THE ANALYSIS OF FLUID NETWORKS

by

Shadrack Nelson Taylor

A thesis presented at the University of Aston
in Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

February 1979

The Analysis of Fluid Networks

Shadrack Nelson Taylor

for the degree of Doctor of Philosophy, 1979

Summary

A review of both analogue and digital computer methods applied in the solution of the fluid network problem precedes a discussion of computational aspects and a description of the matric topological methods used. A new approach to the problem by ordering according to a Hamiltonian route is considered and a special matrix Λ defined for the purpose. Some properties of Λ are investigated and its operational aspects developed using newly created algorithms, especially one for finding such a route in a network. The use of Λ is explored in this context by applying it to the solution of recently published network examples. The computed results are described in some detail, together with the relevant programs.

Key words: Network, graph theory, Hamiltonian path.

ACKNOWLEDGEMENTS

The Directors and the Governors of the Polytechnic, Wolverhampton

for their sponsorship, kind support and facilities provided in connection with this research.

Professor G.V. Jeffreys

for providing research facilities in the Department of Chemical Engineering, the University of Aston in Birmingham.

Dr. B. Gay

for his supervision of this project and the many helpful and constructive comments in the preparation of this thesis.

Computer Centre Staffs

at the Polytechnic, Wolverhampton and the University of Aston in Birmingham for their assistance with processing the computer programs involved with this research.

CONTENTS

Summary

Acknowledgements

		Page
Chapter 1	Introduction	1
Chapter 2	Recent Solution by Computer	5
Chapter 3	Computational Aspects	15
Chapter 4	A New Ordering Approach; Hamiltonian Path Considerations	21
Chapter 5	Some Mathematical Properties of ρ	29
Chapter 6	The Application of ρ in the Network Problem	32
Chapter 7	The Non-Linear Flow Problem	38
Chapter 8	User's Guide to the Chained Algorithms	41
Chapter 9	Results, Discussion, Comparison	45
Chapter 10	Conclusion, Summary of Theory and Further Research	49
APPENDIX	Regula Falsi	57
	Algorithm DIR	58
	- WHE	61
	- MR	65
	- ATOM	70
	- MAL	76
	- BREN	81
	- HAMIL	86
	- DISP	93
	- COL1	97
	- HAMIL9	99
	- DISP3	101
EXAMPLES		103
Bibliography and References		118
Nomenclature		123
Index		125

CHAPTER 1

Introduction

The importance of the connections of systems has long been a preponderating and preoccupying concern of mankind. It has appeared in mythology as a problem in the labyrinth of Knossos and a practical solution with Hercules cleansing the Augean stables. It has held sway in the Roman Empire with the aqueducts and the sewer system which Gibbon (1.1, 1.2) says '...rank....among the noblest monuments of Roman genius and power' and describes as 'stupendous'. The Roman construction of roads linked the Empire for rapid communication, transit of troops and conveyance of tributes to Rome. Navigation and exploration gave rise to cartography and metal mines required roadway enumeration in the course of ore extraction.

As time passed it became apparent that multiplicity meant complexity and that obtaining a solution often resulted in an approximation that was barely tolerable. Today the general network problem is of greater significance than ever, embracing heart by-passes, motorway interchanges, integrated circuit layouts, timetables, bridge construction, chemical process plants, mine ventilation, distribution services, communication and computer links, neurological considerations and the like.

The purpose of this research is to investigate in the field of fluid network analysis with a view to devising improved techniques, implementing new computer methods, reducing repetitious and tedious data handling and compacting the concepts.

Historically the significance of the difficulties became apparent when the problem was expressed analytically. The use of mathematics in its predictive function confronted the network practitioners with the realisation that analytical solutions did not exist, that new theories were needed and that new fields had to be opened before achieving success. The response was surprising and incredible. The most famous is that of Kirchhoff (1.3) who produced the laws of conservation of current at a node and zero sum of potential drops and rises round a circuit. These have been called 'First' and 'Second' but were indiscriminately referred to by Kirchhoff himself (1.4) and are more accurately called Kirchhoff's Current Law (KCL) and Kirchhoff's Voltage Law (KVL). Apart from the Axiom of Connectivity they form the basis of Graph Theory, which developed a century later. The obvious analogy between KCL, KVL and pipe flow was soon discovered. It is observed that the former becomes a conservation of mass law, the latter a summation of pressure differences while the actual disparity, the electromagnetic field, is often overlooked. The Kirchhoff laws, however, are associated with the graph rather than with the electric phenomena.

On a practical note it is surprising that a mining engineer, Atkinson (1.5), put forward a mathematical solution for flow in parallel branches. He had arrived at the problem when he realised that 'coursing' the air through mine workings (i.e. serially) was inferior to 'splitting' the air down parallel branches. The flow being non-linear meant that the root could not be found immediately for the two simultaneous equations. He approximated by using a chord on the curve, the 'Regula Falsi' (1.6) method (cf Appendix). The procedure has been used for a good first approximation before proceeding with, say, the Newton-Raphson method (1.8). Atkinson's name was commemorated by using it for a unit of mine airway resistance, now superseded by the Gaul in Système International d'Unités.

The seminal paper for pipe flow was undoubtedly that of Hardy Cross (1.9). He assumed a small mesh error in the flow, expanded binomially and calculated it -- a first approximation method used by Newton. The process was carried out for each mesh of the network then the calculated errors were re-applied to the branches; some branches were in several meshes and so would have several corrections. This paper was on city water supply and distribution of water in buildings.

The solution of electrical problems had advanced considerably by this time, the difficulties of pipe and pipe network problems were just being realised. The electrical cases were involved with essentially linear elements i.e. the current was proportional to the voltage applied. For pipes the resistance varied with the velocity and so the work of Kármán and Prandtl (1.10) and others indicated that regimes of flow would affect the resistance in different ways. A variety of formulas generated from laboratory data appeared. The Americans preferred something like the Hazen-Williams formula (1.11) and used an index of flow of 1.852 for Q. In mining engineering this approach was also adopted by the Hay Committee (1.12) in 1924; the index was taken as 2, though the practical variation is between about 1.85 and 2.2. There are other difficulties here, connected with the inaccurate measurement of velocity and the rapid decrease of cross-sectional area.

Two classes of analogue computer were used in the period 1930-50. The first type was electrical for modelling fluid flow, the second was hydraulic for simulating air flow (1.7,1.8).

Special methods of solution were also attempted, for example the graphical scheme of Freeman and Howland (1.13). In this they used design curves to find successive working points; by moving from chart to chart an answer would eventually be reached when the system stabilised (1.7).

Slide rules and nomograms were constructed for the appropriate power law to assist calculations.

CHAPTER 2

Recent Solution by Computer

Analogue

The invention of the transistor in 1948 has had the most significant influence on fluid network solution. At that time the response speed was limited and the importance of the device was not immediately grasped. Non-linearities were sought and investigated. Maas (2.1) used the tungsten filament lamp to approximate to the square law and his Lamp Analogue was used successfully for the Dutch State Mines. Scott (2.2) examined this idea for the National Coal Board; the limitations were those of range and power consumption. The Americans constructed the filament itself to specification for McIlroy's (2.3) Fluistor. The overall concept was to connect non-linear elements as per the network to be simulated, apply the appropriate voltages then measure the currents directly. The numbers would be scaled. The disadvantages of analogue computers were evident, that a map had to be made and a patch panel connected.

An analogue of linear resistance elements which were successively adjusted was developed by Scott, Hinsley and Hudson (2.4). The flow formula $P = RQ^2$ was linearised by $P = SQ$ where $S = RQ$ with S recalculated from $S_{m+1} = \frac{1}{2}(S_m + RQ_{m+1})$ and the process iterated to balance. This calculator required patching; it was produced by Nash and Thompson for the National Coal Board. A portable version was made in Japan by Hiramatsu (2.5). An improvement was proposed by Scott and Hudson (2.6) with their automatic analogue. In this a comparator relay indicated balance of the current for square-law flow so that measurements could be taken, otherwise a motor-driven potentiometer correction was achieved by using a negative-feedback signal. This analogue was not built but the prototype behaved successfully.

To complete this brief outline of the analogue approach, the latest computer is transistorised with circuits designed to produce a square-law, 1.8-law or other law for $H=R\dot{Q}^n$ at the setting of a rotary switch. The device was made by Network Analysers Limited (2.7) and is thought to be the largest constructed so far, simulating 410 branches. It was commissioned by the Mining Enforcement and Safety Administration, Denver, Colorado. The elements were newly designed by Williams (2.8) using integrated circuit operational amplifiers with the law of flow formed by a function generator. Suitable buffering using a silicon bi-polar transistor improved the accuracy so that errors near zero were reduced considerably. Digital voltmeters were also given improved accuracy by incorporating integrated operational amplifiers in their design.

In the overall considerations of the analysers and analogues the main disadvantage, of having patch panels, is an advantage for simulation of a network. A combined simulation and solution has very pleasing features; a network change can be designed and calculated on the same machine. The ability to feed signals to the device from the system can be used for monitoring against predicted values, thus the provision for planning well ahead of possible fault combinations is useful, as is the possibility of nearly immediate calculation, simulation and correction with an on-line machine.

Digital

When the transistor was able to be used as a fast switch it was possible to replace thermionic valves with devices that were smaller, more robust, longer lived, less power-consuming and less costly. The digital computer then had the capability of solving network and general scientific problems involving repetitive calculations and large systems. This has advanced even further with the introduction of microprocessors.

The emphasis on the method of network solution changed with more consideration being given to convenient data handling, the realisation that some form of ordering was necessary and that graph theoretic relations would assist in computer programs.

In 1961 Branin (2.9) produced a paper on the theory of linear graphs which was applicable to the electrical network problem. This was an extension of previous work and the prolific fundamental work of Kron (2.10). The elements of graph theory were reviewed and topological matrices presented together with their interrelationships illustrated by Roth's diagram and its extension. The orthogonal transformation method of Kron was introduced and the nodal, mesh and tree method solutions were discussed. The existence of inverses of the orthogonal matrices, proved by Roth, were explained and their uniqueness established. The definitions and details of these matrices are deferred a little because the electrical case has an opposite definition of $\mathbf{A}$, the current flow is different and the effects of mutual inductance do not exist in the case of fluids.

Ingels and Powers (2.11) used an IBM 650 computer to solve networks by a Hardy Cross method. They used Taylor series to expand the law of flow, arriving at the first-order approximation which Hardy Cross had reached by binomial expansion. The next term in the series would give second-order convergence and is the basis of the Newton-Raphson method. Among the networks they solved was one of Doland (2.12) used in a paper immediately after that of Hardy Cross. They concluded that the method always converged. For convenience they took a power of 2; the similar airflow networks had been computed with this power by Taylor (1.8) using a Newton-Raphson method some three years earlier.

Daniel (2.13) seems to have overlooked Branin's considerable work and describes the topology of the network with his own 'circuit matrix' $\mathbf{C}$ (which corresponds to $\tilde{\mathbf{C}}$ in this thesis). The direction of mesh flow was taken arbitrarily. He used the Hardy Cross method of solution for the non-linear networks and made enlightening comments on convergence to a solution. Considerable fluctuation was observed even when simple changes were made to the network. He thought that convergence was achieved by taking the simplified pressure-drop formulas, such as that of Hazen and Williams (1.11). However, the iteration to a solution he called the 'inner cycle'; the process was modified somewhat for compressible gas flow. The graph theory aspect is still of interest, Daniel described the formation of tree and non-tree arrays, showing that there was a one-to-one correspondence between the non-tree branches (links) and the meshes. He stated that it was convenient to take the mesh flow direction from the link direction, there being one link per mesh by definition of a tree. The consideration of the minimum number of branches connected in other meshes he termed 'overlap', '...one of the so-called criteria for convergence of the Hardy-Cross method' (his hyphen). One program was based on the Newton-Raphson method, which he observed would not be affected by large overlapping meshes but would take substantially longer to solve than by the Hardy Cross way. He further commented that the rate of convergence also depended 'strongly' on the pressure-flow characteristics of the common branches and that mesh selection is often made with pipes of nearly the same diameter in each mesh. Scott (2.14,2.15) had commented on this for mine networks, that high resistance branches should not be used as common to two meshes. In an appendix Daniel derived the condition for convergence of a Colebrook-White equation and showed that practical values will cause such convergence.

The paper of Gay and Middleton (2.16) approached the problem from a graph theoretic point of view and used orthogonal transformations to reduce one matrix size and improve inversion times. The incidence matrices used were $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. $\mathbf{A}$ is a branch by node incidence matrix where a branch is represented as directed towards a node by +1 and at its other end node by -1; the remaining nodes are not involved with this branch and so are ascribed 0. Each row of $\mathbf{A}$ therefore has one +1 and one -1. In use, however, the pressure-drops along a branch are relative to some datum node so its column is removed from the augmented $\mathbf{A}$ matrix to avoid a redundant equation and provide a consistent set. The array is sparse, at most consisting of 2b non-zero entries in a size of b(n-1), where b and n represent branches and nodes.

The $\mathbf{B}$ matrix is a branch by tree node-to-datum path array, the direction of a tree branch away from datum being taken as positive. Handling can be improved by forming $\mathbf{B}_t$, the tree branch array, then numbering the links upwards from n inclusive to b; the latter represent the closing branches of the meshes 1 to m, where m is the number of meshes and m=b-n+1. The link direction defines the sense of circulation for its particular mesh (pace Daniel (2.13)), and $\mathbf{B}_\ell = \mathbf{O}$ by definition.

The branch by mesh matrix $\mathbf{C}$ indicates the direction of a branch with respect to a mesh by +1 if in the same direction, -1 if opposite and 0 if absent from that mesh. The matrix size is b by m and its sparsity depends on the branches common to many meshes. Taking the link numbering mentioned, there is one link per mesh, that is +1 on the diagonal of the link-mesh part of the matrix

$$\mathbf{C}_\ell = \mathbf{U} \qquad (2.1)$$

a unit square matrix of size m by m.

The nodal method of solution adopted by Gay and Middleton (2.16) used the $\mathbf{A}$ and $\mathbf{C}$ matrices to transform nodal quantities to branch quantities and mesh quantities to branch quantities, respectively, when pre-multiplying the appropriate vectors. These matrices are related by

$$\tilde{\mathbf{A}}\mathbf{C} = \mathbf{O} \qquad (2.2)$$

and

$$\tilde{\mathbf{C}}\mathbf{A} = \mathbf{O} \qquad (2.3)$$

When $\mathbf{e}'$ is the vector of nodal pressures relative to the datum node then

$$\mathbf{e} = \mathbf{A}\mathbf{e}' \qquad (2.4)$$

is the vector of branch pressure rises and

$$\mathbf{i} = \mathbf{C}\,\mathbf{i}' \qquad (2.5)$$

similarly for the branch flows in terms of the mesh flows.
If the branch pressure sources are $\mathbf{E}$ then the equivalent mesh pressure sources are

$$\mathbf{E}'_{\ell} = \tilde{\mathbf{C}}\,\mathbf{E} \qquad (2.6)$$

while for the vector $\mathbf{I}$ of branch flows due to external inputs the vector of node to datum path flows is

$$\mathbf{I}' = \tilde{\mathbf{A}}\,\mathbf{I} \qquad (2.7)$$

Because of equations (2.2) and (2.3)

$$\tilde{\mathbf{A}}\mathbf{i} = \mathbf{O} \qquad (2.8)$$

and

$$\tilde{\mathbf{C}}\mathbf{e} = \mathbf{O} \qquad (2.9)$$

which are the topological versions of KCL and KVL respectively.

If the network is considered as analogous to an electrical linear steady-state network then a similar set of laws holds

$$\mathbf{V} = \mathbf{E} + \mathbf{e} \qquad (2.10)$$

where $\mathbf{V}$ is the branch pressure vector.

$$\mathbf{J} = \mathbf{I} + \mathbf{i} \qquad (2.11)$$

with $\mathbf{J}$ as the branch flow vector.

$$V = Z\,J \qquad (2.12)$$

where Z is the branch resistance matrix, the Ohm's Law type of equation.

$$J = Y\,V \qquad (2.13)$$

with Y the admittance matrix of Mho's Law, the reciprocal of the resistance matrix. Both Z and Y are diagonal matrices in fluid applications.

Substitution of (2.10) and (2.11) in (2.13) gives

$$I + i = Y(E + e) \qquad (2.14)$$

Rearrangement and pre-multiplication by $\tilde{A}$ leads to

$$\tilde{A}Ye = \tilde{A}(I - YE) + \tilde{A}i \qquad (2.15)$$

The last term is zero, from (2.8). $\tilde{A}\,I$ is I' from (2.7) and $e = Ae'$ from (2.4) hence

$$\tilde{A}YAe' = I' - \tilde{A}YE \qquad (2.16)$$

and inverting to obtain e', the vector of nodal pressures

$$e' = (\tilde{A}YA)^{-1}(I' - \tilde{A}YE) \qquad (2.17)$$

The branch flows can be obtained from the vector J as

$$J = YV = Y(E + Ae') \qquad (2.18)$$

Using the C matrix together with $\tilde{C}e = O$ a mesh method of solution can be reached similarly

$$i' = (\tilde{C}ZC)^{-1}\tilde{C}(E - ZI) \qquad (2.19)$$

A linearisation method propounded by Bending and Hutchison (2.17) was claimed to be simpler in conception, more general in application and often shorter in computation time than the Hardy Cross and diakoptics methods used by Gay and Middleton (2.16). Their method was to derive a set of equations of the network which consisted of mass balances for each node, pressure drop equations for each branch, specified pressure drop equations for each pump, input and output flow rates and enough nodal specifications to define the problem. The laminar and turbulent regimes were separately defined with the latter giving rise to non-linearities which were linearised using an initial guess of velocity. The method converged very slowly with pipe velocity oscillation about the correct values, which could be improved by taking the mean of consecutive velocity results. The large number of sparse simultaneous equations had to be solved for each iteration, a two-stage Gaussian elimination and operator list were employed. The contrast with the Hardy Cross and diakoptics approach is that while the topology and enough extra specifications were needed for input there was no need for mesh specifications nor two sets of matrices to define decomposition. They concluded that linearisation was a satisfactory general method for steady-state solution of a pipe network with a single incompressible fluid.

The solution approach of Mah (2.18) was to restructure the problem by decomposition techniques and use efficient data handling so that both the density of the mathematical representation and the number of operations would be reduced. Mah drew attention to the existence in his matrix of two types of equation, Type 1 being linear (KCL) and Type 2 non-linear (KVL). The coefficients of these equations were given by the incidence matrices $\mathbf{M}'$ and $\mathbf{C}$ ($\tilde{\mathbf{A}}$ and $\mathbf{C}$ in this thesis). The partial ordering was accomplished by using an algorithm which reassigned nodes and branches so that no node was incident with a branch which had already been assigned to a previous node. The structure achieved was upper triangular and fewer operations were required per iteration. The solution could then be performed using the product form inverse, which was further extended to the non-linear equations by taking a row-oriented product form. A second solution method improved on this by eliminating the elements below the diagonal for the first (n-1) rows and performing Gaussian elimination on the remaining submatrix in the right hand corner section of the matrix.

The number of calculations for the mesh equations were reduced by using an algorithm to find a near-minimal set of branches for the meshes to be solved; the multiplications per iteration thereby being reduced. Mah claimed this as a new direction in the field of computation with the use of graph theoretic techniques directly enhancing the efficiency of computation.

A simpler version of the orthogonal mesh method was proposed by Gay and Preece (2.19) where the transformed network can be regarded as consisting of 'open' and 'closed' paths, the tree node-to-datum and mesh paths. The vectors associated with this network are primed and partitioned into tree and link submatrices, thus

$$J' = \begin{bmatrix} I' \\ \hdashline i \end{bmatrix} \qquad (2.20)$$

$$V' = \begin{bmatrix} V'_t \\ \hdashline V'_\ell \end{bmatrix} = \begin{bmatrix} E'_t + e'_t \\ \hdashline E'_\ell + e'_\ell \end{bmatrix} \qquad (2.21)$$

where I' is the vector of nodal flows and constitutes the flows in the node-to-datum paths. Because the sum of the pressure rises and sources round a mesh is zero (from KVL)

$$V' = \begin{bmatrix} E'_t + e'_t \\ \hdashline E'_\ell \end{bmatrix} \qquad (2.22)$$

The flows in the original network may be related to the flows in the orthogonal network by a transformation operator

$$J = \gamma J' \qquad (2.23)$$

The two sets of paths can then be related by

$$\gamma = [B \,\vdots\, C] \qquad (2.24)$$

The pressure vectors are similarly related

$$V' = \tilde{\gamma} V \qquad (2.25)$$

so

$$V' = \tilde{\gamma}(Z\,J) = \tilde{\gamma} Z \gamma J' \qquad (2.26)$$

Using the previous equations (2.1), (2.9), (2.20), (2.21), (2.24) together with $C_\ell = U$

$$i' = (\tilde{C}_t Z_t C_t + Z_\ell)^{-1}(E'_\ell - \tilde{C}_t Z_t B_t I') \qquad (2.27)$$

and

$$E'_\ell = \tilde{C}_t E_t + E_\ell \qquad (2.28)$$

The flows can be calculated using (2.23), (2.12) and (2.10).

The improvement of this method over the usual solution by inversion is that the matrix to be inverted is of size m by m instead of $n-1$ by $n-1$ also links of zero resistance can be included if required. The advantage over the usual transformation method is that smaller topological matrices are used, together with a simpler notation due to ordering.

CHAPTER 3

Computational Aspects

Recent Computer solution

In the case of analogue solutions, which may be dismissed here quite shortly, the major improvement has been made with the design of the function generator to represent the required law. The disadvantage of patch panels has not yet been avoided, automatic patching has been suggested. The nearest approach seems to be that used by Membrain (3.1) with the construction of a digital computer simulating an analogue, which is set by keyboard.

Improvements for the digital solution were made in the handling of data on an orderly scale by Kron (2.10) with the introduction of his tensors, by Branin (2.9) with the topological matrices and use of digital computers. The size of the network caused difficulty when large, the solution then being by Kron's diakoptical method. The initial way of solving appeared to be by setting up the Ohm's Law type of equation and inverting the coefficient matrix or by using some form of elimination. This type of inversion would be sized for the number of branches (assuming the primary concern is with branch flows) and the inversion time would be proportional to $(n-1)^3$. This matrix was usually symmetrical so the inversion time could be reduced to $(n-1)^3/2$. Data input to a computer would usually include all zeroes, hence b^2 items.

The topological matrices used to describe the network reduced the drudgery of setting up some of the equations and were able to disentangle the meshes, thus a column of the **C** array gives all the branches in a mesh and from the Happ (5.1) relationship

$$C_t = -B_t \tilde{A}_\ell \qquad (3.1)$$

C can be generated from the data input of the other arrays. These matrices have the same advantage as the inverted coefficient matrix that once found they can be used repeatedly for that network. As indicated elsewhere, the **A** matrix is sparse and the others may have large sections of zeroes. Handling of these can be improved by using sparsity techniques (3.2). Two cases are immediate, that a resistance matrix is diagonal for a fluid network, hence its trace can be entered as data and the product made without any zeroes. The saving on data input is then $b^2 - b$, on multiplication $b^2 - b$ and on matrix addition $(b-1)^2$. The input of **A** can be made by defining the branches with their end nodes, as in Preece's (6.1) node connection panel. As the unaugmented **A** array has b(n-1) entries this saves b(n-3) zeroes being entered. Quite often two of the three topological matrices have to be entered, it is convenient to enter one and calculate the other and hence reduce data input error and validation time. If the topological matrices can be ordered in some way there can be computational savings of space and time.

The ordering of links by Preece (6.1) had not only reduced the $\mathbf{C}_\ell$ array to diagonal but had made it a unit matrix, thereby eliminating a (b-n+1) by m matrix multiplication. The adoption of the **B** matrix meant that $\mathbf{B}_\ell$ was a null matrix.

Mah (4.2) applied ordering with sparse techniques to the Gay and Middleton (6.6) test network. His occurrence matrix was composed of the partitioned matrices

$$\left[\begin{array}{c:c} \tilde{A}_t & \tilde{A}_\ell \\ \hdashline \tilde{C}_t & \tilde{C}_\ell \end{array}\right]$$

in the notation of this thesis. It was improved by renumbering the branches using his Algorithm PO to obtain an upper triangular matrix for

A_t which was subsequently inverted more economically. The mesh matrix was improved by using Algorithm MLC to find near-minimal length meshes and reduce the density of the occurrence matrix. The operations required for R meshes he quotes as of the order $12R^3$, but having obtained the improved structure it can be used repeatedly with savings in operations each time. With a network directed according to some sign convention the inaccessible node of PO is made datum and combined into Mah's Algorithm PODA.

Inversion and inversion methods have been regarded as the most fruitful way of reducing computation time (which is also related to storage space and cost) because of the wide applications in the scientific field. Gay and Preece (6.8) quoted relative inversion times for Gauss-Jordan, ICL package, Choleski and Caffrey methods and mentioned Branin's LAT (link at a time) solution. The importance of economical working is significant where many solutions are required for a problem or where the network is large. For small one-off problems the data validation print-out and output can take most of the time.

The Bending and Hutchison (2.17) linearisation method used Gaussian elimination for the Gay and Middleton (2.16) test example and ran in Fortran on the Cambridge University Titan (prototype Atlas II), using 12K 48-bit words and taking 2.1 seconds for 6 iterations to finish. Time comparisons are machine- as well as method-dependent; it is difficult to define and compare computer powers.

Mah (2.18) used the Product Form of the Inverse, which expresses the inverse as a product of elementary matrices and corresponds with the

Gauss-Jordan elimination method. With his occurrence matrix including non-linear rows it was possible to modify the PFI to perform for row rather than column orientation. An example of the reduction was quoted for b=200, m=10 with nodes of average degree 3. Gaussian elimination by a complete matrix multiplication required 2.7×10^6 products, PFI 3.1×10^4. A further reduction to 6.1×10^2 multiplications was possible by partitioning and lower triangularisation with back substitution. The computer used was not stated nor was the time. Gay and Middleton (2.16) using Daniel's (2.13) method needed 15 iterations and took 30 minutes on an Elliott 803 computer for the maximum overlap case.

The method of solution affects computation time and storage, fluid networks being non-linear attract the Hardy Cross method, or the Newton-Raphson because of its second order convergence. The emphasis on these methods is linked with the number of iterations on the inner cycle, if these can be reduced a saving of time is achieved. The difficulty has often been to find some usually empirical method that will accelerate convergence. Various factors have been proposed but they seem to be characterised by the type of network. Oscillations in the solution cause an increase in the number of iterations and are not easy to avoid. They may require another form of mesh selection. The diakoptical methods, e.g. Mullineux and Reed (3.3), are usually applied to large networks; node-to-datum analysis can be performed and multiple couplings of effects between branches can be incorporated if required.

Rather than calculate with either the mesh method or the nodal method a combination of both can be used as in the mixed-mode method of Branin (5.2) and the hybrid method of Hamam and Brameller (3.4).

The orthogonal method, used by Kron (2.10) with his tensors has become popular as a similarity transformation with matrix methods. The matrices A and C are rectangular and cannot be inverted but are square when used as $\tilde{A}YA$ or $\tilde{C}ZC$. Similarly with the combined transformation operator $\gamma = \left[B \vdots C\right]$ which is b by (m+n-1) when used as in equation (2.26)

$$V' = (\tilde{\gamma} Z \gamma) J' \qquad (3.2)$$

essentially sets the system as being transformed to

$$V' = Z' J' \qquad (3.3)$$

cf. Kron (2.10).

