

AD A 079621

RADC-TR-79-266
Final Technical Report
October 1979

(12)

LEAD



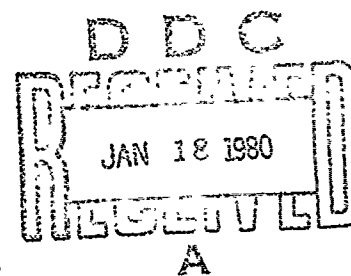
COMPUTER ROUTINES FOR USE IN SECOND ORDER VOLTERRA IDENTIFI- CATION OF EMI

University of South Florida

V. K. Jain
J. S. Osman

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DDC FILE COPY



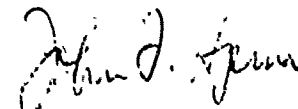
ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

80 1 21 055

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-266 has been reviewed and is approved for publication

APPROVED:



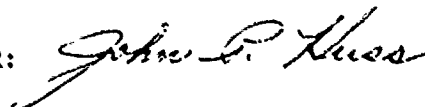
JOHN F. SPINA
Project Engineer

APPROVED:



DAVID C. LUKE, Lt Col, USAF
Chief, Reliability & Compatibility Division

FOR THE COMMANDER:



JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBCT), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DDC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER RADC-TR-79-266	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) COMPUTER ROUTINES FOR USE IN SECOND-ORDER VOLTERRA IDENTIFICATION OF EMI.		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 1 Oct 77 - 30 Sep 78	
7. AUTHOR(s) V. K. Jain J. S. Osman		6. PERFORMING ORG. REPORT NUMBER SS-18VRA	
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of South Florida Department of Electrical Engineering Tampa FL 33620		8. CONTRACT OR GRANT NUMBER(s) F30602-75-C-0118	
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RBCT) Griffiss AFB NY 13441		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 454002P1	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		12. REPORT DATE Nov 1979	
12. 89		13. NUMBER OF PAGES 91	
		15. SECURITY CLASS (of this report) UNCLASSIFIED	
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same			
18. SUPPLEMENTARY NOTES RADC Project Engineer: John F. Spina (RBCT)			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Pencil-of-functions method Transfer functions Adaptive scaling System identification Gram matrix Perturbation theory Parameter estimates Residues of quadratic Iterative correction Integrators transfer function Matrix inversion Volterra system Ill-conditioned matrix			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Computer routines are developed for application in Volterra modeling of weakly nonlinear systems. The FORTRAN program "IGRAM" identifies the parameters of a linear black-box model as given by the pencil-of-functions method. The FORTRAN program "HPMINV" yields high accuracy inversion of residue matrices that arise in Volterra system identification.			

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408631

15

PREFACE

This effort was conducted by University of South Florida under the sponsorship of the Rome Air Development Center Post-Doctoral Program for Rome Air Development Center. Mr. John F. Spina RADC/RBCT was the task project engineer and provided overall technical direction and guidance.

The RADC Post-Doctoral Program is a cooperative venture between RADC and some sixty-five universities eligible to participate in the program. Syracuse University (Department of Electrical Engineering), Purdue University (School of Electrical Engineering), Georgia Institute of Technology (School of Electrical Engineering), and State University of New York at Buffalo (Department of Electrical Engineering) act as prime contractor schools with other schools participating via sub-contracts with prime schools. The U.S. Air Force Academy (Department of Electrical Engineering), Air Force Institute of Technology (Department of Electrical Engineering), and the Naval Post Graduate School (Department of Electrical Engineering) also participate in the program.

The Post-Doctoral Program provides an opportunity for faculty at participating universities to spend up to one year full time on exploratory development and problem-solving efforts with the post-doctorals splitting their time between the customer location and their educational institutions. The program is totally customer-funded with current projects being undertaken for Rome Air Development Center (RADC), Space and Missile Systems Organization (SAMSO), Aeronautical System Division (ASD), Electronics Systems Division (ESD), Air Force Avionics Laboratory (AFAL), Foreign Technology Division (FTD), Air Force Weapons Laboratory (AFWL), Armament Development and Test Center (ADTC), Air Force Communications Service (AFCS), Aerospace Defense Command (ADC), HW USAF, Defense Communications Agency (DCA), Navy, Army, Aerospace Medical Division (AMD), and Federal Aviation Administration (FAA).

Further information about the RADC-Doctoral Program can be obtained from Mr. Jacob Scherer, RADC/RBC, Griffiss AFB, NY, 13441, telephone Autovon 587-2543, Commercial (315) 330-2543.

ACKNOWLEDGEMENTS

The authors are deeply indebted to Mr. John. F. Spina and Mr. D. J. Kenneally for their assistance and useful criticism throughout the duration of this project. They also wish to express their gratitude to Dr. Donald D. Weiner for the many stimulating and helpful discussions on different aspects of this work. Appreciation is also extended to Mr. Jacob Scherer of the Post-Doc Program for his helpfulness.

The authors also acknowledge the assistance of Owen L. Godwin, Jr. in certain parts of this report.

TABLE OF CONTENTS

Chapter	Title	Page
I	Introduction	1
II	Pencil of Function Method for Identification of Network Transfer Functions	2
	Theory	3
	Application examples	8
	Program description	14
III	Computer Routines for use in Second-Order Volterra Modeling of EMI	18
	Adaptive scaling	19
	Perturbation	25
	Iterative correction	29
	Application	32
	Program description	39
	References	42
	Appendix A - IGRAM listing	44
	Appendix B - HPMINV listing	62
	Appendix C - Wide band system considerations	74
	Appendix D - s-domain to z domain conversion	80

LIST OF FIGURES

Chapter II

Fig.	Title	Page
1	Measurement System	4
2	Fourth order model identification of $H_4(s)$	9
3	Common-emitter amplifier circuit	10
4	Second order model identification of $H_L(s)$	13
5	First order model identification of $H_{MH}(s)$	13
6	Second order model identification of $H_H(s)$	13

Chapter III

Fig.	Title	Page
1	Effect of small changes in ϵ on C^{-1}	29
2	Block diagram representation of iterative correction method	30
3	Flow diagram of HPMINV program options	41

Appendix C

Fig.	Title	Page
C1	Common-emitter amplifier circuit	74
C2	Magnitude characteristic (Bode plot) of a wide band system	77

COMPUTER ROUTINES FOR USE IN SECOND-ORDER
VOLTERRA MODELING OF EMI

I. INTRODUCTION

Research described here consists of the development of certain computer routines to aid Volterra modeling of weakly nonlinear systems [1], [2]. Volterra series representation, a dynamic generalization of the familiar power series, is ideal for representing devices and systems with frequency-dependent mild nonlinearities as in the case of a transistor. The technique has been applied to the analysis of communication receiver response to radio frequency interference [3]. Other applications include intermodulation distortion analysis of transistor feedback amplifiers [4], nonlinear characterization of IMPATT diodes and microwave amplifiers [5], and analysis of channels with soft limiter.

Rome Air Development Center has in the recent past supported several efforts to put this analytical tool to use in the electromagnetic interference and compatibility field. (In practical terms one of the major outcomes of the efforts is the IAP program, a computer program for the prediction of Intra-System Electromagnetic Compatibility). A current direction of interest is the estimation of Volterra kernels of a system from its experimentally observed input-output responses. Interest in this black-box approach arises for several reasons, the most salient being cost effectiveness in testing, and simplicity of resulting models for complex on-board communication systems.

Weiner and Ewen [1], [2] have provided an approach to finding the parameters of the kernels, specifically the poles and residues of the multivariable transfer functions $P(s_1, \dots, s_m)$. The poles and residues of the linear TF, $H_1(s)$, are determined using Jain's method [6], [7] (pencil-of-functions identification method). Then, for somewhat larger amplitudes, where the quadratic TF $H_2(s_1, s_2)$ has non-negligible influence, the contribution $y_2(t)$ is determined by subtracting $y_1(t)$, the predicted response of $H_1(s)$, from $y(t)$. The poles of the quadratic TF are known in terms of the poles of the linear TF, so that only the residues of $H_2(s_1, s_2)$ need be -- and in principle, can be -- determined. A similar procedure is adopted for determining the parameters of the cubic kernel, and so on.

The computer programs presented in this report are:

- | | |
|--------|---|
| IGRAM | Program for black-box identification of a linear transfer function using pencil-of-functions method. |
| HPMINV | High precision matrix inversion routine for use in determination of the residues of the quadratic Volterra TF. Perturbation theory and iterative corrections are used to enable accurate inversion even for wideband system matrices. |

II. PENCIL OF FUNCTIONS METHOD FOR IDENTIFICATION OF NETWORK TRANSFER FUNCTIONS

Determining the model of a network from its observed input-output responses represents the inverse of the analysis problem. Interest in this arises from the frequent need for a relatively simple mathematical description of the system so that behavior for other anticipated inputs may be predicted up to acceptable accuracies. Like the analysis problem, there are several approaches available in the literature for the inverse, or, as it is often called, the "identification" problem. To name a few, (a) Prony's method [8], (b) gradient methods, such as Newton [9] and quasi-linearization [10], (c) least-squares and generalized least-squares methods [11],[12], (d) maximum-likelihood methods [13],[14],[15].

All of the methods stated above possess certain advantages and, as may be expected, certain disadvantages peculiar to each particular method. Stated very broadly, sensitivity to noise, slow convergence to the solution, and excessive computational complexity are some of the possible disadvantages. The objective of this section is to describe in a semi-rigorous way the pencil-of-functions identification method [7]. Further, in this section the method is extended to the case where general first order $-s$, $\mu_i(s) = (b_i s + c_i)/(s + a_i)$ are used in the processing system instead of ideal integrators (note that the ideal integrator is a special case of this filter; set $b_i = a_i = 0$). A high accuracy FORTRAN program, developed for the case of ideal-integrator processing units, is also presented.

The method offers the advantages of mathematical simplicity, closed-form solution to the problem, which is optimal in the generalized least-squares sense and suboptimal in the strict least-squares sense, and robustness to noise. The disadvantage of the method is that the variances of additive noise, when present, must be determined in a separate experiment in order that unbiased parameter estimates may be computed.

2.1 THEORY

The problem of identifying the transfer function of a network from its input-output responses can be formulated in discrete-time domain as follows. Given the pair $x(k)$, $y(k)$ (or noise-corrupted versions of these; call them $u(k)$, $v(k)$) find the transfer function $H(z)$ that produces a response matching $y(k)$ (or $v(k)$) when excited by $x(k)$ (or $u(k)$). More specifically, this involves determination of the parameters a_i , b_i in

$$\begin{aligned} Y(z) &= H(z) X(z) \\ &= \left(\frac{b_0 + b_1 z^{-1} + \dots + b_n z^{-n}}{1 + a_1 z^{-1} + \dots + a_n z^{-n}} \right) X(z) \end{aligned} \quad (1)$$

from experimental input-output data. Note (1) can be written in time domain as

$$y(k) = - \sum_{i=1}^n a_i y(k-i) + \sum_{i=0}^n b_i x(k-i), \quad y(k) = 0 \text{ for } k \leq 0 \quad (2)$$

A. Measurement Signals

Before proceeding with the solution of the problem via pencil-of-function method, let us note that (2) can be written as

$$[y(k) \ y(k-1) \ \dots \ y(k-n) \ -x(k) \ -x(k-1) \ \dots \ -x(k-n)] \ \underline{\theta} = 0 \quad (3)$$

or, more concisely,

$$\underline{f}(k)^T \underline{\theta} = 0 \quad (4)$$

where the $2n + 2$ dimensional vector $\underline{f}(k)$ has the obvious definition and the vector $\underline{\theta}$, also $2n + 2$ dimensional, is given as

$$\underline{\theta} = \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} \quad (5)$$

Equation (4) represents a geometrical constraint upon the vectors $\underline{f}(k)$, namely that they are all orthogonal to $\underline{\theta}$.

To cast the problem of identifying $\underline{\theta}$ into a generalized least-squares

problem consider the measurement system of Fig. 1. The matter of choosing the first order filters which make up the measurement units $M_i(z)$ will not be considered here. It will suffice to say they must be chosen so that the cutoff frequencies of $M_i(z)$ span the frequency range of interest (i.e. they are spread in the pass-band of the network under test). Regardless of the choice of the measurement units the following useful observation arises.

Let

$$\underline{z}(k) = [y_0(k) \ y_1(k) \ \dots \ y_n(k) \ -x_0(k) \ -x_1(k) \ \dots \ -x_n(k)]^T$$

and define the matrix

$$C = \begin{bmatrix} c_{00} & c_{01} & \dots & c_{0n} \\ c_{10} & c_{11} & & c_{1n} \\ \vdots & \vdots & & \vdots \\ c_{n0} & c_{n1} & & c_{nn} \end{bmatrix} \quad (6a)$$

where c_{ji} are the coefficients of the polynomial

$$\sum_{i=0}^n c_{ji} z^{-i} = \frac{i}{i+1} (1-P_i z^{-1}) \frac{n}{i+1} (1-Q_i z^{-1}) \quad (6b)$$

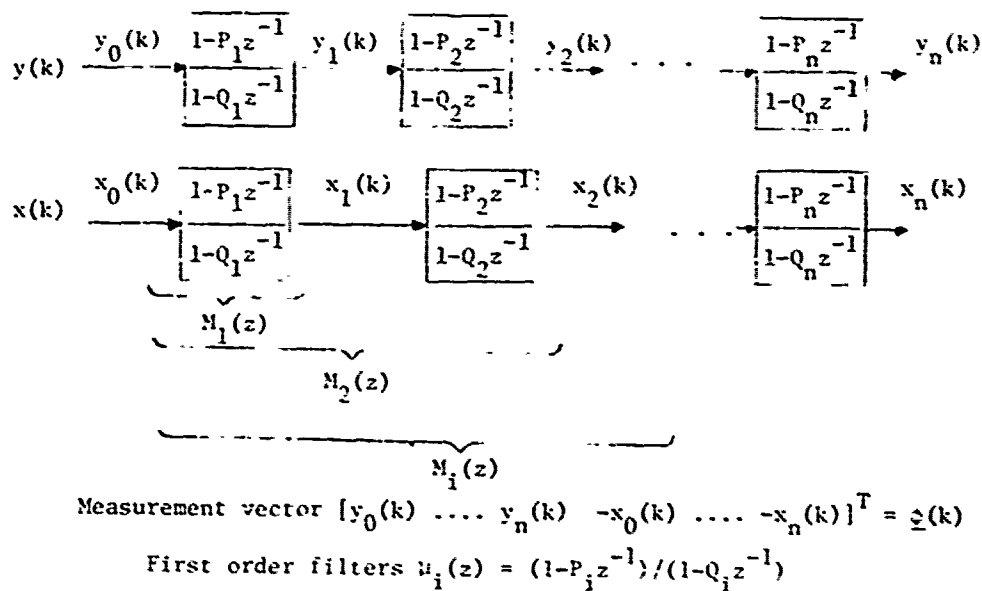


Fig. 1. Measurement system

Also define the vector $\underline{\gamma}$ henceforth called the synthetic parameter vector, as

$$\underline{\gamma} = \begin{bmatrix} \underline{\alpha} \\ \underline{\beta} \end{bmatrix} \quad (7a)$$

$$= \begin{bmatrix} C^{-1}_a \\ C^{-1}_b \end{bmatrix} \quad (7b)$$

$$= C^{-1}_\theta \quad (7c)$$

Then, it can be shown that the measurement vectors $\underline{\phi}_k$ are orthogonal to the synthetic parameter vector. The proof will be given in subsection 2.1C.

B. Solution

The problem of determining the synthetic parameter vector $\underline{\gamma}$ can be shown to reduce to finding the cofactors of a gram matrix. Specifically, it can be shown by use of the pencil-of-functions theorem [6] that the vector $\underline{\gamma}$ can be obtained in the following way:

1. Form the Gram matrix [16]

$$G_N = \sum_{k=0}^{K-1} \underline{\phi}(k) \underline{\phi}^T(k),$$

where $N = 2n + 2$

2. Find the diagonal cofactors of G_N ; call them Δ_{ii}
3. Then

$$\underline{\gamma}^T = \frac{1}{\sqrt{\Delta_{11}}} \left[\sqrt{\Delta_{11}}, \dots, \sqrt{\Delta_{NN}} \right]$$

Finally, using the transformation (7c) we have the desired parameter vector $\underline{\theta} = C \underline{\gamma}$

C. Proof of Measurement Filter Theorem

The relationship in (1), or equivalently (2), may be written as

$$\underline{a}^T \underline{\zeta} Y(z) = \underline{b}^T \underline{\zeta} X(z) \quad (8)$$

where

$$\begin{aligned} \underline{a} &= [1 \ a_1 \ \dots \ a_n]^T \\ \underline{b} &= [b_0 \ b_1 \ \dots \ b_n]^T \\ \underline{\zeta} &= [1 \ z^{-1} \ \dots \ z^{-n}]^T \end{aligned}$$

On the other hand the measurement signals have the following representation. Consider the output measurement signal $y_i(k)$; its transform is

$$\begin{aligned}
 Y_i(z) &= M_i(z) Y(z) \\
 &= \prod_{\ell=1}^i \frac{1-P_\ell z^{-1}}{1-Q_\ell z^{-1}} y(z) \\
 &= \frac{1}{D(z)} \prod_{\ell=1}^i (1-P_\ell z^{-1}) \prod_{\ell=i+1}^n (1-Q_\ell z^{-1}) Y(z) \\
 &= \frac{1}{D(z)} [c_{oi} \ c_{1i} \ \dots \ c_{ni}] \underline{z} Y(z)
 \end{aligned} \tag{9}$$

where

$$D(z) = \prod_{\ell=1}^n (1 - Q_\ell z^{-1}) \tag{10}$$

and the numbers c_{ji} are the same as defined in equation (6). Now the output measurement vectors may be written as

$$\underline{Y}(z) = [Y_0(z) \ \dots \ Y_n(z)]^T = \frac{1}{D(z)} C^T \underline{z} Y(z) \tag{11}$$

where the $(n+1) \times (n+1)$ matrix $C = [c_{ji}]$ is given by the numbers c_{ij} defined above. In a like manner, we have

$$\underline{X}(z) = [X_0(z) \ \dots \ X_n(z)]^T = \frac{1}{D(z)} C^T \underline{z} X(z) \tag{12}$$

We can now state and prove the measurement filter theorem.

Theorem - If the signals $y(k)$ and $x(k)$ satisfy (8) for some parameter vector $(\underline{a}, \underline{b})$, then the measurement vectors satisfy the orthogonality condition

$$[\underline{\alpha}^T \ \underline{\beta}^T] \begin{bmatrix} \underline{y}(k) \\ -\underline{x}(k) \end{bmatrix} = 0 \text{ for all } k \tag{13}$$

where

$$\underline{\alpha} = C^{-1} \underline{a} \tag{14}$$

$$\underline{\beta} = C^{-1} \underline{b} \tag{15}$$

Proof. The matrix C can be shown to be nonsingular. Therefore we can rewrite

(8) as

$$(C^{-1}\underline{a})^T C^T \underline{z} Y(z) = (C^{-1}\underline{b})^T C^T \underline{z} X(z)$$

or

$$\underline{\alpha}^T \frac{1}{d(z)} C^T \underline{z} Y(z) - \underline{\beta}^T \frac{1}{d(z)} C^T \underline{z} X(z) = 0$$

Upon substituting (11) and (12) this equation yields

$$[\underline{\alpha}^T \quad \underline{\beta}^T] \begin{bmatrix} \underline{Y}(z) \\ -\underline{X}(z) \end{bmatrix} = 0 \quad (16)$$

The result sought by the theorem is obtained immediately upon taking the inverse transform.

2.2 APPLICATION EXAMPLES

EXAMPLE 1

As a first example of the identification routine, data from the transfer function

$$H_4(s) = \frac{(10^6)}{s+7(10^6)} + \frac{3(10^7)}{s+7(10^7)} + \frac{(10^6)}{s+(10+j25)10^6} + \frac{(10^6)}{s+(10-j25)10^6}$$

was obtained. The input driving function was a square-wave followed by a decaying exponential. Two cycles of square wave with period 0.05μsec were used, followed by the decaying exponential with time constant approximately equal to 0.05μsec. This input was selected based on apriori knowledge of the network behavior. The spectral content of this input should supply a sufficient amount of energy to the fast mode ($s = -7 \times 10^7$) and to the slow and oscillatory modes for accurate identification.

Five hundred points of data were used (MP1=500) with a sampling interval $\Delta = 0.002\mu\text{sec}$. The option IREM =1 was used in this example because direct transmission could not be assumed. A fourth order model* (N=4) was desired for this network. The identified poles and residues of the model are given below, together with the actual ($H_4(s)$) poles and residues.

$H_4(s)$ Poles	Identified Poles	$H_4(s)$ Residues	Identified Residues
$-7. \times 10^6$	-7.0×10^6	$1.0(10^6)$	$1.0(10^6)$
$-7. \times 10^7$	-7.0×10^7	$30.0(10^6)$	$30.0(10^6)$
$-(10.+j25)10^6$	$-(10.0+j25.0)10^6$	$1.0(10^6)$	$1.0(10^6)$
$-(10.-j25)10^6$	$-(10.0-j25.0)10^6$	$1.0(10^6)$	$1.0(10^6)$

As seen, the identification of the transfer function was very accurate, and the corresponding mean-square and root-mean square errors for this identification are both 0.0%. Plots of the input-output data and the actual and model responses are shown in Fig. 2.

* A rational transfer function with an nth degree denominator is referred here as being of nth order.

