# Computer Methods of Network Analysis

FRANKLIN H. BRANIN, JR., SENIOR MEMBER, IEEE

*Abstract*—**In this tutorial paper, the influence of the computer not only on the *modus operandi* of circuit design, but also on network theory itself, is discussed. The topological properties of linear graphs are reviewed and a matrix-topological formulation of the network problem is described. In addition to the classical mesh, node, and cutset methods, a mixed method of analysis is described which is applicable to dc, ac, and transient problems.**

**Numerical methods of solving linear and nonlinear dc network problems are discussed and a new approach to ac analysis, using the mixed method and a numerical solution of the matrix eigenvalue problem, is described. The extension of this method to the transient analysis of linear networks is also explained. Finally, the problem of instability in the numerical integration of differential equations is discussed and several means of solving the problem are outlined.**

## The Impact of Computers

As POINTED OUT in a review article by Kuo,[1] and more recently by Christiansen,[2] the computer is exerting a profound influence on the *modus operandi* of circuit design. With the aid of existing computer programs, the circuit designer may carry out "numerical experiments" which would be difficult or even impossible to duplicate with actual hardware on the bench. Often, he may be able to do such experiments faster and cheaper on the computer. He may analyze circuits for which certain components have not yet been manufactured—or even invented— and he may vary parameters, either singly or in combinations, seeking for an optimum design. Moreover, he may do all this without the risk of burning up expensive or hard-to-get components even if he inadvertently overdrives the circuit with a megavolt input signal.

Thus, the digital computer provides circuit designers with an analytical tool of great power and versatility. Generally speaking, the computer can be trusted to produce reliable solutions to an arbitrary network problem, except when the problem involves pathological numerical difficulties or when the method of analysis has been poorly chosen. It is the user's responsibility, however, to pose a problem that accurately represents the circuit which the designer has in mind, since the computer does only what it is explicitly instructed to do and has no way of second-guessing the designer's real intent. It is obvious therefore that high-fidelity network models for diodes and transistors will be needed.[3] We shall not attempt to discuss the modeling of devices in this paper, but their importance cannot be ignored.

In addition to its obvious impact on the methodology of circuit design, the computer is also having a more subtle influence on network theory itself. For example, because matrix notation is so convenient in programming a digital computer, many existing network analysis programs are based on a matrix formulation of the network problem. This, together with the inherent elegance of the matrix approach, is helping to promote its acceptance among network theorists.

In another direction, computers are helping to hasten the recognition of the inherently broad scope of network theory by drawing attention to the fact that a network analysis program is theoretically capable of converting a digital computer into a veritable analog computer of great versatility and power. While it may not be practical to write a single program to do this, at least two programs have been written for solving nonelectrical problems by network analytical methods: DYANA,[4] which handles either electrical or mechanical problems, and STRESS,[5] which is based on a network approach to structural analysis.[6]

Computers are also influencing network theory by demanding methods of analysis adapted to the solution of computer-sized problems. As the engineer will appreciate, maximum "power transfer" from the computer to his network problem requires a method which properly matches the computer to the problem. Traditional methods for "hand solution" of networks are not necessarily best for use on a computer with networks of much greater size. The Laplace transform techniques fit this category and should at least be supplemented, if not supplanted, by numerical methods better adapted to the computer. One such method,[7] described below, gives essentially all of the information obtainable by Laplace transforms.

The objective of this paper is to highlight the importance of keeping the computer and its peculiarities in mind when developing the theory on which to base network analysis programs. Since matrix notation is the tool *par excellence* for describing the network problem in a form suitable for programming, we shall use a matrix approach throughout the following discussion of various aspects of dc, ac, and transient analysis of networks on a digital computer.

## Matrix Formulation of the Network Problem

It is an interesting historical coincidence that just prior to the advent of the high-speed electronic computer, the pioneering work of Gabriel Kron[8] demonstrated convincingly the superior organizational powers of the matrix-tensor notation in network theory. Kron emphasized not only the conceptual elegance of this notation but also its virtually automatic way of handling the tedious bookkeeping chores of network analysis. He thereby laid some of the logical and procedural foundations necessary for programming a digital computer to analyze networks—that is, both

Manuscript received February 1, 1967; revised August 3, 1967. The unrevised version of this paper was published in the *IEEE International Convention Record*, vol. 15, pt. 5, pp. 45–63, 1967. A condensed version appeared as "Computer-aided design: Part 4, analyzing circuits by the numbers," *Electronics*, vol. 40, pp. 88–103, January 9, 1967.

The author is with the Systems Development Division Laboratory, IBM Corporation, Kingston, N. Y.

to compile and to solve the network equations automatically.

The following matrix formulation of the network problem[9],[10] is a direct outgrowth of Kron's work[8],[11] and that of Roth[12]-[14] who provided an algebraic-topological explanation of the network problem as an abstraction. This formulation has been used at least in part by such programs as TAP,[15],[16] NET-1,[17] and ECAP.[18] Kron's work has also been used as the basis for much of the computer analysis of networks encountered in the electrical power industry.[19]-[21]

From a physical point of view, the network problem is concerned with predicting the behavior of a system of interconnected elements in terms of the element characteristics and the manner in which these elements are interconnected. Viewed as a mathematical entity, the network problem involves the properties of an underlying *topological* structure, called a linear graph, and a superimposed *algebraic* structure consisting of the interrelations between certain quantities associated with the nodes, branches, and meshes of the graph.

Obviously, the very act of interconnecting the network elements introduces certain constraints on the physical variables of the network problem and this gives rise to part of the bookkeeping chore. But the bookkeeping can be systematized in terms of matrices which describe the connectivity properties of a linear graph. Here is where topology is introduced, albeit in a very rudimentary form. Finally, the physical variables of the network problem correspond to the quantities which enter into the algebraic structure referred to above. The matrix approach nicely correlates these two points of view and has the additional merit of identifying precisely the separate roles of the element characteristics and of the interconnections in determining the overall behavior of the network.

### *The ABCD's of Network Topology*

There are four topological matrices relating to a directed (or oriented) linear graph which are useful in network theory. We shall describe them briefly. (More detailed descriptions are given elsewhere.[6]-[10])

Consider the graph of Fig. 1(a) for which the branch-node connections may be tabulated as follows:

| Branch | Initial Node | Final Node |
|---|---|---|
| 1 | *A* | *B* |
| 2 | *B* | *C* |
| 3 | *B* | *E* |
| 4 | *D* | *E* |
| 5 | *D* | *B* |
| 6 | *E* | *C* |
| 7 | *C* | *C* |

This connection table, which is easily obtained from input data describing the network configuration, is a convenient starting point for compiling the network equations.

An alternative way of storing this connectivity information in a computer is in matrix form as displayed in Fig. 1(b), where the columns show which branches are connected to each node and the polarity of these connections. The way in which this $\bar{A}$ matrix is obtained from the connection table is obvious by inspection. (Branch 7, being short-circuited, requires both a $+1$ and a $-1$ entry in column $C$ so that both entries cancel.)

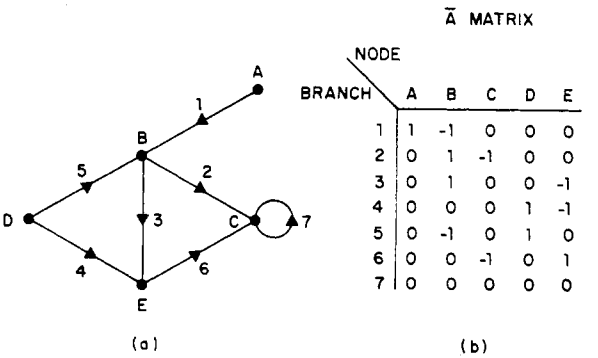


Fig. 1. Linear graph.

Since each row of the $\bar{A}$ matrix contains both a $+1$ and a $-1$ as its nonzero entries, its columns are linearly dependent—that is, their sum is zero so that any column is equal to the negative sum of all the other columns. Hence, any one column may be deleted since it contains redundant information. We choose to delete the column corresponding to node $E$ in Fig. 1 and regard this node as the datum or ground node. The resulting $A$ matrix, called the branch-node matrix, is shown in Fig. 2(b).

If we choose a tree in the graph—for example, branches 1, 2, 3, and 4 shown as solid lines in Fig. 2(a)—and classify the remaining branches as links, we may partition the $A$ matrix into the submatrices $A_T$ and $A_L$ referring to tree-branches and links, respectively. (An algorithm for performing a tree-link sort on the computer is given in Appendix II of Branin.[16] The sequence of branches may need to be rearranged before partitioning the $A$ matrix, but this is a simple task.)

It is easily shown that the submatrix $A_T$ is square and has an inverse which we may determine directly from the graph of the tree in terms of the node-to-datum path matrix $B_T$ for the tree.[10]

For the tree shown in Fig. 3(a) the columns of $B_T$, shown in Fig. 3(b), indicate which branches are included in the paths from each node to the datum node. The desired relation between $B_T$ and the inverse of $A_T$ is

$$A_T^{-1} = B_T^t \tag{1}$$

where $B_T^t$ is the transpose of the $B_T$ matrix. (An algorithm for determining the $B_T$ matrix from the $\bar{A}$ matrix is given in Appendix III of Branin.[16])

The next topological matrix of interest is the branch-mesh or circuit matrix, $C$, shown in Fig. 4(b). The columns of this matrix show which branches are included in each mesh.

If we adopt the convention that each link, together with its contiguous tree branches, defines a basic mesh whose orientation coincides with that of its defining link, then each basic mesh will contain only one link, oriented posi-

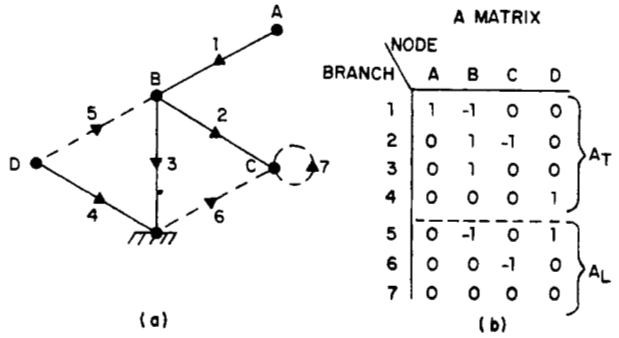


Fig. 2. Tree-link structure of graph.