The Present Work

A description of the computers and languages used for this thesis is apposite. Initial exploration was made on an ICL 1903A 96K words machine by hands-off bureau working. The language used here was Fortran IV. Some approximate times of inversions are covered in the Results, Chapter 9. The other two machines were Hewlett-Packard 2000E and 2000, both used separately in interactive mode. The Wolverhampton version had two fixed and two interchangeable discs with a capacity of 72K words of 16 bits. The speed of operation was not quoted, being overshadowed by the teletype printing speed in any case. The interactive approach was particularly appreciated because of the immediacy of error correction. It was also a spur for data reduction and more problem solving. Several thousand minutes were logged. The two systems were not compatible, having some differing commands, four different teletypes, two different visual display units for the purposes of this research. Acoustic couplers were used on occasion. Paper tapes were not compatible.

The language involved here was Basic, Hewlett-Packard E level was used mainly. The programs were written after flowcharts had been constructed for the algorithms and were tested, rewritten or modified heuristically. Continuous looping was detected by inserting a print order at a flowchart junction with a code letter. Such diagnostic programs were extremely useful for finding repetitious node sequences made by an algorithm traversing a loop in the network when it should have stepped out. In general the program listings in this thesis are not diagnostic but diagnostic statements occasionally appear.

The program size is restricted in Basic and depends on the aggregate size of the matrices used and what might happen to them; this affects the length of a program. The facility of a CHAIN command was used so that a suite of programs could be linked together and worked through sequentially. The variables and arrays common to the programs are stated in the lowest numbered line of the program and repeated in that order for the subsequent programs. After one program has been completed the common material is taken on to the next chained program and the first is overwritten. It is possible to put intermediate answers into the common arrays and carry them forward that way. Temporary arrays are entered under a DIMENSION statement, they are automatically destroyed when the next program is chained. This type of chain is distinct from the chain of graph theory, some care has been taken to keep the meanings separate in the text.

The listings were produced by calling the program from the User Library and punching on tape, thus getting the actual program used. In the absence of a Hewlett-Packard line printer and with the difficulties of sprocket holes and left hand margin space the tape was run on a Friden Flexowriter with the margin adjusted, the line width adjusted manually and the print made on a good quality teletype roll.

CHAPTER 4

A New Ordering Approach; Hamiltonian Path Considerations

Attempts to improve the calculations by ordering have produced tri-diagonal matrices and compressed $\mathbf{A}_\ell$, for example. Such compression can forfeit the advantage of handling a sparse matrix. The approach of Mah (4.2) in his algorithm PODA while reorganising the tree and renumbering the branches causes the creation of nodes on limbs, the pendant nodes. While nodes on the ends of tree branches are not necessarily prohibited they can cause meshes that seem larger than are ideally required. For such a node the branch joining it must be a link by definition, the rest being tree branches which then contribute to the overlap problem by giving to the calculation extra branches in each mesh.

Applying this consideration of ordering to the graph matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ it was realised that certain improvements might be possible. $\mathbf{A}_\ell$ describes where the links are in the network and as these links can be anywhere reorganisation would appear to be awkward. $\mathbf{C}_t$ corresponds with $\mathbf{A}_\ell$ and the mesh numberings are improved by the arrangement for $\mathbf{C}_\ell$ to be a unit matrix. $\mathbf{B}_\ell$ is a zero matrix as the links cannot be in the tree path by definition. This leaves $\mathbf{B}_t$, and it is observed that the datum can be taken arbitrarily and the nodes numbered likewise. From this consideration it seems reasonable that if there is just one tree branch from the datum node to the 'next' node then this branch will appear in all node-to-datum paths. A line of 1's in the $\mathbf{B}_t$ matrix then occurs. Similarly with the second node, if there is one tree branch to its 'next' node then all the remaining node-to-datum paths must go through it and its $\mathbf{B}_t$ row will contain one 0 (for the previous branch) and then all 1's. This argument can be repeated similarly for all the remaining nodes up to the last for (n-1) paths.

As the nodes can be numbered in any fashion it would be more general to start at 1 and finish at n, rather than starting at n; as the datum node has to be excluded it can be numbered 0. The $\mathbf{B}_t$ matrix thus generated is:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots\cdots\cdots & 1 \\ & 1 & 1 & 1 & \cdots\cdots\cdots & 1 \\ & & 1 & 1 & \cdots\cdots\cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots\cdots\cdots & \cdots \\ & & & & & 1 \end{bmatrix}$$

which is not strictly upper triangular but could be called 'filled upper triangular' so that the leading diagonal is included with 1 everywhere else understood. A shorthand notation is suggested to save copious writing. A symbol from the Attic Greek alphabet $\digamma$ (digamma, now unused) is sufficiently distinctive and allusive.

An immediate result is that the $\mathbf{B}_t$ matrix is made implicit

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_t \\ \cdots \\ \mathbf{B}_\ell \end{bmatrix} = \begin{bmatrix} \digamma \\ \cdots \\ 0 \end{bmatrix} \qquad (4.4)$$

Mathematical and computational advantages accrue.

The $\mathbf{B}$ matrix now having been structured gives rise to three further questions, what does it mean geometrically, is it obtainable and is it of practical significance? The answer to the first is that from this definition the tree must go from node to node continuously, i.e. no node is reached twice on such a path. This type of traverse was proposed by Sir W.R. Hamilton, where, if the journey returns to the starting node a Hamiltonian circuit or tour is said to exist. What has been made here in the $\mathbf{B}_t$ matrix is not a circuit but will be referred to as a Hamiltonian path. Clearly, if a Hamiltonian circuit exists there will be at least one Hamiltonian path in it.

Hamiltonian Considerations

Conditions for the existence of a Hamiltonian circuit or path have been the cause of concern since their discovery. At present no necessary and sufficient condition has been found. Various theorems and corollaries of different strengths have been proposed; some of them have been investigated in this context of practical distribution networks.

The many theoretical investigators of graphs included Whitney (4.3), who looked at the aspects of chains and separability. A graph is separable if it can be divided into two or more sub-graphs by the removal of a single node, the cut-node. Whitney produced theorems which included a necessary and sufficient condition for a graph to be non-separable (that it should contain no cut-node).

The investigations of Dirac (4.4) involved the degrees of the nodes, that is the number of their incident branches. In his notation, if the degree of a node is ρ then a sufficient condition that the graph is Hamiltonian is that for each node v of all the n nodes of the graph, $\rho(v) \geq n/2$. For computing purposes this implies that the degree of each node is needed as extra information, which might be obtained by summing the columns of **A** . The condition was strengthened by Pósa (4.5) who considered the Hamiltonian path and stated that for a graph with more than two nodes the degree of each should be at least (n-1)/2.

Berge (4.6) in his classic text proved that in a graph where every pair of nodes is joined in at least one direction, a 'complete' graph, then it contains a Hamiltonian path. He related this to Dirac's theorem. Some interest in matching nodes with their connected nodes led to the Hungarian Method (4.6); it was proved that by taking half the out-degrees of successive nodes then a Hamiltonian path formed a decreasing sequence.

A study of Hamiltonian paths in tree graphs was made by Cummins (4.7), where the nodes themselves represent the trees of a digraph. Kamae (4.8) investigated the generation of all such trees and gave an algorithm for the generation of a Hamiltonian path. Another approach was that of Kishi and Kajitani (4.9) who decomposed tree graphs into subgraphs then constructed Hamiltonian circuits using a 14-step algorithm. Further work on tree graphs was performed by Chen (4.10) who also published more information in that reference. In the utilisation of these works for this thesis it was considered that as only one Hamiltonian path need be found for Λ the importance of tree graph relationships must take a lower priority.

Wang and Kleitman (4.11) considered the degree sequence of the nodes in the graph and gave a proof which under certain conditions was necessary and sufficient for the existence of the graph. One of the iff conditions was that the degrees of all n nodes should be at least n, which is not realistic for practical networks and so their constructing algorithm has not been used here.

The so-called Kozyrev-Grinberg necessary condition for a Hamiltonian circuit is referred to by Gehner (4.12). It is

$$\sum_{i=2}^{n}(i-2)(f_i - f_i') = 0 \qquad (4.5)$$

where the graph has n nodes, f_i is the number of interior regions, f_i' the number of exterior regions bounded by i branches, as determined by some circuit. While of relevant interest it was not worth while constructing an algorithm because a Hamiltonian path was considered as fundamental.

Some more recent results have been published in a book by Andrásfai (4.13). It defines a graph as 'strongly directed' if the branches are orientated so that each node is accessible from any other along directed paths, often referred to as the traffic condition. Then for a simply connected, strongly directed graph of n nodes the sum of the in-degree and out-degree of each node must be at least n.

It is perhaps here that there is a clue to the existence of a Hamiltonian path in practical distribution networks, the traffic condition is designed whereas graph examples are constructed as graphs. Other factors such as economy also influence the practical design.

A recent address by Lesniak-Foster (4.14) summarised the sufficient conditions for Hamiltonian graphs and included various properties. He concludes that the best possible sufficient condition is that of Chvatal. Wide interest in the subject is leading to the development of a new field called 'hamiltonian theory'.

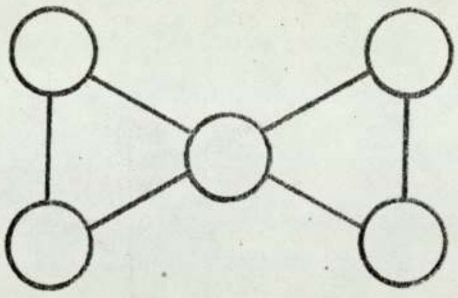
Reverting to the practical problems, an approach to the difficulty can be made by looking at the exclusivity aspect. Mullineux and Reed (6.2) postulated and proved that a graph that can be separated by the removal of two nodes and their incident branches to leave four or more sub-graphs cannot contain a Hamiltonian path. This should assist here by distinguishing such graphs. A graph which does not have four or more subgraphs does not necessarily have a Hamiltonian path; this problem can be tackled in another way. In a paper by Roberts and Flores (6.3) is proposed an **M** array of node connections which could be searched for a Hamiltonian path. While it would appear that this is a brute force and combinatorial way it must be borne in mind that for the conjectured method of solution just one Hamiltonian path is required and that this can reduce the search.

For the system to be of use in the practical network context it is thought that the differences between graph theoretic and fluid networks ought to be regarded. If the Hamiltonian path existence cannot be found for a theoretical network then there may be difficulties in the practical case. Definitions of the practical case should be regarded more closely:
i) a node being the junction of two or more branches means that there is not a node at the end of an isolated branch, in other words the graph is 'connected' in the graph theoretic sense,
ii) parallel branches can be calculated using some reciprocal law such as $1/R^s = \sum 1/r^s$; the parallel branches can be lumped for the purposes of a global solution; the graph theoretic definition of this is that the graph is 'simple'.

In this context it is thought that the practical graphs have a feature not necessary for theoretical graphs. It stems from consideration of the practical networks being distributive; there is a supply and demand at each node, an intake and return, a connected positive and negative. Thus the graph theoretic example of Deo (4.15) of Fig. 4.1 has a Hamiltonian path and no Hamiltonian circuit but is thought to be impractical because of being graphically separable at its cut-node.

In the case of digraphs the **M** array can be completed with the nodes

Fig. 4.1

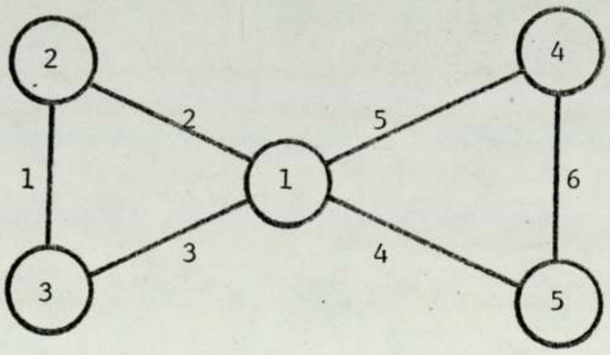


in the array as those which are reached in a positive direction. The search can then be made and if a Hamiltonian path exists it will be found. Because the array is smaller than that of the undirected graph the search would usually be quicker. A combination of the Mullineux-Reed criterion and the **M** algorithm provides the closest answer to the practical Hamiltonian path existence problem at present.

The **M** array and its use - Examples.

In the case of Fig. 4.1 an **M** array description might be made by numbering the branches and the nodes as shown in Fig. 4.2. Although such a numbering can be arbitrary, this case has been chosen to bring out several aspects of the technique.

Fig. 4.2



M =	1	2	3	4	5	(node)
(node	2	1	1	1	1	
connection	3	3	2	5	4	
	4					
column)	5					

Starting at node 1 (simply to order the approach) and looking for a

Hamiltonian path, the first route tried would be

1-2-3 (stops at end of column 3), the next route to be tried

1-3-2 (stops at the end of column 2), the next to be tried

1-4-5 then

1-5-4 then the next route starts at node 2 (for simplicity)

2-1-3 then 2-1-4-5, 2-1-5-4

2-3-1-4-5 First Hamiltonian path and second 2-3-1-5-4

3-1-2 Starts at 3 (for simplicity)

3-2-1-4-5 Third Hamiltonian path and fourth 3-2-1-5-4

4-1-2-3 (note: has gone into a closed loop 1-2-3), 4-1-3, 4-1-5

4-5-1-2-3 Fifth Hamiltonian path and sixth, 4-5-1-3-2

5-1-2-3 (closed loop, node 4 unreachable), 5-1-3-2, 5-1-4

5-4-1-2-3 Seventh Hamiltonian path and eighth, 5-4-1-3-2

All routes have been tested, all Hamiltonian paths found. A Hamiltonian circuit consideration would require that the starting and finishing node must be the same - numerically seen here as not arising.

For a directed network there is a reduced $\mathbf{M}$ array. Suppose, for brevity, that Fig. 4.2 has branches directed from low node number to high node number, then

$$\mathbf{M} = \begin{array}{|ccccc} 1 & 2 & 3 & 4 & 5 \\ \hline 2 & 3 & . & 5 & . \\ 3 & & & & \\ 4 & & & & \\ 5 & & & & \end{array}$$

is half the number of former entries. There is no Hamiltonian path; also this can be deduced because nodes 3 and 5 must both be finishing nodes only, but one alone is needed. The particular example is a variation of a wheel (cf. WHE) with node 1 as a source hub and two pieces of the rim missing (2-4, 3-5), the other pieces of rim are directed oppositely.

CHAPTER 5

Some Mathematical Properties of Λ

The matrix Λ as previously defined can also be regarded as a straight mathematical entity and as an operator. In the case of pre-multiplication by Λ :

$$\Lambda = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ & 1 & 1 & \dots & 1 \\ & & 1 & \dots & 1 \\ & & & & . \\ & & & & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \text{-----------} & a_{1n} \\ a_{21} & & & a_{2n} \\ . & & & \\ . & & & . \\ a_{n1} & \text{-----------} & & a_{nn} \end{bmatrix} \tag{5.1}$$

$$= \begin{bmatrix} \sum\limits_{r=1}^{n} a_{r1} & \sum\limits_{1}^{n} a_{r2} & \text{-----------} & \sum\limits_{1}^{n} a_{rn} \\ \sum\limits_{r=2}^{n} a_{r1} & \sum\limits_{2}^{n} a_{r2} & \text{-----------} & \sum\limits_{2}^{n} a_{rn} \\ \text{-----} & \text{-----} & \text{-----} & \text{-----} \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix} \tag{5.2}$$

This property reduces a pre-multiplication to simple addition, each solution element being the upwards sum of the elements below it with itself e.g.

$$\Lambda \begin{bmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 15 \\ 10 \\ 6 \\ 3 \\ 1 \end{bmatrix} \tag{5.3}$$

and similarly for any further columns.

The transpose of Λ follows normally and its pre-multiplication, $\tilde{\Lambda}\underset{\sim}{a}$, is the downwards sum of the elements e.g.

$$\tilde{\Lambda} \cdot \begin{bmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 9 \\ 12 \\ 14 \\ 15 \end{bmatrix} \tag{5.4}$$

The inverse of Λ is sometimes needed in manipulation; it is easily obtained by the so-called Exchange Method which utilises row operations to alter the matrix positions in

$$[\Lambda \vdots U] \sim [U \vdots \Lambda^{-1}] \tag{5.5}$$

$$\Lambda = \begin{bmatrix} 1 & 1 & 1 & & 1 \\ & 1 & 1 & & 1 \\ & & 1 & & 1 \\ & & & . & 1 \\ & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & . & \\ & & & & 1 \end{bmatrix} \tag{5.6}$$

$$= \begin{bmatrix} 1 & 1 & 1 & & 1 \\ & 1 & 1 & & 1 \\ & \cdots & \cdots & & \\ & & & 1 & 0 \\ & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \cdots & \cdots & \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix} \tag{5.7}$$

$$= \begin{bmatrix} 1 & 1 & 1 & & & 1 \\ & 1 & 1 & & & 1 \\ & \cdots & \cdots & \cdots & & \\ & & & 1 & 0 & 0 \\ & & & & 1 & 0 \\ & & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \cdots & \cdots & & \\ & & & 1 & -1 & 0 \\ & & & & 1 & -1 \\ & & & & & 1 \end{bmatrix} \tag{5.8}$$

The process continues upwards from the bottom row, eventually giving:

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \cdot & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & \cdot & & \\ & & & & 1 & -1 \\ & & & & & 1 \end{bmatrix} \qquad (5.9)$$

Taking the Happ (5.1) relationship

$$\tilde{B}_t = A_t^{-1} \qquad (5.10)$$

and using the proposed definition

$$B_t = \lambda \qquad (5.11)$$

then

$$\tilde{\lambda} = A_t^{-1} \qquad (5.12)$$

Inverting this equation gives

$$A_t = \tilde{\lambda}^{-1} \qquad (5.13)$$

which is the transpose of the inverse found above.

Having defined a particular B_t it is to be expected that A_t is also a particular matrix and this is now known.

Another Happ equation

$$C_t = -B_t \tilde{A}_\ell \qquad (5.14)$$

becomes

$$C_t = -\lambda \tilde{A}_\ell \qquad (5.15)$$

with the multiplication process simplified to cumulative addition.

Similarly the Branin (5.2) equation

$$D = A \tilde{B}_t \qquad (5.16)$$

becomes

$$D = A \tilde{\lambda} \qquad (5.17)$$

and it is realised that λ permutes.

CHAPTER 6

The Application of Λ in the Network Problem

Solution by means of Equation 14 of Gay and Preece (6.8) can be simplified by taking the foregoing considerations and ordering, together with the use of the mathematical properties of Λ. $B_t I'$ becomes $\Lambda I'$,

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ & 1 & 1 & \dots & 1 \\ & & 1 & \dots & 1 \\ & & \dots & \dots & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} I'_{n-1} \\ I'_{n-2} \\ I'_{n-3} \\ \cdot\cdot \\ I'_1 \end{bmatrix} = \begin{bmatrix} \sum_1^{n-1} I'_r \\ \sum_1^{n-2} I'_r \\ \sum_1^{n-3} I'_r \\ \dots \\ I'_1 \end{bmatrix} \tag{6.1}$$

This has a significance that the flow through the datum node (0) is the sum of all the node-to-datum path flows. As B_t is now defined by Λ there is a saving of $(n-1)^2$ places of store, B_t not usually being sparse. Λ in its operational mode can be expressed in program form by an order. The multiplication involving $(n-1)^2$ operations is completely obviated and the summation reduced from $(n-2)^2$ to $(n-2)(n-1)/2$ additions, a saving of $(n^2-5n+6)/2$ operations. A further saving is achieved in the transformation matrix γ,

$$\gamma = \left[B \,\vdots\, C \right] = \left[\begin{array}{c:c} \Lambda & C_t \\ \hdashline O & U \end{array} \right] \tag{6.2}$$

as $B_\ell = O$ only C_t has to be stored. The transformation itself

$$\gamma J' = \gamma \begin{bmatrix} I' \\ i' \end{bmatrix} \tag{6.3}$$

or

$$\left[\begin{array}{c:c} \Lambda & C_t \\ \hdashline O & U \end{array} \right] \begin{bmatrix} I' \\ \cdots \\ i' \end{bmatrix} = \left[\begin{array}{c} \Lambda I' + C_t i' \\ \hdashline i' \end{array} \right] = J \tag{6.4}$$

has the same number of operations, $B_t I'$ having been previously computed in each method.

There are also topological advantages achieved with this convention and method of numbering; the tree branches are numbered and run sequentially from 1 to n-1; the links, if directed from low-number node to high-number node, are all positive and the meshes are sequentially defined by them. The links are described by $C_\ell = U$, (Preece (6.1), also Daniel (6.5)), in the mesh method.

The dual of the mesh method is the nodal method and $\wedge$ is of use here as expected from Chapter 5.

$$\tilde{A}Y \quad \text{becomes} \quad \left[\tilde{A}_t \vdots \tilde{A}_\ell\right]\begin{bmatrix} Y_t \\ \cdots \\ Y_\ell \end{bmatrix} = \left[\tilde{B}_t^{-1} \vdots \tilde{A}_\ell\right]\begin{bmatrix} Y_t \\ \cdots \\ Y_\ell \end{bmatrix} \qquad (6.5)$$

$$= \left[\wedge^{-1} Y_t + \tilde{A}_\ell Y_\ell\right] \qquad (6.6)$$

The computational saving of the $\wedge^{-1} Y_t$ product is made from

$$\begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & 1 & -1 & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} Y_{11} & & & \\ & Y_{22} & & \\ & & Y_{33} & \\ \cdots & \cdots & \cdots & \cdots \\ & & & Y_{n-1,n-1} \end{bmatrix} \qquad (6.7)$$

b-1 differences compared with b(n-1) products and (b-1)(n-1) additions. No further saving for $\tilde{A}YE$ occurs as $\tilde{A}Y$ would usually be carried forward.

The $\wedge^{-1}$ matrix in equation (6.7) corresponds with one of Mullineux and Reed (6.2), being the trunk portion of their $\tilde{A}_a^*$ incidence matrix when A_a^* is unaugmented. Their considerations were with sub-graphs, trunks being an alternative name for Hamiltonian paths.

Algorithms and Overlap Considerations

Apart from the straightforward space and time savings mentioned, there is another aspect of effective saving. The assignment and control of data can be organised so that the minimum amount of data is handled. Instead of entering the $\mathbf{A}$ matrix or the $\mathbf{M}$ matrix, for example, a sparse version of $\mathbf{A}$ with the 0's suppressed can be used. A graph can be considered as made up of nodes and so the branches could be defined as node connections. These need not be numbered because a nodal dyad would be unique for specifying a branch (it had been stipulated that the network was simple and connected). A Hamiltonian path could be expressed as a node string, however in practice it is more usual for both nodes and branches to be numbered. The branches may be taken sequentially and entered as a nodal dyad, thus describing the network in terms of nodes, branch-ordered. From this the $\mathbf{M}$ matrix can be constructed by algorithm ATOM (q.v.). The saving for this method is b for the branch numbers, being now implicit, instead of the $\mathbf{A}$ matrix $b(n-1)$ entries (or $(n-1)(b-n+1)$ if $\mathbf{A}_\ell$ only is needed), $b(n-3)$ numbers. This reduces the chance of data input error, validation time and space.

After the $\mathbf{M}$ matrix has been constructed and an output for validation has been made the next algorithm can be called. With interactive computing this may be made by means of a COMMON statement in BASIC and suppression of now unwanted data. Such an algorithm might be MAL (q.v.) where the matrix is searched to find a Hamiltonian path; a slight modification can be made for the algorithm to find all such paths, but only one is needed for the purposes of calculation. As Roberts and Flores (6.3) remark, '...The computer program is fairly easy to describe and fairly difficult to implement...' It is sometimes helpful to renumber a network to make it more tractable, especially as considerable time could be spent in sorting through nodes which do not have an end node to complete the path. For n nodes Roberts and Flores claim that the time taken by their program is 'essentially independent' while Christofides (6.4) says it varies exponentially, especially when $n > 20$. Several improvements in the procedures have been suggested by Christofides and others. However, the programs used were not for similar purposes, e.g. the minimum cost for the Travelling Salesman problem is not required here, the conjecture is that any Hamiltonian path will suffice.

Apropos of this, solution speed has been related to the way the tree of a network has been taken. Hardy Cross convergence appeared to be partly dependent on the amount of overlap of branches used in each mesh calculation. The observation of Daniel (6.5) is apposite...'the method of finding the minimum overlap suffered from the usual disadvantages, i.e. what is obvious to the eye is not obvious to the computer'. Consider his network and numbering in Fig. 6.1

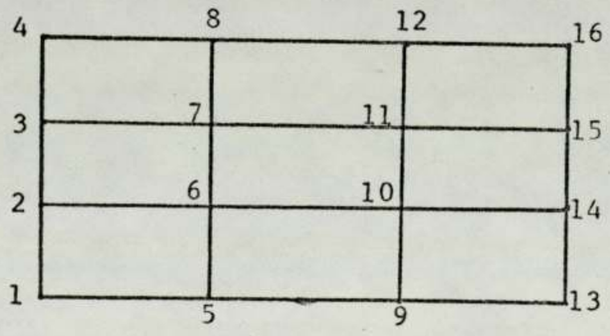


Fig. 6.1

Application of the M algorithm would give the tree structure of Fig. 6.2.

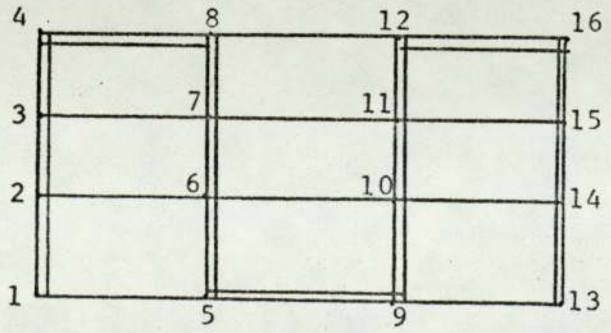


Fig. 6.2

This solution while not minimum overlap effectively renumbers branches and meshes, automatically identifying the branches in each mesh -- a great convenience for computer work.

Middleton (6.6) found that maximum overlap hindered Hardy Cross convergence although the cutting pattern for his diakoptical solution reduced the running time. Preece (6.1) found that minimum overlap gave swift convergence and that the trunk was the easiest to set up for maximum overlap. It would appear that the pattern and selection affect overlap, for whereas in Fig. 6.2 a total of 81 branches would be used in calculation in Fig. 6.3 only 55 are involved. Without a Hamiltonian path the minimum overlap would reduce the number of branches for the calculation to 37. With a Hamiltonian path the overlap is essentially arbitrary and only depends on the selection of the starting point.

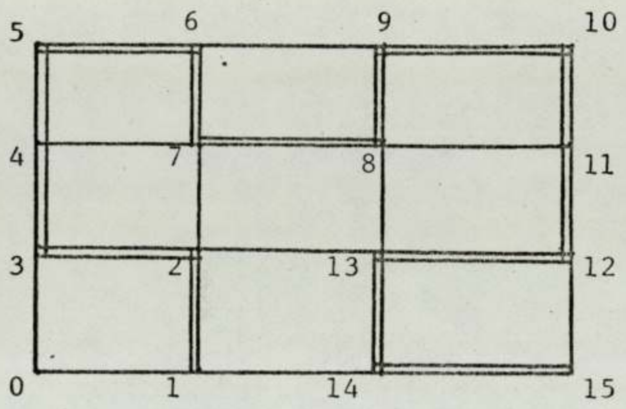


Fig 6.3

In the test example of Gay and Middleton (6.6) their overlap for Case 1 was with 70 (total) branches used, Case 2 105 and Case 3 118. A Hamiltonian path taken through the network with their numbering involves 103 branches for mesh calculations.

