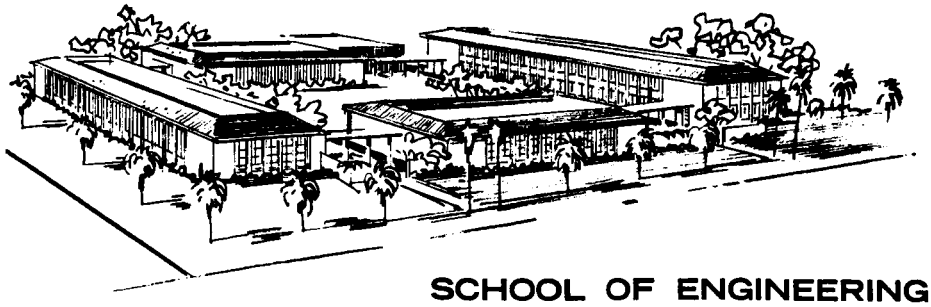
Sc



SCHOOL OF ENGINEERING

ENGINEERING AND APPLIED SCIENCE RESEARCH

UNIVERSITY OF SANTA CLARA

SANTA CLARA, CALIFORNIA AREA CODE 408-246-3200

UNIVERSITY OF SANTA CLARA

DEPARTMENT OF ELECTRICAL ENGINEERING

ANNUAL REPORT NO. 1

NASA GRANT NO. NGR-05-017-012

Richard C. Dorf	Principal Investigator
S. Park Chan	Co-investigator
Raymond B. Yarbrough	Co-investigator
Evor S. Vattuone	
Gregory J. Sauer	
Fritz M. Verduyn	
William R. Dunn	
France Rode	
Robert M. Muñoz	

July 1, 1968

TABLE OF CONTENTS

I. INTRODUCTION

The objectives of the research done under this grant have been: (1) to implement the NASAP program (Network Analysis for Systems Application Program) on an SDS-940 time shared computer, (operated by TYMSHARE INC.) subscribed to by the University of Santa Clara; (2) to extend the sensitivity analysis portion of the program; (3) to develop a nonlinear analysis portion for the program; and (4) to study the topological methods used in NASAP. Reasonable progress toward these objectives has been accomplished and is evidenced by two M.S. theses [1,2] and five related papers [3-7].

The NASAP program as implemented by the University of Santa Clara is in four parts. The original program was segmented into two portions, NASAP I and NASAP II, in order to handle problems of reasonable complexity. NASAP I accepts the coded description of the network and yields as output the T, W and V matrices [8] which are stored on a file for inputs to NASAP II or NASAP IV.

NASAP II finds the first and higher order loops of the flowgraph and produces transfer functions, immittence functions and Bode sensitivity functions in the form of ratios of polynomials.

NASAP III finds the roots of the characteristic equation, the residues at the roots, root sensitivities and weighted root sensitivities. This program is operating on an IBM 1130 and has not yet been implemented on the SDS 940. NASAP IV is a time-domain state space analysis program. It uses the output of NASAP I. By using an augmentation technique a time domain solution is obtained. This program at pre-

sent handles resistive nonlinearities of the piecewise continuous, sinusoidal and exponential varieties. At present an algorithm is being developed to handle hysteresis functions.

---

[1] E.S. Vattuone, "Sensitivity Specifications as a Design Criterion," M.S. thesis, University of Santa Clara, 1968.

[2] G.J. Sauer, "Computerized Nonlinear Circuit Analysis," M.S. thesis, University of Santa Clara, 1968.

[3] F.M. Verduyn and S.P. Chan, "Computer Evaluation of Network Performance by a Topological Flowgraph Technique," Proc. First Asilomar Conf. on Circuit and Systems, 1967, pp. 699-709.

[4] W.R. Dunn, Fr., and S.P. Chan, "On the Choice of a 'Best Tree' in the Flowgraph Analysis of Networks," Proc. First Asilomar Conf. on Circuit and Systems, 1967, pp. 710-720.

[5] R. Munoz and S.P. Chan, "Efficiency in the Use of a Computer for Network Analysis," Proc. First Asilomar Conf. on Circuit and Systems, 1967, pp. 843-855.

[6] W. R. Dunn, Jr. and S.P. Chan, "An Algorithm for Calculating the Gain of a Signal Flowgraph," Proc. First Hawaii International Conf. on System Sciences, 1968, p.

[7] F. Rode and S.P. Chan, "Computer Evaluation of Topological Formulas for Network Analysis," Proc. First Hawaii International Conf. on System Sciences, 1968, p.

[8] W.W. Happ, "Flowgraph Techniques for Closed Systems," IEEE Trans. Aerospace and Electronic Systems, v AES-2, #3, May 1966, pp. 252-264.

II THEORY

INTRODUCTION [1]

From network topology it is known that for a linear active network with dependent sources the following equation can be written.

$$\begin{bmatrix} B_f & 0 \\ 0 & Q_f \\ Y & Z \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ \vdots \\ V_e \\ I_1 \\ \vdots \\ I_e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -I_g(s) - E_e(s) \end{bmatrix} \qquad (1)$$

This equation expresses in matrix form Kirchhoff's voltage law(KVL), Kirchhoff's current law (KCL), the voltage-current relationship (VCR) and the control relationship (CR) equations. Matrix B_f is the fundamental circuit (f-circuit) matrix and Q_f is the f-cutset matrix of the linear graph corresponding to the network. Submatrices Y and X express both the VCR and the CR equations. $V_1 \ldots V_e I_1 \ldots I_e$ is the variable vector for the edge voltages and currents of the linear graph (e is the total number of elements). $I_g(s)$ and $E_e(s)$ are the independent current- and voltage-generators respectively in the network. Equation (1) can be represented by a Mason flowgraph [2], [4] the properties of which are given below.

