THE ANALYSIS OF FLOW IN PIPE NETWORK SYSTEMS

(Vol 1)

by

PAUL ERIC PREECE

A thesis presented at the
University of Aston in Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

October 1972

SUMMARY

Summary

The main purpose of the work described in this thesis is the development of computer programs for the analysis of complex pipe network systems. For the solution of this problem the application of matrix methods and graph theory has been proposed.

Four network describing matrices, two of which have not been previously presented in a chemical engineering context, are used to produce digital computer programs which are more efficient than programs using the traditional theory of Hardy Cross. The programs are easy to use, the data preparation is simple, a flow distribution which satisfies Kirchoff's Laws does not have to be specified and no mesh formation is required. The methods are insensitive to initial estimates of flow and friction factor and therefore these can be assigned by the programs. The results generated by the programs are compared to the results of network analyses available in the literature.

For larger networks the technique of Diakoptics, an extension of the application of matrix methods, is proposed. It is shown that there are two diakoptic solution equations, one of which has not been previously presented. There are advantages, in terms of increased efficiency and the reduction of computer storage requirements, to be gained by the use of diakoptics. The solution equations can also be generated automatically.

Pumps and node to datum pressures may be included in a problem specification and a solution to such a problem achieved with little extension of the basic theory. It is suggested that designers, using basic theory only, can check if network problems are under or over specified.

Finally it is suggested that using basically identical techniques, matrix methods can be applied to a variety of chemical engineering problems such as the solution of systems containing mixed linear and non-linear elements.

Acknowledgements

The Author gratefully acknowledges the assistance of the following:

Professor G V Jeffreys

For providing facilities for the work to be carried out in his department.

Dr B Gay

For his supervision of this project and the many helpful suggestions and constructive comments on the preparation of this thesis.

The Science Research Council

For their research studentship grant.

Miss S Walker

For her diligent typing of this thesis.

Mr I F Preece

For his help in the drawing of the original tracings of some of the network systems discussed in this thesis.

The Staff of the Computer Centre, University of Aston in Birmingham

For their help in processing the computer programs presented in this thesis.

Table of Contents

APPENDICES

CHAPTER 1

INTRODUCTION

Introduction

In recent years problems associated with the design and analysis of pipework, both of chemical plant and distribution systems, have become increasingly complex, necessitating machine-aided solution. The economic considerations associated with large systems are also complex, since between 30 per cent and 50 per cent of the capital cost of a chemical plant is taken up by piping (47), the use of computer facilities is at a premium and design office overheads continue to rise. All of these indicate the need for efficient solutions to pipe network problems.

Chemical engineers have generally used the classical theories of Hardy Cross to solve network problems. However the traditional hand calculation methods are not compatible with modern computer theory, and therefore programs written to implement these methods suffer from the following disadvantages:

1) They require large amounts of computer storage.

2) They require much complex data preparation and a large degree of precalculation, which has to be carried out by a competent engineer, before a solution can be attempted.

3) They are not guaranteed to converge to a solution.

Modern computer theory is orientated towards using machines more efficiently by reducing occupation time and storage requirements accompanied by the minimum of basic input data.

Middleton (6) has shown that efficient computer programs can be constructed for pipe network analysis using modern principles of network theory originally developed by power systems analysts. He has also suggested that there are further extensions of electrical network theory which may be applied with advantage to pipe network analysis.

The purpose of this thesis is to investigate and extend modern network theory and computational techniques and apply these to pipe network problems with a view to producing more efficient computer programs which will satisfy the following criteria:

1) That only a minimum of input data should be necessarily prepared, thus reducing, as far as possible, sources of error.

2) That the programs should be capable of finding any errors in the data preparation.

3) That the programs should necessitate no precalculation before a solution can be attempted and can therefore be used by persons having no detailed knowledge of computers or any specialised branch of mathematics.

4) That in operation the programs should be efficient in time and storage requirements.

It is with these considerations in mind that the present study was undertaken.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

It was reported (6) in 1968 that all the methods of solving pipe network problems were based on techniques developed in 1936 by Hardy Cross (17), although since 1965 various authors (9), (20) have shown interest in the application of matrix techniques to pipe network problems. It has been claimed that these latter techniques have many advantages over established methods of analysis, but chemical engineers in general do not appear to be familiar with the matrix, graph and tensor theory, which is commonly used by electrical engineers involved in the study of power systems analysis to solve network problems.

The advantages claimed for the matrix analysis of networks have not been applied to pipe network problems by more than a few authors (6), (42); therefore any survey of the literature of network analysis by a chemical engineer wishing to apply modern techniques to pipe network problems must be a twofold exercise. On the one hand the existing methods of pipe network analysis must be examined and evaluated to determine any deficiencies and find areas of possible improvement, and on the other the analysis of network solutions outside of the discipline of chemical engineering must be examined and evaluated to discover if any of the techniques used therein can be applied with advantage to pipe network problems. Chapters (2.2) to (2.5) therefore deal with the methods of solution based on the work of Hardy Cross, and Chapter (2.6) reviews the methods of network analysis based on matrix techniques which have been reported by workers in the electrical engineering and power systems discipline.

2.2 The Hardy Cross Methods

In 1936 Hardy Cross (17) proposed two methods of network analysis of fluid flow problems based on the two Kirchoff Laws. For the solution to a network problem to exist, it is a requirement that the following two laws be satisfied.

Law 1:

The algebraic sum of all of the currents flowing into and out of any node or network junction is zero.

Law 2:

The algebraic sum of all the voltage differences around any closed loop or mesh of a network is zero.

The two Kirchoff Laws were developed for use with electrical networks but they may equally well be applied to pipe networks, provided that fluid flow is the analogue of current, and pressure the analogue of potential.

A further relationship is required for the Hardy Cross solutions, it is the relationship for any pipe of pressure, flow and hydraulic resistance which is analagous to Ohms Law, given by equation (2.2.1)

$$V = ZJ \tag{2.2.1}$$

The relationship for fluid networks is assumed to be equation (2.2.2)

$$\Delta P = RQ^n \tag{2.2.2}$$

The two solution methods which require different iteration schemes are relaxation methods. If the flows are taken as unknowns an initial guess is made of the solution which satisfies the first law and an iteration which recalculates the solution is carried out until the flow distribution satisfies the second law. Alternatively if the pressures are taken as unknowns, an initial guess is made of the solution which satisfies the second law and is iterated upon until the first law is satisfied.

The two solution methods are commonly called the methods of balancing flows and the method of balancing heads, but are referred to in this thesis as the Hardy Cross mesh and nodal methods.

For various reasons which are discussed at a later stage, the mesh approach has attracted the most attention, and it is proposed to show the development of the Hardy Cross theory of this method. The theory of the nodal approach is similar and need not be developed.

According to Kirchoff's second law, the solution to a network problem is achieved if the sum of the pressure drops around every closed loop is zero, as expressed by equation (2.2.3)

$$\Sigma \Delta P = \Sigma RQ^{n} = 0 \qquad (2.2.3)$$

For any pipe of the network with an initial guess of flow Qo, the solution of the flow Q is expressed by equation (2.2.4),

$$Q = Qo + \xi \qquad (2.2.4)$$

which may be expressed in terms of pipe pressures as equation (2.2.5)

$$RQ^{n} = R(Qo + \xi)^{n} \qquad (2.2.5)$$

which can be expanded by the binomial method to yield equation (2.2.6)

$$RQ^{n} = RQo^{n} + \sum_{b=1}^{\infty} \frac{R\ n!\ Qo^{n-b}\ \ \xi^{b}}{b!\ (n-b)!} \qquad (2.2.6)$$

If Kirchoff's second law is assumed to be true and ξ is sufficiently small, then equation (2.2.6) may be truncated after the first summation and considering every pipe in any mesh, an equation of the form of (2.2.3) may be set up for every mesh as in equation (2.2.7)

$$\Sigma RQo^{n} = \ -\Sigma RnQo^{n-1}\xi \qquad (2.2.7)$$

This may be manipulated so that for any given mesh the correction of flow to be added to each pipe in the mesh, provided ξ is assumed to be constant for each pipe, is given by equation (2.2.8)

$$\xi = -\left(\frac{\Sigma RQo^{n}}{\Sigma RnQo^{n-1}}\right) \qquad (2.2.8)$$

which can be expressed more readily as equation (2.2.9)

$$\xi = -\frac{\Sigma\Delta P}{\Sigma R'} \qquad (2.2.9)$$

where the signs of $\Sigma\Delta P$ and $\Sigma R'$ of this equation are respectively with and without due reference to the direction of flow in the pipes.

2.3 The Iteration Scheme

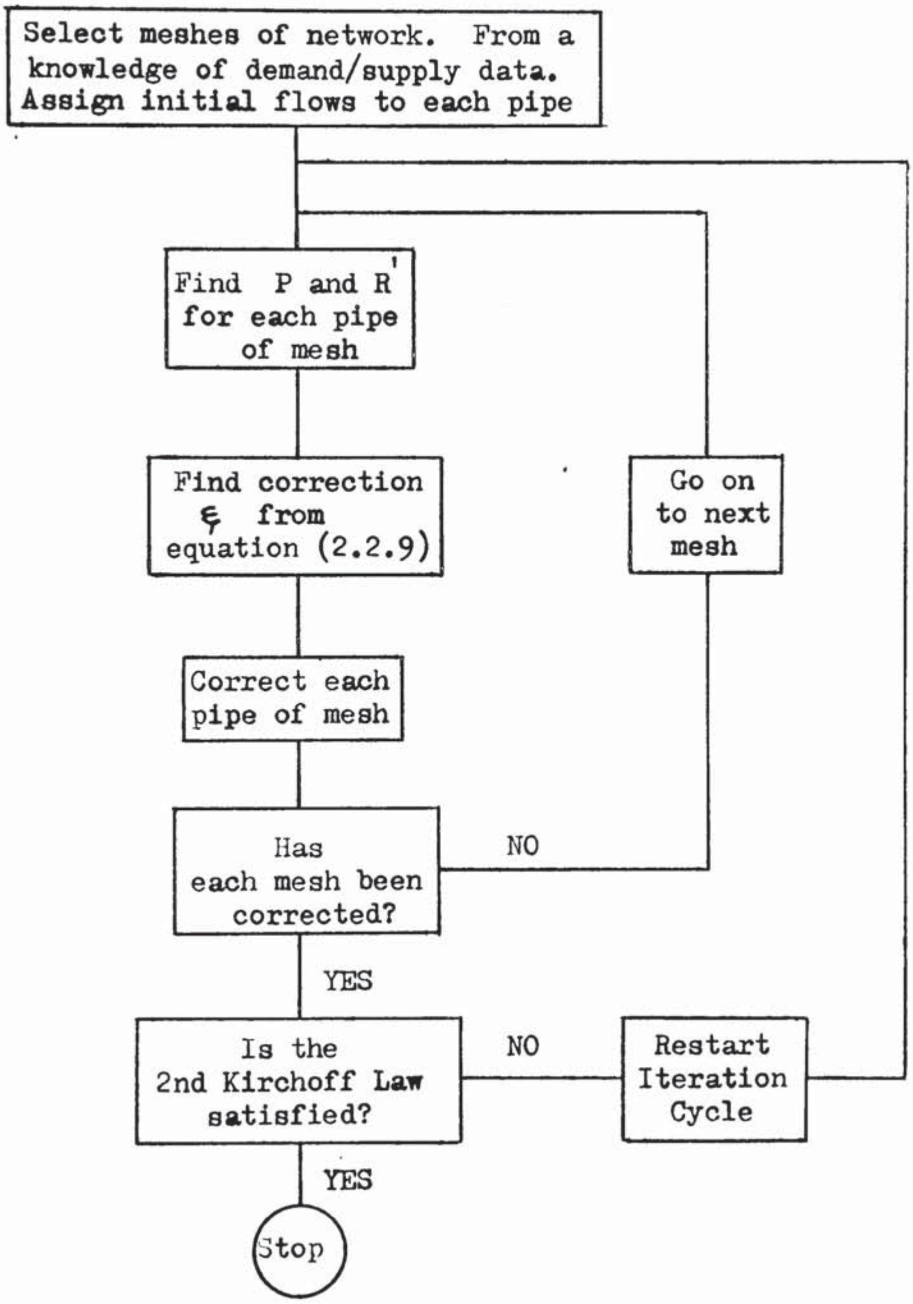
Equations (2.2.3) to (2.2.9) form the basis of the iteration scheme of the mesh method. From a knowledge of the demands and supplies on a network, each pipe is assigned a flow so that for the overall network Kirchoff's first law is satisfied, from which ΔP and R' are evaluated for each pipe. Having specified the pipes which make up particular meshes ξ may be found for each mesh. Hardy Cross selected the meshes of a network by eye based on experience. It is shown in Chapter (3) that there are specific relationships between the pipes, junctions and meshes of a network developed by Veblen (25).

Having found ξ from equation (2.2.9) for a first mesh, the flows in the pipes of that mesh are corrected and the process repeated for all succeeding meshes. If after all the meshes have been corrected, the second Kirchoff Law is not satisfied, the first mesh is recorrected and the whole cycle repeated until the second Law is exactly or very nearly satisfied. A flow chart which is the basis of a computer program developed to utilise the above method is shown in figure (2.2.1).

Perry (43) says that the exponent n of equation (2.2.5) must be a positive integer for the sum formula of the binomial expansion to apply, and Hardy Cross points out that truncation of equation (2.2.6) is only valid if ξ is small. In the fluid flow case n will be between unity and two, depending on the Reynolds Number, and ξ may be very large, especially at the beginning of the iteration scheme. However since any pipe may be included in more than one mesh, and will be corrected more than once during each iteration cycle, Hardy Cross maintained that convergence would be

Figure (2.2.1)

The Iteration Scheme of the Hardy Cross Mesh Method


Select meshes of network. From a knowledge of demand/supply data. Assign initial flows to each pipe
Find P and R' for each pipe of mesh
Find correction ξ from equation (2.2.9)
Go on to next mesh
Correct each pipe of mesh
Has each mesh been corrected?
NO
YES
Is the 2nd Kirchoff Law satisfied?
NO
Restart Iteration Cycle
YES
Stop

sufficiently rapid for practical purposes. It has been observed by others, notably Middleton (6), that this is not always the case.

The Hardy Cross methods were developed for hand calculations, implying that only small networks were considered, and that the calculator was familiar with the type of problem to be solved. This thesis is concerned with the solution of large scale problems, for which it is difficult to acquire a "feel", let alone calculate solutions by hand. This "feel" is almost impossible to program, and so a predetermined solution pattern must be followed, which can lead to lengthy calculation and inefficient convergence.

2.4 Applications of the Hardy Cross Methods

The problems of programming, convergence and data handling associated with the Hardy Cross methods have been discussed by many authors, but before analysing their contributions in detail, it is proposed to outline some of the common points by way of introduction.

Travers says (2) that the convergence of the Hardy Cross methods is dependent on choosing the tree of minimum resistance, ie as far as is possible, the tree is made up of the branches having the least resistance. A tree is assumed to be any path through the network which passes through all the nodes and forms no closed loops. It is also argued by Daniel (3) and Middleton (6) that convergence is most rapidly achieved if the tree of minimum overlap is chosen as a starting point. The condition of minimum overlap is given by having, as far as is possible, the minimum number of branches in each mesh. Applying the nodal method Van der Berg (26) points out that convergence is dependent on the order in which the nodes are analysed. The difficulty of data preparation and handling has been reported by many workers (1), (2), (3), (37), (35), and various attempts have been made to obviate the necessity of defining the meshes by eye, and assigning an assumed solution to the network.

A list of some of the advantages and disadvantages of using the Hardy Cross methods serves as a prologue to the discussion of the computer applications.

2.4.1 Disadvantages of the Hardy Cross Methods

1. Might not converge due to
   (a) meshes being wrongly chosen either because the tree of minimum resistance or minimum overlap has not been chosen;
   (b) nodes have been analysed in the wrong order;
   (c) initial guesses have been wrongly chosen.
2. Data preparation is tedious and likely to be a source of errors.
3. Needs "feel" for problem, which is difficult with large networks.
4. Equations cannot automatically take into account pressure specified nodes or pump terms in pipes.

2.4.2 Advantages of Hardy Cross Methods

1. Easy to program in terms of the simplicity of the method.
2. Choice of methods, depending on the type of network to be analysed.

2.4.3 Computer Application of Kniebes and Wilson (27)

One of the easiest reported computer solutions to the network problem was by Kniebes and Wilson, using the Hardy Cross mesh method based equation (2.2.2) with an exponent n of value 1.8. The program is reported as being most efficient for systems of the order of 250-400 pipes, but it is difficult to assess the value of this information since no details of the numbers of meshes in the systems are available. More importantly, it is reported that the solution was efficient for large error criteria, but the number of iterations increased markedly if greater accuracy was required. This has also been noticed by the author and is discussed in Chapter (5).

2.4.4 The Computer Application of Hunn and Ralph (28)

Quantitative information on a computer application of the mesh method is given by Hunn and Ralph (28), the program allows for the inclusion of pumps or other non pipe elements that have a pressure versus flow relationship that can be expressed as a polynomial. A novel way of assigning the initial guesses of flow to the pipes is demonstrated. It appears to be the same as assigning the demands and supplies to the tree pipes only. This is a feasible solution from which a true solution may be calculated. The actual pipes making up the tree of the network are also used at the end of the calculation to evaluate the pressures at the pipe junctions from a knowledge of the individual pipe pressure drops. Hunn and Ralph (28) state that this assignment of flows in the correct manner to the tree pipes is the most critical operation of the entire data preparation, but it is reported by Middleton (6) and Daniel (3) that the assignment problem is not nearly as influential as loop formation on the rate of convergence. Further investigation of the Hunn and Ralph work reveals however, that the assignment problem is in fact one of correct loop formation.

The fact that complex data preparation and handling were required before a solution could be attempted on the computer cannot be criticised, since it was probably a property of the limited storage and input capability of the early IBM 650 machine that was used. Care was taken to ensure that the data input to the machine was correct by operating data checks, such as testing that all loops were closed paths. The results that Hunn and Ralph obtained in terms of the flows in each pipe at solution are used by Middleton (6) to compare the accuracy in terms of individual pipe flows with the results obtained by the diakoptics method.

2.4.5 The Computer Application of Ingels and Powers (29)

A further application of the mesh method in a computer program is reported by Ingels and Powers (29). They showed that calculations based on the Hazen-Williams relationship with a constant value of the exponent n of equation (2.2.2) can be seriously in error, typically about a 20 per cent error for flows of 100,000 lb/hr in 6 inch pipes. To overcome this, a more realistic flow equation developed by Ingels (30), which approximates the friction factor - Reynolds Number relationship of the Moody diagram (31) is used. They used for Re $>$ 2100, a power series of the form of equation (2.4.1)

$$\phi = a + c\Theta + d\Theta^2 \qquad (2.4.1)$$

where Θ is expressed by equation (2.4.2)

$$\Theta = (-b + \log_{10}(Re))^{-1} \qquad (2.4.2)$$

and a, b, c and d are polynomial functions of the relative roughness $^{\epsilon}/D$ The friction factor found by this method is then used in the Darcy-Wisbech relationship given by equation (2.4.3)

$$\Delta P = \frac{8\phi LQ^2}{g_c\pi^2D^5} \qquad (2.4.3)$$

It is also shown that by truncating a Taylor series expansion of equation (2.4.4)

$$\Delta P = f(Q) \qquad (2.4.4)$$

that R' the resistance term of the Hardy Cross expression of equation (2.2.9) is equivalent to equation (2.4.5)

$$R' = \frac{\partial\Delta P}{\partial Q} \qquad (2.4.5)$$

which may be rewritten using the Darcy-Wisbech relationship and the empirically determined friction factor to give equation (2.4.6)

$$R' = \frac{\partial\Delta P}{\partial Q} = \frac{8L}{g_c\pi^2D^5} \quad \underline{2\phi Q} + \frac{Q^2\partial\phi}{\partial Q} \qquad (2.4.6)$$

which on substitution for $\partial\phi/\partial Q$ gives equation (2.4.7)

$$R' = \frac{2\Delta P}{Q} - KQ\Theta^2 \log e\ (c + 2d\Theta) \qquad (2.4.7)$$

wherein K of (2.4.7) is expressed by equation (2.4.8)

$$K = \frac{8L}{g_c\pi^2D^5} \qquad (2.4.8)$$

The empirical relationship, equation (2.4.7), is said to work well for turbulent flow in rough pipes, but varies considerably from the Moody diagram (31) for low values of roughness at high Reynolds Numbers. Though the importance is realised of dealing with "real" systems, it is felt that the above development is unnecessarily complex and a better method of calculating realistic friction factors is proposed in Chapter (3). The work (29) also proposes a method of assigning the initial guesses of flow to the pipes of the network by starting at the major demand or supply junction and working through the network proportioning the flows by a simple power law. No mention is made of the significance or value of this procedure.

Using equation (2.4.3), three networks previously reported were analysed, the largest of these due to Dolan (32) and the results associated with this network are examined in Chapter (5), together with the results obtained for the same network by the author.

2.4.6 The Work of Travers (2)

The problem of convergence of the Hardy Cross methods has been studied by Travers using a modification of equation (2.2.2) with a value of 1.85 for the exponent n. It is reported that networks of 650 pipes and 500 nodes have been analysed and that the mesh method converges faster. This is to be expected, since such a system would contain only 151 meshes. Significantly less than the number of nodes, thus requiring far fewer computational operations for a solution to be achieved by the mesh method. A further limitation of the work is the apparently unrealistic resistances which have been assumed to include constant friction factors. It was indicated in Chapter (2.4) that Travers proposed that maximum rate of convergence is achieved by choosing the tree of minimum resistance; if, as in most cases, the pipes are of similar resistances, this procedure is equivalent to choosing the tree of minimum overlap as advocated by Daniel (3).

Further evidence of the type of data preparation problems encountered with large scale problems is also outlined, and like previously reported authors, Travers proposes (2) the use of sub-programs to find the meshes of a network from simple input and also the use of error detection procedures for examining the input data. Concluding, Travers points out that there will always be a strong case for using the mesh method since most of the networks, especially distribution networks, that chemical engineers analyse are not heavily meshed.

2.4.7 The Application of the Mesh Method by Daniel (3)

A comprehensive application of the mesh method which takes into account compressible and incompressible flow by constructing polynomials for changes in viscosity and density with respect to pressure and temperature is proposed by Daniel (3). The program which uses as its basis the flow sheet shown in figure (2.2.1), includes an extra iteration loop not shown on that figure. This iteration loop finds the friction factor at high Reynolds Numbers using the Colebrook relationship of equation (2.4.9)

$$\phi = \frac{1}{\left(0.868 \ln \left(\frac{\epsilon}{3.7D} + \frac{2.51}{Re\, \phi^{\frac{1}{2}}}\right)\right)^2} \qquad (2.4.9)$$

This determines the pipe resistances more accurately than any of the previous works. The friction factors are recalculated on the outermost iteration loop of figure (2.2.1) only, by a self contained iteration scheme. Middleton suggests (6) that this has an advantage in terms of the saving in computation time, but it is shown in Chapter (5) that significant reduction in the total time needed for computation is achieved if the friction factors are determined on the innermost loop of figure (2.2.1). Despite this failing the program is useful since it demonstrates a novel way of finding the correction ξ for a mesh, which takes into account the direction of the correction automatically by using a branch-mesh incidence matrix. This matrix is fully described in Chapter (3). The recognition of the existence of this matrix is an important step towards the total matrix analysis of networks, but Daniel (3) used the matrix only as a storage tool which could handle the complex data associated with large networks. Considering this last point, Daniel also advocated that any computer program should be capable of finding inconsistencies in the data preparation. To simplify this preparation he proposed a method of generating a branch-mesh incidence matrix from a knowledge of the tree and link branches. However the resulting meshes remain those which would have been chosen by eye. Daniel notes this fact and declares that in his early work he was disturbed by the fact that the maximum overlap choice of meshes could produce convergence problems, and so developed a program which attempted to find the tree of minimum overlap. It is reported that this program took longer to run than the program which achieved a solution by the Hardy Cross mesh method under the maximum overlap condition.

A comparison with a modified Newton-Raphson iteration scheme using the Jacobian matrix of derivatives was attempted but abandoned because of convergence problems. This particular method has been further successfully investigated by Brameller (20) and is discussed more fully later.

Bearing in mind the necessity to attempt to solve "real" problems, it is worthwhile considering the statement by Daniel that networks should be analysed in terms of mean, minimum, maximum and fire fighting flow conditions, and Middleton (6) has some comments which are discussed later on the convergence of programs under these four conditions.

2.4.8 The Application of the Nodal Method by Van der Berg (26)

All the applications so far discussed have been based on the mesh method; an application which is similar to the nodal method of Hardy Cross was developed by Van der Berg (26). He stated that the convergence of the nodal method was dependent on the order in which the nodes are analysed. To take advantage of this he constructed a procedure for determining the required order.

For every node he constructed a graph by plotting the value of a residual flow r at the node against the actual pressure P at the node. The residual r_i which is the difference between the inputs and the outputs, external sources considered, of any node i is given by equation (2.4.10)

$$r_i = \sum_j Q_{ij} + I_i' \qquad (2.4.10)$$

For a converging solution the residual r calculated after each iteration would be expected to reduce to zero, and the pressure at the node to converge to its true solution. A graph such as the one shown in figure (2.4.1) would therefore be expected for every node.

Van der Berg maintains that the node to be corrected first has the maximum value of all the graphs for the integral of equation (2.4.11)

$$\eta i = \int_{P_i^{(0)}}^{P_i^{(1)}} r_i dP_i \qquad i = 1, 2\ldots \text{ number of nodes} \qquad (2.4.11)$$

This integral which has as its limits the most recently calculated pressure $P_i^{(1)}$ and the previously calculated pressure $P_i^{(0)}$ for a particular node i is approximately equal to the area of the triangle shown on figure (2.4.1) and therefore ηi may be expressed by the approximation shown in equation (2.4.12)

$$\eta i \approx \tfrac{1}{2} r_i \ (\bar{P}_i - P_i^{(0)}) \qquad (2.4.12)$$

where $\bar{P}_i$ is assumed to be the approximate value of the pressure which reduces the residue, and hence the amount of pressure correction to be applied at the node, to zero. The actual value of $\bar{P}_i$ may be determined by letting the pressure $P_i^{(0)}$ at node i be equal to the pressure $P_j^{(0)}$ of any adjacent nodes j. The values of the flows in the pipes ij can then be calculated for two pressures P_i' and P_i'' such that the value of the residual of flow at node i changes sign. To complete the solution of equation (2.4.12), $\bar{P}_i$ may then be obtained by the approximation given in equation (2.4.13)

$$\bar{P}_i \approx \tfrac{1}{2} (P_i' + P_i'') \qquad (2.4.13)$$

For a converging solution with the residual of flows at every node reducing, the calculated value of the nodal pressures $P_i^{(0)}$ will move into the interval

Figure (2.4.1)

Residue as a function of Pressure

Van der Berg (26)

of P_i' and P_i'' eventually to coincide with the value of pressure P_i at solution.

The above procedure does not give a method for calculating the corrected value of pressure $P_i^{(1)}$, it simply indicates where the correction ought to start. It is thought by the author that Van der Berg's selection mechanism is based on correcting first the nodes which are converging to a solution the least quickly.

The method of correcting any nodal pressure derived by Van der Berg is based on finding a correction $\Delta P_i^{(0)}$ for node i which has to be added to the calculated value of the nodal pressure $P_i^{(0)}$ at node i. $\Delta P_i^{(0)}$ is given by equation (2.4.14)

$$\Delta P_i^{(0)} = n\, r_i \Big/ \left(\sum_j Q_{ij}^{(0)} \Big/ \ P_i^{(0)} - P_j^{(0)} \right) \qquad (2.4.14)$$

with the result that the updated value of pressure at node i is given by equation (2.4.15)

$$P_i^{(1)} = P_i^{(0)} + \Delta P_i^{(0)} \qquad (2.4.15)$$

It is claimed that the improved method of calculating updated nodal pressures, together with the knowledge of where to start the corrections, will speed up the calculation procedures. However, for large systems one cannot follow a true optimum strategy for the choice of nodes to be corrected. This is because of the two approximations of equations (2.4.12) and (2.4.13) and also because having corrected a particular node, the integral values in the area immediately surrounding that node will have changed, necessitating their recalculation.

Van der Berg maintained that it is possible to overcome this feature by letting the new nodal pressure leave a residue that has some value greater or less than zero, the accelerating factor being determined by the user's experience.

The final non matrical approach to network analysis is also based on the Hardy Cross Nodal method.

2.4.9 The Application of Knights and Allen (35)

The nodal method was chosen by Knight and Allen (35) because in a preliminary analysis of the methods available they thought that its advantages of simpler data preparation and programming, together with more certainty of a result, seemed to outweigh the fact that convergence was slower. The main criticism of their method apart from the arbitrary node numbering, which is common in the gas industry (41) is the use of the Drew and Generaux (36) relationship, for finding the friction factors, given by equation (2.4.16)

$$\phi = 0.0351\ \mathrm{Re}^{0.152} \qquad (2.4.16)$$

This suffers from the usual errors in straight line plots on a log-log scale when compared with a Moody diagram (31). However the quantitative results produced for a sample network, the geometry of which is indicated, are used in Chapter (5) and compare favourably with the results obtained using programs developed as a result of the work of this thesis.

2.4.10 Summary

Most of the developments in network analysis described so far took place before 1965. In view of the fact that cheap, fast and large storage capability computers were not readily available before that year, it is unjust in the light of the availability of modern machines to over-criticise the various attempts at developing computer programs to solve network problems. However some general comments on the comparison of the various applications is justified.

It is difficult to evaluate many of the results of the applications, different machines were used, different workers used varying programming techniques, but more importantly little information is available about the geometries of the systems analysed. Workers have reported efficiencies of methods for varying numbers of pipes with no reference to the numbers of nodes, meshes and the way the meshes were made up. The very use of the words "efficient convergence" is also difficult to evaluate since some authors (37) say that since the data available about real systems may be subject to errors of plus or minus ten per cent, the convergence criterion should be adjusted accordingly, whilst others concentrating on evaluating "best" methods apply the strictest limits to convergence.

In the late 1960's and early 1970's a new trend towards the use of matrix analysis can be observed. The reason for this is that the use of matrix techniques for the manipulation of large sets of numbers is very compatible with the workings of modern computers. The concept of matrix methods as computation tools is discussed more fully elsewhere in this thesis.

Notwithstanding the use of matrix techniques, the underlying theories of more modern approaches to pipe network problems remain based on the Hardy Cross approaches or some modification of those methods.

2.5 The Matrix Application of Fincham et al (1)

In 1970 a paper by Fincham (1) shows that the application of matrix methods to gas distribution systems could aid the solution of the network analysis. Fincham recognised the existence of two major matrices, a $\underline{C}$ matrix which relates the way in which the branches of a graph of a network are connected to the meshes of the network, and an incidence matrix $\underline{A}$ which relates the way in which the branches of the graph of a network are connected to the nodes of a network. He also realised that this $\underline{A}$ matrix could be partitioned into two submatrices, one of which is square and non-singular, and that a submatrix of the $\underline{C}$ matrix is related to the square non-singular submatrix of $\underline{A}$. The important concept which Fincham's work lacks is that the partitioning of any matrix of a graph of a network is equivalent to partitioning the network itself as demonstrated in Chapter (3).

Through the use of the $\underline{C}$ matrix Fincham developed a set of m linear equations of the form of equation (2.5.1)

$$\underline{Er} = \underline{\check{C}}\ \underline{Zr}\ \underline{C}\ \underline{Xr} \qquad (2.5.1)$$

This is similar to an equation developed in Chapter (3) which gives the relationship between the mesh flows Xr, and the pipe pressure terms Er, and the individual pipe impedances Zr transposed to mesh quantities.

The equation was developed as a result of the linearization of equation (2.2.2) into the form of equation (2.5.2)

$$\Delta P = (R\ |Q|\ )Q \qquad (2.5.2)$$

Equation (2.5.1) was solved by a Newton Raphson method, if an approximation of flow Xr exists a better approximation Xr + 1 can be found. The method was found to be convergent. Further extensions (42) of the matrix theory allowed for the inclusion of non pipe terms such as compressors, but the development is unnecessarily complicated, further matrices being introduced which do not have the true transformation character of the $\underline{A}$ and $\underline{C}$ matrices. A very simple way of representing non pipe terms and pressure specified nodes is shown in Chapter (3), which is inherently more acceptable to the network theorist. In a later work (37), Fincham reviews the methods of network analysis in programmed form which were available to his organisation. It is reported that all the methods were based on the Hardy Cross mesh method (indicating that the theoretical treatment outlined in (1) had not been made use of). Many of these programs are outside of the scope of this thesis since they deal with high pressure and transient flow, but the criteria on which the programs are designed are applicable to all network analyses. Data preparation must be simple, error diagnostics must be written into programs, and in the case of gas networks, especially older distribution systems where the demand and supply data is within only about ten per cent of the assumed value, the convergence criterion must be adjusted accordingly.

An interesting feature of the developed programs (37) is the inclusion of an accelerating factor. Instead of adding to each loop the correction factor ξ, a modified factor $W\xi$ is added where W is given by equation (2.5.3)

$$1.2 < W < 1.6 \qquad (2.5.3)$$

W has been found empirically and it is suggested that the limits given are the most efficient. Increased efficiencies up to 50 per cent are reported.

2.5.1 The Application of the Mesh Method by Brammeller (20)

For every mesh of a network, Brameller developed an equation which gave the solution of the mesh flow in terms of an initial guess of the individual pipe flows and the unknown mesh flows. This is achieved by summing the individual branch pressure drops around every mesh. The result is a set of M equations, where M is the number of meshes, of the form of equation (2.5.4)

$$f(q_k) = \sum_{1}^{b} R_{ij}\ (Q^o_{ij} + q_k)^2 \qquad (2.5.4)$$

where k is from 1 to M, b is the number of branches in any mesh and i, j are the terminal nodes of branch b.

For a converged solution the sum of the pressure drops around every mesh will be zero and the LHS of equation (2.5.4) will also be zero. The Newton-Raphson iteration scheme is used by Brameller to solve the k set of equations. This involves finding the inverse of the matrix of partial derivatives called the Jacobian matrix. From this an updated value of flow Q^1 can be found according to equation (2.5.5)

$$\underline{Q}^1 = \underline{Q}^o - \underline{J}^{-1}\ (\underline{Q}^o)\underline{f}(Q^o) \qquad (2.5.5)$$

Evaluating the method, Brammeller states, "In many instances the method proved very successful but it has the disadvantage of being very sensitive to the correct estimation of initial guesses and the mathematical conditioning of the equations. Even with the most advanced techniques for setting initial values of flow the convergence can be comparitively slow."

The author believes that the "mathematical conditioning of the equations" is the same problem of correct mesh formation noticed by the users of the standard Hardy Cross mesh method. The technique advocated by Brameller has not been further investigated by the author.

2.6 Introduction

It has been indicated (1), (2), (3) that the use of matrix techniques as data handling tools is invaluable, and theoretically (1), (42) that matrices can be an aid in the analysis of large scale distribution problems, using modern computers. Matrix analysis has however been used by workers such as Kron and Happ, not as tools which help the solution of network problems, but in the fundamental approach to network analysis. The use of matrices as network describing devices (1), (2), (3) is only a small facet of the classical work of Kron; this work, "Tensor Analysis of Networks" (4), has remained the fundamental source of information relating to the solution of large scale power distribution systems.

The principles of the orthogonal network and the square non singular transformation tensor were established in the work (4), and later developments (11) in the fields of electric theory related to rotating electrical machines led to the development of diakoptics(5). The word "diakoptics" has been given many interpretations, but it can be taken to mean "tearing apart a system". The three major works (4), (5), (11) were in the early years little understood and discredited as being extensions of matrix partitioning techniques, heavily veiled in tensor theory. For this reason the techniques associated with diakoptics were little used until very recently. The first practical application of diakoptics to fluid flow systems was attempted by Middleton (6) in 1968, though earlier Kron (9) had proposed a theoretical treatment of fluid network systems. The literature is replete with Kron's theoretical treatments of subjects related to disciplines outside his own, (7), (8), (10) and although few of his suggestions have found application it is argued that chemical engineers, using those suggestions, would find many valuable new tools for problem solving, as is discussed in Chapter (5).

2.6.1 The Matrix Approach to Network Analysis and its Extension, Diakoptics

The most basic concept underlying Kron's approach to network analysis is the decision to look at a network in a series of alternative ways. In the preceding discussion the pipes of a network have been the focal point, with the emphasis shifted to the ends of the pipes, as in the nodal approach, or to the way flow takes place in a series of branches, as in the mesh approach. Kron's concept in simple terms can be visualised as taking both of these focal points into consideration at the same time, not treating them as isolated reference frames. Extending this, Kron suggests that the single reference frame can be established by certain paths within a network. There are two types of path, the open path, which is a path which connects nodes but does not form closed loops, and the closed path which is equivalent, but not necessarily identical, to the mesh. The importance of these paths and a rigid definition of their existence is outlined in Chapter (3).

The new reference frame, called the orthogonal reference frame, concerned with the two types of path of which there are in total the same number as there are pipes in the network, is used to set up transformation tensors associated with the network which are square and non singular. From this Kron was able to develop solution equations to the network problem and claimed that these equations are automatically generated (5) by the very existence of the transformation tensors. He was further able to show that any network system may be torn into subdivisions, a quasi-solution obtained for each subdivision, and reconnected to obtain an overall solution. Two distinct advantages of using this latter approach rather than any other are that larger systems may be solved, and because of this, it is possible that faster solution times will be achieved pro ratum using the same computer.

2.6.2 The Application of Orthogonality

Within the discipline of electrical engineering, a few workers have attempted to simplify the work of Kron. In 1950 le Corbeiller applied Kron's interpretation of the orthogonal network to simple resistance electrical networks (12) and produced two matrix methods of attacking network problems; the node to datum and mesh methods. It was claimed that these were different from the classical nodal and mesh methods developed by Roth. However Roth, a graph theorist, was able to justify Kron's tensor approach in terms of topological concepts (14) and show that a direct analogy could be drawn between his own approach and that of le Corbeiller. Roth later showed (13) that Kron's method of tearing could also be justified in terms of topological concepts and that there are distinct advantages to be gained in terms of the solution time of a particular problem, when compared to partitioning, K partitioning and standard inversion. The work (13) unfortunately only takes into account the nodal method of diakoptics and no comparison is given for the mesh diakoptics method of which Roth does not seem to have been aware.

In 1967 Branin, a circuit theorist, states that (15) the ever increasing size of the problems of that discipline were good applications to test the theories of Kron. As early as 1956 Branin had interpreted the tensor concepts of Kron as purely matrix equations (16), but at the same time shows that the orthogonal mesh and nodal equations are not simply sets of algebraic equations but are network equations. Further, Branin points out that (16) there are at least four ways of looking at the two square non singular transformation tensors $\underline{\gamma}$ and $\underline{\alpha}$ in terms of tieset and cutset, incidence, boundary and coboundary and tensor operators, depending on the discipline with which one is concerned. This has probably been one of the most consistent points of confusion over the validity of Kron's work.

Using boundary and coboundary operators, Branin successfully shows the difference between the Kirchoff-Maxwell or Hardy Cross (17) interpretation and that of the orthogonal approach of Kron. He also shows (16) that the operators $\underline{\gamma}$ and $\underline{\alpha}$ are made up of two separate submatrices which can themselves be partitioned.

Convinced of the potential of the Kron methods, Branin went on to develop computer programs (18) to solve circuit problems. Describing this work (19), Branin reiterates the comments of Hardy Cross type of analysts, in that the data input to a computer by a user must be reduced to a minimum to save on the likelihood of errors, and if it is possible to let the machine set up the matrices needed for computation, and that the program must be able to detect errors in the input. A more technical contribution suggests a way of minimizing the total computational time required for a solution to a network problem by reducing the amount of matrix inversion that is normally required for all the Kron methods. Branin shows a method called the "link at a time" method of constructing a matrix identical to the inverted nodal solution matrix without inversion. Branin reports, however, that the proposed algorithm is untested with regard to its computational efficiency. This matter has been investigated by the author and a comparison of solving network problems by this method and ordinary inversion methods is recorded in Chapter (5). The same work (19) outlines a further network defining matrix $\underline{D}$, which although it has no relationship with diakoptics is useful since it explains in topological terms the approach of Brameller (38) and, as shown in Chapter (3), may be used as a substitute for the $\underline{A}$ matrix in the nodal method of solution.

Further work on producing circuit analysis programs by Branin (21). justifies the use of the matrix techniques of Kron in terms of their bookkeeping abilities and their compatability with modern computers. It is also pointed out that since most of the matrices used in describing the network are sparse, it is important that the use of sparsity techniques such as those of Sato and Tinney (22) for improving the efficiency of inversion is considered. The importance of bearing in mind engineering applications in the development of theoretical treatments is adequately shown in the description of a further network analysis program NETI (23). This includes in one of the iteration loops the engineer himself.

Branin has made many more useful contributions than those outlined above and they have been included in the author's theoretical treatment of the network problem and are pointed out in Chapter (3).

2.6.3 Brameller and Diakoptics

Brameller, who was discussed earlier as having made contributions to the Hardy Cross methods of analysis (20), has also applied Kron's diakoptic techniques to electrical systems (39). He demonstrated the manner in which tearing a system into its component parts is analogous to partitioning of a set of equations. Linear electrical problems are solved using tensorial analysis based on the study of concepts that remain invariant when a system is broken up into components and afterwards reconnected into any possible configuration. However the explanation of the significance of the matrices used related to network variables is treated in electrical tensor terms. This could confuse chemical engineers not familiar with tensor concepts. Apart from showing that the method of tearing and reconnection is a viable computational procedure, some valuable comparisons between tearing and partitioning as methods of solution are made:

a) For partitioning the full matrix must be available.

b) The same amount of matrix inversion is required by the two methods, but the matrix multiplications associated with partitioning far exceed those of tearing.

c) If some or all of the subdivisions of a network are identical, fewer operations are required to solve by tearing.

The above three points arise out of the analysis of linear systems;

for systems of mixed linear and non-linear components a fourth point could be added:

d) For repeated iterations only the non-linear components have to be inverted more than once.

Further justification for the use of the methods of tearing is outlined by Brameller (20):

e) Existing digital computers can be used to solve larger scale problems than before.

f) For large problems computation time may be reduced.

g) Alterations to a subdivision may be effected more easily, the overall equations do not have to be changed.

h) The necessity of using matrices greatly helps computation since matrix representation in digital machines is easily achieved and special routines may be used for their handling.

2.6.4 The Work of Happ

In a major work (24), Happ has influenced the development of the theory of pipe networks of this thesis by simplification of the application of Kron's orthogonal approach. Happ's theory has been developed for electrical resistance networks, but the use of simple matrices has made this development easy to understand and apply to fluid networks. It is not proposed to discuss in detail the contributions of Happ at this stage, since Chapter (3) contains most of the theoretical developments of Happ. Their application to fluid networks is pointed out as they are encountered.

2.6.5 The Work of Middleton

Middleton (6) was able to show that there are advantages to be gained by using the diakoptics approach to network analysis. Using the network describing matrices $\underline{A}$ and $\underline{C}$ he was able to show that the two classical methods of nodal and mesh could be derived with no reference to diakoptics. Neither of these two methods was applied to pipe network problems and programmed. This is unfortunate since any comparison of the methods of analysis of pipe network problems ought to include those methods available. However from his theoretical derivation of the classical methods, Middleton produced one solution equation of diakoptics. This was achieved by the construction of an all-mesh or an all-node-to-datum network, which involves the addition of fictitious branches to generate square transformation matrices. The square transformation matrices developed were thought to have the same meaning as $\underline{A}$ and $\underline{C}$. It is shown in Chapter (3) that the square transformation matrices may be formed directly from the existing network, and that in fact $\underline{A}$ and $\underline{C}$ are only constituent parts of two square non singular transformation matrices $\underline{\alpha}$ and $\underline{\gamma}$.

Middleton successfully derives the solution equations of diakoptics and applies the method to pipe network problems by constructing a computer program. This program together with another using the Hardy Cross mesh method was developed for an Elliot 803 machine with backing store.

He showed that under the condition of maximum overlap, the Hardy Cross mesh method when used to solve a test network which he had devised (called TEST 1 in this thesis), did not converge to a solution after two hours, whereas the diakoptic program always converged to a solution in the same number of iterations and in less time, no matter what overlap condition was chosen.

The work (6) includes a comparison of the results available in the literature at that time. The results of the analysis of other authors for three networks are shown. Middleton says that any comparison with his own work in terms of efficiencies of methods is difficult because of a lack of available information. The major significance of the comparison, apart from proving the validity of the diakoptic method, appears to have been a justification for the use of the Colebrook equation for determining friction factors. This gave results in terms of individual pipe flows using the diakoptics program which were in good agreement with the published data.

A network analysed by Middleton but not used in this work because better test networks have been devised, was due to Hunn and Ralph. This shows admirably one of the special facilities of the diakoptics approach, which is that a river which may be the source point of a variety of pipes may be taken as a common datum.

The choice of cutting pattern was also investigated and it was shown that the number of iterations was the same for each choice, but the total computational time however varied, and Middleton proposes the rule of thumb, that the subdivisions should have an equal number of nodes with a minimum of cut branches, to minimise the total computational time.

It was also shown that the effect of initial guess of flows had little effect on the total number of iterations required for solution.

An investigation of the effect of the removal of branches of the network produced the expected result that certain branches played a more important role than others. Starting from a converged solution, a new solution was achieved after the removal of one branch after three iterations, whilst after the removal of an alternative branch a solution was achieved after eight iterations, the same number as were required for the original solution. A similar experience was recorded for changes in inputs and outputs to a system.

Middleton concludes that the nodal diakoptics method is at least as efficient as the Hardy Cross method, but more efficient in terms of storage. The data are easier to compile for diakoptics. The changes in the geometry of a network can be easily handled by diakoptics, whereas a whole new data preparation has to be carried out for the Hardy Cross method. No choice of initial flows has to be made by a user, since because the method is insensitive to the choice, the machine can be instructed to assign initial flows.

