referenced in the following way:

N1 < ATOD(NBIT, PEAK, BT) > N2

The input signal at N1 is assumed analog; the output at N2 is a binary bit stream of level 0 or 1.

B.3.3 DEMODULATORS AND DECODERS

This section provides the definition of the demodulators available to the SYSTID user. As described in section 2.0 of the main text, the capability for efficiently simulating an RF communications system relies on this ability to model such systems at baseband. This is accomplished by translation of RF components to the baseband region. Demodulators for detection of RF modulating signals are prefixed by an "R" in the following descriptions.

B.3.3.1 Amplitude Demodulator (AMDEMOD and RAMDEMOD)

(a) Functional Description - This linear Amplitude Demodulator model provides a rudimentary model for use in the SYSTID Library. The basic model is a full wave rectifier followed by a user selected filter function chosen for the particular application. This filter is external to this model. The full wave rectifier is simply an absolute value function or envelope detector for RF with its peak output value determined by the modulator used in generation of the signal.

(b) Parameters - No parameters are required.

(c) Detailed Description - The AM demodulation process is simply an absolute value function for the baseband AM detector. In the case of an RF AM detector, the output is the envelope of the carrier.:

(d) Block Diagram

AMDEMOD

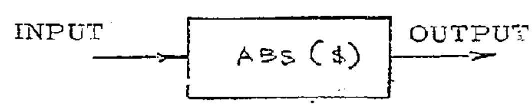


Figure B-18.

RAMDEMOD

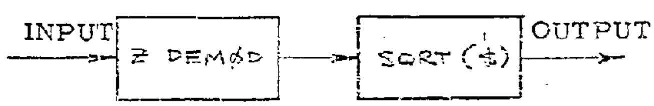


Figure B-19.

(e) Listing

AMDEMOD

```
MODEL=AMDEMOD
      INPUT<ABS(S)>OUTPUT
END
```

RAMDEMOD

```
MODEL=RAMDEMOD
      INPUT<ZDEMOD>N1
      N1< V(VCIN)>OUTPUT
END
```

NOTE: The ZDEMOD function is a cononic SYSTID model, for use in the RF domain to derive the envelope squared and instantaneous frequency versus time functions.

(f) Application - The AM Detector model is for use with an averaging filter to eliminate the carrier.

B.3.3.2 Frequency Demodulator (FMDEMOD and RFMDEMOD)

(a) Functional Description - The linear, idealized FM Demodulator provides a classical to simulate such a function. The models are written in the SYSTID language. The basic model is a limiter, a differentiator, and an envelope detector, which is followed by a filter function chosen for the particular application. This filter is external to this model.

(b) Parameters

DV = Output magnitude per unit frequency deviation (e.g., volts Hz)

FC = Carrier frequency

(c) Detailed description

FMDEMOD

$$\text{Limiter function} - y(z) = \begin{cases} +k & \cos(z) > 0 \\ -k & \cos(z) < 0 \end{cases} \quad \text{(B-3)}$$

Limiter output Fourier series

$$y(z) = k + \frac{4k}{\pi}\left[\cos(z) - \frac{1}{3}\cos(3z) + \frac{1}{5}\cos(5z) - \ldots\right] \quad \text{(B-4)}$$

with $z = w_c t + \phi(t)$ and eliminating the DC term

$$E_L(t) = \frac{4k}{\pi}\left[\cos(w_c t + \phi(t)) - \frac{1}{3}\cos 3(w_c t + \phi(t)) + \ldots\right] \quad \text{(B-5)}$$

the derivative.

$$\dot{E}_L(t) = \frac{dE_L(t)}{dt} = -\frac{4k}{\pi}[w_c + \dot{\phi}(t)]\sin(w_c t + \phi(t))$$
$$+ \frac{4k}{\pi}(w_c + \dot{\phi}(t))\sin 3(w_c t + \phi(t)) \ldots \quad \text{(B-6)}$$

therefore:

$$E_L(t) = \frac{4k}{\pi}(w_c + \dot{\phi}(t))\left[-\sin(w_c t + \phi(t)) + \sin 3(w_c t + \phi(t)) - \ldots\right] \quad \text{(B-7)}$$

The envelope of each term is proportional to $\omega_c + \dot{\phi}(t)$ and can be detected with full wave rectifier followed by a low pass filter. This low pass filter is external to the model. The transfer constant DV is applied internal to the rectifier output $E_e(t)$ as:

$$E_o(t) = \left[E_e(t) * \frac{\pi}{4k} - 2\pi FC\right] * \frac{DV}{2\pi} \qquad \text{(B-8)}$$

RFMDEMOD

The RF FM Demod model is coupled directly to the simulation of the translated RF section of a model as described in Appendix A. Since the output of any model contains both real and imaginary parts for the translated RF section case, i.e.,

$$y(t) = y_r(t) + jy_i(t)$$

$$= \sqrt{y_r^2(t) + y_i^2(t)}\; e^{j\tan^{-1}[y_i(t)/y_r(t)]} \qquad \text{(B-9)}$$

$$y(t) = A(t)e^{j\phi(t)}$$

$$\phi(t) = \tan^{-1}[y_i(t)/y_r(t)] \qquad \text{(B-10)}$$

In order to avoid the arc tangent function and because the instantaneous frequency is of interest in FM, the time derivative is computed:

$$\eta_o(t) = \frac{d\phi(t)}{dt} = \frac{y_r(t)\frac{dy_i(t)}{dt} - y_i(t)\frac{dy_r(t)}{dt}}{y_r^2(t) + y_i^2(t)} \qquad \text{(B-11)}$$

The sign information normally associated with A(t) is carried in the phase and is detected by sensing when both $y_r(t)$ and $y_i(t)$ change sign from one sample to the next. This step change in phase is implemented by $\pm\pi$ causing $\eta_o(t)$ to contain an impulse of magnitude (which is π/DT in the sample data simulation). The impulse is selected alternately as plus and minus since for A(t) to have passed from positive to negative, it must have passed from negative to positive. The above computations are performed in the library function ZDEMOD, which is written in FORTRAN.

B.3.3.3 Phase Demodulator (PMDEMM and RFPDEM)

(a) Functional Description - The phase demodulator model presented here is simply the integral of an FM demodulator output, in the case of the Baseband Demod (PMDEMM). The RF Phase Demod (RFPDEM) represents an ideal wide band phase demodulator.

(b) Parameters

FC - Center Frequency

DV - Output Magnitude per Unit Phase Deviation (Volts/Radians)

(c) Detailed Description

PMDEMM

The phase demodulator output is given by

DV * $\int$ FMDEMOD(t) dt where FMDEMOD(t) is the output of an FMDEMOD with a sensitivity of 1v/radian.

RFPDEM

The RF phase demodulator simply integrates the output of RFMDEMOD (Section 3.3.2), i.e., $\int$ RFMDEMOD(t) with transfer coefficient DV.

(d) Block Diagram

PMDEMM

Figure B-22.

RFPDEM

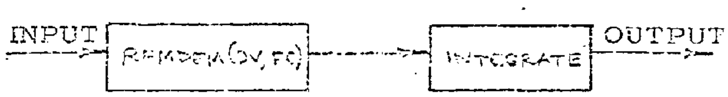


Figure B-23.

(d) Block Diagram

FMDEMOD

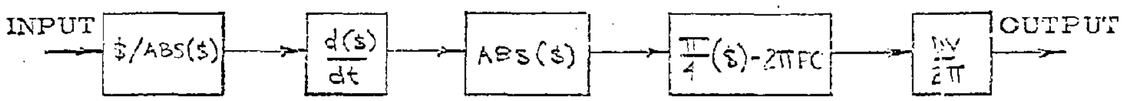


Figure B-20.

RFMDEMOD

Figure B-21.

(e) Listing

FMDEMOD

```
MODEL=FMDEMOD,KV,FC
        INPUT<$/ABS($)>N1
        N1<DIF>N2
        N2<DV*(ABS($)/8.0-FC)>OUTPUT
END
```

REMDEMOD

```
MODEL=RFMDEMOD,KV,FC
        INPUT<ZDEMOD>N1
        N1 < V(VCIN)*DV*2.0*PI > OUTPUT
END
```

(f) Application – The FM demodulators are to be used with external filtering.

(e) Listing

PMDEMM

```
MODEL = PMDEMOD,DV,FC
        INPUT < PMDEMOD(1.0,FC) > N1
        N1 < DV*S > N2
        N2 < INTGRT > OUTPUT
END
```

RFPDEM

```
MODEL = RPMDEMOD,DV,FC
        INPUT < RFMDEMOD(DV,FC) > N1
        N1 < INTGRT > N2
END
```

(f) Application — The phase demodulators are to be used with external filtering.

B.3.3.4 Matched Filter (MFILTR)

(a) Functional Description - The matched filter model is a simple integrate and dump routine clocked to the Bit Time.

(b) Parameters

BT - Bit Time

(c) Detailed Description - The integrate-Dump process is asynchronous and starts at time equal to zero, requiring the user's discretion in its use. The model is written in FORTRAN.

(d) Block Diagram

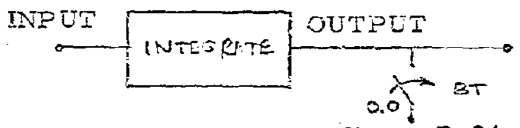


Figure B-24.