1. Properties of the Flowgraph

Each variable in the vector $V_1 \ldots V_e I_1 \ldots I_e$ is represented by one node. The sources defined by Mason [2], [4], are the "known variables." There are e voltage nodes (top row) and e current nodes

(bottom row) in the flowgraph. The KVL matrix equation, $B_f V_e(s) = 0$, governs the relationships between the voltage nodes; and the KCL matrix equation $Q_f I_e(s) = 0$, indicates the relations between the current nodes. The VCR matrix equation gives the "vertical" node-relations in the form $V_i = Z_i I_i$ or $I_i = Y_i V_i$, where a transmittance Z_i is directed from a current node I_i to a voltage node V_i; or, similarly, Y_i is directed from V_i to I_i for the latter. The CR equation gives the generator dependence. Dependence of a dependent generator on a controlling element in the network is given by a transmittance from the controlling variable-node to the node of the controlled element. The gain of the transmittance is the "control-constant." If a variable x_j is a function of x_i: $x_j = a_{ij} x_i$, then a transmittance edge is directed from node x_i to node x_j; the gain associated with the edge is a_{ij}. The following example will clarify the procedure for obtaining the flowgraph.

2. Example of a Flowgraph.

From the equivalent network in Fig. 1, the flowgraph is to be obtained. After the linear graph has been found a tree T is chosen containing all the voltage-generators and possibly some passive elements; T = (1,4,7) where edge i is an independent voltage-generator, edges 4 and 7 are passive elements. The KVL matrix equation can now be stated:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} V_1 \\ \vdots \\ V_7 \end{bmatrix} = [0] \qquad (2)$$

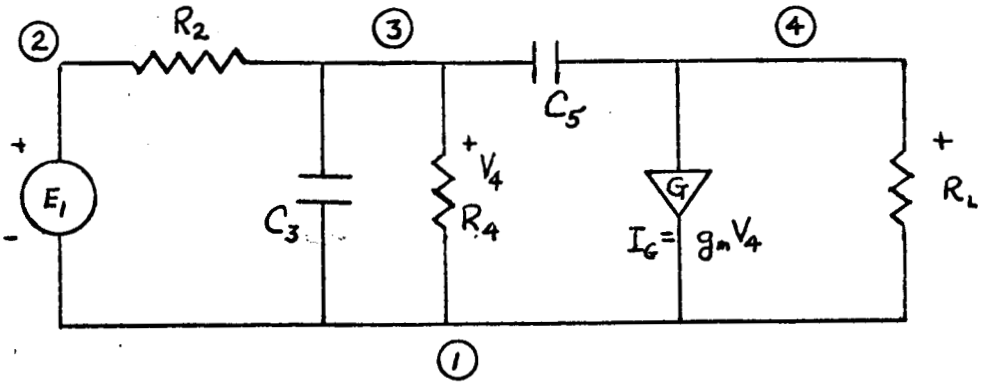


Fig. 1 Equivalent network N

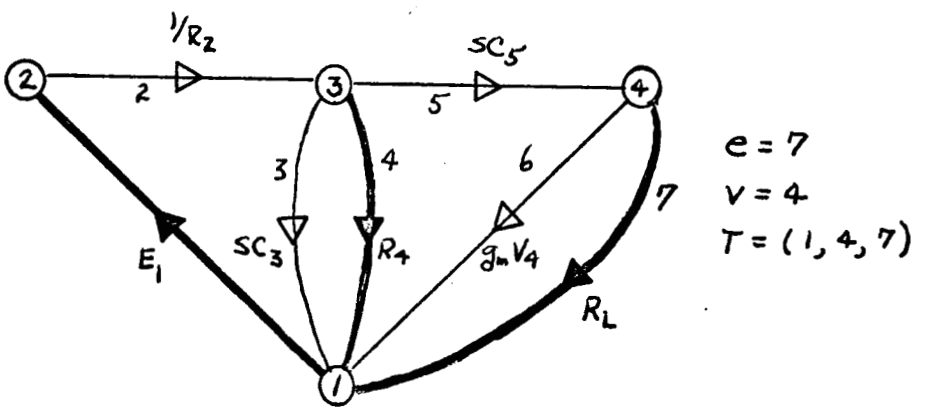


Fig. 2 Linear Graph

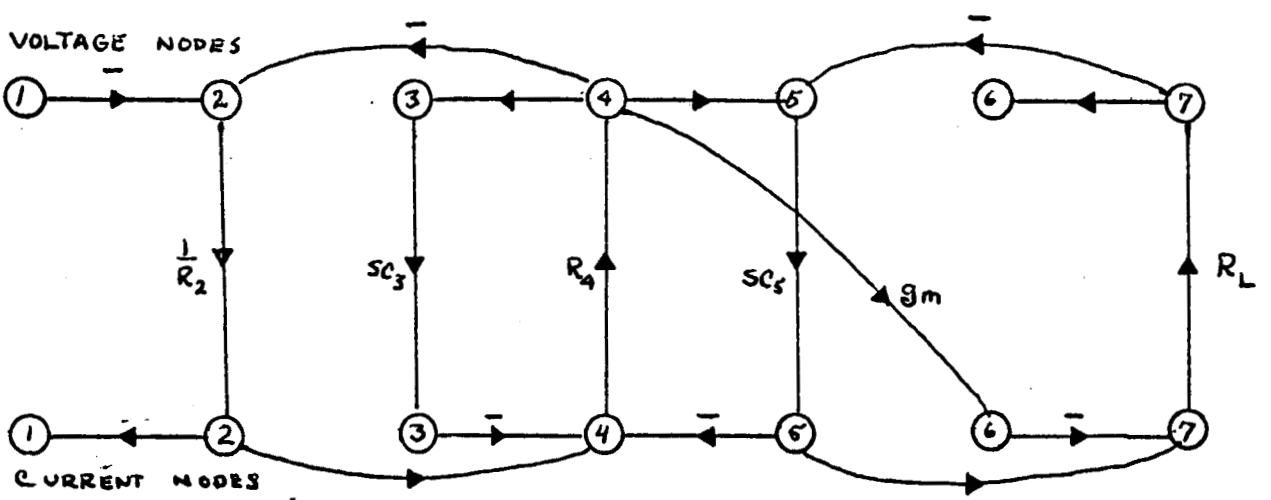


Fig. 3 Flowgraph

In the flowgraph, the chord voltage in an f-circuit is expressed in terms of the branch voltages.

