A DIGITAL PROCEDURE FOR FINDING ALL DIRECTED SUBGRAPHS OF A SIGNAL FLOW GRAPH

A Thesis

Presented to

the Faculty of the Department of Electrical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Jack D. Greenwade

May 1969

484189

ACKNOWLEDGEMENT

The author would like to express his sincere appreciation to Prof. C. F. Chen for his guidance and encouragement during the development of this work. Also to Prof. W. P. Schneider, for his helpful comments during the formulation of this research, the author is very grateful.

A special word of appreciation is also due Mr. K. B. Rennie of Houston Lighting and Power Company for his excellent technical assistance. The completion of this thesis was greatly enhanced by the aid he contributed.

The author would also like to express his full gratitude to his wife, Judy, whose encouragement, understanding, and patience made this research possible.

A DIGITAL PROCEDURE FOR FINDING ALL DIRECTED SUBGRAPHS OF A SIGNAL FLOW GRAPH

An Abstract of a Thesis

Presented to

the Faculty of the Department of Electrical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

---

by

Jack D. Greenwade

May 1969

ABSTRACT

The algebra of flow graphs is well defined and several techniques are available for obtaining input-output relations. A brief review is presented with the addition of the application to sampled data systems. However, all of the techniques for obtaining input-output relations depend upon the users ability to identify all subgraphs within his system flow graph. Several papers have been presented concerning error free identification of these loops and paths, however, the techniques involved are usually laborious and lengthy.

Through the use of the connection matrix approach of Mr. C. V. Ramamoorthy a digital computer program is developed that finds all loops and paths that exist within a flow graph. This research was restricted to only continuous variable systems, however, techniques are suggested for handling the hybrid case, containing discrete as well as continuous variables. Several examples are presented to demonstrate the capability of the program. Suggestions are also made for handling multi-input-output problems.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

In literature today there are many methods of analysis of control systems, of varying degrees of accuracy, complexity, generality, sophistication, and usefulness. For several years people have been aware of topological forms for describing control systems or for that matter any set of differential equations. However, application of these topological form techniques appear to have taken a back seat to some of the more modern techniques for describing the behavior of control systems, perhaps understandably so. The algebra of signal flow graphs,[1-5] or flow graphs,[3] is well defined and reduction techniques[3] lead to the simplest graph in which all the variables of interest appear. The major usefulness of flow graph techniques arises from the topological properties which make it possible to obtain input-output relations by inspection.

Small loosely connected systems lend themselves readily to signal flow techniques. Unfortunately, most physical problems are not small loosely connected systems. It is obvious that a maximal strongly connected system of only 5 variables does not lend itself to these inspection techniques when one considers the number of possible loops

within the graph if every variable is connected to every other variable. Therein lies the secret to successful use of gain equations such as Mason's[1] or Coates:[2] recognition of all existing sub-loops and all existing paths from input to output.

As a matter of note, consider the case of a hybrid system, i.e., containing continuous variables and discrete or sampled variables. In fact, sampled flow graphs until recently were primarily intermediate steps in the evaluation of input-output relations and could not be used for simple representation of the system. Even when the final sampled flow graph was formulated it had little resemblance to the original system, and manipulation of the graph could not be given an easy physical interpretation. Recently, Mr. G. A. Bekey[6] proved that sampled flow graphs could be given a physical interpretation and input-output relations could also be obtained by inspection. It appears that no matter which technique one uses, Mason's, Coates', or Bekey's, the real difficulty lies in inspection. As the number of variables increases and the connection becomes stronger, it becomes increasily more difficult to recognize loops and paths by inspection. One is never quite sure he has not overlooked a loop or a path. It is understandable how the story got out that Mr. Mason, after several papers concerning

topological representations, presented a paper dealing with the avoidance of signal flow graphs.

Not too long ago, Mr. C. V. Romamoorthy[9] developed certain digital algorithms for obtaining certain information concerning the properties of flow graphs. His contribution cannot be overemphasized. The use of the connection matrix and associated algorithms for finding maximal strongly connected (M.S.C.) subgraphs are outstanding in their simplicity. Today with the widely accepted use of the digital computer as an engineering tool, one of the few remaining problems untouched by programmers is the solution of the signal flow graph for input-output relations. Several lengthy techniques[10] have been presented involving multiple matrix operations for obtaining all existing subloops and paths. These techniques use the branch transmission of the flow graph as matrix elements and after lengthy hand manipulations do an excellent job of obtaining the desired loops and paths. However, it would appear to be much more advantageous to know which variables were in a loop or path and the proper flow direction without much hand manipulation. The loop or path transmittances can always be readily obtained knowing the directed variables in a loop or path.

Therefore, it is highly desirable for a digital computer program to be developed to output to its user all directed loops and subloops within a flow graph as well as

all directed paths from input to output. It is also desirable that this program be general in form, such that the only limiting factor as to the size of the system to be handled would be the size of the memory of the computer on which the program would be run. A program of this type would enhance the use of signal flow techniques for obtaining input-output relations by either Mr. Mason's or Mr. Coates' formula.

CHAPTER II

GENERAL SIGNAL FLOW GRAPH THEORY

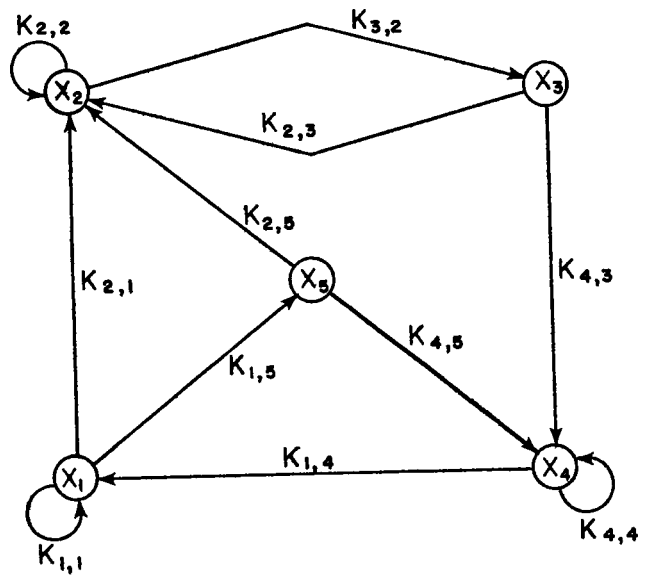
The purpose of this chapter will be to review with the reader the different techniques used to obtain input-output relations, and to define certain terms used with the application of these techniques. The association of a topological structure with a set of linear algebraic equations was introduced by Samuel J. Mason[1] and was called the Signal Flow Graph. A graph is defined as a collection of n objects called vertices or nodes, denoted by x_1, x_2, x_3,.....x_n and of m objects called branches or arcs, denoted by b_1,b_2,b_3,....b_m together with a set of incidence or boundary relations between the nodes and the branches such that some pair of nodes x_p, x_q (where p and q are not necessarily different) is associated with each branch b_j. Such nodes are called the boundaries of the branch. A branch together with its bounding nodes is called an edge. An edge is called a loop if the two boundaries are the same node. Such a node is called a loop boundary. A convenient pictorial representation of a graph consists of a set of circles, one for each node, and a set of line segments, one for each branch, drawn so that the ends of each line segment are the circles which correspond to the boundary nodes of the branch which

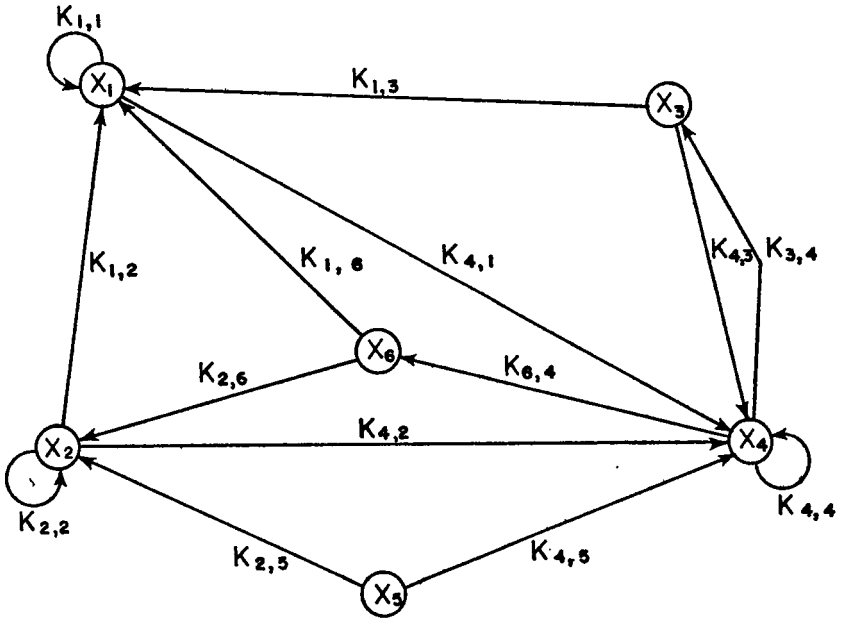
the line segment represents. A graph is oriented or directed by assigning a cyclic order to the bounding nodes of each branch. Pictorially, this is represented by an arrow on the line segment. Examples of such graphs are shown in Figure I.

The node from which the arrow points is called the positive boundary of that branch and that to which the arrow points is called the negative boundary. Similarly, a branch is incident to its negative boundary and incident from its positive boundary. A node that is never the negative boundary for any branch is called a reference node or source node. A node that is never the positive boundary for any branch is called a sink node.

It should be pointed out that there are two types of topological representations generally used for continuous variable problems. These two types are the Signal Flow Graph developed by Mason[1,2] and the Flow Graph developed by Coates.[3] Except for certain normalized graphs, the Signal Flow Graph is generally considered more complex than the Flow Graph. The complexity arises when using the topological formula for the solution of the set. The difference between the Signal Flow Graph and the Flow Graph is basically a difference in branch weights or transmittances. In the case of normalized graphs, this results in a difference in the incidence properties. The following paragraph is a general statement of these differences.

FIGURE I





EXAMPLES OF DIRECTED WEIGHTED GRAPHS.

Each nonreference node of the Signal Flow Graph is the boundary of a loop, the branch weight of which is one more than the branch weight of the loop of the corresponding node of the corresponding Flow Graph. If a nonreference node of the Flow Graph is not a loop boundary, the corresponding node of the corresponding Signal Flow Graph is the boundary of a loop with a branch weight of one. If a loop boundary of the Flow Graph has a branch weight of minus one, the corresponding node of the corresponding Signal Flow Graph is not a loop boundary.[3]

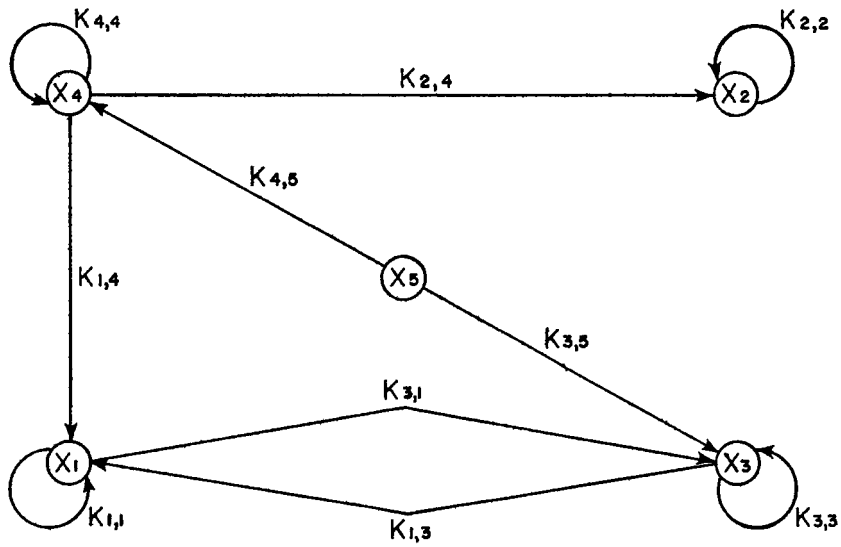
Mr. Coates has developed techniques for obtaining the solution of a set of linear algebraic equations through the use of Flow Graphs. Consider the set of equations:

$$k_x = \begin{pmatrix} k_{1,1} & 0 & k_{1,3} & k_{1,4} & 0 \\ 0 & k_{2,2} & 0 & k_{2,4} & 0 \\ k_{3,1} & 0 & k_{3,3} & 0 & k_{3,5} \\ 0 & 0 & 0 & k_{4,4} & k_{4,5} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = 0 \qquad [1]$$

The associated Flow Graph is shown in Figure II. If the expression for x_p as a function of the variables of the set ν is written

$$x_p = \sum_{j=1}^{S} \zeta_{p,r_j} x_{r_j} \qquad [2]$$

FIGURE II



FLOW GRAPH FOR SET OF LINEAR EQUATIONS, EQUATIONS [1]

then

$$\zeta_{p,r_j} = \frac{\sum_{\mu=1}^{\Gamma} N[\nu_{r_j} : r_j\text{-}p]_\mu (-1)^{P_\mu}}{\sum_{n=1}^{\Gamma} N[\nu:]_\eta (-1)^{P_\eta}} \qquad [3]$$

where ν is the set of variables, x_{r_j} the particular reference node, s denotes the number of members of ν, P_μ denotes the number of directed circuits of the μth $N\{\nu_{r_j} : r_j\text{-}p\}$, where P_η denotes the number of directed circuits of the ηth $N\{\nu:\}$. Equation [3] is Coates' equation for the transfer function from node r_j to node p. At this point it is necessary to define $N[\nu_{r_j} : r_j\text{-}p]$ and $N[\nu:]$. These are two classes of subgraphs. A connection is a subgraph of the Flow Graph (G) such that each included node has both a positive and a negative order of one. The positive order is the number of branches for which the node is the positive boundary and, similarly, the negative order is the number of branches for which it is the negative boundary. As a note, a node with a zero negative order is called a reference node. The connection of (G) with the largest number of branches is that which includes all nonreference nodes. Connections are denoted by $N\{(\ ):\}$ where $(\ )$ denotes the nodes of G which do not belong to the connection; therefore, connection $N\{(\nu):\}$ includes all nonreference nodes. A one connection is a subgraph of G such that all positive and negative orders of

