NASA CR-144889

STICAP
A LINEAR CIRCUIT ANALYSIS PROGRAM WITH STIFF SYSTEMS CAPABILITY
Volume I - Theory Manual

by Charlie H. Cooke

Prepared under Contract NAS1-9434, T25 by
OLD DOMINION UNIVERSITY
School of Engineering
Norfolk, VA 23508
for
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

ABSTRACT

STICAP (Stiff Circuit Analysis Program) is a FORTRAN IV, Version 2.3, computer program written for the CDC-6400-6600 computer series and SCOPE 3.0 operating system. It provides the circuit analyst a tool for automatically computing the transient responses and frequency responses of large linear time invariant networks, both stiff and non-stiff. The circuit description and user's program input language is engineer-oriented, making simple the task of using the program.

Three volumes of documentation are available for the STICAP program; a theory manual, a user's manual, and a system's programmers manual. Volume I describes the engineering theories underlying STICAP and gives further references to the literature. Volume II, the user's manual, explains user interaction with the program and gives results of typical circuit design applications. Volume III depicts the program structure from a system's programmers viewpoint and contains flow charts and other software documentation.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Most computer aided network analysis programs are designed to relieve the user of the following burdens:

(a) Circuit translation - the obtaining of the differential and algebraic equations governing the network, starting from an easily specified description of the circuit in the terminology of the network designer, and

(b) Numerical integration - the obtaining of a numerically accurate solution of the initial value problem for this set of circuit equations.

The present state of the art allows a reasonably effective solution of the former problem. However, most first generation circuit analysis programs are somewhat restricted in scope, as regards the latter, in the instance of stiff networks which are characterized by widely separated time constants. In such circumstances the numerical integration techniques implemented in the first versions of programs like SCEPTRE and ECAP require such prohibitively small time steps that they become impractical as aids to analysis.

The primary raison d'etre for STICAP is the motivation to combine the capabilities of circuit translation using the state variable topological approach with efficient numerical integration techniques for transient analysis, using algorithms which possess both stiff and non-stiff capabilities. The STICAP program is restricted to the analysis of linear time invariant networks. It represents a merging, with some modifications to each, of Pottle's circuit analysis program CORNAP with Gear's program ALGORITH 407 - DIFSUB, for the automatic integration of ordinary differential equations.

The program package is best viewed as consisting of three separate components, or modes of operation, each having some advantages and disadvantages over the others in different circumstances. In each mode the common method of circuit translation is that originally employed in program CORNAP; a topological approach the result of which is a set of first order linear differential equations governing the time evolution of the circuit state variables. The CORNAP mode makes selectable the program CORNAP with all previous capabilities, but optional selection of certain data printing features. These capabilities include calculation of transfer functions, zeroes of tramsmission, frequency and time response of the circuit.

The fourth order numerical integration algorithm implemented in CORNAP for time domain analysis is absolutely stable; hence it may be used for either stiff or non-stiff networks. However, the stepsize is fixed throughout the duration of computation, a feature which can be uneconomical in some instances. Further, other than impulse or step functions, the only type of circuit input is sampled data.

The Gear and Matrix modes may be used to compute time domain transient, impulse, or step responses only, with the option of calling CORNAP subroutines to obtain transfer functions and zeroes of transmission. The Gear mode allows the selection of either an Adam's integration method, suitable for non-stiff equations, or else the methods of Gear, suitable for stiff equations. This mode can be used for analysis of the general linear time invariant network, with forcing functions specified using the full power of the FORTRAN language, or by means of sampled data. In both cases automatic order selection techniques and variations in the step size are employed as the integration procedes, to achieve a desired level of accuracy with the minimum number of integration steps. The maximal order truncation error selectable by changing from one algorithm to another via the automatic order selection process is an eighth order Adam's method or a sixth order stiff algorithm.

The matrix method was developed by personnel of Old Dominion University, Norfolk, Virginia. In the matrix mode a spectral decomposition of the system matrix in terms of its eigen-values is employed, to obtain a closed form solution which avoids a numerical integration. The resulting method is computationally rapid and may be used for either stiff or non-stiff networks; however, it is applicable only in the case of no repeated eigenvalues of the system matrix, and for systems whose forcing functions are linear combinations of sinusoidal, cosinusoids, impulse, or step functions.

The literature concerning the theoretical aspects of the combined program package is rather extensive; the papers considered most helpful are indicated in the bibliographies. In the present document the theory first presented by Pottle, Gear, et.al., is to some extent paraphrased and/or summarized; in some instances clarifying examples are given.

CHAPTER II

CIRCUIT THEORY

The underlying network theory and mathematical algorithms described by C. Pottle [10] [11] and incorporated in the design of program CORNAP will now be reiterated, essentially as in Pottle's description. The programs used by CORNAP to translate the circuit description to a set of first order differential equations in the state variables of the circuit are commonly used by all program modes.

2.1 CIRCUIT TRANSLATION: FROM NETWORK DESCRIPTION TO STATE EQUATIONS

The algorithm for obtaining the state equations of a general active linear network implemented in CORNAP is a modification of that proposed by Dervisoglu [2] and has been described in detail elswhere [1], [3], [4]. In this implementation it is assumed that the state equations sought have a subset of the capacitor voltages and inductor currents as state variables. The problem is, of course, to proceed from an easily given description of the network to a set of first order coupled differential equations which contain only state variables and independent source inputs. All other branch voltages and currents are removed along the way. A summary of the method follows.

1) Form the fundamental loop matrix with respect to a <u>normal</u> tree; that is, a tree which contains all voltage sources as tree branches, all current sources as links, and as many capacitive tree branches and inductive links as possible. This matrix expresses a relation giving all tree-branch currents in terms of link currents and, using its negative

transpose, giving all link voltages in terms of tree-branch voltages.

2) Form and solve a set of algebraic equations relating

a) resistive tree-branch voltages to their currents (and hence to link currents),

b) resistive link currents to their voltages (and hence to tree-branch voltages),

c) controlled sources to their controlling quantities (and hence to link currents and tree-branch voltages).

3) Form a set of initial state equations obtained by replacing inductor voltages by expressions involving derivatives of inductor currents and replacing capacitive currents by expressions involving derivatives of capacitor voltages.

4) By row reduction techniques obtain the final state equations with dependent state variables removed.

The Fundamental Loop Matrix

Although a human may find a normal tree by inspection, for a computer this tree is most easily found from the incidence matrix. This matrix is easily constructed from the given topological input data, each row expressing the Current Law with respect to a region surrounding one node. A systematic, computer-oriented method for obtaining a normal tree and its associated fundamental loop matrix from the incidence matrix is as follows:

1) Arrange the columns (branches) of the incidence matrix $\underset{\sim}{C}$ in the order

voltage sources

capacitors
resistors
inductors
current sources

Note that since source conversion and transportation are not particularly adapted to computerization, the approach taken here requires that each branch consist of only one element; voltage and current sources are branches and are not required to be members of "generalized branches" with passive elements.

2) Apply the standard Gauss-Jordan row reduction technique to the rows of $\underset{\sim}{C}$ with the modification that if, at the i^{th} step, all entries in the i^{th} column below and including the i^{th} row are zero, one simply goes on to the first column where a 'one' can be brought into the i^{th} row. When processing is complete, the $\underset{\sim}{C}$ matrix is in "row echelon form".

3) Rearrange the columns (branches) of $\underset{\sim}{C}$ to produce a matrix of the form

$$\left[\begin{array}{c|c} \underset{\sim}{I} & -\underset{\sim}{F} \end{array}\right]$$

The first columns with the single one in them refer to tree branches and the rest refer to links.

Tree branch voltages and link currents will henceforth appear in upper case and tree-branch currents and link voltages in lower case. With the appearance of the fundamental loop matrix $\underset{\sim}{F}$ we have obtained the relations

$$\underline{i} = \underset{\sim}{F}\,\underline{I} \quad \text{and} \quad \underline{v} = -\underset{\sim}{F}^{T}\,\underline{V} \tag{1}$$

relating all lower case quantities to upper case quantities. The extended Gauss-Jordan algorithm appears at other places in the sequel and is basic to the whole technique. It is introduced here without discussion of roundoff errors and such, since at no step can numbers other than +1, -1, or 0 appear. The restriction on the generality of network which can be handled appearing here is that controlled voltage sources must not appear in loops with other voltage sources, and controlled current sources must not form cut-sets with other current sources. Obviously there are legitimate networks excluded by this restriction, but they are rather more hypothetical than physical.

