CALCULATION OF LINEAR NETWORKS BY TOPOLOGICAL METHOD

By

I. PÁVÓ

Research Group on Mathemetical Logic and Theory of Automata of the Hungarian Academy of Sciences, Szeged

Presented by Prof. Dr. I. VÁGÓ

Introduction

In the exact computerized solution of linear networks difficulties are due to the need of making node admittance or loop impedance matrices to get the node or loop systems of equations, and of the node or loop determinants and their cofactors for determining the elements of inverse matrix. These difficulties are eliminated by topological formulas. Instead, certain subgraphs of the network have to be generated and the network determinant and its cofactors can be written as the sum of subgraph admittance or impedance products [7]. Since the last decade topological formulas have been playing an increased role in the calculation of active electrical networks. Namely, some topological formulas make it possible to take pairs of nullators and norators into consideration in computing the network models [9].

The complicated character of calculating a network by a topological method depends on the procedure of the generation of network subgraphs needed for the applied formulas. Therefore, the efficiency of planning the network depends on the subgraph generation.

The most important subgraphs are known to be k-trees satisfying certain requirements (k-trees mean a subgraph of the graph which consists of k components and does not contain a circuit). This is why several (about ten) methods have been suggested and used to now to generate k-trees. The procedure described here for generating k-trees is relatively simple and in fact, the obtained k-trees are of the type needed for the topological method, also applicable for calculating the network determinant and its cofactors, illustrated on concrete examples.

Generation of k-trees for network calculation

Practically the k-trees needed for computing networks are either:

(A) k-trees, each component of which contains exactly one of the selected vertices of the network graph (Fig. 1a),

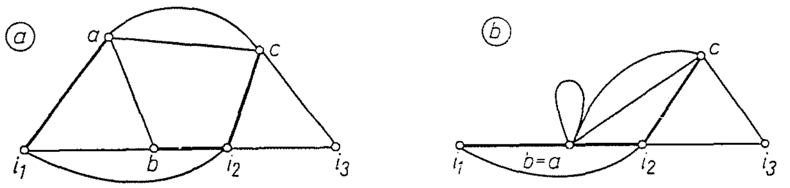


Fig. 1

(B) $(k-r)$-trees which arise by short-circuiting r pairs of vertices from the graphs of type A (Fig. 1b).

The generation of graphs of types A and B is relatively simple, according to the well-known theorem of Ore, ([5] and [6]) such as: Consider a graph G with n vertices, $i_1, \ldots, i_k$ are indexes of the selected vertices of G, $F^k_{i_1,\ldots,i_k}$ is a k-tree of G, each component of which contains exactly one of the selected vertices, then the formula

$$\nu(\mu^{-1}(\mathbf{M}_{i_1,\ldots,i_k})) \tag{1}$$

gives the same subgraphs $F^k_{i_1,\ldots,i_k}$ of G.

In (1) μ denotes the adjacency matrix of a graph (i. e. $\mu(G) = (a_{ij})$, where $a_{ij} = 1$ if there exists an edge in G directing from the i-th vertex to the j-th vertex, otherwise $a_{ij} = 0$), μ^{-1} is the inverse of μ, $\mathbf{M}_{i_1,\ldots,i_k}$ is an $(i_1, \ldots, i_k)$-reduction of $\mu(G)$ (matrix $\mathbf{M}_{i_1,\ldots,i_k}$ arises from $\mu(G)$ so that every element of the i_j-th ($j = 1, \ldots, k$) and all but one non-zero elements of the other rows of $\mu(G)$ are replaced by 0), and finally, the symbol ν denotes the operator which makes a graph undirected.

While matrix $\mathbf{M}_{i_1,\ldots,i_k}$ runs over all the $(i_1, \ldots, i_k)$-reductions of $\mu(G)$, formula (1) produces graph $F^k_{i_1,\ldots,i_k}$ if and only if the complete cycle check performed on the matrix $\mathbf{M}_{i_1,\ldots,i_k}$ is of finite outcome.

Thereby a possible algorithm for generating k-trees of type A results:

I. form the $\mu(G)$ adjacency matrix of G,

II. produce all its $\mathbf{M}_{i_1,\ldots,i_k}$ $(i_1, \ldots, i_k)$-reductions,

III. complete cycle check on the latter, the wanted k-trees arise from $(i_1, \ldots, i_k)$-reductions belonging to the case of finite outcome.

