AD-A078 604 WIDENER UNIV CHESTER PA CENTER OF ENGINEERING F/G 9/3
EFFICIENT FAULT ANALYSIS IN LINEAR ANALOG CIRCUITS.(U)
OCT 79 A T JOHNSON AFOSR-79-0106
UNCLASSIFIED TR-79-03 AFOSR-TR-79-1270 NL

END
DATE
FILMED
1-80
DDC

AFOSR-TR- 79-1270

LEVEL II

AD A078604

DDC
RECEIVED
DEC 18 1979
E

DDC FILE COPY

79 12 18 38

Approved for public release;
distribution unlimited.

DISCLAIMER NOTICE

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

(13)

Approved for public release;
distribution unlimited.

EFFICIENT FAULT ANALYSIS IN LINEAR ANALOG CIRCUITS

Alfred T. Johnson, Jr.
Center of Engineering
Widener University
Chester, PA 19013

DDC
RECEIVED
DEC 18 1979
E

October 1979

Final Report: AFOSR(PKD), Bolling AFB, D.C. 20332

USAF Grant AFOSR-79-0106

Widener Technical Report - TR-79-03

Center of Engineering
Widener University
Chester, PA 19013

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is
approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR-79-1270	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): EFFICIENT FAULT ANALYSIS IN LINEAR ANALOG CIRCUITS	5. TYPE OF REPORT & PERIOD COVERED: Final Report, May 28,1979 to July 28,1979
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Alfred T. Johnson, Jr.	8. CONTRACT OR GRANT NUMBER(s): AFOSR-79-0106
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center of Engineering, Widener University, Chester, PA 19013	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 2305/D9
11. CONTROLLING OFFICE NAME AND ADDRESS: AFOSR/NE Bldg 410, Bolling AFB, D.C. 20032	12. REPORT DATE: 10 October 1979
	13. NUMBER OF PAGES: 77
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Final rept. 28 May – 28 Jul 79	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

TR-79-03

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Analog Electronic Fault Diagnosis
Analog Electronic Fault Analysis
Linear Circuit Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report addresses the problem of high-resolution fault diagnosis in linear, analog electrical circuits in which a limited number of terminals are accessible for measurement. An efficient algorithm is described for computing network responses after a single parameter is changed. The computational complexity of this algorithm has been found to be very superior to that of a conventional algorithm.

In addition, the problem of choosing a set of measurements for fault analysis

(over)

DD FORM 1473 411 503

Unclassified

Unclassified

20. Continued

has been studied. We have found necessary conditions for resolving the fault to the component level. The problems of measurement error and numerical conditioning have also been studied.

TABLE OF CONTENTS

Section		Page
1.	The Problem of Analog Fault Analysis	1
	1.1 Summary of Previous Work	2
2.	New Results in Analog Fault Analysis	4
	2.1 Parameter Changes	7
	2.1.1 Parameter Change is a Passive Admittance	9
	2.1.2 Parameter Change is a Transresistance	13
	2.1.3 Parameter Change is a Current Gain	14
	2.1.4 Parameter Change is a Voltage Gain	15
	2.1.5 Parameter Change is a Transconductance	15
	2.2 Finding Independent Measurements	17
	2.2.1 Computing the Jacobian Matrix	24
3.	Computational Complexity	26
4.	Conclusions	29
5.	Future Work	30
	Appendix 1 Computer Program Listings	32
	Appendix 2 Reprint of "Efficient Fault Analysis in Linear Analog Circuits", by A.T.Johnson, Jr.	64
	References	75

Accession For
NTIS GRA&I
DDC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or special
A 23
05

1. The Problem of Analog Fault Analysis

Analog fault analysis is a method of finding a description (either in terms of the physical analog system or its model) of the way in which an analog system has failed. A failure (or fault) may manifest itself in many different ways, depending on the nature of the system. In the present study we consider the system to be a linear, analog electronic circuit, with at least two accessible terminals. We treat the accessible terminals as a multi-port network. The behavior of the network can be determined by measuring each of the several independent (complex) multi-port parameters (transfer functions), possibly at several frequencies. Thus, if m/2 independent, complex measurements are made, the network performance is given in terms of m real numbers, each of which is a function of the parameters of the network. We denote the measurement set by the real vector $\underline{\hat{M}} \triangleq \text{col}\ (\hat{M}_i)$. Similarly, we denote the corresponding set of multi-port parameters computed from the network model by the real vector $\underline{M} \triangleq \text{col}\ (M_i)$.

We will assume there are n real network parameters, each of which is either a resistance, capacitance, inductance, or controlled source gain. We denote the parameter set by the real vector $\underline{x} \triangleq \text{col}\ (x_i)$.

The fault analysis problem may be stated as follows: given m real measurements and the network model containing n real parameters, find the values of the n real parameters which produce the m real measurements. This requires solving the system of m non-linear equations in n unknowns

$$\underline{M}\ (\underline{x}) = \underline{\hat{M}} \tag{1}$$

Currently available analog automatic test equipment (ATE) allows the preprogramming of a sequence of test signals and measurements [1]. The software interpretation of these measurements is mostly limited to a functional test of the component or module tested. To achieve greater resolution of the fault with currently available ATE (i.e., to find the faulty component or integrated circuit in the module), requires access to more terminals (test points) and the making of a more complex sequence of measurements.

Often in practice the number of terminals available for measurement is limited. For example, in troubleshooting existing equipment, available test points are fixed by the design. In the design of new equipment, it is desirable to keep the number of test points as small as possible to avoid an excessive number of pins in the connectors.

The present study was directed at improving the reliability of high resolution (down to the component or node if necessary) analog fault analysis from measurements at the accessible terminals. The measurements in this study require applying sinusoidal sources and finding the magnitude and phase relationships of the steady state responses, i.e., the measurements are a set of complex multi-port parameters at the accessible terminals. The fault is isolated by solving (1) for the n-dimensional parameter vector $\underline{x}$. Recent work in analog fault analysis has been directed toward finding an efficient way to solve these equations.

1.1 Summary of Previous Work

In solving (1), the number of network parameters may be greater than, equal to, or less than the number of measurements m. If $m < n$ it is not possible to solve for the parameter set $\underline{x}$, since in this case (1) usually has infinitely many solutions. Ransom and Saeks [2]

approached this problem by finding the solution to (1) which minimizes the norm $|| \underline{x} - \underline{x}_0 ||$, where $\underline{x}_0$ is the nominal parameter set. The difficulty with this approach is that the solution assumes, roughly speaking, that the most likely state of the element values in the network is the one which causes the smallest drift from the nominal values consistant with the measurements. This assumption excludes the possibility of catastrophic faults (open and short circuits).

If $m \geq n$, then the non-linear equations given by (1) are often solved using optimization techniques by finding $\min_{\underline{x}} || \underline{M}(\underline{x}) - \hat{\underline{M}} ||$, where sometimes the norm may be reduced to zero if $m = n$. This approach was taken by Chen and Saeks [3], using an efficient algorithm to evaluate $\underline{M}(\underline{x})$. In order to accommodate catastrophic faults (open and short circuits) by this method, each element in the parameter space must be searched over the range $(0, \infty)$, which is not practical because of the prohibitively large number of evaluations of the function $\underline{M}(\underline{x})$.

In this project we continue to seek a way to overcome the difficulties of the two methods described above, viz. that catastrophic faults are overlooked altogether or that they are found at the cost of very lengthy computation. The approach described in this report is to modify the parameter set $\underline{x}$ to include the set of all single short circuits involving inaccessible nodes. The remaining parameters are branch admittances or controlled source gains. Searching over short circuits eliminates the need to search over very large values of the branch admittances. It is also unnecessary to search over very large values of controlled source gains, since this mode of failure is extremely unlikely. Johnson [4], [5] has described a very efficient algorithm for evaluating the function $\underline{M}(\underline{x})$ in the presence of a short

circuit. The efficiency results from the fact that the network change caused by the short circuit results in an alteration of the (inaccessible) nodal admittance matrix by a matrix of unit rank. As a result, the response $\underline{M}(\underline{x})$ of the faulted network can be computed (in terms of the nominal response) using many fewer multiplications than are required for an original analysis. Sparse matrix techniques were also used to reduce computation time. The efficiency of this algorithm makes it practical to exhaustively search for a minimum of $||\ \underline{M}(\underline{x}) - \hat{\underline{M}}\ ||$ over all possible single short circuits involving an inaccessible node. The test examples described by Johnson [4], [5] show that short circuits can be reliably located by the algorithm, even though measurements were made only at one frequency, and parameter drift was not taken into account.

Performing the measurements (and analysis) at more than one frequency increases the number of measurements m without changing the number of network parameters n. This may improve the reliability of the fault analysis if all the measurements are independent. The problem of independent measurements is discussed in some detail by Sen and Saeks [6] - [8], who unfortunately stop short of describing exactly how to choose the measurements and frequencies.

2. New Results in Analog Fault Analysis

The study undertaken by the author was directed toward improving the reliability of analog fault analysis (isolating short circuits, open circuits and parameter changes) while keeping the algorithm as fast and efficient as possible. Specifically two separate studies were done:

(a) Introducing parameter changes into the current algorithm. The current algorithm searches for a minimum of $|| \underline{M}(x) - \hat{\underline{M}} ||$ over short circuits only. By searching over the network parameters (resistances, capacitances, inductances and controlled source gains) as well, the algorithm should be improved, because of the expanded description of the fault. In addition it was hoped that the algorithm would be made more reliable (i.e., it was hoped that the algorithm would not indicate faults which were not present, but instead would find only those that were present). An efficient algorithm for calculating the network responses $\underline{M}(\underline{x})$ in the presence of a parameter change was developed and included in the search for $\min_{\underline{x}} || \underline{M}(\underline{x}) - \hat{\underline{M}} ||$. This is described in detail in Section 2.1.

(b) Independent Measurements. Increasing the dimension m of $\hat{\underline{M}}$ (and also of $\underline{M}(\underline{x})$) by adding independent measurements should improve the accuracy and resolution of the fault analysis. In most practical cases, the number of accessible terminals is fixed, so the only way to increase the number of measurements is to use more than one frequency. It is important to determine if the set of measurements is independent. The work on this problem is described in detail in Section 2.3.

We have discussed how the network parameters may be determined by minimizing the norm of the difference between the measured and computed network responses $|| \underline{M}(\underline{x}) - \hat{\underline{M}} ||$, where $\underline{M}$ and $\hat{\underline{M}}$ are both real vectors of dimension m. In the discussion which follows, the network responses are considered to be y-parameters at the accessible terminals, all referred to a common (accessible) node. Figure 1 shows a network with p + 1 accessible terminals

from which we define the pxp y-parameter matrix

$$\underline{y} \triangleq \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & & & \\ y_{p1} & y_{p2} & \cdots & y_{pp} \end{bmatrix} \tag{2}$$

where, as usual,

$$y_{rs} \triangleq \left. \frac{I_s}{V_r} \right|_{V_k = 0,\ k \neq r} \tag{3}$$

In (3) I_s and V_r are the phasor response and source, respectively. The matrix $\underline{y}$ therefore represents p^2 complex

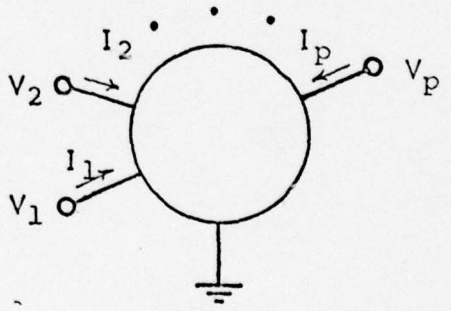


Figure 1 - p-Port Network with p+1 accessible terminals.

parameters or $2p^2$ real parameters. These $2p^2$ real parameters form the vector $\underline{M}$ of dimension m, i.e., $m \triangleq \dim \underline{M} = 2p^2$, an even integer. (The exact order in which the elements of vector $\underline{M}$ are formed from the elements of matrix $\underline{y}$ is not important. However, the measured response vector $\hat{\underline{M}}$ is formed

from the measured y-parameter matrix $\hat{\underline{y}}$ by the same rule used to form one computed response vector $\underline{M}$ from the computed y-parameter matrix $\underline{y}$).

We close this section by displaying a formula derived by Johnson [4], [5], which will be used in subsequent sections.

Johnson shows that the pxp matrix of measurable y-parameters defined above is given by

$$\underline{y} = \underline{A}_1 \underline{Y}_b [\underline{1} - \underline{A}_2^T (\underline{A}_2 \underline{Y}_b \underline{A}_2^T)^{-1} \underline{A}_2 \underline{Y}_b] \underline{A}^T \qquad (4)$$

where

$$\underline{A} = \left(\frac{\underline{A}_1}{\underline{A}_2} \right)$$

is the *node-to-branch incidence matrix* (as defined by and Kuh [9] partitioned into the accessible ($\underline{A}_1$) and the inaccessible ($\underline{A}_2$) nodes, and $\underline{Y}_b$ is the branch admittance matrix [4], [5], [10]. If the network contains b branches, $\underline{Y}_b$ is a sparse bxb matrix whose elements are simple functions of the branch admittances and the controlled source gains. It therefore plays the role of the parameter vector $\underline{x}$ and can be expressed in terms of $\underline{x}$.

2.1 Parameter Changes

In order to find $\min_{\underline{Y}_b} || \underline{y} (\underline{Y}_b) - \hat{\underline{y}} ||$ it is important to compute $\underline{y} (\underline{Y}_b)$ efficiently. Such an algorithm has been found if a change is made in only one parameter at a time. To derive this result, suppose $\underline{Y}_b$ is changed by the bxb matrix $\underline{\Delta}$. Then from (4) the y-parameter matrix becomes

$$\underline{Y}' = \underline{A}_1 (\underline{Y}_b+\underline{\Delta}) [\underline{1} - \underline{A}_2^T (\underline{A}_2(\underline{Y}_b+\underline{\Delta}) \underline{A}_2^T)^{-1} \underline{A}_2(\underline{Y}_b+\underline{\Delta})] \underline{A}^T \quad (5)$$

The major computational difficulty in evaluating the right side of (5) is computing the inverse of the new inaccessible nodal admittance matrix $\underline{Y}_2' = \underline{A}_2 (\underline{Y}_b+\underline{\Delta}) \underline{A}_2^T$. Define the qxq matrix $\underline{Y}_2 \triangleq \underline{A}_2\underline{Y}_b\underline{A}_2^T$, where q is the number of inacessible nodes. Thus $\underline{Y}_2' = \underline{Y}_2 + \underline{A}_2\underline{\Delta}\ \underline{A}_2^T$ and we wish to find the inverse of $\underline{Y}_2'$.

During this investigation we have discovered that if only one network parameter is changed, $\underline{\Delta}$ can be formed as the outer product of two vectors $\underline{u} \triangleq$ col (u_1) and $\underline{v} \triangleq$ col (v_i), i.e. $\underline{\Delta} = \underline{u}\underline{v}^T$. (In this case $\underline{\Delta}$ has unit rank). The specific way in which $\underline{u}$ and $\underline{v}$ are formed depend on whether the network parameter changed is a passive admittance or a controlled source. The several special cases are discussed in Sections 2.1.1 through 2.1.5, which the reader may skip without loss of continuity.

The product $\underline{A}_2\underline{\Delta}\ \underline{A}_2^T$ may therefore be expressed as the outer product $\underline{U}\underline{V}^T$, where $\underline{U} \triangleq \underline{A}_2\underline{u}$ and $\underline{V} \triangleq \underline{A}_2\underline{v}$ are both vectors of dimension q, where q is the number of inaccessible nodes. We may now write

$$\underline{Y}_2' = \underline{Y}_2 + \underline{U}\underline{V}^T. \quad (6)$$

Householder [11] shows that the inverse of $\underline{Y}_2'$ can be formed from

$$\underline{Y}_2'^{-1} = \underline{Y}_2^{-1} [\underline{1} - \underline{U}(1 + \underline{V}^T\underline{Y}_2^{-1}\underline{U})^{-1} \underline{V}^T \underline{Y}_2^{-1}] \quad (7)$$

In (7) the inverse of $\underline{Y}_2$ is known from an initial computation (in terms of the nominal parameter values) which is performed only once, and $1 + \underline{V}^T \underline{Y}_2^{-1} \underline{U}$ is a scaler. Thus the right side of (7) may be found without a matrix inversion.

The Fortran code for computing $\underline{Y}_2^{-1}$ using (7) has been written and may be found in Appendix 1 (SUBROUTINE Y2PI2). Sparse matrix techniques have been used to store the vectors $\underline{U}$ and $\underline{V}$.

The purpose of the next several sections (2.1.1 through 2.1.5 is to show that a change in any network parameter changes the branch admittance $\underline{Y}_b$ by a matrix of unit rank, i.e., $\underline{Y}_b' = \underline{Y}_b + \underline{\Delta}$, where $\underline{\Delta} = \underline{u}\,\underline{v}^T$, and $\underline{u} \triangleq \text{col}\ (u_i)$, $\underline{v} \triangleq \text{col}\ (v_i)$. An efficient computer program for forming the vectors $\underline{u}$ and $\underline{v}$ using sparse matrix techniques has been written, and the listing may be found in Appendix 1 (SUBROUTINE UVVEC).

Johnson and Pennington [10] have shown that in a network containing b branches, the bxb branch admittance matrix can be formed from the element matrices as follows:

$$\underline{Y}_b = (\underline{1} + \underline{Y}\,\underline{R}_m - \underline{\alpha})^{-1}\,(\underline{Y} - \underline{Y}\,\underline{M} + \underline{G}_m) \tag{8}$$

where $\underline{1}$ is the bxb identity matrix, $\underline{Y}$ is the bxb matrix diag (Y_i) and Y_i is the self-admittance of branch i; $\underline{R}_m$ is the bxb matrix $\{R_{ij}\}$, where R_{ij} is the transistance in branch j controlled by the current in branch i; $\underline{\alpha}$, $\underline{\mu}$, and $\underline{G}_m$ are defined similarly and contain branch coupling in the form of current gains, voltage gains, and transconductances respectively.

2.1.1 Parameter Change is a Passive Admittance†

If a change is made in branch admittance k, then $\underline{Y}$ becomes $\underline{Y} + \underline{\Delta}$, where $\underline{\Delta}$ is zero except for element k. A little algebra shows that $\underline{Y}_b$ has been changed to $\underline{Y}_b' = \underline{Y}_b + \underline{\Delta}$, where

†The reader may omit this section without loss of continuity.

$$\underline{\Delta} = \underline{W}\,(\Delta\underline{X}) - (\Delta\underline{W})\,\underline{X} - (\Delta\underline{W})\,(\Delta\underline{X}) \tag{9}$$

and

$$\underline{X} = \underline{Y} - \underline{Y}\,\underline{\mu} + \underline{G}_m$$

$$\underline{W} = \underline{1} - \underline{Y}\,\underline{R}_m + \underline{\alpha}$$

$$\Delta\underline{X} = \underline{\Delta}_y - \underline{\Delta}_y\,\underline{\mu}$$

$$\Delta\underline{W} = -\Delta_y\,\underline{R}_m$$

Equation (9) simplifies since $(\Delta\underline{W})\,(\Delta\underline{X}) = \underline{0}$, which may be shown as follows. After some algebra, using (9),

$$(\Delta\underline{W})\,(\Delta\underline{X}) = (\underline{\Delta}_y\underline{R}_m\,\underline{\Delta}_y)\,(\underline{\mu} - \underline{1}),$$

which is zero if $\underline{\Delta}_y\,\underline{R}_m\,\underline{\Delta}_y = \underline{0}$. Suppose branch admittance k is changed, and define $\underline{B} \triangleq \underline{R}_m\,\underline{\Delta}_y$. Therefore $B_{ij} = \sum_m (R_m)_{im}\,(\Delta_y)_{mj}$. But all elements of $\underline{\Delta}_y$ are zero except for element (k,k).

$$\begin{aligned} B_{ij} &= 0 \quad , \; j \neq k \\ B_{ik} &= (R_m)_{ik}\,(\Delta_y)_{kk} \quad , \; \forall\, i \end{aligned} \tag{10}$$

Now $\underline{\Delta}_y\underline{R}_m\,\underline{\Delta}_y = \underline{\Delta}_y\,\underline{B}$. Consider the elements of column k of this matrix:

$$(\underline{\Delta}_y\,\underline{R}_m\,\underline{\Delta}_y)_{rk} = \sum_i (\Delta_y)_{ri}\,B_{ik} \tag{11}$$

From (10) we see that $\underline{\Delta}_y\,\underline{R}_m\,\underline{\Delta}_y$ is zero except for elements in column k. Furthermore the summation need be carried out over element i = r only, since $(\Delta_y)_{ri} = 0$, $r \neq i$. Finally, $(\Delta_y)_{rr} = 0$ unless r = k, so $\underline{\Delta}_y\,\underline{R}_m\,\underline{\Delta}_y$ contains only one non-zero element, namely $(\underline{\Delta}_y\,\underline{R}_m\,\underline{\Delta}_y)_{kk} = (\Delta_y)_{kk}(R_m)_{kk}(\Delta_y)_{kk}$. But this last expression is zero because branch k never contains a transresistance controlled by branch k. (This is modelled instead as a simple self admittance). QED.

After deleting the last term in (9), the change in the branch admittance matrix $\underline{Y}$ due to a change in branch admittance k becomes

$$\underline{\Delta} = \underline{\Delta}_y - \underline{\Delta}_y \underline{\mu} + \hat{\underline{\alpha}} \underline{\Delta}_y - \hat{\underline{\alpha}} \underline{\Delta}_y \underline{\mu} + \underline{\Delta}_y \underline{R}_m \underline{Y} + \underline{\Delta}_y \underline{R}_m \hat{\underline{G}}_m \tag{12}$$

where $\hat{\underline{\alpha}} \triangleq \underline{\alpha} - \underline{Y} \underline{R}_m$ is the Norton equivalent current coupling matrix and $\hat{\underline{G}}_m \triangleq \underline{G}_m - \underline{Y} \underline{\mu}$ is the Norton equivalent transconductance matrix (see Johnson and Pennington [10]). We will now show that all terms on the right side of (12) contribute elements only to one row or column (except the term $\hat{\underline{\alpha}} \underline{\Delta}_y$, q.v.), if only one network parameter is changed. As a result, $\underline{\Delta}$ may be written $\underline{u} \underline{v}^T$, as explained in Section 2.1.

1. $\underline{\Delta}_y$ places a term in element (k,k) of $\underline{\Delta}$.

2. $$(\underline{\Delta}_y \underline{\mu})_{ij} = \sum_m (\Delta_y)_{im} \mu_{mj}$$
$$= (\Delta_y)_{ii} \mu_{ij}$$
$$= 0 \ , \ i \neq k$$

Therefore $\underline{\Delta}_y \underline{\mu}$ places elements only in row k of $\underline{\Delta}$.

3. $$(\hat{\underline{\alpha}} \underline{\Delta}_y)_{ij} = \sum_m \hat{\alpha}_{im} (\Delta_y)_{mj}$$
$$= \hat{\alpha}_{ij} (\Delta_y)_{jj}$$
$$= 0 \ , \ j \neq k$$

Therefore $\hat{\underline{\alpha}} \underline{\Delta}_y$ places elements in column k of $\underline{\Delta}$, if branch k contains current controlled sources.

4. $$(\underline{\alpha} \underline{\Delta}_y \underline{\mu})_{ij} = \sum_m \hat{\alpha}_{im} (\underline{\Delta}_y \underline{\mu})_{mj}$$
$$= \hat{\alpha}_{iK} (\underline{\Delta}_y \underline{\mu})_{Kj}$$
$$= \hat{\alpha}_{iK} (\Delta_y)_{KK} \mu_{Kj}$$

If such a term were present, the network model would contain a controlled source in branch i ($\hat{\alpha}_{ik}$) which was controlled by the branch current of a branch containing a controlled source (branch k contains source μ_{kj}). Such a condition is not allowed in the model [10], so $\underline{\alpha}\,\underline{\Delta}_y\,\underline{\mu} = \underline{0}$.

The final two terms in (12) contain the product $\underline{\Delta}_y\,\underline{R}_m$. Element (i, j)of this matrix is

$$(\underline{\Delta}_y\,\underline{R}_m)_{ij} = \sum_m (\Delta_y)_{im}\,(R_m)_{mj}$$

$$= (\Delta_y)_{ii}\,(R_m)_{ij} = 0\ ,\ i \neq k$$

$$(\underline{\Delta}_y\,\underline{R}_m)_{ki} = (\Delta_y)_{kk}\,(R_m)_{kj}\ ,\ i = k$$

We now consider the final two terms in (12).

5. $$(\underline{\Delta}_y\,\underline{R}_m\,\underline{Y})_{ij} = \sum_m (\underline{\Delta}_y\,\underline{R}_m)_{im}\,Y_{mj}$$

$$= (\underline{\Delta}_y\,\underline{R}_m)_{ij}\,Y_{jj} = 0\ ,\ i \neq k$$

$$(\underline{\Delta}_y\,\underline{R}_m\,\underline{Y})_{kj} = (\Delta_y)_{kk}\,(R_m)_{kj}\,Y_{ij} \qquad \forall\ j$$

Therefore $\underline{\Delta}_y\,\underline{R}_m\,\underline{Y}$ contributes terms along row k of $\underline{\Delta}$.

6. $$(\underline{\Delta}_y\,\underline{R}_m\,\hat{\underline{G}}_m)_{ij} = \sum_m (\underline{\Delta}_y\,\underline{R}_m)_{im}\,(\hat{G}_m)_{mj} = 0\ ,\ i \neq k$$

$$(\underline{\Delta}_y\,\underline{R}_m\,\hat{\underline{G}}_m)_{\ j} = \sum_m (\underline{\Delta}_y\,\underline{R}_m)_{km}\,(\hat{G}_m)_{mj}$$

$$= \sum_m (\Delta_y)_{kk}\,(R_m)_{km}\,(\hat{G}_m)_{mj}$$

$$= (\Delta_y)_{kk} \sum_m (R_m)_{km}\,(\hat{G}_m)_{mj} \qquad \forall\ j$$

Therefore $\underline{\Delta}_y\,\underline{R}_m\,\hat{\underline{G}}_m$ contributes terms along row k of $\underline{\Delta}$, although few, if any, such terms are likely to exist since the presence of this term requires that the network contain

a transresistance controlled by the current in a branch containing a voltage controlled source.

2.1.2 Parameter Change is a Transresistance†

If a change is made in transresistance $(R_m)_{ij}$, then $\underline{R}_m$ becomes $\underline{R}_m + \underline{\Delta}_r$, where $\underline{\Delta}_r$ is zero except for element $(\Delta_r)_{ij}$. Replacing $\underline{R}_m$ by $\underline{R}_m + \underline{\Delta}_r$ in (8) and performing some algebraic reduction, $\underline{Y}_b$ becomes $\underline{Y}_b + \underline{\Delta}$ where

$$\underline{\Delta} = - \underline{Y}\, \underline{\Delta}_r\, \underline{Y} - \underline{Y}\, \underline{\Delta}_r\, \hat{\underline{G}}_m \qquad (13)$$

Both terms in (13) contain the product $\underline{Y}\, \underline{\Delta}_r$. Expanding element (r, s) of $\underline{Y}\, \underline{\Delta}_r$

$$(\underline{Y}\, \underline{\Delta}_r)_{rs} = \sum_m Y_{rm}\, (\Delta_r)_{ms}$$

$$= Y_{rr}\, (\Delta_r)_{rs} = 0 \quad \text{if } r \neq i,\ s \neq j$$

$$(\underline{Y}\, \underline{\Delta}_r)_{ij} = Y_{ii}\, (\Delta_r)_{ij}$$

Therefore $\underline{Y}\, \underline{\Delta}_r$ has only one non-zero element (i, j). We now expand element (r, s) of both terms of (13).

1. $$(\underline{Y}\, \underline{\Delta}_r\, \underline{Y})_{rs} = \sum_m (\underline{Y}\, \underline{\Delta}_r)_{rm}\, Y_{ms}$$

$$= (\underline{Y}\, \underline{\Delta}_r)_{rs}\, Y_{ss}$$

$$= 0\ ,\ r \neq i\ ,\ s \neq j$$

$$(\underline{Y}\, \underline{\Delta}_r\, \underline{Y})_{ij} = Y_{ii}\, (\Delta_r)_{ij}\, Y_{jj}$$

---

† The reader may skip this section without loss of continuity.

Therefore $\underline{Y}\ \underline{\Delta}_r\ \underline{Y}$ contains a single non-zero element (i, j).

2. $$(\underline{Y}\ \underline{\Delta}_r\ \hat{\underline{G}}_m)_{rs} = \sum_m (\underline{Y}\ \underline{\Delta}_r)_{rm}\ (\hat{G}_m)_{ms}$$
$$= 0\ ,\ r \neq i,\ m \neq j$$

$$(\underline{Y}\ \underline{\Delta}_r\ \hat{\underline{G}}_m)_{is} = (\underline{Y}\ \underline{\Delta}_r)_{ij}\ (\hat{G}_m)_{js}$$
$$= Y_{ii}\ (\Delta_r)_{ij}\ (\hat{G}_m)_{js}\ ,\ \forall\ s$$

Therefore $\underline{Y}\ \underline{\Delta}_r\ \underline{G}_m$ places terms in row i of $\underline{\Delta}$.

2.1.3 Parameter Change is a Current Gain†

If a change is made in current gain α_{ij} , then $\underline{\alpha}$ is changed to $\underline{\alpha} + \underline{\Delta}_\alpha$, where $\underline{\Delta}_\alpha$ is zero except for element $(\Delta_\alpha)_{ij}$. Replacing $\underline{\alpha}$ by $\underline{\alpha} + \underline{\Delta}_\alpha$ in (8), after some algebraic reduction $\underline{Y}_b$ becomes $\underline{Y}_b + \underline{\Delta}$, where

$$\underline{\Delta} = -\ \underline{\Delta}_\alpha\ \underline{Y} - \underline{\Delta}_\alpha\ \hat{\underline{G}}_m \qquad (14)$$

We now expand element (r, s) of both terms of (14).

1. $$(\underline{\Delta}_\alpha\ \underline{Y})_{rs} = \sum_m (\Delta_\alpha)_{rm}\ Y_{ms}$$
$$= (\Delta_\alpha)_{rs}\ Y_{ss} = 0\ ,\ r \neq i,\ s \neq j$$

$$(\underline{\Delta}_\alpha\ \underline{Y})_{ij} = (\Delta_\alpha)_{ij}\ Y_{jj}$$

Therefore $\underline{\Delta}_\alpha\ \underline{Y}$ contains a single non-zero element (i, j) .