Removal of Resistors and Controlled Sources

Topological considerations now permit the replacement of any lower case quantity by a linear combination of upper case quantities. But since each resistor voltage (current) is related to its current (voltage), and each controlled source to its control, a set of algebraic equations may be constructed in the form

$$\left[\begin{array}{c} \\ \\ \underline{M}_1 \\ \\ \\ \end{array}\right] \begin{array}{l} \left[\begin{array}{c} \underline{V}_d \\ \underline{V}_R \\ \underline{I}_R \\ \underline{I}_d \end{array}\right] \end{array} \begin{array}{l} \text{- controlled voltage sources} \\ \text{- tree branch resistors} \\ \text{- link resistors} \\ \text{- controlled current sources} \end{array}$$

$$\left[\begin{array}{c} \\ \\ \underline{M}_2 \\ \\ \\ \\ \end{array}\right] \left[\begin{array}{c} \underline{E} \\ \underline{V}_C \\ \underline{V}_L \\ \underline{I}_C \\ \underline{I}_L \\ \underline{J} \end{array}\right] \begin{array}{l} \text{- independent voltage sources} \\ \text{- tree branch capacitors} \\ \text{- tree branch inductors} \\ \text{- link capacitors} \\ \text{- link inductors} \\ \text{- independent current sources} \end{array} \qquad (2)$$

Figure 1 illustrates a case where the first controlled voltage source is controlled by the first tree branch capacitor, and the first link resistor depends (through $-\underset{\sim}{C}^T$) on (possibly all) the tree branch voltages. Only the shaded areas on a given row represent possibly non-zero areas.

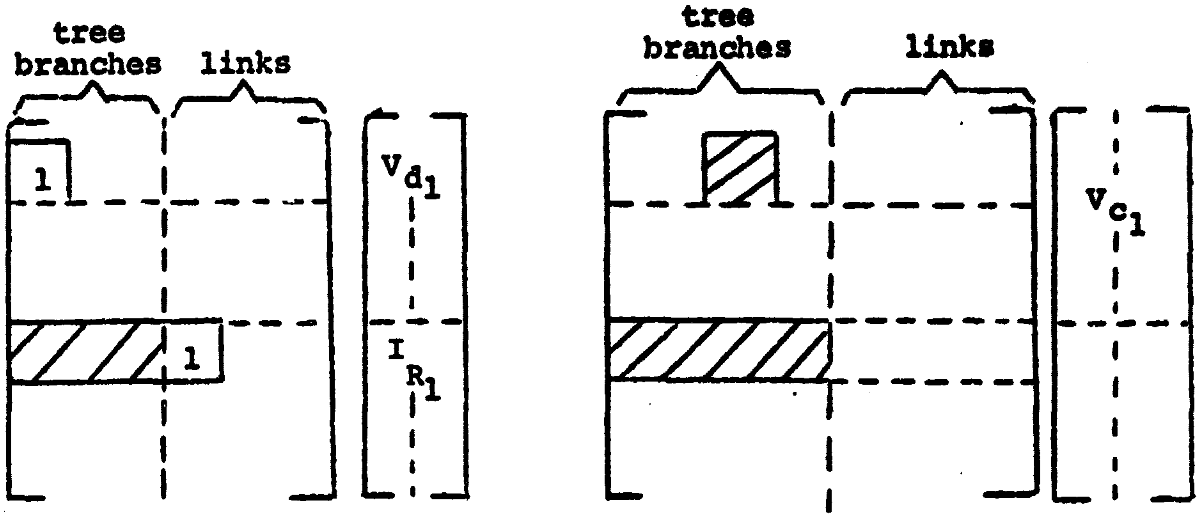

Fig. 1 Formation of Algebraic Eq··ations

Once this set of equations has been formed, it is solved using the reduction to row echelon form mentioned above. This might also be accomplished using the LU algorithm [5], which is both accurate and efficient. Although in the passive case the matrix $\underset{\sim}{M}_1$ is positive definite, in the general active case singularity is a possibility, indicating the fact that one of the quantities being solved for is arbitrary. Such cases are not likely for physical circuits.

Preliminary State Equations

The constraint equations for energy storage elements (including mutual coupling) are now introduced:

$$\begin{bmatrix} \underset{\sim}{C}_1 \\ \hdashline \underset{\sim}{C}_2 \end{bmatrix} \begin{bmatrix} \underline{V}_C \\ \underline{v}_C \end{bmatrix} = \begin{bmatrix} \underline{i}_C \\ \hdashline \underline{I}_C \end{bmatrix} ; \quad \begin{bmatrix} \underset{\sim}{L}_1 \\ \hdashline \underset{\sim}{L}_2 \end{bmatrix} \begin{bmatrix} \underline{i}_L \\ \underline{I}_L \end{bmatrix} = \begin{bmatrix} \underline{v}_L \\ \underline{V}_L \end{bmatrix} \qquad (3)$$

The procedure here is to remove $\underline{i}_C$ and $\underline{v}_L$ using $\underset{\sim}{F}$ and $-\underset{\sim}{F}^T$, while removing dependence on resistive and controlled source quantities using the solution of the algebraic equations (2). $\underline{I}_C$ and $\underline{V}_L$ may then be removed in favor of state variable derivatives using the matrices $\underset{\sim}{C}_2$ and $\underset{\sim}{L}_1$. The final results appear in the form

$$\begin{bmatrix} \\ \underset{\sim}{S} \\ \\ \end{bmatrix} \begin{bmatrix} \dot{\underline{V}}_C \\ \dot{\underline{v}}_C \\ \dot{\underline{i}}_L \\ \dot{\underline{I}}_L \end{bmatrix} = \begin{bmatrix} \\ \underset{\sim}{A} \\ \\ \end{bmatrix} \begin{bmatrix} \underline{V}_C \\ \underline{v}_C \\ \underline{i}_L \\ \underline{I}_L \end{bmatrix} + \begin{bmatrix} \\ \underset{\sim}{B} \\ \\ \end{bmatrix} \begin{bmatrix} \underline{E} \\ \underline{J} \end{bmatrix} \qquad (4)$$

or $\quad \underset{\sim}{S}\,\dot{\underline{x}} = \underset{\sim}{A}\,\underline{x} + \underset{\sim}{B}\,\underline{u}$

It should be clear that branch quantities defined as outputs can be handled in an identical manner during these reductions, yielding a preliminary input-output-state equation

$$\underline{y} = \underset{\sim}{R}\,\dot{\underline{x}} + \underset{\sim}{C}\,\underline{x} + \underset{\sim}{D}\,\underline{u} \qquad (5)$$

<u>Final State Equations</u>

The preliminary state equations would be in final form if S were an identity matrix. Since this fact is generally not so, the procedure continues by reducing the partitioned matrix $\left[\underset{\sim}{S} \;\vdots\; \underset{\sim}{A} \;\vdots\; \underset{\sim}{B}\right]$ to row echelon form. Several possibilities now arise.

a) $\underset{\sim}{S}$ becomes an identity matrix. In this case $\underset{\sim}{S}$ is of full rank and the final state equations are obtained.

b) $\underset{\sim}{S}$ and $\underset{\sim}{A}$ contain at least one zero row. In this case the equation of the last row

$$\underline{b}_n^T \underline{u} = 0 \tag{6}$$

implies a dependence among the independent sources. No solution exists for such a network.

c) $\underset{\sim}{S}$ but not $\underset{\sim}{A}$ contains at least one zero row. In this case the last row expresses the equation

$$o = \underline{a}_n^T \underline{x} + \underline{b}_n^T \underline{u} \tag{7}$$

implying a dependence among the state variables. One may be chosen for elimination and the relation above used to remove a column from the $\underset{\sim}{A}$ and $\underset{\sim}{C}$ matrices. Differentiating the expression gives

$$0 = \underline{a}_n^T \underline{\dot{x}} + \underline{b}_n^T \underline{\dot{u}} \tag{8}$$

which is used to remove the corresponding element from $\underline{\dot{x}}$ and hence a column from $\underset{\sim}{R}$ and $\underset{\sim}{S}$. Derivatives of input signals may thus arise during state variable elimination. When these relations have been exhausted, a smaller matrix $\left[\underset{\sim}{S} \,\vdots\, \underset{\sim}{A} \,\vdots\, \underset{\sim}{B}\right]$ is again reduced to row echelon form and the process repeated until an identity matrix is obtained for $\underset{\sim}{S}$, or all state variables disappear.

In theory this reduction technique is a completely general one in that the algorithm starts with the maximum possible number of state variable candidates and can remove as many as required. In practice, numerical difficulties can arise, but are entirely concentrated in the row reduction of matrices to row echelon form. The procedure requires an algorithm for deciding the rank of a matrix in the presence of round-off noise. This problem is a fundamental one in numerical analysis which has yet to be solved to the satisfaction of all. All network analysis techniques have this requirement built in in some form, since in any case the order of complexity of the network must be determined. The algorithm under discussion here presents the problem in precisely the form which is being studied - as a rank determination problem.

2.2 FROM STATE EQUATIONS TO CRITICAL FREQUENCIES

The designer of a network analysis scheme is at this point faced with the problem of deciding for just what purposes the program will be used. Time and frequency response data seem to be the most often requested; there are, however, users who for one reason or another require the critical frequencies of the network. For this reason, and because it turns out that it can be done accurately without great expenditure of effort, the program proceeds to find the natural modes of the network together with the zeros of transmission between all possible inputs and outputs.

