The inversion of sparse matrices by a strategy derived from their graphs

By A. Nathan and R. K. Even*

An algorithm is derived for the inversion of a matrix which makes use of the structure of the associated flow graph. The flow graph is explored with the help of Boolean matrices in order to determine an efficient strategy for successive elimination of variables. The algorithm reduces the given matrix to a triangular one of lower order which is readily inverted.

1. Introduction

The solution of a set of simultaneous linear algebraic equations is occasionally aided by an examination of an associated graph. Thus Mason (1953, 1955) associates a "signal flow graph" with such a set, and Coates (1959) similarly defines his "flow graph", and both give equivalent (Desoer, 1960) topological rules for their solution. Harary (1959, 1962) provides an algorithm for the partitioning of matrices in order to simplify their inversion. This algorithm is limited to directed graphs which have weakly connected components, and allows only identical permutations of rows and columns thereby preserving the eigenvalues of the matrix. Dulmage and Mendelsohn (1962) have replaced the directed by a bipartite graph, and thus solve a stronger partitioning problem. Parter (1961) shows how Gauss elimination may occasionally be made more efficient by a prior examination of a graph, while Steward (1962) examines a table which essentially shows the topology of a graph in order to determine the most efficient sequence of elimination. Steward has overlooked the fact that a purely topological algorithm cannot be followed through; rather at each step numerical values must be considered for (using our terminology) zero-weight branches may appear. In order to continue, it is then necessary to change the graph into one of different topology.

The algorithm proposed in this paper, while efficient only for sparse matrices, applies in general and leads to the inversion of any non-singular matrix. It is based on Nathan's (1961) converging algorithm for the solution of signal-flow-graphs which consists basically of a succession of cycles of two simple steps. Step 1 requires the identification of a set of principal nodes; step 2 eliminates several nodes at a time if the graph is sufficiently sparse. Each cycle reduces both the order and the sparseness of the graph until a graph corresponding to a lower triangular matrix remains. Nathan's algorithm is primarily intended for manual computation, whereas here we have machine computation in mind and use Boolean adjacency matrices (Hohn and Schissler, 1955; Seshu and Reed, 1961) for the exploration of the graph and for the identification of suitable sets of principal nodes.

The number of operations required by the proposed algorithm is, in general, unknown, since it depends on the topology of the graphs appearing in consecutive cycles. In specific classes of problems much computational effort can be saved. Thus, Even and Wallach (1966) apply the method to elliptic difference equations.

It appears that efficient strategies can be worked out for further classes of problems, e.g., for the solution of electrical networks or for the inversion of blockwise k-diagonal matrices.

2. Matrices and flow graphs

Let $M = [m_{ij}]$ be a non-singular matrix of order n in which no principal diagonal element vanishes, i.e. $m_{ii} \neq 0$, $\forall i$.

We prove that there is no loss in generality, since any non-singular matrix can be brought to this form by at most a permutation of rows. Consider $|M|$, the determinant of M. $|M|$ is a sum of products, each of n entries in M, no two of which are in the same row or column. Since $|M| \neq 0$, at least one of these products does not vanish, and there exists a permutation of the rows of M so that the factors of this product lie on the principal diagonal. Hall's (1956) algorithm may be used to find the permutation.

Denoting the columns of the inverse of M, $M^{-1} = [x_{ij}]$, by X_j $(j = 1, 2, \ldots, n)$, so that $M \cdot [X_1, X_2, \ldots, X_n] = I = [I_1, I_2, \ldots, I_n]$ and

$$MX_j = I_j \; (j = 1, 2, \ldots, n), \tag{1}$$

where I_j is the jth column of unit matrix I, eqn. (1) may be written in the form

$$T_j Y_j = 0 \quad (j = 1, 2, \ldots, n), \tag{2}$$

where

$$T_j = [t_{kl}^{(j)}] = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \quad 2 \ldots \; n \end{array} \\ \begin{bmatrix} 0 & 0 \\ -I_j & M \end{bmatrix} \end{array} ; \; Y_j = \begin{bmatrix} 1 \\ \cdots \\ X_j \end{bmatrix}. \tag{3}$$

* *Department of Electrical Engineering, Technion—Israel Institute of Technology.* This research was supported in part by the Gerard Swope Trust Fund.