To produce (k-r)-trees of type B, all $F^k_{i_1,\ldots,i_k}$ of G are assumed to have arisen according to the previous method. Now, short-circuit r pairs of vertices are given in advance in every k-tree. It may be proved that a short-circuited k-tree is a $(k-r)$-tree if and only if the (generalized) complete cycle check performed on the earlier graph has a finite outcome and its reduced graph is circuitless. So the algorithm to generate $(k-r)$-trees of type B is the following:

I. Produce short circuited k trees from k-trees of type A by short-circuiting r pairs of vertices, then

II. perform complete cycle check on each short-circuited k-tree produced by the above step, and finally

III. if the complete cycle check has a finite outcome then examine whether the suitable reduced graph has a circuit. In the negative case graphs of type B arise.

To generate k-trees and $(k-r)$-trees, simple computer programmes based on the suggested algorithms can be established, simpler than those based on methods of theory of sets [2], or of tree transformation [4], equal in rank with the simplest algebraic methods [3], besides superior to the latter both by giving the initial data without redundancy and by the needed storage capacity of the programme, a special advantage in the analysis of complicated networks. Besides, the suggested method also lends itself for traditional calculations by its organization in case of simple networks.

Calculation of passive networks

Assume that the network consisting of elements R, L and C contains no ideal voltage generator. So its node potencial system of equations can be written in the following form:

$$I = \mathbf{Y} \cdot V \tag{2}$$

where I is the vector of the ideal current generators with reference direction to the vertices, V is the vector of the node potencials and $\mathbf{Y}$ is the node admittance matrix.

Most problems of network calculation can be reduced to the calculation of det $(\mathbf{Y})$ and det $(\mathbf{Y}_{ij})$ where the latter is the cofactor belonging to the element det $(\mathbf{Y})$ of th i-the row and j-th column. The following topological formulas are known from [7]:

$$\det(\mathbf{Y}) = \sum_{\text{all } F} \left(\prod_{e_k \in F} (y_{e_k}(s)) \right) = \sum F, \tag{3}$$

and

$$\det(\mathbf{Y}_{ij}) = \sum_{\text{all } F^2_{ij,n}} \left(\prod_{e_k \in F^2_{ij,n}} (y_{e_k}(s)) \right) = \sum F^2_{ij,n} \tag{4}$$

In (3) and (4) G is a tree, $F^2_{ij,n}$ is a 2-tree of the network, the latter contains the vertices of index i and j in one vertex of index (the reference vertex), in the other component, e_k is an edge of the suitable subgraph and $y_{e_k}(s)$ is the operator admittance corresponding to the e_k.

Evidently, subgraphs in (2) can be regarded as type A outlined in the previous item. Namely, it is produced by marking one vertex of the network graph (for example the reference vertex) and then testing all trees for containing this vertex in the unit component.But subgraphs in (3) can be easily originated from k-trees type A as well. First produce 2-trees which contain the i-th and the n-th vertices in different components and separate 2-trees which contain the j-th vertex in a common component with the i-th vertex. This selection is not a new procedure, namely the cycle check performed on the j-th vertex (a part step in the complete cycle check) ends at the i-th or n-th vertex. This is the case with 2-trees we are interested in.

Network model containing nullator–norator pairs

It is known that each of the usual building elements of a linear network (except ideal generators) can be modelled with elements R, L and C and with nullator–norator pairs [10]. However, as the calculation of arbitrary linear networks is possible with the use of universal parameters, it is very important to know that any universal parameter of a network model can be produced as its node admittance determinant closing accordingly by a pair of nullator–norator [8]. This determinant is called network determinant. The network determinant can be calculated by the topological formula

$$\det(\mathbf{Y}) = \sum_{\text{all } F^{N+1}} \left(\prod_{e_k \in F^{N+1}} (\pm y_{e_k}(s)) \right) \sum F^{N+1} \tag{5}$$

where det (Y) means the network determinant, N is the number of nullator–norator pairs in the network graph, F^{N+1} is such an $(N+1)$-tree of the graph, from which we get a tree either deleting all the norators and short-circuiting all the nullators or vice-versa (i.e. deleting all the nullators and short-circuiting all the norators in the network-graph). $y_{e_k}(s)$ is the operator admittance of the passive element corresponding to the edge e_k of F^{N+1}. The sign of the admittance product in (5) can be determined by references (for example by the Davies rule, [1]). Notice that (5) can be considered as an extension of formula (2). (5) is also the topological formula of an arbitrary universal parameter if the graph of the network model closed with pairs of nullator–norator is regarded as the network graph.