One way of locating the natural modes of the network is to apply an eigenvalue-finding algorithm directly to the $\underset{\sim}{A}$ matrix. Numerical analysts seem to be in rare unanimity that Francis' QR algorithm is the best available [6] , [7] and this algorithm is employed by CORNAP. The remaining problem is then to determine the zeros. In order to use the QR algorithm for this purpose also, the problem of how to construct a matrix whose eigenvalues are the zeros of transmission of a network must be solved. The answer in this case lies in considering the "inverse system", a concept which has been explored in depth by Brockett [8] .

The Inverse System

Consider a network with one input and one output connected in the feedback of an operational amplifier of gain $k > 0$ as shown in Fig. 2.

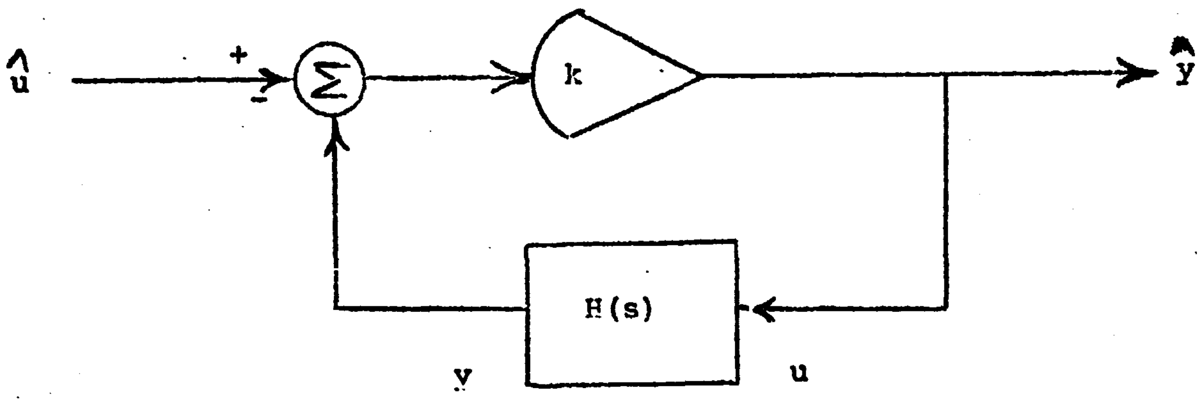

Fig. 2 Network in Feedback Loop of Operational Amplifier

If the system function of the original network is H(s), the system function H(s) of the modified network ("inverse system") will be

$$H(s) = \frac{k}{1 + kH(s)} \tag{9}$$

In the limit as $k \to \infty$, H(s) has the zeros of H(s) as its poles, and the poles of H(s) as its zeros. If the A matrix of the inverse system can be found without excessive effort a solution to our problem will have been obtained, since its eigenvalues will be the zeros of H(s) we are seeking. Fortunately, the machinery for obtaining this matrix is already available.

State Equations of the Inverse System

As can be seen from Fig. 2, from the original preliminary state and output-state equations of a single-input, single-output system

$$\underset{\sim}{S}\,\dot{\underline{x}} = \underset{\sim}{A}\,\underline{x} + \underline{b}\,u \tag{10}$$

$$y = \underline{r}^T\,\dot{\underline{x}} + \underline{c}^T\,\underline{x} + d\,u \tag{11}$$

we may by substitution obtain the preliminary state equations of the inverse system. Since

$$\hat{y} = u = k(\hat{u} - y) \tag{12}$$

$$= k\hat{u} - k\underline{r}^T\,\dot{\underline{x}} - k\underline{c}^T\underline{x} - kdu$$

or

$$u = \frac{k}{1 + kd}\,(\hat{u} - \underline{r}^T\dot{\underline{x}} - \underline{c}^T\underline{x}) \tag{13}$$

we obtain by substitution for u in (10)

$$\left[\underset{\sim}{S} + \frac{k}{1 + kd}\,\underline{b}\underline{r}^T\right]\,\dot{\underline{x}} = \left[A - \frac{k}{1 + kd}\,\underline{b}\,\underline{c}^T\right]\,\underline{x} + \left[\frac{k}{1 + kd}\,\underline{b}\right]\hat{u} \tag{14}$$

Before considering how to proceed to the limit $k \to \infty$, suppose first that the matrix $\left[\underset{\sim}{S} \,\middle|\, \underset{\sim}{A} \,\middle|\, \underline{b}\right]$ has by row operations been reduced so that $\underline{b}$ is a column vector with only one non-zero entry, the last (b_n). The preliminary state equations for the inverse system are now no different than for the original, with the exception of the last row which becomes

$$(\underset{\sim}{s}_n^T + \frac{k\,b_n}{1 + kd}\,\underline{r}^T)\dot{\underline{x}} = (\underset{\sim}{a}_n^T - \frac{k\,b_n}{1 + kd}\,\underline{c}^T)\,\underline{x} + \frac{k\,b_n}{1 + kd}\,\hat{u}\,. \tag{15}$$

Two cases may now be recognized:

Cases 1 ($d \neq 0$): In passing to the limit $k \to \infty$, the new last row of the state matrices become

$$\begin{array}{lll} \underset{\sim}{S}: & \underline{s}_n^T + \frac{b_n}{d}\underline{r}^T & \\ \underset{\sim}{A}: & \underline{a}_n^T - \frac{b_n}{d}\underline{c}^T & \qquad (16) \\ \underline{b}: & \frac{b_n}{d} & \end{array}$$

Case 2 ($d = 0$): Dividing the last row by k and passing to the limit, we obtain the last row

$$\begin{array}{lll} \underset{\sim}{S}: & b_n\underline{r}^T & \\ \underset{\sim}{A}: & -b_n\underline{c}^T & \qquad (17) \\ \underline{b}: & b_n & \end{array}$$

Having found preliminary state equations for the inverse system, we may proceed to use the mechanism already at hand for producing the final state equations with the correct number of state variables. The QR algorithm then finds the zeros of transmission of this particular input-output pair. The process must of course be repeated for each input-output pair.

The Gain Constant

In addition to the poles and zeros of transfer functions which have been requested, some sort of gain constant set is required. These gain constants are most easily found simultaneously by evaluating

$$\underset{\sim}{H}(s) = \underset{\sim}{C} \left[s\underset{\sim}{I} - \underset{\sim}{A} \right]^{-1} \underset{\sim}{B} + \underset{\sim}{D} \tag{18}$$

at a single value of s. This value of s has been chosen as real (to avoid complex matrix inversion) and positive (to avoid network poles). A search is also made to insure that the value of s chosen is not too near a zero of transmission of one of the components of $\underset{\sim}{H}(s)$.

Remarks

In the usual case of proper systems with at most the same number of zeros as poles, the above operations may be performed on the final state equations ($S=\underset{\sim}{I}$) with some saving in processing time. An improper system with more zeros than poles would in this case require a way for the $\underset{\sim}{A}$-matrix to grow in size - no such algorithm is included.

The sequence of events in finding the zeros of a polynomial filter (there are, of course, none) is that a zero row appears at the bottom of $\underset{\sim}{S}$ at each reduction to row echelon form until at last no state variables are left.

Only the observable, controllable natural modes of a network turn up as poles of the transfer function H(s). Since the present method finds <u>all</u> the natural modes of the network, we must be certain that the uncontrollable or onobservable modes are also natural modes of the inverse system, and therefore appear as zeros of H(s) to accompany the poles which should not be present. That these modes are not altered in going to the inverse system may be seen by considering that the inverse system is obtained by operating on the input and output of the original system. Since these modes are either uncontrollable or unobservable, they cannot be altered by operations on both input and output.

2.3 FREQUENCY AND TRANSIENT RESPONSE CALCULATIONS

Once the decision has been made to obtain critical frequencies, conventional techniques such as direct evaluation of the transfer function at desired frequencies, and partial fraction expansion followed by evaluation in the time domain, are immediately available and are fairly general. No more general or accurate approach exists, in fact, for finding the frequency response, and therefore this simple approach is the one used in the overall program. A somewhat more general technique was incorporated for computing the time response, which is explained in detail elsewhere [9]. The method consists of breaking the system function into a cascade of second order systems as shown in Fig. 3. The state equations of each subsystem are solved exactly, while the

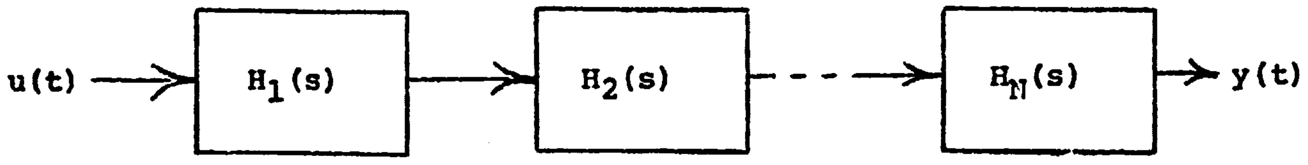

Fig. 3 Transfer Function as a Cascade of Subsystems

convolution integral between each subsystem is approximated by a fourth-order integration procedure. Sampled input signals are easily processed, and the method is completely insensitive to multiple or clustered poles.