(e) Listing - Following is a listing of the FORTRAN routine name MFILTR.

```
"I FOR MFILTR,MFILTR
      SUBROUTINE MFILTR(BT)
      INCLUDE HEDFOR,LIST
      INTEGER Z
      Z=ZZ
      V(Z+2)=V(Z+1)
      V(Z+1)=V(VIN)
      V(Z+3)=V(VOUT)
      V(VOUT)=V(VOUT)+DT2*(V(Z+1)+V(Z+2))
      IF(INT((T-DT)/BT)-INT(T/BT))999,,100
      DO 10 I=1,3
   10 V(Z+I)=0.
  100 ZZ=ZZ+3
      RETURN
  999 WRITE(6,7000)
 7000 FORMAT(1H1,' ERROR IN MATCHED FILTER MODEL')
      STOP
      END
```

(f) Application — The matched filter model must be used with caution since it is asychronous.

B.3.3.4 Frequency Demodulator with Feedback (FMFB)

(a) Functional Description - The Frequency Demodulator with Feedback Model (FMFB) provides an alternate demodulation process capability. The basic model, written in the SYSTID language, consists of a multiplier, IF filter, FM discriminator (FMDEMOD) and a Voltage Controlled Oscillator (FMMOD). The The RF filter and post detection low pass filter are external to the model.

(b) Parameters

NIF - IF Filter Order (≤10)

NTYPE - Type of Filter Function:

= 1 for Butterworth

= 2 for Chebyshev

= 3 for Bessel

= 4 for Butterworth-Thomson

= 5 for Elliptic

AR - Amplitude Ripple (dB)

EM - M-Factor for Butterworth-Thomson

Stop Band Ratio for Elliptic (if positive)

Modular Angle (Degrees) for Elliptic (if negative)

BIF - IF Filter Bandwidth

GAIN - Detector Gain + VCO Amp Gain

FIF - IF Frequency

FC - Carrier Frequency

DV - FM Discriminator Constant (Volts/Hz)

DF - VCO Deviation (e.g., Hz/Volts)

(c) Detailed Description

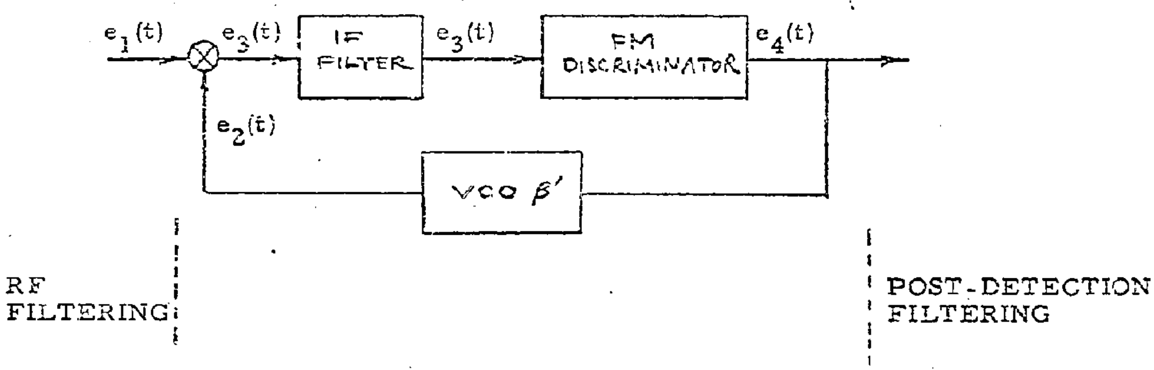


Figure B-25.

Let

$$e_1(t) = A(t) \cos (\omega_{\not{c}} t + \phi_i(t)) \qquad \text{(B-12)}$$

and

$$e_2(t) = -B \sin (\omega_{VCO} t + \theta(t)) \qquad \text{(B-13)}$$

where

$$\phi_i(t) = \beta \int S(t)$$

$$\theta(t) = \beta' \int e_4(t)$$

and

$$e_3'(t) = \frac{A(t)B}{2} \Big\{ \sin [\omega_{IF} t + \phi_i(t) - \theta(t)] - \sin [(\omega_{IF} + \omega_{\not{c}})t + \phi(t) + \theta(t)] \Big\} \qquad \text{(B-14)}$$

assuming the IF filter passes only the first term

$$e_3'(t) = \frac{A(t)B}{2} \sin (\omega_{IF} t + \phi_i(t) - \theta(t)) \qquad \text{(B-15)}$$

the output of the FM discriminator is ideally

$$e_4(t) = \frac{d}{dt}\left[\phi_i(t) - \theta(t)\right] DV = DV\left[\beta S(t) - \beta' e_4(t)\right]$$

$$e_4(t) = DV \frac{\beta}{1 + \beta'} S(t) \qquad \text{(B-16)}$$

(d) Block Diagram

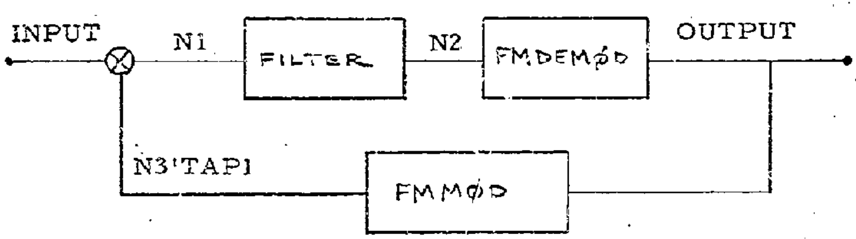


Figure B-26.

(e). Listing

FMFB

```
MODEL = FMFB,NIF,NTYPE,AR,EM,BIF,FIF,GAIN,FC,DV,DF
      INPUT < S*TAP1 > N1
         N1 < FILTER(NIF,NTYPE,3,FIF,BIF,0.,GAIN,AR,EM) > N2
      N2 < FMDEMOD(DV,FIF) > OUTPUT
      OUTPUT < FMMOD(DF,FC-FIF) > N3 'TAP1
END
```

(f) Application — The FMFB demodulator utilizes a tracking principle to achieve good SNR performance.

B.3.4 FILTERS

The modeling of filters, or continuous functions relies on several computer routines previously developed by SAI which perform the various functions described in Appendices A and C. For ease in their use, an interface routine is written called FILTER, with several entry points as is explained below.

When utilizing any Filter model in a simulation of an RF link, translation of the filter to the baseband region is necessary for efficient simulation. The translation parameters are reflected in the reference to the Filter model.

(a) Variable Definitions

NP = Filter order

IF = Filter function

= 1 for Butterworth

= 2 for Chebyshev

= 3 for Bessel

= 4 for Butterworth-Thomson

= 5 for Elliptic

IG = Filter Geometry

= 1 for Low Pass

= 2 for High Pass

= 3 for Band Pass

= 4 for Band Stop

AR = Amplitude Ripple (dB)

EM = M-factor for Butterworth-Thomson or stop-band ratio or modular angle for Elliptic functions

FX = Arithmetic center frequency

BW = Bandwidth

FC = Translation frequency (i.e. translate such that FC becomes zero)

AMP = Voltage gain at FX

(b) Detailed Description - The detailed description for generating the various filter functions in the s domain is described in Appendix A. Once the function of s is known, the bilinear z transform is derived. In order to reduce round-off errors, the function is represented by second degree sections, or quadratic factors. The bilenear z-transform converts a factor of s to a factor of the same degree in z, that is:

$$\frac{O(s)}{I(s)} = \frac{a_2 s^2 + a_1 s + a_o}{b_2 s^2 + b_2 s + b_o} \Bigg| \frac{F_2 z^{-2} + F_1 z^{-1} + F_o}{D_2 z^{-2} + D_1 z^{-1} + D_o} = \frac{O(z)}{I(z)} \qquad \text{B-17)}$$

NOTE: D_o is normalized to entry

z^{-1} is a unit delay

The difference equation for one of the quadratic factors will then be:

$$O(t) = F_2 I(t - 2DT) + F_1 I(t - DT) + F_o I(t)$$
$$- D_2 O(t - 2\,DT) - D_1 O(t - DT)$$

where DT is the sampling time.

If the filter is not being translated, the quadratic factors are cascaded. However, when translating the filter (described in appendix A), both real and imaginary coefficients of s result and the function is represented by parallel quadratic factors. The representation of the function as a sum of terms rather than a product eliminates the necessity of computing the roots of a polynominal in determining $K_r(s)$ and $K_i(s)$, (see appendix A).

When using a translated filter, the run time can be reduced significantly in trade for exact representation of the filter. Reduction of approximately one-fourth is realized by using an equivalent low pass function; or one-half by assuming symmetry of the filter (i.e. $K_{i(s)} = 0$). This is accomplished when referencing one of the functions as described below.

(c) Useage - The general reference is:

FILTER (NP, IF, IC, FX, BW, FC, AMP, AR, EM)

All variables must be included whether they are applicable or not. The above reference is translated verbatum into a FORTRAN call statement.

Alternate references to filters are as follows:

BUTTERWORTH (NP, IG, FX, BW, FC, AMP)

CHEBYSHEV (NP, IG, FX, BW, FC, AMP, AR)

BESSEL (NP, IG, FX, BW, FC, AMP)

BUTTERWORTH THOMSON (NP, IG, FX, BW, FC, AMP, EM)

ELLIPTIC (NP, IG, FX, BW, FC, AMP, AR, EM)

Special Cases:

QFACTOR (AMP, A1, A2, A3, A4, A5, A6)

used to describe:

$$AMP * \frac{A1s^2 + A2s + A3}{A4s^2 + A5s + A6}$$

LEADLAG (AMP, F1, F2, F3, F4)

used to describe:

$$AMP * \frac{\left(\frac{s}{2\ F1} + 1\right)\left(\frac{s}{2\ F2} + 1\right)}{\left(\frac{s}{2\ F3} + 1\right)\left(\frac{s}{2\ F4} + 1\right)} \qquad (B-18)$$

if:

F2 = 0 then one zero is eliminated

F1 = 0 then both zeros are eliminated

F4 = 0 then one pole is eliminated

F3 = 0 then both poles are eliminated

LEAD FUNCTION (AMP, F1, F2, F3)

used to describe:

$$AMP * \frac{\left(\frac{s}{2\ F1} + 1\right)\left(\frac{s}{2\ F2} + 1\right)}{\left(s\frac{s}{2\ F3} + 1\right)} \qquad (B-19)$$

and is otherwise the same as the LEADLAG function.