Similarly the KCL matrix equation, $Q_f I_e(s) = 0$, is:

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} I_1 \\ \vdots \\ I_7 \end{bmatrix} = [0] \qquad (3)$$

where for the flowgraph the branch current in a f-cutset is expressed in terms of the chord currents. The VCR equations are:

$$\begin{aligned} V_4 &= Z_4 I_4 & I_2 &= Y_2 V_2 \\ V_7 &= Z_7 I_7 & I_3 &= Y_3 V_3 \\ & & I_5 &= Y_5 V_5 \end{aligned} \qquad (4)$$

for branches and chords respectively. The control (CR) equation yields:

$$I_6 = g_m V_4 \qquad (5)$$

The flowgraph for eq. 2,3,4 and 5 is shown in Fig. 3.

The flowgraph, and the linear graph are equivalent descriptions of the same network; in the first the "dichotomous" character is explicit, because each element is represented with two variables V_i and I_i . A solution for the unknown variables in terms of the known sources is obtained by evaluating the flowgraph. Using Mason's gain formula [2], [4], the gain G can be expressed as:

$$G = x_j / x_i \qquad (6)$$

where x_i is the independent source node and x_j is the variable node as shown in Fig. 4. A closed flowgraph [6]-[10] is formed, when a trans-

mittance j is added which is directed from node x_j to node x_i (j is used as a tagging parameter [10]) and

$$G = 1/j = \frac{-H(1)}{H(o)} \tag{7}$$

which is known as Shannon's gain formula [5],[9], where $H(1)$ is the summation of gains over all loopsets containing j, and $H(o)$ is the summation of gains for all loopsets devoid of j.

ROOT SENSITIVITY [2]

When sensitivity functions are desired in terms of a parameter k, the transmittance can be written in the form

$$G(s) = \frac{A(s) + kB(s)}{C(s) + kD(s)} \tag{8}$$

where $A(s)$ is the portion of the numerator devoid of the parameter k, $B(s)$ is the numerator coefficient of k, $C(s)$ is the portion of the denominator devoid of k and $D(s)$ is the denominator coefficient of k. The zeros of the denominator are found by determining the roots of the equation $C(s) + k\,D(s) = 0$. For a simple root at r_i, this equation may be rewritten as

$$C(s) + k\,D(s) = (s - r_i)\,N(s) = 0 \tag{9}$$

where $N(s)$ represents the product of the other roots of the equation. Thus, for a small change in k, we have

$$C(s) + (k + \Delta k)\,D(s) = (s - r_i - \Delta r_i)N^*(s) \tag{10}$$

where Δr_i denotes the change in r_i due to the change in k and $N^*(s)$ corresponds to $N(s)$ with the change in k taken into account. Evaluating at $S = r_i$

$$D(r_i)\ \Delta k = -N^*(r_i)\ \Delta r_i \tag{11}$$

as $C(r_i) + kD(r_i) = 0$.

By taking the limit as $\Delta k \to 0$, $N^*(r_i + \Delta r_i) \to N(r_i)$ and

$$\frac{dr_i}{dk} = \lim_{\Delta k \to 0} \frac{\Delta r_i}{\Delta k} = - \frac{D(r_i)}{N(r_i)} \tag{12}$$

For a root r_i of order n :

$$C(s) + k\ D(s) = (s - r_i)^n\ N(s) = 0 \tag{13}$$

Taking the $(n-1)^{st}$ derivatives with respect to s :

$$\frac{\partial^{n-1}}{\partial s^{n-1}} [C(s) + kD(s)] = n!(s - r_i)N(s) + (s - r_i)^2 F(s) \tag{14}$$

where $F(s)$ is a combination of derivatives of $N(s)$ multipled by powers of $(s - r_i)$. For an incremental change in k:

$$\frac{\partial^{n-1} D(s)}{\partial s^{n-1}}\ dk = -n!\ N(s)\ dr_i + n!\ (s-r_i)d\ N(s) + 2(s-r_i)\ F(s)dr_i + (s-r_i)^2 d\ F(s) \tag{15}$$

$C(s)$ vanishes in (15) because it is not a function of k .

Evaluating (15) at $s = r_i$

$$\frac{dr_i}{dk} = \left. \frac{-\frac{\partial^{n-1} D(s)}{\partial s^{(n-1)}}}{n!\ N(s)} \right|_{s = r_i} \qquad (16)$$

It is noted that $n!\ N(s)$ evaluated as $s = r_i$ is the n^{th} derivative of $C(s) + kD(s)$ evaluated at $s = r_i$. Therefore (12) and (16) can be written in the form:

$$\frac{dr_i}{dk} = \left. \frac{-\frac{\partial^{n-1} D(s)}{(n-1)}}{\frac{\partial^n}{\partial s^n} (C(s) + kD(s))} \right|_{s = r_i} \qquad (17)$$

where n is the order of the root at r_i. (17) may be used in any of the various forms of root sensitivities.

THE AUGMENTED STATE MATRIX EQUATION [3,4]

The canonical form of the state vector equation for linear systems is

$$\dot{\underline{x}} = \frac{d}{dt}\underline{x} = A\underline{x} + B\underline{u} , \qquad (18)$$

where $\underline{x}$ is an n-vector containing the n state variables of the system, $\underline{u}$ is an m-vector containing the forcing functions or inputs of the system, the A matrix is of order $n \times n$ and the order of the B matrix is $n \times m$. The elements of the A and B matrices are constants determined from the system parameters.