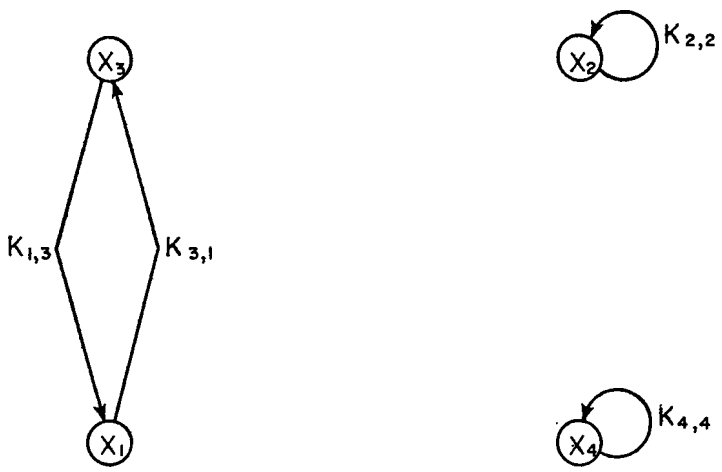
the included nodes are one, except for one negative and one positive order, each of which are zero. One connections are denoted by $\dot{N}\{(\ ):_{p-q}\}$ where ( ) denotes the nodes of G which are not included and where p and q denote subscripts of the nodes for which the negative and positive orders are zero, respectively. A proper one connection is a one connection such that $p \neq q$. With each subgraph is a coefficient which is defined as the product of the branch weights of the subgraph. The coefficients of $\dot{N}\{(\ ):\}$ and $\dot{N}\{(\ ):_{p-q}\}$ are denoted as $N[(\ ):]$ and $N[(\ ):_{p-q}]$, respectively.[3] Illustrations of connections and one connections are shown in Figure III with the associated coefficients.

As an example of Coates' equation consider the Flow Graph of Figure II. Assume one is interested in the transfer function for an input at x_5 and an output at x_2, therefore

$$x_2 = \zeta_{2,5}\, x_5 \qquad [4]$$

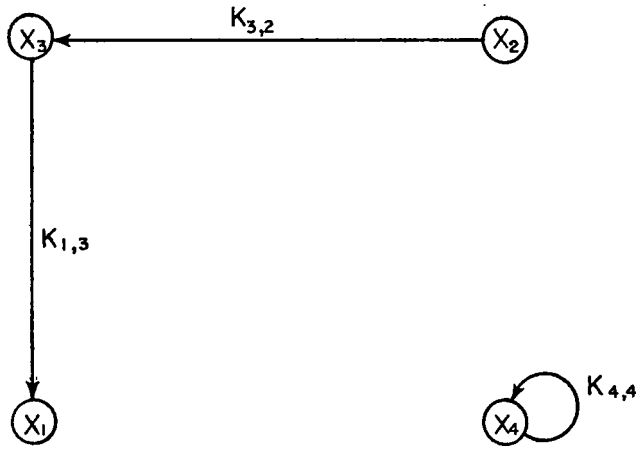
The connections and one connections are shown in Figure IV with associated coefficients. Using equation [3] and the information from Figure IV,

FIGURE III



(a)

a) Example of connection $N\{\nu:\}$



(b)

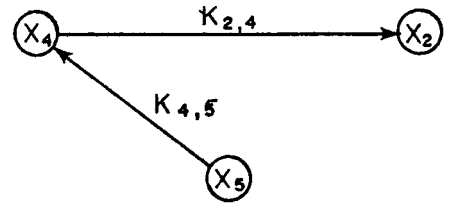
b) Example of one connection $N\{\nu:2\text{-}1\}$

ILLUSTRATIONS OF CONNECTIONS AND ONE CONNECTIONS.

FIGURE IV

The one connection $N\{\nu_5:5\text{-}2\}_1$

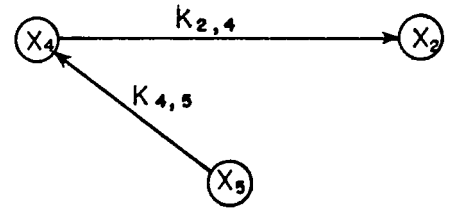
$P_1 = 2$



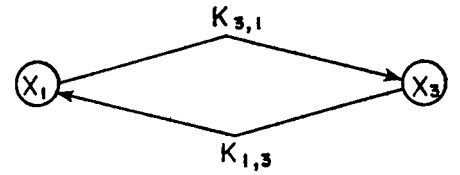
The coefficient $N[\nu_5:5\text{-}2]_1$ is $k_{2,4}k_{4,5}k_{1,1}k_{3,3}$

The next one connection $N\{\nu_5:5\text{-}2\}_2$

$P_2 = 1$



and the coefficient $N[\nu_5:5\text{-}2]_2$ is $k_{2,4}k_{4,5}k_{1,3}k_{3,1}$



The connection $N\{\nu_5:\}_1$

$P_1 = 4$

The coefficient $N[\nu_5:]_1$ is $k_{1,1}k_{2,2}k_{3,3}k_{4,4}$

The connection $N\{\nu_5:\}_2$

$P_1 = 3$

The coefficient $N[\ _5:]_2$ is $k_{2,2}k_{4,4}k_{1,3}k_{3,1}$

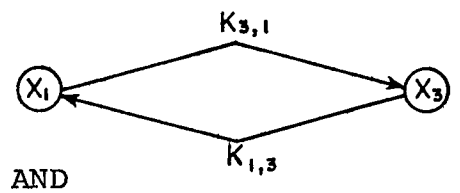


ILLUSTRATION OF CONNECTIONS AND ONE CONNECTIONS FOR FIGURE II WITH THE ASSOCIATED COEFFICIENTS.

$$\zeta_{2,5} = \frac{k_{2,4}\, k_{4,5}[k_{1,1}\, k_{3,3}(-1)^2 + k_{1,3}\, k_{3,1}(-1)^1]}{k_{1,1}\, k_{2,2}\, k_{3,3}\, k_{4,4}(-1)^4 + k_{1,3}\, k_{3,1}\, k_{2,2}\, k_{4,4}\,(-1)^3}$$

$$= \frac{k_{2,4}\, k_{4,5}\,[k_{1,1}\, k_{3,3} - k_{1,3}\, k_{3,1}]}{k_{2,2}\, k_{4,4}\,[k_{1,1}\, k_{3,3} - k_{1,3}\, k_{3,1}]} \qquad [5]$$

$$= \frac{k_{2,4}\, k_{4,5}}{k_{2,2}\, k_{4,4}}$$

A Mr. C. A. Desoer[4] did optimize Mr. Coates' equation. The result takes the form that follows. In general, there is more than one branch connecting the source node to the rest of the graph; obviously, in such cases the individual contributions of each branch must be summed. In order to solve for the variable x_L defined by the set of equations

$$\sum_{j=1}^{n} a\, k_j x_j = b_{k_j} \qquad (k = 1,2,3,\ldots n) \qquad [6]$$

the general equation for solution becomes

$$x_L = \frac{\sum_{\sigma} (-1)^{L_\sigma} C(G;0\text{-}L)_\sigma}{\sum_{\rho} (-1)^{L_\rho} C(G_0)_\rho} \qquad [7]$$

where $C(G;0\text{-}L)_\sigma$ represents the possible one-connection gains of the Flow Graph from the source node 0 to the node L and L_σ is the number of directed loops in the σth one connection

from 0 to L. $C(G_0)_\rho$ are the possible connection gains of the Flow Graph G_0 (which is graph G with the source node 0 deleted) and L_ρ is the number of directed loops in the ρth connection of G_0. Whether one is applying Mr. Coates' equation or Mr. Desoer's equation, it appears obvious the success depends on one's ability to find all the connections and one connections by inspection. Simple 4 or 5 node problems, loosely connected, present no major difficulty. However, the stronger the connection and the larger the number of variables become, the more nightmarish the application of the gain equations become.

Consider the approach taken by Mr. Mason. The general expression for the Signal Flow Graph transmission is

$$T = \frac{\sum_k P_k \Delta_k}{\Delta} \qquad [8]$$

$$= \frac{[(P_1+P_2+\ldots+P_n)(1-L_1)(1-L_2)\ldots(1-L_m)]^*}{[(1-L_1)(1-L_2)\ldots(1-L_m)]^*} \qquad [9]$$

Here the quantity Δ represents the determinant of the Signal Flow Graph rather than the determinant of the network from which the Signal Flow Graph was constructed. The Signal Flow Graph determinant can be evaluated by the loops of the graph. A loop is a simple closed cycle of graph branches with all of the branches pointing in the same direction around the cycle. The word "loop" may also mean a number

equal to the product of the branch transmissions in the geometrical loop. The graph determinant is conveniently defined as the bracketed expression in the denominator of the above equation [9]. Quantities $L_1, L_2, \ldots . L_m$ are the m different loop transmittances in the graph. The asterisk indicates a special rule for multiplying the quantities within the brackets. In carrying out the multiplication indicated by the parentheses, it is to be understood that a term will be dropped if it contains the product of two loops which touch (have a node in common) in the graph. In short, Δ is equal to unity minus an algebraic sum of further terms. Each of these terms is either a single loop or a product of non-touching loops, and the sign of the term is positive (or negative) for an even (or odd) number of loops in the product. The graph transmission T is by definition equal to the quotient of some designated dependent node signal and the source node signal. Quantities P_k in the numerator of equations [8] and [9] stand for different paths from the source node to the designated dependent node. The number P_k to be substituted into the above equations is not, of course, the geometrical path, but rather the product of the branch transmissions along that path. Quantity Δ_k is called the cofactor of path P_k. It is constructed in exactly the same manner as the graph

determinant (Δ) with the additional restriction that any terms containing loops which touch (have a node in common) path P_k shall be dropped. As an example, equation [8] will be applied to the graph of Figure V with node x_1 being the source node and node x_5 being the designated dependent node. Obviously, there is only one path from source x_1 to node x_5, therefore,

$$P_1 = A_{1,2}A_{2,3}A_{3,4}A_{4,5}$$

The loops present in Figure V are shown below:

$$L_1 = A_{2,3}A_{3,2}$$

$$L_2 = A_{4,5}A_{5,4}$$

Since there are only two loops and they do not touch, the graph determinant is:

$$\Delta = 1-L_1-L_2 + L_1L_2$$

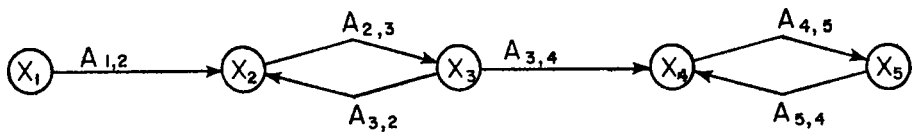
$$= 1-A_{2,3}A_{3,2} - A_{4,5}A_{5,4} + A_{2,3}A_{3,2}A_{4,5}A_{5,4}$$

Since there is only one path and both loops touch the path, the associated cofactor (Δ_1) is one

$$\therefore \quad T = \frac{P_1\Delta_1}{\Delta}$$

$$= \frac{A_{1,2}A_{2,3}A_{3,4}A_{4,5}(1)}{1 - A_{2,3}A_{3,2} - A_{4,5}A_{5,4} + A_{2,3}A_{3,2}A_{4,5}A_{5,4}} \qquad [10]$$

FIGURE V



FLOW GRAPH TO BE USED FOR MASON'S EXAMPLE.

Again, it is obvious that the success of this approach is dependent upon the user's ability to recognize all possible loops and paths in the Signal Flow Graph by inspection.

In 1967, a paper was presented in the IEEE Transactions on Automatic Controls. This paper, authored by Mr. M. Sedlar and Mr. G. A. Bekey,[6] extended the topological representation of either continuous or discrete systems to mixed discrete-continuous (hybrid) systems. Several authors have extended the Signal Flow Graph approach to the study of sampled data systems.[11,12,13] However, none of these extensions make it possible to apply Mason's gain formula directly. The difficulty arises from the presence of samplers in the sampled data systems. Conventional techniques fail because it is not possible to replace the sampler by a transfer function. Introducing a new symbol to represent the sampler, it is possible to generalize Signal Flow Graph techniques so that they can be applied both to sampled data systems and to continuous systems. Mr. Bekey formulated a Signal Flow Graph representation which maintains a topological similarity with the original systems and which makes the algebra of Signal Flow Graphs applicable. Moreover, input-output relations can be obtained by inspection in terms of Laplace transforms, Z transforms, or modified Z transforms.

As stated earlier, a Signal Flow Graph is a network of nodes (circles) connected by directed branches (line

segments). The line segments and circles (branches and nodes) represent operations to which the variables are subjected. In the case of linear continuous systems, the nodes are associated with summing operations and the branches with multiplication. For this reason, the graph contains the same information as the mathematical model or block diagram. In the case of the sampled data system, where the variables can be both continuous and discrete, the operation of sampling is also present. A new symbol is needed for the operation of sampling. Therefore, Mr. Bekey chose an empty circle in the graph to represent a continuous variable and a full circle to represent a discrete variable. He termed them white nodes and black nodes, respectively. The black node is defined as follows:

1. Black nodes (full circles) represent discrete variables.
2. The value of the variable represented by any black node is the sampled form of the sum of all variables entering the black node.

