CANADIAN THESES

THÈSES CANADIENNES

NOTICE

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

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

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

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

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

AVIS

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

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

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

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

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

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

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

Canada

LINEAR SECOND-ORDER SYSTEM SOLUTION ACCURACIES WITH DIGITAL, ANALOG AND HYBRID COMPUTER METHODS

by

Guy A. Camiré

A thesis

submitted to the School of Graduate Studies

in partial fulfilment of the requirements for

the degree of Master of Science

Masters Program in Systems Science

University of Ottawa

JULY 1983

ABSTRACT

Four conceptual models for the solution of a linear second-order differential equation and their implementation on analog, hybrid and digital computers are described. A standard hybrid configuration is adopted whereby the integrations are performed on the analog computer. A discrete time equivalent of the hybrid computer model is solved on a digital computer. The difference between any two model solutions is characterized by a normalized root-mean-square error criterion. Three general digital programs were implemented to determine the accuracy of solution of the computerized models as a function of their parameters. Results are presented concerning the accuracy of the analog computer model, the hybrid computer model and its digital computer equivalent.

ACKNOWLEDGEMENT

I wish to express my warmest thanks to my supervisor, Dr. J. R. Amyot of the Division of Mechanical Engineering, National Research Council of Canada, for his invaluable guidance. I am also very grateful to his fellow-workers in the Analysis Laboratory for the help they have given me.

I owe a special dept of gratitude to Mr. P. A. Hamill for access to the equipment and computer facilities of the Analysis Laboratory.

Thanks also to Dr. L. G. Birta for his co-operation in acting as administrative supervisor at the University of Ottawa.

I would like to make special acknowledgement to Ronald Jubainville for the helpful discussions along the development of this work.

Scholarships from the Natural Sciences and Engineering Research Council and from the University of Ottawa were greatly appreciated.

TABLE OF CONTENTS

	ABSTRACT	i
	ACKNOWLEDGEMENT	ii
	TABLE OF CONTENTS	iii
1.0	INTRODUCTION	1
2.0	OVERVIEW	11
3.0	COMPENSATION THEORY	13
4.0	MATHEMATICAL MODELS	19
	4.1 Continuous-Time	20
	4.2 Discrete-Time Analytical Solution	23
	4.3 Sampled-Data	24
	4.4 Discrete-Time Equivalent of Sampled-Data	27
5.0	COMPUTERIZED MODELS	29
	5.1 System A	30
	5.2 System B	31
	5.3 System C	35
	5.4 System D	43
6.0	PERFORMANCE CRITERIA	44
7.0	PERFORMANCE ANALYSIS PROGRAMS	49
	7.1 Program RESPLT	50
	7.2 Program RESPONS	50
	7.3 Program RMSERR	51

8.0 PERFORMANCE OF COMPUTERIZED MODELS 53

8.1 System D Relative to A 56

8.2 System C Relative to A 59

8.3 System C Relative to D 60

8.4 System B Relative to A 60

9.0 CONCLUSION 61

REFERENCES 64

TABLES 73

FIGURES 80

APPENDICES

A - Normalized RMS Error Illustration Curves Obtained from SOLDE Analytical Solutions 112

B - Computerized Model Performance Data 119

C - System A Documentation 130

D - System B Documentation 133

E - System C Documentation 148

F - System D Documentation 170

G - Program PACMAN Documentation 173

H - Program RESPLT Documentation 180

I - Program RESPONS Documentation 188

J - Program RMSERR Documentation 197

K - Program Run Examples 219

L - Digital Equivalent Stability Boundaries Documentation 227

1.0 INTRODUCTION

Until the late fifty's or early sixty's, the analog computer was the most used tool to solve scientific simulation problems. Then, analog and digital computers were linked together with a communication interface. This combination was called the "Hybrid Computer". The hybrid computer, like the analog computer alone, operates in synchronous time. The information is transferred discretely and at regular intervals between both computers in a high speed loop called "Hybrid Loop".

The hybrid computer offers certain advantages. On the analog computer, for instance, where the computing time is independent of problem complexity, the operations are performed in parallel and continuously. Using the analog computer to solve large sets of differential equations is usually faster and cheaper than using a digital computer alone. The operations of the digital computer are serial and discrete and the computing time is directly related to the complexity of computations. On the other hand, the digital computer is better suited to control the various hybrid simulation runs and to perform certain operations such as function generation, table look-up, logical decision-making and data storage.

Real time simulation of dynamic systems is advantageous and sometimes necessary in some cases, especially for hardware-in-the-loop and training simulation applications. With a hybrid computer, model parameters may be programmed for manual variation at any time during a simulation run and the effect of varying a parameter may be examined as the solution progresses. A real time simulation may be designed to interact with a real world system. Although digital models may be synchronous, their dynamic applications range can be inferior to hybrid models because of the superior speed of hybrid computation. However, the gap is being reduced by parallel digital processing.

In hybrid computation, the programmer must contend with certain sources of inaccuracy. In the hybrid loop, a set of analog variables is sampled, converted into digital form, processed by the digital computer, and converted back to analog signals for analog processing. Sources of error include precision limitations of typical analog computers and analog-digital converters. Another source of error is the time delay arising from digital computation and data conversion time in a hybrid loop. Hybrid computation is constrained by this time delay in that it prevents the analog computer from using all of its high speed capabilities and the digital computer from being fully utilized since digital

computation time increases with the amount of computation. Furthermore, it seriously limits the allowable range of high frequency components in real time models. Digital to analog data reconstruction is another source of error in that the input signals to the analog computer are not the required ideal continuous signals. Instead they are signals reconstructed from discretely sampled data. Fortunately, there are methods of compensation for the effects of sampling and digital execution time upon solution accuracy and the related literature is reviewed briefly below.

Developments both in digital hardware and software have contributed to an increase in the popularity of simulation. High level simulation languages such as GPSS, CSMP, CSSL, ACSL and DARE have been developed to facilitate asynchronous digital computer modeling. Hardware devices such as array processors greatly speed up computations by providing parallel processing capabilities. Systems with built-in parallel processors, like Applied Dynamic International's AD-10, also increase speed but there are programming difficulties associated with scheduling of parallel computations. Ultimately, the speed of analog computation can only be approached digitally and its continuous nature can be obtained from discrete data only by digital to analog conversion. These digital improvements are useful to hybrid computation

also since increased digital speed leads to decreased hybrid loop time delay. Analog hardware developments such as programmable multivariable function generators and automatic patching are also improving the performance of hybrid computers, and their use is being simplified by high-level programming software such as the hybrid time-sharing system MACHYS being developed at the Technical University of Vienna.[1] It is difficult to predict the architecture of future computers precisely but the trend towards parallel computation is clear and analog computation is the supreme example of parallel processing. Hybrid computers, in their present state of evolution, are important and sometimes essential tools in computer simulation of dynamical systems, particularly for synchronous time applications.

Many authors have studied the effect of time delay in a hybrid loop. Various methods of compensation, usually based on approximations by series, have been proposed and analysed. A review of the most relevant literature on this subject follows.

Miura and Iwata[2] proposed three methods of compensation: a) analog compensation, where the derivative is multiplied by a constant and added to the output of the analog integrator, b) linear interpolation, where the output of the digital computer is modified using a first order numerical equation

and c) analog first order hold (FOH) where the step-function output of the digital computer, after a zero order hold (ZOH), is transformed into ramps via analog hardware before being integrated by the analog computer. In this thesis, the term "analog compensation" is used to identify method a) above.

Gilbert[3] suggested the input to the digital computer be modified using its derivatives. This method is effectively very similar to analog compensation. He also applied Z-transform theory and used it to determine the effects of sampling and time delay on accuracy. Other authors[7,9,16,17,18] have used Z-transform theory to predict dynamic errors analytically.

Papers by Matlock[4] and by Deiters and Nomura[5] extended the method of digital prediction. They derived higher order numerical equations to predict inputs to the integrators more precisely.

The analog compensation scheme was also suggested by Karplus[6] who presented an extensive analysis of several sources of error in hybrid computation, such as errors due to sampling.

Vichnevetsky[7] analyzed the analog compensation method as well as first and second order digital prediction. He derived equivalent difference equations for the hybrid loop with and

without these methods of compensation. These equations apply to a system of linear differential equations of any order and to the hybrid computer case where only the integrations are done on the analog computer, the rest of the operations being performed on the digital side. Vichnevetsky also applied Z-transforms to establish stability contours in system parameter space. This method of determining stability contours has been used by other authors,[15,16,19] however, it does not give accuracy information.

A paper by Curry[8] which is closely related to this thesis, gives experimental results with different methods of correction. He compares his results with theoretical results derived by Howe[9] using Z-transforms. The methods of compensation used in his experiments were: (1) first and second order digital compensation, (2) analog FOH, and (3) a method called "kickoff" where the knowledge of the solution over a period corresponding to the first three frame-times is used to start the solution. Curry concludes that the complexity of both producing an analog FOH circuit and programming for kickoff are not worth the relatively insignificant gain in accuracy. The aim of Curry's paper is to make available to the hybrid programmer the results of his experiments with a second order linear differential equation (SOLDE) and a specific method of compensation. He is one of

the first to provide experimental results in readily accessible chart form. From the charts, the programmer can obtain an error estimate for a SDLDE problem with a certain frequency and damping ratio knowing the hybrid sampling frequency.

A paper by Deiters, Kano and Inoue[10] is one of the few other papers providing experimental results. Their experiments include analog compensation and other methods. For analog compensation, they use the best compensation parameter value in the Taylor Series sense.

Most of the accuracy and stability studies were for linear systems since nonlinear systems cannot generally be solved analytically. The requirement that a sufficiently large number of samples per cycle of solution be maintained for stability and accuracy in hybrid implementations is akin to a similar requirement in purely digital methods.

The ultimate goal of the research reported here is to develop a software package for modeling continuous systems in real time on a hybrid computer such that the user will not be required to get involved with analog computer programming. This will be achieved initially by utilizing the analog computer as a set of general purpose integrators with compensation for time delay in a standard patching arrangement. This may be viewed as a hybrid computer

continuous simulation package where the analog subsystem will be called by an integration subroutine. Also, the compensation parameters may be optimized automatically to minimize solution errors for a given set of model parameters and time delay.

In order to produce this package, a compensation method is required. The analog compensation method of Miura and Iwata will be adopted for this package, initially, since it requires a simple hardware arrangement on the analog side and does not increase the digital computation time delay in the hybrid loop. Also, Vichnevetsky has shown that stability is better with this method than first order digital compensation.

Four different but equivalent mathematical models are defined in the sense that each one generates the solution of a second-order linear differential equation with constant coefficients or a close approximation thereof. They are also implemented on computers as computerized models and solutions are compared. One of the mathematical models is the sampled-data model implemented on a hybrid computer. Another is the discrete time equivalent to sampled-data (DTESD) model implemented on a digital computer. It is a theoretical representation of the hybrid computer model for which Vichnevetsky produced stability contours and Deiters,[10] accuracy curves. A third mathematical model is the continuous

time model programmed on an analog computer. It is interesting to investigate the region of accuracy of this model since it represents, in a sense, the hybrid computer model with no digital computing delay. It will be seen that it does not always produce the perfect solution. The discrete time analytical model is the fourth mathematical model. It generates analytical solutions of the SOLDE asynchronously on the digital computer. Its solutions are used as references for accuracy analyses of the three other models.

As it is the objective of this thesis to produce accuracy results for these computerized models, a normalized root mean square (RMS) error criterion is adopted here. It gives a unique value for the difference between two solutions and a feel for its significance can be developed by examining certain well-chosen cases.

The input parameters to be varied and their range of values for acceptable accuracies of solution are also established. In each of the computerized models, the two parameters of the SOLDE, damping ratio and natural frequency, are varied. In addition, the sampled-data model and its discrete time equivalent involve two more parameters, namely, time-delay and compensation factor.

A set of digital programs has been implemented to facilitate the anlyses of accuracy. These programs solve the different computerized models after accepting their input data and they compute the RMS errors between solutions. The errors are then compiled into tables and displayed on graphs for predefinable sets of values of input parameters. The programs are general in the sense that they can perform the error analysis for any additional computerized model with the same parameters or they can easily be modified to handle computerized models involving different parameters.

Useful data are produced and displayed by these programs concerning the accuracy of the first three computerized models relative to the fourth (i.e., the analytical solution) as well as to each other. These data are collected and arranged in such a way that it is relatively easy to estimate the errors for arbitrary values of the parameters within specified ranges.

A more detailed overview is given in Section 2.0. This is followed in Section 3.0 by an explanation of the hybrid loop time delay and the theory of compensatipn for it. Sections 4.0 and 5.0 respectively describe the mathematical models and their documented implementation. Performance criteria to determine the accuracy of one model compared to another are discussed in Section 6.0. The performance

analysis computer programs are described in Section 7.0 with reference to their detailed documentation in Appendices C to J. Section 8.0 discusses the accuracy of the computerized models with the aid of results documented in Appendix B. Overall conclusions are formulated in Section 9.0.

2.0 OVERVIEW

A computerized model solution occurs in model time whereas reality occurs in real time.[11] Here, real time is represented by variable t and model time by $t' = \beta t$ where $\beta = f(t)$. If $f(t)=1$ then the model is solved in real time ($t=t'$). If, however, $f(t)$ is not equal to 1 but is known (i.e., model time is slowed down or speeded up by a known function relative to real time), then the model is 'synchronous' whereas if $f(t)$ is not known, the model is 'asynchronous'. Analog computer models and hybrid computer models are generally synchronous. Digital computer models are usually asynchronous but they may be synchronous. This work and its results are aimed at synchronous hybrid computer models.

A computerized model is defined to be a computer program which implements a mathematical model. An input-process-output diagram of four computerized models is shown in Figure 1. The four mathematical models in this figure can generate good approximations to the solution of a linear second-order

differential equation with step input for sets of parameter values defined by this investigation. The dashed arrows between the mathematical model and computer program blocks indicate that the mathematical models are implemented in the computer programs to produce 'computerized models'. Every computerized model is identified as a system. System A is a discrete-time analytical solution implemented on a digital computer. System B is a continuous-time model which is solved by an analog computer. System C is a sampled-data model implemented on a hybrid computer and System D is an asynchronous digital model equivalent to the hybrid implementation. Each system accepts a set of input data (model parameters) gathered in data sets DSA and DSB. Furthermore, each computerized model produces a set of response data which are stored in data sets DSRA to DSRD, respectively. Response data are discrete values of system solutions.

The purpose of developing these computerized models is to compare their solutions and analyse their performance. This is achieved by a set of three digital computer programs. Program RESPLT (Fig. 2) solves the user specified system (S) and plots the solution as a function of time. The model input parameters are also specified interactively as data sets DSAU and DSBU. A similar program, called RESPONS. (Fig. 3) also

solves a user specified system (S) but for various combinations of input parameter values included in data sets DSA and DSB. The solutions are stored as discrete data points into data set DSRA or DSRB depending on whether System A or System B is being solved. Subsets DSRC1 to DSRC5 or DSRD1 to DSRD5 are used for storage if System C or System D is being solved. These data sets are used as inputs to the third program, called RMSERR (Fig. 4). A root mean square (RMS) error is computed between solutions of two systems (S_1, S_2) which are specified interactively. Finally, program RMSERR displays tables and graphs of the RMS errors. Graphs are also produced to display the range of model input parameters for which the errors in the solutions are acceptable.

3.0 COMPENSATION THEORY

In this section, the different components of the hybrid loop contributing to the time delay are summarized. It is shown how this delay affects the hybrid solutions and how it can be compensated from the point of view of Taylor Series expansions. Finally, the relationship between theory, digital correction and analog compensation is established.

A source of error inherent in sampled-data systems is the sample and hold process whereby the digital computer samples the analog output at discrete time intervals and holds the analog input constant during this interval. A second error,

known as digital computation error or transport lag, is due to the computational delay between sampling and updating of the analog computer. The cycle of calculations, or hybrid loop, is shown in Figure 5 illustrating the associated component delays. The total delay T_C, which will be refered to in the sequel as the digital time of computation, is given by

$$T_C = \tau_1 + \tau_2 + \tau_3 \qquad (1)$$

where τ_1 is the analog to digital (A/D) conversion time, τ_2 is the digital calculation time and τ_3 is the digital to analog (D/A) conversion time.

In 1963, Miura and Iwata[2] applied the Taylor Series extrapolation scheme to modify the analog integrator input. This method requires that the sample and hold delay be fixed. The previous year, Korn[12] had shown the importance of a constant time interval when using digital compensation. Thus a fixed digital computation time T_C and a fixed sampling and update interval T were specified by Miura and Iwata. They then compared the actual input W(t), a staircase function (Fig. 6(a)), with the ideal function Y(t), (Fig. 6(b)). The function W(t) is first delayed by a constant time τ. This fixed τ is the amount of time from the instant the integrators start operating with their initial conditions to the instant they receive their first input value. Usually, τ

is equal to the total digital time of computation T_C. The analog input is held constant until it is updated by the result of the next new digital computation and so on. Since the true input W(t) is shifted from the ideal input Y(t), a compensation scheme is needed which will make both functions equivalent.

Miura and Iwata proved that when the update interval T' is small, the integral of the step function W(t) is equivalent to the integral of the smooth curve shown in Figure 6(c) passing through the middle of the steps. Since this smooth curve is parallel to Y(t) with a phase shift of $\tau + T/2$, they have shown that

$$\int W(t)\,dt = \int Y(t-(\tau+T/2))\,dt \qquad (2)$$

Thus, it is seen that the input to the analog integrator should be $W(t+(\tau+T/2))$ as shown in Figure 6(d). This can be expanded for small $(\tau+T/2)$ in a Taylor Series

$$W(t+(\tau+T/2)) = W(t) + \dot{W}(t)(\tau+T/2) + \ddot{W}(t)\frac{(\tau+T/2)^2}{2!} + \ldots \qquad (3)$$

Thus, a first order digital correction would include the first two terms of (3) and higher order corrections would include higher order terms. For a first order correction, an approximation for $\dot{W}(t)$ in the interval $[iT, (i+1)T]$, $i=1,2,3,\ldots$, is the slope between the outputs of the digital

computer at sampling times (i-1)T and iT computed as:

$$W(t) = \frac{W(iT) - W((i-1)T)}{T}, \quad t \in [iT, (i+1)T] \qquad (4)$$

Replacing (4) into (3) gives a corrected input to the analog integrator and performing the integration on the first two terms of (3) for one step interval T gives the incremental input to the digital computer as:

$$X((i+1)T) - X(iT) = \int_0^T \left\{ W(iT) + \frac{[W(iT)-W((i-1)T)]}{T}(\tau+T/2) \right\} dt \qquad (5)$$

or equivalently,

$$X((i+1)T) = X(iT) + \int_0^T W(iT)\, dt + (\tau+T/2)[W(iT)-W((i-1)T)] \qquad (6)$$

keeping in mind that W(iT) and W((i-1)T) are constants in the T interval.

It can be shown that the analog compensation method is also based on a first order Taylor Series expansion. Figure 7 is a simplified diagram of the continuous operations performed by the analog computer in the hybrid loop. The diagram illustrates the equation

$$X(t) = \int W(t)\, dt + \theta W(t) \qquad (7)$$

where time functions W(t), θW(t) and X(t) are displayed in

Figure 8. The input to the analog computer is W(t) and since the constant W((i+1)T) is set before X((i+1)T) is read, the approximation for W(t) in the interval [iT, (i+1)T] is an interpolation in analog compensation rather than an extrapolation as in the digital correction case. The approximation is:

$$W(t) = \frac{W((i+1)T) - W(iT)}{T}, \quad t \in [iT, (i+1)T] \tag{8}$$

Thus, a result similar to (5) is obtained with analog compensation factor θ over the interval [iT, (i+1)T], i.e.,

$$X((i+1)T) - X(iT) = \int_0^T \left\{ W(iT) + \theta\frac{[W((i+1)T)-W(iT)]}{T} \right\} dt \tag{9}$$

and since W((i+1)T), W(iT) and θ are constants, the input to the digital computer at t=(i+1)T, is

$$X((i+1)T) = X(iT) + \int_0^T W(iT)\, dt + \theta[W((i+1)T)-W(iT)] \tag{10}$$

The best value for θ in the Taylor Series sense is thus (T+T/2) as it was for the digital prediction case.

The updating interval T is the time interval during which the analog input is held constant. But it is usually held constant during the period of conversions and digital

computations T_C. As mentioned above, τ is also usually equal to T_C which means that according to the Taylor Series, the value of θ for the best analog compensation is

$$\theta = (T_C + T_C/2) = 1.5T_C \qquad (11)$$

Indeed, in this study, both τ and T were set equal to T_C. Furthermore it was important to fix the loop time T_C accurately because of its role in the computation of θ.

As Deiters et. al.[10], the analog compensation method yields a larger stability region in the space of parameters but a smaller accuracy region than their first order digital correction method. They used the Gregory-Newton extrapolation formula in a similar way than the Taylor Series to derive the digital correction equations. However their tests were conducted for only one analog compensation case, i.e., $\theta = 1.5T_C$. This might not be the best value for every set of parameters. An objective of this thesis project was to determine ranges of parameter values where $1.5T_C$ is not the optimum value of θ.

An advantage of the analog compensation method is that it does not require additional digital programming in the hybrid loop which would increase the computation time. Furthermore, it is easy to implement on the analog computer.

4.0 MATHEMATICAL MODELS

This section describes the four mathematical models that were used in computerized models as seen in Figure 1. These models were implemented in computer programs in order that their solutions could be compared and their performance analysed. In this thesis, a mathematical model is defined to be either 1) a conceptual model or 2) a mathematical equivalent of a conceptual model or 3) a mathematical equivalent of another mathematical model. A conceptual model is a mathematical representation of reality. A mathematical equivalent is defined to be a set of equations whose solution approximates the solution of another mathematical model.

A second-order linear differential equation (SOLDE) was chosen as a continuous-time conceptual model representing a real system such as a mass-spring-damper system and was solved in synchronous time on an analog computer.

A discrete-time analytical model consisting of equations which generate the solution of the SOLDE was solved asynchronously for very small time intervals on a digital computer. Its solution was the exact solution of the SOLDE used as reference to determine the solution errors of computerized mathematical equivalents.

A third mathematical model was a sampled-data model (i.e., a combination of a continuous-time and discrete-time sub-models) which was implemented on a hybrid computer in such a way that all computations were performed in the digital computer except for compensated integrations which were done in the analog computer. A main objective was to produce results comparing its solutions with those of other models.

The fourth mathematical model was a discrete-time equivalent to the sampled-data (DTESD) model consisting of a set of difference equations. The DTESD model was solved asynchronously on a digital computer to determine the range of parameter values over which equivalence with the hybrid model implementation could be ascertained. With this comparison, the assumptions of perfect equivalence made by Deiters et. al.[10] could be verified. Results were then obtained for extrapolated parameter values to predict the accuracy of hybrid implementations in parameter ranges exceeding the capabilities of the available hybrid facility (e.g., $T_c \lesssim 5$ ms).

4.1 Continuous-Time

The continuous-time model is the conceptual model represented by a second-order linear constant-coefficient differential equation with step input:

$$\ddot{x}(t) + 2\zeta\omega_n^* \dot{x}(t) + \omega_n^{*2} x(t) = \omega_n^{*2} u(t) \qquad (12a)$$

$$\zeta \geq 0 \tag{12b}$$

$$\omega_n^* > 0 \tag{12c}$$

where t is clock time identified as real time, parameters ζ and ω_n^* respectively are the damping ratio and natural undamped frequency of the system and u(t) is the unit step function occurring at time t=0. This second-order system can be solved in real time but it can also be solved in synchronous time by defining a time-scaled equivalent

$$\ddot{x}(t') + 2\zeta\omega_n \dot{x}(t') + \omega_n^2 x(t') = \omega_n^2 u(t') \tag{13}$$

where

$$t' = \beta t \tag{14}$$

is model time and

$$\omega_n = \omega_n^* / \beta \tag{15a}$$

$$\beta > 0 \tag{15b}$$

is natural undamped frequency in model time, and β is a time-scale factor.

A variation in parameter value β results in a variation in solution speed of the system.[11] Precisely, when:

$\beta = 1$, solution is in real time (i.e. corresponds to solution of system (12)),
$\beta > 1$, solution is slower than real time,
$0 < \beta < 1$, solution is faster than real time.

The conversion of system (12) from real-time to model-time can be visualized in the diagram of Figure 9 where it is seen that

$$\dot{x}(t) = \frac{dx(t)}{dt} = \beta \frac{dx(t')}{dt'} = \beta \dot{x}(t') \qquad (16)$$

Likewise,

$$\ddot{x}(t) = \beta^2 \ddot{x}(t') \qquad (17)$$

Replacing equations (16) and (17) into equation (12) leads to:

$$\beta^2 \ddot{x}(t') + 2\zeta\omega_n^*\beta\dot{x}(t') + \omega_n^{*2} x(t') = \omega_n^{*2} u(t') \qquad (18)$$

Dividing by β^2 and using (15) gives system (13).

Since factor β affects only the time scale of the solution, equation (13) can be used to study the system represented by equation (12).

4.2 Discrete-Time Analytical Solution

The analytical solution x(t) of system (12) is computed at discrete time t^i, where i=0, 1, 2, ..., starting with $t^0=0$. Let the response data point $x(t^i)$ simply be x^i. The equations of the solution and its derivative depend on damping ratio ζ as follows:[13] If $0<\zeta<1$, the response is oscillatory:

$$x^i = u(t^i) - \frac{e^{-\zeta \omega_n^* t^i}}{\sqrt{1-\zeta^2}} \sin(\omega_n^* \sqrt{1-\zeta^2}\, t^i + \Phi) \qquad (19)$$

$$\dot{x}^i = \omega_n^* e^{-\zeta \omega_n^* t^i} \left[\frac{\zeta \sin(\omega_n^* \sqrt{1-\zeta^2}\, t^i + \Phi)}{\sqrt{1-\zeta^2}} - \cos(\omega_n^* \sqrt{1-\zeta^2}\, t^i + \Phi) \right] \qquad (20)$$

where

$$\tan \Phi = \sqrt{1-\zeta^2} / \zeta \qquad (21)$$

If $\zeta>1$, the response is nonoscillatory:

$$x^i = u(t^i) + ce^{-at^i} + de^{-bt^i} \qquad (22)$$

$$\dot{x}^i = -ace^{-at^i} - bde^{-bt^i} \qquad (23)$$

where

$$a = \omega_n^* (\zeta - \sqrt{\zeta^2 - 1}) \tag{24}$$

$$b = \omega_n^* (\zeta + \sqrt{\zeta^2 - 1}) \tag{25}$$

$$c = -(\zeta + \sqrt{\zeta^2 - 1}) / 2\sqrt{\zeta^2 - 1} \tag{26}$$

$$d = -(-\zeta + \sqrt{\zeta^2 - 1}) / 2\sqrt{\zeta^2 - 1} \tag{27}$$

If $\zeta = 1$, the system is critically damped and the response is nonoscillatory:

$$x^i = u(t^i) - e^{-\omega_n^* t^i} (1 + \omega_n^* t^i) \tag{28}$$

$$\dot{x}^i = \omega_n^{*2} e^{-\omega_n^* t^i} \tag{29}$$

4.3 Sampled-Data

The sampled-data model consists of the three subsystems shown in Figure 10 : discrete-time, sample and hold, and continuous-time. Algebraic computations are performed in the discrete-time subsystem at every constant time interval of length T_C. This time interval includes the computation time in the discrete subsystem and the data conversion time between continuous and discrete subsystems. As the output $X(t')$ of the continuous subsystem is sampled at times iT_C, i.e.,

$$X^i = X(t'^i) \tag{30}$$

to become the input to the discrete subsystem, the output of the discrete subsystem becomes the input to the continuous subsystem at times $(i+1)T_C$, i.e.,

$$W(t') = W^i, \quad t' \in [t'^i, t'^{i+1}) \tag{31}$$

$$t'^{i+1} = t'^i + T_C, \quad i=0,1,2,\ldots \tag{32}$$

The input $W(t')$ is a staircase function remaining constant over each sampling period.

The continuous-time subsystem performs the integrations on its input with respect to time and also compensates for the time delay T_C as discussed in Section 3. Its operations are illustrated in Figure 7 representing a standard configuration of a set of n integrators and n compensators. Thus, variables W, Z, V and X in the figure are n-vectors and θ is an nxn diagonal matrix. The equations describing these operations are the following

$$\dot{Z}(t') = W(t') \tag{33}$$

$$V(t') = \theta\, W(t') \tag{34}$$

$$X(t') = Z(t') + V(t') \tag{35}$$

where Z is the integration output, V is the compensation vector and θ is the matrix of compensation factors.

The operations of the discrete-time subsystem consists in the calculation of a discrete n-order linear equation expressed in vector notation as

$$W^{i+1} = AX^{i} + BF^{i} \tag{36}$$

where A and B are nxn coefficients matrices and F is the forcing function vector. The special case of the second-order linear differential equation (13) may be expressed in the form of equation (36) by

$$X = (x_1 \quad x_2)^T \tag{37a}$$

$$W = (\omega_1 \quad \omega_2)^T \tag{37b}$$

$$A = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \tag{37c}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & \omega_n^2 \end{bmatrix} \tag{37d}$$

$$F = (0 \quad u)^T \tag{37e}$$

$$x_1 = x \tag{37f}$$

$$x_2 = \omega_1 = \dot{x} \tag{37g}$$

$$\omega_2 = \ddot{x} \tag{37h}$$

or

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}^{i+1} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{i} + \begin{bmatrix} 0 & 0 \\ 0 & \omega_n^2 \end{bmatrix} \begin{bmatrix} 0 \\ u \end{bmatrix}^{i} \tag{38}$$

In scalar notation, this is:

$$w_1^{i+1} = x_2^i \tag{39}$$

and

$$w_2^{i+1} = -\omega_n^2 x_1^i - 2\zeta\omega_n x_2^i + \omega_n^2 u^i \tag{40}$$

4.4 Discrete-Time Equivalent Of Sampled-Data

The sampled-data model can be represented by a discrete-time model since the operations in its discrete-time subsystem are discrete and the input to its continuous-time subsystem are constant over fixed intervals permitting the analytical computation of the latter's outputs. The objective is to develop a difference equation for the solution X(t) representing the operations of the sampled-data model.

Function X(t) is being discretized at every sampling instant $t^i = iT_c$, i=0, 1, 2, ... Thus equation (35) leads to the following equation for the difference of two consecutive discretized value of X(t).

$$X^{i+1} - X^i = (Z^{i+1} - Z^i) + (V^{i+1} - V^i) \tag{41}$$

Similarly, from equation (34) and since θ is constant, the difference of two consecutive discretized value of

the compensation function V(t) is

$$V^{i+1} - V^{i} = \theta(W^{i+1} - W^{i}) \tag{42}$$

The discrete-time difference of the integrator output Z(t) is obtained by integrating equation (33) over the sampling interval T :

$$Z^{i+1} - Z^{i} = \int_{0}^{T_C} W^{i}\, dt = W^{i}\, T_C \tag{43}$$

where W^{i} is constant over the interval (t^{i}, t^{i+1}).

Substitution of equations (42) and (43) into (41) yields:

$$X^{i+1} - X^{i} = T_C W^{i} + \theta(W^{i+1} - W^{i}) \tag{44}$$

and substituting for W^{i} from equation (36) gives

$$X^{i+1} = X^{i} + T_C(AX^{i-1} + BF^{i-1}) + \theta(AX^{i} + BF^{i}) - \theta(AX^{i-1} + BF^{i-1}) \tag{45}$$

or

$$X^{i+1} = (I + \theta A)X^{i} + (T_C - \theta)AX^{i-1} + \theta BF^{i} + (T_C - \theta)BF^{i-1} \tag{46}$$

a matrix equation which is a discrete-time equivalent to the sampled-data model.

The corresponding second-order model can be derived from equation (46) as follows. Let matrices A, B and F be as in equations (37c) to (37e). Thus equation (46) becomes:

$$\begin{bmatrix} x \\ \dot{x} \end{bmatrix}^{i+1} = \begin{bmatrix} 1 & 0 \\ -\theta\omega_n^{*2} & 1-2\theta\zeta\omega_n^{*} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}^{i} + (T_c - \theta) \begin{bmatrix} 0 & 1 \\ -\omega_n^{*2} & -2\zeta\omega_n^{*} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}^{i-1}$$

$$+ \theta \begin{bmatrix} 0 & 0 \\ 0 & \omega_n^{*2} \end{bmatrix} \begin{bmatrix} 0 \\ u \end{bmatrix}^{i} + (T_c - \theta) \begin{bmatrix} 0 & 0 \\ 0 & \omega_n^{*2} \end{bmatrix} \begin{bmatrix} 0 \\ u \end{bmatrix}^{i-1} \qquad (47)$$

In scalar notation, this is :

$$x^{i+1} = x^{i} + \theta\dot{x}^{i} + (T_c - \theta)x^{i-1} \qquad (48)$$

and

$$\dot{x}^{i+1} = -\theta\omega_n^{*2}x^{i} + (1 - 2\theta\zeta\omega_n^{*})\dot{x}^{i}$$

$$+ (T_c - \theta)(-\omega_n^{*2}x^{i-1} - 2\zeta\omega_n^{*}\dot{x}^{i-1})$$

$$+ \theta\omega_n^{*2}u^{i} + (T_c - \theta)\omega_n^{*2}u^{i-1} \qquad (49)$$

5.0 COMPUTERIZED MODELS

A computerized model is a computer program which implements a mathematical model on a particular computer. The four computerized models are identified as Systems A, B, C and D in Figure 1. This section describes the computer implementation of the mathematical models of Section 4.

5.1 System A

The discrete-time analytical solution mathematical model is implemented on a VAX 11/780 digital computer. This computerized model, System A, is implemented as FORTRAN subroutine SYSTMA which uses data set DSA as input, solves the analytical equations, and produces data set DSR as output. Its documentation is given in Appendix C.

Subroutine SYSTMA starts by establishing a final solution time T_F and a data computation/storage period T_S. These variables are computed as:

$$T_F = 2\pi N_T / \omega_n^* \tag{50}$$

and

$$T_S = 2\pi / N_S \omega_n^* \tag{51}$$

where N_T is the required number of cycles for the whole solution and N_S is the required number of data points per cycle to be computed and stored. The remaining operations of SYSTMA depend on whether the damping ratio is less than, equal to, or greater than unity to solve the appropriate equations among (19) to (29) iteratively. Data points x', $\dot{x}'$ and t' where t' is incremented by T_S at each iteration, are stored in data set DSR.

5.2 System B

The continuous-time model is implemented on an EAI 680 analog computer. The analog computer is an electronic computing machine whose operation is quite different from that of a digital computer. Numerical values on an analog computer are established by the measurement of electrical voltages on which mathematical operations are performed. An analog computer operates in a parallel manner wheras a digital computer operates sequentially. Parallel operation means that all variables in a problem are simultaneously updated as time evolves. Consequently, the time required to generate problem solutions on an analog computer is independent of problem complexity. Also, an analog computer solves differential equations synchronously. Furthermore, in this investigation every solution of System B is done in real time ($\beta=1$).

An analog computer is made of electronic components such as integrators, summers, etc. It is programmed by linking these components together with electrical connections. Furthermore, the analog computer operates with values smaller than or equal to 1 at all times. For that purpose, the model's time-dependent variables must be scaled with 'amplitude scaling factors' to prevent

'overloading' the analog components. Figure 11 illustrates a scaled analog circuit for System (13) where, x_M, $\dot{x}_M$ and $\ddot{x}_M$ are the scale factors corresponding to variables x, $\dot{x}$ and $\ddot{x}$, respectively. x^0 and $\dot{x}^0$ are the initial conditions of the system. On the circuit diagram, the symbol numbered 16 represents a summer whereas symbols 20 and 25 represent integrators. A DCA (Digitally Controlled Attenuator), or a potentiometer, has the function of multiplying its own constant coefficient by its input value. DCA's are represented by the circular symbols numbered 15, 20, 22, 23, 25, 27 and 28. The expressions in square brackets are scaled variables whereas the curved brackets indicate DCA coefficients. Since the maximum value of a DCA coefficient is also unity, it is sometimes necessary to introduce an 'amplifier gain' G. Such a gain is shown associated with integrators 20 and 25, allowing the coefficient of DCA's 22 and 27 to be divided by the same factor.

When the analog computer is in operate mode, that is when the model is being solved, variables $\dot{x}$ and $\ddot{x}$ are integrated. Also, variables x and $\dot{x}$, multiplied respectively by the coefficients of DCA's 28 and 23, are added to the forcing term from DCA 15 to produce a new

value for $\ddot{x}$. Each of these steps is executed continuously.

The analog program of System B is controlled by digital computer subroutine SYSTMB whose documentation is contained in Appendix D. There are two main parts in the subroutine. The first one sets the analog computer components; the second stores data (Fig. 12).

The first part of SYSTMB loads program VAXHYB on the PACER 100 digital computer. This program is part of the VAX/HYBRID system and provides for data transfer between the VAX digital computer and the analog computer through the PACER digital computer. SYSTMB then computes the final solution time T_f and the data storage sampling period T_S by equations (50) and (51). It also computes the following scale factors:

$$x_M = 2\,u \tag{52}$$

$$\dot{x}_M = u\,\omega_r \tag{53}$$

$$\ddot{x}_M = u\,\omega_r^2 \tag{54}$$

The amplifier gain G is determined depending on the natural undamped frequency ω_r as follows:

$$G = \begin{cases} 1, & 0 < \omega_r \leq 1 \\ 10, & 1 < \omega_r \leq 10 \\ 100, & 10 < \omega_r \leq 100 \end{cases} \tag{55}$$

Gain G is then set on the integrators by analog mode control using VAX/HYBRID system subroutines (Q Routines). The first part of SYSTMB concludes by computing the DCA values D_K, where K is the DCA number, and sets them on the analog computer. DCA values are computed as follows:

$$D_{15} = u\,\omega_n^2 / 10\,\ddot{x}_M \tag{56}$$

$$D_{20} = x^0 / \dot{x}_M = 0 \tag{57}$$

$$D_{25} = x^0 / x_M = 0 \tag{58}$$

$$D_{22} = \ddot{x}_M / G\,\dot{x}_M \tag{59}$$

$$D_{27} = \ddot{x}_M / G\,x_M \tag{60}$$

$$D_{23} = -2\zeta\omega_n \dot{x}_M / 10\,\ddot{x}_M \tag{61}$$

$$D_{28} = -\omega_n^2\, x_M / 10\,\ddot{x}_M \tag{62}$$

The second part of subroutine SYSTMB is performed by calling subroutine HILOOPB. It puts the analog computer in operate mode and reads variables x_S and $\dot{x}_S$ respectively at a high rate from the analog-to-digital (A/D) converters, unscaling them each time by

$$x = x_S\, x_M \tag{63}$$

and

$$x = x_S \, x_M \tag{64}$$

An accurate clock on the VAX computer determines the time t^J when variables are read from D/A converters. Solution data points are stored in data set DSR as x, x' and t' when t^J is larger than or equal to the last time of storage t^{-1} plus the data storage sampling period T_S. The solution stops when

$$t^J \geq T_F \tag{65}$$

The iterative logic is shown is Figure D-5.

5.3 System C

The sampled-data model is implememted on the hybrid computer as System C in Figure 1. The continuous-time subsystem is implemented on the EAI 680 analog computer while the discrete-time subsystem depends upon two digital computers: the DEC VAX 11/780 and the EAI PACER 100 (Fig. 13). System C is controlled by subroutine SYSTMC on the VAX computer. Program PACMAN (PACER MAIN) on the PACER provides for data transfer between SYSTMC and the analog computer as well as reading the analog computer clock. Data conversion between the PACER and the analog computer is performed by A/D and D/A converters in the sample and hold subsystem. The

documentation for SYSTMC and PACMAN can be found in Appendices E and G respectively.

The circuit diagram of the continuous-time subsystem is illustrated in Figure 14. The symbolic figures in this diagram have the same meaning as in the analog circuit of Figure 11. For each integration, this subsystem continuously integrates the D/A converter values, implements analog compensation by adding a fraction θ therof to the integrator output and delivers the sum to an A/D converters.

Subroutine SYSTMC is implemented in such a way that with minor changes and the standard analog patchboard, it would allow up to fifteen integrators with compensation to be used simultaneously. This would permit, for example, the performance study of other mathematical models such as higher order linear or nonlinear systems or combinations therof. Furthermore, with relays R_{34} and R_{39} positioned as shown in Figure 14, the integrators operate independently. Suppose that R_{34} would be switched to its alternative position, then the integrators would be connected in series. Different compensation factors would probably be required in this case. Such alternatives bear investigation but are not considered here.

Since the hybrid system exchanges data between the VAX computer and the analog computer, the hybrid facility used was such that a program similar to program VAXHYB for System B was required on the PACER computer to allow these transfers to occur. As a replacement for the general-purpose program VAXHYB, program PACMAN was implemented for the particular application of the computerized model of System C. This program has two main functions and performs no important calculations. The first one is to receive data from the VAX into a buffer, set the analog components and D/A converters, read A/D converters and send back the buffer containing new values to the VAX. The second is to fix the required loop-time T_C by looping on one if-statement until the elapsed time since the last reading of D/A converters does not exceed T_C.

Transfer program VAXHYB could have been used for System C as it had been for System B. However, SYSTMC was done before SYSTMB and the accurate clock of the VAX was not available then. Thus, PACMAN remains as a special-purpose alternative to VAXHYB with the advantage that looping to fix computation time T_C is done on the dedicated PACER rather than the multi-user VAX computer, thereby liberating the latter for other users.

The operations of SYSTMC are now summarized. First, the final solution time and data storage period are computed using the same equations as in Systems A and B, i.e., equations (50) and (51). Amplitude scale factors are computed by:

$$\ddot{x}_M = (1.2)\ 2\ u \tag{66}$$

$$\dot{x}_M = (1.2)\ u\ \omega_n \tag{67}$$

$$x_M = (1.2)\ u\ \omega_n^2 \tag{68}$$

The scale factors are set larger than in System B since many solutions tend to be less accurate with System C. The scale factors are adjusted further, as follows, if the coefficients of compensation DCA's (32 and 37 in Figure 14) exceed unity:

$$\dot{x}_M = \begin{cases} 1.2\ \theta\ \dot{x}_{MP}, & \theta\ \dot{x}_{MP} / \dot{x}_M \geq 1 \\ \dot{x}_{MP}, & \theta\ \dot{x}_{MP} / \dot{x}_M < 1 \end{cases} \tag{69}$$

$$\ddot{x}_M = \begin{cases} 1.2\ \theta\ \ddot{x}_{MP}, & \theta\ \ddot{x}_{MP} / \ddot{x}_M \geq 1 \\ \ddot{x}_{MP}, & \theta\ \ddot{x}_{MP} / \ddot{x}_M < 1 \end{cases} \tag{70}$$

where $\dot{x}_{MP}$ and $\ddot{x}_{MP}$ respectively are the past values of $\dot{x}_M$ and $\ddot{x}_M$. Then, the amplifier gain G is given a value such that DCA's 31 and 36 do not exceed unity. This is

achieved by first determining the maximum value of G, i.e.,

$$G_M = MAX\left(x_M / x_M , \; \dot{x}_M / \dot{x}_M \right) \tag{71}$$

Gain G is then set depending on G_M as follows:

$$G = \begin{cases} 1 , & G_M \leq 1 \\ 10 , & 1 < G_M \leq 10 \\ 100 , & 10 < G_M \leq 100 \\ 1000 , & 100 < G_M \leq 1000 \end{cases} \tag{72}$$

Relay variables R_{34} and R_{39}, which are used to define the analog circuit, are given values of zero such that the relays will remain open as shown in Figure 14 for the duration of the solution. The DCA values D_K, where K represents the DCA number, are computed as follows:

$$D_{30} = \ddot{x}^0 / x_M = 0 \tag{73}$$

$$D_{35} = \dot{x}^0 / x_M = 0 \tag{74}$$

$$D_{32} = \theta \; \dot{x}_M / x_M \tag{75}$$

$$D_{37} = \theta \; \dot{x}_M / \dot{x}_M \tag{76}$$

$$D_{31} = -\dot{x}_M / x_M \; G \tag{77}$$

$$D_{36} = -\dot{x}_M / \dot{x}_M \; G \tag{78}$$

In order to set up the analog computer for a particular solution, program PACMAN needs the necessary data. As mentioned above, the data are transmitted from the VAX to the PACER through a buffer. At this point of SYSTMC, the buffer is filled with the values of the following variables: D_{30}, D_{35}, D_{32}, D_{37}, D_{31}, D_{36}, G, T_C, R_{34}, R_{39} and an integer value specifying the number of integrations. The buffer data are received from the VAX as soon as the execution of program PACMAN begins on the PACER computer.

Then the solution of System C begins with the execution of subroutine HILOOP called by SYSTMC. It is solved in a high speed iterative loop which includes the transfer of hybrid data (VAX-Analog-VAX through PACER). The high-speed-loop includes the following operations (Figure E-4):

(1) The A/D converter variables x_S and $\dot{x}_S$ are received in the buffer from the PACER and unscaled:

$$x = x_S x_M \tag{79}$$

$$\dot{x} = \dot{x}_S \dot{x}_M \tag{80}$$

(2) The discrete time solution (t^J, x^J, $\dot{x}^J$) is stored in data set DSR as discrete data (t^I, x^I, $\dot{x}^I$) in a similar manner as SYSTMB (Fig. E-6), where t^J and t^I

are the solution and data storage times respectively incremented by

$$t^{J} = t^{J-1} + T_C \tag{81}$$

and

$$t^{i} = t^{i-1} + T_S \tag{82}$$

(3) Model equation (13) is solved for x and variables x and x are scaled by

$$x_S = x / x_M \tag{83}$$

and

$$x_S = x / x_M \tag{84}$$

These variables are sent to the D/A converters via the PACER.

The VAX/PACER interface system is such that data transfered in either direction must be integers. However, most of the hybrid data are fractions. To overcome this problem, the fractions are multiplied by the largest PACER integer (i.e., 32768) and then converted into integers by the FORTRAN function ININT. For example, variables x_S and x_S are transformed by the following FORTRAN statements before being sent to the PACER:

```
XDS  = ININT(XDS*32768.)                    (85)

XDDS = ININT(XDDS*32768.)                   (86)
```

The inverse transformation (i.e., division by 32768) is then performed by subroutine VXPBUF of program PACMAN to revert back to scaled fractions for D/A conversion purposes. The same idea applies for PACER to VAX transfers. In the PACER, fractions x_S and $\dot{x}_S$ are transformed into integers by subroutine PVXBW and inverse transformed in the VAX by:

```
XS  = FLOATI(XS)/32768.                     (87)

XDS = FLOATI(XDS)/32768.                    (88)
```

where the FORTRAN function FLOATI converts integer variables into real variables.

System C uses a FLAG variable with six possible values to control whether the program should stop or continue.

FLAG = 1 :

This is the normal value indicating that the solution must continue.

FLAG = 2 :

Indicates that the solution must stop because either the final solution time is reached or the solution is growing indefinitly large.

FLAG = 3 :

Indicates a time constraint. It means that the required time of computation T_C is too small and the operations in the high-speed-loop could not be acheived completely during that period.

FLAG = 4 :

Indicates that for some reason, the VAX was unable to send the buffer data to the PACER. This case and the case of FLAG=5 should never happen unless the hybrid system is not functioning properly.

FLAG = 5 :

Indicates that for some reason, the VAX was unable to receive the buffer data from the PACER (see FLAG=4).

FLAG = 6 :

Indicates that an overload on the analog computer is about to be or has already been detected. The data going to the D/A converters and coming from the A/D converters are checked to prevent them from getting too close to unity. If an overload is detected, the amplitude scale factors are multiplied by 1.5 and the high-speed-loop is restarted.

5.4 System D

The discrete-time equivalent of sampled-data computerized model is implemented digitally by FORTRAN subroutine SYSTMD whose documentation appears in Appendix F. Subroutine SYSTMD uses the model parameters of DSA and DSB as input and solves equations (48) and (49). It first computes a final solution time T_F (eq. 50) and a

data storage sampling period T_S (eq. 51). As in System C, the equations are solved at every instant t^J and incremented as in equation (81) and data corresponding to instants t^I (eq. 82) are stored in data set DSR. The solution stops when $t^J \geq T_F$.

6.0 PERFORMANCE CRITERIA

Solution errors are defined as departures from a reference solution. Several criteria such as amplitude error, gain error and root mean square error have been used in the literature to establish the accuracy of dynamic system solutions.

The criterion adopted in this report is a normalized root mean square error of a normal distance approximation. The overall error between a given function x(t) and a reference function $x_R(t)$ (Fig. 15a) is determined as follows. At any given value of time t where both functions exist, the normal distance is approximated by

$$\delta = \Delta x \cos \alpha \tag{89}$$

where

$$\Delta x = x - x_R \tag{90}$$

is the difference in amplitude and

$$\alpha = \tan^{-1} x_R \tag{91}$$

is the slope angle of x_R. Furthermore, defining line L to be parallel to the tangent L_R of x_R and intersecting x at the given value of t, the derivative may be expressed as

$$\dot{x}_R = \Delta x / \Delta t \tag{92}$$

where

$$\Delta t = t - t_1 \tag{93}$$

is the difference in time between the given value of t and t_1, the time corresponding to the horizontal projection of $x_R(t)$ onto L.

The distance δ is normalized by normalizing the differences Δx and Δt respectively to the maximum absolute amplitude x_{MAX} and the time domain T_1 of the functions. Specifically,

$$x_{MAX} = MAX\left[MAX(|x(t)|), MAX(|x_R(t)|)\right], \quad t \in [0, T_1] \tag{94}$$

where T_1 is the minimum of both total solution times. In order to be consistent, every model was solved for a fixed period corresponding to N_T=3 cycles of undamped natural frequency $F_n = \omega_n / 2\pi$ except when a system was severely unstable. Thus the normalized differences are computed by

$$\Delta x_N = \Delta x / x_{MAX} \tag{95}$$

and

$$\Delta t_N = \Delta t \; / \; T_1 \tag{96}$$

and the normalized derivative is

$$\dot{x}_{RN} = \Delta x_N \Big/ \Delta t_N = \Delta x \; T_1 \Big/ \Delta t \; x_{MAX} = \dot{x}_R \; T_1 \Big/ x_{MAX} \tag{97}$$

The normalization of Figure 15a to 15b is completed by:

$$\alpha_N = \tan^{-1} x_{RN} \tag{98}$$

$$\delta_N = \Delta x_N \cos \alpha_N \tag{99}$$

$$x_{RN} = x_R \Big/ x_{MAX} \tag{100}$$

$$x_N = x \Big/ x_{MAX} \tag{101}$$

$$L_{RN} = L_R \Big/ x_{MAX} \tag{102}$$

$$L_N = L \Big/ x_{MAX} \tag{103}$$

$$t_{1N} = t_1 \Big/ T_1 \tag{104}$$

and

$$t_N = t \Big/ T_1 \tag{105}$$

Finally, the root mean square error criterion is given by:

$$\varepsilon_N = \left[\frac{1}{T_1} \int_0^{T_1} \delta_N^2 \, dt \right]^{1/2} = \left[\int_0^1 \delta_N^2 \, dt_N \right]^{1/2} \quad (106)$$

In the case when one of the solutions grows indefinitely, ε_N is given a large value of unity since the borderline of practical acceptability will be in the vicinity of 0.0075.

This error value (ε_N) represents a normalized approximation to the average normal distance between two solutions. In order to gain insight into the significance of ε_N, a set of graphs displaying errors between various SOLDE solutions is presented in Appendix A. These curves are all solutions of the analytical model for different values of damping ratio (ζ= .01, .1, .5, 1, 2, 5) and a natural undamped frequency of 1 Hz for the reference solutions. In Figures A-1 to A-6, the damping ratio was varied until the illustrated errors (ε_N= .0075, .01, .1) were obtained. Similarly, in Figures A-7 to A-12, the undamped natural frequency was varied to achieve the same errors. It is seen that ε_N=.0075 to .01 would likely be acceptable as a figure of merit for many practical engineering applications but that ε_N=.1 would definitely be too large. This normal error criterion was chosen over a simple amplitude error criterion because the latter was too intolerant of frequency deviations, especially for the low damping solutions.

The error criterion was used to determine regions of accuracy in the space of system parameters. Damping ratio ζ and natural undamped frequency ω_n are the two parameters of model equation (13) whereas computation time T_C and compensation factor θ are hybrid computer implementation parameters.

It can be shown by analysis that parameters ω_n and T_C can be combined into one, i.e., $\omega_n T_C$ radians/sample. Other authors[8-10] have also used this parameter in accuracy studies. Furthermore, it was verified by experiment in the course of this work that accuracy is a function of this combined parameter. However, the equivalent 'sampling rate' parameter

$$R = (F_n T_C)^{-1} = 2\pi/\omega_n T_C \quad (107)$$

expressed in samples per cycle, was found to be more meaningful and is used in the presentation of results throughout this report.

The same authors[8-10] presented their results only for $\theta = 1.5T_C$, which is the optimum value of θ in the Taylor Series sense. Vichnevetsky,[7] however, is one of the few authors who presented results for other values of θ, but only for stability studies. An objective in this project was to identify regions of improved accuracy with values of θ other than $1.5T_C$. Also, a 'modified compensation factor'

$$\theta' = \theta / 1.5T_C \qquad (108)$$

was defined such that its optimum compensation value in the Taylor Series sense is unity whereas $\theta'=0$ corresponds to the uncompensated hybrid simulation case.

7.0 PERFORMANCE ANALYSIS PROGRAMS

Three programs were developed on the VAX 11/780 digital computer for the performance analysis of the previously defined computerized models. Program RESPLT simply plots the time response of a system which is specified interactively by one of the characters A, B, C or D. The input parameters in DSA and DSB are also entered by the user. Program RESPONS also accepts a character (A, B, C, D) from the user and produces time responses of the specified system. It then solves the system for every combination of input parameters included in Table 3. The data are then stored in disk files. The performance analysis of a system compared to another system with the same input parameter values is executed by program RMSERR where the root mean square (RMS) error is computed between both. This program asks the user to specify two systems to be compared and then performs the analysis on the corresponding solution data produced by program RESPONS. The RMS errors are tabulated and displayed on graphs. Run examples of the three programs are given in Appendix K.

7.1 Program RESPLT

Program RESPLT is an interactive program in which the user is asked to specify one of the four systems (A, B, C, D) and a set of values for the input parameters ζ, F_n, θ' and T_c. The documentation of program RESPLT is contained in Appendix H. The program uses a plotting package[14] to plot time response solutions. It uses the solution data in data set DSR generated by the four computerized models (subroutines SYSTMA, SYSTMB, SYSTMC and SYSTMD) to plot the curves. The abscissa and ordinate are scaled automatically. Furthermore, it displays a fixed number of cycles of oscillation depending on the value of N_T read from disk file DSC.DAT which also contains the value for N_S, i.e., the number of data points to be stored per cycle of solution. The user is required to specify one of several possible output devices, such as a screen terminal, an X-Y plotter or a printer-plotter.

7.2 Program RESPONS

Program RESPONS is documented in Appendix I. Briefly, the user is asked to supply a letter (A to D) corresponding to the system to be solved. The whole range of values of the input parameters in data sets DSA and DSB (Tables 1 and 3) are read from disk files DSA.DAT

and DSB.DAT as well as file DSC.DAT containing the values of N_T and N_S. For the case when System C or D is to be solved, the values of θ' are displayed on the screen and the user is requested to choose one, whereupon the model is solved for every combination of ζ, F_n and T_C for the chosen value of θ'. The user may then choose another value of θ'. On the other hand, Systems A and B, are solved for every combination of the only two applicable parameters, i.e., ζ and F_n. The solution data points temporarily stored by the computerized models in data set DSR are then stored in disk files. The solutions of Systems A and B are stored in files DSRA.DAT and DSRB.DAT respectively. System C data are stored in files DSRC1.DAT to DSRC5.DAT where integers 1 to 5 correspond to the values θ'_i, i=1,...,5. Likewise, for System D, the files are DSRD1.DAT to DSRD5.DAT.

7.3 Program RMSERR

Program RMSERR is documented in Appendix J. The user is requested interactively to enter two characters corresponding to two of the four systems A, B, C and D. The user's response is checked and an error message is issued if there is an inconsistency. Then, as in program RESPONS, if one of the systems entered is C or D, the user has to supply a number i=1,...,5 corresponding to a

desired value θ_i' so that the program can read the appropriate disk files containing the solution data points. Depending on the systems upon which the performance analysis is to be performed, two of the following twelve data files are read into program arrays: DSRA.DAT, DSRB.DAT, DSRC1.DAT to DSRC5.DAT and DSRD1.DAT to DSRD5.DAT. The normalized RMS error $\mathcal{E}_N$ (eq. 106) is then computed for every time response contained in the above files. The time responses are taken in the same order in which they were created by program RESPONS. The values of $\mathcal{E}_N$ are stored in an array which is organized to be tabulated and plotted in one of two alternative ways.

The first alternative produces a table and a graph of $\mathcal{E}_N$ values. When System C or D is involved, the table gives the error as a function of the normalized sampling rate R and damping ratio ζ. The values of R displayed in the table are entered by the user as their corresponding values of $F_n T_c$ in file DSD.DAT (data set DSD). The associated graphs plot $\mathcal{E}_N$ against ζ for several predetermined values of R. For Systems A and B, where T_c is zero, R is replaced by the natural period $T_n = 1/F_n$. The table is stored in disk file DSE'IJ'.TBL, where System 'I' is compared to reference System 'J'. Such tables and graphs are presented in Appendix B.

The second alternative produces a different graph but, first, the user is asked to supply an acceptable error value ε_{NA}. Appropriate computations are made to find the sampling rate R for which the error is smaller than ε_{NA} for a given value of ζ. The graph axes are R against ζ. For every specified combination of R and ζ a dot is plotted on the graph if the error is acceptable. Again, if Systems A and B are involved, R is replaced by T_n. Such graphs appear in Figures 18 to 36 where the dots have been replaced by regions of accuracy.

8.0 PERFORMANCE OF COMPUTERIZED MODELS

The performance analysis programs described in the previous section were implemented to determine the accuracy of the computerized model solutions. In this section the performance of Systems B, C and D are discussed and the correspondence between Systems C and D is established. The stability of sampled-data systems has been studied by several authors.[7,15,16,19] The regions of stability established by these authors are also discussed in more detail in this section. However, the main objective is to determine regions of accuracy within the regions of stability. It is also interesting to find the values of θ' which increase the regions of accuracy and stability.

The computerized models were solved for a practical range of their input parameters. Damping ratio ζ was varied from .002 (negligible damping) to 5 (high damping). The performances of the sampled-data model, System C, and its digital equivalent, System D, were observed for sampling rate $R=(F_n T_c)^{-1}$ values varying from one to 10 thousand samples per natural period. For these two Systems, results were also obtained for the following discrete values of θ': 0, .5, 1, 1.5, 2. However, some results are also given for other values of θ'. Since the analog computer system, System B, does not involve a computation time T_c, the errors are simply observed as a function of the natural period T_n $(=F_n^{-1})$.

The analysis programs produce graphs and tables as output. On the first series of graphs which are accompanied by a table of the exact values, the normalized root mean square error $(\mathcal{E}_N)$ is plotted against ζ with a curve for different values of the sampling rate R. Such graphs and tables are produced for each predetermined value of θ'. In the case of System B, the curves are plotted for different values of T_n only. These tables and graphs are displayed in Appendix B and they serve the purpose of giving a precise value of the error for a fixed set of input parameter values.

With a different objective, the second series of graphs produced by program RMSERR, displays the region of the parameters R vs ζ where the error is considered acceptable ($\mathcal{E}_N \leq .0075$ and .01). In other words, the region indicates the allowable sampling rate that will produce a small error for a given damping ratio and compensation factor. Each graph is produced for a constant value of θ' and $\mathcal{E}_{NA}$. However, for purposes of condensing and comparing results, several curves have been judiciously included in each graph of Figures 18 to 36. The dashed line 'window' in these graphs defines the range of parameters investigated. In the graphs of Figures 20 to 35, different regions of the window can be identified. One is a region of small error (or high accuracy) indicated by $\mathcal{E}_N \leq .0075$. Another is a region of larger error (or intermediate accuracy) indicated by the cross-hatched areas where $.0075 \leq \mathcal{E}_N \leq .01$. The region below the lower line (short and long dashes) is the DTESD model stability boundary under which the solutions grow increasingly large with time. An interesting comment can be made concerning the intermediate region of accuracy. The width of the intermediate region is an indication of the rate at which the error varies when varying the damping ratio or sampling rate. When this region is large, for a fixed ζ, it can be shown that the error increases smoothly for decreasing sampling rates. However, the intermediate error region is non-existent along some

segments of the high accuracy boundaries indicating a more radical variation in the error as the sampling rate or damping ratio varies. Thus some care has to be taken when simulating with these parameters. This is often seen in the region of large ζ where relatively small variations in damping can produce disproportionately large solution errors or even destabilize the system completely, as seen by the proximity of the high accuracy boundary to the stability boundary of the DTESD model.

8.1 System D Relative To A

Vichnevetsky[7] studied the theoretical limit of stability for the solution of an n'th order linear differential equation with constant coefficients using a hybrid computer where the integrations and compensations are done on the analog computer. He derived stability contours in the complex plane of the characteristic roots of the simulated system and plotted these contours for no compensation ($\theta'=0$) and optimum compensation in the Taylor Series sense ($\theta'=1$). Such stability contours can be quite different for other values of θ' as shown by Allison and Johnson[15] for $\theta'=.5$, .75 and 1.5.

To facilitate the comprehension of the stability contours in the case of a SOLDE, they were transformed to the (ζ, R) plane (viz. Appendix L) in Figure 16 where the stable region lies above the stability boundary for each value of θ'=0, .5, 1, 1.5 and 2. The boundaries for θ'=.5, .75, 1 and 1.25 are shown in Figure 17. These stability boundaries were verified with the DTESD computerized model (System D) solutions. It is clear from this graph that, for a given value of damping ratio, θ'=1 is not necessarily the optimum value for stability with minimum sampling rates. In the curves of Figure 17, it is seen that θ'=1 is optimum only for a small range of damping ratio, i.e., $\zeta \in$(.1, .4). In fact, it can be shown by plotting the stability curves of θ' values between .75 and 1.25 that the θ'=1 boundary is practically never the lower curve even for $\zeta \in$(.1, .4). Furthermore, it was verified that θ'=0.5 is optimum for $\zeta \geq 1$ and θ'=1.25 is optimum for $\zeta \leq 0.1$. For example, however, it is seen that the system is stable for $\zeta \in$(.1, 1) with θ'=1 and R$\geq$10 samples per cycle.

As for the stability boundaries, the accuracy boundaries defining the regions wherein $\varepsilon_N \leq .0075$ (good accuracy) are presented in Figure 18. There is a curve for each of the five θ' values of interest. From this

graph, it is clear that $\theta'=1$ and $\theta'=0.5$ are optimum respectively for $\zeta<2$ and $\zeta\geq 2$. Since $\theta'=1$ is optimum for a large range of damping ratios, it is interesting to look at accuracy curves for values of θ' in the vicinity of unity. Some of these are shown in Figure 19. It seems that the accuracy regions, unlike stability, cannot be improved if $\theta'>1$. However, the curves tend to show that the optimum value of θ' for $\zeta>1$ lies in the semi-closed interval $[0.5, 1)$.

The two different levels of accuracy of the DTESD model and the stability boundaries are illustrated in Figures 20 to 24. It is seen that the region of high accuracy sometimes lies below the limit of stability for low damping. This is understandable since low damping implies marginal stability and, although a System D solution might be slightly unstable, its error relative to the analytical solution may satisfy $\varepsilon_N \leq .0075$.