Note that the LIBRARY index defines the reference name allowable for any model element and can be changed at the user's discretion.

When utilizing the above function, any time FC > 0, an RF filter is referenced (i.e. complex inputs and outputs) rather than a baseband filter (i.e. real inputs and outputs). Table B-3 describes the conditions set up by FC and FX.

TABLE B-3

FC	FX	IG	Result
0	-	-	Baseband filter simulation
>0	>0	3	RF translated filter
>0	0	3	Symmetric translated filter ($\Omega = \infty$)
>0	0	[illegible]	Equivalent low pass function

B. 3. 5 MISCELLANEOUS MODELS – LIMITERS

B. 3. 5. 1 Hard Limiters

A hard limiter in baseband modeling is simply \$/ \$. However, in the RF region, a hard limiter is described as follows:

$$Y(t) = \left[Y_r^2(t) + Y_i^2(t)\right] e^{j \tan^{-1}(Y_i/Y_r)}$$

$$Y'(t) = Ce^{j \tan^{-1}(Y_i'/Y_r')}$$

where (B-20)

$$Y_i'(t) = \frac{CY_i(t)}{\left[Y_r^2(t) + Y_i^2(t)\right]}$$

$$Y'(t) = \frac{CY_r'(t)}{\left[Y_r^2(t) + Y_i^2(t)\right]}$$

Thus the RF limiter is referenced as:

RF LIMITER

with its output normalized to unity (C = 1).

B. 3. 5. 2 Soft Limiters

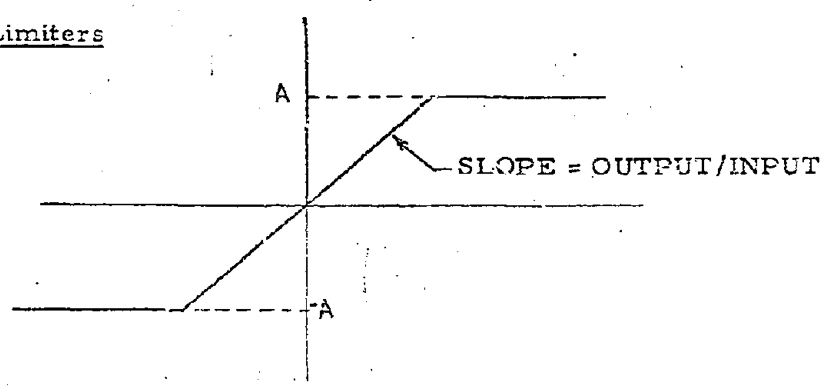


Figure B-27.

APPENDIX C

THEORETICAL BASIS FOR FILTER MODELS

C.1 TRANSFER FUNCTION RESPONSE FROM POLE-ZERO LOCATION

A general transfer function may be expressed in the following form:

$$H(s) = \frac{A \prod_{i=1}^{M} (s - Z_i)}{\prod_{i=1}^{N} (s - P_i)} \qquad \text{(C-1)}$$

where A is some constant multiplier

s = $j\omega$ = complex frequency

P_i = Complex Pole $(s + j\omega_i)$

Z_i = Complex Zero $(s + j\omega_i)$

N = order of the filter (Number of poles)

M = Number of zeros

The poles and zeros are always either complex conjugate pairs or single real values. The roots of most of the functions of interest consist entirely of conjugate pairs, if even, and have one additional real root, if odd.

The computation takes advantage of the computer's ability to do complex arithmetic. The response at a particular frequency (ω_a) is obtained by substituting into equation C-1 which yields

$$H(j\omega_a) = \frac{A \prod_{i=1}^{M} (j\omega_a - Z_i)}{\prod_{i=1}^{N} (j\omega_a - P_i)} = \alpha + j\beta \qquad \text{(C-2)}$$

The complex result has a magnitude and phase response given by the following expressions

$$|H(j\omega_a)| = \sqrt{\alpha^2 + \beta^2} = \text{Magnitude Response}$$

$$\mathrm{Tan}^{-1}\,[\beta/\alpha] = \text{Phase Response}$$

The multiplicative constant, A, is used most often in the program as a normalizing factor. In general, it is desired to make the response be unity at DC.[1] This is accomplished if A is computed as

$$A = \frac{\prod_{i=1}^{N} |P_i|}{\prod_{i=1}^{M} |Z_i|} \qquad \text{(C-3)}$$

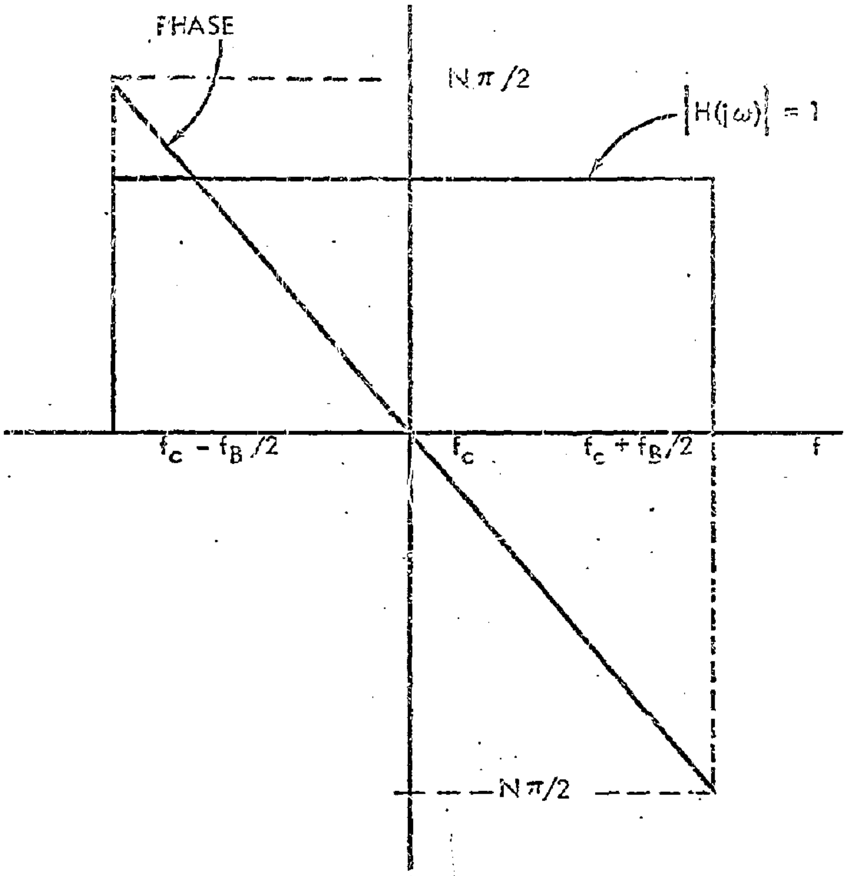
All that is needed then is to determine the poles (and zeros for the elliptic function Case) of the different types of filters. Since each filter is derived differently, the roots of each are found differently and a section is devoted to each type. The poles found are for the normalized low pass (1 rad/sec bandwidth). The response for high pass, etc., will be discussed in the section on transformations.

C.2. GENERATION OF FILTER PROTOTYPE TRANSFER FUNCTIONS

C.2.1 Ideal Filter

An ideal (or zonal) filter is defined as one which has unity gain and linear phase over some bandwidth f_B. This filter has been included in the Filter Subroutine because of its usefulness in various analysis problems. Its characteristics are given in figure C-1.

[1] For low pass functions, otherwise at center frequency for bandpass functions.



Note that the time delay is $N/2f_B$

Figure C-1. Amplitude and Phase Response for Ideal (zonal) Filter.

C.2.[illegible] Butterworth Filters

A Butterworth or maximally flat amplitude filter has a magnitude response given by

$$|H(j\omega)|^2 = \frac{1}{1 + \omega^{2N}} \qquad \text{(C-4)}$$

where N is the order of the filter.

This is an approximation to an ideal low pass as shown in figure C-2. The higher the order, N, the nearer the filter response approaches the ideal.

The transfer function may be obtained by substituting $s^2 = -\omega^2$

$$|H(s)|^2 = H(s)\,H(-s) = \frac{1}{1 + s^{2N}\,(-1)^N} \qquad \text{(C-5)}$$

The roots of equation C-5 are given by the expression in equation C-6

$$s_i = \exp\left[j\pi\left(\frac{N + 1 + 2k}{2N}\right)\right],\ k = 0,\ 2N-1 \qquad \text{(C-6)}$$

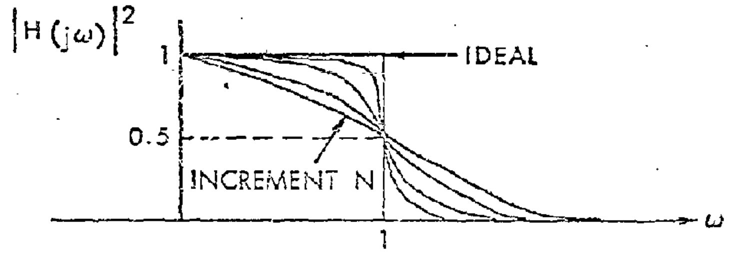


Figure C-2. Butterworth Response Approximation to an Ideal Response.

and determine the poles of the filter function. The poles of $H(s)^2$ will be equally spaced on a unit circle in the complex frequency plane. Those belonging to H(s) will be only in the left half plane. The pole locations are illustrated for odd and even order Butterworth functions in figures C-3(a) and C-3(b), respectively. The equations for the poles for both odd and even order are given by equations C-7 and C-8.