A linear continuous system can be described by the set of equations:

$$\sum_{i=1}^{n} a_{j_i}(s)x_i(s) = r_j(s), \qquad j=1,2,3....n \qquad [11]$$

where $x_i(s)$ are the transforms of the system variables,

$r_j(s)$ are the transforms of input signals, and a_{j_i} are transfer functions. This system can be represented by Mason's Signal Flow Graph techniques. Similarly, a sampled data system can be described by the set of equations:

$$\sum_{i=1}^{n} a_{j_i}(s)x_i(s) = \sum_{i=1}^{n} b_{j_i}(s)x_i^*(s) + r_j(s) \qquad [12]$$

$$j = 1,2,\ldots n$$

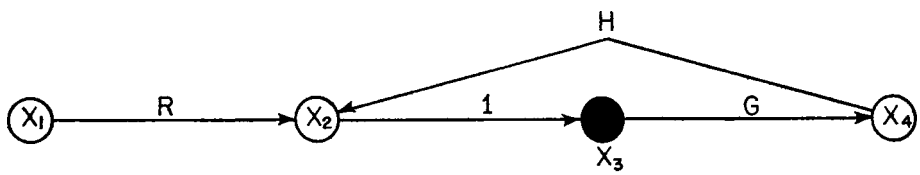
where $x_i^*(s)$ are transforms of sampled variables. This system can be represented by a new Signal Flow Graph which contains both black and white nodes. As an example, the sampled data system described by the mathematical model

$$x_1(s) = 1$$
$$x_2(s) = R(s)x_1(s) - H(s)x_4(s)$$
$$x_3(s) = x_2^*(s)$$
$$x_4(s) = G(s)x_3(s)$$

is completely described by Figure VI.

Since paths and loops were defined earlier for a continuous variable system, it is now necessary to define the types of paths and loops found in the hybrid system. Paths and loops containing only white (continuous) nodes will be referred to as being of type 1 and denoted by $\mu^{(1)}$ and $\nu^{(1)}$, respectively. Type 2 paths and loops, denoted by $\mu^{(2)}$ and $\nu^{(2)}$, will be those paths and loops which contain at least

FIGURE VI



FLOW GRAPH FOR HYBRID (CONTINUOUS AND DISCRETE) SYSTEM.

one black (discrete) node. Also, a Type 1 path is elementary if it does not meet the same node more than once. A Type 1 loop is elementary if, apart from the coincident initial and terminal nodes, every other node it meets is distinct. A Type 2 elementary path is one composed of distinct segments such that no black node is met more than once. Mr. Bekey points out that if a white node is met more than once it must belong to different segments. Similarly, an elementary loop of Type 2, apart from its coincident initial and terminal nodes, must meet distinct other black nodes. Consider Figure VI, there are no paths of Type 1 or loops of Type 1. There is one path of Type 2 and one loop of Type 2, therefore:

$$\mu_1^{(2)} = R*G$$

$$\nu_1^{(2)} = (GH)*$$

Once the two separate types of paths and loops are established, certain topological connections between them can be defined. A Type 1 loop is connected with a segment or another Type 1 loop if and only if they contain a node in common. A Type 2 loop is connected with any path or another loop of Type 2 if and only if they contain a black node in common. It should be noted that in this case (Type 2) the presence of a white node in common is of no consequence. Path and loop transmissions have already been

defined in this Chapter. However, their definitions must be repeated with slight modification due to the presence of discrete variables. A path transmission, $P(\mu)$, is the product of the individual transmissions of the branches which appear in that path. Individual transmissions are taken in the order in which the branches appear, and the presence of a black node denotes that the sampled form of the preceding product should be taken. A loop transmission, $L(\nu)$, can be defined in the same manner, however, if there are any black nodes in the loop, then a black node should be taken as the initial node.

To apply the notation of Mr. Bekey one should remember the equation for the determinant of a graph. However, now there are two determinants. The first may be defined for loops of Type 1 and is the same as the graph determinant developed by Mr. Mason.

$$\Delta^{(1)} = (1-L_1^{(1)})(1-L_2^{(1)})(1-L_3^{(1)}).....(1-L_i^{(1)})^* \qquad [13]$$

where the asterisk denotes the special multiplication whereby a term will be dropped if it contains the product of loops that touch (have a node in common) in the graph. Similarly, the second determinant is defined as:

$$\Delta^{(2)} = (1-L_1^{(2)})(1-L_2^{(2)})(1-L_3^{(2)})....(1-L_i^{(2)})^* \qquad [14]$$

This time, the asterisk denotes the special multiplication where a term is dropped if it contains the product of any Type 2 loops that touch (contain a black node in common) in the graph.

The Input-Output relation (C) of the hybrid system is given by:

$$C = \frac{\Delta_j^{(1)}}{\Delta^{(1)}} \times \frac{\sum_{i=1}^{n} P_i \Delta_i^{(2)}}{\Delta^{(2)}} \qquad [15]$$

where P_i is the path transmittance of any path (Type 1 or Type 2) connecting input and output nodes, $\Delta^{(2)}$ is the second determinant of the graph, and $\Delta_i^{(2)}$ is the subdeterminant of the path μ_i. $\Delta_i^{(2)}$ is obtained from $\Delta^{(2)}$ by omitting all terms of $\Delta^{(2)}$ that contain loop transmissions of the loops $\nu_j^{(2)}$ connected with the path μ_i. The symbol, (x), represents the operation of multiplication of $\Delta_j^{(1)}/\Delta^{(1)}$ by each segment transmission appearing in $\sum_i P_i \Delta_i^{(2)}/\Delta^{(2)}$. If any of the segment transmissions appear in sampled form, the multiplication must be performed on the corresponding continuous quantities, and the product sampled. $\Delta^{(1)}$ is the first determinant of the graph and $\Delta_j^{(1)}$ is the subdeterminant of the related segment σ_j. $\Delta_j^{(1)}$ is obtained from $\Delta^{(1)}$ by omitting all terms containing loop transmissions of the loops $\nu^{(1)}$ connected with segment σ_j. As an example, consider the graph of Figure VI where

$$P_1^{(2)} = R^*G$$

and

$$L_1^{(2)} = (HG)^*$$

there are no paths or loops of Type 1, therefore

$$\Delta_j^{(1)} = \Delta^{(1)} = 1 \text{ for all } j$$

and since $L_1^{(2)}$ touches $P_1^{(2)}$ then $\Delta_i^{(2)} = 1$. Therefore, using equation [15]

$$C = \frac{1}{1} \times \frac{R^*G(1)}{1-(HG)^*}$$

$$= \frac{(R)^*G}{1-(HG)^*}$$

where the asterisk represents the sampled form of the quantities in parentheses.

There are two special cases that must be discussed. First, all loops of the system contain at least one sampler. In this case, the graph has only loops of Type 2 and therefore,

$$\Delta^{(1)} = \Delta_j^{(1)} = 1 \text{ for all } j$$

This was shown in the above example. The equation for C then reduces to:

$$C = \frac{\sum_i P_i \Delta_i^{(2)}}{\Delta^{(2)}} \qquad [16]$$

The second case, all the loops of the system are continuous. In this case, the graph contains only loops of Type 1. Therefore

$$\Delta^{(2)} = \Delta_i^{(2)} = 1 \text{ for all } i$$

Then C reduces to

$$C = \frac{\Delta_j^{(1)}}{\Delta^{(1)}} \times \sum_i P_i \qquad [17]$$

Moreover, if this graph contains no black nodes, then each P_i is a segment and the symbol x reduces to simple multiplication and the equation for C reduces to Mason's equation:

$$C = \frac{\sum_i P_i \Delta_i^{(1)}}{\Delta^{(1)}} \qquad [18]$$

It appears that after all the work of Mr. Coates, Mr. Mason, and Mr. Bekey, application of their work is largely restricted to small problems that are designed to be solved by one of their equations. A practical problem involving 20 or more variables strongly connected could be a nightmare to solve by inspection. However, if by some means all loops and paths could be readily obtained, any one of the above mentioned equations could be used quite easily to obtain input-output relations.

CHAPTER III

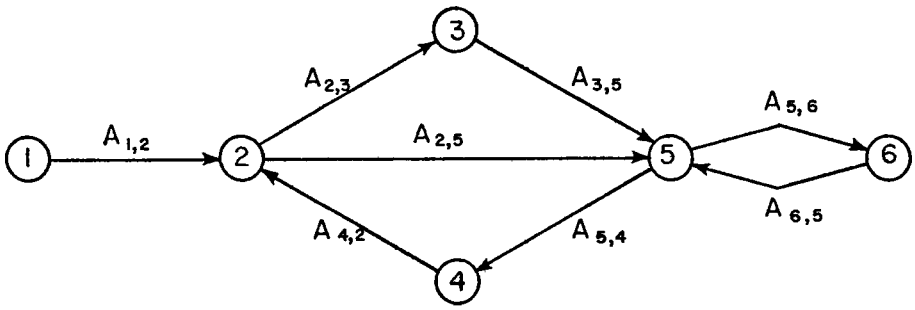
DEVELOPMENT AND APPLICATION OF THE CONNECTION MATRIX

A connection matrix may be defined as a matrix representing a Signal Flow Graph. This seems rather a simple definition and in fact is. However, for this application a more complex definition is not required. The connection or connectivity matrix contains either zeros or ones as its elements. The matrix is generated from the graph and is always a square matrix, each row and corresponding column representing exits and entries to that node, respectively. As an example, consider Figure VII. The connectivity matrix (IC) for the graph of Figure VII appears below:

$$IC = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \qquad [19]$$

Observing row 2 which corresponds to exits of node 2, there are 2 exits, one to node 3 and one to node 5. Similarly, observing column 2 one sees there are 2 entries to node 2, one from node 1 and one from node 4. Therefore, an element of the connection matrix will be a zero if there is not a

FIGURE VII



FLOW GRAPH FOR ILLUSTRATION OF CONNECTION MATRIX DEFINITION.

single branch from the element's row index node to its column index node. Similarly, the element will be a one if there is a single branch from the element's row index node to its column index node.

The approach taken in this paper is that of C. V. Ramamoorthy[9] and relates the connectivity considraations of directed graphs to the quantitative aspects of physical systems represented by weighted graphs. The first step is definition and construction of the reachability matrix (IR). A row vector IR_s of the IR (reachability) matrix will be defined as the reachability vector of node s, a one in its k-th column implies then that node k is reachable from node s. Thus, the row vector IR_s gives all nodes reachable from node s. Therefore, the reachability matrix defines which nodes are reachable from which nodes. It should be pointed out that for the purpose of this paper the first order node will be the input and the highest order node will be the output. Therefore, checking the last element of the first row vector of the reachability matrix immediately tells the user if the output is reachable from the input. The steps used by Mr. Ramamoorthy to obtain the reachability matrix are shown below:

Step 1. Let IR_s be a row vector of dimension η such that it is initially equal to IC_s, the row vector of the connection matrix corresponding to the

starting node, s, i.e., $IR_s = IC_s$

Step 2. Examine the column elements of IR_s. Let $(IR_s)_i$ be its i-th column element, which is non zero and not previously examined. Update IR_s by performing a component wise logical "or" operation to the i-th row of IC, i.e., $IR_s(new) = IC_i \cup IR_s(old)$.

Step 3. Repeat Step 2 above until no additional changes appear in IR_s.

Step 4. Pick the next node as a new starting node and repeat the above steps.

This algorithm is very efficient because it only involves logical "or" operations between row vectors, rather than repeated matrix manipulation. There are some interesting additional properties of the IR matrix. If the main diagonal elements of the IR matrix are all zero, then the graph is loopless. Also, if any row of IR and its corresponding column are all ones, then the graph is strongly connected.

There is one more matrix that is of primary concern in this paper and it too was developed by Mr. Ramamoorthy. The largest strongly connected subgraph that contains a given node is defined as a maximal strongly connected (M.S.C.) subgraph. It is unique for any given node in its set. The procedure for determination of all maximal strongly connected subgraphs follows:

Step 1. Construct the reachability (IR) matrix for the given graph.

Step 2. Construct the transpose of the IR matrix or the reachability matrix of the transpose of the connection matrix (IC^T).

Step 3. Construct the IM matrix such that IM_s (row vector) = $IR_s \cap IR_s^T$. The symbol $\cap$ represents the logical "and" operation.

The number of M.S.C. subgraphs is given by the number of distinct non zero row vectors of the IM matrix. The nodes of the M.S.C. subgraph correspond to the non zero column elements of the IM row vector. However, this algorithm does not direct the nodes in the subgraph nor does it indicate the presence of any subgraphs within the M.S.C. subgraph. On the other hand, it is a very efficient means of determining the number of distinct loop sets within a given graph.