The smallest allowable sampling rate that can be found in these figures is approximately 20 samples/cycle. It occurs when $\theta'=0.5$ (Fig. 21), with $\zeta\simeq 5$ and when $\theta'=1$ (Fig. 22) with $\zeta\simeq 1$. For example, this means that with $\theta'=1$, $\zeta=1$ and $T_C=10$ msec, the highest allowable frequency for good accuracy is approximately 5 hertz, or 10 hertz if $T_C=5$ msec. Also, with $\theta'=1$, good accuracy can be

obtained for $\zeta \in (.002, 2)$ with a sampling rate $R \geq 50$ samples/cycle and the range can be extended to include $\zeta \in (2, 5)$ with $R \geq 90$. However, the same extension can be obtained with $\theta' = 0.5$ and $R \geq 32$.

8.2 System C Relative To A

Accuracy boundaries of System C relative to System A and the digital equivalent stability boundaries are presented in Figures 25 to 29. It can be assumed that these stability curves are not too different from the actual stability limits of System C since the regions of accuracy follow a similar pattern as those of System D. This can be seen by superimposing the graphs of Figures 20 to 24 on those of Figures 25 to 29 respectively. However, there are a few exceptions where the regions of accuracy do not correspond. One of them is the case of $\theta = 1$ (Figs. 22 and 27) and $\zeta \in (.5, 2)$ where the $\varepsilon_N = .0075$ boundary is somewhat lower for System C than for System D. However, the general discussion of System D relative to A is equally applicable to System C. In addition, Figure 30 demonstrates the effect of increasing the acceptable accuracy by decreasing ε_N from .0075 to .005.

8.3 System C Relative To D

For the comparison of the sampled-data computerized model and its discrete-time equivalent, it is more appropriate to talk about regions of equivalence ($\varepsilon_N \leq .0075$) rather than accuracy. The graphs of equivalence regions are shown in Figures 31 to 35.

It is observed that for small values of compensation factor ($\theta' = 0, .5$), both systems are equivalent somewhat beyond the stability boundary in the low damping and high sampling rate region. Otherwise, the equivalence curves all follow the stability boundaries very closely on the stable side. This means that as long as the solutions are stable, System D is equivalent to System C. This total equivalence supports the assumption made by Deiters, Kano and Inoue[10] on the basis of a few verification tests.

8.4 System B Relative To A

Since there is no time-delay in the analog computer solution of the continuous-time model, only one graph of results is required. Two accuracy regions ($\varepsilon_N \leq .0075$ and $\varepsilon_N \geq .01$) of System B are seperated by a single boundary in the natural period versus damping ratio plane of Figure 36. The lack of any shaded area along the

boundary indicates a rapid deterioration of accuracy from one side to the other. To maintain good accuracy, the natural frequency F_n of the SOLDE must be less than 15 hertz (T_n=.067s) if $\zeta \in [.002, .5]$ and less than 30 hertz (T_n=.033s) if $\zeta \in [2, 5]$. The range $\zeta \in (.5, 2)$ is transitional as far as the upper bound of F_n is concerned. Furthermore, these upper limits of frequency for good accuracy represent a sort of ideal which could only be approached with System C by reducing T_c as much as possible.

9.0 CONCLUSION

Four mathematical models describing different methods of solution of a linear second order system were defined. A computerized model corresponding to each mathematical model was implemented and documented. One of the computerized models was a discrete-time analytical solution implemented on a digital computer as System A. A second, System B, was a real-time model solved continuously on an analog computer. System C was an implementation of a real-time sampled-data mathematical model on a hybrid computer. The well-known computational time delay in hybrid loops and a method compensating for it were discussed. The fourth computerized model solved the equations for a discrete-time equivalent of the sampled-data model and was implemented on a digital

computer as System D.

A performance criterion, defined to measure the root mean square of a normal distance approximation between the solutions of any two computerized models, gave an overall idea of the difference between two solutions.

A set of three general programs for the performance analysis of the computerized models were described and documented. With some minor changes, these programs could be used to investigate other systems such as nonlinear or higher order linear systems.

Useful results were obtained concerning the accuracy of the computerized models. These results are not only useful for the continuing work of this research but can also be useful to the hybrid programmer who would want to specify, for example, a safe value of sampling rate or the optimum compensation factor for a given SOLDE application. Although it is risky to extrapolate, they can at least serve as a starting point for second order dominant systems. Accuracy and stability boundaries for the DTESD model showed that the optimum value of analog compensation factor is a function of damping ratio and is, in general, different from the optimum value in the Taylor series sense. Also, it was found that System D is equivalent to System C in the stable region of parameters. The upper limits of frequency for good accuracy

of the analog computer model (System B) were determined. These limits represent, in a sense, the limits of System C when the hybrid time-delay tends to zero.

The programs developed in this work can serve as basic tools in the performance analysis of other computerized models. One interesting model would be to connect the integrators in series rather than independently as in Systems C and D. For this, a different compensation scheme would probably be needed. It is the ultimate goal of this continuing research to produce a hybrid computer programming package which would automate analog computer programming for the user. Such a package might include optimization of analog compensation factors. The option of using digital as well as analog integration would be provided. For example, in case of shortage of analog integrators, an equivalent to sampled-data model similar to System D could be used as a supplement. Its equations would thus be solved synchronously on the digital computer. Variables would be computed at every hybrid computation interval T_C and compensation parameter θ would be adjusted accordingly.

REFERENCES

1. Kleinert, W. (1982). "The New Hybrid Time-Sharing System MACHYS", EDP Centre, Technical University of Vienna, No. 19, December 1982, 18 pages.

2. Miura, T., and Iwata, J. (1963). "Effects of Digital Execution Time in a Hybrid Computer", Proceedings AFIPS Fall Joint Computer Conference, Vol. 24, 1963, pp. 251-266.

3. Gilbert, E. G. (1966). "Dynamic-Error Analysis of Digital and Combined Analog-Digital Computer Systems", Simulation, Vol. 6, No. 4, April 1966, pp. 241-257.

4. Matlock, David L. (1966). "Pulsed Prediction Filters Applied to Digital and Hybrid Simualtion", Simulation, Vol. 6, No. 3, March 1966, pp. 153-157.

5. Deiters, Robert M., and Nomura, Tamiya (1967). "Circle Test Evaluation of a Method of Compensating Hybrid Computing Error by Predicted Integral", Simulation, Vol. 8, No. 1, January 1967, pp. 33-40.

6. Karplus, Walter J. (1966). "Error Analysis of Hybrid Computer Systems", Simualtion, Vol. 6, No. 2, February 1966, pp. 120-136.

7. Vichnevetsky, R. (1969). "Stability Contours for the Analysis of Analog/Digital Hybrid Simulation Loops", Proceedings AFIPS Spring Joint Computer Conference, Vol. 34, 1969, pp. 859-866.

8. Curry, W. H. (1977). "An Evaluation of a Correction Method for Digital Computer Lag in Hybrid Systems", Proceedings Summer Computer Simulation Conference,

Chicago, July 1977, pp. 94-101.

9. Howe, Robert M. (1975). "Frame-Rate Requirements for Digital Function Generation in Hybrid Computing Loops", Winter Computer Simulation Conference, Sacremento, December 1975, pp. 457-461.

10. Deiters, R. M., Kano, K., and Inoue, K. (1971). "Practical Stability Error Charts for some Commonly Used Methods of Hybrid Computing Error Compensation", Proceedings AICA Symposium Simulation Complex Systems, Tokyo, September 1971, pp. D2.1-6.

11. Amyot, J. R. (1982). "Defining a Real-Time Model", Simulation, Vol. 38, No. 1, January, 1982.

12. Korn, G. A. (1962). "Approximations of Variable Time Delays Require Caution", Transactions of IRE on Electronic Computers, Vol EC-11, No. 1, February 1962, p. 82.

13. Gille J-C, Pelegrin M.J., Decaulne P., 'Feedback Control System', McGraw-Hill, 1959.

14. 'PLOT 10 ADVANCED GRAPHING II, USER'S MANUAL', Tektronix, 1977

15. Allison, J. S., and Johnson H. M. (1979). "Stability of Hybrid Simulation of Dynamic Systems", Mathematics and Computers in Simulation XXI, North-Holland Publishing Company, 1979, pp. 289-303.

16. Little, Warren D. (1973). "Serial Hybrid Computation and Errors in Hybrid Loops", Transactions of IEEE on Computers, Vol. C-22, No. 4, April 1973, pp. 367-370.

17. El-Sherif, A. K., Kamal, A. A., and Bilial, A. Y. (1976). "Error Analysis in Hybrid Computer Simulation of

Dynamic Systems", Transactions IMACS (Annales de l' AICA), Vol. 18, No. 3, pp. 141-147.

18. Kemp, N. H. (1969). "Extensions and Analysis of Use of Derivatives for Compensation of Hybrid Solution of Linear Differential Equations", Proceedings AFIPS Fall Joint Computer Conference, Vol. 35, 1969, pp. 761-770.

19. Vansteenkiste, G. C. (1973). "Precision and Efficiency of Some Hybrid Algorithms", Proceedings 7th AICA Conference, Part 1, Prague, August 1973, pp. 94-97.

20. Bekey, G. A., and Karplus, W. J. (1968). "Hybrid Computation", John Wiley and Sons Inc., New York, 1968.

21. Boullart, Dr. I. L. (1982). "Hybrid Computing, Evaluation of 10 Years Experience in a University Environment", Tenth IMACS World Congress Proceedings, Vol. 2, Montreal, August 1982, pp. 9-12.

22. Connely, Marc E. (1961). "Real-Time Analog-Digital Computation", IRE Convention Record, Part 2, 1961, pp. 182-195.

23. Crosbie, Roy E., and Hay, John L. (1976). "Experience in Using the HCS-1 Hybrid Compiler", Simulation, Vol. 6, No. 2, December 1976, pp. 127-131.

24. Crosbie, Roy E., and Hay, John L. (1982). "Towards New Standards for Continuous System Simulation Languages", Proceedings Summer Computer Simulation Conference, Denver, July 1982, pp. 186-190.

25. Danilin, A. B., Spiro, A. G., and Yadykin, I. B. (1981). "Accuracy of Simulation of Hybrid Control Systems with a Digital Controller", Autom. and Remote Control, 42(11,Part 2), November 1981, pp 1577-81.

26. Deiters, Robert M., and Nomura, Tamiya (1967). "Predicted Integral Method of Compensating for Sampling and Time-Delay Errors in a Hybrid Computing System", Proceedings 5th AICA Conference, Lausanne, August 1967, pp. 181-190.

27. El-Sherif, A. K., Kamal, A. A., and Bilial, A. Y. (1975). "An Improved Scheme for Hybrid Computer Simulation of Dynamic Systems", Proceedings 6th Conference on Modeling and Simulation, Pittsburgh, April 1975, pp. 249-252.

28. Fleischer, Paul E. (1972). "Digital Realization of Complex Transfer Functions", Simulation, Vol. 6, No. 3, March 1966, pp. 171-180.

29. Gelman, Robert (1963). "Corrected Inputs--A Method for Improved Hybrid Simulation", Proceedings AFIPS Fall Joint Computer Conference, Vol. 24, 1963, pp. 267-276.

30. Genin, J., and Genin, R. (1973). "Etudes Analogiques des Equations", Annales de l' AICA, Vol. 15, No. 3, July 1973, pp. 104-114.

31. Gibson, J. A., and Gibbard, R. W. (1980). "Hybrid Serial-Parallel Digital Computation", Computer and Electrical Engineering, Vol. 7, 1980, pp. 25-34.

32. Gilbert, E. G., and Vansteenkiste, G. C. (1970). "Dynamic Error Analysis of Hybrid Computer Loops", Proceedings 6th AICA Conference, Munich, August 1970, pp. 227-234.

33. Gilbert, E. G. (1967). "Error Analysis of Hybrid Computer Loops", Annales de l' AICA, Vol. 9, No. 1, January 1967, pp. 35-36.

34. Giloi, W. (1967). "Error Corrected Operation of Hybrid Computer Systems", Proceedings 5th AICA Conference, Lausanne, August 1967, pp. 172-180.

35. Grierson, W. O., Lipski, D. B., and Tiffany, N. O. (1980). "Simulation Tools: Where can we go? ", Proceedings Summer Computer Simulation Conference, Seattle, August 1980, pp. 3-6.

36. Hammond, J. L., and Alford C. O. (1969). "Sampling Errors in Closed-Loop Hybrid Computer", Simulation, Vol. 13, No. 6, December 1969, pp. 307-313.

37. Howe, R. M. (1971). "The Future of Automatically Patched Hybrid Computers", Proceedings AICA Symposium Complex Systems, Tokyo, September 1971, pp. A1.1-8.

38. Howe, R. M., Holstien, R. B., and Moran, R. A. (1972). "Hardware/Software Considerations in the AD/FOUR Electronic Patched Hybrid Computer", Information Processing 71, North-Holland Publishing Company, 1972, pp. 668-674.

39. Howe, R. M., and Landauer, J. P., "A Quantitative Method of Speed Comparison Between Analog/Hybrid and Digital Computers", Computer, July 1976.

40. Howe, R. M. (1967). "Some Techniques for Accuracy Improvement in Analog Computation", Simulation, Vol. 9, No. 4, October 1967, pp. 173-179.

41. Justice, Harold R. (1981). "Programming an Analog Computer with a High Level Language", Proceedings Summer Computer Simulation Conference, Washington D.C., July 1981, pp. 166-169.

42. Karplus, Walter J., and Wood, Roger C. (1965). "Error

Analysis of Digital-Computer-Oriented Hybrid Computer Systems", Proceedings IFIP Conference, New York, May 1965, pp. 428-429.

43. Katz, S., and Neumann, D. (1980). "A Novel Approach to Real-Time Hybrid Simulation", Simulation of Systems 79, North-Holland Publishing Company, 1980.

44. Kerckhoffs, E. J. H., Dekker, L., and van Gelderen, J. A. (1973). "A Block-Iteration Method for the Hybrid Solution of Large Systems of Ordinary Differential Equations", Proceedings 7th AICA Conference, Part 1, Prague, August 1973, pp. 125-129.

45. Kerckhoffs, E. J. H. (1978). "On the Analog Computation of Integrals", Simulation, Vol. 30, No. 5, May 1978, pp. 169-170.

46. Korn, G. A. (1967). "Automated Computing-Error Studies in Fast Hybrid Computations", Annales de l' AICA, Vol. 9, No. 1, January 1967, p. 34.

47. Korn, G. A. (1978). "Digital Continuous-System Simulation", Prentice-Hall, 1978.

48. Korn, G. A. (1963). "Parameter-Pertubation Generator for Analog-Computer Error and Optimization Studies", Annales de l' AICA, Vol. 5, No. 2, April 1963, pp. 93-95.

49. Korn, G. A. (1964). "A Simple First-Order Hold Circuit", Simulation, Vol. 2, No. 2, February 1964, p. 17.

50. Korn, G. A. (1969). "Variable Time-Delay Simulation", Simulation, Vol. 13, No. 4, October 1969, p. 218.

51. Kuo, B. C. (1962). "Automatic Control Systems", Prentice-Hall, 1962.

52. Maslo, Ronald M. (1982). "A survey of Hybrid Computer Uses in Control Systems Design", Proceedings Summer Computer Simulation Conference, Denver, July 1982, pp. 83-85.

53. Mitchell, E. E. L. (1975). "Advanced Continuous Simulation Language User Guide/Reference Manual", Mitchell and Gauthier Associates Inc., Concord, Massachusetts, 1975.

54. Mitchell, E. E. L. (1982). "Advanced Continuous Simulation Language (ACSL): An Update", Tenth IMACS World Congress Proceedings, Vol. 1, Montreal, August 1982, pp. 462-464.

55. Mitchell, E. E. L. (1967). "The Effect of Digital Compensation for Computation Delay in a Hybrid Loop", Proceedings AFIPS Fall Joint Computer Conference, Vol. 31, 1967, pp. 103-108.

56. Miura, T. (1964). "Hybrid Computation Techniques for Differential Equations", Proceedings 4th International AICA Meeting, Brighton, September 1964, pp. 47-58.

57. Miura, T., and Iwata, J. (1965). "Study on Time-Shared Analog Computation Technique" Simulation, Vol. 5, No. 5, November 1965, pp. 318-327.

58. Paul, J. F. (1980). "The Effect of Recent Developments in Hybrid Simulation on Modern Scientific Computation", Proceedings Summer Computer Simulation Conference, Seattle, August 1980, pp. 7-8.

59. Phillips, Robert A. (1980). "Getting Productivity from a

Hybrid Computer Facility", Proceedings Summer Computer Simulation Conference, Seattle, August 1980, pp. 9-11.

60. Van Remoortere, P. (1982). "Introduction to the Session on the User's Experience in Hybrid Computation", Tenth IMACS World Congress Proceedings, Vol. 2, Montreal, August 1982, pp. 1-2.

61. Saydam, Tuncay (1982). "Systems Simulations and Computer Graphics Interface Issues", Tenth IMACS World Congress Proceedings, Vol. 4, Montreal, August 1982, pp. 216-218.

62. Sheppard, Sallie (1983). "Applying Software Engineering to Simulation", pp. 13-19.

63. Strauss, Jon C. (1967). "The SCi Continuous System Simulation Language", Simulation, Vol. 9, No. 6, December 1967, pp. 281-303.

64. Trier, Reinhold (1982). "The Operation of a Hybrid Computer System Using a Simulation Language", Tenth IMACS World Congress Proceedings, Vol. 2, Montreal, August 1982, pp. 7-8.

65. Valisalo, P. (1982). "Hybrid Computation--Today and Tomorrow", Tenth IMACS World Congress Proceedings, Vol. 2, Montreal, August 1982, p. 13.

66. Vansteenkiste, G. C. (1971). "Analog or Digital Compensation for the Simulation of Complex Systems in a Hybrid Way", Proceedings AICA Symposium Simulation Complex Systems, Tokyo, September 1971, pp. D2.1-6.

67. Vansteenkiste, G. C. (1976). "Mixed Data Processing Using Parallel Compensation", ***(UNKNOWN)

68. Vansteenkiste, G. C. (1969). "Study of the Dynamic

Errors Introduced by the Delays Iinherent on Combined Analog-Digital Computation in Hybrid Systems", Annales de l' AICA, Vol. 11, No. 4, October 1969, pp. 226-232.

69. Vichnevetsky, Robert (1967). "Error Analysis in the Computer Simulation of Dynamic Systems: Variational Aspects of the Problem", Transactions of IEEE on Computers, Vol. EC-16, No. 4, August 1967, pp. 403-411.

70. Walli, C. R. (1964). "Quantizing and Sampling Errors in Hybrid Computation", Proceedings AFIPS Fall Joint Computer Conference, Vol. 26, Part 1, October 1964, pp. 545-558.

TABLES

TABLE	CONTENT	PAGE
1	Data Set Definitions	74
2	Symbol Definitions	75
3	Specific Set of Model Parameter values	79

Table 1: Data Set Definitions

Data set name	Contents *
DSA	ζ, F_n
DSAU	ζ, F_n, specified interactively by user
DSB	θ', T_c
DSBU	θ', T_c, specified interactively by user
DSC	N_T, N_S
DSD	$F_n T_c$ values to be displayed in error tables
DSE	ε_N table
DSEBA	ε_N table for System B relative to A
DSECA	ε_N table for System C relative to A
DSECD	ε_N table for System C relative to D
DSEDA	ε_N table for System D relative to A
DSR	x^i, t^i
DSR1	DSR used in program RMSERR to compare solutions of S_1 and S_2
DSR2	DSR used in program RMSERR to compare solutions of S_1 and S_2
DSRA	x^i, t^i of System A
DSRB	x^i, t^i of System B
DSRC'k'	x^i, t^i of System C corresponding to k'th value of θ' in DSB
DSRD'k'	x^i, t^i of System D corresponding to k'th value of θ' in DSB

* Contents are defined in terms of parameter names. The sets of values associated with these parameters constitute the data set.

Table 2: Symbol Definitions

SYMBOL	FORTRAN NAME	DEFINITION
A		System coefficient matrix
B		Forcing function coefficient matrix
D_k	DCA(k)	Value of DCA number k
F		Forcing function vector
F_i		Value of F at time t_i
F_n	FN	Natural undamped frequency, hertz
G	G	Integrator gain
i	NPOINT	Discretization index for solution storage
J		Discretization index for solution computation
L		Line parallel to L_R and intersecting x at a given time t
L_N		Line L normalized
L_R		Line tangent to x_R at a given time t
L_{RN}		Line L_R normalized
N_S	NS	Total number of samples per cycle to be stored
N_T	NT	Number of cycles per solution
R		Normalized sampling rate (samples/cycle)
R_k	RELAY(k)	Relay number k; state (0/1)
S	SYSTM	User specified system
S_1	SYSTM	User specified system
S_2	SYSTM	User specified reference system
t	TIME	Clock time
T		Sampling and update time interval
t'	TIME	Model time
T_c	TC	Computation and conversion time in in hybrid loop

SYMBOL	FORTRAN NAME	DEFINITION
T_F	FINALTIME	Final solution time
t^i	RESTIME	Discrete time for storage of solution
t^j	TIME	Discrete time for computation of solution
T_n		Undamped natural period of a SOLDE
T_S	SAVINCTIM	Sampling period for storage of solution data
T_1	T1	Time period over which ε_N is computed
t_1		Time corresponding to the horizontal projection of x_R onto L
t_{1N}		Normalized value of t_1
u	STEP	Unit step function
V		Compensation variable
V^i		Value of V at time t^i
v_1		First element of V
v_2		Second element of V
W		Input vector to continuous-time subsystem in sampled-data model
W^i	DAC	Value of W at time t^i
w_1		First element of W
w_2		Second element of W
x	X	Response function
X		Response function vector
$\dot{x}$	XD	First derivative of x with respect to time
$\ddot{x}$	XDD	Second derivative of x with respect to time
x^i	RESPONS	Value of x at time t^i
X^i	ADC	Value of X at time t^i
x_M	XM	Amplitude scale factor for x

SYMBOL	FORTRAN NAME	DEFINITION
x_M	XDM	Amplitude scale factor for x
$\ddot{x}_M$	XDDM	Amplitude scale factor for $\ddot{x}$
x_{MAX}	RESPM	Maximum amplitude of x
x_R		Reference response function
$\dot{x}_R$		Derivative of x_R
x_{RN}		Function x_R normalized
$\dot{x}_{RN}$		Normalized derivative of x_R
x_S	XS	Scaled value of x
$\dot{x}_S$	XDS, XDAS	Scaled value of $\dot{x}$
$\ddot{x}_S$	XDDAS	Scaled value of $\ddot{x}$
x_0		Initial condition on x
$\dot{x}_0$		Initial condition on $\dot{x}$
x_1		First element of X
x_2		Second element of X
Y		Ideal input function to continuous-time subsystem in sampled-data model
Z		Continuous-time integration output vector
Z^i		Value of Z at time t^i
z_1		First element of Z
z_2		Second element of Z
α		Slope angle of x_R at a given time t
α_N		Normalized value of α
β	BETA	Time-scale factor
ε_N	ERMSN	Normalized root mean square error

SYMBOL	FORTRAN NAME	DEFINITION
ε_{NA}	ACCEPT	Acceptable value of ε_N
ζ	ZETA	Damping ratio
θ	THETA	Compensation factor
θ'	THETAP	Modified compensation factor
θ'_i	NUMTHETA	i'th θ' value in data set DSB
δ		Approximate distance in the direction perpendicular to x_R
δ_N		Normalized value of δ
Δt		Approximate distance in horizontal direction between two solutions
Δt_N		Normalized value of Δt
Δx		Difference in amplitude between two solutions
Δx_N		Normalized value of Δx
τ		Initial time-delay in the hybrid loop
τ_1		Analog to digital conversion delay
τ_2		Digital computation delay
τ_3		Digital to analog conversion delay
ω_n	OMEGAN	Natural undamped frequency in time-scaled systems, rad/sec
ω_n^*	OMEGAN	Natural undamped frequency in real systems, rad/sec

Table 3: Specific Set of Model Parameters for which Systems C and D were solved in the performance analysis

θ°	T_c	F_n	ζ
0.0	0.01	0.01	0.002
0.5	0.025	0.1	0.02
1.0	0.05	0.5	0.05
1.5	0.1	1.0	0.1
2.0	0.2	3.0	0.2
		5.0	0.5
		7.0	1.0
			2.0
			5.0

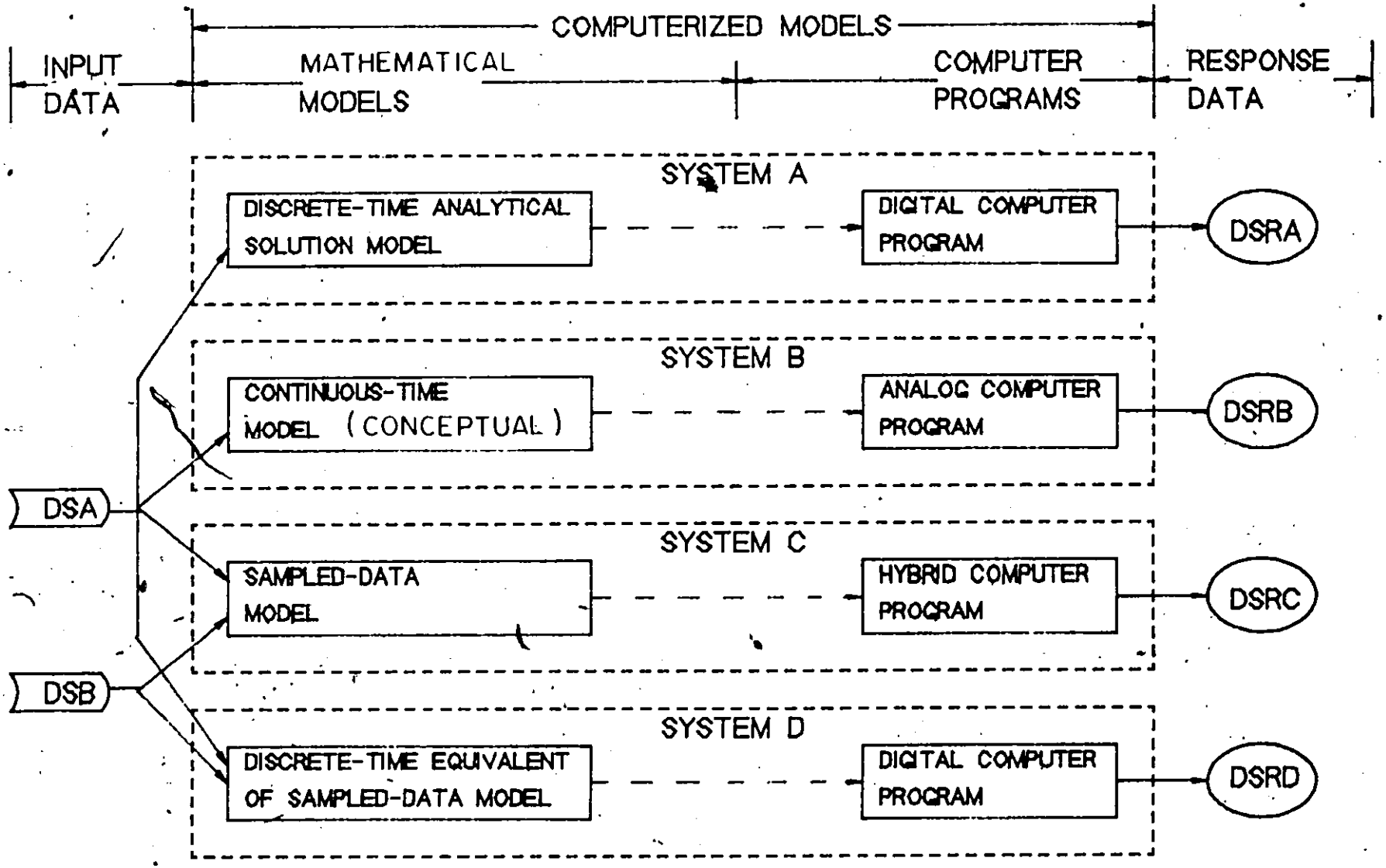


Figure 1 : Input–process–output diagram of computerized models

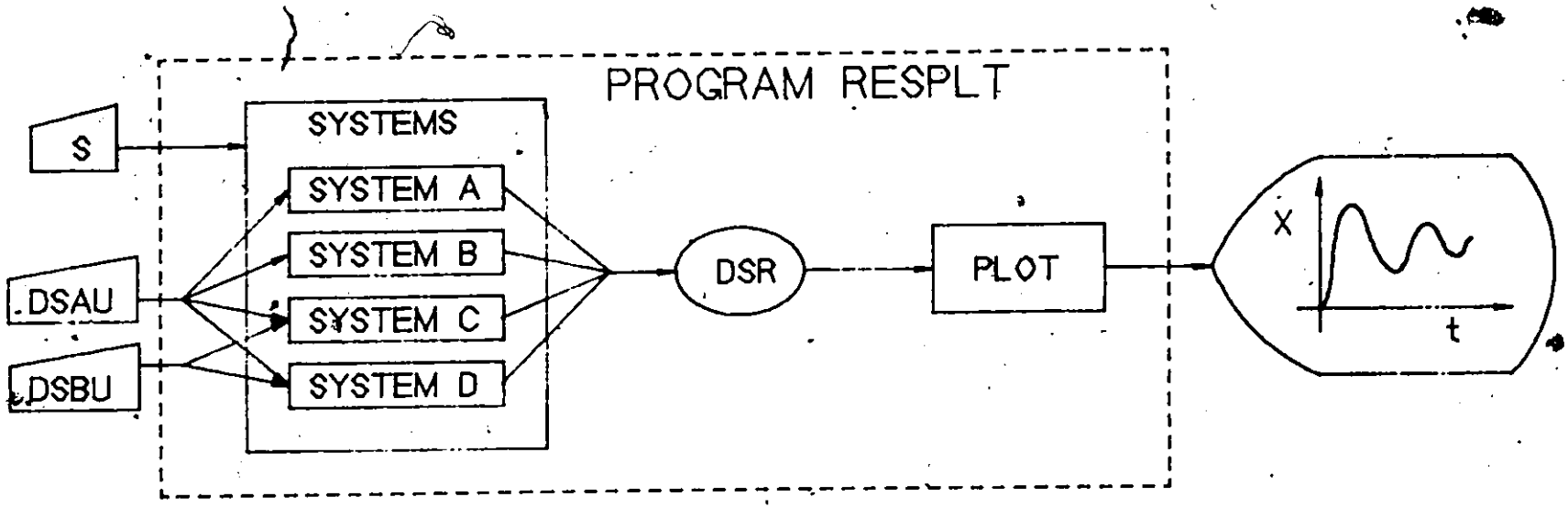


Figure 2 : File flow diagram of program RESPLT

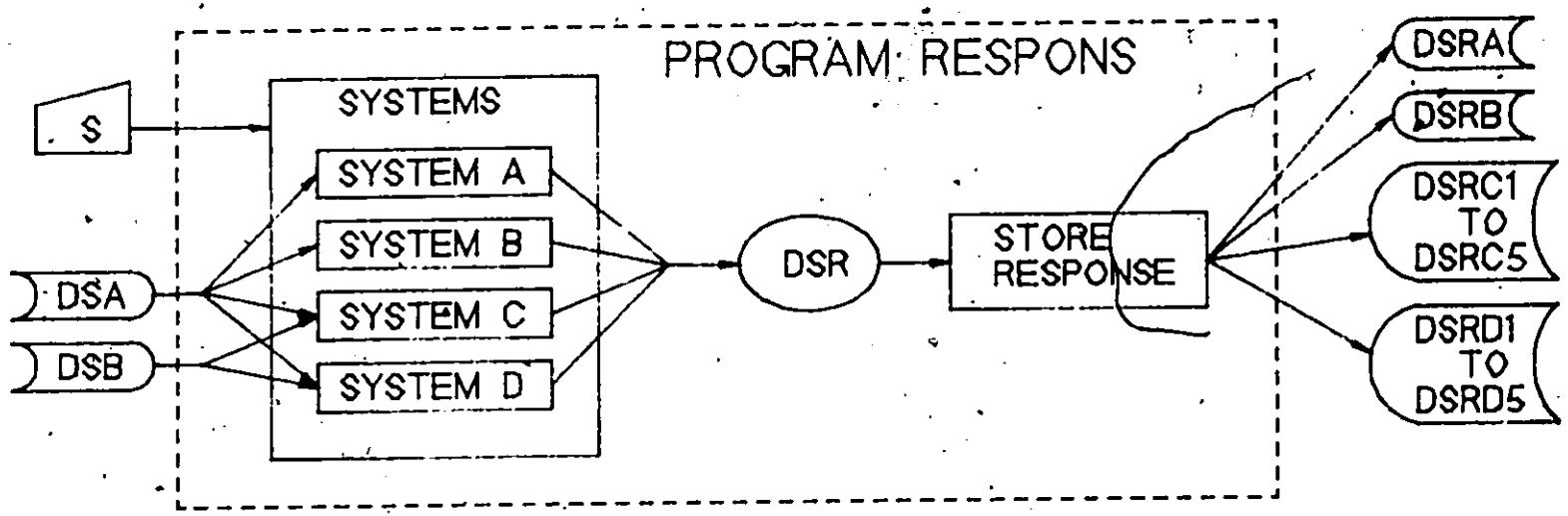


Figure 3 : File flow diagram of program RESPONS

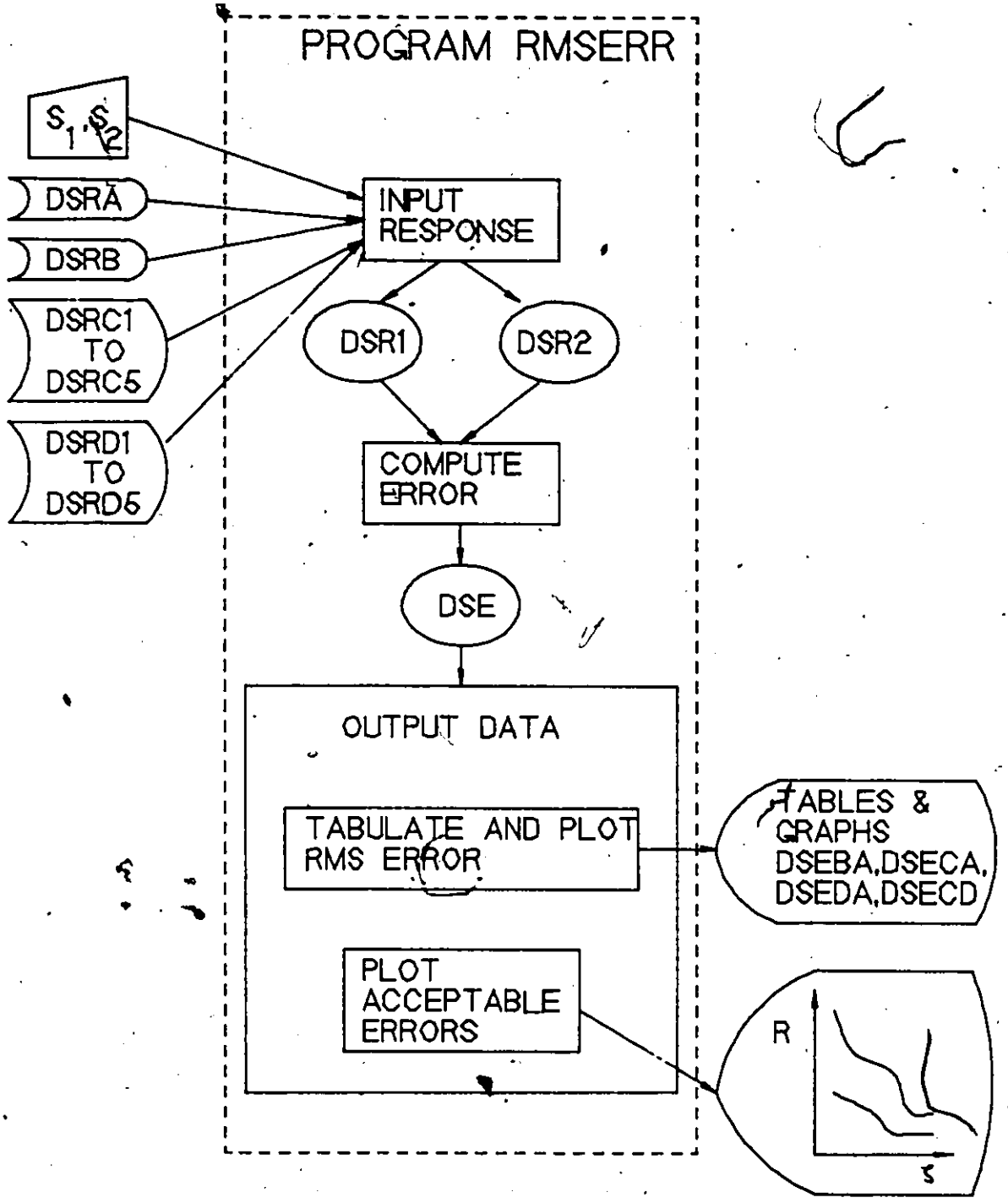


Figure 4 : File flow diagram of program RMSERR

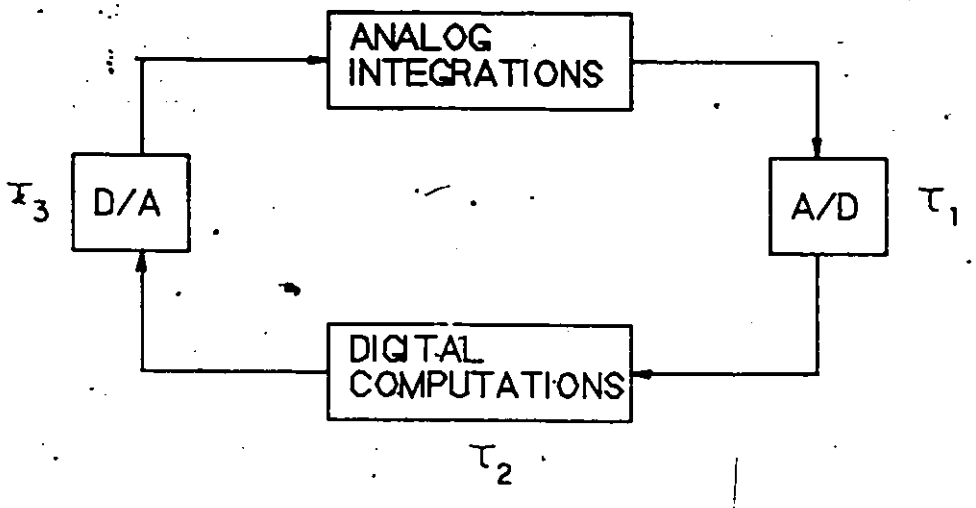


Figure 5 : Hybrid–loop with component delays

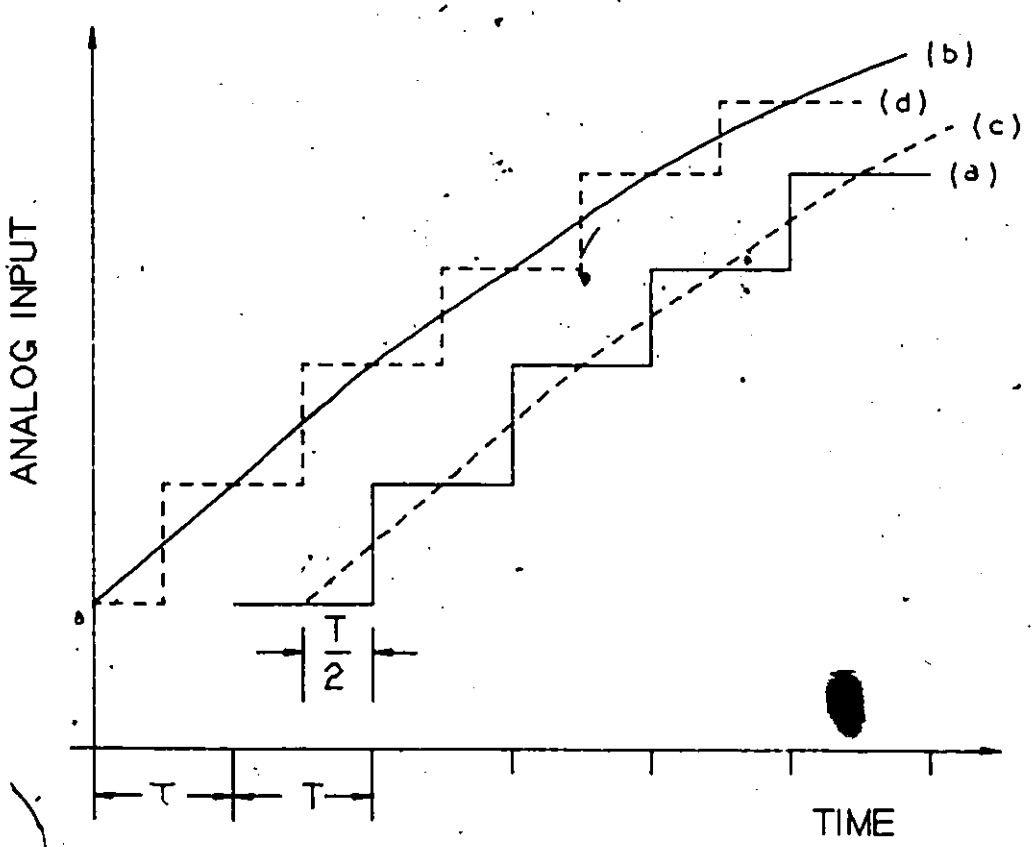


Figure 6 : Analog integrator input functions

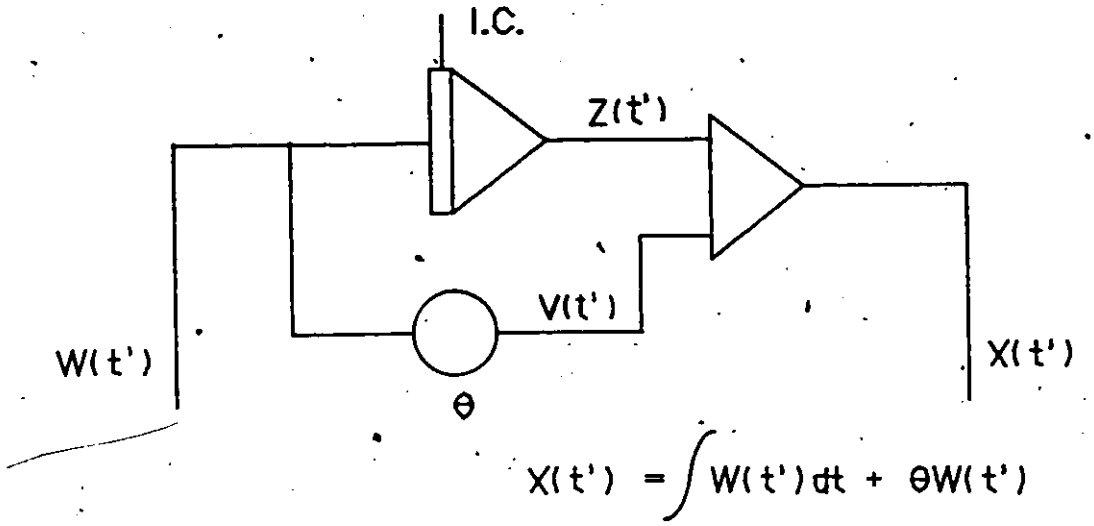


Figure 7 : Simplified diagram of analog operations in hybrid loop

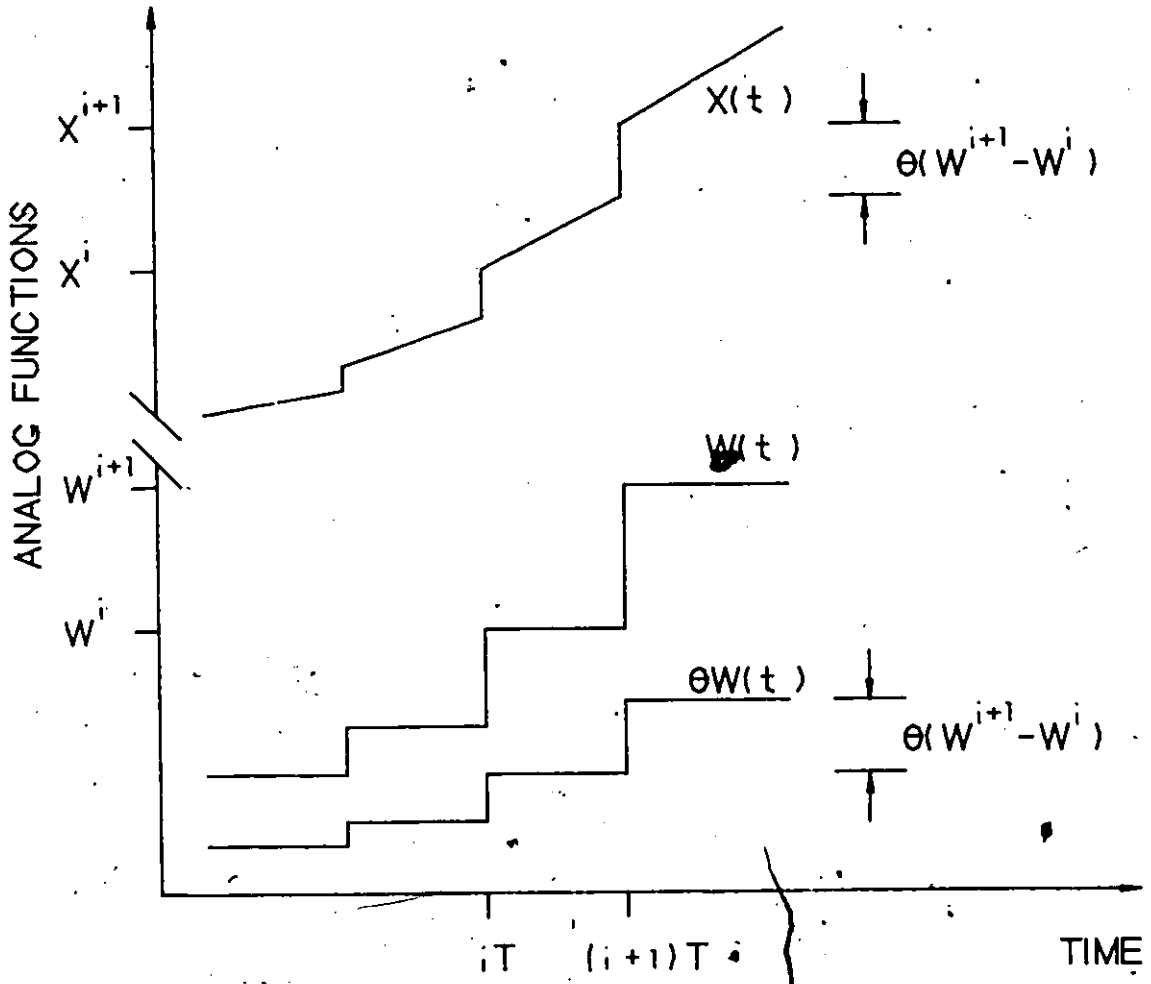


Figure 8 : Analog time functions in hybrid loop

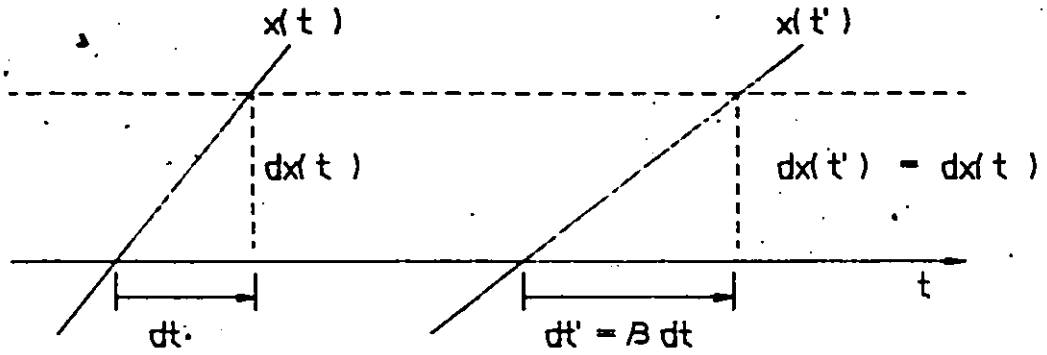


Figure 9 : Real–time to model–time conversion diagram

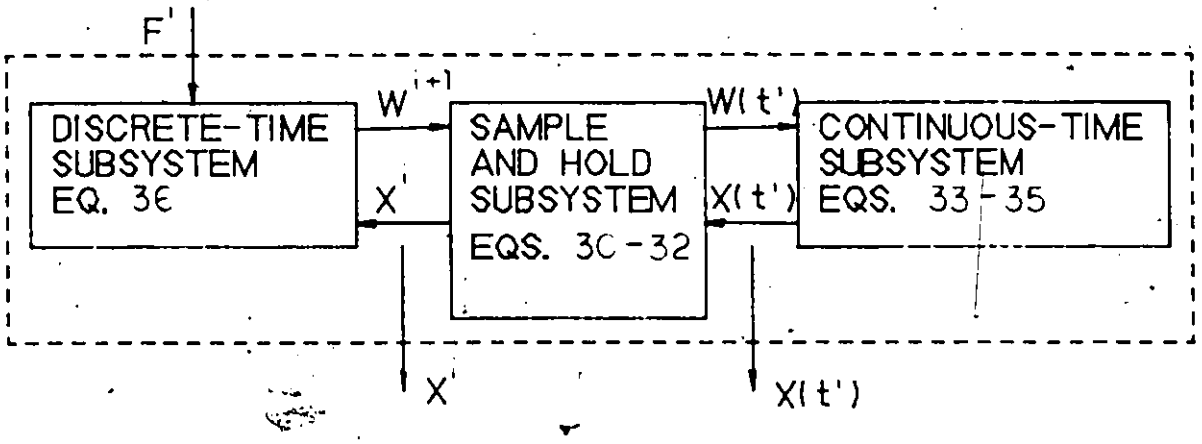


Figure 10 : Sampled–data mathematical model

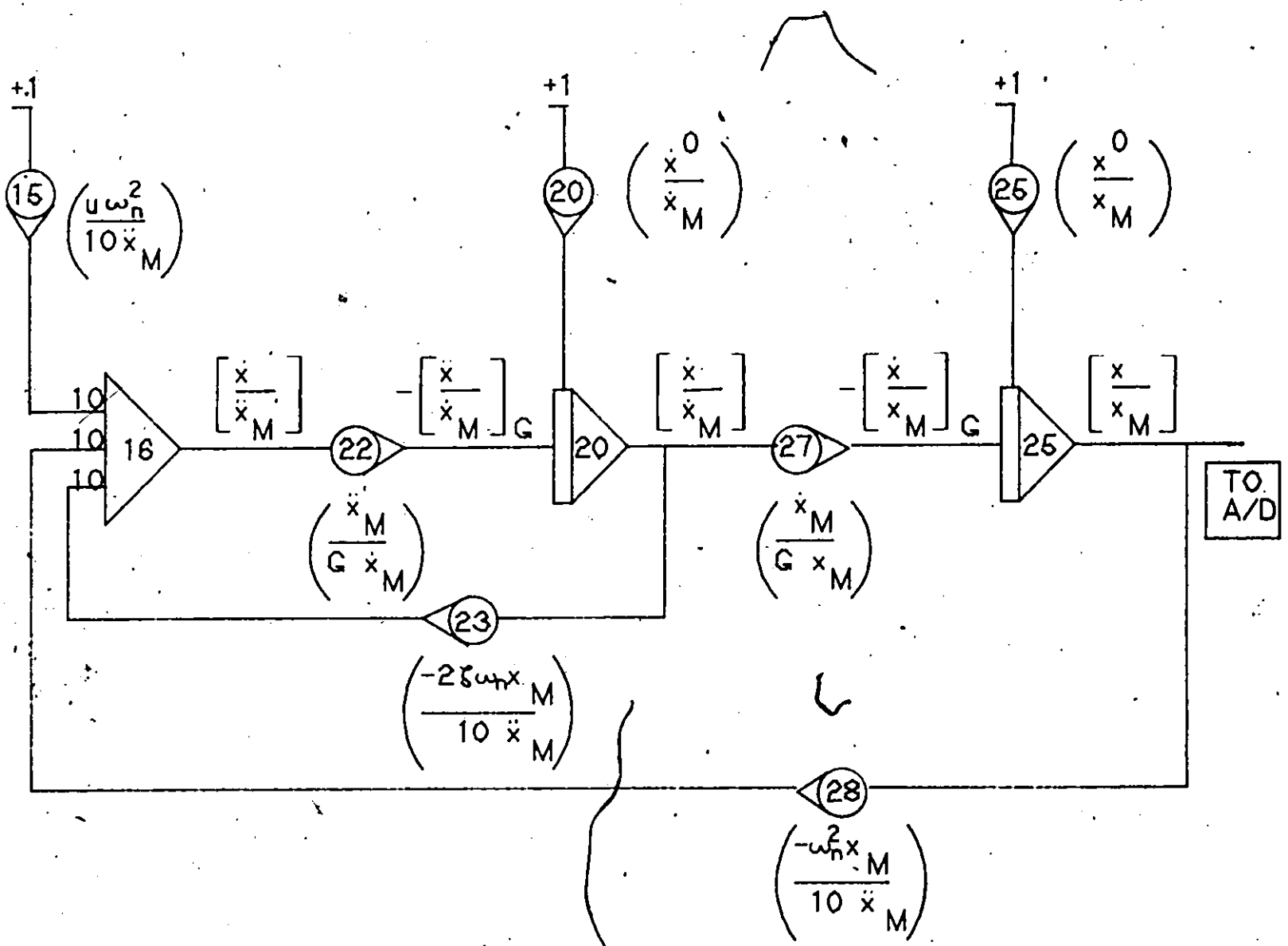


Figure 11 : Analog computer circuit diagram

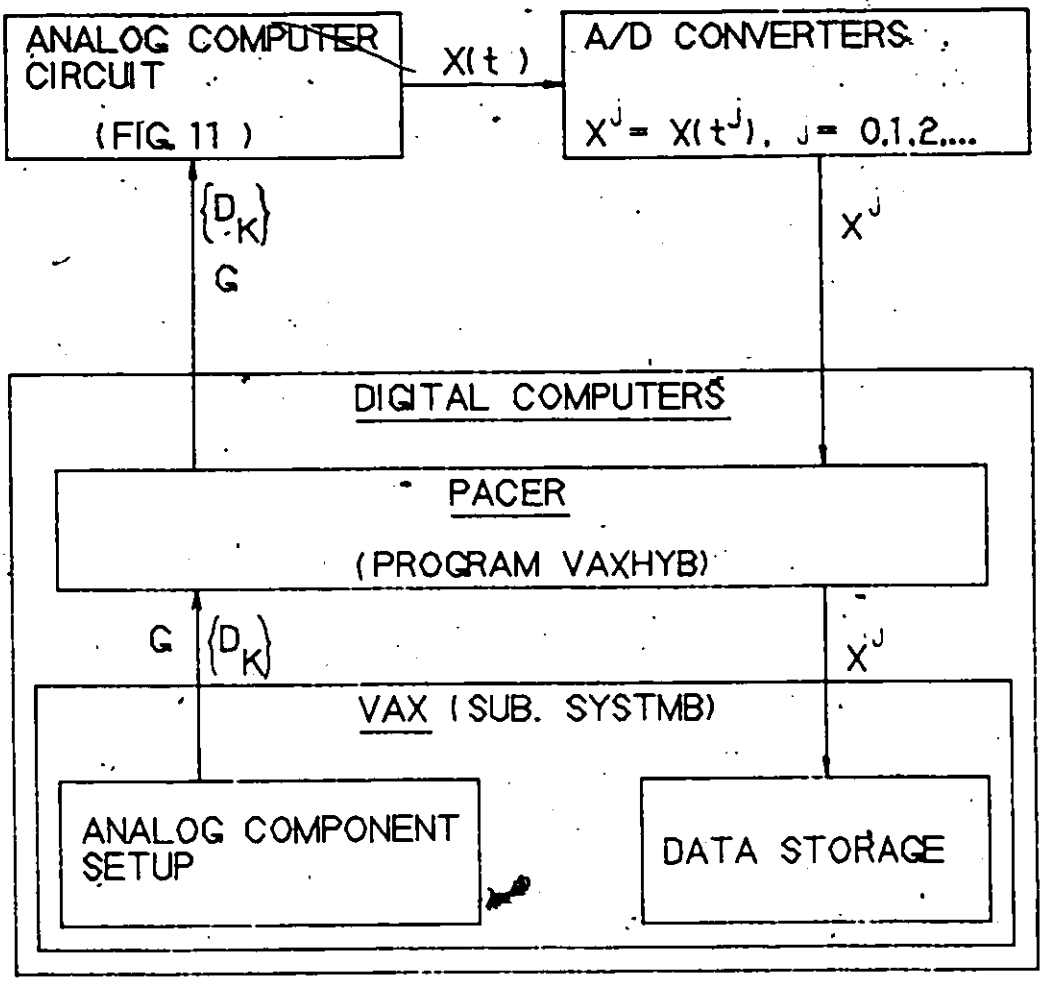


Figure 12 : System B information flow diagram

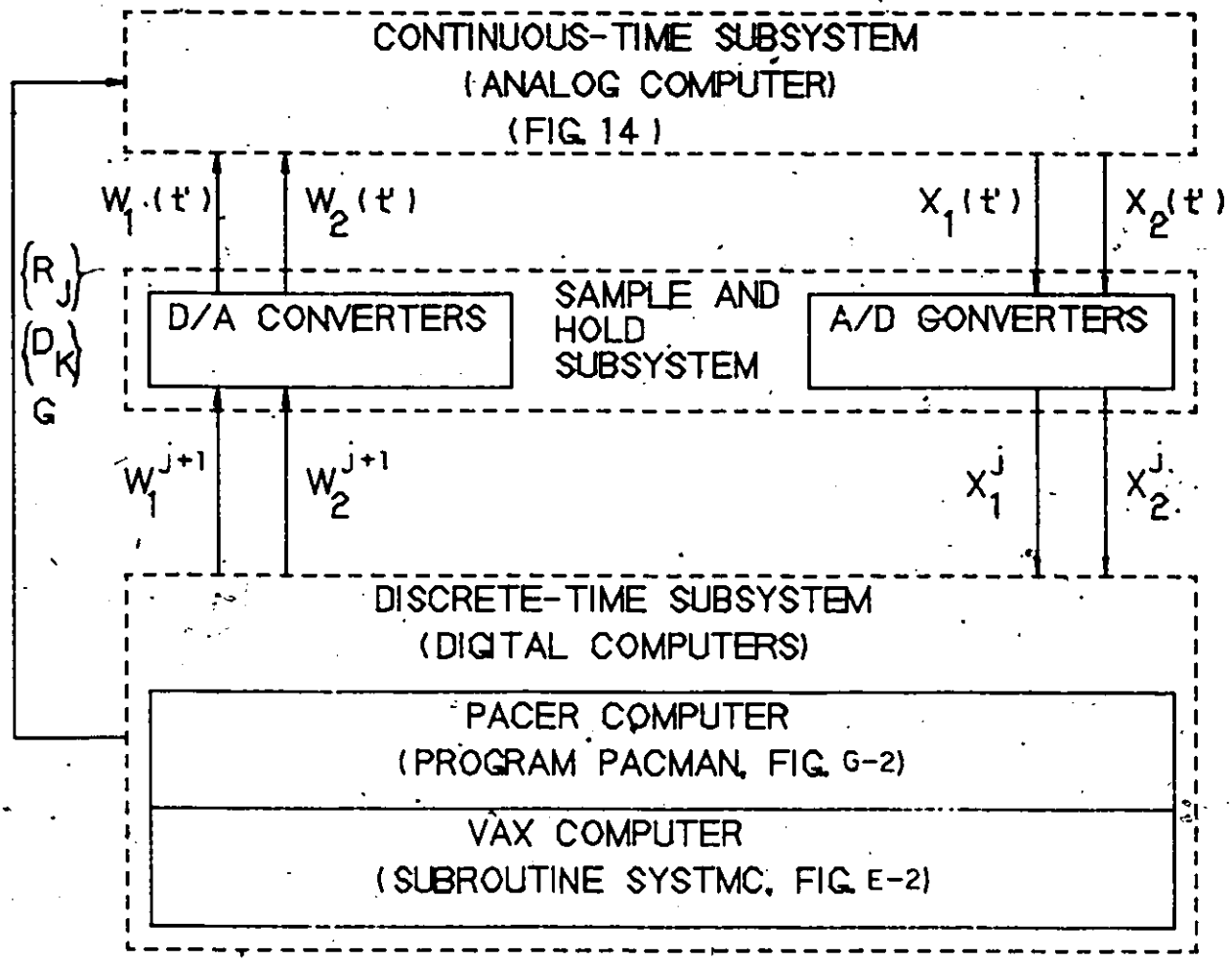


Figure 13 : System C information flow diagram

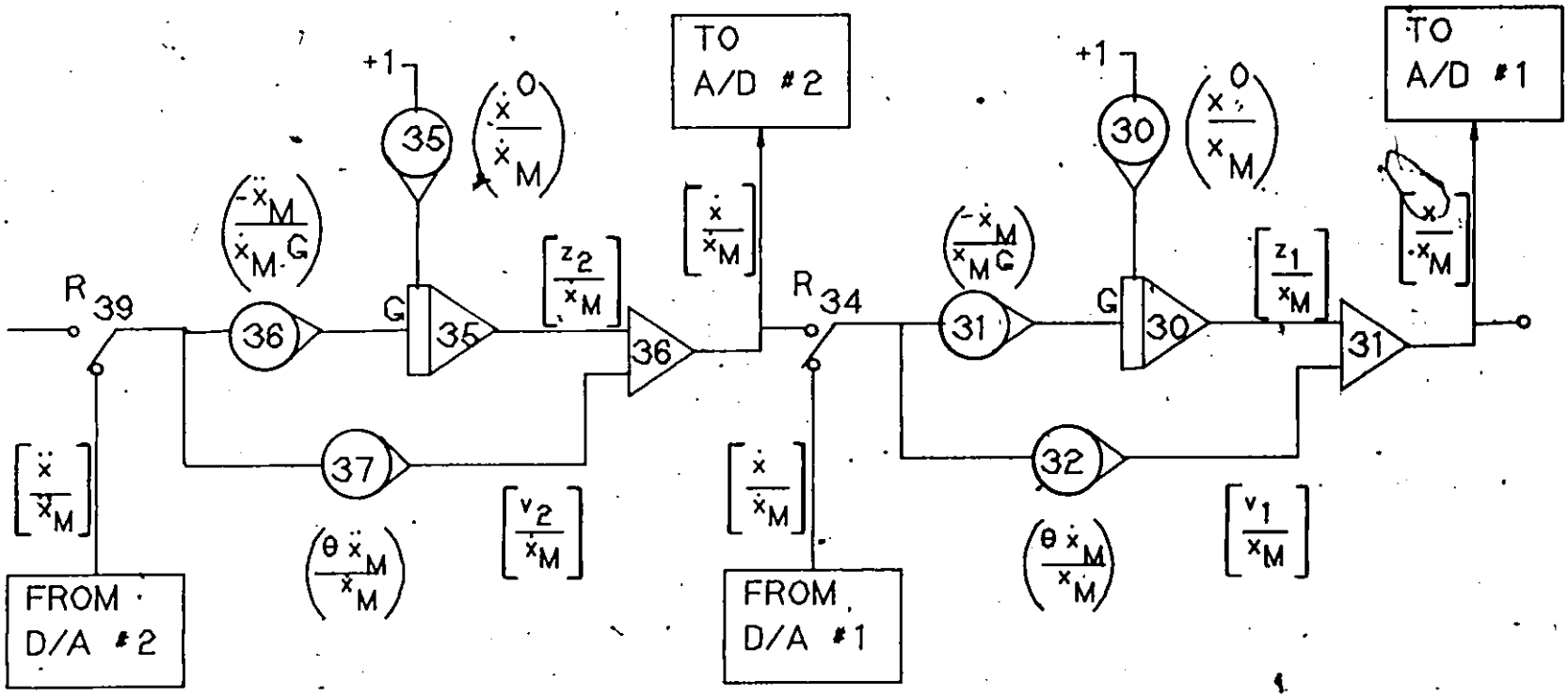
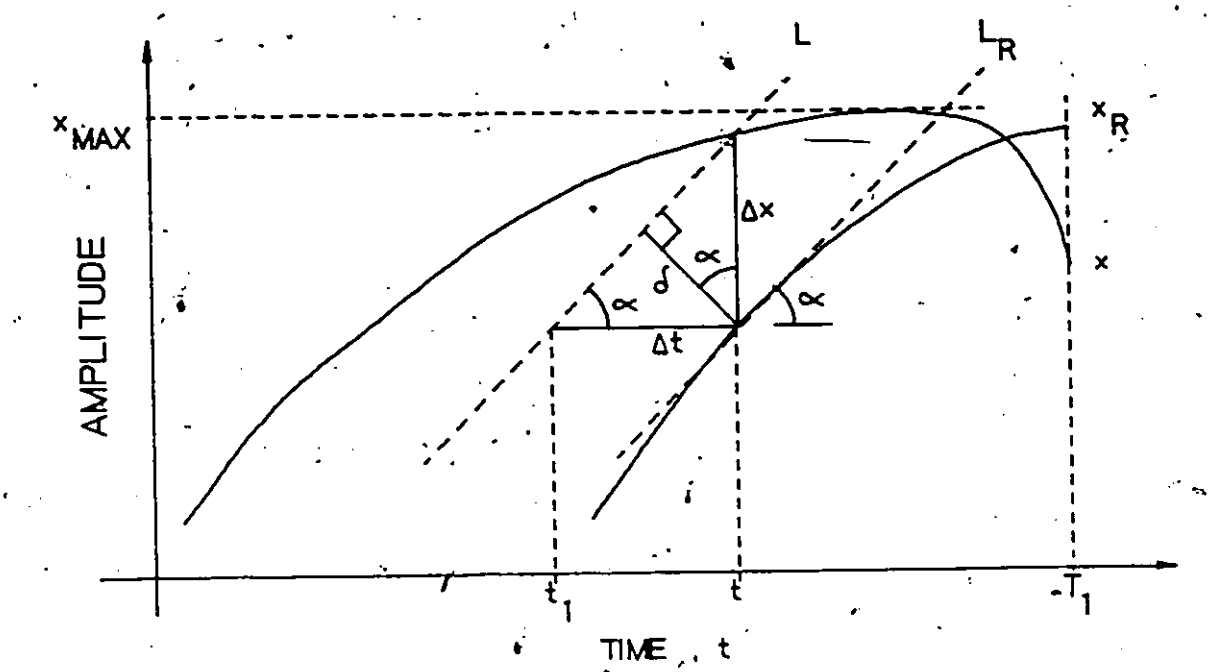
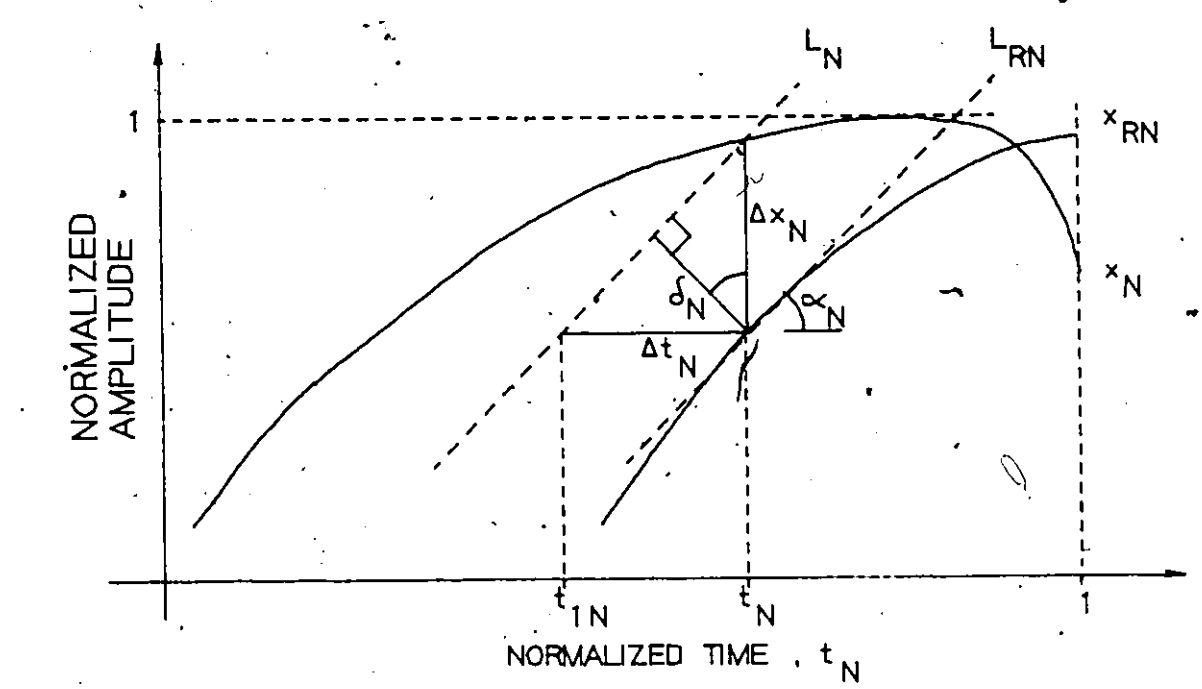