2. $$(\underline{\Delta}_\alpha\ \hat{\underline{G}}_m)_{rs} = \sum_m (\Delta_\alpha)_{rm}\ (\hat{G}_m)_{ms}$$
$$= 0\ ,\ r \neq i,\ m \neq j$$

$$(\underline{\Delta}_\alpha\ \ \underline{G}_m)_{is} = (\Delta_\alpha)_{ij}\ (\hat{G}_m)_{js}\ ,\ \forall\ s$$

Therefore $\underline{\Delta}_\alpha\ \underline{G}_m$ contributes elements in row i of $\underline{\Delta}$.

---

† The reader may skip this section without loss of continuity.

2.1.4 Parameter Change is a Voltage Gain†

If a change is made in voltage gain μ_{ij} , then $\underline{\mu}$ is changed to $\underline{\mu} + \underline{\Delta}_{\mu}$, where $\underline{\Delta}_{\mu}$ is zero except for element $(\Delta_{\mu})_{ij}$. Replacing $\underline{\mu}$ by $\underline{\mu} + \underline{\Delta}_{\mu}$ in (8), after some algebraic reduction $\underline{Y}_b$ becomes $\underline{Y}_b + \underline{\Delta}$, where

$$\underline{\Delta} = - \underline{Y}\, \underline{\Delta}_{\mu} - \hat{\underline{\alpha}}\, \underline{Y}\, \underline{\Delta}_{\mu} \tag{15}$$

We now expand element (r, s) of both terms of (15).

1. $$(\underline{Y}\, \underline{\Delta}_{\mu})_{rs} = \sum_{m} Y_{rm}\, (\Delta_{\mu})_{ms}$$

$$= Y_{rr}\, (\Delta_{\mu})_{rs} = 0 \ , \ r \neq i \ , \ s \neq j$$

$$(\underline{Y}\, \underline{\Delta}_{\mu})_{ij} = Y_{ii}\, (\Delta_{\mu})_{ij}$$

Therefore $\underline{Y}\, \underline{\Delta}_{\mu}$ contributes only one term to $\underline{\Delta}$.

2. $$(\hat{\underline{\alpha}}\, \underline{Y}\, \underline{\Delta}_{\mu})_{rs} = \sum_{m} \hat{\alpha}_{rm}\, (\underline{Y}\, \underline{\Delta}_{\mu})_{ms}$$

$$= 0 \ , \ m \neq i \ , \ s \neq j$$

$$(\hat{\underline{\alpha}}\, \underline{Y}\, \underline{\Delta}_{\mu})_{rj} = \hat{\alpha}_{ri}\, (\underline{Y}\, \underline{\Delta}_{\mu})_{ij}$$

$$= \hat{\alpha}_{ri}\, Y_{ii}\, (\Delta_{\mu})_{ij} \ , \ \forall\ r$$

Therefore $\hat{\underline{\alpha}}\, \underline{Y}\, \underline{\Delta}_{\mu}$ contributes terms down column j of $\underline{\Delta}$.

2.1.5 Parameter Change is a Transconductance†

If a change is made in transconductance $(G_m)_{ij}$ then $\underline{G}_m$ is changed to $\underline{G}_m + \underline{\Delta}_g$, where $\underline{\Delta}_g$ is zero except for element $(\underline{\Delta}_g)_{ij}$.

---

† The reader may skip this section without loss of continuity.

Replacing $\underline{G}_m$ by $\underline{G}_m + \underline{\Delta}_g$ in (8), after some algebraic reduction $\underline{Y}_b$ becomes $\underline{Y}_b + \underline{\Delta}$, where

$$\underline{\Delta} = \underline{\Delta}_g + \hat{\underline{\alpha}}\,\underline{\Delta}_g \tag{16}$$

We now discuss both terms in (16).

1. $\underline{\Delta}_g$ contributes only one term, $(\Delta_g)_{ij}$, to $\underline{\Delta}$.
2. $(\hat{\underline{\alpha}}\,\underline{\Delta}_g)_{rs} = \sum_m \hat{\alpha}_{rm}\,(\Delta_g)_{ms} = 0$, $m \neq i$, $s \neq j$

   $(\hat{\underline{\alpha}}\,\underline{\Delta}_g)_{rj} = \hat{\alpha}_{ri}\,(\Delta_g)_{ij}$, $\forall$ r

Therefore $\hat{\underline{\alpha}}\,\underline{\Delta}_g$ contribute terms down column j of $\underline{\Delta}$.

2.2 Finding Independent Measurements

In Section 1 we discussed the problem of solving the nonlinear equation $\underline{M}(\underline{x}) = \hat{\underline{M}}$, in which $\underline{x}$ is the unknown parameter vector of dimension n, $\hat{\underline{M}}$ is the vector of m network responses measured at the accessible terminals, and $\underline{M}(\underline{x})$ is the vector of m network responses calculated from the network model. The three cases ($m<n$, $m=n$, $m>n$) must be considered separately. Ransom and Saeks [2] have discussed the underdetermined case ($m<n$), which is not very practical because the fault cannot be resolved satisfactorily in this case. The second and third cases ($m\geq n$) can be solved by finding the parameter vector $\underline{x}$ which minimizes the norm $||\underline{M}(\underline{x}) - \hat{\underline{M}}||$. (This approach has the advantage of taking measurement error into account in a sensible way in the overdetermined case, $m>n$). In Section 2, we discussed the solution to this optimization problem by means of a search over the parameter set $\underline{x}$. An alternative approach, which provides some insight into the question of independent measurements, is to expand $\underline{M}(\underline{x})$ in a Taylor series about the nominal parameter set $\underline{x}_o$. If we truncate the Taylor series after the linear terms, we obtain the approximation

$$\underline{M}(\underline{x}) \approx \underline{M}(\underline{x}_o) + \underline{F}(\underline{x}_o)(\underline{x} - \underline{x}_o) \tag{16}$$

where $\underline{F}(\underline{x})$ is the Jacobian Matrix defined by

$$\left\{\underline{F}(\underline{x})\right\}_{ij} \triangleq \frac{\partial\{\underline{M}(\underline{x})\}_i}{\partial x_j} \qquad \left\{\begin{array}{l} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n \end{array}\right\} \tag{17}$$

From (16) the linearized form of the equation $\underline{M}(\underline{x}) = \hat{\underline{M}}$ is determined to be $\underline{F}(\underline{x}_o)(\underline{x} - \underline{x}_o) = \hat{\underline{M}} - \underline{M}(\underline{x}_o)$, in which, as before, dim $(\underline{x} - \underline{x}_o) = n$ and dim $(\hat{\underline{M}}-\underline{M}) = m$. Since we are treating the overdetermined case $m>n$,

we proceed by minimizing $||\underline{F}_o(\underline{x}-\underline{x}_o) - (\hat{\underline{M}}-\underline{M}_o)||$, where $\underline{F}_o \triangleq \underline{F}(\underline{x}_o)$ and $\underline{M}_o \triangleq \underline{M}(\underline{x}_o)$. The minimization of the norm is accomplished by first defining the scalar function

$$\begin{aligned}\phi(\underline{x}) &\triangleq ||\underline{F}_o(\underline{x}-\underline{x}_o) - (\underline{M}-\underline{M}_o)||^2 \\ &= [\underline{F}_o(\underline{x}-\underline{x}_o) - (\underline{M}-\underline{M}_o)]^T [\underline{F}_o(\underline{x}-\underline{x}_o) - (\underline{M}-\underline{M}_o)] \\ &= (\underline{x}-\underline{x}_o)^T \underline{F}_o^T \underline{F}_o (\underline{x}-\underline{x}_o) \\ &\quad -2(\underline{M}-\underline{M}_o)^T \underline{F}_o (\underline{x}-\underline{x}_o) \\ &\quad + (\underline{M}-\underline{M}_o)^T (\underline{M}-\underline{M}_o)\end{aligned} \tag{18}$$

and then taking the differential of $\phi(\underline{x})$:

$$\begin{aligned}d\phi &= 2[(\underline{x}-\underline{x}_o)^T \underline{F}_o^T \underline{F}_o - (\underline{M}-\underline{M}_o)^T \underline{F}_o]d\underline{x} \\ &\quad + d\underline{x}^T \underline{F}_o^T \underline{F}_o \, d\underline{x}\end{aligned} \tag{19}$$

At a minimum of $\phi(\underline{x})$, $d\phi$ may not change sign when $d\underline{x}$ changes sign. Therefore a necessary condition for a minimum of $\phi(\underline{x})$ is

$$(\underline{x}-\underline{x}_o)^T \underline{F}_o^T\underline{F}_o - (\underline{M}-\underline{M}_o)^T \underline{F}_o = \underline{0}$$

or

$$\underline{F}_o^T\underline{F}_o \, (\underline{x}-\underline{x}_o) = \underline{F}_o^T \, (\underline{M}-\underline{M}_o) \, . \tag{20}$$

The deviation $\underline{x}-\underline{x}_o$ of the element values from their nominal value may be found from (20) if the nxn matrix $\underline{F}_o^T\underline{F}$ is non-singular, which requires in this case that rank $(\underline{F}) = n$.†

† We do not necessarily recommend using (20) to determine the parameter set $\underline{x}$ of the faulted network, although this may be the most efficient approach. See Section 3 for a discussion of computational complexity.

Since $m \geq n$, the mxn Jacobian matrix $\underline{F}$ has at least as many rows as columns, and the maximum rank of $\underline{F}$ is n. The rank of $\underline{F}$ is less than n if any column of $\underline{F}$ is a linear combination of the remaining columns. The simplest case is that a column is a constant multiple of another column. Suppose, for example, (col i) = a (col j), i.e.

$$\frac{\partial M_k}{\partial x_i} = a \frac{\partial M_k}{\partial x_j} \quad (k = 1, 2, \ldots, m) \tag{21}$$

One obvious way this can occur is if parameters x_i and x_j are two like elements in series or parallel. Suppose that x_i and x_j are the conductances of two resistors in parallel, so that $y \triangleq x_i + x_j$ is the combined conductance. All transfer functions which do not access separately the inputs or outputs of x_i and x_j may be written $M_k(y,\underline{z})$, where $\underline{z}$ is the vector of all network parameters except x_i and x_j. Now the partial derivatives of M_k with respect to x_i and x_j are

$$\frac{\partial M_K}{\partial x_i} = \frac{\partial M_K}{\partial y} \cdot \frac{\partial y}{\partial x_i} = \frac{\partial M_k}{\partial y} \tag{22}$$

$$\frac{\partial M_k}{\partial x_i} = \frac{\partial M_k}{\partial y} \cdot \frac{\partial y}{\partial x_i} = \frac{\partial M_k}{\partial y} \tag{23}$$

Thus (21) applies with a=1, as asserted. This result, that no distinction can be made between imbedded like element kinds in series or parallel, is intuitively reasonable.

A less obvious example of column dependency is illustrated by the simplified cascaded amplifier circuit shown in Figure 2. Define the transfer functions and network parameters

$$M_1 = z_{11} = R_1 \qquad x_1 = R_1$$
$$M_2 = z_{21} = g_{m1}g_{m2}R_1R_2R_3 \qquad x_2 = R_2$$
$$M_3 = z_{22} = R_3 \qquad x_3 = R_3$$
$$x_4 = g_{m1}$$
$$x_5 = g_{m2}$$

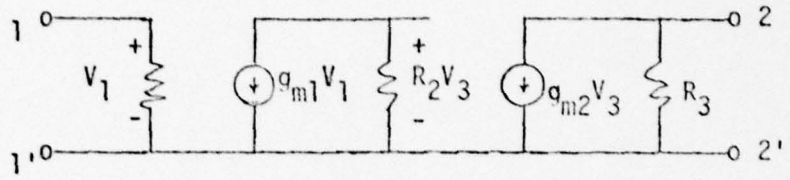


Figure 2 - Simplified Cascaded Amplifier

The transimpedance z_{12} is not used in this example because its value is zero. The Jacobian matrix $\left\{\partial M_i/\partial x_j\right\}$ is

$$\underline{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ g_{m1}g_{m2}R_2R_3 & g_{m1}g_{m2}R_1R_3 & g_{m1}g_{m2}R_1R_2 & g_{m2}R_1R_2R_3 & g_{m1}R_1R_2R_3 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

If g_{m1} and g_{m2} are equal, then columns 4 and 5 are identical. This doesn't reduce the rank of $\underline{F}$ in this case (the rank of $\underline{F}$ is 3), but it could in practical cases where $\underline{F}$ has more rows than columns (m>n).

Continuing with the problem of column dependency, suppose x_i is the conductance of a resistor in parallel with a capacitor of capacitance x_j, and $x = x_i + j\omega x_j$ is the combined admittance. All transfer functions which do not access separately the inputs or outputs of parameters x_i and x_j may be written $M_K(x,\underline{z})$, where $\underline{z}$ is the vector of all network parameters except x_i and x_j. The partial

derivatives of M_K with respect to x_i and x_j are

$$\frac{\partial M_k}{\partial x_i} = \frac{\partial M_k}{\partial x} \cdot \frac{\partial x}{\partial x_i} = \frac{\partial M_K}{\partial x} \tag{24}$$

$$\frac{\partial M_k}{\partial x_j} = \frac{\partial M_k}{\partial x} \cdot \frac{\partial x}{\partial x_j} = j\omega \frac{\partial M_k}{\partial x} \tag{25}$$

Thus (21) applies with $a=j\omega$. Columns i and j of $\underline{F}$ are dependent, so rank $\underline{F} < n$ and the nxn matrix $\underline{F}^T\underline{F}$ cannot be inverted, since rank $\underline{F}^T\underline{F} < n$, if $m \geq n$. As a consequence, (20) cannot be solved for the new parameter vector. We have discovered the surprising result that the identification problem cannot be perfectly resolved if the number of measurements is at least as great as the number of parameters and the network contains imbedded elements in parallel (or in series). That rank $(\underline{F}) < n$ in this case will be illustrated now by an example.

In this example it is necessary that $m \geq n$, and we have chosen the simple 2-port network illustrated in Figure 3 in which $m=n=3$.

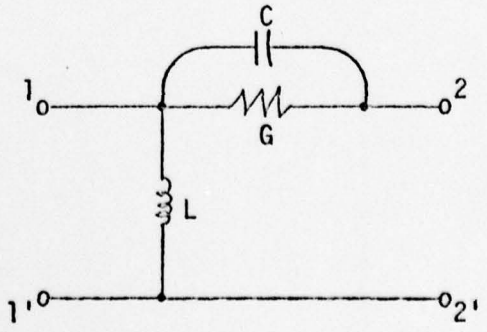


Figure 3 - Two-port Network.

We choose the three transfer functions and parameters

$$M_1 = y_{11} = \frac{1}{j\omega L} + j\omega C + G \qquad x_1 = L$$

$$M_2 = y_{21} = j\omega C + G \qquad x_2 = C$$

$$M_3 = y_{22} = j\omega C + G \qquad x_3 = G$$

The Jacobian matrix $\left\{ \frac{\partial M_i}{\partial K_j} \right\}$ is

$$\underline{F} = \begin{bmatrix} -\frac{1}{j\omega L^2} & j\omega & 1 \\ 0 & j\omega & 1 \\ 0 & j\omega & 1 \end{bmatrix}$$

as asserted, columns 2 and 3 of $\underline{F}$ are dependent. Since rank $\underline{F} = 2$, the fault analysis problem cannot be fully resolved for this problem.

In some cases the column dependency can be removed by increasing the number of measurements by using another frequency. In this example if M_1, M_2 and M_3 are defined as before at frequency $\omega=\omega_1$ and M_4, M_5 and M_6 are the same transfer functions at frequency $\omega=\omega_2$, the Jacobian matrix is

$$\underline{F} = \begin{bmatrix} -\frac{1}{j\omega_1 L^2} & j\omega_1 & 1 \\ 0 & j\omega_1 & 1 \\ 0 & j\omega_1 & 1 \\ -\frac{1}{j\omega_2 L^2} & j\omega_2 & 1 \\ 0 & j\omega_2 & 1 \\ 0 & j\omega_2 & 1 \end{bmatrix}$$

and rank ($\underline{F}$) = 3.

Another way to handle the problem is to split the transfer functions into their real and imaginary parts. Continuing the example, define the transfer functions and parameters

$$M_1 \triangleq \mathrm{Re}(y_{11}) = G \qquad x_1 = L$$
$$M_2 \triangleq \mathrm{Im}(y_{11}) = \omega C - \frac{1}{\omega L} \qquad x_2 = C$$
$$M_3 \triangleq \mathrm{Re}(y_{21}) = G \qquad x_3 = G$$
$$M_4 \triangleq \mathrm{Im}(y_{21}) = \omega C$$

Now m = 4, n = 3, and the Jacobian matrix is

$$\underline{F} = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{\omega L^2} & \omega & 0 \\ 0 & 0 & 1 \\ 0 & \omega & 0 \end{bmatrix}$$

and again rank ($\underline{F}$) = 3. Since M_1 and M_3 are identical, rows 1 and 3 of $\underline{F}$ are the same. This doesn't reduce the rank of $\underline{F}$, but it is clear that M_3 may be discarded without reducing the information of the measurements.

From the discussion above we conclude that the measurements should be split into their real and imaginary parts, and more than one frequency should be used, if necessary, to make the number of measurements at least as great as the number of network parameters ($m \geq n$).

In practical examples a considerable amount of computation is needed to find the functional form of the elements of the Jacobian matrix. Sen and Saeks use such functional forms in their discussion of the problem of independent measurements [6] - [8], but admit that no theory yet exists for choosing measurements while taking numerical

considerations into account.

In view of the discussion above, it appears that the most practical approach to the problem of identifying a sufficient number of independent measurements ($m \geq n$) is to examine the numerical conditioning of the nxn matrix $\underline{F}^T\underline{F}$, evaluated from the nominal network parameters. An algorithm for evaluating cond $(\underline{F}^T\underline{F})^\dagger$ is given by Forsythe and Moler [12].

Establishing a useful set of independent measurements requires searching for min cond $(\underline{F}^T\underline{F})$ over various sets of trial measurements and frequencies. This time consuming search seems most suitable to a "simulation before test" process; i.e., the best set of measurements is established from the circuit model before testing the faulty circuit. The contract period was not sufficiently long to permit trying this procedure, but a working algorithm is in preparation.

2.2.1 Computing the Jacobian Matrix

The Jacobian matrix is defined by $\underline{F} \triangleq [\partial M_i(\underline{x})/\partial x_j]$. We have taken the network responses $\underline{M}(\underline{x})$ to be the real and imaginary parts of the y-parameters at the accessible terminals, where, from (4),

$$\underline{y} = \underline{A}_1\underline{Y}_b\underline{A}_1^T - \underline{A}_1\underline{Y}_b\underline{A}_2^T (\underline{A}_2\underline{Y}_b\underline{A}_2^T)^{-1} \underline{A}_2\underline{Y}_b\underline{A}_1^T \tag{26}$$

$$\underline{Y}_b = (\underline{1} - \underline{Y}\,\underline{R}_m + \underline{\alpha})\,(\underline{Y} - \underline{Y}\,\underline{M} + \underline{G}_m)$$

---

† If $\underline{A}$ is a square matrix, cond $(\underline{A})$ is defined to be the ratio of the largest to smallest eigenvalue in absolute value. Therefore cond $(\underline{A}) \geq 1$, and large values for cond $(\underline{A})$ indicate nearly singular matrices for which the numerical evaluation of $\underline{A}^{-1}$ may be meaningless.

and so we want to compute $\partial \underline{y}/\partial x_i$. Making use of the identity

$$\frac{\partial \underline{\underline{B}}^{-1}}{\partial x} \equiv - \underline{\underline{B}}^{-1} \frac{\partial \underline{\underline{B}}}{\partial x} \underline{\underline{B}}^{-1}$$

and after much algebraic simplification, we find that

$$\frac{\partial \underline{y}}{\partial x_i} = \underline{A}_1 \left\{ [\underline{1} - \underline{Y}_b \underline{A}_2^T (\underline{A}_2 \underline{Y}_b \underline{A}^T)^{-1} \underline{A}_2] \frac{\partial \underline{Y}_b}{\partial x_i} [\underline{1} - \underline{A}_2^T (\underline{A}_2 \underline{Y}_b \underline{A}_2^T)^{-1} \underline{A}_2 \underline{Y}_b] \right\} \underline{A}_1^T \tag{27}$$

In (27) the second term in square brackets, $[\underline{1} - \underline{A}_2^T (\underline{A}_2 \underline{Y}_b \underline{A}_2^T)^{-1} \underline{A}_2 \underline{Y}_b]$, is available from the nominal analysis.† The first term in square brackets, $[\underline{1} - \underline{Y}_b \underline{A}_2^T (\underline{A}_2 \underline{Y}_b \underline{A}^T)^{-1} \underline{A}_2]$, is very similar to the second term in square brackets. In fact, if the network model contained no controlled sources, $\underline{Y}_b$ would be symmetrical, and these two terms would be the transpose of one another. The right side of (27) could be computed very efficiently in this case, especially since $\partial \underline{Y}_b/\partial x_i$ is very sparse. Unfortunately we cannot take advantage of this simplification, because the circuits of greatest concern contain controlled sources.

An algorithm for computing the right side of (27) as efficiently as possible, taking advantage of the sparseness of $\partial \underline{Y}_b/\partial x_i$, has been written. The reader is referred to the listing of subroutine JACCOL in Appendix 1.

† The term appears in equation (4) and is computed as the array $D(\cdot,\cdot)$ in subroutine YMAT. The listing may be found in Appendix 1.

3. Computational Complexity

The algorithm described in Section 2.1 makes use of a formula of Householder [11], which eliminates the need to invert the inaccessible nodal admittance matrix $\underline{Y}'$ to calculate the accessible transfer functions $\underline{y}$ each time a parameter is changed.† In this section we compare the amount of computation required to complete the search for $\min_{\underline{x}} ||\underline{M}(\underline{x})-\hat{\underline{M}}||$ using this efficient algorithm to the amount of computation which would be required if $\underline{Y}'_n$ were directly inverted at each step of the search.

The computational complexity will be described in terms of the number of complex multiplications performed by the algorithm. This depends on the network configuration and specifically on the number of accessible nodes, inaccessible nodes, branches, controlled sources, etc. The computational complexity of the efficient algorithm is even more difficult to estimate since it depends in addition, on such measures as the fraction of branches terminating on two inaccessible nodes, the fraction of branches terminating on only one inaccessible node, etc. For these reasons we have made the following assumptions: the total number of network parameters n is 1.1 times the number of branches, the number of branches is 2.2 times the number of inaccessible nodes, 45% of the branches terminate on 2 inaccessible nodes, 55% of the branches terminate on exactly 1 inaccessible node. Using these assumptions, the computational complexity can be calculated in terms of the number of inaccessible nodes, q.

The basic steps required to compute the measurable y-parameter matrix $\underline{y}$ are the same for both the direct and the efficient algorithms: (1) the new branch admittance matrix $\underline{Y}'_b$ is formed; (2) then the new nodal admittance matrix $\underline{Y}'_n$ is formed and inverted; (3) finally $\underline{y}$ is computed.

† See equations (4) - (7).

Steps (1) and (3) are identical in both the efficient and the direct methods. Making use of the assumptions discussed above, step (1) requires about $0.05q^2$ complex multiplications, and step (3) requires about $13.31q^2$ complex multiplications for a total of about $13.36q^2$ complex multiplications for steps (1) and (3) combined.

Using the efficient algorithm to invert $\underline{Y}'_n$ discussed in Section 2.1, step (2) requires about $q^2 + 3.45q + 4.21$ complex multiplications. Using the direct inversion of $\underline{Y}'_n$, step (2) requires about $5q^3 + 0.61q$ complex multiplications.

The total number of complex multiplications necessary to find the measurable y-parameters $\underline{y}$ after a single parameter is changed is therefore $14.36q^2 + 3.45q + 4.21$ using the efficient algorithm and $5q^3 + 13.36q^2 + 0.16q$ using the direct algorithm. The efficient algorithm is clearly very superior.

A Fibonacci search using 7 evaluations of $\underline{y}$ is performed for each parameter, and the number of parameters is assumed to be 1.1 times the number of branches or about 2.42q. Thus the number of complex multiplications required to search over all parameters is 16.94q times the values cited in the previous paragraph. These results are shown in Figure 4 as a function of the number of inaccessible nodes q.

We close this section with a discussion of the computational complexity of forming (20) and solving for the unknown parameters $\underline{x}$. Three steps are required:

(1) form the Jacobian matrix $\underline{F}$, which has a column for each parameter and a row for every measurement;

(2) form the product $\underline{F}.^{T}F$;

(3) solve for $\underline{x}$.

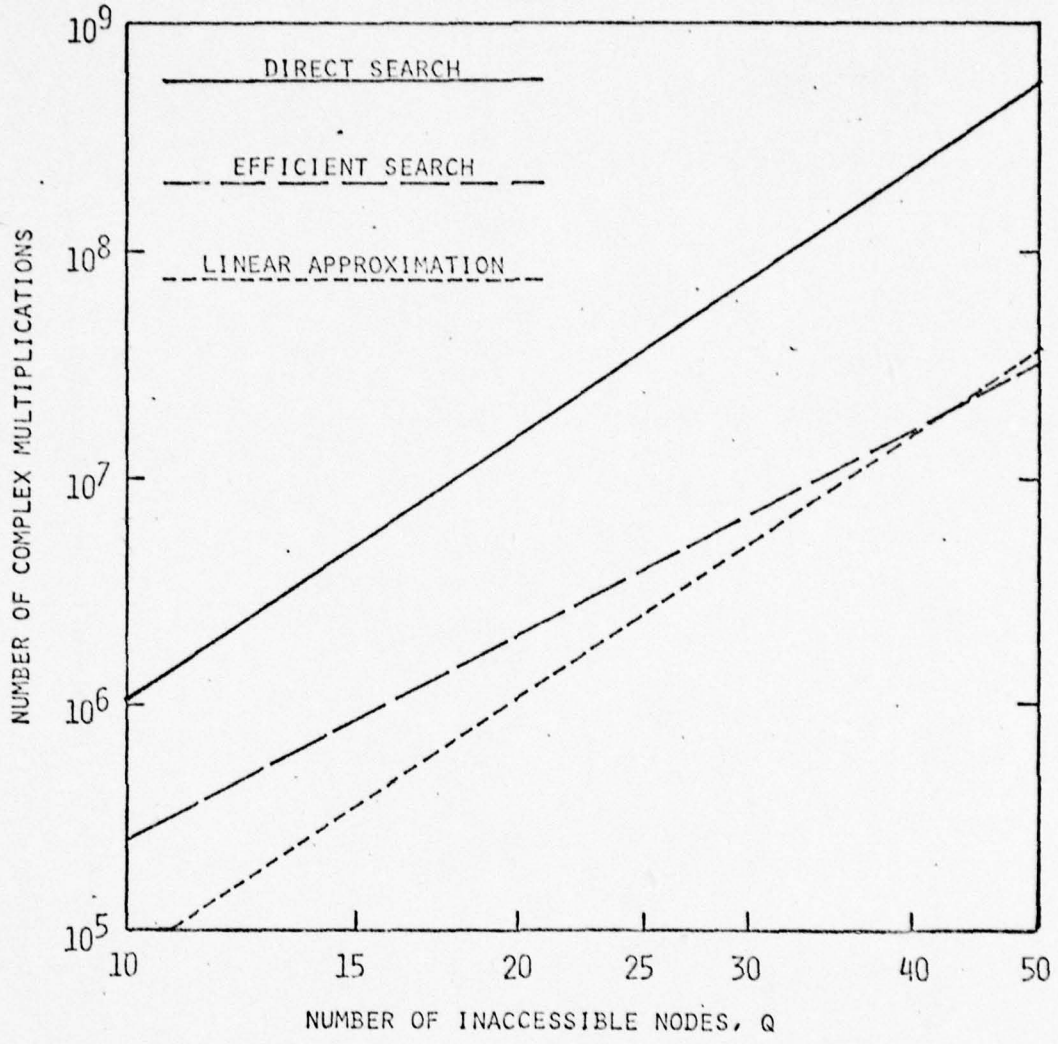


Figure 4 - The computational complexity of the parameter search.

Using the assumptions discussed previously, step (1) requires about $9.29q^3 + 26.14q^2$ real multiplications per parameter for a total of $22.48q^4 + 63.26q^3$ real multiplications. Step (2) requires about $2.93q^2m$ real multiplications, where m is the number of measurements. Assuming that $m = 2n \simeq 4.84q$, then step (2) requires about $14.18q^3$ real multiplications. Finally step (3) requires about $4.72q^3 + 5.86q^2 - 0.81q$ real multiplications. Therefore the total number of complex multiplications (1 complex multiplication = 4 real multiplications) to form (20) and solve for $\underline{x}$ is about $5.62q^4 + 20.54q^3 + 1.47q^2$. This is also plotted in Figure 4.

4.0 Conclusions

We have studied the problems of high resolution fault diagnosis in linear, analog electrical circuits in which a limited number of terminals are accessible for measurement. We have chosen to describe the faults in terms of short circuits between pairs of nodes and the values of parameters in the network model (resistances, capacitances, inductances, and controlled source gains). This approach is an outgrowth of our previous study [4], [5] in which the fault description was given only in terms of short circuits.

Specifically we have developed and tested an efficient algorithm for computing network responses (transfer functions at accessible terminals) after a single parameter is changed. The parameter change does not need to be small, and in fact the algorithm calculates exactly the new network responses, no matter how large the parameter change is. This algorithm was developed for use in a search over the network parameters for the network responses which best match the measured responses. The computational complexity (in terms of the number of complex multiplications) of this algorithm, which takes advantage of the special relationship between the network before and after a parameter is changed, was found to be very much better than the computational complexity of a direct algorithm. (See Figure 4).

In addition we studied the problem of choosing a set of measurements for fault analysis. We have found that the suitability of a set of measurements for resolving a fault can be expressed in terms of the Jacobian matrix $\underline{F}$ of the network responses (the partial derivatives of the network responses with respect to the network parameters). The fault can be resolved to the network parameter level only if the number of measurements m is at least as great as the number of network parameters n and, in addition, the rank of $\underline{F}$ is n. This latter condition can fail in the various ways illustrated in Section 2.2. The most promising way to find a set of independent measurements in practical cases appears to be to examine the numerical conditioning of the nxn matrix $\underline{F}^T\underline{F}$ over various combinations of measurements ($m \geq n$). We have written an efficient algorithm for forming the Jacobian matrix. Finding the "best-conditioned" nxn matrix $\underline{F}^T\underline{F}$ appears to be a time consuming process which seems best suited to a "simulation-before-test" process.