Instead of a direct solution by orthogonal means the indirect construction of the $\tilde{C}_t Z_t C_t + Z_\ell$ matrix, after Percival (6.7) for example, further utilises the Λ ordering. The tree branches being link-defined are signed negatively, hence the overall product is positive. The fluid resistance matrix is then composed:

i) On the diagonal -- the sum of the branch resistances round each mesh respectively.

ii) Off the diagonal -- the sum of the branch resistances common to meshes i and j.

Hence the input of C is avoided and only the sparse A_ℓ is needed for the input.

An algorithm such as HAMIL (q.v.) can be used to perform the calculation, where the chained programs can be:

$$\begin{tikzcd}
{} \arrow[r] & \text{ATOM} \arrow[r] & \text{MAL} \arrow[r] & \text{BREN} \arrow[r] & \text{HAMIL} \arrow[r] & \text{DISP}
\end{tikzcd}$$

The DISP algorithm (q.v.) displays the results, being too long to be incorporated with HAMIL, while BREN (q.v.) renumbers the branches.

The foregoing remarks apply to the solution of the linear case, as in Gay and Preece (6.8) and which is solved by the ρ method in the illustrative example of the Appendix EXAMPLES with numerical results.

The Hamiltonian tree from the M algorithm has an arbitrary degree of overlap since only the first tree found is used and the procedure is in no way concerned with the overlap property. Therefore if this algorithm is to be used for a Hardy Cross solution it may be modified, however Preece stated that the degree of overlap of the tree is not of significance when based on matrix iterative methods, so a tree found by this method has advantages over the Hardy Cross method.

CHAPTER 7

The Non-Linear Flow Problem

The problem of flow in pipes has an additional difficulty in that the fluid resistance depends upon the Reynolds Number and the relative roughness of each pipe. There are many empirical formulas covering various conditions but that of Colebrook and White is considered to be of greater accuracy. Where $\phi = \phi(Re, \epsilon/D)$, with ϵ the absolute roughness and D the diameter, the rough pipe version is:

$$\phi^{-\frac{1}{2}} = -2.5 \ln(0.27\ \epsilon/D + 0.885\ Re^{-1}\phi^{-\frac{1}{2}}) \qquad (7.1)$$

as used by Gay and Middleton (6.6) in their test example. Their network consisted of 38 pipes, each 100 feet long, 6 inches in diameter and considered hydraulically smooth. It was decided to choose an initial value of $\phi = 0.005$ as being representative of the range in which the flows should occur; from this it was possible to use equation (7.1) to obtain the initial Reynolds Numbers:

$$Re = 0.885\ \phi^{-\frac{1}{2}}/(\exp(-\phi^{-\frac{1}{2}}/2.5) - 0.27\ \epsilon/D) \qquad (7.2)$$

From equation (7.2) the branch flows $\mathbf{J}$ could be calculated using

$$J = \pi\ D\ \mu\ Re\ /\ 4\rho \qquad (7.3)$$

where ρ is the density, J the quantity flow rate in cusecs, D the diameter and μ the coefficient of viscosity, taken as 0.000 672 in these calculations. This gave for each branch:

$$J = 0.00000423\ Re \qquad (7.4)$$

When the approximate flows had been calculated it was possible to compute the first fluid resistances approximately. Considering the branch flows and the nodal flows in cusecs, it is convenient to calculate in heads rather than pressures. With L feet the length of pipe and u the fluid velocity in feet per second, the formula:

$$\Delta p = 4\ \phi\ L\ u^2\rho/\ D \qquad (7.5)$$

can be adapted to give the linearised fluid resistance:

$$\Delta h = \left[\frac{64\ \phi\ L}{\pi^2\ D^5}\rho.|J|\right] Q \qquad (7.6)$$

The new flow Q can then be calculated by considering the system as instantaneously linear and solving it by using the Hamiltonian path method. A comparison of the new flows and the old flows can be made to determine whether the latter are accurate enough for the required purposes or whether to update the **J** array of the previous flows with the **Q** array of the flows just calculated. As the new flows affect the Reynolds Numbers for the branches these can be calculated to determine the new values of ϕ for each branch. Also at this stage some form of convergence accelerator can be used, the average of the last two flows was taken. The organisation of the calculation is shown in Fig. 7.1. Due to the non-linear nature of the flow it is necessary to recalculate the fluid resistances before the next linear solution in a program such as HAMIL9.

Implementation of the system for interactive computer solution was made by the chained Algorithms COL1, HAMIL9 and DISP3 (q.v.) and examples of their use are given in EXAMPLES following the Algorithm Appendix.

The computation for the non-linear case is of interest when compared with previous solutions. The interactive working has the advantage that the pipe lengths, diameters and roughness (for the whole system with these programs) can be varied on input from case to case; this allows a study of flow and pressure changes with ageing to be performed rapidly if desired. The data for input having been reduced to the minimum thus reduces input time, fatigue and error; also errors can be corrected rapidly, the print-out of data for validation assists with this. Convergence to a stable solution can be fairly rapid, depending upon the programmed accuracy constant (which itself can be altered by the change of one program line).

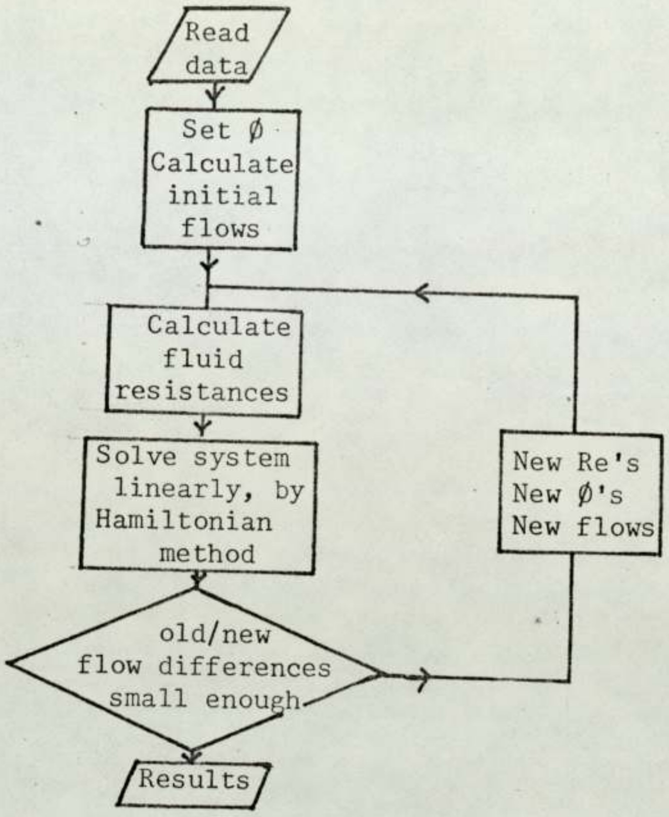

Read data
Set ∅ Calculate initial flows
Calculate fluid resistances
Solve system linearly, by Hamiltonian method
old/new flow differences small enough
Results
New Re's New ∅'s New flows

Fig. 7.1

It was found, for example, that the Gay and Middleton (6.6) test network could be entered by teletype and converged in four iterations in 7 minutes, the flow accuracy being set at 10%. This time could probably be reduced by using a data tape input prepared off-line and by excising the print-out of the final fluid resistances. In this case the HAMIL9 program involved a 17x17 matrix inversion without further printing other than to count the loops, a delay of about 7 seconds a line.

The first non-linear example referenced is a network similar to one of Gay and Preece (6.8) in which there are 9 nodes, 12 branches, 4 nodal flows but with all the pipes 100 feet long, 6 inches in diameter and with a relative roughness of 0.000 3. The second non-linear example is that of Gay and Middleton (6.6) computed to a flow accuracy within 0.1% of the previous individual pipe flow.

CHAPTER 8

User's Guide to the Chained Algorithms

Details of the algorithms are included with their descriptions in the Algorithms section of the Appendix but it is convenient to consider the overall handling of data for problem solving at this stage. The initial assumptions are that flows in a network have to be determined and that the network has already been numbered for nodes and branches. Details of the fluid resistances, pump pressures and nodal flows are also available; the resistances are linear. In order to contribute to the data handling the branches of the network are taken in natural order and described as being directed from the smaller node number to the larger node number; this then directs the branch and the pump pressure in that branch, being - if in the same direction, + if opposite. The nodal flows are prefixed (+) when directed to the node, - if away from the node.

The size of the matrix to be inverted is m by m. Because the Gay and Preece (6.8) Equation 14 is to be used, this has to be put into HAMIL8, line 22 as a DIMENSION statement e.g.

```
    22   DIM C(4,4),P(4), F(4)
```

Instructions are incorporated in the algorithms, which also print when they have been entered in the chaining process, though this is partly for diagnostic use. Starting with

```
GET-ATOM21
```

from the User Library, the first data asked for are nodes and branches

```
n,b
```

is entered. An upper limit of these has to be fixed because of program and storage size considerations. Generally, the final number of nodes provided for was 41 and the number of branches 40 (originally it was 80);

the size is checkable by inspecting A(40,2) in line 10, the common statement of any chained program in the suite.

The branches are next entered in natural order, low node to high

1,2,2,3,2,5,5,7,........... etc.

An immediate print out of this occurs, allowing for data errors and correction errors such as ...5,6←5,... The branches are in original natural sequence downwards e.g. line 9 is branch 9. The node-by-node **M** array is constructed by this algorithm, its printout is for checking purposes. This can be performed by taking any node, finding its column then seeing if the other nodes listed in this column are joined to it. Reading down the columns should show low numbers (nodes) to high. The final row is all zeroes, the end of column markers (originally -1 was used). The depth of the **M** matrix is dependent on how the nodes are connected; the algorithm does not count rows for this but truncates at the first row of zeroes reached. Modifications would have to take this into account, M(20,40) is assumed as an arbitrary depth of 20.

There is not enough storage space to allow this program a Hamiltonian path search so the **M** algorithm search is entered, MAL41. This starts at node 1 (arbitrarily), goes to the first node in its column, goes to that column and so on. If all the nodes can be reached in this way they are printed from the J(I) array and that is the Hamiltonian path. However it is quite likely that a node to go to has already been visited, in this case the next node down the column is taken and the process tried again. If this is repeatedly unsuccessful the end of column terminator 0 is reached and the process must step back to the previous node column then down one row from the node there that was last used. It is important that the process of stepping forward and stepping back avoids a loop of any length in the network, probably the most frustrating and difficult error to encounter. When all the paths have been tested the process is checked back to node 1. Failure to find a path has a message giving the starting node and that it has been checked back to node 1. The next starting node is tried similarly. When all nodes have been tried as starting nodes a message that all nodes have been tested is given, indicating that a Hamiltonian path does not exist in the network. It is sometimes useful to check this (again) with the MR algorithm which looks for 4 or more subgraphs, the proof that no Hamiltonian path exists. At the successful end the numbers 0,1,2,... for λ are printed below the J(I) path to indicate the numerical correspondence of nodes.

The next input is for original resistances and pump pressures, in pairs, of the original branches. This is then followed by an input of nodal flows, + if directed to the node, - if away. A reminder is given and it is noted that all nodal flows are entered, even any original datum node flow. The algorithm BREN5 renumbers the branches for the pattern, sorting the tree branches from the link branches, re-directing pump pressures where necessary and shuffling the nodal flows.

The algorithm HAMIL8 solves the system for the mesh flows i'. In effect at this stage the original network has been transformed to a Hamiltonian path network and this transformed to a mesh system for solution

$$\mathbf{V}' = \mathbf{Z}'\mathbf{J}' \tag{8.1}$$

which is solved by inversion. The indirect method of creating the $\mathbf{Z}'$ matrix means that it has off-diagonal elements.

The algorithm DISP adjusts the flows to the original network numbering. The sign with the flow is the direction in relation to the network and is useful for diagnosing errors, a modulus of the flow was found to conceal any direction error. Numerical checking is straightforward; as the tree pressure rises are also printed out they can contribute to this.

CHAPTER 9

Results, discussion, comparison

The lengthiest part of the calculation was considered to be that taken up by inversion. This was regarded at the outset for the purpose of familiarisation and time reduction if possible. The Gauss-Jordan method used by Brameller, Allan and Hamam (9.1) was tried on the ICL 1903A computer for various matrix sizes and ICL times noted. The ICL package FPMGEIN was thought to be worth comparing with as it was designed for making the most of the hardware. The comparison is indicated for some test matrices in Table 1.

Table 1. Comparison of Matrix Inversion Times.

Brameller, Allan & Hamam				ICL package		
Matrix Size	Mill Time	Occupation Time min.sec	K words	Mill Time	Occupation Time min.sec	K words
0(compile)	0.10	0.33		0.06	0.24	138
4x4	0.06	0.26	139	0.06	0.25	138
8x8	0.06	0.25		0.06	0.25	138
17x17	0.12	0.29		0.09	0.29	139
21x21	0.13	0.33		0.10*	0.30	139
$\sum$ 4,8,17,21	0.17	0.39	$\sum$above	0.19	0.41	142
40x40	0.48	1.14	143	0.31	0.54	143
40x40 +Trace 2	4.11	4.37				
$\sum$ 4,17	0.09	0.29				

*with re-inversion test

The fastest inversion time was thought by Gay and Preece (6.8) to be by using the Caffrey (9.2) algorithm; when compared with a Gauss-Jordan method it took about 43% of the time. A similar comparison here shows that the ICL FPMGEIN package is about 35% faster than the Brameller-Allan-Hamam Gauss-Jordan inversion. Comparisons are difficult to make because of differing packages and their program efficiencies. The occupation time is also involved with the data handling as well as the type of hardware, for example the Titan (Atlas II) of Bending and Hutchison (2.17) taking 2.1 seconds on the 17-mesh example mentioned earlier.

It would be expected that a manufacturer's package would be written in a low-level language to make the process faster and more economic in storage space.

When a 17x17 matrix was inverted on the Hewlett-Packard 2000E on an interactive basis there was no visible delay, the time taken was masked by the slow rate of printing; it was barely detectable on the visual display unit, again because of the data validation print precautions.

In direct comparison with the Gay and Preece (6.8) problem the approach offered some interesting features;

1. The choice of the tree can be made automatically by algorithm. The overlap considerations would appear to imply that this is neither maximum nor minimum.
2. The data input: here all the nodal flows can be entered if the datum node is to be found by an algorithm. The branch resistances and pressure sources are entered in the same way.
3. The topological matrices **C** and **B** are not entered, saving handling, space, time and possibility of error and checking. Instead the **A** matrix, at most a bx2 array, can be used.
4. The Equation 14 of Preece, a very economical method of solution, is created differently; the first array to be inverted $\tilde{C}_t Z_t C_t + Z_\ell$ is found by putting the mesh resistance sums as the diagonal elements

and mutual mesh resistances as the off-diagonal elements -- a summation process rather than a matrix transposition, triple matrix product and sum. (This is not necessarily the same for the electrical case where mutual effects and special devices such as gyrators would require modifications for the matrix operations in program form.)

5. The second factor $E_\ell' - \tilde{C}_t Z_t B_t I'$ has the product $B_t I'$ as a summation and can be calculated over i meshes of j tree branches from

$$\left\{E_\ell + \tilde{C}_t(E_t - Z_t B_t I')\right\}_i = E_{i+n-1} - \sum_{j=M(i,1)+1}^{M(i,2)} (E_t - Z_t B_t I')_j \qquad (9.1)$$

where $n-1 < i \leq b$. The premultiplication by $\tilde{C}_t$ distributes the tree branch potentials to the meshes, but it can be replaced because of the network numbering by summing from the lower node of each link, going up the tree as far as the link's upper node, thereby dispensing with C_t, and as C_ℓ is known, C altogether.

6. In the calculation of the branch flows, where for mesh i the end-nodes are M(i,1) and M(i,2) respectively,

$$J = \begin{bmatrix} I' + C_t i' \\ \cdots\cdots \\ i' \end{bmatrix} \qquad (9.2)$$

C_t assigns the mesh flows to the tree branches of that mesh and can be expressed as an addition

$$-(C_t i')_j = \sum_{k=1}^{m} f_k \qquad (9.3)$$

where $f_k = (i')_k$ if $M(k,1)+1 \leq j \leq M(k,2)$ or $f_k = 0$ for all other cases. C_t can again be eliminated because the network has been ordered and directed, so the tree links in each mesh are known from the start and finish of the mesh-defining link. As the tree branch directions are always opposed to the link directions (by convention) all non-zero terms are -1, hence the minus sign in equations 9.1 and 9.3.

The numerical results all compare favourably with those published, in particular the Gay and Middleton (6.6) non-linear test example was very interesting. When calculated to an accuracy of 0.1%, that is $|\text{Flow2} - \text{Flow1}| \leqslant |0.001\ \text{Flow2}|$ as included in EXAMPLES (q.v.), it was found that the greatest difference of flow (in branch 15) was 8.03%. Working with the printed pressure drops at 3 decimal places showed for KVL that each of the 17 meshes had an error less than 0.000 0, while working with 5 decimal places from the flow results gave for KCL an error of 0.000 4 (at node 20, with the smallest flow of the network in branch 20), the rest being better than 0.000 0. This implies that the 8% maximum error lies with the Hardy Cross method.

The convergence to a solution was investigated for different accuracies and is shown in Table 9.2.

Table 9.2. Gay and Middleton (6.6) Test Example, using Λ

% accuracy	Iterations	Time (approx.), min.
10	4	7
1	7	17
0.1	11	12
0.01	17	10
0.001	100 oscillates	19

The times are not really significant, they included putting in all the data by hand (not tape) and also depended on the time-sharing load. Altering the relative roughness to 0.000 3 required the same number of iterations for the first four accuracies listed. At 10% accuracy the maximum error against the Middleton results (E1) was just below 8%, while the 4 iterations here compare favourably with the 15 quoted.

It is concluded that this method has the potential for giving faster, more accurate results with quicker convergence and less data handling than other methods and in particular outruns Hardy Cross.

CHAPTER 10

Conclusion, Summary of Theory and Further Research.

At the close of this thesis it is apposite to consider the success of the methods and their influence on the results generally as they will provide pointers for future work. The conjecture that a Hamiltonian path exists in practical networks allows the ρ method to be used; this orders the network and together with a branch direction convention implicitly numbers it. In such a network it appears that the description has been reduced to that of showing where the links are connected. Topological matric solution now has the advantage that $\mathbf{C}$ and $\tilde{\mathbf{C}}$ can be dispensed with, $\mathbf{B}_t$ is now defined as ρ, so leaving some form of $\mathbf{A}$ for link description. Because $\tilde{\rho}^{-1}$ is now known, only $\mathbf{A}_\ell$ is needed and it can be handled not as a sparse matrix but as the smaller node-connection array of m pairs of numbers.

The solution of the network can occur at one of two levels, those of the expert and non-expert network investigator. The expert most probably has to draw up a master layout for calculation purposes, normally this would be unnumbered or numbered with dummy ciphers for identification. It is then easy to find a Hamiltonian path by eye from any node as datum, or in the case where the proposed datum is not a starting point for a Hamiltonian path then from the nearest suitable node to it. The numbers assigned, say from 0 upwards naturally, then order the network and only the minimum data input is thus required. The solution can then be made orthogonally by summing mesh resistances and by finding mutual mesh resistances. In the linear case the inversion then leads straightforwardly to the solution; the non-linear case requires the resistances to be found and iteration to give convergence to a solution.

In the case of the non-expert it is possible to handle all the information by suitable algorithms of some complexity. Data is entered under the system as numbered, a Hamiltonian path is found and the network automatically re-numbered. Solution then takes place and the results output in terms of the original numbering. The matrix size for inversion is the same as for the expert, no extra time would be needed for this but time for the data handling, validation print-outs and searching would be needed. In the case where a network does not have a Hamiltonian path (so far not encountered in practice) an algorithm can state this. It is assumed that the Mullineux-Reed (6.2) test has been applied before attempting solution because it definitely shows when no Hamiltonian path can exist. If required, the check product $\tilde{\mathbf{C}}\mathbf{e}$ can be added at the end with a small algorithm generating $\mathbf{C}_t$ from the link data entered.

The Λ method has advantages and disadvantages; being based on the conjecture that at least one Hamiltonian path exists in the practical network the disadvantage is that it must be found. When it has been found then the path defines the tree, links and meshes; the numbering can then be natural or implicit; with the disadvantage that an already-numbered network would have to be re-numbered and a housekeeping system set up and kept, together with handling algorithms. The major advantage of defining $\mathbf{B}_t$ as Λ makes $\mathbf{B}$ implicitly known and leads to dispensing with $\mathbf{C}$, $\tilde{\mathbf{C}}$ and $\mathbf{A}_t$, while the only disadvantage here appears to be that if the other matrices are required they would have to be generated algorithmically, via $\mathbf{C}_t = -\Lambda\tilde{\mathbf{A}}_\ell$ etc. As far as data input is concerned, at most only a bx2 array is needed to describe the network (at least, an mx2 array) an advantage which has no matching drawback. If Λ is used then Λ^{-1} is known for any size and hence all the formulas involving these two operators present little difficulty in operation. For input in the Λ method only a node-to-node branch list is needed to describe the network instead of an A matrix, for example. The nodal pressures are easily calculated up the tree now that B_t is ordered, an operation sums the branch pressure rises. However, the algorithms themselves present disadvantages in that they can be difficult to construct and guarantee; neither are they dynamic and it is easy to hypothesize that unforeseen difficulties may occur. Many programs were developed to implement the algorithms for computer operation, the kernel being that for finding a Hamiltonian path in a network. Also, diagnostic programs were constructed and proved useful but risked exceeding the limited storage size allowed for one program.

Compared with other methods the ρ approach is different and arguably simpler, for example that of Mah (4.2) which can leave pendant nodes. The SEARCH BT algorithm of Preece (6.1) has been obviated but at the expense of devising an algorithm for finding a Hamiltonian path. The ρ process appears to be one step beyond Mah's claim (4.2) '...that graph-theoretic techniques are used directly to enhance computational efficiency..' in that by postulating a route the graph-theoretic methods are enhanced as well.

At this juncture it is convenient to summarise some definitions and properties of ρ as discussed in the text. Table 10.1 shows some of these properties and relationships with the topological matrices.

The theoretical results are promising, as well as being mathematically satisfying in relation to the trunk of Mullineux and Reed (6.2). Computations with this system have the advantage that several matrices are eliminated and some operations reduced; the main

Table 10.1 Some Properties of Λ

What it is, $$\Lambda = \begin{bmatrix} 1 & 1 & 1 & . & 1 \\ & 1 & 1 & . & 1 \\ & & 1 & . & 1 \\ & O & & . & 1 \\ & & & & 1 \end{bmatrix}$$ i.e. a square array (10.1)

Its inverse $$\Lambda^{-1} = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & O & \\ & & . & & \\ & O & & 1 & -1 \\ & & & & 1 \end{bmatrix} \qquad (10.2)$$

How it multiplies, converting matrix multiplication to cumulative addition:

$$\Lambda \{a,b,c,d\} = \{a+b+c+d,\ b+c+d,\ c+d,\ d\} \qquad (10.3)$$

$$\tilde{A}_t^{-1} = \Lambda \qquad (10.4)$$

$$B_t = \Lambda \qquad (10.5)$$

$$C_t = -\Lambda \tilde{A}_\ell \qquad (10.6)$$

$$D = A\tilde{\Lambda} \qquad (10.7)$$

$$D_\ell = -\tilde{C}_t \quad = A_\ell \tilde{\Lambda} \qquad (10.8)$$

$$\gamma = \left[B \,\vdots\, C \right] = \left[\begin{array}{c:c} \Lambda & C_t \\ \hdashline O & U \end{array} \right] \qquad (10.9)$$

matrix to disappear is B_t , which becomes implicit and is handled in the operator mode of ρ . The ordering process has neatness, simplicity and possibly elegance -- a long way from Sir W.R. Hamilton's original idea of a 'Tour of the World', sold for twenty-five guineas in 1859.

The interactive programs developed were chosen to explore the problem, the commencement being the tests for Hamiltonian paths or circuits in a network. When the positive tests of Dirac (4.4) and Pósa (4.5) had been completed the negative test (i.e. no path) of Mullineux and Reed (6.2) was programmed; its major difficulty was to define separability numerically then count the subgraphs. Input of the network was made by describing a branch by its end nodes but for the path-searching requirement these had to be converted to the $\mathbf{M}$ array of Roberts and Flores (6.3). The Hamiltonian path search program then found the first such path and renumbered the nodes. The branches had to be renumbered as it was assumed that all branches and nodes had been numbered in the raw data; a program was constructed for this. The solution program for the network used the new numbering because of the improved efficiency of computation. The results for this system then had to be converted to original branch and node numbering together with direction re-assignment; a program for this was also developed. Notes on the programs available are shown in Table 10.2.

Looking towards the future from this stance reveals several promising paths for research. The major requirement is for a proof of the existence of a Hamiltonian path in every practical network. No such proof has been found but the exclusion condition of Mullineux and Reed (6.2) is of great use.

The inspection of all Hamiltonian paths in a network might be of

Table 10.2 Programs available and used in solution

Name	Mark	Purpose
DIR	9	Dirac test for Hamiltonian circuit, followed by Pósa test.
MR	44	Mullineux-Reed test for no Hamiltonian path.
WHE	16	Tests if the network is a wheel.
ATOM	21	Converts nodal description of branches to $\mathbf{M}$.
MAL	41	Searches $\mathbf{M}$ for a Hamiltonian path.
BREN	5	Renumbers the branches according to Λ .
HAMIL	8 linear	Computes flows, pressures in Λ numbering.
	9 non-linear	
DISP	- linear	Displays results with original numbering.
DISP3	- non-linear	Displays results. Λ numbering assumed.
COL1		Colebrook-White solution for non-linear flow.

benefit where both the overlap problem and considerations for minimisation are concerned. This could also be of interest in the context of tree graphs for these networks. Faster selection methods need to be regarded as well.

The algorithms used here are based on experimental requirements, these can be blended for further economies in time, space, cost and data. The most obvious improvement has been that of the sparse (nodal) array of the links to define the network.

Λ itself may have further useful properties; the dissection of large networks into Λ-based elements needs investigation. The Travelling Salesman problem with a Λ approach should lead to a path-cost minimisation that may be of value. Other similar problems might be worth considering, especially as Λ can represent infinite grids.

The problems of convergence to a solution in the non-linear cases have usually been bound up with the method of solution, often Newton-Raphson. The next term in the expansion

$$f(x+h) = f + hf' + h^2f''/2 + \dots \qquad (10.10)$$

could be taken as in Bailey's method, with

$$h \simeq \frac{-f}{f' - ff''/2f'} \qquad (10.11)$$

or by the Laguerre method, as advocated by Wilkinson (10.1, 10.2),

$$h = -nf/f' \pm \left[\left(\frac{n-r}{r}\right)\left\{(n-1)(f')^2 - nf'f''\right\}\right]^{\frac{1}{2}} \qquad (10.12)$$

where n is the degree of the polynomial.