An extension of the theoretical treatment of the problem shows how the equations of diakoptics may be used to check the specification of mixed unknown and known properties during design analysis.

It is shown later in this thesis that the situation of mixed unknown and known nodal pressures and demands or supplies may be easily written in into a computer program.

The work of this thesis is an extension of the work of Middleton. Different concepts are used, and it is believed that the concepts of diakoptics have been applied with greater understanding.

2.7 Summary of 2.2 to 2.6

It is apparent from the preceding discussion that there are two alternative approaches to the solution of network problems. One, the Hardy Cross method favoured by chemical engineers. It has been shown that this method has been modified by some workers to improve the convergence of solution, limit the amount of data handling, and apply the method to modern computers, by the implementation of some simple matrix techniques. However, the method still requires large amounts of storage space, and Middleton (6) reports that under the condition of maximum overlap, the method does not guarantee a solution. In the test network shown in figure (5.2.1) for instance, there is a choice of about 350 million trees, some of which may give convergent conditions.

The alternative approach, the orthogonal method of Kron, is relatively untried in the discipline of chemical engineering, apart from the practical application of Middleton (6), and the theoretical developments of Brameller (38) and Fincham (42). Other analysts suggest that this alternative approach has many advantages over the Hardy Cross method, the equations of solution are self generating, data preparation is reduced to a minimum, convergence is guaranteed, compatability with modern computers is of the first order, convergence is not dependent on initial guesses of solution, and most importantly, a whole new range of equations called diakoptics can be developed from the basic theory which enables engineers to tackle problems previously outside their scope, and to see the whole spectrum of network analysis from an entirely different viewpoint.

The advantages claimed for this alternative approach, and its apparent conceptual simplicity, suggested that further investigation of the application of electrical theory to pipe network problems, which led immediately to a study of linear graph theory, matrix representation, tensor transformation and diakoptics, would produce theories which would promote a better understanding of the design and analysis of pipe network problems, but at the same time be a computationally useful tool.

The next chapter describes this investigation.

CHAPTER 3

THE THEORY OF PIPE NETWORK ANALYSIS

3.1 Introduction

The development of a series of equations which permit the solution of both large and small scale problems is presented. The equations, which result from using an electrical analogy for fluid flow, are derived using Kron's (11) orthogonal network concepts of the existence, within the network, of node to datum and mesh paths.

A new equation of diakoptics is presented which gives a route for an entirely different method of solution, which does not appear to have been published elsewhere.

It is further shown that the inclusion of pump terms, other non pipe elements and pressure specified nodes may also be taken into account with little extension of the basic theory.

The chapter commences with a discussion of basic topological and graphical network theory, leading to a discussion of the integer matrix representations of graph theory, being based on the work of Veblen (25) and Branin (16). The electrical analogy for fluid flow is established which leads to the discussion of orthogonal network theory. The concept of tensor transformation favoured by electrical engineers is dispensed with, tensors being treated as transformation matrices. The matrices of the orthogonal network are then related to the matrices of graph theory.

It is then shown how the classical nodal and mesh methods developed by Roth (14) may be set up. The network equations are then expanded to include the concept of tearing. There are two approaches to this problem, each resulting in a different method of solution.

Throughout the theoretical development, computational aspects are pointed out, showing that the orthogonal concepts are not only powerful numerical techniques, but that they also lead to a different understanding of network problems which will help engineers to tackle the problems of model building, design and the study of large scale network problems.

3.2 Topological Concepts

The analysis of any network system requires the knowledge of some topological concepts. A network of pipes may be represented by a graph which has no physical properties associated with it. Figure (3.2.1) shows the graph which is a schematic representation of a pipe network that is used later to show worked examples of the theoretical developments of this chapter.

Each pipe of the network is represented by a branch of the graph and these branches are assumed to be connected in the same way as the pipes of the network. Every branch is assumed to run between two junction points or nodes.

Every graph must consist of the two elements:

(a) branches

(b) nodes

and may also contain the following elements which are discussed more fully below:

(c) subgraphs

(d) meshes

(e) node to datum paths

If a path can be traced out through all the nodes, travelling along branches only, then the graph is a connected graph. However a connected graph may itself be a subgraph of a larger unconnected graph.

A property of a connected graph is that it contains at least one tree, a tree being any path traced out within a connected graph, through all of the nodes, which forms no closed loops or meshes (ie the traced path does not pass through the same node twice). A mesh is generally referred to as a collection of branches within a connected network which forms a closed loop. However, a more precise definition of a mesh is specified for use in this thesis. A mesh is a series of branches within a connected graph which contains one unique non tree branch (called a link branch) and sufficient tree branches to make up a closed loop. The mesh path, a term which is used later in this thesis, refers to the path traced out through the graph by the branches which make up a particular mesh. A connected graph therefore contains as many meshes as there are link branches, and the same number of tree branches minus one as there are nodes.

Once the tree of a graph has been selected the meshes are automatically generated, or alternatively, if the link branches of the meshes of a graph have been selected, then the tree branches must be automatically generated.

A further concept necessary to the discussion of graph theory is that if each branch has assigned to it a positive direction, then the graph is said to be directed. It is also assumed that there is a direction associated with each mesh of the graph, and by convention the direction of the mesh is assumed to be that of its defining link.

The dual of the mesh path is the node to datum path which is associated, for the purpose of this work, with tree branches only. The node to datum path, as suggested by its name, is a path traced out from any node through to the datum node (which for the time being is any node so selected), and is automatically generated once the tree of the graph is chosen. There are therefore, as many node to datum paths as there are non datum nodes.

Figure (3.2.1) shows the graph of a network which has 12 branches and 9 nodes. Each branch has assigned to it a direction and both branches and nodes have individual reference numbers. The tree branches are shown on the graph by the heavier lines.

Figure (3.2.2) shows the same graph which has 4 meshes and associated

Figure (3.2.1)

The graph of a network showing a tree

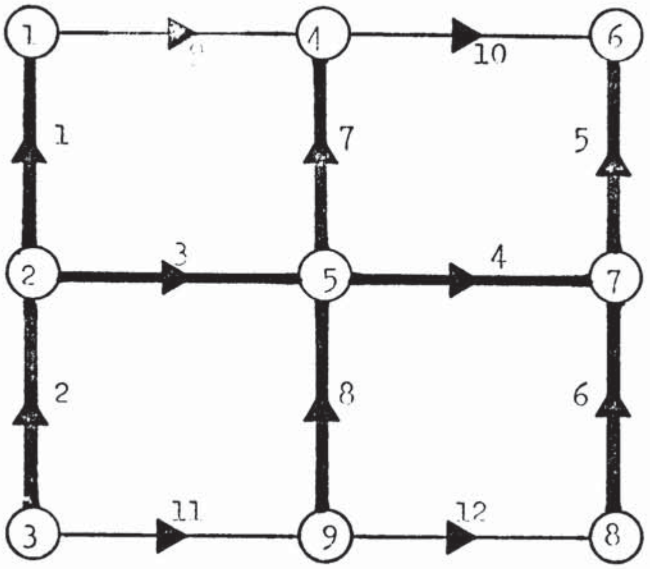
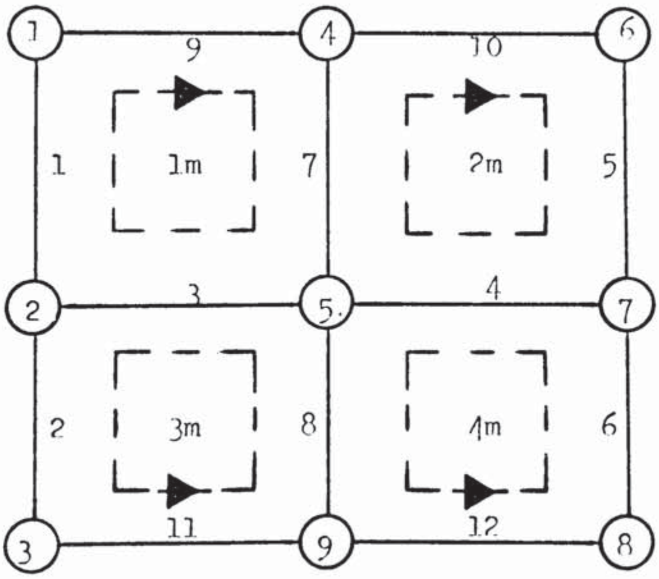


Figure (3.2.2)

The graph of a network showing meshes

corresponding to the tree of figure (3.2.1)



with each, a direction.

A summary of the relationships between the elements of the graph of a connected network has been developed by Veblen (25) and is shown below:

Meshes = Branches - Non-datum nodes

Node to Datum Paths = Non-datum nodes

Tree Branches = Non-datum nodes

Link Branches = Meshes

3.3 Matrix Relationships of a Directed Connected Graph

Five matrices, $\underline{A}$, $\underline{B}$, $\underline{C}$, $\underline{D}$ and $\underline{F}$, which describe a connected directed graph can be established. An important consideration of this chapter is not only the definition of these matrices, but also the way in which they are interrelated, and how each may be established from a simple connection list. This connection list which is discussed more fully at a later stage, contains the basic information about the branch-node interconnections.

The computer programs which have been developed use the five matrices to a large extent, and therefore their size, and the speed at which they may be manipulated, is of paramount importance. The five matrices contain only +1s, -1s and 0s, and since they are usually sparse, special techniques may be implemented for their manipulation.

In order to establish the matrices it is helpful to assign each branch, node and mesh a reference number. In figure (3.2.1) the graph has the branches numbered 1 to 12, the nodes numbered 1 to 9 and the meshes numbered 1 to 4.

3.3.1 The incidence matrix $\underline{A}$

The incidence matrix $\underline{A}$ is derived from the augmented incidence matrix $\underline{A}^a$ which describes the relationship between the nodes and branches of the graph. $\underline{A}^a$ has the dimensions branches by nodes (ie the rows correspond to branches and the columns to nodes). Each row of $\underline{A}^a$ contains a +1, a -1, and 0s such that the a^a (k,j) element contains a +1, -1 or 0 if the kth branch is positively, negatively or not directed towards the jth node. Since each row of $\underline{A}^a$ contains one +1 and one -1, the sum of the elements in each row is zero and the columns are not linearly independent, and one column may be deleted. It is convenient for reasons of storage to delete the column corresponding to the node of the highest reference number. This node is henceforth referred to as the datum node. The matrix formed as a result of this deletion is called the incidence matrix $\underline{A}$. The rows of $\underline{A}$ may be partitioned into tree and link branches as in equation (3.3.1)

$$\underline{A} = \left|\begin{array}{c} \underline{A}_T \\ \hline \underline{A}_L \end{array}\right. \qquad (3.3.1)$$

and figure (3.3.1) shows the $\underline{A}$ matrix in this state for graph of figure (3.2.1). The partitioning of $\underline{A}$ is most easily achieved if the tree branches have reference numbers starting at 1, through ND where ND is the number of tree branches, the link branch reference numbers being assigned from ND + 1 through ND + M where M is the number of link branches.

3.3.2 The node to datum path matrix $\underline{B}$

The inverse of $\underline{A}_T$ is related to the node to datum path matrix $\underline{B}_T$, which is the tree partitioned part of another matrix $\underline{B}$. The $\underline{B}$ matrix describes the relationship between the node to datum paths and the branches of the graph, and has the dimensions branches by node to datum paths. Each row of the matrix may contain +1s, -1s or 0s, such that the b (k,j) element contains a +1, -1 or 0 if the kth branch is positively, negatively or not included in the jth node to datum path. The positive direction of the node to datum path is away from the datum node. The requirement that node to datum paths are limited to tree branches allows the $\underline{B}$ matrix to be partitioned as in equation (3.3.2), Figure (3.3.1) The A matrix of the graph of figure (3.2.1)

$$
\begin{array}{c|cccccccc}
 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\underline{A}_T & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\underline{A}_L & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

Figure (3.3.2) The B matrix of the graph of figure (3.2.1)

$$
\begin{array}{c|cccccccc}
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
\underline{B}_T & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\underline{B}_L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

$$\underline{B} = \left|\begin{array}{c}\underline{B}_T \\ \hline \underline{B}_L\end{array}\right. = \left|\begin{array}{c}\underline{B}_T \\ \hline \underline{0}_L\end{array}\right. \qquad (3.3.2)$$

giving a null sub-matrix $\underline{0}_L$ of dimensions meshes by nodes. An example of $\underline{B}$ in its partitioned form is given in figure (3.3.2). The relationship between $\underline{A}_T$ and $\underline{B}_T$, which has been shown by Happ (24), is given by equation (3.3.3)

$$\underline{A}_T^{-1} = \breve{\underline{B}}_T \qquad (3.3.3)$$

The use of the $\underline{A}$ and $\underline{B}$ matrices, their storage, handling and setting up in computer programs, is discussed in appendix (1).

3.3.3 The branch-mesh incidence matrix $\underline{C}$

Since most graphs contain more than one mesh, it is convenient to have available a further connection matrix called the branch-mesh incidence matrix $\underline{C}$. The $\underline{C}$ matrix contains the information which indicates the way in which particular branches make up the meshes of the graph, and has the dimensions branches by meshes. Each column of $\underline{C}$ contains +1s, -1s and 0s, such that the c (k,j) element contains a +1, -1 or 0 if the kth branch is positively, negatively or not included in the jth mesh. Like the $\underline{A}$ and $\underline{B}$ matrices, $\underline{C}$ may also be partitioned into tree and link parts as in equation (3.3.4),

$$\underline{C} = \left|\begin{array}{c}\underline{C}_T \\ \hline \underline{C}_L\end{array}\right. = \left|\begin{array}{c}\underline{C}_T \\ \hline \underline{U}_L\end{array}\right. \qquad (3.3.4)$$

This produces the result that the submatrix $\underline{C}_L$ can be shown to be a unit matrix, which is achieved if the referencing of the meshes is related to the referencing of the link branches. It has already been indicated that the link branches have been referenced with the highest numbers. The unit matrix $\underline{C}_L$ is produced if that mesh referenced as mesh 1 has as its link branch the lowest referenced link branch. The remaining link branches in ascending order of reference numbers then determine the reference numbers of the remaining meshes. The algorithm which defines the referencing procedure is shown below.

Mesh Reference (k) = Link Branch Reference (k) - Non-datum Nodes

$k = 1,2 \; . \; . \; . \; M$

The value of this overall procedure is that the unit matrix never has to be stored by the computer, and only information about the tree branches has to be handled, but it is only achieved by careful referencing of branches and meshes. It is shown in appendix (1) that the formation and referencing of the meshes may be performed by a computer program.

Figure (3.3.3) shows the $\underline{C}$ matrix in partitioned form for the graph of figure (3.2.1).

Equations (3.3.5) and (3.3.6) show the readily proved (24) relationship between $\underline{A}$, $\underline{B}$ and $\underline{C}$.

$$\underline{\breve{A}}.\underline{C} = \underline{\breve{C}}.\underline{A} = 0 \qquad (3.3.5)$$

$$\underline{C}_T = -\underline{B}_T \, \underline{\breve{A}}_L \qquad (3.3.6)$$

Equation (3.3.6) is particularly useful, since it implies that $\underline{C}_T$ may be made available to a computer program by inputting the branch-node connection list and set up by a matrix multiplication, greatly reducing the amount of input required by a program. The full matrix multiplication shown in equation (3.3.6) is not required since special techniques which take advantage of the sparsity of $\underline{A}_L$ may be used, as is shown in appendix (1).

3.3.4 The cutset matrix $\underline{D}$

The $\underline{D}$ matrix is not commonly used by network theorists but since it gives rise to an alternative way of solving network problems (19) as outlined in appendix (2) its inclusion is necessary. The $\underline{D}$ matrix describes the relationship between the cutsets of a graph and the branches. A cut-set is defined as a collection of branches which, when cut across their axes, produce, from a connected graph, two disconnected subgraphs. In the ultimate case a single branch cut across its axis may also produce a cutset and two subgraphs, one of which will contain only one node.

Each cutset is assumed to be associated with a unique tree branch just as each mesh is associated with a unique link branch. Cutsets may be defined by enclosing the non-datum terminal node of a node to datum path by a circle. Any link branches cut by this circle together with a single tree branch belong to the same cutset. The tree branch which defines the cutset is assumed to be that tree branch which is the first tree branch in the node to datum path that originates at the node which has been encircled. Each cutset also has a positive direction associated with it, and each cut link is assumed to have an orientation to the cutset. The orientation of each cut link is assumed to be positive or negative depending on whether its direction relative to the encircled node is the same as or oppositive to the defining tree branch. That is the orientation of the cutset itself is independent of the direction of the tree branch relative to the encircled node. The $\underline{D}$ matrix for the graph of figure (3.2.1) is shown in partitioned form in figure (3.3.4), and for example if node 1 of figure (3.2.1) is enclosed by a circle, tree branch 1 is cut and becomes the cutset defining branch. Link branch 9 is also cut and since its direction is opposite to that of the tree branch, the entry in $\underline{D}$ is -1.

The $\underline{D}$ matrix has the same dimensions as $\underline{A}$ and may be partitioned into tree and link parts as in equation (3.3.7).

$$\underline{D} = \left| \begin{array}{c} \underline{D}_T \\ \hline \underline{D}_L \end{array} \right. = \left| \begin{array}{c} \underline{U}_T \\ \hline \underline{D}_L \end{array} \right. \qquad (3.3.7)$$

The partitioning produces the unit matrix $\underline{U}_T$, which serves to explain more fully the concept of one tree branch being associated with each cutset. The submatrix $\underline{D}_L$ is related to $\underline{C}_T$ as shown in equation (3.3.8).

$$\underline{D}_L = -\breve{\underline{C}}_T \qquad (3.3.8)$$

Figure (3.3.3) The $\underline{C}$ matrix of the graph of figure (3.2.1)

$$\begin{array}{c|cccc}
 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & -1 & 0 \\
 & -1 & 0 & -1 & 0 \\
 & 0 & -1 & 0 & -1 \\
\underline{C}_T & 0 & -1 & 0 & 0 \\
 & 0 & 0 & 0 & 1 \\
 & -1 & 1 & 0 & 0 \\
 & 0 & 0 & 1 & -1 \\
\hline
 & 1 & 0 & 0 & 0 \\
 & 0 & 1 & 0 & 0 \\
\underline{C}_L & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 1
\end{array}$$

Figure (3.3.4) The $\underline{D}$ matrix of the graph of figure (3.2.1)

$$\begin{array}{c|cccccccc}
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\underline{D}_T & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \\
\underline{D}_L & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 \\
 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1
\end{array}$$

This relationship implies that any $\underline{D}$ matrix required by a computer program may be set up within the machine, rather than input by a user. This is further discussed in appendix (1).

3.3.5 The link mesh matrix $\underline{F}$

The matrix has the same dimensions as $\underline{C}$ and has the function of defining the meshes within the graph of a network in terms of the link branches only. Since link branches cannot be tree branches and vice versa, the form of $\underline{F}$ is very simple. The $\underline{F}$ matrix for figure (3.2.1) is shown in figure (3.3.5), in its partitioned form. A helpful way of visualising $\underline{F}$ is to see it as the dual of $\underline{B}$, where $\underline{B}$ defines node to datum paths within the graph in terms of tree branches only. The result is that $\underline{F}$ takes the form given by equation (3.3.9).

$$\underline{F} = \left|\begin{array}{c} \underline{F}_T \\ \hline \underline{F}_L \end{array}\right. = \left|\begin{array}{c} \underline{0}_T \\ \hline \underline{F}_L \end{array}\right. = \left|\begin{array}{c} \underline{0}_T \\ \hline \underline{U}_L \end{array}\right. \qquad (3.3.9)$$

3.3.6 The compound matrices $\underline{\gamma}$ and $\underline{\alpha}$

Two square matrices which are fundamental to the concepts of the orthogonal network theory, and which are combinations of the $\underline{A}$, $\underline{B}$, $\underline{C}$ and $\underline{F}$ matrices described above, may be set up for the graph of any network. The $\underline{\gamma}$ matrix which relates the paths of a network, both node to datum and mesh, to the branches is a combination of the $\underline{B}$ and $\underline{C}$ matrices. It has the dimensions branches by branches and may be partitioned as shown in equation (3.3.10).

$$\underline{\gamma} = \begin{array}{c|c} \underline{\gamma}_{TT} & \underline{\gamma}_{TL} \\ \hline \underline{\gamma}_{LT} & \underline{\gamma}_{LL} \end{array} = \begin{array}{c|c} \underline{B}_T & \underline{C}_T \\ \hline \underline{0}_L & \underline{U}_L \end{array} = \underline{B} \,\big|\, \underline{C} \qquad (3.3.10)$$

The $\underline{\gamma}$ matrix which represents the graph in figure (3.2.1) is shown in its partitioned form in figure (3.3.6).

The $\underline{\alpha}$ matrix which is also square and has the dimensions branches by branches may be thought of as a matrix which describes for the overall graph, the incidence of the branches at the nodes and in the meshes. It

Figure (3.3.5) The F matrix of the graph of figure (3.2.1)

$$\begin{array}{c|cccc}
 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 \\
\underline{F}_T & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 \\
\hline
 & 1 & 0 & 0 & 0 \\
 & 0 & 1 & 0 & 0 \\
\underline{F}_L & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 1
\end{array}$$

Figure (3.3.6) The γ matrix for the graph of figure (3.2.1)

$$\begin{array}{cccccccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
-1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & -1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

$$\underline{\gamma} = \begin{array}{c|c} \underline{\gamma}_{TT} & \underline{\gamma}_{TL} \\ \hline \underline{\gamma}_{LT} & \underline{\gamma}_{LL} \end{array}$$

Figure (3.3.7) The $\underline{\alpha}$ matrix for the graph of figure (3.2.1)

$$\begin{array}{rrrrrrrr|rrrr}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
-1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}$$

$$\underline{\alpha} = \begin{array}{c|c} \underline{\alpha}_{TT} & \underline{\alpha}_{TL} \\ \hline \underline{\alpha}_{LT} & \underline{\alpha}_{LL} \end{array}$$

is a combination of the $\underline{A}$ and $\underline{F}$ matrices and may be partitioned into four submatrices as shown in equation (3.3.11).

$$\underline{\alpha} = \begin{array}{c|c} \underline{\alpha}_{TT} & \underline{\alpha}_{TL} \\ \hline \underline{\alpha}_{LT} & \underline{\alpha}_{LL} \end{array} = \begin{array}{c|c} \underline{A}_{T} & \underline{0}_{T} \\ \hline \underline{A}_{L} & \underline{U}_{L} \end{array} = \underline{A} \,|\, \underline{F} \qquad (3.3.11)$$

The $\underline{\alpha}$ matrix for the graph of figure (3.2.1) is shown in figure (3.3.7).

3.4 The Electrical Analogy

Water and gas distribution networks have an obvious electrical analogue, and it is advantageous to discuss the solution of these types of network problems in electrical terms for two reasons.

These are that most of the research into, and the application of, modern solution methods of network analysis has been conducted by electrical engineers. Therefore the discussions available in published form are almost wholly conducted in electrical engineering terminology. Secondly, the simplicity of linear electrical resistance networks is understood by engineers of all disciplines.

It is assumed that voltage, current and resistance are respectively the analogue of pressure, fluid flow and hydraulic resistance. For fluid flow a simple linear relationship between pressure and flow is only possible at low Reynolds Numbers. At higher Reynolds Numbers the hydraulic resistance is a function of flow and friction factor.

The physical properties associated with the branches of a network may be analysed by first considering a single branch only. It is assumed that this branch may consist of the following three elements:

a) an admittance or coil Y;

b) a voltage source E, which is analogous to a compressor in a pipe network;

c) a current source I which is analogous to a demand or supply to a pipe network.

A branch containing these elements is shown in figure (3.4.1).

The equation relating the properties associated with the coil is equation (3.4.1),

$$J = Y*V \tag{3.4.1}$$

or alternatively equation (3.4.2),

$$V = Z*J \tag{3.4.2}$$

where Y and Z are respectively the admittance and impedance of the coil, and J and V are respectively the current flowing in the coil and the potential across the coil.

The voltage V may be related to the potential across the extremities of the branch, denoted by e, and the potential developed by the active voltage source E. If both E and e are orientated as shown in figure (3.4.2), then the relationship for V is given by equation (3.4.3),

$$V = E + e \tag{3.4.3}$$

The current J may be likewise expressed as the sum of two individual currents. One of these, I, is assumed to be due to the external current source. In the fluid flow case I may be assumed to be the current component in the branch due to the flows entering or leaving the branch which are demands on, or supplies to, the overall network. The other component which makes up J is i. This is assumed to be a component of current in the branch due to all other causes other than I. For instance it may be due to voltage sources or the way in which the branch is interconnected within a network. The resulting equation for J, provided the two components I and i are assumed to flow as in figure (3.4.2), is given by equation (3.4.4),

$$J = I + i \tag{3.4.4}$$

The single branch of figure (3.4.2), excited by a current source I, will have J made up of one current component I only. If, on the other hand, only a potential source E is applied across the extremities of the branch, J will be made up of i only, but for this to arise the branch must

Figure (3.4.1) A single branch of in a primitive state containing a voltage generator, current generator and a coil

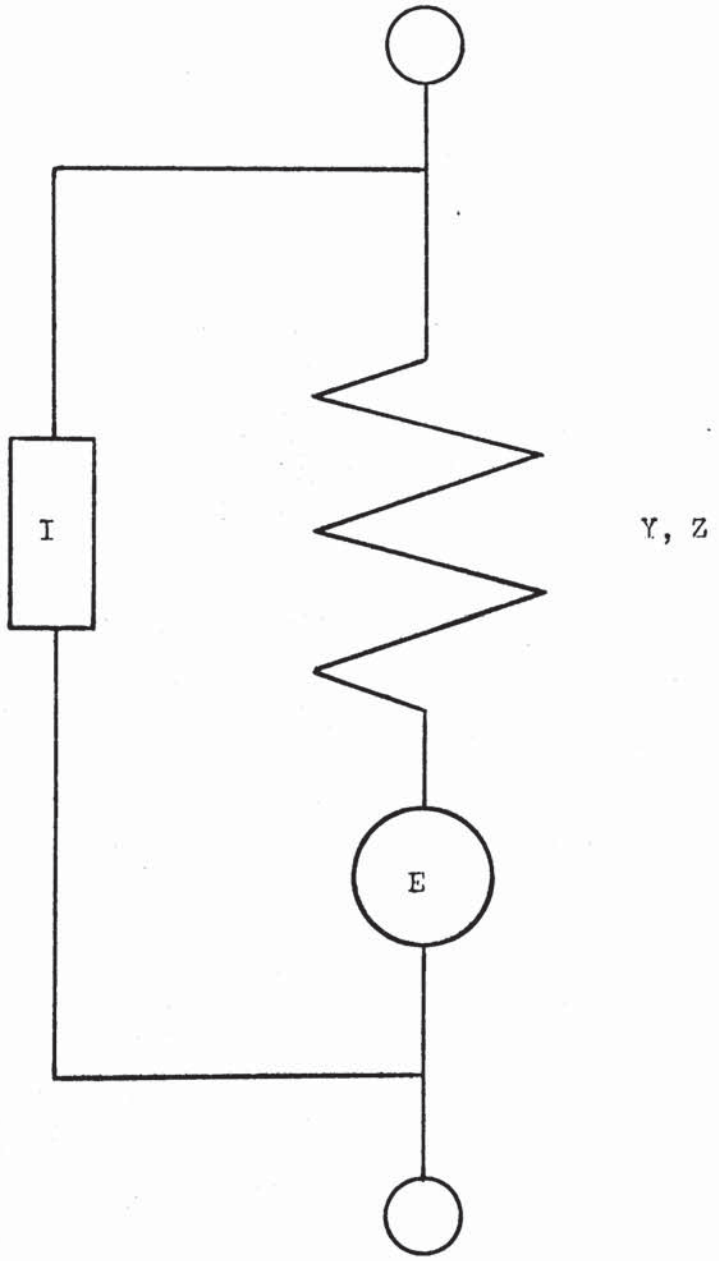
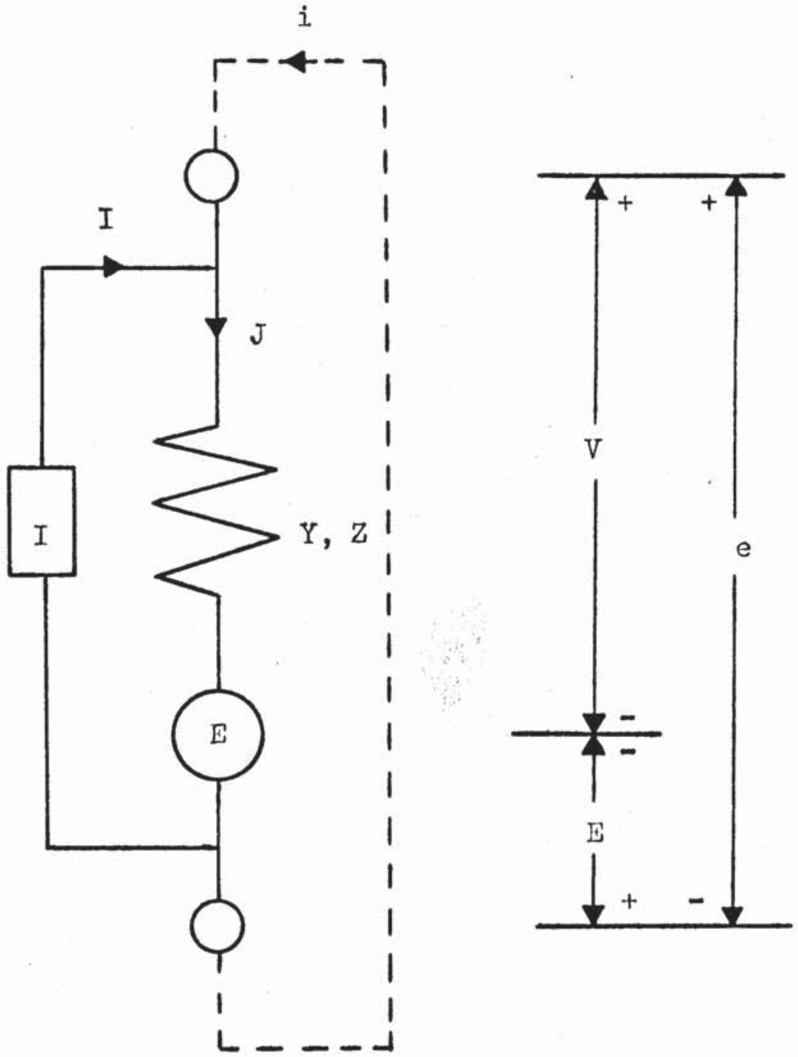


Figure (3.4.2) A primitive branch showing the orientation of E, I, e and i



be connected within the network, since a return path must be available for the i component to flow.

Alternative representations of the components of current and voltage are discussed later and the significance of i and I in a single branch will become more obvious at that stage.

Equations (3.4.1) and (3.4.2) can now be expressed as equations (3.4.5) and (3.4.6),

$$E + e = Z\,(I + i) \tag{3.4.5}$$

$$I + i = Y\,(E + e) \tag{3.4.6}$$

These equations are now used to interrelate the physical properties of the connected network and those of another network, the primitive network.

3.5 Transformation Matrices and their relation to Graph Theory

The primitive network may be considered to be a "torn model" of a connected network. It is a network which consists of as many unconnected or single branches as exist in the connected network. Each branch of the primitive network is assumed to be identical to the single branch discussed in Chapter (3.4).

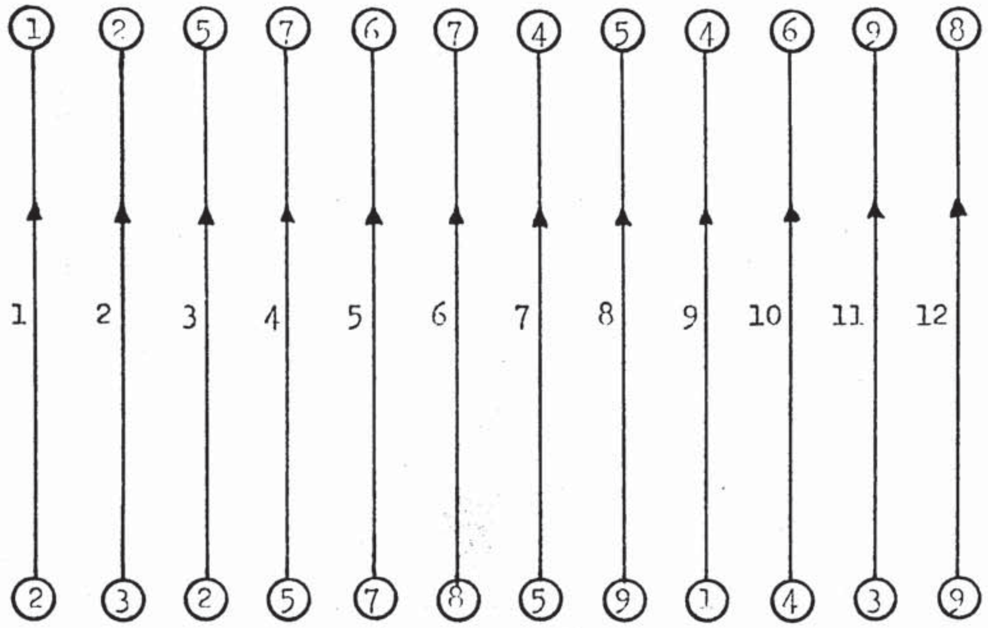
The graph of the primitive network representing the connected network of the graph of figure (3.2.1) is shown in figure (3.4.3). It is next shown that it is possible to transform variables associated with the primitive network to those associated with the connected network, and vice versa.

Having established the electrical analogy equations for a network, equations (3.4.1) and (3.4.2) expressed in vector form are assumed to represent the relationship of flow, pressure and resistance associated with the b branches of the primitive network, given by equations (3.5.1) and (3.5.2).

$$\underline{V} = \underline{Z}.\underline{J} \tag{3.5.1}$$

$$\underline{J} = \underline{Y}.\underline{V} \tag{3.5.2}$$

Figure (3.4.3) The graph of the primitive network representing the network of the graph of figure (3.2.1)



A further set of b equations in vector form represents the relationships of the variables associated with the b branches of the connected network; these equations are (3.5.3) and (3.5.4).

$$\underline{V}c = \underline{Z}c.\underline{J}c \qquad (3.5.3)$$

$$\underline{J}c = \underline{Y}c.\underline{V}c \qquad (3.5.4)$$

where the subscript c indicates that the equations are equations of the connected network.

There is a link between variables such as $\underline{J}$ and $\underline{J}c$. It is assumed that there exists a device $\underline{\gamma}$ for transforming the quantities associated with the connected network to those of the primitive network, such that the relationship is given by equation (3.5.5)

$$\underline{J} = \underline{\gamma}.\underline{J}c \qquad (3.5.5)$$

and that the device $\underline{\gamma}$ has a reverse role such that the relationship given by equation (3.5.6) also exists,

$$\underline{J}c = \underline{\check{\alpha}}.\underline{J} \qquad (3.5.6)$$

ie the inverse of $\underline{\gamma}$, $\check{\alpha}$ transforms variables associated with the primitive network to those associated with the connected network. The relationship between $\underline{\gamma}$ and $\underline{\alpha}$ is given by equations (3.5.7) and (3.5.8),

$$\underline{\gamma}^{-1} = \underline{\check{\alpha}} \qquad (3.5.7)$$

$$\underline{\alpha}^{-1} = \underline{\check{\gamma}} \qquad (3.5.8)$$

which imply that both $\underline{\gamma}$ and $\underline{\alpha}$ are square non singular transformation matrices.

In electrical terms the power requirement imposed upon a transformation of the sort outlined above, that the power input or dissipated remains invariant under transformation of variables, has been shown by Happ (24) to give the result expressed by equation (3.5.9),

$$\underline{\check{\alpha}}.\underline{\gamma} = \underline{U} \qquad (3.5.9)$$

The variables $\underline{Z}c$ and $\underline{Z}$ may also be transformed. Premultiplication of equation (3.5.1) by $\underline{\check{\gamma}}$ and substitution for $\underline{J}$ from equation (3.5.5) gives equation (3.5.10),

$$\check{\underline{\gamma}}.\underline{V} = \check{\underline{\gamma}}.\underline{Z}.\underline{\gamma}.\underline{Jc} \qquad (3.5.10)$$

It is assumed that the left hand side of (3.5.10) may be re-expressed to give equation (3.5.11),

$$\underline{Vc} = \check{\underline{\gamma}}.\underline{Z}.\underline{\gamma}.\underline{Jc} \qquad (3.5.11)$$

with the result that an expression for $\underline{Z}$c in terms of $\underline{Z}$ is given by equation (3.5.12),

$$\underline{Z}c = \check{\underline{\gamma}}.\underline{Z}.\underline{\gamma} \qquad (3.5.12)$$

The equivalent expression for $\underline{Y}$c may also be easily derived to give equation (3.5.13),

$$\underline{Y}c = \check{\underline{\alpha}}.\underline{Y}.\underline{\alpha} \qquad (3.5.13)$$

The whole procedure for both $\underline{Z}$c and $\underline{Y}$c may also be shown in reverse to transform variables associated with the connected network to those of the primitive network. In the next section it is shown how the two devices $\underline{\gamma}$ and $\underline{\alpha}$ are used in network theory and how they are related to the matrices of graph theory described in chapter (3.3.6).

3.6 Orthogonal Network Theory

Two states of a network have been described, the connected and the primitive. A further state, called the orthogonal state by Kron (5), is introduced from which the solution equations to network problems may be established.

The orthogonal network has the same graph as the connected network and the two networks are therefore equivalent. The way in which the current and voltage quantities are represented in the orthogonal network is however different from their representation in the connected network. Conventional network theory usually regards the branch or the node as the focal point of attention, orthogonal network theory is concerned with series of branches called paths which exist within the network. Two types of path are assumed to exist simultaneously. They are the open and the closed path. It is shown later that these two sets of paths are respectively identical to the node to datum and mesh paths of the graph of a network, but the nomenclature of open and closed path is retained for the time being since network variables are being discussed.

The current and voltage quantities, I, i, E and e are associated in a specific manner with the open and closed paths; assumptions are made which restrict currents and voltages to certain paths within the network. Happ (24) defines the two types of path in the very widest sense and for a more detailed account of the significance of these paths the reader should refer to his work. This thesis is restricted to a more precise definition of the two types of path, which simplifies the derivation of solution equations and is more easily understood.

3.6.1 The open paths of a network

In the widest sense these are paths in the network which form no closed loops and they can be made up of tree and link branches, and are paths along which properties may act from one node to another. For the purpose of this work however, the open paths are restricted to the tree branches of the graph of the network and are therefore identical to the node to datum paths of chapter (3.3.2). In this way it is possible to use the describing matrices set up for the graph of a network to relate properties associated with the branches or pipes of the actual network, to the other components of the network. The closed paths are similarly defined.

3.6.2 The closed paths of a network

In the widest sense a closed path is made up of any series of branches within a network which forms a closed loop, along which a property may act. For the purpose of this work however, the closed paths are restricted to the single defining link branches and their associated tree branches of the graph of the network, and are therefore identical to the mesh paths of chapter (3.3.3).

The devices $\underline{\gamma}$ and $\underline{\alpha}$ of chapter (3.5) were shown to be transforming devices which were able to interrelate variables associated with two types of network, the primitive and the connected, but with no indication of their structure. By looking at the connected network as a network which is made up of a combination of two types of path it is possible to show that the $\underline{\gamma}$ and $\underline{\alpha}$ of chapter (3.5) are equivalent to the $\underline{\gamma}$ and $\underline{\alpha}$ of chapter (3.3.6). This concept of using the graph describing matrices of chapter (3.3.6) to interrelate the network variables of chapter (3.4) is of the utmost importance in the understanding of orthogonal network theory.

Orthogonal network theory is the study of the variables associated with the two types of path. Consider the superimposition of current variables on the connected network. The current $\underline{J}c$ is made up of two components, $\underline{I}c$ and $\underline{i}c$; both of these components are assumed to flow in each branch, with no mention of a particular current component being confined to a particular branch or set of branches. In the orthogonal network current components $\underline{I}'$ and $\underline{i}'$ are associated with the sets of branches which make up the two types of path and are therefore not branch properties but path properties. $\underline{I}'$ is associated with the node to datum paths and $\underline{i}'$ is associated with the mesh paths. A similar assumption is made for the voltage components.

The next two chapters describe the definition and significance of these path components.

3.7 Superimposing Current Components on the Node to Datum and Mesh Paths of a Network

The physical significance of the superimposition of current components on an orthogonal network will be established.

The current vector $\underline{I}'$ which is imposed on the node to datum paths represents the current in the network due to the external current sources, or in the case of fluid flow, the demands and supplies of the network. These currents within the network are assumed to be the external currents which enter the network at a particular node and flow through the network to the datum node, along the node to datum path associated with the entry node. None of these currents can therefore appear in the link branches. There will be the same number of node to datum currents as there are node to datum paths, but some of the currents may have zero value. For example, superimposition of four node to datum currents on the network shown in figure (3.7.1) shows that although the network contains eight tree branches there are only seven of these utilised to carry node to datum flows. Since there is no demand or supply at node (4), branch 7 has a zero flow, and similarly since there is no demand or supply at the datum node (node (9) in figure (3.7.1)), the sum of the flows in branch 8 is also zero.

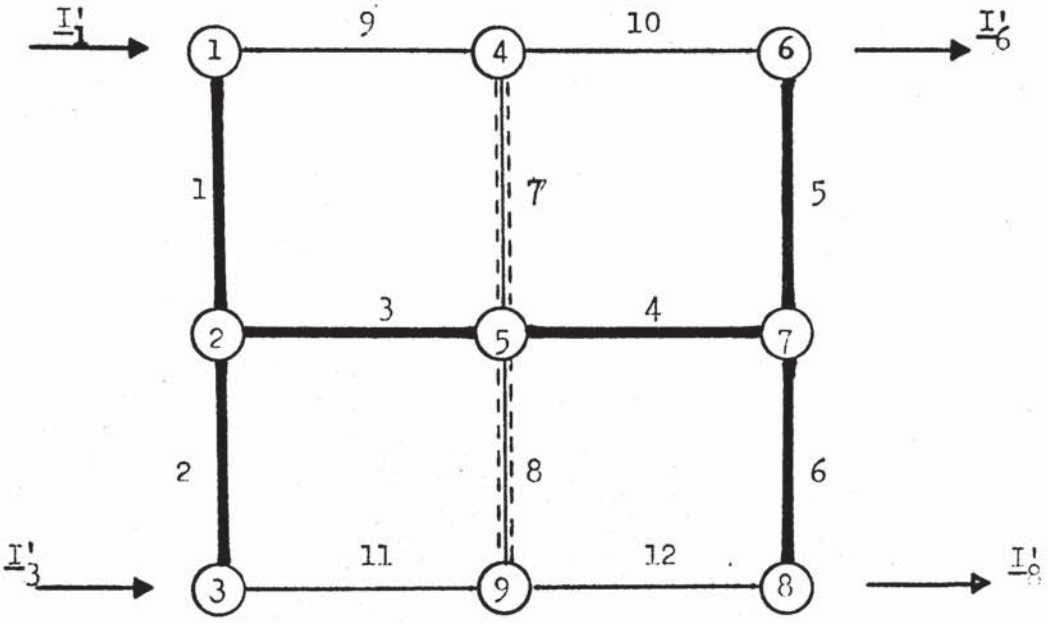
The second orthogonal current vector is $\underline{i}'$. This is the current vector due to all sources other than $\underline{I}'$. In the network shown in figure (3.7.2) there are four i' current components each associated with a link branch, the flow in that link representing the flow in the mesh defined by that link.

The connected network and the orthogonal network are assumed to be equivalent networks, and therefore $\underline{J}c$, the flow down any branch of either network must be identical. Equation (3.7.1) shows the conceptual difference in the assignment of the individual current components to the branches. The partitioned form of the equation further emphasises the difference between the two types of network.

$$\underline{J}c = \left|\begin{array}{c}\underline{J}c_T \\ \hline \underline{J}c_L\end{array}\right. = \left|\begin{array}{c}\underline{I}_T + \underline{i}_T \\ \hline \underline{I}_L + \underline{i}_L\end{array}\right. \equiv \left|\begin{array}{c}\underline{I}' \\ \hline \underline{i}'\end{array}\right. \qquad (3.7.1)$$

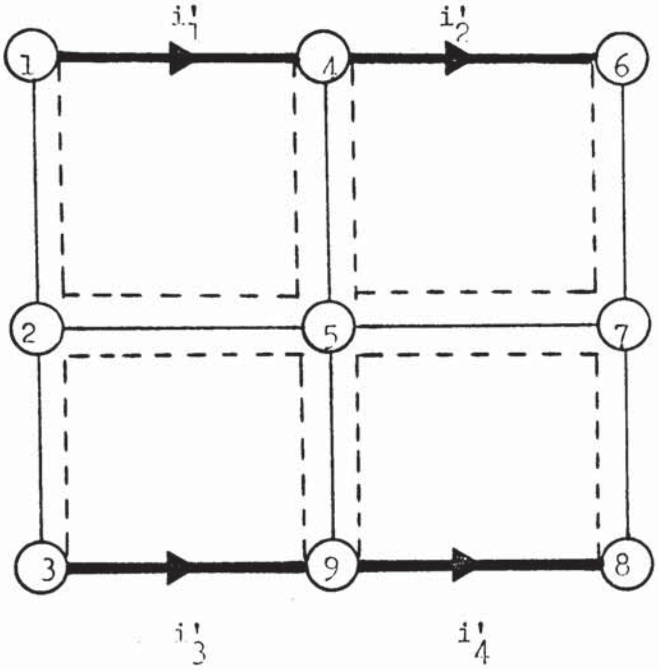
Connected Network   Orthogonal Network

Figure (3.7.1) Superimposition of node to datum current sources on a network



The four external flows produce node to datum flows in seven branches only.