Figure 14 : Continuous-time subsystem circuit diagram of System C



(a)



(b)

Figure 15 : Normal distance approximation : (a) nominal, (b) normalized

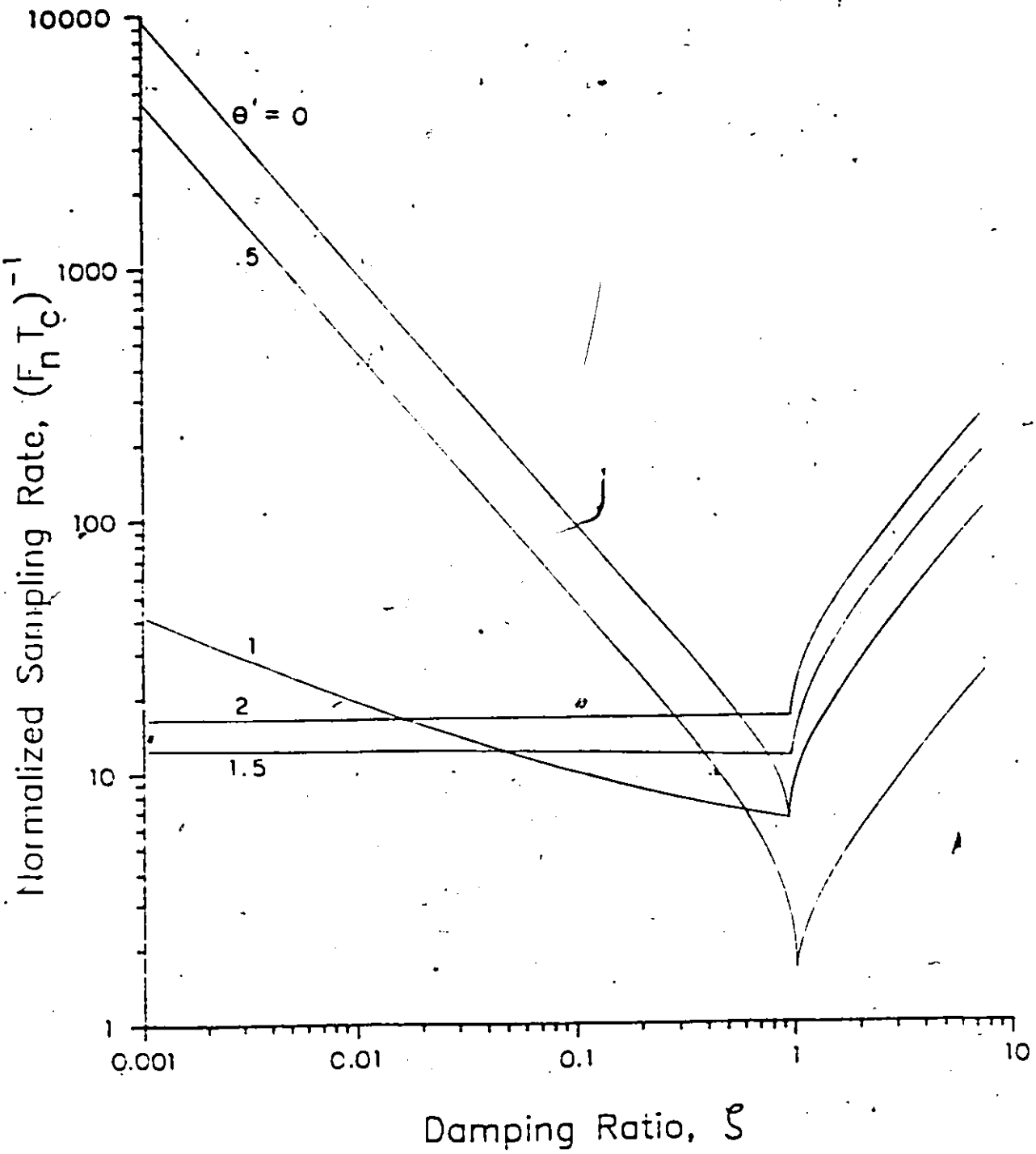


Figure 16 : DTESD model (System D) stability boundaries for θ'=0, .5, 1, 1.5, 2

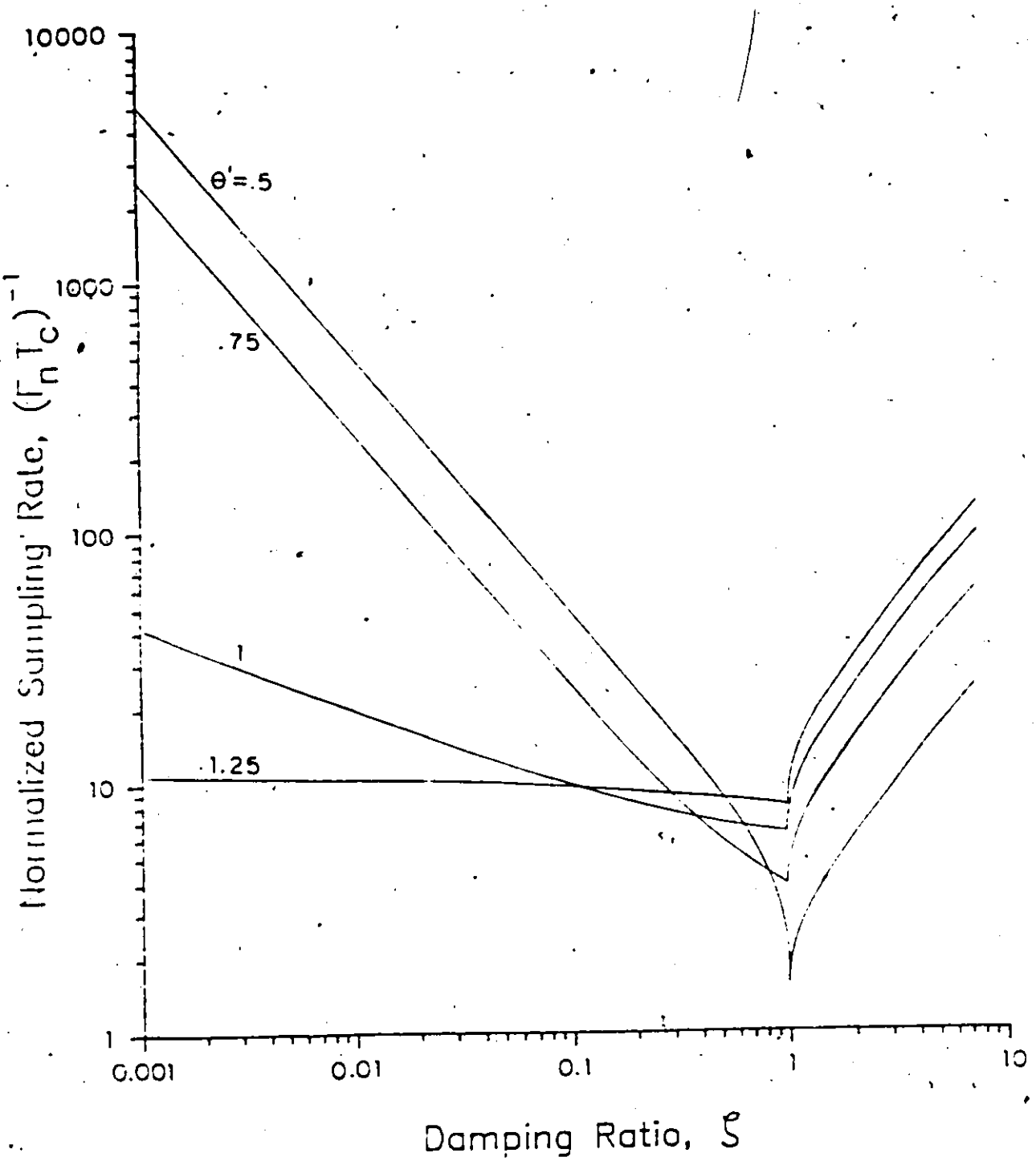


Figure 17 : DTESD model (System D) stability boundaries for θ'=.5, .75, 1, 1.25

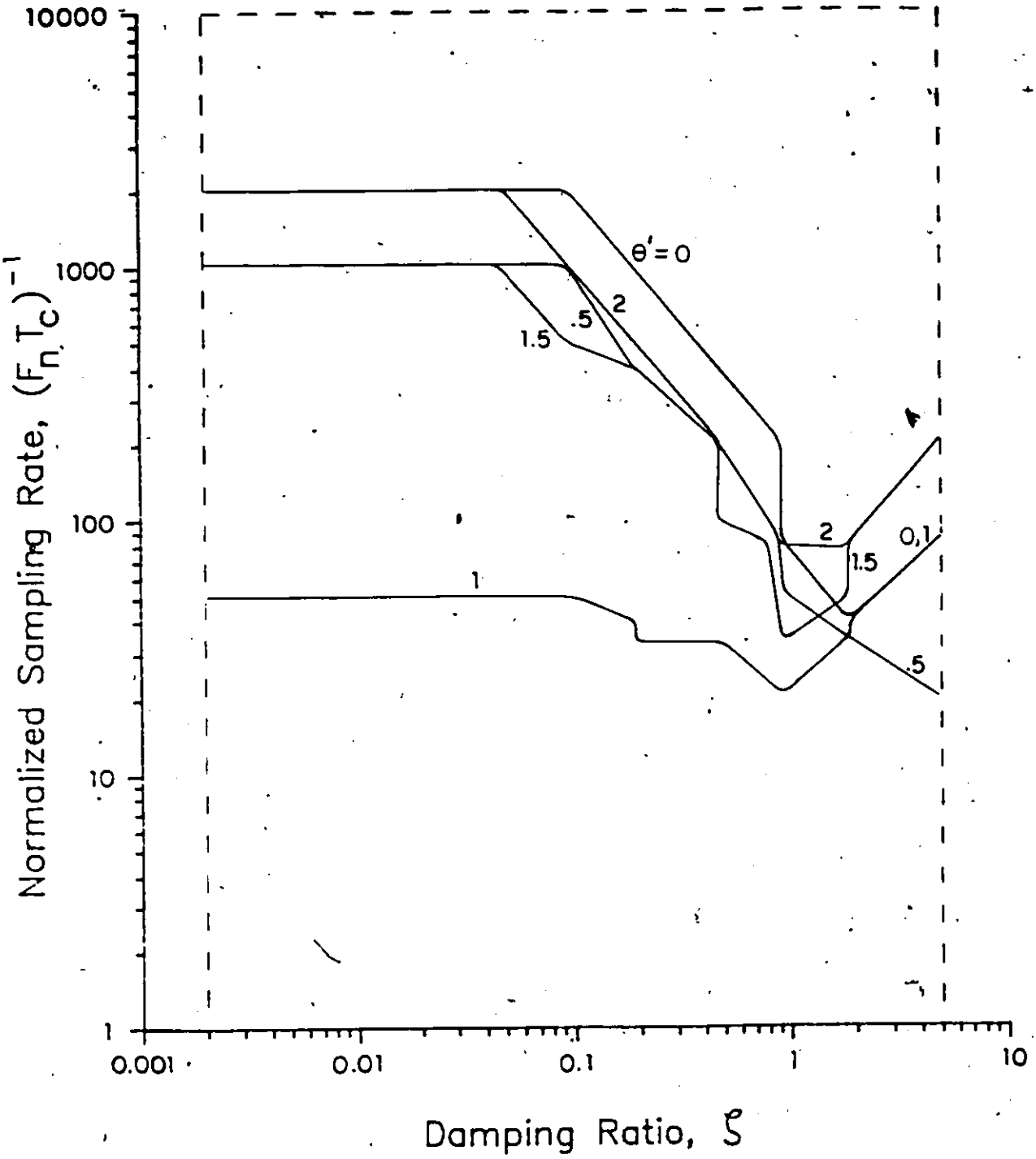


Figure 18 : System D accuracy boundaries (ε_N=.0075) for θ'=0, .5, 1, 1.5, 2

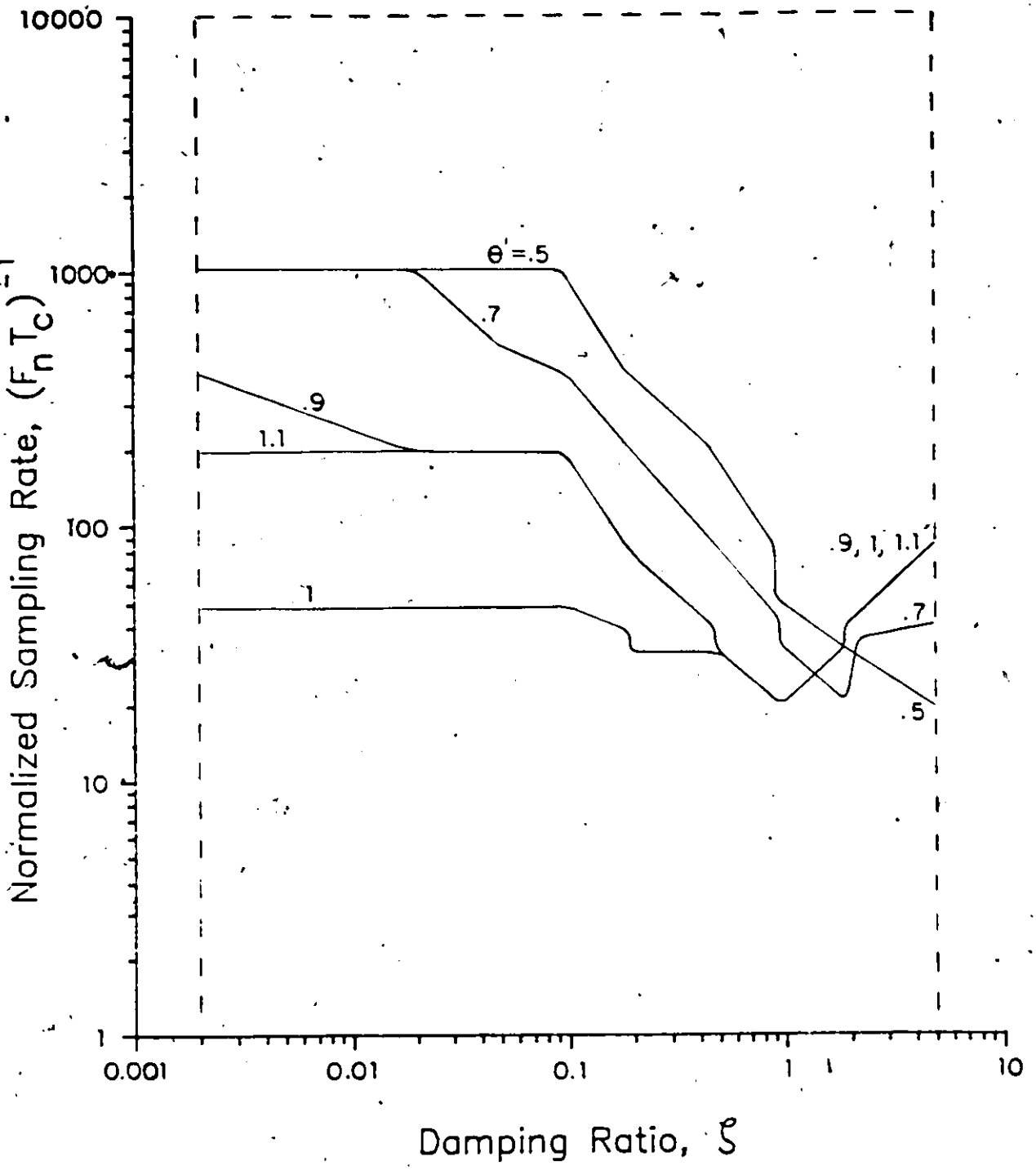


Figure 19 : System D accuracy boundaries (ε_N=.0075) for θ'=.5, .7, .9, 1, 1.1

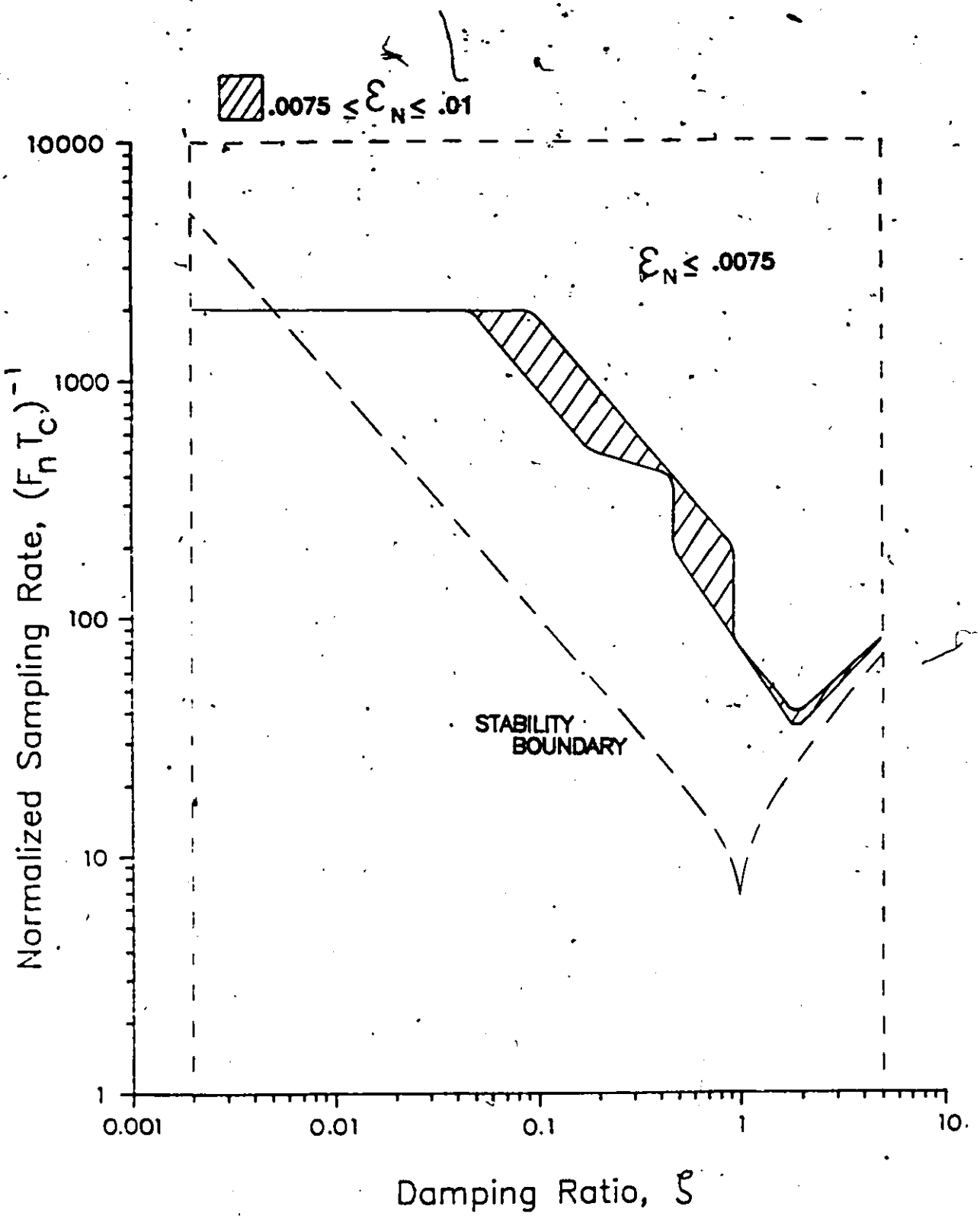


Figure 20 : System D accuracy regions for $\theta'=0$

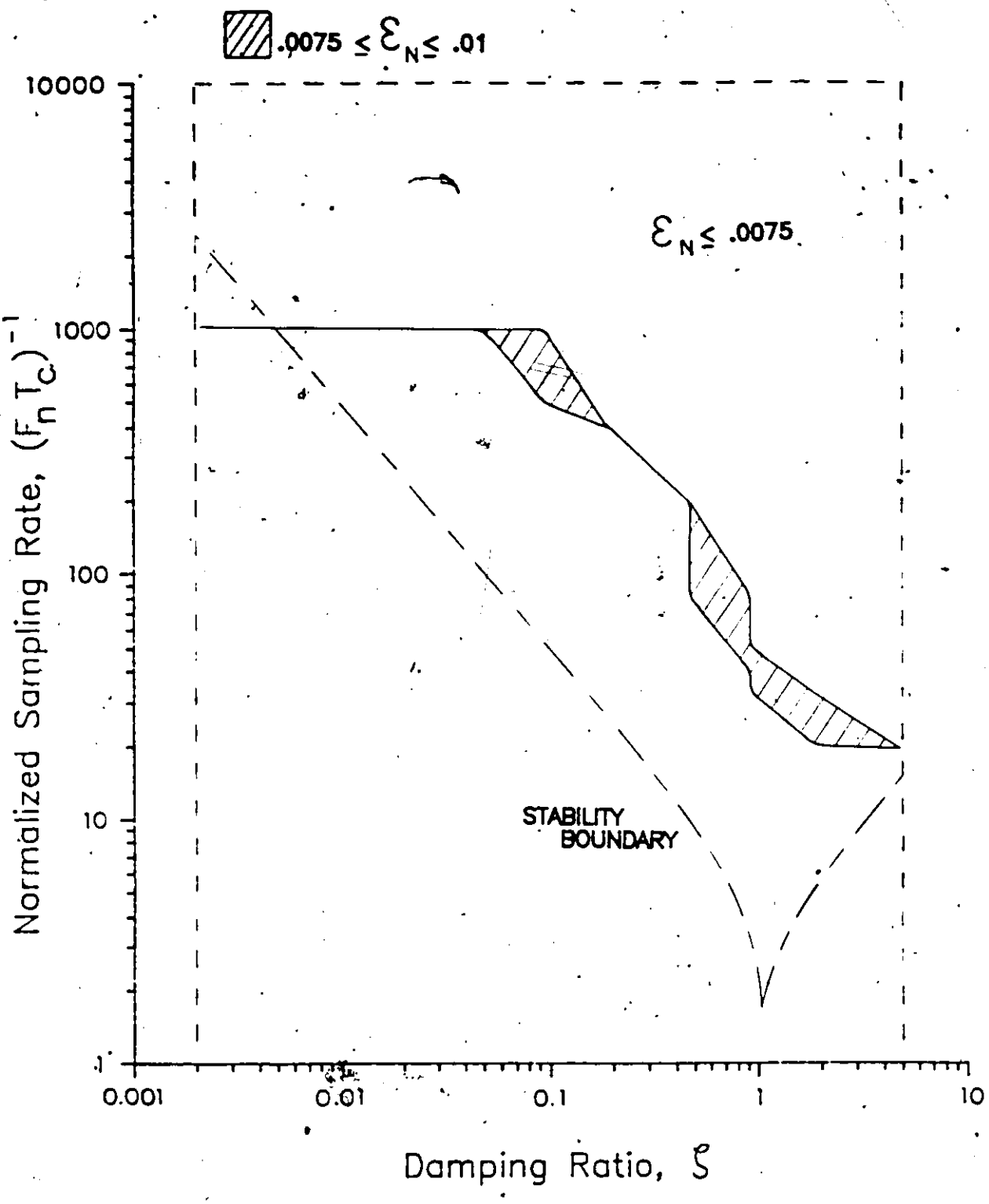


Figure 21 : System D accuracy regions for $\theta'=.5$

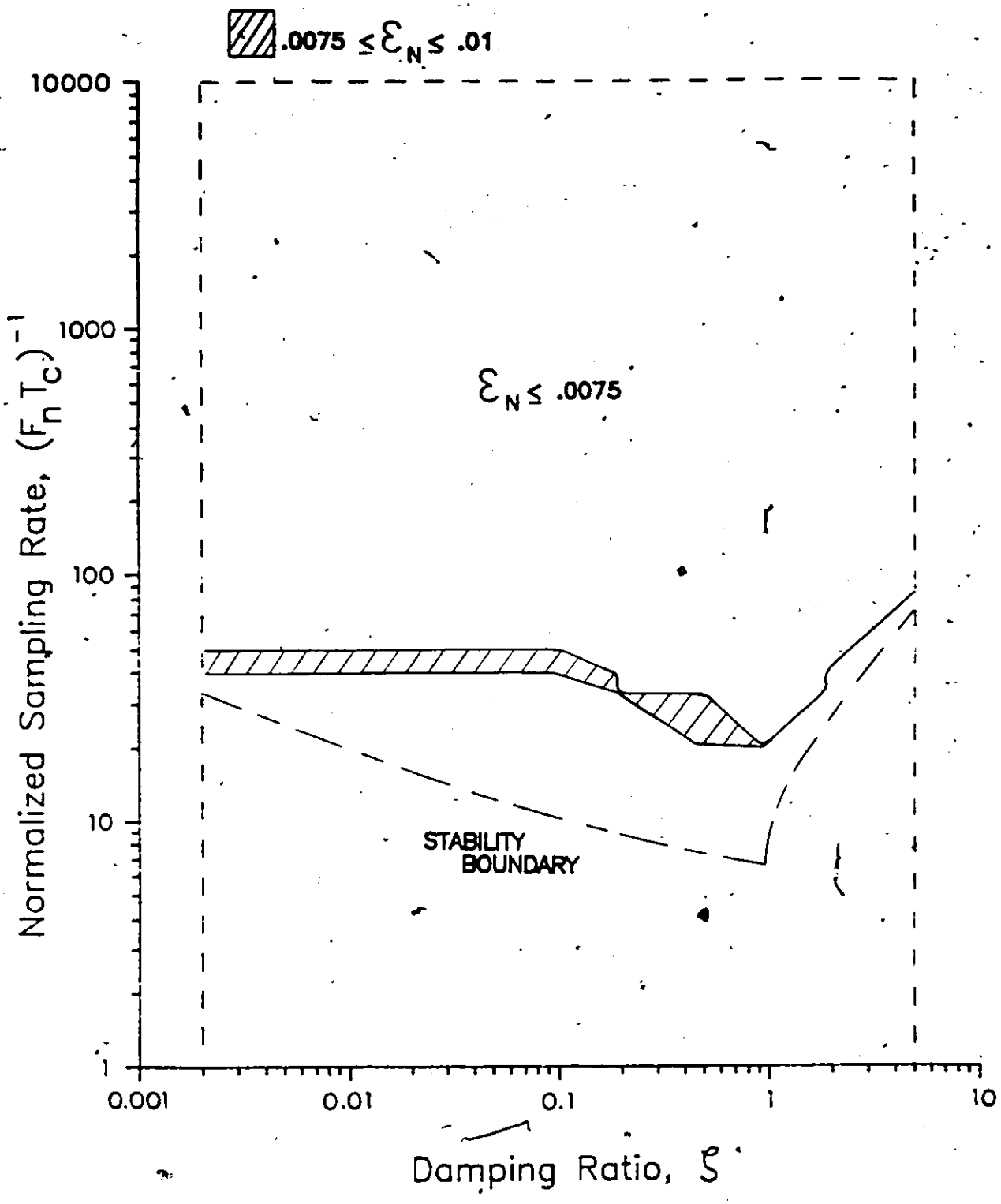


Figure 22 : System D accuracy regions for $\theta'=1$

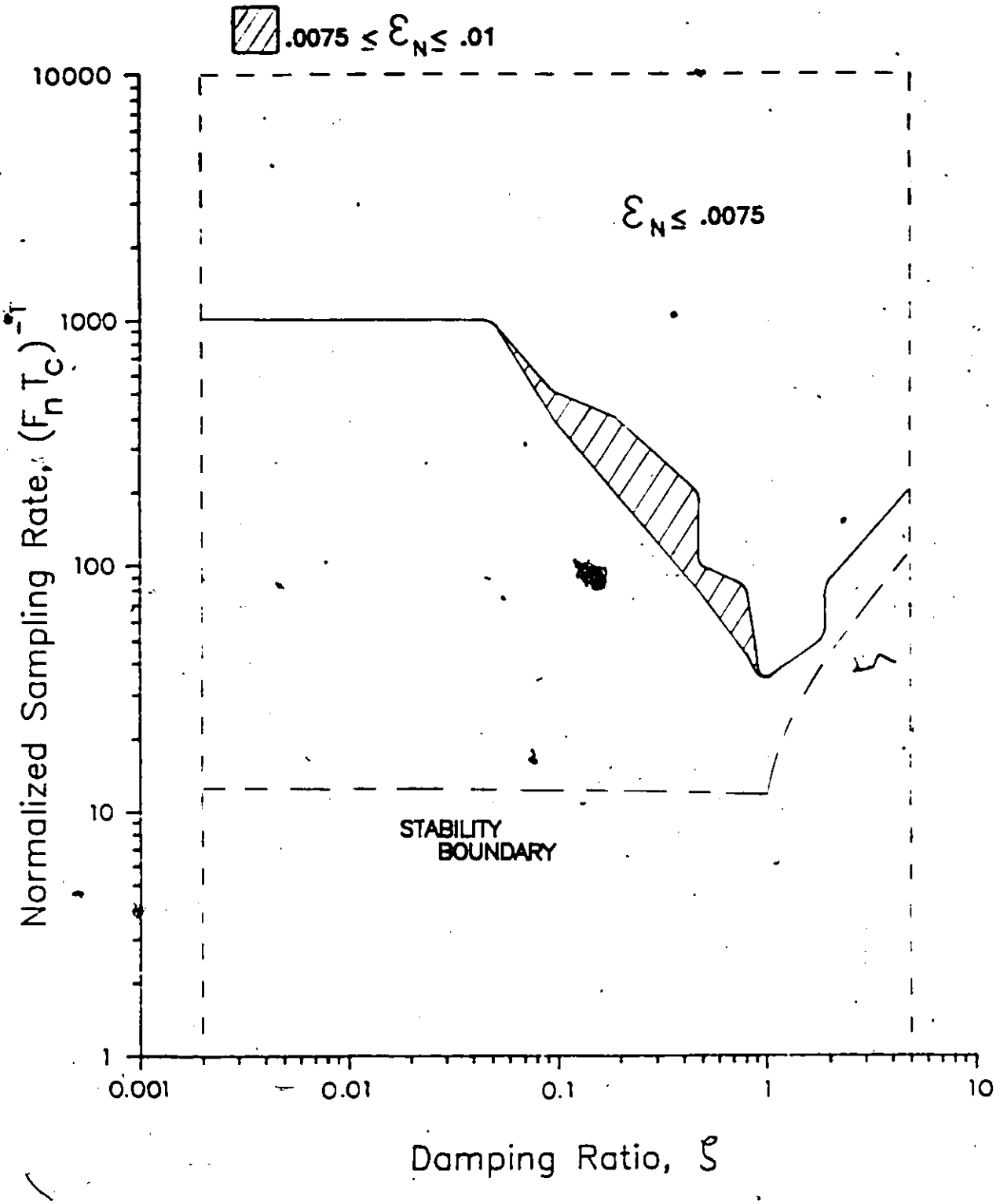


Figure 23 : System D accuracy regions for $\theta'=1.5$

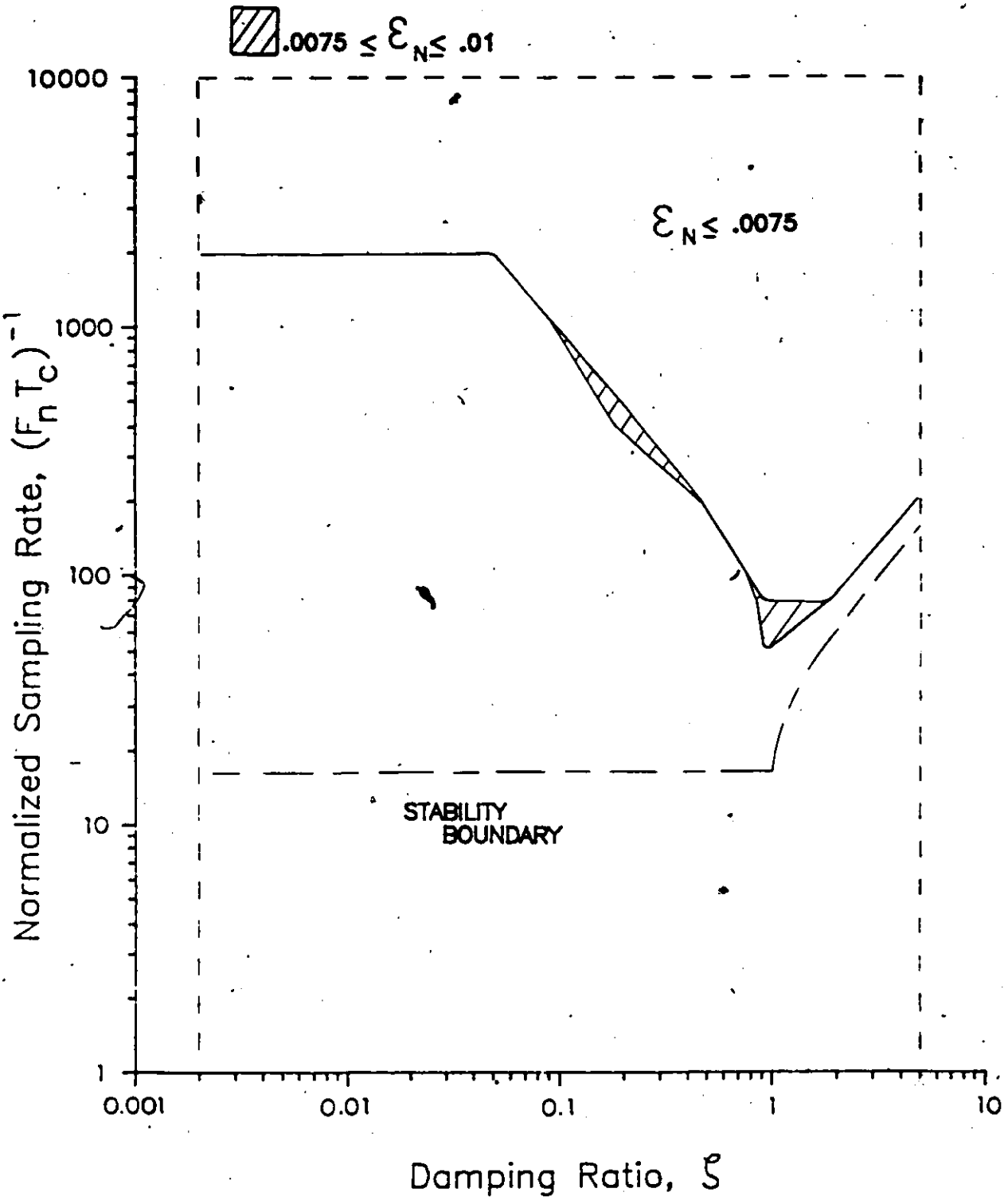


Figure 24 : System D accuracy regions for $\theta'=2$

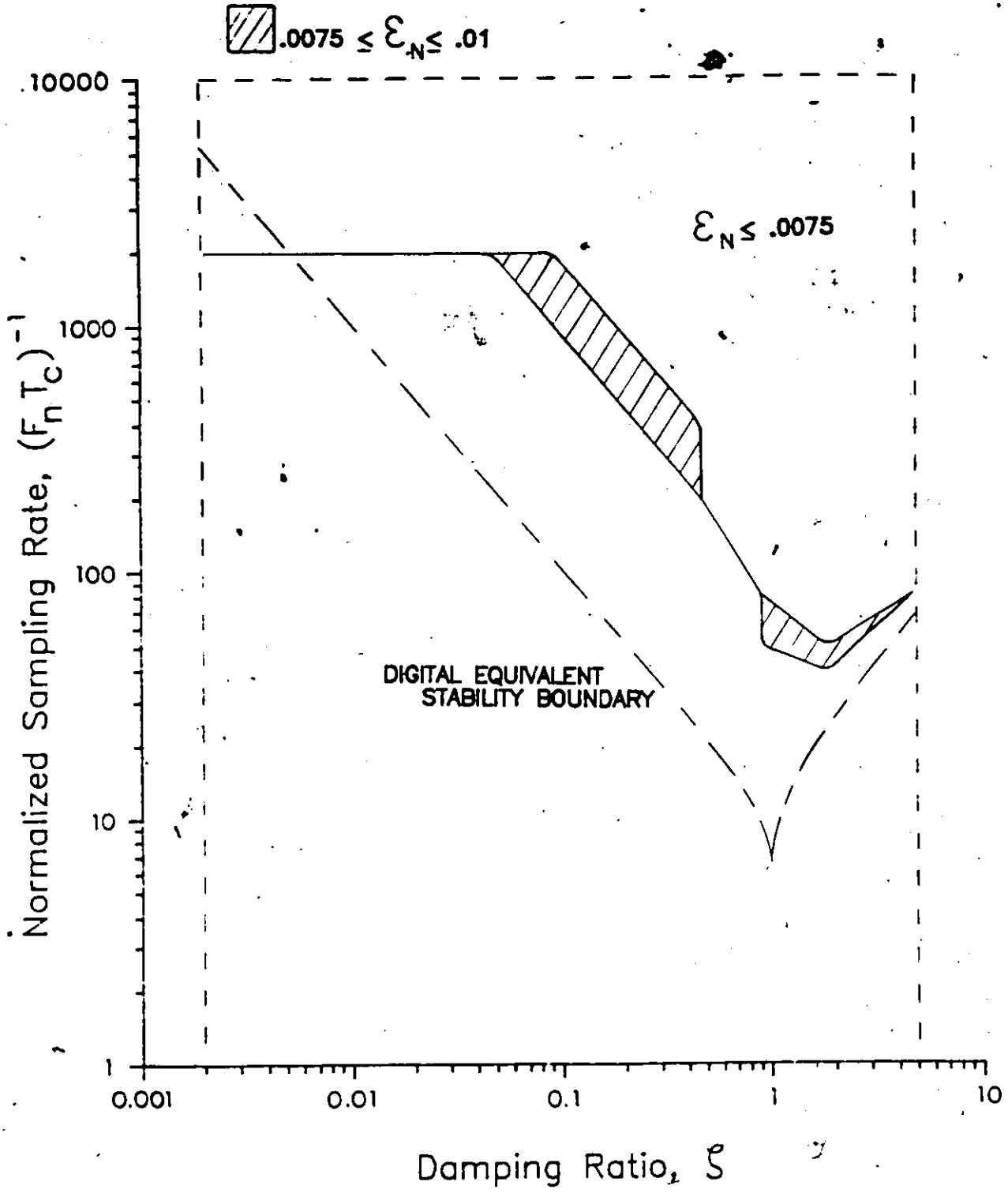


Figure 25 : System G accuracy regions for $\theta'=0$

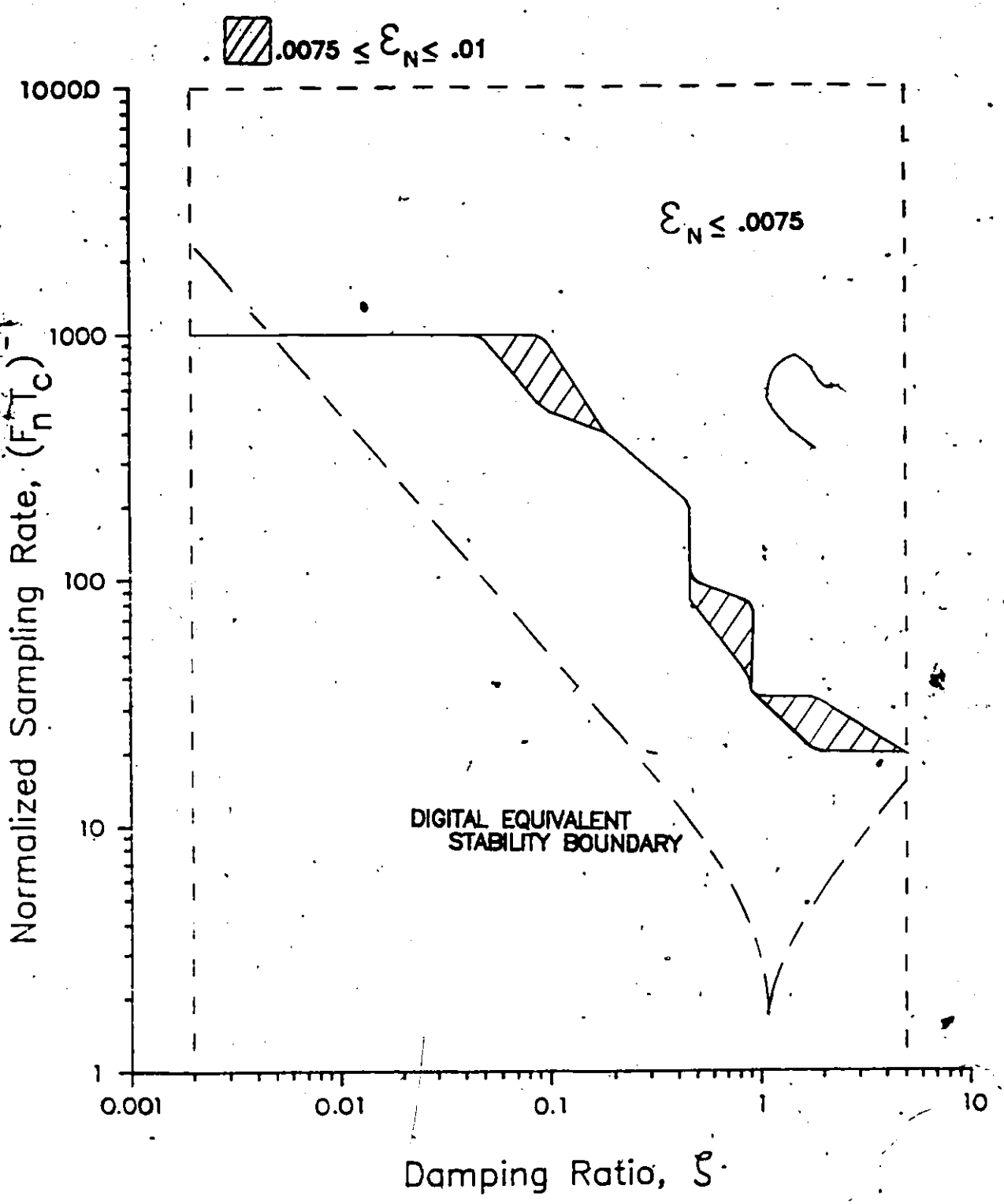


Figure 26 : System C accuracy regions for $\theta' = .5$

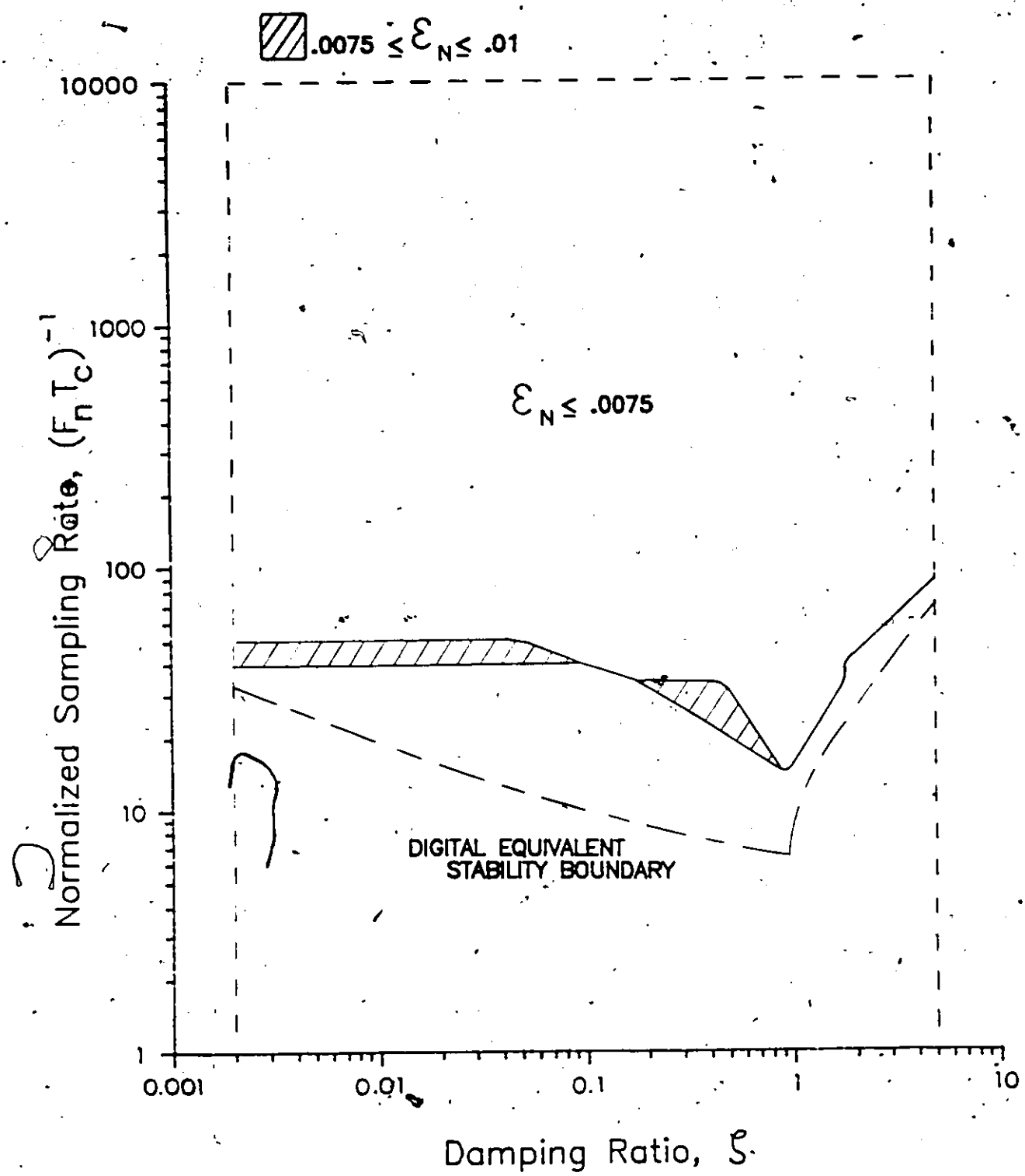


Figure 27 : System C accuracy regions for θ'=1

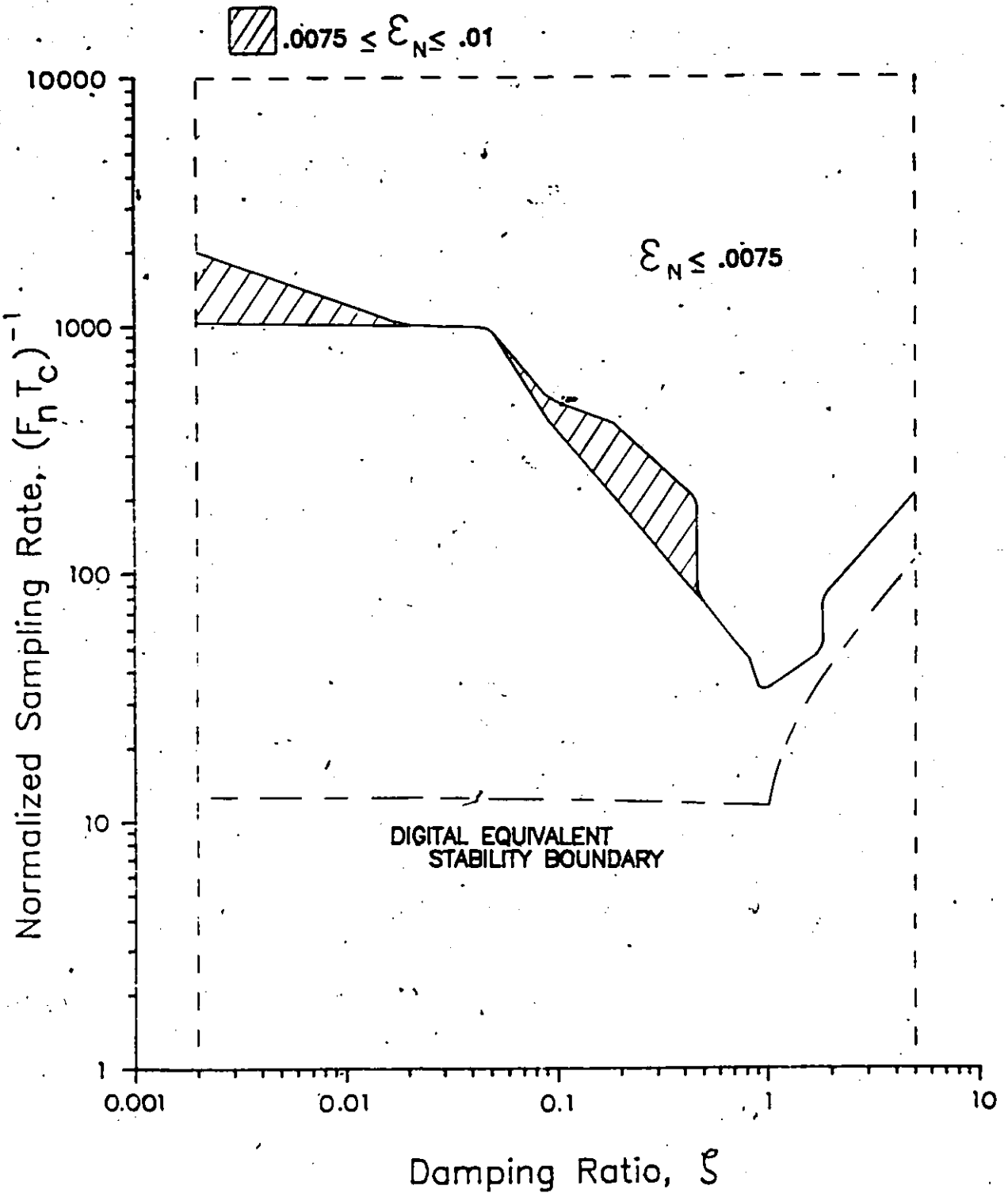


Figure 28 : System C accuracy regions for $\theta'=1.5$

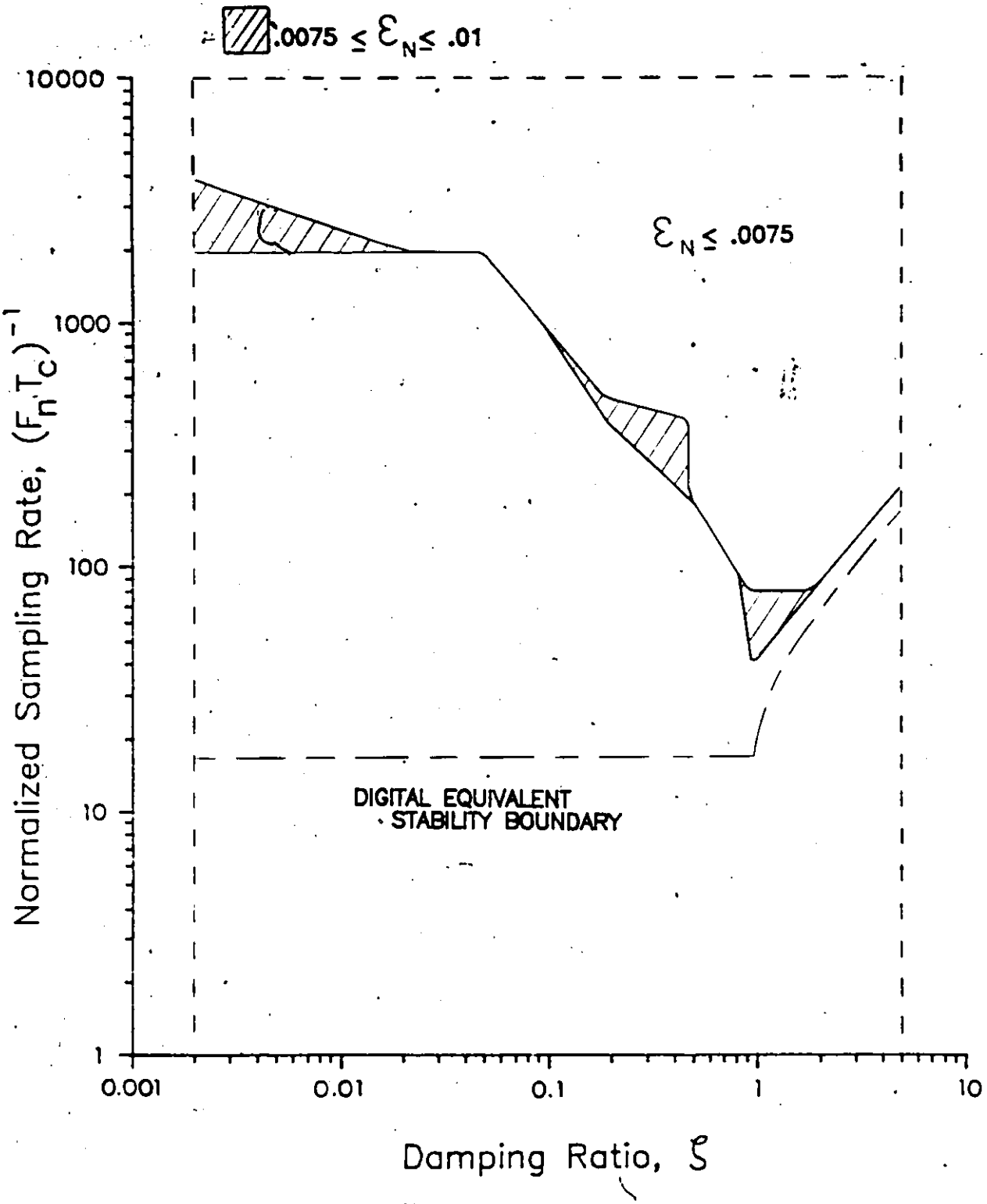


Figure 29 : System C accuracy regions for $\theta'=2$

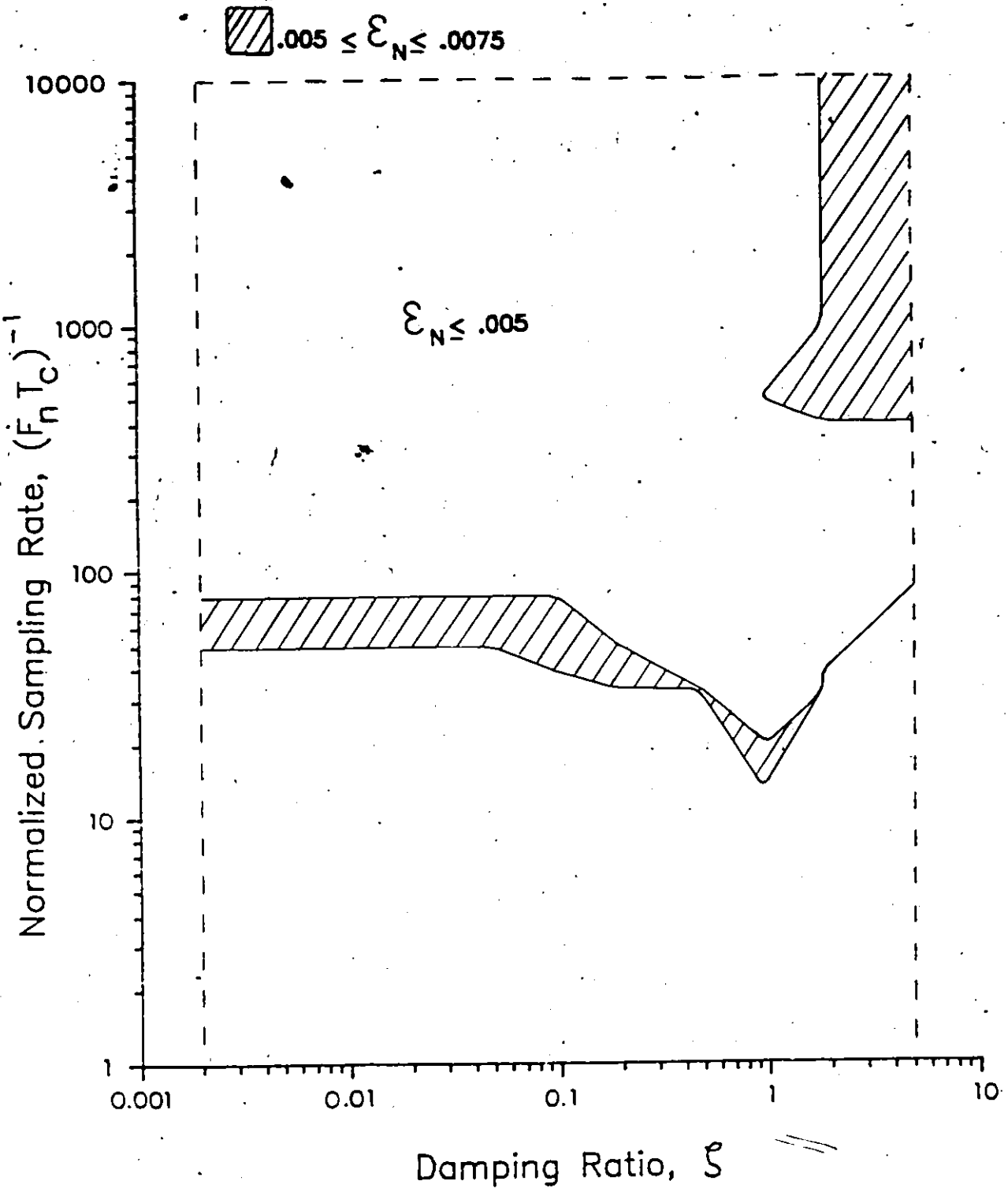


Figure 30 : System C accuracy regions for $\theta'=1$ and lower acceptable error values