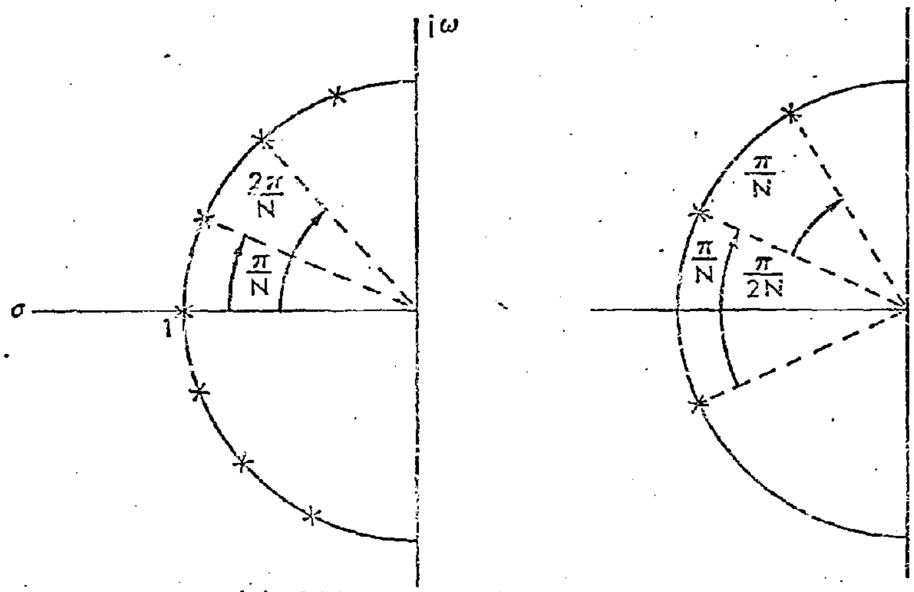
Odd:

$$\left.\begin{matrix} P_{2i-1} \\ P_{2i} \end{matrix}\right\} = -\cos\left(\frac{i\pi}{N}\right) \pm j \sin\left(\frac{i\pi}{N}\right) \quad i = 1, 2, \ldots, \frac{N-1}{2} \qquad \text{(C-7)}$$

$$P_N = -1 + j0$$

Even:

$$\left.\begin{matrix} P_{2i-1} \\ P_{2i} \end{matrix}\right\} = -\cos\left[\frac{(i-0.5)\pi}{N}\right] \pm j \sin\left[\frac{(i-0.5)}{N}\pi\right] \quad i = 1, 2, \ldots, \frac{N}{2} \qquad \text{(C-8)}$$



(a) Odd Order (b) Even Order

Figure C-3. Butterworth Pole Locations.

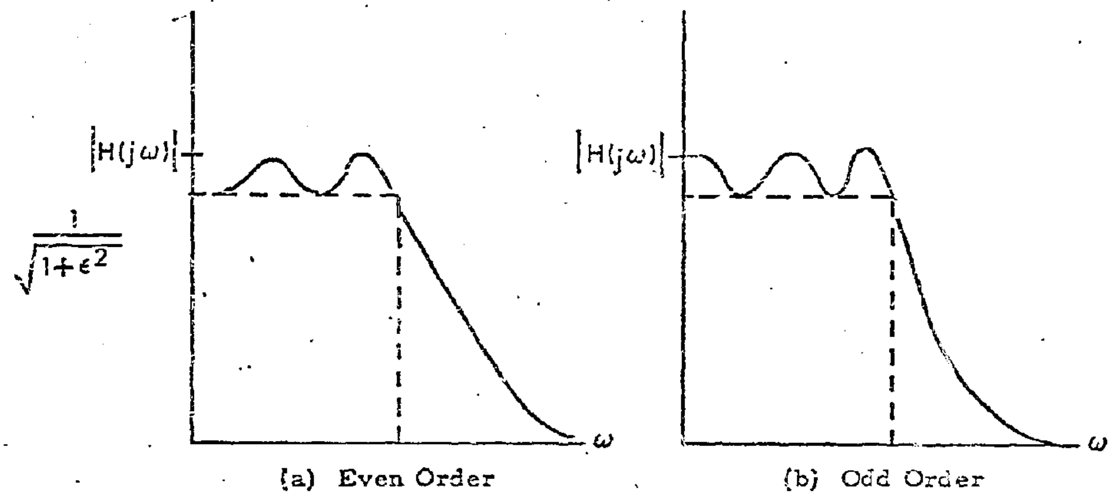


Figure C-4. Chebyshev Amplitude Response.

C.2.3 Chebyshev Filters

Chebyshev filters are characterized by an equiripple pass band as shown in figure C-4(a) and C-4(b) for even and odd orders, and a monotonic passband response. An alternate Chebyshev polynomial approximation provides an inverse response, i.e. monotone passband and ripply stopband response.

The equiripple approximation has been shown to provide the sharpest cut-off filters thus the Chebyshev is the sharpest possible all pole filter function.

The magnitude squared of the transfer function for the Chebyshev filter response function is given in equation C-9

$$|H(j\omega)|^2 = \frac{1}{1 + \epsilon^2 \, T_N^{\,2}(\omega)} \tag{C-9}$$

[1] Of all transfer functions whose zeros lie at infinity.

where

$$T_N(\omega) = \cos\left[N\cos^{-1}(\omega)\right]$$

$$= \cos\left[N\cos^{-1}(\omega)\right] \quad 0 \leq \omega \leq 1$$

$$= \cosh\left[N\cosh^{-1}(\omega)\right] \quad \omega > 1$$

T_N can be put in polynominal form, yielding the Chebyshev polynominals of order N.

The roots can be found by solving the denominator of equation for s after substituting $\omega = s/j$ and selecting the poles in the left half plane.

This expression has been solved in reference 54 and the Chebyshev poles have been shown to be on an ellipse in the s plane. It has also been shown that the Chebyshev poles are simply related to the Butterworth poles. This relationship is given by defining an intermediate variable, ϕ, to be

$$\phi = \frac{1}{N}\sinh^{-1}\left(\frac{1}{\epsilon}\right) \qquad \text{(C-10)}$$

or

$$\phi = \frac{1}{N}\left[\ln\left(\frac{1}{\epsilon} + \sqrt{1 + \frac{1}{\epsilon^2}}\right)\right]$$

The Chebyshev poles are then computed to be

$$P_i = \alpha_{B_i}\cosh(\phi) + j\,\beta_{B_i}\sinh(\phi) \qquad \text{(C-11)}$$

where α_{Bi} and β_{Bi} are the real and imaginary parts respectively of the Butterworth pole positions as defined in equations C-7 and C-8.

The value of ϵ may be computed in terms of the ripple amplitude (A_R). The ripple in dB is given by

$$A_R = -20 \log_{10}\left[\frac{1}{\sqrt{1+\epsilon^2}}\right] = 10 \log_{10}\left[1+\epsilon^2\right]$$

which yields

$$\epsilon = \sqrt{10^{A_R/10} - 1} \qquad \text{(C-12)}$$

C.2.4 Bessel Filters*

The Bessel filter is characterized by its maximally flat time delay (i.e., linear phase) characteristic. The linear phase characteristic is obtained without regard for the amplitude response and the result is a non-selective amplitude characteristic.

The transfer function for an ideal time delay (= 1 sec) is given by

$$H(s) = e^{-s} = \frac{1}{\cosh(s) + \sinh(s)} \qquad \text{(C-13)}$$

This expression cannot be expanded directly and truncated at N terms because it is not Hurwitz (all poles in left half plane) for $N > 4$. The problem is to find a Hurwitz denominator. A Hurwitz polynominal is the sum of even and odd parts of some reactance functions $m(s)/n(s)$ (even/odd).

Note that sinh is odd and cosh is an even function. If we expand the following in a series and then in a continued fraction representation the result is

$$\frac{\cosh s}{\sin h\, s} = \frac{1 + \frac{s^2}{2!} \cdots}{s + \frac{s^3}{3!} \cdots} = \frac{1}{s} + \cfrac{1}{\frac{3}{s} + \cfrac{1}{\frac{5}{s} + \cfrac{1}{\frac{7}{s} \cdots}}}$$

---

* Bessel filters, named for the Bessel polynomial, used in their realization were derived by W. E. Thomson, and are sometimes referred to as Thomson filters.

This is the form of a reactance function (all coefficients are positive) and may be truncated at the Nth step to form:

$$H(s) = \frac{K}{m(s) + n(s)} = \frac{K}{B_N(s)}$$

$$H(s) = \frac{b_o}{b_o + b_1 s --- b_N s^N} \qquad \text{(C-14)}$$

It has been shown that m(s) + n(s) is a Bessel polynomial which is defined by the following recursion relationship

$$B_N = (2N-1)B_{N-1} + s^2 B_{N-2}$$

where

$$B_o = 1$$

$$B_1 = s + 1$$

The Bessel coefficients are of the following form

$$b_k = \frac{(2N - k)!}{2^{N-k}\,(N-k)!\;k!} \quad k = 0, N$$

The poles of this function can be found by digital computer and are published in reference 58. Note that equation C-14 has been derived for 1 sec time delay, but it would be desirable to work with a function normalized to a given bandwidth. The half power bandwidths for orders N= 1 to 12 were computed and are given in table C-1.