The state vector equation can be written

$$\dot{\underline{x}} = [A \quad B] \cdot \begin{bmatrix} \underline{x} \\ \underline{u} \end{bmatrix} \tag{19}$$

Then an augmented state vector $\underline{y} = [\underline{x}, \underline{u}]^T$ can be employed, and the state vector equation written in augmented form:

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\underline{u}} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{x} \\ \underline{u} \end{bmatrix} \tag{20}$$

This requires that the forcing functions be constant. Thus time varying forcing functions must be quantized for calculation, and the time intervals for calculations must be sufficiently short that no large change in any forcing function occur during the calculation time interval. The augmented A matrix A^* is defined as

$$A^* = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \tag{21}$$

which allows the augmented state matrix equation to be written in the form:

$$\dot{\underline{y}} = A^* \underline{y} \ , \tag{22}$$

which has the solution:

$$y(t+T) = e^{A^*T} \, y(t), \tag{23}$$

where T is a time interval and e^{A^*T} is defined from the Maclaurin expansion of e^x with $x = A^*T$:

$$\phi(T) = e^{A^*T} = I + A^*T + \frac{(A^*T)^2}{2!} + \frac{(A^*T)^3}{3!} + \frac{(A^*T)^4}{4!} + \ldots \quad (24)$$

$\phi(T)$ is called the transition matrix. The augmentation is necessary to perform the multiplications involved in forming the transition matrix $\phi(T)$. The augmented state matrix equation is effective only for the calculation of the state variables $\underline{x}$ and is not valid for the calculation of the inputs $\underline{u}$, which must be specified from another information at each instant of time.

THE NONLINEAR AUGMENTED STATE MATRIX EQUATION

The method used to treat nonlinear systems is a straightforward extension of the method described above. In this case the state equation is

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} + C\underline{v} \quad (25)$$

where $\underline{v}$ is the k-vector of the nonlinear outputs and C is a constant matrix of order $n \times k$. Here the outputs of the nonlinearities are treated as forcing functions. This results in an augmented state matrix equation of the form:

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\underline{u}} \\ \dot{\underline{v}} \end{bmatrix} = \begin{bmatrix} A & B & C \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{x} \\ \underline{u} \\ \underline{v} \end{bmatrix} \quad (26)$$

As the augmented A matrix contains no information concerning $\underline{u}$ or $\underline{v}$, both are considered constant during a calculation. For valid

results it is necessary that neither $\underline{u}$ nor $\underline{v}$ change significantly during any calculation time interval. Equation (26) can be solved by finding the transition matrix $\phi(T)$ for a given time interval as indicated by eq. (24). The resulting difference equation is:

$$\begin{bmatrix} \underline{x}(t+T) \\ \underline{u}(t+T) \\ \underline{v}(t+T) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \cdot \begin{bmatrix} \underline{x}(t) \\ \underline{u}(t) \\ \underline{v}(t) \end{bmatrix} \tag{27}$$

This equation yields valid information only concerning the state variables. Again the value of $\underline{u}$ must be determined by specification of input values. It is also necessary to obtain $\underline{v}(t+T)$, the nonlinear output vector, from the describing models of the nonlinear elements. The nonlinear elements are functions of the state variables $\underline{x}$, the forcing functions $\underline{u}$ and the nonlinear outputs $\underline{v}$. The j^{th} nonlinear element has a nonlinear model which is symbolically written

$$v_j = f_j(\underline{x},\underline{u},\underline{v}) \ . \tag{28}$$

The interactions between the nonlinear elements then requires some iterative procedure be used to determine $\underline{v}(t+T)$, using $\underline{x}(t+T)$, $\underline{u}(t+T)$ and $\underline{v}(t+T)$.

---

[1] F.M. Veduyn and S.P. Chan, "Computer Evaluation of Network Performance by a Topological Flowgraph Technique," Proc. First Asilomar Conf. on Circuits and Systems, 1967, pp. 699-709.

[2] E.S. Vattuone, "Sensitivity Specifications as a Design Criterion,"

M.S. Thesis, University of Santa Clara, 1968.

[3] S.P. Chan, Introductory Topological Analysis of Electrical Networks, Holt, Reinhart and Winston, 1969, ch. 7.

[4] G.J. Saur, "Computerized Nonlinear Circuit Analysis," M.S. Thesis, University of Santa Clara, 1968.

III USING NASAP FOR CIRCUIT ANALYSIS

3.1 INPUT CODING

The network topology, parameter values, inputs and desired outputs must be entered as input to NASAP I in a particular form. The form is a list or row of ten variables which describe each element. The first nine of the variables are entered in input format I2, and the tenth in input format E11.3. The preliminary procedure is as follows:

(1) Draw a directed graph of the circuit. The assigned edge (branch) direction defines both the positive sense of current through the branch (positive current is in the assigned direction), and the direction of voltage rise across the branch (positive voltage rises in the assigned direction). The input voltage or input current must be included as an edge (branch) of the graph.

(2) Number all vertices (nodes) of the graph, with the reference node as number 1.

(3) Number all edges (branches) of the graph.

(4) Choose a tree with the following priorities:

   (a) All controlled voltage sources and input voltage sources are to be included in the tree.

   (b) All controlled current sources and input current sources are to be excluded from the tree.

   (c) If the desired output is the voltage across a certain branch, that branch should be included in the tree.

   (d) If the desired output is the current through a certain branch, that branch should be excluded from the tree.

(e) If the voltage across a certain branch controls a dependent source elsewhere, the branch having the controlling voltage across it should be included in the tree.

(f) If the current through a branch controls a dependent source elsewhere, the branch containing the controlling current should be excluded from the tree.

(g) After all the above have been considered, all possible capacitors should be included in the tree.

(h) All possible inductors should be excluded from the tree.

(i) The tree is completed when there is a branch connecting every nade and no closed paths exist. The tree should be completed with resistances if possible. If this is not possible inductors can be included as a last resort.

(j) For nonlinear circuits, items (g) and (h) take precedence over items (c), (d), (e) and (f).

Once this procedure has been completed the input matrix can be generated. The input matrix consists of a ten column by e-row matrix where e is the number of edges (branches) in the network graph. Each row describes a single element by the following ten terms:

A,[IVO(J)]*: The vertex (node of origin of the directed branch** J. That is, the directed branch J is directed away from vertex (node) IVO(J).