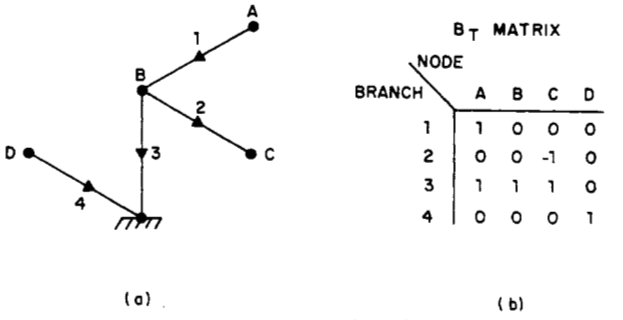


Fig. 3. Tree of graph.

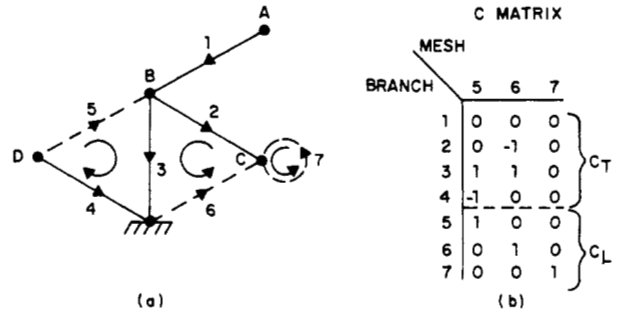


Fig. 4. Basic meshes of graph.

tively, as shown in Fig. 4(a). Consequently, the submatrix $C_L$, referring to the links, is a unit matrix as shown in Fig. 4(b) and it need not be stored explicitly in the computer. Only the submatrix $C_T$, which gives the path-in-tree for each basic mesh, is needed and this may be obtained as follows.

It is a standard theorem that, for any linear graph, the branch-node and branch-mesh matrices obey the fundamental relations,

$$A^t C = 0 \quad \text{and/or} \quad C^t A = 0 \tag{2}$$

where the product is a null matrix. As a consequence, we may write the first of these expressions in partitioned form,

$$\begin{bmatrix} A_T^t A_L^t \end{bmatrix} \begin{bmatrix} C_T \\ U_L \end{bmatrix} = A_T^t C_T + A_L^t = 0 \tag{3}$$

where the unit matrix $U_L$ replaces $C_L$. We may solve this equation for $C_T$ by inverting $A_T^t$ and using (1) to obtain the relation[10]

$$C_T = -B_T A_L^t. \tag{4}$$

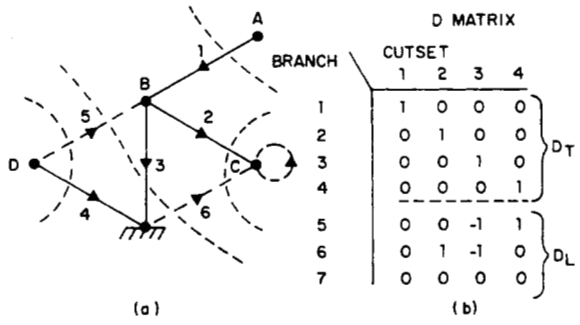


Fig. 5. Basic cutsets of graph.

The final topological matrix we need is the cutset matrix $D$. Just as each link, together with a certain set of tree branches, defines a basic mesh, so each tree branch together with a certain set of links defines a basic cutset. This is shown in Fig. 5(a).

The columns of the $D$ matrix show which branches are included in each cutset. With each tree branch positively oriented in its corresponding basic cutset, the submatrix $D_T$, referring to the tree branches, is a unit matrix while the submatrix $D_L$, referring to the links, turns out to be the negative transpose of $C_T$. (See Branin,[10] for example.) This is shown in Fig. 5(b). As a result, it follows that

$$D = \begin{bmatrix} D_T \\ D_L \end{bmatrix} = \begin{bmatrix} U_T \\ -C_T^t \end{bmatrix} = \begin{bmatrix} A_T \\ A_L \end{bmatrix} B_T^t = A B_T^t \tag{5}$$

so that the cutset matrix $D$ is readily computed in terms of $A$, $B_T$, and/or $C_T$. Except for pointing out that

$$D^t C = 0 \quad \text{and/or} \quad C^t D = 0 \tag{6}$$

this completes our description of topological matrices. Their use in performing the bookkeeping tasks in network analysis will be explained in the next section.

*The Electrical Network Laws*

Consider as a typical branch in an electrical network the configuration shown in Fig. 6.

Here, the $r$th branch is shown consisting of an impedance (or admittance) element $Z_{rr}$ (or $Y_{rr}$) in series with a voltage source $E_r$ and in parallel with a current source $I_r$, either of which sources may have any value. Thus, there are three distinct voltage and current variables to identify in this branch. As shown in the drawing, branch current $i_r$ enters the branch at its initial node but the element current $J_r$ actually passing through $Z_{rr}$ is, according to the sign conventions depicted,

$$J_r = I_r + i_r. \tag{7}$$

Similarly, although the branch voltage, measured from initial to final node, is $e_r$, the element voltage $V_r$ actually appearing across $Z_{rr}$ is

$$V_r = E_r + e_r \tag{8}$$

with the sign conventions taken as shown.

Clearly, since $E_r$ and $I_r$ may be nonzero, Ohm's law for this branch must be written in terms of the element voltage and element current variables $V_r$ and $J_r$. Accordingly, we

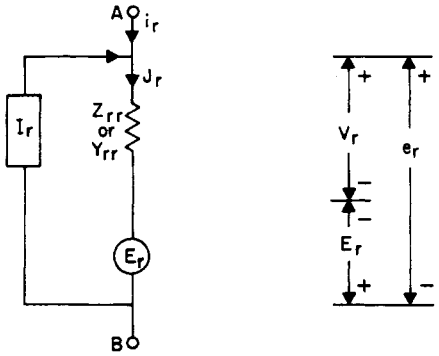


Fig. 6. The $r$th branch of an electrical network.

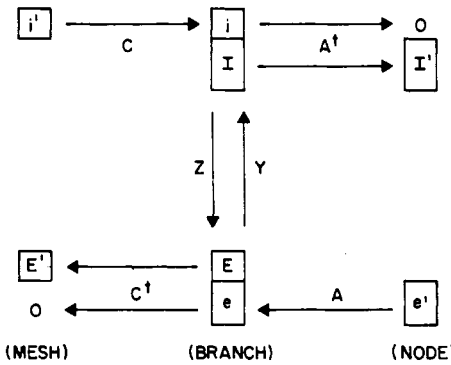


Fig. 7. Algebraic structure of the network problem.

may express Ohm's law for the entire network as the matrix equations

$$V = ZJ \tag{9}$$

and/or

$$J = YV \tag{10}$$

where the vectors $V$ and $J$ consist of the entire set of element voltage and element current variables for the network and where the admittance matrix $Y$ is the inverse of the impedance matrix $Z$. The diagonal terms of $Z$ (or $Y$) are the self-impedances (or self-admittances) of each branch, whereas the off-diagonal terms are the transimpedances (or transadmittances) between pairs of branches. If corresponding off-diagonal terms are equal, they correspond to mutual impedances or admittances. Active devices, or so-called "dependent sources," may be represented by off-diagonal terms in the $Z$ or $Y$ matrix.

In order to account for the constraints on the branch voltages $e$ and branch currents $i$ imposed by the interconnections, we invoke Kirchhoff's voltage and current laws. In their original form, these laws correspond to the matrix relations

$$C^t e = 0 \tag{11}$$

and

$$A^t i = 0 \tag{12}$$

where the topological matrix $C^t$ acts as an operator which sums all the branch voltages around each basic mesh while the matrix $A^t$ sums all the branch currents leaving each node.

Although we may not always be aware of the fact, we are actually using alternative and mathematically equivalent forms of Kirchhoff's voltage and current laws when we say "the branch voltages are a linear combination of the node voltages" and "the branch currents are a linear combination of the mesh currents." These two statements correspond to the matrix equations,

$$e = Ae' \tag{13}$$

and

$$i = Ci' \tag{14}$$

where $e'$ is the vector of node-to-datum voltages and $i'$ is the vector of mesh currents. Here, as in (11) and (12), the topological matrices automatically do the bookkeeping for us. Incidentally, it is easy to derive (11) and (12) from (13) and (14) by premultiplying the latter by $C^t$ and $A^t$, respectively, and then invoking (2). The reverse derivations can also be made, proving that both forms of Kirchhoff's laws are equivalent.

The basic interrelations between the network variables and the topological matrices may be summarized by a transformation diagram, given by Roth,[12] as shown in Fig. 7. This diagram, whose relation to (9)–(13) is evident, characterizes the algebraic structure associated with a linear graph. The only additional relations are $C^tE=E'$, which defines the equivalent mesh voltage sources $E'$, and $A^tI=I'$, which defines the equivalent nodal current sources $I'$.

By a proper interpretation of this algebraic structure, it can be shown that a wide variety of nonelectrical systems—e.g., mechanical and structural systems—fit into this algebraic pattern and so can be treated as network problems. Moreover, by extending this algebraic structure to characterize topological structures containing surface and volume elements as well as points and lines, a direct correspondence with the operational structure of the vector calculus emerges.[22] As a result, the use of network analogies for a wide class of partial differential equations—including Maxwell's equations—can be justified. The exploration and exploitation of this interesting topic is one of the broader aspects of network theory whose development the computer is helping to stimulate.

### *The Electrical Network Problem*

A formal statement of the electrical network problem may be expressed as follows: "given a network whose configuration determines the topological matrices $A$ and $C$, given its impedance matrix $Z$ (or admittance matrix $Y$), and given the voltage source vector $E$ and current source vector $I$, find the branch voltages $e$ and branch currents $i$ such that (7)–(12) hold true."

We shall discuss four distinct ways of attacking this problem. In each case, it will be necessary to introduce auxiliary variables such as the node voltages $e'$ or mesh currents $i'$ appearing in (13) and (14). In each case, it will also be neces-

sary to invoke *both* of Kirchhoff's laws and, since each of these laws has two equivalent forms, the use of one form is tantamount to invoking the other.