Following Coates (1959), a flow graph (F.G.) is associated with T_j and eqns. (2) in a one-one manner as follows:

(*a*) The F.G. has a source node 0 and n nodes $1 \ldots n$.

(*b*) To each non-vanishing entry $t_{kl}^{(j)}$ of T_j there corresponds a directed branch from node l to node k of weight $t_{kl}^{(j)}$. $t_{kk}^{(j)}$ corresponds to a "self-loop".

(*c*) The variable x_{ij} is associated with node i $(i = 1, 2, \ldots, n)$ and the value $x_{0j} = 1$ with source node 0.

(*d*) The F.G. induces the constraints

$$\sum_{l=0}^{n} t_{kl}^{(j)} x_{lj} = 0 \quad (k = 1, 2, \ldots, n) \tag{4}$$

i.e., the weighted inflowing variables add up to zero at each of nodes $k = 1, 2, \ldots, n$.

Two facts are to be noted:

(i) M corresponds in a one-one manner to the sub-graph of the F.G. formed by omitting source-node 0 and the branches emanating from it.

(ii) There is a self-loop at each of nodes $1, \ldots, n$ because $t_{kk}^{(j)} \neq 0$ $(k = 1, \ldots, n)$.

Solving the F.G. means finding $x_{lj} (l = 1, 2, \ldots, n)$, and this is usually done by "node elimination". The interested reader is referred back to Mason (1953, 1955) or Coates (1959). Fact (ii) means that any node $1, 2, \ldots, n$ can be eliminated from the F.G.

3. The principal node algorithm for F.G. solution

Step by step node elimination is a laborious method for the determination of X_j, the jth column of M^{-1}. If M is a sparse non-singular matrix, it is much faster to reduce the associated F.G. by the following algorithm (Nathan, 1961).

Let G be the F.G. corresponding to eqns. (2). Denote as a "proper loop" a loop that passes through each of its nodes only once and is not a self-loop. Then:

(*a*) In G, select a set of "principal nodes" P which includes source node 0 and further nodes so as to break all proper loops, i.e. so that any proper loop passes through at least one node $\in P$.

(*b*) Eliminate all nodes $\notin P$. Denote the resulting F.G. by $G^{(1)}$. Because any proper loop in G contains at least one node $\in P$, the eliminated variables are readily expressed in terms of those associated with the nodes $\in P$.

Repeat steps (*a*) and (*b*) to obtain $G^{(2)}, G^{(3)}, \ldots$, until no proper loops are left. If required by the disappearance of self-loops in some $G^{(\nu)}$, rearrange the sequence of the corresponding equations so that $M^{(\nu)}$ is transformed into $M^{(\nu)*}$ having non-vanishing elements on its diagonal (cf. Section 2), and continue with the associated F.G., $G^{(\nu)*}$, in lieu of $G^{(\nu)}$.

For the inversion of M, of order n, j must assume all values $1, \ldots, n$. A change in the value of j changes only the 0th column of T_j (cf. eqns. (3)), i.e. the branch emanating from source-node 0 in the F.G. flows into a different node. No loop passes through the source node and thus the successive sets of principal nodes and the necessary row permutations are independent of j.

For the algorithm to progress, it is sufficient that the sets of principal nodes have successively smaller numbers of nodes. This can certainly be achieved provided that in any graph containing at least one proper loop $\exists P$ that does not include all nodes. But evidently, omitting one of the nodes of a loop generates an admissible P.

We now give an algebraic formulation of the algorithm. Step 1 determines a set of variables ("pivots") that are to be eliminated in step 2: they are the variables associated with the non-principal nodes.

After ν such cycles have been applied to eqns. (1), one is left with

$$M^{(\nu)} X_j^{(\nu)} = N_j^{(\nu)} \tag{5}$$

where $M^{(\nu)}$ is a non-singular matrix of order n_ν.