Figure (3.7.2) Superimposition of mesh current on a network



Four link flows and their associated meshes

The orthogonal variables $\underline{I}'$ and $\underline{i}'$ may be transformed to primitive variables using equation (3.5.5) at the same time establishing the significance of the device $\underline{\gamma}$. The transformation equation is given in figure (3.7.3). $\underline{\gamma}$ is formed in figure (3.7.3) from the actual live currents flowing in the branches of the orthogonal network and those flowing in the primitive network. $\underline{\gamma}$ may be partitioned vertically, producing two submatrices which are seen to be identical to the two graph description matrices $\underline{B}$ and $\underline{C}$ developed in chapters (3.3.2) and (3.3.3). Thus transformations of the type given by equation (3.5.5) may be carried out by using combinations of simple graph describing matrices. This establishes the concept that the orthogonal network is indeed equivalent to the connected network. In chapter (3.3.6) $\underline{\gamma}$ was set up for the graph of the connected network with no mention of the physical properties associated with the network, whilst $\underline{\gamma}$ has been established in this chapter by reference to actual live currents in the branches of the network.

In a similar manner, $\underline{\alpha}$ of equation (3.5.6) may also be constructed from the live currents of a network and be shown to be identical to the $\underline{\alpha}$ matrix of chapter (3.3.6).

From the relationship expressed in equation (3.5.5) a further equation which shows the two individual transformations of $\underline{I}'$ and $\underline{i}'$ to give $\underline{J}$ may be set up as in equation (3.7.2).

$$\underline{J} = \begin{array}{c|c} \underline{B}_T & \underline{C}_T \\ \hline \underline{O}_L & \underline{U}_L \end{array} \cdot \left| \begin{array}{c} \underline{I}' \\ - \\ \underline{i}' \end{array} \right. \qquad (3.7.2)$$

The concepts presented so far may be summarised by reference to figures (3.7.4a) and (3.7.4b), which are graphs of the network used in appendices (3) and (9) to solve worked examples of network problems. They have various components of current assigned to their branches. Figure (3.7.4a) shows the currents as they have been assumed to be

Figure (3.7.3) Equation (3.5.5) - The transformation of orthogonal variables to primitive variables

$$
\left|\begin{array}{c}
J_1 \\ J_2 \\ J_3 \\ J_4 \\ J_5 \\ J_6 \\ J_7 \\ J_8 \\ \hline J_9 \\ J_{10} \\ J_{11} \\ J_{12}
\end{array}\right|
=
\left|\begin{array}{cccccccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
-1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & -1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right|
\cdot
\left|\begin{array}{c}
I'_1 \\ I'_2 \\ I'_3 \\ I'_4 \\ I'_5 \\ I'_6 \\ I'_7 \\ I'_8 \\ \hline i'_1 \\ i'_2 \\ i'_3 \\ i'_4
\end{array}\right|
$$

$$\underline{J} = \underline{\gamma}.\underline{J}c \qquad (3.5.5)$$

assigned to a connected graph. Each of the twelve branches has been assigned two components of current. Figure (3.7.4b) shows the currents as they have been assigned to the orthogonal network. Each of the paths, node to datum and mesh, has been assigned one component of flow only, and the network is therefore described in terms of twelve individual components of current as opposed to the 24 individual components of the connected network. Conceptually the currents of figure (3.7.4b) may be viewed as being made up of a combination of the external currents assigned to the branches of figure (3.7.1) and the resultant mesh currents assigned to the branches of figure(3.7.2).

3.8 Superimposing Voltage or Pressure Contours on the Node to Datum and Mesh Paths

Just as it has been shown that two orthogonal current components can be assigned to restricted paths within a network, it is also possible to show that orthogonal voltage components may be established which are also associated with the node to datum and mesh paths. The significance of $\underline{E}$ and $\underline{e}$ associated with a primitive network has been established, and this can be used to show the two voltage components $\underline{E}'$ and $\underline{e}'$ that are associated with the orthogonal equation by using the transformation equation (3.8.1),

$$\underline{Vc} = \underline{\breve{\gamma}}.\underline{V} \qquad (3.8.1)$$

This equation is the dual of equation (3.5.6). The form of $\underline{\gamma}$ is known, and if the primitive $\underline{V}$ is represented by its individual components $\underline{E}$ and $\underline{e}$ in partitioned form, then $\underline{Vc}$ is given by equation (3.8.2),

$$\underline{Vc} = \begin{array}{c|c} \underline{\breve{B}}_T & \underline{\Omega}_L \\ \hline \underline{\breve{C}}_T & \underline{U}_L \end{array} \cdot \begin{array}{c} \underline{E}_T + \underline{e}_T \\ \hline \underline{E}_L + \underline{e}_L \end{array} \qquad (3.8.2)$$

It is now necessary to suppose that $\underline{Vc}$ is made up of two components of flow which themselves may be partitioned into tree and link parts to

Figures (3.7.4a) and (3.7.4b) The superimposition of current components on the connected and orthogonal networks

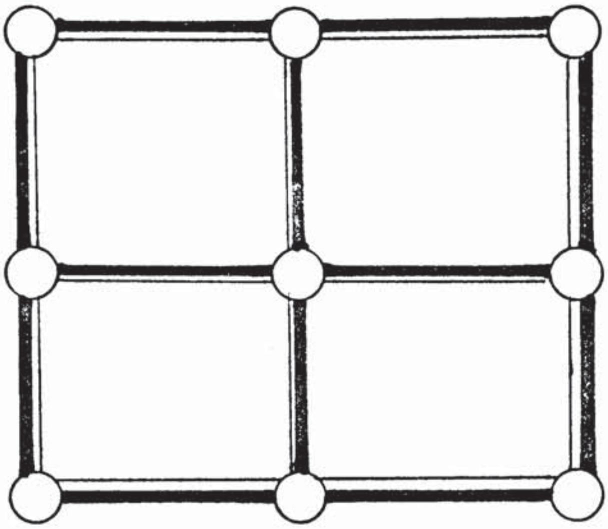


Figure (3.7.4a)

The connected network

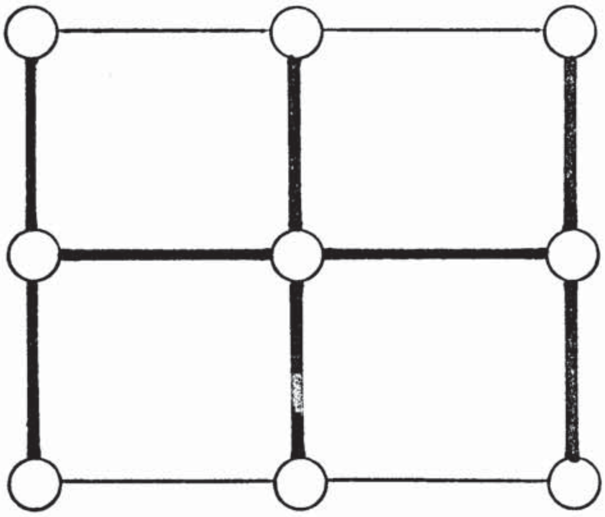


Figure (3.7.4b)

The orthogonal network

give equation (3.8.3),

$$\left|\begin{array}{c} \underline{Ec}_T + \underline{ec}_T \\ \hline \underline{Ec}_L + \underline{ec}_L \end{array}\right. = \begin{array}{c|c} \check{\underline{B}}_T & \underline{O}_L \\ \hline \check{\underline{C}}_T & \underline{U}_L \end{array} \cdot \left|\begin{array}{c} \underline{E}_T + \underline{e}_T \\ \hline \underline{E}_L + \underline{e}_L \end{array}\right. \qquad (3.8.3)$$

By performing the multiplications of this equation each of the components of voltage on the LHS may be examined in terms of those on the RHS. It is found that $\underline{ec}_L$ is given by equation (3.8.4),

$$\underline{ec}_L = \check{\underline{C}}_T.\underline{e}_T + \underline{U}_L.\underline{e}_L \qquad (3.8.4)$$

which sums the individual branch potentials around every mesh of the network and therefore according to Kirchoff's Second Law, $\underline{ec}_L$ reduces to a null vector. The remaining components of voltage are given in equations (3.8.4b) to (3.8.4d):

$$\underline{Ec}_T = \check{\underline{B}}_T.\underline{E} \qquad (3.8.4b)$$

$$\underline{Ec}_L = (\check{\underline{C}}_T.\underline{E}_T + \underline{E}_L) \qquad (3.8.4c)$$

$$\underline{ec}_T = \check{\underline{B}}_T.\underline{e}_T \qquad (3.8.4d)$$

The orthogonal vector $\underline{V}c$ is now given by equation (3.8.5),

$$\underline{V}c = \left|\begin{array}{c} \underline{Ec}_T + \underline{ec}_T \\ \hline \underline{Ec}_T + \underline{O}_L \end{array}\right. \qquad (3.8.5)$$

To unify the index notation it is necessary to introduce superscripts instead of subscripts and the equation for $\underline{V}c$ then becomes (3.8.6),

$$\underline{V}c = \underline{E}' + \underline{e}' \qquad (3.8.6)$$

Equation (3.5.3) of the orthogonal network can now be expressed in terms of the individual components of currents and voltages, as in equation (3.8.7),

$$\underline{E}' + \underline{e}' = \check{\underline{\gamma}}.\underline{Z}.\underline{\gamma} \, . \, \frac{\underline{I}'}{\underline{i}'} \qquad (3.8.7)$$

which may be expanded to demonstrate the role of the transformation matrices as in equation (3.8.8),

$$\underline{E}' + \underline{e}' = \begin{array}{c|c} \underline{\breve{B}}_T\cdot\underline{Z}_T\cdot\underline{B}_T & \underline{\breve{B}}_T\cdot\underline{Z}_T\cdot\underline{C}_T \\ \hline \underline{\breve{C}}_T\cdot\underline{Z}_T\cdot\underline{B}_T & \underline{\breve{C}}_T\cdot\underline{Z}_T\cdot\underline{C}_T + \underline{Z}_L \end{array} \cdot \left|\begin{array}{c} \underline{I}' \\ \hline \underline{i}' \end{array}\right. \tag{3.8.8}$$

from which a solution equation in terms of $\underline{i}'$ or $\underline{e}'$ may be obtained provided that $\underline{E}'$ is expressed in partitioned form as in equation (3.8.9),

$$\underline{E}' = \left|\begin{array}{c} \underline{E}'_T \\ \hline \underline{E}'_L \end{array}\right. \tag{3.8.9}$$

In equation (3.8.8), $\underline{e}'$ is an overall branch vector; in the following equations $\underline{e}'$ is regarded as a tree branch vector since the link branch part of the vector is a null vector.

$$\underline{E}'_T + \underline{e}' = (\underline{\breve{B}}_T\cdot\underline{Z}_T\cdot\underline{B}_T)\cdot\underline{I}' + (\underline{\breve{B}}_T\cdot\underline{Z}_T\cdot\underline{C}_T)\cdot\underline{i}' \tag{3.8.10}$$

$$\underline{E}'_L = (\underline{\breve{C}}_T\cdot\underline{Z}_T\cdot\underline{B}_T)\cdot\underline{I}' + (\underline{\breve{C}}_T\ \underline{Z}_T\ \underline{C}_L + \underline{Z}_L)\cdot\underline{i}' \tag{3.8.11}$$

$$\underline{i}' = (\underline{\breve{C}}_T\cdot\underline{Z}_T\cdot\underline{C}_T + \underline{Z}_L)^{-1}\cdot(\underline{E}'_L - \underline{\breve{C}}_T\cdot\underline{Z}_T\cdot\underline{B}_T\cdot\underline{I}') \tag{3.8.12}$$

$$\underline{e}' = (\underline{\breve{B}}_T\cdot\underline{Z}_T\cdot\underline{B}_T)\cdot\underline{I}' + (\underline{\breve{B}}_T\cdot\underline{Z}_T\cdot\underline{C}_T)\cdot\underline{i}' \tag{3.8.13}$$

Of the two solution equations, (3.8.12) is the more immediately interesting since it can be compared with the equation (3.8.14),

$$\underline{i}' = (\underline{\breve{C}}.\underline{Z}.\underline{C})^{-1}.\underline{\breve{C}}.(\underline{E} - \underline{Z}.\underline{I}) \tag{3.8.14}$$

which is called the classical mesh equation, used by Middleton (6), to express the solution of the mesh flows $\underline{i}'$ in terms of primitive quantities only. Under which condition, since no primitive expression for $\underline{I}$ was available, equation (3.8.14) remained insoluble.

Equation (3.8.12) is used in a modified form in the worked example of appendix (3) to show that network problems may be solved by the equation, but using orthogonal instead of primitive variables.

The equivalent orthogonal equations in terms of $\underline{\alpha}$ are outlined in equation (3.8.15),

$$\left.\begin{array}{c} \underline{I}' \\ \hline \underline{i}' \end{array}\right. = \begin{array}{c|c} \underline{\breve{A}}_T\cdot\underline{Y}_T\cdot\underline{A}_T + \underline{\breve{A}}_L\cdot\underline{Y}_L\cdot\underline{A}_L & \underline{\breve{A}}_L\cdot\underline{Y}_L \\ \hline \underline{Y}_L\cdot\underline{A}_L & \underline{Y}_T \end{array} \cdot \left|\begin{array}{c} \underline{E}'_T + \underline{e}' \\ \hline \underline{E}'_L \end{array}\right. \tag{3.8.15}$$

which is expanded in appendix (4) to yield the solution equations for $\underline{i}'$ and $\underline{e}'$ and show a comparison with the classical nodal method. A worked example for the nodal method is shown in appendix (5). The comparison with the classical nodal method is easily achieved if, of the two solution equations that can be developed out of equation (3.8.15), only the one in $\underline{e}'$ is considered.

A further simplifying concept that can be used to advantage is that of equivalent voltage or current sources. Equation (3.4.5) may be manipulated to give, for the primitive network, equation (3.8.16),

$$\underline{Z}.\underline{i} \quad = \quad \underline{e} + (\underline{E} - \underline{Z}.\underline{I}) \qquad (3.8.16)$$

The current vector $\underline{I}$ has been transformed to an equivalent voltage source ($\underline{Z}.\underline{I}$). A similar substitution can be made using ($\underline{Y}.\underline{E}$) in equation (3.4.6) to transform $\underline{E}$ into an equivalent current source, with the result that equation (3.8.8) can be re-expressed with the current source vector as zero in equation (3.8.17),

$$\begin{array}{c} \underline{E}_T^{*\prime} + \underline{e}' \\ \hline \underline{E}_L^{*\prime} \end{array} = \begin{array}{c|c} \underline{\breve{B}}_T.\underline{Z}_T.\underline{B}_T & \underline{\breve{B}}_T.\underline{Z}_T.\underline{C}_T \\ \hline \underline{\breve{C}}_T.\underline{Z}_T.\underline{C}_T & \underline{\breve{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L \end{array} \cdot \left| \begin{array}{c} \underline{0}_T \\ \hline \underline{i}' \end{array} \right. \qquad (3.8.17)$$

and the solution for $\underline{i}'$ is given by equation (3.8.18),

$$\underline{i}' \quad = \quad (\underline{\breve{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L)^{-1} . \underline{E}_L^{*\prime} \qquad (3.8.18)$$

where $\underline{E}_L^{*\prime}$ is given by equation (3.8.19),

$$\underline{E}_L^{*\prime} \quad = \quad \underline{\breve{C}}_T.(\underline{E}_T - \underline{Z}_T.\underline{I}_T) + \underline{E}_L \qquad (3.8.19)$$

with the result that equation (3.8.18) can be seen to be comparable with equation (3.8.14), save for the partitioning of the $\underline{C}$ matrix. Appendix (4) shows a similar comparison for $\underline{e}'$.

Middleton (6) used Roth's (14) algebraic diagram to develop the classical solution equations, and then proceeded to develop the solution to the diakoptic equations. There is little similarity in the concepts of the classical approach and the diakoptic approach, and therefore

Middleton's (6) development does not follow smoothly. It has been shown above however, that the classical methods may be derived out of the orthogonal approach, but not vice versa. The next two chapters show that the equations of diakoptics follow naturally from the orthogonal approach.

3.9 The Transformation of one Orthogonal Network to another

The development of the equations of diakoptics necessitates the setting up of a network which is equivalent to the orthogonal network but which is in a subdivided state, and requires the ability to transform variables associated with torn networks to those of the other. This chapter proposes the transformation matrix for this purpose, and shows that it has a special form.

Equation (3.5.5) describes the transformation of orthogonal variables to those of the primitive. If the torn network contains the same number of branches as the orthogonal network, then it transforms to the primitive in a similar manner, as in equation (3.9.1),

$$\underline{J} = \underline{\gamma}^{*} . \underline{Jc}^{*} \qquad (3.9.1)$$

where the superscript refers to the torn network. $\underline{J}$ for the primitive network must be the same for both transformations such that a relation between the orthogonal and torn networks is given by equation (3.9.2),

$$\underline{\gamma} . \underline{Jc} = \underline{\gamma}^{*} . \underline{Jc}^{*} \qquad (3.9.2)$$

This equation may be premultiplied by $\underline{\breve{\alpha}}$ to give the transformation of torn variables to orthogonal by the transformation matrix $\underline{\gamma}^{**}$ as in equation (3.9.3),

$$\underline{Jc} = \underline{\gamma}^{**} . \underline{Jc}^{*} \qquad (3.9.3)$$

The transformation matrix $\underline{\gamma}^{**}$ may be partitioned as in equation (3.9.4),

$$\underline{\gamma}^{**} = \left[\begin{array}{c|c} \underline{\gamma}^{**}_{TT} & \underline{\gamma}^{**}_{TL} \\ \hline \underline{\gamma}^{**}_{LT} & \underline{\gamma}^{**}_{LL} \end{array}\right] \qquad (3.9.4)$$

and the form of the four components of $\underline{\gamma}^{**}$ is given by equation (3.9.5),

$$\underline{\gamma}^{**} = \underline{\check{\alpha}}.\underline{\gamma}^{*} = \left.\begin{array}{c|c} \underline{\check{A}}_{T}.\underline{B}_{T}^{*} & \underline{\check{A}}_{T}.\underline{C}_{T}^{*} + \underline{\check{A}}_{L} \\ \hline \underline{0}_{LT} & \underline{U}_{LL} \end{array}\right. \tag{3.9.5}$$

If the torn and orthogonal networks have identical node to datum paths, then equation (3.9.5) reduces to equation (3.9.6),

$$\underline{\gamma}^{**} = \left.\begin{array}{c|c} \underline{U}_{TT} & \underline{\check{A}}_{T}.\underline{C}_{T}^{*} + \underline{\check{A}}_{L} \\ \hline \underline{0}_{LT} & \underline{U}_{LL} \end{array}\right. \tag{3.9.6}$$

If both networks also have the same meshes, then equation (3.9.6) reduces to equation (3.9.7),

$$\underline{\gamma}^{**} = \left.\begin{array}{c|c} \underline{U}_{TT} & 0 \\ \hline 0 & \underline{U}_{LL} \end{array}\right. \tag{3.9.7}$$

which is the one to one transformation matrix that one would require to transform the variables of one network to those of another identical network.

In the same way that a dual $\underline{\alpha}$ exists of the transformation matrix $\underline{\gamma}$, the dual of $\underline{\gamma}^{**}$ also exists and is generated by equation (3.9.8),

$$\underline{\alpha}^{**} = \underline{\check{\gamma}}.\underline{\alpha}^{*} \tag{3.9.8}$$

$\underline{\alpha}^{**}$ can be shown to consist of four submatrices which, if the node to datum paths are the same, give rise to the form of equation (3.9.9),

$$\underline{\alpha}^{**} = \left.\begin{array}{c|c} \underline{U}_{TT} & \underline{0}_{TL} \\ \hline \underline{A}_{L}^{*} + \underline{\check{C}}_{T}.\underline{A}_{T}^{*} & \underline{U}_{LL} \end{array}\right. \tag{3.9.9}$$

It was shown in chapter (3.5) that the $\underline{Z}c$ variables of the orthogonal network could be transformed to the $\underline{Z}$ variables of the primitive network using the transformation matrices $\underline{\alpha}$ and $\underline{\gamma}$.

The variables $\underline{Z}\overset{*}{c}$ of a torn network can be transformed to the $\underline{Z}c$ variables of an orthogonal network using $\underline{\alpha}^{**}$ and $\underline{\gamma}^{**}$, and employing the same algebra as in equations (3.5.10) to (3.5.12) to give the expression for $\underline{Z}\overset{*}{c}$, equation (3.9.10),

$$\underline{Zc}^{*} = \breve{\underline{\gamma}}^{**}.\underline{Zc}.\underline{\gamma}^{**} \qquad (3.9.10)$$

The equivalent expression for $\underline{Yc}^{*}$ is equation (3.9.11),

$$\underline{Yc}^{*} = \breve{\underline{\alpha}}^{**}.\underline{Yc}\underline{\alpha}^{**} \qquad (3.9.11)$$

The four equations (3.9.11), (3.9.10), (3.9.6) and (3.9.9) have an important role in the solution of network problems using diakoptic methods.

3.10 Introduction to Diakoptics

It is proposed to show that the concepts of node to datum and mesh paths which were used in chapter (3.8) to obtain solution equations to network problems may be further extended to produce two new solution equations called the equations of diakoptics. These equations make possible the analysis of much larger systems than can be analysed by existing methods.

The technique of solving network problems by diakoptics can be visualised as a two stage operation. The equations of the primitive branches of a network are solved sequentially. By removing selected branches from the graph of a network, it is possible to produce a series of subgraphs which would normally be linked together by the cut branches. The primitive branches of the network which represent the sub networks are interconnected and a quasi-solution found for each of the separate sub networks in terms of the orthogonal variables. This is the first stage of solution called low-level interconnection.

Using the solution equations of the individual sub networks, and the information about the cut branches, the sub networks are interconnected and a solution is found for the overall network. This is the second stage of solution called high-level interconnection. In terms of computing efficiency this means that the low-level interconnections for a series of sub networks may be carried out separately and since only the solution equation for each sub network has to be retained for high-level interconnections, the storage requirements are the same for the overall network as for the largest sub network. A second advantage of the technique is that the necessary inversion of the solution matrix is reduced to the inversion of the matrix of the largest sub network. For example, the solution of the network by the nodal method in the worked example of appendix (5) requires the inversion of an 8 by 8 matrix, whilst the solution of the same network by nodal diakoptics, appendix (8), requires the inversion of a 5 by 5 and 2 by 2. The number of operations required to obtain the inverse of a matrix is approximately a cubic function of the size of the matrix, therefore the nodal diakoptic method is the much more efficient of the two methods.

3.10.1 Low-level interconnection

To establish the concepts underlying the equation of diakoptics, the equations will be developed for a single low-level interconnection. The primitive branches of the total network will be interconnected in one operation. The equations of solution arrived at in this manner are similar to those derived in chapter (3.8), but the development is expanded, and the equations later applied to the solution of sub networks of larger systems.

The nodal method of diakoptics sums the solution of voltage effects from the external sources, and the voltage effects due to all other sources, to produce a solution to the overall network problem. The significance of the two effects is pointed out in the development. The orthogonal equation (3.8.8) may be solved in terms of two components $\underline{V}_T$ and $\underline{V}_L$ to give equations (3.10.1) and (3.10.2).

$$\underline{V}_T = \underline{E}'_T + \underline{e}' = \underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}' + \underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{C}_T \cdot \underline{i}' \qquad (3.10.1)$$

$$\underline{V}_L = \underline{E}'_L = \underline{\breve{C}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}' + (\underline{\breve{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L) \cdot \underline{i}' \qquad (3.10.2)$$

$\underline{V}_T$ can be expressed as two further components as in equations (3.10.3) and (3.10.4),

$$\underline{V}_T^{(1)} = \check{\underline{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}' \qquad (3.10.3)$$

$$\underline{V}_T^{(2)} = \check{\underline{B}}_T \cdot \underline{Z}_T \cdot \underline{C}_T \cdot \underline{i}' \qquad (3.10.4)$$

These two equations may be re-expressed as equations (3.10.5) and (3.10.6),

$$\underline{V}_T^{(1)} = \check{\underline{B}}_T \cdot \underline{Z}_T \cdot \underline{I}_T \qquad (3.10.5)$$

$$\underline{V}_T^{(2)} = \check{\underline{B}}_T \cdot \underline{Z}_T \cdot \underline{i}_T \qquad (3.10.6)$$

$\underline{V}_T^{(1)}$ is assumed to be the voltage component of solution due to the external sources and $\underline{V}_T^{(2)}$ the component due to all other sources.

Equations (3.10.5) and (3.10.6) can be rewritten as equations (3.10.7) and (3.10.8),

$$\underline{V}^{(1)} = \underline{Z}_T \cdot \underline{I}_T \qquad (3.10.7)$$

$$\underline{V}^{(2)} = \underline{Z}_T \cdot \underline{i}_T \qquad (3.10.8)$$

such that $\underline{V}$ is given by equation (3.10.9),

$$\underline{V} = \underline{V}^{(1)} + \underline{V}^{(2)} \qquad (3.10.9)$$

Also let $\underline{V}_T$ be expressed by equation (3.10.10),

$$\underline{V}_T = \check{\underline{B}}_T \cdot \underline{V} \qquad (3.10.10)$$

The component $\underline{V}_L$ may also be expressed in terms of the two components given in equations (3.10.11) and (3.10.12),

$$\underline{V}_L^{(1)} = \check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}' \qquad (3.10.11)$$

$$\underline{V}_L^{(2)} = (\check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L) \cdot \underline{i}' \qquad (3.10.12)$$

and these equations can be reduced to equations (3.10.13) and (3.10.14),

$$\underline{V}_L^{(1)} = \check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{I}_T \qquad (3.10.13)$$

$$\underline{V}_L^{(2)} = \check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{i}_T \qquad (3.10.14)$$

From equation (3.10.7), $\underline{V}_L^{(1)}$ can be expressed by equation (3.10.20),

$$\underline{V}_L^{(1)} = \underline{\check{C}}_T.\underline{V}^{(1)} \qquad (3.10.20)$$

The solution equation (3.10.9) is now shown to produce an algorithm for the solution of the nodal pressures of the network.

The component $\underline{V}^{(1)}$ of equation (3.10.9) is calculated from equation (3.10.7). The component $\underline{V}^{(2)}$ is calculated using equation (3.10.2), which may be reduced by expressing the voltage component as $\underline{E}^*$ as in equation (3.10.15),

$$\underline{E}^* = \underline{E}_L' - \underline{V}_L^{(1)} \qquad (3.10.15)$$

where $\underline{V}_L^{(1)}$ may be expressed in terms of $\underline{V}^{(1)}$ from equation (3.10.20) to give for $\underline{i}'$ equation (3.10.16),

$$\underline{i}' = (\underline{\check{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L)^{-1}.\underline{E}^* \qquad (3.10.16)$$

and $\underline{i}'$ can be transformed as in equation (3.10.17),

$$\underline{i}_T = \underline{\check{C}}_T.\underline{i}' \qquad (3.10.17)$$

with the result that $\underline{V}^{(2)}$ can be found from equation (3.10.8). Notice that $\underline{E}_L'$ of equation (3.10.15) may be found from equation (3.8.4c).

From the two components of voltage $\underline{V}^{(1)}$ and $\underline{V}^{(2)}$ the node to datum voltages $\underline{V}_T$ can be found from equation (3.10.10) and by subtraction of the pressure source vector given by equation (3.8.4b),

$$\underline{E}_T' = \underline{\check{B}}_T.\underline{E} \qquad (3.8.4b)$$

The node to datum voltage $\underline{e}'$ is given by equation (3.10.19),

$$\underline{e}' = \underline{V}_T - \underline{E}_T' \qquad (3.10.19)$$

The above development has shown that the node to datum voltages $\underline{e}'$ may be calculated from two separate components $\underline{V}^{(1)}$ and $\underline{V}^{(2)}$. However it was shown that the much simpler development of chapter (3.8) produces a solution in terms of the nodal pressure. The difference between the two developments is conceptual. The development of this chapter treats the tree branches of a network as a separate radial network for which a solution component of voltage is found. The link branches are then interconnected into the tree branches in equation (3.10.16) to produce a further component of voltage, the interconnection component, which added to the tree component produces the required solution. A worked example of low-level interconnection showing the principle of two component solution is shown in appendix (7).

The concepts are further expanded. The equations of the sub networks of a large system, if they can be treated as their radial equivalents, may be treated as the voltage component $\underline{V}^{(1)}$ of a solution. The equations of the branches which interconnect together the sub networks may be treated as the link branches of the above development to produce a voltage component of solution $\underline{V}^{(2)}$.

3.10.2 High-level interconnection and nodal diakoptics

The principles will be developed for the interconnection of the sub networks of a large system which has been torn apart. A worked example which uses these principles is shown in appendix (8).

Just as the component of voltage $\underline{V}^{(1)}$ was calculated for the tree branches in chapter (3.10.1), a component $\underline{V}^{(1)}$ is calculated for an equivalent radial network for high level interconnection. To do this it is necessary to assume that there exists for each sub network an equivalent radial network, the equations of which represent the equations associated with the interconnected sub network, which may contain tree and link branches, in an equivalent radial form. These radial equivalents are then assumed to be a series of primitive branches of a much larger system which themselves are to be interconnected. The node to datum potentials $\underline{e}'$ of each subdivision may be calculated by any of the methods developed in chapter (3.8). Each of the sub networks will then have associated with it a vector $\underline{e}'$. Now this vector must be thought of as a single component of a larger vector $\underline{V}^{(1)}$, so that $\underline{V}^{(1)}$ is thought of as being made up of radial equivalent potentials $\underline{e}'$, not individual potentials e' as in chapter (3.10.1).

For each sub network, $\underline{e}'$ is most easily obtained by firstly converting the primitive potential sources in the subdivisions to equivalent current sources using the dual of equation (3.8.16). This produces a solution of the node to datum potentials for each subdivision given by equation (3.10.21),

$$\underline{e}' = (\breve{\underline{A}}_T.\underline{Y}_T.\underline{A}_T + \breve{\underline{A}}_L.\underline{Y}_L.\underline{A}_L)\underline{I}^* \qquad (3.10.21)$$

where $\underline{I}^*$ is the node to datum vector which contains both primitive current and voltage sources.

Now if $\underline{e}'$ of each subdivision is assumed to be one component of $\underline{V}^{(1)}$ then $(\breve{\underline{A}}_T.\underline{Y}_T.\underline{A}_T + \breve{\underline{A}}_L.\underline{Y}_L.\underline{A}_L)^{-1}$ must be assumed to be the equivalent radial form of a subnetwork which corresponds to $\underline{Z}_T$ of equation (3.10.7). That is each matrix $(\breve{\underline{A}}_T.\underline{Y}_T.\underline{A}_T + \breve{\underline{A}}_L.\underline{Y}_L.\underline{A}_L)^{-1}$ is thought of as a single component of the matrix $\underline{Z}_T$.

To find $\underline{V}^{(2)}$ in chapter (3.10.1), $\underline{C}$ was used for transforming orthogonal mesh variables to tree variables, although only $\underline{C}_T$ was written into the equations because $\underline{C}_L$ is a unit matrix. A similar matrix also exists for transforming orthogonal mesh variables to equivalent radial variables. This matrix is a partitioned part of the $\underline{\gamma}^{**}$ matrix of chapter (3.9), just as $\underline{C}$ is a partitioned part of $\underline{\gamma}$ of chapter (3.7). The form of $\underline{\gamma}^{**}$ has been indicated previously and its partitioned part $\underline{\gamma}^{**}_{TL}$ of equation (3.9.6) is exactly analogous to $\underline{C}_T$. It is also true that the partitioned part $\underline{\gamma}^{**}_{LL}$ is analogous to $\underline{C}_L$ and is a unit matrix.

The concept which has to be established is that as $\underline{C}$ is the transforming matrix of the primitive and orthogonal system, $\underline{\gamma}^{**}_{TL}$ combined with $\underline{\gamma}^{**}_{LL}$ is the transforming matrix of the new primitive system consisting of the equivalent radial forms of the sub networks and the connected orthogonal network.

In the following equations transformations using $\underline{\gamma}^{**}_{TL}$ and $\underline{\gamma}^{**}_{LL}$ are made which are analogous to the transformations made in chapter (3.10.1), where the transformation of a variable by $\underline{C}$ was reduced to a transformation of the type shown in equation (3.8.8), given by equation (3.8.8a)

$$(\check{\underline{C}}.\underline{Z}.\underline{C}) = (\check{\underline{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L) \qquad (3.8.8a)$$

where the unit matrix is omitted. Any transformation which follows, using the partitioned form of $\underline{\gamma}^{**}$, implies transformation by both $\underline{\gamma}^{**}_{TL}$ and $\underline{\gamma}^{**}_{LL}$ but multiplication by $\underline{\gamma}^{**}_{LL}$ omitted.

The precise form of $\underline{\gamma}^{**}_{TL}$ is that it shows the incidence of the cut branches at their nodes and has the dimensions total number of nodes by number of cut branches. That is, it is capable of transforming properties associated with nodes to those of cut branches and vice versa.

The equivalent equation to (3.10.20) is given by (3.10.22),

$$\underline{v}_L^{(1)} = \check{\underline{\gamma}}^{**}_{TL}\ \underline{v}^{(1)} \qquad (3.10.22)$$

where, as outlined above, the individual components of $\underline{v}^{(1)}$ are the nodal vectors of the sub networks and the components of $\underline{v}_L^{(1)}$ are cut branch properties as opposed to link branch properties. Similarly the equivalent to equation (3.10.6) is given by equation (3.10.23),

$$\underline{i}' = (\check{\underline{\gamma}}^{**}_{TL}.\underline{Z}_T.\underline{\gamma}^{**}_{TL} + \underline{Z}_L)^{-1}.\underline{E}_L^{*} \qquad (3.10.23)$$

where the individual components of $\underline{Z}_T$ are made up of the matrices $(\check{\underline{A}}_T.\underline{Y}_L.\underline{A}_T + \check{\underline{A}}_L.\underline{Y}_L.\underline{A}_L)^{-1}$ of each of the sub networks and the $\underline{Z}_L$ components are cut branch components as opposed to link branch components of chapter (3.10.1). $\underline{i}_T$ can be found from the equivalent of equation (3.10.17) as shown in equation (3.10.24),

$$\underline{i}_T = \underline{\gamma}^{**}_{TL}\ \underline{i}' \qquad (3.10.24)$$

$\underline{v}^{(2)}$ is then given by the equivalent of (3.10.8), remembering the form of the components of $\underline{Z}_T$. From which equation (3.10.9) may be solved, which together with equations (3.10.18) and (3.10.19) gives a route to find the node to datum potentials of the connected network. A computation algorithm may be developed from the above reasoning and is shown in figure (3.10.1). This algorithm forms the basis of the program written to utilise the nodal diakoptic method which is discussed in chapter (4). The steps of the algorithm may be described as follows:

Step 1: This converts the potential sources to equivalent current sources and transforms the result to a node to datum quantity $\underline{I}^*$ for every sub network.

Step 2: Transforms the primitive variables expressed in $\underline{I}^*$ to their radial equivalents for every sub network.

Step 3: Transforms any primitive potential sources in the cut branches to orthogonal potentials.

Step 4: Transforms the equivalent radial potentials to connected orthogonal variables.

Step 5: Combines the two types of orthogonal potentials, one due to the link branches, the other due to the equivalent radial networks.

Step 6: Transforms the equivalent radial networks and cut branches to orthogonal mesh variables, ie the interconnection step.

Step 7: Transforms the mesh variables to orthogonal tree current variables.

Step 8: Finds the orthogonal branch potentials due to interconnection.

Step 9: Sums the two components of solution of voltages.

Steps 10 and 11: Evaluates the node to datum potentials $\underline{e}'$ for the overall network.

3.10.3 Mesh diakoptics

A second series of diakoptic equations may also be developed, and since it relies on mesh solutions it is felt it will be more useful to engineers than the nodal diakoptic approach for the reason that most engineering networks are not heavily meshed.

It was realised by the author sometime ago that a dual of the nodal

Figure (3.10.1) An algorithm for solving network problems by the nodal diakoptics technique

1. $\underline{I}^{*} = \breve{\underline{A}}(\underline{I}_{T} - \underline{Y}.\underline{E})$

2. $\underline{V}^{(1)} = (\breve{\underline{A}}_{T}.\underline{Y}_{T}.\underline{A}_{T} + \breve{\underline{A}}_{L}.\underline{Y}_{L}.\underline{A}_{L})^{-1}.\underline{I}^{*}$

3. $\underline{E}'_{L} = \breve{\underline{\gamma}}^{**}_{TL}.\underline{E}_{T} + \underline{E}_{L}$

4. $\underline{V}_{L}^{(1)} = \breve{\underline{\gamma}}^{**}_{TL}.\underline{Z}_{T}.\underline{I}^{*} = \breve{\underline{\gamma}}^{**}_{TL}.\underline{V}^{(1)}$

5. $\underline{E}^{*} = \underline{E}'_{L} - \underline{V}_{L}^{(1)}$

6. $\underline{i}' = (\breve{\underline{\gamma}}^{**}_{TL}.(\breve{\underline{A}}_{T}.\underline{Y}_{T}.\underline{A}_{T} + \breve{\underline{A}}_{L}.\underline{Y}_{L}.\underline{A}_{L})^{-1}.\underline{\gamma}^{**}_{TL} + \underline{Z}_{L}).\underline{E}^{*}$

7. $\underline{i}_{T} = \underline{\gamma}^{**}_{TL}.\underline{i}'$

8. $\underline{V}^{(2)} = (\breve{\underline{A}}_{T}.\underline{Y}_{T}.\underline{A}_{T} + \breve{\underline{A}}_{L}.\underline{Y}_{L}.\underline{A}_{L})^{-1}.\underline{i}_{T}$

9. $\underline{V} = \underline{V}^{(1)} + \underline{V}^{(2)}$

10. $\underline{V}_{T} = \breve{\underline{B}}_{T}.\underline{V}$

11. $\underline{e}' = \underline{V}_{T} - \underline{E}'_{T}$

diakoptic technique ought to exist, just as most other concepts of network analysis are matched by a dual (for example the dual of the node to datum path is the mesh path). However it was also realised that removing a series of branches from the graph of a network, in the same way as the nodal diakoptic method, which reduces the number of meshes but leaves the same number of nodes, was not a viable proposition, because there would remain insufficient mesh information to produce a solution.

An alternative method of cutting which does not appear to have been presented elsewhere is now proposed.

The nodal diakoptic technique relies on cutting selected branches across their axes, thereby producing separate sub networks with the cut branches treated in isolation. It is suggested that the mesh diakoptic method relies on cutting the network along the axes of the selected cut branches, through the terminal nodes of the cut branches. With the cut branches removed, sub networks are produced which now contain as many unclosed meshes as there are cut branches. To maintain enough equations to solve the subnetworks in terms of their mesh equivalents, it is necessary to provide a closing path for the opened meshes. This is achieved by assuming that there exists a short circuit between the two nodes from between which a cut branch has been removed.

The total number of meshes in the sub networks is then the same as that in the fully interconnected network, thereby giving sufficient equations for solution of the overall network.

The outline of the proposed equations is shown below, but the explanation is not as comprehensive as that for nodal diakoptics since many of the same principles are involved.

The starting point is the orthogonal equation (3.8.15),

$$\left|\begin{array}{c}\underline{J}_T\\\hline \underline{J}_L\end{array}\right. = \left|\begin{array}{c}\underline{I}'\\\hline \underline{i}'\end{array}\right. = \begin{array}{c|c} \breve{\underline{A}}_T\cdot\underline{Y}_T\cdot\underline{A}_T + \breve{\underline{A}}_L\cdot\underline{Y}_L\cdot\underline{A}_L & \breve{\underline{A}}_L\cdot\underline{Y}_L \\\hline \underline{Y}_L\cdot\underline{A}_L & \underline{Y}_L\end{array} \cdot \left|\begin{array}{c}\underline{E}'_T+\underline{e}'\\\hline \underline{E}'_L\end{array}\right. \qquad (3.8.15)$$

From this equation the principle of the summation of two current components, one due to the external sources and the other due to interconnection, is shown to produce a solution to a low-level interconnection. The progression to the interconnection of subdivisions (high-level interconnections) is discussed later in this chapter.

Equation (3.8.15) can be solved in terms of $\underline{J}_T$ and $\underline{J}_L$ to give equations (3.10.25) and (3.10.26),

$$\underline{J}_T = \underline{I}' = (\breve{\underline{A}}_T\cdot\underline{Y}_T\cdot\underline{A}_T + \breve{\underline{A}}_L\cdot\underline{Y}_L\cdot\underline{A}_L)\cdot(\underline{E}'_T + \underline{e}') + \breve{\underline{A}}_L\cdot\underline{Y}_L\cdot\underline{E}'_L \qquad (3.10.25)$$

$$\underline{J}_L = \underline{i}' = \underline{Y}_L\cdot\underline{A}_L\cdot(\underline{E}'_T + \underline{e}') + \underline{Y}_L\cdot\underline{E}'_L \qquad (3.10.26)$$

Let $\underline{J}_T$ consist of two components $\underline{J}_T^{(1)}$ and $\underline{J}_T^{(2)}$ as given by equations (3.10.27) and (3.10.28),

$$\underline{J}_T^{(1)} = (\breve{\underline{A}}_T\cdot\underline{Y}_T\cdot\underline{A}_T + \breve{\underline{A}}_L\cdot\underline{Y}_L\cdot\underline{A}_L)\cdot\underline{E}'_T + \breve{\underline{A}}_L\cdot\underline{Y}_L\cdot\underline{E}'_L \qquad (3.10.27)$$

$$\underline{J}_T^{(2)} = (\breve{\underline{A}}_T\cdot\underline{Y}_T\cdot\underline{A}_T + \breve{\underline{A}}_L\cdot\underline{Y}_L\cdot\underline{A}_L)\cdot\underline{e}' \qquad (3.10.28)$$

Since $\underline{A}$ is made up of $\underline{A}_T$ and $\underline{A}_L$ and $\underline{Y}$ is made up of $\underline{Y}_T$ and $\underline{Y}_L$, equation (3.10.27) can be rewritten as equation (3.10.29),

$$\underline{J}_T^{(1)} = \breve{\underline{A}}.\underline{Y}.(\underline{A}.\underline{E}'_T + \underline{E}'_L) \qquad (3.10.29)$$

The primitive potential source $\underline{E}$ can be expressed by equation (3.10.30),

$$E = (\underline{A}.\underline{E}'_T + \underline{E}'_L) \qquad (3.10.30)$$

and therefore equation (3.10.29) can be expressed as equation (3.10.31),

$$\underline{J}^{(1)} = \underline{Y}.\underline{E} \qquad (3.10.31)$$

which is an expression for the current component of solution due to voltage sources, which can be transformed as in equation (3.10.43),

$$\underline{J}_T^{(1)} = \breve{\underline{A}}\,\underline{J}^{(1)} \qquad (3.10.43)$$

The second component due to interconnection is obtained using $\underline{J}_T^{(2)}$ which can be expressed as equation (3.10.32),

$$\underline{J}_T^{(2)} = \underline{\breve{A}}.\underline{Y}.\underline{A}.\underline{e}' \qquad (3.10.32)$$

$\underline{e}'$ can be transformed to the primitive $\underline{e}$ and $\underline{J}_T^{(2)}$ transformed using the equation equivalent to (3.10.43) to give for $\underline{J}^{(2)}$ equation (3.10.33),

$$\underline{J}^{(2)} = \underline{Y}.\underline{e} \qquad (3.10.33)$$

Equation (3.10.26) may also be expressed in terms of two components given by equations (3.10.34) and (3.10.35),

$$\underline{J}_L^{(1)} = \underline{Y}_L.\underline{A}_L.\underline{E}_T' + \underline{Y}_L.\underline{E}_L' \qquad (3.10.34)$$

$$\underline{J}_L^{(2)} = \underline{Y}_L.\underline{A}_L.\underline{e}' \qquad (3.10.35)$$

Equation (3.10.34) can be shown (equation (3.10.30)) to reduce to equation (3.10.36),

$$\underline{J}_L^{(1)} = \underline{Y}_L.\underline{E}_L \qquad (3.10.36)$$

It is seen that in equation (3.8.15), multiplications by the matrix $\underline{F}$ have not been shown because of the special form of the matrix. Jt must be realised however, that in the mesh diakoptic development it plays an important role. For instance the multiplication of a mesh vector by the $\underline{F}$ matrix results in a branch vector, but since only the link branch partition of the branch vector is significant the tree branch partition is neglected.

It can also be shown that equation (3.10.35) will reduce to equation (3.10.37),

$$\underline{J}_L^{(2)} = \underline{F}.\underline{Y}.\underline{e} \qquad (3.10.37)$$

To find the solution of flow $\underline{J}$, which is given by equation (3.10.38),

$$\underline{J} = \underline{J}^{(1)} + \underline{J}^{(2)} \qquad (3.10.38)$$

$\underline{J}^{(1)}$ may be evaluated from equation (3.10.31) and $\underline{J}^{(2)}$ from equation (3.10.33).

Equation (3.10.25) may be reduced by expressing the current components by $\underline{I}^*$ and $\underline{J}_T^{(1)}$ as in equation (3.10.39),

$$\underline{I}^* = \underline{I}' - \underline{J}_T^{(1)} \qquad (3.10.39)$$

where $\underline{I}'$ is found from $\underline{I}' = \check{A}\ I$. Equation (3.10.39) is the dual of equation (3.10.15). Substituting equation (3.10.39) into (3.10.25) gives a solution in terms of $\underline{e}'$ of the form of equation (3.10.40),

$$\underline{e}' \;=\; (\underline{\check{A}}_T\cdot\underline{Y}_T\cdot\underline{A}_T + \underline{\check{A}}_L\cdot\underline{Y}_L\cdot\underline{A}_L)^{-1}\ \underline{I}^* \qquad (3.10.40)$$

This equation may be transformed by $\underline{A}$ to give $\underline{e}$. $\underline{J}^{(2)}$ may then be found from equation (3.10.33).