---

* bracketed terms are the variable names in NASAP

**The voltage rise across a branch and the current through the branch are defined as positive in the assigned direction

The input format used is I2, two decimal integers corresponding to the assigned number of vertex IVO(J). If the number is less than 10, it may be written as (space) N or ON, ie -6 for vertex #6.

B,[IVT(J)]: The target vertex (node) of the directed branch J. That is, directed branch J is directed toward vertex (node) IVT(J). the input format is I2. For example if branch 3 is directed from vertex 2 to vertex 5, the coding for A and B is either: 0205 or _2_5.

C,[ICO(J)]: Indicator of type of controlling variable. If the directed branch J is controlled by a voltage ICO(J) = 00. The input format is again I2. The following examples will clarify this term:

[1] A passive tree branch has its own voltage controlled by its current so that its control is a current: ICO(J)=01

[2] A passive link has its own current controlled by its voltage, so that its control is voltage: ICO(J) = 00

[3] An active element controlled by a voltage has ICO(J) = 00

[4] An active element controlled by a current has ICO(J) = 01

[5] If the output is a current, the input is considered to be current controlled: ICO(J) = 01

[6] If the output is a voltage, the input is considered to be voltage controlled: ICO(J) = 00

D,[IDD(J)]: This term is the number of the controlling branch, or for an input source it identifies the transmittance desired. For a passive element IDD(J) = J. For an active element controlled by branch k, IDD(J) = K. The input format is I2.

For an input source where the transferrence between itself, (J), and the output taken from element E is the desired transmittance of the analysis, IDD(J) = E. The input format is I2.

E,[IEL(J)]: This is J, the given number of the branch, the format is I2.

F,[IST(J)]: This is the frequency dependence of the branch:

For resistance IST(J) = 00

For inputs IST(J) = 00

For capacitors

(a) in tree branches IST(J) = -1

(b) in tree links IST(J) = 01

For inductors

(a) in tree links IST(J) = -1

(b) in tree branches IST(J) = 01

G,[IGE(J)]: For a tree branch IGE(J) = 00

For a tree link IGE(J) = 01

H,[IJTAG(J)]: For the input source IJTAG(J) = 01

For all other branches IJTAG(J) = 00

K,[IKTAG(J)]: For sensitivity analysis, this term identifies the variable element or parameter. For the variable branch IKTAG(J) = 01, for all other branches IKTAG(J) = 00.

Z,[Z(J)]: This is the parameter value relating the branch variable (current for a link and voltage for a tree branch) to its control, exclusive of frequency dependence. The following will clarify this:

For a link resistor Z(J) = 1/R

For a tree branch resistor Z(J) = R

For a link capacitor Z(J) = C

For a tree branch capacitor Z(J) = 1/C

For a link inductor Z(J) = 1/L

For a tree branch inductor Z(J) = L

For an input source Z(J) = 1.0

For a current source i(J) controlled by a voltage V(K)

(i.e. $i_j = g_{jk}v_k$) Z(J) = g_{jk}

For a current source i(J) controlled by a current i(K)

(i.e., $i_j = a_{jk}i_k$) Z(J) = a_{jk}

For a voltage source V(J) controlled by a voltage V(K)

(i.e., $v_j = a_{jk}v_k$) Z(J) = a_{jk}.

For a voltage source V(J) controlled by a current I(k)

(i.e., $v_j = r_{jk}i_k$) Z(J) = r_{jk}.

For a nonlinear element Z(J) = 0.0

The input format for Z(J) is E 11.3 which is of the form:

.XXXE±XX

where E±XX indicates power of ten. There can be up to 10 significant digits and the decimal point may be placed anywhere in the field. If the decimal point is omitted it is placed to the left of the third last digit.

EXAMPLE

The circuit shown in fig. 1 has the network graph shown in fig. 2, and the chosen tree is shown in fig. 3.

Assuming that the desired output is the voltage across element 7, and we are interested in the sensitivity of the voltage gain to the gain of the active element (6) the following input matrix is obtained:

```
A B C D E F G H K Z
0102000701000001001.0
020300020200010000.010
030100030300010000.001
0103010404-10000001.00E+08
030400050501010000l.00E-10
040100040600010001.100
0104010707000000001.00E+03
```

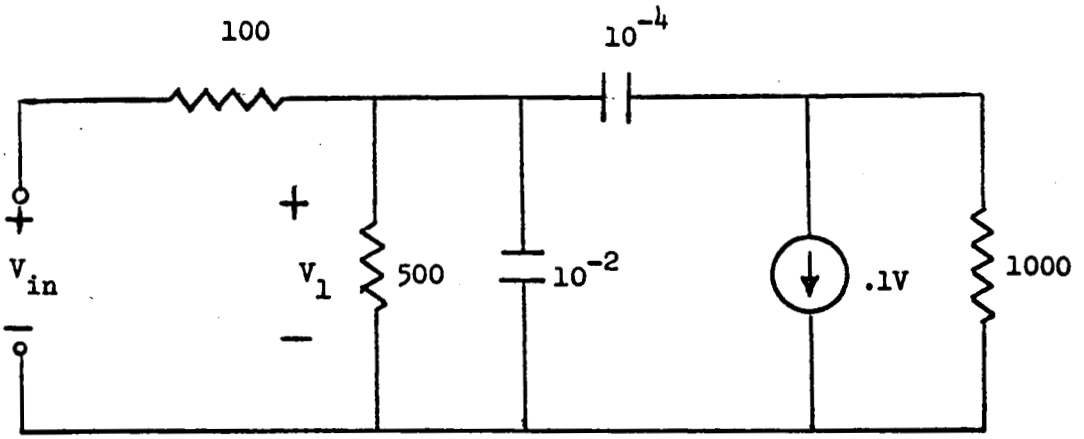


Fig. 3.1. EXAMPLE CIRCUIT

Resistance in ohms, capacitance in μF.