Applying next step 1 of the algorithm to eqns. (5), the $n_{\nu+1}$ "principal variables" are determined and the equations are rearranged as

$$\begin{matrix} \\ n_{\nu+1} \end{matrix} \overbrace{\begin{matrix} \\ \end{matrix}}^{n_{\nu+1}} \left\{ \begin{bmatrix} M_a^{(\nu)} & M_b^{(\nu)} \\ M_c^{(\nu)} & M_d^{(\nu)} \end{bmatrix} \cdot \begin{bmatrix} X_j^{(\nu+1)} \\ X_{jb}^{(\nu)} \end{bmatrix} = \begin{bmatrix} N_{ja}^{(\nu)} \\ N_{jb}^{(\nu)} \end{bmatrix}, \right. \tag{6}$$

where $X_{jb}^{(\nu)}$ is a $(n_\nu - n_{\nu+1}) \times 1$ vector of pivots and $X_j^{(\nu+1)}$ consists of the principal variables.

Elimination of $X_{jb}^{(\nu)}$ from eqns. (6) is carried out as follows:

$$X_{jb}^{(\nu)} = [M_d^{(\nu)}]^{-1} N_{jb}^{(\nu)} - [M_d^{(\nu)}]^{-1} M_c^{(\nu)} X_j^{(\nu+1)}; \tag{7}$$

$$M^{(\nu+1)} X_j^{(\nu+1)} = N_j^{(\nu+1)}, \tag{8}$$

where

$$M^{(\nu+1)} = M_a^{(\nu)} - M_b^{(\nu)} [M_d^{(\nu)}]^{-1} M_c^{(\nu)}; \tag{9}$$

$$N_j^{(\nu+1)} = N_{ja}^{(\nu)} - M_b^{(\nu)} [M_d^{(\nu)}]^{-1} N_{jb}^{(\nu)}. \tag{10}$$

If eqns. (5) are suitably ordered, no principal diagonal entry in $M^{(\nu)}$ vanishes. $M_d^{(\nu)}$ corresponds to a sub-graph (of the F.G. corresponding to $M^{(\nu)}$) that has no proper loops, which means that its rows and columns can be subjected to identical permutations in order to bring $M_d^{(\nu)}$ to a lower triangular form. Since such permutations do not change the product of the principal diagonal entries, $M_d^{(\nu)}$ is non-singular, and the elimination can always be carried out.

4. Example

A simple example will clarify the algorithm. The fifth column of the inverse of the coefficient matrix in eqns. (11) is to be computed:

$$\left[\begin{array}{ccc:ccc} a_1 & c & 0 & 0 & 0 & e \\ b & 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & a_3 & 0 & g & 0 \\ \hdashline 0 & 0 & 1 & a_4 & 1 & 0 \\ 0 & 0 & f & 0 & 1 & 0 \\ d & -1 & 0 & 0 & 1 & 1 \end{array}\right] \left[\begin{array}{c} x_{15} \\ x_{25} \\ x_{35} \\ \hdashline x_{45} \\ x_{55} \\ x_{65} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hdashline 0 \\ 1 \\ 0 \end{array}\right]. \tag{11}$$

Eqns. (11) are our eqns. (1) with $j = 5$, $n = 6$. The corresponding F.G. is given in **Fig. 1** where the principal nodes (the choice of which is not unique) are encircled.

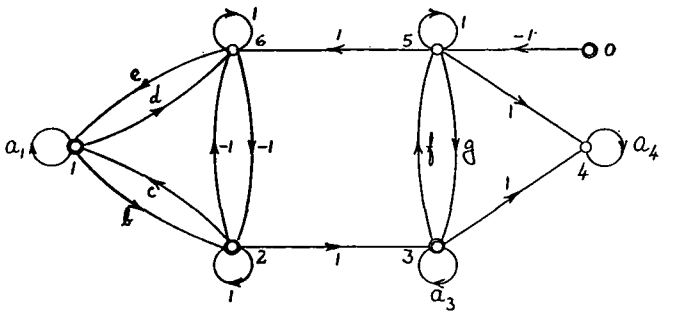

Fig. 1. Flow graph of the example

The matrices in eqns. (11) are partitioned according to the elimination scheme dictated by this principal node set.