$(N+1)$-trees in the formula (5) can be derived from k-trees of A and B. Graphs F^{N+1} in (5) hold two requirements, namely they become trees after short-circuiting in two manners. Graphs F^{N+1} generated as $(N+1)$-trees of type A as written before, after short-circuiting in two manners, would be (circuitless) short-circuited $(N+1)$-trees of type B. So graphs F^{N+1} could be generated by the method outlined earlier as well. A simpler procedure for generating $(N+1)$-trees will be shown, involving only graphs types A and B, so graphs F^{N+1} may be generated as described in this paper. Short-circuiting is necessary only once, namely, it is sufficient to make trees type B only once.

Our procedure is the following:

First, edges are left according to nullators in the network graph, then construct the $\mu(G)$ adjacency matrix of the modified network graph.

Further, only trees of G containing exactly N norator edges will be generated. For such a generation only the reductions of $\mu(G)$ have to be taken into consideration which contain excatly N number of elements 1 concerning the norator edges. So the number of the reductions was decreased for the cycle checks of the generation of trees. Be $\mathbf{M}_{i_1}$ a reduction of $\mu(G)$ containing exactly N elements 1 concerning the norator edges and the complete cycle check performed on it is of finite outcome (where i_1 is the index of the row

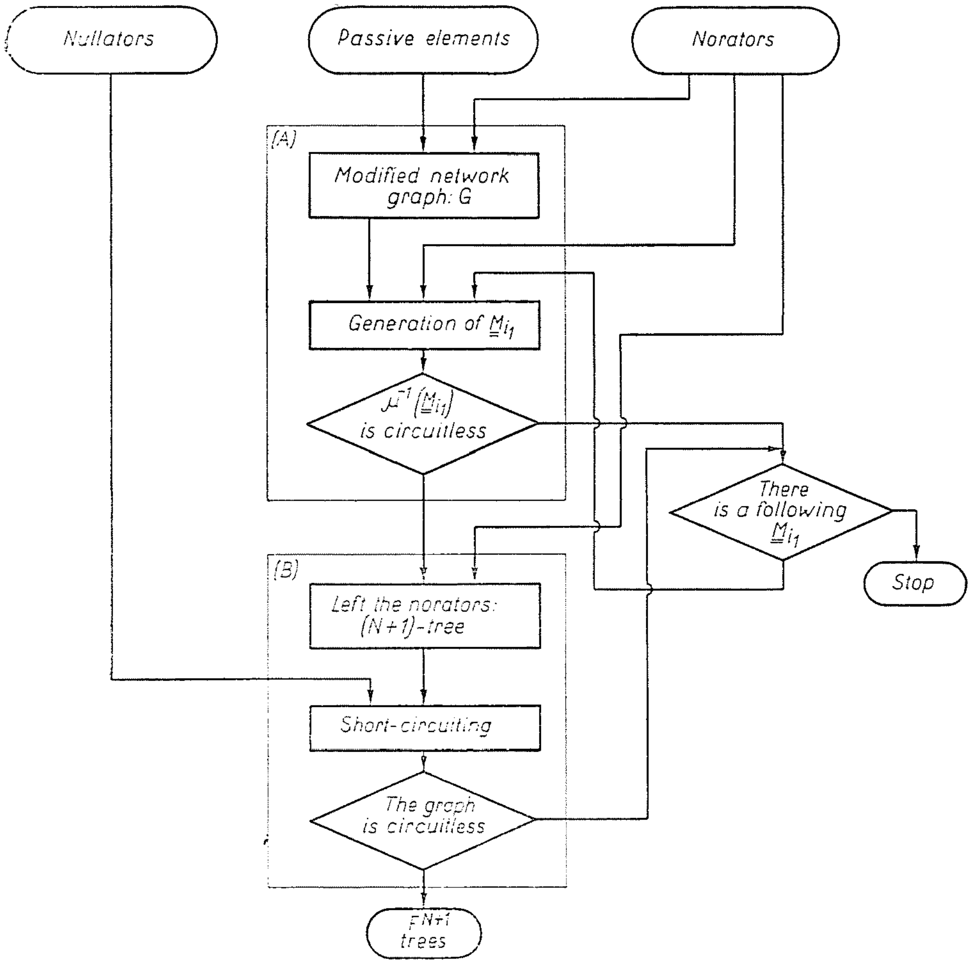

Nullators
Passive elements
Norators
(A)
Modified network graph: G
Generation of $\underline{\underline{M}}_{i_1}$
$\mu^{-1}(\underline{\underline{M}}_{i_1})$ is circuitless
There is a following $\underline{\underline{M}}_{i_1}$
Stop
(B)
Left the norators: (N+1)-tree
Short-circuiting
The graph is circuitless
F^{N+1} trees

Fig. 2

which consists purely of zeros). Then $\mu^{-1}(\mathbf{M}_{i_1})$ is obviously a tree of G which contains all the norator edges, and all similar trees of G arise in this manner. Hereupon replace elements 1 concerning to the norator edges by 0 in $\mathbf{M}_{i_1}$, then the matrix $\mathbf{M}_{i_1,\ldots,i_{N+1}}$ is an $(i_1, \ldots, i_{N+1})$-reduction of $\mu(G)$ so that graph $\mu^{-1}(\mathbf{M}_{i_1,\ldots,i_{N+1}})$ is obviously an $(N+1)$-tree becoming a tree after short-circuiting the norator edges ($i_2, \ldots, i_{N+1}$ mean the row indexes of element 1 according to the norator edges of G). All $(N+1)$-trees of such property can be generated in this way.