2.4 CORNAP REFERENCES

[1] C. Pottle, "Comprehensive active network analysis by digital computer -- A state-space approach," Proc. Third Allerton Conf. on Circuit and System Theory (University of Illinois, Urbana, November 1965), pp.659-668.

[2] A. Dervisoglu, "State models of active RLC networks," Report R-237, Coordinated Science Laboratories, University of Illinois, December 1964.

[3] C. Pottle, "State-space techniques for general active network analysis," Chapter 3 of System Analysis by Digital Computer, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley, 1965, pp.59-98.

[4] D. A. Calahan, Computer-Aided Network Design. New York: McGraw-Hill, 1968, Chapter 2.

[5] J. H. Wilkinson, The Albebraic Eigenvalue Problem. Oxford: Clarendon Press, 1965, Chapter 4.

[6] Ibid, Chapter 8.

[7] B. N. Parlett, "The LU and QR algorithms," Chapter 5 of Mathematical Methods for Digital Computers, Vol. 2, A. Ralston and H. S. Wilf, Eds. New York: Wiley, 1967, pp.116-130.

[8] R. W. Brockett, "Poles, zeros, and feedback: State space interpretation," IEEE Trans. Automatic Control, Vol. AC-10, pp.129-134, April 1965.

[9] C. Pottle, "Rapid computer time response calculation for systems with arbitrary input signals," Proc. Fifth Allerton Conf. on Circuit and System Theory (University of Illinois, Urbana, October 1967), pp.523-533.

[10] C. Pottle, "A "Textbook" Computerized State Space Network Analysis Algorithm," University Report, System Theory Group, Electrical Engineering Research Laboratory, Cornell University, Ithaca, New York, September, 1968.

[11] C. Pottle, "Program CORNAP-FORTRAN Computer Routine," System Theory Group, School of Electrical Engineering, Cornell University, Ithaca, New York, 1968.

CHAPTER III

STIFFLY STABLE NUMERICAL INTEGRATION

3.1 INTRODUCTION

The output from the CORNAP circuit translation routines consists of the differential and algebraic equations of the circuit; namely, a <u>state equation</u>

$$\frac{d\bar{x}}{dt} = A_1\bar{x} + B_1\bar{u} + E_1\frac{d\bar{u}}{dt} \tag{3.1}$$

and an <u>output equation</u>[1]

$$\bar{y} = C_1\bar{x} + D_1\bar{u} \ . \tag{3.2}$$

Here

$\bar{x}$ is the s x 1 state vector of the circuit;

$\bar{u}$ is a T x 1 time varying vector whose components are the independent sources of the circuit;

$\bar{y}$ is a v x 1 vector whose components are the outputs requested by the user; and A_1, B_1, C_1, D_1, E_1 are constant matrices of dimensions consistent with the above equations (3.1 - 3.2).

If E_1 is nonzero, the change of variables

$$\bar{x} = \bar{q} + E_1\bar{u}$$

transforms the above equations to a form free of source derivatives·

---

[1]It is clear that the state and output equations for the general circuit could involve higher derivatives of the inputs; i.e.

$$\frac{d\bar{x}}{dt} = A_1\bar{x} + B_1\bar{u} + E_1\frac{d\bar{u}}{dt} + F_1\frac{d^2\bar{u}}{dt^2} + \ . \ . \ .$$

$$y = C_1\bar{x} + D_1\bar{u} + G_1\frac{d\bar{u}}{dt} + H_1\frac{d^2\bar{u}}{dt^2} + \ . \ . \ .$$

However, only equations of the form of (3.1 - 3.2) can be processed by STICAP, since no provisions are made to allow user input of derivatives of the independent sources.

$$\frac{d\bar{q}}{dt} = A\bar{q} + B\bar{u} \tag{3.3}$$

$$\bar{y} = C\bar{q} + D\bar{u} \,. \tag{3.4}$$

Here

$$A = A_1 \qquad\qquad C = C_1$$
$$B = A_1E_1 + B_1 \qquad\qquad D = C_1E_1 + D_1, \tag{3.5}$$

and $\bar{q}$ is the transformed state vector.

The problem of obtaining time domain circuit response is now reduced to finding a solution of eqns. (3.3 - 3.4). In the Gear mode this is accomplished by numerically integrating the initial value problem

$$\frac{d\bar{q}}{dt} = A\bar{q} + B\bar{u},\ \bar{q}(t_0) = \bar{q}_0. \tag{3.6}$$

Stiffly stable numerical integration techniques may be applied, if the system is stiff; otherwise, an Adam's integration algorithm may be selected. In this chapter the necessity of using stiff methods is indicated, together with a survey of the related theory.

3.2 ACCURACY AND STABILITY OF NUMERICAL INTEGRATION

Consider the scalar initial value problem

$$\frac{dz}{dt} = f(t,z),\ z(t_0) = z_0 \tag{3.7}$$

consisting of one first order ordinary differential equation, for which a solution is desired on the domain $t \geq t_0$, satisfying initially $z(t_0) = z_0$. In the absence of a closed form solution one settles for some form of a numerically approximate solution

$$\{y(t_j):\ t_j \geq t_0,\ j = 1,2,\ldots\},$$

with the set of time points t_j dense enough to yield information sufficient for the purposes at hand. The t_j often are chosen uniformly spaced, with $t_{n+1} = t_n + h$, where h is called the step size. The equations employed to generate this approximate solution may be called a numerical integration algorithm. A customary notation for the approximating solution values is $y(t_n) = y_n$, $t_n = t_0 + nh$.

Suppose the solution of (3.7) is expressed as

$$z_{n+1} = z_n + \int_{t_n}^{t_n+h} f[u, z(t_0, z_0, u)]du, \tag{3.8}$$

and let the integral in (3.8) be approximated by the trapezoidal rule

$$\int_a^{a+h} g(x)dx \approx \frac{h}{2}[g(a) + g(a+h)]. \tag{3.9}$$

Then we obtain the recursive numerical integration algorithm

$$y_{n+1} = y_n + \frac{h}{2}[f_n + f_{n+1}], \tag{3.10}$$

where $f_n = f(t_n, y_n)$, and y_n is a numerical approximation to z_n.

Given values y_n, f_n the next solution point, recursively obtained, is implicitly specified by (3.10). In contrast, the Euler algorithm

$$y_{n+1} = y_n + hf_n \tag{3.11}$$

explicitly specifies the next solution point. These algorithms provide respectively very simple examples of typical members from the classes of implicit and explicit linear multistep methods. Fundamental differences in character of these two algorithms will now be indicated. The phenomena discussed are also characteristic of more complicated linear multistep methods.[2]

The <u>accuracy</u> of a numerical integration algorithm is customarily measured in terms of its performance when applied to initial value problems (3.7) having as <u>exact</u> solutions the polynomials $\{1,t,t^2,\ldots,t^\ell\}$. If, when applied to problems whose solutions are members of the test set $\{1,t,t^2,\ldots,t^p\}$, and in the absence of roundoff error (i.e., on an infinite precision machine), the numerical and the exact solutions are identical, but where error appears in the numerical solution when the exact solution is t^{p+1}, the algorithm is said to be of <u>order p</u>. Due to the linearity, if the order is p, the linear multistep algorithm, whether implicit or explicit, yields an exact solution to the initial value problems whose time solution is an arbitrary polynomial of degree p or less, in the absence of computational errors in the initial conditions.

The order p of the algorithm dictates the magnitude of the stepsize h needed to produce a given degree of accuracy, when the algorithm is applied to a problem whose solution is not a polynomial of maximal degree p. This relation may be more explicitly stated by means of the <u>one step truncation error</u>, the error made in computing y_{n+1} when <u>exact</u> solution values of all required previous values y_{n-j}, f_{n-j} ($j = 0,1,\ldots,k-1$; for a k step algorithm) are known. For

---

[2]The general linear multistep method of k steps is of the form

$$\sum_{j=0}^{k} [\alpha_{k-j}y_{n-j} - h\beta_{k-j}f_{n-j}] = 0, \ \alpha_k \neq 0,$$

with the α_i,β_i real constants. The method is implicit if $\beta_k \neq 0$; otherwise, it is explicit. For implicit methods with nonlinear $f_n=f(t_n,y_n)$, some type of predictor corrector iteration must be used to solve for y_n, given values of $\{y_{n-1},y_{n-2},\ldots,y_{n-k}; f_{n-1},f_{n-2},\ldots,f_{n-k}\}$. The linear one step methods typified by (3.10) and (3.11) are considered a subclass of the general family.

an algorithm whose order is p the one step truncation error is of the form [1]

$$T = C_{p+1}h^{p+1}z^{(p+1)} + 0(h^{p+2}), \tag{3.12}$$

where C_{p+1} is a constant and $0(h^{p+2})$ indicates a function which approaches zero as h approaches zero in the same manner as does h^{p+2}. Thus, the accuracy of the algorithm, as well as its economy in terms of stepsize magnitude for a given degree of precision, increases directly with the integer p.