Using these bandwidths we may then normalize the 1 sec. time delay poles to a 1 rad/sec bandwidth by using equation C-15.

$$P_i \text{ (1 rad)} = P_i \text{ (1 sec)}/BW(N) \qquad \text{(C-15)}$$

A table of the 1 radian/sec poles is given in table C-2.

Table C-1.

To be supplied.

2.1.5

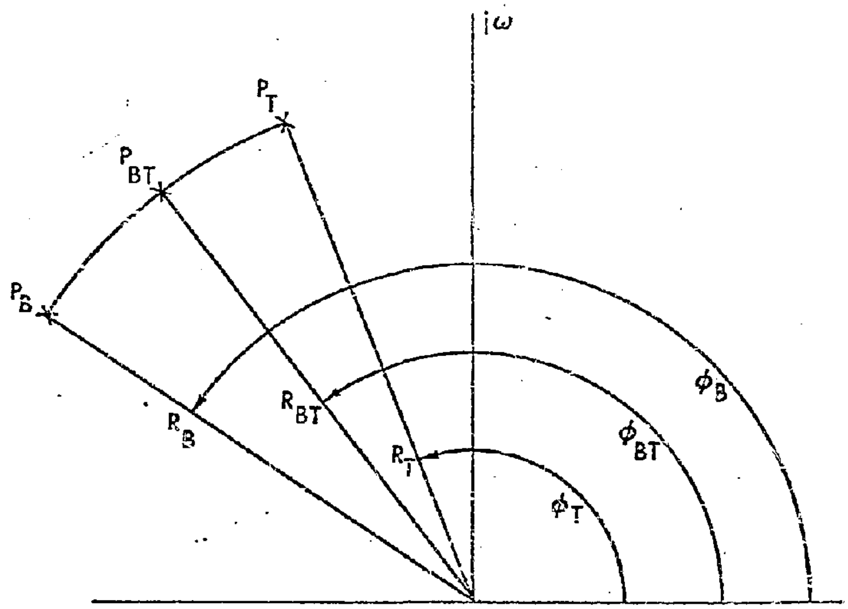
To be supplied.

Table C.2 Bessel Poles.

BESSEL POLES NORMALIZED TO 1 RAD/SEC BANDWIDTH

ORDER	COMPLEX POLES	
1	-1.ØØØØØØØØ	
2	-1.1Ø16Ø132	J Ø.636ØØ982
3	-1.Ø474Ø916	J Ø.99926443
	-1.32267579	
4	-1.37ØØ6782	J Ø.41Ø24971
	-Ø.9952Ø876	J 1.2571Ø572
5	-1.38Ø87733	J Ø.7179Ø359
	-Ø.95757655	J 1.47112432
	-1.5Ø231627	
6	-1.5714ØØ39	J Ø.32Ø89637
	-1.381858Ø8	J Ø.97147188
	-Ø.93Ø65652	J 1.66186325
7	-1.612Ø3876	J Ø.58924451
	-1.3789Ø321	J 1.19156677
	-Ø.9Ø986778	J 1.83645135
	-1.68436818	
8	-1.7574Ø841	J Ø.27286757
	-1.63693944	J Ø.82279563
	-1.37384123	J 1.38835658
	-Ø.89286973	J 1.99832587
9	-1.8Ø717Ø54	J Ø.51238373
	-1.65239649	J 1.Ø3138956
	-1.36758831	J 1.56773372
	-Ø.87839927	J 2.1498ØØ54
	-1.8566ØØ51	
1Ø	-1.84219623	J Ø.72725759
	-1.92761967	J Ø.24162347
	-1.66181Ø22	J 1.2211ØØ2[illegible]
	-1.36Ø69226	J 1.7335Ø573
	-Ø.86575689	J 2.2926Ø48Ø
11	-1.98Ø16Ø65	J Ø.45959875
	-1.86736125	J Ø.92311559
	-1.66719365	J 1.39596291
	-1.35348069	J 1.88829686
	-Ø.85451259	J 2.428Ø5946
	-2.Ø167Ø149	

STOP



P_T = Bessel Poles (1 rad/sec)

P_B = Butterworth Poles

$R_{BT} = R_T^M$

$\phi_{BT} = (1 - M)\phi_B + M\phi_T$

Figure C-5. Butterworth and Bessel Pole Transition loci.

Reference C-113 uses the 1 second time delay Bessel poles normalized so that

$$\prod_{k=1}^{N} |P_k| = 1$$

for the analysis. This produces a set of filters for which the bandwidth is a function of both the order and the parameter M. The bandwidth varies monotonically from 1 for M = 0 (Butterworth) to a number greater than 1 for the M = 1 (Bessel). Note that any normalization of poles (to change the bandwidth) only changes R and has no effect on ϕ.

From this we can see that the response of the resulting BT filter is independent (except for bandwidth scaling) of the bandwidth (and correspondingly the Rs) of the Bessel poles. Noting the monotonic behavior of the bandwidth as a function of M, it is logical to choose Bessel poles with a 1 rad/sec bandwidth, for then the bandwidth of the BT filter will be close to 1 rad/sec for all values of M. Care must be used in pairing the poles from Bessel and Butterworth. A rule that may be used is: choose a pole from each Bessel and Butterworth with the largest real part, then choose the next pair with the next largest real part, etc.

C.2.6 Elliptic Function Filter*

The elliptic function filter has been shown to be the optimum filter (sharpest cutoff for a given complexity) when both poles and zeros are permitted. The magnitude response is given by:

$$|H(j\omega)|^2 = \frac{1}{1 + \epsilon^2 R_n^2} \qquad \text{(C-17)}$$

where

$$R_n = \frac{K_1\omega\left(\omega_2^2 - \omega^2\right) --- \left(\omega^2_{n-1}\ \omega^2\right)}{\left(1 - \left(\frac{\omega_2}{\omega_s}\right)^2 \omega^2\right) --- \left(1 - \left(\frac{\omega_{n-1}}{\omega_s}\right)^2 \omega^2\right)}$$

for n = odd, and

$$R_n = \frac{K_2\left(\omega_1^2 - \omega^2\right) --- \left(\omega^2_{n-1}\ \omega^2\right)}{\left(1 - \left(\frac{\omega_1}{\omega_s}\right)^2 \omega^2\right) -- \left(1 - \left(\frac{\omega_{n-1}}{\omega_s}\right)^2 \omega^2\right)}$$

for n = even.

There are two common ways of normalizing the elliptic filter function. We have chosen to normalize to the end of the passband ($\omega_n = 1$). The bandwidth specified then will be the "ripple bandwidth". With this normalization the zeros of H(s) are inversely proportional (with constant ω_s) to the maxima in the passband. The other method

*Also called "Cauer parameter" filters, and "rational Chebyshev" filter.

normalizes to the geometric mean of the end of the passband (ω_n) and the beginning of the stop band (ω_s) so that $\sqrt{\omega_n \omega_s} = 1$ and $\omega_n = 1/\omega_S$. With the latter normalization the zeros of H(s) are reciprocals of the maxima of the passband and the critical frequencies are

$$\omega_i = \sqrt{k_2}\, SN^1 \left[\frac{iK(k_2)}{N} \right] \qquad (C-18)$$

where $K(k)$ = complete elliptical integral

$$k_2 = \frac{\omega_N}{\omega_s}; i = 1, N$$

The even ordered filters are referred to as "hypothetical filters" since they cannot be synthesized without transformers. It has been shown, (reference C-114), however, that by applying a transformation a realizable filter function can be obtained while retaining the equiripple property. This transformation moves the lowest zero of R_N (ω_1) to zero and the highest pole (ω_s/ω_1) to infinity. The resulting R_N is given by equation C-19.

$$R_N' = \frac{K_3\, \omega^2 \left(\omega_3'^2 - \omega^2\right) -- \left(\omega_n'^2 - 1 - \omega^2\right)}{\left(1 - \left(\frac{\omega_3'}{\omega_s}\right)^2 \omega^2\right) --- \left(1 - \left(\frac{\omega_{n-1}'}{\omega_s}\right)^2 \omega^2\right)} \qquad (C-19)$$

where

$$\omega_i' = \frac{1}{\omega_n''} \sqrt{\frac{\omega_i^2 - \omega_1^2}{1 - \left(\frac{\omega_1}{\omega_s}\right)^2 \omega_i^2}}$$

$$\omega_n'' = \sqrt{\frac{1 - \omega_1^2}{1 - \left(\frac{\omega_1}{\omega_s}\right)^2}}$$

The factor ω_n'' is necessary so that $\omega_n' = 1$.

The three types of filter magnitude responses are shown in figure C-6.

---

[1] SN = Jacobi sine function.