On a larger scale the Λ method should extend to non-planar networks and compressible flow problems, as well as to other disciplines yielding to similar methods of attack. Inversion times can be halved because of the symmetry of the mesh resistance matrix, but the asymmetrical case needs some consideration.

The aims of this research of investigating fluid network analysis, of trying to devise new techniques, of implementing them in new computer methods as well as compacting the concepts and reducing tedious data handling have been achieved with some success. The Λ concept in its many forms (a matrix, a route, an operator) is novel, it eliminates the B_t array and in effect removes B from the data, from handling and from storage space, leaving it to be programmed implicitly. The matrices C and $\tilde{C}$ need not be used and only A_ℓ need be entered to describe the network; even so it is not a sparse matrix in the usual sense but is a node-connection array. A_t is not needed but is known to any size. C_t can be generated from A_ℓ if required. The computing algorithms complement the theoretical considerations and enable computation, the most important being the search for a Hamiltonian path, to be made by means of a special node-connection array. There is also an advantage that checking is readily achieved by eye from the data output. The connection of these programs in a suite forms a powerful analytical tool, as shown in the detailed numerical results of the EXAMPLES. A theoretical advantage of the method advocated is that usually a Hamiltonian path is not unique and this does not affect the final outcome.

APPENDIX

Regula Falsi, attributed to Robert Recorde, 'Ground of Artes', 1558 edition, Folio Z4.

"Gesse at this woorke as happe doth leade.
By chaunce to truthe you may procede.
And first woorke by the question,
Although no truthe therein be don.
Suche falsehode is so good a grounde,
That truth by it may soon be founde.
From many bate to many mo,
From to fewe take to fewe also.
With to much ioyne to fewe againe,
To to fewe adde to manye plaine.
In crossewaies multiplye contrary kinde,
All truthe by falsehode for to fynde."

i.e. the solution of

$$ag_1 + b = f_1$$
$$ag_2 + b = f_2$$

which is

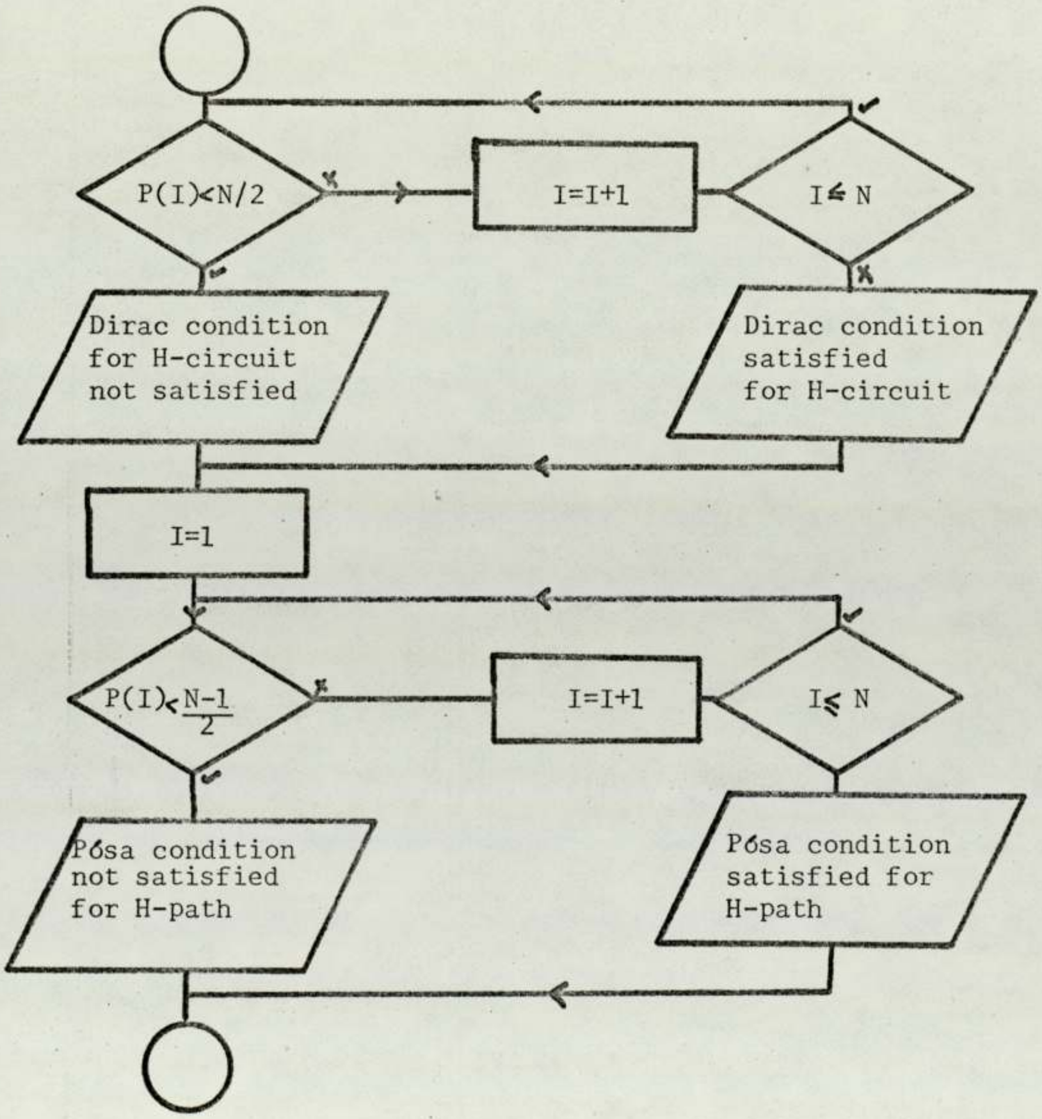
$$\begin{vmatrix} x & 1 & 0 \\ g_1 & 1 & -f_1 \\ g_2 & 1 & -f_2 \end{vmatrix} \qquad (1.6)$$

ALGORITHM DIR

The conditions of Dirac and Pósa can be used to test if a Hamiltonian circuit or path exists in a network. The former condition states that if the degree of every node is greater than or equal to n/2 then such a circuit exists. The Pósa condition is somewhat stronger, depending on whether the degree of each node is not less than (n-1)/2.

This algorithm is probably a reasonable investigatory test, a necessary and sufficient condition has not yet been found. The practical disadvantage is that most distribution networks have many nodes of small degree, as seen in the test example of Gay and Middleton (6.6) which has 22 nodes but the highest degree there is 5.

ALGORITHM DIR


P(I)<N/2
I=I+1
I≤N
Dirac condition for H-circuit not satisfied
Dirac condition satisfied for H-circuit
I=1
P(I)<(N-1)/2
I=I+1
I≤N
Pósa condition not satisfied for H-path
Pósa condition satisfied for H-path

```
LIS
DIR9

10  PRINT "TESTS FOR HAMILTONIAN CIRCUITS AND PATHS,
                                         DUE TO DIRAC"
20  PRINT "AND POSA. THESE ARE SUFFICIENT BUT NOT
                                        NECESSARY CONDITIONS."
30  PRINT "ENTER THE NUMBER OF NODES, N"
40  INPUT N
50  PRINT "ENTER THE DEGREE OF EACH NODE"
60  PRINT "NUMBER OF NODES";N
70  DIM P[40]
80  MAT P=ZER
90  LET I=1
100  FOR I=1 TO N
110  INPUT P
120  NEXT I
130  LET I=1
140  IF P[I]<N/2 THEN 200
150  I=I+1
160  IF I <= N THEN 140
170  PRINT "DIRAC CONDITION SATISFIED FOR H-CIRCUIT"
180  GOTO 201
190  IF I <= N THEN 140
200  PRINT "DIRAC CONDITION FOR H-CIRCUIT NOT SATISFIED"
201  LET I=1
210  IF P[I]<(N-1)/2 THEN 260
220  I=I+1
230  IF I <= N THEN 210
240  PRINT "POSA CONDITION SATISFIED FOR H-PATH"
250  GOTO 270
260  PRINT "POSA CONDITION NOT SATISFIED FOR H=PATH"
270  END
```

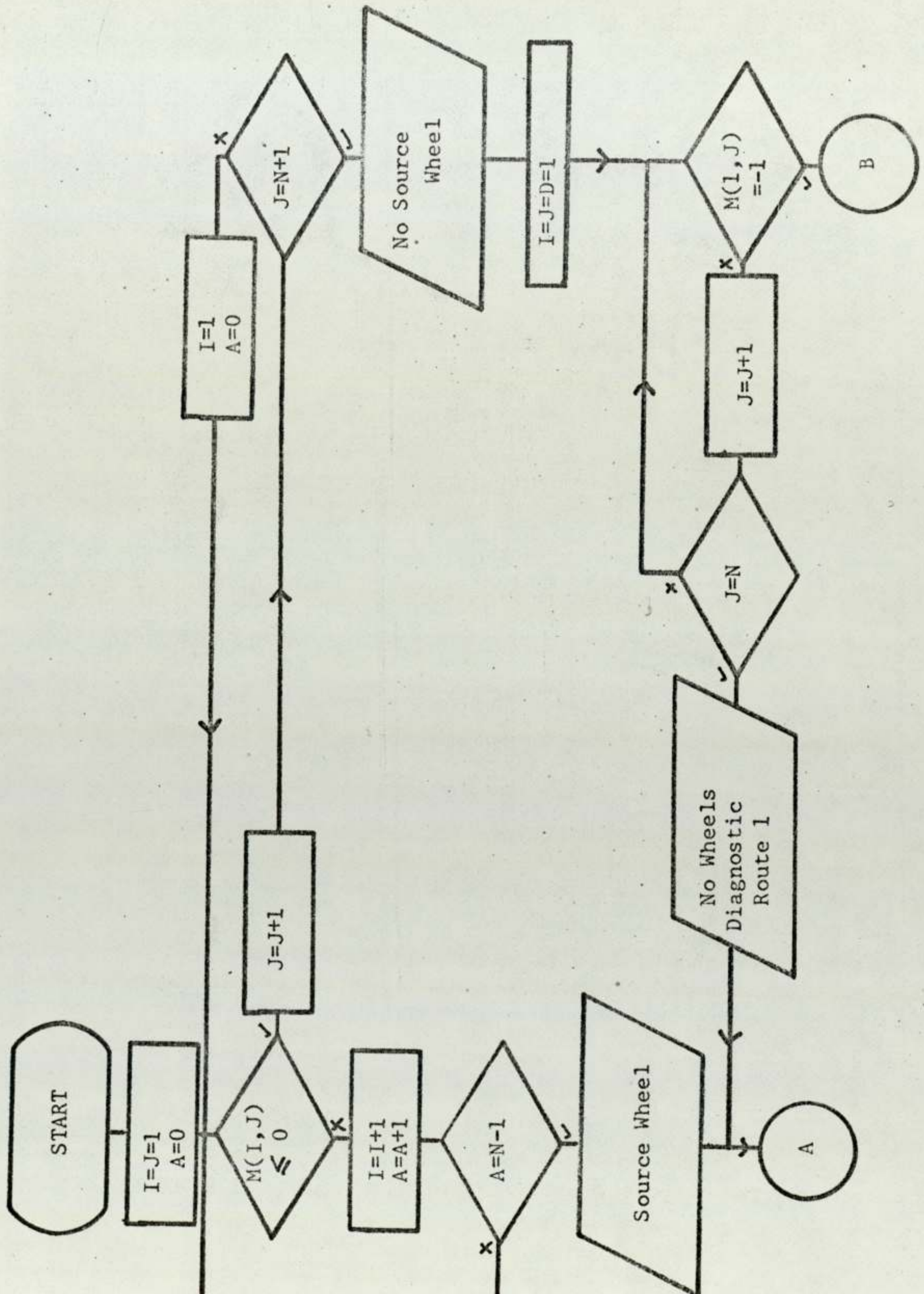
ALGORITHM WHE

This algorithm is designed to test whether a source or sink wheel exists in a digraph. Should such a wheel exist then a Hamiltonian path must start or finish respectively at the hub because the rim can only be reached once from the hub or the hub once from the rim; branches on the rim are disposed clockwise or anticlockwise exclusively.

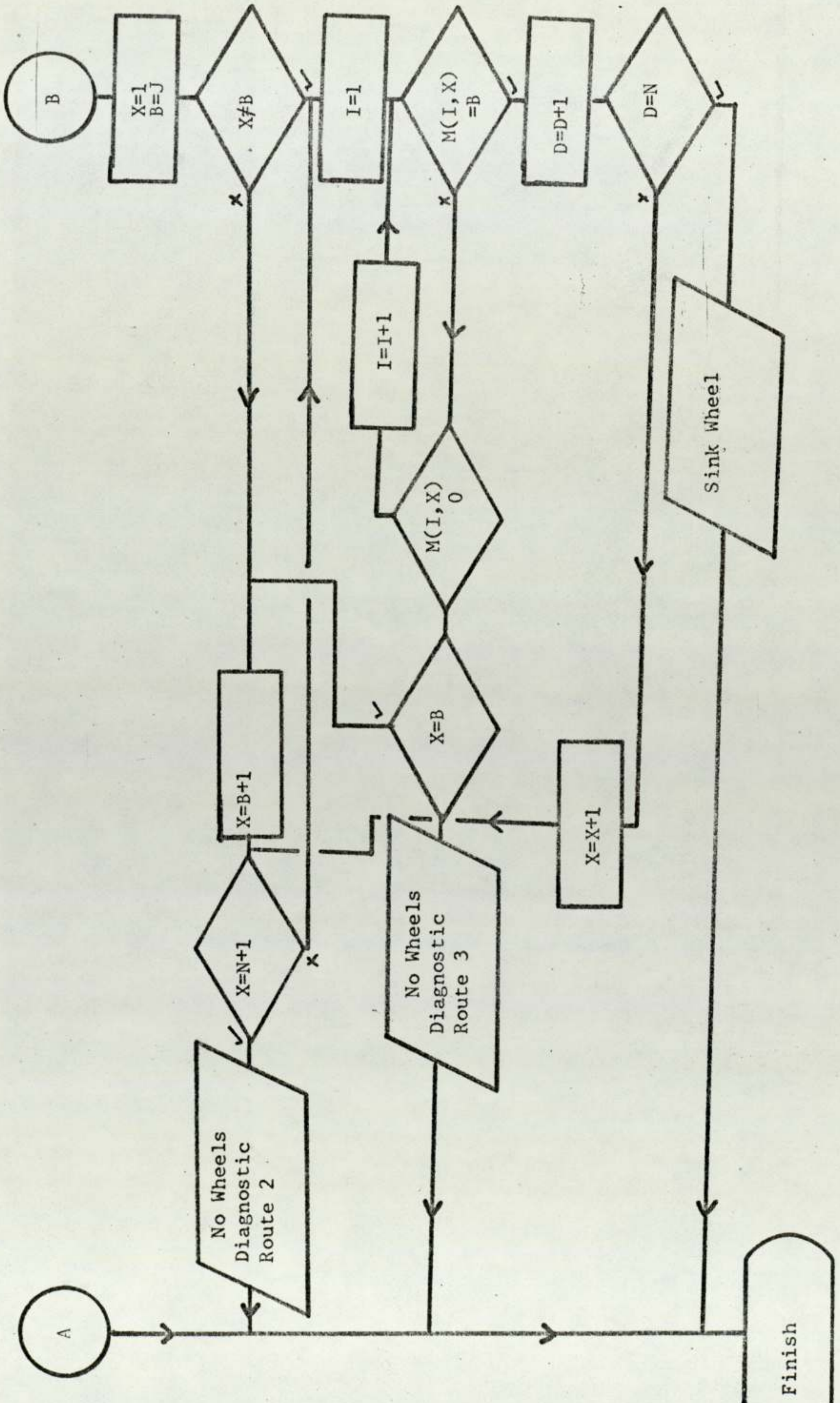
In the algorithm M(I,J) represents the element in the Roberts and Flores (6.3) **M** array; the number of rows of this has been called Y. The symbol A is a node counter. -1 is used as a bottom of column marker. The test is based on the fact that any hub must reach every node, if this is so then any commencement or termination (-1) on the first row of the **M** array would indicate a source or sink wheel. A hub is identified.

As the input is common to several programs it has not been shown on the flow diagram; it is the number of nodes, the **M** array and the number of rows it has.

ALGORITHM WHE



ALGORITHM WHE



WHE16

```
10  REM-TESTS IF THE NETWORK CONTAINS A WHEEL
20  PRINT "ENTER NODES N,ROWS OF M Y"
30  PRINT "ENTER M MATRIX"
40  INPUT N,Y
50  PRINT "NUMBER OF NODES";N
60  DIM M[40,40],X[1,40],J[40]
70  PRINT N
80  MAT  INPUT M[Y,N]
90  LET I=J=1
100  MAT  PRINT M
110  LET A=0
120  IF M[I,J] <= 0 THEN 170
130  LET I=I+1
140  LET A=A+1
150  IF A=N-1 THEN 480
160  GOTO 120
170  LET J=J+1
180  IF J=N+1 THEN 230
190  LET I=1
210  LET A=0
220  GOTO 120
230  PRINT "NO SOURCE WHEEL"
240  LET I=J=D=1
250  IF M[1,J]=-1 THEN 290
260  LET J=J+1
270  IF J=N THEN 500
280  GOTO 250
290  LET X=1
300  LET B=J
310  IF X <> B THEN 340
320  LET X=B+1
330  IF X=N+1 THEN 520
340  LET I=1
350  IF M[I,X]=B THEN 420
360  IF M[I,X] <= 0 THEN 380
370  GOTO 400
380  IF X=B THEN 320
390  GOTO 540
400  LET I=I+1
410  GOTO 350
420  LET D=D+1
430  IF D=N THEN 460
440  LET X=X+1
450  GOTO 330
460  PRINT "SINK WHEEL";B
470  GOTO 560
480  PRINT "SOURCE WHEEL, NODE";J
490  GOTO 550
500  PRINT "NO WHEELS, DIAGNOSTIC ROUTE 1"
510  GOTO 550
520  PRINT "NO WHEELS, DIAGNOSTIC ROUTE 2"
530  GOTO 550
540  PRINT "NO WHEELS, DIAGNOSTIC ROUTE 3"
550  STOP
560  END
```

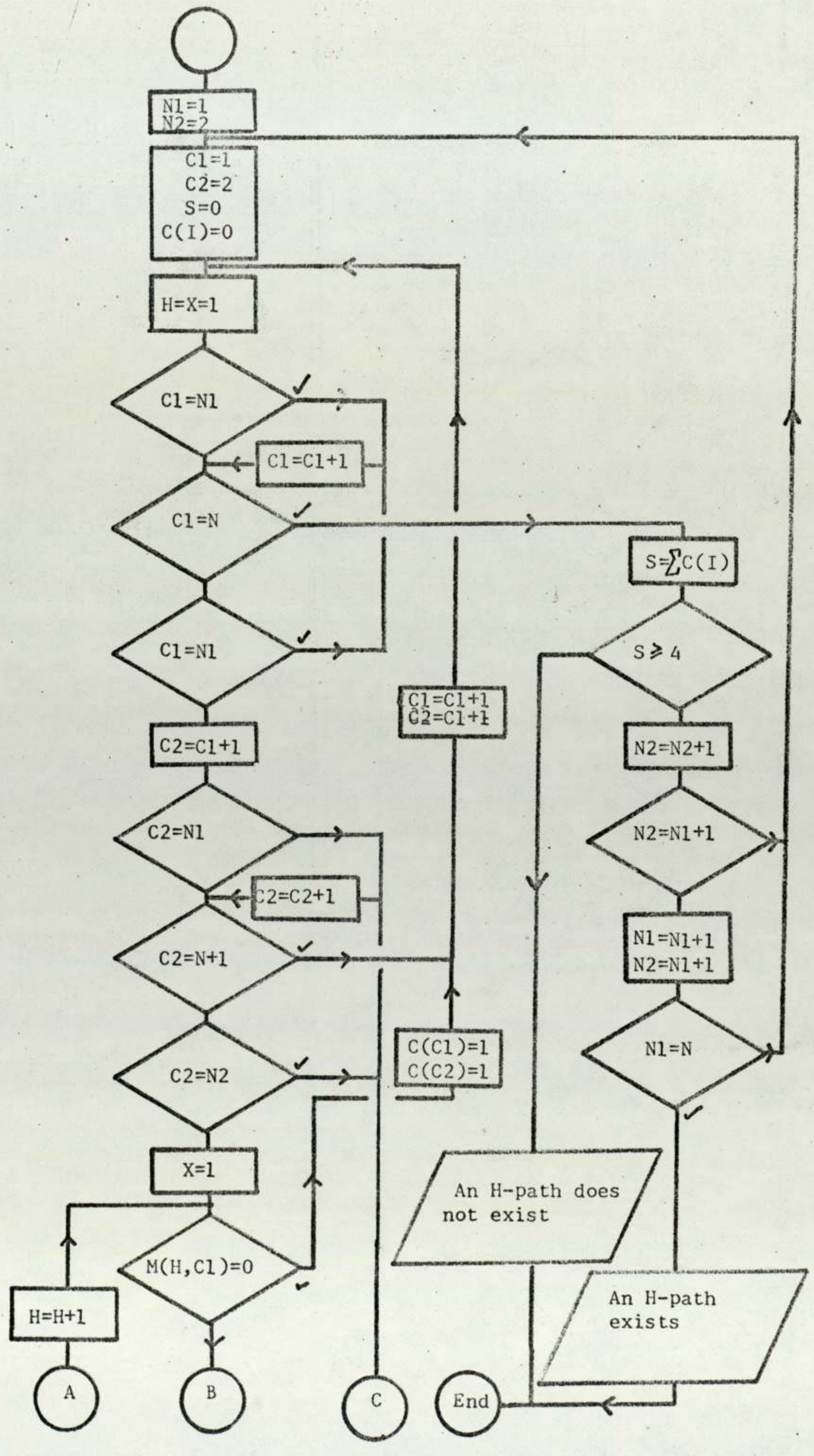
ALGORITHM MR

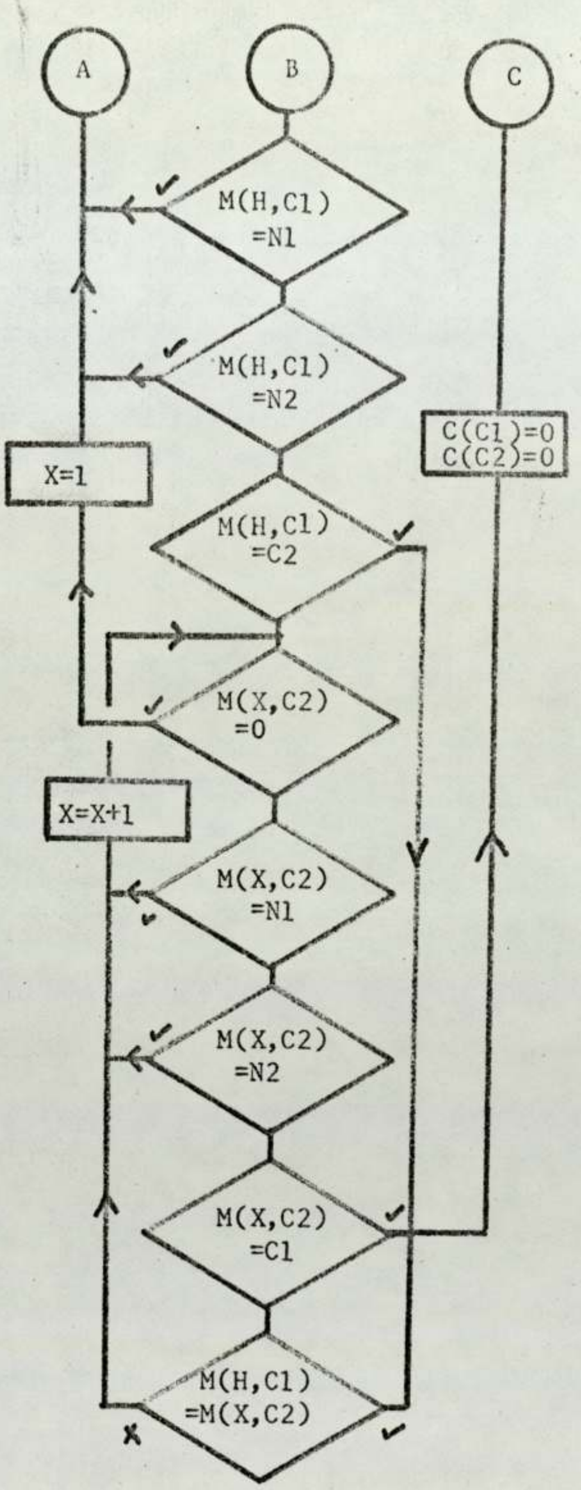
This algorithm is based on the test of Mullineux and Reed (6.2) which states that in a simple connected graph if the removal of two nodes and their incident branches leads to four or more subgraphs being formed then the graph has no Hamiltonian path.

The scheme here is to start with the separating nodes N1 and N2 at 1 and 2 then compare the columns C1 and C2 which run in order and are first compared with N1 and N2 and adjusted to be different from them. The Roberts and Flores (6.3) M array, which has been read in as data, is scanned down columns C1 and C2 to see if they have nodes in common. An array C(I) is used as a counter for every distinct subgraph. The end of a column is 0; when reached the second column is stepped on until all the nodes have been scanned, then C1 is advanced by1 if possible, the next C2 scanned and so on. At the end of this process the array C(I) is summed to see if it contains 4 or more. The appropriate output statement is made.

The beginning and end connectors are assumed to be chained between other programs.

ALGORITHM MR





ALGORITHM MR

MR44

```
10  INPUT N,Y
20  DIM M[40,40],C[40]
30  MAT  INPUT M[Y,N]
40  MAT  PRINT M
50  LET N1=1
60  LET N2=2
65  PRINT "A";
70  LET C1=1
80  LET C2=2
90  LET S=0
100  FOR I=1 TO 40
110  LET C[I]=0
120  NEXT I
125  PRINT "B";
130  LET H=X=1
140  IF C1=N1 THEN 165
160  GOTO 175
165  PRINT "C";
170  LET C1=C1+1
175  PRINT "D";
180  IF C1=N THEN 540
185  IF C1=N2 THEN 165
190  LET C2=C1+1
195  IF C2=N1 THEN 205
200  GOTO 215
205  PRINT "E";
210  LET C2=C2+1
215  PRINT "F";
220  IF C2=N+1 THEN 240
230  GOTO 250
240  LET C1=C1+1
241  LET C2=C1+1
242  GOTO 130
250  IF C2=N2 THEN 205
260  LET X=1
270  PRINT "G";
280  IF M[H,C1]=0 THEN 490
290  IF M[H,C1]=N1 THEN 330
300  IF M[H,C1]=N2 THEN 330
310  IF M[H,C1]=C2 THEN 510
320  GOTO 360
330  PRINT "H";
340  LET H=H+1
350  GOTO 270
360  PRINT "K";
370  IF M[X,C2]=0 THEN 390
380  GOTO 410
390  LET X=1
400  GOTO 330
410  IF M[X,C2]=N1 THEN 460
420  IF M[X,C2]=N2 THEN 460
430  IF M[X,C2]=C1 THEN 510
440  IF M[H,C1]=M[X,C2] THEN 460
450  GOTO 510
```

```
460  PRINT "L";
470  LET X=X+1
480  GOTO 360
490  LET C[C1]=C[C2]=1
500  GOTO 240
510  PRINT "M";
520  LET C[C1]=C[C2]=0
530  GOTO 205
540  PRINT "N";
550  FOR I=1 TO 40
560  LET S=S+C[I]
570  NEXT I
580  IF S >= 4 THEN 700
590  LET N2=N2+1
600  IF N2=N+1 THEN 630
610  PRINT "O";
620  GOTO 640
630  PRINT "P";
635  GOTO 65
640  LET N1=N1+1
650  LET N2=N1+1
660  IF N1=N THEN 680
670  GOTO 630
680  PRINT "AN H-PATH EXISTS"
690  GOTO 710
700  PRINT "AN H-PATH DOES NOT EXIST"
710  END
```

ALGORITHM ATOM

This algorithm is the first in a suite that will solve many network problems. It can have been connected to any previous tests for Hamiltonian circuits or paths if necessary. Assuming that a start is made here then the degrees of each node are not required for data but the nodes and branches are required for each program of the suite. The network is described by ordered input of the I branches, given by their end nodes, A(I,1) and A(I,2). The convention being used for this is that of low node number to high node number as branch direction (which might be arbitrary). Allowance is made for the higher valued node to be called L. Inconsistent numbering is checked and an error message given if found. The **M** of Roberts and Flores (6.3) is annotated M(K,L). The first node is then put into the a_{11} position of the **M** array and the branch number checked to see if all branches have been processed; if not, the next branch is put into the array. The counter K fits the node into its proper row in the first empty space of **M**, L represents the appropriate column.