While investigating the measurement problem we discovered that it may be possible to accurately approximate the parameter set by solving a set of linear equations in which the coefficient matrix is $\underline{F}^T\underline{F}$. The computational complexity of forming and solving these equations increases as q^4 (q is the number of inaccessible nodes), whereas the computational complexity of an efficient search increases as q^3. However, Figure 4 shows that one cycle of the linear approximation is less complex than one cycle of an efficient search (using a seven point Fibonacci search in each coordinate direction) for networks having fewer than 40 inaccessible nodes. Until testing is completed, however, it is impossible to judge how many cycles of the linear approximation are needed for a good solution, and problems of convergence are also possible.

5.0 Future Work

We have developed several efficient algorithms for computing network responses and partial derivatives, which can be put together in various ways

to solve for the parameter values and short circuits, if any exist. Work is currently in progress to evaluate the various ways these algorithms can be used in forming an automatic fault diagnosis tool for analog electrical circuits which has the best combination of efficiency and reliability. This process will require a considerable amount of testing, including the use of real measurements on real circuits.

Appendix 1

Computer Program Listings

FAULT

```
ROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        MONDAY    10/08/79   01:43 PM
/(E99000)  /FAULTO
                  $DYNAMIC 13000                                                                  00000100                OFAULTO
                  $INTRINPACK 8                                                                   00000200                OFAULTO
                  $SAVELCM                                                                        00000300                OFAULTO
                  $ERRORTRACE                                                                     00000400                OFAULTO
                  FILE  3=FAULT3,UNIT=PRINTER                                                     00000500                OFAULTO
                  FILE  5=FAULT3DATA,UNIT=DISK                                                    00000600                OFAULTO
                  C                                                                               00000700                OFAULTO
                  C       THIS PROGRAM ISOLATES FAULTS IN ANALOG NETWORKS.  THE FAULT             00000800                OFAULTO
                  C       IS ISOLATED BY COMPARING COMPUTED Y-PARAMETERS AT THE                   00000900                OFAULTO
                  C       ACCESSIBLE NODES (BY A SEARCH OVER THE PARAMETER VALUES                 00001000                OFAULTO
                  C       FOLLOWED BY SHORTING EVERY PAIR OF NODES INVOLVING AT LEAST             00001100                OFAULTO
                  C       ONE INACCESSIBLE NODE) TO THE MEASURED Y-PARMETERS AT THE SAME          00001200                OFAULTO
                  C       ACCESSIBLE NODES.  FOR A DESCRIPTION OF THE EFFICIENT                   00001300                OFAULTO
                  C       ALGORITHM SEE:                                                          00001400                OFAULTO
                  C                                                                               00001500                OFAULTO
                  C         A.T. JOHNSON, JR.,"EFFICIENT FAULT ANALYSIS IN ANALOG                 00001600                OFAULTO
                  C           CIRCUITS", FINAL REPORT, USAF-ASEE SUMMER FACULTY RESEARCH          00001700                OFAULTO
                  C           PROGRAM, WPAFB, OHIO, AUGUST 1978                                   00001800                OFAULTO
                  C                                                                               00001900                OFAULTO
0003              C       COMMON/DELP/  ITYPE,   Y2IP, COSTMN,   JPAR,   NPAR,  ICORR,           00002000     000075     OFAULTO
0003              *             IX, IW, W,    WROW,    WCOL,      X,    XROW,   XCOL              00002100     000075     OFAULTO
0004              C       COMPLEX   Y2IP(17,17), W(80), X(80)                                     00002200     000075     OFAULTO
0005              C       INTEGER   WROW(80), WCOL(80), XROW(80), XCOL(80)                        00002300     000075     OFAULTO
0006              C       INTEGER ITYPE(80), ICORR(80)                                            00002400     000075     OFAULTO
                  C                                                                               00002500                OFAULTO
0007              C       COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),                00002600     000075     OFAULTO
0007              1             YB(65), YROW(65), GMROW(15), GMCOL(15),                           00002700     000075     OFAULTO
0007              2             RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),                     00002800     000075     OFAULTO
0007              3             AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,                    00002900     000075     OFAULTO
0007              4             IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR                           00003000     000075     OFAULTO
0008              C       INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,               00003100     000075     OFAULTO
0008              1             AROW, ACOL, YBROW, YBCOL                                          00003200     000075     OFAULTO
0009              C       COMPLEX Y, YB                                                           00003300     000075     OFAULTO
                  C                                                                               00003400                OFAULTO
0010              C       COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),            00003500     000075     OFAULTO
0010              1             Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),                   00003600     000075     OFAULTO
0010              2             A2MNUS(50), A2PLSP(50), A2MNSP(50), I, J, IP,                     00003700     000075     OFAULTO
0010              3             N, IB, ITOT, YN(3,3)                                              00003800     000075     OFAULTO
0011              C       INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                                  00003900     000075     OFAULTO
0012              C       INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                                  00004000     000075     OFAULTO
0013              C       COMPLEX YN, Y2, Y2I, Y2PI                                               00004100     000075     OFAULTO
                  C                                                                               00004200                OFAULTO
                  C                                                                               00004210                OFAULTO
0014              C       COMMON/SRCH/ PAR, DELPAR, YNNOM                                         00004220     000075     OFAULTO
                  C                                                                               00004230                OFAULTO
0015              C       DIMENSION DENOM(3,3)                                                    00004300     000075     OFAULTO
0016              C       COMPLEX YNNOM(3,3), DELYN(3,3)                                          00004400     000075     OFAULTO
                  C                                                                               00004600                OFAULTO
                  C       READ NETWORK DATA AND COMPUTE NODE-TO-BRANCH INCIDENCE MATRICES         00004700                OFAULTO
                  C       (SPARSE STORAGE) OF ORIGINAL NETWORK, A1PLUS, A1MNUS, A2PLUS,           00004800                OFAULTO
                  C       A2MNUS.                                                                 00004900                OFAULTO
                  C                                                                               00005000                OFAULTO
0017              C       CALL INPUT                                                              00005100     000075     OFAULTO
                  C                                                                               00007000                OFAULTO
                  C       READ MEASURED Y-PARAMETERS: YNNOM                                       00007100                OFAULTO
                  C                                                                               00007200                OFAULTO
0018              C       DO 5 K=1,IP                                                             00007300     000592     OFAULTO
0019              C       DO 5 L=1,IP                                                             00007400     000663     OFAULTO
0020              C       READ(5,108) YNNOM(K,L)                                                  00007500     000774     OFAULTO
0021              C       DENOM(K,L) = CABS(YNNOM(K,L))                                           00007600     001286     OFAULTO
```

```
0022    :       IF(DENOM(K,L) .EQ. 0.) DENOM(K,L) = 1.E-9                    : 00007700    001846   OFAULTO
0024    :     5 CONTINUE                                                     : 00007800    002361   OFAULTO
0025    :       WRITE(3,109)                                                 : 00007900    002706   OFAULTO
0026    :       DO 1 M=1,IP                                                  : 00008000    002981   OFAULTO
0027    :  1    WRITE(3,100)(YNNOM(M,NX),NX=1,IP)                            : 00008100    003072   OFAULTO
0028    :       IQ= N-IP                                                     : 00008200    003996   OFAULTO
        :C                                                                   : 00008225             OFAULTO
        :C                                                                   : 00008300             OFAULTO
        :C      COMPUTE YB MATRIX. SEE REF., EQ.(3).                         : 00008400             OFAULTO
        :C                                                                   : 00008500             OFAULTO
0029    :       CALL YBMAT                                                   : 00008600    004144   OFAULTO
        :C                                                                   : 00008700             OFAULTO
        :C      COMPUTE Y2 = A2 * YB * A2T.  SEE REF., EQ.(2).               : 00008800             OFAULTO
        :C                                                                   : 00008900             OFAULTO
0030    :       CALL Y2PMAT(Y2,IQ,17,A2PLUS,A2MNUS)                          : 00009000    004249   OFAULTO
        :C                                                                   : 00009100             OFAULTO
        :C      COMPUTE INVERSE OF Y2.                                       : 00009200             OFAULTO
        :C                                                                   : 00009300             OFAULTO
0031    :       IF(IQ .NE. 1) GO TO 8                                        : 00009320    004664   OFAULTO
0033    :         Y2I(1,1) = (1.,0.)/Y2(1,1)                                 : 00009340    004797   OFAULTO
0034    :         GO TO 7                                                    : 00009360    005252   OFAULTO
0035    :     8 CALL MINVC(Y2,Y2I,IQ,17)                                     : 00009400    005338   OFAULTO
        :C                                                                   : 00009500             OFAULTO
        :C      COMPUTE Y-PARAMETERS AT ACCESSIBLE TERMINALS. REF., EQ.(8).  : 00009600             OFAULTO
        :C                                                                   : 00009700             OFAULTO
0036    :     7 CALL YMAT(A1PLUS,A1MNUS,A2PLUS,A2MNUS,Y2I,YN,IP,I8,17)       : 00009800    005753   OFAULTO
0037    :       ISAVE = 0                                                    : 00009900    006456   OFAULTO
0038    :       JSAVE = 0                                                    : 00010000    006547   OFAULTO
        :C                                                                   : 00010100             OFAULTO
        :C      PRINT Y-PARAMETERS OF UNFAULTED NETWORK WITH NOMINAL         : 00010200             OFAULTO
        :C      PARAMETER VALUES.                                            : 00010300             OFAULTO
        :C                                                                   : 00010400             OFAULTO
0039    :       WRITE(3,102)                                                 : 00010500    006638   OFAULTO
0040    :       DO 2 M=1,IP                                                  : 00010600    006911   OFAULTO
0041    :  2    WRITE(3,100)(YN(M,NX),NX=1,IP)                               : 00010700    007002   OFAULTO
        :C                                                                   : 00010800             OFAULTO
        :C      COMPUTE COST FUNCTION OF UNFAULTED NETWORK.  CALL IT COSTMM. : 00010900             OFAULTO
        :C                                                                   : 00011000             OFAULTO
0042    :       COSTMN = COSTF(YNNOM)                                        : 00011100    007926   OFAULTO
0043    :       WRITE(3,107) COSTMN                                          : 00011200    008187   OFAULTO
        :C                                                                   : 00011400             OFAULTO
        :C      COMPUTE AND PRINT PARTIAL DERIVITIVES OF Y-PARAMETERS WITH   : 00011500             OFAULTO
        :C      RESPECT TO EACH NETWORK PARAMETER.                           : 00011600             OFAULTO
        :C                                                                   : 00011700             OFAULTO
0044    :       SENMAX = 0.                                                  : 00011750    008485   OFAULTO
0045    :       DO 3 I = 1,IPAR                                              : 00011800    008613   OFAULTO
0046    :         WRITE(3,110) I                                             : 00011850    008704   OFAULTO
0047    :         SENS = 0.                                                  : 00011900    009002   OFAULTO
0048    :         CALL JACCOL(DELYN, I)                                      : 00012000    009130   OFAULTO
0049    :         DO 4 K = 1,IP                                              : 00012010    009397   OFAULTO
0050    :         DO 4 L = 1,IP                                              : 00012020    009488   OFAULTO
0051    :           RY1 = REAL( YN(K,L) )                                    : 00012030    009579   OFAULTO
0052    :           XY1 = AIMAG( YN(K,L) )                                   : 00012040    009934   OFAULTO
0053    :           RY2 = REAL( YNNOM(K,L) )                                 : 00012050    010298   OFAULTO
0054    :           XY2 = AIMAG( YNNOM(K,L) )                                : 00012060    010653   OFAULTO
0055    :               IF(RY2 .EQ. 0.) RY2 = 1.                             : 00012062    011017   OFAULTO
0057    :               IF(XY2 .EQ. 0.) XY2 = 1.                             : 00012064    011304   OFAULTO
0059    :           SENS = SENS + (RY1 - RY2)*REAL( DELYN(K,L) )/(RY2*RY2)   : 00012070    011591   OFAULTO
0059    :    *                  + (XY1 - XY2)*AIMAG( DELYN(K,L) )/(XY2*XY2)  : 00012080    011591   OFAULTO
0060    :     6     CONTINUE                                                 : 00012082    012536   OFAULTO
0061    :         WRITE(3,100)(DELYN(K,NX), NX=1,IP)                         : 00012084    012739   OFAULTO
```

```
0062    :     4      CONTINUE                                                                    :  000012090     013519   DEFAULTO
0063    :            IF(ITYPE(I) .EQ. 1) GO TO 520                                               :  000012100     013722   DEFAULTO
0065    :            IF(ITYPE(I) .GE. 4) GO TO 525                                               :  000012110     013918   DEFAULTO
0067    :              IF(AIMAG( Y(ICORR(I)) ) .LT. 0.) GO TO 510                                :  000012120     014114   DEFAULTO
0069    :                ELE = AIMAG(Y(ICORR(I)))/OMEGA                                          :  000012130     014534   DEFAULTO
0070    :              GO TO 530                                                                 :  000012140     014943   DEFAULTO
0071    :    510       ELE = -1./(OMEGA*AIMAG(Y(ICORR(I))))                                      :  000012150     015029   DEFAULTO
0072    :              GO TO 530                                                                 :  000012160     015436   DEFAULTO
0073    :    520       ELE = REAL(Y(ICORR(I)))                                                   :  000012170     015552   DEFAULTO
0074    :              GO TO 530                                                                 :  000012180     015938   DEFAULTO
0075    :    525       GO TO (526,527,528,529), ITYPE(I) - 10                                    :  000012190     016024   DEFAULTO
0076    :    526       ELE = GM(ICORR(I))                                                        :  000012200     016252   DEFAULTO
0077    :              GO TO 530                                                                 :  000012210     016528   DEFAULTO
0078    :    527       ELE = RM(ICORR(I))                                                        :  000012220     016808   DEFAULTO
0079    :              GO TO 530                                                                 :  000012230     016848   DEFAULTO
0080    :    528       ELE = ALPHA(ICORR(I))                                                     :  000012240     016934   DEFAULTO
0081    :              GO TO 530                                                                 :  000012250     017174   DEFAULTO
0082    :    529       ELE = XMU(ICORR(I))                                                       :  000012260     017400   DEFAULTO
0083    :    530       SENS = 2.*SENS*ELE                                                        :  000012270     017500   DEFAULTO
0084    :              WRITE(3,104) I, SENS                                                      :  000012280     017656   DEFAULTO
0085    :              IF(ABS(SENMAX) .GT. ABS(SENS)) GO TO 3                                    :  000012290     018128   DEFAULTO
0087    :                SENMAX = SENS                                                           :  000012300     018291   DEFAULTO
0088    :                IPSAVE = I                                                              :  000012310     018388   DEFAULTO
0089    :     3    CONTINUE                                                                     :  000012350     018496   DEFAULTO
0090    :          WRITE(3,105) SENMAX, IPSAVE                                                  :  000012360     018699   DEFAULTO
0091    :    104   FORMAT(///10X, 'ERROR SENSITIVITY WITH RESPECT TO PARAMETER',               :  000012400     019131   DEFAULTO
0091    :     *          I3, ') =', E11.3)                                                      :  000012500     019131   DEFAULTO
0092    :    105   FORMAT(///10X, 'MAXIMUM SENSITIVITY =', E11.3,                              :  000012550     019131   DEFAULTO
0092    :     *          ' WITH RESPECT TO PARAMETER', I3)                                      :  000012600     019131   DEFAULTO
        :C                                                                                     :  000012700              DEFAULTO
        :C         *** BEGIN PARAMETER SEARCH ***                                              :  000012800              DEFAULTO
        :C                                                                                     :  000012900              DEFAULTO
0093    :          DO 41 I = 1, IPAR                                                           :  000013000     019131   DEFAULTO
0094    :            J = ICORR(I)                                                              :  000013100     019222   DEFAULTO
0095    :            NPAR = J                                                                  :  000013200     019399   DEFAULTO
0096    :            IF(ITYPE(I) .GE. 4) GO TO 35                                              :  000013700     019507   DEFAULTO
0098    :              JPAR = 1                                                                :  000013800     019703   DEFAULTO
0099    :            IF(ITYPE(I) .NE. 1) GO TO 31                                              :  000013900     019794   DEFAULTO
0101    :              PAR = REAL( Y(J) )                                                      :  000013700     019990   DEFAULTO
0102    :            GO TO 40                                                                  :  000013800     020293   DEFAULTO
0103    :     31       PAR = AIMAG( Y(J) )                                                     :  000013900     020379   DEFAULTO
0104    :            GO TO 40                                                                  :  000014000     020591   DEFAULTO
0105    :     35     JPAR = ITYPE(I) - 10                                                      :  000014100     020777   DEFAULTO
0106    :            IF(JPAR .EQ. 1) JPAR = 5                                                  :  000014200     020976   DEFAULTO
0108    :            GO TO(36, 37, 38, 39), JPAR - 1                                           :  000014300     021204   DEFAULTO
0109    :     36       PAR = RM(J)                                                             :  000014400     021399   DEFAULTO
0110    :              GO TO 40                                                                :  000014600     021576   DEFAULTO
0111    :     37       PAR = ALPHA(J)                                                          :  000014700     021662   DEFAULTO
0112    :              GO TO 40                                                                :  000014900     021839   DEFAULTO
0113    :     38       PAR = XMU(J)                                                            :  000015000     021955   DEFAULTO
0114    :              GO TO 40                                                                :  000015200     022102   DEFAULTO
0115    :     39       PAR = GM(J)                                                             :  000015300     022198   DEFAULTO
0116    :     40     CALL SEARCH( 0., 1.1*PAR, 0.05*PAR, .TRUE., PAR, COSTMX)                  :  000015500     022395   DEFAULTO
0117    :            COSTMN = 1./COSTMX                                                        :  000015501     022776   DEFAULTO
0118    :            DO 41 NX = 1,10                                                           :  000015502     022938   DEFAULTO
0119    :            DO 41 NY = 1,10                                                           :  000015503     023029   DEFAULTO
0120    :     41     Y2I(NX,NY) = Y2IP(NX,NY)                                                  :  000015540     023120   DEFAULTO
        :C                                                                                     :  000015600              DEFAULTO
        :C       * PRINT LOWEST COST AND NEW PARAMETER VALUES AFTER PARAMETER                  :  000015700              DEFAULTO
        :C         SEARCH.                                                                     :  000015800              DEFAULTO
        :C                                                                                     :  000015900              DEFAULTO
0121    :          WRITE(3,111) COSTMN                                                         :  000016000     024138   DEFAULTO
```

```
0122   :        WRITE(3,106)                                                    : 00016020     024435   OFAULTO
0123   :        DO 43 I = 1,IP                                                  : 00016040     024709   OFAULTO
0124   :     43 WRITE(3,100) (YM(I,NX), NX = 1,IP)                              : 00016060     024800   OFAULTO
0125   :        WRITE(3,112)                                                    : 00016100     025724   OFAULTO
0126   :        DO 50 I = 1,IB                                                  : 00016200     025997   OFAULTO
0127   :          IF(ITYPE(I) .EQ. 1) GO TO 60                                  : 00016300     026088   OFAULTO
0129   :          IF(AIMAG(Y(I)) .LT. 0.) GO TO 55                              : 00016400     026234   OFAULTO
       :C                                                                       : 00016500              OFAULTO
       :C         BRANCH I IS A CAPACITOR: ABS(Y) = OMEGA*C                     : 00016600              OFAULTO
       :C                                                                       : 00016700              OFAULTO
0131   :          ELE = AIMAG(Y(I))/OMEGA                                       : 00016800     026641   OFAULTO
0132   :          LABEL = 'C = '                                                : 00016900     026987   OFAULTO
0133   :          DIM = ' F  '                                                  : 00017000     027115   OFAULTO
0134   :          GO TO 70                                                      : 00017100     027243   OFAULTO
       :C                                                                       : 00017200              OFAULTO
       :C         BRANCH I IS AN INDUCTOR: ABS(Y) = 1/(OMEGA*L)                 : 00017300              OFAULTO
       :C                                                                       : 00017400              OFAULTO
0135   :     55   ELE = -1./(OMEGA*AIMAG(Y(I)))                                : 00017500     027329   OFAULTO
0136   :          LABEL = 'L = '                                                : 00017600     027723   OFAULTO
0137   :          DIM = ' H  '                                                  : 00017700     027851   OFAULTO
0138   :          GO TO 70                                                      : 00017800     027979   OFAULTO
       :C                                                                       : 00017900              OFAULTO
       :C         BRANCH I IS A RESISTOR: Y = 1/R                               : 00018000              OFAULTO
       :C                                                                       : 00018100              OFAULTO
0139   :     60   ELE = 1./REAL(Y(I))                                           : 00018200     028065   OFAULTO
0140   :          LABEL = 'R = '                                                : 00018300     028416   OFAULTO
0141   :          DIM = ' OHM'                                                  : 00018400     028544   OFAULTO
0142   :     70 WRITE(3,113) ICORR(I), LABEL, ELE, DIM                          : 00018500     028672   OFAULTO
0143   :        IF(LPH .EQ. 0) GO TO 163                                        : 00018600     029435   OFAULTO
0145   :        DO 62 K = 1,IRM                                                 : 00018700     029563   OFAULTO
0146   :          IF(RMROW(K) .EQ. I) GO TO 82                                  : 00018800     029659   OFAULTO
0148   :     62 CONTINUE                                                        : 00018900     029857   OFAULTO
0149   :    163 IF(IALPHA .EQ. 0) GO TO 164                                     : 00019000     030070   OFAULTO
0151   :        DO 63 K = 1,IALPHA                                              : 00019100     030203   OFAULTO
0152   :          IF(AROW(K) .EQ. I) GO TO 83                                   : 00019200     030294   OFAULTO
0154   :     63 CONTINUE                                                        : 00019300     030502   OFAULTO
0155   :    164 IF(IXMU .EQ. 0) GO TO 165                                       : 00019400     030705   OFAULTO
0157   :        DO 64 K = 1,IXMU                                                : 00019500     030833   OFAULTO
0158   :          IF(XMUROW(K) .EQ. I) GO TO 84                                 : 00019600     030929   OFAULTO
0160   :     64 CONTINUE                                                        : 00019700     031137   OFAULTO
0161   :    165 IF(IGM .EQ. 0) GO TO 50                                         : 00019800     031340   OFAULTO
0163   :        DO 65 K = 1,IGM                                                 : 00019900     031473   OFAULTO
0164   :          IF(GMROW(K) .EQ. I) GO TO 85                                  : 00020000     031564   OFAULTO
0166   :     65 CONTINUE                                                        : 00020100     031772   OFAULTO
0167   :        GO TO 50                                                        : 00020200     031975   OFAULTO
0168   :     82 SOURCE = 'RM ='                                                 : 00020300     032061   OFAULTO
0169   :          DIM = ' OHM'                                                  : 00020400     032189   OFAULTO
0170   :          GAIN = RM(K)                                                  : 00020500     032317   OFAULTO
0171   :          GO TO 51                                                      : 00020600     032494   OFAULTO
0172   :     83 SOURCE = 'AL ='                                                 : 00020700     032580   OFAULTO
0173   :          GAIN = ALPHA(K)                                               : 00020800     032708   OFAULTO
0174   :          DIM = '  '                                                    : 00020900     032885   OFAULTO
0175   :          GO TO 51                                                      : 00021000     033013   OFAULTO
0176   :     84 SOURCE = 'MU ='                                                 : 00021100     033099   OFAULTO
0177   :          GAIN = XMU(K)                                                 : 00021200     033227   OFAULTO
0178   :          DIM = '  '                                                    : 00021300     033404   OFAULTO
0179   :          GO TO 51                                                      : 00021400     033532   OFAULTO
0180   :     85 SOURCE = 'GM ='                                                 : 00021500     033618   OFAULTO
0181   :          GAIN = GM(K)                                                  : 00021600     033745   OFAULTO
0182   :          DIM = ' MHO'                                                  : 00021700     033923   OFAULTO
0183   :     51 WRITE(3,114) SOURCE, GAIN, DIM                                  : 00021800     034051   OFAULTO
```

```
0184    :     50 CONTINUE                                                                  : 00021900     034617   DFAULTD
        :C                                                                                 : 00022000              DFAULTD
        :C       LOOP THROUGH ALL PAIRS OF SHORT CIRCUITS INVOLVING AN                     : 00022100              DFAULTD
        :C       INACCESSIBLE NODE.  I AND J ARE THE SHORTED NODES.                        : 00022200              DFAULTD
        :C                                                                                 : 00022300              DFAULTD
0185    :       DO 10 K = IP+1,N                                                           : 00022400     034820   DFAULTD
0186    :       J = N + IP + 1 - K                                                         : 00022500     034956   DFAULTD
0187    :       NCASES = NCASES - J                                                        : 00022600     035160   DFAULTD
0188    :       DO 10 L=1,J                                                                : 00022700     035308   DFAULTD
0189    :       I = L-1                                                                    : 00022800     035399   DFAULTD
        :C                                                                                 : 00022900              DFAULTD
        :C       COMPUTE NEW NODE-TO-BRANCH INCIDENCE MATRICES (A1PLSP, A1MNSP,            : 00023000              DFAULTD
        :C       A2PLSP, A2MNSP) WITH NODES I AND J SHORTED.  SEE REF., EQ. (13).          : 00023100              DFAULTD
        :C                                                                                 : 00023200              DFAULTD
0190    :       CALL ANEW                                                                  : 00023300     035535   DFAULTD
        :C                                                                                 : 00023400              DFAULTD
        :C       COMPUTE INVERSE OF Y2P FOR FAULTED NETWORK.  SEE REF., EQ.(10).           : 00023500              DFAULTD
        :C                                                                                 : 00023600              DFAULTD
0191    :       IF(IQ .EQ. 1) GO TO 52                                                     : 00023650     035640   DFAULTD
0193    :       CALL SNEW                                                                  : 00023700     035773   DFAULTD
        :C                                                                                 : 00023800              DFAULTD
        :C       COMPUTE Y-PARAMETERS AT ACCESSIBLE TERMINALS OF FAULTED                   : 00023900              DFAULTD
        :C       NETWORK.  SEE REF., EQ.(23).                                              : 00024000              DFAULTD
        :C                                                                                 : 00024100              DFAULTD
0194    :    52 CALL YMAT(A1PLSP,A1MNSP,A2PLSP,A2MNSP,Y2PI,YN,IP,IB,16)                    : 00024200     035878   DFAULTD
        :C                                                                                 : 00024300              DFAULTD
        :C       COMPUTE COST FUNCTION OF CURRENT Y-PARAMETERS (NODES I AND J              : 00024400              DFAULTD
        :C       SHORTED).                                                                 : 00024500              DFAULTD
        :C                                                                                 : 00024600              DFAULTD
0195    :       COST = COSTF(YNNOM)                                                        : 00024700     036581   DFAULTD
0196    :       IF(COST - COSTMN) 22,21,21                                                 : 00024800     036842   DFAULTD
0197    :  22   COSTMN = COST                                                             : 00024900     037023   DFAULTD
0198    :       ISAVE = I                                                                  : 00025000     037131   DFAULTD
0199    :       JSAVE = J                                                                  : 00025100     037239   DFAULTD
        :C                                                                                 : 00025200              DFAULTD
        :C       PRINT NODES I AND J AND THE COMPUTED Y-PARAMETERS AT THE                  : 00025300              DFAULTD
        :C       ACCESSIBLE TERMINALS.                                                     : 00025400              DFAULTD
        :C                                                                                 : 00025500              DFAULTD
0200    :  21   WRITE(3,101) I,J                                                          : 00025600     037347   DFAULTD
0201    :       DO 30 M=1,IP                                                               : 00025700     037779   DFAULTD
0202    :  30   WRITE(3,100) (YN(M,NX),NX=1,IP)                                          : 00025800     037970   DFAULTD
0203    :  10   WRITE(3,107) COST                                                         : 00025900     038794   DFAULTD
        :C                                                                                 : 00026000              DFAULTD
        :C       PRINT NODES OF LOWEST COST FAULT AND COST FUNCTION FOR                    : 00026100              DFAULTD
        :C       THIS PAIR OF SHORTED NODES.                                               : 00026200              DFAULTD
        :C                                                                                 : 00026300              DFAULTD
0204    :       WRITE(3,103) ISAVE, JSAVE, COSTMN                                          : 00026400     039380   DFAULTD
0205    :       STOP                                                                       : 00026500     039946   DFAULTD
0206    :  100  FORMAT (3(10X,2E12.4,' J')//)                                             : 00026600     040064   DFAULTD
0207    :  101  FORMAT('1',10X,'Y-PARAMETERS WITH NODES',I3,                               : 00026700     040064   DFAULTD
0207    :     *    ' AND',I3,' SHORTED'///)                                                : 00026800     040064   DFAULTD
0208    :  102  FORMAT(///,10X,'BEGIN SEARCH USING NOMINAL NETWORK.'///,                   : 00026900     040064   DFAULTD
0208    :     *    10X, 'Y-PARAMETERS WITH NO SHORT CIRCUITS, MHOS:'/)                     : 00027000     040064   DFAULTD
0209    :  103  FORMAT('1',10X,'MOST LIKELY SINGLE SHORT:'//11X,'NODES',                   : 00027100     040064   DFAULTD
0209    :     *       I3,' AND',I4,' SHORTED.'////10X,'(COST FUNCTION =',                  : 00027200     040064   DFAULTD
0209    :     *       E11.4,')' )                                                          : 00027300     040064   DFAULTD
0210    :  106  FORMAT(/30X, 'Y-PARAMETERS'/)                                             : 00027325     040064   DFAULTD
0211    :  107  FORMAT(//,56X,'COST =', E11.3)                                            : 00027400     040064   DFAULTD
0212    :  108  FORMAT(E10.4,1X,E10.4)                                                    : 00027500     040064   DFAULTD
0213    :  109  FORMAT(///,10X,'MEASURED Y-PARAMETERS, MHOS.'/)                           : 00027600     040064   DFAULTD
0214    :  110  FORMAT(5(/), 10X, 'PARTIAL DERIVATIVES OF Y-PARAMETERS',                  : 00027625     040064   DFAULTD
0214    :     *     ' WITH RESPECT TO NETWORK PARAMETER', I3/)                           : 00027650     040064   DFAULTD
```

```
0215   : 111  FORMAT(///, 41X, 'LOWEST COST AFTER PARAMETER SEARCH =',          : 00027700   040064  OFAULTO
0215   :     *      E11.3)                                                      : 00027800   040064  OFAULTO
0216   : 112  FORMAT(///, 9X, 'BRANCH', T22, 'PASSIVE', T43, 'ACTIVE',         : 00027900   040064  OFAULTO
0216   :     *      /, 9X, 'NUMBER', T22, 'ELEMENT', T43, 'ELEMENT'/)           : 00028000   040064  OFAULTO
0217   : 113  FORMAT( I14, 3X, A4, E10.4, A4 )                                 : 00028100   040064  OFAULTO
0218   : 114  FORMAT( '+', T39, A4, E10.4, A4 )                                 : 00028200   040064  OFAULTO
0219   : 290  FORMAT(8(F8.2,F6.2,' J'))                                        : 00028300   040064  OFAULTO
0220   :      END                                                               : 00028400   040064  OFAULTO

0 ERRORS AND NO WARNINGS IN 220 STATEMENTS, CODE EMITTED = 40182 BITS (5023 BYTES)OFAULTO
COMPILE TIME IS 7 MINUTES AND 44.9 SECONDS FOR 314 CARDS AT 41 CARDS/MINUTE.
```

```
BURROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:10 PM
      /(E99000)   /FAULT3O
0001  :       BLOCK DATA                                                    : 00028500    000075  OFAULT3O
0002  :       COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),  : 00028600    000075  BLOCK.
0002  :     1             Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),  : 00028700    000075  BLOCK.
0002  :     2             A2MNUS(50), A2PLSP(50), A2MNSP(50), I, J, IP,    : 00028800    000075  BLOCK.
0002  :     3             N, IB, ITOT, YN(3,3)                             : 00028900    000075  BLOCK.
0003  :       INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                        : 00029000    000075  BLOCK.
0004  :       INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                        : 00029100    000075  BLOCK.
0005  :       DATA A2PLUS,A2MNUS/100*0/                                     : 00029200    000075  BLOCK.
0006  :       DATA A1PLUS,A1MNUS/100*0/                                     : 00029300    000075  BLOCK.
0007  :       END                                                           : 00029400    000075  BLOCK.

NO ERRORS AND NO WARNINGS IN 7 STATEMENTS,CODE EMITTED = 258 BITS (33 BYTES)BLOCK.
COMPILE TIME IS 26.9 SECONDS FOR 10 CARDS AT 22 CARDS/MINUTE.
```

```
UNROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
      /(F99999)  /FAULT30
0001  :      SUBROUTINE INPUT                                                                  : 00029500   000075  OFAULT30
      :C                                                                                       : 00029600             OFAULT30
      :C     THIS SUBROUTINE READS THE NETWORK DATA AND STORES THE NODE-                       : 00029700             OFAULT30
      :C     TO-BRANCH INCIDENCE MATRICES.                                                     : 00029800             OFAULT30
      :C                                                                                       : 00029900             OFAULT30
      :C     VARIABLE DEFINITIONS:                                                             : 00030000             OFAULT30
      :C                                                                                       : 00030100             OFAULT30
      :C       IB      - NUMBER OF BRANCHES IN NETWORK (PASSIVE ELEMENTS)                      : 00030200             OFAULT30
      :C       NB      - BRANCH NUMBER                                                         : 00030300             OFAULT30
      :C       IPAR    - PARAMETER NUMBER. ASSIGNED SEQUENTIALLY IN THE                        : 00030400             OFAULT30
      :C                 ORDER PARAMETERS ARE READ.                                          : 00030500             OFAULT30
      :C       IITYPE(.)     VECTOR OF PARAMETER TYPES: 1 = RESISTOR,                        : 00030600             OFAULT30
      :C                       2 = INDUCTOR, 3 = CAPACITOR, 11 = GM,                         : 00030700             OFAULT30
      :C                      12 = RM, 13 = ALPHA, 14 = MU                                   : 00030800             OFAULT30
      :C       ICORR(IPAR) = BRANCH NUMBER OF PARAMETER IPAR                                 : 00030900             OFAULT30
      :C                                                                                       : 00031000             OFAULT30
      :C       IGM     - NUMBER OF TRANSCONDUCTANCES                                         : 00031100             OFAULT30
      :C       IRM     - NUMBER OF TRANSRESISTANCES                                          : 00031200             OFAULT30
      :C       IXMU    - NUMBER OF VOLTAGE GAINS                                             : 00031300             OFAULT30
      :C       IALPHA  - NUMBER OF CURRENT GAINS                                             : 00031400             OFAULT30
      :C       N       - NUMBER OF NODES (NOT COUNTING REFERENCE NODE)                       : 00031500             OFAULT30
      :C       IP      - NUMBER OF ACCESSIBLE NODES ( " )                                    : 00031600             OFAULT30
      :C       FREQ    - FREQUENCY AT WHICH ANALYSIS IS PERFORMED (HZ)                       : 00031700             OFAULT30
      :C       Y       - MATRIX OF PASSIVE ELEMENT ADMITTANCES (SPARSE)                      : 00031800             OFAULT30
      :C       GM      - MATRIX OF TRANSCONDUCTANCES (SPARSE)                                : 00031900             OFAULT30
      :C       RM      - MATRIX OF TRANSRESISTANCES (SPARSE)                                 : 00032000             OFAULT30
      :C       XMU     - MATRIX OF VOLTAGE GAINS (SPARSE)                                    : 00032100             OFAULT30
      :C       ALPHA   - MATRIX OF CURRENT GAINS (SPARSE)                                    : 00032200             OFAULT30
      :C       YROW(.)  VECTOR STORING ROW OF Y(.)                                           : 00032300             OFAULT30
      :C       GMROW(.) VECTOR STORING ROW OF GM(.)                                          : 00032400             OFAULT30
      :C       GMCOL(.) VECTOR STORING COLUMN OF GM(.)                                       : 00032500             OFAULT30
      :C                                                                                       : 00032600             OFAULT30
      :C         SIMILARLY FOR RMROW, RMCOL, XMUROW, XMUCOL, AROW, ACOL.                       : 00032700             OFAULT30
      :C                                                                                       : 00032800             OFAULT30
      :C                                                                                       : 00032900             OFAULT30
0002  :      COMMON/CELP/  IITYPE,   Y2IP, COSTHN,   IPAR,   NPAR, ICORR,                     : 00033000   000075  INPUT
0002  :     *          IX, IW,   *,   NROW,   NCOL,      X,   XROW,   XCOL                   : 00033100   000075  INPUT
0003  :      COMPLEX   Y2IP(17,17), W(80), X(80)                                             : 00033200   000075  INPUT
0004  :      INTEGER   NROW(80), NCOL(80), XROW(80), XCOL(80)                                 : 00033300   000075  INPUT
0005  :      INTEGER   IITYPE(80), ICORR(80)                                                 : 00033400   000075  INPUT
      :C                                                                                       : 00033500             INPUT
0006  :      COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),                         : 00033600   000075  INPUT
0006  :     1              YB(65), YROW(65), GMROW(15), GMCOL(15),                          : 00033700   000075  INPUT
0006  :     2              RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),                    : 00033800   000075  INPUT
0006  :     3              AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,                   : 00033900   000075  INPUT
0006  :     4              IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR                          : 00034000   000075  INPUT
0007  :      INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,                        : 00034100   000075  INPUT
0007  :     1        AROW, ACOL, YBROW, YBCOL                                               : 00034200   000075  INPUT
0008  :      COMPLEX Y, YB                                                                   : 00034300   000075  INPUT
0009  :      COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),                     : 00034400   000075  INPUT
0009  :     1            Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),                  : 00034500   000075  INPUT
0009  :     2            A2MNUS(50), A2PLSP(50), A2MNSP(50), I, J, IP,                    : 00034600   000075  INPUT
0009  :     3            N, IB, ITOT, YN(3,3)                                             : 00034700   000075  INPUT
0010  :      INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                                          : 00034800   000075  INPUT
0011  :      INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                                          : 00034900   000075  INPUT
0012  :      COMPLEX Y, Y2, Y2I, Y2PI                                                        : 00035000   000075  INPUT
0013  :      READ(5,100) IP,N,FREQ                                                           : 00035100   000075  INPUT
0014  :      OMEGA = 2.*3.141593*FREQ                                                        : 00035200   001099  INPUT
0015  :      IB = 0                                                                          : 00035300   001009  INPUT
```

```
0016  :        IPAR = 0                                                    : 00035400   001100  INPUT
0017  :        IGM = 0                                                     : 00035500   001191  INPUT
0018  :        IRM = 0                                                     : 00035600   001232  INPUT
0019  :        IXMU = 0                                                    : 00035700   001373  INPUT
0020  :        IALPHA = 0                                                  : 00035800   001464  INPUT
0021  :        WRITE(3,102) N, IP, FREQ                                    : 00035900   001555  INPUT
0022  :    1   READ(5,101)NB,N1,N2,ITYP,VALUE                              : 00036000   002121  INPUT
0023  :        IF(NB .EQ. 99999) GO TO 200                                 : 00036100   002955  INPUT
0025  :        WRITE(3,105)NB,N1,N2,ITYP,VALUE                             : 00036200   003124  INPUT
0026  :        IF (NB - 10000) 10,20,20                                    : 00036300   003958  INPUT
0027  : 10     IB = IB+1                                                   : 00036400   004153  INPUT
0028  :          IPAR = IPAR + 1                                           : 00036500   004239  INPUT
0029  :          ITYPE(IPAR) = ITYP                                        : 00036600   004425  INPUT
0030  :          ICORR(IPAR) = IB                                          : 00036700   004609  INPUT
0031  :        IF(N1 .EQ. 0) GO TO 15                                      : 00036800   004775  INPUT
0033  :        IF (N1-IP) 16,16,17                                         : 00036900   004908  INPUT
0034  : 16     A1PLUS(NB) = N1                                             : 00037000   005039  INPUT
0035  :        GO TO 15                                                    : 00037100   005264  INPUT
0036  : 17     A2PLUS(NB) = N1-IP                                          : 00037200   005350  INPUT