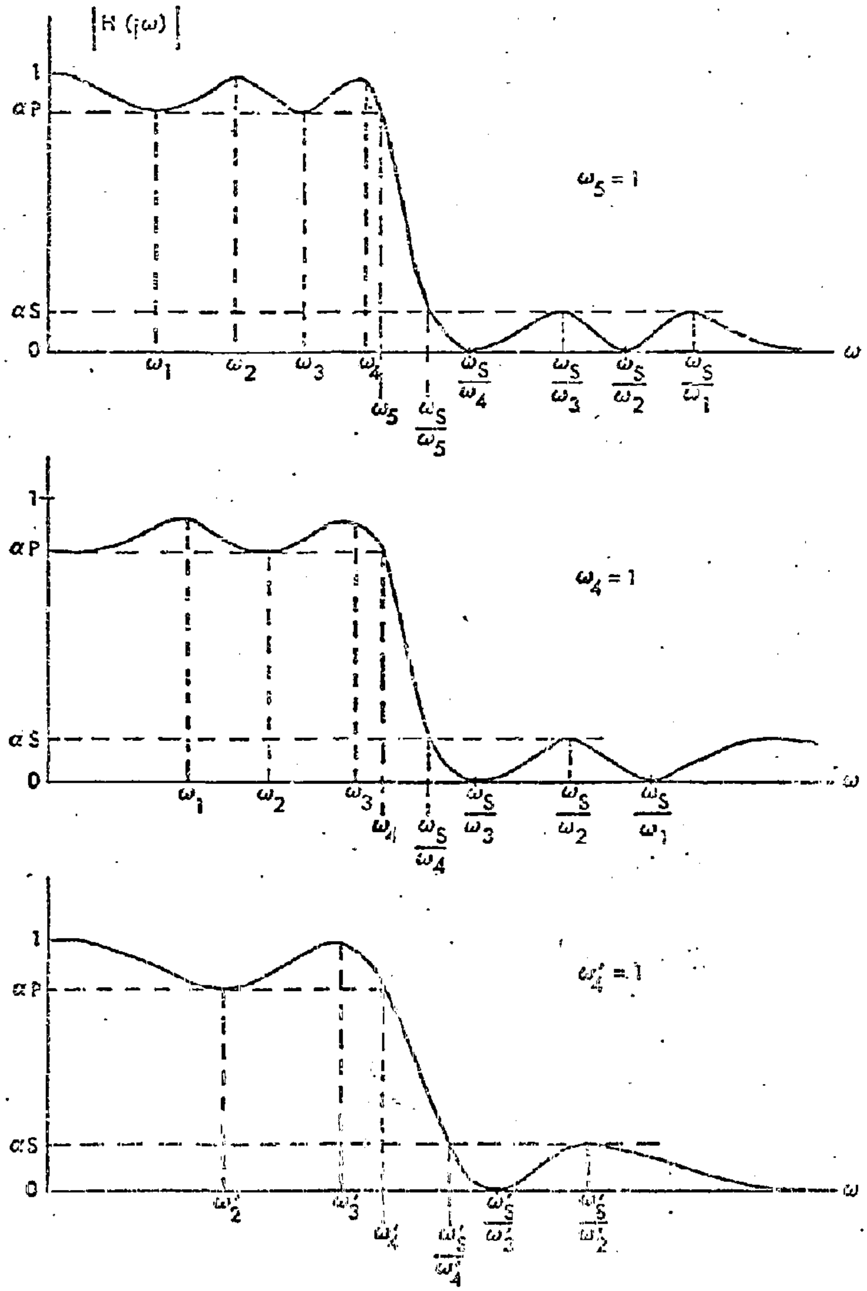


Figure C-6. Normalized and Hypothetical Filter Magnitude Responses.

The location of the ω_i was first found by Cauer from elliptic function theory to be

$$\omega_i = SN\left(\frac{iK(k_2)}{N}\right) i = 1, N \qquad \text{(C-20)}$$

where

$$k_2 = \frac{1}{\omega_S}$$

Note that $\omega_N = 1$, since $SN\left[K(k)\right] = 1$.

The parameters of the filters are:

α_P = Minimum passband gain

α_S = Maximum stop band gain

ω_s^{-1} = Transition bandwidth

N = Order (complexity) of the filter:

$$H(s) = \frac{\left(K\ s^M + s^{M-1} + \ldots a_o\right)}{s^N + s^{N-1} + \ldots b_o}$$

where

M = N-1 for odd N

M = N for hypothetical even N

M = N-2 for transformed even N

Only three of the four parameters are needed to specify the response since

$$k_1 = \sqrt{\frac{1/\alpha_P^2 - 1}{1/\alpha_S^2 - 1}} \qquad k_2 = 1/\omega_s \tag{C-22}$$

$$q(k_1) = \left[q(k_2)\right]^N \quad \text{or} \quad \frac{K(k_1')}{K(k_1)} = N\frac{K(k_2')}{K(k_2)} \tag{C-23}$$

If the order (N), α_P, and k_2 are given k_1 may be computed from equation C-23 and α_S from equation C-22. α_P and α_S are usually expressed in DB.

$$A_P = 20 \log_{10} \alpha_P \tag{C-24}$$

$$A_S = 20 \log_{10} \alpha_S \tag{C-25}$$

$$A_S = 10 \log_{10} \left[1 + \frac{1}{k_1^2}\left(\frac{1}{\alpha_P^2} - 1\right)\right]$$

An alternate set of specifications used by the Telefunken Design Tables (reference C-115) provide a smoother range of parameters which are related to the ones given above. They are:

$$\rho = \sqrt{1 - \alpha_P^2} = \text{Reflection Coefficient} \tag{C-26}$$

$$\theta = \sin^{-1}(k_2) = \text{Modular Angle} \tag{C-27}$$

A typical design might be made by specifying

1. ρ or A_p
2. θ or ω_s
3. A_s

The order needed may then be obtained from the tables in Reference C-115, or various nomographs available.

The poles and zeros of elliptic function filters have been found both by algebraic means and by use of conformal mapping (Reference C-116) through the Jacobi elliptic functions. The mapping is

$$s = j\,SN\left[-j\,(U + j\,V),\ k_2\right] \qquad \text{(C-28)}$$

$$k_2 = 1/\omega_s$$

The poles and zeros lie equally spaced on parallel lines in the W plane ($W = U + jV$) with the following coordinates:

	Zeros	Poles
U	$-K(k_2')$	$\dfrac{K(k_2)F\left[\sin^{-1}(\alpha_p),\ k_1'\right]}{N\,K(k_1)}$
V_i	$\pm\left(1 - \dfrac{2i-1}{N}\right)K(k_2)$	$\pm\left(1 - \dfrac{2i-1}{N}\right)K(k_2)$
		0 (Real pole for N odd)

$i = 1, \dfrac{N-1}{2}$

Using an identy for complex arguments of the SN( ), we can write down the poles and zeros of H(s) in the s plane

$$Z_i = j\,\frac{1}{k_2 SN(V_i,\ k_2)} \qquad \text{(C-29)}$$

$$P_i = \frac{-CN(V_i, k_2)\, DN(V_i, k_2)\, SN(U, k_2')\, CN(U, k_2') + j\, SN(V_i, k_2)\, DN(U, k_2')}{1 - SN^2(U, k_2')\, DN^2(V_i, k_2)} \quad \text{(C-30)}$$

Note that the real pole for N odd is

$$P_o = \frac{SN\ (U,\ k_2')}{CN\ (U,\ k_2')} \quad \text{(2-31)}$$

The poles and zeros given for N even are for the hypothetical filter and must be transformed to the realizable filter by following

$$\left[\ s_i' = \frac{1}{\omega_N} \sqrt{\frac{s_i^2 + \omega_1^2}{s_i^2 + \left(\frac{\omega_s}{\omega_1}\right)^2}} \qquad \begin{array}{l} s_i = \text{complex pole or zero} \\ \omega_1 = SN\left[\frac{K\ (k_2)}{N}\right] \end{array} \right. \quad \text{(C-32)}$$

This transformation is applied to both poles and zeros. The complex pair of zeros at (ω_s/ω_1) is deleted.

This transformation does not alter the pass or stop band ripple but it does increase the transition interval. The new beginning of the stop band is:

$$\omega_s' = \omega_s \left[\frac{1 - (\omega_1/\omega_s)^2}{1 - \omega_1^2}\right] \quad \text{(C-33)}$$

C. 2. 7 Other Transfer Functions

L-FILTERS

"Optimum filters with monotonic response",[1] or L-filters, are a class of filters optimum with respect to the properties: (1) monotonic response, with (2) sharpest possible cutoff.

Thus, if the L-filter amplitude function is

$$A(\omega) = \frac{A_o}{\sqrt{1 + L_n(\omega^2)}}$$

then the nth order L-filter is defined by the nth order polynomial L_n which satisfies these three properties:

1. For all ω, $\frac{d\,L_n(\omega^2)}{d\omega} \geq 0$

2. $L_n(1) = 1$ (arbitrary normalization)

3. $\left.\frac{dL_n(\omega^2)}{d\omega}\right|_{\omega = 1}$ is maximum

[1] "On the Approximation Problem in Filter Design", A. Papoulis, IRE National Convention Record, Vol. 5, pt. 2, pp. 175-185, 1957.

C.3 FREQUENCY TRANFORMATIONS

Frequency transformations are used in filter synthesis so that one basic (normalized low pass) filter may be synthesized and then other types may be derived from it. The transformations provide for both scaling of the frequency scale and for obtaining a different kind of response (e.g., bandpass) through mapping s_λ. Once the desired response is obtained by choosing the correct transformations and transformation parameters (ω_0, ω_b), the basic network can be altered appropriately without deriving the actual transfer functions. The details of the transformation are discussed in Weinberg (reference 58).

In the following discussion table C-3 will be useful to refer to the four commonly used transformations which are implemented in the FILTER program.