It is clear that the explicit Euler method (3.11) is of order p = 1, and it is readily verified that the trapezoidal method (3.10) is of order p = 2.

We now examine another characteristic phenomena of these algorithms; namely, their numerical stability, or as the statistician might say, robustness under variations in the step size. We shall consider as test system the initial value problem

$$\frac{dz}{dt} = \lambda z, \; z(t_0) = z_0 \tag{3.13}$$

where λ is allowed to be a real or <u>complex</u> number.[3]

Here the exact solution is

$$z = z_0e^{\lambda(t-t_0)}; \tag{3.14}$$

the desired sequence to be produced by numerical integration is the set of complex numbers

$$z_n = z_0e^{n\lambda h}, \; (t_n = t_0+nh).$$

However, the Euler algorithm produces the sequence

$$y_n = (1+h\lambda)^n \, z_0,$$

[3]Curiosity concerning the permissiveness of complex values of λ will be dispelled in the next section.

and the trapezoidal rule the sequence

$$\bar{y}_n = \left(\frac{1 + h/2\ \lambda}{1 - h/2\ \lambda}\right)^n z_0.$$

In the limit as $n \to \infty$ and $h \to 0$ with the product $nh = t_n - t_0$ held constant, both sequences converge to the true solution (3.14).[4] However, for h fixed, as $n \to \infty$ the Euler sequence diverges, regardless of the size of h, whenever $|1 + h\lambda| > 1$, while the trapezoidal sequence diverges if and only if $Re(h\lambda) > 0$. This phenomena is referred to as numerical instability; when it occurs the accuracy of the computation rapidly degenerates.

For the general linear multistep method, the degree of the difference equation usually exceeds that of the differential equation it is intended to approximate. Thus, certain parasitic solutions are present, in addition to the desired solution. When excited by roundoff or other computational error, these parasitic solutions tend to spoil the computation, in proportion to the degree that their rate of growth exceeds that of the desired solution.

A stability requirement which prevents the spoiling of the computation by excitation of parasitic solutions will now be defined: A linear multistep method is absolutely stable if the numerical solution approaches zero for h fixed, as n approaches infinity, when applied to all initial value problems (3.13) whose eigenvalue λ has negative real part; $Re(h\lambda) < 0$. Should this phenomena occur only when $h\lambda$ is restricted to some proper subset D of the left half plane, the algorithm is said to be D-strongly stable, or absolutely stable on a restricted domain.

[4]The numerical solution converges pointwise to the exact solution, as h approaches zero. An algorithm characterized by this property is said to be a convergent algorithm [1].

The $h\lambda$ plane regions of stability and instability of each algorithm (3.10) and (3.11) are indicated in Figure 1.

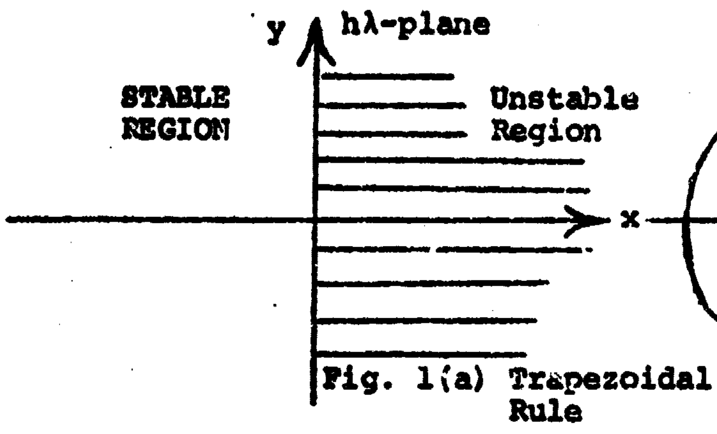

Fig. 1(a) Trapezoidal Rule

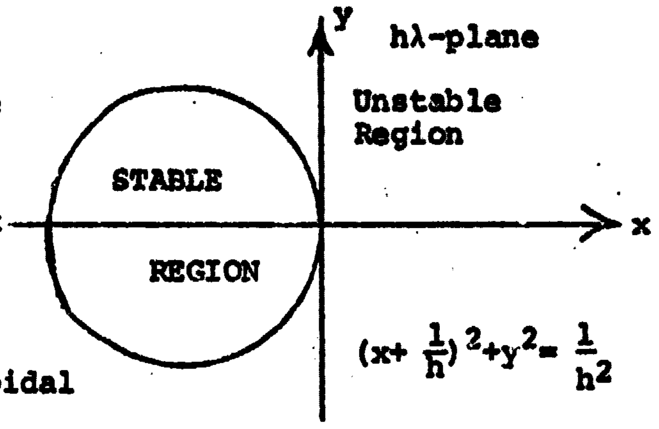

Fig. 1(b) Euler Method

Figure 1

The trapezoidal rule is absolutely stable, whereas the region of stability for the Euler method is the interior of the circle centered at $(-\frac{1}{h}, 0)$ with radius $R = \frac{1}{h}$. The Euler method is strongly stable on the interior of the circle. For this method the stability requirements may be expressed in terms of stepsize by the relations

$$h < \frac{2}{|\lambda|}\ , \ \mathrm{Re}(h\lambda) < 0.$$

Hence the stepsize h is very restricted by the modulus of the eigenvalue λ of the system being integrated.

We now make a fundamental observation, exemplified by the preceding examples. Namely, for all convergent explicit linear multistep methods, the $h\lambda$ plane stability region is bounded and has the origin as either an interior point or a boundary point. This observation also holds true for some implicit algorithms, the exception being the stiffly stable algorithms of Gear [2], [3]; and some instances of algorithms of other types, such as the absolutely stable Rosenbrock rule reported by Calahan [4]. Hence, for many commonly used techniques (Runge-Kutta, Adam's-Bashforth, etc.), there is a restriction on the stepsize, to avoid instability, usually of the form

$$0 < h < \frac{A}{|\lambda_{max}|}, \text{ A a positive constant,}$$

where λ_{max} is the largest eigenvalue of the Jacobian matrix of the system. In the case of stiff circuits this restriction is particularly hard to abide, since the step size is determined by the system component with minimum time constant, which in the large contributes least to the solution.

For the general stiff algorithm the stability region is unbounded in the left half plane and possesses a corridor of stable approach to the origin (see Fig. 2).

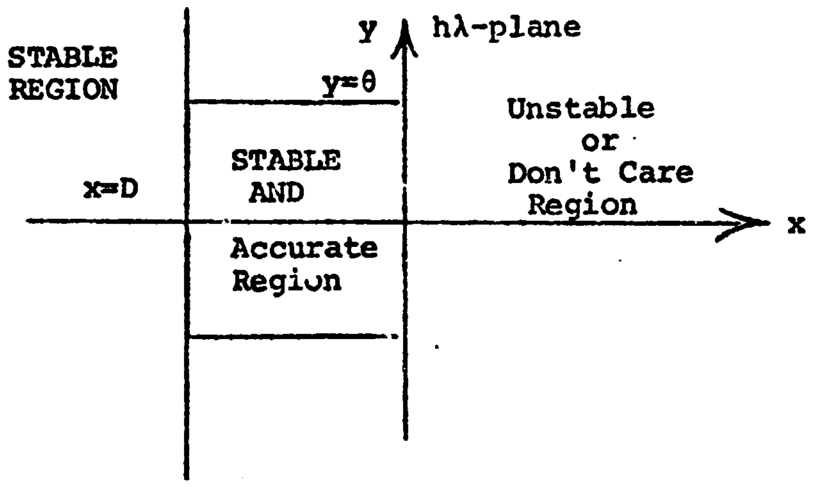

Figure 2. Stiff Stability Requirements

In the part of the hλ plane to left of the line $x = D$, corresponding to occurrence of system eigenvalues related to the high frequency components, the stiff algorithm is required to be stable. In the region bounded by the lines $x = D$, $x = 0$ and $y = \pm\theta$, the algorithm must be both stable and accurate; the rest of the plane is a "Don't

Care" region and can be stable or unstable. Hence stiff stability requires an algorithm which is numerically stable in the region of $h\lambda$ corresponding to rapidly decaying system components of diminishing significance, and which is both accurate and stable in the region corresponding to reasonably large step sizes and less slowly decaying system components.

Thus, for physically stable systems, the stability of the stiff algorithm is unaffected by the presence of left half plane eigenvalues of large modulus; for eigenvalues of other types, stability can be achieved by adjustments in step size (see Fig. 3 for further clarification of this statement). The off shoot of such methods is that the character of the computation is no longer so utterly restricted by the largest eigenmode; when high frequency effects are of diminished importance, the integration can proceed using a time step determined only by the needs of accuracy. Economical computation times may be achieved by suitably varying the stepsize h and the order p (by changing from one stiff algorithm to another) as the integration proceeds, so as to maximize the step size while simultaneously maintaining stability and a desired level of accuracy specified by the user [9].