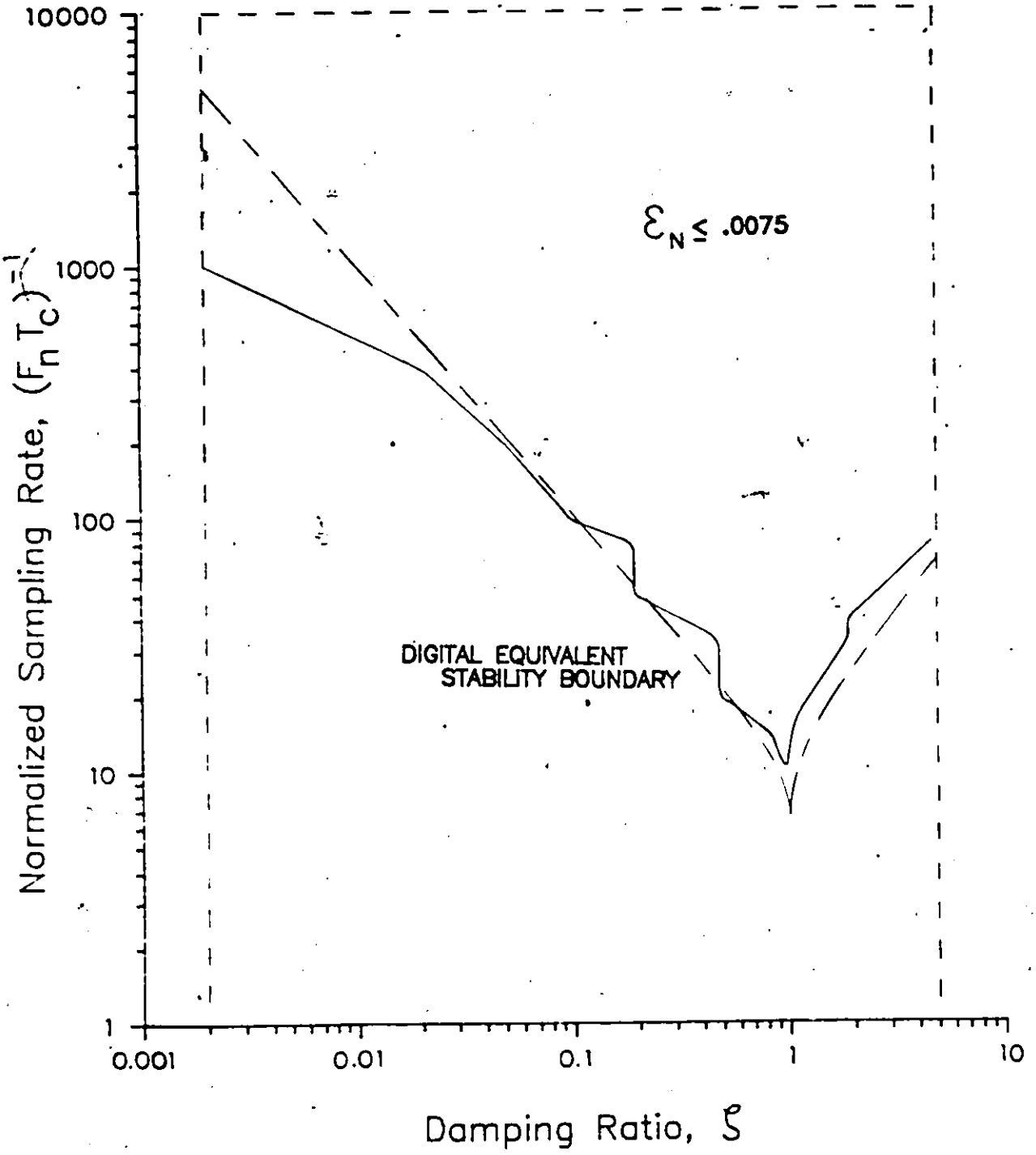


Figure 31 : Equivalence regions between Systems C and D for $\theta'=0$

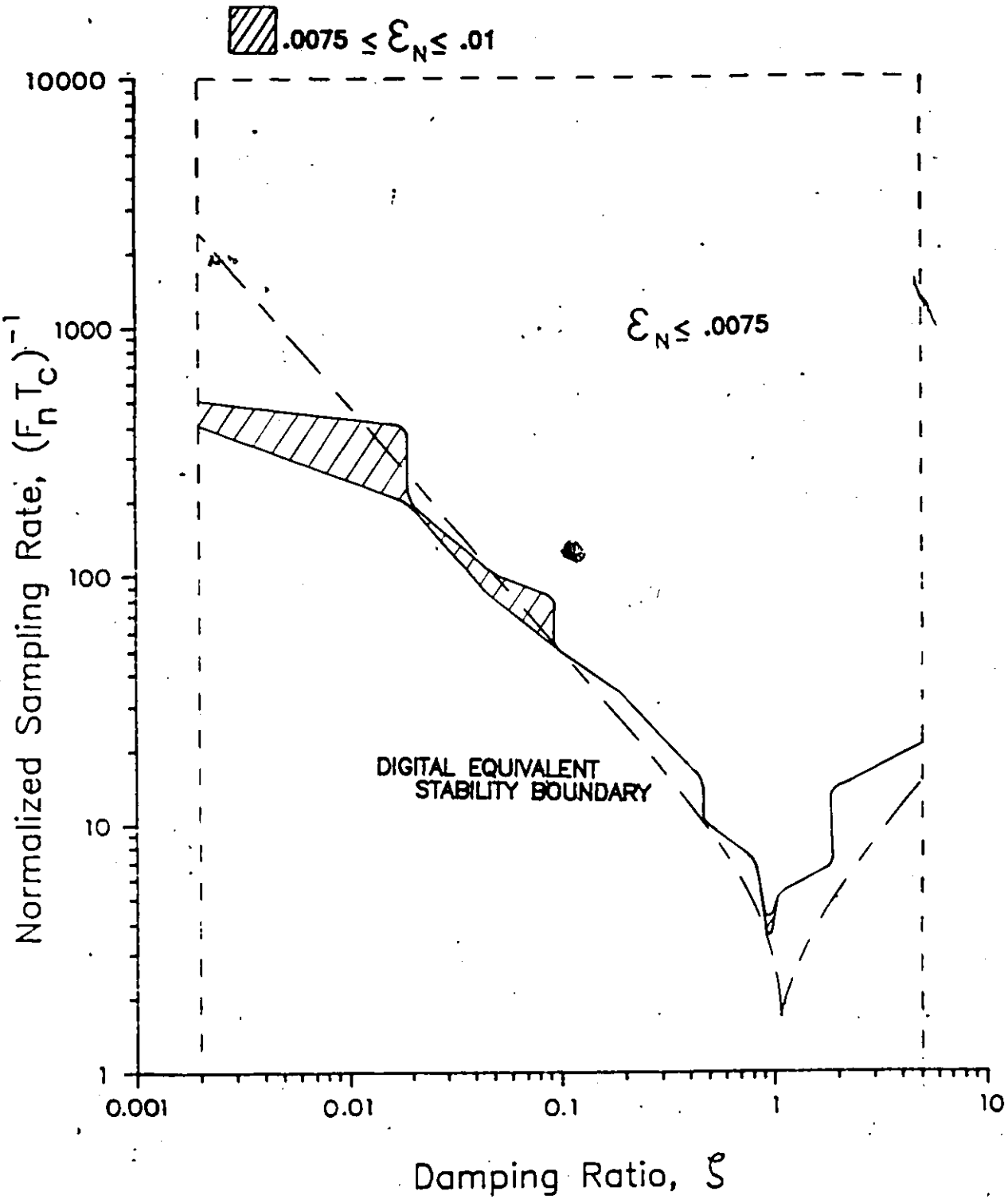


Figure 32 : Equivalence regions between Systems C and D for θ'=.5

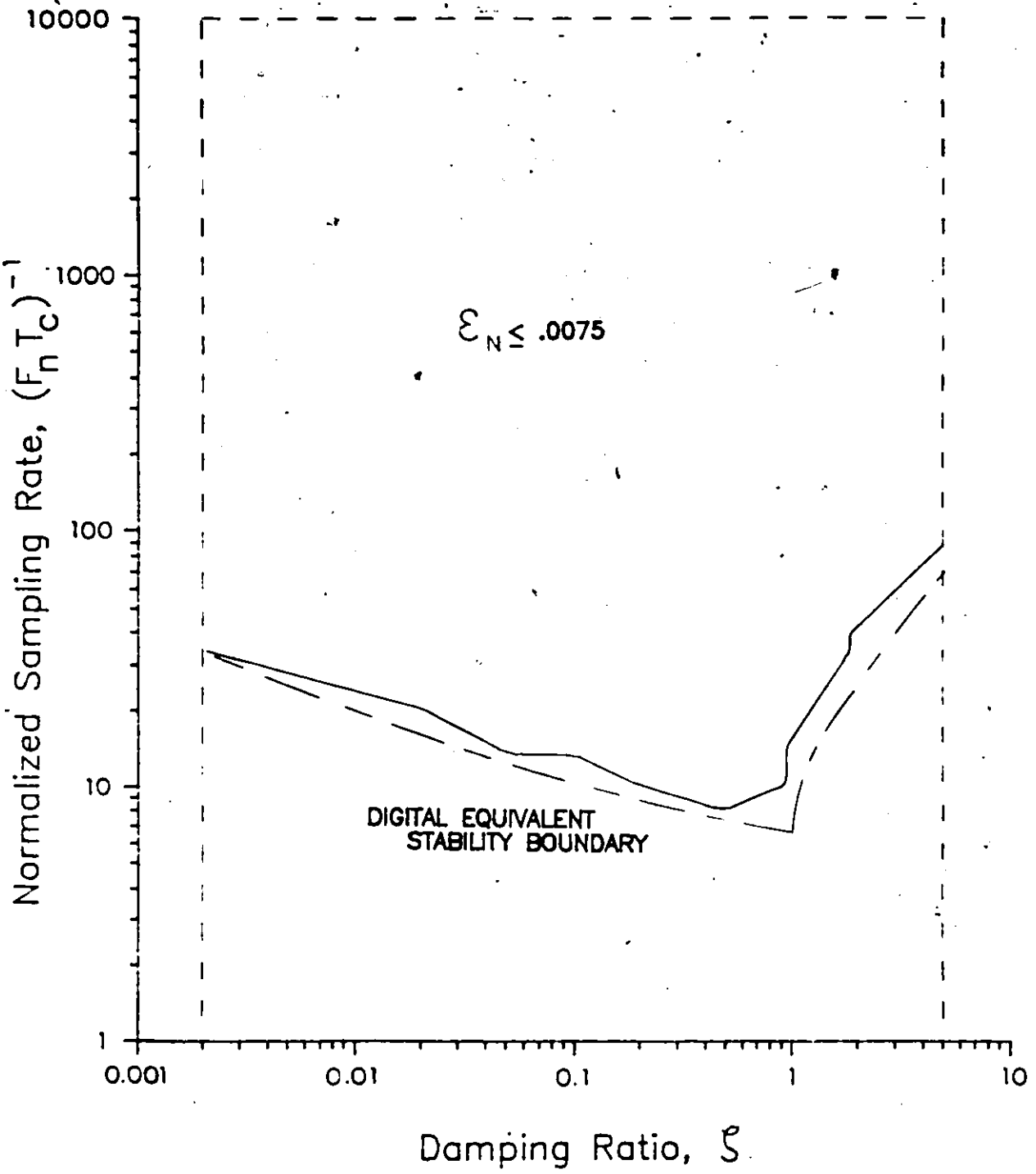


Figure 33 : Equivalence regions between Systems C and D for $\theta'=1$

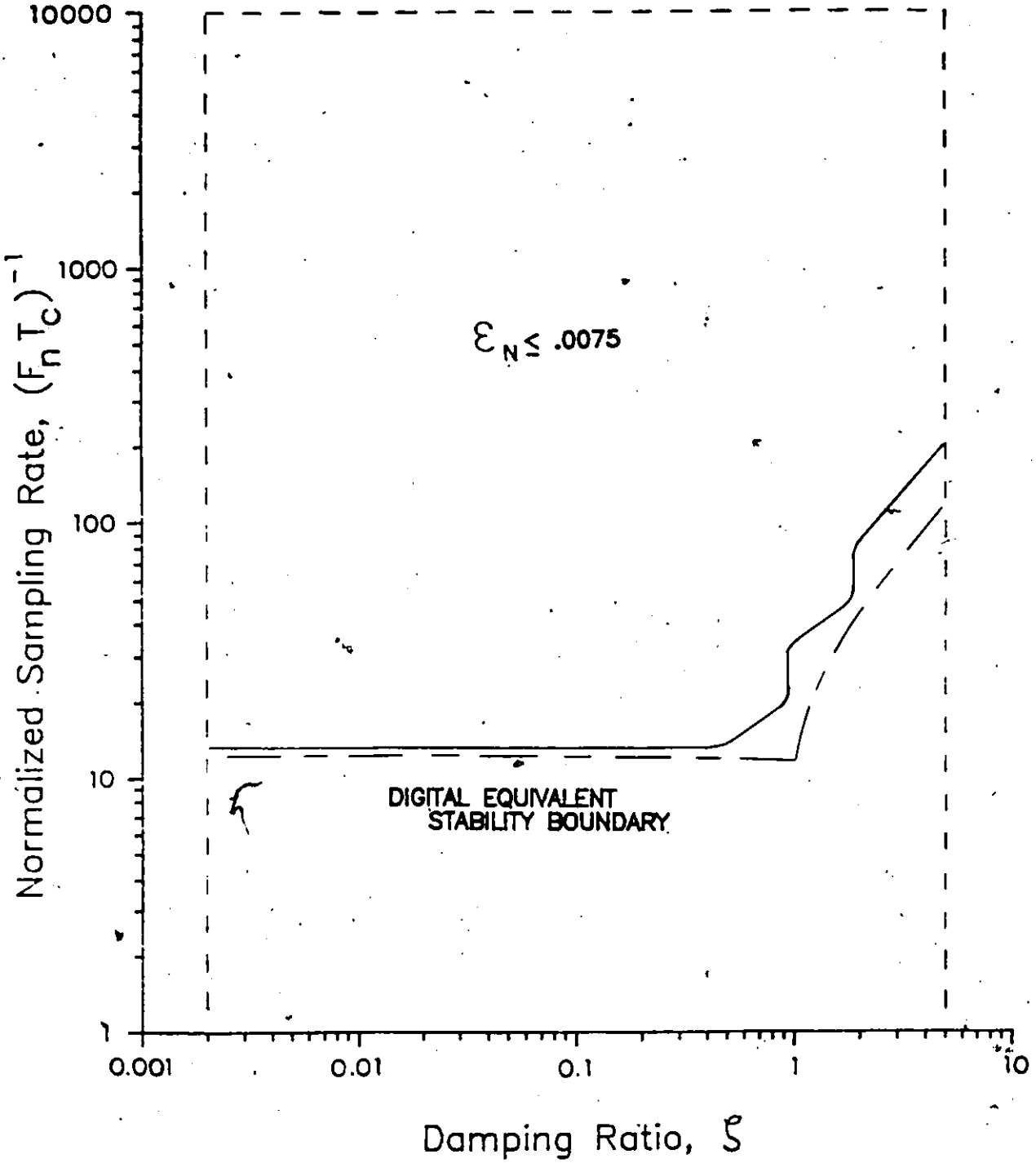


Figure 34 : Equivalence regions between Systems C and D for $\theta'=1.5$

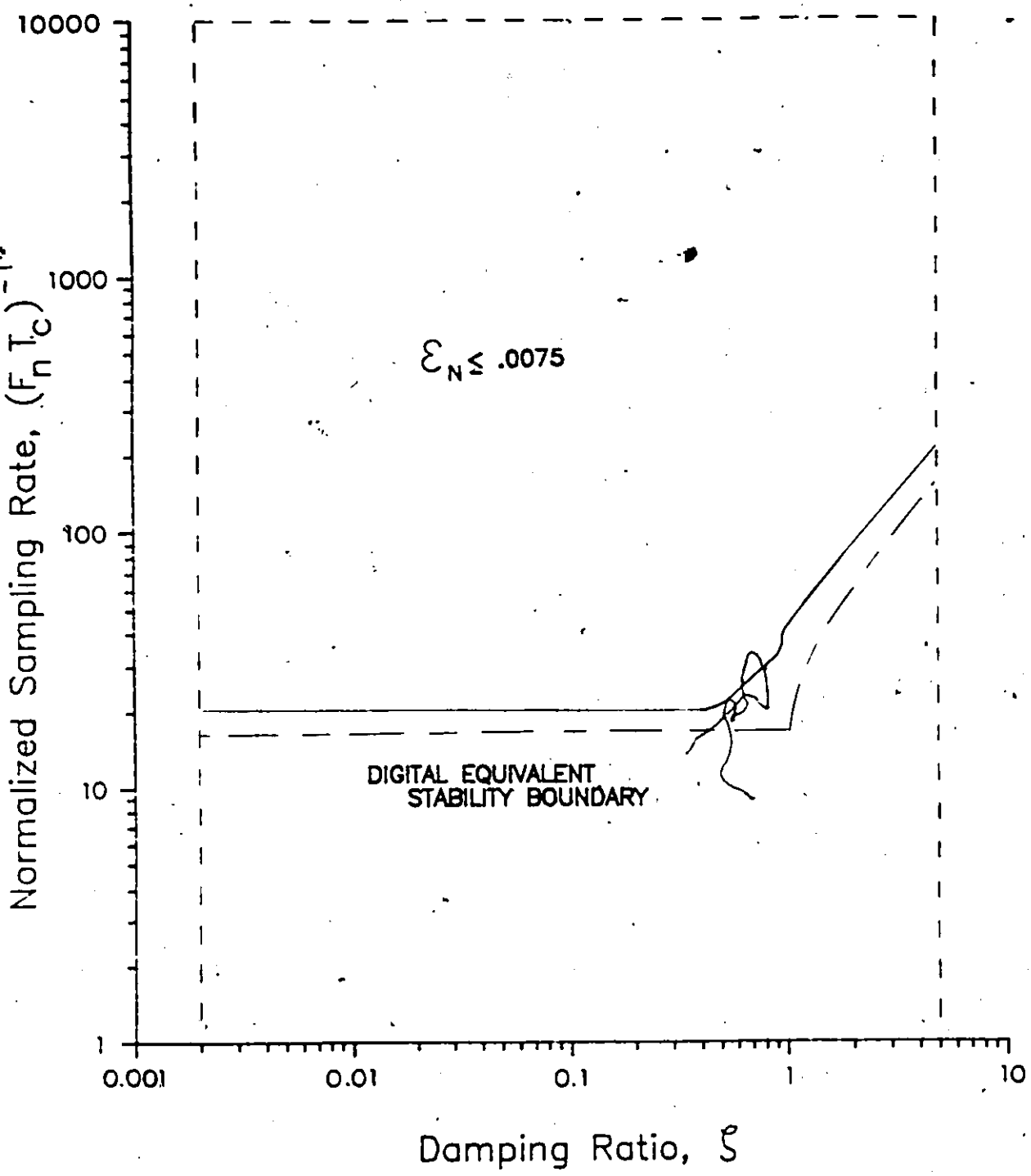


Figure 35 : Equivalence regions between Systems C and D for $\theta'=2$

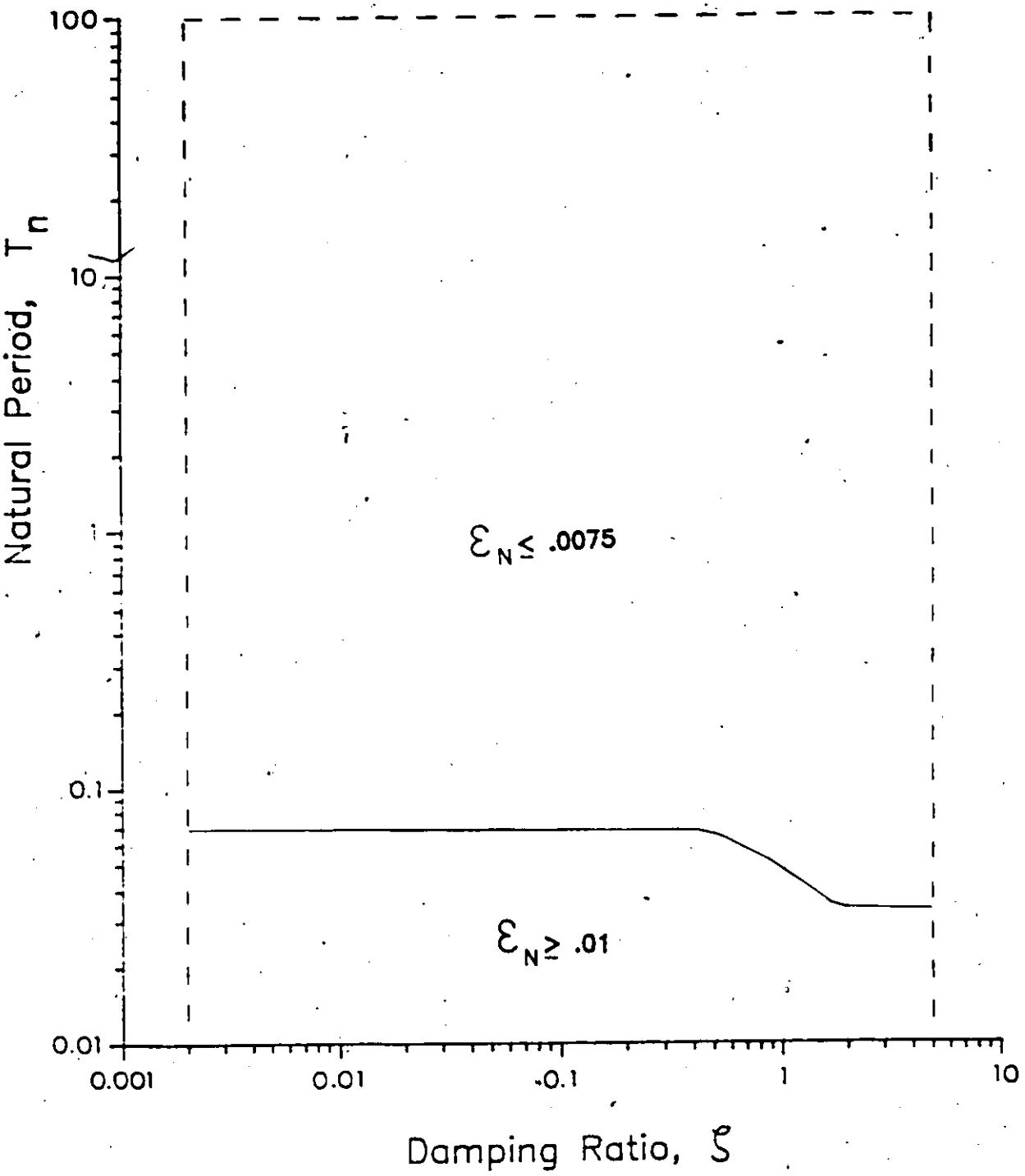


Figure 36 : System B accuracy regions

APPENDIX A

Normalized RMS Error Illustration Curves Obtained from SOLDE Analytical Solutions

NOTE : The undamped natural frequency of the analytical solution is 1.0 Hz in all cases.

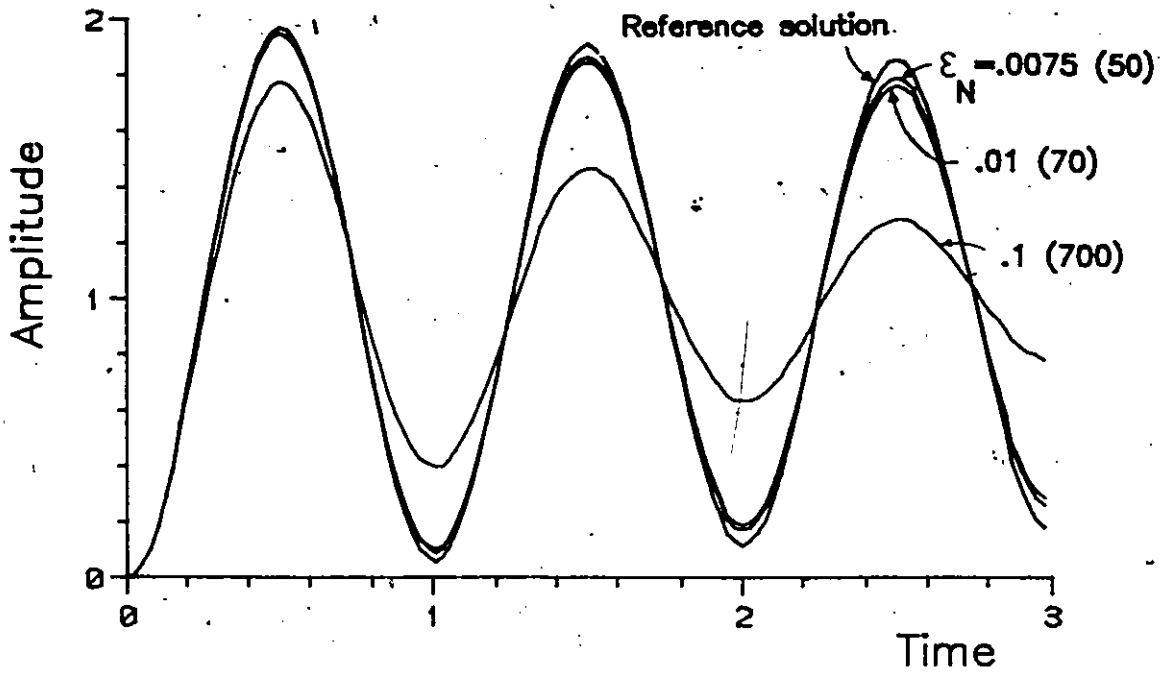


Figure A–1 : Normalized error with damping ratio of .01 for the reference solution
(Numbers in brackets are percentage variations in damping ratio)

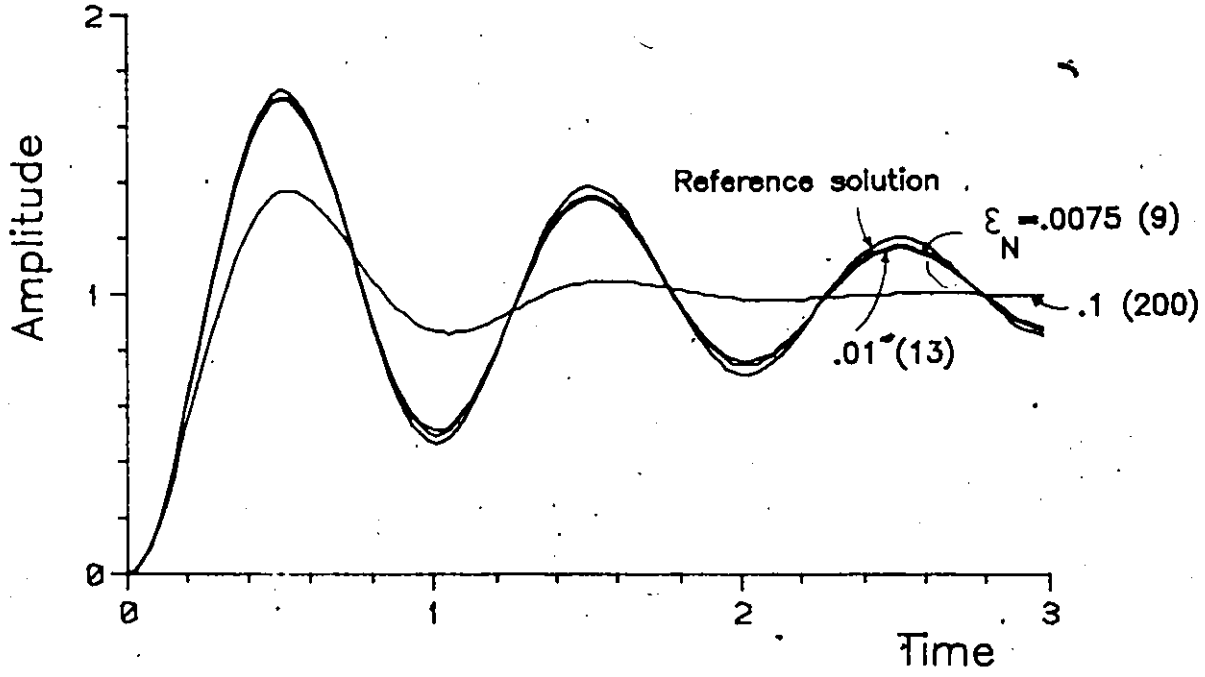


Figure A–2 : Normalized error with damping ratio of .1 for the reference solution
(Numbers in brackets are percentage variations in damping ratio)

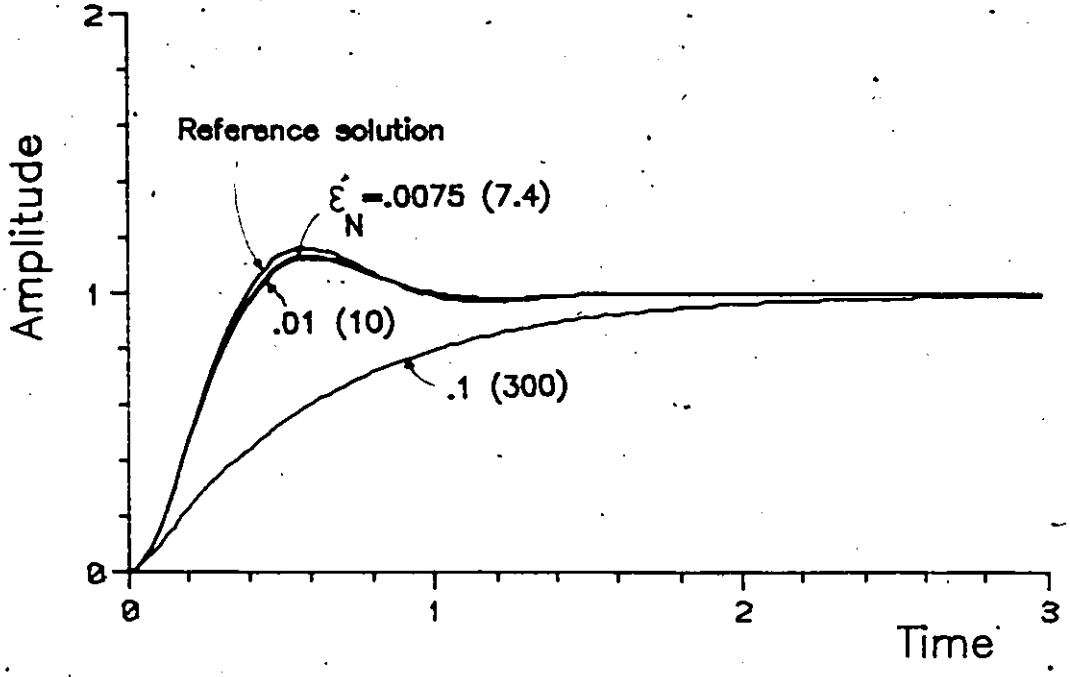


Figure A–3 : Normalized error with damping ratio of .5 for the reference solution
(Numbers in brackets are percentage variations in damping ratio)

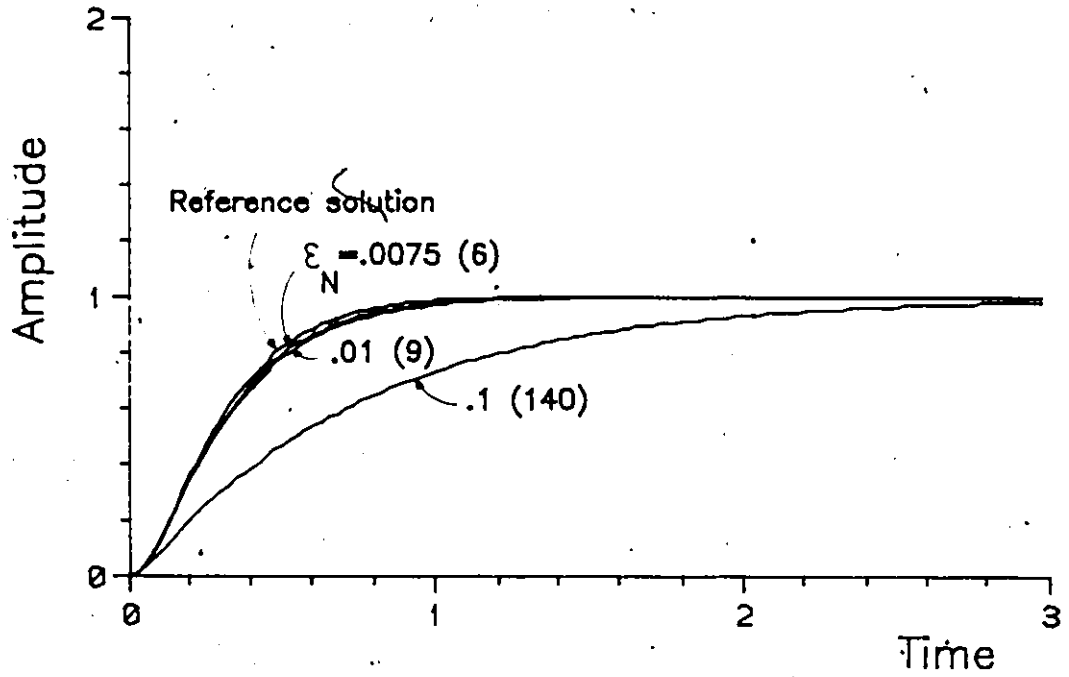


Figure A–4 : Normalized error with damping ratio of 1 for the reference solution
(Numbers in brackets are percentage variations in damping ratio)

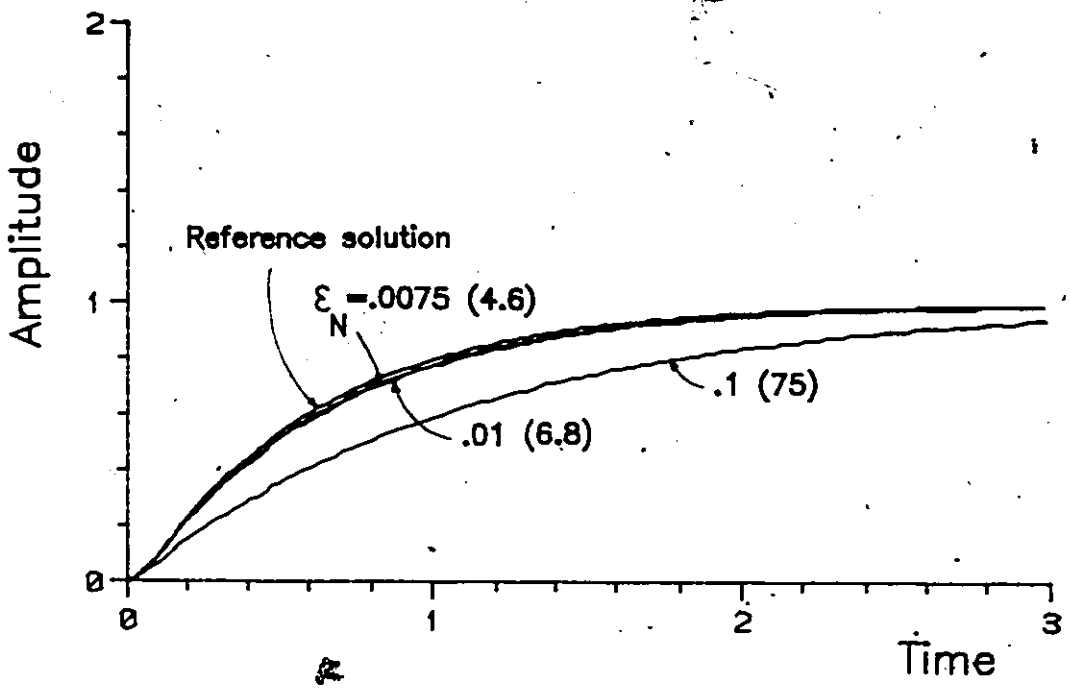


Figure A–5 : Normalized error with damping ratio of 2 for the reference solution
(Numbers in brackets are percentage variations in damping ratio)

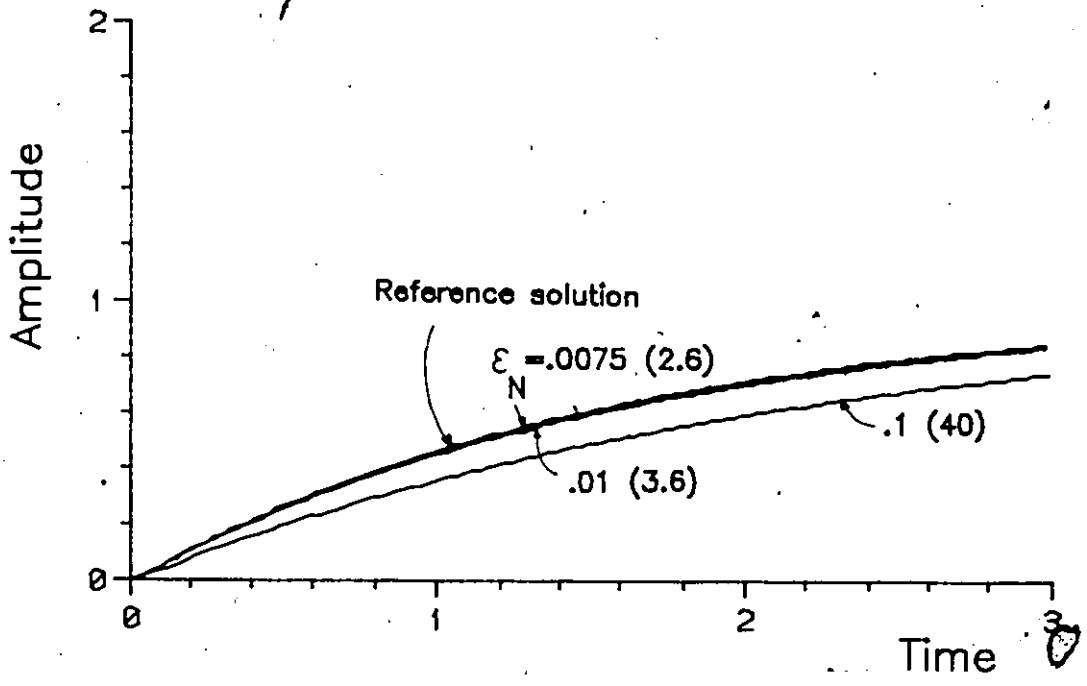


Figure A–6 : Normalized error with damping ratio of 5 for the reference solution
(Numbers in brackets are percentage variations in damping ratio)

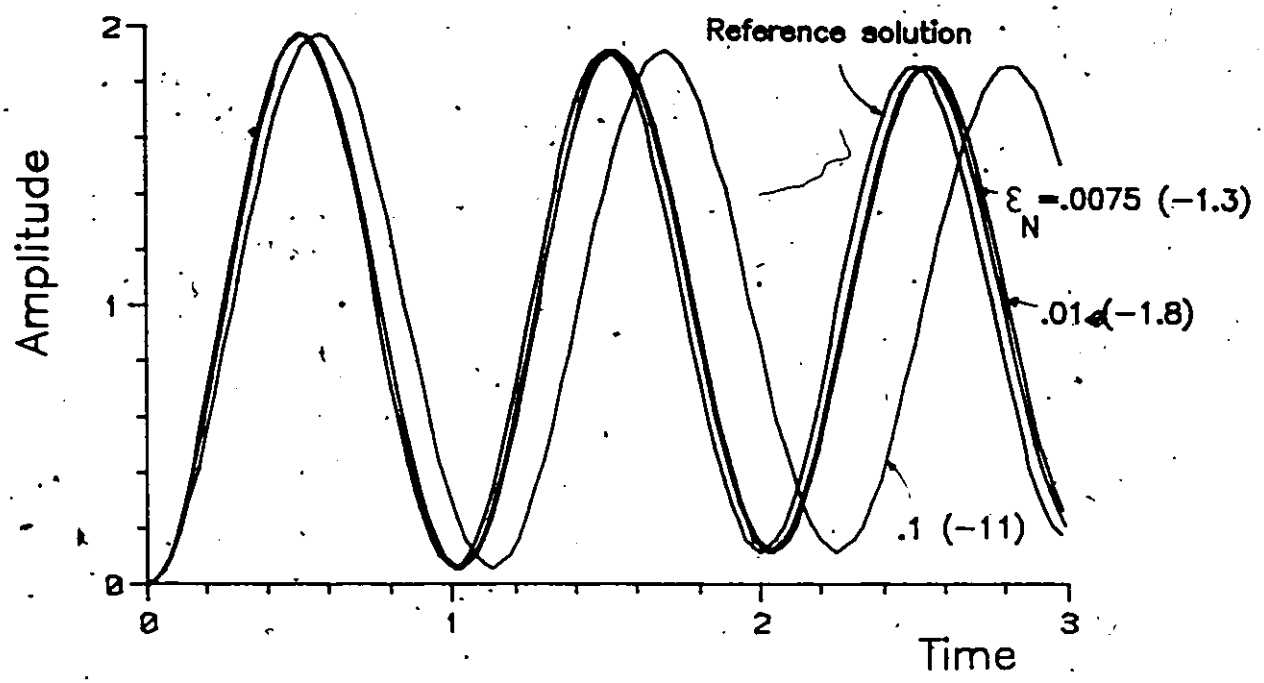


Figure A–7 : Normalized error with damping ratio of .01 for all solutions
(Numbers in brackets are percentage variations in undamped natural frequency)

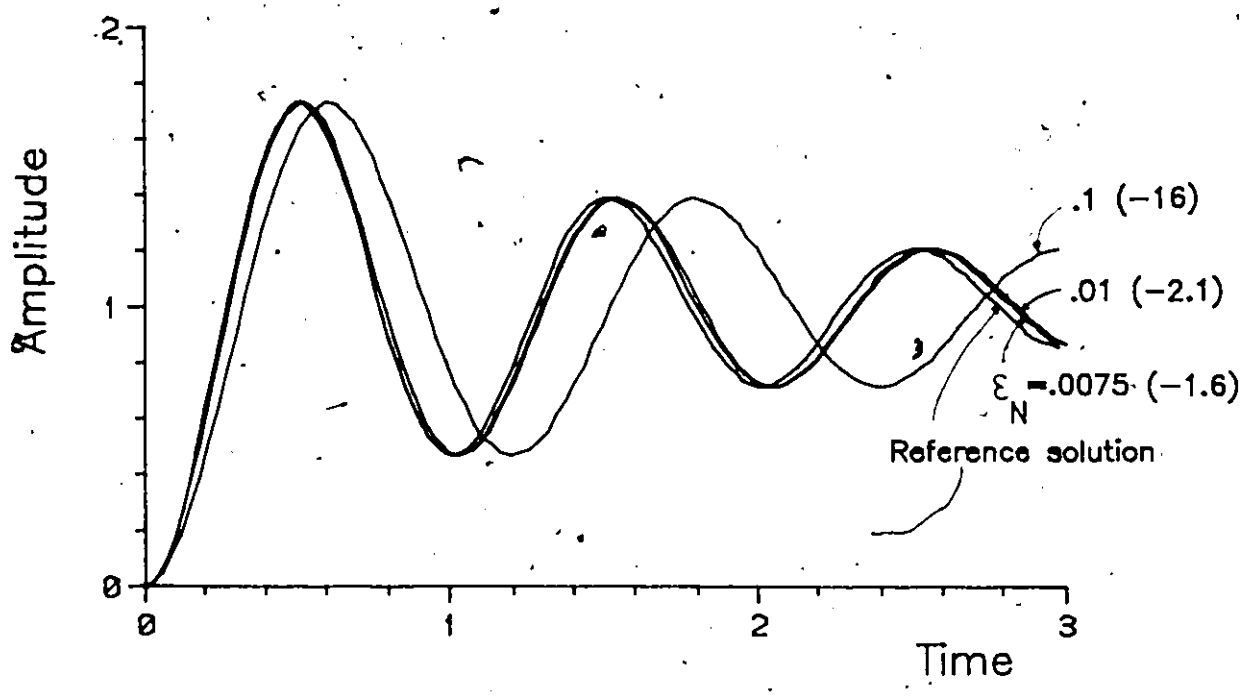


Figure A–8 : Normalized error with damping ratio of .1 for all solutions
(Numbers in brackets are percentage variations in undamped natural frequency)

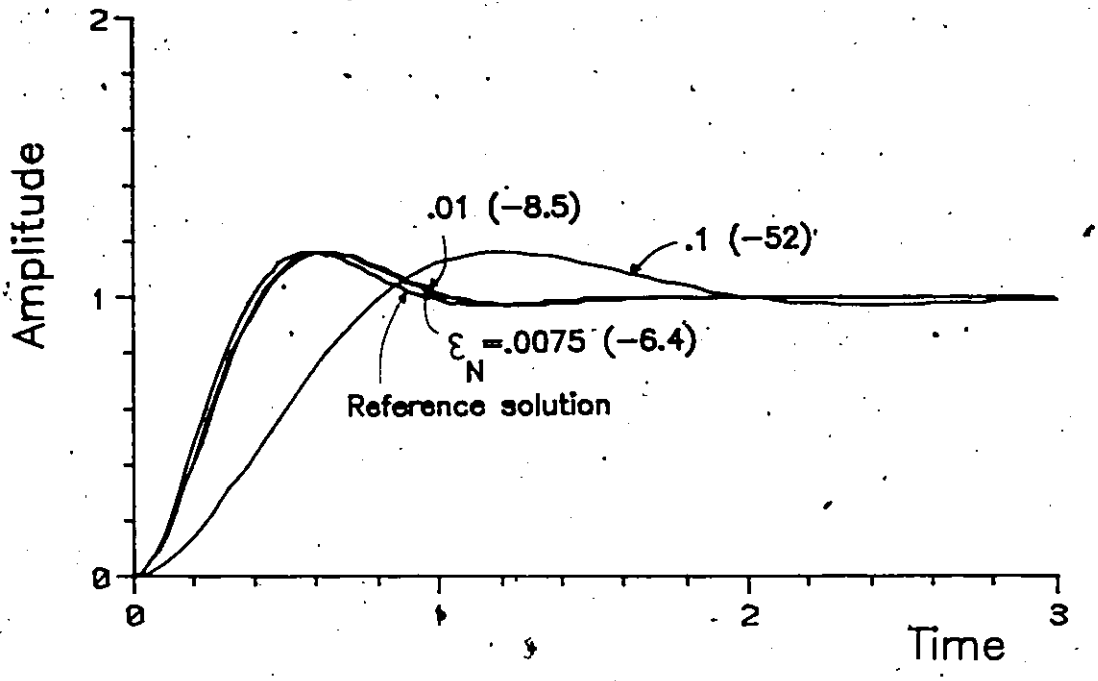


Figure A−9 : Normalized error with damping ratio of .5 for all solutions
(Numbers in brackets are percentage variations in undamped natural frequency)

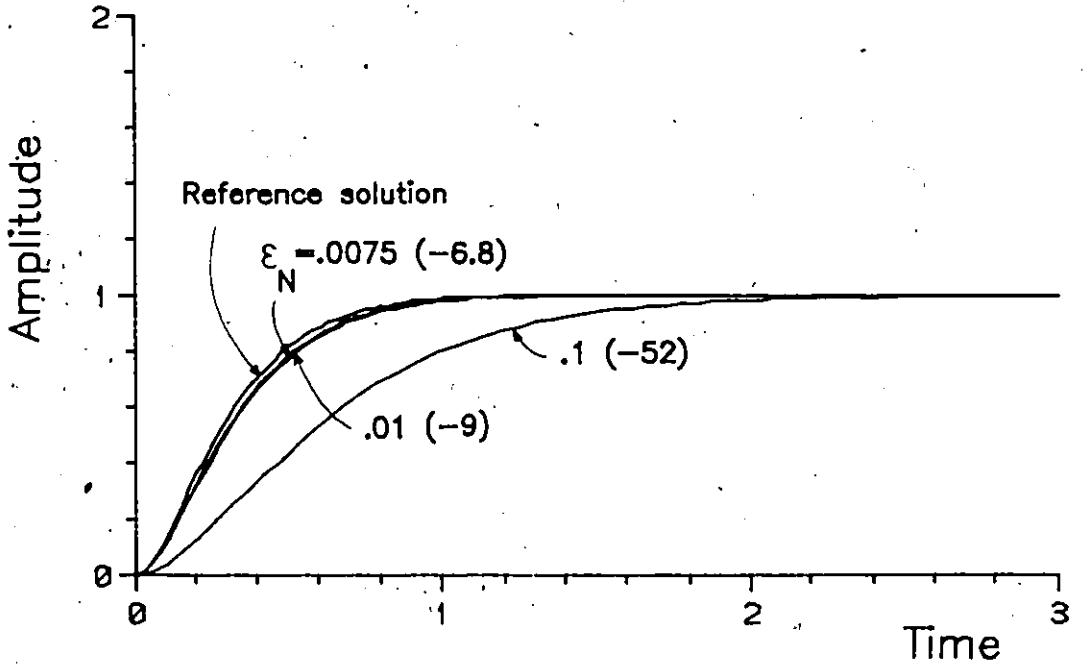


Figure A−10 : Normalized error with damping ratio of 1 for all solutions
(Numbers in brackets are percentage variations in undamped natural frequency)

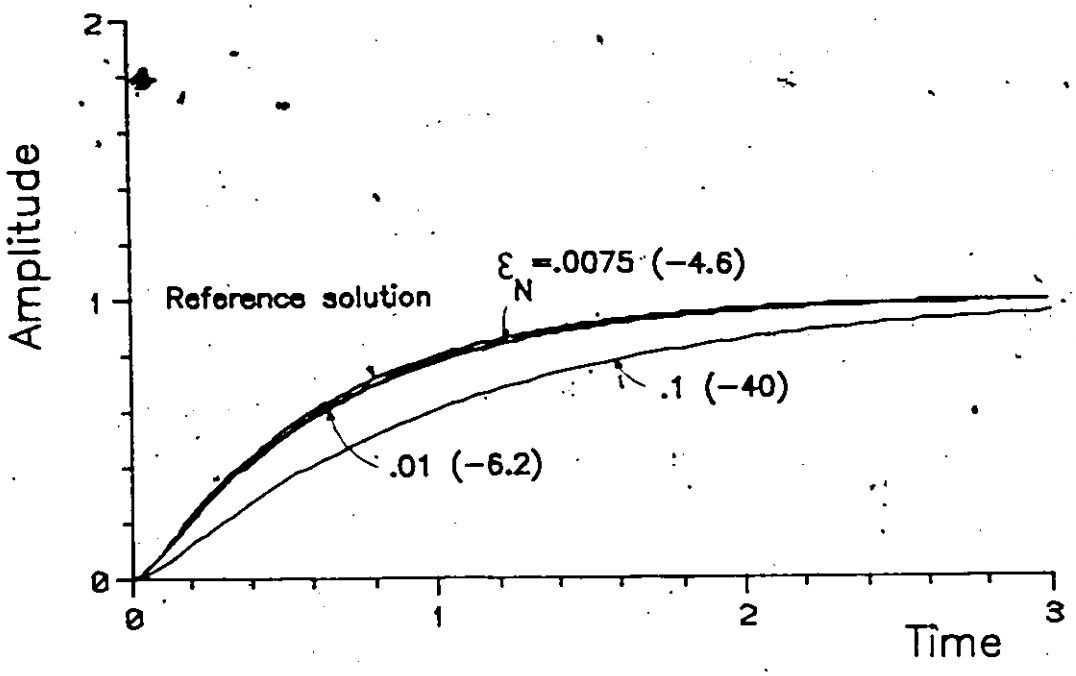


Figure A−11 : Normalized error with damping ratio of 2 for all solutions
(Numbers in brackets are percentage variations in undamped natural frequency)

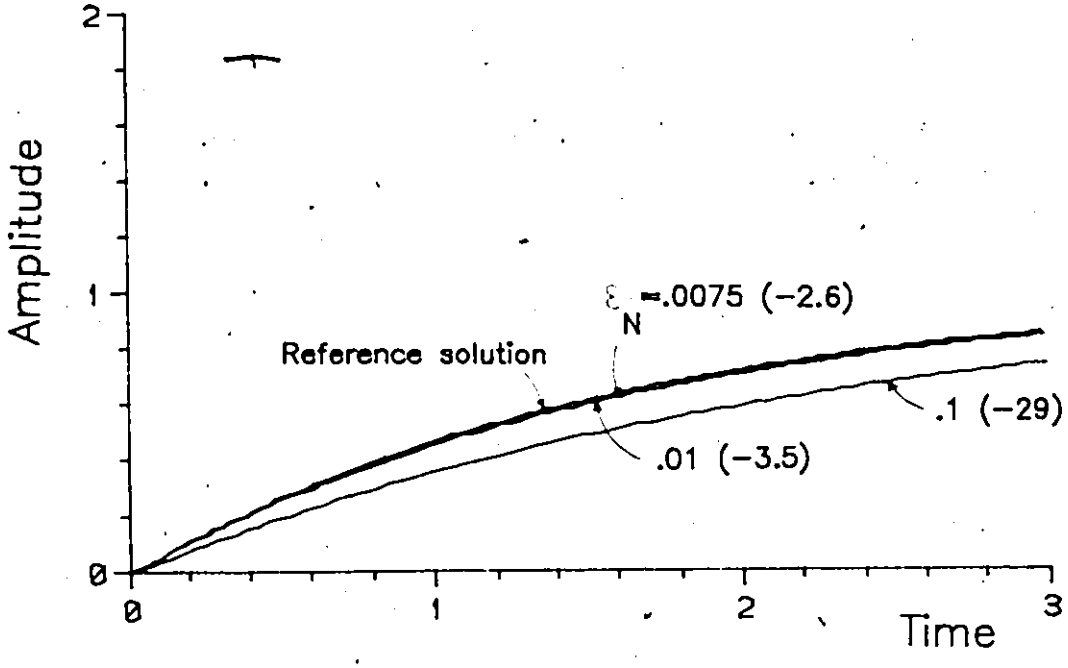


Figure A−12 : Normalized error with damping ratio of 5 for all solutions
(Numbers in brackets are percentage variations in undamped natural frequency)

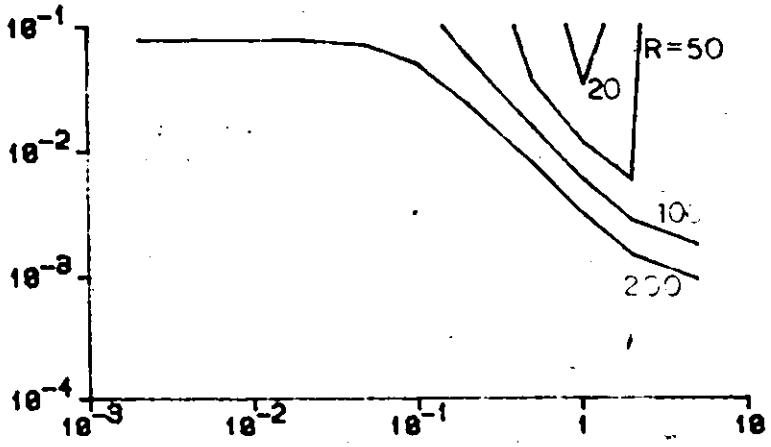
APPENDIX B

Computerized Model Performance Data

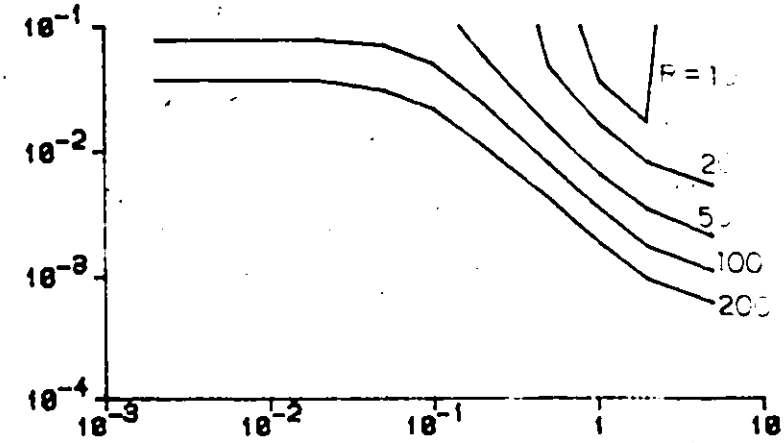
TABLE OF CONTENTS

	Table and Graph	θ'	Page
System D relative to A	DA 1	0	120
	DA 2	0.5	120
	DA 3	1.0	121
	DA 4	1.5	121
	DA 5	2.0	122
System C relative to A	CA 1	0	123
	CA 2	0.5	123
	CA 3	1.0	124
	CA 4	1.5	124
	CA 5	2.0	125
System C relative to D	CD 1	0	126
	CD 2	0.5	126
	CD 3	1.0	127
	CD 4	1.5	127
	CD 5	2.0	128
System B relative to A	BA		129

R	5.	10.	20.	50.	100.	200.	500.
BETA							
0.0001	.272	.358	.329	.261	.159	.079	.031
0.0201	.278	.358	.276	.262	.159	.077	.030
0.0501	.289	.365	.236	.275	.155	.071	.024
0.1001	.316	.358	.223	.229	.133	.050	.017
0.2001	.393	.224	.290	.206	.059	.024	.008
0.5001	.280	.320	.219	.035	.015	.008	.003
1.0001	.222	.156	.034	.012	.006	.003	.001
2.0001	1.00	1.00	.210	.006	.003	.001	.7E-3
5.0001	1.00	1.00	1.00	.002	.9E-3	.5E-3	



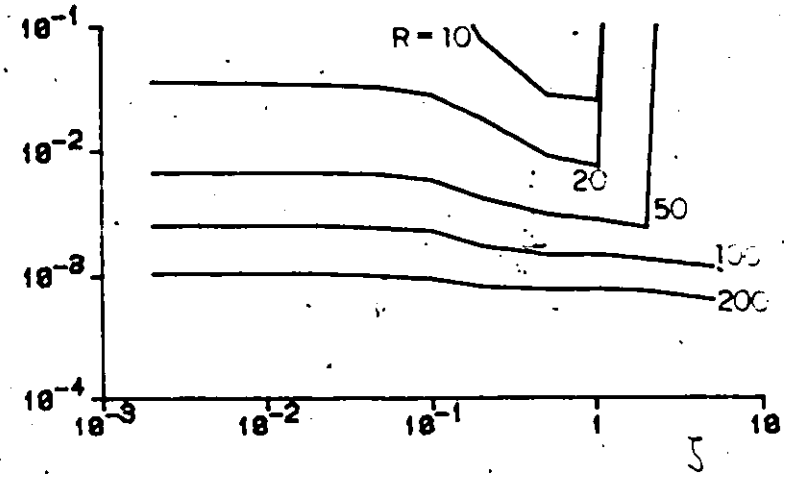
R	5.	10.	20.	50.	100.	200.	500.
BETA							
0.0001	.346	.347	.300	.152	.070	.038	.015
0.020	.347	.337	.304	.152	.072	.037	.014
0.050	.361	.350	.306	.153	.070	.031	.012
0.100	.361	.334	.309	.132	.049	.022	.008
0.200	.236	.269	.288	.059	.024	.011	.004
0.500	.381	.195	.048	.015	.003	.004	.002
1.000	.093	.039	.016	.006	.003	.002	.7E-3
2.000	.176	.017	.008	.003	.002	.9E-3	.4E-3
5.000	1.00	1.00	.005	.002	.001	.6E-3	.3E-3



\R / ZETA\	5.	10.	20.	50.	100.	200.	500.
0.002	1.00	.227	.035	.007	.002	.001	4E=3
0.020	1.00	.269	.034	.007	.002	.001	4E=3
0.050	1.00	.266	.032	.006	.002	1E=2	4E=3
0.100	1.00	.217	.028	.006	.002	9E=3	4E=3
0.200	1.00	.077	.017	.004	.002	8E=3	4E=3
0.500	.245	.027	.009	.003	.001	8E=3	3E=3
1.000	1.00	.025	.007	.003	.001	8E=3	3E=3
2.000	1.00	1.00	1.00	.002	.001	7E=3	3E=3
5.000	1.00	1.00	1.00	1.00	.001	6E=3	3E=3

TABLE DA 3

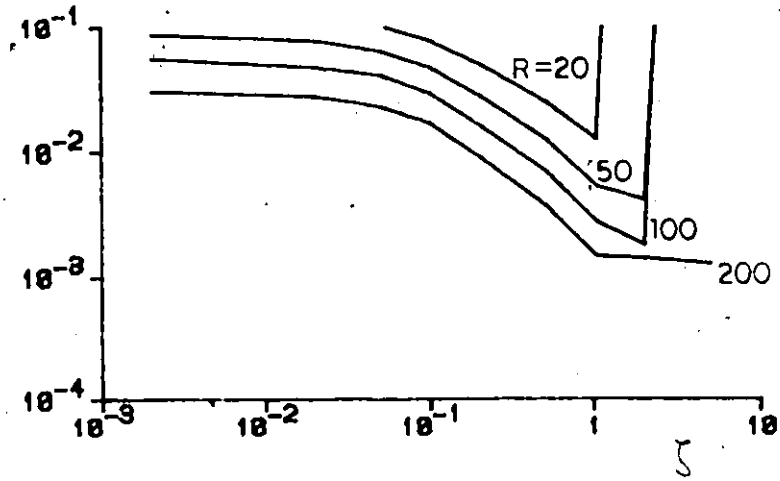
$\beta' = 1.0$



\R / ZETA\	5.	10.	20.	50.	100.	200.	500.
0.002	1.00	1.00	.130	.085	.054	.031	.014
0.020	1.00	1.00	.118	.077	.048	.028	.012
0.050	1.00	1.00	.101	.065	.041	.023	.010
0.100	1.00	.231	.079	.049	.030	.017	.007
0.200	1.00	.217	.051	.028	.016	.009	.004
0.500	1.00	.235	.026	.012	.007	.004	.002
1.000	1.00	.199	.012	.005	.003	.001	2E=3
2.000	1.00	1.00	1.00	.004	.002	.001	6E=3
5.000	1.00	1.00	1.00	1.00	1.00	.001	6E=3

TABLE DA 4

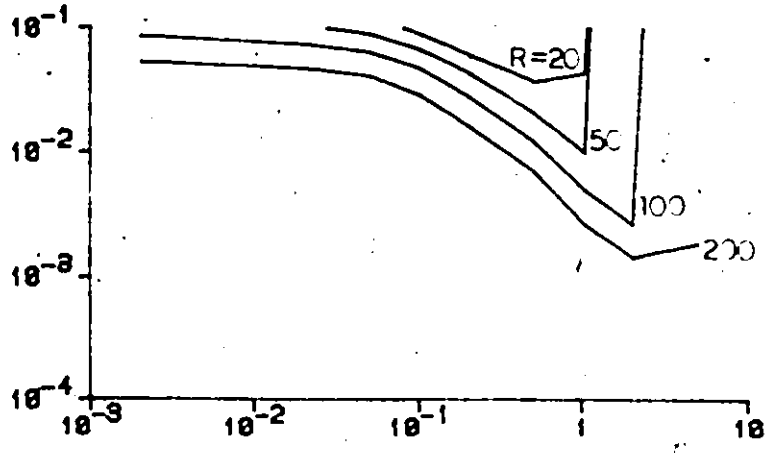
$\beta' = 1.5$



ZETA \ R	5.	10.	20.	50.	100.	200.	500.
0.002	1.00	1.00	.132	.114	.084	.054	.026
0.020	1.00	1.00	.128	.104	.076	.049	.023
0.050	1.00	1.00	.113	.089	.064	.040	.019
0.100	1.00	1.00	.091	.068	.048	.030	.014
0.200	1.00	1.00	.062	.043	.028	.016	.007
0.500	1.00	1.00	.038	.020	.012	.007	.003
1.000	1.00	1.00	.043	.010	.005	.003	.001
2.000	1.00	1.00	1.00	1.00	.003	.001	8E-3
5.000	1.00	1.00	1.00	1.00	1.00	.002	8E-3

TABLE DA 5

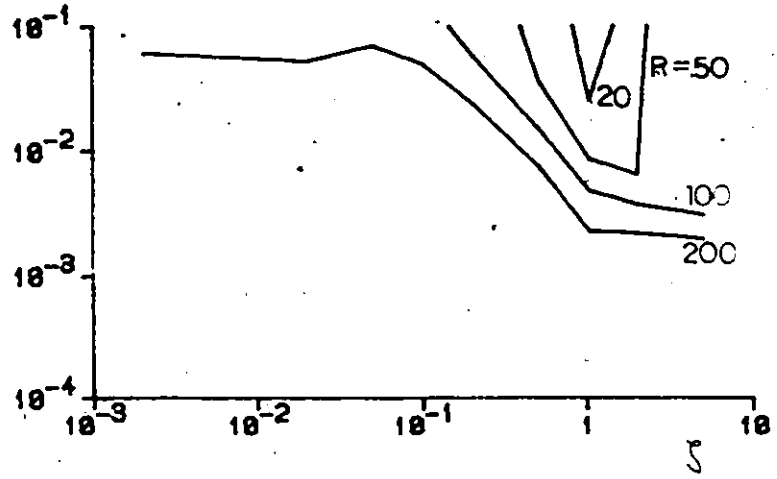
θ' = 2.0



\R	5.	10.	20.	50.	100.	200.	500.
ZETA\							
0.0021	.358	.256	.142	.221	.124	.060	.023
0.0201	.363	.259	.146	.214	.138	.053	.029
0.0501	.369	.264	.173	.175	.101	.069	.024
0.1001	.375	.272	.265	.201	.132	.049	.017
0.2001	.383	.282	.288	.204	.059	.023	.009
0.5001	.367	.291	.217	.035	.015	.007	.004
1.0001	.243	.153	.025	.009	.005	.002	.004
2.0001	1.00	1.00	.200	.006	.004	.002	.006
5.0001	1.00	1.00	1.00	1.00	.003	.002	.007