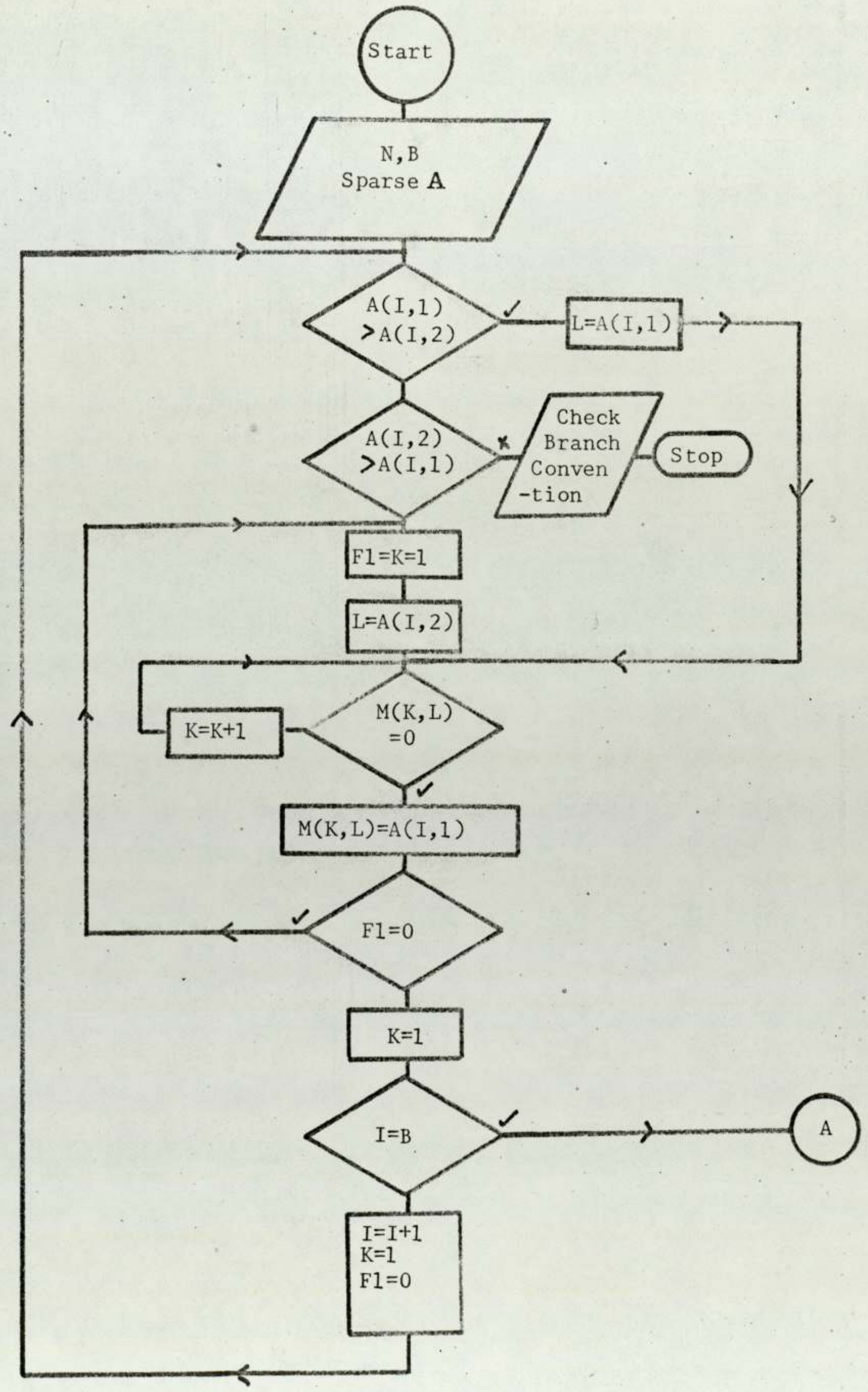
The nodes now in the array have not been ordered and have to be shuffled into low to high number order down each column. Various constants are reset.

It is possible for an element in the first row to be zero (e.g. for a wheel) so provision for this must be allowed. If a column of five 0's exists then the next column is commenced provided it is not the last; when it is, then the **M** array is printed so that the last row is all 0 and lower rows are suppressed.

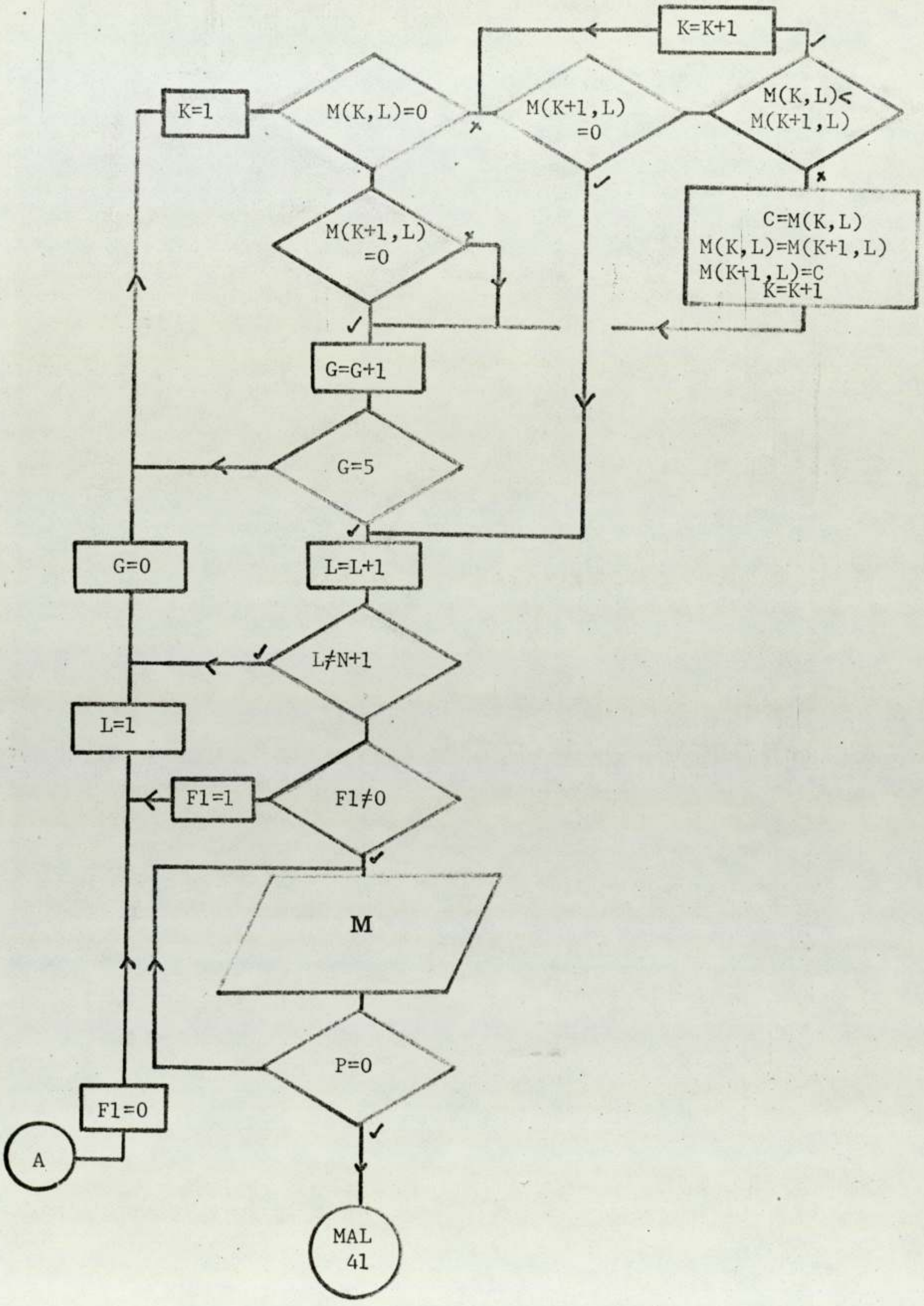
When the first element is not zero the next row down for that column is tested for zero. If that is not zero then the relative sizes of the two numbers are checked; if they are correctly aligned a row is stepped down and these two numbers checked and so on until the end of the column is reached. When two numbers are not in the required form they are interchanged and a counter G incremented by 1. The test G=5 is arbitrary and will probably have to be increased when there are many rows of **M**. The columns are shuffled in turn from L=1 to L=n.

The program has been extensively patched since its inception.

ALGORITHM ATOM



ALGORITHM ATOM



```
LIS
ATOM21

10  COM N,B,M[20,40],A[40,2],J[41],Z[40],E[40],I[40],
                                      V[50],B[40]
11  REM-CONVERTS THE SPARSE DIRECTED A MATRIX TO
                                   NODAL MATRIX M.
12  PRINT "INPUT N,B. MAT A IS READ NEXT"
13  PRINT "INPUT N=NODES, B=BRANCHES. N<=41, B<=80"
14  INPUT N,B
30  MAT M=ZER
40  LET I=J=K=L=C=1
41  PRINT "SPARSE A MATRIX, LOW NODE TO HIGH NODE"
42  F1=0
50  MAT  INPUT A[B,2]
60  MAT  PRINT A
70  IF A[I,1]>A[I,2] THEN 171
80  IF A[I,2]>A[I,1] THEN 111
90  PRINT "CHECK BRANCH CONVENTION"
100  STOP
110  LET F1=K=1
111  LET L=A[I.2]
120  IF M[K,L]=0 THEN 150
130  LET K=K+1
140  GOTO 120
150  LET M[K,L]=A[I.1]
151  IF F1=0 THEN 170
152  LET K=1
160  GOTO 220
170  LET F1=K=1
171  LET L=A[I,1]
180  IF M[K,L]=0 THEN 210
190  LET K=K+1
200  GOTO 180
210  LET M[K,L]=A[I,2]
211  IF F1=0 THEN 110
220  IF I=B THEN 260
230  LET I=I+1
240  LET K=1
245  LET F1=0
250  GOTO 70
260  LET F1=0
261  LET L=1
262  G=0
265  LET K=1
267  IF M[K,L]=0 THEN 275
269  IF M[K+1,L]=0 THEN 360
271  IF M[K,L]<M[K+1,L] THEN 302
273  GOTO 310
275  IF M[K+1,L]=0 THEN 280
280  G=G+1
285  IF G=5 THEN 360
290  GOTO 265
302  LET K=K+1
303  GOTO 267
310  LET C=M[K,L]
320  LET M[K,L]=M[K+1,L]
330  LET M[K+1,L]=C
341  LET K=K+1
350  GOTO 280
360  LET L=L+1
370  IF L#N+1 THEN 262
375  IF F1#0 THEN 390
380  F1=1
```

```
385  GOTO 261
390  FOR I=1 TO B
391  LET P=0
395  FOR J=1 TO N
400  PRINT M[I,J];
401  LET P=P+M[I,J]
405  NEXT J
406  PRINT
407  IF P=0 THEN 420
410  NEXT I
420  CHAIN "MAL41"
421  END
```

ALGORITHM MAL

The search for a Hamiltonian path in the **M** array is undertaken in this much modified algorithm. The various arrays and constants are first set to zero. The rows and columns markers J and L have already been set to 1 (not shown). J is tested for the last node and if so skips to a temporary Stop. The first element of the **M** array is tested for greater than zero; if this is not so then the column is tested for last node (i.e. last column of **M**). When a zero is found the column is stepped along to the next, which is also tested for zero. Two consecutive zeroes constitute an error, as only one first row zero, the sink hub of a wheel, can be entertained.

Assuming a normal **M** array, the X(1,J) array is begun with a -1, indicating the starting node. The J array is to hold the Hamiltonian path, hence the column number J is entered each time. K is a counter. The array W(J) is a device, superseding two flags, to enable a row to be stepped down smoothly from its row used previously. When X(1,J)=1 it means that J has already been used in the recent Hamiltonian path attempt and that an alternative must be searched for.

The counter K is tested to see if the process has been finished, if so the J array is printed, being the node succession in original numbering of the Hamiltonian path. The ρ path is defined as 0,1,2..... ...(n-1) and this is simply printed under the J succession (note the relationship is implicitly made). The next program, BREN 5, is chained.

When K does not indicate the end, the **M** array element is called T; if it is zero then it would mean a sink hub, but this must be at the end of the path and should have been found under the immediately previous

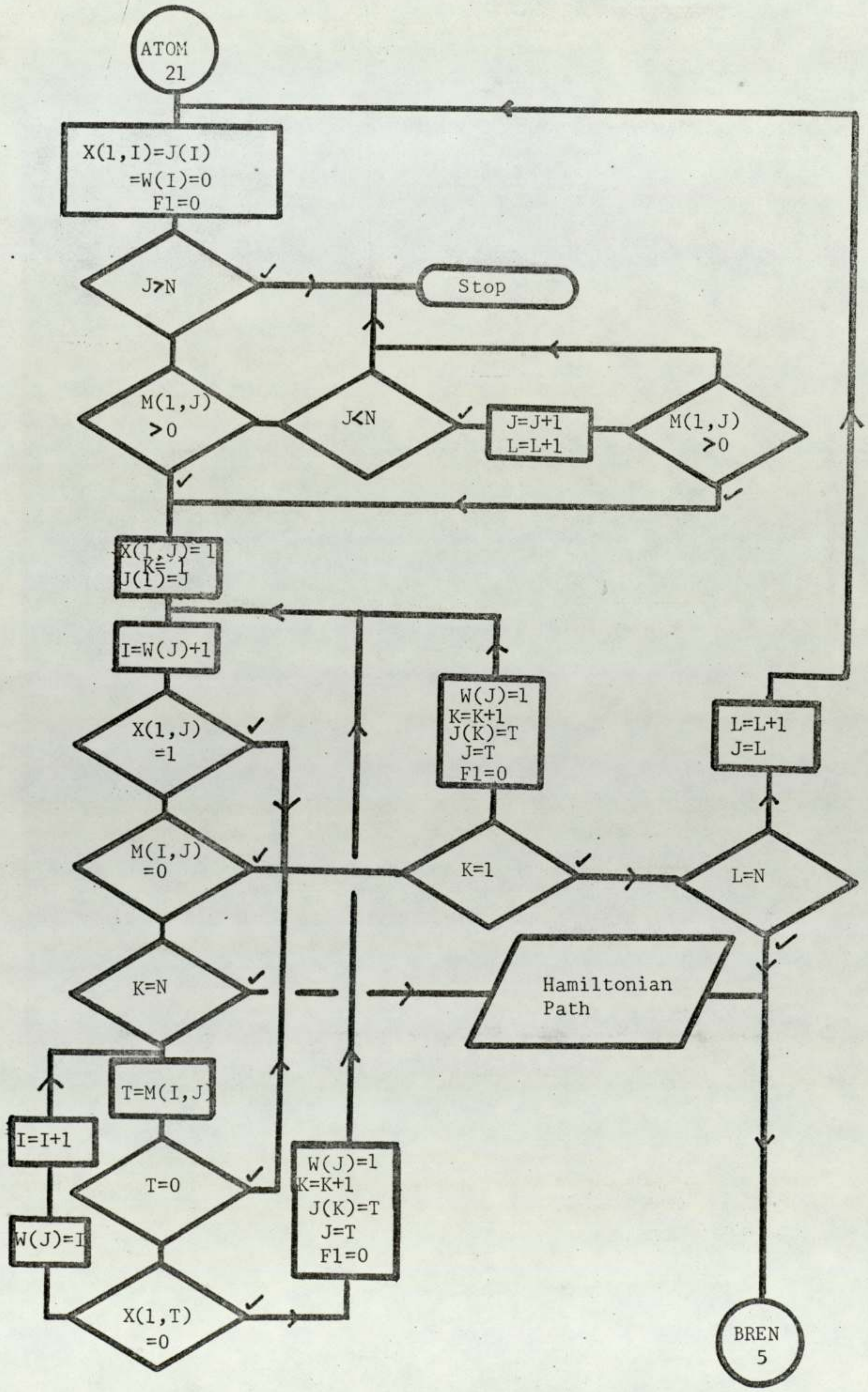
M(I,J)=0 test, therefore a step-back routine must be entered (see below). When T≠0 the X array is scanned to see if X(1,T) contains a marker 1; if it has been used (i.e. the node is already in the Hamiltonian chain) then the column is stepped down to the next number and the row number W(J) is incremented by 1. A loop is thus traversed until an unused node is encountered or the end of column zero is reached.

When the X array indicates that a node can be used, a 'step-on' routine is initiated; K is incremented, the node is added to the Hamiltonian node path string and the new column to be used next is noted. The next element of the **M** array is inspected similarly.

When a column has been tested until a zero is reached and it is not the last node, a 'step-back' procedure has to be entered. As stepping back must terminate at the starting node, the counter K is tested if =1. When K≠1 the X array for this column must be reset to 0 because it could be accessed from another node later on; the individual row marker W(J) is also set to zero. K is also stepped back by 1 and the previous node returned to. The previous node column has obviously been stepped down row by row, there is no need to start at the first row again because (a) this would lead to a looping procedure (b) if an earlier row had been successful there would be no need to step back; hence its W(J) is increased by 1.

When a Hamiltonian path cannot be reached from a node (see EXAMPLE) the step-back process leads to K=1 and hence a test if L, the column counter, has been tried for all nodes. If it has a message that all nodes have been tested is printed before stopping. When all nodes have not been tested the node counter L is incremented once, also J which then starts from its next node.

ALGORITHM MAL



MAL41

```
10  COM N,B,M[20,40],A[40,2],J[41],Z[40],E[40],I[40],V[50],
                                                        B[40]
12  PRINT "MAL41 ENTERED"
13  REM-USE FOR CASES THAT HAVE PASSED THE M-R TEST
50  DIM X[2,40],W[40]
80  LET J=L=1
90  LET F1=0
100  FOR I=1 TO 40
110  LET X[1,I]=J[I]=W[I]=X[2,I]=0
120  NEXT I
121  I=1
140  IF J>N THEN 750
150  IF M[1,J]>0 THEN 230
160  IF J<N THEN 180
170  PRINT "LAST NODE, STARTS WITH 0"
171  GOTO 750
180  LET J=J+1
190  LET L=L+1
200  IF M[1,J]>0 THEN 230
210  PRINT "TWO CONSECUTIVE 0 STARTS"
220  GOTO 750
230  LET X[1,J]=1
240  LET J[1]=J
250  LET K=1
260  LET I=W[J]+1
261  W[J]=I
263  IF F1=N THEN 410
265  IF M[I,J]=0 THEN 400
270  IF K=N THEN 610
280  LET T=M[I,J]
290  IF T=0 THEN 400
292  GOTO 370
325  GOTO 260
370  IF X[1,T]=0 THEN 535
380  LET I=I+1
385  W[J]=I
390  GOTO 280
400  IF K=1 THEN 580
410  LET X[1,J[K]]=0
411  W[J[K]]=0
415  X[2,J[K]]=1
420  LET K=K-1
440  LET J=J[K]
470  GOTO 260
535  LET K=K+1
540  LET J[K]=T
550  LET J=T
560  LET X[1,T]=1
564  FOR I=1 TO K
566  X[2,J[I]]=0
568  NEXT I
570  GOTO 260
580  PRINT L;"STARTING NODE, NO PATH; CHECKED BACK TO FIRST
                                                        NODE"
590  GOTO 700
610  PRINT "H-PATH ";
620  FOR I=1 TO N
630  PRINT J[I];
640  NEXT I
```

```
650  PRINT
660  PRINT "DIGAMMA";0;
670  FOR I=1 TO N-1
680  PRINT I;
690  NEXT I
691  PRINT
692  GOTO 751
700  IF L=N THEN 740
710  LET L=L+1
720  LET J=L
730  GOTO 90
740  PRINT "ALL NODES TESTED"
750  STOP
751  CHAIN "BREN5"
760  END
```

ALGORITHM BREN

The branches of the original network having been searched by the previous chained program MAL have now to be numbered according to the $\mathcal{H}$ path. At the end of the solution stage the old branch numbers must be referred to again; as the **M** array has been completed and need not be retained it can be discarded by overwriting with link information for the next program.

The original data of resistance (**U**), pressure (**S**) and all nodal flows (**W**) are entered in sequence from the old branch numbering. A branch array of the first node (a smaller ordinal than that of the second node), A(I,1) is tested against the Hamiltonian path array containing J(L) to find L. An error check is incorporated to allow for mis-typing. Similarly the node at the end of the branch is checked for its $\mathcal{H}$ number M. The branch direction convention is still necessary for the next chained program HAMIL so if the branches have 'changed ends' then the branch pressure source sign must be changed. This is achieved by testing $L<M$ and reversing the sign if not.

The next test simply sorts tree branches from link branches by seeing if the branch nodes are now consecutively numbered. An array of new branch numbers B(J) is formed by adjusting one of the node ordinals, for with datum node numbered 0 a branch to n is numbered n . The resistance and pressure for this branch are now set in arrays **Z** and **E** .

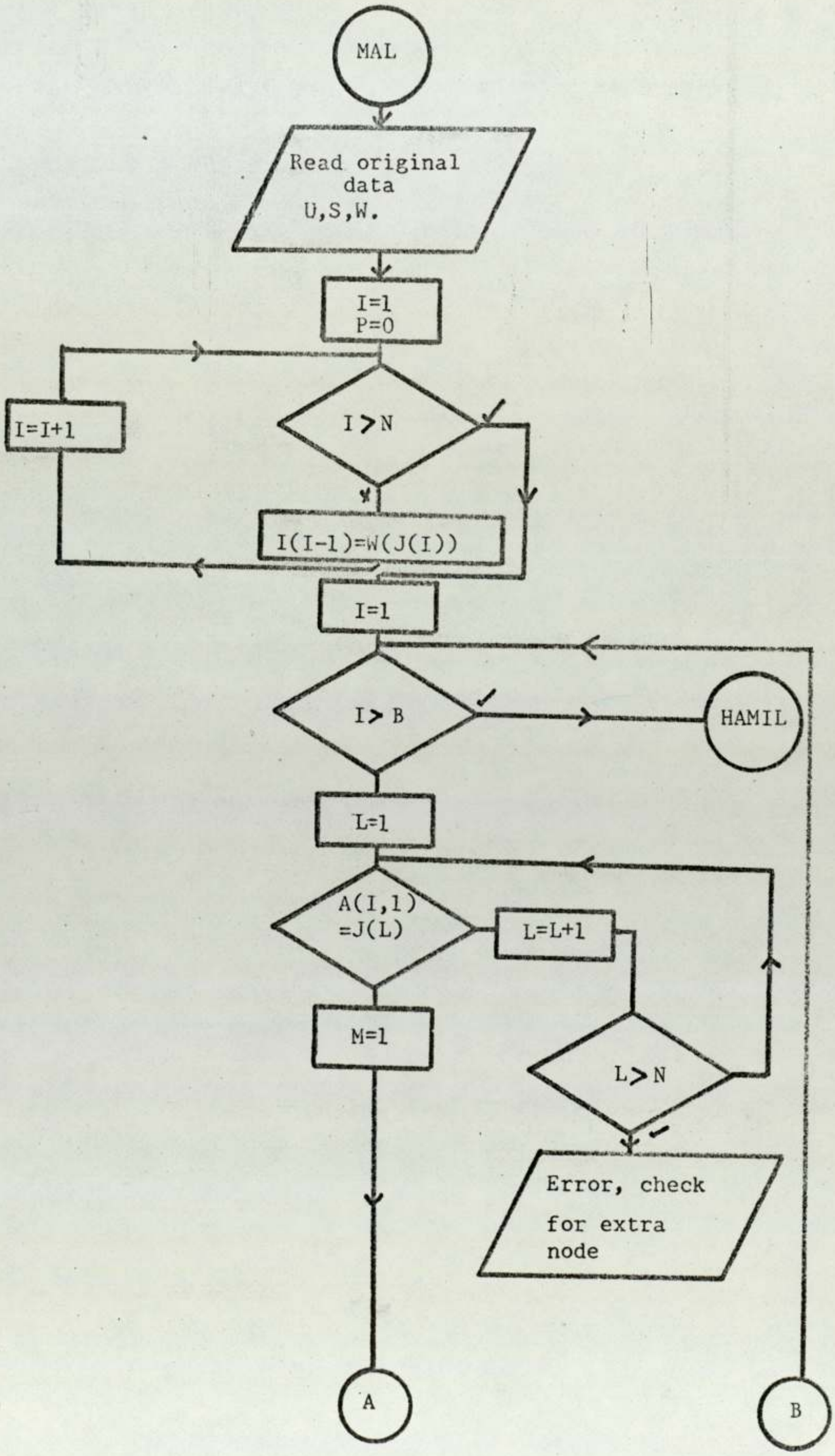
The link branches are numbered differently and put into the **M** array as M(P,1), M(P,2). P is the link counter (it also numbers the meshes, though this is not used). By adding P to (n-1) the new link branch resistances and pressure sources are entered in the **Z** and **E** arrays. The link branches are next tested and numbered according to the end nodes. Link arrays are set up as M(P,1), M(P,2) as the M array is common to the next program. The tree branches are now implicit.