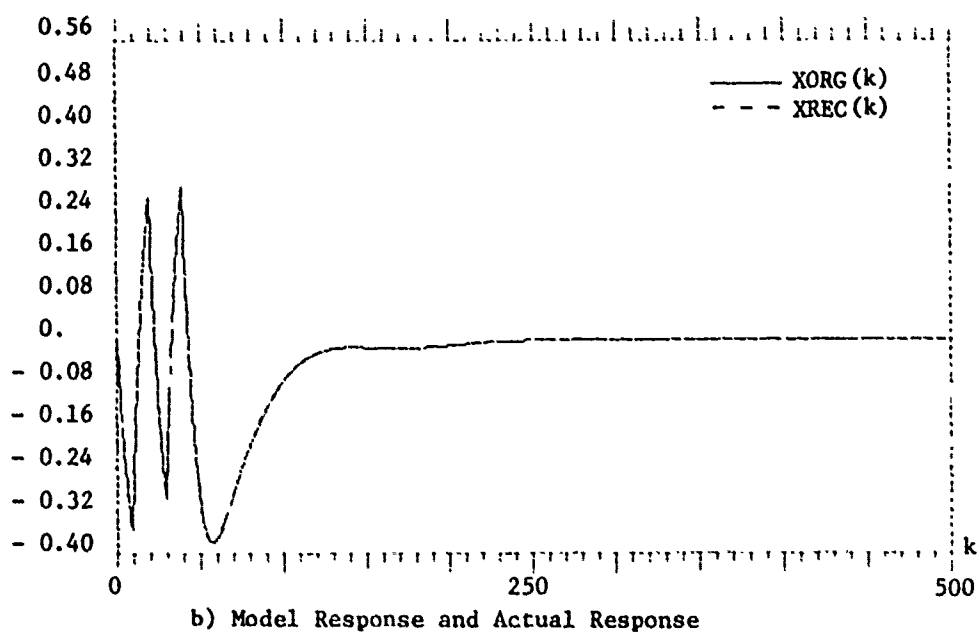
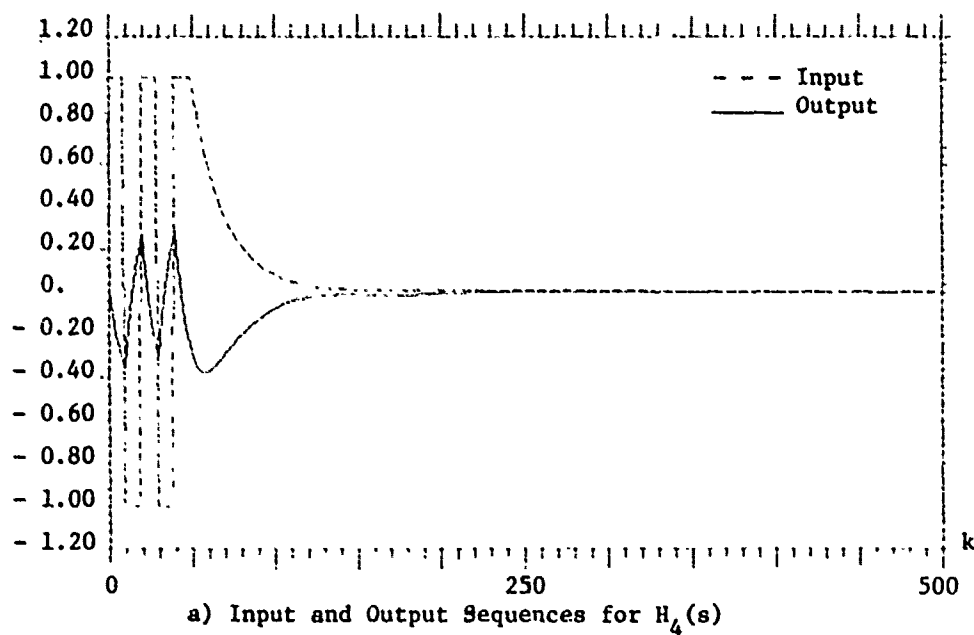


Fig. 2 . Fourth Order Model Identification of $H_4(s)$

Example 2

We examine the applicability of the identification technique to responses obtained from a wide-band transistor amplifier circuit. The schematic and equivalent model of the circuit are shown in Fig. 3.

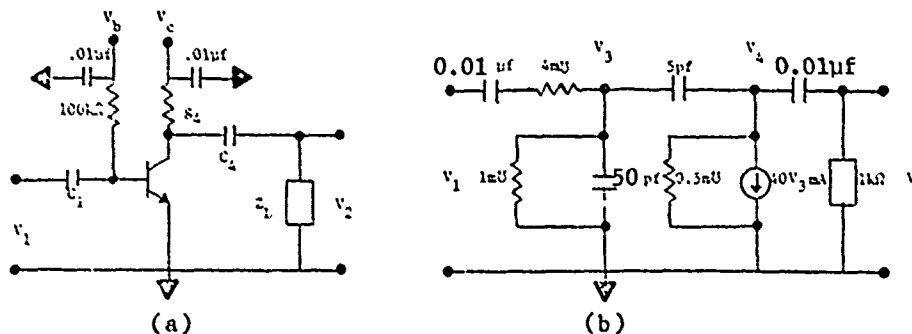


Fig. 3. (a) Schematic of common emitter amplifier circuit
(b) Equivalent circuit model

As shown in Appendix C, the network transfer function is

$$H(s) = \frac{V_2(s)}{V_1(s)} = \frac{8(10^7) s^2 (s-8000 (10^6))}{(s+.033(10^6))(s+.080(10^6))(s+25.2(10^6))(s+1205.1(10^6))} \quad (17)$$

The network function can be identified successfully only by performing separate tests in three different frequency regions:

- (i) Low frequency region (L)
- (ii) Mid to High frequency transition (MH)
- (iii) High Frequency region (H)

A discussion of these three regions is given in Appendix C. Here we shall focus primarily on the results of identification.

(i) Low frequency region

An adequate low frequency description of eq. (17) is given by *

$$H_L(s) = - \frac{(21.04) s^2}{(s+.033(10^6))(s+.080(10^6))} \quad (18)$$

It is therefore desirable that we seek a second-order ($N=2$) model of the network given by eq. (17). The input used is a single triangular pulse of duration $125\mu\text{sec}$. One thousand points ($NP1=1000$) of input-output responses,

* In practical applications, such approximations will of course not be available. But the designer, or even the test engineer, frequently has some idea of the critical frequencies of the system.

sampled at $\Delta = 0.25\mu s$ are used for modeling. The option IREM = 0 was used because our low-frequency model will exhibit direct transmission.

The computer program IGRAM yields the following low-frequency model:

$$\hat{H}_L(s) = - 20.125 \frac{(s-0.0015(10^6))(s+0.0012(10^6))}{(s+0.034(10^6))(s+0.075(10^6))} \quad (19)$$

Comparison of the identified model, eq. (19), with our low frequency approximation (eq. (18)) shows close agreement. The rms error between the measured network time response and the model response is 1.206%. Plots of the input-output signals and of the actual and model responses are given in Fig. 4.

ii) Mid to High frequency transition

As discussed in Appendix C, an adequate mid to high frequency transition description of eq. (17) is given by

$$H_{MH}(s) = - \frac{531.1 (10^6)}{(s + 25.2(10^6))} \quad (20)$$

which is a single pole function. Thus, we will attempt to model the circuit with a first-order transfer function ($N=1$). Since this approximate description (eq. (20)) does not exhibit direct transmission, IREM $\neq$ 0 should be used in the program. For this identification, IREM = 1 was selected. For reasons mentioned in Appendix C, bias was assumed present (IBIAS = 1) on the data. Five hundred points (MP1 = 500) of input-output signals, sampled at $\Delta = 0.01\mu sec$, were used. The excitation used was a narrow band signal with a center frequency near the critical frequency ($s=-25.2(10^6)$).

The computer program IGRAM yields the following mid to high frequency transition model:

$$\hat{H}_{MH}(s) = \frac{520.0 (10^6)}{(s+24.92(10^6))} \quad (21)$$

The rms error between this model and the actual network response (obtained from the original 4th order transfer function) is 1.99%. A graphical comparison is given in Fig. 5.

iii) High frequency region

An approximate high frequency description of eq. (17) is shown (Appendix C) to be

$$L_H(s) = \frac{8(10^7)(s-8000(10^6))}{(s+25.2(10^6))(s+1205.1(10^6))} \quad (22)$$

which is a two-pole function. We will therefore attempt to model the circuit with a second-order transfer function ($N=2$). Direct transmission cannot be assumed, so that $IREM = 1$ was used for the model. Five hundred points ($MP1 = 500$) of input-output signals, sampled at $\Delta = 0.00025 \mu\text{sec}$ were used. A narrow-band signal with a center frequency of 6300 Mrad/sec was used for network excitation.

The computer program yields the following high frequency model:

$$\hat{H}_H(s) = \frac{2.79(10^6)((s-19060(10^6))}{(s+25.7(10^6))(s+1139.5(10^6))} \quad (23)$$

The rms error between this model and the actual network response is 0.919%. A graphical comparison is given in Fig. 6.

We have now obtained a description of the network behavior in the three frequency regions of interest. The models obtained may be pieced together in such a way as to synthesize the overall behavior of the system. Omitting the details involved, the following model may be obtained from eq's. (19), (21) and (23):

$$\hat{H}(s) = \frac{62.2(10^6)(s-.0015(10^6))(s+.0012(10^6))(s-199060(10^6))}{(s+.034(10^6))(s+.075(10^6))(s+24.9(10^6))(s+1139.5(10^6))}$$

Comparison of this model with eq. (17) shows favorable agreement.

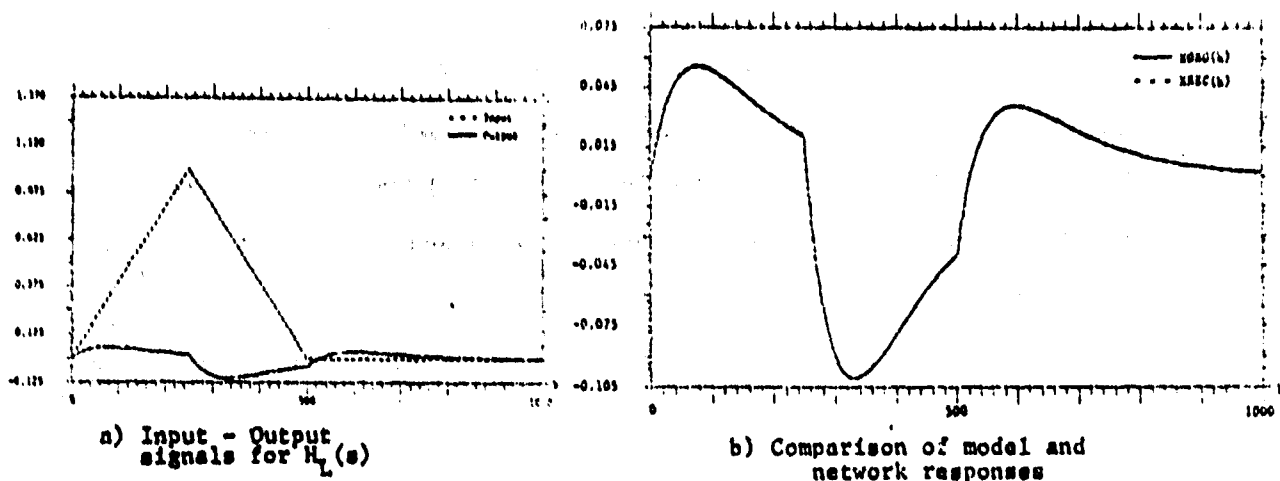


Fig. 4. Second Order Model Identification of $H_L(s)$

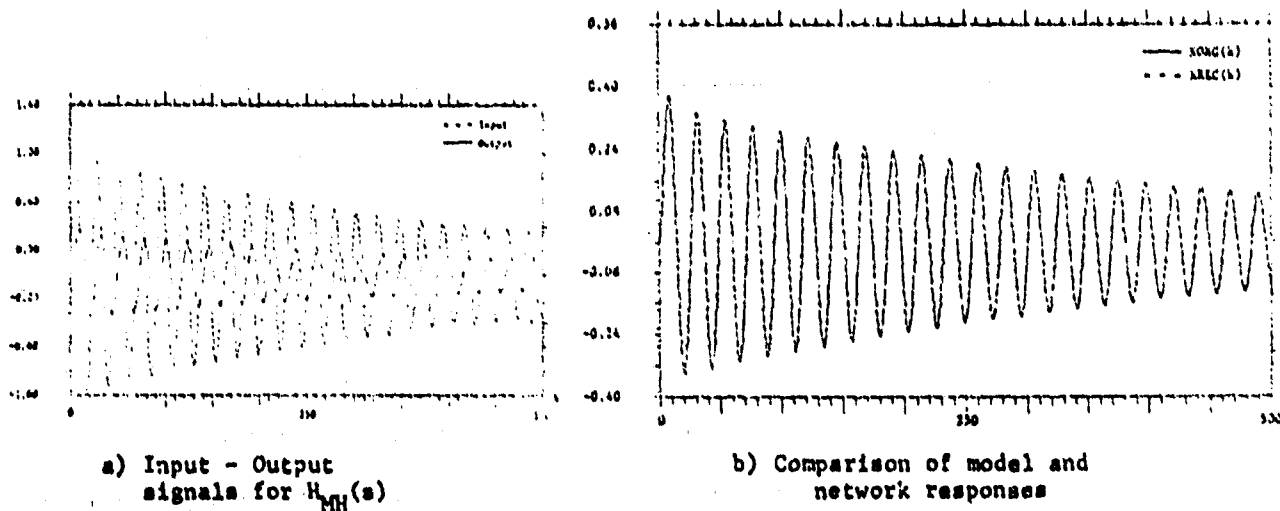


Fig. 5. First Order Model Identification of $H_{MH}(s)$

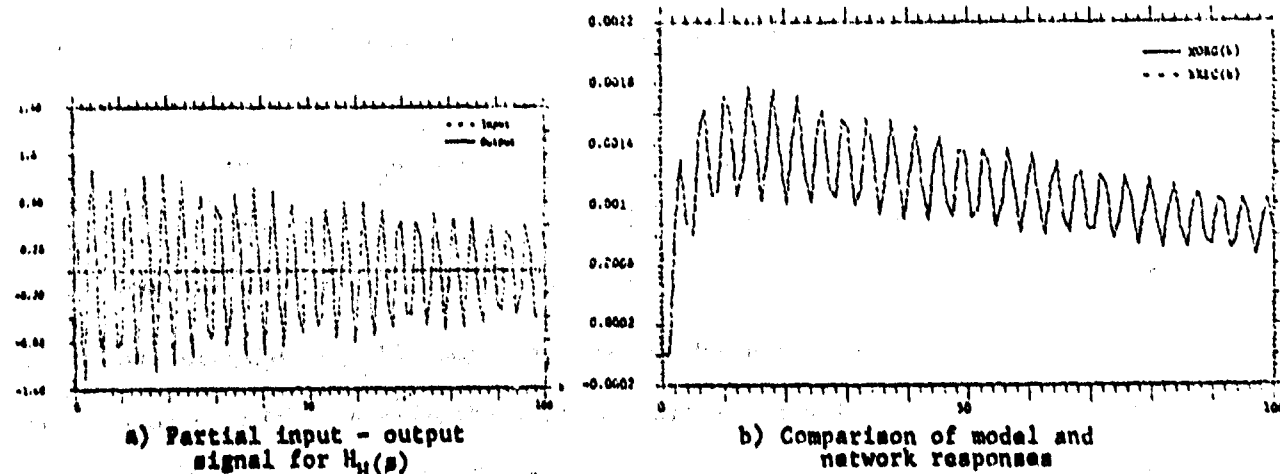


Fig. 6. Second Order Model Identification of $H_H(s)$

2.3 Program Description

This FORTRAN IV program determines a linear model (transfer function) of a network from recorded laboratory responses. The linear model is obtained via the pencil-of-functions method discussed in Section 2.1. The program has certain features which are discussed below.

Network modeling involves the determination of the coefficients α_i and β_i of a rational transfer function of the form

$$H(s) = \frac{\beta_0 + \beta_1 s^1 + \dots + \beta_n s^n}{\alpha_0 + \alpha_1 s^1 + \dots + \alpha_n s^n} \quad (17)$$

such that the output of this model to a given input will approximate the actual network output to the same input. Equivalently*, in discrete time [17] we wish to determine the coefficients a_i and b_i of a function of the form

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_n z^{-n}}{a_0 + a_1 z^{-1} + \dots + a_n z^{-n}} \quad (18)$$

If the network under study is assumed to have direct transmission, the numerator coefficient b_0 is nonzero. This choice of model structure is implemented by setting IREM = 0. When direct transmission cannot be assumed (i.e., it is known on physical grounds that the impulse response of the network will not contain an impulse), then b_0 should be set to zero. This is accomplished with IREM ≠ 0. For example, if IREM=1, the coefficient b_0 in equation (18) is set to zero; for IREM=2 the coefficients b_0 and b_1 are set to zero. It is recommended to use IREM=1 whenever direct transmission cannot be assumed.

All calculations are performed in discrete time; finally $H(z)$ is transformed by means of a pulse invariant transformation*(IZTS=2) to the corresponding continuous time model $H(s)$.

After modeling has been accomplished, the

* See Appendix D

Normalized mean-square error (and its square root) comparing the model and actual network responses are calculated (subroutine ERROR). These errors are calculated as shown below.

$$\text{N.M.S.E.} = \frac{\sum_k [x(k) - x_{\text{model}}(k)]^2}{\sum_k x^2(k)}$$

$$\text{R.N.M.S.E.} = \sqrt{\text{N.M.S.E.}}$$

Another feature of the program is the capability for bias-removal from the recorded laboratory responses (IBIAS=1). This feature allows consideration for bias that may have been introduced through the laboratory measurement system to the recorded output-input data.

Finally, a plot option (IPLT) is available. When IPLT=1, two sets of plots are given. The first shows the original output-input data measured from the network. The second plot contains the original network response and the identified linear model response. This plot allows visual inspection of the closeness of the model fit to the actual (desired) response.

To enable the test engineer to effectively use the program, a description of the input data cards is given below.

INPUT DESCRIPTION

CARD #1 The first card is a title card. Columns 1 through 80 are available for an alpha-numeric title.

CARD #2 Option card containing three variables

Variable Name (Format)	Description	Columns
N(15)	Order of the system	1-5
MP1(15)	Number of data points (output-input data)	6-10
IPLT(15)	Plotter option; IPLT = 0 no plots IPLT = 1 plots on line printer	11-15

CARD #3 through CARD [2+NOUT]

$NOUT = [(MP1+7)/8]$, where $[X]$ is the truncated value of X.

The output data is entered on these cards in 8F10.0 fields.

CARD #[3+NOUT] through CARD [2+NOUT+NIN]

$NIN = [(MP1+7)/8]$, where $[X]$ is the truncated value of X.

The input data is entered on these cards in 8F10.0 fields.

*CARD #[3+NOUT+NIN] Second option card containing six variables.

Variable Name (Format)	Description	Columns
N(15)	Order of the system	1-5
MP1(15)	Number of data points (output-input data)	6-10
ISkip(15)	This variable determines the sequence of points plotted on the printer. If ISkip = 1 every data point is plotted, and if ISkip = 5 every fifth point is plotted, etc.	11-15
IREM(15)	Variable used to specify model structure for the identified system. If direct transmission is assumed, IREM=0. For IREM=m, the first m terms (in ascending order) of the model numerator polynomial are set to zero. It is recommended IREM=1 when direct transmission cannot be assumed.	16-20

*At first glance, this card may seem partially repetitious with CARD #2. However, when multiple identification runs on the same output-input data are desired, then more than one such option card may be placed here, with the option variables changed as desired (for instance, a run on only part of the output-input sequence may at times be needed).

IBIAS(I5)	Bias-removal option	21-25
	IBIAS = 0 no bias is assumed present on the output-input data.	
	IBIAS = 1 bias, assumed to have been introduced by the measurement system, is removed before identification is performed.	
DELTA(F5.0)	Sampling interval	26-30
<u>END OF FILE CARD</u>		

A listing of the FORTRAN programs used is given in Appendix A .

III. COMPUTER ROUTINE FOR HIGH PRECISION INVERSION OF SECOND ORDER VOLTERRA RESIDUE MATRICES

The determination of the residues of the quadratic TF, $H_2(s_1, s_2)$, involves the solution of a set of linear equations. Unfortunately, the number of equations involved are large, for example 12 [2] even for a modest single pole-pair situation (i.e., where the linear TF has two distinct poles). Solution of these equations can lead to computational errors unless extreme caution is exercised in the inversion of the associated matrix. In fact the problem is further aggravated in cases where the system is wide band, i.e., when the poles of $H_1(s)$ are spaced several decades apart. In such situations, the poles of $H_2(s_1, s_2)$ involve sums and differences of the linear TF poles which can result in precariously close values. For example, if

$$\lambda_1 = 50 \text{ radians/s}$$

$$\lambda_2 = 50 \text{ M radians/s}$$

then

$$\lambda_1 + \lambda_2 = 50.00005 \text{ Mradians/s}$$

$$\lambda_1 - \lambda_2 = 49.99995 \text{ Mradians/s}$$

This in turn causes the associated columns of the coefficient matrix corresponding to these poles to be almost scalar multiples of each other. The matrix thus becomes nearly singular, or highly ill-conditioned to invert.

The program presented in this section is designed to deal with such wideband cases, and more generally, to invert ill-conditioned matrices where ever they may arise. It is hoped that by mastering the various capabilities of this routine the analyst can cope with almost all situations of practical interest.

The program possesses the following features which enable high-precision inversion:

- Adaptive Scaling
- Application of Perturbation Theory to Ill-Conditioned Matrices
- Iterative Correction

Before discussing each of these in detail, it is useful to define the term "ill-conditioned" matrix. We will call a matrix ill-conditioned if (a) the rows (or columns) of the matrix are nearly dependent, (b) "small" changes in one or more entries of the matrix result in large changes in its inverse, or (c) the nonzero entries of the matrix differ widely by several orders of magnitude (and remain so even after appropriate scaling has been performed) [18], [19], [20]. Note, the above conditions are not mutually exclusive.