*The Mesh Method:* Starting with the impedance form of Ohm's law given in (9), we introduce (7) and (8) and rearrange terms to obtain the expression

$$(E - ZI) + e = Zi. \tag{15}$$

If we premultiply by $C^t$ and take into account Kirchhoff's voltage law in the form of (11), we find that

$$C^t(E - ZI) = C^tZi. \tag{16}$$

We may then incorporate Kirchhoff's current law by using (14) and writing the matrix equivalent of the mesh equations

$$C^t(E - ZI) = C^tZCi' \tag{17}$$

where $C^tZC$ is the mesh impedance matrix.

If we solve this system of equations for the mesh currents $i'$—or write the formal solution for $i'$ as

$$i' = (C^tZC)^{-1}C^t(E - ZI) \tag{18}$$

where $(C^tZC)^{-1}$ is the mesh solution matrix—we may compute the branch currents $i$ by using (14) and finally obtain the branch voltages $e$ by reverting to (15).

*The Node Method:* If we start with the admittance form of Ohm's law given in (10), we may proceed to treat the network problem via the node method, following steps quite similar to those just described for the mesh method. Without describing the steps in detail, we simply list the corresponding equations:

$$(I - YE) + i = Ye \tag{19}$$

$$A^t(I - YE) = A^tYe \tag{20}$$

$$A^t(I - YE) = A^tYAe' \tag{21}$$

$$e' = (A^tYA)^{-1}A^t(I - YE). \tag{22}$$

In (21), $A^tYA$ is the nodal admittance matrix while in (22) $(A^tYA)^{-1}$ is the nodal solution matrix. The desired variables $e$ and $i$ may be computed from (13) and (19).

*The Cutset Method:* In both of the foregoing methods of analysis, the auxiliary variables $i'$ and $e'$ constitute a "basis" for all the branch currents $i$ or branch voltages $e$, respectively. A basis such as $e'$ is a linearly independent set of voltages in terms of which all the branch voltages may be computed, using (13). Any other basis besides the node-to-datum voltages will serve as well. In particular, we may use the set of tree-branch voltages $e_T$ which are linearly independent and in terms of which the branch voltages may be computed using the expression

$$e = De_T. \tag{23}$$

Here $D$ is the cutset matrix. This is a statement of Kirchhoff's voltage law and it is the counterpart of (13). The counterpart of (12) for Kirchhoff's current law is

$$D^ti = 0 \tag{24}$$

which follows directly from (6) and (14).

Using these relations, the steps involved in the cutset method of analysis are readily seen to be

$$(I - YE) + i = Ye \tag{25}$$

$$D^t(I - YE) = D^tYe \tag{26}$$

$$D^t(I - YE) = D^tYDe_T \tag{27}$$

$$e_T = (D^tYD)^{-1}D^t(I - YE). \tag{28}$$

Here, $D^tYD$ is the cutset admittance matrix and $(D^tYD)^{-1}$ the cutset solution matrix. The variables $e$ and $i$ may be evaluated using (23) and (25).

*The Mixed Method:* In addition to these three standard methods of analysis, there is a fourth which is beginning to be more widely used, namely the mixed method. This method, which is the basis for the writer's extension[16] of Bashkow's $A$-matrix formulation of the transient problem[23] and for a new method of ac analysis,[7] can also be used for dc analysis. Moreover, it underlies both Bryant's treatment[24] of Bashkow's formulation and the so-called state-variable approach[25] to transient analysis.

The idea behind the mixed method is that Ohm's law may be expressed in a mixed form involving the admittance matrix for certain branches of the network and the impedance matrix for the rest of the branches. Thus, we may write Ohm's law in the form

$$\begin{bmatrix} J_y \\ V_z \end{bmatrix} = \begin{bmatrix} Y_y & 0 \\ 0 & Z_z \end{bmatrix} \begin{bmatrix} V_y \\ J_z \end{bmatrix} \tag{29}$$

where the subscripts $y$ and $z$ refer to the branches which are treated as admittances or as impedances, respectively. With this equation, or its equivalent

$$\begin{bmatrix} I_y + i_y \\ E_z + e_z \end{bmatrix} = \begin{bmatrix} Y_y & 0 \\ 0 & Z_z \end{bmatrix} \begin{bmatrix} E_y + e_y \\ I_z + i_z \end{bmatrix} \tag{30}$$

as a starting point, we may analyze the admittance branches by the cutset method and the impedance branches by the mesh method—virtually considering the two sets of branches as comprising separate subnetworks. We then couple the resulting two sets of equations together by means of an appropriate topological matrix. The procedure is as follows.

Taking all the $y$-branches (admittances) first, a tree-link sort is performed. Any $y$-links thus identified will define basic meshes which contain *only* $y$-tree branches since no $z$-branches (impedances) have been considered up to this point. We will designate the paths-in-tree for these $y$-meshes by the submatrix $C_{Tyy}$.

The tree-link sort is then resumed, taking the $z$-branches into account. Now, however, the $z$-links define basic meshes whose paths-in-tree may involve both $y$-tree branches and $z$-tree branches, designated by the submatrices $C_{Tyz}$ and $C_{Tzz}$, respectively. Thus, the $C_T$ matrix for the entire network has the form

$$C_T = \begin{bmatrix} C_{Tyy} & C_{Tyz} \\ 0 & C_{Tzz} \end{bmatrix}. \tag{31}$$

If we express the $y$-branch voltages $e_y$ in terms of the $y$-tree branch voltages, $e_{Ty}$, and recall (5) and (23) we have

$$e_y = \begin{bmatrix} e_{Ty} \\ e_{Ly} \end{bmatrix} = \begin{bmatrix} U_{Ty} \\ -C_{Tyy}^t \end{bmatrix} e_{Ty} = D_y e_{Ty} \tag{32}$$

where $D_y$ is the cutset matrix for the $y$-branches alone with the $z$-branches regarded as being open-circuited. For the entire network, the corresponding relation is

$$\begin{bmatrix} e_{Ty} \\ e_{Tz} \\ e_{Ly} \\ e_{Lz} \end{bmatrix} = \begin{bmatrix} U_{Ty} & 0 \\ 0 & U_{Tz} \\ -C_{Tyy}^t & 0 \\ -C_{Tyz}^t & -C_{Tzz}^t \end{bmatrix} \begin{bmatrix} e_{Ty} \\ e_{Tz} \end{bmatrix} \tag{33}$$

where the $D$ matrix for all of the $y$- and $z$-branches is required.

Similarly, we may express the $z$-branch currents $i_z$ as a linear combination of the $z$-link currents $i_{Lz}$ which are identical with the mesh currents, obtaining the relation

$$i_z = \begin{bmatrix} i_{Tz} \\ i_{Lz} \end{bmatrix} = \begin{bmatrix} C_{Tzz} \\ U_{Lz} \end{bmatrix} i_{Lz} = C_z i_{Lz}. \tag{34}$$

Here, $C_z$ is the circuit matrix for the $z$-branches alone with the $y$-branches regarded as being short-circuited. Again, for the entire network, we have

$$\begin{bmatrix} i_{Ty} \\ i_{Tz} \\ i_{Ly} \\ i_{Lz} \end{bmatrix} = \begin{bmatrix} C_{Tyy} & C_{Tyz} \\ 0 & C_{Tzz} \\ U_{Ly} & 0 \\ 0 & U_{Lz} \end{bmatrix} \begin{bmatrix} i_{Ly} \\ i_{Lz} \end{bmatrix} \tag{35}$$

where the $C$ matrix for all of the $y$- and $z$-branches is required.

Now it is true for the *entire* network that $D^t i = 0$ and that $C^t e = 0$. As a consequence of these relations and (32)–(35), it may be shown that

$$D_y^t i_y = \begin{bmatrix} U_{Ty} - C_{Tyy} \end{bmatrix} \begin{bmatrix} i_{Ty} \\ i_{Ly} \end{bmatrix} = C_{Tyz} i_{Lz} \tag{36}$$

and

$$C_z^t e_z = \begin{bmatrix} C_{Tzz}^t \; U_{Lz} \end{bmatrix} \begin{bmatrix} e_{Tz} \\ e_{Lz} \end{bmatrix} = -C_{Tyz}^t e_{Ty}. \tag{37}$$

Returning to (30), we may rearrange terms and write

$$\begin{bmatrix} (I_y - Y_y E_y) \\ (E_z - Z_z I_z) \end{bmatrix} = \begin{bmatrix} Y_y & 0 \\ 0 & Z_z \end{bmatrix} \begin{bmatrix} e_y \\ i_z \end{bmatrix} - \begin{bmatrix} i_y \\ e_z \end{bmatrix}. \tag{38}$$

The next step is to premultiply the first row of this equation by $D_y^t$ and the second row by $C_z^t$ and to replace $e_y$ and $i_z$ by the expressions given in (32) and (34). The result thus obtained,

$$\begin{bmatrix} D_y^t (I_y - Y_y E_y) \\ C_z^t (E_z - Z_z I_z) \end{bmatrix} = \begin{bmatrix} D_y^t Y_y D_y & 0 \\ 0 & C_z^t Z_z C_z \end{bmatrix} \begin{bmatrix} e_{Ty} \\ i_{Lz} \end{bmatrix} - \begin{bmatrix} D_y^t i_y \\ C_z^t e_z \end{bmatrix}, \tag{39}$$

is—except for the second term on the right—exactly what would follow from using the cutset method of analysis on the $y$-branches, with the $z$-branches open-circuited, and the mesh method on the $z$-branches, with the $y$-branches short-circuited.

The final step is to replace the terms involving $i_y$ and $e_z$ by taking advantage of the relations developed in (36) and (37). When this is done, we find that

$$\begin{bmatrix} D_y^t (I_y - Y_y E_y) \\ C_z^t (E_z - Z_z I_z) \end{bmatrix} = \begin{bmatrix} D_y^t Y_y D_y & -C_{Tyz} \\ C_{Tyz}^t & C_z^t Z_z C_z \end{bmatrix} \begin{bmatrix} e_{Ty} \\ i_{Lz} \end{bmatrix} \tag{40}$$

where the topological matrix $C_{Tyz}$ effectively couples the two systems of cutset and mesh equations.