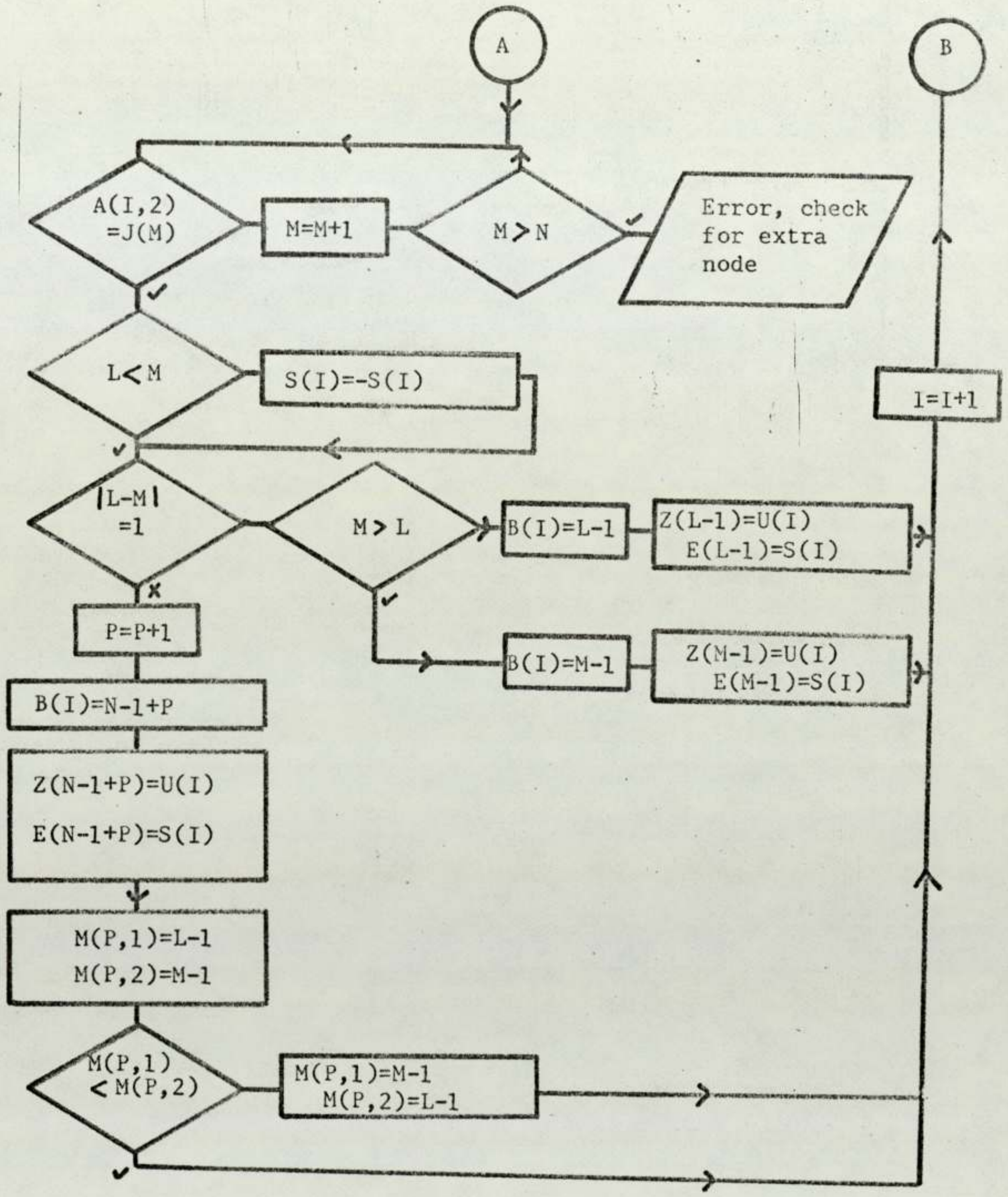
The process is repeated for all branches.

This algorithm was modified from earlier forms as the specifications changed, it therefore contains some redundant information.

The next program chained is HAMIL, the solution routine.

ALGORITHM BREN





ALGORITHM BREN

BREN5

```
10  COM N,B,M[20,40],A[40,2],J[41],Z[40],E[40],I[40],V[50],
                                                   B[40]
20  DIM U[80],S[80],W[40]
30  PRINT "BREN5 ENTERED"
40  PRINT "ENTER U (OLD Z), S (OLD E), ALTERNATELY"
50  FOR I=1 TO B
60  INPUT U[I],S[I]
70  NEXT I
75  PRINT "ENTER ALL NODAL FLOWS"
80  FOR J=1 TO N
90  INPUT W[J]
100  NEXT J
110  LET I=1
120  LET P=0
130  FOR I=2 TO N
140  I[I-1]=W[J[I]]
150  NEXT I
160  LET I=1
170  IF I>B THEN 560
180  LET L=1
190  IF A[I,1]=J[L] THEN 240
200  LET L=L+1
210  IF L>N THEN 230
220  GOTO 190
230  PRINT "ERROR,CHECK IF EXTRA NODE"
240  LET M=1
250  IF A[I,2]=J[M] THEN 295
260  LET M=M+1
270  IF M>N THEN 290
280  GOTO 250
290  PRINT "ERROR,CHECK IF EXTRA NODE"
295  IF L<M THEN 298
297  S[I]=-S[I]
298  GOTO 300
300  IF ABS(L-M)=1 THEN 340
310  LET P=P+1
320  LET B[I]=N-1+P
330  GOTO 470
340  IF M>L THEN 400
350  B[I]=L-1
360  Z[L-1]=U[I]
370  E[L-1]=S[I]
380  GOTO 450
400  LET B[I]=M-1
410  LET Z[M-1]=U[I]
420  LET E[M-1]=S[I]
450  LET I=I+1
460  GOTO 170
470  LET Z[N-1+P]=U[I]
480  LET E[N-1+P]=S[I]
481  LET M[P,1]=L-1
482  LET M[P,2]=M-1
483  IF M[P,1]<M[P,2] THEN 490
484  M[P,1]=M-1
485  M[P,2]=L-1
490  GOTO 450
560  CHAIN "HAMIL8"
565  END
```

ALGORITHM HAMIL

This algorithm performs the main calculation in the program suite for the solution of the network. The special numbering of the nodes, branches and links allows some of the structure to be known implicitly and the programming to be condensed. An explanation of the arrays used in this program is of assistance here because of certain storage reduction techniques. The **M** array was that of Roberts and Flores (6.3) but as the Hamiltonian path has already been found and the network renumbered it has no further use per se; it can be overwritten and so has been used in algorithm BREN as a convenient place to contain the new link numbering. **A** is still the original end-node numbering of the branches. The initial resistances **U** , pump pressures **S** and the nodal flows **W** (meaning I') have been shuffled into the arrays **Z** , **E** and **I** in their new numbering and were carried into HAMIL by the COMMON statement. **J** is the key array holding the ρ path node array (in original node numbers). **B** is an array containing the old branch node numbers, needed for reference later and carried from BREN. **Q** is the final array of flows found by HAMIL.

Intermediate arrays are needed for this particular program and are entered under a DIMENSION statement so that they can be lost on chaining the next program, DISP. They are **C** , an array built up piece by piece to be the fluid resistance matrix $(\tilde{C}_t Z_t C_t + Z_\ell)$, which is inverted into itself, **P** is used to find the net pressure in a mesh, incorporating the link pressure sources and the tree pressure rises given by **V** . **F** holds the mesh flows resulting from an Ohm's Law type of calculation.

Due to a limitation of this BASIC dialect it was not possible to use computed dimensions, which meant that the temporary arrays **C** , **P** and **F** had to be dimensioned manually before entering the program, ergo before entering the suite. As this can cause a dimensioning error diagnostic attention was drawn to the required provision at the commencement in ATOM21. To check this at the outset it is helpful to start the suite with

```
                GET-HAMIL8
                LIS-22,22
(output)        22  DIM C(4,4),P(4), F(4)
                22  DIM C(17,17), P(17), F(17)    (i.e. changing from a
                                                   4-mesh to a 17-mesh
                                                   network)
                KIL-HAMIL8 (H-P 2000E dialect)/PUR-HAMIL8 (H-P 2000 dialect)
                NAM-HAMIL8
                SAV
                GET-ATOM21
```

This has then set the size of C to be inverted.

The algorithm itself constructs the C array by first adding the link resistances into the diagonal matrix positions. The tree resistances have to be added next but are influenced by the way the links are arranged and how they overlap or otherwise. A counter J1 denotes the next branch after a node (i.e. by adding 1 to the node number), M(a,1) and M(a,2) represent the end nodes of branch a. Running up the tree the tree branch resistances are added into the diagonal positions of the matrix being formed, which should thus contain the sums of the appropriate mesh resistances. A counter J1 is set up for the next branch to the node under consideration; the off-diagonal element is zeroed and the ends of the branch inspected. They can be as in Fig. H1

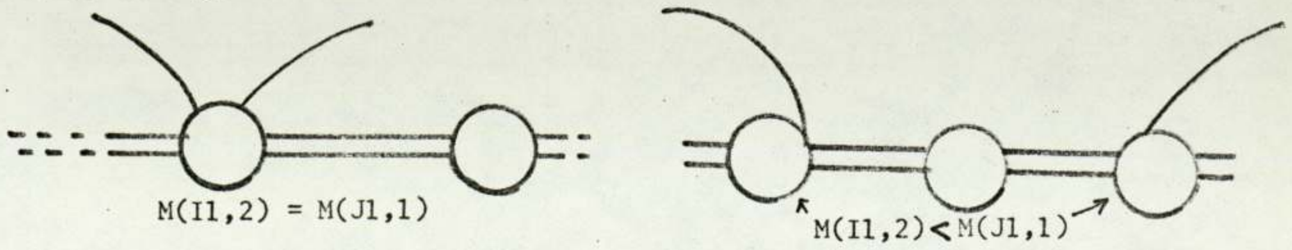


Fig. H1

and can be either left to right or right to left in general. A positive result means no mesh and so the next step is to set up the symmetrical component in **C** . When this test breaks down a counter K2 is set at the lower node number and 1 added to make it the next branch. A test is then carried out to see if the 'lower' node is smaller than the other end node; when it is, the end of the mesh is called K3. A further test is carried out to see if the mesh end has reached the other node end; if it has not then the calculation adds all the tree branch resistances for the mesh into the off-diagonal matrix position (the link resistances only appear in the elements of the leading diagonal). Again the symmetrical element is constructed. I1 being a mesh counter and J1 referring to links, the other cases are shown in Fig. H2.

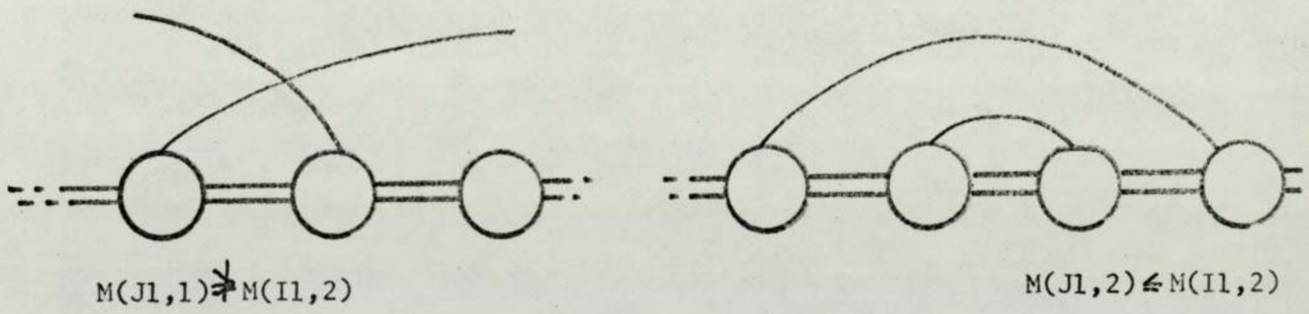


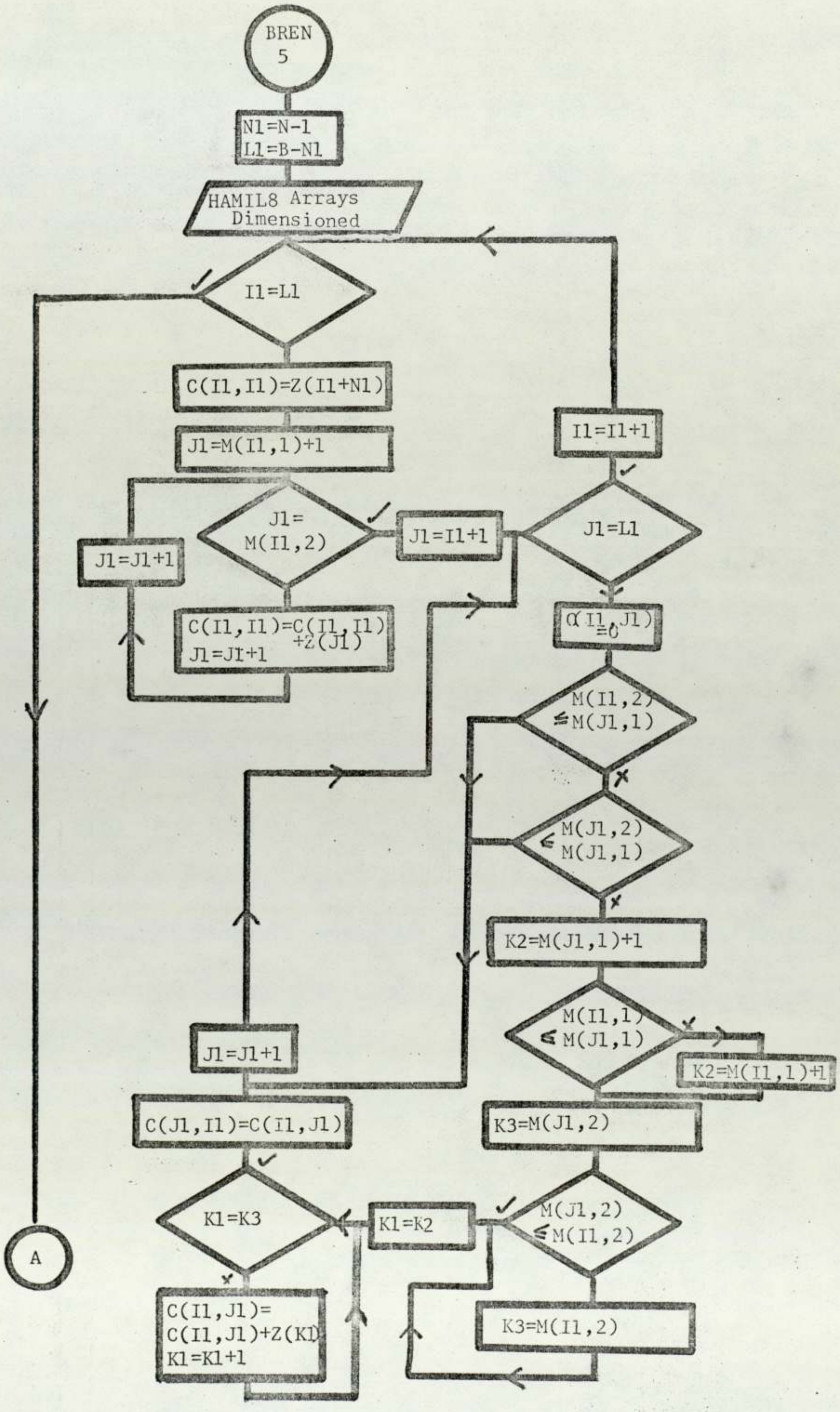
Fig. H2

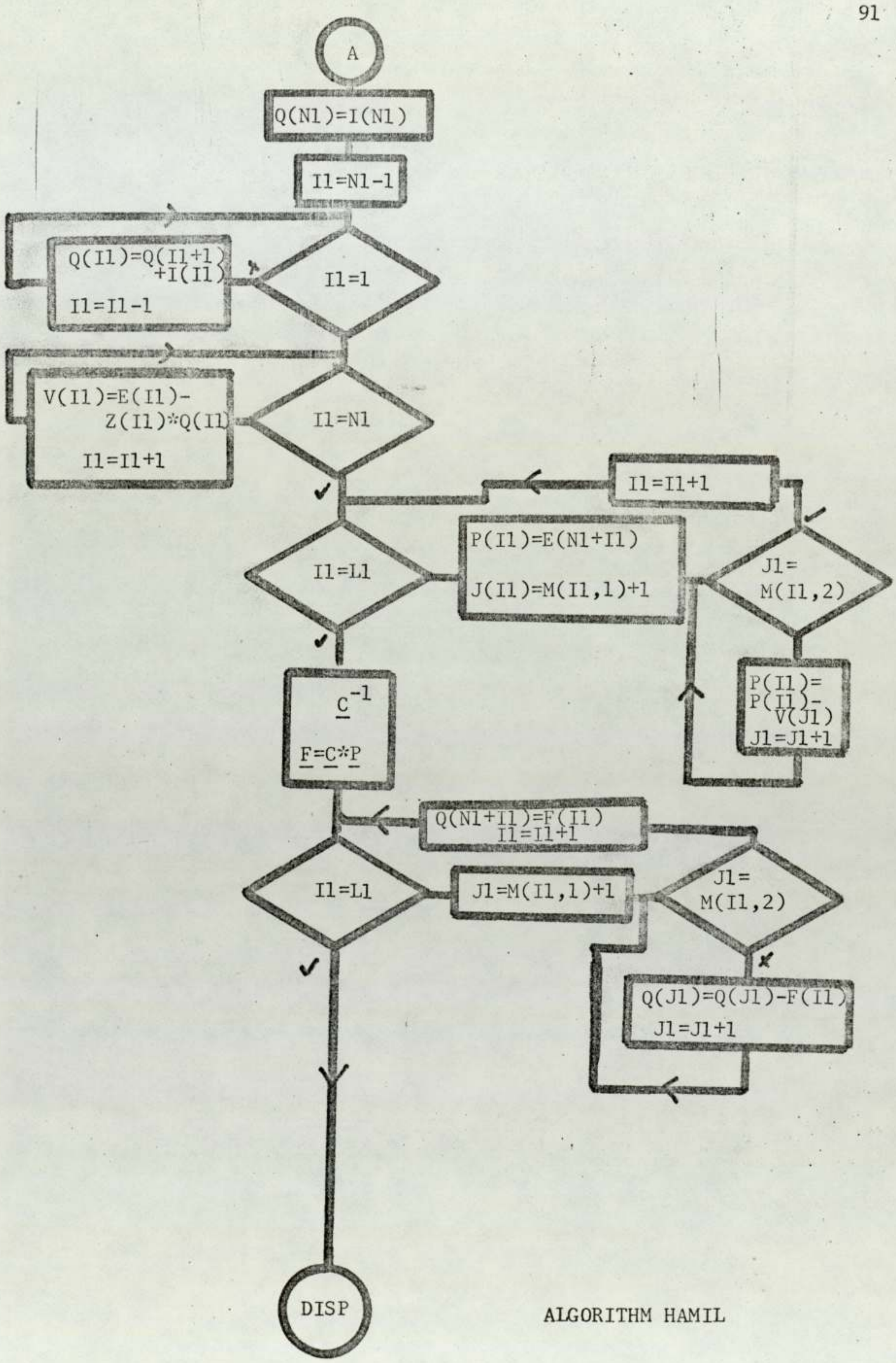
In this way the orthogonal transformation matrix can be set up directly, without reference to **C** or **A** .

Starting with the last nodal flow, this must flow along the tree branch to that node, hence the nodal flow can be assigned to the branch flow array and by stepping down the tree the other nodal flows can be added into the tree branch flow array. A temporary vector **V** of tree pressures is set up and a temporary vector **P** of the link pump pressures is also created; the former is subtracted from the latter to give net mesh pressure differences. The orthogonal transformation matrix is inverted and multiplied into the mesh pressure matrix to give the mesh circulating flows, i' , in array **F** . Working round each mesh in turn these corrections are added to the tree branches, some of them having several corrections (cf. overlap). As F contains i' the link branches in the branch flow array have already been found and are now assigned. This completes the solution for flows. In order to keep within the storage size requirements and to allow for larger meshes to be used, the output of results was handled separately in the algorithm DISP (q.v.) which is chained next.

Two versions of HAMIL were constructed, HAMIL8 applies to linear network systems and HAMIL9 to non-linear; these are different because of the comparison of flows necessary to obtain convergence.

ALGORITHM HAMIL





ALGORITHM HAMIL

HAMIL8

```
10   COM N,B,M[20,40],A[40,2],J[41],Z[40],E[40],I[40],V[50],
                                                  B[40],Q[40]
12   N1=N-1
13   L1=B-N1
17   PRINT "HAMIL8 ENTERED"
22   DIM C[4,4],P[4],F[4]
70   FOR I1=1 TO L1
75   C[I1,I1]=Z[I1+N1]
80   FOR J1=M[I1,1]+1 TO M[I1,2]
85   C[I1,I1]=C[I1,I1]+Z[J1]
90   NEXT J1
95   FOR J1=I1+1 TO L1
100   C[I1,J1]=0
105   IF M[I1,2] <= M[J1,1] OR M[J1,2] <= M[I1,1] THEN 155
110   K2=M[J1,1]+1
115   IF M[I1,1] <= M[J1,1] THEN 125
120   K2=M[I1,1]+1
125   K3=M[J1,2]
130   IF M[J1,2] <= M[I1,2] THEN 140
135   K3=M[I1,2]
140   FOR K1=K2 TO K3
145   C[I1,J1]=C[I1,J1]+Z[K1]
150   NEXT K1
155   C[J1,I1]=C[I1,J1]
160   NEXT J1
165   NEXT I1
170   Q[N1]=I[N1]
175   FOR I1=N1-1 TO 1 STEP -1
180   Q[I1]=Q[I1+1]+I[I1]
185   NEXT I1
190   FOR I1=1 TO N1
195   V[I1]=E[I1]-Z[I1]*Q[I1]
200   NEXT I1
205   FOR I1=1 TO L1
210   P[I1]=E[N1+I1]
215   FOR J1=M[I1,1]+1 TO M[I1,2]
220   P[I1]=P[I1]-V[J1]
222   NEXT J1
225   NEXT I1
226   PRINT "CTTRANSZTCTZL"
227   MAT  PRINT C
230   MAT C=INV(C)
232   PRINT "INVERSE"
233   MAT  PRINT C
235   MAT F=C*P
240   FOR I1=1 TO L1
245   FOR J1=M[I1,1]+1 TO M[I1,2]
250   Q[J1]=Q[J1]-F[I1]
255   NEXT J1
260   Q[N1+I1]=F[I1]
265   NEXT I1
267   CHAIN "DISP"
269   END
```

ALGORITHM DISP

The output requirements could not be met within the HAMIL program due to the length and storage already taken, hence a further program had to be chained for this. The calculation of the solution had taken place in the simplified notation of the Λ conjecture, the results have to be transformed from the new system back to the original branch numbers.

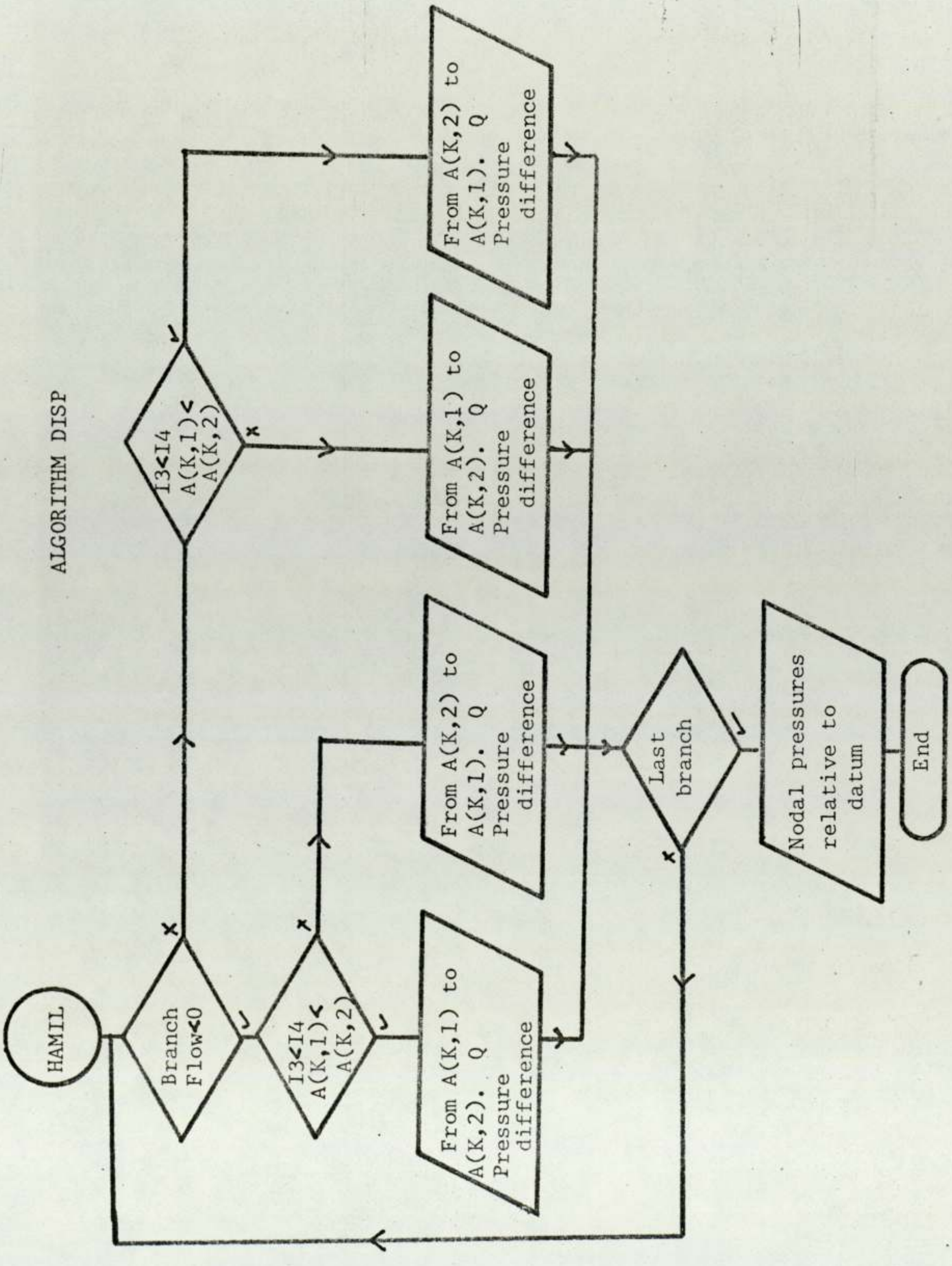
This algorithm commences by finding in natural order the corresponding old branch number, B(I). It then finds the old node numbers at the ends of this branch from the Hamiltonian path array J(I) using the current node numbering. The flow for the branch can be printed but it may have to be adjusted according to the flow convention and node numbering.

If a particular flow is negative the convention is that the flow is in the same direction as the graph direction; this was from low node number, A(K,1), to high node number, A(K,2). However the calculation was performed in the new system and the selection of the Hamiltonian path could have interchanged them as new node numbers. These are almost found in the algorithm as I3 and I4. From the J(I) array I3 and I4 are only needed relatively, the actual Λ numbering of the nodes is (I3-1) and (I4-1) when the datum is chosen as 0, but the criterion here is whether $I3 < I4$. The new nodes were numbered implicitly in the MAL algorithm.

Assuming that a particular flow Q(I1) was negative then $I3 < I4$ would mean that the flow was in the original graph direction A(K,1) to A(K,2). A print of the flow from A(K,1) to A(K,2) would be made, together with the pressure difference. When the nodes have been interchanged, $I4 < I3$, the flow is directed from A(K,2) to A(K,1) and this is printed, together with the value and the pressure difference.

The other two cases occur where the value of the flow is positive. If the nodes are in order, I3<I4, then the direction has reversed and is A(K,2) to A(K,1), the flow is printed and the pressure difference. When I4<I3 the flow direction has two reversals and so is from A(K,1) to A(K,2), the flow and pressure difference are printed. The sign of the flow has not been suppressed as this is a useful diagnostic because it relates the practical and theoretical directions.