3.1 ADAPTIVE SCALING

In many applications the entries of a matrix differ widely in their respective sizes. For a linear system of equations this situation arises when the values of the (unknown) variables are orders of magnitude different and/or the various equations have right-hand-sides which are orders of magnitude different.

This situation can be remedied in many cases as follows. Denote the matrix of interest as A_0 . Then it is possible to factorize A_0 as

$$A_0 = PA_1 = PAQ \quad (1)$$

where P and Q are suitable diagonal scaling matrices [19]. The following method was developed to obtain the diagonal matrices P and Q , and hence the new matrix, A .

The diagonal entries of matrix P are successively computed from the product of all "significant" terms in the successive rows. The term "significant" can be specified by the user (in the examples presented here any entry greater than 15 orders of magnitude below the largest entry in the row of interest was considered significant); a default value of 15 orders of magnitude is assumed. Then, the P_{ii} entry of the diagonal matrix P is computed as the (n_i) th root of the magnitude of the aforementioned product, where n_i is the number of terms in the product.¹

¹The scaling of the i th row may be stated mathematically as follows; let

$$\alpha_i = \max_j \text{ABS}(A_0)_{ij} \quad [\text{largest entry of } i\text{th row}]$$

$$\beta_i = \alpha_i * 10^{-m} \quad [\text{threshold for } i\text{th row (m chosen by the user)}]$$

$$(A_0)_{ij} : \text{ABS}(A_0)_{ij} > \beta_i \quad [\text{qualifying entries of } i\text{th row}]$$

$$n_i = \text{number of qualifying entries in } i\text{th row}$$

Then

$$P_{ii} = \prod_{\substack{\text{qualifying} \\ \text{entries}}} \left\{ (A_0)_{ij} \right\}^{\frac{1}{n_i}}$$

At this point, we have factorized matrix A_0 into two matrices, i.e., $A_0 = P * A_1$, where the entries of A_1 are specified as follows:

$$(A_1)_{ij} = (A_0)_{ij} / P_{ii}, \quad j = 1, \dots, n \quad (2)$$

It should be noted that in certain cases, the above row scaling will suffice, and further scaling may not be necessary. However, in general, the above process may be repeated, this time utilizing column scaling. Specifically, the column scaling involves factorizing A_1 such that $A_1 = A * Q$, where Q is diagonal. The entries of A are obtained as

$$(A)_{ij} = (A_1)_{ij} / Q_{jj}, \quad i = 1, \dots, n \quad (3)$$

The entries of the scaling matrix Q are chosen in the same manner as those of P , except that columns rather than rows (of A_1) are examined.

Utilizing this technique, the desired inverse is seen to be

$$A_0^{-1} = Q^{-1} A^{-1} P^{-1}. \quad (4)$$

Example 1

$$A_0 = \begin{bmatrix} 0.10000000 \text{ E}+03 & 0.20000000 \text{ E}-04 & 0.29999999 \text{ E}-01 \\ 0.19999989 \text{ E}+06 & 0.40000000 \text{ E}-01 & 0.60000000 \text{ E}+02 \\ -0.10000000 \text{ E}+10 & 0.10000000 \text{ E}+03 & 0.0 \end{bmatrix}$$

Inspection of matrix A_0 shows that its entries differ widely in relative magnitudes; in fact there is a difference of fourteen orders of magnitude.

Therefore, scaling can be used, yielding the following: $A_0 = PA_1 = PAQ$, where

$$P = \begin{bmatrix} 0.391486768\text{E}-01 & 0.0 & 0.0 \\ 0.0 & 0.78297339\text{E}+02 & 0.0 \\ 0.0 & 0.0 & 0.31622776\text{E}+06 \end{bmatrix}$$

Row Scaled Matrix, $A_1 = \begin{bmatrix} 0.25543647\text{E}+04 & 0.51087295\text{E}-03 & 0.76630943\text{E}+00 \\ 0.25543639\text{E}+04 & 0.51087304\text{E}-03 & 0.76630956\text{E}+00 \\ -0.31622776\text{E}+04 & 0.31622776\text{E}-03 & 0.0 \end{bmatrix}$

$$Q = \begin{bmatrix} 0.27427647\text{E}+04 & 0.0 & 0.0 \\ 0.0 & 0.43538684\text{E}-03 & 0.0 \\ 0.0 & 0.0 & 0.76630949\text{E}+00 \end{bmatrix}$$

Row and column scaled matrix,

$$A = \begin{bmatrix} 0.93131018E+00 & 0.11733771E+01 & 0.999999910E+00 \\ 0.93130987E+00 & 0.11733773E+01 & 0.10000001E+01 \\ -0.11529525E+01 & 0.726314477E+00 & 0.0 \end{bmatrix}$$

Now, $A_o^{-1} = Q^{-1} A^{-1} P^{-1}$, and the inversions yield:

$$A_o^{-1} = \begin{bmatrix} 0.20000000E+05 & -0.10000000E+02 & -0.54977761E-18 \\ 0.20000000E+12 & -0.10000000E+09 & 0.99999999E-02 \\ -0.19999996E+09 & 0.10000000E+06 & -0.66666666E-05 \end{bmatrix}$$

The product $A_o (A_o)^{-1}$ is given by

$$A_o (A_o)^{-1} = \begin{bmatrix} 0.10000000E+01 & -0.17053025E-12 & 0.26469779E-22 \\ -0.95367431E-06 & 0.10000000E+01 & 0.54210108E-19 \\ 0.0 & -0.95367431E-06 & 0.10000000E+01 \end{bmatrix}$$

Checking Product of A_{orig} and Its Computed Inverse

After having computed the inverse, it is desirable to check the accuracy of inversion. The obvious way to do so is to obtain the product of A_o and A_o^{-1} and to examine how close it is to the unit matrix. However, in view of the finite word-length of the computer, this product must be computed carefully. Whenever scaling methods are employed, the product of A_o and its inverse may be determined in a number of ways. This will be discussed for three different cases: (i) row scaling, (ii) column scaling, (iii) both row and column scaling.

(i) Row Scaling: $A_o = PA$

In this case, $A_o^{-1} = A^{-1} P^{-1}$. Therefore, as a check on the dependability of A_o^{-1} , the product $A_o A_o^{-1}$ would probably be examined as follows:

$$A_o A_o^{-1} = P(A A^{-1}) P^{-1} \quad (5a)$$

On the other hand,

$$A_o^{-1} A_o = A^{-1} (P^{-1} P) A = A^{-1} A \quad (5b)$$

Equations (5a) and (5b), while representing the same quantity ($A_o A_o^{-1} = A_o^{-1} A_o$), may not be found equal due to the available computer accuracy (finite word-length). Example 2 illustrates this point with an extreme case.

EXAMPLE 2:

$$A_o = \begin{bmatrix} 0.10000000E+03 & 0.20000000E-04 & 0.29999999E-01 \\ 0.19999990E+19 & 0.40000000E+12 & 0.60000000E+15 \\ -0.50000000E-08 & 0.49999999E-11 & 0.0 \end{bmatrix}$$

Factorizing A_o , we obtain $A_o = PA$, where

$$P = \begin{bmatrix} 0.39148676E-01 & 0.0 & 0.0 \\ 0.0 & 0.78297339E+15 & 0.0 \\ 0.0 & 0.0 & 0.15811383E-09 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.25543647E+04 & 0.51087295E-03 & 0.75630943E+00 \\ 0.25543639E+04 & 0.51087304E-03 & 0.76630956E+00 \\ -0.31622776E+02 & 0.31622776E-01 & 0.0 \end{bmatrix}$$

Now, matrix A is inverted, yielding

$$A^{-1} = \begin{bmatrix} 0.78297352E+03 & -0.78297339E+03 & -0.22097009E-14 \\ 0.78297352E+06 & -0.78297339E+06 & 0.31622776E+02 \\ -0.26104324E+07 & 0.26104333E+07 & -0.210818510E-01 \end{bmatrix}$$

The product AA^{-1} is computed as

$$\begin{bmatrix} 0.99999999E+00 & 0.23283064E-09 & -0.43368086E-17 \\ -0.23283064E-09 & 0.10000000E+01 & -0.43368086E-17 \\ -0.45475735E-11 & 0.27284841E-11 & 0.99999999E+00 \end{bmatrix}$$

The required inverse, A_o^{-1} , is computed as $A_o^{-1} = A^{-1}P^{-1}$. Once A_o^{-1} has been computed, we can obtain the product of A_o and A_o^{-1} by use of equation (5a) or (5b). Utilizing Eq. (5a), yields

$$\Lambda_o \Lambda_o^{-1} = P(AA^{-1})P^{-1} = \begin{bmatrix} 0.99999999E+00 & 0.0 & -0.21475700E-09 \\ 0.0 & 0.99999999E+00 & -0.42951393E+07 \\ 0.0 & 0.0 & 0.99999999E+00 \end{bmatrix}$$

Comparison of this matrix with the unit matrix (the (2,3) entry in particular) would yield the faulty conclusion that an accurate inverse has not been obtained. If, on the other hand, Eq (5b) is used as a check, we find that

$$\Lambda_o^{-1} \Lambda_o = A^{-1}(P^{-1}P)A = A^{-1}A = \begin{bmatrix} 0.99999999E+00 & -0.69876856E-16 & -0.17053025E-12 \\ -0.29795324E-06 & 0.99999999E+00 & -0.10186340E-09 \\ 0.25428985E-05 & 0.36878730E-12 & 0.10000000E+01 \end{bmatrix}$$

Comparison of this matrix with the unit matrix would be favorable. Thus, in the case of row scaling, equation (5b) must be used as a check on the product Λ_o and Λ_o^{-1} due to the finite computer word-length. Use of this equation avoids the possible problem (present in Eq. 5a) that the matrices P , (AA^{-1}) and P^{-1} may not be compatible for multiplication, as exemplified above. To reiterate, in the case of row-scaling the product of Λ_o and Λ_o^{-1} must be computed as $\Lambda_o^{-1} \Lambda_o = A^{-1}A$, where A is the scaled matrix.

ii) Column Scaling: In a similar manner we will show that the product Λ_o^{-1} must be used as a check on the goodness of Λ_o^{-1} in the case of column scaling. This can be seen from a comparison of the following equations, where $\Lambda_o = AQ$ and $\Lambda_o^{-1} = Q^{-1}A$.

$$\Lambda_o^{-1} \Lambda_o = Q^{-1}(A^{-1}A)Q \quad (6a)$$

$$\Lambda_o \Lambda_o^{-1} = A(QQ^{-1})A^{-1} = AA^{-1} \quad (6b)$$

Again due to finite word length in the computer, equation (6b) should be used to verify the goodness of the inverse when column-scaling is performed.

iii) Row and Column Scaling: In this case, matrix Λ_o is factorized as $\Lambda_o = PAQ$, with the required inverse, $\Lambda_o^{-1} = Q^{-1}A^{-1}P^{-1}$. The decision to use $\Lambda_o \Lambda_o^{-1}$ or $\Lambda_o^{-1} \Lambda_o$ as a check toward the accuracy of the inverse obtained may be arrived at as follows. The equations representing $\Lambda_o \Lambda_o^{-1}$ and $\Lambda_o^{-1} \Lambda_o$ are

$$A_o A_o^{-1} = P(A(QQ^{-1})A^{-1}P^{-1}) = P(AA^{-1})P^{-1} \quad (7a)$$

and

$$A_o^{-1} A_o = Q^{-1} A^{-1} (P^{-1} P) A Q = Q^{-1} (A^{-1} A) Q \quad (7b)$$

The difference in magnitude between the largest and smallest entries in the diagonal matrices P and Q is calculated. These two differences are compared, and the matrix with the largest difference is attempted to be eliminated (either P or Q). Thus, if the entries of P are more incompatible for multiplication than those of Q, equation (7b) would be used as a check rather than equation (7a). An example will help to clarify this procedure.

EXAMPLE 3:

$$A_o = \begin{bmatrix} 0.10000000E+03 & 0.20000000E-04 & 0.29999999E-01 \\ 0.19999990E+19 & 0.40000000E+12 & 0.60000000E+15 \\ -0.50000000E-08 & 0.49999999E-11 & 0.0 \end{bmatrix}$$

The matrix is now expressed as $A_o = PAQ$, where

$$P = \begin{bmatrix} 0.39148676E-01 & 0 & 0.0 \\ 0.0 & 0.78297339E+15 & 0.0 \\ 0.0 & 0.0 & 0.15811388E-09 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.59091075E+03 & 0.0 & 0.0 \\ 0.0 & 0.20208866E-02 & 0.0 \\ 0.0 & 0.0 & 0.76630949E+00 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.43227589E+01 & 0.25279643E+00 & 0.99999991E+00 \\ 0.43227575E+01 & 0.25279647E+00 & 0.10000000E+01 \\ -0.53515317E-01 & 0.15647971E+02 & 0.0 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 0.46266748E+06 & -0.46266740E+06 & -0.13086035E-11 \\ 0.15823007E+04 & -0.15823005E+04 & 0.63906048E-01 \\ -0.20003991E+07 & 0.20003998E+07 & -0.16155222E-01 \end{bmatrix}$$

Comparing the magnitude difference between the largest and smallest entries of P and Q, it is found that in P, the difference is approximately 10^{24} , while in Q the difference is approximately 10^5 . Therefore, Eq. (7b) should be used, in order that multiplication involving P and P^{-1} is avoided. Performing the required multiplication indicated in Eq. (7b), we obtain

$$A_o^{-1} A_o = Q^{-1} (A^{-1} A) Q = \begin{bmatrix} 0.99999999E+00 & -0.70030334E-16 & -0.56613984E-13 \\ -0.23651533E-06 & 0.99999999E+00 & -0.43109444E-10 \\ 0.89538983E-06 & 0.15661482E-12 & 0.10000000E+01 \end{bmatrix}$$

3.2 PERTURBATION

In some applications, the given matrix may be ill-conditioned to the point that the scaling method described above will not allow inversion of the matrix to the desired precision. In this case, application of perturbation theory to the scaled matrix, A, may be helpful. Several methods will be described.

A. Diagonal Perturbation

The first method consists of forming a new matrix [18]

$$C = A + \epsilon D \quad (8)$$

where A is the scaled version of the original matrix and D is a diagonal matrix

whose entries can be taken as those of the diagonal of A. The multiplier, ϵ , is chosen to be suitably small (more will be said about this choice later) such that C is invertible (allowing for the available computer accuracy). We can now write $A = C - \epsilon D$ so that we have parameterized A in terms of the small parameter, ϵ . Since $A = A(\epsilon)$ is an analytic function of ϵ , its inverse is also an analytic function of ϵ ; therefore, a power-series may be written. Indeed,

$$\begin{aligned} A^{-1} &= (C - \epsilon D)^{-1} \\ &= (1 - \epsilon C^{-1} D)^{-1} C^{-1} \\ &= [C^{-1} + \epsilon C^{-1} D C^{-1} + \epsilon^2 (C^{-1} D)^2 C^{-1} + \dots] \end{aligned} \quad (9)$$

Thus, using C^{-1} , D, and ϵ , the inverse of A may be computed. An advantage of this method is that it can be performed without any visual inspection of the given matrix. This is so because the diagonal matrix D is specified automatically. A disadvantage worth noting is that the required inverse, A^{-1} , is represented as an infinite series of matrices and cannot be expressed in closed form. Thus, an exact representation of A^{-1} cannot be obtained, although it can be approximated to the required accuracy by computing and summing enough terms of the series.

We now return to the matter of choosing a suitable value of ϵ . Clearly, the smaller the values of ϵ , the faster will be the convergence of the series (9). On the other hand, ϵ must be large enough so that it results in adequate perturbation, i.e., such that C is invertible with the available computer accuracy. It is useful to note that a theoretical upper bound on ϵ can be obtained; indeed for the series (9) to converge it is necessary that $|\epsilon|$ be less than $1.0/|\text{largest eigenvalue of } C^{-1} D|$. [19], [20].

B. Single-entry Perturbation

An alternate perturbation method consists again of forming a new matrix

$$C = A + \epsilon D$$

where we now restrict the perturbation matrix D to have

only one non-zero entry, again on the diagonal. Without loss of generality, this non-zero entry can be taken to be unity. Then, the matrix D has rank-one, trace-one and hence can be written as

$$D = XY^T \quad (10)$$

Thus

$$\begin{aligned} A^{-1} &= (C - \epsilon XY^T)^{-1} \\ &= [1 - \epsilon (C^{-1}X)Y^T]^{-1} C^{-1} \\ &= 1 + \epsilon C^{-1}X[1 + \epsilon Y^T C^{-1}X + (\epsilon Y^T C^{-1}X)^2 + \dots](Y^T C^{-1}) \end{aligned} \quad (11)$$

Now, the quantity in brackets in equation (11) is a power-series involving scalar terms. Therefore, equation (11) can be expressed as

$$A^{-1} = 1 + \epsilon C^{-1}X \left[\frac{1}{1 - \epsilon Y^T C^{-1}X} \right] (Y^T C^{-1}) \quad (12)$$

It can be shown that the series converges for all ϵ . It is, however, best to choose $\epsilon < \frac{1}{Y^T C^{-1}X}$. Thus, equation (12) represents an exact solution for A^{-1} . A disadvantage of this method is that visual inspection of the matrix, A , is necessary to determine the row(s) and/or column(s) causing difficulty in the inversion process, and thus a suitable entry to perturb. An illustrative example of this method is given below.

EXAMPLE 4:

$$A_0 = \begin{bmatrix} 0.10000000E+03 & 0.20000000E-04 & 0.29999999E-01 \\ 0.19999989E+06 & 0.40000000E-01 & 0.60000000E+02 \\ -0.10000000E+10 & 0.10000000E+03 & 0.0 \end{bmatrix}$$

A visual inspection of this matrix shows that the first and second rows are nearly identical to within a multiplicative factor of $\sim 10^3$. Therefore, suitable diagonal entries for perturbation would be either the (1,1) or (2,2) entries. The (2,2) entry was selected for perturbation, and through iteration, a suitable value of ϵ was found to be $\epsilon = 1. \times 10^{-10}$. Scaling was first performed, and then the perturbation yielding:

$$A = C - \epsilon D = \begin{bmatrix} 0.93131018E+00 & 0.11733771E+01 & 0.99999991E+00 \\ 0.93130987E+00 & 0.11733773E+01 & 0.10000000E+01 \\ -0.11529525E+01 & 0.72631447E+00 & 0.0 \end{bmatrix} - 10^{-10} \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.173377 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Note that D can be expressed as $D = XY^T$ so that

$$D = \begin{bmatrix} 0.0 & 1.173377 & 0.0 \end{bmatrix} \begin{bmatrix} 0.0 & 1.0 & 0.0 \end{bmatrix}$$

The inverse of the original matrix can now be computed as discussed:

$$A_o^{-1} = \begin{bmatrix} 0.20000000E+05 & -0.10000000E+02 & 0.52504835E-18 \\ 0.20000000E+12 & -0.10000000E+09 & 0.10000000E-01 \\ -0.19999996E+09 & 0.10000000E+06 & -0.66666666E-05 \end{bmatrix}$$

This yields the product

$$A_o A_o^{-1} = \begin{bmatrix} 0.10000000E+01 & -0.51159076E-12 & 0.66174449E-22 \\ -0.95367431E-06 & 0.10000000E+01 & 0.0 \\ 0.0 & -0.95367431E-06 & 0.99999999E+00 \end{bmatrix}$$

C. PERTURBATION: THE LIMITING CASE: In the previous discussions on perturbation, it was shown that matrix A is a function of matrix C and the scalar quantity, ϵ . That is $A = (C(\epsilon), \epsilon)$. Recall also that whereas A was ill-conditioned (for inversion), there were certain values of ϵ for which the newly formed matrix C could be made well-behaved. Inspection of equation (8) shows that

$$A^{-1} = \lim_{\epsilon \rightarrow 0} C^{-1} \quad (13)$$

This observation can be exploited in the following way. Successively small values of ϵ , say ϵ_1 , are used to form a family of $C = C(\epsilon_1)$ matrices. The inverse of $C(\epsilon_1)$ is computed for each value of ϵ_1 used, and the successive inverses are examined. There will exist a region wherein reducing the value of ϵ from ϵ_1 to ϵ_{i+1} will have little effect on the entries of C^{-1} . This is shown graphically in Figure 1.

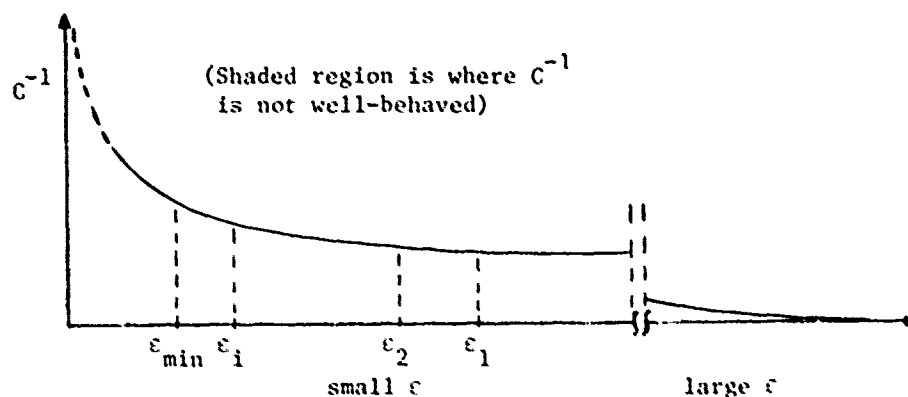


Fig. 1. Effect of small changes in ϵ on C^{-1} .

In this region, C^{-1} can be taken as an approximation to A^{-1} . As seen in Figure 1, there will exist some $\epsilon_{\min}$ for which the inverse of C is well-behaved. The closer the selected value of ϵ is to this value of $\epsilon_{\min}$, the better will become the approximation $A^{-1} \approx C^{-1}$. Although this method may only yield an approximation to the actual required inverse, A^{-1} , it may be further refined by use of the method described in the next section. In fact, the iterative correction method (of the next section) may be used in conjunction with all of the methods discussed earlier.