0037  : 15     IF(N2.EQ.0) GO TO 18                                        : 00037300   005559  INPUT
0039  :        IF(N2-IP) 19,19,30                                          : 00037400   005692  INPUT
0040  : 19     A1MNUS(NB) = N2                                             : 00037500   005873  INPUT
0041  :        GO TO 18                                                    : 00037600   006046  INPUT
0042  : 30     A2MNUS(NB) = N2-IP                                          : 00037700   006134  INPUT
0043  : 18     GO TO (11,12,13), ITYP                                      : 00037800   006343  INPUT
0044  : 11     Y(NB) = CMPLX(1./VALUE, 0.)                                 : 00037900   006498  INPUT
0045  :        GO TO 2                                                     : 00038000   006905  INPUT
0046  : 12     Y(NB) = (0.,-1.)/(OMEGA*VALUE)                              : 00038100   006991  INPUT
0047  :        GO TO 2                                                     : 00038200   007589  INPUT
0048  : 13     Y(NB) = (0.,1.)*OMEGA*VALUE                                 : 00038300   007675  INPUT
0049  : 2      YROW(NB) = NB                                               : 00038400   008218  INPUT
0050  :        GO TO 1                                                     : 00038500   008393  INPUT
0051  : 20     IPAR = IPAR + 1                                             : 00038600   008479  INPUT
0052  :        ITYPE(IPAR) = ITYP + 10                                     : 00038700   008615  INPUT
0053  :        GO TO (21,22,23,24), ITYP                                   : 00038800   008812  INPUT
0054  : 21     IGM = IGM + 1                                               : 00038900   008985  INPUT
0055  :        ICORR(IPAR) = IGM                                           : 00039000   009121  INPUT
0056  :        GM(IGM) = VALUE                                             : 00039100   009296  INPUT
0057  :        GMROW(IGM) = N1                                             : 00039200   009471  INPUT
0058  :        GMCOL(IGM) = N2                                             : 00039300   009646  INPUT
0059  :        GO TO 1                                                     : 00039400   009821  INPUT
0060  : 22     IRM = IRM + 1                                               : 00039500   009907  INPUT
0061  :        ICORR(IPAR) = IRM                                           : 00039600   010043  INPUT
0062  :        RM(IRM) = VALUE                                             : 00039700   010218  INPUT
0063  :        RMROW(IRM) = N1                                             : 00039800   010393  INPUT
0064  :        RMCOL(IRM) = N2                                             : 00039900   010568  INPUT
0065  :        GO TO 1                                                     : 00040000   010743  INPUT
0066  : 23     IALPHA = IALPHA + 1                                         : 00040100   010829  INPUT
0067  :        ICORR(IPAR) = IALPHA                                        : 00040200   010965  INPUT
0068  :        ALPHA(IALPHA) = VALUE                                       : 00040300   011140  INPUT
0069  :        AROW(IALPHA) = N1                                           : 00040400   011315  INPUT
0070  :        ACOL(IALPHA) = N2                                           : 00040500   011490  INPUT
0071  :        GO TO 1                                                     : 00040600   011665  INPUT
0072  : 24     IXMU = IXMU + 1                                             : 00040700   011751  INPUT
0073  :        ICORR(IPAR) = IXMU                                          : 00040800   011887  INPUT
0074  :        XMU(IXMU) = VALUE                                           : 00040900   012062  INPUT
0075  :        XMUROW(IXMU) = N1                                           : 00041000   012237  INPUT
0076  :        XMUCOL(IXMU) = N2                                           : 00041100   012412  INPUT
0077  :        GO TO 1                                                     : 00041200   012587  INPUT
0078  : 200    ITOT = IGM + IRM + IALPHA + IXMU                            : 00041300   012673  INPUT
0079  :        WRITE(3,105) IB, ITOT                                       : 00041400   012889  INPUT
0080  :        WRITE(3,107) IGM, IRM, IALPHA, IXMU                         : 00041500   013321  INPUT
```

```
0081    :       RETURN                                                                  : 00041600      014021  INPUT
0082    :  100  FORMAT (2I5,E10.5)                                                      : 00041700      014135  INPUT
0083    :  101  FORMAT (4I5,E10.5)                                                      : 00041800      014135  INPUT
0084    :  102  FORMAT('1',25X,'NETWORK DATA:'//16X,I3,' NODES'/16X,I3,' ACCESSIBL: 00041900      014135  INPUT
0084    :       *E NODES'//16X,'ANALYSIS AT',E12.4,' HZ',10(/),2X,'NETWORK DATA CAR: 00042000      014135  INPUT
0084    :       *DS'//)                                                                 : 00042100      014135  INPUT
0085    :  105  FORMAT(///25X,'NETWORK INPUT COMPLETE:',I3,' BRANCHES,',                : 00042200      014135  INPUT
0085    :       1     I4,' CONTROLLED SOURCES INCLUDING:'//)                            : 00042300      014135  INPUT
0086    :  106  FORMAT(20X,I3,3I5,E16.7)                                                : 00042400      014135  INPUT
0087    :  107  FORMAT(I60,' TRANSCONDUCTANCES'/I60,' TRANSRESISTANCES'/                : 00042500      014135  INPUT
0087    :       1     I60,' CURRENT GAINS'/I60,' VOLTAGE GAINS')                        : 00042600      014135  INPUT
0088    :       END                                                                     : 00042700      014135  INPUT

NO ERRORS AND NO WARNINGS IN 88 STATEMENTS, CODE EMITTED = 14249 BITS (1782 BYTES)INPUT
COMPILE TIME IS 50.5 SECONDS FOR 133 CARDS AT 158 CARDS/MINUTE.
```

```
URROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79    05:18 PM
      /(F92000)   /FAULT30
0001  :       SUBROUTINE YBMAT                                                  : 00042800   000075  FAULT30
      :C                                                                        : 00042900           FAULT30
      :C        THIS SUBROUTINE COMPUTES MATRIX YB.  SEE REF., EQ.(3)           : 00043000           FAULT30
      :C                                                                        : 00043100           FAULT30
0002  :       COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),           : 00043200   000075  YBMAT
0002  :     1               YB(65), YROW(65), GMROW(15), GMCOL(15),             : 00043300   000075  YBMAT
0002  :     2               RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),       : 00043400   000075  YBMAT
0002  :     3               AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,      : 00043500   000075  YBMAT
0002  :     4               IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR             : 00043600   000075  YBMAT
0003  :       INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,          : 00043700   000075  YBMAT
0003  :     1         AROW, ACOL, YBROW, YBCOL                                  : 00043800   000075  YBMAT
0004  :       COMPLEX Y, YB, CMPLX                                              : 00043900   000075  YBMAT
      :C                                                                        : 00044000           YBMAT
0005  :       COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),       : 00044100   000075  YBMAT
0005  :     1         Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),           : 00044200   000075  YBMAT
0005  :     2         A2MNUS(50), A2PLSP(50), A2MNSP(50), I, J, IP,             : 00044300   000075  YBMAT
0005  :     3         N, IB, ITOT, YN(3,3)                                      : 00044400   000075  YBMAT
0006  :       INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                            : 00044500   000075  YBMAT
0007  :       INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                            : 00044600   000075  YBMAT
0008  :       COMPLEX YN, Y2, Y2I, Y2PI                                         : 00044700   000075  YBMAT
      :C                                                                        : 00044800           YBMAT
0009  :         COMMON/DELP/  ITYPE,    Y2IP, COSTMN,    JPAR,    NPAR,  ICORR,  : 00044900   000075  YBMAT
0009  :     *             IX, IW,     M,     WROW,     WCOL,      X,    XROW,   XCOL  : 00045000   000075  YBMAT
0010  :       COMPLEX    Y2IP(17,17), W(80), X(80)                              : 00045100   000075  YBMAT
0011  :       INTEGER    WROW(80), WCOL(80), XROW(80), XCOL(80)                 : 00045200   000075  YBMAT
0012  :       INTEGER ITYPE(80), ICORR(80)                                      : 00045300   000075  YBMAT
      :C                                                                        : 00045400           YBMAT
      :C       FORM X = -Y*XMU.  ALL MATRICES ARE STORED SPARSELY.              : 00045500           YBMAT
      :C       SINCE Y IS 'DIAGONAL', COMPUTATION IS SIMPLIFIED.                : 00045600           YBMAT
      :C                                                                        : 00045700           YBMAT
0013  :       IF(IXMU .EQ. 0) GO TO 11                                          : 00045800   000075  YBMAT
0015  :       DO 10 L = 1,IXMU                                                  : 00045900   000360  YBMAT
0016  :         I1 = XMUROW(L)                                                  : 00046000   000451  YBMAT
0017  :         XROW(L) = I1                                                    : 00046100   000628  YBMAT
0018  :         XCOL(L) = XMUCOL(L)                                             : 00046200   000803  YBMAT
0019  :         DO 9 K=1,IB                                                     : 00046300   001041  YBMAT
0020  :           IF(YROW(K) .EQ. I1) GO TO 10                                  : 00046400   001132  YBMAT
0022  :  9      CONTINUE                                                        : 00046500   001340  YBMAT
0023  : 10      X(L) = -Y(K)*XMU(L)                                             : 00046600   001543  YBMAT
0024  : 11    IX = IXMU                                                         : 00046700   002459  YBMAT
      :C                                                                        : 00046800           YBMAT
      :C       FORM W = -Y*RM.  ALL MATRICES ARE STORED SPARSELY.               : 00046900           YBMAT
      :C       SINCE Y IS 'DIAGONAL', COMPUTATION IS SIMPLIFIED.                : 00047000           YBMAT
      :C                                                                        : 00047100           YBMAT
0025  :       IF(IRM .EQ. 0) GO TO 13                                           : 00047200   002567  YBMAT
0027  :       DO 12 L=1,IRM                                                     : 00047300   002700  YBMAT
0028  :         I1 = RMROW(L)                                                   : 00047400   002791  YBMAT
0029  :         WROW(L) = I1                                                    : 00047500   002968  YBMAT
0030  :         WCOL(L) = RMCOL(L)                                              : 00047600   003143  YBMAT
0031  :         DO 14 K=1,IB                                                    : 00047700   003321  YBMAT
0032  :           IF(YROW(K) .EQ. I1) GO TO 12                                  : 00047800   003472  YBMAT
0034  : 14      CONTINUE                                                        : 00047900   003680  YBMAT
0035  : 12      W(L) = -Y(K)*RM(L)                                              : 00048000   003883  YBMAT
0036  : 13    IW = IRM                                                          : 00048100   004799  YBMAT
      :C                                                                        : 00048200           YBMAT
      :C       FORM X = Y + X + GM.  ALL MATRICES ARE STORED SPARSELY.          : 00048300           YBMAT
      :C       SINCE Y IS 'DIAGONAL', AND X AND GM HAVE NEITHER DIAGONAL        : 00048400           YBMAT
      :C     . ELEMENTS NOR ELEMENTS IN COMMON, COMPUTATION IS SIMPLIFIED.      : 00048500           YBMAT
      :C                                                                        : 00048600           YBMAT
0037  :       DO 20 L=1,IB                                                      : 00048700   004907  YBMAT
```

```
0038    :         X(IX+L) = Y(L)                                                    : 000043800    004998   YBMAT
0039    :         YROW(IX+L) = YROW(L)                                              : 000043900    005514   YBMAT
0040    :  20     XCOL(IX+L) = YROW(L)                                              : 000049000    005786   YBMAT
0041    :       IX = IX + IB                                                        : 000049100    006202   YBMAT
0042    :         IF(IGM .EQ. 0) GO TO 31                                           : 000049200    006350   YBMAT
0044    :         DO 30 L=1,IGM                                                     : 000049300    006425   YBMAT
0045    :         X(IX+L) = CMPLX( GM(L), 0.)                                       : 000049400    005574   YBMAT
0046    :         YROW(IX+L) = GMROW(L)                                             : 000049500    007030   YBMAT
0047    :  30     XCOL(IX+L) = GMCOL(L)                                             : 000049600    007302   YBMAT
0048    :  31   IX = IX + IGM                                                       : 000049700    007713   YBMAT
        :C                                                                          : 000049800             YBMAT
        :C        FORM W = 1 + W + ALPHA.  ALL MATRICES ARE STORED SPARSELY.        : 000049900             YBMAT
        :C        SINCE W AND ALPHA HAVE NO COMMON ELEMENTS AND NO DIAGONAL         : 000050000             YBMAT
        :C        ELEMENTS, COMPUTATION IS SIMPLIFIED.                              : 000050100             YBMAT
        :C                                                                          : 000050200             YBMAT
0049    :       DO 40 L=1,IB                                                        : 000050300    007866   YBMAT
0050    :         W(IW+L) = (1.,0.)                                                 : 000050400    007957   YBMAT
0051    :         WROW(IW+L) = L                                                    : 000050500    008354   YBMAT
0052    :  40     WCOL(IW+L) = L                                                    : 000050600    008573   YBMAT
0053    :       IW = IW + IB                                                        : 000050700    008926   YBMAT
0054    :         IF(IALPHA .EQ. 0) GO TO 51                                        : 000050800    009074   YBMAT
0056    :         DO 50 L=1,IALPHA                                                  : 000050900    009207   YBMAT
0057    :         W(IW+L) = CMPLX( -ALPHA(L), 0.)                                   : 000051000    009298   YBMAT
0058    :         WROW(IW+L) = AROW(L)                                              : 000051100    009763   YBMAT
0059    :  50     WCOL(IW+L) = ACOL(L)                                              : 000051200    010035   YBMAT
0060    :  51   IW = IW + IALPHA                                                    : 000051300    010451   YBMAT
        :C                                                                          : 000051400             YBMAT
        :C         FORM YB = W*X.  ALL MATRICES ARE STORED SPARSELY.                : 000051500             YBMAT
        :C         COMPUTATION IS COMPLICATED BY THE FACT THAT EXISTING             : 000051600             YBMAT
        :C         ELEMENTS MUST BE SOUGHT BEFORE THE MULTIPLICATION                : 000051700             YBMAT
        :C         CAN BE PERFORMED.                                                : 000051800             YBMAT
        :C                                                                          : 000051900             YBMAT
0061    :       IYB = 0                                                             : 000052000    010599   YBMAT
0062    :       DO 90 K=1,IB                                                        : 000052100    010830   YBMAT
0063    :         DO 80 L=1,IW                                                      : 000052200    010781   YBMAT
0064    :           IF(WCOL(L) - K) 80,55,80                                        : 000052300    010872   YBMAT
0065    :  55         DO 70 M=1,IX                                                  : 000052400    011118   YBMAT
0066    :             IF(XROW(M) - K) 70,65,70                                      : 000052500    011207   YBMAT
0067    :  65           IF(IYB .EQ. 0) GO TO 67                                     : 000052600    011451   YBMAT
0069    :               DO 60 N1=1,IYB                                              : 000052700    011504   YBMAT
0070    :                 IF(YBROW(N1) - WROW(L)) 60,75,60                          : 000052800    011573   YBMAT
0071    :  75             IF(YBCOL(N1) - XCOL(M)) 60,85,60                          : 000052900    011982   YBMAT
0072    :  60           CONTINUE                                                    : 000053000    012209   YBMAT
0073    :  67           IYB = IYB + 1                                               : 000053100    012492   YBMAT
0074    :                YB(IYB) = W(L)*X(M)                                        : 000053200    012828   YBMAT
0075    :                YBROW(IYB) = WROW(L)                                       : 000053300    013205   YBMAT
0076    :                YBCOL(IYB) = XCOL(M)                                       : 000053400    013743   YBMAT
0077    :                GO TO 70                                                   : 000053500    013981   YBMAT
0078    :  85           YB(N1) = YB(N1) + W(L)*X(M)                                : 000053600    014067   YBMAT
0079    :  70         CONTINUE                                                     : 000053700    015213   YBMAT
0080    :  80       CONTINUE                                                       : 000053800    015416   YBMAT
0081    :  90     CONTINUE                                                         : 000053900    015619   YBMAT
0082    :       RETURN                                                             : 000054000    015822   YBMAT
0083    :       END                                                                : 000054100    015935   YBMAT

NO ERRORS AND NO WARNINGS IN 83 STATEMENTS, CODE EMITTED = 15995 BITS (2000 BYTES)YBMAT
COMPILE TIME IS 46.1 SECONDS FOR 114 CARDS AT 148 CARDS/MINUTE.
```

```
BURROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
     /(E99000)   /FAULT30
0001   :       SUBROUTINE ANEW                                                  : 000054200    000075   OFAULT30
       :C                                                                       : 000054300              OFAULT30
       :C      THIS SUBROUTINE COMPUTES THE NODE-TO-BRANCH INCIDENCE MATRICES   : 000054400              OFAULT30
       :C      (SPARSELY STORED VECTORS A1PLSP, A1MNSP, A2PLSP, A2MNSP) FOR     : 000054500              OFAULT30
       :C      THE NETWORK WITH NODES I AND J SHORTED.  SEE REF., EQ.(13).      : 000054600              OFAULT30
       :C                                                                       : 000054700              OFAULT30
0002   :       COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),     : 000054800    000075   ANEW
0002   :     1             Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),    : 000054900    000075   ANEW
0002   :     2              A2MNUS(50), A2PLSP(50), A2MNSP(50), I, J, IP,     : 000055000    000075   ANEW
0002   :     3              N, IB, ITOT, YN(3,3)                              : 000055100    000075   ANEW
0003   :       INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                           : 000055200    000075   ANEW
0004   :       INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                           : 000055300    000075   ANEW
0005   :       COMPLEX YN, Y2, Y2I, Y2PI                                        : 000055400    000075   ANEW
0006   :       DO 150 L=1,IB                                                    : 000055500    000075   ANEW
0007   :       A1PLSP(L) = A1PLUS(L)                                            : 000055600    000318   ANEW
0008   :       A1MNSP(L) = A1MNUS(L)                                            : 000055700    000556   ANEW
0009   :       IF(A2PLUS(L) - (J-IP)) 20,30,40                                  : 000055800    000794   ANEW
0010   :  20   A2PLSP(L) = A2PLUS(L)                                            : 000055900    001072   ANEW
0011   :       GO TO 90                                                         : 000056000    001310   ANEW
0012   :  30   IF(I-IP) 50,50,60                                               : 000056100    001395   ANEW
0013   :  50   IF(I .EQ. 0) GO TO 55                                           : 000056200    001577   ANEW
0015   :       IF(A1MNSP(L) - I) 52,51,52                                       : 000056300    001710   ANEW
0016   :  51   A1MNSP(L) = 0                                                   : 000056400    001954   ANEW
0017   :       GO TO 55                                                         : 000056500    002117   ANEW
0018   :  52   A1PLSP(L) = I                                                   : 000056600    002203   ANEW
0019   :  55   A2PLSP(L)  = 0                                                  : 000056700    002378   ANEW
0020   :       GO TO 90                                                         : 000056800    002541   ANEW
0021   :  60   IF(A2MNUS(L) - (I-IP)) 80,70,80                                : 000056900    002627   ANEW
0022   :  70   A2PLSP(L) = 0                                                   : 000057000    002905   ANEW
0023   :       A2MNSP(L) = 0                                                    : 000057100    003068   ANEW
0024   :       GO TO 90                                                         : 000057200    003231   ANEW
0025   :  80   A2PLSP(L) = I-IP                                                : 000057300    003317   ANEW
0026   :       GO TO 90                                                         : 000057400    003522   ANEW
0027   :  40   A2PLSP(L) = A2PLUS(L) - 1                                       : 000057500    003612   ANEW
0028   :  90   IF(A2MNUS(L) - (J-IP)) 100,110,120                             : 000057600    003872   ANEW
0029   : 100   A2MNSP(L) = A2MNUS(L)                                           : 000057700    004150   ANEW
0030   :       GO TO 150                                                        : 000057800    004388   ANEW
0031   : 110   IF(I-IP) 130,130,135                                            : 000057900    004474   ANEW
0032   : 130   IF(I .EQ. 0) GO TO 134                                          : 000058000    004555   ANEW
0034   :       IF(A1PLSP(L) - I) 132,131,132                                    : 000058100    004788   ANEW
0035   : 131   A1PLSP(L) = 0                                                   : 000058200    005032   ANEW
0036   : 132   A1MNSP(L) = I                                                   : 000058300    005195   ANEW
0037   : 134   A2MNSP(L) = 0                                                   : 000058400    005370   ANEW
0038   :       GO TO 150                                                        : 000058500    005533   ANEW
0039   : 135   IF(A2PLUS(L) - (I-IP)) 145,140,145                              : 000058600    005619   ANEW
0040   : 140   A2PLSP(L) = 0                                                   : 000058700    005897   ANEW
0041   :       A2MNSP(L) = 0                                                    : 000058800    006060   ANEW
0042   :       GO TO 150                                                        : 000058900    006223   ANEW
0043   : 145   A2MNSP(L) = I-IP                                                : 000059000    006309   ANEW
0044   :       GO TO 150                                                        : 000059100    006518   ANEW
0045   : 120   A2MNSP(L) = A2MNUS(L) - 1                                       : 000059200    006604   ANEW
0046   : 150   CONTINUE                                                        : 000059300    006754   ANEW
0047   :       RETURN                                                           : 000059400    007067   ANEW
0048   :       END                                                              : 000059500    007181   ANEW

NO ERRORS AND NO WARNINGS IN 48 STATEMENTS, CODE EMITTED = 7240 BITS (905 BYTES)ANEW
COMPILE TIME IS 35.5 SECONDS FOR 54 CARDS AT 92 CARDS/MINUTE.
```

```
BURROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
     /(E99000)   /FAULT30
0001 :       SUBROUTINE YMAT(A1PLSP,A1MNSP,A2PLSP,A2MNSP,Y2PI,YN,IPP,IB,IQT)  : 00059600    000075  OFAULT30
     :C                                                                       : 00059700            OFAULT30
     :C      THIS SUBROUTINE COMPUTES THE Y-PARAMETERS AT THE ACCESSIBLE      : 00059800            OFAULT30
     :C      NODES.  SEE REF., EQ.(8).  SPARSE MATRIX TECHNIQUES ARE          : 00059900            OFAULT30
     :C      USED, AS DESCRIBED IN REF., PP. 18-22.                           : 00060000            OFAULT30
     :C                                                                       : 00060100            OFAULT30
0002 :       COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),         : 00060200    000075  YMAT
0002 :      1              YB(65), YROW(65), GMROW(15), GMCOL(15),           : 00060300    000075  YMAT
0002 :      2              RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),     : 00060400    000075  YMAT
0002 :      3              AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,    : 00060500    000075  YMAT
0002 :      4              IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR           : 00060600    000075  YMAT
0003 :       INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,        : 00060700    000075  YMAT
0003 :      1        AROW, ACOL, YBROW, YBCOL                                 : 00060800    000075  YMAT
0004 :       COMPLEX Y, YB                                                    : 00060900    000075  YMAT
0005 :       COMPLEX Y2PI(IQT,IQT),YN(3,3)                                    : 00061000    000075  YMAT
0006 :       INTEGER A2PLSP(50),A2MNSP(50)                                    : 00061100    000075  YMAT
0007 :       INTEGER A1PLSP(50),A1MNSP(50)                                    : 00061200    000075  YMAT
     :C                                                                       : 00061220            YMAT
0008 :       COMMON/DEL/ C,D                                                  : 00061240    000075  YMAT
     :C                                                                       : 00061260            YMAT
0009 :       COMPLEX W,C(50,50),D(50,50),E(50,50)                             : 00061300    000075  YMAT
     :C                                                                       : 00061400            YMAT
0010 :       DO 100 I=1,IB                                                    : 00061500    000075  YMAT
0011 :         I1 = A2PLSP(I)                                                 : 00061600    000421  YMAT
0012 :         I2 = A2MNSP(I)                                                 : 00061700    000564  YMAT
0013 :       DO 100 J=1,IB                                                    : 00061800    000747  YMAT
0014 :           D(I,J) = (0.,0.)                                           : 00061900    000838  YMAT
0015 :           E(I,J) = (0.,0.)                                           : 00062000    001283  YMAT
0016 :         J1 = A2PLSP(J)                                                 : 00062100    001688  YMAT
0017 :         J2 = A2MNSP(J)                                                 : 00062200    001851  YMAT
0018 :         W = (0.,0.)                                                    : 00062300    002014  YMAT
0019 :         IF(I1*J1 .NE. 0) W = W + Y2PI(I1,J1)                           : 00062400    002211  YMAT
0021 :         IF(I2*J2 .NE. 0) W = W + Y2PI(I2,J2)                           : 00062500    002509  YMAT
0023 :         IF(I1*J2 .NE. 0) W = W - Y2PI(I1,J2)                           : 00062600    003407  YMAT
0025 :         IF(I2*J1 .NE. 0) W = W - Y2PI(I2,J1)                           : 00062700    004069  YMAT
0027 : 100   C(I,J) = W                                                       : 00062800    004731  YMAT
     :C                                                                       : 00062900            YMAT
0028 :       DO 5 I=1,IB                                                      : 00063000    005416  YMAT
0029 :       DO 5 L=1,IYB                                                     : 00063100    005507  YMAT
0030 :         K = YBROW(L)                                                   : 00063200    005598  YMAT
0031 :         J = YBCOL(L)                                                   : 00063300    005775  YMAT
0032 :   5   D(I,J) = D(I,J) - C(I,K)*YB(L)                                  : 00063400    005952  YMAT
     :C                                                                       : 00063500            YMAT
0033 :       DO 10 K=1,IB                                                     : 00063600    007542  YMAT
0034 :  10   D(K,K) = D(K,K) + (1.,0.)                                        : 00063700    007633  YMAT
     :C                                                                       : 00064000            YMAT
0035 :       DO 15 J=1,IB                                                     : 00064100    008514  YMAT
0036 :       DO 15 L=1,IYB                                                    : 00064200    008605  YMAT
0037 :         I = YBROW(L)                                                   : 00064300    008696  YMAT
0038 :         K = YBCOL(L)                                                   : 00064400    008873  YMAT
0039 :  15   E(I,J) = E(I,J) + YB(L)*D(K,J)                                  : 00064500    009050  YMAT
     :C                                                                       : 00064600            YMAT
0040 :       DO 20 I=1,IPP                                                    : 00064700    010640  YMAT
0041 :       DO 20 J=1,IPP                                                    : 00064800    010731  YMAT
0042 :  20   YN(I,J) = (0.,0.)                                               : 00064900    010822  YMAT
     :C                                                                       : 00065000            YMAT
     :C     COMPUTE A1 * E * A1T.  A1 IS STORED SPARSELY.                     : 00065100            YMAT
     :C                                                                       : 00065200            YMAT
0043 :       DO 30 L=1,IB                                                     : 00065300    011485  YMAT
```

```
0044   :          IP = A1PLSP(L)                                                    : 00065400    011576  YMAT
0045   :          IM = A1MNSP(L)                                                    : 00065500    011739  YMAT
0046   :       DO 30 K=1,IB                                                         : 00065600    011902  YMAT
0047   :          JP = A1PLSP(K)                                                    : 00065700    011993  YMAT
0048   :          JM = A1MNSP(K)                                                    : 00065800    012156  YMAT
0049   :          IF(IP*JP .NE. 0) YN(IP,JP) = YN(IP,JP) + E(L,K)                   : 00065900    012319  YMAT
0051   :          IF(IM*JM .NE. 0) YN(IM,JM) = YN(IM,JM) + E(L,K)                   : 00066000    013348  YMAT
0053   :          IF(IP*JM .NE. 0) YN(IP,JM) = YN(IP,JM) - E(L,K)                   : 00066100    014377  YMAT
0055   :          IF(IM*JP .NE. 0) YN(IM,JP) = YN(IM,JP) - E(L,K)                   : 00066200    015406  YMAT
0057   : 30    CONTINUE                                                            : 00066300    016435  YMAT
0058   :       RETURN                                                              : 00066400    016782  YMAT
0059   :       END                                                                 : 00066500    016896  YMAT

ERRORS AND NO WARNINGS IN 59 STATEMENTS, CODE EMITTED = 16955 BITS (2120 BYTES)YMAT
COMPILE TIME IS 40.5 SECONDS FOR 71 CARDS AT 105 CARDS/MINUTE.
```

```
BURROUGHS B1800/B1700 FORTRAN COMPILER - MARK 8.0 RA 11/23/78 01:37 -        TUESDAY   10/02/79    05:16 PM
      /(E99000)  /FAULT3D
0001  :       SUBROUTINE Y2PMAT(Y2P,IQP,IQPT,A2PLSP,A2MNSP)                          : 00066600    000075  OFAULT3D
      :C                                                                           : 00066700            OFAULT3D
      :C       THIS SUBROUTINE FORMS EITHER Y2 = A2 * YB * A2T OR                  : 00066800            OFAULT3D
      :C       Y2P = A2P * YB * A2PT.  SEE REF., EQ.(2).  SPARSE MATRIX            : 00066900            OFAULT3D
      :C       TECHNIQUES HAVE BEEN USED TO STORE YB.                              : 00067000            OFAULT3D
      :C                                                                           : 00067100            OFAULT3D
0002  :       COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),             : 00067200    000075  Y2PMAT
0002  :      1              YB(65), YROW(65), GMROW(15), GMCOL(15),               : 00067300    000075  Y2PMAT
0002  :      2              RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),         : 00067400    000075  Y2PMAT
0002  :      3              AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,        : 00067500    000075  Y2PMAT
0002  :      4              IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR               : 00067600    000075  Y2PMAT
0003  :       INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,            : 00067700    000075  Y2PMAT
0003  :      1 AROW, ACOL, YBROW, YBCOL                                            : 00067800    000075  Y2PMAT
0004  :       COMPLEX Y, YB                                                       : 00067900    000075  Y2PMAT
0005  :       COMPLEX Y2P(IQPT,IQPT)                                              : 00068000    000075  Y2PMAT
0006  :       INTEGER A2PLSP(50), A2MNSP(50)                                      : 00068100    000075  Y2PMAT
      :C                                                                           : 00068200            Y2PMAT
      :C       INITIALIZE Y2P TO 0.                                                : 00068300            Y2PMAT
      :C                                                                           : 00068400            Y2PMAT
0007  :       DO 10 I=1,IQP                                                       : 00068500    000075  Y2PMAT
0008  :       DO 10 J=1,IQP                                                       : 00068600    000401  Y2PMAT
0009  :  10   Y2P(I,J) = (0.,0.)                                                  : 00068700    000492  Y2PMAT
      :C                                                                           : 00068800            Y2PMAT
      :C       COMPUTE Y2P = A2 * YB * A2T                                         : 00068900            Y2PMAT
      :C                                                                           : 00069000            Y2PMAT
0010  :       DO 100 M=1,IYB                                                      : 00069100    001155  Y2PMAT
0011  :        L = YBROW(M)                                                       : 00069200    001246  Y2PMAT
0012  :        K = YBCOL(M)                                                       : 00069300    001423  Y2PMAT
0013  :        IP  = A2PLSP(L)                                                    : 00069400    001600  Y2PMAT
0014  :        IM = A2MNSP(L)                                                     : 00069500    001763  Y2PMAT
0015  :        JP  = A2PLSP(K)                                                    : 00069600    001926  Y2PMAT
0016  :        JM = A2MNSP(K)                                                     : 00069700    002089  Y2PMAT
0017  :        IF(IP*JP .NE. 0) Y2P(IP,JP) = Y2P(IP,JP) + YB(M)                   : 00069800    002252  Y2PMAT
0019  :        IF(IM*JM .NE. 0) Y2P(IM,JM) = Y2P(IM,JM) + YB(M)                   : 00069900    003229  Y2PMAT
0021  :        IF(IP*JM .NE. 0) Y2P(IP,JM) = Y2P(IP,JM) - YB(M)                   : 00070000    004206  Y2PMAT
0023  :        IF(IM*JP .NE. 0) Y2P(IM,JP) = Y2P(IM,JP) - YB(M)                   : 00070100    005183  Y2PMAT
0025  :  100  CONTINUE                                                           : 00070200    006160  Y2PMAT
0026  :       RETURN                                                              : 00070300    006363  Y2PMAT
0027  :       END                                                                 : 00070400    006477  Y2PMAT

0 ERRORS AND NO WARNINGS IN 27 STATEMENTS,CODE EMITTED = 6536 BITS (817 BYTES)Y2PMAT
COMPILE TIME IS 31.3 SECONDS FOR 39 CARDS AT 75 CARDS/MINUTE.
```

```
0001  :       SUBROUTINE SNEW                                                  : 00070500  000075  OFAULT30
      :C                                                                       : 00070600          OFAULT30
      :C       THIS SUBROUTINE FORMS THE INVERSE OF Y2P, CALLED Y2PI,          : 00070700          OFAULT30
      :C       WITH NODES I AND J SHORTED, USING EQ.(20) IN THE REF.           : 00070800          OFAULT30
      :C       SPARSE MATRIX TECHNIQUES ARE USED.  SEE REF., PP. 14-18.        : 00070900          OFAULT30
      :C                                                                       : 00071000          OFAULT30
0002  :       COMPLEX YN, Y2, Y2I, Y2PI                                        : 00071100  000075  SNEW
0003  :       COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),      : 00071200  000075  SNEW
0003  :     1             Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),     : 00071300  000075  SNEW
0003  :     2             A2MNUS(50), A2PLSP(50), A2MNSP(50), I, J, IP,       : 00071400  000075  SNEW
0003  :     3             N, IB, ITOT, YN(3,3)                                : 00071500  000075  SNEW
0004  :       INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                           : 00071600  000075  SNEW
0005  :       INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                           : 00071700  000075  SNEW
0006  :       COMPLEX XK(16,16),XM(16),C(16),XI(16,16),W,MC                    : 00071800  000075  SNEW
0007  :       INTEGER V(17), XIROW(16)                                         : 00071900  000075  SNEW
      :C                                                                       : 00072000          SNEW
      :C       FORM VECTOR V.  SEE REF., P. 17                                 : 00072100          SNEW
      :C                                                                       : 00072200          SNEW
0008  :       DO 10 K=1,17                                                     : 00072300  000075  SNEW
0009  : 10    V(K) = 0                                                         : 00072400  000318  SNEW
0010  :       IF(J .EQ. IP+1) GO TO 25                                         : 00072500  000613  SNEW
0012  :       DO 20 K=1,J-IP-1                                                 : 00072600  000869  SNEW
0013  : 20    V(K) = K                                                         : 00072700  001164  SNEW
0014  : 25    IQ = N-IP                                                        : 00072800  001386  SNEW
0015  :       DO 30 K = J-IP,IQ-1                                              : 00072900  001534  SNEW
0016  : 30    V(K) = K+1                                                       : 00073000  001875  SNEW
0017  :       V(IQ) = J-IP                                                     : 00073100  002099  SNEW
      :C                                                                       : 00073200          SNEW
      :C       FORM MATRICES K AND M, DEFINED IN REF., EQ.(22).  SPARSE        : 00073300          SNEW
      :C       MATRIX TECHNIQUES ARE USED.  SEE REF., EQ.(24) AND (25).        : 00073400          SNEW
      :C                                                                       : 00073500          SNEW
0018  :       DO 50 IR=1,IQ                                                    : 00073600  002308  SNEW
0019  :       DO 50 IS=1,IQ-1                                                  : 00073700  002399  SNEW
0020  :       W = Y2I(V(IR),V(IS))                                             : 00073800  002583  SNEW
0021  :       IF(IR-IQ) 60,65,65                                               : 00073900  003167  SNEW
0022  : 60    XK(IR,IS) = W                                                    : 00074000  003348  SNEW
0023  :       GO TO 50                                                         : 00074100  003745  SNEW
0024  : 65    XM(IS) = W-Y2I(I-IP,V(IS))                                       : 00074200  003831  SNEW
0025  : 50    CONTINUE                                                         : 00074300  004649  SNEW
      :C                                                                       : 00074400          SNEW
      :C       FORM R22 * Y2 * U.  SEE REF., EQ.(26).                          : 00074500          SNEW
      :C                                                                       : 00074600          SNEW
0026  :       DO 70 M=1,IQ-1                                                   : 00074700  004879  SNEW
0027  : 70    C(M) = Y2(V(M),J-IP)                                             : 00074800  005163  SNEW
0028  :       IF(I .GT. IP) C(I-IP) = C(I-IP) + Y2(J-IP,J-IP)                  : 00074900  005821  SNEW
      :C                                                                       : 00075000          SNEW
      :C       FORM 1 - M * (R22 * Y2 * U).  SEE REF., EQ.(21).                : 00075100          SNEW
      :C                                                                       : 00075200          SNEW
0030  :       MC = (1.,0.)                                                     : 00075300  006980  SNEW
0031  :       DO 80 M=1,IQ-1                                                   : 00075400  007157  SNEW
0032  :       IF(REAL(C(M)) .NE. 0. .AND. AIMAG(C(M)) .NE. 0.) MC=MC-XM(M)*C(M) : 00075500  007441  SNEW
0034  : 80    CONTINUE                                                         : 00075600  008902  SNEW
      :C                                                                       : 00075700          SNEW
      :C       FORM XI = C * XM / MC.   SINCE C HAS MANY ZERO ELEMENTS,        : 00075800          SNEW
      :C       XI HAS MANY ZERO ROWS.  ONLY THE NON-ZERO ROWS OF XI ARE        : 00075900          SNEW
      :C       FORMED BELOW.  SEE REF., EQ. (21).                              : 00076000          SNEW
      :C                                                                       : 00076100          SNEW
0035  :       IROW = 0                                                         : 00075200  008988  SNEW
0036  :       DO 85 K=1,IQ-1                                                   : 00075300  009079  SNEW
0037  :         IF(REAL(C(K)) .EQ. 0. .AND. AIMAG(C(K)) .EQ. 0.) GO TO 85      : 00076400  009363  SNEW
```

```
0039  :          IROW = IROW + 1                                          : 000076500    010077  SNEW
0040  :          XIROW(IROW) = K                                          : 000076600    010213  SNEW
0041  :          DO 85 M=1,IQ-1                                           : 000076700    010388  SNEW
0042  :          XI(IROW,M) = C(K)*XM(M)/MC                               : 000076800    010672  SNEW
0043  :  85      CONTINUE                                                 : 000076900    011749  SNEW
      :C                                                                  : 000077000            SNEW
      :C         FORM Y2PI = XK * (XI + 1).   RECALL THAT ONLY THE NON-   : 000077100            SNEW
      :C         ZERO ROWS OF XI HAVE BEEN SAVED.  SEE REF., EQ.(21).     : 000077200            SNEW
      :C                                                                  : 000077300            SNEW
0044  :          DO 95 I1=1,IQ-1                                          : 000077400    011862  SNEW
0045  :          DO 95 J1=1,IC-1                                          : 000077500    012146  SNEW
0046  :           W = (0.,0.)                                             : 000077600    012430  SNEW
0047  :           DO 90 L=1,IROW                                          : 000077700    012627  SNEW
0048  :            K = XIROW(L)                                           : 000078000    012718  SNEW
0049  :  90        W = W + XK(I1,K)*XI(L,J1)                              : 000078000    012895  SNEW
0050  :  95      Y2PI(I1,J1) = W + XK(I1,J1)                              : 000078000    013912  SNEW
0051  :          RETURN                                                   : 000078100    014675  SNEW
0052  :          END                                                      : 000078200    014789  SNEW

0 ERRORS AND NO WARNINGS IN 52 STATEMENTS, CODE EMITTED = 14848 BITS (1856 BYTES)SNEW
OMPILE TIME IS 39.0 SECONDS FOR 78 CARDS AT 120 CARDS/MINUTE.
```

```
URROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
      /(E99000)  /FAULT30
0001  :C       SUBROUTINE MINVC(B,C,NN,NT)                                  : 00078300   000075  OFAULT30
      :C                                                                    : 00078400          OFAULT30
      :C         THIS SUBROUTINE INVERTS THE COMPLEX VALUED MATRIX B.       : 00078500          OFAULT30
      :C         IN THE CALLING PROGRAM, B HAS BEEN DIMENSIONED NT X NT.    : 00078600          OFAULT30
      :C         THE ACTUAL DIMENSION USED IN THE CALLING PROGRAM IS NN X NN. : 00078700         OFAULT30
      :C         THE INVERSE OF B IS RETURNED AS C, WHICH IS DIMENSIONED    : 00078800          OFAULT30
      :C         AS B.                                                      : 00078900          OFAULT30
      :C                                                                    : 00079000          OFAULT30
0002  :         COMPLEX ALPHA(17),BETA(17),SUM,SUMP,U(17),V(17),LAMBDA      : 00079100   000075  MINVC
0003  :         COMPLEX A(17,17),B(NT,NT),C(NT,NT)                          : 00079200   000075  MINVC
0004  :         DO 1 I=1,NN                                                : 00079300   000075  MINVC
0005  :         DO 1 J=1,NN                                                : 00079400   000451  MINVC
0006  :         C(I,J) = (0.,0)                                            : 00079500   000542  MINVC
0007  :     1   A(I,J) = B(I,J)                                            : 00079600   000917  MINVC
0008  :         DO 3 I=1,NN                                                : 00079700   001741  MINVC
0009  :         A(I,I) = A(I,I) - (1.,0.)                                  : 00079800   001832  MINVC
0010  :     3   C(I,I) = (1.,0.)                                           : 00079900   002569  MINVC
0011  :         DO 40 N = 1,NN                                             : 00080000   003088  MINVC
0012  :         DO 10 I = N+1,NN                                           : 00080100   003179  MINVC
0013  :         ALPHA(I) = A(I,N)/A(N,N)                                   : 00080200   003715  MINVC
0014  :    10   BETA(I) = A(N,I)                                           : 00080300   004298  MINVC
0015  :         DO 17 I=1,NN                                               : 00080400   004974  MINVC
0016  :         SUM = (0.,0.)                                              : 00080500   005065  MINVC
0017  :         SUMP = (0.,0.)                                             : 00080600   005262  MINVC
0018  :         DO 15 K=N,NN                                               : 00080700   005459  MINVC
0019  :         SUM = SUM + C(I,K)*A(K,N)                                  : 00080800   005556  MINVC
0020  :    15   SUMP = SUMP + A(N,K)*C(K,I)                                : 00080900   005779  MINVC
0021  :         U(I) = SUM                                                 : 00081000   007346  MINVC
0022  :    17   V(I) = SUMP                                                : 00081100   007691  MINVC
0023  :         SUM = (0.,0.)                                              : 00081200   008180  MINVC
0024  :         DO 21 J = N,NN                                             : 00081300   008377  MINVC
0025  :    21   SUM = SUM + V(J)*A(J,N)                                    : 00081400   008474  MINVC
0026  :         LAMBDA = SUM + A(N,N)                                      : 00081500   009439  MINVC
0027  :         DO 30 I = 1,NN                                             : 00081600   009920  MINVC
0028  :         DO 30 J = 1,NN                                             : 00081700   010011  MINVC
0029  :    30   C(I,J) = C(I,J) - U(I)*V(J)/LAMBDA                         : 00081800   010102  MINVC
0030  :         IF(N .EQ. NN) RETURN                                       : 00081900   011688  MINVC
0032  :         DO 38 I = N+1,NN                                           : 00082000   011947  MINVC
0033  :         DO 38 J = N+1,NN                                           : 00082100   012083  MINVC
0034  :    38   A(I,J) = A(I,J) - ALPHA(I)*BETA(J)                         : 00082200   012219  MINVC
0035  :    40   CONTINUE                                                   : 00082300   013757  MINVC
0036  :         END                                                        : 00082400   013960  MINVC

 ERRORS AND NO WARNINGS IN 36 STATEMENTS,CODE EMITTED = 14074 BITS (1760 BYTES)MINVC
MPILE TIME IS 31.3 SECONDS FOR 42 CARDS AT 81 CARDS/MINUTE.
```