From the two components $\underline{J}^{(2)}$ and $\underline{J}^{(1)}$ the current component $\underline{J}$ is found. $\underline{J}$ may be transformed to $\underline{J}_L$ by $\underline{F}$ and since there is no component of flow $\underline{I}_L'$, the overall solution to the mesh currents given by equation (3.10.41),

$$\underline{i}' \;=\; \underline{J}_L - \underline{I}_L' \qquad (3.10.41)$$

reduces to equation (3.10.42),

$$\underline{i}' \;=\; \underline{J}_L \qquad (3.10.42)$$

The equations shown above may be rearranged to produce a computational algorithm for the solution by high-level interconnection of large scale systems, as shown in figure (3.10.2).

A much simpler method of finding the solution of the mesh currents, which is similar to the classical mesh method, was developed in chapter (3.8). Equations (3.10.25) to (3.10.42) propose a method which obtains the same result but is conceptually different. The mesh flows are a direct result of two components, one due to $\underline{J}^{(1)}$, the current invoked by the external sources $\underline{E}$ and $\underline{I}$, and the other, $\underline{J}^{(2)}$, invoked by the interconnection. From this development the extension to high-level interconnection follows.

Consider a large network that has been subdivided by the elimination of the cut branches, which have been removed in accordance with the principle laid down at the beginning of this chapter. The solutions to the subdivisions are obtained in terms of their mesh equivalents by the mesh method proposed in chapter (3.8) with the current sources treated as equivalent voltage sources.

A new network can then be visualised. This network which is an orthogonal network is equivalent to the connected orthogonal network and consists of a collection of mesh equivalent sub networks and a series of unique cut branches. Using the $\underline{\alpha}^{**}$ transformation matrix which has the ability to transform the variables associated with one orthogonal network to another, it is possible to transform the variables of the mesh equivalent sub networks and cut branches to variables of the connected orthogonal network, provided they have the same node to datum paths.

Equation (3.10.43) shows how the current component $\underline{J}^{(1)}$ can be transformed to a connected orthogonal network variable by $\breve{\underline{A}}$ for low-level interconnection. In the same way that a partitioned part of $\underline{\gamma}^{**}$ is the transformation matrix for torn to orthogonal variables analogous to $\underline{C}$, so a partitioned part of $\underline{\alpha}^{**}$ acts as a transformation matrix to transform the equivalent mesh and cut branch quantities to orthogonal quantities and is the analogue of $\underline{A}$.

In chapter (3.9) it was stated that under certain conditions $\underline{\alpha}^{**}$ has a special form. Provided that the node to datum paths of the two equivalent networks are identical, then the submatrices $\underline{\alpha}^{**}_{TT}$ and $\underline{\alpha}^{**}_{LL}$ reduce to unit matrices and $\underline{\alpha}^{**}_{TL}$ is zero. The important submatrix $\underline{\alpha}^{**}_{LT}$ is an incidence matrix and indicates the manner in which the cut branches are included in their original meshes. The exact form of the submatrices $\underline{\gamma}^{**}_{TL}$ and $\underline{\alpha}^{**}_{LT}$ is shown in the worked examples of appendices (7) and (9).

Just as the primitive variable $\underline{Y}$ can be transformed to the orthogonal variable $(\breve{\underline{A}}.\underline{Y}.\underline{A})$, variables associated with the equivalent mesh networks together with the cut branches can be transformed to connected orthogonal variables of the form $(\breve{\underline{\alpha}}^{**}.\underline{Y}.\underline{\alpha}^{**})$. If each equivalent mesh equation is thought of as equivalent to a single mesh path, then the overall torn network may be thought of in terms of a collection of individual mesh paths to be joined together by link branches. For low-level interconnection a series of tree branches and link branches were interconnected by equation (3.10.40). The equivalent equation for the torn to orthogonal transformation is given as equation (3.10.44),

$$\underline{e}' = (\breve{\underline{\alpha}}^{**}_{LT}.(\underline{C}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L)^{-1}.\underline{\alpha}^{**}_{LT} + \underline{Y}_T)^{-1}.\underline{I}^* \qquad (3.10.44)$$

where $\underline{Y}_T$ is associated with link branches.

The mesh equations in $(\breve{\underline{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L)^{-1}$ have been treated as mesh branches $(\underline{Y}_L)$ of an equivalent system which has been transformed according to equation (3.10.45),

$$(\breve{\underline{\alpha}}^{**}_{LT}.\underline{Y}_L.\underline{\alpha}^{**}_{LT} + \underline{Y}_T) = \underline{U}_{TT}.\breve{\underline{\alpha}}^{**}_{LT}.\left|\begin{array}{c|c} \underline{Y}_T & \underline{0} \\ \hline \underline{0} & \underline{Y}_L \end{array}\right| . \left|\begin{array}{c} \underline{U}_{TT} \\ \hline \underline{\alpha}^{**}_{LT} \end{array}\right| \qquad (3.10.45)$$

Further transformation using $\underline{\alpha}^{**}_{LT}$ may be developed which are analogous to the transformations employing $\underline{A}$ used in low-level interconnections. The result is that the algorithm shown in figure (3.10.2) may be developed for the solution of large scale problems by high-level mesh diakoptics. The algorithm has been used as the basis of a computer program which is discussed at a later stage.

Step 1: Converts the node to datum current source to equivalent potential sources and transforms the primitive branch quantities to orthogonal mesh quantities for each sub network.

Step 2: Finds the component of current for solution due to the external sources for each subnetwork.

Step 3: Converts any current sources in the cut branches to orthogonal quantities.

Step 4: Combines the equivalent mesh currents and any of those in the cut branches in terms of orthogonal quantities.

Step 6: Transforms the equivalent mesh quantities and cut branch quantities to give the node to datum potentials of the

Figure (3.10.2) An algorithm for solving network problems by the mesh diakoptic technique

1) $\underline{E}^{*} = \underline{\breve{C}}(\underline{E} - \underline{Z}.\underline{I})$

2) $\underline{J}^{(1)} = (\underline{\breve{C}}_{T}.\underline{Z}_{T}.\underline{C}_{T} + \underline{Z}_{L})^{-1}.\underline{E}^{*}$

3) $\underline{I}' = \underline{\breve{\alpha}}^{**}_{LT}.\underline{I}_{T}$

4) $\underline{J}_{T}^{(1)} = \underline{\breve{\alpha}}^{**}_{LT}.\underline{J}^{(1)}$

5) $\underline{I}^{*} = \underline{I}' - \underline{J}_{T}^{(1)}$

6) $\underline{e}' = (\underline{\breve{\alpha}}^{**}_{LT}.(\underline{\breve{C}}_{T}.\underline{Z}_{T}.\underline{C}_{T} + \underline{Z}_{L})^{-1}.\underline{\alpha}^{**}_{LT} + \underline{Y}_{T})^{-1}.\underline{I}^{*}$

7) $\underline{e} = \underline{\alpha}^{**}_{LT}.\underline{e}'$

8) $\underline{J}^{(2)} = (\underline{\breve{C}}_{T}.\underline{Z}_{T}.\underline{C}_{T} + \underline{Z}_{L})^{-1}.\underline{e}$

9) $\underline{J} = \underline{J}^{(1)} + \underline{J}^{(2)}$

10) $\underline{J}_{L} = \underline{J}$

11) $\underline{i}'_{L} = \underline{J}_{L}$

equivalent system. The interconnection step.

Step 7: Gives the branch potentials of the mesh equivalent system.

Step 8: Transforms the mesh equivalent potentials to the component of solution $\underline{J}^{(2)}$ to give a mesh current vector for each of the sub networks.

Step 9: Combines the two current components of solution.

Steps 10 and 11: The node to datum current sources have to be restricted to tree branches in this thesis and therefore $\underline{J}$ calculated in step 9 is in fact the individual mesh flows $\underline{i}'$ of the sub networks.

3.10.4 Summary

From the mathematical development it is perhaps difficult to see the wider implications of applying the matrix methods to complex pipe network problems. These are discussed later.

It has been shown that network problems can be solved by four alternative methods. It is assumed that each of the primitive sources of current and potential are completely specified at the outset of the problem. Two methods solve the node to datum potentials of a network problem. One of these methods does so by evaluating the potential component due to all of the tree branches, then adding to this a potential component which is due to the interconnection of all of the link branches in one operation. The second node to datum potential method is the diakoptics method. This may be regarded as an extension of the first method. Instead of interconnecting all of the link branches in one operation, diakoptics uses the first method to interconnect the links of the sub networks and then interconnects the sub networks themselves. The two other methods solve the network problem in terms of the individual mesh flows. The orthogonal mesh method solves the mesh flows by interconnecting the tree and link branches in one operation. The mesh diakoptic method solves a component of the overall mesh flows for each of a series of sub networks, these sub networks are then themselves interconnected to provide the overall solution.

It is shown later that the concepts of sequential interconnection may be further expanded.

3.11 Interconnection by Groups

Consider the solution of network problems by nodal diakoptics. The development has assumed that the solution equations to the radial equivalent are interconnected in a single operation. It is conceivable that in some circumstances it is undesirable to interconnect the subdivision solutions in one complete operation. For instance, when there are a large number of cut branches, the size of the interconnection matrix will be large, and its inversion time consuming.

In these cases the equations of diakoptics may be expanded. Consider a large network which has been divided into four subdivisions by a large number of cut branches. The radial equivalents of two sub networks may then be interconnected by using the cut branches which interconnect the two subdivisions. The radial equivalent equations of the single sub network produced may then be evaluated, and stored. The process can then be repeated for the remaining two of the original sub networks.

Two further sub networks are therefore produced, and their equations in terms of radial equivalents may be interconnected using the information about the remaining cut branches. This procedure of interconnection by groups can be extended to any degree, limited only by the number of cut branches between particular sub networks.

3.12 Extension of Network Concepts

Middleton has shown (6) that to achieve a solution to a network of b branches, 2b primitive source quantities must be specified. The node to datum flow $\underline{I}'$ constrained to tree branches only automatically specify the primitive current sources associated with the m link branches as zero. The remaining 2b-m quantities must be specified by the engineer at the outset of the problem. They are usually made up of b-m node to datum currents and b potential sources.

Designers, especially chemical engineers, often wish to specify a combination of primitive potential sources, node to datum flows and node to datum potentials. However if the designer chooses not to specify a particular node to datum source, then by definition each of the primitive branches along which that node to datum source is assumed to act will have its primitive current source incompletely specified. This is because the primitive current source in a single branch of a network is the result of the summation of the effects of all the node to datum sources which act along that branch.

If the designer specifies, as an alternative to a node to datum current source, a node to datum potential source, the primitive branch sources will remain incompletely specified and the problem cannot be solved by the methods outlined so far. For example, if the node to datum flow at node 1 of the network of figure (3.7.1) is not specified the primitive current sources in branches 1, 3 and 8 cannot be completely specified, and the problem is insoluble, even if the node to datum potential has been specified. This is because there is no known method of transforming node to datum potentials $\underline{e}$ to primitive current sources $\underline{I}$, unlike the primitive potential sources $\underline{E}$ which may easily be transformed to primitive current sources.

From the opening sentence of this chapter it is quite obvious that the addition of a further branch to an existing completely specified network necessitates the specification of a further primitive source before a solution can be attempted. This fact can be used to advantage. If a branch which consists of a potential source $\underline{E}$ only is added to the network between the datum node and any other, then this latter node will be maintained at a potential equal to the value of the potential source. This potential source can also be transformed to a current source I.

It is now necessary to suppose that the node to datum path which is associated with the potential specified node is neglected and that it is transferred to branch added to the network. In real terms this means that the primitive current sources associated with the branches of the network become once again fully specified, since the unspecified node to datum current now acts along the additional branch. For a single branch the primitive potential source can be easily transformed to a current source and vice versa.

The source vector $(\underline{E} - \underline{Z}.\underline{I})$ of equation (3.8.14) can thus be fully specified for a network even if the node to datum current sources are not fully specified. The addition of the further branch however increases the size of the problem, necessitating greater amounts of computation.

There is a further consideration concerning the additional branch. If the node to datum potential of that branch is to be a certain specified value determined at the outset, and the branch has been assumed to have a source, either of potential or equivalent current, of the same value as the node to datum potential, then it is quite obvious that e for the branch must be zero. This means that the additional branch must have a zero impedance.

A worked example of solving a network problem which has one of its node to datum potentials specified is shown in appendix (6). A complete program called NPSMESHFLAN has been written which is able to solve the type of problem outlined above and is discussed in Chapter (4).

From the development outlined in this chapter the writing of a computer program to implement the method appears to be a difficult task since it is inferred that a new node to datum path and hence a new set of meshes has to be defined for every potential specified node. It is shown below that this is not necessarily the case and that implementation of the method is quite simple.

The addition of a further branch to a network must create an additional mesh. If as outlined above the additional branch has been specified as the node to datum path of the potential specified node then the first branch in the original node to datum path of that node must become the link branch of the additional mesh. If all the node to datum sources, both $\underline{I}$ and $\underline{e}$, are then specified together as a vector and the pressure source in the additional branch is specified as zero, then the orthogonal mesh flows may be solved, but in terms of the modified node to datum and mesh equations.

The same result can be achieved more simply. Maintain the node to datum paths and mesh paths as for a complete node to datum current specification. Assign the value of the unknown node to datum current as zero. The primitive current sources in all of the branches of the network are therefore completely specified. The node to datum potential is then treated as a potential source in the additional branch. Since no current due to external sources flows down either the additional branch or the first branch in the node to datum path, then in fact either of these branches may be assumed to be the link branch of the additional mesh.

Therefore the nodal potential specified problem may be solved by setting up the equations as for the complete current source specification.

with the unknown node to datum currents specified as zero. A branch of zero impedance containing a potential source equal to the value of the potential specified at the node is then added to the network as a link branch. With the result that only an additional mesh path must be evaluated for solution, the node to datum paths remain the same. In Chapter (4) it is shown how this procedure may be performed automatically by a computer program.

3.13 Non Linear Systems

A variety of methods of solving linear network problems has been proposed. It has been implicit in the development of these methods that the branches are of constant admittance or impedance. This is not, of course, the case for fluid networks, and it is necessary to propose an iteration scheme for their solution.

For a single pipe the pressure drop between two nodes expressed in standard engineering notation is given by equation (3.13.1),

$$\Delta P = 4\phi \frac{L}{D} \rho U^2 \qquad (3.13.1)$$

the velocity term in (3.13.1) can be re-expressed to give equation (3.13.2),

$$\Delta P = 4\phi \frac{L}{D} \rho \frac{Q^2}{A^2} \qquad (3.13.2)$$

this equation can be expressed in for form of equation (3.4.2) as in equation (3.13.3),

$$\Delta P = \left(\frac{4\phi L \rho Q}{D \;\; A^2} \right) Q \qquad (3.13.3)$$

It follows that from equation (3.13.1) an expression for $\phi^{\frac{1}{2}}$Re may be given by equation (3.13.4),

$$\phi^{\frac{1}{2}} \text{ Re} = \left(\frac{\Delta P \; D^3 \rho}{4L \mu^2} \right)^{\frac{1}{2}} \qquad (3.13.4)$$

and it is known from the work of Colebrook and White (16) that the friction factor for turbulent flow in smooth and rough pipes is given by equation (3.13.5),

$$\phi^{\frac{1}{2}} = 2.5 \ln\left(\frac{\varepsilon/D}{3.7} + \frac{1}{1.13\phi^{\frac{1}{2}}\,Re}\right) \qquad (3.13.5)$$

At the start of computation an initial estimate is made of the flow Q and the friction factor ϕ for every pipe of the network. Then, using any of the solution equations proposed, the branch pressure drop is calculated. The value of the friction factor is then recalculated using equation (3.13.4) and branch flows recomputed with a changed admittance or impedance. The iteration cycle may be repeated until a desired degree of convergence is obtained.

3.14 Convergence of Non Linear Solutions

Various methods have been proposed as efficient convergence tests. Middleton (6) suggests a convergence test of the form of equation (3.14.1),

$$\sqrt[2]{\sum_{i=1}^{n} (\Delta \underline{e}')^2} \leqslant \text{limit} \qquad (3.14.1)$$

This implies that the square root of the sum of the squares of the differences of the node to datum pressures between two successive iterations is less than a value chosen by the user. This has the drawback that convergence is not uniformally applied across the network, since if some nodal pressures are much larger than the majority, the difference between successive iterations is likely to be large and convergence therefore slow. Alternatively if some of the pressures are smaller than the majority the difference between two successive iterations will be small and the system will appear to have converged according to equation (3.14.1), but some nodal pressures may be wrongly evaluated. The method also assumes that the node to datum pressures are calculated. In engineering applications attention is more usually focussed on the flows in, and the pressure drops along a pipe. Bixby (41) using a modified Hardy Cross balancing heads method, proposes the criterion that the correction ξ for every mesh of a converged system be less than 50 cu ft. This has the obvious drawback that for low flow systems, an unconverged solution will satisfy the criterion. Bixby maintains however that since the criterion is specific to high flow gas distribution systems, it is quite adequate for that purpose. The convergence criterion used by the author is based on the difference of two successively iterated branch flows according to equation (3.14.2),

$$\frac{|\Delta J_i|}{J_i} \leqslant \text{Convergence Criterion} \qquad (3.14.2)$$

$$i = 1 \ldots BR$$

The difference between two successively iteration branch flows must be less than a percentage of the most recently calculated value of the branch flow for every pipe of the network before a solution is assumed to be converged.

This system has one disadvantage which is that if the value of the flow in some of the branches is oscillating markedly whilst the others may have nearly converged, the overall convergence test will never be satisfied. This event may happen in the case of very small pipes or very low flows. However from the performance of the programs it has been realised that solutions will converge by any of the four methods to within about 0.1 per cent in approximately thirteen iterations. It is therefore quite easy to write into a program some device which outputs the available information about two successive iterations once a maximum number of iterations has been exceeded.

3.15 Further Applications of Matrix Methods

It was suggested in chapter (3.12) that a variety of network variables may be specified at the outset of a problem. It was also shown that the equations of network analysis could be successfully manipulated to produce a solution to such a problem. This concept can be further extended in a way which is useful to the design engineer.

In design problems, especially those concerned with piping in chemical plants, network systems may be underspecified or more unusually overspecified. Since there are only b solution equations, the 2b variables associated with the ($\underline{E}$ - $\underline{Z}.\underline{I}$) matrix, or its dual, must be specified. If less than 2b variables are specified then simply by looking at the primitive describing equations the designer can determine what further information is necessary to complete the specification. For overspecified systems, which is in fact the original network with node to datum pressures specified as shown in figure (3.12), the designer has a choice. He can respecify the system correctly, or alternatively fictitious branches can be added to the original network to accommodate the overspecification and the problem is then solved.

The solution to some network problems requires the manipulation of admittances or impedances which may contain a mixture of linear and non-linear quantities. Serious computational difficulties can arise with these cases since such systems fail to converge when standard non-linear techniques are used. Diakoptics can be used to advantage in such cases.

For the nodal diakoptics method the non-linear elements may be confined to the link branches and therefore all the linear non variant elements confined to the cutsections. The result is that a solution which has to iterate through the high-level interconnection process only is possible, resulting in a faster iteration cycle.

CHAPTER 4

DISCUSSION OF COMPUTER PROGRAMS

4.1 Introduction

The programs described below were written for an ICL 1905 series machine which is briefly described in appendix (11). For two reasons, the ALGOL language has been used exclusively. The author was more familiar with this language than with any other, and qualitatively it was felt that this language was more suited to the complex dynamic programming needed to handle the matrix manipulations of the proposed programs.

It has been suggested by some authors, (37), (44), (20), that the comparison of computational methods of analysing network problems is for many reasons a difficult task. Four of these reasons are outlined below:

1) Differing machines are used by a variety of authors.

2) Authors have varying experience and ability.

3) Differing methods of attacking the same problems have been used, for example the use of matrix methods by some authors and the use of a variety of friction factor finding procedures.

4) A whole range of widely varying convergence criteria have been used to assess the performance of particular programs.

The difficulty of comparing the relative merits of differing methods of solving pipe network problems is emphasised by the fact that apart from the limited comparison of the Hardy Cross mesh method and the nodal method of diakoptics by Middleton (6), no such comparison exists.

Five programs together with three supplementary programs have been written to compare the efficiencies of five varying methods of solving pipe network problems. It is hoped that such a comparison will obviate some of the difficulties expressed above. The programs may be classed into three separate groups. These are:

1) The Hardy Cross mesh method which has been developed to resemble as closely as possible the program due to Daniel (3), and a modification of this method which incorporates the friction factor finding routine in a different iteration loop to that used by Daniel.

2) The orthogonal nodal and mesh methods equivalent to the classical mesh and nodal methods. This group includes two other programs, one which is very similar to the orthogonal mesh method which has been extended to include the realistic possibility that nodal pressures in addition to nodal flows may be specified at the outset of a problem. The second extension utilises the link at a time algorithm proposed by Branin (21), further discussed in appendix (10), which gives an alternative method of calculating the inverse of the nodal solution matrix.

3) The nodal and mesh diakoptic methods.

4.2 The Hardy Cross Programs

The basic method for solving pipe network problems by the Hardy Cross mesh technique has been outlined in Chapter (2). A detailed flowsheet indicating the methodology of the program HCMESHIN, the author's interpretation of the program due to Daniel (3), is shown in figure (4.2.1). The flowsheet for the modified Hardy Cross mesh method called HCMESHOUT is shown in figure (4.2.2). Only that part of the flowsheet which is different from HCMESHIN is given. The computer listing for these two programs is shown in appendices (12.1) and (12.2). Marked on the flowsheets are BLOCK numbers; these are devices which it is thought will simplify the discussion of flowsheets and listings. It is not intended that the BLOCKS define any specific limits of the area of program occupation within a computer. BLOCK is simply a device into which certain functions of the programs may be grouped.

The reasons for writing the program HCMESHOUT are discussed later, but since it is basically similar to HCMESHIN it is advantageous to discuss the two programs simultaneously.

BLOCKS 1-6 and 14 are common to both flowsheets. They incorporate operations which are basically bookkeeping routines. These organise the information available about the network into a form acceptable to the calculation procedures.

In the discussion of all of the programs below reference is made to the input data required by the programs. For reasons of clarity this is not treated in detail at the present time but a complete discussion of the data required by each program will be found in chapter (4.9).

BLOCK 1 Inputs basic network data, the number of branches, meshes, and the number of branches in the largest mesh. From this data the size of the arrays needed to handle to the hydraulic data and the arrays necessary for the calculation routines is determined and computer storage assigned.

BLOCK 2 Inputs the fluid data of viscosity and density together with the convergence criterion.

BLOCK 3 Inputs the network geometry. Each branch, mesh and node of the network is assigned a reference number by the user, and the data prepared as is discussed later in chapter (4.9). The input is in the form of an integer list.

BLOCK 4 From the information contained in BLOCK 3 a modified branch-mesh incidence matrix is formed.

BLOCK 5 Inputs the initial guesses of flow, with direction relative to that of the branches, as they have been chosen by the user. Also input at this stage are the hydraulic data for the network in terms of the lengths, diameters and relative roughnesses.

BLOCK 6 Outputs all the information about the network which is available to the computer. This serves two purposes. A data check may be easily performed to establish that the data prepared is identical to the data input. A record of particular data input and the final results of an analysis is available together.

BLOCK 14 Outputs for every branch of the network a converged solution in terms of the individual pipe flow, friction factor, length, diameter, relative roughness and Reynolds number. The number of iterations required for solution is also output.

4.2.1 The program due to Daniel (3). HCMESHIN

BLOCK 7 This is the start of the iterative section of the program and is identified in the program listing as label L4:. Immediately preceding L4: the integer variable QQ is assigned an initial value of 1 and as iteration proceeds the value of QQ is updated to keep a record of the number of iterations.

It is felt that the use of a sophisticated friction factor finding routine, for branches in which the guess of flow is likely to be inaccurate, is computationally wasteful. On the first iteration cycle therefore an estimated value of the friction factor of 0.002 is assigned to each branch. From the results shown in Chapter (5) it can be seen that for the smooth pipes of water networks this is a reasonable value. On subsequent iteration cycles a new friction factor is determined using the Colebrook White equation by the procedure FINDPHI, which is described at a later stage.

BLOCK 8 For each branch of the network R' is evaluation using equation (4.2.1),

$$R' = \frac{2\phi L \rho}{D^5} \qquad (4.2.1)$$

A second vector of flow Q3 is set equal to the initial guess of flow Q1. This array Q3 is used later in the program to test for convergence.

BLOCK 9 The start of this BLOCK is associated with label L5:. A further flow vector Q2 which is used in the convergence tests is set equal to the initial guess or the last calculated value of flow. For a single mesh of the network this block finds the correction factor ξ to be added to each branch from equation (2.2.9).

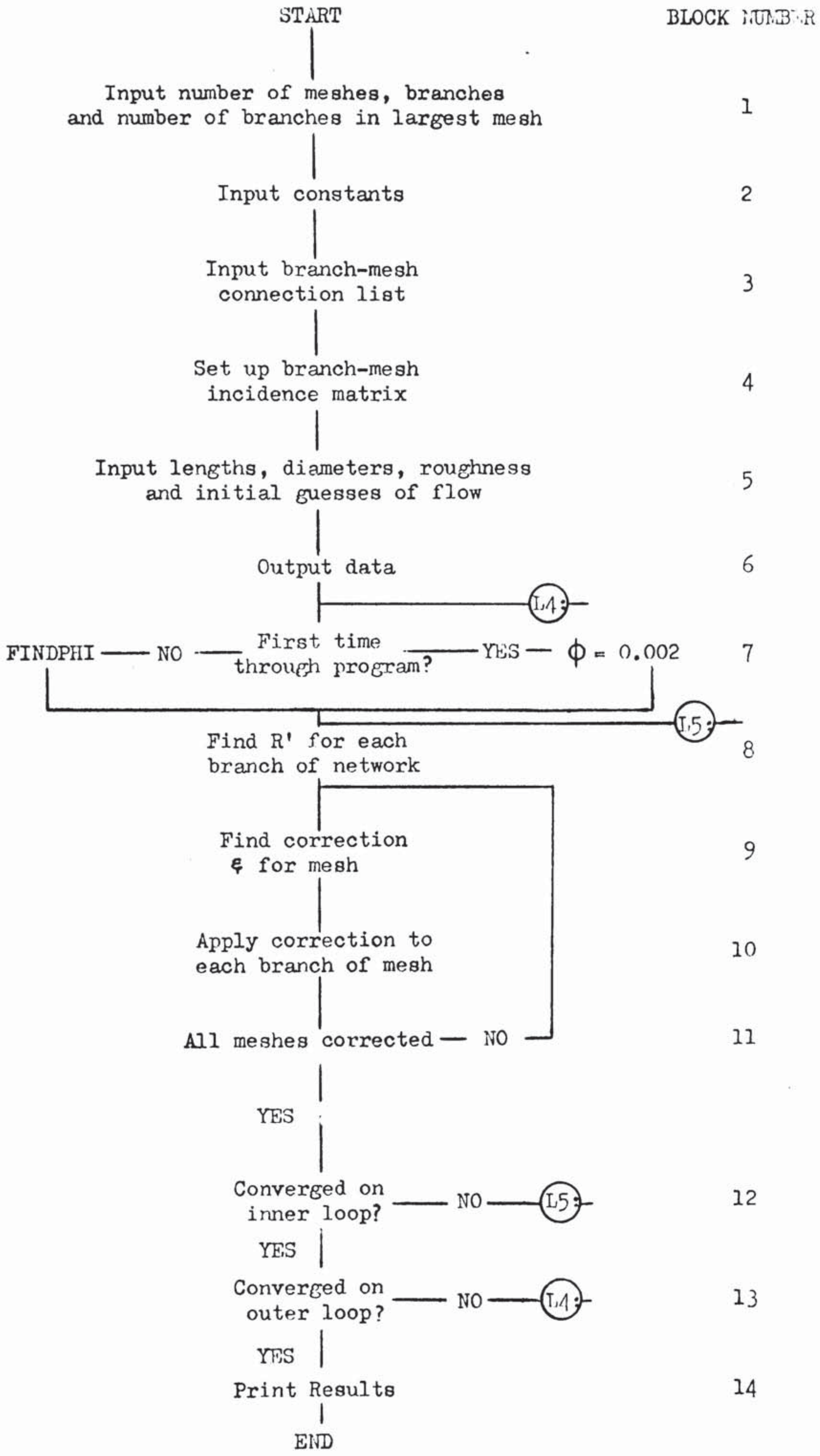
BLOCK 10 Adds on to the value of flow in each branch of a mesh the value of ξ called ERROR in the program, with due regard for the sign of ξ.

BLOCK 11 This takes the form of a counter which notes the number of meshes corrected. In the program it is in fact an ALGOL 'FOR' loop. If all the meshes have not been corrected, control returns to BLOCK 9 for the next mesh.

BLOCK 12 This tests whether or not the difference between the most recently calculated flow Q1 (from BLOCK 10) and the uncorrected flow Q2 is less than a percentage of Q1, determined by the convergence criterion, for every branch of the network. If one or more branches of the network do not satisfy the criterion, control passes back to BLOCK 8 (via L5: on the flow-sheet) which recorrects the first mesh etc. If convergence on this iteration loop is satisfactory then control is passed on to BLOCK 13.

BLOCK 13 This is a further test which determines if convergence is satisfactory for the outermost iteration loop. The value of the newly corrected flow Q1, which has been calculated by iteration with a constant value for the friction factor, is compared with the value of flow Q3. The value of Q3 is either the initial guess of flow or the value of flow calculated at the previous constant value of friction factor. If the difference between the two flows is not less than a percentage of the corrected flow, determined by the convergence criterion for all of the branches, then number of branches which have converged is output and control is returned to BLOCK 7 (via L4:). A new friction factor is determined and the iteration procedure restarted. If convergence is satisfactory for all of the branches, then control continues to the output procedure of BLOCK 14.

Figure (4.2.1) The Flowsheet for HCMESHIN


START
BLOCK NUMBER
Input number of meshes, branches and number of branches in largest mesh
1
Input constants
2
Input branch-mesh connection list
3
Set up branch-mesh incidence matrix
4
Input lengths, diameters, roughness and initial guesses of flow
5
Output data
6
L4
FINDPHI
NO
First time through program?
YES
$\phi = 0.002$
7
L5
Find R' for each branch of network
8
Find correction ϵ for mesh
9
Apply correction to each branch of mesh
10
All meshes corrected
NO
11
YES
Converged on inner loop?
NO
L5
12
YES
Converged on outer loop?
NO
L4
13
YES
Print Results
14
END

4.2.2 Modification of the Daniel program HCMESHOUT

The construction of BLOCKS 7-11 of figure (4.2.2) is basically similar to those of HCMESHIN except that the routing of the return of control for the iteration cycles is modified.

BLOCK 7 An initial value of 1 is assigned to QQ, the counter which notes the number of iteration cycles. The label L4: determines the start of the iteration procedure. On the first iteration a friction factor of value 0.002 is assigned to each branch of the network. On subsequent iteration cycles the friction factor is determined using the procedure FINDPHI.

BLOCK 8 Determines, using equation (4.2.1), R' for every branch of the network. A second vector of flow Q2 is also established and set equal to the present value of the branch flow which is to be corrected.

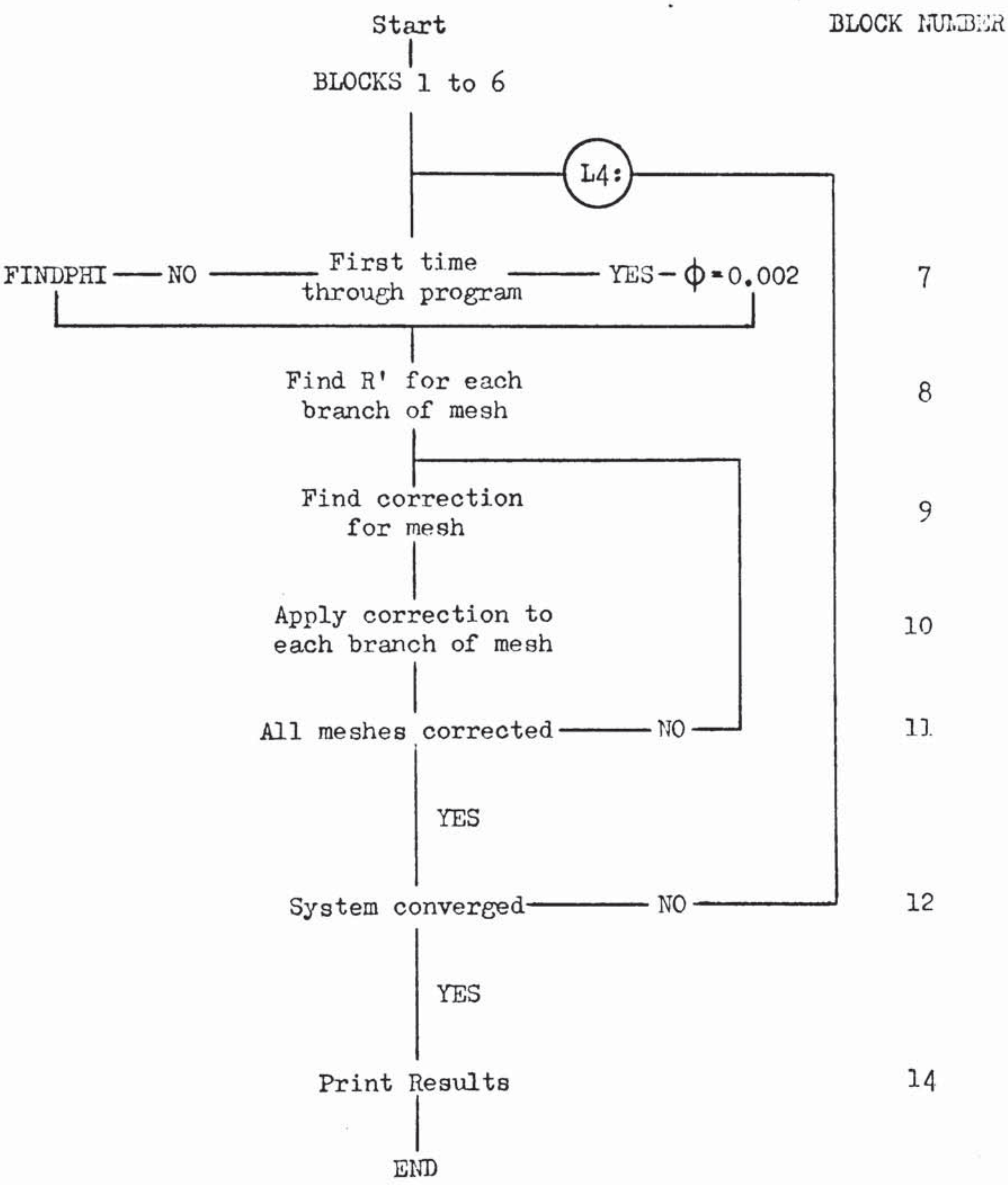
BLOCK 9 Using a modification of equation (2.2.9) the correction ξ called ERROR in the program is determined for a single mesh.

BLOCK 10 With due regard for the its sign the correction is applied to each branch of the mesh.

BLOCK 11 Determines if all the meshes have been corrected. If not, control is returned to BLOCK 9 and the next mesh treated. Otherwise control passes automatically to BLOCK 12.

BLOCK 12 The newly corrected value of the branch flows is stored in the computer as the array Q1. The convergence test is applied to the difference between Q1 and Q2 for every branch. If the absolute value of the difference is not less than a percentage, determined by the convergence criterion, of the flow Q1 for any of the branches then, having output the number of branches converged, control is returned (via L4: on flowsheet) to BLOCK 7 and the iteration cycle repeated. Otherwise control is

Figure (4.2.2) The Flowsheet for HCMESHOUT



transferred to the output procedure of BLOCK 14.

The difference between HCMESHIN and HCMESHOUT can be discussed by reference to the two sets of program description notes outlined above. Chronologically, HCMESHIN was the first network program written by the author, and it was intended that the program be constructed to resemble as closely as possible the Hardy Cross program due to Daniel (3). It was felt that this Hardy Cross mesh analysis was representative of the existing programs, and that it could be used as a comparison with the programs written as a result of the matrix method studies. However in studies of the efficiencies of the matrix programs it became clear that it was advantageous to calculate the friction factor after each iteration and not only after finding a converged solution with a constant friction factor. The reason for this phenomena is not fully understood, but it is obviously related to the number of operations required to find a realistic friction factor. The calculation of these friction factors using the Colebrook White equation necessitates the use of a separate iteration procedure because of the nature of the equation. Therefore the use of the equation at every iteration would be time consuming. This is probably the reason that Daniel (3) calculated the friction factors on the outer iteration loop only. It was found that calculation of only a first approximation of the friction factor at every iteration was sufficient to achieve a converged solution and eventually a correct value of the friction factor, in a more rapid time than iterating the friction factor on an outer loop only.

From this knowledge it was clear that the program due to Daniel (3) might be improved. It then became necessary to produce the improved program HCMESHOUT. One of these reasons was to determine exactly the improvement in convergence that could be found for the Hardy Cross method and to determine whether or not the improvement was as marked as with the matrix methods. The second reason was that it would be scientifically unwise to attempt a comparison between the unimproved Hardy Cross program and the newly developed matrix programs. The significance of the improvement of convergence is discussed in Chapter (5).

The proposed improvement was applied in the following manner. In HCMESHIN the first test for convergence is applied at BLOCK 12 and if the system is not converged then control is returned via L5: to BLOCK 8, which then recalculates the mesh corrections for constant R' (hence constant friction factor). If the flows in this inner loop have converged a further convergence test is applied at BLOCK 13 and for an unconverged solution a new friction factor is calculated in BLOCK 7 and iteration on the inner loop recommenced. In HCMESHOUT a single convergence test is applied at BLOCK 12 and for an unconverged solution control is transferred immediately via L4: to BLOCK 7 to determine realistic friction factors.

Apart from any improvement to be gained by the modification a further small computational advantage for HCMESHOUT is achieved since the convergence test is applied fewer times.

4.3 NODEFLAN

The basis of the program, a listing of which is given in appendix (12.3), is the solution of the equation which gives the node to datum pressures of a network given the pump terms, pipe admittances and node to datum flows. The equation is solved iteratively since the pipe admittances are dependent on flow and friction factor. To start the iteration procedure an estimate has to be made of individual branch flows and friction factors. Fortunately the estimation of initial values is not critical, as is shown in Chapter (5), nor is there any necessity that the initial flow distribution satisfy Kirchoffs First Law.

The input procedures used in NODEFLAN are used in three other programs which are described at a later stage. The similarity of the inputs to each of the programs is more than a computational phenomena. Data preparation is identical for the four programs. This reduces the amount of user time required to produce input for differing programs, and also aids comparison of the performance of the programs, since the computation time spent reading data is identical.

The program, the flowsheet of which is shown in figure (4.3.1), is constructed as a series of self-contained procedures as is explained below.

BLOCK 1 Inputs the number of nodes and branches. From this information the program sets up the array storage required for computation.

BLOCK 2 Inputs the lengths, diameters, relative roughness and pump terms for every pipe of the network. This information is stored in the labelled arrays set up in BLOCK 1.

BLOCK 3 Inputs the network geometry as a connection list. This list is a two column integer list which is described in appendix (1).

BLOCK 4 Inputs the fluid data which are the node to datum fluid flows assigned to each node, the density and viscosity of the fluid, the value of the constant $\barwedge$ and the convergence criterion. Input as the last piece of data is the datum node pressure. The node to datum solution equation generates the node to datum pressures and therefore the pressure at any node will be relative to the pressure at the datum node. Rather than leave the datum node at a value of zero units of pressure, it was decided to give the user the choice of specifying the datum node pressure. This has been achieved with only a small increase in computational effort.

BLOCK 5 Assigns to each pipe an initial estimate of flow and friction factor.

BLOCK 6 Using the procedure INPUTS the available information about the network system presented to the computer is output to the Lineprinter. This facilitates error detection and records the input data for correlation with the results.

BLOCK 7 This is the main computation section of the program and can be identified with label L1: on both the flowsheet of figure (4.3.1) and the listing of appendix (12.3). Immediately preceding L1: the counter Q is assigned an initial value of 1. By updating the value of this counter by 1 at each iteration, a record is kept of the number of iterations performed by the program.

The main calculation routine consists of the two procedures FORMY and EVALUATE; these compute the algorithm of figure (4.3.2), which is constructed as a series of Steps.

Step 1 is performed by the small procedure FORMY which finds the admittance Y for every branch of the network. Steps 2 to 8 are computed by EVALUATE.

From the branch admittances the $(\breve{\underline{A}}.\underline{Y}.\underline{A})$ matrix is computed. It would be expected that to obtain such a matrix two sets of matrix multiplications $(\underline{Y}.\underline{A})$ and $(\breve{\underline{A}}.(\underline{Y}.\underline{A}))$ would be required. This would be computationally time consuming. From the worked example in appendix (5) it can be seen that $(\breve{\underline{A}}.\underline{Y}.\underline{A})$, for pipe networks which have no mutual admittance between the primitive branches of the network,has a special form. The matrix is square and has the dimensions of non datum nodes. The diagonal elements are the sums of the admittances of the branches which are incident at the node whose reference number corresponds to the row or column number of the matrix. The off diagonal terms are the negative of the admittances of the branches which run between two nodes. All other elements of $(\check{\underline{A}}.\underline{Y}.\underline{A})$ are zero.

It is therefore a more simple procedure to set up $(\check{\underline{A}}.\underline{Y}.\underline{A})$ directly using the information about the branch admittances and the information contained in the connection list.

Obtaining the inverse of $(\check{\underline{A}}.\underline{Y}.\underline{A})$ for Step 3 is a computationally time consuming task, but since the matrix has the following properties, its inversion may be speeded up. The matrix has its largest elements on the diagonal and is most usually very sparse, it is also symmetric. The procedure which takes advantage of these properties and is used in the program to find the inverse of $(\check{\underline{A}}.\underline{Y}.\underline{A})$ is called MYTRIX.

The solution matrix has to be inverted many times in the case of fluid flows because of the iteration technique required. The importance of finding an efficient inversion routine which takes advantage of the properties of the matrix is great. Different inversion routines have been investigated by the author and are discussed in chapter (4.10).

Step 4 of the algorithm is easily achieved by the multiplication of two column vectors $\underline{Y}$ and $\underline{E}$, from which $(\check{\underline{A}}.\underline{Y}.\underline{E})$, also a column vector, can be constructed by selecting, with reference to the connection list, the correct elements from the $(\underline{Y}.\underline{E})$ vector. In the theory of network analysis of chapter (3.8), the admittance matrix $\underline{Y}$ is square, but since there are no mutual admittances between any of the primitive branches, it has a very simple form. The diagonal elements of the matrix are the only non-zero entries and therefore this information may be stored in the computer as a column vector.

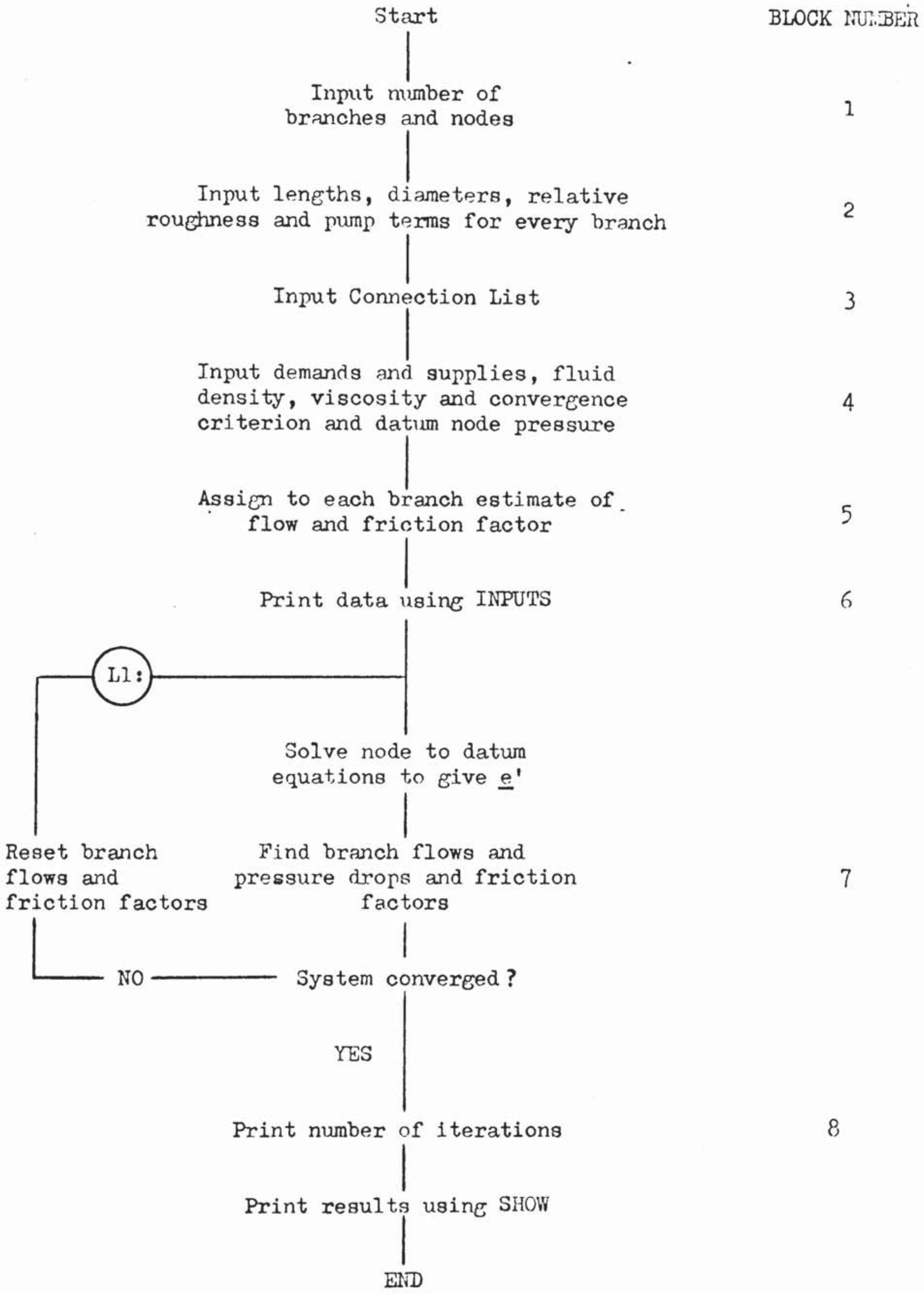
Step 5 is computed by subtracting $(\check{\underline{A}}.\underline{Y}.\underline{E})$ from the column vector of node to datum flows (the demand and supply vector). Premultiplication of the matrix of Step 5 by the inverse of the node to datum solution matrix yields the solution for the node to datum pressure computed for the initial values of branch flow and friction factor.