3.3 ITERATIVE CORRECTION

Consider a matrix, X , and assume that its inverse has been computed as $Y \approx X^{-1}$. The iterative correction method (see Fig. 2) consists of forming the product, XY , comparing it with the identity matrix, and improving the computed inverse, Y , by an amount proportional to the error between XY and the unit matrix. To examine the effect of this operation, let

$$Y = X^{-1}(I + E) \quad (14)$$

where I = identity matrix and E is equal to the difference matrix between XY and I .

$$XY = (I + E) \quad (15)$$

$$E = (XY - I) \quad (16)$$

Now consider the iteration $Y_{\text{improved}} = Y - YE$ [12]. Clearly (17a)

$$\begin{aligned} Y_{\text{improved}} &= Y - YE \\ &= X^{-1}(I + E)(I - E) \\ &= X^{-1}(I - E^2) \end{aligned} \quad (17b)$$

Upon the second iteration,

$$Y_{\text{improved}} = X^{-1}(I + E^4)$$

and so on. This procedure can be depicted in block-diagram form as in Figure 2.

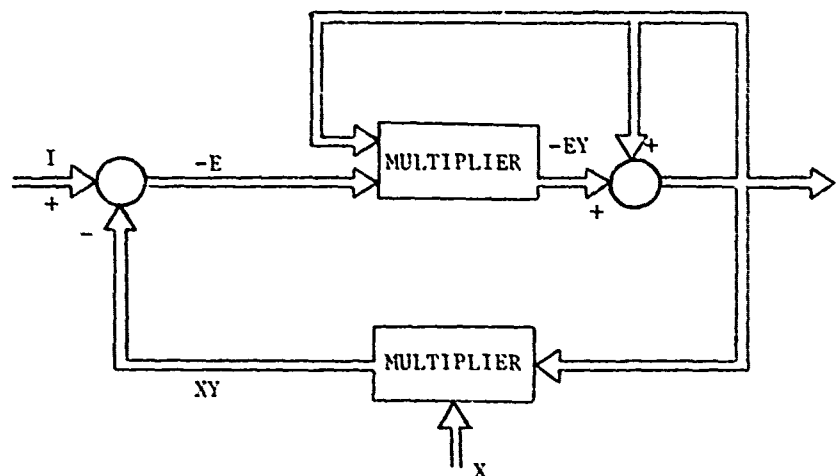


Fig. 2: Block diagram representation of iterative correction method.

The number of iterations to be used may be specified by the user. For this work, n iterations have usually been used, where n denotes the dimension of the matrix in question. Note that a more general version of (17a) is

$$Y_{\text{improved}} = Y - \beta * YE \text{ where } \beta \text{ is a suitable positive fraction.}$$

EXAMPLE 5: (Effect of iterative correction)

$$A_0 = \begin{bmatrix} 0.10000000E+03 & 0.20000000E-04 & 0.29999999E-01 \\ 0.19999999E+06 & 0.40000000E-01 & 0.60000000E+02 \\ -0.10000000E+10 & 0.10000000E+03 & 0.0 \end{bmatrix}$$

Assume that the inverse matrix has been computed as

$$\tilde{A}_o^{-1} = \begin{bmatrix} 0.19999999E+06 & -0.99999998E+02 & -0.10058593E-16 \\ 0.19999999E+13 & -0.99999998E+09 & 0.99999998E-02 \\ -0.19999999E+10 & 0.99999998E+06 & -0.66666665E-05 \end{bmatrix}$$

so that

$$A_o \tilde{A}_o^{-1} = \begin{bmatrix} 0.99999999E+00 & 0.36379788E-11 & 0.52939549E-22 \\ -0.15258789E-04 & 0.99999999E+00 & 0.0 \\ -0.11718750E-01 & 0.15258789E-04 & 0.99999999E+00 \end{bmatrix}$$

Now, if 1 iteration of the correction method is performed,

$$A_o \tilde{A}_o^{-1} = \begin{bmatrix} 0.10000000E+01 & 0.0 & 0.0 \\ 0.0 & 0.10000000E+01 & -0.54210108E-19 \\ -0.15625000E-01 & 0.0 & 0.99999999E+00 \end{bmatrix}$$

with N(=3) iterations performed,

$$A_o \tilde{A}_o^{-1} = \begin{bmatrix} 0.99999999E+00 & 0.0 & -0.13234889E-22 \\ 0.0 & 0.10000000E+01 & 0.54210108E-19 \\ -0.78125000E-02 & 0.0 & 0.99999999E+00 \end{bmatrix}$$

It can be seen that the product $A_o \tilde{A}_o^{-1}$ is approaching the unit matrix. The worst entry, (3,1), in the product has been reduced to about 60% of its original value in the 3 iterations.

The improved inverse has been computed as

$$\tilde{A}_o^{-1} = \begin{bmatrix} 0.19999999E+06 & -0.99999999E+02 & -0.76657257E-17 \\ 0.19999999E+13 & -0.99999999E+09 & 0.99999999E-02 \\ -0.19999999E+10 & 0.99999999E+06 & -0.66666665E-05 \end{bmatrix}$$

This example has demonstrated the usefulness of the correction method in improving the calculated inverse.

3.4 APPLICATION

Determination of the kernels of the volterra model can be accomplished by identifying the various link transfer functions $H_1(s_1)$, $H_2(s_1, s_2)$, etc. The first step, of course, is to determine $H_1(s_1)$ for the small signal linear case. Next, identification of $H_2(s_1, s_2)$ is attempted. However, it is shown in [1] that the poles of $H_2(s_1, s_2)$ are given as $\gamma_{ik} = \lambda_i + \lambda_k$ where λ_i , λ_k are the poles of the linear transfer function $H_1(s_1)$. The residues at these poles can then be determined by solving a set of linear equations from data obtained for larger amplitudes where quadratic effects become nonnegligible.

The set of equations which must be solved (for computing residues of the response) may form a nearly dependent set if the network is a wideband (i.e., its poles are separated by two or more orders of magnitude) network. Accurate inversion of the corresponding matrix can be quite a formidable task in this case.

In this section, a 12×12 matrix generated in the analysis of a wideband network is examined. The equation of interest is

$$R = (A_0)^{-1}Y$$

where R = column vector of residues of poles of the system response

Y = column vector of integrated system outputs at time T

A_0 = 12×12 matrix whose entries are generated by

$$(A_0)_{ij} = e^{\frac{\lambda_j T}{\lambda_i}} - \sum_{m=1}^i \left(\frac{T^{m-1}}{(m-1)!} \right) \frac{1}{\lambda_j^{i+1-m}}$$

The linear portion of the system examined has two poles. These poles are

$$\lambda_1 = -0.011550998(2\pi)(10^6) \text{ rad/sec}$$

$$\lambda_2 = -10.616986 (2\pi)(10^6) \text{ rad/sec}$$

The system is excited by the input function

$$X_{in}(t) = [e^{\alpha_1 t} + e^{\alpha_2 t}] U(t)$$

where

$$\alpha_1 = -10^7 \text{ rad/sec}$$

$$\alpha_2 = -1.75 \times 10^7 \text{ rad/sec}$$

The poles of the quadratic response of this network are found in terms of the input and linear transfer function poles as

$$\gamma_1 = \lambda_1$$

$$\gamma_2 = \lambda_2$$

$$\gamma_3 = 2\lambda_1$$

$$\gamma_4 = 2\lambda_2$$

$$\gamma_5 = \lambda_1 + \lambda_2$$

$$\gamma_6 = \alpha_1 + \lambda_1$$

$$\gamma_7 = \alpha_1 + \lambda_2$$

$$\gamma_8 = \alpha_2 + \lambda_1$$

$$\gamma_9 = \alpha_2 + \lambda_2$$

$$\gamma_{10} = 2\alpha_1$$

$$\gamma_{11} = 2\alpha_2$$

$$\gamma_{12} = \alpha_1 + \alpha_2$$

The matrix A_0 was generated for $T = 6.0$ nanoseconds. The difficulty in inverting this matrix arises because $\gamma_2 \approx \gamma_5$ and consequently

$$C(I,2) \approx C(I,5)$$

for all I , thereby making accurate inversion by standard routines impractical.

The matrix A_0 is shown below.

ORIGINAL MATRIX, λ_0

0.567175/21272173.70+00 0.140743165943663211-01 0.1107373534.30435700-11	0.150116.0.1257917760-11 0.0004335056732207700-02 0.4900369278900233300-01	0.1746145570.045003700+00 0.1503537614027274.00+01 0.70571420.447.411300-01	0.7495304106053535820-02 0.560353405652516330-01 0.36363332315387720-01
0.1774154201471671900+00 0.37003507767020.75070-12 0.09071047.342005070-02	0.0709640001734112000-02 0.47346551336090700-11 0.2787001536353003300-01	0.1746145570.045003700+00 0.70571420.447.411300-01 0.10320551012203.000-01	0.4441002000127035110-02 0.500350057020725500-01 0.204353670505005430-01
0.356114720715252340-01 0.250410579405054370-02 0.205404213700132700-02	0.25600714202634300-12 0.123306700230270170-11 0.702400251073455500-02	0.55220.0.535204107300-01 0.2245094037.055002700-01 0.4070004050032460300-02	0.13150007200.069000-02 0.3404000321677310.50-02 0.30001502500.0147400-02
0.555309320352001010-02 0.50007330.00307700-13 0.4031114715035621250-03	0.5011032950051550010-03 0.7200111753453102120-02 0.1110750933401527100-02	0.5307200720104107300-02 0.4470104771235007300-03 0.3043004331533002710-03	0.253900110722331600-03 0.1505021511403750.00-02 0.1100170350.413521320-02
0.0435201.11122304070-03 0.73030040365005400-04 0.59330557250030340-04	0.734362413170233000-04 0.5387400241310607300-03 0.19000240079933500-03	0.353703702210461030-03 0.540501717445530.00-04 0.128715416135610340-03	0.35526102003134500-04 0.2101010193703200.10-03 0.150429350750593250-03
0.643100990030244300-04 0.00047814.0161900510-05 0.6990510014027072500-05	0.0613000190557030120-05 0.33000752.0220134330-04 0.2244007500500331910-04	0.5470240010001007310-04 0.7640574170000000-00 0.1403070730.41000400-04	0.450010220732600.00-05 0.244507051004307000-04 0.170752745179413400-04
0.5526545035324397710-05 0.041040913213320220-06 0.60600405031216700200-06	0.0422704013.71201000-06 0.300300240.2207030400-15 0.7117550245190034100-05	0.5430.0047450300471500-05 0.745002060.042070700-10 0.142752700.31117090-05	0.451455710250307050-06 0.2295074353900774.10-05 0.17063550530.3023.50-05
0.502321407074034370-06 0.705700003002305070-07 0.570012301104503900-07	0.706350055740067070-07 0.7447.03300085704130-06 0.1710314734542940050-06	0.4125343200574010070-06 0.3000.70710050.043010-07 0.1170075121067027.020-06	0.300247202657002700-07 0.1054072702.00400700-00 0.1330240021000970.00-00
0.4702271010034070300-06 0.5101120505522044540-03 0.42000.1320322052590-03	0.5105704102025330700-10 0.170030513270007100-07 0.1223691775505672020-07	0.270100511070004610-07 0.401330500030201510-06 0.553325475.0435.7300 00	0.2035654074001.000-00 0.1310110000.000.00.00-07 0.100590773227532040-07
0.6070327052.055000.00-05 0.5307744256007400040-03 0.2791990103100019200-03	0.3305725451007030460-03 0.10031202301434.0200-06 0.770771541555.130040-09	0.265057034000000000-10 0.101.1.250057072030-09 0.5400.02100000000790-03	0.100012004050.11.00-03 0.03100207070.000400-03 0.64405004090402.02200-03
0.0376309700001103300-04 0.190000142409150000-10 0.164720551720037550-10	0.190000142409150000-10 0.5100191350247427300-17 0.444702006753000000-10	0.131001000032007500-07 0.1770000110.0450.0720-10 0.519.00010100.000720-10	0.110034367611101.020-10 0.474000020040011.00-10 0.31719747300007.000-10
-0.1154137100300000000-02 0.100521017157000130-11 0.000315020001227000-12	0.1004100070005117410-11 0.59707004400700.31410-11 0.224.00015064772150-11	-0.003470000000000000-07 0.3070000000000000.1200-12 0.1000.2700.017102.0000-11	0.50001243302231003110-12 0.217.00400702100.000-11 0.190332154007040270-11

It is readily apparent, through visual inspection, that row scaling is necessary, yielding

(R00) SCALED MATRIX A

0.1416431146821427270+02	0.3616471647067210530+01	0.13362627055124400+02	0.1500236673533605100+00
0.3612541293349396150+00	0.2389424725817919340+01	0.51460114000143211+00+00	0.1372530731200200040+01
0.2864904060149400020+00	0.1206235391202413270+01	0.0002055102075039070+00	0.0772096147000000000+00
0.0309270369363113350+01	0.414680/177622143100+00	0.020000767444113150+01	0.200000767444113150+01
0.4142425007536139160+00	0.2351754055749047340+01	0.351020011000000000+00	0.3401412003477312970+01
0.3302520907535742520+00	0.133036036035501340+01	0.772310014002714300+00	0.360160005030004300+00
0.6201721234963196440+01	0.4515562121103282280+00	0.011754714050104410+01	0.2314550070520010030+00
0.4510000267977265250+00	0.227503301749222000+01	0.3352533000000000+00	0.140257513213012110+01
0.361444901217097700+00	0.1341373557642047210+01	0.022663176001604710+00	0.102055444107781000+01
0.5136117845461637050+01	0.4009495071351100220+00	0.503190136737143230+01	0.240420055555072000+00
0.4503050642307697060+00	0.213713001000040430+01	0.4221672527510000+00	0.150229307333354450+01
0.3507566501540779260+00	0.1351100304485376270+01	0.0500000000000000+00	0.105362175600412300+01
0.442837194005190000+01	0.5055025474340471550+00	0.43000013000003300+01	0.285193514635001000+00
0.5050052251435563070+00	0.2125204582332799710+01	0.44500190000014345+00	0.150193023010170000+01
0.408466998641043550+00	0.137025302633757340+01	0.0000000000000000+00	0.107679446206236650+01
0.3939783971325955920+01	0.5209202029035163570+00	0.3915537973494069760+01	0.27947201720173047730+00
0.5264217281450560320+00	0.206001426515002430+01	0.40023510112350730+00	0.140035277101064150+01
0.4276045443061770540+00	0.13732507719933400+01	0.3976773340000000+00	0.109357023740001300+01
0.3501386424337355020+01	0.5459140003943012120+00	0.3551154331291672470+01	0.2926073985586640070+00
0.5454017423332287090+00	0.207213594000256000+01	0.4032077300010055520+00	0.143792529251341280+01
0.4443524905065001170+00	0.137254139643153130+01	0.325237530475120510+00	0.1100959425432832270+01
0.3063122473933206440+01	0.5658516635663800700+00	0.33105003000100310+01	0.3060335092250704410+00
0.5003307159445734200+00	0.1963732120570210360+01	0.5030600000000000+00	0.148013065015051340+01
0.4630947099460392670+00	0.137034436922102730+01	0.943205000001101300+00	0.112269392754260400+01
0.4140263395000650420+02	0.4565017074001610740+00	0.24402770000017760+01	0.2436211000942210070+00
0.456091156076459000+00	0.1502052168222105500+01	0.4061117330000000+00	0.1157001560295250770+01
0.3759442881391970370+00	0.1077216128210349020+01	0.7511776001005212100+00	0.185565425493152300+00
-0.5260200374007620320+04	0.340650702154004330+00	-0.2715051765020442310+00	0.192464673062100850+00
0.340343797000100000+00	0.1034756303353771760+01	0.3110037000000000+00	0.0506040082040584110+00
0.2075108186513397330+00	0.7998421132134452020+00	0.5560230000000000+00	0.06327354205947500+00
0.5359511529657197190+05	0.1273400542507463330+00	0.041324340000017400+02	0.709478337607121330+01
0.127230210700944300+00	0.503146151352301300+00	0.1130056220000000+00	0.3030152220059330210+00
0.105334106630272020+00	0.20450753625335300+09	0.204139335021724520+00	0.257839745106370000+00
-0.5859033152242720440+03	0.5402004062356765240+01	-0.4535770000000000+00	0.303640726001463500+01
0.5397438173493613970+01	0.1562713185615004290+00	0.4037001120000000+00	0.125392350624998300+00
0.448072233100154320+01	0.1170015731056520540+00	0.1555400000000000+00	0.291013086267684710+01

Program Steps Used:

The row scaling indicated on the earlier pages was achieved by choosing (see program description on page 39)

ISP = 1

ISQ = 0

As mentioned earlier, the second and fifth columns of the original matrix A_0 form a linearly dependent set. Therefore, a perturbation procedure is needed to obtain the inverse. We used

IPERT = 2

so that the 2,2 entry of the scaled matrix A was perturbed. The amount of perturbation used was FAC = 0.1E-05. The inverse of the perturbed matrix C was obtained by the high precision routine GKRDCT. From C^{-1} , an estimate of A^{-1} was obtained by use of the deperturbation routine DPERT1. This approximate inverse was then refined by use of the routine CORCT2 via the option parameter

ICORCT = 1

The resulting inverse matrix A^{-1} was then descaled. The inverse of the original matrix, A_0^{-1} , as well as the product $A_0^{-1}A_0$ are printed on the succeeding pages.

(Note that a preconditioning of the original matrix was employed by setting ICOND = 1. This caused diagonal entries of A_0 to be multiplied by $1.000000001 = 1 + 1.0E-09$.,

INVERSE OF ORIGINAL MATRIX, A_0

0.11370430704403510-07	-0.2555425600503407670-06	0.4770140106210433300-05	-0.55703517946636451930-04
-0.16123460550111054450-03	0.4810947574331070610-01	-1.141091211075150430-01	0.552500050133404370-02
-0.4277730111057997500-03	0.5239445713277060140-01	-0.1170475527917904700-01	-0.1005506572340741130-04
-0.7906037625136143070-03	0.7026802737023613250-01	-0.000550000000000000-01	0.11302549028500000000-01
-0.13211001303042700-01	0.10258055456230200-01	-0.751100000000000000-01	-0.30125978747025000000-01
0.3056754203064000000-01	-0.15905503921192010-01	-0.510751057100000000-01	-0.157753711025513510-01
-0.1557012201001430700-01	0.33070102602793000-02	-0.7150350973104000-01	0.752000910000000000-01
0.207011610115054700-01	-0.707051049420572000-01	0.270000000000000000-01	-0.415100000000000000-01
0.544646461050303000-01	-0.4130002759907001000-01	0.140000000000000000-01	0.105040195700000000-01
-0.646020071074030500-01	0.176300170300000000-01	-0.207001000000000000-01	0.204000000000000000-01
0.304005703300000000-01	-0.122000000000000000-01	0.500000000000000000-01	-0.300000000000000000-01
-0.672000000000000000-01	0.536100464400000000-01	0.150000000000000000-01	0.600000000000000000-01
0.870702767755100150-02	-0.31734518241001500-01	0.750000000000000000-01	-0.120000000000000000-01
0.132100323500000000-01	-0.600000000000000000-01	-0.500000000000000000-01	0.123000000000000000-01
-0.103000000000000000-01	0.337101000000000000-01	0.770000000000000000-01	0.300000000000000000-01
0.161520000000000000-01	-0.200000000000000000-01	0.550000000000000000-01	-0.130000000000000000-01
-0.205717750000000000-01	0.000000000000000000-01	0.170000000000000000-01	-0.300000000000000000-01
0.775000000000000000-01	-0.277297621500000000-01	-0.500000000000000000-01	-0.300000000000000000-01
-0.590201300000000000-01	0.103000000000000000-01	-0.500000000000000000-01	0.370000000000000000-01
0.265205321000000000-01	-0.937377130000000000-01	0.100000000000000000-01	-0.200000000000000000-01
0.130975169170000000-01	-0.367400150000000000-01	-0.500000000000000000-01	-0.420000000000000000-01
-0.165000000000000000-01	0.333000000000000000-01	-0.500000000000000000-01	0.200000000000000000-01
0.293427252000000000-01	0.400000000000000000-01	-0.500000000000000000-01	0.150000000000000000-01
-0.119000000000000000-01	0.425000000000000000-01	0.500000000000000000-01	0.400000000000000000-01
0.549950500000000000-01	-0.101355000000000000-01	0.200000000000000000-01	-0.200000000000000000-01
-0.297200000000000000-01	0.057000000000000000-01	-0.100000000000000000-01	0.120000000000000000-01
-0.710900000000000000-01	0.100000000000000000-01	0.400000000000000000-01	0.200000000000000000-01
0.375000000000000000-01	-0.700000000000000000-01	0.100000000000000000-01	-0.500000000000000000-01
-0.654000000000000000-01	-0.150000000000000000-01	0.100000000000000000-01	-0.300000000000000000-01
0.301400000000000000-01	-0.107220000000000000-01	-0.200000000000000000-01	-0.100000000000000000-01
0.677000000000000000-01	-0.100000000000000000-01	0.100000000000000000-01	0.100000000000000000-01
-0.530000000000000000-01	-0.100000000000000000-01	-0.100000000000000000-01	-0.100000000000000000-01
0.903100000000000000-01	-0.100000000000000000-01	-0.100000000000000000-01	-0.100000000000000000-01
-0.663000000000000000-01	0.241000000000000000-01	-0.100000000000000000-01	0.100000000000000000-01
0.994000000000000000-01	0.700000000000000000-01	-0.100000000000000000-01	0.100000000000000000-01
-0.720000000000000000-01	0.700000000000000000-01	0.100000000000000000-01	0.100000000000000000-01

[illegible]

(CAT-1): RNSL = 0.61370-04 RNX = 0.42230-03

3.5 Program Description

This FORTRAN IV program performs high-precision matrix inversion. This computer program has capabilities for automatic adaptive scaling of the original matrix, application of perturbation techniques in finding the inverse and iterative correction of an (approximate) inverse matrix; each of these has been discussed in the earlier sections.