Poles and Zeros

The general form of a filter transfer function can be represented in the form of equation C-34. The relationships

$$H(\lambda) = \frac{A\prod_{i=1}^{M}(\lambda - ZN_i)}{\prod_{i=1}^{N}(\lambda - PN_i)} \qquad \text{(C-34)}$$

where

A = real constant $\qquad A = \dfrac{\Pi PN_i}{\Pi ZN_i}$

λ = complex frequency (normalized)

ZN_i = complex zero (normalized)

PN_i = complex pole (normalized)

M, N = number of zeros and number of poles, respectively.

between λ and the complex frequency s ($s=j\omega$) are given in table C-4. The objective is to utilize the transform relationships and the normalized low pass filter prototypes in order to obtain equation C-35 in the form of equation C-34. This objective is

$$H(s) = \frac{B \prod_{i=1}^{NZ} (s - Z_i)}{\prod_{i=1}^{NP} (s - P_i)} \qquad \text{(C-35)}$$

achieved by determining the relationship between the normalized poles and zeros (PN_i and ZN_i) and the poles and zeros in equation C-35 (P_i and Z_i). Table C-4 and C-5 presents a summary of these results which defines the transform expression for each of the four filter types. These expressions are implemented in the FILTER program.

Table C-3. Frequency Transformations.

Filter	Transformations*	
	λ	Ω
Low Pass	s/ω_b	ω/ω_b
High Pass	ω_b/s	ω_b/ω
Band Pass	$\frac{s^2 + \omega_o^2}{\omega_b s}$	$\frac{\omega^2 - \omega_o^2}{\omega\omega_b}$
Band Stop	$\frac{\omega_b s}{s^2 + \omega_o^2}$	$\frac{\omega_b \omega}{\omega^2 - \omega_o^2}$

*$s = j\omega$

$\lambda = \Phi + j\Omega$

Table 2-4. Transformation Relationships.

Filter Type	Transform
Low Pass	$\lambda = s/\omega_b$
High Pass	$\lambda = \omega_b/s$
Band Pass	$\lambda = \dfrac{s^2 + \omega_o^2}{s\,\omega_o}$
Band Stop	$\lambda = \dfrac{s\,\omega_b}{s^2 + \omega_o^2}$

ω_o = Center frequency

ω_b = Bandwidth

Group Delay

Group delay(t_g) for a filter is defined as shown in equation C-36.

$$t_g = \left. \frac{-d\,\phi(\omega)}{d\omega} \right|_{s = j\omega} \qquad \text{(C-36)}$$

where

ω = radian frequency

$\phi(\omega)$ = steady state phase response of filter.

The complex steady state phase response for the normalized filter is found by substituting $\lambda = j\Omega$ in equation C-34. This expression is given in equation C-37.

$$H(j\Omega) = A \frac{\prod_{i=1}^{M} (j\Omega - ZN_i)}{\prod_{i=1}^{N} (j\Omega - PN_i)} \qquad \text{(C-37)}$$

Table C-5. Poles and Zeros of Transformed Filters.

Type	Transformation	Transformed Multiplication Constant, B	No. of Poles Generated	Poles	No. of Zeros Generated	Zeroes
Low Pass	$\lambda = s/\omega_b$	$A \cdot \omega_b^{N-M}$	N	$\omega_b PN_i$	M	$\omega_b ZN_i$
High Pass	$\lambda = \omega_b/s$	1	N	$\omega_b PN_i$	M	ω_b/ZN_i
					M	$0 + j0$
Band Pass	$\lambda = \frac{s^2 + \omega_o^2}{\omega_b s}$	$A \cdot \omega_b^{N-M}$	2N	$1/2\left[\omega_b PN_i \pm \sqrt{\omega_b^2 PN_i^2 - 4\omega_o^2}\right]$	2M	$1/2\left[\omega_b AN_i \pm \sqrt{\omega_b^2 ZN_i - 4\omega_o^2}\right]$
					N-M	$0 + j0$
Band Stop	$\lambda = \frac{\omega_o s}{s^2 + \omega_o^2}$	1	2N	$1/2\left[\frac{\omega_b}{PN_i} \pm \sqrt{\left(\frac{\omega_b}{PN_i}\right)^2 - 4\omega_o^2}\right]$	2N	$1/2\left[\left(\frac{\omega_b}{ZN_i}\right)^2 \pm \sqrt{\left(\frac{\omega_b}{ZN_i}\right)^2 - 4\omega_o^2}\right]$
					2(N-M)	$\pm j\omega_o$

ZN_i = Normalized Zero — M = Number of Normalized Zeros

PN_i = Normalized Pole — N = Number of Normalized Poles

ω_o = Center Frequency

ω_b = Bandwidth

where

ZN_i = Normalized Zero = $\alpha_i + j\beta_i$

PN_i = Normalized Pole = $\alpha_i + j\epsilon_i$

The phase response of this function is readily computed below in equation C-38

$$\phi(\Omega) = \sum_{i=1}^{N} \tan^{-1}\left(\frac{\Omega - \beta_i}{-a_i}\right) - \sum_{i=1}^{M} \tan^{-1}\left(\frac{\Omega - \epsilon_i}{-\gamma_i}\right) \qquad \text{(C-38)}$$

In order to find the phase response of the transformed filter the equality in equation C-39 is used, where $T(\omega)$ is the appropriate transformation taken from table C-6. Using equation C-38 and

$$\phi(\Omega) = \phi\left(T(\omega)\right) \qquad \text{(C-39)}$$

and

$$t_g = -\frac{d\phi(\omega)}{d(\omega)} = -\frac{d\,\phi(\Omega)}{d\Omega}\,\frac{d\Omega}{d\omega} \qquad \text{(C-40)}$$

the group delay of a normalized lowpass filter can be obtained as given in equation C-41.

$$t_{gn}(\Omega) = \sum_{i=1}^{N} \frac{a_i}{a_i^2 + (\beta_i - \Omega)^2} + \sum_{i=1}^{M} \frac{\gamma_i}{\gamma_i^2 + (\epsilon_i - \Omega)^2} \qquad \text{(C-41)}$$

and

$$t_g(\omega) = t_{gn}\left(T(\omega)\right)\frac{d\Omega}{d\omega} \qquad \text{(C-42)}$$

The group delay for the transformed filters is derived from these relationships and the multiplier $d\Omega/d\omega$ which is a function of the type of transformation being made. Equation C-42 shows the expression for group delay as a function of the normalized low pass expression and table C-6 lists the group delay functions for the four transformations being considered. The group delay at zero frequency and at center frequency are also of interest. These functions are tabulated in table C-7. Also, there are several interesting observations to be made about the group delay, and they are as follows

- Always continuous and bounded and is zero only at $\omega = \infty$.
- Poles and zeros with a zero real part contribute nothing to group delay.
- If $|P_i| = 1$ group delay for low pass is the same as for high pass and bandpass is the same as band stop (i.e., Butterworth filters case).

C.4 EQUIVALENT NOISE BANDWIDTH

There are numerous definitions of the equivalent noise bandwidth (ENB) of a filter. The one which is implemented in this FILTER program is given in equation C-43.

$$ENB = \frac{\frac{1}{2\pi}\int_0^{\infty} |H(j\omega)|^2 \, d\omega}{|H(j\omega)|^2_{max}} \qquad \text{(C-43)}$$

This is the bandwidth of a hypothetical filter having unity gain in the passband and zero gain in the stop band followed by an amplifier with gain = H_{max}. The noise power passed by such a filter and amplifier would be

$$P_n = N_o \, ENB \, H^2_{max} \qquad \text{(C-44)}$$

where N_o is the one-sided power spectral density of the (flat) noise in watts/Hz and where $H(j\omega)$ is the filter transfer function with $s = j\omega$. The integral can be evaluated in the following way:

$$H(j\omega)^2 = H(j\omega)\, H^*(j\omega) = H(s)\, H(-s) \qquad \text{(C-45)}$$

Table C-6. Formulas for Group Delay of Transformed Filters.

Filter	$T(\omega)$	$\frac{d\,T(\omega)}{d\omega}$	$t_g(\omega)$
Low Pass	$\Omega = \omega/\omega_b$	$1/\omega_b$	$\omega_b\left[\sum_{i=1}^{M}\frac{\alpha_i}{\alpha_i^2\omega_b^2+(\beta_i\omega_b-\omega)^2}-\sum_{i=1}^{N}\frac{\gamma_i}{\gamma_i^2\omega_b^2+(\epsilon_i\omega_b-\omega)^2}\right]$
High Pass	$\Omega = -\omega_b/\omega$	$\frac{\omega_o}{\omega^2}$	$\omega_b\left[\sum_{i=1}^{M}\frac{\alpha_i}{\alpha_i^2\omega_b^2+(\beta_i\omega_b+\omega)^2}-\sum_{i=1}^{N}\frac{\gamma_i}{\gamma_i^2\omega_b^2+(\epsilon_i\omega_b+\omega)^2}\right]$
Band Pass	$\Omega = \frac{\omega^2-\omega_o^2}{\omega\omega_b}$	$\frac{\omega^2+\omega_o^2}{\omega^2\omega_b}$	$\omega_b(\omega^2+\omega_o^2)\left[\sum_{i=1}^{M}\frac{\alpha_i}{\alpha_i^2\omega^2\omega_b^2+(\beta_i\omega\omega_b+\omega_o^2-\omega^2)^2}-\sum_{i=1}^{N}\frac{\gamma_i}{\gamma_i^2\omega^2\omega_b^2+(\epsilon_i\omega\omega_b+\omega_o^2-\omega^2)^2}\right]$
Band Stop	$\Omega = \frac{\omega\omega_b}{\omega_o^2-\omega^2}$	$\frac{\omega_b(\omega^2+\omega_o)^2}{(\omega^2-\omega_o^2)^2}$	$\omega_b(\omega^2+\omega_o^2)\left[\sum_{i=1}^{M}\frac{\alpha_i}{\alpha_i^2(\omega^2-\omega_o^2)^2+[\beta_i(\omega^2-\omega_o^2)+\omega\omega_b]^2}-\sum_{i=1}^{N}\frac{\gamma_i}{\gamma_i^2(\omega^2-\omega_o^2)^2+[\epsilon_i(\omega^2-\omega_o^2)+\omega\omega_b]^2}\right]$

where $PN_i = \gamma_i + j\,\epsilon_i$ = Normalized Poles

$ZN_i = \alpha_i + j\,\beta_i$ = Normalized Zeroes

ω_o = Geometric Center Frequency

ω_b = Bandwidth

M = Number of Normalized Zeros

N = Number of Normalized Poles

Table C-7. Group Delay at Zero and Center Frequency.