TABLE CA 1

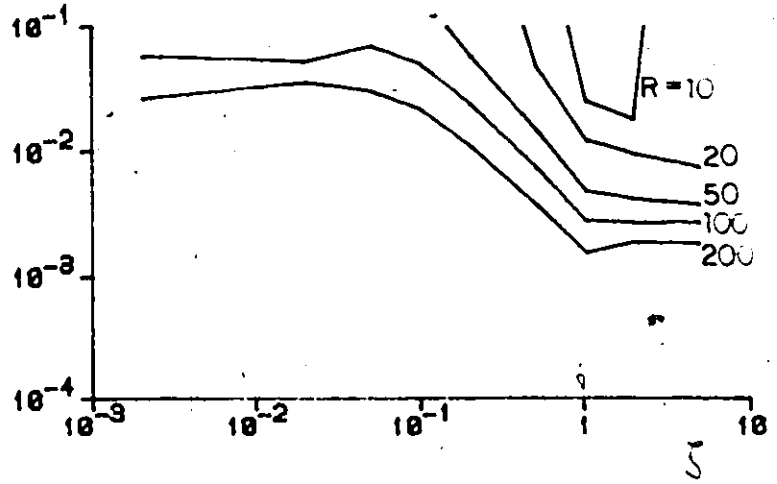
B' = 0.0



R	5.	10.	20.	50.	100.	200.	500.
ZETA\							
0.0021	.242	.141	.162	.121	.052	.027	.012
0.0201	.248	.150	.214	.134	.053	.035	.014
0.0501	.252	.217	.232	.101	.069	.030	.012
0.1001	.260	.260	.196	.130	.049	.021	.008
0.2001	.263	.285	.258	.058	.024	.011	.005
0.5001	.289	.191	.047	.015	.007	.004	.004
1.0001	.092	.026	.012	.005	.003	.002	.004
2.0001	1.00	.018	.009	.004	.003	.002	.005
5.0001	1.00	1.00	.007	.004	.003	.002	.007

TABLE CA 2

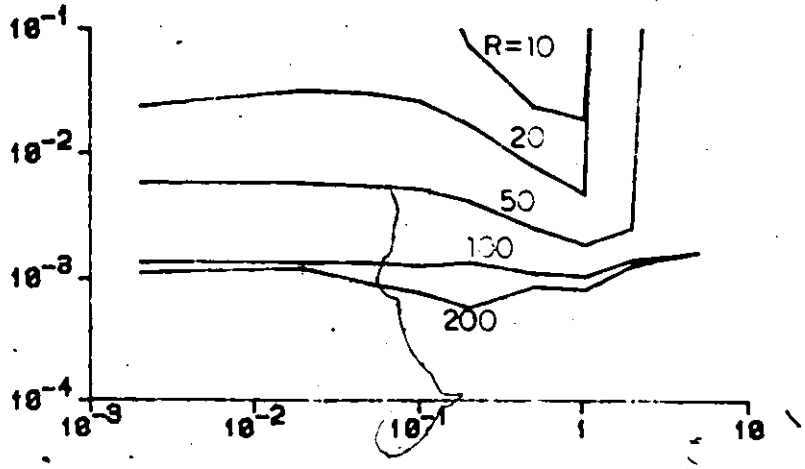
B' = 0.5



R / ZETA	.5	10.	20.	50.	100.	200.	500.
0.0021	.160	.121	.024	.006	.001	.001	.001
0.0201	.161	.120	.033	.006	.001	.001	.001
0.0501	.166	.119	.031	.005	.001	9E=31	.001
0.1001	.185	.189	.026	.005	.001	7E=31	.001
0.2001	.245	.025	.017	.004	.001	6E=31	.002
0.5001	.255	.024	.008	.003	.001	8E=31	.003
1.0001	1.00	.019	.005	.002	.001	8E=31	.004
2.0001	1.00	1.00	1.00	.002	.001	.001	.005
5.0001	1.00	1.00	1.00	1.00	.002	.002	.002

TABLE CA 3

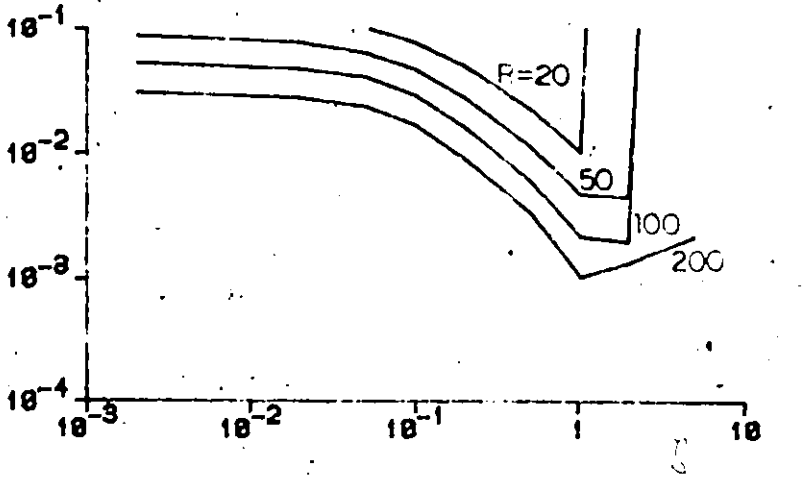
$M_1 = 1.0$



R / ZETA	5.	10.	20.	50.	100.	200.	500.
0.0021	.202	.156	.130	.085	.054	.031	.014
0.0201	.208	.161	.118	.022	.048	.028	.013
0.0501	.209	.146	.101	.065	.041	.023	.010
0.1001	.229	.189	.079	.048	.030	.012	.007
0.2001	.230	.251	.050	.028	.016	.009	.004
0.5001	.228	.230	.023	.011	.006	.003	.003
1.0001	1.00	1.00	.010	.005	.002	.001	.004
2.0001	1.00	1.00	1.00	.004	.002	.001	.005
5.0001	1.00	1.00	1.00	1.00	1.00	.002	.007

TABLE CA 4

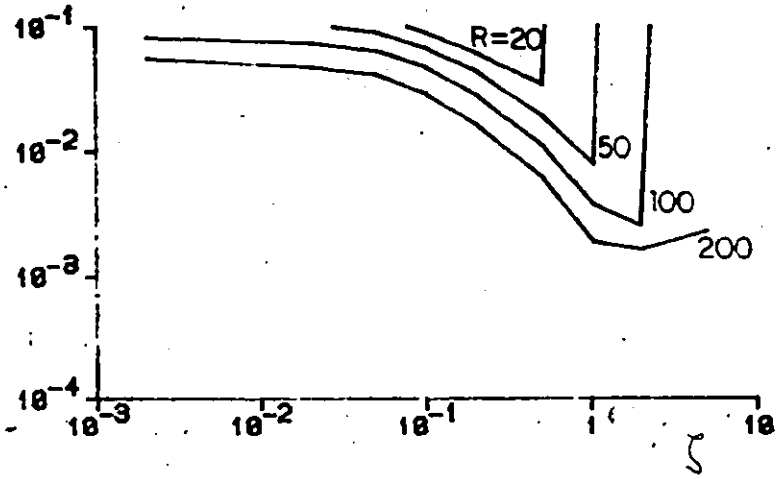
$M_1 = 1.5$



\R / ZETA\	5.	10.	20.	50.	100.	200.	500.
0.002	.240	.138	.139	.114	.084	.054	.026
0.020	.246	.142	.128	.104	.076	.049	.024
0.050	.257	.137	.112	.088	.064	.041	.019
0.100	.270	.189	.090	.068	.048	.030	.014
0.200	1.00	.224	.061	.043	.028	.016	.007
0.500	1.00	.189	.034	.019	.011	.006	.004
1.000	1.00	1.00	1.00	.008	.004	.002	.004
2.000	1.00	1.00	1.00	1.00	.003	.002	.005
5.000	1.00	1.00	1.00	1.00	1.00	.002	.007

TABLE CA 5

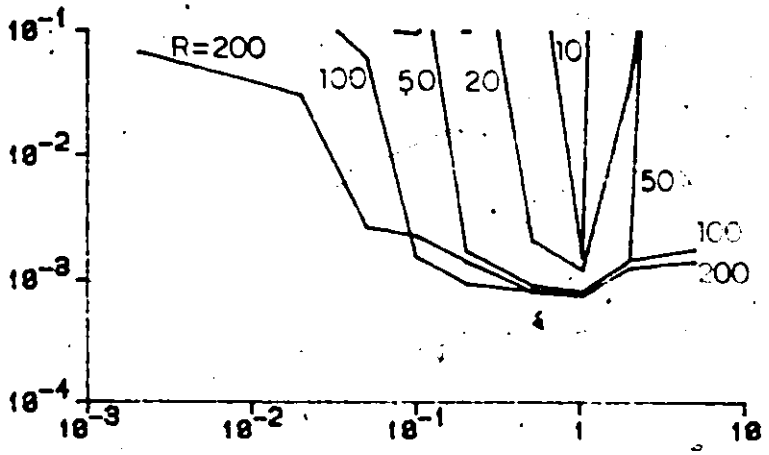
$\theta' = 2.0$



R \ S	5.	10.	20.	50.	100.	200.	500.
0.0021	.153	.153	.134	.166	.133	.067	.015
0.0201	.148	.149	.118	.163	.129	.030	.001
0.0501	.154	.142	.103	.147	.061	.003	1E-21
0.1001	.147	.145	.092	.131	.001	.002	.001
0.2001	.148	.029	.158	.002	9E-31	.001	.002
0.5001	.086	.141	.002	8E-31	8E-31	8E-31	.003
1.0001	.032	.002	.001	8E-31	8E-31	2E-31	.004
2.0001	1.00	1.00	.037	.001	.001	.001	.005
5.0001	1.00	1.00	1.00	1.00	.002	.001	.002

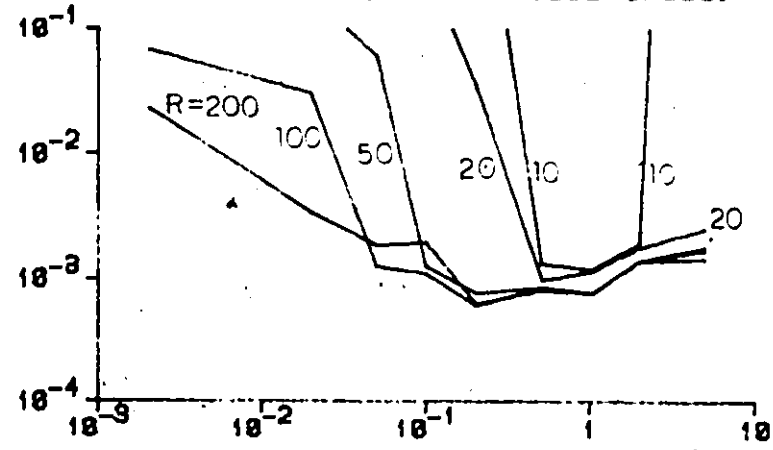
TABLE CD

I = 0.0



R \ S	5.	10.	20.	50.	100.	200.	500.
0.0021	.143	.140	.157	.132	.063	.013	.001
0.0201	.143	.139	.150	.132	.030	.003	.001
0.0501	.138	.114	.105	.062	.003	.002	.001
0.1001	.135	.110	.142	.001	.001	.002	.001
0.2001	.105	.152	.020	8E-31	8E-31	8E-31	.002
0.5001	.137	.001	1E-21	8E-31	8E-31	8E-31	.003
1.0001	.001	.001	.001	8E-31	8E-31	2E-31	.004
2.0001	1.00	.002	.002	.001	.001	.001	.005
5.0001	1.00	1.00	.003	.002	.002	.001	.002

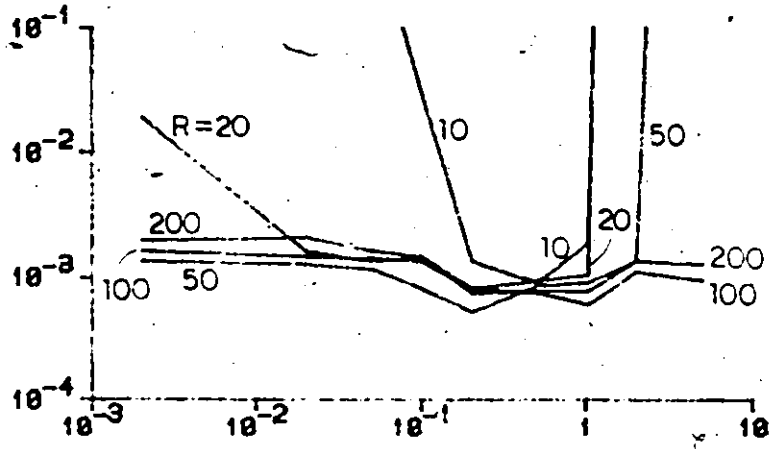
TABLE CD



ZETA \ R	5.	10.	20.	50.	100.	200.	500.
0.002	1.00	.152	.012	.001	.002	.002	.9E=3
0.020	1.00	.142	.002	.001	.001	.002	.8E=3
0.050	1.00	.174	.001	.001	.001	.002	.9E=3
0.100	1.00	.032	.001	.8E=3	.001	.001	.001
0.200	1.00	.001	.8E=3	.5E=3	.8E=3	.2E=3	.002
0.500	.025	.9E=3	.9E=3	.8E=3	.2E=3	.8E=3	.003
1.000	1.00	.002	.001	.9E=3	.6E=3	.8E=3	.004
2.000	1.00	1.00	1.00	.001	.001	.001	.005
5.000	1.00	1.00	1.00	1.00	.9E=3	.001	.002

TABLE CD 3

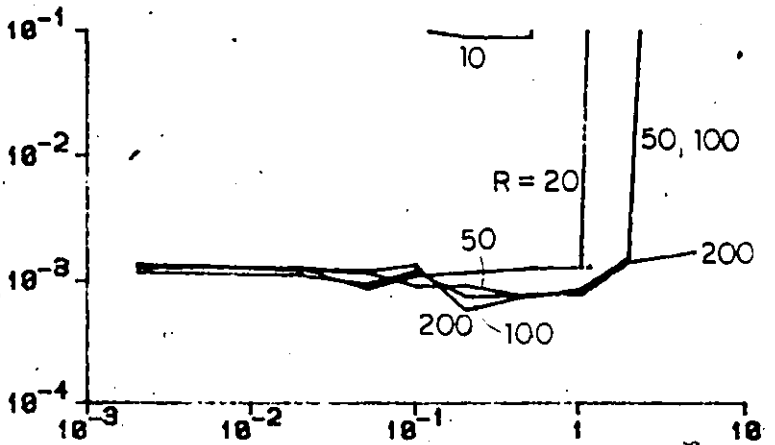
R' = 1.0



ZETA \ R	5.	10.	20.	50.	100.	200.	500.
0.002	1.00	1.00	.001	.001	.001	.001	.8E=3
0.020	1.00	1.00	.001	.001	.001	.001	.8E=3
0.050	1.00	1.00	.8E=3	.001	.9E=3	.001	.8E=3
0.100	1.00	.103	.001	.9E=3	.001	.001	.001
0.200	1.00	.089	.001	.9E=3	.2E=3	.6E=3	.002
0.500	1.00	.092	.001	.2E=3	.8E=3	.8E=3	.003
1.000	1.00	1.00	.001	.8E=3	.8E=3	.8E=3	.004
2.000	1.00	1.00	1.00	.001	.001	.001	.005
5.000	1.00	1.00	1.00	1.00	1.00	.002	.002

TABLE CD 4

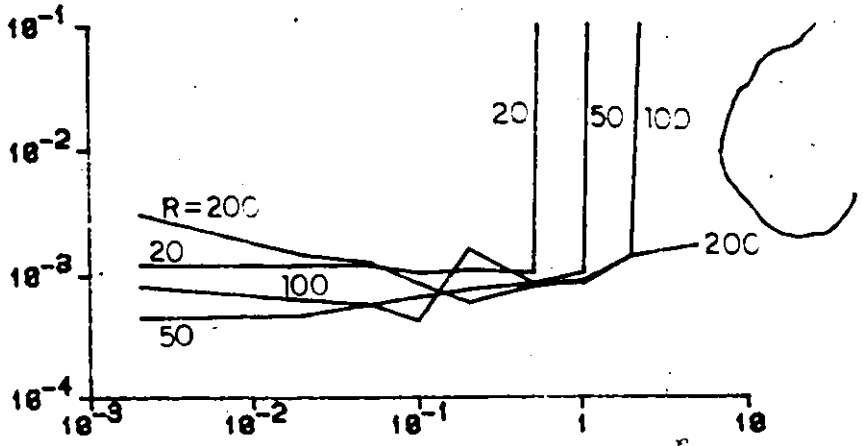
R' = 1.5



ZETA \ R	5.	10.	20.	50.	100.	200.	500.
0.002	1.00	1.00	1.001	1.4E=31	1.8E=31	1.003	1.9E=3
0.020	1.00	1.00	1.001	1.5E=31	1.6E=31	1.001	1.1E=2
0.050	1.00	1.00	1.001	1.6E=31	1.6E=31	1.001	1.001
0.100	1.00	1.00	1.001	1.7E=31	1.4E=31	1.9E=31	1.001
0.200	1.00	1.00	1.001	1.8E=31	1.002	1.6E=31	1.002
0.500	1.00	1.00	1.001	1.8E=31	1.8E=31	1.8E=31	1.003
1.000	1.00	1.00	1.00	1.001	1.8E=31	1.9E=31	1.004
2.000	1.00	1.00	1.00	1.00	1.001	1.001	1.005
5.000	1.00	1.00	1.00	1.00	1.00	1.002	1.007

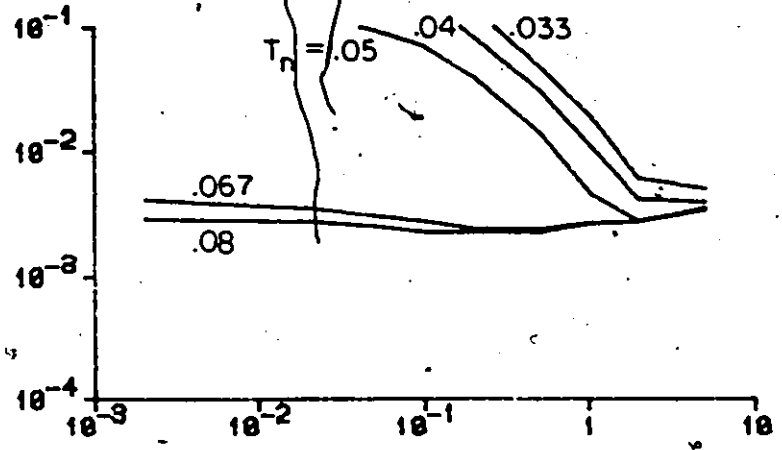
TABLE CD 5

M = 2.0



ZETA \ Tn	0.025	0.033	0.040	0.050	0.067	0.080	0.100
0.002	.181	.208	.203	.127	.004	.003	.003
0.020	.175	.192	.189	.113	.003	.003	.002
0.050	.168	.181	.168	.095	.003	.002	.002
0.100	.162	.159	.135	.070	.003	.002	.002
0.200	.152	.117	.083	.039	.002	.002	.002
0.500	.091	.048	.031	.014	.002	.002	.002
1.000	.033	.018	.011	.004	.003	.003	.003
2.000	.041	.006	.004	.003	.003	.003	.003
5.000	.022	.005	.004	.003	.003	.003	.004

TABLE BA



APPENDIX C

System A Documentation

Contents

- Flowchart
- Listing

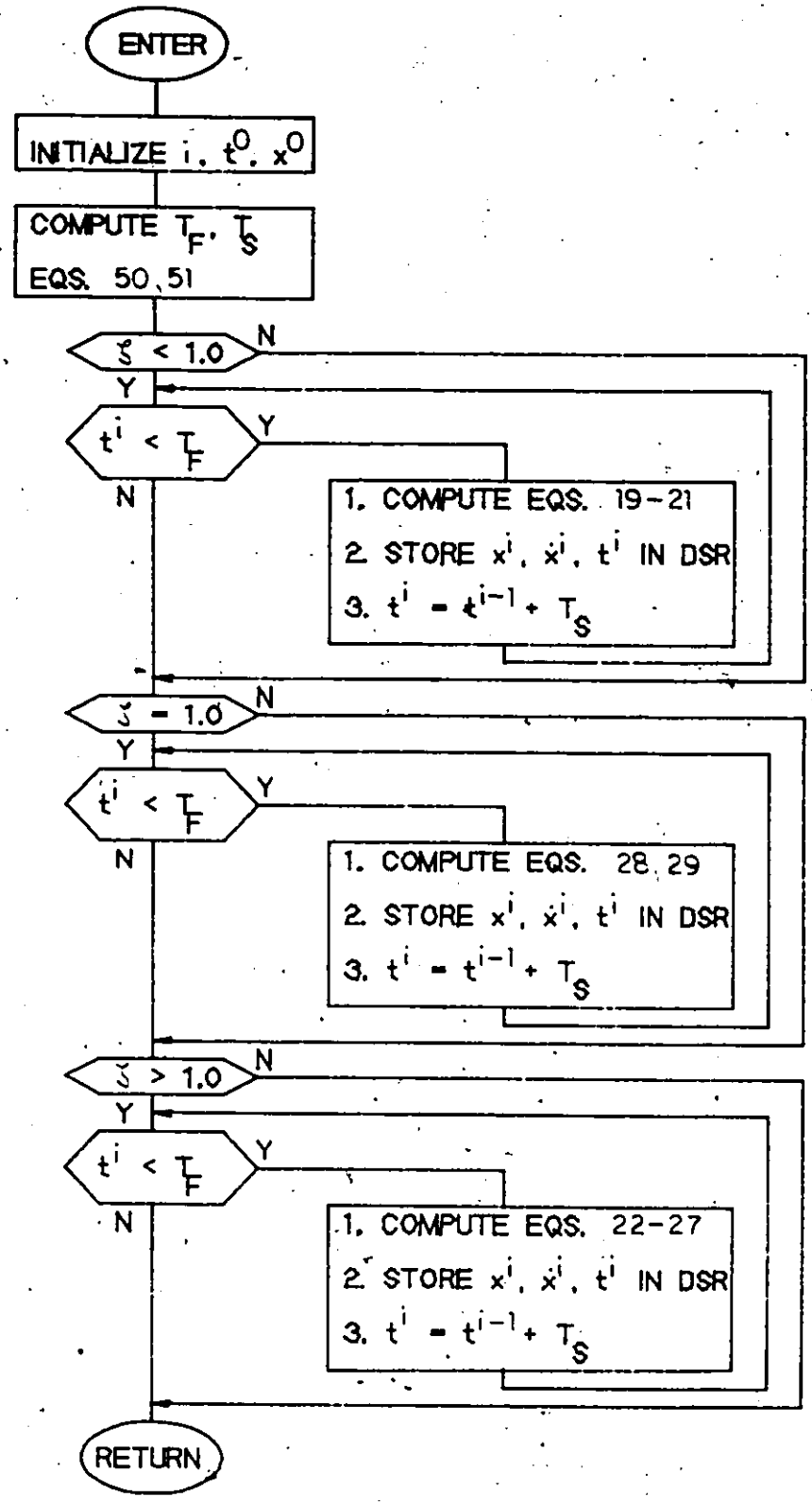


Figure C-1 : Subroutine SYSTMA flowchart

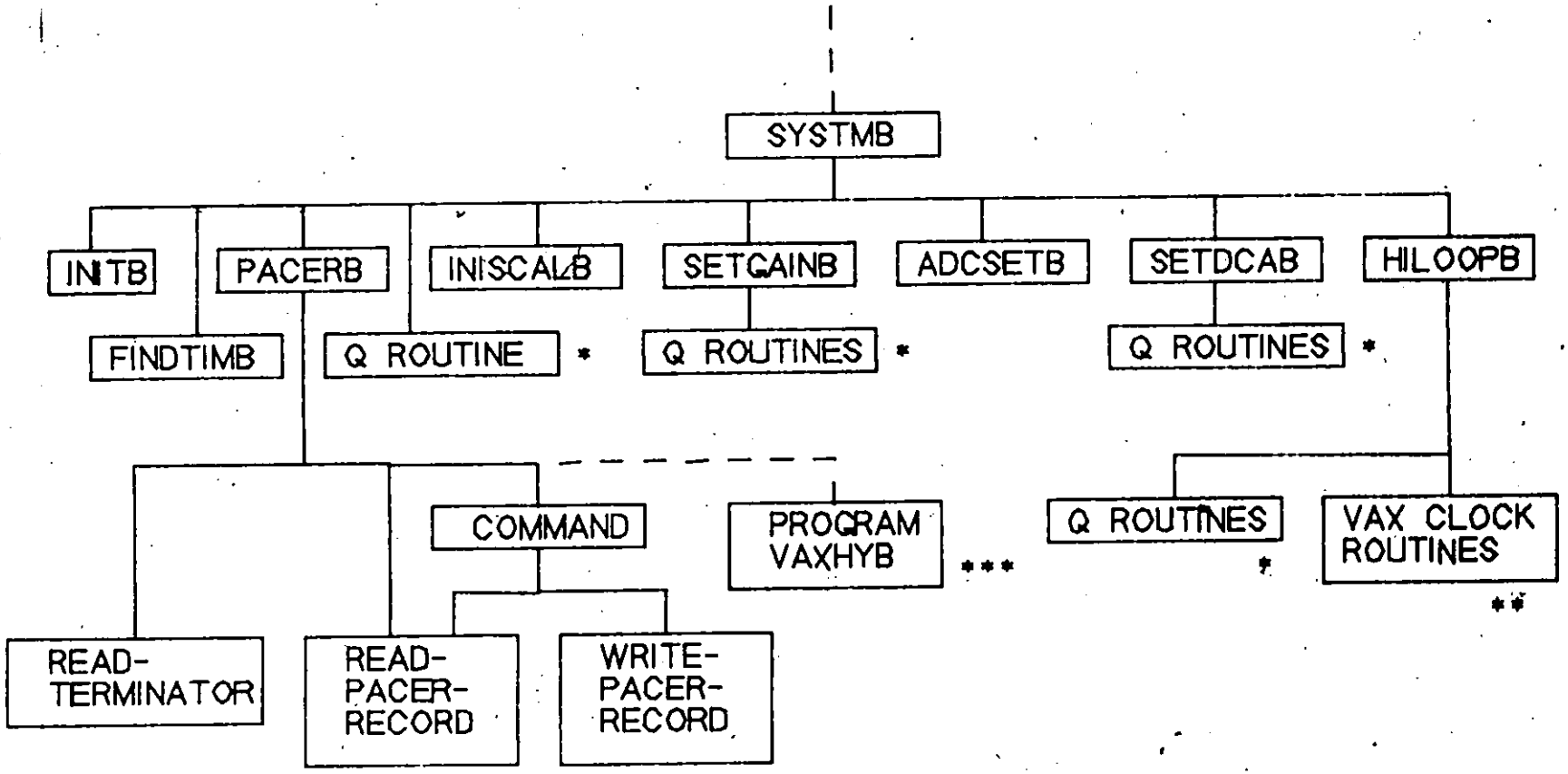
```
C_____Subroutine SYSTMA
C_____Computes response data points for the analytical system
      SUBROUTINE SYSTMA

C_____Specification statements
      DIMENSION RESPONSE(1000), RESPONSD(1000), RESTIME(1000)

      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /CONSTANTS/ XF, PI
      COMMON /FORCE/ STEP
      COMMON /NUMTIM/ NT, NS
      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME
      COMMON /RESPONSDOT/ RESPONSD

C_____Initialization
      TIME = 0.0
      X = 0.0
      NPOINT = 0

C_____Compute final solution time and computation time
C_____interval
      FINALTIME = (NT*2.*PI)/OMEGAN
      TIMEINC = (2.*PI)/(NS*OMEGAN)

C_____Compute response data if ZETA is less than 1
      IF(ZETA.LT.1.0) THEN

C_____Initialize constants
        SQROOT = SQRT(1. - ZETA**2.)
        PHI = ATAN(SQROOT/ZETA)

C_____Compute data points until final solution time
        DO WHILE(TIME.LE.FINALTIME)
          X = STEP - EXP(-ZETA*OMEGAN*TIME)/SQROOT*
     -              SIN(OMEGAN*SQROOT*TIME + PHI)
          XD = OMEGAN*EXP(-ZETA*OMEGAN*TIME)
     -         *((ZETA/SQROOT)*SIN(OMEGAN*SQROOT*TIME + PHI)
     -         - COS(OMEGAN*SQROOT*TIME + PHI))
          NPOINT = NPOINT + 1
          RESPONSE(NPOINT) = X
          RESPONSD(NPOINT) = XD
          RESTIME(NPOINT) = TIME
          TIME = TIME + TIMEINC
        END DO
      END IF

C_____Compute response data if ZETA equals 1
      IF(ZETA.EQ.1.0) THEN

C_____Compute data points until final solution time
        DO WHILE(TIME.LE.FINALTIME)
          X = STEP - EXP(-OMEGAN*TIME)*(1. + OMEGAN*TIME)
          XD = EXP(-OMEGAN*TIME)*(OMEGAN**2.)*TIME
          NPOINT = NPOINT + 1
          RESPONSE(NPOINT) = X
          RESPONSD(NPOINT) = XD
          RESTIME(NPOINT) = TIME
          TIME = TIME + TIMEINC
        END DO
      END IF

C_____Compute response data if ZETA is greater than 1
      IF(ZETA.GT.1.0) THEN

C_____Compute constant and coefficients
        SQROOT = SQRT(ZETA**2. - 1.)
        CA = OMEGAN*(ZETA - SQROOT)
        CB = OMEGAN*(ZETA + SQROOT)
        CC = -(ZETA + SQROOT)/(2.*SQROOT)
        CD = -(-ZETA + SQROOT)/(2.*SQROOT)

C_____Compute data points until final solution time
        DO WHILE(TIME.LE.FINALTIME)
          X = STEP + CC*EXP(-CA*TIME) + CD*EXP(-CB*TIME)
          XD = -CA*CC*EXP(-CA*TIME) - CB*CD*EXP(-CB*TIME)
          NPOINT = NPOINT + 1
          RESPONSE(NPOINT) = X
          RESPONSD(NPOINT) = XD
          RESTIME(NPOINT) = TIME
          TIME = TIME + TIMEINC
        END DO
      END IF

      RETURN
      END
```

APPENDIX D

System B Documentation

Contents

- Subroutine Hierarchical Diagram
- Hierarchy
- Input-Process-Output Table of Subroutines
- Flowcharts
- Listings



* Q ROUTINES ARE VAX/HYBRID SYSTEM SUBROUTINES

** VAX CLOCK ROUTINES ARE VAX SYSTEM SUBROUTINES

*** PROGRAM VAXHYB IS A VAX/HYBRID SYSTEM PROGRAM

Figure D–1 : Subroutine hierarchical diagram of subroutine SYSTMB

Subroutine SYSTMB Hierarchy

1. Initialize variables ... sub INITB
2. Compute T_F and T_S ... sub FINDTIMB
3. Prepare PACER computer to run PACER program VAXHYB ... sub PACERB
4. Establish connection with analog computer ... sub GSHYIN
5. Initialize x_M, $\dot{x}_M$, $\ddot{x}_M$ ... sub INISCALB
6. Set amplifier gain G ... sub SETGAINB
7. Set A/D channel numbers ... sub ADCSETB
8. Set DCA channel numbers and set DCA values on analog computer ... sub SETDCAB
9. Execute high-speed-loop ... sub HILOOPB

3. Sub PACERB

3.1 Assign a channel to PACER

3.2 Set up terminator mask for PACER read operation ... sub READ-TERMINATOR

3.3 Obtain PACER monitor ... sub READ-PACER-RECORD

3.4 Load program VAXHYB on PACER and start its execution ... sub COMMAND

6. Sub SETGAINB

6.1 Set G=1 on analog integrators ... sub GSSECN

6.2 Set G=10 ... sub GSSECF

6.3 SET C=100 ... sub GSMSN

8. Sub SETDCAB

8.1 Set DCA channel numbers

8.2 Set DCA values

8.3 Establish number of DCA's used

8.4 Set DCA values on analog computer ... sub GWRPTR

9. Sub HILOOPB

9.1 Initialize time variable

9.2 Put analog computer in IC mode ... sub GINIT

9.3 Read starting time on VAX clock ... fctn USER-GET-REALTIME

9.4 Put analog computer in operate (OP) mode ... sub GOPER

9.5 Read data from A/D converters ... sub GRBADS

9.6 Descale A/D data

9.7 Compute time since start of solution ... sub SUBQUAD

9.8 Store response solution data

9.9 Put analog computer in IC mode ... sub GINIT

Input-Process-Output Table of Subroutines

Subroutine	Inputs	Process	Outputs
SYSTMB	ζ, ω_n, π, u, N_T, N_S	Fig. D-2	DSR
FINDTIMB	ω_n, π, N_T, N_S	Eqs. 50, 51	T_F, T_S
INISCALB	ω_n, u	Eqs. 52-54	x_M, $\dot{x}_M$, $\ddot{x}_M$
SETGAINB	ω_n	Fig. D-4	Q
SETDCAB	ω_n, ζ, u, Q, x^0, $\dot{x}^0$, x_M, $\dot{x}_M$, $\ddot{x}_M$	Eqs. 56-62	D_{15}, D_{20}, D_{25}, D_{22}, D_{27}, D_{23}, D_{28}
HILOOPB	T_F, T_S, x_M, $\dot{x}_M$, x_S, $\dot{x}_S$	Fig. D-5	DSR

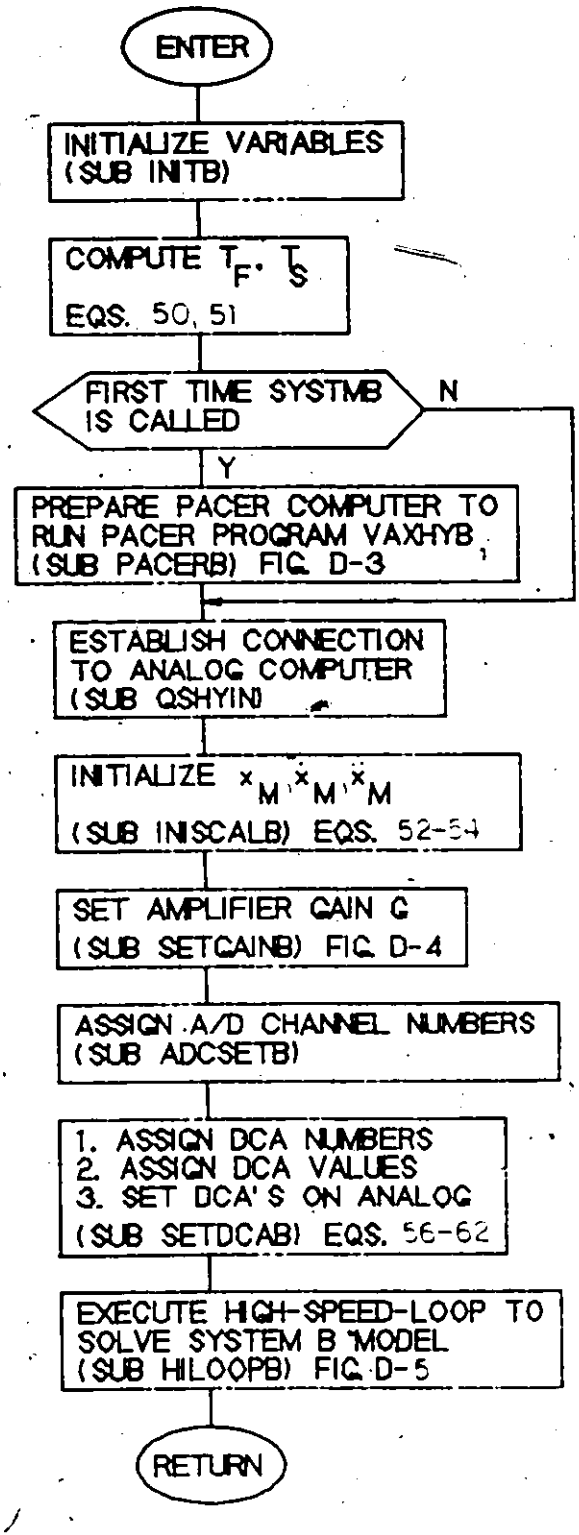


Figure D–2 : Subroutine SYSTMB flowchart

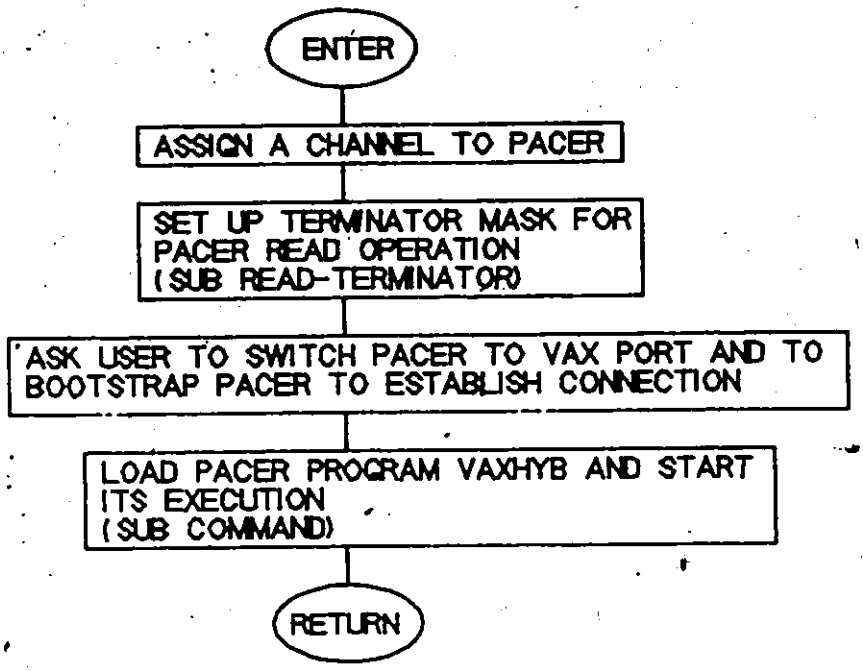


Figure D–3 : Subroutine PACERB flowchart

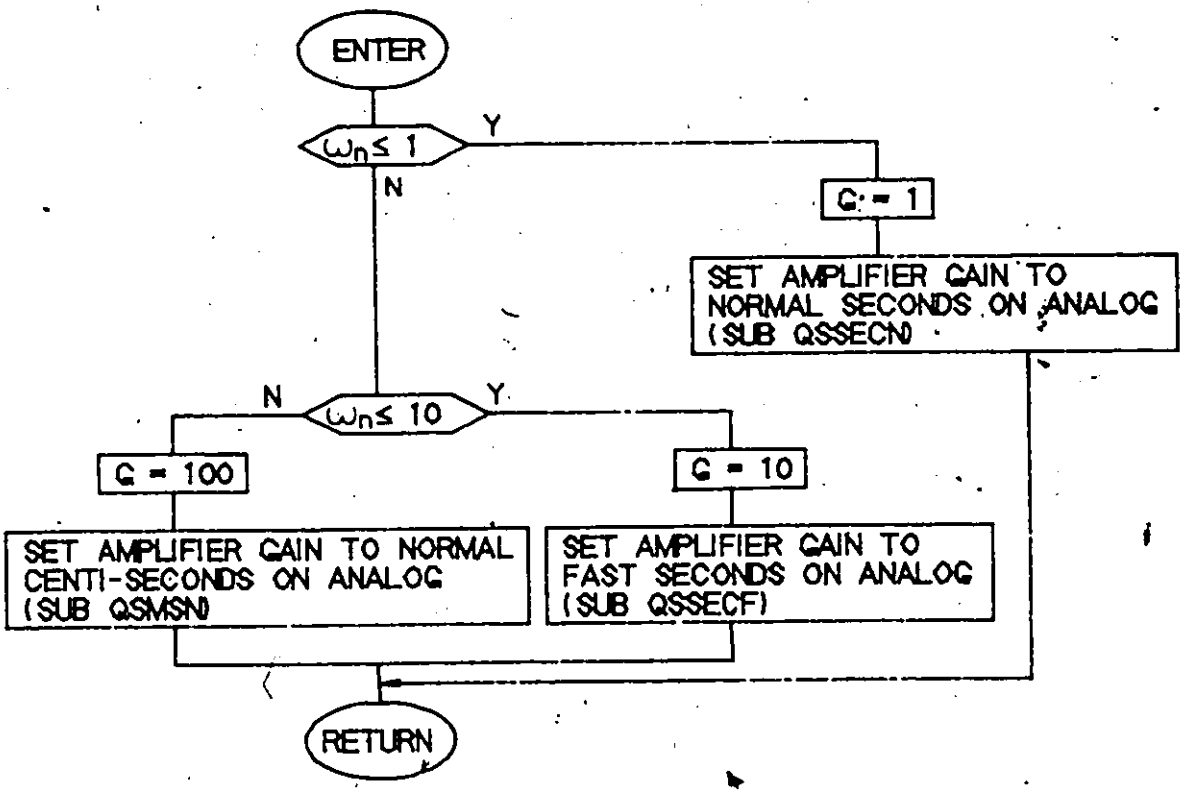


Figure D–4 : Subroutine SETGAINB flowchart

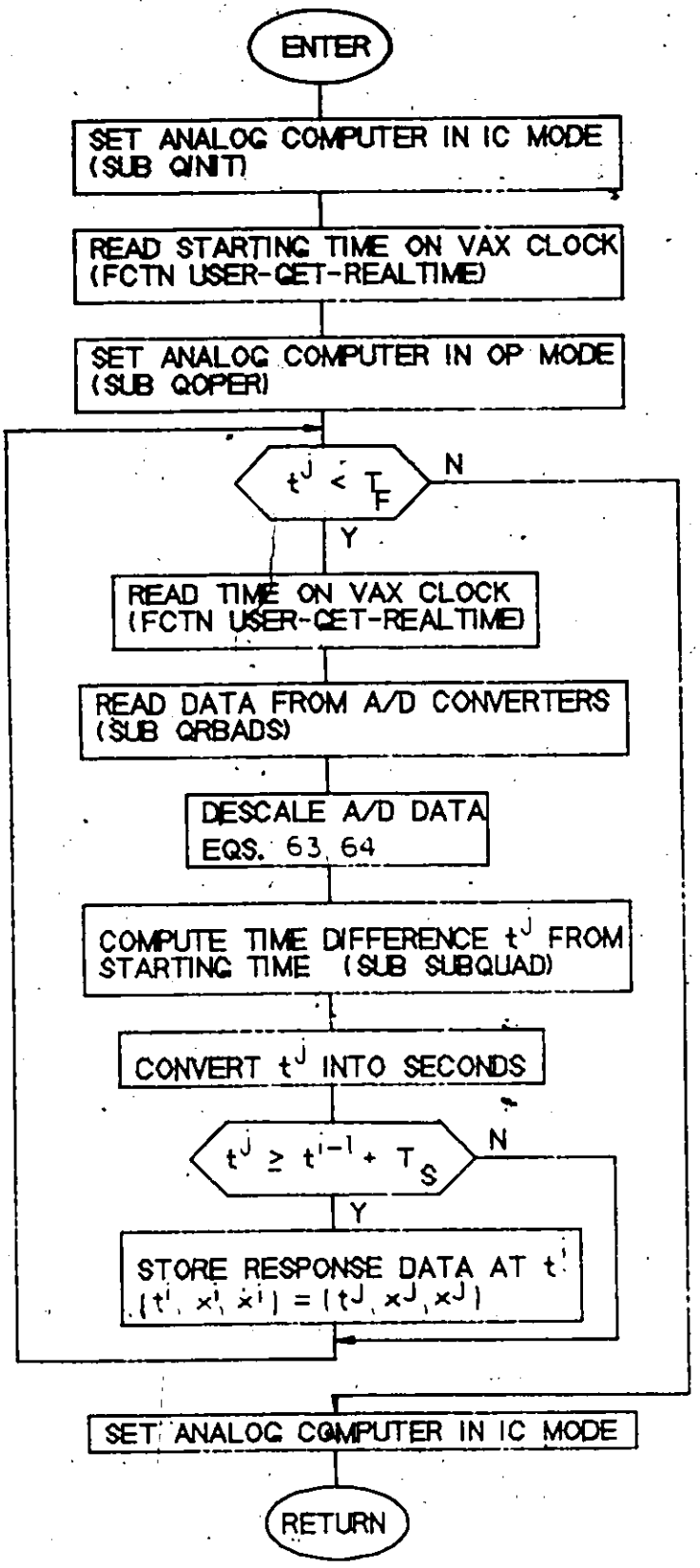


Figure D–5 : Subroutine HILOOPB flowchart

```
C_____Subroutine SYSTMB
C_____Computes response data points for the continuous-time
C_____system (Analog computer)
      SUBROUTINE SYSTMB

C_____Specification statements
      LOGICAL FIRSTRUNB, FIRSTRUNC

      COMMON /FIRSTRUN/ FIRSTRUNB, FIRSTRUNC

C_____Prepare the PACER to enable VAX/HYBRID running
C_____if it is the first time subroutine SYSTMB is called or
C_____if subroutine SYSTMC has been called in the same main
C_____program run
      IF(FIRSTRUNB) THEN
        CALL PACERB
      END IF

C_____Establish connection to analog computer
      CALL OSHYIN(IER,680,680)

C_____Initialize constants
      CALL INITB

C_____Compute final solution time and data storage sampling
C_____period
      CALL FINDTIMB

C_____Initialize scaling variables
      CALL INISCALB

C_____Set amplifier gain value for analog integrators
      CALL SETGAINB

C_____Assign A/D channel numbers
      CALL ADCSETB

C_____Assign DCA numbers, assign DCA values and set DCA's
C_____on analog computer
      CALL SETDCAB

C_____Execute high-speed-loop to solve System B model
      CALL HILOOPB

      CALL RELEASE_DR11B

      RETURN
      END



C_____Subroutine INITB
C_____Initializes constants
      SUBROUTINE INITB

C_____Specification statements
      REAL LASTSAVTIM

      DIMENSION RESPONSE(1000), RESPONSD(1000), RESTIME(1000)

      COMMON /CONSOLE/ NOCNLS
      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME
      COMMON /RESPONSDOT/ RESPONSD
      COMMON /TIM/ TIME, FINALTIME, SAVINCTIM, LASTSAVTIM

C_____Initialize console number
      NOCNLS= 1

C_____Initialize response data pointer
      NPOINT= 1

C_____Initialize first time data
      TIME = 0.0

C_____Store first response and time data
      RESPONSE(NPOINT) = 0.0
      RESPONSD(NPOINT) = 0.0
      RESTIME(NPOINT)  = 0.0

C_____Initialize last time data that has been stored
      LASTSAVTIM = 0.0

      RETURN
      END
```

```
C_____Subroutine FINDTIMB
C_____Finds final solution time for a fixed number (NT) of cycles
C_____and finds the data storage sampling period
      SUBROUTINE FINDTIMB

C_____Specification statements
      REAL LASTSAVTIM

      COMMON /CONSTANTS/ XF, PI
      COMMON /NUMTIM/ NT, NS
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /TIM/ TIME, FINALTIME, SAVINCTIM, LASTSAVTIM

C_____Compute final solution time
      FINALTIME = (NT*2 *PI)/OMEGAN

C_____Compute data storage sampling period
      SAVINCTIM = (2 *PI)/(NS*OMEGAN)

      RETURN
      END




C_____Subroutine PACERB
C_____Prepares the PACER to enable VAX/HYBRID runing for
C_____System B
      SUBROUTINE PACERB

C_____Specification statements
      EXTERNAL SS$_NORMAL

      INTEGER*4 SYS$ASSIGN, SYS$ALLOC
      INTEGER*2 CHANNEL2, LENGTH

      CHARACTER RECORD_STRING*80, TERM_MASK*32

      COMMON/GIO/CHANNEL2, LENGTH/MESSAGE/RECORD_STRING, TERM_MASK

C_____Assign a channel to PACER
      I=SYS$ASSIGN('PACER100:',CHANNEL2,,)
      IF(I NE %LOC(SS$_NORMAL)) THEN
        TYPE 100, I
      END IF

C_____Set up terminator mask for PACER read operation
      CALL READ_TERMINATOR

C_____Give instructions to user on how to obtain PACER monitor
      TYPE *,'Switch PACER to VAX port and press <RETURN>'
      ACCEPT *
      TYPE *
      TYPE *,'Bootstrap PACER to establish connection'
      TYPE *,'(message when connection established)'
      CALL READ_PACER_RECORD
      IF (RECORD_STRING(1:2) EQ 'M_') THEN
        TYPE *
        TYPE *,'        *    PACER CONNECTION ESTABLISHED    *'
      END IF

C_____Load VAX/HYBRID program on PACER and start execution
      CALL COMMAND('@L,VAXHYB,21'          ,12,'LD'     ,2 ,6 )
      CALL COMMAND('@R,,45000,70000'       ,15,' '      ,0 ,3 )
      CALL COMMAND('@G,1000'               ,7 ,' '      ,0 ,2 )

      TYPE *
      TYPE *,'          *    VAX/PACER NOW RUNING    *'
      TYPE *

C_____Format statement
  100 FORMAT(' CANNOT ASSIGN A CHANNEL TO PACER, STATUS = ',I8)

      RETURN
      END
```

```
C_____Subroutine WRITE_PACER_RECORD
C_____Sends a record to PACER
      SUBROUTINE WRITE_PACER_RECORD

      EXTERNAL SS$_NORMAL, IO$_WRITEVBLK, IO$M_NOFORMAT

C_____Set up storage
      COMMON/GIO/CHANNEL2, LENGTH/MESSAGE/RECORD_STRING, TERM_MASK
      INTEGER*4 SYS$QIOW
      INTEGER*2 CHANNEL2, LENGTH, CHAR_VALU, BIT_EIGHT, IOSB(4)
      BYTE RECORD(80)
      CHARACTER RECORD_STRING*80, TERM_MASK*32
      EQUIVALENCE (RECORD(1), RECORD_STRING(1:1))
      DATA BIT_EIGHT/'FF80'X/

C_____Insure record is in 8 bit ASCII code
      DO I=1, LENGTH
              CHAR_VALU=RECORD(I)
              CHAR_VALU=IIOR(CHAR_VALU, BIT_EIGHT)
              RECORD(I)=CHAR_VALU
      END DO

C_____Add eight bit carriage return
      LENGTH=LENGTH+1
      RECORD(LENGTH)='8D'X

C_____Send record to PACER
      I=SYS$QIOW(, %VAL(CHANNEL2), %VAL(%LOC(IO$_WRITEVBLK) OR
     - %LOC(IO$M_NOFORMAT)), IOSB, , , RECORD, %VAL(LENGTH), , , , )
      IF(I .NE. %LOC(SS$_NORMAL)) THEN
        TYPE 100, I
      END IF

C_____Format statement
  100 FORMAT(' QIO WRITE FUNCTION TO PACER FAILED, STATUS= ', I8)

      RETURN
      END
```

```
C_____Subroutine READ_TERMINATOR
C_____This routine establishes the terminator
C_____mask for PACER read function
      SUBROUTINE READ_TERMINATOR

C_____Set up storage
      COMMON/MESSAGE/RECORD_STRING, TERM_MASK
      INTEGER*2 LINE_FEED, CAR_RET, BYTENO, SHIFT
      BYTE BYTEVAL, BYTEVAL1
      CHARACTER RECORD_STRING*80, TERM_MASK*32, DUMYCHAR
      EQUIVALENCE (DUMYCHAR, BYTEVAL)
      DATA LINE_FEED, CAR_RET/'008A'X, '008D'X/

C_____First for the eight bit line feed
      BYTENO=LINE_FEED/8+1
      SHIFT=LINE_FEED-(BYTENO-1)*8
      BYTEVAL1=2**SHIFT

C_____Now for the eight bit carriage return
      BYTENO=CAR_RET/8+1
      SHIFT=CAR_RET-(BYTENO-1)*8
      BYTEVAL=2**SHIFT+BYTEVAL1
      TERM_MASK(BYTENO:BYTENO)=DUMYCHAR

      RETURN
      END
```

```
C_____Subroutine READ_PACER_RECORD
C_____Reads a record from PACER
      SUBROUTINE READ_PACER_RECORD

      EXTERNAL SS$_NORMAL,IO$_READVBLK

C_____Set up storage
      COMMON/GIO/CHANNEL2,LENGTH/MESSAGE/RECORD_STRING,TERM_MASK
      INTEGER*4 SYS$QIOW
      INTEGER*2 CHANNEL2,LENGTH,CHAR_VALU,BIT_EIGHT,IOSB(4)
      BYTE RECORD(80)
      CHARACTER RECORD_STRING*80,TERM_MASK*32
      EQUIVALENCE (RECORD(1),RECORD_STRING(1:1))
      DATA BIT_EIGHT/%7F'X/

C_____Get record from PACER
      I=SYS$QIOW(,%VAL(CHANNEL2),%VAL(%LOC(IO$_READVBLK)
     - ),IOSB,,,RECORD,%VAL(81),%VAL(10),TERM_MASK,,)
      IF(IOSB(1).NE.%LOC(SS$_NORMAL)) THEN
        RECORD_STRING='@'
        RETURN
      END IF

C_____Convert record to 7 bit ASCII code
      LENGTH=IOSB(2)
      DO I=1,LENGTH
              CHAR_VALU=RECORD(I)
              CHAR_VALU=IIAND(CHAR_VALU,BIT_EIGHT)
              RECORD(I)=CHAR_VALU
      END DO

      RETURN
      END



C_____Subroutine COMMAND
C_____Processes a PACER command
      SUBROUTINE COMMAND(STATEMENT,STAT_LEN,ANSWER,ANS_LEN,NREADS)

C_____Set up storage
      COMMON/GIO/CHANNEL2,LENGTH/MESSAGE/RECORD_STRING,TERM_MASK
      INTEGER*2 CHANNEL2,LENGTH,ANS_LEN,STAT_LEN,NREADS
      CHARACTER RECORD_STRING*80,TERM_MASK*32,STATEMENT*80,ANSWER*80

C_____Send command to PACER
      RECORD_STRING=STATEMENT
      LENGTH=STAT_LEN
      CALL WRITE_PACER_RECORD

C_____Read responses from PACER
      DO J=1,NREADS
              CALL READ_PACER_RECORD
      END DO

C_____Check if response is correct
      IF (RECORD_STRING(1:ANS_LEN).NE.ANSWER(1:ANS_LEN)) THEN
              TYPE *,'ERROR OCCURED DURING EXECUTION OF PACER COMMAND'
              TYPE *,STATEMENT(1:STAT_LEN)
              STOP
      END IF

      RETURN
      END



C_____Subroutine *INISCALB
C_____Initializes scaling variables
      SUBROUTINE INISCALB

C _____Specification statements
      COMMON /INPUTPARAM/ THETA, IC, ZETA, OMEGAN
      COMMON /FORCE/ STEP
      COMMON /SCALMAX/ XM, XDM, XDDM

C_____Initialize maximum values of X, XD, XDD
      XM = 2.*STEP
      XDM = STEP*OMEGAN
      XDDM = STEP*OMEGAN**2

      RETURN
      END
```

```
C_____________________________Subroutine SETGAINB
C_____Sets amplifier gain value for analog integrators
      SUBROUTINE SETGAINB

C_____Specification statements
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /AMPLISCALB/ GAIN

C_____Assign amplifier gain value depending on natural
C_____undamped frequency (OMEGAN) and set gain value on
C_____analog integrators
      IF(OMEGAN.LE.1.0) THEN
        GAIN = 1.0
        CALL GSSECN(IER)
      ELSE IF(OMEGAN.LE.10.0) THEN
        GAIN = 10.0
        CALL GSSECF(IER)
      ELSE
        GAIN = 100.0
        CALL GSMSN(IER)
      END IF

      RETURN
      END


C_____________________________Subroutine ADCSETB
C_____Assigns A/D channel numbers
      SUBROUTINE ADCSETB

C_____Specification statements
      DIMENSION ADC(2), IADC(2)

      COMMON /ADCONVER/ ADC, IADC, NADC

C_____Assign A/D channel numbers
      IADC(1) = 22
      IADC(2) = 23

C_____Establish how many A/D channels
      NADC = 2

      RETURN
      END


C_____________________________Subroutine SETDCAB
C_____Assigns DCA number, assigns DCA value and sets DCA's
C_____on analog computer
      SUBROUTINE SETDCAB

C_____Specification statements
      DIMENSION DCA(7), IDCA(7)

      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /AMPLISCALB/ GAIN
      COMMON /FORCE/ STEP
      COMMON /SCALMAX/ XM, XDM, XDDM

C_____Assign DCA number
      IDCA(1) = 15
      IDCA(2) = 20
      IDCA(3) = 25
      IDCA(4) = 22
      IDCA(5) = 27
      IDCA(6) = 23
      IDCA(7) = 28

C_____Assign DCA values
      DCA(1) = STEP*(OMEGAN**2.)/(10.*XDDM)
      DCA(2) = 0.0
      DCA(3) = 0.0
      DCA(4) = XDDM/(GAIN*XDM)
      DCA(5) = XDM/(GAIN*XM)
      DCA(6) = -2.*ZETA*OMEGAN*XDM/(10.*XDDM)
      DCA(7) = -(OMEGAN**2.)*XM/(10.*XDDM)

C_____Establish how many DCA's used
      NDCA = 7

C_____Set DCA's on analog computer
      CALL GWRPTR(DCA, IDCA, NDCA, 1, IER)

      RETURN
      END
```

```
C_____________________________Subroutine HILOOPB
C_____Executes high-speed-loop
      SUBROUTINE HILOOPB

C_____Specification statements
      INTEGER STARTIM(2), PRESNTIM(2), DATATIM(2)
      INTEGER USER_GET_REALTIME

      REAL LASTSAVTIM

      DIMENSION ADC(2), IADC(2)
      DIMENSION RESPONSE(1000), RESPONSD(1000), RESTIME(1000)

      COMMON /TIM/ TIME, FINALTIME, SAVINCTIM, LASTSAVTIM
      COMMON /ADCONVER/ ADC, IADC, NADC
      COMMON /CONSOLE/ NOCNLS
      COMMON /SCALMAX/ XM, XDM, XDDM
      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME
      COMMON /RESPONSDOT/ RESPONSD

      EQUIVALENCE(XS, ADC(1))
      EQUIVALENCE(XDS, ADC(2))

C_____Type final solution time on terminal screen
      TYPE *,'Solution time = ',FINALTIME,' Sec '

C_____Initialize saving time variable
      TIMSAVE = 0.0

C_____Put Analog computer in initial condition (IC) mode
      CALL GINIT(NOCNLS)

C_____Read the starting time of solution on VAX clock
      MM = USER_GET_REALTIME(STARTIM)

C_____Compute response data until final solution time is reached
      NLOOP = 0
      DO WHILE(TIME.LT.FINALTIME)

C_____Put analog computer in operate mode and read starting
C_____time only in the second loop because the VAX takes more
C_____time to execute the first loop
        NLOOP = NLOOP + 1
        IF(NLOOP.EQ.2) THEN

C_____Put Analog computer in operate (OP) mode
          CALL GOPER(NOCNLS)

C_____Read the starting time of solution on VAX clock
          MM = USER_GET_REALTIME(STARTIM)

        END IF

C_____Read data from A/D converters
        CALL GRDADS(ADC, IADC, NADC, IER)

C_____Read present time of solution on clock
        MM = USER_GET_REALTIME(PRESNTIM)

C_____Descale A/D values
        X = XS*XM
        XD = XDS*XDM

C_____Compute time difference from now to starting time
C_____Computation is  DATATIM = PRESNTIM - STARTIM
        CALL SUBQUAD(PRESNTIM, STARTIM, DATATIM)

C_____Convert DATATIM into seconds
        TIME = TIMSAVE + DATATIM(1)/10000000

C_____At every 200 seconds of solution time (TIME), save
C_____present time in variable TIMSAVE and re-initialize
C_____the starting time.  This is to overcome the
C_____possibility for array DATATIM to get numbers much
C_____larger than 2.0E9
        IF(DATATIM(1)/10000000.0.GE.200.) THEN
          TIMSAVE = TIME
          STARTIM(1) = PRESNTIM(1)
          STARTIM(2) = PRESNTIM(2)
        END IF
```

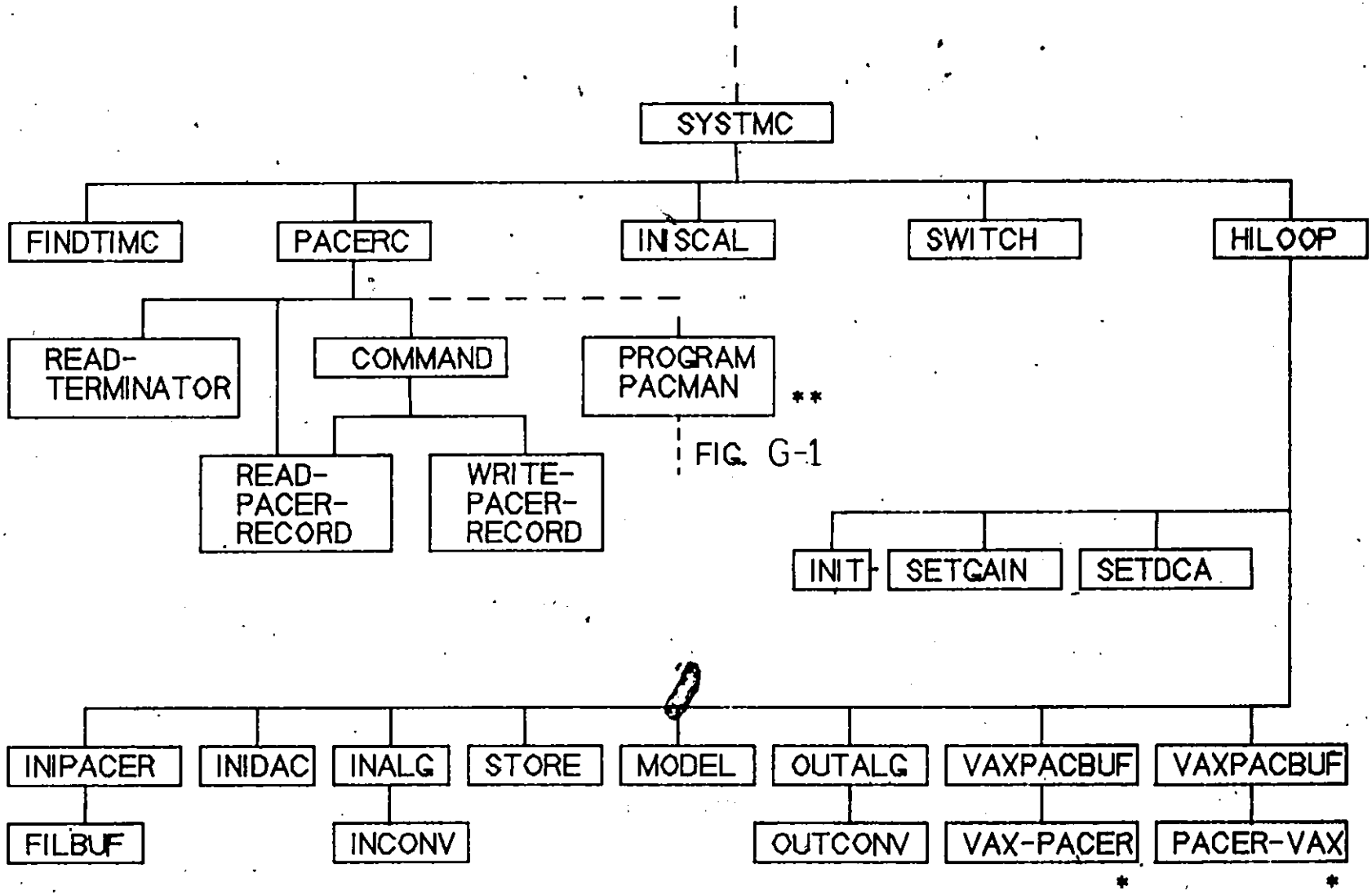
```
C_______Store response data and time data if time is past
C_______the last saved time plus the data storage sampling
C_______period
        IF((TIME.GE.LASTSAVTIM+SAVINCTIM).AND.(NLOOP.GT.1)) THEN
          NPOINT = NPOINT + 1
          RESPONSE(NPOINT) = X
          RESPONSD(NPOINT) = XD
          RESTIME(NPOINT) = TIME
          LASTSAVTIM = TIME
        END IF

      END DO

C_____Put Analog computer in IC mode
      CALL GINIT(NOCNLS)

      RETURN
      END
```