From the node to datum pressure the vector of individual branch pressures e can be found from Step 7 and the overall branch pressure drop found by summing the effect of pump sources and pressures as in Step 8. The program then uses the overall branch pressure drop to find the Reynolds Number for each branch. If the flow is in either the laminar or transitional region, a message is printed out accordingly. A new value of the friction factor and of the flow is then determined for every pipe.. A percentage convergence test as described previously is then applied. The number of branches which are assumed to have converged is printed out and if convergence is not satisfactory, the newly calculated values of flow and friction factor are used to restart the iteration cycle, via L1:, otherwise control is passed to the procedure which prints the converged results called SHOW.

BLOCK 8 Immediately preceding SHOW the number of iterations that the solution required to converge is printed out. The procedure SHOW then prints the pipe reference number, the node from which direction the pipe flow is orientated, the node to which the flow is orientated, the flow, the pressure drop along the pipe and the friction factor in the pipe. The procedure then prints a node reference number and the pressure at that node.

The program then terminates.

Figure (4.3.1) The Flowsheet for NODEFLAN


Start
BLOCK NUMBER
Input number of branches and nodes
1
Input lengths, diameters, relative roughness and pump terms for every branch
2
Input Connection List
3
Input demands and supplies, fluid density, viscosity and convergence criterion and datum node pressure
4
Assign to each branch estimate of flow and friction factor
5
Print data using INPUTS
6
L1:
Solve node to datum equations to give e'
Reset branch flows and friction factors
Find branch flows and pressure drops and friction factors
7
NO
System converged ?
YES
Print number of iterations
8
Print results using SHOW
END

<u>Figure (4.3.2)</u> <u>The algorithm for the method of calculating node to datum pressures</u>

STEP NO

1 $$Y_k = \frac{1811.25.D_k^5.\pi^2}{J_k.L_k.\phi_k.\rho}$$

k = 1,2 .. number of branches

2 $(\underline{\breve{A}}.\underline{Y}.\underline{A})$

3 $(\underline{\breve{A}}.\underline{Y}.\underline{A})^{-1}$

4 $\underline{\breve{A}}.\underline{Y}.\underline{E}$

5 $(\underline{I}' - \underline{\breve{A}}.\underline{Y}.\underline{E})$

6 $\underline{e}' = (\underline{\breve{A}}.\underline{Y}.\underline{A})^{-1}.(\underline{I}' - \underline{\breve{A}}.\underline{Y}.\underline{E})$

7 $\underline{e} = \underline{A}.\underline{e}'$

8 $\underline{V} = \underline{E} + \underline{e}$

4.4 MESHFLAN

BLOCKS 1 to 3 of figure (4.4.1), which is the basic flowsheet for MESHFLAN, the listing of which is given in appendix (12.4), are similar to those of the flowsheet for NODEFLAN as shown in figure (4.3.1).

The remaining BLOCKS of the flowsheet organise the input data, print it out on the line printer as a data check and for referencing purposes, and set up the equations of mesh flow analysis and solve these equations. The solution equation for the mesh orthogonal method is equation (3.8.12). The main procedure of the program called EVALUATOR solves this equation for the mesh flows by an iteration procedure. The basic algorithm for one iteration loop is given in figure (4.4.2). The implementation of this algorithm is a computationally more difficult task than the solution of the nodal equation. Both the $\underline{C}_T$ and $\underline{B}_T$ matrices have to be established before computation can begin.

It has been shown that $\underline{B}_T$ may be obtained from equation (3.3.3) and that $\underline{C}_T$ may be obtained from equation (3.3.6). The calculation of the inverse of $\underline{A}_T$, needed to find $\underline{B}_T$, is a time consuming task, and so the alternative method of generating $\underline{B}_T$ from the connection list is used. This procedure, developed in appendix (1), called SEARCHBT, has been shown to be a computationally more efficient method of generating $\underline{B}_T$ than inversion of $\underline{A}_T$.

The bulk of the program can now be described.

BLOCK 4 Using the input information of BLOCK 3 the procedure SEARCHBT is called and the $\underline{B}_T$ matrix established. The procedure MAKEC is next called and using the information about $\underline{B}_T$ and $\underline{A}_L$, from the connection list, the tree branch mesh incidence matrix $\underline{C}_T$ is set up. The unit matrix $\underline{C}_L$ is also established. It has been argued previously in this thesis that the special form of $\underline{C}_L$ allows the matrix to be omitted from equations, calculations and storage space. However it is established in MESHFLAN to enable the solution of the equations as if they were in the classical unpartitioned form. For a more efficient program, accompanied by an increase in available storage, the elimination of $\underline{C}_L$ is a trivial task.

From a knowledge of node to datum demand and supply vector $\underline{I}'$ the primitive tree current sources $\underline{I}_T$ are established using $\underline{B}_T$. The primitive source vector $\underline{I}_T$ is then stored as a constant constant.

Using the information about the primitive source vector, the initial flows are assigned to each branch of the network. The tree branches are assigned a flow equal to the primitive current source plus 1 cu ft/min and the link branches assigned a flow of 1 cu ft/min. A value of 0.002 is assigned to each branch as the friction factor.

BLOCK 5 Inputs the fluid properties and convergence criteria. Two constants X and Y are next evaluated. This is an operation to obviate the necessity of recalculating these constants for every branch on every iteration, Y has no connection with admittances, it is used only as a constant in this program.

BLOCK 6 This is a self contained procedure called INPUTS which prints the data available about the network on the lineprinter.

BLOCK 7 This BLOCK contains the main elements of the iteration loop and corresponds to label L1: in the program listing of appendix (12.4) and in the flowsheet of figure (4.4.1). At label L1: the program calls the procedure FORMZ which finds the impedance of every branch of the network. The steps outlined in the algorithm of figure (4.4.2) are now computed by the procedure EVALUATOR.

Since the $\underline{C}$ matrix exists as a complete matrix $(\breve{\underline{C}}.\underline{Z}.\underline{C})$ cannot be computed in the same way as $(\breve{\underline{A}}.\underline{Y}.\underline{A})$. However, because the primitive impedance matrix $\underline{Z}$ can be represented as a vector (only the diagonal elements of the square matrix are non-zero), $(\breve{\underline{C}}.\underline{Z}.\underline{C})$ may be calculated using the algorithm shown below.

$$\underline{C}.\underline{Z}.\underline{C}\ (i,j) = \sum_{k=1}^{\text{branch}} \underline{C}(I,k)^{*}\underline{Z}(k)^{*}\underline{C}(j,k)$$

$$j = 1,2\ .\ .\ .\ \text{mesh}$$

$$i = 1,2\ .\ .\ .\ \text{mesh}$$

Alternatively if $\underline{C}$ has been retained in list form, as is used in HCMESHIN, then $(\breve{\underline{C}}.\underline{Z}.\underline{C})$ can be found directly since the completed matrix has a special form. The diagonal elements are the sum of the impedances of all the branches making up that mesh which has as its reference number the diagonal element reference numbers. The off diagonal elements are the negative or positive of the impedances of the tree branches, which are common to one or more meshes, depending on whether they are orientated relative to the link branches in a like or unlike manner.

The mesh solution matrix is square, symmetric and sparse and the special inversion procedure MYTRIX may be used to implement step 2 of the algorithm.

Step 3 is accomplished by a vector multiplication using the stored primitive current source vector.

Step 4 can be accomplished in one operation. $\underline{E}_L'$ is derived from the primitive voltage sources $\underline{E}$ and subtraction from this primitive vector of the equivalent voltage sources in $\underline{Z}_T.\underline{I}_T$

Figure (4.4.1) The Flowsheet for MESHFLAN

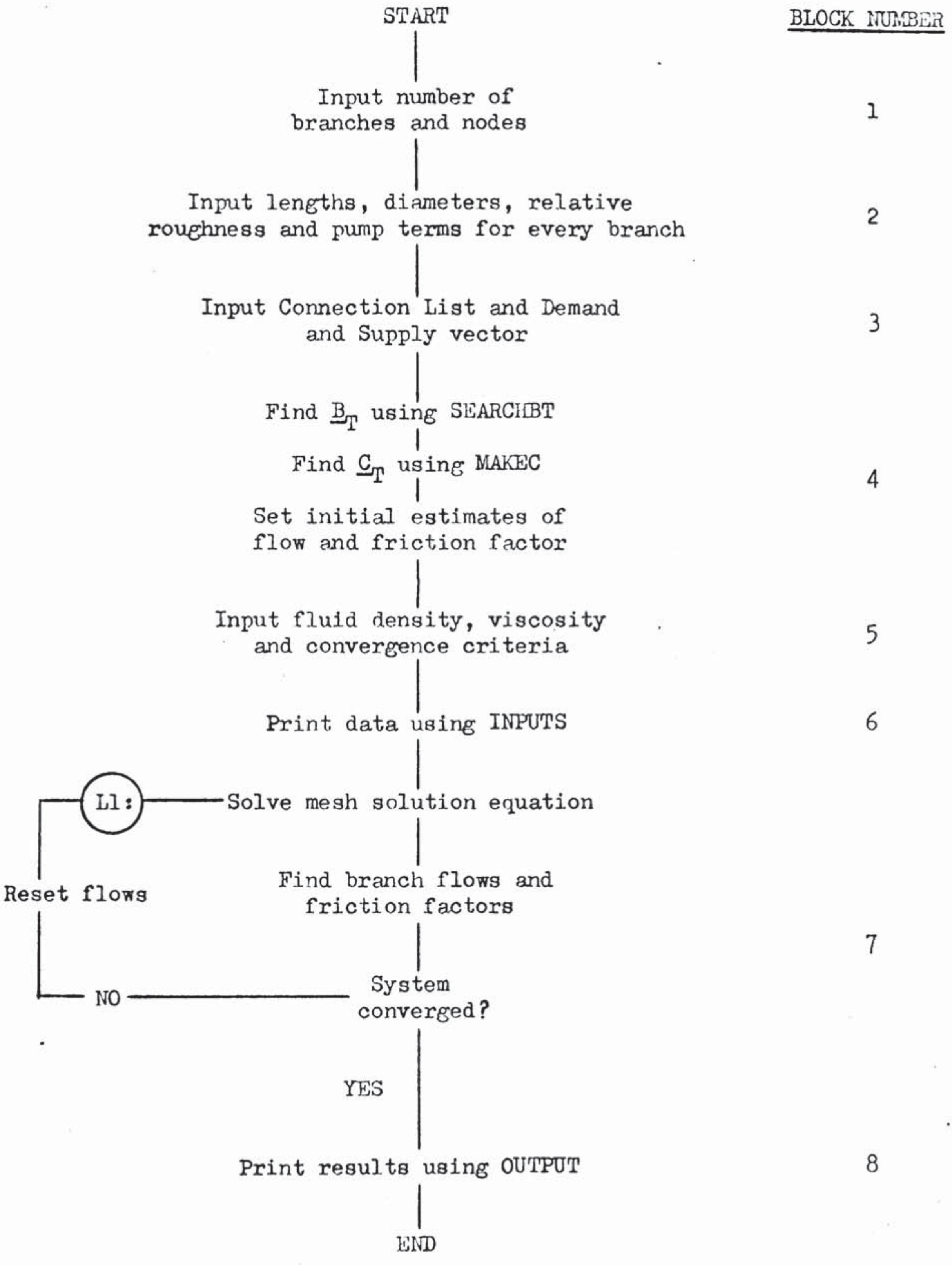

START
BLOCK NUMBER
Input number of branches and nodes
1
Input lengths, diameters, relative roughness and pump terms for every branch
2
Input Connection List and Demand and Supply vector
3
Find $\underline{B}_T$ using SEARCHBT
Find $\underline{C}_T$ using MAKEC
Set initial estimates of flow and friction factor
4
Input fluid density, viscosity and convergence criteria
5
Print data using INPUTS
6
L1:
Solve mesh solution equation
Reset flows
Find branch flows and friction factors
7
NO
System converged?
YES
Print results using OUTPUT
8
END

Figure (4.4.2) The algorithm for solution of network problems by the mesh orthogonal method

Step No	
1	$(\check{\underline{C}}.\underline{Z}.\underline{C})$
2	$(\check{\underline{C}}.\underline{Z}.\underline{C})^{-1}$
3	$\underline{Z}_T.\underline{I}_T$
4	$(\underline{E}'_L - \check{\underline{C}}_T.\underline{Z}_T.\underline{I}_T)$
5	$\underline{i}' = (\check{\underline{C}}.\underline{Z}.\underline{C})^{-1}.(\underline{E}'_L - \check{\underline{C}}_T.\underline{Z}_T.\underline{I}_T)$
6	$\underline{i} = \underline{C}.\underline{i}'$
7	$\underline{J}_T = \underline{I}_T + \underline{i}_T$
8	$\underline{J}_L = \underline{i}'$

gives a vector which is then premultiplied by $\underline{C}$. The full matrix multiplications of step 5 are then carried out to yield the vector of mesh flows $\underline{i}'$. The branch flows $\underline{i}$ due to the mesh flows are then found in step 6.

Step 7 sums for the tree branches the branch flows due to the mesh flows and the primitive current sources. The total flow in the link branch is due only to a single mesh flow as in step 8.

The program next evaluates the pressure drop along every branch and if the flow is in the laminar region ($Re<2100$) or in the transitional region ($2100<Re<3000$) then a message is output accordingly. The friction factor is then evaluated for every branch.

A percentage convergence test is then applied to the individual branch flows and if the system is not converged the number of branches which have converged is printed and control is passed back to the start of BLOCK 7 via L1:. Otherwise control is passed to L2: which labels BLOCK 8.

BLOCK 8 The procedure OUTPUT prints the results in terms of pipe reference number, flow, pressure drop and friction factor. The program then terminates.

4.5 LATTNODEFLAN

This program, a listing of which is given in appendix (12.5), solves the same node to datum pressure equation as NODEFLAN and uses basically the same algorithm with the omission of Step 2 as is shown in figure (4.3.2).

The program uses the untried method of developing the nodal solution matrix ($\check{\underline{A}}.\underline{Y}.\underline{A}$) from a starting point of the tree solution matrix

$(\breve{\underline{B}}_T.\underline{Z}_T.\underline{B}_T)$. The elements of this latter matrix are then updated by an amount given by equation (4.5.1),

$$z^{k+1}(i,j) = z^k(i,j) - \frac{(z^k(i,p) - z^k(i,q)).(z^k(p,j) - z^k(q,j))}{z^k(p,p) + z^k(q,q) - z^k(p,q) - z^k(q,p) + Z_L} \qquad (4.5.1)$$

which determines the effect of adding into the network a single link branch which runs between nodes p and q. The method is further discussed in appendix (10).

To construct $(\breve{\underline{B}}_T.\underline{Z}_T.\underline{B}_T)$ it is of course necessary to have available the $\underline{B}_T$ matrix. It has already been shown in chapter (4.4) that the $\underline{B}_T$ matrix may be established within a computer program from a connection list. The impedances $\underline{Z}_T$ may also be made available since they represent the reciprocal of the admittances $\underline{Y}_T$.

The flowsheet for LATTNODEFLAN is shown in a simple form in figure (4.5.1).

BLOCK 1 This performs the same function as BLOCK 1 of NODEFLAN.

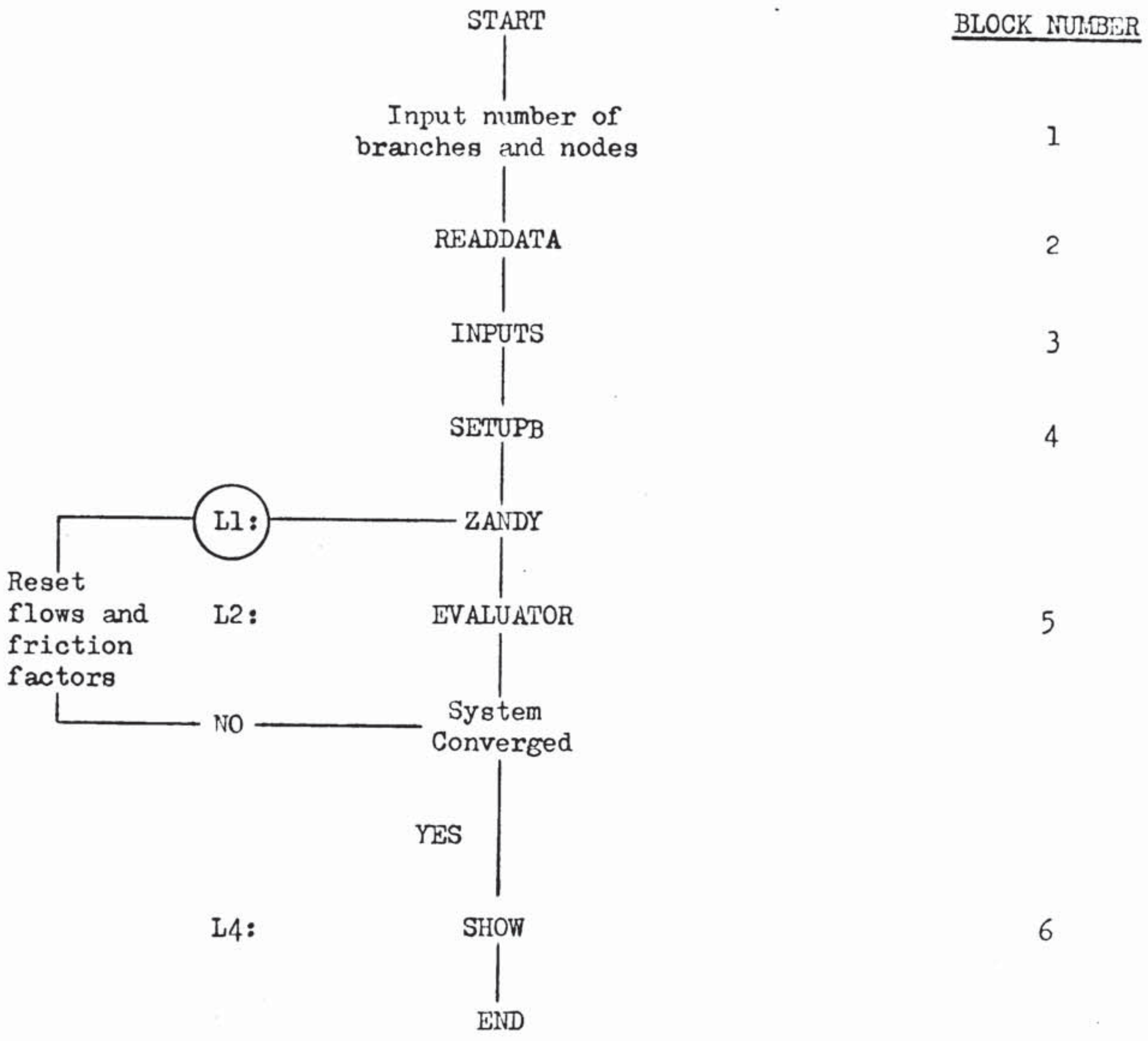
BLOCK 2 This is a self contained routine which assigns the initial guesses of flow and friction factor, and reads in as data the following information: pipe lengths, diameters, pump terms, relative roughness, connection list, node to datum demands and supplies, fluid density, viscosity, the value of λ and the convergence criteria.

BLOCK 3 Uses the routine INPUTS to record together with the expected results the data required for solution.

BLOCK 4 It is shown in appendix (1) that there are two ways of producing $\underline{B}_T$. The more efficient procedure has been shown in MESHFLAN, but for the sake of comparison the method which has now been obviated is shown in this program. The method relies on the straight forward inversion of the $\underline{A}_T$ matrix, by the Guass-Jordan technique. This can only be achieved by partitioning of the $\underline{A}$ matrix by a row changing procedure or by careful referencing of the tree and link branches. The procedure SETUPB is used to perform the inversion.

BLOCK 5 This is the main solution and iteration BLOCK which comprises the two procedures ZANDY and EVALUATOR. Label L1: on the flowsheet and in the listing of appendix (12.5) indicates the start of the iteration loop. Immediately preceding label L1: the counter Q which records the number of iterations performed by the computer is assigned the value of 1. From the initial estimates of flow and friction factor the vector of branch impedance $\underline{Z}$ is established. In this program the $(\breve{\underline{B}}_T.\underline{Z}_T.\underline{B}_T)$ matrix is constructed by using the method proposed in chapter (4.4) for $(\breve{\underline{C}}.\underline{Z}.\underline{C})$ which takes advantage of the fact that the matrix $\underline{Z}_T$ can be expressed as a vector. An alternative method of constructing $(\breve{\underline{B}}_T.\underline{Z}_T.\underline{B}_T)$ can be derived by examining the form of the completed matrix (see the smaller of the two subnetworks of appendix (8)). The elements on the diagonal of the matrix are the sum of the impedances of all of the branches in the node to datum path which has its reference number the same as the column or row of the diagonal element. The off diagonal terms are the sum of the impedances which are common to the two node to datum paths corresponding to the elemental positions of the array. The matrix is therefore symmetric and has its largest elements on the diagonal. With the $(\breve{\underline{B}}_T.\underline{Z}_T.\underline{B}_T)$ matrix constructed each element is then updated as previously described. It is important to note that every element of the matrix has to be updated before the effect of adding in another link branch is considered. For instance for

Figure (4.5.1) The Flowsheet for LATTNODEFLAN



the network of figure (3.7.1) the $(\breve{\underline{B}}_T.\underline{Z}_T.\underline{B}_T)$ matrix is an 8 x 8 and the addition of one link branch requires the updating of 64 elements and the inclusion of the further three link branches requires 192 more updating operations.

After all the link branches have been added into the tree solution matrix the steps 4 to 8 of the algorithm in figure (4.3.2) are evaluated in a similar way to the operations of NODEFLAN. A percentage convergence test is then applied to the individual branch flows and if the system is not satisfactorily converged control passes back via L1: to the start of BLOCK 5 and the iteration cycle repeated, otherwise control automatically transfers to L4:.

BLOCK 6 The procedure SHOW outputs the converged solution in terms of pipe flows, pressure drops and friction factors together with the number of iterations the program has performed.

The three programs NODEFLAN, MESHFLAN and LATTCLAN are compared in terms of their relative efficiencies and their ability to produce accurate results by solving a variety of network problems in Chapter (5).

4.6 NPSMESHFLAN

This program, a listing of which is shown in appendix (12.6), is a modification of MESHFLAN and it is not necessary to produce a flow-sheet to describe it, since this would be very similar to figure (4.4.1). NPSMESHFLAN was designed to solve network problems that have their node to datum pressures specified in addition to node to datum flows. A worked example which shows the mathematics of the method is demonstrated in appendix (6).

A basic requirement of the method is that for every pressure specified node a fictitious branch has to be added to the network. This branch becomes the link branch of an additional network mesh. The pressure at the specified node is then assumed to be a pump source and is included as such in the fictitious branch, which is assumed to have a zero impedance. Computation then proceeds in much the same way as for the straight forward mesh method with the exception that the size of the network describing matrices must be increased.

In NPSMESHFLAN the procedure is accomplished in the following manner, where any discussion of BLOCKS refers to figure (4.4.1). After the operations in BLOCK 1 have been performed, a list which contains the information about pressures at specified nodes is input. The list takes the form of a vector which contains zero entries and non zero entries, the non zero entries corresponding to the pressure that has been specified at a particular node. The program notes the sign of the non zero entry and the reference number of the node at which the pressure has been specified. The program then makes provision for updating the storage which it is to allocate to the network matrices. For instance if there are two nodes that have been assigned pressures, then the dimensions of the $\underline{C}$ matrix are increased by 2 along both axes.

BLOCK 2 is then operated and the network data is input and stored in the network arrays according to the reference number of the branches. It is important to note that only information about the existing pipes of the network has to be input.

The connection list for the existing branches is then input in BLOCK 3, together with the vector of node to datum supplies. Zero entries are made in this latter vector at the positions which correspond to the reference number of the pressure specified nodes.

Immediately preceding the operation of BLOCK 4 the main part of the additional computation required for the method is performed. The information contained in the nodal pressure specified vector called NODPRES is again examined. The non zero entries encountered are used to provide the additional information required to supplement the connection list which is later used to find the meshes of the network. If a positive entry in NODPRES is encountered, the fictitious branch to be added to the network will be directed towards the datum node and away from the pressure specified node. Two entries are thus made in the connection list for every pressure specified node. One entry is positive and corresponds to the reference node towards which the fictitious branch is directed and the other is the negative value of the reference node away from which the fictitious branch is directed.

The value of the non zero entry in NODPRES is then transferred to the enlarged pump term vector as the highest referenced element.

Computation then proceeds almost exactly as for MESHFLAN with the provision that only information about the real branches of the network has to be handled, except in the computation of the additional mesh flows.

In some cases it may be necessary to specify both node to datum flows and a nodal pressure. This is accomplished by assigning non zero quantities to both the node to datum flow vector and the node to datum potential vector in the appropriate element which corresponds to the node reference number.

4.7 NODEDIAK

This program was developed using the algorithm for the solution of network equations in terms of the high-level interconnection of radial equivalent sub networks as described in chapter (3.10.2). The program is similar in many ways to the diakoptics program developed by Middleton (6). Unfortunately however, because of changes in the computer service that operates at the University of Aston, the author was not able to have the "hands on" facility that Middleton was able to use to great advantage.

Middleton was able to test the effect of removing or adding branches and changing the nodal demands or any other network variable using an already converged solution. It is felt that this facility would normally be incorporated in a commercial extension of the diakoptics programs presented. The program described below is inherently more simple than that of Middleton, because of the straightforward once at a time calculation that is performed. Further advantages of using the "hands on" facility are discussed in Chapter (5).

The program, a listing of which is shown in appendix (12.7), is written as a series of self contained procedures. This has the advantage that the different routines may be written and tested separately and the logic of the computation steps is more straightforward. The program may be described in the broadest terms by the following. It finds and stores for every sub network the radial equivalent solution equations. The radial equivalent equations of the sub networks are then interconnected using the link branches and the node to datum potentials evaluated. This process is repeated until a convergence criterion is satisfied, whereupon the results are output. Because information about one sub network only is held in the main core store, the disc backing store must be used. There are special procedures particular to the ICL series which are designed to transfer information from the main store to the discs. These procedures must therefore be regarded as routines which cannot necessarily be transferred to an alternative series of machines. The use of the backing store also means that much of the program relies on the correct functioning of the logic of the basic bookkeeping procedures.

There are three backing store procedures used in this program and their function is described below.

'PROCEDURE' WORKSTORE (N, A, B)

The routine allocates scratch storage on the disc. The entry N allocates a number to a particular file on the disc. Four separate files referenced 1 to 4 are used. The entry A is a string quote ('('ED')') particular to the ICL series and the entry B is a store allocation device. When calling the procedure the file number, string and the amount of store required are entered in the brackets.

'PROCEDURE' GETARRAY (N, A, B) and 'PROCEDURE' PUTARRAY (N, A, B)

These two routines are very similar and are used for the actual transfer of information. PUTARRAY transfers whole arrays from the main store to the backing store and GETARRAY transfers information from the disc to the main core store. The entries in the brackets are N, the file number of the disc storage, A, the name of the array, and B, the starting value of where the array is to be stored or obtained from. One added feature of the B entry is that it updates itself. So that if, for instance, the procedure PUTARRAY is called, the array to be stored will be stored on the disc from the elemental position of B onwards. The result will be that the final value of B will be the next available vacant storage space, and a further array can be stored automatically above the last.

It has been stated that the algorithm presented in chapter (3.10.1) is used as the basis of this program. In fact the computation algorithm is different from the theoretical algorithm, because of the use of computer backing store. The computational algorithms for both NODEDIAK and MESHDIAK are shown in figure (4.7.2). It is hoped that the algorithms written side by side will show the duality of the two approaches.

The flowsheet for NODEDIAK is shown in figure (4.7.1).

BLOCK 1 Inputs the density and viscosity of the network fluid, the value of π, g, and the convergence criterion.

BLOCK 2 Inputs overall network data consisting of the number of cutsections, cutbranches and the total number of nodes.

BLOCK 3 The procedures WORKSTORE, GETARRAY, PUTARRAY are established. The inversion procedure MYTRIX is established. WORKSTORE is then used to assign the storage space required on the disc. The array storage is then assigned to the named arrays.

BLOCK 4 Inputs the cut branch hydraulic data, diameters, lengths, pump terms and relative roughnesses, and also sets up the initial guesses of flow and friction factor for the cut pipes. The $\underline{\gamma}^{**}_{TL}$ matrix indicating the incidence of the cut pipes at the network nodes is input in a special form. This form is a two column integer list similar to the connection list outlined in NODEFLAN. It is important to note that the network, having been subdivided, is allocated local and global references for the nodes and branches. The connection list for $\underline{\gamma}^{**}_{TL}$ refers to the incidence of the cut branches to the global referenced nodes. The complete $\underline{\gamma}^{**}$ matrix need not be set up, since only $\underline{\gamma}^{**}_{TL}$ is used in the solution equations.

BLOCK 5 Inputs for each cutsection the number of branches and nodes in those cutsections.

BLOCK 6 Various counters are used in this program to check the number of iterations, organise the data transfers, and to simplify the matrix multiplications associated with each cutsection. In this BLOCK these counters are assigned their particular initial values and the global arrays are assigned zero values in preparation for the matrix multiplications within the bulk of the program. The label L7: which is associated with the start of this BLOCK is used as the indicator of the start of the main iteration loop.

BLOCK 7 This is the start of the part of the program which finds the equivalent radial equations for each sub network. For an individual cutsection the number of branches and nodes from the input of BLOCK 5 are used to assign the core storage space to the cutsection arrays.

BLOCK 8 This is essentially the main calculation area of the program which uses a procedure called CALCULATE, which is itself made up of a collection of procedures outlined in BLOCKS 8.1 to 8.11. Of these BLOCKS 8.1 to 8.5 organise the handling of the hydraulic data of a cutsection.

On the first iteration CALCULATE instructs the computer to read in the node to datum flows assigned to each node, the pipe diameters, lengths, pump terms and relative roughness for each branch of the cutsection. The cutsection connection list is also input and the initial estimates of flow and friction factor are then assigned. The information about diameters, lengths, pump terms, roughnesses, node to datum flows and connection list is then stored on the disc using the procedure PUTARRAY outlined above.

If CALCULATE is encountered other than the first time for any cutsection then the stored information is brought down from the disc using GETARRAY, ready to be used for computation.

BLOCK 8.6 The information made available to the program about a particular cutsection is printed out, as a data check and as a record to correlate with the results, when CALCULATE is encountered for the first time for any cutsection.

BLOCK 8.7 If the solution in terms of the node to datum potentials for the connected network has converged according to criterion laid down in the procedure TEST which is discussed later, then the flows in every branch of the cutsection are evaluated using FINDFLOW. These flows together with the branch pressure drops and node to datum pressures are then printed out using the procedure RESULTS. The positioning of 8.7 is thought to be the computationally most efficient possible, since after employing RESULTS, control is transferred to the main program at BLOCK 9 and further computation in CALCULATE is not required.

BLOCK 8.8 If CALCULATE is encountered for any cutsection other than the first time, the flows and friction factors in each branch of the cutsection are calculated from the node to datum pressures found on the preceding iteration using the procedure FINDFLOW. Otherwise the initial estimate of flow and friction factor is used for further computation.

BLOCK 8.9 The remainder of the procedure CALCULATE computes steps 1 to 4 of figure (4.7.2). This particular BLOCK uses the procedure ADMIT to compute step 1 for a particular cutsection. As indicated in the discussion of the program NODEFLAN, the $(\breve{\underline{A}}.\underline{Y}.\underline{A})$ matrix does not have to be set up by two matrix multiplications. Advantage can be taken of the knowledge of the final form of the matrix. The connection list and the impedance vector $\underline{Y}$ only being used. The inverse of $(\underline{\breve{A}}.\underline{Y}.\underline{A})$ is computed using MYTRIX and the result stored.

BLOCK 8.10 The procedure EDASH computes steps 2 and 3 of the algorithm and the resulting component of pressure $V^{(1)}$ for a cutsection is transferred to a global array EDASHA which, when all of the cutsections have been considered, will contain all of the components of pressure due to the external sources in equivalent radial form ready for high level interconnection.

BLOCK 8.11 The procedure YBDASH is responsible for forming the radial equivalent part of the high-level interconnection matrix called YB. This is achieved by evaluating step 4 of the algorithm for each cutsection using the particular part of $\underline{\gamma}^{**}_{TL}$ which refers to that cutsection. The matrix computed in step 4 is transferred to the global array YB which is successively updated as each cutsection is considered.

BLOCK 9 This is essentially the terminal statement of the 'FOR' loop which determines whether or not all of the cutsections have been considered. If all the cutsection solution equations have been found, control returns to BLOCK 7 for the next cutsection.

BLOCK 10 The high-level interconnection matrix YB is updated along its diagonal elements by the values of the impedances of the link branches as in step 6 of the algorithm to give the completed interconnection solution matrix. Its inverse is then found

Figure (4.7.1) The Flowsheet for NODEDIAK

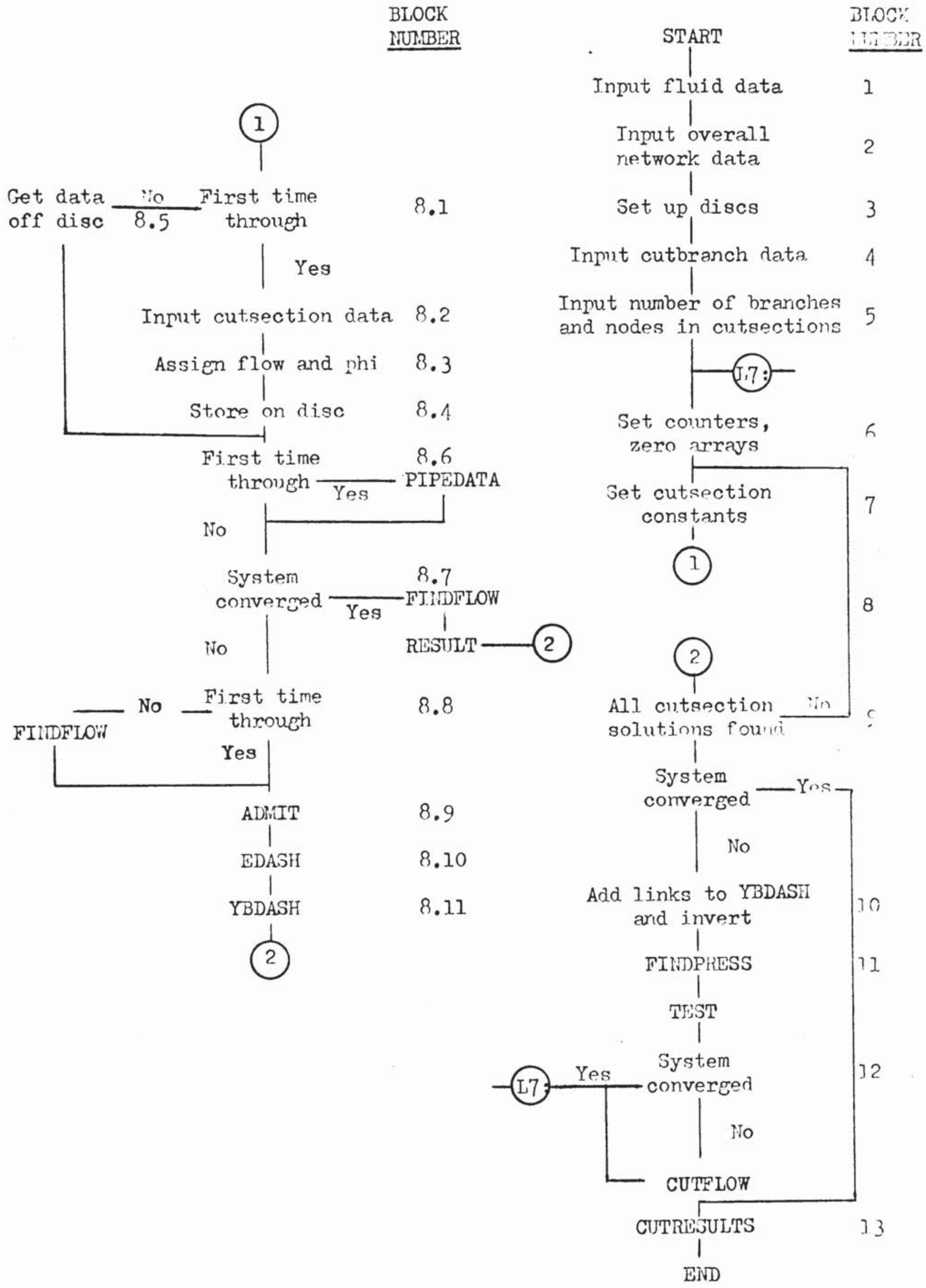


Figure (4.7.2) The computational algorithms of the diakoptic methods

	STEP NUMBER	
$(\breve{\underline{A}}.\underline{Y}.\underline{A})$	1	$(\breve{\underline{C}}.\underline{Z}.\underline{C})$
$(\underline{I}' - \breve{\underline{A}}.\underline{Y}.\underline{E})$	2	$(\underline{E}'_L - \breve{\underline{C}}_T.\underline{Z}_T.\underline{I}_T)$
$\underline{V}^{(1)} = (\breve{\underline{A}}.\underline{Y}.\underline{A})^{-1}.(\underline{I}' - \breve{\underline{A}}.\underline{Y}.\underline{E})$	3	$\underline{J}^{(1)} = (\breve{\underline{C}}.\underline{Z}.\underline{C})^{-1}.(\underline{E}'_L - \breve{\underline{C}}_T.\underline{Z}_T.I_T)$
$(\breve{\underline{\gamma}}^{**}_{TL}.(\breve{\underline{A}}.\underline{Y}.\underline{A})^{-1}.\underline{\gamma}^{**}_{TL})$	4	$\breve{\underline{\alpha}}^{**}_{LT}(\breve{\underline{C}}.\underline{Z}.\underline{C})^{-1}.\underline{\alpha}^{**}_{LT}$
Go to Step 1 if all cutsections not considered	5	Go to Step 1 if all cutsections not considered
$\breve{\underline{\gamma}}^{**}_{TL}.(\breve{\underline{A}}.\underline{Y}.\underline{A})^{-1}.\underline{\gamma}^{**}_{TL} + \underline{Z}_L$	6	$\breve{\underline{\alpha}}^{**}_{LT}.(\breve{\underline{C}}.\underline{Z}.\underline{C})^{-1}.\underline{\alpha}^{**}_{LT} + \underline{Y}_T)$
$\underline{V}^{(1)}_L = \breve{\underline{\gamma}}^{**}_{TL}.\underline{V}^{(1)}$	7	$\underline{J}^{(1)}_L = \breve{\underline{\alpha}}^{**}_{LT}.\underline{J}^{(1)}$
$\underline{E}^{*}_L = \underline{E}'_L - \underline{V}^{(1)}_L$	8	$\underline{I}^{*} = - \underline{J}^{(1)}_L$
$\underline{i}' = (\breve{\underline{\gamma}}^{**}_{TL}.(\breve{\underline{A}}.\underline{Y}.\underline{A})^{-1}.\underline{\gamma}^{**}_{TL} + \underline{Z}_L).\underline{E}^{*}_L$	9	$\underline{e} = (\breve{\underline{\alpha}}^{**}_{LT}.(\breve{\underline{C}}.\underline{Z}.\underline{C})^{-1}.\underline{\alpha}^{**}_{LT} + \underline{Y}_T).\underline{I}^{*}$
$\underline{i}_T = \underline{\gamma}^{**}_{TL}.\underline{i}'$	10	$\underline{e}_T = \underline{\alpha}^{**}_{LT}.\underline{e}'$
$\underline{V}^{(2)} = (\breve{\underline{A}}.\underline{Y}.\underline{A})^{-1}.\underline{i}_T$	11	$\underline{J}^{(2)} = (\breve{\underline{C}}.\underline{Z}.\underline{C}).\underline{e}_T$
Repeat Step 11 for all cutsections	12	Repeat Step 11 for all cutsections
$\underline{V} = \underline{V}^{(1)} + \underline{V}^{(2)}$	13	$\underline{J} = \underline{J}^{(1)} + \underline{J}^{(2)}$

and stored.

BLOCK 11 The component of pressure $\underline{V}^{(2)}$ due to the interconnection of the equivalent radial sub networks and the link branches is found using the procedure FINDPRESS. This corresponds to steps 7 to 11 of the algorithm. The pressure components $\underline{V}^{(2)}$ and $\underline{V}^{(1)}$ are then summed in step 13 to give the overall solution of node to datum potentials for any iteration.

BLOCK 12 To determine if the solution in terms of the node to datum potentials has converged to within the limit stated by the convergence criterion input in BLOCK 12, a percentage test, as outlined previously in applied.

If the system has converged, control is sent immediately back to BLOCK 6 via label L7: with the flag DECIDE set to a value of 2, which will permit CALCULATE to print out the converged solution using the procedure RESULTS. Alternatively the new flows and friction factors in the cutpipes are computed using CUTFLOW and control passes back to BLOCK 6 via label L7: with DECIDE set to a value of 1, which allows CALCULATE to perform another full iteration.

BLOCK 13 Immediately preceding BLOCK 10 is a test statement which transfers control immediately to CUTRESULTS after CALCULATE has printed a converged solution. CUTRESULTS outputs the flows and pressure drops for every cut pipe.

4.8 <u>MESHDIAK</u>

This program, a listing of which is given in appendix (12.8), was developed using the algorithm of figure (3.10.3) for the solution of network equations in terms of high level interconnection of mesh equivalent sub networks as described in chapter (3.10.3).

The program is written as a series of self contained procedures and the basic flowsheet is shown in figure (4.8.1). The disc backing store is used extensively and the three ICL procedures outlined in chapter (4.7) are used to transfer information from the main core store to the discs and vice versa.

The computation algorithm shown in figure (4.7.2) is slightly different from the algorithm of figure (3.10.2) since each cutsection is treated in isolation by the program. The basis of the program is that it finds for each cutsection the mesh equivalent component of flow due to the external sources for each cutsection and stores this in the form of the mesh solution equation. The individual cutsections are interconnected and the second component of flow due to the interconnection of the equivalent mesh equations by the cut branches is found. The two components of flow are then summed to give the individual mesh flows for the connected network. An iteration process is maintained until the mesh flows have converged to a satisfactory degree. The degree of convergence being decided by the user.

BLOCK 1 Inputs the density and viscosity of a fluid and the value of π

BLOCK 2 Inputs the overall network data: number of cutsections, cutbranches, and total number of nodes and meshes.

BLOCK 3 The procedures PUTARRAY, GETARRAY and WORKSTORE are set up and the latter used to assign four files on the disc storage.

BLOCK 4 Sets up the global arrays, then inputs the cutbranch hydraulic data of diameter, length, pump terms and relative roughness. The initial estimates of flow and friction factor are then assigned to the cutbranches. Inputs the matrix $\underline{\alpha}^{**}_{LT}$ which indicates the incidence of the cutbranches in the original meshes of the network. The data preparation of the matrix together with the data preparation for all of the programs is discussed in the next chapter.

BLOCK 5 Inputs the number of branches and meshes in each of the cutsections.

BLOCK 6 The various counters which control the use of the discs and those which are used as flags are assigned their initial values. The high-level interconnection matrix ZHICON is set to zero. The label L7: is associated with the start of this BLOCK and labels the first statement in the main iteration loop.

BLOCK 7 This is essentially the start of the loop which calculates for every cutsection the equivalent mesh solution equation. For an individual cutsection the number of branches, meshes and hence nodes obtained from the input of BLOCK 5 are used to assign core storage space to the cutsection arrays.

BLOCK 8 This is the part of the program which contains the computational procedures to find the equivalent mesh solution equations. The main procedure is CALCULATE which consists of BLOCKS 8.1 to 8.11

BLOCK 8.1 to 8.8

On the first iteration for any cutsection the vector of node to datum flows is input. The hydraulic data is input in the same order as for the cutbranches. The branch-node connection list is input and the estimates of the individual branch flows and friction factors assigned. In order to evaluate the $\underline{C}_T$ matrix and to assign the primitive tree branch current sources the $\underline{B}_T$ matrix is evaluated using the technique described in appendix (1) by the procedure SEARCHBT. The procedure MAKEC is then used to evaluate $\underline{C}_T$. The primitive current sources are then evaluated. The information necessary for succeeding iterations is then stored on the disc. Notice that $\underline{B}_T$ does not have to be used on subsequent iterations and can be neglected.

On iterations other than the first the necessary information about a cutsection is transferred from disc storage to the core store as in BLOCK 8.8.

BLOCK 8.9 On any iteration, other than the first, the corrected value of flow for every branch of a cutsection is evaluated using MESHFLOW. From a knowledge of the individual mesh flows the tree branch flow due to the mesh flows is evaluated and summed together with the current source to give the corrected branch flow. The link branch flows of the sub network are identical to the mesh flow of the mesh which the links define. A corrected value of the friction factor is also evaluated using FINDPHI. This procedure contains a further procedure called RESULTS which is called if the system has converged on the preceding iteration. RESULTS prints the pipe reference number, the pipe flow, pressure drop and friction factor control is then passed directly to BLOCK 9, not necessitating further computational effort in procedure CALCULATE.