## Methods of Solving the Network Problem

The foregoing discussion relates to four general methods of network *analysis*—that is, "separation of (the network problem) into constituent parts or elements."[1] The present section refers to methods, primarily numerical, for *solving* network problems of three main types: dc, ac, and transient.

### *Dc Network Problems*

The most straightforward way of solving linear dc network problems is by Gaussian elimination[26],[27] of the variables in the mesh or node equations. For a general system of $n$ equations in $n$ unknowns having a full matrix of coefficients, the amount of computation required to carry out this elimination process is about $n^3/3$ operations (multiplications and additions). Indeed, it has recently been proved[28] that "no general system of linear algebraic equations can be solved in fewer operations than are required by Gaussian elimination."[27] However, for systems of equations having sparse coefficient matrices, which is the case in most network problems, a marked improvement in computational efficiency can be obtained by taking advantage of this sparsity.[29],[30] More about this later.

When several solutions are desired for a given network problem (or system of equations) with different voltage and/or current sources (right-hand sides), the usual method used is to compute the mesh or nodal solution matrix (inverse matrix) and then to multiply it by each different source vector (right-hand side). For a full matrix, the cost of inversion is about $n^3$ operations while the cost of each matrix-vector multiplication is $n^2$ operations.

A better and cheaper way to accomplish the same result, however, is to use the so-called $LU$ decomposition.[27] This involves finding a lower triangular matrix $L$ and an upper triangular matrix $U$ whose product $LU$ is equal to the coefficient matrix $A$ in the general system of equations $Ax = b$. But the objective of the Gaussian elimination procedure is to produce explicitly the upper triangular matrix $U$ with all its diagonal elements equal to 1. At the same time, the elements of the lower triangular matrix $L$ are implicitly generated and need only be saved—instead of being discarded, as is usually the case.

Fortunately, the essential elements of $L$ can be stored on the diagonal, in place of the diagonal 1's of the upper triangular matrix $U$, and below the diagonal in place of the zeros generated during the elimination process. Moreover,

[1] *Webster's Collegiate Dictionary.*

this can be done with negligible extra cost so that the $LU$ decomposition requires only $n^3/3$ operations. This is 1/3 the cost of matrix inversion. Then, since $LU=A$, the original system of equations $Ax=b$ can be replaced by the two *triangular* systems $Ly=b$ and $Ux=y$ which can both be solved in $n^2$ operations. This is equal to the cost of matrix-vector multiplication. Thus, the $LU$ decomposition technique is considerably cheaper than the conventional matrix inversion procedure for obtaining solutions for a succession of several right-hand side vectors.

There are several refinements, such as scaling and pivoting (interchanging rows and columns), which can improve the accuracy of the $LU$ decomposition.[26],[27] Iterative improvement of the first solution $x_1$ can also be achieved by a double-precision computation of the corresponding *residual* $r_1=b-Ax_1$ followed by a single-precision solution of the system $Ax_2=r_1$. The improved solution then is $x=x_1+x_2$. A program containing all these features has recently been published by Forsythe and Moler[27] (in ALGOL 60, FORTRAN, and PL/1) for solving the general system of equations $Ax=b$ using $LU$ decomposition.

In most network problems the coefficient matrix is so sparse, however, that it is worthwhile to take full advantage of this sparsity in programming the solution technique. Unfortunately; we electronic practitioners of the network analysis programming art have not been as alert to this possibility as our confreres in the electrical power industry. This may be due to the fact that the power engineer is being forced to consider much larger problems—in the order of 1000 to 2000 variables—than is the electronics engineer, for whom 50 to 100 variables seems to be an upper limit at present. With the advent of large-scale integrated circuits, though, there will soon be problems of ten times this size to be solved by the electronics engineer!

In two significant papers, Sato and Tinney[29] and more recently Tinney and Walker[30] have described techniques for modifying the $LU$ decomposition procedure to take advantage of the sparsity of the nodal admittance matrix. These techniques, which store and process only the nonzero elements of the matrices involved, greatly reduce both the computation time and the memory space required to solve network problems. In particular, these authors show that the *order* in which the variables are processed during Gaussian elimination strongly affects the amount of computation needed to triangularize a sparse matrix. This question has also been considered by Parter.[31]

One rule which considerably reduces the computation is to eliminate, at each stage, that variable for which the corresponding row in the still-to-be-triangularized matrix has fewest nonzero elements. Although this simple rule may not produce an optimum strategy, it does yield improvements in the order of 4 to 1 in computing speed.[29] This is because the rule helps to preserve sparsity in the upper triangular matrix which results from the elimination procedure. A better rule, proposed by Bree,[32] is based on a graph theoretical representation of the problem similar to that described by Parter.

As Sato and Tinney point out, the $LU$ decomposition procedure can be represented as a product of elementary matrices. These are matrices that differ from a unit matrix by either one diagonal or one off-diagonal entry.[26] And the (nonzero) elements of the $L$ and $U$ triangular matrices correspond to the nontrivial entries of these elementary matrices. Thus, the $LU$ decomposition can be used to provide a "product form of the inverse,"[33] in effect, in which only the essential terms are retained. For a very sparse matrix and with a proper ordering of the variables, the number of essential terms can be kept much smaller than that required for a full matrix so that a significant economy in both storage space and computation time can be effected. The price to be paid for this economy, however, is a very intricate indexing scheme to keep track of the row and column indices of the nonzero elements of the $L$ and $U$ matrices.

The order of elimination of variables affects not only the amount of computation in the $LU$ decomposition of sparse matrices, but also the accuracy of the solution—for both full and sparse matrices. To minimize round-off errors, the rule generally followed is to interchange rows and columns at each step of the elimination procedure so as to bring the largest possible term into the "pivotal" position.[26],[27],[33] This is the position on the diagonal corresponding to the next variable to be eliminated. In many network problems, the matrices are diagonally dominant so that pivoting may not be necessary. Even so, for best accuracy, this technique—or at least partial pivoting, in which only the rows are interchanged—is worthwhile and not too expensive in machine time. The program by Forsythe and Moler[27] previously mentioned incorporates the effect of row interchanges by indexing rather than by moving the data in memory. Whether or not this technique can be combined with that of Sato and Tinney remains to be explored.

Sometimes, inaccuracies arise which are not related to the round-off errors introduced in the process of solving equations or inverting a matrix. For example, suppose the node method is being used and it is desired to compute the current in a branch whose node voltages are almost equal. Clearly, a severe loss of accuracy will occur when these two node voltages are subtracted during the evaluation of (13), even if $e'$ is known to full machine precision. This kind of problem, then, is a function of the source vectors $E$ and $I$.

One way of avoiding this difficulty, which was actually used in TAP,[16] is to compute currents using the mesh method and voltages using the node method. An alternative which has not been tried, but merits investigation, is the mixed method.

Kron's method of interconnecting solutions[9]-[11] provides some tools which are very useful for certain applications. In particular, if it is desired to study the effect of varying a single resistance, letting it take on a succession of values, Kron's method provides a much more efficient way of handling this problem than solving each case from scratch. This method, which is the basis of the MODIFY feature of ECAP, permits the nodal solution matrix for the original network to be updated in accordance with changes

in one or more branch resistances. The amount of computation involved is trivial compared to that of solving the nodal equations or inverting the nodal admittance matrix all over again unless a sparse matrix version of the $LU$ decomposition is used.

A simple way of understanding the use of Kron's method in this application is the following. Consider the network configuration shown in Fig. 8(a) which consists of a single link connected from node $p$ to node $q$ of a tree all of whose branches converge on the datum node. The path-in-tree for the single mesh in this network is given by the $C_T$ matrix depicted in Fig. 8(b).

The mesh impedance matrix $C^tZC$ for this network is easily shown to be

$$C^tZC = C_T^tZ_TC_T + Z_L \tag{41}$$

where $Z_T$ is the impedance matrix for the tree while $Z_L$ is the impedance "matrix" of the single link being added to the tree.

Now by substituting (14) and (18) into (15), it can be shown that

$$e = [Z - ZC(C^tZC)^{-1}C^tZ](I - YE). \tag{42}$$

Then, by partitioning this equation into submatrices and subvectors relating to trees and links, it can further be shown that[10]

$$e_T = [Z_T - Z_TC_T(C^tZC)^{-1}C_T^tZ_T]D^t(I - YE). \tag{43}$$

Finally, by comparing this result with (28) and using (41), it follows that

$$(D^tYD)^{-1} = Z_T - Z_TC_T(C_T^tZ_TC_T + Z_L)^{-1}C_T^tZ_T. \tag{44}$$

Thus, we have derived from the *mesh* method an expression for updating the cutset solution matrix for any tree to which a single link has been added. (Remember that for a *tree*, the impedance matrix $Z_T$ is identical with the tree solution matrix $(D^tYD)^{-1}$ since $D=D_T$ is a unit matrix.) But in view of (4) and (5) we can extend this to an expression for updating the nodal solution matrix as well,[10] namely

$$(A^tYA)^{-1} = Z_1 - Z_1A_L^t(A_L^tZ_1A_L + Z_L)^{-1}A_LZ_1 \tag{45}$$

where $Z_1 = B_T^tZ_TB_T$.

Returning now to our original problem, that of updating the nodal solution matrix when a single branch resistance (between nodes $p$ and $q$) is changed, we assume that this change is effected by adding another resistance in parallel, either positive or negative in value. We assume also that the nodal solution matrix for the original network has been computed. This matrix may then be interpreted as the impedance matrix $Z_T$ of a tree having the "star" configuration shown in Fig. 8(a). Since $B_T$ for this tree is a unit matrix, it follows that we may substitute $Z_T$ for $Z_1$ in (45) which thus enables us to update the nodal solution matrix one link at a time.

Equation (45) may be used recursively for adding several links, one after the other.[10] If this technique is used, however, to compute a sequence of solutions as a given resistance is stepped through several different values, round-

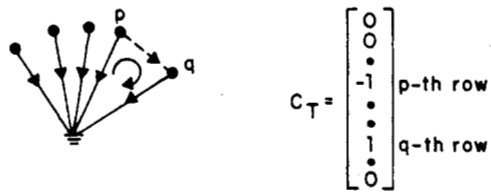


Fig. 8. Connection of single link to a tree.

off errors will accumulate from one step to the next. If the range of variation of resistance is large enough and many steps are taken, the errors may eventually become serious enough to vitiate the results. A better technique, but one requiring more memory space, would be to update only the *original* nodal solution matrix each time. This would preclude the accumulation of errors from one step to the next.