As an example, consider the graph of Figure VII, having the connection matrix shown in equation [19]. Constructing the reachability matrix IR:

$$IR_1 = IC_1 = 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$IR_1(\text{new}) = IC_2 \cup IR_1(\text{old})$$

$$= 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \cup 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$= 0 \quad 1 \quad 0 \quad 1 \quad 0$$

$$IR_1(\text{new}) = IC_3 \cup IR_1(\text{last})$$

$$= 0\ 0\ 0\ 0\ 1\ 0 \cup 0\ 1\ 1\ 0\ 1\ 0$$

$$= 0\ 1\ 1\ 0\ 1\ 0$$

$$IR_1(\text{new}) = IC_5 \cup IR_1(\text{last})$$

$$= 0\ 0\ 0\ 1\ 0\ 1 \cup 0\ 1\ 1\ 0\ 1\ 0$$

$$= 0\ 1\ 1\ 1\ 1\ 1$$

continuing

$$IR_1(\text{final}) = 0\ 1\ 1\ 1\ 1\ 1$$

In a similar manner, the remaining row vectors of IR are obtained such that

$$IR = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Now construct the transpose of IR

$$IR^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Now construct the IM matrix

$$IM_1 = IR_1 \cap IR_1^T$$
$$= 0\ 1\ 1\ 1\ 1\ 1 \cap 0\ 0\ 0\ 0\ 0\ 0$$
$$= 0\ 0\ 0\ 0\ 0\ 0$$

$$IM_2 = IR_2 \cap IR_2^T$$
$$= 0\ 1\ 1\ 1\ 1\ 1 \cap 1\ 1\ 1\ 1\ 1\ 1$$
$$= 0\ 1\ 1\ 1\ 1\ 1$$

Similarly, the remaining row vectors of IM are obtained such that:

$$IM = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

This matrix tells one that there is one maximal strongly connected subgraph involving nodes 2, 3, 4, 5, and 6. However, this matrix gives no indication of the subgraphs that exist within the M.S.C. subgraph found above. The next chapter deals with the interpretation of this data and formating it to be meaningful through digital techniques.

CHAPTER IV

DEVELOPMENT OF THE DIGITAL PROCEDURES AND COMPUTER PROGRAM

The purpose of this chapter is to explain the development of a digital computer program for finding all directed loops and paths within a given signal flow graph. This program will be restricted to two classes of graphs, those containing all continuous variables and those containing all discrete variables. This restriction is imposed by the definitions for paths and loops for these two classes, primarily in each case no node (variable) may be encountered more than once in any path or in any loop (excluding the initial and terminal node of the loop). Since the primary purpose of finding all paths and loops within a given graph is to determine input-output relations, one further restriction is imposed. That restriction is that the input node must be ordered as the first node in the graph and that the output node must be the highest ordered node in the graph. For example, if there were 13 nodes in a system, node 1 must be the input and node 13 must be the output. However, it is felt that this is not a severe restriction. It must be assumed that the user has no knowledge of how many loops or paths exist within the system under investigation. Therefore, an arbitrary limit must be used to alarm

the user when the program has gone beyond its dimensions. The program developed for this paper was arbitrarily designed for a maximum of 20 nodes with a maximum of 40 loops and 40 paths. To use this program for larger problems only requires re-dimensioning of the arrays involved. It should be pointed out that the major restriction as to the size of the problem, is dictated by the amount of memory available within the machine to be used. There are a number of arrays that must be stored and, therefore, the amount of memory available becomes very important. The input data required is very simple, consisting of the number of nodes in the graph and the connection matrix for the graph as described in Chapter III. The output of this program will be the loops and paths that exist within the given graph. The nodes associated with these loops and paths will appear in their correct order of transmission. With this information as well as having the flow graph before him, the user will be able to formulate all of the loop transmissions and path transmissions necessary for application of one of the gain equations described in Chapter II.

The program begins by inspecting the main diagonal of the connection matrix (IC). Since the index of the row and column vectors of the IC matrix correspond to the order of the nodes in the system, a non zero element on the diagonal indicates a self loop on the associated node. Upon

identification of the nodes having self loops, they are stored in temporary vectors called ISLOOP vectors and the number of self loops existing is called ISLCNT. This information will be used later in the program to add these self loops to the directed loop vectors found. Upon identifying all self loops the main diagonal of the IC matrix is set to zero. This eliminates duplication of effort in a later stage of the program. The next step is to construct the reachability matrix (IR) using the algorithm developed by Mr. Ramamoorthy as described in Chapter III. This portion of the program requires an n by n operation to insure all elements are taken into account. The symbol n represents the number of nodes existing in the graph being considered. From the IR matrix, the IM matrix is constructed using the algorithm of Mr. Ramamoorthy as described in Chapter III for finding M.S.C. subgraphs. This operation is straightforward and requires little machine time since all operations are in fixed point format. Once the IM matrix is obtained the nodes associated with each maximal strongly connected (M.S.C.) subgraph must be found. This is accomplished very simply by identifying unique non zero row vectors in the IM matrix. Once a non zero row vector is found and stored as a IMLOOP vector, it is not necessary to check the row vectors of any of the nodes appearing in that non zero row vector. It is also not necessary to check the final row vector in the IM matrix, for it is associated with

the output node and if it were in a loop or subgraph it would be detected earlier. As an example, if the first non zero row vector of the IM matrix was the row vector associated with node 2 and the 2nd, 3rd, rth, and 5th column elements were non zero, it would not be necessary to investigate the 3rd, 4th, or 5th row vector of the IM matrix. The contents of these row vectors would be identical to the 2nd row vector just investigated in that the 2nd, 3rd, 4th, and 5th column elements of each of the other three row vectors would be non zero. Therefore, the nodes in the example above would be stored by the program in the form of a IMLOOP vector such that:

```
IMLOOP (1,1) = 2
IMLOOP (1,2) = 3
IMLOOP (1,3) = 4
IMLOOP (1,4) = 5
```

At this point the program only knows that these four nodes are in a subgraph. They may actually be in a loop or they may be in the form of several loops touching each other. However, at this point only the fact that they are in a subgraph is important. It should be noted that the first subscript of the IMLOOP vector represents the number of the subgraph, i.e., the first subgraph, second subgraph, etc. The second subscript represents the number or index of the node as it appears in the subgraph, i.e., the first, second,

or third node found, etc. Again, at this point the index has nothing to do with the transmission order of the node in any loop. This procedure is followed until every maximal strongly connected (M.S.C.) subgraph is obtained from the total system IM matrix. It is necessary to find out if within each of these M.S.C. subgraphs there exist any smaller subgraphs. Using the theory of superposition, these smaller subgraphs may be obtained.

The first step in investigating for smaller subgraphs is to consider each IMLOOP vector separately. Beginning with the first vector, a new connection matrix is constructed containing only the nodes involved with the IMLOOP vector. This is accomplished by extracting the proper elements from the original IC matrix for the system in question. It is at this point that deletion of the main diagonal elements of the original IC matrix saves time. This new connection matrix is called the ITEST matrix. It is now necessary to determine if there exists the possibility of any subgraphs appearing inside the IMLOOP vector. This is accomplished by finding the number of exits that appear in each ITEST row vector. If the number of exits in each row vector never exceeds one, then there can be no inside subgraphs. At that point, the next IMLOOP vector is selected and a new ITEST matrix constructed for it and checked for the presence of inside subgraphs. This continues until all

IMLOOP vectors have been checked. If, on the other hand, the number of exits in a row vector exceed one, then an approach must be used to identify the additional subgraphs and create new additional IMLOOP vectors. This is accomplished through the use of superposition as mentioned earlier. The program eliminates one node from the ITEST matrix by temporarily storing its corresponding row and column vectors then setting those elements to zero in the ITEST matrix. Then a new IR and IM matrix are constructed from the ITEST matrix. Any new smaller subgraphs will appear in the new IM matrix. If any are found, they are stored as new IMLOOP vectors similar to the initial IMLOOP vectors. The program then restores the row and column vectors of the eliminated node and then eliminates the next node in a similar manner. This process is repeated for every node in the IMLOOP vector under investigation. All new IMLOOP vectors found are added to the number of IMLOOP vectors and the whole process is repeated for each IMLOOP vector until no new IMLOOP vectors are found. It must be remembered that all the IMLOOP vector implies is that the associated nodes are in a subgraph of some configuration.

After all possible IMLOOP vectors are found, an algorithm was developed to order the nodes in the IMLOOP vector according to their sequence of transmission, if, in fact, a loop does exist containing those nodes. If the loop does exist, then it will be ordered by transmission and called

a LOOP vector. This is accomplished by again constructing an ITEST matrix that corresponds to only that portion of the system connection matrix (IC) that contains the nodes of the IMLOOP vector under investigation. It should be remembered that the program knows how many IMLOOP vectors exist and how many nodes are in each one. These quantities are stored as IMLCNT and IMLNDE, respectively. An ITLOOP vector will now be constructed that will direct the flow through the nodes of the IMLOOP vector. For each IMLOOP vector only ITLOOP vectors will be stored that contain the same number of nodes as the IMLOOP vector. There may, indeed, be more than one directed loop around a given set of nodes. However, every possible loop containing these given nodes must be identified. It should be obvious that the index nodes of the ITEST matrix do not correspond 1 for 1 to the correct nodes of the system. For example, if the IMLOOP vector contained nodes 2, 3, 4, and 5 of the system, the ITEST matrix would refer to these nodes as nodes 1, 2, 3, and 4, respectively. The LOOP vector is obtained through the use of the directed ITLOOP vector and the not directed IMLOOP vector. Before discussing how the ITLOOP vector is obtained and directed, the above-mentioned construction of the LOOP vector will be described. Assume that the ITLOOP vector is directed containing nodes 1, 2, 3, 4, and 3 in that transmission order for the IMLOOP vector containing nodes 2, 3, 4, and 5 of the

system. Since it is known that the new LOOP vector will contain four nodes, a program "Do" statement may be used to direct the LOOP vector. This is accomplished by the following instructions, statement numbers shall be fictitious in comparison to the actual program.

```
      DO   1     I = 1, NONODE
      IA =   ITLOOP(II,I)
    1 LOOP (IC,I) = IMLOOP (K,IA)
```

where NONODE corresponds to the number of nodes in the IMLOOP vector, II is the number of the ITLOOP vector being used, IC is the number of the LOOP vector being constructed, K is the number of the IMLOOP vector being investigated. Therefore, if the ITLOOP vector was:

```
ITLOOP (1,1) = 1
ITLOOP (1,2) = 2
ITLOOP (1,3) = 4
ITLOOP (1,4) = 3
```

then

```
for I = 1   IA = 1   LOOP (IC,1) = IMLOOP (K,1)
    I = 2   IA = 2   LOOP (IC,2) = IMLOOP (K,2)
    I = 3   IA = 4   LOOP (IC,3) = IMLOOP (K,4)
    I = 4   IA = 3   LOOP (IC,4) = IMLOOP (K,3)
```

if the IMLOOP vector was:

IMLOOP (K,1) = 2

IMLOOP (K,2) = 3

IMLOOP (K,3) = 4

IMLOOP (K,4) = 5

then the corresponding directed LOOP vector would be:

LOOP (IC,1) = 2

LOOP (IC,2) = 3

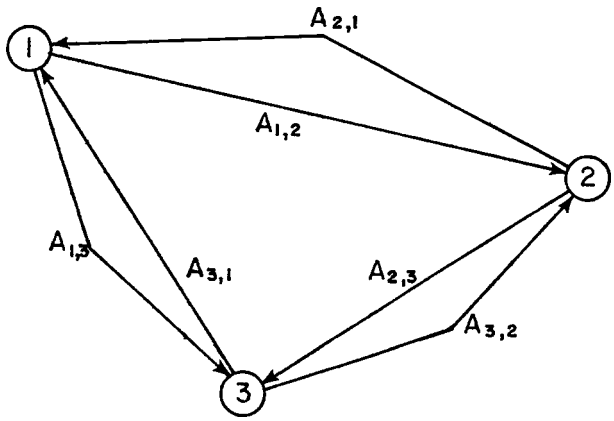
LOOP (IC,3) = 5

LOOP (IC,4) = 4

This would confirm and order by transmission the fact that a loop (2,3,5,4) did in fact exist within the system graph. It is implied that there is a branch from node 4 to node 2, and this fact is confirmed during the construction of the ITLOOP vector.

The construction of the directed ITLOOP vector will now be discussed. Recalling that an ITEST matrix was constructed for the IMLOOP vector under test, this matrix is now investigated for non zero elements. The best way to describe this algorithm is by an example. It is a very simple approach and is very accurate. Consider the subgraph in Figure VIII. It must be remembered that although this subgraph does contain three other loops involving sets of 2 nodes, they have already been identified as additional IMLOOP vectors. This example is only concerned with constructing the ITLOOP vectors containing all three nodes. Therefore,

FIGURE VIII



FLOW GRAPH FOR DEMONSTRATION OF LOOP FINDING ALGORITHM.

the ITEST matrix for the subgraph of Figure VIII is as follows:

$$\text{ITEST} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

The first node is considered the initial node of any loop appearing. It then follows by checking row 1 for non zero elements that node 1 reaches node 2 on a single branch, therefore,

$$\text{ITLOOP}(1,1) = 1$$

$$\text{ITLOOP}(1,2) = 2$$

However, node 1 also reaches node 3 on a single branch, therefore, a new ITLOOP vector is constructed such that

$$\text{ITLOOP}(2,1) = 1$$

$$\text{ITLOOP}(2,2) = 3$$

Upon completion of checking row 1, the program steps to the last stored node in ITLOOP (1,) which is node 2. It now checks the corresponding row of the ITEST matrix for non zero elements and finds

$$\text{ITLOOP}(1,3) = 1$$

However, comparing this node to the previous nodes in ITLOOP(1,) it finds node 1 appears more than once. Therefore, it continues to check row 2 and finds

$$\text{ITLOOP}(1,3) = 3$$

Now that ITLOOP (1,) contains 3 unique nodes (the same

number of nodes as in the IMLOOP vector) the program steps to the last stored node in ITLOOP(2,) which is node 3 and checks its corresponding row vector in the ITEST matrix. Therefore:

ITLOOP(2,3) = 1

Again node 1 appears more than once in the loop so the program continues and finds

ITLOOP(2,3) = 2

The program now has two ITLOOP vectors directed and stored, if, in fact, the last node in each vector does reach the first node in each vector on a single branch, then the vectors are retained. Otherwise, they are discarded. Therefore, within the subgraph described by Figure VIII there are two loops involving 3 nodes and they are directed node 1, node 2, and node 3 and node 1, node 3, and node 2.