BLOCK 8.10 The procedure FORMCZC evaluates the impedance vector and stores the result. The branches in the cutsections which act as short circuits are then given zero impedances. The mesh solution matrix $(\check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L)$, step 1 of figure (4.7.2), is then formed and inverted. In the program MESHFLAN a special algorithm was developed to obviate the double matrix multiplication, this is once again used. Notice too that only $\underline{C}_T$ of the branch-mesh incidence matrix is used. Step 2 and 3 of the algorithm of figure (4.7.2) are then computed. The resulting mesh flows of the cutsection are part of a larger vector $\underline{J}^{(1)}$ which contains the component of flow due to the external sources. These mesh flows are then transferred to the global matrix ready for use at the interconnection stage. The whole of the operation of BLOCK 8.10 is very similar to the procedure EVALUATOR of the program MESHFLAN.

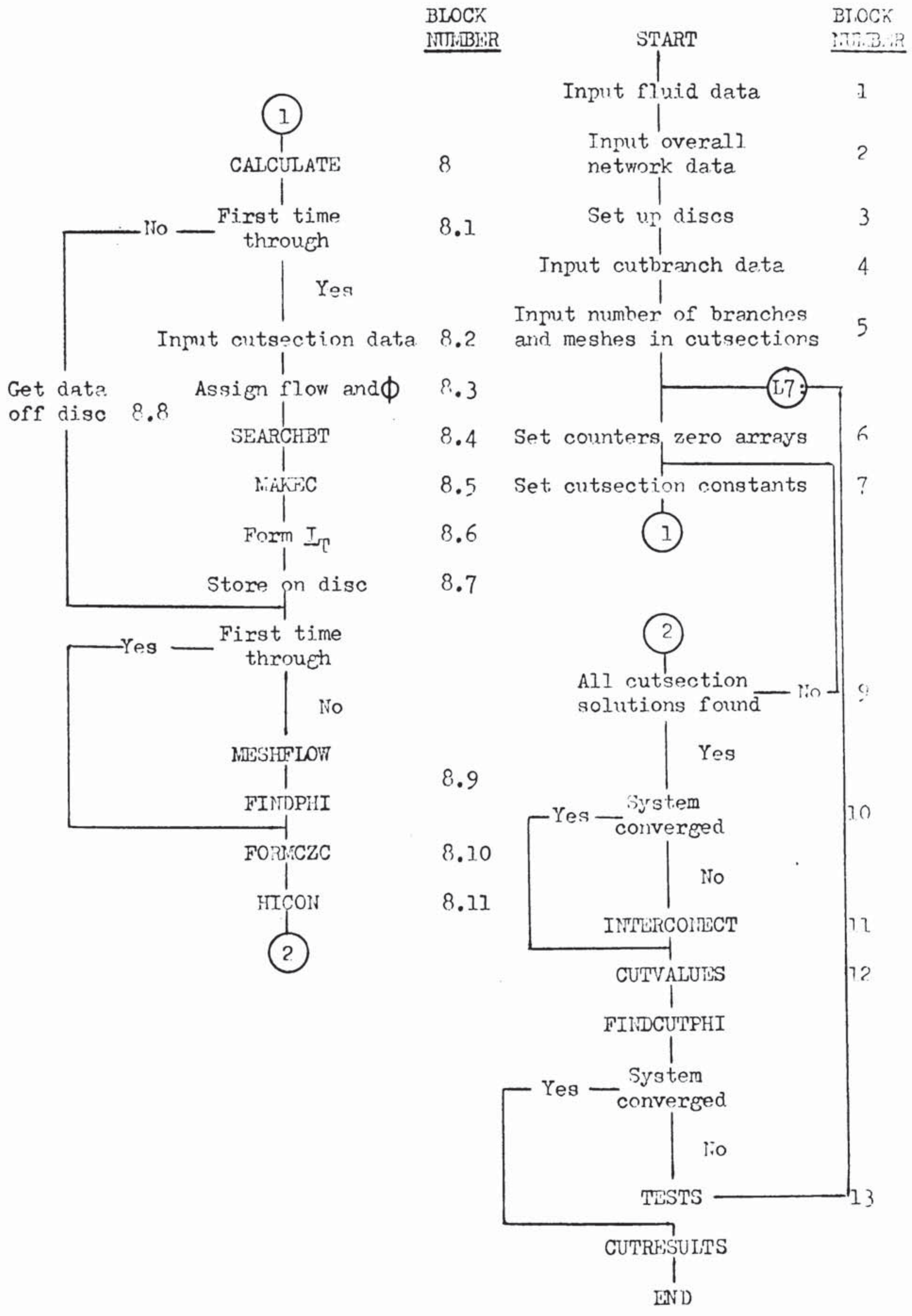
BLOCK 8.11 Step 4 of the algorithm, which indicates the formation of the interconnection matrix, is achieved in a sequential manner. For every cutsection the mesh solution matrix is post- and premultiplied by the partitioned part of $\underline{\alpha}^{**}_{LT}$ which relates the cutbranches to that cutsection. (See worked example in appendix (9)). The interconnection matrix therefore has the value of its elements updated by every cutsection. When all the cutsections have been considered the interconnection solution matrix will be complete save for the addition of the admittances of the cutbranches along the diagonal elements.

BLOCK 9 Essentially the end of a 'FOR' loop, this determines if all the cutsections have been considered. If so, control passes to BLOCK 10, otherwise control is passed back to BLOCK 7.

BLOCK 10 This checks if a converged solution was achieved on the last iteration. If so, BLOCK 8.9 will have output the results for every cutsection and only the results for the cutpipes have to be evaluated, no further calculation being necessary. Procedure INTERCONECT is therefore by-passed. Alternatively iteration is continued by using INTERCONECT to complete steps 7 to 13 of the algorithm.

The admittances of the cutpipes are first evaluated. The completed high-level interconnection matrix is formed as in step 6 and inverted. The component of flow due to the external sources is then transformed by $\breve{\underline{\alpha}}_{LT}^{**}$ to a cutbranch variable by step 7. In the algorithm of NODEDIAK step 8 is summed the effect of external sources $\underline{V}_L^{(1)}$ due to cutsections and the pump sources, ie external sources, in the link branch branches. There is no equivalent step in MESHDIAK since the cutbranches are assumed to be the link branches of the original network and therefore contain no external flows. $\underline{I}^*$ is therefore the negative of $\underline{J}_L^{(1)}$ only. Step 9 evaluates a cut branch pressure component which is transformed by $\underline{\alpha}_{LT}^{**}$ in step 10 to give for every mesh of the cutsections a pressure component. The individual mesh solution matrices stored in BLOCK 8.10 are brought down from the disc and the component of flow due to interconnection is found in step 11. This step is repeated for every cutsection. The two mesh

Figure (4.8.1) The Flowsheet for MESHDIAK



components of flow are summed in step 13 to give the overall component of mesh flow in the interconnected network.

BLOCK 12 Using the individual mesh flows the cutpipe flows and pressure drops are found in CUTVALUES. The procedure FINDPHI then evaluates a new friction factor for each cutpipe. CUTVALUES then determines if a converged solution was achieved on the last iteration. If so the results for the cutpipes are output using CUTRESULTS and control is immediately passed to the end of the program label L200:. Otherwise control is transferred to BLOCK 13.

BLOCK 13 This consists of the procedure TESTS which by applying a percentage convergence test to the individual mesh flows evaluates whether or not a converged solution has been achieved. If a converged solution is achieved, the flag TRIGGER is set to a value of 1, (which will indicate to BLOCK 9 to print out the converged results), otherwise TRIGGER is set to 0 (which will indicate to BLOCK 9 to compute another iteration), and control is sent back to BLOCK 6 via label L7:.

4.9 Data Preparation

As an aid to the understanding of the programs discussed previously, the data preparation, its format and a recommended manner of presentation for card input is demonstrated. The twelve branch network which is used extensively throughout this thesis is the subject of the data preparation.

The Hardy Cross Method

Both HCMESHIN and HCMESHOUT require identical data preparations. It has previously been the subject of discussion that data preparations for this method are tedious. Each branch is assigned a reference number and given a direction, which is the direction of assumed flow. The meshes of the network are then chosen by eye and referenced, each particular mesh being assigned a positive direction which is the direction of its defining link. A list is then made up of, in the first column, the number of branches in a particular mesh and in succeeding columns the reference numbers of those branches with their direction relative to the direction of the mesh in question. This is repeated for all the meshes. This method of inputting the mesh information has a major computational drawback. No advantage may be taken of the partitioning of the branch-mesh incidence matrix since the referenced link branches will appear randomly in the rows of the matrix set up. As discussed in chapter (3.3.3), this can be obviated by referencing the link branches with the highest numbers.

The lengths, diameters and relative roughness are all tabulated in ascending order of reference number. The flows which have been assigned to each branch to satisfy Kirchoff's First Law are also input in ascending order of reference together with their direction relative to that of the assumed direction of the branch.

A typical data preparation would then be:

4	Number of meshes
12	Number of branches
4	Maximum number of branches in any one mesh
0.001	Convergence criterion
2	Exponent n of equation (2.2.2)
62.4	Density of fluid
2.42	Viscosity of fluid
4 9 -7 -3 1	
4 10 7 -4 -5	Number of branches in mesh, and branches which make up the mesh
4 11 -2 -3 8	
4 12 -8 6 -4	

10 -20 5 . . . etc	Flows assigned to each branch
100 100 100 . . . etc	Length of pipes
0.5 0.5 0.5 . . . . etc	Diameter of pipes
0 0 0 . . . etc	Relative roughness of pipes

<u>The Orthogonal Methods</u>

The data preparation for NODEFLAN, MESHFLAN and LATTCLAN is identical, and that for NPSMESHFLAN very similar. Each branch and node is assigned a reference number from 1 onwards. It appears that no importance is attached to the referencing of tree or link branches for NODEFLAN, but as discussed previously referencing of the link branches with the highest numbers results in the unit matrix $\underline{C}_L$ which can be used to advantage in the other programs.

No mesh information has to be prepared since this is automatically generated by the programs which require the information.

A typical data preparation for the first three programs would be:

12	Number of branches
8	Number of non datum nodes
100 100 100 . . . etc	Length of pipes
0.5 0.5 0.5 . . . etc	Diameter of pipes
0 0 0 . . . etc	Pressure sources in pipes
0 0 0 . . . etc	Relative roughness of pipes
-1 2	Branch-node connection list
2 -3	
-2 5	
. . . etc	
-10 0 20 . . . etc	Node to Datum demand-supply terms
62.4	The density of the fluid
2.42	The viscosity of the fluid
3.141	The value of the constant π
0.001	Convergence criterion

NODEFLAN requires one supplementary piece of data, the datum pressure which is input as the last item.

A complete data preparation for NPSMESHFLAN is achieved by supplementing the list shown above. A list of the pressures at the pressure specified node with correct referencing is prepared and input immediately after the first two pieces of data.

The computer output is similar for the four programs; an example, which is taken from NODEFLAN, is shown below.

Pipe No	From Node	To Node	Flow cu/ft	Pressure lb/sq ft	F/Factor
1	1	2	88.382	-148.58	0.00170
2	2	5	37.531	-31.64	0.00201
3	5	3	31.464	-23.04	0.00208
37	15	13	35.138	28.10	0.00204
38	13	21	66.932	-89.81	0.00179

The Diakoptic Method

The data preparations for the two diakoptic methods are very similar. However, the program NODEDIAK is much more developed than MESHDIAK and therefore the data preparation is better organised. It is wiser therefore to consider the two separately.

NODEDIAK

Any number of cutbranches up to M where M is the number of link branches in the connected network may be removed. It is quite obvious, however, that the removal of only those branches which are required to isolate the cutsections, provided each of the cutsections is connected by at least one branch to the datum node, is the most computationally efficient. Each branch and node of the cutsections is assigned its own local reference number, each cutsection also being assigned a reference number. Every branch and node of the connected network having been assigned a reference number, the branches and nodes of the cutsections have two reference numbers. The global reference numbers being used to prepare the cutbranch-node connection list. The data is then prepared as follows; the example used is that of appendix (8), assuming all the pipes are 100 ft by 0.5 ft in diameter, smooth and there are no pumps in the system.

62.4	Density of fluid
2.42	Viscosity of fluid
3.141	Value of the constant π
32.2	Value of the constant g_c
0.001	Convergence criterion
2	Number of cutsections
2	Number of cutbranches
8	Total number of non-datum nodes
0.5 0.5	Cutpipe diameters
100 100	Cutpipe lengths
0 0	Cutpipe pump terms
0 0	Cutpipe roughness
-4 6	Cutpipe connection list
-5 7	
7 5	The number of branches and non-datum
3 3	nodes in each cutsection
10 0 20 0 0	Node to datum flows for cutsection 1
0.5 0.5 0.5 0.5 0.5 0.5 0.5	Diameters, lengths, pump terms and
100 100 100 100 100 100 100	roughnesses for pipes of cutsection
0 0 0 0 0 0 0	1
0 0 0 0 0 0 0	
1 -2 2 -3 -3 6 5 -6	Connection list for cutsection 1
4 -5 -1 4 -2 5	

```
-15  0    -15                          Node to datum flows for cutsection 2
0.5  0.5  0.5                          Diameters, lengths, pump terms and
100  100  100                          relative roughnesses for cutsection
0    0    0                            2
0    0    0
3  -4   1  -2   2  -3                  Connection list for cutsection 2
```

This small network has not in fact been analysed by the program NODEDIAK but some of the output for a larger network which is discussed later is shown below.

```
Results for Section No 1

Pipe No         Node to Node         Friction Factor         Flow cu ft/min
   1              1     2               0.00307                  104.64
   2              2     3               0.00264                  151.68
   etc

Node No         Pressure lb/sq ft
   1                 -54.80
   2                 -48.59

Results for Section No 2

Pipe No         Node to Node         Friction Factor         Flow cu ft/min
   1              1     3               0.00247                  264.99
   2              2     3               0.00297                  120.10
   etc

Results for Cutpipes

Cutpipe No        Flow cu ft/min
    1                130.80
    2                 40.28
```

MESHDIAK

Not being as well developed as NODEDIAK, certain restrictions must at present be applied in the data preparation for MESHDIAK. One of these is that the cut branches must be numbered with the highest local reference numbers in the cutsections. This is a computational requirement, not a failing of the theory of the method. For instance, the data preparation shown below is for the network of appendix (9), but in that worked example the cutbranches are not referenced with the highest numbers and the reader will have to mentally adjust the branch reference numbers accordingly. This is easily achieved since the node reference numbers are unchanged for the data preparation shown below. Also, only branches which will appear in two or more cutsections may be removed from a network for MESHDIAK. Any further limitations in the data preparation will be explained as the restriction is encountered. Assume that all pipes are 100 ft in length, 0.5 ft in diameter, smooth and have no pump terms associated with them.

62.4	Density of fluid
2.42	Viscosity of fluid
3.141	The value of the constant π
0.001	Convergence criterion
2	Number of cutsections
2	Number of cutbranches
8	Total number of non-datum nodes
4	Total number of meshes
0.5 0.5	Diameter of cutpipes
100 100	Length of cutpipes
0 0	Pump terms in cutpipes
0 0	Relative roughness of cutpipes

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

This is the cutpipe-mesh incidence matrix. It shows the way in which the cutpipes were included in the untorn meshes. This is the full $\underline{\alpha}_{LT}^{**}$ matrix for the problem. In NODEDIAK $\underline{\gamma}_{TL}^{**}$ was formed by the program using only a cutpipe-node connection list. With the restriction that the short circuits in the torn meshes are referenced with the highest numbers it is a trivial task to remove this requirement of data preparation.

```
7   2
7   2
```

The number of branches (including short circuits) and meshes in each cutsection

```
10   0   20   0   0
```

The node to datum flows assigned to the nodes of cutsection 1. Since the method of cutting proposed bisects nodes, the question of which cutsection to assign the node to datum flow of the bisected nodes arises. The flow can be assigned to either, or proportionatly to both.

```
0.5  0.5  0.5  0.5  0.5  0.5  0.5
100  100  100  100  100  100  100
0    0    0    0    0    0    0
0    0    0    0    0    0    0
```

The diameters, lengths, pump terms and relative roughnesses of the pipes of cutsection 1

```
1  -2   2  -3   -2  5   -1  4
-3  6   4  -5   5  -6
```

Connection list of cutsection 1

```
-15  0    -15  0    0                   Node to datum flows for cutsection 2
0.5  0.5  0.5  0.5  0.5  0.5  0.5       Diameters, lengths, pump terms and
100  100  100  100  100  100  100       relative roughnesses of the pipes of
0    0    0    0    0    0    0         cutsection 2
0    0    0    0    0    0    0
1  -2   2  -3   2  -5   1  -4           Connection list of cutsection 2
3  -1   4  -5   5  -6
```

4.10 Matrix Inversion

All of the matrix methods described in this chapter necessitate the inversion of at least one solution matrix. Irrespective of the method in question, the solution matrix will have the following properties.

It will be sparse, symmetric and have its largest elements on the diagonal. Inversion routines which take advantage of these properties have been investigated.

Four procedures, a self contained ICL package, a Guass-Jordan elimination routine, a Choleski routine and a modified Choleski routine due to Caffrey (33) were studied. The results in terms of the time taken for inversion are discussed in Chapter (5).

It was quite obvious at the start of this research project that the inversion of the solution matrices was likely to be one of the main obstacles in the way of producing efficient programs. The existing methods of inversion were therefore investigated. However, it became equally obvious that this investigation could develop into a major task, when the original aim of this research was to produce solutions to network problems. The author admits that there are probably more efficient inversion routines than the one which was finally selected, for instance the techniques of Sato and Tinney (22) could be considered. However the existence of more efficient inversion routines does not detract from the value of the solution methods proposed since these improved routines can only increase the overall efficiencies of the developed programs. The production of more efficient programs and the question of inversion is further discussed in Chapter (6).

The Choleski method was used by Middleton (6), and he found the method to be efficient, ie sufficiently efficient to make the diakoptics program converge to a solution more quickly than the Hardy Cross program due to Daniel (3). The Caffrey routine is more efficient than the Choleski method. It was constructed to invert a positive definite symmetric matrix by a simplified variant of the square root method. However it was noted by Randell and Broyden (34) that the method would invert non-positive symmetric matrices provided that the leading minors of the matrix were non-zero. The method was therefore suitable for the inversion of network solution matrices. It has the advantage that fewer temporary storage locations are required, no identity matrix is used, no square roots are computed, only n (where n is the order of the matrix) divisions are performed, and as n becomes large the number of multiplications approaches $n^3/2$.

The routine which has been constructed is called MYTRIX and can be found implemented in all of the programs developed from the matrix methods, with the exception of LATTNODEFLAN, in appendices (12.3) to (12.8).

CHAPTER 5

RESULTS

5.1 Introduction

Six different network problems have been analysed using the programs described in Chapter (4). The examples have been chosen because it is thought that they represent the variety of realistic cases. For instance, one network consists of a large number of branches and is heavily meshed, whilst another has a smaller number of branches and is not heavily meshed. The smallest network consists of 28 branches and the largest, of 112 branches. One of the major difficulties of selecting representative network problems is the lack of available networks for analysis. The author is therefore indebted to W S Atkins and Partners for permission to reproduce in a modified form the two networks of figures (5.2.8) and (5.4.8).

The following chapters describe the results obtained in the solution of six network problems by a variety of methods.

5.2 The Hardy Cross Methods

In order to verify the results obtained by Middleton and Gay (15), who used a program based on the Hardy Cross mesh method, similar to HCMESHIN, and to compare this program with HCMESHOUT, the test network developed by Middleton was reconstructed. This water distribution network called TEST1 contains 21 non-datum nodes, 38 branches and therefore 17 meshes. Each branch is 100 ft in length and 0.5 ft in diameter and is smooth. The system contains no pumps. TEST1 with 6 node to datum flows specified in cu ft/min is shown in figure (5.2.1).

Middleton (6) examined three different cases of loop formation;

Figure (5.2.1) TEST1 The Network due to Middleton (6) minimum, arbitrary and maximum overlap. These three cases are reproduced in figures (5.2.2), (5.2.3) and (5.2.4) respectively. In all cases the tree branches are the more heavily drawn and have the lowest reference numbers. The choice of node referencing is the same for each of the cases since this is not important for the Hardy Cross methods. The mesh reference numbers are assumed to be associated with the reference numbers of the link branches as described in chapter (3.3.3). Each branch and mesh is assumed to be directed. The directions of the branches of figures (5.2.2) to (5.2.4) are not indicated on the network since they may be found in table (1) and the node to datum flows in table (2), which together with all tables referred to hereafter will be found in appendix (13).

The results obtained by Middleton and Gay (45) in terms of the number of iterations required for solution for the three cases outlined above are shown in table (3).

The results obtained by the author for the same analysis are shown in tables (4) and (5). In all three tables the difference in the relative number of iterations and the relative computational times is similar for the minimum and arbitrary overlap choices. Too much significance should not be attached to the computation time since it represents the total 'mill' time, which includes time elements other than the mathematical computation times. However there is a marked difference in the actual number of iterations needed to achieve convergence. Unfortunately Middleton and Gay do not indicate the value of the convergence criterion nor the manner in which it is applied. The convergence criterion used throughout this thesis, unless otherwise stated, is that the difference between a corrected and an uncorrected branch flow be less than 0.1 per cent of the corrected flow applied in the manner described in chapter (3.14). For minimum and arbitrary

Figure (5.2.2) TEST1 Minimum Overlap Condition

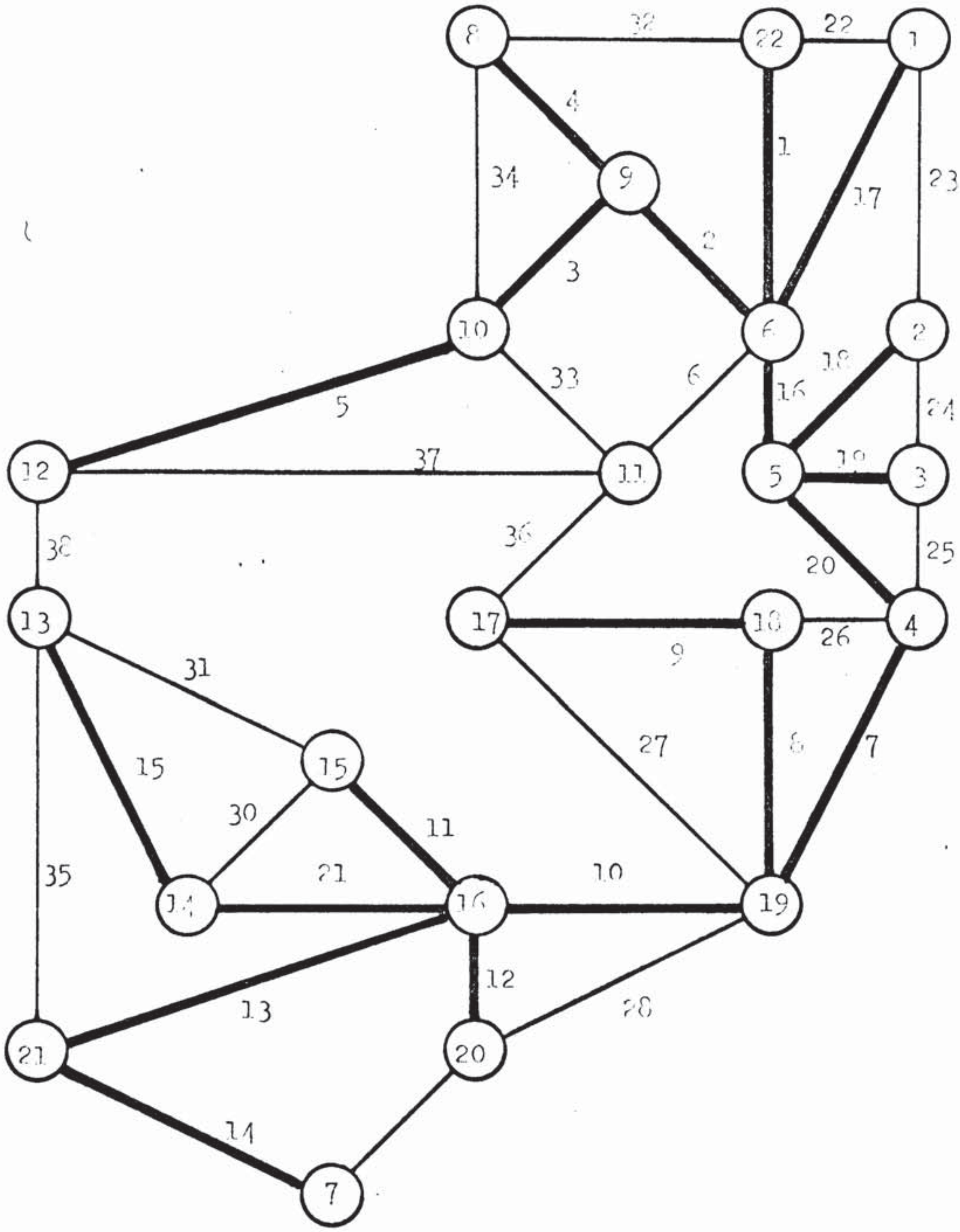


Figure (5.2.3) TEST1 Arbitrary Overlap Condition

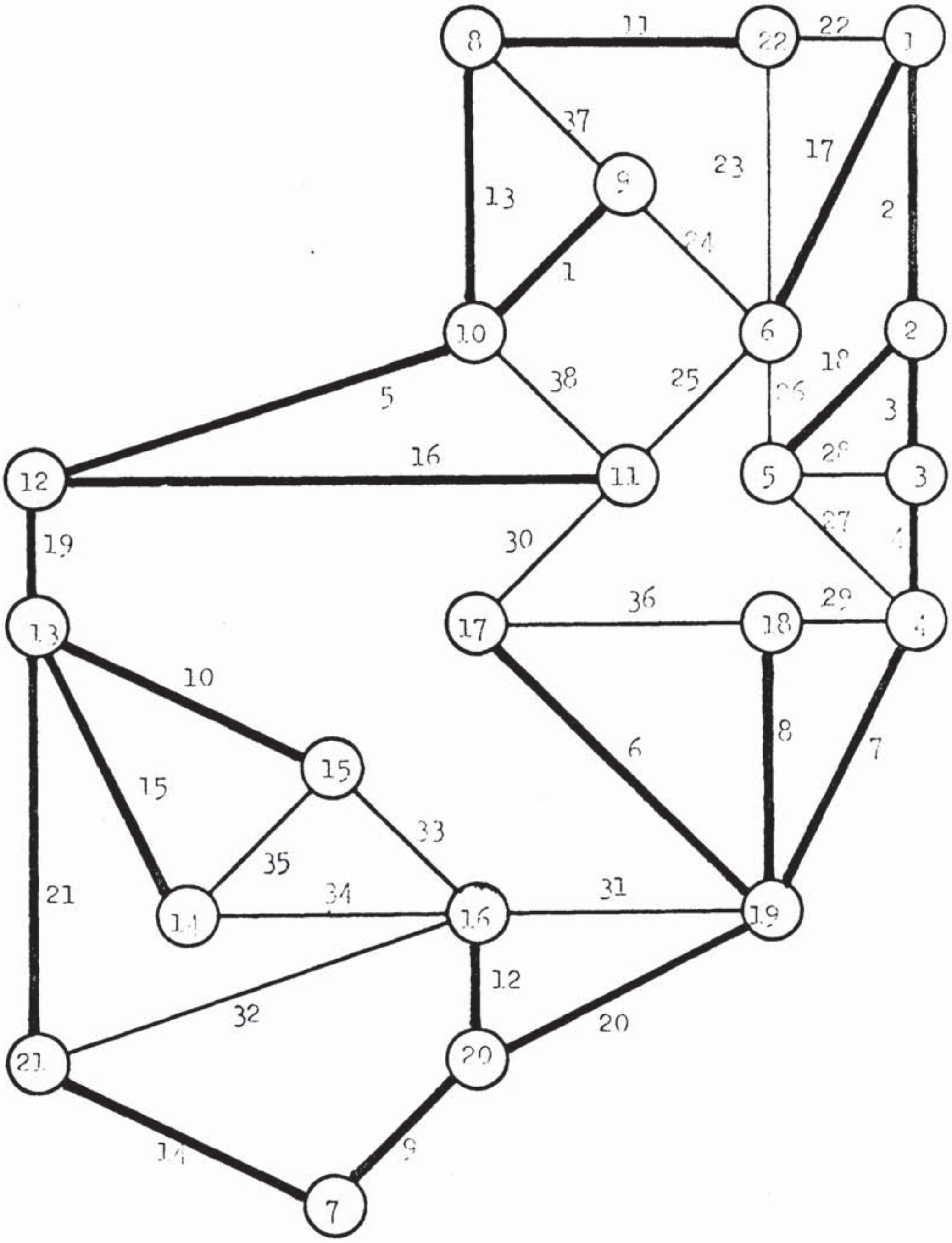
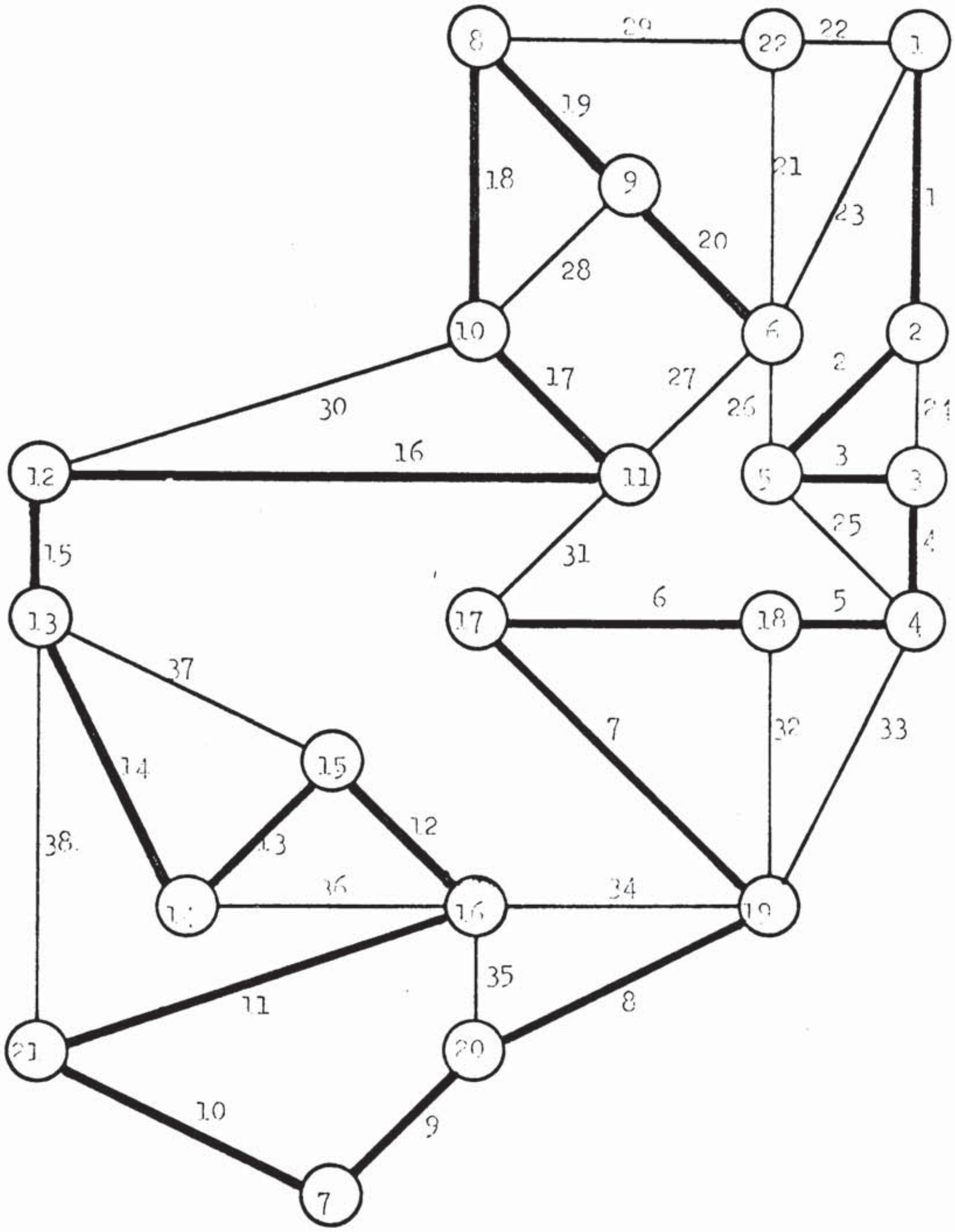


Figure (5.2.4) TEST1 Maximum Overlap Condition



overlap conditions the results from HCMESHOUT agree with those of Middleton and Gay. However the difference in the number of iterations for arbitrary and maximum overlap choices of table (3) does not agree with the relative difference for the same two cases in table (4), nor for the improved program of table (5). These latter two tables in fact indicate that there is little difference between the arbitrary and maximum cases, both in the number of iterations and the speed of convergence. One can only conjecture that the results obtained by Middleton and Gay were not the results for the maximum overlap condition of TEST1 quoted, due to a mistake in the data preparation, or due to the fact that the initial estimate of flow distribution caused the solution to be unstable, though the latter is unlikely.

The results in terms of the individual pipe flows for HCMESHIN and HCMESHOUT for all three cases of overlap are compared with the results of the minimum overlap case of Middleton (6) in tables (6) and (7) respectively. There is good agreement between the flows calculated by HCMESHIN, HCMESHOUT and the results obtained by Middleton (6).

The reduction in the time required for solution of HCMESHOUT in terms of the solution time of HCMESHIN is also well demonstrated. Table (8) shows the percentage reduction in convergence time for the improved program HCMESHOUT in terms of the equivalent solution time for HCMESHIN.

A similar analysis was carried out on the three cases of minimum, arbitrary and maximum overlap for the network of Knight and Allen (35) as shown in figures (5.2.5) to (5.2.7). The results obtained in terms of the individual pipe flows for the maximum condition of overlap give good agreement with the flows calculated by the orthogonal matrix methods as shown in table (12). Knights and Allen analysed their network as a town's gas distribution system, but Middleton reports that they

Figure (5.2.5) The network due to Knights and Allen (35)

Minimum Overlap Condition

Figure (5.2.6) The network due to Knights and Allen (35)
Arbitrary Overlap Condition

Figure (5.2.7) The network due to Knights and Allen (35)

Maximum Overlap Condition

did not define the values of viscosity and density. Their network has been analysed as a water distribution system for the comparison of HCMESHIN and HCMESHOUT.

The connection list for all three cases of overlap is shown in table (9) and the dimensions of the network for minimum overlap are given in table (10). The node to datum supply vector used in all cases for this network is shown in table (11). The convergence times and the number of iterations required for solution for HCMESHIN and HCMESHOUT are shown in tables (13) and (14). Further evidence can be deduced from these two tables that the program HCMESHOUT is more efficient than HCMESHIN as shown in table (15).

The network W S Atkins No 1 and also that due to Dolan were also analysed under three conditions of overlap. The connection lists for the three cases of overlap for the water distribution network W S Atkins No 1 of figures (5.2.8) to (5.2.10) are shown in table (16), and the dimensions of the network corresponding to the case of minimum overlap are shown in table (17). The node to datum flows for this network are shown in table (18). The connection lists for the three cases of overlap for the water network due to Dolan of figures (5.2.11) to (5.2.13) are shown in table (19) and the dimensions of the network corresponding to the case of minimum overlap are shown in table (20). The node to datum flows for this network are shown in table (21). The percentage reduction in solution time for HCMESHOUT for the network W S Atkins No 1 is shown in table (22) and that for the network due to Dolan is shown in table (23).

The relative accuracy of the results of HCMESHIN and HCMESHOUT are discussed at a later stage, but it is necessary to consider the philosophy of employing either of these two methods at the present time. The time dependence of achieving a converged solution on the choice of mesh