By suitably extending this idea of updating the nodal solution matrix to take into account changes of a single resistance value, differential changes in resistance can be dealt with. In other words, the partial derivative or sensitivity of the nodal solution matrix with respect to any resistance change can be computed explicitly.[10]

Rather than describing how this may be done starting from (45), we may derive a similar and broader relation by differentiating the nodal equations themselves. Thus, from (21) it follows that

$$A^t(dI - Y\,dE - dYE) = A^tdYAe' + A^tYA\,de' \tag{46}$$

or

$$de' = (A^tYA)^{-1}A^t[dI - Y\,dE - dY(E + e)]. \tag{47}$$

The latter expression gives the variation in node voltages as a function not only of changes in resistance (specified as $dY$) but also of changes in the $I$ and $E$ vectors. This equation is used in ECAP for computing sensitivities.

The question of solving nonlinear dc network problems is an important one and difficult to answer with complete assurance. One standard technique that is useful in many cases, however, is the Newton-Raphson iteration method.[34] For solving a single nonlinear equation $f(x)=0$, this method uses the recursion formula,

$$x_{i+1} = x_i - f(x_i)/f'(x_i) \tag{48}$$

where $f'(x_i)$ is the value of the derivative $df/dx$ at the point $x_i$.

This formula may be extended to handle a system of $n$ nonlinear equations,

$$f_j(x_1, x_2, \cdots, x_n) = 0;\ j = 1, 2, \cdots, n \tag{49}$$

by generalizing the role of the derivative. If we define the Jacobian matrix $H$ in terms of its $jk$th element,

$$H_{jk} = \partial f_j/\partial x_k \tag{50}$$

then (48) may be replaced by

$$x_{i+1} = x_i - H^{-1}f_i \tag{51}$$

where $x_i$ and $f_i$ are vectors. An adaptation of this method has been used with some success in TAP[16] and in the NET-1 program.[17]

When the Newton-Raphson method converges, it does so quadratically. That is, at each iteration the error is proportional to the square of the previous error. However, the method may diverge under certain circumstances, for example, in the case of transistor switching circuits with positive feedback.

Broyden[35] has described a class of methods which remedy the two main drawbacks of the Newton-Raphson method; namely, the liability to diverge and the need to evaluate and invert the Jacobian matrix. Broyden's method cannot diverge because it never allows the length of the vector $f_i$ to increase. It also provides a method for updating the inverse Jacobian matrix at each iteration, starting with an arbitrary approximation to this inverse. Branin and Wang[36] have applied this method to the solution of nonlinear dc network problems and find that it is fast and reliable. They have also extended it to statistical analysis of both linear and nonlinear networks.

Another approach to solving the nonlinear network problem has been described by Katzenelson.[37] This method converges in a finite number of steps if the resistance characteristics are continuous, monotonically increasing, and piecewise linear. Under these restrictions, the network problem has a unique solution. The solution is found by solving a succession of linear network problems which are *locally* equivalent to the actual problem.

This sequence of linear problems is generated by gradually increasing the voltage and/or current sources from zero to full value and solving each locally linear problem within a finite region whose boundaries are determined by the breakpoints in the piecewise linear characteristics of each resistor. Thus, a solution curve, consisting of a finite number of straight line segments, is traced out in the multidimensional space of solutions, ending at the desired solution point.

The key to the computational effectiveness of Katzenelson's algorithm is the use of Kron's method for updating the cutset solution matrix by means of the link-at-a-time technique indicated in (44). Although the method might be used successfully for nonlinear problems with multiple solutions (e.g. tunnel diode circuits, multivibrators, etc.) it is guaranteed to converge only under the conditions mentioned which insure a unique solution.

### *Ac Network Problems*

Since all of the theoretical developments up through (47) apply to complex as well as to real numbers, steady-state ac network problems may be handled using any of the four basic methods of analysis described above as well as Kron's method. For example, by programming the Gaussian elimination procedure for complex arithmetic, the mesh or nodal equations for any ac problem may be solved directly, at a computational cost of approximately four times that for real arithmetic. However, since the coefficient matrix is frequency-dependent, the equations must be solved at each new frequency. This approach is usually quite reliable, although it may involve reduced numerical accuracy in the vicinity of a resonant frequency.

A new approach to the ac problem,[7] which is based on the mixed method of analysis, tackles the job of matrix inversion by first solving the much more difficult problem of computing the eigenvalues and eigenvectors[26] of the matrix. At first glance, solving the eigenvalue problem appears to be "doing it the hard way," since the amount of computational effort required is an order of magnitude greater than that for inverting a matrix. This would indeed be the case if only a single inverse matrix were needed. But in calculating frequency response, a new inverse, or more precisely, a new solution vector, is needed at each frequency. In this case, using the eigenvalues and eigenvectors turns out to be just the right thing to do because it reduces the problem to one of inverting a diagonal matrix at each new frequency.

Starting with (40) of the mixed method, we can derive the differential equations for an RLC network in first-order form as follows. All capacitors are treated as admittance branches and hence may have a nonzero conductance in parallel. Thus, the admittance matrix becomes

$$Y_y = pK + G \tag{52}$$

where $p = d/dt$ and the symbol $K$ is used for capacitance so as not to conflict with the symbol $C$ for the circuit matrix. On the other hand, all inductors are regarded as impedance branches with (possibly nonzero) resistance in series so that the impedance matrix is

$$Z_z = pL + R. \tag{53}$$

(Branches consisting of resistance only can also be handled. But, since these give rise to algebraic equations[16] which unduly complicate the discussion, we shall not consider them here.)

Using (52) and (53) together with (40), we may write the differential equations (for constant $K$ and $L$)

$$\begin{bmatrix} D^tKD & 0 \\ 0 & C^tLC \end{bmatrix} \begin{bmatrix} pe_T \\ pi_L \end{bmatrix} + \begin{bmatrix} D^tGD & -C_{Tyz} \\ C_{Tyz}^t & C^tRC \end{bmatrix} \begin{bmatrix} e_T \\ i_L \end{bmatrix} = \begin{bmatrix} I_T \\ E_L \end{bmatrix} \tag{54}$$

where the symbols $I_T$ and $E_L$ designate the driving currents and voltages.

In more compact form, (54) may be written as

$$P\dot{x} + Qx = f(t) \tag{55}$$

where $\dot{x} = dx/dt$ and $x$ is the state-variable vector of capacitive tree-branch voltages and inductive link currents. By inverting $P$ we may rewrite (55)

$$\dot{x} = Ax + g(t) \tag{56}$$

where $g = P^{-1}f$ and $A = -P^{-1}Q$, which corresponds to Bashkow's $A$ matrix.[23] (This matrix is not to be confused with the branch-node matrix $A$, which will not be referred to any further.)

For steady-state ac problems, the driving function may be written as

$$g(t) = g_0 e^{st} \tag{57}$$

with $s$ a complex number. The steady-state response then is

$$x(t) = x_0 e^{st}. \tag{58}$$

Substituting these relations in (56) we have the linear system of equations

$$(sU - A)x_0 = g_0 \tag{59}$$

where $g_0$ is a known vector and $x_0$ is to be determined.

The formal solution for (59) is

$$x_0 = (sU - A)^{-1} g_0 \tag{60}$$

and so we need to find an effective way of computing $(sU - A)^{-1}$. This may be done using the eigenvalues $\lambda_i$ and eigenvectors $\chi_i$ of the $A$ matrix as follows.

Let $\Lambda$ be the diagonal matrix of eigenvalues and $X$ the matrix whose columns are the corresponding eigenvectors. The eigenproblem, then, is characterized by

$$AX = X\Lambda \tag{61}$$

which may be recast into the form

$$X^{-1}AX = \Lambda. \tag{62}$$

This defines a similarity transformation[26] which *diagonalizes* the matrix $A$.

The same transformation also diagonalizes $(sU - A)$ giving

$$X^{-1}(sU - A)X = sU - \Lambda. \tag{63}$$

Solving this expression for $(sU - A)$ and then inverting, we find that

$$(sU - A)^{-1} = X(sU - \Lambda)^{-1}X^{-1} \tag{64}$$

so that (60) becomes

$$x_0 = X(sU - \Lambda)^{-1}X^{-1}g_0. \tag{65}$$

This is the basic result of the new method, requiring the inversion of the diagonal matrix $(sU - \Lambda)$ at each frequency.

If we let $d = X^{-1}g_0$, which we may calculate once for all, then the $i$th element of the $x_0$ vector may be computed using the relation

$$x_i = \sum_{j=1}^{n} \frac{X_{ij}d_j}{s - \lambda_j}. \tag{66}$$

This partial fraction expansion of the state variable $x_i$ may be evaluated at a computational cost proportional to $n$ operations per frequency as opposed to $n^3$ operations if (59) were solved by Gaussian elimination.

Now the cost of solving the eigenproblem is not negligible since it is equivalent to solving (59) at roughly 10 frequencies. Accordingly, the present method would not pay off if solutions were required at only a few frequencies. But for a frequency response calculation, this method is very effective.

It is quite easy to extend this method to the computation of driving point and transfer functions of the network and to derive sensitivity formulas. For example, the frequency sensitivity of $x_0$ is

$$\frac{dx_0}{ds} = -X(sU - \Lambda)^{-2}X^{-1}g_0 \tag{67}$$

(see Branin[7] for details). But one of the more interesting features of the method is the fact that it is far better suited for computer-sized problems than the traditional Laplace transform techniques involving ratios of polynomials and the poles and zeros thereof.