To enable the test engineer to effectively use the program, a description of the input data cards is given below.

Input Data Deck

The input data deck consists of one card containing input variables, followed by $\left[\frac{N^2+2}{3} \right]$ cards containing the entries of the matrix to be inverted, where N is the dimension of the matrix and $[X]$ is the truncation function.

CARD #1 Option card which contains eight variables.

Variable Name (Format)	Description	Columns
N(I2)	dimension of the square matrix to be inverted	1-2
ISP(I2)	adaptive scaling option ISP = 0 row scaling is not performed ISP = 1 row scaling is performed	3-4
ISQ(I2)	adaptive scaling option ISQ = 0 column scaling is not performed ISQ = 1 column scaling is performed	5-6
IPERT(I2)	diagonal perturbation option IPERT = 0 perturbation is not employed IPERT = 1,2,...,N.perturbation of the $(A_o)_{IPERT,IPERT}$ entry is performed IPERT = N + 1 perturbation of the entire diagonal is carried out	7-8
ISLID(I2)	de-perturbation option to be used with IPERT = N+1 ISLID= 0 deperturbation is performed with subroutine DPERT2 ISLID= 1 deperturbation is performed through a "sliding" correction method (utilizing a family of matrices)	9-10

Variable Name (Format)	Description	Columns
ICOND(I2)	Pre-conditioning option ICOND = 0 no conditioning of the original matrix is employed ICOND = 1 the original matrix A_0 is preconditioned with a multiplier $1 + 1.0 \text{ E-}9$ along the diagonal	11-12
ICORCT(I2)	option to be used in conjunction with IPERT = 0 ICORCT = 0 iterative correction is not used when IPERT = 0 ICORCT = 1 iterative correction is used when IPERT = 0	13-14
IPRINT(I2)	printing option IPRINT = 0 suppresses printing of many of the intermediate matrix quantities used for computation IPRINT = 1 all intermediate matrices are printed	15-16

CARD #2 through card # $\left[\frac{N^2}{3} + 1 \right]$

These cards contain the matrix values, and are read in 3D25.18 format. The matrix entries are entered on the cards in an order prescribed by columns. That is, they are entered as first row, columns 1 through N; second row, columns 1 through N; etc.

END OF FILE CARD

Note a logical flow diagram depicting the effect of various option parameters is given below.

A listing of the complete FORTRAN program is given in Appendix B.

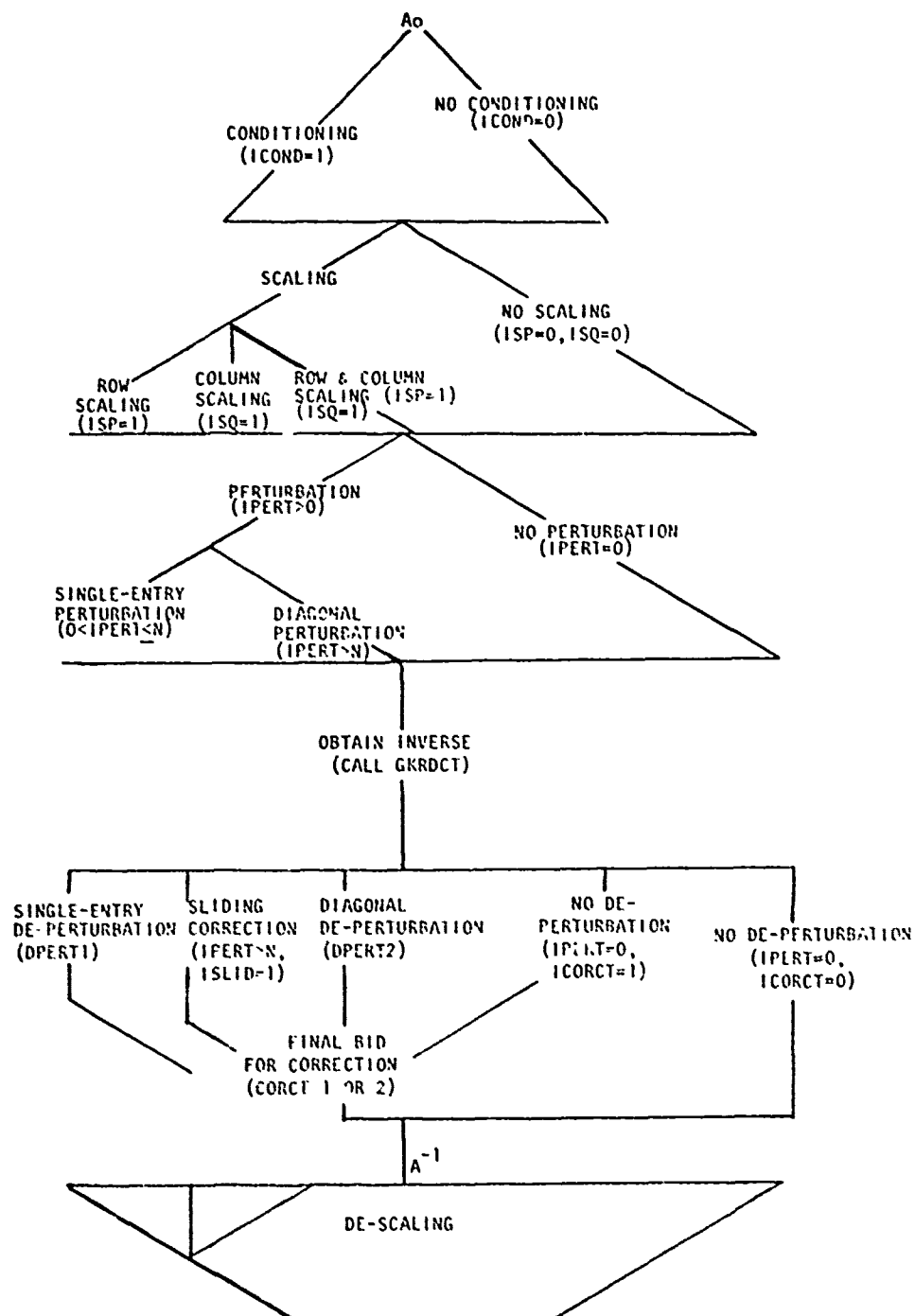


Fig. 3 Flow diagram of HPMINV program options

REFERENCES

- [1] E. J. Ewen and D. D. Weiner, "Identification of Weakly Nonlinear Systems," Midwest Symposium on Circuits and Systems (Lubbock, Texas), Aug. 1977.
- [2] E. Ewen, "Black-Box Identification of Nonlinear Volterra Systems," Ph.D. Dissertation, Syracuse University, Dec. 1975.
- [3] J. Bussgang, L. Ehrman, and J. Graham, "Analysis of Nonlinear Systems with Multiple Inputs," Proc. IEEE, Vol. 62, pp 1088-1119, Aug. 1974.
- [4] S. Narayanan, "Application of Volterra Series to Intermodulation Distortion Analysis of Transistor Feedback Amplifiers," IEEE Trans. Circuit Theory, Vol. CT-17, pp 518-527, Nov. 1970.
- [5] H. J. Kuno, "Analysis of Nonlinear Characteristics and Transient Response of IMPATT Amplifiers," IEEE Trans. Microwave Theory Tech., Vol. MTT-21, pp 694-702, Nov. 1973.
- [6] V. K. Jain, "Filter Analysis by use of pencil of functions," IEEE Trans. Circuits and Systems, Vol. CAS-21, pp 580-583, Sept. 1974.
- [7] V. K. Jain, D. D. Weiner, J. Nebat and T. K. Sarkar, "System identification by pencil of functions method," Proc. RADC Workshop on Spectral Analysis, pp 99-102, May 1978.
- [8] M. L. Van Blaricum and R. Mittra, "A technique for extracting the poles and residues of a system directly from its transient response," IEEE Trans. Ant. Prop., AP-23, pp 777-781, 1975.
- [9] K. J. Iliff and L. W. Taylor, "Determination of stability derivatives from data using a Newton-Raphson minimization technique," NASA technical report, TN D-6579, 1972.
- [10] J. A. Cadzow, "Recursive digital filter synthesis via gradient based algorithms," IEEE Trans. Acous. Sp. Signal Proc., Vol ASSP-24, pp 349-355, Oct. 1976.
- [11] K. J. Astrom and P. Eykhoff, "System identification - A survey," Automatica, pp. 123-162, 1971.
- [12] E. G. Evans and Fischl, "Optimal least-squares time-domain synthesis of recursive digital filters," IEEE Trans. Audio Electroacous., Vol. AU-21, pp. 61-65, Feb. 1973.
- [13] M. J. Levin, "Estimation of a system pulse transfer function in the presence of noise," IEEE Trans. Autom. Control, Vol. AC-9, pp 229-235, July 1964.
- [14] R. L. Kashyap, "Maximum likelihood identification of stochastic linear systems," IEEE Trans. Autom. Control, Vol. AC-15, pp. 25-34, Feb. 1970.

- [15] R. K. Mehra, "Identification of stochastic linear dynamic systems using Kalman filter representation," AIAA J., Vol. 9, pp. 28-31, Jan. 1971.
- [16] E. W. Chenny, Introduction to Approximation Theory. New York: McGraw-Hill 1966.
- [17] W. D. Stanley, Digital Signal Processing. Reston (Prentice-Hall): Reston, 1975.
- [18] D. K. Faddeev and V. N. Faddeeva, Computational Methods of Linear Algebra. San Francisco: W. H. Freeman and Co., 1963.
- [19] G. H. Stewart, Introduction to Matrix Computations. New York: Academic Press, 1973.
- [20] J. H. Wilkinson, The Algebraic Eigenvalue Problem. Oxford: Clarendon Press, 1965.

APPENDIX A

IGRAM PROGRAM

LISTING

```

C      PROGRAM NAME: IGRAM
C      NETWORK TRANSFER-FUNCTION IDENTIFICATION ROUTINE UTILIZING
C      PENCIL-OF-FUNCTIONS METHOD (PURE INTEGRATORS USED)
C
      DIMENSION X(1024),V(1024),XORG(1024),VORG(1024),XREC(1024)
      DIMENSION DATA(1024,2),DATA2(1024,2),BUFF(5072)
      DIMENSION G(20,20),Z(20,20),GAMMA(20),XLAMDA(20),COEFF(20)
      DIMENSION HPULSE(64)
      DIMENSION TITLE(60)
      REAL*8 G,Z,GAMMA,XLAMDA,COEFF
      REAL*8 DELTA,Q,QSAV,DELSAV,AVGQ,AVGM,SUMV2,XSAV
      EQUIVALENCE (Z(1,1),BUFF(1)),(G(1,1),BUFF(1501))
      EQUIVALENCE (DATA(1,1),X(1)),(DATA(1,2),V(1))
      EQUIVALENCE (XORG(1),DATA2(1,1)),(XREC(1),DATA2(1,2))
      COMMON /GKR0/IGKR
      COMMON /NUMBER/N
      COMMON /BIAS/IBIAS
C
      RDEL=3.01
      MAX=20
      MAXPL=1024
      NPT=MAXPL
C
      WRITE(6,2)
      WRITE(6,1022)
      READ(5,1021)TITLE
      WRITE(6,1021)TITLE
      WRITE(6,1023)
4320  READ(5,1001)N,HP1,IPLT
      READ(5,5095)(XORG(K),K=1,HP1),(VORG(K),K=1,HP1)
4321  READ(5,1,END=1234)N,HP1,ISKIP,IREF,IBIAS,DELTA
      IF(N.EQ.10000)GO TO 4320
      WRITE(6,1000)N,HP1,DELTA,IREF,IBIAS
C
      IGKR=1
      IZTS=1
      QSAV=1.0D00
      Q=QSAV
      NN1=N-1
      HP1=N+1
      HP2=N+2
      NPH1=N+N+1
      NPH2=N+N+2
      NN=N-IREF
C
      GENERATING SEQUENCE X(K)
23  DO24I=1,HP1

```



```

SUBROUTINE GRAM(X,V,NP1,N,DELTA,QSAV,KOPT,GAMMA,XLANDA,G,Z,MAX,
1IREM)
C THIS SUBROUTINE PERFORMS THE GRAM TECHNIQUE
C
C DIMENSION X(1),V(1),G(MAX,1),Z(MAX,1),GAMMA(1),XLANDA(1),Q(20),
IDFL(20)
C DIMENSION GAM(25)
C DOUBLE PRECISION GAM
C DOUBLE PRECISION G,Z,GAMMA,XLANDA,DELTA,DEL,PROP,Q,QSAV
C REAL*8 VARQ,VARW,FAC
C REAL*8 SCALE(20),SCAL
C COMMON /GKRD/IGKR
C COMMON /NUMER/NN
C COMMON /BIAS/IBIAS
C
C WRITE(G,1000)
1000 FORMAT(1H1,20X,'THE GRAM TECHNIQUE')
C JOPT = 0 IF DIRECT TRANSMISSION IS ASSUMED
C JOPT=0
C IDLY=IREM
C IF(IREM.NE.0)JOPT=1
C
C DEL IS THE NUMERATOR OF THE KNOWN FIRST ORDER DIGITAL FILTERS
C DO19I=1,N
C DEL(I)=1.0D00
19 Q(I)=QSAV
C WRITE(G,2020)
2020 FORMAT(30X,'Q PARAMETERS')
C CALL PRVEC(Q,N)
C NP1=N+1
C NP2=N+2
C NPNP1=N+N+1
C NPNP2=N+N+2
C NR=NP1-IREM
C NP1PIR=NP1+IREM
C DO 12 I=1,MAX
C DO 12 J=1,MAX
12 Z(I,J)=0.0D00
C VARQ=0.0
C VARW=0.0
C DO 300 I=1,NP1
C VARW=VARW+V(I)*V(I)
300 VARQ=VARQ+X(I)*X(I)
C VARQ=DSQRT(VARQ/IRP1)
C VARW=DSQRT(VARW/IRP1)
C
C
C CALCULATING THE G MATRIX
C IF(IBIAS.EQ.0)GO TO 11
C NPNP2=N+N+2+1
C NR=NP1-IREM+1
11 CONTINUE
C
C DO10I=1,NPNP2
C GAM(I)=0.0
C GAMMA(I)=0.0D00
C DO10J=1,NPNP2
10 G(I,J)=0.0D00
C GAM(1)=1.0
C DO50K=1,NP1
C IF(K-IDLY)25,25,24
25 GAMMA(NP2)=0.0D00
C GO TO 26
24 FAC=1.0
C GAMMA(NP2)=V(K-IDLY)/FAC
C FAC=1.0

```

```

      GAMMA(1)=X(K)/FAC
26  CONTINUE
      DO301=1,N
      GAM(1+1)=GAM(1+1)*Q(1)+GAM(1)*DEL(1)
      GAMMA(1+1)=GAMMA(1)*DEL(1)+GAMMA(1+1)*Q(1)
30  GAMMA(1+NP2)=GAMMA(1+NP1)*DEL(1)+GAMMA(1+NP2)*Q(1)
      IF(1BIAS.EQ.1)GAMMA(NP2)=GAM(1+1)
      DO 40 1=1,NP2
        DO 40 J=1,NP2
40  G(1,J)=G(1,J)+GAMMA(1)*GAMMA(J)
      CONTINUE
      C PREFORM SCALING ON G-MATRIX
      DO 735 1=1,NP2
735 SCALE(1)=DSQRT(G(1,1))
      DO 736 1=1,NP2
      DO 736 J=1,NP2
736 G(1,J)=G(1,J)/(SCALE(1)*SCALE(J))
      WRITE(6,1009)
1009 FORMAT(10X, ---G MATRIX---)
      DO 55 1=1,NP2
55  WRITE(6,3)(G(J,1),J=1,1)
3  FORMAT(1X,10D13.5)
42  CONTINUE
      DOG01=2,NP2
      K=1-1
      DOG0J=1,K
60  G(1,J)=G(J,1)
      C
      IF(JOPT)70,90,70
70  DO30J=1,NP2
      DO801=1,HR
      G(NP1+1,J)=G(NP1PIR+1,J)
80  CONTINUE
      NP2=NP2-IREM
      DO85J=1,NP2
      SCALE(NP1+J)=SCALE(NP1PIR+J)
      DO851=1,HR
      G(J,NP1+1)=G(J,NP1PIR+1)
95  CONTINUE
90  CONTINUE
      CALL GKRDC(G,Z,XLAMD,N,NP2,MAX)
      C DE-SCALE SYNTHETIC COEFFICIENT VECTOR, XLAMD
      DO 741 1=1,NP2
741 XLAMD(1)=XLAMD(1)/SCALE(1)
      IF(1BIAS.EQ.0)GO TO 45
      NP2=NP2-1
      HR=HR-1
43  CONTINUE
      XMEAN=XLAMD(NP2+1)
      IF(JOPT)120,130,120
120 NP2=NP2+IREM
      DO1221=1,HR
122 XLAMD(NP2-1+1)=XLAMD(NP2+HR-1)
      DO1231=1,IREM
123 XLAMD(NP1+1)=0.0D00
130 CONTINUE
      FAC=1.0
      DO3011=NP2,NP2
301 XLAMD(1)=XLAMD(1)*FAC
      WRITE(6,1001)
1001 FORMAT(10X,'THE SYNTHETIC COEFFICIENT VECTOR, XLAMD, IS')
      CALL PRVEC(XLAMD,NP2)
      DO 150 1=1,NP2
150 GAMMA(1)=XLAMD(1)
      C
      C GENERATING GAMMA FROM XLAMD

```

```

      CALL BUIL (DEL,H,MAX)
      DO160I=1,N
      GAMMA(I)=0.0D00
      DO160J=1,NPMP2
160    GAMMA(I)=GAMMA(I)+G(I,J)*XLAMDA(J)
165    CONTINUE
      DO200I=2,NPMP2
200    GAMMA(I)=GAMMA(I)/GAMMA(1)
      GAMMA(1)=1.0D00
      IF(IDLY.EQ.0)GO TO 172
      IDLY1=IDLY+1
      DO 170 I1=IDLY1,NP1
      I=NPMP2+1-I1
170    GAMMA(I+IDLY)=GAMMA(I)
      DO 172 I=1,IDLY
      GAMMA(I+NP1)=0.0D00
172    CONTINUE
      C
      C      CALCULATING THE EQUIVALENT CONTINUOUS DESCRIPTION
      CALL IZTOS(GAMMA,H,DELTA,KOPT)
      WRITE(6,1003)
1003  FORMAT(///,1X,100(1H-),/,1X,100(1H-))
      RETURN
      END

```

```

      SUBROUTINE GKRDC(X,Y,XLAMDA,MH,H,MAX)
      REAL*8 X(MAX,1),Y(MAX,1),A,B,C,D,E,DET,CCC(20,20),XLAMDA(1)
      INTEGER MH(2,20)
      COMMON /GKRDC/IGKR
      C      IGKR=0 USE IS MADE OF THE FIRST ROW OF ADJOINT
      C      1 DIAGONAL (NEGATIVE ENTRIES SET TO ZERO)
      C      2 ABSOLUTE VALUE OF DIAGONAL
      DO 6 I=1,N
      DO 6 J=1,N
6      Y(J,I)=X(J,I)
      A=1.0D0
      DO 43 I=1,N
      B=0.0D0
      L=I
      M=I
      C
      C      FIND LARGEST ENTRY A(L,M) IN LOWER DIAGONAL SUBMATRIX
      C
      DO 18 J=I,N
      DO 18 K=I,N
      IF(DABS(Y(K,J)).LE.B)GO TO 18
      B=DABS(Y(K,J))
      L=K
      M=J
18    CONTINUE
      C
      C      INTERCHANGE ROWS
      C
      IF(L.EQ.I)GO TO 24
      DO 23 J=1,N
      C=Y(L,J)
      Y(L,J)=Y(I,J)

```

```

23  Y(I,J)=C
C
C      INTERCHANGE COLUMNS
C
24  IF(I.EQ.1)GO TO 29
    DO 28 J=1,N
      C=Y(J,I)
      Y(J,I)=Y(J,1)
      Y(J,1)=C
28  C
C      BEGIN SWEEP COLUMNS TO THE RIGHT
C      ARRAYS HUM(1,..),HUM(2,..) KEEP RECORD
C      OF ROW AND COLUMN INTERCHANGES
C
29  HUM(1,1)=L
    HUM(2,1)=N
    B=Y(1,1)
    Y(1,1)=A
    DO 42 J=1,N
      IF(J.EQ.1)GO TO 42
      C=-Y(1,J)
      Y(1,J)=0.000
      DO 41 K=1,N
        D=Y(K,1)+C
        E=Y(K,J)+B*D
        IF(DABS(E).LT.1.00-10*DABS(N))E=0.000
41  Y(K,J)=E/A
42  CONTINUE
43  A=B
C
C      RESTORE COLUMNS
C
    DO 58 I=2,N
      J=N+1-I
      K=HUM(2,J)
      IF(K.EQ.J)GO TO 52
      DO 51 L=1,N
        C=Y(K,L)
        Y(K,L)=Y(J,L)
        Y(J,L)=C
51  Y(J,L)=C
52  K=HUM(1,J)
C
C      RESTORE ROWS
C
    IF(K.EQ.J)GO TO 58
    DO 57 L=1,N
      C=Y(L,K)
      Y(L,K)=Y(L,J)
      Y(L,J)=C
57  Y(L,J)=C
58  CONTINUE
    DET=1/A
C
***** SET IPRINT *****
    IPRINT=0
    IPRINT=1
    IF(IPRINT.NE.1)GO TO 111
    WRITE(6,101)
101  FORMAT(/IX,'DET OF GRAM MATRIX IS ')
    CALL PRVTC(DET,1)
    WRITE(6,102)
102  FORMAT(/IX,'ADJOINT MATRIX IS')
    CALL PRMAT(Y,N,N,MAX)
    DO100I=1,N
    DO100J=1,N
    CCC(I,J)=0.0000
    DO100K=1,N
100  CCC(I,J)=CCC(I,J)+X(I,K)*Y(K,J)