Figure (5.2.8) The network W S Atkins No 1

Minimum Overlap Choice

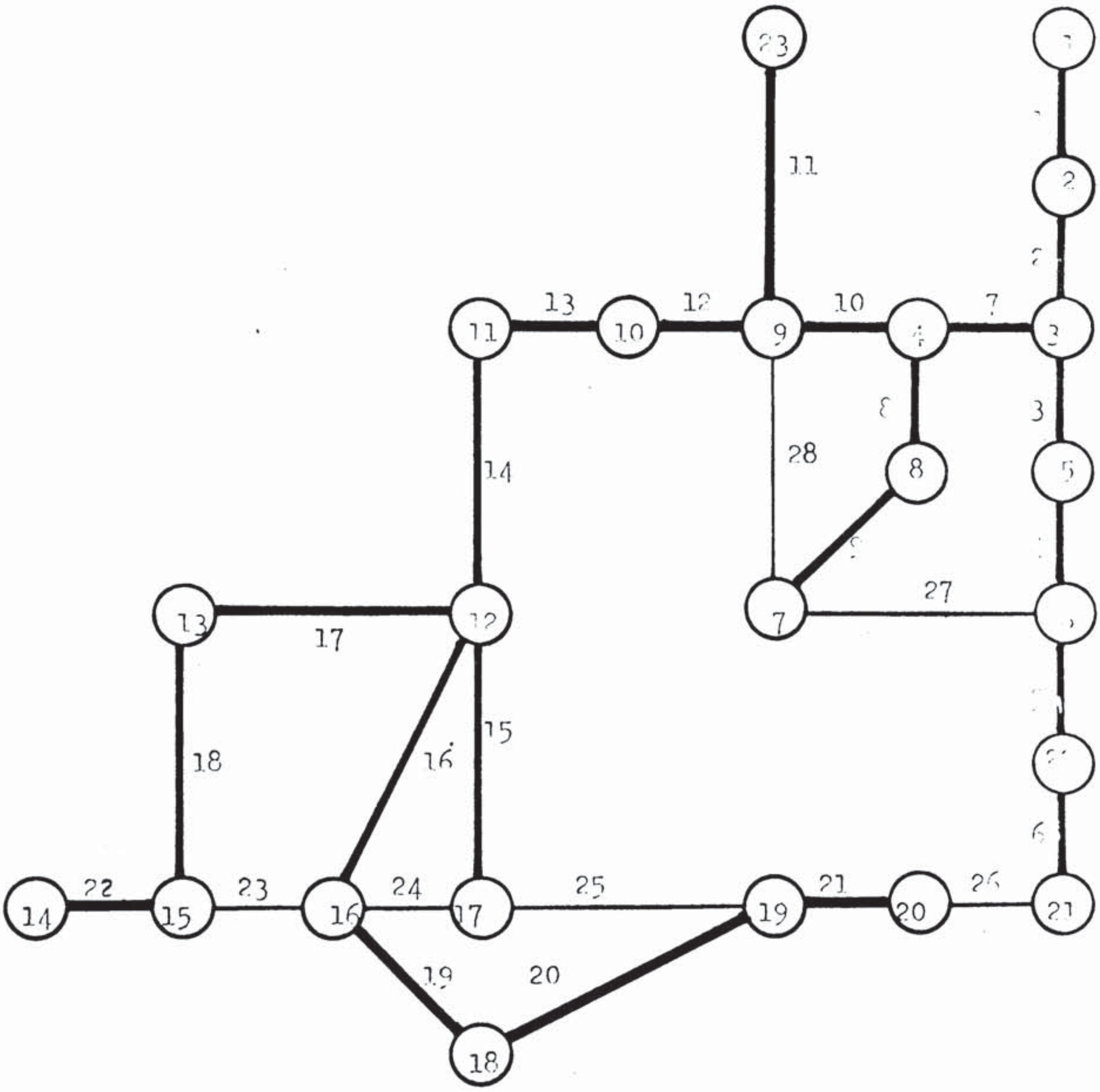


Figure (5.2.9) The Network W S Atkins No 1

Arbitrary Overlap Choice

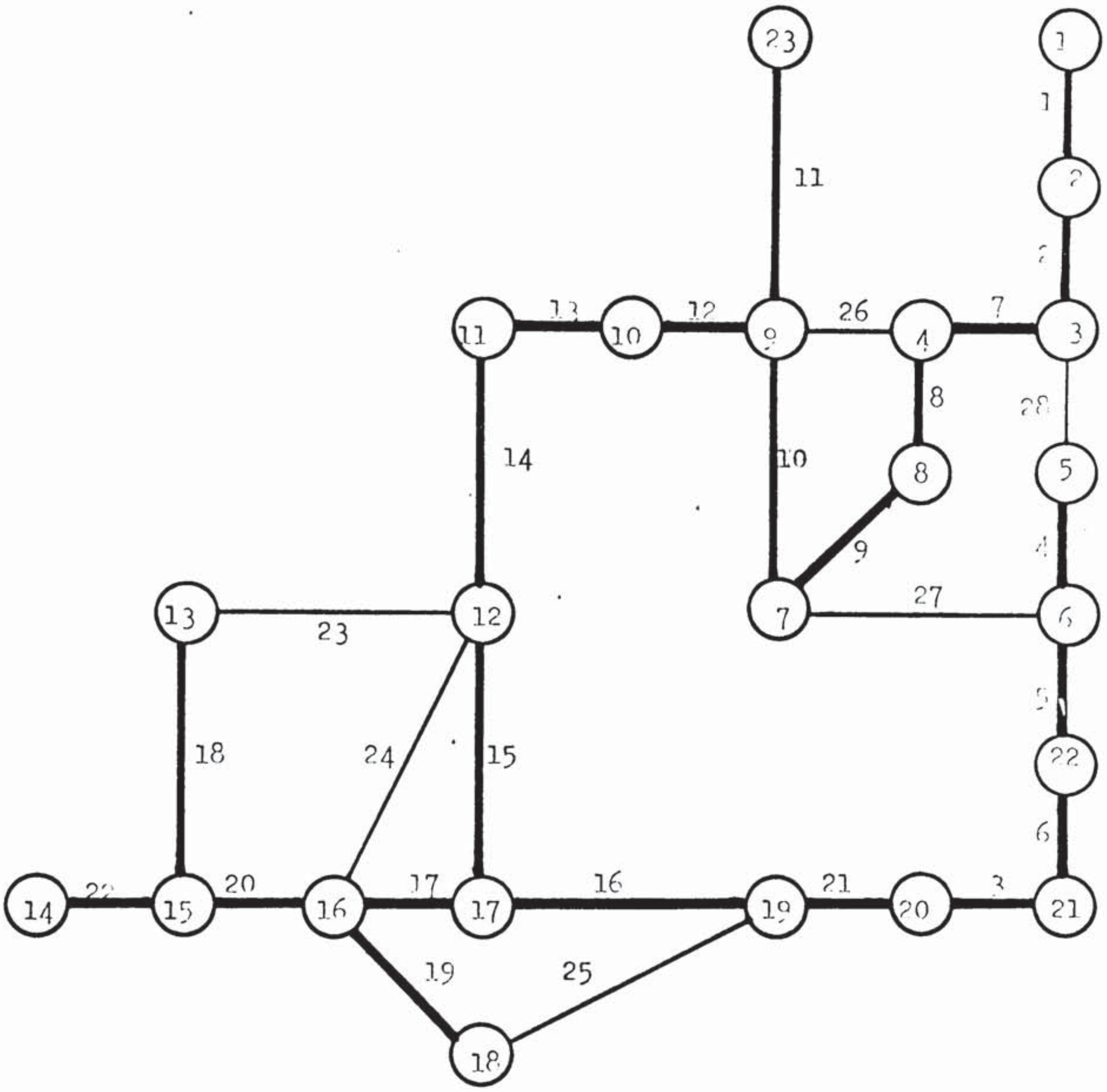


Figure (5.2.10) The Network W S Atkins No 1

Maximum Overlap Choice

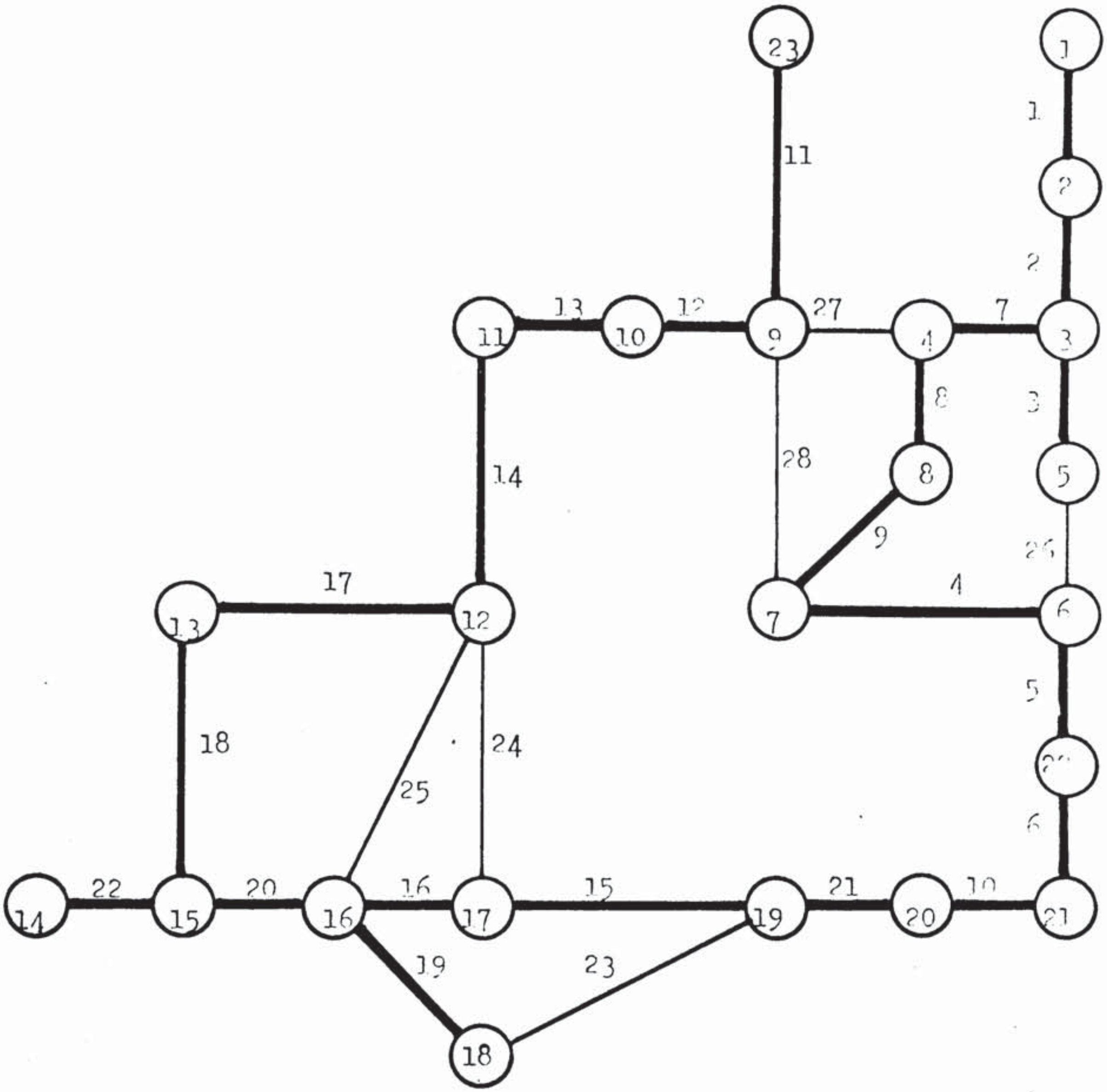


Figure (5.2.11) The Network due to Dolan

Minimum Overlap Case

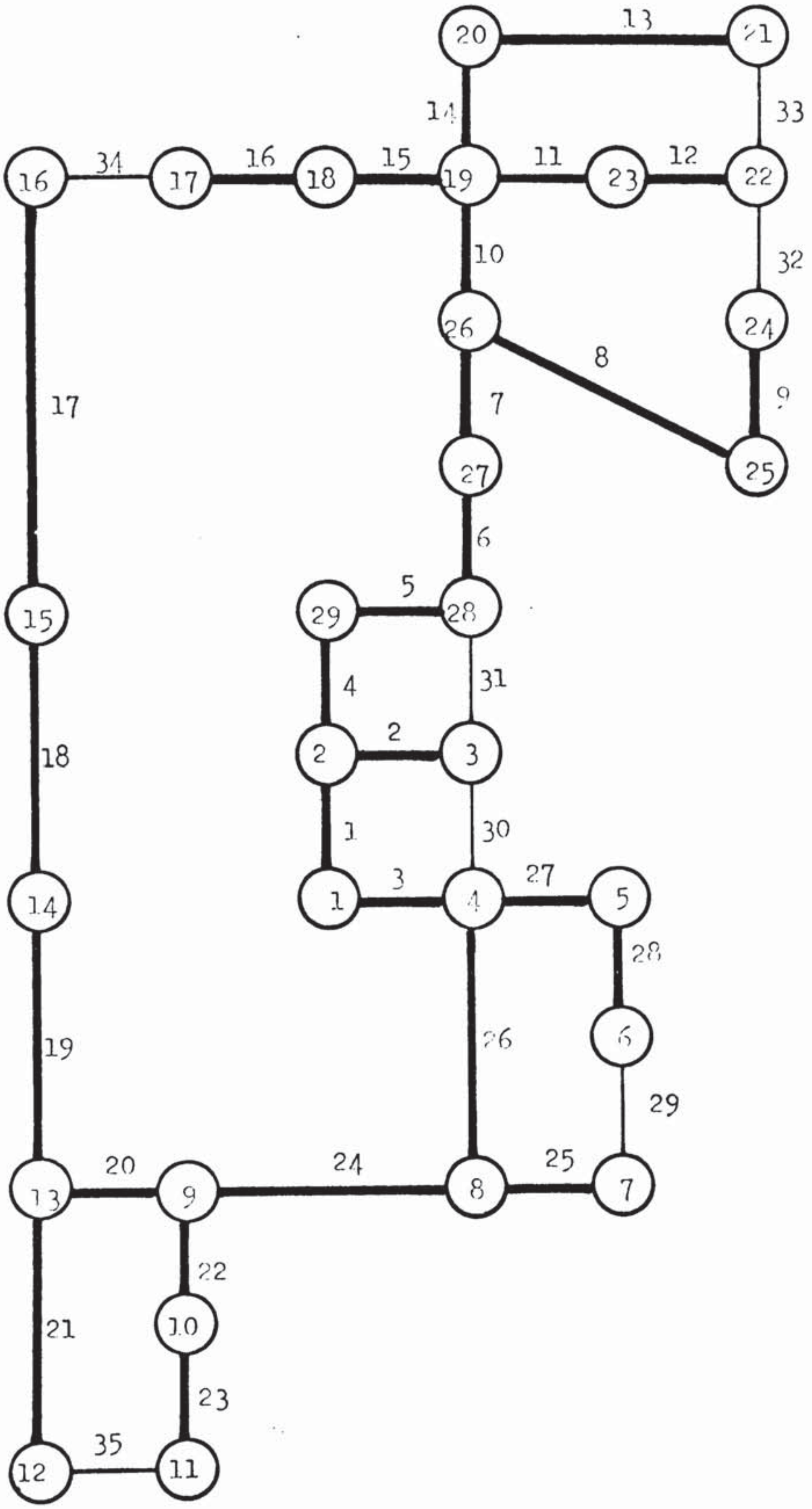


Figure (5.2.12) The Network due to Dolan

Arbitrary Overlap Case

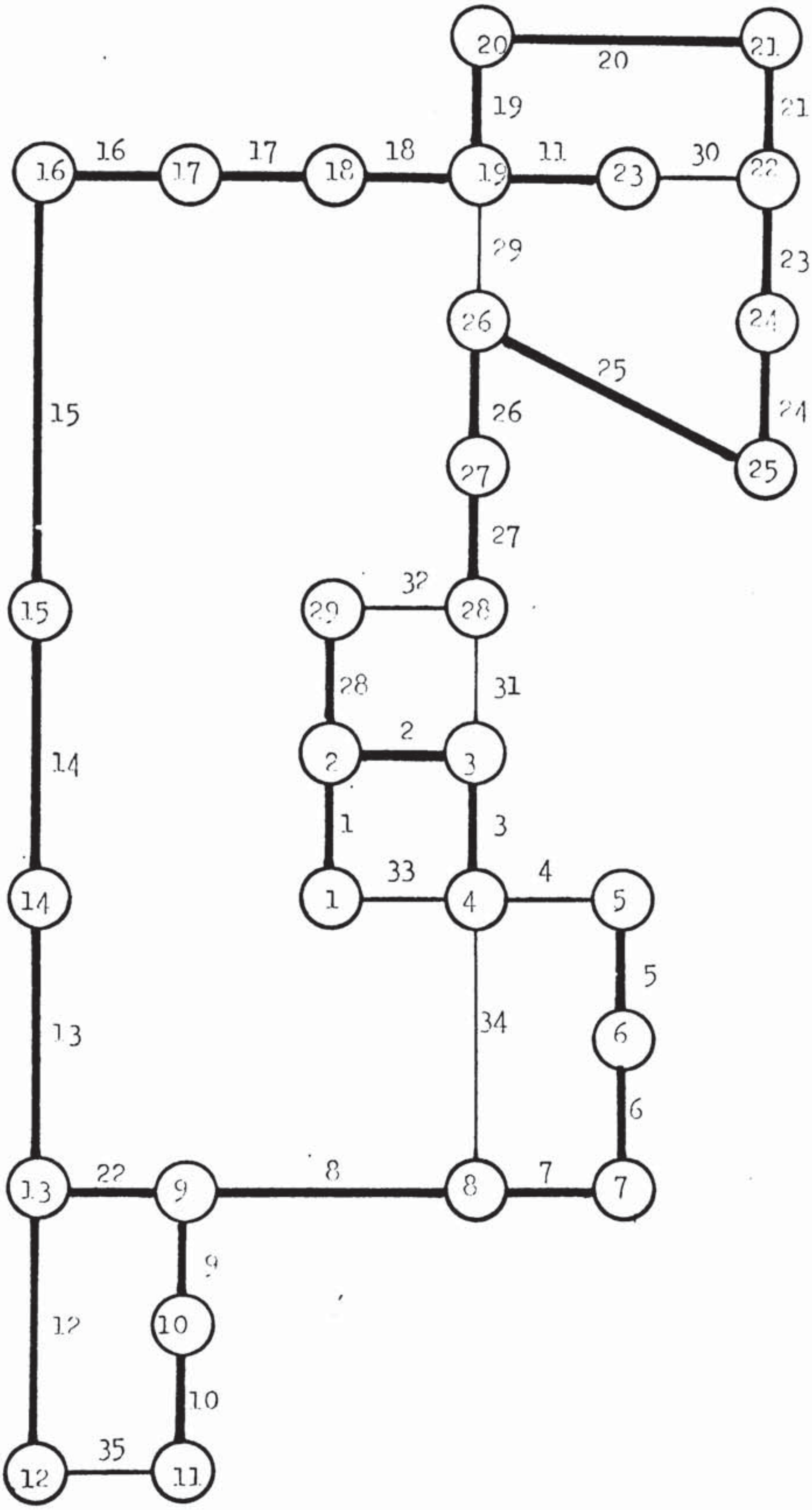
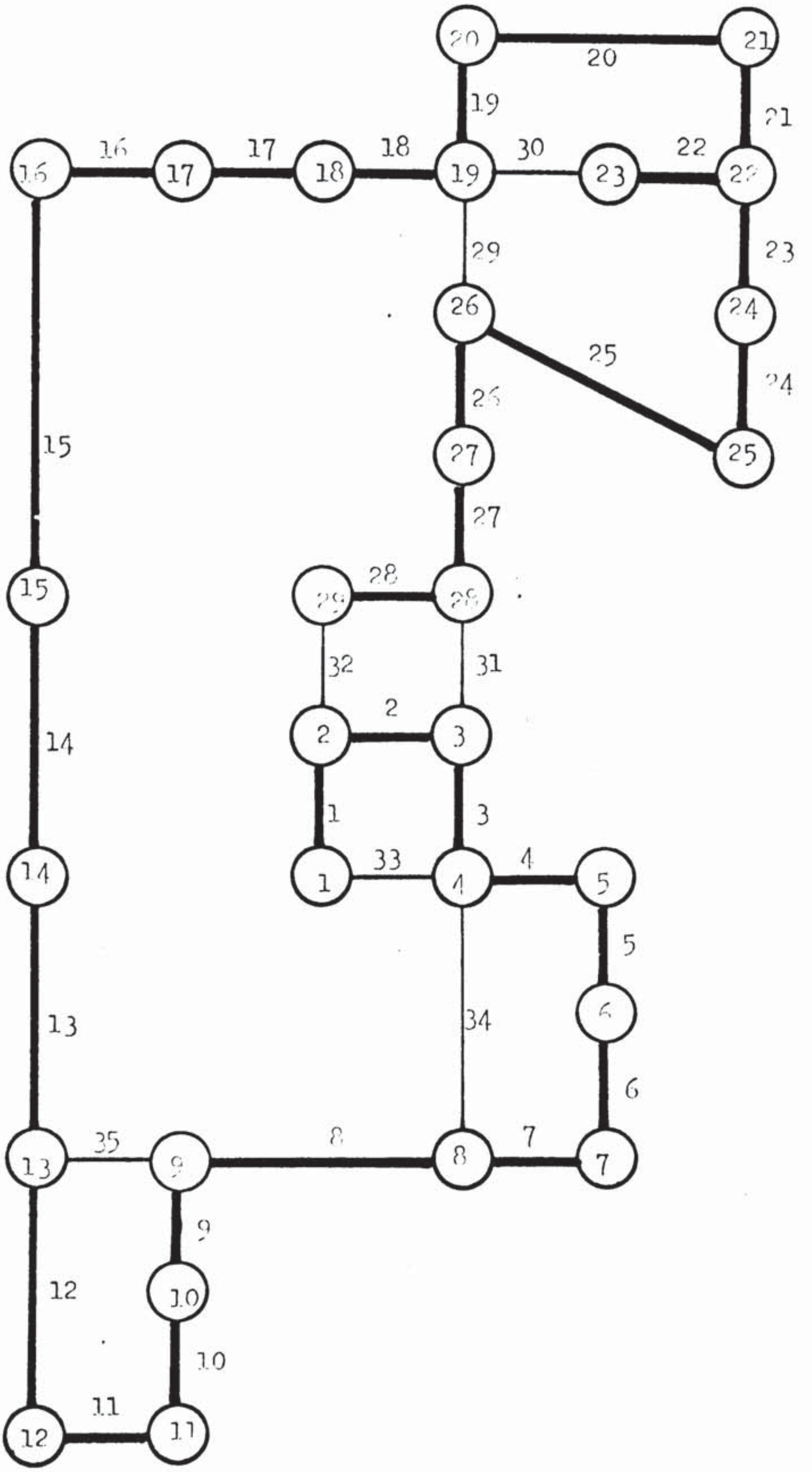


Figure (5.2.13) The Network due to Dolan

Maximum Overlap Choice



formation is well demonstrated. The choices are of course extremes, but it is important to note the relatively small difference in convergence times and number of iterations for the arbitrary choice and maximum overlap choices. It is a reasonable conclusion therefore that the choice of even an arbitrarily selected set of meshes can lead to convergence problems and that only the minimum overlap condition leads to swift convergence.

The maximum overlap condition, the trunk, is however the easiest to set up, but for a large network the setting up of the minimum overlap condition requires great skill and experience. The data preparations for HCMESHIN and HCMESHOUT under any condition of overlap are tedious, especially so with maximum overlap because of the greater number of branches in the meshes. Flows also have to be assigned to each branch together with their assumed direction relative to the direction of the branches. By way of an example the data preparation for the 112 branch network TEST2 required 190 cards, five times as many as for the matrix methods.

The user of the Hardy Cross methods is therefore faced with a dilemma; the easiest choice of meshes leads to tedious data preparations and convergence problems, the most difficult choice of meshes requires expertise or long periods of time to set up.

The results obtained for the two Hardy Cross programs are compared with the results obtained using the matrix methods later in this thesis. To set up the comparison the more efficient program, HCMESHOUT, only will be considered and the condition of arbitrary overlap used to compare convergence times. The reason for this is that the program HCMESHOUT has conclusively been shown to be the more efficient Hardy Cross program and that the arbitrary choice of overlap is representative of mesh formations likely to be set up.

5.3 The Matrix Methods

There are four orthogonal matrix programs and three of these will now be examined; the fourth, NPSMESHFLAN, is constructed similarly to MESHFLAN and its general performance is also similar. However because of its special function it is discussed at the end of this section.

It will be shown that the choice of mesh formation has no effect on the convergence, the choice of initial guess has no effect on convergence and that convergence of a network analysed under a range of node to datum supplies is only minimally affected by the range of flows.

The network TEST1 was solved using NODEFLAN, MESHFLAN and LATTNODEFLAN for the three cases shown in figures (5.2.2) to (5.2.4). The results in terms of the individual pipe flows are given in table (24) for the three analyses of the maximum overlap condition.

The lack of effect on convergence of the choice of meshes is well demonstrated in table (25). The three cases of the network due to Knights and Allen shown in figures (5.2.5) to (5.2.7) were similarly examined and the results in terms of individual branch flows for the maximum overlap condition are shown in table (12). The effect of the choice of overlap on convergence is further indicated in table (26) which shows the number of iterations and the time required for solution for the three overlap cases. Table (27) shows the number of iterations and the time required for the solution of the three overlap cases of the network W S Atkins No 1 and table (28) shows the same information for the three cases of the network due to Dolan.

The most obvious conclusion to be drawn from the results presented so far for the matrix methods is the overwhelming disadvantage in terms of time required for the solution of network problems using the program LATTNODEFLAN. In the analysis of the sixteen network cases presented, there is no evidence to support Branin's claim (19) that the method would be more efficient than the standard orthogonal method. Even in the analysis of a network which is not heavily meshed, for instance that due to Dolan, where only a few link branches have to be added into the tree solution matrix, there is no advantage in using the method. In this particular case the program MESHFLAN is the most efficient of the three programs.

Further discussion of the orthogonal matrix programs therefore excludes any reference to the program LATTNODEFLAN since it has been shown that this method is in no way more successful in the analysis of network problems than even the Hardy Cross methods.

5.3.1 The stability of the matrix methods

To analyse the stability of the matrix methods the program NODEFLAN has been used. Various estimates of the initial flows have been used, various convergence criteria have been used, and a range of node to datum flows has been used. This range of node to datum flows satisfies the criterion established by Daniel (3) which is discussed in chapter (2.4.7).

To analyse each of the matrix methods, including the diakoptics methods, under all of the above conditions would be an onerous task. However, since the general approach underlying all the matrix methods is basically similar, it was thought that the analysis of NODEFLAN and MESHFLAN only would indicate the stability of all of the methods. It soon was clear that both methods were equally stable, and that the results of tests applied to one of the methods would be representative of the results from all methods. NODEFLAN was then selected as the single program on which to base further stability studies. Some of the results obtained in stability studies using MESHFLAN are however presented below to establish that the stability of all of the matrix methods is excellent. To further demonstrate stability a comparison was made with the Hardy Cross program HCMESHOUT.

5.3.2 Convergence of the matrix methods

Middleton (6) examined the effect of the choice of the initial estimate of the individual branch flows on the convergence of his diakoptics program. He noted that only minor changes in the number of iterations required for solution occurred. However the range of initial estimates was not large. Table (29) shows the number of iterations required for the solution of the maximum overlap case of figure (5.2.4) for a wide range of initial estimates of branch flow, using NODEFLAN. This table quite clearly shows that the choice of initial estimates of flows has little effect on the rate of convergence. This is because the node to datum flows will dictate on the first iteration that the calculated branch flows will be of the correct order, irrespective of the value of the branch admittances. For succeeding iterations therefore the branch admittances quickly become better approximations, because of the relatively small changes in friction factor.

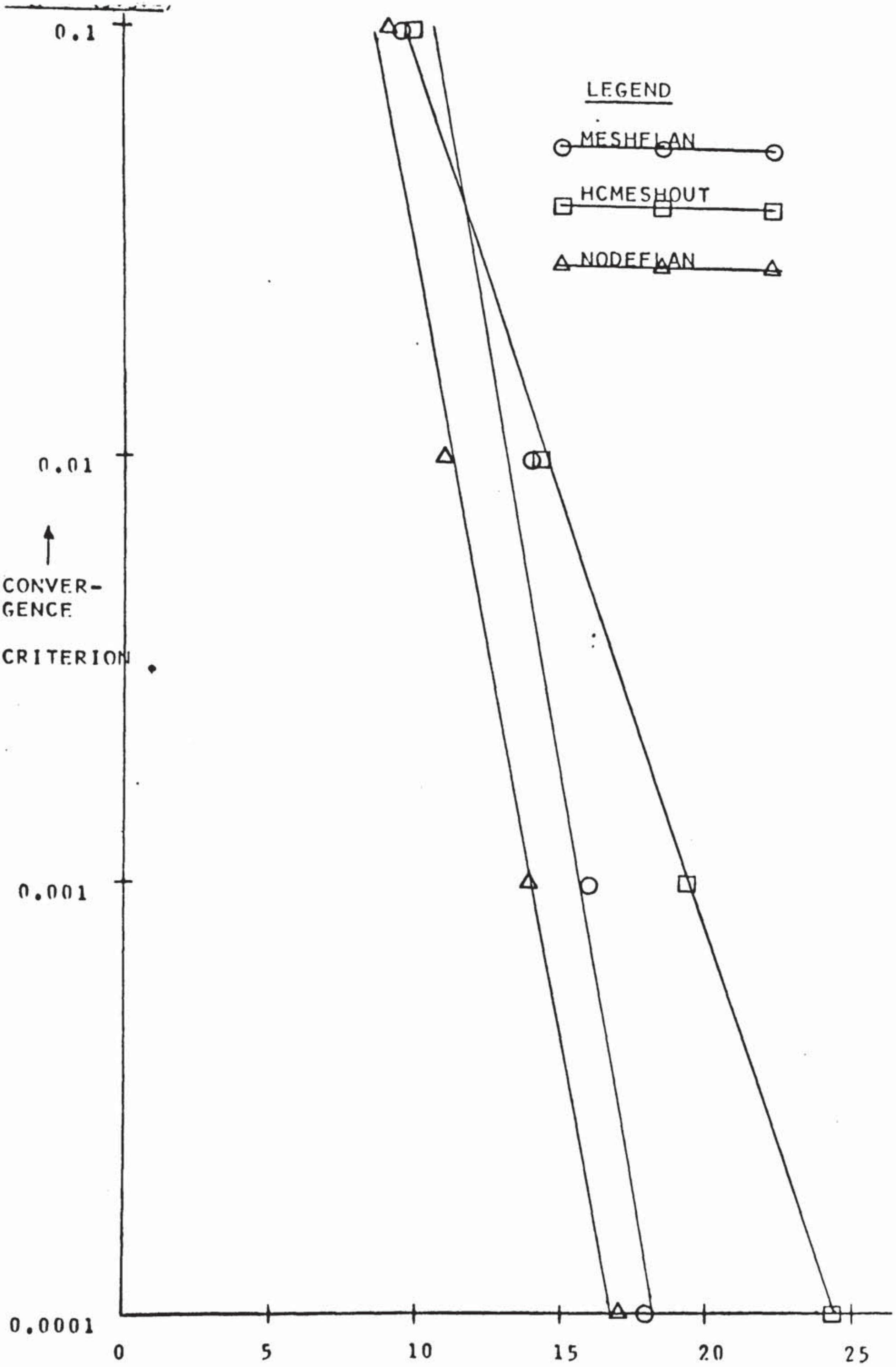
The major conclusion to be drawn from the above discussion is that it is reasonable to allow the program to assign the initial estimates of branch flows internally. With the result that less time is spent by the user preparing and inputting these estimates.

The convergence criterion used to study the effect of initial estimates of flow was uniformly applied as 0.1 per cent of the calculated flow. To further demonstrate the stability of the matrix methods, the maximum overlap case of TEST1 was analysed with the initial estimate of flow of 1 cu ft/min for every branch with a wide range of convergence criterion. The results obtained in terms of the number of iterations required for solution for NODEFLAN, MESHFLAN and HCMESHOUT are shown in table (30). In view of the large difference in the number of iterations required by the matrix methods and the Hardy Cross method, the significance of the results may be obscure. Figure (5.3.1) indicates the significance clearly, the graph shows a log-linear plot of the convergence versus the number of iterations required for solution. NB that the scale of the x axis, the number of iterations, for the Hardy Cross method is one fifth of that for the matrix methods.

Two important conclusions can be drawn from the graph. The slopes of the NODEFLAN and MESHFLAN plots are very similar showing that the methods are equally dependent on convergence. The fact that MESHFLAN constantly requires more iterations than NODEFLAN at any particular convergence is discussed at a later stage. The second and most important conclusion is that as the convergence criterion is decreased, the Hardy Cross method requires relatively more iterations to converge to a solution than either of the two matrix methods, demonstrating further the inadequacy of the Hardy Cross methods.

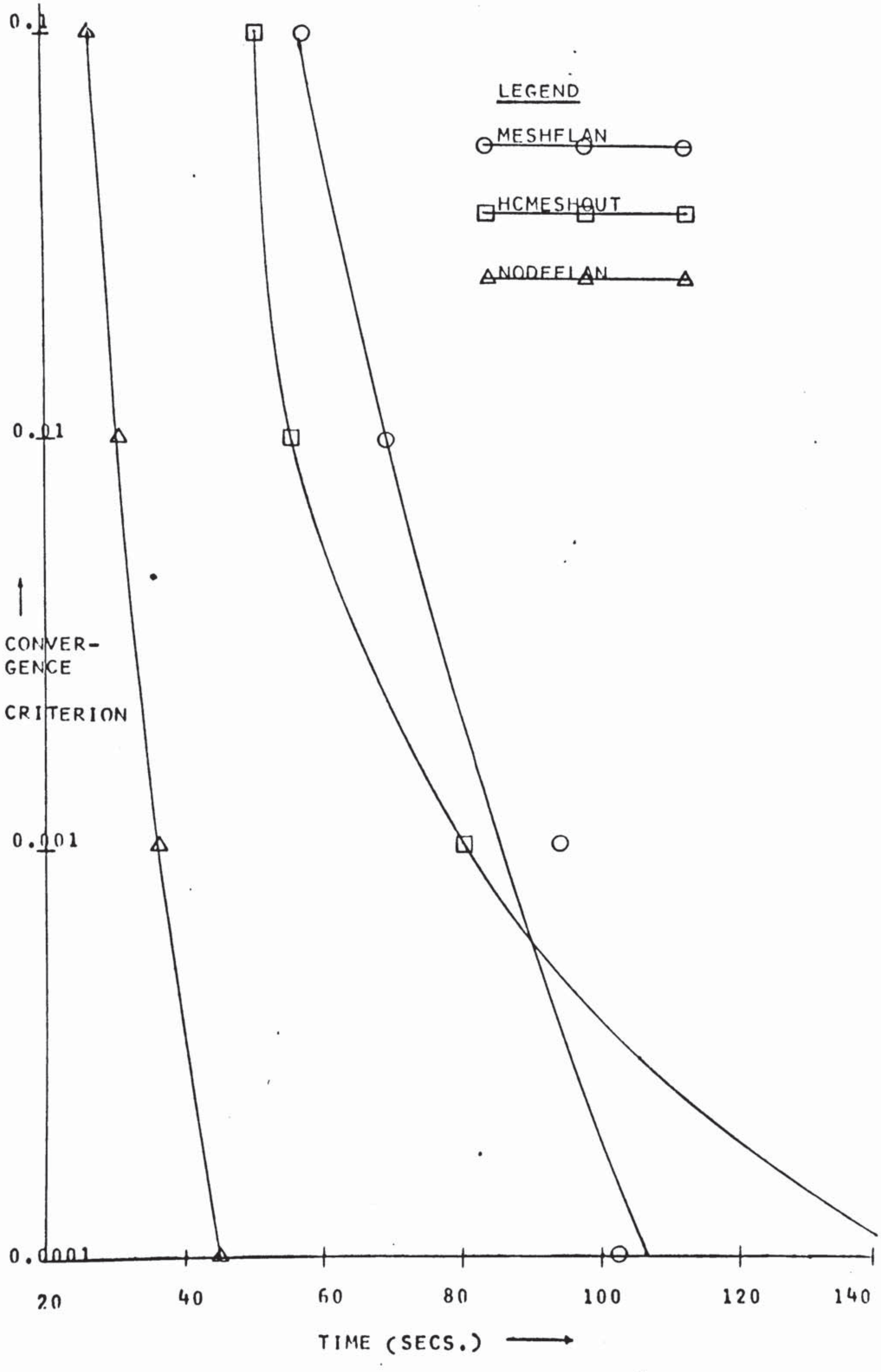
It has previously been stated that the time required for solution as indicated by the computer output must be treated with some caution. With this in mind the graph in figure (5.3.2) has been constructed to show only the general time dependence of the three programs on the choice of convergence criterion. No absolute time comparison can be evaluated, however the general conclusion, that the program MESHFLAN requires more computation time to converge than NODEFLAN, is formed and also that HCMESHOUT requires many more iterations at lower convergence criterion.

The final test of the stability of the matrix methods was achieved by changing the node to datum supplies at a fixed value of the initial estimate of flow and at a fixed value of convergence criterion. The maximum overlap case of TEST1 was used and the node to datum flows taken as those in table (2) times some constant. The results obtained



The effect of convergence criteria on the number of iterations required for solution

Figure (5.3.2) The effect of convergence criteria on the time required for solution



for the analysis are shown in table (31), which shows that at high flow rates convergence is marginally slower because the initial guess of flow is small compared to calculated branch flow at solution, as discussed previously, and that at low flow rates convergence is markedly slower because of the difficulty of maintaining accurate friction factors from iteration to iteration at low Reynolds Numbers. In the intermediate range little or no effect on convergence is apparent.

5.3.3 Specifying nodal pressures

The analysis of the minimum overlap case of TEST1 by the program NODEFLAN gave the results for the node to datum pressures shown in table (32). The information contained in this table was then used to verify the performance of the program which solves network problems that may have some of their node to datum pressures specified, called NPSMESHFLAN. From a knowledge of the converged node to datum pressure at node (8) of TEST1 the input data to NPSMESHFLAN was arranged so that the node to datum flow at node (8) was initially specified as zero, and the node to datum pressure at node (8) specified at the same pressure as the resultant node to datum pressure calculated by NODEFLAN for the same node.

According to the theory previously discussed the successfully converged program NPSMESHFLAN ought to give the same individual branch flows for the above situation as the node to datum flow fully specified case. Table (33) shows the results obtained by NPSMESHFLAN for the above network problem and the results obtained from NODEFLAN for the flow fully specified case. As can be seen, the program NPSMESHFLAN converged with very similar individual branch flows as the solution to the flow fully specified analysis of NODEFLAN. Further verification of the performance of NPSMESHFLAN was attempted using the results obtained by NODEFLAN for the flow fully specified case of maximum overlap of the network due to Knights and Allen. Table (34) shows the node to datum pressures obtained by NODEFLAN for the above analysis. Using the information contained in this table, two analyses were attempted. The first with flow specified as zero at node (9) and a pressure of 65,308.4 specified at the same node, and the extension of this case with the flow also specified as zero at node (11) but the pressure at that node specified as 16,906.4. The results obtained in terms of the individual pipe flows for both of these analyses are compared to the results obtained by NODEFLAN for the flow fully specified case.

As can be seen from table (35), the results for the two analyses of the Knights and Allen network fully support the proposition that NPSMESHFLAN is an excellent program for analysing the realistic pressure and flow specified problem. In all three analyses NPSMESHFLAN required one further iteration than the flow fully specified analysis of MESHFLAN; the reason for this is that on the first iteration the individual branch flows are not calculated as accurately as in the flow fully specified case. The fact that the program requires in general only one further iteration is not thought to be a computational drawback.

5.4 The Diakoptic Methods

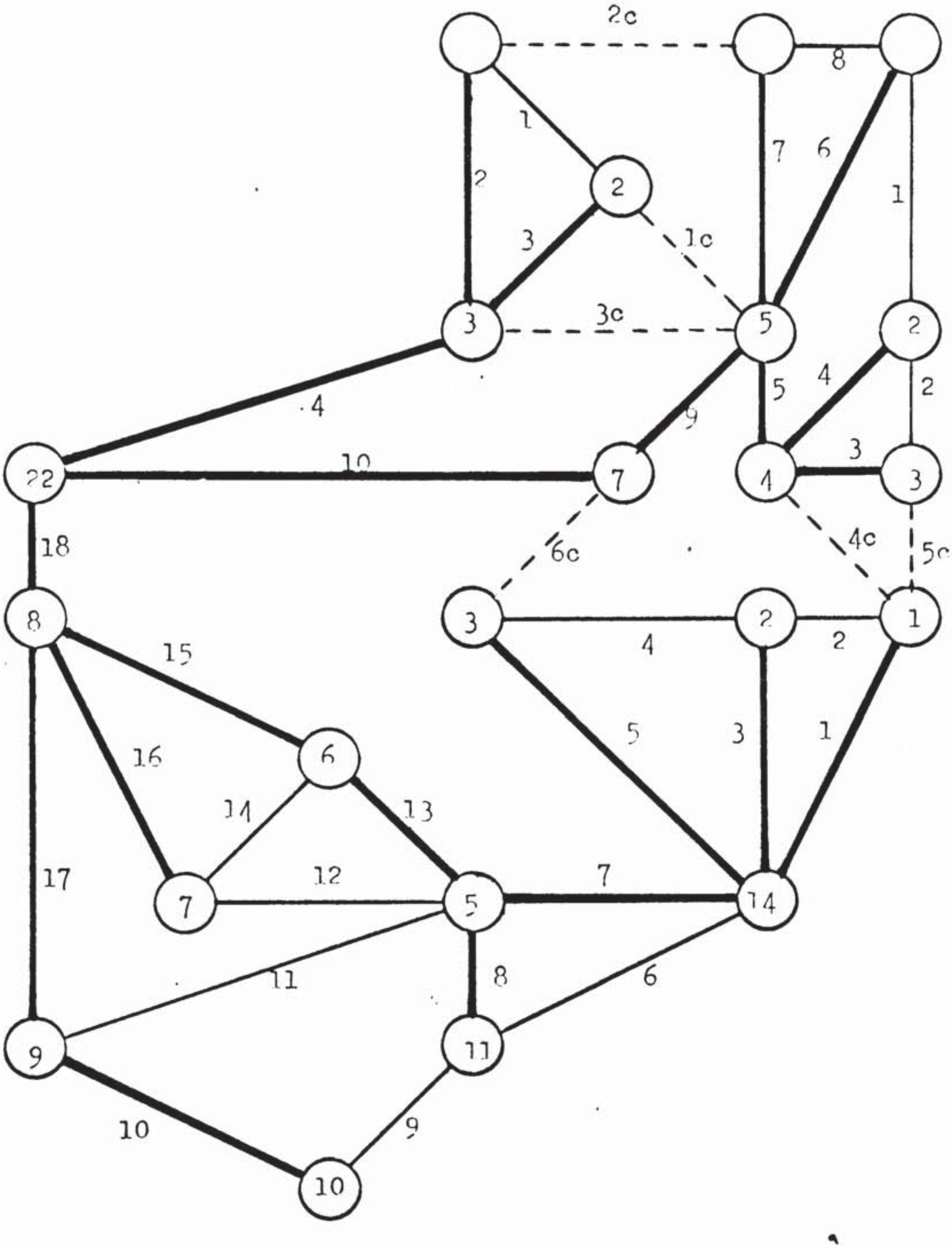
A small scale analysis of the convergence of the diakoptics programs under a variety of conditions has shown the methods involved to be stable. This agrees with the suggestion previously put forward that the stability of all the matrix methods could be established by the analysis of one of the simpler matrix programs. The analysis of the performance of NODEFLAN has shown conclusively over a wide range of conditions that the matrix methods are stable. Middleton (6) has also given evidence of the stability of the nodal diakoptic method and therefore no further investigation of the stability of the diakoptics methods in general is necessary.

The following discussion is therefore concerned only with the relative accuracy of the results produced by the two diakoptics programs, the saving in storage and the speed of convergence.

To verify the results obtained by NODEDIAK the TEST1 network was torn into three sub networks as shown in figure (5.4.1). The connection list and node to datum flows for each of the cutsections are given in tables (36) and (37) respectively. The results in terms of the individual pipe flows and the comparison with the results of Middleton are given in table (38). As can be seen from this table, the results from NODEDIAK are in good agreement with those of Middleton save for the flow in branch 8 of cutsection 1. However further analysis of Middleton's results show that the flow recorded for that branch could not have been arrived at by the program because of the values of the flows in the surrounding branches. The flow should be 13.98, not 14.98 cu ft/min, and the error in Middleton's results is therefore probably typographical. No valuable information can be gained by analysing the small errors between the two sets of results since Middleton has not recorded his convergence criterion. Further evidence which shows the similarity between the results obtained by Middleton and those obtained from NODEDIAK is given in table (39). This table shows the node to datum pressures for the network TEST1 shown in figure (5.4.1), together with the equivalent results from a different cutting pattern analysed by Middleton. As can be seen there is excellent agreement between the two sets of results.

The network due to Knights and Allen as stated previously has been examined as a water distribution network in this thesis; however to

Figure (5.4.1) The network TEST1 subdivided into three sub networks by six cutbranches for analysis by NODEDIAK



compare the results of the program NODEDIAK with the results obtained by Middleton (6) and Knights and Allen (35) the network was analysed once only as a gas distribution system. The values of the gas density and viscosity used are the same as those used by Middleton, the information being obtained from Perry (43). The connection list for the network shown in figure (5.4.2) is given in table (40) and the node to datum flows for each section in table (41). The results in terms of the individual branch flows are given in table (42). All three sets of results are in good agreement, larger errors only occuring in branches of low flow. The agreement is better than that reported by Ingels and Powers (29) who experienced a 20-30 per cent difference between their own results and those of Knights and Allen.

The small network due to Dolan (32) has been analysed by Ingels and Powers (29) and Middleton (6). The connection list for the analysis of this network by NODEDIAK is given in table (43) and the dimensions of the network for the figure (5.4.3) are shown in table (42) and the node to datum flows for the same figure in table (45). The results obtained by four analyses are shown in table (46). The maximum overlap choice using NODEFLAN is in good agreement with the results obtained by NODEDIAK and there in turn are in good agreement with those obtained by Middleton. However the results obtained by Ingels and Powers do not completely agree with those of the three other analyses. Most of the disagreement arises in branches of low flow, of which there are many in this small network. The fact that so many low flow branches are present is difficult to understand since the original analysis by Dolan was for a fire-fighting system, for which the demands on the system are of at least an order too small.

The program NODEDIAK has been shown to give results which are in good agreement both with those available in the literature and with

Figure (5.4.2) The network due to Knights and Allen subdivided into four cutsections by nine cutbranches for analysis by NODEDIAK

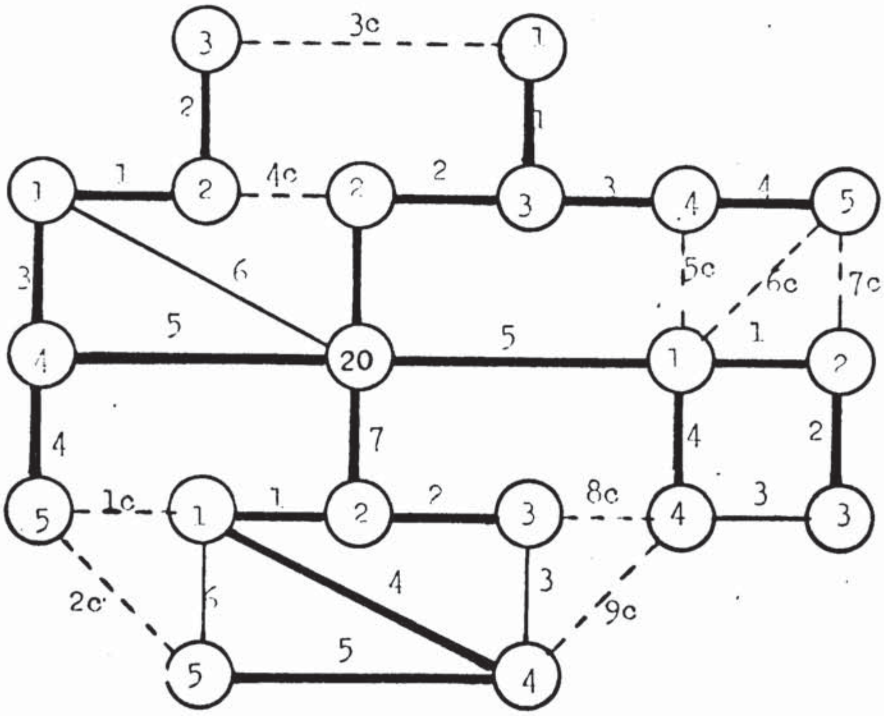
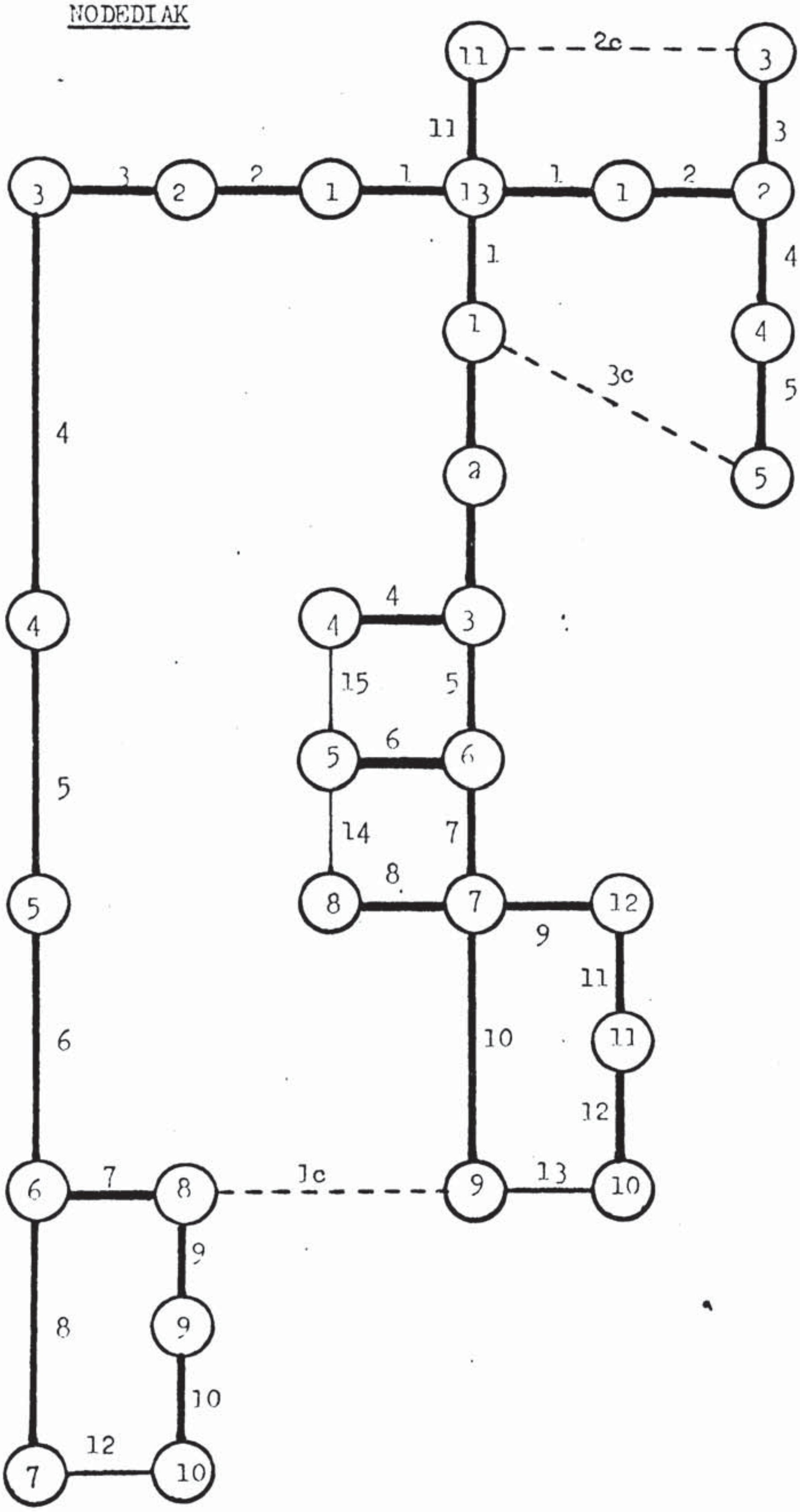


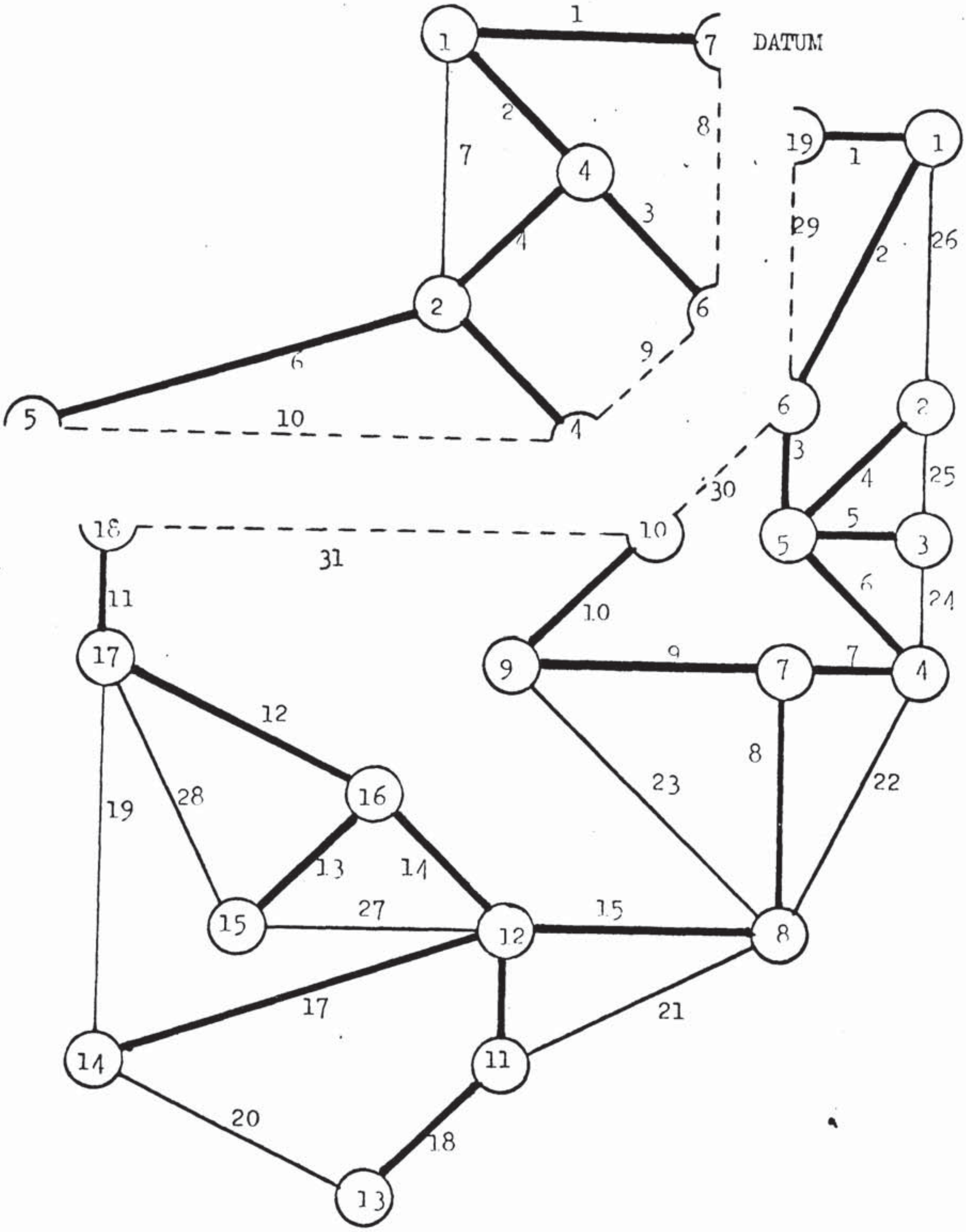
Figure (5.4.3) The network due to Dolan subdivided into three cutsections by three cutbranches for analysis by NODEDIAK



the results obtained from the programs HCMESHOUT, NODEFLAN and MESHFLAN. No comparison of the relative speed of convergence or the storage requirement of NODEDIAK has been evaluated at this stage. It is felt that the networks analysed in this chapter are not large enough to show clearly the advantages of using the diakoptics methods. The reason for this is that the computation time given as output from the programs includes the compilation time and it is known approximately that the compilation time for NODEDIAK is in the order of 30 seconds and that for NODEFLAN in the order of 10 seconds. Therefore any analysis of efficiency of the programs which converge to solutions in the order of 30-50 seconds for the above networks is likely to be inconclusive. Two further networks have been examined, one is a water/gas distribution network by W S Atkins called TEST3, and the other called TEST2 has been constructed by the author to represent the type of network problem encountered in heat transfer studies.