In particular, the task of computing the coefficients of the polynomials in a network function $P(s)/Q(s)$ is not only time-consuming but also prone to serious numerical inaccuracies, especially when the polynomials are of high degree. The so-called "topological formula" approach to computing these network functions involves finding all the trees of a network and then computing the sum of the corresponding tree-admittance products. But the number of trees may run into millions for a network with only 20 nodes and 40 branches.[2] And even if this were not enough of an impediment, the computation of the roots of the polynomials $P(s)$ and $Q(s)$ is hazardous because these roots may be extremely sensitive to errors in the coefficients.[26]

In the writer's judgment, therefore, the polynomial approach is just not matched to the network analysis tasks which the computer is called upon to handle. The eigenvalue approach is much better suited and gives all of the theoretical information that the Laplace transform methods are designated to provide. For example, the eigenvalues are identical with the poles of the network functions, as (66) indicates. Moreover, any network function desired may be computed straightforwardly and its sensitivity obtained, either with respect to frequency or with respect to any network parameter. Finally, even the pole sensitivities can be calculated.[7]

This is not to say the eigenvalue method is free of all difficulties. Even with the $Q$-$R$ method of Francis[38] which, according to Wilkinson,[26] is "the most effective of known methods for the solution of the general algebraic eigenvalue problem," it is possible that numerical errors of a crippling nature may arise when networks having a large ratio of greatest to smallest eigenvalue are treated. The difficulty with this kind of "pathological" network is that although the large eigenvalues can be computed accurately, the small ones cannot. But it is the small ones which are principally responsible for determining the behavior of the network, as (66) implies.

One possible remedy for this situation is to solve the eigenvalue problem for the inverse matrix $A^{-1}$. Then the situation is reversed, for the large eigenvalues of $A^{-1}$, which will be obtained accurately, are the reciprocals of the small eigenvalues of $A$. Computing $A^{-1}$ may be error-prone in its own right because of the large ratio of eigenvalues. But if the matrix $Q$ in (55) is nonsingular, then we may use the relation

$$A^{-1} = -Q^{-1}P \tag{68}$$

to compute a more accurate inverse for $A$. This question remains to be investigated.

Fortunately, the problem of multiple eigenvalues presents

no difficulty since the *Q-R* method is able to handle this situation quite effectively. If, however, the matrix is such that it does not have a complete set of linearly independent eigenvectors, it may be necessary to perturb the matrix slightly so as to avoid this difficulty. From an engineering point of view, this subterfuge would probably be completely acceptable in most instances.

*Transient Network Problems*

Of the three types of network problem under discussion, transient problems present the greatest computational difficulty. This is because the numerical integration of the differential equations is burdensome for nonlinear networks and extremely slow when the network has a wide spread of eigenvalues. The nature of these problems will be discussed presently.

*Analytic Solution of Linear Problems:* To fix ideas, let us first consider the analytic solution of the state-variable equations for a linear network with constant parameters. Referring to (56), we assume a solution of the form

$$x(t) = e^{At}v(t) \tag{69}$$

where $v(t)$ is a vector function of time and where $e^{At}$ is defined as the matrix function

$$e^{At} = U + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \cdots. \tag{70}$$

By differentiating (69) and using the definition of $e^{At}$, it follows that

$$\dot{x} = Ae^{At}v + e^{At}\dot{v} = Ax + e^{At}\dot{v}. \tag{71}$$

Combining this with (56), we obtain a differential equation for $v(t)$ in the form

$$\dot{v} = e^{-At}g(t) \tag{72}$$

whose solution may be written as

$$v(t + h) = v(t) + \int_0^h e^{-A(t+\tau)}g(t + \tau)\,d\tau. \tag{73}$$

Multiplying this equation by $e^{A(t+h)}$ and using (69) we find that the solution to (56) is

$$x(t + h) = e^{Ah}x(t) + \int_0^h e^{A(h-\tau)}g(t + \tau)\,d\tau. \tag{74}$$

This solution is exact, but its utility depends on the computation of $e^{Ah}$.

Liou[1],[39] has proposed the use of a finite number of terms in the series expansion of $e^{Ah}$ according to (70), asserting that this method "completely avoids eigenvalue problems." In one sense, this assertion is true in that Liou's method does not require the calculation of eigenvalues. On the other hand, the method runs headlong into another kind of eigenvalue problem which seriously limits its usefulness; namely, that when the matrix $A$ has a large eigenvalue, the integration step size $h$ must be kept small, roughly less than the reciprocal of $|\lambda_{\max}(A)|$, in order to permit rapid convergence of (70). Precisely the same limitation arises in numerical integration using, say, the Runge-Kutta or Adam's method, where $h$ must be kept small to avoid numerical instability.[34] More about this later.

If, instead of avoiding the computational eigenvalue problem, we solve it, then (74) becomes as practical to use as (65) and (66). From (70) it can be shown that the transformation which diagonalizes $A$ also diagonalizes $e^{Ah}$, yielding the relation

$$X^{-1}e^{Ah}X = e^{\Lambda h} \tag{75}$$

where $e^{\Lambda h}$ is a diagonal matrix whose terms $e^{\lambda_i h}$ are trivial to compute. Solving this expression for $e^{Ah}$, we may recast (74) into the form

$$x(t + h) = Xe^{\Lambda h}X^{-1}x(t) + \int_0^h Xe^{\Lambda(h-\tau)}X^{-1}g(t + \tau)\,d\tau \tag{76}$$

and our computational problem reduces to one of evaluating the integral expression. If the driving function $g(t+\tau)$ is approximated as a polynomial in $\tau$, the integration can be carried out exactly, term by term. This provides a very satisfactory way of evaluating the integral on a digital computer.

Equation (76) is instructive from a theoretical point of view. For example, if $g(t)=0$ and we apply initial conditions to the network which correspond to, say, the $i$th eigenvector $\chi_i$, then we may write the solution as

$$x(t) = Xe^{\Lambda t}X^{-1}\chi i. \tag{77}$$

But since $X^{-1}X$ is a unit matrix, $X^{-1}\chi_i$ is equal to the $i$th column of a unit matrix. (Remember that the columns of $X$ are the eigenvectors $\chi_i$.) Thus, the eigenfunction $e^{\lambda_i t}$ is singled out and (77) reduces to

$$x(t) = \chi_i e^{\lambda_i t}. \tag{78}$$

In other words, the $i$th eigenvector prescribes the particular initial conditions required to excite the $i$th characteristic mode (eigenmode) of response and suppress all the others. For arbitrary initial conditions, given by a vector $x_0$, all of the eigenmodes will be excited, in general. But the matrix-vector product $X^{-1}x_0$ automatically produces the appropriate linear combination of eigenfunctions in terms of which to express the solution.

For the case where $g(t)\neq 0$ the "initial conditions" are continually changing, so to speak, and so an additional term involving a convolution integral of the eigenfunctions and driving functions is required, as in (76).

*Numerical Integration:* The numerical methods of integration currently in use are, for the most part, based on the use of finite difference approximations to the actual differential equations to be solved. For a single differential equation of the form

$$dx/dt = f(x, t) \tag{79}$$

a solution may be effected by the general finite difference expression[34]

$$x_n = \sum_{i=1}^{M} a_i x_{n-i} + h \sum_{j=0}^{N} b_j f_{n-j} \tag{80}$$

where $h$ is the integration interval, $x_n = x(nh)$, and $f_{n-j} = (x_{n-j}, (n-j)h)$. The number and values of the coefficients $a_i$ and $b_i$ are determined by the number of terms of the Taylor expansion of $x(nh)$ which are to be matched by the integration formula so defined.

If $b_0 = 0$, the integration formula is called a "predictor," since the new value of $x_n$ is predicted in terms of previous values ($x_{n-i}$) and derivatives ($f_{n-j}$) only. If $b_0 \neq 0$, the formula is called a "corrector," since it is generally used in conjunction with a predictor formula to provide a corrected value of $x_n$. This is accomplished by using the *predicted* value of $x_n$ to compute an approximate value of the "new" derivative $f_n$, which is then used in (80).

Now, in the simple case of one homogeneous linear differential equation with constant coefficient, where $f(x, t) = gx(t)$, (80) may be written in the form

$$(1 - hgb_0)x_n - (a_1 + hgb_1)x_{n-1} - \cdots - (a_r + hgb_r)x_{n-r} = 0. \quad (81)$$

Here, at least one of the coefficients $a_r$ or $b_r$ is nonzero and all coefficients of higher index are zero. That is, $r = \max(M, N)$. If we assume a particular solution of the form $x_n = p^n$, then (81) may be reduced to the polynomial equation[34]

$$(1 - hgb_0)p^r - (a_1 + hgb_1)p^{r-1} - \cdots - (a_r + hgb_r) = 0. \quad (82)$$

Obviously, any value of $p$ which is a root of this *characteristic equation* is a candidate for the particular solution $x_n = p^n$, but the general solution takes the form of a linear combination of particular solutions, namely

$$x_n = \sum_{k=1}^{r} c_k p_k^n \quad (83)$$

where the $p_k$ are all the roots of (82).

One of these roots, lying near the point $1 + j0$ in the complex plane, will be the *principal* root since it primarily determines the desired approximation to the solution of the differential equation in question. All the other roots are parasitic roots and it is clear from (83) that if any parasitic root lies outside the unit circle, $x_n$ will eventually diverge. This situation is called "numerical instability" because it vitiates the solution.

In order to prevent numerical instability, the integration step $h$ must be kept small enough to insure that all of the parasitic roots of (82) remain inside the unit circle. In the case of a *system* of linear differential equations with constant coefficients, as typified by (56), the requisite condition on $h$ for most integration formulas is

$$h|\lambda_{\max}(A)| \leq c. \quad (84)$$

Here, $\lambda_{\max}(A)$ is the largest eigenvalue of the coefficient matrix $A$ while the constant $c$ is in the order of 1 or 2, depending on the integration method used. Even in the case of nonlinear differential equations, an equivalent restriction, involving the (instantaneously) largest eigenvalue of the Jacobian matrix, must be imposed to prevent numerical instability.

It is just this constraint on the integration interval which makes most integration techniques so painfully slow in the case of network problems having a wide spread of eigenvalues. For the largest eigenvalue (or, equivalently, its reciprocal, the smallest time constant) controls the permissible size of the integration step $h$. But the smallest eigenvalues (largest time constants) control the network response and so determine the total length of time over which the integration must be carried out to characterize the response. In the case of a network with a 1000 to 1 ratio of largest to smallest eigenvalue, for instance, it might be necessary to take in the order of 1000 times as many integration steps with, say, the Runge-Kutta method as with some method which was free of the restriction given by (84). This so-called "minimum time-constant" barrier is without doubt the biggest obstacle to be overcome in the transient analysis of networks.