Using eqns. (7) through (10), we eliminate the last three variables in eqns. (11) as follows:

$$\begin{bmatrix} x_{45} \\ x_{55} \\ x_{65} \end{bmatrix} = \begin{bmatrix} -\frac{1}{a_4} \\ 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & \frac{1-f}{a_4} \\ 0 & 0 & f \\ d & -1 & -f \end{bmatrix} \begin{bmatrix} x_{15} \\ x_{25} \\ x_{35} \end{bmatrix}; \quad (12)$$

$$M^{(1)} = \begin{bmatrix} a_1-de & c+e & ef \\ b+d & 0 & -f \\ 0 & 1 & a_3-fg \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & 0 & -f \\ 0 & 1 & b_5 \end{bmatrix}; \quad N_5^{(1)} = \begin{bmatrix} e \\ -1 \\ -g \end{bmatrix}. \quad (13)$$

Since $m_{22}^{(1)} = 0$, we permute rows 2 and 3 in $M^{(1)}$ and $N_5^{(1)}$. Subsequently we bring row and column 3 in $M^{(1)}$ and row 3 in $N_5^{(1)}$ into leading positions, in accordance with the F.G. of **Fig. 2.**

$$\left[\begin{array}{c|cc} -f & b_4 & 0 \\ \hline b_3 & b_1 & b_2 \\ b_5 & 0 & 1 \end{array}\right] \left[\begin{array}{c} x_{35} \\ \hline x_{15} \\ x_{25} \end{array}\right] = \left[\begin{array}{c} -1 \\ \hline e \\ -g \end{array}\right] \quad (14)$$

The variables x_{15} and x_{25} are now to be eliminated by the same procedure:

$$\begin{bmatrix} x_{15} \\ x_{25} \end{bmatrix} = \begin{bmatrix} \frac{e-b_2g}{b_1} \\ -g \end{bmatrix} - \begin{bmatrix} \frac{b_3-b_2b_5}{b_1} \\ b_5 \end{bmatrix} x_{35}; \quad (15)$$

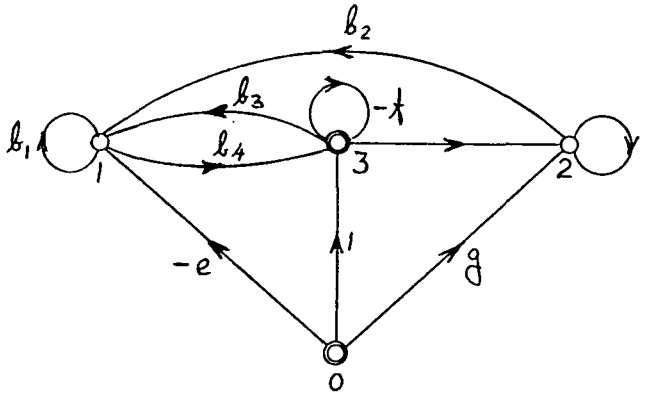

Fig. 2. The rearranged F.G. of Fig. 1 after nodes 4 through 6 have been eliminated

$$\left.\begin{aligned} M^{(2)} &= -f - \frac{b_4}{b_1}(b_3 - b_2b_5) = c_1; \\ N_5^{(2)} &= -1 - \frac{b_4}{b_1}(e - b_2g) = c_2; \end{aligned}\right\} \quad (16)$$

and hence

$$c_1x_{35} = c_2. \quad (17)$$

Having found the value of x_{35}, we substitute it in eqns. (15) to find the values of x_{15} and x_{25}. Subsequently, x_{45}, x_{55} and x_{65} are computed from eqns. (12).

The computation of any other column of M^{-1} follows the same steps, but only the matrices $N_j^{(1)}$ and $N_j^{(2)}$ must be recomputed.

5. Determination of principal node sets

5.1. Definitions of topological matrices

We require an algorithm for the determination of principal node sets, and shall explore the structure of F.G.'s by means of topological Boolean matrices.

Let G be a directed graph. With G we associate square Boolean matrices whose elements are 0 or 1, and whose rows and columns correspond to the nodes of G. For the elements we have the operations of

union: $1 \cup 1 = 1 \cup 0 = 0 \cup 1 = 1, 0 \cup 0 = 0$

and intersection:

$$1 \cap 1 = 1; 1 \cap 0 = 0 \cap 1 = 0 \cap 0 = 0.$$

Similarly, for matrices,

union: $A = B \cup C \longleftrightarrow a_{ij} = b_{ij} \cup c_{ij}$;

intersection: $A = BC \longleftrightarrow a_{ij} = \cup_k (b_{ik} \cap c_{ki})$.