The tradeoffs which can be made between the competing factors of stability and accuracy, for the stiff algorithms programmed in Algorithm 407, are indicated by Figure 3. Typical $h\lambda$ plane stability regions are indicated, for Gear's algorithms of orders p=2,3,4, 5,6. In these figures the regions of stability and instability are symmetric with respect to the axis of the reals; only the left half plane portions are of any significance, assuming physically stable

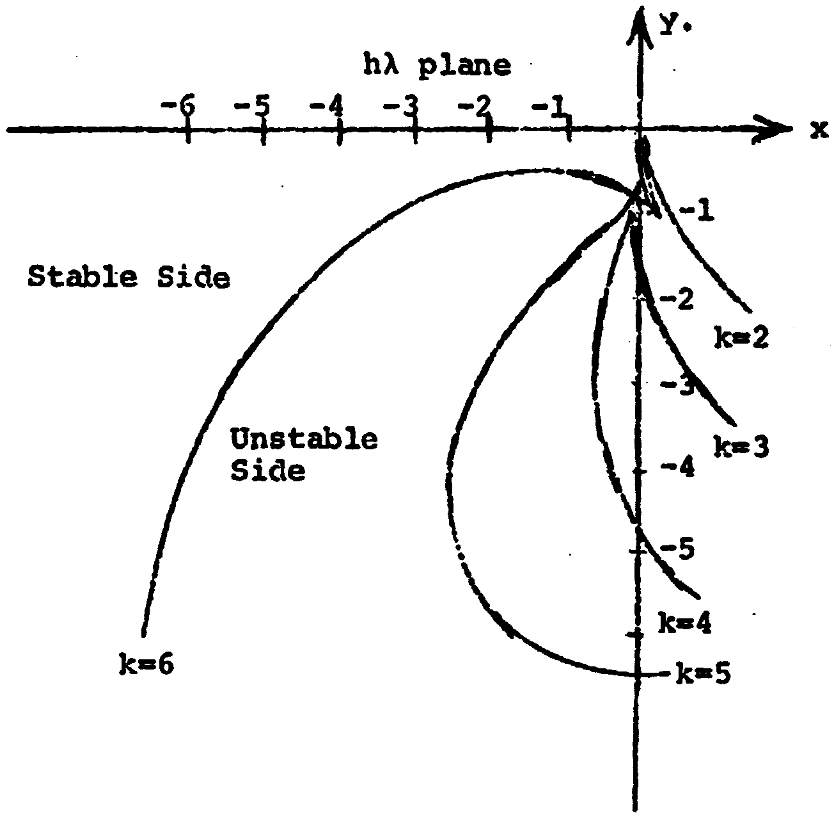

Figure 3. Sections of stable regions. Curves symmetric with respect to the x-axis.

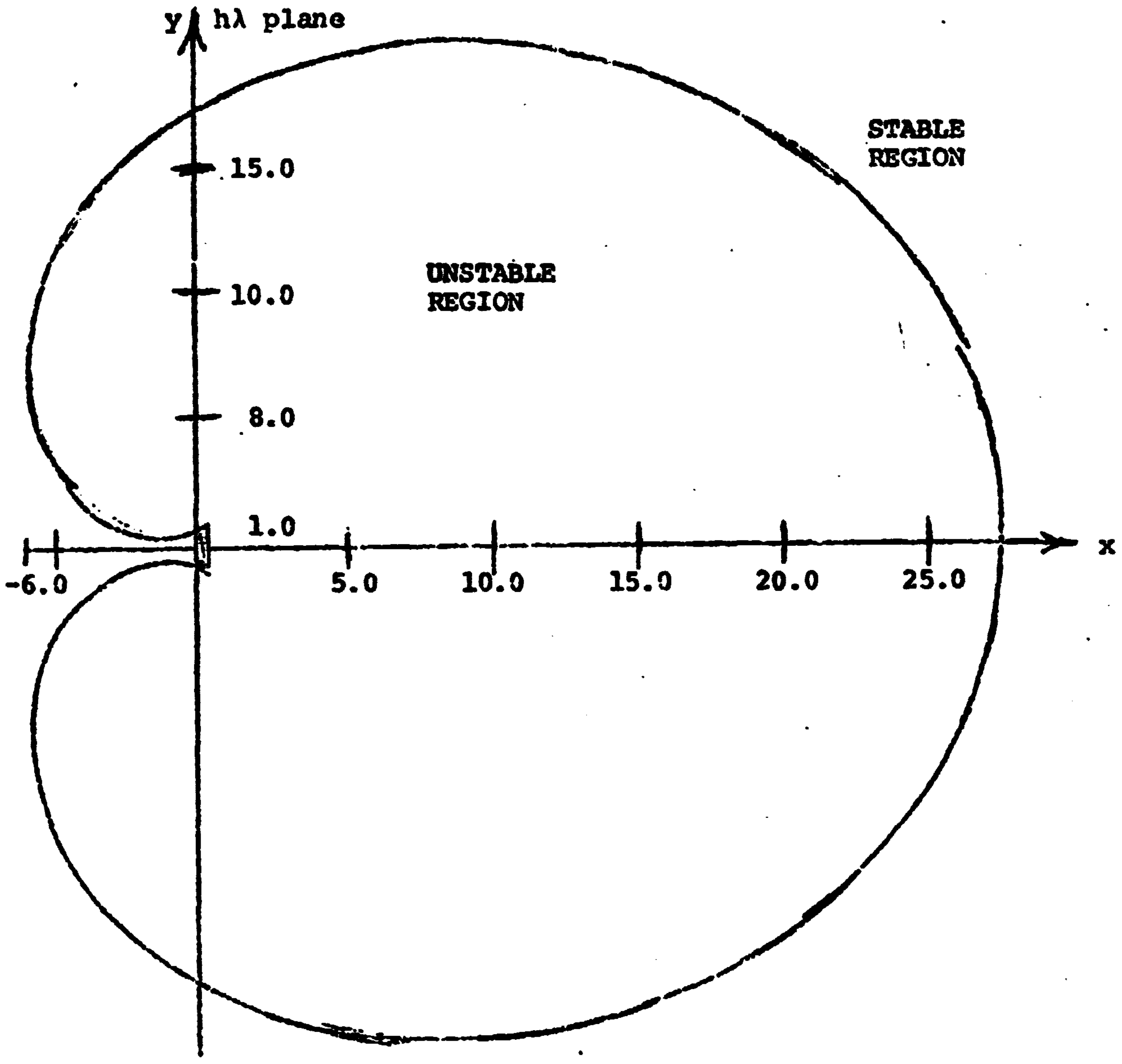

Figure 4. STABILITY Region
$p = k = 6$

networks. All regions of instability for the lower order methods are contained within the region of instability of the order $p = 6$ algorithm (see Figure 4). These plots also appear in references [3], [7].

3.3 THE NUMERICAL TREATMENT OF A VECTOR SYSTEM

A study of the previous section presumably arouses curiosity concerning the permissiveness of complex λ in the test equations (3.13). The dispellation of this curiosity is readily achieved when one considers the application of an algorithm for solution of a scalar equation to a vector system of simultaneous equations. We now provide an example, again using the trapezoidal rule.

Suppose the numerical integration of equation (3.4)

$$\frac{d\bar{q}}{dt} = A\bar{q} + B\bar{u},\ \bar{q}(t_0) = \bar{q}_0,$$

were to be accomplished by the vector trapezoidal rule, a parallel application of equation (3.10); namely,

$$\bar{y}_{n+1} = \bar{y}_n + \frac{h}{2}(\bar{f}_{n+1} + \bar{f}_n),\ \bar{y}_0 = \bar{q}_0. \tag{3.15}$$

In this instance the $s \times 1$ vectors $\bar{f}_n$ are defined by

$$\bar{f}_n = A\bar{q}_n + B\bar{u}_n. \tag{3.16}$$

This further reduces (3.15) to the form

$$(I - \frac{h}{2}A)\bar{y}_{n+1} = (I + \frac{h}{2}A)\bar{y}_n + \frac{hB}{2}[\bar{u}_{n+1} + \bar{u}_n], \tag{3.17}$$

where I is the identity matrix, $s \times s$.

Consider for purposes of illustration the case in which the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_s$ of A are distinct. Then there exists a nonsingular matrix Q for which

$$D = Q^{-1} AQ = \mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_s)$$

is a diagonal matrix, whose diagonal elements are the eigenvalues of A. Then under the transformation

$$\bar{y}_n = Qz_n$$

the algorithm (3.17) becomes

$$(I - \frac{h}{2}D)\bar{z}_{n+1} = (I + \frac{hD}{2})\bar{z}_n + \frac{h}{2}Q^{-1}B[\bar{u}_{n+1} + \bar{u}_n]. \qquad (3.18)$$

Thus the transformation uncouples the algorithm, so that there now arises a system of s simultaneous equations, $i = 1,2,\ldots,s$,

$$(1 - \frac{h}{2}\lambda_i)P_{i,n+1} = (1 + \frac{h}{2}\lambda_i)P_{i,n}+\frac{h}{2}g_n, \qquad (3.19)$$

where

$$g_n = \sum_j a_{ij}[u_{j,n+1} + u_{j,n}].$$

Here

λ_i is an eigenvalue of A,

$P_{i,n}$ is the ith component of $\bar{z}_n$,

$u_{j,n}$ is the jth independent source, evaluated at time n,

the a_{ij} are the elements of $Q^{-1}B$, and

the summation on j is over all independent sources.