There are, however, some integration formulas in common use for which the constraint on $h$ given in (84) does not apply; for example,

$$x_n = x_{n-1} + hf_n \quad (85)$$

and

$$x_n = x_{n-1} + \frac{h}{2}(f_n + f_{n-1}). \quad (86)$$

Both of these have characteristic equations of the first degree and so have no parasitic roots. But the main disadvantage in using these formulas is that they are of such low-order accuracy, particularly (85), that the integration interval must often be shortened anyway to insure sufficient accuracy. Another disadvantage is that both are *implicit* formulas because they involve $f_n$, which cannot be computed until $x_n$ is known. When applied to (56), therefore, these formulas require the solution of a linear system of equations (or matrix inversion) which complicates their use.

Schemes for retaining the numerical stability inherent in (85) and (86) without the need to solve simultaneous equations have been reported by Stineman[40] and by Richards, Lanning, and Torrey[41] but the resulting methods are again of low-order accuracy. Treanor[42] on the other hand, describes a method of high (fourth-order) accuracy which is identical with the Runge-Kutta method for short integration intervals but retains numerical stability even for every large integration intervals. His method, however, involves computation of the matrix function $e^{Ah}$ and so presents no real advantage over the eigenvalue approach for linear problems. In the case of nonlinear problems, the difficulty in using Treanor's method is compounded by the fact that $e^{Ah}$ must be recomputed at each time step.

*Quasi-Analytic Solutions:* A quite different attack on this problem has been reported by Cohen and Flatt[43],[44] and by Certaine.[45] (Incidentally, the title of Certaine's paper is misleading since what he calls "time constant" is the reciprocal of the time constant usually referred to by electrical engineers.) The essence of this attack is the following: in (56) the diagonal terms of the coefficient matrix $A$ are handled analytically, as was $A$ itself in (74), while the off-diagonal terms of $A$ are lumped together with $g(t)$ and considered as part of the driving function.

In other words, letting $A = D + B$, where $D$ contains all of the diagonal terms of $A$ while $B$ contains all the off-diagonal terms, we may write (56) as

$$\dot{x} = Dx + [Bx + g(t)] \tag{87}$$

with $Bx + g(t)$ serving as a driving function. The analytic solution to this equation can be obtained by modifying (74) to read

$$x(t+h) = e^{Dh}x(t) + \int_0^h e^{D(h-\tau)}[Bx(t+\tau) + g(t+\tau)]\,d\tau. \tag{88}$$

Here, $e^{Dh}$ is easy to compute, since $D$ is a diagonal matrix. The integration can be carried out by using a polynomial approximation of any desired degree for $Bx+g$. (The method of integration used in NET-1 is based on a modification of this approach.)

When the matrix $A$ is dominantly diagonal, the use of (88) may permit a much larger integration step size than that allowed in the usual methods. However, for network problems, this property of diagonal dominance of the $A$ matrix does not generally obtain and so (88) is not of itself sufficient to break though the integration barrier.

The writer has found that (88) has its own peculiar numerical instability which seems to be avoidable by choosing the integration interval subject to a restriction of the form

$$h|\lambda_{\max}(B)| \leq c \tag{89}$$

where $c$ is a small constant. In other words, the largest eigenvalue of the *off-diagonal* matrix $B$ controls the stability of the numerical integration process used in applying (88).

From a theoretical point of view, it is possible to choose a similarity transformation on $A$ of the form

$$T^{-1}AT = \bar{A} = \bar{D} + \bar{B} \tag{90}$$

where $\bar{D}$ is the diagonal component and $\bar{B}$ the off-diagonal component of $\bar{A}$, such that $\lambda_{\max}(\bar{B})$ is arbitrarily small. Indeed, if $T$ is so chosen that $\lambda_{\max}(\bar{B}) = 0$, then $\bar{D}$ contains all the eigenvalues of $\bar{A}$. (In effect, the $Q$-$R$ algorithm[38] involves an iterative process for finding a succession of similarity transformations each of which reduces $\lambda_{\max}(\bar{B})$ until it becomes vanishingly small.)

If we choose a transformation of variables of the form

$$y = T^{-1}x \tag{91}$$

to be used in (90)—with the intent of making $\lambda_{\max}(\bar{B})$ small, but not necessarily zero—then (56) becomes

$$T\dot{y} = ATy + g(t) \tag{92}$$

or

$$\dot{y} = T^{-1}ATy + T^{-1}g(t). \tag{93}$$

Accordingly, in view of (87), (88), and (90), we may write the solution as

$$y(t+h) = e^{\bar{D}h}y(t) + \int_0^h e^{\bar{D}(h-\tau)}[\bar{B}y(t+\tau) + w(t+\tau)]\,d\tau \tag{94}$$

where $w(t) = T^{-1}g(t)$. In effect, then, we solve the original problem by transforming to the domain of the new variables $y$ before integrating.

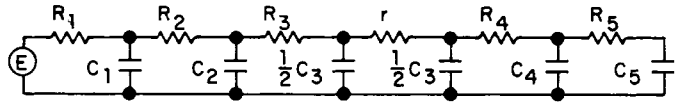

Fig. 9. RC ladder network.

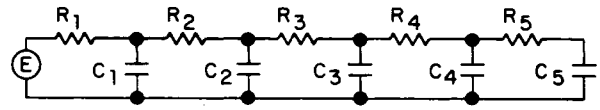

Fig. 10. Modified RC ladder network.

From (94) and the relation $x = Ty$, a solution to the problem $\dot{x} = Ax + g(t)$ may be computed. Moreover, if $T$ has been chosen so as to reduce $\lambda_{\max}(B)$ sufficiently, a large integration step $h$ may be used without risking numerical instability. The validity of this procedure has been verified in handling the following simple problem.

Consider the RC network shown in Fig. 9. With all resistances equal and with all capacitances equal, the maximum integration step which could be taken safely using (88) was found to be about equal to the time constant, RC. In the case of the modified network of Fig. 10, with $C_3$ split in half and the bridging resistor $r$ having a very small value relative to $R$, the integration step was found to be limited by the time constant $rC$.

Originally, the state-variable vector used for both networks consisted of all the capacitor voltages. However, when a different combination of voltages was used in treating the second network—namely, the voltages across capacitors $C_1$, $C_2$, $C_4$, and $C_5$ but the *sum* and *difference* of the voltages across the two half-size capacitors—it became possible to increase the integration step size until it was equal to that used in the first network. This simple change of variables defined a transformation which was sufficient to reduce significantly the maximum eigenvalue of $\bar{B}$ in (90), though not sufficient to expose all the eigenvalues explicitly.

The lesson to be learned from the foregoing considerations is evidently this: one key to the minimum time constant problem seems to lie in operating on the coefficient matrix $A$ in some meaningful way. But if the $A$ matrix is left intact, then we must accept the limitations implicit in (84). The simplest kind of operation on the $A$ matrix is to extract its diagonal terms to be handled analytically as in (88). The next more difficult operation is to transform the matrix as in (90)–(94). The ultimate, of course, is to solve the eigenproblem completely.

Thus, there are clear guidelines to follow in tackling the integration problem for nonlinear systems where the Jacobian matrix now plays the role of the $A$ matrix. Obviously, solving the eigenproblem repeatedly is inappropriate, since the Jacobian matrix changes during the course of the solution. However, finding a "quick and dirty" transformation $T$ to be used in conjunction with (90)–(94) does seem plausible. Moreover, it should not be necessary to update $T$ at each integration step since it is likely that a given trans-

formation could be used for many integrations before becoming degraded to the point of uselessness.

The feasibility of this approach would depend not only on minimizing the frequency of updating $T$ but also on developing an efficient algorithm for computing this transformation. If the average computational cost per integration step can be kept proportional to the square of the number of equations, then the minimum time-constant barrier can be breached for nonlinear problems as well as for linear.

In the case of linear systems, the superiority of the eigenvalue approach for both ac and transient analysis is unmistakable, since it goes directly, and with reasonable computational efficiency, to the very heart of the problem. This is not too surprising in view of the well known facts that the eigenproblem is the most fundamental problem in matrix theory and that its solution provides essentially *all* there is of real value to be learned about a matrix. Fortunately, the advent of the digital computer, and the *Q-R* algorithm, now brings this powerful theoretical tool down to a practical working level.

## Future Developments

It is evident from the topics discussed in this paper that much work needs to be done in the immediate future to upgrade the network analysis programs currently available to circuit designers. In addition to the incorporation of new and improved analytical techniques, these upgraded programs will need to provide facilities for statistical analysis of circuits, for storage and retrieval of device models, for handling transmission lines, and for treating much larger problems. Graphical presentation of output variables in a variety of forms will be essential and a much closer degree of man-machine interaction will become possible through the use of CRT displays and remote access to the computer. A good example of this kind of development is Katzenelson's AEDNET program.[1],[46]

In another direction, there is a growing need to handle integrated and monolithic circuits. This new technology is already placing strong demands on our analytical, numerical, and programming techniques and it will inevitably force the development of more powerful methods for solving partial differential equations. This, in turn, is likely to stimulate the exploration and exploitation of such techniques as Kron's method of interconnecting solutions,[9]-[11] the use of higher dimensional network models,[22] and a novel numerical method for treating partial differential equations recently described by Polozhii.[47]

A step beyound the production of more powerful and effective network analysis programs will be the development of *optimization* techniques—in which the analysis tasks will likely be relegated to subroutine status. Significant steps are already being taken in this direction.[48]-[50] Moreover, automatic network synthesis is also being carried out to some extent.[48] It is probable, however, that for a long time to come a majority of computer-aided design will be based primarily on analysis and optimization techniques, with the engineer himself not only playing an essential role in the "design loop" but also controlling the entire process of design. Thus, while the computer will certainly take over the routine computational tasks of the engineer, this will enhance rather than supplant his creative abilities—which are really his *raison d'être*.

## References


[1] F. F. Kuo, "Network analysis by digital computer," *Proc. IEEE*, vol. 54, pp. 820–829, June 1966.

[2] D. Christiansen, "Computer-aided design: Pt. I. The man-machine merger," *Electronics*, vol. 39, pp. 110–123, September 19, 1966.