```

```

WRITE(6,103)
103  FORMAT(/1X,'PRODUCT OF ADJOINT AND GRAM MATRICES')
      CALL PRMAT(CCC,H,H,20)
111  CONTINUE
      DO2001=2,H
      IF(IGKR.EQ.0)XLAMDA(I)=Y(1,1)/Y(1,1)
      IF(IGKR.EQ.0)GO TO 200
      A=Y(1,1)
      IF(Y(1,1).LT.0.)A=0.0000
      IF(IGKR.EQ.2)A=DABS(Y(1,1))
      XLAMDA(I)=DSQRT(A/Y(1,1))
      IF(Y(1,1).LT.0.0)XLAMDA(I)=-XLAMDA(I)
200  CONTINUE
      XLAMDA(1)=1.000
      RETURN
      END

```

```

SUBROUTINE IZTOS(GAMMA,H,DELTA,IZTS)
C
C      IZTOS SEPARATES THE NUMERATOR FROM THE DENOMINATOR PARAMETERS
C      IN GAMMA
C
      DIMENSION GAMMA(1),X1(10),X2(10)
      DOUBLE PRECISION GAMMA,X1,X2,DELTA
      COMMON /NUMER/HH
      NP1=H+1
200  DO31=1,NP1
      X1(I)=GAMMA(I)
      X2(I)=-GAMMA(NP1+I)
      CALL ZTOS(X1,X2,H,DELTA,IZTS)
      IZTS=IZTS+1
      IF(IZTS.EQ.4) GO TO 200
      RETURN
      END

```



```

471  CONTINUE
      IF(NH.EQ.0)CAA(1)=1.000
C
C
C      NOW THE FIRST NH ENTRIES OF CA CONTAIN THE S-DOMAIN ZEROES OF NUMERATOR
C      AND THE REMAINING ENTRIES ARE ZEROED OUT.
C      IF(NH.NE.0)CALL POLCON(CA,CAA,0,H)
C
C      WORK ON DENOMINATOR
C
C
919  CALL POLRT(B,TEMP,H,RR,RI,IER)
      DO15 I=1,H
      CR(I)=DCI'PLX(RR(I),RI(I))
16    CF(I)=1.0D'70/CR(I)
909  WRITE(6,1002)
      CALL PROVEC(CF,H)
      IF(IZTS.EQ.0) GO TO 900
235  DO6 I=1,H
6    CR(I)=(-1.0/DELTA)*CDLOG(CR(I))
      WRITE(6,240)
240  FORMAT(' LOGARITHMIC TRANSFORMATION')
      WRITE(6,999)
      WRITE(6,2000)
2000 FORMAT(' POLES IN S DOMAIN')
      CALL PROVEC(CR,H)
      DO3000 I=1,H
3000 CR(I)=-CR(I)
      CALL POLCON(CR,CG,0,H)
C
C
C      ADJUST DC GAIN CONSTANT
C
C
      A2=CAA(1)
      B2=CR(1)
      FAC=(A1/B1)*(B2/A2)
      DO 603 I=1,NHPI
603  CAA(I)=CAA(I)*FAC
      GO TO 2010
C
C
C      DELAYED PULSE INVARIANT TRANSFORMATION
C
C
C      SHIFTS NUMERATOR COEFFICIENTS FOR DELAY
C
C
250  CONT=A(1)
      DO 300 I=1,H
300  A(I)=A(I+1)-CONT*B(I+1)
      A(NHPI)=0.0
400  CALL POLRT(B,TEMP,H,RR,RI,IER)
      DO61 I=1,H
61    CR(I)=DCI'PLX(RR(I),RI(I))
      CF(I)=1.0D'70/CR(I)
      WRITE(6,1002)
1002 FORMAT(1X,'THE POLES OF THE Z-DOMAIN')
      CALL PROVEC(CF,H)
C
C
C      PARTIAL FRACTION EXPANSION
C
C

```

```

      DO3I=1,H
      CON1=1.0D00
      CON2=0.0D00
      DO4J=1,H
      CON2=CON2*CR(I)+A(N-J+1)
      IF(I-J)5,4,5
      CON1=CON1*(1.0D00-CR(I)*CF(J))
      CONTINUE
      CA(I)=CON2/CON1
C
C
C
C
C
C
      TRANSFORMATION OF DENOMINATOR AND NUMERATOR
224  DO2I=1,H
      CR(I)=CDLOG(CR(I))/DELTA
      CA(I)=CA(I)*CR(I)/(1.0D00-CF(I))
      CONTINUE
      CA(NP1)=0.0D00
      WRITE(6,241)
241  FORMAT(' DELAYED PULSE TRANSFORMATION')
      WRITE(6,999)
225  WRITE(6,1004)
1004  FORMAT(' NEGATIVE OF THE POLES IN THE S-DOMAIN')
      CALL PRVVEC(CR,N)
      WRITE(6,1003)
1003  FORMAT(1X,'NUMERATOR CONSTANTS OF FACTORIZED H(S)')
      CALL PRVVEC(CA,H)
      CALL POLCON(CR,CB,0,H)
      DO7I=1,NP1
71    CAA(I)=0.0D00
      DO9K=1,H
      CALL POLCON(CR,CF1,K,H)
      DO9J=1,H
9      CAA(J)=CAA(J)+CF1(J)*CA(K)
      CAA(NP1)=0.0D00
2010  CONTINUE
      DO 450 I=1,NP1
450   CAA(I)=CAA(I)+CONT*CB(I)
C
C
C
403  WRITE(6,1005)
1005  FORMAT(' S-DOMAIN DENOMINATOR')
      CALL PRVVEC(CB,NP1)
      WRITE(6,1006)
1006  FORMAT(' S-DOMAIN NUMERATOR')
      CALL PRVVEC(CAA,NP1)
      DO20I=1,NP1
      B(I)=CB(I)
20    A(I)=CAA(I)
900  RETURN
      END

```

```

SUBROUTINE POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
C
C      COMPUTES THE REAL AND COMPLEX ROOTS OF A REAL POLYNOMIAL
C
C      DESCRIPTION OF PARAMETERS
C      XCOF -VECTOR OF N+1 COEFFICIENTS OF THE POLYNOMIAL
C            ORDERED FROM SMALLEST TO LARGEST POWER
C      COF  -WORKING VECTOR OF LENGTH N+1
C      M    -ORDER OF POLYNOMIAL
C      ROOTR-RESULTANT VECTOR OF LENGTH N CONTAINING REAL ROOTS
C            OF THE POLYNOMIAL
C      ROOTI-RESULTANT VECTOR OF LENGTH N CONTAINING THE
C            CORRESPONDING IMAGINARY ROOTS OF THE POLYNOMIAL
C      IER  -ERROR CODE WHERE
C            IER=0  NO ERROR
C            IER=1  N LESS THAN ONE
C            IER=2  N GREATER THAN 36
C            IER=3  UNABLE TO DETERMINE ROOT WITH 500 ITERATIONS
C                   ON 5 STARTING VALUES
C            IER=4  HIGH ORDER COEFFICIENT IS ZERO
C
C      DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)
C      DOUBLE PRECISION XO,YO,X,Y,XPR,YPR,UX,UY,V,Y1,XT,U,XT2,YT2,SUMSQ,
1  DX,DY,TEMP,ALPHA,XCOF,COF,ROOTR,ROOTI,ER1,ER2,XSS,XS,YSS,YS,TOL
C
C      LIMITED TO 36TH ORDER POLYNOMIAL OR LESS.
C      FLOATING POINT OVERFLOW MAY OCCUR FOR HIGH ORDER
C      POLYNOMIALS BUT WILL NOT AFFECT THE ACCURACY OF THE RESULTS.
C
C      METHOD
C      NEWTON-RAPHSON ITERATIVE TECHNIQUE. THE FINAL ITERATIONS
C      ON EACH ROOT ARE PERFORMED USING THE ORIGINAL POLYNOMIAL
C      RATHER THAN THE REDUCED POLYNOMIAL TO AVOID ACCUMULATED
C      ERRORS IN THE REDUCED POLYNOMIAL.
C
C      ER2=1.0D+50
C      TOL=1.0D-3
C      IFIT=0
C      N=M
C      IER=0
C      IF(XCOF(N+1))10,25,10
10  IF(N) 15,15,32
C
C      SET ERROR CODE TO 1
C
15  IER=1
20  IF(IER)200,201,200
230  WRITE(6,203)IER
203  FORMAT(1X,'ERROR CALLED FROM POLRT, IER = ',I3)
201  RETURN
C
C      SET ERROR CODE TO 4
C
25  IER=4
GO TO 20
C
C      SET ERROR CODE TO 2
C
30  IER=2
GO TO 20
32  IF(N-36) 35,35,30
35  NX=N
NXX=N+1
N2=1
KJ1 = N+1
DO 40 L=1,KJ1

```

```

      MT=KJ1-L+1
40 COF(MT)=XCOF(L)
C
C      SET INITIAL VALUES
C
45 XO=.00500101
   YO=0.01000101
C
C      ZERO INITIAL VALUE COUNTER
C
      IN=0
50 X=XO
C
C      INCREMENT INITIAL VALUES AND COUNTER
C
      XO=-10.0*YO
      YO=-10.0*X
C
C      SET X AND Y TO CURRENT VALUE
C
      X=XO
      Y=YO
      IN=IN+1
      GO TO 59
55 IF IT=1
   XPR=X
   YPR=Y
C
C      EVALUATE POLYNOMIAL AND DERIVATIVES
C
59 ICT=0
60 UX=0.0
   UY=0.0
   V =0.0
   YT=0.0
   XT=1.0
   U=COF(IN+1)
   IF(U) 65,130,65
65 DO 70 I=1,IN
   L =IN-I+1
   TEMP=COF(L)
   XT2=X*XT-Y*YT
   YT2=X*YT+Y*XT
   U=U+TEMP*XT2
   V=V+TEMP*YT2
   FI=I
   UX=UX+FI*XT*TEMP
   UY=UY-FI*YT*TEMP
   XT=XT2
70 YT=YT2
   SUMSQ=UX*UX+UY*UY
   IF(SUMSQ) 75,110,75
75 DX=(V*UY-U*UX)/SUMSQ
   X=X+DX
   DY=-(U*UY+V*UX)/SUMSQ
   Y=Y+DY
   XSS=X
   YSS=Y
   IF(YSS.EQ.0.000)YSS=1.000
   IF(XSS.EQ.0.000)XSS=1.000
   ER1=DABS(PX/XSS)+DABS(DY/YSS)
   IF(ER1.GT.ER2)GO TO 73
   ER2=ER1
   XS=XSS
   YS=YSS
78 IF(ER1-TOL)100,30,30

```

```

C
C      STEP ITERATION COUNTER
C
80 ICT=ICT+1
   IF(ICT-500) 60,85,85
85 IF(IFIT)100,90,100
90 IF(IN-5) 50,95,95

C
C      SET ERROR CODE TO 3
C
95 IER=3
   X=XS
   Y=YS
   ER1=ER2
100 DO 105 L=1,NXX
   MT=KJ1-L+1
   TEMP=XCOF(MT)
   XCOF(MT)=COF(L)
105 COF(L)=TEMP
   ITEMP=N
   N=NX
   NX=ITEMP
   IF(IFIT) 120,55,120
110 IF(IFIT) 115,50,115
115 X=XPR
   Y=YPR
120 IFIT=0
122 IF(PASS(Y)-1.00-3*PASS(X))135,125,125
125 ALPHA=X*X
   SUMSQ=X*X+Y*Y
   N=N-2
   GO TO 140
130 X=X-1
   NX=NX-1
   NXX=NXX-1
135 Y=Y-1
   SUMSQ=X*X
   ALPHA=X
   N=N-1
140 COF(2)=COF(2)+ALPHA*COF(1)
145 DO 150 L=2,N
150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)
155 ROOT1(N2)=Y
   ROOT1(N2)=X
   IF(ER1.GT.TOL)WRITE(6,554)N2,ER1
554 FORMAT(1X,'ERROR ON ',I3,'TH ROOT IS ',D10.3)
   ER2=1.0D+50
   N2=N2+1
   IF(SUMSQ) 160,165,160
160 Y=-Y
   SUMSQ=0.0
   GO TO 155
165 IF(N) 20,20,45
   END

```

```

SUBROUTINE PLOTIT(X,NF,NPT,I1,I2,I3,MAXX,ISC,SCALE)
DIMENSION X(MAXX,NF),IPL(97),IP(97),ICH(9),IS(9),XMX(9),XMH(9),F(9)
1)
REAL*8 XMAX,XMIN
DATA ICH/1H*,1H*,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
DATA IHI,1H,1HG,1HL/1H1,1H,1H2,1H</
DO2011=1,97
201 IP(1)=IH
DO2001=1,8
J=17*1+1
200 IP(J)=IHI
IP(1)=IHI
IF(NF-3)15,15,17
15 NP=NF
GO TO 16
17 NP=3
16 DO111J=1,NF
XMIN=X(1,J)
XMAX=X(1,J)
DO11=2,NPT
IF(XMAX-X(1,J))3,2,2
2 IF(XMIN-X(1,J))1,1,4
4 XMIN=X(1,J)
GO TO 1
3 XMAX=X(1,J)
1 CONTINUE
XMX(J)=XMAX
XMH(J)=XMIN
NFF=NF
IF(ISC.NE.1.OR.NF.EQ.1)GO TO 116
XMAX=XMX(1)
XMIN=XMH(1)
DO117J=2,NF
IF(XMAX.LT.XMX(J))XMAX=XMX(J)
IF(XMIN.GT.XMH(J))XMIN=XMH(J)
117 CONTINUE
DO113J=1,NF
XMX(J)=XMAX
113 XMH(J)=XMIN
NFF=1
116 IF(SCALE.EQ.1.0)GO TO 114
DO115J=1,NFF
XX=0.5*(XMX(J)*(1+SCALE)+XMH(J)*(1-SCALE))
XMH(J)=0.5*(XMX(J)*(1.0-SCALE)+XMH(J)*(1.0+SCALE))
115 XMX(J)=XX
114 DO1113J=1,NFF
XMAX=XMX(J)
XMIN=XMH(J)
XX=DABS(XMAX)
XM=DABS(XMIN)
IF(XX.LT.XM)GO TO 503
MAX=1
XSAV=XX
GO TO 507
503 MAX=0
XSAV=XM
507 IF(XMAX-XMIN-1.0D-6*XSAV)5,5,6
5 XMAX=XMAX+XX+1.0D00
XMIN=XMIN-XM-1.0D00
XX=DABS(XMAX)
XM=DABS(XMIN)
IF(XX.LT.XM)GO TO 513
MAX=1
XSAV=XX
GO TO 6
513 MAX=0

```

```

        XSAV=XH
6      XH=ALOG10(XSAV)
        JJ=XH
        XX=JJ
        JJ=JJ-1
        IF(XH-XX,NE.0.0.AND.XSAV.LT.1.0)JJ=JJ-1
        TEN=10.0**JJ
        K=XSAV/(TEN*2.0)
        IF(KAX.EQ.1)GO TO 505
        IF(XMIN.LT.0.9)K=-(K+1)
        XX=XMAX, YEN
        KMIN=K*3
        XH=XMIN
505     XH=XH+3.0
        IF(XH.LT.XX)GO TO 505
        KMAX=XH
        IF(KMAX+KMIN.LT.0)KMAX=-KMIN
        GO TO 1112
503     IF(XMAX.GT.9.0)K=-(K+1)
        XX=XMIN/TEN
        KMAX=-K*5
        XH=XMAX
504     XH=XH-3.0
        IF(XH.GT.XX)GO TO 504
        KMIN=XH
        IF(KMAX+KMIN.LT.0)KMIN=-KMAX
1112    XMX(J)=KMAX*TEM
1113    XMN(J)=KMIN*TEM
        IF(ISC,NE.1.OR,NF.EQ.1)GO TO 119
        DO120J=2,NF
        XMX(J)=XMX(1)
        XMN(J)=XMN(1)
120     DO112J=1,NFF
119     F(1)=XMX(J)
        XMN=(XMX(J)-XMN(J))/8.0D30
        DO71=1,3
7       F(1+1)=F(1)+XMN
        IF(F(1)*F(9).LT.9.9)F(5)=9.0
        WRITE(6,100)J,F
100     FORMAT(5X,'FUNCTION',12,7X,9E12.4)
112     WRITE(6,106)ICH(J),IP,IP
106     FORMAT(5X,'SYMBOL ',1X,A1,18X,97A1,/,33X,97A1)
        DOS1=1,97
        IPL(1)=IP(1)
        DO1131=1,NF
113     F(1)=96.9D30/(XMX(1)-XMN(1))
        DO6001=1,NF
600     IS(1)=1
        IF(NP.EQ.1)ASSIGN 1001 TO NPPP
        IF(NP.EQ.2)ASSIGN 1002 TO NPPP
        IF(NP.EQ.3)ASSIGN 1003 TO NPPP
        DO9J=11,12,13
        KMAX=1
        KMIN=97
        DO101=1,NF
        K=F(1)*(X(J,1)-XMN(1))*1.5
        IS1=IS(1)
        IF(K)662,662,6621
6621    IF(97-K)6331,633,633
6331    K=97
        ICC=1HG
        GO TO 644
662     K=1
        ICC=1HL
        GO TO 644
633     ICC=ICH(1)

```

```

644 IF (ISI-K) GO 11, C01, 601
6011 DOG02L=ISI, K
602 IPL(L)=ICCC
IF (KMIN.GT.ISI) KMIN=ISI
IF (KMAX.LT.K) KMAX=K
GO TO 603
601 DOG04L=K, ISI
604 IPL(L)=ICCC
IF (KMIN.GT.K) KMIN=K
IF (KMAX.LT.ISI) KMAX=ISI
603 IS(I)=K
10 CONTINUE
GO TO HPPP, (1001, 1002, 1003)
1001 WRITE(G, 102) J, X(J, 1), IPL
102 FORMAT(1X, I4, E12.4, 16X, 97A1)
GO TO 10
1002 WRITE(G, 103) J, X(J, 1), X(J, 2), IPL
103 FORMAT(1X, I4, 2E12.4, 4X, 97A1)
GO TO 10
1003 WRITE(G, 104) J, X(J, 1), X(J, 2), X(J, 3), IPL
104 FORMAT(1X, I4, 3E9.3, 1X, 97A1)
18 DO 19 I=KMIN, KMAX
19 IPL(I)=IP(I)
9 CONTINUE
WRITE(G, 105)
105 FORMAT(1H1)
RETURN
END

```

```

SUBROUTINE RESPON(X, V, N, GAMMA, XLAMDA, IP1)
DIMENSION X(1), V(1), GAMMA(1), XLAMDA(1)
REAL*8 XSAV, GAMMA, XLAMDA
NM1=N-1
NP1=N+1
NPNP1=N+N+1
NPNP2=N+N+2
DO 19 I=1, NPNP1
19 XLAMDA(I)=0.0D00
XSAV=0.0D00
DO 20 K=1, NP1
IF (N.EQ.1) GO TO 25
DO 21 I=1, NM1
J=NP1-I
21 XLAMDA(J)=XLAMDA(J-1)
25 CONTINUE
DO 22 I=1, N
J=NPNP2-I
22 XLAMDA(J)=XLAMDA(J-1)
XLAMDA(1)=XSAV
XLAMDA(NP1)=V(K)
XSAV=0.0D00
DO 23 I=1, NPNP1
23 XSAV=XSAV-GAMMA(I+1)*XLAMDA(I)
IF (DABS(XSAV).GE.1.0D10) XSAV=0.0D00
20 X(K)=XSAV
RETURN
END

```

```

SUBROUTINE ERROR(XREC,V,GAMMA,HP1,N,XLAMBDA,XORG)
  DIMENSION XREC(1),V(1),XORG(1)
  DIMENSION VVV(20)
  REAL*8 GAMMA(1),XLAMBDA(1),AVGW,SUMV2,AVGQ
  CALL RESPON(XREC,V,N,GAMMA,XLAMBDA,HP1)
  AVGW=0.0000
  SUMV2=0.0000
  DO26 I=1,HP1
    SUMV2=SUMV2+XORG(I)*XORG(I)
    AVGQ=XORG(I)-XREC(I)
    AVGW=AVGW+AVGQ*AVGQ
  26  AVGQ=AVGW/SUMV2
    AVGQ=DSQRT(AVGQ)
    AVGW=100.0*AVGQ
    AVGW=100.0*AVGW
    WRITE(6,27)AVGW,AVGQ
  27  FORMAT(1X,'PER CENT MEAN POWER ERROR OF RECONSTRUCTION',F8.3,///,
    1X,'PER CENT OF SQUARE ROOT OF POWER ERROR IN RECONSTRUCTION',F8.3)
  RETURN
  END

```

```

SUBROUTINE BUILD(A,Q,DEL,N,MAX)
  REAL*8 A(MAX,1),Q(1),DEL(1),PROD
  HP1=N+1
  HP2=N+2
  A(1,HP1)=1.0000
  PROD=1.0000
  DO312 K=1,N
    I=HP1-K
    PROD=PROD/DEL(I)
    A(1,I)=PROD
  312  A(K+1,HP1)=1.0000
    DO313 I=2,HP1
      DO313 K=1,N
        J=HP1-K
  313  A(1,J)=(A(1,J+1)-Q(J)*A(1-1,J+1))/DEL(J)
      DO314 I=1,HP1
        DO314 J=1,HP1
          A(1,J+HP1)=0.0000
          A(1+HP1,J)=0.0000
  314  A(1+HP1,J+HP1)=A(1,J)
  1005  WRITE(6,1005)
    FORMAT(1X,'A-MATRIX')
    CALL PRMAT(A,HP2,HP2,MAX)
  RETURN
  END