From the ITLOOP vectors found in this manner, LOOP vectors are constructed in the manner described earlier from the IMLOOP vector in question. The next IMLOOP vector is selected and the whole procedure is repeated until all loops are obtained and directed. It should be pointed out that this procedure does not overlook a single loop that exists within the system flow graph no matter how complex. It is at this point that the self loops (ISLOOP) are inserted as LOOP vectors and therefore all loops are now directed and stored to be printed later.

The next step is to find all directed paths from input to output. This algorithm is very simple and yet very accurate. The program calls in the original connection matrix (IC) for the system and begins with the first row vector, corresponding to exits from the input node. This row vector is examined for non zero elements in much the same manner as described in the preceding paragraphs for finding the ITLOOP vectors. It is assumed that there is at least one path from input to output. Therefore, the path count (IPCNT) or number of paths is initialized to one. Also, the first node in any path will be node 1, the input node. Upon identification of the first non zero element in the first row vector of the IC matrix, the second node in the first path (IPATH) vector is found and stored. Should there be any other non zero elements in that row new path vectors are created for each non zero element. The path count (IPCNT) is incremented by one for each additional path vector. After completion of the investigation of the first row vector of the IC matrix, the last node stored in the first path vector is checked to see if it is the output node. If it is, then the first path is complete and the next path vector, if it exists, is called. If, on the other hand, the last node stored in the first path vector is not the output node, then the corresponding row vector is called from the IC matrix for investigation. Again, the program looks for non

zero elements, upon identification of the first non zero element, the number of nodes (IPNODE) in the first path vector is increased by one and the corresponding node is stored in the IPATH vector. The remainder of the row vector is checked for additional non zero elements and new path vectors are created for each one identified, repeating, of course, the nodes of the path vector that stepped the program to the row vector of the IC matrix it is now checking. This procedure is followed until every possible path or directed segment has been found that originates from the input node. It should be pointed out that as each new node is found for a path it is compared with each previous node in that path. If the node appears more than once, it is rejected and the program looks for another node that is reachable on a single branch from the preceding node in the path. If there were no more nodes reachable and the last node was not the output node then that path vector represents only a directed segment and is not retained as a path vector. Only those segments that reach the output node are retained as valid paths. This algorithm requires very little machine time and no existing paths in the system flow graph will escape undetected.

Upon completion of the two sections of the program (LOOP finding and IPATH finding) all the LOOP vectors are compared to one another to insure that duplicate LOOP vectors

will not be printed. Once this has been accomplished, the following information will be available to the user:

1. The directed LOOP matrix which has as its row vectors the directed LOOP vectors, each element of the row vector representing the node in the correct transmission order as the loop appears in the system graph.
2. The directed IPATH matrix which has as its row vectors the directed IPATH vectors, each element representing the node in the correct transmission order as the path appears in the system graph.

As an example, consider the graph shown in Figure IX. The connection matrix is shown below:

$$IC = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

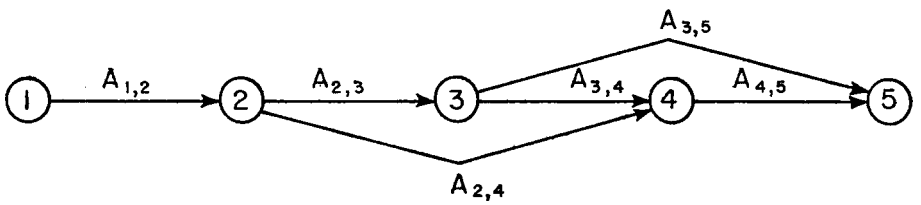
Starting with row 1

$$IPATH(1,1) = 1$$

$$IPATH(1,2) = 2$$

Since there are no other non zero elements, the last stored node in path 1 is node 2. Therefore, row 2 is now examined and

FIGURE IX



FLOW GRAPH FOR DEMONSTRATION OF PATH FINDING ALGORITHM.

IPATH(1,3) = 3

However, node 2 also reaches node 4 on a single branch. Therefore,

IPATH(2,3) = 4

and repeating the path taken to arrive at node 2

IPATH(2,1) = 1

IPATH(2,2) = 2

There are no more non zero elements in row 2. The last stored node in path 1 was node 3. Therefore, checking row 3:

IPATH(1,4) = 4

Node 3 also reaches node 5 on a single branch. Therefore,

IPATH(3,4) = 5

and repeating the path taken to arrive at node 3

IPATH(3,1) = 1

IPATH(3,2) = 2

IPATH(3,3) = 3

Again, the last stored node in path 1 is node 4. Now row 4 is examined and

IPATH(1,5) = 5

Since IPATH(1,) now reaches the output node IPATH(2,) is now called and the last stored node is node 4 again checking row 4.

IPATH(2,4) = 5

Now IPATH(2,)reaches the output node and IPATH(3,) is called

and the last stored node is node 5 and this is the output node. There are no more paths available for Figure IX and the results are shown below:

IPATH(1,1) = 1 IPATH(2,1) = 1 IPATH(3,1) = 1

IPATH(1,2) = 2 IPATH(2,2) = 2 IPATH(3,2) = 2

IPATH(1,3) = 3 IPATH(2,3) = 4 IPATH(3,3) = 3

IPATH(1,4) = 4 IPATH(2,4) = 5 IPATH(3,4) = 5

IPATH(1,5) = 5

The two sections of the program described in the preceding paragraphs make up the algorithms that are used in the finished program. Of course, all the minute details and house keeping quantities that are necessary in the program have not been discussed in this chapter and are not required for an understanding of the algorithms. The actual program listing with appropriate comments is found in the Appendix. The following chapter will demonstrate the capability of the program.

CHAPTER V

This chapter will be concerned only with examples of flow graphs and the use of the program for finding all existing paths and loops within each flow graph. The first two examples will be those used in Chapter II and will be carried through to determination of input-output relationships. However, the remaining examples will be concerned with use of the program only for error-free determination of all loops and paths. The purpose of these examples is to demonstrate the capability of the program.

Example I. Consider the flow graph depicted in Figure II. The nodes will be re-numbered for use by the program. Since the transfer function desired is that between x_5 and x_2, these two nodes will be numbered 1 and 5, respectively. The numbering of the other three nodes is irrelevant as long as none of them is ordered higher than 5. Therefore, assume x_3 is 2, x_1 is 3, and x_4 is 4. The connectivity matrix then follows as

$$IC = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

The computer output is shown on the following page. Using this output and Mr.Coates' equation, the transfer function then follows:

The one connections are $A_{1,4}A_{4,5}A_{2,3}A_{3,2}(-1)^1$ and $A_{1,4}A_{4,5}A_{2,2}A_{3,3}(-1)^2$. These are determined from the directed LOOP matrix and directed PATH matrix. There is only one path ($A_{1,4}A_{4,5}$) transmittance and 2 of the 5 LOOPs touch that path. The LOOP transmittances are, using the LOOP matrix: $A_{2,3}A_{3,2}$; $A_{2,2}$; $A_{3,3}$; $A_{4,4}$; and $A_{5,5}$. The connections are $A_{2,2}A_{3,3}A_{4,4}A_{5,5}(-1)^4$ and $A_{2,3}A_{3,2}A_{4,4}A_{5,5}(-1)^3$. Therefore, the transfer function becomes:

$$\zeta_{1,5} = \frac{A_{1,4}A_{4,5}\ [A_{2,2}A_{3,3}(-1)^2 + A_{2,3}A_{3,2}(-1)^1]}{A_{4,4}A_{5,5}\ [A_{2,2}A_{3,3}(-1)^4 + A_{2,3}A_{3,2}(-1)^3]}$$

$$= \frac{A_{1,4}A_{4,5}}{A_{4,4}A_{5,5}}$$

To show that this is the same result as that found in Chapter II it is only necessary to recall that node 1 corresponds to x_5, node 5 to x_2, node 2 to x_3, node 3 to x_1, and node 4 to x_4. Therefore, $A_{1,4}$ corresponds to $k_{4,5}$, $A_{4,5}$ to $k_{2,4}$, $A_{4,4}$ to $k_{4,4}$ and $A_{5,5}$ to $k_{2,2}$. Thus

$$\zeta_{1,5} = \frac{k_{2,4}k_{4,5}}{k_{2,2}k_{4,4}}$$

```
 CONNECTIVITY,IC,MATRIX
0 1 0 1 0
0 1 1 0 0
0 1 1 0 0
0 0 1 1 1
0 0 0 0 1

 DIRECTED LOOP MATRIX
2 3
2
3
4
5

 DIRECTED PATH MATRIX
1 4 5
```

Example II. Consider the flow graph of Figure V. No change in ordering will be necessary for the desired transfer function is that between node x_1 and node x_5. The connectivity matrix then follows as

$$IC = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

The computer output is shown on the following page. There are two loops whose transmittances are $A_{2,3}A_{3,2}$ and $A_{4,5}A_{5,4}$, respectively. The single path transmittance is $A_{1,2}A_{2,3}A_{3,4}A_{4,5}$. Using Mason's equation the transfer function becomes

$$T = \frac{P_1\Delta_1}{\Delta}$$

where $\Delta_1 = 1$

and $\Delta = 1 - L_1 - L_2 + L_1L_2$

$$\therefore \quad T = \frac{A_{1,2}A_{2,3}A_{3,4}A_{4,5}(1)}{1-A_{2,3}A_{3,2}-A_{4,5}A_{5,4}+A_{2,3}A_{3,2}A_{4,5}A_{5,4}}$$

This equation agrees with that shown in Chapter II as an example of the use of Mason's equation.

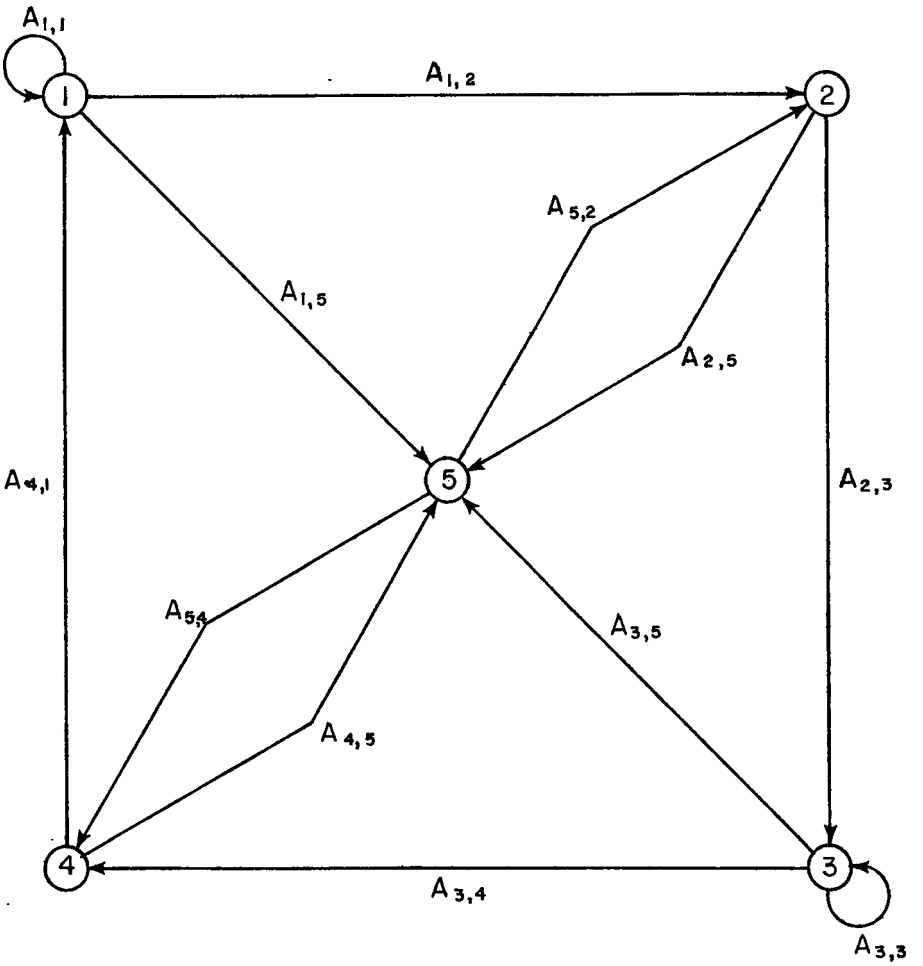
```
 CONNECTIVITY,IC,MATRIX
0 1 0 0 0
0 0 1 0 0
0 1 0 1 0
0 0 0 0 1
0 0 0 1 0

 DIRECTED LOOP MATRIX
2 3
4 5

 DIRECTED PATH MATRIX
1 2 3 4 5
```

Example III. This example will be used to demonstrate the capability of the program. Consider the flow graph of Figure X. This example is taken directly from Mr. D. C. Fielder's paper[10] and was chosen to show the error free approach to determine all existing loops and paths. The connectivity matrix for this graph follows.

$$IC = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

The computer output is shown on the following page. There are 11 loops within this flow graph and the loop transmittances are $A_{1,5}A_{5,2}A_{2,3}A_{3,4}A_{4,1}$, $A_{1,2}A_{2,3}A_{3,5}A_{5,4}A_{4,1}$, $A_{2,3}A_{3,4}A_{4,5}A_{5,2}$, $A_{1,5}A_{5,4}A_{4,1}$, $A_{1,2}A_{2,5}A_{5,4}A_{4,1}$, $A_{2,3}A_{3,5}A_{5,2}$, $A_{1,2}A_{2,3}A_{3,4}A_{4,1}$, $A_{4,5}A_{5,4}$, $A_{2,5}A_{5,2}$, $A_{1,1}$, and $A_{3,3}$. Similarly, the path transmittances are $A_{1,2}A_{2,3}A_{3,4}A_{4,5}$, $A_{1,5}$, $A_{1,2}A_{2,5}$, and $A_{1,2}A_{2,3}A_{3,5}$. The paper referenced for this example does not find the paths existing in the flow graph, however, it is accurate in finding the loops that exist. Its major drawback is that it is a hand method requiring lengthy matrix manipulations. The program presented in this paper accomplishes the same results with no hand or lengthy matrix manipulations. The user is now presented with all the

FIGURE X



FLOW GRAPH FOR SYSTEM OF EXAMPLE III.

```
 CONNECTIVITY,IC,MATRIX
1 1 0 0 1
0 0 1 0 1
0 0 1 1 1
1 0 0 0 1
0 1 0 1 0

 DIRECTED LOOP MATRIX
 1   5   2   3   4
 1   2   3   5   4
 2   3   4   5
 1   5   4
 1   2   5   4
 2   3   5
 1   2   3   4
 4   5
 2   5
 1
 3

 DIRECTED PATH MATRIX
 1   2   3   4   5
 1   5
 1   2   5
 1   2   3   5
```

information he needs to determine the input-output relations for this example.