APPENDIX E

System C Documentation

Contents

- Subroutine Hierarchical Diagram
- Hierarchy
- Input-Process-Output Table of Subroutines
- Flowcharts
- Listings



* VAX-PACER AND PACER-VAX ARE VAX/HYBRID SYSTEM SUBROUTINES

** PROGRAM PACMAN RUNS ON PACER COMPUTER

Figure E–1 : Subroutine hierarchical diagram of subroutine SYSTMC

Subroutine SYSTMC Hierarchy

1. Compute T_F and T_S ... sub FINDTIMC
2. Prepare PACER computer to run PACER program PACMAN ... sub PACERC
3. Initialize x_M, $\dot{x}_M$, $\ddot{x}_M$ ... sub INISCAL
4. Set value (0/1) to relay switches R_K's ... sub SWITCH
5. Execute high-speed-loop ... sub HILOOP

2. Sub PACERC

2.1 Assign a channel to PACER

2.2 Set up terminator mask for PACER read operation ... sub READ-TERMINATOR

2.3 Obtain PACER monitor ... sub READ-PACER-RECORD

2.4 Load program PACMAN on PACER and start its execution ... sub COMMAND

5. Sub HILOOP

5.1 Initialize time variables

5.2 Set amplifier gain G ... sub SETGAIN

5.3 Set DCA values ... sub SETDCA

5.4 Set up PACER buffer ... sub INIPACER

5.5 Assign initial conditions for D/A channels ... sub INIDAC

5.6 Convert A/D data from integers to reals and descale data ... sub INALG

5.7 Store response solution data ... sub STORE

5.8 Compute model equation ... sub MODEL

5.9 Scale D/A data and convert them from reals to integers ... sub OUTALG

5.10 Send buffer to PACER ... sub VAXPACBUF

5.11 Read buffer from PACER ... sub PACVAXBUF

5.12 Communication message if end of solution was not a normal stop

Input-Process-Output Table of Subroutines

Subroutine	Inputs	Process	Outputs
SYSTMC	ζ, ω_n, π, u, N_T, N_S θ, T_c	Fig. E-2	DSR
FINDTIMC	ω_n, π, N_T, N_S	Eqs. 50, 51	T_F, T_S
INISCAL	ω_n, u	Eqs. 66-68	x_M, $\dot{x}_M$, $\ddot{x}_M$
SWITCH		R_K=0, K=34, 39	R_{34}, R_{39}
HILOOP	T_F	Fig. E-4	DSR
SETGAIN	θ, x_M, $\dot{x}_M$, $\ddot{x}_M$	Fig. E-5	G
SETDCA	θ, u, G, x^0, $\dot{x}^0$ x_M, $\dot{x}_M$, $\ddot{x}_M$	Eqs. 73-78	D_{30}, D_{31}, D_{32} D_{35}, D_{36}, D_{37}
INALG	x_S, $\dot{x}_S$, x_M, $\ddot{x}_M$	Eqs. 79, 80	x, $\dot{x}$
INCONV	x_S, $\dot{x}_S$, x_M, $\ddot{x}_M$	Eqs. 87, 88, x_M=1.5x_M, $\ddot{x}_M$=1.5$\ddot{x}_M$	x_S, $\dot{x}_S$, x_M, $\ddot{x}_M$
STORE	T_S, x, t, T_c	Fig. E-6	DSR
MODEL	ω_n, ζ, u, x, $\dot{x}$	Equ. 13	$\ddot{x}$
OUTALG	$\dot{x}$, $\ddot{x}$, x_M, $\ddot{x}_M$	Eqs. 83, 84	$\dot{x}_S$, $\ddot{x}_S$
OUTCONV	$\dot{x}_S$, $\ddot{x}_S$, x_M, $\ddot{x}_M$	Eqs. 85, 86, $\dot{x}_M$=1.5$\dot{x}_M$, $\ddot{x}_M$=1.5$\ddot{x}_M$	$\dot{x}_S$, $\ddot{x}_S$, x_M, $\ddot{x}_M$

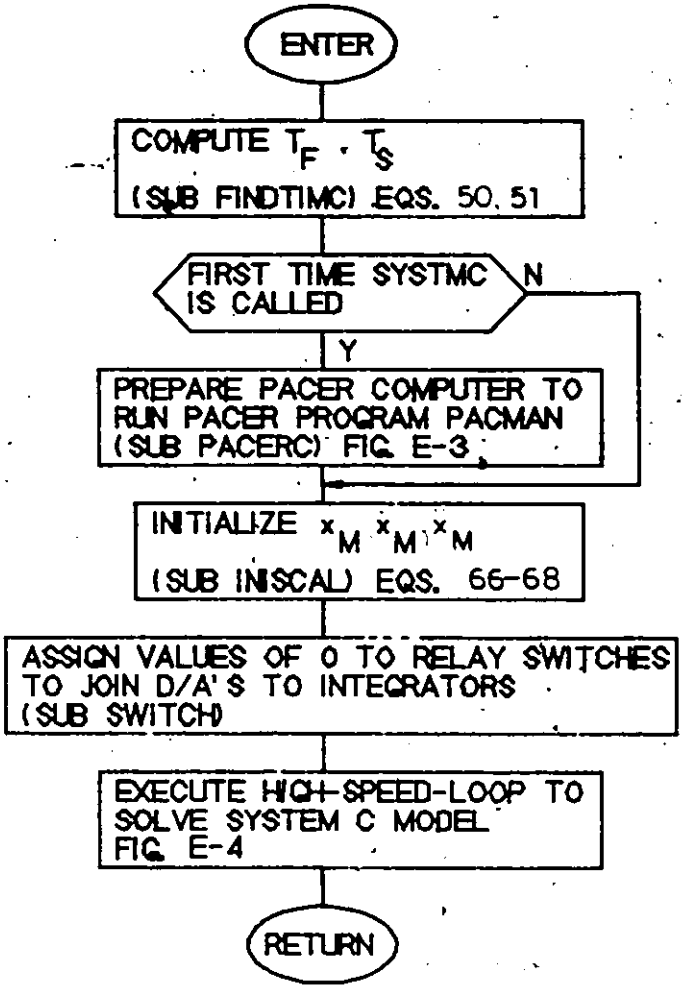


Figure E–2 : Subroutine SYSTMC flowchart

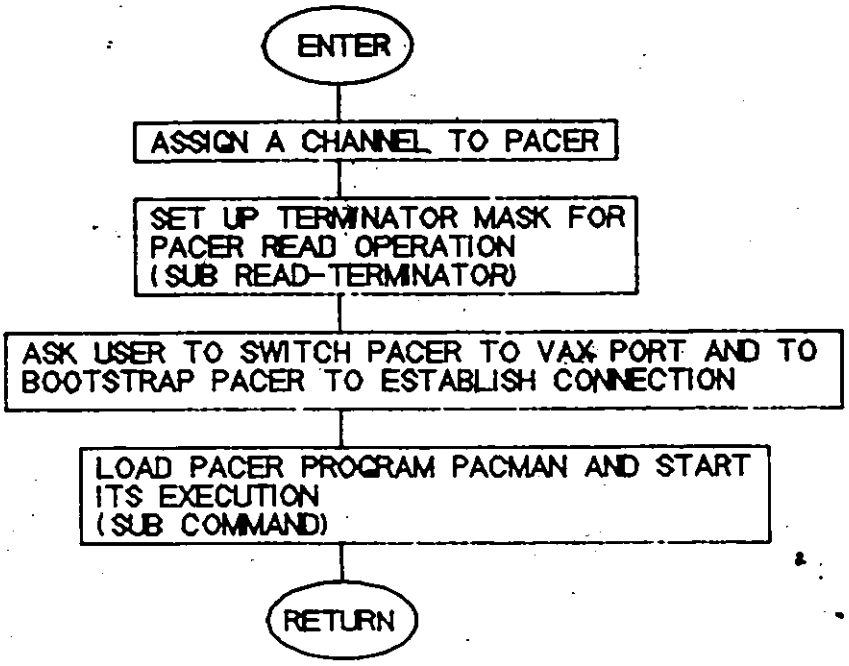


Figure E-3 : Subroutine PACERC flowchart

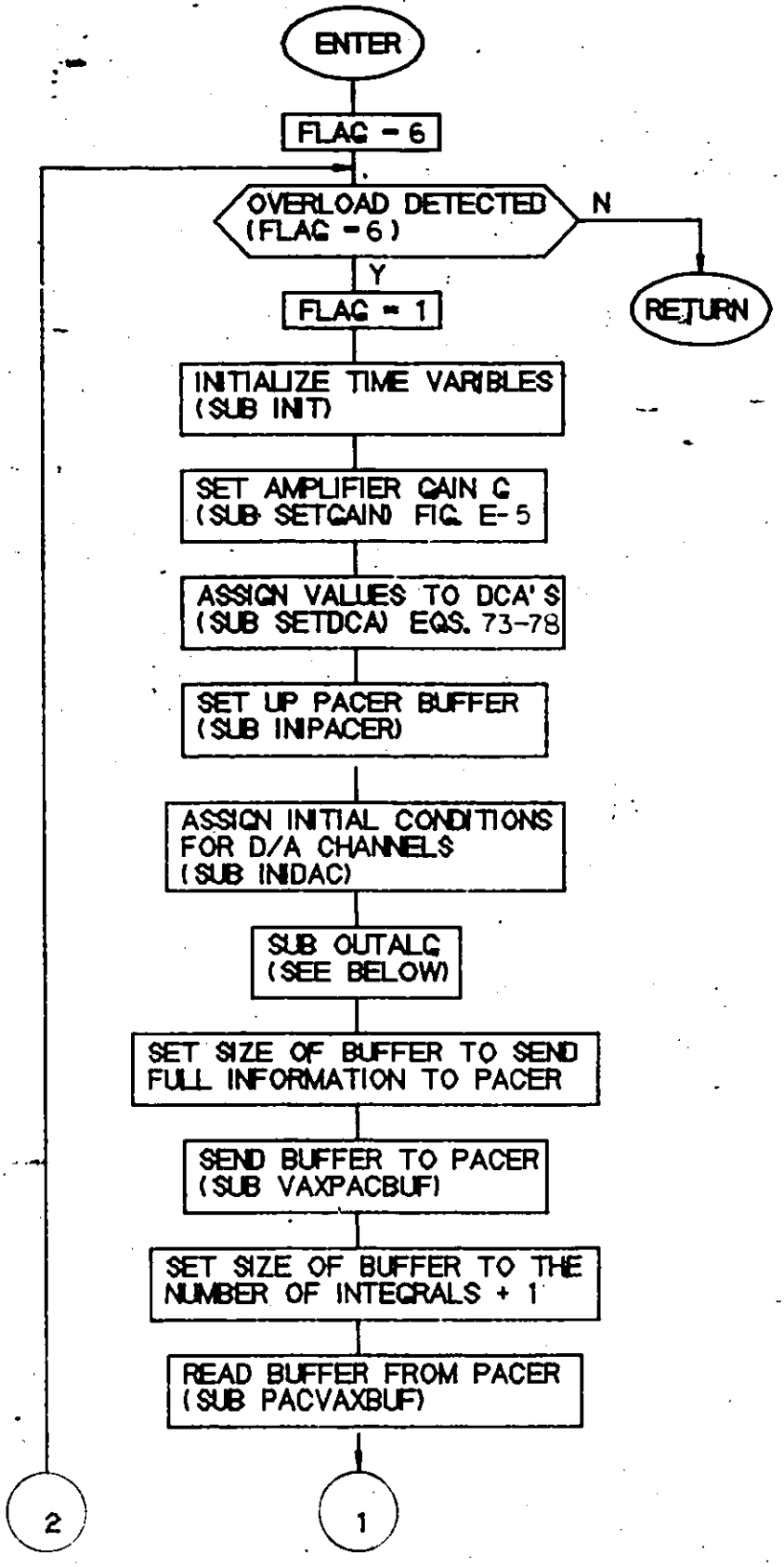


Figure E–4 : Subroutine HILOOP flowchart

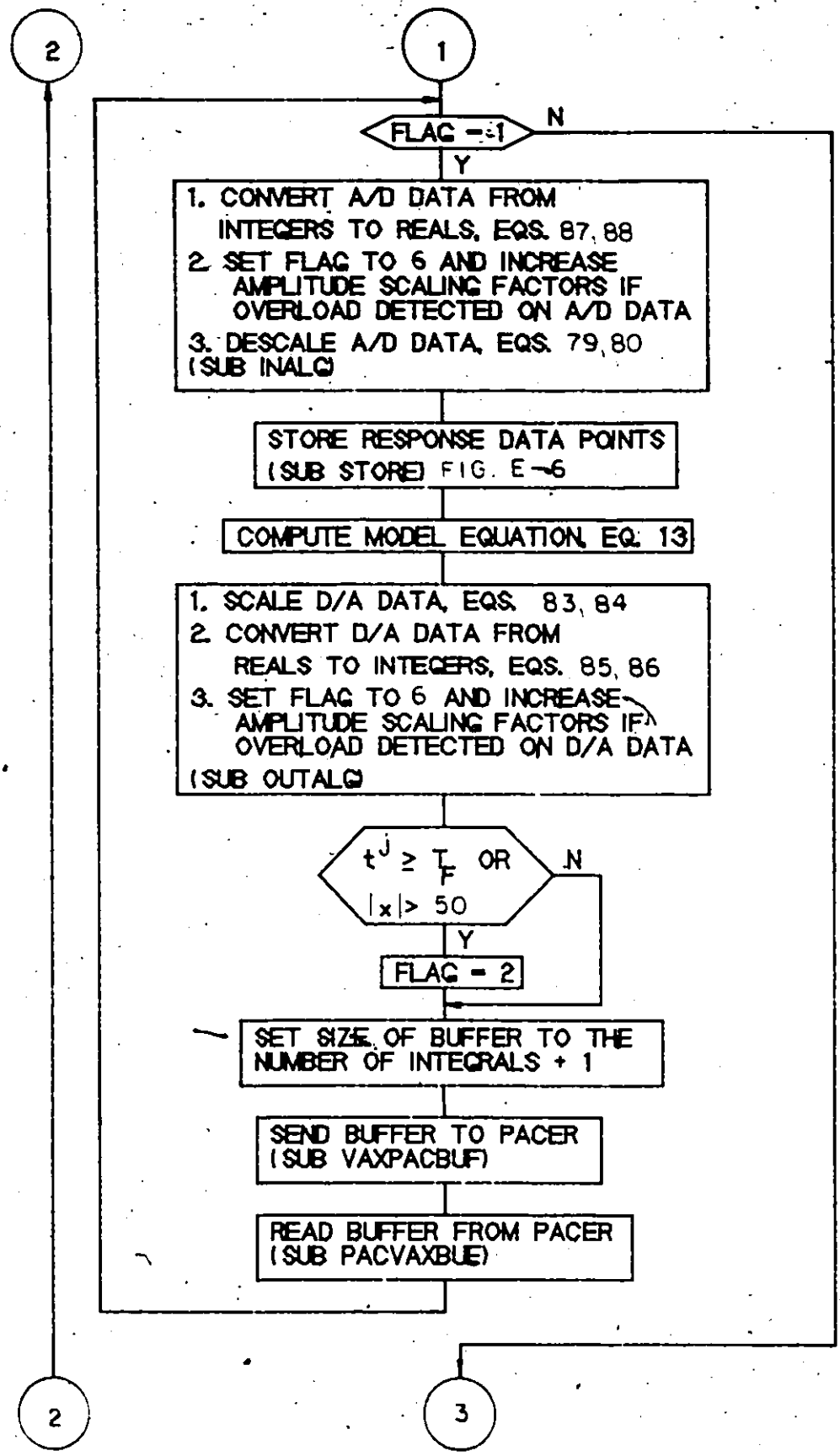


Figure E-4 : (cont'd)

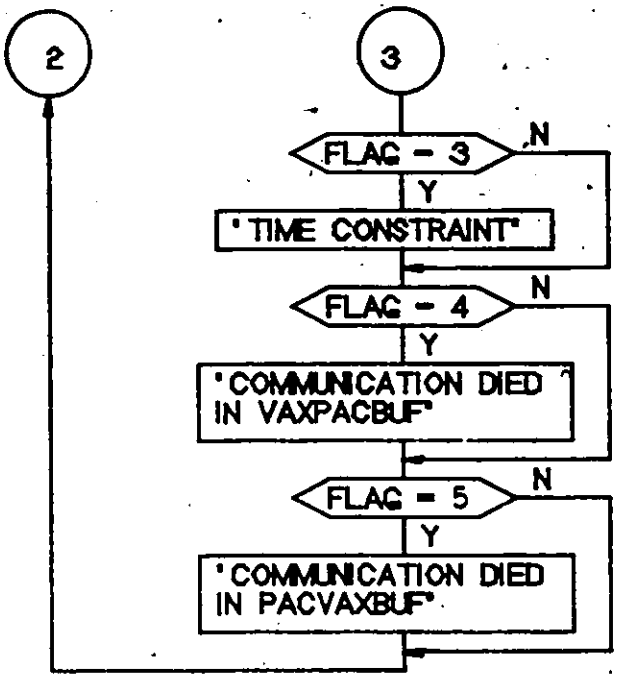


Figure E–4 : (End)

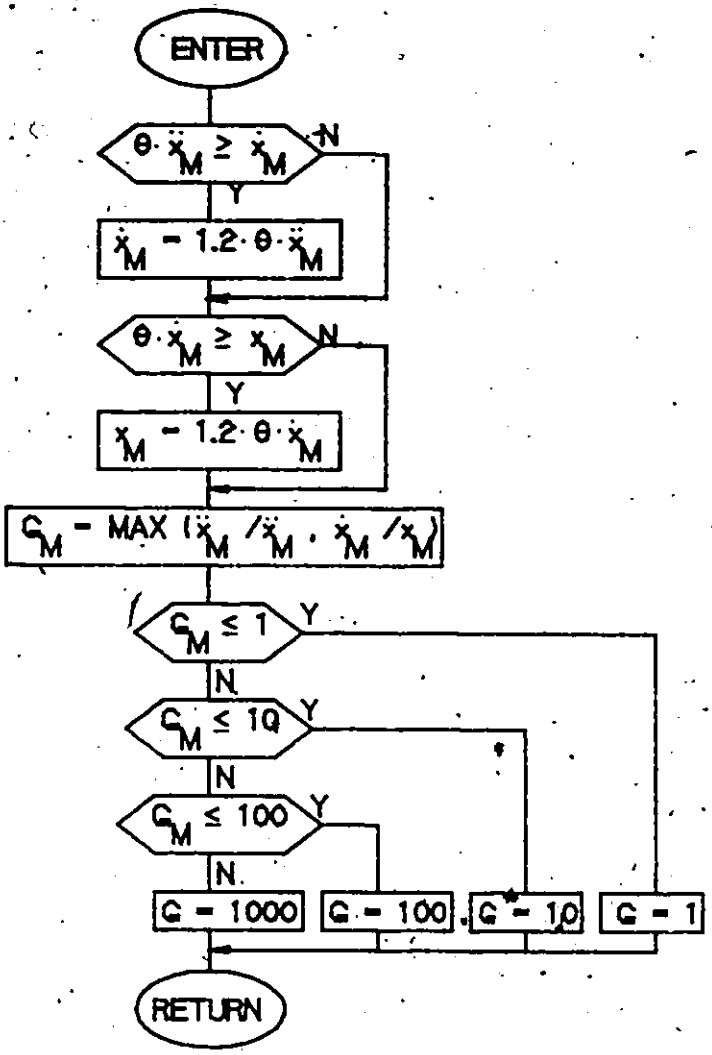


Figure E–5 : Subroutine SETGAIN flowchart

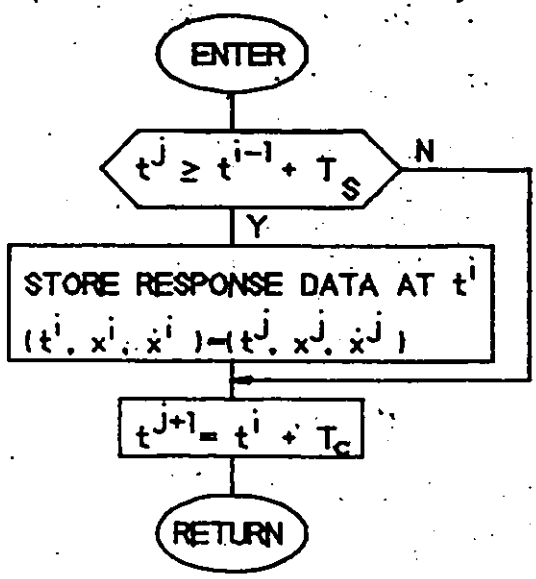


Figure E–6 : Subroutine STORE flowchart

```
C_____________________________Subroutine SYSTMC
C_____Computes response data points for the sampled-data
C_____system (Hybrid computer)
      SUBROUTINE SYSTMC

C_____Specification statements
      LOGICAL FIRSTRUNB, FIRSTRUNC

      COMMON /FIRSTRUN/-FIRSTRUNB, FIRSTRUNC

C_____Start PACER program PACMAN the first time subroutine
C_____SYSTMC is called or if subroutine SYSTMB has been
C_____called in the same main program run
      CALL SETUP_DR11B
      IF(FIRSTRUNC) THEN
        CALL PACERC
      END IF

C_____Compute final solution time and data storage sampling period
      CALL FINDTIMC

C_____Initialize scaling variables
      CALL INISCAL

C_____Assign values to relay switches (0/1) to define
C_____the circuit on analog patchboard
      CALL SWITCH

C_____Execute high-speed-loop to solve System C model
      CALL HILOOP

      CALL RELEASE_DR11B

      RETURN
      END


C_____________________________Subroutine FINDTIMC
C_____Computes final solution time for a fixed number (NT) of cycles
C_____and computes data storage sampling period
      SUBROUTINE FINDTIMC

C_____Specification statements
      REAL LASTSAVTIM

      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /CONSTANTS/ XF, PI
      COMMON /NUMTIM/ NT, NS
      COMMON /TIM/ TIME, FINALTIME, SAVINCTIM, LASTSAVTIM


C_____Compute final solution time
      FINALTIME = (NT*2 *PI)/OMEGAN

C_____Compute data storage sampling period
      SAVINCTIM = (2 *PI)/(NS*OMEGAN)

      RETURN
      END


C_____________________________Subroutine PACERC
C_____Loads program PACMAN on PACER and starts its execution
C_____to enable VAX/HYBRID runing for System C
      SUBROUTINE PACERC

C_____Specification statements
      EXTERNAL SS$_NORMAL
      INTEGER*4 SYS$ASSIGN, SYS$ALLOC
      INTEGER*2 CHANNEL2, LENGTH
      CHARACTER RECORD_STRING*80, TERM_MASK*32

      COMMON/QIO/CHANNEL2, LENGTH/MESSAGE/RECORD_STRING, TERM_MASK

C_____Assign a channel to PACER
      I=SYS$ASSIGN('PACER100:',CHANNEL2,,)
      IF(I .NE. %LOC(SS$_NORMAL)) THEN
        TYPE 100, I
      END IF

C_____Set up terminator mask for PACER read operation
      CALL READ_TERMINATOR
```

```
C_____Give instructions to user on how to obtain PACER monitor
      TYPE *, 'Switch PACER to VAX port and press <RETURN>'
      ACCEPT *
      TYPE *
      TYPE *, 'Bootstrap PACER to establish connection'
      TYPE *,' (message when connection established)'
      CALL READ_PACER_RECORD
      IF (RECORD_STRING(1:2) EQ. 'M_') THEN
        TYPE *
        TYPE *,'        *    PACER CONNECTION ESTABLISHED    *'
      END IF

C_____Load PACER program (PACMAN) and start execution
      CALL COMMAND('@L.PACMAN.21'          ,12,'LD'     ,2 ,6 )
      CALL COMMAND('@R..60000.65000'       ,15,' '      ,0 ,3 )
      CALL COMMAND('@G.1000'               ,7 ,' '      ,0 ,2 )

      TYPE *
      TYPE *,'          *   VAX/PACER NOW RUNING    *'
      TYPE *

C_____Format statement
  100 FORMAT(' CANNOT ASSIGN A CHANNEL TO PACER. STATUS = ',ZB)

      RETURN
      END



C_____________________________Subroutine INISCAL
C_____Initialize scaling variables
      SUBROUTINE INISCAL

C_____Specification statements
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /FORCE/ STEP
      COMMON /SCALMAX/ XM, XDM, XDDM

C_____Initialize maximum values of X, XD and XDD
C_____Set XM smaller for a large damping for better scaling
      IF(ZETA.GE.1.0) THEN
        XM = 1.2*STEP
      ELSE
        XM = 1.2*(2.*STEP)
      END IF

      XDM = 1.2*(STEP*OMEGAN)
      XDDM = 1.2*(STEP*OMEGAN**2.)

      RETURN
      END



C_____________________________Subroutine SWITCH
C_____Assigns values to relay switches (0/1) to define
C_____the circuit on analog patchboard  For System C.
C_____every switches are opened (0) to connect every D/A's
C_____to the integrators
      SUBROUTINE SWITCH

C_____Specification statements
      INTEGER*2 RELAY(15)

      COMMON /RELAYCHANL/ RELAY

C_____Subroutine statements
      DO I=1,15
        RELAY(I) = 0
      END DO

      RETURN
      END
```

```
C_____________________________Subroutine HILOOP
C_____Executes high-speed-loop
      SUBROUTINE HILOOP

C_____Specification statements
      REAL LASTSAVTIM
      INTEGER*2 MAXINT, NUMINT
      INTEGER*4 LINK_BLOCK(5), IERR
      INTEGER*2 OUT_SIZE, IN_SIZE, OUT_BUFFER(256), IN_BUFFER(256)
      INTEGER*2 FLAG, CHANNEL, END_BLOCK

      COMMON /DR11B_BUF/ LINK_BLOCK, OUT_SIZE, IN_SIZE,
     -                   OUT_BUFFER, IN_BUFFER, IERR, CHANNEL,
     -                   END_BLOCK
      COMMON /NUMINTGRL/ NUMINT, MAXINT
      COMMON /FUNCTIONS/ X,XD,XDD
      COMMON /FLG/ FLAG
      COMMON /TIM/ TIME, FINALTIME, SAVINCTIM, LASTSAVTIM

C_____Type final solution time on terminal screen
      TYPE *,'Solution time =',FINALTIME,' Sec '

C_____Execute high-speed-loop as often as there is an overload
C_____detected
      FLAG = 6
      DO WHILE(FLAG EQ 6)

C _____Set FLAG to 1 to indicate to PACER that data
C _____will be transmitted
        FLAG = 1

C_____Initialize constants
        CALL INIT

C_____Determine amplifier gain value for analog integrators
        CALL SETGAIN

C_____Assign values to DCA's
        CALL SETDCA

C_____Set up BUFFER to send to PACER and start PACER program
        CALL INIPACER

C_____Assign initial conditions for D/A channels
        CALL INIDAC

C_____Scale D/A data and convert real values to scaled fractions
C_____before sending them to PACER
        CALL OUTALG

C_____Set size of buffer to send the full information
C_____to PACER
        OUT_SIZE = 5*MAXINT + 4

C _____Send BUFFER to PACER
        CALL VAXPACBUF

C_____Set size of buffer to the number of integrals + 1
        IN_SIZE = NUMINT + 1

C_____Read BUFFER from PACER
        CALL PACVAXBUF

C_____Execute high-speed-loop while FLAG is 1
        DO WHILE(FLAG EQ 1)

C_____Convert PACER values from scaled fractions
C_____to real values and descale A/D data
          CALL INALG

C_____Store response data points and increment time
C_____variable
          CALL STORE

C_____Compute model equation
          CALL MODEL

C_____Scale D/A data and convert real values to scaled
C_____fractions before sending them to PACER
          CALL OUTALG
```

```
C__________Test if time has reached final solution time
C__________or if response data are getting to large and set
C__________FLAG to 2 to indicate a normal stop of the solution
          IF((TIME.GT.FINALTIME) .OR. (X.GT.500.) .OR.
     -       (X.LT.-500.)) THEN
            FLAG = 2
          END IF

C__________Set size of buffer to the number of integrals + 1
          OUT_SIZE = NUMINT + 1

C__________Send BUFFER to PACER
          CALL VAXPACBUF

C__________Set size of buffer to the number of integrals + 1
          IN_SIZE = NUMINT + 1

C__________Read BUFFER from PACER
          CALL PACVAXBUF

        END DO

C_________The value of FLAG=3 comes from the PACER program. It means
C_________that the high-speed-loop took more time than required time
C_________of computation (TC)
        IF(FLAG.EQ.3) THEN
          TYPE *
          TYPE *,'     *******   TIME CONSTRAINT   *******'
          TYPE *

C_________The value of FLAG=4 comes from subroutine VAXPACBUF.
C_________The value of FLAG=5 comes from subroutine PACVAXBUF.
C_________It means that for some unknown exact reason, the buffer
C_________cannot be sent to PACER or read from PACER
        ELSE IF(FLAG.EQ.4 .OR. FLAG.EQ.5) THEN
          TYPE *
          TYPE *,'     ****** COMMUNICATION DIED IN VAX TO PACER ******'
          TYPE *,'     ****** OR PACER TO VAX TRANSFER.          ******'
          TYPE *

        END IF

      END DO

      RETURN
      END


C_____________________________Subroutine INIT
C_____Initializes constants
      SUBROUTINE INIT

C_____Specification statements
      REAL LASTSAVTIM

      DIMENSION RESPONSE(1000), RESTIME(1000)

      COMMON /TIM/ TIME, FINALTIME, SAVINCTIM, LASTSAVTIM
      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME

C_____Initialize resppnse data pointer
      NPOINT= 0

C_____Initialize time variable
      TIME = 0.0

C_____Initialize to a very small value, the variable
C_____representing the last time data that has been stored
      LASTSAVTIM = -1.0E30

      RETURN
      END
```

```
C_____________________________Subroutine SETGAIN
C_____Determines amplifier gain value G for analog integrators
      SUBROUTINE SETGAIN

C_____Specification statements
      INTEGER*2 GAIN

      COMMON /SCALMAX/ XM, XDM, XDDM
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /AMPLISCAL/ GAIN, GAINA

C_____Adjust XM and XDM if compensation DCA's are overscaled
      IF(THETA*XDDM GE XDM) THEN
        XDM = 1 2*THETA*XDDM
      END IF

      IF(THETA*XDM GE XM) THEN
        XM = 1 2*THETA*XDM
      END IF

C_____Determine maximum of ratios in front of integrators
      GAINMAX = MAX(XDDM/XDM, XDM/XM)

C_____Set amplifier gain depending on GAINMAX
      IF(GAINMAX LE 1 0) THEN
        GAIN = 1
      ELSE IF(GAINMAX LE 10 0) THEN
        GAIN = 10
      ELSE IF(GAINMAX LE 100 0) THEN
        GAIN = 100
      ELSE
        GAIN = 1000
      END IF

      RETURN
      END



C_____________________________Subroutine SETDCA
C_____Assign values to DCA's
      SUBROUTINE SETDCA

C_____Specification statements
      INTEGER*2 GAIN

      DIMENSION IDCA(6)
      DIMENSION DCA(45)

      COMMON /SCALMAX/ XM, XDM, XDDM
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /AMPLISCAL/ GAIN, GAINA
      COMMON /DCACHANL/ DCA, NDCA


C_____Assign initial conditions on DCA's
      DCA(1) = 0 0
      DCA(4) = 0 0

C_____Assign compensation factor values on DCA's
      DCA(3) = THETA*XDM/XM
      DCA(6) = THETA*XDDM/XDM

C_____Assign values to DCA's in front of integrators
      DCA(2) = -XDM/(XM*GAIN)
      DCA(5) = -XDDM/(XDM*GAIN)

C_____Assign the rest of the 45 DCA's to zero
      DO I = 7, 45
        DCA(I) = 0 0
      END DO

      RETURN
      END
```

```
C_____________________________Subroutine INIPACER
C_____Initializes some constants and fills the BUFFER
      SUBROUTINE INIPACER

C_____Specification statements
      INTEGER*2 MAXINT, NUMINT

      DIMENSION DAC(15), ADC(15), DCA(45)

      COMMON /NUMINTGRL/ NUMINT, MAXINT
      COMMON /DACHANL/ DAC, NDAC
      COMMON /ADCHANL/ ADC, NADC
      COMMON /DCACHANL/ DCA, NDCA

C_____Set maximum number of integrators, set number of
C_____integrators used for System C, set number of
C_____D/A and A/D channels for this problem and set
C_____number of DCA
      MAXINT = 15
      NUMINT = 2
      NDAC = NUMINT
      NADC = NUMINT
      NDCA = 3*NUMINT

C_____Fill the BUFFER to send to PACER
      CALL FILBUF

      RETURN
      END



C_____________________________Subroutine FILBUF
C_____Fills the BUFFER with variables that are to be
C_____sent to PACER
      SUBROUTINE FILBUF

C_____Specification statements
      INTEGER*2 MAXINT, NUMINT, GAIN, RELAY(15)
      INTEGER*4 LINK_BLOCK(5), IERR
      INTEGER*2 OUT_SIZE, IN_SIZE, OUT_BUFFER(256), IN_BUFFER(256)
      INTEGER*2 CHANNEL, END_BLOCK

      DIMENSION DCA(45)

      COMMON /DR11B_BUF/ LINK_BLOCK, OUT_SIZE, IN_SIZE,
     -                   OUT_BUFFER, IN_BUFFER, IERR, CHANNEL,
     -                   END_BLOCK
      COMMON /NUMINTGRL/ NUMINT, MAXINT
      COMMON /DCACHANL/ DCA, NDCA
      COMMON /AMPLISCAL/ GAIN, GAINA
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /RELAYCHANL/ RELAY

C_____Fill the BUFFER except for D/A and A/D channel values,
C_____convert real variables DCA and TC to scaled fraction values
      DO I=1,MAXINT
        OUT_BUFFER(I+1) = 0
      END DO
      DO I=1,3*MAXINT
        OUT_BUFFER(I+MAXINT+1) = ININT(DCA(I)*32768.)
      END DO
      OUT_BUFFER(4*MAXINT+2) = ININT(TC*32768.)
      OUT_BUFFER(4*MAXINT+3) = NUMINT
      OUT_BUFFER(4*MAXINT+4) = GAIN
      DO I=1,MAXINT
        OUT_BUFFER(I+4*MAXINT+4) = RELAY(I)
      END DO

      RETURN
      END



C_____________________________Subroutine INIDAC
C_____Assigns initial conditions for D/A channels
      SUBROUTINE INIDAC

C_____Specification statements
      COMMON /FUNCTIONS/ X, XD, XDD

C_____Assign initial conditions for D/A channels
      XD = 0.0
      XDD = 0.0

      RETURN
      END
```

```
C_____Subroutine INALG
C_____Converts PACER values from scaled fractions
C_____to real values and descales A/D data
      SUBROUTINE INALG

C_____Specification statements
      DIMENSION ADC(15)

      COMMON /ADCHANL/ ADC, NADC
      COMMON /SCALMAX/ XM, XDM, XDDM
      COMMON /FUNCTIONS/ X, XD, XDD

      EQUIVALENCE(ADC(1),XS)
      EQUIVALENCE(ADC(2),XDS)

C_____Convert PACER values from scaled fractions
C_____to real values
      CALL INCONV

C_____Descale A/D data
      X = XS*XM
      XD = XDS*XDM

      RETURN
      END



C_____Subroutine INCONV
C_____Converts PACER values from scaled fractions
C_____to real values and checks if one of these values are
C_____overloading (>1) or close to
      SUBROUTINE INCONV

C_____Specification statements
      INTEGER*4 LINK_BLOCK(5), IERR
      INTEGER*2 OUT_SIZE, IN_SIZE, OUT_BUFFER(256), IN_BUFFER(256)
      INTEGER*2 FLAG, CHANNEL, END_BLOCK

      DIMENSION ADC(15)

      COMMON /DR11B_BUF/ LINK_BLOCK, OUT_SIZE, IN_SIZE,
     -                   OUT_BUFFER, IN_BUFFER, IERR, CHANNEL,
     -                   END_BLOCK
      COMMON /ADCHANL/ ADC, NADC
      COMMON /SCALMAX/ XM, XDM, XDDM
      COMMON /FLG/ FLAG

C_____Convert A/D values
      ADC(1) = FLOATI(IN_BUFFER(2))/32768.0
      ADC(2) = FLOATI(IN_BUFFER(3))/32768.0

C_____Set FLAG to 6 and adjust scaling factors if an
C_____overload is detected
      IF((ADC(1).GT.0.95).OR.(ADC(1).LT.-0.95)) THEN
        XM = 1.5*XM
        FLAG = 6
      END IF
      IF((ADC(2).GT.0.95).OR.(ADC(2).LT.-0.95)) THEN
        XDM = 1.5*XDM
        FLAG=6
      END IF

      RETURN
      END
```

```
C_____________________________Subroutine STORE
C_____Store response data points and increment time variable
      SUBROUTINE STORE

C_____Specification statements
      REAL LASTSAVTIM

      DIMENSION RESPONSE(1000), RESPONSD(1000), RESTIME(1000)

      COMMON /FUNCTIONS/ X, XD, XDD
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /TIM/ TIME, FINALTIME, SAVINCTIM, LASTSAVTIM
      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME
      COMMON /RESPONSDOT/ RESPONSD

C_____Store response data and time data if time
C_____is past the last saved-time plus the data-storage
C_____sampling period
      IF(TIME.GE.LASTSAVTIM+SAVINCTIM) THEN
        NPOINT = NPOINT + 1
        RESPONSE(NPOINT) = X
        RESPONSD(NPOINT) = XD
        RESTIME(NPOINT) = TIME
        LASTSAVTIM = TIME
      END IF

C_____Increment time variable
      TIME = TIME + TC

      RETURN
      END


C_____________________________Subroutine MODEL
C_____Computes model equation
      SUBROUTINE MODEL

C_____Specification statements
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /FORCE/ STEP
      COMMON /FUNCTIONS/ X, XD, XDD

C_____Compute equation
      XDD = -(OMEGAN**2)*X - 2.*ZETA*OMEGAN*XD
     -      + (OMEGAN**2)*STEP

      RETURN
      END


C_____________________________Subroutine OUTALG
C_____Scales D/A data and converts real values to scaled fractions
C_____before sending them to PACER
      SUBROUTINE OUTALG

C_____Specification statements
      DIMENSION DAC(15)

      COMMON /FUNCTIONS/ X, XD, XDD
      COMMON /SCALMAX/ XM, XDM, XDDM
      COMMON /DACHANL/ DAC, NDAC

      EQUIVALENCE(XDAS,DAC(1))
      EQUIVALENCE(XDDAS,DAC(2))

C_____Scale D/A data
      XDAS = -XD/XDM
      XDDAS = -XDD/XDDM

C_____Convert real values to scaled fractions
C_____before sending them to PACER and check for an
C_____overload on analog computer
      CALL OUTCONV

      RETURN
      END
```

```
C_____Subroutine OUTCONV
C_____Converts real values to scaled fractions before
C_____sending them to PACER and checks if one of these
C_____values are overloading (>1) or close to
      SUBROUTINE OUTCONV

C_____Specification statements
      INTEGER*4 LINK_BLOCK(5), IERR
      INTEGER*2 OUT_SIZE, IN_SIZE, OUT_BUFFER(256), IN_BUFFER(256)
      INTEGER*2 FLAG, CHANNEL, END_BLOCK

      DIMENSION DAC(15)

      COMMON /DR11B_BUF/ LINK_BLOCK, OUT_SIZE, IN_SIZE,
     -                   OUT_BUFFER, IN_BUFFER, IERR, CHANNEL,
     -                   END_BLOCK
      COMMON /DACHANL/ DAC, NDAC
      COMMON /SCALMAX/ XM, XDM, XDDM
      COMMON /FLG/ FLAG

C_____Set FLAG to 6 and adjust scaling factors if an
C_____overload is detected
      IF(FLAG NE 6) THEN
        IF((DAC(1) GT 0 95) OR (DAC(1) LT -0 95)) THEN
          XDM = 1 5*XDM
          FLAG = 6
        END IF
        IF((DAC(2) GT 0 95) OR (DAC(2) LT -0 95)) THEN
          XDDM = 1 5*XDDM
          FLAG=6
        END IF
      END IF

C_____Convert D/A values if no overload detected
      IF(FLAG NE 6) THEN
        OUT_BUFFER(2) = ININT(DAC(1)*32768 )
        OUT_BUFFER(3) = ININT(DAC(2)*32768 )
      END IF

      RETURN
      END



C_____Subroutine VAXPACBUF
C_____Sends a block of data (OUT_BUFFER) to PACER
      SUBROUTINE VAXPACBUF

C_____Specification Statements
      EXTERNAL SS$_NORMAL

      INTEGER*4 LINK_BLOCK(5), IERR
      INTEGER*2 OUT_SIZE, IN_SIZE, OUT_BUFFER(256), IN_BUFFER(256)
      INTEGER*2 FLAG, CHANNEL, END_BLOCK

      COMMON /DR11B_BUF/ LINK_BLOCK, OUT_SIZE, IN_SIZE,
     -                   OUT_BUFFER, IN_BUFFER, IERR, CHANNEL,
     -                   END_BLOCK
      COMMON /FLG/ FLAG

C_____Assign FLAG value to the first element of the buffer
      OUT_BUFFER(1) = FLAG

C_____Send BUFFER to PACER and check if PACER was ready
C_____Set FLAG to 4 if no success
      CALL VAX_PACER
      IF(IERR NE %LOC(SS$_NORMAL)) THEN
        FLAG = 4
      END IF

      RETURN
      END
```

```
C_____________________________Subroutine PACVAXBUF
C_____Reads a block of data (IN_BUFFER) from PACER
      SUBROUTINE PACVAXBUF

C_____Specification Statements
      EXTERNAL SS$_NORMAL

      INTEGER*4 LINK_BLOCK(5), IERR
      INTEGER*2 OUT_SIZE, IN_SIZE, OUT_BUFFER(256), IN_BUFFER(256)
      INTEGER*2 FLAG, CHANNEL, END_BLOCK

      COMMON /DR11B_BUF/ LINK_BLOCK, OUT_SIZE, IN_SIZE,
     -                   OUT_BUFFER, IN_BUFFER, IERR, CHANNEL,
     -                   END_BLOCK
      COMMON /FLG/ FLAG

C_____Read BUFFER from PACER and check if PACER was ready.
C_____Set FLAG to 5 if no success
      CALL PACER_VAX
      IF(IERR .NE. %LOC(SS$_NORMAL)) THEN
        FLAG = 5
      ELSE

C_________Assign the first element of the buffer to FLAG
        FLAG = IN_BUFFER(1)

      END IF

      RETURN
      END
```

APPENDIX F

System D Documentation

Contents

- Flowchart
- Listing

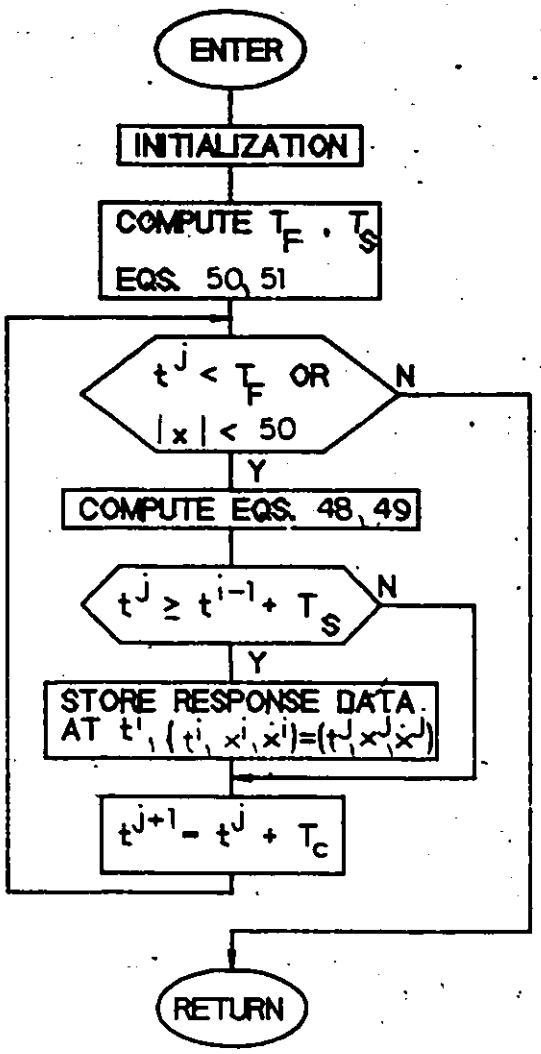


Figure F—1 : Subroutine SYSTMD flowchart

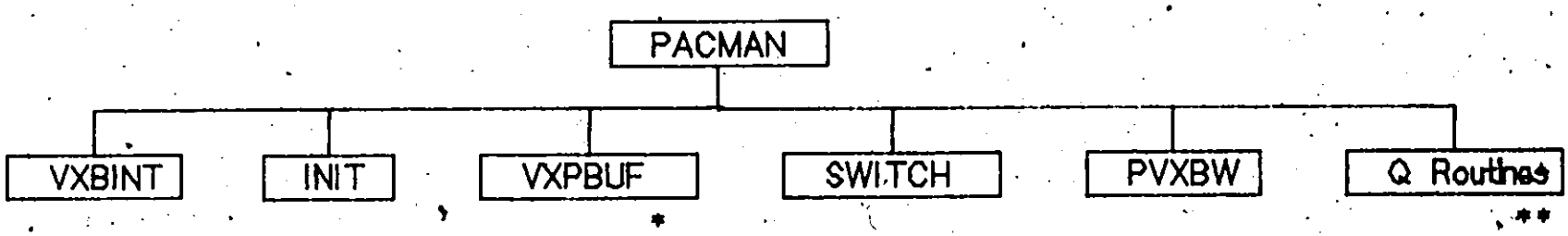
```
C_____Subroutine SYSTMD
C_____Computes response data points for the
C_____discrete-time equivalent to sampled-data system
      SUBROUTINE SYSTMD

C_____Specification statements
      DOUBLE PRECISION X, XD, X1, X2
      DOUBLE PRECISION LASTSAVTIM, TIME
      DIMENSION RESPONSE(1000), RESPONSD(1000), RESTIME(1000)
      DIMENSION X1(0:1), X2(0:1)

      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /CONSTANTS/ XF, PI
      COMMON /FORCE/ STEP
      COMMON /NUMTIM/ NT, NS
      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME
      COMMON /RESPONSDOT/ RESPONSD

C_____Initialization
      STEPO = 0
      X = 0.0
      XD = 0.0
      X1(0) = 0.0
      X2(0) = 0.0
      X1(1) = 0.0
      X2(1) = 0.0
      NPOINT = 1
      TIME = TC

C_____Compute final solution time and data storage sampling
C_____period and initialize last time data value that has
C_____been stored
      FINALTIME = (NT*2.*PI)/OMEGAN
      SAVINCTIM = (2.*PI)/(NS*OMEGAN)
      LASTSAVTIM = 0.0

C_____Store first response data point and first time data
      RESPONSE(NPOINT) = 0.0
      RESPONSD(NPOINT) = 0.0
      RESTIME(NPOINT) = 0.0

C_____Compute response data until final solution time or until
C_____gets too large
      DO WHILE((TIME.LE.FINALTIME).AND.(ABS(X).LT.500.))

C_____Compute model equations
        X = X1(1) + THETA*X2(1) + (TC - THETA)*X2(0)
        XD = -THETA*(OMEGAN**2.)*X1(1) + (1 - 2.*THETA
     -       *ZETA*OMEGAN)*X2(1) + (TC - THETA)
     -       *(-(OMEGAN**2)*X1(0) - 2.*ZETA*OMEGAN
     -       *X2(0)) + STEP*(OMEGAN**2)*THETA
     -       + STEPO*(TC - THETA)*(OMEGAN**2.)

C_____Update temporary variables
        STEPO = STEP
        X1(0) = X1(1)
        X1(1) = X
        X2(0) = X2(1)
        X2(1) = XD

C_____Store response data and time data if time is
C_____past the last saved time plus the data storage
C_____sampling period
        IF(TIME.GE.LASTSAVTIM + SAVINCTIM) THEN
          NPOINT = NPOINT + 1
          RESPONSE(NPOINT) = X
          RESPONSD(NPOINT) = XD
          RESTIME(NPOINT) = TIME
          LASTSAVTIM = TIME
        END IF

C_____Increment time variable
        TIME = TIME + TC

      END DO

      RETURN
      END
```

APPENDIX C

Program PACMAN Documentation

Contents

- Subroutine Hierarchical Diagram
- Flowchart
- Listings



* VXPBUF IS AN INTERUPT SUBROUTINE

** Q ROUTINES ARE PACER/HYBRID SYSTEM SUBROUTINES

Figure G–1 : Subroutine hierarchical diagram of program PACMAN

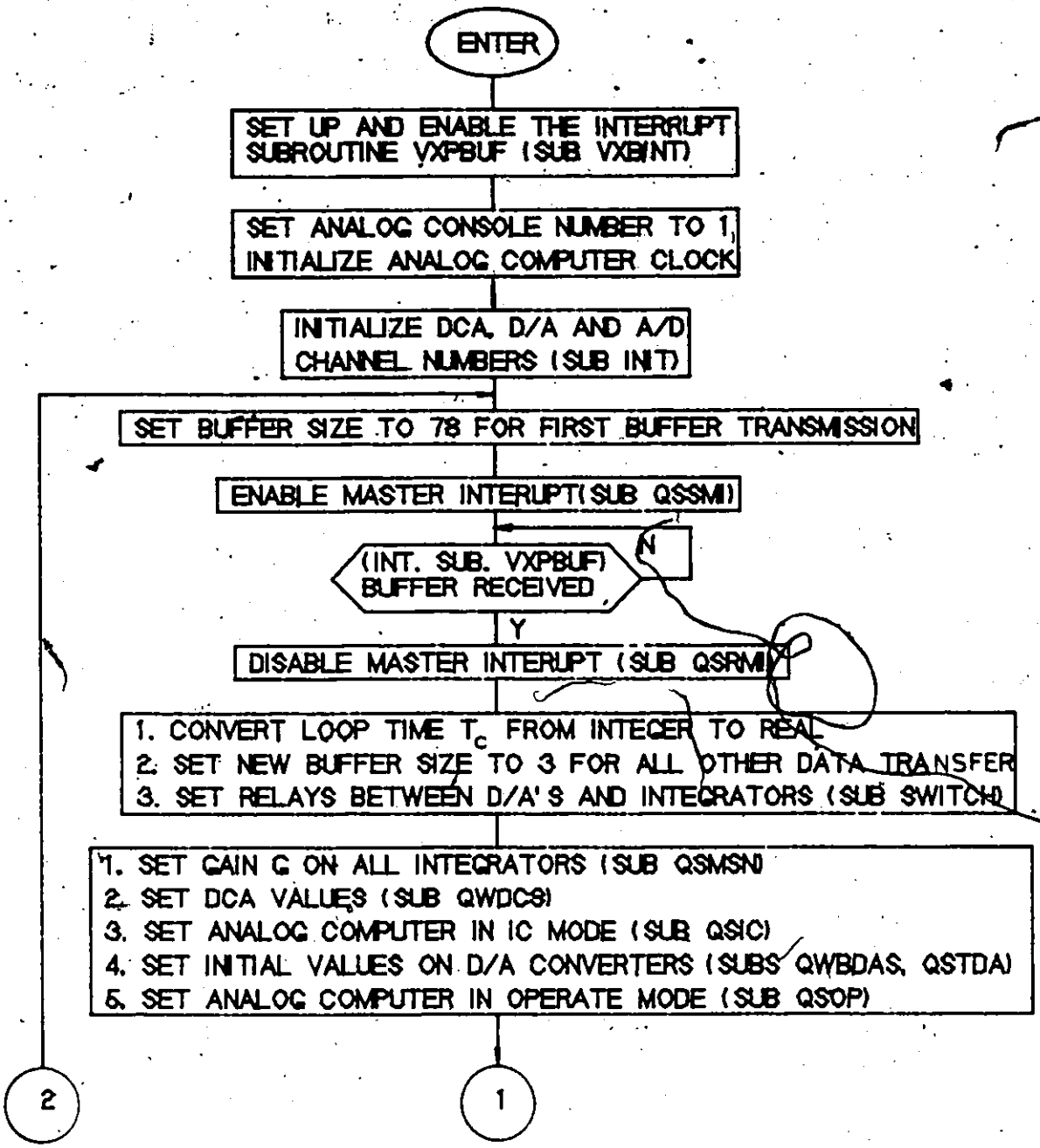


Figure G–2 : Program PACMAN flowchart

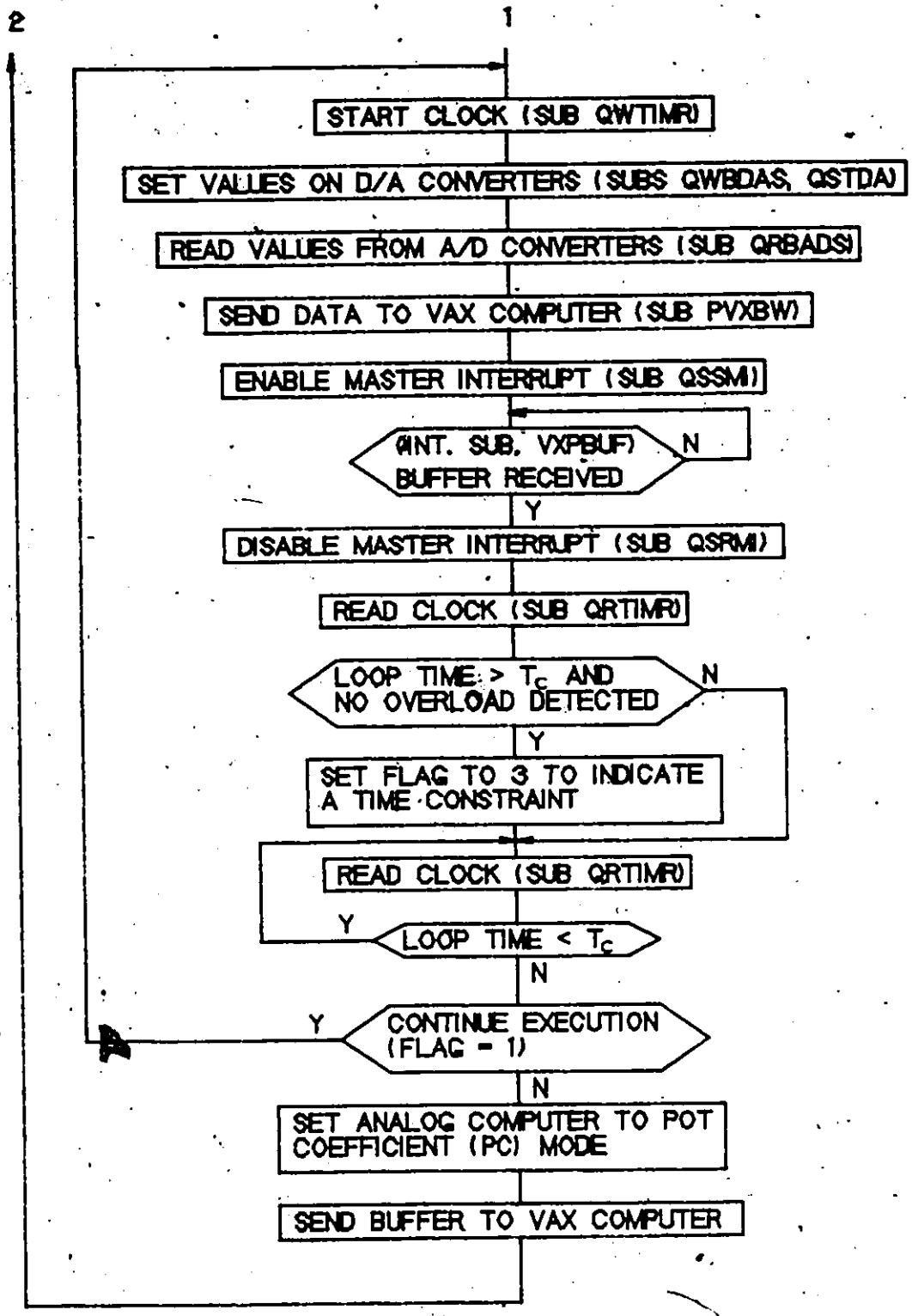


Figure G–2 : (End)

```
C_____________________________Program PACMAN
C_____This program sets up the analog computer to run the System C
C_____model, then within a loop, it receives data from the VAX
C_____program, sets D/A converters, reads A/D converters
C_____and sends data to the VAX. It also sets a constant high-
C_____speed-loop time (TC)

C_____Specification statements
      INTEGER FLAG, GAIN, RELAY(15), DUMMY(177)
      SCALED FRACTION BUFFER, DCA, DAC, ADC
      LOGICAL INTFLG

      DIMENSION BUFFER(79)
      DIMENSION IDCA(45), IDAC(15), IADC(15)
      DIMENSION  DCA(45),  DAC(15),  ADC(15)

      COMMON /VAXPAC/ INTFLG, NUMBUF, FLAG, DAC, DCA, ITC,
     -                NUMINT, GAIN, RELAY, DUMMY
      COMMON /CHANEL/ IDCA, IDAC, IADC

      EQUIVALENCE(BUFFER(1), FLAG)
      EQUIVALENCE(DAC(1), ADC(1))

C_____Set up and enable the interrupt subroutine VXPBUF as
C_____the only interrupt allowed on PACER
      CALL VXBINT

C_____Set an initial value for the analog computer clock
      TSET=50000

C_____Initialize the DCA channel numbers, the D/A channel
C_____numbers and the A/D channel numbers
      CALL INIT

C_____Start program loop. The program starts from here when
C_____subroutine SYSTMC is called the second time or more in
C_____the same VAX program
    5 CONTINUE

C_____Set BUFFER size to 79. The first buffer of data
C_____transmitted from VAX to PACER contains a maximum of
C_____79 numbers
      NUMBUF = 79

C_____Set INTFLG .FALSE. to indicate that data have not
C_____been transmitted from VAX to PACER. INTFLG is set .TRUE.
C_____in subroutine VXPBUF when data are received
      INTFLG = .FALSE.

C_____Enable master interupt and loop until the buffer is
C_____received, then disable master interrupt. This means
C_____that no interrupt can occur
      CALL GSSMI
   10 IF( NOT INTFLG) GO TO 10
      CALL GSRMI

C_____Convert computation-time (loop-time) from integer to real
      TC = FLOAT(ITC)/32768.0

C_____Adjust TC such that the error of setting and reading the
C_____clock is compensated
      TC = TC - 0.00055

C_____Set new buffer size for high-speed loop  Buffer size is
C_____the number of integrators used + 1
      NUMBUF = NUMINT + 1

C_____Set appropriate relays between D/A's and integrators
      CALL SWITCH(RELAY)

C_____Set amplifier gain on integrators
      CALL SETGAN(GAIN)

C_____Set DCA values on analog computer
      CALL GWDCS(IDCA, DCA, 45, IER)

C_____Set analog computer to initial condition mode
      CALL GSIC(IER)

C_____Set initial values on D/A converters
      CALL GWBDAS(DAC, IDAC, 15, IER)
      CALL GSTDA

C_____Set analog computer in operate mode
      CALL GSOP(IER)
```

```
C_______Start high-speed-loop. Stop when the VAX program sends
C_______a FLAG value other than 1 or when the loop-time is
C_______larger than the required value TC

C_______Set the timer on analog computer. The timer counts downward
      CALL GWTIMR(TSET, IER)

   20 CONTINUE

C_______Set values on D/A converters. The array DAC is a
C_______COMMON part of the array BUFFER
      CALL GWBDAS(DAC, IDAC, 15, IER)
      CALL GSTDA

C_______Read values from A/D converters. The array ADC is a
C_______COMMON part of the array BUFFER
      CALL GRBADS(ADC, IADC, NUMINT, IER)

C_______Send data to VAX through BUFFER
      CALL PVXBW(BUFFER,NUMBUF)

C_______Set INTFLG .FALSE. to indicate that data have not been
C_______received from VAX
      INTFLG = .FALSE.

C_______Enable master interupt and loop until the buffer is
C_______received, then disable master interrupt.
      CALL GSSMI
   30 IF( NOT INTFLG) GO TO 30
      CALL GSRMI

C_______Read timer and test if there was a time-constraint
C_______(i e  if loop-time is greater than TC required)  Test
C_______also if the VAX program has detected an overload on
C_______the analog computer  If there is a time-constraint
C_______without an overload then set FLAG to 3 to indicate
C_______this fact to the VAX program
      CALL GRTIMR(TREAD,IER)
      IF(((TSET-TREAD)/100000  GT TC) AND (FLAG NE 6)) FLAG=3

C_______Loop until loop-time has reached the constant value TC
   40 CONTINUE
        CALL GRTIMR(TREAD, IER)
      IF((TSET-TREAD)/100000  LT TC) GO TO 40

C_______Set the timer on analog computer. The timer counts downward
      CALL GWTIMR(TSET, IER)

C_______End of loop
      IF(FLAG EQ 1) GO TO 20

C_______Set analog computer to Pot Coefficient (PC) mode
      CALL GSPC(IER)

C_______Write a message on PACER printer if there is a time constraint
      IF(FLAG EQ 3) WRITE(6,100)

C_______Send buffer to VAX with a FLAG value other than 1
C_______indicating that program is stoping
      CALL PVXBW(BUFFER,NUMBUF)

C_______Go back to statement labeled 5 to be ready to start
C_______a new solution for SYSTEM C
      GO TO 5

C_______Format Statement
  100 FORMAT(30H *****  TIME CONSTRAINT  *****)

      END
```

```
C_____________________________Subroutine INIT
C_____Initializes the DCA channel numbers, the D/A channel
C_____numbers and the A/D channel numbers
      SUBROUTINE INIT

C_____Specification statements
      DIMENSION IDCA(45), IDAC(15), IADC(15)

      COMMON /CHANEL/ IDCA, IDAC, IADC

C_____Initialize DCA numbers
      DO 10 I=1,45,3
        IDCA(I) = ((I-1)/3)*5 + 30
        IDCA(I+1) = ((I-1)/3)*5 + 31
        IDCA(I+2) = ((I-1)/3)*5 + 32
   10 CONTINUE

C_____Initialize D/A Converter numbers
      DO 20 I=1,15
        IDAC(I) = I
   20 CONTINUE

C_____Initialize A/D converter numbers
      DO 30 I=1,15
        IADC(I) = I
   30 CONTINUE

      RETURN
      END



C_____________________________Subroutine SWITCH
C_____Sets appropriate relays between D/A converters and
C_____integrators
      SUBROUTINE SWITCH(RELAY)

C_____Specification statement
      INTEGER RELAY(15), IRELAY(15)

C_____Assign relay numbers
      DO 5 I=1,15
        IRELAY(I) = 34 + (I-1)*5
    5 CONTINUE

C_____Set relays
      DO 10 I=1,15
        IF (RELAY(I).EQ.0) CALL GWFRL(IRELAY(I),.FALSE.,IER)
        IF (RELAY(I).EQ.1) CALL GWFRL(IRELAY(I),.TRUE.,IER)
   10 CONTINUE

      RETURN
      END



C_____________________________Subroutine SETGAN
C_____Sets gain on all integrators. The value of gain
C_____depends on integer value GAIN sent from VAX
      SUBROUTINE SETGAN(GAIN)

C_____Specification statement
      INTEGER GAIN

C_____Test for gain value and set it on integrators
      IF(GAIN.EQ.1)    CALL GSSECN(IER)
      IF(GAIN.EQ.10)   CALL GSSECF(IER)
      IF(GAIN.EQ.100)  CALL GSMSN(IER)
      IF(GAIN.EQ.1000) CALL GSMSF(IER)

      RETURN
      END
```