We define:

(*a*) The "k-connectivity matrix" $C^{(k)} = [c_{ij}^{(k)}]$ is defined by

$$c_{ij}^{(k)} = \begin{cases} 1 \text{ if } i \neq j \text{ and } \exists \text{ in } G \text{ a path from } i \text{ to } j \\ \quad \text{of length } k; \\ 0 \text{ otherwise.} \end{cases}$$

$C^{(1)} = C$ is the "connectivity matrix" of G.

(*b*) The "k-adjacency matrix" $A^{(k)} = [a_{ij}^{(k)}]$ is defined by

$$A^{(k)} = C^{(k)} \cup I.$$

$A^{(1)} = A$ is the "adjacency matrix" of G.

(*c*) Matrix $B^{(k)} = [b_{ij}^{(k)}]$ is defined by

$$b_{ij}^{(k)} = \begin{cases} 1 \text{ if } i \neq j \text{ and } \exists \text{ in } G \text{ a path from } i \text{ to } j \text{ of} \\ \text{length} \leqslant k; \\ 0 \text{ otherwise.} \end{cases}$$

(*d*) A^k, B^k, and C^k denote the kth power of the corresponding matrices.

(*e*) G_k denotes a graph that does not contain any proper loops of length $\leqslant k$. Thus any G is a G_1.

(*f*) In these definitions paths are simple, i.e. they include no loops.

5.2. Properties of the matrices

(*a*) If $i \neq j$: $(A^t)_{ij} = 1 \longleftrightarrow \exists$ in G a path from i to j of length $\leqslant t$.

$$\text{E.g. } (A^2)_{ij} = \bigcup_{u=1}^{n} (a_{iu} \cap a_{uj})$$

which is 1 if and only if, for at least one value of u, $a_{iu} = a_{uj} = 1$.

Note that $a_{ii} = 1$, $\forall i$, and, more generally, that

$$(A^s)_{ij} = 1 \longrightarrow (A^t)_{ij} = 1, \quad t \geqslant s.$$

This follows from

$$(A^{s+1})_{ij} = \bigcup_{u=1}^{n} [(A^s)_{iu} \cap a_{uj}]$$

$$= \left\{ \bigcup_{u \neq j} [(A^s)_{iu} \cap a_{uj}] \right\} \cup [(A^s)_{ij} \cap a_{jj}]$$

$$= \left\{ \bigcup_{u \neq j} [(A^s)_{iu} \cap a_{uj}] \right\} \cup (A^s)_{ij},$$

which is equal to 1 if $(A^s)_{ij} = 1$.

(*b*) $B^{(k)}$ is produced from A^k by the substitution of 0 for all principal diagonal elements in A^k.

(*c*) If $G = G_k$: $(C^t)_{ij} = 1$, $t \leqslant k \longleftrightarrow \exists$ in G a path from i to j of length t. It follows that in G_k, $C^{(t)} = C^t$, $t \leqslant k$.

The theorem is evidently true for $k = 2$, and the proof follows by induction. Assume that in G_k, $C^{(t-1)} = C^{t-1}$. Then

$$(C^t)_{ij} = \bigcup_{u=1}^{n} [(C^{t-1})_{iu} \cap c_{uj}] = \bigcup_{u=1}^{n} [(C^{(t-1)})_{iu} \cap c_{uj}]$$

and since, in G_k, $(C^{(t-1)})_{ii} = 0$, $\forall i$ when $t \leqslant k$, we have

$$(C^t)_{ij} = \bigcup_{u \neq i,j} [(C^{(t-1)})_{iu} \cap c_{uj}].$$

Hence $(C^t)_{ij} = 1 \longleftrightarrow \exists$ in G_k at least one value of u for which $(C^{(t-1)})_{iu} = c_{uj} = 1$, i.e. $\exists$ a path from i to u of length $t - 1$ and a path from u to j of length 1.