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

```
UROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
     /(599999)  /FAULT30
0001    :          FUNCTION COSTF(YNNOM)                                          : 00082500    000075  DFAULT30
0002    :          COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),   : 00082600    000075  COSTF
0002    :     1                Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),  : 00082700    000075  COSTF
0002    :     2                A2MNUS(50), A2PLSP(50), A2MNSP(50), II, JJ, IP,  : 00082800    000075  COSTF
0002    :     3                N, IB, ITOT, YN(3,3)                             : 00082900    000075  COSTF
0003    :          INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                         : 00083000    000075  COSTF
0004    :          INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                         : 00083100    000075  COSTF
0005    :          COMPLEX YN, Y2, Y2I, Y2PI                                      : 00083200    000075  COSTF
0006    :          COMPLEX YNNOM(3,3)                                             : 00083300    000075  COSTF
0007    :          DIMENSION DENOM(3,3)                                           : 00083400    000075  COSTF
0008    :          COSTF = 0.                                                     : 00083500    000075  COSTF
0009    :          DO 10 I = 1,IP                                                 : 00083600    000411  COSTF
0010    :          DO 10 J = 1,IP                                                 : 00083700    000502  COSTF
0011    :            D = CABS(YNNOM(I,J))                                         : 00083800    000593  COSTF
0012    :            IF(D .EQ. 0.) D = 1.E-9                                      : 00083900    000990  COSTF
0014    :    10    DENOM(I,J) = D                                                 : 00084000    001277  COSTF
0015    :          DO 20 I = 1,IP                                                 : 00084100    001792  COSTF
0016    :          DO 20 J = 1,IP                                                 : 00084200    001863  COSTF
0017    :    20    COSTF = COSTF + CABS(YN(I,J) - YNNOM(I,J))/DENOM(I,J)          : 00084300    001974  COSTF
0018    :          RETURN                                                         : 00084400    003218  COSTF
0019    :          END                                                            : 00084500    003332  COSTF

0 ERRORS AND NO WARNINGS IN 19 STATEMENTS,CODE EMITTED = 3391 BITS (424 BYTES)COSTF
OMPILE TIME IS 25.2 SECONDS FOR 21 CARDS AT 50 CARDS/MINUTE.
```

```
IROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
     /(F99000)  /FAULT30
0001   :          SUBROUTINE SEARCH(A, B, H, K, X, Y)                                  000084600    000075  FAULT30
0002   :          LOGICAL K                                                            000084700    000075  SEARCH
0003   :          REAL L1, L2                                                          000084800    000075  SEARCH
       :C                                                                              000084900            SEARCH
       :C         THIS SUBROUTINE PERFORMS A FIBONACI SEARCH FOR A MINIMUM             000085000            SEARCH
       :C         OF THE FUNCTION F(.) OVER THE INTERVAL (A,B).  THE FINAL             000085100            SEARCH
       :C         INTERVAL IS NO LARGER THAN H.  Y(X) IS THE APPROXIMATE               000085200            SEARCH
       :C         MAXIMUM OF THE FUNCTION.                                             000085300            SEARCH
       :C                                                                              000085400            SEARCH
0004   :          C = B - A                                                            000085500    000075  SEARCH
0005   :          FN = C/H                                                             000085600    000399  SEARCH
0006   :          G1 = 1.                                                              000085700    000547  SEARCH
0007   :          G2 = 1.                                                              000085800    000675  SEARCH
0008   :          DO 10 I=2,25                                                         000085900    000803  SEARCH
0009   :            G3 = G1 + G2                                                       000086000    000698  SEARCH
0010   :            G = G2                                                             000086100    001046  SEARCH
0011   :            G2 = G3                                                            000086200    001143  SEARCH
0012   :            G1 = G                                                             000086300    001240  SEARCH
0013   :            IF(G2 .GE. FN) GO TO 20                                            000086400    001337  SEARCH
0015   :     10   CONTINUE                                                            000086500    001482  SEARCH
0016   :     20   L2 = G1/G2                                                          000086600    001673  SEARCH
0017   :          L1 = 1.                                                              000086700    001821  SEARCH
0018   :          IF(K) WRITE(3,100)                                                   000086800    001949  SEARCH
0020   :          X = A + L2*C                                                         000086900    002333  SEARCH
0021   :          X1 = X                                                               000087000    002515  SEARCH
0022   :          X2 = X                                                               000087100    002623  SEARCH
0023   :          F1 = F(X1)                                                           000087200    002731  SEARCH
0024   :          DO 50 N = 2,I-1                                                      000087300    002902  SEARCH
0025   :            IF(A - X2) 30, 40, 40                                              000087320    003180  SEARCH
0026   :     30       X = B - L2*C                                                     000087340    003371  SEARCH
0027   :              X1 = X2                                                          000087360    003553  SEARCH
0028   :              X2 = X                                                           000087380    003650  SEARCH
0029   :              F2 = F(X2)                                                       000087400    003758  SEARCH
0030   :            GO TO 45                                                           000087420    003923  SEARCH
0031   :     40       X = A + L2*C                                                     000087440    004015  SEARCH
0032   :              X2 = X1                                                          000087460    004197  SEARCH
0033   :              X1 = X                                                           000087480    004294  SEARCH
0034   :              F1 = F(X1)                                                       000087500    004402  SEARCH
0035   :     45       IF(K) WRITE(3,101) X1, X2, F1, F2                                000087520    004573  SEARCH
0037   :              IF(F1 - F2) 46, 47, 47                                           000087540    005384  SEARCH
0038   :     46         B = X1                                                         000087560    005565  SEARCH
0039   :                F1 = F2                                                        000087580    005673  SEARCH
0040   :              GO TO 48                                                         000087600    005770  SEARCH
0041   :     47         A = X2                                                         000087620    005856  SEARCH
0042   :                F2 = F1                                                        000087640    005954  SEARCH
0043   :     48       Z = L2                                                           000087660    006061  SEARCH
0044   :              L2 = L1 - L2                                                     000087680    006158  SEARCH
0045   :              L1 = Z                                                           000087700    006306  SEARCH
0046   :     50   CONTINUE                                                            000087720    006403  SEARCH
0047   :          X = (A + B)/2.                                                       000087740    006489  SEARCH
0048   :          Y = F(X)                                                             000087760    006565  SEARCH
0049   :          WRITE(3,102) X, Y, A, B, I                                           000087780    006656  SEARCH
0050   :          RETURN                                                               000087800    007690  SEARCH
0051   :  100   FORMAT(//, 10X, 'X1', 10X, 'X2', 10X, 'F1', 10X, 'F2'/)               000087820    007804  SEARCH
0052   :  101   FORMAT(4E12.5)                                                       000087840    007804  SEARCH
0053   :  102   FORMAT(/5X, 'X =', E12.4, 5X, 'Y =', E12.4/5X,                       000087860    007804  SEARCH
0053   :     *     'MAXIMUM LIES BETWEEN', E12.4, ' AND', E12.4,                     000087880    007804  SEARCH
0053   :     *.    5X, '(', I2, ' FUNCTION EVALUATIONS)'/ )                          000087890    007804  SEARCH
0054   :          END                                                                  000087900    007804  SEARCH
```

```
RROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        MONDAY     10/08/79    01:43 PM
    /(E99000)   /FAULTO
 0001  :          SUBROUTINE UPDATE(PAR)                                          : 00088800    000075  DFAULTO
 0002  :          COMMON/DELP/  ITYPE,    Y2IP, COSTMN,    JPAR,    NPAR,  ICORR, : 00088900    000075  UPDATE
 0002  :     *                        W,    WROW,    WCOL,       X,    XROW,  XCOL  : 00089000    000075  UPDATE
 0003  :          COMPLEX    Y2IP(17,17), W(80), X(80)                            : 00089100    000075  UPDATE
 0004  :          INTEGER    WROW(80), WCOL(80), XROW(80), XCOL(80)               : 00089200    000075  UPDATE
 0005  :          INTEGER ITYPE(80), ICORR(80)                                    : 00089300    000075  UPDATE
 0006  :          COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),        : 00089400    000075  UPDATE
 0006  :      1                 YB(65), YROW(65), GMROW(15), GMCOL(15),          : 00089500    000075  UPDATE
 0006  :      2                 RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),    : 00089600    000075  UPDATE
 0006  :      3                 AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,   : 00089700    000075  UPDATE
 0006  :      4                 IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR          : 00089800    000075  UPDATE
 0007  :          INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,       : 00089900    000075  UPDATE
 0007  :      1         AROW, ACOL, YBROW, YBCOL                                  : 00090000    000075  UPDATE
 0008  :          COMPLEX Y, YB                                                   : 00090100    000075  UPDATE
 0009  :          COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),    : 00090200    000075  UPDATE
 0009  :      1                 Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),  : 00090300    000075  UPDATE
 0009  :      2                 A2MNUS(50), A2PLSP(50), A2MNSP(50), I1, J1, IP,  : 00090400    000075  UPDATE
 0009  :      3                 N, IB, ITOT, YN(3,3)                             : 00090500    000075  UPDATE
 0010  :          INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                          : 00090600    000075  UPDATE
 0011  :          INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                          : 00090700    000075  UPDATE
 0012  :          COMPLEX YN, Y2, Y2I, Y2PI, CMPLX                                : 00090800    000075  UPDATE
       :C                                                                        : 00090900            UPDATE
 0013  :          GO TO (1,2,3,4,5), JPAR                                         : 00091000    000075  UPDATE
 0014  :      1   XT = REAL( Y(NPAR) )                                           : 00091100    000422  UPDATE
 0015  :          YT = AIMAG( Y(NPAR) )                                           : 00091200    000725  UPDATE
 0016  :          IF( ABS(XT) .GT. ABS(YT) ) GO TO 30                             : 00091300    001037  UPDATE
 0018  :            Y(NPAR) = CMPLX(0., PAR)                                      : 00091400    001200  UPDATE
 0019  :          GO TO 10                                                        : 00091500    001559  UPDATE
 0020  :     30     Y(NPAR) = CMPLX(PAR, 0.)                                      : 00091600    001645  UPDATE
 0021  :          GO TO 10                                                        : 00091700    002004  UPDATE
 0022  :      2   RM(NPAR) = PAR                                                 : 00091800    002090  UPDATE
 0023  :          GO TO 10                                                        : 00091900    002265  UPDATE
 0024  :      3   ALPHA(NPAR) = PAR                                              : 00092000    002351  UPDATE
 0025  :          GO TO 10                                                        : 00092100    002526  UPDATE
 0026  :      4   XMU(NPAR) = PAR                                                : 00092200    002612  UPDATE
 0027  :          GO TO 10                                                        : 00092300    002787  UPDATE
 0028  :      5   GM(NPAR) = PAR                                                 : 00092400    002873  UPDATE
 0029  :     10   CALL YBMAT                                                      : 00092850    003048  UPDATE
 0030  :          RETURN                                                          : 00092900    003153  UPDATE
 0031  :          END                                                             : 00093000    003267  UPDATE

 ERRORS AND NO WARNINGS IN 31 STATEMENTS,CODE EMITTED = 3326 BITS (416 BYTES)UPDATE
MPILE TIME IS 23.3 SECONDS FOR 40 CARDS AT 103 CARDS/MINUTE.
```

```
URROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        MONDAY    10/08/79    01:43 PM
    /(E99000)   /FAULTO
   0001   :          FUNCTION F(X)                                                    : 00087920    000075   OFAULTO
          :C                                                                          : 00087940             OFAULTO
   0002   :          COMMON/SRCH/ PAR, DELPAR, YNNOM                                  : 00087960    000075   F
   0003   :          COMPLEX YNNOM(3,3)                                               : 00087980    000075   F
          :C                                                                          : 00088000             F
   0004   :          COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),     : 00088002    000075   F
   0004   :      1                Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),  : 00088004    000075   F
   0004   :      2                A2MNUS(50), A2PLSP(50), A2MNSP(50), I, J, IP,    : 00088006    000075   F
   0004   :      3                N, I8, ITOT, YN(3,3)                             : 00088008    000075   F
   0005   :          INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                           : 00088010    000075   F
   0006   :          INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                           : 00088012    000075   F
   0007   :          COMPLEX YN, Y2, Y2I, Y2PI                                        : 00088014    000075   F
          :C                                                                          : 00088016             F
   0008   :          COMMON/DELP/  ITYPE, Y2IP, COSTMN, JPAR, NPAR, ICORR,            : 00088018    000075   F
   0008   :      *       IX, IW, W, WROW, WCOL, Q, XROW, XCOL                         : 00088020    000075   F
   0009   :          COMPLEX Y2IP(17,17), W(80), Q(80)                                : 00088022    000075   F
   0010   :          INTEGER WROW(80), WCOL(80), XROW(80), XCOL(80)                    : 00088024    000075   F
   0011   :          INTEGER ITYPE(80), ICORR(80)                                     : 00088026    000075   F
          :C                                                                          : 00088028             F
   0012   :          DELPAR = X - PAR                                                 : 00088030    000075   F
   0013   :          CALL UPDATE( X )                                                 : 00088035    000427   F
   0014   :          CALL UVVEC(DELPAR, A2PLUS, A2MNUS)                               : 00088040    000568   F
   0015   :          CALL Y2PI2                                                       : 00088060    000835   F
   0016   :          CALL YHAT(A1PLUS,A1MNUS,A2PLUS,A2MNUS,Y2IP,YN,IP,I8,17)          : 00088080    000940   F
   0017   :          F = 1./COSTF(YNNOM)                                              : 00088100    001643   F
   0018   :          RETURN                                                           : 00088120    001952   F
   0019   :          END                                                              : 00088700    002066   F

0 ERRORS AND NO WARNINGS IN 19 STATEMENTS,CODE EMITTED = 2125 BITS (266 BYTES)F
OMPILE TIME IS 21.8 SECONDS FOR 27 CARDS AT 74 CARDS/MINUTE.
```

```
URROUGHS B1800/B1700 FORTRAN COMPILER - MARK 8.0 RA 11/23/78 01:37 -        TUESDAY   10/02/79   05:18 PM
      /(E99000)  /FAULT30
0001  :          SUBROUTINE UVVEC( DELPAR, A2PLSP, A2MNSP )                                  000093100  000075  OFAULT30
      :C                                                                                    000093200          OFAULT30
0002  :          COMMON/VECTOR/  U, NU, IU, V, NV, IV                                        000093300  000075  UVVEC
0003  :          COMPLEX   U(20), V(20)                                                      000093400  000075  UVVEC
0004  :          INTEGER IU(20), IV(20)                                                      000093500  000075  UVVEC
      :C                                                                                    000093600          UVVEC
0005  :          COMMON/DELP/  ITYPE,    Y2IP, COSTMN,    JPAR,    NPAR,   ICORR,          000093700  000075  UVVEC
0005  :     *          IX,   IW,    W,    WROW,    WCOL,       X,    XROW,   XCOL           000093800  000075  UVVEC
0006  :          COMPLEX   Y2IP(17,17), W(80), X(80)                                         000093900  000075  UVVEC
0007  :          INTEGER   WROW(80), WCOL(80), XROW(80), XCOL(80)                             000094000  000075  UVVEC
0008  :          INTEGER ITYPE(80), ICORR(80)                                                000094100  000075  UVVEC
      :C                                                                                    000094200          UVVEC
0009  :          COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),                     000094300  000075  UVVEC
0009  :     1                  YB(65), YROW(65), GMROW(15), GMCOL(15),                      000094400  000075  UVVEC
0009  :     2                  RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),                000094500  000075  UVVEC
0009  :     3                  AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,               000094600  000075  UVVEC
0009  :     4                  IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR                      000094700  000075  UVVEC
0010  :          INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,                    000094800  000075  UVVEC
0010  :     1          AROW, ACOL, YBROW, YBCOL                                            000094900  000075  UVVEC
0011  :          COMPLEX Y, YB                                                               000095000  000075  UVVEC
      :C                                                                                    000095100          UVVEC
0012  :          COMPLEX ULC(20), VLC(20), CMPLX                                             000095200  000075  UVVEC
0013  :          INTEGER IULC(20), IVLC(20)                                                  000095300  000075  UVVEC
      :C                                                                                    000095400          UVVEC
0014  :           INTEGER A2PLSP(50), A2MNSP(50)                                           000095500  000075  UVVEC
      :C                                                                                    000096200          UVVEC
0015  :          DO 10 K=1,16                                                              000096300  000075  UVVEC
0016  :            ULC(K) = (0.,0.)                                                        000096400  000330  UVVEC
0017  :            VLC(K) = (0.,0.)                                                        000096500  000703  UVVEC
0018  :            U(K) = (0.,0.)                                                          000096600  001076  UVVEC
0019  :       10   V(K) = (0.,0.)                                                          000096700  001449  UVVEC
0020  :          GO TO (1,2,3,4,5), JPAR                                                   000096800  001954  UVVEC
      :C                                                                                    000096900          UVVEC
      :CC                        ***** Y IS VARIED *****                                    000097000          UVVEC
      :CC                                                                                   000097100          UVVEC
      :CC                  FORM VECTOR ULC:                                               000097200          UVVEC
      :C                                                                                    000097300          UVVEC
0021  :       1  NULC = 1                                                                  000097400  002145  UVVEC
0022  :          IULC(1) = NPAR                                                            000097500  002236  UVVEC
0023  :          ULC(1) = (1.,0.)                                                          000097600  002350  UVVEC
      :C                                                                                    000097700          UVVEC
      :CC                  FORM VECTOR VLC                                                000097800          UVVEC
      :CC                                                                                   000097900          UVVEC
0024  :          NVLC = 1                                                                  000098000  002547  UVVEC
0025  :          IVLC(1) = NPAR                                                            000098100  002638  UVVEC
0026  :          VLC(1) = CMPLX( DELPAR, 0.)                                               000098200  002752  UVVEC
0027  :          IF(IXMU .EQ. 0) GO TO 11                                                  000098300  002935  UVVEC
0029  :          DO 15 K=1,IXMU                                                            000098400  003068  UVVEC
0030  :            NVLC = NVLC + 1                                                         000098500  003159  UVVEC
0031  :            IVLC(NVLC) = XMUCOL(K)                                                  000098600  003295  UVVEC
0032  :            VLC(NVLC) = CMPLX( -DELPAR*XMU(K), 0.)                                  000098700  003533  UVVEC
0033  :       15   CONTINUE                                                                000098800  003998  UVVEC
0034  :       11   IF(IRM .EQ. 0) GO TO 21                                                 000098900  004201  UVVEC
0036  :          DO 20 M=1,IRM                                                             000099000  004334  UVVEC
0037  :            IF(RMROW(M) .NE. NPAR) GO TO 20                                         000099100  004425  UVVEC
0039  :            DO 14 K=1,IX                                                            000099200  004633  UVVEC
0040  :              IF(XROW(K) .NE. RMCOL(M)) GO TO 14                                    000099300  004724  UVVEC
0042  :                NVLC = NVLC + 1                                                     000099400  004995  UVVEC
0043  :                IVLC(NVLC) = GMCOL(K)                                               000099500  005131  UVVEC
0044  :                VLC(NVLC) = CMPLX( RM(M)*XROW(K)*DELPAR, 0.)                        000099600  005369  UVVEC
```