APPENDIX H

Program RESPLT Documentation

Contents

- Subroutine Hierarchical Diagram
- Flowcharts
- Listings

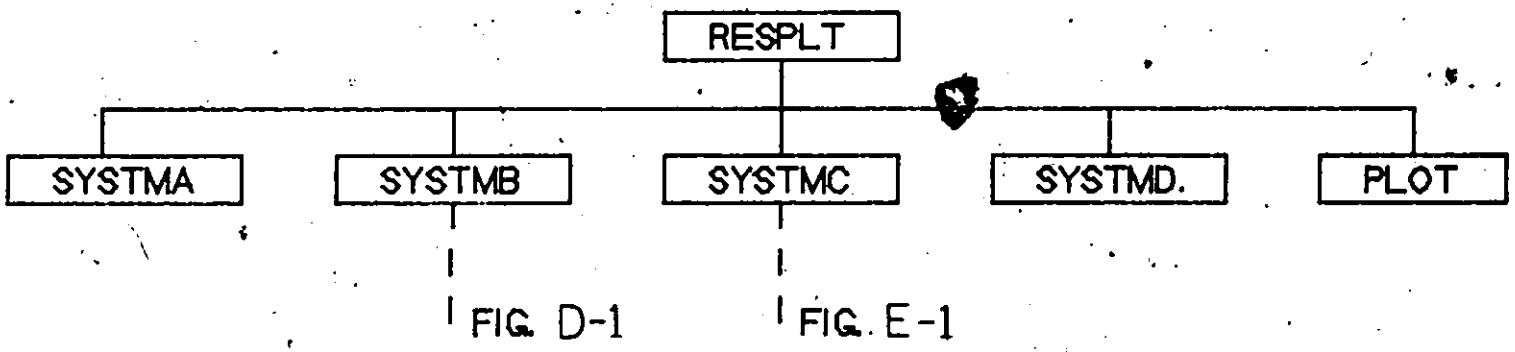


Figure H-1 : Subroutine hierarchical diagram of program RESPLT

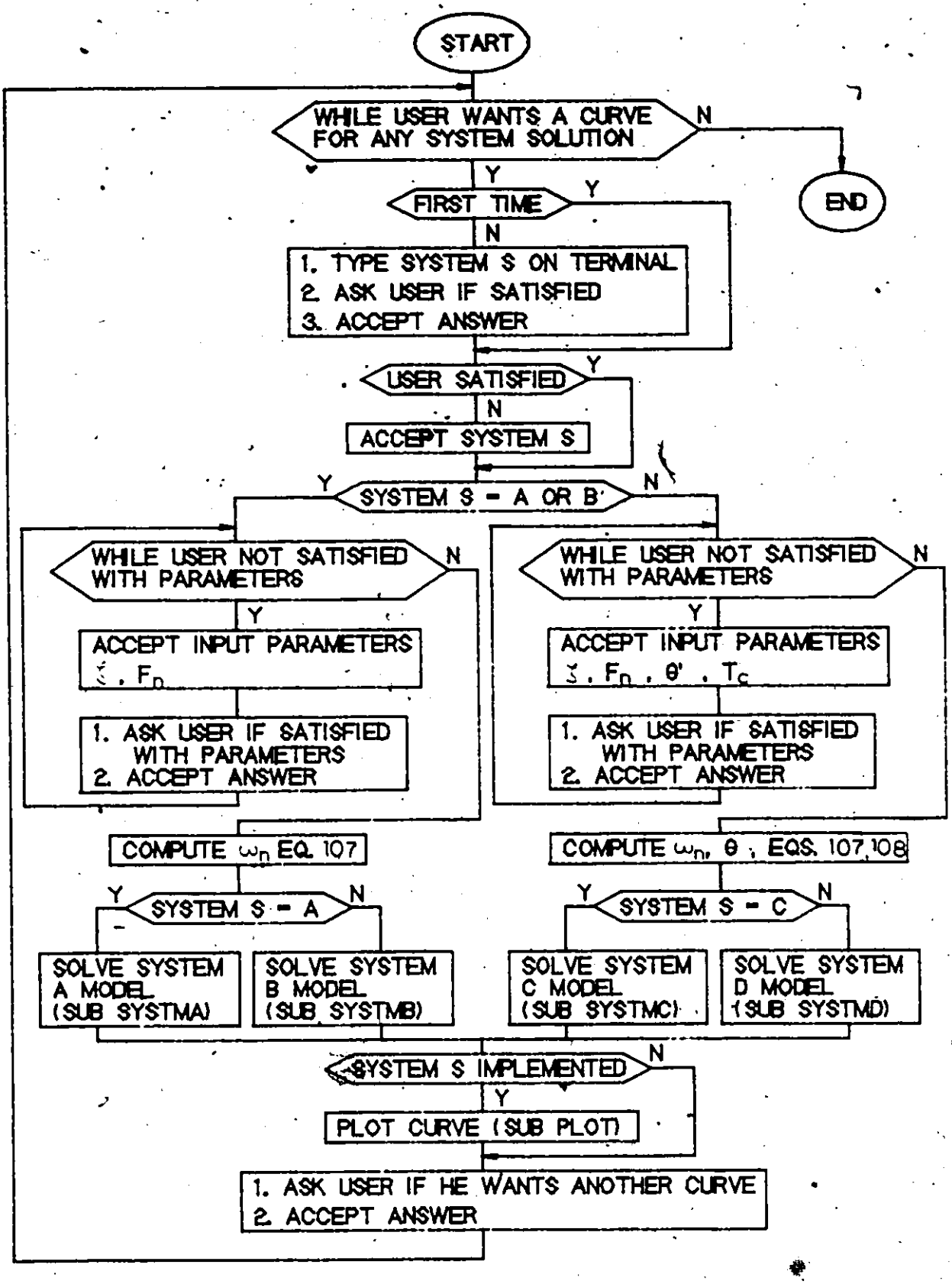


Figure H–2 : Program RESPLT flowchart

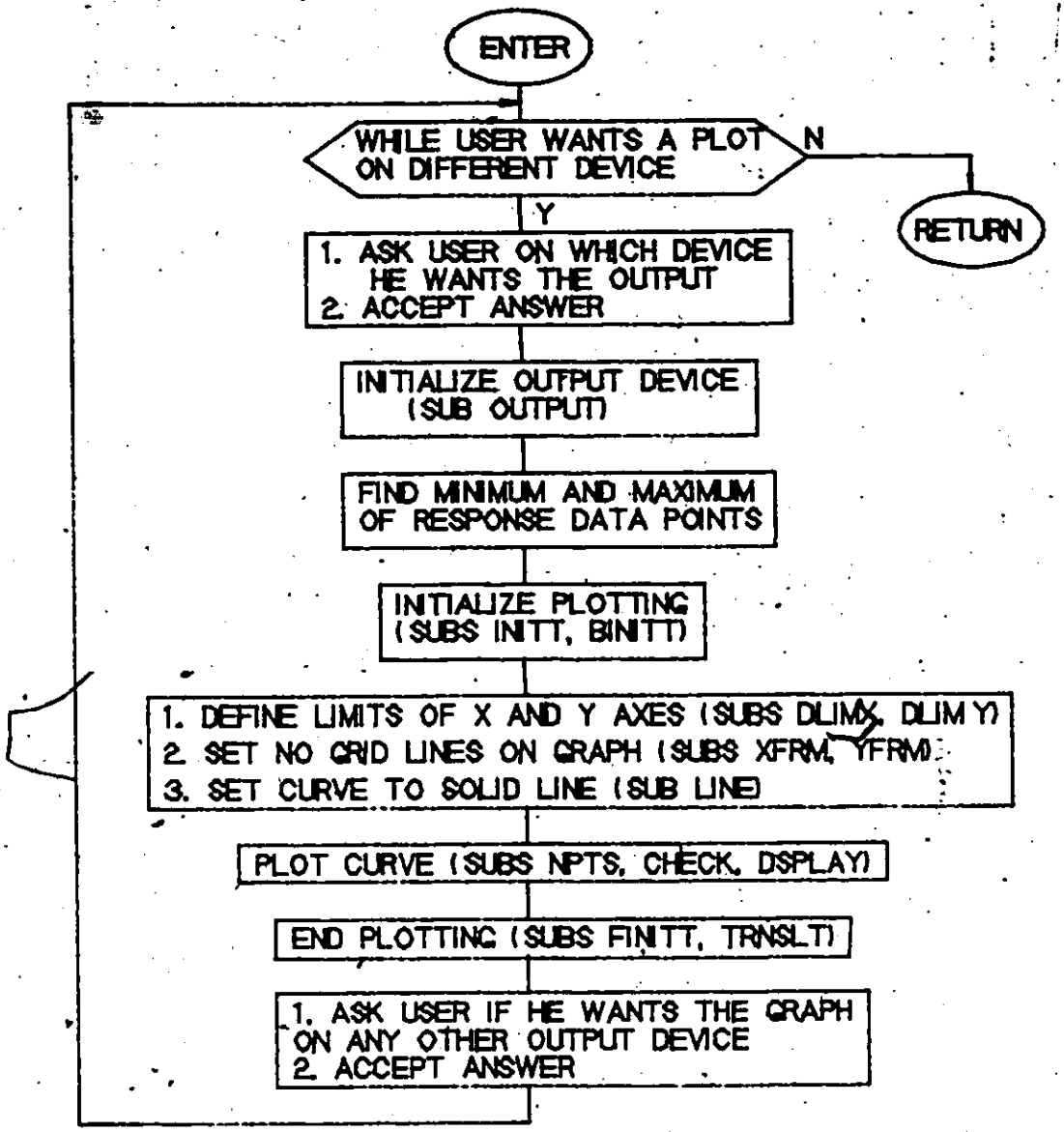


Figure H-3*: Subroutine PLOT flowchart

```
C_____Program RESPLT
C_____Plots response curve of any system.  System
C_____letters and system input parameters are specified
C_____interactively

C_____Specification statements
      CHARACTER ANSRMORE, INPUTOK, ANSRSYS, SYSTM

      LOGICAL FIRSTIME, CURVE, FIRSTRUNB, FIRSTRUNC

      DIMENSION RESPONSE(1000), RESTIME(1000)

      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /CONSTANTS/ XF, PI
      COMMON /FORCE/ STEP
      COMMON /NUMTIM/ NT, NS
      COMMON /FIRSTRUN/ FIRSTRUNB, FIRSTRUNC
      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME

C_____Open data set DSC.DAT
      OPEN(30, FILE='[GAC.ERR]DSC.DAT', STATUS='OLD')

C_____Initialize constants
      PI=3.1415927
      XF=1.0
      STEP=1.0

C_____Read number of cycles per solution and data storage
C_____sampling period
      READ(30,*) NT, NS

C_____Initialize flags to indicate the first run of
C_____System B or C
      FIRSTRUNB = .TRUE.
      FIRSTRUNC = .TRUE.

C_____Plot as many curves as requested by user
      CURVE=.TRUE.
      FIRSTIME=.TRUE.
      ANSRSYS='N'
      DO WHILE(CURVE)
        IMPLEMENTED=.FALSE.

C_____If not the first curve, ask user if satisfied with
C_____system entered.  If not, accept system from user
        IF( .NOT.FIRSTIME) THEN
          TYPE *
          TYPE *,'System = ',SYSTM
          TYPE *
          TYPE *,'Satisfied with system ? (Y/N) '
          READ(6,100) ANSRSYS
        END IF

C_____Accept system letter from user if he is not satisfied or if
C_____first time
        IF(ANSRSYS.EQ.'N') THEN
          TYPE *
          TYPE *,'Enter System   A, B, C or D'
          READ(6,200) SYSTM
        END IF

C_____Test which system, accept input parameters
C_____and compute response data for System A or B
        IF(SYSTM.EQ.'A' .OR. SYSTM.EQ.'B') THEN

C_____Accept input parameters
          IMPLEMENTED=.TRUE.
          INPUTOK='N'
          DO WHILE(INPUTOK.EQ.'N')
            TYPE *
            TYPE *,'Enter input parameters '
            TYPE *,'       Damping Ratio,       ZETA'
            TYPE *,'       Natural Frequency,   Fn  '
            READ(6,*) ZETA,FN
            TYPE *
            TYPE *,'Satisfied with parameters ? (Y/N) '
            TYPE 300,'   ZETA= ',ZETA
            TYPE 300,'   Fn   = ',FN
            READ(6,100) INPUTOK
          END DO
```

```
C_________Compute response data for System A or System B
          OMEGAN=2.*PI*FN
          IF(SYSTM.EQ.'A') THEN
            CALL SYSTMA
          ELSE
            CALL SYSTMB
            FIRSTRUNB = .FALSE.
            FIRSTRUNC = .TRUE.
          END IF

C_________Test which system, accept input parameters
C_________and compute response data for System C or D
          ELSE IF(SYSTM.EQ.'C'.OR.SYSTM.EQ.'D') THEN

C_________Accept input parameters
          IMPLEMENTED=.TRUE.
          INPUTOK='N'
          DO WHILE(INPUTOK.EQ.'N')
            TYPE *
            TYPE *,'Enter input parameters :'
            TYPE *,'      Damping Ratio.       ZETA'
            TYPE *,'      Natural Frequency.   Fn'
            TYPE *,'      Compensation Factor. THETA PRIME'
            TYPE *,'      Computation Time.    Tc'
            READ(6,*) ZETA,FN,THETAP,TC
            TYPE *
            TYPE *,'Satisfied with parameters ? (Y/N):'
            TYPE 400,'   ZETA=          ',ZETA
            TYPE 400,'   Fn   =          ',FN
            TYPE 400,'   THETA PRIME=  ',THETAP
            TYPE 400,'   Tc   =          ',TC
            READ(6,100) INPUTOK
          END DO

C_________Compute response data for System C or System D
          OMEGAN=2.*PI*FN
          THETA=1.5*TC*THETAP
          IF(SYSTM.EQ.'C') THEN
            CALL SYSTMC
            FIRSTRUNC = .FALSE.
            FIRSTRUNB = .TRUE.
          ELSE
            CALL SYSTMD
          END IF

C_________Inform user that the system he entered is not implemented
          ELSE
            TYPE *
            TYPE *,'****  System ''',SYSTM,'''is not implemented  ****'
          END IF

C_________Test if system has been implemented. If not,
C_________do not plot
          IF(IMPLEMENTED) THEN
            FIRSTIME=.FALSE.
            CALL PLOT
          ELSE
            FIRSTIME=.TRUE.
          END IF

C_________Ask user if he wants plot of another curve
          TYPE *
          TYPE *,'Output of another curve ? (Y/N):'
          READ(6,100) ANSRMORE
          IF(ANSRMORE.EQ.'N') THEN
            CURVE=.FALSE.
          END IF

        END DO

C_______Format statements
  100   FORMAT(A)
  200   FORMAT(A)
  300   FORMAT(A9,F7.3)
  400   FORMAT(A16,F8.4)

        END
```

```
C_____________________________Subroutine PLOT
C_____Plots response data points
      SUBROUTINE PLOT

C_____Specification statements
      CHARACTER ANSRDEVI, DEVICE
      LOGICAL MOREDEVIC

      DIMENSION RESPONSE(1000), RESTIME(1000)

      COMMON /RESPONSDATA/ NPOINT,RESPONSE,RESTIME

C_____Plot curve while user wants output on any device
      MOREDEVIC=.TRUE.
      DO WHILE(MOREDEVIC)

C_____Ask user for output device
        TYPE *
        TYPE *,'Enter output device:'
        TYPE *,'      Enter  S, for VGT Screen'
        TYPE *,'             P, for Line-Printer'
        TYPE *,'             H, for HP Plotter'
        READ(6,100) DEVICE

C_____Initialize output device
        IF(DEVICE.EQ.'P') THEN
          CALL OUTPUT('PRINTX','JKL',0)
        ELSE IF(DEVICE.EQ.'H') THEN
          CALL OUTPUT('HPPLOT','JKL',0)
        ELSE
          CALL OUTPUT('VGTERM','JKL',0)
        END IF

C_____Find minimum and maximum of response data points
        YMIN=1.0E30
        YMAX=-1.0E30
        DO I=1,NPOINT
          YMIN=MIN(YMIN,RESPONSE(I))
          YMAX=MAX(YMAX,RESPONSE(I))
        END DO

C_____Initialize plotting
        CALL INITT(30)
        CALL BINITT

C_____If output is going on printer,
C_____define location on paper and size of graph,
C_____set density of tic marks, and set software
C_____characters in order to adjust label sizes
        IF(DEVICE.EQ.'P') THEN
          CALL SLIMX(100,375)
          CALL SLIMY(50,190)
          CALL YDEN(8)
          CALL SWCHAR(1)
          CALL PLCHAR(30,50)
        END IF

C_____Define limits of X and Y axes
        CALL DLIMX(0.0,RESTIME(NPOINT))
        CALL DLIMY(YMIN,(1.05)*YMAX)

C_____Set no grid lines
        CALL XFRM(2)
        CALL YFRM(2)

C_____Set line to solid
        CALL LINE(0)

C_____Plot curve
        CALL NPTS(NPOINT)
        CALL CHECK(RESTIME,RESPONSE)
        CALL DSPLAY(RESTIME,RESPONSE)

C_____End plotting
        CALL FINITT(0,0)
        CALL TRNSLT(1)
```

```
C_________Ask user if he wants output on any other device
          TYPE *
          TYPE *,'Any other output device ? (Y/N):'
          READ(6,100) ANSRDEVI
          IF(ANSRDEVI.EQ.'N') THEN
            MOREDEVIC=.FALSE.
          END IF

        END DO

C_______Format statement
  100   FORMAT(A)

        RETURN
        END
```

APPENDIX I

Program RESPONS Documentation

Contents

- Subroutine Hierarchical Diagram
- Input-Process-Output Table of Subroutines
- Flowcharts
- Listings

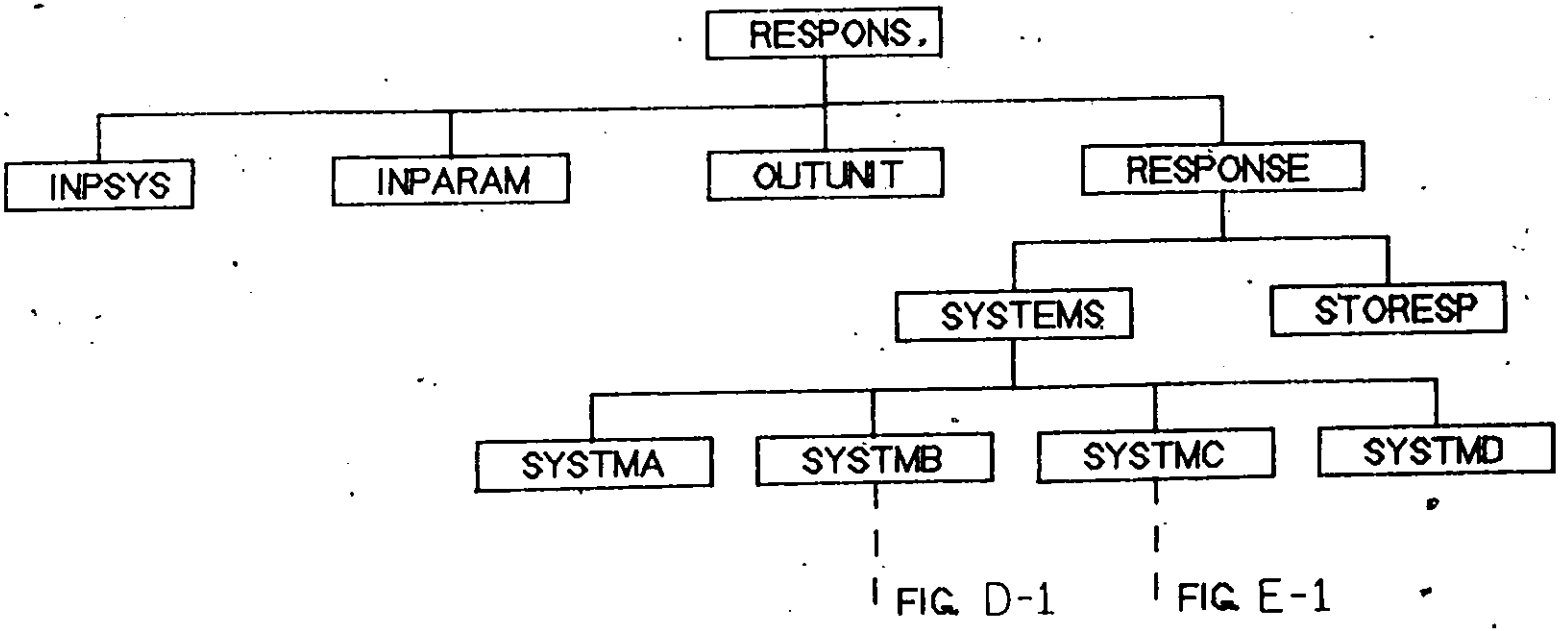


Figure I–1 : Subroutine hierarchical diagram of program RESPONS

Input-Process-Output Table of Subroutines

Subroutine	Inputs	Process	Outputs
INPSYS	S	Interactivity	S
INPARAM	DSA, DSB, DSC, θ'_i	Interactivity, Read files	DSA, DSB, DSC, θ'_i
RESPONSE	S, DSA, DSB, π, θ'_i	Fig. I-3	DSRA, DSRB, DSRC'i' DSRD'i'

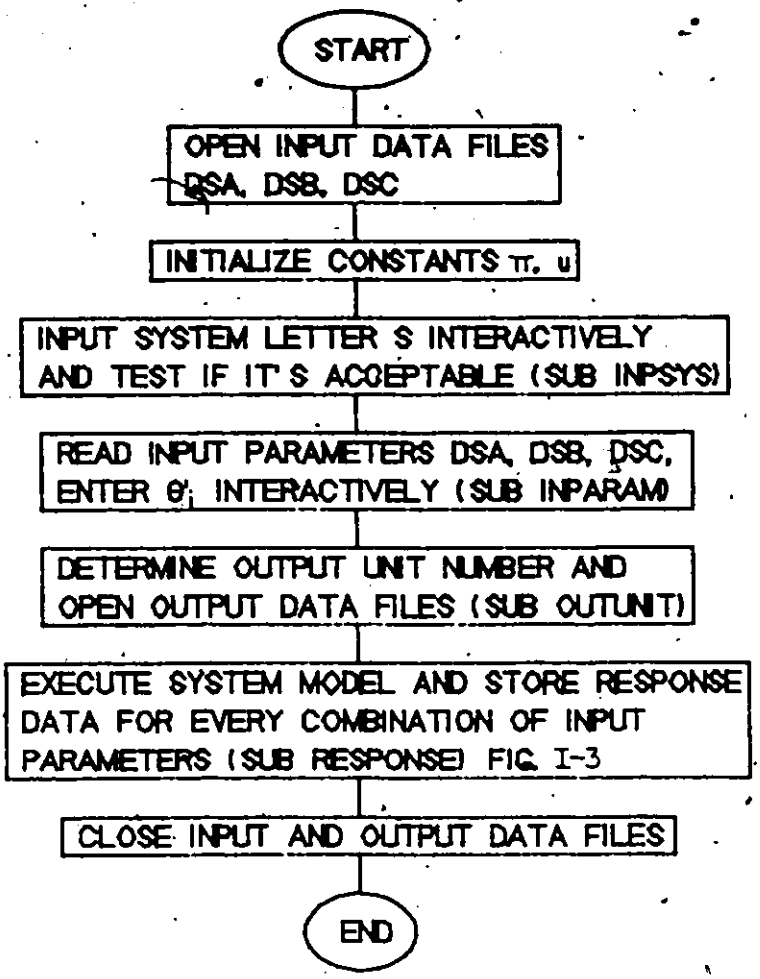


Figure I–2 : Program RESPONS flowchart

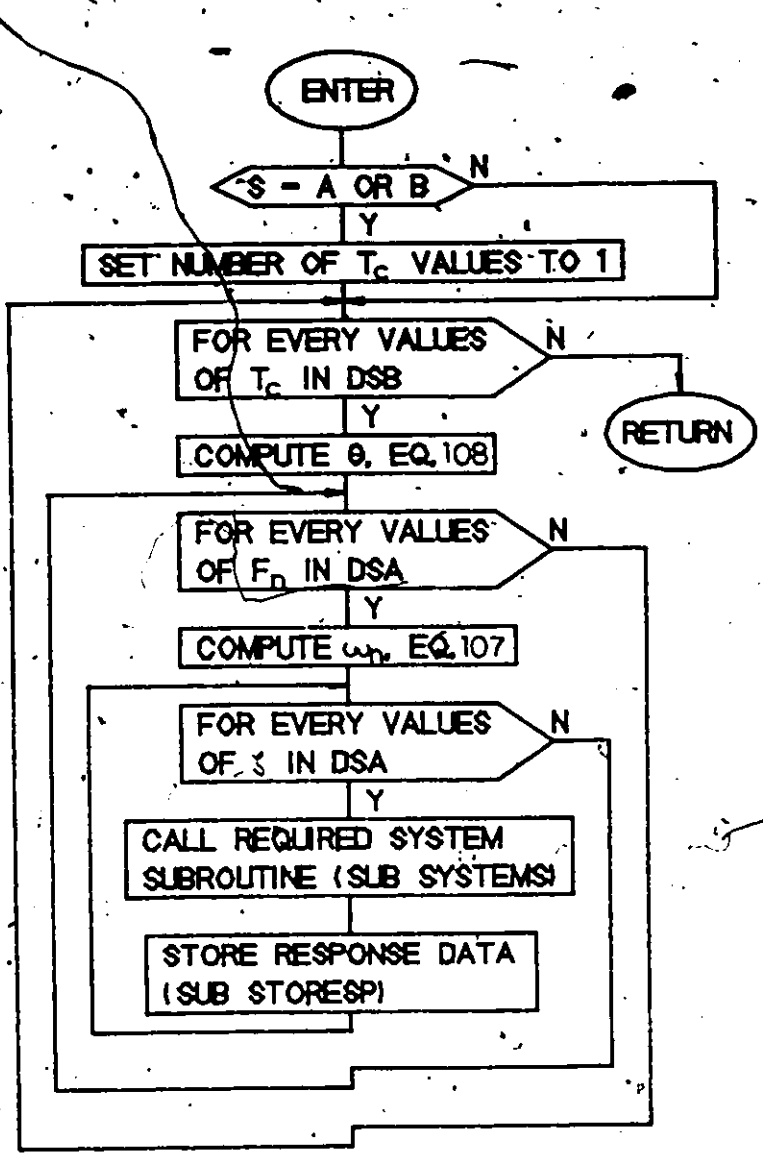


Figure I–3 : Subroutine RESPONSE flowchart

```
C_____________________________Program RESPONS
C_____This program executes a user specified system. It stores
C_____response data for this system for every set of input
C_____parameters contained in data sets DSA & DSB

C_____Specification statements
      LOGICAL FIRSTRUNB, FIRSTRUNC

      COMMON /FIRSTRUN/ FIRSTRUNB, FIRSTRUNC
      COMMON /CONSTANTS/ XF, PI
      COMMON /FORCE/ STEP
      COMMON /INPOINTER/ NUMTHETA

C_____Initialize contants
      PI = 3.1415927
      STEP = 1.0

C_____Initialize flags indicating the first run of Systems B and C
      FIRSTRUNB = .TRUE.
      FIRSTRUNC = .TRUE.

C_____Input System letter interactively
      CALL INPSYS

C_____Open input parameter data files
      OPEN(10, FILE='DSA.DAT', STATUS='OLD')
      OPEN(20, FILE='DSB.DAT', STATUS='OLD')
      OPEN(30, FILE='DSC.DAT', STATUS='OLD')

C_____Read input parameters
      CALL INPARAM

C_____Determine output unit number and open output data files
      CALL OUTUNIT

C_____Execute System and store response for every set of input
C_____parameters
      CALL RESPONSE

C_____Close data files
      CLOSE(01)
      CLOSE(02)
      CLOSE(03)
      CLOSE(04)
      CLOSE(10)
      CLOSE(20)
      CLOSE(30)

      END



C_____________________________Subroutine INPSYS
C_____Inputs System letter interactively
      SUBROUTINE INPSYS

C_____Specification statements
      CHARACTER SYSTM, ANSRSYS

      LOGICAL GOODSYS

      COMMON /SYS/ SYSTM

C_____Initialize system flag .FALSE. indicating that the
C_____inputed system letter is not acceptable
      GOODSYS = .FALSE.

C_____Accept System letter until it's a valid one and the user
C_____is satisfied
      DO WHILE( .NOT. GOODSYS)
        TYPE *
        TYPE *,'Enter System   A, B, C OR D'
        READ(6,100) SYSTM
        IF(SYSTM.EQ.'A' .OR. SYSTM.EQ.'B' .OR. SYSTM.EQ.'C' .OR.
     -     SYSTM.EQ.'D') THEN
          GOODSYS = .TRUE.
          TYPE *
          TYPE *,'System = ',SYSTM
          TYPE *
          TYPE *,'Satisfied with System ?'
          READ(6,100) ANSRSYS
          IF(ANSRSYS.EQ.'N') THEN
            GOODSYS = .FALSE.
          END IF
        ELSE
```

```
            TYPE *,
            TYPE *,'****  System ''',SYSTM,''' not implemented  ****'
          END IF
        END DO

C_______Format statement
  100   FORMAT(A)

        RETURN
        END


C_____________________________Subroutine INPARAM
C_______Read the system's input parameters from files DSA & DSB
        SUBROUTINE INPARAM

C_______Specification statements
        CHARACTER SYSTM, ANSRSYS
        DIMENSION COMPENSATION(40), COMPUT_TIME(40)
        DIMENSION FREQUENCY(40), DAMPING(40)

        COMMON /SYS/ SYSTM
        COMMON /NUMTIM/ NT, NS
        COMMON /INPARRAYS/ COMPENSATION, COMPUT_TIME,
                           FREQUENCY, DAMPING
        COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP
        COMMON /INPOINTER/ NUMTHETA

C_______Read number of cycles per solution and number of sampling
C_______data per cycle
        READ(30,*) NT, NS

C_______Read system parameters (files DSA & DSB)
        READ(10,*) NFREQ, (FREQUENCY(I), I=1,NFREQ)
        READ(10,*) NDAMP, (DAMPING(I), I=1,NDAMP)
        READ(20,*) NCOMP, (COMPENSATION(I), I=1,NCOMP)
        READ(20,*) NTIME, (COMPUT_TIME(I), I=1,NTIME)

C_______Enter the pointer for THETA_PRIME value
        IF((SYSTM EQ 'C') OR (SYSTM EQ 'D') OR
     -     (SYSREF EQ 'C') OR (SYSREF EQ 'D')) THEN
          TYPE *
          TYPE *,'Enter Theta Prime number where: '
          TYPE 100,'Theta Prime number:  ',(I, I = 1,NCOMP)
          TYPE 200,'Theta Prime value:   ',(COMPENSATION(I), I = 1,NCOMP)
          READ(6,*) NUMTHETA
        END IF

C_______Format Statements
  100   FORMAT(A21,10(I3,4X))
  200   FORMAT(A21,10(F5.3,2X))

        RETURN
        END


C_____________________________Subroutine OUTUNIT
C_______Determines output unit number and opens output files
        SUBROUTINE OUTUNIT

C_______Specification statements
        CHARACTER SYSTM

        INTEGER SYSUNIT

        COMMON /SYS/ SYSTM
        COMMON /INPOINTER/ NUMTHETA
        COMMON /UNIT/ SYSUNIT

C_______Set output unit number and open appropriate output file
        IF(SYSTM EQ 'A')THEN
          SYSUNIT = 1
          OPEN(01,FILE='DSRA.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(SYSTM EQ 'B')THEN
          SYSUNIT = 2
          OPEN(02,FILE='DSRB.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
```

```
      ELSE IF(SYSTM.EQ.'C')THEN
        SYSUNIT = 3
        IF(NUMTHETA.EQ.1) THEN
          OPEN(03,FILE='DSRC1.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.2) THEN
          OPEN(03,FILE='DSRC2.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.3) THEN
          OPEN(03,FILE='DSRC3.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.4) THEN
          OPEN(03,FILE='DSRC4.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.5) THEN
          OPEN(03,FILE='DSRC5.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        END IF
      ELSE IF(SYSTM.EQ.'D')THEN
        SYSUNIT = 4
        IF(NUMTHETA.EQ.1) THEN
          OPEN(04,FILE='DSRD1.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.2) THEN
          OPEN(04,FILE='DSRD2.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.3) THEN
          OPEN(04,FILE='DSRD3.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.4) THEN
          OPEN(04,FILE='DSRD4.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        ELSE IF(NUMTHETA.EQ.5) THEN
          OPEN(04,FILE='DSRD5.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
        END IF
      END IF

      RETURN
      END


C_____________________________Subroutine RESPONSE
C_____Executes required system for every combination of
C_____the appropriate input parameters
      SUBROUTINE RESPONSE

C_____Specification statements
      CHARACTER SYSTM

      DIMENSION COMPENSATION(40), COMPUT_TIME(40)
      DIMENSION FREQUENCY(40), DAMPING(40)

      COMMON /SYS/ SYSTM
      COMMON /INPARRAYS/ COMPENSATION, COMPUT_TIME,
     &                   FREQUENCY, DAMPING
      COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP
      COMMON /CONSTANTS/ XF, PI
      COMMON /INPUTPARAM/ THETA, TC, ZETA, OMEGAN
      COMMON /INPOINTER/ NUMTHETA

C_____Set number of Tc values to 1 if System A or B is involved
      IF(SYSTM.EQ.'A'.OR.SYSTM.EQ.'B') THEN
        NTIME = 1
        NUMTHETA = 1
        COMPUT_TIME(1) = 1.0
      END IF

C_____Solve appropriate system for every combination of
C_____input parameters
      THETA_PRIME = COMPENSATION(NUMTHETA)
      DO ITC = 1, NTIME
        TC = COMPUT_TIME(ITC)
        THETA = 1.5*TC*THETA_PRIME
        DO IFREQ = 1, NFREQ
          OMEGAN = 2.*PI*FREQUENCY(IFREQ)
          DO IZETA = 1, NDAMP
            ZETA = DAMPING(IZETA)

C_____________Call required system subroutine
              CALL SYSTEMS

C_____________Store response data
              CALL STORESP

          END DO
        END DO
      END DO

      RETURN
      END
```

```
C_____________________________Subroutine SYSTEMS
C_____Calls required system subroutine
      SUBROUTINE SYSTEMS

C_____Specification statements
      CHARACTER SYSTM

      LOGICAL FIRSTRUNB, FIRSTRUNC

      COMMON /FIRSTRUN/ FIRSTRUNB, FIRSTRUNC
      COMMON /SYS/ SYSTM

C_____Check required system, call its subroutine and set
C_____flags to indicate the first run of Systems B or C
      IF(SYSTM.EQ.'A')THEN
        CALL SYSTMA
      ELSE IF(SYSTM.EQ.'B')THEN
        CALL SYSTMB
        FIRSTRUNB = .FALSE.
        FIRSTRUNC = .TRUE.
      ELSE IF(SYSTM.EQ.'C')THEN
        CALL SYSTMC
        FIRSTRUNC = .FALSE.
        FIRSTRUNB = .TRUE.
      ELSE IF(SYSTM.EQ.'D')THEN
        CALL SYSTMD
      END IF

      RETURN
      END



C_____________________________Subroutine STORESP
C_____Stores response data
      SUBROUTINE STORESP

C_____Specification statements
      DIMENSION RESPONSE(1000), RESPONSD(1000), RESTIME(1000)

      COMMON /RESPONSDATA/ NPOINT, RESPONSE, RESTIME
      COMMON /RESPONSDOT/ RESPONSD

C_____Write response data in file numbered SYSUNIT
      WRITE(SYSUNIT) NPOINT, (RESPONSE(I), I=1,NPOINT),
     -                       (RESTIME(I),  I=1,NPOINT),
     -                       (RESPONSD(I), I=1,NPOINT)

      RETURN
      END
```

APPENDIX J

Program RMSERR Documentation

Contents

- Subroutine Hierarchical Diagram
- Input-Process-Output Table of Subroutines
- Flowcharts
- Listings

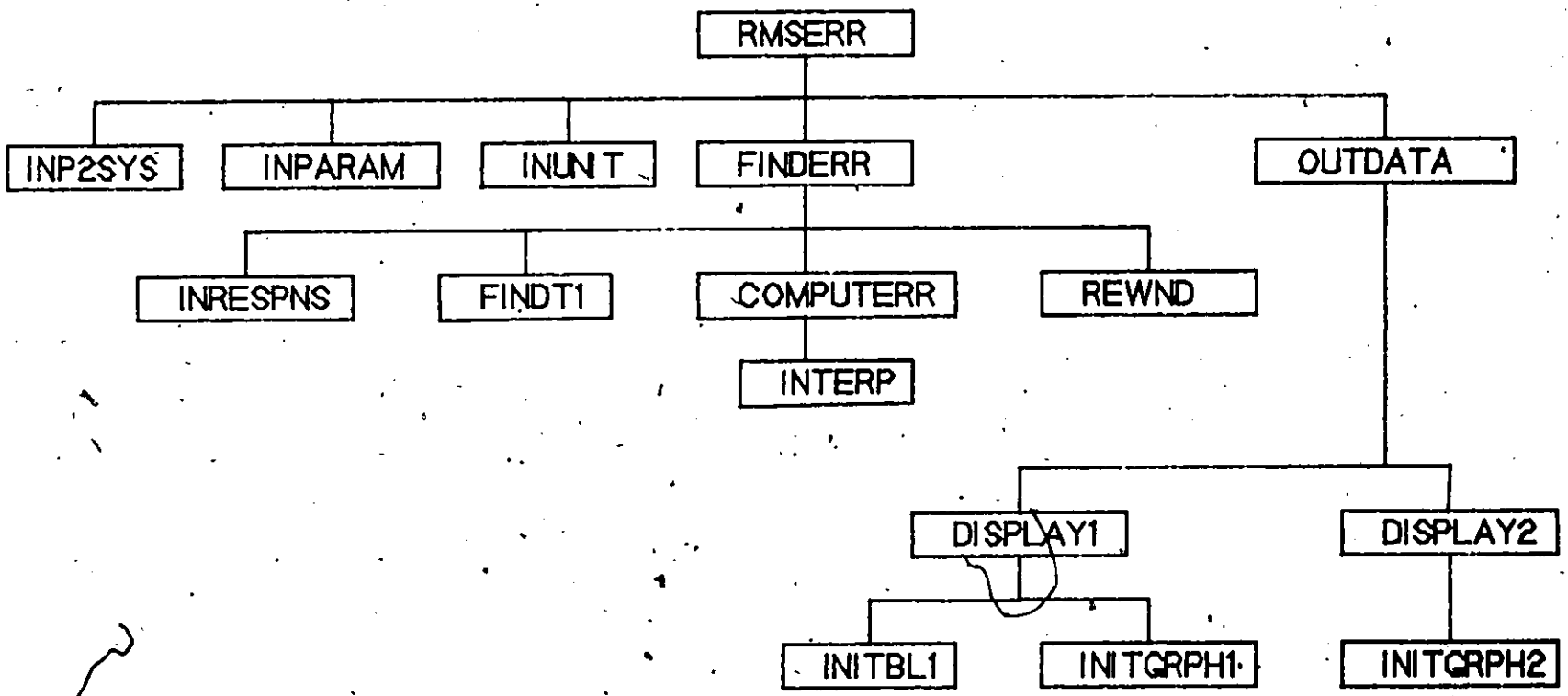


Figure J–1 : Subroutine hierarchical diagram of program RMSERR

Input-Process-Output Table of Subroutines

Subroutine	Inputs	Process	Outputs
INP2SYS	S_1, S_2	Interactivity	S_1, S_2
INPARAM	DSA, DSB, DSC, θ'_i	Interactivity, Read files	DSA, DSB, DSC, θ'_i
FINDERR	DSA, DSB, θ'_i, S_1, S_2	Fig. J-3	DSE
INRESP	DSR1, DSR2	Read files	DSR1, DSR2
FINDT1	DSR1, DSR2	Find minimum final time of both solutions	T_1
COMPUTERR	DSR1, DSR2, T_1	Eqs. 90, 94, 95, 97-99, 106	ε_N
DISPLAY1	DSA, DSB, DSD, DSE	Fig. J-5	DSEBA, DSECA, DSEDA, DSECD and graphs
DISPLAY2	DSA, DSB, DSE ε_{NA}	Fig. J-6	Graphs of $\varepsilon_N \leq \varepsilon_{NA}$