Now if (3.19) is to be stable for all bounded independent source inputs, it must in particular be stable when these sources are switched off; i.e., the natural modes of the algorithm must be stable. However, setting $g_n = 0$ in (3.19), it emerges that a necessity and sufficient condition for stability is that for each $i = 1,2,3,\ldots,5$, $\mathrm{Re}(h\lambda_i)$ be negative. But this is the same criterion as was established for the scalar case. Moreover, the result is equally valid for occurrence of repeated eigenvalues, as can readily be established.

3.4 THE STIFF ALGORITHMS OF GEAR

The general form of the vector implicit linear multistep algorithm for solution of equation (3.6),

$$\frac{d\bar{q}}{dt} = A\bar{q} + B\bar{u},\ \bar{q}(t_0) = \bar{q}_0, \tag{3.20}$$

is given by the relation

$$\alpha_k\bar{y}_{n+k} + \alpha_{k-1}\bar{y}_{n+k-1} + \dots \alpha_0\bar{y}_n = h[\beta_k\bar{f}_{n+k} + \dots + \beta_0\bar{f}_n]. \tag{3.21}$$

Here $\alpha_k\beta_k \neq 0$, the functions $\bar{f}_{n+j}$ satisfy[5]

$$\bar{f}_{n+j} = A\bar{y}_{n+j} + B\bar{u}_{n+j},\ j = 0,1,2,\dots,k$$

and $\bar{y}_n$ is the numerical approximation of $\bar{q}_n$.

The algorithm is of order p if the relations $C_0=C_1=\dots=C_p=0$, $C_{p+1} \neq 0$, are satisfied, where [1]

$$\begin{aligned}
C_0 &= \alpha_0 + \alpha_1 + \dots + \alpha_k \\
C_1 &= \alpha_1 + 2\alpha_2 + \dots + k\alpha_k - (\beta_0 + \beta_1 + \dots + \beta_k) \\
&\cdot \\
&\cdot \\
&\cdot \\
C_q &= \frac{1}{q!}[\alpha_1+2^q\alpha_2+\dots+k^q\alpha_k] - \frac{1}{(q-1)!}[\beta_1+2^{q-1}\beta_2+\dots+k^{q-1}\beta_k].
\end{aligned} \tag{3.22}$$

Once having restricted the α's and β's to produce an algorithm of order p, it emerges that the β values are uniquely specified in terms of the α values by eqns. (3.22). Conversely, the additional specification that p = k allows a unique specification of the α values in terms of the β values. In either case, for a fixed order

[5]In this section the theory of linear multistep algorithms, specialized to the case of a linear dynamical system, is given. The general theory, as indicated in the literature tabulated in the bibliography, applies equally well to the nonlinear case.

p, the arbitrary parameters may then be chosen to obtain satisfactory stability properties. Criteria upon which such a value judgment may be based will now be indicated.

The numerical stability of the algorithm, as defined in the previous sections, is governed by the following theorem [2] [5]:

<u>Stability Theorem</u>

The numerical solution of (3.21) is stable if and only if the stepsize h is restricted so that the roots of the polynomial equations

$$\sum_{j=0}^{k} (\alpha_{k-j} - h\lambda_i \beta_{k-j})\rho^{k-j} = 0, \qquad (3.23)$$

(with λ_i, $i = 1,2,\ldots,s$, the eigenvalues of the matrix A) are within the unit circle in the complex plane, or else on the unit circle and not repeated.

In view of the foregoing discussion, it appears that in order to design the optimal k-step algorithm one should maximize simultaneously the integer p (for accuracy) and the region of the complex hλ plane which yields a stable root configuration for <u>each</u> of eqns. (3.23). However, a theorem of Dahlquist [1] concerned with convergence restricts the maximal p, for k fixed, to either k + 1 or k + 2, depending upon whether k is cdd or even. Fixing p at its maximal value, one next seeks the parameter choice yielding the broadest range of stability. Assuming a physically stable differential equation, absolute stability of the difference equation is the natural criterion, for broad general application. Unfortunately, again a result of Dahlquist [5] limits k to the value $k = 2$, $p \leq 2$, for absolute stability. However, the low order involved here implies an algorithm of limited usefulness. Thus the motivation for discovery of the stiff methods arises; methods not quite absolutely stable, but strongly stable on a restricted domain of the left half plane, with the characteristic corridor of stable approach to the origin (Fig. 2).

For stiff algorithms, a result of Widland [6] imposes the restriction $p \leq k$, for $k > 1$; $p = 2$, for $k = 1$. Specific stiff algorithms with $p = k$, $k = 1$ (1) 8 have been discovered by Gear [3], [7]; the general question is still the subject of research.

The algorithms discovered by Gear, with $p = k$; $k = 1,2,3,4,5,6$ are implemented in ALGORITHM 407. These algorithms are of the form

$$\bar{y}_{n+1} = \alpha_{k-1}\bar{y}_n + \alpha_{k-2}\bar{y}_{n-1} + \ldots + \alpha_0\bar{y}_{n+1-k} + h\beta_k\bar{f}_{n+1}, \qquad (3.24)$$

and require, in the programmed formulation [7], the fewest function evaluations and saving of least information from step to step of all stiff algorithms in the class $p = k$. The values of the α's and β's are given in Table I. A sample plot of the stability region, for $k = 6$, is given in Fig. (4). The stability regions for each algorithm appear in Fig. (3) (the stability plots are symmetric with respect to the real axis).

Table I. Coefficients of Stiff Algorithms

k	β_0	α_1	α_2	α_3	α_4	α_5	α_6
2	2/3	4/3	-1/3	0	0	0	0
3	6/11	18/11	-9/11	2/11	0	0	0
4	12/25	48/25	-36/25	16/25	-3/25	0	0
5	60/137	300/137	-300/137	200/137	-75/137	12/137	0
6	60/147	360/147	-450/147	400/147	-225/147	72/147	-10/147

These algorithms are obtained as follows: Equations (3.23) define a stability mapping [8]

$$H = \frac{\alpha_k \rho^k + \dots + \alpha_0}{\beta_k \rho^k + \dots + \beta_0} \tag{3.25}$$

of the complex root plane (the ρ plane) onto the $H = h\lambda$ plane. As h approaches infinity the roots of (3.23) approach the poles of H; as h approaches zero the roots of (3.23) approach the zeroes of H. Since the roots of a polynomial vary continuously with its coefficients, the choice of the poles of H to be strictly within the unit circle assures that each $h\lambda$ point in some neighborhood of infinity will be stable. By continuity, this stability region can be extended so that at least each point exterior to the image of the unit circle $|\rho| = 1$ is a stable point. (Moreover, when H is one-to-one on the unit circle with poles inside this image locus separates the stable and unstable regions [8] of the $h\lambda$ plane.) The unique stability mapping obtained by choosing all poles of H at the origin and determining the zeroes by the requirement that the order of the corresponding algorithm be $p = k$ yields stiffly stable algorithms, for $k = 1,2,3,4,5,6$, characterized by the stability regions of Figure 4.

3.5 PROGRAMMED FORMULATION

In the preceding sections we have presented the general theory underlying the linear multistep integration techniques exploited by Gear, and indicated the basic algorithms implemented in ALGORITHM 407. The mathematical formulation of these stiff methods and the Adam's-Bashforth methods for non-stiff equations actually programmed will not be discussed here.[6] The interested reader may refer to references [9], [10] for a detailed discussion of the mathematical formulation employed, the error controls used, and criterion for varying the stepsize and order of the methods.

As indicated in the preceding discussion, the stiff system theory applies equally well to nonlinear systems, and ALGORITHM 407 is programmed for such. Hence it might be argued that the predictor-corrector techniques used for solving equation (3.21) iteratively for $\bar{y}_{n+k}$ in terms of preceding values could perhaps be inefficient when applied to a linear system. However this conjecture is not justified; in the linear case the Newton iteration involved converges with only one iteration [9]. Hence the process reduces to a matrix inversion at each step, which cannot be avoided, regardless of the linearity, if a linear multistep method is assumed. Furthermore, in its programmed version the mathematical formulation automatically provides the information needed for determining the necessity of a change in stepsize or order of the method, an essential feature of any numerical integration technique which is to be effective over a broad range of network analysis problems.

[6]We remark that the formulation employed is chosen for the facility with which it lends itself to stepsize and order changing.

3.6 GEAR REFERENCES

1. Henrici, P., Discrete Variable Methods in Ordinary Differential Equations, John Wiley, New York, 1962.

2. Gear, C. W., "Numerical Integration of Stiff Ordinary Differential Equations," Report #221, Department of Computer Science, University of Illinois, Urbana, January 20, 1967.

3. Gear, C. W., "The Automatic Integration of Stiff Ordinary Differential Equations," Proceedings, 1968 IFIPS Conference, A81-A85, North Holland Publishing Company, Amsterdam, 1969.