[3] H. N. Ghosh, F. H. de la Moneda, and N. R. Dono, "Computer-aided transistor design, characterization and optimization," this issue, p. 1897.

[4] J. J. Olsztyn and T. L. Theodoroff, "GMR DYANA Programming Manuals I and II," General Motors Research Labs., Warren, Mich., 1959.

[5] S. J. Fenves, "STRESS (structural engineering system solver) a computer programming system for structural engineering problems," M.I.T., Cambridge, Mass., Tech. Rept. T63-2, June 1963.

[6] S. J. Fenves and F. H. Branin, Jr., "A network-topological formulation of structural analysis," *ASCE J. Struct. Div.*, vol. 89, pp. 483–514, August 1963.

[7] F. H. Branin, Jr., "A new method for steady state A-C analysis of RLC networks," *IEEE Internat'l Conv. Rec.*, pt. 7, pp. 218–223, March 1966.

[8] G. Kron, *Tensor Analysis of Networks*. New York: Wiley, 1939.

[9] F. H. Branin, Jr., "Kron's method of tearing and its applications," *Proc. 2nd Midwest Symp. on Circuit Theory* (Michigan State University, East Lansing, Mich.), December 1956.

[10] ——, "The relation between Kron's method and the classical methods of network analysis," *IRE WESCON Conv. Rec.*, pt. 2, pp. 3–28, August 1959.

[11] G. Kron, "A set of principles to interconnect the solutions of physical systems," *J. Appl. Phys.*, vol. 24, pp. 965–980, 1953.

[12] J. P. Roth, "An application of algebraic topology to numerical analysis: on the existence of a solution to the network problem," *Proc. Nat'l Acad. Sciences*, vol. 41, pp. 518–521, 1955.

[13] ——, "The validity of Kron's method of tearing," *Proc. Nat'l Acad. Sciences*, vol. 41, pp. 599–600, 1955.

[14] ——, "An application of algebraic topology: Kron's method of tearing," *Quart. Appl. Math.*, vol. 17, pp. 1–24, 1959.

[15] N. G. Brooks and H. S. Long, "A program for computing the transient response of transistor switching circuits—PE TAP," IBM Development Lab., Poughkeepsie, N. Y., Tech. Rept. 00.700, 1959.

[16] F. H. Branin, Jr., "D-c and transient analysis of networks using a digital computer," *IRE Internat'l Conv. Rec.*, pt. 2, pp. 236–256, March 1962.

[17] A. F. Malmberg, F. L. Cornwell, and F. N. Hofer, "NET-1 network analysis program," Los Alamos Scientific Lab., Los Alamos, N. Mex., Rept. LA-3119, 1964.

[18] "1620 electronic circuit analysis program (ECAP)," IBM Corporation, Data Processing Div., White Plains, N. Y., Application Program 1620-EE-02X.

[19] J. B. Ward and H. W. Hale, "Digital computer solution of power-flow problems," *Trans. AIEE (Power Apparatus and Systems)*, vol. 75, pt. III, pp. 398–404, June 1956.

[20] A. H. El-Abiad, M. Watson, and G. W. Stagg, "The load-flow problem: its formulation and solution, pt. I," AIEE Paper GlCP1054, 1961.

[21] H. E. Brown, G. K. Carter, H. H. Happ, and C. E. Person, "Power flow solution by impedance matrix iterative method," *IEEE Trans. Power and Apparatus*, vol. 82, pp. 561–564, August 1963.

[22] F. H. Branin, Jr., "The algebraic-topological basis for network analogies and the vector calculus," *Proc. Symp. on Generalized Networks*, vol. 16, Microwave Research Inst. Symposia Ser., Polytechnic Inst. of Brooklyn, pp. 453–491, 1966.

[23] T. R. Bashkow, "The *A* matrix, new network concept," *IRE Trans. Circuit Theory*, vol. CT-4, pp. 117–119, September 1957.

[24] P. R. Bryant, "The explicit form of Bashkow's A matrix," *IRE Trans. Circuit Theory (Correspondence)*, vol. CT-9, pp. 303–306, September 1962.

[25] E. S. Kuh and R. A. Rohrer, "The state-variable approach to network analysis, *Proc. IEEE*, vol. 53, pp. 672–686, July 1965.

[26] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. London: Oxford University Press, 1965.

[27] G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*. Englewood Cliffs, N. J.: Prentice-Hall, 1967.

[28] V. V. Klyuyev and N. I. Kokovkin-Shcherbak, "On the minimization of the number of arithmetic operations for the solution of linear

algebraic systems" (translated by G. J. Tee), Computer Science Dept., Stanford University, Stanford, Calif., Tech. Rept. CS24, 1965.

[29] N. Sato and W. F. Tinney, "Techniques for exploiting the sparsity of the network admittance matrix," *IEEE Trans. Power and Apparatus*, vol. 82, pp. 944–950, December 1963.

[30] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," this issue, p. 1801.

[31] S. Parter, "The use of linear graphs in gauss elimination," *SIAM Rev.*, vol. 3, pp. 119–130, April 1961.

[32] D. Bree, Jr., "Some remarks on the application of graph theory to the solution of sparse systems of linear equations," thesis, Dept. of Math., Princeton University, Princeton, N. J., May 1965.

[33] A. Orden, "Matrix inversion and related topics by direct methods," in *Mathematical Methods for Digital Computers*, A. Ralston and H. S. Wilf, Eds. New York: Wiley, 1960, ch. 2.

[34] F. B. Hildebrand, *Introduction to Numerical Analysis*. New York: McGraw-Hill, 1956.

[35] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Math. Comp.*, vol. 19, pp. 577–593, October 1965.

[36] F. H. Branin, Jr., and H. H. Wang, "A fast reliable iteration method for dc analysis of nonlinear networks," this issue, p. 1819.

[37] J. Katzenelson, "An algorithm for solving nonlinear resistive networks," *Bell Sys. Tech. J.*, vol. 44, pp. 1605–1620, November 1965.

[38] J. G. F. Francis, "The *Q-R* transformation, pt. I," *Computer J.*, vol. 4, pp. 265–271, October, 1961; "The *Q-R* transformation, pt. II," *Computer J.*, vol. 4, pp. 332–345, January 1962.

[39] M. L. Liou, "A numerical solution of linear time-invariant systems," *Proc. 3rd Allerton Conf. on Circuit and System Theory*, pp. 669–676, 1965.

[40] R. W. Stineman, "Digital time-domain analysis of systems with widely separated poles," *J. Assoc. Comp. Mach.*, vol. 12, pp. 286–293, April 1965.

[41] P. I. Richards, W. D. Lanning, and M. D. Torrey, "Numerical integration of large, highly-damped, nonlinear systems," *SIAM Rev.*, vol. 7, pp. 376–380, July 1965.

[42] C. E. Treanor, "A method for the numerical integration of coupled first-order differential equations with greatly different time constants," *Math. Comp.*, vol. 20, pp. 39–45, January 1966.

[43] E. R. Cohen, "Some topics in reactor kinetics," *Proc. 2nd UN Internat'l Conf. on Peaceful Uses of Atomic Energy* (Geneva, Switzerland), vol. 11, pp. 302–309, 1958.

[44] E. R. Cohen and H. P. Flatt, "Numerical solution of quasi-linear equations," *Proc. Seminar on Codes for Reactor Computations* (Vienna, Austria), pp. 461–484, 1960.

[45] J. Certaine, "The solution of ordinary differential equations with large time constants," in *Mathematical Methods for Digital Computers*, A. Ralston and H. S. Wilf, Eds. New York: Wiley, 1960, ch. 11.

[46] J. Katzenelson, "AEDNET: a simulator for nonlinear networks," *Proc. IEEE*, vol. 54, pp. 1536–1552, November 1966.

[47] G. N. Polozhii, *The Method of Summary Representation for Numerical Solution of Problems of Mathematical Physics*. New York: Pergamon, 1965.

[48] *System Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley, 1966.

[49] G. C. Temes and D. A. Calahan," Computer-aided network optimization—The state-of-the-art," this issue, p. 1832.

[50] G. D. Hachtel and R. A. Rehrer, "Techniques for the optimal design and synthesis of switching circuits," this issue, p. 1864.

# Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization

WILLIAM F. TINNEY, SENIOR MEMBER, IEEE, AND JOHN W. WALKER, MEMBER, IEEE

*Abstract*—**Matrix inversion is very inefficient for computing direct solutions of the large sparse systems of linear equations that arise in many network problems. Optimally ordered triangular factorization of sparse matrices is more efficient and offers other important computational advantages in some applications. With this method, direct solutions are computed from sparse matrix factors instead of from a full inverse matrix, thereby gaining a significant advantage in speed, computer memory requirements, and reduced round-off error. Improvements of ten to one or more in speed and problem size over present applications of the inverse can be achieved in many cases. Details of the method, numerical examples, and the results of a large problem are given.**

## I. Introduction

USUALLY, the objective in the matrix analysis of networks is to obtain the inverse of the matrix of coefficients of a system of simultaneous linear network equations. However, for large sparse systems such as occur in many network problems, the use of the inverse is very inefficient. In general, the matrix of the equations formed from the given conditions of a network problem is sparse, whereas its inverse is full. By means of an appropriately ordered triangular decomposition, the inverse of a sparse matrix can be expressed as a product of sparse matrix factors, thereby gaining an advantage in computational speed, storage, and reduction of round-off error.

The method consists of two parts: 1) a scheme of recording the operations of triangular decomposition of a matrix such that repeated direct solutions based on the matrix can be obtained without repeating the triangularization, and 2) a scheme of ordering the operations that tends to conserve the sparsity of the original system. The first part of the method is not altogether new, but its computational possibilities are not widely recognized. The second part appears to be of recent origin, but it is probably only a rediscovery. Either part of the method can be applied independently, but the greatest benefit is obtained from the combined application of both parts.

The first part of the method is applicable to any matrix. Application of the second part, ordering to conserve sparsity, is limited to sparse matrices in which the pattern of nonzero elements is symmetric and for which an arbitrary

Manuscript received February 10, 1967; revised August 9, 1967. The unrevised version of this paper was published in the *Power Industry Computer Applications Conf. Rec.*, pp. 367–376, 1967. This conference was sponsored by the IEEE Power Group.

The authors are with the Bonneville Power Administration, Portland, Ore.