```

```

SUBROUTINE PRMAT(A,N,H,MAX)
  DOUBLE PRECISION A
  C
  C THIS SUBROUTINE OUTPUTS DOUBLE PRECISION DOUBLE DIMENSIONED ARRAY
  DIMENSION A(MAX,1)
  WRITE(6,1)
  DO21 I=1,H
  2  WRITE(6,3)(A(I,J),J=1,N)
  3  FORMAT(1X,10D13.5)
  WRITE(6,1)
  WRITE(6,1)
  1  FORMAT(//)
  RETURN
  END

```

```

SUBROUTINE PRVVEC(A,N)
C
C THIS SUBROUTINE PRINTS OUT A COMPLEX SINGLE DIMENSIONED ARRAY
C A COMPLEX NUMBER OF THE FORM A + B J IS OUTPUTTED IN THE FORM
C ( A, B J) WHERE J = SQUARE ROOT OF -1
  DIMENSION A(1)
  COMPLEX*16 A
  WRITE(6,2)
  WRITE(6,1)(A(I),I=1,N)
1  FORMAT(1X,1H(,D17.10,1H,,D17.10,3H J))
  WRITE(6,2)
2  FORMAT(/)
  RETURN
END

```

```

SUBROUTINE PRVEC(A,N)
C
C THIS SUBROUTINE OUTPUTS DOUBLE PRECISION SINGLE DIMENSIONED ARRAY
  DIMENSION A(1)
  DOUBLE PRECISION A
  WRITE(6,31)
  WRITE(6,1)(A(I),I=1,N)
1  FORMAT(1X,10D13.5)
  WRITE(6,31)
  WRITE(6,31)
31 FORMAT(/)
  RETURN
END

```

```

SUBROUTINE POLCON(C,R2,K,N)
C
C A POLYNOMIAL CONSTRUCTION PROGRAM NEEDED FOR ZTOS
C
  DIMENSION C(1),R2(1)
  COMPLEX*16 C,R2,COMP
  REAL*8 DC(2)
  EQUIVALENCE (COMP,DC)
  NP1=N+1
  DO10 I=2,NP1
10  R2(I)=0.0D00
  R2(1)=1.0D90
  DO4 I=1,N
  COMP=C(I)
  IF(I.EQ.K.OR.(DC(1).EQ.0.0D0.AND.DC(2).EQ.0.0D0))GO TO 2
  DO2 JJ=1,I
  J=I-JJ+1
2  R2(J+1)=R2(J)+C(I)+R2(J)
  R2(1)=R2(1)+C(I)
  CONTINUE
  RETURN
END

```

APPENDIX B


```

20 IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,Q)
C CONTINUE
C END OF SCALING
IF(IPRINT.EQ.1)WRITE(6,105)
IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,A)
CALL MEQUAT(N,MAX,A,C)
IF(IPERT.EQ.0)GO TO 90
C
C PERTURB MATRIX A, C=A+(FAC)*(DIAG A)=A+FAC*(DA)
C
FAC=1.0D-04
DO 30 INC=1,1
FAC=FAC*1.0D-02
WRITE(6,100)
WRITE(6,500)FAC
DO 35 I=1,N
DO 35 J=1,N
DA(I,J)=0.0D00
B(I,J)=A(I,J)
IF(IPERT.GT.N)GO TO 82
IF(I.NE.IPERT)GO TO 83
82 CONTINUE
IF(I.EQ.J)B(I,J)=A(I,J)+FAC*A(I,J)
IF(I.EQ.J)DA(I,J)=A(I,J)
83 CONTINUE
C(I,J)=B(I,J)
85 CONTINUE
IF(IPRINT.EQ.1)WRITE(6,130)
IF(IPRINT.EQ.1)CALL PRMAT(N,MAX,C)
IF(IPERT.GT.N.AND.ISLID.EQ.1)GO TO 90
IF(IPERT.LE.N)GO TO 90
C
C NOW 'C' IS THE PERTURBED MATRIX
C INV A= INV C+ FAC*(INV C)*(DA)*(INV C)+(FAC)**2*((INV C)*
C (DA))**2*(INV C)+...
C NAPPRX=1 FIRST 2 TERMS OF THE SERIES FOR INV A ARE USED
C =2 FIRST 3 TERMS OF THE SERIES FOR INV A ARE USED
C
NAPPRX=2
CALL DPRT2(N,MAX,NAPPRX,FAC,C,DA,B)
WRITE(6,141)
CALL PRMAT(N,MAX,B)
GO TO 2295
C APPLICATION OF PERTURBATION METHOD OVER-----
90 CONTINUE
CALL GKRPCT(N,MAX,C,B,DA)
IF(PDIF-QDIF)666,666,667
666 CALL CORCT1(C,B,N,MAX,5,1)
GO TO 668
667 CALL CORCT2(C,B,N,MAX,5,1)
668 CONTINUE
IF(IPERT.EQ.0)GO TO 2295
IF(IPERT.GT.N.AND.ISLID.EQ.1)GO TO 2291
CALL DPRT1(A,B,N,MAX,FAC,IPERT)
2291 CONTINUE
IF(IPERT.LE.N)GO TO 2295
FFAC=FAC*1.0D-04
DO 91 K=1,11
WRITE(6,300)K
300 FORMAT(/,10X,'VALUE OF K =',12,/)
DO 89 I=1,N
DO 89 J=1,N
89 G(I,J)=C(I,J)-FFAC*DA(I,J)
IF(PDIF-QDIF)777,777,778
777 CALL CORCT1(G,B,N,MAX,5,0)
GO TO 779
778 CALL CORCT2(G,B,N,MAX,5,0)
779 CONTINUE
IF(K.LE.4)FFAC=FFAC*10.0D00
IF(K.GT.4.AND.K.LE.7)FFAC=FFAC*1.773279D00
IF(K.GE.8)FFAC=FFAC*1.154782D0

```

```

91  CONTINUE
2295 CONTINUE
    IF(1PERT.EQ.0.AND.1CORCT.EQ.0)GO TO 783
    WRITE(6,580)
580  FORMAT(5X,'FINAL BID FOR IMPROVEMENT')
    FRAC=1.000
    NIT=H
    IF(PDIF-QDIF)781,731,782
781  CALL CORCT1(A,B,H,MAX,NIT,0)
    GO TO 783
782  CALL CORCT2(A,B,H,MAX,NIT,0)
783  CONTINUE
    WRITE(6,109)
    CALL PRMAT(H,MAX,B)

C
C      INVERSE AA = (INVERSE Q)*(INVERSE A)*(INVERSE P)
C
    CALL HEQUAT(H,MAX,B,C)
    IF(1SP.EQ.0.AND.1SQ.EQ.0)GO TO 92
    CALL HEQUAT(H,MAX,B,DA)
    IF(1SQ.EQ.1)CALL DHUT(H,H,H,MAX,FAC,0,Q,B,DA)
    IF(1CP.HF.1)CALL HEQUAT(H,MAX,DA,C)
    IF(1SP.EQ.1)CALL DHULT(H,H,H,MAX,FAC,0,DA,P,C)
92  CONTINUE
    IF(1PRINT.EQ.1)WRITE(6,107)
    IF(1PRINT.EQ.1)CALL PRMAT(H,MAX,C)
    IF(PDIF-QDIF)93,93,94
93  CONTINUE
    CALL DHULT(H,H,H,MAX,FAC,0,A,B,G)
    IF(1SP.EQ.0)GO TO 95
    CALL DHULT(H,H,H,MAX,FAC,0,G,P,B)
    CALL DINV(H,MAX,P)
    CALL DHULT(H,H,H,MAX,FAC,0,P,B,G)
    GO TO 95
94  CONTINUE
    CALL DHULT(H,H,H,MAX,FAC,0,B,A,G)
    IF(1SQ.EQ.0)GO TO 95
    CALL DHULT(H,H,H,MAX,FAC,0,Q,G,B)
    CALL DINV(H,MAX,Q)
    CALL DHULT(H,H,H,MAX,FAC,0,B,Q,G)
95  CONTINUE
    IF(1PRINT.EQ.1)WRITE(6,103)
    IF(1PRINT.EQ.1)CALL PRMAT(H,MAX,G)
    CALL HERROR(H,MAX,G,ERROR)
    WRITE(6,249)
    IF(1PERT.EQ.0)STOP
80  CONTINUE
100  FORMAT(//////////)
101  FORMAT(5X,'DIAGONAL SCALE MATRIX, P')
102  FORMAT(5X,'INVERSE P MATRIX')
103  FORMAT(5X,'DIAGONAL SCALE MATRIX, Q')
104  FORMAT(5X,'INVERSE Q MATRIX')
105  FORMAT(5X,'MATRIX A')
106  FORMAT(5X,'INVERSE A MATRIX')
107  FORMAT(5X,'INVERSE OF ORIGINAL MATRIX, AA')
108  FORMAT(5X,'PRODUCT OF ORIGINAL MATRIX, AA, AND ITS COMPUTED INVERS
1E')
109  FORMAT(/,10X,'IMPROVED INVERSE MATRIX')
127  FORMAT(5X,'(ROW) SCALED MATRIX A')
130  FORMAT(5X,'C=A+ EPS*D MATRIX')
141  FORMAT(5X,'INV(A) = INV(C) + EPS*( ) + EPS**2 *( )')
140  FORMAT(10X,'MATRIX DIMENSION =',12,/,10X,'ISP =',12,/,10X,'ISQ =',
1,12,/,10X,'1PERT =',12,/,10X,'1SLID =',12,/,10X,'1COND =',12,
2,/,10X,'1CORCT =',12,/)
240  FORMAT(912)
249  FORMAT(5X,'*****
1*****')
250  FORMAT(3D25.13)
350  FORMAT(////,5X,'ORIGINAL MATRIX, AA')
600  FORMAT(//,5X,'FAC=',D20.11,/)
    STOP
    END

```

```

SUBROUTINE GKRDCT(N,MAX,X,Y,CCC)
REAL*8 X(MAX,1),Y(MAX,1),A,B,C,D,E,DET,CCC(MAX,1)
INTEGER NUM(2,20)
COMMON /PR/PRINT
COMMON /PROD/PDIF,QDIF
DO 6 I=1,N
DO 6 J=1,N
6 Y(J,1)=X(J,1)
A=1.0D0
DO 43 I=1,N
B=0.0D0
L=1
M=1
C
C      FIND LARGEST ENTRY A(L,M) IN LOWER DIAGONAL SUBMATRIX
C
DO 18 J=1,N
DO 18 K=1,N
IF(DABS(Y(K,J)).LE.B)GO TO 18
B=DABS(Y(K,J))
L=K
M=J
18 CONTINUE
C
C      INTERCHANGE ROWS
C
IF(L.EQ.1)GO TO 24
DO 23 J=1,N
C=Y(L,J)
Y(L,J)=Y(1,J)
23 Y(1,J)=C
C
C      INTERCHANGE COLUMNS
C
IF(M.EQ.1)GO TO 29
DO 28 J=1,N
C=Y(J,M)
Y(J,M)=Y(J,1)
28 Y(J,1)=C
C
C      BEGIN SWEEP COLUMNS TO THE RIGHT
C      ARRAYS NUM(1,.) ,NUM(2,.) KEEP RECORD
C      OF ROW AND COLUMN INTERCHANGES
C
29 NUM(1,1)=L
NUM(2,1)=M
B=Y(1,1)
Y(1,1)=A
DO 42 J=1,N
IF(J.EQ.1)GO TO 42
C=-Y(1,J)
Y(1,J)=0.0D0
DO 41 K=1,N
D=Y(K,1)*C
E=Y(K,J)+B*D
C IF(DABS(E).LT.1.0D-10*DABS(D))E=0.0D0

```

```

41 Y(K,J)=E/A
42 CONTINUE
43 A=B
C
C RESTORE COLUMNS
C
DO 58 I=2,N
J=N+1-I
K=NUM(2,J)
IF(K.EQ.J)GO TO 52
DO 51 L=1,N
C=Y(K,L)
Y(K,L)=Y(J,L)
51 Y(J,L)=C
52 K=NUM(1,J)
C
C RESTORE ROWS
C
IF(K.EQ.J)GO TO 53
DO 57 L=1,N
C=Y(L,K)
Y(L,K)=Y(L,J)
57 Y(L,J)=C
58 CONTINUE
DET=A
WRITE(6,101)DET
101 FORMAT(/1X,'DET OF ORIG MATRIX IS ',D24.17)
DET=1.0D00/DET
DO 111 I=1,N
DO 111 J=1,N
111 Y(I,J)=Y(I,J)*DET
WRITE(6,102)
102 FORMAT(/1X,'INVERSE MATRIX IS ')
C
C
C CALL PRMAT(N,NAX,Y)
C
C
DO100I=1,N
DO100J=1,N
CCC(I,J)=0.0D00
DO100K=1,N
IF(PDIF-QDIF)66,66,67
66 CCC(I,J)=CCC(I,J)+X(I,K)*Y(K,J)
GO TO 100
67 CCC(I,J)=CCC(I,J)+Y(I,K)*X(K,J)
100 CONTINUE
C
C
C
WRITE(6,103)
IF(IPRINT.EQ.1)CALL PRMAT(N,NAX,CCC)
CALL MERROR(N,NAX,CCC,ER)
103 FORMAT(/1X,'PRODUCT OF INVERSE AND ORIG MATRICES ')
RETURN
END

```

```

SUBROUTINE CONCT1(X,Y,H,MAXX,ITER,IPR)
DIMENSION X(MAXX,1),Y(MAXX,1),F(20,20),E(20,20),Z(20,20)
COMMON /SOL/FRAC1
DOUBLE PRECISION X,Y,F,E,Z
DOUBLE PRECISION FRAC,ER1,ER2,ER
DOUBLE PRECISION FRAC1,FRSAV
WRITE(6,317)FRAC1
DO 290 ITER=1,ITER
ER2=1.0D10
ER=ER2
FRAC=FRAC1
ER1=0.0D00

C
C   CALCULATE ERROR MATRIX,F=X*Y-I
C
DO 210 I=1,H
DO 210 J=1,H
F(I,J)=0.0D00
DO 200 K=1,H
200 F(I,J)=F(I,J)+X(I,K)*Y(K,J)
IF(I.EQ.J)F(I,J)=F(I,J)-1.0D00
ER1=ER1+F(I,J)*F(I,J)
210 CONTINUE
ER1=DSQRT(ER1/(H*H))
WRITE(6,320)ER1
320 FORMAT(/,10X,'ORIGINAL RMSE = ',D15.8,/)
IF(IPR.EQ.1)WRITE(6,300)
IF(IPR.EQ.1)CALL PRMAT(H,12,F)

C
C   CALCULATE IMPROVED INVERSE,Y(IMPROVED)=Y-Y*F
C
DO 230 I=1,H
DO 230 J=1,H
E(I,J)=0.0D00
DO 220 K=1,H
220 E(I,J)=E(I,J)+Y(I,K)*F(K,J)
230 CONTINUE
IF(IPR.EQ.1)WRITE(6,310)
IF(IPR.EQ.1)CALL PRMAT(H,12,E)
235 CONTINUE
ER2=ER
CALL IEQUAT(H,20,Y,Z)
DO 240 I=1,H
DO 240 J=1,H
240 Z(I,J)=Y(I,J)-FRAC*E(I,J)
CALL DMULT(H,H,H,20,FRAC,0,X,Z,F)
CALL MERROR(H,20,F,ER)
IF(ER.GE.ER2)GO TO 230
FRSAV=FRAC
FRAC=FRAC*0.5D00
GO TO 235
230 CONTINUE
DO 270 I=1,H
DO 270 J=1,H
270 Y(I,J)=Y(I,J)-FRSAV*E(I,J)
290 CONTINUE
300 FORMAT(5X,'F MATRIX')
310 FORMAT(5X,'E MATRIX')
317 FORMAT(10X,'INITIAL FRAC= ',D11.4)
RETURN
END

```

```

SUBROUTINE CORCT2(X,Y,H,MAXX,HITER,IPR)
DIMENSION X(MAXX,1),Y(MAXX,1),F(20,20),E(20,20),Z(20,20)
COMMON /SOL/FRAC1
DOUBLE PRECISION X,Y,F,E,Z
DOUBLE PRECISION FRAC,ER1,ER2,ER
DOUBLE PRECISION FRAC1,FRSAV
WRITE(5,317)FRAC1
DO 290 ITER=1,HITER
ER2=1.0D10
ER=ER2
FRAC=FRAC1
ER1=0.0D00

C
C   CALCULATE ERROR MATRIX,F=Y*X-I
C
DO 210 I=1,H
DO 210 J=1,H
F(I,J)=0.0D00
DO 200 K=1,H
200 F(I,J)=F(I,J)+Y(I,K)*X(K,J)
IF(I.EQ.J)F(I,J)=F(I,J)-1.0D00
ER1=ER1+F(I,J)*F(I,J)
210 CONTINUE
ER1=DSQRT(ER1/(H*H))
WRITE(6,320)ER1
320 FORMAT(/,10X,'ORIGINAL RMSE = ',D15.8,/)
IF(IPR.EQ.1)WRITE(6,300)
IF(IPR.EQ.1)CALL PRMAT(4,12,F)

C
C   CALCULATE IMPROVED INVERSE,Y(IMPROVED)=Y-F*Y
C
DO 230 I=1,H
DO 230 J=1,H
E(I,J)=0.0D00
DO 220 K=1,H
220 E(I,J)=E(I,J)+F(I,K)*Y(K,J)
230 CONTINUE
IF(IPR.EQ.1)WRITE(5,310)
IF(IPR.EQ.1)CALL PRMAT(4,12,E)
235 CONTINUE
ER2=ER
CALL MEQUAT(H,20,Y,Z)
DO 240 I=1,H
DO 240 J=1,H
240 Z(I,J)=Y(I,J)-FRAC*E(I,J)
CALL DMULT(H,H,H,20,FRAC,0,Z,X,F)
CALL MERROR(H,20,F,ER)
IF(ER.GE.ER2)GO TO 280
FRSAV=FRAC
FRAC=FRAC*0.5D00
GO TO 235
280 CONTINUE
DO 270 I=1,H
DO 270 J=1,H
270 Y(I,J)=Y(I,J)-FRSAV*E(I,J)
290 CONTINUE
300 FORMAT(5X,'F MATRIX')
310 FORMAT(5X,'E MATRIX')
317 FORMAT(10X,'INITIAL FRAC= ',D11.4)
RETURN
END

```

```

SUBROUTINE DPERT1(X,Y,H,MAXX,FAC,IPERT)
DIMENSION X(MAXX,1),Y(MAXX,1),U(20),V(20),F(20,20)
DOUBLE PRECISION X,Y,U,V,F
DOUBLE PRECISION FAC,ALPHA,ER
COMMON /PR/ IPR

C
C
      IPR=2
      IPR=1
      WRITE(6,91)
      CALL DMULT(H,H,H,20,FAC,0,X,Y,F)
      IF(IPR.GE.1)WRITE(6,95)
      IF(IPR.GE.1)CALL PRMAT(H,20,F)
      CALL MERROR(H,20,F,ER)
      WRITE(6,320)ER
320  FORMAT(/,10X,'ORIGINAL RMSE = ',D15.6,/)
C
C      CALCULATE IMPROVED INVERSE
C
      ER=FAC
      FAC=FAC*X(IPERT,IPERT)
      ALPHA=Y(IPERT,IPERT)
      ALPHA=1.000-FAC*ALPHA
      ALPHA=FAC/ALPHA
      IF(IPR.GE.1)WRITE(6,201)FAC,ALPHA
      FAC=ER
      DO 150 I=1,H
        J(I)=Y(I,IPERT)
        V(I)=Y(IPERT,I)
150  CONTINUE
      IF(IPR.EQ.2)WRITE(6,103)(U(I),I=1,H)
      IF(IPR.EQ.2)WRITE(6,105)
      IF(IPR.EQ.2)WRITE(6,103)(V(I),I=1,H)
      DO 240 I=1,H
        DO 240 J=1,H
240  F(I,J)=U(I)*ALPHA*V(J)
      IF(IPR.EQ.1)WRITE(6,300)
      IF(IPR.EQ.1)CALL PRMAT(H,20,F)
      DO 250 I=1,H
        DO 250 J=1,H
250  Y(I,J)=Y(I,J)+F(I,J)
      CALL DMULT(H,H,H,20,FAC,0,X,Y,F)
      IF(IPR.GE.1)WRITE(6,97)
      IF(IPR.GE.1)CALL PRMAT(H,20,F)
      CALL MERROR(H,20,F,ER)
      WRITE(6,109)
      CALL PRMAT(H,MAXX,Y)
290  CONTINUE
      WRITE(6,91)
      FORMAT(5X,'*****')
1  *****')
95  FORMAT(5X,'APERT: A*INV(C)',/)
97  FORMAT(5X,'APERT: A*IMPROVED INV(C)',/)
103 FORMAT(2X,4(D25.10,3X))
105 FORMAT(/)
109 FORMAT(/,10X,'IMPROVED INVERSE MATRIX')
201 FORMAT(5X,'A *EPS=',D15.4,'ALPHA=',D15.4,/)
300 FORMAT(5X,'F MATRIX')
      RETURN
      END

```

```

SUBROUTINE DPRT2(N,MAX,HAPPRX,FAC,C,DA,B)
DOUBLE PRECISION C(MAX,1),DA(MAX,1),B(MAX,1),G(20,20)
DOUBLE PRECISION FAC,PDIF,QDIF
COMMON /PROD/PROD,PDIF,QDIF
CALL GKROCT(N,MAX,C,B,G)
IF(PDIF-QDIF)51,51,52
51 CALL CORCT1(C,B,N,MAX,N,0)
GO TO 53
52 CALL CORCT2(C,B,N,MAX,N,0)
53 CONTINUE
CALL DMULT(N,N,N,MAX,FAC,1,B,DA,G)
CALL DMULT(N,N,N,MAX,FAC,3,G,B,C)
WRITE(6,113)
CALL PRMAT(N,MAX,C)
IF(HAPPRX.EQ.1)GO TO 37
CALL DMULT(N,N,N,MAX,FAC,9,G,G,DA)
CALL DMULT(N,N,N,MAX,FAC,0,DA,G,G)
WRITE(6,121)
CALL PRMAT(N,MAX,G)
87 CONTINUE
DO 90 I=1,N
DO 90 J=1,N
IF(HAPPRX.EQ.1)S(I,J)=B(I,J)+C(I,J)
IF(HAPPRX.EQ.2)S(I,J)=B(I,J)+C(I,J)+G(I,J)
90 CONTINUE
116 FORMAT(5X,'EPS* INV(C) *D*INV(C)')
121 FORMAT(5X,'EPS*INV(C) *D*INV(C) + EPS**2*(INV(C)*D)**2 *INV(C)')
RETURN
END