Second, let us consider the $(N+1)$-trees generated by the first step and short-circuit the endpoints of the left nullators one by one in each. Among these $(N+1)$-trees those graphs correspond to the searched graphs F^{N+1} which are circuitless after short-circuiting. In fact, only graphs type B need to be generated in the second step.

The flow chart of the algorithm of the generation of graphs F^{N+1} is seen in Fig. 2. The reduction of the generation to production of graph types A and B is illustrated clearly by Fig. 2. The advantage of using a computer is to output graphs F^{N+1} one by one, to print off immediately, managing the storage unit of the computer. (Otherwise, this property of the suggested method follows from the similar property of the procedures for graphs type A and B).

Finally, a dual generation of graphs F^{N+1} results from commuting the role of the norators with that of the nullators in the procedure outlined above.

Application

1. *Transfer impedance function of an RLC network*

Let us show the circuit diagram in Fig. 3, together with the terminating the input, to determine the transfer function.

The transfer impedance is given by formula

$$Z(s) = \frac{U(s)}{I(s)} = \frac{\sum F^{2'}}{\sum F}, \tag{6}$$

where $F^{2'}$ is a 2-tree of the network graph which separates both the input and the output vertices. Graph $F^{2'}$ is a 2-tree affected by a sign. The admittance

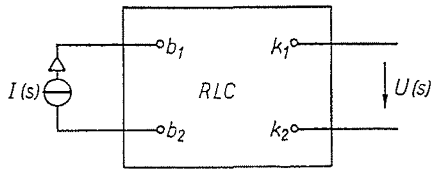


Fig. 3

product corresponding to $F^{2\prime}$ is a positive sign if it contains the b_1-th and k_1-th vertices in a common component, else it is negative (Percival's rule [7]).

The generation of $F^{2\prime}$ together with its sign for (6) can be reduced to generating k-trees of type A by generating first 2-trees $F^2_{b_1,b_2}$, then selecting among them 2-trees according to the conditions. Such a selection may be possible by two cycle checks. Starting a cycle check from the k_1-th vertex and another from the k_2-th vertex, $F^2_{b_1,b_2}$ in question corresponds to a positive 2-tree $F^{2\prime}$ if the first cycle check is finished at the b_1-th vertex, the second at the b_2-th vertex, in case of the reserve outcome $F^{2\prime}$ is negative. In all other cases, $F^2_{b_1,b_2}$ corresponds to no $F^{2\prime}$.