The following examples will be brief and only involve use of the computer program. Since finding the transfer function for any of the examples is now merely a plug-in operation to one of the gain equations, this operation will be left to the reader.

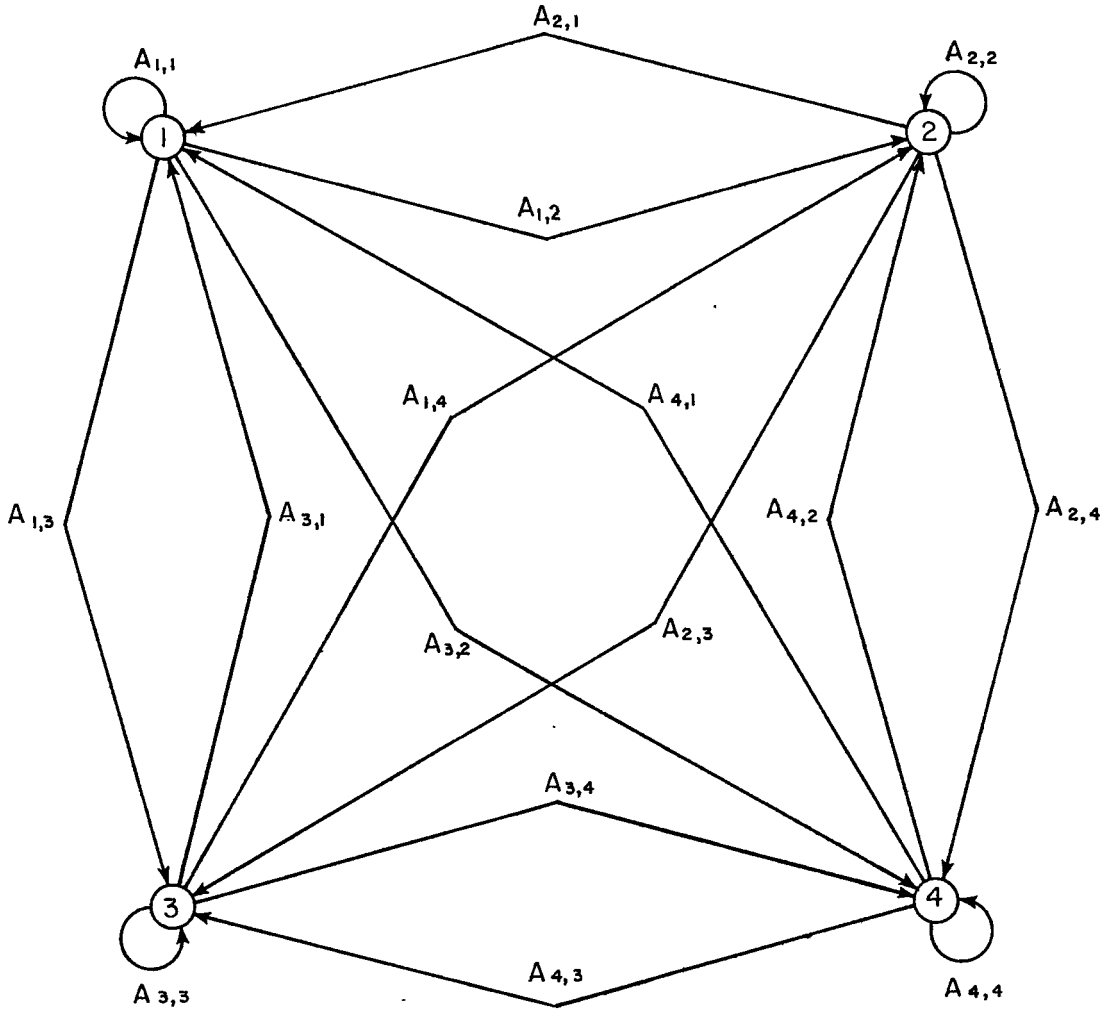
Example IV. This example illustrates a maximal strongly connected system. The flow graph is shown in Figure XI. Every node is connected to every node, including itself. The connectivity matrix is shown below:

$$IC = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 11 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

The computer output is shown on the following page. There are in fact 24 loops and 5 paths. It is implied that the last node in a LOOP vector output is connected to the first node in that vector.

Therefore, a typical LOOP transmittance would be $A_{1,2}A_{2,3}A_{3,4}A_{4,1}$. Similarly, a typical PATH transmittance is $A_{1,2}A_{2,3}A_{3,4}$. The information given in the computer output would enable one to obtain the transfer function for an input at node 1 and an output at node 4.

FIGURE XI



FLOW GRAPH FOR SYSTEM OF EXAMPLE IV.

CONNECTIVITY,IC,MATRIX

```
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

DIRECTED LOOP MATRIX

```
1 2 3 4
1 3 2 4
1 4 2 3
1 2 4 3
1 3 4 2
1 4 3 2
2 3 4
2 4 3
1 3 4
1 4 3
1 2 4
1 4 2
1 2 3
1 3 2
3 4
2 4
2 3
1 4
1 3
1 2
1
2
3
4
```

DIRECTED PATH MATRIX

```
1 2 3 4
1 3 2 4
1 4
1 2 4
1 3 4
```

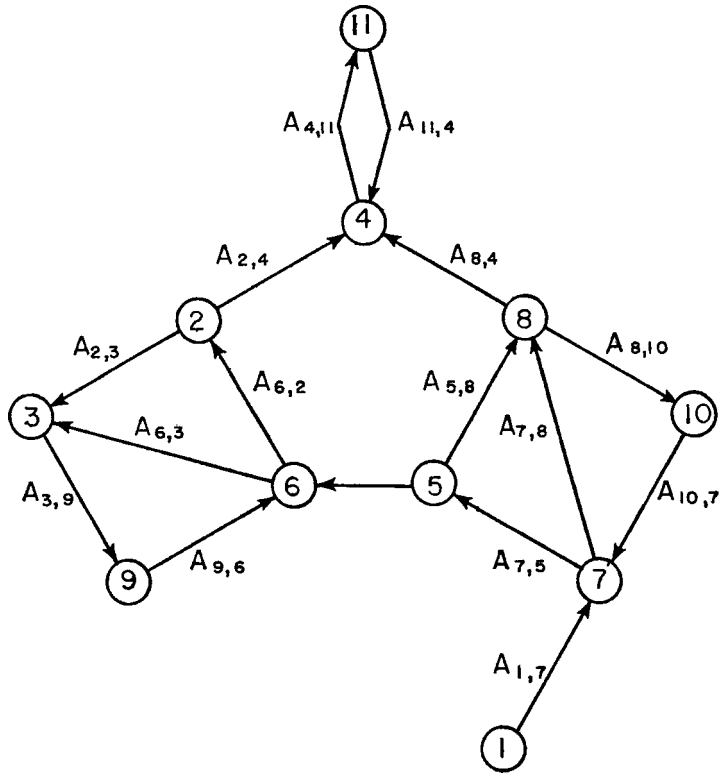
Example V. Consider the flowgraph of Figure XII. This is an 11 node system and is used as an example only for demonstration purposes. The connectivity matrix is shown below:

IC =

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	1	0	0	0	0
2	0	0	1	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	1	0	1	0	0	0
6	0	1	1	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	1	0	0	0
8	0	0	0	1	0	0	0	0	0	1	0
9	0	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0	0
11	0	0	0	1	0	0	0	0	0	0	0

There exist 5 loops and 3 paths in this flow graph and these are shown in the computer output on the following page. This information again will allow the reader to obtain the transfer function for the system having an input and node 1 and the output at node 11.

FIGURE XII



FLOW GRAPH FOR SYSTEM OF EXAMPLE V.

CONNECTIVITY,IC,MATRIX

```
0 0 0 0 0 0 1 0 0 0 0
0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 1 0 0 0
0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0
```

DIRECTED LOOP MATRIX

```
2 3 9 6
411.
5 810 7
3 9 6
7 810
```

DIRECTED PATH MATRIX

```
1 7 8 411
1 7 5 8 411
1 7 5 6 2 411
```

Example VI. This is the final example. The flow graph is shown in Figure XIII. The connectivity matrix for this 16 node system is shown below:

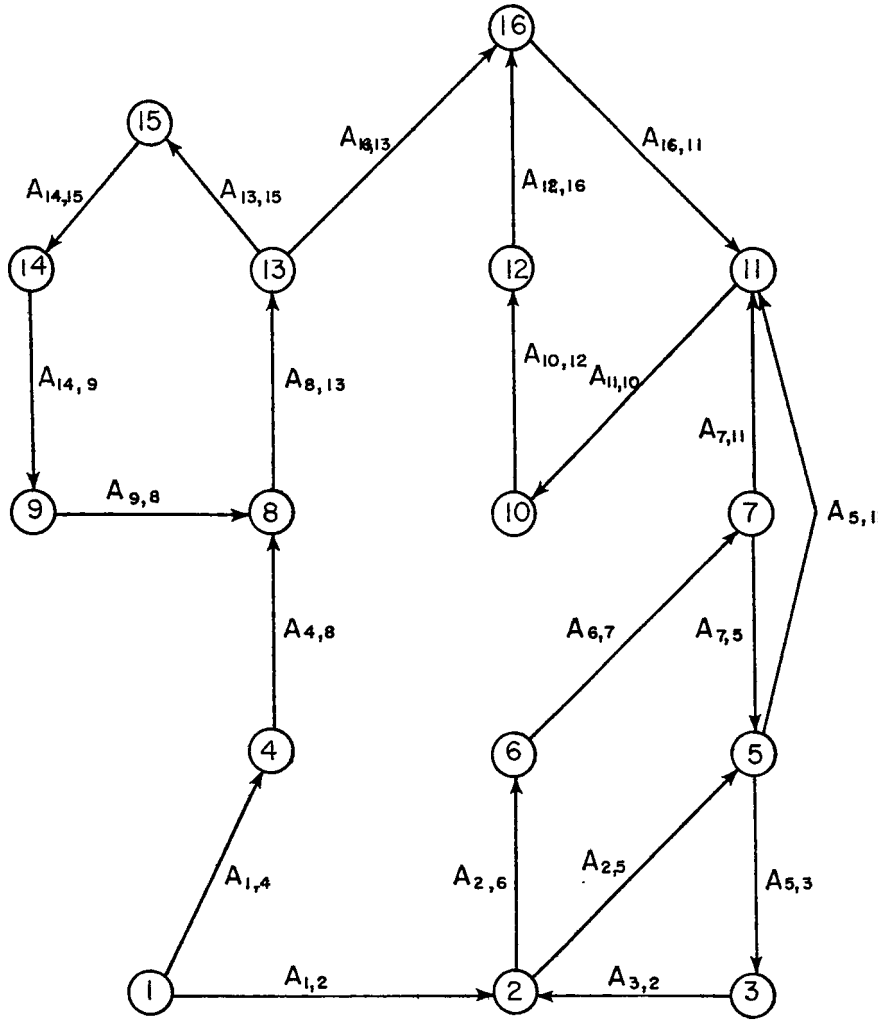
IC=

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
16	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

In this system there are 4 LOOPs and 4 PATHS as shown on the computer output on the following page.

Again, this information would allow the reader to obtain the transfer function for this system with an input at node 1 and an output at node 16. This example along with the preceding examples in this chapter should demonstrate the accuracy and capability of the developed program.

FIGURE XIII



FLOW GRAPH FOR SYSTEM OF EXAMPLE VI.

CONNECTIVITY,IC,MATRIX

```
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
```

DIRECTED LOOP MATRIX