ALGORITHM DISP


HAMIL
Branch Flow<0
I3<I4 A(K,1)< A(K,2)
From A(K,1) to A(K,2). Q Pressure difference
From A(K,2) to A(K,1). Q Pressure difference
I3<I4 A(K,1)< A(K,2)
From A(K,1) to A(K,2). Q Pressure difference
From A(K,2) to A(K,1). Q Pressure difference
Last branch
Nodal pressures relative to datum
End
x
✓

DISP

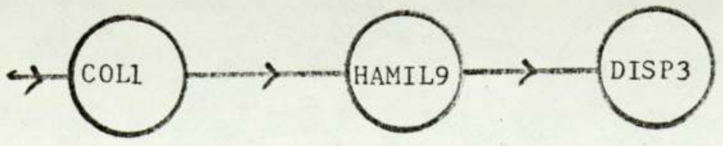
```
10   COM N,B,M[20,40],A[40,2],J[41],Z[40] E[40],I[40],
                                   V[50],B[40],Q[40]
12   PRINT "DISP ENTERED"
15   S=0
20   FOR I1=1 TO B
30   FOR K=1 TO B
40   IF B[K]=I1 THEN 60
50   NEXT K
60   FOR I3=1 TO N
70   IF J[I3]=A[K,1] THEN 90
80   NEXT I3
90   FOR I4=1 TO N
100   IF J[I4]=A[K,2] THEN 110
105   NEXT I4
110   IF Q[I1]<0 THEN 140
115   IF I3<I4 AND J[I3]<J[I4] THEN 130
120   PRINT "FLOW FROM";A[K,1];"TO"A[K,2];Q[I1];"DP";
                                   -E[I1]-Z[I1]*Q[I1]
125   GOTO 160
130   PRINT "FLOW FROM"A[K,2];"TO";A[K,1];Q[I1];"DP";
                                  -E[I1]-Z[I1]*Q[I1]
135   GOTO 160
140   IF I3<I4 THEN 150
145   PRINT "FLOW FROM";A[K,2];"TO";A[K,1];Q[I1];"DP";
                                  -E[I1]+Z[I1]*Q[I1]
147   GOTO 160
150   PRINT "FLOW FROM";A[K,1];"TO";A[K,2];Q[I1];"DP";Z[I1]
                                   *Q[I1]-E[I1]
160   NEXT I1
170   PRINT "NODAL PRESSURES RELATIVE TO DATUM NODE"J[1];
                                   "ARE:"
180   FOR I1=1 TO N-1
190   S=S+Z[I1]*Q[I1]-E[I1]
200   PRINT "NODE";J[I1+1];S
210   NEXT I1
220   END
```

ALGORITHM COL1

For the solution of networks with non-linear elements this program adjusts the initial resistances, takes in, arranges and validates the data. It also issues instructions and information, such as the precautionary note regarding HAMIL9 where certain arrays cannot be given computed dimensions.

The arrays and data held in COMMON include nodes (all of them), branches and a counter 0 for the number of loops performed to convergence. The **M** array is for the node-connected links only, the network being assumed to be already numbered according to the Hamiltonian path concept. **Q** and **J** are flow arrays, the latter being set up by COL1, the former being repeatedly found in the next program, HAMIL9. **Z** and **E** are the branch fluid resistances and branch pump pressures; the former depends upon the flow and the array is calculated in HAMIL9 and subsequently updated there. **I** is the array for the non-datum flows $\mathbf{I}'$. L , D and K are the pipe constants, length, diameter (in feet) and relative roughness.

The initial calculation assumes that all pipes will be operating in a range of ϕ at about $\phi = 0.005$ and the **S** array is constructed from this figure. The Reynolds Number for each branch is calculated from this information and the branch flows calculated from the Reynolds Numbers and entered in the **J** array. The program HAMIL9 (q.v.) is chained next.



COL1

```
10  COM N,B,O,M[4Ø,2],Q[4Ø],Z[4Ø],E[4Ø],I[4Ø],J[4Ø],L,D,K,S[4Ø],
15  PRINT "NETWORK ASSUMED NUMBERED AS PER DIGAMMA METHOD" R(40)
2Ø  PRINT "HAMIL 9 LINE 22 MUST BE ADJUSTED FOR MESH SIZE"
21  PRINT "AND VECTOR V DIMENSIONED (N-1)"
25  PRINT "ENTER NODES N, BRANCHES B"
27  LET O=-1
3Ø  INPUT N,B
4Ø  PRINT "ENTER LINKS AS LOW-HIGH NODE CONNECTIONS, ARRAY M"
54  MAT  INPUT M[B-N+1,2]
6Ø  MAT  PRINT M
7Ø  PRINT "ENTER L IN FEET, D IN FEET, RELATIVE ROUGHNESS E/D=K"
8Ø  PRINT "PROGRAM TAKES G AS 32.2 FT/S*S, PI AS 3.14159"
9Ø  INPUT L,D,K
1ØØ  PRINT "ENTER PUMP PRESSURE HEADS FOR ALL BRANCHES, ARRAY E"
1Ø3  FOR J=1 TO B
1Ø6  INPUT E[J]
1Ø8  NEXT J
12Ø  PRINT "ENTER NON-DATUM FLOWS, ARRAY I"
123  FOR L=1 TO N-1
126  INPUT I[L]
128  NEXT L
13Ø  FOR B1=1 TO B
14Ø  LET S[B1]=.ØØ5
15Ø  LET K1=.27*K
16Ø  LET A=.885
17Ø  LET X1=1/SQR(S[B1])
18Ø  R[B1]=A*X1/(EXP(-X1/2.5)-K1)
19Ø  J[B1]=4.23E-Ø6*R[B1]
2ØØ  NEXT B1
21Ø  CHAIN "HAMIL9"
215  STOP
22Ø  END
```

ALGORITHM HAMIL9

This is a revised version of the HAMIL8 algorithm for the purposes of non-linear, Colebrook-White equation solution. As the low-high convention is considered as having been adopted then the counting is made up the tree only (HAMIL8 allowed either way). The process of array creation is much the same but the fluid resistances vary with each iteration, hence they have to be calculated to give a fair approximation in the $\mathbf{Z}$ array before going into the calculation proper. The new arrays needed in the program are then $\mathbf{J}$, a flow vector for comparison and $\mathbf{S}$ a Colebrook-White vector to hold the recalculated $\phi^{-\frac{1}{2}}$ for each branch. O is an outer loop counter to check for convergence difficulties.

When the difference between all the newly calculated flows $\mathbf{Q}$ and the last calculated flows $\mathbf{J}$ is less than 10% of the latter the display algorithm DISP3 (q.v.) is chained; otherwise the flow vector $\mathbf{J}$ is updated with $\mathbf{Q}$ and the new $\phi^{-\frac{1}{2}}$ calculated for each branch and the process recycled to the calculation of the next set of fluid resistances.

HAMIL9

```
10  COM N,B,O,M[40,2],Q[40],Z[40],E[40],I[40],J[40],L,D,K,S[40],
                                                          R(40)
12  N1=N-1
13  L1=B-N1
22  DIM C[4,4],P[4],F[4],V[11]
25  FOR B2=1 TO B
30  Z[B2]=S[B2]*L*64*ABS(J[B2]/32.2*D↑5*3.14159↑2)
40  NEXT B2
70  FOR I1=1 TO L1
75  C[I1,I1]=Z[I1+N1]
80  FOR J1=M[I1,1]+1 TO M[I1,2]
85  C[I1,I1]=C[I1,I1]+Z[J1]
90  NEXT J1
95  FOR J1=I1+1 TO L1
100  C[I1,J1]=0
105  IF M[I1,2] <= M[J1,1] THEN 155
110  K2=M[J1,1]+1
125  K3=M[J1,2]
130  IF M[J1,2] <= M[I1,2] THEN 140
135  K3=M[I1,2]
140  FOR K1=K2 TO K3
145  C[I1,J1]=C[I1,J1]+Z[K1]
150  NEXT K1
155  C[J1,I1]=C[I1,J1]
160  NEXT J1
165  NEXT I1
170  Q[N1]=I[N1]
175  FOR I1=N1-1 TO 1 STEP -1
180  Q[I1]=Q[I1+1]+I[I1]
185  NEXT I1
190  FOR I1=1 TO N1
195  V[I1]=E[I1]-Z[I1]*Q[I1]
200  NEXT I1
205  FOR I1=1 TO L1
210  P[I1]=E[N1+I1]
215  FOR J1=M[I1,1]+1 TO M[I1,2]
220  P[I1]=P[I1]-V[J1]
222  NEXT J1
225  NEXT I1
230  MAT C=INV(C)
235  MAT F=C*P
240  FOR I1=1 TO L1
245  FOR J1=M[I1,1]+1 TO M[I1,2]
250  Q[J1]=Q[J1]-F[I1]
255  NEXT J1
260  Q[N1+I1]=F[I1]
265  NEXT I1
266  O=O+1
267  PRINT "OUTER LOOP TRAVERSED";O;"TIMES"
270  FOR P=1 TO B
280  IF ABS(Q[P]-J[P]) >= ABS(.1*Q[P]) THEN 310
290  NEXT P
300  CHAIN "DISP3"
310  FOR B3=1 TO B
315  J[B3]=(J[B3]+Q[B3])/2
317  K1=.27*K
318  A=.885
319  R[B3]=ABS(J[B3]/4.23E-06)
320  S[B3]=-2.5*LOG(K1+A/(R[B3]*SQR(S[B3])))
330  NEXT B3
340  GOTO 25
350  END
```

ALGORITHM DISP3

This algorithm simply prints out the vector of branch flows, followed by the nodal pressures up the tree. Because of the nature of the fluid resistances these were also printed to give a record and so that the final resistances could be used as a check if required.

DISP3

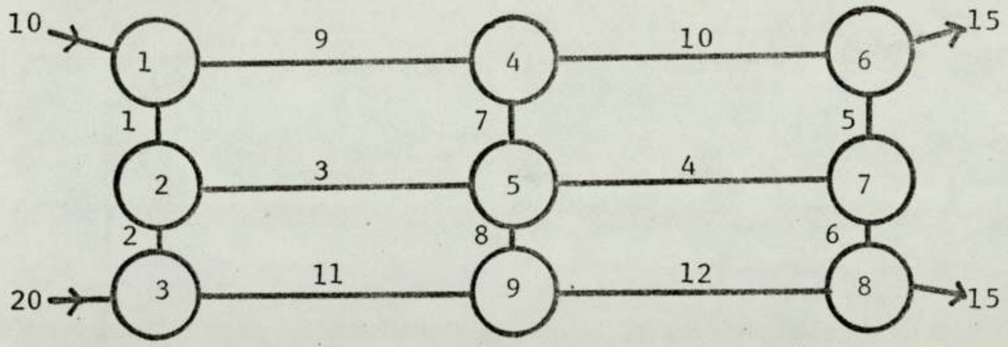
```
10  COM N,B,O,M[40,2],Q[40],Z[40],E[40],I[40],J[40],L,D,K,S[40],
                                                         R(40)
20  PRINT "BRANCH FLOWS ARE:"
30  FOR B6=1 TO B
31  IF R[B6]<2000 THEN 35
32  IF R[B6]<4000 THEN 37
33  PRINT Q[B6];"TURBULENT"
34  GOTO 40
35  PRINT Q[B6];"LAMINAR"
36  GOTO 40
37  PRINT Q[B6];"TRANSITIONAL"
40  NEXT B6
50  PRINT "NODAL PRESSURES ARE:"
60  W=0
70  FOR B5=1 TO N-1
80  W=W+Z[B5]*Q[B5]-E[B5]
90  PRINT W;Z[B5]
91  NEXT B5
110  STOP
120  END
```

EXAMPLES

Treatment of the Gay and Preece (6.8) network by the ρ method.

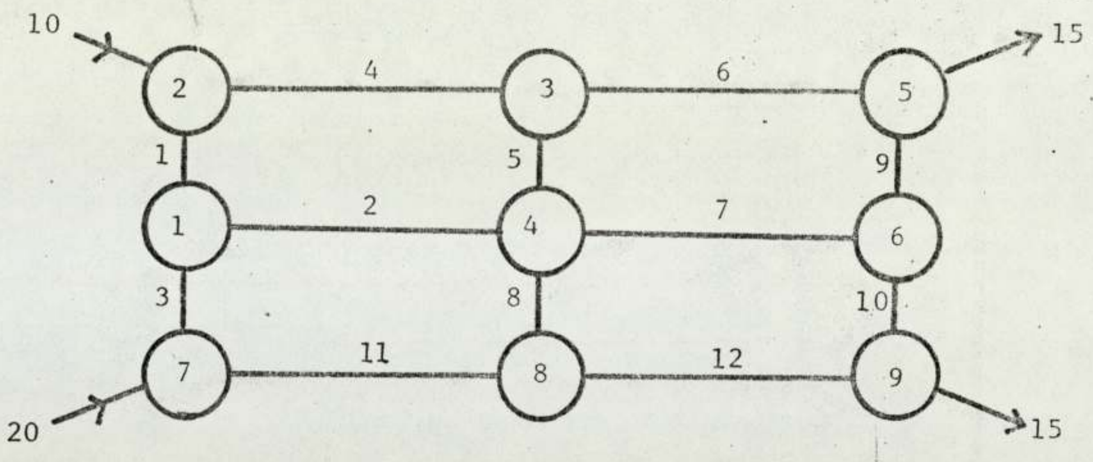
Their numbering of nodes and branches is taken to enable convenient comparison. The branch directions cannot be transferred because the ρ method assumes the convention of low node number to high node number as the defining direction for the input of **A**, the node-connected branch matrix.

Recapitulating the network for use in this context:



The algorithm ATOM takes the data for sequential branches as node connections e.g. branch 9 is entered as 1,4 for the ninth pair. An immediate print is made for data validation. The number of nodes and branches is common to all the chained programs. This algorithm produces the **M** matrix of Roberts and Flores (6.3) which is printed for checking purposes.

The MAL algorithm is chained next and scans the **M** array for one Hamiltonian path. The first path found is printed together with the corresponding ρ path numbering; visual checking is easily performed. This network was given alternative numberings to test the searching powers of the algorithm, for example the pattern:



has no Hamiltonian path starting at 1, thereby causing the process to step back, reach this decision then step on to a node 2 start.

The program next chained is BREN which takes in the data of resistance and pump pressure per branch with the original numbering. The nodal flows are next entered, differing from Gay and Preece non-datum flows entry because the datum in the Λ system has been found (0 - 1) but not yet assigned. The branches are all renumbered by the algorithm, which then chains HAMIL.

The HAMIL algorithm solves the system which is now in the Λ numbering. The array $(\tilde{C}_t Z_t C_t + Z_\ell)$ is printed next for validation purposes, inverted and the inverse also recorded. The program is too lengthy to contain detailed output orders and so these are chained in the DISP algorithm.

The DISP algorithm relates the solution in the new numbering to the original numbering and prints it. Because the branch flow directions can change, the possible negative sign is avoided and the end nodes interchanged for this. The branches are printed sequentially in Λ order e.g. the third line is Λ branch 3 which corresponds with original branch 11; the Λ tree branches are printed first followed by the link branches. The latter are sequential so that the ninth row is the first link, which defines mesh 1 , etc. Both branch flows and directed pressure drops are printed. This is followed by a summation of branch pressures up the tree. The datum node number is printed because it is not necessarily 1.

It is observed that neither $\mathbf{C}$ nor $\mathbf{B}_t$ are entered and that $\mathbf{A}_\ell$, a sparse matrix, has been entered as a node-connection list.

An alternative numbering of the same Example is included to allow for a start from another node. The numerical results are essentially the same as those published (with the exception of the mis-print for their branch 12, which should be a pressure rise of -78.1).

A network of Gay and Middleton (2.16) with pumps in two meshes is next solved linearly. This is illustrative of the $\nearrow$ method, showing how the nodes are renumbered and the links are taken into account. The numerical results when rounded agree with those published.

A 4-mesh network of pipes solved non-linearly is included next to investigate convergence and handling convenience. The problem is assumed already sorted about a Hamiltonian path, which is different from the two previous examples of 4-mesh networks. The compactness of the data input is shown, i.e. only 4 links instead of 12 branches. Only 4 iterations were required for a 10% accurate solution.

The Gay and Middleton (2.16) test example is next shown solved non-linearly from the $\nearrow$ numbering illustrated. The compactness of the results is evident. The accuracy constant was taken as 0.1% (cf. p.48) and both KCL and KVL check to better than 0.000. These results had to be Xeroxed due to a breakdown of the Friden Flexowriter.

Gay and Preece (6.8) network, linear

```
RUN
ATOM21

INPUT N,B. MAT A IS READ NEXT
INPUT N=NODES, B=BRANCHES. N<=41, B<=80
?9,12
SPARSE A MATRIX, LOW NODE TO HIGH NODE
?1,2,2,3,2,5,5,7,6,7,7,8,4,5,5,9,1,4,4,6,3,9,8,9
 1             2

 2             3

 2             5

 5             7

 6             7

 7             8

 4             5

 5             9

 1             4

 4             6

 3             9

 8             9

 2    1    2    1    2    4    5    7    3
 4    3    9    5    4    7    6    9    5
 0    5    0    6    7    0    8    0    8
 0    0    0    0    9    0    0    0    0
 0    0    0    0    0    0    0    0    0
MAL41 ENTERED
H-PATH  1    2    3    9    5    4    6    7    8
DIGAMMA 0    1    2    3    4    5    6    7    8
BREN5 ENTERED
ENTER U (OLD Z), S (OLD E), ALTERNATELY
?4,0
?5,0
?6,0
?6,0
?5,0
?8,0
?5,0
?7,0
?4,0
?4,0
?6,0
?7,0
ENTER ALL NODAL FLOWS
?10
?0
?20
?0
```

```
?0
?-15
?0
?-15
?0
HAMIL8 ENTERED
CTTRANSZTCTZL
 24              0               18              7

 0               20              5               14

 18              5               31              12

 7               14              12              36

INVERSE
 7.71358E-02     1.54555E-02     -4.49486E-02    -6.02623E-03

 1.54555E-02     7.18185E-02     -9.85466E-03    -2.76498E-02

-4.49486E-02    -9.85466E-03      .06324         -8.50762E-03

-6.02623E-03    -2.76498E-02     -8.50762E-03     4.25381E-02

DISP ENTERED
FLOW FROM 2     TO 1       .921299     DELTAP-3.6852
FLOW FROM 3     TO 2      7.80787      DELTAP-39.0394
FLOW FROM 3     TO 9      12.1921      DELTAP-73.1528
FLOW FROM 9     TO 5      1.02942      DELTAP-7.20591
FLOW FROM 5     TO 4      1.21021      DELTAP-6.05104
FLOW FROM 4     TO 6      12.1315      DELTAP-48.526
FLOW FROM 7     TO 6      2.86849      DELTAP-14.3425
FLOW FROM 7     TO 8      3.83729      DELTAP-30.6983
FLOW FROM 2     TO 5      6.88657      DELTAP-41.3194
FLOW FROM 5     TO 7      6.70578      DELTAP-40.2347
FLOW FROM 1     TO 4      10.9213      DELTAP-43.6852
FLOW FROM 9     TO 8      11.1627      DELTAP-78.139
NODAL PRESSURES RELATIVE TO DATUM NODE 1     ARE:
NODE 2      3.6852
NODE 3      42.7246
NODE 9     -30.4282
NODE 5     -37.6341
NODE 4     -43.6852
NODE 6     -92.2112
NODE 7     -77.8687
NODE 8     -108.567

DONE
```

Gay and Preece (6.8) network with alternative numbering. Linear.

```
ATOM21

INPUT N,B. MAT A IS READ NEXT
INPUT N=NODES, B=BRANCHES. N<=41, B<=80
?9,12
SPARSE A MATRIX, LOW NODE TO HIGH NODE
?1,2,1,4,1,7,2,3,3,4,3,5,4,6,4,8,5,6,6,9,7,8,8,9
 1            2

 1            4

 1            7

 2            3

 3            4

 3            5

 4            6

 4            8

 5            6

 6            9

 7            8

 8            9

 2     1     2     1     3     4     1     4     6
 4     3     4     3     6     5     8     7     8
 7     0     5     6     0     9     0     9     0
 0     0     0     8     0     0     0     0     0
 0     0     0     0     0     0     0     0     0
MAL41 ENTERED
 1    STARTING NODE, NO PATH; CHECKED BACK TO FIRST NODE
H-PATH  2     1     4     3     5     6     9     8     7
DIGAMMA 0     1     2     3     4     5     6     7     8
BREN5 ENTERED
ENTER U (OLD Z), S (OLD E), ALTERNATELY
?4,0
?6,0
?5,0
?4,0
?5,0
?4,0
?6,0
?7,0
?5,0
?8,0
?6,0
?7,0
ENTER ALL NODAL FLOWS
```

```
?0
?10
?0
?0
?-15
?0
?20
?0
?-15
HAMIL8 ENTERED
CTTRANSZTCTZL
 46             11             14             29

 11             19             5              5

 14             5              20             14

 29             5              14             36

INVERSE
 5.04786E-02   -1.82914E-02   -5.60084E-03   -3.59447E-02

-1.82914E-02    .06324        -9.85466E-03    9.78376E-03

-5.60085E-03   -9.85467E-03    7.18185E-02   -2.20489E-02

-3.59447E-02    9.78377E-03   -2.20489E-02    .063949

DISP ENTERED
FLOW FROM 1    TO 2     .921305    DP-3.68522
FLOW FROM 1    TO 4    -6.88656    DP-41.3194
FLOW FROM 4    TO 3    -1.21021    DP-6.05104
FLOW FROM 3    TO 5    -12.1315    DP-48.5261
FLOW FROM 6    TO 5     2.86849    DP-14.3424
FLOW FROM 6    TO 9    -3.83729    DP-30.6983
FLOW FROM 8    TO 9     11.1627    DP-78.139
FLOW FROM 7    TO 8     12.1921    DP-73.1528
FLOW FROM 7    TO 1     7.80787    DP-39.0394
FLOW FROM 2    TO 3    -10.9213    DP-43.6852
FLOW FROM 4    TO 6    -6.70578    DP-40.2347
FLOW FROM 8    TO 4     1.02942    DP-7.20597
NODAL PRESSURES RELATIVE TO DATUM NODE 2    ARE:
NODE 1     3.68522
NODE 4    -37.6342
NODE 3    -43.6852
NODE 5    -92.2113
NODE 6    -77.8689
NODE 9    -108.567
NODE 8    -30.4283
NODE 7     42.7245

DONE
```

Gay and Middleton (6.6) network. Linear.

```
ATOM21

INPUT N,B. MAT A IS READ NEXT
INPUT N=NODES, B=BRANCHES. N<=41, B<=80
?4,5
SPARSE A MATRIX, LOW NODE TO HIGH NODE
?1,2,1,3,1,4,2,4,3,4
 1              2

 1              3

 1              4

 2              4

 3              4

 2    1    1    1
 3    4    4    2
 4    0    0    3
 0    0    0    0
MAL41 ENTERED
H-PATH  1      2      4      3
DIGAMMA 0      1      2      3
BREN5 ENTERED
ENTER U (OLD Z), S (OLD E), ALTERNATELY
?4,0
?1,5
?5,-4
?3,0
?2,2
ENTER ALL NODAL FLOWS
?-2
?4
?2
?-4
HAMIL8 ENTERED
CTTRANSZTCTZL
 10             7

 7              12

INVERSE
 .169014        -9.85915E-02

-9.85915E-02     .140845

DISP ENTERED
FLOW FROM 2     TO 1      1.16901     DP-4.67606
FLOW FROM 2     TO 4     -2.83099     DP-8.49296
FLOW FROM 4     TO 3     -.394366     DP 1.21127
FLOW FROM 3     TO 1      2.39437     DP-7.39437
FLOW FROM 1     TO 4     -1.56338     DP-3.8169
NODAL PRESSURES RELATIVE TO DATUM NODE 1      ARE:
NODE 2      4.67606
NODE 4     -3.8169
NODE 3     -2.60563

DONE
```

COL1

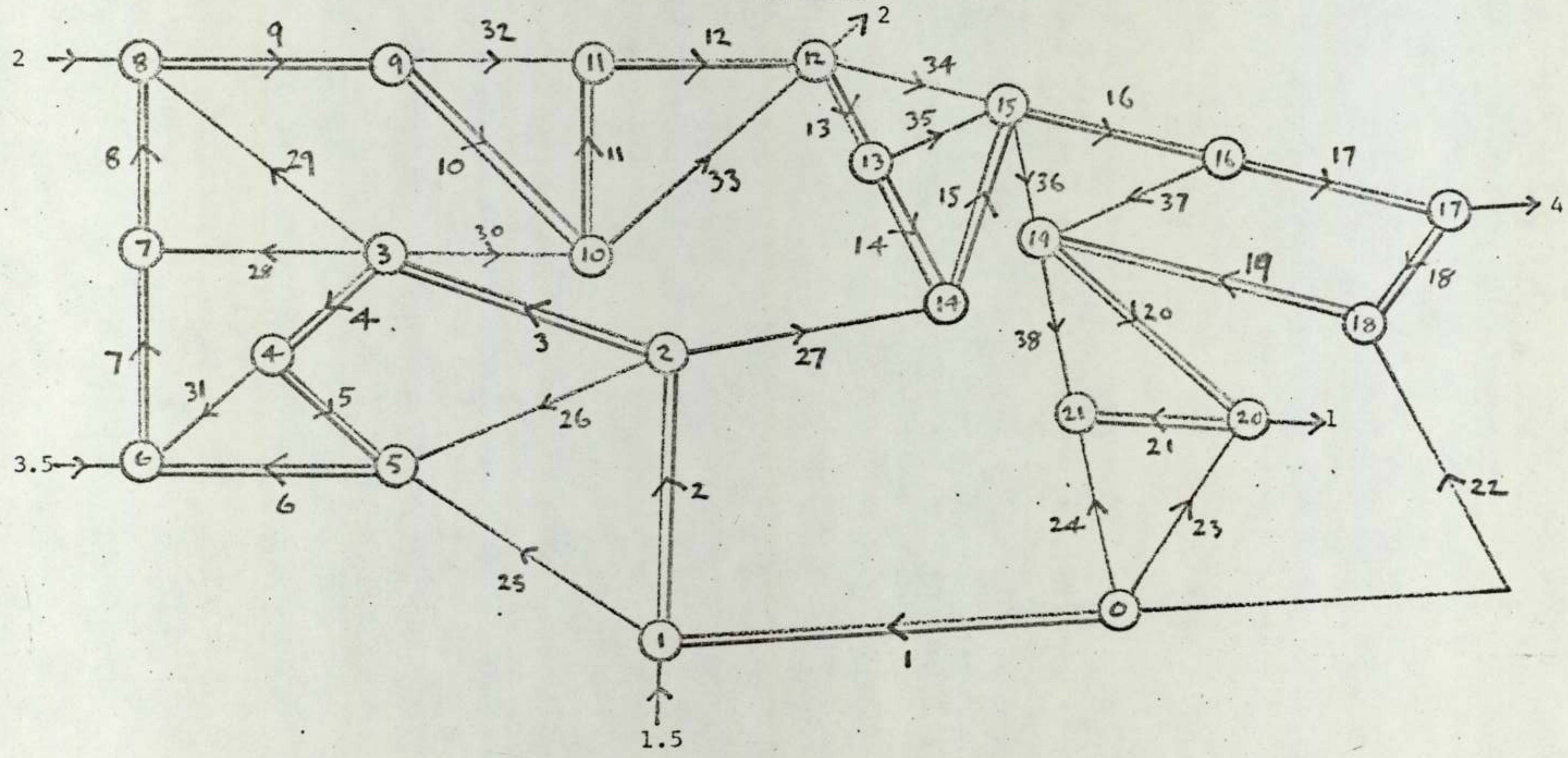
A 4-mesh network of pipes. Non-linear.

```
NETWORK ASSUMED NUMBERED AS PER DIGAMMA METHOD
HAMIL 9 LINE 22 MUST BE ADJUSTED FOR MESH SIZE
AND VECTOR V DIMENSIONED (N-1)
ENTER NODES N, BRANCHES B
?9, 12
ENTER LINKS AS LOW-HIGH NODE CONNECTIONS, ARRAY M
?Ø, 3, Ø, 5, Ø, 7, 1, 8
 Ø              3

 Ø              5

 Ø              7

 1              8

ENTER L IN FEET, D IN FEET, RELATIVE ROUGHNESS E/D=K
PROGRAM TAKES G AS 32.2 FT/S*S, PI AS 3.14159
?1ØØ, .5, .ØØØ3
ENTER PUMP PRESSURE HEADS FOR ALL BRANCHES, ARRAY E
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
ENTER NON-DATUM FLOWS, ARRAY I
?Ø
?1
?Ø
?2
?Ø
?-1.5
?Ø
?-1.5
```

```
OUTER LOOP TRAVERSED Ø     TIMES
OUTER LOOP TRAVERSED 1     TIMES
OUTER LOOP TRAVERSED 2     TIMES
OUTER LOOP TRAVERSED 3     TIMES
OUTER LOOP TRAVERSED 4     TIMES
BRANCH FLOWS ARE:
-.121933     TURBULENT
 .9Ø9Ø2      TURBULENT
-9.Ø98Ø5E-Ø2   TURBULENT
 .8Ø4626     TURBULENT
-1.19537     TURBULENT
-1.Ø4386     TURBULENT
 .456136     TURBULENT
-.469Ø48     TURBULENT
 .8956Ø6     TURBULENT
 .151511     TURBULENT
-.925184     TURBULENT
-1.Ø3Ø95     TURBULENT
NODAL PRESSURES ARE:
-1.78364      14.6281
 98.7758      11Ø.624
 97.78Ø2      1Ø.9424
 176.621      97.9841
 2.81427      145.399
-129.575      126.826
-1Ø4.192      55.6468
-13Ø.913      56.9679

DONE
```

The Gay and Middleton (2.16) Test Example - Hamiltonian path and Λ numbering.