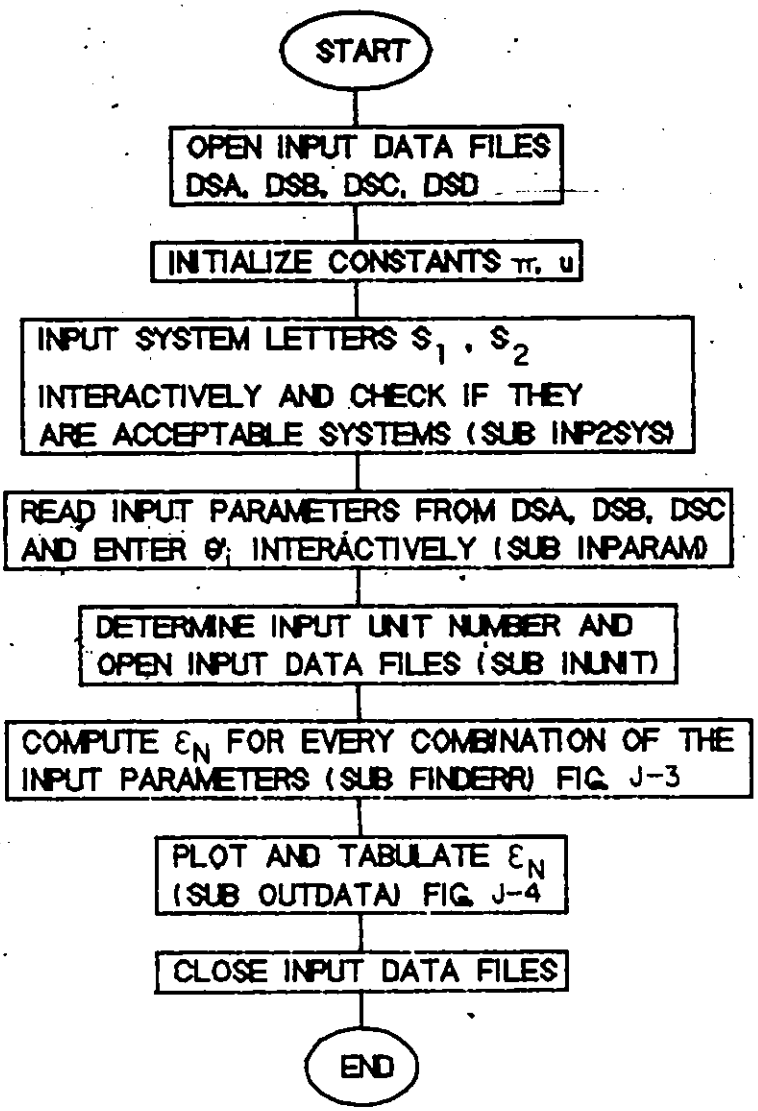


Figure J–2 : Program RMSERR flowchart

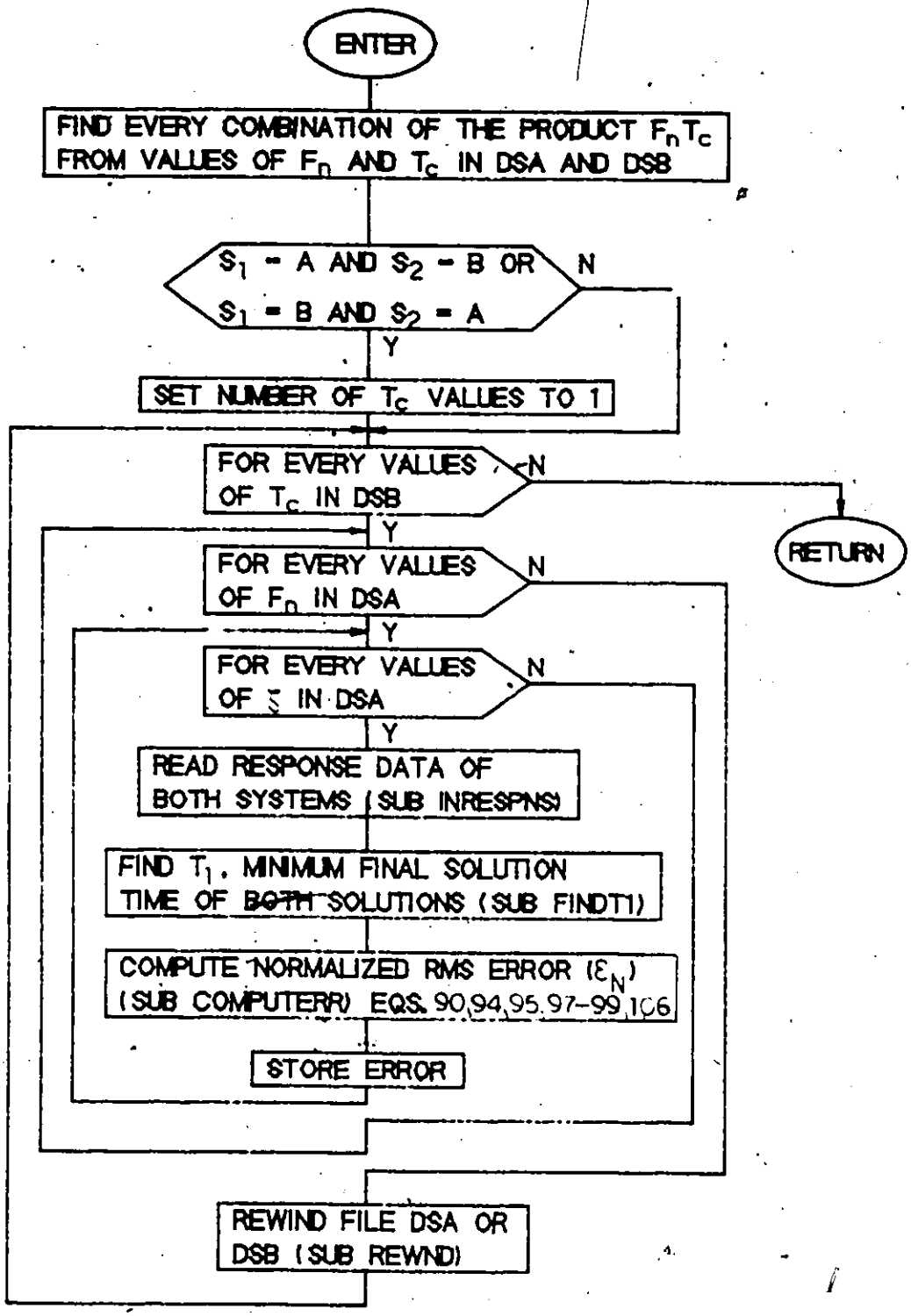


Figure J–3 : Subroutine FINDERR flowchart

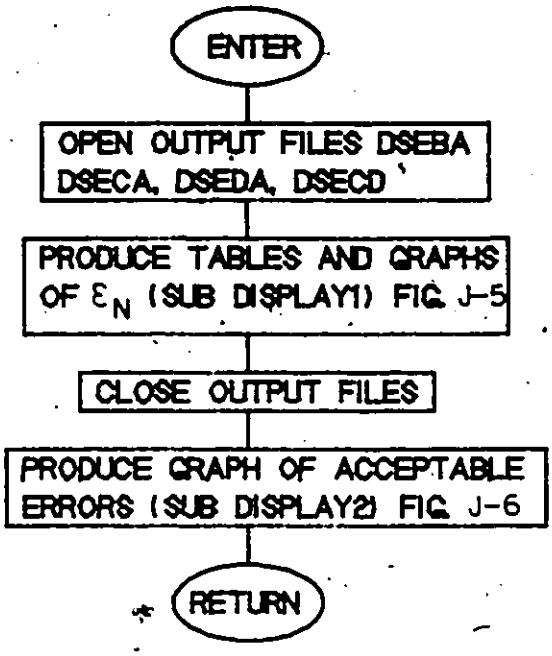


Figure J–4 : Subroutine OUTDATA flowchart

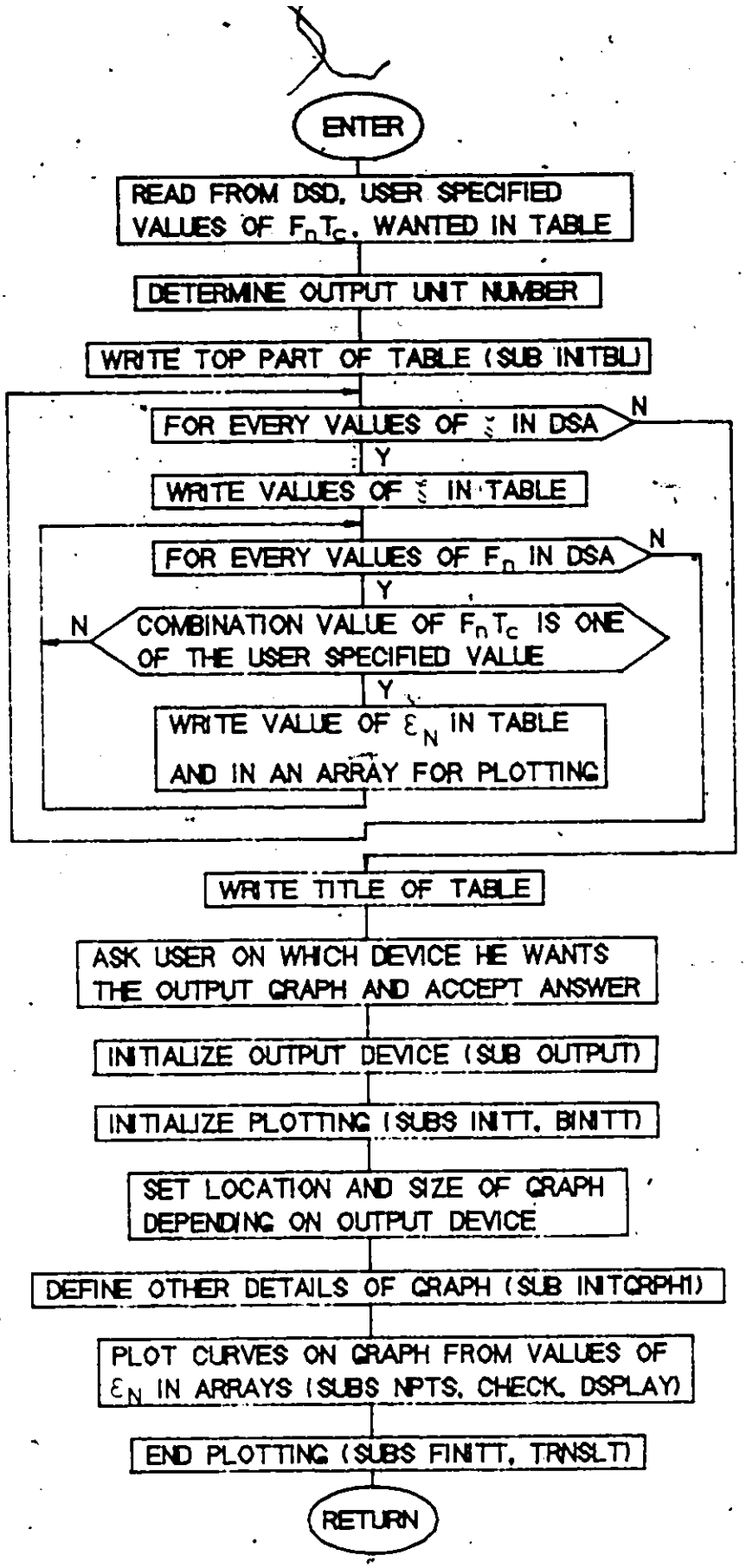


Figure J–5 : Subroutine DISPLAY1 flowchart

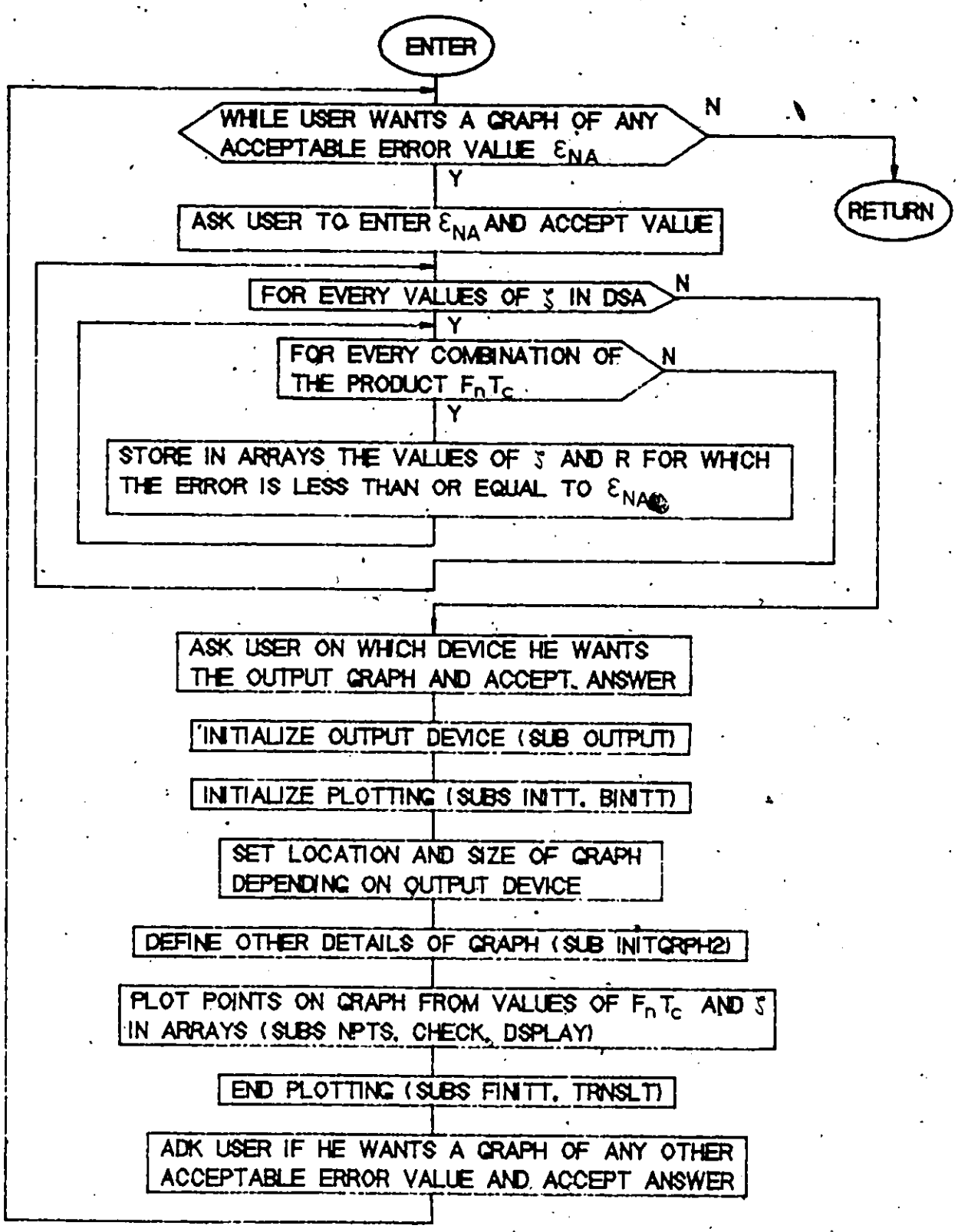


Figure J–6 : Subroutine DISPLAY2 flowchart

```
C_____Program RMSERR
C_____This program computes the RMS error between
C_____the solution of one system to another

C_____Specification statements
      COMMON /CONSTANTS/ XF, PI
      COMMON /FORCE/ STEP

C_____Open input parameter data files
      OPEN(10, FILE='DSA.DAT', STATUS='OLD')
      OPEN(20, FILE='DSB.DAT', STATUS='OLD')
      OPEN(30, FILE='DSC.DAT', STATUS='OLD')
      OPEN(40, FILE='DSD.DAT', STATUS='OLD')

C_____Initialize contants
      PI = 3.1415927
      STEP = 1.0

C_____Interactively input both systems to be compared
      CALL INP2SYS

C_____Read input parameters
      CALL INPARAM

C_____Determine the input device unit numbers
      CALL INUNIT

C_____Compute RMS error for every combination of the input
C_____parameters
      CALL FINDERR

C_____Plot and tabulate RMS errors
      CALL OUTDATA

C_____Close input data files
      CLOSE(01)
      CLOSE(02)
      CLOSE(03)
      CLOSE(04)
      CLOSE(10)
      CLOSE(20)
      CLOSE(30)
      CLOSE(40)

      END


C_____Subroutine INP2SYS
C_____Interactively inputs both systems to be compared
      SUBROUTINE INP2SYS

C_____Specification statements
      CHARACTER SYSTM, SYSREF, ANSRSYS

      LOGICAL GOODSYS

      COMMON /SYS/ SYSTM, SYSREF

C_____Initialize system flag, FALSE  indicating that the
C_____inputed system letters are not acceptable
      GOODSYS = .FALSE.

C_____Accept systems until they are valid and the user
C_____is satisfied
      DO WHILE( .NOT. GOODSYS)
        TYPE *
        TYPE *,'Compare System ''I'' to System ''J'''
        TYPE *,'where ''I'' and ''J'' = A, B, C or D'
        TYPE *
        TYPE *,'Enter ''I'',''J'''
        READ(6,100) SYSTM, SYSREF
        IF(SYSTM .EQ. 'B' .AND. SYSREF .EQ. 'A' .OR.
     -     SYSTM .EQ. 'C' .AND. SYSREF .EQ. 'A' .OR.
     -     SYSTM .EQ. 'D' .AND. SYSREF .EQ. 'A' .OR.
     -     SYSTM .EQ. 'C' .AND. SYSREF .EQ. 'D') THEN
          GOODSYS = .TRUE.
          TYPE *
          TYPE *,'Comparing System ''',SYSTM,''' to System ''',SYSREF,''''
          TYPE *
          TYPE *,'Satisfied with Systems ?'
          READ(6,200) ANSRSYS
          IF(ANSRSYS .EQ. 'N') THEN
            GOODSYS = .FALSE.
          END IF
        ELSE
```

```
            TYPE *
            TYPE *,'*****    System ''',SYSTM,'''              *****'
            TYPE *,'*****    cannot be compared     *****'
            TYPE *,'*****    to System ''',SYSREF,'''          *****'
          END IF
        END DO

C_____Format statement
  100   FORMAT(A,1X,A)
  200   FORMAT(A)

        RETURN
        END


C_____________________________Subroutine INPARAM
C_____Reads systems' input parameters from files DSA & DSB
C_____and enter pointer for THETA_PRIME value
        SUBROUTINE INPARAM

C_____Specification statements
        CHARACTER SYSTM, SYSREF
        DIMENSION COMPENSATION(40), COMPUT_TIME(40)
        DIMENSION FREQUENCY(40), DAMPING(40)

        COMMON /SYS/ SYSTM, SYSREF
        COMMON /NUMTIM/ NT, NS
        COMMON /INPARRAYS/ COMPENSATION, COMPUT_TIME,
     -                     ,FREQUENCY, DAMPING
        COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP
        COMMON /INPOINTER/ NUMTHETA, ITC

C_____Read number of cycles per solution and data storage
C_____sampling period
        READ(30,*) NT, NS

C_____Read system parameters (files DSA & DSB)
        READ(10,*) NFREQ, (FREQUENCY(I),I=1,NFREQ)
        READ(10,*) NDAMP, (DAMPING(I),I=1,NDAMP)
        READ(20,*) NCOMP, (COMPENSATION(I),I=1,NCOMP)
        READ(20,*) NTIME, (COMPUT_TIME(I),I=1,NTIME)

C_____Enter the pointer for THETA_PRIME value
        IF((SYSTM EQ 'C') OR (SYSTM EQ 'D') OR
     -     (SYSREF EQ 'C') OR (SYSREF EQ 'D')) THEN
          TYPE *
          TYPE *,'Enter Theta Prime number where '
          TYPE 100,'Theta Prime number  ',(I, I = 1,NCOMP)
          TYPE 200,'Theta Prime value   ',(COMPENSATION(I), I = 1,NCOMP)
          READ(6,*) NUMTHETA
        END IF

C_____Format statements
  100   FORMAT(A21,10(I3,4X))
  200   FORMAT(A21,10(F5.3,2X))

        RETURN
        END


C_____________________________Subroutine INUNIT
C_____Determines the input device unit numbers and opens
C_____input files
        SUBROUTINE INUNIT

C_____Specification statements
        CHARACTER SYSTM, SYSREF

        INTEGER SYSUNITX, SYSUNITY

        COMMON /SYS/ SYSTM, SYSREF
        COMMON /INPOINTER/ NUMTHETA, ITC
        COMMON /UNIT/ SYSUNITX, SYSUNITY

C_____Set required input unit device
        IF(SYSTM EQ 'A')THEN
          SYSUNITY = 1
        ELSE IF(SYSTM EQ 'B')THEN
          SYSUNITY = 2
        ELSE IF(SYSTM EQ 'C')THEN
          SYSUNITY = 3
        ELSE IF(SYSTM EQ 'D')THEN
          SYSUNITY = 4
        END IF
```

```
      IF(SYSREF EQ 'A')THEN
        SYSUNITX = 1
      ELSE IF(SYSREF EQ 'B')THEN
        SYSUNITX = 2
      ELSE IF(SYSREF EQ 'C')THEN
        SYSUNITX = 3
      ELSE IF(SYSREF EQ 'D')THEN
        SYSUNITX = 4
      END IF

C     Open appropriate input file
      IF(SYSTM EQ 'A' OR SYSREF EQ 'A') THEN
        OPEN(01,FILE='DSRA.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
      END IF
      IF(SYSTM EQ 'B' OR SYSREF EQ 'B') THEN
        OPEN(02,FILE='DSRB.DAT',STATUS='UNKNOWN',FORM='UNFORMATTED')
      END IF
      IF(SYSTM EQ 'C' OR SYSREF EQ 'C') THEN
        IF(NUMTHETA EQ 1) THEN
          OPEN(03,FILE='DSRC1.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 2) THEN
          OPEN(03,FILE='DSRC2.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 3) THEN
          OPEN(03,FILE='DSRC3.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 4) THEN
          OPEN(03,FILE='DSRC4.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 5) THEN
          OPEN(03,FILE='DSRC5.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        END IF
      END IF
      IF(SYSTM EQ 'D' OR SYSREF EQ 'D') THEN
        IF(NUMTHETA EQ 1) THEN
          OPEN(04,FILE='DSRD1.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 2) THEN
          OPEN(04,FILE='DSRD2.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 3) THEN
          OPEN(04,FILE='DSRD3.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 4) THEN
          OPEN(04,FILE='DSRD4.DAT',STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        ELSE IF(NUMTHETA EQ 5) THEN
          OPEN(04,FILE='DSRD5.DAT' STATUS='UNKNOWN',
     -         FORM='UNFORMATTED')
        END IF
      END IF

      RETURN
      END


C     ------------------------Subroutine FINDERR
C     Computes RMS error for every combinations
C     of the input parameters
      SUBROUTINE FINDERR

C     Specification statements
      CHARACTER SYSTM, SYSREF

      DIMENSION COMPENSATION(40), COMPUT_TIME(40)
      DIMENSION FREQUENCY(40), DAMPING(40)
      DIMENSION ERRARRAY(40,40)
      DIMENSION FNTC(40)

      COMMON /SYS/ SYSTM, SYSREF
      COMMON /INPARRAYS/ COMPENSATION, COMPUT_TIME,
     -                   FREQUENCY, DAMPING
      COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP
      COMMON /ERRPARAM/ K1
      COMMON /FINTIM/ T1
      COMMON /NUMTIM/ NT, NS
      COMMON /INPOINTER/ NUMTHETA, ITC
      COMMON /ERRDAT/ ERMSN
      COMMON /ERRARR/ ERRARRAY
      COMMON /FNTCARR/ FNTC, NFNTC
```

```
C_______Initialize constant K1, the number of squared difference
C_______data to be taken in COMPUTERR
        K1 = 300

C_______Give a value of 1 to the computation time indicating that
C_______the program will not deal with Tc if comparing solutions
C_______of Systems A and B
        IF(SYSTM.EQ.'A'.AND.SYSREF.EQ.'B'.OR.
     -     SYSTM.EQ.'B'.AND.SYSREF.EQ.'A') THEN
          NTIME = 1
          COMPUT_TIME(1) = 1.0
        END IF

C_______Find and store in array FNTC every possible combination of
C_______the product Fn with Tc  Define number NFNTC of such
C_______combinations
        NFNTC = 1
        FNTC(1) = 10000.
        DO ITC = 1, NTIME
          DO IFREQ = 1, NFREQ
            FREQTIME = FREQUENCY(IFREQ)*COMPUT_TIME(ITC)
            ICNT = 1
            DO WHILE(FREQTIME - FNTC(ICNT).GT..000001)
              ICNT = ICNT + 1
            END DO
            IF(ABS(FREQTIME - FNTC(ICNT)).GT. 000001) THEN
              DO I = NFNTC, ICNT, -1
                FNTC(I+1) = FNTC(I)
              END DO
              FNTC(ICNT) = FREQTIME
              NFNTC = NFNTC + 1
            END IF
          END DO
        END DO
        NFNTC = NFNTC - 1

C_______Inform user for which Theta Prime value the errors are being
C_______computed; only if Systems C or D are involved
        IF(NTIME.NE.1) THEN
          TYPE *
          TYPE 100,'Theta Prime = ',COMPENSATION(NUMTHETA)
        END IF

C_______Compute errors for every values of Tc
        DO ITC = 1, NTIME

C_________Inform user which systems are being compared and with which
C_________Tc value
          IF(NTIME.NE.1) THEN
            TYPE 200,'Comparing System ''',SYSTM,''' to System ''',
     -               SYSREF,''' for Tc = ',COMPUT_TIME(ITC)
          ELSE
            TYPE 300,'Comparing System ''',SYSTM,''' to System ''',
     -               SYSREF,''''
          END IF

C_________Compute errors for every values of Fn
          DO IFREQ = 1, NFREQ

C___________Find pointer IFNTC for which the combination of product
C___________in array FNTC matches the present product FREQTIME
            FN = FREQUENCY(IFREQ)
            FREQTIME = FN*COMPUT_TIME(ITC)
            IFNTC = 1
            DO WHILE(ABS(FREQTIME - FNTC(IFNTC)).GT..000001)
              IFNTC = IFNTC + 1
            END DO

C___________Compute errors for every values of ZETA
            DO IZETA = 1, NDAMP

C_____________Read response data of both systems to be compared
              CALL INRESPNS

C_____________Find the minimum value of both final solution times
              CALL FINDT1(FN)

C_____________Compute the normalized RMS error between both solutions
              CALL COMPUTERR

C_____________Store error
              ERRARRAY(IZETA,IFNTC) = ERMSN

            END DO
          END DO
```

```
C_________Rewind file DSRA or DSR8
          CALL REWND

        END DO

C_______Format statements
  100   FORMAT(' ',A14,F4.2)
  200   FORMAT(' ',A18,A,A13,A,A11,F5.3,/)
  300   FORMAT(' ',A18,A,A13,A,/)

        RETURN
        END


C_____________________________Subroutine INRESPNS
C_______Reads response data points of both systems being compared
        SUBROUTINE INRESPNS

C_______Specification statements
        INTEGER SYSUNITX, SYSUNITY

        DIMENSION RESPONSX(1000), RESTIMX(1000)
        DIMENSION RESPONSY(1000), RESTIMY(1000)
        DIMENSION RESPONSDX(1000), RESPONSDY(1000)

        COMMON /UNIT/ SYSUNITX, SYSUNITY
        COMMON /RESPONSDATX/ NPOINTX, RESTIMX, RESPONSX, RESPONSDX
        COMMON /RESPONSDATY/ NPOINTY, RESTIMY, RESPONSY, RESPONSDY

C_______Read one solution of both systems
        READ(SYSUNITX) NPOINTX, (RESPONSX(I), I=1,NPOINTX),
     -                          (RESTIMX(I),  I=1,NPOINTX),
     -                          (RESPONSDX(I),I=1,NPOINTX)

        READ(SYSUNITY) NPOINTY, (RESPONSY(I), I=1,NPOINTY),
     -                          (RESTIMY(I),  I=1,NPOINTY),
     -                          (RESPONSDY(I),I=1,NPOINTY)

        RETURN
        END


C_____________________________Subroutine FINDT1
C_______Finds a fixed solution-time to study the error between both
C_______The solution-time T1 is fixed to NT cycle periods of the
C_______analytical solution
        SUBROUTINE FINDT1(FN)

C_______Specification statements
        DIMENSION RESTIMX(1000), RESPONSX(1000)
        DIMENSION RESTIMY(1000), RESPONSY(1000)
        DIMENSION RESPONSDX(1000), RESPONSDY(1000)

        COMMON /RESPONSDATX/ NPOINTX, RESTIMX, RESPONSX, RESPONSDX
        COMMON /RESPONSDATY/ NPOINTY, RESTIMY, RESPONSY, RESPONSDY
        COMMON /NUMTIM/ NT, NS
        COMMON /FINTIM/ T1

C_______Find the minimum of the last response-time of both solutions
        TMIN = MIN(RESTIMX(NPOINTX), RESTIMY(NPOINTY))

C_______If the minimum final time (TMIN) of both solutions is within
C_______90% of NT cycle periods then set T1 to the min of TMIN and NT
C_______cycle periods  If not, set T1 to -1 indicating that one solution
C_______was not solved long enough to find the error (i e  error too large)
        IF (TMIN GE 0.90*NT/FN) THEN
            T1 = MIN(TMIN,NT/FN)
        ELSE
            T1=-1
        END IF

C_______Find the pointer for the largest time data value smaller
C_______than T1 for both solutions
        I = 1
        DO WHILE(RESTIMX(I) LT T1)
          I = I + 1
        END DO
        NPOINTX = I
```

```
      I = 1
      DO WHILE(RESTIMY(I).LT.T1)
        I = I + 1
      END DO
      NPOINTY = I

      RETURN
      END


C_____________________________Subroutine COMPUTERR
C_____Computes the normalized RMS error between the two
C_____time-responses X and Y
      SUBROUTINE COMPUTERR

C_____Specification statements
      DIMENSION RESTIMX(1000), RESPONSX(1000)
      DIMENSION RESTIMY(1000), RESPONSY(1000)
      DIMENSION RESPONSDX(1000), RESPONSDY(1000)

      COMMON /RESPONSDATX/ NPOINTX, RESTIMX, RESPONSX, RESPONSDX
      COMMON /RESPONSDATY/ NPOINTY, RESTIMY, RESPONSY, RESPONSDY
      COMMON /ERRPARAM/ K1
      COMMON /FINTIM/ T1
      COMMON /ERRDAT/ ERMSN


C_____Set error to a large value (1.0) if T1 equals -1 as set in
C_____subroutine FINDT1
      IF (T1 EQ -1) THEN
          ERMSN=1.0
      ELSE

C_____Initialization of variables
      DELTASQR = 0.0
      NDELTA = 0
      IPTRX = 1
      IPTRY = 1

C_____Initialize square difference computation interval
      DELTAT = T1/K1

C_____Find maximum values of responses X and Y
      RESPXM = -1.0E30
      RESPYM = -1.0E30
      DO I = 1, NPOINTX
        RESPXM = MAX(RESPXM, ABS(RESPONSX(I)))
      END DO

      DO I = 1, NPOINTY
        RESPYM = MAX(RESPYM, ABS(RESPONSY(I)))
      END DO

C_____Find maximum of both maximums
      RESPM = MAX(RESPXM, RESPYM)

C_____Find response X pointer where time of X is greater than
C_____or equal to first time-data value of response Y
      DO WHILE(RESTIMX(IPTRX) LT RESTIMY(IPTRY))
        IPTRX = IPTRX + 1
      END DO

C_____Initialize present time and present response values for
C_____response X
      TIMX = RESTIMX(IPTRX)
      RESPX = RESPONSX(IPTRX)
      RESPDX = RESPONSDX(IPTRX)

C_____Compute RMS error until the final solution time
      DO WHILE(TIMX LT T1)

C_________Find response Y pointer where time of Y is greater than
C_________present time of response X
        DO WHILE(RESTIMY(IPTRY) LE TIMX)
          IPTRY = IPTRY + 1
        END DO

C_________Interpolate to find value of response Y (RESPY) at
C_________present time of response X (TIMX)
        CALL INTERP(RESTIMY,RESPONSY,IPTRY,TIMX,RESPY)

C_________Compute difference between solutions only if the vertical
C_________difference is larger than .05% of the maximum amplitude
        IF(ABS(RESPY - RESPX) GT 0.0005*RESPM) THEN
```

```
C__________Count number of observed squared differences between
C__________both responses
           NDELTA = NDELTA + 1

C__________Add every squared differences (perpendicular to response X)
           DELTASQR = DELTASQR + (((RESPY - RESPX)/RESPM)
                                 *COS(ATAN(RESPDX*T1/RESPM)))**2

         END IF

C_________Increment present time of response X by a pre-defined
C_________constant interval
          TIMX = TIMX + DELTAT

C_________Find new response X pointer where time of X is greater
C_________than or equal to present time of X
          DO WHILE(RESTIMX(IPTRX) LT TIMX)
            IPTRX = IPTRX + 1
          END DO

C_________Interpolate to find value of response X (RESPX) at
C_________new present time of X (TIMX)
          CALL INTERP(RESTIMX,RESPONSX,IPTRX,TIMX,RESPX)

C_________Interpolate to find value of derivative of X (RESPDX) at
C_________new present time of X (TIMX)
          CALL INTERP(RESTIMX,RESPONSDX,IPTRX,TIMX,RESPDX)

        END DO

C_______Compute normalized RMS error dividing the sum of
C_______every squared differences by the number of observed
C_______squared differences and taking the square root
        IF(NDELTA EQ 0) THEN
          ERMSN = 0.0
        ELSE
          ERMSN = SQRT(DELTASQR/NDELTA)
        END IF

        END IF

        RETURN
        END


C_____________________________Subroutine INTERP
C_______Interpolate to find response value (RESPZ) at present
C_______time (TIMX) between response at IPTRZ and response at
C_______IPTRZ-1
        SUBROUTINE INTERP(RESTIMZ,RESPONSZ,IPTRZ,TIMX,RESPZ)

C_______Specification statements
        DIMENSION RESTIMZ(1000), RESPONSZ(1000)

C_______Find slope between response at IPTRZ and response at
C_______IPTRZ-1
        SLOP = (RESPONSZ(IPTRZ) - RESPONSZ(IPTRZ-1))/
               (RESTIMZ(IPTRZ) - RESTIMZ(IPTRZ-1))

C_______Find value of response at time TIMX
        RESPZ = SLOP*(TIMX - RESTIMZ(IPTRZ)) + RESPONSZ(IPTRZ)

        RETURN
        END


C_____ _____ _____ _____Subroutine REWND
C_______Rewinds files DSRA or DSRB
        SUBROUTINE REWND

C_______Specification statements
        INTEGER SYSUNITX, SYSUNITY

        COMMON /UNIT/ SYSUNITX, SYSUNITY

C_______Rewind appropriate file
        IF(SYSUNITX EQ 1 OR SYSUNITY EQ 1) THEN
          REWIND(01)
        END IF
        IF(SYSUNITY EQ 2 OR SYSUNITX EQ 2) THEN
          REWIND(02)
        END IF

        RETURN
        END
```

```
C_____________________________Subroutine OUTDATA
C_____Produce table and graph of RMS error and graph of
C_____acceptable error
      SUBROUTINE OUTDATA

C_____Open output data files
      OPEN(21,FILE='DSEBA.TBL',STATUS='UNKNOWN')
      OPEN(31,FILE='DSECA.TBL',STATUS='UNKNOWN')
      OPEN(41,FILE='DSEBA.TBL',STATUS='UNKNOWN')
      OPEN(34,FILE='DSECD.TBL',STATUS='UNKNOWN')

C_____Produce table and graph of RMS error
      CALL DISPLAY1

C_____Close output data files
      CLOSE(21)
      CLOSE(31)
      CLOSE(41)
      CLOSE(34)

C_____Produce graph of acceptable error
      CALL DISPLAY2

      RETURN
      END


C_____________________________Subroutine DISPLAY1
C_____Produce formatted tables of RMS error
      SUBROUTINE DISPLAY1

C_____Specification statements
      INTEGER SYSUNIT, SYSUNITX, SYSUNITY

      CHARACTER SYSTM, SYSREF, DEVICE, MOREDEVIC, LOC

      LOGICAL FOUND, FIRSTLINE

      DIMENSION COMPENSATION(40), COMPUT_TIME(40),
     -          FREQUENCY(40), DAMPING(40)
      DIMENSION ERRARRAY(40,40), FNTC(40), FNTCW(40)
      DIMENSION XARRAY(40), YARRAY(40)
      DIMENSION YTABLE(40,40), XTABLE(40,40)
      DIMENSION NPOINT(40)

      COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP
      COMMON /SYS/ SYSTM, SYSREF
      COMMON /UNIT/ SYSUNITX, SYSUNITY
      COMMON /INPARRAYS/ COMPENSATION, COMPUT_TIME,
     -                   FREQUENCY, DAMPING
      COMMON /INPOINTER/ NUMTHETA, ITC
      COMMON /ERRARR/ ERRARRAY
      COMMON /FNTCARR/ FNTC, NFNTC
      COMMON /FNTCWARR/ FNTCW, NFNTCW

C_____Read (from DSD) values of FnTc wanted in table; values
C_____of Fn in case of System B
      READ(40,*) NFNTCW, (FNTCW(K), K = 1, NFNTCW)

C_____Determine output file unit number
      SYSUNIT = 10*SYSUNITY + SYSUNITX

C_____Write on a new page
      WRITE(SYSUNIT,100)

C_____Write top part of the table
      CALL INITBL1(SYSUNIT)

C_____Initialize minimum and maximum of Y axis
      YMIN = 1.0E30
      YMAX = -1.0E30

C_____Write RMSERR values in table for every
C_____value of ZETA
      DO IZETA = 1, NDAMP

C_____Write ZETA value in table
        WRITE(SYSUNIT,200) DAMPING(IZETA)

C_____Write RMSERR values in table for every
C_____frequency values
        NFNTCF = 0
        DO IFNTC = NFNTC, 1, -1
          FOUND = .FALSE.
```

```
C_____Check if the combination product of Fn and Tc is one
C_____of the values requested by user in DSD.  If so, write
C     error value in table
      DO IFNTCW = NFNTCW, 1, -1
        IF(ABS(FNTC(IFNTC) - FNTCW(IFNTCW)).LT..000001) THEN
          NFNTCF = NFNTCF + 1
          FOUND =  TRUE
        END IF
      END DO
      IF(FOUND) THEN

C_____Choose different format depending on output
C_____value
        IF(ERRARRAY(IZETA, IFNTC).LT.0.001) THEN
          WRITE(SYSUNIT,300) ERRARRAY(IZETA, IFNTC)
        ELSE IF(ERRARRAY(IZETA, IFNTC).LT.1.) THEN
          WRITE(SYSUNIT,400) ERRARRAY(IZETA, IFNTC)
        ELSE
          WRITE(SYSUNIT,500) ERRARRAY(IZETA, IFNTC)
        END IF

C_____Store error value and damping in table arrays for
C_____plotting.  Find minimum and maximum of Y axis
        IF(ERRARRAY(IZETA, IFNTC).LE.1.0) THEN
          NPOINT(NFNTCF) = NPOINT(NFNTCF) + 1
          YTABLE(NFNTCF, NPOINT(NFNTCF)) = ERRARRAY(IZETA, IFNTC)
          XTABLE(NFNTCF, NPOINT(NFNTCF)) = DAMPING(IZETA)
          YMIN = MIN(YMIN, ERRARRAY(IZETA, IFNTC))
          YMAX = MAX(YMAX, ERRARRAY(IZETA, IFNTC))
        END IF
      END IF

    END DO

C_____Write a full horizontal line
    WRITE(SYSUNIT,600)
    WRITE(SYSUNIT,700)

  END DO

C_____Write values of THETA PRIME at bottom of table
  IF(NTIME EQ 1) THEN
    WRITE(SYSUNIT,800) SYSTM, SYSREF
  ELSE
    WRITE(SYSUNIT,900) SYSTM, SYSREF, NUMTHETA
    WRITE(SYSUNIT,1000) COMPENSATION(NUMTHETA)
  END IF

C_____Plot curves while user wants output on any device
  MOREDEVIC = 'Y'
  DO WHILE(MOREDEVIC EQ 'Y')

C_____Ask user to specify output device
    TYPE *
    TYPE *,'Enter output device where you'
    TYPE *,'want graph of RMS error '
    TYPE *,'    Enter S, for VQT Screen'
    TYPE *,'          P, for Line-Printer'
    TYPE *,'          H, for HP Plotter'
    READ(6,1100) DEVICE

C_____Initialize output device
    IF(DEVICE EQ 'P') THEN
      CALL OUTPUT('PRINTX','JKL',0)
    ELSE IF(DEVICE EQ 'H') THEN
      CALL OUTPUT('HPPLOT','JKL',0)
    ELSE
      CALL OUTPUT('VGTERM','JKL',0)
    END IF

C_____Initialize minimum and maximum of X axis
    XMIN=DAMPING(1)
    XMAX=DAMPING(NDAMP)

C_____Initialize plotting
    CALL INITT(30)
    CALL BINITT

C_____Set location and size of graph depending on output device
    IF(DEVICE EQ 'P') THEN

C_____Initialize location and size of graph
      CALL SWCHAR(1)
      MINHOR = 200
      MAXHOR = MINHOR + 195
      MINVER = 370
      MAXVER = MINVER + 115
```

```
C_________Set page location
          CALL SLIMX(MINHOR,MAXHOR)
          CALL SLIMY(MINVER,MAXVER)

        ELSE IF(DEVICE.EQ.'H') THEN

C_________Initialize location and size of graph
          CALL SWCHAR(1)
          MINHOR = 260
          MAXHOR = 510
          TYPE *
          TYPE *,'Top or Bottom ? (T/B)'
          READ(6,1100) LOC
          IF(LOC.EQ.'T') THEN
            MINVER = 450
            MAXVER = 590
          ELSE
            MINVER = 120
            MAXVER = 260
          END IF

C_________Set page location
          CALL SLIMX(MINHOR,MAXHOR)
          CALL SLIMY(MINVER,MAXVER)

        END IF

C_________Set other details of graph
          CALL INITGRPH1

C_________Plot some lines
          FIRSTLINE = .TRUE.
          DO IFNTCF = NFNTCF -.5, NFNTCF - 1

C_________Write error and damping values into arrays
            DO K = 1, NPOINT(IFNTCF)
              YARRAY(K) = YTABLE(IFNTCF,K)
              XARRAY(K) = XTABLE(IFNTCF,K)
            END DO

C_________Plot one line
            IF (NPOINT(IFNTCF).NE.0)THEN
              CALL NPTS(NPOINT(IFNTCF))
              IF(FIRSTLINE) THEN
                CALL CHECK(XARRAY,YARRAY)
                CALL DSPLAY(XARRAY,YARRAY)
                FIRSTLINE = .FALSE.
              ELSE
                CALL CPLOT(XARRAY,YARRAY)
              END IF
            END IF

          END DO

C_________End plotting
          CALL FINITT(0,0)
          CALL TRNSLT(1)

C_________Ask user if he wants output on any other device
          TYPE *
          TYPE *,'Any other output device ? (Y/N): '
          READ(6,1100) MOREDEVIC

        END DO

C_________Format statements
  100   FORMAT('1')
  200   FORMAT(1X,10X,'|',F6.3,'|',$)
  300   FORMAT('+',' ',E5.1E1,'|',$)
  400   FORMAT('+',' ',F4.3,' |',$)
  500   FORMAT('+',' ',F4.2,' |',$)
  600   FORMAT('+')
  700   FORMAT('+',10X,' _____',7('_______'))
  800   FORMAT(//'+',10X,'TABLE ',A1,A1,//)
  900   FORMAT(//'+',10X,'TABLE ',A1,A1,1X,I1,//)
 1000   FORMAT(1X,10X,'O'' = ',F3.1)
 1100   FORMAT(A)

        RETURN
        END
```

```
C_____Subroutine INITBL1
C_____Initializes table
      SUBROUTINE INITBL1(SYSUNIT)

C_____Specification statements
      INTEGER SYSUNIT

      DIMENSION FNTC(40), FNTCW(40)

      COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP
      COMMON /FNTCARR/ FNTC, NFNTC
      COMMON /FNTCWARR/ FNTCW, NFNTCW

C_____Write table variables
      WRITE(SYSUNIT,100)
      IF(NTIME.EQ.1) THEN
        WRITE(SYSUNIT,200)
      ELSE
        WRITE(SYSUNIT,300)
      END IF

C_____Check if values of FnTc requested by user are values
C_____for which the error was found and write these values
C_____in table
      DO I = NFNTCW, 1, -1
        DO J = NFNTC, 1, -1
          IF(ABS(FNTCW(I) - FNTC(J)) LT  000001) THEN
            IF(NTIME EQ 1) THEN
              WRITE(SYSUNIT,400) 1 /FNTCW(I)
            ELSE
              WRITE(SYSUNIT,500) 1 /FNTCW(I)
            END IF
          END IF
        END DO
      END DO
      WRITE(SYSUNIT,600)

      WRITE(SYSUNIT,700)
      WRITE(SYSUNIT,100)

C_____Format statements
  100 FORMAT('+',10X,'  _____',7('  _____'))
  200 FORMAT(1X,10X,':    \Tn:',$)
  300 FORMAT(1X,10X,':    \R  :',$)
  400 FORMAT('+',F6.3,':',$)
  500 FORMAT('+',F6.0,':',$)
  600 FORMAT('+')
  700 FORMAT(1X,10X,':ZETA\  :',7(6X,':'))

      RETURN
      END


C_____Subroutine INITGRPH1
C_____Sets details of graph
      SUBROUTINE INITGRPH1

C_____Set limits of X and Y axes
      CALL DLIMX( 0001.10 )
      CALL DLIMY( 0001.  1)

C_____Set type of frame, no grid line
      CALL XFRM(2)
      CALL YFRM(2)

C_____Set tic marks length
      CALL XLEN(4)
      CALL YLEN(4)

C_____Set density of tic marks
      CALL XDEN(5)
      CALL YDEN(5)

C_____Set maximum number of character in tic mark labels
C_____of Y axis
      CALL YWDTH(5)

C_____Set logarithmic X and Y axes
      CALL XTYPE(2)
      CALL YTYPE(2)

C_____Set character size for tic mark labels
      CALL PLCHAR(25,40)

      RETURN
      END
```

```
C_____________________________Subroutine DISPLAY2
C_____Produces formatted tables of acceptable RMS error
      SUBROUTINE DISPLAY2

C_____Specification statements
      CHARACTER DEVICE, MOREDEVIC, MOREACCEPT

      LOGICAL FOUND

      DIMENSION COMPENSATION(40), COMPUT_TIME(40),
     -          FREQUENCY(40), DAMPING(40)
      DIMENSION YARRAY(2000), XARRAY(2000)
      DIMENSION FNTC(40)
      DIMENSION ERRARRAY(40,40)

      COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP
      COMMON /INPARRAYS/ COMPENSATION, COMPUT_TIME,
     -                   FREQUENCY, DAMPING
      COMMON /INPOINTER/ NUMTHETA, ITC
      COMMON /ERRARR/ ERRARRAY
      COMMON /FNTCARR/ FNTC, NFNTC

      MOREACCEPT = 'Y'
      DO WHILE(MOREACCEPT EQ 'Y')

C_____Ask user for acceptable RMSERR value
        TYPE *
        TYPE *,'Enter acceptable RMSERR value'
        READ(5,*) ACCEPT

C_____Initialize minimum and maximum of Y axis
        YMIN = 1.0E30
        YMAX = -1.0E30

C_____Find FNTC values where RMSERR is acceptable and store
C_____values of 1/FnTc and damping in arrays for plotting
        NPNT = 0
        DO IZETA = 1, NDAMP
          DO IFNTC = NFNTC, 1, -1
            YMIN = MIN(YMIN, 1./FNTC(IFNTC))
            YMAX = MAX(YMAX, 1./FNTC(IFNTC))
            IF(ERRARRAY(IZETA,IFNTC) LE ACCEPT) THEN
              NPNT = NPNT + 1
              XARRAY(NPNT) = DAMPING(IZETA)
              YARRAY(NPNT) = 1./FNTC(IFNTC)
            END IF
          END DO
        END DO

C_____Plot the curve while user wants output on any device
        MOREDEVIC = 'Y'
        DO WHILE(MOREDEVIC EQ 'Y')

C_____Ask user to specify output device
          TYPE *
          TYPE *,'Enter output device where you'
          TYPE *,'want graph of acceptable error'
          TYPE *,'       Enter S, for VGT Screen'
          TYPE *,'             P, for Line-Printer'
          TYPE *,'             H, for HP Plotter'
          READ(5,100) DEVICE

C_____Initialize output device
          IF(DEVICE EQ 'P') THEN
            CALL OUTPUT('PRINTX','JKL',0)
          ELSE IF(DEVICE EQ 'H') THEN
            CALL OUTPUT('HPPLOT','JKL',0)
          ELSE
            CALL OUTPUT('VGTERM','JKL',0)
          END IF

C_____Initialize minimum and maximum of X axis
          XMIN=DAMPING(1)
          XMAX=DAMPING(NDAMP)

C_____Initialize plotting
          CALL INITT(30)
          CALL BINITT
```

```
C__________Set location and size of graph depending on output
C__________device
           IF(DEVICE.EQ.'P') THEN

C____________Initialize location and size of graph
             CALL SWCHAR(1)
             MINHOR = 80
             MAXHOR = 400
             MINVER = 150
             MAXVER = 400

C____________Set page location
             CALL SLIMX(MINHOR,MAXHOR)
             CALL SLIMY(MINVER,MAXVER)

           ELSE IF(DEVICE EQ 'H') THEN

C____________Initialize location and size of graph
             CALL SWCHAR(1)
             MINHOR = 120
             MAXHOR = 520
             MINVER = 160
             MAXVER = 630

C____________Set page location
             CALL SLIMX(MINHOR,MAXHOR)
             CALL SLIMY(MINVER,MAXVER)

           END IF

C__________Set other details of graph
           CALL INITGRPH2

C__________Plot the points on graph
           IF(NPNT NE.0)THEN
             CALL NPTS(NPNT)
             CALL CHECK(XARRAY,YARRAY)
             CALL DSPLAY(XARRAY,YARRAY)
           END IF

C__________End plotting
           CALL FINITT(0.0)
           CALL TRNSLI(1)

C__________Ask user if he wants output on any other device
           TYPE *
           TYPE *,'Any other output device ? (Y/N) '
           READ(6,100) MOREDEVIC

         END DO

C_________Ask user if he wants output for another acceptable
C_________RMS error value
         TYPE *
         TYPE *,'Any other acceptable RMS error value ? (Y/N) '
         READ(6,100) MOREACCEPT

       END DO

C_______Format statement
  100  FORMAT(A)

       RETURN
       END


C_____________________________Subroutine INITGRPH2
C_______Sets details of graph
       SUBROUTINE INITGRPH2

C_______Specification statements
       COMMON /NUMINPUT/ NCOMP, NTIME, NFREQ, NDAMP

C_______Set limits of X and Y axes depending on which system
C_______is involved
       CALL DLIMX( 001.10 )
       IF(NTIME EQ 1) THEN
         CALL DLIMY( 01.100 )
       ELSE
         CALL DLIMY(1 .10000 )
       END IF

C_______Set type of frame, no grid line
       CALL XFRM(2)
       CALL YFRM(2)
```

```
C_____Set tic marks length
      CALL XLEN(6)
      CALL YLEN(6)

C_____Set density of tic marks
      CALL XDEN(10)
      CALL YDEN(8)

C_____Set maximum number of character in tic mark labels
C_____of Y axis
      CALL YWIDTH(5)

C_____Set logarithmic X and Y axes
      CALL XTYPE(2)
      CALL YTYPE(2)

C_____Set no solid line (dots only)
      CALL LINE(-4)

C_____Set character size for tic mark labels
      CALL PLCHAR(25.45)

      RETURN
      END
```

APPENDIX K

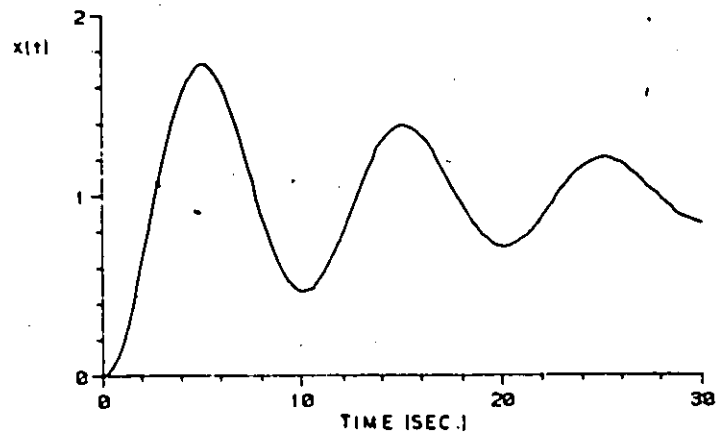
Program Run Examples

Contents

- Program RESPLT
- Program RESPONS
- Program RMSERR

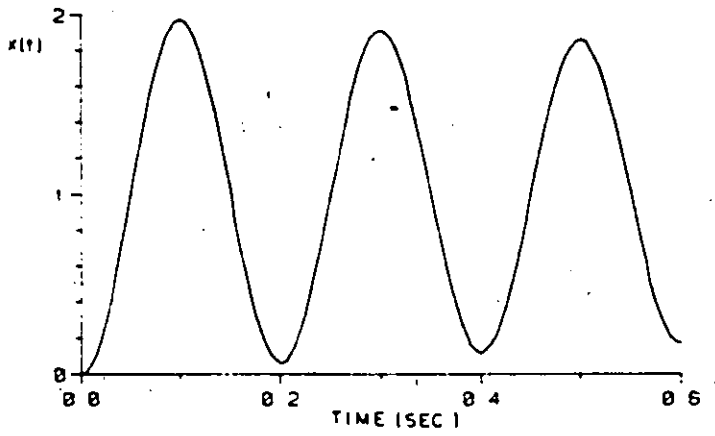
Program RESPLT

```
$ SET WORKING_SET_LIMIT 512
$ SET PROCESS_PRIORITY 8
$
$ RUN RESPLT

Enter System : A, B, C or D
C

Enter input parameters :
      Damping Ratio:       ZETA
      Natural Frequency:   Fn
      Compensation Factor: THETA PRIME
      Computation Time:    Tc
.1 .1 1 .09

Satisfied with parameters ? (Y/N):
  ZETA=          0.1000
  Fn  =          0.1000
  THETA PRIME=   1.0000
  Tc  =          0.0900
Y
Switch PACER to VAX port and press RETURN.


Bootstrap PACER to establish connection
(message when connection established)

        *    PACER CONNECTION ESTABLISHED    *

             *   VAX/PACER NOW RUNING    *

Solution time =    30.00000      Sec.

Enter output device:
      Enter  G: for VGT Screen
             P: for Line-Printer
             H: for HP Plotter
G
```



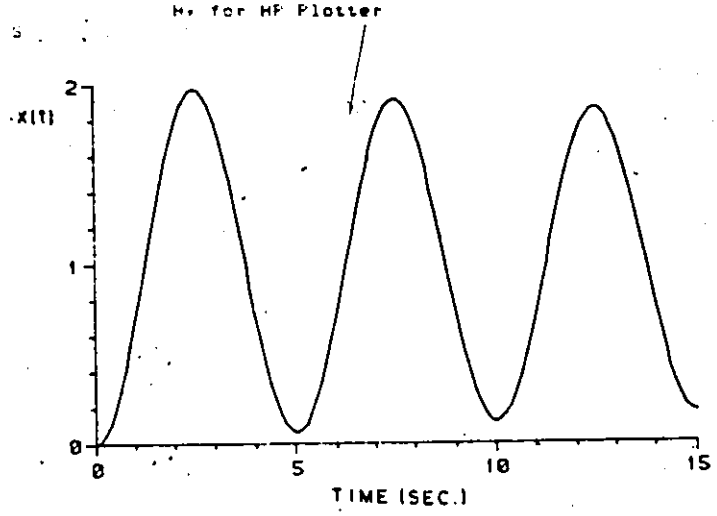

```
Any other output device ? (Y/N):
N

Output of another curve ? (Y/N):
Y


System = C

Satisfied with system ? (Y/N):
N
```

```
Enter System : A, B, C or D
A

Enter input parameters :
      Damping Ratio,      ZETA
      Natural Frequency,  Fn
.01 5

Satisfied with parameters ? (Y/N):
  ZETA=   0.010
  Fn  =   5.000

Enter output device:
      Enter  S, for VDT screen
             P, for Line-Printer
             H, for HP Plotter
S
```



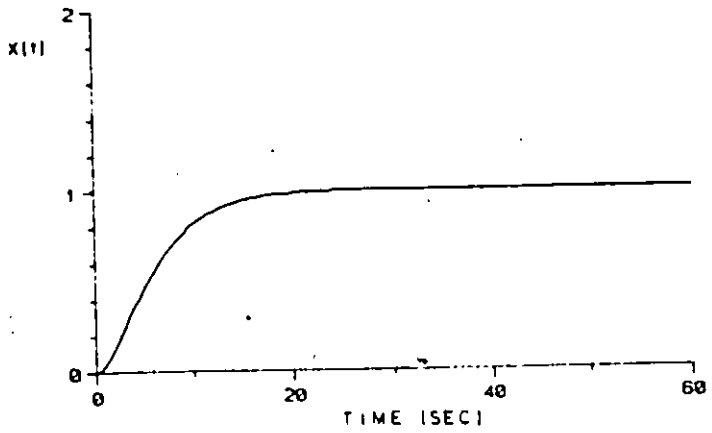

```
Any other output device ? (Y/N):
N

Output of another curve ? (Y/N):
Y

System
N

Satisfied with System ? (Y/N):
N

Enter System : A, B, C or D
D

**** System 'D' is not implemented ****

Output of another curve ? (Y/N):

Enter System : A, B, C or D

Enter input parameters :
      Damping Ratio,       ZETA
      Natural Frequency,   Fn
      Linear Extrapolation Factor, THETA PRIME
      Extrapolation Time,  Tc
.01 5 .9 .2

Satisfied with parameters ? (Y/N):
  ZETA            0.0100
  Fn              5.0000
  THETA PRIME=    0.9000
  Tc              0.2000
```

```
Enter input parameters :
      Damping Ratio:       ZETA
      Natural Frequency:   Fn
      Compensation Factor: THETA PRIME
      Computation Time:    Tc
.01 .2 0 .005

Satisfied with parameters ? (Y/N):
  ZETA=          0.0100
  Fn  =          0.2000
  THETA PRIME=   0.0000
  Tc  =          0.0050
Y
Solution time =   15.00000     Sec.

Enter output device:
      Enter  S: for VGT Screen
             P: for Line-Printer
             H: for HP Plotter
S
```




```
Any other output device ? (Y/N):
N

Output of another curve ? (Y/N):
Y

System = 1

Satisfied with system ? (Y/N):
Y

Enter input parameters :
      Damping Ratio:       ZETA
      Natural Frequency:   Fn
      Compensation Factor: THETA PRIME

      Computation Time:    Tc
.02 .2 0 .003

Satisfied with parameters ? (Y/N):
  ZETA=          0.0200
  Fn  =          0.2000
  THETA PRIME=   0.0000
  Tc  =          0.0030
Y
Solution time =   15.00000     Sec.

    ******   TIME CONSTRAINT   ******

Enter output device:
      Enter  S: for VGT Screen
             P: for Line-Printer
             H: for HP Plotter
```

```
Any other output device ? (Y/N):
N

Output of another curve ? (Y/N):
Y

System : C

Satisfied with system ? (Y/N):
N

Enter System : A, B, C or D
B

Enter input parameters :
      Damping Ratio:      ZETA
      Natural Frequency:  Fn
: ..5

  ...ied with parameters ? (Y/N):
  ZETA=    1.500
  Fn  =    0.050
Y
Switch FACER to VAX port and press  RETURN


Bootstrap FACER to establish connection
(message when connection established:

        %    FACER CONNECTION ESTABLISHED    %

           %    VAX/FACER NOW RUNING    %

Solution time =    ..0000      Sec.

Enter output device:
      Enter  S: for VDT Screen
             P: for Line-Printer
             H: for HP Plotter
S
```




```
Any other output device ? (Y/N):
N

Output of another curve ? (Y/N):
N
$
```

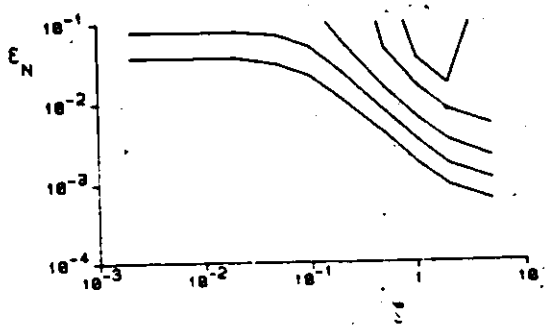
Program RESPONS

```
$ RUN RESPONS

Enter System : A, B, C OR D
A

System : A

Satisfied with System ?
Y
$
$ RUN RESPONS

Enter System : A, B, C OR D
D

System : D

Satisfied with System ?
Y

Enter Theta Prime number where:
Theta Prime number:   1       2       3       4       5
Theta Prime value:  0.000   0.500   1.000   1.500   2.000
2
$
```

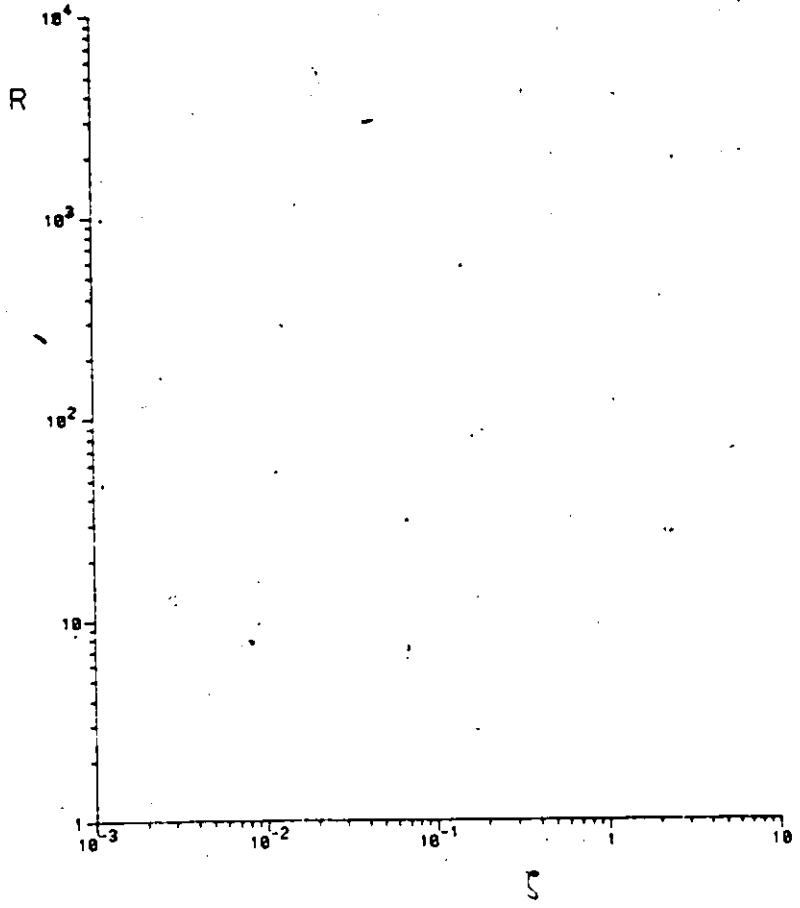
Program RMSERR

```
$ RUN RMSERR

Compare System 'I' to System 'J'
where 'I' and 'J' = A, B, C or D

Enter 'I','J'
D,A

*****    System 'D'              *****
*****    cannot be compared      *****
*****    to System 'A'           *****

Compare System 'I' to System 'J'
where 'I' and 'J' = A, B, C or D

Enter 'I','J'
D,A

Comparing System 'D' to System 'A'

Satisfied with Systems ?
Y

Enter Theta Prime number where:
Theta Prime number:   1       2       3       4       5
Theta Prime value:  0.000   0.500   1.000   1.500   2.000
2

Theta Prime = 0.50
Comparing System 'D' to System 'A' for Tc = 0.010

Comparing System 'D' to System 'A' for Tc = 0.025

Comparing System 'D' to System 'A' for Tc = 0.050

Comparing System 'D' to System 'A' for Tc = 0.100

Comparing System 'D' to System 'A' for Tc = 0.200
```

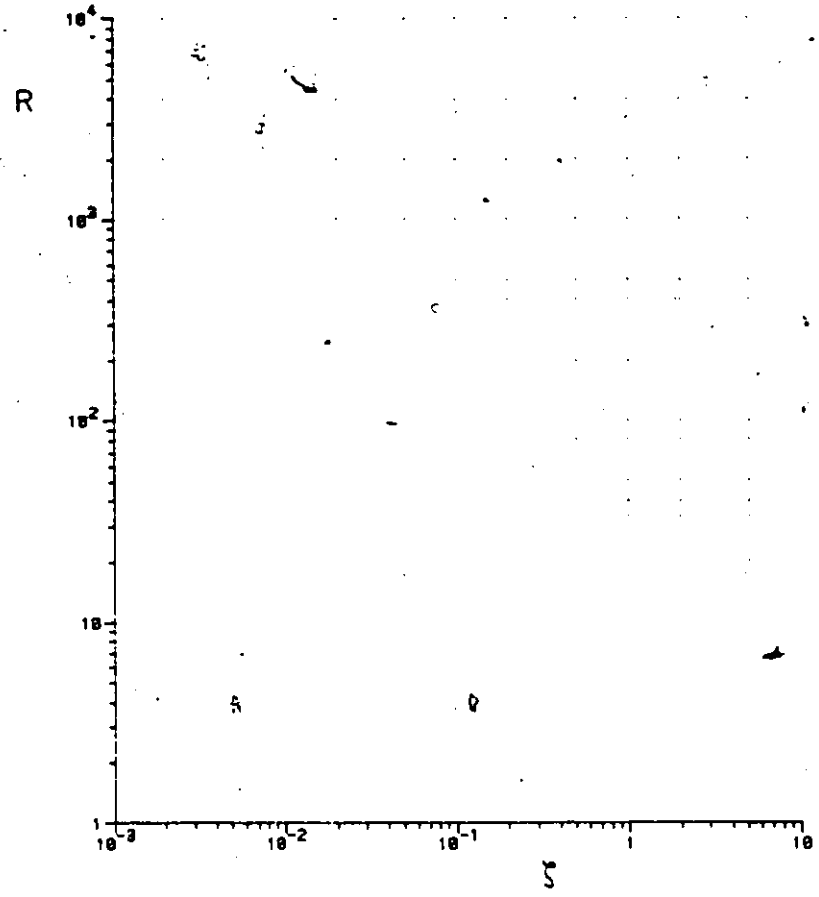
```
Enter output device where you
want graph of RMS error:
      Enter S for VGT Screen
            P for Line-Printer
            H for HP Plotter
S
```




```
Any other output device ? (Y/N):
N

Enter acceptable RMS error value
.01

Enter output device where you
want graph of acceptable error:
      Enter S for VGT Screen
            P for Line-Printer
            H for HP Plotter
S
```



```
Any other output device ? (Y/N):
N

Any other acceptable RMS error value ? (Y/N):
Y

Enter acceptable RMS error value
.01

Enter output device where you
want graph of acceptable error:
     Enter S: for VGT Screen
           P: for Line-Printer
           H: for HP Plotter
S
```




```
Any other output device ? (Y/N):
N

Any other acceptable RMS error value ? (Y/N):
N
$
```

APPENDIX L

Digital Equivalent Stability Boundaries Documentation

Contents

- Mathematical Basis
- Flowchart
- Listings

Mathematical Basis

The stability contour in the complex plane of system characteristic roots derived by Vichnevetsky[7] is given by

$$\lambda T_c = \frac{(Z-1)Z}{\theta_v Z + (1-\theta_v)} \qquad (L1)$$

where

$$\theta_v = 1.5\,\theta' \qquad (L2)$$

This may be transformed further to the (ζ, $\bar{R}$) plane by substituting

$$Z = e^{j\gamma} = \cos\gamma + j\sin\gamma \qquad (L3)$$

on the right hand side of equation (L1), letting γ vary in the interval $(0, \gamma_L]$, where γ_L is given by equation (L7), and substituting for the roots of the SOLDE

$$\lambda = \omega_n(-\zeta \pm \sqrt{\zeta^2 - 1}) = \frac{2\pi}{RT_c}(-\zeta \pm \sqrt{\zeta^2 - 1}) \qquad (L4)$$

where $\omega_n = 2\pi F_n$ and $R = (F_n T_c)^{-1}$. By these substitutions and algebraic and trigonometric manipulations, equation (L1) may be expressed in the form:

$$\frac{2\pi}{R}(-\zeta \pm \sqrt{\zeta^2 - 1}) = \mathrm{Re}(\theta_v, \gamma) + j\,\mathrm{Im}(\theta_v, \gamma) \qquad (L5)$$

where

$$Re(\theta_v, \gamma) = \frac{(2\theta_v - 1)\cos\gamma + (1-\theta_v)\cos 2\gamma - \theta_v}{\theta_v^2 + 2\theta_v(1-\theta_v)\cos\gamma + (1-\theta_v)^2} \qquad \text{(L5a)}$$

and

$$Im(\theta_v, \gamma) = \frac{(2\theta_v - 1)\sin\gamma + (1-\theta_v)\sin 2\gamma}{\theta_v^2 + 2\theta_v(1-\theta_v)\cos\gamma + (1-\theta_v)^2} \qquad \text{(L5b)}$$

The stability boundary (viz. Figure L-1 for a typical example) is given by equations (L5a) and (L5b) for a given value of θ_v and $\gamma \in [0, \gamma_L]$ where γ_L is the value of γ which satisfies

$$Im(\theta_v, \gamma) \Big|_{\gamma = \gamma_L} = 0 \qquad \text{(L6)}$$

i.e.,

$$\gamma_L = \begin{cases} \cos^{-1}((1 - 2\theta_v)/(2 - 2\theta_v)), & 0 \le \theta_v < .75 \\ \pi, & \theta_v \ge .75 \end{cases} \qquad \text{(L7)}$$

This limiting value of γ was determined by the fact that the roots of the system remain real for $\zeta = [1, \infty)$.

The characteristic roots of the system are given by the left hand side of equation (L5). Typical root loci are shown in Figure L-1. They are complex conjugates when $\zeta \in [0, 1)$ and real when $\zeta \in [1, \infty)$. The root locus in the $\lambda\tau_c$ complex plane

intersects the stability boundary as follows:

(1) If $\zeta \in [0, 1)$, the intersection is given by equating the real and imaginary parts:

$$Re(\theta_v , \gamma) = -2\pi\zeta / R \qquad (L8)$$

or

$$R = -2\pi\zeta / Re(\theta_v , \gamma) \qquad (L8a)$$

and

$$Im(\theta_v , \gamma) = \pm 2\pi\sqrt{1 - \zeta^2} / R \qquad (L9)$$

Substituting equation (L8a) into (L9) and solving for ζ yields:

$$\zeta = \left[(Im / Re)^2 + 1 \right]^{-1/2} \qquad (L10)$$

The stability boundary in the (ζ, R) plane is obtained by substituting values of $\gamma \in [0, \gamma_L]$ into equations (L8a) and (L10).

(2) If $\zeta \in [1, \infty)$, the intersection is given by equating the real parts of (L5) and solving for R to obtain

$$R = -2\pi (\zeta + \sqrt{\zeta^2 - 1}) / Re(\theta_v , \gamma_L) \qquad (L11)$$

where γ_L is given by equation (L7) and $Re(\theta_V, \gamma_L)$ is obtained by substituting $\gamma = \gamma_L$ into equation (L5a).

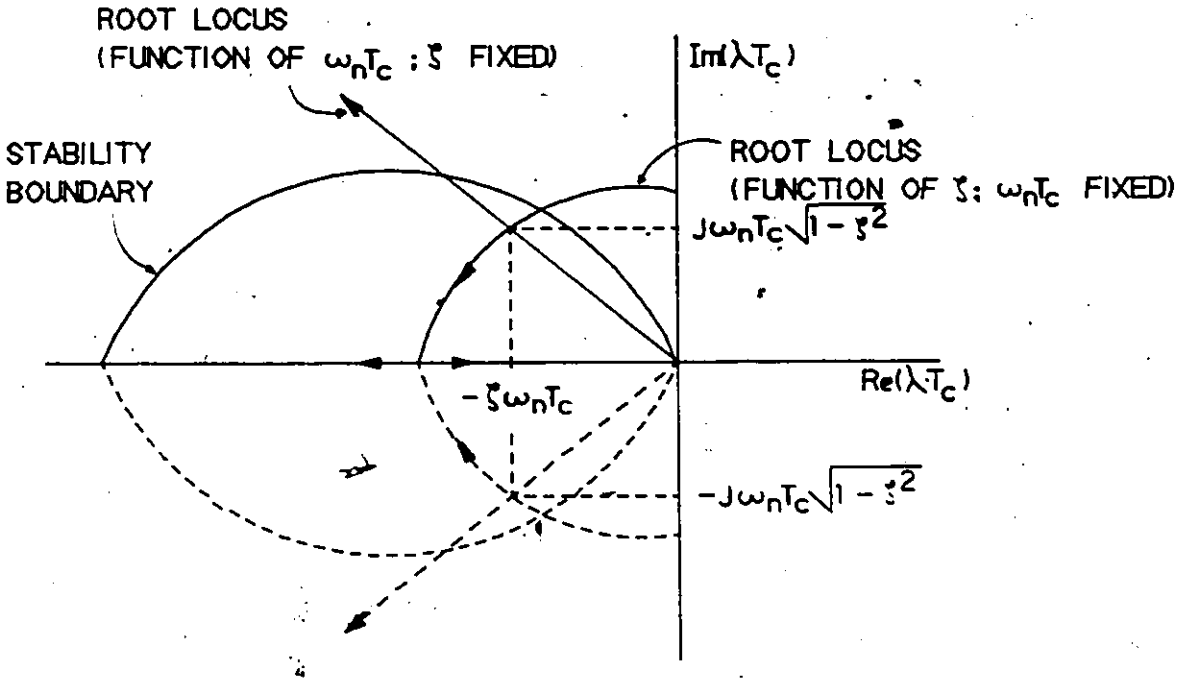


Figure L–1 : Typical stability boundary and root locus in λT_c complex plane

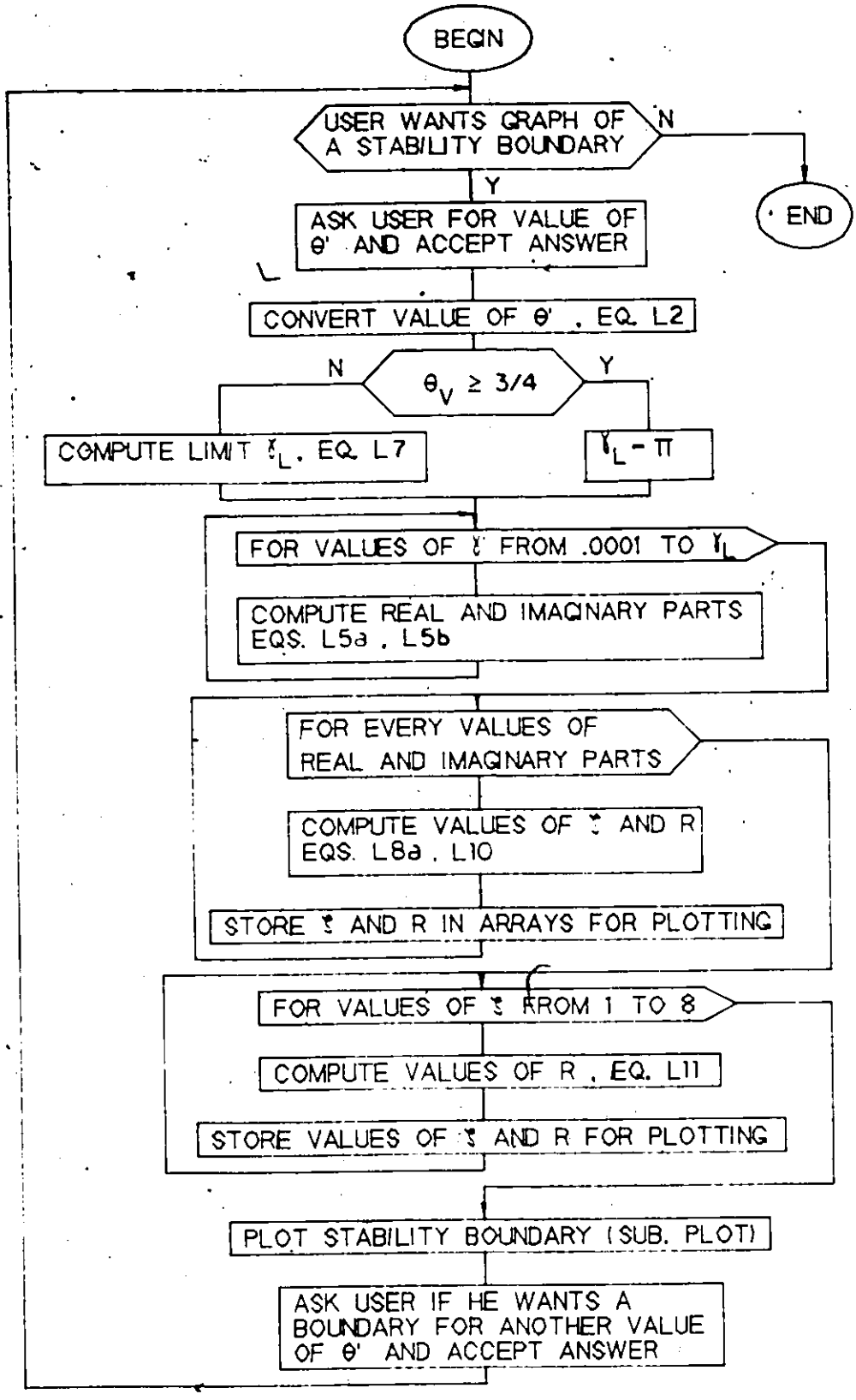


Figure L–2 : Program STABILITY flowchart

```
C_____Program STABILITY
C_____Computes the stability boundaries for the discrete-time
C_____of sampled-data model

C_____Specification statements
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION IMAG(1000), REEL(1000)
      REAL XAXS(1000), YAXS(1000)
      REAL XAXIS(1000), YAXIS(1000)

      CHARACTER MORECURV

      COMMON /RESPONSDAT/ NPOINT, YAXIS, XAXIS

C_____Initialize constant PI
      PI = 3.14159200

C_____Compute stability boundaries while user wants one for
C_____any value of theta prime
      MORECURV = 'Y'
      DO WHILE(MORECURV.EQ.'Y')

C_____Ask user for value of theta prime and accept answer
        TYPE *
        TYPE *,'Enter Theta Prime '
        READ(6,*) THETAP

C_____Convert value of theta prime
        THETA = 1.5*THETAP

C_____Compute limit of angle GAMMA (i.e., where imaginary part
C_____becomes zero)
        IF(THETA.GE..3/4.) THEN
          GAMMAL = PI
        ELSE
          GAMMAL = DACOS((1-2.*THETA)/(2.-2.*THETA))
        END IF

C_____Compute real and imaginary parts for angle values GAMMA=.0001
C_____to .01
        I = 0
        DO GAMMA = .0001, .01, .0001
          I=I+1
          DEN = THETA**2 + 2.*(1-THETA)*THETA*COS(GAMMA) + (1-THETA)**2
          REEL(I) = ((1 - THETA)*DCOS(2.*GAMMA) - (1-2.*THETA)*DCOS(GAMMA)
     -              - THETA)/DEN
          IMAG(I) = ((1-THETA)*DSIN(2.*GAMMA) - (1-2.*THETA)*DSIN(GAMMA))/
     -              DEN
        END DO

C_____Compute real and imaginary parts for angle values GAMMA=.01
C_____to .9
        DO GAMMA = .01, .9, .01
          I=I+1
          DEN = THETA**2 + 2.*(1-THETA)*THETA*COS(GAMMA) + (1-THETA)**2
          REEL(I) = ((1 - THETA)*DCOS(2.*GAMMA) - (1-2.*THETA)*DCOS(GAMMA)
     -              - THETA)/DEN
          IMAG(I) = ((1-THETA)*DSIN(2.*GAMMA) - (1-2.*THETA)*DSIN(GAMMA))/
     -              DEN
        END DO

C_____Compute real and imaginary parts for angle values GAMMA=.9
C_____to limit GAMMAL
        DO GAMMA = .9, GAMMAL, .003
          I=I+1
          DEN = THETA**2 + 2.*(1-THETA)*THETA*COS(GAMMA) + (1-THETA)**2
          REEL(I) = ((1 - THETA)*DCOS(2.*GAMMA) - (1-2.*THETA)*DCOS(GAMMA)
     -              - THETA)/DEN
          IMAG(I) = ((1-THETA)*DSIN(2.*GAMMA) - (1-2.*THETA)*DSIN(GAMMA))/
     -              DEN
        END DO

C_____Compute real and imaginary parts for the exact value of the limit
C_____GAMMAL
        GAMMA = GAMMAL
        I=I+1
        DEN = THETA**2 + 2.*(1-THETA)*THETA*COS(GAMMA) + (1-THETA)**2
        REEL(I) = ((1 - THETA)*DCOS(2.*GAMMA) - (1-2.*THETA)*DCOS(GAMMA)
     -            - THETA)/DEN
        IMAG(I) = ((1-THETA)*DSIN(2.*GAMMA) - (1-2.*THETA)*DSIN(GAMMA))/
     -            DEN
        NPNT = I
```

```
C_________Compute and store values of ZETA and R for each values of IMAG and
C_________REEL computed above .The real part must be negative
          I=0
          DO J = 1. NPNT
            IF(REEL(J) LT 0 ) THEN
              ZETA = 1 /(DSQRT((IMAG(J)/REEL(J))**2  + 1 ))
              R = -(2 *PI*ZETA)/REEL(J)
              I=I+1
              XAXIS(I) = ZETA
              YAXIS(I) = R
            END IF
          END DO

C_________Compute and store values of R for values of ZETA=1 to 2
          DO ZETA = 1 . 2 .  05
            I=I+1
            R = -(2 *PI*(ZETA+DSQRT(ZETA**2 -1 )))/REEL(NPNT)
            XAXIS(I) = ZETA
            YAXIS(I) = R
          END DO

C_________Compute and store values of R for values of ZETA=2 to 8
          DO ZETA = 2 . 8 .  3
            I=I+1
            R = -(2 *PI*(ZETA+DSQRT(ZETA**2 -1 )))/REEL(NPNT)
            XAXIS(I) = ZETA
            YAXIS(I) = R
          END DO
          NPOINT = I

C_________Plot the stability boundary
          CALL PLOT

C_________Ask user if he wants a boundary for another theta prime
C_________value
          TYPE *
          TYPE *,'Another stability boundary ?'
          READ(6,100) MORECURV
        END DO

        Format statement
  100   FORMAT(A)

        END


C_____________________________Subroutine PLOT
C_______Plots stability boundary
        SUBROUTINE PLOT

C_______Specification statements
        CHARACTER ANSRDEVI, DEVICE
        LOGICAL MOREDEVIC

        DIMENSION RESPONSE(1000), RESTIME(1000)

        COMMON /RESPONSDAT/ NPOINT,RESPONSE,RESTIME

C_______Plot curve while user wants output on any device
        MOREDEVIC= TRUE
        DO WHILE(MOREDEVIC)

C_________Ask user for output device
          TYPE *
          TYPE *,'Enter output device'
          TYPE *,'     Enter  S. for VGT Screen'
          TYPE *,'            P. for Line-Printer'
          TYPE *,'            H. for HP Plotter'
          READ(6,100) DEVICE

C_________Initialize output device
          IF(DEVICE EQ 'P') THEN
            CALL OUTPUT('PRINTX','JKL',0)
          ELSE IF(DEVICE EQ 'H') THEN
            CALL OUTPUT('HPPLOT','JKL',0)
          ELSE
            CALL OUTPUT('VGTERM','JKL',0)
          END IF
```

```
C_________Initialize plotting
          CALL INITT(30)
          CALL BINITT

C_________Define limits of X and Y axes
          CALL DLIMX(0.002,8 )
          CALL DLIMY(1.,10000 )

C_________Set type of axes to logarithmic
          CALL XTYPE(2)
          CALL YTYPE(2)

C_________Set no grid lines
          CALL XFRM(2)
          CALL YFRM(2)

C_________Define length of tic marks
          CALL XLEN(6)
          CALL YLEN(6)

C_________Define density of tic marks
          CALL XDEN(8)
          CALL YDEN(8)

C_________If output is going on printer or plotter, define
C_________location on paper and size of graph, and set software
C_________characters in order to adjust label sizes
          IF(DEVICE EQ 'P') THEN
            CALL SLIMX(100,375)
            CALL SLIMY(50,190)
            CALL SWCHAR(1)
            CALL PLCHAR(30,50)
          ELSE IF(DEVICE EQ 'H') THEN
            CALL SLIMX(120,520)
            CALL SLIMY(160,630)
            CALL SWCHAR(1)
            CALL PLCHAR(25,45)
          END IF

C_________Set line to solid
          CALL LINE(0)

C_________Plot curve
          CALL NPTS(NPOINT)
          CALL CHECK(RESTIME,RESPONSE)
          CALL DSPLAY(RESTIME,RESPONSE)

C_________End plotting
          CALL FINITT(0,0)
          CALL TRNSLT(1)

C_________Ask user if he wants output on any other device
          TYPE *
          TYPE *,'Any other output device ? (Y/N) '
          READ(6,100) ANSRDEVI
          IF(ANSRDEVI EQ 'N') THEN
            MOREDEVIC= FALSE
          END IF

        END DO

C_________Format statement
  100   FORMAT(A)

        RETURN
        END
```