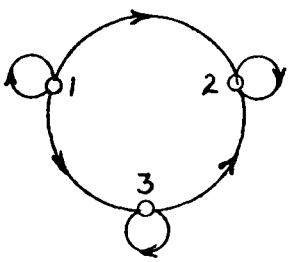

Fig. 3

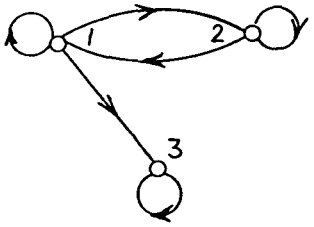

Fig. 4

As an illustration consider the graphs of Figs. 3 and 4, which are, respectively, a G_3 and a G_1. Using the subscript 1 in G_1 and 3 in G_3, we have

$$C_3 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; \quad A_3 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix};$$

$$C_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \quad A_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C_3^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \quad A_3^2 = A_3;$$

$$C_1^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}; \quad A_1^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$C_3^q = 0$, $q \geqslant 3$ $A_3^q = A_3$, $q \geqslant 1$.

$C_1^{2q} = C_1^2$; $C_1^{2q+1} = C_1$, $q \geqslant 1$; $A_1^q = A_1^2$, $q \geqslant 2$.

From $C_3^3 = 0$ it follows that $C_3^{(3)} = 0$ and therefore that there are no paths of length 3 in Fig. 3, but $C_1^3 \neq 0$ does not indicate that there are any in Fig. 4.

(*d*) Let

$$d_t^{(r,s)} = [C^{(r)} B^{(s)}]_{tt} = \bigcup_{u=1}^{n} [c_{tu}^{(r)} \cap b_{ut}^{(s)}]. \tag{18}$$

Then, if $G = G_k$ and $1 \leqslant r, s \leqslant k$,

$d_t^{(r,s)} = 1 \longleftrightarrow \exists$ in G a proper loop of length $r + 1$, $r + 2, \ldots$, or $r + s$ passing through node t.

Indeed, since $G = G_k$, any contiguous sequence of $1, 2, \ldots$, or k branches in G, excluding self-loops, forms a (simple) path. It follows that $d_t^{(r,s)} = 1 \longleftrightarrow \exists$ in G a path tu of length r and a path ut of length $1, 2, \ldots$, or s. But tut forms a proper loop of the stated length.

In particular, $d_t^{(k,k)} = d_t^{(2k)} = 1 \longleftrightarrow \exists$ in G a proper loop of length $k + 1$, $k + 2, \ldots$, or $2k$ passing through node t.

Property (*d*) is useful for the identification in G_k of nodes taking part in loops of length $k + 1$, $k + 2, \ldots$, or $2k$.

5.3. The determination of principal node sets

Any F.G. is necessarily a G_1 in the sense of definition (*e*). In order to determine a set P of principal nodes in

G_1, we first determine a subset $P_2 \subset P$ which breaks all loops of length 2. Next, P_2 is "removed" from G_1 by omitting from it all nodes $\in P_2$ and all the branches in G_1 connected with them. The residual graph is a G_2. In G_2, subset $P_4 \subset P$, which breaks all loops of length 3 or 4, is determined. Removal of P_4 from G_2 produces G_4. The pattern is continued, producing

$$P_2 \rightarrow G_2 \rightarrow P_4 \rightarrow G_4 \rightarrow P_8 \rightarrow G_8 \rightarrow P_{16} \ldots$$

until a graph with no proper loops is reached. Then $P = P_2 \cup P_4 \cup P_8 \cup \ldots$. The kth cycle produces $P_{(2^k)}$ and $G_{(2^k)}$ and the sequence is broken off as soon as k obeys

(number of nodes in G) $- s \leqslant 2^k$; s = number of nodes in $P_2 \cup P_4 \cup \ldots \cup P_{(2^k)}$.

The process necessarily leads to a set P containing fewer nodes than G, and this ensures the convergence of the principal node algorithm.

Since $C^{(1)} = B^{(1)} = C$, P_2 is determined in G_1 through the computation of

$$d_t^{(2)} = [CC]_{tt}, \tag{19}$$

cf. eqn. (18). $d_t^{(2)}$ is computed for $t = 1, 2, \ldots$ until the first value, $t = t_1$ say, for which $d_t^{(2)} = 1$ is reached. Then $t_1 \in P_2$. Node t_1 is removed from G_1. Removal of t_1 means the deletion of the corresponding row and column in C. The process is repeated with the new connectivity matrix, yielding t_2, and subsequently $t_3, \ldots$, until no loops of length 2 are left.