Remark that the cycle checks in question form a part of the complete cycle check performed on $F^2_{b_1,b_2}$, so, during its generation the sign of $F^{2\prime}$ will immediately appear.

A flow chart for computing the transfer impedance is shown in Fig. 4. It is also suitable for constructing a computer programme.

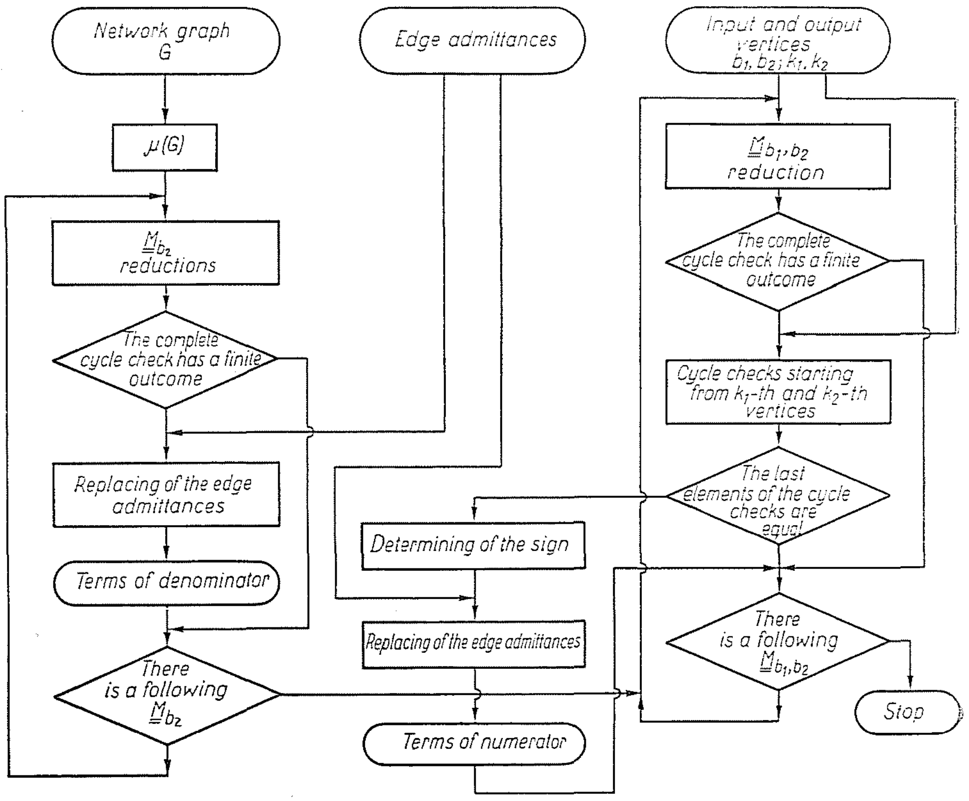

Network graph G
Edge admittances
Input and output vertices $b_1, b_2; k_1, k_2$
$\mu(G)$
$\underline{\underline{M}}_{b_2}$ reductions
The complete cycle check has a finite outcome
Replacing of the edge admittances
Terms of denominator
There is a following $\underline{\underline{M}}_{b_2}$
Determining of the sign
Replacing of the edge admittances
Terms of numerator
$\underline{\underline{M}}_{b_1,b_2}$ reduction
The complete cycle check has a finite outcome
Cycle checks starting from k_1-th and k_2-th vertices
The last elements of the cycle checks are equal
There is a following $\underline{\underline{M}}_{b_1,b_2}$
Stop

Fig. 4

2. *Immittances of a one-terminal-pair network*

The calculation of the operator impedance of a one-terminal-pair network consisting of elements R, L and C can be reduced to the calculation discussed before if we take the opportunity of choosing $b_1 = k_1$ and $b_2 = k_2$, moreover, the operator admittance is the reciprocal of the impedance.