```
 2  6  7  5  3
 8 13 15 14  9
10 12 16 11
 2  5  3
```

DIRECTED PATH MATRIX

```
 1  2  5 11 10 12 16
 1  4  8 13 16
 1  2  6  7 11 10 12 16
 1  2  6  7  5 11 10 12 16
```

CHAPTER VI

CONCLUSIONS

Techniques for determining directed loops and paths within a flow graph are developed. Through the use of these techniques input-output relations may be found for much larger systems than are usually investigated by flow graph methods. Any of the techniques described in Chapter II may be applied once the loop and path components of the graph are determined. It is not felt that the restrictions placed on the use of the program are unduly prohibitive.

The problems encountered in attempting to apply this approach to the hybrid case have not been discussed. However, several suggestions will be offered as an approach that might be used to determine paths and loops of both types, i.e., those that are strictly continuous and those that contain at least one discrete variable. It should be pointed out that the case involving all discrete variables is no different than the case involving all continuous variables due to the definition of a hybrid loop or path described in Chapter II. The theory of superposition could be applied in considering the hybrid case. The input data could be in the form of 2 connection matrices, one containing only the continuous variables and one containing all the variables, continuous and discrete. Using these two matrices

all the continuous variable paths and loops could readily be identified and all the paths and loops containing at least one discrete variable could be obtained for those loops and paths that do not encounter the same variable more than once. Additional programming could compare loops and paths found from the continuous connection matrix with those found from the hybrid connection matrix. Those loops and paths that appear in both sets could be eliminated from those found from the hybrid connection matrix leaving only those that contain at least one discrete variable. The real problem that has to be solved is that of those paths and loops of Type 2 (containing at least one discrete variable) that pass through a continuous variable more than once. These types of paths and loops are permitted in accordance with Mr. Bekey's definition as stated in Chapter II. It is believed that with a little more effort this approach could be completed and a general program could be written that could handle all classes of signal flow graphs. However, that was beyond the scope of this research.

A digital computer program has been developed that will identify every loop that exists within a given flow graph as well as every path from input to output. This program will handle large systems as long as the systems are either all continuous or all discrete. Also, this program lends itself to multi-input-output problems in

that all that is necessary is to shift the ordering of the variables within the system such that the new input is the first ordered node and/or the new output is the highest ordered node. It is believed that this digital approach to the identification of paths and loops within a flow graph is simple and very useful.

REFERENCES

1. Mason, S. J., "Feedback Theory - Som Properties of Signal Flow Graphs," Proc. IRE, Vol. 41, pp. 1144-1156, September, 1953.

2. Mason, S. J., "Feedback Theory - Further Properties of Signal Flow Graphs," Proc. IRE, Vol. 44, pp. 920-926, July, 1956.

3. Coates, C. L., "Flow Graph Solutions of Linear Algebraic Equations," IRE Trans. on Circuit Theory, Vol. CT-6, pp. 170-187, June, 1959.

4. Desoer, C. A., "The Optimum Formula for the Gain of a Flow Graph or a Simple Derivation of Coates' Formula," Proc. IRE, Vol. 48, pp. 883-889, May, 1960.

5. Mason, S. J., "About Such Things as Unistors, Flow Graphs, Probability, Partial Factoring, and Matrices," IRE Trans. on Circuit Theory, pp. 90-97, September, 1957.

6. Bekey, G. A., and M. Sedlar, "Signal Flow Graphs of Sampled Data Systems: A New Formulation," IEEE Trans. on Automatic Control, Vol. AC-12, No. 2, pp. 154-161, April, 1967.

7. Salzer, J. M., "Signal Flow Reduction in Sampled Data Systems," IRE Wescon Conv. Rec., Pt. 4, pp. 166-169, 1957.

8. Hoskins, R. F., "Signal Flow Graph Analysis and Feedback Theory," The Institute of Electrical Engineers, Monograph No. 388E, July, 1960.

9. Ramamoorthy, C. V., "Analysis of Graphs by Connectivity Considerations," Journal of the Association for Computing Machinery, Vol. 13, No. 2, pp. 211-222, April, 1966.

10. Fielder, D. C., "An Approach to Error-Free Flow Graph Equations," IEEE Trans. on Education, pp. 233-237, December, 1967.

11. Lendaris, G. G. and E. I. Jury, "Input-Output Relationships for Multiloop Sampled Systems," Trans. AIEE (Application and Industry), pp. 375-385, January, 1960.

12. Ash, R., W. H. Kim, and G. M. Kranc, "A General Flow Graph Technique for Solution of Multiloop Sampled Systems," J. Basic Engrg. Trans., ASME, pp. 360-370, June, 1960.

APPENDIX

```
C      PURPOSE
C       DETERMINE ALL SUBLOOPS AND PATHS THAT EXIST
C      WITHIN A SIGNAL FLOW GRAPH
       INTEGER*2 IC(20,20),IR(20,20),IM(20,20),IB(20),KB(20),LNODE(40),
      1IMLNDE(40),ITEST(20,20),IEXIT(20),ITROW(20),ITCOL(20),
      1IMLOOP(40,20),LOOP(40,20)/800*0/,ITLOOP(40,20),ISLOOP(20),
      1IPATH(40,20)/800*0/,IPNODE(40)
C      DESCRIPTION OF MATRICES
C        IC-CONNECTION MATRIX FOR MAIN SIGNAL FLOW GRAPH
C        IR-REACHABILITY MATRIX - THIS MATRIX AREA IS
C       USED MORE THAN ONCE - IT DETERMINES THE
C       REACHABILITY FOR THE SIGNAL FLOW GRAPH OR
C       ANY SUBGRAPH THAT APPEARS WITHIN THE SIGNAL
C       FLOW GRAPH
C        IM-MATRIX CONTAINING UNIQUE ROW VECTORS
C       CORRESPONDING TO SUBGRAPHS THAT EXIST WITHIN
C       THE SIGNAL FLOW GRAPH OR ANY OTHER SUBGRAPH
C       OF THE SYSTEM
C        IB-COUNTER USED IN CONSTRUCTION OF IR MATRIX
C        KB-COUNTER USED IN CONSTRUCTION OF IM MATRIX
C        LNODE-CONTAINS NUMBER OF NODES THAT APPEAR
C       IN A GIVEN LOOP
C        IMLNDE-CONTAINS NUMBER OF NODES THAT APPEAR
C       IN A POSSIBLE LOOP CONFIGURATION
C        ITEST-A MATRIX THAT CORRESPONDS TO A CONNECTION
C       MATRIX FOR ANY POSSIBLE SUBGRAPH. THIS MATRIX IS
C       CREATED AS NEEDED FOR EACH POSSIBLE SUBGRAPH AND
C       AFTER IT IS USED IT IS DESTROYED
C        IEXIT-CONTAINS THE NUMBER OF EXITS FOR A GIVEN NODE. AFTER
C       IT IS USED IT IS DESTROYED TO BE USED AGAIN LATER.
C        ITROW-USED TO TEMPORARILY STORE A ROW VECTOR FROM
C       THE ITEST MATRIX UNDER INVESTIGATION
C        ITCOL-USED TO TEMPORARILY STORE A COLUMN VECTOR
C       FROM THE ITEST MATRIX UNDER INVESTIGATION
C        IMLOOP-THIS MATRIX CONTAINS ROW VECTORS OF NODES
C       WHICH MAY BE IN A LOOP AND MAY CONTAIN INSIDE
C       LOOPS OF FEWER NODES
C        LOOP-THIS MATRIX CONTAINS ROW VECTORS OF ORDERED
C       NODES AS THEY APPEAR IN A LOOP WITHIN THE SYSTEM
C        ITLOOP-THIS MATRIX CONTAINS ROW VECTORS OF NODES
C       THAT MAY BE IN A LOOP
C        ISLOOP-THIS CONTAINS THOSE NODES HAVING SELF LOOPS
C       AND ARE ADDED TO THE LOOP MATRIX UPON COMPLETION OF
C       THE LOOP FINDING ALGORITHM
C        IPNODE-CONTAINS THE NUMBER OF NODES THAT APPEAR
C       IN A GIVEN PATH
C        IPATH-CONTAINS ROW VECTORS CORRESPONDING TO ORDERED
C       NODES AS THEY APPEAR IN A PATH FROM THE INPUT NODE
C       TO THE OUTPUT NODE
C        IPCNT-CONTAINS THE NUMBER OF PATHS THAT EXIST
C       WITHIN THE SYSTEM
C        LCOUNT-CONTAINS THE NUMBER OF LOOPS THAT EXIST
C       WITHIN THE SYSTEM
C        IMLCNT-CONTAINS THE NUMBER OF IMLOOP VECTORS FOUND
       READ(5,1)N
       READ(5,2)((IC(K,I),I=1,N),K=1,N)
     1 FORMAT(I2)
     2 FORMAT(40I2)
       WRITE(6,15)
```

```
   15 FORMAT('1',1X,'CONNECTIVITY,IC,MATRIX')
      DO 16 J=1,N
   16 WRITE(6,10)(IC(J,IK),IK=1,N)
   10 FORMAT(20I2)
      LIM=40
C     DETERMINE NODES HAVING SELF LOOPS AND STORE THEM IN
C      ISLOOP VECTOR
      ISLCNT=0
      DO 87 I=1,N
      IF(IC(I,I).EQ.1)GO TO 88
   87 CONTINUE
      GO TO 89
   88 ISLCNT=ISLCNT+1
      ISLOOP(ISLCNT)=I
      IC(I,I)=0
      GO TO 87
C     CALCULATE IR MATRIX FOR MAIN SYSTEM
   89 DO 3 K=1,N
      DO 4 I=1,N
      IB(I)=0
    4 IR(K,I)=IC(K,I)
      DO 3 M=1,N
      DO 3 I=1,N
      IF(IR(K,I).NE.0)GO TO 5
      GO TO 3
    5 IF(IB(I).NE.0)GO TO 3
      IB(I)=1
      II=I
      DO 6 J=1,N
      IX=IR(K,J)+IC(II,J)
      IF(IX.NE.0)GO TO 7
      IR(K,J)=0
      GO TO 6
    7 IR(K,J)=1
    6 CONTINUE
    3 CONTINUE
      DO 11 J=1,N
C     CALCULATE IM MATRIX FOR MAIN SYSTEM
      DO 12 I=1,N
      KB(I)=0
   12 IM(J,I)=IR(I,J)
      DO 11 M=1,N
      DO 11 I=1,N
      IF(IM(J,I).NE.0)GO TO 13
      GO TO 11
   13 IF(KB(I).NE.0)GO TO 11
      KB(I)=1
      II=I
      DO 11 K=1,N
      IY=IM(J,K)+IR(II,K)
      IF(IY.NE.2)GO TO 14
      IM(J,K)=1
      GO TO 11
   14 IM(J,K)=0
   11 CONTINUE
      IMLCNT=0
      K=1
      NN=N-1
      I=1
```

```
   17 M=0
C     DETERMINE UNIQUE ROW VECTORS OF IM MATRIX. THESE ARE
C      POSSIBLE LOOP VECTORS BUT MAY CONTAIN SMALLER LOOPS
C      INSIDE
      DO 18 J=1,N
      IF(IM(I,J).EQ.1)GO TO 19
   18 CONTINUE
      IF(M.GT.1)GO TO 20
      IF(IMLCNT.GT.0)GO TO 21
      I=I+1
      IF(I.GT.NN)GO TO 22
      GO TO 17
   19 M=M+1
      IMLOOP(K,M)=J
      GO TO 18
   20 IMLCNT=IMLCNT+1
      IF(IMLCNT.GT.LIM)GO TO 126
      GO TO 127
  126 WRITE(6,128)
  128 FORMAT('0',1X,'IMLOOP HAS EXCEEDED LIM, REDIMENSION IMLOOP')
      GO TO 120
  127 IMLNDE(IMLCNT)=M
      K=K+1
   21 I=I+1
      DO 23 IK=1,IMLCNT
      IJM=IMLNDE(IK)
      DO 23 IV=1,IJM
      NUM=IMLOOP(IK,IV)
      IF(I.EQ.NUM)GO TO 21
   23 CONTINUE
      IF(I.GT.NN)GO TO 22
      GO TO 17
   22 IF(IMLCNT.GT.0)GO TO 24
      GO TO 25
   24 K=1
   28 NONODE=IMLNDE(K)
C     BEGIN DETERMINATION OF SMALLER ADDITIONAL POSSIBLE
C      LOOPS BY FORMING ITEST MATRIX FOR A GIVEN IMLOOP
C      VECTOR
      DO 29 I=1,NONODE
      DO 29 J=1,NONODE
      II=IMLOOP(K,I)
      JJ=IMLOOP(K,J)
   29 ITEST(I,J)=IC(II,JJ)
C     CHECK NUMBER OF EXITS FOR EACH NODE OF ITEST MATRIX
      DO 30 I=1,NONODE
      IEXIT(I)=0
      DO 31 J=1,NONODE
   31 IEXIT(I)=IEXIT(I)+ITEST(I,J)
C     IF EXIT IS NOT GREATER THAN ONE THERE CAN BE NO
C      INSIDE LOOP, INVESTIGATE NEXT IMLOOP VECTOR
C     IF EXIT IS GREATER THAN ONE TEMPORARILY STORE FIRST
C      ROW VECTOR OF ITEST MATRIX IN ITROW VECTOR AND THEN
C      ZERO FIRST ROW VECTOR OF ITEST MATRIX. ALSO DO THE
C      SAME FOR THE FIRST COLUMN VECTOR USING ITCOL VECTOR.
C      NOW NEW IR AND IM MATRICES ARE OBTAINED, CREATING
C      NEW IMLOOP VECTORS. THIS OPERATION IS TO BE REPEATED
C      FOR EVERY NODE OF ITEST MATRIX WITH ONE NODE REMOVED
C      AT A TIME. OF COURSE AFTER EACH INVESTIGATION THE
```