```

```

SUBROUTINE HTRANS(N,MAX,X)
DOUBLE PRECISION X(MAX,1),Y(20,20)
C
DO 1 I=1,N
DO 1 J=1,N
1 Y(I,J)=X(J,1)
DO 2 I=1,N
DO 2 J=1,N
2 X(I,J)=Y(I,J)
RETURN
END

```

```

SUBROUTINE HEQUAT(N,MAX,X,Y)
DOUBLE PRECISION X(MAX,1),Y(MAX,1)
C
C
C EQUATE MATRIX Y TO MATRIX X
DO 1 I=1,N
DO 1 J=1,N
1 Y(I,J)=X(I,J)
RETURN
END

```

```

SUBROUTINE HSCALE(N,MAX,THRES,IS,DIF,X,Y)
DOUBLE PRECISION X(MAX,1),Y(MAX,1),B(20,20)
DOUBLE PRECISION THRES,ZERO,BTOT,THR,BTHR,YMAX,YMIN,DIF
C
C      ROW OR COLUMN SCALING OF MATRIX X
C      IS=1, ROW SCALING
C      IS=0, COLUMN SCALING
C
ZERO=1.0D-50
YMAX=-1.0D+10
YMIN=1.0D+10
IF(IS.EQ.0)CALL MTRANS(N,MAX,X)
DO 30 I=1,N
  THR=-50.0
  DO 10 J=1,N
    IF(DABS(X(I,J)).GT.ZERO)GO TO 5
    B(I,J)=-50.0
  GO TO 10
5  CONTINUE
  B(I,J)=DLOG10(DABS(X(I,J)))
  IF(B(I,J).GT.THR)THR=B(I,J)
10 CONTINUE
  BTOT=THR-THRES
  BTOT=0.0D00
  ICT=0
  DO 20 J=1,N
    Y(I,J)=0.0D00
    IF(B(I,J).LT.BTHR)GO TO 20
    ICT=ICT+1
    BTOT=BTOT+B(I,J)
20 CONTINUE
  BTOT=BTOT/ICT
  Y(I,1)=10.**BTOT
  IF(BTOT.GT.YMAX)YMAX=BTOT
  IF(BTOT.LT.YMIN)YMIN=BTOT
  DO 30 J=1,N
    X(I,J)=X(I,J)/Y(I,1)
30 CONTINUE
  DIF=YMAX-YMIN
  IF(IS.EQ.0)CALL MTRANS(N,MAX,X)
  RETURN
END

```

```

SUBROUTINE PMULT(M,N,L,MAX,S,IS,A,B,C)
DOUBLE PRECISION A(MAX,1),B(MAX,1),C(MAX,1),S,TEMP
C
C      DOUBLE PRECISION MATRIX MULTIPLICATION, C=A*S*S
C
DO 10 I=1,M
  DO 10 J=1,L
    C(I,J)=0.0D00
    DO 10 K=1,N
      C(I,J)=C(I,J)+A(I,K)*B(K,J)
    IF(IS.EQ.1.AND.K.EQ.N)C(I,J)=S*C(I,J)
10 CONTINUE
  RETURN
END

```

```

SUBROUTINE PRMAT(N,MAX,A)
103  FORMAT(2X,4(D25.16,3X))
105  FORMAT(//)
      REAL*8 A(MAX,1)
      WRITE(6,105)
      M=((N-1)/4)+1
      DO 10 I=1,N
      WRITE(6,107)
      DO 15 L=1,M
      J1=(L-1)*4+1
      J2=J1+3
      IF(L.EQ.M)J2=M
      WRITE(6,103)(A(I,J),J=J1,J2)
15   CONTINUE
10   CONTINUE
      WRITE(6,104)
104  FORMAT(////)
107  FORMAT(/)
      RETURN
      END

```

```

SUBROUTINE MEHRROR(N,MAX,C,ERROR)
DOUBLE PRECISION C(MAX,1),TEM,ATEM
DOUBLE PRECISION ERROR,AER
COMMON /ME/AER
C
C
      ERROR=0.0D00
      AER=0.0D00
      DO 1 I=1,N
      DO 1 J=1,N
      TEM=C(I,J)
      IF(I.EQ.J)TEM=C(I,J)-1.0D00
      ATEM=DABS(TEM)
      IF(ATEM.GT.AER)AER=ATEM
      ERROR=ERROR+TEM*TEM
1   CONTINUE
      ERROR=ERROR/(N*N)
      ERROR=DSQRT(ERROR)
      WRITE(6,100)ERROR,AER
100  FORMAT(12X,'(MAT-1): RMSE=',D11.4,' MAX=',D11.4)
      RETURN
      END

```

```

SUBROUTINE DINV(N,MAX,X)
DOUBLE PRECISION X(MAX,1)
C
C      INVERSION OF DIAGONAL MATRIX, X
C
      DO 1 I=1,N
      DO 1 J=1,N
1   IF(I.EQ.J)X(I,J)=1.0D00/X(I,J)
      RETURN
      END

```

- APPENDIX C -

In this appendix we present the details involved in Example 2, page 10, of Chapter 2. The example deals with application of the identification technique to data obtained from a single-stage transistor amplifier. The schematic and equivalent model of the circuit are repeated, for convenience, in Fig. C1.

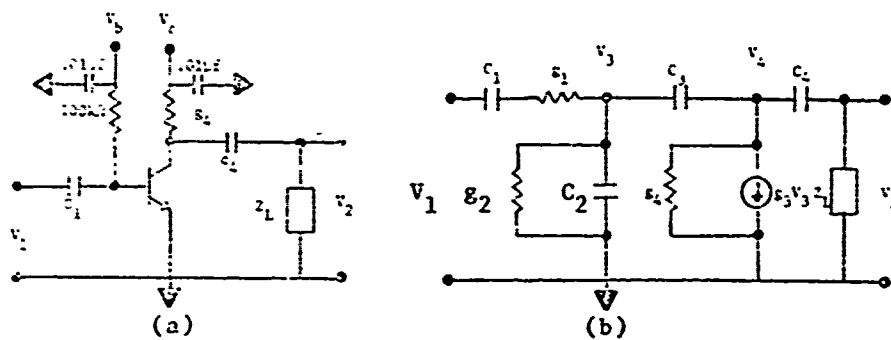


Fig. C1. (a) Schematic of Common-Emitter Amplifier Circuit.
(b) Equivalent Circuit Model

From Fig. C1b, the following equations may be written by inspection:

$$\begin{bmatrix} Y_1 + G_2 + (C_2 + C_3)s & -C_3s \\ G_3 - C_3s & Y_2 + G_4 + C_3s \end{bmatrix} \begin{bmatrix} V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} Y_1 \\ 0 \end{bmatrix} V_1 \quad (C1)$$

$$V_2 = \left(\frac{Z_L C_4 s}{Z_L C_4 s + 1} \right) V_4 \quad (C2)$$

where, by definition,

$$Y_1 \equiv \frac{G_1 C_1 s}{G_1 + C_1 s} \quad (C3)$$

$$Y_2 = \frac{C_4 s / Z_L}{1/Z_L + C_4 s} \quad (C4)$$

Equations (C1) and (C2) may be solved simultaneously, to yield the desired transfer function, as follows. From (C1),

$$V_4 = \frac{\begin{vmatrix} Y_1 + g_2 + (C_2 + C_3)s & Y_1 \\ g_3 - C_3 s & 0 \end{vmatrix}}{\begin{vmatrix} Y_1 + g_2 + (C_2 + C_3)s & -C_3 s \\ g_3 - C_3 s & Y_2 + g_4 + C_3 s \end{vmatrix}} V_1 \quad (C5)$$

Substitution of equation (C5) into (C2) yields the required expression, i.e.,

$$H(s) = \frac{V_2}{V_1} = \left(\frac{Z_L C_4 s}{Z_L C_4 s + 1} \right) \frac{\begin{vmatrix} Y_1 + g_2 + (C_2 + C_3)s & Y_1 \\ g_3 - C_3 s & 0 \end{vmatrix}}{\begin{vmatrix} Y_1 + g_2 + (C_2 + C_3)s & -C_3 s \\ g_3 - C_3 s & Y_2 + g_4 + C_3 s \end{vmatrix}} \quad (C6)$$

Simplification of the expression for $H(s)$ requires unwieldy algebraic manipulation, and will therefore not be presented. However, it can be shown that for Z_L real $H(s)$ assumes the following form:

$$H(s) = \frac{K s^2 (s + \alpha_1)}{(s + \alpha_1)(s + \alpha_2)(s + \alpha_3)(s + \alpha_4)} \quad (C7)$$

For the particular transistor circuit shown in Fig. C1b, the following parameter values are assumed:

$$\begin{aligned} C_1 &= 0.01\mu\text{f} & C_3 &= 5 \text{ pf} & g_1 &= 4\text{m}\Omega & g_3 &= 40\text{m}\Omega & Z_L &= 1\text{K}\Omega \text{ (real)} \\ C_2 &= 0.0\text{pf} & C_4 &= 0.01\mu\text{f} & g_2 &= 1\text{m}\Omega & g_2 &= 0.5\text{m}\Omega \end{aligned}$$

Substituting these values in Equation (C6) and performing the required simplifications yields

$$H(s) = \frac{8(10^7) s^2 (s - 8000(10^6))}{(s + 0.033(10^6))(s + 0.080(10^6))(s + 25.2(10^6))(s + 1205.1(10^6))} \quad (\text{C8})$$

A Bode (corner) plot of Equation (C8) is given in Fig. C2. Inspection of this frequency response shows that the circuit is broad-band (i.e., its poles are separated by several decades). In order that the network transfer function be identified reliably, the spectrum of the input signal must contain sufficient energy concentrated in the vicinity of the network poles and zeros (for excitation and manifestation of these critical frequencies). However, it is not convenient to synthesize -- and realize in the laboratory -- an input having such characteristics.

Therefore, the network function may (and in this case will) be broken into constituent functions, each of which will be valid for a particular frequency range.

Inspection of Equation (C8) reveals that an adequate low-frequency description of $H(s)$ is

$$H_L(s) = \frac{(21.04) s^2}{(s + 0.033(10^6))(s + 0.080(10^6))} \quad (\text{Low-freq.}) \quad (\text{C9})$$

which is valid up to 1 Mr/s. That is, Equation (C9) will closely approximate $H(s)$ for radian frequencies below approximately 10^6 rad/sec. This observation can be seen clearly from the Bode plot in Fig. C2.

Through similar considerations, expressions describing the mid-high and high frequency characteristics of $H(s)$ are obtained:

$$H_{MH}(s) = - \frac{531.1(10^6)}{(s+25.2(10^6))} \quad (\text{Mid to High Frequency Transition}) \quad (C10)$$

$$H_H(s) = \frac{8(10^7)(s-8000(10^6))}{(s+25.2(10^6))(s+1205.1(10^6))} \quad (\text{High Frequency}) \quad (C11)$$

Equation (C10) will be valid in the frequency range from approximately 10^6 to 10^9 rad/sec, while Equation (C11) will be valid for frequencies from 10^9 rad/sec onward.

The identification technique may now be used to determine models for the network behavior by considering each of the three regions separately. Improvement in the methodology and reliability of identification of broad-band networks (systems) is being investigated under a new research task. For example, pre-filtering the output data in order to isolate the various frequency regions is now being pursued.

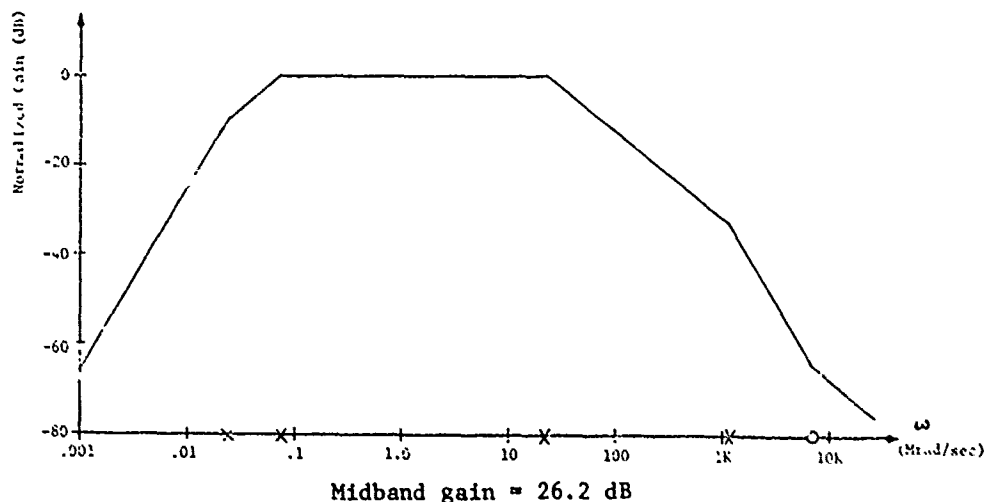


Fig. C2. Magnitude characteristic (Bode plot) of a wide band amplifier.

i) Low Frequency Region

Our approximate description of the low-frequency behavior of $H(s)$ is given in eq. (C9). In order that a reliable model for this region be obtained, via the program IGRAM, several factors must be considered. First, a careful choice of the input must be made in order to excite the low frequency modes of the system. We need to isolate these modes of the response, and will therefore use an input signal whose spectral content is concentrated in the low frequency region. A satisfactory choice is a single triangular pulse of duration 125 μ sec. This signal will supply sufficient energy to the low frequency modes and relatively small amounts to the higher frequency modes.

Next, we must decide upon a sampling interval, Δ . A useful rule-of-thumb in making this choice is to sample at a frequency f_s at least ten times the highest frequency of interest. For the system under consideration, the highest frequency of interest is $0.013(10^6)\text{Hz}^*$. Thus, a sampling interval $\Delta = 1/f_s = 0.25 \mu\text{sec}$ should be quite adequate. Notice that while we are sampling at an adequate rate for the low frequency modes, we are undersampling the high frequency modes. That is, the system as a whole is broad-band and we are sampling at a rate suitable only for the low frequency portion. Therefore, frequency aliasing can be expected to occur. The effect of this aliasing, however, (of the high frequency modes) appears as evenly distributed noise of relatively small power spectrum density.

An important, but less obvious, consideration is the total duration of the test record used in modeling. Whenever possible, a record long enough to have a few time constants, say one to four, of the slowest mode must be used. Using this criterion, a 1000 point record ($MPl = 1000$) for the network under consideration should suffice.

ii) Mid to High Frequency Transition

Our approximate description of the mid to high frequency transition behavior of $H(s)$ is given by eq. (C10). Considerations similar to those made in the last section yield the following choices. Realizing that a narrowband signal must be

* It is unrealistic to expect that the design or test engineer know the exact frequencies of interest. However, it is assumed that he has some idea of the critical frequencies of the system.

used -- so as to excite only the mid-high frequency mode ($s=-25.2(10^6)$), an exponentially decaying sinusoid was chosen as the input signal. The center frequency of this input lies in the frequency range of interest. The sampling interval was chosen to be 0.01 μsec , and a 500-point record was used for modeling.

In modeling this region, the option IBIAS = 1 was used. The reason for this choice is as follows. Due to the low-frequency modes, a transient response will appear in the system output in addition to the desired mid-frequency response. However, over the short duration of our record (5 μsec) this slowly varying transient will appear relatively constant, resembling a d.c. bias. The option IBIAS = 1 allows the program to separate this "bias" and hence calculate a more reliable model for the mid-high transition range.

iii) High Frequency Region

The approximate high frequency description of $H(s)$ is given in eq. (C11). The input signal used for network excitation must be narrowband (for previously mentioned reasons). Thus, a slowly decaying sinusoid with center frequency in the critical region was chosen. Five hundred points of input-output signals, with a sampling interval $\Delta = 0.00025 \mu\text{sec}$, were used for modeling.

Once the results for each of the frequency regions have been obtained, they may be used to synthesize the overall network response. This can be done by correctly combining the model descriptions of the various frequency regions. Details of such a synthesis will not be discussed.

APPENDIX D

s-Domain to z-Domain Conversion

Sampled Signal

When sampled at uniformly spaced time instants $k\Delta$, an analog signal $x(t)$ yields a numerical sequence $\xi = \{x_k\}$, where $x_k = x(k\Delta)$. To this numerical sequence we can associate a continuous-time signal $x^*(t) = \sum_{k=-\infty}^{\infty} x_k \delta(t-k\Delta)$, called the (ideally-)sampled signal. If the original signal is bandlimited by $1/2\Delta$ Hz, then $x(t)$ can be recovered from $x^*(t)$ through low-pass filtering, and the sampling process may be regarded as a one-to-one mapping. We define the Laplace transform of the sampled signal in the customary way; this gives

$$X^*(s) = \sum_{k=-\infty}^{\infty} x_k (e^{-s\Delta})^k \quad (D1)$$

Now, since the z transform of $\xi = \{x_k\}$ is

$$X(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k} \quad (D2)$$

we make the extremely interesting observation that

$$X^*(s) = X(z) \Big|_{z=e^{s\Delta}} \quad (D3)$$

Note: It should be borne in mind that the substitution $z=e^{s\Delta}$ into $X(z)$ yields the Laplace transform of $x^*(t)$, not of $x(t)$. Under the condition of bandlimitedness (by $1/2\Delta$ Hz) this substitution yields a transform that agrees with $X(s)$ in a suitable neighborhood of $s=0$ in the s -plane.

We now focus attention on the matter of conversion of transfer functions from s -domain to z -domain and vice-versa. An exhaustive treatment is given in reference [17]. Here, we summarize three of the most widely used conversion techniques.

1. Logarithmic Pole-Zero Conversion

This technique uses the relation $z = e^{s\Delta}$, or $s = \frac{1}{\Delta} \ln z$, upon the poles and zeros of the function under consideration. Thus

$$H(s) = \frac{s^{\ell} \prod_{i=1}^m (s+b_i)}{\prod_{i=1}^n (s+a_i)} \leftrightarrow H_1(z) = \Delta^{(n-\ell-m)} \frac{(z-1)^{\ell} \prod_{i=1}^m (z-\beta_i)}{\prod_{i=1}^n (z-\alpha_i)} \quad (D4)$$

where

$$\alpha_i = e^{-a_i \Delta} \quad \text{D5a'}$$

$$\beta_i = e^{-b_i \Delta} \quad \text{D5b}$$

2. Pulse-Invariant Conversion

This technique has the merit that the response of $H(s)$ to an input $x_{\text{pulse}}(t) = \sum_{k=-\infty}^{\infty} x_k p(t-k\Delta)$, where $p(t)$ = square pulse over $(0, \Delta)$, coincides with the response of $H(z)$ to the sequence $\xi = \{x_k\}$. In many cases of practical interest $x_{\text{pulse}}(t)$ is an excellent approximation to $x(t)$; in such cases this technique of conversion promises close agreement of the response of $H(s)$ to $x(t)$ and of $H(z)$ to $\{x_k\}$, at the sampling instants. The conversion is described by

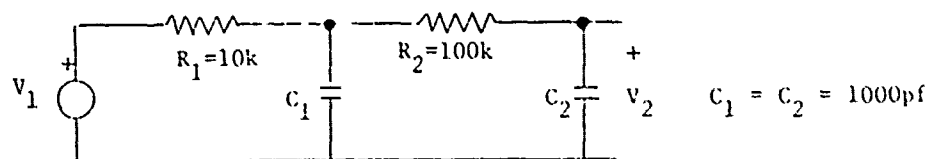
$$H(s) = \sum_{i=1}^n \frac{b_i}{s + a_i} \leftrightarrow H_2(z) = \sum_{i=1}^n \frac{b_i (1 - e^{-a_i \Delta})}{a_i (1 - e^{-a_i \Delta} z^{-1})} \quad \text{(D6)}$$

3. Impulse-Invariant Conversion

When this technique is used the response of $H(s)$ to $x^*(t)$ coincides with that of $H(z)$ to $\{x_k\}$ (at the sampling instants). The conversion is described by

$$H(s) = \sum_{i=1}^n \frac{b_i}{s + a_i} \leftrightarrow H_3(z) = \sum_{i=1}^n \frac{\Delta b_i}{1 - e^{-a_i \Delta} z^{-1}} \quad \text{(D7)}$$

Example: Sampling Interval $\Delta = 5\mu\text{s}$



$$H(s) = \frac{1 \times 10^9}{s^2 + (1.2 \times 10^5) s + (1 \times 10^9)}$$

$$= \frac{1 \times 10^9}{(s+9009.8)(s+110,990.2)} \leftrightarrow H_1(z) = \frac{0.025}{(z-0.95595)(z-0.57410)} \text{ by (14)}$$

$$= \frac{9805.8}{s+9009.8} - \frac{9805.8}{s+110990.2} \leftrightarrow H_2(z) = \frac{0.047941z}{(z-0.95595)} - \frac{0.037628z}{(z-0.57410)} \quad \text{by (D6)}$$

$$\begin{array}{ccc} " & " & \\ & & \leftrightarrow H_3(z) = \frac{0.049029z}{(z-0.95595)} - \frac{0.049029z}{z-0.57410} \quad \text{by (D7)} \end{array}$$

Remark: In 'IGRAM', conversion techniques 1 and 2 have been programmed.
 However, the present setting IZTS=1 (see page 44) leads to logarithmic conversion.



MISSION of Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.