But there is a simpler procedure using the topological formula [7]

$$Z(s) = \frac{\sum F^2_{i,n}}{\sum F} \tag{7}$$

where i and n refer to the two terminals of the network (that is, they are two vertices of the network graph). It is not a problem to generate trees for the denominator of (7) by the former method. Let us modify the former order of the generation of the trees so as to produce also the 2-trees for the nominator of (7). Let us consider all the possible reductions $\mathbf{M}_n$ of the network graph. Let two reductions belong to the same class of $\mathbf{M}_n$, if they agree exactly with its i-th row ($i = 1, \ldots, n - 1$). Then a detailed complete cycle check is performed on the elements of the i_1-th class ($1 \leq i_1 \leq n - 1$). The detailed complete cycle check means a complete cycle check and if it were of infinite outcome, then another complete cycle check is performed on the matrix $\mathbf{M}_{i_1,n}$. The 2-trees in form $\nu(\mu^{-1}(\mathbf{M}_{i_1,n}))$ for (7) arise from all the $\mathbf{M}_n$ reductions which belong to the i_1-th class, and any of the two complete cycle checks of the detailed complete cycle check performed on them has a finite outcome. Naturally, to generate trees for (7), only a complete cycle check has to be performed on the elements $\mathbf{M}_n$ of the other classes.

To illustrate the method, let us consider the network in Fig. 5 and calculate its driving-point impedance.

The generating matrix M_G of the network [5] has the form:

$$M_G = \begin{bmatrix} 0 & 2 & 0 & 4 & 0 \\ 1 & 0 & 3 & 0 & 5 \\ 0 & 2 & 0 & 4 & 5 \\ 1 & 0 & 3 & 0 & 5 \\ 0 & 2 & 3 & 4 & 0 \end{bmatrix} \tag{8}$$

Table 1 contains all the reductions $\mathbf{M}_5$ as row vector representations belonging to either of two classes. In column k the number marks the class of the reduction, while the number in column i is the serial number of the row vector representation in question.

As it appears from Table 1, the row vector representations are deleted if element 5 or (in the first class) element 1 is missing, as under such circum-

Table I

k	i	Row vector representation	Tree	2-tree	Term of numerator		Term of denominator	
					coeff.	degree of s	coeff.	degree of s
1	1	21210		+	1	—1		
	2	21230		+	1	—1		
	3	21250		+	½	—3		
	4	21410		+	1	1		
	5	21430						
	6	21450		+	½	—1		
	7	21510		+	1	0		
	8	21530		+	1	0		
	9	21550		+	½	—2		
	10	23210						
	12	23250						
	13	23410		+	1	1		
	15	23450	+	+	½	—1	½	—2
	16	23510	+	+	1	0	1	—1
	17	23530	+	+	1	0	1	—1
	18	23550	+	+	½	—2	½	—3
	19	25210	+	+	2	1	2	0
	20	25230	+	+	2	1	2	0
	21	25250	+	+	1	—1	1	—2
	22	25410	+	+	2	3	2	2
	23	25430						
	24	25450	+	+	1	1	1	0
	25	25510	+	+	2	2	2	1
	36	25530	+	+	2	2	2	1
	27	25550	+	+	1	0	1	—1
2	3	41250	+				½	—2
	6	41450	+				½	0
	7	41510						
	8	41530	+				1	1
	9	41550	+				½	—1
	12	43250						
	15	43450	+				½	0
	16	43510						
	17	43530	+				1	1
	18	43550	+				½	—1
	19	45210						
	20	45230	+				2	2
	21	45250	+				1	0
	22	45410						
	23	45430						
	24	45450	+				1	2
	25	45510						
	26	45530	+				2	3
	27	45550	+				1	1

stances the complete cycle check performed on the corresponding representations has no finite outcome. So 44 representations remained. Performing a detailed complete cycle check on the suitable reductions of the row vector representation belonging to the 1-st class, marks "+" in the 4-th and the