4. Calahan, D. A., "A Stable Accurate Method of Numerical Integration for Nonlinear Systems," IEEE Proceedings, April 1968, p. 744

5. Dahlquist, G. G., "A Special Stability Problem for Linear Multistep Methods," BIT 3 (1963), pp. 27-43.

6. Widlund, O. B., "A Note on Unconditionally Stable Linear Multistep Methods," BIT 7, 1967, pp. 65-70.

7. Gear, C. W. and Dill, C., "A Graphical Search for Stiffly Stable Methods for Ordinary Differential Equations," J. ACM, Vol. 18, No. 1, January, 1971.

8. Cooke, C. H., "On the Class of Stiffly Stable Implicit Linear Multistep Methods," SIAM J. Numerical Analysis, Vol. 9, No. 1, March, 1972.

9. Gear, C. W., "The Automatic Integration of Ordinary Differential Equations," Communications of the ACM, March, 1971, Vol. 14, No. 3, 176-179.

10. Gear, C. W., "Algorithm 407 - DIFSUB for Solution of Ordinary Differential Equations [D2]," Communications of the ACM, March, 1971, Vol. 14, No. 3, 185-190.

CHAPTER IV

THE MATRIX METHOD

4.1 INTRODUCTION

As indicated in Chapter 3, the differential and algebraic equations of a network processable by STICAP may be described in the form

$$\frac{dx}{dt} = Ax + Bu, \qquad x(t_0) = x_0 \tag{4.1}$$

$$y = cx - Du,$$

where x is the state vector (or transformed state vector); y is the vector of outputs; u is the vector of independent sources; A, B, C, D are constant matrices.

The solution of the state equation is

$$x = [\exp(A(t-t_0)]\; x_0 + \int_{t_0}^{t} \exp(A(t - \tau)\; v(\tau)\; d\tau, \tag{4.2}$$

where

$$v(t) = Bu(t).$$

One way to avoid the problems which arise in numerical integration of stiff circuits is to integrate (4.2) exactly for certain waveforms such as sinusoids and block pulses, which constitute many of the excitations encountered in electronic circuits.

This explicit integration can be carried out if the transition matrix exp(A(t - τ) can be expressed as a linear combination of the eigenmodes of the system. The eigenvalues of the A matrix are obtained by the QR transformation, the best method available. The particular expansion in terms of the eigenmodes essential for the STICAP matrix algorithm was studied by Kirchner [1]. In the present algorithm the expansion is limited to the case of non-degenerate modes; the general case is more complex, but still solvable.

4.2 EIGENMODE EXPANSION OF exp(Λt)

Consider the case for which the eigenvalues of A are distinct. For this case the transition matrix is expressible as

$$\exp(At) = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} A^{j-1} \exp(\lambda_i t) \tag{4.3}$$

The coefficients C_{ij} form a matrix which turns out to be the inverse of the Vandermonde matrix

$$C = [C_{ij}] = \begin{bmatrix} 1 & 1 & \cdot & \cdot & \cdot & 1 \\ \lambda_1 & \lambda_2 & \cdot & \cdot & \cdot & \lambda_n \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \cdot & \cdot & \cdot & \lambda_n^{n-1} \end{bmatrix}. \tag{4.4}$$

The Vandermonde inverse can be computed efficiently through the method developed by Kaufman [2],

$$C_{ij} = \frac{\sum_{k=0}^{n-j} a_k \lambda_i^{n-j-k}}{\prod_{\substack{k=1 \\ k \neq 1}}^{n} (\lambda_i - \lambda_k)} \tag{4.5}$$

where the a_k values are the coefficients of the characteristic equation

$$\underline{P}(\lambda) = a_o\lambda^n + a_1\lambda^{n-1} + \ldots + a_{n-1}\lambda + a_n \tag{4.6}$$

with $a_o = 1$

The a_k coefficients can be formed from the combination of the traces of A^k, $k = 1, \ldots, n$

$$a_k = -\frac{1}{k} \sum_{m=1}^{k} a_{k-m} T_m \tag{4.7}$$

$$T_m = \mathrm{tr} A^m = \sum_{\ell=1}^{n} (\lambda_\ell^m) \tag{4.8}$$

The total number of operations involved in forming the $\underset{\sim}{C}$ matrix is about 4.5 N^2 operations, which is considerably faster than Gauss elimination, for large N. It is very cumbersome to store the transition matrix as expressed by Equation (4.3). Since the matrix is always multiplied into a vector such as $\exp(A(t-\tau))\bar{v}$, then it would be convenient to consider the vectors

$$\exp(A(t-\tau))\bar{v} = \underset{\sim}{p}_1 \exp(\lambda_1(t-\tau)) + \ldots + \underset{\sim}{p}_n \exp(\lambda_n(t-\tau)) \tag{4.9}$$

where

$$\begin{aligned} \underset{\sim}{p}_j &= (C_{j1} I + C_{j2} A + \ldots + C_{jn} A^{n-1})\bar{v} \\ &= (C_{j1}\bar{Y}_1 + C_{j2}\bar{Y}_2 + \ldots + C_{jn}\bar{Y}_n) \end{aligned} \tag{4.10}$$

with the $\bar{Y}_{m+1}$ vector defined by

$$\bar{Y}_1 = \bar{v}$$

$$\bar{Y}_{m+1} = A\,\bar{Y}_{m-1} \tag{4.11}$$

This can be easily set up as a recursive process, if use is made to take advantage of the sparseness of the $\underset{\sim}{A}$ matrix so that the programming is more efficient.

The integral in Equation (4.2) can now be expressed as

$$\int_{t_o}^{t} \exp(A(t-\tau))\ \bar{v}\ (\tau)\ d\tau = \sum_{j=1}^{n} \int_{t_o}^{t} \bar{p}_j \exp(\lambda_j(t-\tau))\ d\tau \qquad (4.12)$$

For a finite number of excitation sources $\bar{p}_j$ can be written as

$$\bar{p}_j = \sum_{k=1}^{k=M} \bar{r}_k s_k(\tau) \qquad (4.13)$$

where $\bar{r}_k$ is a constant column vector, and Equation (4.12) becomes

$$\sum_{j=1}^{n} \sum_{k=1}^{M} \bar{r}_k \int_{t_o}^{t} \exp(\lambda_j(t-\tau))s_k(\tau)\ d\tau \qquad (4.14)$$

Notice the integrals in Equation (4.14) are now scalar quantities and these are nothing more than convolution integrals. Since the integral

$$\int_{t_o}^{t} f\ d\tau = \int_{o}^{t} f\ d\tau - \int_{o}^{t_o} f\ d\tau \qquad (4.15)$$

it is sufficient to examine the integral

$$\int_{t_o}^{t} f \exp(\lambda_j(t-\tau))s_k(\tau)\ d\tau \qquad (4.16)$$

from which Equation (4.15) can be computed. The Laplace transform of the above integral is

$$\frac{1}{p-\lambda_j} S(p) = \frac{1}{p-\lambda_j} \frac{n(p)}{d(p)} = \frac{a}{p-\lambda_j} + \frac{b(p)}{d(p)} \qquad (4.17)$$

The first term in the above equation is the natural mode in the circuit while the remainder term is the response due to the forcing function. If λ_j and the poles of the driving function are situated widely apart, en the ill-conditioned case comes up. In the present algor ., an exact explicit integration is performed for Equation (4.16), for certain excitation waveforms, thereby eliminating the difficulty. When there are complex eigenvalues, then the following integral pair is considered

$$\left\{\int_0^t \exp(\lambda_i(t-\tau))rs(\tau)\ d\tau + \int_0^t \exp(\lambda_i^*(t-\tau))r^*s(\tau)\ d\tau\right\} \qquad (4.18)$$

where r is an element in the column vector of Equation (4.14). Equation (4.18) is valid if the original state equations consist of real quantities only. For a full discussion of the algorithm mathematics concerning the exact integration of (4.14) for the functions STICAP allowable as excitations see [3], [4], [5].

4.3 MATRIX MODE REFERENCES

1. Kirchener, R. B., "An Explicit Formula for e^{At}," AMM, 1967, pp.1200-1204.

2. Kaufman, I. "Evaluation of an Analytical Function of a Companion Matrix with District Eigenvalues," Proceedings of IEEE Letters, 1959, pp.1180-1181.

3. Young, E. H., Ranson, M. N., and Heinbockel, J. H., "Numerical Solution of Linear State Equations with Large Eigenvalue Spreads," Proceedings, Fourth Asilomar Conference on Circuits and Systems, Santa Clara, California, 1970, pp.381-385.

4. Cooke, C. H., and Young, E. Y., "Numerical Integration and Other Techniques for Computer Aided Network Design Programming," Final Report, NGR-47-003-026, NASA CR-111837, January, 1971.

5. Ransom, M. N., "A Matrix Algorithm for the Solution of Linear Systems with Widely Separated Eigenvalues," M. S. Thesis, School of Engineering, Old Dominion University, May, 1971.