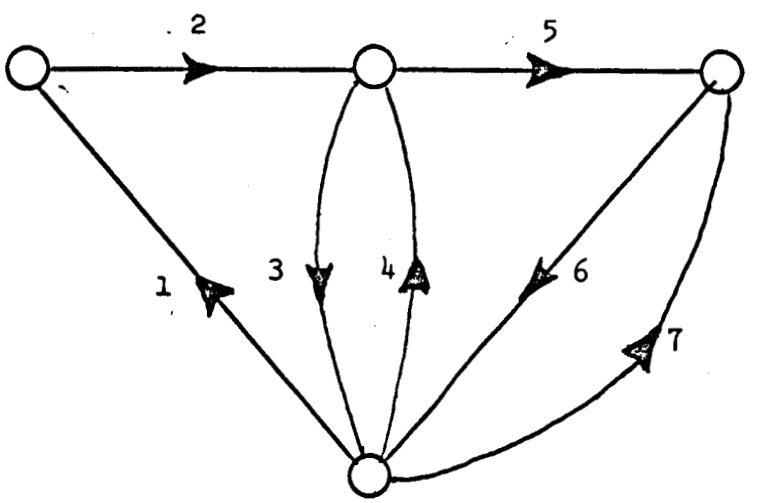


Fig. 3.2 DIRECTED NETWORK GRAPH

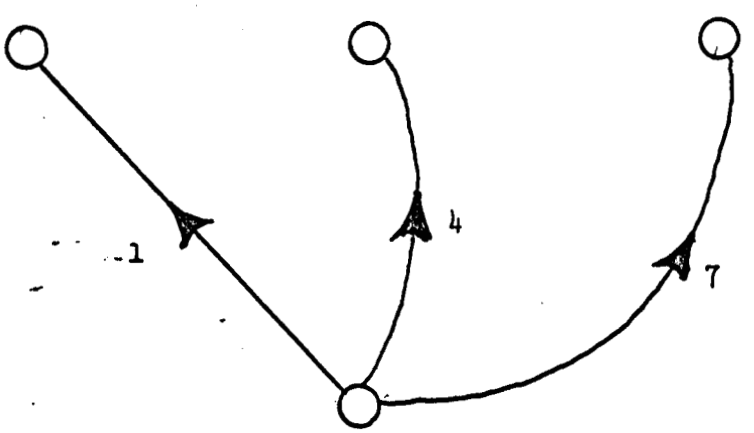


Fig. 3.3 SELECTED TREE

3.2 NASAP I

Input data for the NASAP I program is first loaded into a file named /TEST/ by means of the EDITOR operating system. Initially, this data can be typed directly into editor or read from paper tape into EDITOR. Here it is temporarily stored at which time changes in the input data may be made if necessary. Once the input data is in its final form, it is written onto the file names /TEST/ for permanent storage.

The NASAP I program, at execution time, will read the input data from the file /TEST/. After the NASAP I program has completed its calculations, it outputs information pertinent to the execution of NASAP II onto a file named /DATA/. Here the data from NASAP I is stored until NASAP II or NASAP IV is executed, at which time file /DATA/ will be read to obtain input data for NASAP II or NASAP IV.

The following is a complete list of instructions.

STEP 1: LOG-IN

The usual log-in procedure is followed.

The computer will then type a dash(-) in the far left margin when it is ready for its first instruction. This indicates that the computer is in the EXECUTIVE mode of operation.

STEP 2: LOAD DATA INTO EDITOR

a) After the dash is typed by the computer, the user then types the instruction "EDITOR" followed by a carriage return.

b) Once in EDITOR, the computer types back an asterisk in the far left margin and then waits for the next instruction;

this will be to read the teletype.

c) The input data can be either typed in or read in by the paper tape reader. The paper tape reader is activated by turning the console switch on the left of the teletype to START. At the end of the input data, depress the control key and the alphabetic D key simultaneously; this will indicate the end of the input data. The data is now loaded into EDITOR where it will be stored temporarily.

-EDITOR cr

*READ TELETYPE cr

Type in data* or read in data from the paper tape reader. The format for each row of the input data is 9I2,E11.3. A 0 (zero) in column 2 of the row after the last row of data will indicate to the NASAP I program that there is no more data. Push the control key and the D key simultaneously to indicate the end of the input text to EDITOR. The computer then responds with an asterisk at which time the computer is ready for the next instruction which will be to write this data onto file /TEXT/.

*WRITE/TEXT/ cr

OLD FILE cr

xxxx WORDS

*

*

---

*For Data Format see section 3.1

Now depress the escape key twice to return to the EXECUTIVE mode. This will be indicated by a dash typed in the far left margin.

STEP 3: LOAD AND EXECUTE NASAP I

a) In the EXECUTIVE mode, access to the FORTRAN II operating system can be obtained by typing "FOS". The computer will then ask for the binary file which is to be executed and the name of the binary file which contains the library subroutines which are to be used by the main program. After loading, execution is started by typing a ;G (semi-colon, G). cr indicates carriage return.

```
-FOS
LOAD MAIN PROGRAM
FROM FILE /BcTESTEVORI/cr

LOAD SUBPROGRAMS
FROM FILE "LIB" cr
LOADING COMPLETE
xxxxWORDS OF STORAGE UNUSED
;G
```

(program is executed)

Note that NASAP I is on a file named /B^{c}TESTEVORI/. The B^{c} refers to the control key and the B key simultaneously depressed. This allows other users of the same account number to use this program. Note also that "LIB" is the name of the binary file which contains the library subroutines used by the main program. After execution, the instruction "GO TO PART 2 OF NASAP" is printed. Now depress the escape key twice to return to the EXECUTIVE mode.

3.3 NASAP II

STEP 4: LOAD AND EXECUTE NASAP II

In the EXECUTIVE mode, the following set of instructions must be given to execute NASAP II:

```
-FOS
LOAD MAIN PROGRAM
FROM FILE /BTE2/ cr

LOAD SUBPROGRAMS
FROM FILE "LIB"
LOADING COMPLETE
xxxxWORDS OF STORAGE UNUSED
;G
PRINTOUT YES OR NO
YES (or no)
```

YES requests that all loops in terms of nodes be printed out, this provides either a symbolic solution or a program check.