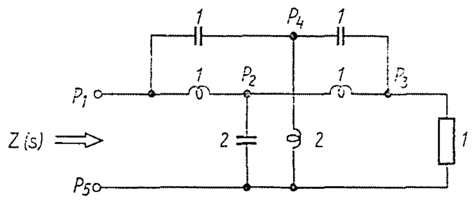


Fig. 5

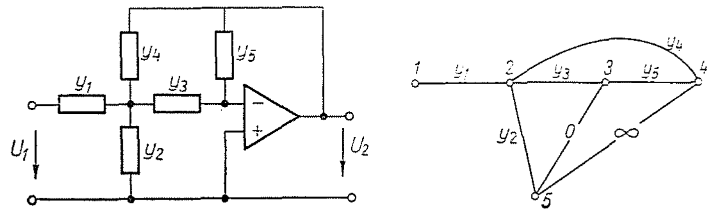


Fig. 6

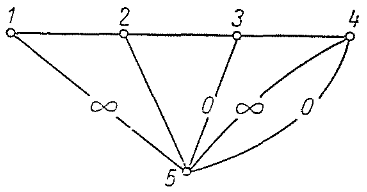


Fig. 7

5-th columns of Table 1 show which are the representations resulting in trees or in 2-trees. The 5-th and 7-th columns of Table 1 contain the suitable tree and 2-tree admittance products. To control them, use the "rules of the displacement" which can be immediately read off Fig. 5 and (8):

The										
1^{st}		2^{nd}			3^{rd}			4^{th}		
component of the row vector representation with element										
2	4	1	3	5	2	4	5	1	3	5
corresponds to factor										
s^{-1}	s	s^{-1}	$2s$	$2s$	s^{-1}	s	1	s		$1/2s^{-1}$
in the tree (or 2-tree) admittance product										

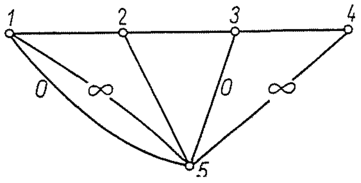


Fig. 8

Finally, considering (7) and Table 1, after some calculation we get:

$$Z(s) = \frac{4s^6 + 8s^5 + 14s^4 + 10s^3 + 8s^2 + 2s + 1}{4s^6 + 10s^5 + 14s^4 + 4s^3 + 8s^2 + 4s + 1}.$$

Remark that the calculation by the node potential system of equations or using two terminal-pair parameters or after conversion of the network is so tedious that the calculation by topological formulas is almost simpler. This method is however very elegant for computer analyzis of a complicated network.

3. *Active network containing operational amplifier*

Let us determine the transfer voltage function of the network in Fig. 6. The (ideal) operational amplifier simulated by a nullator–norator pair is shown together with the network model in Fig. 6 as well.

Namely:

$$\frac{U_2}{U_1} = \frac{B_u}{P_u} \tag{9}$$

where B_u and P_u are universal parameters of the network [8] to be determined.

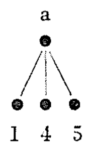
To calculate B_u, close the network model on the input side with a norator, on the output side with a nullator. Fig. 7 shows the modified network model which has the following (modified) adjacency matrix:

$$\mathbf{M}_B = \begin{bmatrix} 0 & 2 & 0 & 0 & 5_\infty \\ 1 & 0 & 3 & 4 & 5_0 \\ 0 & 2 & 0 & 4 & 5 \\ 0 & 2 & 3 & 0 & 5_{0,\infty} \\ 1_\infty & 2 & 3_0 & 4_{0,\infty} & 0 \end{bmatrix} \tag{10}$$

where indexes "0" and "∞" refer to the presence of a nullator and a norator between two corresponding nodes of the network model.

According to the above method, the further procedure of calculation is recapitulated in Table 2. Its column marked $\mathbf{M}_5'$ contains all the (5)-reductions generated from (10) which include both norators of the modified network model. The column marked "tree" summarizes the outcomes of the complete

Table 2

M_5' representation	Tree	3-tree	3=4=5=a 12aaa	Circuit-less	Reduced graph	B_u
$5_\infty\ 125_\infty\ 0$	y	01200	01200	y		y_1y_3
$5_\infty\ 145_\infty\ 0$	y	01400	01a00	n		—
$5_\infty\ 325_\infty\ 0$	n	—	—	—		—
$5_\infty\ 345_\infty\ 0$	y	03400	0aa00	n		—
$5_\infty\ 425_\infty\ 0$	y	04200	0a200	n		—
$5_\infty\ 445_\infty\ 0$	y	04400	0aa00	n		—
$5_\infty\ 525_\infty\ 0$	y	05200	0a200	n		—
$5_\infty\ 545_\infty\ 0$	y	05400	0aa00	n		—

cycle checks performed on the row representations in question, symbols "y" or "n" indicating that the corresponding graph is circuitless or not. The 3-rd column of Table 2 contains representations of the 3-trees which arise from the earlier graphs by cmitting the norator edges. In the further columns the circuitless short-circuited 3-trees are calculated (actually there is a single one). Going into details, the 4-th and the 5-th column show the short-circuited condition and the corresponding reduced graph (which is obviously circuitless) and finally, we can see the (unique) term of B_u which has a positive sign because of the Davies rule.