THIS DOCUMENT IS BEST QUALITY FRACTICABLE. COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, VOL. CAS-26, NO. 7, JULY 1979

```
0045    :    14     CONTINUE                                                    : 000997700    005922   UVVECCCC
```

```
0095    :            IVLC(NVLC) = K                                    : 00105800    015022  UVVEC
0096    :            VLC(NVLC) = -Y(I)*DELPAR*X(M)                     : 00105900    015197  UVVEC
0097    :   50   CONTINUE                                              : 00106000    016200  UVVEC
        :C                                                            : 00106100            UVVEC
        :CC     *** FORM VECTORS U AND V: ***                         : 00106200            UVVEC
        :C                                                            : 00106300            UVVEC
0098    :        GO TO 22                                             : 00106400    016403  UVVEC
        :C                                                            : 00106500            UVVEC
        :CC      ALPHA IS VARIED:                                     : 00106600            UVVEC
        :C                                                            : 00106700            UVVEC
0099    :    3   I = AROW(NPAR)                                       : 00106800    016489  UVVEC
0100    :        J = ACOL(NPAR)                                       : 00106900    016666  UVVEC
        :C                                                            : 00107000            UVVEC
        :CC                FORM VECTOR ULC:                           : 00107100            UVVEC
        :C                                                            : 00107200            UVVEC
0101    :        NULC = 1                                             : 00107300    016843  UVVEC
0102    :        IULC(1) = I                                          : 00107400    016934  UVVEC
0103    :        ULC(1) = (1.,0.)                                     : 00107500    017048  UVVEC
        :C                                                            : 00107600            UVVEC
        :CC                FORM VECTOR VLC:                           : 00107700            UVVEC
        :C                                                            : 00107800            UVVEC
0104    :        NVLC = 0                                             : 00107900    017245  UVVEC
0105    :        DO 60 M=1,IX                                         : 00108000    017336  UVVEC
0106    :          IF(XROW(M) .NE. J) GO TO 60                        : 00108100    017427  UVVEC
0108    :            K = XCOL(M)                                      : 00108200    017635  UVVEC
0109    :            NVLC = NVLC + 1                                  : 00108300    017812  UVVEC
0110    :            IVLC(NVLC) = K                                   : 00108400    017948  UVVEC
0111    :            VLC(NVLC) = -DELPAR*X(M)                         : 00108500    018123  UVVEC
0112    :   60   CONTINUE                                              : 00108600    018314  UVVEC
        :C                                                            : 00108700            UVVEC
        :CC      FORM VECTORS U AND V:                                : 00108800            UVVEC
        :C                                                            : 00108900            UVVEC
0113    :        GO TO 22                                             : 00109000    019017  UVVEC
        :C                                                            : 00109100            UVVEC
        :CC                ***** XMU IS VARIED *****                  : 00109200            UVVEC
        :C                                                            : 00109300            UVVEC
0114    :    4   I = XMUROW(NPAR)                                     : 00109400    019103  UVVEC
0115    :        J = XMUCOL(NPAR)                                     : 00109500    019280  UVVEC
        :C                                                            : 00109600            UVVEC
        :CC                FORM VECTOR VLC:                           : 00109700            UVVEC
        :C                                                            : 00109800            UVVEC
0116    :        NVLC = 1                                             : 00109900    019457  UVVEC
0117    :        IVLC(1) = J                                          : 00110000    019548  UVVEC
0118    :        VLC(1) = (1.,0.)                                     : 00110100    019662  UVVEC
        :C                                                            : 00110200            UVVEC
        :CC                FORM VECTOR ULC:                           : 00110300            UVVEC
        :C                                                            : 00110400            UVVEC
0119    :        NULC = 0                                             : 00110500    019859  UVVEC
0120    :        DO 70 M=1,IW                                         : 00110600    019950  UVVEC
0121    :          IF(WCOL(M) .NE. I) GO TO 70                        : 00110700    020041  UVVEC
0123    :            K = WROW(M)                                      : 00110800    020249  UVVEC
0124    :            NULC = NULC + 1                                  : 00110900    020426  UVVEC
0125    :            IULC(NULC) = K                                   : 00111000    020562  UVVEC
0126    :            ULC(NULC) = -W(M)*Y(I)*DELPAR                    : 00111100    020737  UVVEC
0127    :   70   CONTINUE                                              : 00111200    021786  UVVEC
        :C                                                            : 00111300            UVVEC
        :CC      FORM VECTORS U AND V:                                : 00111400            UVVEC
        :C                                                            : 00111500            UVVEC
0128    :        GO TO 22                                             : 00111600    021989  UVVEC
        :C                                                            : 00111700            UVVEC
        :CC                ***** GM IS VARIED *****                   : 00111800            UVVEC
        :C                                                            : 00111900            UVVEC
```

```
0129    :     5   I = GMROW(NPAR)                                    : 00112000    022075  UVVEC
0130    :         J = GMCOL(NPAR)                                    : 00112100    022252  UVVEC
        :C                                                           : 00112200            UVVEC
        :C              FORM VECTOR VLC:                             : 00112300            UVVEC
        :C                                                           : 00112400            UVVEC
0131    :         NVLC = 1                                           : 00112500    022429  UVVEC
0132    :         IVLC(1) = J                                        : 00112600    022520  UVVEC
0133    :         VLC(1) = (1.,0.)                                   : 00112700    022634  UVVEC
        :C                                                           : 00112800            UVVEC
        :C              FORM VECTOR ULC:                             : 00112900            UVVEC
        :C                                                           : 00113000            UVVEC
0134    :         NULC = 0                                           : 00113100    022831  UVVEC
0135    :         DO 80 M=1,IW                                       : 00113200    022922  UVVEC
0136    :           IF(WCOL(M) .NE. I) GO TO 80                      : 00113300    023013  UVVEC
0138    :             K = WROW(M)                                    : 00113400    023221  UVVEC
0139    :             NULC = NULC + 1                                : 00113500    023398  UVVEC
0140    :             IULC(NULC) = K                                 : 00113600    023534  UVVEC
0141    :             ULC(NULC) = W(M)*DELPAR                        : 00113700    023709  UVVEC
0142    :    80   CONTINUE                                           : 00113800    024327  UVVEC
        :C                                                           : 00113900            UVVEC
        :C      FORM VECTORS U AND V:                                : 00114000            UVVEC
        :C                                                           : 00114100            UVVEC
0143    :         GO TO 22                                           : 00114200    024530  UVVEC
0144    :         END                                                : 00114300    024615  UVVEC

1 ERRORS AND NO WARNINGS IN 144 STATEMENTS.CODE EMITTED = 24730 BITS (3092 BYTES)UVVEC
OMPILE TIME IS 56.3 SECONDS FOR 207 CARDS AT 221 CARDS/MINUTE.
```

```
JRROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
      /(F99000)  /FAULT30
0001  :          SUBROUTINE Y2PI2                                                    : 00114400   000075  OFAULT30
0002  :          COMMON/DELP/  ITYPE,   Y2IP, COSTHN,   JPAR,   NPAR,  ICORR,       : 00114500   000075  Y2PI2
0002  :      *                      W,   WROW,   WCOL,      X,   XROW,   XCOL       : 00114600   000075  Y2PI2
0003  :          COMPLEX   Y2IP(17,17), W(80), X(80)                                 : 00114700   000075  Y2PI2
0004  :          INTEGER   WROW(80), WCOL(80), XROW(80), XCOL(80)                     : 00114800   000075  Y2PI2
0005  :          INTEGER ITYPE(80), ICORR(80)                                        : 00114900   000075  Y2PI2
      :C                                                                            : 00115000           Y2PI2
0006  :          COMMON/VECTOR/  U, NU, IU, V, NV, IV                                : 00115100   000075  Y2PI2
0007  :          COMPLEX  U(20), V(20)                                               : 00115200   000075  Y2PI2
0008  :          INTEGER IU(20), IV(20)                                              : 00115300   000075  Y2PI2
      :C                                                                            : 00115400           Y2PI2
0009  :          COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),       : 00115500   000075  Y2PI2
0009  :     1                 Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),     : 00115600   000075  Y2PI2
0009  :     2                 A2MNUS(50), A2PLSP(50), A2MNSP(50), I1, J1, IP,     : 00115700   000075  Y2PI2
0009  :     3                 N, IB, ITOT, YN(3,3)                                : 00115800   000075  Y2PI2
0010  :          INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                              : 00115900   000075  Y2PI2
0011  :          INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                              : 00116000   000075  Y2PI2
0012  :          COMPLEX YN, Y2, Y2I, Y2PI                                           : 00116100   000075  Y2PI2
0013  :          COMPLEX E(20), D, ESUM                                              : 00116200   000075  Y2PI2
0014  :          INTEGER EROW(20), ECOL(20)                                          : 00116300   000075  Y2PI2
      :C                                                                            : 00116400           Y2PI2
0015  :          IQ = N-IP                                                           : 00116500   000075  Y2PI2
0016  :          DO 5 I = 1,IQ                                                       : 00116600   000375  Y2PI2
0017  :          DO 5 J = 1,IQ                                                       : 00116700   000466  Y2PI2
0018  :     5    Y2IP(I,J) = Y2I(I,J)                                                : 00116800   000557  Y2PI2
0019  :          D = (1.,0.)                                                         : 00116900   001431  Y2PI2
0020  :          DO 10 M = 1,NV                                                      : 00117000   001628  Y2PI2
0021  :          DO 10 K = 1,NU                                                      : 00117100   001719  Y2PI2
0022  :    10    D = D * V(M)*Y2I(IV(M),IU(K))*U(K)                                  : 00117200   001810  Y2PI2
0023  :          NE = 0                                                              : 00117300   003394  Y2PI2
0024  :          DO 30 I = 1,NU                                                      : 00117400   003485  Y2PI2
0025  :            K = IU(I)                                                         : 00117500   003576  Y2PI2
0026  :          DO 30 L = 1,IQ                                                      : 00117600   003753  Y2PI2
0027  :            ESUM  =(0.,0.)                                                    : 00117700   003844  Y2PI2
0028  :            DO 20 J = 1,NV                                                    : 00117800   004041  Y2PI2
0029  :              M = IV(J)                                                       : 00117900   004132  Y2PI2
0030  :    20        ESUM = ESUM + V(M)*Y2I(M,L)                                     : 00118000   004309  Y2PI2
0031  :            NE = NE + 1                                                       : 00118100   005274  Y2PI2
0032  :            E(NE) = ESUM*U(I)/D                                               : 00118200   005410  Y2PI2
0033  :            EROW(NE) = K                                                      : 00118300   006188  Y2PI2
0034  :    30      ECOL(NE) = L                                                      : 00118400   006363  Y2PI2
0035  :          DO 40 K = 1,NE                                                      : 00118500   006826  Y2PI2
0036  :              J = ECOL(K)                                                     : 00118600   006917  Y2PI2
0037  :              M = EROW(K)                                                     : 00118700   007094  Y2PI2
0038  :            DO 40 I = 1,IQ                                                    : 00118800   007271  Y2PI2
0039  :              Y2IP(I,J) = Y2IP(I,J) - Y2I(I,M)*E(K)                           : 00118900   007362  Y2PI2
0040  :    40    CONTINUE                                                            : 00119000   008664  Y2PI2
0041  :          RETURN                                                              : 00119100   009011  Y2PI2
0042  :          END                                                                 : 00119200   009125  Y2PI2

0 ERRORS AND NO WARNINGS IN 42 STATEMENTS,CODE EMITTED = 9184 BITS (1148 BYTES)Y2PI2
OMPILE TIME IS 37.7 SECONDS FOR 49 CARDS AT 78 CARDS/MINUTE.
```


IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, VOL. CAS-26, NO. 7, JULY 1979

```
RROUGHS B1800/B1700 FORTRAN COMPILER , MARK 8.0 RA 11/23/78 01:37 ,        TUESDAY   10/02/79   05:18 PM
     /(E99999)   /FAULT30
0001 :          SUBROUTINE JACCOL( DELY, IPAR)                                          : 00119300   000075  OFAULT30
     :C                                                                                 : 00119400             OFAULT30
0002 :          COMMON/DELP/  ITYPE,   Y2IP, COSTMN,   JPAR,   MPAR,  ICORR,          : 00119500   000075  JACCOL
0002 :     *          IX, IW,   W,  WROW,  WCOL,     X,  XROW,  XCOL                  : 00119600   000075  JACCOL
0003 :          COMPLEX   Y2IP(17,17), W(80), X(80)                                     : 00119700   000075  JACCOL
0004 :          INTEGER  WROW(80), WCOL(80), XROW(80), XCOL(80)                         : 00119800   000075  JACCOL
0005 :          INTEGER ITYPE(80), ICORR(80)                                           : 00119900   000075  JACCOL
     :C                                                                                 : 00120000             JACCOL
0006 :          COMMON/OTHER/A1PLUS(50), A1MNUS(50), A1PLSP(50), A1MNSP(50),           : 00120100   000075  JACCOL
0006 :     1                 Y2(17,17), Y2I(17,17), Y2PI(16,16), A2PLUS(50),         : 00120200   000075  JACCOL
0006 :     2                 A2MNUS(50), A2PLSP(50), A2MNSP(50), I1, J1, IP,         : 00120300   000075  JACCOL
0006 :     3               NN, IB, ITOT, YN(3,3)                                     : 00120400   000075  JACCOL
0007 :          INTEGER A2PLUS, A2MNUS, A2PLSP, A2MNSP                                 : 00120500   000075  JACCOL
0008 :          INTEGER A1PLUS, A1MNUS, A1PLSP, A1MNSP                                 : 00120600   000075  JACCOL
0009 :          COMPLEX YN, Y2, Y2I, Y2PI                                              : 00120700   000075  JACCOL
     :C                                                                                 : 00120800             JACCOL
0010 :          COMMON/ELEMNT/Y(65), GM(15), RM(15), XMU(15), ALPHA(15),               : 00120900   000075  JACCOL
0010 :     1                  YB(65), YROW(65), GMROW(15), GMCOL(15),                 : 00121000   000075  JACCOL
0010 :     2                  RMROW(15), RMCOL(15), XMUROW(15), XMUCOL(15),           : 00121100   000075  JACCOL
0010 :     3                  AROW(15), ACOL(15), YBROW(65), YBCOL(65), IGM,          : 00121200   000075  JACCOL
0010 :     4                  IRM, IXMU, IA, IALPHA, IYB, OMEGA, IPAR1                : 00121300   000075  JACCOL
0011 :          INTEGER YROW, GMROW, GMCOL, RMROW, RMCOL, XMUROW, XMUCOL,              : 00121400   000075  JACCOL
0011 :     1          AROW, ACOL, YBROW, YBCOL                                        : 00121500   000075  JACCOL
0012 :          COMPLEX Y, YB                                                          : 00121600   000075  JACCOL
     :C                                                                                 : 00121700             JACCOL
0013 :          COMMON/DEL/ C,D                                                        : 00121800   000075  JACCOL
0014 :          COMPLEX C(50,50), D(50,50), DELY3(80), G(50,50), F(50,50),             : 00121900   000075  JACCOL
0014 :     *          DELY(3,3), CMPLX                                                : 00122000   000075  JACCOL
0015 :          INTEGER R, DELROW(80), DELCOL(80)                                      : 00122100   000075  JACCOL
     :C                                                                                 : 00122200             JACCOL
     :C        INITIALIZE DELY TO ZERO:                                                : 00122300             JACCOL
     :C                                                                                 : 00122400             JACCOL
0015 :          DO 10 I=1,IP                                                           : 00122500   000075  JACCOL
0017 :          DO 10 J=1,IP                                                           : 00122600   000330  JACCOL
0018 :     10   DELY(I,J) = (0.,0.)                                                    : 00122700   000421  JACCOL
     :C                                                                                 : 00122800             JACCOL
0019 :          IDELYB = 0                                                             : 00122900   001084  JACCOL
0020 :          I = ICORR(IPAR)                                                        : 00122950   001175  JACCOL
0021 :          IF(ITYPE(IPAR) .GE. 4) GO TO 200                                       : 00123000   001352  JACCOL
     :C                                                                                 : 00123100             JACCOL
     :C        COMPUTE DELYB:                                                          : 00123200             JACCOL
     :C                                                                                 : 00123300             JACCOL
     :C        PARAMETER IS PASSIVE (R, L, C):                                         : 00123400             JACCOL
     :C                                                                                 : 00123450             JACCOL
     :C                                                                                 : 00123500             JACCOL
0023 :          DO 100 M = 1,IW                                                        : 00123600   001548  JACCOL
0024 :            R = WROW(M)                                                          : 00123700   001639  JACCOL
0025 :            J = WCOL(M)                                                          : 00123800   001616  JACCOL
0026 :            I = ICORR(IPAR)                                                      : 00123900   001993  JACCOL
0027 :            IF(J .NE. I) GO TO 100                                               : 00124000   002170  JACCOL
0029 :            IDELYB = IDELYB + 1                                                  : 00124100   002315  JACCOL
0030 :            DELROW(IDELYB) = R                                                   : 00124200   002451  JACCOL
0031 :            DELCOL(IDELYB) = I                                                   : 00124300   002626  JACCOL
0032 :            DELYB(IDELYB) = W(M)                                                 : 00124400   002801  JACCOL
0033 :            DO 50 N = 1,IXMU                                                     : 00124500   003283  JACCOL
0034 :              J = XMUROW(N)                                                      : 00124600   003374  JACCOL
0035 :              IF(J .NE. I) GO TO 50                                              : 00124700   003551  JACCOL
0037 :              IDELYB = IDELYB + 1                                                : 00124800   003596  JACCOL
0038 :              DELROW(IDELYB) = R                                                 : 00124900   003332  JACCOL
```

```
0039  :              DELCOL(IDELYB) = XMUCOL(N)                                  : 00125000    004007  JACCOL
0040  :              DELYB(IDELYB) = -W(M)*XMU(N)                                : 00125100    004245  JACCOL
0041  :    50      CONTINUE                                                      : 00125200    005017  JACCOL
0042  :   100    CONTINUE                                                        : 00125300    005220  JACCOL
0043  :           GO TO (300, 120, 130), ITYPE(IPAR)                            : 00125400    005423  JACCOL
0044  :   120    VALUE = -1./( OMEGA*AIMAG( Y(I) ) )                           : 00125450    005641  JACCOL
0045  :           DO 125 K = 1,IDELYB                                           : 00125500    006035  JACCOL
0046  :   125    DELYB(K) = CMPLX(0., 1./(OMEGA*VALUE*VALUE) )*DELYB(K)        : 00125600    006126  JACCOL
0047  :           GO TO 300                                                     : 00125700    007140  JACCOL
0048  :   130    DO 135 K = 1,IDELYB                                           : 00125800    007226  JACCOL
0049  :   135    DELYB(K) = CMPLX(0., OMEGA)*DELYB(K)                          : 00125900    007317  JACCOL
0050  :           GO TO 300                                                     : 00126000    008215  JACCOL
      :C                                                                       : 00126100            JACCOL
      :C        PARAMETER IS AN AMPLIFIER GAIN (RM, ALPHA, MU, GM)             : 00126200            JACCOL
      :C                                                                       : 00126300            JACCOL
0051  :   200    JPAR = ITYPE(IPAR) - 10                                       : 00126400    008301  JACCOL
0052  :           IF(JPAR .EQ. 1) JPAR = 5                                      : 00126500    008500  JACCOL
0054  :           IF(JPAR .GE. 4) GO TO 250                                     : 00126600    008728  JACCOL
0056  :           IF(JPAR .EQ. 3) GO TO 201                                     : 00126700    008851  JACCOL
0058  :             M = RMROW(I)                                                : 00126800    008994  JACCOL
0059  :             N = RMCOL(I)                                                : 00126900    009171  JACCOL
0060  :           GO TO 202                                                     : 00127000    009348  JACCOL
0061  :   201       M = AROW(I)                                                 : 00127100    009434  JACCOL
0062  :             N = ACOL(I)                                                 : 00127200    009611  JACCOL
0063  :   202     DO 210 K = 1,IX                                               : 00127300    009788  JACCOL
0064  :             IF(XROW(K) .NE. N) GO TO 210                                : 00127400    009879  JACCOL
0065  :               IDELYB = IDELYB + 1                                       : 00127500    010087  JACCOL
0067  :               DELROW(IDELYB) = M                                        : 00127600    010223  JACCOL
0068  :               DELCOL(IDELYB) = XCOL(K)                                  : 00127700    010398  JACCOL
0069  :               DELYB(IDELYB) = X(K)                                      : 00127800    010636  JACCOL
0070  :             IF(JPAR .EQ. 3) DELYB(IDELYB) = -Y(M)*DELYB(IDELYB)         : 00127900    011118  JACCOL
0072  :   210     CONTINUE                                                      : 00128000    012164  JACCOL
0073  :           GO TO 300                                                     : 00128100    012357  JACCOL
0074  :   250     IF(JPAR .EQ. 5) GO TO 251                                     : 00128200    012453  JACCOL
0076  :             M = XMUROW(I)                                               : 00128300    012586  JACCOL
0077  :             N = XMUCOL(I)                                               : 00128400    012763  JACCOL
0078  :           GO TO 252                                                     : 00128500    012940  JACCOL
0079  :   251     M = GMROW(I)                                                  : 00128600    013026  JACCOL
0080  :             N = GMCOL(I)                                                : 00128700    013203  JACCOL
0081  :   252     DO 260 K = 1,IW                                               : 00128800    013380  JACCOL
0082  :             IF(XCOL(K) .NE. M) GO TO 260                                : 00128900    013471  JACCOL
0084  :               IDELYB = IDELYB + 1                                       : 00129000    013679  JACCOL
0085  :               DELROW(IDELYB) = WROW(K)                                  : 00129100    013815  JACCOL
0086  :               DELCOL(IDELYB) = M                                        : 00129200    014053  JACCOL
0087  :               DELYB(IDELYB) = W(K)                                      : 00129300    014228  JACCOL
0088  :             IF(JPAR .EQ. 4) DELYB(IDELYB) = -Y(M)*DELYB(IDELYB)         : 00129400    014710  JACCOL
0090  :   260     CONTINUE                                                      : 00129500    015756  JACCOL
      :C                                                                       : 00129600            JACCOL
      :C        COMPUTE DELY/DELPAR:                                           : 00129700            JACCOL
      :C                                                                       : 00129800            JACCOL
0091  :   300     DO 310 I = 1,IB                                               : 00130000    015959  JACCOL
0092  :           DO 310 J = 1,IB                                               : 00130100    016050  JACCOL
0093  :           F(I,J) = (0.,0.)                                              : 00130200    016141  JACCOL
0094  :   310     G(I,J) = (0.,0.)                                              : 00130300    016566  JACCOL
      :C                                                                       : 00130400            JACCOL
0095  :           DO 320 J = 1,IB                                               : 00130500    017279  JACCOL
0096  :           DO 320 L = 1,IYB                                              : 00130600    017370  JACCOL
0097  :             I = YBROW(L)                                                : 00130700    017461  JACCOL
0098  :             K = YBCOL(L)                                                : 00130800    017638  JACCOL
0099  :   320     F(I,J) = F(I,J) - YB(L)*C(K,J)                                : 00130900    017815  JACCOL
0100  :           DO 330 K = 1,IB                                               : 00131000    019405  JACCOL
0101  :   330     F(K,K) = F(K,K) + (1.,0.)                                     : 00131100    019496  JACCOL
```

```
0102  :          DO 340 I = 1,IB                                          : 00131200    020377   JACCOL
0103  :          DO 340 J = 1,IB                                          : 00131300    020468   JACCOL
0104  :          DO 340 M = 1,IDELYB                                      : 00131400    020559   JACCOL
0105  :            K = DELROW(M)                                          : 00131500    020650   JACCOL
0106  :            L = DELCOL(M)                                          : 00131600    020827   JACCOL
0107  :  340     G(I,J) = G(I,J) + F(I,K)*DELYB(M)*D(L,J)                 : 00131700    021004   JACCOL
      :C                                                                  : 00131800             JACCOL
      :C           COMPUTE A1 * G * A1T.  A1 IS STORED SPARSELY.          : 00131900             JACCOL
      :C                                                                  : 00132000             JACCOL
0108  :          DO 350 L = 1,IB                                          : 00132100    023139   JACCOL
0109  :            IPL= A1PLUS(L)                                         : 00132200    023230   JACCOL
0110  :            IM = A1MNUS(L)                                         : 00132300    023407   JACCOL
0111  :          DO 350 K = 1,IB                                          : 00132400    023584   JACCOL
0112  :            JP = A1PLUS(K)                                         : 00132500    023675   JACCOL
0113  :            JM = A1MNUS(K)                                         : 00132600    023852   JACCOL
0114  :            IF(IPL*JP .NE. 0) DELY(IPL,JP) = DELY(IPL,JP) + G(L,K) : 00132700    024029   JACCOL
0116  :            IF(IM*JM .NE. 0) DELY(IM,JM) = DELY(IM,JM) + G(L,K)    : 00132800    025058   JACCOL
0118  :            IF(IPL*JM .NE. 0) DELY(IPL,JM) = DELY(IPL,JM) - G(L,K) : 00132900    026087   JACCOL
0120  :            IF(IM*JP .NE. 0) DELY(IM,JP) = DELY(IM,JP) - G(L,K)    : 00133000    027116   JACCOL
0122  :  350     CONTINUE                                                 : 00133100    028145   JACCOL
0123  :          RETURN                                                   : 00133230    028492   JACCOL
0124  :          END                                                      : 00133300    028606   JACCOL

NO ERRORS AND NO WARNINGS IN 124 STATEMENTS,CODE EMITTED = 28665 BITS (3584 BYTES)JACCOL
COMPILE TIME IS 59.1 SECONDS FOR 143 CARDS AT 145 CARDS/MINUTE.
```