(program is executed)

Note: the NASAP II program is on a binary file named /BTE2/. After execution, return to the EXECUTIVE mode and log out:

```
-LOG
TIME USED xx:xx
```

3.3 NASAP II OUTPUTS

I Loops

These are given in terms of the node of the closed flowgraph. The loop number is printed out on the far left and its associated nodes are printed out consecutively to the right.

II Topology Equation (as a function of one variable)

1) Topology equations devoid of 1/G(s) [$\Delta(\overline{G},s)$]

2) Topology equation containing 1/G(s) [$\Delta(G',s)$]

III Transfer function or immittance function.

1) Numerator

2) Denominator

IV Bode-Sensitivity function

1) Numerator

2) Denominator

V Topology equation (as a function of two variables)

1) That portion containing imaginary generator but devoid of sensitivity generator. $G(s)H(G',\overline{k},s)$

2) That portion containing imaginary generator and sensitivity generator, $KG(s)H(G',k',s)$

3) That portion devoid of imaginary generator and devoid of sensitivity generator. $H(\overline{G},\overline{k},s)$

4) That portion devoid of imaginary generator but containing sensitivity generator. $KH(G,k',s)$

3.4 NASAP III

NASAP III has not yet been implemented on the SDS 940 computer. It is presently operating on an IBM 1130 computer, and the output from NASAP II must be transferred to cards.

Input to NASAP III

The input to the NASAP III program consists of the arrays of a's, b's, c's, d's, A's, B's, C's and the order of each of the functions' numerator and denominator. The input data consists of 17 I.B.M. data cards which are listed as follows:

Card #1 ORDERS OF:

1) transfer function numerator
2) transfer function denominator
3) root sensitivity function numerator (for first parameter)
4) root sensitivity function denominator (for first parameter)
5) root sensitivity function numerator (for second parameter, if any)
6) root sensitivity function denominator (for second parameter, if any)

Note: The input format for each of these variables is I3.

Card #2 NUMERATOR OF TRANSFER FUNCTION

$a_{1,j}$ in descending orders of s (coefficients multiplying K_1)

Card #3 NUMERATOR OF TRANSFER FUNCTION

$c_{1,j}$ terms in descending orders of s (coefficients multiplying K_2)

Card #4 NUMERATOR OF TRANSFER FUNCTION

$b_{1,j}$ terms in descending orders of s (coefficients multiplying just s)

Card #5 DENOMINATOR OF TRANSFER FUNCTION

$a_{s,k}$ terms in descending orders of s (coefficients multiplying K_1)

Card #6 DENOMINATOR OF TRANSFER FUNCTION

$c_{2,k}$ terms indescending orders of s (coefficients multiplying K_2)

Card #7 DENOMINATOR OF TRANSFER FUNCTION

$b_{2,k}$ terms in descending orders of s (coefficients multiplying just s)

Card #8 PARAMETER VALUES

1) initial value of first parameter K_1
2) value for which K_1 is to be incremented

3) final value of K_1

4) initial value of second parameter K_2

5) value for which K_2 is to be incremented

6) final value of K_2

Card #9 SENSITIVITY FUNCTION NUMERATOR

$A_{1,j}$ terms in descending orders of s (coefficients multiplying K_1)

Card #10 SENSITIVITY FUNCTION NUMERATOR

$C_{1,j}$ terms in descending orders of s (coefficients multiplying both K_1 and K_2

Card #11 SENSITIVITY FUNCTION NUMERATOR

$D_{1,j}$ terms in descending orders of s (coefficients multiplying just s)

Cardd #12 SENSITIVITY FUNCTION DENOMINATOR

$ka_{s,k}$ terms indescending orders of s (coefficients multiplying K_1)

Card #13 SENSITIVITY FUNCTION DENOMINATOR

$kc_{2,k}$ terms in descending orders of s (coefficients multiplying K_2)

Card #14 SENSITIVITY FUNCTION DENOMINATOR

$kb_{2,k}$ terms in descending orders of s (coefficients multiplying just s)

Card #15 SENSITIVITY FUNCTION NUMERATOR (SECOND PARAMETER)

$B_{2,j}$ terms in descending orders of s (coefficients multiplying K_2)

Card #16 SENSITIVITY FUNCTION NUMERATOR (SECOND PARAMETER)

$C_{2,j}$ terms in descending orders of s (coefficients multiplying K_1 and K_2)

Card # 17 SENSITIVITY FUNCTION NUMERATOR (SECOND PARAMETER)

$D_{2,j}$ terms in descending orders of s (coefficients multiplying just s)

The coefficients of the denominator of the second sensitivity function are not read in as input data since the denominator is the same as the denominator for the first sensitivity function.

OUTPUT TO NASAP III

The output to NASAP III consists of:

1) The value of the parameter (s) is printed out
2) The roots of the characteristic equation (poles of transfer function) are printed out
   a) If the roots are complex conjugate, then the damping ratio and the natural frequency of these roots are printed out.
3) The magnitude(s) of the root sensitivity(s) at the roots printed out in 1) is printed out.
4) The residue of the transfer function at these roots is printed out.
5) The weighted root sensitivity(s) (second approximation) is printed out.
6) The total weighted root sensitivity is printed out.
7) The angle(s) of the root sensitivity(s) is printed out.

3.5 NASAP IV

NASAP IV is a time domain circuit analysis program which allows

certain nonlinearities to be included. This program first reads the output of NASAP I stored in the file /DATA/. Then it is necessary to provide initial conditions for all energy storage devices, descriptions of the nonlinear elements, and information concerning the sources or forcing functions is required. Finally the time increment, the number of time increments for which calculations are desired and the number of time increments for which output is desired are requested.

At each time interval requested the values of the state variables and the nonlinear element outputs are printed out. Optional outputs are explained in the following.