More generally, in $G_{2^{k-1}}$, $P_{(2^k)}$ is determined by the computation from eqn. (18) of

$$d_t^{(2^k)} = [C^{(2^{k-1})} B^{(2^{k-1})}]_{tt} \tag{20}$$

until the first value of t for which it is equal to 1. Here B, C, etc. refer to $G_{(2^{k-1})}$, and therefore $C^{(2^{k-1})} = C^{2^{k-1}}$, and $B^{(2^{k-1})}$ is obtained from $A^{2^{k-1}}$ through property (b). The node thus found is removed, causing the deletion of the corresponding row and column in C. The new connectivity matrix is used to compute the appropriate $C^{(2^{k-1})}$ and $B^{(2^{k-1})}$ matrices for the determination of the next node in $P_{(2^k)}$. The process is continued until no more loops of length $\leqslant 2^k$ are left.

All phases of a convergent algorithm are now available, but the resulting set P need by no means be optimal, in the sense of consisting of a minimal number of nodes that break all proper loops. Some means are desirable to ensure an efficient choice of nodes.

One possible criterion is the "node index" which results from the replacement of the union $\cup$ in eqn. (18) by a summation:

$$f_t^{(2k)} = \sum_{u=1}^{n} [c_{tu}^{(k)} \cap b_{ut}^{(k)}]. \tag{21}$$

$f_t^{(2)}$ is the number of loops of length 2 passing through node t. More generally, in a G_k, $f_t^{(2k)}$ is equal to the number of nodes s in proper loops of length $k+1$, $k+2, \ldots$ or $2k$ passing through node t such that ts is of length k, where s denotes any one of these nodes. Nodes that participate in a great number of loops can be expected to have large indices, and are therefore to be preferred in forming P. The node index also constitutes an indication for the sparseness of the graph for in a complete graph, $f_t^{(2)}$ = (number of nodes) -1, $\forall t$.

References

COATES, C. L. (1959). "Flow Graph Solutions of Linear Algebraic Equations", *IRE Trans. on Circuit Theory*, Vol. CT-6, p. 170.

DESOER, C. A. (1960). "The Optimum Formula for the Gain of a Flow Graph or a Simple Derivation of Coates' Formula", *Proc. IRE*, Vol. 48, p. 883.

DULMAGE, A. L., and MENDELSOHN, N. S. (1962). "On the Inversion of Sparse Matrices", *Math. Comp.*, Vol. 16, p. 494.

EVEN, R. K., and WALLACH, Y. A. (1966). "A Graph-Theoretic Method for the Solution of Elliptic Difference Equations", (to be published).

HALL, M. (1956). "An Algorithm for Distinctive Representatives", *Amer. Math. Monthly*, p. 716.

HARARY, F. (1959). "A Graph-Theoretic Method for the Complete Reduction of a Matrix with a View toward Finding its Eigenvalues", *J. of Math. and Phys.*, Vol. 38, p. 104.

HARARY, F. (1962). "A Graph-Theoretic Approach to Matrix Inversion by Partitioning", *Numerische Mathematik*, Vol. 4, p. 128.

HOHN, F., and SCHISSLER, L. (1955). "Boolean Matrices and the Design of Combinational Relay Switching Circuits", *Bell Syst. Tech. J.*, Vol. 34, p. 177.

MASON, S. J. (1953). "Feedback Theory—Some Properties of Signal Flow Graphs", *Proc. IRE*, Vol. 41, p. 1144.

MASON, S. J. (1955). "Feedback Theory—Further Properties of Signal Flow Graphs", *Proc. IRE*, Vol. 44, p. 920.

NATHAN, A. (1961). "A Two-Step Algorism for the Reduction of Signal Flow Graphs", *Proc. IRE*, Vol. 49, p. 1431.

PARTER, S. (1961). "The Use of Linear Graphs in Gauss Elimination", *SIAM Rev.*, Vol. 3, p. 119.

SESHU, S., and REED, M. B. (1961). *Linear Graphs and Electrical Networks*, Reading, Mass.: Addison-Wesley Publishing Co.

STEWARD, D. V. (1962). "On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations", *SIAM Rev.*, Vol. 4, p. 321.