Filter	$t_g(0)$	$t_g(\omega_o)$
Low Pass	$-\frac{1}{\omega_b}\sum_{i=1}^{N}\frac{P_i}{\lvert P_i\rvert^2}$	------
High Pass	$-\frac{1}{\omega_b}\sum_{i=1}^{N} P_i$	------
Band Pass	$-\frac{\omega_b}{\omega_o^2}\sum_{i=1}^{N} P_i$	$-\frac{2}{\omega_b}\sum_{i=1}^{NP}\frac{P_i}{\lvert P_i\rvert^2}$
Band Stop	$-\frac{\omega_b}{\omega_o^2}\sum_{i=1}^{N}\frac{P_i}{\lvert P_i\rvert^2}$	$-\frac{2}{\omega_b}\sum_{i=1}^{NP} P_i$

Notes: 1) Zeroes are assumed to have zero real parts.

2) Poles are real or conjugate pairs.

Substituting s = jω in the integral in equation C-43 yields

$$ENB = \frac{\left(\frac{1}{2\Pi j}\right)\left(\frac{1}{2}\right)\int_{-j\infty}^{j\infty} H(s)\,H(-s)\,ds}{H_{max}^2} = \frac{1}{2\Pi j}\,\frac{1}{2H_{max}^2}\int_C H(S)\,H(-S)\,ds \qquad (C-46)$$

where C is a closed contour in the complex s-plane. Since the right half plane poles are the mirror image of the left half plane, the contour of integration need only contain those poles in the left half plane. The integral in equation C-46 can be evaluated by finding its residues. The integrand can be represented as follows

$$H(S)\,H(-S) = -A^2 \frac{\prod_{i=1}^{NZ} (S - Z_i)(-S - Z_i)}{\prod_{i=1}^{NP} (S - P_i)(-S - P_i)} \qquad (C-47)$$

The definition of the residues of equation C-47 is given as

$$R_m(P_j) = \lim_{S \to P_j} \left[H(S)\,H(-S)\,(S - P_j) \right]$$

which reduces to

$$R_m(P_j) = -A^2 \frac{\prod_{i=1}^{NZ} \left(Z_i^{\,2} - P_j^{\,2}\right)}{2P_j \prod_{i=1}^{NP} \left(P_i^{\,2} - P_j^{\,2}\right)} \quad i \neq j \qquad (C-48)$$

substituting this result into equation C-46 yields the appropriate expression (equation C-49) for the ENB in terms of the residues of the poles in the left half of the S-plane.

$$ENB = \frac{1}{2H^2_{max}} \sum_{j=1}^{NP} R_m\,(P_j) \qquad (C-49)$$

It is important to note the following:

(1) H(s) is the transformed filter transfer function

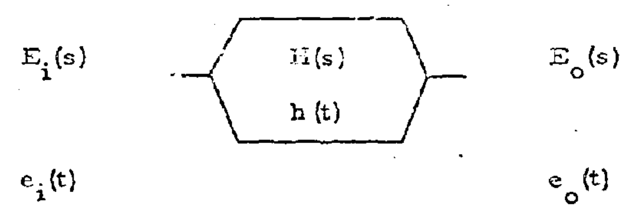
(2) H(s) can have no poles on the jω axis

(3) Multiple poles are not allowed

(4) The number of poles must be greater than the number of zeros.

C.5 TRANSIENT RESPONSE

The time response of a filter may be obtained by evaluating the inverse Laplace transforms of the input signal and transfer function.



where

$e_i(t) \Longleftrightarrow E_i(s)$	input signal
$h(t) \Longleftrightarrow H(s)$	impulse response
$e_o(t) \Longleftrightarrow E_o(s)$	output signal

The Laplace transform pairs are given in equations C-50 and C-51.

$$G(s) = \int_0^\infty g(t)e^{-st}\,dt \tag{C-50}$$

$$g(t) = \int_{\epsilon-j\infty}^{\epsilon+j\infty} G(s)e^{st}\,ds \tag{C-51}$$

We write

$$G(s) = L\,(g(t))$$

$$g(t) = L^{-1}\,(G(s))$$

The transfer function H(s) is known (computed by the program) and the Laplace transforms for various input signals have been tabulated. The Laplace transform of the output signal is:

$$E_o(s) = E_i(s)\,H(s) \tag{C-52}$$

The problem is to evaluate the inverse Laplace transform $L^{-1}(E_o(s)) = e_o(t)$ from the inversion formula

$$e_o(t) = \frac{1}{2\Pi j}\int_{c-j\infty}^{c+j\infty} E_o(s)e^{st}\,ds = \frac{1}{2\Pi j}\int_C E_o(s)e^{st}\,ds$$

In useful cases $E_o(s)$ has poles only in the left half plane ($Re(s) < 0$) or on the $j\omega$ axis and has more poles than zeros. The integrand then vanishes at $s = \infty$ for $t > 0$, and the integral can be replaced by a contour integral around the left half plane shown in figure C-7.

Then $e_o(t) = \Sigma$ residues of $E_o(s)e^{st}$ at poles of $E_o(s)$

$$E_o'(s) = \frac{B\prod_{i=1}^{NZ'}(s - Z_i)}{\prod_{i=1}^{NP'}(s - P_i)} \qquad NP' > NZ'$$

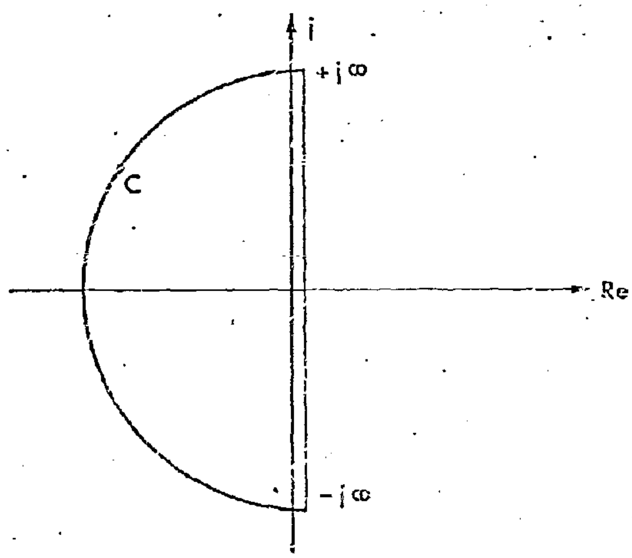


Figure C-7. Contour of Integration.

where NP' and NZ' are the number of poles and zeros of $E_o(s)$.

For simple poles and a general function G(s)

$$R(P_h) = (s - P_h) G(s) \qquad \text{(C-53)}$$

$$\lim s \to P_h$$

for poles of multiplicity m

$$R(P_h) = \frac{1}{(m-1)!} \frac{d^{m-1}}{ds^{m-1}} \left[(s-P_h)^m G(s)\right] \qquad \text{(C-54)}$$

$$\lim s \to P_h$$

If there are no multiple poles the modified residue $R'(P_i)$ may be evaluated:

$$R'(P_h) \frac{A \prod_{i=1}^{NZ'} (P_h - Z_i) e^{+P_h t}}{\prod_{i=1}^{NP'} (P_h - P_i)} = R(P_h) e^{P_h t}, \quad i \neq h \qquad \text{(C-55)}$$

Note that $R(P_h)$ is the residue at P_h of $E_o(s)$.

The output is thus:

$$e_o(t) = \sum_{h=1}^{NP'} R(P_h) e^{P_h t} \qquad \text{(C-56)}$$

Note that while this equation contains complex quantities, the sum is real.

If there is a double pole at zero (a ramp input), the residue can be found from equation C-54.

$$E_o(s) = \frac{1}{s^2} H(s)$$

$$R(o) = \frac{d}{ds}\left[H(s)e^{st}\right]$$

$$s \rightarrow o$$

$$= t\,H(s)e^{st} + e^{st}\frac{dH(s)}{ds}\Big]_{s=o}$$

$$= tH(o) + \frac{dH(s)}{ds}\Big|_{s=o}$$

$H(s) = P(s)/Q(s)$ $\qquad$ $P(s) = b_n s^n \ldots b_1 s = b_o$

$Q(s) = a_n s^n \ldots a_1 s = a_o$

$$\frac{dH}{ds} = \frac{QP' - PQ'}{Q^2}\Big|_{s=o} = \frac{a_o b_1 - b_o a_1}{a_o^2}$$

note that

$a_c = \Pi e P_i$ $\qquad$ $b_o = \Pi Z_i$

$a_1 = \Sigma P_i$ $\qquad$ $b_i = \Sigma Z_i$

$$H'(o) = \frac{\left[\sum Z_i\right]\left[\Pi P_i\right] - \left[\sum P_j\right]\left[\Pi Z_i\right]}{\left[\Pi P_i\right]^2}$$

Therefore for $e_i(t) = t$

$$e_o(t) = tH(o) + H'(o) + \sum_{i=1}^{NP} R(P_h)e^{P_h t}$$

While it is possible to compute the response for multiple poles other than the one described, these cases have not been implemented in the program described later. If a multiple pole (other than zero) is encountered, an overflow will result.