3.5.1 USE OF NASAP IV

At the termination of NASAP I the computer prints the instruction:

GO TO PART II

It is then necessary to return to the executive mode by pushing the escape key twice. The operator then requests fortran:

Underlining indicates computer response. cr indicates carriage return.

```
-FOS
LOAD MAIN PROGRAM
FROM FILE /PREC/cr
LOADING COMPLETE
xxxxx WORDS OF STORAGE UNUSED
```

The operator then begins the program:

```
;G
```

LPOUT = 1 for printout of the T,A* and ϕ matrices (program testing)

0 otherwise

JQUEST = 1 for printout of loops, nodes, loop gains, INODE AND RPROD arrays (program testing)

0 otherwise

3.5.2 INITIAL CONDITIONS

AMPSIC(J) = This requests the initial current in inductor numbered in inductor numbered J. Input format E10.6.

VOLTSIC(J) = This requests the initial voltage across capacitor numbered J. Input format E 10.6

3.5.3 NONLINEAR DESCRIBING EQUATIONS

TYPE (J) = J is the number assigned to the nonlinear element. The responses may be:

0,N if the nonlinearity is piecewise continuous, N indicates the number of segments or sections, the format for N is I2.

1 cr for a sinusoidal nonlinearity

2 cr for an exponential nonlinearity

3 cr for hysteresis (not yet implemented)

PIECEWISE CONTINUOUS NONLINEARITY

After a piecewise continuous nonlinearity has been indicated the computer requests information on each of the N sections. Each section must be described by a cubic equation of the form:

$x(k) = DMP + CMP \cdot Y^1 + BMP \cdot Y^2 + AMP \cdot Y^3$ where DMP, CMP, BMP and AMP are constants supplied by the operator, x is the output variable y is the input variable. The greatest value of y in each section is designated by the term TMP. The operator must enter the piecewise

continuous information for each of the N segments in the order TMP, DMP, CMP, BMP, AMP. The computer requests this information by typing the number of the section:

N = 1 IMP(J,1), DMP(J,1), CMP(J,1), BMP(J,1), AMP(J,1) cr

N = 2 TMP(J,2), DMP(J,2), CMP(J,2), BMP(J,2), AMP(J,2) cr

⋮

n = N TMP(J,N), DMP(J,N), CMP(J,N), BMP(J,N), AMP(J,N) cr

The input format is 5E10.6. The value of TMP(J,N) must be greater than any expected value of y(J).

SINUSOIDAL NONLINEARITY

A sinusoidal nonlinearity is indicated by a response of 1 to the computer's request: TYPE J = 1. The sinusoidal nonlinearity is described by four parameters:

$$x = DMP\ SIN((6.283Y/TMP) + CMP) + BMP$$

For the sinusoidal nonlinearity the computer requests the parameter values of typing N = 1:

N = 1 TMP(J,1), DMP(J,1), CMP(J,1), BMP(J,1) cr

The format is 5E10.6

EXPONENTIAL NONLINEARITY

An exponential nonlinearity is indicated by a response of 2 to the computer's request: Type J = 2 . In this case the nonlinearity is described by four parameters:

$x = DMP + CMP\exp(TMP(Y) + BMP)$

The computer requests information in the same form as the sinusoidal nonlinearity:

N = 1 TMP(J,1), DMP(J,1), CMP(J,1), BMP(J,1) cr

The format is 5E10.6

HYSTERESIS*

The hysteresis nonlinearity is indicated by typing a 3 in response to the computer request of type: Type J = e. The operator must then supply the four pairs of coordinates (y,x) identifying the four corners of the hysteresis loop. The parameters are Y = DMP, x = CMP for this nonlinearity. The computer requests this information for four quadrants:

QUAD 1: DMP(J,1), CMP(J,1) cr

QUAD 2: DMP(J,2), CMP(J,2) cr

QUAD 3: DMP(J3,), CMP(J,3) cr

QUAD 4: DMP(J,4), CMP(J,4) cr

The format is E 10.6. This function only allows straight-line approximation to the hysteresis.

*This portion has not yet been implemented.

3.5.4 SOURCE DESCRIPTION

The form of the source information is similar to that of the nonlinearities. The sources types are described by three parameters, and the specifications of the sources by an additional entry. The

three parameters identifying the type are JTYPE (K), NSECT(K) and TPER(K):

TYPE (J) = JTYPE(J), NSECT(J), TPER(J) cr

JTYPE(J): Format I1

0 for piecewise continuous input (includes step, ramp and rectangular inputs)

1 for sinusoid

2 for exponential

NSECT(J):

The number of sections per period of the source. This is 1 for JTYPE = 1 or 2. Format is I2.

TPER(J):

The period of the source function in seconds. If TPER is read in as 0.0 or omitted it is read as 1.0E + 75. For sinusoidal sources TPER can be neglected as it is read in as TMP. Format is E10.6.

The computer will immediately request further information on each source term by typing:

N = 1: TMP, DMP, CMP, BMP, AMP

The operator response of TMP, DMP, CMP, BMP, AMP, is the same for the corresponding nonlinearities (sect 3.5.3) except that time t (seconds) is substituted for the input y.

3.5.5 OUTPUT PARAMETERS

The last inputs necessary for the program are T, NUMOUT and NDIV. T is the time increment which is used in calculating the transition matrix (see ch. 2). NUMOUT is the number of increments for which the operator wants the computer to calculate

the system states. NDIV indicates the fraction of calculations the operator wants printed out, i.e., if at every tenth time increment the output is desired NDIV = 10.

T = (time in seconds, format E 10.4)
NMOUT = (number of calculation increments, format I6)
NDIV = (fraction of calculations to be printed, format I6.

3.5.6 RECOMMENDATIONS FOR FURTHER DEVELOPMENT

The criticisms of this program (NASAP IV) can be seen to be the following:

1. There is no provision for a state variable to be nonlinear. It is then necessary to make provisions, either in terms of defining new variables or in further development of the programming to take care of this difficulty.
2. Output printout is available only for the state variables and nonlinear outputs. This requires further programming to:
   (a) Define desired outputs in terms of the state variables and the nonlinear outputs, and (b) generate the desired outputs.
3. The hysteresis portion of the program is not yet implemented. At the present time, it appears that suitable models for square-loop magnetic cores can be implemented by using two parallel nonlinearities in the form of switching functions. The subprogram for hysteresis as included in the report is under development however and should be available in the next few months.