```
C       DELETED ROW AND COLUMN VECTORS ARE REPLACED BEFORE
C       DELETING THE NEXT PAIR
        IF(IEXIT(I).GT.1)GO TO 32
   30   CONTINUE
        K=K+1
        IF(K.GT.IMLCNT)GO TO 25
        GO TO 28
   32   DO 33 I=1,NONODE
        DO 34 IJ=1,NONODE
        ITROW(IJ)=ITEST(I,IJ)
   34   ITCOL(IJ)=ITEST(IJ,I)
        DO 35 J=1,NONODE
        ITEST(I,J)=0
   35   ITEST(J,I)=0
        DO 36 IK=1,NONODE
        DO 37 II=1,NONODE
        IB(II)=0
   37   IR(IK,II)=ITEST(IK,II)
        DO 36 IN=1,NONODE
        DO 36 II=1,NONODE
        IF(IR(IK,II).NE.0)GO TO 38
        GO TO 36
   38   IF(IB(II).NE.0)GO TO 36
        IB(II)=1
        IJ=II
        DO 39 JJ=1,NONODE
        IX=IR(IK,JJ)+ITEST(IJ,JJ)
        IF(IX.NE.0)GO TO 40
        IR(IK,JJ)=0
        GO TO 39
   40   IR(IK,JJ)=1
   39   CONTINUE
   36   CONTINUE
        DO 41 IJ=1,NONODE
        DO 42 II=1,NONODE
        KB(II)=0
   42   IM(IJ,II)=IR(II,IJ)
        DO 41 IN=1,NONODE
        DO 41 II=1,NONODE
        IF(IM(IJ,II).NE.0)GO TO 43
        GO TO 41
   43   IF(KB(II).NE.0)GO TO 41
        KB(II)=1
        III=II
        DO 41 IK=1,NONODE
        IY=IM(IJ,IK)+IR(III,IK)
        IF(IY.NE.2)GO TO 44
        IM(IJ,IK)=1
        GO TO 41
   44   IM(IJ,IK)=0
   41   CONTINUE
        IK=IMLCNT+1
        IIB=0
        II=1
   47   M=0
        DO 48 J=1,NONODE
        IF(IM(II,J).EQ.1)GO TO 49
   48   CONTINUE
        IF(M.GT.1)GO TO 50
```

```
      IF(IIB.EQ.0)GO TO 145
      GO TO 53
  145 II=II+1
   51 IF(II.GT.NONODE)GO TO 52
      GO TO 47
   49 M=M+1
      IMLOOP(IK,M)=J
      GO TO 48
   50 IMLCNT=IMLCNT+1
      IF(IMLCNT.GT.LIM)GO TO 126
      IMLNDE(IK)=M
      IIB=IIB+1
      IK=IK+1
      ILID=IK-II3
   53 II=II+1
      DO 54 IIK=ILID,IMLCNT
      INUM=IMLNDE(IIK)
      DO 54 IV=1,INUM
      IP=IMLOOP(IIK,IV)
      IF(II.EQ.IP)GO TO 53
   54 CONTINUE
      GO TO 51
   52 IF(IIB.GT.0)GO TO 55
      GO TO 56
   55 DO 57 J=ILID,IMLCNT
      IV=IMLNDE(J)
      DO 57 IJ=1,IV
      IA=IMLOOP(J,IJ)
   57 IMLOOP(J,IJ)=IMLOOP(K,IA)
   56 DO 60 IJ=1,NONODE
      ITEST(I,IJ)=ITROW(IJ)
   60 ITEST(IJ,I)=ITCOL(IJ)
   33 CONTINUE
      K=K+1
      IF(K.GT.IMLCNT)GO TO 25
      GO TO 28
   25 ICP=0
      LCCUNT=0
      K=1
   61 NONODE=IMLNDE(K)
C     AT THIS POINT ALL POSSIBLE LOOPS HAVE BEEN STORED AS
C      IMLOOP VECTORS AND IT IS NOW NECESSARY TO ESTABLISH
C      IF THEY ARE IN FACT LOOPS AND IF SO ORDER THEM
C     THE ITEST MATRIX IS AGAIN FORMED FOR EACH IMLOOP
C      VECTOR AS IT IS INVESTIGATED
      DO 62 I=1,NONODE
      DO 62 J=1,NONODE
      II=IMLOOP(K,I)
      JJ=IMLOOP(K,J)
   62 ITEST(I,J)=IC(II,JJ)
C     ITLOOP VECTORS ARE CONSTUCTED AS ORDERED NODES AS THE
C      NONZERO ELEMENTS OF THE ITEST MATRIX ARE FOUND. NO NODE
C      MAY APPEAR MORE THAN ONCE IN ANY LOOP AND THE LAST NODE
C      IN THE LOOP MUST BE CONNECTED TO THE FIRST NODE IN THAT
C      LOOP. EACH LOOP CONTAINS THE SAME NUMBER OF NODES AS
C      THE IMLOOP VECTOR FROM WHICH THE LOOP IS FOUND, NO MORE
C      AND NO LESS
      DO 121 I=1,LIM
      DO 121 J=1,NONODE
```

```
121 ITLOOP(I,J)=0
    I=1
    ITLC=1
    M=0
    IL=1
    IK=2
    ITLOOP(1,1)=1
 63 DO 64 J=1,NONODE
    IF(ITEST(I,J).EQ.1)GO TO 65
    GO TO 64
 65 M=M+1
    IF(M.EQ.IL)GO TO 66
    M=ITLC+1
    ITLOOP(M,IK)=J
    IMN=IK-1
    DO 67 KK=1,IMN
 67 ITLOOP(M,KK)=ITLOOP(IL,KK)
    DO 68 JK=1,IMN
    IF(ITLOOP(M,IK).EQ.ITLOOP(M,JK))GO TO 64
 68 CONTINUE
    ITLC=M
    IF(ITLC.GT.LIM)GO TO 123
    GO TO 64
123 WRITE(6,125)
125 FORMAT('0',1X,'POSSIBLE LOOPS EXCEED LIM, REDIMENSION ITLOOP')
    GO TO 120
 66 ML=0
    ITLOOP(M,IK)=J
    ID=IK-1
    DO 69 JK=1,ID
    IF(ITLOOP(M,IK).EQ.ITLOOP(M,JK))GO TO 70
 69 CONTINUE
    IF(IK.EQ.NONODE)GO TO 71
    GO TO 64
 70 M=M-1
    ML=1
 64 CONTINUE
    IF(ML.EQ.1)GO TO 72
    M=IL
    I=ITLOOP(M,IK)
    M=M-1
    IK=IK+1
    IF(IK.GT.NONODE)GO TO 71
    GO TO 63
 72 M=M+1
 71 IL=M+1
    M=IL
    IF(M.GT.ITLC)GO TO 73
    DO 74 JK=1,NONODE
    IF(ITLOOP(M,JK).EQ.0)GO TO 75
 74 CONTINUE
    GO TO 71
 75 IK=JK-1
    KK=IK-1
    DO 76 IBC=1,KK
    IF(ITLOOP(M,IK).EQ.ITLOOP(M,IBC))GO TO 71
 76 CONTINUE
    I=ITLOOP(M,IK)
    IF(IK.GT.NONODE)GO TO 71
```

```
      M=M-1
      IK=IK+1
      GO TO 63
   73 DO 77 I=1,ITLC
      MC=ITLOOP(I,NONODE)
      IF(ITEST(MC,1).EQ.1)GO TO 78
      GO TO 77
   78 IAD=NONODE-1
      DO 160 JJ=1,IAD
      IF(ITLOOP(I,NONODE).EQ.ITLOOP(I,JJ))GO TO 77
  160 CONTINUE
      ICP=ICP+1
C     THE LOOP VECTOR IS NOW FORMED USING THE ITLOOP VECTOR AND
C      ITS CORRESPONDING IMLOOP VECTOR. THERE MAY BE MORE THAN
C      ONE ITLOOP VECTOR FOR ANY GIVEN IMLOOP VECTOR
      DO 80 IJ=1,NONODE
      IA=ITLOOP(I,IJ)
   80 LOOP(ICP,IJ)=IMLOOP(K,IA)
      LCOUNT=LCOUNT+1
      IF(LCOUNT.GT.LIM)GO TO 129
      GO TO 130
  129 WRITE(6,131)
  131 FORMAT('0',1X,'LCOUNT HAS EXCEEDED LIM, REDIMENSION LOOP')
      GO TO 120
  130 LNODE(LCOUNT)=IMLNDE(K)
   77 CONTINUE
   79 K=K+1
      IF(K.GT.IMLCNT)GO TO 81
      GO TO 61
   81 IF(ISLCNT.GT.0)GO TO 82
      GO TO 83
   82 DO 84 I=1,ISLCNT
      LCOUNT=LCOUNT+1
      IF(LCOUNT.GT.LIM)GO TO 129
      LOOP(LCOUNT,1)=ISLOOP(I)
   84 LNODE(LCOUNT)=1
   83 IF(LCOUNT.GT.1)GO TO 156
      IF(LCOUNT.GT.0)GO TO 155
      WRITE(6,157)
  157 FORMAT('0',1X,'THERE ARE NO LOOPS')
      GO TO 158
  156 IIN=1
  146 KILT=IIN+1
  147 IED=LCOUNT
      DO 143 J=KILT,IED
      IF(LNODE(IIN).EQ.LNODE(J))GO TO 149
      GO TO 148
  149 IEA=LNODE(IIN)
      DO 150 II=1,IEA
      IF(LOOP(IIN,II).NE.LOOP(J,II))GO TO 148
  150 CONTINUE
      DO 151 IJ=1,IEA
  151 LOOP(J,IJ)=0
      IF(J.EQ.LCOUNT)GO TO 152
      IEB=LCOUNT-1
      DO 153 KI=J,IEB
      KK=KI+1
      LNODE(KI)=LNODE(KK)
      IEC=LNODE(KK)
```

```
      DO 153 IJ=1,IEC
  153 LOOP(KI,IJ)=LOOP(KK,IJ)
      LCOUNT=LCOUNT-1
      KILT=J
      GO TO 147
  152 LCOUNT=LCOUNT-1
      GO TO 154
  148 CONTINUE
  154 IIN=IIN+1
      IEF=LCOUNT-1
      IF(IIN.GT.IEF)GO TO 155
      GO TO 146
  155 WRITE(6,85)
   85 FORMAT('0',1X,'DIRECTED LOOP MATRIX')
      DO 86 K=1,LCOUNT
      ICP=LNODE(K)
   86 WRITE(6,117)(LOOP(K,M),M=1,ICP)
  117 FORMAT(2013)
C     THE IPATH VECTORS ARE OBTAINED FROM THE ORIGINAL IC MATRIX
C      BY INVESTIGATING THE ROW VECTORS OF THE IC MATRIX FOR
C      NONZERO ELEMENTS. THE ELEMENTS (NODES) OF THE IPATH VECTORS
C      ARE ORDERED AS THEY ARE FOUND. AGAIN NO NODE MAY APPEAR
C      MORE THAN ONCE IN ANY PATH
  158 I=1
      IPCNT=1
      M=0
      IL=1
      K=2
      IPATH(1,1)=1
  100 DO 90 J=1,N
      IF(IC(I,J).EQ.1)GO TO 91
      GO TO 90
   91 M=M+1
      IF(M.EQ.IL)GO TO 92
      M=IPCNT+1
      IPATH(M,K)=J
      NM=K-1
      DO 93 KK=1,NM
   93 IPATH(M,KK)=IPATH(IL,KK)
      DO 94 IK=1,NM
      IF(IPATH(M,K).EQ.IPATH(M,IK))GO TO 90
   94 CONTINUE
      IPCNT=M
      IF(IPCNT.EQ.LIM)GO TO 104
      GO TO 90
   92 ML=0
      IPATH(M,K)=J
      ID=K-1
      DO 96 IK=1,ID
      IF(IPATH(M,K).EQ.IPATH(M,IK))GO TO 97
   96 CONTINUE
      IF(IPATH(M,K).EQ.N)GO TO 98
      GO TO 90
   97 M=M-1
      ML=1
   90 CONTINUE
      IF(ML.EQ.1)GO TO 99
      M=IL
      I=IPATH(M,K)
```

```
      M=M-1
      K=K+1
      IF(K.GT.N)GO TO 98
      GO TO 100
   99 M=M+1
   98 IL=M+1
      M=IL
      IF(M.GT.IPCNT)GO TO 105
      DO 101 IK=1,N
      IF(IPATH(M,IK).EQ.0)GO TO 102
  101 CONTINUE
      GO TO 98
  102 K=IK-1
      KK=K-1
      DO 103 IBA=1,KK
      IF(IPATH(M,K).EQ.IPATH(M,IBA))GO TO 98
  103 CONTINUE
      I=IPATH(M,K)
      IF(IPATH(M,K).EQ.N)GO TO 98
      M=M-1
      K=K+1
      GO TO 100
  104 WRITE(6,106)
  106 FORMAT('0',1X,'PATHCOUNT HAS REACHED LIM, REASSIGN')
      GO TO 120
  105 KIL=1
  107 DO 108 IJM=KIL,IPCNT
      DO 109 IK=1,N
      IF(IPATH(IJM,IK).EQ.N)GO TO 111
  109 CONTINUE
      DO 110 IJ=1,N
  110 IPATH(IJM,IJ)=0
      IF(IJM.EQ.IPCNT)GO TO 112
      IMA=IPCNT-1
      DO 113 KI=IJM,IMA
      KK=KI+1
      DO 113 J=1,N
  113 IPATH(KI,J)=IPATH(KK,J)
      IPCNT=IPCNT-1
      KIL=IJM
      GO TO 107
  111 IPNODE(IJM)=IK
  108 CONTINUE
      GO TO 114
  112 IPCNT=IPCNT-1
  114 WRITE(6,115)
  115 FORMAT('0',1X,'DIRECTED PATH MATRIX')
      DO 116 K=1,IPCNT
      ICP=IPNODE(K)
  116 WRITE(6,117)(IPATH(K,I),I=1,ICP)
  120 STOP
      END
```