Appendix 2

Reprint of

"Efficient Fault Analysis
in
Linear Analog Circuits"

by

A. T. Johnson, Jr.

Efficient Fault Analysis in Linear Analog Circuits

ALFRED T. JOHNSON, JR., MEMBER, IEEE

Abstract—Fault analysis in analog networks is a form of network parameter identification. The problem of finding network parameters from measurements at the accessible terminals can be expressed as the solution of a system of nonlinear equations. Such a system of equations is usually solved by a multidimensional search. Every step of the search requires solving for the network responses in terms of the parameters, comparing the solution to the measured responses, and then adjusting the parameters in such a way as to produce a response closer to the measured response. The computation time required for this analysis is often excessive in practical applications, and poses a major impediment to fault analysis in analog circuits. The representation of short and open circuits poses additional problems, depending upon how the network equations are formulated.

We have used nodal analysis, representing short circuits by a change in the network graph, and open circuits by a zero branch admittance. A formula of Householder and sparse matrix techniques are used to efficiently compute the response of an electrical circuit with either a short circuit or other large parameter change. The number of complex multiplications and divisions to find the network response at the accessible terminals when the network has a single short circuit involving at least one inaccessible terminal is reduced by about a factor of 15 compared with direct methods.

I. Introduction

IT is surprising perhaps that although the problem of fault isolation in analog circuits has been studied longer than the comparable problem in digital circuits,[1] the digital problem is much better understood, and automatic test equipment (ATE) for isolating faults in digital equipment to the chip and even to the gate level is available, but ATE available for isolating faults in analog circuits is much less sophisticated [13], [14]. The explanation of this phenominon lies partly with the fact that analog signals are inherently more complex than digital signals. They occur continuously in time, rather than at discrete times, and their values (in principle) have infinite resolution, instead of being truncated to a finite number of bits.

In the linear analog circuits discussed in this report, the problem of signal variety can be partly overcome by restricting observation to the sinusoidal steady state, since the system behavior to any input can be determined from a knowledge of the sinusoidal steady state response at all frequencies. Clearly *all* frequencies cannot be considered, and the problem of how to choose a set of frequencies has only begun to receive attention [9], [10]. In addition, the response at any frequency is related to the parameter values and the network graph by nonlinear equations, to which the only method of solution possible is a time consuming search.

In this paper we show how the search for single short circuits can be accomplished efficiently enough to make the computation practical. We also demonstrate the effectiveness of this method in identifying short circuit faults in analog circuits whose resistance, capacitance, inductance, and controlled source gains differ significantly from their nominal values.

II. Formulating the Y-Parameters at the Accessible Nodes

We will consider linear lumped-parameter networks having $p+1$ accessible terminals, numbered 0–p, as shown in Fig. 1. The network of Fig. 1 contains $n+1$ nodes, including the reference node, and b branches and is assumed to contain no internal independent sources but may contain dependent sources. We will formulate the nodal equations using the standard branch shown in Fig. 2.

The passive element in branch k, denoted by its admittance Y_{kk}, must be nonzero, but any of the sources may be zero or all may be present, if desired. In the equations that follow, $\boldsymbol{v}$ is the branch voltage vector, $\boldsymbol{j}$ is the branch current vector, $\boldsymbol{e}$ is the node voltage vector, $\boldsymbol{v}_s$ is the independent voltage source vector, and $\boldsymbol{j}_s$ is the independent current source vector, using the notation of Desoer and Kuh [1]. The dimension of each of these vectors is $b\times 1$, except $\boldsymbol{e}$ which is $n\times 1$. Considering the kth standard branch shown in Fig. 2, it is clear that the branch $v-i$ constraints are given by

$$\boldsymbol{j}=\boldsymbol{j}_s+(\boldsymbol{G}_m+\boldsymbol{Y}-\boldsymbol{Y}\boldsymbol{\mu})\boldsymbol{v}+(\boldsymbol{\alpha}-\boldsymbol{Y}\boldsymbol{R}_m)\boldsymbol{j}-\boldsymbol{Y}\boldsymbol{v}_s \qquad (1)$$

where $\boldsymbol{Y}=\text{diag}\,(Y_k)$ is the branch passive admittance matrix, and $\boldsymbol{G}_m$, $\boldsymbol{\mu}$, $\boldsymbol{\alpha}$, and $\boldsymbol{R}_m$ are the coupling matrices due to the controlled sources. The dimension of each of these matrices is $b\times b$. (The elements of matrices $\boldsymbol{G}_m$ and $\boldsymbol{\alpha}$ are current sources, and the elements of matrices $\boldsymbol{\mu}$ and $\boldsymbol{R}_m$ are voltage sources controlled, respectively, by voltages and currents.) Following the convention of Desoer and Kuh, define the $n\times 1$ node-to-branch incidence

Manuscript received September 22, 1978; revised March 8, 1979. This work was supported by the Air Force Office of Scientific Research through the USAF-ASEE Summer Faculty Research Program (WPAFB) under Contract F44620-76-C-0052, The Ohio State University, Columbus, OH.

The author is with the Center of Engineering, Widener College, Chester, PA 19013.

[1] An extensive bibliography of both digital and analog fault isolation is given by Sacks and Liberty [12].

0098-4094/79/0700-0475$00.75 ©1979 IEEE

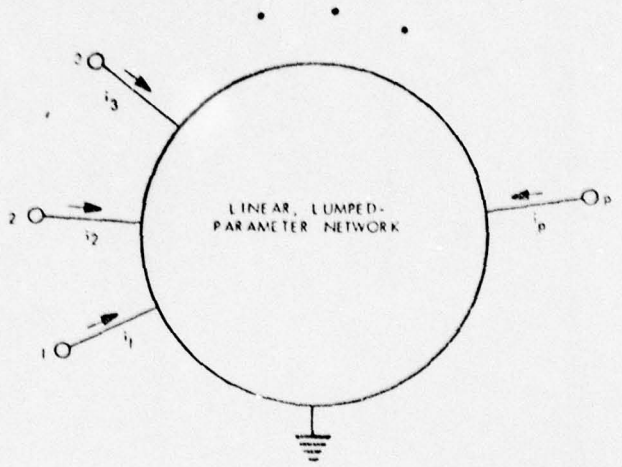


Fig. 1. Linear lumped-parameter network with $p+1$ accessible terminals.

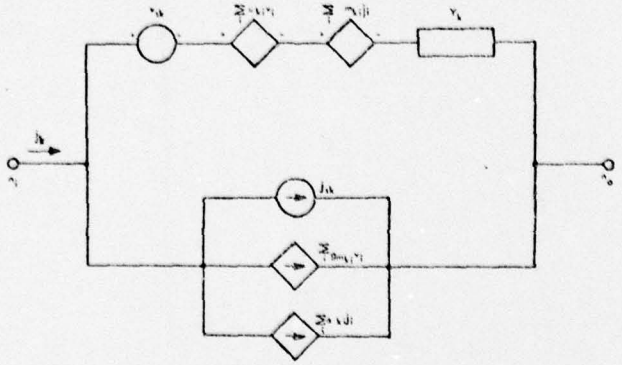


Fig. 2. Standard branch k.

matrix A,

$$\{A\}_{ij}=\begin{cases}+1, & \text{if branch } j \text{ leaves node } i\\ -1, & \text{if branch } j \text{ enters node } i\\ 0, & \text{if branch } j \text{ is not incident on node } i\end{cases}$$

Solving (1) for j and using the Kirchhoff current law $Aj=0$ and the Kirchhoff voltage law $A^T e=v$, it can be shown that the network node equations are

$$AY_bA^Te=i \tag{2}$$

where Y_b is the $b\times b$ branch admittance matrix given by

$$Y_b=(1+YR_m-\alpha)^{-1}(Y-Y\mu+G_m) \tag{3}$$

and i is the $n\times 1$ vector of independent source currents

$$i=A(1+YR_m-\alpha)^{-1}(Ye_s-j_s). \tag{4}$$

Note that Ye_s is a Norton transformation of the independent voltage sources. Because of the choice of standard branch shown in Fig. 2, the network can always be modeled with current sources only..

In both (3) and (4) it is necessary to invert the matrix $1+YR_m-\alpha$ in order to formulate the node equations. Johnson and Pennington [2] have shown that if the network can be modeled so that the branch current j_k of a branch which contains a current controlled source does not control any source, then the inverse of $1+YR_m-\alpha$ is $1-YR_m+\alpha$. Since most networks can be modeled this way, forming the node equations does not present any difficulty.

The accessible nodes have been numbered 0–p, and the inaccessible nodes have been numbered $p+1$ through n. Partition the rows of the node-to-branch incidence matrix A into accessible and inaccessible nodes

$$A=\left[\begin{array}{c}A_1\\ \hline A_2\end{array}\right] \tag{5}$$

where A_1 is the $p\times n$ incidence matrix of accessible nodes, and A_2 is the $q\times n$ incidence matrix of inaccessible nodes, and $q=n-p$ is the number of inaccessible nodes. Using (5) in (2) and partitioning i and e as follows:

$$i=\left[\begin{array}{c}i_1\\ \hline i_2\end{array}\right] \qquad e=\left[\begin{array}{c}e_1\\ \hline e_2\end{array}\right]$$

where i_1 and e_1 are $p\times 1$ vectors, and i_2 and e_2 are $q\times 1$ vectors, we obtain

$$\left[\begin{array}{c:c}A_1Y_bA_1^T & A_1Y_bA_2^T\\ \hline A_2Y_bA_1^T & A_2Y_bA_2^T\end{array}\right]\left[\begin{array}{c}e_1\\ \hline e_2\end{array}\right]=\left[\begin{array}{c}i_1\\ \hline i_2\end{array}\right]. \tag{6}$$

Now eliminate e_2 from (6) and make use of the fact that $i_2=0$, since no independent sources are connected to inaccessible nodes, to obtain

$$i_1=A_1Y_b\left[1-A_2^T\left(A_2Y_bA_2^T\right)^{-1}A_2Y_b\right]A_1^Te_1 \tag{7}$$

so that

$$y_i\triangleq A_1Y_b\left[1-A_2^T\left(A_2Y_bA_2^T\right)^{-1}A_2Y_b\right]A_1^T \tag{8}$$

is the $p\times p$ matrix of measurable y-parameters defined at the ports having a common "ground" node (node 0 in Fig. 1). The subscript i indicates that the y-parameters are evaluated at frequency ω_i.

III. Fault Analysis

Let $\hat{y}_i$ be the matrix of measured y-parameters, where the measurement is made at frequency ω_i. A fault represented by a shift of the network element values from their nominal values could be identified by solving the nonlinear equation

$$\hat{y}_i=y_i(Y_b) \tag{9}$$

for the unknown parameter values. If a single frequency is used, as in (9), the number of network parameters usually exceeds the number of independent y-parameters, so that (9) has no unique solution. One approach, suggested by Ransom and Saeks [3], is to find the solution to (9) that minimizes a norm $\|Y_{b0}-Y_b\|$, where Y_{b0} is evaluated from the nominal network element values. Ransom and Saeks give an approximate solution to this problem using linear methods. The difficulty with this approach is that the solution assumes, roughly speaking, that the most likely state of the element values in the network is the one which

causes the smallest drift from the nominal values consistent with the measurements. This assumption excludes the possibility of catastrophic faults (open and short circuits).

Another approach is to augment (9) by measurements at a sufficient number of frequencies that a unique solution exists. The relationship between fault resolution and the number of frequencies and measurements is discussed by Sen and Saeks [9], [10]. Although their algorithm requires choosing a number of frequencies, it is not clear how this is done. Thus we might define:

$$\hat{y} = (\hat{y}_1\, \hat{y}_2 \cdots \hat{y}_k)^T$$
$$y = (y_1\, y_2 \cdots y_k)^T \tag{10}$$

and solve the equation

$$\hat{y} = y(Y_b). \tag{11}$$

Nonlinear equations, such as (11), are often solved by defining a norm $c = \|\hat{y} - y\|$, and then making a multidimensional search for min c. The amount of computation required by such a search could be prohibitive. Chen and Saeks [4] show that if only one parameter is changed at a time, the measurable system function can be evaluated without the need to invert a matrix, which is apparently necessary in (8). Their method, is based on a formula given by Householder [5].

In this paper we show how the Householder formula can be used to efficiently analyze networks containing short circuits by considering the change in the network graph. The measurable effects of open circuits and finite parameter changes can be analyzed by considering changes in Y_b, and although this is essentially equivalent to the method described by Chen and Saeks, it is given here in the context of nodal analysis so that all parameter changes can be analyzed with one formulation of the network equations.

A. Short Circuits

If two nodes of a network become shorted, the new node-to-branch incidence matrix A' is related to the original incidence matrix in a very simple way. If nodes i and j are shorted, then A' is formed by replacing row i by the sums of rows i and j, and then deleting row j. If node i is the ground node, then A' is formed by deleting row j. These row transformations can be obtained by premultiplying A by a matrix R, each element of which is either 0 or 1. Fig. 3 illustrates the row transformation matrix describing two different short circuits (one involving the ground node and the other not involving the ground node) in a network with 5 nodes and 7 branches. The incidence matrices are partitioned by the dashed lines under the assumption that nodes 3 and 4 in the original network are inaccessible. Thus the top half of the A matrix is A_1 and the bottom half is A_2. A similar partition is shown for $A' = (A_1' \,\vdots\, A_2')^T$, but in the transformed network some of the formerly inaccessible nodes may have been made accessible (by virtue of their having been shorted to an accessible node).

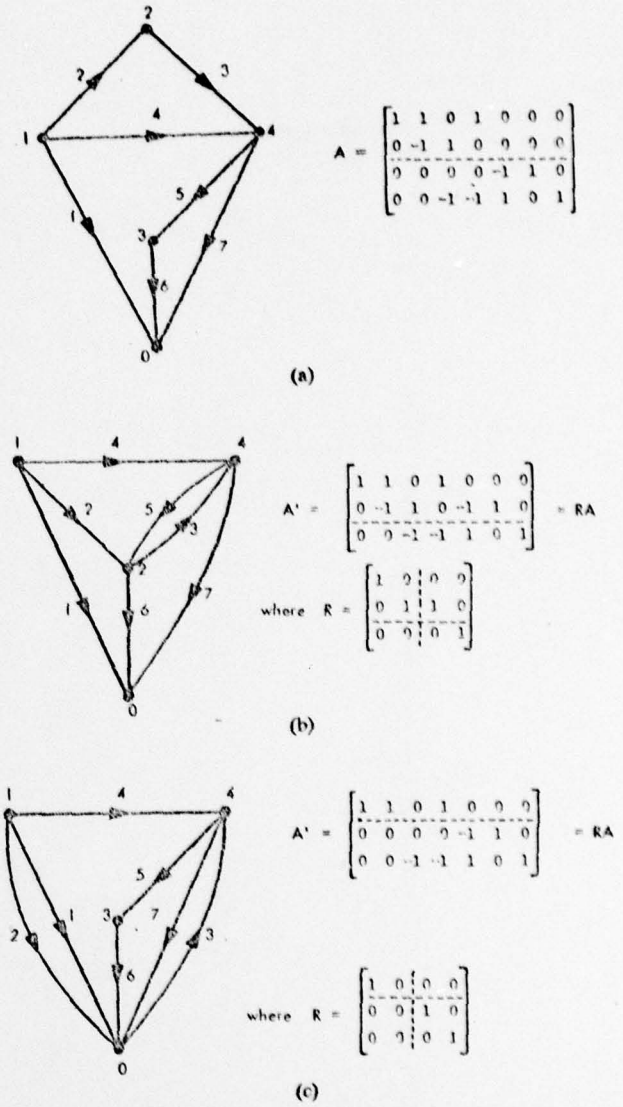


Fig. 3. Effect of short circuits on node-to-branch incidence matrix, A. (a) Original network graph. (b) Short circuit between nodes 2 and 3. (c) Short circuit between nodes 2 and ground or 0.

The R matrix is the "after-to-before" node transformation matrix, which has a column for every original node in the network and a row for every node after the fault. (Since node 0 does not enter into A, it does not appear in R either.) Thus R may be partitioned by columns and rows into the accessible and inaccessible nodes before (columns) and after (rows) the fault, as shown in Fig. 3.

Denote the partitioning of R by

$$R \triangleq \begin{bmatrix} R_{11} & R_{12} \\ \mathbf{0} & R_{22} \end{bmatrix} \tag{12}$$

where the lower, left partition in (12) is 0 since the faulted network cannot have any inaccessible nodes which were accessible in the unfaulted network. In other words, if a short circuit occurs between an accessible and an inacces-

sible node, the combined node becomes accessible. Thus

$$A' = RA = \left[\begin{array}{c} R_{11}A_1 + R_{12}A_2 \\ \hline R_{22}A_2 \end{array}\right]$$

and, therefore,

$$\begin{aligned} A_1' &= R_{11}A_1 + R_{12}A_2 \\ A_2' &= R_{22}A_2. \end{aligned} \tag{13}$$

B. Forming the y-Parameters of the Shorted Network

The y-parameters of networks containing a short circuit can be computed from (8) after replacing A_1 by A_1' and A_2 by A_2'. Define $Y_2 \triangleq A_2 Y_b A_2^T$, which is the $q \times q$ nodal admittance matrix of inaccessible nodes. In the analysis that follows, we assume that Y_2 is known. (In fact Y_2^{-1} must be computed only once for any circuit using the nominal element values, so the cost of its computation is a negligible part of the total cost of analysis.)

If the short circuit occurs between two accessible nodes, then $A_2' = A_2$, and the new y-parameters are found from (8) with no matrix inversion. In fact, if a short circuit exists between two accessible nodes, it can be easily observed by making resistance measurements between all pairs of accessible nodes, so the method described here is not needed.

If the short circuit involves any inaccessible node, then $A_2' \neq A_2$ and the inverse of $Y_2' = A_2' Y_b A_2'^T$ must be computed. The main contribution of this paper is to show that the inverse of Y_2' may be found in terms of Y_2 with about one fifteenth as many multiplications as are required to find the inverse of a general $q \times q$ matrix. This saving, described in detail in the next section, is due to the fact that Y_2' is closely related to Y_2, whose inverse is kown. An additional saving of computation and memory is realized by the use of sparse matrix techniques, which are discussed in Section IV.

C. Finding the Inverse of Y_2' Efficiently

From the definitions of A_2', R_{22}, Y_2, and Y_2' it is easy to show that

$$Y_2' = R_{22} Y_2 R_{22}^T. \tag{14}$$

Let us add a row to R_{22} as follows

$$S \triangleq \left[\begin{array}{c} R_{22} \\ \hline u^T \end{array}\right] \tag{15}$$

where u^T is a row unit vector containing all zeros except for a 1 in the right-most column representing a shorted node. (At most two columns of R_{22} are involved in the short circuit. If only one is involved, then that column of u^T is unity. If two columns of R_{22} are involved, then only the right-most one is unity. This convention could be reversed by changing the definition of R.) We now decompose S into U and P,

$$S = U + P \tag{16}$$

where P is formed as follows. If the short circuit is between an accessible and an inaccessible node, then $P = 0$. If the short circuit is between two inaccessible nodes, then $\{P\}_{i-p,j-p}$ is unity, and the remaining elements are zero. (In this case, the unit element in P is element $\{R\}_{ij}$, where nodes i and j are shorted, and $i < j$ due to our numbering convention.) With these definitions, it is easy to see that U is orthogonal, since its columns are independent unit vectors and thus form an orthonormal se' [7]. Consequently $U^{-1} = U^T$, and if the short circuit is between an accessible and an inaccessible node, then $S^{-1} = S^T$. If the short circuit is between two inaccessible nodes, then S^{-1} is only slightly more difficult to compute, as we will show. Now define $C \triangleq S Y_2 S^T$, and, therefore,

$$C = \left[\begin{array}{c} R_{22} \\ \hline u^T \end{array}\right] Y_2 \left[\begin{array}{c:c} R_{22}^T & u \end{array}\right] = \left[\begin{array}{c:c} Y_2' & R_{22}Y_2 u \\ \hline u^T Y_2 R_{22}^T & u^T Y_2 u \end{array}\right]$$

and

$$C^{-1} = (S^T)^{-1} Y_2^{-1} S^{-1} \triangleq \left[\begin{array}{c:c} K & L \\ \hline M & N \end{array}\right]. \tag{17}$$

Faddeeva [8] shows that $K = Y_2'^{-1}[1 - (R_{22}Y_2 u)M]$, and, therefore,

$$Y_2'^{-1} = K\left[1 - (R_{22}Y_2 u)M\right]^{-1}. \tag{18}$$

The inverse on the right side of (18) can be computed efficiently using a formula given by Householder [5], [6], which states that if $A = B - vw^T$, where v and w are column vectors, then

$$A^{-1} = B^{-1} + \frac{B^{-1}vw^T B^{-1}}{1 - w^T B^{-1} v} \tag{19}$$

where the division on the right side of (19) is by a scalar. Thus the inverse of Y_2' is

$$Y_2'^{-1} = K\left[1 + \frac{(R_{22}Y_2 u)M}{1 - M(R_{22}Y_2 u)}\right]. \tag{20}$$

In order to use (20), K and M must be found from (17). We have mentioned that if the short circuit is between an accessible node and an inaccessible node, then $S^{-1} = S^T$. In fact, if the short circuit is between an accessible node and node n, then $S = I$, and K and M are simply partitions of (17). If the short circuit is between two inaccessible nodes, we have shown that $S = U + P$, where $U^{-1} = U^T$, and the rank of P is unity. Thus P can be written in the form vw^T, and (19) can be used to find S^{-1}. Proceeding in this manner it can be shown that $S^{-1} = U^T + Q$, where

$$Q_{rs} = \begin{cases} -1, & \text{if } r = i-p,\ s = n-p \\ 0, & \text{otherwise} \end{cases}.$$

And so, even in the most complex case, K and M can be evaluated without a matrix inversion. Some time can be saved in computing the triple matrix product in (17) by noting that L and N (each column vectors) are not needed, so the last column of C^{-1} need not be evaluated. Using straightforward matrix multiplication to find K and

M from (17) and then $Y_2'^{-1}$ from (20) requires $q^2(q^4-2q^3+3q^2-q+1)$ complex multiplications, where q is the number of accessible nodes in the network.

This method of computing $Y_2'^{-1}$ is unnecessarily inefficient since many of the matrices in (17) and (20) are sparse, and others (S and R_{22}), in addition, are special since their nonzero elements are either $+1$ or -1. In Secton IV-A we show how to reduce the number of complex multiplications required to evaluate $Y_2'^{-1}$ to $\beta q^2(q-1)$, where β is typically about 0.3 and is always a positive number less than unity.

Once $Y_2'^{-1}$ is known, the y-parameters at the accessible nodes are found from (8). Evaluating the matrix products in (8) in a straightforward manner requires $2b^3+(2p+q-1)b^2+(q-1)^2b$ multiplications, where b is the number of branches in the network, p is the number of accessible nodes not counting the reference node, and q is the number of inaccessible nodes. Using the sparse matrix techniques described in Section IV-A, the number of multiplications needed to evaluate y_i from $Y_2'^{-1}$ can be reduced to $2(1+\gamma)b^2$, where γ is typically about 0.1. For example, a network with 20 branches, 3 accessible nodes (plus the accessible reference node), and 8 inaccessible nodes requires 14180 multiplications to evaluate y_i directly from (8). By using sparse matrix techniques, this is reduced to 880 multiplications, a saving of a factor of 16 in this case. The straightforward evaluation in general requires nearly b times as many multiplications as the more efficient approach using sparse matrix techniques, so the saving increases with network complexity.