P_u will be obtained in accordance with the closed network model in Fig. 8, where from the (modified) adjacency matrix:

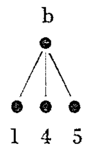
$$\mathbf{M}_P = \begin{bmatrix} 2 & 0 & 0 & 0 & 5_{0,\infty} \\ 1 & 0 & 3 & 4 & 5 \\ 0 & 2 & 0 & 4 & 5_0 \\ 0 & 2 & 3 & 0 & 5_0 \\ 1_{0,\infty} & 2 & 3_0 & 4_\infty & 0 \end{bmatrix} \tag{11}$$

(5)-reductions $\mathbf{M}_5''$ constructed with (11) and containing all the norators of the modified network model are seen to coincide with the reductions in the 1-st column of Table 2, the 2-nd and 3-rd columns of Table 2 are true as well. But the condition of short-circuiting changes, so the corresponding short-circuited 3-trees are different as well. The new condition of short-circuiting, the corresponding short-circuitless 3-trees and the calculation of the terms P_u have been compiled in Table 3. All terms of P_u are stated in [1] to be negative.

According to the last columns of Tables 2, 3 and formula (9), the transfer voltage function of the (original) network is:

$$\frac{U_2}{U_1} = -\frac{y_1y_3}{y_5(y_1+y_2+y_3+y_4)+y_3y_4}\,.$$

Table 3

1=3=5=b b2b4b	Circuit-less	(Common) reduced graph	Terms of P_u
0b200	n		—
0b400	y		$-y_1y_5$
—	—		—
0b400	y		$-y_3y_5$
04200	y		$-y_4y_3$
04400	y		$-y_4y_5$
0b200	n		—
0b400	y		$-y_2y_5$

Summary

For computing linear networks by topological formulae, including nullator-pair (active) network models, k-trees of the network graph are needed. These can be advisably generated by methods previously suggested by the author, or by their combination, resulting in a homogeneous topology method for the design of (active and passive) linear networks. The presented method lends itself for the computerized solution of network function formulae with letter coefficients, requiring a minimum of operative strorage capacity, and even facilitating manual calculation. Its application is illustrated by means of numerical examples.

References

1. Davies, A. C.: Norator-Nullator Equivalent Networks for Controlled Sources. Proc. of IEEE 1967, pp. 722—723.
2. Hakimi, S. L.—Green, D. G.: Generation and Realization of Trees and k-Trees. IEEE Trans. on Circuit Theory 1964, pp. 247—255.
3. Maxwell, L. M.—Cline, Jr. J. M.: Topological Network Analysis by Algebraic Methods. Proc. of IEEE 1966, p. 1344—1437.
4. Mayeda, W.—Seshu, S.: Generation of Trees Without Duplications: IEEE Trans. on Circuit Theory 1965, pp. 181—185.
5. Pávó, I.: Generation of the k-Trees of a Graph. Acta Cybernetica, Szeged, 1972, Tom. 1, Fasc. 2, pp. 57—68.
6. Pávó, I.: Short-Circuited k-Trees. Acta Cybernetica, Szeged. 1976, Tom. 2, Fasc. 4.
7. Seshu, S.—Reed, M. B.: Linear Graphs and Electrical Networks. Addison-Wesley, London, 1961.
8. Szepesi, T.—Guttermuth, M.: Topological Analysis of Linear Active Networks (in Hungarian). Híradástechnika, Budapest, 1972, XXIII., No. 2. pp. 56—61.
9. Vágó, I.: Calculation of Networks Models Containing Nullators–Norators. Periodica Polytechnica Electr. Eng. Budapest, 17 (1973), pp. 311—319.
10. Vágó, I.—Hollós, E.: Two-Port Models with Nullators and Norators. Periodica Polytechnica Electr. Eng. Budapest, 17, (1973) pp. 302—309.

Dr. Imre Pávó, H-6720 Szeged, Somogyi u. 7. Hungary