Non-linear.



```
GET-COL1
RUN
COL1

NETWORK ASSUMED NUMBERED AS PER DIGAMMA METHOD
HAMIL 9 LINE 22 MUST BE ADJUSTED FOR MESH SIZE
AND VECTOR V DIMENSIONED (N-1)
ENTER NODES N, BRANCHES B
?22,38
ENTER LINKS AS LOW-HIGH NODE CONNECTIONS, ARRAY M
?Ø,18,Ø,2Ø,Ø,21,1,5,2,5,2,14,3,7,3,8,3,1Ø,4,6
??9,11,1Ø,12,12,15,13,15,15,19,16,19,19,21
 Ø            18

 Ø            2Ø

 Ø            21

 1            5

 2            5

 2            14

 3            7

 3            8

 3.           1Ø

 4            6

 9            11

 1Ø           12

 12           15

 13           15

 15           19

 16           19

 19           21

ENTER L IN FEET, D IN FEET, RELATIVE ROUGHNESS E/D=K
PROGRAM TAKES G AS 32.2 FT/S*S, PI AS 3.14159
?1ØØ,.5,Ø.Ø
ENTER PUMP PRESSURE HEADS FOR ALL BRANCHES, ARRAY E
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
```

```
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
?Ø
ENTER NON-DATUM FLOWS, ARRAY I
?1.5
?Ø
?Ø
?Ø
?Ø
?3.5
?Ø
?2
?Ø
?Ø
?Ø
?-2
?Ø
?Ø
?Ø
?Ø
?-4
?Ø
?Ø
?-1
?Ø
```

```
OUTER LOOP TRAVERSED 0     TIMES
OUTER LOOP TRAVERSED 1     TIMES
OUTER LOOP TRAVERSED 2     TIMES
OUTER LOOP TRAVERSED 3     TIMES
OUTER LOOP TRAVERSED 4     TIMES
OUTER LOOP TRAVERSED 5     TIMES
OUTER LOOP TRAVERSED 6     TIMES
OUTER LOOP TRAVERSED 7     TIMES
OUTER LOOP TRAVERSED 8     TIMES
OUTER LOOP TRAVERSED 9     TIMES
OUTER LOOP TRAVERSED 10    TIMES
OUTER LOOP TRAVERSED 11   TIMES
BRANCH FLOWS AND FINAL RESISTANCES ARE:
 2.31365      TURBULENT 1181.29
-7.72835E-02    TURBULENT 38.2724
 .805455      TURBULENT 381.582
 .64949       TURBULENT 302.808
-.517612      TURBULENT 237.23
 1.26691      TURBULENT 620.24
-1.06599      TURBULENT 515.47
-.257042      TURBULENT 111.511
-1.48506      TURBULENT 735.294
-.64184       TURBULENT 298.993
-.562748      TURBULENT 259.561
-1.40597      TURBULENT 693.449
-.096704      TURBULENT 38.6366
 .711931      TURBULENT 334.206
-1.06439      TURBULENT 514.641
-1.41115      TURBULENT 696.191
-2.04336      TURBULENT 1034.47
 1.95664      TURBULENT 987.577
 .878003      TURBULENT 418.597
-5.55062E-02    TURBULENT 21.0997
 .294944      TURBULENT 129.352
-1.07864      TURBULENT 522.012
-.64955       TURBULENT 302.841
-.585466      TURBULENT 270.825
 .890937      TURBULENT 425.233
 .893585      TURBULENT 426.591
-1.77632      TURBULENT 890.589
 .808945      TURBULENT 383.373
 .771984      TURBULENT 364.595
-1.42496      TURBULENT 703.476
 1.1671       TURBULENT 568.047
-.843219      TURBULENT 400.832
-1.50406      TURBULENT 745.357
-.813319      TURBULENT 385.595
-.808635      TURBULENT 383,209
-1.27519      TURBULENT 624.595
 .632206      TURBULENT 294.169
 .290522      TURBULENT 127.278
NODAL PRESSURES ARE:
 2733.1
 2730.76
 3038.11
 3234.78
 3111.98
 3897.77
 3348.29
 3319.62
 2227.67
 2035.76
 1889.7
 914.73
 910.994
 1148.93
 601.146
-381.287
-2495.08
-562.744
-195.215
-196.386
-158.235

DONE
```

BIBLIOGRAPHY AND REFERENCES

1.1 Gibbon, E. 'The Decline and Fall of the Roman Empire', Chapter 2.

1.2 Ibid., Chapters 31, 39.

1.3 Kirchhoff, G. 'On the solution of the equations obtained from the investigation of the linear distribution of galvanic currents' 1958, Institution of Radio Engineers Transactions on Circuit Theory Vol. CT-5 pp. 4-7 (translation by J.B. O'Toole).

1.4 Kirchhoff, G. 'Über die Auflösung der Gleichungen, auf welche man bei der Untersuchungen der Linearen Verteilung Galvanisher Ströme geführt wird' 1847, Poggendorf Ann. Physik 72 pp. 497-508.

1.5 Atkinson, J.J. 'On the Theory of the Ventilation of Mines' 1854-5, Transactions of the North of England Mining Engineers 3 p. 318.

1.6 Smith, D.E. 'History of Mathematics' 1953, Vol. II pp. 437-442 Dover.

1.7 Taylor, S.N. 'The Solution of Ventilation Network Problems' 1954, B.Sc. (Mining Engineering) Honours Thesis, University of Nottingham.

1.8 Taylor, S.N. 'Solution of Two-Mesh Mine Ventilation Networks by Computer' 1966, Colliery Guardian, January 14 pp. 59-63.
'The Solution of Ventilation Network Problems by Automatic Electronic Digital Computer' 1961, M.Sc. Thesis, University of Nottingham.

1.9 Cross, H. 'Analysis of Flow in Networks of Conduits or Conductors' 1936, Engineering Experimental Station, University of Illinois Bulletin 286 Vol. XXX Part 1, pp. 7-29.

1.10 Prandtl, L. 'Neuere Ergebnisse der Turbulenz-forschung' 1933, Zeitschrift des Vereins deutscher Ingenieure 77.

1.11 Jeppson, R.W. 'Analysis of Flow in Pipe Networks' 1976, Ann Arbor Science, p. 40.

1.12 Hay, D. 'The Theory of Ventilation; A review of its Present Treatment (First Report of the Midland Institute Committee on the Ventilation of Mines)' 1924, Transactions of the Institution of

Mining Engineers Vol. LXVII pp. 268-273.

1.13 Howland, W.E. 'Expansion of the Freeman Method for the Solution of Pipe Flow Problems' 1934, Journal of the New England Water Works Association 48 p.408.

2.1 Maas, W. 'An Electrical Analogue for Mine Ventilation' 1951, Colliery Engineering 28 p.24.

2.2 Scott, D.R. 'The Computation of Mine Ventilation Networks' 1951, M.Sc. Thesis, University of Nottingham.

2.3 McIlroy, M.S. 'Nonlinear Electrical Analogy for Pipe Networks' 1952, Proceedings of the American Society of Civil Engineers Vol.78, July.

2.4 Scott, D.R., Hinsley, F.B., Hudson, R.F. 'A Calculator for the Solution of Ventilation Network Problems' 1953, Transactions of the Institution of Mining Engineers Vol. 112 pp. 623-637.

2.5 Hiramatsu, Y. 'Ermittlung der Stärke von Wetterströmen in Grubenwetternetzen nach Formeln für elektrischen Strom' 1953, Gluckauf Vol. 89, No. 15/16 11 April p. 355.

2.6 Scott, D.R., Hudson, R.F. 'An Automatic Analogue Computer for the Solution of Mine Ventilation Networks' 1952, Journal of Scientific Instruments Vol.30, No.6 June, pp. 185-188.

2.7 Personal invitation to view, 1977, at Newbury, Berkshire, 11th May.

2.8 Williams, R.W. 'Recent Developments in the Analogue Techniques for Mine Ventilation Analysis' 1978, Mining Technology Vol. 60 June pp. 217-219.

2.9 Branin, F.H., Jr. 'Machine Analysis of Networks' 1961, IBM Report TN 00.490 Poughkeepsie, New York pp. 1-48.

2.10 Kron, G. 'Tensor Analysis of Networks' 1939, John Wiley & Sons, New York.

2.11 Ingels, D.M., Powers, J.E. 'Analysis of Pipe Networks' 1964, Chemical Engineering Progress Vol. 60, No. 2 February pp. 65-70.

2.12 Doland, J.J. 'Simplified Analysis of Flow in Water Distribution

Systems 1936, Engineering News-Record October 1st pp. 475-477.

2.13 Daniel, P.T. 'The Analysis of Compressible and Incompressible Fluid Networks' 1966, Transactions of the Institution of Chemical Engineers Vol.44 pp. T77-T84.

2.14 Scott, D.R. in conversation, nescio.

2.15 Scott, D.R., Hinsley, F.B. 'Ventilation Network Theory - Part 3' 1951 Colliery Engineering Vol. 28 June p. 129.

2.16 Gay, B., Middleton, P. 'The Solution of Pipe Network Problems' 1971, Chemical Engineering Science Vol. 26 pp. 109-123.

2.17 Bending, M.J., Hutchison, H.P. 'The Calculation of Steady State Incompressible Flow in Large Networks of Pipes' 1973, Chemical Engineering Science Vol. 28 pp. 1857-1864.

2.18 Mah, R.S.H. 'Pipeline Network Calculations Using Sparse Computation Techniques' 1974, Chemical Engineering Science Vol. 29 pp. 1629-1638.

2.19 Gay, B., Preece, P.E. 'Matrix Method for the Solution of Fluid Network Problems; Part I - Mesh Methods' 1975, Transactions of the Institution of Chemical Engineers Vol. 53 pp. 12-15.

3.1 Personal invitation, Membrain Limited, Wimborne, Dorset. Nescio.

3.2 Happ, H.H. 'Diakoptics and Networks' 1971, Academic Press, New York.

3.3 Mullineux, N., Reed, J.R. 'Diakoptics in Node-to-Datum Analysis' 1974 Proceedings of the Institution of Electrical Engineers, Vol. 121 No.7 July, pp.639-644.

3.4 Hamam, Y.M., Brameller, A. 'A Hybrid Method for the Solution of Piping Networks' 1971, Proceedings of the Institution of Electrical Engineers Vol. 118 No. 11 November, pp. 1607-1612.

4.1 Preece, P.E., Gay, B., Ti, H.C. 'Matrix Methods for the Solution of Fluid Network Problems: A Note Dealing with Mixed Pressure and Flow' 1977 Fourth Annual Research Meeting of the Institution of Chemical Engineers, Swansea.

4.2 Mah, R.S.H. 'Pipeline Network Calculations Using Sparse Computing

Techniques' 1974, Chemical Engineering Science Vol. 29, pp.1629-1638.

4.3 Whitney, H. 'Non-Separable and Planar Graphs' 1932, Transactions of the American Mathematical Society, pp. 339-362.

4.4 Dirac, G.A. 'Some Theorems on Abstract Graphs', 1952, Proceedings of the London Mathematical Society 2, pp. 69-81.

4.5 Pósa, L. 'Hamiltonian Circuits in Random Graphs' 1976, Discrete Mathematics 14, pp.359-364.

4.6 Berge, C. 'The Theory of Graphs and its Applications' 1962, Methuen pp. 110-111.

4.7 Cummins, R.L. 'Hamilton Circuits in Tree Graphs' 1966, Institute of Electrical and Electronic Engineering Transactions on Control Theory CT-13 No.1, March pp.82-90.

4.8 Kamae, T. 'The Existence of a Hamilton Circuit in a Tree Graph' 1967 Institute of Electrical and Electronic Engineering Transactions CT-14 No. 3, September pp. 279-283.

4.9 Kishi, G., Kajitani, Y. 'On Hamilton Circuits in Tree Graphs' 1968 Institute of Electrical and Electronic Engineering Transactions on Circuit Theory Vol. CT-15 No. 1, March pp. 42-50.

4.10 Chen, W.K. 'Applied Graph Theory' 1975, North-Holland.

4.11 Wang, D.L., Kleitman, D.J. 'On the Existence of n-Connected Graphs with Prescribed Degrees $(n \geq 2)$' 1973 Networks 3 pp. 252-239.

4.12 Gehner, K.R. 'A Necessary Condition for the Existence of a Circuit of any Specified Length' 1976 Networks 6 No. 2 pp.131-138.

4.13 Andrásfai, B. 'Introductory Graph Theory' 1977, Adam Hilger, Bristol.

4.14 Lesniak-Foster, L. 'Some Recent Results in Hamiltonian Graphs' 1977 Journal of Graph Theory, Vol. 1 pp. 27-36.

4.15 Deo, N. 'Graph Theory with Applications to Engineering and Computer Science' 1974, Prentice-Hall.

5.1 Happ, H.H. 'Diakoptics and Networks', 1971, Academic Press, New York.

5.2 Branin, F.H. 'Computer Methods in Network Analysis' 1967, IBM Technical Report No. TR00.1562, 9 January (Poughkeepsie, N.Y.:IBM).

6.1 Preece, P.E. 'The Analysis of Flow in Pipe Network Systems' 1972, Ph.D. Thesis, University of Aston in Birmingham.

6.2 Mullineux, N., Reed, J.R. 'The Trunk in the Topology of Electrical Networks' 1966, Journal of the Franklin Institute, Vol. 281 No. 3.

6.3 Roberts, S.M., Flores, B. 'Systematic Generation of Hamiltonian Circuits' 1966, Communications of the ACM, Vol. 9 No. 9 (Business Applications) pp. 690-694.

6.4 Christofides, N. 'Graph Theory, An Algorithmic Approach' 1975, Academic Press.

6.5 Daniel, P.T. 'The Analysis of Compressible and Incompressible Fluid Networks' 1966, Transactions of the Institution of Chemical Engineers Vol. 44 pp. T77-T81.

6.6 Gay, B., Middleton, P. 'The Solution of Pipe Network Problems' 1971, Chemical Engineering Science Vol. 26, pp.109-123.

6.7 Percival, W.S. 'The Graphs of Active Networks' 1955, Proceedings of the Institution of Electrical Engineers Vol. 102, April pp.270-278.

6.8 Gay, B., Preece, P.E. 'Matrix Methods for the Solution of Fluid Network Problems: Part 1 - Mesh Methods' 1975, Transactions of the Institution of Chemical Engineers, Vol. 53 pp. 12-15.

7.1 Coulson, J.M., Richardson, J.F. 'Chemical Engineering' 1970, Vol. 1, Pergamon.

9.1 Brameller, A., Allan, R.N., Hamam, Y.M. 'Sparsity' 1976, Pitman.

9.2 Caffrey, J. 'ALGORITHM 66, INVRS' 1961, Communications of the ACM, 4 p. 322.

10.1 Wilkinson, J.H. Private communication, 27.1.76.

10.2 Wilkinson, J.H. 'The Algebraic Eigenvalue Problem' 1965, Oxford University Press, p. 443.

E1 Middleton, P. 'Solution of the Complex Pipe Network' 1968, Ph. D. Thesis, University of Aston in Birmingham.

NOMENCLATURE

$\mathbf{A}$ etc.	Matrices in Clarendon Medium/Century Schoolbook Bold type
$\mathbf{U}$	unit matrix
$\mathbf{A}$	branch by node topological matrix
$\mathbf{B}$	branch by path-to-datum matrix
$\mathbf{C}$	branch by mesh matrix
$\mathbf{D}$	branch by cutset matrix
$\boldsymbol{\gamma}$	transformation matrix
$\boldsymbol{\Lambda}$	a special topological matrix (defined on p. 22)
n	nodes in configuration
b	branches in configuration
m	meshes in configuration
KCL	Kirchhoff's Current Law (the 'First Law')
KVL	Kirchhoff's Voltage Law (the 'Second Law')
$\mathbf{O}$	null matrix
$\mathbf{e}'$	vector of node-to-datum path pressure rises, $(\overline{n-1}\text{x}1)$
$\mathbf{e}$	vector of branch pressure rises, (bx1)
$\mathbf{E}$	vector of branch pressure sources, (bx1)
$\mathbf{E}'$	vector of path pressure sources, (bx1)
$\mathbf{i}$	vector of branch flows not due to external flows, (bx1)
$\mathbf{i}'$	vector of mesh flows, (mx1)
$\mathbf{I}$	vector of branch flows due to external inputs, (bx1)
$\mathbf{I}'$	vector of node-to-datum path flows, $(\overline{n-1}\text{x}1)$
$\mathbf{V}$	vector of branch pressure rises due to branch resistances, (bx1)
$\mathbf{J}$	vector of branch flows, (bx1)
$\mathbf{Z}$	branch resistance matrix, (bxb)
$\mathbf{Y}$	branch admittance matrix, (bxb)
ρ	the degree of a node, i.e. the number of incident branches. The demidegree is $\rho/2$
iff	'if and only if'

Re	Reynolds Number
ρ	density
μ	Viscosity
Δh	head difference
D	diameter
L.	length
ln	natural logarithm
u	velocity
Q	quantity flow rate
ϕ	= (Re, ϵ/d), where ϵ is the roughness

Subscripts and superscripts

t	tree branches
ℓ	link branches
-1	matrix inverse
$\sim$	matrix transpose

INDEX

A 9, 10, 11, 12, 16, 17, 19, 21, 31, 33, 34, 37, 46, 49, 51, 52, 56, 72, 73, 86, 87, 105

Algorithm ATOM 34, 37, 41, 54, 70-75, 78, 87, 104, 107, 109, 110

- BREN 37, 43, 54, 76, 78, 80, 81-85, 86, 90, 104, 106, 108
- COL 39, 54, 97-98, 111, 114
- DIR 54, 58, 59, 60
- DISP 37, 39, 43, 54, 89, 91, 92, 93-96, 97, 101-102, 105
- HAMIL 37, 39, 40, 41, 43, 54, 82, 83, 85, 86-92, 93, 95, 97, 98, 99-100, 104, 107, 108, 110
- MAL 34, 37, 42, 54, 73, 75, 76-80, 81, 83, 93, 106, 108, 110
- MLC 17, (2.18)
- MR 43, 54, 65-69, 79
- PO 16, 17, (2.18)
- PODA 17, 21, (2.18)
- SEARCH BT 51, (6.1)
- WHE 28, 53, 61-64

Allan 45, (9.1), 46

Analogue computer 3, (1.7), (1.8), 5, (2.1), (2.2), (2.3), (2.4), (2.5), (2.6), 6, (2.7), (2.8), 13, (3.1)

Andrásfai 25, (4.13)

Atkinson 2, (1.5)

Atlas II 17

B 9, 13, 14, 16, 19, 21, 22, 31, 32, 33, 46, 47, 49, 50, 51, 52, 53, 56, 86, 105

Bailey's method 55

BASIC 19, 20, 34, 86

Bending 11, (2.17), 17, 46

Berge 23, (4.6)

Brameller 18, (3.4), 45, (9.1), 46

Branin 7, (2.9), 8, 15, 17, 18, (5.2)

C 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 21, 23, 31, 32, 33, 36, 37, 46, 47, 49, 50, 52, 56, 87, 88, 105

Caffrey 17, (9.2), 45

CHAIN 20

Chen 24, (4.10)

Choleski 17

Christofides 35, (6.4)

Chvatal 25

Colebrook-White 8, 38, 54, 99

Cross -see Hardy Cross

Cummins 24, (4.7)

D 31, 52

Daniel 8, (2.13), 9, 18, 33, 35, (6.5)

Deo 26, (4.15)

Diakoptics 11, 12, 15, 18, 36

Ϝ (digamma) 22, 24, 29-33, 36, 37, 43, 47, 49, 51, 52, 53, 54, 55, 56, 76, 80, 81, 86, 93, 103, 104, 105, 106, 108, 110, 111

Dirac 23, (4.4), 24, 53, 54, 58, 59, 60

Doland 7, (2.12)

Elliott 803 18

Equation 14 32, 41, 46, (6.8)

EXAMPLES 56, 103-117

Flores 26, 27, (6.3), 34, 35, 53, 61, 65, 70, 86

FORTRAN 17, 19

Freeman 3, (1.13), (1.7)

Gaussian elimination 12, 17, 18

Gauss-Jordan 17, 18, 45, 46, (9.1)

Gay 9, (2.16), 10, 11, 13, (2.19), 16, (6.6), 17, (6.8), 18, 36, 37, 38, 40, 41, 45, 46, 48, 58, 103, 104, 105, 106, 108, 110, 113

Gehner 24, (4.12)

Gibbon 1, (1.1), (1.2)

Graph theory 2, 7, 9, 13, 20, 23, 24, 25, 26, 51

Hamam 18, (3.4), 45, (9.1), 46

Hamilton 22, 53

Hamiltonian circuit 22, 23, 24, 26, 28, 54, 58, 60, 70

- path 21-28, 33, 34, 36, 39, 42, 43, 49, 50, 51, 53, 56, 59, 60, 61, 65, 69, 70, 76, 77, 78, 79, 81, 86, 93, 97, 103, 105

Happ 15, (5.1), 31

Hardy Cross 3, (1.9), 7, 8, 11, 12, 18, 35, 36, 37, 48

- - method 7

Hay 3, (1.12)

Hazen-Williams 3, (1.11), 8

Hercules 1

Hewlett-Packard 2000, 2000E 19, 20, 46

Hinsley 5, (2.4), (2.5), (2.6)

Hiramatsu 5, (2.5)

Howland 3, (1.7), (1.13)

Hudson 5, (2.4), (2.5), (2.6)

Hungarian method 24, (4.6)

Hutchison 11, (2.17), 17, 46

IBM 7

ICL 1903A 19, 45

Ingels 7, (2.11)

Inverse, inversion 12, 14, 15, 17, 18, 30, 31

Kajitani 24, (4.9)

Kamae 24, (4.8)

Kármán 3, (1.10)

KCL 2, 10, 12, 105

Kirchhoff 2, (1.3), (1.4)

Kishi 24, (4.9)

Kleitman 24, (4.11)

Knossos 1

Kozyrev-Grinberg 24

Kron 7, (2.10), 15, 19

KVL 2, 10, 12, 13, 105

Laguerre 55

LAT 17

Lesniak-Foster 25, (4.14)

M 26, 28, 34, 35, 37, 42, 53, 54, 61, 65, 70, 71, 73, 76, 77, 81, 82, 86, 97, 103

Maas 5, (2.1)

Mah 12, (2.18), 13, 16, (4.2), 17, 21, 51

Matrix 12, 14, 15, 16, 17, 19, 21, 22, 29, 30, 31, 33, 34, 36, 37, 41, 42, 46, 47, 50, 51, 52, 53, 56, 64, 85, 105, 108

- inversion 7, 14, 15, 17, 19, 30, 31, 43, 45, 46, 49, 50, 56, 104, 107, 108, 110

McIlroy 5, (2.3)

Membrain 15, (3.1)

Mesh method 7, 13, 18, 33

Middleton 8, (2.16), 10, 11, 16, (6.6), 17, 18, 36, 38, 40, 48, 57, 105, Middleton (cont'd) 110, 113, (E1)

Mixed method 7, 18

Mullineux 18, (3.3), 25, (6.2), 27, 33, 50, 51, 53, 54, 65

Nash & Thompson 5

Network 1, 7, 8, 10, 15, 18, 20, 25, 26, 32, 34, 49, 50, 51, 53, 56, 58, 64, 81, 98, 103, 106, 108, 110, 111, 114

Network Analysers Ltd. 6, (2.7)

Newton 3

Newton-Raphson method 2, (1.8), 7, 8, 18, 55

Nodal method 7, 18, 33

Ohm's law 11, 15, 86

Orthogonal method 7, 9, 13, 19, 49

Overlap 8, 21, 34, 35, 36, 37, 46, 55

Percival 36, (6.7)

PFI 12, 17, 18

Pósa 23, (4.5), 53, 54, 58, 59, 60

Powers 7, (2.11)

Prandtl 3, (1.10)

Preece 13, (2.19), 16, (6.1), 17, (6.8), 33, 36, 37, 40, 41, 45, 46, 51, 103, 104, 105, 106, 108

Reed 18, (3.3), 25, (6.2), 27, 33, 50, 51, 53, 54, 65

Regula Falsi 2, (1.6), 57

Roberts 26, 27, (6.3), 34, 35, 53, 61, 65, 70, 86

Roth 7

Scott 5, (2.2), (2.4), (2.6), 8, (2.14), (2.15)

Sparse, sparsity 9, 16, (3.2), 21, 31, 46, 55, 74

Taylor series 7

Taylor, S.N. 8, (1.7), (1.8)

Titan 17, 46

Travelling Salesman 35, 60

Wang 24, (4.11)

Whitney 23, (4.3)

Wilkinson 55, (10.1), (10.2)

Williams 6, (2.8)