D. Open Circuits and Other Large Parameter Changes

For completeness, we include a discussion of open circuits and other large parameter changes, although this has been discussed elsewhere [4]. The main purpose in this section is to show how the analysis can be carried out in the context of nodal analysis.

Parameter changes are accounted for in (8) by a change in the $b\times b$ branch admittance matrix Y_b. In order to evaluate the new y-parameters from (8), the inverse of the inaccessible nodal admittance matrix $Y_2=A_2Y_bA_2^T$ must be found. If a single parameter in the network is changed, then the new branch admittance matrix is $Y_b' \triangleq Y_b+\Delta$, where Δ has only one nonzero element representing the change in one parameter. Suppose a change occurs in element ij. Then Δ can be written duv^T, where d is a scalar equal to the parameter change, u is a unit vector in direction i, and v is a unit vector in direction j. Now define $U \triangleq A_2u$ and $V \triangleq A_2v$, so that $Y_2'=Y_2+dUV^T$. Householder shows that the inverse of Y_2' can be found from

$$Y_2'^{-1}=Y_2^{-1}\left[1-U\left(d^{-1}+V^TY_2^{-1}U\right)^{-1}V^TY_2^{-1}\right]. \quad (21)$$

The inverse of Y_2 is known, and $d^{-1}+V^TY_2^{-1}U$ is a scalar, so (21) can be evaluated without a matrix inversion.

IV. Sparse Matrix Techniques

Sparse matrices arise in both the evaluation of the inverse if Y_2' and in the evaluation of the accessible y-parameter matrix y_i. These are discussed separately in Sections IV-A and IV-B.

A. Evaluating the Inverse of Y_2'

We have shown that the inverse of Y_2' may be found from (20), which is repeated below for convenience:

$$Y_2'^{-1}=K\left[1+(R_{22}Y_2u)M/(1-M(R_{22}Y_2u))\right]$$

where K and M are defined by

$$\begin{bmatrix} K & L \\ M & N \end{bmatrix}=(S^T)^{-1}Y_2^{-1}S^{-1}. \quad (22)$$

The dimensions of K and M are, respectively, $(q-1)\times(q-1)$ and $1\times(q-1)$.

In (22) Y_2^{-1} is known, and S is defined by

$$S=\begin{bmatrix} R_{22} \\ u^T \end{bmatrix}. \quad (23)$$

R_{22} is the matrix which transforms A_2, the (inaccessible node)-to-branch incidence matrix before the fault, into A_2', the (inaccessible node)-to-branch incidence matrix after the fault, so that

$$A_2'=R_{22}A_2. \quad (24)$$

In the discussion that follows, the reader may find it helpful to refer to the illustration of Fig. 3. Suppose nodes i and j are shorted, where $j>i$. Then rows 1 through $i-1$ of R_{22} contain zeros except that the diagonal element is unity. This indicates that nodes 1 through $i-1$ in the faulted network are identical to nodes 1 through $i-1$ in the unfaulted network. Row i of R_{22} contains all zeros except for unity elements in columns i and j, indicating that node i in the faulted network is formed by combining nodes i and j in the unfaulted network. Finally, the remaining rows of R_{22} contain all zeros, except that the first row to the right of the diagonal is unity. This indicates that the remaining nodes of the faulted network are identical to the remaining nodes of the unfaulted network. The faulted network contains one less node than the unfaulted network, so the dimension of R_{22} is $(q-1)\times q$, where, as usual, q is the number of inaccessible nodes.

We have chosen to number the nodes consecutively, beginning with the accessible nodes. We have also avoided analyzing networks with a short circuit between two accessible nodes, since such a fault may be found from driving point impedance measurements between all pairs of accessible nodes. Thus node j is always inaccessible, but node i may be either accessible or inaccessible. In the description above we have assumed that node i is inaccessible. If it is accessible, then R_{22} is simpler than described above, since all rows contain a single unit element. (The unit element is on the diagonal in rows 1 through $j-1$ and in the first column to the right of the diagonal in all other rows. This indicates that all inaccessible nodes in the faulted network correspond to an inacces-

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&1&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0 \end{bmatrix}$$

(b)

Fig. 4. S matrices for a network with 8 accessible nodes. (a) Short circuit between nodes 4 and 6. (b) Short circuit between nodes 6 and any accessible node.

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&-1\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

(b)

Fig. 5. (a) Inverse of the matrix of Fig. 4(a). (b) Inverse of the matrix of Fig. 4(b).

sible node in the unfaulted network, but one inaccessible node in the unfaulted network is now accessible, and so it has no row in R_{22}.)

In (23) u^T is a row vector containing all zeros except that column j is unity.

To illustrate the formation of the S matrix, consider a network with 8 inaccessible nodes. (For convenience in this discussion they are numbered 1–8, instead of $p+1$ through $p+8$.) Fig. 4(a) shows the S matrix for a short circuit between nodes 4 and 6, and Fig. 4(b) shows the S matrix for a short circuit between node 6 and any accessible node. The formation of the inverse of the S matrix is given by $S^{-1} = U^T + Q$, where

$$[Q]_{rs} = \begin{cases} -1, & \text{if } r = i-p,\ s = n-p \\ 0, & \text{otherwise} \end{cases}$$

as explained in Section III-C, where $S = U + P$, and all elements of P are zero except if the shorted nodes i and j are both inaccessible, in which case $[P]_{ij}$ is unity. The unity element of P is circled in Fig. 4(a). In this illustration we have taken $p = 0$ for convenience, where p is the number of accessible nodes. Also note that $n - p = q$. The inverse of the S matrix can be formed by inspection, and in particular, Fig. 5 shows the inverses of the matrices of Fig. 4.

We now turn to the problem of evaluating K and M. The right side of (22) may be written

$$(U+Q^T)Y_2^{-1}(U^T+Q) = UY_2^{-1}U^T + UY_2^{-1}Q + Q^TY_2^{-1}U^T + Q^TY_2^{-1}Q. \quad (25)$$

Recall that the right-most column (column q) of (25) is not needed. The second and fourth terms on the right side of (25) are zero except for the right-most column, so they need not be evaluated.

To evaluate $UY_2^{-1}U^T$, define the row vector $V^T \triangleq [v_k]$, where $v_k = (\text{unit column of row } k \text{ in } U)$, $k = 1, 2, \cdots, q-1$. Thus

$$[K]_{mn} = [UY_2^{-1}U^T]_{mn} = [Y_2^{-1}]_{v_m, v_n}, \quad \begin{matrix} m = 1,2,\cdots,q-1 \\ n = 1,2,\cdots,q-1 \end{matrix} \quad (26)$$

Finally M is the $1 \times (q-1)$ row vector whose elements are the first $q-1$ columns of the last row (row q) of the sum of terms one and three of (25). Thus

$$[M]_n = [UY_2^{-1}U^T]_{q,n} + [Q^TY_2^{-1}U^T]_{q,n} = [Y_2^{-1}]_{j,v_n} - [Y_2^{-1}]_{i,v_n}, \quad n = 1,2,\cdots,q-1. \quad (27)$$

K and M are evaluated from (26) and (27) with no multiplications, and the evaluation of K, the larger of the two, requires no additions.

Now that K and M are known, Y_2^{-1} can be found from (21). In evaluating (21) it is useful to first find the $(q-1) \times 1$ column vector $x \triangleq R_{22}Y_2u$. Recall that u is a vector with all zero elements except that row j is unity. Thus x is the jth column of $R_{22}Y_2$. R_{22} is the partition of S formed from the first $q-1$ rows (23). We may make use of the

row vector V^T to form x as follows:

$$[x]_n = [R_{22}Y_2u]_n = \begin{cases} [Y_2]_{v_n,j}, & \text{if node } i \text{ is accessible} \\ \left.\begin{aligned} &[Y_2]_{v_n,j}, && v_n \neq i \\ &[Y_2]_{v_n,j} + [Y_2]_{j,j}, && v_n = i \end{aligned}\right\}, & \text{if node } i \text{ is inaccessible} \end{cases}$$
$$n = 1,2,\cdots,q-1. \quad (28)$$

Using the definition of x in (28), (21) may be written

$$Y_2'^{-1} = K + K[xM/(1-Mx)] \quad (29)$$

which requires evaluating three matrix products, two of which directly involve x. From (28) it is clear that x is formed by reordering the elements of column j of Y_2. Since only a few inaccessible nodes are normally coupled to node j, x is usually sparse. Define β equal to the fraction of nonzero elements of column j of Y_2. The number of multiplications required to compute Mx is $\beta q \leq q$. β is usually significantly less than one, and so it is worth designing the algorithm to avoid the multiplications by zero. Similarly, βq rows of xM are zero, and it is worth forming only the nonzero rows. Thus (29) can be evaluated with $\beta q^2(q-1)$ multiplications and divisions. Note that the multiplications in (29) are the only multiplications required to evaluate $Y_2'^{-1}$, since K and M have been formed in (26) and (27) without multiplications.

B. Evaluating the y-Parameters

In this section we discuss the evaluation of the y-parameter matrix y_i given by (8). In the faulted network, the node-to-branch incidence matrices are A_1' and A_2', so that (8) becomes

$$y_i = A_1'Y_b[1 - A_2'^TY_2'^{-1}A_2'Y_b]A_1'^T. \quad (30)$$

Two forms appear in (30) which must be discussed separately: A^TYA and AYA^T.

The most convenient way to store the node-to-branch incidence matrices, which we will refer to generally by A in the discussion which follows, is by two column vectors A^+ and A^-, defined as follows. If the kth column of A has a $+1$ element, then $[A^+]_k \triangleq$ (row number of the $+1$ element of the kth column of A), otherwise $[A^+]_k \triangleq 0$. Similarly, if the kth column of A has a -1 element, then $[A^-]_k \triangleq$ (row number of the -1 element of the kth column of A), otherwise $[A^-]_k = 0$. The formation of A^+ and A^- corresponding to a given node-to-branch incidence matrix A is illustrated in Fig. 6. Note that in practice it is necessary to store only the vectors A^+ and A^- and not the matrix A.

Now consider the evaluation of $X = A^TYA$. Element $[X]_{m,n}$ is given by

$$\begin{aligned}[X]_{m,n} &= \sum_k [A^T]_{m,k} \sum_l [Y]_{k,l} \cdot [A]_{l,n} \\ &= \sum_{k,l} [A]_{k,m} \cdot [A]_{l,n} \cdot [Y]_{k,l} \\ &= [Y]_{[A^+]_m,[A^+]_n} + [Y]_{[A^-]_m,[A^-]_n} \\ &\quad - [Y]_{[A^+]_m,[A^-]_n} - [Y]_{[A^-]_m,[A^+]_n}. \end{aligned} \quad (31)$$

$$A = \begin{pmatrix} -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

$$A^+ = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \\ 3 \\ 3 \\ 4 \\ 0 \\ 5 \end{pmatrix} \qquad A^- = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 2 \\ 5 \\ 4 \\ 4 \\ 0 \\ 5 \\ 0 \end{pmatrix}$$

Fig. 6. Node-to-branch incidence matrix A, and the corresponding compact equivalent vectors A^+ and A^-.

If any of the subscripts of $[Y]$ in the last expression is zero, the term is zero.

The evaluation of $X' = AYA^T$ is somewhat different because the summation is along the rows of A instead of down the columns. In particular, element $[X']_{m,n}$ is given by

$$\begin{aligned}[X']_{m,n} &= \sum_k [A]_{m,k} \sum_l [Y]_{k,l} \cdot [A^T]_{l,n} \\ &= \sum_{k,l} [A]_{m,k} \cdot [A]_{n,l} \cdot [Y]_{k,l}. \end{aligned} \quad (32)$$

The right side of (32) cannot be evaluated by choosing m and n as in (31). Instead k and l are chosen and, for each choice, $[Y]_{k,l}$ is added to elements $[X']_{[A^+]_k,[A^+]_l}$ and $[X']_{[A^-]_k,[A^-]_l}$ and subtracted from elements $[X']_{[A^+]_k,[A^-]_l}$ and $[X']_{[A^-]_k,[A^+]_l}$. If any of the subscripts of $[X']$ is zero, that term is zero.

Products of the forms $X' = AYA^T$ and $X = A^TYA$ both appear in (30), and we have shown how to evaluate them without multiplication. An additional product of the form $X' \cdot Y_b$ must be evaluated, where Y_b is given by (3). Y_b is typically very sparse, so in the software we have chosen to store it as three vectors: one containing the nonzero elements of Y_b, and the other two containing, respectively, the row and column of that element. The saving in storage can be appreciated by recognizing that Y_b is a $b \times b$ matrix, where b is the number of branches in the network. The elements of Y_b which are not zero are the b diagonal elements plus one additional element for each controlled source. If the number of controlled sources is $b/10$, then

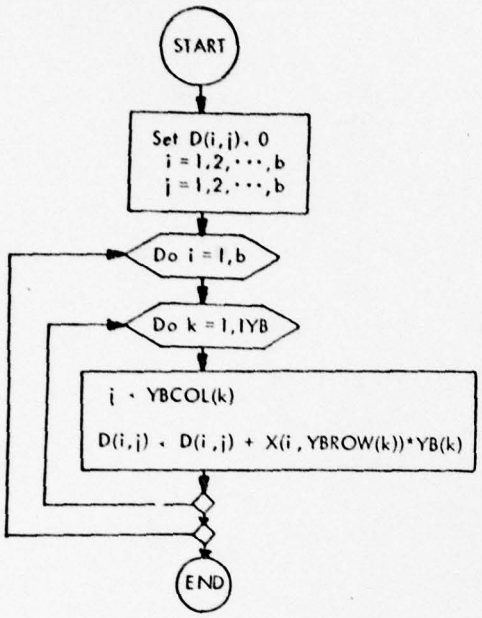


Fig. 7. Efficient formation of the matrix product $X' \cdot Y_b$.

the sparse storage of Y_b requires storing $1.1b$ complex numbers and $2.2b$ integers, compared with the b^2 complex numbers required for conventional storage. If the software is written to accommodate networks with 100 branches, then conventional storage requires setting aside room for 10^4 complex numbers, compared to about 120 complex numbers and 240 integers required by compact storage.

Considerable computation time is also saved by storing Y_b sparsely. The matrix product $X' \cdot Y_b$ in (30) would require b^3 complex multiplications if performed directly, since the dimensions of both X' and Y_b are $b \times b$. The algorithm of Fig. 7 forms the product in $(1+\gamma)b^2$ complex multiplications, where γ is the fraction of the b branches containing a controlled source. Typically γ is about 0.1. In Fig. 7, $YB(\cdot)$ is a vector containing the IYB nonzero elements of Y_b, and $YBROW(\cdot)$ and $YBCOL(\cdot)$ are integer vectors containing the row and column numbers of each element of $YB(\cdot)$. The product $X' \cdot Y_b$ is called $D(\cdot,\cdot)$. $D(\cdot,\cdot)$ is not sparse, so it is stored conventionally.

Examination of (30) reveals another matrix product of the form $Y_b \cdot Z$, where in this case $Z = 1 + A_2^T Y_2'^{-1} A_2' Y_b$. This matrix product is formed in a similar manner to $X' Y_b$ in $(1+\gamma)b^2$ complex multiplications. Thus altogether $2(1+\gamma)b^2$ complex multiplications are required to formulate y_i. A direct evaluation of (30) requires $2b^3 + (q-1+p)b^2 + ((q-1)^2+p^2)b$ multiplications. Taking $p=2$, $b=20$, $q=11$ we find that y_i requires 22880 multiplications by direct methods, but only 880 multiplications by the methods described in this section with $\gamma = 0.1$, a saving of a factor of 26 in this example.

Before closing this section it should be noted that the y-parameters of the unfaulted network are evaluated before evaluating the y-parameters of any faulted networks. This is done by applying (30) with A_1' replaced by A_1, A_2' replaced by A_2, and $Y_2'^{-1}$ replaced by Y_2^{-1}. The inverse of Y_2 must be computed by conventional methods, but the methods of this section may be used for the remaining computation.

C. Summary of Efficient Computation

The y-parameters at the accessible ports of the faulted network are found from (30), where in the context of this report the faulted network is formed from the unfaulted network by shorting a single pair of nodes.

In Section IV-A we showed how to form the inverse of the $(q-1)\times(q-1)$ matrix Y_2' with only $\beta q^2(q-1)$ complex multiplications, where β is the fraction of nonzero elements in Y_2. (Recall that $Y_2 = A_2 Y_b A_2^T$ is the inaccessible nodal admittance matrix.) Inversion of a $(q-1)\times(q-1)$ matrix requires about $5(q-1)^3$ complex multiplications using Wilf's rank annihilation method[2] [11], which is an efficient application of the Householder formula (19) [5]–[7].

In Section IV-B we showed that (30) can be evaluated in $2(1+\gamma)b^2$ complex multiplications. This result applies, whether or not the efficient evaluation of the inverse of Y_2' of Section IV-A is used. Thus the total number of complex multiplications required to evaluate the y-parameters at the accessible ports of the network with one pair of shorted terminals is $2(1+\gamma)b^2 + q^2(q-1)$. If the inverse of Y_2' is found by conventional methods but the sparse matrix techniques of Section IV-B are used to form y_i, then $2(1+\gamma)b^2 + 5(q-1)^3$ complex multiplications are required.

The y-parameters resulting from all possible single short circuits involving at least one inaccessible node must be evaluated. If the network contains p accessible terminals and q inaccessible terminals, then it can be shown that $N = q(p+q) - q(q-1)/2$ distinct pairs of shorted terminals can be identified. The total number of complex multiplications is, therefore, $N[2(1+\gamma)b^2 + \beta q^2(q-1)]$ using the efficient method in inverting Y_2' and $N[2(1+\gamma)b^2 + 5(q-1)^3]$ using a conventional inversion method. To show the benefit of the efficient matrix inversion, let us assume that the number of accessible ports p is 2, the number of branches b is twice the number of nodes, $p+q+1$, the fraction γ of branches having controlled sources is 0.1, and the fraction β of nonzero elements of Y_2' is 0.3. Using these assumptions, the total number of complex multiplications necessary to analyze all the inaccessible short circuits using both the efficient and the conventional methods of inverting Y_2' are shown in Fig. 8. The advantage of the efficient method is especially noticable for the larger networks.

V. Example

We have described an efficient way of computing the y-parameters at the accessible terminals of a network with

[2]An error appears in our edition of Ralston and Wilf [11]. In the flow chart on p. 75, step 11 should be $1 \rightarrow i$, not $2 \rightarrow i$.

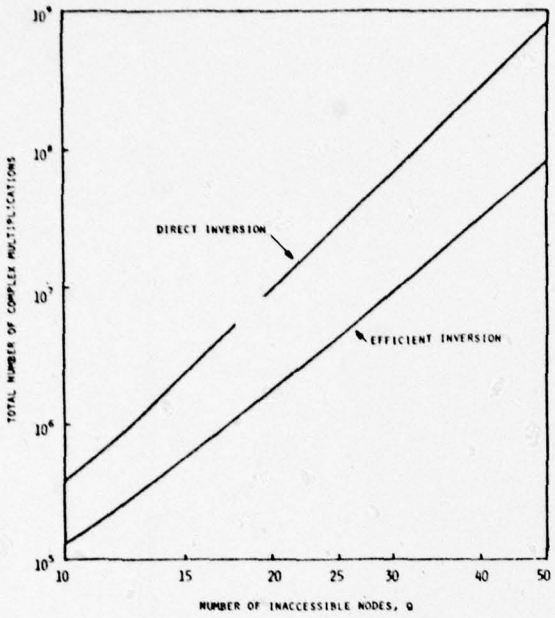


Fig. 8. Number of complex multiplications required to analyze all inaccessible short circuits in a network with q inaccessible nodes.

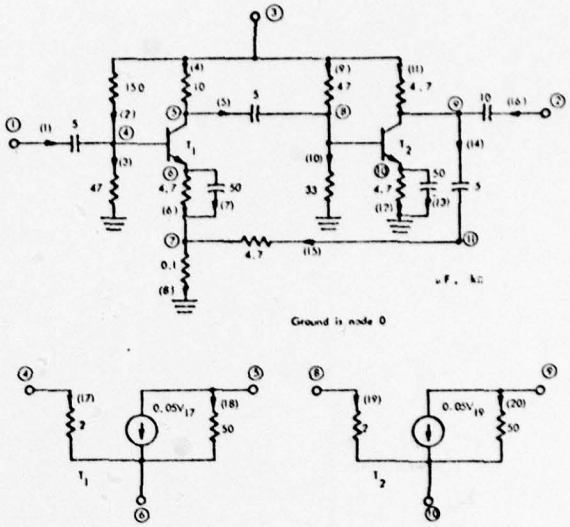


Fig. 9. Feedback amplifier.

an internal short circuit. The question remains how useful this is in locating short-circuit faults, since the network parameters (resistance, capacitance, inductance, and controlled source gains) of the faulted network will be different from their nominal values. One possible way to locate the fault is to simulate all single short circuits (involving at least one inaccessible terminal) using the nominal parameters and the algorithm described in this paper, compare each of the simulated y-parameters to the measured y-parameters, and choose the closest set to locate the short circuit. Next a search over the parameter values could be conducted, as described by Chen and Saeks [4] or in Section III-D.

To date the short circuit algorithm has been tested using the network of Fig. 9. Measured data for the nine complex y-parameters at the three accessible ports (nodes 1-0, 2-0, and 3-0) were obtained by simulating single short circuits and setting the network parameters between 10 and 20 percent above or below nominal. It was expected that this might completely mask the effect of some short circuits. The cost function used to compare the "measured" and simulated y-parameters was

$$c = \sum_{i,j} \frac{|y_{ij} - \hat{y}_{ij}|}{|\hat{y}_{ij}|}$$

where y_{ij} is the y-parameter simulated from the fault model and $\hat{y}_{ij}$ is the "measured" y-parameter.

Table I shows the actual fault, the fault determined by the minimum cost function, and the value of the cost function of the correct fault. Of the seven examples at 1000 Hz, the algorithm correctly located five pairs of shorted nodes. The two errors seemed to be caused by the fact that the magnitude of the capacitor impedances at 1000 Hz were much lower than the impedance levels of the network resistances. (The magnitude of the impedance of a 10-μF capacitor at 1000 Hz is 15.9 Ω.) It was expected that the short circuit between nodes 6 and 7 would be difficult it identify, since the 50-μF bypass capacitor is only about 3 Ω at 1000 Hz, and so the changes in the parameters of the measured circuit could be expected to mask the short circuit. In fact, one of the reasons for this test was to determine how much the network parameters could be changed before this change prevented reliable identification of the short circuit. In this example, variations of the parameters of 10–20 percent from nominal still permitted correct identification of 5 out of 7 short circuits, or 71 percent of the cases tried. If the parameters of the "measured" network were set to the nominal values, then correct identification would occur, of course, in 100 percent of the cases, since then no error would exist between the "measured" and simulated y-parameters.

TABLE I
SUMMARY OF TEST RESULTS I

FREQUENCY, Hz	ACTUAL SHORTED NODES	SHORTED NODES OF MINIMUM COST FUNCTION	MINIMUM COST FUNCTION VALUE	COST FUNCTION VALUE OF CORRECT FAULT
1000	0-8	0-8	1.087	1.087
	0-5	0-5	1.115	1.115
	0-10	0-10	1.290	1.290
	6-7	5-8	1.271	1.34
	3-5	3-8	1.122	1.20
	3-9	3-9	2.828	2.828
	7-8	7-8	1.632	1.632

VI. Conclusions

We have shown an efficient way to analyze networks containing a single short circuit or other larger parameter change. The efficiency results from the fact that the network change caused by either the short circuit or other large parameter change results in an alteration of the inaccessible nodal admittance matrix by a matrix of unit rank. The response of the faulted network can therefore be computed in terms of the nominal response with many fewer multiplications than are required for and original analysis. Sparse matrix techniques were also used to reduce execution time.

The efficiency of the algorithm makes it practical to exhaustively search all possible single short circuits involving an inaccessible node in order to find a solution to the problem of locating a short circuit in an analog network from measurements at the accessible terminals. The test examples described in Section V show that short circuits can be reliably located by the algorithm, even though measurements were made at only one frequency and differences between the actual and the nominal parameter values were not taken into account. We expect that the reliability can be improved by making measurements at more than one frequency and by solving for the parameter values.

References

[1] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory*. New York: McGraw-Hill, 1969, pp. 409–463.

[2] A. T. Johnson, Jr., and A. J. Pennington, "Network equations with coupling elements," *Proc. 14th Midwest Symp. Circuit Theory*, pp. 2.5-1–2.5-7, May 1971.

[3] M. N. Ransom and R. Saeks, "Fault isolation with insufficient measurements," *IEEE Trans. Circuit Theory*, vol. CT-20, pp. 416–417, July 1973.

[4] H. S. M. Chen and R. Saeks, "Research on fault analysis", in *Annual Research Rep*, Texas Tech, Lubbock, TX, 1977.

[5] A. S. Householder, "A survey of some closed methods for inverting matrices," *J. SIAM*, vol. 5, pp. 155–169, 1957.

[6] B. Noble, *Applied Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1969, pp. 147–148.

[7] ——, *Applied Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1969, p. 313.

[8] V. N. Faddeeva, *Computational Methods of Linear Algebra*. New York: Dover, 1959, pp. 102–103.

[9] N. Sen and R. Saeks, "A measure of testability and its application to test point selection–Theory," *Proc. 20th Midwest Symp. Circuits Sys.* Texas Tech. Univ., Lubbock, Texas, Aug. 1977, pp. 576–583.

[10] ——, "A Measure of testability and its application to test point selection–Computation," *Proc. AUTOTESTCON '77*, pp. 212–219 (Hyannis, MA), Nov. 1977.

[11] A. Ralston and H. Wilf, *Mathematical Methods for Digital Computers*, Vol. 1. New York: Wiley, 1967, pp. 73–77.

[12] R. Saeks and S. R. Liberty, *Rational Fault Analysis*. New York: Marcel Dekker, 1977.

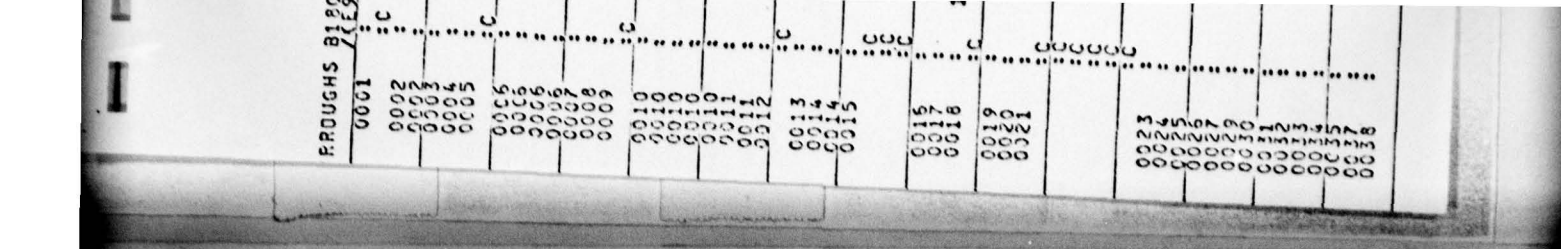
[13] "Large ATE systems," in *Eval. Eng.*, vol. 17, no. 2, March/April 1978, pp. 28–50

[14] E. A. Torrero, "ATE–Not so easy," *IEEE Spectrum*, vol. 14, pp. 29–34, Apr. 1977.

Alfred T. Johnson, Jr. (M'63) was born in Philadelphia, PA, on June 24, 1941. He received the B.S. degree in electrical engineering (highest honors) from Drexel Institute of Technology, Philadelphia, PA, in 1963, and the Ph.D. in electrical engineering from the University of Pennsylvania, Philadelphia, in 1969.

From 1970 to 1974 he was Assistant Professor of Electrical Engineering at Drexel University and is presently Associate Professor of Engineering at Widener College, Chester, PA. He studied the analog fault analysis problem during the summer of 1978 at the Avionics Laboratory at Wright-Patterson Air Force Base supported by a USAF/ASEE Summer Faculty Research Fellowship. He is also interested in the design of digital filters.

Dr. Johnson is a member of Eta Kappa Nu, Tau Beta Pi, Phi Kappa Phi, and Pi Nu Epsilon.



References

[1] "Large ATE Systems", Eval. Eng., vol. 17, no. 2, March/April 1978, pp. 28-50

[2] M.N. Ransom and R. Saeks, "Fault Isolation with Insufficient Measurements," IEE Trans. on Circuit Theory, Vol. CT-20, No. 4, pp. 416-417, July 1973.

[3] H.S.M. Chen and R. Saeks, "A Search Algorithm for the Solution of the Multifrequency Fault Diagnosis Equations", IEEE Trans. on Circuits and Systems, Vol. CAS-26, No. 7, pp. 589-594, July 1979.

[4] A.T.Johnson, Jr., "Efficient Fault Analysis in Analog Circuits", Final Report: USAF-ASEE Summer Faculty Research Program, WPAFB, Ohio, Aug. 1978.

[5] A.T.Johnson, Jr., "Efficient Fault Analysis in Linear Analog Circuits", IEEE Trans. on Circuits and Systems, Vol. CAS-26, no. 7, pp. 475-484, July 1979.

[6] N. Sen and R. Saeks, "A Measure of Testability and its Application to Test Point Selection - Theory", Proc. of the 20th Midwest Symposium on Circuits and Systems, Texas Tech. Univ., Lubbock, Texas, August 1977, pp. 576-583.

[7] N. Sen and R. Saeks, "A Measure of Testability and its Application to Test Point Selection - Computation", Proc. of AUTOTESTCON '77. pp. 212-219, Hyannis, Mass., Nov. 1977.

[8] N. Sen and R. Saeks, "Fault Diagnosis for Linear Systems via Multifrequency Measurements", IEEE Trans. on Circuits and Systems, Vol. CAS-26, no. 7, pp. 457-465, July 1979.

[9] C.A.Desoer and E.S.Kuh, Basic Circuit Theory. New York: McGraw-Hill, 1969, pp. 409-463.

[10] A.T.Johnson, Jr., and A.J.Pennington, "Network equations with coupling elements", Proc. of the 14th Midwest Symposium on Circuit Theory, pp. 2.5-1 - 2.5-7, May 1971.

[11] A.S.Householder, "A survey of some closed methods for inverting matrices", J. SIAM, Vol. 5, pp. 155-169, 1957.

[12] G. Forsythe and C.B. Moler, "Computer Solution of Linear Algebraic Systems", Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967, pp. 56-76.