The following discussion is mainly concerned with the performance of NODEDIAK and MESHDIAK in the analysis of these two networks. However since no comparison can be made, for these two networks, with published information, it is first necessary to establish the accuracy of the results of MESHDIAK. The network TEST1 shown in a subdivided state in figure (5.4.4) was analysed by MESHDIAK. The connection lists and node to datum flow vectors for the cutsections are given in tables (47) and (48) respectively. The results obtained by MESHDIAK are compared in table (49) with the results obtained by MESHFLAN for the maximum overlap analysis of TEST1. These latter results have previously been shown in table (22) to be true and accurate results. As can be seen from table (49), the results produced by MESHDIAK are in good agreement with those produced by MESHFLAN. This is especially true in the more critical low flow branches, for instance cutbranch 3.

Figure (5.4.4) The network TESTl subdivided for analysis by MESHDIAK



5.4.1 The analysis of larger networks

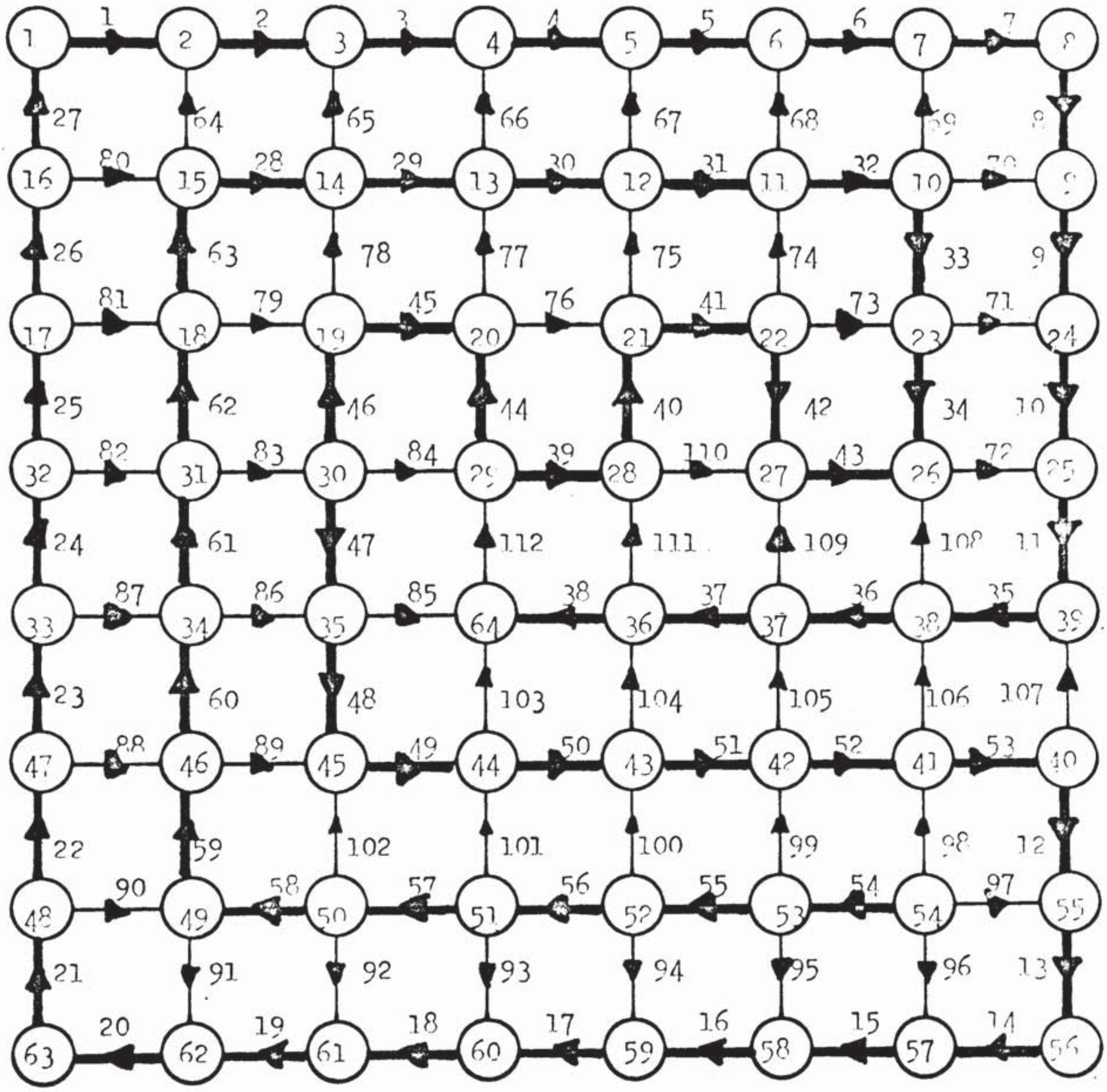
To establish the performance in terms of solution time and storage requirements for the diakoptics methods, the network TEST2 shown in figure (5.4.5) was devised.

This network was analysed using HCMESHOUT, NODEFLAN and MESHFLAN for the arbitrary overlap case shown. For the analysis the network was assumed to be a water distribution system of 112 identical smooth pipes of length 100 ft and 0.5 ft in diameter with no pumps in the system. It is thought that the preparation of separate tables to indicate the connection list of the three cases of TEST2 analyses would be bulky, therefore the directions of the branches for all the cases are shown on the figures. A single table (50) shows the node to datum flows for the arbitrary overlap case shown in figure (5.4.5), the node to datum flows for the other two analyses of MESHDIAK and NODEDIAK are identical, and therefore the vector of node to datum flows for these cases may be obtained by cross referencing from the figures (5.4.6), (5.4.7) and table (50).

The program HCMESHOUT analysing the arbitrary overlap case did not converge to a solution and the iteration scheme was stopped after 2000 seconds. Approximately 350 iterations had been completed and the information about the number of branches assumed to have converged indicated that a solution was converging as shown below.

Number of iterations	Number of branches converged
60	2
120	8
180	18
240	62
300	89
350	95
Stopped	

Figure (5.4.5) The Arbitrary Overlap Condition of TEST2



For the major part of the time of this research project less than 20K of core storage was available for data handling; under this condition the programs NODEFLAN and MESHFLAN could not solve the arbitrary overlap case of TEST2 because they required too much storage space. However, when more core store became available the programs were re-run with the same data. Sufficient storage was available but neither of the programs had converged to a solution after 550 seconds, when the iteration cycle was interrupted by the University's computer operating system. From the information about the number of branches converged at each iteration both programs were converging to a solution, but slowly, only two branches being converged by NODEFLAN and one branch being converged by MESHFLAN.

One of the most important factors of the NODEDIAK and MESHDIAK analyses is the fact that they both converged to a successful solution using less than 20K of storage and therefore these two programs provided the only way in which a network problem of this size could be analysed on the smaller machine. A further decisive factor is that NODEDIAK converged to a solution in 369 seconds and MESHDIAK in 310 seconds, proving beyond all doubt the efficiency of these methods. The results obtained by both programs are given in table (51). The agreement between the two sets of results is excellent both in the low flow branches and in the critical cutbranches.

A second large network has also been examined. This network called TEST3 is shown in figure (5.4.8). The branches and nodes are numbered with the tree branches indicated by the heavier lines. The branches are assumed to be identical, 100 ft in length, 0.5 ft in diameter and smooth. There are no pumps in the system. The node to datum flow vector for this network is shown in table (52). This overall system was analysed using MESHFLAN and NODEFLAN and the results in

Figure (5.4.6) The network TEST2 subdivided into two sub networks by seven cutbranches for analysis by MESHDIAK

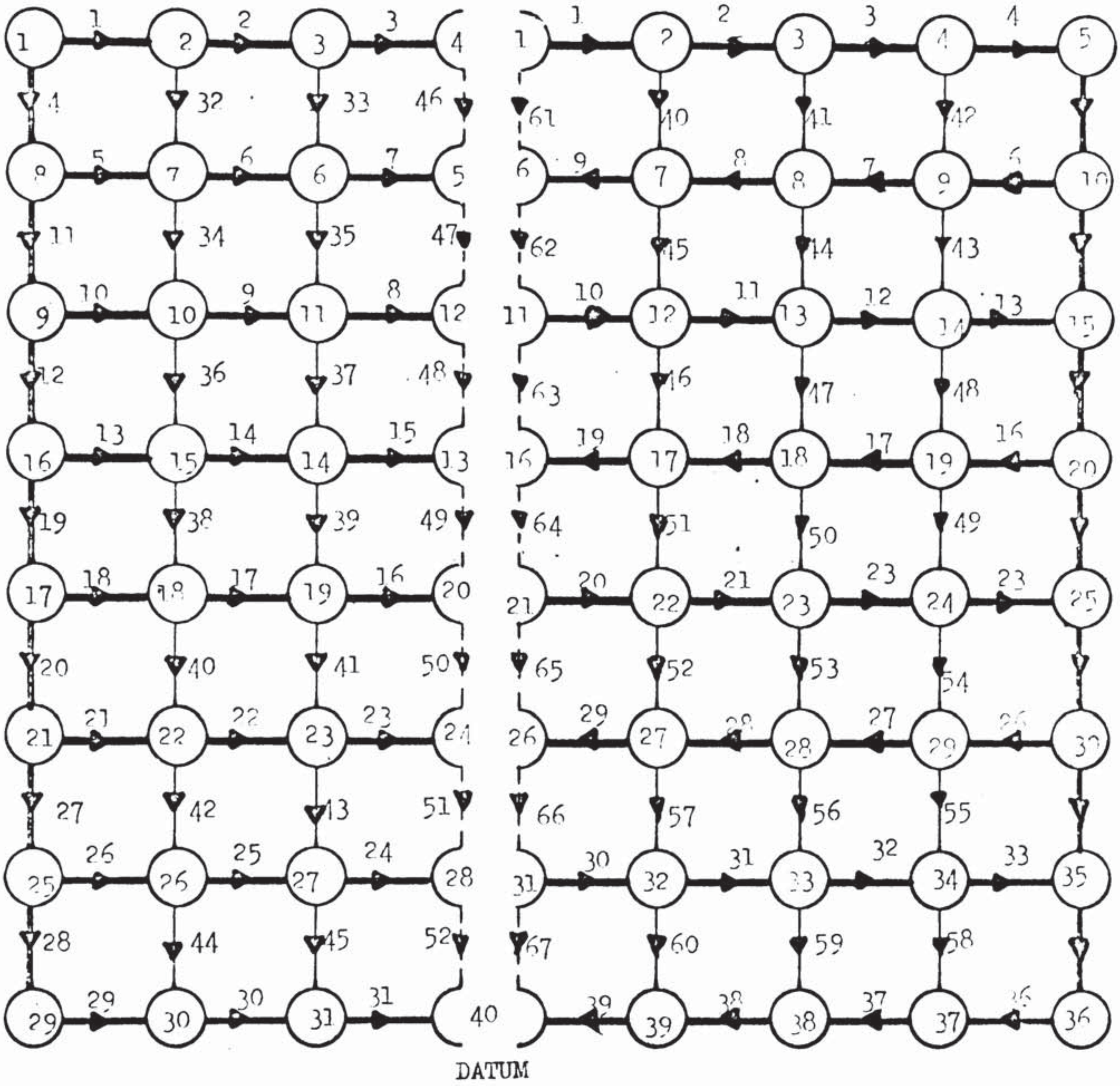


Figure (5.4.7) The network TEST2 subdivided into two sub networks by seven cutbranches for analysis by NODEDIAK

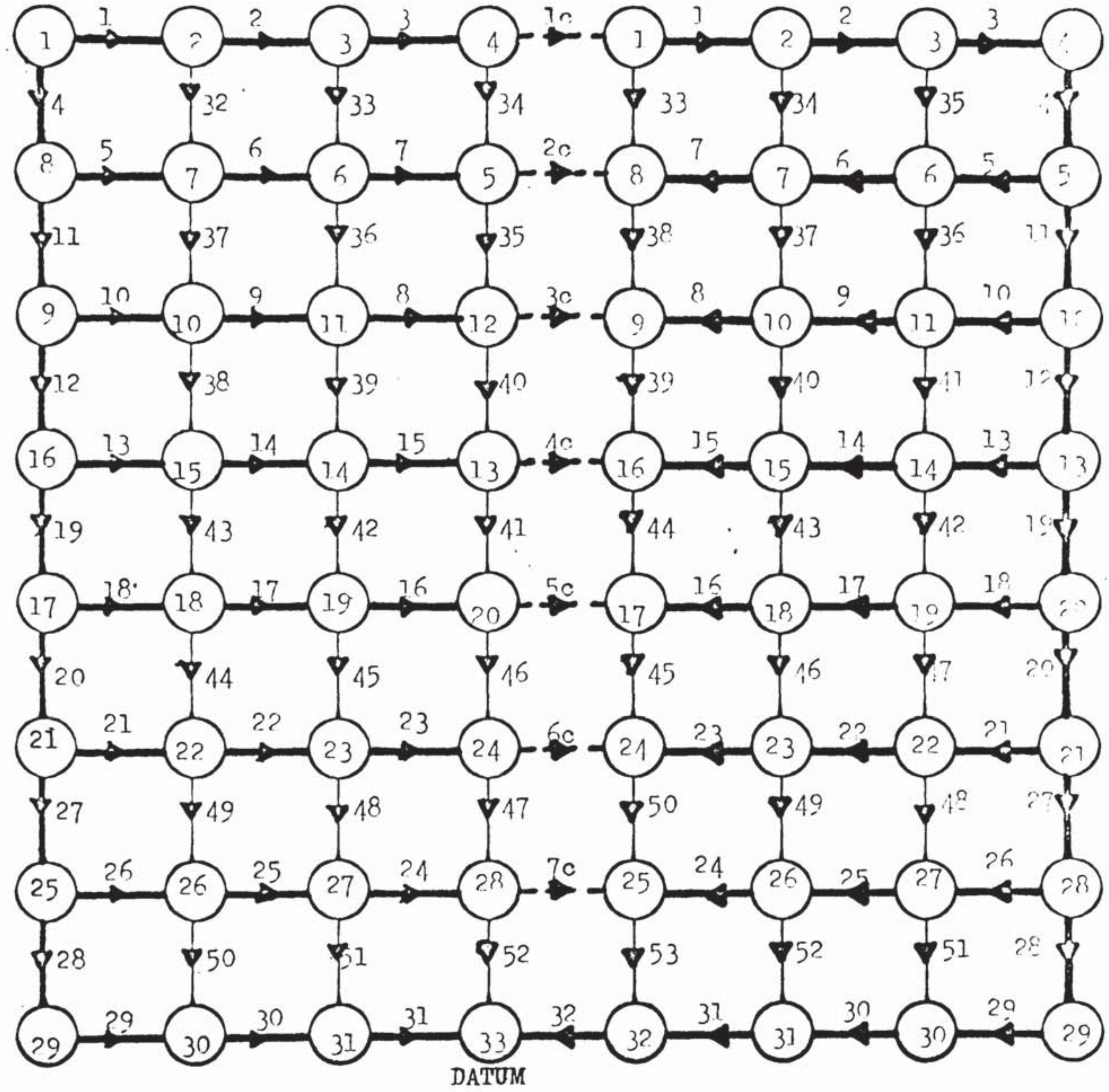

1 1 2 2 3 3 4 1c 1 1 2 2 3 3 4
4 32 33 34 33 34 35 4
8 5 7 6 6 7 5 2c 8 7 7 6 6 5 5
11 37 36 35 38 37 36 11
9 10 10 9 11 8 12 3c 9 8 10 9 11 10 12
12 38 39 40 39 40 41 12
16 13 15 14 14 15 13 4c 16 15 15 14 14 13 13
19 43 42 41 44 43 42 19
17 18 18 17 19 16 20 5c 17 16 18 17 19 18 20
20 44 45 46 45 46 47 20
21 21 22 22 23 23 24 6c 24 23 23 22 22 21 21
27 49 48 47 50 49 48 27
25 26 26 25 27 24 28 7c 25 24 26 25 27 26 28
28 50 51 52 53 52 51 28
29 29 30 30 31 31 33 32 32 31 31 30 30 29 29
DATUM

Figure (5.4.8) The arbitrary overlap case of TEST3

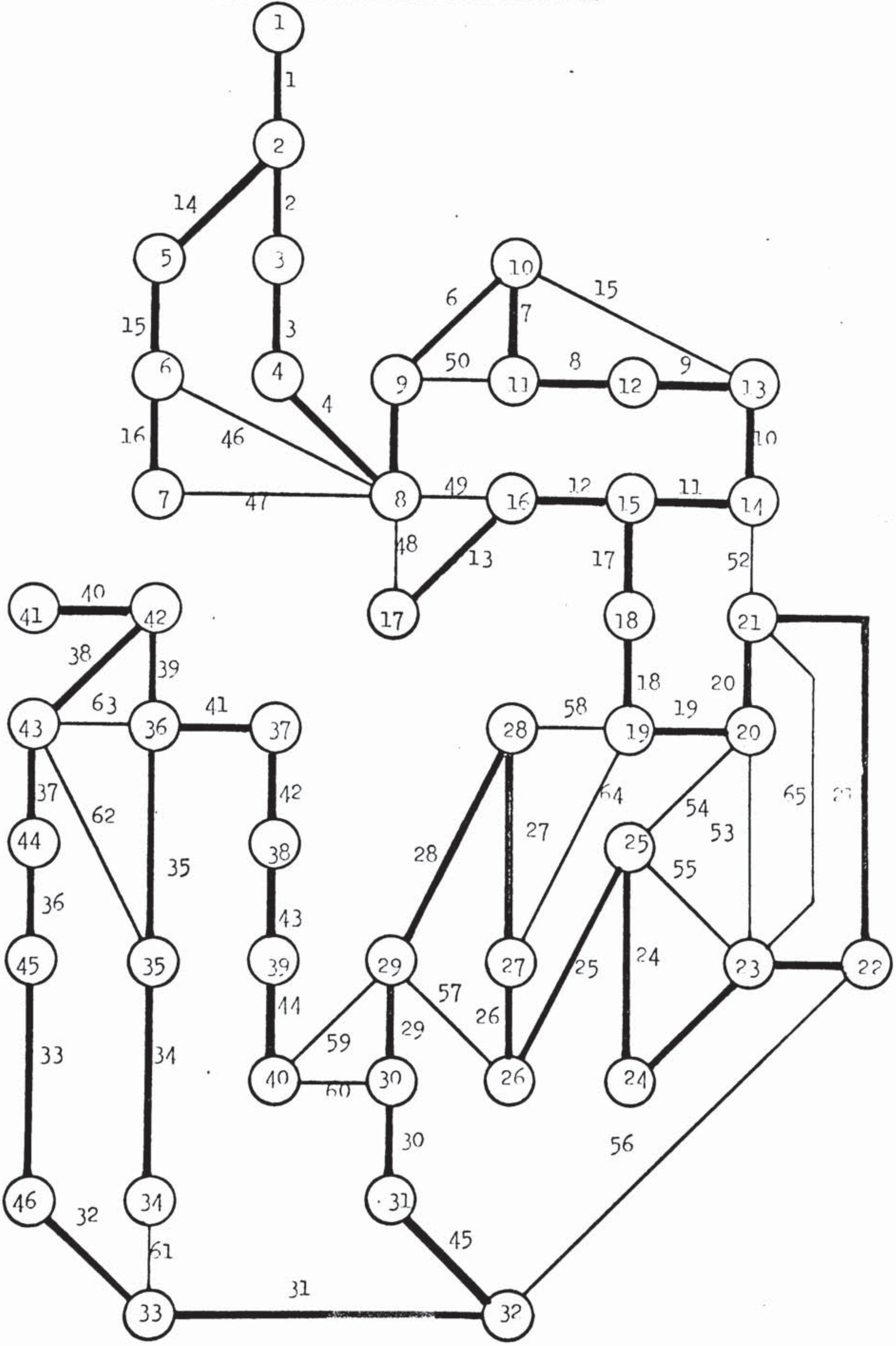


Figure (5.4.9) The network TEST3 subdivided for analysis by NODAL

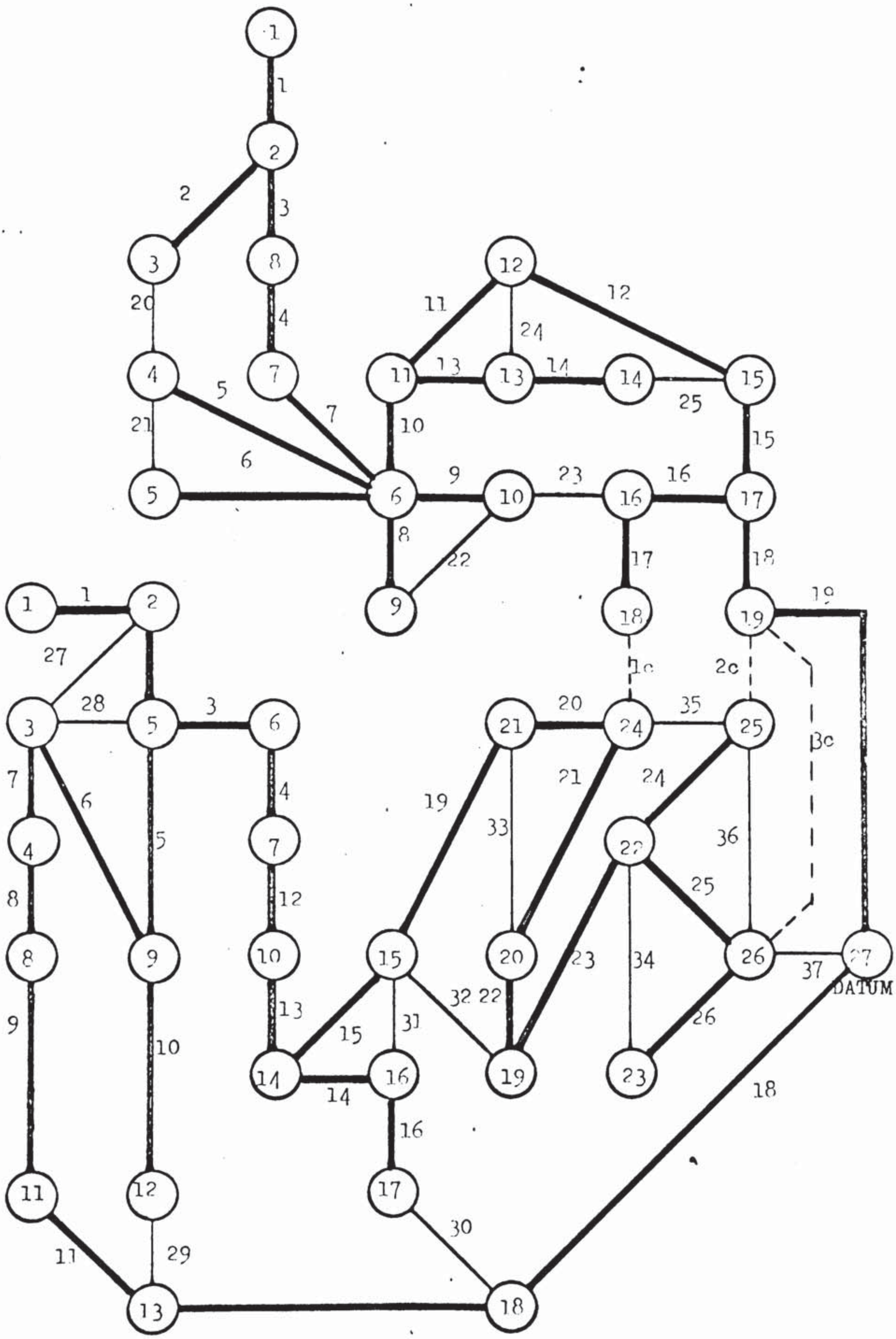
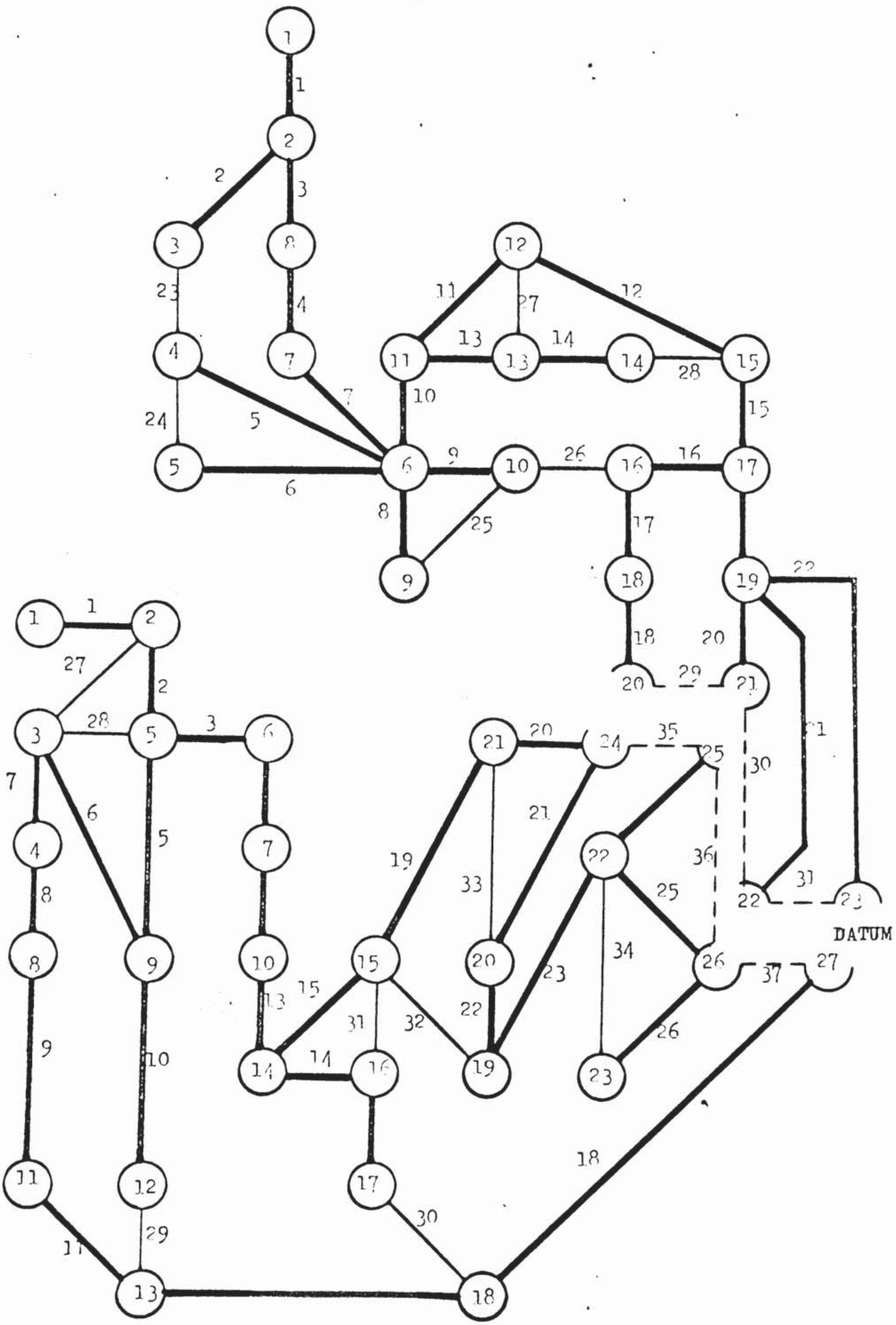


Figure (5.4.10) The network TEST3 subdivided for analysis by MESHDIAK



terms of the individual pipe flows given in table (53).

The network was also subdivided as shown in figures (5.4.9) and (5.4.10) and analysed using NODEDIAK and MESHDIAK respectively. The relative accuracy of the results produced by these two programs has previously been established and interest in the results obtained from the analysis of TEST3 centres only around the relative solution times and number of iterations required for solution. The following results were obtained from the four analyses of TEST3.

Program	Solution Time secs	Iterations
MESHFLAN	134	11
NODEFLAN	163	12
MESHDIAK	94	11
NODEDIAK	130	12

TEST3 has a large number of branches, but is not over meshed. This is borne out in the above results since the program MESHFLAN is more efficient than NODEFLAN. However since the network is large the saving in solution time by the use of the diakoptics programs is also well demonstrated. Notice too how the program MESHDIAK is more efficient than NODEDIAK because of the few meshes in the network.

5.5 Inversion Procedures

As previously discussed, four procedures for the inversion of the network solution matrices were investigated. Using NODEFLAN to analyse an arbitrary overlap choice of TEST1 the following results were obtained:

Inversion Procedure	Solution Time secs
Gauss-Jordan	72
ICL Package	58
Choleski	43
Modified Choleski	31

Further analyses of the sort outlined above were performed with MESHFLAN with the same result that the modified Choleski procedure due to Caffrey (33) was found to be the inversion procedure which gave the most rapid convergence.

5.6 Discussion of Results

The results clearly show that there are advantages to be gained using the matrix methods for the computation of network problems. In the analysis of the smaller networks the programs NODEFLAN and MESHFLAN have been shown to be more efficient than the Hardy Cross methods. Each one of these programs is of course suited to a particular type of network problem. For a heavily meshed system NODEFLAN is clearly the most efficient program, whilst for networks which are not heavily meshed MESHFLAN is the most efficient program. In the intermediate range, there is little to choose between the two programs but it must be remembered that because of the nature of the solution method employed in MESHFLAN, the number of computational operations required by this program is greater than that required for NODEFLAN. The reason for this is that the $\underline{A}$ matrix normally associated with the node to datum method may be very simply represented within the computer. The $\underline{C}$ matrix of the mesh method has been stored as a full matrix and apart from the special algorithm proposed for forming the mesh solution matrix, any matrix multiplications involving $\underline{C}$ must therefore be carried out in full.

In the analysis of larger network systems the two diakoptics programs have been shown to be the most efficient. Clearly however the same problem arises with MESHDIAK as with MESHFLAN in that the $\underline{C}$ matrix for each of the sub networks is stored in its entirety, apart from the omission of the link partition part $\underline{C}_L$. No definitive analysis has been carried out to determine the relative efficiencies of NODEDIAK and MESHDIAK but it is felt that NODEDIAK is more suitable for the analysis of large heavily meshed networks and MESHDIAK suitable for the analysis of large networks which are not heavily meshed.

The question of the relative efficiencies of these two programs is of course related to the problem of choice of cutting pattern. Middleton (6) has proposed the rule of thumb for the most efficient cutting pattern associated with nodal diakoptics. It is that the network should be subdivided into the maximum number of sub networks of approximately equal numbers of nodes by the minimum number of cutbranches. A similar rule is suggested for mesh diakoptics. It is that the network should be subdivided into the maximum number of subnetworks of approximately equal numbers of meshes by the minimum number of cutbranches.

The data preparations are easier to compile for all of the matrix methods. An extension of this is that an engineer analysing a network system by adding or removing branches, or interconnecting series of networks can achieve this easily with no necessity to recompile new data preparations. Middleton (6) has shown this particular facility of the diakoptics method to be at least as important as the increased efficiency of the method. As discussed previously any commercial application of MESHDIAK or NODEDIAK would be expected to have incorporated into it the facility of calculating a solution to a changed analysis from an already converged solution.

The matrix methods have also been shown to be insensitive to the initial estimate of flow and friction factor with the results that these can be assigned by the program rather than input as data. It has further been shown that the recalculation of accurate friction factors on every iteration is not necessary, for turbulent flow the calculation of only a first order correction being necessary to maintain convergence to an accurate solution.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The results produced in Chapter (5) justify the assumption put forward in the introduction to this thesis, that the use of matrix methods applied to the solution of pipe network problems would result in more efficient means of solving those problems. It is the author's opinion however that the simplicity and conceptual clarity of the new methods is qualitatively of equal importance to the increase in efficiency.

Using four basic network describing matrices, six computer programs have been developed. LATTNODEFLAN, the program which incorporates the algorithm due to Branin (19) for the updating of a tree solution matrix by the addition of link branches, has been shown to be less efficient than even the Hardy Cross methods. The link at a time method suffers from the fact that the $(\breve{\underline{B}}_T.\underline{Z}_T.\underline{B}_T)$ matrix has to be constructed on every iteration by the full matrix multiplications, and since the completed matrix is not very sparse, no special techniques can be implemented for the lengthy updating procedure. Further, the amount by which each element is updated requires for its computation five subtractions, three additions, one multiplication and one division. LATTNODEFLAN may therefore be disregarded as an efficient method of solving network problems.

The remainder of the programs can be classed into two groups: the orthogonal methods and the diakoptic methods. Both groups of programs have been shown to be more efficient than the Hardy Cross methods. The data are easy to prepare and therefore the likelihood of error generation is decreased. The convergence of the methods is not dependent on the initial estimates of flow and friction factor, and because a flow distribution which satisfies Kirchoff's Laws does not have to be specified, the initial estimates may be generated automatically by the programs. The methods have been shown to be very stable over a large range of node to datum flow specifications and mesh formations. Friction factors are accurately calculated using a first order correction of the Colebrook White equation on every iteration.

The inclusion into a network specification of non pipe elements such as pump terms and node to datum pressures may also be accomplished with the minimum of effort, whilst for the Hardy Cross methods this is a difficult task.

The diakoptics programs have been shown to be the most efficient for the solution of large scale problems. As the size of the problem is increased the overall performance of the diakoptics programs, relative to the performance of other methods, is enhanced. No detailed investigation of any optimum cutting patterns has been made, however it is felt that the two sets of simple rules for subdividing networks will ensure efficient convergence.

The data preparation technique for all of the matrix methods has been refined, with the result that to set up the complete solution equations, only information about the physical data and a branch-node connection list is required. Since none of the matrix methods is dependent on the choice of mesh formation, the branches of the graph of a network may be arbitrarily numbered from 1 to BR, where BR is the number of branches in the system. This is the case even for programs which use the $\underline{C}$ matrix since this matrix can be automatically generated by the program. However, it has been shown that fewer operations are required to compute the $\underline{B}_T$ and $\underline{C}_T$ matrices if the tree branches are assigned the lowest reference numbers, and that in this case the mesh formation of $\underline{C}_T$ is identical to that chosen by the user. A refined technique of generating $\underline{B}_T$ from a connection list has also been proposed.

Partitioning of the connection list has the further advantage that less computer storage is required since $\underline{B}_L$ and $\underline{C}_L$ are respectively null and unit matrices, neither of which have to be assigned storage.

The partitioned form of all the network describing matrices gives the two square transformation matrices simple characteristics, from which it is quite easy to see the equivalence of partitioning the describing matrices and partitioning of the network itself. The partitioning of the network led to the important concept of the node to datum path which is the dual of the mesh path. Without the node to datum path concept, the two mesh equations of orthogonal and diakoptics solutions could not have been applied to pipe network flows. The two path concept has simplified the transformation between two types of network to such an extent that only the simplest of matrices is needed to perform the transformation. It is important to node that the graph describing matrices can be related directly to physical phenomena in the actual network.

The transformation matrices can be constructed directly as in the orthogonal methods, or by a series of matrix multiplications as in the diakoptics methods. However, algorithms have been suggested for the formation of the diakoptic solution matrices which, from a knowledge of their expected form, enable them to be set up with the minimum of effort. Similarly, rules have been formulated for the direct formation of the mesh and node to datum solution matrices $(\underline{\breve{A}}.\underline{Y}.\underline{A})$ and $(\underline{\breve{C}}.\underline{Z}.\underline{C})$, greatly reducing the normal computation time required to establish these matrices. An inversion procedure which takes advantage of the special properties of the solution matrices has been proposed, together with a technique for further reducing computer storage requirements by reducing the diagonal matrix of primitive admittances to a vector.

It was realised in the early development of the theory that the concepts of node to datum and mesh paths have a much wider application to chemical engineering problems. For instance, the concepts can be applied to the network of finite difference approximations of partial differential equations and therefore the solution equations to such problems can be automatically set up with a minimum of effort. In this particular case the solution matrices of the subdivided system can be made, in general, equivalent and therefore large amounts of computer storage and time can be saved as only one of these matrices has to be stored and inverted.

Systems of mixed linear and non-linear elements may also be solved more easily by using diakoptics, allowing the solution equations to cycle through the non-linear iteration only.

6.2 Recommendations for further Investigation

It is a feature of modern chemical engineering that designers use Computer Aided Design principles in the construction of chemical plant. It is suggested that the techniques of transformation matrices can be applied to the design problem. The components of a chemical plant can be represented as a collection of unique parts, each associated with a describing equation. Such a system can be visualised as equivalent to the collection of primitive branches of a network. These primitive branches may then be interconnected in any manner desired using the transformations that have previously been discussed.

One of the further advantages of using the diakoptics methods which has not been analysed in this thesis is the efficiency of the 'on-line' method of operation. A designer can effect changes to the shape of a network from an already converged solution, greatly reducing the amount of computation time required, since a new network formulation does not have to be carried out, although for the Hardy Cross methods this would be a necessity. Changes in node to datum flows may also be facilitated easily since no new flow distribution has to be specified. Solutions to a series of individual network problems may be obtained and stored, with the result that the designer can at any time interconnect any of the series in any manner with the minimum of effort. This is a particularly important facility for dealing with network systems in which the boundary conditions are other than physical, for example the interconnection of systems belonging to different area boards in the gas industry.

It is strongly recommended therefore that any commercial application of the two diakoptics programs should incorporate some interactive facility.

Although it has been shown that certain of the matrix methods are more efficient than the remainder when analysing particular problems, for instance for heavily meshed systems the node to datum methods are most efficient, much work remains to be done on improving the performance of the programs. Sparsity and reduced storage techniques can be investigated. The full potential of the mesh diakoptics method must also be further investigated and compared to the fully developed node to datum diakoptics method.

List of Symbols

Transformation Matrices

$\underline{A}$	Branch-node incidence matrix
$\underline{B}$	Node to datum path matrix
$\underline{C}$	Branch-mesh incidence matrix
$\underline{D}$	Cutset matrix
$\underline{F}$	Link-branch matrix
$\underline{\gamma}$	Square orthogonal transformation matrix
$\underline{\alpha}$	Square orthogonal transformation matrix

Supplementary Matrix Notation

$\underline{A}_T$ etc	Node to datum partition of $\underline{A}$ etc
$\underline{A}_L$ etc	Mesh partition of $\underline{A}$ etc
$\underline{\gamma}_{TT}$	Node to datum - node to datum partition of $\underline{\gamma}$
$\underline{\alpha}_{LT}$	Mesh - node to datum partition of $\underline{\alpha}$
$\underline{A}^a$	Augmented branch-node incidence matrix
$\underline{A}^*$ etc	Branch-node incidence matrix for torn network
$\underline{\gamma}^*$ etc	Square orthogonal transformation matrix relating torn and primitive systems etc
$\underline{\gamma}^{**}$ etc	Square orthogonal transformation matrix relating torn and connected networks
a (i,j) etc	Element (i,j) of $\underline{A}$ matrix etc

General Matrix Notation

$\underline{U}$	Identity matrix
$\underline{O}$	Null matrix
$\underline{\breve{A}}$	Transpose of matrix
$\underline{A}^{-1}$	Inverse of matrix
$\underline{A}.\underline{C}$	Matrix multiplication

Vectors and Matrices of Network Variables

$\underline{E}$	Primitive non pipe pressure term usually associated with pumps
$\underline{e}$	Primitive branch potential rise across extremities of branch
$\underline{J}$	Primitive total flow in branch
$\underline{I}$	Primitive branch current source
$\underline{i}$	Primitive branch component of current due to mesh flows
$\underline{V}$	Total potential rise in primitive branch
$\underline{Y}$	Primitive branch admittance
$\underline{Z}$	Primitive branch impedance
$\underline{E}'$	Orthogonal potential source
$\underline{e}'$	Orthogonal node to datum potential
$\underline{I}'$	Orthogonal current source usually node to datum flows
$\underline{i}'$	Orthogonal mesh component of flow
$\underline{J}c$, $\underline{V}c$	Vectors associated with connected network
I_1, I_2 etc	Components of vector $\underline{I}$ etc
$\underline{E}^*$	Voltage source which incorporates equivalent current sources
$\underline{I}^*$	Current source which incorporates equivalent voltage sources

Physical Network Variables

A	Cross sectional area of pipe
D	Diameter of pipe
L	Length of pipe
P	Fluid pressure
Q	Fluid flow in pipe

R	Hydraulic resistance
R'	Resistance term due to Hardy Cross
U	Fluid velocity
Re	Reynolds Number

Other Notation

b, BR	Number of branches in network
m, M	Number of meshes in network
ND	Number of non datum nodes or node to datum paths in network
r_i	Van der Berg flow residue at node i
n	Hardy Cross Exponent

Greek Symbols

ε	Pipe roughness
ρ	Fluid density
ϕ	Friction factor
μ	Fluid viscosity
ξ	Correction term due to Hardy Cross
η	Correction term due to Van der Berg
θ	Variable used by Ingels and Powers in friction factor determination

References

1. FINCHAM A E, 1970 August, The Gas Council Research Station Technical Report LRST 49

2. TRAVERS K, 1967, The Gas Journal, November, page 167

3. DANIEL P T, 1966, Trans I Chem E, 44 page T77

4. KRON G, 1939, "Tensor Analysis of Networks", Wiley NY

5. KRON G, 1963, "Diakoptics", Macdonald London

6. MIDDLETON P, 1968, Ph D Thesis, The University of Aston in Birmingham

7. KRON G, 1944, ASME TJ App Mech 11, page A149-16

8. KRON G, 1956, J Aero Sci 23, page 557

9. KRON G, 1945, J Aero Sci 12, page 221

10. KRON G, 1945, J App Phys 16, page 172

11. KRON G, 1959, "Tensor Analysis for Circuits", Dover Publications NY

12. Le CORBEILLER P, 1950, "Matrix Analysis of Electric Networks", Wiley NY

13. ROTH J P, 1959, Quart App Math 17, No 1, April, page 1

14. ROTH J P, 1955, Proc NAS 41, page 518

15. BRANIN F H, 1967, Proc IEEE 55, No 11, November, page 1787

16. BRANIN F H, 1956, Proc 2nd Midwest Symp Circ Theory, Mich State Univ, December

17. CROSS H, 1936, "Analysis of flow in networks of conduit or conduction", Bul 286, Univ Illinois Exp Sta Urbana Ill

18. OLSZTYN J T, THEODOROFF T J, 1959, GMR DYANA Program, General Motors Research Lab Warren Mich

19. BRANIN F H, 1962, IBM Tech Rep TR 00 85, Data Systems Div Poughkeepsie NY, March

20. BRAMELLER A, 1971, IGE Journal, March, page 188

21. BRANIN F H, 1967, Electronics 40, January, page 88

22. SATO N, TINNEY W F, 1963, IEEE Pow App 82, December, page 944

23. MALMBERG A F, CORNWELL F L, HOFER F N, 1964, Los Alamos Scientific Lab, Los Alamos, N Mex Rep LA 3119

24. HAPP H H, 1971, "Diakoptics and Networks", Academic Press NY

25. VEBLEN O, 1931, Analysis Situs Amer Math Soc, Providence, Rhode Island

26. VAN DER BERG, 1963, De Ingenieur JRG 75/NR5D/13-11-63

27. KNIEBES D V, WILSON G G, 1960, Chem Eng Prog Symp 56, No 31, September, page 49

28. HUNN R J, RALPH J I A, I Chem E Computer Program No 10

29. INGELS D M, POWERS J E, 1964, Chem Eng Prog 60, No 2, February, page 65

30. INGELS D M, 1962, M Ch E Thesis, Univ of Oklahoma

31. MOODY L F, 1944, Trans ASME 66, page 671

32. DOLAN J J, 1936, Eng-News Record 117, page 475

33. CAFFREY J, 1961, Comm ACM, July

34. RANDELL B, BROYDEN C G, 1961, Comm ACM, July

35. KNIGHTS I A, ALLEN J W, 1964, Midland Junior Gas Association - 60th Session

36. DREW T B, GENERAUX R P, 1936, 32 No 2, page 17

37. FINCHAM A E, 1971, IGE 37th Autumn Research Meeting London, November

38. BRAMELLER A, Unpublished Work

39. BRAMELLER A, SCOTT M R, JOHN M N, 1969, "Practical Diakoptics for Electrical Networks", Chapman and Hall London

40. ABBOT J M, 1971, "Computer Users Handbook", The University of Aston in Birmingham, November

41. BIXBY M, Private Communication

42. FINCHAM A E, 1971, The Gas Council Tech Report LRST 80, May

43. PERRY J H, 4th Ed Chem Eng Handbook, page 2-8

44. SASSON A M, JAIMES F J, 1967, Proc IEEE Pow App PA86, page 860

45. MIDDLETON P, GAY B, 1971, Chem Eng Sci 26, page 109

46. COLEBROOK C F, 1939, J Inst Civ Eng 11, page 133

47. PASE H F, 1963, "Piping Design for Process Plants", NY